Machine Learning Methods for Product Quality Monitoring in Electric Resistance Welding

Zur Erlangung des akademischen Grades eines
DOKTORS DER INGENIEURWISSENSCHAFTEN (Dr. -Ing.)

von der KIT-Fakultät für Maschinenbau des
Karlsruher Instituts für Technologie (KIT)

angenommene
DISSERTATION
von

M.Sc. Baifan Zhou

Tag der mündlichen Prüfung:	26 April 2021
Hauptreferent:	apl. Prof. Dr.-Ing. Ralf Mikut
Korreferent:	Prof. Dr.-Ing. habil. Volker Schulze

Kurzfassung

Elektrisches Widerstandsschweißen (Englisch: Electric Resistance Welding, ERW) ist eine Gruppe von vollautomatisierten Fertigungsprozessen, bei denen metallische Werkstoffe durch Wärme verbunden werden, die von elektrischem Strom und Widerstand erzeugt wird. Eine genaue Qualitätsüberwachung von ERW kann oft nur teilweise mit destruktiven Methoden durchgeführt werden. Es besteht ein großes industrielles und wirtschaftliches Potenzial, datengetriebene Ansätze für die Qualitätsüberwachung in ERW zu entwickeln, um die Wartungskosten zu senken und die Qualitätskontrolle zu verbessern. Datengetriebene Ansätze wie maschinelles Lernen (ML) haben aufgrund der enormen Menge verfügbarer Daten, die von Technologien der Industrie 4.0 bereitgestellt werden, viel Aufmerksamkeit auf sich gezogen. Datengetriebene Ansätze ermöglichen eine zerstörungsfreie, umfassende und präzise Qualitätsüberwachung, wenn eine bestimmte Menge präziser Daten verfügbar ist. Dies kann eine umfassende Online-Qualitätsüberwachung ermöglichen, die ansonsten mit herkömmlichen empirischen Methoden äußerst schwierig ist.

Es gibt jedoch noch viele Herausforderungen bei der Adoption solcher Ansätze in der Fertigungsindustrie. Zu diesen Herausforderungen gehören: effiziente Datensammlung, die das Wissen von erforderlichen Datenmengen und relevanten Sensoren für erfolgreiches maschinelles Lernen verlangt; das anspruchsvolle Verstehen von komplexen Prozessen und facettenreichen Daten; eine geschickte Selektion geeigneter ML-Methoden und die Integration von Domänenwissen für die prädiktive Qualitätsüberwachung mit inhomogenen Datenstrukturen, usw.

Bestehende ML-Lösungen für ERW liefern keine systematische Vorgehensweise für die Methodenauswahl. Jeder Prozess der ML-Entwicklung erfordert ein umfassendes Prozess- und Datenverständnis und ist auf ein bestimmtes Szenario zugeschnitten, das schwer zu verallgemeinern ist. Es existieren semantische Lösungen für das Prozess- und Datenverständnis und Datenmanagement. Diese betrachten die Datenanalyse als eine isolierte Phase. Sie liefern keine Systemlösungen für das Prozess- und Datenverständnis, die Datenaufbereitung und die ML-Verbesserung, die konfigurierbare und verallgemeinerbare Lösungen für maschinelles Lernen ermöglichen.

Diese Arbeit versucht, die obengenannten Herausforderungen zu adressieren, indem ein Framework für maschinelles Lernen für ERW vorgeschlagen wird, und demonstriert fünf industrielle Anwendungsfälle, die das Framework anwenden und validieren. Das Framework überprüft die Fragen und Datenspezifitäten, schlägt eine simulationsunterstützte Datenerfassung vor und erörtert Methoden des maschinellen Lernens, die in zwei Gruppen unterteilt sind: Feature Engineering und Feature Learning. Das Framework basiert auf semantischen Technologien, die eine standardisierte Prozess- und Datenbeschreibung, eine Ontologie-bewusste Datenaufbereitung sowie halbautomatisierte und Nutzer-konfigurierbare ML-Lösungen ermöglichen. Diese Arbeit demonstriert außerdem die Übertragbarkeit des Frameworks auf einen hochpräzisen Laserprozess.

Diese Arbeit ist ein Beginn des Wegs zur intelligenten Fertigung von ERW, der mit dem Trend der vierten industriellen Revolution korrespondiert.

Abstract

Electric Resistance Welding (ERW) is a group of fully automated manufacturing processes that join metal materials through heat, which is generated due to electric current and resistance. Precise quality monitoring of ERW can often be performed only partially by using destructive methods. There is huge industrial and economical potential to develop data-driven approaches for quality monitoring in ERW, to reduce maintenance cost and improve quality control. Data-driven approaches, such as Machine Learning (ML), have attracted much attention due to the enormous amount of available data provided by technologies of Industry 4.0. Data-driven approaches allow non-destructive, all-covering and precise quality monitoring if a certain amount of precise data are collected. This can enable large-scale and online quality monitoring that are otherwise extremely difficult through traditional empirical methods.

However, there remain many challenges of adoption of such approaches in manufacturing industry. These challenges include: adequate data collection which requires knowledge of necessary data amount and relevant sensors for successful machine learning; understanding of sophisticated process and multi-faceted data and efficient data management; selection of machine learning methods and integration of domain knowledge for predictive quality monitoring with inhomogeneous data structures, etc.

Existing ML solutions for ERW do not provide systematic approaches for method selection. Each process of ML development requires extensive process and data understanding and is tailored to a specific scenario, thus difficult to generalise. There exist semantic solutions for process understanding, data understanding and management. These solutions consider data analysis as several isolated stages. They do not provide system solutions for process and data understanding, data preparation, and ML enhancement that allows configurable and generalisable machine learning solutions.

This work strives to address these challenges by proposing a framework of machine learning for ERW and demonstrates five industrial use cases that apply and evaluate the framework. The framework revisits the questions and data particularities, suggests simulation-aided data collection, discusses machine learning methods organised in two groups, feature engineering and feature learning, and relies on semantic technologies to allow standardised process and data understanding, ontology-aware data preparation, and semi-automated and user-configurable machine learning solutions.

Furthermore, this work also applies the same framework on another manufacturing process with a use case: the high-precision laser process, to demonstrate the transferability of the framework.

This work starts the journey towards a more intelligent manufacturing, which will merge to the grand trend of the Fourth Industrial Revolution

Acknowledgment

Firstly I greatly appreciate Prof. Ralf Mikut for recommending and providing me the opportunity to do PhD study at the Institute for Automation and Applied Informatics (IAI) at Karlsruhe Institute of Technology (KIT), for his guidance through the entire time of my PhD study, the encouragement during my trough time, the patience and sympathy to help me overcome predicaments, the support for solving many data science problems, the advice for academic development, correcting and reviewing my dissertation, and many more.

My deep gratitude to Prof. Volker Schulze for being the seconder reviewer and his valuable comments on this work.

My special thanks to Dr. Tim Pychynski, my supervisor at Bosch, for hiring me to conduct research at Corporate Research Center at Robert Bosch GmbH, for his inspirational ideas, the constructive discussions on the study, the constant support in helping me with organisational as well as technical issues at Bosch. I want to thank Robert Bosch GmbH for providing me the position and all resources necessary for the study.

I express my sincere and deep gratitude to Prof. Evgeny Kharlamov for his initiative of collaboration, enlightening discussions, valuable lessons in scientific writing, and the opening of a new world of semantic technologies to me.

I am deeply grateful to Dr. Dmitriy Mikhaylov for providing me the topic and data of high-precision laser processes. We had really efficient and productive join work.

Dr. Yulia Svetashova also deserves my deep gratitude, for providing me the support of semantic technologies, designing and conducting user studies together, and pointing to me learning resources of ontologies. We had great experience in developing the topic of semantic machine learning step-by-step to broader and very promising research directions.

Prof. Markus Reischl provided great support for statistic estimation with small sample size. I thank him sincerely for his excellent derivation of formulas and patient explanation.

I would like to thank the students that I have supervised or collaborated for their great support: Alexander Schif, Anusha Achutan, Seongsu Byeon, Carolin Diemer, Barun Mazumdar, Xinyuan Ge, and Fu Xing Long.

My appreciation to my group leader, Dr. Friedhelm Günter for the discussions during my presentations and the support in organisational issues.

Many thanks to my colleagues at Bosch and at IAI for the comments and ideas during my presentation discussions.

Foremost I want to thank my family. I thank my mother and father for their support and precious education about life attitude, views, and personal development.

Leonberg, 21 September 2020 *Baifan Zhou*

Contents

Symbols and Acronyms . . . . **xi**

1 Introduction **1**

1.1 Background and Motivation 1

1.2 Electric Resistance Welding 5

1.2.1 Description of Electric Resistance Welding Process . 5

1.2.2 State-of-the-art of Process Development, Quality Monitoring, and Maintenance in ERW 6

1.2.3 Typical Difficulties in Electric Resistance Welding . 9

1.3 Introduction to Machine Learning and Ontology 11

1.3.1 Feature Engineering + Classic Machine Learning . . 14

1.3.2 Feature Learning + Neural Networks (Deep Learning) 15

1.3.3 Introduction to Ontology 16

1.4 Machine Learning in Manufacturing 19

1.4.1 Overview of Machine Learning in Manufacturing . . 19

1.4.2 Machine Learning in Condition Monitoring 20

1.4.3 Machine Learning in Metal Welding 21

1.4.4 Previous Studies of Machine Learning in Electric Resistance Welding 22

1.5 Open Questions 25

1.6 Objectives and Thesis Outline 28

2 A Framework of Machine Learning in Electric Resistance Welding **31**

2.1 Question Definition Revisited 31
2.1.1 Quality Indicators of Different Fidelity Levels and Accessibility 32
2.1.2 Multi-levels of Temporal Structures in Data 34
2.1.3 Insufficient Data Problem and Three Data Challenges 37
2.1.4 Multi-fidelity Data Model and Application Questions 40
2.2 Data Collection: Data Acquisition and Evaluation 43
2.2.1 Overview: Simulation-supported Interactive Data Acquisition and Analysis 44
2.2.2 Similarity Analysis between Datasets 45
2.2.3 Inverse Modelling of the Simulation Model 48
2.2.4 Data Amount and Feature Sets Analysis 49
2.2.5 Evaluation of Datasets 50
2.3 Data Preparation: Ontology-supported Data Understanding and Integration 56
2.3.1 Ontology-supported Process & Data Understanding . 58
2.3.2 Ontology-supported Data Integration 64
2.3.3 Linkage to Machine Learning Ontology 68
2.4 Data Preprocessing and ML Modelling 69
2.4.1 Data Preprocessing: Handling Data with Hierarchical Temporal Structures 71
2.4.2 Data Preprocessing: Domain Knowledge Integrated Feature Engineering 74
2.4.3 Data Preprocessing: Feature Reduction 80
2.4.4 Data Preprocessing: Normalisation 81
2.4.5 ML Modelling: Classic Machine Learning Method Selection 82
2.4.6 Data Preprocessing and ML Modelling: Ontology-enhanced Machine Learning for FE-CML 82
2.4.7 Data Preprocessing and ML Modelling: Hierarchical Feature Learning 89

2.4.8 Evaluation: Discussion of Performance Metrics . . . 90

3 Implementation 93

3.1 Implementation with Python 93

3.2 Implementation with SciXMiner 99

4 Use Cases 103

4.1 Spot Diameter Estimation and Guidance for Simulation Data Collection 103

4.1.1 Question Definition 104

4.1.2 Data Description 105

4.1.3 Experiment Settings 106

4.1.4 Results and Discussion 108

4.1.5 Conclusion and Outlook 110

4.2 Evaluation of Production Datasets 114

4.2.1 Question Definition 114

4.2.2 Data Description 115

4.2.3 Experiment Settings 117

4.2.4 Results and Discussion 118

4.2.5 Conclusion and Outlook 123

4.3 Quality Prediction with Feature Engineering and Feature Learning 124

4.3.1 Question Definition 124

4.3.2 Data Description 125

4.3.3 Experiment Settings 126

4.3.4 Results and Discussion 133

4.3.5 Conclusion and Outlook 137

4.4 A Closer Study on Feature Engineering and Feature Evaluation 139

4.4.1 Question Definition 139

4.4.2 Data Description 139

4.4.3 Experiment Settings 140
4.4.4 Results and Discussion 146
4.4.5 Conclusion and Outlook 154
4.5 Transferability of Models between Datasets 156
4.5.1 Question Definition 156
4.5.2 Data Description 157
4.5.3 Experiment Settings 157
4.5.4 Results and Discussion 159
4.5.5 Conclusion and Outlook 162
4.6 Transferability to Another Application: Laser Welding . . . 164
4.6.1 Question Definition 165
4.6.2 Data Description 167
4.6.3 Experiment Settings 167
4.6.4 Results and Discussion 169
4.6.5 Conclusion and Outlook 172

5 Conclusion and Outlook 175

A Appendix 181
A.1 List of Figures and Tables 181
A.2 List of Open Questions and Application Questions 189
A.3 Following Use Case 4.2: Dataset Evaluation of Additional Welding Machines 192
A.4 Determination of Minimal Number of Data Tuples for Statistical Estimation 193

Symbols and Acronyms

#	Number of
a	Value variable of a data tuple in raw data
ANN	Artificial Neutral Networks
ANOVA	ANalysis Of VAriance
AQ	Application Question
b	Value variable of a data tuple in the trend
BN	Bayesian Network
BRNN	Bidirectional Recurrent Neural Networks
C	Classification
CART	Classification And Regression Trees
χ^2_{n-1}	Random variable with chi-square distribution of $n-1$ degree of freedom
CM	Condition Monitoring
CNN	Convolutional Neural Networks
C_p	Indicator of process capability
C_{pk}	Adjusted C_p for non-centred distribution
CPS	Cyber-Physical Systems
csv	Comma separated values
D	Diameter
DKFE	Domain Knowledge based Feature Engineering
DL	Deep Learning
DM	Data Mining
DNN	Deep Neural Networks
DT	Decision Trees

D_{tr}	Training dataset
D_{trx}	Training and validation dataset, also referred to as trainingx data
D_{tst}	Test dataset
D_{val}	Validation dataset
E_{pitt}	Pitting potential
ERW	Electric Resistance Welding
ExpR	Linear Regression with Exponential terms
F	Force
FE	Feature Engineering
$FeatSet_{prod}$	Feature Set of production
FEM	Finite Element Method
FL	Feature Learning
FN	False Negative
FP	False Positive
FS	Feature Selection
FSW	Friction Stir Welding
$\{F_t\}$	Single features describing the temporal structures of data
FuzzyNN	Fuzzy Neural Networks
F_X	Cumulative distribution function of a random variable X
GA	Genetic Algorithm
GMAW	Gas Metal Arc Welding
GRNN	General Regression Neural Networks
h	Height
HopfieldNN	Hopfield Neural Networks
HS	Hot-Staking
I	Current
IoT	Internet of Things
IQR	Interquartile Range

k	One time step of any temporal structure features
KELM	Kernel Extreme Learning Machine
kNN	K-Nearest Neighbours
Lab	Laboratory
LDA	Linear Discriminant Analysis
len	Length of time series
LogisticR	Logistic Regression
LogR	Linear Regression with Logarithmic terms
LR	Linear Regression
LSTM	Long Short-Term Memory
LTL	Lower Tolerance Limit
LVQ	Linear Vector Quantisation
M5	One type of decision tree learner for regression task
max	Maximum
MD	Mahalanobis Discrimination
min	Minimum
ML	Machine Learning
MLP	Multilayer Perceptron
$mnpo$	Minimum position
μ	Mean value
$mxpo$	Maximum position
n	Number of data tuples in a dataset
nBest	N Best features selection
N_{DC}	Dress Count
N_{EC}	Electrode Count
N_{WC}	Wear Count
O	Optimisation
o	Value variable of a data tuple of outliers
OECD	Organisation for Economic Cooperation and Development

OEM	Original Equipment Manufactures
OQ	Open Question
OWL	Ontology Web Language
P, $ProgNo$	Welding program number
PCA	Principle Component Analysis
PolyR	Polynomial Regression
Prod	Production
Prog	Welding program
PSO	Particle Swarm Optimization
Q_1	First quartile
Q_3	Third quartile
QDA	Quadratic Discriminant Analysis
QI	Quality Indicator
QI_{est}	Estimated value of a Quality Indicator
QI_{true}	Ground-truth of a Quality Indicator
QMM	Quality Monitoring in Manufacturing
Q-Value	One type of quality indicator for welding
R	Regression
rdf	Resource description framework
rdfs	Resource description framework schema
rms	Root mean squared
RNN	Recurrent Neural Networks
RSW	Resistance Spot Welding
SAE	Stacked Auto-Encoder
SAM	Scanning Acoustic Microscopy
SAW	Submerged Arc Welding
SBFS	Sequential Backwards Floating Selection
SBS	Sequential Backwards Selection
SF	Single Features
SFFS	Sequential Forwards Floating Selection
SFS	Sequential Forwards Selection

σ, std	Standard deviation of a random variable
Sim	Simulation
SOM	Self-Organising Maps
SPC	Statistic Process Control
$s_{T,n}$	Estimated standard deviation for a random variable. n data tuples used for the estimation
SVM	Support Vector Machines
SWRL	Semantic Web Rule Language
sz	Number of time series features
t_{n-1}	Random variable with t-distribution of $n-1$ degree of freedom
t	Time
T	Temperature
TN	True Negative
TP	True Positive
t_P	Sliding window length, including only points of program P
tr	Training
TS	Time Series
TSFE	Time Series Feature Engineering
TSS	Tensile Shear Strength
tst	Test
U	Voltage
UTL	Upper Tolerance Limit
val	Validation
x	Value of a random variable
X	Random variable
xsd	XML schema definition
$x_{T,n}$	Estimated mean value for a random variable. n data tuples are used for the estimation

1 Introduction

1.1 Background and Motivation

Manufacturing industry has long been playing an important role in economic growth and employment in Germany and worldwide. The term **manufacturing** can be generally understood as *transformation of material inputs and immaterial inputs into new products* [1, 2]. It is the main contributor to innovation, exports and productivity growth [3]. In 2017, manufacturing industry contributes to about 30% economic growth in Germany and worldwide (Figure 1.1), over 90% to export, 65% to innovation, and 40% to productivity growth in the 24 countries (Table 1.1). Manufacturing is especially important for Germany, as Germany is one of the leading export countries in the world [4]. Productivity describes the rate at which goods and services are produced [5, 6], and is important to a country's living-standard [5] and economic well-being [7].

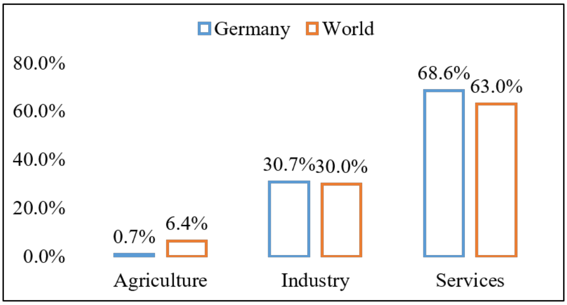


Figure 1.1: GDP by sectors in Germany and worldwide [8, 9]

Table 1.1: Contribution of each sector (percent) of 24 countries in OECD for 2000-14 [10]

Sector	Share employment	Share value added	Share exports	Share R&D spending	Productivity growth
Agriculture	2.34	4.61	6.90	1.24	32.91
Manufacturing	16.92	16.96	**92.01**	**64.86**	**42.32**
Service	**80.74**	**78.43**	1.09	33.90	24.77

Therefore, many major economies have initiatives to foster the development of manufacturing industry [11]. Among which, **Industry 4.0**, first initiated by the German government [12], is also increasingly affecting European policy [11]. The term Industry 4.0, coined by Kagermann in 2011, represents the fourth industrial revolution [13]. It results from the introduction of the Internet of Things and Services into the manufacturing environment [14]. **The Internet of Things (IoT)** is defined as *a global infrastructure enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies* [15]. Kagermann describes Industry 4.0 in the way that *smart machines, storage systems and production facilities will be incorporated into an aggregate, called* ***Cyber-Physical Systems (CPS)*** [14, 16]. The resulting **Smart factory** is a production solution that integrates networked manufacturing systems and products to enable flexible and agile production [17] to meet individual customer requests [14].

According to these definitions and descriptions, this work understands Industry 4.0 as a series of profound changes towards to omniscient sensoring, ubiquitous connecting, and decentralised, intelligent information processing in manufacturing industry.

There exist various surveys for key technologies required for Industry 4.0 and Internet of Things (IoT) [18, 14, 19, 20]. This work summarises these research requirements in a non-comprehensive list from a data-centric view:

1. **Sensor technology** enabling data collection

2. **Communication technology** facilitating data exchange between devices
3. **Computational technology** providing sufficient computational power and intelligent data processing algorithms
4. **Actuator technology** meaningfully transforming data into machine action

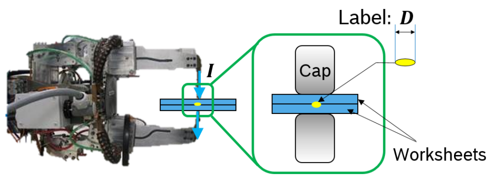
Machine Learning, categorised into computational technology, is one of the key technologies of Industry 4.0 [21, 11, 22]. Machine learning provides intelligent data processing algorithms. In manufacturing, it has created new intelligent tools for automatic information extraction and knowledge discovery [23, 24], which are important for practices like root cause identification or decision-making [25]. A big data problem arises because an unprecedented amount of data are collected in manufacturing industry nowadays [26]. The resulting "4Vs" requirements need to be addressed: *Volume, Velocity, Variety, Veracity (Authenticity)* [27, 28].

In an Industrial 4.0 scenario, voluminous data are collected through interconnected devices, which by themselves possess a certain degree of processing power. The large amount of data are stored in a central or distributed database, and then centrally or distributedly processed by some intelligent algorithms, to extract organised information and systematic rules, which culminate as "knowledge" learned from the data, enabling machine guided process optimisation, namely machine learning. The key differences between the human guided process optimisation and machine guided process optimisation lies thereupon, that the machine guided process optimisation is not explicitly programmed, and requires no or little manual effort in designing all the optimisation parameters and even procedures. Instead, the process of optimisation, or learning in another word, happens in an at least partially automatic way. Yet, no machine learning method is able to build a technologically meaningful information processing from an unstructured dataset on its own. Each application domain needs to be analysed and then appropriate machine learning processes can be designed.

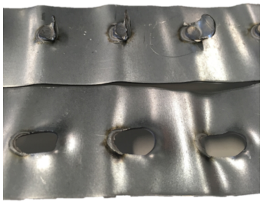
Electric Resistance Welding (ERW), widely applied in automotive production, is a typical automatic manufacturing process with inhomogeneous data structures as well as statistical and systematic dynamics [29]. Quality monitoring is to estimate or predict the quality characterised by some categorical or numerical values. It is an essential task in ERW as well as in manufacturing (Other tasks include process development and optimisation, etc.). There exists huge industrial and economic potential to develop data-driven approaches, especially machine learning approaches, for quality monitoring in ERW, to reduce maintenance cost and improve quality control. Machine learning approaches are non-destructive and can potentially monitor the quality of all welding processes. Recently many ML-tools [30] have become available thanks to extensive research in this area.

The central question of this thesis is to identify challenges of developing machine learning approaches for quality monitoring in ERW and to explore solutions to them.

The rest of this chapter is organised as follows. Section 1.2 introduces the electric resistance welding process and summarises the typical difficulties in this field. Section 1.3 gives a short overview of machine learning and ontology background knowledge necessary for this work. Section 1.4 takes a glance at recent development of machine learning in manufacturing to see if there exist similar challenges and solutions in other manufacturing fields, and narrows the scope step by step from manufacturing in general, to condition monitoring, metal welding technology, and electric resistance welding. Section 1.5 derives the open questions in the field of *machine learning in electric resistance welding* by translating the difficulties in Section 1.2 and partially adapting the challenges in previous studies in Section 1.4. Section 1.6 decomposes the central question into sub-objectives and lists the organisation of the thesis.



(a)



(b)

Figure 1.2: (a) RSW as an example of ERW. The current *I* flows through the welding zone and results in a spot-form connection [29]
(b) Measuring the welding spot diameter by tearing the welded worksheets apart [29].

1.2 Electric Resistance Welding

This section first gives a short description of the ERW process [31] (Section 1.2.1), then introduces state-of-the-art of process development, monitoring and maintenance in ERW (Section 1.2.2), and at the end summarises the typical difficulties (Section 1.2.3).

1.2.1 Description of Electric Resistance Welding Process

Electric resistance welding (ERW) is *a group of welding processes with pressure in which the heat necessary for welding is produced by resistance to an electrical current flowing through the welding zone* [32]. The welding zone is between the electrodes (Figure 1.2), including the small volume of worksheets and their contacting surface. After the welding process, a connection will be formed in the contact areas. The connection can have the form of e.g. spots (Resistance Spot Welding, RSW) or wire-hook joint (Hot Staking, HS).

In industrial practice, ERW is ubiquitously applied. The wide application of the ERW is the result of numerous advantages of ERW, including minimal deformation of the welded products, excellent chemical and mechanical properties of the welding areas, time- and energy-efficiency, high degree of automation, etc. However, there still exist many difficulties in ERW (Section 1.2.3).

1.2.2 State-of-the-art of Process Development, Quality Monitoring, and Maintenance in ERW

To understand the central question of quality monitoring in ERW, it is necessary to understand a general process pipeline in ERW (Figure 1.3). The raw products, such as chassis for RSW and electric motors for HS, are transformed into products by a welding process. The welding process is controlled by a welding control, normally an *adaptive control*, which forces the actual process variables, such as process curves of current, voltage, to follow predefined values or profiles named as setpoints or reference curves. The welding control also constantly collects measurement data from the process, and compares these measurements with the references stored in the adaptive control, to give estimations of *quality indicators*. Quality indicators describe quality, which is the degree to which a set of inherent characteristics fulfils

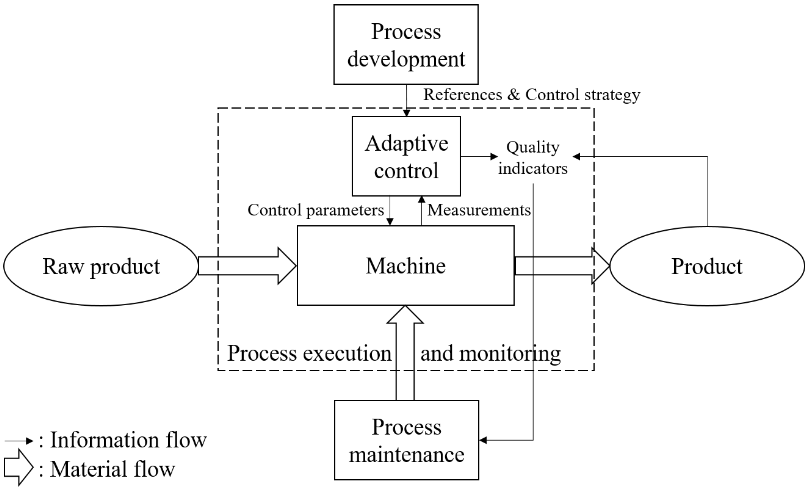


Figure 1.3: A general process pipeline for electric resistance welding. Thick arrows indicate material and energy flow. Thin arrows indicate information flow.

requirements [33]. The references of quality indicators are obtained in **process development**, where a certain number of manufacturing operations are carried out without the adaptive control. Some of these operations, whose quality indicators are estimated as good, are selected for determining the references.

Quality monitoring assesses various quality indicators. Some quality indicators are measured (QI_{meas}), including e.g. welding spot diameter for RSW [34, 35] or *tensile shear strength (TSS)* for both RSW and HS. For these two, relatively precise measurements can only be obtained through destructive methods (Figure 1.2b). In automotive industry, this means tearing the welded chassis apart. There also exist non-destructive methods to measure them, like ultrasonic and X-ray tests in RSW, but these methods also cause problems in terms of costs and facilities [36, 37], and are therefore only used for random samples [38]. Since the measured quality indicators are very costly, some empirical quality indicators (QI_{emp}) are developed, which are estimated through comparing references and measurements. Their expected behaviour is to be stable around a setpoint pre-defined by the welding program. The empirical quality indicators can reflect the measured quality indicators in most cases, but their correspondence is not perfect.

Measured quality indicators QI_{meas} and the estimated empirical value QI_{emp} both are assumed to follow Gaussian distribution (Figure 1.4). These empirical quality indicators are monitored according to principles of *Statistical Process Control (SPC)* [39]. A process is capable if *it adheres to the tolerance specifications defined with respect to the evaluated quality characteristics* [40], quantified by *process capability indices*, such as those in Equation (1.1). In the equation, UTL and LTL represent *upper tolerance limit* and *lower tolerance limit* (determined by experiments or empirically by process development engineers), respectively. $\hat{\mu}$ and $\hat{\sigma}$ stand for the mean and standard deviation (estimated from samples) of the quality indicators to be characterised. These indices evaluate the probability that the quality indicators lie in the tolerance limits, compared to a baseline of 6σ

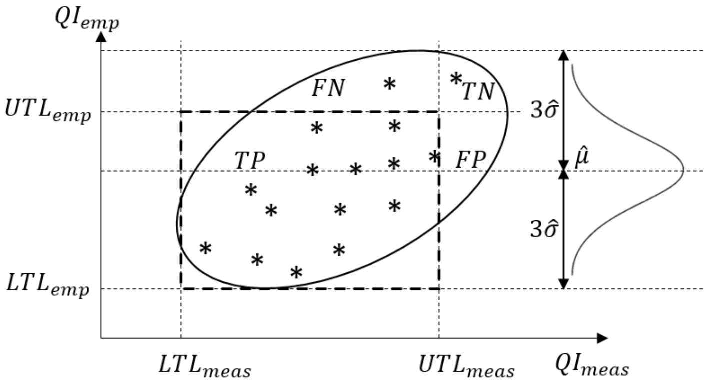


Figure 1.4: Two-dimensional Gaussian distribution (6σ area is drawn within an eclipse) of measured quality indicators QI_{meas} and empirical quality indicators QI_{emp}. UTL_{meas} and LTL_{meas} stand for the upper tolerance limit and lower tolerance limit of the QI_{meas}; UTL_{emp} and LTL_{emp} stand for the estimated upper tolerance limit and lower tolerance limit for QI_{emp}. The intersection area of the rectangle with thick dashed lines is the area of *True Positive (TP)*, where the QI_{emp} as well as the QI_{meas} lie within the respective tolerance bands. The other areas are *False Positive (FP)*, *False Negative (FN)*, and *True Negative (TN)* (Table 1.2). In the figure only the TP/FP/TN/FN for exceeding the UTL are drawn.

(a probability of 99.73%). The bigger $\hat{C}_p, \hat{C}_{pk}$ are, the more probable the quality indicators lie in the tolerance limits.

$$\begin{aligned} \hat{C}_p &= \frac{UTL - LTL}{6\hat{\sigma}} \\ \hat{C}_{pk} &= \frac{\min(UTL - \hat{\mu}, \hat{\mu} - LTL)}{3\hat{\sigma}} \end{aligned} \tag{1.1}$$

There exist four cases of whether QI_{emp} deviates from QI_{meas}, represented by a confusion matrix (Table 1.2). False Negative (FN) means a product of good quality is estimated to be of poor quality, which is a waste of production. False Positive (FP) means a product of poor quality is estimated to be of good quality, which is undetected quality failure. FP is normally much worse than FN, because FN is merely a waste of time and resource, but quality failure in e.g. vehicles could cause severe accidents.

Process maintenance in ERW include regular maintenance and maintenance in case of quality failure. The former one is needed because the

Table 1.2: Confusion matrix of estimated value and predicted value

	$LTL_{meas} \leq QI_{meas} \leq UTL_{meas}$	$QI_{meas} < LTL_{meas}$ or $UTL_{meas} < QI_{meas}$
$LTL_{emp} \leq QI_{emp} \leq UTL_{emp}$	True Positive (TP)	False Positive (FP)
$QI_{emp} < LTL_{emp}$ or $UTL_{emp} < QI_{emp}$	False Negative (FN)	True Negative (TN)

electrodes directly touching the parts wear due to e.g. oxidation, material sticking, etc. Common regular maintenance can be e.g. *Dressing* [41], that is to remove a thin surface layer of the electrodes to restore the surface condition, or that the electrodes will be changed when dressing is not working anymore. The regular maintenance is currently performed with a fixed period. In case a quality failure is detected by the adaptive control, the whole production line will be stopped and necessary manual interference is needed, causing enormous economic loss.

1.2.3 Typical Difficulties in Electric Resistance Welding

To achieve successful quality monitoring in ERW, many questions and difficulties need to be solved. This work summarises these difficulties in two aspects.

A priori knowledge problems are problems of limited domain knowledge.

- *Empirically understood process*. There exist many effects that are limitedly and empirically understood, e.g. contact resistance between electrodes and worksheets, strong electric-magnetic interferences, and empirical quality indicators. Their mechanism and influence on quality can therefore not thoroughly described with domain knowledge.
- *Systematic variance*. The systematic variance is caused by systematic change of welding conditions, e.g. the wearing effect of electrodes through

time, system and environmental temperature change. The mechanism of how these changes influence welding is limitedly understood.

- *Unknown discrepancy between process development and deployment.* The references of quality indicators are obtained in process development, but the conditions are often different to real production conditions. This discrepancy may cause a discrepancy between the quality of process deployment and the quality of process development.

Data problems are problems of lacking of data.

- *Partial coverage of quality monitoring.* Measured quality indicators require destructive methods like tearing the welded products apart, or non-destructive methods like X-ray or ultrasonic test. These methods are only carried out sample-wise on the products and data of measured quality indicators are mostly unavailable. A full-coverage quality monitoring is needed.
- *Expensive data acquisition.* Both destructive methods that destroy the products and non-destructive methods that require extra X-ray or ultrasonic equipment are extremely expensive. Moreover, extra sensor measurements that can potentially improve quality monitoring are also costly and therefore not available in production.
- *Data variety.* Data collected from multiple sources have various differences. Production data are generated by multiple versions of production systems due to user individualisation. Laboratory or simulation data often contain more sensor measurements and are stored with various formats, variable names, sampling frequency, etc.
- *Statistic variance.* The statistic variance is the result of multiple uncontrollable variables, e.g. different properties of raw products that are manufactured by different suppliers in multiple production batches, despite that they are nominally identical.

To address or partially solve these difficulties, this thesis suggests using *Machine Learning*, which is a data-driven approach, limiting the cost, and has great modelling power for complicated problems. Only after properly translating the difficulties in ERW and manufacturing to questions in machine learning (Section 1.5), it becomes possible to develop machine learning approaches.

1.3 Introduction to Machine Learning and Ontology

The term **machine learning (ML)** was coined by Samuel in 1952 [42]. From a handful of engineering practices and statistics, machine learning has developed into a broad discipline both rich in applications and theory [43]. There exist different definitions or descriptions for machine learning and data mining [44, 45, 46]. One common understanding of machine learning (ML) is *using statistical theory in building mathematical models to enable computers to make inference from data without being explicitly programmed* [46, 47]. This seems to emphasise the modelling aspect mentioned in the definition of **data mining**, *the complete process of data collection, preparation, preprocessing, modelling and interpretation* [48]. Since modelling would be not possible or meaningful without the other steps, especially in manufacturing, data mining and machine learning in this work are treated as having the same meaning. In the following text, only the term *machine learning* will be used. Machine learning, in a higher view, is a *process of extracting information and learning knowledge from data, to support decision-making*, and in the long-term to form knowledge and wisdom (Figure 1.5) [48, 49].

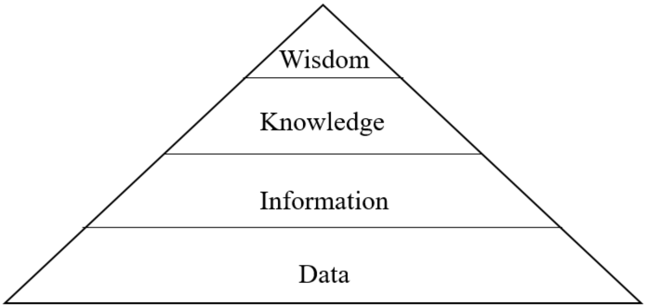


Figure 1.5: Pyramid of data mining: extracting information from data, learning knowledge from information, and achieving wisdom with knowledge [49]

The pre-requisite of successful machine learning practice is a good **question definition** in the specific domain and a meaningful transformation of the question into mathematical representation [48] (question definition in machine learning). Figure 1.6 shows a typical workflow of machine learning. The question can be defined before or after data collection.

Data collection is the process of collecting raw data. This process could be measuring of some physical quantities manually, or by sensors, cameras, etc. Raw data are collected in various formats, e.g. csv files, txt files, SQL databases.

Data preparation is the process of merging different sources of data into one uniform format, possibly also changing their naming, to facilitate visualisation or analysis. Until this step, the statistic properties of data are unchanged.

Data preprocessing and **modelling** are two deeply intertwined steps. *Data preprocessing* is to change the representation of data so that it can be used by the subsequent machine learning algorithms. Common data preprocessing procedures are feature extraction, selection, normalisation, etc. The statistic properties of data will be changed in this step. *Modelling* is to use machine learning algorithms, ranging from classic machine learning meth-

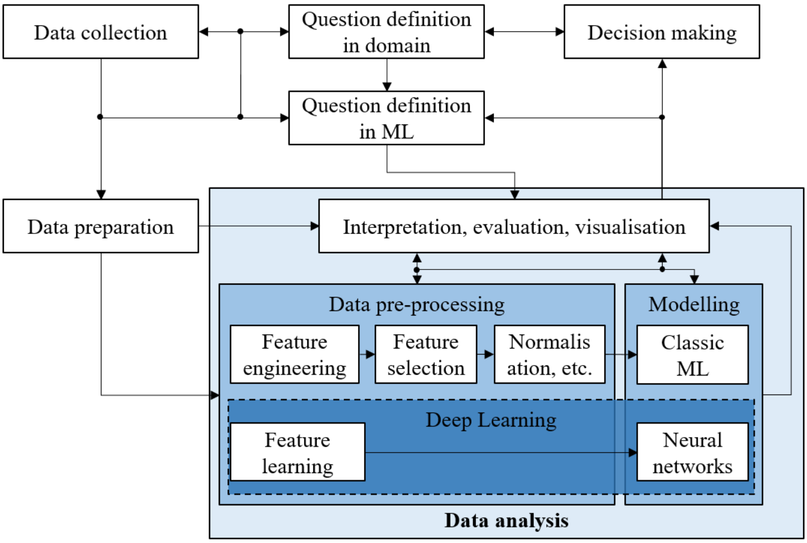


Figure 1.6: A typical machine learning workflow [44, 50, 51]. The question can be defined before or after data collection.

ods to complicated neural networks, to extract statistic regularities from the input data and make predictions.

Interpretation of results is to combine the results delivered by data analysis and problem-specific domain knowledge and its mathematical formulation, to **evaluate** and **visualise** the meanings of the results to support **decision-making**.

Significant effort of machine learning flows into data preprocessing so that the resulting data representation or features [52] can facilitate effective machine learning modelling [53, 54]. The choice of **feature extraction** methods, which can be categorised into two groups, *feature engineering* (Section 1.3.1) and *feature learning* (Section 1.3.2), greatly impact the choice of the subsequent machine learning algorithms.

From this point of view, machine learning approaches can be largely divided into two groups [51]:

- feature engineering + classic machine learning algorithms;
- feature learning + neural networks, or more commonly known as *deep learning*.

It is important to note that there exists no clear boundary between the two groups in a mathematical or even sometimes methodological sense. Yet these two groups follow two different philosophies. The former emphasises on integration of domain knowledge in feature engineering and interpretability of the extracted features, ML algorithms and results. Instead, the latter strives to minimise the effort to use domain knowledge for feature extraction, and to build rather general machine learning algorithms for different purposes.

1.3.1 Feature Engineering + Classic Machine Learning

Feature Engineering (FE) is the process of transforming features through domain expert knowledge [51] or mathematics so that machine learning algorithms work. The choice of data representation is crucial to the performance of machine learning algorithms that are applied on the data. Feature engineering integrates human ingenuity and prior knowledge to compensate that some machine learning models are not effective and delicate enough to capture the subtle discriminative information lying in the data [52].

Common feature engineering methods include:

- Generating new single features from raw single features using unary or binary operations [55], e.g. logarithmic, reciprocal, sine/cosine, correlation features, etc.;
- Extracting single features from data with temporal or spatial structures, e.g. segmenting [56], subsampling [57, 58], calculating statistic properties like maximum, variance [59], filtering [60], transforming to frequency domain [61, 62] or time-frequency domain [62], etc.;
- Reducing the raw features to a lower dimensional space, e.g. Principal Component Analysis (PCA) [63], Linear Discriminant Analysis (LDA) [64], etc.

After feature engineering, even very simple machine learning algorithms can deliver good results. These simple algorithms can be Linear Regression (LR), Polynomial Regression (PolyR) [45], k-Nearest Neighbours (kNN) methods [65], etc. The advantage of the pipeline of feature engineering + classic machine learning is that the models are more transparent and understandable than feature learning. Which features are more important and how they influence the model quality is easily interpretable from the model results.

1.3.2 Feature Learning + Neural Networks (Deep Learning)

According to [52], **Feature Learning (FL)**, or representation learning, refers to an algorithm that *can learn to identify and disentangle the underlying explanatory factors hidden in the observed milieu of low-level sensory data*. Feature learning was initiated to avoid labour intensive and domain-specific feature engineering. The advantage of feature learning is that the learning algorithms become less dependent on feature engineering so that novel applications could be developed faster.

The current feature learning study has developed into two parallel approaches: probabilistic graphical models and neural networks (or deep learning) [52]. The former approach attempts to recover latent random variables describing a distribution, while the latter one uses a computational graph to extract abstract features from the data. There exists no formal definition for the term deep learning, but various literature [52, 66, 67, 68, 69] shares the view that **Deep Learning (DL)** is *a group of machine learning algorithms for multiple layers (or levels) of non-linear information processing to learn hierarchical representations (or abstractions) of data, mainly with neural networks*. Some authors thinks *the hierarchical feature learning* is the most important characteristic for deep learning [70]. Others think *more than one* hidden layer is required [71]. For simplicity and clearance, this work refers to deep learning as *neural networks with more than one hidden layer for hierarchical feature learning*, i.e. deep learning has at least two hidden layers, one input layer, and one output layer. Artificial neural networks with fewer than two hidden layers, or that do not possess the characteristic of hierarchical feature learning, are categorised into classic machine learning.

A basic type of deep learning is Deep Neural Networks (DNN), which is MLP with at least two hidden layers [72]. There exist other types of neural networks designed for specific data structures, e.g. Convolutional Neural Networks (CNN) [73] for data with temporal or spatial structures, Recurrent Neural Networks (RNN) [74] or Bidirectional Recurrent Neural Networks (BRNN) [75], and Long Short-Term Memory (LSTM) [76] for data with temporal structures, etc.

1.3.3 Introduction to Ontology

An ontology [77, 78] is a shared conceptualisation of a domain of interest written in a formal language. The ontological modelling in this work follows the description logic of OWL 2 [79], which is built upon the World Wide Web Consortium's (W3C) XML standard for objects called the Resource Description Framework (RDF) [80] and compatible with RDF Schema (RDFS) [81].

Ontology and its related semantic technologies, represent semantics (meaning) and associations between data [82]. Semantic technologies have recently gained a considerable attention in industry for a wide range of applications and automation tasks such as modelling of industrial assets [83] and industrial analytical tasks [84], integration [85, 86, 87, 88] and querying [89] of production data, and for process monitoring [90] and equipment diagnostics [91].

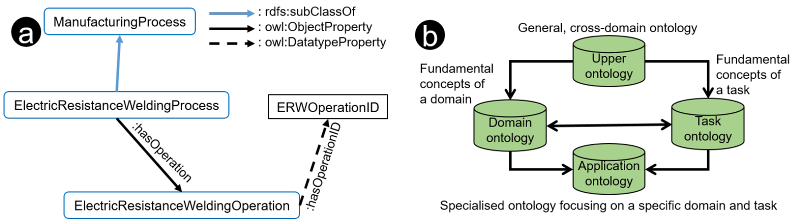

a
ManufacturingProcess
: rdfs:subClassOf
: owl:ObjectProperty
: owl:DatatypeProperty
ElectricResistanceWeldingProcess
ERWOperationID
:hasOperation
:hasOperationID
ElectricResistanceWeldingOperation
b
General, cross-domain ontology
Upper ontology
Fundamental concepts of a domain
Fundamental concepts of a task
Domain ontology
Task ontology
Application ontology
Specialised ontology focusing on a specific domain and task

Figure 1.7: (a) An example of ontology. (b) Different types of ontologies [92].

All entities, including physical objects and abstract concepts are described with *rdfs:Class*[1] (blue rounded squares in Figure 1.7a). Data are modelled with *rdfs:Literal* and connected by *owl:DatatypeProperty* (black squares in Figure 1.7a) to *rdfs:Class*, described with *xsd:<schema>*. The prefixes indicate their following namespace.

Each formula in ontology states that one complex class is a subclass of another class, or a property is a subproperty of another property. Complex classes are composed from the atomic classes and properties using logical AND, OR, and Negation, as well as universal and existential quantifiers. Reasoning over ontologies allows to compute logical entailments.

Figure 1.7a illustrates a simple ontology. The default namespace here is *Manufacturing Ontology*, for which the prefix is *mo:*, or simply a colon :. In the ontology, three classes and one datatype property are defined. These classes are connected with two types of predicates, *rdfs:subClassOf*, indicating a hierarchy of categories. It can be serialised in Manchester Syntax [77]:

```
Class: mo:ElectricResistanceWeldingProcess
    SubClassOf: core:ManufacturingProcess
```

and *owl:ObjectProperty*, indicating a non-hierarchical relationship:

```
ObjectProperty: mo:hasOperation
    Domain: mo:ElectricResistanceWeldingProcess
    Range: mo:ElectricResistanceWeldingOperation
```

The *owl:DatatypeProperty* is a property connected to a feature in data:

[1] Note there exists no space after the colon in the representation of ontologies.

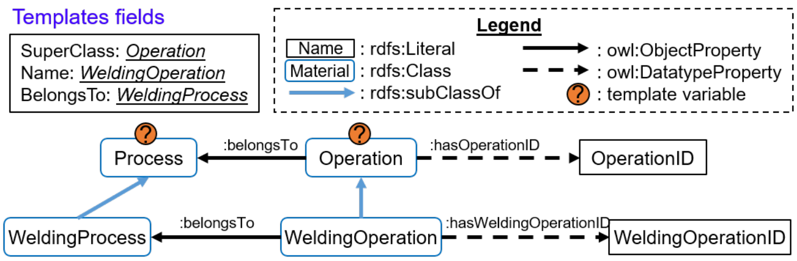


Figure 1.8: An example of ontology templates. By providing values for the variables in the template (middle row in the figure), an ontology can be instantiated (lower row).

DataProperty: *mo:hasOperationID*

Domain: *mo:ElectricResistanceWeldingOperation*

Range: *xsd:string*

The predicates of object properties and data properties can also be grouped hierarchically using *rdfs:subPropertyOf*.

Ontologies are of different types (Figure 1.7b). An *Upper Ontology* is to describe a general and cross-domain ontology, e.g. a *Manufacturing Ontology*. A *Domain Ontology* contains terms that are fundamental concepts of a domain of interest, e.g. a *Electric Resistance Welding Ontology* and they are subclasses of the upper ontology. A *Task Ontology* includes fundamental concepts of a task, e.g. a *Machine Learning Ontology* for machine learning analysis in this work. An *Application Ontology* is a specialised ontology focusing on a specific domain and task, e.g. a *Quality Monitoring Ontology* of using machine learning for welding quality monitoring.

Ontology templates are a language and tools for creating OWL ontology models with parametrised ontologies by providing values for each variable [93]. Figure 1.8 gives an example for creating a small ontology that contains *Welding Operation*, which belongs to *WeldingProcess* and has *WeldingOperationID*. The user needs to select the super class *Operation*, specifies its class name, and chooses its belonging *WeldingProcess* (this class is created beforehand). Then the ontology for welding (domain ontology) is created from the template (upper ontology), and serialised as OWL axioms. Templates guarantee good quality and consistency of the created ontology, as well as the relative simplicity of the ontology creation process.

1.4 Machine Learning in Manufacturing

This section first takes a glance at machine learning in manufacturing in general, and then narrows the scope to condition monitoring, other metal welding processes, and finally ERW.

1.4.1 Overview of Machine Learning in Manufacturing

In recent years, machine learning in manufacturing has been increasingly drawing attention [94, 24, 95]. It is developed and applied in various subfields in the broad and comprehensive realms of manufacturing industry, including plant-wide optimisation, sustainable production, agile supply chain [26], and product lifecycle management [28], etc., to achieve goals [94] like cost estimation, quality monitoring and improvement, fault diagnosis, condition monitoring, process optimisation, etc.

