FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Neural Computation Methods for Industrial Data Processing

PhD THESIS

submitted in partial fulfillment of the requirements for the degree of

Doctor of Technical Sciences

within the

Vienna PhD School of Informatics

by

M.Sc. Guodong Wang
Registration Number 01429973

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing. Dr.rer.nat. Radu Grosu

Second advisor: Asst.-Prof. Dr. Ezio Bartocci

External reviewers:
Tenured Assc.-Prof. Dr. Qi Zhu. Northwestern University, USA.
Asst.-Prof. Dr. Pedro M. Ferreira. University of Lisbon, Portugal.

Vienna, 10th December, 2018

Guodong Wang

Declaration of Authorship

M.Sc. Guodong Wang
Treitlstrasse 3, 1040 Vienna, Austria

I hereby declare that I have written this Doctoral Thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work - including tables, maps and figures - which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.

Vienna, 10th December, 2018

Guodong Wang

Acknowledgements

It has been a long journey since I started my PhD program at Vienna University of Technology, almost four years ago. A lot of happiness and sadness happened to me during this period. I really appreciate the great help from my PhD supervisor Prof. Dr. Radu Grosu for taking me as his student and giving me a lot of chances to be part of EU and Austrian projects and get in touch with many advanced EU companies. I learn a lot from these cooperation, and I believe that without these activities, I could not be the person with so much passion on scientific research and engineering management.

I also would like to thank my colleague Ramin Hasani for all his great help during this period. Ramin is an excellent artificial intelligence researcher who has been published many interesting works on the top international conferences and journals, e.g., IJCNN, NIPS, ICML and so on. We were always discussing some interesting ideas and topics together, even during some weekends. Among these thoughtful discussions, I learn a lot about how to do research and how to improve the quality of my research works.

Mrs. Gerda Belkhofer-Fohrafellner is the secretary of our research group. She is such a kind person who is always trying to help the people around her. I still remember that tough time when I came to Vienna for the first time. I barely knew any people here, I did not know German, and I was alone. Gerda encouraged me and helped me to meet some new friends.

I would like to thank my colleagues Denise and Haris. Without their help, I could not finish my study so smoothly.

I also would like to thank my hosting supervisor Dr. Alexandra Brintrup from the department of engineering, University of Cambridge. Alexandra supervised me as a visiting PhD student at Cambridge. She taught me a lot about how to do research rigorously. I think I will never forget that precious time when I was studying at Cambridge.

I would like to thank my parents, Kaiying Zhang and Xiubing Wang, for raising me as a person who is always full of passion and dream. My parents sacrificed everything they have just for getting me a chance to go to college.

I would like to thank my wife Yujuan and my lovely daughter Xiaobu. They accompanied with me and went through so many tough days. They are the source of my brave fighting with life and never be afraid of anything.

This thesis got financial support from the Doctoral College: Cyber-Physical Production Systems Project, funded by Vienna University of Technology, and the Productive 4.0 project, funded by EU-ECSEL. I would like to thank all the partners from these EU companies, without their help, I could not finish the thesis on time.

致谢

在博士毕业的前夕向远去的时光和充满希望的未来致敬。感谢我的博士导师 Radu Grosu教授让我有机会参与或者负责数个欧盟大型研究项目，我从中学习到了很多。Ramin Hasani是我在维也纳学习工作期间的同事与好友，我们一起完成了很多有趣的研究项目。Gerda Belkhofer Fohrafellner女士是我们Cyber-Physical Systems课题组的秘书。无论我遇到什么样的困难，她总是竭尽全力帮助我。Gerda 曾经在中国生活和工作过4年，她对中国的一切充满了怀恋，希望有一天她能重新回到那里。Ezio Bartocci教授是我的博士副导师，他是一位非常不错的指导老师与好朋友。我要特别感谢庄严又不失可爱的维也纳技术大学，它充满人文主义的情怀让我面对一个又一个艰深晦涩的工程技术难题时总是以一种积极乐观的态度面对。在整个博士学习工作期间，我的研究工作受到维也纳技术大学 Cyber-Physical Production System 项目以及 EU-ECSEL Productive 4.0项目的资助。同时，我在英国剑桥大学访问学习期间，维也纳技术大学也为我提供了国际研究合作资金（International Research Fellowship）。

感谢父母的养育之恩，感谢我的爱人与女儿的陪伴，人生旅途一路上有了他们，我才学会了坦然面对生活。

谨以此纪念那些时光 ...
愿世界和平、繁荣 ...

王国栋, 一八年秋
维也纳，奥地利共和国

Abstract

Modern manufacturing systems produce sensory data every day and methods for efficiently uncovering useful information (quickly, accurately, and with little or no human effort) within this raw data are crucial. A very promising solution in this respect is end-to-end data processing. It allows to capture complex tasks through a single complete model. Recent progress within the deep-learning community has shed light on the implementation of end-to-end data-processing techniques in manufacturing. These techniques take advantage of the learning ability of Artificial Neural Networks (ANNs) to extract essential system features from the inputs. Based on this information, a simple model is constructed to complete modeling. This thesis focuses on neural-computing-based approaches to processing industrial sensory data. Predictive machine maintenance, work-in-progress on product-quality prediction, and fast data modeling are selected to show how neural computing can be used to uncover useful information from industrial sensory data. In this thesis, neural computing refers to ANNs-based methods for solving specific problem-modeling tasks.

The author's scientific publications constitute the main chapters of the thesis. In summary:

- The thesis starts with an introduction to Echo-State Networks (ESNs). A compositional method is proposed for modeling the milling process, combining ESNs with statistical features. The main goal of this work is to achieve predictive maintenance for milling machines. While ESNs are a powerful tool for modeling time-series processes, selecting appropriate hyper-parameters proves to be very challenging. To tackle this issue and ensure that ESNs are practical, we propose a heuristic random-search method we call Zoom-In-Zoom-Out (ZIZO).
- A deep neural network solution is than proposed for predictive maintenance without directly relying on statistical features. This solution takes advntage of ANN-based autoencoders to extract the relevant features of the system. These features are subsequently fed to a filter, to monitor and predict the performance of the production system.
- In practical applications, system-maintenance engineers usually wish to have a quick overview of the performance of a production systems in particular circumstances. Quickly producing

such an overview is difficult however, as it requires modeling of both normal and abnormal system behaviors. For such situations, a novel deep neural network architecture is proposed, which we called a Multi-Bias Recurrent Neural Networks (MBRC-RNN)..

- ESNs, autoencoders, and MBRC-RNNs are used to design algorithms for predictive maintenance and fast data modeling. The quality of work-in-progress products however, has a strong connection to the health status of production-line machines. For better product-quality control, it is necessary to investigate quality prediction issues. A generative neural network is proposed and experimentally validated to solve the quality-prediction issue.

- Statistical-analysis-based methods have been used in industry for decades. In order to clearly show the advantages and disadvantages of statistical methods in production quality control, a novel Bayesian network has been designed to monitor and predict a wafer manufacturing process. Since this part of the work does not have a strong connection to neural computing, it is presented as supplementary material in the Appendix.

Contents

List of Figures

List of Tables

Publications

Our publications in scientific conferences and journals are divided in two groups: Ph.D. Thesis Publications, and Other Publications. The first group is the basis for the Ph.D. thesis. The second group does not have a sharp or direct connection with the thesis, itself. However, it allowed us to explore the scientific principles of neural computing and to increase our expertise in this area.

Ph.D. Thesis Publications

- **Guodong, Wang**, and Radu Grosu. *Milling-Tool Wear-Condition Prediction with Statistic Analysis and Echo-State Networks.* In Proceedings of S2M'16, the International Conference on Sustaniable Smart Manufacturing. Taylor & Francis, 2016.

- **Guodong, Wang**, Mohamed Amin Ben Sassi, and Radu Grosu. *ZIZO: A Novel Zoom-In-Zoom-Out Search Algorithm for the Global Parameters of Echo-State Networks.* In IEEE Canadian Journal of Electrical and Computer Engineering 40, no. 3 (2017): 210-216.

- **Guodong, Wang**, Ramin M. Hasani, Yungang Zhu, and Radu Grosu. *A novel Bayesian network-based fault prognostic method for semiconductor manufacturing process.* In Proceedings of ICIT'17, the IEEE International Conference on Industrial Technology, pp. 1450-1454. IEEE, 2017.

- **Guodong, Wang**, Mohamed Amin Ben Sassi, and Radu Grosu. *A Muti-bias Recurrent Neural Network for Modeling Milling Sensory Data.* In Proceedings of ICPS'18, the IEEE Industrial Cyber-Physical Systems Conference, pp. 71-78. IEEE, 2018.

- **Guodong, Wang**, Anna Ledwoch, Ramin M. Hasani, Radu Grosu and Alexandra Brintrup. *A Generative Neural Network Model for the Quality Prediction of Work in Progress Products.* In print at the Applied Soft Computing Journal, 2018.

- Hasani, Ramin M., **Guodong Wang**, and Radu Grosu. *A Machine Learning Suite for Machine Components' Health Monitoring.* In Proceedings of IAAI'19, the 31st Innovative Applications of Artificial Intelligence Conference, Hawaii, USA, 2019.

Other Publications

- Hasani, Ramin M., **Guodong Wang**, and Radu Grosu. *Towards Deterministic and Stochastic Computations with the Izhikevich Spiking-Neuron Model.* In Proceedings of IWANN'17, the International Work-Conference on Artificial Neural Networks, pp. 392-402. Springer, 2017.
- Zhu, Yungang, Hongying Duan, Xinhua Wang, Baokui Zhou, **Guodong Wang**, and Radu Grosu. *Gaussian convex evidence theory for ordered and fuzzy evidence fusion.* In Journal of Intelligent & Fuzzy Systems 33, no. 5 (2017): 2843-2849.
- Zhu, Yungang, Dayou Liu, Radu Grosu, Xinhua Wang, Hongying Duan, and **Guodong Wang**. *A Multi-Sensor Data Fusion Approach for Atrial Hypertrophy Disease Diagnosis Based on Characterized Support Vector Hyperspheres.* In Sensors 17, no. 9 (2017): 2049.

CHAPTER 1

Introduction

In 1948, Norbert Wiener published one of his most important works: "Cybernetics: Or Control and Communication in the Animal and the Machine" [Wie61]. He proposed the use of the term "Cybernetics" to refer to self-regulating mechanisms. The essential goal of cybernetics is to understand and define the functions and processes of systems which move from sensing to actuation, and compare results with desired goals. Its focus is on how things (digital, mechanical, or biological) digest information, respond to it and modify it, to accomplish the first two tasks optimally [Kel09]. Concepts studied in cybernetics include, but are not limited to: learning, cognition, adaptation, social control, emergence, convergence, communication, efficiency, efficacy, and connectivity. However, modern scientific works mostly use cybernetics to refer to control engineering.

With the speed of development in information technology, especially the emergence of the "Internet of Things (IoT)", distributed systems seem to fit the concept of cybernetics better. The IoT is the concept of connecting different devices and remotely controlling them. For instance, in modern factories, the industrial IoT has been widely adopted for collecting generated data from production processes and using it to improve the intelligence level of modern manufacturing. The disadvantage of IoT is that it loses part of the interaction between facilities. In such a circumstance, the US National Science Foundation (NSF) proposed in 2005 the idea of Cyber-Physical Systems (CPS) to keep the US in an advanced position in manufacturing. CPS tightly integrate computing processes and networks and physical processes. There are many debates on the relation between CPS and IoT. In verbatim, CPS focuses more on the interaction between the cyber domain and physical domain, while IoT emphasizes more the communication and interconnection among devices. CPS could, therefore, be regarded as the upgraded version of the IoT, with intelligence. Later, the German federal government proposed a new national industry plan called "Industry 4.0s" to improve Germany

manufacturing technology. Industry 4.0s, also called "CPPS" (cyber-physical production systems), aims to design modern manufacturing systems with self-X properties, such as self-healing and self-scheduling, and enables manufacturers to provide more personalized products for customers' requirements. Concerning the essential aspects of cybernetics, one of the chapters discussed in Wiener's book is "Computing Machines and the Nervous System" [Wie61]. Against a background of modern industry, the machines mentioned in Wiener's book can be computerized production facilities or even whole production lines. In order to realize the idea of CPPS, adapting the intelligence of "nervous systems" seems promising. Therefore, in this thesis, industrial IoT is considered the infrastructure for collecting sensory data and neural computing techniques are adopted to increase the intelligence of factories, for example by adding predictive functions to production lines.

1.1 Research Motivation

The support of complex industrial processes by Information Communication Technology (ICT) is a foundation of what is often called the next industrial revolution or "Industry 4.0". It is therefore considered to be a crucial research question in this field. Not surprisingly, "Industry 4.0" has been recognized by national and European policy makers. The topic is prominent in funding schemes such as the "Factories of the Future" program from the European Union, the US Advanced Manufacturing initiative, and major national programs such as "Produktion der Zukunft" (Austrian Research Promotion Agency (FFG)) in Austria.

Manufacturing systems are extensively equipped with sensors, actuators, and controllers. The automation pyramid is linked through a communication network to an either strictly centralized or strictly decentralized IT structure. A fixed set of products and services are typically offered, and their realization is carefully planned, which is unfortunately too sequential, very comprehensive, and hardware- or product-specific. An alternative method is to equip production systems with monitors. The monitored status can later be used for predicting emergent behaviors at run-time, for example predicting the failure time of machines. This approach makes CPS self-aware and opens up new perspectives on designing smart production systems.

Self-X properties aim at enabling manufacturing systems to handle unseen or unexpected events so that the systems can run freely with little or no human supervision. Hence, at least two types of functions should be added to production systems: (1) Automatic updating of decision-making parts to achieve autonomy. (2) Efficient handling of noise in the dataset. Artificial Neural Networks (ANN)s have demonstrated their flexibility and robustness in various classification and regression tasks. Two properties of ANNs are considered as very significant: (1) Automatically adapting to the dynamical environment through advanced off-line or online learning algorithms. (2) Transforming input data

into higher or lower dimensions through nonlinear (linear) activation functions. Post-processing parts of ANNs take the processed data, which we called neural representation, as inputs to solve the prediction and control problems. The learning ability of ANNs enables them to learn the essential information underlying the provided training data and generate the prediction or control models. Moreover, ANNs can learn to remove the noise in the provided dataset.

Given the above benefits, this thesis focuses on designing artificial neural networks-based general solutions for achieving the partial goal of Industry 4.0, that is, designing flexible and intelligent manufacturing systems through smart data analysis.

1.2 Problem Statement

The history of CPPS was presented above. From a technological point of view, there are still many challenges to be urgently solved. To narrow down the research scope and present the scientific contribution of the thesis, the main research problems are given in this section. The specific research questions are described in Section 1.3. The scientific contribution of the thesis is discussed in Section 1.4. Finally, Section 1.5 briefly describes the organization of this thesis.

When a team sets up a data analysis project, its first step is always to conducting data preprocessing. This amounts to removing data noise and corrupted data, before actually modeling the industrial processes. Another essential step is to selecting key features (or variables) from the data. They approximately represent the most useful information within the collected data. The critical features mentioned here primarily determine the modeling accuracy of the proposed solutions.

Over the past decades, statistical-analysis methods have played a vital role in industry, especially for feature extraction (eg. wavelet analysis for bearing [PPY01, ED01], and vibration analysis [OAC06] for mechanical-systems fault diagnosis). However, in spite of equipping modern manufacturing systems with a huge number of sensors, which continuously transmit and store vast amounts of data every second, statistical analysis methods still lose critical system information. One reason for such a loss is that statistical-analysis methods are based on human experience and idealized assumptions. These may not always hold in the real world. In other words, the proposed methods (pure statistical-analysis methods) may lack the ability to properly handle uncertainty in the collected data. Most importantly, statistical methods are often problem-specific solutions. For instance, the users must manually select the desired statistical features for every application.

Deep neural networks (DNN)s on the other hand, have demonstrated that they can take advantage of deep structures to automatically explore the relevant information patterns within a system. This is precisely the primary goal of the thesis. It should therefore not be surprising, that recent progress

in machine learning techniques and DNNs are the focus of this thesis. More precisely, developing automatic methods for industrial data processing is the main goal of this thesis.

Methods based on DNNs work well for industrial applications. However, they have one weakness worthy to note: While DNNs can detect patterns that experts may not be aware of, the generated result may be unexplainable. This is the reason why most people take neural networks as black boxes. Under such circumstances, part of the thesis looks into a classical statistical method, called "Bayesian networks", which presents the steps of modeling a system with multi-states. Specifically, a novel Bayesian network is used to monitor and predict product quality in a wafer manufacturing process. The underlying idea of the work is to combine a DNN with a Bayesian network and create an explainable industrial solution, described in the future work. As the main topic of this thesis is neural computing, this part of the work is presented as supplementary materials in the Appendix.

The theoretical goal of this thesis is to study neural computing for monitoring and predicting the performance of CPPS. The practical goal of this thesis is to provide general end-to-end industrial data process solutions for predictive machine maintenance, product quality prediction, and fast industry data modeling. These research statement goals mean that the solutions to be developed within the scope of the thesis should possess the following attributes:

- *Simplify industrial data processing by deep neural networks.*
- *Present general solutions for industrial data processing by machine learning techniques.*
- *Develop suggestions for making modern factories smarter and more reliable.*

Solutions having the above attributes are attractive from two perspectives. First, they facilitate traditional industries to move towards Industry 4.0. Second, they introduce general neural-computing solutions for: (1) Automated AEC health-monitoring-and-prognostic methods for machine bearings. (2) Work-in-progress product-quality predictions with generative neural networks. (3) Fast data modeling with multi-bias complex neural networks. These solutions could work as general industrial data processing frameworks. They also allow users to apply future more advanced Artificial Intelligence (AI) technologies to manufacturing industries to understand the proposed frameworks.

1.3 Research Questions

The following research questions underpin the research conducted within the scope of the thesis:

RQ 1: *How to select significant features from system inputs?*

Selecting useful features is always challenging in the AI community. For example, if one wants to design a face recognition system, the choice of essential features is determined manually by researchers and engineers with professional experience. Although these manual-selection-based methods can solve the so-called feature engineering problems, it seems that manual selection is significantly low in efficiency. Recent progress in the deep learning community has shown that one can use DNNs and automatically determine relevant features. Specifically, they require much less time than a human, while uncovering much more valuable information.

The efficient selection of features solves two issues. First, because of the massive amount of data being generated every day, time is limited. Engineers must identify meaningful information in short periods of time, which is difficult and the performance of the final solutions is not guaranteed. Second, humans are limited by their own experience. The features in statistical analysis are those one can understand correctly. However, the world may be much more complex.

For those reasons, for feature-selection we use two different approaches. The first is to merely choose as many statistical features as possible and feed them into a complex neural work (ESN). This is "semi-automatic industrial process modeling." The other is to use DNNs to extract essential features and use the results within models for monitoring and prediction of system performance.

RQ 2: *How to achieve fast-data modeling for industrial processes?*

Modern factories are complex systems equipped with many digitalized subsystems. They may contain chemical reaction processes, physical interaction processes, digital computing processes, and so forth. The processes or running environments of these complex systems may evolve in a short time span, while the properties of other components may be updated very slowly. The underlying problem here is that different components in one manufacturing system may evolve at different timescales. Hence, a model-free approach is needed to capture the changes in a complex system. This part of the thesis, therefore, looks at one type of neural network: MBRC-RNN, which captures precisely the evolutionary trend in complex systems.

RQ3: *How to effectively choose the hyper-parameters of ESNs?*

ESNs are an essential tool for modeling the evolving trends in complex systems. Unfortunately, ESNs are hard to design as they contain several hyper-parameters (or global parameters) that control their properties. These have a significant effect on the generation of the internal states of ESNs. Without proper internal states, ESNs cannot simulate the evolving trends of a complex system.

The exploration of hyper-parameters is a common issue for the AI community, and many researchers have proposed useful algorithms. Since the structure of an ESN is very different from traditional neural networks, research is necessary to arrive at practical solutions for ESNs.

RQ 4: *What should be end-to-end industrial data processing?*

Modeling the manufacturing processes requires substantial effort to carefully select appropriate models. For instance, to process the industrial sensory data, one uses a model to extract features, and than other models to predict properties or to control the manufacturing system. The problem here is that frequently switching models reduces the performance of complete solutions.

End-to-end modeling techniques apply a complete method for solving a problem. In this thesis, an end-to-end solution refers to a DNN or a complex neural network that can be constructed (possibly through sub-networks) and trained as a single model. Sub-networks are used to solve specific sub-problems and that are stacked together to solve overall problems. This method is feasible as the stacked neural models can be trained at the same time, by applying one single training algorithm.

RQ 5: *Special case study: a novel Bayesian network for predicting the performance of a wafer manufacturing process.*

The semiconductor industry is a booming industrial field. Every achievement in this area pushes significant advances in computer-science-related fields. The principal products from Semiconductor industry are wafers, which are designed and produced in hundreds of steps of sophisticated production lines. Seamless production is one of the essential features of wafer manufacturing. The underlying issue is that after feeding materials into production machines, it is challenging to detect and predict correlations among the individual steps.

In common practice, there are always some defective chips in a wafer. In this part of the thesis, research on the wafer-manufacturing procedures is conducted using a Bayesian-networks approach. The purpose of this part of the work is to combine DNNs with Bayesian networks to constructing explainable industrial data processing platforms (in the future work part).

1.4 Scientific Contribution

This section gives more details on the scientific contributions of the thesis. More technical details and experimental results are presented in the following chapters.

Contribution 1: We demonstrate that ESNs can automatically merge various statistical features and generate accurate prediction models.

Statistical analysis techniques are the foundation of modern science, notably in industry decision-making. However, given various batches of statistical features, it is often difficult to manually select the most relevant ones. This thesis demonstrates that without sophisticated selection (e.g., applying only the Pearson correlation to remove redundant information), ESNs can automatically merge these

inputs and generate accurate models. In order to help the ESNs obtain more hidden information, the only thing that needs to be taken care of before training the ESN is to provide as many statistical features as possible. The method consists of two steps: (1) Apply statistical analysis to pre process the inputs. (2) Train the ESNs by ZIZOand linear gradient descent. Details of this method, the experimental results, and the relevant statistical features selected are presented in Chapter 4.

Contribution 2: We developed an end-to-end industrial data processing platform, based on deep neural networks.

End-to-end industrial data processing is valuable for the modern manufacturing sector, as it creates accurate models and makes real applications much more accessible. This part of the thesis provides two types of end-to-end platforms: One based on deep learning techniques, specifically on autoencoders. The other is based on ESNs. For the DNN method, an autoencoder is first trained to extract essential features of inputs. This is followed by simple but effective filters calculating the system's deviation from normal behavior. The details of this part of the work are given in Chapter 6.

Contribution 3: We designed a heuristic random-search algorithm for selecting (learning) appropriate hyper-parameters for ESNs

Effectively adjusting hyper-parameters for machine learning algorithms is a barrier for AI researchers and engineers, killing their passion to learning this advanced technology. In this part, a ZIZO hyper-parameter searching algorithm is presented to explore the appropriate global parameters of ESNs. There are two important aspects of this algorithm that should be paid attention to: (1) How to control the value-range of every variable after updating internal states of the ESN? (2) How to select new central positions for generating the next batch of hyper-parameters, after several generations of hyper-parameters were already generated? This algorithm is presented in Chapter 5.

Contribution 4: We designed a complex multi-bias neural network for simultaneously processing sensory and non-sensory data.

A novel multi-bias complex neural network for end-to-end data processing is also proposed in this thesis. It is considered as a useful tool for fast-data modeling. Two of its essential properties are: (1) ESN properties are adapted to model the fast-evolving inputs. (2) Internal states of a neural network, which are updated in a short time, are then combined with the inputs that changed dynamically in large time scale. Details of this model are presented in Chapter 7.

As a conclusion, it is demonstrated that end-to-end is a feasible and valuable solution for Industry 4.0. End-to-end processing is not only applicable to image processing and natural language processing, but also a promising toolkit for real industrial applications.

1.5 Structure of the Thesis

The rest of this thesis is organized as follows:

Chapter 2 briefly reviews the cutting-edge research related to cyber-physical production systems, smart factories, and smart predictive maintenance. More details of other existing works are introduced in specific chapters, because: (1) The thesis wants to compare these works with the proposed methods simultaneously. (2) It is convenient for reading and comprehension.

To provide a better understanding of the ideas presented in this work, Chapter 3 briefly introduces the technical background of the thesis. This chapter investigates two fundamental subjects in production systems: (1) What are the smart attributes of a CPPS? In this regard, the thesis supplies CPPS with learning skills. (2) Define a general technical framework for constructing smart factories. Machine learning and deep learning algorithms are adopted for constructing the smart factories framework.

According to the general framework proposed in Chapter 3, Chapter 4 studies a semi-automatic method for modeling manufacturing physical processes. The neural networks used here are ESNs. The experimental results demonstrate that with much less human effort, ESNs can automatically merge various statistical features and generate accurate outputs. One milling dataset, collected from the NASA data repository, is used to prove the performance of the proposed method.

Chapter 5 presents the algorithm designed to effectively search for hyper-parameters of ESNs. During the preparing procedure of Chapter 4, we found out that manually adjusting the hyper-parameters of an is tough work, as there is some parameters effect on each. Motived by such challenge, the ZIZO algorithm is proposed. Generally, hyper-parameters are always barriers to users applying deep learning techniques. This section demonstrates that with proposed ZIZO heuristic searching algorithm, one can easily use an ESN and adopt it for real industrial applications.

Building on top of the framework mentioned in Chapter 3, Chapter 6 introduces a fully automatic industrial data-processing approach. Specifically, the means of automatically extracting valuable information from inputs are investigated. In the end, a simple but powerful industrial data-processing platform is proposed. A series of run-to-failure test of a bearing dataset are tested on the platform, to demonstrate its performance.

Chapter 7 proposes a general architecture of a multi-bias complex neural network for modeling industrial sensory data. This model can process sensory and non-sensory data in parallel. The dataset from Chapter 4 is reused to evaluate the performance of the proposed model. Chapter 8 presents a generative neural network model for predicting the quality of work-in-progress products.

Until now, the thesis includes most of the work done for solving specific problems. The typical

character of this work is that our models are trained on an industrial data set, for which they use a variety of learning algorithms to find the best models fitting the probability distribution of collected information. Additionally, all the proposed models are artificial neural network-based methods. They learn to automatically extract the essential information out of training data and generate the desired output with different neural networks. Therefore, we conclude that neural computing could work as a learning engine helping CPPS gain self-x properties, e.g., self-awareness and self-configuration, which generally are called self-X attributes (Chapter 3). Since the proposed concept is a purely data-driven, the same ideas can be applied to different layers of CPPS, e.g., device connection monitoring (smart connection level). Alternatively, Cyber management makes use of learning technologies to detect a malicious cyber attacks automatically.

Although the idea of using neural computing as the learning engines for industrial data processing looks promising, it is still far too abstract. Hence, we came out with a concrete neural computing pipeline under the concept of "smart attributes for CPPS" for practical applications (Chapter 3). Users can easily copy the pipeline adapt to their applications.

Appendix comprises additional work in support of neural computing-based approaches, i.e., constructing explainable DNN-based methods by adopting a Bayesian network. Although neural network-based techniques are powerful, more efforts should be made to explain the learning results. One of the great advantages of Bayesian networks is that they are explainable, though their drawback is that they cannot scale to large datasets or more complex circumstances, unlike neural network methods. This part of the thesis, therefore, intends to identify the mathematical logistics inside problems. In future work, Bayesian networks will be combined with neural networks to provide powerful and explainable neural computing-based approaches for industries. In this chapter, a special industrial case is considered, i.e., fault monitoring and prediction for chips manufacturing process. The underlying idea of this study case is to show the performance of a novel Bayesian network and set up the foundation for future work, i.e., combining deep nets with a Bayesian network to produce simple but effective industrial models.

Chapter 9 concludes all the research activities among the Ph.D work. Future work as the continuous of this thesis is also mentioned in this Chapter. The main goal of the thesis is to propose a generic framework for solving a number of urgent problems, in which modern industries are facing today. A conclusion is derived from the findings and a neural computing-based general technical framework for CPPS is proposed at the end of the Ph.D work. The overview of the thesis is depicted in Figure 1.5. Chapter 1, Chapter 2, and Chapter 3 are presented in sequence. Chapter 4, Chapter 6, Chapter 7, Chapter 8 and Appendix are presented in parallel as they are developed based on the theory presented in 3. Chapter 5 as the support tool directly linked to Chapter 4.

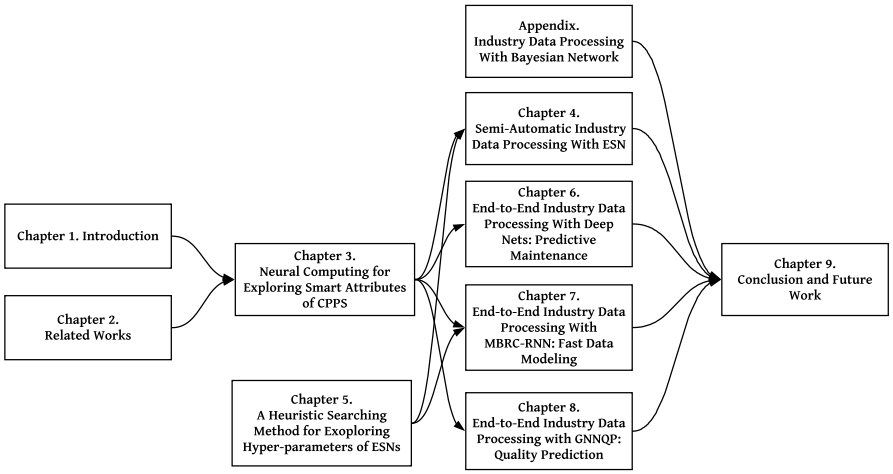


Figure 1.1: Overview of the thesis.

CHAPTER 2

Related Works

This chapter presents an overview of the existing works related to this thesis topic. As mentioned in the first chapter, this thesis intends to explore methods of building smart manufacturing systems (from the perspective of information processing). The research work was conducted by solving several particular industrial issues, and then an end-to-end industrial sensory data processing framework is proposed (Chapter 6) and Chapter 7). The research work is constituted by three main elements: (1) modeling the physical part of CPPS (Chapter 4); (2) sensory data processing for CPPS (Chapter 6); and (3) Bayesian networks for monitoring and predicting the wafer production status in the semiconductor industry (Appendix Section). The literature review will separately introduce the state-of-the-art works for Chapters 4, 5, 6, 7, 8 and Appendix. Additionally, since the effective selection of appropriate hyper-parameters for ESNs is considered, a short review of the relevant methods is also presented in this chapter. Please note that although wafer quality monitoring and prediction is chosen as one of the demonstrations, the presented method is Bayesian-based which is not the main goal of the thesis. Hence, related works are introduced shortly.

2.1 The General Application Scenario for CPPS

Before introducing the technical issues surrounding the construction of smart factories, this thesis presents an overview of CPPS to illustrate the smart manufacturing systems which this thesis intends to explore. Figure 2.1 depicts the general idea of CPPS, in which the research activities are conducted under similar physical facilities.

Taking into account the physical environment of modern factories, the manufacturing systems are classified of either the physical world or data world (cyber world). Components in the physical world

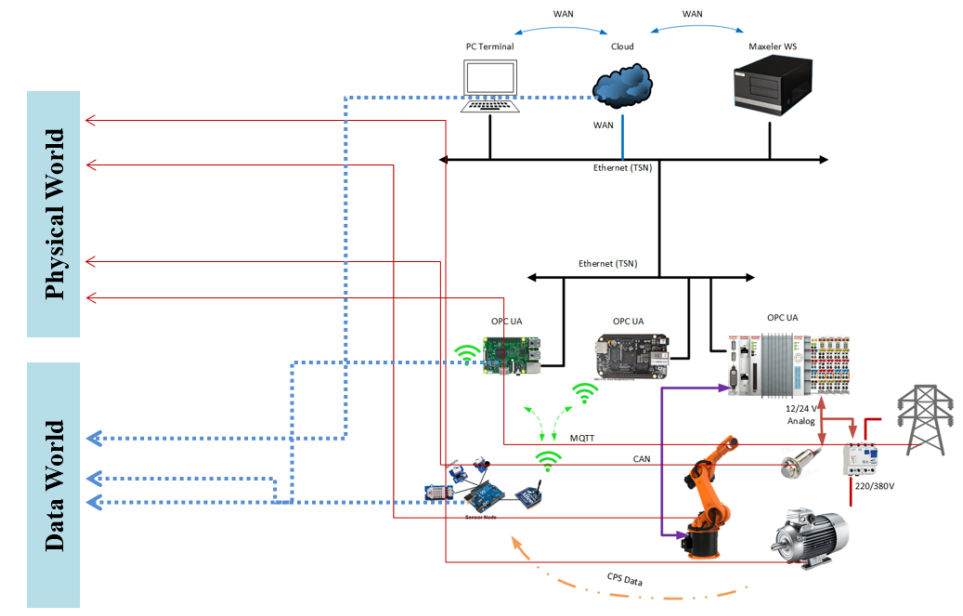


Figure 2.1: General idea of CPPS.

are connected via industrial internet. The cloud depicted in Figure 2.1 functions as a data processing server. As noted above, this thesis focuses on industrial data processing using neural computing techniques. The general goal of the work is to provide a general technique framework for smartly managed production systems, such as diagnostics, prognostics, systems performance monitoring, and prediction. Hence, this research work begins by deciding the source of the data in the cloud to be used and developing various algorithms or models for solving different problems facing the industry today.

From a technological point of view, automatic processing of the sensory data could be the most promising approach to implement smart production systems as it would release engineers from repetitive, tedious work and save production expenses. However, with a massive amount of data continually flowing into the cloud, traditional methods face two main issues: being unable to handle the essential information from the sensory data and not being sufficiently robust for handling dynamically changed production environment. Another challenge is that various sensors installed around production platforms in modern factories make the task of identifying essential information more difficult. For the reasons as mentioned above, one can conclude that the core issue is sensory data feature selection, and this is the basis on which the models are built later in this work. This thesis takes neural network-based approaches to implement automatic data processing by making use of their learning ability. The artificial neural network has been adopted mainly for feature extraction, e.g., deep autoencoder could provide a deep neural representation of inputs which cannot be obtained

from human experience.

The following sections introduce the relevant works on physical process modeling and hyper-parameter searching for the application of neural computing methods. Additionally, Bayesian network applications in manufacturing process management are explored, specifically wafer production process monitoring. In order to ground the research work, the following section takes the industrial milling process as a research object and seeks to answer the opening questions regarding automatic industrial data processing. The remaining part of the scientific discussion and methods development takes the bearing process and wafer manufacturing process as study cases.

2.2 Related Work on Physical Process Modeling of CPPS

Against the background of CPPS, the idea of automatic data processing was implemented by applying it to solving the prognostics and diagnostics issues of manufacturing systems, i.e., monitoring the current status of systems and predicting remaining useful life (RUL) through automatic data processing methods. Relevant issues are covered, such as industrial data analysis, big industrial data modeling, and the manufacturing process model. Researchers from the prognostic and health management community have developed useful tools for monitoring the running performance of industrial machine tools, such as bearing machines [JLB06, LWZ+14]. Some statistical features from the time domain are frequently adopted for monitoring the work condition of the critical parts of production systems, for example, kurtosis, root-mean-squared (RMS), and spectral kurtosis [QLLY06]. However, these features do not always work well for some specific applications because they are not able to identify enough useful information from the collected data. Hence, the created models which make use of statistical features lack generalizability for similar application scenarios. Accordingly, methods of efficiently and automatically selecting property features are urgently needed to address diagnostic and prognostic problems [Yu12a]. Manual selection of features could lead to unexpected issues and a lack of flexibility, as manually selected features are based only on human experience.

Unsupervised feature extraction techniques seem promising for solving the issue as mentioned above. For instance, He et al. [HKY07] propose a principle components analysis (PCA)-based method. PCA was first adopted for analyzing inputs correlation and reducing inputs dimensions. A Hidden Markov Model (HMM) was then applied to monitor system performance degradation. This approach provides a reasonable technical framework for predicting systems coming states [Yu12a]. Although such methods provide a reasonable model and accurate predictions, they are not executed automatically. Additionally, the generality of this type of model is not investigated. Another method, such as a mixture of features from the frequency domain (for example, a mixture of HMM with Wavelet packet

decomposition) could be applied for modeling the RUL of production systems [TMMZT12].

Tobon et al. [TMMZT12] propose a method of extracting the trend of bearing machine's RUL and successively applying extracted features on several run-to-fail bearing machine experiments. However, it is necessary to change part of the configuration of the proposed method in order to apply the method to other application scenarios [TMMZT12]. Methods such as the Kalman filter (KF) are also widely applied in condition monitoring. [RM14, WPZ+16]. Reuben et al. [RM14] present a method of conducting diagnostics and prognostics simultaneously on collected data by applying switching KF [RM14]. Wang et al. [WPZ+16] propose a method of accurately predicting the health condition of a machine by combining enhanced KF with an expectation maximization approach. Generally, these approaches lack automation property because a significant amount of human effort is needed for data pre-processing.

Recent achievements in the development of artificial intelligence (AI) have shed light on conducting diagnostics and prognostics on critical parts of manufacturing systems. AI-based methods mostly rely on collected data. Thus the process of data analysis can be executed automatically with little or no human effort, in contrast to the techniques mentioned above. Various neural networks have been developed for data processing, and for example, various autoencoders are successfully used to classify different types of faults. Autoencoder-based approaches adopt time or frequency statistical features, as well as other features generated from prior knowledge, to achieve much more accurate results [YSMP02, JLL+16, SSZ+16, TDS+16]. Although those methods are not fully automated, it is clear that they require much less human effort. One conclusion that can be reached from the reviewed literature is that data-driven and AI-based methods are not yet being executed in a fully automatic way to solve prognostic and diagnostic issues in the industry. The following literature review will focus on concrete research objects to make the results of the thesis more applicable. It begins by introducing maintenance related works of milling tool-wear.

Milling tools are frequently adopted in manufacturing systems. They have been selected here because the shaping of metals is quite common in the industry, especially in large-scale machinery manufacturing companies. Milling is the machining process of using rotary cutters to remove the metal surface of a workpiece, at a predefined direction and using cutting angels (the axis of the tool) [GNGGVR16]. Production machines vary from small sized materials to large facilities. Milling operations, therefore, differ for different materials and production systems. It is well known that suitably shaped metals indirectly indicate an excellent condition of machining tools as the milling process is serious time-series procedure, e.g., a broken tool surface causes bumpy faces in the shaped metals. Additionally, tool wear reduces the RUL and makes machining tools increasingly fragile. In the physical interaction process between the metal surface and tool, more friction is introduced by a gradually worn cutting tool. This friction then increases the temperature of materials or tools, which

is released in the physical process, damages the health of the machining systems, and causes valuable resources to be wasted. The modeling tool wear process has naturally attracted attention in industry and academia over recent decades.

As mentioned earlier, tool wear has a significant effect on the quality of final products. That has led to high demand for efficient solutions to tackle tool wear monitoring and prediction issues. Various tool wear proposals have been studied extensively, including rounding wear of the cutting edge, crater wear on the rake face, and friction-caused flank wear. Meanwhile, the corresponding causes of tool wear have been discussed, for example, rotator moving speed, cutting depths and width, and feeding rates. These factors have a strong connection with tool wear degree, while indirect features such as material types and acoustic and vibration signals have also received attention, as they can accelerate the tool wear speed and break down the tool's sharp degree. Generally, identifying the complex relationship between tool wear and other factors is essential for producing high-quality products and minimizing production costs, because the products generated by broken machining tools are not acceptable.

In recently published works, many researchers have explored milling tool wear-related topics and developed many valuable techniques for increasing the RUL of production systems and lowering production costs at the same time. For instance, [ASH+12] suggests applying an acoustic spectrum, while Jesus [dJGIC03] proposes adopting a current signal for analyzing tool wear status. The disadvantage of both of these methods is that they only consider single statistical feature and ignore features such as table vibration and spindle vibration, which may be essential for systems performance analysis. Without complete investigation, the milling tool wear models usually produce unreliable results.

With the rise of AI, feed forward neural networks (FFNN)s are considered an effective way of monitoring tool wear status [GRP+07, PRS08], though the weakness is that these FFNN-based approaches do not consider the time variation property of sensory signals. Machining processes, for example bearing and milling, are essentially time-series procedures because the tool wear condition has a strong connection with the historical states. Mosallam [MMZ14] proposes a method for extracting the trend of multidimensional sensory signals. The sensory data is collected in sequence and labeled with a sampling time stamp. In order to model systems performance, extracted successive features from machining sensory samples are adopted. However, the underlying problem is that the proposed method does not consider the causal effects of different time steps. With predefined time intervals, samples from timestamp t have a strong connection with those sampled from $t-1$, $t-2$, or $t-n$. Study of the time-series production process is, therefore, the key to increasing the RUL of manufacturing systems.

2.3 Neural Network-based Methods for Industrial Data Processing

With advanced ICT, modern production systems are increasingly equipped with high precision sensors installed around operation platforms. These sensors can help to monitor the health status of and detect deviations in manufacturing systems. With massive amounts of data generated by ICT infrastructure every day, the compelling collection of data is a challenge. Numerous methods have been developed for analyzing these sensory data. From a technological point of view, these approaches can be divided into two different categories: model-based and data-driven [BMZR15]. In order to optimize sensory image analysis based on a region growing algorithm, Zhu et al. developed a morphological component analysis model [ZY17]. To predict milling tool wear, Garcia Nieto et al. [GNGGVR16] and Li et al. [LXHW16] mixed particle swarm optimization and support vector machines (SVM) to develop practical hybrid regression methods for tackling relevant problems. Benkedjouh et al. [BMZR15] and Kimotho et al. [KSWMS13] adopted SVM with other approaches proposed novel methods for monitoring machining processes. Benkedjouh et al. and Kimotho et al. adopted SVM and other approaches to develop novel methods of monitoring machining processes. Specifically, they propose learning the function relationship between PCA extracted features and actual tool wear status.

Hong et al.[HYM+16] propose that combining Wavelet packet transforms (WPT), Fisher linear discriminant (FLD) and HMM together to model tool wear status. Specifically, WPT and FLD are adopted for modeling the micro-milling process, with HMM then applied to estimate the specific states. Gaussian mixture HMM models are selected for detecting delicate features in the work of Chen et al. [KCL17]. The main idea here is feeding extracted features from the Gaussian mixture HMM model into an artificial neural network for modeling tool wear states. In the work of Pati et al. [PGK16], WPT extracted features are applied as inputs of artificial neural networks.

Summarizing these neural network-based methods, one can conclude that there is a simple technique development pattern, i.e., adopting statistical features as the inputs and applying neural networks for modeling manufacturing process. It is clear that neural network-based methods are promising for industrial data analysis and predictive maintenance. The following sections will focus on introducing neural computing solutions for industrial application.

Ramin et al. [HHB+17] designed a sparse autoencoder for extracting bearing machine running time features. The original dataset is provided with 20 x 480 dimensions. Autoencoder runs as a features coder, automatically picking up the most important information retained inside the data. Finally, a smooth autoregressive model takes coded data as inputs and models the trend of the bearing machine. Guo et al. [GLJ+17] developed a RNN-based approach to modeling bearing machines. Xiao et

al. [XYY+17] suggest the adoption of RNN-modeling intensity function as a point process. The experiment results indicate that the issue of randomly generating time steps in predictive maintenance could be solved by RNN. FFNNs also work for other similar scenarios, such as monitoring gearbox bearing [BT15, DKN+16].

Instead of combining statistical analysis with artificial neural networks, other works propose mixture methods, e.g., combing nature-inspired techniques with artificial neural networks for improving the automation of the whole technical framework. For instance, Jaouher et al. [ACMS+15] combine simplified fuzzy adaptiveSimplified Fuzzy Adaptive Resonance Theory Map Neural Network (SFAMNN) with Weibull distribution. SFAMNN is capable of processing time-series data, which makes it perfect for modeling bearing degradation in time sequence. Corne et al. [CNEMK16] adopt neural network predicting tool wear degree in real time. Other works such as DNA coding-based methods also seem promising, e.g., Addona et al. [DUM17] propose to apply DNA-based computing techniques to extracting inputs features. The performance of other methods including genetic algorithms, evolution algorithms, statistical analysis-based methods, and hybrid neural networks models is also demonstrated by experiment simulations and industrial applications [LWH17, CLY17, CZC+16, Lia14, YZ15, BKN17, FLZ16, SPMK16].

2.4 Related Work on Product Quality Control and Prediction

Predictive quality control is an important issue for customer satisfaction in the manufacturing sector [CTF+14] and non-conforming products create a waste of time and production resources [NTC+13]. Quality control is the process of monitoring, predicting and improving the quality of products within manufacturing processes [WCQS18]. The common practice in industry is to adopt statistical process control (SPC) toolkits for tackling quality control issues [Mon09, Rya11], which includes sampling from batches produced during a process, and examining the associated control chart containing sample dimensions, as well as the the upper and lower bounds of the sample dimensions forming the specification around the dimension[Rya11, BPP07, LSL11]. Any product samples either greater than the upper limit or lower than the bottom limit suggest increased likelihood of products that violate quality control specifications. The quality control specifications of work-in-progress (WIP) products on subsequent processes are gradually tightened as they are passed through one process stage to another [WWB07].

During past decades, many valuable works have been developed for SPC, such as statistical mechanics [ABBB07] and data-driven methods [YL12, SHCC06]. Statistical mechanics-based approaches make use of statistical features such as mean, variance, maximum, minimum, kurtosis, skewness and root-mean-square (RMS), as the direct or indirect indicators of the products' quality [WG16a], to gain a

better understanding of the status of the WIP products. However, due to the increased complexity of production systems, statistical mechanics are not sufficiently geared to handle increased data and uncertainty from modern factories [LKY14]. More specifically, abnormal patterns have been traditionally identified through statistical tests. However, these are only able to detect pattern presence, not which one occurs. Moreover, these techniques suffer from errors such as false alarms and missed disturbances.

With the recent development of more advanced data collection and monitoring such as Internet of Things based systems [MKZ+16], [CBC17, WIT14] and improved computational power manufacturers are increasingly investigating machine learning based methods for predictive quality control [Wan11, WLS10, Qin12, YWG16, Yu12b, ACCPM13]. The popularity of machine learning is due to better accuracy without the need for manually detecting and interpreting patterns [HG12]. However, the main drawbacks from machine learning based approaches have thus far included their ability to handle noise and imbalance in data.

The roots of SPC as a quality control mechanism go back to the 1920s, when W.A. Shewhart first proposed the use of control charts [She31], to detect out-of-control signals indicating the existence of particular causes that affect process stability by providing a visual indication of the behavior of critical quality variables. Common patterns in control charts facilitate the detection of a set of possible causes - for example, shift patterns may indicate changes in material, machine, or operator, while trend patterns may indicate tool wear. An in-depth survey of control chart recognition literature can be found in [HG12].

