Operation Point Detection for Impeller Pumps with Low-Cost Sensors by Using Neural Networks and Signal Processing

Vom Fachbereich Maschinenbau und Verfahrenstechnik

der Rheinland-Pfälzischen Technischen Universität Kaiserslautern-Landau

zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

genehmigte

Dissertation

von

Frau

Jun Feng, M.Sc.

aus Hubei, China

Tag der mündlichen Prüfung:	06.12.2024
Dekan:	Prof. Dr. rer. nat. Roland Ulber
Vorsitzender:	Jun.-Prof. Dr.-Ing. Leyu Lin
Berichterstatter:	Prof. Dr.-Ing. Martin Böhle
	Univ.-Prof. Dr.-Ing. habil. Uwe Janoske

D 386

Vorwort

Diese Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiterin am Gemeinsames Forschungszentrum für Strömungsmechanik und Strömungsmaschinen der Jianghan Universität (JHU) und der Rheinland-Pfälzischen Technischen Universität Kaiserslautern-Landau (RPTU).

Mein besonderer Dank gilt meinem Doktorvater Herrn Prof. Dr.-Ing. Martin Böhle für sein Vertrauen und seine wertvolle Unterstützung. Er hat mir uneingeschränkt geholfen, obwohl ich nur in China bleiben konnte, um meine Arbeit zu erledigen. Ebenso möchte ich mich bei Herrn Prof. Dr.-Ing. habil. Uwe Janoske bedanke, der das Koreferat übernommen und großes Interesse an dieser Arbeit gezeigt hat. Herrn Jun.-Prof. Dr.-Ing. Leyu Lin danke ich ferner für die Übernahme des Vorsitzes der Prüfungskommission.

Zudem möchte ich allen meinen Kollegen an der JHU und der RPTU für ihre wertvolle Unterstützung während meiner Promotionszeit herzlich danken. Ich möchte Herrn Prof. Yu und Frau Tang besonders danken, da sie mich bei meiner Bewerbung um eine Arbeitsmöglichkeit in Deutschland großartig unterstützt haben.

Ein weiterer Dank geht an meine ehemaligen Studenten der RPTU und der JHU, die mich während meiner Arbeit tatkräftig unterstützt haben. Besonders Wei Hong hat mir beim Aufbau des Prüfstands große Unterstützung geleistet.

Abschließend gilt mein herzlichster Dank meiner Mutter, meiner Tante, meiner Familie und meinen Freunden, die mich mit ihrem außergewöhnlichen Verständnis und ihrer liebevollen Unterstützung immer begleitet haben.

Abstract

The present dissertation investigates the applicability of a simple neural network (NN) based on vibration signals in detecting the operating points of centrifugal pumps. The effects of different NN topologies and hyperparameters were evaluated by utilizing the frequency- and time-domain signals. Their performances were compared under the same NN model. The simplest and most economical vibration sensors and evaluation boards were used in the experiments, and the effects of different sensor positions and numbers on the detection results were tested.

The results show that a single point measurement model for flow rate detection at a large interval achieved a NN test accuracy of 0.99 over 10 epochs, however, the aggregation of multiple sensor data did not improve the NN model performance. For simultaneous detection of rotation speed and flow rate based on frequency-domain signals, the single hidden-layer NN model achieved an accuracy of 0.90 in 116s training duration time. For time-domain signals, the simple NN model performance in flow rate detection was limited, achieving only 0.5 test accuracy. In addition, this study demonstrates the feasibility of low-cost hardware (<€60) in centrifugal pump operating point detection, proving that the simple NN model can provide accurate detection results under resource-constrained conditions.

A centrifugal pump test bench and its measurement and control systems including hardware and software, were developed and set up at Jianghan University (JHU) with the support from University of Kaiserslautern-Landau (RPTU) to facilitate these experiments.

Overall, this research confirms the feasibility of using simple NN classifiers in centrifugal pump operating point detection, especially based on frequency-domain signals. Future research should further optimize the NNs for time-domain signal training and develop a low-cost Raspberry Pi-based detection system to improve the detection performance and breadth of applications.

Kurzfassung

Die vorliegende Dissertation untersucht die Anwendbarkeit eines einfachen neuronalen Netzes (NN) zur Detektion von Betriebspunkten der Kreiselpumpen durch Schwingungssignalen. Die Detektionen verschiedener NN-Topologien und Hyperparameter wurden anhand von Signalen im Frequenz- und Zeitbereich bewertet. Die Ergebnisse wurden mit identischen NN-Modellen verglichen. In den Experimenten wurden die einfachsten und kostengünstigsten Schwingungssensoren und Auswerteplatinen verwendet. Die Auswirkungen unterschiedlicher Sensorpositionen und Sensoranzahlen auf die Detektionsergebnisse wurden getestet.

Die Ergebnisse zeigen, dass ein Einzel-Punkt-Messmodell für die Detektion der Förderstrom in einem großen Intervall eine NN-Testgenauigkeit von 0.99 über 10 Epochen erreicht, die Aggregation mehrerer Sensordaten die Leistung des NN-Modells nicht verbessert. Für die gleichzeitige Detektion von Drehzahl und Förderstrom durch Signale im Frequenzbereich erreichte das NN-Modell mit einer einzigen Hidden Layer eine Genauigkeit von 0.90 bei einer Trainingsdauer von 116 Sekunden. Für Signale im Zeitbereich war die Leistung des einfachen NN-Modells bei der Detektion der Förderstrom begrenzt und es erreichte nur eine Testgenauigkeit von 0.5. Darüber hinaus demonstriert diese Studie die Machbarkeit von kostengünstiger Hardware für die Detektion der Betriebspunkten von Kreiselpumpen und beweist, dass das einfache NN-Modell genaue Detektion liefern kann.

Um diese Experimente zu ermöglichen, wurden an der Jianghan Universität (JHU) mit Unterstützung der Rheinland-Pfälzische Technischen Universität Kaiserslautern-Landau (RPTU) ein Kreiselpumpenprüfstand und die zugehörigen Mess- und Steuersysteme, einschließlich Hard- und Software, aufgebaut und entwickelt.

Insgesamt bestätigen diese Untersuchungen die Machbarkeit des Einsatzes einfacher NN-Klassifikatoren zur Detektion von Betriebspunkten der Kreiselpumpen bei Schwingungssignalen. Zukünftige Forschungsarbeiten sollten die NN für das Training von Signalen im Zeitbereich weiter optimieren und ein kostengünstiges, auf einem Raspberry Pi basierendes Detektionssystem entwickeln, um die Detektionsleistung und die Anwendungsbreite zu verbessern.

Contents

Abstract ..... **I**

Kurzfassung ..... **II**

Nomenclature ..... **VI**

1 Introduction **1**

1.1 Overview 1

1.2 Current Research 2

1.2.1 Pump Vibration Signal Analysis 2

1.2.2 Application of Artificial Intelligent in Fluid Machinery 4

1.3 Research Purpose 6

2 Theoretical Background **8**

2.1 Operating and Vibration Behavior of Centrifugal Pumps 8

2.1.1 Operating Behavior of Centrifugal Pumps 9

2.1.2 Vibration of Centrifugal Pumps 12

2.2 Signal Processing 13

2.2.1 Signal Classification 14

2.2.2 Analog-to-Digital Conversion 15

2.2.3 Time-Domain Analysis 17

2.2.4 Frequency-Domain Analysis 21

2.3 Classification 24

2.3.1 Neural Networks 25

2.3.2 Multi-Class Classification 41

2.3.3 Neural Network Classifier 44

2.4 Programming Languages 46

3 Design of the Experimental **50**

3.1 Design of the Experimental Setup 50

3.1.1 Test Bench 50
3.1.2 Control and Measuring Boxes 53
3.1.3 ADXL345 Accelerometer 60
3.2 Software Development 63
3.2.1 Test Bench Control and Measurement Interface (MATLAB) 64
3.2.2 Test Bench Control and Measurement Interface (Python) 66
3.3 Description of the Test Process 67
3.3.1 Acquisition of Vibration Signals 67
3.3.2 Data Sets Processing 69
3.3.3 Setting up Different Neural Networks 72
4 Results 74
4.1 Process of Training and Evaluation 74
4.1.1 The Training and Evaluation Program 74
4.1.2 Nomenclature of the Test Series and Evaluation 74
4.1.3 Results Presentation and Evaluation 76
4.2 Experiments Utilizing Frequency-domain Signals 79
4.2.1 Single Measurement Point Capability Investigation 79
4.2.2 Minimum Detectable Flow Rate Difference Recognize 83
4.2.3 The Impact of Dataset Settings on Training Results 86
4.2.4 The Impact of Hidden Layer on Neural Network Accuracy 89
4.2.5 The Impact of Hyperparameters on Neural Network Training Results 92
4.2.6 Classification of normal and non-normal operating points 97
4.2.7 Rotation Speed Detection 99
4.2.8 Flow Rate Detection 101
4.2.9 Rotation Speed and Flow Rate Detection 105
4.3 Experiments Utilizing Time-domain Signals 108

4.3.1 Classification of Normal and Non-normal Operating Points 108

4.3.2 Rotation Speed and Flow Rate Detection 115

5 Discussion 123

5.1 Impact of the Neural Network Models 123

5.2 Classification of normal and non-normal operating points 124

5.3 Rotation Speed and Flow Rate Detection 125

6 Summary 128

6.1 Research 128

6.2 Conclusion 129

6.3 Outlook 131

Bibliography 132

Appendixes 137

List of Figures 143

List of Tables 147

Publications and Conferences 149

List of Supervised Student Thesis 150

Curriculum Vitae 151

Nomenclature

Symbols		
H	Head	m
P (2.1)	Power	W
η	Efficiency	-
Q	Volume flow rate	m^3/h
n	Rotation speed	rpm
Q_{opt}	Optimal flow rate	m^3/h
ρ	Fluid density	kg/m^3
g	Gravitational acceleration	m/s^2
P	Pressure	
ΔP_{static}	Static pressure increase	N/m^2
A	Cross-sectional area	m^2
h	Height	m
ω	Angular velocity	rad/s
M (2.1)	Torque	N·m
n_p	Rotation speed of the pump	rpm
a (2.1)	Vibration acceleration	m/s^2
L (2.1)	Vibration displacement	m
f	Frequency	Hz
v	Vibration speed	m/s
T_s	Sampling period	s
f_s	Sampling frequency	Hz
f_N	Nyquist frequency	Hz

r_n	Final residue	-
Ω	Analog frequency	rad/s
ω_s	Normalized digital frequency	rad/sample
N	Length of a signal	-
W_N	Twiddle factor	-
w	Weight	-
y	Output (true value)	-
F	Activation function	-
σ	Sigmoid function	-
L (2.3)	Loss function	-
$\hat{y}$	Output (predicted value)	-
x	Input	-
b	Bias	-
η (2.3)	Learning rate	-
a (2.3)	Activation output	-
δ	Gradient of the loss	-
θ	Parameters of a neuro network	-
G	Accumulated sum of squared gradients	-
g	Gradient	-
ϵ	Constant to avoid division by zero	-
γ	Decay rate	-
P (2.3)	Probability	-
Nq	Specific speed of the pump	-
C	Total number of the classes	-
$\mathrm{N_{dh}}, \mathrm{c}$	number of the direct hits test sets of the c -th class	-

$N_{\text{test sets}, c}$	number of the test sets of this class	-
K	Number of the classes	-
M	Number of rows of the datasets	-
N	Length of the feature signal	-

Indices

s	Suction side of the pump
p	Pressure side of the pump
hyd	Hydrodynamic
me	Mechanical
k	Number of the spectral line under consideration
t	time step

Acronyms

NN	Neural network
JHU	Jianghan University
RPTU	University of Kaiserslautern-Landau
AI	Artificial intelligence
EFD	Early Failure Detection
FFT	Fast Fourier Transform
CR	Congruence rate
PID	Proportional Integral Derivative
CNN	Convolutional neural network
PCA	Principal component analysis
LSTM	Long Short-Term Memory

SAM	Institute of Fluid Mechanics and Fluid Machinery
ADC	Analog-to-digital conversion
RMS	Mean and Root Mean Square
PtP	Peak-to-Peak
AR	Autoregression
MA	Moving average
EMD	Empirical Mode Decomposition
IMF	Intrinsic Mode Functions
DTFT	Discrete-time Fourier Transform
DSP	Digital Signal Processing
DFT	Discrete Fourier Transform
SVM	Support Vector Machines
k-NN	k-Nearest Neighbors
DNN	Deep neural network
NLP	Natural language processing
RNN	Recurrent Neural Networks
ReLU	Rectified Linear Unit
Tanh	Hyperbolic Tangent
MSE	Mean Squared Error
BCE	Binary Cross-Entropy Loss
CCE	Categorical Cross-Entropy Loss
MAE	Mean Absolute Error
SGD	Stochastic Gradient Descent
Adagrad	Adaptive Gradient
RMSProp	Root Mean Square Propagation

Adam	Adaptive Moment Estimation
P	Perceptron model
FF	Feed forward model
RBF	Radial basis network
HN	Hopfield network
BM	Boltzmann machine neural network
DFF	deep feed forward model
OvA	One-vs-All
OvR	One-vs-Rest
IO	Input/output
PWM	Pulse Width Modulation
ICSP	In-Circuit Serial Programming
SSR	Solid State Relays
GPIO	General-Purpose Input/Output
RTD	resistance temperature detector
IC	Integrated circuit
UI	User interface
t-SNE	t-Distributed Stochastic Neighbor Embedding

1 Introduction

1.1 Overview

With the rapid development of modern society, pumps, as general mechanical equipment, have found applications in various fields. The pump is the mechanical equipment that must be used for water supply and drainage systems, and beyond water pumps, there are condensate pumps, oil and gas mixing pumps, circulating pumps, and ash pumps. These pumps handle different substances, some of which are hazardous chemicals that can pose risks to human health or mechanical equipment.

In order to ensure the optimal operation of the pump, it is generally required to operate at the rated operating point. Typically, expensive and complex sensors are required to monitor this operating condition. To accurately detect the condition of a pump loop system requires monitoring key parameters such as pressure and flow rates within the pump loop. However, this method leads to several challenges: (1) High investment costs; (2) Difficulty in measurement—the pump loop needs to be modified, or measurement devices must be pre-installed; and (3) Handling hazardous chemicals—there are special requirements for the materials used in flowmeters and pressure sensors.

With the rapid advancement of artificial intelligence (AI), signal analysis, and computer technologies, the state analysis technology of pump loop systems has made considerable progress [1] [2]. In recent years, researchers have primarily focused on Early Failure Detection (EFD) of pumps using vibration analysis, and have proposed various feasible detection methods, such as spectrum analysis, power spectrum estimation, wavelet analysis, fuzzy fault diagnosis, and support vector machines. These developments suggest that detecting the operating point of a pump loop through vibration signal analysis is also a viable approach.

AI technology has been used increasingly applied in mechanical engineering in recent years. Since the term “artificial intelligence” first coined by John McCarthy in 1956, nearly 70 years have passed. Machine learning has played a vital role in the rapid development of AI over the last 20 years. With the reinvigoration of NN in the 2000s, deep learning has become an extremely active area of research that is paving the way for modern machine learning [1].

In this dissertation, the applicability of a simple NN for detecting operating conditions based on vibration signals from a centrifugal pump was investigated. The evaluation involved analyzing and comparing both time-domain and frequency-domain signals to assess their distinct characteristics under the same simple NN model. Additionally, various NN topologies and hyperparameters were explored to optimize performance. The study also utilized the simplest and most cost-effective vibration sensors, examining their effectiveness at different positions within the pump loop system and assessing the impact of varying the number of sensors on detection outcomes.

1.2 Current Research

1.2.1 Pump Vibration Signal Analysis

Vibration analysis for operating condition detection of machines is widely used in various fields. In pump vibration monitoring, the most common application is EFD. By monitoring the operating condition of the pump, early detection and forecasting of potential faults can ensure the safe and reliable operation of the pump, thereby yielding significant economic and social benefits. EFD research based on vibration signals traces back to 1980s [3]. In the 1980s, vibration monitoring was used for failure diagnosis [4]. In 2013, Henriquez [5] examined vibration-based detection in centrifugal pumps. But in addition to the vibration signal, the power of the pump, pressures and temperatures, and flow rates are used. Based on the composite signal, the system differentiates between normal operation, cavitation, gas mixed delivery, wear, unbalance, dry operation and search blockage, with the aid of pressure, temperature, power and flow data of the test pump. The decision tree method ID3 [6] was used as a classifier. The principle of this method is to create a tree structure, the leaves of the tree contain different attributes, and the branches of the tree correspond to associated values. The outermost leaves of such a decision tree represent each category, i.e., the operating state of the recorded operating point of the pump. Through the work of Kafka, the feasibility of determining the operating condition of a pump from the vibration signals has been demonstrated.

Licht mentioned in his dissertation [7], that based on Kafka's research, in the following ten years, this method was further studied. Huhn [8] increased the trefferquote to 0.85 by improving the decision method. Kollmar [9] used fewer sensors to get the vibration signals and his experiment proving that, the quality of the data and the characteristics of

the sensors have an effect on the results. The focus of Nuglisch's [10] research is to determine optimal sensor locations and demonstrate device independence. He used signals from vibration sensors, acceleration sensors and point-of-load sensors. With a simple calibration, the influences of different pump systems can be eliminated. By recording only one operating point of the test pump in normal operation and comparing the signal with the training data of the existing EFD system, the EFD can be achieved by different pump systems. But this method is for different rotation speeds not applicable. Kiggen [11] used Hall sensors to demonstrate another method of determining the pump's operating point. But Hall sensors collect the current signal instead of the vibration signal.

The studies mentioned above are all based on decision tree methods. Recent research on EFD in centrifugal pumps has also emerged. These studies, along with further investigations, form the foundation for developing advanced pump monitoring systems.

Almost all major pump manufacturers have their own pump monitoring systems. For example, KSB offers the "Pump Expert" system, which records all connected sensor data, including temperatures, pressures, flow rates, and performance. In addition, vibrations and speed are typically monitored and analyzed. Similarly, there are many smaller companies in China specializing in pump monitoring systems. Their systems also record all operating data and vibrations. However, the collected vibration signals in these systems are not used to determine the operating point but rather for EFD.

The only known system that uses vibration to detect operating points is what KSB calls the "Sonolyzer", introduced in 2015 [7]. This smart phone application can detect three operating states of a pump through the recorded pump motor noise, namely, partial state, rated state or overload state. Additionally, there are companies that offer condition monitoring and fault diagnosis software for pump systems under specific conditions. Nevertheless, these monitoring and diagnostic tools or methods are not universally applicable and expensive.

Moreover, in the method described above, the vibration signals are all preprocessed, including filtering and denoising. Specifically, time-domain signals are often converted into frequency-domain signals using Fourier transform before further processing. Licht [7] used time-domain signals directly in his early fault detection system for gas pressure regulators, extracting eigenvalues without converting to the frequency domain. The accuracy was consistently above 0.8, demonstrating the feasibility of determining the

operating status of the pump using a simple device. Feng [12] explored the feasibility of applying a NN for operating point detection of an impeller pump using vibration signals. In this study, time-domain vibration signals were converted into the frequency domain via Fast Fourier Transform (FFT) before being classified by a sorting algorithm and used as training or test data sets for the NN. Vibration signals from 8 operating conditions were measured in this work, and the congruence rate (CR) of over 0.9 confirmed the applicability of the simple NN method.

1.2.2 Application of Artificial Intelligent in Fluid Machinery

AI is a discipline that studies how to design intelligent machines or systems to simulate, enhance, and extend human intelligence [13]. As a key area of advancement in modern technology, AI represents an important direction for new technologies, products, and industries [14]. Its development is marked by various strategic advancements and significant trends [15]. AI's advantages in addressing challenges such as remote control, fault diagnosis, and nonlinearity offer valuable guidance for the advancement of mechanical systems [16]. In essence, AI focuses on creating intelligent machines or systems that can mimic, enhance, and expand upon human cognitive capabilities [17].

AI has seen a profound impact on various aspects of mechanical systems. For instance, in mechanical design, AI contributes significantly to model synthesis and analysis, which involves extensive numerical calculations and the formulation of optimal plans [18]. The quest for finding the most effective solutions is a critical area of AI research. In machine manufacturing, AI systems facilitate the control of all assembly machines within a factory. These systems can set objectives based on real-time assessments or external inputs, employing "planning" techniques to achieve these goals. This process, integral to automatic problem-solving, highlights AI's capability in optimizing manufacturing operations [19]. Furthermore, in mechatronic engineering, the complexity of internal structures often poses challenges for process control and EFD. While traditional Proportional Integral Derivative (PID) controllers address some issues, they may not suffice in more complex scenarios. Here, advanced AI algorithms offer more inclusive solutions [20]. In the realm of mechanical system fault diagnosis, AI plays a crucial role. Sensors collect data, which is then analyzed to determine whether various components are operating normally. This process involves monitoring system performance and

diagnosing abnormal conditions based on the extracted features from vibration and other signals [21].

AI has become increasingly significant in various applications within mechanical systems. One prominent application is the use of expert systems, which are designed to address complex problems and provide decision-making capabilities similar to those of a human expert. The concept of expert systems, a key element of AI, was first successfully implemented in 1970 [22]. This foundational technology was further advanced in 2021 when Wei [23] integrated an expert system with a PID algorithm for oil pump mechanical devices. This integration notably improved operational efficiency by addressing issues such as high oil failure rates, significant fluctuations in external pressure, low levels of automation, and high labor intensity, thereby transforming outdated practices. In addition to expert systems, NNs have also seen extensive application in AI. NNs are modeled after the biological NNs in the human brain, consisting of interconnected nodes arranged in multiple layers. These networks, designed and constructed by humans, emulate cognitive processes and are capable of processing and learning from data. The use of NNs in computer science and machine learning reflects their ability to replicate human-like learning processes [20].

These advancements highlight the evolving capabilities of AI in enhancing mechanical systems, from improving decision-making processes to simulating complex cognitive functions.

In the last decade, several prominent deep learning frameworks have been introduced, including TensorFlow, PyTorch, and Caffe. TensorFlow, developed by Google in 2015, is built on C++ and is versatile, finding applications across nearly every field. PyTorch, developed by the Torch7 team and released in 2017, is based on Python and has gained widespread adoption due to its dynamic computational graph capabilities. Caffe, developed in 2013 by Jia Yangqing at the University of California, is specifically designed to support convolutional neural networks (CNNs) [24].

These advanced deep learning frameworks have demonstrated their effectiveness in various applications, including those related to pumps, fans, compressors, and turbines. For instance, Wang [25] used of Long Short-Term Memory (LSTM) networks based on TensorFlow to achieve the distinction between normal and non-normal operating conditions, to ensure the safe operation of nuclear power plants. Similarly, Khan [26]

employed a hybrid approach combining principal component analysis (PCA) and deep learning within the TensorFlow framework to identify hidden patterns in wind data and accurately forecast wind power for large-scale wind turbines.

However, it is worth noting that these successful implementations often rely on complex NNs that incorporate various layers such as conventional, recurrent, or LSTM layers. These networks typically require prolonged training periods and intricate hyperparameters tuning, demanding high computational performance and a controlled experimental environment.

1.3 Research Purpose

This dissertation investigates the applicability of a simple NN classifier for detecting the operating points of centrifugal pumps using vibration signals. A NN model selection program, developed in Python with TensorFlow framework, aims to identify the most effective simple NN model. The NN topologies includes an input layer, up to two hidden layers, and one or two output layers. The selection program allows customization of the NN topology and hyperparameters based on experimental needs. The training results and processes for each NN model are recorded in a text document. The program then selects the optimal NN model based on performance metrics. By varying the NN topology and hyperparameters, the research examines how different NN configurations affect the detection performance of centrifugal pump operating points. The evaluation criteria include accuracies, confusion matrix, and training duration. Furthermore, the testing phase utilizes two types of vibration signals: time-domain signals and frequency-domain signals. This approach ensures a comprehensive analysis of the NN's effectiveness in operating points detection.

To obtain the training and test data for the NN, a pump loop system was constructed. The vibration signal from different operating points of the pump were recorded using simple acceleration sensors, ADXL345 (Analog Devices), which were adhered to the pump casing. In traditional vibration signal analysis, frequency-domain signals are commonly used because they directly reflect the vibration characteristics in the frequency spectrum. Therefore, the time-domain signals collected in this study were first converted into frequency-domain signals using FFT for NN model training. However, since time-domain signals theoretically encompass all the necessary information, this dissertation also

explores the direct use of time-domain signals. This approach compares the performance of NN classifiers using both time-domain and frequency-domain signals.

In this dissertation, the effects of NN topology and hyperparameters on classifier performance are first investigated using frequency-domain signals. Subsequently, the potential of NN classifiers for operating point detection is explored through a comparison of frequency-domain signals and time-domain signals. Specifically, the NN models were compared across scenarios: detection of normal versus non-normal operating conditions, rotation speed and flow rate detection under normal conditions. Through these analyses, the performance differences of various signal processing methods for pump condition detection are assessed, and the feasibility of using vibration signals for determining operating points in a centrifugal pump is explored.

The experimental results show that:

1. The maximum accuracies of a NN classifier by utilizing frequency-domain signals are significantly higher than that by utilizing time-domain signals.
2. The topology structure and hyperparameters of a NN exert distinct influences on its performance, which are critical factors to consider when optimizing a model for a specific task.
3. Simple NN classifiers are feasible for the detection of normal and non-normal condition of a centrifugal pump, regardless of whether frequency-domain or time-domain signals are used. For rotation speed detection, the accuracy of the simple NN classifiers using both signals can also achieve higher than 0.8. For flow rate detection, the accuracy of the NN classifier reaches 0.85 when frequency-domain signals are used. However, when time-domain signals are employed, the simple NN is unable to effectively discriminate between different flow rates, with a maximum accuracy of only 0.5.

A detailed discussion of the results is provided in Chapter 5.

2 Theoretical Background

In this dissertation, the applicability of a simple NN classifier for operating point detection by vibration signals of a centrifugal pump was investigated. This chapter describes the essential theoretical background of this work. The operating characteristics of the centrifugal pump will be first studied. What follows is a summary of the different signal analysis methods. The chapter concludes with an introduction to NNs, and the programming language used for their construction.

2.1 Operating and Vibration Behavior of Centrifugal Pumps

The investigations of this research were carried out with the help of a test pump. It was a single-stage centrifugal pump with a replaceable impeller, which is designed and manufactured by the Institute of Fluid Mechanics and Fluid Machinery (SAM) at RPTU. The exact designation can be found in the chapter describing the experiment (see Section 3.1). Figure 2.1 shows a sectional view of such a centrifugal pump.

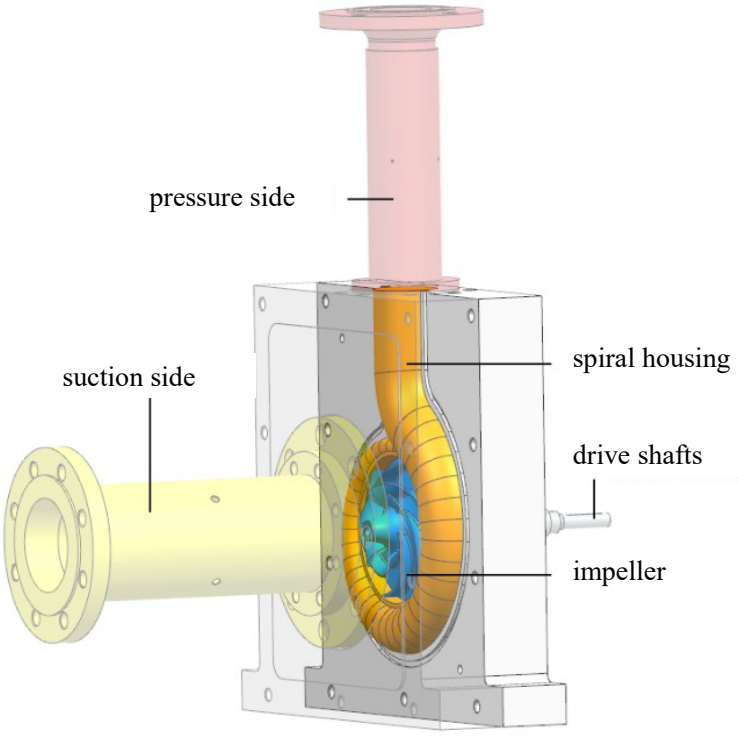


Figure 2.1: Representation of components of an impeller pump

The impeller is the core component of a centrifugal pump, which transfers energy to the liquid through rotation. The pump casing is a structure that surrounds the impeller and

guides the flow of liquid, usually in a spiral shape. The suction port is the channel where the liquid enters the pump. The discharge port is the channel where the liquid is discharged from the pump. The shaft and bearings support the impeller and enable it to rotate

2.1.1 Operating Behavior of Centrifugal Pumps

The centrifugal impeller of the pump is driven by the shaft. The medium enters through the liquid inlet pipe (suction side) into the pump and is accelerated radially by the impeller. The medium is collected in the volute and passed to the outlet pipe (pressure side). At the same time, the medium is decelerated, the kinetic energy of the motion is converted into potential pressure energy, i.e. into static pressure. The operating characteristics of the pump are described by the pump characteristic curves. By the characteristic curves, the head H, the electric drive power P and the efficiency η are generally described relative to the volume flow rate Q delivered at a constant rotation speed n. These curves are Head-Flow Rate Curve (H-Q), Efficiency-Flow Rate Curve (η-Q) and Power-Flow Rate Curve (P-Q).

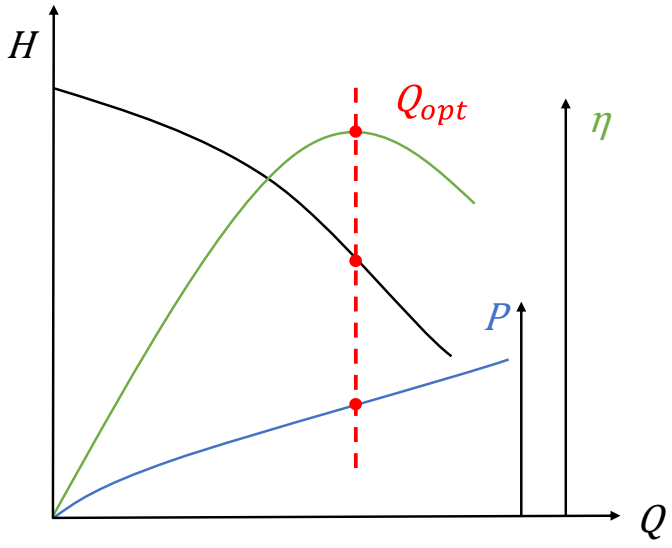


Figure 2.2: Three characteristic curves of a pump

Figure 2.2 shows an example of three characteristic curves for a pump. The red dots indicate the optimal flow rate Q_{opt}.

Head

The head of a pump is a measure of the mechanical energy transferred to the pumped medium and is calculated as follows.

$$H = \frac{\Delta P_{static}}{\rho g} + \frac{1}{2g}\left(\left(\frac{Q}{A_p}\right)^2 - \left(\frac{Q}{A_s}\right)^2\right) + h_p - h_s \quad (2.1)$$

Heat loss is ignored in this formula. The indices s and p stand for the suction and pressure side of the pump. The head H can be detected by the fluid density ρ, gravitational acceleration g, the static pressure increase ΔP_{static} between suction and pressure side, flow volume in the pump loop Q, pipe cross-sectional area of the discharge side A_p, pipe cross-sectional area of the suction side A_s and the height difference between the pressure sensors on the suction and pressure side $(h_p - h_s)$.