The **challenges** of machine learning in manufacturing [94] include

- acquisition of relevant data [95];
- big data problems like handling high-dimensional and voluminous data;
- skewed distribution of failure types;
- selection of suitable machine learning algorithm;
- interpretation of the results;
- adaptability to changing environment, conditions or problems.

Solutions to address these problems include:

- *Data lakes* [96] were proposed to store unstructured raw data;
- *Data warehouse* [97] for integrated data;
- Various descriptions or solutions on big data problems were proposed in [27, 28];
- Machine learning-algorithms were organised to different groups according to nature of defined question and data structure [94].

All these challenges also exist in machine learning in ERW. The solutions to address them were also adapted to ERW in this thesis. Details see Section 1.5 and 1.6.

1.4.2 Machine Learning in Condition Monitoring

Condition monitoring is a sub-discipline in manufacturing, referred to as *activity intended to assess the characteristics of the physical actual state of an item* [98]. Condition monitoring is frequently performed for **predictive maintenance**, which is defined as *condition-based maintenance carried out following a forecast derived from repeated analysis and evaluation of the degradation of the item* [98].

Condition monitoring and the subsequent predictive maintenance can be performed for two types of purposes. **Machine health monitoring** [99] means to maintain the healthy state of an equipment, while **quality monitoring** is to ensure the product quality within acceptable limits.

The **challenges** in condition monitoring include using meta-information and sensor measurements for

- feature extraction, including feature fusion, dimension reduction [62], feature learning [99], etc.;
- fault detection and state classification, for machine health, e.g. wind turbine [100], bearing [62], line insulators [101], and for product [102] in e.g. plastic injection moulding [103];
- estimation (regression) of a machine health indicator or quality indicator, e.g. tool wear [104].

Solutions are feature engineering and classic machine learning models, e.g. random forests [105], support vector machines [106], fuzzy logic [107], Bayesian networks [107], and feature learning with Deep Learning algorithms, e.g. stacked auto-encoder [108], Deep Neural Networks [109].

The same challenges also exist in quality monitoring in ERW. Many of the proposed ML methods can also be used, but the **challenge** of selection of suitable machine learning algorithm needs to be solved. This thesis will address quality monitoring for welding in detail in the following two sections.

1.4.3 Machine Learning in Metal Welding

Metal welding is defined as *welding processes that join metal by means of heat or pressure, or both, in such a way that there exists continuity in the nature of the metal which has been joined* [32]. Welding technologies for non-metal material will not be addressed in this thesis.

In the literature, many studies have used machine learning approaches for quality monitoring in welding technologies. Here some examples are listed in detail.

The **challenge** is always estimation or prediction of quality indicators using classification or regression analysis, and optimisation of the process, examples include

- estimation of *lap-shear strength* and *welding seam width* in laser welding [110, 111];
- classification the *welding seam type* as good, lack of fusion, or burn through in Gas Metal Arc Welding (GMAW) [112, 113, 114, 115, 116], and process optimisation [117] ;
- estimation of *bead width*, *height* and *hardness* in Submerged Arc Welding (SAW) [118], and process optimisation;
- estimation of *tensile shear strength*, *yield strength*, *elongation* and *hardness*, in Friction Stir Welding (FSW) [119, 120] and process optimisation.

Solutions are data collection from sensor measurements and statistical modelling using various machine learning algorithms, examples include

- using features of welding speed, power, stand-off distance, clamp pressure in laser welding and modelling with Multilayer Perceptrons (MLP) and Polynomial Regression (PolyR) [110];
- using feature of welding speed, and statistic features extracted from the process curves of current (I) and voltage (U) in GMAW, and modelling with J48 Random Forest [112];

- using features of voltage, current, welding speed, nozzle-to-plate distance in SAW, evaluating with analysis of variance, and then modelling with polynomial regression [118];
- using features of tool rotation speed, profile, feed speed, and hardness in FSW, and modelling with Multilayer Perceptrons [119].

It is also worthy to notice that all these studies have collected data from laboratory experiments. Due to high cost of data collection their data amount are relatively small, e.g. 14 in [118] and 26 in [110].

1.4.4 Previous Studies of Machine Learning in Electric Resistance Welding

Many previous studies have adopted the approach of machine learning *to predict quality indicators* in ERW based on the historically recorded data. The most studied process is Resistance Spot Welding (RSW). No literature about machine learning in Hot-Staking (HS) is found. This section will review the previous studies from the following perspectives and summarise the detailed information in Table 1.3 and 1.4.

Question Definition. The previous studies have defined the question either as Classification (C) - that is to predict the category of quality: good, bad, and sometimes the concrete failure types - or Regression (R) - that is to assess a numerical value of the quality. Each welding operation is treated as an independent event. The quality indicators to be classified or predicted are usually spot diameter (D), spot height (h), Tensile Shear Strength (TSS), spatter, failure load (load). Special quality indicators include gaps [121], pitting potential (E_{pitt}) [122], and penetration [36]. Some works also studied Optimisation (O) of the process.

Data Collection. It is worthy to note that almost all of the previous studies have collected data from laboratory experiments, or from welding machines for production but under experimental settings, except for [123], which used accumulated production data. In fact, there exist at least three data sources throughout process pipeline of development, monitoring and maintenance: simulation, laboratory, and production data. The dataset size, i.e. number of welding spots, ranges from 10 to around 3000. For a summary of the previous two aspects see Table 1.3.

Data Preprocessing and Modelling. In the input features, **Single features (SF)** are recorded as constants for a welding process, including electrode force (F), welding current (I), welding time (t), sheet thickness (thickness), sheet coating (coating), pressure. Some features are recorded as **Time series (TS)**, including dynamic resistance (Re), electrode force (F), welding current (I), electrode displacement (s), welding voltage (U), ultrasonic oscillogram, power. Besides the two common types, [127] used images collected from Scanning Acoustic Microscopy (SAM).

Most of these studies have extracted geometric features or statistic features based on or inspired by domain knowledge, denoted as domain knowledge based feature engineering (DKFE), but the usage of domain knowledge for machine learning and interpretation is still limited. The methods for **Time Series Feature Engineering (TSFE)** include subsampling, segmentations, histogram, and transition points. Some studies extracted geometric features like slopes, lengths of slopes, signal drops from a peak to the following valley, or statistic features (stats), e.g. maximum (max), minimum (min), maximum position (mxpo), mean, standard deviation (std), range, root mean squared (rms), because these features are considered meaningful also from the ERW process perspective. A special method named as scale-space filtering [149] is used in [142] for times series segmentation. Some works applied a further step of Feature Engineering (FE) on raw single features and time series extracted features, using PCA, polynomial feature generation, discretisation, radar chart, Chernoff face. Feature Selection (FS) are performed in [135], [136], [139], applied methods including analysis of variance (ANOVA), power of the test, Sequential Forward Selection (SFS), Sequential Backward Selection (SBS), Sequential Forward Floating selec-

Table 1.3: An overview of related works of machine learning in ERW, Part 1. All studies are carried out with **laboratory** data source on RSW except for [123]. Explanations for acronyms see Section 1.4.4 or Table of Acronyms and Symbols.

Citation	Year	Author	Question	#Data
[38]	2000	Cho	R: TSS	50
[124]	2000	Li	R: D	170
[125]	2001	Lee	R: TSS	80
[126]	2002	Cho	R: D,TSS	60
[127]	2003	Lee	C: D	390
[128]	2004	Cho	C: TSS	10
[56]	2004	Junno	C: D	192
[129]	2004	Laurinen	C: D	192
[57]	2004	Park	C: TSS, h	78
[130]	2004	Podržaj	C: spatter	30
[131]	2005	Haapalainen	C: TSS	3879
[132]	2006	Tseng	R, O: load	25
[121]	2007	Hamedi	R, O: gaps	39
[133]	2007	Koskimaki	C: TSS	3879
[134]	2007	Martin	C: D	438
[135]	2008	El-Banna	C: D	1270
[136]	2008	Haapalainen	C: TSS	3879
[137]	2009	Martin	R: TSS	242
[36]	2010	El Ouafi	R: h, penetration, D	54
[122]	2010	Martin	R: E_{pitt}	242
[138]	2012	Li	R: D	145
[139]	2013	Panchakshari	R, O: D, TSS, load	25
[140]	2014	Afshari	R: D	54
[141]	2015	Yu	C: TSS, spatter	473
[58]	2015	Zhang	C: TSS, D	200
[142]	2016	Boersch	R: D	3241
[143]	2016	Pashazadeh	R, O: D, h	48
[144]	2016	Wan	C,R: D, load	60
[145]	2017	Summerville	R: D	126
[146]	2017	Summerville	R: TSS	170
[147]	2017	Sun	C: TSS	67
[148]	2017	Zhang	C: TSS	120
[123]	2018	Kim	R: D	3344

tion (SFFS), Sequential Backward Floating Selection (SBFS) and N-Best Features Selection (nBest).

Most of the subsequent **machine learning models** are classical machine learning methods like Linear Regression (LR), Polynomial Regression (PolyR), k-nearest neighbours (kNN), Bayesian Network (BN), Decision Trees (DT), Genetic Algorithm (GA), Support Vector Machines (SVM), etc. The Artificial Neutral Networks (ANN) used in previous studies, include Fuzzy Neural Networks (FuzzyNN), Self-organising Maps (SOM), General Regression Neural Networks (GRNN), Hopfield Neural Networks (HopfieldNN), and Multilayer-Perceptrons. Since these networks either have fewer than two hidden layers, or do not demonstrate the characteristic of hierarchical feature learning (see Section 1.3), this thesis subsumes them into the category of classic machine learning. A summary of data preprocessing and machine learning modelling see Table 1.4.

1.5 Open Questions

After viewing typical difficulties in ERW (Section 1.2.3), challenges and solutions of machine learning in manufacturing and in ERW (Section 1.4), this thesis summarises **Open Questions (OQ)** below.

- *OQ 1. Insufficient data problem.* In order to build a all-covering quality monitoring system, data collection is crucial. Data collected from production however often lack the most important quality indicator, e.g. the diameters, due to expensive measuring methods. Data collected from laboratory experiments can have the quality indicators, but the amount is limited compared to production data, and cannot fully cover the various production conditions exhaustively. Furthermore, some important features such as temperature, force, displacement, are only available in laboratory. Previous studies have very limited addressed the insufficient data problem. The questions remain

Table 1.4: An overview of related works of machine learning in ERW, Part 2. Explanations of the acronyms see Section 1.4.4 or Table of Acronyms and Symbols.

Citation	Features & Preprocessing	Methods
[38]	TS: Re; TSFE: DKFE (slope, max, std, mxpo)	MLP
[124]	SF: F, t, I; TS: F, Re, s; TSFE: subsampling; FE: PCA, rms	MLP
[125]	TS: Re, s; TSFE: subsampling, DKFE (slope, mxpo, range).	Fuzzy NN
[126]	TS: Re; TSFE: DKFE (max, slope, min, std), stats	LR, LogR, MLP
[127]	Image from SAM; ImageEF: dilation, quantisation	MLP
[128]	TS: Re; TSFE: TS to bipolarised image	HopfieldNN
[56]	TS: I, F, U; TSFE: means of segmented TS	SOM
[129]	TS: I, F, U; TSFE: histogram, discretisation of quartiles	BN
[57]	TS: s; TSFE: subsampling, slopes	LVQ, MLP
[130]	TS: U, I, Re, s, F; TSFE: subsampling, std	LVQ
[131]	TS: I, U; TSFE: transitions, segmentation mean; FE: PCA.	LDA, QDA, MD, kNN, LVQ
[132]	SF: I, F, t, thickness	GRNN, GA
[121]	SF: I, t, thickness	MLP, GA
[133]	TS: I, U; TSFE: segmentation mean	kNN, SOM
[134]	TS: ultrasonic oscillogram; TSFE: DKFE (heights & distances of echoes)	MLP
[135]	TS: R; TSFE: segmentation, DKFE (max, min, mean, std, range, rms, slope); FS: power of the test	LVQ
[136]	TS: I, U; TSFE: transitions, segmentation mean; FE: PCA; FS: SFS, SBS, SFFS, SBFS, nBest	kNN
[137]	SF: t, I, F	LR, MLP
[36]	SF: Thickness, I, F, t; TS: Re; TSFE: subsampling	MLP
[122]	SF: t, I, F; FE: polynomial features	MLP
[138]	TS: s; TSFE: DKFE (geometric features)	MLP
[139]	SF: I, welding time, holding time, squeezing time; FS: ANOVA	LR, GA
[140]	SF: F, t, I	MLP
[141]	SF: coating, F; TS: Power; TSFE: DKFE (mxpo, max, drop)	LogisticR
[58]	TS: s; TSFE: segmentation, mean, radar chart, geometric features	DT ID3
[142]	TS: I, U, T; TSFE: derivative, segmentation, scale-space filtering, geometric features, stats	LR, M5 model tree [150], M5Rules, RF, SVM, kNN
[143]	SF: t, I, Pressure	PolyR, MLP, GA
[144]	SF: F, I, t, TS: Re; TSFE: PCA	MLP
[145]	TS: Re; TSFE: PCA	LR
[146]	TS: Re; TSFE: PCA	PolyR
[147]	SF: I, t, Pressure	PSO, KELM
[148]	TS: s; TSFE: DKFE stats, Chernoff images, binary features	HopfieldNN
[123]	SF: material, thickness, t, coating, I	SWRL, DT, CART

whether the available data amount is sufficient, and whether further features are necessary to be collected from production.

- *OQ 2. Standardised process description, data description and management.* Machine learning projects in manufacturing involve multiple parties with distinct knowledge background, e.g. process experts, measurement experts, data scientists, managers. A standardised process description is needed to ease the communication between these experts. Based on the process description, the data also requires to be modelled in a standardised way, to facilitate the data management of different data sources and processes.
- *OQ 3. Dataset evaluation for identifying conspicuity.* Large volumes of data are generated from manufacturing processes like electric resistance welding. It is very desired to gain a quick overview of the data collected from all welding machines, to evaluate the collected datasets and identify which welding machine, which welding program or which dress cycle are conspicuous and may be subject to risks of quality failures.
- *OQ 4. Quality monitoring with temporal structure and probabilistic estimation.* Previous studies estimated quality monitoring as single values for each welding operation independently. In fact, these welding operations comprise a welding process with temporal structures caused by the continuous wearing effect of the electrodes. The estimation of quality indicators should be a probabilistic distribution.
- *OQ 5. Evaluation and selection of machine learning methods.* Various machine learning methods have been experimented for ERW and Manufacturing. Yet it remains unclear, when and how to use which methods, and how to evaluate the machine learning models.
- *OQ 6. Integration of domain knowledge problem.* Previous studies have used domain knowledge to some degree in feature engineering,

but there exists still improvement space for more systematic discussions of the role of domain knowledge in machine learning, such as what is the best role of domain knowledge in machine learning, how the influence of domain knowledge can be intensified, etc.

- *OQ 7. Concept drift monitoring.* When the production conditions change during the production process, the developed methods for quality monitoring could fail, because the data under the changed conditions differs for the data collected for model training. A mechanism is needed to detect the data drift and adapt the model accordingly.
- *OQ 8. Transfer of developed ML approaches to other datasets and processes.* As mentioned in Section 1.1, there exists no machine learning method that can perform technologically meaningful information processing from unstructured datasets. Manufacturing datasets with different structures can be generated frequently from different data sources, i.e. operation conditions, machines, plants. Little work has been done in ERW to make machine learning methods scalable to different sources, versions of datasets, and to different manufacturing processes.

1.6 Objectives and Thesis Outline

The aim of the Ph.D. study is to study and develop effective and generalisable machine learning frameworks, for quality monitoring in electrical resistance welding, i.e. assessment and prediction of the quality indicators. In an attempt to achieve this central goal, this work has the following sub-objectives.

- Discussing the central goal of quality monitoring and studying the particularities of data of electrical resistance welding in depth;

- Exploring, developing and organising the machine learning methods and other methods for handling and modelling the data;
- Implementing software for ML analysis and data handling;
- Applying, validating and demonstrating the methods on industrial datasets in use cases.

This work is organised as follows. Chapter 2 proposes a ML framework that attempts a comprehensive coverage for collecting, understanding, preparing, preprocessing, and modelling the data in ERW and elaborates the methodological details. The framework is a result of deep intertwining of ML methods, domain knowledge of ERW, and semantic technologies. Chapter 3 describes one implementation of the methodologies in this work on two programming platforms, Python and SciXMiner (MATLAB). Chapter 4 selects six industrial use cases to demonstrate and validate the application of methodologies, to evaluate their performances and test their transferability. Chapter 5 summarises the thesis, lists the contribution, concludes the work and previews the outlook.

2 A Framework of Machine Learning in Electric Resistance Welding

In developing methodologies of machine learning in ERW, it often reveals that the development of new algorithms are not the most urgent tasks. Machine learning approaches are not able to retrieve meaningful information to support decision making from unstructured and not-understood datasets. It is not trivial to use and adapt the existent algorithms in the field of ERW as well as manufacturing. This work will therefore focus more on identification of a suitable framework that makes the machine learning approach effective for the datasets we have understood once, and scalable for other datasets with similar structures. This work will go across the boundaries of data science, understanding data characteristics of ERW from a combined view of data science, domain knowledge of engineering, and semantic technologies, revealing some aspects that are limitedly discussed in previous studies.

2.1 Question Definition Revisited

This section discusses questions resulting from data characteristics in ERW as well as in manufacturing. The questions are addressed in the procedures of machine learning workflow (Section 1.3): data collection, data preparation, data preprocessing and machine learning modelling.

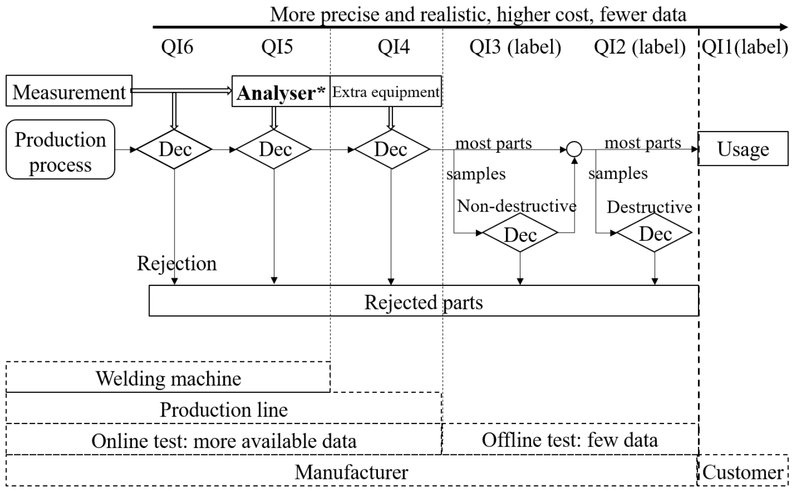


Figure 2.1: Quality indicators of different levels. "Dec" stands for decision. The asterisk indicates data-driven methods for estimating quality indicators, which is the central goal of this thesis, that can be seen as online analysers. The optimal target (labels) of machine learning analysis are usually QI1 to QI3.

2.1.1 Quality Indicators of Different Fidelity Levels and Accessibility

The central question of the thesis is to estimate quality indicators. Various quality indicators have been discussed in the literature [151, 152, 153], but they have not been systematically organised or are not applicable to ERW [154, 155, 156, 157]. This thesis groups them into different levels according to their fidelity and accessibility. Figure 2.1 illustrates these levels based on the workflow of quality control.

After a welding process, the welded part will go through a series of quality control gates. At each gate, a rejection decision can be made according to the corresponding judgement based on the respective quality indicators.

- **QI1** is the "final" quality indicator of a product, the optimal label to be predicted in machine learning. This can be e.g. the lifespan of a product before quality failure. However, QI1 is usually unavailable, unless extremely huge effort is spent on measuring this indicator, e.g. collecting voluminous historical usage data from customers.

- **QI2** and **QI3** stand for quality indicators that can only be measured offline, i.e. after the production process, and are therefore only measured sample-wise due to cost reasons. QI3 are those measured using non-destructive methods, e.g. ultrasonic test results or X-ray test results, QI2 are those that can only be measured using destructive methods, e.g. spot diameters in RSW, tensile shear strength in RSW and HS. Since QI2 is already relatively precise and much more available than QI1, QI2 is established as a standard quality indicator [32, 34]. QI2 and QI3 are also usual target (labels) for ML prediction.
- **QI4** are the quality indicators that can also be measured online in production, but require extra equipment, which are usually even more expensive than the welding machines themselves. An example would be the contact resistance between the welded wire and hook (called as the GDG-resistance [31]) in Hot-staking (HS).
- **QI5** and **QI6** are quality indicators that can be measured or calculated online by the machine software systems. QI6 indicates the direct measured physical quantities, e.g. current, resistance, force, etc. QI6 is replaced by QI5, if QI5 exists. QI5 represents the calculated quality indicators developed with process know-how. QI5 functions as online analyser during the production process. Examples of QI5 include the *Q-Value*, spatter occurring time, the process stability factor in the Bosch Rexroth welding system [31].

The goal of all data-driven methods can be seen as to build an offline or online analyser to estimate an improved QI5 quality indicator, to save QI1 to Q4 as a long-term goal. This thesis treads the starting steps towards the long-term goal by proposing the framework (Chapter 2), studying QI5 that

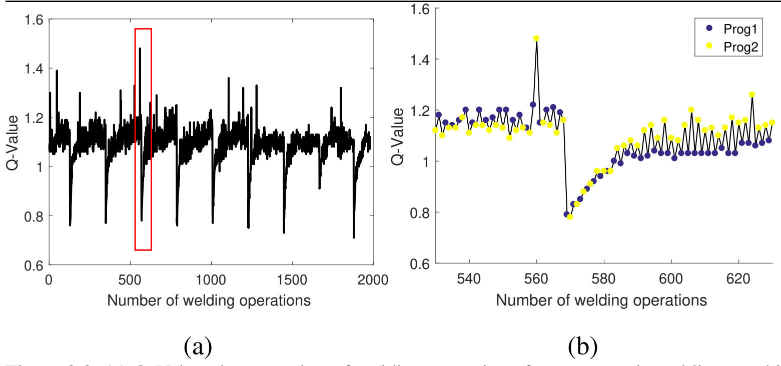


Figure 2.2: (a) Q-Value along number of welding operations for an example welding machine. The red rectangle indicates the area for a closer look in (b).
(b) Welding operations performed with different welding programs often possess different dynamics, e.g. the *mean* of Q-Values are different

replaces QI2 with simulation data (Use Case 4.1), and forecasting quality indicators (Use Case 4.3), etc.

2.1.2 Multi-levels of Temporal Structures in Data

Previous studies have treated each welding operation independently (Section 1.4.4). If we closely examine the data, e.g. Figure 2.2 showing the Q-Value along the number of welding operations for an example welding machine, we can see clearly that the data has relatively strong periodicity, which indicates the data very likely have temporal dependencies. With the example dataset of RSW, this section elaborates the multi-levels of temporal structures in data, which is an intrinsic result of the structure of production processes.

The first time level is the **welding time level** during a single welding operation, which usually takes several hundreds of milliseconds (Figure 2.3a). From each welding operation, data of single features and process curves are recorded (Figure 2.3b). The consecutive welding operations constitute the second time level, the **welding operation level**. All process curves on the welding time level for one single welding operation (including e.g. the process curves in Figure 2.3a) are aggregated onto one time step on the welding operation level (Figure 2.3b). These operations are controlled by the adaptive control system and are operated according to welding programs. A closer look at a small area reveals that the welding programs of the operations are arranged with specific orders prescribed by the production schedule (Figure 2.2b). These operations with each welding program form the **welding program level**. As the welding process goes on, the electrode wears. Since there exists no available feature that reflects the wearing effect in a physically meaningful way, the wearing effect is quantified using the single feature WearCount (Figure 2.5).

Table 2.1: Correspondence table of temporal structure features ($\mathbf{F}_t$) and time levels

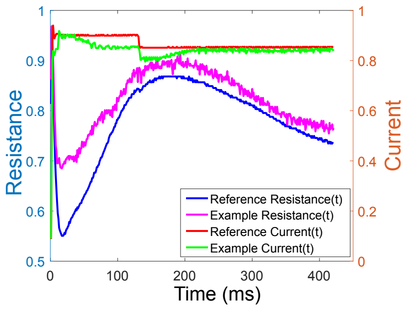
$\mathbf{F}_t$	Symbols	Time level
TimeStamp	t	Welding time level
WearCount	N_{WC}	Welding operation level
DressCount	N_{DC}	Dress cycle level
ElectrodeCount	N_{EC}	Electrode cycle level
ProgramNumber	$ProgNo$	Welding program level
MachineID	-	Machine level

A regular welding process can include welding, dressing, and short-circuit measurements before and after dressing (Figure 2.4). A complete dressing-welding-dressing procedure forms a *Dress Cycle*. The wearing effect repeats in each dress cycle. The consecutive dress cycles form the **dress cycle level**. According to the domain expert, the periodicity of the Q-Value is caused by the wearing effect. The Q-Values in Figure 2.2 begin with small values at each start of dress cycle, rises as the electrode wears, and normally reaches stable conditions at the end of the dress cycle. After a certain number of dress cycles, the electrode needs to be changed (for RSW the electrode cap is changed). The welding dynamics is thus influenced by the new electrode. All operations welded by one electrode constitute an *Electrode Cycle*. The consecutive electrode cycles comprise the **electrode cycle level**. Note that for some manufacturing processes, some levels are optional. For example, the *dress cycle* level does not exist for HS, since in HS the electrode will be directly changed instead of being dressed. Table 2.1 lists the correspondence between temporal structure features ($\mathbf{F}_t$) and time levels.

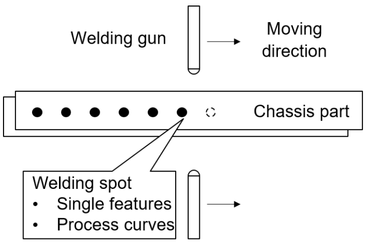
Until here, the time levels of a single welding machine are explained. A step further to see the multiple machines organised in production lines reveals two typical structures of production lines. Figure 2.6 schematically illustrates a production line for RSW with a sequential structure, while Figure 2.7 schematically illustrates a production line for Hot-staking with a parallel structure. These organisations of welding machines constitute the **machine level**. Depending on the structures of production lines, the collected data points need to be organised with different temporal structures.

Other time-levels, including the level of car bodies, production batches of car bodies and machines, suppliers of the car bodies and machines, are also important, but this information is currently not available in the data studied by this work. They will not be addressed in this work.

Further levels above the production lines are the levels of plant locations, and different original equipment manufacturers (OEM). These levels are no longer temporal levels and are normally treated independently.



(a)



(b)

Figure 2.3: (a) Examples of process curves. The adaptive control will try to force the process curves follow the reference curves defined by the welding program. Left y-axis: Resistance (anonymised), right y-axis: Current (anonymised), x-axis: Samples
(b) Consecutive welding operations constitute the welding operation level. Data of single features on the welding operation level and process curves on the welding time level (Section 2.1.2) are recorded by the machine software system.

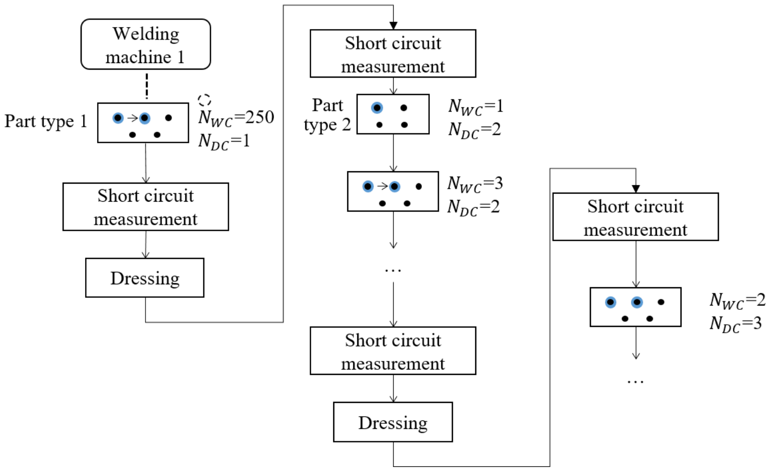


Figure 2.4: Schematic illustration of a welding process from the perspective of one machine, including maintenance of a welding machine in RSW. N_{WC} stands for WearCount, N_{DC} stands for DressCount. Before and after each dressing operation, short-circuit measurements (optional) are performed to monitor the electrode status. The machine is responsible for several designated spots on each chassis part (marked with blue circles).

In data analysis, a data instance is an atomic object. Depending on the time level of analysis, a data instance can be a welding operation, an electrode dress cycle, a welding program, a welding machine, etc.

2.1.3 Insufficient Data Problem and Three Data Challenges

As mentioned in Section 1.2.3 and 1.4, data acquisition is difficult in manufacturing in general as well as in ERW, especially the labelled data (QI1-QI3). Most of the previous studies have mentioned that their data were collected from laboratory experiments. The laboratory data and the models trained on them are different than that of the real production, as the welding conditions (such as cooling time and wearing effect) are usually different. Apart from the labelled data amount problem, other difficulties exist for machine learning in ERW. This work summarises these difficulties as three *Data Challenges*.

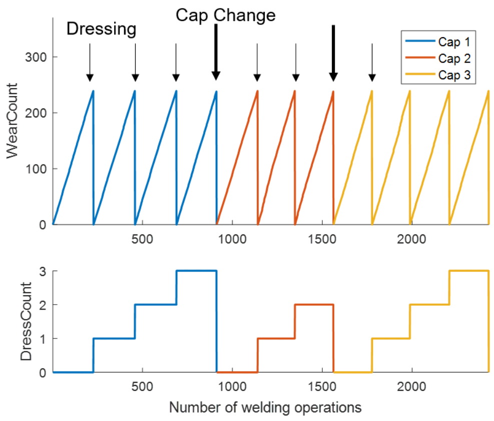


Figure 2.5: Example of temporal structures in the RSW data quantified by WearCount, DressCount, and CapCount. WearCount is a counter on the welding operation level. It increases by one after one welding operation, and is reset to zero after a dressing is performed, or the cap is changed. DressCount is a counter on dress cycle level. It increases by one after one dressing operation, and is reset to zero after the cap is changed.

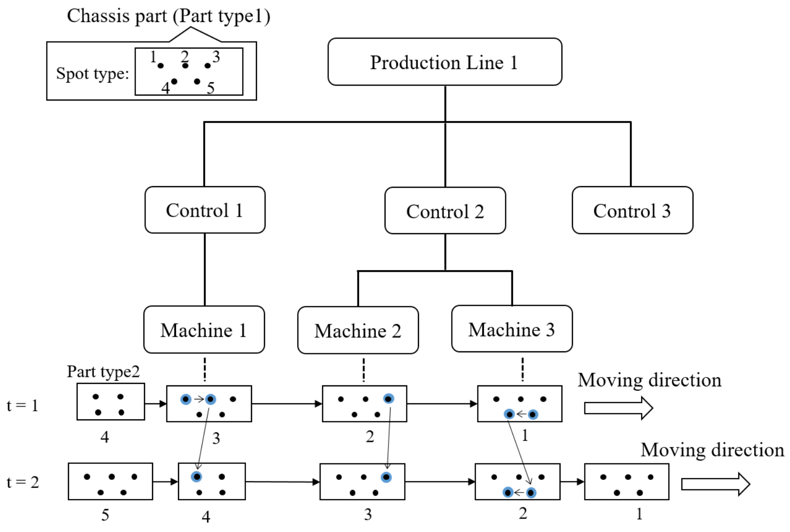


Figure 2.6: Sequential structure of a simplified RSW production line, where multiple types of car chassis part, each with a certain amount of welding spots with specified types, go through a sequence of welding machines. Each machine is responsible for several designated spots (marked with blue circles). Each spot type is welded with one pre-defined welding program by one machine in a fixed order.

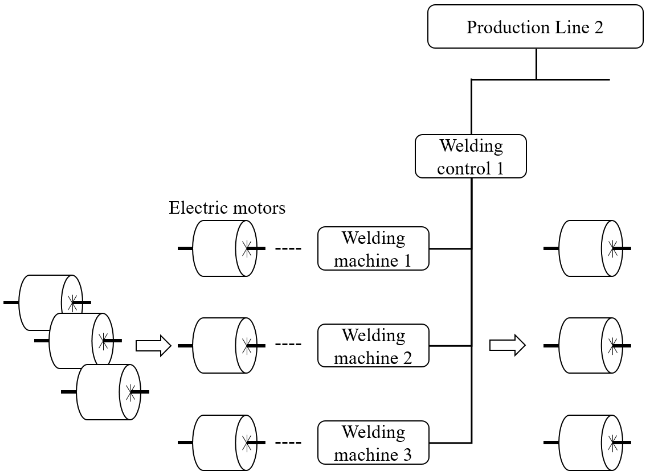


Figure 2.7: Parallel structure of a simplified hot-staking production line, where several electric motors are welded by several welding machines simultaneously, and these welding machines can be controlled by one welding controls system.

Challenge 1 is the limit on labelled data amount, which results from the difficulty in collecting labelled data in production or in the laboratory.

Challenge 2 is the limit on features. Additional sensors, e.g. electrode displacement, or extra measurements, such as actual worksheet thickness or material properties, may be necessary for reliable quality prediction in manufacturing processes. However, more sensors mean higher costs, change of equipment design, larger installation space, increased risks of expensive machine stops caused by sensor failures, etc. The installation of some sensors are even physically infeasible. When deciding whether to install an extra sensor, its benefit needs to outweigh the disadvantages. It is therefore important to understand which sensors or measurements (or features in data science) are necessary and to quantify their benefit to justify the higher costs.

Challenge 3 is the limit on coverage of relevant situations. Quality failures are very unusual in manufacturing data, because the welding quality is good under normal conditions. Moreover, quality failures may be of different types, making it even more difficult to collect sufficient data for reliable quality prediction.

These three data challenges can be interpreted as concrete questions:

- Questions to Challenge 1: How much labelled data should be collected?
- Questions to Challenge 2: Which sensors should be installed? Which precision level should the measurements have?
- Questions to Challenge 3: What production conditions are meaningful to study? Are they covered in the available data?

The methods to address the three Data Challenges are proposed in Section 2.2.4.

2.1.4 Multi-fidelity Data Model and Application Questions

Manufacturing data are collected from three typical sources, the production plants at OEMs, the laboratories for process development, and simulation in research centre for a better understanding of the process. In comparison to literature, where almost only laboratory data are analysed and the different data sources are not separately addressed, this work proposes the multi-fidelity data model (Figure 2.8) to cover these three data sources. This section will discuss the three sources in aspects of data properties and possible application questions. The three data sources have different properties (Table 2.2), and therefore can be potentially combined to compensate their respective disadvantages.

Properties of the three data sources. The ultimate goal of any process monitoring, development or diagnostics is to improve the actual production process and produced goods. The *production data (prod)* collected from plants are the most realistic data. Yet the disadvantages are obvious. Production data are provided *as is*. As a by-product of the produced goods, they can be constantly collected but their conditions (normally) cannot be specified or designed. The available features from production are limited to the already implemented sensors. The cost to collect labels from production

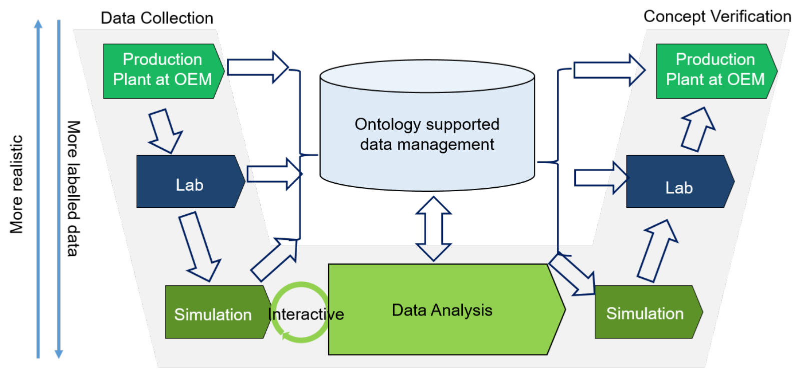


Figure 2.8: Multi-fidelity data model. Each data source can have multiple variants due to e.g. customer individualisation for production data, various experimental settings for lab data or simulation data.

data is enormous, considering that the produced goods, e.g. chassis, needs to be destroyed. Only very few labelled data are collected.

The *laboratory data (lab)*, collected from laboratory experiments or production lines under experimenting settings, can be almost very authentic to production data, if the setting is exactly tailored to mimic the real production conditions. There still exist some differences from the production condition, e.g. normally worksheets are used instead of real chassis. The cost is therefore lower than that in production. The degree of freedom of installing sensors in the laboratory is much higher than that in the production, but is still limited due to physical reasons, e.g. temperature at the welding spot can never be measured.

Simulation data (sim) of physical reality (also referred to as "digital twin" [95]) are very advantageous because there exist almost no limitation in installing "sensors", and the costs of data collection are only computation, once a good simulation model has been established. The more similar the simulation settings are compared to that in the production, the more valuable simulation data can be. The costs come from the preliminary effort to build a realistic and robust simulation model.

Table 2.2: Properties of three data sources

Data source	Veracity	Degree of freedom	Cost
Production (unlabelled)	High	Low	Low
Production (labelled)	High	Low	High
Laboratory	Relatively high	Medium	Medium
Simulation	Relatively high	High	Medium

Application questions. Based on the properties of different data sources in the multi-fidelity data model, many concrete *Application Questions (AQ)* are worthy to study. For some of them, this work will give concrete use cases in Chapter 4. For the others, this work will only limitedly address them and leave them for future research. An overview of addressed AQs, their corresponding Open Questions (Section 1.5) and their use cases (Chapter 4) is listed in Table 2.3. These AQs categorised below in two groups:

Analysis of one data source, including:

- *AQ 1. Dataset evaluation* to quickly gain an overview of datasets, for the purpose of identifying conspicuous welding machines, welding programs, dress cycles, outliers from huge datasets collected from various welding machines, and pointing out interesting datasets for further investigation.
- *AQ 2. Data amount analysis* to assess minimal necessary data amount for effective quality monitoring with machine learning for guiding data collection and saving costs.
- *AQ 3. Feature evaluation* to evaluate importance and benefits of features collected from sensors or measurements or generated in data preprocessing, for understanding process, analysing root causes of failures, guiding data collection, etc.
- *AQ 4. Classification and regression analysis for estimating* the value of current quality indicators using data-driven methods.
- *AQ 5. Classification and regression analysis for predicting* the value of future quality indicators with historical data.

- *AQ 6. Process parameter optimisation* using algorithms such as Bayesian optimisation and reinforcement learning, which are usually built upon the classification / regression models for AQ4 and AQ5.
- *AQ 7. Semi-supervised learning* for predicting partially labelled data, e.g. using a large amount of unlabelled production data to train an auto-encoder and then using a small amount of labelled data to train a decoder.

Analysis across data sources, including:

- *AQ 8. Similarity analysis* between datasets to determine how similar or dissimilar two datasets are, for determining the fidelity of a data source and guiding data collection. E.g., similarity analysis between simulation dataset and laboratory dataset can reveal whether the simulation scenario is designed sufficiently realistic as the laboratory data so as to determine whether simulation data can be used for process understanding, etc.
- *AQ 9. Interactive data acquisition and analysis* by analysing a data source with fewer data to guide collection of another data source with more data amount, for example, analysing the laboratory data to help with designing simulation scenarios, then determining similarity between the two datasets, then designing new simulation scenarios.
- *AQ 10. Transferability* of models trained on one dataset to other datasets, or trained on one data source (typically with more data and lower costs, e.g. simulation data) to other data sources (typically with less data amount and higher costs, e.g. production data).

2.2 Data Collection: Data Acquisition and Evaluation

This work addresses the data collection topic with the following aspects: (2.2.1) simulation-supported interactive data acquisition and analysis; (2.2.2) data similarity analysis; (2.2.3) inverse modelling of simulation model;

Table 2.3: Overview of application questions (AQ), their corresponding open questions (OQ) (Section 1.5), method sections (M), and use cases (Chapter 4). Some method sections are partially evaluated and their use cases are not presented in this thesis due to space limit.

AQ	OQ	Methods	Use case
AQ2, AQ3, AQ4	OQ1, OQ5	M2.1.1, M2.1.2, M2.1.3, M2.2.4, M2.4.2, M2.4.5, M2.4.8	Use Case 4.1
AQ1	OQ3, OQ4	M2.2.5	Use Case 4.2
AQ3, AQ5	OQ4, OQ5, OQ6	M2.1.1, M2.1.2, M2.4.1, M2.4.2, M2.4.7	Use Case 4.3
AQ3, AQ5	OQ4, OQ5, OQ6	M2.1.1, M2.1.2, M2.2.4, M2.4.1, M2.4.2, M2.4.5, M2.4.3,	Use Case 4.4
AQ5, AQ10	OQ4, OQ8	M2.1.1, M2.4.5	Use Case 4.5
AQ2, AQ6	OQ1, OQ5, OQ8	M2.1.3, M2.2.3, M2.2.4	Use Case 4.6
-	OQ2	M2.3.1, M2.3.2	-
-	OQ5, OQ6, OQ8	M2.3.3, M2.4.6	-

(2.2.4) determining necessary data amount for ML modelling; (2.2.5) identifying conspicuous data areas for a further analysis.

2.2.1 Overview: Simulation-supported Interactive Data Acquisition and Analysis

The interactive data acquisition and analysis relies on generating data of user-defined scenarios by physics-based simulation with a verified Finite Element Method (FEM) model [158].

Simulation-supported data collection. The FEM simulation models mechanical effects (elastic-plastic deformation and thermal expansion), thermal effects (temperature changes and heat transfer) and electrical effects (change of electric current density and electric potential field). Non-linear changes of material and contact properties, such as electrical and thermal contact conductivity, are also taken into account. Strong interactions between all three fields result in very dynamic process behaviour and require a multi-field coupled simulation. The FEM models should be verified using data from laboratory tests or production lines under controlled conditions. An established simulation model has four advantages:

- Each simulation run produces one labelled data point. Running a lot of simulations can thus offer a large amount of labelled data.
- There is little limit for obtaining features measured by sensors that are costly or difficult to install in the laboratory or production.
- By designing the simulation scenarios astutely, rare situations in production can be studied in detail.
- There are no precision issues of sensor measurements that are present in the laboratory or in production.

The main effort of the FEM simulation lies in two points: (1) gathering relevant information as simulation inputs, such as worksheet and electrode geometries, temperature-dependent material data, contact properties, etc.; (2) setting up a parametric Finite Element Model in an automated simulation loop, including simulation preparation and data extraction, using commercial and open source tools, such as Python, software packages for Finite Element Method, etc.

Interactive data analysis and collection. The simulation supported data collection follow the six steps iteratively (Figure 2.9).

1. Data collection from laboratory or production.
2. Initial simulation scenario definition.
3. Data collection from simulation.
4. Similarity analysis between simulation data and laboratory/production data.
5. Data amount and feature evaluation.
6. New simulation scenario definition.

2.2.2 Similarity Analysis between Datasets

Dataset similarity analysis is to answer the question of how similar two datasets are. A deep understanding of this topic requires rigorous mathematical deduction. In the literature, various methods have been proposed,

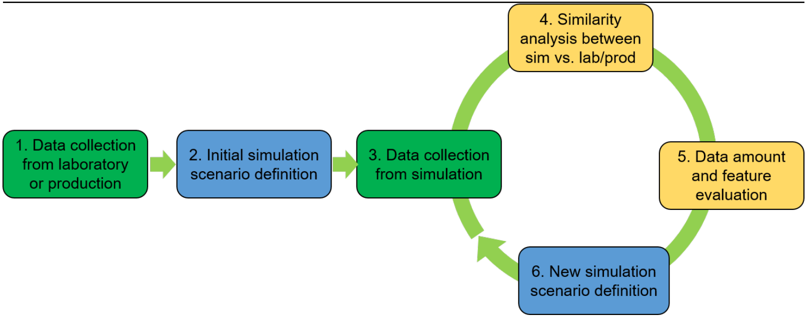

1. Data collection from laboratory or production
2. Initial simulation scenario definition
3. Data collection from simulation
4. Similarity analysis between sim vs. lab/prod
5. Data amount and feature evaluation
6. New simulation scenario definition

Figure 2.9: Simulation-supported interactive data analysis and collection

e.g. similarity measures for one-dimensional datasets [159], an effective statistical evaluation of genomic dataset similarity [160], measuring similarity between homogeneous datasets for distributed data mining [161], similarity of condensed data models (e.g. for check data anonymisation) across high-dimensional datasets [162]. However, similarity comparison methods for ERW or manufacturing have been limitedly discussed in the literature. No method can be directly applied.

This work suggests some practical concepts and methods for similarity analysis of ERW datasets, for which there exist many important application examples: (1) comparing simulation data with laboratory/production data to test veracity of simulation data; (2) comparing the dataset collected for model development with another dataset collected for model deployment to test model deployability; (3) comparing datasets collected from different machines, production lines, welding conditions to test transferability. If the simulation model is realistic enough, then it can be used to generate more labelled data, for providing engineering insights, or substitute the costly lab data or production data. The pre-requisite for similarity analysis is that the two datasets should share a common subset that contains the same features representing the same physical meanings. The term **dataset similarity** can be divided into two aspects:

- *Mechanism*, whether the change of features in two datasets follow the same patterns.

- *Distribution*, whether the data points in two datasets have similar value distributions and lie in similar operating areas, defined by statistic properties e.g. amplitudes, changing range, etc.

Mechanism similarity analysis. The analysis of mechanism similarity has two levels. The first level is to identify whether the features in two datasets follow the same correlation patterns, i.e. whether the correlation relationships between features of two datasets are similar (Figure 2.10a). The second level is to figure out whether the causality patterns of two datasets are similar (Figure 2.10b).

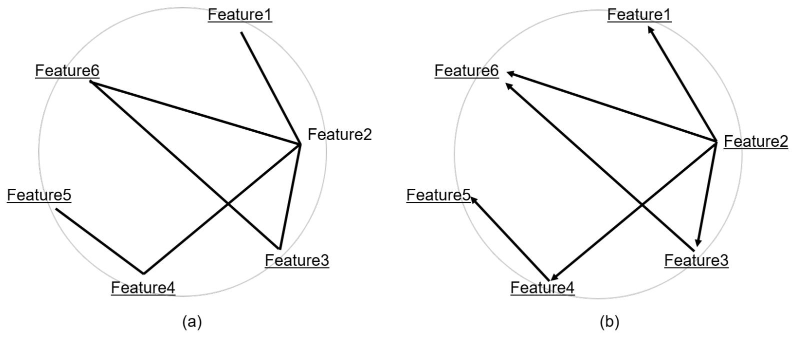


Figure 2.10: Correlation pattern is generated by drawing lines between pairs of features whose correlation coefficients are greater than a defined threshold.

The *correlation pattern* [163, 164] is to calculate correlation matrix using e.g. Pearson coefficient [165]. the *causality pattern* is to calculate pair-wise transfer entropy matrix [166, 163].

Distribution similarity analysis. The analysis of distribution is to compare the statistic properties of the two datasets, including mean, median, standard deviation, range, maximum, minimum, ratio between important features selected using domain knowledge, etc.

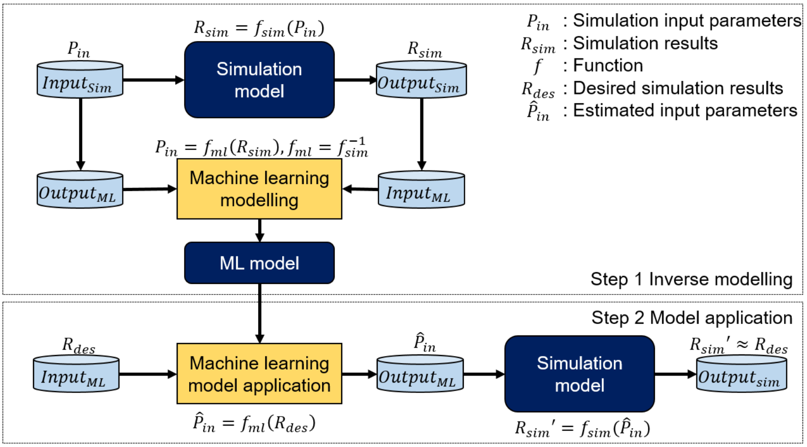

P_{in} : Simulation input parameters
R_{sim} : Simulation results
f : Function
R_{des} : Desired simulation results
$\hat{P}_{in}$: Estimated input parameters
$R_{sim} = f_{sim}(P_{in})$
P_{in}
$Input_{Sim}$
Simulation model
R_{sim}
$Output_{Sim}$
$P_{in} = f_{ml}(R_{sim}), f_{ml} = f_{sim}^{-1}$
$Output_{ML}$
Machine learning modelling
$Input_{ML}$
ML model
Step 1 Inverse modelling
Step 2 Model application
R_{des}
$Input_{ML}$
Machine learning model application
$\hat{P}_{in}$
$Output_{ML}$
Simulation model
$R_{sim}' \approx R_{des}$
$Output_{sim}$
$\hat{P}_{in} = f_{ml}(R_{des})$
$R_{sim}' = f_{sim}(\hat{P}_{in})$

Figure 2.11: ML for inverse modelling of a simulation model

2.2.3 Inverse Modelling of the Simulation Model

Inverse modelling aims at facilitating the fine-tuning of the simulation model so that the simulation data become more similar to the laboratory data or production data.