In the last decade, researchers have argued that machine learning based approaches, in particular, Artificial Neural Networks (ANNs) could produce more accurate results and thus numerous studies have been developed [WWB07, Mit16, Mon14, LBK15, WIT14, BMA+17]. In what follows we first briefly go through the classical machine learning based approaches and argue their advantages and disadvantages, then focus on ANNs based methods.

Melhem et al., [MAD+15] introduced an approach for wafer quality prediction by adopting the historical data of health indicator. Melhem et al. suggested to use principal component analysis (PCA) to extract significant indicators first, and then applied the indicator as labels to the training process in Bayesian networks, regression models, neural networks, SVMs, K-Near Neighborhood (KNN) and other clustering methods. The proposed approach demonstrated a reasonable performance however it was noted that the indicator extraction method would not generalize to multiple applications. Chou et al., [CWC10] successfully combined genetic algorithm with SVMs, to predict the wafer quality. Arif et al. [ASH13], suggested a mixture PCA method with an Iterative Dichotomiser algorithm (ID3), for multi-stage quality prediction, while Kao et al., [KHCL17] adopted classification and rule

mining approach for solving a similar problem. Diao et al. [DZY15], introduced another analytical tool called an improved dominant factor (DFs), which is based on an improved principle component analysis (iPCA), for dynamic quality control. The shortcoming of the proposed method is similar to that of PCA-based techniques. Essentially, PCA is a linear map [Cha18] which is not suitable for handling the uncertainty of the dataset.

Model-based approaches are also popular [SGLMH13]. Rajagopal and Castillo [RDC06] proposed a Bayesian method that can set a tolerance limit on one or more responses to provide a given probability of conformance, which was used to determine the optimal setting of the control factors. A novel structure for the Bayesian network was proposed to monitor and predict products quality of wafers during the manufacturing process [WHZG17]. Although model-based techniques provided good performance depending on the application, they are designed to solve specific problems and require prior knowledge on the system under test. Thus the analysis and modeling processes are not entirely automated.

Neural network-based methods for quality prediction have been especially popular, due to their ability in modeling non-linear dependencies [CTW$^+$08, SZLP11, AIVRGAMM12, TÖU13, HHG16, HHB$^+$17]. The general procedure of developing an ANN for quality prediction involves (1) synthesizing data at the initial step. (2) extracting useful information from raw data and presenting extracted information as new input-features of a neural network. (3) choosing a specific neural network structure based on practical application.

The way users apply ANNs for solving quality control issues can be classified into two categories: The ANN is directly applied for implementing quality control without input feature processing. For example, Pugh [Pug89] might be the first researcher to use Artificial Neural Networks (ANNs) for detecting mean shifts of control charts [TÖU13]. A self-organization map by using the back-propagation learning technique for a neural network was tested to predict product quality within a plastic injection process [CTW$^+$08]. The main setback in these works is that only the utilized shallow feed forward neural network cannot be adapted for extracting all the necessary features in similar problems. Although users may input the neural network with handcrafted features, such methods lack generalization and automation.

The other method used with ANNs are feature-based techniques [YX09], which extract input-features automatically. For example, with two-step feature learning, Bai et al., [BSD17] used a regression neural network to assess the product quality. Two-step feature learning methods such as the one proposed in [BSD17], eases the pre-processing stages by automatically extracting features from raw sensory data. These features are then learned by a second neural network to provide a prediction. Hasani et al., [HHB$^+$17] designed a compositional neural-network for monitoring and predicting the

product quality of complex analog circuits. Other methods, with significant improvement of modeling performance, included wavelet-ANN [ZD09, WZL15], statistical features-ANN [JMM09, WG16a] and deep neural network [WGY17, WCQS18].

A notable advantage of neural network-based methods is their automated learning process. However, it is worth noting that learning might be difficult in when not having enough data or having imbalanced data. For any of these cases, it has been noted that neural networks were harder to train and it was harder to achieve satisfactory predictive outcomes [DL15]. The case study faces both issues: the way the quality control process is currently implemented prevents from collecting more than a few samples from each batch of products in every sampling operation. Secondly, control limit violation situations are quite rare.

To tackle this issue, the data balancing technique called Synthetic Minority Oversampling Technique (SMOTE) [CBHK02] is adopted firstly, then an autoencoding neural network architecture prepares informative features from rebalanced data. The method then applies a feature based technique, namely the autoencoding neural network with a time-delayed multi-layer perception. The last step is decoding the coded prediction. In summary, the contribution of the work includes using the SMOTE methodology for data imbalance, and conducting the prediction on auto-coded data instead of using normalized inputs, as auto-encoder can handle uncertainty and noise from original data set well and is thus more promising for handling small sample sizes. Next, it postulates the methodology.

2.5 Related Work on Hyper-Parameter Searching for ESN

ESNs are increasingly attracting attention in various research communities and industries, due to ESN being useful for modeling dynamic manufacturing processes and analyzing industrial sensory data. Various practical applications have also been considered, for example automatic speech recognition [SH07b], chaotic time-series prediction [LHW12], feed-forward control [XLP05], motion identification [IvDZBP04], and prediction of telephone call load [BSU+15]. Modeling of time-series procedures is one field seeing intensive research attention. Within benchmark testing of the Mackey-Glass system [Jae01] and chaotic time-series signal prediction [LHW12], the presented results demonstrate the ability of ESN to model dynamic non-linear systems effectively and accurately. If one looks into the essential part of the milling process, tool wear gradually changes in a time-series fashion. ESN is one of the most powerful tools for modeling time sequential behaviors and is adopted in this thesis to explore the smart properties of CPPS.

Hyper-parameters such as leaking rate (LR), inputs scaling, spectral radius (SR), or size of a reservoir are essential for controlling ESN. However, there is no clear way of helping users to adopt ESN for practical application quickly. The most frequently used methods are brute-force searching and manual

trials. For instance, Mantas [Luk12] suggests users systematically try different hyper-parameters for specific problems. Skowronski [SH07b] believes that manual trials could obtain a group of hyper-parameters and generate a more robust ESN model for input signals. The disadvantage of these type of methods is: (1) users must conduct thorough experiment trials; and (2) users must be equipped with rich professional experience of applying ESN, which is not always the case in industrial applications. David [VS09] proposes the use of the dynamic profile of Jacobian reservoir instead of the static. He demonstrates that the approach could provide a more accurate description of reservoir dynamics and serve as a predictor of modeling performance. Bianchi [BLA16] presents a valuable tool for exploring the hyper-parameters of ESN. It can analyze the time-series data of neuron activation with recurrence plots and quantification analyses. This technique gives users a better chance of visualizing and characterizing high-dimensional dynamic systems. Although these two methods work well in the presented case studies, the prior information of many systems is needed, and this is difficult to obtain from system running environments. As ESN is one of the neural computing techniques included in the theory of reservoir computing, Oger implements the neural structure of ESN and conducts grid searching to explore the optimal and sub-optimal hyper-parameters of ESN. It is quite clear that computing performance is still a serious issue. Boedecker et al. [BOL+12] investigate the information processing of ESN. They found out that the edge of chaos is existing between the stable and unstable states of neural networks. To explore the best performance of similar randomly connect neural networks, Boedecker et al suggest the self-organized ways for determining the hyper-parameters of ESN. Similar work is also done by Livi et al. [LBA18].

A user-friendly searching algorithm is the primary goal of this part of the work. The core idea is to set up a tool which allows users easily apply ESN without caring the echo-state property. Theoretically, the works as mentioned above are rigorous while not feasible for practical applications.

2.6 Related Work on CPPS Monitoring with Bayesian Networks

Detecting various faults among different production phases is one of the significant challenges the semiconductor manufacturing industry faces today [SS16]. Upon investigation, one concludes that the techniques for verifying semiconductor chips can be divided into the following categories: pre-silicon fault diagnostic approaches [Low00, ACL+11, HHG16]; post-silicon fault detection phases [CMB07, MSN10, LKM+16], and defect prognostics during the fabrication process of the chip [FD97, WBGC99, FLT16]. The semiconductor fabrication process is very complex, comprising various productions and different processes [FLT16]. In order to improve product quality in different fabrication processes, it is essential to simultaneously check the critical deviation in the early stages

of the production line. The efficient monitoring and extracting of useful underlying information for supporting fault detection and monitoring of semiconductor fabrication has been requested for decades.

In order to reduce the waste of production materials, and also to save time and protect other types of investment, defects should be found in the early stage of abnormal behavior. To avoid critical behavior deviation, SPC and advanced process control methods are usually adopted to monitor and control processes and equipment [CHC13, MR16, ILS97, IS96]. However, with real-world problems, product level abnormal behaviors are only detected after parameter control and chip-level wafer test measurements. For example, Zhang et al. [ZWC16] propose an additive Gaussian process (AGP) model to estimate the standard wafer geometric properties and calculate the deviations from the criteria when products are in the manufacturing process. Among production lines, work-in-progress products interact with various machines or tools. At some extent, the status of those machines is essential indicators for the quality detection of wafers. Inspired by the similar idea, Melhem et al. [MAOP16, MAO+17] suggest to make use of the log file data of machines' status and apply regression model predicting the wafer quality. Besides the works as mentioned earlier, directly estimating wafer yield is another feasible and cheap way to do quality control and prediction. Because, with a specific circumstance, the wafer yield indicates the health status of manufacturing machines, and the correctness of various operations happened during manufacturing processes [DCW17, KŢBP17, WLT+18, KN17]. Among existing methods, Gaussian process-based models have attracted a lot of research attention in different aspects [DGD18].

The automatic method of exploring maps between abnormal behaviors and process control data could help to improve process control setup and reconfiguration [CWM18]. This strategy will help manufacturers to detect deviation as early as possible in production lines. It worth noting that wafer testing data should be sampled at the end of the process, and the process controlling data should be collected at the earliest stage.

Bayesian network is a powerful model for handling stochastic processes [ABP17]. It is usually able to summarize a complex problem and reconstruct it systematically and straightforwardly, using conditional independence. Hence, it is essential for understanding the core parts of problems. For instance, users may be lost in the big data applications, as one cannot see the overall picture of the data. However, with the Bayesian policy of divide and conquer, big data can be organized into small pieces and presented in a clear structure.

CHAPTER 3

Neural Computing for Exploring Smart Attributes of CPPS

This chapter introduces the main ideas raised during the preparation of this thesis, covering most of the theoretical aspects of the work. Section 3.2 presents the proposed idea: what are the essential attributes that modern smart factories should be equipped with. Section 3.3 introduces the idea of end-to-end industrial data processing, and explains why it is so important to the industry. The last section emphasizes the neural computing technology that is the core of developing smart manufacturing systems. Before beginning to describe the theoretical contribution of the thesis, the technical background is presented firstly.

3.1 Technical Background

This chapter introduces the technical background of this thesis, which is essential for understanding the remaining part of the work. Specifically, Chapters 4, 7, and 5 are related to ESN, Chapter 6 is related to deep representation and Appendix describes the Bayesian networks which is applied for modeling the wafer manufacturing procedure. For details of these techniques, the following references may be of interest: [Jae07] for ESN, [GBC16] for deep learning, autoencoding and deep representation, and [BG07] for Bayesian networks.

3.1.1 Supervised Learning

If we can not logically design some rules for systems, then this is the place where learning is most needed. For example, it is tough to select features to detect objects for different applications manually. Every image has a different shooting angle, different background, different lightness and so on. Learning techniques can automatically extract useful information in the form of features if we can collect enough data for training our models by adopting supervised learning algorithms.

The supervised learning problems can be defined as follows:

- X, the input data or training data.
- Y, usually called labels for classification problems, or target values for regression applications.
- D, the data distribution over the space of $X \circ Y$ which represent the data we try to collect. Regarding X and Y, possibly they are very noisy, due to the environmental issues, e.g., sensors are not stable.
- M is the total amount of samples we collected from the observation.
- $f : X \to Y$, the model we intend to figure out from provided data set.
- L is the loss function which used for learning the models fitting the training dataset with the best performance.

The learning process can be generalized as follows:

$$E_{(x,y)\sim D}[L(f(x); y)] \tag{3.1}$$

As aforementioned, the goal of learning process is to find out the models fit our date set with best performance, i.e., with the smallest errors between the predicted value and desired output (Equation 3.2).

$$\arg\min E_{(x,y)\sim D}[L(f(x); y)] \tag{3.2}$$

So that our models later can predict unseen samples with nearly the same accuracy of training results, which is determined as model generalization property. Supervised learning is the fundamental technique used in the thesis for training various of our proposed neural network-based methods.

3.1.2 Gradient Descent

Gradient descent is the technique used for minimizing the different between prediction and expect ion. Suppose we have a model $y = f(x)$, the gradient of our model is described as following (Equation 3.3):

$$f'(x) = \frac{dy}{dx} \tag{3.3}$$

The gradient specifies the direction and scale of the systems at state x. If we make any changes to the systems, the exact difference will be implemented by (Equation 3.4):

$$f(x + \epsilon) = f(x) + \epsilon f'(x) \tag{3.4}$$

When the gradient values equal to 0, then there is not enough information regarding how to update the states of the systems. The gradient descent possibly leads the final solution to the local minimum where $f(x)$ is smaller than all its neighbors but not other state points. Ideally, we should be able to find the global optimal solutions, while for practical applications, it is unnecessary, as long as the local minimum surpasses the predefined threshold.

In order to minimize the loss function, gradient descent is usually adopted for training our models, e.g., deep neural networks, which formalized as following (Equation 3.5):

$$x' = x - \epsilon \Delta_x f(x) \tag{3.5}$$

In the following part, the idea of back-propagation is introduced. The gradient descent is embedded in the back-propagation operations and used for training deep neural networks. Since there are multiple neurons in each layer, so the technique of partial derivatives is needed for separately calculating the direction and scale of the systems from different aspects.

3.1.3 Back-propagation

The main advantages of neural computing are its learning and robust modeling abilities. It is well known that with sufficient complexity, an artificial neural network can model any complex systems. In order to design a trainable neural network, the machine learning community proposes regularization techniques for optimizing neural networks structure, e.g., weight decay [KH92] and Adam [KB14]. These approaches guarantee that with less complex neural structure, users can easily model complex systems.

This part introduces the learning capabilities of artificial neural networks. It explains why neural computing techniques can create a model that dynamically captures a system's status. Figure 3.1 depicts a simple FFNN, and the back-propagation technique is introduced based on its structure. The "+1" in this Figure represents the neural bias. θ^1 and θ^2 represent the neural connections among the

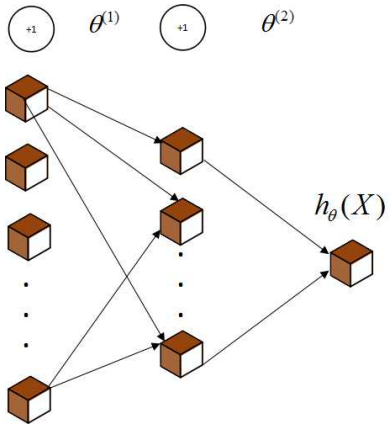


Figure 3.1: A simple artificial feed forward neural network .

different layers. $h_\theta(X)$ is the output of the last layer, which represents the hypothesis based on the collected data. a^i represents the outputs from the ith layer and $Z^{(i)}$ represents inputs from (i-1)th layer, for example $a^{(2)} = g(Z^{(2)})$, $Z^{(2)} = \theta^{(1)} \cdot a^{(1)}$: here, g is the neuron active function and could be sigmoid, tangent hyperbolic, or any other type of functions. For the input layer, $a^1 = X$. X is set as the input. To model systems in the form of collected data, parameter $\theta^{(i)}$ needs to be tuned (trained). A group of fine-tuned parameters should be able to force the predicted results (in the form of $\widetilde{h}_\theta(X)$) approximate the actual results $h_\theta(X)$ (desired outputs). In order to tun parameters θ, one firstly needs to calculate the cost function $J(\theta)$ (Equation 3.6).

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(\widetilde{h}_\theta(X) - h_\theta(X))^2 \tag{3.6}$$

where, m is the number of samples or instance in training dataset. An additional item for regularization will usually be added to Equation 3.6 to avoid the over fitting issue for training neural networks, for instance, Equation 3.6 could be reformatted to Equation 3.7:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(\widetilde{h}_\theta(X) - h_\theta(X))^2 + \frac{\lambda}{m}\sum_{i=1}^{m}(\theta_{j,k}^i)^2 \tag{3.7}$$

Here, λ is a constant value, $\theta_{j,k}^i$ is the neural weight from kth layer to jth layer. Since the predicted results $\widetilde{h}_\theta(X)$ are presented by a neural network, the target of modeling complex systems becomes tuning parameter Θ. The commonly used technique for updating Θ is developed by Rumelhart et al. [RHW86]and called the "gradient descent-based error back propagation training algorithm." The calculation chain of back propagation is presented as follows (in line with the neural structure depicted in Figure 3.1).

Step 1 Feed the training data into the neural network and save the internal calculation results of Z

and a.

Step 2 Calculate the error between prediction and actual outputs:

$$\delta^{(3)} = a^{(3)} - h_\theta(X) \tag{3.8}$$

Step 3 For the hidden layer:

$$\delta^{(2)} = (\Theta^{(2)})^T \cdot \delta^{(3)} \cdot *g'(Z^{(2)}) \tag{3.9}$$

Step 4 Accumulate the gradient:

$$\Delta^{(l)} = \Delta^{(l)} + \delta^{(l+1)}(a^{(l)})^T \tag{3.10}$$

Step 5 Calculate the gradient of the neural network (without regularization):

$$\frac{\partial J(\theta)}{\partial \theta_{ij}^{(l)}} = D_{ij}^{(l)} = \frac{1}{m}\Delta_{ij}^{(l)} \tag{3.11}$$

These 5 steps clearly show how to manipulate each neural connection among the neurons and achieve an accurate modeling performance. As the neural network is an entirely data-driven method, if there are dynamic changes inside the systems, the neural network will update itself again, based on back propagation.

3.1.4 Training Algorithms

How to quickly search the best parameters so that the models can fit the provided data set accurately motives the research work developing more and more efficient training algorithms, e.g., stochastic gradient descent, Momentum, Nesterov accelerated gradient, Adagrad, Adadelta, RMSprop, Adam and so on. To clearly show the relations among different training algorithms, we first present the relation map which indicates how the different algorithms related others (Figure 3.2). Regarding the details of these training algorithms, readers can refer to relevant references. Stochastic gradient descent, Momentum, and Adam are the essential parts of the thesis adopted in most of our research activities. Although other algorithms are not used in our works, they are maybe helpful for future case studies. Therefore, the abstract implementation of these listed algorithms is introduced as follows:

Stochastic gradient descent (SGD) [MMN18] performances parameters update for each training sample pair $(x^{(i)}, y^{(i)})$, where i is the index of samples. The steps of implementing this training algorithm is presented algorithm 3.1. SGD does very efficient parameter update, because it updates the parameter on every single sample, but this also causes heavy fluctuation to the loss function. Mini-batch gradient descent is then proposed for reducing such fluctuation. The idea is to randomly

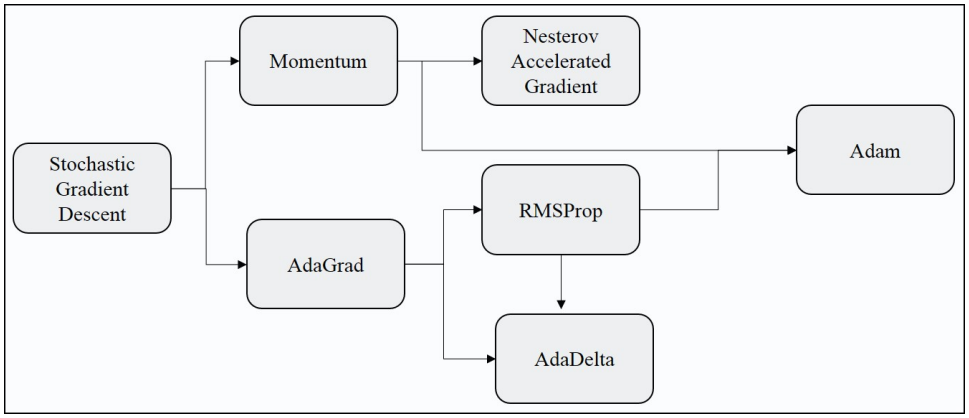


Figure 3.2: Relation among different training algorithms.

Algorithm 3.1: SGD

Data: data set for training / validation / testing
Result: parameter w
Initialize: parameter w and learning rate η;
while *approximate minimum is obtained* **do**
  for $sample = 1, 2, ..., m$ **do**
    $w = w - \eta \Delta L(x)$;
  end
end

select small batch of samples and update the parameter after each batch, instead of every sample. The parameter update is formalized as equation 3.12:

$$w = w - \eta \cdot \Delta_w L(w; x^{(i:i+n)}; y^{(i:i+n)}) \tag{3.12}$$

Challenges SGD and mini-batch SGD are facing now are: (1) it is hard to choose the learning rate η. (2) During the training process, η should be updated simultaneously, as the gradient of loss function gets changed by all the time. (3) Learning rate should be different regarding different parameters. (4) For most the applications, the loss functions are non-convex. Try to avoid trapped in the local minimum is essential.

Momentum [Qia99] is the algorithm tries to accelerate SGD algorithm in the sense of gradient direction searching and oscillation reduction. The idea is implemented in algorithm 3.2. Where t is the index of the steps for parameter updating. Essentially, momentum algorithm borrows the idea from physic communicate. The idea is like pushing a ball down a hill. The ball accumulates momentum during the rolling down process. The scenario is similar to conducting gradient descent, i.e., we try to reduce the error between prediction and expect step by step. With the help of momentum, we can

Algorithm 3.2: Mini-batch based Momentum

Data: data set for training / validation / testing
Result: parameter w
Initialize: parameter w and learning rate η;
while *approximate minimum is obtained* **do**
  for $batch = 1, 2, ..., n$ **do**
    $v_t = \gamma v_(t-1) + \eta \Delta_w L(w)$;
    $w = w - v_t$;
  end
end

dynamically adjust the learning rate and verifying the loss function gradient. Hence, the learning process is accelerated.

Nesterov accelerated gradient algorithm aims at precisely guiding the update direction based on loss function gradient, which is implemented by looking ahead by calculating the gradient, i.e., what will be the approximate gradient of the parameter in the next steps. This procedure is defined by algorithm 3.3.

Algorithm 3.3: Nesterov accelerated gradient

Data: data set for training / validation / testing
Result: parameter w
Initialize: parameter w and learning rate η;
while *approximate minimum is obtained* **do**
  for $batch = 1, 2, ..., n$ **do**
    $v_t = \gamma v_(t-1) + \eta \Delta_w L(w - \gamma v_{t-1})$;
    $w = w - v_t$;
  end
end

Adagrad [DHS11] tries to solve the learning rate update problem of SGD. The core idea is performing a small change for the parameters associated with frequently occurring features; The big updates for the ones associated with infrequent features. For brevity, the implementation of Adagrad is presented in algorithm 3.4. where, g is the gradient of each upate. G is the sum of the squares of the gradients . ϵ is a smoothing parameter avoiding the division by zero.

Adadelta [Zei12] extends the work of Adagrad to reduce its aggressive, monotonically decreasing learning rate, i.e., instead of accumulating all past gradient, Adadelta only take recent gradient into account (algorithm 3.5).

Algorithm 3.4: Adagrad

Data: data set for training / validation / testing
Result: parameter w
Initialize: parameter w and learning rate η;
while *approximate minimum is obtained* **do**
  for $batch = 1, 2, ..., n$ **do**
    $g_{t,i} = \Delta_w L(w_t, i)$;
    $G_{t,i} = G_{t,i} + g_{t,i} \odot g_{t,i}$;
    $w_{t+1,i} = w_{t,i} - \frac{\eta}{\sqrt{G_{t,i}+\epsilon}} \odot g_{t,i}$;
  end
end

Algorithm 3.5: Adadelta

Data: data set for training / validation / testing
Result: parameter w
Initialize: parameter w and learning rate η;
while *approximate minimum is obtained* **do**
  for $batch = 1, 2, ..., n$ **do**
    $g_t = \Delta_w L(w_t)$;
    $G_t = \gamma G_t + (1-\gamma) g_t \odot g_t$;
    $\Delta w_t = -\frac{\sqrt{\Delta_{t-1}+\epsilon}}{\sqrt{G_t+\epsilon}} \odot g_t$
    $w_{t+1,i} = w_{t,i} + \Delta w_t$;
    $\Delta_t = \gamma + (1-)\Delta w_t \odot \Delta w_t$
  end
end

RMSprop is introduced by Geoff Hinton. It also aims at solving the problem of Adagrad's radically diminishing learning rates. The implementation is presented as algorithm 3.6.

Adam [KB14] is one of the most powerful adaptive learning rate training algorithm. It keeps the idea of an exponentially decaying average of past squared gradient which is similar to Adadelta and RMSProp. Adam also has the part that similar to momentum algorithm, i.e., exponentially decaying average of past gradients. The implementation of Adam is presented in algorithm 3.7. where, m is the momentum. and β are the constant parameters.

Algorithm 3.6: RMSprop

Data: data set for training / validation / testing
Result: parameter w
Initialize: parameter w and learning rate η;
while *approximate minimum is obtained* **do**
  for $batch = 1, 2, ..., n$ **do**
    $g_t = \Delta_w L(w_t)$;
    $G_t = \gamma G_t + (1 - \gamma) g_t \odot g_t$;
    $w_{t+1} = w_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t$;
  end
end

Algorithm 3.7: Adam

Data: data set for training / validation / testing
Result: parameter w, , β, learning rate η
Initialize: parameter w and learning rate η;
while *approximate minimum is obtained* **do**
  for $batch = 1, 2, ..., n$ **do**
    $g_t = \Delta_w L(w_t)$;
    $m_t = \beta m_{t-1} + (1 - \beta) g_t$;
    $G_t = \gamma G_t + (1 - \gamma) g_t \odot g_t$;
    $\alpha = \eta \frac{\sqrt{1 - ^t}}{1 - \beta^t}$
    $w_{t+1} = w_t - \frac{m_t}{\sqrt{G_t + \epsilon}}$;
  end
end

3.1.5 Feedforward Neural Networks

FFNN is one of the most simple artificial neural networks and extremely important for understanding other advanced neural networks which are constituted by combing different FFNN blocks. The information flows through the previous layer directly to the next layer without any cycle (see figure 3.1). The mathematical expression of FFNN is $f(x) = f^n(f^{n-1}(f^{n-2}(...f^1(x)...)))$, where n is the number of layers, x is the input fed to the neural networks. A FFNN is constituted by the input layer, which used for receiving input data from users, followed by hidden layers and the output layer. In each layer, we can place a different amount of nodes there with specific functions (called activation function) and these nodes called neurons, akin to biology nerve systems. For example, design a FFNN for object classification, the problem can be defined as $y = f(x; w)$, where x is the images, and y is the labels indicates in which categories objects in image x belong to. In order to obtain the

best classifiers, we need to adjust the parameter w in the sense of neural network training.

The number of neurons placed in each layer decides the width of neural networks. The number of layers for the entire neural networks defines the depth. When users feed data into the neural network, different activation functions start to transform the input received from the neurons in the previous layer. Adding more layers to hidden layers forms the so-called the deep feedforward neural networks. Generally, the idea for these hidden layers is taken as a feature representation which maps the input data into another space. The last layer is considered as functioning the feature combination and generating the desired output.

3.1.6 Recurrent Neural Networks

RNNs are designed for processing time-series data stream by adding feedback connections from output to input, any hidden layers, hidden layer to hidden layer or even self-cycle connection. We can design a variety of RNNs with different kinds of feedback loops. It hardly defines the unique characters of RNNs, while there are some important common patterns that RNNs usually have:

- A RNN produces an output at each time step and feedback loop added to hidden neurons.
- A RNN generates an output at every time step, and feedback connection directly links to its input.
- Feedback connections show up between hidden neurons. The connections read an entire sequence and only produce one output.

The dynamics of RNNs can be described by the deterministic transitions, which flow from previous to current hidden states [ZSV14]:

$$RNN : h_t^{l-1}, h_{t-1}^l \rightarrow h_t^l$$

where h is the state of a hidden neuron, l is the index of the neural layer. For a classical RNN, the hidden sate h is defined as:

$$h_t^l = f(T_{n,n} h_t^{l-1} + T_{n,n} h_{t-1}^l)$$

The possible activation functions of f are sigmoid or hyperbolic tangent function.

As a summary, different architectures of RNNs are list as following:

- Fully recurrent. A FFNN adds self-cycle to every neuron with specific time delay.
- Eleman networks. A three layers FFNN neural network with additional content neurons. The number of the context neurons are the same as the hidden layer. The recurrent connections occur between ith hidden neuron and the corresponding context neuron.
- Hopfield. A FFNN with symmetric connections between the neurons within the same layer.
- Echo-state. A three-layer neural network. The hidden layer of the neural network is designed with randomly connected neurons. The neural weights of input - hidden layer and hidden - hidden layer are fixed after initialization. The only trainable layer is the readout layer (output).
- Independent RNN. It is mainly designed for addressing the gradient vanishing issue of RNNs. Each neuron only accepts its context. Hence, every neuron is independent of others.
- Recursive. applying the same set of weights recursively over a differential graph-like structure by traversing the structure in topological order.
- Second order RNNs. Using higher order weights instead of normal weights. Additionally, it allows states to do product operation.
- Long short-term memory. An LSTM cell is constituted by four different gates with a specific function. It is a deep learning system designed for tackling the gradient vanishing issue.
- Gated recurrent unit. Similar to LSTM RNNs without output gate.
- Bi-directional. It predicts a sequential data stream by adopting historical and future contexts.
- Continuous-time. Ordinary differential equations are applied to predict the effects of coming spiking signals.
- Hierarchical. Decomposing complex RNNs into subprograms.
- Recurrent multilayer perceptron network. Constituted by cascaded sub ffnns.
- Neural Turing machines. Add attention mechanism to RNNs.

In the common practice, gradient descent based back-propagation training algorithms are used for searching optimal or sub-optimal weights for neural networks. While it is always challenging to conduct derivative operators on the complex recurrent connections, which is merely computing expensive and hard to guarantee the convergence of training results. That's why part of work presented in the thesis looks to a very special recurrent neural networks, i.e., echo-state network. It

only applies linear combination for the last layer, and only the last layer is trainable. The remaining weights are all kept fixed during the training process. The details of an echo-state network are introduced in the following section and chapters.

3.1.7 Echo-state Networks

Although RNNs are powerful tools in modeling time-series data, they are naturally difficult to be trained, because of the complex feedback connection among different layers. At the year of 1985, Rumelhart et al. [RHW85] introduced the idea of error back-propagation (BP) which became the stander training algorithm for FFNNs these days. Later on, BP-based methods have also been extended to RNNs [WZ89, Wer90]. Since then, RNNs have been widely applied to solve amount of practical problems with various RNNs variants, e.g., long short-term memory neural networks (LSTM) [HS97] are successfully applied for speech recognition and achieved the state-of-the-art performance. One of the limitations of BP for RNNs is the non-converging issue. Besides, even when the proposed RNN models do converge, but the entire training process is computing expensive. To tackle such issue, Jaeger et al. [Jae07] proposed the so-called echo state networks. ESNs provided architecture and supervised learning principle for RNN. The main ideas are (1) to drive a random, large, and fixed RNNs with the input signal, thereby inducing in each neuron within this "reservoir" network a nonlinear response signal, and (2) to combine the desired output signal with a trainable linear combination of all of these response signals [Jae07]. ESNs are similar to liquid state machines, which were developed independently from and simultaneously with ESNs by Wolfgang Maass [NMM02], from the perspective of neural network structure and training performance. The term "reservoir" here replaces the "hidden layer" of the traditional neural network (i.e., the gray part in Figure 3.3), due to the random neural connection in one hidden layer. Additionally, "reservoir" is also used to represent the most important character, i.e., modeling the evolving trend in external systems by manipulating the internal states of ESN.

The Basic Model of ESNs

ESNs are suitable for modeling time-series data or sequences. Given a input sequence $u(n)$ and desired output Y, the learning task is to fit a model which minimizes the difference (error) between the predicted and desired output. The measurement of calculating the difference is typically a Mean-Square Error. The normalization and the square root parts of MSE is mainly for us better understand the trained models. Following equations show how to govern the dynamic states of ESNs with leaky-integrated discrete-time continuous-value units.

$$x(n+1) = f(W^{in}[u(n+1)] + Wx(n) + W^{fb}y(n)) \tag{3.13}$$

$$y(n) = g(W^{out} z(n)) \tag{3.14}$$

$$z(n) = [x(n); u(n)] \tag{3.15}$$

Where $x(n)$ is the reservoir state of ESN at time n; $u(n)$ is the inputs at time n; W^{in} is the randomly generated weights between input layer and reservoir; W and W^{out} are the reservoir weights and weights of readout layer, respectively. $y(n)$ is the prediction and $y^{target}(n)$ is the desired output. E is deviation between $y(n)$ and $y^{target}(n)$. Figure 3.3 does not show the weights W^{fb} which stands for the feedback weights from readout layer back to reservoir.

Steps of applying ESNs

In order to apply an ESN model for a practical application, there are three main steps need to be followed:

Step 1. Randomly generate an RNN. Some key notes for designing the structure of an ESN are as follows:

[Note 1] One can adopt any neural model as the computing unit.
[Note 2] The size of the neural network depends on its practical application.
[Note 3] Connections between inputs and reservoir are in an all-to-all fashion.
[Note 4] The output or readout layer is the only trainable layer. The feedback from the output layer to the reservoir layer is dependent on the circumstances of the real application.

Step 2. Harvest reservoir states. The basic discrete-time, sigmoid-unit ESN with N reservoir units, K inputs, and L outputs are governed by the state update equations. The general structure of ESN is depicted in Figure 3.3.

Step 3. Learning equations. The training (or learning process) only occurs in the readout layer. The primary training method is based on gradient searching and linearly combining the internal states of the reservoir (Equation 3.16).

$$W^{out} = (S^{\dagger} D)^T \tag{3.16}$$

where, S is state collection matrix constituted with the size $N \cdot K$. $\dagger$ is the pseudo-inverse operation. D is the sate collection matrix of K and L. In the AI community, researchers are continually working on ESN and have proposed many effective algorithms: for example, the works [XLL17, SWP16].

Hyper-parameters of ESNs

Implementing a RNN model is easy while there are some factors need users to pay more attention to them so that the designed model finally work. These factors are "Size of the Reservoir," i.e., the

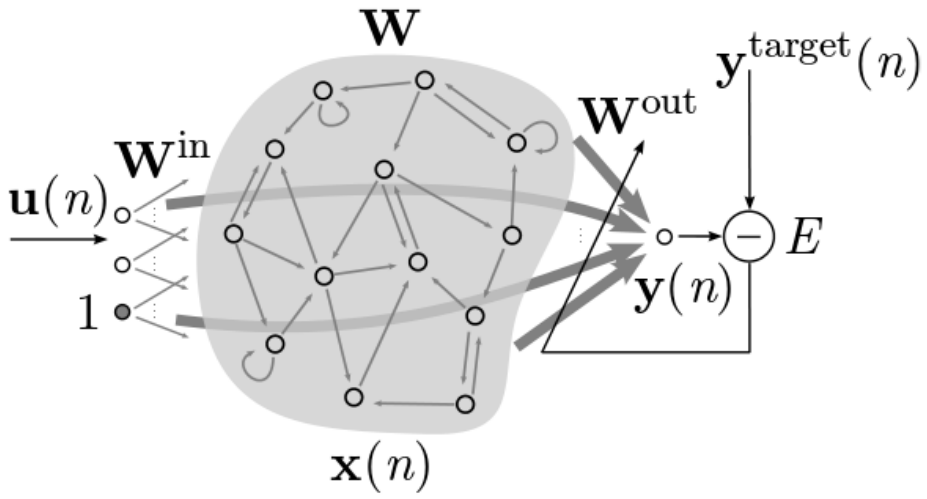


Figure 3.3: General structure of ESN.

size of the only hidden layer, which essentially defines the performance of the model, as the general wisdom saying that the bigger reservoir, the better modeling performance. While considering the computing expense, sometimes we can also add the regularization to a ESN, for instance, "Sparsity of the reservoir". It has two advantages which quite obviously are lower the computing cost and prevent the over-fitting issue.

The "Distribution of Nonzero Elements" is another important factor related to the initialized values of neural networks weights. The possible probability distribution adopted for generating the values could be uniform distribution and Gaussian distribution. "Spectral Radius" is the hyper-parameter influences the so-called "echo-state" property which indicates that the fading memory of ESNs. This hyper-parameter is the maximal absolute eigenvalue of generated reservoir weights. Generally, the "Spectral Radius" should be smaller than 1 to guarantee the "echo-state" property. "Input Scaling" is the factor used for scaling the weights of the input layer. It mainly defined by which type of probability distribution adopted for generating the reservoir weights. Finally, the "Leaking Rate" is the factor could be taken as the parameter which controls the updating speed of reservoir states.

Effectively selecting these hyper-parameters are tough. Many works have been conducted to investigate the automation ways of parameters-searching [JLPS07, BBBK11]. In the thesis, we also presented one of the possible solutions (details introduced in Chapter 5).

3.1.8 Deep Representation

Figure 3.4 shows an example of how the information distributed in the data set. In the depicted figure, two features are dominating the image content. Figuring out the main factors (features) which independently contributes partial information of original data is our main target for knowledge representation. Deep representation is essentially making use of the learning ability of deep neural

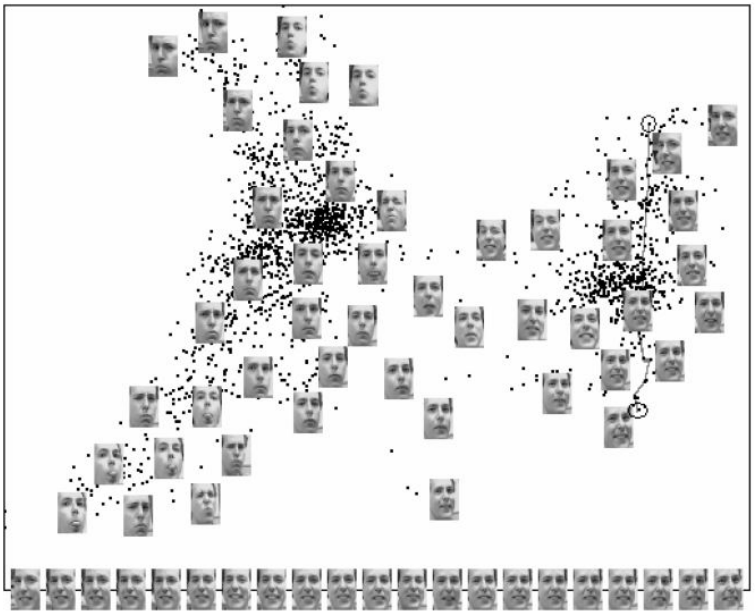


Figure 3.4: Two dimensional embedding of Laplacianfaces [HYH+05].

networks estimate the density of provided data set, i.e., learn a probability distribution for the observed data V, and the learned probability distribution is described by latent variables h. We take h as the deep representation of our data set. The variables h reserve the most important information of the original data set with much less noise (ideally). There are many proposed deep neural networks could be used for feature extraction or generating a neural representation for our data set, e.g., convolutional neural networks for image feature extraction, LSTMs for sequential data processing and so forth, but this thesis focuses on introducing the general idea of autoencoder, and the application scenarios presented in the thesis are mainly from the manufacturing sector. The data collected from production lines is usually with high dimensions. Hence, we need effective ways to shrink the sensory data and keep most of the important information at the same time.

General Framework of Autoencoder

In 2006, Geoff Hinton, Osindero and Teh [HOT06] presented a many-layered FFNN which can be effectively pre-trained layer-by-layer: regarding each layer in turn as an unsupervised restricted Boltzmann machine, and then fine-tuning it using a supervised backpropagation algorithm [Hin07]. This publication is the first time to prove that with a multi-layer (deep) neural network, systems can essentially learn the abstract presentation of inputs. Autoencoder, as one of the simplest and most

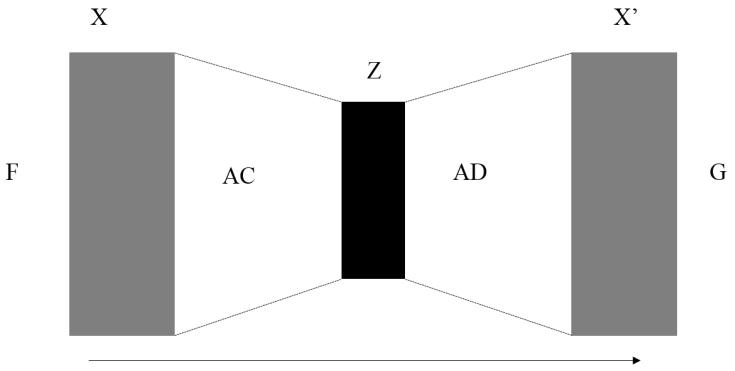


Figure 3.5: General structure of autoencoder.

representative deep neural networks, is fundamental to this thesis. Its basic neural network structure is depicted in Figure 3.5: X is the inputs data, X' is the expected outputs Y, i.e., $Y = X'$. Z is the outputs of the middle layer of the entire neural network: the "deep representation" of inputs. Accordingly, the training target is to reduce the deviation between inputs X and outputs X', i.e., teaching the neural network to copy its inputs. Its loss function is defined as follows (Equation 3.17):

$$L(x, g(f(x))) \tag{3.17}$$

Where, L is a loss function, $g(\cdot)$ is the decoder AD, $f(\cdot)$ is the coder AC.

A fairly general framework of autoencoder (Figure 3.5) is defined following factors [Bal12]:

- $\mathbb{F}$ and $\mathbb{G}$ are sets.
- X, X' and Z are positive integers. Generally, $X == X'$ and $Z < X$.
- AC is a set of functions from $\mathbb{F}^{\mathbb{X}}$ to $\mathbb{G}^{\mathbb{Z}}$, generally called as coder, in latter introduction we use f for formal representation.
- AD is a set of functions from $\mathbb{G}^{\mathbb{Z}}$ to F^X, generally called as decoder, latter introduction uses symbol g.

- $X = \{x_1, x_2, ..., x_m\}$ is a training data set with m samples in $\mathbb{F}^{\mathbb{X}}$. Similarly, the corresponding targets are represented as $Y = \{y_1, y_2, ..., y_m\}$ in $\mathbb{F}^{\mathbb{X}}$.
- Δ here is used for calculating the errors between the predicted X' and the input X (e.g., L_p norm).

The map functions AC and AD are the activiation functions of autoencoder which transforms an input $x_i \in \mathbb{F}^{\mathbb{X}}$ into an output vector $AC \circ AD \in \mathbb{F}^{\mathbb{X}}$. The core idea of autoencoder is to train a neural network to copy its inputs by minimizing the overall difference between input and predicted output:

$$min \sum (L(x_i, g(f(x_i))))$$

Classes of Autoencoder

Autoencoding is essentially a data compression technique which constituted by two main parts, i.e., coder and decoder. Existing works have proposed many powerful autoeconders, which can be classified into different categories based on: (1) the way of how to place coder and decoder. (2) the way of how to train neural networks. (3) the pre-processing of input.

Saprse Autoencoders is a straightforward idea of how to guarantee the neural network will finally learn some useful information, which is done by adding a sparsity penalty $\Omega(Z)$ on the layer shrinking the input size, e.g., the layer Z (Figure 3.5). This type of autoencoders are adopted in Chapter 6. The loss function of sparse autoencoders is formalized as follows (Equation 3.18):

$$L(x, g(f(x))) + \Omega(Z) \tag{3.18}$$

Denoising Autoencoders. Instead of adding penalty parameter to the loss function, the denoising autoencoders will first add some noise to the original data set. Then the autoencoder leans to reconstruct the original dataset with corrupted input. The con responding loss function is (Equation 3.19):

$$L(x, g(f((\tilde{x})))) \tag{3.19}$$

where, $\tilde{x}$ is the corrupted input x. The denoising autoencoder must learn to remove the added noise.

Variational Autoencoders (VAEs) are essentially generative model. Rather than only learning to copy the input from output, VAEs also can generate the unseen data set. This ability is gained by forcing the neural network to learn the probability distribution of the training data set. Hence, the loss function is formalized as following (Equation 3.20):

$$L_i(\theta, \phi) = -E_{z \backsim f_\theta(z|x_i)}[\log g_\phi(x_i|z)] + KL(f_\theta(z|x_i)||g(z)) \tag{3.20}$$

where, θ is the parameter of coder neural network, ϕ is the parameter of decoder. KL is the Kullback-Leibler divergence used for measuring the difference between the probability distribution of prediction output and the desired output.

Convolutional Autoencoders (CAEs) are originally designed for solving computer vision problems, e.g., object detection and classification. Compared with the classic autoencoders, the difference is convolutional layers are adapted for replacing the simple feedforward layers. In the common practice, pooling operation is also used after convolutional layer(s). The loss function used for training the neural network could be cross-entropy (Equation 3.21) or MSE (Equation 3.22).

$$L = y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \tag{3.21}$$

where y is the target, $\hat{y}$ is the predicted output.

$$L = ||\hat{y}_i - y_i||_2^2 \tag{3.22}$$

Sequence-to-sequence Autoencoders are proposed for modeling sequential data points, which can capture the temporal structure, e.g., LSTMs have achieved the state-of-the-art performance for voice recognition. LSTM-based autoencoders use LSTM neural networks as the encoder that transforms the input into a single vector. Then, we repeat this vector several times (depends on the time-steps in the output sequence). Finally, an LSTM decoder reconstructs this continuous sequence into the target sequence. Other possible time-series neural networks could also be adopted encoding and decoding temporal data points.

Many different types of autoencoders have been introduced. Generally, they are used for classification, regression or generating new samples. We have adopted some them, the others are considered as the basis of the work we plan to do in future work.

Deep representation: a MNIST example

In order to intuitively understand how the autoencoder works, a handy digit recognition example from the Kreas help manual is introduced. The task is to detect ten different handy digits from 0 to 9. The "original inputs" in Figure 3.7 are the input data fed into the encoder, "reconstructed inputs" are the outputs of the decoder. The desired out is not the outputs from the decoder, but the outputs from the coder, i.e., the white-black blocks in Figure 3.5. To give a general idea of how are the outputs from the decoder looks like, the Kreas help manual shows one of the digital number codes

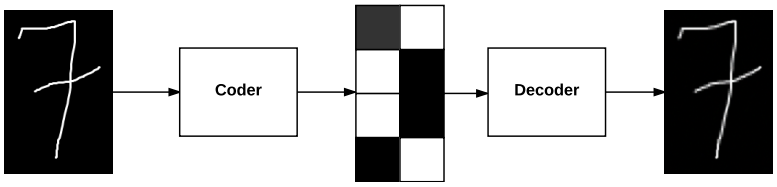


Figure 3.6: Principle of Autoencoder.