Power

The hydrodynamic power P_{hyd}, the power transferred to the medium, is produced by the impeller volume flow Q and the calculated head H.

$$P_{hyd} = \rho g Q H \quad (2.2)$$

Due to engine losses from the clutch and motor, another power is usually defined. This is the mechanical power P_{me} fed directly to the pump from the motor.

$$P_{me} = \omega \cdot M = 2\pi n \cdot M_{motor} \quad (2.3)$$

where ω presents the angular velocity of the motor, M is the torque produced by the motor and M_{motor} the motor torque.

Efficiency

The efficiency η of a pump is the division between the usable hydrodynamic power and the mechanical power transmitted by the clutch from the motor.

$$\eta = \frac{P_{hyd}}{P_{me}} \quad (2.4)$$

Operating Point

The respective characteristic curves apply to the invariable rotation speed n_p of the pump. Figure 2.3 shows the head-flow rate curve under the influence of a rotation speed variation. Within the range independent of Reynolds number, characteristic curves of different

rotation speeds can be transformed into each other. The necessary equation is called the affinity law [27].

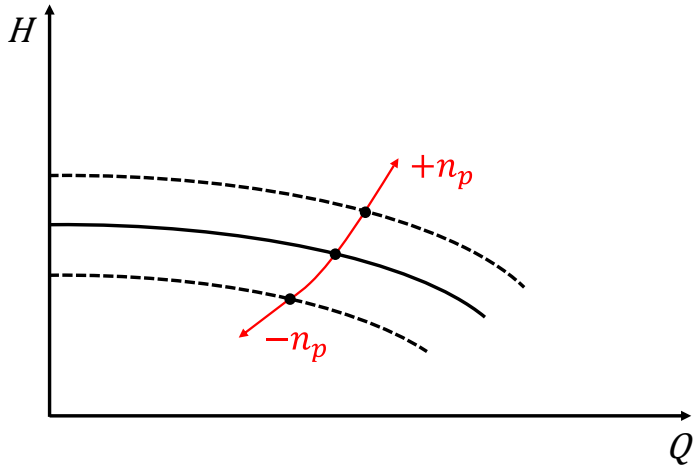


Figure 2.3: The influence of a rotation speed variation on the H - Q curve

Every operating point at which a pump is operated depends not only on the pump but also on the connected system. The pressure loss effect of all components results in a specific system characteristic curve for each system. The head of the system is plotted against the delivery flow. Since the flow velocity c is quadratically proportional to the pressure loss, the head therefore increases quadratically with the flow (See Figure 2.4 O1).

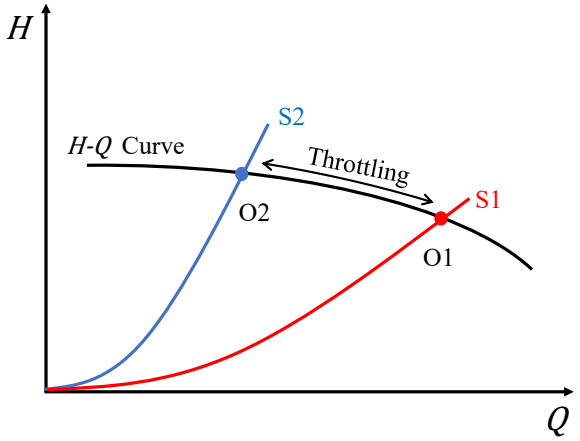


Figure 2.4: The system characteristic curve due to valve control

The intersection point of the system characteristic curve with the pump characteristic curve represents the operating point of the pump. To change the operating point, the pump characteristic curve or the system characteristic curve must therefore be varied.

During the investigations in this work, the system characteristic curve was affected by valve control or motor rotation speed. By throttling the flow rate, the system resistance is

increased, and the characteristic curve is thus made steeper (see Figure 2.4 O2). This is the most used control method for pumps, although the entire excess energy or delivery head is dissipated at the regulating valve. Operating point control through variation in rotation speed is less commonly used due to the high cost associated with frequency converters required to adjust motor speed. However, in this dissertation, the experimental setup includes a frequency converter, which enables the application of rotational speed control to modify the operating point.

2.1.2 Vibration of Centrifugal Pumps

All matter-solid, liquid and gaseous-is capable of vibration, e.g. vibration in liquids is almost always longitudinal and can cause large forces because of the low compressibility of liquids, e.g. pipes conveying water can be subjected to high inertia forces when a valve or tap is suddenly closed [28].

Any machine has many moving parts, each of which is a potential source of vibration. The causes of centrifugal pump vibration can be divided into two categories, hydraulic and mechanical vibration [27]. Superimposed on these main components is also the vibration excitation from the environment, which must be regarded as a disturbance signal in the traditional analysis method. Vibration acceleration is usually obtained by measurement, using an accelerometer. Theoretically, the relationship between vibration acceleration a (m/s^2) and vibration displacement L (m) and frequency f (Hz) can be expressed by the following equation [29]:

$$a = (2\pi f)^2 L \quad (2.5)$$

Vibration standards for pumps are usually given by bodies such as the International Organization for Standardization (ISO) and the American Society of Mechanical Engineers (ASME). ISO 10816-3 [30] provides standards for evaluating the vibration of rotating machinery, for medium-sized machines (rated between 15kW and 300kW), the vibration speed in operation should normally be between 1.8 mm/s (good) and 4.5 mm/s (alarm). Estimated the vertical vibration acceleration of an impeller pump at rotation speed n = 2000 rpm, vibration speed v = 3 mm/s = 0.003 m/s:

$$f = \frac{n}{60} = \frac{2000rpm}{60} = 33.\bar{3}Hz \quad (2.6)$$

$$L = \frac{v}{2\pi f} = \frac{0.003m/s}{2\pi \times 33.\bar{3}Hz} \approx 1.43 \times 10^{-5}m \quad (2.7)$$

Substituting x into Equation (2.5)

$$a \approx (2\pi \times 33.\bar{3}Hz)^2 \cdot 1.43 \cdot 10^{-5}m = 0.627m/s^2 \quad (2.8)$$

However, actual values may vary depending on the specific pump condition, installation conditions, loads, etc., but this order of magnitude is not far off when selecting an accelerometer.

Regarding the detection of the operating point of a centrifugal pump, it is crucial to analyze this type of signal using a suitable method. In order not to lose information at different flow rates, there are some methods that reduce the signal but still don't remove any information, a holistic analytical approach is favored. The analysis method based on NN used in this dissertation, reduced the amount of calculation through convolutional or other type of NNs. The method does not do any reduction processing on the vibration signal, the signal contains all the information, including the environmental vibration component of the pump system under actual conditions.

2.2 Signal Processing

The vibration signal of centrifugal pumps is in this work to be processed. The first step to process the vibration signal is through an accelerometer to implement the analog-to-digital conversion (ADC). The accelerations of the investigated pump are basically analog signals, i.e., continuous value and time. However, after recording by the simple accelerometer (see Section 3.1.3), these signals are available in digital form, i.e., in time discrete. Only in this way is vibration signal analysis on a computer technically feasible. To minimize the loss of information caused by the accelerometer, the largest possible sampling rate is selected. Apart from the inherent filtering process of the accelerometer, there is none filtering processing on the extracted digital signals. So that the impact of working environment on ADC can also be retained in the generated data.

To be able to perform a classification, the signal characteristics must be generated so that the extracted data become significantly smaller and at the same time contain all the important information. This means that in this work, the signal components can provide information about the flow and rotation speed of the pump. During the evaluation of this

work, two different analysis methods were successfully applied. One approach is to extract characteristics from time-domain signals directly. With FFT, another deterministic approach was pursued. The FFT identifies the frequency content in a signal, which is one of the most well-used methods in all frequency-domain signal analysis. In the following, these two analysis methods are examined in more detail. However, the basic characteristics of the vibration signals are shown first.

2.2.1 Signal Classification

Signals can be classified in terms of the continuity of the independent and dependent variables as follows [31]:

Analog Signal: The independent and dependent variables that define the signal are continuous in time and amplitude. This means that the signal has a specified amplitude value at each specified time. Analog signals cannot be directly processed by a computer.

Continuous time signal: The time variable is continuous within the range defined by the signal. If the signal variable is represented by x and the time variable is t, then the signal is denoted as $x(t)$.

Discrete time signal: The time variable is discrete within the range defined by the signal. If the signal variable is x and the time variable was sampled at time n, where $n = n'T$, then the signal is represented as $x(n)$. A discrete time signal is also called a sampled signal because it is obtained by directly sampling the target signal. It is to be noted that the amplitude of the sampled signal can take any value within the specified amplitude range. Therefore, we say that the amplitude of a discrete time signal is continuous.

Digital signal: This is a time and amplitude discrete signal. It is represented as a time discrete signal and can be processed by computers.

Deterministic Signals： Deterministic signals are those signals whose values are precisely known and can be described by a mathematical function. The future values of these signals can be accurately predicted given their past values [32].

Stochastic Signals: Stochastic signals, also known as random signals, are those signals whose values cannot be precisely predicted. These signals are described by probabilistic models rather than exact functions.

The vibration of centrifugal pumps is a continuous-time analog signal. After sampling through an accelerometer, it is converted into a discrete-time digital signal, making the analysis of the vibration signal on a computer technically feasible. The digital vibration signal is composed of both deterministic and stochastic signal components. Deterministic parts of the vibration signal are predictable and can be completely characterized. The analysis assumes that the underlying signal is composed of many periodic vibrations of different frequencies and periods and can therefore be described completely analytically. Stochastic signals exhibit randomness or unpredictability. Unlike deterministic signals, which can be precisely described by mathematical equations or functions, an analytical description of such signals can only ever be partially successful. As already described, different flow-induced vibration signals primarily cause stochastic signal components and broadband excitations. Moreover, as the operating environment of the pump changes, the stochastic signal characteristics will also change accordingly.

2.2.2 Analog-to-Digital Conversion

The vibration of centrifugal pumps is analog signal. In order to process the signal on a computer, an accelerometer tuned to the signal's characteristics must be used to convert the analog signal into digital signal. The common output of an accelerometer is electrical signal, voltage or current, which is proportional to the signal being measured.

ADC occurs in two steps. The first step is sampling. Figure 2.5 (a) represents an analog signal $x(t)$ that is to be sampled. At each sampling point, the analog signal is sampled, and the value of the signal is held steady until the next sampling point. This process is called sample-and-hold. The sample-and-hold signal for the original signal is showed in Figure 2.5 (b) in green. The vertical lines mark the sampling point which mathematically is represented by Equation (2.9)

$$s(t) = \sum_{-\infty}^{\infty} \delta(t - nT_s) \quad (2.9)$$

where $\delta(t)$ represents the Dirac delta function, T_s is the sampling period and nT_s the sampling points. The sampled signal (Figure 2.5) is given by $y(t)$ and represents as follows:

$$y(t) = x(t) * s(t) = x(t) * \sum_{-\infty}^{\infty} \delta(t - nT_s) = \sum_{-\infty}^{\infty} x\left(n \sum_{-\infty}^{\infty} \delta(t - nT_s)\right) \delta(t - nT_s) \tag{2.10}$$

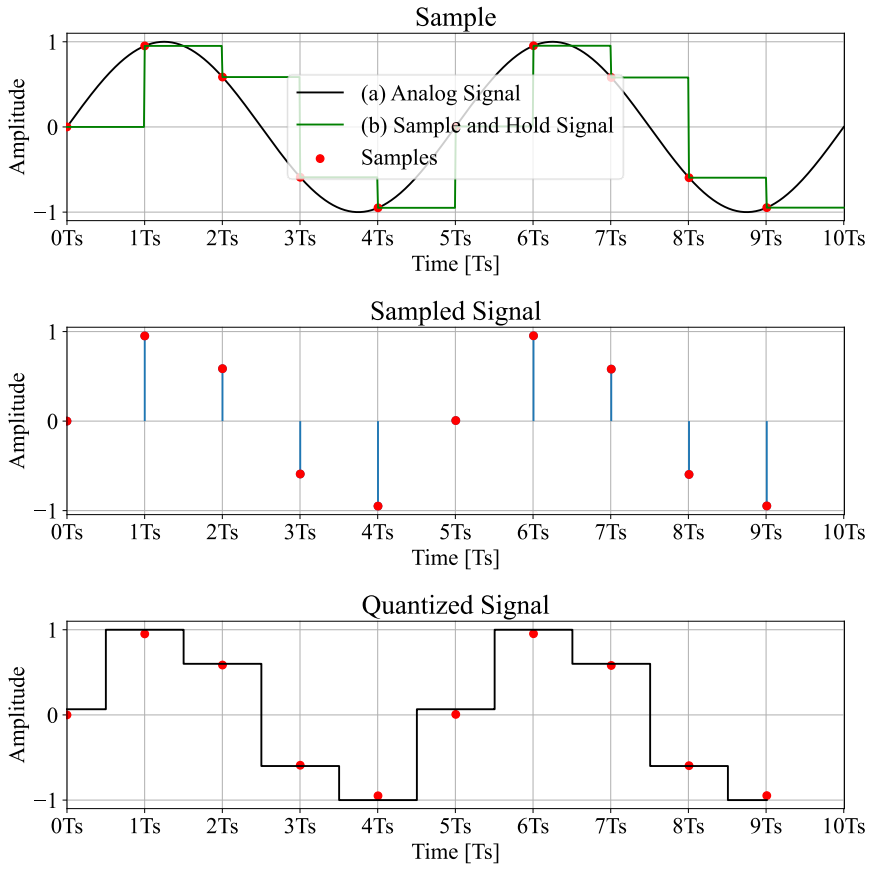


Figure 2.5: Process of an analog-to-digital conversion

The sampling period T_s is the time in seconds between samples. The sampling frequency f_s is the number of samples per second, measured in Hz [33]. Therefore,

$$f_s = \frac{1}{T_s} \tag{2.11}$$

Sampling must be fast enough to capture the most rapid changes in the signal being sampled. According to Nyquist theory [33], a signal with maximum frequency of f_N Hz must be sampled at least $2f_N$ times per second to make it possible to reconstruct the original signal from the samples.

The second step in the conversion between an analog signal and a digital signal is to quantize and digitize the analog values. A quantization level that approximates the

sample-and-hold value will be selected and then a binary digital code that identifies the quantization level will be assigned. The quantized signal represented in Figure 2.5.

2.2.3 Time-Domain Analysis

As mentioned above, the vibration signal of pumps after ADC is a discrete-time-domain signal. This dissertation analyzes the vibration signal first in time domain and imports the analysis results into NN to examination the feasibility for the application of a simple NN topology for operation point detection of an impeller pump. There are many methods for time-domain signal analysis, the following are some of the key methods for time-domain signal analysis:

Time Waveform Analysis: Directly observe and analyze the signal's behavior over time to identify patterns, periodicities, transients, and spikes. The signal characteristics embodied by the waveform are often used for fault diagnosis of mechanical equipment. The original time-domain vibration signals under two different operating point are shown in Figure 2.6 as an example with small amount of data. Ten rows of signal are displayed at each operating point class, with a sampling rate of 1000 Hz and 128 sampling points. By directly observing and analyzing the changes of signals over time, it is found that there is no obvious regularity in the time domain-signals at the same operating point. But the difference between the waveforms at the two different operating points is obvious. This kind of original signal will be used as a type of input data for network training in experiments of this study.

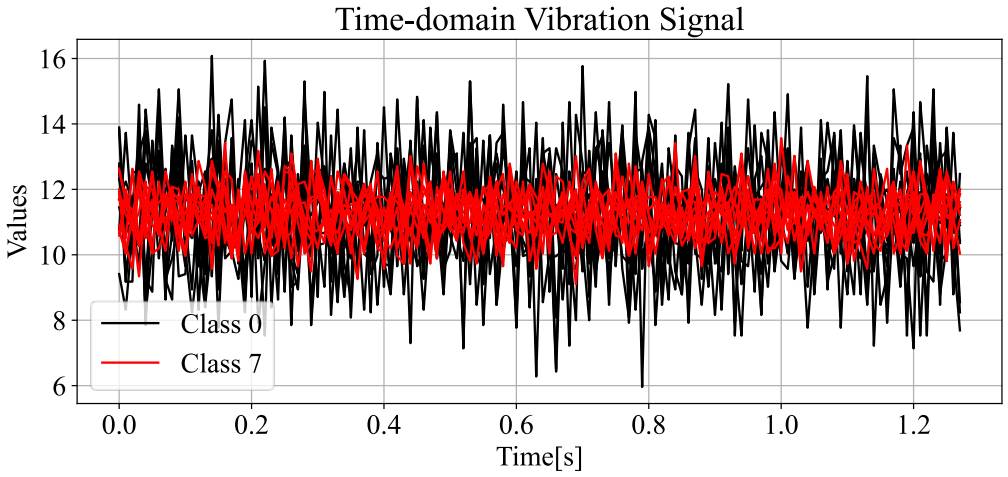


Figure 2.6: Original time-domain vibration signal in two different classes

Statistical Analysis: In Mean and Root Mean Square (RMS) analysis, the average value and energy of the signal can be assessed. Peak-to-Peak (PtP) Value of a signal are used to detect abnormal spikes and vibrations. To detect deviations from normal operation, analyzing the symmetry and sharpness of the signal distribution is effective.

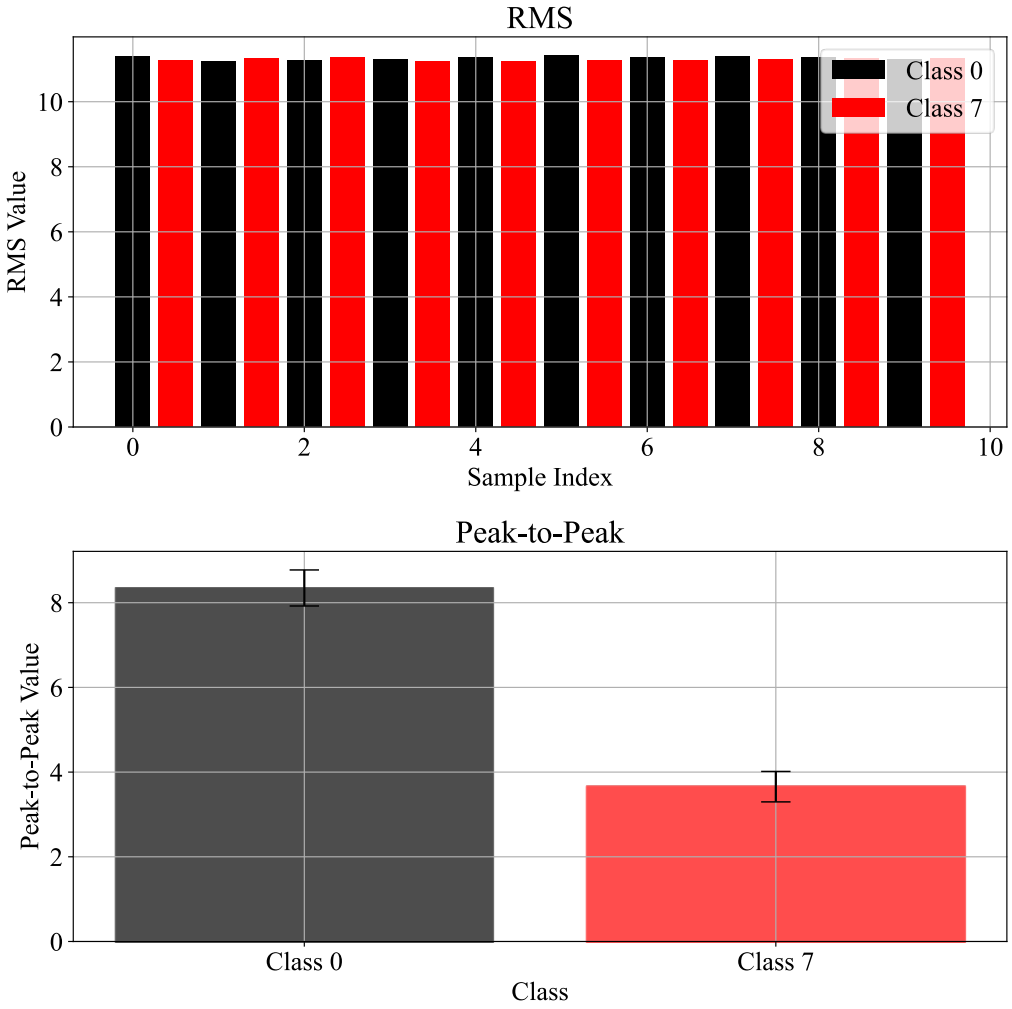


Figure 2.7: Statistical analysed vibration signal in two different classes

Figure 2.7 shows the statistical analyzed signal in the two different classes from Figure 2.6, since both operation states are normal operation points, the difference in RMS is very small, but the difference in PtP is obvious. In addition, after this analysis method, the amount of calculation is significantly decreased. The required memory space and computation time will also be greatly reduced.

Time Series Analysis: The Series analysis involves examining patterns, trends, and behaviors inherent in the signal to make predictions or derive insights [34]. The methods of time series analysis include descriptive analysis, time series decomposition,

autoregression (AR) models, moving average (MA) models, etc. These are analytical methods that make predictions directly and are not suitable for NNs after processing.

Empirical Mode Decomposition (EMD): EMD is an analysis method used to decompose a time series into a finite set of Intrinsic Mode Functions (IMFs) and a residue component. It was developed by Huang et al. in 1998 to adaptively decompose non-stationary and nonlinear time series data into components that have well-defined instantaneous frequencies [35]. EMD is an adaptive method that does not require any predefined basis functions, making it highly suitable for analyzing complex signals.

The EMD process identifies first all local extrema (maxima and minima) in the original signal. Using these extreme values, the upper and lower envelopes are constructed by spline interpolation. Then the average of these two envelopes is calculated to obtain the local mean. By subtracting this local mean from the original signal, a new signal is obtained. This process will be repeated until the new signal meets the IMF criteria, which means the counts of extrema and zero crossings are either equal or differ by at most one, and the local mean is zero at every point. Figure 2.8 shows the first IMF and its calculation process.

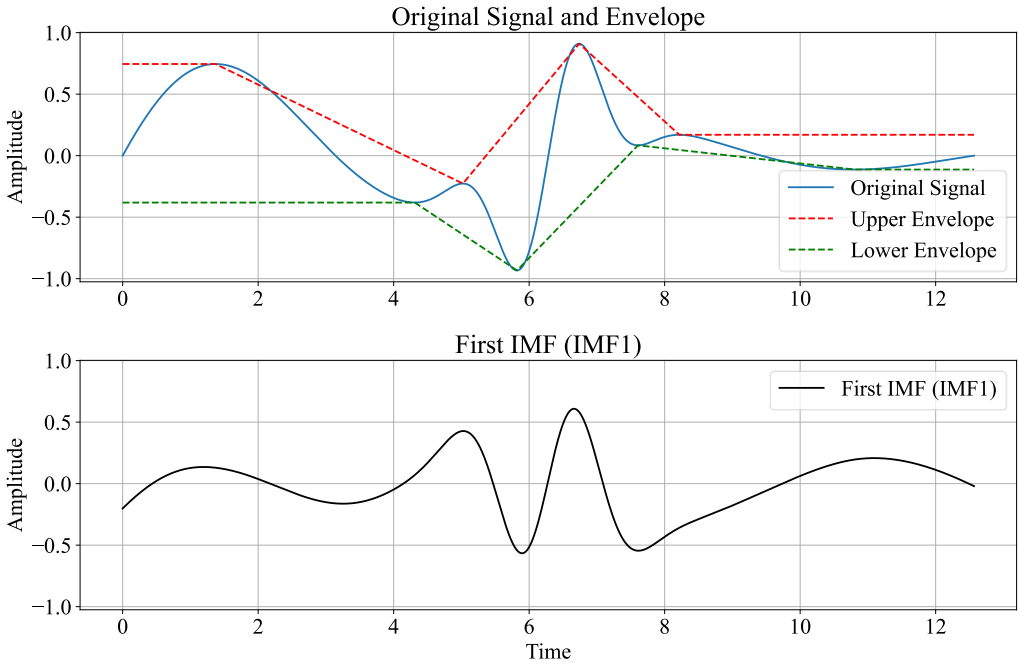


Figure 2.8: EMD Process

After obtaining the first IMF, the next IMF can be obtained by subtracting the previously extracted IMF from the original signal. The remaining signal after removing the IMF is

used as the new signal for further decomposition, this process continues until the residue becomes a monotonic function or contains no more meaningful IMFs. The result is that the original signal is expressed as the sum of its IMFs and a final residue:

$$x(t) = \sum_{i=1}^{n} c_i(t) + r_n(t) \quad (2.12)$$

where $c_i(t)$ are the extracted IMFs, n is the number of IMFs and $r_n(t)$ is the final residue. Figure 2.9 illustrates significant variations in the EMD of a single row vibration signal across different operating points.

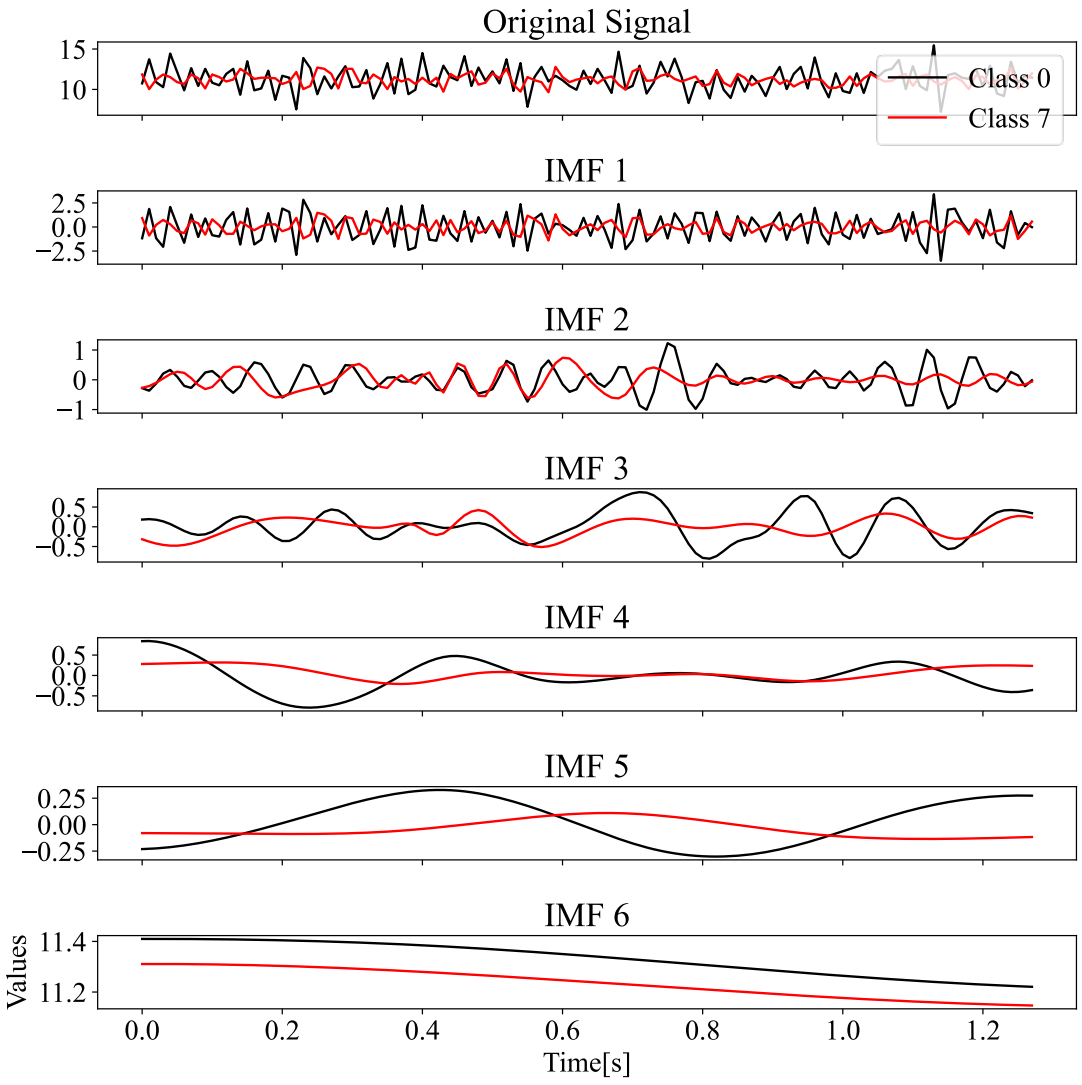


Figure 2.9: EMD of the vibration signal in two different classes

Therefore, in the NN training of this research, the input signals after time-domain analysis include the wave form of the original signal, its RMS and PtP values, and the signal processed by EMD.

2.2.4 Frequency-Domain Analysis

Frequency-domain signal analysis is the process of studying the characteristics and behavior of a signal over a frequency range. Different from time-domain analysis, frequency domain analysis focuses on the frequency components of a signal. A frequency-domain signal can be represented by its spectrum, which means the amplitude and phase of different frequency components. To convert the time-domain into frequency-domain the Fourier Transform is used [34].

Fourier Transform

Fourier Transform is a mathematical model which transforms the signals between two different domains, such as transforming signal from time-domain to frequency-domain or vice versa. The Fourier Transform of a continuous-time signal is given by

$$X(\Omega) = \int_{-\infty}^{\infty} x(t)e^{-j\Omega t}dt \quad (2.13)$$

where Ω represents the analog frequency in radians/s. If $x(t)$ is sampled at a sampling frequency $1/T_s$ then the integration becomes a summation over the variable n as

$$X\left(e^{j\Omega T_s}\right) = \sum_{n=-\infty}^{\infty} x(nT_s)e^{-j\Omega nT_s} \quad (2.14)$$

Because $\Omega T_s = \frac{2\pi f}{f_s} = \omega_s$, where $f_s = \frac{1}{T_s}$ and ω_s is defined as the normalized digital frequency in units of radians per sample. If we define $x(n)$ to represent sample $x(nT_s)$ the expression comes to

$$X\left(e^{j\omega_s}\right) = \sum_{-\infty}^{\infty} x(n)e^{-j\omega_s n} \quad (2.15)$$

which is the Discrete-time Fourier Transform (DTFT) of $x(n)$.