Inverse modelling has two steps. Step 1 is to train a ML model with simulation data. For the training, simulation results (R_{sim}) serve as the inputs (Figure 2.11), and simulation input parameters (P_{in}) serve as the outputs of ML modelling. In this way, the ML model mimics an inverse model of the simulation model ($f_{ml} = f_{sim}^{-1}$). Step 2 is to feed the desired simulation results (R_{des}), which can be laboratory data or production data, as inputs to the ML model. The outputs of ML model are then the estimated simulation input parameters ($\hat{P}_{in}$), which should generate (R_{des}). After that, these $\hat{P}_{in}$ are delivered to the simulation model, whose results (R'_{sim}) should be very similar to the desired results (R_{des}).

For the case when multiple sets of simulation input parameters (P_{in}) can produce the same simulation results (R_{des}), the ML model may output several sets of estimated simulation input parameters (P'_{in}) in case of multiple times of application. From these estimated input parameters, one can be chosen based on manual preferences, or some performance metrics, like higher similarity, less simulation time, etc.

2.2.4 Data Amount and Feature Sets Analysis

This section aims at addressing the three data challenges in Section 2.1.3. The approaches centre on training machine learning models with different data subsets and comparing their performance.

Analysis of training data numbers. Data can be split into several subsets of different training data numbers for answering the question: How much labelled data should be collected (Data Challenge 1). A series of training subsets is built by randomly selecting a different number of data points from the training dataset, and the larger training subsets always contain the smaller ones. To allow direct comparison of the testing results, the test dataset always contains the same data points. ML models are trained with these subsets of different training data numbers, and tested on the same test set (Table 2.4).

Table 2.4: Data splitting to subsets of different training data size. D_{tr1} stands for training set 1, $D_{tr,all}$ stands for training set with all data, and D_{tst} stands for test set, where $D_{tr1} \subset D_{tr2} \subset D_{tr3} \subset ... \subset D_{tr,all}$.

Training data	D_{tr1}	D_{tr2}	D_{tr3}	...	$D_{tr,all}$
Test data	D_{tst}	D_{tst}	D_{tst}	...	D_{tst}

Analysis of feature sets. Data can be split into several subsets of different features for answering the question: Which features are important (Data Challenge 2). All available features, i.e. time series features as well as single features, are divided into four subsets (Table 2.5). The feature importance is evaluated by comparing the performance of ML models trained with the four different feature subsets.

Analysis of noisy sets. Data can be augmented with additional noise to answer the question: Which precision level should the sensors have (Data Challenge 2)? Adding noise to the aforementioned non-noisy subsets of

Table 2.5: Data splitting into four subsets of different features

Feature set	Description
$FeatSet_{prod}$	Features that are always available in production.
$FeatSet_{lablow}$	$FeatSet_{prod}$ + features available in laboratory with relatively low cost.
$FeatSet_{labhigh}$	$FeatSet_{lablow}$ + features available in laboratory with higher cost.
$FeatSet_{all}$	$FeatSet_{labhigh}$ + other features in simulation that are difficult or extremely costly to realise in reality.

different training data and feature sets results in a series of subsets with noise. The way of adding noise and the noise levels can be determined collectively with process and measurement experts for single features and time series.

- For time series, the noise should be added for every single sample point.
- For single features, the noise should be added for each feature.

To address questions regarding Challenge 3 requires particular simulation scenarios defined by the process experts, which can be hardly achieved by pure data analysis using existent data.

2.2.5 Evaluation of Datasets

The manufacturing industry produces a huge volume of data. Considering the number of cars produced every day, each car body with 3000 - 6000 welding spots, the amount of data generated even only from one plant of one OEM is enormous. It is impossible to analyse all data in detail in ML development as well as in application. Some methods are needed to evaluate the data and select the most interesting areas, where quality failures are more likely to happen, or where the behaviour of quality indicators is unstable. If any data area is deemed to constitute an abstract "dataset", identification of *interesting data areas* is to perform *dataset evaluation*, or *anomalous dataset detection*. This is to evaluate the already collected data, and is therefore an offline analysis. Dataset evaluation targets on analysing a large dataset in these aspects: information summarisation and visualisation, identifying conspicuity, outlier detection, and subset selection for further analysis.

Decomposition of data behaviour. The behaviour of quality indicators is expected to be stable, e.g. the optimal Q-Values should be around the optimal value "1", but the actual Q-Values rise and drop in each dress cycle (Figure 2.2). Depending on the granularity of analysis, the behaviour of data can be decomposed into two or three components, as shown in Equation 2.1, where X is the raw data array with temporal structures, and $\mathbf{X}_{Trend}, \mathbf{X}_{Scattering}, \mathbf{X}_{Outlier}$ are defined as follows.

$$\mathbf{X} = \mathbf{X}_{Trend} + \mathbf{X}_{Scattering} (+\mathbf{X}_{Outlier}) \tag{2.1}$$

- *Trend*, denoted by $\mathbf{X}_{Trend}$, which describes the data behaviour excluding local variances. Important to note is the $\mathbf{X}_{Trend}$ here is to be understood as a "shape" of the data, not the general rising or dropping tendency of the time series. If $\mathbf{X}_{Trend}$ is to indicate the general tendency, Equation 2.1 can only hold if the time series contains no periodical components, as which has been extensively studied in energy forecasting [167]. Multiple ways to estimate the *Trend* will be elaborated later in this section. The decomposition of data behaviour can also be done by seasonal decomposition, since the data possess obvious seasonality (Figure 2.12). Yet the period of the data may vary from dataset to dataset due to user configuration of the welding system.
- *Scattering*, denoted by $\mathbf{X}_{Scattering}$, characterising the local statistic variances. It can be calculated by $\mathbf{X}_{Scattering} = \mathbf{X} - \mathbf{X}_{Trend}$.
- *Outliers*, denoted by $\mathbf{X}_{Outlier}$, are large local deviations from the Trend. They are a special type of scattering, and therefore are in an optional bracket in Equation 2.1. Outliers are caused by unusual local disturbances.

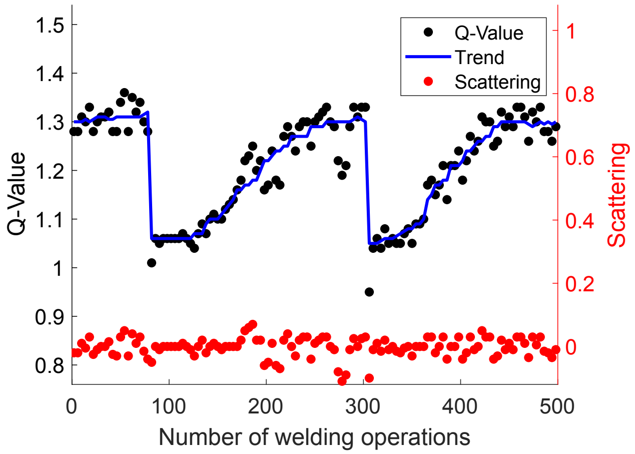


Figure 2.12: Q-Values, Trend, and Scattering across multiple dress cycles. The Trend of Q-Values usually resets to a low level at the start of each dress cycle, rises as wear effect intensifies, and reaches a stable level at the end of each dress cycle. The pattern of Scattering of Q-Values is not distinctively visible.

Detection of outliers. Outliers can be detected using a statistic method similar to the *Interquartile Range rule* [168]. $X[k] \in \mathbf{X}_{Outlier}$ if:

$$X[k] \leq Q_1 - 1.5 \times IQR \mid X[k] \geq Q_3 + 1.5 \times IQR \qquad (2.2)$$

where k indicates a time step of any temporal structure feature listed in Table 2.1, Q_1 stands for the first quartile, Q_3 stands for the third quartile, and $IQR = Q_3 - Q_1$. The calculation of quartiles depends on the method of Trend estimation (explained later in paragraphs of estimation of Trend).

This work also proposes an adjusted method for outlier detection, to take into account the “shape” of the data:

$$X[k] \leq X_{Trend}[k] - 2 \times IQR \mid X[k] \geq X_{Trend}[k] + 2 \times IQR \qquad (2.3)$$

in which $X_{Trend}[k]$ is used instead of Q_1 or Q_3, and therefore the coefficient of IQR is adjusted to 2 since $X_{Trend}[k]$ can be seen as approximately $Q_3 - Q_1$.

Estimation of Trend. Three types of methods are to estimate the Trend.

- *Local Trend*, which is obtained by using an acausal sliding window to calculate the moving average (median) of data points belonging to the same welding program (Equation 2.4, 2.5). The sliding window has a length of t_P, where P means only the points that belong to the program P are included in the window, and X_P denotes the sub-time-series of X that belong to the same Prog. The calculation can be acausal because this is an offline analysis. *Median filter* is less sensitive to local outliers compared to *mean filter*. If the local trend is used for outlier detection, the quartiles can be calculated within the sliding window.

$$X_{Trend}[k] = \frac{1}{t_P + 1} \sum_{i=k-t_P/2}^{k+t_P/2} (X_P[i]) \tag{2.4}$$

$$X_{Trend}[k] = median(X_P[k - t_P/2], ..., X_P[k + t_P/2]) \tag{2.5}$$

If the sliding window approaches the start or end of a dress cycle, the window should be shortened, only to include the data points within the same dress cycle, because the data behaviour of end of the previous dress cycle differs from the start of the next dress cycle significantly. This can be observed from data and reasoned from the wearing effect of electrodes using domain knowledge. It should be noticed, the selection of window length t_P is not trivial. Especially in case where a welding machine performs many welding programs, or where a dress cycle is very short because of e.g. manual interference, the data points belonging to the same welding program in one dress cycle become very few, the estimation of the Trend using very few points becomes unreliable. Appendix A.4 derives the minimal number of data points (often referred to as data tuples in this work to avoid confusion) required for a reliable estimation.

- *Global Trend* is an average (or median) of all Trends across all dress cycles (or electrode cycles). For the k-th point, the global trend $X_{Trend}[k]$ is calculated as the mean value (or median value) (Equation 2.6, 2.7) of all data points with the same WearCount (N_{WC_i}) as the k-th point (denoted as $N_{WC_i} = N_{WC_k}$). The calculation should be also performed for each welding program separately. The calculation of Q_1 and Q_3 can use the global data when the global trend is used for outlier detection.

$$X_{Trend}[k] = \frac{1}{N}\sum_{i}^{N}(X_P[i]), for\ all\ i\ where\ N_{WC_i} = N_{WC_k} \tag{2.6}$$

$$X_{Trend}[k] = median(X_P[i]), for\ all\ i\ where\ N_{WC_i} = N_{WC_k} \tag{2.7}$$

  N denotes the number of all points $X_P[i]$ whose $N_{WC_i} = N_{WC_k}$. This calculation method assumes that the Trends of all dress cycles (or electrode cycles) should be the same, as the different dress cycles (or electrode cycles) should be nominally identical, according to domain knowledge. This is partially correct, since the data behaviour in different dress cycles is somehow similar for some welding machines (Figure 2.2 and Figure 4.15), but more different for some other welding machines with change of production plan (Figure 4.20).

- *ML Model Trend* is to use the model prediction of the target quality indicator as the Trend. The ML models can be linear or non-linear, and should only take the temporal structure features ($\{\mathbf{F}_t\}$) as inputs (Equation 2.8, 2.9). Two examples are:

$$\mathbf{X}_{Trend} = LR(\{\mathbf{F}_t\}) \tag{2.8}$$

$$\mathbf{X}_{Trend} = MLP(\{\mathbf{F}_t\}). \tag{2.9}$$

Overall metrics and discrepancy metrics. The three components of data behaviour can be quantified and reduced to six metrics.

- The average value of Trend. This metric reflects the general level. The difference of this metric across dress cycles or between welding programs reflects the discrepancy between the dress cycles and welding programs: $Trend_{mean} = mean(Trend)$
- The degree of the change of Trend (the shape of the Trend is not addressed): $Trend_{range} = max(Trend) - min(Trend)$
- The average degree of Scattering: $Scattering_{mean} = mean(|Scattering|)$
- The degree of the change of Scattering: $Scattering_{range} = max(Scattering) - min(Scattering)$
- The number of outliers: $\#Outliers$
- The mean value of the outlier deviations: $OutlierDev_{mean} = mean(\{o_i - b_i\}), o_i \in Outliers, b_i \in Trend$

These six metrics can describe the data behaviour of each small data area, e.g. dress cycles, electrode cycles, sub-time-series of different programs. As can be seen in Figure 2.12, the data behaviour in different dress cycles of RSW (electrode cycle in case of HS) could slightly be different. According to domain knowledge, the degree of difference depends on the dressing operation, or electrode properties. To characterise the overall data behaviour and discrepancies between data areas, *overall metrics* are calculated as the mean values of the six metrics across data areas to evaluate the overall behaviour, and the *discrepancy metrics* are calculated as standard deviations to describe the difference across data areas. Table 2.6 gives details.

Table 2.6: Metrics to quantify data behaviour. std stands for standard deviation. N_t denotes a data area prescribed by a temporal structure feature in Table 2.1.

Overall Metric	Discrepancy Metric
$Trend_{mean}$	$std(Trend_{mean,N_t})$
$Trend_{range}$	$std(Trend_{range,N_t})$
$Scattering_{mean}$	$std(Scattering_{mean,N_t})$
$Scattering_{range}$	$std(Scattering_{range,N_t})$
$\#Outliers$	$std(\#Outliers_{N_t})$
$OutlierDev_{mean}$	$std(OutlierDev_{mean,N_t})$

2.3 Data Preparation: Ontology-supported Data Understanding and Integration

In order to perform data preparation in a way that can facilitate the subsequent machine learning, this work suggests to rely on semantic technologies, including ontologies, templates, reasoning, etc. The methodologies in this section were developed in cooperation with Yulia Svetashova. The author's contribution is conceptualisation, co-design of the ontologies (including the upper ontology, manufacturing templates, ML ontology, ML templates, ML pipeline ontology), and design of mechanism for retrieving information from ontologies for data preprocessing and ML modelling. Related publications include: [169, 170, 171, 172, 173]

Process and data understanding. Successful and efficient data preparation requires necessary understanding of process and deep insights of data. Different types of manufacturing processes often involve complex physical or engineering domain knowledge. However, to gain understanding of process and data is often laborious and fallible for data scientists. The communication between data scientists, domain experts and measurement experts can be time-consuming and error-prone due to their asymmetric knowledge background and vocabulary discrepancy.

This work relies on semantic technologies to describe the process and data with knowledge graphs. After a minimal training of knowledge of semantic technologies, process experts, measurement experts and data scientists can work together to encode knowledge (Figure 2.13) in domain ontologies by filling in fields of *Manufacturing Templates*, based on an upper ontology (named as *Core Ontology*, or shortly *core*) for manufacturing. The *core* models the discrete manufacturing process [174] as entities connected to each atomic operation that produces a product, and allows extension to various specific domain ontologies for many variants of welding processes. After that, they can use the ontologies as a "lingua franca" for process and

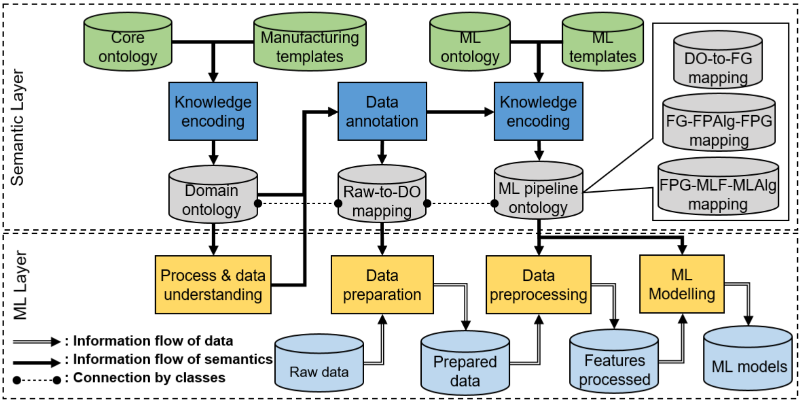


Figure 2.13: Overview of ontology-supported machine learning development, where the *Core ontology* is an upper ontology, *ML ontology* is a task ontology, and the *ML pipeline ontology* is an application ontology (Figure 1.7).

data understanding. The domain ontologies constrain the process and data description with formal language in a pre-defined way, so that the description becomes unambiguous, high-quality, easy to understand, and machine processable.

Data integration. For the same manufacturing process, data collected from different conditions and sources (Section 2.1.4) can have discrepancies due to software versioning, individualisation for customers or specific engineering solutions. Data collected from different processes can have much commonness from the view of data science, e.g. similar mathematical structures or semantic patterns.

By annotating data with uniformly defined structures and feature names from ontologies, data collected from different conditions, sources or even different processes with similarity can be integrated into a *Uniform Data Format*. This simplifies the subsequent data analysis practice.

ML generalisability. Extensibility of machine learning approaches is very desired in manufacturing industry. There exist numerous types of manufacturing processes and their variants. If some groups of processes can be abstracted to common knowledge structures, so that machine learning approaches developed on one process can be extended to other processes with acceptable amount of adaptation effort, progress of Industry 4.0 can be largely accelerated.

This is possible with semantic technologies. After semantic annotation, different datasets can be integrated into one *Uniform Data Format* (UDF). The features in UDF are connected to common terms in *core*. The terms in *core* are then linked to *ML Pipeline Ontologies*, which encode the knowledge of some pre-designed *ML Pipelines*. By doing so, ML development is facilitated significantly. A set of efficient ML pipelines needs to be developed beforehand, and then these pipelines can be configured, and generalised to other datasets through data preparation. This saves effort and costly greatly, compared to developing individual ML pipelines for a variety of similar work for each dataset with even slight discrepancy.

This section introduces the mechanism of ontologies for *Knowledge Encoding* and *Data Integration*. *ML Generalisability* will be elaborated in Section 2.4.

2.3.1 Ontology-supported Process & Data Understanding

In the following, the upper ontology, manufacturing templates and domain ontologies will be introduced.

Upper ontology: *core*. To provide a common knowledge architecture that should be general for manufacturing processes, the upper ontology is developed. Manufacturing processes can be largely divided into discrete processes and continuous processes [174], among which discrete processes are comprised of single operations, each producing a distinct, countable item, e.g. a welding spot on a car-body. Products of such manufacturing are easily identifiable and differ greatly from continuous processes where the products are undifferentiated, e.g. salt, oil, petrol. This thesis only discusses discrete processes, for they are more relevant to electric resistance welding.

In discrete manufacturing processes, there exist the following entities or concepts:

- Physical entities and their organisation: machines, machine parts, production lines, raw products, resulting products, product parts, sensors, etc.
- Processes: manufacturing processes, maintenance processes, and sequences of operations that comprise these processes.
- Software entities: manufacturing software such as control systems, functional modules of software, pre-designed programs for manufacturing, etc.
- Data or concepts to describe these entities and processes:
  - Descriptive parameters of the physical entities, e.g. nominal and actual measured geometry, material, surface and interaction properties of the machines, machine parts, products, product parts, etc.
  - Descriptive parameters of the processes, e.g. operation count, maintenance count, etc.
  - Prescriptive parameters of the processes, e.g. reference curves or setpoints of sensor measurements, control status, monitoring status, etc.
  - Actual measurements of the processes or products, e.g. actual sensor measurements, quality indicators, etc.
  - Calculations or soft sensors of the software system, e.g. statistics of sensor measurements, etc.

This work develops an upper ontology *core* to encode the general manufacturing knowledge. It is an OWL 2 ontology and can be expressed in the Description Logics $S(\mathscr{D})$. Currently, it has 1170 axioms, which define 95 classes, 70 object properties and 122 datatype properties. Of course, the *core* allows further extension.

The main part of *core* is visualised in Figure 2.14, where the classes are represented by rounded blue squares, object properties by black arrows, domain of object properties by starts of the arrows, and range of object properties by the ends of the arrows. This is an operation-centric view, intuitive

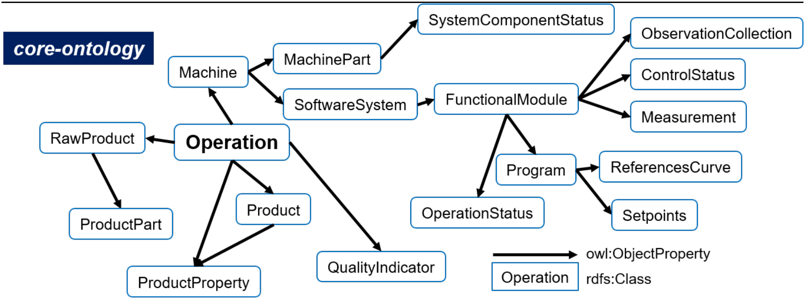


Figure 2.14: Schematic illustration of the upper ontology *core*, serving as a common knowledge architecture of manufacturing. Labels of object properties are omitted for simplicity.

to process experts and convenient for data analysis, for manufacturing data are often organised with operations as basic units. As is in Figure 2.14 illustrated, an operation is the atomic process that takes raw product in and produces a product, e.g. a RSW operation takes worksheets in and produces a welding nugget. Operations have quality indicators. Some of them are product properties, e.g. nugget diameter, and others are operation status, e.g. Q-Value. Processes are sequences of operations.

The operations are performed by machines, which have machine parts. Machines are controlled by software systems. Software systems have functional modules. Functional modules can be measurement modules that connect sensor measurements, modelled by observation collection reusing part of *sosa:Observation*, or control modules that stores the control status and pre-designed programs, etc. Programs prescribe the reference curves for sensor measurement curves, and setpoints for other status or single-valued measurements in adaptive control.

Manufacturing templates. Built on the Reasonable Ontology Templates (OTTR) framework [93], manufacturing templates are fragments of knowledge encoded in parametrised ontologies. By providing values (arguments) for each parameter, even non-ontologists can easily instantiate templates to consistent ontologies as instances of templates, which are then serialised as

OWL axioms. The manufacturing templates contain 30 templates and are connected to *core*. Templates use the same prefix *core:*.

Figure 2.15 schematically illustrates the physical entity templates, where *rdfs:subClassOf* is represented by blue arrows, data type properties by dashed black arrows, *rdfs:Literals* by black squares, and the rest follows the same style of Figure 2.14. The physical entity templates can be used to create subclasses like *core:Machine*, *core:MachinePart*, and *core:RawProduct*. To create a class using this template, the user needs to first select a super class, e.g. *core:Machine*, and fills in fields of the template, like *Name: RSWMachine*, and *Performs operation: RSWOperation*. After this, a series of classes and properties will be automatically created for *RSWMachine*, including many descriptive parameters, such as material, geometry and surface condition. Each of them represents an “Form” [175], or super class and its properties. For example, *core:Material* is a super class of all materials. It can have the name (*core:hasName*) *xsd:"Steel"*, and many material properties modelled with *rdfs:Class*. One example is *core:EModulus*, which has value (*core:hasEModulusValue*), and unit (*core:hasEModulusUnit*).

The class *Interaction* is used to model interaction properties between physical entities. For example, between the two worksheets of RSW, there could exist adhesive; between any two contacting physical entities, the con-

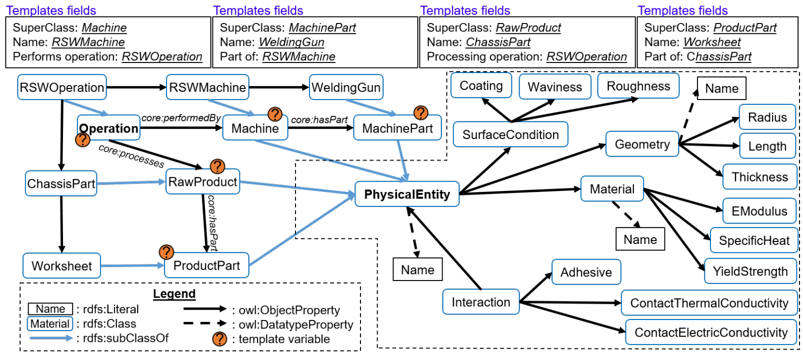


Figure 2.15: Schematic illustration of the templates of physical entity, its properties, and subclasses, including machine, machine part, raw product and product part.

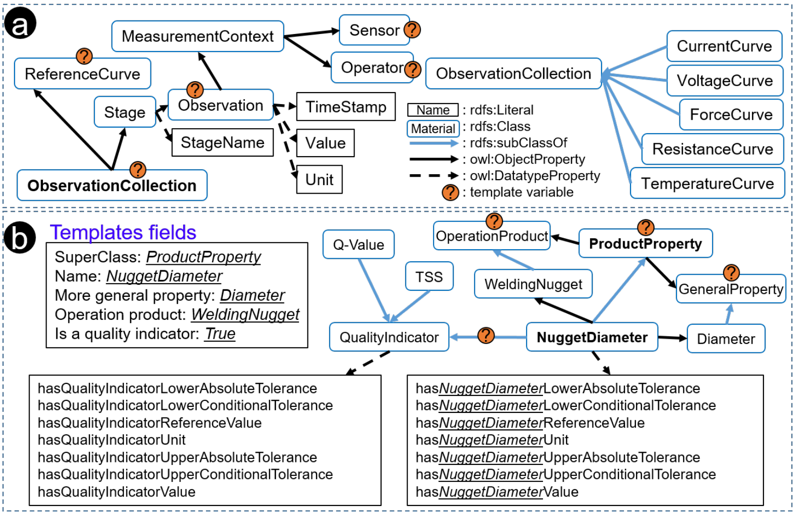


Figure 2.16: Schematic illustration of (a) the templates of observation collection. (b) the templates of quality indicator.

tact properties like thermal or electric conductivity need to be described for simulation data.

Figure 2.16a schematically illustrates the observation collection template for creating classes of process curves (e.g. current curve, voltage curve), which include sequences of measurements (observations) with value, unit, and ordered with time stamps. The welding process is comprised of several stages, e.g. in RSW there exist initial, start, welding, cooling stages. In adaptive control mode, the process curves have reference curves (ideal curve profiles). The observation has measurement contexts like the measuring sensor, or the operator who performed the measurement, etc.

Figure 2.16b demonstrates the template mechanism in datatype property level. The example here is *NuggetDiameter*, which is the product property of the RSW process. The user needs to select a super class (product property), give a name, select the more general property, link it to the product, and check if it is a quality indicator. The datatype properties of a quality indicator will then be created to the nugget diameter with the corresponding internationalised resource identifiers (IRI) changed, e.g. *hasQualityIndicatorUnit* is changed to *hasNuggetDiameterUnit*.

Domain ontologies. With help of templates and *core*, domain ontologies can be easily created. Figure 2.17a is an illustration of the domain ontology for resistance spot welding *rsw*. All classes are subclasses of classes in *core*. Their object properties and datatype properties "inherit" that of *core* and templates. The *rsw* is also operation-centric, where an *RSWOperation* takes chassis part in and produces welding spot. The chassis part consists of worksheet top and worksheet bottom. These two worksheets have sheet-sheet-interaction, where their interaction properties, like adhesive, contact thermal conductivity, etc., are modelled. The welding spot has an important product property, the spot diameter, which is a subclass of quality indicator.

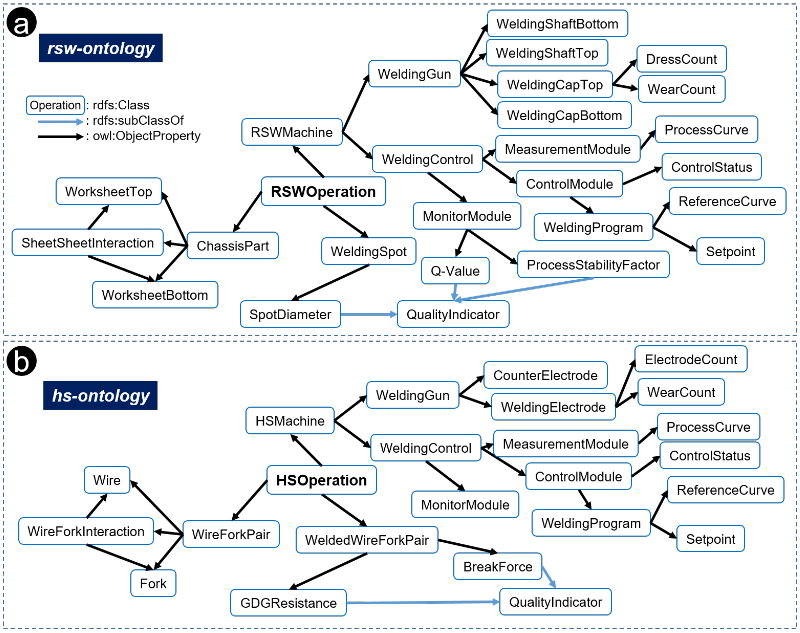


Figure 2.17: Schematic illustration of the domain ontology *rsw*. Labels of object properties are omitted for simplicity.

As is illustrated in Figure 2.17, the *RSWOperation* is performed by an *RSWMachine*, which has machine parts of welding gun, welding shaft top and bottom, welding cap top and bottom, etc. The welding caps have system component status of *WearCount* and *DressCount*, which are operation count and maintenance count. The welding machine is controlled by the software system *WeldingControl*, which has three functional modules. *Measurement-Module* collects the process curves, *ControlModule* performs adaptive control and stores welding programs, prescribing the reference curves and set-points and the *MonitorModule* that monitors the welding operation and calculates operation status like *Q-Value* and *ProcessStabilityFactor*, which are both quality indicators.

Figure 2.17b is an illustration of the domain ontology for hot-staking *hs*. It is very similar to *rsw* when the domain knowledge is represented using the common architecture *core* and templates.

After ontology construction, the domain experts, measurement experts, and data scientists can talk using the visualised ontologies. Since they all have understanding of the upper ontology *core*, it becomes very easy for them to discuss the detailed entities, concepts and data in each welding process. They can continue to expand the ontologies using templates when new aspects of process or new data need to be considered. This greatly simplifies process and data understanding and facilitates the machine learning development process.

2.3.2 Ontology-supported Data Integration

The constructed ontologies do not only serve as a means for understanding, but also for data integration. This section elaborates the issues and solutions of data integration.

Variety of data. Following the same physical laws and engineering principles, the three sources in the multi-fidelity data model (Section 2.1.4) are very similar in nature, but can be very different in format. The effort to unify the data formats is non-trivial because of the variety of data:

- Variety in physical storage level: they are collected from different locations, and stored in storage devices.
- Variety in measurement technology level:
  - They can have different features because different sensors are installed. For example, in simulation there exist a lot more (virtual) sensors than in lab data or production data.
  - The same features may have differences in physical meanings. E.g., the sensors may be installed at different positions. Reasons could be that simulations do not model the complete machine so that some virtual sensors in simulation need to be adjusted to realise their similar functionality.
  - Sensor signals can be collected with different measuring frequencies due to e.g. different measuring devices.
- Variety in data level:
  - Semantic difference: features representing the same or similar physical meanings usually have different variable names. Reasons could be there exist multiple versions of the same control system, or simulation data follow another naming convention, etc.
  - Format difference: The data can be stored in different formats, including SQL databases, xlsx (Excel) tables, json files, xml files, and further custom defined formats.

Raw data formats. Depending on the software system of welding processes, raw data formats are very different. They are recorded by different devices, sometimes partially merged in the software system. All of them are recorded with time stamps (in normal cases). The main categories include:

- SQL databases, which are essentially tables with data instances (of welding operations) in rows and features in columns [176]. For example, RSW production data contain four *protocols* stored in SQL databases.
  - *main protocol*: with data recorded by the welding software systems. Each row records a welding operation, columns are features including available welding quality, control information, system component status;
  - *error protocol*: with minor errors relevant to possible quality deterioration or inefficiency;
  - *failure protocol*: with more severe quality failures;
  - *change protocol*: with recorded manual interference by operators;
- Excel, csv, and txt files, which are still SQL-like relational databases. Examples are: *feedback curves database* with sensor data of resistance, pulse width modulation, force, etc., measured per millisecond during the welding process; and *meta settings database* with general configurations of welding sheet material, geometry, adhesive, etc.
- JavaScript Object Notation (JSON) is human-readable text to store and transmit data objects consisting of attribute-value pairs and serialisable values [177]. JSON is used for e.g. storing data collected from multiple control, measurement and analysing systems of Hot-Staking.
- Extensible Markup Language (XML) [178], which is a both human-readable and machine-readable. This is used for e.g. online changing welding control configuration.
- Output database (ODB) files generated by simulation (e.g. using Abaqus). These files are mainly the output of simulation data source, and need to be post-processed and extracted to other formats.
- RUI files, a special format used by Bosch Rexroth for storing process feedback curves and reference curves.

Feature types and unified file formats. A manufacturing process or an operation can be described by many features. A *Data Tuple (DT)* [50] is used to indicate a single data instance that contains all features fully describing the instance. The common types of formats of these features are listed below.

1. *Quality indicators*: *product quality indicators* or *operation quality indicators*, whose estimation is the central task of quality monitoring.
2. *Identifiers*: features to identify data tuples, e.g. unique IDs for each manufacturing operation.
3. *Single numeric features*: single numbers for describing a welding operation, corresponding to the *ratio* scale in level of measure [179], e.g. actual welding time, temporal structure features (Section 2.1.2), measurements statistics of feedback curves.
4. *Single categorical features*: these are distinct classifications and do not involve a quantitative value or order, corresponding to the *nominal* scale [179], such as failure types, program numbers, etc.
5. *Time series*: sequences of numeric values with temporal structure, e.g. the force measurement continuously collected by sensors.
6. Other types of data formats, e.g. images, videos, log-files.

Among which data of Type 1 should always be present, otherwise the task of process monitoring has no target feature; then data of at least one of Types 3–6 should be present, serving as the input features of ML model; data of Type 2 is needed to provide correspondence between data items of the Types 3–6 and between them and the ones of Type 1.

The *Unified File Formats* is a series of formats with pre-defined structures, and should allow transformation between each other. For example: (1) csv files with Type 1-4 stored in one big csv table and Type 5-6 stored in many csv tables in subfolders; (2) mat files with all types stored in MATLAB structures; (3) JSON files with all types stored in nested attribute-value pairs; and many other file formats.

Uniform data format. After data are transformed into unified file formats, their feature names need to be changed to *unified feature names*. This is essential for allowing any algorithms of data visualisation, data preprocessing or ML modelling in the subsequent steps to access the features in a uniform manner.

The domain experts or measurement experts can inspect the raw feature names, then map them to the domain feature names (datatype properties) of the domain ontologies created by themselves, generating the raw-to-domain mapping (Raw-to-DO). These domain feature names are then simplified to unified feature names by dropping repeating terms like "has", "Value(s)", resulting in a table with the example illustrated in Table 2.7. This table maps the raw feature names from different datasets to the system of unified feature names. The resulting data with unified feature names in unified file formats are named as *Uniform Data Format* (UDF) in this work.

2.3.3 Linkage to Machine Learning Ontology

The next step is to connect the domain feature names with terms in machine learning ontologies, so that data analysis algorithms can access these features through the machine learning feature groups. This is achieved by first mapping the domain feature names (in domain ontologies) to *core* feature names (Table 2.8), which are then mapped to feature groups in machine learning ontology *ml* (DO-to-Core-to-FG, or DO-to-FG) (Section 2.4.6).

The DO-to-FG mapping allows the subsequent data visualisation or analysis to operate only with ML feature groups, without tailoring to specific features or formats in different raw data, which greatly increases the reusability of scripts of data visualisation and ML pipelines. Furthermore, the mapping can also be configured by the users (domain experts, measurement experts, data scientists, etc.). With a set of scripts established beforehand, these users

Table 2.7: The raw-to-domain mapping that maps raw feature names to terms in domain ontology *rsw* and the unified feature names.

Raw variable name	**Domain feature name**	**Unified feature name**
SimulationID	rsw:hasRSWOperationID	RSWOperationID
welding_diameter_top_max	rsw:hasNuggetDiameterValue	NuggetDiameter
Cap_wear	rsw:hasCapWearStatusValue	WearCount
EPOT_ShaftTop	rsw:hasCurveVoltageValues	CurveVoltage
TotalResistance	rsw:hasCurveResistanceValues	CurveResistance

can directly perform data visualisation or data analysis without diving deep into the scripts, which largely facilitates the accessibility and generalisability of ML pipelines.

2.4 Data Preprocessing and ML Modelling: ML Development for Quality Monitoring in ERW

Data preprocessing and ML modelling are deeply interdependent and can be seen as the core activity of ML-based quality monitoring in ERW. To address the OQ 5, *Evaluation and Selection of ML Methods*, literature attempted to categorise machine learning methods into different groups (Section 1.4.1). Similar to this approach, this work organises a wide range of suitable ML methods into the *machine learning pipelines* illustrated in Figure 2.18. This schema is adapted from the pipeline by Fayyad [44] and Mikut [48].

In the workflow, questions need first to be defined and raw data should be prepared to *Uniform Data Format* (UDF). After that, the data are analysed in two steps: preprocessing and ML modelling. The results are interpreted, evaluated, and visualised, based on which decisions are made and a best ML model is selected. The ML model is deployed in industrial application and should be constantly monitored to cope with concept drift. During the model deployment and monitoring, data are continuously collected, and new questions are defined. These steps form a closed loop, and many of them are iterative between each other, depicted by double-headed arrows.

Table 2.8: Example of the DO-to-FG mapping that maps domain feature names in *rsw* to ML feature groups in *ml*

Domain feature name	**Core feature name**	**ML feature group**
rsw:hasRSWOperationId	core:hasOperationID	ml:Identifier
rsw:hasNuggetDiameterValue	core:hasQualityIndicatorValue	ml:QualityIndicator
rsw:hasCapWearStatusValue	core:hasToolWearingStatusValue	ml:SingleFeature
rsw:hasCurveVoltageValues	core:hasObservationCollectionValues	ml:TimeSeries
rsw:hasCurveResistanceValues	core:hasObservationCollectionValues	ml:TimeSeries

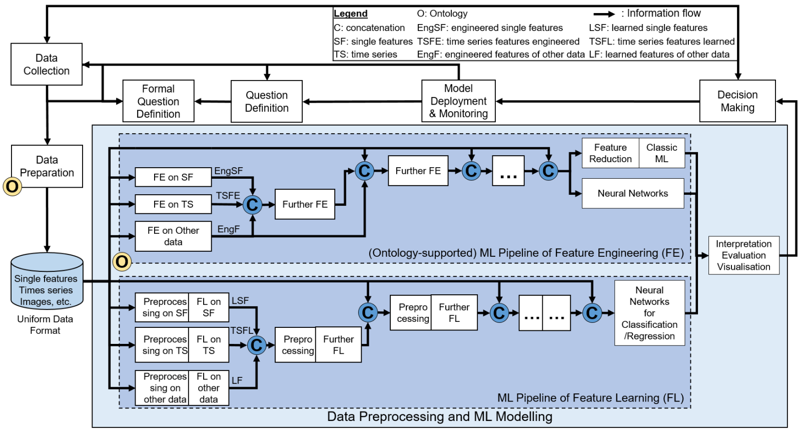

Legend
O: Ontology
: Information flow
C: concatenation
EngSF: engineered single features
LSF: learned single features
SF: single features
TSFE: time series features engineered
TSFL: time series features learned
TS: time series
EngF: engineered features of other data
LF: learned features of other data
Data Collection
Formal Question Definition
Question Definition
Model Deployment & Monitoring
Decision Making
Data Preparation
Single features Times series Images, etc.
Uniform Data Format
FE on SF
FE on TS
FE on Other data
Further FE
Feature Reduction
Classic ML
Neural Networks
(Ontology-supported) ML Pipeline of Feature Engineering (FE)
Interpretation Evaluation Visualisation
Preprocessing on SF
FL on SF
Preprocessing on TS
FL on TS
Preprocessing on other data
FL on other data
Preprocessing
Further FL
Neural Networks for Classification /Regression
ML Pipeline of Feature Learning (FL)
Data Preprocessing and ML Modelling

Figure 2.18: Workflow of ontology-supported machine learning pipelines adapted from [44] and [48]. Ontologies are optional for both data preparation and ML pipeline of FE.

The methods of ML-based data analysis are organised into two types of ML pipelines [51], including Feature Engineering (FE) and Feature Learning (FL, or representation learning). Two traditional ML pipelines are

Feature Engineering (FE) - Classic Machine Learning (CML)

Feature Learning (FL) - Neural Networks (NN)

Moreover, these four components are not mutually exclusive. Instead, they can also be combined as FE-NN and FL-CML.

The outputs of machine learning methods can have the following formats: categorical values (for classification problems), numerical values (for regression problems), time series (for regression with temporal structures, e.g. forecasting), images (for regression with spatial structures, e.g. image de-noising, repairing, generation, etc.), etc. These outputs need to be interpreted using domain knowledge. Only after this step, the results can be used to support decision-making.

The following subsections will dive deep into particularities of data preprocessing and ML modelling for quality monitoring in ERW.

2.4.1 Data Preprocessing: Handling Data with Hierarchical Temporal Structures

Hierarchical feature extraction. There exist several time levels in the hierarchical time structures (Figure 2.19). Features extracted from the time series on the *welding time level* are vectors containing compressed information from the time series. These time series extracted features are on the *welding operation level*, i.e. each welding operation corresponds to a set of time series extracted features. The time series extracted features can be concatenated with other features of the *welding operation level*, e.g. single features, and the consecutive concatenated features form another time series on the welding operation level. These concatenated features can be further extracted to features on the same time level or on the next time level, e.g. *welding program level* or *dress cycle level*. The final achieved level depends on the desired granularity of time levels.

In the FE pipelines (Figure 2.18), different types of features in UDF are processed by corresponding feature engineering algorithms. They are then concatenated for further feature engineering. This procedure can continue to repeat in a cascading manner. Similarly, in the FL pipelines, different types of features in UDF are preprocessed to adapt their format (e.g. ma-

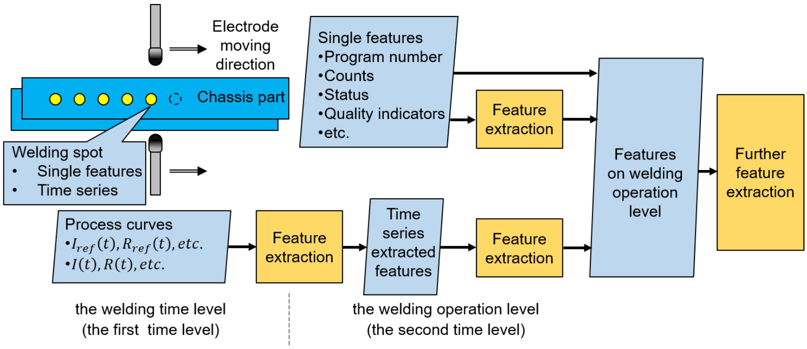

Electrode moving direction
Chassis part
Welding spot
• Single features
• Time series
Single features
•Program number
•Counts
•Status
•Quality indicators
•etc.
Process curves
•$I_{ref}(t), R_{ref}(t), etc.$
•$I(t), R(t), etc.$
Feature extraction
Time series extracted features
Feature extraction
Feature extraction
Features on welding operation level
Further feature extraction
the welding time level (the first time level)
the welding operation level (the second time level)

Figure 2.19: An example of hierarchical feature extraction on single features and process curves (time series). Feature extraction can be performed with feature engineering or feature learning, resulting in two types of ML pipelines in Figure 2.18

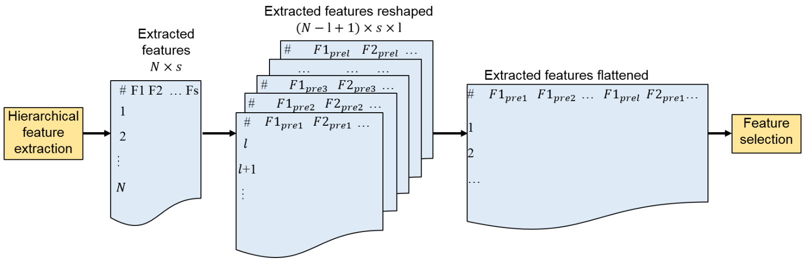
trix dimension) for corresponding feature learning neural networks. They are then concatenated and can go through further preprocessing and feature learning. This procedure can continue to repeat in a cascading manner.

At the end, the extracted features (or latent variables) can be modelled by further neural networks or classic ML methods after feature reduction, for solving classification or regression tasks.

Data reshaping to accentuate short-time dependency. For predicting quality of the future welding operations (typically one time step in the future, i.e. the next welding operation), the input features, extracted features or latent features need to be reshaped to small time snippets of a certain look-back length.

If the consecutive welding operations are assumed independent, the look-back length should be one and the output feature will be at the same time step as the input features. Since the temporal dependency of welding operations is assumed in this work, the look-back length will be normally greater than one.

The look-back length cannot be infinite or varying. This is because, on the one hand, most machine learning algorithms require the input features to have a fixed length, and on the other hand, the data just before the next operation are assumed to be more influential for the next operation. Therefore, only a certain number of welding operations before the next welding operation will be taken into account for predicting the quality of the next operation.



Hierarchical feature extraction
Extracted features
$N \times s$
# F1 F2 … Fs
1
2
⋮
N
Extracted features reshaped
$(N - l + 1) \times s \times l$
# $F1_{prel}$ $F2_{prel}$ …
# $F1_{pre3}$ $F2_{pre3}$ …
# $F1_{pre2}$ $F2_{pre2}$ …
# $F1_{pre1}$ $F2_{pre1}$ …
l
l+1
⋮
Extracted features flattened
# $F1_{pre1}$ $F1_{pre2}$ … $F1_{prel}$ $F2_{pre1}$…
1
2
…
Feature selection

Figure 2.20: An example of data reshaping to handle data of hierarchical temporal structures

In this way, the data are reshaped to accentuate short-time dependency (Figure 2.20). The extracted features are represented by $N \times s$ matrix, where N is the number of data tuples in the dataset to be preprocessed and s is the number of features. The matrix will be folded to a three dimensional matrix with the extra dimension l for the number of previous welding operations to be considered. If the subsequent machine learning algorithm is a classic machine learning method that can only handle data in flat table format (consisting of rows and columns based on relational models [176], often can be queried with SQL), the 3D matrix will be flattened to a flat table and fed into classic machine learning methods. If the subsequent machine learning algorithm is a method that can handle temporal structure, e.g. RNN, the 3D matrix will be directly fed into the ML method.

Data splitting according to temporal structures. Data splitting also needs to take the temporal structures of data into consideration. The data are split into training, validation and test set according to some complete units of a time level. Figure 2.21 illustrates data splitting of Q-Value according to complete dress cycles. According to domain knowledge, the application scenario will only be to test the developed machine learning methods on complete dress cycles. It is therefore more meaningful to split the data also

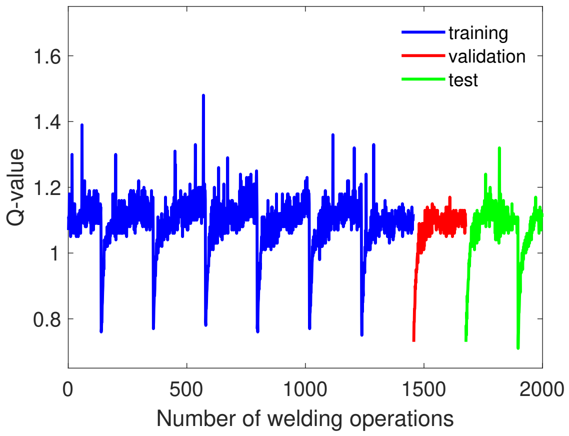


Figure 2.21: Example of data splitting rounded to complete dress cycles

in this way, to test the generalisability of the methods on complete dress cycles.

There exists an issue about data overlapping. For example, a dataset is split into trainingx set (including training and validation data) and test set according to the output variable QI (Figure 2.22). Quality prediction of the first operation in the test dataset (QI_{i+1}) would need its input features of previous l time steps (from $Inputs_{i-l+1}$ to $Inputs_i$), among which $l-1$ data tuples overlap with the input features (from $Inputs_{i-l}$ to $Inputs_{i-1}$) for quality prediction of the last operation in the trainingx dataset (QI_i).

It is arguable whether this issue of data overlapping would cause information leakage since the input features of different time steps possess different temporal meanings for predicting the output. This work will strictly avoid data overlapping, resulting that the first l output in the test dataset cannot be predicted, as is the same case for the trainingx dataset. The data overlapping between training dataset and validation dataset is not critical since the information of validation dataset is used anyway for model selection.

2.4.2 Data Preprocessing: Domain Knowledge Integrated Feature Engineering

FE on TS: Unifying time series lengths. The *Time Series* (TS) in UDF need to be unified to the same length before further processing. Possible methods [180] include *Truncation*, *Padding*, and *Resampling*.