Figure 3.7: Reconstruction results from the convolutional autoencoder.

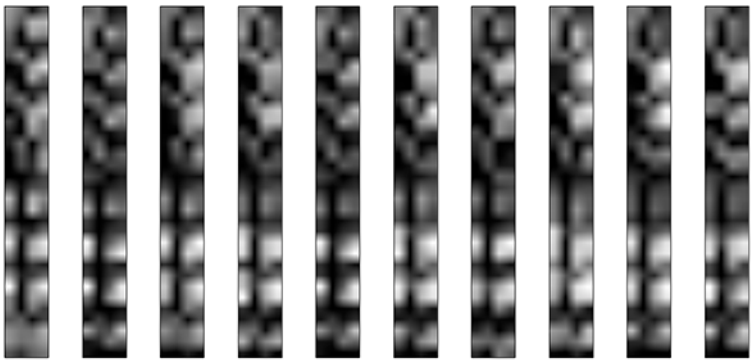


Figure 3.8: Deep representation from the coder Z.

(Figure 3.8). From a data science perspective, an image is one type of signal, similar to the signals generated by different types of industrial sensor. In this thesis, the research goals are to compress the high dimension inputs, remove the noise, make up the corrupted data, and extract the most useful features from the sensory information. All of these goals could be achieved by applying various autoencoders, for example, the variant autoencoder or convolutional autoencoder, as the middle layer of the autoencoder can lower the original inputs into much fewer dimensions represented by the outputs of the hidden layer, while keeping most of the significant information. Chapter 6 will present the high dimension industrial data compress and modeling platform designed based on the theory of autoencoder.

3.1.9 Bayesian Networks and wafer industry

This subsection intends to mention the work related to the Bayesian network. It introduces the abstract process of producing wafers. Part of the content discusses the work we have done with Bayesian network regarding wafer quality control. The details of introducing the wafer manufacturing process are out of the scope of the thesis. Readers can refer to the works [VSH02, LHV+06]. The details of this part of work are presented in Appendix.

Neural computing or neural network-based methods are powerful tools for modeling issues, but this does not mean that all problems could be solved easily. One of the goals of this thesis is to

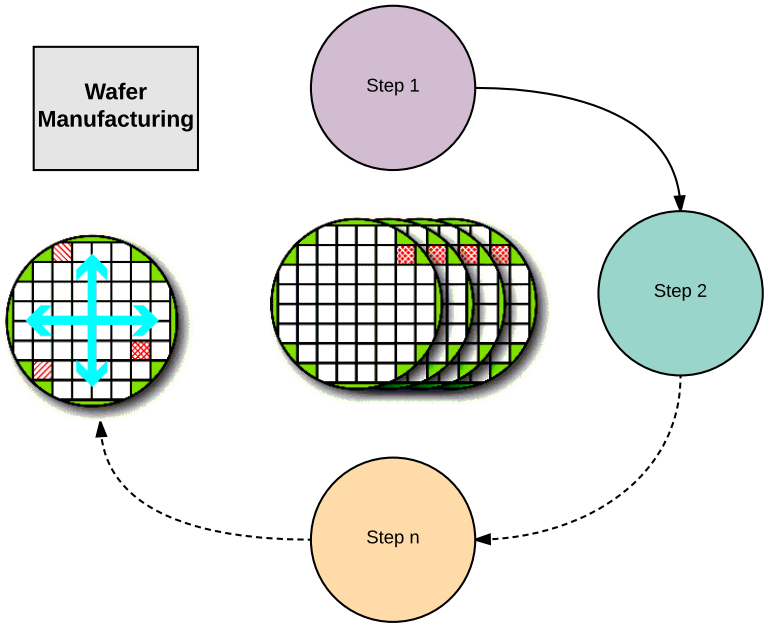


Figure 3.9: General wafer manufacturing process.

model and monitor manufacturing processes. The processes studied include bearing machine and milling (drilling) machine. Neural computing-based methods work extremely well in all but one industry field: that of semiconductors. Due to the complex manufacturing procedure and ultra clean production environment requirements, hundreds of production steps are executed in a giant black box, which means that only limited information can be accessed. Low-quality chips cause billions of loss every year. Thus the monitoring of the semiconductor manufacturing process is essential. One alternative solution is predicting the future status of wafers during the manufacturing process.

A Bayesian network is a directed acyclic graphical model constituted by various random variables and conditional dependencies. Generally, a Bayesian network can be defined by $B = (G, \theta)$:

- **B** is a Bayesian network.
- **G** is a directed acyclic graph. It is constitute by nodes which represent various random variables x_i i.e., the ith node. G encodes independence assumptions by which each variable x_i is independent of its non-descendants given its parents in G.
- θ is a set of parameters of B.

In the real application, we actually use Bayesian rule to do causal relationship reasoning, which is defined as:

$$P(X = x_i|Y) = \frac{P(Y|X = x_i)P(X = x_i)}{\sum_{k=1}^{n_X} P(Y|X = x_k)P(X = x_k)} \tag{3.23}$$

where, $P(X = x_i)$ is prior probability. $P(X = x_i|Y)$ is the so called posterior probability. Formula 3.23 indicates that the post probability is related to every parent connection of $P(Y|X = x_k)$, which is impossible and unnecessary for real problems. For example, if we want to predict the wafer quality shifting problem, we may just need to know recent 2 hours historical data from production line, instead of the entire past states. Therefore, the typical Bayesian model i.e., Hidden Markov , is more practical for reasoning the relation among different events. A Hidden Markov Model can be defined by following parameters:

- X, the states of the systems.
- V, possible observation from the systems.
- A, states transition probabilities.
- B, output probabilities.

The graphic presentation of such probability relation is depicted in figure 3.10. If we unfold a Hidden Markov Model in the time-series fashion, it could be described as figure 3.11. Therefore, as a complete definition of Hidden Markov Model, following parameters are essential:

- N, number of states x_n and the state at time t is Z_t.
- M, number of observation V_m and the observation at time t is O_t.
- The state transition probability matrix $P = \{P_{ij}\}$, where

$$P_{ij} = P(Z_{t+1} = x_j|Z_t = x_i), 1 \leq i, j \leq N \tag{3.24}$$

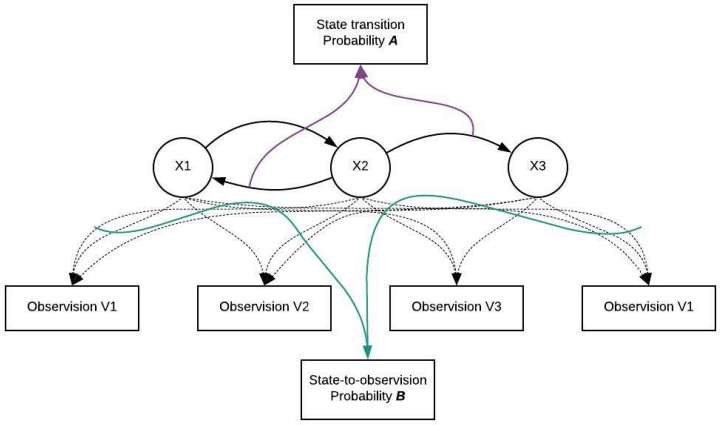


Figure 3.10: Example: probabilistic relation of a hidden Markov model.

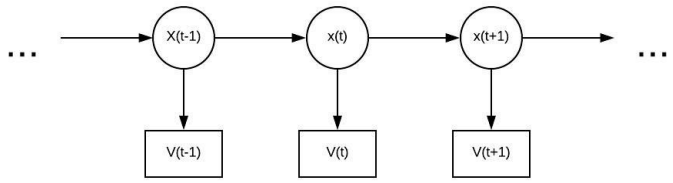


Figure 3.11: unfolded probabilistic relation of a hidden Markov model.

- the observation probability in state x_n, $B = b_j\{k\}$, where

$$b_j(k) = P(O_t = V_k | Z_t = x_j), 1 \leq j \leq N, 1 \leq k \leq M \tag{3.25}$$

- prior probability distribution $\pi = \{\pi_i\}$ of X where

$$\pi_i = P(Z_0 = x_i), 1 \leq i \leq N \tag{3.26}$$

Once collecting enough data set, we are ready to train the Bayesian network. The maximum likelihood is usually adopted as the loss function for estimating of the parameters of a Hidden Markov Model by given a set of output sequences. While there is no method can help us find the global maximum likelihood. Baum-Welch algorithm is the common practice for searching the local maximum likelihood of a given model. The implementation of such a training algorithm is presented as following (algorithm 3.8):

Generally, Bayesian networks can be used for reasoning, e.g., a Bayesian network can represent the probable relationships between diseases and symptoms. This thesis also treats the cause of the low

Algorithm 3.8: Baum-Welch algorithm

Data: data set for training / validation / testing
Result: parameter θ
Initialize: $\theta = (A, B, \pi)$ with random initial conditions;

Forward processing;

Let $\alpha_i(t) = P(V_1 = O_1, ..., V_t = O_t, Z_t = i|\theta)$;
Following code is done recursively ;

- ${}_i(1) = \pi_i b_i(O_i)$;
- $\alpha_i(t+1) = b_i(O_{t+1} \sum_{j=1}^{N} \alpha_j(t))\alpha_{ji}$;

Backward processing;

Let $\beta_(t) = P(V_{t+1} = O_{t+1}, ..., V_T = O_T|Z_t = i, \theta)$;

- $\beta_1(T) = 1$;
- $\beta_i(t) = \sum_{j=1}^{N} \beta_j(t+1)\alpha_{ij}\beta_j(O_{t+1})$;

Updating;

According to Bayesian rule, the temporary variables is calculated as:
${}_i(t) = P(X_t = i|V, \theta) = \frac{{}_i(t)\beta_i(t)}{\sum_{j=1}^{N} {}_j(t)\beta_j(t)}$;
$\xi_{ij}(t) = \frac{\alpha_i(t)\alpha_{ij}\beta_j(t+1)b_j(V_{t+1})}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i(t)\alpha_{ij}\beta_j(t+1)b_j(O_{t+1})}$;
$\pi_i^* = \gamma_i(1)$;
$\alpha_{ij}^* = \frac{\sum_{t=1}^{T-1}\xi_{ij}(t)}{\sum_{t-1}^{T-1}\gamma_i(t)}$;
$b_i^*(v_k) = \frac{\sum_{t=1}^{T} 1_{O_t=V_k}\gamma_i(t)}{\sum_{t=1}^{T}\gamma_i(t)}$;
where, $1_{O_t=V_k} = 1$ if $y_t = V_K$, otherwise $1_{O_t=V_k} = 0$;

quality or fault as the disease, and models various fault dependencies. The learned Bayesian network model is then used to monitor the wafer manufacturing process. Figure 3.9 depicts the manufacturing procedure of wafer. For the first step, materials are fed into the production line. This is followed by step 2, ..., step n. This procedure is strictly unseen in the production process. At the last step, the generated wafer is tested and checked to see whether each function works as designed in the chip. Concerning monitoring, the technical task is applying the Bayesian network model to this manufacturing procedure, and then monitoring the status of the wafer production line and predicting the future status. In Appendix, a novel Bayesian network is designed to show some experimental results of monitoring a specific wafer production procedure.

3.2 Theoretical Contribution: Smart Attributes for CPPS

A cyber-physical production system, or Industry 4.0s, is an abstract concept for developing more advanced modern manufacturing companies. Within the concept of CPPS, there are three main components: communication ("cyber"), physical aspects ("production systems"), and relevant subsystems. In order to reduce the repetitive, dangerous, and low-value tasks for human workers and make the system more robust in its running environment, for example, predicting the failure time of components, automatically upgrading the systems is urgently required. Smart controllers are essential for integrating the "cyber" with the "physical". Lee et al. [LBK15] propose a cyber-physical systems architecture for Industry 4.0s-based manufacturing systems. In this framework, modern factories can be compared with Industry 4.0s-based smart factories, and it is proposed that "self-X" (self-aware, self-predict, self-compare, self-compare, self-maintain, self-organize, and so on) are smart attributes for CPPS. The abstract structure is depicted below in Figure 3.12. Regarding the entire structure of CPS, Lee et al. believe that different components of different layers should collaborate and independently handle the unexpected events during the running phases. These characters conclude the significant properties of CPPS. Although this framework proposes a generic model for designing smart factories, one of the weaknesses is that it does not identify the principle of the technical solution. This thesis, therefore, proposes that learning is one of the most promising technical principles for implementing CPPS. The proposed extended framework is depicted in Figure 3.13. The 5C architecture of Lee et al. is adopted, with the learning engine added as the technical basis for the other parts. The learning engine contains a learning library, production systems models, and meta-learning library.

The research community began exploring the smart attributes of production systems decades ago, and generally considers self-X as the basis for this. With the development of AI technologies, research attention is increasingly attracted to machine learning techniques. The ability to quickly modeling

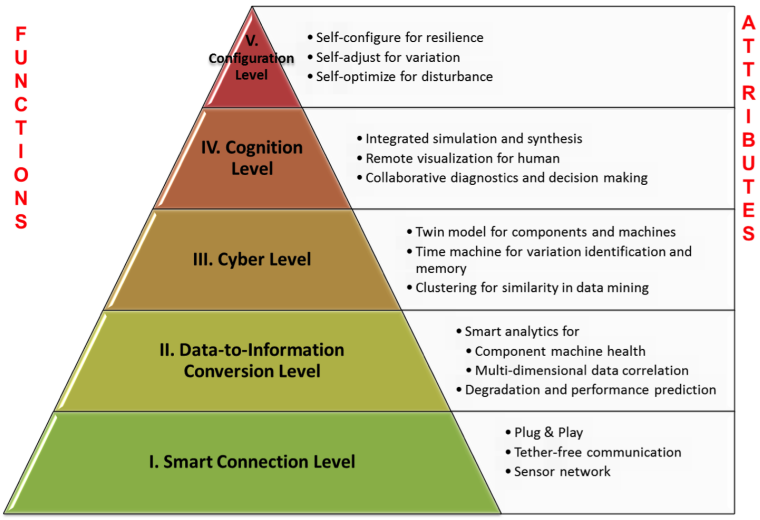


Figure 3.12: 5C architecture for implementation of Cyber-Physical System [LBK15].

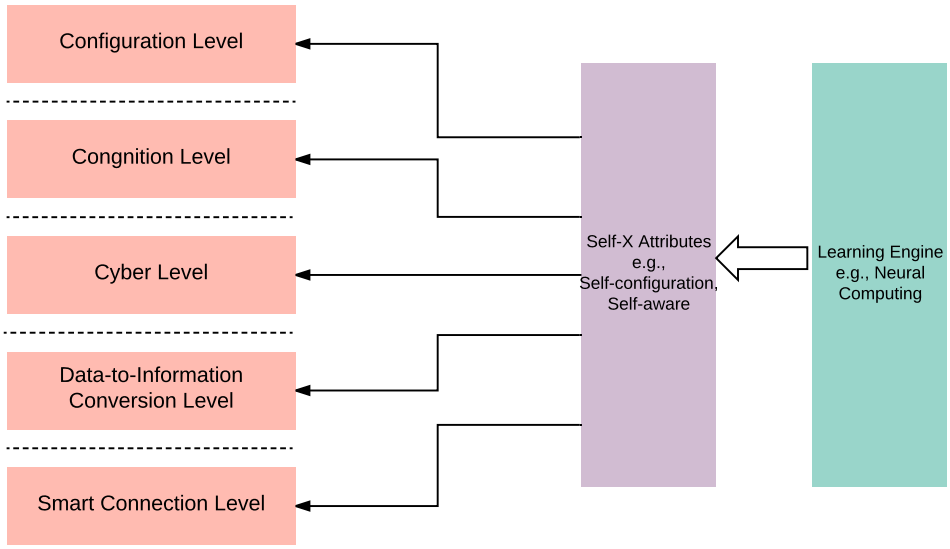


Figure 3.13: Smart Attributes for CPPS and its technical principle (Learning).

complex systems is one of the most significant advantages of machine learning. Neural computing as one of the majority components of machine learning is even more advanced, which allows users to model systems as a black box and generate a model with strong, robust properties. For instance, deep neural network models can easily identify faces among different photos. The general idea of deep neural networks is to represent inputs as neural weights and re-generate a type of feature. Those features can then be easily modeled by classical machine learning methods.

Although the aforementioned deep learning or machine learning seems promising, the problem is how to make a learning engine smart, which is the main reason for the adoption here of the meta-learning library as one of the essential components. Meta-learning is outside of the scope of this thesis, thus, for more detail, readers are encouraged to reference [GBC16, LBG15, BGCSV09].

In summary, presented here is an extended 5C CPPS framework with self-X smart attributes and technical principles. An end-to-end service is promoted as the smart attribute for CPPS. Section 3.3 introduces the concept of end-to-end industrial data processing. Chapters 6 and 7 present this concept in the form of two different practical industry cases.

3.3 Theoretical Contribution: End-to-End Industrial Data Processing by Neural Computing

The smart attributes for CPPS are presented in Section 3.2. In this part, it is shown how one can construct this type of smart production system, as well as the advantages of adopting a learning engine as the basic attribute of CPPS. Figure 3.14 presents an overview of how end-to-end industry service works. From a technological point of view, end-to-end service means little or no human effort to achieve predefined goals. Figure 3.14 is constituted by two parts: manual processing and automatic processing. The manual part conducts only basic activities: normalizing inputs, removing missed data, making up corrupted data, and so forth. The "automatic" part is run automatically, leveraging the power of learning techniques while implementing dynamic evolution systems. Theoretically, the technical pipeline is generic for two reasons align with the principle of data processing: (1) feature attraction is the first step needed to be done for any applications. The reason is that it is hard to know whether the provided features could provide enough information for solving the problems. Additionally, data collected through sensors contains noise is the common case. Hence, practicing feature engineering is essential. In the proposed architecture, various autoencoders could be applied, e.g., GAN-Autoencoder, Variational Autoencoders, Denoising Autoencoder and so on. (2) neural networks are powerful model-fitting tools, even without understanding the physical meaning of the data set.

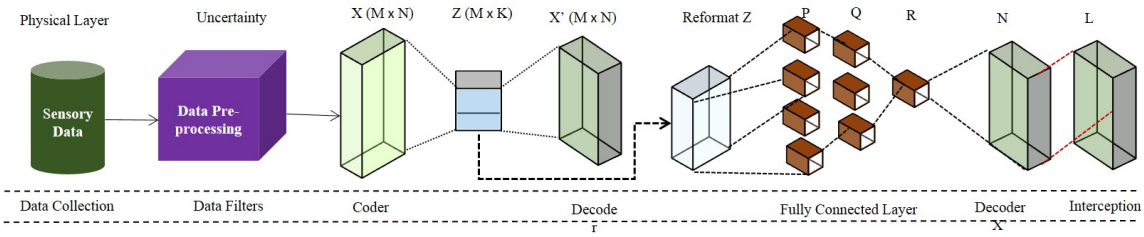


Figure 3.14: Technical pipeline of end-to-end industry data processing. In this thesis, this technical pipeline is used for predictive maintenance of manufacturing machines, and product quality monitoring and prediction. M, K, N, Z, P, Q, R, and L are the sizes of the different neural layers. From the end-to-end industry data processing point of view, autoencoder and dense layers are trained separately after data pre-processing. The stacked model later is trained through a unique learning algorithm, which is fine-tuning. This pip line is the main technical basis of this thesis. The proposed algorithms or methods adopted such a neural network structure, and later on, they are adapted to specific application circumstances.

The proposed technical pipeline can be defined as follows:

- **Data Pre-processing**. Usually, there are two main steps mandatory for almost all data analysis methods, i.e., (1) compensate the missed data points. For example, applying advanced sampling methods by simply adding some artificial data points. It is also very common if users remove the missed data. The only precondition is that these data do not have a strong effect on the final results. For example, time-series modeling does not accept too many data-missing points. (2) The other important step is to normalize the inputs. For neural network-based methods, this step is essential. If the input with high dimension and the corresponding values are too big, e.g., > 100, then the neural networks will become too excited and the internal state of all neurons close to 1. Oppositely, if the values are too small, then almost all the neurons will keep silent by all the means. The fundamental reason is that applying neuron networks on specific tasks-modeling is mainly making use of non-linear slope, in case we set the activation function as the sigmoid function, the slope is the part between the two read lines (Figure 3.15).

  There are three commonly applied methods for data normalization. Let's assume the data set is X, and i is the index of the sample in the data set:
  Min-Max

$$X_{normalized_i} = \frac{X_i - X_{min}}{X_{max} - X_{min}} \tag{3.27}$$

  Standardization

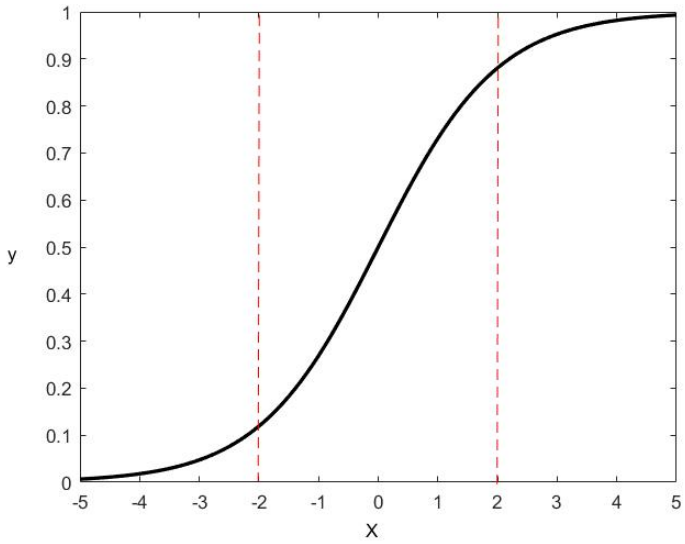


Figure 3.15: Sigmoid function.

$$X_{normalized_i} = \frac{X_i - X_{mean}}{\rho} \tag{3.28}$$

where, ρ is the stander deviation of data set X.
the stander deviation is calculated as:

$$\rho = \frac{(\sum X_i - X_{mean})^2}{M - 1} \tag{3.29}$$

where, M is the number of samples.
L_2 **normalization**

$$X_{normalized_i} = \sqrt{\frac{X_i}{\sqrt{X_1 + X_2 + ...}}} \tag{3.30}$$

- **feature extraction by various autoencoders**. It could be naive, sparsity, convolutional, denosing, sequence-to-sequence autoencoders and so on. This is generally defined as:

$$X = f_{decoder}(f_{encoder}(X)) \tag{3.31}$$

$$z = f_{encoder}(X) \tag{3.32}$$

Where z is the features, we want to extract from original data set in the form of neural representation. The typical adopted cost functions L used for training autoencoders are mean-squared-error and cross-entropy.

$$L_{MSE} = \frac{1}{M} \sum \sqrt{(h_w(x_i) - y_i)^2} \tag{3.33}$$

in order to clearly show the cross-entropy cost function, here we choose binary cross-entropy as an example, which is defined as:

$$L_c = -\frac{1}{n} \sum_X y \log a + (1 - y) \log (1 - a) \tag{3.34}$$

where, a is the output of neuron, y is target label.
Finally, for training the neural network, gradient descent based back-propagation methods are the most common used methods,e.g., training algorithm Adam, stochastic gradient descent and so on.

- **reformat latent space** Z. Generally, we do not need to reformat the Z. In case, this pipe line used for time-series data processing, the reformation operation is needed:

$$Z_{new} = [Z_1, Z_2, ..., Z_t]^T \tag{3.35}$$

where t is the time-window, in which our systems only remember the past t-steps historical states.

- **modeling systems with FFNN**. After successfully extracting the significant features, the remaining task is pretty simple. Theoretically, FFNN can model any systems, especially the recent research progress in deep neural network field, adopting a deep structure for this part is quite common in practical applications. The relevant training algorithms and cost function for training this part of neural networks are similar to autoencoders.

- **fine-tuning**. After finishing the steps mentioned above, we usually could obtain outstanding models. However, if we want to improve the performance of the entire solution, then fine-tuning is necessary. The training algorithms can be the same as the way training autoencoders or deep feedforward neural networks while the difference is the cost function, which is defined as

$$L = L_{autoencoder} + L_{deep\_ffnn} \tag{3.36}$$

Based on this new loss function, retraining the entire neural network again will improve the prediction accuracy.

From the implementation point of view, the thesis would like to introduce some tips for designing application-model: (1) construct an autoencoder only for extracting features and train the neural network. (2) design another neural network for fitting the labeled data set. (3) Train these two models separately. Alternatively, replacing (3) with (4), which makes use of the idea of transfer learning, i.e., if practitioners have done similar projects before, the suggestion is simply adopting the neural network structure and retrain it again on current data set. (5) fine-tuning the concatenated model on the training data set again. (6) if it is hard to precisely design neural network structure, one of the simple ways is simply applying Bayesian tools for automatically searching hyper-parameter.

Statistical models could replace the proposed pipeline, or statistical methods at least could achieve the same performance. While this thesis would like to argue that the main advantage of taking this pipeline is the automation, i.e., it does not need manually design any physical models and handle the uncertainty in the large-scale data set, which is main weakness of statistical analysis. Such an advantage is achieved by the learning ability of neural networks.

3.3.1 Application Scenario: Neural Computing for CPPS

In order to illustrate vividly how to apply neural computing techniques to the manufacturing field, the following part presents an industrial application scenario (Figure 3.16).

Various components, machines, robots, and industrial computers communicate with each other via internet cables or wireless networks. Sensors are installed on or attached to different machines and continuously collect relevant states of machines. There are few more steps need before uploading collected data into cloud or data processing center, i.e., (1) noise-filtering. Machines usually run in a complex environment, and many possible factors may effect on the data we care about, so filter out the noise as much as possible at the very early stage is essential for later data modeling. Besides, if we do not process noise at the data-collection layer after uploaded into the cloud, it will be much more difficult to be handled. (2) Data scaling. As aforementioned, neural networks are frequently used for solving a variety of real problems, due to the activation constraint, input scaling could help us project training data into an appropriate range so that neural networks could make use of their full potential abilities producing an accurate prediction.

After being well organized, the collected data is stored in the industrial cloud. The constructed model then accesses the cloud and automatically analyzes these sensory data, and then presents the results showed in figures for users. Here, we have two types of working modes, i.e., off-line and online. The

off-line working mode is much easier for modeling data set, as engineers can work on the stored data and select the most suitable models, while this is very expensive for companies in tow aspects. First, off-line means if we want to update the current system, we have to stop it and wait until new data processing model deployed in the system again, which is a costly strategy. The other reason is related to engineers. Companies will have to spend more resource on maintaining high skillful experts on such tasks. The online working mode is a reasonable solution. It makes use of neural networks or machine learning hyper-parameters automatic searching algorithms to finish the systems updating in advance so that companies do not have to stop the machines, while the underlying problem is that practical applications are time-critical.

Time-critical means that we do not know when the hyper-parameters searching algorithm could reach the optimal solutions. It is a time-consuming task. One of the rough solutions is supplying this part with more computing power to make the computation operator run faster. The more reasonable way is, of course, providing more advanced searching algorithms in which we are working on it and progress slowly. In general, There are few challenges still exits here need to be solved or tackled:

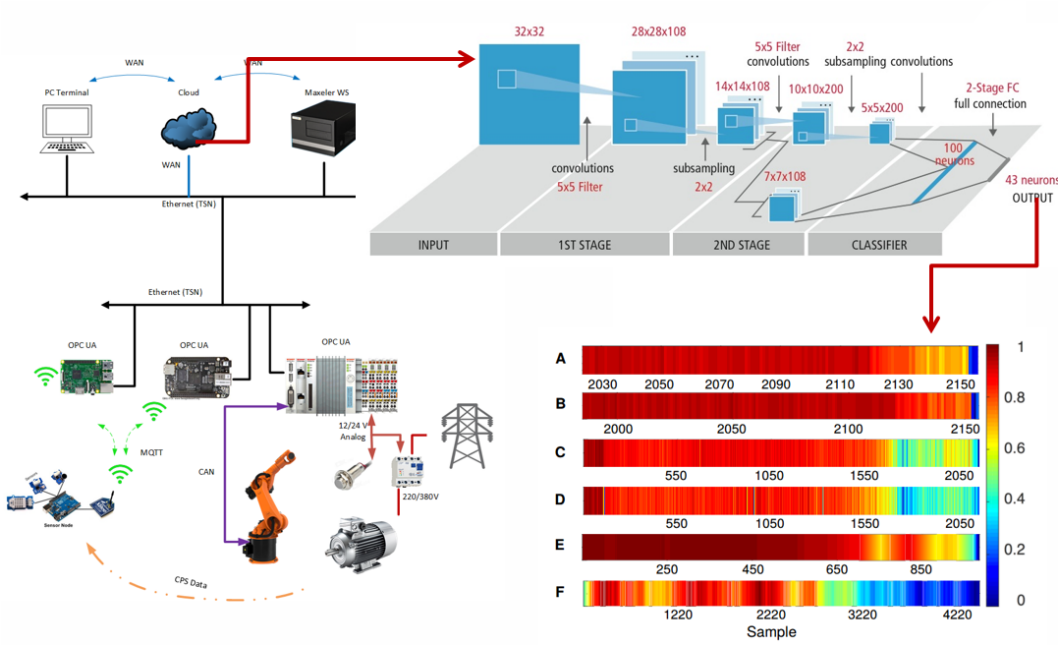


Figure 3.16: An end-to-end industrial data processing service platform.

(1) how to guarantee the time delay for uploading local data to the cloud. In time-critical applications, the platform architecture should be adjusted. For instance, applying edge computing

techniques to finish part of data processing tasks at the end nodes, so that the algorithms staying in the cloud only process more abstract data.

(2) Although part of the collected data could be processed at end computing devices while still batches of data will directly be stored in the cloud database. The existing problem is that those sensory data generated from various equipment may be stored in a different format. Therefore, a unique data transformation interface should be employed as a gate for the cloud database, e.g., SOAP is one of the possible solutions [Cle02].

(3) We have proposed the general computing framework and present several concrete models while the structure of neural networks is not fixed. Hence, the quickly adapting those hyper-parameters is an issue for time constraint scenarios.

In summary, this chapter introduces the necessary technical background of the thesis and present the main ideas the thesis designed and applied in various applications. The thesis proposes that learning ability is main the smart attributes for CPPS based on the existing work. In following a general industrial data processing framework is introduced. It shows that neural computing methods can guarantee the modeling performance of complex systems with the advanced back propagation training algorithm. Relevant application scenario is also shown at the end of the chapter. It presents an overview of the application of neural computing to real-world problems, based on end-to-end industrial data processing platforms.

The following chapters introduce the work developed for different industrial applications under the technical framework proposed in Section 3.3 (Figure 3.14). The experiment results demonstrate that neural computing methods are promising tools for modeling complex manufacturing systems.

CHAPTER 4

Semi-Automatic Industry Data Processing with ESN

This chapter discusses how to address the industrial data feature selection issue by easily combining classic statistical features with ESN. The milling process is taken as the research object and used to demonstrate the idea of the proposed method.

Milling metal with machining tools is a fundamental operation in the heavy industry. After a period of use, machining tools gradually become dull and less sharp, reducing the quality of their products because the worn or damaged part of the tools irregularly removes the metal surfaces. More seriously, the damage will eventually cause machines to fail and stop working. A novel method is here proposed for modeling tool wear procedure and predicting future tool wear degree in order to tackle this type of issue. Specifically, this begins with an analysis of statistical features, based on collected sensory data. The Pearson correlation coefficient (PCC) is then used as a feature selection rule. Finally, the selected features are fed into a neural network (ESN) for modeling the time-series procedure. The running outputs of ESN are the predicted tool wear states in future milling operations. To demonstrate the performance of the algorithm, several simulations are conducted on a milling dataset collected from the NASA data repository. Compared with commonly applied methods such as FFNN and time series neural networks (TSNN), this algorithm shows a better modeling accuracy. It worth noting that the ESN-based model requires much less effort to model similar industrial physical processes, as the neural network structure is randomly initialized and trained by merely using the linear regression.

4.1 ESN for Milling Process

In this part of the work, an ESN-based mixture model is designed for modeling and predicting the future tool wear status of milling operations. An ESN is a distinctive neural network, especially suited for time-series procedure modeling, because its randomly connected hidden neurons have various memories, e.g., leaky memory. The main advantage of ESN is its simplified neural structure and training methods. Typically, the work conducted here is the extraction of sensory features for milling tools and modeling healthy states for future steps by making use of the memory property of ESN. The contribution of this part of the work is largely the introduction of a semi-automatic technical framework for milling tool wear modeling. Details of the contributions of this work can be briefly summarized as follows:

- , A tool wear prediction method, is constructed based on statistical analysis techniques and ESN.
- A numerical comparison demonstrates the prediction accuracy of the technique.

In the following parts, Section 4.2 explains the principle of ESN. Section 4.3 presents the relevant statistical features, randomly selected in this part of the work. Numerous published works have been investigated, and the conclusion is that these features are commonly applied for milling process modeling. Section 4.4 introduces the main algorithm for the tool wear state prediction. Section 4.5 introduces the configuration of ESN, explains the experimental dataset and signals collection setup, and discusses the simulation results. Finally, the work is concluded, and several opening issues are presented.

Before introducing the technical details of the proposed algorithm, the ESN model and its hyper-parameters (for example, Spectral Radius (SR), sparsity of reservoir (Size of ESN (SoR)), input scaling (Input Scaling (IS)), size of neural network, and LR) are explained. To understand the application scenarios, the sensory data applied for demonstrating the algorithm is introduced in the following section.

4.2 ESN Model

ESN was first proposed by Jaeger [JH04] in 2004 as a distinctive RNNs. At the initial stage, the neural weights of the input-to-hidden layer and hidden-to-layer are randomly initialized based on a specific probability distribution, which is problem-dependent. They should be kept fixed in the following training procedure. Linear regression is the basic training method commonly applied for

ESN. Researchers are increasingly attracted by ESN and contribute to exploring neural structure and training techniques.

The excellent modeling performance of ESN for time-series signals carried out plenty of successful engineering applications, for instance, dynamic pattern recognition, robot control, and time-series prediction [JH04, SH07a]. The mini structure of ESN is presented in Figure 3. Generally, it is constituted by three different layers: the input layer, the reservoir layer (or hidden layer), and the output layer (or readout layer). For the provided inputs $u(n) \in \mathbb{R}^{N_u}$, target outputs $y^{target(n)} \in \mathbb{R}^{N_y}$

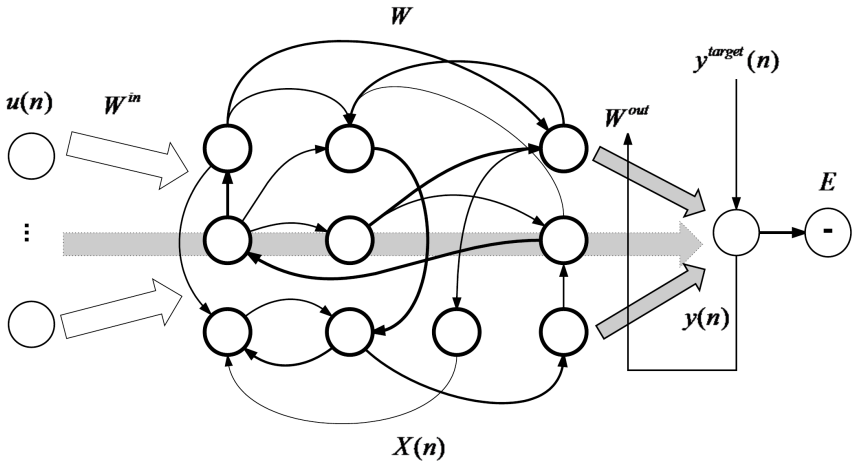


Figure 4.1: Basic ESN structure.

are known and the actual outputs are $y(n)$, where $N_* = 1, ..., T$ are discrete time stamps. T is the number of samples in the training dataset. W^{in} is the weight matrix of input-to-reservoir. W is the randomly generated neuron weights matrix of reservoir-to-reservoir. $W^{out} \in \mathbb{R}^{out}$ is trainable neural weight matrix of reservoir-to-output. RMS error is adopted to minimize the difference between desired outputs y^n and prediction y^{target}, i.e., minimizing the value of $E(y, y^{target})$.

$$E(y, y^{target}) = \frac{1}{N_y} \sum_{i=1}^{N_y} \sqrt{\frac{1}{T} \sum_{n=1}^{T} (y_i(n) - (y_i)^{target}(n))^2} \tag{4.1}$$

With input sequences fed into neural network, reservoir sates of ESN start to evolve as following principle.

$$\dot{x}(n) = f(W^{in}[u(n)] + Wx(n-1) + W^{out}x(n)), \tag{4.2}$$

$$x(n) = (1-\alpha)x(n-1) + a\dot{x}(n) \tag{4.3}$$

where, f is activation function (in the verification experiments, sigmoid function is selected as the active function) and α is leaking rate LR used to control the memory capacity of ESN. According to

the definition of ESN, the readout layer is the layer only conducting linear combination of reservoir states and inputs.

$$y(n) = f(W^{out}[u(n); x(n)]) \tag{4.4}$$

where, $[:; :]$ presents column vector (or matrix) concatenation operation. Accordingly, the computing performance of ESN depends on the following key parameters i.e., SR, SoR, size of reservoir, IS, and LR:

SR is one of the critical hyper-parameters. The maximal absolute eigenvalue of the reservoir connection matrix W decides the parater SR. The longer time inputs $u(n)$, states of reservoir $x(n)$ has weaker connection with the past states which occurs at the very early stage. That's because the later states have retained partial information in previous steps. This is one of the principles of ESN guarantees that linear model is sufficient for modeling the system. In order to generate appropriate echo states, the maximum eigenvalue of SR should always be kept as the value less than 1.

Sparsity of Reservoir (SoR) decides sparsity degree of neural connections inside the reservoir because too many connections with one specific neuron will cause the overshooting issue and the designed neural network is not able to learn anything out of training data set. The practical applications demonstrate that with appropriate sparse connections, the ESN model could reach the best performance and reduce the computing cost at the same time. The sparsity property generally solves one problem may happen to the neural network, i.e., hard-coding provided data set. With the less complex structure, the neural network has to find the right routine among different neural connections for describing the system's states.

The size of ESN-**SoR**. SoR, working together with SR, affects the modeling performance of ESN, in the way of controlling the complexity of the neural network, e.g., the larger reservoir always performs better than smaller ones. One therefore needs to synthetically select SoR and SR among different trade-off.

IS is the item used for re-scaling inputs into a reasonable range with two reasons: (1) large input values will cause overshoot problem, and neurons have the same states which close to 1; (2) small input values are not able to activate neurons generating any dynamic internal reservoir states. The neural states are basically all close to 0. Besides, if W^{in} is sampled based on uniform distribution, the parameters IS should be generated from interval $[-\alpha; \alpha]$, based on the distribution property of W^{in}.

LR responds to the convergence speed of neural networks training algorithms. It decides the evolutionary property of a reservoir discretized in time [Luk12]. Two types of LR are possibly picked up for ESN: (1) every neuron shares the same learning rate, which is an easy solution and could be

easily implemented; (2) setting various learning rate for every neuron. It is theoretically feasible and reasonable, but that is hard to be adjusted during the training process.

Existing works have developed numerous methods for manually or automatically exploring the hyper-parameters as mentioned earlier. In common practice, grid searching works well for solving specific problems. The brute-force searching method takes a long time to obtain satisfying results with appropriate hyper-parameters. Comparing ESN with classical RNN in the perspective of engineering implementation, neural network training, and model optimization, the known advantages of ESN are presented as follows:

- Random generated reservoir connections can afford rich, dynamic internal states for predicting outputs.
- Random generated connection matrix of input-to-reservoir and reservoir-to-reservoir overcome the computing limitation problem of RNN due to non-trainable bi-direction RNN.
- Through a nonlinear combination of reservoir states, ESN provides an easy and accurate way for modeling complex systems, for example, manufacturing systems.

4.3 Features Extraction and Selection

Raw sensory data is usually stored with noise, corrupted samples, and other unexpected issues. As aforementioned, in order to create accurate models for practical application, it is essential to make up the missing data values, remove noise, and extract meaningful features for the remaining works. Researchers from the industrial data analysis community have sought to mathematically and practically prove that statistical analysis is a powerful tool handling industrial data set [DSSW$^+$06, WWG13]. Hence, this thesis begins with applying statistical features on the milling process modeling. 8 dffierent statistical features are selected in our experiments. The reason is that they are commonly occurred in published works for monitoring the performance of mechanical systems, e.g., tool wear degree detection and prediction. Details of the selected features are listed in Table 5.8. CF is the abbreviation of crest factor. P/AR peak to average ratio and skewness." Different features indicate the different health states of machining tools, e.g., "RMS" is the measurement of a varying quantity, which is related to the energy level of signals. Kurtosis indicates the pulse of the signals and skewness characterizes the degree of asymmetry of the distribution around the mean value of a variable.

Of course, not all of the sensory data is interesting to users, but that's difficult to select those which are sensitive or essential to practical applications, i.e., meaningful features should always indicate the

Table 4.1: Summary of extracted features

Features	Expression
Mean value	$\bar{X} = \frac{1}{N}\sum_{i=1}^{N}\|x_i\|$
RMS	$X_{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i)^2}$
Variance	$X_{var} = \frac{1}{N}\sum_{N}(x_i - \bar{x})^2$
Maximum	$X_M = max(x_i)$
CF	$C_F = \frac{X_M}{X_{RMS}}$
Kurtosis	$X_{KURT} = \frac{1}{N}\sum_{N}(\frac{1}{\sigma}(x_i - \mu))^4$
P/AR	$X_{PAR} = \frac{X_M}{\bar{X}}$
Skewness	$X_{SKEW} = \frac{1}{N}\sum_{N}(\frac{1}{\sigma}(x_i - \mu))^3$

consistent trend of defect propagation [WWG13]. PCC (Equation 4.5) as a powerful statistic filter is adopted in this work for ranking various features for later uage.

$$PCC = \frac{\sum_i (x_i - \bar{x})(z_i - \bar{z})}{\sqrt{\sum_i (x_i - \bar{x})^2 (z_i - \bar{z})^2}} \tag{4.5}$$

4.4 Statistical ESN Algorithm

This section introduces the general idea of the proposed statistical features-based ESN algorithm for predicting milling tool wear degree. ESN plays an essential role in modeling machining tools' healthy states. The inputs of ESN are selected by statistic filters. Details of the proposed algorithm are as follows (Algorithm 4.1).

Algorithm 4.1: Statistical Echo-State Network

- Collect raw dataset from sensors U_n and make appropriate configuration for ESN, W^{in} and W.
- Clean the dataset: remove and make up the missed or abnormal value.
- Based on the cleaned data of the second step, use statistical analysis techniques to extract interesting information: we adopt 8 different features.
- Use PCC to rank those features for different signals.
- Apply the rest of the data as inputs to ESN and predict the future states.
- Return tool wear condition VB.

Two key sub-steps constitute the first step of the proposed algorithm. Firstly, the statistical features for sensory data are analyzed, such as the mean values and deviation values of sensory data. Based

on the analysis results of the first step, one can predefine the hyper-parameters of ESN which respond to the relevant step in the algorithm ("make appropriate configuration for ESN, i.e., W^{in} and W"). Hyper-parameter selection is an empirical task for ESN without any clear definition or rule of how to select the best ones for individual applications. One of the suggestion comes out from this work is trying to find similar works and borrow their hyper-parameters, fine-tuning the structure and retrain the entire neural network on the application data set.

Another aspect of the first step for data pre-processing is making up of missed data points: as this algorithm is specially designed to deal with time-series procedures, naturally if too many missing points missed, then it would lead to an unstable simulation prediction, and the results are doubtful. Missing data manipulating already has long-term research activities. Recent research works have shown great achievement in this field, e.g., estimating missing data by autoencoder [MKK+12, BJM17], or other types of neural networks [NMM07, SFS+17].

The second step is the common practice in data mining projects, i.e., cleaning noisy data and sensory data re-normalization. It is essential for better training performance of ESN and obtaining global optimal results. After finishing the data pre-processing step, PCC is then applied on the reformatted data set as the features filter selecting the most valuable information underlying the data, which indicates the most important patterns of systems behaviors. The reason here for selecting PCC determining main features is because of the information redundancy. PCC tries to measure the linear correlation coefficient between two different variables [BCHC09] and finally figure out principle features. These selected features can re-produce other ones by linear combination. After ranking all features, selecting top n positions as the principal components are done by adopting users' engineering experience. Since PCC is applied along with the experience of engineers in relevant fields, our proposed method is called semi-automatic feature selection.

After successfully selecting the most useful features, the remaining work is to create accurate models based on them. ESN is then adopted for modeling the tool wear status of machining systems with the pre-defined hyper-parameters in the first step. During the training procedure, the re-configuring of the hyperparameters is possibly happening simultaneously.

4.5 Experiment & Experiment Setup

4.5.1 Industrial Dataset

This section introduces the details of the experiment dataset and presents the technical structure of the experiment's setup. The experiment data set is sampled within the relavent physical setup. The run-to-broken simulations conducted with one type of industrial milling machine and operated under

various running environment, i.e., three different sensors installed on milling platform for monitoring the status of the machining systems. A variety of measurements regarding the tool wear degree monitoring and prediction are recorded: acoustic emissions, vibration signals, and the Amperage values for the machines. Accordingly, the dataset is well organized in a structured array, with the size of 1 x 167. The datset is classified into 16 different case studies with various running times. The variables and associated meanings are presented in Table 1 [Goe96]. The experiment dataset is

Table 4.2: Set of attributes used in this study, with descriptions

Field name	Description
case	Case number (1-16)
run	Counter for experimental runs in each case
VB	Flank wear, measured after runs,but not for each run
time	Duration of experiment (restarts for each case)
DOC	Depth of cut (does not vary for each case)
feed	Feed (does not vary for each case)
material	Material (does not vary for each case)
smcAC	AC spindle motor current
smcDC	DC spindle motor current
vib_table	Table vibration
vib_spindle	Spindle vibration
AE_table	Acoustic emission at table
AE_spindle	Acoustic emission at spindle

collected from NASA data repository [AA07]. The 16 cases are enumerated in Table 4.3.