The continuous spectrum consists of an infinite number of samples, the storage of which requires infinite memory and infinite time for processing. It is not feasible to process such signals in a Digital Signal Processing (DSP) processor. The solution to this problem is to process a finite number of samples from $X\left(e^{j\omega}\right)$. The number of samples taken is normally equal to the number of samples of the sequence $x(n)$. These samples form the Discrete Fourier Transform (DFT) for a finite length sequence. The DFT of a finite length sequence $x(n)$, $0 \leq n \leq N$ -1 is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn} \text{ for } 0 \leq k \leq N\text{-}1 \quad (2.16)$$

which, $x(n)$ is the discrete time signal, $X(k)$ is the discrete frequency domain, N is the length of the signal. The integral for calculating the continuous signal is replaced by a sum of N samples and the index k indicates the number of the spectral line under consideration. In order to simplify the representation a twiddle factor as is defined in $W_N = e^{-j\frac{2\pi}{N}}$. Substituting the factor into Equation (2.16) the DFT is given by

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \text{ for } 0 \leq k \leq N\text{-}1 \quad (2.17)$$

However, DFT is a finite transform of a finite length sequence that requires a finite amount of memory to store samples and a finite time to process the samples. Sometimes sequences and transforms are so long that the processing time required is too long and the memory required to store the samples is too large. Such sequences cannot be processed in real time in a DSP processor. Some properties of DFT can be exploited to reduce the computation time and the memory required. The transforms that are formed after such simplification are referred to as the FFT. There are several versions of the FFTs, the most commonly used FFT algorithm is the Cooley-Tukey algorithm, which recursively breaks down a DFT of any composite size N into many smaller DFTs first. When N is a power of 2, the Cooley-Tukey algorithm can be employed, the computational load is reduced to only $\frac{N}{2} log_2 N$ [36]. Such an algorithm is based on a divide and conquer technique by breaking a length-N DFT into two length-$\frac{N}{2}$ DFTs. For an input sequence $x(n)$ of length N:

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} \left(x(2n) e^{-\frac{j2\pi k(2n)}{N}} + x(2n+1) e^{-\frac{j2\pi k(2n+1)}{N}} \right) \quad (2.18)$$

These terms correspond to the DFTs of the even-indexed part and the odd-indexed part, respectively. The algorithm recursively computes the DFT of the even and odd parts until it reaches a DFT of size 1, and these values are the sequence itself. The final DFT will be given by the combining these values. Figure 2.10 shows the frequency-domain signal of the original signal in Figure 2.6 after FFT.

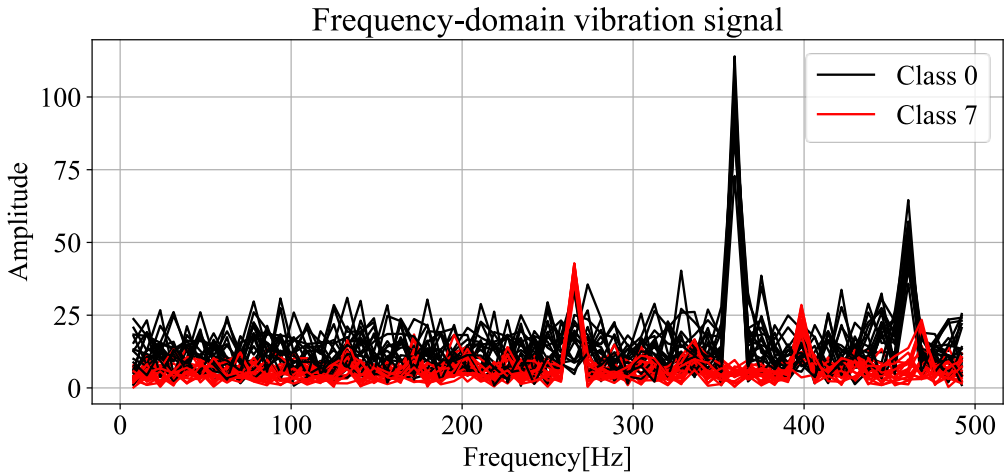


Figure 2.10: Frequency-domain vibration signal in two different classes

The result of FFT is a complex array containing the positive and negative frequency parts of the signal. The FFT result of a real signal satisfies conjugate symmetry, that is, the positive and negative frequency parts in the spectrum are mirror symmetric. Displaying the first half of the spectrum (0 to $\frac{N}{2}$) is sufficient to fully represent the spectrum information of real signals and the first element represents the 0 Hz component (usually removed). This approach can also effectively save memory and processing time.

After obtaining the frequency spectrum, it can be further processed for noise reduction and feature extraction to improve the computational efficiency and accuracy of the NN model. However, the NN itself is also adapted to complex data, so in this dissertation there is non-complex noise reduction and feature extraction on the spectrum.

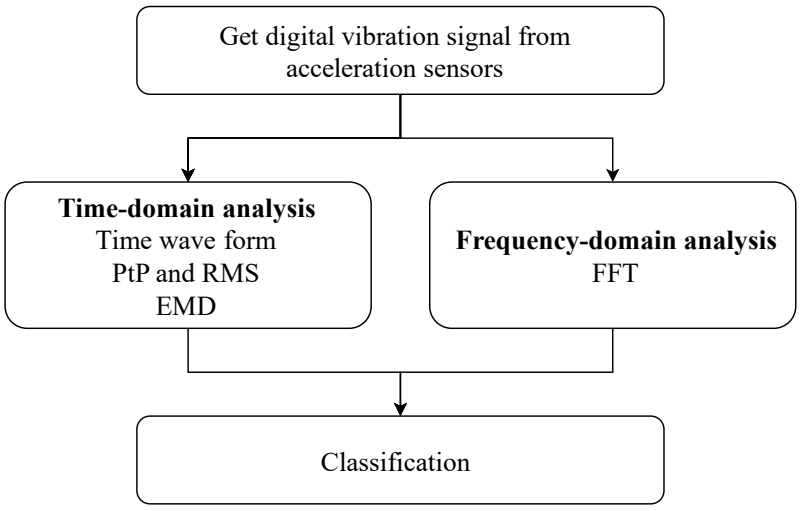


Figure 2.11: Data recording, data processing and classification

As in Figure 2.11 presents, after conducting time- and frequency-domain analysis and processing, the vibration signal will serve as input to the NN-based classifier for classification, aiming to distinguish different operating points, as elaborated further in Section 2.3.

2.3 Classification

Classification is a fundamental concept in machine learning and data science that involves classifying data into predefined classes or groups. The point of a classification algorithm is to learn from a set of labeled data (training data) and predict or classify new, unseen data [20]. Classifiers are based on more or less complicated decision functions that are generated based on a set of training data (training set). A set of test data (test set) is then used to validate the respective classifiers. Then some evaluation metrics will be used to evaluate the performance of this classifier such as Accuracy, Recall etc.

Classification arises in various domains, including image recognition, natural language processing, and medical diagnosis. In this dissertation, the problem of distinguishing the operating point of a centrifugal pump from the vibration signal is equivalent to a multi-classification problem. In order to make the classifier achieve the ideal performance, it is necessary to select a suitable classification algorithm based on the characteristics of the data. Common classification algorithms are as follows:

Logistic Regression: A linear model used for binary classification that estimates probabilities using the logistic function.

Decision Trees: A non-linear model that splits the data into branches based on feature values, leading to a decision node.

Random Forest: An ensemble method that combines multiple decision trees to improve robustness and accuracy.

Support Vector Machines (SVM): A model that finds the optimal hyperplane that maximizes the margin between different classes.

k-Nearest Neighbors (k-NN): A simple algorithm that assigns a class to a data point based on the majority class of its k-nearest neighbors.

Neural Networks: Complex models capable of capturing intricate patterns in data.

Licht [7] successfully used SVM and k-NN to classify the operating conditions. Classification results of about 70% were achieved at all tested pump speeds. Deep neural networks (DNNs) are particularly powerful for multi-class classification due to their ability to learn complex, non-linear feature representations. This dissertation attempts to classify the operating conditions using a simple NN with a single hidden layer.

2.3.1 Neural Networks

A NN is a mathematical model that tries to simulate the structure and functionalities of biological NNs [37], it has been around for at least 50 years [38]. The basic units of NNs are nodes, which based on biological neurons in the mammalian brain. The connections between neurons (nodes) are also modeled after the biological brain, and these connections develop over time (through training). NNs are widely used in classification problems. They mimic the workings of neurons in the human brain and are able to learn from substantial amounts of data and perform complex pattern recognition and classification. With the reinvigoration of NN in the 2000s, deep learning has become an extremely active area of research that is paving the way for modern machine learning [1]. The following are some specific applications and principles of NN in the field of classification:

Image classification

NN, especially CNN, performs well in image classification. They can automatically extract features from images and classify them. For example, CNN can be used to identify objects in images, facial recognition, handwritten digit recognition, etc.

Speech recognition

NN is also widely used in speech recognition systems. By converting speech signals into eigenvectors, NNs can learn and classify different speech patterns, thereby achieving speech-to-text conversion.

Natural Language Processing

In the field of natural language processing (NLP), Recurrent Neural Networks (RNN) and LSTM Networks are used for tasks such as text classification, sentiment analysis, and language translation.

In this dissertation, the applicability of a simple NN topology for operating condition classification by vibration signals of a centrifugal pump was investigated. The above cases have proved the success of this method. Some basic concepts of NNs are first described.

2.3.1.1 Neuron

A biological neuron is basic building block of biological NNs (systems) which includes the brain, spinal cord and peripheral ganglia [37]. The neurons send information to each other in the form of activation signals through directed connections [39]. Figure 2.12 shows the basic unit of a biological brain neuron.

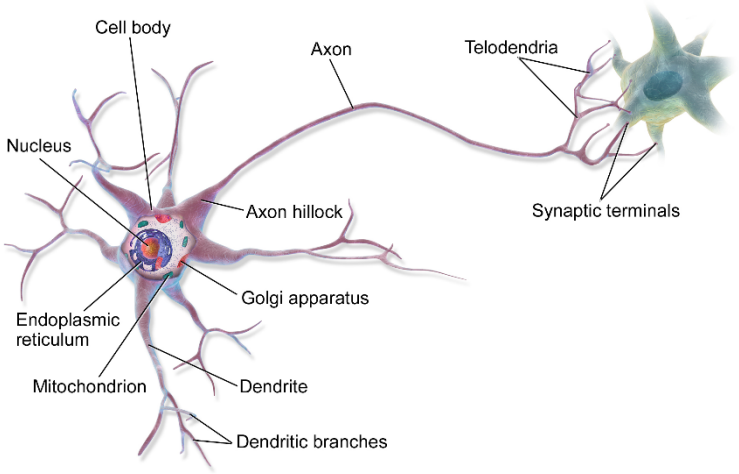


Figure 2.12: The basic unit of a biological brain neuron [40]

A NN is an information processing system, whose structure and function are modeled on the nervous system of animals and humans. The artificial neuron is a basic building block of every NN. Its design and functionalities are derived from observation of the biological neuron. An artificial neuron accepts more than one electrical input signal and outputs another electrical signal. A simplified model can be built to represent the functioning of an artificial neuron. But a biological neuron does not produce an output, which is simply a linear function of the input. In addition, the neurons don't respond immediately, instead suppressing the input until it's large enough to trigger an output signal. When building a model, it can be thought of as a threshold that must be reached before an output signal can be generated. In summary, a model as shown in Figure 2.13 was built to represent the functioning of an artificial neuron.

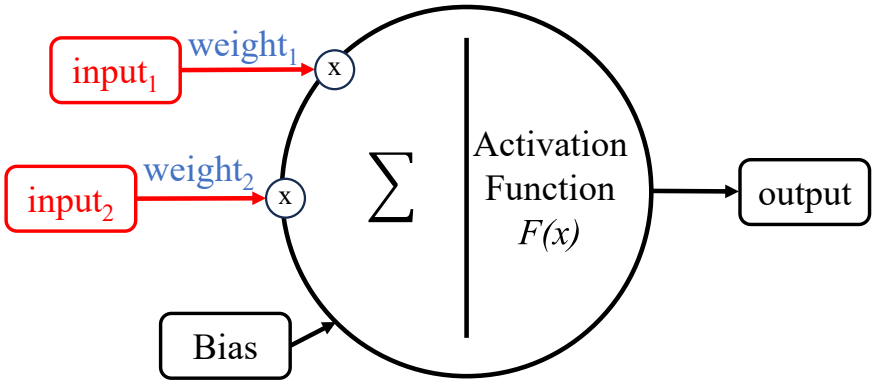


Figure 2.13: Working principle of an artificial neuron

The main feature of this model is that the neuron accepts multiple weighted inputs, bias and generates only one output with an activation function. The simplicity of the mathematical description of the artificial neuron model can be seen below:

$$y(k) = F\left(\sum_{i=0}^{m} w_i(k) \cdot x_i(k) + b\right) \quad (2.19)$$

where $x_i(k)$ is input value in discrete time where goes from 0 to m, $w_i(k)$ is weight value in discrete time where goes from 0 to m, b is bias, F is an activation function, $y(k)$ is output value in discrete time. In NNs, neurons are also called nodes.

2.3.1.2 Activation Function

As seen from the model of an artificial neuron and its Equation (2.19) the major unknown variable of the model is its activation function. Activation function defines the properties of artificial neuron and can be any mathematical function. As mentioned above, biological neurons do not produce linear outputs. The activation function introduces a non-linear factor to the model, so that the model function can approximate any non-linear function.

The to be used non-linear activation functions in this dissertation include Sigmoid function, Rectified Linear Unit (ReLU) function, Hyperbolic Tangent (Tanh) function, SoftMax function etc.

The Sigmoid function is a popular activation function used in NNs, especially in early models and logistic regression. It maps any real-valued number into a value between 0 and 1, making it useful for binary classification tasks. The Sigmoid function is defined as [13]:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.20)$$

where e is the base of the natural logarithm, and x is the input to the function. The derivative of the sigmoid function is easily to calculate and is given by:

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)) \quad (2.21)$$

it is useful for backpropagation in NNs.

The Sigmoid function has an S-shaped curve (see in Figure 2.14), which is smooth and differentiable.

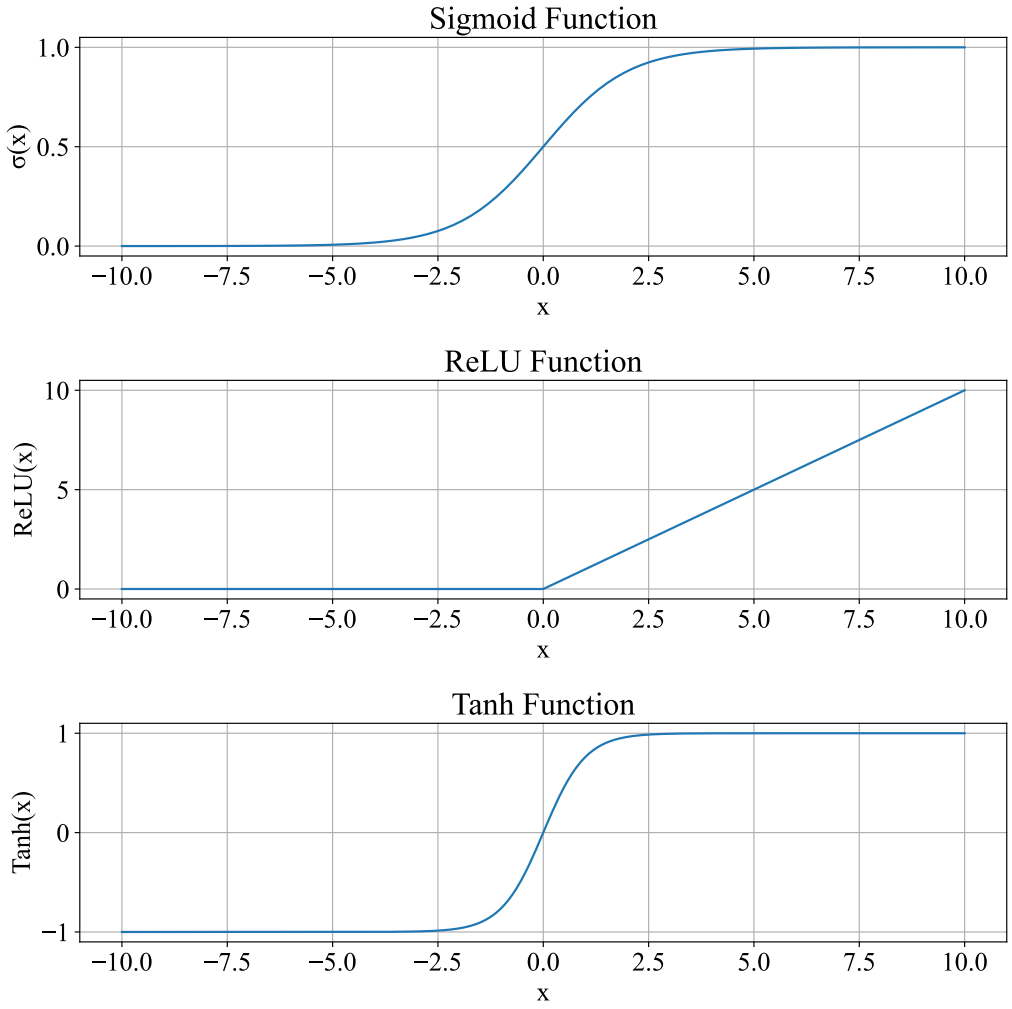


Figure 2.14: Activation functions

The ReLU function has become the default activation function for many types of NNs due to its simplicity and effectiveness. The ReLU function is defined as [41]:

$$\text{ReLU}(x) = \max(0, x) \quad (2.22)$$

The output of the function is the input directly if the input > 0, otherwise will be 0. It is computationally efficient because it involves a simple thresholding of the matrix values

at 0, which can improve computational efficiency and the convergence of the network. The curve of the ReLU function is shown in Figure 2.14.

The Tanh function is similar to the sigmoid function but has some advantages that often make it preferable in practice [42]. The definition of the function is:

$$\mathrm{Tanh}(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad (2.23)$$

The S-shaped curve (Figure 2.14) of the Tanh function is also similar to the curve of the Sigmoid function, but it is zero-centered, which can lead to faster convergence because the gradients will not be biased towards any particular direction.

The SoftMax function is commonly used in the output layer of a NN for multi-class classification tasks. It converts a vector of values into a probability distribution, where the probability of each value is proportional to the exponent of the input value. The SoftMax function is defined as [20]:

$$\mathrm{SoftMax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \quad (2.24)$$

where x is the input vector, x_i is the i-th element of the input vector, and n is the number of elements in the input vector. The output of the SoftMax function is a vector of values in the range (0, 1), which standing for a probability distribution. The sum of the probabilities is 1, the output probabilities can be interpreted as the probability of each class in a multi-class classification problem. The subtraction of the maximum value of the input vector from each element before exponentiating it to avoid numerical overflow. A simple example of the SoftMax function is shown in Figure 2.15.

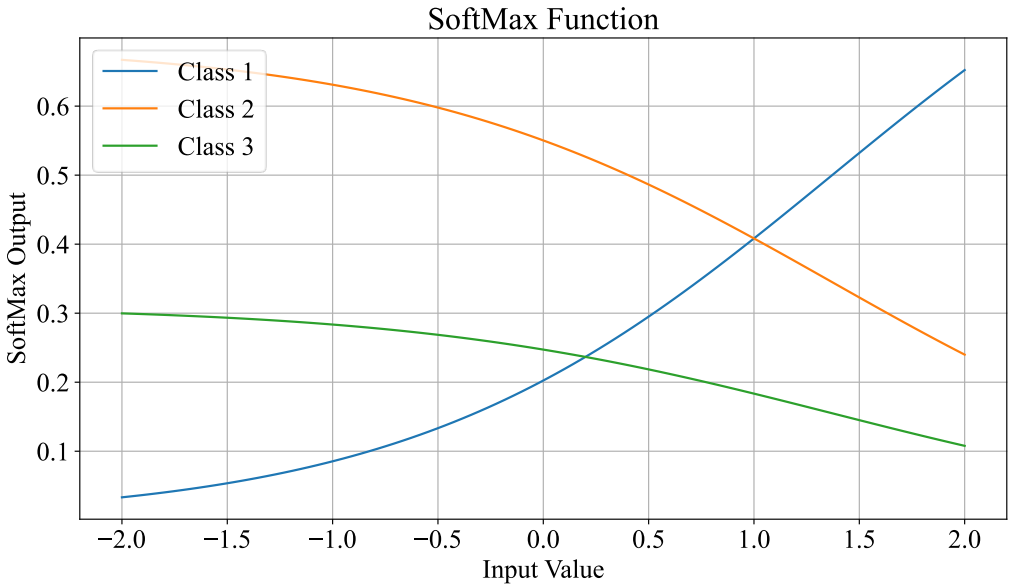


Figure 2.15 SoftMax function

2.3.1.3 Layer Structure

Numerous interconnected artificial neurons form a NN. NNs contain diverse types of layers, which are arranged and connected in several ways. A typical NN consists of the following types of layers (seen Figure 2.16):

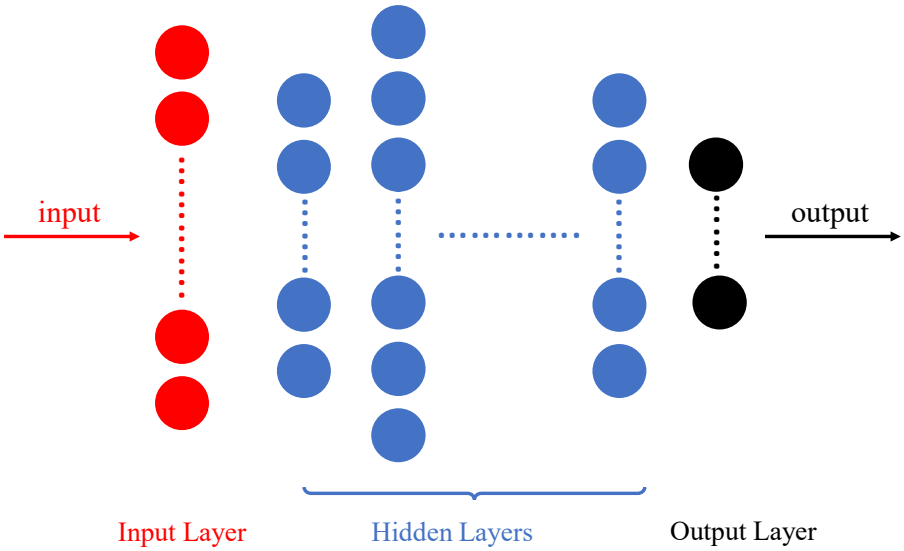


Figure 2.16: Layer structure of typical NNs

Input layer: This first layer is how the input data fed into the network, so called input layer. The number of neurons in an input layer is typically the same number as the input feature to the network [38]. Input layer is followed by hidden layers.

Hidden layer: The hidden layer is located between the input layer and the output layer, performing feature extraction and transformation to extract complex patterns in the data. The weight values on the connections between the layers are how NNs encode the learned information extracted from the raw training data. A NN can have one or more hidden layers. The more hidden layers there are, the deeper the network is (DNN).

Output layer: The last layer represents the output layer. The prediction of the model comes from this layer. The final output may be a regression or a set of probabilities.

2.3.1.4 Loss Function

The loss function, also known as the cost function, is a function that measures the difference between the output of a NN and the true labels [13]. It plays a crucial role during the training process of NNs and directly affects the optimization process. There are various common loss functions, and the specific choice depends on the type of problem, such as regression problems, classification problems, etc. Here are some common loss functions:

Mean Squared Error (MSE): The MSE calculates the average squared error between the predicted values and the true values. The Formula is:

$$L(\hat{y}, y) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y_i} - y_i)^2 \quad (2.25)$$

where $\hat{y_i}$ is the predicted value, y_i is the true value, and n is the number of samples [43].

Cross-Entropy Loss: The Cross-Entropy Loss commonly used for classification problems, especially multi-class classification problems. It measures the difference between the predicted probability distribution and the actual distribution. For binary classification (Binary Cross-Entropy Loss (BCE)), the formula is:

$$L(\hat{y}, y) = -\frac{1}{n}\sum_{\mathrm{i}=1}^{n} y_i \log(\hat{y_i}) + (1 - y_i)\log(1 - \hat{y_i}) \quad (2.26)$$

where $\hat{y_i}$ is the predicted value, y_i is the true value, and n is the number of samples.

For multi-class classification (Categorical Cross-Entropy Loss (CCE)), the formula is:

$$L(\hat{y}, y) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} y_{i,c} \log\left(\widehat{y_{i,c}}\right) \quad (2.27)$$

where $\hat{y}_i$ is the predicted value, y_i is the true value, n is the number of samples, and C is the number of classes [13].

Mean Absolute Error (MAE): The MAE also used for regression problems, it calculates the average absolute difference between the predicted values and the true values, it definite as:

$$L(\hat{y}, y) = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \quad (2.28)$$

where $\hat{y}_i$ is the predicted value, y_i is the true value, and n is the number of samples [43].

Custom Loss Function: In certain complex or specific applications, you can define your own loss function to better meet specific needs.

The training process of a NN adjusts the model parameters to minimize the value of the loss function, thereby improving the model's performance [20]. Different tasks and datasets are suited to different loss functions and choosing the appropriate loss function is a key factor in the success of model training.

2.3.1.5 Iteration Process - Training

A simple NN with three layers is shown in Figure 2.17. This NN has an input layer, a hidden layer, and an output layer. Each layer has only 3 neurons. Through this simple NN, the mathematical model of the iteration process also the training of the NN can be simplified and present.

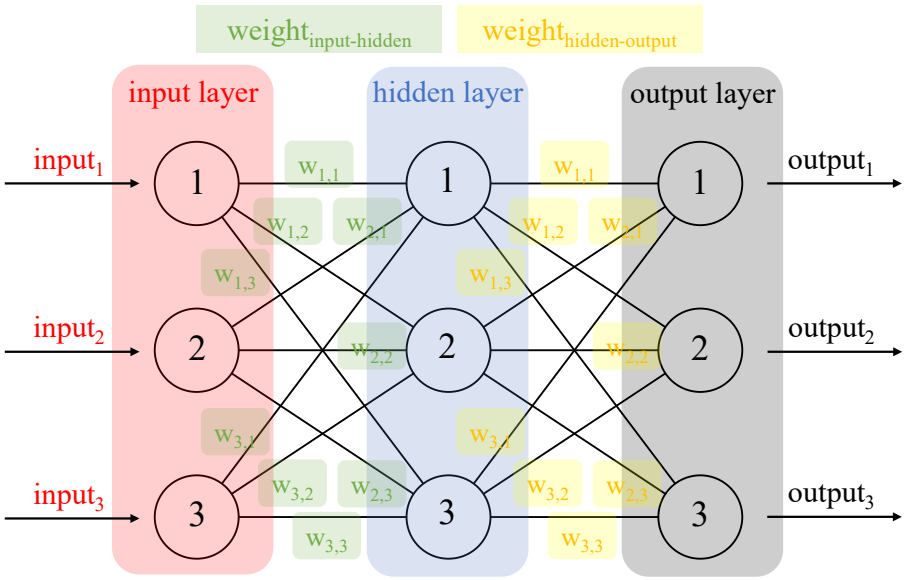


Figure 2.17: Model of a NN

Forward propagation

Forward propagation is the process by which a NN makes predictions. It involves computing the output of the network layer by layer, in the direction from the input layer to the output layer, using the current set of weights and biases. The forward propagation process is crucial for both training and inference in NNs. The information flow is from left to right.

In Figure 2.17, the input eigenvector is formulated as $\mathbf{X} = [x_1,\ x_2, x_3]^T$, the weight matrix $\mathbf{W_1} \in \mathrm{R}^{3\times3}$ and bias vector $\mathbf{b_1} \in R^3$ are:

$$\mathbf{W_1} = \begin{pmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} \\ w_{31}^{(1)} & w_{32}^{(1)} & w_{33}^{(1)} \end{pmatrix} \quad \text{and} \quad \mathbf{b_1} = [b_1^{(1)}, b_2^{(1)}, b_3^{(1)}]^T$$

The linear combination for the hidden layer $\mathbf{Z_1}$ is computed as follows:

$$\mathbf{Z_1} = \mathbf{W_1 X} + \mathbf{b_1} \quad (2.29)$$

substituting the vector is

$$\mathbf{Z_1} = \begin{pmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} \\ w_{31}^{(1)} & w_{32}^{(1)} & w_{33}^{(1)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \end{pmatrix} \quad (2.30)$$

The activation output of the hidden layer $\mathbf{A_1}$ is:

$$\mathbf{A_1} = SoftMax(\mathbf{Z_1}) \quad (2.31)$$

where $SoftMax$ is the activation function of the hidden layer as example.

The weight matrix $\mathbf{W_2} \in R^{3\times3}$ and bias vector $\mathbf{b_2} \in R^3$are:

$$\boldsymbol{W}_2 = \begin{pmatrix} w_{11}^{(2)} & w_{12}^{(2)} & w_{13}^{(2)} \\ w_{21}^{(2)} & w_{22}^{(2)} & w_{23}^{(2)} \\ w_{31}^{(2)} & w_{32}^{(2)} & w_{33}^{(2)} \end{pmatrix} \quad \text{and} \quad \mathbf{b}_2 = [b_1^{(2)}, b_2^{(2)}, b_3^{(2)}]^T$$

The linear combination for the output layer $\mathbf{Z}_2$ is computed as follows:

$$\mathbf{Z}_2 = \mathbf{W}_2\mathbf{A}_1 + \mathbf{b}_2 \quad (2.32)$$

substituting the vector is

$$\mathbf{Z_2} = \begin{pmatrix} w_{11}^{(2)} & w_{12}^{(2)} & w_{13}^{(2)} \\ w_{21}^{(2)} & w_{22}^{(2)} & w_{23}^{(2)} \\ w_{31}^{(2)} & w_{32}^{(2)} & w_{33}^{(2)} \end{pmatrix} \begin{pmatrix} a_1^{(1)} \\ a_2^{(1)} \\ a_3^{(1)} \end{pmatrix} + \begin{pmatrix} b_1^{(2)} \\ b_2^{(2)} \\ b_3^{(2)} \end{pmatrix} \quad (2.33)$$

The activation output of the output layer $\mathbf{A_2}$ is:

$$\mathbf{A_2} = \sigma\,(\mathbf{Z_2}) \quad (2.34)$$

where σ is the activation function (Sigmoid as example) of the hidden layer.

Back propagation

Backpropagation, short for backward propagation of errors, is the algorithm used for training NNs. The goal is to compute the gradient of the loss function with respect to each parameter (weight and bias) and then update them using gradient descent. This enables the adjustment of weights to minimize the loss. This process is fundamental to gradient-based optimization algorithms, such as stochastic gradient descent.

Loss Calculation

The MSE is chosen to describe the computational process as example. The target value is definite as $\mathbf{y} = [y_1,\ y_2,\ y_3\ ]^T$, the loss function of NN is defined as:

$$L = \frac{1}{3}\sum_{i=1}^{3}\left(a_i^{(2)} - y_i\right)^2 \quad (2.35)$$

First, in output Layer, the gradient of the loss with respect to the output layer's inputs will be calculate.

Gradient of loss with respect to output is:

$$\frac{\partial L}{\partial \mathbf{a}^{(2)}} = \mathbf{a}^{(2)} - \mathbf{y} \quad (2.36)$$

Gradient of output with respect to input is:

$$\frac{\partial \mathbf{a}^{(2)}}{\partial \mathbf{z}^{(2)}} = \sigma'\left(\mathbf{z}^{(2)}\right) \quad (2.37)$$

Therefore, the gradient of the loss with respect to the input to the output layer is:

$$\delta^{(2)} = \frac{\partial L}{\partial \mathbf{z}^2} = \left(\mathbf{a}^{(2)} - \mathbf{y}\right) \odot \sigma'\left(\mathbf{z}^{(2)}\right) \quad (2.38)$$

Then, within the hidden layers, gradients are propagated backward through the network, and the gradient of the loss with respect to the input of each layer is calculated.