- Truncation is to cut all time series to the same length, often the minimal length in the dataset. This causes loss of information.
- Padding is to elongate the time series to the same length, often the maximal length in the dataset, with different values that are physically meaningful. For example, current, voltage and pulse width modulation can be padded with zero, since after welding they are de facto zero, while resistance should be padded with the last value, for resistance is the intrinsic property of matter and does not disappear after welding. This does

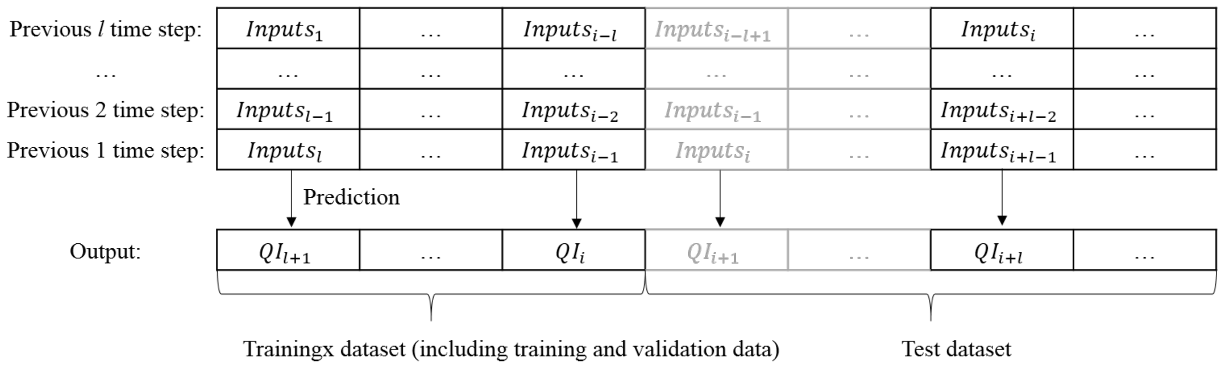


Figure 2.22: Data overlapping issue because of data reshaping: Quality prediction of the first operation in the test dataset (QI_{i+1}) would need its input features of previous l time steps (from $Inputs_{i-l+1}$ to $Inputs_i$), among which $l-1$ data tuples overlap with the input features (from $Inputs_{i-l}$ to $Inputs_{i-1}$) for quality prediction of the last operation in the trainingx dataset (QI_i). The grey area is the area with the issue of data overlapping.

not cause information loss, and is the most physically meaningful, but padding will create artificial information if e.g. the subsequent data pre-processing is PCA (See the sudden jumps of the principle components in Figure 2.23a after padding).

- Resampling is to scale the time series to the same length. This preserves all information and does not create artefacts, but it will destroy some patterns in data. For example, the time point between the initial stage and start stage in the raw data will be always at t_0 ms. After resampling, this time point for time series of different lengths will be moved to different places.

FE on TS: Feature extraction. After unifying the lengths, various feature extraction strategies can be adopted [56, 57, 58, 125, 135, 141], depending on the application scenario. Some are discussed below:

- Statistics: eight statistic features of minimum [135], maximum [38], minimum position, maximum position [125], mean [135], median, standard deviation [135], and length (welding time [139]) are the most basic design. A minimal understanding of domain knowledge is required.

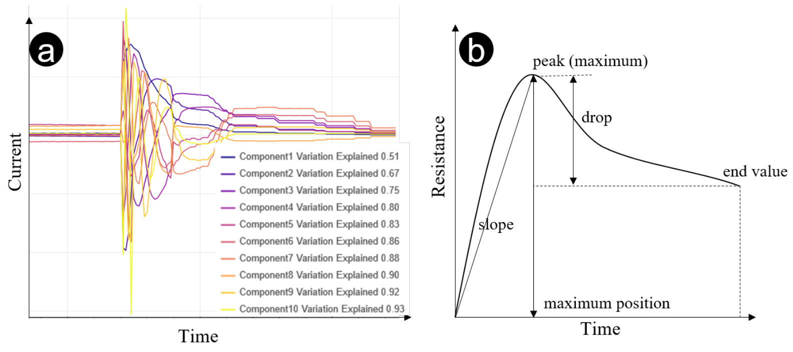


Figure 2.23: (a) Artefacts in principle components caused by padding current curve with zero. (b) Example of domain knowledge integrated feature extraction for resistance curve [38, 135, 36, 125]

- Domain knowledge integrated feature extraction: as Figure 2.23b illustrates, features like slope ($slope = (max - min)/(mxpo - mnpo)$ [126]), drop, end value, root mean square, skewness, etc., can be of interest, depending on the domain knowledge and application scenario.
- Segmentation [180]: time series can be divided into segments according to the welding stages. Feature extraction can be performed on each segment to provide more detailed information.
- Principle component analysis (PCA) [181]: time series can be aggregated to several principle components and to preserve the desired variance level by selecting the number of components. With PCA, the original time series can be reconstructed from the components. This is especially desired for application of e.g. optimisation of future process curves which requires inverse calculation of the optimal time series.

The feature extracted from time series using feature engineering will be referred to as *Time Series Features Engineered (TSFE)* in this work.

FE on SF: Processing temporal structure features. Strategies for feature engineering on single features are designed based on the meaning of features in domain knowledge, changing the representations of the raw features. The extracted features are denoted as *Engineered Single Features (EngSF)*. One proven strategy [170] is to generate new features based the temporal structure features to highlight some temporal relationships. Three examples are listed:

- *WearDiff* is calculated as the difference between WearCount of two consecutive data tuples, characterising the degree of change of wearing effect. The value is normally ONE if the data are continuous; if some data tuples are missing, the value will be other numbers that correctly describe the wearing effect; and the value will be a large negative value after each fresh dressing.
- *NewDress* will be ONE after each dressing, and ZERO for other welding operations.

- *NewCap* will be ONE after each Cap Change, and ZERO for other welding operations.

One advantage of these features is that before the next welding operation happens, the WearCount, DressCount, and CapCount of the next operation are already known, since they are artificially designed incremental features. The EngSF based on them are therefore also known. These features corresponding to the next welding operation can therefore be used for forecasting quality of future welding operations.

FE on SF: Encoding categorical features. Categorical features are features with non-numerical values (e.g. material type is steel or aluminium) or features whose values are numerical but they can not be treated as ordinal numbers (e.g. failure type I, II). Most machine learning algorithms can only deal with numerical values (except for a few exceptions, such as Random Forests). The categorical features therefore need to be first encoded as numerical features. Common methods include *Ordinal Encoding* and *One-Hot Encoding* [182].

- *Ordinal Encoding* is to simply transform the categorical feature to a series of integers. For example, steel can be encoded as 1, aluminium as 2, copper as 3, etc. The advantage is its simplicity, while the disadvantage is the misrepresentation of relationships between the encoded values, e.g. aluminium (2) is not twice "big" as steel (1).
- *One-Hot Encoding* is to encode categorical features as a "switch" between models. For example, steel can be encoded as a vector [1, 0, 0], aluminium as [0, 1, 0], and copper as [0, 0, 1]. The advantage is that there exists no misrepresentation of relationships between the encoded values. The disadvantage is the length of the vector is determined by the possible values of the feature. The consequence is, e.g. a model trained on a dataset with three material types cannot be transferred to another dataset with four material types, if the newly added feature value has not been foreseen in the encoding.

These two methods all have their disadvantages, and are not suitable for encoding the feature “Welding program" (Prog1, Prog2, etc.), which is a very important feature in welding. According to domain knowledge and observing from Figure 2.2b, Q-Values with different welding programs have different behaviours, but this work does not consider Program Number (ProgNo) as a good feature for machine learning modelling. The same value of ProgNo would have different meanings for different welding machines in case of using raw values of ProgNo for modelling, and the number of features may change in case of *One-Hot Encoding*. Therefore, this work creates another type of features to incorporate the information of ProgNo implicitly, avoiding using the feature ProgNo (Figure 2.24).

Further FE: Incorporating welding program information. Firstly, all single features that form time series on the welding operation level will be decomposed into sub-time series, each only belonging to one ProgNo. Secondly, the aforementioned EngSF are extracted separately from each sub-time series. We give a name to this group of features: *Engineered Single Features considering ProgNo, (EngSF_Prog)*. For example, *WearDiff_Prog* is calculated as the difference between consecutive WearCounts that belong to the same ProgNo. *NewDress_Prog* and *NewCap_Prog* are calculated similarly.

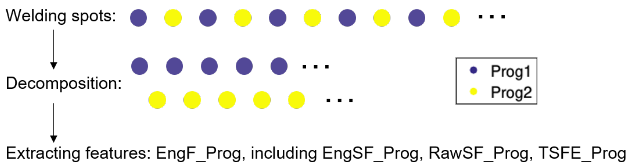

Welding spots:
Decomposition:
Prog1
Prog2
Extracting features: EngF_Prog, including EngSF_Prog, RawSF_Prog, TSFE_Prog

Figure 2.24: Example for generating EngF_Prog with data of Welding Machine 1, modified from [170]. Each dot indicates a welding spot and its data. Purple dots belong to Welding Program 1 and yellow dots belong to Welding Program 2. All single features that form time series on the welding operation level are decomposed into sub-time series, each only belonging to one ProgNo. The Engineered Features considering ProgNo (EngF_Prog) are extracted separately from each sub-time series.

Moreover, the following features are also created to incorporate the information of welding program implicitly.

- *RawSF_Prog* indicates the features generated by decomposing the *raw single features* of the data points belonging to the same ProgNo.
- *TSFE_Prog* indicates the features generated by decomposing the *time series features engineered* of the data points belonging to the same ProgNo.

EngSF_Prog, *RawSF_Prog*, and *TSFE_Prog* are grouped under the name *Engineered Features considering ProgNo (EngF_Prog)*.

Other feature engineering strategies on welding operation level are possible, depending on the domain knowledge and application scenarios, e.g. creating polynomial features (I^2), interaction features ($I \times U$), exponential features (e^t), logarithmic features ($ln(F)$), etc.

2.4.3 Data Preprocessing: Feature Reduction

Feature reduction is crucial for applying classic ML methods on manufacturing data for many reasons: (1) Too many features can lead to over complex machine learning models, which tend to overfitting; (2) Sometimes the number of features even exceed the number of data tuples when data collection is costly; (3) Too many features also lead to long computation time, which is not desired in an industrial application; (4) Not all features contain necessary information of building a successful model; (5) Many features in manufacturing data contain information redundancy, e.g. resistance can be derived from current and voltage, and high collinearity between input features are therefore prevalent.

This work suggests using *step-wise forward feature selection* [48] (SFFS) for feature reduction in ML pipelines of FE-CML, which is efficient and effective. Other feature selection methods, e.g. recursive feature elimination, univariate F-Test, normally yield worse performance than SFFS in experimenting. Highly correlated features will cause problems in feature selection and regression. Step-wise forward feature selection is relatively insensitive to high collinearity problem.

For ML pipelines of FL-NN, feature reduction is automatically handled by neural networks.

2.4.4 Data Preprocessing: Normalisation

Normalisation is necessary to make data analysis scale-independent, especially important for calculation of distance (dissimilarity) or for ML methods that assume a zero-average, e.g. MLP. Normalisation methods used in this work include:

- *Min-Max normalisation* [183] (also called One-Zero normalisation) is to scale the data into a range of (0,1) (Equation 2.10, where $\mathbf{x}_n$ is the normalised data).
$$\mathbf{x}_n = \frac{\mathbf{x} - \mathbf{x}_{min}}{\mathbf{x}_{max} - \mathbf{x}_{min}} \tag{2.10}$$
- *Z-score normalisation* [183] (also called standard-score normalisation) assumes the data conform to a Gaussian distribution and sets the mean value of data to zero and standard deviation to 1 (Equation 2.11), where μ_x is the mean value of $\mathbf{x}_n$ and σ_x is the standard deviation.
$$\mathbf{x}_n = \frac{\mathbf{x} - \mu_x}{\sigma_x} \tag{2.11}$$

For input features of ML models, normalisation should always be performed. For output features, if their scale is fixed to a level, sometimes it's possible or even better to not perform normalisation. For example, when *mape* is used as the performance metric, z-score normalisation would turn many target values to zero and *mape* becomes inapplicable. In this case, *mape* should be calculated after "de-normalisation", that is to rescale the prediction results back to the original scale of the prepared data. The normalisation parameters (e.g. $\mathbf{x}_{min}$, $\mathbf{x}_{max}$, μ_x, σ_x) are learned from the training data, when the data are split into training data and test data), or from the trainingx data (including the training data and the validation data), when the data are split into training data, validation data and test data.

2.4.5 ML Modelling: Classic Machine Learning Method Selection

Classic machine learning methods require the input features to be reshaped to a flat table-like data format (consisting of columns and rows similar to a SQL relational table), and normally assume these input features are independent from each other. *Linear regression* (LR) is suggested to begin with for solving every problem, according to the principle of Ockham's razor [184]. The performance of LR models provides a first estimation of problem complexity. *Polynomial regression* (PolyR) is suggested to be tested after linear regression, for it provides insights into degree of data non-linearity and feature importance when interpreted with domain knowledge. *Multi-layer perception* (MLP) with one hidden layer is recommended for data with relatively high complexity or high non-linearity. *K-nearest neighbour* (kNN) is recommended for small datasets and locally linear data.

If classic methods do not yield satisfactory performance, testing feature learning methods becomes more interesting (Section 2.4.7).

2.4.6 Data Preprocessing and ML Modelling: Ontology-enhanced Machine Learning for FE-CML

Semantic technologies like ontologies and ontology templates allow constructing semi-automated and configurable machine learning solutions. The pre-requisites are: (1) ML solutions comprised of modularised machine learning components, and (2) a set of ontological models, including a pre-designed (and extensible) ML ontology, ML templates, and ad hoc designed ML pipeline ontologies (Figure 2.13). The intuition is to parametrise ML solutions with ontological models. In particular, the ML ontology prescribes the allowed solution space for constructing ML pipelines; ML templates are the parametrised ontology fragments to instantiate modularised ML components; and ML pipeline ontologies encode knowledge of concrete ML solutions for specific questions. With these ontological models, users like data scientists, engineers and process experts, can configure, extend and construct ML pipelines without diving into details of ML scripts. Instead, based on a set of pre-designed ML solutions, the users only need to adapt ML solutions to new datasets by annotating raw data with domain ontology terms or feature groups for the new dataset; and configure or design new ML solutions by constructing ML pipeline ontologies using ML templates. This thesis discusses ontology-enhanced Feature Engineering-Classic Machine Learning (FE-CML) solutions. Ontology-enhanced ML solutions for FL-NN awaits future research.

ML ontology. Following the same principles of process description using ontologies, ML knowledge is encoded beforehand into formal language in the *ML Ontology*, (*ml*, Figure 2.13). *ML Ontology* is a *Task Ontology* (Figure 1.7b), which prescribes the general knowledge of the task of machine learning analysis, such as the general workflow of ML pipelines, feature groups, their applicable processing algorithms, resulting processed features, and ML algorithms. *ML Ontology* will be combined with *ML templates* to create ML pipeline ontologies (application ontology in Figure 1.7b) for specific application scenarios.

Figure 2.25 schematically illustrates *ml* for describing a general workflow of ML pipelines in Figure 2.18, in which the visualisation techniques follow that of Figure 2.14. An ML pipeline starts from the prepared data (modelled in *feature prepared layer*), goes through a series of data preprocessing steps (*feature processing layer* and *feature processed layer*) and finally reaches the ML modelling steps, including specifying the input and output features (*ML feature layer*), choosing ML modelling algorithms (*ML modelling layer*), resulting in ML models (*ML model layer*). The prepared data are annotated with *FeatureGroup* in feature prepared layer, which can be reasoned from the DO-to-FG mapping (Table 2.8) and allows manual configuration. Feature processing algorithms are linked to suitable/allowed input *Feature Groups* and output *Feature Processed Groups*. The feature processing and ML modelling can be better understood with examples of ML templates in the following text.

ML templates. Figure 2.26a illustrates ontology-supported feature processing with the example of the algorithm: *GetTSStats*. To instantiate the *GetTSStats* into an ML pipeline ontology, the user needs to select the super class, give a name, and choose the input feature group from a list of allowed feature types encoded in *ml*. If *FG-TimeSeries* is chosen, it means all its subclass features (e.g. time series of current, voltage, resistance) will be processed by *GetTSStats*. The output from *FPG-TSStats* are reasoned from *ml*, and all subclass names (i.e. feature names) are also reasoned using a pre-defined naming convention. More fine-granular manual configuration on feature level and naming is also allowed.

Figure 2.26b schematically illustrates some example templates of feature processing algorithms for *FG-Wear*, *FG-Count*, *FG-SingleFeature* and their resulting feature processed groups. Apart from feature engineering to generate *WearDiff*, *NewDress* and *NewCap* introduced in Section 2.4.2, some

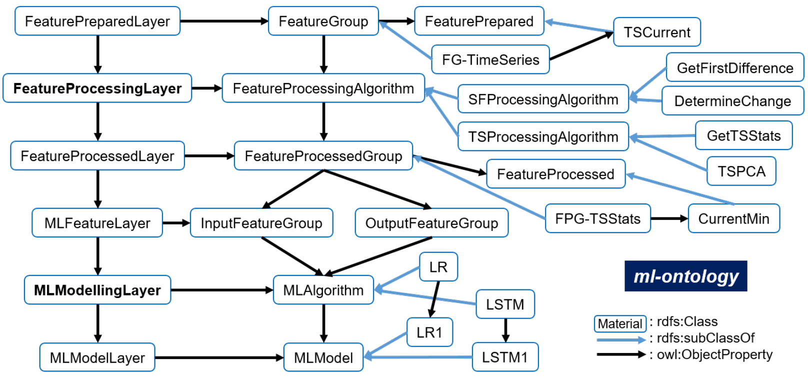


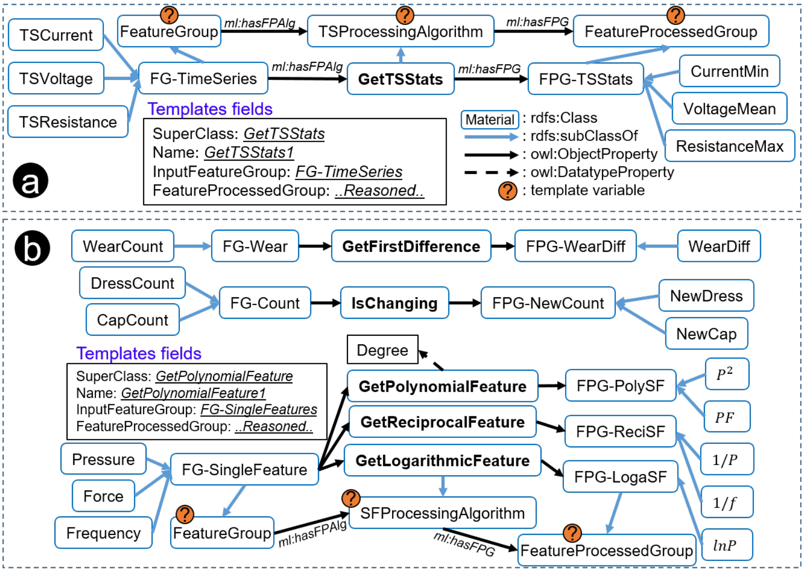
Figure 2.25: Schematic illustration of part of the machine learning ontology *ml*.

common feature engineering strategies to generate polynomial features, reciprocal features, etc. are also illustrated in the figure.

Figure 2.27 schematically illustrates the template for ML modelling, in which the user needs to choose some Feature Processed Groups (FPG) as the input ML features (MLF-Input) and output ML features (MLF-Output). The model class will be reasoned.

ML pipeline ontology. ML pipeline ontologies are instantiated using *ml* and *ML templates*. Figure 2.28 schematically illustrates a simple ML pipeline ontology, where statistic features are extracted from time series, and concatenated with single features and used as the input features to predict the output feature, quality indicator, using linear regression.

Ontology-enhanced machine learning. The ML pipeline ontologies encode knowledge of ML solutions, which are ML pipelines starting from Feature Groups (FG), through Feature Processing Algorithms (FPAlg) and Feature Processed Groups, to ML Algorithms (MLAlg). With the domain



SuperClass: GetTSStats
Name: GetTSStats1
InputFeatureGroup: FG-TimeSeries
FeatureProcessedGroup: ..Reasoned..

SuperClass: GetPolynomialFeature
Name: GetPolynomialFeature1
InputFeatureGroup: FG-SingleFeatures
FeatureProcessedGroup: ..Reasoned..

Figure 2.26: Schematic illustration of the templates for feature processing

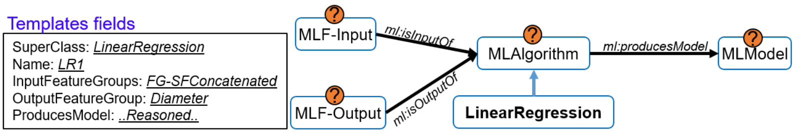

Templates fields
SuperClass: LinearRegression
Name: LR1
InputFeatureGroups: FG-SFConcatenated
OutputFeatureGroup: Diameter
ProducesModel: ..Reasoned..
MLF-Input
MLF-Output
ml:isInputOf
ml:isOutputOf
MLAlgorithm
ml:producesModel
MLModel
LinearRegression
?

Figure 2.27: Schematic illustration of the template for ML modelling.

ontology, the core ontology, the ML ontology and an ML pipeline ontology, three mappings (Figure 2.13) can be inferred via e.g. SPARQL queries: Domain feature names to Feature Group mapping (DO-to-FG), Feature Group to Feature Processing Algorithm to Feature Processed Group mapping (FG-FPAlg-FPG), and Feature Processed Group to ML Feature to ML Algorithm mapping (FPG-MLF-MLAlg). Figure 2.29 illustrates the complete ontology-enhanced ML pipeline with the simple example pipeline in Figure 2.28 in the following steps:

1. The raw feature names are retrieved from raw data and features are renamed to domain feature names using the Raw-to-DO mapping (Section 2.3.2);
2. All features are assigned with *FG* based on the DO-to-FG mapping (Section 2.3.3);
3. Based on the FG-FPAlg-FPG mapping, the corresponding *FPAlg* for each *FG* is retrieved. All *FPAlg* are connected with ML script modules to

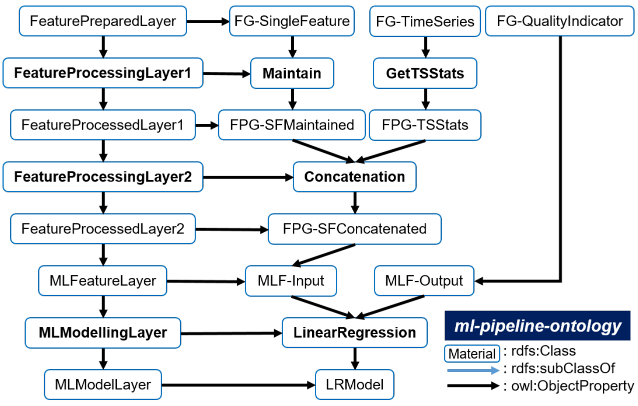


Figure 2.28: Schematic illustration of an example ML pipeline ontology.

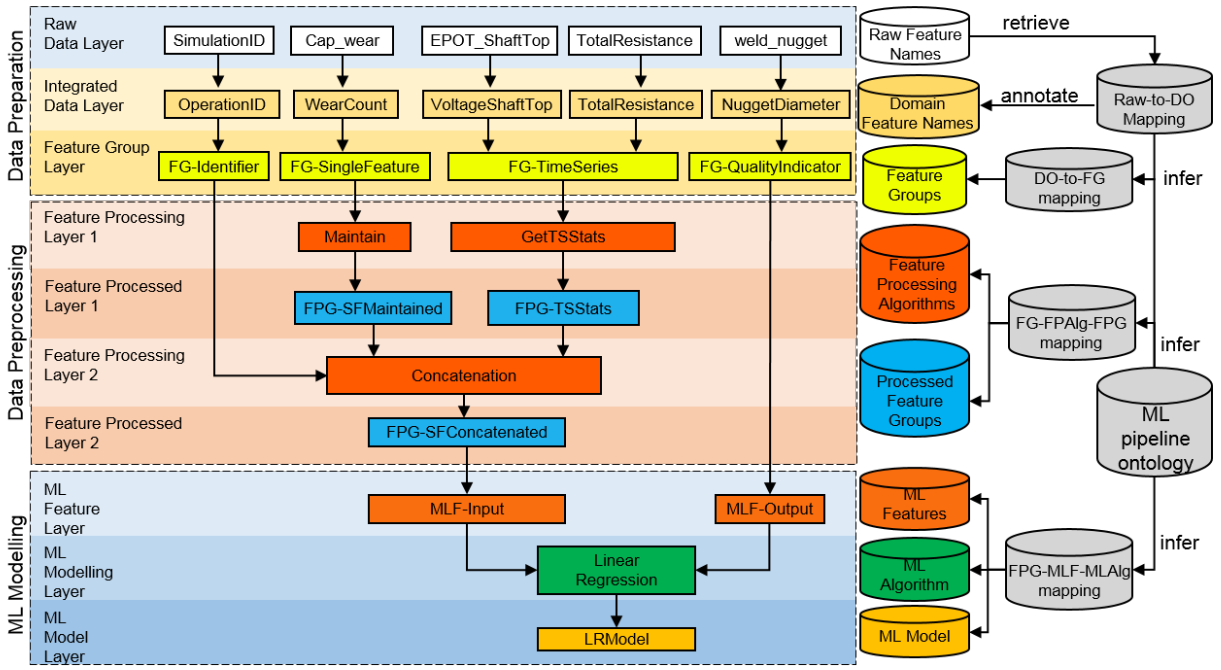


Figure 2.29: An example of ontology-enhanced ML pipeline of FE-LR, modified from [170]

process the features. The processed features are annotated with *FPG*. For example, the *GetTSStats* is retrieved for *FG-TimeSeries*. Thus, the ML module *GetTSStats* processes all time series features and generate the time series statistic features, like *VoltageMean*, *ResistanceMinimum*, etc. The time series statistic features are annotated with *FPG-TSStats*;

4. Feature processing can have multiple sub-steps, where the newly generated features are further processed by FPAlgs to output new FPGs. In the example, the time series statistic features (annotated with *FPG-TSStats*) are concatenated with the unchanged single features (annotated with *FPG-SFMaintained*). The OperationID (annotated with *FG-Identifier*) is used for finding the correspondence between them so that they can be correctly concatenated.
5. According to the FPG-MLF-MLAlg mapping, some FPGs are selected as the input for ML modelling, and some other FPGs (also FGs are possible) as output. In the example, the concatenated features (annotated with *FPG-SFConcatenated*) are selected as *MLF-Input* and the NuggetDiameter (annotated with *FG-QualityIndicator*) as the *MLF-Output*;
6. The MLAlg trains a model on the input and output data pairs. In the example, linear regression is used to take in the concatenated features to predict the diameter, and generate a *LRModel*.

The execution of the described pipeline is possible due to the knowledge encoding with ontologies for 1) domain knowledge, 2) ML feature processing strategies and 3) classic ML algorithms and order of their application. Semantic enhancement enables users to start with the initial (Raw-to-DO) mapping and execute these ML pipelines with small adjustments and adaptations to new datasets and new questions. The generated four mappings during data preparation, data preprocessing and ML modelling can also be used to inspect the resulting models and the underlying data, to explain the feature importance and to interpret the best model.

2.4.7 Data Preprocessing and ML Modelling: Hierarchical Feature Learning

An alternative ML pipeline to the hierarchical feature engineering introduced in Section 2.4.2 is hierarchical feature learning. The general schema is already illustrated in Figure 2.18. Preprocessing is required to make the data suitable for feature learning, e.g. align the data matrices dimensions by reshaping, padding, truncation, etc. Feature learning (FL) for single features can be performed with MLP. FL for time series (sequences with temporal dependencies) can be performed with RNN, FL for images (matrices with spatial dependencies) can be performed with CNN, and FL for features with temporal and spatial structures (e.g. videos) can be performed with a combination of RNN-CNN. After the initial feature learning on different feature types, the resulting latent features can often be concatenated for further feature learning. After all feature learning layers, some neural network layers are followed to perform ML modelling. Note that there exists no clear boundaries between feature learning and ML modelling. This separated description is only of philosophical meaning.

Figure 2.30 illustrates an example of the ML pipeline of FL-NN for predicting the Q-Value of future welding operations. Process curves padded to length k form time series on the welding time level. A total number of sz of these curves of l previous welding operations are first reshaped and fed into a time distributed LSTM (it is a RNN) network for feature learning. The resulting Time Series Features Learned (TSFL) are on the welding operation

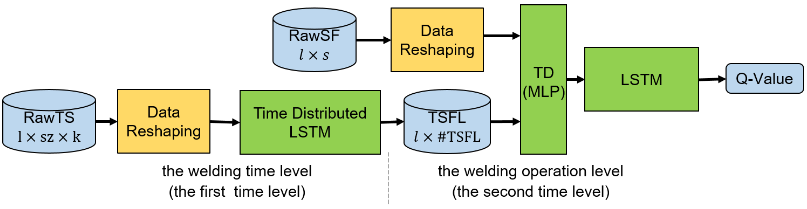


Figure 2.30: An example of hierarchical feature learning pipeline with LSTM (Use Case 4.3). PaddedTS: time series padded to the same length. RawSF: raw single features. TD(MLP): time distributed MLP. TSFL: time series features learned.

level, and are concatenated with the raw single features. The concatenated features form time series on the welding operation level, and are fed into a time distributed MLP network for a further step of feature learning. The resulting features are latent features that contain compressed information of the previous l welding operations. A further LSTM network is followed to aggregate the latent features to predict the Q-Value. The complete neural network is trained together using back-propagation.

2.4.8 Evaluation: Discussion of Performance Metrics

Prediction accuracy of ML models is the classic performance metric in the ML community. However, to ensure that ML approaches have a good chance to be widely adopted in manufacturing industry, not only prediction accuracy is important, but also many other performance metrics that influence the adoptability of ML approaches.

Prediction accuracy. In this work, prediction accuracy is defined in the following way:

- *rmse* (or *mse*) [137, 142] stands for root mean squared error (or mean squared error) between the target values and estimations. The disadvantages of rmse are that it is scale-dependent [185], and not particularly intuitive for domain experts, compared to other measures.
- *mae* stands for mean absolute error between the target values and estimations. It is less sensitive to outliers, compared to *rmse*.
- *mape* stands for mean absolute percentage error. It is scale-independent and intuitive for domain experts. The disadvantage is this becomes inapplicable when there exists zeros in target values.
- *Correlation Coefficient* (or another similar metric, the coefficient of determination R^2) [186] is good for judging the ability of ML models to predict the general trend of the training data.
- The most intuitive measures for process experts are perhaps *User-Defined Errors* (*UDE*) [142]. These are more suitable for helping to set the safety

factor, thus closer to the goal of minimising product failures. Two common UDEs are *Error Within 5%*, which means the percentage of predictions with relative errors smaller than 5% of the target value (In welding 5% error is fairly good performance). *Error Within 10%* is likewise.

Adoptability metrics. Machine learning in manufacturing has more requirements than prediction accuracy. Many other factors, e.g. the time and effort for development of ML approaches, data collection, and implementation, should also be taken into consideration. This work suggests using the following performance dimensions for a more comprehensive evaluation of adoptability of ML approaches.

- *Domain understanding* reflects the effort and time for data scientists or engineers to understand the domain to enable successful ML approaches. This can be measured by questionnaires or time spent for understanding.
- *Feature understanding* measures the time required for detailed understanding of features for successful data preprocessing. This can be measured by questionnaires of cognitive difficulty or time spent for understanding.
- *Data preparation* measures how efficient or generalisable the merging and integration of data from different sources are. This can be measured by questionnaires or time spent for adapting developed ML pipelines for new datasets.
- *Hyper-parameter tuning* measures the effort required for tuning hyper-parameters of ML models. This can be measured by questionnaires, number of tuning experiments, or time spent for tuning.
- *Required training data amount* indicates the minimal data amount necessary for training a machine learning model with acceptable performance. This is especially critical, as the data collection in manufacturing can be very time-consuming and expensive. This can be directly measured by the number of necessary training data tuples, or by time spent on data collection.

- *Average training time and test time* measures the mean value of training time of ML models, including data preprocessing. Training time influences the time delay resulted from model adaptation when new Trends are detected in the data so that existing models are no longer successful. Test time determines whether the approach is suitable for online implementation in hardware. This is measured by time with description of computing hardware.
- *Model explainability* measures whether the selected features, ML model and prediction results are easy to understand, especially for non-ML-experts. This can be measured by questionnaires or time spend for model explanation.

3 Implementation

This chapter concisely describes the implementation of the proposed framework in Chapter 2. The framework is implemented with two programming platforms: (3.1) Python [187] language, including various libraries for machine learning like scikit-learn [188], tensorflow [189], and keras [190]; (3.2) the SciXMiner [191] data analysis toolbox based on MATLAB [192].

3.1 Implementation with Python

Data collection is done by process experts in production, measurement experts in laboratory, or simulation engineers. This work implements the ML pipeline from data preparation, through data preprocessing, and ML modelling (Figure 3.1).

Data preparation. This step is the process of *extract-transform-load*.

Extract. Various Bosch internal tools were used to extract data from raw formats of SQL database, RUI files, control backup files etc., and store them in text/csv, mat (MATLAB format) or json formats. For all three data sources (production, lab, simulation), the extracted data have two major groups. If the data are stored in text/csv format, these two groups are: (1) A set of relational tables of single features, with each row corresponding to one welding operation, and each column corresponding to one feature; (2) A large number of relational tables. In each table, the process curves (time series) corresponding to one welding operation are stored. In case of json or mat, the two groups are stored in lists of dictionaries or structures, in which each field is a single feature or time series feature.

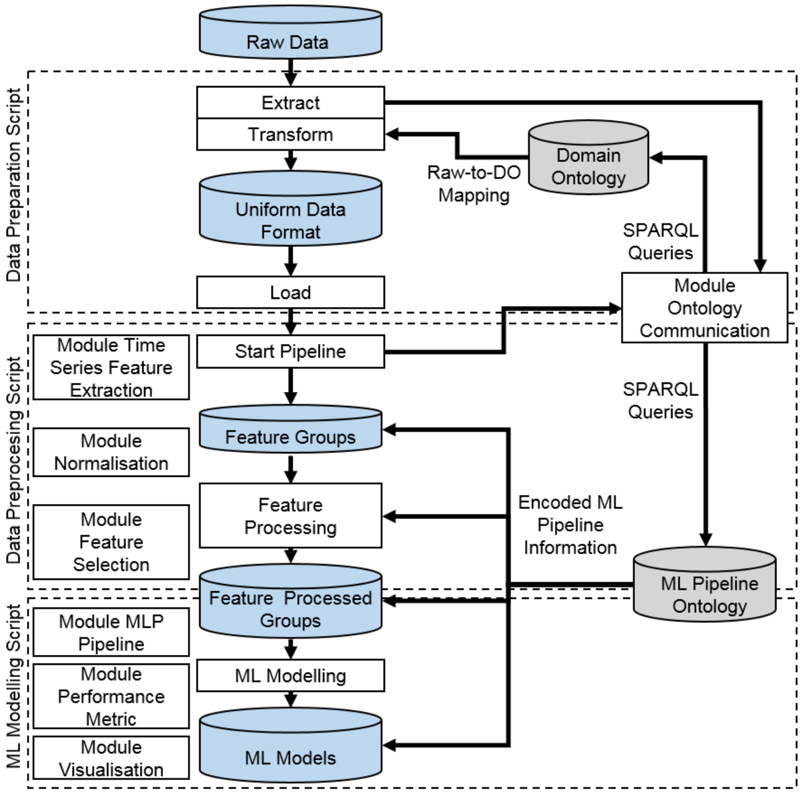


Figure 3.1: Architecture of implementation scripts. The three modules in Data Preprocessing are called in Feature Processing. The three modules in ML Modelling are called in ML Modelling.

Transform. Data need to be transformed to *Uniform Data Format* with seven procedures: combining headers and values, splitting, sorting, matching, padding, merging, and feature renaming. Some procedures are not mandatory under certain circumstances.

- *Combining headers and values.* It is necessary in the case when headers and values are stored separately, e.g. text/csv data or prjz data.
- *Splitting.* A dataset is normally comprised of data of dozens of welding machines. In most analysis, these welding machines are analysed inde-

pendently. Therefore, the big dataset needs to be split into smaller sets, with each only containing data of one welding machine.

- *Sorting*. Welding operations are sorted using the logical-temporal order of *WearCount-DressCount-CapCount*. Time stamps are not always reliable, because manual interference of operations will cause time stamps renewed. *CapCount* is missing for some datasets and therefore an ad hoc *CapCount* needs to be added.
- *Matching*. The various features types (Section 2.3.2) of data are normally stored in different files. Sometimes there exist no *Identifiers* to find the correspondence between data, especially for the two groups, single features and time series. Due to software settings of the welding control, the time stamps of data entries are not perfectly aligned. After various testing, this work suggests to use *WearCount*, *DressCount* as the logical-temporal order to match data. These two features exist in files of both groups. Note these two features cannot identify data entries uniquely, because the *WearCount-DressCount* combination will repeat when a new *electrode cycle* begins. *CapCount* is missing for some datasets, and therefore it cannot be used for matching. A dynamic matching algorithm is needed: (1) narrowing down the search area using time stamps, allowing e.g. one day tolerance as the search area; (2) finding the matching *WearCount-DressCount* combination within the search area for the two groups.
- *Padding with NaN*. For single features without values, *NaN* (not a number) is filled. For time series with different lengths, *NaN* is padded to normalise the lengths. The NaN-padding avoids errors caused by dimension inconsistency.
- *Merging*. Now the correspondence between different feature types is present, and dimension inconsistency is eliminated. Data can be merged into a single dictionary and stored in numpy array or pickle. Data can also be logically merged by adding *Identifiers* in all files and storing them in a pre-designed manner. A typical style for text/csv format is (similar to merge SQL table): (1) All single features are stored in one relational table,

adding one column for the identifier. Some feature values need to be replicated. For example, the meta configurations of all welding programs were stored separately in a small relational table, with each row corresponding to one welding program. It will be repeatedly filled into the merged single feature table of welding operations, since each welding program is executed by a large number of welding operations. (2) All time series features are stored in a series of relational tables in a sub-folder, with all time series files named as the identifier.

- *Feature renaming.* All feature names will be changed to the *Unified feature names* using the Raw-to-DO mapping (raw to domain ontology terms) to simplify the subsequent scripts.

Load. Data in UDF can be easily loaded by reading the json or pickle files or reading the text/csv files using *pandas*.

ML modules. *Uniform data format* allows the subsequent scripts to be standardised and repeatedly used. This work implements frequently used functions in seven modules. Modularisation of ML scripts are essential for readability and ontology-enhanced ML pipelines.

- *ModuleTimeSeriesFeatureExtraction* is written in three levels:
  - (1) The TS feature extraction for a welding operation. The input is a 2D-matrix with shape *[len,sz]*, where k is the TS length after dimension-normalisation and sz is the number of TS features. The exact algorithm can be various: statistic features, geometrical features, segmentation, PCA features, etc.
  - (2) The corresponding feature name generation function for (1).
  - (3) The TS feature extraction for a dataset. It takes a 3D-matrix with shape *[n,len,sz]* as input, where *n* is the number of data tuples in the dataset. The output is a 2D-matrix of shape *[n,sts]*, where *sts* is the number of extracted time series features. Different algorithms written in (1) can be used as plug-in in (3). New algorithms in level (1) can also be easily added.

- *ModuleFeatureSelection* implements a stepwise forward linear regression algorithm using *Dynamic Programming*. In each iteration, it selects one best feature using wrapper method from the appointed dataset, and deliver the rest sub-dataset into the next iteration, and solve the sub-problem repeatedly. The return is an array of the order of selected features and the corresponding feature scores evaluated using correlation coefficient.
- *ModuleNormalisation* implements *Z-score* normalisation and *Min-Max* normalisation, and provides functions of *fit*, *transform*, *inverse-transform* for single feature and *fitts* for time series.
- *ModuleMLPPipeline* implements all functions for a complete MLP pipeline, including loading data, normalisation, denormalisation, constructing MLP architectures according to input, training and testing the models, hyper-parameter tuning, performance evaluation and saving results.
- *ModulePerformanceMetric* implements the metrics of prediction accuracy discussed in Section 2.4.8.
- *ModuleVisualisation* implements frequent visualisation functions in *matplotlib* and *bokeh*, e.g. line-plot and scatter-plot along welding operations or *WearCount* for quality indicators, row-wise independent heatmaps for dataset evaluation.
- *ModuleOntologyCommunication* implements the mechanism of ML pipeline communicating with ontologies. Details please see the next paragraph.

Information retrieval from ontologies. Ontology-enhanced ML pipelines need to communicate with the ontologies to retrieve information encoded in ML pipeline ontology. This is realised by sending SPARQL queries to query ontologies. These scripts are capsuled in the *ModuleOntologyCommunication*.

- *Load graph.* The ontologies serialised in *Turtle* [193] can either be loaded into memory from a *URL* using the *rdflib* package, or be queried online using the *stardog* package.
- *Query.* All encoded information are retrieved using queries. For example, to retrieve the stored ML pipeline ontology, a query *SELECT * WHERE ?s a :MLPipeline.* is sent to the loaded graph. The return will be processed to extract the label of the *class* or the string of *literal*. The return of the example is the label of the ML pipeline stored in the graph: *['MLPipeline'].* Further examples include:
  (1) *SELECT ?FG WHERE:FeaturePreparedLayer :hasMemberFG ?FG*
  with the return: *['QualityIndicator1', 'SFGroup1', 'TSGroup2']*
  (2) *SELECT ?FPAlg WHERE:SingleFeatureGroup1 :hasFPAlg ?FPAlg*
  with the return: *['Maintain'].*

Exploratory data analysis. In this step the data are visualised, conspicuities are examined, and benchmarks can be calculated.

- *Visualisation.* There exist two important types of visualisation. (1) Three features, *WearCount*, *DressCount*, and *CapCount*, are visualised using line-plot with *matplotlib* or *bokeh*. The left y-axis will be *WearCount* and the right y-axis will be *DressCount* and CapCount. The x-axis can be time stamps or number of welding operations. With this visualisation, it is easy to obtain an overview of all dress cycles and electrode cycles. Incomplete cycles are obvious. (2) Any single feature of interest can be visualised using line-plot or scatter-plot. Conspicuities can be detected. A typical example is the Q-Value, as discussed in detail in Use Case 4.2.
- *Conspicuity and Benchmark.* For calculation of trends or benchmarks, the most efficient solution is to use nested dictionaries to store the calculated values, e.g. the trend value of WM1, Prog1, Cap2, Dress2, Wear120 is stored in *trend['WM1']['Prog1']['CapDress202']['Wear120'].*

Data preprocessing. It includes cleaning, dimension-normalisation, feature extraction, normalisation.

- *Cleaning* is to delete features (columns) or data entries (rows) with *NaN* that exceed a threshold.
- *Dimension-normalisation* for time series is to make the TS lengths equal by truncation (deletion), padding (with physically meaningful values, see Section 2.4.2) or resampling (using e.g. *numpy.interp*).
- *Feature extraction* includes various feature engineering strategies implemented in *ML modules*, or feature learning, discussed together in ML modelling.
- *Normalisation* is implemented using *ModuleNormalisation* for *Z-score* normalisation and *Min-Max* normalisation.

ML modelling. All modelling is implemented in four levels.

- *ML architecture construction:* For classic ML, this level is simply to use a function written in established packages, e.g. *scikit-learn*. For feature learning, this level constructs the designated network architecture and returns the constructed architecture.
- *Training and test:* This level capsules one round of training and test in a function. It takes preprocessed data and ML architecture as input, and outputs the ML prediction results for both training and test data.
- *Performance evaluation:* This level takes the prediction results as input, and uses the *ModulePerformanceMetric* to calculate all performance metrics and returns a *DataFrame* of performance summary.
- *Saving results:* This level takes the prediction results and performance summary as input, saves all these results, and visualises the prediction results using line-plot and scatter-plot in *ModuleVisualisation*.

3.2 Implementation with SciXMiner

ML pipeline with SciXMiner. A ML pipeline in MATLAB can execute a series of ML experiments that are designed to achieve one goal. For example, the same scripts run on a series of datasets with different training subsets/feature subsets to analyse the influence of training data amount/feature sets on model performance. The architecture of ML pipeline scripts is organised in three levels: a batch file, experiment instances, and several script components.

Batch file. In this file, the project directory and script directory are specified for SciXMiner to find data and scripts.

Experiment instance. After specification of directories, a series of experiment instances are executed. For each there exist (1) a prjz file (SciXMiner project), (2) a configuration script and (3) a master script.

- *Prjz file.* This is the SciXMiner project that stores the data. The prjz file is similar to a MATLAB structure, which has fields and values. The values can be a single string or float, or matrices, or nested structures.
- *Configuration script.* In this script the number of CPU cores are specified to help the system allocate computing resource. A new directory for the current experiment instance is created with a unique name, by e.g. using the current time in the folder name. A directory (usually the project directory) needs to be assigned to a variable, working directory. The working directory is important for execution of script components.
- *Master script.* This script has two parts. (1) Part I prescribes the hyperparameters, the target output feature, and creates sub-directories for subexperiments of e.g. cross-validation. (2) Part II contains all script components from data preprocessing, ML modelling, to saving results. The ML algorithms coded in SciXMiner can only access global variable namespace, i.e. cannot process data passed into a function internally. Therefore, all script components are directly executed in global variable namespace.

Script components. These are modularised components with different functionalities for data preprocessing and ML modelling. In particular, there exist the following types of components:

- *Creating project backup.* Since all script components are executed in global variable namespace, a backup of the project data needs to be created before any experiments. The backup includes single features (d_org), single feature names (dorgbez), time series (d_orgs), time series feature names (bez_code), output class encodings (code_alle), output class feature names (var_bez), and output class terms (zgf_y_bez).
- *Data preprocessing.* These components include data splitting into training and test sets, deletion of irrelevant features, feature extraction (sometimes this can be omitted if the features are already extracted by Python scripts), univariate or multivariate feature selection. If wrapper method is adopted, the feature selection is coupled with ML modelling.
- *ML modelling.* These components include ML model configuration using the hyper-parameters prescribed by the master script, or selected from the hyper-parameter list in case of sub-experiments for hyper-parameter selection. After that, the model training and testing are executed.
- *Visualisation and saving results.* These components include visualisation of prediction results in line-plot or scatter plot, performance evaluation, and saving all these results in the sub-directory for the current experiment instance.
- *Restoring project backup.* When all components are executed, the project data need to be restored from the backup, so that the next experiment instance can be executed.
- *Hyper-parameter selection.* This component is to visualise and evaluate all performance after all sub-experiments are executed. It is needed when the goal of the experiments is to find the best hyper-parameters.

4 Use Cases

This chapter will demonstrate use cases, validating the methods proposed in Chapter 2 on data collected from simulation, laboratory and production. Relevant rows of the overview table (Table 2.3) are separately displayed again for each use case.

4.1 Spot Diameter Estimation and Guidance for Simulation Data Collection

Overview: This use case studies estimation of welding quality for resistance spot welding process. The target feature of the ML models is the welding nugget diameter. The data are collected from a validated Finite Element Method (FEM) simulation model [158]. This is a supervised regression problem. Since the classification of welding quality into good/bad categories depends on specific welding configurations, the study of regression is more general than classification, and can provide more insights to process experts. Besides, the classification can still be made after regression.

Strategies of data preprocessing are extracting statistic features from time series and feature selection. The ML methods are three classic methods: polynomial regression, "shallow" multi-layer perceptron (with only one hidden layer), k-nearest neighbours. Analysis of the influence of feature sets and amount of training data on prediction accuracy is conducted to gain insights for preparing costly data collection from laboratory and production.

The related publication is [29]. An overview of the corresponding application questions, open questions and methods is listed in Table 4.1.

Table 4.1: Overview of application questions (AQ), open questions (OQ) and methods (M)

AQ	OQ	Methods
AQ2, AQ3, AQ4	OQ1, OQ5	M2.1.1, M2.1.2, M2.1.3, M2.2.4, M2.4.2, M2.4.5, M2.4.8

4.1.1 Question Definition

An enormous amount of data is generated every day in automatic manufacturing. Yet the acquired data do not always contain relevant information (e.g. labels) and thus are sometimes not suitable for data analysis [94]. In resistance spot welding, the level 2 quality indicators (QI2, see Section 2.1.1), such as weld nugget diameter data, are often scarce and can only partially cover quality monitoring. Moreover, additional sensor data, such as welding force and temperature, are not always available because of sensor installation cost and accessibility.

Machine learning enables data-driven methods for more reliable and all-covering quality monitoring for RSW, by replacing QI2 with QI5 generated by ML models. In most cases, the more data are used for training, the higher the chance is to obtain accurate models. However, the cost of data collection in manufacturing is extremely high. It would be optimal to gain some insights of the RSW process before starting a costly data collection plan from laboratory or production, so that data collection can have some references. Interesting questions include, e.g. how many data are necessary for successful machine learning modelling? What features are more informative?