Table 4.3: Experiment conditions

case	Depth of Cut	Feed	Material
1	1.5	0.5	1-cast iron
2	0.75	0.5	1-cast iron
3	0.75	0.25	1-cast iron
4	1.5	0.25	1-cast iron
5	1.5	0.5	2-cast iron
6	1.5	0.5	2-cast iron
7	0.75	0.25	2-cast iron
8	0.75	0.5	2-cast iron
9	1.5	0.5	1-cast iron
10	1.5	0.25	1-cast iron
11	0.75	0.25	1-cast iron
12	0.75	0.5	1-cast iron
13	0.75	0.25	2-cast iron
14	0.75	0.5	2-cast iron
15	1.5	0.25	2-cast iron
16	1.5	0.5	2-cast iron

Goebel [Goe96] introduces the details of the experiment setup, describing the properties of the physical setup, e.g., the size of work-pieces and the kind of materials used for cutting. Experiment running conditions are also covered in Goebel's work, e.g., where the sensors are installed, the moving speed of materials fed into milling machines, the cutting speed, and the depth for shaping metals. The collected signals from different sensors fall into various ranges. Thus it is necessary to conduct pre-processing before storing the data in industrial computers, i.e., amplifying or filtering streaming data, or feeding data into RMS devices. It is worth noting that applying RMS to sensory data during sampling phases is to smooth collected signals and make them more accessible for later data analysis usage. RMS is always regarded as the signature of the energy contents of specific signals [Goe96]. The governing formula of RMS is presented in Equation 4.6. While it is not necessarily applying filtering operation (e.g., RMS pre-processing) for all of the sensory data, e.g., samples generated by spindle motor are simply fed into the database without further processing.

$$RMS = \sqrt{\frac{1}{\Delta T}\int_0^{\Delta T} f^2(x)\,dt} \tag{4.6}$$

where $\Delta T = 8.00ms$, sampling rate is 250HZ, and $f(x)$ is the signal function. The data sampling process is depicted as follows in Figure 4.2:

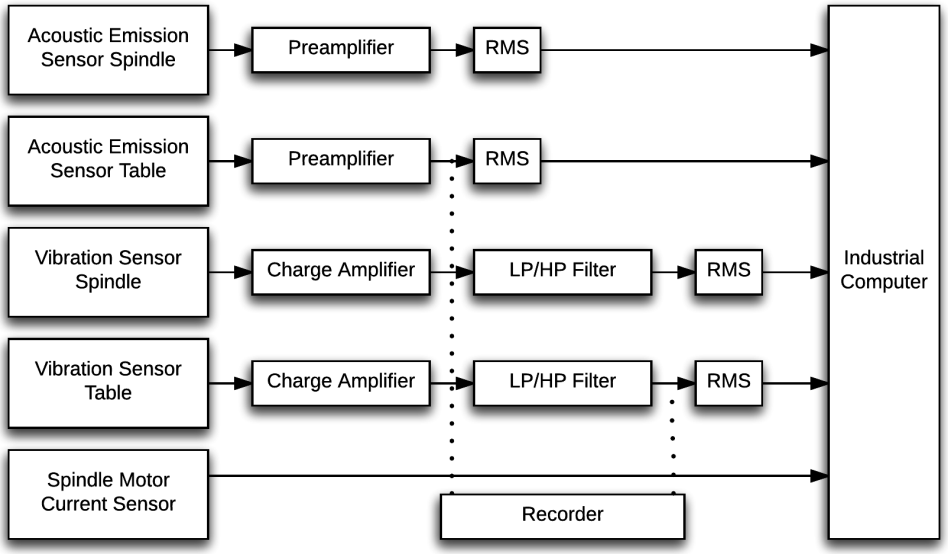


Figure 4.2: Experiment setup.

4.5.2 Configuration of ESN

Based on the introduction of the experiment dataset (Section 4.5.1), each running case is considered as a one time-series process. For every different application scenario, one specific ESN is designed

for that running case under the same technical framework. The hyper-parameters for each case study are explored by manual trials. The details of configurations are presented in Table 5.7. Three different running cases are selected to verify the performance of the proposed algorithm. Selected datasets are marked as Case 1, Case 3, and Case 11. To comprehensively demonstrate the performance of our

Table 4.4: Configuration for three different simulations.

No.experiment	Case	SR	SoR	SoR	IS	LR
1	1	0.6839	0.95	100	0.0258	0.009
2	3	0.6839	0.95	100	0.025	0.174
3	11	0.6839	0.95	100	1.35	0.25

algorithm, the comparison among the simulation results from TSNN, FFNN, and our method are presented in following part. The measurement criteria is mean-squared-error (the so called MSE) (Equation 4.7):

$$MSE = \frac{1}{N_y}\sum_{i=1}^{N_y}(y_i(n) - (y_i)^{target}(n))^2 \tag{4.7}$$

The prediction outputs of the algorithm are the flank wear, labeled as VB, which is commonly accepted as an indicator for evaluating tool wear degree. VB is measured by the running time of milling operations for calculating the distance from the cutting edge to the end of the abrasive wear on the flank face of the tool. For every measurement, the whole machining process has to be terminated.

4.5.3 Results Analysis

The experiment dataset has six columns, representing six different variables or features. Each feature contains 9000 values. If directly feeding inputs into ESN, the input dimension of the neural network would be 6 x 9000 which is too complex for the tasks. Hence, in the work, the first step is extracting the interesting features and reducing input dimension at the same time. Table 5.8 lists out eight types of statistical feature that are commonly accepted for industrial data analysis. It is difficult to say which one is better suited to the modeling goal, this is the reason why PCC is applied in the algorithm for ranking these features. The features gaining the highest scores are selected as inputs to ESN for modeling tool wear trends. Details of the experiment results for the case studies are listed in Table 4.5.

The proposed algorithm was implemented by Matlab toolkits. To clearly show the relation among results and various hyper-parameters configuration, the results from case study 11 are presented to explain how ESN works on time-series process monitoring and prediction. Figure 5.7 is case 11 simulation result, the corresponding weights of ESN depicted in 4.5. It is clear that the prediction results from our model are very stable, the underlying reason is that the internal neural weights W^{out}

Table 4.5: PCC Ranking

smcAC	smcDC	vib_table	vib_spindle	AE_table	AE_spindle
CASE 1					
0.4882	0.942	-0.7773	-0.89	0.9503	0.9436
0.9799	0.9312	-0.7862	-0.8934	0.9507	0.9396
0.9746	0.7527	-0.8455	-0.8868	0.9271	0.8583
0.9779	0.854	-0.7394	0.596	0.7811	0.8005
-0.9465	-0.9623	0.376	0.7758	-0.7938	-0.4209
-0.9599	0.9169	0.8394	0.4854	0.1843	-0.1214
-0.9457	-0.9625	0.1645	0.7663	-0.7755	-0.3725
0.06397	-0.9188	-0.3318	0.4733	-0.9457	-0.875
CASE 3					
0.815	0.9467	-0.8006	-0.4802	0.9547	0.8333
0.9514	0.9521	-0.826	-0.4687	0.9541	0.826
0.9574	0.9145	-0.8089	0.4894	0.898	0.6431
0.9232	0.9466	-0.86	0.7322	0.9228	0.6618
0.03612	0.3632	-0.7639	0.7641	0.1259	-0.1147
-0.08484	-0.4043	0.7817	0.7425	-0.0007	0.4092
-0.9385	0.4407	-0.806	0.7633	0.2067	-0.08621
0.8161	0.3082	-0.7845	0.7389	-0.8739	-0.8058
CASE 11					
0.8195	0.9481	-0.8054	-0.7834	0.9682	0.8716
0.9544	0.9542	-0.8141	-0.7414	0.9716	0.8356
0.9535	0.9533	-0.7336	0.2935	0.9548	0.578
0.9594	0.9624	-0.7926	0.6211	0.9749	0.8057
0.4876	0.5018	-0.1323	0.6659	0.9404	0.6712
-0.1635	0.2024	0.5981	0.6964	0.7098	0.7369
-0.9567	0.5915	-0.2627	0.6624	0.9491	0.6978
0.8574	-0.1603	-0.4359	0.7411	-0.8573	0.4807

all fall into the range of $[-1, 1]$, i.e., reservoir weights respond to the dynamic property of ESN, which control internal neurons' pulse activities in the form of balancing exhibition and inhibition connections. Such control mechanism keeps the entire neural network always from chaos status which stays at the edge between stable and unstable states.

ESN, TSNN and FFNN

TSNN is a typical recurrent neural network with feedback connection directly from output to input under the constrain of specific time-step delay. The structure of TSNN is presented in figure 4.3. where, x_1, x_2 and x_n are input of TSNN, $h_k^i(a_k)$ is the hidden state of kth neuron in i th layer with

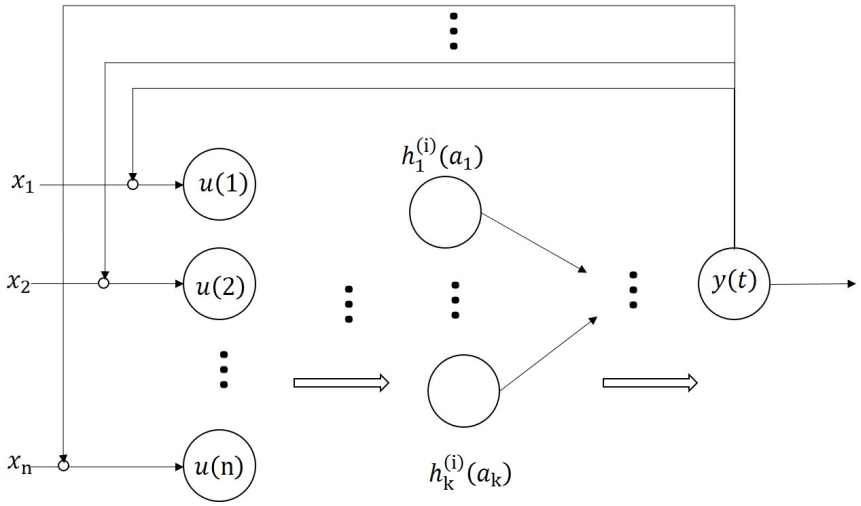


Figure 4.3: Neural network structure of the TSNN adopted for experiment comparison.

input a_k. The input of the neural network is then can be defined as:

$$u(n) = x_n + y(t - d) \tag{4.8}$$

where, x_n is the original input, $y(t)$ is the output of TSNN at time t, d is the delayed time steps. With input, TSNN first moves forward and obtains output $y(t)$. Then, feedback the output to input. If the delayed time steps larger than the predefined threshold, $y(t)$ will be merged with new input x_n as new input fed into a neural network. If we use the same of defining the input of ESN, the input could be like:

$$u(n) = x_n + \alpha y(t) \tag{4.9}$$

where, α is the forget factor. FFNN could be considered as the simplified version of TSNN which directly removes all the historical information.

$$u(n) = x_n + 0 \tag{4.10}$$

As one of the typical RNNs, TSNN is designed for processing a temporal dataset, even though there are many more advanced time-series data processing methods while trying to make the solutions simpler but effective are always mandatory. That is why TSNN is selected as one of the comparison methods. Regarding FFNN, besides the same reason as aforementioned, we do not know whether the data points have any strong time-series property. Therefore, FFNN is chosen here as the baseline for comparing the experimental results.

Experimental results comparison

To compare the performance of the proposed algorithm, the same inputs as those used for the ESN are fed into neural network TSNN and FFNN. The setup of these two neural networks is also kept the same as that of ESN to ensure the fairness of the comparison. For instance, if the SoR is 100 and the SoR is 0.95, then the hidden neurons for TSNN and FFNN are 5. For all simulations, the same percentage dataset for training and testing is applied. The entire simulation results are listed in Table

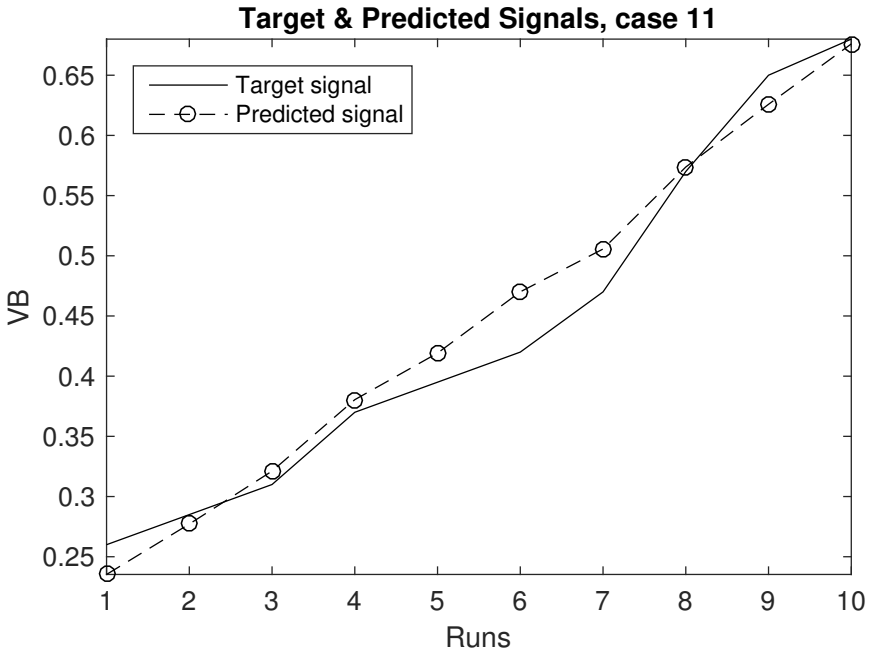


Figure 4.4: Case 11: VB prediction, MSE=5.9E-4.

4.6. It is clear that ESN provides much more accurate results than TSNN and FFNN. Cases 1 and 3 have the same simulation process as Case 11, but with different configurations.

The demonstration shows that ESN always performs better than TSNN and FFNN in terms of prediction accuracy, which is due to the way in which ESN trains the RNNs. Specifically, ESN

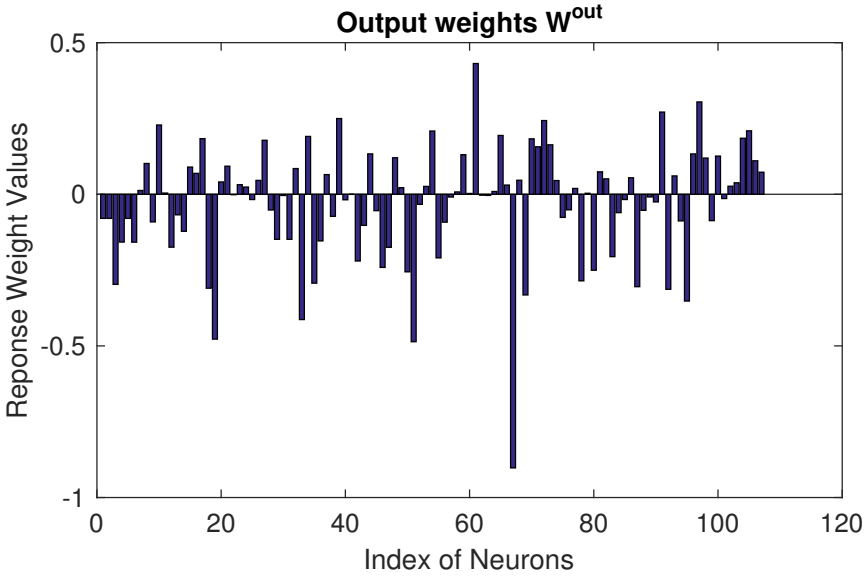


Figure 4.5: Case 11: internal weight matrix.

Table 4.6: Simulation results: error MSE for Cases 1 and 3.

Case	ESN	TSNN	FFNN
1	1.5e-3	1.5e-3	1.5e-3
3	2.4e-3	2.5e-2	3.0e-3
11	5.9e-4	6.0e-3	3.9e-3

carefully choosing hyper-parameters controls the echo-state following by adopting linear regression training algorithm, i.e., simply combining the internal states generated by ESN. The reservoir states reflect the fading memory of the neural network and these fading states represent the time-series relation among historical states with the current one, which is the main difference between ESN and classical recurrent neural networks in the way of dealing historical systems states. RNNs model the system through a way, which could be called hard-coding, to remember and forget states, e.g., setting the time-delay of one neural network as 4 which is a discrete way processing information. However, ESN achieves the accuracy prediction performance in a more continues way.

Besides, the other advantage of ESN is the training cost. TSNN and FFNN apply $Levenberg$ - $Marquardt$ optimization techniques to train neural networks. With the optimal approach, the backpropagation algorithm is adopted for training artificial RNNs. It is obvious that linear regression training has much lower computing time and computation complexity.

Another interesting find out from the experiments is that with the same amount of neural connections, ESN can generate richer internal states (also called reservoir states), the main reason is that ESN neu-

rons share their states with the neighbor neurons, i.e., the so-called information-sharing mechanism. The results from Case Studies 1 and 3 are close to each other among three methods. It is because, with a small amount of training data, ESN is not fully wake up (not wash out unstable and noisy signals) and generate appropriate internal states. Therefore, the washout time of ESN is essential for modeling performance. In practical applications, the washout time has to be chosen precisely to remove the unstable states and regenerate useful internal features so that the readout layer can linearly combine them modeling evolving trend.

4.6 Summary

This chapter introduces the semi-automatic industrial data processing algorithm by combining statistical features with ESN. As the cutting process is essentially a time-series process, ESN could be an excellent candidate for it which is roughly common in CPPS. Simulation results demonstrate the performance of the algorithm. Comparison indicates that ESN always performs better than TSNN and FFNN with a large number of datasets. To the best of the available knowledge, this is the first time that ESN has been introduced for modeling milling tool-wear condition.

CHAPTER 5

A Heuristic Searching Method for Exploring Hyper-parameters of ESNs

ESNs provide a distinct architecture, and a supervised-learning principle for RNN [JH04] and are very similar to the liquid state machine developed by Maass [MNM02]. The main idea of ESNs is to feed an input signal to a large, randomly generated, recurrent, dynamic hidden layer, called reservoir, whose outputs are linearly combined by a memory-less layer called readout [LBJ17]. The connections among input, reservoir, and output layers are randomly generated and fixed at the initial stage. As a consequence, for training the ESN, only linear regression algorithms are required. These algorithms only update the weights between reservoir and output layer, while traditional RNN models need a long period of hard training through back-propagation optimization algorithms, as the learning procedure in RNN need to adjust all the network weights, including input, output, and RLs.

This work proposes an alternative method based on the bootstrap (BS) sampling method for automatically searching optimal values. A uniform distribution is adopted, from which independent and identically distributed samples are drawn in order to generate the global parameters matrix. The proposed algorithm automatically scans the matrix, row by row, until reaching the predefined termination condition. The main contribution of this work is to provide an efficient algorithm for automatically computing the optimal or sub-optimal global parameters of ESN.

The remainder of this chapter is organized as follows. In Section 5.1, the basic structure of ESN is reviewed. In Section 5.2, the ZIZO searching algorithm is presented. To demonstrate the utility of the proposed method, Section 5.3 presents the simulation results and relevant experiment analysis. Finally, Section 5.5 ends this part of the work with the conclusion.

5.1 ESNs

5.1.1 State Equations

The basic structure of an ESN is depicted in Figure 2 where n is the discrete time-steps for the time-series procedure and $u(n)$ is the input at time-step n. $X(n)$ is the generated states of the

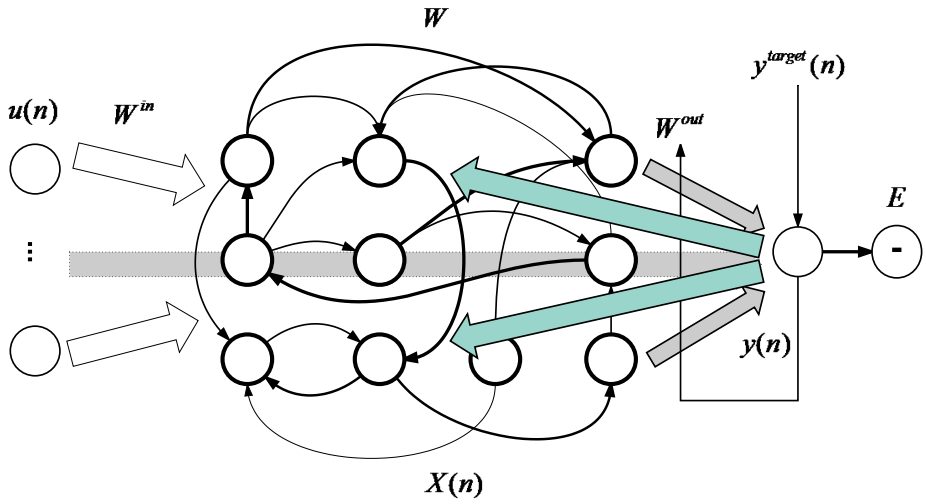


Figure 5.1: Basic structure of ESN.

reservoir and used to train the ESN with feedback outputs, W^{in} are the input-to-reservoir connection weights which are initialized randomly, and W is the internal weights of the reservoir. W and W^{in} are both randomly generated and kept fixed during entire training process after initial phase. $y(n)$ is the output of readout layer and $y_{target}(n)$ is the reference output. E is the unit used for computing the deviation between $y(n)$ and $y_{target}(n)$.

In the reservoir layer, for every neuron one can apply various activation functions, such as $sigmoid$, $tanh$, and other non-linear functions. This work uses the tanh function. The state equations of ESN are presented as follows:

$$X'(n) = f(W^{in}[\beta \cdot u(n)] + WX(n-1)) \tag{5.1}$$

$$X(n) = (1-\alpha)X(n-1) + \alpha \cdot X'(n) \tag{5.2}$$

where f is the activation function and α is the LR used as memory loss factor of ESN regarding the historical states. The output layer is defined in Equation 5.3:

$$y(n) = g(W^{out}[u(n); X(n)]) \tag{5.3}$$

where [:; :] is the concatenation of a column vector (or matrix). For the output activiation function, one can choose different functions based on practical application problems.

It is worth noting that in the presented structure of ESN, the neural model is equipped with feedback inputs marked with a light green color. In the states' control formulas, the feedback weights are not added to the model.

5.1.2 Global Control Parameters

Given Equations 3, 4 and 5.3, an ESN model can be defined by a 6-tuple $(W^{in}, W, W^{out}, \alpha, \beta, m)$, m is the size of the $reservoir$. Analogous to RNN, W^{in}, W, W^{out} are called the $local\ parameters$ whereas α, β, m are called the $meta - parameters$, as they work together to affect the distribution of W^{in}, W, W^{out} [Luk12]. This work uses the term "global parameters", which was introduced by Mantas [Luk12] to reflect the nature of ESN parameters. There are also other important global parameters, e.g., teacher scaling, feedback scaling, sparsity of connections in a reservoir, and noise in state-update. This work focuses on those which listed as below.

Size of Reservoir m: The first global parameter is the size of a reservoir m, which has a high impact on the performance of ESN. Intuitively, the larger m is, the better the performance could be obtained, as the larger reservoir space could lead to a higher chance of finding appropriate internal states producing accurate outputs. Akin to the traditional machine learning problem, the over-fitting issue is also commonly observed, which means that a larger number of reservoir neurons does not always mean the better results. Another problem is that a too large reservoir may require extensive data for proper training, and these data may be lacking due to engineering difficulties.

Input Scaling β: This is used for shifting input values to the activation range of neurons in a reservoir. It determines the operating point of nonlinear reservoir behavior. Finding this point is very difficult in general, and it is application-dependent. One should choose appropriate IS based on the degree of nonlinearity of the tasks, while the nonlinearity degree of a task is difficult to be defined quantitatively. Therefore, manually testing different IS is tough work when practicing ESN.

Spectral Radius ρ: The very nature of an ESN model is the echo-state property. The large ρ can lead to reservoirs hosting multiple fixed points, periodic, or even chaotic (when sufficient nonlinearity in the reservoir is reached) spontaneous model modes, violating the echo-states property [Luk12], as the state of a reservoir x(n) should be uniquely defined by the fading history of inputs $u(n)$. If ρ is too big, the reservoir state $x(n)$ would heavily depend on the initial conditions that are existent before the inputs. A stronger $u(n)$ push activation of the reservoir neurons away from 0, in which their tanh nonlinearities have a unitary slope, to the regions with smaller gradient, thus reducing the gains of the neurons and the effective strength of feedback connections. Therefore, the recommended

range of the SR should be less than 1, which ensures ESN produce echo-state in most circumstances. In practical applications, the SR is calculated as follows:

$$\rho = |eigs(W)| \tag{5.4}$$

$$W = W \cdot (1.25/\rho) \tag{5.5}$$

where $eigs$ is the function used to calculate the max eigenvalue of a reservoir weight-matrix W, $|\cdot|$ is the absolute value of one variable.

Leaking Rate α: This is used to define the speed with which the reservoir states are updated based on the new input and output information. A smaller α means a lower update speed of the ESN. In practice, it is difficult to define an appropriate update rate for ESN.

This work attempts to develop an algorithm that can automatically generate α, β, m and ρ. As described in Equations 5.4 and 5.5, ρ is defined by m, hence, in this work, just the natural features of a landscape of automatically computing the global parameters, i.e., α, β and m, are considered.

5.2 Zoom-In-Zoom-Out Searching Method

The ZIZO searching algorithm is inspired by the BS method proposed in [Efr92]. The BS is a straightforward way of estimating standard errors and confidence intervals. If one finds a sample with acceptable Mean Squared Error (MSE) (a value lower than a specified threshold), then that sample is selected. Otherwise, the sample with the lowest MSE is chosen and a box randomly generated around it by sampling left and right boundaries for each dimension of the sample within the original range of the parameters. The search is then continued as before until the MSE is lower than the predefined threshold. The theoretical results will be presented in following with explanations of the thought behind the proposed algorithm.

The principle of ZIZO is presented in Figure 5.2. At the initial stage, according to the ranges (α_i, β_i, m_i) of the global parameters p_i, the box with the dotted line is obtained (it is assumed that the parameters fall into two dimensions). N random samples are then generated inside the box, picking every sample as parameters of ESN and running the defined ESN model. In the next step, the MSE of the output is verified. If MSE is less than θ, then program is terminated. After exploring N samples, if the desired global parameters have not been found, the one with the minimum value achieved by MSE is chosen as a reference point for generating a new searching box. For every random box, random ranges (α_i, β_i, m_i) are selected based on the reference points. Specifically, for each global parameter p with value v_p, a random range is chosen $[a_p, b_p]$, where a_p lay in the interval $[u_{p^m}, v_p]$ and b_p lay in the interval $[v_p, u_{p^M}]$, where $[u_{p^m}, u_{p^M}]$ is the range of the parameter p within the dashed box. As

Algorithm 5.1: The Proposed Method - ZIZO Algorithm

Data: Input X, Output Y, sampling rate N, maximum re-sampling times M, desired mean square error E_d, prediction length L, RS is the range of global parameters (α, β, m), update rate γ_α, γ_β, γ_m, randomly generated radius BR.

Result: Optimal or sub-optimal (α_i, β_i, m_i).

1 **begin**
2 **for** $j = 1$ *to* M **do**
3 $PARA$ = rand(N, $[\alpha, \beta, m]) \subset RS$
4 **while** $i = 1$ *to* N && $\epsilon_{ji} > E_d$ **do**
5 ESN=$\{\alpha_i, \beta_i, m_i\}$ = $PARA_{(i,:)}$
6 $\epsilon_{ji} = E(y, y^{target}) = \frac{1}{L}\sum_{i=1}^{L}\sqrt{(y_i - (y_i)^{target})^2}$
7 **end**
8 **if** $i <= N$ **then**
9 $break$;
10 **end**
11 **else**
12 $(\alpha, \beta, m) = (min(\epsilon_\alpha), min(\epsilon_\beta), min(\epsilon_m))$
13 BR = rand(1, $[BR_\alpha, BR_\beta, BR_m]$)
14 RS_α = [α - $\gamma_\alpha \cdot BR_\alpha$, α + $\gamma_\alpha \cdot BR_\alpha$]
15 $s.t.$ α - $\gamma_\alpha \cdot BR_\alpha >= \alpha_{lowerBound}$
16 α + $\gamma_\alpha \cdot BR_\alpha <= \alpha_{uperBound}$
17
18 RS_β = [β - $\gamma_\beta \cdot BR_\beta$, β + $\gamma_\beta \cdot BR_\beta$]
19 $s.t.$ β - $\gamma_\beta \cdot BR_\beta >= \beta_{lowerBound}$
20 β + $\gamma_\beta \cdot BR_\beta <= \beta_{uperBound}$
21
22 RS_m = [m - $\gamma_m \cdot BR_m$, m + $\gamma_m \cdot BR_m$]
23 $s.t.$ m - $\gamma_m \cdot BR_m >= m_{lowerBound}$
24 m + $\gamma_m \cdot BR_m <= m_{uperBound}$
25
26 RS = [RS_α, RS_β, RS_m]
27 **end**
28 **end**
29 **if** j <= M **then**
30 $Get\ the\ optimal\ (\alpha_i, \beta_i, m_i) = min(\epsilon)$
31 **end**
32 **else**
33 $Get\ the\ sub-optimal\ (\alpha_i, \beta_i, m_i) = min(\epsilon)$
34 **end**
35 **end**

shown in Figure 5.2, after the first N trials, the sub-optimal reference point C_1 is obtained. Then a new box around C_1 is randomly generated. The manner of generating the random box is shown in lines 11, 16 and 20. γ_α, γ_β and γ_m are used for adjusting the boundary of the generated random

box. Following by the previous steps, the program runs the loop until reaches predefined conditions. Finally, reference points C_2, C_3, C_4, ..., C_i, C_n, C_{star} are obtained. By fixing the size of the box in advance, it is possible to miss the optimal solution. For instance, in tree searching, it has both width-searching and depth-searching. If missing the optimal solution in the searching process, the program has no chance to back to the previously gained sub-optimal values without backtracking. In the ZIZO searching algorithm, a random-size-box search is proposed. The ZIZO strategy gives searching algorithm an opportunity returning to possibly-missed optimal solution space (akin to backtracking). According to BS theory, if randomly selecting q different samples in one specific box

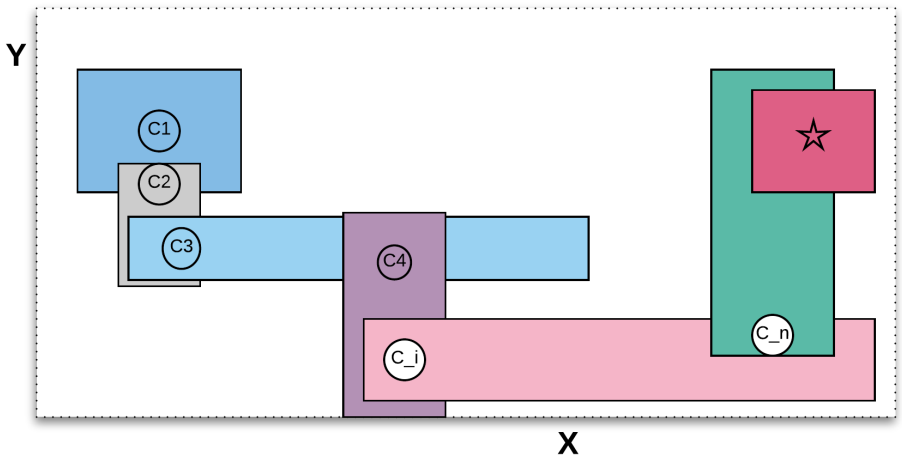


Figure 5.2: Principle of ZIZO.

with uniform distribution, then every sample is chosen with a probability of $1/q$. Theoretically, if keeping sampling an infinite number of times, the optimal parameters could eventually be obtained (for example, the standard deviation), while the percentage of non-selected samples in the dataset is $\lim_{q\to+\infty}(1-\frac{1}{q})^q \approx e^{-1} = 0.368$. In contrast to brute-force grid searching method, the proposed method can, in the worst case, obtain a batch of optimal parameters by only searching 63.2% of the solution space.

The time complexity of the proposed algorithm is $O(M \cdot N)$, the computing cost of the algorithm is constant which is $O(C)$. That is the most significant advantage of the proposed algorithm, compared to other existing methods, including brute-force searching, grid searching, and evolutionary algorithms.

5.3 Algorithm Validation

The demonstration is conducted on two different datasets: (1) the data generated from the Mackey-Glass model and (2) the data generated from the current signals of a machining process.

5.3.1 Mackey-Glass Model

A sample of the Mackey-Glass model generated data is depicted in Figure 3, where 10000 data points

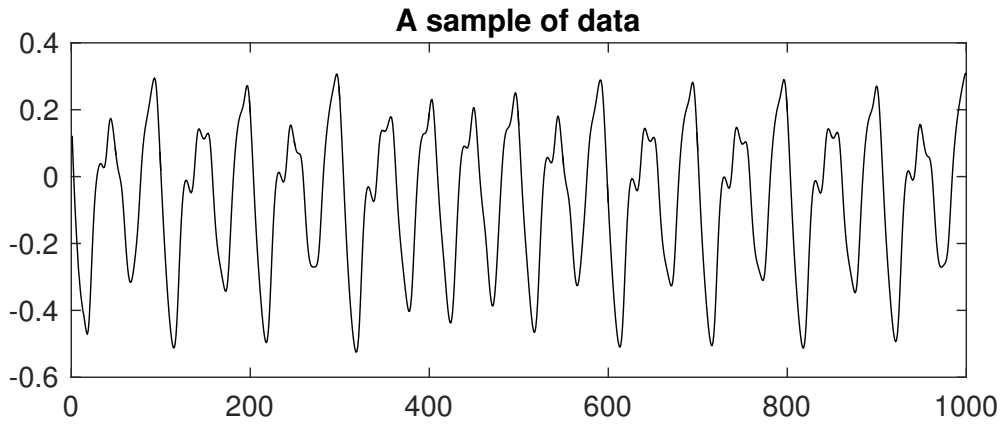


Figure 5.3: A sample of Mackey-Glass Model.

are plotted. The presented samples show that the system evolves periodically with slight difference among various oscillation period, i.e., periodic patterns look similar but not the same. Within the training dataset, the first 2000 samples are used for training ESN and the remaining data applied for prediction. For comparing the simulation results with the benchmark test, the first 500 predicted points are used for calculating the MSE. Through the proposed algorithm, different parameters of ESN are obtained through batches of simulation. The results are listed in Table 5.1. The relevant simulation setup is: $M = 500$, $N = 100$ and the predefined threshold $MSE = 1.0e - 6$.

In [Luk12], an MSE with the order $1.0e - 6$ is obtained. The comparison between the best result and the proposed method is shown in Table 5.2

Table 5.1 and Table 5.2 show that the ten times better prediction accuracy can be automatically achieved, compared with the outputs from existing works. Table 5.2 presents the simulation results. In the first row, 101 iterations are performed to obtain MSE = 7.5562e-07. In the 5th row, after the same number of iterations, a different MSE with 9.9136e-07 is obtained. Comparing these two cases, the deviation between them is tiny, while the main reason for the difference is that it is caused by the random search box. To visualize what happens inside the ESN, the states of weight distribution (Figure 5.4) and $Reservoir\ States$ (Figure 5.5) are plotted. To explore what happens inside the

Table 5.1: Selected parameters for Mackey-Glass model

Case	Iterations	α	β	m
1	101	0.47666	0.80618	444
2	38	0.81052	0.93495	379
3	112	0.32762	0.89783	364
4	59	0.6172	0.97062	393
5	101	0.46795	0.87622	407
6	5	0.64911	0.80069	447
7	245	0.54739	0.53543	351
8	266	0.68985	0.9884	216
9	43	0.50805	0.78367	427
10	106	0.50058	0.90311	304

Table 5.2: Selected parameters for Mackey-Glass model

Case	MSE	Computing Time	Atuomation
case 1	7.5562e-07		
case 2	8.3911e-06		
case 3	9.2214e-06		
case 4	3.2251e-06		
case 5	9.9136e-07	$\leq 30s$	automation
case 6	7.7338e-06		
case 7	1.5869e-06		
case 8	3.9363e-07		
case 9	6.6814e-07		
case 10	7.3933e-07		
reference[10]	1.0e-6	$\geq 30s$	semi-automation

reservoir during the running time, parts of the states of the reservoir in the first 200 time steps are plotted (as for "Table 8.1: Selected Parameters for Mackey-Glass Model"). The deviation between the target and the prediction signals (Figure 5.6) is also presented to show the accuracy of the obtained results. Besides the prediction accuracy, the time complexity of ZIZO is constant $O(M \cdot N)$, M X N is the maximum iterations ZIZO needs to search the optimal solutions. All the executed simulations are finished within 30 seconds while the existing methods need much more time.

According to the theory of ESN, the weight distribution and reservoir states work together and control the prediction accuracy. Figure 5.4 shows that the range of weight values of ESN is between $(-2.9, 7.8)$ and Figure 5.5 shows that the obtained ESN has very rich reservoir-states. These states should be independent with each other as far as possible, as the output is a linear combination of reservoir-states and too many similar internal states will lead to unstable prediction outputs. Figure 5.6 shows the prediction results generated by the trained ESN. For the first 1000 data points, the prediction error is close to zero. Moreover, in all of the study cases, the generated global parameters lead to good

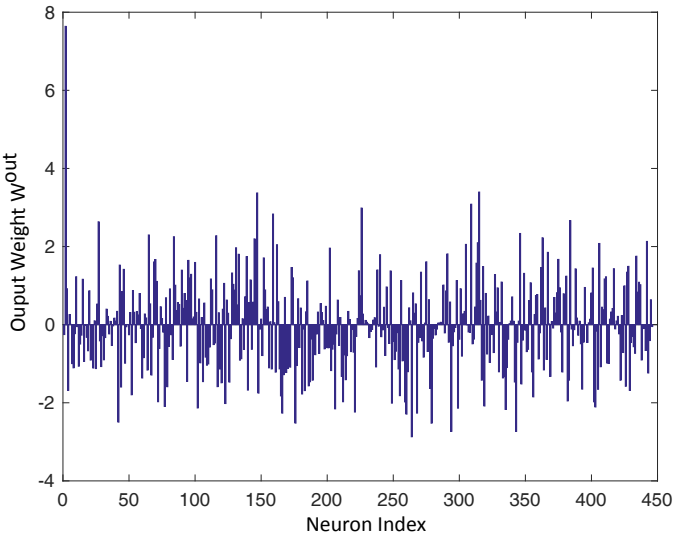


Figure 5.4: Weight distribution.

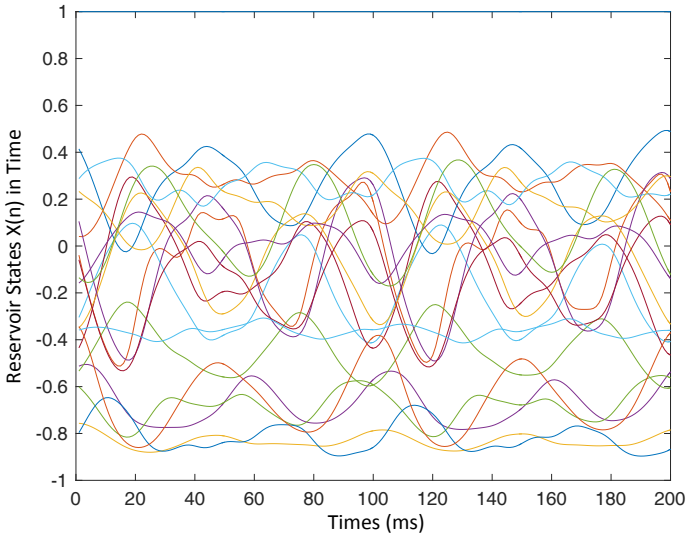


Figure 5.5: Reservoir states.

prediction accuracy. For practical application, selecting appropriate parameters is constrained by specific selection rules [Luk12]. To acquire a better ESN, several rules could be followed as tips. From the stability point of view, the longer "zero-error" time (points) in Figure 5.6, the better stability of ESN. The weight distribution and the correlation among the lines of the reservoir states are two important signs when selecting appropriate global parameters. First, the weights should be very close

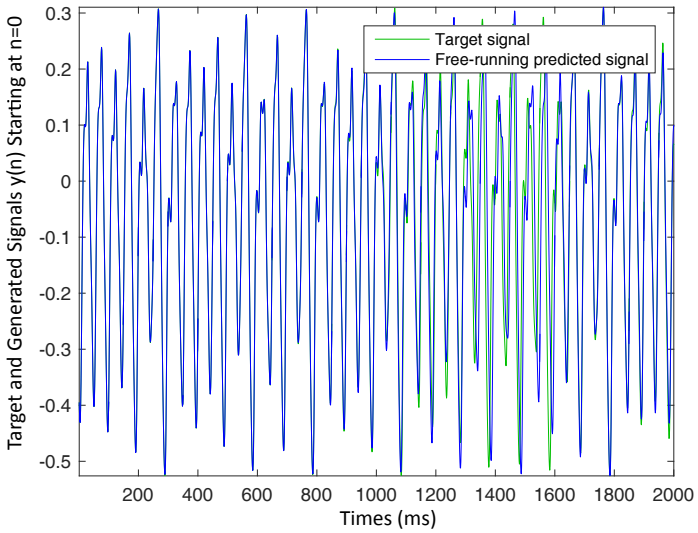


Figure 5.6: Target and prediction signals.

to each other. When the range is smaller, the stability of ESN is better. Second, the reservoir states should be independent with each other, as far as possible. According to the simulation results of the Mackey-Glass benchmark test and the selection rules, it is obvious that when the weight is in the range $[-10, 10]$ and reservoir states fall into $[-1, 1]$, the ESN it would be more stable.

5.3.2 Motor-Current Signal Prediction

Motor-current is one of the most important parameters for monitoring the health states of machining tools [SA07]. In this section, the milling process-generated data in [AA07] is used. The motor-current of the DC spindle is selected to evaluate the performance of the proposed method. A sample of the current data is depicted in Figure 5.7. In industrial engineering, the signal in the first 3,000 time-steps can be dropped, because during this period the system is very unstable (However, even after a long running time, the collected signals contain oscillatory states. See the bottom part of Figure 5.7). Therefore, the data points in the range [3000, 9000] are selected in the experiments. The simulation setup is M = 500, N = 100, the threshold MSE = 1.0e-2, and the predicted first 500 data points are used for calculating MSE.

As the sensory data from the machining process contains a considerable amount of noise, the model is not very accurate. While, from the industrial engineering point of view, a good prediction accuracy is around 1.0e-3 which demonstrates that our model still can automatically achieve the desired outputs. Similar to the previous benchmark test, 10 different runs of simulation of the algorithm are provided

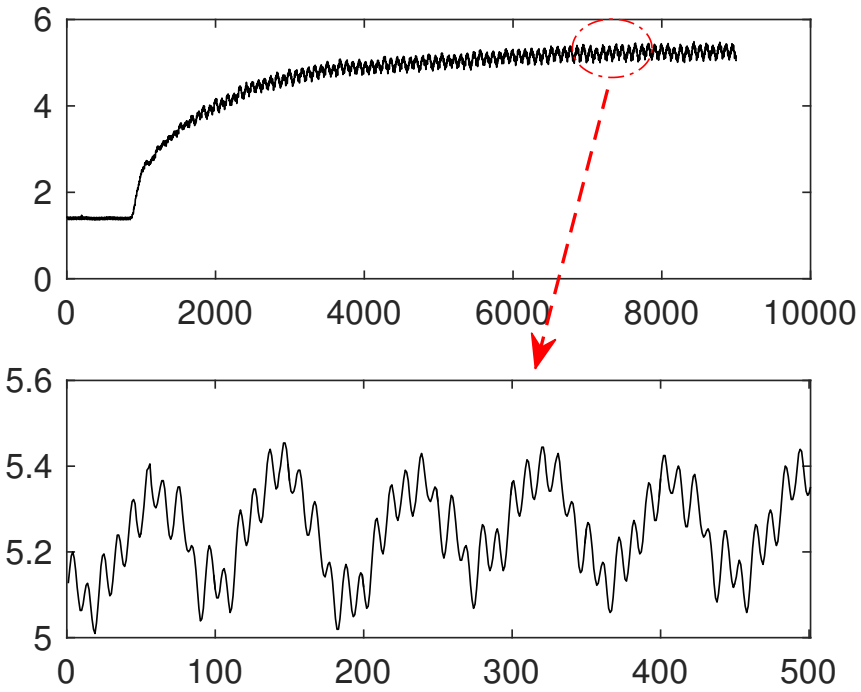


Figure 5.7: A sample of DC motor-generated current signal (up) and points from 7500 to 8000 marked by the red circle (down).

Table 5.3: Experiment results for Case 1.

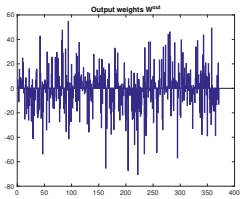
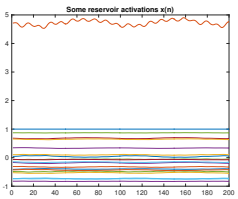
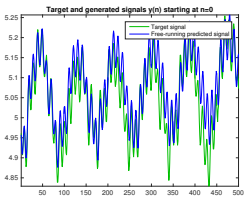
Weight Distribution	Reservoir States	Prediction Result
		

Table 5.4: Experiment results for Case 2.

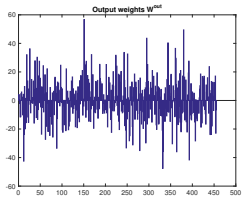
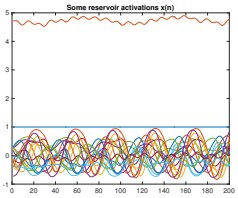
Weight Distribution	Reservoir States	Prediction Result
		

Table 5.5: Experiment results for Case 3.

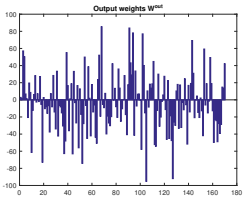
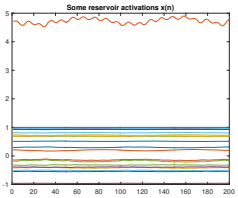
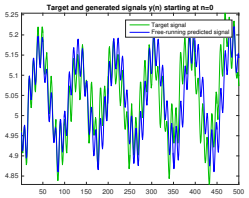
Weight Distribution	Reservoir States	Prediction Result
		

Table 5.6: Experiment results for Case 4.

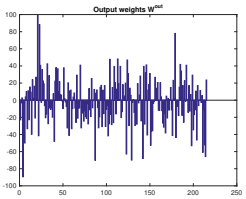
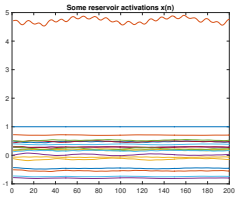
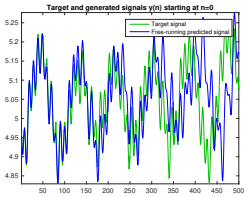
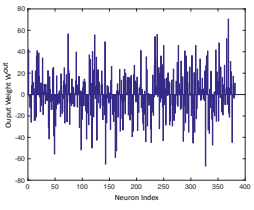
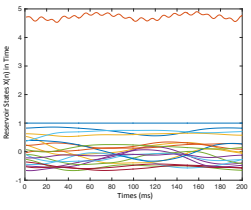
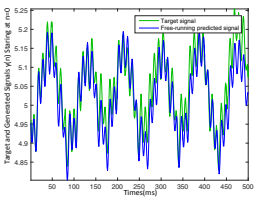
Weight Distribution	Reservoir States	Prediction Result
		

Table 5.7: Experiment results for Case 5.