The gradient of the output layer input to the hidden layer output:

$$\frac{\partial \mathbf{z}^{(2)}}{\partial \mathbf{a}^{(1)}} = \mathbf{W}^{(2)} \quad (2.39)$$

The gradient of the hidden layer output to the hidden layer input:

$$\frac{\partial \mathbf{a}^{(1)}}{\partial \mathbf{z}^{(1)}} = g'(\mathbf{z}^{(1)}) \quad (2.40)$$

Therefore, the gradient of the loss with respect to the hidden layer input is:

$$\delta^{(1)} = \frac{\partial L}{\partial \mathbf{z}^1} = \left(\mathbf{W}^{(2)}\right)^T \delta^{(2)} \odot SoftMax'\left(\mathbf{z}^{(1)}\right) \quad (2.41)$$

Update the network weights and biases

Update for output layer parameters:

$$\mathbf{W}^{(2)} \coloneqq \mathbf{W}^{(2)} - \eta\delta^{(2)}\left(\mathbf{a}^{(1)}\right)^T \quad (2.42)$$

$$\mathbf{b}^{(2)} \coloneqq \mathbf{b}^{(2)} - \eta\delta^{(2)} \quad (2.43)$$

Update for hidden layer parameters:

$$\mathbf{W}^{(1)} \coloneqq \mathbf{W}^{(1)} - \eta\delta^{(1)}(\mathbf{x})^T \quad (2.44)$$

$$\mathbf{b}^{(1)} \coloneqq \mathbf{b}^{(1)} - \eta\delta^{(1)} \quad (2.45)$$

where η presents the learning rate of the NN.

The flow chart of the iteration process is shown in Figure 2.18.

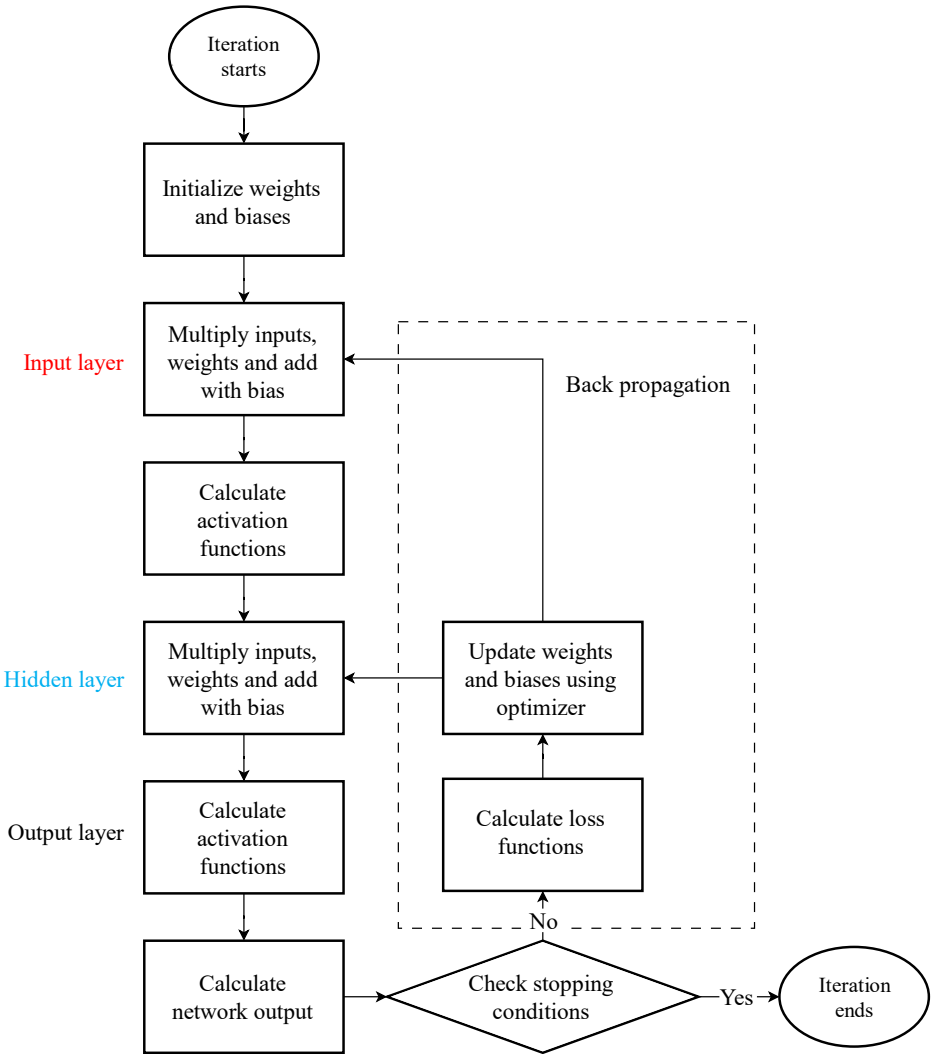


Figure 2.18: Iteration process of a NN

The process describes the forward propagation and back propagation algorithms of the three-layer NN. The output is calculated through forward propagation, the error is calculated using the loss function, the gradient is calculated through back propagation, and finally the weights and biases of the network are updated to complete a training iteration. The last step involves checking whether the stopping condition has been met. If the stopping condition has not been met, the process will return to the forward propagation step. Once the stopping condition is met, the iteration process concludes.

2.3.1.6 Learning Rate

Learning rate of a NN determines the step size of each parameter update [13]. The appropriate learning rate selection is crucial to the training speed and final performance of the model. Affects the stability of the training process. Too high a learning rate can

cause the model to overshoot minima, while too low a learning rate can result in a slow convergence or getting stuck in local minima.

A fixed learning rate throughout the training is the so-called constant learning rate. An appropriately chosen learning rate allows the model to converge efficiently to the optimal solution. The update rule for the parameters with a constant learning rate η is given by

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L(\theta_t) \quad (2.46)$$

where: θ_t denotes the parameters of the t-th iteration, $\nabla_\theta L(\theta_t)$ is the gradient of the loss function with respect to the parameters.

The learning rate can also be manual adjusted during training, including Step Decay learning rate (reduced by a factor after a fixed number of epochs), Exponential Decay learning rate (decreased exponentially over epochs), 1/t-Decay learning rate (reduced in inverse proportion to the epoch number) and Annealing learning rate (gradually reduced as training progresses).

In practical conditions, the adaptive learning rates are used more frequently, which means the learning rate will be automatically adjusted based on the training progress. Some NN optimizers include this capability.

2.3.1.7 Optimizers

Optimizers are used to update the parameters (e.g., weights and biases) of a NN to minimize the loss function. And some optimizers are able to adjust the learning rate. Common optimizer algorithms are listed below.

Stochastic Gradient Descent (SGD):

The SGD algorithm is the most basic optimizer with simple computation and low memory requirements [38]. The update rule for SGD is given by:

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta L\left(\theta; x^{(i)}, y^{(i)}\right)$$

where θ_t represents the model parameters at the t-th iteration, η is the learning rate, $\nabla_\theta L\left(\theta; x^{(i)}, y^{(i)}\right)$ is the gradient of the loss function with respect to the parameters θ, computed using a single training example $x^{(i)}, y^{(i)}$.

By updating the parameters for each training example, SGD can converge faster than Batch Gradient Descent, especially for large datasets. But the randomness in the updates can cause the loss function to fluctuate rather than steadily decrease.

Adaptive Gradient (Adagrad):

The Adagrad algorithm, individually adapts the learning rates of all model parameters by scaling them inversely proportional to the square root of the sum of all the historical squared values of the gradient [41]. For each parameter θ_t and gradient g_t , the accumulated sum of squared gradients up to time step t for each parameter i is:

$$G_{t,i} = \sum_{\tau=1}^{t} g_{\tau,i}^2 \quad (2.47)$$

The update rule for each parameter i is:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,i} + \epsilon}} g_{t,i} \quad (2.48)$$

where η is the initial learning rate, and ϵ is a small constant to avoid division by zero. Adagrad is a powerful optimization algorithm for dealing with sparse data and varying gradients by adapting the learning rate for each parameter based on the accumulated gradient information. However, the accumulated gradient $G_{t,i}$ can become very large, leading to excessively small learning rates and causing the algorithm to stop learning.

Root Mean Square Propagation (RMSProp)

The RMSProp algorithm [40] modifies Adagrad to perform better in the nonconvex setting by changing the gradient accumulation into an exponentially weighted moving average. The RMSProp algorithm updates the model parameters as follows:

For each iteration t:

$$g_t = \nabla L(\theta_t) \quad (2.49)$$

where g_t is the gradient, L is the loss function and θ is the parameter.

The moving average of the squared gradients is updated as:

$$\mathbf{E}[g^2]_t = \gamma \mathbf{E}[g^2]_{t-1} + (1-\gamma) g_t^2 \quad (2.50)$$

where the initial $\mathbf{E}[g^2] = 0$ is the moving average vector and γ is a decay rate (typically $\gamma = 0.9$).

The update rule for the parameters is:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \quad (2.51)$$

where η is the learning rate and ϵ is a small constant (e.g., 10^{-8}) to prevent division by zero.

RMSProp adapts the learning rate based on the gradients, which helps in training models with sparse gradients or when the magnitude of gradients varies significantly. By smoothing out the gradient updates, RMSProp can lead to more stable and faster convergence compared to standard gradient descent. However, this algorithm needs to maintain a moving average of the squared gradient of each parameter, which increases the amount of computation and memory.

Adaptive Moment Estimation (Adam):

Adam [40] is yet another adaptive learning rate optimization algorithm. The name Adam derives from the phrase adaptive moments. It is perhaps best seen as a variant combining RMSProp and momentum for adaptive learning rates. Adam computes individual adaptive learning rates for different parameters from estimates of first-order and second-order moments (mean and uncentered variance) of the gradients, the algorithm updates the parameters as follows:

Initialization: parameters θ, first-order moment vector $\mathbf{m} = 0$ and second-order moment vector $\mathbf{v} = 0$.

Set hyperparameters: learning rate η, decay rates γ_1, γ_2 (typically γ_1 = 0.9, γ_2 = 0.999), and a small constant $\epsilon = 10^{-8}$ to prevent division by zero.

For each iteration t:

the gradient:

$$g_t = \nabla L(\theta_t) \quad (2.52)$$

Update biased first moment estimate:

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) g_t \quad (2.53)$$

Update biased second moment estimate:

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) g_t^2 \quad (2.54)$$

Compute bias-corrected first moment estimate:

$$\mathbf{m}_t = \frac{\mathbf{m}_t}{1-\beta_1^t} \quad (2.55)$$

Compute bias-corrected second moment estimate:

$$\mathbf{v}_t = \frac{\mathbf{v}_t}{1-\beta_2^t} \quad (2.56)$$

Update parameters:

$$\theta_{t+1} = \theta_t - \eta \frac{\widehat{\mathbf{m}_t}}{\sqrt{\widehat{\mathbf{v}_t}} + \epsilon} \quad (2.57)$$

Compare with RMSProp, Adam is computationally efficient and requires less memory.

The combination of momentum and adaptive learning rates leads to fast and reliable convergence, and it is less sensitive to the initial learning rate and other hyperparameters compared to other optimization algorithms. However, if the learning rate is not tuned properly in this algorithm, it can sometimes lead to overfitting.

2.3.1.8 Network Architectures

The network architecture refers to the structure and organization of NN, also known as its topology. Neuronal interconnections can be made in various ways, leading to many different topologies such as perceptron model (P), feed forward model (FF), radial basis network (RBF), RNN, LSTM, Hopfield network (HN), Boltzmann machine NN (BM) and deep feed forward model (DFF) [44]. These different topologies are depicted in Figure 2.19.

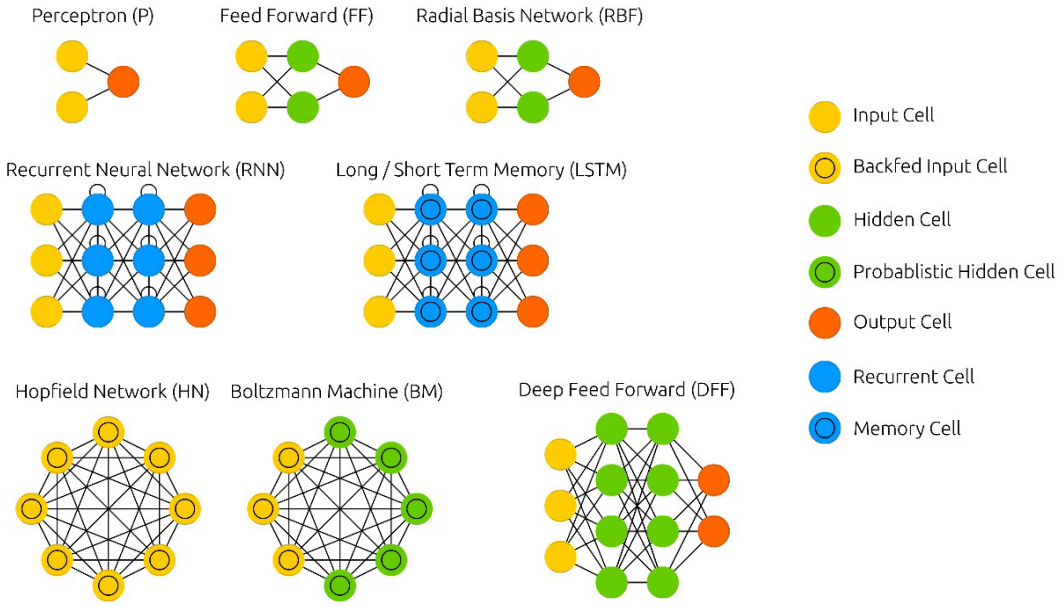


Figure 2.19: Different architectures of NNs [45]

NN with feed-forward topology is called feed-forward NN and as such has only one condition: information must flow from input to output in only one direction with no back-loops [37]. It means, in this topology there is no back propagation. The simplest feed-forward NN is a single perceptron that is only capable of learning linear separable problems.

NN with the recurrent topology is called RNN. In these cases, information is no longer transmitted only in one direction, but it is also transmitted backwards during the back loops of the topology. The simple NN shown in Figure 2.17 is a fully recurrent NN, characterized by the complexity of its interconnections between neurons. The most basic topology of NN is fully RNN where every node (neuron) is directly connected to each other in all direction. The NN topology such as HN, LSTM, bi-directional and other networks are just exceptional cases of RNNs.

In the above, basic concepts of NNs are described. In the next section, the theoretical basis of using NNs for classification problems will be explained.

2.3.2 Multi-Class Classification

The objective of multi-class classification is to classify instances into more than two categories. Operating points of the centrifugal pump is in this work viewed as multiple

categories. In multi-classification problems, there are a variety of strategies that can be used to handle classification tasks for multiple categories:

One-vs-All (OvA) or One-vs-Rest (OvR)

This strategy breaks down a multi-class problem into multiple binary classification problems [46]. Each classifier is responsible for distinguishing one class from the rest. For a multi-class problem with K classes, train K binary classifiers $f_k(x)$, each solving the following binary classification problem:

$$f_k(x) = \begin{cases} 1 & \text{if } x \text{ blongs to class } k \\ 0 & \text{if } x \text{ not blongs to class } k \end{cases} \quad (2.58)$$

The final classification result is determined by selecting the classifier with the highest score:

$$\hat{y} = \arg\max_k f_k(x) \quad (2.59)$$

One-vs-One (OvO)

This strategy trains a classifier for each pair of classes, responsible for distinguishing between those two classes. For K classes, $\frac{K(K-1)}{2}$ classifiers are trained. For each pair of classes (i, j), train a classifier $f_{ij}(x)$ with the following output:

$$f_{ij}(x) = \begin{cases} i & \text{if } x \text{ blongs to class } i \\ j & \text{if } x \text{ blongs to class } j \end{cases} \quad (2.60)$$

The final classification result is decided by voting, choosing the class with the most votes:

$$\hat{y} = \arg\max_k \sum_{i \neq j} 1(f_{ij}(x) = k) \quad (2.61)$$

Multinomial Classification

This strategy uses a single model to directly output one of the K classes. Common models include SoftMax regression and NNs. The model outputs a probability distribution over K classes [20]:

$$P(y = k|x) = \frac{\exp(\theta_k^{\mathrm{T}} x)}{\sum_{j=1}^{K} \exp(\theta_j^{\mathrm{T}} x)} \quad (2.62)$$

where x is an input, θ_k is the parameter vector for class k. The final classification result is the class with the highest probability:

$$\hat{y} = \arg\max_k P(y = k|x) \quad (2.63)$$

A schematic diagram of these strategies is shown in Figure 2.20， It can be clearly seen that compared with OvA or OvO, the multinomial classification strategy simplifies the model by handling all classes within a single framework, eliminating the need to decompose the problem into multiple binary classification tasks. This reduces complexity in implementation and maintenance. In many real-world scenarios, class distributions may be imbalanced. Multinomial classification naturally handles class imbalances without requiring additional techniques or adjustments. It offers several advantages including simplicity in implementation, optimized model objectives, accurate probability estimation, robust handling of class imbalances, and scalability for large and complex multi-class classification tasks [47].

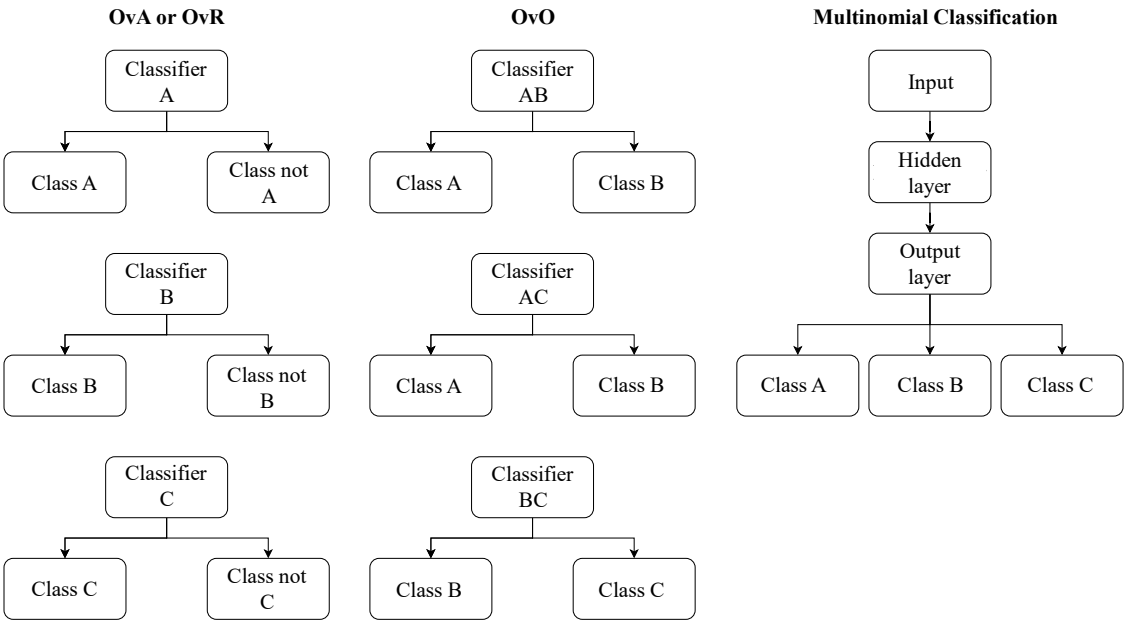


Figure 2.20: Multi-class classification strategies

DNNs are particularly powerful for multi-class classification due to their ability to learn complex, non-linear feature representations [13]. A typical DNN used for this task consists of multiple layers of neurons, including fully connected layers and potentially convolutional or recurrent layers, depending on the nature of the input data. The final layer of the network usually employs a SoftMax activation function to produce a probability distribution over the classes. As mentioned in Section 2.3.1.3, the more hidden

layers there are, the deeper the network is. In this dissertation, the applicability of a simple NN with only a single hidden layer for classify the operating points by vibration signals of a centrifugal pump was investigated.

2.3.3 Neural Network Classifier

Figure 2.21 shows a schematic diagram of the training process of a NN classifier.

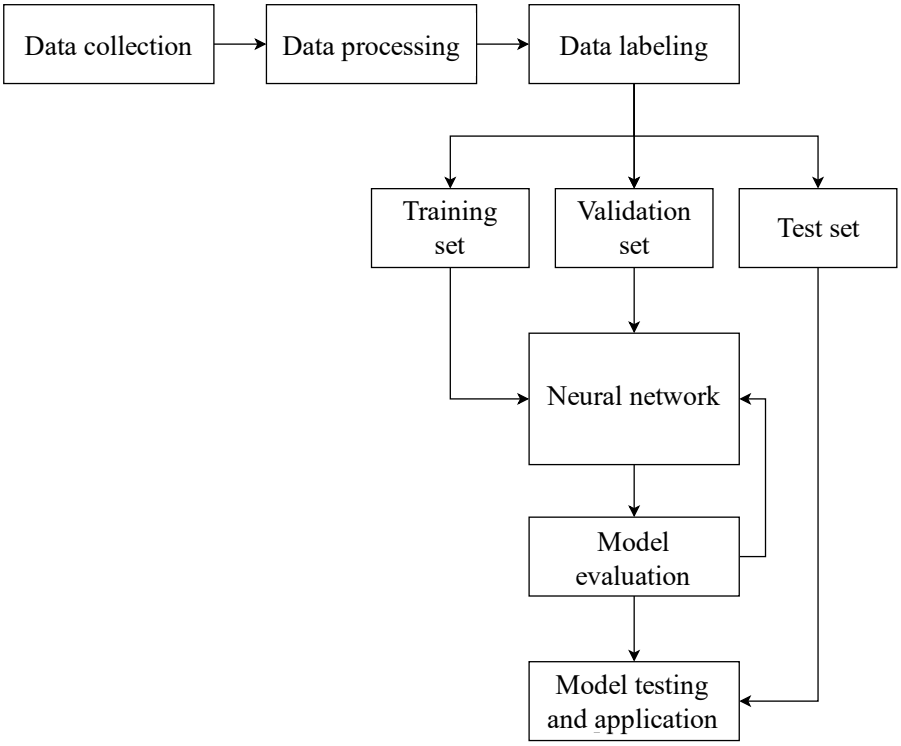


Figure 2.21: NN classifier training flow chart

The data collection and processing are in Section 2.2 in detail described. The data labeling in this work involves adding the operating point classification to each collected vibration signal (see Figure 2.22). Each label/class corresponds to an operation point (e.g., seven flow ranges) and each class corresponds to multiple vibration signals.

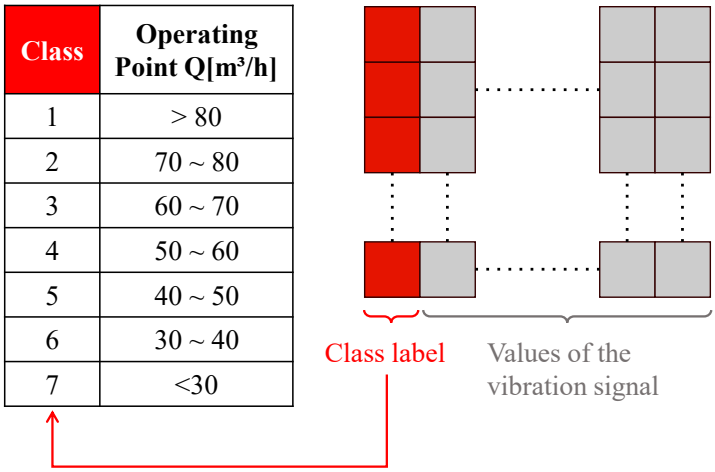


Figure 2.22: Data labelling for NN model training

Training a NN model typically involves using a training set, a validation set, and a test set to evaluate the model's performance and generalization ability [13].

The training set is the primary dataset used to train the NN model. It contains a large number of labeled examples, and the model learns and optimizes its parameters by processing data from this set. The training set should ideally cover various input data scenarios to ensure the model learns patterns and features effectively.

The validation set is used to assess the model's performance and adjust hyperparameters during the training process. At the end of each training epoch, the model's performance is evaluated on the validation set to monitor its effectiveness. This helps prevent overfitting, where the model performs well on the training data but poorly on unseen data. Typically, the validation set is separated from the training set and does not participate in model parameter training.

The test set is reserved for the final evaluation of the model's generalization ability. This dataset consists of independent data that the model has not encountered during training or validation. The purpose of the test set is to simulate the model's performance in real-world applications, providing an assessment of its accuracy and performance on unseen data. Results from the test set are used to report and compare different models' effectiveness, ensuring the model's reliability in practical applications.

In practice, the method to divide dataset is Random Splitting. The dataset is randomly divided into training, validation, and test sets. For instance, a common split as example might be 70% for training, and 15% each for validation and testing or the other proportion. When splitting dataset, it's crucial to maintain consistent feature distributions across sets

to avoid bias [48]. This ensures more accurate evaluation of the model's performance and generalization capabilities, facilitating further model tuning and optimization.

The iteration process inside the NN is described in Section 2.3.1.5. To evaluate the performance of a NN classifier, the evaluation metrics such as Accuracy, Precision, Recall, F1-score and Confusion Matrix are used [41]. The Accuracy indicates the proportion of instances that are correctly classified. The Confusion Matrix is a table showing the true vs. predicted classifications. For imbalanced dataset, Accuracy may not be sufficient to fully reflect the performance of the model. In this case, indicators such as Precision, Recall, and F1-score can provide more detailed evaluation.

2.4 Programming Languages

NN classifiers can be implemented using a variety of programming languages and frameworks. Popular choices include Python, C++, MATLAB, Java etc. [40].

Python is the most widely used language due to its extensive machine learning and deep learning library support, such as TensorFlow, PyTorch, Keras, and scikit-learn. TensorFlow and PyTorch are two of the most popular deep learning frameworks that provide rich NN layers and optimizers for implementing various types of NN classifiers.

C++ is a high-performance language suitable for implementing large-scale NN models. Libraries like Caffe and TensorFlow C++ application programming interface (API) can be used for implementing NNs in C++.

MATLAB provides the Deep Learning Toolbox, which can be used for rapid prototyping and training of NN models, but it typically has slower computation speeds.

In this dissertation, Python is used to build the NN classifier. The libraries TensorFlow and Keras are employed for programming. Based on the experimental requirements, a model selection program was developed to identify the most effective NN configuration.

TensorFlow

TensorFlow is a powerful open-source software library for numerical computation, particularly well suited and fine-tuned for large-scale Machine Learning [40]. Most importantly, it is possible to break up the graph into several chunks and run them in parallel across multiple CPUs or GPUs (as shown in Figure 2.23). The central unit of data in TensorFlow is the tensor, which is a multi-dimensional array. Tensors are the building

blocks for defining and executing computations. TensorFlow can train networks with millions of parameters on training sets consisting of billions of instances, each with millions of features. Because TensorFlow was developed by the Google Brain team, it supports many of the major Google services, such as Google Cloud Speech, Google Photos, and Google Search.

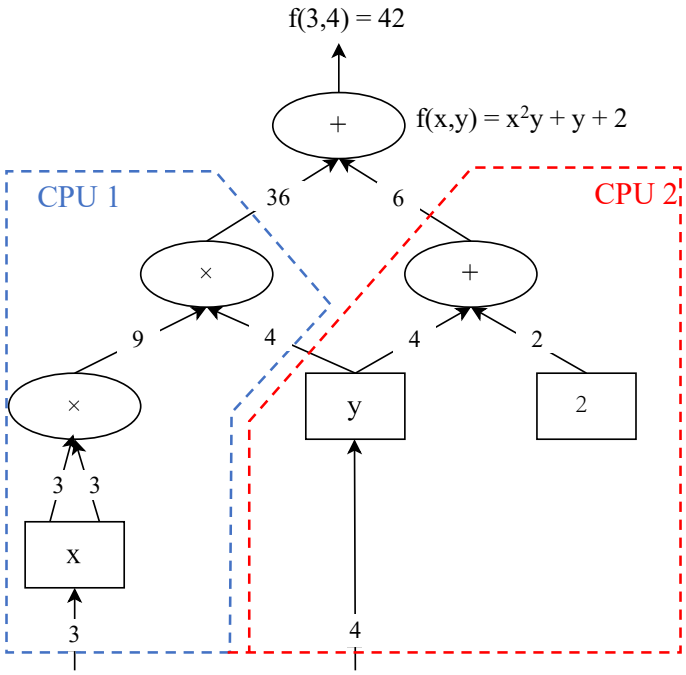


Figure 2.23: Parallel computation on multiple CPUs/GPUs/servers [40]

TensorFlow was released as open source in November 2015, there were already many popular open-source libraries for Deep Learning (Table 2.1 lists some) existed. However, TensorFlow's clean design, scalability, flexibility, and excellent documentation quickly pushed it to the top of the list.

Table 2.1: Open-source deep learning libraries [40]

Library	API	Platforms	Started by	Year
Caffe	Python, C++, Matlab	Linux, macOS, Windows	Y Jia, UC Berkeley (BVLC)	2013
Deeplearning4j	Java, Scala, Clojure	Linux, macOS, Windows, Android	A. Gibson, J.Patterson	2014
H2O	Python, R	Linux, macOS, Windows	H2O.ai	2014
MXNet	Python, C++, others	Linux, macOS, Windows, iOS, Android	DMLC	2015
TensorFlow	Python, C++	Linux, macOS, Windows, iOS, Android	Google	2015
Theano	Python	Linux, macOS, iOS	University of Montreal R. Collobert, K. Kavukcuoglu, C.	2010
Torch	C++, Lua	Linux, macOS, iOS, Android	Farabet	2002

TensorFlow runs not only on computers but also on mobile devices. It provides a simple API, allowing various types of NNs to be trained in just a few lines of code. Several high-level APIs based on TensorFlow, such as Keras and Pretty Tensor, are available. Programming with the main TensorFlow API is more flexible and allows for the creation of any NN architecture. It has also a C++ API to define higher performance computations. TensorFlow is widely used for time series forecasting, anomaly detection, and other temporal data analysis tasks.

Keras

As mentioned above, Keras is a high-level API that provides a Python interface for NNs and is designed to enable rapid experimentation with DNNs [49]. Keras models are

created by simply connecting configurable building blocks such as layers, optimizers, and activation functions. It includes many pre-built layers such as convolutional layers, recurrent layers, and fully connected layers. Keras has been a part of the TensorFlow library since TensorFlow 2.0. This integration provides a seamless experience for building and training models using TensorFlow's powerful compute engine.

In addition, two separate measurement and control programs of the test bench for collecting vibration signals have been developed, one using MATLAB and the other using Python.

3 Design of the Experimental

In order to obtain the training and test data for the NNs, a pump loop system was designed and constructed at the School of Intelligent Manufacturing at JHU. The measurement and control software were developed.

3.1 Design of the Experimental Setup

3.1.1 Test Bench

Initially, a test bench was designed based on the experimental requirements and the expertise of SAM in creating experimental setups. Figure 3.1 shows the 3D-representation of the test bench of JHU.