This section uses physics-based simulation data generated with a verified Finite Element Method (FEM) model [158] to address the insufficient data problem and the three data challenges (Section 2.1.3). Concretely, *three questions* regarding the first two data challenges in collecting costly labelled data will be studied:

- Q1: How many labelled data to collect first? (Challenge 1)
- Q2: Which sensors should be installed first? (Challenge 2)
- Q3: Which precision level should the sensors and measurements of welding spot diameters have? (Challenge 2)

4.1.2 Data Description

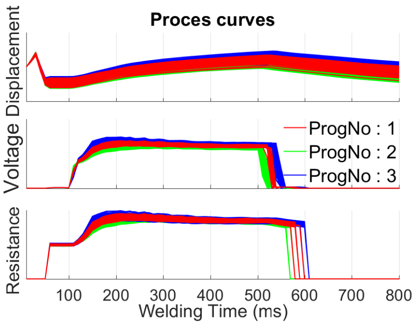
Simulation scenario. The most normal conditions in production, i.e. only random variation without spatter or other disturbances, are selected as the simulation scenario for the data studied in this use case. This simple scenario consists of one welding machine, one type of worksheet pair with identical nominal sheet thickness and material, and three welding programs for three different target spot diameters (Figure 4.1). A total of 13,952 welding spots with diameter measurements are simulated, which contain 14 electrode cycles.

The FEM simulation models mechanical effects (elastic-plastic deformation and thermal expansion of chassis parts), thermal effects (temperature change and heat transfer) and electrical effects (electric current density and electric potential field), taking into account non-linear changes of material and contact properties (such as electrical and thermal contact conductivity). Strong interactions between all three fields result in a very dynamic process behaviour and a multi-field coupled simulation.

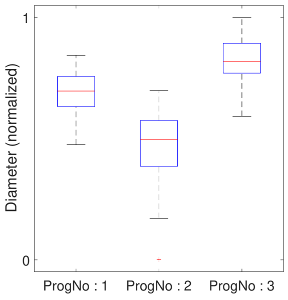
Features. For each weld spot, there are two types of data:

- *Time series* (or process curves): In total, 20 time series were simulated, including two types.
  - *Process input curves* are input time series of the FEM simulation model, such as electric current (I), and force applied on the electrode (F).
  - *Process feedback curves* are output time series of the FEM model, such as voltage (U), resistance (R) , displacement (s) of the electrode, temperature (T) of certain measuring positions, etc.
- *Single features*: They are nominal and measured geometry or material properties of the caps and worksheets, the number of spots welded, posi-

tions of the welding spots, temporal structure features (Section 2.1.2), etc. There exist up to 235 single features in the simulation dataset.



(a)



(b)

Figure 4.1: (a) Process curves for the three welding programs. ProgNo indicates the program number. The welding programs prescribe the way the welding process is performed, by specifying the process curves and some additional welding parameters [29].
(b) Boxplot of the diameters for the three welding programs [29].

4.1.3 Experiment Settings

The approach for exploring answers to the three questions centres on training machine learning models with different data subsets and comparing their performance.

Data splitting according to temporal structure. As mentioned in Section 2.4.1, data splitting should be performed according to some complete units of a time level. In this study, the *electrode cycle level* will be used as complete time units. Data generated with 9 electrode caps (7973 data points) are used for training and selecting hyper-parameters, denoted as “trainingx data”. Data from the remaining 5 electrode caps (5979 data points) are used for testing.

Data splitting into subsets of different training data numbers. ML models are trained with different number of data tuples to study how the number of training data influence the model performance (Section 2.2.4), for answering Question 1 in Section 4.1.1. The trainingx dataset is split into seven subsets (Table 4.2), with the condition that the larger subsets always contain the smaller subsets to avoid the random effect of data on model performance, e.g. D_2 contains all data in D_1, D_3 contains all data in D_2.

Table 4.2: Data splitting into seven subsets of different training data number

Dataset name	D_1	D_2	D_3	D_4	D_5	D_6	D_{all}
#Trainingx data	100	250	500	1000	2000	5000	7973
#Test data	5979	5979	5979	5979	5979	5979	5979

Data splitting into subsets of different feature sets. For answering Question 2, ML models are trained with different feature subsets to investigate the influence of each feature subset on the performance (Table 4.3). As introduced in Section 2.2.4, the features are split into Feature Set available in production, in the lab with low cost, in the lab with high cost, and all features in simulation. The feature set splitting is applied to each of the seven subsets of training data number, resulting in a total of 28 subsets ($7 \times 4 = 28$).

Table 4.3: Data splitting into four subsets of different features

Feature set	$FeatSet_{prod}$	$FeatSet_{lablow}$	$FeatSet_{labhigh}$	$FeatSet_{all}$
#Features	15SF, 4TS	16SF, 10TS	29SF, 14TS	235SF, 20TS

Generating additional noisy datasets. For answer Question 3, noise is added to the 28 subsets, resulting in an additional set of 28 noisy datasets. The noise levels are a best engineering guess derived from discussions with process experts and measurement experts (Table 4.4). In total, $7 \times 4 \times 2 = 56$ subsets are used to train and test machine learning models and compare their performance.

Feature engineering and selection. The feature engineering strategies applied to the time series are extracting 8 statistic features, including min-

Table 4.4: Data splitting into four subsets of different features

Features	Added noise
Sensor data (time series and single features)	Gaussian noise with 2% standard deviation for every single sample point
Diameter measurements	Gaussian noise with 0.1 mm standard deviation

imum, maximum, minimum position, maximum position, mean, median, standard deviation, and length.

These extracted features combined with all other single features are then evaluated and selected using step-wise forward selection (Section 2.4.3), starting with the evaluation of each single feature, and incrementally adding more features. The features resulting in models with the best regression accuracy (RMSE) for the prediction of spot diameters are selected. Results show that in most cases, the model performance increases as the number of selected features increases. For some models the performance does not increase any more from about 10 features. For some models the performance does not improve any more up to 20 selected features. To make the comparison fair and reduce computation time, all models are trained with 20 selected features.

ML methods and hyper-parameter tuning. Three machine learning methods, *polynomial regression*, *neural networks* and *k-nearest neighbours* are studied. The hyper-parameters are chosen with 5-fold cross validation with the largest trainingx data (training set of D_{all} with 7973 training data points) and the Feature Set of Production ($FeatSet_{prod}$). The resulting hyper-parameters are shown in Table 4.5.

4.1.4 Results and Discussion

Figure 4.2 illustrates the performance on test set of best polynomial models (Polynomial1) trained on non-noisy subsets of different training data number and feature sets, evaluated with four performance metrics.

Table 4.5: ML models and hyper-parameters selected through 5-fold cross validation. Note that some polynomial models with higher degree than 1 do not show statistically significant improvement than first order polynomial models. According to the principle of Ockham's razor, polynomial order of 1 is selected.

Machine learning model	**Hyper-parameter**	**Value**
Multivariate polynomial regression	Polynomial order	1
Multi-layer perceptron (one hidden layer)	#Neurons in the hidden layer	16
K-nearest neighbours	k (#Neighbours)	3

Feature sets. Comparing models trained on different feature sets reveals, as expected, when more features are available for training, the models also have better performance in testing in most cases: $P_{all} > P_{labhigh} > P_{lablow} > P_{prod}$ (P indicates performance).

Training data number. Comparing models trained on different training data subsets shows that, when the training data number is greater than 250, the performance improves insignificantly as the training data number further increases.

Performance metrics. Comparing the performance represented by different metrics, it can be seen that RMSE, Correlation Coefficient and Error Within 5% have similar trends. This is also the case for the multi-layer perceptron models (Figure 4.3) and k-nearest neighbour models (Figure 4.4). Error Within 10% cannot differentiate the models trained with different data number and feature sets, since they always show very good results. Therefore, only Error Within 5% will be compared in the following discussions.

ML methods. Comparing models trained using different machine learning methods can be made by comparing Figure 4.2c, Figure 4.3, and 4.4, or in Figure 4.5. These figures reveal that the performance of the models built with three machine learning methods is: Polynomial $\approx$ MLP > KNN. Figure 4.3 shows that MLP16 tends to overfit for the $FeatSet_{lablow}$ and $FeatSet_{labhigh}$. In this regard, Polynomial1 is more advantageous.

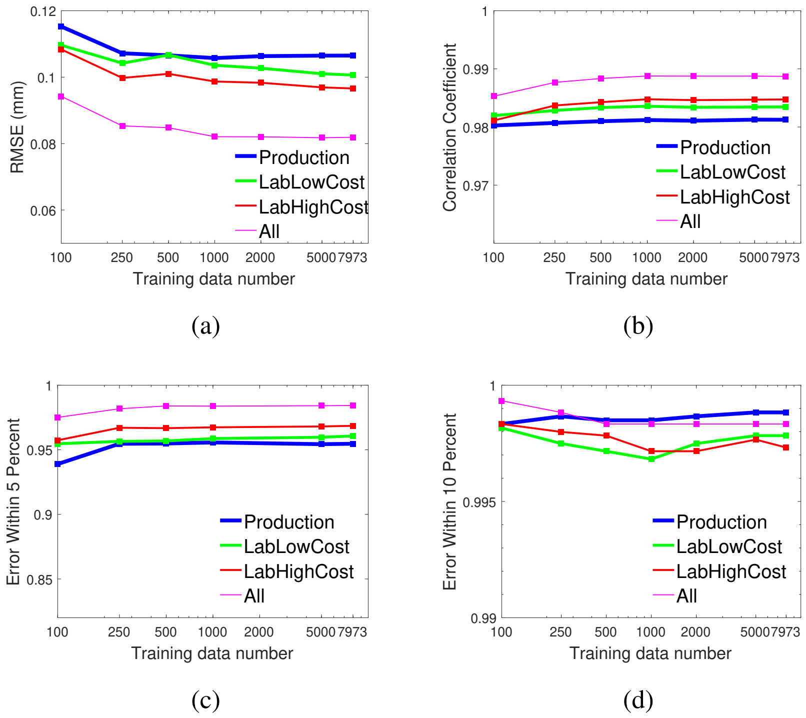


Figure 4.2: Comparison of performance on test set of Polynomial1 models trained on non-noisy subsets of different training data number and feature sets [29]. Note that in (d) Error Within 10% cannot differentiate the models, since all models are extremely good and have more than 99% predictions with less than 10% errors.

Noise in data. Comparing models trained with non-noisy data and noisy data can be made by contrasting Figure 4.2 c and Figure 4.6. The comparison suggests the performance deteriorates strongly with noisy data. Generally speaking, the difference between models trained on these three feature subsets, $FeatSet_{prod}$, $FeatSet_{lablow}$, and $FeatSet_{labhigh}$ is insignificant.

4.1.5 Conclusion and Outlook

Conclusion. Simulation data can provide a large amount of economically generated labelled data, and features that would be otherwise unavailable.

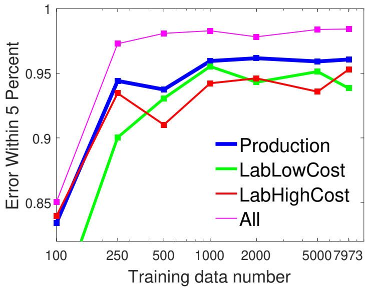


Figure 4.3: Testing results MLP16 on non-noisy datasets [29]

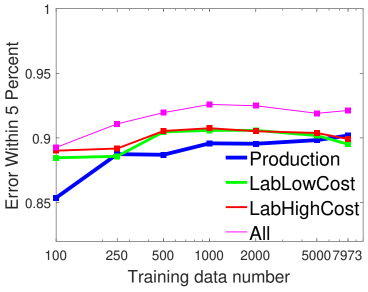


Figure 4.4: Testing results KNN3 on non-noisy datasets [29]

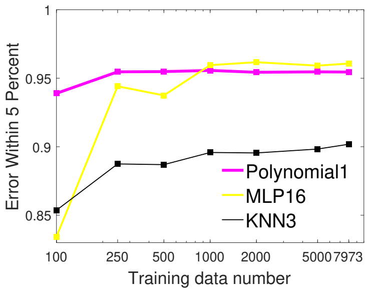


Figure 4.5: Testing models trained on non-noisy Production Set [29]

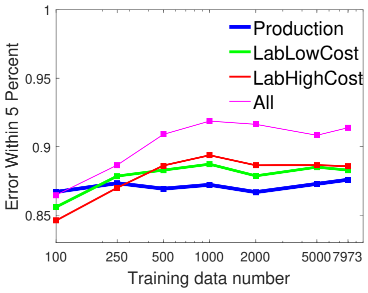


Figure 4.6: Testing Polynomial1 on noisy datasets [29]

These data contain no or low measurement error. As a result, it is possible to guide the collection of costly data from production or laboratory with the insights gained from analysing simulation data. Reviewing the starting point of the three questions, the following insights can be interpreted from the analysis results.

- Answer to Question 1: Starting with a collection of 250 labeled data points.
- Answer to Question 2: Features available in Production can already be useful.
- Answer to Question 3: Measurement uncertainty can lead to significant deterioration of the prediction performance. In our case, precision better than Gaussian errors of 2% standard deviation for sensors and 0.1mm

standard deviation for diameter measurements, respectively, is recommended.

It is important to note that the conclusions regarding the necessary number of data points and features are highly dependent on the dataset, and the simulation scenario from which the data are collected. It is therefore safe to say that the results can be generalised over the simulated situations. They can serve as a first guidance for data collection, but should NOT be assumed as valid in general. Since the selected simulation scenario of one welding machine and three welding programs is a simple scenario, these conclusions may not hold for other scenarios or for complex production data. The optimal data collection plan may vary depending on the complexity of the actual situations compared to the simulated situation.

Outlook. In future research, several topics can be explored for combining the simulated data and laboratory or production data.

- The first option is to use laboratory data or production data to further improve and fine-tune the simulation model so that realistic physical effects and authentic variance of influencing factors can be rebuilt in the simulation. This realistic simulation data can be used for further data analysis.
- A further step is to apply the model trained on simulated data on laboratory or production data to see the generalisability of the models.
- A third option is to transfer learning, i.e. to pre-train a model on simulated data, then to fine-tune the model on laboratory data or production data, and to see how much improvement in performance can be gained.
- The improved simulation model and generated data can be used for production condition evaluation before a real production process begins, so that the influence of welding parameters changes, such as welding gun properties, material properties, distance between the worksheets, can be systematically evaluated.

- Alternative feature extraction methods (such as features based on domain knowledge or Principal Component Analysis) can be analysed to evaluate the difference of prediction accuracy using different feature extraction methods.
- As actual production conditions may change due to a change in supplier, new engineering design, etc., it is always questionable to what extent a model trained on historic or lab data can be deployed on production data in the future, which constitutes a general problem in manufacturing. This issue of concept drift remains an open question.

This use case strives to gain insights from data analysis for guidance of data collection. The iterative process data-collection $\rightarrow$ data analysis $\rightarrow$ data-analysis guided data collection $\rightarrow$ data analysis is an attempt to combine data collection and analysis (Figure 2.9). A similar practice would be particularly necessary in fields where limits on the amount of labelled data, features, and covering situations is a problem of interest.

4.2 Evaluation of Production Datasets

Overview: This use case demonstrates using multiple metrics and visual analytics techniques to gain a quick overview of large manufacturing datasets, including categorical features (ProgramNumber), numerical features (Q-Value), features with multi-level temporal structures (time series, WearCount, DressCount, etc., Section 2.1.2). In this example, the data are collected from various welding machines and their Q-Value behaviour is inspected for each machine. Similar practices can be done for other units (e.g. chassis part types, worksheet thickness combination types *STC*), and other quality indicators (e.g. process stability factor, spatter occurring time). Visual analytics is used to identify conspicuous welding machines, welding programs, or dress cycles that are subject to risks of quality failures. Visual analytics refers to methods and practice of information visualisation for analytical reasoning [194], and has found a wide range of applications [195, 196]. An overview of the corresponding application questions, open questions and methods is listed in Table 4.6.

Table 4.6: Overview of application questions (AQ), open questions (OQ) and methods (M)

AQ	OQ	Methods
AQ1	OQ3, OQ4	M2.2.5

4.2.1 Question Definition

ERW processes produce a large volume of data. It is therefore very desired for process experts to gain a quick overview of the data collected from all welding machines, to identify which welding machine, which welding program or which dress cycle are conspicuous and may subject to risks of quality failures. These conspicuities include deviation from the expected behaviour, large scattering or outliers (Section 2.2.5).

The users can choose any single feature as the criteria, e.g. *Process Stability Factor*, *Resistance Mean*. Among these, *Q-Value* is especially important, and will be used as the criteria in this use case for identifying conspicuities. Q-Value, developed by Bosch Rexroth through engineering know-how and long-time experience, is calculated after each welding operation, based on statistic features (e.g. mean, maximum) from sensor data. The optimal Q-Value is one and any value that deviates from one indicates quality deterioration or inefficiency.

In dataset evaluation, an overview of the general information (e.g. #welding operations, #welding programs), and the behaviour of *Q-Value* should be presented to the users (e.g. process experts, data scientists), to identify potential problematic welding machines. These problems should then be attributed to specific welding programs or dress cycles, to narrow down the search realm for root causes of problems. Appropriate visualisation of the general information and Q-Value behaviour should be illustrated in an condensed and intuitive way to enable comprehensive evaluation of datasets (i.e. welding machines in this use case) and conspicuity (outliers in some cases).

In this use case, the general information of a sample welding production line is summarised in heatmaps by using the metrics proposed in Section 2.2.5. The heatmaps then serve as a guidance to identify potentially problematic welding machines, welding programs or dress cycles. Illustration of the Q-Values of the area after narrowing down confirms that the located data snippet is indeed subject to risks of quality failure.

4.2.2 Data Description

The sample dataset has two welding production lines, which consist of 22 welding machines, responsible for 195 unique welding programs (welding spot types), including 263 fields, 53.2 million records and 1.4 billion items in total. In the use case, a data tuple (Section 2.3.2) corresponds to a welding operation, and contains *Single Features* and *Time Series*.

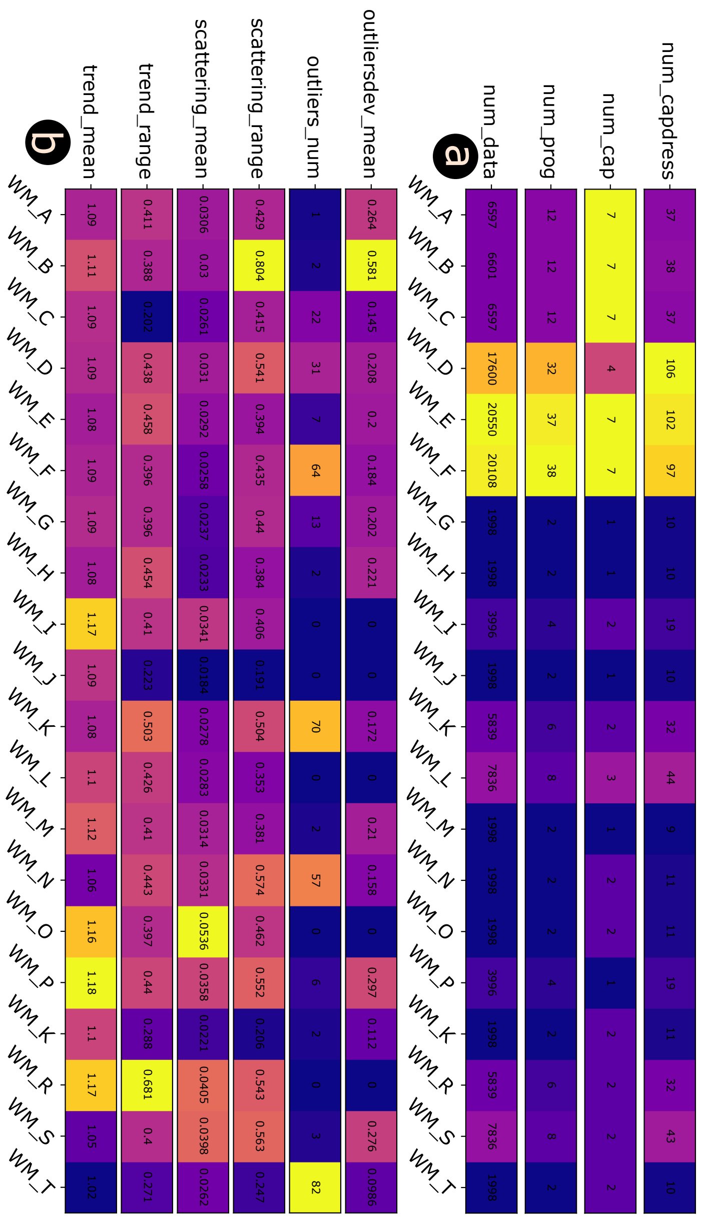


Figure 4.7: (a) General information of all welding machines.
(b) Overview of dataset metrics in the time level of welding machine. The colour map is a smooth continuum through yellow (conspicuously large values), purple (middle values), and blue (small values, thus with de-emphasised text) hues.

4.2.3 Experiment Settings

The information in different protocols, databases, and files are transformed into the *Uniform Data Format* (UDF) (Section 2.3) before evaluation.

Firstly, a general information overview will be given to the 22 welding machines, including #welding operations, #welding programs, #electrode caps and #dress cycles.

Secondly, the six overall metrics of trend, scattering and outliers of Q-Value (Table 2.6) of these 22 welding machines will be visualised by heatmaps, using colour contrast to help the users to identify conspicuity quickly. Moreover, the discrepancy metrics with respect to welding programs and dress cycles (and electrode caps for welding machines with many electrode cap changes) will be illustrated also by heatmaps, to give users further information of whether there exist obvious discrepancies between welding programs or dress cycles. This use case adopts the *global trend* in Section 2.2.5 for calculation of the metrics, assuming that the data behaviour of Q-Values should be independent from *DressCount*. The assumption is based on the domain knowledge that nominally welding quality in different dress cycles should be the same.

After that, the users dive into specific welding machines where the dataset metrics are conspicuous, which indicate these machines may be subject to risks of quality failures. The six overall metrics of trend, scattering and outliers of Q-Value (Table 2.6) are then visualised by heatmaps for each welding program, each dress cycle, and electrode cap. Thus the users can narrow down the search area to specific welding programs or dress cycles. The colours in the heatmaps adopt the "plasma" colour map [197], with a pleasant smooth continuum through yellow (conspicuously large values), purple (middle values), and blue (small values) hues. In the next step, the Q-Values, trend, scattering and outliers of noticeable welding programs are visualised using scatter plots and line plots. Two types of such Q-Value plots exist (both types of such plots are overlapped with line plot of trend):

(1) scatter plot of Q-Value along welding operations, which is suitable for identifying conspicuous dress cycles, and (2) scatter plot of Q-Value over WearCount, which is suitable when there exist too many data tuples.

4.2.4 Results and Discussion

The visualisation results show that WM_B, WM_I, and WM_R are conspicuous and will be discussed in this section. WM_O and WM_T are also conspicuous and are discussed in Appendix A.3 due to space limit.

General information overview. An overview of the production line data is illustrated in a heatmap in Figure 4.7a. Note each row is an independent heatmap with its own colour scale. The bottom row "num_data" summarises the #data tuples, with each data tuple corresponding to a welding operation. The row "num_prog" indicates #welding programs of each welding machine. Welding programs are designed for different worksheet combinations and spot positions on the chassis. Different welding programs therefore have distinctive dynamics. The number of welding programs can therefore be used as an indicator for data complexity. It can be observed that the number of data tuples correlate with the number of welding programs. This is caused by the settings of data collection, that 999 data tuples were collected from each welding program. The row "num_cap" is #electrode caps in the dataset. The row "num_capdress" is #dress cycles in the dataset. In summary, the most complex datasets are collected from WM_A (Welding Machine A), WM_B, WM_C, WM_D, WM_E, and WM_F.

Metrics on the welding machine level. The six Overall Metrics of all welding machines are illustrated in Figure 4.7b. These six metrics from bottom to top are mean values of the trend $Trend_{mean}$ ("trend_mean"), range of the trend $Trend_{range}$ ("trend_range"), mean values of the scattering $Scattering_{mean}$ ("scattering_mean"), range of the scattering $Scattering_{range}$ ("scattering_range"), #outliers ("outliers_num"), and mean values of the outlier deviation $OutlierDev_{mean}$ ("outliersdev_mean").

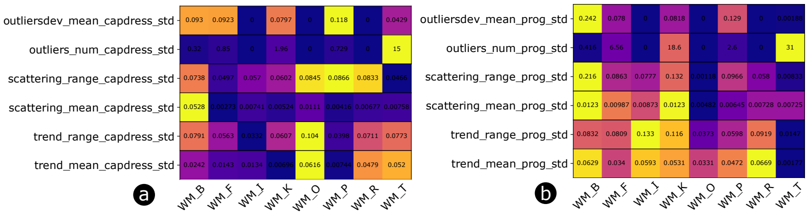


Figure 4.8: Discrepancy metrics of conspicuous welding machines. Yellow: conspicuously large values, purple: middle values, blue: small values with de-emphasised text.

It can be observed that WM_B, WM_F, WM_I, WM_K, WM_O, WM_P, WM_R and WM_T have high values in some metrics and may be subject to risks of quality failure. In particular, WM_B has a high level of $Scattering_{range}$ and $OutlierDev_{mean}$. WM_F, WM_K, WM_N, and WM_T have a high level of $\#Outliers$, indicating large local deviation, which could be caused by problematic chassis parts or unstable process settings. WM_I, WM_O, WM_P and WM_R have a high level of $Trend_{mean}$ (recall that the optimal Q-Value is 1), indicating more energy was spend on welding than the optimal setting, because Q-Value is positively correlated to energy input, according to domain knowledge. WM_O has a high level of $Scattering_{mean}$, indicating complex Q-Value behaviour, which could be caused by unstable process settings or control behaviour. WM_R has a high level of $Trend_{range}$, indicating a relatively large trend variation, which could be caused by strong wearing effects.

The six Discrepancy Metrics with respect to electrode dress cycles and welding programs are illustrated in Figure 4.8. The bottom row "trend_mean_capdress_std" in Figure 4.8a is the standard deviation of the $Trend_{mean}$ calculated for each electrode cap, reflecting the discrepancy of $Trend_{mean}$ across different dress cycles. The other metrics are defined similarly, to reflect the discrepancies of $Trend_{range}$, $Scattering_{mean}$, $Scattering_{range}$, $Outliers_{num}$, $OutliersDev_{mean}$ across dress cycles. These metrics together reflect the discrepancies of Q-Value behaviour across dress cycles. Metrics to reflect discrepancies of Q-Value behaviour across welding programs are defined similarly and visualised in Figure 4.8b.

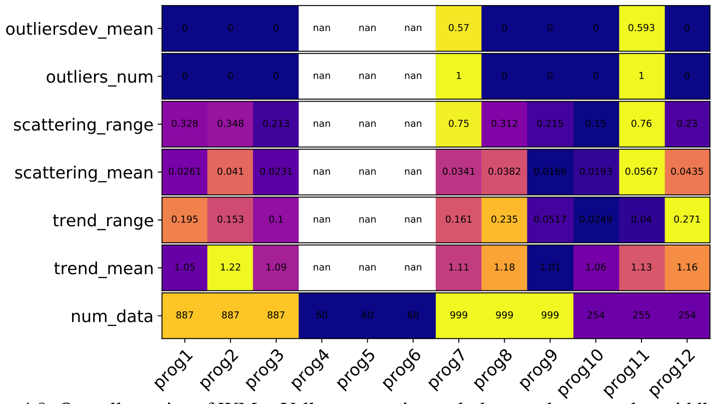


Figure 4.9: Overall metrics of WM$_B$. Yellow: conspicuously large values, purple: middle values, blue: small values, thus with de-emphasised text. For Prog4, Prog5 and Prog6 there exist not enough data tuples to estimate the trend (see paragraph of Estimation of Trend in Section 2.2.5).

Identifying conspicuity of Welding Machine B. WM$_B$ has the highest level of $Scattering_{range}$ and $OutlierDev_{mean}$, while a very low $Scattering_{mean}$. Recall that outliers are a special type of scattering with large deviations (Section 2.2.5). WM$_B$ therefore may have actually low level of scattering, but the high $Scattering_{range}$ is caused by the two outliers. Figure 4.8 shows the discrepancies of $Scattering_{range}$ between welding programs are more pronounced than that between dress cycles, and supports this postulation. Diving into the metrics of WM$_B$ in Figure 4.9, it can be seen that there exist only two outliers in Prog7 and Prog11, whose $Scattering_{range}$ are very high. To interpret from the domain view, WM$_B$ should be working in a relatively stable condition. The two local outliers should be attributed to anomalous chassis part or welding spots, not the welding machine. Note that for Prog4, Prog5, and Prog6 there exist so few data tuples (60 for each) that a reliable estimation of trend is not possible (see paragraph of Estimation of Trend in Section 2.2.5). The metrics for these three programs are therefore not calculated and shown as nan (not a number). In a later machine learning analysis, the dynamics of these programs will more be extrapolated from other welding programs.

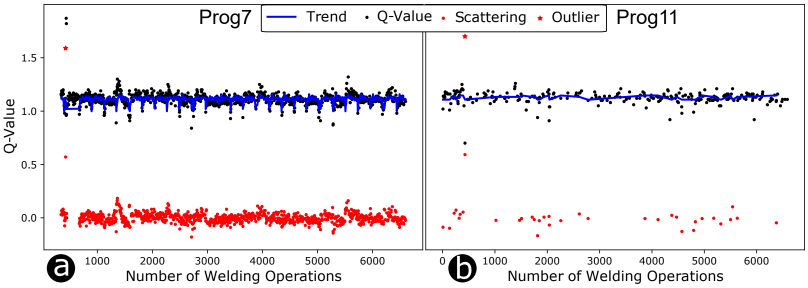


Figure 4.10: Q-Value along welding operations of WM_B for Prog7 (a) and Prog11 (b). Note in (a) the trend is missing for some points at the beginning due to too few data tuples, and therefore is replaced by a straight line.

After narrowing down the search area for conspicuity on Prog7 and Prog11, the Q-Value along welding operations can be visualised and presented to the users in Figure 4.10. It is quite clear that WM_B is relatively stable except for the two outliers at the starting dress cycles. The postulation is finally confirmed. Note that the data tuples in Figure 4.10b are much sparser than that in Figure 4.10a because Prog11 has only 255 data tuples in the collected dataset. In Figure 4.10a there also exists a small area of the starting dress cycles, where there are no scattering plotted. The reason is also too few data tuples for a reliable estimation of the trend.

Identifying conspicuity of Welding Machine I. WM_I has a high level of $Trend_{mean}$ (Figure 4.7b). A closer view on the metrics of WM_I (Figure 4.11a) reveals that the high levels concentrate on Prog1 and Prog5. After narrowing down the search area on Prog1 and Prog5, the Q-Value plots of these two programs in Figure 4.11b and Figure 4.11c show that almost all Q-Values of these two programs are above one (the optimal value), which could

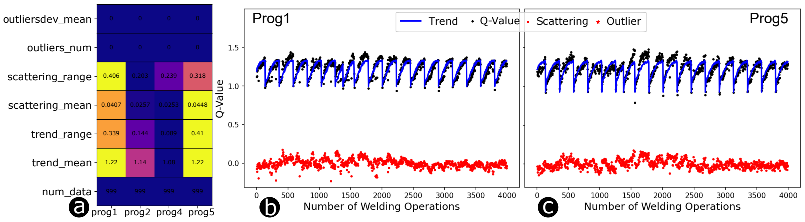


Figure 4.11: Metrics and scatter plots of Q-Values of WM_I. Yellow: conspicuously large values, purple: middle values, blue: small values, thus with de-emphasised text.

indicate energy inefficiency or higher security assurance, since the more energy flows into welding, the stronger the connection will likely be. Among which, some Q-Values are exceptionally high in the middle dress cycles, after a very short dress cycle, which indicates that the electrode cap is changed before the WearCount reaches the normal dress cycle length (around 220).

Identifying conspicuity of Welding Machine R. WM_R has a high level of $Trend_{mean}$ and the highest $Trend_{range}$. The overall metrics with respect to welding programs in Figure 4.13 reveal these high $Trend_{range}$ concentrate on Prog3, Prog6 and Prog7. Besides, Prog3 also has a relatively high level of $Scattering_{mean}$ and $Scattering_{range}$. The scatter plots of Q-Values over WearCount in Figure 4.12 confirm this. The Q-Values are almost always above one for these three welding programs, indicating inefficiency. Especially at the later stages of the dress cycles, the trend of Q-Values fluctuates and the scattering increases significantly for Prog3 (Figure 4.12a). The trend of Q-Values rises constantly for Prog6 and Prog7 (Figure 4.12b and Figure 4.12c), increasingly deviating from the optimal value of one, which could be caused by relatively strong wearing effects.

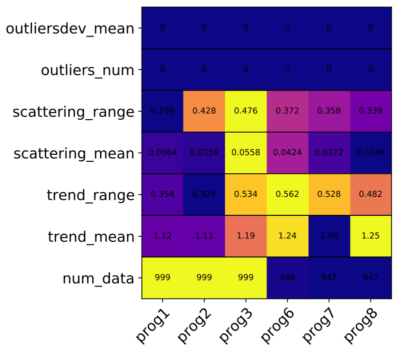


Figure 4.13: Metrics of WM_R. Yellow: conspicuously large values, purple: middle values, blue: small values, thus with de-emphasised text.

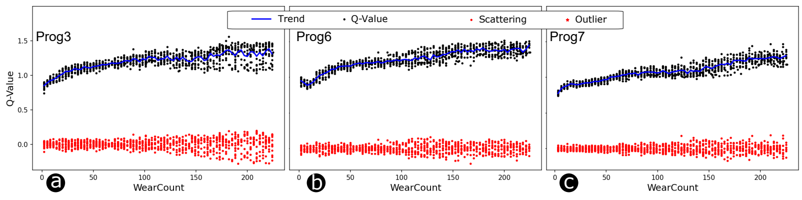


Figure 4.12: Q-Values over WearCount of WM_R

4.2.5 Conclusion and Outlook

The conspicuous welding machines, welding programs and dress cycles are identified through metrics and visualisations. Although there exist no absolute criteria for formally defining conspicuities, it is still possible to narrow down the search to areas of interests and help the users to gain insights and evaluate the datasets with help of the metrics and illustrations. A direction for future study can be to figure out the root causes responsible for these conspicuities.

Because the *global trend* is adopted, there exist not enough data for estimation of trend of some welding programs, causing missing values of trend values (e.g. WM_B). The global trend assumes the data behaviour are independent from *DressCount*, which is the optimal situation, but in actual data some conspicuities are concentrated on specific dress cycles (e.g. WM_T). Adopting the *local trend* or *ML model trend* may obviate these problems by assuming data behaviour can be different in different dress cycles. This remains as a direction for future study.

WM_G and WM_I are selected for a further step analysis in this thesis. Their data are complete, and trend estimations are also complete. Their complexity range from low (WM_G with 2 welding programs) to moderate (WM_I with 4 welding programs). Their $Trend_{range}$ and $Scattering_{mean}$ are moderate. WM_G reflects a rather normal welding condition. It is thus suitable to test the performance of developed ML pipelines on normal welding conditions. WM_I has a high level $Trend_{mean}$. It is thus suitable to test the ML pipelines on a slightly more complex condition. Besides, these two welding machines are also of special interest in a research project in our group. WM_R, with 6 welding programs, has a high level of $Trend_{mean}$ and the highest $Trend_{Range}$. Its data are complete and the estimation of trend is also complete. WM_R is therefore optimal for case study with relatively high complexity.

In the following sections, WM_G, WM_I, and WM_R will be referred to as WM1 (Welding Machine 1), WM2 and WM3 for simplicity.

4.3 Quality Prediction with Feature Engineering and Feature Learning

Overview: This use case studies predicting (or forecasting) quality of resistance spot welding. The target output of the ML models in this use case is the quality indicator, Q-Value of a future welding operation. Regression will be studied instead of classification for the same reason as in Use Case 4.1. The studied data are collected from WM1 (or WM_G) and WM2 (WM_I) in Use Case 4.2, since their data are complete and representative (Section 4.2.5). Two types of machine learning methods are compared to gain insights into suitability of ML methods: feature engineering and feature learning. The following publications are based on/related to this use case: [169], [170], [171]. An overview of the corresponding application questions, open questions and methods is listed in Table 4.7.

Table 4.7: Overview of application questions (AQ), open questions (OQ) and methods (M)

AQ	OQ	Methods
AQ3, AQ5	OQ4, OQ5, OQ6	M2.1.1, M2.1.2, M2.4.1, M2.4.2, M2.4.7

4.3.1 Question Definition

Previous studies have focused on estimating the welding quality after the welding operation (Section 1.4.4). Predictive quality monitoring with ML for ERW has been little discussed. This section demonstrates machine learning (ML) methods for **predicting** the welding quality, Q-Value, (Section 2.1.1) before the actual welding process happens (Figure 4.14). Figure 4.14 illustrates the envisioned workflow of welding with the prediction function incorporated in the welding system. If the quality of the next welding operation is predicted to be inadequate (largely deviating from the optimal value of 1), necessary measures can be undertaken to prevent quality failures, e.g. performing electrode dressing, or adapting the reference curves

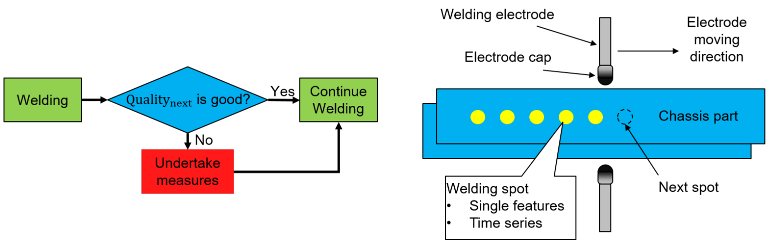


Figure 4.14: Envisioned welding workflow with quality prediction. The welded spots between the worksheets are shown on the worksheets for illustration purpose [170].

in the welding control. Two types of ML methods categorised as *Feature Engineering* (FE) or *Feature Learning* (FL) are developed and compared to analyse what ML methods are suitable for ERW (Section 2.4).

The formal representation of the defined question is to find a function between the available information and the Q-Value of the next welding operation Q_{k+1}, shown in Equation 4.1, where $X_1,...,X_{k-1},X_k$ are data tuples (including single features and time series) of welding operations from time step 1 to step k, and SF^*_{k+1} are known features of the next welding operation (e.g. welding program).

$$Q_{k+1} = f(X_1,...,X_{k-1},X_k,SF^*_{k+1}). \tag{4.1}$$

In the following text, the subscripts $1,...,k-1,k,k+1$ will be replaced by $1,...,pre2,pre1,next1$ for a better understanding:

$$Q_{next1} = f(X_1,...,X_{pre2},X_{pre1},SF^*_{next1}) \tag{4.2}$$

4.3.2 Data Description

Two example datasets collected from WM_1 (or WM_G) and WM_2 (WM_I) in Use Case 4.2 are studied in this use case (Table 4.8). The prepared data contain 164 *Single Features* and 4 effective *Time Series* (Section 2.3.2). Four reference process curves (time series) are excluded from the analysis since

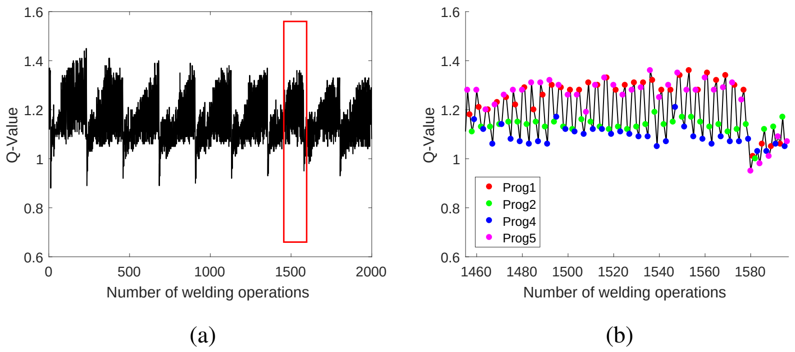


Figure 4.15: (a) Q-Value along number of welding operations for Welding Machine 2 (partially shown). The red rectangle indicates the area for a closer look in (b). (b) Welding operations with different welding programs are performed for spots of different welding positions on the car part, thus often possessing different dynamics, e.g. the means of Q-Values are different [170].

in this use case the reference curves for the same program are always the same, and their information are therefore represented by the welding programs.

The behaviour of Q-Value is illustrated in Figure 2.2 for WM1 and Figure 4.15 for WM2, with x-axis being number of welding operations. This strong periodicity of Q-Values confirm that the example datasets have temporal dependencies described in Section 2.1.2.

4.3.3 Experiment Settings

FE- and FL- based ML pipelines of two different feature settings give four pipelines, and two complexity levels of each pipeline result in eight ML models in total (Table 4.11).

Table 4.8: Two example datasets for prediction of the next Q-Value. Prog: welding programs. DT: data tuples. *tr* : training set, *val*: validation set, *tst*: test set.

Dataset	Welding machine	#Prog	#*DT*	$\#DT_{tr}$	$\#DT_{val}$	$\#DT_{tst}$
Dataset 1	Welding Machine 1	2	1998	1598	200	200
Dataset 2	Welding Machine 2	4	3996	3127	448	421

Four pipelines. The two basic pipelines are: FE-LR and FL-LSTM (Figure 4.16). Apart from the two classic pipelines, it is also very interesting to see what would be the effect if these two pipelines are mixed up, resulting in two more pipelines: FE-LSTM and FL-LR.

Data splitting. The data splitting in this use case follows a classic way, that is to split the dataset into training, validation, and test set in a ratio of 0.8 : 0.1 : 0.1, without considering the temporal structures.

ML pipeline of feature engineering in two steps. The ML pipelines of feature engineering are FE-LR or FE-LSTM. Experiments with two complexity levels, namely the *Base* version and the *Full* version, are designed to test feature engineering on time series or both on single features and time series.

Base is to perform only FE on time series, represented by the pipeline shown in Figure 4.17a after dropping the optional boxes. Padded time series features (Padded TS) are raw time series processed by length normalisation using padding. The padded TS on the *welding time level* are then processed by feature engineering to be reduced to time series features engineered (TSFE) on the *welding operation level*. The TSFE, the same as that in Use Case 4.1, include 8 statistic features of minimum, maximum, minimum position, maximum position, mean, median, standard deviation, and length, serving as supplementary information to the *ProcessCurveMeans* in RawSF. Note that the reference curves of the same welding program are normally identical (unless they are manually changed by process experts), so that no TSFE are generated from them to avoid redundancy. The information of welding programs is captured in *Full*. RawSF and TSFE are

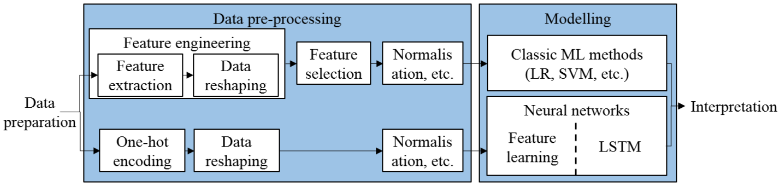


Figure 4.16: Machine learning pipeline of feature engineering and feature learning

concatenated and reshaped. After that, they can directly be fed into LSTM, or fed into LR after flattening and feature selection.

Full is to perform further FE on time series and single features, represented by the pipeline shown in Figure 4.17a with the optional boxes. The raw single features go through feature engineering, resulting in engineered single features (EngSF) (Section 2.4.2). The RawSF, EngSF and TSFE are concatenated and further processed by *Advanced feature engineering with respect to ProgNo* (Section 2.4.2), resulting in engineered features considering program number (EngF_Prog). After that, they are reshaped for modelling, and follow the same procedures of *Base*.

ML pipeline of feature learning in two steps. The ML pipelines of feature learning are FL-LR or FL-LSTM. Two steps of experiments are to test feature learning on time series or both on single features and time series.

In the *Base* pipelines FL is only performed on time series, represented by the pipeline shown in Figure 4.17b after dropping the optional boxes. The padded TS are first reshaped, and then processed by a time distributed LSTM network to automatically "learn" important features, resulting in time series features learned (TSFL), thus reducing time series on the welding time level to TSFL on the welding operation level. Program numbers (ProgNo) are one-hot encoded to incorporate information of welding programs (strictly speaking, this is also a kind of feature engineering), in contrast to feature engineering's decomposition of time series with ProgNo. (The pros and cons will be discussed in the discussion section.) After that, RawSF are reshaped and concatenated with TSFL. They can directly be fed to a further LSTM network for modelling, or be flattened and fed into LR.

In the *Full* pipelines further FL is performed on time series and single features, represented by the pipeline shown in Figure 4.17b with the optional boxes. The only difference from *Base* is that an extra time distributed MLP is added after the concatenation layer, resulting in learned single features

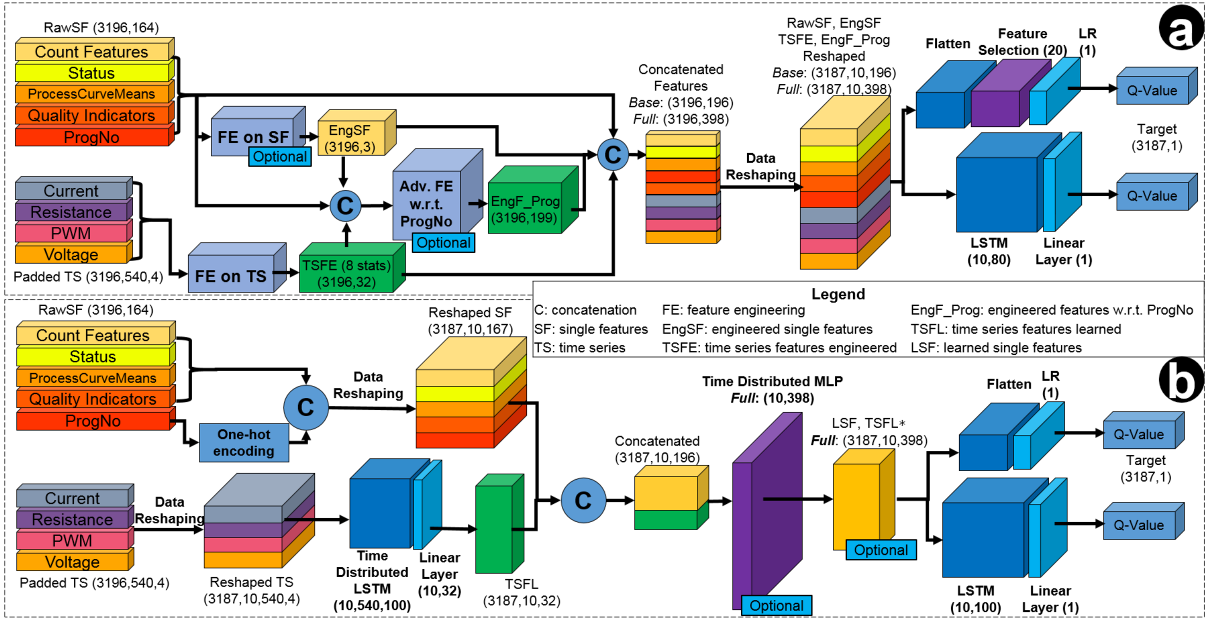


Figure 4.17: ML pipeline of feature engineering (a) [170] and ML pipeline of feature learning (b). Boxes with “Optional” indicate optional procedures tested in Experiment *Full*.

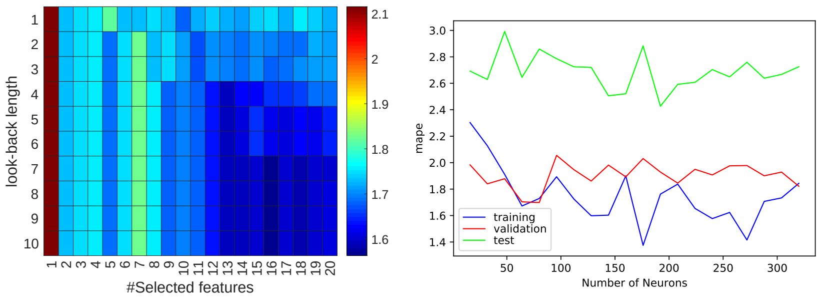
(LSF) and another time series features learned (TSFL*). The optional time distributed MLP provides the possibility to "learn" feature from RawSF.

The feature learning part of neural networks can be seen as a *Encoder*, transforming information of the input features into some abstract representations. While the modelling part can be seen as a *Decoder*, which takes the *Readout* from the *Encoder* and performs the information aggregation to output features.

Feature reduction and model alignment for comparison. Feature engineering generates a large number of features. From each of the 4 actual process curves, 8 features are extracted ($4 \times 8 = 32$). Adding the RawSF, the number of features amounts to $164 + 32 = 196$ before reshaping for *Base* (Figure 4.17a). For *Full*, EngSF adds 3 more features, and EngF_Prog doubles the feature number. The number of features amount to $((164 + 32 + 3) \times 2 = 398)$ before reshaping.