Weight Distribution	Reservoir States	Prediction Result
		

(Table 5.8). In Case 5, the simulation results show that the ESN generates very robust reservoir states.

Table 5.8: Selected parameters for the prediction of motor-current

Case	Iterations	α	β	m	MSE
1	132	0.29062	0.33634	369	2.2221e-03
2	16	0.56803	0.023929	453	1.0332e-03
3	12	0.36892	0.44524	168	2.5729e-03
4	93	0.57715	0.17624	213	6.8746e-03
5	182	0.39731	0.039942	379	2.0152e-03
6	193	0.6857	0.1667	444	2.2767e-02
7	452	0.9722	0.088102	437	1.6422e-03
8	141	0.29936	0.4698	155	2.0604e-03
9	99	0.87626	0.045372	371	3.0254e-03
10	57	0.78899	0.10019	351	2.1324e-03

As the data are noisy, the range of weight distribution and reservoir states fall into $[-80, 80]$ and $[-1, 8]$ are reasonable. This has significantly impact on the prediction accuracy, which indirectly indicates that reducing the noise is an essential step for using ESN. In spite of this oscillatory behavior, the simulation demonstrates that the ZIZO algorithm can also quickly find sub-optimal global parameters for practical applications.

5.4 Extension of ZIZO Searching Algorithm

Although ZIZO worked smoothly for searching the hyper-parameters of ESNs, but if we take look of what ever happened within one of the searching processes in our experiment, it approached the optimal solutions exactly as following without any regular patterns (Figure 5.8). Since the searching process is completely random, to improve the stability of searching results, the ZIZO algorithm is extend as following (Algorithm 5.2). There are two main differences between the original version and improved one: (1) no matter what kind of loss function ESNs adopted, we add square-operation over it. (2) A global result recording table added. It is used for counting the best result among the same batch and updating the searching space for the next batch of random searching. When a program termination condition is reached, we select the best parameter within the table.

To test the improved algorithm, we re-simulated the experiment again, now the searching process smoothly reduced to a relative stable result, which is presented in the figure 5.9.

Additionally, except searching hyper-parameters for ESNs, we are now also working on solving complex equations or solving multi-equations at the same time by applying to extend ZIZO, our numerical results demonstrate its efficiency. Besides, we also try to add the idea of momentum to ZIZO for speeding up the entire searching process. The main problem of applying gradient descent

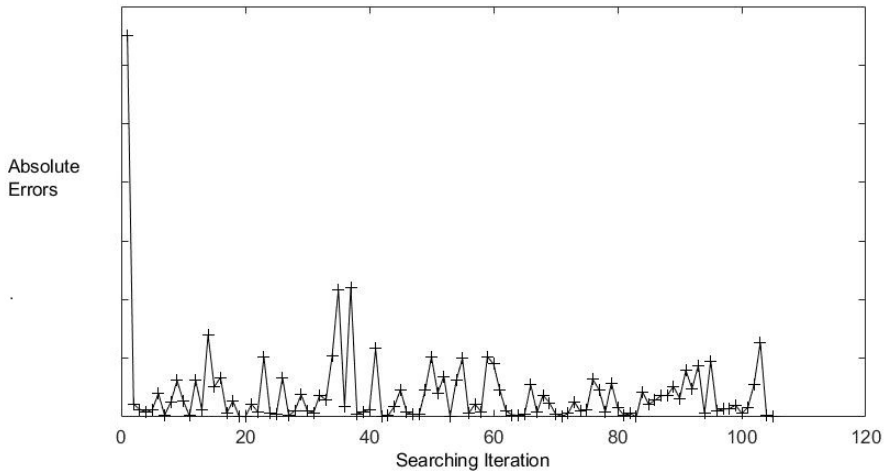


Figure 5.8: Example of original version of ZIZO: parameter searching process.

Algorithm 5.2: Extend ZIZO Searching Algorithm

Data: data set for training / validation / testing
Result: ESN Hyper-parameters
Initialize: upper and lower boundary of every hyper-parameters (searching space). Usually we set them between [-1, 1];
while *Condition is reached* **do**
  for $batch = 1, 2, ..., m$ **do**
    for $iteration = 1, 2, ..., n$ **do**
      ZIZO Searching with squared error, instead of absolute value ;
    end
    Select the best parameter pair and record in a table ;
    Update the boundaries of searching space ;
  end
  Select the best parameter pair as the final result ;
end

based back-propagation training algorithms to ESNs is that it barely can conduct the derivative operation if the size of the reservoir is too large. Manually setting up the derivative calculation, or using searching algorithm first detect the neurons connection, then execute all the derivative calculation makes it even more painful for practical applications.

5.5 Summary

This work proposes a simple but effective BS-inspired ZIZO parameter searching method for identifying the optimal parameters of ESNs. The great advantage of ZIZO is its simplicity and performance

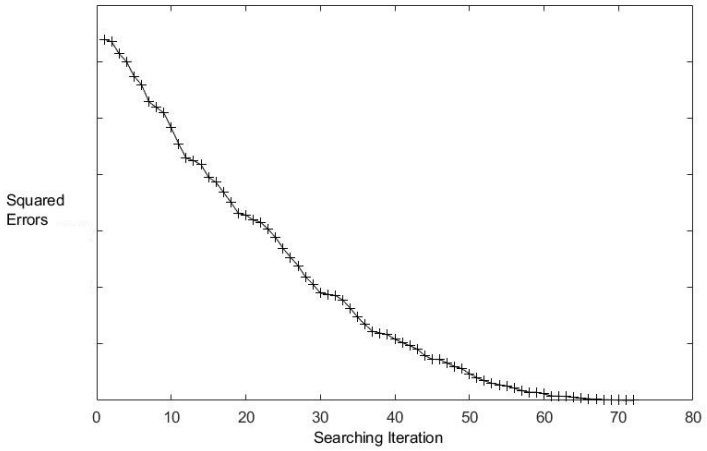


Figure 5.9: Extend version of ZIZO: parameter searching process.

in efficiently generating batches of meaningful and accurate parameters (α, β, m) with considering one specific application scenario.

CHAPTER 6

End-to-End Industry Data Processing with Deep Nets: Predictive Maintenance

Chapter 4 introduces a semi-automatic method for feature selection and milling process modeling. This chapter uses the same strategy as Chapter 4 to present the work. It addresses how to automatically generate useful features and model sensory inputs at the same time. The idea of a general technical framework for monitoring and predicting the health status of rotary machines is presented.

6.1 Principle of Automatic Bearing Machine Monitoring

This section introduces the core idea of automatic machine status monitoring and prediction methods for bearing machines. In one run-to-failure bearing machine experiment, a trained sparse autoencoder is adapted to extract unsupervised features from a group of vibration sensory data. It is then assumed that the status of the machine at the starting point is healthy. In that case, the correlation between initial states and succeeding data is calculated. The correlation calculation results are smoothed through the moving-average filter. The final normalized outputs from the filter are used as an indicator of the machine health status. It can precisely locate the degradation starting place. Through several experiments, it is demonstrated that the autoencoder correlation-based (AEC) method works well and its prediction accuracy is higher compared to that of existing models. The advantage of this work is that AEC can automatically generate rich unsupervised features for machine sensory data. The

comparison demonstrates that AEC is superior to many other state-of-the-art techniques for modeling and predicting the health status of bearing machines.

6.2 Health Management

Machine health management is essential for most industries. One of the key activities in health management is condition-based monitoring, which detects abnormal states in machines by analyzing sensory data and machine parameters. Specifically, it first analyzes the sensory data and extracts useful information from the collected data. In general, condition-based monitoring is constituted by two important steps: first extracting features from a run-to-failure experiment, then monitoring the defect propagation and predicting at which time point in the future the system will begin to degrade. Plenty of models have been developed for monitoring the performance of key machine components. In most of these, data from sensors installed on the machine is collected and used to generate handcrafted time and frequency-domain features. These features are then used directly or indirectly by following methods for predicting the health conditions of target machines or components [LWZ+14]. Recent development in AI technologies enable us to solve such issue much more efficiently [JLL+16, SSZ+16, TDS+16]. Advanced AI techniques raise the possibility of generating high-quality features for sensory data, specifically generating useful neural features for post modeling procedures. However, AI methods are still not applied in a fully automated fashion for solving condition monitoring problems, without the use of prior experience or knowledge.

The ideal health condition monitoring and prediction methods should be able to automatically extract useful unsupervised features from sensory data without considering the size of the dataset. This kind of data-driven method should be able to run both online and offline, predict the future status of systems, and locate the degradation starting points. It would make the model work better if the proposed method could automatically combine the steps of condition-based monitoring (CBM) as a seamless model and require no human effort. More importantly, the ideal method has a unique technical framework and could easily be applied to prognostics for various machine components, such as bearing parts, gears, and spindles.

In this work, a novel prognostic approach is developed for bearing machines. It is called the "AEC prognostic algorithm" and has the majority of the characteristics of the ideal CBM method described above. AEC is a trained sparse autoencoder for extracting unsupervised features from collected sensory data, specifically data collected in several run-to-failure experiments. The Pearson correlation is then calculated based on the extracted features of initial samples, with the future data points. A moving-average filter is adapted for smoothing outputs of the Pearson filter. AEC then finally normalizes the filter's outputs and applies them as machine health condition indicators. To

demonstrate this proposed idea, the AEC runs on the collected sensory data of several run-to-failure experiments from bearing machines. The experiment results show AEC's superiority in finding the degradation starting point compared to the existing methods.

To clearly illustrate the idea, Section 6.3 presents the design method of the AEC algorithm, including the neural structure of the autoencoder. Section 6.4 presents various experiment results generated by AEC. Section 6.4 quantitatively and qualitatively compares AEC with other existing techniques. Section 6.5 summarizes this work.

6.3 Autoencoder Correlation-Based Fault Prognostic Method

This section begins with an introduction to the technical framework of the automated AEC general fault prognostic method within the run-to-failure test of bearing machines. Figure 6.1 presents the technical structure of AEC: an autoencoder trained using vibration sensory data samples. The autoencoder's application of a well-designed neural structure generates a neural presentation for inputs of each sampled sensory data. The correlation coefficient matrix among past samples and current ones is then calculated. The correlation coefficients of the features extracted for the samples generated at the beginning of the run-to-failure process, along with the other available samples, are normalized and correspondingly filtered utilizing a moving average (MA) filter. Its outputs can be used for predicting the state of the system at each sampling time step. In the following part, the steps for designing the AEC are introduced in detail.

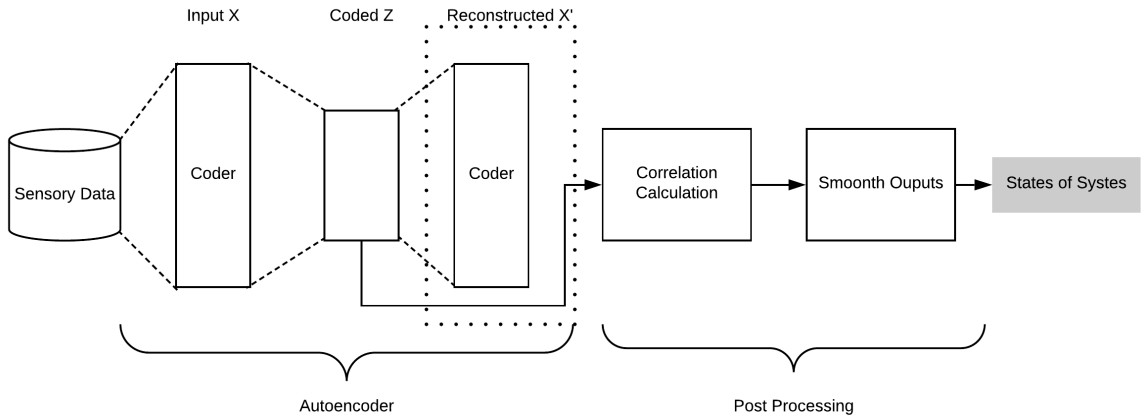


Figure 6.1: The architecture of the AEC fault prognostic method. A single-layer autoencoder is trained on M-input vibration samples (here, four networks are designed for four different datasets). Correspondingly, the correlation coefficient of the extracted features from each input sample is calculated. The correlation coefficient of the first recorded vibration sample with the other sample-sets is normalized between 0 and 1 and is fed into a moving average filter. Such structure outputs of a rate between 0 and 1 correspond to the health status of the system under test.

6.3.1 Sparse Autoencoder Revisit

An Autoencoder [BC+08] is an artificial neural network learning to copy the inputs to outputs with coded essential information or features. For practical application, the structure of the autoencoder is data dependent. Within appropriate layers and neurons, the essential features of collected sensory data can be generated from the middle layer of autoencoder.

Accordingly, one can define $x \in \mathbb{R}^{D_x}$, which is a D_x dimension input to the Autoencoder. The encoder initially maps x to a lower dimension vector $z \in \mathbb{R}^D$, and correspondingly generates an estimation of x, $\hat{x} \in \mathbb{R}^D$:

$$z = f(Wx + b_1); \;\; \hat{x} = f(W^T z + b_2), \tag{6.1}$$

where f is a saturating linear transfer function in the designed technical framework denoted in Equation 6.2, $W \in \mathbb{R}^{D_x \times D}$ stands for the weight matrix, and vectors $b_1 \in \mathbb{R}^{\mathbb{D}}$, $b_2 \in \mathbb{R}^{\mathbb{D}_\frown}$ represent the bias values.

$$f(z) = \begin{cases} 0, & \text{if } z \leq 0 \\ z, & \text{if } 0 < z < 1 \\ 1, & \text{if } z \geq 1 \end{cases} \tag{6.2}$$

The formula below is the cost function to be subsequently optimized:

$$\begin{aligned} C = & \underbrace{\frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{I} (x_{in} - \hat{x}_{in})^2}_{\text{mean squared error}} + \\ & \underbrace{\lambda R_{L2}}_{L_2 \text{ regularization}} + \underbrace{\sigma R_{sparse}}_{\text{sparsity regularization}} . \end{aligned} \tag{6.3}$$

The first item in C represents the mean squared error. The second one is a L_2 regularization, which is used for tackling the issue of over-fitting, where λ is the regularization coefficient. L_2 regularization is computed as follows:

$$R_{L2} = \frac{1}{2} \sum_{j}^{n} \sum_{i}^{k} (w_{ji})^2, \tag{6.4}$$

where n and k are the number of samples (observations) and number of variables in the training data, respectively.

The third item in the cost function determines a sparsity regularization with an effectiveness coefficient σ. The average output activation of a neuron i is denoted as follows:

$$\hat{\rho}_i = \frac{1}{m} \sum_{j=1}^{m} f(w_i^T x_j + b_i), \tag{6.5}$$

and define ρ as the desired output value of a neuron. The difference is measured by applying the Kullback-Leibler divergence function, which is described as follows:

$$R_{sparse} = \sum_{i=1}^{D} KL(\rho \| \hat{\rho}_i) = \sum_{i=1}^{D} \rho \log(\frac{\rho}{\hat{\rho}_i}) + (1-\rho)\log(\frac{1-\rho}{1-\hat{\rho}_i}). \tag{6.6}$$

KL function outputs zero when ρ and $\hat{\rho}_i$ are close to each other and increases when they diverge from one another, which implies sparsity in the network, and it is defined in the cost function as the sparsity regularization R_{sparse}. Finally, the autoencoder neural network is trained by a scaled conjugate gradient algorithm [Møl93] within the MATLAB neural network toolbox.

6.3.2 PCA, Variational Autoencoder, and Sparse Autoencoder

Principle component analysis (PCA) is a linear - reduction method for transforming the high dimension data into lower case. It selects top k linearly independent variables as the principal components which reserve most of the important information of the original data set. Let's briefly show the general steps of implementing a PCA :

- Collect training X_{tr}, validation X_{val} and testing data X_{te}.
- Subtract the mean of X_{tr}. In order to let PCA works properly, we first need to calculate the mean values of every features. Let's suppose X is a 2-dimension data set and define the mean values as $\bar{X_1}$ and $\bar{X_2}$.

  $$\bar{X} = [\frac{\sum x_{i1}}{M}, \frac{\sum x_{i2}}{M}] \tag{6.7}$$

  where i is the index of sample, M is the number of samples in X_{tr}.
- Calculate the covariance matrix.

  $$Covariance(X_1, X_2) = \frac{\sum (X_{i1} - \bar{X_1})(X_{i2} - \bar{X_2})}{M-1} \tag{6.8}$$

  Similaritly, we can also calculate covariance (x_1, x_1), (x_2, x_1) and (x_2, x_2). Putting them together is the covariance matrix Cov:

  $$Cov = \begin{pmatrix} x_1x_1, x_1x_2 \\ x_2x_1, x_2x_2 \end{pmatrix} \tag{6.9}$$

  where, we assume $M = 2$.

- Calculate the eigenvectors and eigenvalues of the covariance matrix. The eigenvalues is calculated through solving following equation:

$$|X - \lambda I| = 0 \tag{6.10}$$

  where λ is the eigenvalues. The eigenvectors is then calcuated as:

$$|X - \lambda I|\beta = 0 \tag{6.11}$$

  The non-zero solution of β is the so-called eigenvectors.

- Choosing components and forming a feature vector. This step is strongly associated with the applications, i.e., how much original information the users would like to reserve for the applications.

- Test the selected features on validation and testing data and verify the generalization of these features.

As a summary of the technical character of PCA, "linear transformation" is the keyword. It linearly shrinks the original data set and uses lower dimension features (not generate new features, keep some of the most relevant ones) to represent the entire data set.

Variational Autoencoders (VAEs) are essentially generative models. The structure of them is quite similar to Sparsity ones. The core idea of VAEs is to learn the probability distribution of the original data set, instead of just simply fitting the desired outputs. Mathematically, this can be described as follows:

$$P(X) = \int P(X|Z;\theta)P(Z)d_z \tag{6.12}$$

Where $P(X)$ is the probability distribution, and Z is the lower dimension representation called latent space. θ is the parameter need to be adjusted so that later we can regenerate some samples which also fit the same distribution as original data set.

Variational and sparsity autoencoders are neural network-based methods. One of the hyper-parameters that we can manually select is the activation function. For sparsity autoencoder, if the activation function is a linear one, then there is no difference between PCA and sparsity autoencoder. In our work here, we used the saturating linear transfer function, which is a non-linear function. As the health states evolving procedure of bearing components are essentially is a non-linear process. Therefore, here adopting saturating activation is reasonable. Compared with sparsity autoencoder, variational autoencoder seems to have higher precision. While, in our case study, the goal is to monitor and predict the time-series signals, instead of generating new health state. Hence, variational autoencoder is not applied in this part of the work.

6.3.3 Correlation Analysis, Normalization, and Filtering

During the training process, each sample is adopted for generating D nonlinear features, specifically constructed by the autoencoder. The linear dependencies of the abstract representation of each sample are calculated by applying PCC, which is described below for samples A and B:

$$r(A,B) = \frac{1}{D-1}\sum_{i=1}^{D}(\frac{A_i - \mu_A}{\sigma_A})(\frac{B_i - \mu_B}{\sigma_B}), \tag{6.13}$$

where μ_A and σ_A are the mean and the standard deviation of A, respectively, and μ_B and σ_B are the mean and standard deviation of B [Fis25]. The correlation coefficient matrix for the available samples during the run-to-failure test is calculated. The first column of the CC matrix represents the correlation of the first sample data recorded at the beginning of the run-to-failure process, with the other available samples. This vector is normalized between 0 and 1 and set as the criteria for predicting the next samples and detecting whether the status of the system is in a healthy state or not.

Finally, as mentioned above, the shape of the output is smoothed by passing it through a moving average filter. The filter is designed as follows:

$$\hat{m} = \frac{1}{w_{size}}(y(n) + y(n-1) + \cdots + y(n-(w_{size}-1))). \tag{6.14}$$

For the sample data y, the filter slides a sample-window of length w_{size}, over the data and calculates the average of the covered data in each window.

6.4 Case Study with AEC

In this section, the performance of the proposed AEC algorithm is demonstrated by running it on several run-to-failure bearing experiment data. The dataset and main target of the work with the testing dataset are explained, and then based on the experiment results, the performance of the AEC method is shown. Finally, there is a comparison of this work with the state-of-the-art techniques in fault prognostics of bearing machines.

6.4.1 IMS Bearing Dataset from PCoE NASA Datasets

The experiments used data collected from the Center of Intelligent Maintenance Systems of the University of Cincinnati [LQY+07], shared by the Prognostics Data Repository of NASA [PCo16]. The experiment setup is depicted in Figure 6.2. A shaft is coupled to an AC motor rotating at 2000 RPM, with a 6000lbs load installed on it. Four force-lubricated bearings are mounted on the shaft. Accelerometers with high sensitivity are placed for each bearing for recording the vibrations (the

position of the sensors is shown in Figure 6.2). Three test-to-failure experiments are performed separately. According to the tests, critical damage usually happened at the end of the test [QLLY06]. In the first experiment, two accelerometers are used for each bearing, while for the other two experiments, only one accelerometer is applied. Datasets contain one-second recordings from the accelerometers with a sampling frequency of 20KHz, every 10 minutes during the run-to-failure tests [QLLY06]. Table 6.1 represents the properties of the collected data in each experiment.

Table 6.1: IMS Bearing tests specification

Experiment	**# of Samples**	**sample size**	**Faulty Bearing**	**test-to-failure time**
S1	2156	4×20480	B3 and B4	35 days
S2	984	4×20480	B1	8 days
S3	4448	4×20480	B3	31 days

Within the experiments, the autoencoder neural network is only trained on the faulty bearings. Therefore, there are four different simulated experiment cases: (1) dataset one bearing 3 (S1B3), (2) dataset 1 bearing 4 (S1B4), (3) dataset 2 bearing 1 (S2B1), and (4) dataset 3 bearing 3 (S3B3).

6.4.2 Results

AEC is implemented in Matlab, and the autoencoder is trained with Matlab. Matlab has high compatibility with most production systems. It is particularly well designed for online test-to-failure

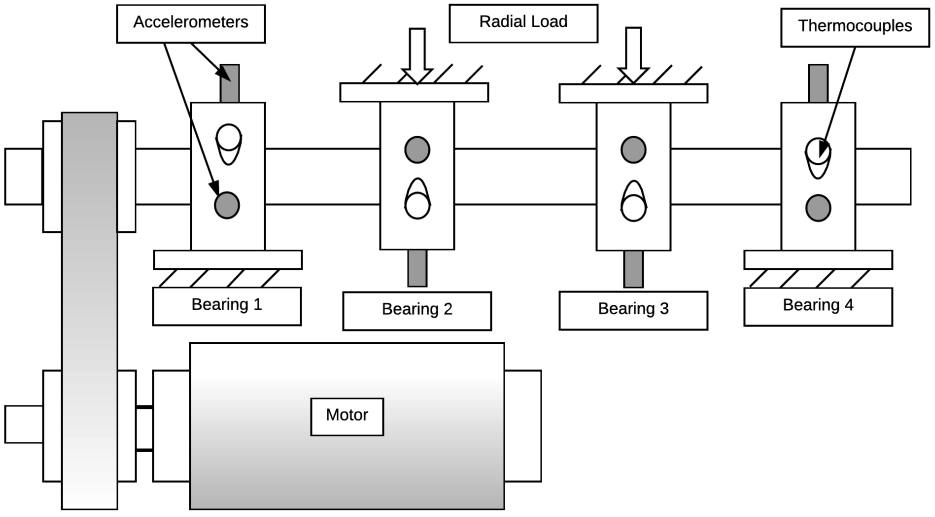


Figure 6.2: Bearing test implementation [QLLY06]. Four bearings are set-up on a shaft and vibration data is collected in three run-to-failure tests.

numerical simulation. The training process is performed on a Microsoft Azure NC-Series virtual machine powered by one NVIDIA Tesla K80 GPU. In order to illustrate the performance of the method, two general frameworks are presented. In the first one, the autoencoder is trained with all of the datasets for every experiment. In this case, each experiment is performed to monitor the status of the bearing machine. In the second framework, autoencoder is trained with 70% of the data. The remainder of the data is kept for testing, i.e., predicting the future status of the bearing machine if only part of the data was made use of and checking the prediction accuracy. According to the feedback from Matlab, each experiment takes between 55 minutes and 80 minutes, which is strongly connected with the size of the training and testing dataset.

Since only the vibration samples of one bearing in each simulation are considered, the input sample dimension, under a sampling frequency of 20 kHz, is a 20480-length vector. One thousand hidden units were chosen for the AE, which enabled the extraction of 1000 features from each large input vector. The correlation coefficient matrix of the input samples is then calculated according to the

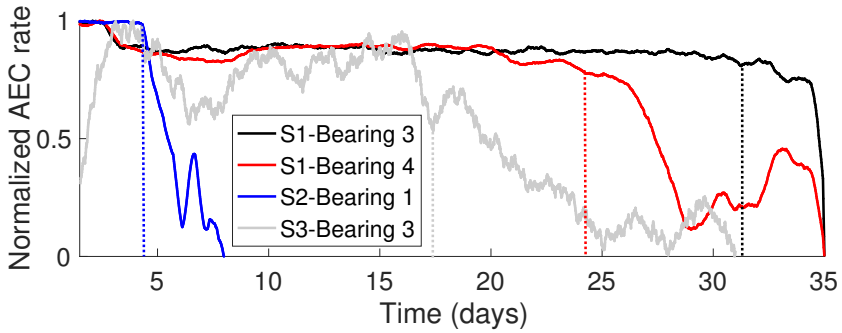


Figure 6.3: AEC evaluation rate for the different datasets. For every dataset, the AEC platform generates a rich trend corresponding to the status of the bearing based on the vibration sensory data. One can feasibly detect abrupt changes in the AEC rate. The vertical dot lines correspond to the sample time after which AEC platform starts decreasing dramatically, which indicates the degradation starting point.

framework in which the researchers were working. For the first case, all of the available data is fed in. For the second framework, 70% of the data is dedicated to training, and the network had to output a prediction on the status of the system based on the previously observed samples. The correlation coefficient of the initial sample is normalized (which is considered to represent the health status of the system) with the next samples. The output is then filtered to provide a representation of the status of the system.

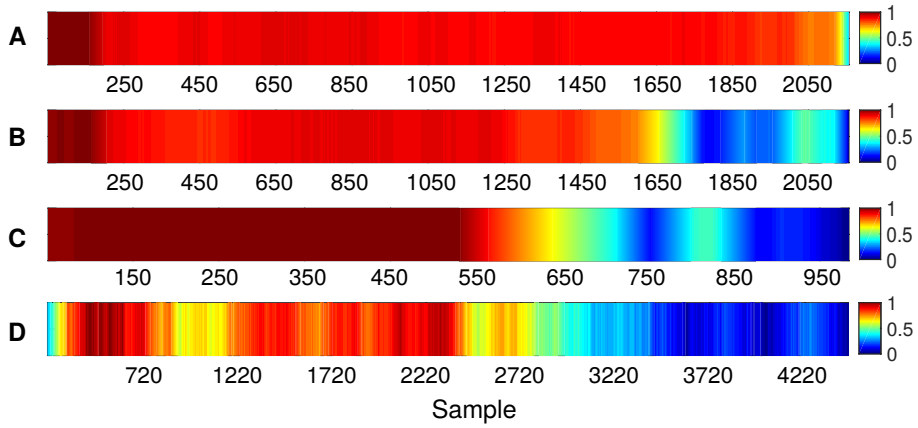


Figure 6.4: Visualization of the status of four different bearings in four run-to-failure experiments. The color bar represents the value of the normalized AEC rate. The higher the rate (the redder is the color bar), the less probable it is for the system to be in a weak condition. The horizontal axes show the samples collected for the run-to-failure test in every experiment: (a) S1B3, AEC rate recording; (b) AEC for S1B4; (c) AEC for S2B1; and (d) AEC for S3B3. Note that traditional statistical analysis on this dataset has failed to provide accurate bearing status monitoring, due to the high level of noise corrupting the vibration data. However, the AEC platform provides a clear representation of the status of the system.

Framework1 - Health-Condition Monitoring

Figure 6.3 presents the system monitoring outputs of the AEC for the four different simulations on three run-to-failure cases. In the Figure, the high AEC rate represents the healthy behavior of the system, while a decreasing trend indicates the starting point of an abnormal state. The high AEC means more samples correlated with the initial health status of the system. The AEC identifies precisely the beginning of a weak trend, together with its propagation effect. It is worth noting that with the noisy dataset collected (for instance, the dataset S3B3), many existing works could not successfully monitor the health degradation trend. However, with this approach, the Figure clearly shows that it works well. Figure 6.4 offers a color bar visualization of the status of the four experiment results for S1B3, S1B4, S2B1, and S3B3 in parts A, B, C, and D, respectively. The color bar illustrates the AEC rating of the health states, ranking them between 0 (blue) and 1 (red). One can easily identify where a significant change is happening, allowing engineers or managers to stop the running process. A sample is set as abnormal when its AEC rate is measured 90% below the recorded sample at t-100. The AEC indicates the start of a weak state and represents its gradual propagation.

Table 6.2 compares the monitoring performance of the AEC method with that of some existing data-driven monitoring methods, applying the same dataset (the IMS dataset). The monitoring of a system's performance is conducted using the sample time (sample number) at which the algorithm

catches the start point of a degradation. The earlier the fault detection, the better the monitoring performance.

Table 6.2: Detection performance. HMM-DPCA: Hidden Markov model with dynamic PCA, HMM-PCA: Hidden Markov model with PCA [Yu12a]. MAS-Kortusis: Moving average spectral kurtosis [KPK+16]. VRCA: Variable-replacing-based contribution analysis [Yu12a]. - Means that the dataset has not been analyzed in the corresponding experiment

Algorithm	S1B3	S1B4	S2B1	S3B3
	Degradation starting datapoint			
AEC	2027	1641	547	**2367**
HMM-DPCA	2120	1760	539	-
HMM-PCA	-	1780	538	-
RMS	2094	1730	539	No detection
MAS-Kurtosis	1910	1650	710	No detection
VRCA	-	1727	-	No detection

The conclusion can be reached as that only the AEC can identify the degradation starting point for the experiments. It is worth noting that detecting the test-to-failure process of Experiment S3B3 is considered a hard challenge to perform. Accordingly, there are few approaches which can capture such a state.

Framework2 - Online Prognostic

In this section, an online mode of AEC is introduced for predicting the system degradation trend. Specially, the autoencoder is trained by adopting the first 70% of the collected data to evaluate the prediction performance of the model in an online monitoring phase. Six different cases are studied: S1B3-sensor1, S1B3-sensor2, S1B4-sensor1, S1B4-sensor2, S2B1, and S3B3; where Figures 6.5A to 6.5F represent the predicted AEC rates in each experiment, respectively. A sample is defined as an abnormal state where the AEC rate reaches 90% of the rate of the 100 steps earlier sampled. The prediction process begins with the samples collected from day five onwards, and as previously mentioned, the status of the system is considered to be normal [QLLY06] at initial stage. Using this definition, the degradation starting point is calculated with a high level of accuracy and the propagation of fault is captured during the simulated run-to-failure test. Table 6.3 summarizes the predicted degradation starting point, together with the prediction accuracy of each experiment. The prediction error is computed as the ratio of the deference between the predicted sample and the monitored fault starting point, to the total number of samples in each experiment.

Table 6.3: Prediction accuracy of the AEC method in different bearing run-to-failure tests

Experiment	Fault starting point	Prediction Accuracy
S1B3-sensor1	2120	95.68%
S1B3-sensor2	2122	95.59%
S1B4-sensor1	1681	98.14%
S1B4-sensor2	1673	98.51%
S2B1	610	93.60%
S3B3	2435	98.47%

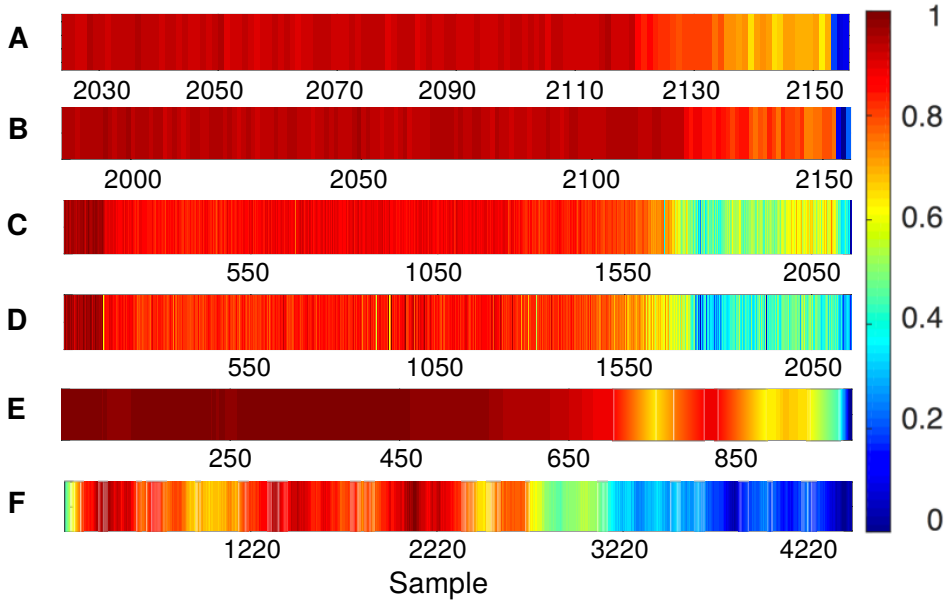


Figure 6.5: Online monitoring of the status of the system in different experiments: (a) S1B3 first sensor, (b) S1B3 second sensor, (c) S1B4 first sensor, (d) S1B4 second sensor, (e) S2B1, (f) S3B3. The color bar represents the AEC rate. The degradation starting point of the system in each experiment is predicted with a high level of accuracy and the AEC rate comprehensively elucidates how the fault propagates during each run-to-failure test.

AEC is compared with most of the existing methods for fault monitoring and prognostic systems concerning several attributes. These are defined as follows:

- Generalization: Ability of the method to provide quick results for various bearing status monitoring experiments.
- Status Monitoring: Ability of the method to provide a useful health-status trend during the

Table 6.4: Qualitative comparison of the performance of the existing approaches of the bearing dataset prognostic. HMM-DPCA: Hidden Markov model with dynamic PCA; HMMPCA: Hidden Markov model with PCA [Yu12a]. MAS-Kortusis: Moving Average Spectral Kurtosis [KPK+16]. VRCA: Variable-replacing-based contribution analysis [Yu12a]. EET: Energy Entropy trend [KPK+16]. WPSE-EMD: Wavelet packet sample Entropy [WZZS11] - Empirical mode decomposition [LHZH07]. Spectral-ANN: Third-order spectral + artificial neural networks [YSMP02]. Fuzzy-BP: Fuzzy logic with back-propagation [SS05]. SVM: Support Vector Machine [YZZ07]. GA-SVR: Genetics algorithm-Support vector regression [FZJJ09]. GLR-ARMA: Generalized likelihood ratio - Autoregressive moving average [GFD08]. ++: Highly satisfies. +: Satisfies. -: The attribute is not covered -+: The attribute is fairly covered.

Algorithm	Generalization	status monitoring	Automated	unsupervised	Detection sensitivity	fault-type detection
AEC	+	+	+	+	++	-
HMM-DPCA	-+	+	-	+	+	-
HMM-PCA	-	+	-	+	+	-
RMS	-	+	-	-	-+	-
Kurtosis	-	+	-	-	-+	-
MAS-Kurtosis	-	+	-	-	-+	-
Spectral-ANN	-+	-	-	-	+	+
VRCA	-+	+	-	-	++	-
EET	+	+	-	+	-	-
WPSE+EMD	-	+	-	-	+	-
Fuzzy-BP	-+	-	+	-	+	+
SVM	-	-	-	-	+	++
GA-SVR	-	+	-	-	+	-
GLR-ARMA	-	+	-	-	+	-+

test-to-failure experiment of the bearings.

- Automated: A fully autonomous fault prognostic method.
- Unsupervised: Capability of the method to extract information from the sensory without any prior knowledge.
- Detection Sensitivity: Ability of the method to provide a fast-enough alert on the starting point

of the degradation in the test.

- Fault-type Detection (Diagnostics): Ability of the method to detect a certain type of defect in the system and classify them into different fault classes.

Table 6.4 comprehensively illustrates a qualitative comparison of various fault prognostic methods utilized for bearings, based on the attributes as mentioned earlier. The assessment of the performance of each method is carefully performed based on the provided results and the detailed specifications of the methods in their corresponding report. Following such an evaluation, the results suggest the superiority of the AEC algorithm over existing methods where it can precisely capture the degradation starting point and provide a useful trend for the fault spread while working autonomously.

6.5 Summary

A new autonomous technique for fault prognostics in machine bearings was introduced. The method is based on unsupervised feature extraction from sensors utilizing an autoencoder. A correlation analysis is performed on the extracted features, and a useful trend is corresponding to the status of the bearing during the test-to-failure was provided, which shows that AEC successfully monitors the status of the bearings in various experiments, which confirms its generalizability. AEC accurately predicts the degradation starting point, while highlighting an informative trend in propagation. Furthermore, the AEC algorithm generates rich unsupervised features from the vibration data and is automated.

Future work could improve the quality of the method and apply it to the prognostics of crucial other machine components, such as gears, cutting tools, and spindles. One could also identify a more general solution where a deep convolutional autoencoder was designed and trained accordingly to evaluate the health status of a system independently, without any other follow-up steps.

CHAPTER 7

End-to-End Industry Data Processing With MBRC-RNN: Fast Data Modeling

Modeling high dimension sensory data (such as milling process sensory data) is a challenge for cyber-physical production systems. There are several reasons make high dimension industrial data processing so difficult, for example: (1) the complexity of inputs. (2) The complicated methods of sensory data analysis. In this research, an end-to-end industrial sensory data analysis framework is proposed. It is a multi-bias randomly connected RNNs, which makes use of the recurrent structure of an artificial neural network and the property of multi-bias implementing high efficiency and accurate modeling performance. Given the recurrent structure of the proposed multi-bias RNNs, the parameters searching algorithm ZIZO is applied in order to quickly obtain a group of appropriate parameters based on the training and test data adopted for the work. To evaluate the proposed method, experiments are conducted using empirical data labeled with tool wear-degree. The experiment results demonstrate that the technique could generate more accurate results in a more efficient way than existing works.

7.1 End-to-End Industrial Data Processing

Chapter 2 presented reviews of much neural networks-based work on modeling industrial sensory data. These methods each adopted advanced optimal methods and feature-extraction techniques for

improving the modeling performance of neural networks. Through practical data simulation analysis, these have been shown to work well for real-world problems, though the adopted optimization methods themselves have numerous hyper-parameters to solve when attempting to resolve the problems. In effect, they introduce additional parameters that make the problems even more difficult to solve, for instance, the number of principal components in PCA techniques and the number of neighbors for the K-means algorithm. Generally, the main weakness of the reviewed methods is that they are difficult to implement due to a large number of hyper-parameters of external models that need to be adjusted for different applications.

To tackle the abovementioned issue for neural network-based applications, a multi-bias random RNNs is proposed for CPPS to model the sensory data generated by multi-industrial sensors. MBRC-RNN provides an end-to-end manufacturing systems modeling and tool-wear degree prediction service. Users are required only to evaluate the different sizes of the application datasets gradually. More precisely, a three-layer RNNs is generated in the first step, constituted by the input layer, a randomly generated hidden layer, and a readout layer (output layer). It worth noting that the neural connections among the middle layer (reservoir layer) are randomly generated, which is entirely different from the classical RNNs. In this work, the input layer has 5,400 variables with one additional bias. Neurons in the reservoir layer receive inputs not only from the input layer but also from other neurons that have direct neural connections with it. Organizing the RNNs in such a structure dramatically reduces the size of the neural network and the information confusion and ensures that modeling prediction works better.

With random connections, one neuron in the reservoir layer may receive too many inputs from others. To avoid the overfitting issue, the initial neural weights are defined as falling to the range $[-0.001, 0.001]$. Afterward, the parameter searching algorithm ZIZO is adopted to explore the appropriate weights for RNN [XLX+17].

Multi-bias is one of the essential properties of the proposed MBRC-RNN. Besides the only one bias adopted by traditional neural networks, additional features from the dataset are also considered as biases, for example, the depth and speed of one specific cutter or the type of materials cutters work on. These are treated as biases because features like those as mentioned above are not changed in one specific period. Thus they are called static features here. It is not appropriate if one merges static features with quickly updated sensory data, such as directly mixture vibration signals and noise signals with a type of materials.

Static features working together with dynamically updated sensory data significantly affect tool wear degree. Specifically, the dynamically changed sensory data is sensitive to short-term systems status while the static features look much more ahead and decide the long-term performance of systems.

Therefore, this work considers static features as multi-bias and combines them with the outputs of the reservoir layer. The following step is to model the performance of the system with the last step mixed signals. The modeling method is called a linear combination and is trainable with straightforward gradient descent. To evaluate the performance of MBRC-RNN, a milling sensory dataset is applied, as the system status changes quickly and massive data flows into the database in a short time. The milling dataset is collected from NASA data repository [AA07]. The experiment results show that MBRC-RNN can precisely and efficiently model complex milling process-generated sensory data.

The main contributions of this work are presented as follows: (1) a novel RNNs structure for modeling milling sensory data is proposed. (2) ZIZO is applied to explore neural parameters, because the gradient descent technique is mostly applicable to regular RNNs. While, for random RNNs with bi-direction connections, it usually does not work well.

The remaining sections are organized as follows. Section 7.2 presents the general structure of MBRC-RNN with the steps required to apply it to model milling sensory data. In Section 7.3, several experiments are conducted, applying the MBRC-RNN and relevant experiment results. In Section 7.3, qualitative and qualitative explanations of the characteristics of MBRC-RNN are presented.

7.2 MBRC-RNN-Based Modeling Method for Milling Sensory Data

This section introduces the design of MBRC-RNN for modeling milling sensory data. Figure 7.1 depicts the general neural network structure of MBRC-RNN. In this work, the collected milling sensory data is used for training an MBRC-RNN model. Input u is constituted by two parts: one is directly fed into the colorful (blue $\&$ red) and randomly generated recurrent layer, the other part is simply copied to the bias [b_1, ..., b_p] (all the dash lines in Figure 2 stand for copy operation). Afterwards, ZIZO [XLX$^+$17] is adopted to generate one group of weights for RNN, then linear combination is applied to train the readout layer. The prediction error E - Mean Squared Error (MSE) between the targets y^{target} and the trained outputs y is finally calculated. If E reaches predefined threshold, then terminating the program immediately.

The main function of the colorful hidden RL is to reduce redundant input information and extract more useful knowledge of systems. MBRC-RNN is specially designed for merging dynamic and static sensory data. Modernly advanced sensors can generate massive amounts of data in short time intervals, for example, a vibration sensor can generate 20,480 samples in one microsecond [HWG17]. The collected sensory data contains valuable information about the systems. It is not usually a wise choice to feed the multi-time-scale directly into a neural network as these data are updated in different

time scale and are thus full of noise [Section 7.2]. Besides, training a neural network with inputs in high dimension is significantly time-consuming. The mathematical model of MBRC-RNN is presented in the following section. Additionally, a ZIZO-searching algorithm is also introduced for understanding the principles behind exploring the hyper-parameters of MBRC-RNN.

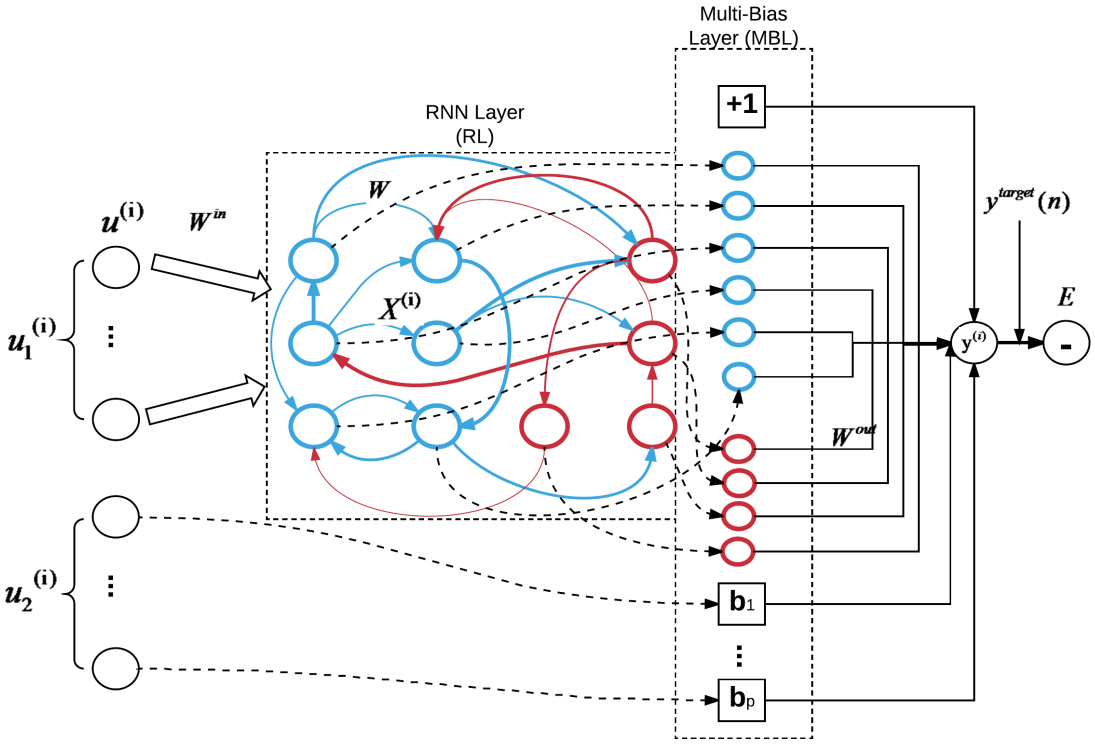


Figure 7.1: Multi-bias randomly connected recurrent neural network (RNN) structure. Blue neurons represent excited neurons, and red ones inhibited neurons. Bold connections mean neurons have a larger impact on others. Thick ones mean lower weight values. This figure only demonstrates how MBRC-RNN works, and it is unnecessary to place all inhibitory neurons in the right part of this figure.