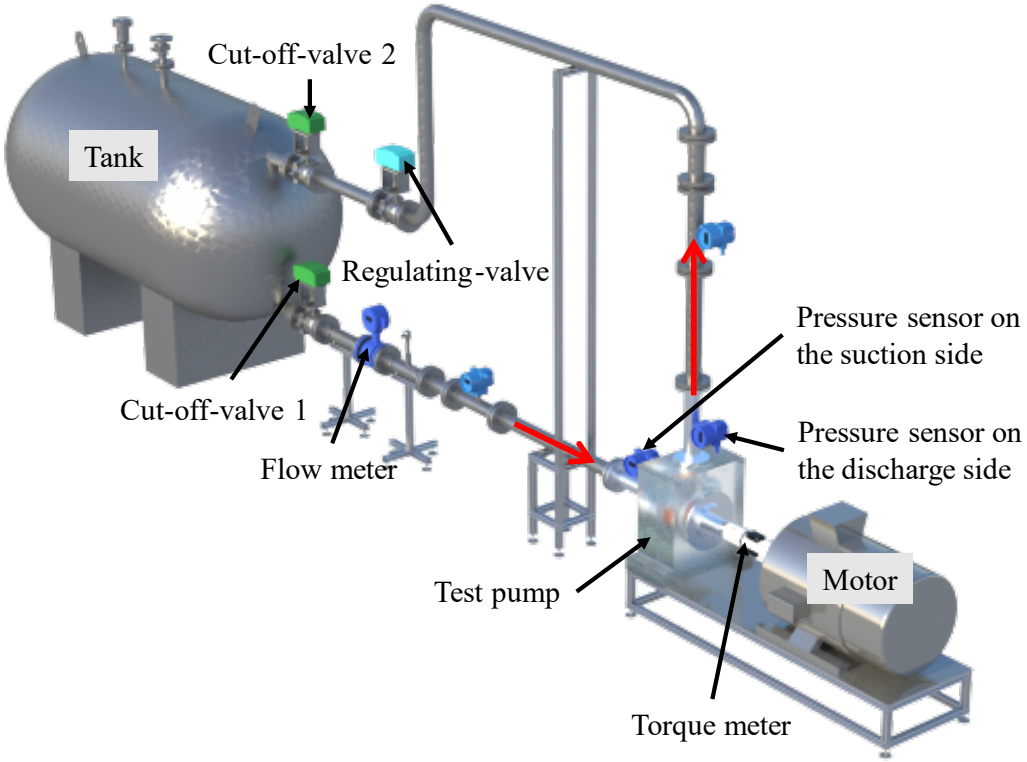


Figure 3.1: 3D-representation of the test bench of JHU

The core component of the test bench is an impeller pump with a replaceable impeller, which is designed and manufactured by SAM of RPTU. The specific speed of the pump Nq = 39, head H = 30 m, volume flow rate Q = 100 m^3/h, rated rotation speed n = 3000 rpm. The pump has 6 blades, blade_exit_angle (hub) = 22.8°, blade_exit_angle (middle)

= 24.4°, blade_exit_angle (shroud) = 25.9°. To enhance the compatibility of the test bench with various other experiments, the volute of the test pump is constructed from transparent acrylic. Figure 3.2 is a physical diagram depicting the connection between the test pump and motor drive.

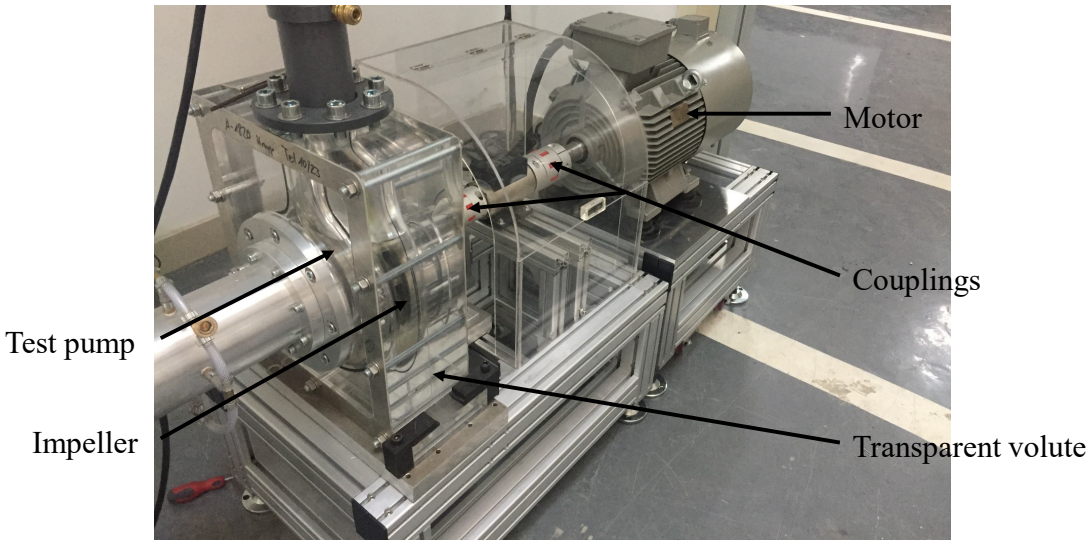


Figure 3.2: Drive components of the test pump of JHU

The pump is driven by an asynchronous motor with a rated rotation speed of 2950 rpm. The rotation speed of the motor is controlled by a frequency converter. Between the pump and the motor is a torque meter and the couplings.

The red arrows in Figure 3.1 indicate the flow direction. On the left of the test bench a 4000L water tank is installed. Two cut-off valves are next to the tank, which can cut off the tank from the pipe loop. A regulating-valve is next to the cut-off-valve 2 in order to regulate the flow volume Q in the loop. The flow volume can be measured by the flow meter below the regulating valve. Two pressure sensors are located on the suction and discharge side of the test pump in order to measure the static pressure increase ΔP_{static} in Equation (2.1)

$$\Delta P_{static} = P_{static,p} - P_{static,s} \quad (3.1)$$

All components except the pump are manufactured in China. A list of the models and characteristics of the individual instruments and sensors is given in Table 3.1.

Table 3.1: Electrical equipment list of the test bench

Equipment	Manufacturer	Model	Performance	
Tank	Denuoer Fluid Equipment (Wuhan) Co.	Customized	Materia	Stainless steels
			Pressure range	-0.900~2 bar
			Size	Φ1400·3000·12
			Volume	4000 L
			Accuracy	0.50%
Cut-off-valve	Tianjin Aipaike Fluid Auto-control Valve Co.	Aipaike-30	Method of control	Manual/Electrical
			Motor Power	45W
			Operating temperature	-30~80 ℃
			Output/input signal	Digital
			Power supply	220 V AC
Regulating valve	Tianjin Aipaike Fluid Auto-control Valve Co.	Aipaike-50	Torques	250 N·m
			Accuracy	0.50%
			Input signal	0~10 V
			Method of control	Manual/Electrical
			Motor Power	90 W
			Operating temperature	-30~80 ℃
			Output signal	4~20 mA
			Power supply	220V AC
Flowmeter	Jiangsu Huahai M & C Technology Co., Ltd.	HHD-80	Torques	500 N·m
			Range	9~180 m^3/h
			Accuracy	0.005
			Electrode material	316 L
			Fluid temperature	Room temperature
			Inner lining	Polychloroprene
			Nominal diameter	DN-80
			Output signal	4~20 mA
			Power supply	220V AC
Pressure sensor	Jiangsu Huahai M & C Technology Co., Ltd.	HPS401	Pressure class	0.6 Mpa
			Output signal	4~20 mA
			Power supply	24 V (DC)
			Pressure transmitter	Silicon
			Range	0~5 bar
			Threaded connection	M20·1.5

Torque meter	Beijing Sea Bohua technology Co., Ltd.	HCNJ-101	Output signal	4~20 mA
			Power supply	220 V (AC)
			Range	-50~50 Nm
			Rotation speed	0~6000 rpm
			Sampling Rate	5~15 kHz
Motor	Siemens AG Division Digital Factory	1LE0001-1DA3	Frequency	50 HZ
			Power factor	0.86
			Power supply	380 V
			Rated current	29.5 A
			Rated efficiency	90.30%
			Rated power	15 kw
			Rated rotation speed	2935
Frequency Converter	Siemens Numerical Control Ltd.Nanjing	Sinamics V20	Rated Output Power	15 kW
			Control Method	Electrical /Manual (control panel)
			Input signal	0~10 V
			Output current at 480 V (4 kHz/40 °C)	31 A
			Output frequency	0~550 Hz
			Power supply	380~480 V (AC) (±10%)
			Rated Frequency	50/60 Hz
			Rated Input Current	38.1 A
			Rated Output Current	31 A
			Tolerance	±1%

The medium in the pump loop is water, when the vibration signal is collected.

3.1.2 Control and Measuring Boxes

Two control and measuring boxes are designed in this dissertation. One is based on Arduino, programmed in MATLAB, and requires connection to a computer to control the test bench. The other is based on Raspberry Pi, programmed in Python, and only needs to

be connected to a monitor, a keyboard, and a mouse to control the test bench, without needing a computer. Both the Arduino Mega 2560 and Raspberry Pi are cost-effective and feature-rich options.

3.1.2.1 Arduino Based Control and Measuring Box

A self-designed measuring box base on Arduino-microcontroller-board Mega 2560 is used for the test bench to communicate with a computer (Figure 3.3).

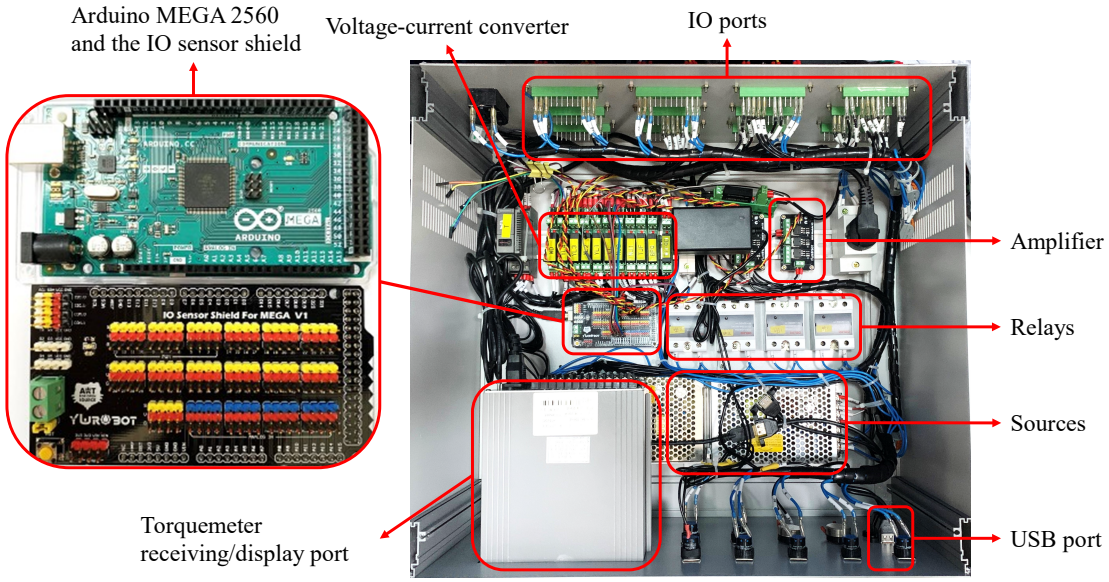


Figure 3.3: Control and measuring box based on Arduino MEGA 2560

The Arduino-microcontroller-board Mega 2560 is from the Arduino company [50] costs only €28. It is a microcontroller board based on the ATmega2560, has 54 digital input/output (IO) pins (of which 15 can be used as Pulse Width Modulation (PWM) outputs), 16 analog inputs, 4 hardware serial ports, a 16 MHz crystal oscillator, a USB connection, a DC power supply, an In-Circuit Serial Programming (ICSP) header, and a reset button [51]. To make the wiring more convenient, an IO port-sensor shield designed by YwRobot is connected to the board [52], it offers in addition two I^2C ports and two serial ports. The pin categories and connections of the Arduino board in the control box are shown in Figure 3.4.

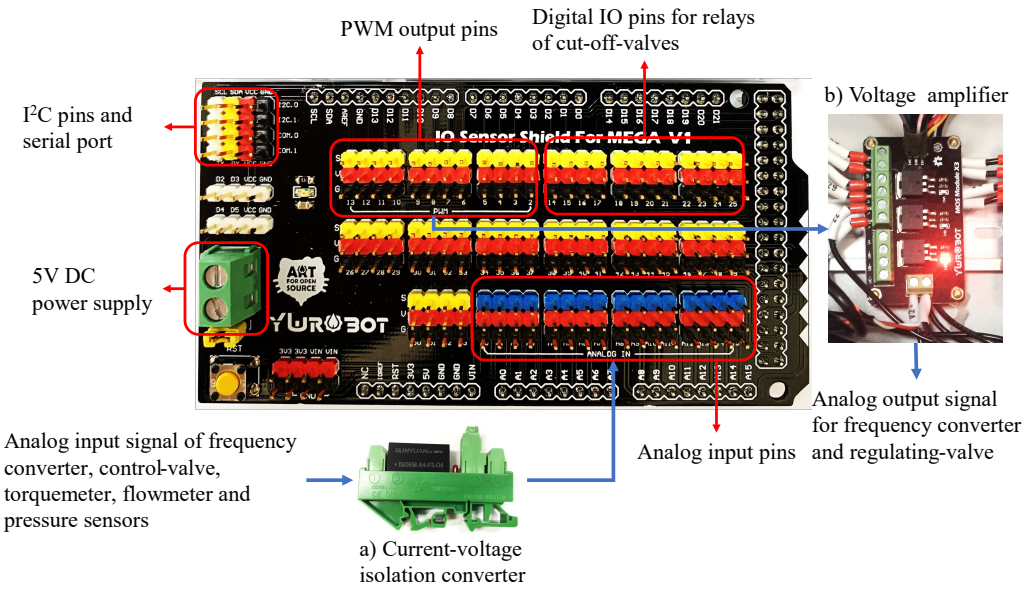


Figure 3.4: Control board pins assignment (Arduino)

According to Table 3.1, the cut-off-valves controlled by digital signals and connected with Solid State Relays (SSR) to switch the close or open status. The SSRs are controlled by logic high or low and connected to the digital IO pins of the board. The frequency converter controls the motor speed through a 0-10V analog signal, and the regulating-valve needs also a 0-10V voltage control signal. Since the Arduino is powered by 5V DC, the output voltage level tops out at about 5V. So that, a voltage amplifier is needed to amplify the PWM voltage, seen Figure 3.4 b). All feedback signals of the sensors and instruments are 4-20mA analog signals (to reduce the error caused by wire loss), it is necessary to use a current-voltage converter (Figure 3.4 a)) to convert the 4-20mA current signal into a 0-5V voltage signal, so that the board can read and record the signals. The performance of the electrical components in the control box are listed in Table 3.2.

Table 3.2: The electrical components in the control box (Arduino)

Electrical components	Manufacturer	Model	Performance	
SSR	Delixi	CDG1-1DA/25	AC voltage range	12~25 A, 480 V (AC)
			Control voltage range	3~32V (DC)
			Size	69.4·41.7 mm
Amplifier	YwRobot	MOS Module X3	Power supply	3.3 V, 5 V
			Output voltage	0~24 V
			Output load current	< 6 A
DC power supply	Delixi	CDKU-S200	Power supply	170~364V
			Output stability	≤ 1%
			Output voltage	5~48 V (DC)
			Number of output circuits	3
			Overload protection	110%~150%
			Overvoltage protection	110%~140%
			Operating temperature	-30~70 ℃
Converter	SUNYUAN SZ Technology	ISOEM A4-P3-O4	Input current range	4~20 mA
			Power supply	5 V
			Output voltage	0~5 V
			Operating temperature	-40~85 ℃

The panels of the box are shown in Figure 3.5, each instrument and sensor are connected to the control box through the multi-pin connectors on the back panel. The connector has a 7-pin circular panel, which includes a power supply pin, a Ground pin, a shield pin and

4 signal pins. It can simultaneously transmit power and signals and provides a stable mechanical connection that prevents accidental disconnection.

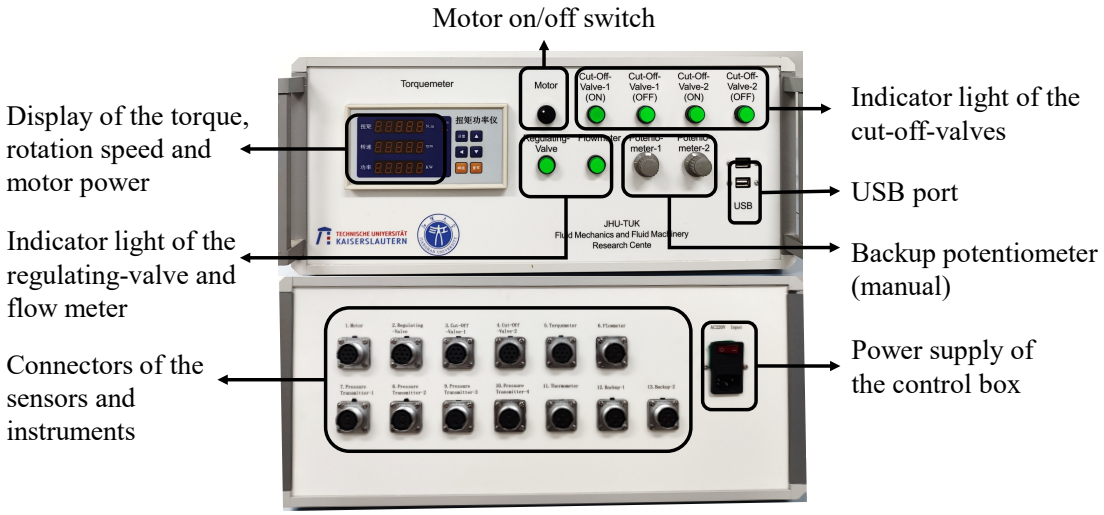


Figure 3.5: Control box panels diagram (Arduino)

Additionally, a motor “off/on” switch is designed on the front panel to facilitate easier and safer motor startup and shutdown. The USB ports and power cables are located on both the front and back panels.

3.1.2.2 Raspberry Pi Based Control and Measuring Box

To enhance portability and versatility in control and measurement, another control box based on the Raspberry Pi was designed. The key advantage of the Raspberry Pi in this case is that it functions as a standalone microcomputer and has 40 pin General-Purpose Input/Output (GPIO) header. It integrates the microcontroller-board with the basic functions of a computer and is only the size of a credit card.

The Raspberry Pi is based on Atmel’s ATmega644 microcontroller, developed by the Raspberry Pi Foundation, a UK-based charity, with Eben Upton as the project leader in 2012 [53]. It runs a variant of the Linux operating system, typically Raspberry Pi OS, and supports various programming languages such as Python, C++, and Java. The Raspberry Pi has been developed rapidly and is now at “Raspberry Pi 5”. But the box, designed in 2019, used a “Raspberry Pi 3 Model B”, which cost only 300 CNY (approximately €40). An IO expansion HAT from DFRobot is also used to reduce and stabilize the cabling. The layout of the box is shown in Figure 3.6.

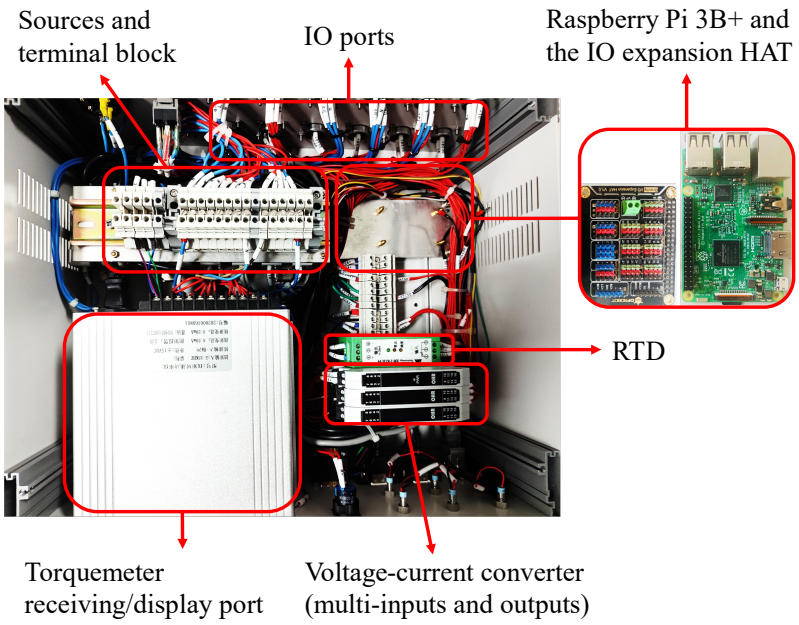


Figure 3.6: Control and measuring box base on Raspberry Pi

The top layer of the box is the Raspberry Pi with the IO expansion HAT plugged in, which has been removed separately to make it possible to see the internals. The box is much simpler than the Arduino setup. The converter has been replaced with a multi-input, multi-output voltage-current converter. A resistance temperature detector (RTD) has been added, the relays are now arranged vertically below the torque meter display and the amplifier has been removed, resulting in a reduced volume of the box.

The pins assignment of Raspberry Pi with the IO expansion HAT is showed in Figure 3.7, the function and connection of its IO pins are basically the same as the Arduino's, but Raspberry Pi as a microcomputer has peripheral interfaces.

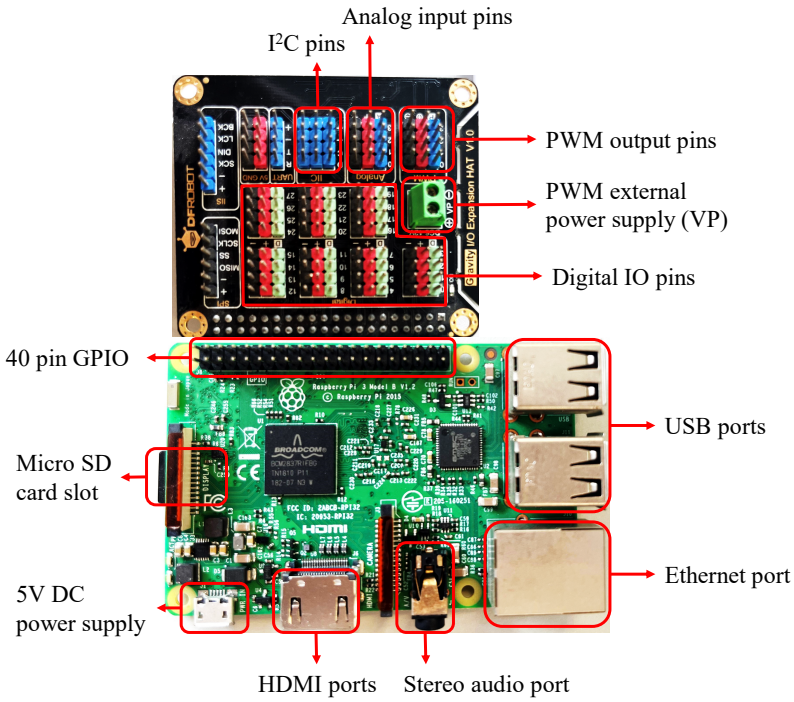


Figure 3.7: Control board pins assignment (Raspberry Pi)

The “Raspberry Pi 3 Model B” is the earliest model of the third generation. It has a 64-bit CPU and 1GB of RAM. Four USB 2.0 ports are available for connecting a keyboard, a mouse, and other USB interface devices. The full-size HDMI port is used for connecting a display. The Raspberry Pi’s operating system is installed on a micro-SD card, which is plugged into the board through the card slot. The Raspberry Pi itself only needs a 5V DC power supply, and the HAT’s VP port can be connected to a 6-12V DC power supply. When the VP port is not powered by an external source, the PWM voltage is maximum 5V. When the VP port is powered by an external source, the maximum PWM voltage is equal to the VP external power (6-12V). Therefore, the amplifier can be removed.

The panels of the box have changed a little. The back panel is basically the same as the Arduino’s, the changes are mainly on the front panel (Figure 3.8).

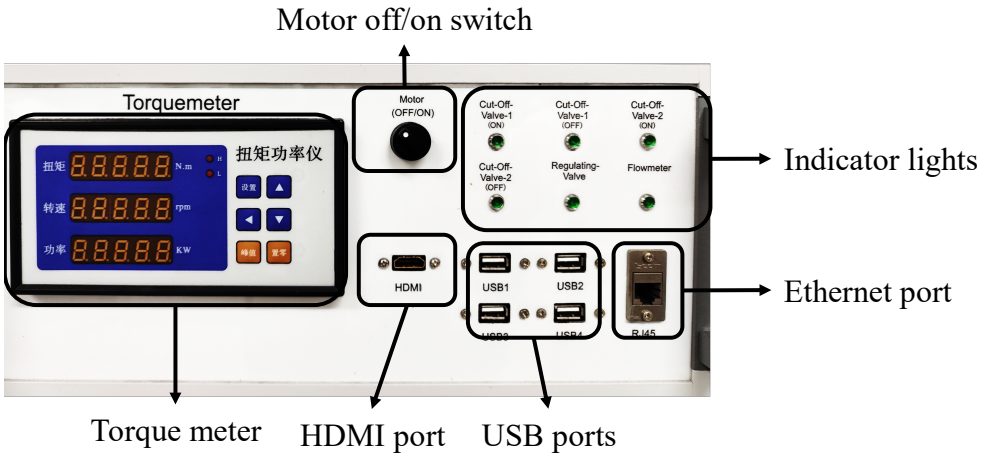


Figure 3.8: Control box front panel diagram (Raspberry Pi)

Because of the overall reduction of the box, the indicator lights were replaced with a smaller size, and the ports for the ethernet, USBs and display cable were added.

3.1.3 ADXL345 Accelerometer

As shown in Figure 3.9, the vibration signal of the test pump was taken by the three-axis accelerometer ADXL345, which is glued on the transparent casing of the test pump.

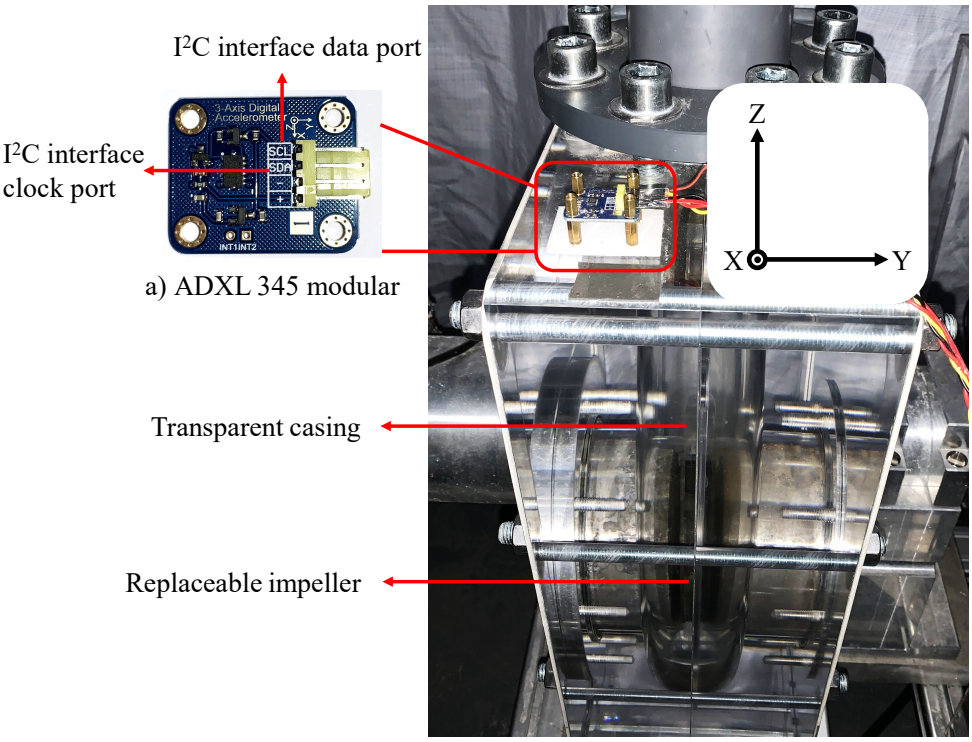


Figure 3.9: Vibration signal acquisition of the test pump

This encapsulated modular (Figure 3.9 a)) is from ALSROBOT Inc., the dimension of the modular is 30mm×25mm and it weighs only 4gram. Levelling can be adjusted by the four support feet. The main chip in the center of the modular is the accelerometer sensor ADXL345.

The ADXL345 is manufactured by Analog Devices, Inc., a well-known company specializing in the design and manufacturing of analog, mixed-signal, and DSP integrated circuits (ICs) used in various electronic equipment. It is a widely utilized, low-power, three-axis accelerometer with high-resolution (13-bit) measurement capabilities up to ±16g [54]. Its digital output data can be accessed via I^2C or SPI interfaces, ensuring easy integration with a variety of microcontrollers and embedded systems.

One of the distinguishing features of the ADXL345 is its versatility in range selection, allowing users to configure the accelerometer for ±2g, ±4g, ±8g, or ±16g scales, thus optimizing performance for specific application needs. Operating within a supply voltage range of 2.0V to 3.6V and consuming minimal power (as low as 23 μA in measurement mode), the ADXL345 is suitable for battery-powered applications. Its compact 3mm×5mm×1mm package further facilitates integration into space-constrained designs. The accelerometer also supports a high data output bandwidth of up to 3200 Hz, making it ideal for high-speed, real-time data acquisition systems. The main performance parameters of ADXL345 are shown in the following Table 3.3.

Table 3.3: Specifications of ADXL345 [54]

	Unit	Detail
Interface voltage range (VDD I/O)	V	1.7~VS
Measurement range	g	±2, ±4, ±8, ±16
Noise X-, Y-Axes	LSB·rms	0.75
Noise Z-Axis	LSB·rms	1.1
Nonlinearity	/	±0.5%
Operating temperature range	°C	-40 ~85
Operating voltage range (VS)	V	2~3.6
Output data rate (ODR)	Hz	0.1~3200
Resolution	g	0.0039, 0.0078, 0.0156, 0.0312
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	LSB/g	230 ~
Sensitivity changes due to temperature	/	±0.01%

The collected data will be communicated with the microcontroller boards through the I^2C interface.

Based on the values estimated in Section 2.1.2, a range of ±2g is chosen, which means that the range is selected as ±19.6m/s^2. The raw reading of the ADXL345 in the ±2g range is a digital value in the range of -256 to 255, which needs to be converted to an actual acceleration value by the resolution of the sensor by

$$a_{acual} = digital\ value \cdot Resolution \cdot 9.81 m/s^2 \quad (3.2)$$

These steps are performed in the program that acquires the vibration signal. The NN calculations require an acceleration profile and do not require an accurate reading of the actual acceleration value, so no accurate sensor calibration is performed.

The ADXL345 can detect acceleration signals in three directions, as shown in Figure 3.9. This dissertation utilizes the acceleration signal in the z-axis direction, which is perpendicular to the ground, as the feature signal for detecting operating points. The signals from the other two directions are parallel to the ground. Given that the centrifugal pump is horizontally fixed by the water pipe and motor couplings and the impeller's

displacement in the horizontal direction is minimal. Consequently, only the z-axis acceleration is considered. As long as the sensor's z-axis remains perpendicular to the ground and the sensor is affixed to the top of the centrifugal pump's transparent casing, the required vibration signals will be correctly detected.

The actual measurement frequency in this dissertation is only about 500 Hz (2ms average interval between two data samples taken at 3200Hz sampling rate) which means that the Nyquist Frequency is 250Hz. Of course there are transducers with higher actual measurement frequencies, but the packaged ADXL345 modular costs only 30 CNY (approximately €4) each piece. The results show that, at this sampling rate, the NN classifier for the operating points demonstrates reliable performance, the detailed results are also shown in Chapter 4.