For FE-LR, suppose the look-back length l is 10, for *Base* a total of 1960 (196×10) features are generated after flattening, while for *Full* 3980 features. The large amount of features need to be reduced before modelling. This study adopts a wrapper method with *step-wise forward feature selection* (Section 2.4.3) to reduce the feature number for LR modelling, since other methods (e.g. F-Test, recursive feature elimination) yield worse performance in experiments. For FE-LSTM, the feature reduction is already performed by neural networks.

Feature learning handles the issue of excessive features by adjusting the number of neurons. To make these approaches more comparable, the number of neurons in some layers should be aligned with the feature engineering approaches. The number of *ReadOut* features from time distributed LSTM network is therefore set (Figure 4.17b) to 8 for each time series, amounting to 32 in total (corresponding to the number of TSFE). To not constrain the flexibility of network architecture, the *ReadOut* is realised by adding a linear layer after the time distributed LSTM network. Similarly, the number



(a) (b)

Figure 4.18: (a) Hyper-parameter tuning by evaluating mape on validation set of WM1 for LR. Blue: low mape. Yellow: middle mape. Red: high mape.
(b) Hyper-parameter tuning by evaluating mape on validation set of WM1 for LSTM. Performance on three all sets are illustrated for monitoring [170].

of neurons of the time distributed MLP in *Full* also aligns with the feature number after FE in *Full*, amounting to 398.

Performance metric and data normalisation. This study uses the performance metric *mape* (Section 2.4.8), as it is scale-independent and intuitive to the domain experts.

Z-score normalisation (Section 2.4.4) is applied to the input features before modelling. The parameters for normalisation, including mean values and standard deviations, are learned from the training and validation set and applied to all input data.

No normalisation is applied to the output feature, the Q-Value, because the scale of Q-Value is fixed to a level (the optimal value is one).

ML model training and hyper-parameter selection. Eight ML models are trained on the training set and the hyper-parameters are selected based on the performance evaluated on the validation set.

(1) For the pipeline FE-LR, the model is the simplest: linear regression. Two hyper-parameters are to be selected for LR, the *#selected features (ω)* fed into LR and *look-back length (l)* for data reshaping (Table 4.10). These two hyper-parameters are similar in the sense, that as they increase, the data

Table 4.9: Hyper-parameters that are fixed once chosen

Hyper-parameter	Value	Reason
Look-back length	10	Kept the same as FE-LR
Optimiser	Adeldelta	Fast convergence
Data shuffling	True	Better performance
Memory	Stateless	Better performance
Early-stopping	Save the best model	Better performance

amount provided to the ML model will increase and therefore the model performance evaluated on the training set should always increase.

These two features are selected with LR and then fixed for other FE pipelines, for several reasons. 1) To make a fair comparison, the data amount delivered to the model should remain the same, so that the performance difference is indeed caused by the quality of features or models, not the data amount. 2) In industrial application, it is desired to find suitable hyper-parameters that provide relatively good performance, avoid overfitting, and ideally make the model insensitive to the hyper-parameters.

A grid search is performed to find these two hyper-parameters. It can be clearly seen from Figure 4.18a that after around 12 of selected features and a look-back length of 5, the performance of models trained on Dataset 1 reaches a plateau (or basin in sense of mape). The models trained on Dataset 2 show similar characteristics. At the end, $\omega = 20$ and $l = 10$ are selected because they are in a relatively large and insensitive area of the two hyper-parameters.

(2) For FE-LSTM, there exist many hyper-parameters. Most of them need only to be tested once because these will normally deliver better performance than their alternatives. A series of experiments are conducted and these hyper-parameters are then fixed as shown in Table 4.9.

Other hyper-parameters include #layers of the Decoder LSTM and #neurons in each layer. They are tuned using limited grid search (varying one while fixing others), and the results (Figure 4.18b) show model performance is relatively insensitive to the hyper-parameters, as the difference between

Table 4.10: Hyper-parameter selection of ML methods

Methods	Hyper-parameters	Selection method
FE-LR	#Selected features , look-back length	Grid search
FE-LSTM	#Layers of Decoder LSTM, #neurons and Table 4.9	Limited grid search
FL-LR	#Layers of FL LSTM, #neurons and Table 4.9	
FL-LSTM	#Layers of FL and Decoder LSTM, #neurons and Table 4.9	

the worst performance and the best performance is about 0.3% mape (for validation data). After experimenting, models with two layers of LSTM have almost the same performance as models with one layer. Therefore, one layer of LSTM is selected according to the principle of Ockham's razor.

(3) For FL-LR, the *look-back length* is the same as FE pipelines for a fair comparison. Other hyper-parameters include #layers of feature learning (Encoder) LSTM and #neurons in each layer. They are selected with the best performance on the validation sets. Through limited grid search it turns out the models are also relatively insensitive to the number of neurons. The same as FE-LSTM, one layer of LSTM is good enough to be selected.

(4) For FL-LSTM, the *look-back length* and optimiser are the same as that for FL-LR. Other hyper-parameters include #layers of FL LSTM, #layers of Decoder LSTM, and #neurons in each layer. They are selected with the same practice as that for FL-LR.

4.3.4 Results and Discussion

After hyper-parameter selection, the ML models are trained again with the training and validation sets (trainingx sets), then tested on the test set. The two levels of complexity (*Base* and *Full*) and four pipelines result in eight best models. Two benchmarks are used as the baselines comparison for model performance, which is Benchmark 2 ($\hat{Q}_{next} = Q_{pre1}$) and Benchmark 3 ($\hat{Q}_{next} = Q_{pre1\_Prog}$). [1]

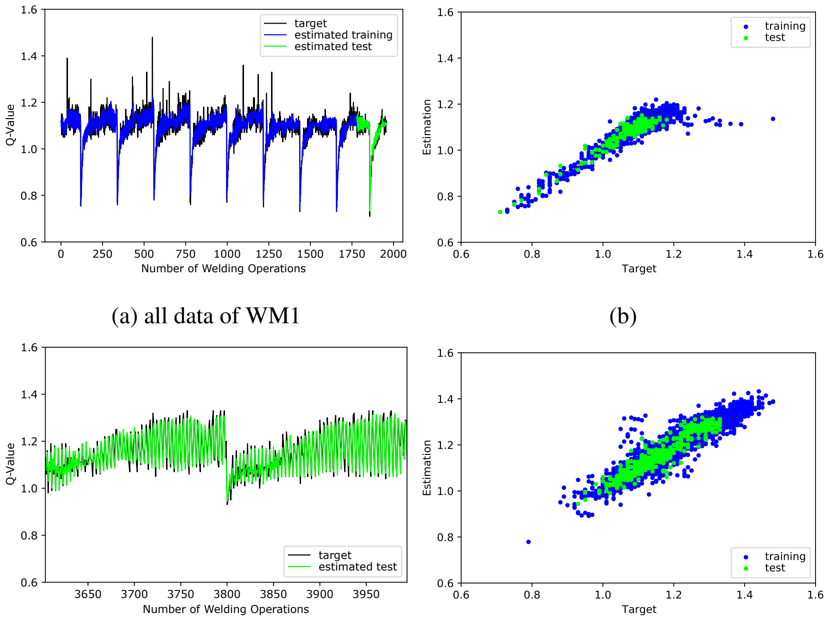
Comparison of prediction accuracy. The model performance and feature settings are listed in Table 4.11. The simple benchmarks are already good for Welding Machine 1, perhaps for its lower complexity with only 2 welding programs, while much worse for Welding Machine 2 with 4 welding programs. It can be seen that the results of the eight models are always better than the benchmarks. The best models in *Full* are always better than that in *Base*, as expected. This means high level of feature engineering and feature learning brings improvement. In general, the performance of the FE pipelines and FL pipelines are very good and comparable, indicating adequate feature engineering and effective feature learning.

Table 4.11: Model performance tested on test sets

	Data preprocessing and feature setting		Modelling	mape (WM1)	mape (WM2)
Benchmark2: $\hat{Q}_{next} = Q_{pre1}$				3.19%	7.74%
Benchmark3: $\hat{Q}_{next} = Q_{pre1\_Prog}$				2.51%	2.51%
Base	FE	RawSF, TSFE	LR	2.38%	2.50%
			LSTM	2.35%	2.27%
	FL	RawSF, TSFL	LR	2.14%	2.11%
			LSTM	2.72%	2.37%
Full	FE	RawSF, TSFE, EngSF, EngF_Prog	LR	1.61%	2.10%
			LSTM	2.04%	1.94%
	FL	LSF, TSFL*	LR	2.01%	1.94%
			LSTM	2.45%	2.30%

The best model for Welding Machine 1 is FE-LR (1.61%), while for Welding Machine 2 FE-LSTM and FL-LR are equally good (1.94%). FE-LR outperforms FE-LSTM for Welding Machine 1, and for Welding Machine 2 the other way around. By comparing FE-LR and FE-LSTM, the reason can be postulated that Welding Machine 1 has less complex data than Welding

[1] Benchmark 1 is $\hat{Q}_{next} = mean(Q_{pre1}, Q_{pre2}, ..., Q_{pre10})$, established for reference. It is always worse than Benchmark 2 and 3. Therefore, only Benchmark 2 and 3 are used for performance comparison.



(a) all data of WM1 (b)

(c) test set area of WM2 (d)

Figure 4.19: Results of the best FE model for WM1 in line plot (a) and scatter plot (b). Results of the best FL model for WM2 in line plot (c) (closer zoom into the test set area) and scatter plot (d) [170]

Machine 2. The most complicated model FL-LSTM is however not the best. The reason may be that the data are still too simple for FL-LSTM to demonstrate its power, as neural networks are known for its capacity of capturing very complicated data [198].

The best prediction results are illustrated in Figure 4.19 with line plots and scatter plots. It can be seen from Figure 4.19a that the model predictions for WM1 follow the target trend very closely. Figure 4.19b shows the estimation for targets with high values in training data are not good, but these targets are actually outliers in Figure 4.19a. This indicates that the model is insensitive to outliers. Figure 4.19c zooms into the test set area of WM2, and shows that the predictions match the target exceptionally well.

Table 4.12: More comprehensive evaluation. 1: on NVIDIA Titan V, 2: on AMD EPYC 7801.

	Domain understanding	Feature understanding	Data preparation	Hyper-parameter tuning	Mean training time	Model explainability
FL	8 meetings	2 meetings	less general	250 experiments	about 355s[1]	mediocre
FE	8 meetings	10 meetings	less general	200 experiments	about 15s[2]	good

The FE-LR can provide insights by interpreting what features are selected. The resulted models show the most important features are the Q_{pre1}_Prog, WearDiff, I_{mean}, R_{std}. These indicate (1) the previous spot with the same program number contain important information for predicting the quality of the next welding operation; (2) wearing effect has large impact for quality; (3) average value of current and standard deviation of resistance contain the most important information of quality.

Besides, from results of hyper-parameter tuning for FE-LR (Figure 4.18), it can be interpreted that the quality of next spot is truly dependent on previous 1 to 4 spots, as *mape* decreases until a look-back of 5. The number of features should be better more than 12, indicating the quality is influenced by multiple factors.

Comparison of adoptability metrics. The prediction accuracy of the FL and FE pipelines are comparable. Yet for adoption and deployment in industry, not only prediction accuracy is important, but also many other adoptability metrics. Table 4.12 summarises these metrics. To evaluate these metrics, an ideal way is to strictly design and conduct experiments of developing and deploying ML pipelines across numerous use cases and analyse the collected statistics. This is however not realistic in most cases, not only because that any companies would not have abundant ML projects to economically test and evaluate, but also for huge discrepancies between different projects to be compared. However, this does not prevent obtaining insights before numerous use cases are conducted.

Number of meetings between data scientists and domain experts (averaged to about one hour each) is used for evaluating effort of domain understanding and feature understanding, and the number of experiments for evaluating hyper-parameter tuning. Training time is an average value for each run of model training, including data pre-processing. Clearly, in these metrics some aspects are quite subjective, but in practice it gives a reasonable estimation of adoptability for ML pipelines in an industrial setting.

Domain understanding for data scientists takes the same time for FL and FE. After that, 2 more meetings about the detailed meaning of features were needed, and then FL was possible. While for FE, 10 meetings were spent on understanding the details of features.

For data preparation, both FL and FE needed to pull data from different datasets with discrepancies in naming and formats. Effort was needed to use different versions of scripts to extract data from SQL database, Excel table, text files, RUI-files (for time series), etc. Thus the effort was the same and scripts were rather less generalisable.

Hyper-parameter tuning for FL was arduous. For each LSTM and each dataset, 10 hyper-parameters with about 250 experiments needed to be determined. Considering the time for each model training, the hyper-parameter tuning for FL easily exceeded several hours. While for FE, the training time was much faster, and the tuning was quite handy even for grid search.

In terms of model explainability, there was not much to interpret from FL models, because their neurons were abstractions of features, lacking physical meanings. In contrast, the FE models indeed could provide some insights.

4.3.5 Conclusion and Outlook

Conclusion. This use case compares the performance of eight models from ML pipelines following two feature extraction philosophies, in terms of prediction accuracy, time and effort expense for domain and data understanding, hyper-parameter tuning and training time, and model explainability. The ML pipelines of feature engineering and feature learning are comparable in terms of prediction accuracy, but relatively different in terms of other performance metrics of adoptability (Table 4.12). These adoptability metrics are especially important for industrial deployment.

Outlook. In future research, the two ML pipelines can be evaluated more thoroughly on other datasets and other scenarios. Both two pipelines can be enhanced with semantic technologies to improve their adoptability. The feature engineering pipeline will be studied thoroughly more in direction of feature evaluation and selection in Use Case 4.4, to provide more insights for the welding domain from a data-driven perspective.

4.4 A Closer Study on Feature Engineering and Feature Evaluation

Overview: The same as in Use Case 4.3, this use case studies predicting (or forecasting) quality of resistance spot welding. The difference is that this study focuses on feature engineering, feature evaluation and interpretation. The goal is to gain insights from these features. The studied data are collected from WM1 (or WM_G) and WM3 (WM_R) in Use Case 4.2, since they are complete and representative datasets of low complexity and high complexity (Section 4.2.5). The studied ML methods are feature engineering with three classic methods: linear regression, multi-layer perceptron (with one hidden layer), and support vector machine. An overview of the corresponding application questions, open questions and methods is listed in Table 4.13.

Table 4.13: Overview of application questions (AQ), open questions (OQ) and methods (M)

AQ	OQ	Methods
AQ3, AQ5	OQ4, OQ5, OQ6	M2.1.1, M2.1.2, M2.2.4, M2.4.1, M2.4.2, M2.4.5, M2.4.7

4.4.1 Question Definition

The same question in Use Case 4.3 is studied, to predict Q-Value of the next welding operation. The special emphasis is to experiment on four feature engineering strategies (two of them are already introduced in Use Case 4.3) to analyse the influence of features on model performance. This is also an attempt to integrate domain knowledge in machine learning modelling, combining views from data science and manufacturing domain.

4.4.2 Data Description

An overview of the datasets collected from WM1 and WM3 are given in Table 4.14. The data contain 164 *Single Features* and 8 *Time Series* (Section

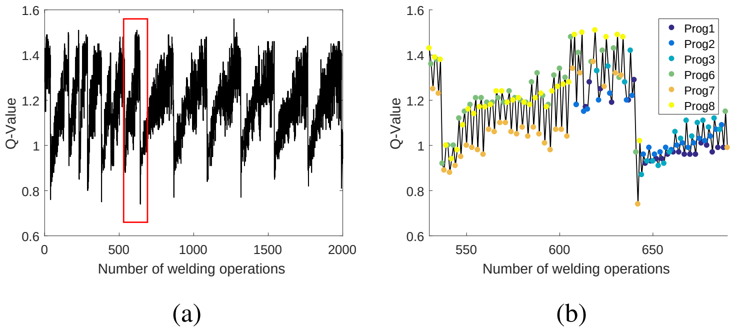


Figure 4.20: (a) Q-Value along number of welding operations for WM3 (partially shown). The irregular trends of Q-Value indicate complex production conditions. The red rectangle indicates the area for a closer look in (b).
(b) Welding operations performed with different welding programs. Note that there exists a change of production arrangement. Before the 618th welding operation, only three welding programs were performed (Prog6, 7 and 8). After that three MORE welding programs were performed (Prog1, 2 and 3) due to a change of production plan. Possibly a new chassis part is welded. The change is a typical situation in manufacturing.

2.3.2). The reference process curves are not excluded this time since the data of WM3 are collected from a period with changed production schedule (Figure 4.20b). The behaviour of quality indicator Q-Value is illustrated in Figure 2.2 for WM1 and Figure 4.20 for WM3. This strong periodicity of Q-Values indicate that the example datasets very likely have temporal dependencies described in Section 2.1.2.

Table 4.14: Two example datasets for prediction of the next Q-Value

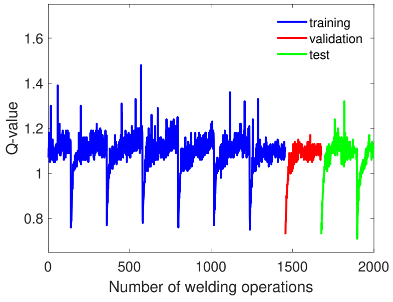
Dataset	Welding machine	#Prog	$\#DT$	$\#DT_{tr}$	$\#DT_{val}$	$\#DT_{tst}$
Dataset 1	WM1	2	1998	1456	219	323
Dataset 3	WM3	6	5839	4474	602	763

4.4.3 Experiment Settings

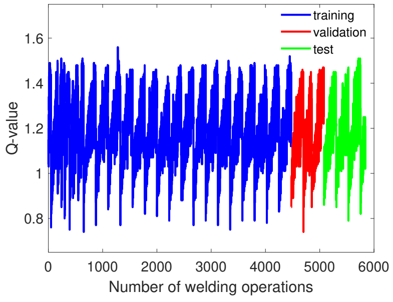
The data analysis in this use case follows the branch of feature engineering and classic machine learning modelling introduced in Section 2.4. The implementation in the use case is illustrated in the upper part of Figure 4.16.

After data preparation, hierarchical feature extraction will be performed on the two time levels using feature engineering. These engineered features then will be reshaped to accentuate short-time dependency. The reshaping look-back length adopts the same one (10) as in Use Case 4.3. Twenty features will be selected and fed into three classic machine learning methods. After hyper-parameter selection based on the model performance on the validation data, the best model of each method will be retrained using all training data and validation data (trainingx set). In the end, the models will be compared and the results will be interpreted with deep involvement of engineering know-how.

Data splitting according to temporal structure. In this use case, the data splitting points are chosen at complete units of dress cycle levels. The datasets are first split into to training, validation, and test in a 0.8 : 0.1 : 0.1 ratio, and then rounded to complete electrode dress cycles. It is also important to note that the validation data and test data should contain at least one complete dress cycle to ensure that they cover the wearing effect through a full dress cycle. Details of data tuple numbers are illustrated in Figure 4.21 and Table 4.14.



(a) WM1



(b) WM3

Figure 4.21: Example of data splitting rounded to complete dress cycles. Note that the data of WM3 is much more complicated than that of WM1. Especially at the beginning cycles, complicated dressing operations were performed. This is caused by the change of production arrangement (Figure 4.20)

Four settings of feature engineering. Feature engineering is performed on both time series features and single features. Four settings of feature engineering are designed. From Setting 0 to Setting 3, the degree of feature engineering deepens, to study whether and to which degree feature engineering can increase model prediction power. The feature engineering follows the strategies introduced in Section 2.4.2.

- **Setting 0**, no feature engineering. Only the *raw single features* will be used in machine learning modelling. Note that the *ProcessCurveMeans* in the *raw single features* already provide some information of the time series. A total of 1640 (164×10) features are generated before feature selection.
- **Setting 1** (almost equal to *Base* of FE pipeline in Section 4.3), only performing feature engineering on time series. Additional four reference curves are added in analysis. Ten features are extracted from each time series (explained in the next paragraph). A total of 2440 ($(164+8\times 10)\times 10$) features are generated before feature selection.
- **Setting 2**, performing feature engineering on time series and single features. The time series features engineered (TSFE), raw single features (RawSF) and engineered single features (EngSF) will be concatenated for machine learning modelling. A total of 2470 ($(164+3+8\times 10)\times 10$) features are generated before feature selection.
- **Setting 3** (almost equal to *Full* of FE pipeline in Section 4.3), performing a further step of feature engineering on time series and single features. A total of 4940 ($(164+3+8\times 10)\times 2\times 10$) features are generated before feature selection.

Feature engineering on time series. Different from the strategies in Use Case 4.3, this use case deepens the degree of domain knowledge integration. According to domain knowledge, the time series contain an initial stage and a welding stage, pre-determined by the adaptive control. After referring to various literature, ten *TSFE* are designed, extracted from the welding stage

(Figure 4.22): length (WeldTime) [139], maximum (WeldMax) [38], minimum (WeldMin) [135], maximum position (WeldMxPo) [125], minimum position (WeldMnPo), slope (WeldSlope, a local extrema between the WeldMxPo and WeldMnPo is very unlikely according to domain knowledge) [126] ($slope = (max - min)/(mxpo - mnpo)$), mean (WeldMean) [135], median (WeldMedian), standard deviation (WeldStd) [135], and end value (WeldEndValue).

Figure 4.22 takes the resistance curve as example, which is the most complicated process curve, and often deemed as the most informative time series feature by the literature [38, 135, 36, 125]. It can be seen from the figure that these features can characterise the statistic and geometric properties of resistance time series to a large degree. Current, voltage and pulse width modulation curves are much simpler than the resistance curve and can also be described by these features.

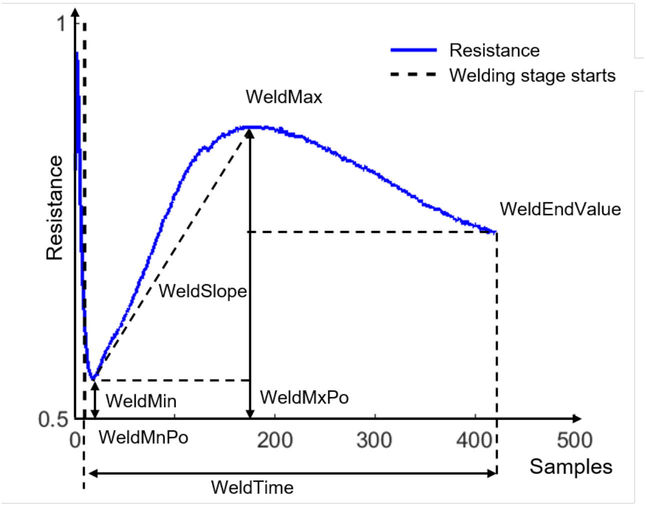


Figure 4.22: Domain knowledge supported time series feature extraction: from the welding stage 10 statistic and geometric features are extracted. Most features are shown except for: WeldMean, WeldMedian, WeldStd [56, 57, 58, 125, 135, 141].

Feature selection. The number of features grow enormously because of feature engineering. After feature extraction from time series, 80 TSFE are generated. After reshaping, 2470 features are generated for Setting 2 and 4940 for Setting 3. These features need to be reduced before modelling. Feature selection is optimal as it preserves the features for interpretation (the other option is feature aggregation, which loses the meanings of the features). The multivariate wrapper method, *step-wise forward feature selection* [50] is applied with LR. Experiments have confirmed that features selected by MLP and SVR are almost the same as that by LR, because the number of selected features is set to 20, which allows the selected features cover a larger range of relevant information than what is normally needed. Therefore, the same features selected by LR are also used for modelling with MLP and SVR, to reduce the computational time.

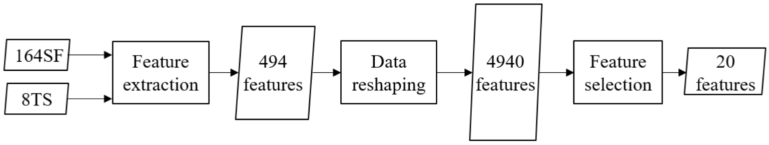


Figure 4.23: Reducing features by feature selection. SF: single features. TS: time series.

Performance metric. This study adopts the performance metric *mape* (Section 2.4.8), as it is scale-independent and intuitive to the domain experts.

Normalisation. Z-score normalisation (Section 2.4.4) is applied to the input features before feeding them into modelling. The parameters for normalisation, including mean values and standard deviations, will be learned from the training and validation set and applied to all input data. No normalisation is applied to the output feature, the Q-Value, for the scale of Q-Value is fixed to a level (the setpoint is 1).

ML model training. Three representative classical ML methods, linear regression (LR), multi-layer perceptron with one hidden layer (MLP), and support vector regression (SVR) are studied. They are trained on the training

Table 4.15: Hyper-parameter selection of ML methods

Methods	Hyper-parameters	Selection method
LR	#Selected features, features, look-back length	Limited grid search
MLP	#Selected features, features, look-back length	Kept the same as LR
	#Neurons, activation function	Limited grid search
SVR	#Selected features, features, look-back length	Kept the same as LR
	Kernel type, regularisation factor, degree	Limited grid search

set, and their hyper-parameters are selected based on the performance on the validation set.

Hyper-parameter selection for LR. Two hyper-parameters are to be selected for linear regression, the *number of selected features* (ω) and *look-back length* (l). Apart from the study in Use Case 4.3, additional limited grid search is done for the four settings. The results in Figure 4.24 illustrate the model performance evaluated on validation set of WM1 and WM3. The models become insensitive to the hyper-parameters after more than approximately 20 features are selected and when the length of look-back window is longer than about 10. To note in Setting 3, where the EngF_Prog is used, the effective look-back time step is *look-back length* $\times$ *number of welding program* ($l \times \#Prog$). The performance of Setting 0 and Setting 1 for WM1 is more sensitive to hyper-parameters.

Hyper-parameter selection for MLP and SVR. The same feature sets determined by LR are used for MLP and SVR. Other hyper-parameters are selected using limited grid search. MLP has two hyper-parameters, #neurons in the hidden layer and the activation function. SVR has three hyper-parameters, kernel type, regularisation factor, and degree in case of polynomial kernel types (Table 4.15).

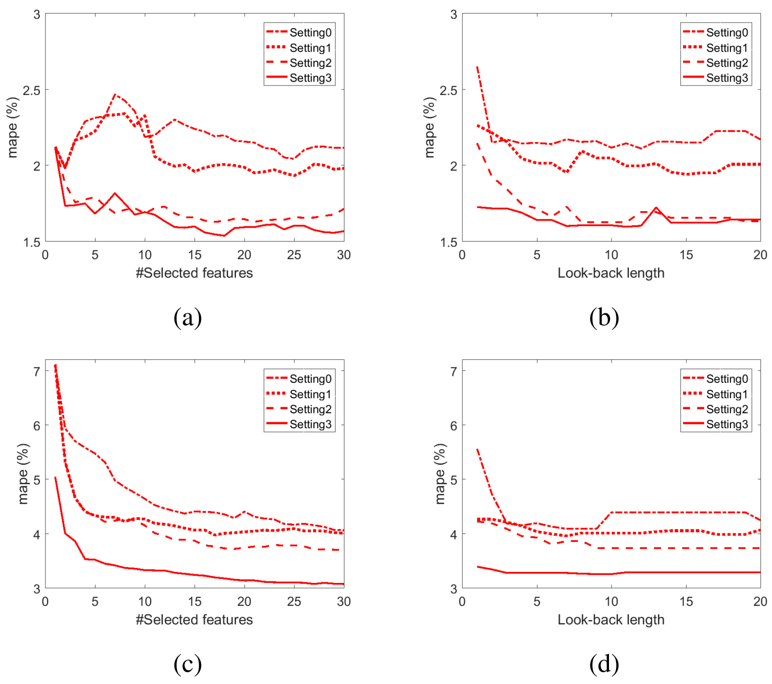


Figure 4.24: Hyper-parameter selection with limited grid search based on performance of validation set of WM1 for #selected features ω (a) and look-back length l (b). The performance reaches a basin in the area where $\omega \approx 20$ and $l \approx 10$ features are selected. The same for WM3 in (c) and (d).

4.4.4 Results and Discussion

The models are trained again with the selected hyper-parameters on the combined set of training data and validation data (trainingx set), and tested on the test sets. The model performance is compared and percentage improvements are calculated with respect to Setting 0 and the best benchmark (Benchmark 3) for WM1 (Table 4.16) and WM3 (Table 4.17). The prediction results on WM1 are illustrated in Figure 4.25.

Interpretation of prediction results. Based on the Table 4.16 and Table 4.17, it is easy to conclude that the model performance increases as the degree of feature engineering deepens. The performance of Setting 0 and

Table 4.16: LR performance on test set of WM1. Imprv. w.r.t.: improvement with respect to.

Feature settings or benchmarks		mape (LR)	Imprv. w.r.t. Setting 0	Imprv. w.r.t. Benchmark3
Benchmark3	$\hat{Q}_{next1} = Q_{pre1\_Prog}$	2.69%	-0.75%	-
Setting 0	RawSF	2.67%	-	0.74%
Setting 1	RawSF, TSFE	2.55%	4.49%	5.20%
Setting 2	RawSF, TSFE, EngSF	2.10%	21.35%	21.93%
Setting 3	RawSF, TSFE, EngSF, EngF_Prog	1.92%	28.09%	28.62%

Table 4.17: LR performance on test set of WM3. Imprv. w.r.t.: improvement with respect to.

Feature settings or benchmarks		mape (LR)	Imprv. w.r.t. Setting 0	Imprv. w.r.t. Benchmark3
Benchmark 3	$\hat{Q}_{next1} = Q_{pre1\_Prog}$	5.17%	-7.48%	-
Setting 0	RawSF	4.81%	-	6.69%
Setting 1	RawSF, TSFE	4.55%	5.41%	11.99%
Setting 2	RawSF, TSFE, EngSF	4.19%	12.89%	18.96%
Setting 3	RawSF, TSFE, EngSF, EngF_Prog	3.40%	29.31%	34.24%

Setting 1 on WM1 shows slight improvement compared to Benchmark 3. The performance of Setting 0 and Setting 1 on dataset of WM3 shows more improvement with respect to Benchmark 3, because the Q-Value behaviour of WM3 (Figure 2.2d) is much more complicated.

The performance improvement of Setting 1 compared to Setting 0 is insignificant, but consistent (Figure 4.24), which implies the features in the RawSF that contain time series information (the ProcessCurveMeans) already provide valuable information. A significant improvement begins with Setting 2, which indicates the feature engineering strategies on the welding operation level are meaningful. The performance difference between Setting 2 and Setting 3 analysed on dataset of WM1 is rather small, but that on dataset of WM3 is evident.

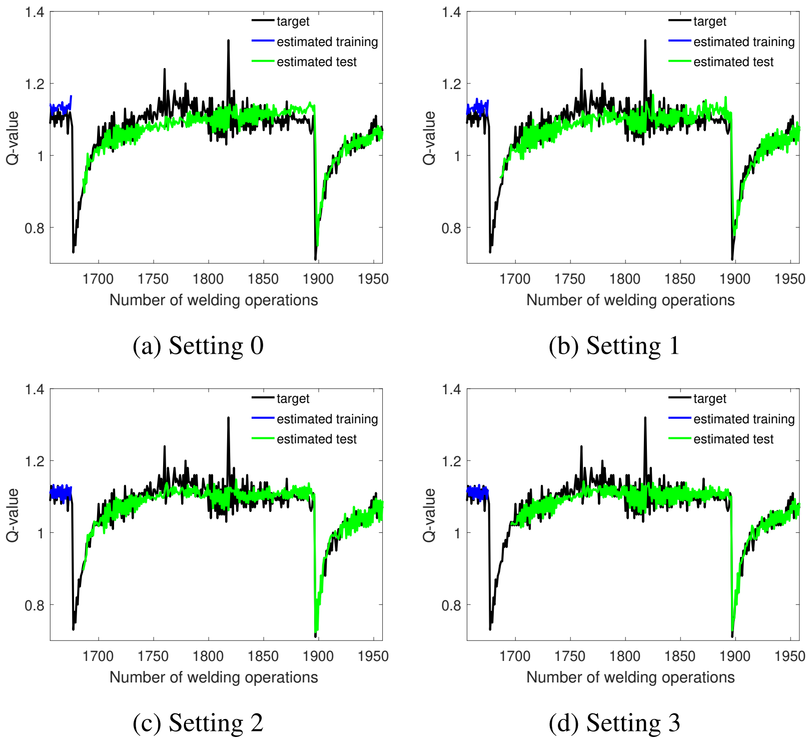
A further inspection on the two datasets reveals that the welding programs of WM1 are always arranged in a fixed interlaced order (Figure 2.2b), i.e. the ProgNo always repeat the same pattern: Prog1, Prog2, Prog1, Prog2, ..., but the welding programs of WM3 are not arranged in a fixed order. A change of production arrangement happened in the 618th welding operation (Figure 2.2d), before which the production arrangement was comprised of three welding programs. After the 618th welding operation, three extra welding programs were added to the production arrangement, which is typical in manufacturing industry since the production arrangement could change at any time in agile manufacturing. This explains why Setting 3, in which the welding program information is specially handled, has a significant improvement on dataset of WM3, but an insignificant improvement on dataset of WM1.

Interpretation of prediction visualisations. Figure 4.25 illustrates the visualisation of the target values and estimated values of WM1. It can be seen that the Setting 0 and Setting 1 learned a general rising trend of the behaviour of Q-Values. However, the dynamics of behaviour of Q-Values are more complex than a simple rising trend. The Q-Value trend first rises then declines slightly, and remains stable at the end. These dynamics are better learned by more complicated feature settings.

Besides, although most of the Q-Values are predicted with small errors, there exist quite a few outliers that have an apparent different behaviour than the "normal" Q-Values. These outliers seem to be random, and cannot be explained with the trained models.

The complex dynamics of Q-Value behaviours implicate that the welding quality is not solely influenced by the assumedly linear wearing effect. According to domain knowledge, other influential factors include the statistic and stochastic variance caused by dressing, the chassis to be welded, etc.

Interpretation of hyper-parameter tuning results. Reviewing the hyper-parameter results (Figure 4.24b), the selection of look-back length imply



(a) Setting 0

(b) Setting 1

(c) Setting 2

(d) Setting 3

Figure 4.25: Prediction results zoomed into test set area of LR models on WM1.

that the hypothesis holds, that there exist temporal dependencies between welding spots, since the performance of the models improves as the look-back length increases. That is to say, the Q-Value of the next welding spot is indeed dependent on the previous welding operations. Figure 4.24 also reveals that the improvements along degree of feature engineering is not random but systematic, although some times small.

Feature interpretation of WM1 with engineering knowledge. Table 4.18 lists the 5 most important features with a order of descending importance for Setting 0, Setting 1, Setting 2 and Setting 3 on dataset of WM1, respectively.

Table 4.18: The most important 5 features selected from feature settings in the analysis of dataset of WM1, listed in ranking of descending importance. Note that the score is evaluated on a multivariate basis, i.e. the importance of ω-th feature is the combined score of the 1st to ω-th feature. The correlation coefficient between the model estimation and target value is chosen as the feature score for an intuitive comparison.
The prefixes RawSF, TSFE, EngSF, EngF_Prog indicate the feature source, the suffixes indicate the time stamp of the features, and the stems indicate the physical meanings, e.g. Q for Q-Value, R for resistance, I for current, I2_Mean for ProcessCurveMeans of current, I_WeldMin for the minimum extracted from the *welding stage*.

#	Setting 0	Score	Setting 1	Score
1	RawSF_Q_pre2	0.72	RawSF_Q_pre2	0.72
2	RawSF_I2_Mean_pre1	0.78	RawSF_I2_Mean_pre1	0.78
3	RawSF_WearCount_next1	0.79	RawSF_WearCount_next1	0.79
4	RawSF_Q_pre9	0.81	TSFE_I_WeldMin_pre3	0.81
5	RawSF_Q_pre3	0.82	TSFE_R_WeldStd_pre8	0.82
#	Setting 2	Score	Setting 3	Score
1	RawSF_Q_pre2	0.72	EngF_Prog_Q_pre1	0.72
2	EngSF_NewDress_next1	0.80	EngF_Prog_WearDiff_next1	0.86
3	RawSF_I2_Mean_pre1	0.84	EngF_Prog_I2_Mean_pre1	0.87
4	TSFE_R_WeldStd_pre1	0.86	TSFE_R_WeldStd_pre1	0.88
5	RawSF_I_Mean_pre3	0.87	EngF_Prog_Q_pre2	0.88

The Q-Value of the previous second welding spot (RawSF_Q_pre2) is selected as the most important feature in three settings. Since the ProgNo always repeat the same pattern: Prog1, Prog2, Prog1, Prog2, ..., the feature RawSF_Q_pre2 is therefore equal to EngF_Prog_Q_pre1. EngF_Prog-_Q_pre1 is namely the Q-Value of the previous spot welded with the same welding program as the next welding spot (this feature is identical to Benchmark 3). This means the quality of welding usually does not have abrupt changes.

Moreover, the features RawSF_WearCount_next1, EngSF_NewDress _next1, EngF_Prog_WearDiff_next1 are selected, which means the wearing effect has strong influence on the welding quality, and therefore quality prediction should always consider features characterising the wearing effect.

The features RawSF_I2_Mean_pre1, TSFE_R_WeldStd_pre1, TSFE_I_-WeldMin_pre3, RawSF_I_ Mean_pre3 are selected, which means the time series features extracted from the welding stage indeed contain some information for predicting the next spot quality. Note that these features are of previous first or third spots, which are not welded with the same ProgNo as the next spot. This indicates that the information provided by historical spots that are not welded with the same ProgNo as the next spot are also important. The reason may be that the wearing effect during these temporally adjacent welding operations has influence on the next welding quality.

In Setting 0 and 1, the selected features of relatively early operations, like RawSF_Q_pre9, TSFE_R_WeldStd_pre8, are questionable, because their influence should not be greater than temporally more adjacent features. Selection of these questionable features is avoided in Setting 2 and Setting 3, which feature engineering is more complex.

Feature interpretation of WM3 with engineering knowledge. As for WM3 (Table 4.19), the selected features are different. The most obvious difference is that RawSF_Q_pre2 is no longer selected as the most important feature. As mentioned in Section 4.4.4, a change of arrangement of welding programs happened in the 618th welding operation. For the same reason, no EngSF is selected in the most important features, since EngSFs do not incorporate information of welding programs. Although the feature RawSF_WearCount_next1 can also describe the dependency of Q-Value on wearing effect to some degree, as it is selected through Setting 0 to 2, the performance of models trained on Setting 3 demonstrates a significant improvement (Table 4.17). This highlights the advantage of the feature EngF_Prog_Q_pre1 in Setting 3.

Many TSFEs extracted from reference curves are selected as more important than those from actual process curves. Reference curves are prescribed

Table 4.19: The most important 5 features selected from feature settings in the analysis of dataset of WM3, listed in ranking of descending importance. Feature scores are similar to Table 4.18.
The prefixes RawSF, TSFE, EngSF, EngF_Prog indicate the feature source, the suffixes indicate the time stamp of the features, and the stems indicate the physical meanings, e.g. Q for Q-Value, I2_Mean for ProcessCurveMeans of current, R for resistance, RefU_WeldEndValue for the end value extracted from the *welding stage* of the reference curve of voltage, I_WeldMin for the minimum extracted from the *welding stage*, RefPWM for the reference curve of the *Pulse Width Modulation*

#	Setting 0	Score	Setting 1	Score
1	RawSF_WearCount_next1	0.75	RawSF_WearCount_next1	0.75
2	RawSF_Time_pre2	0.81	TSFE_RefU_WeldEndValue_pre2	0.85
3	RawSF_I2_Mean_pre2	0.85	TSFE_RefR_WeldMax_pre1	0.88
4	RawSF_R_Mean_pre1	0.86	TSFE_I_WeldMin_pre3	0.90
5	RawSF_Power_Mean_pre10	0.87	TSFE_R_WeldMax_pre1	0.90
#	Setting 2	Score	Setting 3	Score
1	RawSF_WearCount_next1	0.75	EngF_Prog_Q_pre1	0.79
2	TSFE_RefU_WeldEndValue_pre2	0.85	EngF_Prog_WearDiff_next1	0.91
3	TSFE_RefR_WeldMax_pre1	0.88	TSFE_RefR_WeldSlope_pre2	0.92
4	TSFE_I_WeldMin_pre3	0.90	RawSF_WearCount_next1	0.93
5	TSFE_R_WeldMax_pre1	0.90	TSFE_RefPWM_WeldMean_pre4	0.93

by the welding programs, and are therefore always identical for a specific program. This implicates that the next spot quality is also dependent on the welding programs performed on the previous spots, rather than the corresponding actual process curves. This phenomenon is not evident for WM1, since its welding program arrangement is fixed. Similar to the case of WM1, complex feature engineering avoids selection of questionable features like RawSF_Power_Mean_pre10.

Interpretation of results of MLP and SVR models. Table 4.20 and 4.21 list the results of MLP and SVR models trained with the features determined by LR models and their hyper-parameters selected using limited grid search.

Table 4.20: MLP and SVR models evaluated on test set of WM1. “rbf": Radial-Basis-Function.

Feature settings	mape (MLP)	Activation function	#Neurons	mape (SVR)	Kernel type	Regularisation factor
Setting 0	2.27%	tanh	5	2.25%	rbf	8.17
Setting 1	2.42%	tanh	25	1.94%	rbf	10.19
Setting 2	2.03%	tanh	50	2.08%	rbf	6.15
Setting 3	1.92%	tanh	53	1.87%	rbf	8.17

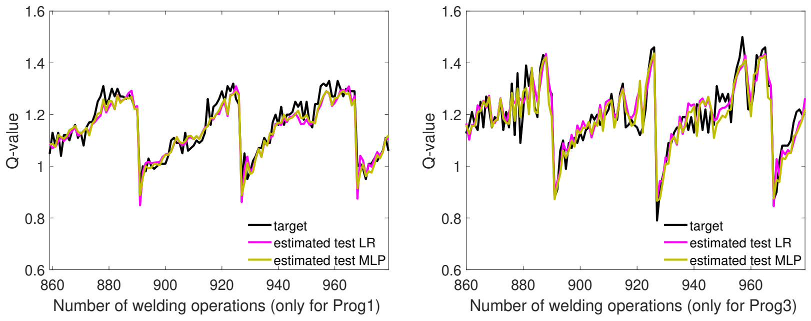
Table 4.20 and 4.21 demonstrate that performance can indeed be improved by non-linear models. This indicates that there exist non-linearity and interaction between the selected features, which cannot be described using the LR models.

The performance of MLP models are usually better than LR models. The performance gain becomes significant for complicated datasets (WM3) with the highest degree of feature engineering (Setting 3). Conspicuous is that the Setting 1 model performance is worse than that of Setting 0 for both welding machines (Note that the number of selected features are the same for all settings. Only the available features before feature selection are different.). This indicates the time series features engineered may cause overfitting in some cases.

A closer look at the results of LR and MLP models on the test set of WM3 for two example welding programs with Setting 3 is illustrated in Figure 4.26. It reveals that although the different welding programs have very different behaviour of Q-values, both LR and MLP models are able to

Table 4.21: MLP and SVR models evaluated on test set of WM3. “rbf": Radial-Basis-Function.

Feature settings	mape (MLP)	Activation function	#Neurons	mape (SVR)	Kernel type	Regularisation factor
Setting 0	3.87%	tanh	50	4.02%	rbf	19.02
Setting 1	4.07%	tanh	47	4.49%	rbf	21.29
Setting 2	3.57%	logistic	60	4.32%	rbf	14.79
Setting 3	2.97%	tanh	44	3.34%	rbf	16.00



(a) Setting 3 Prog 1 (b) Setting 3 Prog 3

Figure 4.26: Examples of prediction results on test set of WM3 illustrated for two welding programs, with the model of MLP with the Setting 3 of feature engineering, 20 features selected, and a look-back length of 10.

capture the different dynamics. From the figure the performance of the two models are not easy to differentiate. This indicates the importance of the numerical performance metric in Table 4.17 and 4.21.

The performance of SVR models are on par with the LR models. Also conspicuous is that the performance of SVR models does not always improve as the degree of feature engineering increases. A further investigation of SVR models reveals that the performance of them fluctuates irregularly quite a few (up to 1.2% mape) as the regularisation factor changes, which means SVR models often fall into local optimums. Some SVR models perform well on training and validation sets but perform badly on test sets, which indicates a tendency of overfitting.

4.4.5 Conclusion and Outlook

Conclusion. This use case studies the ML pipelines of feature engineering with more domain knowledge integration than that in Use Case 4.3, the degree of which is intensified through four feature settings. In the area of the selected two hyper-parameters, the machine learning method is insensitive to the hyper-parameters, *number of features* and *look-back length*, which is desired for industrial application. Furthermore, the results and selected features are extensively interpreted to provide more insights to the domain perspective. A blind training of ML models would select questionable and less robust features. Through domain knowledge-supported feature engineering, visualisation of prediction results and interpretation of selected features, the explainable ML pipelines of feature engineering can avoid this, enabling a relatively transparent understanding of machine learning modelling.

Outlook. In future work, the following directions are valuable from both industrial and academic points of view:

- Testing the proposed approach on more datasets to verify the generalisability.
- Exploring other feature extraction strategies on complex welding machines.
- Investigating other machine learning methods, e.g. other types of artificial neural networks, especially recurrent neural networks, which are suitable for processing data with temporal structures.
- Predicting the Q-Value of the next welding spot as a probabilistic distribution, by using e.g. quantile regression [199]. Before the next welding actually happens, the next Q-Value can in fact not be deterministically predicted. The prediction of the next Q-Value in this work is actually a prediction of the mean value. A better way is probabilistic forecasting.
- Using the prediction results as a basis for process optimisation. After the next spot quality is predicted, there exist several possible measures to undertake, e.g. flexible dressing, adaptation of the reference curves, or switching to non-adaptive control mode.

4.5 Transferability of Models between Datasets

Overview: Once ML models are developed for some datasets, it is desired and necessary to transfer these models to other datasets. This section studies whether the ML model trained on one dataset can be transferred to datasets collected from other welding machines. The feature settings and models are the same as in Use Case 4.4. The target output of the ML models is the same: Q-Value of future welding operations. The studied data are collected from WM1 (or WM_G), WM2 (WM_I), and WM3 (WM_R) in Use Case 4.2. An overview of the corresponding application questions, open questions and methods is listed in Table 4.22.

Table 4.22: Overview of application questions (AQ), open questions (OQ) and methods (M)

AQ	OQ	Methods
AQ5, AQ10	OQ4, OQ8	M2.1.1, M2.4.5

4.5.1 Question Definition

This use case studies the transferability between models trained on different datasets, e.g. whether it is possible to train ML models on the dataset of one welding machine and apply them on another welding machine (Figure 4.27). The transferability is very desired in industrial scenarios, because there exist many advantages if a model trained on the dataset of one machine can be transferred to another new machine, without change or with small adaptation:

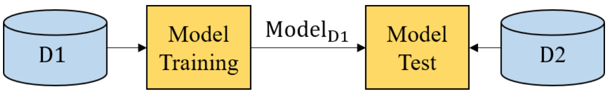


Figure 4.27: The simple workflow of model transfer. D1: Dataset 1. D2: Dataset 2.

- The cost of extra data collection on the new welding machine can be obviated or at least reduced;

- The cost of new model development, or old model adaptation, for the new welding machine can be saved;
- The time, in which the new machine performs manufacturing tasks but its product quality is not fully covered by data-driven monitoring, can be minimised.

ML models of Setting 3 introduced in Use Case 4.4 provide good prediction accuracy and are explainable. Their transferability will be investigated.

4.5.2 Data Description

Apart from the two datasets described in Section 4.4, the dataset collected from WM2 is also studied. The three datasets are summarised again for readers' convenience in Table 4.23. Their Q-Values are illustrated in Figure 2.2 for WM1, Figure 4.15 for WM2, and Figure 4.20 for WM3.

Table 4.23: Example datasets for studying model transferability

Dataset	Welding machine	#Prog	$\#DT$	$\#DT_{tr}$	$\#DT_{val}$	$\#DT_{tst}$
Dataset 1	WM1	2	1998	1456	219	323
Dataset 2	WM2	4	3996	3127	448	421
Dataset 3	WM3	6	5839	4474	602	763

4.5.3 Experiment Settings

This section first introduces a preliminary experiment and the problem revealed by this experiment. Then it explains the solution to the problem and the experiment settings to implement the solution.

Data splitting according to temporal structure. The data splitting strategy is exactly the same as in Use Case 4.4. Since after the hyper-parameter selection, the ML models need to be trained again using training and validation sets, the training and validation dataset can be deemed as the "training dataset" in a more general sense, and will be referred to as the "trainingx set" (D_{trx}).