7.2.1 Formalism of MBRC-RNN

A MBRC-RNN model is trained for modeling industrial multi-sensor data, and in this work, it is used for predicting milling tool wear degree. The input signals $u^{(i)} = [u_1^{(i)}, u_2^{(i)}]^T$ (i, the ith instance). u_1 represents different types of sensory data, e.g., spindle vibration and acoustic emissions. Milling sensory data contain critical underlying information about cutters, thus extracting useful features or information is a good start. Here, one first makes use of the structure of RNNs to transform input signals into much lower dimensions and extract the useful information represented by hidden layers

with small-sized neurons. u_2 is static input in a predefined operation period. It is the configuration of systems for different application scenarios. The output of the RL is governed by Equations 7.1 and 7.2.

$$X^{(i)} = f(W^{in} \cdot u(n)^{(i)}) \tag{7.1}$$

$$x_n = f(W \cdot X^{(i)}) \tag{7.2}$$

where $X^{(i)} = \{x_1, x_2, ..., x_n\}$ is the state of RL, x_n is the nth neuron's state. W is the weights of recurrent connection, W^{in} is the weights from the input layer to RL. W in Equation 7.2 is a key component in MBRC-RNN. It is used to receive the status information of neighbor neurons. The states of one neuron represent partial information from inputs. f is the activation function, and the possible choices could be a sigmoid function, hyperbolic tangent function, or rectified linear units (Relu), depending on the application. In this work, sigmoid is used as the activation function, and all inputs are normalized into the range [0,1] in later experiments, and the flank tool wear degree is greater than or equal to 0 in the case studies. The recurrent connection used here is intended to model high dimension inputs efficiently. This type of neural network is inspired by biological RNNs, as the way that neurons communicate with each other mimics the neural systems of animals. With such a connection, neurons need less effort to acquire additional information as they can receive the information from their neighbor neurons, in contrast to other existing artificial neural network techniques; for example, deep neural structure. In a MBRC-RNN, each neuron shares its knowledge with others via biological connections. Generally, the benefit of applying recurrent biological connection is the improvement in information presentation capability for each neuron and the simultaneous shrinking of the size of the entire neural network. It is also worth noting that: if the number of hidden layer neurons is greater than the predefined threshold, the neural network will quickly adapt to stable states (in effect, a trained model). While the performance of generalization is worse than the training results, which indicates that too many neurons added to the hidden layer, which leads to too much information overlapping among hidden neurons.

In order to accurately model the tool wear status, it is necessary to stack static features together with the outputs of RL. The mathematical operation is formalized as Equation 7.3. The multi-bias is essential for modeling physical process-generated data, as it determined the basic picture of sensory data. For example, the type of materials and feed speed are static parameters of systems in a specific period, and they represent the background of the collected data. These static properties are therefore modeled as multi-bias within the neural network.

$$X_{mbl} = [X^{(i)}; b_1, ..., b_p] \tag{7.3}$$

The readout layer makes use of the reformed outputs X_{mbl} as inputs. Accordingly, RL layer extracts essential and relative independent information from inputs. Each neuron in the MBL layer represents

one key component of the entire inputs. Hence, in the readout layer, the linear combination technique is applied, specifically ridge regression, for modeling the W^{out}. The prediction outputs are generated by Equation 7.4.

$$W_{out} = (R^{(i)} + \alpha^2 \cdot I)^{-1} \cdot P^{(i)} \tag{7.4}$$

$$R^{(i)} = (X^{(i)}_{mbl})^T \cdot X^{(i)}_{mbl} \tag{7.5}$$

$$P^{(i)} = (X^{i}_{mbl})^T \cdot (y^{(i)})^{target} \tag{7.6}$$

where α^2 is a non-negative constant value. It is predefined as $1.0e-8$ in this work. I is an identity matrix. R is the correlation matrix of X_{mbl}. P is the cross-correlation matrix of between X_{mbl} and the desired outputs.

Finally, in order to terminate the searching and training process, MSE is measured within training process and stop the program when it falls below than predefined threshold. MSE is calculated as Equation 7.7.

$$E(y, y^{target}) = \frac{1}{M}\sum_{i=1}^{M}(y^{(i)} - (y^{(i)})^{target})^2 \tag{7.7}$$

$$y = g(W_{out} \cdot X_{mbl}) \tag{7.8}$$

where M is the number of instances, y is the predicted outputs. g is the readout activation function, in this work, linear function $g(X) = X$ is adopted, because the tool wear degree of some experimental cases is great than 1. It would not work if one applies sigmoid or tangent function for readout layer. Another possible option is using Relu, but this is inefficient as one already knows all of the damage degrees greater than or equal to 0. Applying a linear function for the readout layer is therefore reasonable.

7.2.2 Searching Appropriate Values for Input Weights W^{in} & RL Weights W

Training RNNs requires significant relevant experience, especially for a randomly connected RNN. Potentially, there are three possible ways of finding a set of appropriate weights for MBRC-RNN: gradient searching with back propagation, randomly generating weights works for ESNs [XLL17], and the ZIZO searching method for RNN.

Gradient search with backpropagation is highly time-consuming for RNN as the main idea is to calculate accumulated errors. First of all, one needs to feed forward inputs through the entire neural network. The following step is to compute errors between y and y^{target}. Derivative operations and backpropagation techniques are then applied, and the deviation fed back through the entire neural network. The structure of the MBRC-RNN's RL layer is similar to that of the ESN, though MBRC-RNN does not have a memory of historical inputs, unlike ESN. Neither does it have washout

Table 7.1: The qualitative comparison among back-propagation-based gradient search, random method and ZIZO. method and ZIZO. $++$: Highly satisfies. $+$: Satisfies. $-$: Much less satisfies.

	Gradient Search	Random Search	ZIZO
Complexity	++	-	+
Time Consuming	++	-	+-
Accuracy	+	-	+-

operations [XLL17]. This makes MBRC-RNN significantly different to ESN. However, the principle of a readout layer is used, and the output layer trained by linear combination with the gradient descent. Randomly generating W^{in} and W works well for ESN. In this work, we could also randomly initialize W^{in}, W and train W^{out} by Equation 7.4, though it takes time to adjust the hyper-parameters for ESNs and MBRC-RNN using manually tuning or grid searching methods [Luk12]. These require users to be equipped with professional experience of tuning hyper-parameters for neural networks.

ZIZO is a type of heuristic random searching-based method [XLX$^+$17]. The principle of it is, firstly, generating a batch of possible weights for neural networks in predefined ranges. As there is not sufficient information to predict the probability distribution of neural weights, a batch of possible parameters is randomly generated according to a uniform distribution. Second, verifying one of the possible candidates. If E reaches the predefined threshold, the program is terminated. Otherwise, new ranges are regenerated based on the sub-optimal results generated in the last steps, i.e., regarding these points as centers of possible solutions and regenerating a new box, accordingly. The final step is repeating the previous two steps until the program hits predefined conditions. In order to avoid the issue of overfitting for neural networks, one defines one of the stop conditions as $E_{generalization} \geq 3 * E_{training}$ when ZIZO reaches a group of weights. Here, $E_{generalization}$ and $E_{training}$ are the prediction and training errors, respectively.

A qualitative comparison of the gradient search, random method, and ZIZO are presented in Table 7.1. The gradient descent searching-based back-propagation training algorithm theoretically works for all neural networks, although it must conduct derivative operations on neuron states in different hidden layers. In order to update each neuron's weights, back-propagation is applied to the entire neural network in the form of the Jacobian matrix. With a larger neural network, the calculation complexity will increase exponentially and is usually time expensive. A random searching algorithm does not have logic inside, and the only effort is in keeping randomly generated samples in the predefined probability density distribution. ZIZO is a heuristic way of searching for the appropriate weights of MBRC-RNN. It does not have any derivative operations. The core reason for using ZIZO to update neural weights is the evolving systems based on the last step in generated searching results, i.e., if current training performance is getting better, the weights from this trial are applied to re-generate a

new solution space. Otherwise, the weights generated in the last step are used to reproduce another group of possible weights. In the following experiments, one can see that the hidden layer is much smaller than the input layer. In order to tackle the issue of overshooting in neuron states (the outputs of neurons being always equal to 1 or 0), it is necessary to limit the weight searching space to a small range. One disadvantage of ZIZO is that it cannot guarantee neural networks will ultimately achieve global optimal states, though the fast searching speed saves time for users, allowing more trials to search for optimal solutions to be conducted.

7.3 Experimental Results

In this section, the performance of the proposed model is evaluated by employing it with a milling dataset collected from the NASA data repository [AA07]. Firstly, the dataset and experiment setup is introduced. The performance of MBRC-RNN for modeling the complex milling sensory data is then illustrated. Finally, there is a comparison of the experimental results with those of existing works, and a consideration of the differences.

7.3.1 Milling Tool Wear Data Set

The dataset concerns experiments using different runs on a milling machine under various operating conditions. The milling tool wear is measured using a regular cut, as well as entry and exit cuts. The sampled data were gathered by three different types of sensors: an acoustic emission sensor, a vibration sensor, and a current sensor. The signals from the sensors are amplified and filtered, then fed through two RMS operations, before entering the industrial computer for data acquisition. Signals from a spindle motor current sensor are fed into the computer without further processing. Data on other factors that may affect tool wear are also collected, including the duration of the experiment and the depth of cut and material. There are 16 cases with a varying number of runs. The number of runs depends on the degree of flank wear, which is measured among runs at irregular intervals up to a wear limit. The basic setup encompasses the spindle and the table of the Matsuura machining center MC-510V. The matrix of parameters chosen for the experiments is guided by industrial applicability and recommended manufacturers settings. For details of the experiment setup, one can reference the work [GNGGVR16].

7.3.2 Data Preprocessing

Figure 7.2 depicts the contour of the six types of normalized sensory data. The X-axis indicates the different runs in one case study and the Y-axis the different instances in different case studies. Data pre-analysis shows that spindle motor current signals and spindle vibration signals are very noisy and

corrupted. Properly merging this massive amount of data into a single neural network and providing an end-to-end service is crucial and challenging. In this work, MBRC-RNN is specially designed for this purpose. In order to find appropriate weights for MBRC-RNN, the map sensory data is first mapped into [0, 1] which is collected from the same sensors, except for flank wear VB. Flank wear is used as the indicator of health status of cutters. Directly inputting 0 and one into a neural network could cause fluctuation in the final outputs. Hence, all inputs with 0 are reset to 0.01 and replace 1 with 0.99. Additionally, instances marked with NaN were simply deleted. After normalization and noise clean operations, there was a total of 145 instances selected for conducting experiments. The method used to map the data into the range [0, 1] is presented in Equation 7.9.

$$Y = \frac{(Y_{max} - Y_{min}) \cdot (X - X_{min})}{(X_{max} - X_{min})} + Y_{min} \tag{7.9}$$

where Y_{max} and Y_{min} user-defined up a boundary and lower boundary, respectively. X_{max} and X_{min} are the maximum and minimum values of the collected signals. To demonstrate the method, six

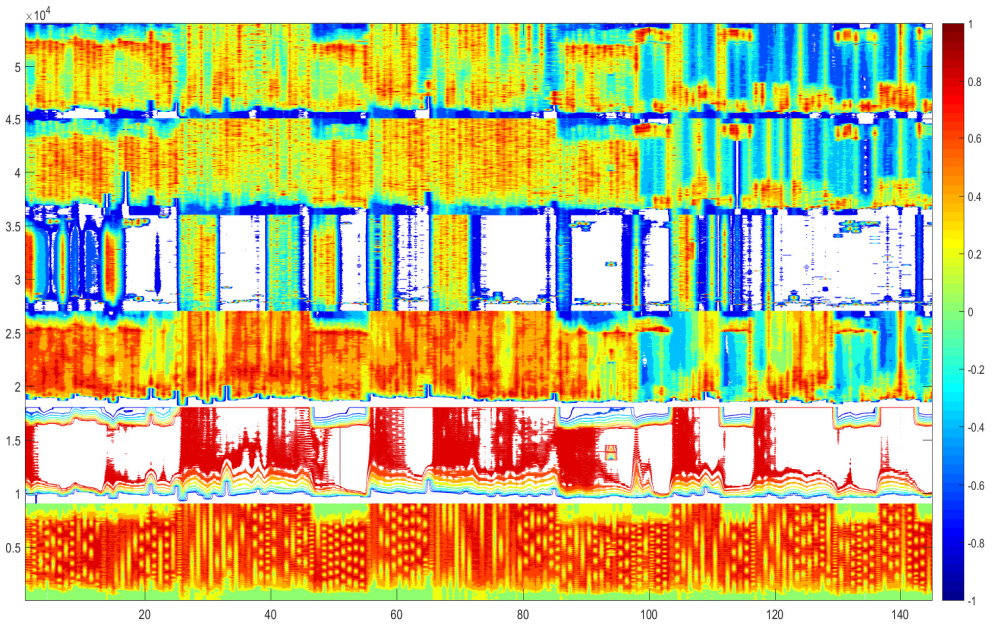


Figure 7.2: Contour of normalized experiment dataset. The horizontal axis is the index of various runs, vertical axis constituted by spindle motor current, spindle motor current, table vibration, spindle vibration, acoustic emission from table and spindle.

different case studies were selected (Figure 7.3). These six case studies reflect six types of milling tool wear trend. The experiment goal is applying MBRC-RNN to these sensory data and modeling the cutter tool wear status.

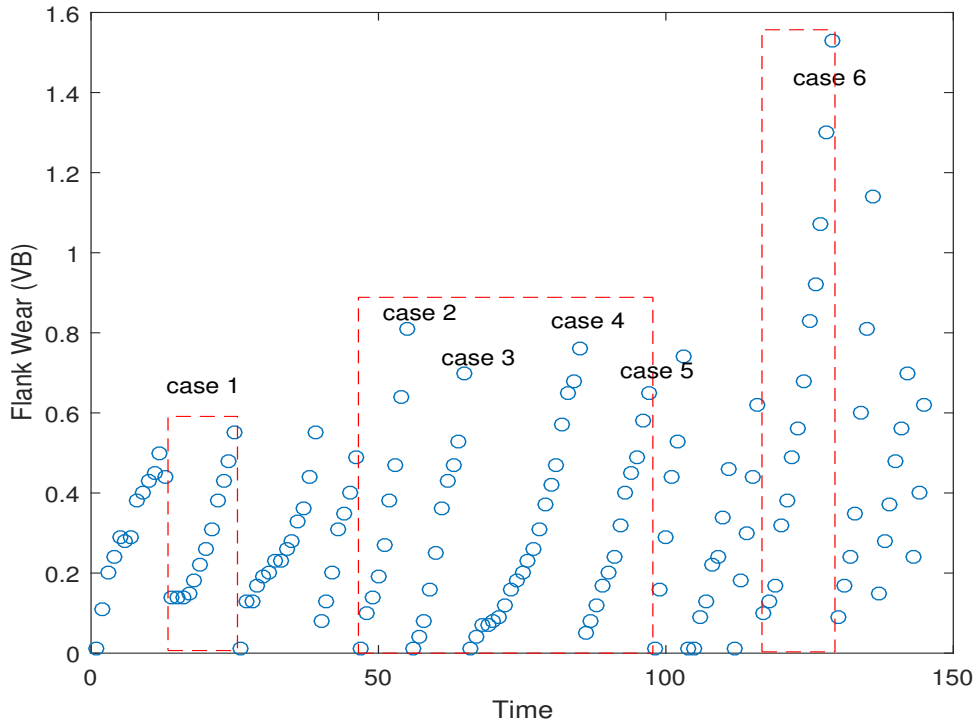


Figure 7.3: Six different case studies. Each case represents one independent milling process under various operation conditions.

7.3.3 Case Study

The MBRC-RNN model is implemented in MATLAB, as this is widely used in modeling industrial manufacturing systems. The entire experiment process is performed on a Microsoft Azure NCSeries virtual machine powered by one NVIDIA Tesla K80 GPU. The method's functionality is demonstrated using three techniques. The MBRC-RNN model is trained with part of the data, and the obtained results are highly accurate. The second is the modeling of the complex milling process with a limited number of neurons in the hidden layer. Third, MBRC-RNN could be a general milling process modeling method under various operating conditions. The experiment results demonstrate that with fewer neurons in the RL layer, MBRC-RNN could achieve high prediction accuracy. The searching and training times are varied, but each experiment is less than 20 minutes.

Results We discretized the deterioration trend of tool wear in experiments and treated each instance as an independent run. With 80% of experiment data randomly selected for training, the remaining part is applied for testing. Figures 7.4-7.9 depict the correlation efficient between prediction and

Table 7.2: Experiments with different MBRC-RNN structures((input neurons)-(hidden neurons)-(output neuron)). Training and prediction errors are calculated by mean-square-error as Equation 7.7

Case No.	Structure	Training Errors	Prediction Errors
Case 1	$54001-11-1$	$2.09e-4$	$3.38e-4$
Case 2	$54001-12-1$	$0.439e-4$	$6.38e-4$
Case 3	$54001-9-1$	$1.46e-4$	$7.78e-4$
Case 4	$54001-20-1$	$3.54e-4$	$4.9e-3$
Case 5	$54001-11-1$	$9.14e-4$	$1.61e-4$
Case 6	$54001-17-1$	$2.1e-4$	$6.56e-4$

desired outputs among six different cases. The correlation calculated here is all correlation. In effect, after the training dataset trains the model, the whole dataset (includes training dataset and testing dataset) is used to test the trained model and calculate the correlation (Equation 7.10). The corresponding experiment results are presented in Table 7.2.

$$R = \frac{\sum_m \sum_n (A_{mn} - \overline{A})(B_{mn} - \overline{B})}{\sqrt{(\sum_m \sum_n (A_{mn} - \overline{A})^2)(\sum_m \sum_n (B_{mn} - \overline{B})^2)}} \tag{7.10}$$

The results demonstrate that MBRC-RNN is capable of providing end-to-end complex sensory data modeling and prediction service. It automatically merges six different types of sensory data and precisely fits tool wear states. It is worth noting that, in this work, the experiment dataset is highly corrupted, but the results are still accurate, which demonstrates the robustness of MBRC-RNN.

Table 7.3 illustrates a qualitative comparison of various modeling methods for milling sensory data. Assessments of the performance of each method are carefully performed based on the provided results and detailed specifications of methods in their corresponding reports. Under such evaluations, the results suggest the superiority of MBRC-RNN over existing methods as it can easily and automatically model noisy milling process-generated data.

Predicting Tool Wear with a Small-Sized Hidden Layer If one used a deep neural network to tackle this work, it would need several layers with hundreds of neurons in total for the hidden layers, as the data is corrupted and retained in such high dimensions; in effect, the neural network will be designed with 54,001 inputs. In the experiments, the number of hidden neurons is 11, 12, 9, 20, 11, and 17 (Table 7.2). The reason for such configuration is that neurons share their states with others through complete recurrent connections. With the help of ZIZO, it is much easier, compared with classic neural networks-based methods, to conduct activities like checking modeling accuracy and then adjust the number of neurons accordingly to avoid information overlapping. It gathers information from these two steps without considering any other hyper-parameters, for example

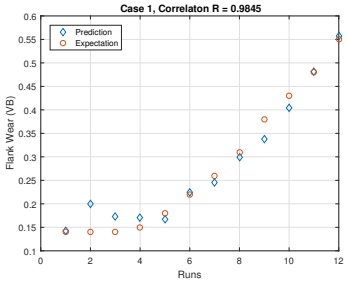


Figure 7.4: Case 1 experiment results. The correlation efficient between prediction and desired outputs is $R = 0.9845$.

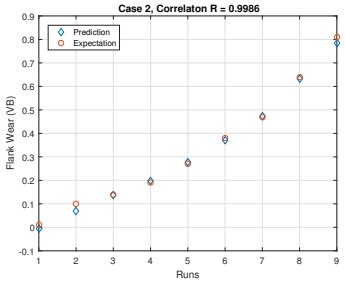


Figure 7.5: Case 2 experiment results. The correlation efficient between prediction and desired outputs is $R = 0.9986$.

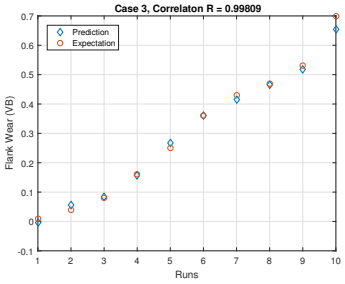


Figure 7.6: Case 3 experiment results. The correlation efficient between prediction and desired outputs is $R = 0.99809$.

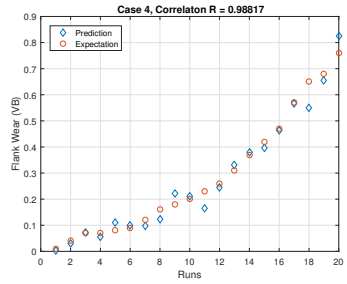


Figure 7.7: Case 4 experiment results. The correlation efficient between prediction and desired outputs is $R = 0.98817$.

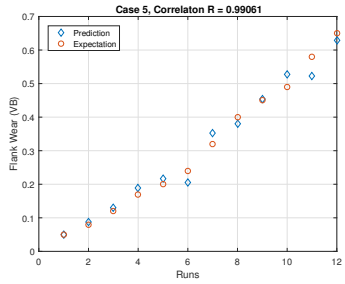


Figure 7.8: Case 5 experiment results. The correlation efficient between prediction and desired outputs is $R = 0.99061$.

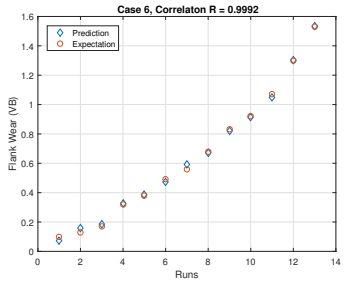


Figure 7.9: Case 6 experiment results. The correlation efficient between prediction and desired outputs is $R = 0.9992$.

learning rate, which is very user-friendly. Potentially, integrating the step of searching appropriate size for neural networks with ZIZO and the program will automatically select the best structure based on practical applications. This will be conducted in the next work.

Additionally, the hyper-parameter searching results also indicate that:

- the dataset itself is kind of simple, meaning with a small number of reservoir neurons, the ESN can fully represent the underlying pattern of the original data set. Hence, ESNs can also be played as the complexity detector of the provided training data set. It gives the insights into how the data distributed and the complexity degree of our data.
- the results indirectly implies that ZIZO can also automatically generate sparsity neural network or sparsity echo-state networks. For example, in our case study, with around 5000 input, only

20 around hidden neurons applied. If we compare this with the so-called neural network drop-off techniques, ZIZO tries to minimize the size of the reservoir and achieve the same effect as the drop-off operations.

Table 7.3: Quality comparison among BMRC-RNN and existing methods. $++$: Highly satisfies. $+$: Satisfies. $-$: Much less satisfies.

Method	Accuracy	Complexity	Generalization
MBRC-RNN	+	−	++
PSO-SVM [GNGGVR16]	+	++	+
Stochastic [WG16b]	+-	++	+-
SA-ESN [WG]	+	++	+-
S-Transform [RHZC13]	+-	+++	+-

Explainable MBRC-RNN Since the proposed MBRC-RNN neural network is similar to the classic echo-state networks, so we applied the same trick to exam the internal states of the neural network. Figure 7.10 shows one of the reservoir weights distribution of the ESN introduced previously. In

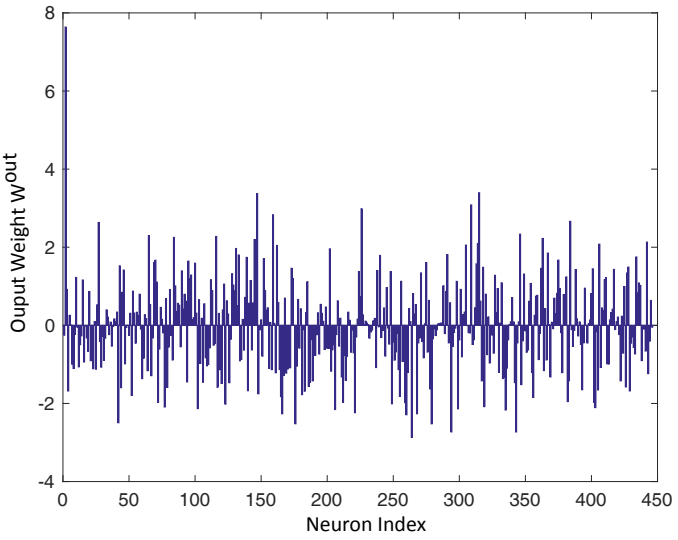


Figure 7.10: Reservoir states.

Chapter 3, we have introduced the properties of ESNs. We know that the essential properties of ESNs is the echo-sate. The formal behavior of such property is that neurons smoothly generate the independent states curves as we have been presented. The number of corresponding positive and negative weights for generating such behaviors is quite close for our case study. Since we use the

same data set test our proposed model, hence, in this work, we use the same strategy to exam the property of MBRC-RNN as following introduced.

In order to identify how the MBRC-RNN works, in particular how each neuron contributes to the final prediction, the input weights W^{in} and RL weights W are visualized for six different case studies (Figure 7.11 - Figure 7.16). Different color blocks represent the connection weights for every two neurons. It is worth noting that, in this work, neurons from the hidden layer have self-connections. It is clear that ZIZO searching obtained neural weights follow a well-organized statistical distribution, i.e., they are gradually distributed without a big jump.

From the presented figures, we can conclude that the MBRC-RNN is equipped with symmetric number of positive and negative weights, which ensure the neural network produce appropriate results. With such parameters, we are able to understand the neural network much better and see how each part of the neural network interact with each other and output the desired values, so that later we can manually fine-tuning the hyper-parameter again and improve the performance of the designed neural networks.

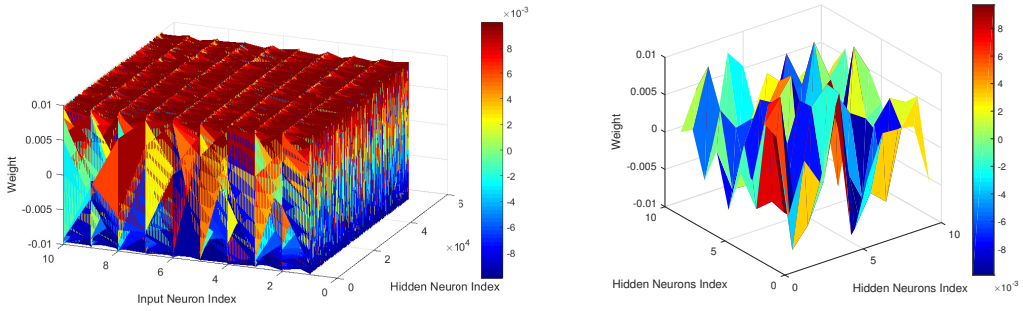


Figure 7.11: Case 1. Weights of input-hidden layers & weights of hidden-hidden recurrent layer (RL).

7.4 Summary

A novel neural network, MBRC-RNN is proposed for modeling the milling process-generated sensory data and predicting the tool wear degree. The new structure adds multi-bias to the recurrent hidden layer, specifically taking the non-sensory features (static features) as additional biases. With the help of the ZIZO sampling method, a group of appropriate weights is quickly obtained for input layer W^{in} and RL layer W. The readout layer uses a linear combination technique to train the W^{out}. MBRC-RNN totally has three different layers: input layer, RL layer, and readout layer. The RL layer

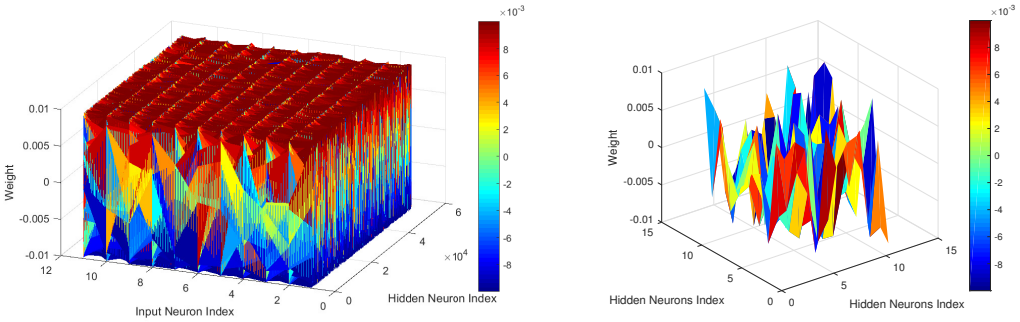


Figure 7.12: Case 2.Weights of input-hidden layers & weights of hidden-hidden RL.

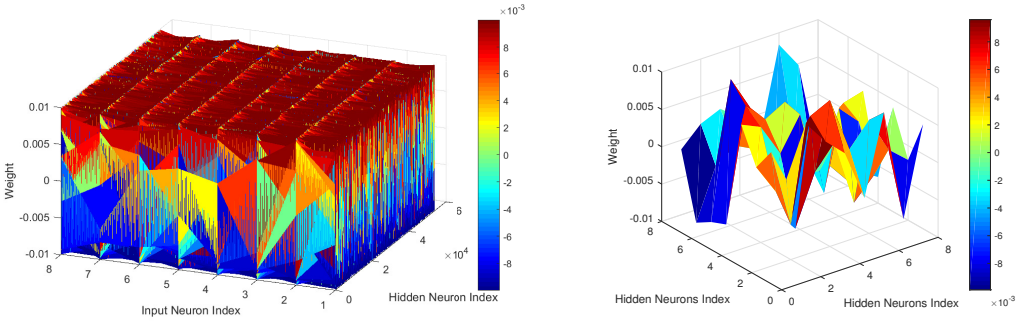


Figure 7.13: Case 3. Weights of input-hidden layers & weights of hidden-hidden RL.

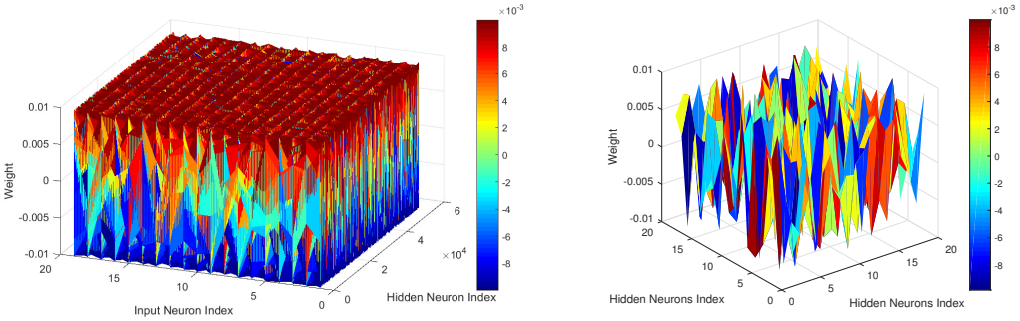


Figure 7.14: Case 4. Weights of input-hidden layers & weights of hidden-hidden RL.

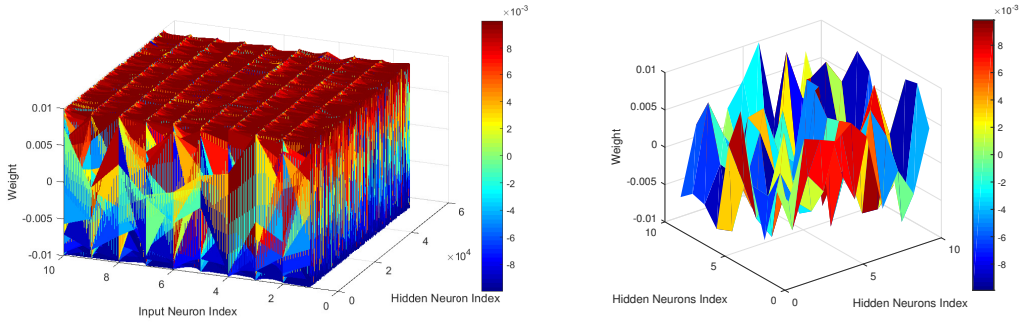


Figure 7.15: Case 5. Weights of input-hidden layers & weights of hidden-hidden RL.

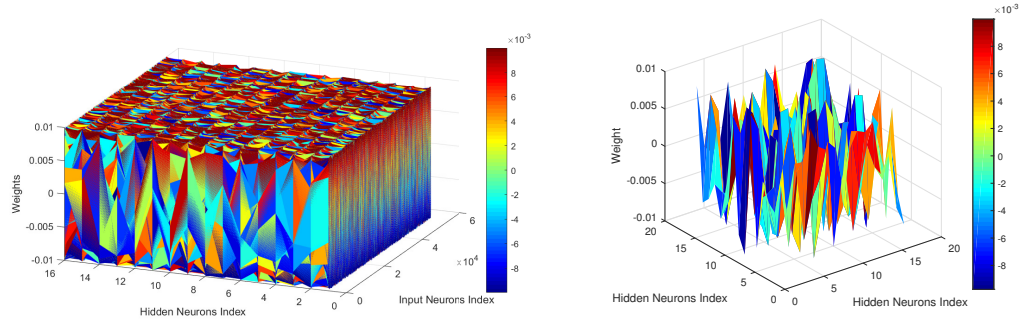


Figure 7.16: Case 6. Weights of input-hidden layers & weights of hidden-hidden RL.

is a bio-inspired way of either transforming the inputs to a high dimension space or reducing the redundancy information of inputs according to the different application scenarios. It is demonstrated that MBRC-RNN can successfully model the tool-wear status of the milling process in various operation conditions (the dataset is collected with different operation parameters). In this work, it is shown that with much fewer neurons in the hidden RL, one can predict precisely the tool wear status of the different scenarios and obtain very stable and robust results.

CHAPTER 8

End-to-End Industry Data Processing with GNNQP: Quality Prediction

In this chapter, the work aims to predict quality violations at time $t+1$ based on historical sample data up to time t. Quality control engineers randomly select N samples out of M WIP products, in which $N \ll M$, and measure respective quality characteristics. Since manual inspection of samples is time and resource consuming, the amount of samples that can be analyzed is low. Low samples sizes mean that one can not extract useful information by directly inputting the data into a feature-extraction ANN. With a limited amount of samples, any probability assumption about the product quality is not appropriate. Besides, out of the limited sample set, even fewer samples violate quality limits, creating an imbalanced dataset. Therefore, the research gap addressed here is how to predict the continuous change in product quality with a small, imbalanced dataset, without any underlying assumption of a probability distribution.

8.1 Work-in-Progress Product Quality Prediction

In order to propose an automated method of predicting the quality of products generated at the next step, a so-called neural computing method is developed: the Generative Neural Network for Quality Prediction (GNNQP). The proposed idea of applying neural computing methods to quality control is constituted by three parts. (1) instead of adopting handcrafted features such as statistical features, an artificial neural network (autoencoder) is used to extract features and apply outputs from neural networks as the features, called neural representation, and later used for modeling the status of products. (2) The core part of the neural computing method is the learning methods developed by the

machine learning community. Learning enables the modeling of complex manufacturing systems in easy and efficient ways. The only effort required is the collection of as much data as possible. The final part is modeling complex systems with deep neural networks. At the initial stage of the theory of artificial neural networks, it is shown that even with three-layer FFNN, one can theoretically approximate any non-linear functions. The problem here is if the size of each layer of the neural network is too big, it becomes impossible to train it, and the approach is not practical for real-world problems. With the recent achievements of deep learning, instead of giant three-layer neural networks, one can design a neural network with many layers, each layer equipped with small-sized neurons. It is done so by:

- Showing that an artificial neural network structured in an auto-encoding topology can provide a rich latent space representation of powder metallurgy process data automatically. By auto-extracting features, the algorithm predicts the quality of the products better than the existing approaches, even with a small data sample. The autoencoder network enables us to reduce the dimensionality of the data in a latent space representation. SMOTE method is used to mitigate data imbalance.
- Illustrating that the method can capture the underlying probability distribution of the experimental data set. Besides providing a reasonable degree of accuracy in predicting next step quality character, with an average prediction accuracy close to 80%, the probability distribution of the predicted data covers a wider range than the original data. With a wider-ranged probability distribution, the method can overcome the noisy parts of the original data set randomly generated at products sampling phase.

As a result, the method can effectively capture features of inferior quality products and is at least 10% better at predicting unqualified sample products by analyzing historical data.

The chapter is organized as follows: Section 8.2 introduces the data preprocessing steps, and explains the autoencoder method developed. Section 8.3 test the method on a case example from the powder metallurgy industry and compares performance with the existing approaches. Section 8.4 concludes the work.

8.2 The Generative Neural Network Method

The main objective of the model is to predict the WIP product quality at time t, based on the measurement of the product status at time $t-1$, ..., 1. The work only considers one specific quality characteristic since in the industrial domain of application if any one of the attributes of the product is

out of the control-chart boundaries, the product is considered disqualified. The quality characteristic used refers to a certain dimension such as the diameter or length of a WIP product.

8.2.1 Pre-processing methodologies

There are three pre-processing stages before data was fed into a learning system including (1) interweaved input data extension. (2) data-class balancing. (3) data normalization.

Interweaved input data extension

At each stage of the product monitoring process, a small sub-sample of the products are randomly selected to undergo quality assessment. The ratio of sample to population size in the problem domain is 5:2000. With such a limited number of products the neural network cannot decode a reasonable global behavior of the products since it struggles with over-fitting issues. To tackle this problem, the work interweaved multiplicative input samples, T x 6 matrix, and enriched the data set with more hidden information, as follows. Suppose matrix X is the original data without additional labels:

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} & x_{1,5} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} & x_{2,5} \\ \dots & & & & \\ x_{t,1} & x_{t,2} & x_{t,3} & x_{t,4} & x_{t,5} \end{bmatrix} \tag{8.1}$$

where t is the time at which the work samples within the specific product quality monitoring period, then the left-hand-side pairs shown in the below vector:

$$[x_{1,7}, ..., x_{1,20}] = \begin{bmatrix} x_{1,1} \cdot x_{1,1} & x_{1,1} \cdot x_{1,2} & \dots & x_{1,1} \cdot x_{1,5} \end{bmatrix} \tag{8.2}$$

stands for new augmented data to be incorporated to train the neural network. Equation 8.2, illustrates how to extend the first row of the status matrix X. The operations follow for all rows in a similar fashion. In this way this work extends the product status matrix to a T x 21 with the last column corresponding to the labels of the sample.

Data-class balancing

When sampling products to perform quality control, it is usually the case that the number of positive and negative samples is not uniformly distributed in the dataset. Here, a negative sample indicates a product within control limits, and the positive sample indicates a product that violates control limits. In the four study cases, the imbalance ratio of negative/positive samples is $66.82, 35.46, 5.83$ and 32.19, respectively. This imbalance weakens the generalizability of a data-driven approach. To tackle this, SMOTE [CBHK02] is adopted, to re-balance the collected dataset. First, every row of the matrix

of products status is plotted as a point in a 5-dimension space. Then, SMOTE calculates the distance between every point and its neighbors. Based on the imbalance degree of the dataset, one can decide how many points SMOTE should generate artificially to re-balance the data. For instance, if there are three products within control limits and one product that violates it, SMOTE augments the dataset to find two more positive samples around the region to re-balance the dataset.

Note that such samples are utilized during the training to increase the generalization of the method but have no contribution to the testing environment.

Normalization

The final step for data preprocessing is resetting outliers and normalizing all feature values into the range [0, 1] by a normalization postulated in Equation 8.3).

$$Y = \frac{(Y_{max} - Y_{min}) \cdot (X - X_{min})}{(X_{max} - X_{min}) + Y_{min}} \tag{8.3}$$

here, Y_{max} and Y_{min} are user defined upper bound (defined as 1) and lower bound (defined as 0), respectively. X_{max} and X_{min} are the maximum and minimum values of collected data. Regarding the aforementioned operation to reset outliers, the principle is to reset then slightly lower or greater than X_{min} or X_{max}, respectively.

The pre-processed data are then used to train neural network models, which are described as follows.

8.2.2 Stacked unsupervised and supervised learning systems

Initially, an autoencoding neural network is designed for automatically extracting meaningful information from the augmented X. The autoencoder maps the augmented X to a latent space representation. Then, a simple multi-layer perceptron (MLP) is applied to the latent coded data to predict the quality of a product in a next step transition. To decrypt the coded prediction, they are fed into a decoding layer to obtain a rescaled output, which falls into the range of [0, 1]. It then makes use of Equation 8.3 and the predefined control limits to get the predicted values back to real feature (i.e., quality characteristic) values. The actual samples from the augmented X are picked up at the last decoding layer, as the final predicted values of quality characteristic (Figure 8.1). In the following, it describes the structure of each neural network component.

The autoencoder [BC+08] provides an automatic way to learn a low dimensional latent representation for input features by reproducing inputs to at their output. Accordingly, one can define $x \in \mathbb{R}^{D_x}$, which is a D_x dimension input to the autoencoder; The encoder initially maps x to a lower dimension vector $z \in \mathbb{R}^D$, and generates an estimation of x, $\hat{x} \in \mathbb{R}^D$:

$$z = f(Wx + b_1); \;\; \hat{x} = f(W^T z + b_2), \tag{8.4}$$

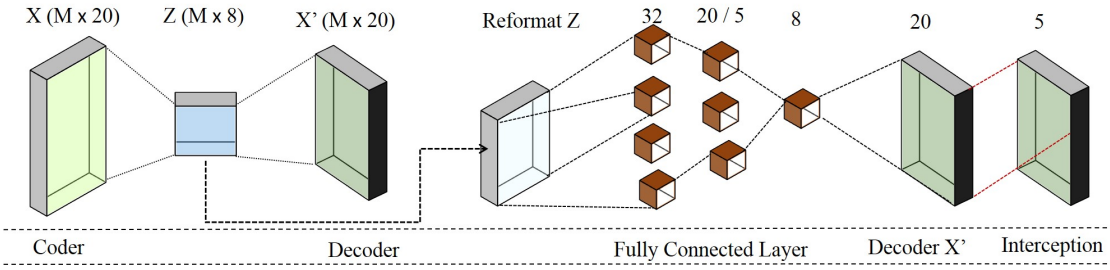


Figure 8.1: The abstract structure of the proposed generative neural network method for product quality prediction. The collected WIP product status (X) is fed into a naive autoencoder. The trained autoencoder then learns how to code the inputs. With new inputs fed into autoencoder, the coded information need to be prepared to be input to the fully connected layer for the quality prediction. Based on trial and error, it is found that WIP product status is strongly connected to the past four states, which is the reason for Z reformation after coding operation. After fully connected layers are adopted, the coded prediction is output. According to the features S1, S2, S3, and S4, two different neural network structures of the fully connected layer are designed. The trained Decoder X' is then applied again for decoding the prediction results. The "Interception" copies the outputs from Decoder X' and selects the top 5 entries as the values of predicted values of the feature (i.e., the quality characteristic).

where f is a sigmoid function, $W \in \mathbb{R}^{D_x \times D}$ is the weight matrix, and vectors $b_1 \in \mathbb{R}^{\mathbb{D}}$, $b_2 \in \mathbb{R}^{\mathbb{D}}$ represent the bias values.

In order to train the autoencoder, the scaled conjugate gradient (SCG) learning algorithm [Møl93] is adopted. The cost function for the network is represented as C_1 (Equation 8.5) and is optimized by SCG subsequently:

$$C_1 = \underbrace{\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{I}(x_{in} - \hat{x}_{in})^2}_{\text{mean squared error}} + \underbrace{\lambda R_{L2}}_{L_2 \text{ regularization}} \tag{8.5}$$

The first item in C_1 represents the mean squared error (MSE). The second item is a L_2 regularization that is used for tackling overfitting issue, where λ is the regularization coefficient. L_2 regularization is computed as follows:

$$R_{L_2} = \frac{1}{2}\sum_{j}^{t}\sum_{i}^{k}(w_{ji})^2 \tag{8.6}$$

Where, n and k are the number of samples and the number of features in the training data, respectively.

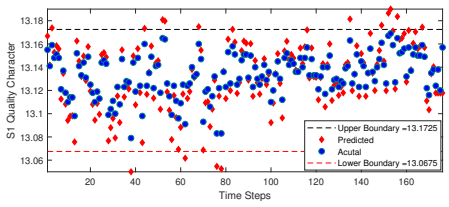
The pre-trained autoencoder then provides rich features for given input data. The latent space is used to train the MLP. The cost function for the MLP is chosen to be the same as C_1, with a slight difference in the number of neurons and network layers. This objective function is named as C_2. The autoencoder is stacked together with the MLP as a complete model. A fine-tuning training stage is

then deployed on the overall network topology where the objective function is determined as the summation of the two as $C_{stacked} = C_1 + C_2$.

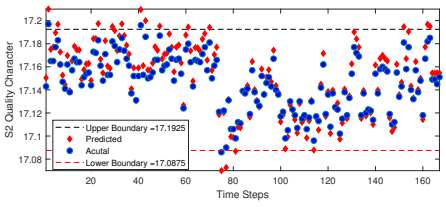
8.3 Experimental setting and results

This section evaluates the performance of the proposed method for quality prediction on a dataset collected from a powder metallurgy process. It initially describes the specification of the data before applying the method and assessing its performance. Finally, it compares the results with existing methods applied for solving similar problems.

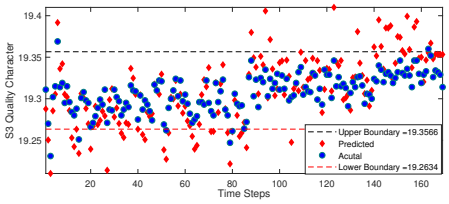
The process under query involves four steps, including powder blending (mixing), compaction, sintering and post-production processes [ASM+14]. The dataset is from the compaction stage of a given product, where five samples are randomly selected out of a batch of 2000 products at each time step. A given product quality characteristic is measured for these five samples.



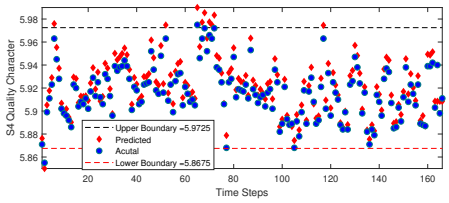
(a) S1.



(b) S2.



(c) S3.



(d) S4.

Figure 8.2: Episodes of the algorithm's predictions in action. Red diamond is the predicted value, and blue circles are the actual value of the attribute of a product. The samples outside of the bounds are considered bad samples.

The specification of the experimental dataset is summarized in Table 8.1. In the experiments, four different groups of the dataset are selected for evaluating the performance of the method, i.e., experiment S1, S2, S3, and S4. The corresponding sub data sets come from four different production orders, i.e., O1, O2, 03 and O4, respectively.