This implies that, excluding the control and test systems used for traditional operating point detection, the cost of the hardware system for vibration signal acquisition alone is less than €60.

3.2 Software Development

In this dissertation, two sets of programs are designed for measurement. One set is based on Arduino and written in MATLAB. The other set is based on Raspberry Pi, written in Python. Both programs have the same function, which is to control the electrical valves and motor automatically of the test bench, read the status of each instrument, record the test data and display the characteristic curves of the test pump by traditional method base on Equation (2.1). The flowchart of the measurement and control software is shown in Figure 3.10.

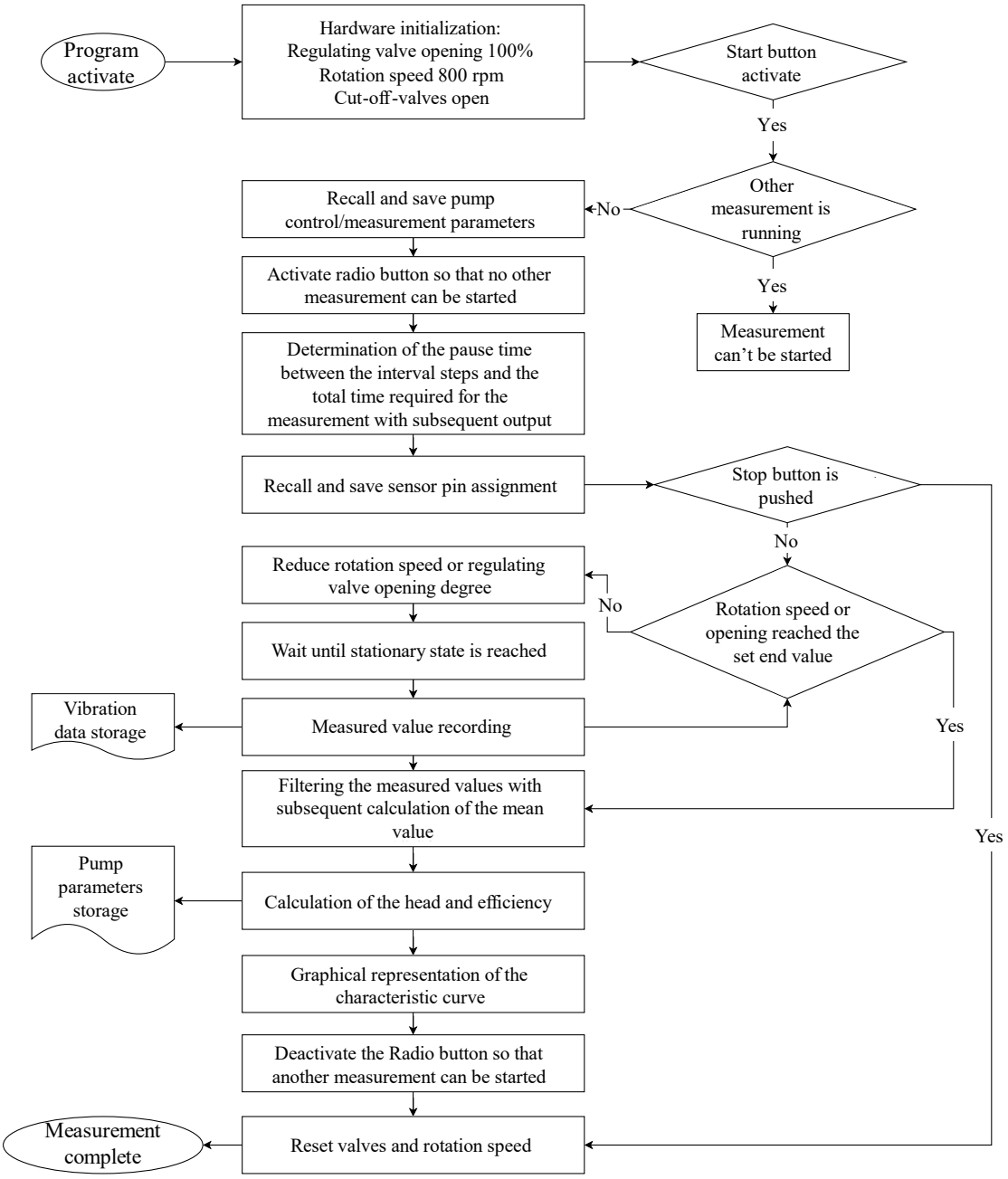


Figure 3.10: Flow chart of the control and measurement software

3.2.1 Test Bench Control and Measurement Interface (MATLAB)

After uploading the MATLAB driver on the Arduino, with the MATLAB Support Package for Arduino Hardware, the GUIDE Toolbox, the Instrument Control Toolbox, and MATLAB Plotting Functions, a software based on the Mega 2560 board was programmed. The pins of the board are connected to the hardware of the test bench as shown in Table 3.4.

Table 3.4: Pin mapping of the program

Function	Pin-Module	Pins
Cut-off-valve 1 ON	Digital output	D25
Cut-off-valve 1 OFF	Digital output	D23
Cut-off-valve 2 ON	Digital output	D21
Cut-off-valve 2 OFF	Digital output	D19
Regulating valve	PWM	D2
Frequency converter ON	Digital output	D17
Frequency converter	PWM	D3
Pressure sensor $P_{static,p}$	Analog input	A1
Pressure sensor $P_{static,s}$	Analog input	A2
Flow rate Q	Analog input	A3
Speed rotation n	Analog input	A4
Torque	Analog input	A5
Vibration	I^2C	SCL/SDA

Figure 3.11 shows the graphical user interface (UI) of the developed control program.

Figure 3.11: Graphical user interface of the control program (MATLAB)

In the program, a connection to the Arduino board of the control box must first be established. Before starting each experiment, the motor should be first switch to “on” status, two cut-off-valves should be opened, the initial value of the control valves should normally be set to 100%, and the motor rotation speed should be set to 800 rpm in order to warm up the test bench. These steps can be done via the initialization part on the program interface.

Individual operating states can then be set on the modification part by adjusting the start value, stop value and step size of the rotation speed or the regulating valve opening degree. The “determine” button can be used to determine the characteristic values of the respective operating state, the vibration signals at each operating point are also recorded. Furthermore, the “determine pump curve” function field can be used to generate pump and system characteristic curves for each experiment.

To close the program and turn off the test bench, simply press the “close program” button, switch the motor knob on the front panel to “off”, and then confirm that the program is closed. The entire program code can be found in Appendixes.

3.2.2 Test Bench Control and Measurement Interface (Python)

The Raspberry Pi-based UI is an application created directly on the Raspberry Pi and utilizes the extensive library resources that Python provides, such as Matplotlib for plotting, PyQt5 for creating the UI, NumPy for numerical computations, DFRobot_RaspberryPi_Expansion_Board for interfacing with the DFRobot Expansion Board, and RPi.GPIO for controlling the GPIO pins on the Raspberry Pi. The pin mapping of the program is the same as the box based on Arduino. The graphical UI is shown Figure 3.12.

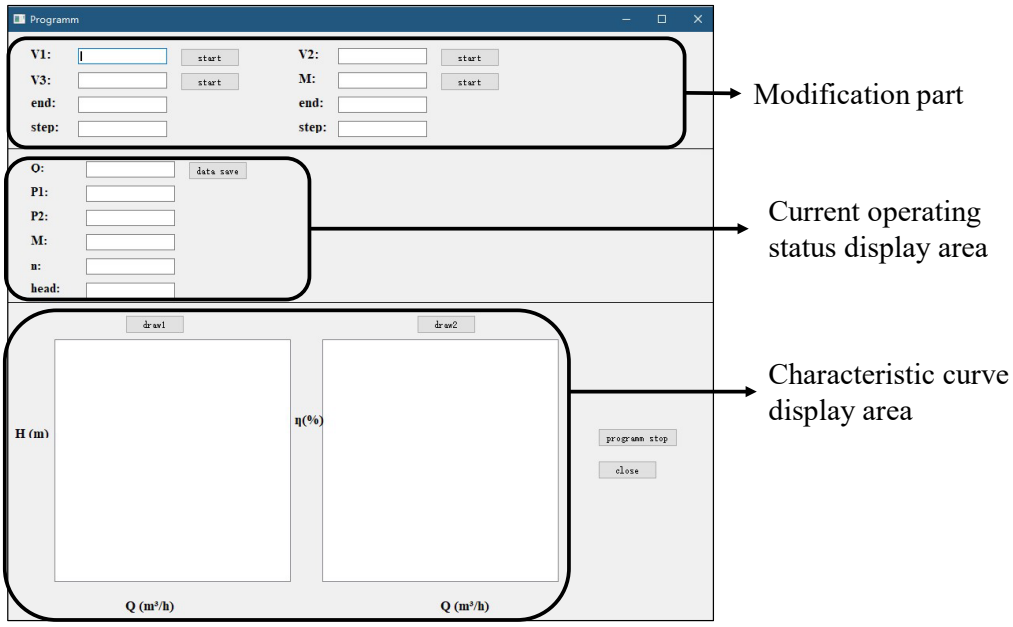


Figure 3.12: Graphical user interface of the control program (Python)

Because the program runs directly on the Raspberry Pi, there is no need to connect the computer like the program based on Arduino. The other steps of the operation are the same as the program written by MATLAB. The entire program code can be found in Appendixes.

In the above two programs, the vibration signals at each operating point are automatically recorded in each experiment. The acceleration range and sampling rate can be realized by uploading a different program on the Arduino or by changing the code in the Raspberry Pi program.

3.3 Description of the Test Process

3.3.1 Acquisition of Vibration Signals

The control and measurement boxes should be calibrated before the acquisition process. Because the test bench was designed with the need for calibration in mind, all instruments are equipped with a display to allow for data calibration after the programming is completed. Rotation speed calibration is accomplished with a tachometer and the torque meter. The data of calibration are shown in the Appendixes.

Before starting an automatic measurement, the operator must make various settings (see Figure 3.13).

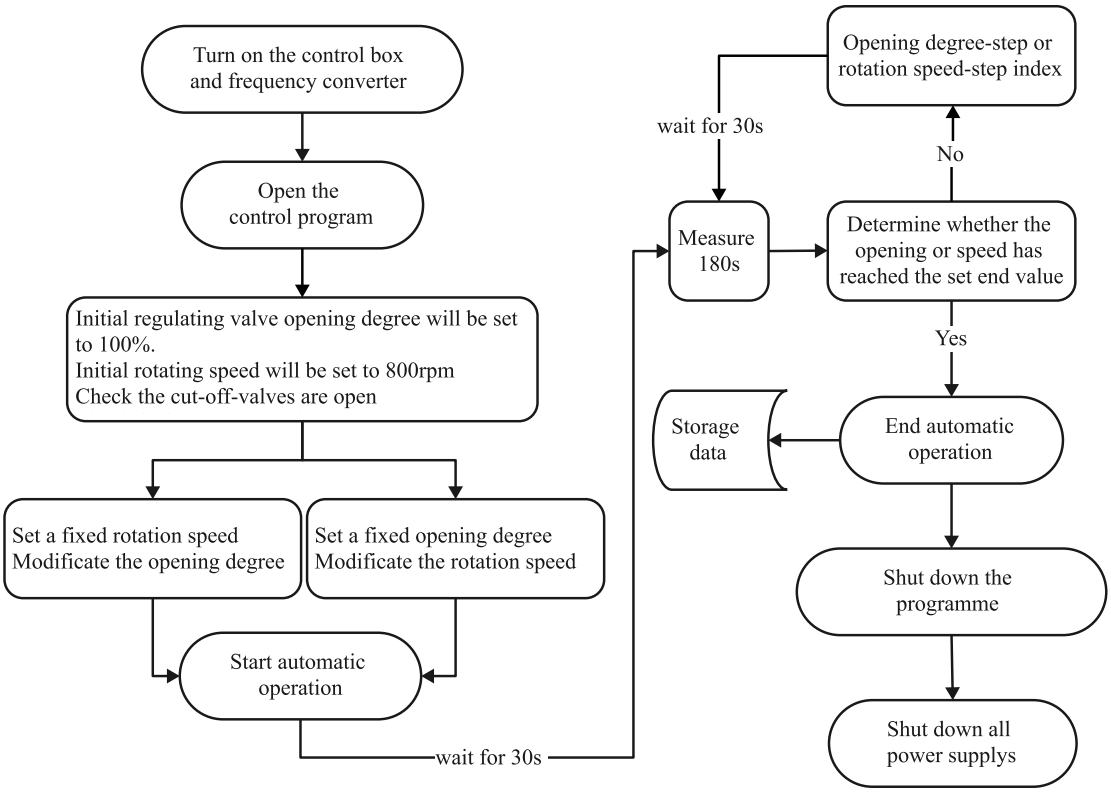


Figure 3.13: Settings of automatic measurement and control program

As described in Section 3.2, the sampling rate of the accelerator, the measuring time of each operating point, the waiting time after adjusting the operating point and the file storage location are set in the code or in the driver of the control board before the start of the automatic program. In the UI, the start value, the decrease step width and the end value of the valve opening, or rotation speed must also be specified. Table 3.5 below summarizes all the settings for the test.

Table 3.5：Experiment settings of the test bench

	f_s	n_p step width	Opening degree step width	Range of n_p	Range of the opening degree	Range of Q
Unit	Hz	rpm	/	rpm	/	m^3/h
Value	3200	200/300	5%	650~2500	60%~100%	10~110

The measurement duration of each operating point is 180s to ensure that the data is valid, and the storage is not too large. The flow rate from the flow meter will be averaged and recorded as one of the markers of this operating point. Different step sizes were used to

adjust the rotational speed and valve opening degree, which allowed for the collection of vibration signals at various operating points. All pump character curves were calculated with speeds from 1000 rpm to 2500 rpm, so the tests below 1000 rpm will be relatively fewer. The flow rate ranges at each rotation speed are shown in the Table 3.6

Table 3.6：Flow rate ranges at each rotation speed

	Rotation speed (rpm)							
	1000	1200	1500	1800	2000	2200	2300	2500
Rang of Q (m^3/h)	30.0, 35.6	34.0, 44.9	48.6, 58.4	60.8, 73.5	66.9, 81.1	67.2, 90.5	86.6, 94.3	101,3, 104.1

The range of flow rates below 1000 rpm is not listed because all data below 1000 rpm are considered as non-normal operating points and will not be used in the operating point detections.

3.3.2 Data Sets Processing

Once the vibration signals had been collected, a set of signals had to be analyzed as described in Section 2.2 before being used as training and test data in different NN topologies. The training results of NN are also affected by the method of data processing.

Each vibration signal is associated with one or two labels, representing the flow rate and rotation speed. The vibration signal serves as the feature data corresponding to these labels, forming a row of the data for NN-based analysis. The analysis and processing of a vibration signal at operating point Q = 104 m^3/h, n = 2500 rpm shows in Figure 3.14.

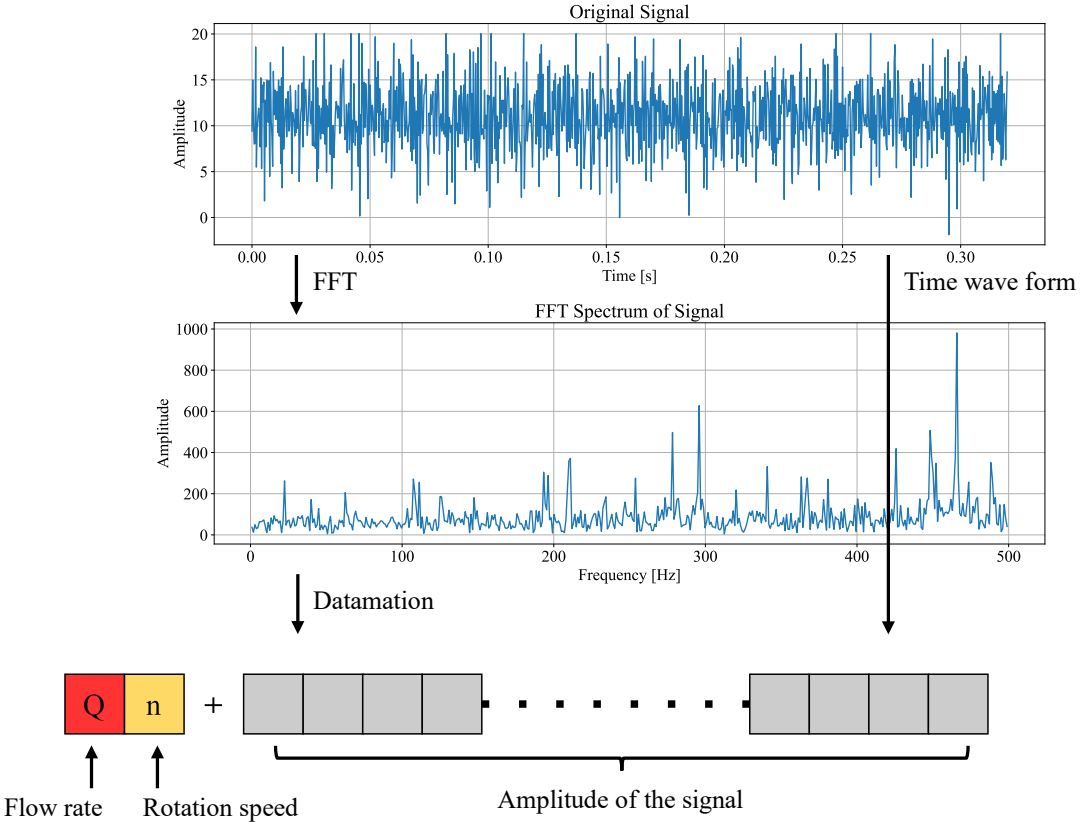


Figure 3.14: Vibration signal at Q = 104 m^3/h, n = 2500 rpm

After each vibration signal is transformed into feature data with labels and subsequently combined, the resulting data is the dataset of NNs. A processed dataset structure used to train and evaluation the NNs is shown in Figure 3.15.

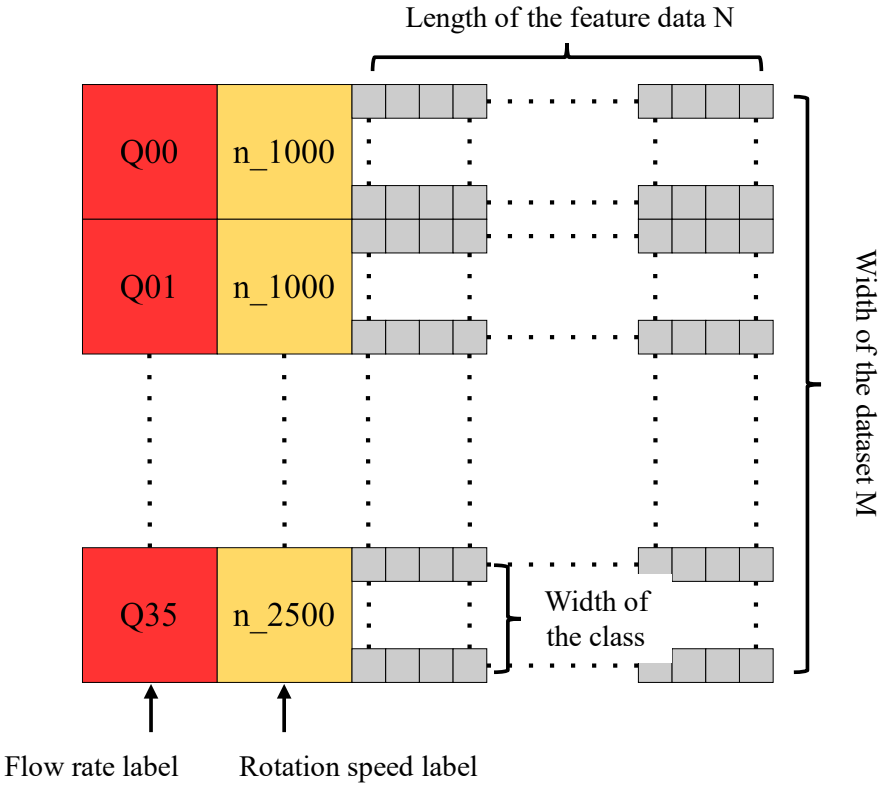


Figure 3.15: Data structure of a processed dataset

The dataset comprises M rows, where each row corresponds to a specific class and includes N feature data. Each class contains M_{Qi} rows feature data. The value of M_{Qi} represents the amount of feature data for this classification. The first column of the data is the classification to which the flow rate belongs, and the second column is the classification to which the rotation speed belongs. If the classification is by flow rate or rotation speed alone, there is only one column at the beginning of the data. The dimension of the whole dataset is described as M×N and will have different settings in the training experiment. The larger N is, the more complex the feature data becomes. As mentioned in Section 2.3, the dataset will be divided into training, validation and test sets, this step will be done randomly by the training and evaluation program under a given ratio. The settings for the prepare of the dataset in this dissertation are shown in Table 3.7.

Table 3.7 Data settings

Dimension of the dataset	N≥512 (Except PtP and RMS)
Selected vibration signals	Time Waveform, PtP and RMS, EMD, FFT
Sets ratios	(0.8, 0.1, 0.1), (0.7, 0.15, 0.15), (0.6, 0.2, 0.2)

In this dissertation, an attempt has been made to use multiple data dimensions in NN training, as data dimension has a significant impact on computational speed. It is shown that even a small data dimension can achieve a certain classification accuracy, the specific results will be presented in Chapter 4. As already mentioned in Section 2.2, both time-domain analysis method and frequency-domain analysis method will be used in the preparation of the dataset of the NN. For time waveform and EMD method, N is the length of the vibration signal in selected sampling period. For PtP and RMS, N = 2. To reduce the error in the FFT results, N is a power of 2.

After analysis, the data will be directly fed into the NN for training. For comparison purposes, another dataset, which has been cleaned using the Euclidean distance-based method, will also be used with the same network model for training. All data processing in this dissertation was performed using Python.

3.3.3 Setting up Different Neural Networks

The topology and hyperparameters of a NN are parameters that need to be set before the training process, and they have different impact on the performance of the classifier model. The NN topology used in this dissertation is shown in Figure 3.16.

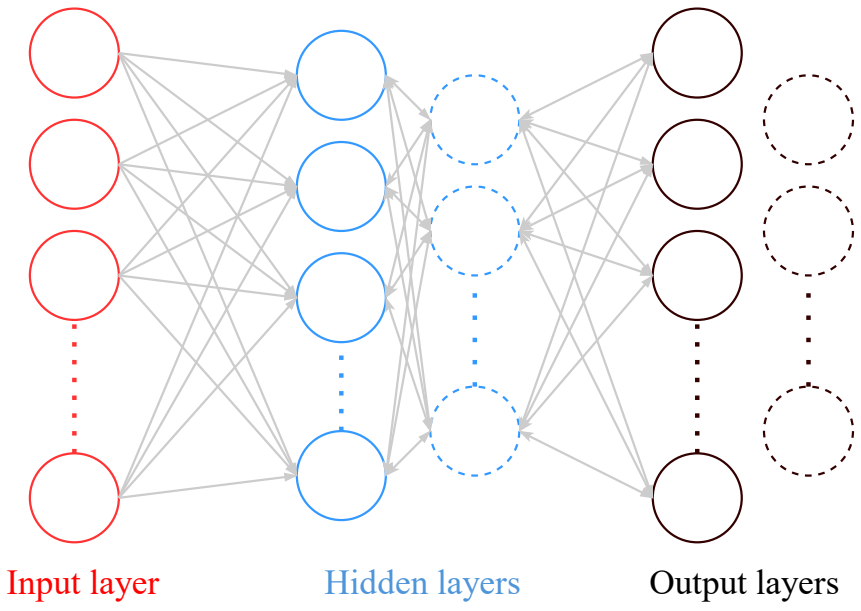


Figure 3.16: NN topology of the test process

The NN employed is a fully connected architecture, consisting of an input layer, 1 to 2 hidden layers, and 1 or 2 output layers. The hyperparameter settings of the NN are shown in Table 3.8.

Table 3.8: Hyperparameter settings for experiment

Hyperparameter	
Activation function	ReLU, Tanh, Sigmoid, SoftMax
Batch size	2~512
Dropout rate	None, 0.1, 0.2, 0.3, 0.4, 0.5
Early stopping	Validation set accuracy, epochs = 50
Initial learning rate	0.1~0.0005
Loss function	BCE, CCE
Optimizer	SGD, Adam, RMSprop, Adagrad

Dropout rate is to prevent overfitting, it randomly sets a fraction of the input units to zero at each update cycle during training, which helps the model to generalize better by preventing it from relying too heavily on any individual neuron. Batch size is the number of training examples used in an iteration. Early Stopping is to monitor the validation performance and stop the training if no improvement is observed over a certain number of epochs.

The most important results are explained in the next chapter. The terminology used in all test series is introduced first to provide a better understanding of the results.

4 Results

4.1 Process of Training and Evaluation

Before presenting the main results, the training and evaluation process will be briefly explained. First, the training and evaluation program is explained. Secondly, the nomenclature of the tests and evaluations must be explained in order to present the result clearly. Finally, the method of presenting and evaluating the results of all the experiments will be shown by means of some examples.

4.1.1 The Training and Evaluation Program

The object-oriented programming language Python was used to generate features and classify the vibration signals. The Evaluation Program is designed on PyCharm 2024.1.3 (Community Edition) IDE, which provides advanced features such as code completion, debugging, and integration with version control systems, streamlining the coding and testing processes, the software is completely free of charge. The TensorFlow based model mentioned in Section 2.4 assess various performance metrics by systematically modifying common hypotheses of the NN. The evaluation program entailed the testing of all possible combinations of the specified topologies and hypotheses, in addition to the selected dataset dimensions. After the NN has been trained on the training and validation sets, it will be tested on the test set. The training accuracy and validation accuracy at each epoch, the test accuracy and the Confusion Matrix are recorded as evaluation criteria for this model. The training and evaluation program is attached in Appendixes. The most significant results of each combination will be examined in the remainder of this chapter. First, the nomenclature of the different results will be explained. It is also necessary to explain in advance the format in which the results are presented.

4.1.2 Nomenclature of the Test Series and Evaluation

In the course of this study, the comparative analysis of diverse dataset and NN models was conducted. Firstly, the nomenclature of the tests and evaluations must be elucidated in order to facilitate the interpretation of the results. Table 4.1 summaries the test parameter abbreviations.

Table 4.1: Nomenclature of the experiments and evaluations

Measurement	
Flow rate class	Q00~Q35
Opening degree of the regulating valve	V60%~V100%, △V = 5%
Rotation speed class	n_rotaion speed in rpm(n_1500)
Dataset	
Dimensions of the data	Width (M) × Length (N)
Empirical Mode Decomposition	EMD
Fast Fourier Transforms	FFT
Time Waveform	TW
NN settings	
Activation Function	activation
Dropout Rate	dropout
Learning Rate	η
Loss Function	loss
Number of Epochs	epochs
Number of the hidden layer nodes	n_{hidden}
Number of the hidden layers	$n_{hidden\text{-}layer}$
Number of the input layer nodes	n_{input}
Number of the output layer nodes	n_{onput}
Optimizer	optimizer
Evaluations	
Accuracy on the test set of the NN	TA_{NN}
Accuracy on the training set of the NN	Acc_train
Accuracy on the validation set of the NN	Acc_val

Confusion Matrix	CM
The overall test accuracy of the classification	TA
The test accuracy of each class	TA_C

Furthermore, abbreviations for the signal processing methods, for the topology parameters and hyperparameters of the NN classifiers, and for the evaluation parameters are also provided.

This dissertation used three different methods to process the vibration signals and trained various NNs with the dataset processed by each method. The dimension of the dataset was also employed as test variables. The remaining variables have been explained in detail in Chapters 2 and 3.

4.1.3 Results Presentation and Evaluation

The result of the trained NN model includes first the accuracy at each epoch. The accuracy on the training and validation sets at each epoch by each NN model in this dissertation is presented in the form of Figure 4.1.

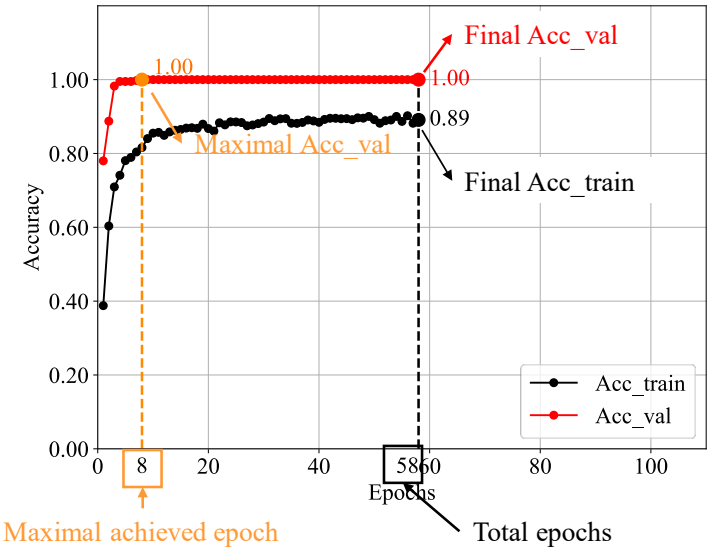


Figure 4.1: Accuracy per epoch of the NN model

The black curve shows the trend of Acc_train per epoch, while the red curve shows the Acc_val. The yellow point on the validation set curve marks the maximum value of accuracy on the validation set, and the epoch is marked on the horizontal coordinate. The final accuracy at the end of training is indicated at the conclusion of both curves, with the

total number of epochs marked on the horizontal axis. The evaluation program employs an early stopping rule to mitigate overfitting, terminating the training process if the Acc_train does not improve within 50 epochs. Consequently, the total number of epochs usually exceeds 50 more than the epoch when the maximum Acc_val is reached.

The results of multi-class classifications are usually presented in so-called confusion matrices. An example of such a matrix is shown in Figure 4.2. In this type of graph, the classes assigned by the classifier (horizontal) are plotted above the actual classes (vertical). This type of representation provides a good overview of all test results and enables quick identification of problem areas.

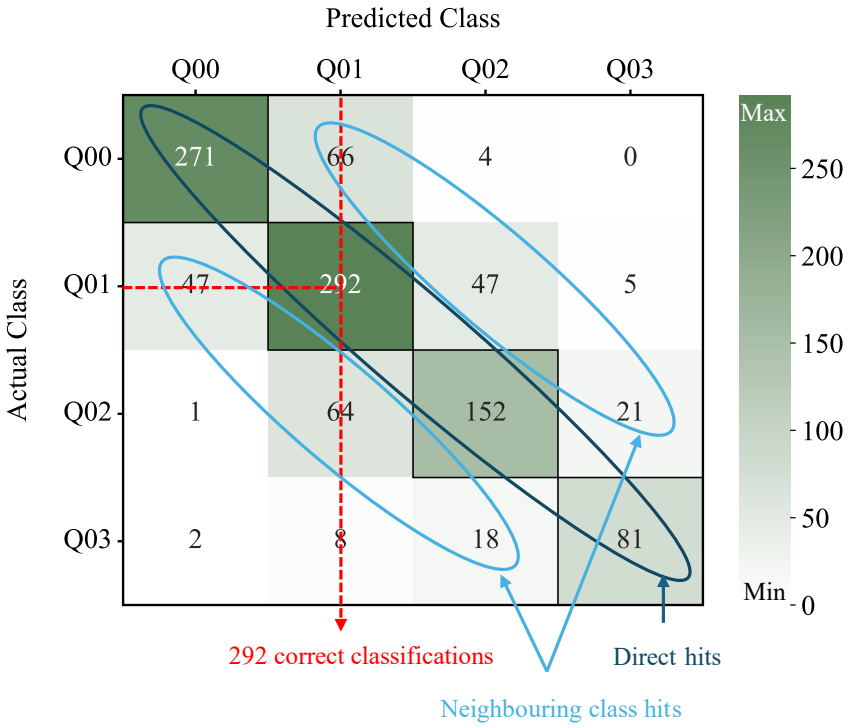


Figure 4.2: Confusion matrix with absolute classes division

The numbers along the diagonal from the top left to the bottom right represent the instances where the predicted class matches the actual class, indicating the direct hits. The numbers outside the diagonal indicate misclassifications. Each cell in the matrix represents the count of instances where a particular actual class was predicted as a different class. The values immediately adjacent to the diagonal can be interpreted as neighbouring class hits. For class Q01, for example, 292 of a total of 391 test data sets were correctly assigned, 47 test data sets were assigned to Q00, 47 were assigned to Q02 and 5 to Q03. For better visualization, the entire diagram is provided with a color scale.