Preliminary experiment: Training on WM1 and testing on WM3. A first step of the experiment is to simply train a model on one dataset and then directly apply on another dataset without any adjustment. To make the results in this use case comparable to that in Use Case 4.4, it is needed to ensure the training set is exactly as in Use Case 4.4. Therefore the trainingx set of WM1 ($D1_{trx}$) is used to train a MLP model and the model is tested on WM3 ($D3_{trx}$) dataset. Again to make the performance comparable to that in Use Case 4.4, the performance of the model should be evaluated on the trainingx set ($D3_{trx}$) and test set of WM3 ($D3_{tst}$), respectively. The input features are normalised using Z-score normalisation while the output is not. The normalisation parameters (denoted as P^X_{D1trx}) are therefore learned from the $D1_{trx}$ for input features and applied on $D1_{tst}$, $D3_{trx}$, and $D3_{tst}$. The resulting workflow becomes Figure 4.28. The models and their hyper-parameters and selected features are directly taken from Setting 3 in Use Case 4.4.

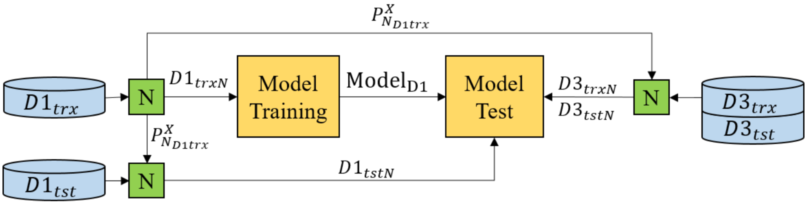


Figure 4.28: The workflow of model transfer with normalisation (N) on input features using normalisation parameters (P^X_{D1trx}) learned from the trainingx set of WM1 ($D1_{trxN}$). $D1_{trx}$ and $D1_{tst}$ stand for the trainingx set and test set of WM1. $D1_{trxN}$, $D1_{tstN}$, $D3_{trxN}$ and $D3_{tstN}$ stand for normalised $D1_{trx}$, $D1_{tst}$, $D3_{trx}$ and $D3_{tst}$, respectively.

Revised experiment with normalisation. The result of the preliminary experiment will reveal that using normalisation parameters learned from WM1 to normalise data in WM3 causes problems. The problem rises from the discrepant statistics between the datasets.

In light of the problem of discrepancy of statistics, the experiment will be revised to a version that allows learning “local" normalisation parameters from each trainingx set for both input feature and output feature, and

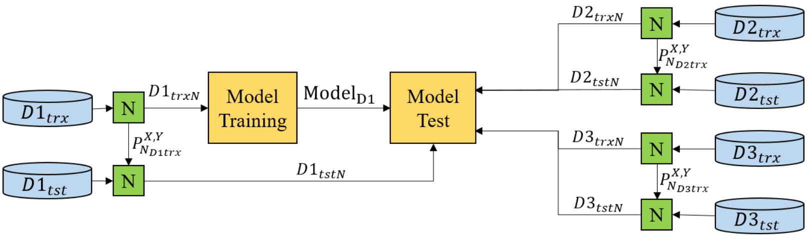


Figure 4.29: Normalisation parameters learned from each trainingx set and applied to trainingx set and test set before model transfer. The results are better than that in Figure 4.28, and therefore the model transfer is tested on WM2 and WM3.

applying the learned normalisation parameters on the trainingx set and its corresponding test set, respectively, as shown in Figure 4.29. Before testing the models on WM2 and WM3, the normalisation parameters will be learned from trainingx set of WM2 and applied to trainingx set and test set of WM2. The same workflow repeats for models trained on trainingx set of WM3 and tested on WM1 and WM2.

4.5.4 Results and Discussion

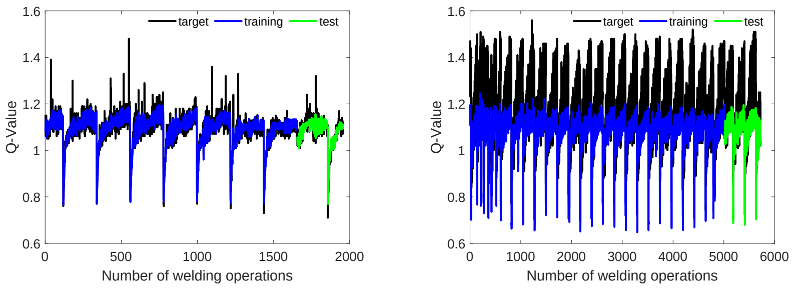
Preliminary experiment and the revealed problem. The results of the preliminary experiment are listed in Table 4.24 and illustrated in Figure 4.30.

Table 4.24: Performance of the model trained on $D1_{trx}$ (trainingx set of WM1) and tested on $D1_{tst}$ and $D3_{tst}$ (test set of WM1 and WM3) in the preliminary experiment. Normalisation parameters are only learned from $D1_{trx}$. The blue texts indicate the training set and test set are collected from the same welding machine.

	Training set	Method	mape of model testing			
			$D1_{trx}$	$D1_{tst}$	$D3_{trx}$	$D3_{tst}$
Row1	$D1_{trx}$	MLP	1.92%	1.81%	10.3%	10.0%

Table 4.21 shows that the model transfer results on WM3 are significantly worse than that of WM1, on which the model is trained. Comparing the (a) and (b) in Figure 4.30, it can be seen clearly that there exists a discrepancy of mean value when testing the model.

Since normalisation is only performed on input features, not on output (Q-Value), it is natural to hypothesise that the mean value of Q-Values of WM1 is different from that of WM2. A simple calculation gives that the mean value of Q-Value in $D1_{trx}$ is 1.09, and that in $D2_{trx}$ is 1.17 (anonymised). This confirms the hypothesis. Furthermore, the statistics, e.g. mean values and standard deviations, of input features in WM1 also differ from that in WM3. The experiment of directly transferring a model trained on WM1 to WM3 reveals the problem caused by the difference of statistics between datasets. To improve the results, normalisation parameters should be learned "locally" from the dataset to be transferred to.



(a) (b)

Figure 4.30: (a) Results of the MLP model trained on and tested on WM1. It is the complete version of Figure 4.25d.
(b) Results of the MLP model trained on the trainingx set of WM1 and tested on WM3.

Revised experiments. The results of the revised experiment are listed in Table 4.25. Each row lists the performance of the models trained on the dataset in the "Training set" column and tested on several datasets. The blue texts indicate the case when the data are trained and tested on the datasets collected from the same welding machine.

In particular, Row 1 and Row 2 list performance of ML models trained on $D1_{trx}$ and tested on $D1$, $D3$, and $D2$ with the methods LR and MLP. Row 3 and Row 4 list performance of ML models trained on $D3_{trx}$ and tested on $D1$, $D3$, and $D2$ with the methods LR and MLP. Note that to evaluate the performance of model transfer to WM2, the performance of models trained on $D2$ and tested on $D2$ is needed, and is added in Row 5. Comparing Row 1 in Table 4.24 and Row 2 in Table 4.25 reveals that learning normalisation parameters from each dataset improves the performance slightly, from about 10% mape to 8% or 9%.

However, in general the performance of models transferring is not good. The performance of Benchmark 3 for WM1, WM2, and WM3 is 2.69%, 2.67%, and 5.17%. All performance of model transfer is worse than that. By comparing the model performance in each column, it shows that the model performance deteriorates significantly when testing on the datasets collected from other welding machines. In particular, looking at Column 1 and Column 2, it reveals that the model performance trained on $D1$ and testing on $D1$ is about 2% mape, but the mape is more than 5% when the model is trained on $D2_{trx}$. The same phenomena repeat when looking at Column 3 and Column 4. The added Row 5 further confirms this.

Figure 4.31 illustrates the prediction results of MLP and LR models trained on $D1_{trx}$ and tested on $D3$ and $D2$. It reveals although the problem of mean shift is eliminated by learning "local" normalisation parameters, the discrepant dynamics in behaviours of Q-Values still cause problems for model transfer. On one hand, the Q-Values of WM3 have a larger range

Table 4.25: Performance of the model trained on $D1_{trx}$ (trainingx set of WM1) or $D3_{trx}$ (trainingx set of WM3) and tested on $D1_{tst}$, $D2_{tst}$, and $D3_{tst}$ (test set of WM1, WM2 and WM3) in the revised experiment. Normalisation parameters are learned from each trainingx set. The blue texts indicate the training and test set are collected from the same Welding Machine.

	Training set	Method	mape of model testing					
			Col1	Col2	Col3	Col4	Col5	Col6
			$D1_{trx}$	$D1_{tst}$	$D3_{trx}$	$D3_{tst}$	$D2_{trx}$	$D2_{tst}$
Row1	$D1_{trx}$	LR	1.93%	1.92%	9.11%	8.65%	5.71%	4.53%
Row2	$D1_{trx}$	MLP	1.93%	2.03%	8.93%	8.40%	5.64%	4.36%
Row3	$D3_{trx}$	LR	6.00%	6.05%	3.02%	3.40%	3.95%	3.49%
Row4	$D3_{trx}$	MLP	5.36%	5.38%	2.72%	3.10%	3.90%	3.56%
Row5	$D2_{trx}$	LR	-	-	-	-	2.25%	2.07%

than that of WM1 and have more complex dynamics (WM3 has 6 welding programs). The models trained on $D1_{trx}$ are not capable to cope with the complexity. On the other hand, the models trained on $D3_{trx}$ are too complex for transferring to WM1. By comparing Figure 4.30 and Figure 4.31c and d, it reveals that the predicted Q-Values are "chunkier" than they should be, because the Q-Values of WM3 are "chunkier" than that of WM1 in the illustrated shape.

4.5.5 Conclusion and Outlook

Conclusion. In summary, transferring a model by simply training the model on one dataset and applying on another is not sufficient. Normalisation of

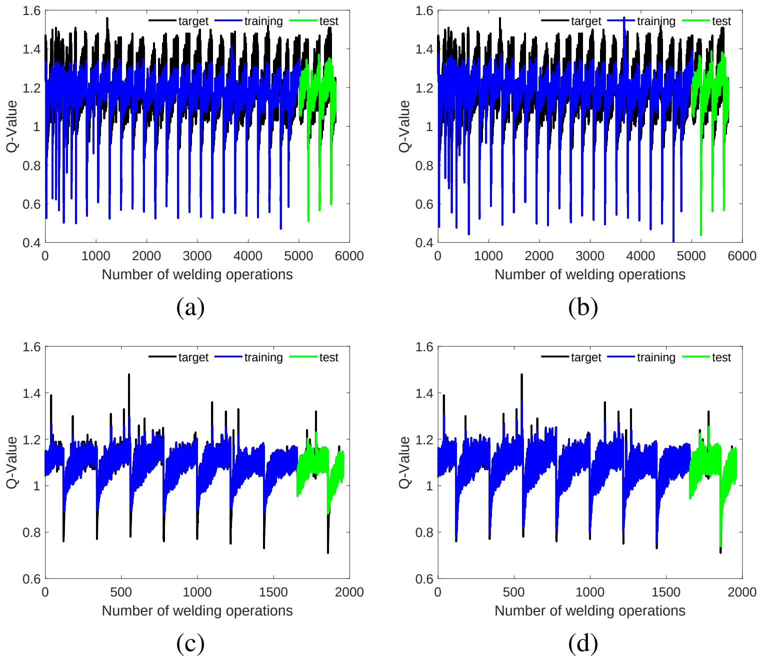


Figure 4.31: The results of the MLP model (a) or LR model (b) trained on $D1_{trx}$ and tested on $D3$. The results of the MLP model (c) or LR model (d) trained on $D3_{trx}$ and tested on $D1$.

the new dataset with the learned parameters brings a slight improvement but also does not provide satisfactory results. Transferring a model trained with more complex data to simpler data is more promising than the other way around.

Outlook. In future work, the following directions can potentially improve the model transfer performance:

- In this use case, learning "local" normalisation parameters has used a large amount of data, which does not reduce the effort and cost of data collection. In the future, less data should be used to learn the normalisation parameters, and a dynamic approach for adjusting the normalisation parameters can be investigated, so that learning "local" normalisation parameters indeed becomes practical.
- To adopt a transfer learning approach to fine-tune ML models on new datasets and studying the necessary data amount for fine-tuning.
- To develop an active learning approach to constantly adjusting ML models on new datasets. This could also handle the OQ7 concept drift problem.
- To train a Master Model on a large dataset collected from a group of welding machines and fine-tuning the master model to tailor to specific welding machines. The Master Model should be trained on a very complex dataset, as the results indicate transferring from a model trained from complex data to simpler data is more promising than the other way around.

4.6 Transferability to Another Application: Laser Welding

Overview: This use case tests the transferability of the proposed framework and approaches to another quite different welding process, laser welding. The methods in this use case are however not limited to laser welding, but also apply to other laser processes, e.g. high-precision laser machining and surface treatment. This use case study is accomplished in cooperation with Dmitriy Mikhaylov. The author's contribution is conceptualisation, design, and implementation of ML methods of Neural Networks. The following publications are based on/related to this use case: [200], [201]. An overview of the corresponding application questions, open questions and methods is listed in Table 4.26.

Table 4.26: Overview of application questions (AQ), open questions (OQ) and methods (M)

Application Question	Open Question	Methods
AQ2, AQ6	OQ1, OQ5, OQ8	M2.1.3, M2.2.3, M2.2.4

The target output of the machine learning models is the input laser amplitude matrix to an optic system for achieving the desired output laser amplitude distribution. This is a supervised multivariate regression problem. The results of ML models are tested in laboratory experiments to validate the methods and compared to conventional non-ML benchmarks.

In addition, an analysis of the influence of amount of training data on prediction accuracy is conducted to gain insights for preparing costly data collection from laboratory or production.

The data for model development are collected in a laboratory setting. The tested machine learning methods are linear regression and convolutional neural networks.

4.6.1 Question Definition

Liquid crystal on silicon phase-only spatial light modulator (LCoS-SLM) is a type of optical device often used to generate beam splitter patterns (denoted as a matrix A) for parallelised high-precision laser processes. The accuracy of laser power amplitude distribution of these processes is essential for many laser applications, e.g. multifocal fluorescence microscopy [202, 203, 204], two-photon polymerisation [205] or high-precision laser material processing.

In application, the laser light from the laser source (Figure 4.32) is expanded to a broader single laser beam by the beam expander, then transformed to a set of beams by the spatial light modulator, and goes through a series of optical devices and reaches the workpiece. The output laser amplitude distribution can be measured by a camera if an extra beam attenuator is installed to scale down the laser amplitude. The measured amplitude distribution can be described by a matrix A_{meas}. An ideal optical system will output the desired amplitude distribution (A_{des}) so that $A_{meas} = A_{des}$, while in reality the A_{meas} normally differs from A_{des} to some degree.

In order to achieve an A_{meas} more close to A_{des}, a phase hologram (ϕ) needs to be generated and fed into the SLM. Many algorithms [206] (known as *phase retrieval algorithms*) attempted to calculate the phase hologram ϕ from an input amplitude distribution matrix (A_{in}), by iterative or non-iterative methods. Among which, the weighting *Iterative Fourier Transform algorithm (IFTA)* is a most commonly used method because it has fewer constraints and enables fast implementation and calculation [207, 208]. Traditionally, the A_{in} is directly set to A_{des} and ideally A_{meas} should be equal to A_{des}.

However, the actual A_{meas} deviates from A_{des} to a certain degree, because IFTA assumes an ideal optical setup, differing from the real experimental system. To overcome this small deviation, an extended IFTA with a closed camera feedback loop (referred to as Camera-IFTA) was proposed [209],

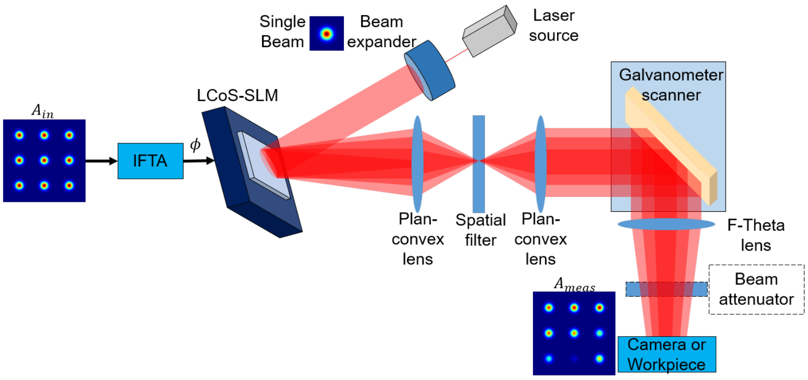


Figure 4.32: Schematic illustration of an application of the liquid crystal on silicon phase-only spatial light modulator (LCoS-SLM). ϕ indicates the phase hologram calculated by Iterative Fourier Transformation Algorithm (IFTA). An extra beam attenuator is needed for scaling down the laser amplitude to measure the output amplitude distribution (A_{meas}) using a camera. The figure is modified from [201].

achieving higher accuracy, but increasing the computational time significantly.

To achieve both high accuracy and fast computation, this section proposes novel approaches of **ML-aided phase retrieval algorithms** to combine IFTA, camera feedback, and machine learning algorithms.

The basic idea is to extend the standard IFTA with an ML model for computing the phase hologram, so that the ML-aided IFTA can achieve higher accuracy than IFTA, and with much less time than the Camera-IFTA. The ML model is pre-trained using a dataset with A_{meas} as the input data and A_{in} as the output data ($A_{in} = f_{ml}(A_{meas})$), so that the ML model can be seen as an **inverse model** of the optical system ($f_{ml} = f_{optics}^{-1}$). The optical system includes IFTA, SLM and the subsequent optical setups until the camera in Figure 4.32. After that, the desired distribution A_{des} is input to the ML model to estimate the output of the ML model, which is an estimated input of the optical system $\hat{A}_{in}$ that should have A_{des} as output, i.e. $\hat{A}_{in} = f_{ml}(A_{des}) \Rightarrow A_{des} = f_{optics}(\hat{A}_{in})$. The extended ML-aided IFTA pipeline is illustrated in Figure 4.33.

Three performance metrics of the proposed ML-aided phase retrieval algorithms, including prediction accuracy evaluated using mape, minimal training data number and computational time, are of interest and are compared with the standard IFTA and Camera-IFTA.

4.6.2 Data Description

The data are collected from LCoS-SLM, including the input matrices A_{in}, and the actual output matrices A_{meas} of the optical system. In total, 1000 instances of each type are collected, and each instance is a 10×10 matrix, representing the laser amplitude distribution. Therefore, a data tuple is a 10×10 matrix in this use case. Three types of hologram matrices are collected: random beam profiles, homogeneous distributed profiles, and profiles of linear ramps. Since the actual laser amplitude will be scaled up and down in different components of the optical system according to the requirements of different applications, the absolute scale is not important. Hence the data are normalised to sum to 1 after collection.

4.6.3 Experiment Settings

Two steps of experiments (Figure 4.33) are designed to evaluate the proposed ML-aided phase retrieval algorithms.

Step 1: Inverse modelling. In Step 1, laser amplitude distribution matrices A_{in} (equal to the desired distribution matrices A_{des}) are input into the optical system, passing through IFTA, SLM and optical setups, resulting the actually measured amplitude distribution matrices A_{meas}. These two matrices are used to train ML models that map the A_{meas} to A_{in}, exactly inverse to the input and output of the optical system.

The data comprised of pairs of input and output, i.e. A_{meas} and A_{in}, are split into trainingx set and test set with a ratio of $0.8 : 0.2$. The trainingx set is again split into training set and validation set with a ratio of $4 : 1$. Two

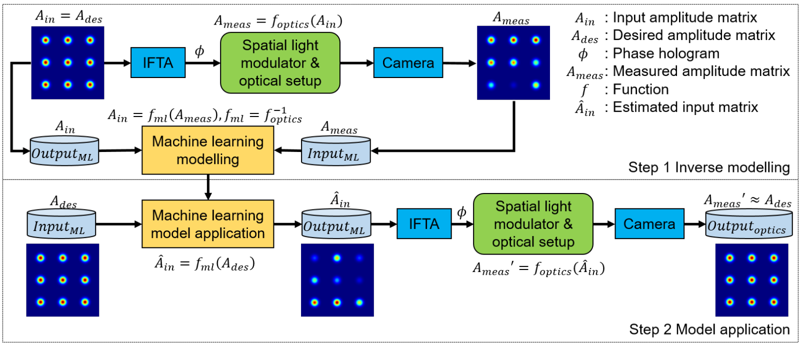

$A_{in} = A_{des}$
IFTA
ϕ
$A_{meas} = f_{optics}(A_{in})$
Spatial light modulator & optical setup
Camera
A_{meas}
A_{in} : Input amplitude matrix
A_{des} : Desired amplitude matrix
ϕ : Phase hologram
A_{meas}: Measured amplitude matrix
f : Function
$\hat{A}_{in}$: Estimated input matrix
$A_{in} = f_{ml}(A_{meas}), f_{ml} = f_{optics}^{-1}$
A_{in}
$Output_{ML}$
Machine learning modelling
A_{meas}
$Input_{ML}$
Step 1 Inverse modelling
A_{des}
$Input_{ML}$
Machine learning model application
$\hat{A}_{in} = f_{ml}(A_{des})$
$\hat{A}_{in}$
$Output_{ML}$
IFTA
ϕ
Spatial light modulator & optical setup
$A_{meas}' = f_{optics}(\hat{A}_{in})$
Camera
$A_{meas}' \approx A_{des}$
$Output_{optics}$
Step 2 Model application

Figure 4.33: Extended ML-aided IFTA with 2 steps: inverse modelling and model application. The figure is modified from [201].

types of ML models, regularised Linear Regression (LR) and Convolutional Neural Networks (CNN) are trained.

The CNN models are first trained on the training set. The hyper-parameters of CNN models are selected based on the performance on the validation set. A series of architectures, including different layers, number of layers and number of neurons, were experimented using limited grid search (varying one while fixing other hyper-parameters). In particular, using C denoting the Conv2d layer, D denoting the Dense layer, the tested architectures include: C, CC, CCC, CCCC, CD, CCD, and CDC. Among which, two architectures, CC and CD, prevail in performance and are selected. The best CC network selected based on the performance of validation set is Conv256-Conv1 with filter sizes of (6,6) and (4,4) for each layer. The best CD network is Conv320-Dense100 with a filter size of (6,6) in the Conv2d layer. All activation functions are ReLU (rectified linear units). The “Adam” optimiser is adopted, and loss function is mape.

After that, CC and CD with the best hyper-parameters and LR are trained again with the trainingx set (combining training and validation set) and tested on the test set (200 matrices), and evaluated with mape and maxape (maximal absolute percentage error).

Table 4.27: Data splitting to seven subsets of different training data number

Dataset name	D_1	D_2	D_3	D_4	D_5	D_6	D_{all}
#Trainingx data	10	50	101	150	200	400	800
#Test data	200	200	200	200	200	200	200

Step 2: Model application. In Step 2, the ML models are applied and the results are validated in an experimental setting. In particular, the desired distribution matrices A_{des} of the test set (200 matrices) are used as input to the ML models (in contrast to Step 1 where A_{des} is used as output) to estimate the input distribution of optical systems $\hat{A}_{in}$, that should have A_{des} as output.

Then the estimated distribution matrices $\hat{A}_{in}$ are input into the optical system in the laboratory to generate the output, measured as A'_{meas}. These A'_{meas} are compared with A_{des} using mape and maxape to quantify the performance of ML models in experimental validation.

Data splitting into subsets of different training data numbers. To address the open question how many minimal data are necessary, the training data set is split into a series of subsets of different numbers of data tuples (Table 4.27), similar to Use Case 4.1. All three selected ML models, LR, CC and CD are trained with each subset of training data and evaluated in terms of prediction accuracy and experimental validation.

4.6.4 Results and Discussion

The performance in model application (Step 2) of the best ML-aided IFTA, whose ML models are trained with 800 trainingx set, are validated with the test set (200 matrices) and compared with two benchmarks in Table 4.28. Note that the IFTA calculation in each phase retrieval algorithm has internal errors. Thus the error of any IFTA-based phase retrieval algorithm cannot be lower than the IFTA calculation error. This error is also listed in Row 1 in Table 4.28 for reference. The two benchmarks are standard IFTA (referred to as IFTA) and extended IFTA with a closed camera feedback loop (referred to as *Camera-IFTA*).

Table 4.28 reveals that the ML-aided phase retrieval algorithms have significantly better performance compared to the standard IFTA. Especially the LR-aided IFTA and CC-aided IFTA can even approximate the performance of Camera-IFTA.

The mape is the mean absolute error calculated by comparing the A'_{meas} and A_{des}. Since each A'_{meas} is a 10×10 matrix, the comparison of each matrix has a mape. The average values of mape ($\hat{\mu}$) of the 200 test data tuples for each method are listed in Column 2, while the standard deviation of mape ($\hat{\sigma}$) are listed in Column 3. Similarly, the maxape is the maximal absolute error of the comparison between A'_{meas} and A_{des}. Its average values and standard deviations are listed in Column 4 and Column 5.

Table 4.28: Performance comparison of ML-aided IFTA with the best ML models

Method	mape (%)		maxape (%)	
	$\hat{\mu}$	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
IFTA calculation	1.11	0.37	5.06	1.99
IFTA	7.14	0.35	24.7	1.92
Camera-IFTA	1.93	0.31	7.80	2.33
LR-aided IFTA	2.36	0.40	8.46	1.66
CC-aided IFTA	2.46	0.29	8.99	1.95
CD-aided IFTA	3.59	0.47	13.00	2.50

A further analysis of necessary training data number for ML models is illustrated in Figure 4.34. The two figures on the left side are the results of model prediction accuracy in Step 1 of Figure 4.33, and on the right side are the results of model application accuracy in Step 2.

Figure 4.34 gives the relationship between method performance and number of data tuples used for model training and validation. Sub-figures of both prediction performance (left column) and experimental validation performance (right column) indicate that LR and CC are much better than CD when training data number < 400. After that their performance becomes comparable. LR and CC are quite close in performance when training data

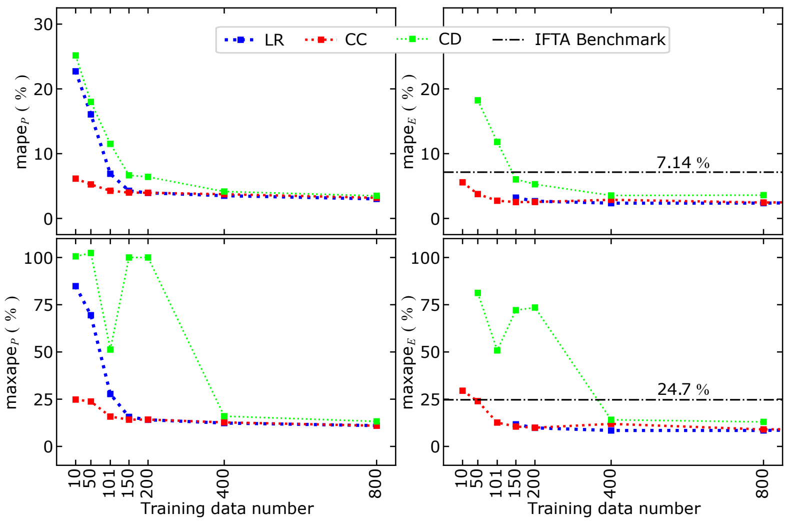


Figure 4.34: ML models for prediction and experimental validation compared to IFTA. The figure is modified from [201].

number ≥ 150. CC can still achieve relatively good performance when training data number is only 101, while LR needs the training data number to be at least 150, because the Theory of Equation requires the number of equations > number of variables.

In summary, the ML-aided IFTA need at least 150, 101, 400 training data tuples for the ML model of LR, CC, and CD respectively, to achieve significantly improvement compared to standard IFTA. This is summarised in Table 4.29.

Table 4.29 evaluates the ML-aided methods considering minimal number of training data, and computational time. The standard IFTA takes 40 seconds for each hologram computation with relatively large errors. The Camera-IFTA takes 280 seconds for each hologram computation with significantly better performance, serving as benchmark for the proposed ML-aided methods. The LR-aided IFTA needs at least 150 training data, requiring about 270 seconds for data collection and neglectable training time.

Once the LR model is trained, no further data collection time and training time is needed. After that, the computation of each hologram takes 40 seconds, since the method is based on IFTA. This means the LR-aided IFTA becomes more efficient from the computation of the 2nd hologram (270s + 40s $\times$ 2 = 350s) than the Camera-IFTA (280s $\times$ 2 = 560s), while achieving a comparably good performance as Camera-IFTA. Similarly, the CC-aided IFTA saves time already from the 1st hologram (182s + 40s = 222s $<$ 280s) than Camera-IFTA, and CD-aided IFTA saves time from the 4th hologram (720s + 40s $\times$ 4 = 880s, better than that of Camera-IFTA 280s $\times$ 4 = 1120s).

Table 4.29: Evaluation of ML-aided methods considering time

Method	Minimal #trainingx data	Minimal data collection time	Training time	Hologram computation time	#Data from which ML method saves time
IFTA	-	-	-	40s	-
Camera-IFTA	-	-	-	280s	Benchmark
LR-aided IFTA	150	270s	<1s	40s	2
CC-aided IFTA	101	182s	5s	40s	1
CD-aided IFTA	400	720s	38s	40s	4

4.6.5 Conclusion and Outlook

Conclusion. This use case has studied improving phase retrieval algorithm in laser beam splitting application by extending the standard IFTA with ML models. The performance of the ML models is validated both from ML perspective with training and test, and from domain perspective with laboratory experiments. To further improve the adoptability and efficiency, the influence of training data number on model performance is studied. All ML methods have shown comparable performance and improvement in computation time, compared to the state-of-the-art phase retrieval algorithm, the Camera-IFTA, given that sufficient training data are provided. ML-aided IFTA has better comprehensive performance if several holograms need to be calculated. Especially CC-aided IFTA surpasses Camera-IFTA in terms of efficiency for even the calculation of the first hologram.

Outlook. In the studied use case the input and output of the optical system are required to be *dense matrices*. In industrial application, sometimes *sparse matrices* are used. In this case, the three proposed ML methods, LR, CC and CD, become not suitable anymore. The sparse matrices have two complexity levels. The simpler version is when the number of laser spots in the input amplitude matrix A_{in} equals the number of laser spots in the output matrix A_{meas}. In this case, the spatial information of the laser spots can be provided as features for ML modelling. The more complex version is when the number of laser spots in the input matrix A_{in} does not equal that in the output matrix A_{meas}. This case may be handled by padding the matrix with the shorter length. These questions remain as future research direction.

5 Conclusion and Outlook

Aiming on studying and developing effective and generalisable machine learning frameworks for quality monitoring in electrical resistance welding (Section 1.6), this work goes beyond the boundaries of data science, combining perspective of machine learning, domain knowledge and semantic technologies. This chapter first summarises the study concisely, then makes several conclusions, and finally previews future research.

Summary. Firstly, this work reviews (Chapter 1) state-of-the-art of machine learning methods for product quality monitoring in electric resistance welding (ERW) and summarises open questions in this field. To address these open questions, this work proposes a framework of ML methods (Chapter 2), which expands the angle from traditional machine learning to deep integration of domain knowledge and to semantic technologies such as ontologies and templates. Then, a concise description of some technical points for implementation of the framework (Chapter 3) is given. After that, this work demonstrates six industrial use cases (Chapter 4) to exemplify and evaluate some methods in the framework.

Contribution. The major contribution of this work is a framework of machine learning in ERW that pursues a comprehensive coverage on topics of quality monitoring in ERW. In particular, it includes the following contributions.

1. Revealing many domain aspects and data particularities in ERW that are little discussed in the literature (Section 2.1), including: (1) quality indicators subsumed into six fidelity levels and accessibility; (2)

the multi-levels of temporal structures in data; (3) insufficient data problem summarised into three data challenges; (4) a comprehensive list of application questions related to three data sources: production, lab, and simulation;

2. A novel simulation-supported data collection regime for ERW (Section 2.2), including: (1) procedures and methods of interactive data acquisition and analysis; (2) datasets similarity analysis; (3) inverse modelling of simulation model; (4) data amount and feature sets analysis; (5) datasets evaluation;

3. A new ontology-supported data preparation scheme for ERW (Section 2.3), including: (1) convenient domain ontology construction by users based on upper ontology and templates for process and data understanding; (2) ontology-aware data integration to the *Uniform Data Format* to unify the data from different sources, versions, and processes; (3) a mechanism to link the *Uniform Data Format* to machine learning;

4. A systematic approach for selecting suitable ML methods and constructing ML pipelines for ERW (Section 2.4), including: (1) a novel data preprocessing design to handle the temporal structures by special data splitting, reshaping, and hierarchical feature extraction; (2) several feature engineering strategies with deep integration of domain knowledge; (3) discussion of feature selection, normalisation and classic machine learning methods; (4) ontology-enabled ML enhancement that allows semi-automated and user-configurable machine learning pipeline construction; (5) a special design of neural networks for hierarchical feature learning; (6) discussion of performance metrics, such as prediction accuracy and several adoptability metrics;

5. Successful validation of the proposed ML framework with five industrial use cases of ERW, including: (1) simulation-supported data analysis and collection (Section 4.1), (2) visualisation and evaluation of a large amount of production data to detect conspicuous welding machines and welding programs for further analysis (Section 4.2), (3) regression analysis to predict the next quality for comparing two school of ML pipelines: feature engineering - classic ML and feature learning - neural networks (Section 4.3), (4) regression analysis with deep integration of domain knowledge and gaining insights from feature evaluation (Section 4.4); (5) transferability analysis of ML models trained with data of one machine to data of other machines (Section 4.5).

6. Successful demonstration of the transferability of the proposed ML framework with an industrial use case of high-precision laser processes (Section 4.6);

7. The implementation of the proposed ML framework with two programming platforms, Python and SciXMiner (MATLAB) (Chapter 3), including six ML modules and ML pipeline components of data preparation, data preprocessing and ML modelling.

Conclusion. From the review of state-of-the-art and open questions, it can be seen that the difficulties of efficient machine learning in ERW, as well as in manufacturing, are not development of novel and complex ML algorithms. Instead, a system solution is needed that covers multi-faceted topics from data collection, through data management, and to ML development.

This work proposes a framework of ML in ERW for quality monitoring, aiming at establishing a system solution to cover these broad topics, which naturally is not confined to the application of quality monitoring in ERW but also applicable to other questions of similar mathematical nature in discrete manufacturing. The target-oriented approach results in an interdisciplinary framework that combines three perspectives: data science, manufacturing technologies, and semantic technologies. This work designs the architecture of the framework and provides theoretical corner stones as well as evaluation use cases in the five aforementioned topics: question definition, data collection, data preparation, data preprocessing and ML modelling.

Nevertheless, much work remains to be done to further evaluate and improve the proposed framework. This work starts the journey towards more intelligent manufacturing, which merges to the grand trend of the Fourth Industrial Revolution (Industry 4.0).

Outlook. For the following topics, this work has only proposed theoretical approaches. They need to be further evaluated and validated: similarity analysis between datasets to support data collection and evaluation, inverse modelling of simulation models, ontology-supported process & data understanding, ontology-supported data integration, ontology-enhanced machine learning for feature engineering - classic machine learning.

The theories in the following aspects need to be further developed and completed: addressing the third data challenge of limit on coverage of relevant situations, a more general semantic framework for knowledge encoding and data modelling in discrete manufacturing, a better combined use of multiple data sources in the multi-fidelity data model, quality monitoring with probabilistic estimation, concept drift monitoring, ontology-enhanced machine learning for feature learning - neural networks.

The ML framework proposed by this work awaits further study in other processes of discrete manufacturing and process manufacturing [174] to explore the transferability, e.g. hardening [210], machining [211, 212], where factors like system input parameters (e.g. force, speed), system component status (e.g. tool wear), and system feedbacks (e.g. temperature, stress) are extensively studied for quality analysis. In summary, a deep intertwining of domain knowledge, machine learning, and semantic technologies is expected as the general manner for many other research realms in Industry 4.0.

A Appendix

A.1 List of Figures and Tables

List of Figures

1.1 GDP by sectors in Germany and worldwide 1
1.2 Resistance spot welding and measuring the spot quality using destructive methods 5
1.3 A general process pipeline for electric resistance welding 6
1.4 Two-dimensional Gaussian distribution of QI_{meas} and QI_{emp}. . . 8
1.5 Pyramid of data mining 12
1.6 A typical machine learning workflow 13
1.7 Ontology and its types 17
1.8 An example of ontology templates 18

2.1 Quality indicators of different levels 32
2.2 Q-Value along number of welding operations 34
2.3 Examples of process curves and consecutive welding operations 36
2.4 Schematic illustration of a welding process from the perspective of one machine 37
2.5 Temporal structures in the data 38
2.6 Sequential structure of a simplified RSW production line 38
2.7 Parallel structure of a simplified hot-staking production line . . . 39
2.8 Multi-fidelity data model 41
2.9 Simulation-supported interactive data analysis and collection . . 46
2.10 Correlation and causality pattern of a dataset 47
2.11 ML for inverse modelling of a simulation model 48
2.12 Q-Values, Trend, and Scattering across multiple dress cycles . . 52
2.13 Overview of ontology-supported machine learning development 57
2.14 Schematic illustration of the upper ontology *core* 60

2.15 Schematic illustration of the templates of physical entity, its properties, and subclasses, including machine, machine part, raw product and product part. 61
2.16 Schematic illustration of the templates of observation collection and quality indicator 62
2.17 Schematic illustration of the domain ontology *rsw* 63
2.18 Workflow of ontology supported machine learning pipelines . . 70
2.19 Hierarchical feature extraction 71
2.20 An example of data reshaping to handle data of hierarchical temporal structures 72
2.21 Example of data splitting rounded to complete dress cycles . . . 73
2.22 Data overlapping issue because of data reshaping 75
2.23 Feature engineering on welding time level 76
2.24 Example for generating EngF_Prog with data of WM1 79
2.25 Schematic illustration of part of the machine learning ontology *ml*. 84
2.26 Schematic illustration of the templates for feature processing . . 85
2.27 Schematic illustration of the template for ML modelling. 86
2.28 Schematic illustration of an example ML pipeline ontology. . . . 86
2.29 An example of ontology-enhanced ML pipeline of FE-LR 87
2.30 An example of hierarchical feature learning pipeline with LSTM 89

3.1 Architecture of implementation scripts 94

4.1 Line-plot of process curves and boxplot of diameters 106
4.2 Model performance of Polynomial1 trained on non-noisy subsets 110
4.3 Testing results MLP16 on non-noisy datasets 111
4.4 Testing results KNN3 on non-noisy datasets [29] 111
4.5 Testing models trained on non-noisy Production Set 111
4.6 Testing Polynomial1 on noisy datasets [29] 111
4.7 Overview of dataset metrics in the time level of welding machine 116

4.8 Discrepancy metrics of conspicuous welding machines. 119
4.9 Overall metrics of WM_B 120
4.10 Q-Value along welding operations of WM_B 121
4.11 Metrics and scatter plots of Q-Values of WM_I 121
4.13 Metrics of WM_R. 122
4.12 Q-Values over WearCount of WM_R 122
4.14 Envisioned welding workflow with quality prediction 125
4.15 Q-Value along number of welding operations for WM2 126
4.16 Machine learning pipeline of feature engineering and feature learning 127
4.17 ML pipeline of feature engineering and feature learning 129
4.18 Hyper-parameter selection on Welding Machine 1 131
4.19 Results of the best FE and FL models 135
4.20 Q-Value along number of welding operations for WM3 140
4.21 Example of data splitting rounded to complete dress cycles . . . 141
4.22 Domain knowledge supported time series feature extraction . . . 143
4.23 Reducing features by feature selection 144
4.24 Hyper-parameter selection with limited grid search on WM1 and WM3 146
4.25 Prediction results zoomed into test set area of LR models on WM1. 149
4.26 Examples of prediction results on test set of WM3 154
4.27 The simple workflow of model transfer 156
4.28 Workflow of preliminary experiment for model transfer 158
4.29 Revised workflow for model transfer 159
4.30 Results of the MLP model trained on the trainingx set of WM1 . 160
4.31 The results of the model trained on WM1 and tested on WM3 and vice versa 162
4.32 Schematic illustration of an application of the liquid crystal on silicon phase-only spatial light modulator (LCoS-SLM) 166
4.33 Extended ML-aided IFTA with 2 steps 168

4.34 ML models for prediction (P) and experimental (E) validation compared to IFTA 171

A.1 Metrics and scatter-plots of Q-Values of WM$_O$ 192
A.2 Metrics with respect to welding programs and cap dresses of WM$_T$ 193
A.3 Q-Values along welding operation and over WearCount of WM$_T$ 193

List of Tables

1.1 Contribution of each sector (percent) of 24 countries in OECD for 2000-14 2
1.2 Confusion matrix of estimated value and predicted value 9
1.3 An overview of related works of machine learning in ERW, Part 1 24
1.4 An overview of related works of machine learning in ERW, Part 2 26

2.1 Correspondence table of temporal structure features ($\mathbf{F}_t$) and time levels 35
2.2 Properties of three data sources 42
2.3 Overview of all AQs, OQs, methods and use cases. 44
2.4 Data splitting to subsets of different training data size 49
2.5 Data splitting into four subsets of different features 50
2.6 Metrics to quantify data behaviour 55
2.7 The raw-to-domain mapping that maps raw feature names to terms in domain ontology *rsw* and the unified feature names. . . . 68
2.8 Example of the DO-to-FG mapping that maps domain feature names in *rsw* to ML feature groups in *ml* 69

4.1 Overview of application questions (AQ), open questions (OQ) and methods (M) 104
4.2 Data splitting into seven subsets of different training data number 107
4.3 Data splitting into four subsets of different features 107
4.4 Data splitting into four subsets of different features 108
4.5 ML models and hyper-parameters selected through 5-fold cross validation 109

4.6 Overview of application questions (AQ), open questions (OQ) and methods (M) . . . 114
4.7 Overview of application questions (AQ), open questions (OQ) and methods (M) . . . 124
4.8 Two example datasets for prediction of the next Q-Value 126
4.9 Hyper-parameters that are fixed once chosen . . . 132
4.10 Hyper-parameter selection of ML methods . . . 133
4.11 Model performance tested on test sets . . . 134
4.12 More comprehensive evaluation of ML pipelines . . . 136
4.13 Overview of application questions (AQ), open questions (OQ) and methods (M) . . . 139
4.14 Two example datasets for prediction of the next Q-Value 140
4.15 Hyper-parameter selection of ML methods . . . 145
4.16 LR performance on test set of WM1. Imprv. w.r.t.: improvement with respect to. . . . 147
4.17 LR performance on test set of WM3. Imprv. w.r.t.: improvement with respect to. . . . 147
4.18 Feature evaluation for WM1 . . . 150
4.19 Feature evaluation for WM2 . . . 152
4.20 MLP and SVR models evaluated on test set of WM1 . . . 153
4.21 MLP and SVR models evaluated on test set of WM3 . . . 153
4.22 Overview of application questions (AQ), open questions (OQ) and methods (M) . . . 156
4.23 Example datasets for studying model transferability . . . 157
4.24 Transfer of model trained on WM1 and tested on Wm1 and WM3 159
4.25 Performance of the model trained on the trainingx set of WM1 or WM3 . . . 161
4.26 Overview of application questions (AQ), open questions (OQ) and methods (M) . . . 164
4.27 Data splitting to seven subsets of different training data number . 169

4.28 Performance comparison of ML-aided IFTA with the best ML models 170
4.29 Evaluation of ML-aided methods considering time 172

A.2 List of Open Questions and Application Questions

List of Open Questions

Open Question 1 Insufficient data problem. 25
Open Question 2 Standardised process description, data description and management. 27
Open Question 3 Dataset evaluation for identifying conspicuity. 27
Open Question 4 Quality monitoring with temporal structure and probabilistic estimation. 27
Open Question 5 Evaluation and selection of machine learning methods. 27
Open Question 6 Integration of domain knowledge problem. 27
Open Question 7 Concept drift monitoring. 28
Open Question 8 Transfer of developed ML approaches to other datasets and processes. 28

List of Application Questions

Application Question 1 Dataset evaluation 42
Application Question 2 Data amount analysis 42
Application Question 3 Feature evaluation 42
Application Question 4 Classification and regression analysis for estimating 42
Application Question 5 Classification and regression analysis for predicting 42
Application Question 6 Process parameter optimisation 43
Application Question 7 Semi-supervised learning 43
Application Question 8 Similarity analysis 43
Application Question 9 Interactive data acquisition and analysis . . 43
Application Question 10 Transferability 43

A.3 Following Use Case 4.2: Dataset Evaluation of Additional Welding Machines

Identifying conspicuity of Welding Machine O. WM_O has a high level of $Trend_{mean}$ and the highest $Scattering_{mean}$ (Figure 4.7b). The overall metrics of WM_O in Figure A.1a reveals that the Prog2 has a higher $Trend_{mean}$ and Prog1 has a higher $Scattering_{mean}$. These two welding programs are both conspicuous in some way. The scatter-plots of Q-Value over WearCount for these two programs in Figure A.1b and Figure A.1c show that Q-Values of WM_O indeed have some conspicuous behaviour. The Q-Values of Prog1 diverge into two groups in the later phases of dress cycles. One group falls back to the optimal value of one, while the other group continues to rise. The Q-Values of Prog2 have a discernible phase of rising in the middle phases of dress cycles and reach a stable plateau in the later phases of dress cycles, deviating from the optimal value of one significantly.

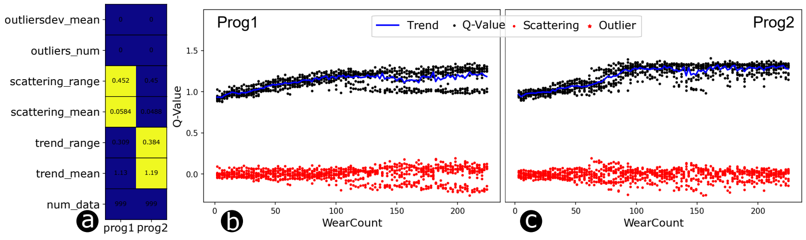


Figure A.1: Metrics and scatter-plots of Q-Values of WM_O

Identifying conspicuity of Welding Machine T. WM_T has the most $\#Outliers$ (Figure 4.7b) and the discrepancy metrics show the differences are more attributed to welding programs than dress cycles (Figure 4.8). The overall metrics with respect to welding programs in Figure A.2a reveals that almost all these outliers are from Prog27. The overall metrics with respect to dress cycles in Figure A.2b further reveals that most of these outliers concentrate on one dress cycle (cap1 dress0). After narrowing down the search area, the scatter-plots of the Q-Values along welding operation (Figure A.3a) and

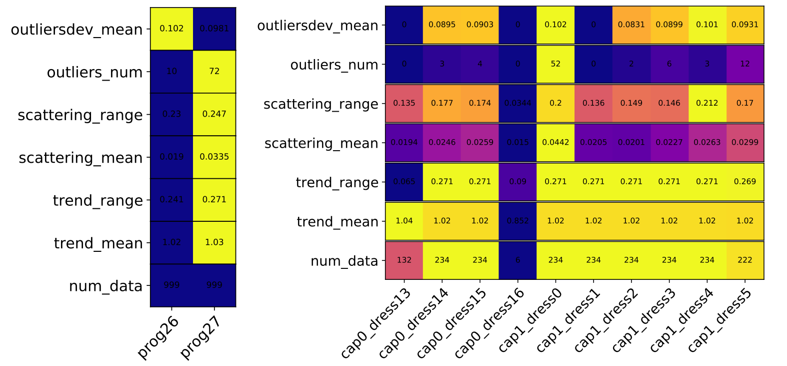


Figure A.2: Metrics with respect to welding programs and cap dresses of WM_T

over WearCount (Figure A.3b) further confirm that almost all outliers are of Prog27 in one dress cycle, where the actual Q-Values systematically deviate from the estimated trend.

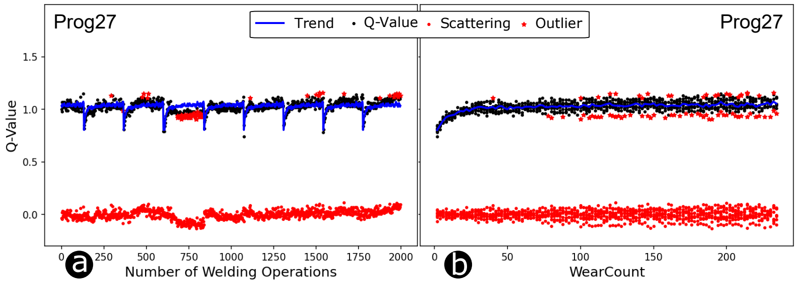


Figure A.3: Q-Values along welding operation and over WearCount of WM_T

A.4 Determination of Minimal Number of Data Tuples for Statistical Estimation

This section derives the minimal number of data tuples for estimation of mean value and standard deviation of an (assumedly) Gaussian distributed random variable X. This section is written with great support from Prof. Markus Reischl.