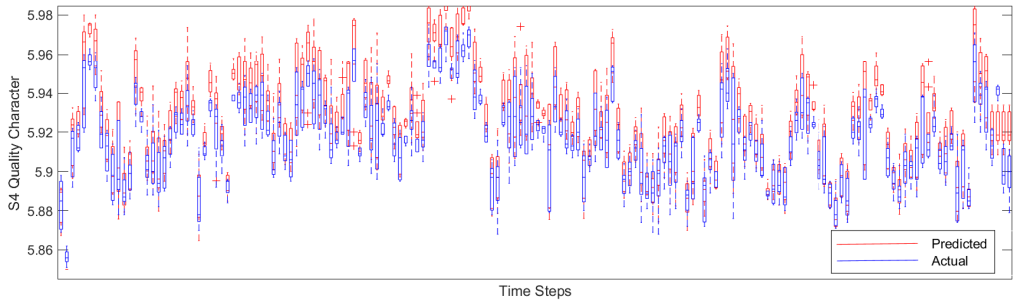


Figure 8.3: Box plot of the sample products' quality together with their prediction from the method for S4. A reasonable degree of accuracy over time is observed.

8.3.1 Experimental Results

The implemented method is written in MATLAB by using the Neural Network Toolbox. The training process is performed on a Microsoft Azure NC-Series virtual machine powered by one NVIDIA Tesla K80 GPU.

In order to train and test all four different orders, the designed neural network topologies as depicted in Table 8.2. The performed experiments with initially training the autoencoder to obtain a feature extractor. Before the training processes, the re-balanced dataset is split into 70% negative samples and 30% positive samples. 70% of the data in each class is used for training the autoencoder, 15% for the validation and 15% for the testing.

For the MLP, the latent space of the autoencoder is reshaped. The determined intermediate products' status at time t, which has a strong correlation with the four time-lagged states of $t-1$, $t-2$, $t-3$, $t-4$. Then, the re-shaped inputs of the MLP are organized in the form of Z, as:

$$Z = \begin{bmatrix} z_{1,1} & z_{1,2} & z_{1,3} & z_{1,4} & z_{1,5} \\ z_{1,2} & z_{1,3} & z_{1,4} & z_{1,5} & z_{1,6} \\ \dots & & & & \end{bmatrix} \tag{8.7}$$

Where, $z_{i,j}$ is the coded output from the autoencoder shown in Figure 8.1. For instance, the autoencoder structure for experiment S1 is 20-8-20, i.e. the size of $z_{i,j}$ is 8. After reformatting the coded Z, the size of inputs of fully connected layers is 4×8.

Time-lagged inputs, together with the MLP form a time-delayed neural network architecture which incorporates the temporal dependencies in the data. The recurrent neural network RNN structure is deployed to overcome the vanishing and exploding gradient problems in the standard RNNs [BSF94].

Table 8.1: Experiment data set of 4 different orders from FB1060 part. The data set was collected within 365 days.

Experiment	# of Samples	sample size	Oder #
S1	1899	1×5	O1
S2	1896	1×5	O2
S3	1911	1×5	O3
S4	1892	1×5	O4

Experimental Training Status of Generative Neural Network

The same neural network structures are applied on four sub data sets, which includes the structure for autoencoder and the structure for multi perceptron (perceptron layer). The experimental results demonstrate the generic of the method. With same structure of generative neural network, this work looks into the training process and show how the neural networks get trained (Figures 8.7a - 8.6f). The first rows of Figure 8, 9, 10 and 11 are the training status of autoencoder. The second rows of of Figure 8, 9, 10 and 11 are the training status of fully connected layer. In the following figures, R is the correlation coefficient between the prediction and the targets.

Results

The experimental results are shown in Figures 8.2a - 8.2d, for four experiments. For all experimental settings, the method reasonably predicted the quality of the products. It is observed that the algorithm predicts the samples that are outside or close to the specification boundaries, with higher degrees of sensitivity. The specification boundaries in all test beds are delicately narrow. This is because high-quality products are required, and therefore, a sensitive prediction is demanding. Experiment S3 has a higher number of miss-predictions due to a higher level of noise when compared with other experiments. A different neural network topology is utilized for this case (32-5-10) to overcome this issue. The experimental evaluations suggest that the method generalizes well even on samples that have never been seen during the training.

Table 8.2: Neural network structure for different orders. AE: autoencoder, FC: Fully Connected Layer. The activation function for both networks is a logistic sigmoid function.

Experiment	Structure of AE	Structure of FC
S1	20-8-20	32-20-8
S2	20-8-20	32-20-8
S3	20-8-20	32-5-20
S4	20-8-20	32-20-8

The experiments quantitatively measured the predictive classification performance of the method and illustrated it by the confusion plots shown in Figures 8.9a, 8.9b, 8.9c and 8.9d. In each sub-figure, the prediction accuracy for positive and negative classes was computed. The absolute accuracy for these tests was calculated as follows: 79.5% for S1, 74.9% for S2, 71.0% for S3 and 89.8% for S4. More importantly, it observes a good performance for the method of capturing most of WIP products which violate control limits. In the case study, with four different orders, the method predicted approximately 100% of samples that violated control limits.

Figure 7.3 represents a box plot for the experiment S4. It can empirically evaluate a reasonable degree of the prediction accuracy. To quantify the overall prediction performance for each testbed, it averages the candle for the actual and the predicted points and demonstrates them in Figure 3.

Figure 7.4 - 7.9 visualizes the performance of the method in all the experiments. The prediction boxes show that the predictions essentially capture the majority of the desired outputs with few outliers. These prediction outliers may come from the policy of resetting the outliers from original data set and the data generated by data balance technique SMOTE. Additionally, the predicted boxes are larger than the boxes of the desired outputs due to the randomness of the selected samples.

Comparison with existing methods

This section compares the results with the existing methods for quality prediction. As mentioned previously, ANN based approaches for quality control, especially for quality prediction, attracted a lot of research and application attentions [CTW$^+$08]. K-NN [LZL12] and SVM [CWC10]. Therefore, in the comparison, it selected the NN, K-NN and SVM based approaches as benchmark methods and compared their performance on the data set with the method. Table 8.3 represents a comparison

Table 8.3: Quantitative and qualitative comparison of the performance amongst existing approaches and the method on powder metallurgy data set of quality character of intermediate products. NN, Simple Feed-Forward Neural Network. k-NN, k-near neighbours algorithm. SVM, Support Vector Machine. PC Rate, positive class rate for capturing bad products. ++: Highly satisfies. +: Satisfies. -: The attribute is not covered -+: The attribute is fairly covered.

Algorithm	Accuracy	PC Rate	Generalization	Automated	unsupervised
GNNQP	78.8.0%	**98.0%**	++	+	++
NN based	89.0%	**55.6.0%**	+	+	+
k-NN based	54.0%	**89.3.0%**	-	-	-+
SVM based	65.0%	**31.0%**	-	-	-+

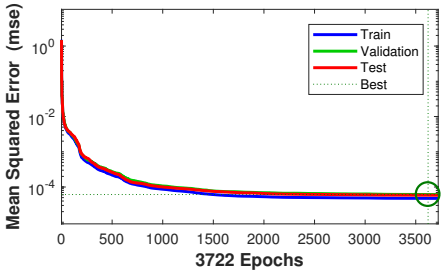
between the results with existing solutions. As discussed, it manually added labels for every five samples which sampled at the same time, i.e., 0 for the ones all fall into the control range, otherwise, labeled as "1". Therefore, the prediction accuracy of the method is calculated based on the correct classification of different samples. Under such evaluation, the results demonstrate the superiority of the classification performance of the method concerning identifying true positives. Besides, the method is entirely data-driven and automated.

Table 8.3, shows that neural network-based methods can realize high degrees of accuracy. Feature extraction can presumably be a big issue for industrial data analysis. Hence, the modeling effort for the K-NN and SVM-based methods is critically limited. The auto-encoded approach can predict higher number of true positives, which is very important in quality control as this measure would indicate that it is more likely that detected control limit violation might not be a true control limit violation (false alarms), but there is a lower probability that the method will miss an occurrence of quality issues when they are present (missed disturbances), which is crucial for customer satisfaction. Additionally, by equipping the method with an automated feature extraction toolkit, this work generalizes better by predicting correctly 98% of products violating control limits.

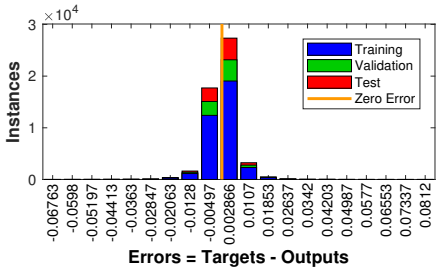
8.4 Conclusion

This chapter introduces a generative neural network method for predicting WIP product quality. As currently reported machine learning methods in this domain suffer from manual feature extraction, and noise, the method is based on an unsupervised learning setting and a time-delayed feed-forward neural network. By conducting the prediction of product status on coded inputs, the generative neural network method can tackle such issues with the limited amount of measurement data. According to the experimental results, reasonable predictive classification accuracy is achieved, and the method precisely captures the majority of the WIP products that violate control limits during the manufacturing process. Furthermore, autoencoder as an unsupervised feature extractor introduces an automated way for generating features from original data set without adding prior knowledge or presumption to the data set. The performance of the methodology over four different products using the same quality characteristic is the conclusion.

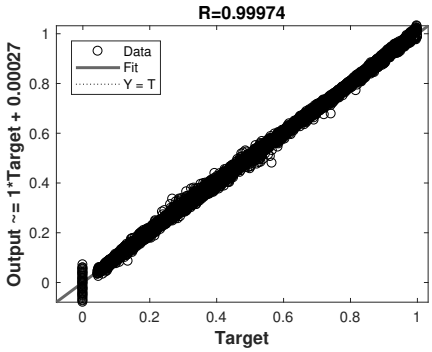
As part of future work, it is planned to improve the method to process multiple product quality characteristics simultaneously, instead of only handling one specific characteristic at one time. Moreover, more effort will be put to adopt auto-machine learning techniques to automatically choose a neural network structure that fits the best to a given dataset.



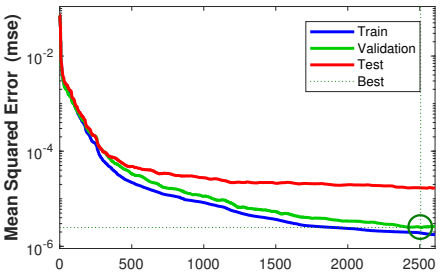
(a) Training performance of Autoencoder for experiment S2.



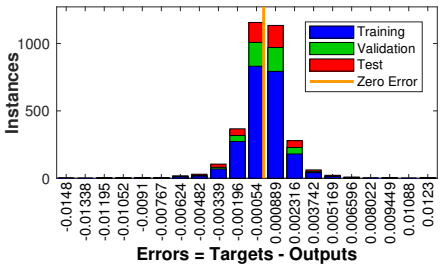
(b) Training-error distribution of Auto-Encoder for experiment S1.



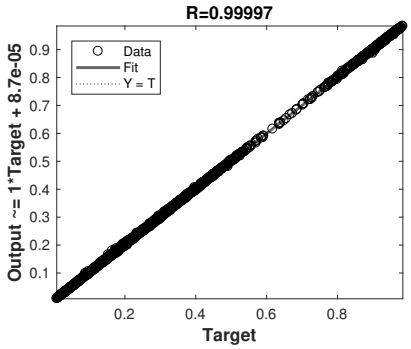
(c) Autoen-Encoder: correlation coefficient between Prediction and Targets for experiment S2.



(d) Training performance of fully connected layer for experiment S1.

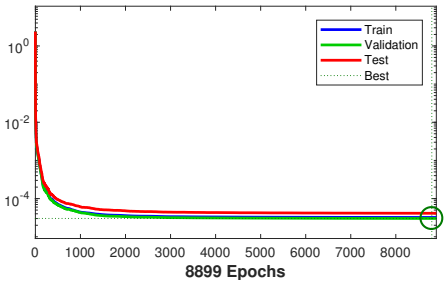


(e) Training-error distribution of fully connected layer for experiment S2.

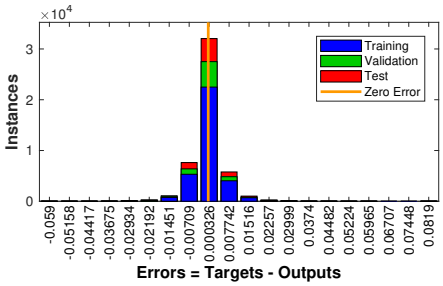


(f) Fully Connected Layer: correlation coefficient between Prediction and Targets for experiment S2.

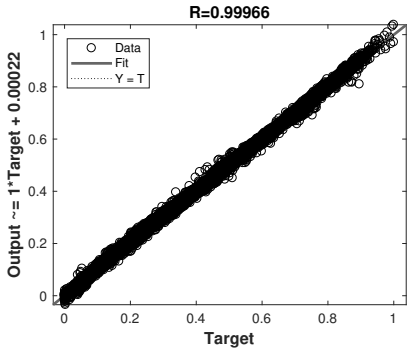
Figure 8.4: Internal training status of GNNQP for S2. The training processes are stable after 4000-time iteration. It worth to note that the training performance of the perceptron layer has stair-style training error reduction which is related to the initial values of the neural network. The maximum training time also affects this issue. Theoretically, with more training iteration, the curve of training performance of perceptron layer should be similar to the autoencoder training status, which is reduced smoothly.



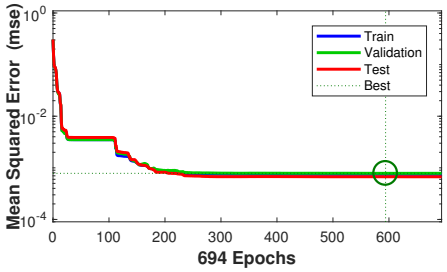
(a) Training performance of Auto-Encoder for experiment S3.



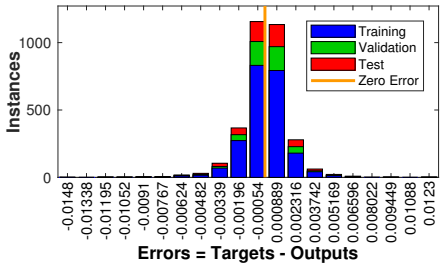
(b) Training-error distribution of Auto-Encoder for experiment S3.



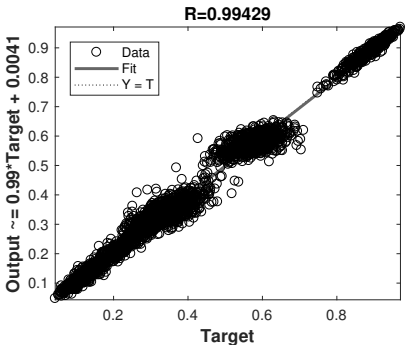
(c) Autoen-Encoder: correlation coefficient between Prediction and Targets for experiment S3.



(d) Training performance of fully connected layer for experiment S3.

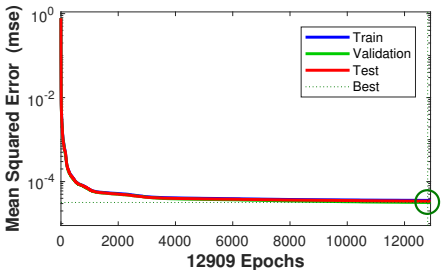


(e) Training-error distribution of fully connected layer for experiment S2.

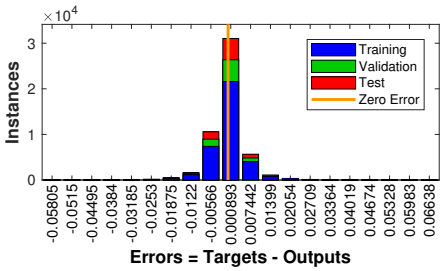


(f) Fully Connected Layer: correlation coefficient between Prediction and Targets for experiment S3.

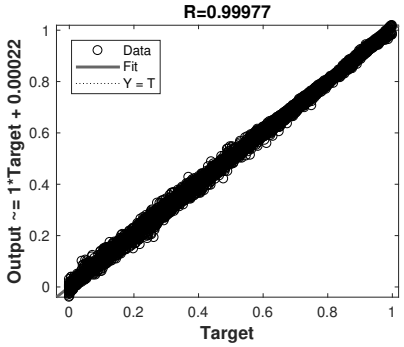
Figure 8.5: Internal training status of GNNQP for S3. This sub data set is fully imbalanced. From the prediction results of coded outputs of perceptron layer, there are gaps between the prediction and targets. This is also reflected on the error histogram, where has amount of non-zero errors from perceptron layer.



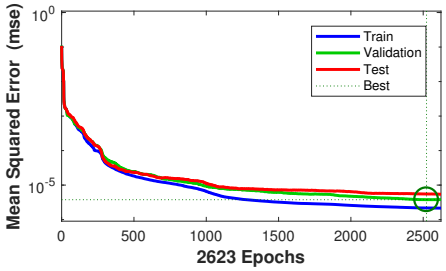
(a) Training performance of Auto-Encoder for experiment S4.



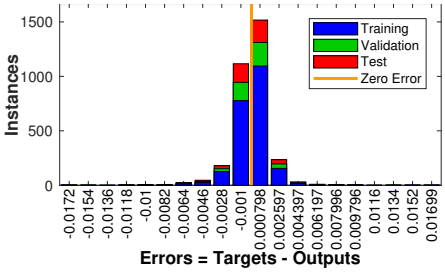
(b) Training-error distribution of Auto-Encoder for experiment S3.



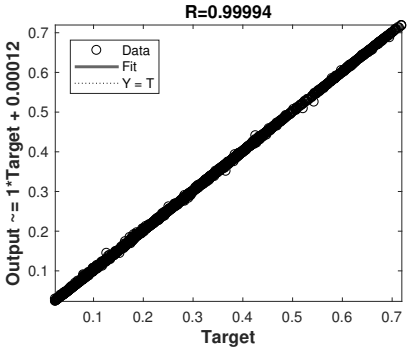
(c) Autoen-Encoder: correlation coefficient between Prediction and Targets for experiment S4.



(d) Training performance of fully connected layer for experiment S4.

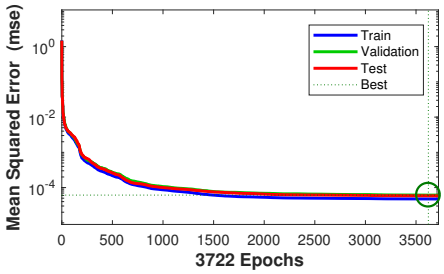


(e) Training-error distribution of fully connected layer for experiment S4.

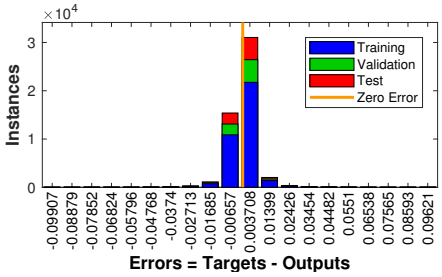


(f) Fully Connected Layer: correlation coefficient between Prediction and Targets for experiment S4.

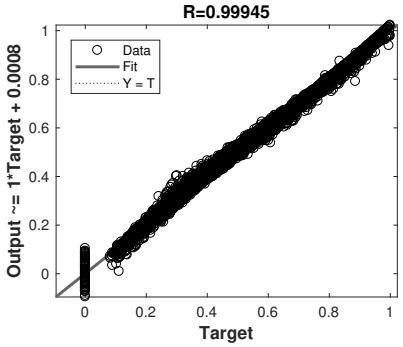
Figure 8.6: Internal training status of GNNQP for S4. The training status shows the superiority of the method. GNNQP gradually reaches the stable points with 12000 and 2500 iteration for autoencoder and perceptron layer, respectively. Most of the errors close to 0.0003 and 0.0012. Regression analysis shows the high modeling performance of the method.



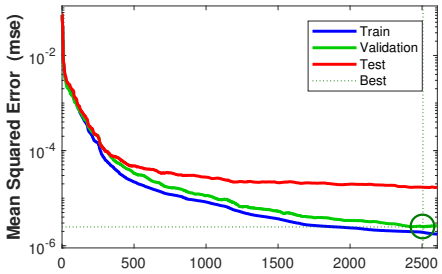
(a) Training performance of Auto-Encoder for experiment S1.



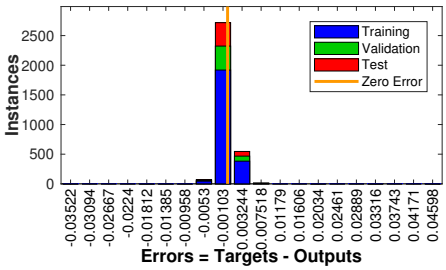
(b) Training-error distribution of Auto-Encoder for experiment S1.



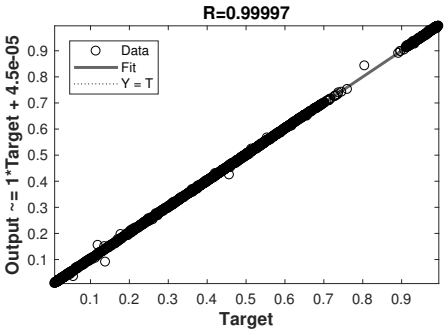
(c) Autoen-Encoder: correlation coefficient between Prediction and Targets for experiment S1.



(d) Training performance of fully connected layer for experiment S1.

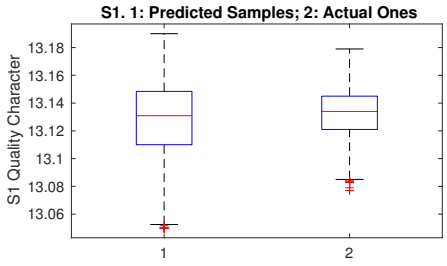


(e) Training-error distribution of fully connected layer for experiment S1.

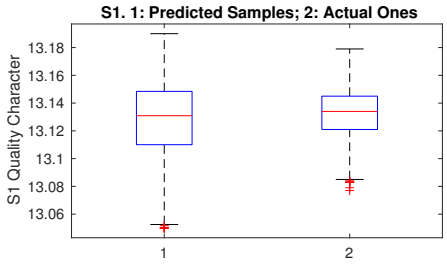


(f) Fully Connected Layer: correlation coefficient between Prediction and Targets for experiment S1.

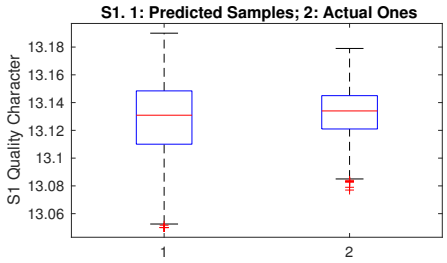
Figure 8.7: Internal training status of GNNQP for S1. After 3500 and 2500 steps iterations, autoencoder and Perceptron layer smoothly reach solid points. The histogram figures show that the training, validation and testing errors (Mean-Squared-Error) of autoencoder and perceptron layer are mostly close to 0.0036 and 0.001, respectively. Last two regression analysis which is performed on coded targets and coded prediction. It shows the general training performance of the method on this data set.



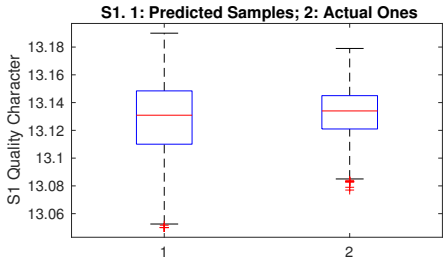
(a) Statistic property of Predicted results of S1



(b) Statistic property of Predicted results of S2

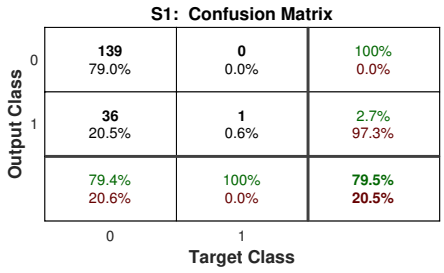


(c) Statistic property of Predicted results of S3

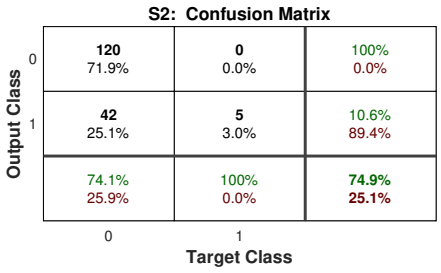


(d) Statistic property of Predicted results of S4

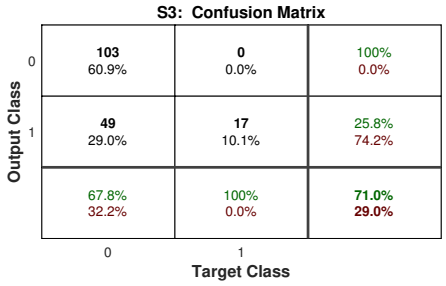
Figure 8.8: Statistic analysis of prediction. The box plot shows the median, 25% quartile, 75% quartile and the range.



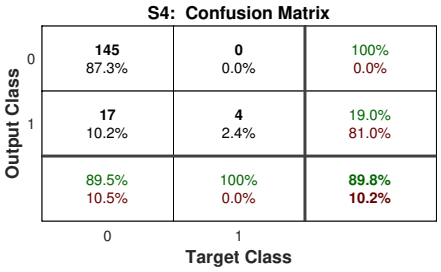
(a)



(b)



(c)



(d)

Figure 8.9: Confusion matrices for 4 testbeds, which includes positive class, negative class, prediction precision, recall rate and F-Measure.

CHAPTER 9

Conclusions and Future Work

This chapter summarizes the research results and concludes this thesis. Section 9.1 summarizes the scientific contribution of this thesis. Section 9.2 revisits the research questions proposed at the beginning of this thesis. Section 9.3 briefly describes the open work and proposes some problems or issues to be addressed in future work.

9.1 Summary of Scientific Contribution

With more and more advanced computing equipment, networks, and sensors being implemented in modern factories, CPPS is facing great challenges in industrial big data processing. In order to design flexible and smart CPPS, researchers and engineers are looking to the AI community. This thesis addresses how deep neural networks and ESNs can be used in industrial data feature extraction, high dimension industrial data compressing, and manufacturing process modeling.

First, it considered how engineers apply statistical features (without unique selection) to model physical processes by adopting complex neural network ESN. This ought to reduce the effort required of engineers when picking up specific features and quickly to adopt ESN for modeling the manufacturing processes. The thesis used the milling procedure as a case study and demonstrated the performance of the proposed solution.

Second, in order to make ESN more effective in industrial applications, e.g., milling process modeling, a novel searching algorithm ZIZO is proposed. It is a heuristic researching strategy. It is combined with gradient descent techniques to search the solution space.

Third, a deep neural network-based industrial data processing platform was developed. Noisy and corrupted sensory data are always the core problem in the manufacturing process modeling. This work began with high dimension data compressing, applying the autoencoder technique and then applying the simple similarity computing method. This platform provided an end-to-end data analysis service.

Fourth, as the semiconductor is one of the most complicated industrial fields, the future status of the wafer during the production procedure was monitored and predicted using a new Bayesian network structure.

Regarding these contributions, there is one underlying common goal: design flexible and smart CPPS. One promising way of constructing smart CPPS is the provision of an end-to-end industrial data processing service. The technical basis for achieving end-to-end service is the smart processing of sensory data, which is the core idea presented in this thesis.

9.2 Research Questions Revisited

RQ 1: *How to select significant features from systems' inputs?*

The solution to this is the complex neural network and deep neural network automatically generating useful features. This is specifically addressed in Chapters 4 and 6. It is shown that without seriously selecting any statistical features, ESN can successfully model the input data. Second, the deep neural network autoencoder was applied to generate the deep representation of inputs, which reduced much of the user effort in selecting features.

RQ 2: *How to do fast data modeling for industrial processes?*

This research question has been addressed in Chapters 4, Chapter 6, Chapter 7, Chapter 8 and Appendix. Some of the existing modeling manufacturing models are based on statistical methods and classical machine learning techniques. Although these models have worked well in the past, with the significant data flow into the industry every day, more efficient methods are required. This solution applies advanced neural computing technology to industrial data processing and modeling. Additionally, some of the neural computing techniques are designed for image processing or natural language processing. These techniques have to be adapted for industrial data processing, especially in the manufacturing sector. For instance, it has been demonstrated that ESN works well for modeling milling process by adopting statistical features, while end-to-end analysis fashion for image processing or audio analytic cannot be directly applied to manufacturing industries. Therefore, Chapter 7 presents the design of a novel multi-bias RNNs based on ESN and demonstrates that it can accurately model milling sensory data.

RQ3: *How to effectively choose the hyper-parameters for Echo-State Networks?*

This research question was addressed with the design of a new random heuristic searching algorithm called ZIZO in Chapter 5. Since the solution space of the hyper-parameter is quite large, ZIZO first generates random hyper-parameter candidates based on uniform distribution. It then selects the sub-optimal solution and re-generates another batch of possible solutions. The critical point of ZIZO is the manner in which it updates its new range after the previous searching.

RQ 4: *What does it mean by End-to-End industrial data processing ?*

End-to-end industrial service is the core idea presented in this thesis. The case studies in Chapters 6, 7 and 8 demonstrate that it works well in modern factories. Chapter 6 presents the deep neural network-based method for modeling the bearing process, Chapter 8 shows how to predict WIP quality by making use of our generic framework, and Chapter 7 demonstrates that with specially designed multi-bias RNNs, with which one can precisely model the milling process.

RQ 5: *Special case study: A novel Bayesian network for predicting the performance of a wafer manufacturing process.*

A novel Bayesian network structure for monitoring and predicting the wafer production status is developed in Appendix. First, the historical data properties are investigated. Basic machine learning methods are then used for learning the parameters of the Bayesian network. In this part of the work, the main focus is solving the question of one-step forward prediction of wafer quality.

Through monitoring, it can detect what is happening and what is going to happen in the next step(s), which is more effective than just naively guessing the root cause of a problem that happened at the last step of manufacturing process.

9.3 Future Work

This thesis presented the algorithms and methods for industry data processing, several specific manufacturing process modeling, and relevant supporting tools for easy application of complex neural networks. Additionally, it also studied the semiconductor industry (Appendix Section), specifically monitoring and predicting the status of the wafer during the production procedure. In order to cover more industrial fields and propose more common practical technology frameworks, several issues to be addressed in future work are summarized as follows:

- Based on Chapter 4, the author combined ESN with statistical features for semi-automatically predicts the milling process status. Through the investigation, it was concluded that ESN is

sensitive to the evolutionary trend of physical systems, which indicates that, first, better control strategies for ESN are needed. Second, the transfer learning concept from the deep learning community may also work for ESN if users apply it to similar industrial fields. The future work: (1) adopt statistical features as much as possible and (2) design adaptive ESN based on the statistical features collected from step (1). The idea is to analyze the statistical properties of systems, and then apply the designed principle to generate models for modeling collected data.

- Chapter 6 proposed an industrial data process platform that could be used for compressing high dimension inputs. A general technical framework for industrial data compression was created, though it considers several compressing techniques, for example, simple autoencoder, variant autoencoder, and convolutional autoencoder. The adoption of other similar algorithms to make this technology framework much stronger is proposed. Similar activities are also planned for Chapter 8.

- A novel hyper-parameter searching algorithm called ZIZO is proposed and applied in Chapter 4 and 7. For future work, improvements to the performance of the weights searching process are proposed, via automatically detecting the optimal or suboptimal structures of MBRC-RNN. The core idea is to keep the size of the RL layers as small as possible while improving the prediction accuracy as much as possible. To generalize, it is important to apply MBRC-RNN to other key machining processes, such as gearing and drilling.

- Additional case study is conducted in the Appendix, i.e., a Bayesian network-based method is proposed for monitoring and predicting the future status of a wafer in the manufacturing process. The model presented in Appendix works well on single step prediction of wafer states. A batch of research investigations was also conducted for multi-step predictions. According to the recent research results, there were no effective multi-step modeling methods. Therefore, future work aims to assess this approach in more experiments to provide further validation. Besides, conducting more experiments by using the proposed diagnosis method and multi-step prognostic method is also planned.

9.4 Conflict of Interests

The thesis has presented several algorithms and methods for solving specific industrial problems. Those publications have been submitted to online databases, including *IEEE, Springer, Taylor & Francis* and so forth. The experiments used publicly accessible dataset. Readers are encouraged to reference these datasets from the NASA Data Repository-Intelligent Systems Division. The author

has finished batches of source code for implementing these methods and algorithms, which are not available to the public, because of interest conflict.

List of Algorithms

Glossary

Industry 4.0 The fourth industry revolution compared with previous ones.. 2–4, 7

Acronyms

AEC auto-encoder correlation-based. xiii, xvii, xviii, xxi, 87–89, 93–100

AI Artificial Intelligence. 4, 5, 14, 15, 35, 46, 88, 133

ANN Artificial Neural Networks. 2, 3

CBM condition-based monitoring. 88

CPPS Cyber-Physical Production Systems. xi, xii, xvi, 2–4, 8, 9, 11–13, 15, 20, 21, 23, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46–48, 50, 52, 54, 69, 102, 133, 134

CPS Cyber-Physical Systems. 1, 2, 46

ESN Echo-State Networks. xi, xii, xvi, xvii, 5–8, 11, 20, 21, 23, 34–36, 55–64, 66–69, 71–80, 82–84, 106, 107, 113, 133–136

FFG Austrian Research Promotion Agency. 2

FFNN feed forward neural networks. 15, 17, 25, 31, 33, 34, 38, 51, 55, 64, 66–69

GNNQP Generative Neural Network for Quality Prediction. 117

HMM Hidden Markov Model. 13, 16

ICT Information Communication Technology. 2, 16

IoT Internet of Things. 1, 2

IS Input Scaling. 56, 58, 64, 73

KF Kalman filter. 14

LR Leaking Rate. xxii, 56–58, 64, 72, 99

MBRC-RNN Multi-Bias Recurrent Neural Networks. x, xiii, xviii, xx, xxii, 5, 101–114, 116, 136, 167

MSE Mean Squared Error. 74, 77, 78, 80, 83, 103, 106

PCA principle components analysis. 13, 16

PCC Pearson correlation coefficient. xxi, 55, 60, 61, 64, 65, 93

Relu rectified linear units. 105, 106

RNN recurrent neural networks. xviii, xxii, 16, 17, 32–35, 56, 59, 67, 68, 71, 73, 101–106, 113, 134, 135

RUL remaining useful life. 13–15

SFAMNN Simplified Fuzzy Adaptive Resonance Theory Map Neural Network. 17

SoR Size of ESN. 56, 58, 64, 67

SoR Sparsity of Reservoir. 58, 64

SR Spectral Radius. 56, 58, 64, 74

TSNN time series neural networks. 55, 64, 67–69

ZIZO Zoom-In-Zoom-Out. ix, xvii, xxii, 7, 8, 71, 74–76, 78, 83, 84, 101–104, 106–108, 111, 112, 114, 133, 135, 136, 139

Bibliography

[AA07] K. Goebel A. Agogino. Best lab, uc berkeley. "milling data set". *NASA Ames Research Center, Moffett Field, CA*, 2007.

[ABBB07] Sofiane Achiche, Luc Baron, Marek Balazinski, and Mokhtar Benaoudia. Online prediction of pulp brightness using fuzzy logic models. *Engineering Applications of Artificial Intelligence*, 20(1):25–36, 2007.

[ABP17] Toyosi Toriola Ademujimi, Michael P Brundage, and Vittaldas V Prabhu. A review of current machine learning techniques used in manufacturing diagnosis. In *IFIP International Conference on Advances in Production Management Systems*, pages 407–415. Springer, 2017.

[ACCPM13] Siam Aumi, Brandon Corbett, Tracy Clarke-Pringle, and Prashant Mhaskar. Data-driven model predictive quality control of batch processes. *AIChE Journal*, 59(8):2852–2861, 2013.

[ACL+11] Allon Adir, Shady Copty, Shimon Landa, Amir Nahir, Gil Shurek, Avi Ziv, Charles Meissner, and John Schumann. A unified methodology for pre-silicon verification and post-silicon validation. In *2011 Design, Automation & Test in Europe*, pages 1–6. IEEE, 2011.

[ACMS+15] Jaouher Ben Ali, Brigitte Chebel-Morello, Lotfi Saidi, Simon Malinowski, and Farhat Fnaiech. Accurate bearing remaining useful life prediction based on weibull distribution and artificial neural network. *Mechanical Systems and Signal Processing*, 56:150–172, 2015.

[AIVRGAMM12] A Alvarado-Iniesta, DJ Valles-Rosales, JL García-Alcaraz, and A Maldonado-Macias. A recurrent neural network for warpage prediction in injection molding. *Journal of applied research and technology*, 10(6):912–919, 2012.

[ASH+12] CS Ai, YJ Sun, GW He, XB Ze, W Li, and K Mao. The milling tool wear monitoring using the acoustic spectrum. *The International Journal of Advanced Manufacturing Technology*, 61(5-8):457–463, 2012.

[ASH13] Fahmi Arif, Nanna Suryana, and Burairah Hussin. A data mining approach for developing quality prediction model in multi-stage manufacturing. *International Journal of Computer Applications*, 69(22), 2013.

[ASM+14] Amir Arifin, Abu Bakar Sulong, Norhamidi Muhamad, Junaidi Syarif, and Mohd Ikram Ramli. Material processing of hydroxyapatite and titanium alloy (ha/ti) composite as implant materials using powder metallurgy: a review. *Materials & Design*, 55:165–175, 2014.

[Bal12] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49, 2012.

[BBBK11] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554, 2011.

[BC+08] Y-lan Boureau, Yann L Cun, et al. Sparse feature learning for deep belief networks. pages 1185–1192, 2008.

[BCHC09] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[BG07] Irad Ben-Gal. Bayesian networks. *Encyclopedia of statistics in quality and reliability*, 2007.

[BGCSV09] Pavel Brazdil, Christophe Giraud-Carrier, Carlos Soares, and Ricardo Vilalta. Metalearning: Concepts and systems. *Metalearning: Applications to Data Mining*, pages 1–10, 2009.

[BJM17] Brett K Beaulieu-Jones and Jason H Moore. Missing data imputation in the electronic health record using deeply learned autoencoders. In *PACIFIC SYMPOSIUM ON BIOCOMPUTING 2017*, pages 207–218. World Scientific, 2017.

[BKN17] Aliasghar Bazdar, Reza Baradaran Kazemzadeh, and Seyed Taghi Akhavan Niaki. Fault diagnosis within multistage machining processes using linear discriminant

analysis: a case study in automotive industry. *Quality Technology & Quantitative Management*, 14(2):129–141, 2017.

[BLA16] Filippo Maria Bianchi, Lorenzo Livi, and Cesare Alippi. Investigating echo-state networks dynamics by means of recurrence analysis. *IEEE transactions on neural networks and learning systems*, 2016.

[BMA+17] José Blasco, Sandra Munera, Nuria Aleixos, Sergio Cubero, and Enrique Molto. Machine vision-based measurement systems for fruit and vegetable quality control in postharvest. In *Measurement, Modeling and Automation in Advanced Food Processing*, pages 71–91. Springer, 2017.

[BMZR15] T Benkedjouh, Kamal Medjaher, Noureddine Zerhouni, and Saïd Rechak. Health assessment and life prediction of cutting tools based on support vector regression. *Journal of Intelligent Manufacturing*, 26(2):213–223, 2015.

[BOL+12] Joschka Boedecker, Oliver Obst, Joseph T Lizier, N Michael Mayer, and Minoru Asada. Information processing in echo state networks at the edge of chaos. *Theory in Biosciences*, 131(3):205–213, 2012.

[BPP07] Sotiris Bersimis, Stelios Psarakis, and John Panaretos. Multivariate statistical process control charts: an overview. *Quality and Reliability engineering international*, 23(5):517–543, 2007.

[BSD17] Yun Bai, Zhenzhong Sun, and Jun Deng. Manufacturing quality prediction based on two-step feature learning approach. In *Sensing, Diagnostics, Prognostics, and Control (SDPC), 2017 International Conference on*, pages 260–263. IEEE, 2017.

[BSF94] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[BSU+15] Filippo Maria Bianchi, Simone Scardapane, Aurelio Uncini, Antonello Rizzi, and Alireza Sadeghian. Prediction of telephone calls load using echo state network with exogenous variables. *Neural Networks*, 71:204–213, 2015.

[BT15] Pramod Bangalore and Lina Bertling Tjernberg. An artificial neural network approach for early fault detection of gearbox bearings. *IEEE Transactions on Smart Grid*, 6(2):980–987, 2015.

[CBC17] Ni-Bin Chang, Kaixu Bai, and Chi-Farn Chen. Integrating multisensor satellite data merging and image reconstruction in support of machine learning for better water quality management. *Journal of environmental management*, 201:227–240, 2017.

[CBHK02] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[Cha18] Chris Chatfield. *Introduction to multivariate analysis*. Routledge, 2018.

[CHC13] Chen-Fu Chien, Chia-Yu Hsu, and Pei-Nong Chen. Semiconductor fault detection and classification for yield enhancement and manufacturing intelligence. *Flexible Services and Manufacturing Journal*, 25(3):367–388, 2013.

[Cle02] Tom Clements. Overview of soap. *Oracle Sun Developer Network*, 2002.

[CLY17] Hongrui Cao, Denghui Li, and Yiting Yue. Root cause identification of machining error based on statistical process control and fault diagnosis of machine tools. *Machines*, 5(3):20, 2017.

[CMB07] Kai-hui Chang, Igor L Markov, and Valeria Bertacco. Automating post-silicon debugging and repair. In *2007 IEEE/ACM International Conference on Computer-Aided Design*, pages 91–98. IEEE, 2007.

[CNEMK16] Raphael Corne, Chandra Nath, Mohamed El Mansori, and Thomas Kurfess. Enhancing spindle power data application with neural network for real-time tool wear/breakage prediction during inconel drilling. *Procedia Manufacturing*, 5:1–14, 2016.

[CTF+14] Marcello Colledani, Tullio Tolio, Anath Fischer, Benoit Iung, Gisela Lanza, Robert Schmitt, and József Váncza. Design and management of manufacturing systems for production quality. *CIRP Annals-Manufacturing Technology*, 63(2):773–796, 2014.

[CTW+08] Wen-Chin Chen, Pei-Hao Tai, Min-Wen Wang, Wei-Jaw Deng, and Chen-Tai Chen. A neural network-based approach for dynamic quality prediction in a plastic injection molding process. *Expert systems with Applications*, 35(3):843–849, 2008.

[CWC10] Pao-Hua Chou, Menq-Jiun Wu, and Kuang-Ku Chen. Integrating support vector machine and genetic algorithm to implement dynamic wafer quality prediction system. *Expert Systems with Applications*, 37(6):4413–4424, 2010.

[CWM18] Caoimhe M Carbery, Roger Woods, and Adele H Marshall. A bayesian network based learning system for modelling faults in large-scale manufacturing. 2018.

[CZC+16] Mariela Cerrada, Grover Zurita, Diego Cabrera, René-Vinicio Sánchez, Mariano Artés, and Chuan Li. Fault diagnosis in spur gears based on genetic algorithm and random forest. *Mechanical Systems and Signal Processing*, 70:87–103, 2016.

[DCW17] Hang Dong, Nan Chen, and Kaibo Wang. Wafer yield prediction using derived spatial variables. *Quality and Reliability Engineering International*, 33(8):2327–2342, 2017.

[DGD18] Tyler Darwin, Roman Garnett, and Dragan Djurdjanovic. Gaussian process regression for virtual metrology of microchip quality and the resulting selective sampling scheme. In *International Conference on the Industry 4.0 model for Advanced Manufacturing*, pages 250–264. Springer, 2018.

[DHS11] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[dJGIC03] Romero-Troncoso Rene de Jesus, Herrera-Ruiz Gilberto, Terol-Villalobos Iván, and Jáuregui-Correa Juan Carlos. Driver current analysis for sensorless tool breakage monitoring of cnc milling machines. *International Journal of Machine Tools and Manufacture*, 43(15):1529–1534, 2003.

[DKN+16] Cyril Drouillet, Jaydeep Karandikar, Chandra Nath, Anne-Claire Journeaux, Mohamed El Mansori, and Thomas Kurfess. Tool life predictions in milling using spindle power with the neural network technique. *Journal of Manufacturing Processes*, 22:161–168, 2016.

[DL15] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3079–3087, 2015.

[DSSW+06] Jianfei Dong, KVR Subrahmanyam, Yoke San Wong, Geok Soon Hong, and AR Mohanty. Bayesian-inference-based neural networks for tool wear estimation. *The International Journal of Advanced Manufacturing Technology*, 30(9-10):797–807, 2006.

[DUM17] Doriana M D'Addona, AMM Sharif Ullah, and D Matarazzo. Tool-wear prediction and pattern-recognition using artificial neural network and dna-based computing. *Journal of Intelligent Manufacturing*, 28(6):1285–1301, 2017.

[DZY15] Guangzhou Diao, Liping Zhao, and Yiyong Yao. A dynamic quality control approach by improving dominant factors based on improved principal component analysis. *International Journal of Production Research*, 53(14):4287–4303, 2015.

[ED01] Levent Eren and Michael J Devaney. Motor bearing damage detection via wavelet analysis of the starting current transient. In *Instrumentation and Measurement Technology Conference, 2001. IMTC 2001. Proceedings of the 18th IEEE*, volume 3, pages 1797–1800. IEEE, 2001.

[Efr92] Bradley Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics*, pages 569–593. Springer, 1992.

[FD97] Paul M Frank and Xianchun Ding. Survey of robust residual generation and evaluation methods in observer-based fault detection systems. *Journal of process control*, 7(6):403–424, 1997.

[Fis25] Ronald Aylmer Fisher. *Statistical methods for research workers*. Genesis Publishing Pvt Ltd, 1925.

[FLT16] Shu-Kai S Fan, Shou-Chih Lin, and Pei-Fang Tsai. Wafer fault detection and key step identification for semiconductor manufacturing using principal component analysis, adaboost and decision tree. *Journal of Industrial and Production Engineering*, 33(3):151–168, 2016.

[FLZ16] Yi Feng, Baochun Lu, and Dengfeng Zhang. Multifractal manifold for rotating machinery fault diagnosis based on detrended fluctuation analysis. *Journal of Vibroengineering*, 18(8), 2016.

[FZJJ09] Fu Zhou Feng, Dong Dong Zhu, Peng Cheng Jiang, and Hao Jiang. Ga-svr based bearing condition degradation prediction. In *Key engineering materials*, volume 413, pages 431–437. Trans Tech Publ, 2009.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[GFD08] F Antonio Galati, David Forrester, and Subhrakanti Dey. Application of the generalised likelihood ratio algorithm to the detection of a bearing fault in a

helicopter transmission. *Australian Journal of Mechanical Engineering*, 5(2):169–176, 2008.

[GLJ+17] Liang Guo, Naipeng Li, Feng Jia, Yaguo Lei, and Jing Lin. A recurrent neural network based health indicator for remaining useful life prediction of bearings. *Neurocomputing*, 240:98–109, 2017.