A color ranges from white to dark red was selected for the corresponding result diagrams in this dissertation.

In all the results of this dissertation, the number of test sets in each class may not be consistent, because the number of training or test sets in each class is also one of the factors affecting the NN results. So, the hits should be given in the form of relative values (see Figure 4.3).

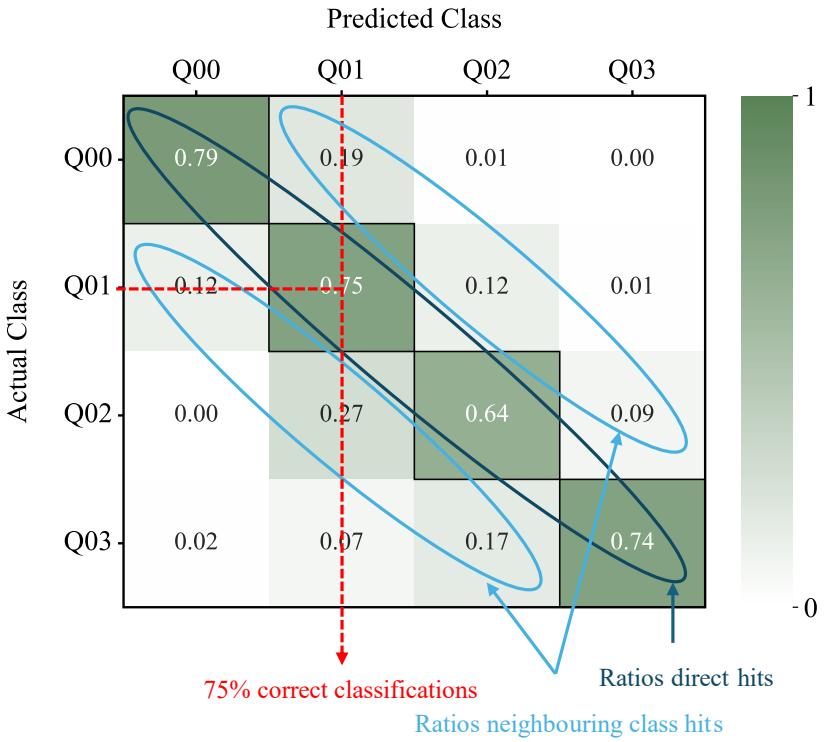


Figure 4.3: Confusion matrix with relative classes division

In this confusion matrix, all divisions are given relative to the respective test sets of the individual classes. The test accuracy of each class TA_C is calculated by

$$TA_C = \frac{N_{dh},c}{N_{\text{test sets}},c} \quad (4.1)$$

Where N_{dh},c is the number of the direct hits test sets of the c -th class, $N_{\text{test sets}},c$ is the number of the test sets of this class. An overall test accuracy of the classification (TA) can be calculated by Equation (4.2) from the mean value of the individual class hit rates, where C indicated the total number of the classes.

$$TA = \frac{1}{C}\sum_{c=1}^{C} TA_C \quad (4.2)$$

This TA will be slightly different than the accuracy rate of the NN classifier TA_{NN}, because each class may contain an inconsistent number of test sets. The formula for calculating the TA_{NN} is as follows:

$$TA_{NN} = \frac{\sum N_{dh,c}}{\sum N_{test\ sets,c}} \quad (4.3)$$

As mentioned in Section 4.1.1, TA_{NN} will be calculated by the evaluation program automatically and recorded after training of the NN classifier model.

4.2 Experiments Utilizing Frequency-domain Signals

In the realm of signal processing, frequency-domain signals can show the characteristics of vibration signals at any frequency. This section specifically addresses the application of frequency-domain signals within the framework of NN training. The feature representation for NNs is enhanced by leveraging the spectral characteristics of signals, which subsequently improves their classification capabilities. The transformation of time-domain data into the frequency-domain facilitates the extraction of informative features that are critical for accurate model training and classification. The subsequent subsections will elaborate on the process of integrating frequency-domain analysis into NN training, the impact on model performance, and the strategies employed to achieve effective classification outcomes.

4.2.1 Single Measurement Point Capability Investigation

As mentioned in Section 3.1.3, the z-axis acceleration of will be used to detect operating points. However, the number of points required, whether signals from multiple points need to be combined, and the optimal placement of the sensors must first be determined to achieve the best training results. The purpose of this dissertation is to train the NN using vibration signals from a single measurement point. To verify this, a series of tests were first carried out. Four sensor positions were symmetrically distributed on both sides of the upper surface of the pump casing (Figure 4.4).

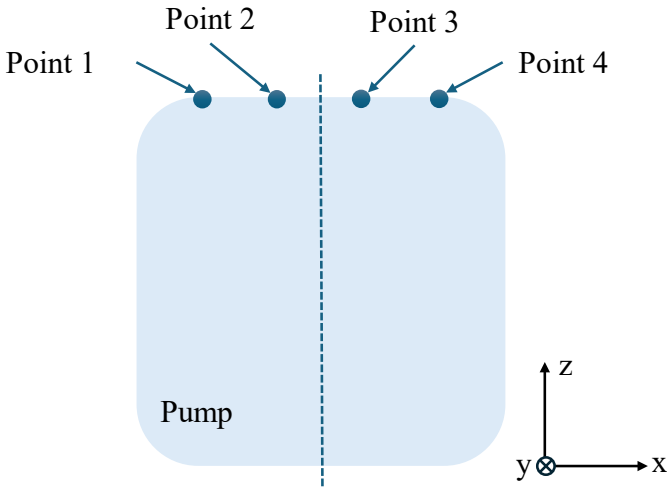


Figure 4.4: Sensor positions

The parameters of the dataset and the settings of the NN are shown Table 4.2 and Table 4.3.

Table 4.2: Dataset and NN settings for single measurement point investigation

	Dimension	Sets ratios
Value	4000×512	(0.8, 0.1, 0.1)

Table 4.3: NN settings for single measurement point investigation

	n_{hidden}	activation	optimizer	batch size	dropout	output activation
Value	10	ReLU	Adam	16	0.3	SoftMax

The NN model is kept simple and fixed, the signal from different measuring points was used as dataset. At the same time the $\triangle Q = 10\ m^3/h$ is chosen to make the classification easier. The classes in the dataset for sensor positions determine show in Table 4.4.

Table 4.4: Classes details for sensor positions determination

Classes	Q00	Q01	Q02	Q03	Q04	Q05
Range (m^3/h)	31~40	41~50	51~60	61~70	71~80	80~90
Dimension	635	900	760	560	525	620

The purpose of the settings is to make the whole system simple so that differences between measurement points can be observed. For comparison, the signals from two points were also combined and trained in the same NN model. The training results are shown in Figure 4.5.

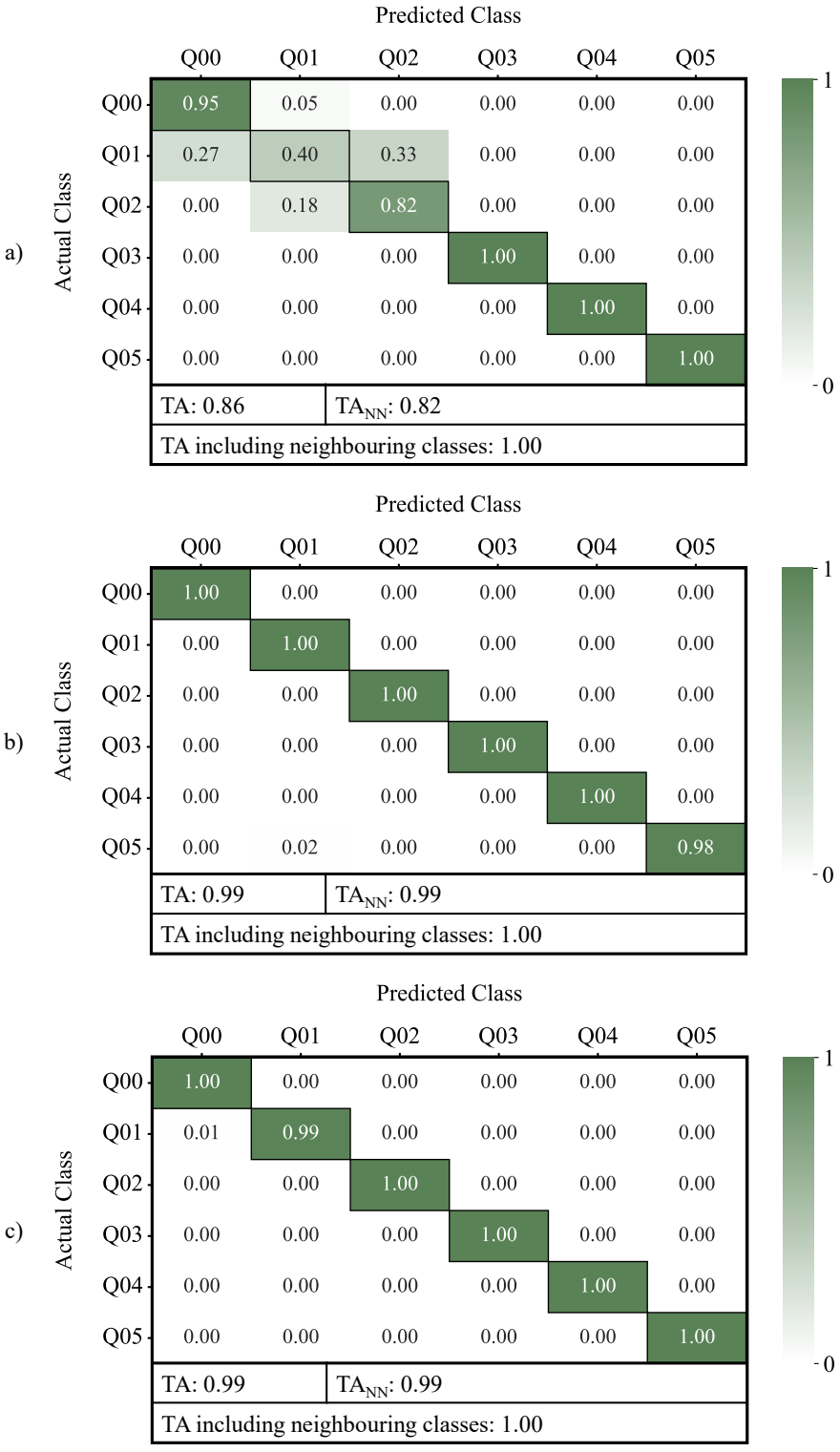


Figure 4.5: Simplified training results of the sensor positions

The results of the training show that the detections on Q01and Q02 by using dataset from Point1 have larger errors (Figure 4.5 a)), especially, the TA_{Q01} is only 0.40. The training results for the other three positions show that a single measuring point is able to detect

the operating point of the pump. The training results for the Point 4 are shown as an example in Figure 4.5 b). By simple classifications and NN model, the accuracy of individual point measurements on the test set can reach 0.99, within 10 epochs. The training results of the combined signals by Point 1 and Point 2 are shown in Figure 4.5 c), the performance of the NN model is not significantly improved, and it can be clearly seen that because of the inclusion of the signal from point1, TA_{Q01} has been reduced instead.

Consequently, sensor position Point 4 was used for all further investigations in this work.

4.2.2 Minimum Detectable Flow Rate Difference Recognize

At rotation speeds of n_1000 and n_1200, the flow rate gradually increases within the flow rate range of 30 m³/h to 45 m³/h as the opening degree of regulating valve increases. So, the vibration signals at these two rotation speeds were used for this case. The details of the dataset and the NN settings are listed in Table 4.5 and

Table 4.6.

Table 4.5: Dataset settings for single measurement point investigation

	Dimension	Sets ratios
Value	1872×1024	(0.8, 0.1, 0.1)

Table 4.6: NN settings for single measurement point investigation

	n_{hidden}	activation	optimizer	batch size	dropout	output activation
Value	1024	ReLU	Adam	2	0	SoftMax

The table indicates that the NN for this case is configured with 1024 hidden layer nodes, reflecting the choice of a complex network topology. The dataset corresponding to each flow rate difference are presented in the following Table 4.7.

Table 4.7: Classes for minimum detectable flow rate difference

Dataset	△Q (m^3/h)	Classes	Dimension
1	1	Q00~Q11	100, 203, 100, 256, 101, 303, 101, 103, 101, 101, 100, 304
2	2	Q00~Q07	303, 356, 404, 101, 103, 101, 201, 304
3	3	Q00~Q04	403, 660, 101, 204, 505

As observed from the class tables of each case, the data distribution across classes is highly imbalanced. This setup reflects practical scenarios where such inconsistencies in data volume across classes are common. The impact of dataset size on NN training will be discussed in detail in subsequent sections.

The training results show in Figure 4.6. By flow rate difference △Q = 3 m^3/h, the TA has already close to 0.90 (Figure 4.6 a)), with the neighboring class hit the TA can reach 0.99. After a decrease to △Q = 2 m^3/h (Figure 4.6 b)), the TA decreased by 0.03, but the TA including neighboring class only dropped 0.02. When △Q = 1 m^3/h, (Figure 4.6 c)), although the accuracy of the network has decreased considerably, the TA including neighbouring classes is still above 0.90, which illustrates the possibility of selecting 1 m^3/h as the minimum difference.

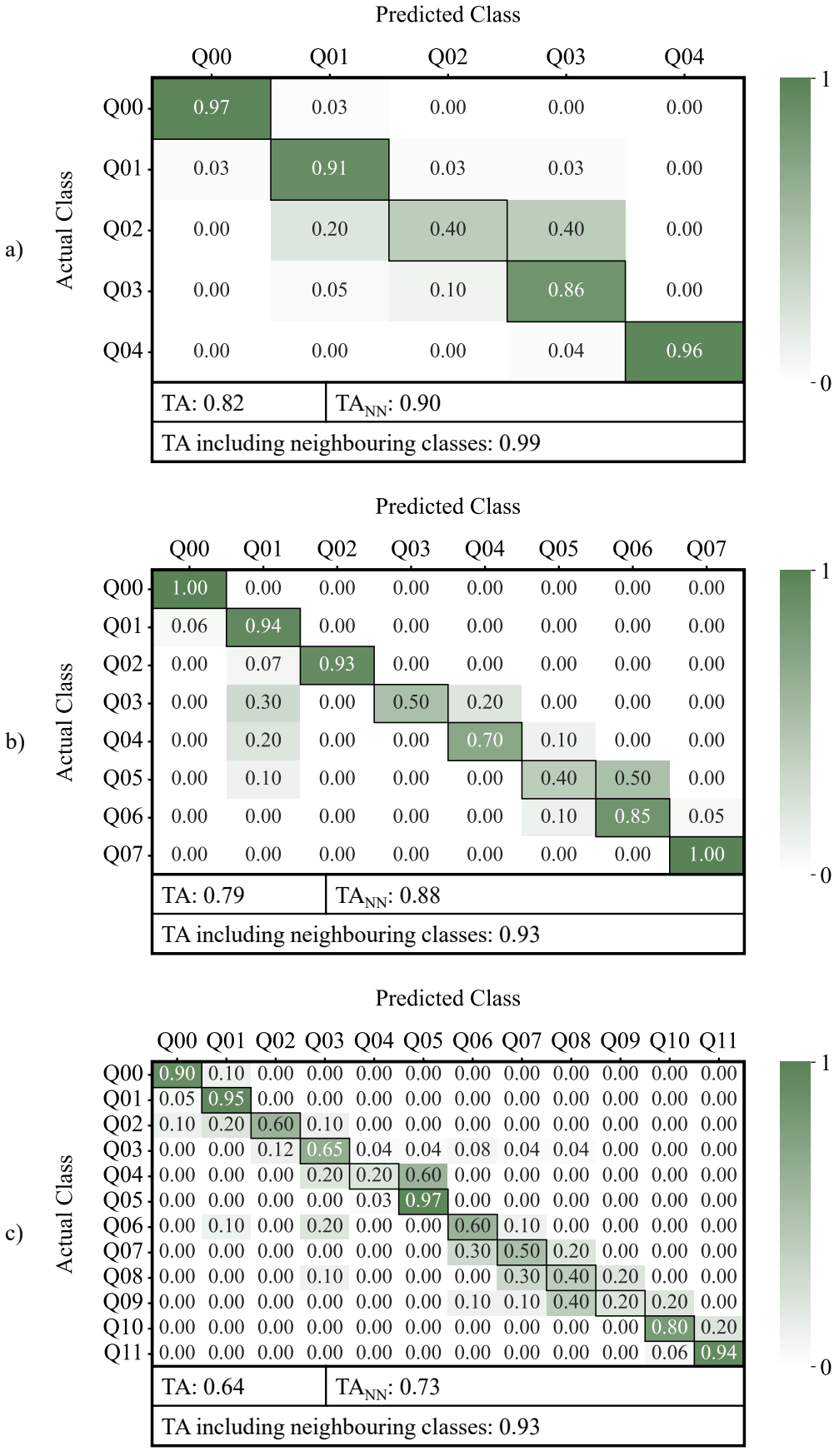


Figure 4.6: Training results of different △Q

4.2.3 The Impact of Dataset Settings on Training Results

In theory, the larger the data volume, the more accurate the training results of a NN. However, larger data volumes result in longer sampling and training times. This section investigates first the minimum volume of the dataset that can be used for operating point detection. The used data is the signals from rotation speeds of n_1800, n_2000 and n_2200. The details of the total data set are listed in Table 4.8.

Table 4.8: Dataset settings for investigation the minimum size of the training dataset

	Total data Dimension	Sets ratios
Value	3591×1024	(0.8, 0.1, 0.1)

Total Dimension of the total data in the Range n_1800 to n_220 is 3594×1024. To ensure accuracy on the test set, the minimum dataset size was determined to be 50×6×1024. This size is based on a minimum quantity (M_{Qi}) of 50 and a dataset split ratio of 0.8 for training, 0.1 for validation, and 0.1 for testing. Since the Q02 has only 423 rows, the maximum dataset dimension in this case is 400×6×1024. The NN settings are listed in Table 4.9.

Table 4.9: NN settings for investigation the minimum size of the training dataset

	n_{hidden}	activation	optimizer	batch size	dropout	output activation
Value	10	ReLU	Adam	8	0	SoftMax

The classes details of the dataset are listed in Table 4.10

Table 4.10: Classes details for examination the minimum size of the total data

	Classes					
	Q00	Q01	Q02	Q03	Q04	Q05
Range (m^3/h)	60~64	65~68	69~71	72~75	76~81	82~90
Dimension	446	544	423	666	762	750

The results of the training of training the same NN model using dataset with different dimensions are shown in Figure 4.7.

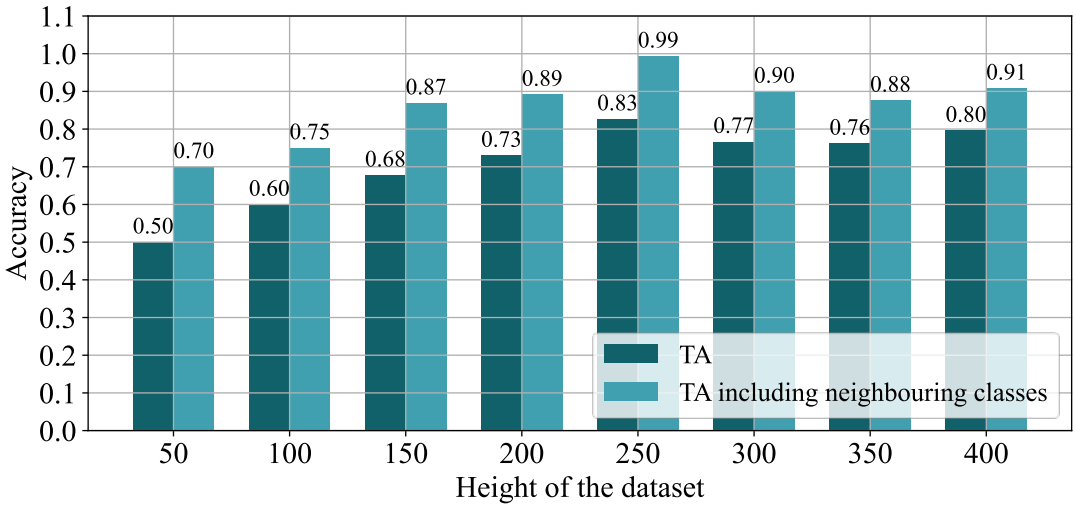


Figure 4.7: Accuracy of the NN model with different dataset height

The overall trend of the accuracy shows that the training results of the NN get better as the data volume increases. The results show that the test accuracy has reached an acceptable range when $M_{Qi} = 100$. The highest value of the TA occurs at $M_{Qi} = 250$, but this is due to the fact that the dataset of each dimension is randomly sampled from the total dataset. A comparison of the training results of these two dimensions shown in Figure 4.8.

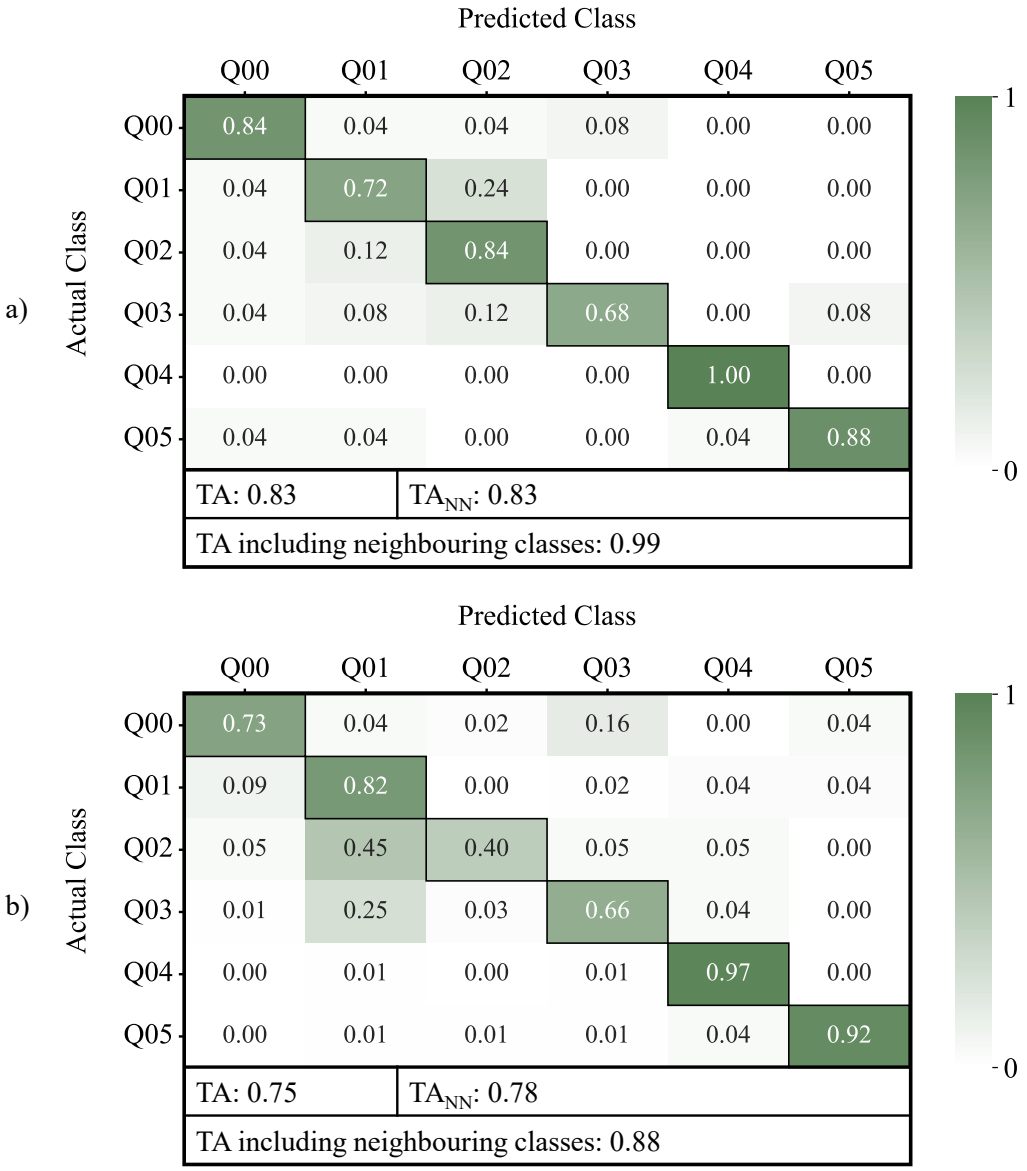


Figure 4.8: Training results of the total dataset

$TA_{Q2} = 0.84$ at $M_{Qi} = 250$ (Figure 4.8 a)) is outweigh $TA_{Q2} = 0.40$ at total dataset (Figure 4.8 b)). It is obvious that at $M_{Qi} = 250$ the highest accuracy was achieved because the sampling process happened to select a more optimal subset in class Q02 of the total dataset.

The process then involves reducing the signal length to determine the minimum dataset volume that can be achieved. In order to accurately acquire spectral information, the minimum length of the feature signal should be N = 128. The signals in dataset with M_{Qi} = 300 was directly pruned to different sizes and then input into the same NN for training. The training results are presented in Figure 4.9.

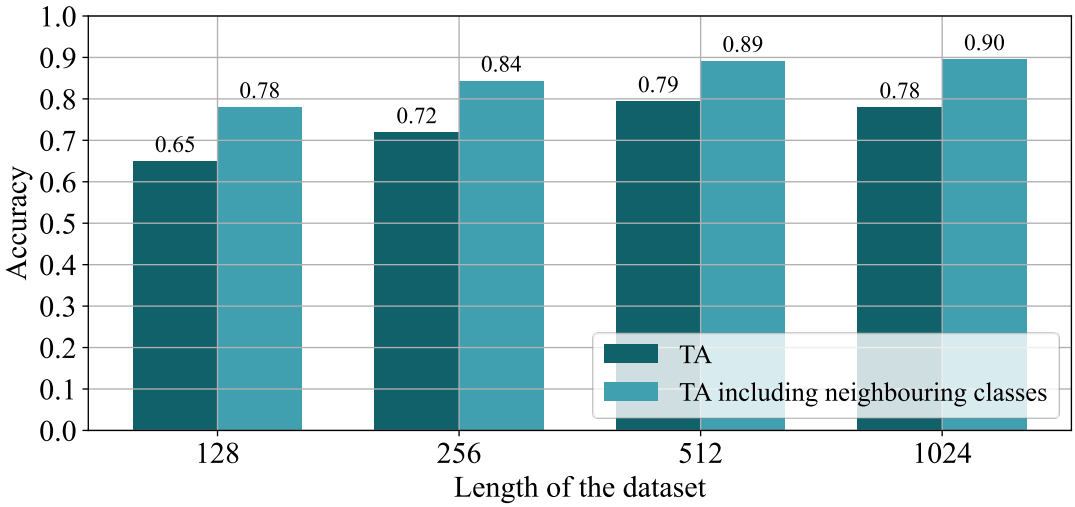


Figure 4.9: Accuracy of the NN model with different dataset length

In summary, when constructing dataset, each class should contain a minimum of 300 samples and feature signals should have a length of at least 256 to achieve an accuracy greater than 0.8.

The results from this case indicate also that discrepancies in data size between classes can influence the training results. However, when the dataset volume is sufficiently large, the impact of this imbalance on training results is minimal and, in practical scenarios, often unavoidable.

4.2.4 The Impact of Hidden Layer on Neural Network Accuracy

As the aim of this dissertation is to use simple NNs to detect operating point of the impeller pump, the effect of the NN topology on the training results can be simplified to the impact of the hidden layers of the NN. This section uses the same data as in the Section 4.2.3 used to investigate the effect of the hidden layer on the training results of the NN. The details of the dataset and the hyperparameter settings of then are listed in Table 4.11 and Table 4.12.

Table 4.11: Dataset settings for investigate the hidden layer effect

	Total data Dimension	Sets ratios
Value	2400×1024	(0.8, 0.1, 0.1)

Table 4.12: NN settings for investigate the hidden layer effect

	activation	optimizer	batch size	dropout	output activation
Value	ReLU	Adam	16	0.3	SoftMax

Firstly, a NN with a single hidden layer was evaluated. Figure 4.10 illustrates the impact of varying the number of nodes in the hidden layer on the training performance of the network.

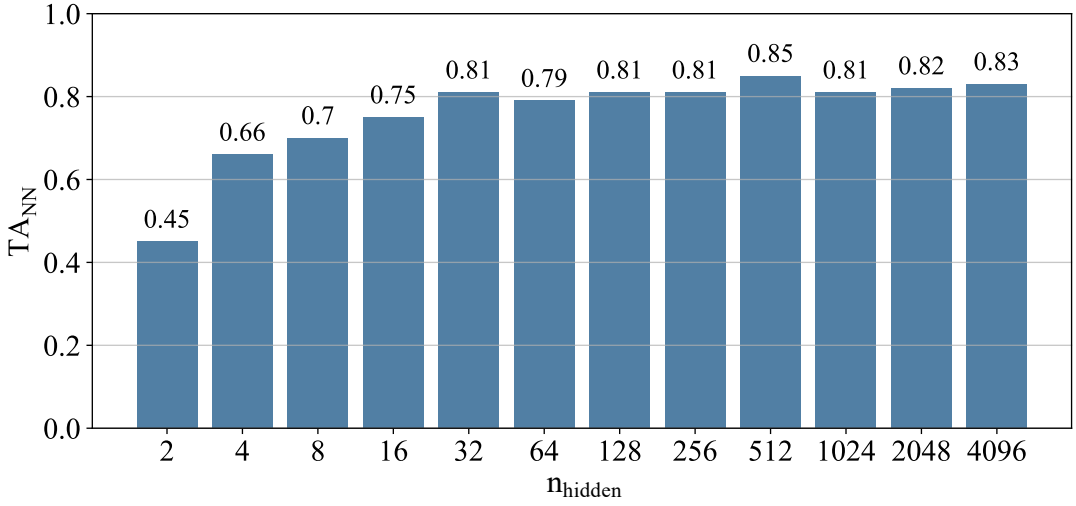


Figure 4.10: Effect of n_{hidden} on TA_{NN}

The n_{hidden} is ranging from 2 to 4096, the network accuracy stabilizes around 0.8 for $n_{hidden} \geq 32$, with a peak of $TA_{NN} = 0.85$ occurring at $n_{hidden} = 512$. Increasing the number of nodes in hidden layers increases the computational demands and memory requirements. More nodes mean that each forward pass and backpropagation step becomes more computationally intensive, which can significantly increase training time. Figure 4.11 shows the epochs to reach the maximum TA_{NN} at different n_{hidden}.