Estimation of Mean. Suppose the estimated mean value is estimated with n data tuples, expressed as $x_{T,n}$, and it cannot deviate from the *true mean value* μ by 10%. The probability of $|x_{T,n} - \mu| \leq 0.1\mu$ should be greater than or equal 95%. This is expressed in the following inequation:

$$P(0.9\mu \leq x_{T,n} \leq 1.1\mu) \geq 95\% \tag{A.1}$$

Inside the probability parenthesis of the left hand side of the inequation it is also an inequation. A μ can be deducted from all sides in the ineuqation in the parenthesis, and thus the inequation can be changed to:

$$P(0.9\mu - \mu \leq x_{T,n} - \mu \leq 1.1\mu - \mu) \geq 95\% \tag{A.2}$$

This can be further changed by multiplying $\sqrt{n}/\sigma$ in all sides inside the parenthesis and it becomes:

$$P(-\frac{0.1\mu}{\sigma}\sqrt{n} \leq \frac{x_{T,n} - \mu}{\sigma}\sqrt{n} \leq \frac{0.1\mu}{\sigma}\sqrt{n}) \geq 95\% \tag{A.3}$$

Since x is assumed to be Gaussian distributed, and $x_{T,n}$ is the estimated mean value of x with small sample size, then $(x_{T,n} - \mu)/\sigma$ should be normal distributed, and $\sqrt{n}(x_{T,n} - \mu)/\sigma$ should be t-distributed [1] [213, 214]. $\sqrt{n}(x_{T,n} - \mu)/\sigma$ can therefore be replaced with t_{n-1}, indicating t-distributed random variable with $n - 1$ degree of freedom. Thus the inequation can be expressed as:

$$P(-0.1\frac{\mu}{\sigma}\sqrt{n} \leq t_{n-1} \leq 0.1\frac{\mu}{\sigma}\sqrt{n}) \geq 95\% \tag{A.4}$$

[1] Student's t-distribution refers to any continuous probability distributions that often arises for estimating the mean of a normally-distributed population whose sample size is small and the population's standard deviation is unknown

To estimate the value of the probability, μ and σ will be replaced with estimated values, i.e. $x_{T,n}$ and $s_{T,n}$, respectively.

$$P(-0.1\frac{x_{T,n}}{s_{T,n}}\sqrt{n} \leq t_{n-1} \leq 0.1\frac{x_{T,n}}{s_{T,n}}\sqrt{n}) \geq 95\% \tag{A.5}$$

Using $F_X(x)$ as the cumulative distribution function of the random variable X, the inequation becomes:

$$F_X(0.1\frac{x_{T,n}}{s_{T,n}}\sqrt{n}) - F_X(-0.1\frac{x_{T,n}}{s_{T,n}}\sqrt{n}) \geq 95\% \tag{A.6}$$

This is not analytically solvable, but can be solved numerically with tables of cumulative distribution function of t-distribution.

Estimation of Standard Deviation. Suppose the estimated standard deviation is estimated with n data tuples, expressed as $s_{T,n}$, and it cannot deviate from the *true standard deviation* σ by 10%. The probability of $|s_{T,n} - \sigma| \leq 0.1\sigma$ should be greater than or equal 95%. This is expressed in the following inequation:

$$P(0.9\sigma \leq s_{T,n} \leq 1.1\sigma) \geq 95\% \tag{A.7}$$

All sides in the inequation in the parenthesis are strictly greater than 0. Thus, they can be squared, and the inequation still holds:

$$P(0.81\sigma^2 \leq s_{T,n}^2 \leq 1.21\sigma^2) \geq 95\% \tag{A.8}$$

All sides in the inequation in the parenthesis can be multiplied by a positive number $(n-1)/(\sigma^2)$ ($n > 1$), and the inequation is changed to:

$$P(0.81\sigma^2\frac{(n-1)}{\sigma^2} \leq s_{T,n}^2\frac{(n-1)}{\sigma^2} \leq 1.21\sigma^2\frac{(n-1)}{\sigma^2}) \geq 95\% \tag{A.9}$$

The σ^2 in the numerator and denominator cancel themselves and the inequation becomes:

$$P(0.81(n-1) \leq (n-1)\frac{s_{T,n}^2}{\sigma^2} \leq 1.21(n-1)) \geq 95\% \tag{A.10}$$

Since X is Gaussian distributed, thus $(n-1)s_{T,n}^2/(\sigma^2)$ is chi-square distributed with $n-1$ degree of freedom [214]. The inequation can be expressed by

$$P(0.81(n-1) \leq \chi_{n-1}^2 \leq 1.21(n-1)) \geq 95\% \tag{A.11}$$

Using $F_X(x)$ as the cumulative distribution function of the random variable X, the inequation becomes:

$$F_X(1.21(n-1)) - F_X(0.81(n-1)) \geq 95\% \tag{A.12}$$

This ineuqation is not analytically solvable, but can be solved numerically with tables of cumulative distribution function of chi-square distribution.

Bibliography

[1] Department of Economic and Social Affairs, Statistics Division: International Standard Industrial Classification of All Economic Activities (ISIC), Rev.4. United Nations (2008)

[2] European Commission: NACE: Statistical Classification of Economic Activities in the European Community. European Commission (2011)

[3] Manyika, J., Sinclair, J., Dobbs, R., Strube, G., Rassey, L., Mischke, J., Remes, J., Roxburgh, C., George, K., O'Halloran, D., Ramaswamy, S.: Manufacturing the Future: The Next Era of Global Growth and Innovation. Tech. rep., McKinsey Global Institute (2012)

[4] Dillinger, J.: The World's 20 Largest Exporting Countries. Online (2019). URL https://www.worldatlas.com/articles/exports-by-country-20-largest-exporting-countries.html. Accessed 11 September 2019

[5] Mankiw, N.G.: Principles of Economics. Cengage Learning, Boston (2014)

[6] Oxford Learner's Dictionaries, Word Entry: Productivity. Online. URL https://www.oxfordlearnersdictionaries.com/definition/english/productivity?q=productivity. Accessed 19 September 2019

[7] Saari, S.: Production and Productivity as Sources of Well-Being. Management Information Development Oy 25 (2011)

[8] statista: Germany Share of GDP by Sectors in 2018. URL https://www.statista.com/statistics/295519/germany-share-of-economic-sectors-in-gross-domestic-product/. Accessed 2 April 2019

[9] CIA: The World Factbook (Sector Composition) (2019). URL https://www.cia.gov/library/publications/resources/the-world-factbook/fields/214.html. Accessed 18 December 2019

[10] Santacreu, A.M., Zhu, H.: Manufacturing and Service Sector Roles in the Evolution of Innovation and Productivity. Economic Synopses 2, 1–3 (2018)

[11] Thoben, K.D., Wiesner, S., Wuest, T.: Industrie 4.0 and Smart Manufacturing–A Review of Research Issues and Application Examples. Int. J. Autom. Technol 11(1), 4–16 (2017)

[12] BMBF: Industrie 4.0. URL https://www.bmbf.de/de/zukunfts-projekt-industrie-4-0-848.html. Accessed 11 September 2019

[13] Kagermann, H., Lukas, W.D., Wahlster, W.: Industrie 4.0: Mit dem Internet der Dinge auf dem Weg zur 4. Industriellen Revolution. VDI Nachrichten 13(1), 2–3 (2011)

[14] Kagermann, H.: Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0. Tech. rep., acatech–National Academy of Science and Engineering, Frankfurt, DE (2013)

[15] ITU: Recommendation ITU-T Y.2060: Overview of the Internet of Things. Standard, International Telecommunication Union, Geneva, CH (2012)

[16] NSF: NSF 11-516: Cyber-Physical Systems (CPS). Tech. rep., National Science Foundation, Virginia, USA (2010)

[17] Wang, S., Wan, J., Zhang, D., Li, D., Zhang, C.: Towards Smart Factory for Industry 4.0: A Self-Organized Multi-Agent System with Big Data Based Feedback and Coordination. Computer Networks 101, 158–168 (2016)

[18] Hermann, M., Pentek, T., Otto, B.: Design Principles for Industrie 4.0 Scenarios. In: 49th Hawaii International Conference on System Sciences (HICSS), pp. 3928–3937. IEEE (2016)

[19] Lasi, H., Fettke, P., Kemper, H.G., Feld, T., Hoffmann, M.: Industry 4.0. Business & Information Systems Engineering 6(4), 239–242 (2014)

[20] Mueller, E., Chen, X.L., Riedel, R.: Challenges and Requirements for the Application of Industry 4.0: A Special Insight with the Usage of Cyber-Physical System. Chinese Journal of Mechanical Engineering 30(5), 1050 (2017)

[21] Rüßmann, M., Lorenz, M., Gerbert, P., Waldner, M., Justus, J., Engel, P., Harnisch, M.: Industry 4.0: The Future of Productivity and Growth in Manufacturing Industries. Tech. rep., Boston Consulting Group, Boston, USA (2015)

[22] Zhang, Y., Ren, S., Liu, Y., Si, S.: A Big Data Analytics Architecture for Cleaner Manufacturing and Maintenance Processes of Complex Products. Journal of Cleaner Production 142, 626–641 (2017)

[23] Wang, K.: Applying Data Mining to Manufacturing: The Nature and Implications. Journal of Intelligent Manufacturing 18(4), 487–495 (2007)

[24] Nagorny, K., Lima-Monteiro, P., Barata, J., Colombo, A.W.: Big Data Analysis in Smart Manufacturing: A Review. International Journal of Communications, Network and System Sciences 10(03), 31 (2017)

[25] Zhu, S., Kong, L., Chen, L.: Data Mining of Sensor Monitoring Time Series and Knowledge Discovery. In: 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), pp. 2914–2917. Zhengzhou, China (2011)

[26] Chand, S., Davis, J.: What is Smart Manufacturing. Time Magazine Wrapper 7, 28–33 (2010)

[27] Mikut, R.: Big Data & Automatisierung. at-Automatisierungstechnik 64(7), 503–506 (2016)

[28] Woo, J., Shin, S.J., Seo, W., Meilanitasari, P.: Developing a Big Data Analytics Platform for Manufacturing Systems: Architecture, Method, and Implementation. The International Journal of Advanced Manufacturing Technology 99(9-12), 2193–2217 (2018)

[29] Zhou, B., Pychynski, T., Reischl, M., Mikut, R.: Comparison of Machine Learning Approaches for Time-Series-Based Quality Monitoring of Resistance Spot Welding (RSW). Archives of Data Science, Series A (Online First) 5(1), 1–27 (2018)

[30] Bartschat, A., Reischl, M., Mikut, R.: Data Mining Tools. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 9(4), e1309 (2019)

[31] Zhou, B., Pychynski, T.: Internal Report: Process Description of Electric Resistance Welding. Tech. rep., Robert Bosch GmbH, Renningen, DE (2018)

[32] DIN: EN14610: 2005-02: Schweißen und verwandte Prozesse-Begriffe für Metallschweißprozesse. Standard, Beuth Verlag GmbH, Berlin, DE (2004)

[33] DIN: EN ISO 9001:2015: Qualitätsmanagementsysteme - Anforderungen. Standard, Beuth Verlag GmbH, Berlin, DE (2008)

[34] ISO: 14327:2004: Resistance Welding – Procedures for Determining the Weldability Lobe for Resistance Spot, Projection and Seam Welding. Standard, International Organization for Standardization, Geneva, CH (2004)

[35] DVS: 2902–3: Widerstandspunktschweißen von Stählen bis 3 mm Einzeldicke – Konstruktion und Berechnung. Standard, Deutscher Verband für Schweißen und verwandte Verfahren e. V., Düsseldorf, DE (2016)

[36] El Ouafi, A., Bélanger, R., Méthot, J.F.: An On-line ANN-Based Approach for Quality Estimation in Resistance Spot Welding. Advanced Materials Research 112, 141–148 (2010)

[37] Zhang, Y., Chen, G., Lin, Z.: Study on Weld Quality Control of Resistance Spot Welding Using a Neuro-Fuzzy Algorithm. In: International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, pp. 544–550. Springer, Berlin/Heidelberg, Wellington, New Zealand (2004)

[38] Cho, Y., Rhee, S.: New Technology for Measuring Dynamic Resistance and Estimating Strength in Resistance Spot Welding. Measurement Science and Technology 11(8), 1173 (2000)

[39] DIN: Begriffe der Qualitätssicherung und Statistik. Standard, Beuth Verlag GmbH, Berlin, DE (1989)

[40] Gamweger, J.: Design for Six Sigma: Kundenorientierte Produkte und Prozesse fehlerfrei entwickeln. Carl Hanser Verlag, Munich (2009)

[41] Rexroth: Operation Instructions: Weld Timer with Medium-Frequency Inverter. User Manual, Rexroth Bosch Group (2019). URL http://spotweldinc.com/PDFs/Bosch/1070080028_06.pdf. Accessed 18 December 2019

[42] Samuel, A.L.: Some Studies on Machine Learning Using the Game of Checkers. IBM Journal of Research and Development 3, 211–229 (1959)

[43] Mitchell, T.M.: The Discipline of Machine Learning, vol. 9. Carnegie Mellon University, School of Computer Science, Machine Learning Department, Pittsburgh (2006)

[44] Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: From Data Mining to Knowledge Discovery in Databases. AI Magazine 17(3), 37–37 (1996)

[45] Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Berlin/Heidelberg (2006)

[46] Alpaydin, E.: Introduction to Machine Learning. MIT Press, Massachusetts (2009)

[47] Samuel, A.L.: Some Studies in Machine Learning Using the Game of Checkers. IBM Journal of Research and Development 44(1.2), 206–226 (2000)

[48] Mikut, R.: Data Mining in der Medizin und Medizintechnik, vol. 22. KIT Scientific Publishing, Karlsruhe (2008)

[49] Frické, M.: The Knowledge Pyramid: A Critique of the DIKW Hierarchy. Journal of Information Science 35(2), 131–142 (2009)

[50] Mikut, R., Reischl, M., Burmeister, O., Loose, T.: Data Mining in Medical Time Series. Biomedizinische Technik 51(5/6), 288–293 (2006)

[51] LaCasse, P.M., Otieno, W., Maturana, F.P.: A Survey of Feature Set Reduction Approaches for Predictive Analytics Models in the Connected Manufacturing Enterprise. Applied Sciences 9(5), 843 (2019)

[52] Bengio, Y., Courville, A., Vincent, P.: Representation Learning: A Review and New Perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence 35(8), 1798–1828 (2013)

[53] Pyle, D.: Data Preparation for Data Mining. Morgan Kaufmann Publishers, San Francisco (1999)

[54] Ramírez-Gallego, S., Krawczyk, B., García, S., Woźniak, M., Herrera, F.: A Survey on Data Preprocessing for Data Stream Mining: Current Status and Future Directions. Neurocomputing 239, 39–57 (2017)

[55] Zöller, M.A., Huber, M.F.: Survey on Automated Machine Learning. ArXiv Preprint ArXiv:1904.12054 (2019)

[56] Junno, H., Laurinen, P., Tuovinen, L., RÃ¶ning, J.: Studying the Quality of Resistance Spot Welding Joints Using Self-Organising Maps. In: In: Fourth International ICSC Symposium on Engineering of Intelligent Systems (EIS), pp. 705–711. Madeira, Portugal (2004)

[57] Park, Y.J., Cho, H.: Quality Evaluation by Classification of Electrode Force Patterns in the Resistance Spot Welding Process Using Neural Networks. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 218(11), 1513–1524 (2004)

[58] Zhang, H., Hou, Y., Zhang, J., Qi, X., Wang, F.: A New Method for Nondestructive Quality Evaluation of the Resistance Spot Welding Based on the Radar Chart Method and the Decision Tree Classifier. International Journal of Advanced Manufacturing Technology 78(5-8), 841–851 (2015)

[59] Zhang, H., Wang, F., Xi, T., Zhao, J., Wang, L., Gao, W.: A Novel Quality Evaluation Method for Resistance Spot Welding Based on the Electrode Displacement Signal and the Chernoff Faces Technique. Mechanical Systems and Signal Processing 62-63, 431 – 443 (2015)

[60] Jain, R., Kasturi, R., Schunck, B.G.: Machine Vision, vol. 5. McGraw-Hill New York (1995)

[61] Hamilton, J.D.: Time Series Analysis, vol. 2. Princeton University Press, New Jersey (1994)

[62] Guo, L., Gao, H., Huang, H., He, X., Li, S.: Multifeatures Fusion and Nonlinear Dimension Reduction for Intelligent Bearing Condition Monitoring. Shock and Vibration 2016 (2016)

[63] Jolliffe, I.T., Cadima, J.: Principal Component Analysis: A Review and Recent Developments. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 374(2065), 20150202 (2016)

[64] Martínez, A.M.: PCA Versus LDA. IEEE Transactions on Pattern Analysis and Machine Intelligence 23(2) (2001)

[65] Altman, N.S.: An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. The American Statistician 46(3), 175–185 (1992)

[66] Deng, L., Yu, D., et al.: Deep Learning: Methods and Applications. Foundations and Trends in Signal Processing 7(3–4), 197–387 (2014)

[67] Schmidhuber, J.: Deep Learning in Neural Networks: An Overview. Neural Networks 61, 85–117 (2015)

[68] LeCun, Y., Bengio, Y., Hinton, G.: Deep Learning. Nature 521(7553), 436 (2015)

[69] Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Massachusetts (2016)

[70] Amaratunga, Thimira: How 'Deep' Should It Be to Be Called Deep Learning? Online (2017). URL https://towardsdatascience.com/how-deep-should-it-be-to-be-called-deep-learning-a7b1a6ab5610. Accessed 16 September 2019

[71] Nicholson, Chris: A Beginner's Guide to Neural Networks and Deep Learning. Online. URL https://skymind.ai/wiki/neural-network. Accessed 16 September 2019

[72] Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Kingsbury, B., et al.: Deep Neural Networks for Acoustic Modeling in Speech Recognition. IEEE Signal Processing Magazine 29(6), 82–97 (2012)

[73] LeCun, Y., Haffner, P., Bottou, L., Bengio, Y.: Object Recognition with Gradient-Based Learning. In: Shape, Contour and Grouping in Computer Vision, pp. 319–345. Springer, Berlin/Heidelberg (1999)

[74] Lipton, Z.C., Berkowitz, J., Elkan, C.: A Critical Review of Recurrent Neural Networks for Sequence Learning. ArXiv Preprint ArXiv:1506.00019 (2015)

[75] Schuster, M., Paliwal, K.K.: Bidirectional Recurrent Neural Networks. IEEE Transactions on Signal Processing 45(11), 2673–2681 (1997)

[76] Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Computation 9(8), 1735–1780 (1997)

[77] Hitzler, P., et al.: OWL 2 Web Ontology Language Primer. W3C Recommendation 27(1), 123 (2009)

[78] Baader, F., Calvanese, D., McGuinness, D., et al.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, Cambridge (2003)

[79] Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C., et al.: OWL 2 Web Ontology Language Profiles. W3C Recommendation 27, 61 (2009)

[80] W3C, et al.: OWL 2 Web Ontology Language Document Overview (2012). URL https://www.w3.org/TR/owl2-overview/. Accessed 27 July 2020

[81] McGuinness, D.L., Van Harmelen, F., et al.: OWL Web Ontology Language Overview. W3C Recommendation 10(10), 2004 (2004)

[82] Polikoff, I., Allemang, D.: Semantic Technology. URL https://lists.oasis-open.org/archives/regrep-semantic/200402/pdf00000.pdf. Accessed 7 January 2021

[83] Kharlamov, E., Grau, B.C., Jiménez-Ruiz, E., Lamparter, S., Mehdi, G., Ringsquandl, M., Nenov, Y., Grimm, S., Roshchin, M., Horrocks, I.: Capturing Industrial Information Models With Ontologies and Constraints. In: Proceedings of the 15th International Semantic Web Conference, pp. 325–343. Springer, Berlin/Heidelberg, Kobe, Japan (2016)

[84] Kharlamov, E., Kotidis, Y., Mailis, T., Neuenstadt, C., Nikolaou, C., Özçep, Ö.L., Svingos, C., Zheleznyakov, D., Ioannidis, Y.E., Lamparter, S., Möller, R., Waaler, A.: An Ontology-Mediated Analytics-Aware Approach to Support Monitoring and Diagnostics of Static and Streaming Data. Journal of Web Semantics 56, 30–55 (2019)

[85] Kharlamov, E., Hovland, D., Skjæveland, M.G., Bilidas, D., Jiménez-Ruiz, E., Xiao, G., Soylu, A., Lanti, D., Rezk, M., Zheleznyakov, D., Giese, M., Lie, H., Ioannidis, Y.E., Kotidis, Y., Koubarakis, M., Waaler, A.: Ontology Based Data Access in Statoil. Journal of Web Semantics 44, 3–36 (2017)

[86] Kharlamov, E., Mailis, T., Mehdi, G., Neuenstadt, C., Özçep, Ö.L., Roshchin, M., Solomakhina, N., Soylu, A., Svingos, C., Brandt, S., Giese, M., Ioannidis,

Y.E., Lamparter, S., Möller, R., Kotidis, Y., Waaler, A.: Semantic Access to Streaming and Static Data at Siemens. Journal of Web Semantics 44, 54–74 (2017)

[87] Horrocks, I., Giese, M., Kharlamov, E., Waaler, A.: Using Semantic Technology to Tame the Data Variety Challenge. IEEE Internet Computing 20(6), 62–66 (2016)

[88] Kalaycı, E.G., González, I.G., Lösch, F., Xiao, G., ul Mehdi, A., Kharlamov, E., Calvanese, D.: Semantic Integration of Bosch Manufacturing Data Using Virtual Knowledge Graphs. In: Proceedings of the 19th International Semantic Web Conference, pp. 464–481. Springer, Berlin/Heidelberg, Athens, Greece (2020)

[89] Soylu, A., Kharlamov, E., Zheleznyakov, D., Jiménez-Ruiz, E., Giese, M., Skjæveland, M.G., Hovland, D., Schlatte, R., Brandt, S., Lie, H., Horrocks, I.: Optiquevqs: A Visual Query System Over Ontologies for Industry. Semantic Web 9(5), 627–660 (2018)

[90] Ringsquandl, M., Kharlamov, E., Stepanova, D., Hildebrandt, M., Lamparter, S., Lepratti, R., Horrocks, I., Kröger, P.: Event-Enhanced Learning for KG Completion. In: Proceedings of the 15th Extended Semantic Web Conference, pp. 541–559. Springer (2018)

[91] Kharlamov, E., Mehdi, G., Savkovic, O., Xiao, G., Kalayci, E.G., Roshchin, M.: Semantically-Enhanced Rule-Based Diagnostics for Industrial Internet of Things: The SDRL Language and Case Study for Siemens Trains and Turbines. Journal of Web Semantics 56, 11–29 (2019)

[92] Obitko, M.: Translations Between Ontologies in Multi-Agent Systems. Ph.D. thesis, Faculty of Electrical Engineering, Czech Technical University in Prague (2007)

[93] Skjæveland, M.G., Lupp, D.P., et al.: Practical Ontology Pattern Instantiation, Discovery, and Maintenance With Reasonable Ontology Templates. In: Proceedings of the 17th International Semantic Web Conference, pp. 477–494. Springer, Monterey, USA (2018)

[94] Wuest, T., Weimer, D., Irgens, C., Thoben, K.D.: Machine Learning in Manufacturing: Advantages, Challenges, and Applications. Production & Manufacturing Research 4(1), 23–45 (2016)

[95] Tao, F., Cheng, J., Qi, Q., Zhang, M., Zhang, H., Sui, F.: Digital Twin-Driven Product Design, Manufacturing and Service with Big Data. The International Journal of Advanced Manufacturing Technology 94(9), 3563–3576 (2018)

[96] Fang, H.: Managing Data Lakes in Big Data Era: What's a Data Lake and Why Has It Became Popular in Data Management Ecosystem. In: 2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), pp. 820–824. IEEE, Shenyang, China (2015)

[97] Hernandez-Matias, J., Vizan, A., Perez-Garcia, J., Rios, J.: An Integrated Modelling Framework to Support Manufacturing System Diagnosis for Continuous Improvement. Robotics and Computer-Integrated Manufacturing 24(2), 187–199 (2008)

[98] DIN: EN 13306:2017: Maintenance-Maintenance Terminology. Standard, Beuth Verlag GmbH, Berlin, DE (2018)

[99] Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., Gao, R.X.: Deep Learning and Its Applications to Machine Health Monitoring. Mechanical Systems and Signal Processing 115, 213–237 (2019)

[100] Kusiak, A., Verma, A.: A Data-Mining Approach to Monitoring Wind Turbines. IEEE Transactions on Sustainable Energy 3(1), 150–157 (2012)

[101] Potnuru, S.P., Bhima, P.R.: Image Processing and Machine Learning Applied for Condition Monitoring of 11-kV Power Distribution Line Insulators Using Curvelet and LTP Features. In: 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), pp. 3012–3017. Chennai, India (2017)

[102] Wuest, T., Irgens, C., Thoben, K.D.: An Approach to Monitoring Quality in Manufacturing Using Supervised Machine Learning on Product State Data. Journal of Intelligent Manufacturing 25(5), 1167–1180 (2014)

[103] Ribeiro, B.: Support Vector Machines for Quality Monitoring in a Plastic Injection Molding Process. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 35(3), 401–410 (2005)

[104] Wang, J., Xie, J., Zhao, R., Zhang, L., Duan, L.: Multisensory Fusion Based Virtual Tool Wear Sensing for Ubiquitous Manufacturing. Robotics and Computer-Integrated Manufacturing 45, 47–58 (2017)

[105] Wang, D.: K-Nearest Neighbors Based Methods for Identification of Different Gear Crack Levels Under Different Motor Speeds and Loads: Revisited. Mechanical Systems and Signal Processing 70, 201–208 (2016)

[106] Widodo, A., Yang, B.S.: Support Vector Machine in Machine Condition Monitoring and Fault Diagnosis. Mechanical Systems and Signal Processing 21(6), 2560–2574 (2007)

[107] Baraldi, P., Podofillini, L., Mkrtchyan, L., Zio, E., Dang, V.N.: Comparing the Treatment of Uncertainty in Bayesian Networks and Fuzzy Expert Systems Used for a Human Reliability Analysis Application. Reliability Engineering & System Safety 138, 176–193 (2015)

[108] Jia, F., Lei, Y., Lin, J., Zhou, X., Lu, N.: Deep Neural Networks: A Promising Tool for Fault Characteristic Mining and Intelligent Diagnosis of Rotating Machinery with Massive Data. Mechanical Systems and Signal Processing 72, 303–315 (2016)

[109] Sun, J., Yan, C., Wen, J.: Intelligent Bearing Fault Diagnosis Method Combining Compressed Data Acquisition and Deep Learning. IEEE Transactions on Instrumentation and Measurement 67(1), 185–195 (2018)

[110] Acherjee, B., Mondal, S., Tudu, B., Misra, D.: Application of Artificial Neural Network for Predicting Weld Quality in Laser Transmission Welding of Thermoplastics. Applied Soft Computing 11(2), 2548–2555 (2011)

[111] Huang, W., Kovacevic, R.: A Neural Network and Multiple Regression Method for the Characterization of the Depth of Weld Penetration in Laser Welding Based on Acoustic Signatures. Journal of Intelligent Manufacturing 22(2), 131–143 (2011)

[112] Sumesh, A., Rameshkumar, K., Mohandas, K., Babu, R.S.: Use of Machine Learning Algorithms for Weld Quality Monitoring Using Acoustic Signature. Procedia Computer Science 50, 316–322 (2015)

[113] Aktepe, A., Öncel, Ç., Ersöz, S.: An Artificial Neural Network Model on Welding Process Control of 155 mm Artillery Ammunition. In: 6th International Advanced Technologies Symposium (IATS'11), pp. 16–18. Elazıg, Turkey (2011)

[114] Bhattacharya, S., Pal, K., Pal, S.K.: Multi-Sensor Based Prediction of Metal Deposition in Pulsed Gas Metal Arc Welding Using Various Soft Computing Models. Applied Soft Computing 12(1), 498–505 (2012)

[115] Aviles-Viñas, J.F., Rios-Cabrera, R., Lopez-Juarez, I.: On-line Learning of Welding Bead Geometry in Industrial Robots. The International Journal of Advanced Manufacturing Technology 83(1), 217–231 (2016)

[116] Yao, P., Xue, J., Zhou, K.: Study on the Wire Feed Speed Prediction of Double-Wire-Pulsed MIG Welding Based on Support Vector Machine Regression. The International Journal of Advanced Manufacturing Technology 79(9), 2107–2116 (2015)

[117] Romero-Hdz, J., Saha, B., Toledo-Ramirez, G., Lopez-Juarez, I.: A Reinforcement Learning Based Approach for Welding Sequence Optimization. In: Transactions on Intelligent Welding Manufacturing, pp. 33–45. Springer, Singapore, Singapore (2018)

[118] Vedrtnam, A., Singh, G., Kumar, A.: Optimizing Submerged Arc Welding Using Response Surface Methodology, Regression Analysis, and Genetic Algorithm. Defence Technology 14(3), 204–212 (2018)

[119] Dhas, J.E.R., Dhas, S.J.H.: A Review on Optimization of Welding Process. Procedia Engineering 38, 544–554 (2012)

[120] Fleming, P., Lammlein, D., Wilkes, D., Fleming, K., Bloodworth, T., Cook, G., Strauss, A., DeLapp, D., Lienert, T., Bement, M., et al.: In-Process Gap Detection in Friction Stir Welding. Sensor Review 28(1), 62–67 (2008)

[121] Hamedi, M., Shariatpanahi, M., Mansourzadeh, A.: Optimizing Spot Welding Parameters in a Sheet Metal Assembly by Neural Networks and Genetic Algorithm. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 221(7), 1175–1184 (2007)

[122] Martín, T., De Tiedra, P., López, M.: Artificial Neural Networks for Pitting Potential Prediction of Resistance Spot Welding Joints of AISI 304 Austenitic Stainless Steel. Corrosion Science 52(7), 2397–2402 (2010)

[123] Kim, K.Y., Ahmed, F.: Semantic Weldability Prediction with RSW Quality Dataset and Knowledge Construction. Advanced Engineering Informatics 38, 41–53 (2018)

[124] Li, W., Hu, S.J., Ni, J.: On-line Quality Estimation in Resistance Spot Welding. Journal of Manufacturing Science and Engineering 122(3), 511–512 (2000)

[125] Lee, S.R., Choo, Y.J., Lee, T.Y., Kim, M.H., Choi, S.K.: A Quality Assurance Technique for Resistance Spot Welding Using a Neuro-Fuzzy Algorithm. Journal of Manufacturing Systems 20(5), 320–328 (2001)

[126] Cho, Y., Rhee, S.: Primary Circuit Dynamic Resistance Monitoring and Its Application to Quality Estimation During Resistance Spot Welding. Welding Journal 81(6), 104–111 (2002)

[127] Lee, H.T., Wang, M., Maev, R., Maeva, E.: A Study on Using Scanning Acoustic Microscopy and Neural Network Techniques to Evaluate the Quality of Resistance Spot Welding. International Journal of Advanced Manufacturing Technology 22(9-10), 727–732 (2003)

[128] Cho, Y., Rhee, S.: Quality Estimation of Resistance Spot Welding by Using Pattern Recognition with Neural Networks. IEEE Transactions on Instrumentation and Measurement 53(2), 330–334 (2004)

[129] Laurinen, P., Junno, H., Tuovinen, L., Röning, J.: Studying the Quality of Resistance Spot Welding Joints Using Bayesian Networks. In: Proceedings of the Artificial Intelligence and Applications, pp. 705–711. Toulouse, France (2004)

[130] Podržaj, P., Polajnar, I., Diaci, J., Kariž, Z.: Expulsion Detection System for Resistance Spot Welding Based on a Neural Network. Measurement Science and Technology 15(3), 592 (2004)

[131] Haapalainen, E., Laurinen, P., Junno, H., Tuovinen, L., Röning, J.: Methods for Classifying Spot Welding Processes: A Comparative Study of Performance. In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, pp. 412–421. Springer, Berlin/Heidelberg, Bari, Italy (2005)

[132] Tseng, H.Y.: Welding Parameters Optimization for Economic Design Using Neural Approximation and Genetic Algorithm. International Journal of Advanced Manufacturing Technology 27(9-10), 897–901 (2006)

[133] Koskimaki, H.J., Laurinen, P., Haapalainen, E., Tuovinen, L., Roning, J.: Application of the Extended *k* nn Method to Resistance Spot Welding Process Identification and the Benefits of Process Information. IEEE Transactions on Industrial Electronics 54(5), 2823–2830 (2007)

[134] Martín, Ó., López, M., Martin, F.: Artificial Neural Networks for Quality Control by Ultrasonic Testing in Resistance Spot Welding. Journal of Materials Processing Technology 183(2-3), 226–233 (2007)

[135] El-Banna, M., Filev, D., Chinnam, R.B.: Online Qualitative Nugget Classification by Using a Linear Vector Quantization Neural Network for Resistance Spot Welding. The International Journal of Advanced Manufacturing Technology 36(3-4), 237–248 (2008)

[136] Haapalainen, E., Laurinen, P., Junno, H., Tuovinen, L., Röning, J.: Feature Selection for Identification of Spot Welding Processes. In: Informatics in Control Automation and Robotics, pp. 69–79. Springer, Berlin/Heidelberg (2008)

[137] Martín, O., Tiedra, P.D., López, M., San-Juan, M., García, C., Martín, F., Blanco, Y.: Quality Prediction of Resistance Spot Welding Joints of 304 Austenitic Stainless Steel. Materials and Design 30(1), 68–77 (2009)

[138] Li, Y., Zhao, W., Xue, H., Ding, J.: Defect Recognition of Resistance Spot Welding Based on Artificial Neural Network. Advances in Intelligent and Soft Computing 115(2), 423–430 (2012)

[139] Panchakshari, A.S., Kadam, M.S.: Optimization of the Process Parameters in Resistance Spot Welding Using Genetic Algorithm. International Journal of Multidisciplinary Science and Engineering 4(3), 1438–1442 (2013)

[140] Afshari, D., Sedighi, M., Karimi, M.R., Barsoum, Z.: Prediction of the Nugget Size in Resistance Spot Welding with a Combination of a Finite-Element Analysis and an Artificial Neural Network. Materiali in Tehnologije 48(1), 33–38 (2014)

[141] Yu, J.: Quality Estimation of Resistance Spot Weld Based on Logistic Regression Analysis of Welding Power Signal. International Journal of Precision Engineering and Manufacturing 16(13), 2655–2663 (2015)

[142] Boersch, I., Füssel, U., Gresch, C., Großmann, C., Hoffmann, B.: Data Mining in Resistance Spot Welding. The International Journal of Advanced Manufacturing Technology 99(5-8), 1085–1099 (2018)

[143] Pashazadeh, H., Gheisari, Y., Hamedi, M.: Statistical Modeling and Optimization of Resistance Spot Welding Process Parameters Using Neural Networks and Multi-Objective Genetic Algorithm. Journal of Intelligent Manufacturing 27(3), 549–559 (2016)

[144] Wan, X., Wang, Y., Zhao, D.: Quality Monitoring Based on Dynamic Resistance and Principal Component Analysis in Small Scale Resistance Spot Welding Process. The International Journal of Advanced Manufacturing Technology 86(9-12), 3443–3451 (2016)

[145] Summerville, C., Adams, D., Compston, P., Doolan, M.: Nugget Diameter in Resistance Spot Welding: A Comparison Between a Dynamic Resistance Based Approach and Ultrasound C-Scan. Procedia Engineering 183, 257–263 (2017)

[146] Summerville, C.D.E., Adams, D., Compston, P., Doolan, M.: Process Monitoring of Resistance Spot Welding Using the Dynamic Resistance Signature. Welding Journal 11, 403–412 (2017)

[147] Sun, H., Yang, J., Wang, L.: Resistance Spot Welding Quality Identification with Particle Swarm Optimization and a Kernel Extreme Learning Machine Model. The International Journal of Advanced Manufacturing Technology 91(5-8), 1879–1887 (2017)

[148] Zhang, H., Hou, Y., Zhao, J., Wang, L., Xi, T., Li, Y.: Automatic Welding Quality Classification for the Spot Welding Based on the Hopfield Associative Memory Neural Network and Chernoff Face Description of the Electrode Displacement Signal Features. Mechanical Systems and Signal Processing 85, 1035 – 1043 (2017)

[149] Witkin, A.P.: Scale-Space Filtering. In: Readings in Computer Vision, pp. 329–332. Morgan Kaufmann, San Francisco, USA (1987)

[150] Holmes, G., Hall, M., Prank, E.: Generating Rule Sets From Model Trees. In: Australasian Joint Conference on Artificial Intelligence, pp. 1–12. Sydney, Australia (1999)

[151] BokTech-Team: Quality Control Process (2020). URL https://www.boktech.cc/learn/detail/8.html. Accessed 29 October 2020

[152] Prístavka, M., Kotorová, M., Savov, R.: Quality Control in Production Processes. Acta Technologica Agriculturae 19(3), 77–83 (2016)

[153] Rahani, A., Al-Ashraf, M.: Production Flow Analysis Through Value Stream Mapping: A Lean Manufacturing Process Case Study. Procedia Engineering 41, 1727–1734 (2012)

[154] Franceschini, F., Galetto, M., Maisano, D.: Classification of Performance and Quality Indicators in Manufacturing. International Journal of Services and Operations Management 2(3), 294–311 (2006)

[155] Effendi, M., Razali, M., Rohana, A., Adi, S.: A Study on the Development of Key Performance Indicators (Kpis) at an Aerospace Manufacturing Company. The International Journal of Advanced Manufacturing Technology 2, 1–17 (2008)

[156] Roriz, C., Nunes, E., Sousa, S.: Application of Lean Production Principles and Tools for Quality Improvement of Production Processes in a Carton Company. Procedia Manufacturing 11, 1069–1076 (2017)

[157] Schnell, J., Reinhart, G.: Quality Management for Battery Production: A Quality Gate Concept. Procedia CIRP 57, 568–573 (2016)

[158] Schif, A.: Examination of Machine Learning Methods for the Evaluation of Resistance Spot Welds Based on Simulated Process Scenarios. Master's thesis, University of Stuttgart (2017)

[159] Gonzalez-Abril, L., Gavilan, J.M., Velasco Morente, F.: Three Similarity Measures between One-Dimensional DataSets. Revista Colombiana de Estadística 37(1), 79–94 (2014)

[160] D. Chikina, M., G. Troyanskaya, O.: An Effective Statistical Evaluation of Chipseq Dataset Similarity. Bioinformatics 28(5), 607–613 (2012)

[161] Parthasarathy, S., Ogihara, M.: Exploiting Dataset Similarity for Distributed Mining. In: International Parallel and Distributed Processing Symposium, pp. 399–406. Springer, Berlin/Heidelberg, Cancun, Mexico (2000)

[162] Sequeira, K., Zaki, M.J.: Exploring Similarities Across High-Dimensional Datasets. In: Research and Trends in Data Mining Technologies and Applications, pp. 53–84. IGI Global, Pennsylvania (2007)

[163] Zhou, B., Chioua, M., Bauer, M., Schlake, J.C., Thornhill, N.F.: Improving Root Cause Analysis by Detecting and Removing Transient Changes in Oscillatory Time Series with Application to a 1, 3-Butadiene Process. Industrial & Engineering Chemistry Research 58(26), 11234–11250 (2019)

[164] Kühnert, C., Gröll, L., Heizmann, M., Reischl, M., Mikut, R.: Methoden zur Datengetriebenen Formulierung und Visualisierung von Kausalitätshypothesen. at-Automatisierungstechnik Methoden und Anwendungen der Steuerungs-, Regelungs-und Informationstechnik 60(10), 630–640 (2012)

[165] Pearson, K.: Notes on Regression and Inheritance in the Case of Two Parents. In: Proceedings of the Royal Society of London, vol. 58, pp. 240–242. London, UK (1895)

[166] Schreiber, T.: Measuring Information Transfer. Physical Review Letters 85(2), 461 (2000)

[167] González Ordiano, J.Á., Waczowicz, S., Hagenmeyer, V., Mikut, R.: Energy Forecasting Tools and Services. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8(2), e1235 (2018)

[168] Tukey, J.W.: Exploratory Data Analysis: Past, Present and Future. Princeton University Press Princeton, Department of Statistics, New Jersey (1993)

[169] Svetashova, Y., Zhou, B., Pychynski, T., Schmidt, S., Sure-Vetter, Y., Mikut, R., Kharlamov, E.: Ontology-Enhanced Machine Learning Pipeline: A Bosch Use Case of Welding Quality Monitoring. In: Proceedings of the 19th International Semantic Web Conference, pp. 531–550. Springer, Berlin/Heidelberg, Athens, Greece (2020)

[170] Zhou, B., Svetashova, Y., Byeon, S., Pychynski, T., Mikut, R., Kharlamov, E.: Predicting Quality of Automated Welding with Machine Learning and Semantics: a Bosch Case Study. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 2933–2940. Association for Computing Machinery, Galway, Ireland (2020)

[171] Zhou, B., Svetashova, Y., Pychynski, T., Kharlamov, E.: SemFE: Facilitating ML Pipeline Development with Semantics. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 3489–3492. Association for Computing Machinery, Galway, Ireland (2020)

[172] Svetashova, Y., Zhou, B., Pychynski, T., Kharlamov, E.: SemML: Reusable ML for Condition Monitoring in Discrete Manufacturing. In: Proceedings of the 19th International Semantic Web Conference, vol. 2721, p. 214. Athens, Greece (2020)

[173] Svetashova, Y., Zhou, B., Pychynski, T., Kharlamov, E.: Semantic ML for Manufacturing Monitoring at Bosch. In: Proceedings of the 19th International Semantic Web Conference. Springer, Berlin/Heidelberg, Athens, Greece (2020)

[174] Ocampo-Martinez, C., et al.: Energy Efficiency in Discrete-Manufacturing Systems: Insights, Trends, and Control Strategies. Journal of Manufacturing Systems 52, 131–145 (2019)

[175] Ross, D.: Plato's Theory of Ideas. Cambridge University Press, Cambridge (1953)

[176] Codd, E.F.: A Relational Model of Data for Large Shared Data Banks. In: Software Pioneers, pp. 263–294. Springer, Berlin/Heidelberg (2002)

[177] ECMA, E.: 404: The JSON Data Interchange Format. ECMA (European Association for Standardizing Information and Communication Systems), Geneva, Switzerland (2013)

[178] W3C, et al.: Extensible Markup Language (XML) 1.0 (Fifth Edition) (2008). URL https://www.w3.org/TR/xml/. Accessed 29 July 2020

[179] Kirch, W.: Level of Measurement. Encyclopedia of Public Health 2, 851–852 (2008)

[180] Chatfield, C., Xing, H.: The Analysis of Time Series: An Introduction With R. CRC press, Florida (2019)

[181] Barnett, T., Preisendorfer, R.: Origins and Levels of Monthly and Seasonal Forecast Skill for United States Surface Air Temperatures Determined by Canonical Correlation Analysis. Monthly Weather Review 115(9), 1825–1850 (1987)

[182] Brownlee, J.: Deep Learning With Python: Develop Deep Learning Models on Theano and Tensorflow Using Keras. Machine Learning Mastery, Vermont Victoria (2016)

[183] Freedman, D., Pisani, R., Purves, R.: Statistics. New York: WW Norton (1998)

[184] Schaffer, J.: What Not to Multiply Without Necessity. Australasian Journal of Philosophy 93(4), 644–664 (2015)

[185] Hyndman, R.J., Koehler, A.B.: Another Look At Measures of Forecast Accuracy. International Journal of Forecasting 22(4), 679–688 (2006)

[186] Muhammad, N., Manurung, Y.H.P.: Design Parameters Selection and Optimization of Weld Zone Development in Resistance Spot Welding. Proceedings of World Academy of Science, Engineering and Technology 2012(71), 1220 (2012)

[187] Rossum, G.: Python Reference Manual. Tech. rep., Centre for Mathematics and Computer Science, Amsterdam, The Netherlands (1995)

[188] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-Learn: Machine Learning in Python. Journal of Machine Learning Research 12, 2825–2830 (2011)

[189] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. arXiv preprint arXiv:1603.04467 (2016)

[190] Chollet, F., et al.: Keras. https://keras.io (2015). Accessed 7 January 2021

[191] Mikut, R., Bartschat, A., Doneit, W., Ordiano, J.Á.G., Schott, B., Stegmaier, J., Waczowicz, S., Reischl, M.: The MATLAB Toolbox SciXMiner: User's Manual and Programmer's Guide. ArXiv:1704.03298 (2017)

[192] The Mathworks, Inc., Natick, Massachusetts: MATLAB Version 9.5.0.944444 (R2018b) (2018)

[193] RDF 1.1 Turtle - Terse RDF Triple Language (2014). URL https://www.w3.org/TR/turtle/. Accessed 27 July 2020

[194] Wong, P.C., Thomas, J.: Visual analytics. IEEE Computer Graphics and Applications 24(5), 20–21 (2004)

[195] Feller, D.J., Burgermaster, M., Levine, M.E., Smaldone, A., Davidson, P.G., Albers, D.J., Mamykina, L.: A Visual Analytics Approach for Pattern-Recognition in Patient-Generated Data. Journal of the American Medical Informatics Association 25(10), 1366–1374 (2018)

[196] Canossa, A., Nguyen, T.H.D., El-Nasr, M.S.: G-Player: Exploratory Visual Analytics for Accessible Knowledge Discovery. In: Proceedings of the First International Joint Conference of DiGRA/FDG, pp. 1–16. Digital Games Research Association, Dundee, Scotland (2016)

[197] Hunter, J.D.: Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering 9(3), 90–95 (2007)

[198] Han, Y., et al.: Production Capacity Analysis and Energy Saving of Complex Chemical Processes Using LSTM Based on Attention Mechanism. Applied Thermal Engineering 160, 114072 (2019)

[199] Ordiano, J.Á.G., Gröll, L., Mikut, R., Hagenmeyer, V.: Probabilistic Energy Forecasting Using the Nearest Neighbors Quantile Filter and Quantile Regression. International Journal of Forecasting 36(2), 310–323 (2020)

[200] Mikhaylov, D., Zhou, B., Kiedrowski, T., Mikut, R., Lasagni, A.F.: ML Aided Phase Retrieval Algorithm for Beam Splitting With an LCoS-SLM. In: Laser Resonators, Microresonators, and Beam Control XXI, vol. 10904, p. 109041M. International Society for Optics and Photonics, San Francisco, USA (2019)

[201] Mikhaylov, D., Zhou, B., Kiedrowski, T., Mikut, R., Lasagni, A.F.: High Accuracy Beam Splitting Using SLM Combined With ML Algorithms. Optics and Lasers in Engineering 121, 227–235 (2019)

[202] Bewersdorf, J., Pick, R., Hell, S.W.: Multifocal Multiphoton Microscopy. Optics Letters 23(9), 655–657 (1998)

[203] Shao, Y., Qin, W., Liu, H., Qu, J., Peng, X., Niu, H., Gao, B.: Multifocal Multiphoton Microscopy Based on a Spatial Light Modulator. Applied Physics B 107(3), 653–657 (2012)

[204] Coelho, S., Poland, S., Krstajic, N., Li, D., Monypenny, J., Walker, R., Tyndall, D., Ng, T., Henderson, R., Ameer-Beg, S.: Multifocal Multiphoton Microscopy with Adaptive Optical Correction. In: Multiphoton Microscopy in the Biomedical Sciences XIII, vol. 8588, p. 858817. International Society for Optics and Photonics, Bellingham, USA (2013)

[205] Obata, K., Koch, J., Hinze, U., Chichkov, B.N.: Multi-Focus Two-Photon Polymerization Technique Based on Individually Controlled Phase Modulation. Optics Express 18(16), 17193–17200 (2010)

[206] Romero, L.A., Dickey, F.M.: Theory of Optimal Beam Splitting by Phase Gratings. I. One-Dimensional Gratings. JOSA A 24(8), 2280–2295 (2007)

[207] Gerchberg, R.W.: A Practical Algorithm for the Determination of Phase from Image and Diffraction Plane Pictures. Optik 35, 237–246 (1972)

[208] Fienup, J.: Iterative Method Applied to Image Reconstruction and to Computer-Generated Holograms. Optical Engineering 19(3), 193297 (1980)

[209] Matsumoto, N., Okazaki, S., Fukushi, Y., Takamoto, H., Inoue, T., Terakawa, S.: An Adaptive Approach for Uniform Scanning in Multifocal Multiphoton Microscopy with a Spatial Light Modulator. Optics Express 22(1), 633–645 (2014)

[210] Mühl, F., Damon, J., Dietrich, S., Schulze, V.: Simulation of Induction Hardening: Simulative Sensitivity Analysis With Respect to Material Parameters and the Surface Layer State. Computational Materials Science 184, 109916 (2020)

[211] Zanger, F., Boev, N., Schulze, V.: Surface Quality After Broaching With Variable Cutting Thickness. Procedia CIRP 13, 114–119 (2014)

[212] Dehen, S., Segebade, E., Gerstenmeyer, M., Zanger, F., Schulze, V.: Milling Parameter and Tool Wear Dependent Surface Quality in Micro-Milling of Brass. Procedia CIRP 87, 95–100 (2020)

[213] Pearson, E.S., Gosset, W.S., Plackett, R., Barnard, G.A.: Student: A Statistical Biography of William Sealy Gosset. Oxford University Press, USA (1990)

[214] Hartung, J., Knapp, G., Sinha, B.K.: Statistical Meta-Analysis with Applications, vol. 738. John Wiley & Sons, New Jersey (2011)