[GNGGVR16] Paulino José García-Nieto, E García-Gonzalo, JA Vilán Vilán, and A Segade Robleda. A new predictive model based on the pso-optimized support vector machine approach for predicting the milling tool wear from milling runs experimental data. *The International Journal of Advanced Manufacturing Technology*, 86(1-4):769–780, 2016.

[Goe96] Kai Frank Goebel. *Management of uncertainty in sensor validation, sensor fusion, and diagnosis of mechanical systems using soft computing techniques*. University of California, Berkeley, 1996.

[GRP+07] N Ghosh, YB Ravi, A Patra, S Mukhopadhyay, S Paul, AR Mohanty, and AB Chattopadhyay. Estimation of tool wear during cnc milling using neural network-based sensor fusion. *Mechanical Systems and Signal Processing*, 21(1):466–479, 2007.

[HG12] Wafik Hachicha and Ahmed Ghorbel. A survey of control-chart pattern-recognition literature (1991–2010) based on a new conceptual classification scheme. *Computers & Industrial Engineering*, 63(1):204–222, 2012.

[HHB+17] Ramin M Hasani, Dieter Haerle, Christian F Baumgartner, Alessio R Lomuscio, and Radu Grosu. Compositional neural-network modeling of complex analog circuits. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 2235–2242. IEEE, 2017.

[HHG16] Ramin M Hasani, Dieter Haerle, and Radu Grosu. Efficient modeling of complex analog integrated circuits using neural networks. In *Ph. D. Research in Microelectronics and Electronics (PRIME), 2016 12th Conference on*, pages 1–4. IEEE, 2016.

[Hin07] Geoffrey E Hinton. Learning multiple layers of representation. *Trends in cognitive sciences*, 11(10):428–434, 2007.

[HKY07] Qingbo He, Fanrang Kong, and Ruqiang Yan. Subspace-based gearbox condition monitoring by kernel principal component analysis. *Mechanical Systems and Signal Processing*, 21(4):1755–1772, 2007.

[HOT06] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HWG17] Ramin M Hasani, Guodong Wang, and Radu Grosu. An automated auto-encoder correlation-based health-monitoring and prognostic method for machine bearings. *arXiv preprint arXiv:1703.06272*, 2017.

[HYH+05] Xiaofei He, Shuicheng Yan, Yuxiao Hu, Partha Niyogi, and Hong-Jiang Zhang. Face recognition using laplacianfaces. *IEEE transactions on pattern analysis and machine intelligence*, 27(3):328–340, 2005.

[HYM+16] Young-Sun Hong, Hae-Sung Yoon, Jong-Seol Moon, Young-Man Cho, and Sung-Hoon Ahn. Tool-wear monitoring during micro-end milling using wavelet packet transform and fisher's linear discriminant. *International Journal of Precision Engineering and Manufacturing*, 17(7):845–855, 2016.

[ILS97] Anna M Ison, Wei Li, and Costas J Spanos. Fault diagnosis of plasma etch equipment. In *Semiconductor Manufacturing Conference Proceedings, 1997 IEEE International Symposium on*, pages B49–B52. IEEE, 1997.

[IS96] A Ison and Costas J Spanos. Robust fault detection and fault classification of semiconductor manufacturing equipment. In *Proceedings of the Fifth International Symposium on Semiconductor Manufacturing*, pages 1–4. Citeseer, 1996.

[IvDZBP04] K Ishu, Tijn van Der Zant, Vlatko Becanovic, and P Ploger. Identification of motion with echo state network. In *OCEANS'04. MTTS/IEEE TECHNO-OCEAN'04*, volume 3, pages 1205–1210. IEEE, 2004.

[Jae01] Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13, 2001.

[Jae07] Herbert Jaeger. Echo state network. *Scholarpedia*, 2(9):2330, 2007.

[JH04] Herbert Jaeger and Harald Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science*, 304(5667):78–80, 2004.

[JLB06] Andrew KS Jardine, Daming Lin, and Dragan Banjevic. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical systems and signal processing*, 20(7):1483–1510, 2006.

[JLL+16] Feng Jia, Yaguo Lei, Jing Lin, Xin Zhou, and Na Lu. 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.

[JLPS07] Herbert Jaeger, Mantas Lukoševičius, Dan Popovici, and Udo Siewert. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural networks*, 20(3):335–352, 2007.

[JMM09] AB Johnston, LP Maguire, and TM McGinnity. Downstream performance prediction for a manufacturing system using neural networks and six-sigma improvement techniques. *Robotics and computer-integrated manufacturing*, 25(3):513–521, 2009.

[KB14] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KCL17] Dongdong Kong, Yongjie Chen, and Ning Li. Force-based tool wear estimation for milling process using gaussian mixture hidden markov models. *The International Journal of Advanced Manufacturing Technology*, pages 1–13, 2017.

[Kel09] Kevin Kelly. *Out of control: The new biology of machines, social systems, and the economic world*. Hachette UK, 2009.

[KH92] Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957, 1992.

[KHCL17] Hung-An Kao, Yan-Shou Hsieh, Cheng-Hui Chen, and Jay Lee. Quality prediction modeling for multistage manufacturing based on classification and association rule mining. In *MATEC Web of Conferences*, volume 123, page 00029. EDP Sciences, 2017.

[KN17] Yuting Kong and Dong Ni. A practical yield prediction approach using inline defect metrology data for system-on-chip integrated circuits. In *Automation Science and Engineering (CASE), 2017 13th IEEE Conference on*, pages 744–749. IEEE, 2017.

[KPK+16] Seokgoo Kim, Sungho Park, Ju-Won Kim, Junghwa Han, Dawn An, Nam Ho Kim, and Joo-Ho Choi. A new prognostics approach for bearing based on entropy decrease and comparison with existing methods. *Annual Conference of the Prognostics and Health management Society*, 2016.

[KSWMS13] James K Kimotho, Christoph Sondermann-Woelke, Tobias Meyer, and Walter Sextro. Machinery prognostic method based on multi-class support vector machines and hybrid differential evolution-particle swarm optimization. *Chemical Engineering Transactions*, 33:619–624, 2013.

[KŢBP17] Ingrid Kovacs, Marina Ţopa, Andi Buzo, and Georg Pelz. An accurate yield estimation approach for multivariate non-normal data in semiconductor quality analysis. In *Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), 2017 14th International Conference on*, pages 1–4. IEEE, 2017.

[LBA18] Lorenzo Livi, Filippo Maria Bianchi, and Cesare Alippi. Determination of the edge of criticality in echo state networks through fisher information maximization. *IEEE Transactions on Neural Networks and Learning Systems*, 29(3):706–717, 2018.

[LBG15] Christiane Lemke, Marcin Budka, and Bogdan Gabrys. Metalearning: a survey of trends and technologies. *Artificial intelligence review*, 44(1):117–130, 2015.

[LBJ17] Sigurd Løkse, Filippo Maria Bianchi, and Robert Jenssen. Training echo state networks with regularization through dimensionality reduction. *Cognitive Computation*, 9(3):364–378, 2017.

[LBK15] Jay Lee, Behrad Bagheri, and Hung-An Kao. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters*, 3:18–23, 2015.

[LHV+06] Dae-Eun Lee, Inkil Hwang, Carlos MO Valente, JFG Oliveira, and David A Dornfeld. Precision manufacturing process monitoring with acoustic emission.

In *Condition Monitoring and Control for Intelligent Manufacturing*, pages 33–54. Springer, 2006.

[LHW12] Decai Li, Min Han, and Jun Wang. Chaotic time series prediction based on a novel robust echo state network. *IEEE Transactions on Neural Networks and Learning Systems*, 23(5):787–799, 2012.

[LHZH07] Yaguo Lei, Zhengjia He, Yanyang Zi, and Qiao Hu. Fault diagnosis of rotating machinery based on multiple anfis combination with gas. *Mechanical systems and signal processing*, 21(5):2280–2294, 2007.

[Lia14] Linxia Liao. Discovering prognostic features using genetic programming in remaining useful life prediction. *IEEE Transactions on Industrial Electronics*, 61(5):2464–2472, 2014.

[LKM+16] Doowon Leey, Tom Kolanz, Arkadiy Morgenshteinz, Vitali Sokhinz, Ronny Moradz, Avi Ziv, and Valeria Bertaccoy. Probabilistic bug-masking analysis for post-silicon tests in microprocessor verification. In *Proceedings of the 53rd Annual Design Automation Conference*, page 24. ACM, 2016.

[LKY14] Jay Lee, Hung-An Kao, and Shanhu Yang. Service innovation and smart analytics for industry 4.0 and big data environment. *Procedia Cirp*, 16:3–8, 2014.

[Low00] William M Lowe. Transaction based windowing methodology for pre-silicon verification, June 6 2000. US Patent 6,073,194.

[LQY+07] J Lee, H Qiu, G Yu, J Lin, et al. Bearing data set. *IMS, University of Cincinnati, NASA Ames Prognostics Data Repository, Rexnord Technical Services*, 2007.

[LSL11] Chi-Jie Lu, Yuehjen E Shao, and Po-Hsun Li. Mixture control chart patterns recognition using independent component analysis and support vector machine. *Neurocomputing*, 74(11):1908–1914, 2011.

[Luk12] Mantas Lukoševičius. A practical guide to applying echo state networks. In *Neural networks: Tricks of the trade*, pages 659–686. Springer, 2012.

[LWH17] Zhimeng Li, Guofeng Wang, and Gaiyun He. Milling tool wear state recognition based on partitioning around medoids (pam) clustering. *The International Journal of Advanced Manufacturing Technology*, 88(5-8):1203–1213, 2017.

[LWZ+14] Jay Lee, Fangji Wu, Wenyu Zhao, Masoud Ghaffari, Linxia Liao, and David Siegel. Prognostics and health management design for rotary machinery systems-reviews, methodology and applications. *Mechanical systems and signal processing*, 42(1):314–334, 2014.

[LXHW16] Dongdong Li, Boping Xiao, Haiping Huang, and Aoqing Wang. A method of predicting demand for aircraft follow-up spare based on discrete particle swarm optimization algorithm and rbf neural network. In *Industrial Engineering and Engineering Management (IEEM), 2016 IEEE International Conference on*, pages 1488–1492. IEEE, 2016.

[LZL12] Ming Luo, Ying Zheng, and Shujie Liu. Data-based fault-tolerant control of the semiconductor manufacturing process based on k-nearest neighbor nonparametric regression. In *Intelligent Control and Automation (WCICA), 2012 10th World Congress on*, pages 3008–3012. IEEE, 2012.

[MAD+15] Mariam Melhem, Bouchra Ananou, Mohand Djeziri, Mustapha Ouladsine, and Jacque Pinaton. Prediction of the wafer quality with respect to the production equipments data. *IFAC-PapersOnLine*, 48(21):78–84, 2015.

[MAO+17] Mariam Melhem, Bouchra Ananou, Mustapha Ouladsine, Michel Combal, and Jacques Pinaton. Product quality prediction using alarm data: Application to the semiconductor manufacturing process. In *Control and Automation (MED), 2017 25th Mediterranean Conference on*, pages 1332–1338. IEEE, 2017.

[MAOP16] Mariam Melhem, Bouchra Ananou, Mustapha Ouladsine, and Jacques Pinaton. Regression methods for predicting the product鈥 quality in the semiconductor manufacturing process. *IFAC-PapersOnLine*, 49(12):83–88, 2016.

[Mit16] Amitava Mitra. *Fundamentals of quality control and improvement*. John Wiley & Sons, 2016.

[MKK+12] Vladimiro Miranda, Jakov Krstulovic, Hrvoje Keko, Cristiano Moreira, and Jorge Pereira. Reconstructing missing data in state estimation with autoencoders. *IEEE Transactions on power systems*, 27(2):604–611, 2012.

[MKZ+16] Klas Meyer, Simon Kern, Nicolai Zientek, Gisela Guthausen, and Michael Maiwald. Process control with compact nmr. *TrAC Trends in Analytical Chemistry*, 83:39–52, 2016.

[MMN18] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018.

[MMZ14] Ahmed Mosallam, Kamal Medjaher, and Noureddine Zerhouni. Time series trending for condition assessment and prognostics. *Journal of manufacturing technology management*, 25(4):550–567, 2014.

[MNM02] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560, 2002.

[Møl93] Martin Fodslette Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks*, 6(4):525–533, 1993.

[Mon09] Douglas C Montgomery. *Introduction to statistical quality control*. John Wiley & Sons (New York), 2009.

[Mon14] László Monostori. Cyber-physical production systems: roots, expectations and r&d challenges. *Procedia Cirp*, 17:9–13, 2014.

[MR16] Sathyan Munirathinam and Balakrishnan Ramadoss. Predictive models for equipment fault detection in the semiconductor manufacturing process. *International Journal of Engineering and Technology*, 8(4):273, 2016.

[MSN10] Subhasish Mitra, Sanjit A Seshia, and Nicola Nicolici. Post-silicon validation opportunities, challenges and recent advances. In *Proceedings of the 47th Design Automation Conference*, pages 12–17. ACM, 2010.

[NMM02] Thomas Natschläger, Wolfgang Maass, and Henry Markram. The" liquid computer": A novel strategy for real-time computing on time series. *Special issue on Foundations of Information Processing of TELEMATIK*, 8(LNMC-ARTICLE-2002-005):39–43, 2002.

[NMM07] Fulufhelo V Nelwamondo, Shakir Mohamed, and Tshilidzi Marwala. Missing data: A comparison of neural network and expectation maximization techniques. *Current Science*, pages 1514–1521, 2007.

[NTC+13] Mélanie Noyel, Philippe Thomas, Patrick Charpentier, Andre Thomas, and Thomas Brault. Implantation of an on-line quality process monitoring. In

Industrial Engineering and Systems Management (IESM), Proceedings of 2013 International Conference on, pages 1–6. IEEE, 2013.

[OAC06] Sadettin Orhan, Nizami Aktürk, and Veli Celik. Vibration monitoring for defect diagnosis of rolling element bearings as a predictive maintenance tool: Comprehensive case studies. *Ndt & E International*, 39(4):293–298, 2006.

[Ols01] Robert T Olszewski. Generalized feature extraction for structural pattern recognition in time-series data. Technical report, DTIC Document, 2001.

[PCo16] PCoE. PCoE, Prognostics Center of Excellence of NASA: The prognostics data repository. `https://ti.arc.nasa.gov/tech/dash/pcoe/prognostic-data-repository/`, Accessed 2016.

[PGK16] Aditi B Patil, Jitendra A Gaikwad, and Jayant V Kulkarni. Bearing fault diagnosis using discrete wavelet transform and artificial neural network. In *Applied and Theoretical Computing and Communication Technology (iCATccT), 2016 2nd International Conference on*, pages 399–405. IEEE, 2016.

[PPY01] W Tse Peter, YH Peng, and Richard Yam. Wavelet analysis and envelope detection for rolling element bearing fault diagnosis-their effectiveness and flexibilities. *Journal of vibration and acoustics*, 123(3):303–310, 2001.

[PRS08] P Palanisamy, I Rajendran, and S Shanmugasundaram. Prediction of tool wear using regression and ann models in end-milling operation. *The International Journal of Advanced Manufacturing Technology*, 37(1):29–41, 2008.

[Pug89] G Allen Pugh. Synthetic neural networks for process control. *Computers & industrial engineering*, 17(1-4):24–26, 1989.

[Qia99] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[Qin12] S Joe Qin. Survey on data-driven industrial process monitoring and diagnosis. *Annual reviews in control*, 36(2):220–234, 2012.

[QLLY06] Hai Qiu, Jay Lee, Jing Lin, and Gang Yu. Wavelet filter-based weak signature detection method and its application on rolling element bearing prognostics. *Journal of sound and vibration*, 289(4):1066–1090, 2006.

[RDC06] Ramkumar Rajagopal and Enrique Del Castillo. A bayesian method for robust tolerance control and parameter design. *IIE Transactions*, 38(8):685–697, 2006.

[RHW85] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

[RHZC13] Javad Soltani Rad, Ensieh Hosseini, Youmin Zhang, and Chevy Chen. Online tool wear monitoring and estimation using power signals and s-transform. In *Control and Fault-Tolerant Systems (SysTol), 2013 Conference on*, pages 234–238. IEEE, 2013.

[RM14] Lim Chi Keong Reuben and David Mba. Diagnostics and prognostics using switching kalman filters. *Structural Health Monitoring*, 13(3):296–306, 2014.

[Rya11] Thomas P Ryan. *Statistical methods for quality improvement*. John Wiley & Sons, 2011.

[SA07] DR Salgado and FJ Alonso. An approach based on current and sound signals for in-process tool wear monitoring. *International Journal of Machine Tools and Manufacture*, 47(14):2140–2152, 2007.

[SFS+17] Daiji Sakurai, Yoshikazu Fukuyama, Adamo Santana, Yu Kawamura, Kenya Murakami, Tatsuya Iizaka, and Tetsuro Matsui. Estimation of missing data of showcase using artificial neural networks. In *Computational Intelligence and Applications (IWCIA), 2017 IEEE 10th International Workshop on*, pages 15–18. IEEE, 2017.

[SGLMH13] Libo Sun, Shunsheng Guo, Yibing Li, and Maw Maw Htay. Quality prediction model based on mechanical product gene. *Advanced Science, Engineering and Medicine*, 5(10):1090–1096, 2013.

[SH07a] Zhiwei Shi and Min Han. Support vector echo-state machine for chaotic time-series prediction. *IEEE Transactions on Neural Networks*, 18(2):359–372, 2007.

[SH07b] Mark D Skowronski and John G Harris. Automatic speech recognition using a predictive echo state network classifier. *Neural networks*, 20(3):414–423, 2007.

[SHCC06] Yu-Chuan Su, Min-Hsiung Hung, Fan-Tien Cheng, and Yeh-Tung Chen. A processing quality prognostics scheme for plasma sputtering in tft-lcd manufacturing. *IEEE Transactions on semiconductor manufacturing*, 19(2):183–194, 2006.

[She31] Walter Andrew Shewhart. *Economic control of quality of manufactured product*. ASQ Quality Press, 1931.

[SPMK16] Michał Szydłowski, Bartosz Powałka, Marcin Matuszak, and Paweł Kochmański. Machine vision micro-milling tool wear inspection by image reconstruction and light reflectance. *Precision Engineering*, 44:236–244, 2016.

[SS05] B Satish and NDR Sarma. A fuzzy bp approach for diagnosis and prognosis of bearing faults in induction motors. In *Power Engineering Society General Meeting, 2005. IEEE*, pages 2291–2294. IEEE, 2005.

[SS16] Drako Stanisavljevic and Michael Spitzer. A review of machine learning in manufacturing. In *In Proceedings of International Conference on Knowledge Technologies and Data-driven Business, Graz, Austria*, 2016.

[SSZ+16] Wenjun Sun, Siyu Shao, Rui Zhao, Ruqiang Yan, Xingwu Zhang, and Xuefeng Chen. A sparse auto-encoder-based deep neural network approach for induction motor faults classification. *Measurement*, 89:171–178, 2016.

[SWP16] Simone Scardapane, Dianhui Wang, and Massimo Panella. A decentralized training algorithm for echo state networks in distributed big data applications. *Neural Networks*, 78:65–74, 2016.

[SZLP11] Xiao Fang Sun, Peng Fei Zhu, Ying Jun Lu, and Hai Tian Pan. Prediction of quality index of injection-molded parts by using artificial neural networks. In *Advanced Materials Research*, volume 291, pages 432–439. Trans Tech Publ, 2011.

[TDS+16] Raghuveer Thirukovalluru, Sonal Dixit, Rahul K Sevakula, Nishchal K Verma, and Al Salour. Generating feature sets for fault diagnosis using denoising stacked auto-encoder. In *Prognostics and Health Management (ICPHM), 2016 IEEE International Conference on*, pages 1–7. IEEE, 2016.

[TMMZT12] Diego Alejandro Tobon-Mejia, Kamal Medjaher, Noureddine Zerhouni, and Gerard Tripot. A data-driven failure prognostics method based on mixture of gaussians hidden markov models. *IEEE Transactions on Reliability*, 61(2):491–503, 2012.

[TÖU13] S Taghizadeh, A Özdemir, and O Uluer. Warpage prediction in plastic injection molded part using artificial neural network. *Iranian Journal of Science and Technology. Transactions of Mechanical Engineering*, 37(M2):149, 2013.

[VS09] David Verstraeten and Benjamin Schrauwen. On the quantification of dynamics in reservoir computing. *Artificial Neural Networks–ICANN 2009*, pages 985–994, 2009.

[VSH02] Jiri L Vasat, Andrei Stefanescu, and Thomas M Hanley. Semiconductor wafer manufacturing process, April 23 2002. US Patent 6,376,395.

[Wan11] David Wang. Robust data-driven modeling approach for real-time final product quality prediction in batch process operation. *IEEE Transactions on Industrial Informatics*, 7(2):371–377, 2011.

[WBGC99] Qingsu Wang, Gerald Barnett, R Michael Greig, and Yi Cheng. System and method for performing real time data acquisition, process modeling and fault detection of wafer fabrication processes, 1999. US Patent 5,859,964.

[WCQS18] Tian Wang, Yang Chen, Meina Qiao, and Hichem Snoussi. A fast and robust convolutional neural network-based defect detection model in product quality control. *The International Journal of Advanced Manufacturing Technology*, 94(9-12):3465–3471, 2018.

[Wer90] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[WG] Guodong Wang and Radu Grosu. Milling-tool wear-condition prediction with statistic analysis and echo-state networks. In *Proceedings of S2M'16, the International Conference on Sustaniable Smart Manufacturing*.

[WG16a] Guodong Wang and Radu Grosu. Milling-tool wear-condition prediction with statistic analysis and echo-state networks. In *Proceedings of S2M'16, the International Conference on Sustaniable Smart Manufacturing*, 2016.

[WG16b] Peng Wang and Robert X Gao. Stochastic tool wear prediction for sustainable manufacturing. *Procedia CIRP*, 48:236–241, 2016.

[WGY17] Peng Wang, Robert X Gao, and Ruqiang Yan. A deep learning-based approach to material removal rate prediction in polishing. *CIRP Annals*, 66(1):429–432, 2017.

[WHZG17] Guodong Wang, Ramin M Hasani, Yungang Zhu, and Radu Grosu. A novel bayesian network-based fault prognostic method for semiconductor manufacturing process. In *Industrial Technology (ICIT), 2017 IEEE International Conference on*, pages 1450–1454. IEEE, 2017.

[Wie61] Norbert Wiener. *Cybernetics or Control and Communication in the Animal and the Machine*, volume 25. MIT press, 1961.

[WIT14] Thorsten Wuest, Christopher Irgens, and Klaus-Dieter Thoben. An approach to monitoring quality in manufacturing using supervised machine learning on product state data. *Journal of Intelligent Manufacturing*, 25(5):1167–1180, 2014.

[WLS10] David Wang, Jun Liu, and Rajagopalan Srinivasan. Data-driven soft sensor approach for quality prediction in a refining process. *IEEE Transactions on Industrial Informatics*, 6(1):11–17, 2010.

[WLT+18] Hao Wang, Bo Li, Seung Hoon Tong, In-Kap Chang, and Kaibo Wang. A discrete spatial model for wafer yield prediction. *Quality Engineering*, 30(2):169–182, 2018.

[WPZ+16] Yu Wang, Yizhen Peng, Yanyang Zi, Xiaohang Jin, and Kwok-Leung Tsui. A two-stage data-driven-based prognostic approach for bearing degradation problem. *IEEE Transactions on Industrial Informatics*, 12(3):924–932, 2016.

[WWB07] James D Williams, William H Woodall, and Jeffrey B Birch. Statistical monitoring of nonlinear product and process quality profiles. *Quality and Reliability Engineering International*, 23(8):925–941, 2007.

[WWG13] J Wang, P Wang, and RX Gao. Tool life prediction for sustainable manufacturing. *10.14279/depositonce-3753*, 2013.

[WZ89] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[WZL15] Wenchuan Wang, Miao Zhang, and Xinggao Liu. Improved fruit fly optimization algorithm optimized wavelet neural network for statistical data modeling for industrial polypropylene melt index prediction. *Journal of Chemometrics*, 29(9):506–513, 2015.

[WZZS11] Fengtao Wang, Yangyang Zhang, Bin Zhang, and Wensheng Su. Application of wavelet packet sample entropy in the forecast of rolling element bearing fault trend. In *Multimedia and Signal Processing (CMSP), 2011 International Conference on*, volume 2, pages 12–16. IEEE, 2011.

[XLL17] Fangzheng Xue, Qian Li, and Xiumin Li. The combination of circle topology and leaky integrator neurons remarkably improves the performance of echo state network on time series prediction. *PloS one*, 12(7):e0181816, 2017.

[XLP05] Dongming Xu, Jing Lan, and Jose C Principe. Direct adaptive control: an echo state network and genetic algorithm approach. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 3, pages 1483–1486. IEEE, 2005.

[XLX+17] Min Xia, Teng Li, Lin Xu, Lizhi Liu, and Clarence W de Silva. Fault diagnosis for rotating machinery using multiple sensors and convolutional neural networks. *IEEE/ASME Transactions on Mechatronics*, 2017.

[XYY+17] Shuai Xiao, Junchi Yan, Xiaokang Yang, Hongyuan Zha, and Stephen M Chu. Modeling the intensity function of point process via recurrent neural networks. In *AAAI*, pages 1597–1603, 2017.

[YL12] Lei Yang and Jay Lee. Bayesian belief network-based approach for diagnostics and prognostics of semiconductor manufacturing systems. *Robotics and Computer-Integrated Manufacturing*, 28(1):66–74, 2012.

[YSMP02] D-M Yang, AF Stronach, P MacConnell, and J Penman. Third-order spectral techniques for the diagnosis of motor bearing condition using artificial neural networks. *Mechanical systems and signal processing*, 16(2-3):391–411, 2002.

[Yu12a] Jianbo Yu. Health condition monitoring of machines based on hidden markov model and contribution analysis. *IEEE Transactions on Instrumentation and Measurement*, 61(8):2200–2211, 2012.

[Yu12b] Jie Yu. Online quality prediction of nonlinear and non-gaussian chemical processes with shifting dynamics using finite mixture model based gaussian process regression approach. *Chemical Engineering Science*, 82:22–30, 2012.

[YWG16] Shen Yin, Guang Wang, and Huijun Gao. Data-driven process monitoring based on modified orthogonal projections to latent structures. *IEEE Transactions on Control Systems Technology*, 24(4):1480–1487, 2016.

[YX09] Jian-bo Yu and Li-feng Xi. A neural network ensemble-based model for on-line monitoring and diagnosis of out-of-control signals in multivariate manufacturing processes. *Expert systems with applications*, 36(1):909–921, 2009.

[YZ15] Shen Yin and Xiangping Zhu. Intelligent particle filter and its application to fault detection of nonlinear system. *IEEE Transactions on Industrial Electronics*, 62(6):3852–3861, 2015.

[YZZ07] Junyan Yang, Youyun Zhang, and Yongsheng Zhu. Intelligent fault diagnosis of rolling element bearing based on svms and fractal dimension. *Mechanical Systems and Signal Processing*, 21(5):2012–2024, 2007.

[ZD09] M Shafafi Zenoozian and Sakamon Devahastin. Application of wavelet transform coupled with artificial neural network for predicting physicochemical properties of osmotically dehydrated pumpkin. *Journal of Food Engineering*, 90(2):219–227, 2009.

[Zei12] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[ZLJT12] Yungang Zhu, Dayou Liu, Haiyang Jia, and Dafferianto Trinugroho. Incremental learning of bayesian networks based on chaotic dual-population evolution strategies and its application to nanoelectronics. *Journal of Nanoelectronics and Optoelectronics*, 7(2):113–118, 2012.

[ZSV14] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[ZWC16] Linmiao Zhang, Kaibo Wang, and Nan Chen. Monitoring wafers's geometric quality using an additive gaussian process model. *IIE Transactions*, 48(1):1–15, 2016.

[ZY17] Kunpeng Zhu and Xiaolong Yu. The monitoring of micro milling tool wear conditions by wear area estimation. *Mechanical Systems and Signal Processing*, 93:80–91, 2017.

Appendix

A. Principle of MBRC-RNN (Chapter 7)

In order to simply present how MBRC-RNN works, the neural structure of the proposed neural network is simplified into an abstract structure, which is depicted in 1. First, the entire normalized

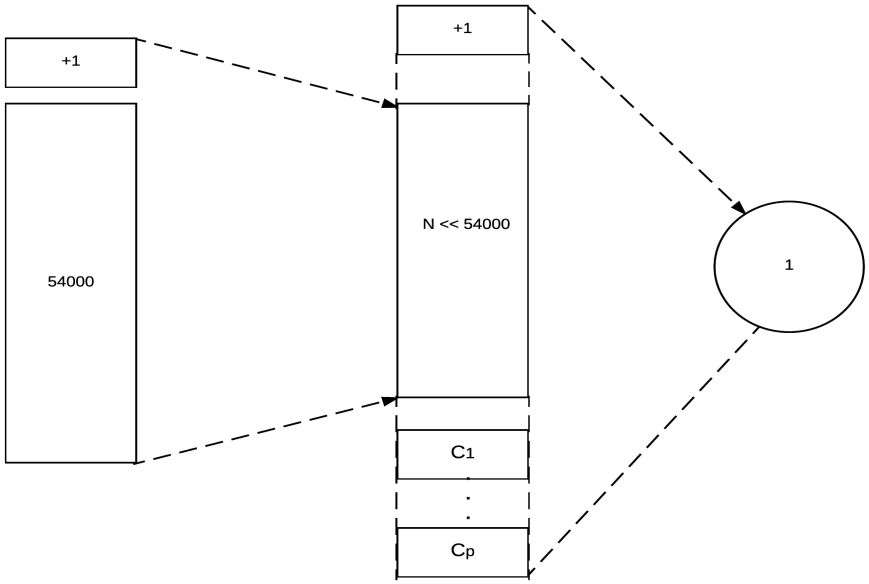


Figure 1: Principle of MBRC-RNN neural network. $N << 54000$ is the size of hidden layer. It is much less than 54000. In order to merge the noisy high dimension inputs, the inputs are shrunk to a much lower dimension, which is the output of neurons from the hidden layer.

sensory data is fed into the neural network. In this work, it has 54000 inputs with 1 bias. Second, the recurrent hidden layer fuses the input data and merges into much lower dimensions. Finally, linearly combined states of hidden layer are applied to model the tool wear status.

B. Industry Data Processing with Bayesian Network: Quality Prediction

This chapter develops a Bayesian network-based prognostic approach to monitoring critical systems deviation in the semiconductor manufacturing field. A Bayesian network is adopted to present relationships between the current process status, the fault of current step and the coming steps. The network proposed is organized in a hybrid way and is capable of processing continuous and discrete variables. Once the structure of the Bayesian network is decided, an exploration of the parameters is essential. A learning method is also designed to tackle the problem. With such learning ability, even it is only modeling one step ahead, failures will be predicted for multiple steps in advance.

This part of the work covers the preparation for exploring the possibility of combining neural computing methods with a Bayesian network to create explainable methods. A Bayesian network will be used to present a clear structure of how a system updates its states, step-by-step. Neural networks will then be adopted for modeling one specific state.

The rest of the work is organized as follows: It first gives the definition and notations of the problem and provides a brief background on Bayesian networks. The proposed approach is described. Then, the experiments for verifying the proposed method are presented. Finally, the conclusions are summarized.

Problem & Preliminaries

Unexpected faults always cause plenty of loss to manufactures every year, especially in semiconductor field. Prognostic and Diagnostic have been discussed and explored decades, recent great achievement in smart fault detection proposes some promising methods for better monitoring, predicting and controlling the products quality in semiconductors manufacturing. One of the issues need to be addressed urgently is precisely detecting abnormal behaviors of production tools in a very early stage.

In this part of work, we aim at investigating Bayesian Networks based approaches for solving such urgent problems, due to the fact that Bayesian Networks clearly present the mathematical steps of addressing problems from very beginning step to the final outputs. We designed a novel Bayesian network wiht the capability of processing continuous and discrete inputs at the same time, specifically the continuous inputs used for representing critical physical faults, the discrete inputs describe quality control data accordingly.

Various installed sensors are used for monitoring current status of manufacturing systems. These collected information then fed into the proposed network. The prognostic system will perform accuracy multi-step prediction which efficiently guarantees the fabrication process runs properly. To efficiently explore parameters of Bayesian network, we also proposed a layer-wise method to these parameters. To demonstrate the performance of our prognostic model, several experiments were conducted on a wafer fabrication dataset where our model performs precise next-step fault prognostic by using the control sensory data.

We assume that the semiconductor fabrication process constituted by m steps. In each step, there are some sensors installed for monitoring its status, specifically n_t sensors at the time-step t. In each step, our goal is to fully make use of the collected sensory information to detect and predict whether there is or will have fault in the coming time steps or not.

Bayesian Network

A Bayesian network is a graphical model representing the joint probability distribution of n variables $X = \{X_1, X_2, ..., X_n\}$. It includes two components: G and Θ. G is a directed acyclic graph where each node of the graph represents a random variable. Each arc represents a dependency relationships between variables. $\Theta = \{P(X_i|\Pi_i), 1 \leq i \leq n\}$ represents the conditional probabilities distribution of each node given the values of its parent nodes where Π_i stands for the parent set of X_i in the network [ZLJT12]. For step t, we construct a Bayesian network as shown in Figure 2: Our proposed Bayesian network constituted by n_t continuous variables and 2 discrete variables. The continuous variables are placed in lower levels. $X_1, ..., X_{n_t}$ representing the n_t sensors monitoring the status at step t. The value of these variables are the sensory outputs. The discrete variables on the top level, i.e., Y_t and Y_{t+1} predict whether there is any fault at current step and next step, respectively. Y_t and Y_{t+1} are binary variables. If they assigned as 1, it implies that the status is faultless, otherwise it suggests a faulty status. It is worth to note that the described network is not a general Bayesian network, it realizes a modified network which contains both continuous and discrete variables. The structure of the Bayesian network is designed as following: A fault varies the value of the sensors, so there are arcs from Y_t and Y_{t+1} to each variable of $X_1, ..., X_{n_t}$. A defect in the current step may result in the occurrence of a fault in the next step. Therefore, there exist an arc from Y_t to Y_{t+1}. It can use

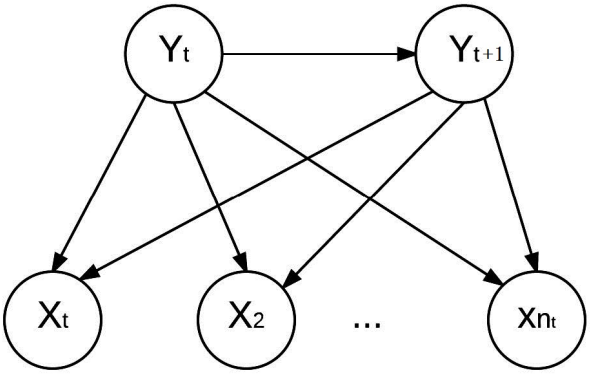


Figure 2: Proposed Bayesian network for prognostic.

$P(Y_t|X_1, ..., X_{n_t})$ to detect whether there is a fault in the step t, and employ $P(Y_{t+1}|X_1, ..., X_{n_t})$ to predict whether the next step is faulty or not.

Parameter learning of the proposed Bayesian network

Based on our proposed Bayesian network, the next step is to explore its parameters. We trained our model i.e., Bayesian network by adopting collected wafer data set stored by a semiconductor corporation.

The parameters of a Bayesian network are the conditional probability of each variable given the values of its parent variables. For X_1, ... , X_{n_t} , conventionally a specific probability distribution is assumed for $P(X_i|Y_t, Y_{t+1})$. To be precisely:

$$P(X_i = x|Y_t = j, Y_{t+1} = k) = \frac{1}{\sqrt(2\pi)\sigma_{ijk}} exp(-\frac{(x - \mu_{ijk})^2}{2\sigma_{ijk}^2}) \tag{1}$$

where $1 \leq i \leq n_t$, x is the value of X_i, j and k stand for the possible values of Y_t and Y_{t+1}, respectively. σ_{ijk} represents the variance of the normal distribution and μ_{ijk} appears for the mean of the distribution.

To obtain the conditional distribution, we compute the mean and variance for each variable (X_1, ..., X_{n_t}) as follows:

$$\mu_{ijk} = \frac{\sum_{l=1}^{M}(x_{i,l} \cdot \delta_{jk,l})}{\sum_{l=1}^{M} \delta_{jk,l}} \tag{2}$$

$$\sigma_{ijk}^2 = \frac{\sum_{l=1}^{M}((x_{i,l} - \mu_{ijk}) \cdot \delta_{jk,l})}{\sum_{l=1}^{M} \delta_{jk,l}} \tag{3}$$

where

$$\delta_{jk,l} = \begin{cases} 1, & \text{if } Y_{t,l} = j \text{ and } y_{t+1,l} = k, \\ 0, & \text{otherwise.} \end{cases}$$

$1 \leq i \leq nt$, M is the total training instances, $x_{i,l}$ is the value of sample l within training data set, and $y_{t,l}$ and $y_{t+1,l}$ are the sample l's value of Y_t and Y_{t+1} within the training data set as well.

A conventional method for learning the parameters of Bayesian network is performed, specifically $P(Y_t)$ and $P(Y_{t+1}|Y_t)$ are obtained as follows:

$$P(Y_t = j) = \frac{N_j}{M} \tag{4}$$

where $N_j = \sum_{l=1}^{M} \delta_{j,l}$, and

$$\delta_{j,l} = \begin{cases} 1, & \text{if } Y_{t,l} = j, \\ 0, & \text{otherwise.} \end{cases}$$

namely, N_j is the number of cases (samples) in which $Y_t = j$.

$$P(Y_{t+1} = k|Y_t = j) = \begin{cases} 0, \text{ if } j\text{=0 and } k\text{=1}, \\ \dfrac{N_{jk}}{N_J}, \text{ otherwise} \end{cases} \tag{5}$$

where $N_{jk} = \sum_{l=1}^{M} \delta_{jk,l}$, and

$$\delta_{jk,l} = \begin{cases} 1, & \text{if } y_{t,l} = j \text{ and } y_{t+1,l} = k, \\ 0, & \text{otherwise.} \end{cases}$$

i.e. N_{jk} is the number of samples in which $Y_t = j$ and $Y_{t+1} = k$.

Fault detection, prognostic and diagnosis

As aforementioned, we can use $P(Y_t|X_1, ..., X_{n_t})$ to detect whether there is fault in step t. In order to do so, we first derive the formula of $P(Y_t|X_1, ..., X_{n_t})$ as follows:

$$P(Y_t|X_1, ..., X_{n_t}) = \frac{P(Y_t, X_1, ..., X_{n_t})}{P(X_1, ..., X_{n_t})} \propto P(Y_t, X_1, ..., X_{n_t})$$

$$\begin{aligned} P(X_1, ..., X_{n_t}) &= \sum_{Y_{t+1}} P(Y_T, Y_{t+1}, X_1, ..., X_{n_t}) \\ &= \sum_{Y_{t+1}} P(Y_t) P(Y_{t+1}|Y_t) \prod_{i=1}^{n_t} P(X - i|Y_t, Y_{t+1}) \end{aligned}$$

Here we get,

$$P(Y_t|X_1,...,X_{n_t}) \propto \sum_{Y_{t+1}} P(Y_t)P(Y_{t+1}|Y_t)\prod_{i=1}^{n_t} P(X_i|Y_t,Y_{t+1}) \tag{6}$$

where each probability term in equation 6 is the conditional probability distribution of each variable in the Bayesian network, and can be obtained from data using the method introduced in this Section . Within above introduction of how to derive $P(Y_t|X_1,...,X_{n_t})$, an important property of Bayesian network is adopted, i.e., the joint probability distribution of Bayesian network can be represented as the product of the conditional probability distributions of all variables in Bayesian network. We can use $P(Y_{t+1}|X_1,...,X_{n_t})$ to predict whether there will be fault in next step $t+1$, we then derive the formula of $P(Y_{t+1}|X_1,...,X_{n_t})$ as follows:

$$\begin{aligned} P(Y_{t+1}|X_1,...,X_{n_t}) =& \frac{P(Y_{t+1},X_1,...,X_{n_t})}{P(X_1,...,X_{n_t})} \\ & \propto P(Y_{t+1},X1,...,X_{n_t}) \end{aligned}$$

$$\begin{aligned} P(Y_{t+1},X_1,...,X_{n_t}) &= \sum_{Y_t} P(Y_{t+1},X_1,...,X_{n_t}) \\ &= \sum_{Y_t} P(Y_t)P(Y_{t+1}|Y_t)\prod_{i=1}^{n_t} P(X_i|Y_t,Y_{t+1}) \end{aligned}$$

So we obtained,

$$\begin{aligned} &P(Y_{t+1}|X_1,...,X_{n_t}) \propto \\ &\sum_{Y_t} P(Y_t)P(Y_{t+1}|Y_t)\prod_{i=1}^{n_t} P(X_i|Y_t,Y_{t+1}) \end{aligned} \tag{7}$$

where each probability term in equation 7 is the conditional probability distribution of each variable in the Bayesian network.

Additionally, we can also specify which sensor caused the defect by calculating the divergence of the sensor value form the normal mean value:

$$\arg \max_{i=1,...,n_t} |x_i - \mu_{i11}| \tag{8}$$

Discussion about model construction for multi steps

This section introduces the idea of predicting possible fault in the future manufacturing steps. The semiconductor fabrication process comprises m steps. Accordingly, we combine m Bayesian networks of each step and create a relatively large Bayesian network as shown in Figure 3.

To obtain such Bayesian network, we learn the small Bayesian network for each step separately, and then stack up the networks together. Such an approach is called a divide-and-conquer approach and

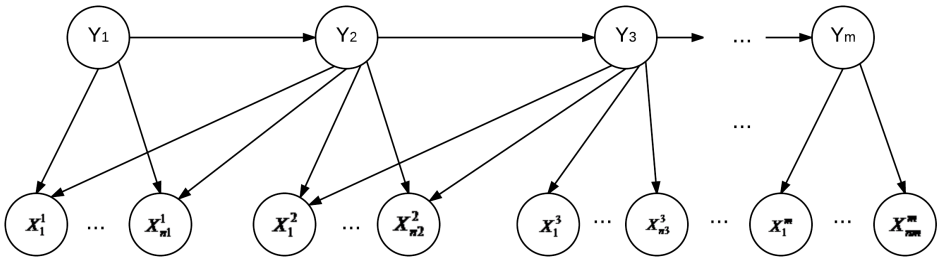


Figure 3: Proposed Bayesian network for all steps.

it realizes a cost-efficient procedure since if we learn an entire Bayesian network that represents the relationships among all the sensors and faults variables of all the steps from the whole data, the computational cost will be tremendously go high. Using the proposed stacked Bayesian network, we can make further prognostic. For instance, at step t, we can predict the possibility of having a defect at $t+2$, $t+3$ and so on. For example, we can predict whether there will be a fault in step 5 given the status values of step 1 by $P(Y_5|X_1^1, ..., X_{n_1}^1)$.

We can generalize the prognostic to the kth step $t+k(k>0)$ given the status of step t by the probability of $P(Y_{t+k}|X_1^t, ..., X_{n_t}^t)$. This probability can be calculated through Bayesian inference, however, it is not time efficient. Therefore, we present an approximation method as follows:

$$P(Y_{t+k}|X_1^t, ..., X_{n_t}^t) \propto \sum_{Y_t Y_{t+1} ... Y_{t+k-1}} P(Y_t) \prod_{i=t}^{t+k-1} P(Y_{i+1}|Y_i) \prod_{j=1}^{n_t} P(X_j|Y_t, Y_{t+1}) \tag{9}$$

Assuming the final product is unqualified, it can also diagnose in which step the fault is occurred, by computing $P(Y_t|Y_m = 0)(1 \leq t \leq m)$.

Case Study

We evaluate our prognostic model on a wafer fabrication dataset [Ols01]. In the dataset, six parameters have been identified by domain experts, the parameters are crucial for monitoring the state of the manufacturing process. Specifically, the parameters include: radio frequency forward power, radio frequency reflected power, chamber pressure, 405 nanometer (nm) emission, 520 nanometer (nm) emission, and direct current bias. Radio frequency forward power and radio frequency reflected power are used to measured the electrical power applied to the plasma, and the parameter chamber pressure calculates the pressure in the etch chamber, the 405 and 520 nanometer emission measure the intensity of two different wavelengths (i.e., colors) of light emitted by the plasma, and the direct current bias parameter calculates the direct current electrical potential difference in the tool.

Table 1: Prognostic accuracy using 10-fold cross validation

Predict Accuracy(%) / fold	step 1	step 2	step 3	step 4	step 5
1	94	54	100	100	100
2	100	100	100	97	56
3	35	92	64	97	93
4	100	100	79	100	100
5	100	100	79	95	100
6	100	85	36	61	100
7	100	100	93	97	100
8	100	92	86	97	100
9	100	92	86	100	100
10	100	92	100	82	70
average	92.9	90.8	82.1	92.6	91.9

Based on prior knowledge on the parameters of the dataset, we reshape and reconstruct the data such that we can apply our Bayesian net. Therefore, the data is a simulated version of the dataset. In the dataset, we divide the process of semiconductor fabrication into six steps; In each step, there are six parameters (sensors) monitoring the state of the manufacturing process for quality control. The dataset contains the value of six parameters and whether there is a fault in the current step and next step. To our knowledge, most existing methods are to detect a fault of current step in real time, few methods can predict for further step, so in this work we only test the performance of the proposed method. We use such data to train our Bayesian network while employing 10-fold cross validation in order to validate the accuracy of the proposed method. Consequently, the dataset of each step is divided into ten groups, at each time we determine 9 sets to be the training set and one set to be the test set. The status (values of the six parameters) in each step are used to predict the occurrence of a defect in the next step. Table 1 illustrates the prognostic accuracy at each time-step together with their average a precision at each step.

In Table 1, the columns denote the process step. For instance, values in step 1 column show the prognostic accuracy where given the sensory values in step 1, the Bayesian net provides a prognostic on occurrence of fault in step 2. The rows represent each test of the 10-fold cross validation together with the average accuracy for each step. Such results suggest a qualified precision is achieved by means of our stacked Bayesian network.

Summary

In order to effectively detect manufacturing faults as early as possible for the semiconductor industry,

a Bayesian network-based solution is proposed, in which a novel Bayesian network structure is designed, that contains both discrete and continuous variables, representing the relationships between the status of the current process and the probability of faults occurring in the current and / or next steps. The network was then trained in order to obtain the optimal parameters from the data. The experiment results illustrate the effectiveness of the proposed method. The prognostic model is accurate and functional when provided with adequate training data, which contains the value of sensors and information on whether the state is normal or abnormal.