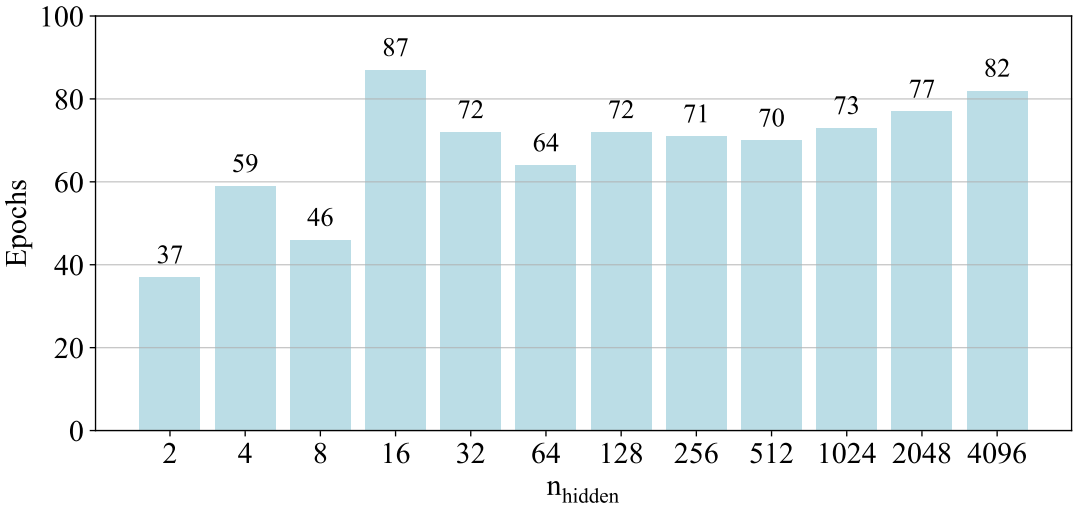


Figure 4.11: Epochs to reach the maximum TA_{NN} at different n_{hidden}

As $n_{hiddenn}$ increases beyond 128, the number of epochs appears to stabilize somewhat, with less fluctuation and the values clustering around a similar range, and there is a minimum value at n_{hidden} = 512. For this dataset, using 512 hidden units appears to be the optimal configuration when employing a single hidden layer in the NN.

Additionally, configurations with 2 hidden layers were also explored. Theoretically, a 2-layer NN should not require more nodes than a single-layer network to achieve the maximum accuracy. Therefore, with two hidden layers, the maximum number of nodes per layer is set to 512. The result of TA_{NN} is shown in Figure 4.12.

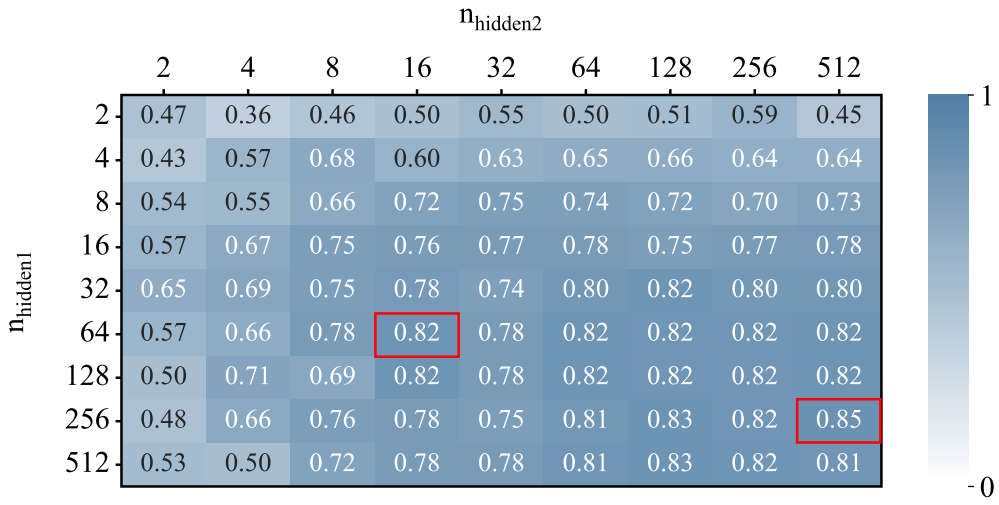


Figure 4.12: Effect of n_{hidden} in 2 hidden layers on TA_{NN}

The maximum accuracy of the network maintained at 0.85, when $n_{hidden1}$=256 and $n_{hidden2}$ = 512. This is clearly a more complex NN model than when there is only one hidden layer,

and it does not achieve a higher accuracy. However, adding a hidden layer shows an improvement in TA_{NN} when $n_{hidden1} = 64$. After introducing a second hidden layer with $n_{hidden2} = 16$, the TA_{NN} increased from 0.79 to 0.82.

4.2.5 The Impact of Hyperparameters on Neural Network Training Results

To investigate the effect of hyperparameters on training results, the dataset from Section 4.2.4 is used, and the network topology is fixed to the optimal structure identified in Section 4.2.4, a single hidden layer with 512 hidden nodes. Since it is a multiclassification problem, the activation function of the output layer is fixed as SoftMax. The effect of other hyperparameters on the training results is as follows.

Batch size:

The batch size is the number of samples processed by the model at each parameter update. In each iteration with a small batch size, the model processes a limited number of samples, resulting in more frequent updates to the model parameters. Conversely, with a larger batch size, more numbers of samples are processed per iteration, leading to higher computational overhead per update. However, the lower frequency of updates allows for more efficient utilization of parallel computing resources, potentially accelerating the training process. The training curve of batch size 2 and 64 are presented in Figure 4.13.

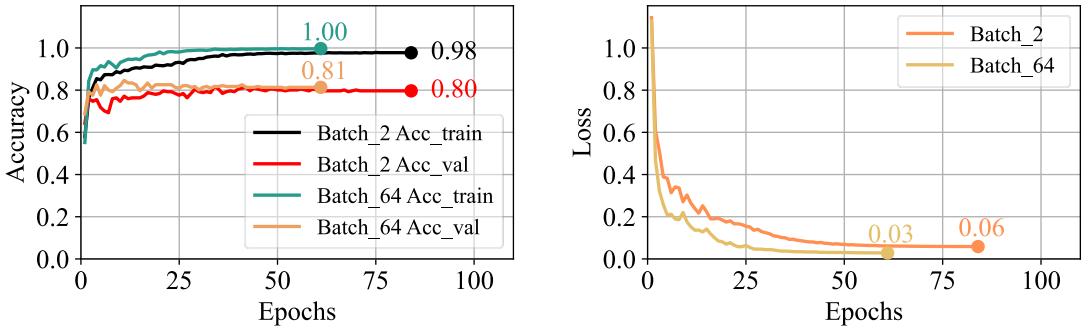


Figure 4.13: Training curve of different batch sizes

With a larger batch size, the update direction tends to be smoother, leading to faster convergence during training. However, larger batch sizes can increase the risk of converging to a local optimum, potentially reducing the network's final accuracy. The network accuracy at more batch sizes is shown in Figure 4.14.

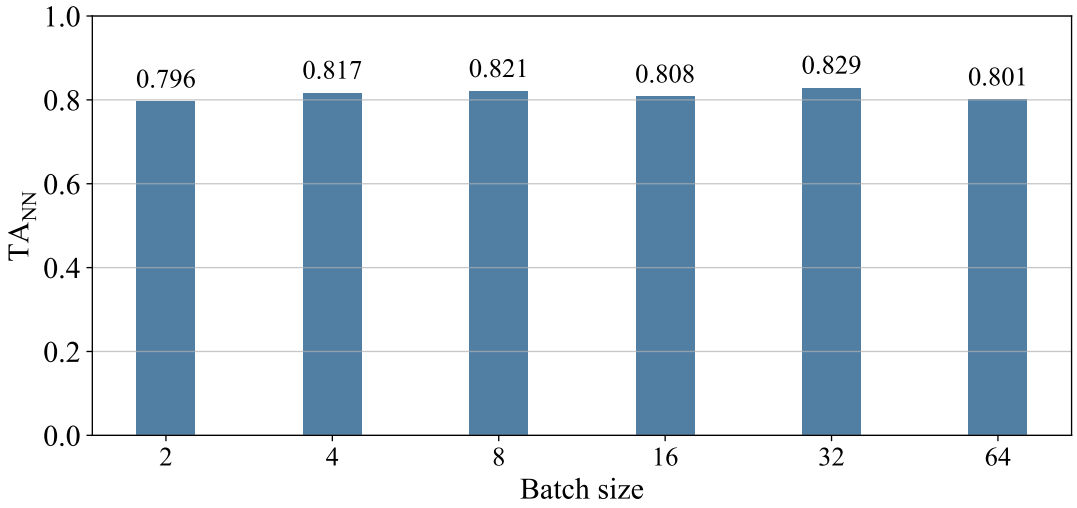


Figure 4.14: Effect of batch size on TA_{NN}

The results indicate that TA_{NN} does not exhibit a linear relationship with batch size. The selection of batch size should be experimentally tuned based on the specific task, model, and available hardware resources.

Activation function:

Sigmoid is now less commonly used for hidden layers due to the gradient vanishing problem and non-zero centered outputs, in this section, only the activation function ReLU and Tahn were used. Figure 4.15 shows the training curve using these two activation functions, with all other parameters held constant.

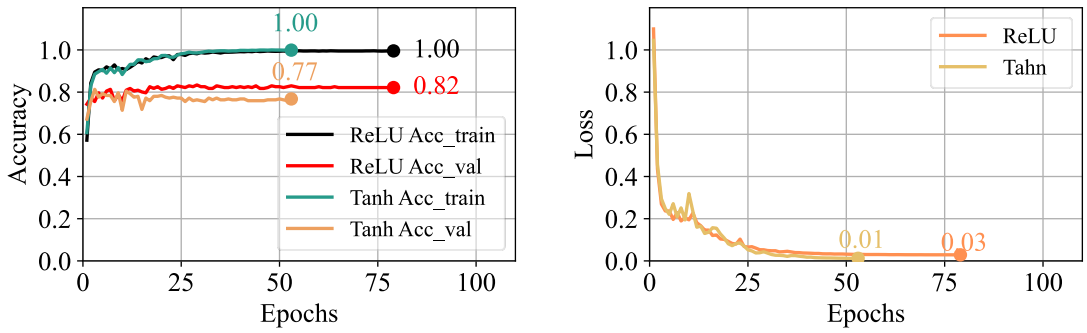


Figure 4.15: Training curve using activation function of ReLU and Tahn

Using Tanh as the activation function facilitates faster convergence in gradient descent within the first 60 epochs, achieving a smaller loss of 0.01. However, it is prone to the gradient vanishing problem, which can adversely impact training outcomes. TA_{NN} by

using Tahn is 0.78 and by using ReLU is 0.83. In contrast, ReLU is generally preferred due to its computational efficiency and reduced susceptibility to gradient vanishing.

Learning rate:

The learning rate controls the size of the steps taken during optimization, and its choice can impact convergence speed, training stability, and the quality of the final model. In the evaluation program, a custom learning rate scheduler was defined to adjusts the learning rate during training based on the current epoch. Specifically, for the first 10 epochs, learning rate remains constant. After the 10^{th} epoch, learning rate begins to decay exponentially by a factor of exp (-0.1). This method is to enhance training stability and efficiency, allowing the model to make significant progress initially and then refine its parameters more carefully as training progresses. The network accuracy by different initial learning rate shows in Figure 4.16.

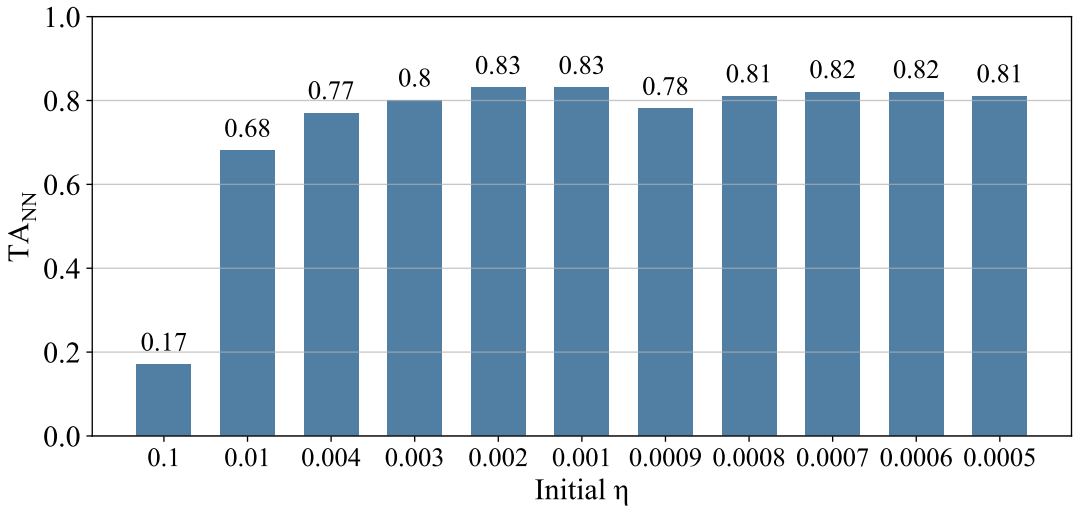


Figure 4.16: Effect of initial learning rate on TA_{NN}

When the initial learning rate is set above 0.004, the step size becomes too large, preventing the NN from being effectively trained. The best training results were obtained when initial learning rate is set to 0.001 and 0.002. Figure 4.17 shows the training curve by these two initial learning rates.

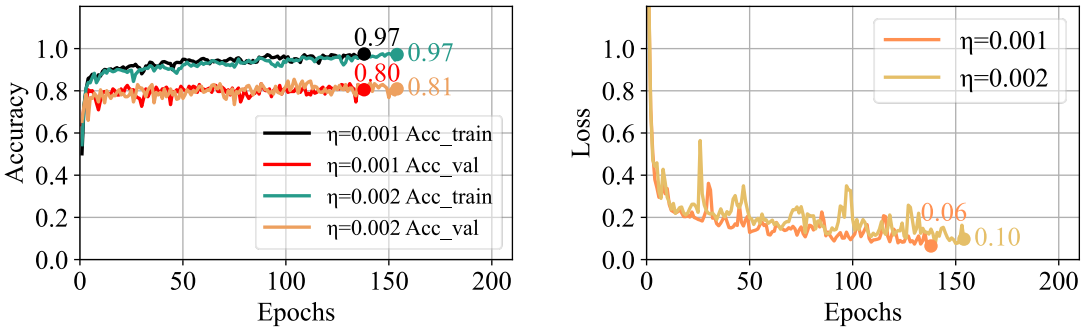


Figure 4.17: Training curve by initial learning rate at 0.001 and 0.002

The training converges faster when the initial learning rate is at 0.001. Although a larger learning rate usually leads to faster convergence, a larger learning rate may make the update too large, which may skip the optimal solution or lead to unstable training. This can also be seen in the loss function curve, where the oscillations are larger at 0.002 than at 0.001. On the contrary, a smaller learning rate mitigates this by making the gradient update more stable and accelerating convergence instead.

Optimizer:

Different optimizers have their own characteristics in terms of handling gradient updates, learning rate adjustments and model convergence. The custom learning rate scheduler mentioned above mainly affects the global learning rate of the optimizer, which subsequently maintains its adaptive tuning mechanism based on this adjusted global rate. Figure 4.18 illustrates the effects of different optimizers on the training results, both with and without the scheduler.

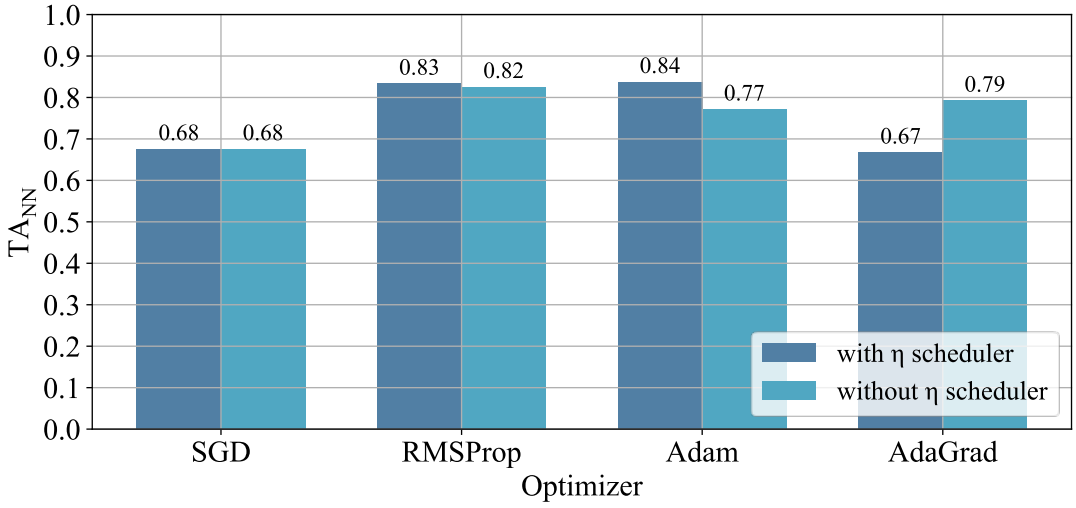


Figure 4.18: Effect of different optimizer on TA_{NN}

In this case, the results are better when RMSProp and Adam are used as the optimizer, the scheduler improves the training results with the exception of using Adagrad. Suitable optimizers likewise need to be experimented with and adapted to specific cases, datasets, and model architectures.

Dropout rate:

The dropout rate is commonly used in NNs to prevent overfitting. A well-chosen dropout rate can improve the generalization of the model, leading to better performance. Typically, a rate between 0.2 and 0.5 is common, but it requires experimentation. The result of the experimentation for this case is shown in Figure 4.19.

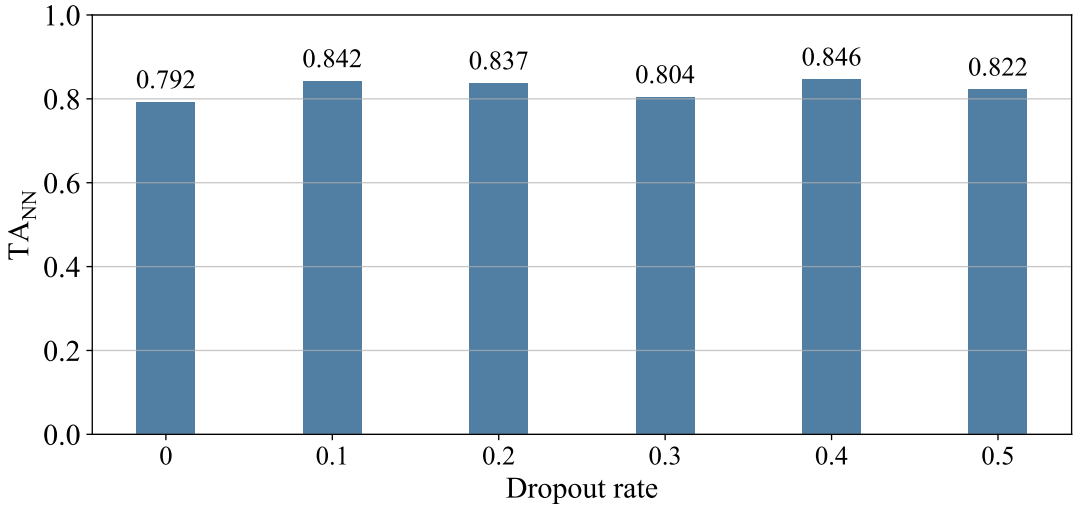


Figure 4.19: Effect of different dropout rate on TA_{NN}

The dropout rate of 0.1 or 0.4 is suitable for this case. The effect of with dropout without dropout on the training process is shown in Figure 4.20.

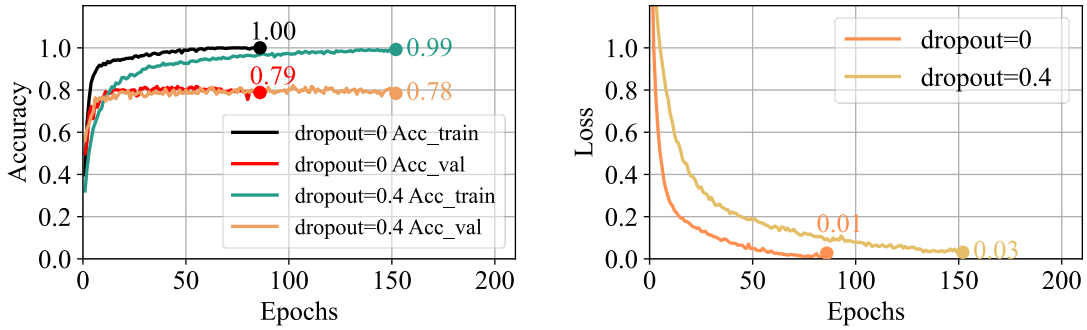


Figure 4.20: Training curve by dropout rate at 0 and 0.4

The results with dropout in this NN model have better test accuracy. Although the model without dropout achieved higher training, validation accuracy and lower training loss, it did not perform well on the validation or test set due to overfitting.

4.2.6 Classification of normal and non-normal operating points

Before carrying out the classification of the normal operating points, this dissertation first examined the classification of the non-normal and normal operating points, which is a binary classification. A speed of 1000 rpm is used here as the division point, the operating points below 1000 rpm are considered to be non-normal and those above or equal to 1000 rpm are considered to be normal operating points (Table 4.13).

Table 4.13: Classes details of non-normal and normal

Classes	Non-normal	Normal
Range	n_650~n_950	n_1000~n_2000
Dimension	199	477

The fixed parameters of the dataset and the settings of the NN are shown in Table 4.14 and

Table 4.15.

Table 4.14: Dataset settings for non-normal and normal classification

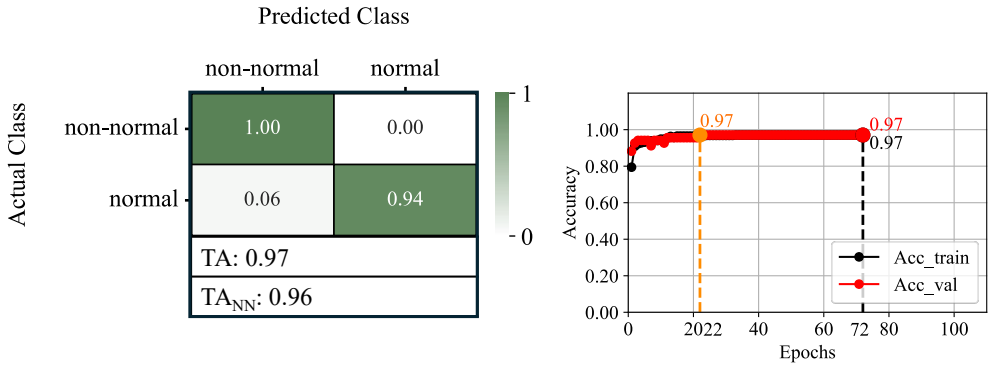
	Dimension	Sets ratios
Value	676×1024	(0.8, 0.1, 0.1)

Table 4.15: NN settings for non-normal and normal classification

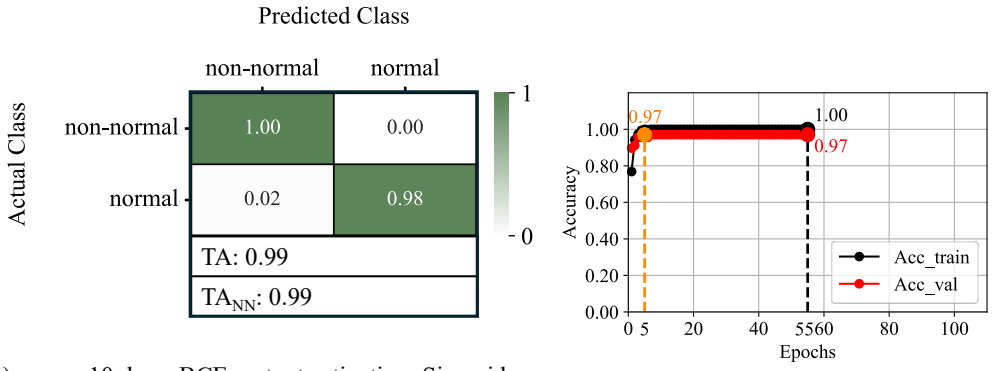
	$n_{hidden\text{-}layer}$	activation	optimizer	batch size	dropout
Value	1	ReLU	Adam	2	0

The NN model in the binary classification varied in n_{hidden} and output layer activation. Specifically, n_{hidden} was set to 2 and 10. For the output layer, both the Sigmoid and SoftMax activation functions were used, with BCE being used for the Sigmoid and CCE

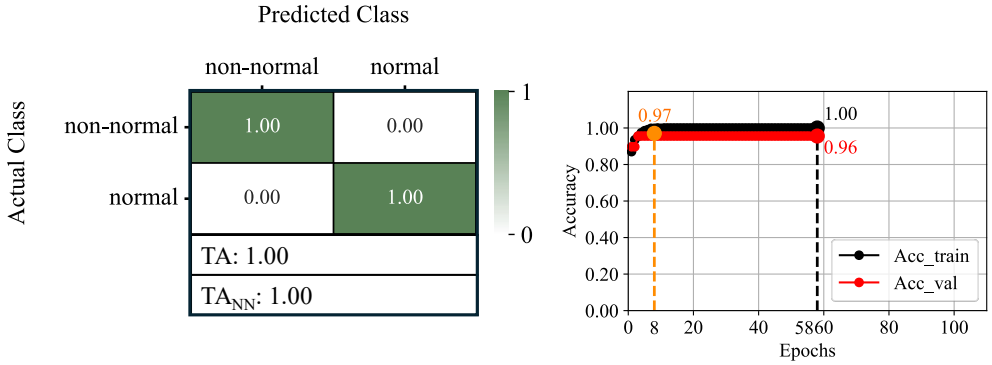
for the SoftMax as the loss function. The training results of four topologies are shown in Figure 4.21.



a) n_{hidden}=2, loss=BCE, output activation=Sigmoid



b) n_{hidden}=10, loss=BCE, output activation=Sigmoid



c) n_{hidden}=2 and 10, loss=CCE, output activation=SoftMax

Figure 4.21: Classification of the non-normal and normal operating points

By Sigmoid activation function (Figure 4.21 a) and b)), the topology with n_{hidden} = 2 achieved accuracy 0.97 in 22 epochs. The topology with n_{hidden} = 10 achieved 1.00 accuracy in predicting non-normal cases and 0.92 accuracy in predicting normal cases, with 0.08 of normal cases misclassified as non-normal. The Acc_train (black line) rapidly increased and maintaining above 0.99, the Acc_val (red line) achieved maximum 0.97 by the 5th epoch. The TA of this model achieved 0.96 in less than one minute. However,

when the activation function of the output layer of the model was changed to SoftMax, the performance of the simple model became even better (seen Figure 4.21 c)). The performance of the complex model has already reached 1.00 and cannot be improved, so only the same performance can be achieved after changing the n_{hidden}. It is worth noting that the classifier still performs very well in this binary classification case, even on the simplest NN model.

4.2.7 Rotation Speed Detection

As expected, the rotational speed of the pump is not difficult to detect as the rotational speed is reflected in the different frequencies of the vibration. To validate these results, a flow rate independent speed classification was performed on all the signals from the JHU test bench. However, various hyperparameters of the NN model affect the training results, and in this dissertation, the fixed NN setting is a single hidden layer. Therefore, it is also necessary to program a training selection program and perform several trials to determine the optimal NN model based on this case.

The selection program aims to find the simplest NN model with the highest test accuracy. The number of nodes was tested within 2 to 256 (growing in powers of 2) and the maximum TA_{NN} under each n_{hidden} are shown in Figure 4.22.

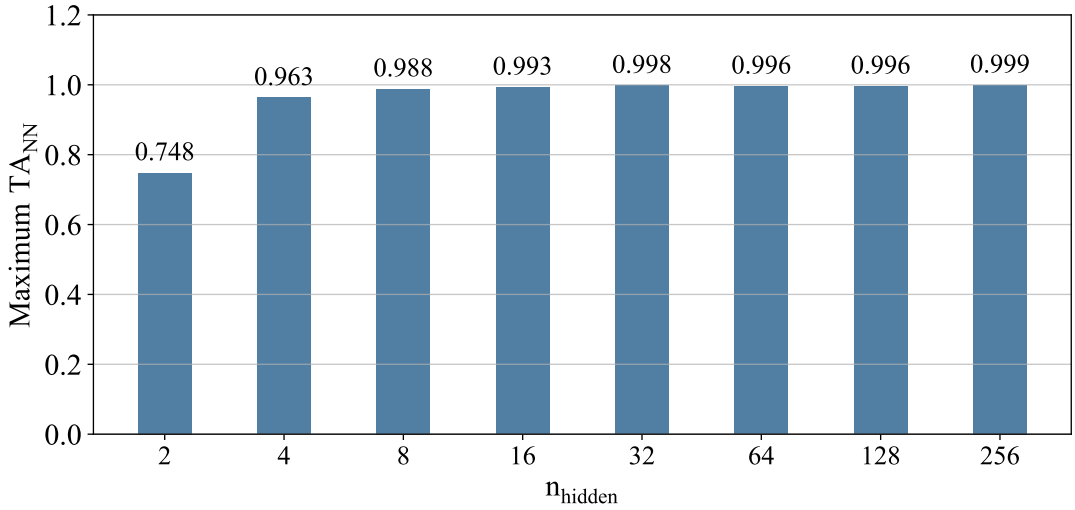


Figure 4.22: Maximum TA_{NN} of the rotation speed classification with different n_{hidden}

In this case, when the hidden layer contains four nodes, the TA_{NN} already exceeds 0.9. The maximum TA_{NN} is achieved with 256 hidden nodes. The NN model hyperparameters are different by each maximum TA_{NN}, details list in Table 4.16.

Table 4.16: Hyperparameters and training duration of each optimal NN model

n_{hidden}	2	4	8	16	32	64	128	256
Batch Size	128	128	128	128	64	64	128	128
Dropout	0.2	0.2	0.1	0.2	0.1	0.3	0.5	0.4
Initial η	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004	0.0003
Optimizer	RMSProp	RMSProp	RMSProp	RMSProp	RMSProp	RMSProp	Adam	Adam
Training duration (s)	10.57	15.98	15.74	16.95	20.66	14.5	17.22	29.41

For selecting the simplest NN model, a hidden layer with eight nodes ($n_{hidden} = 8$) is the optimal choice, as it offers a balance between model simplicity, higher accuracy and reduced training time.

The training curves and confusion matrixes of the optimal NN model ($n_{hidden} = 8$) and the model with maximum TA_{NN} ($n_{hidden} = 256$) are shown in Figure 4.23 and Figure 4.24.

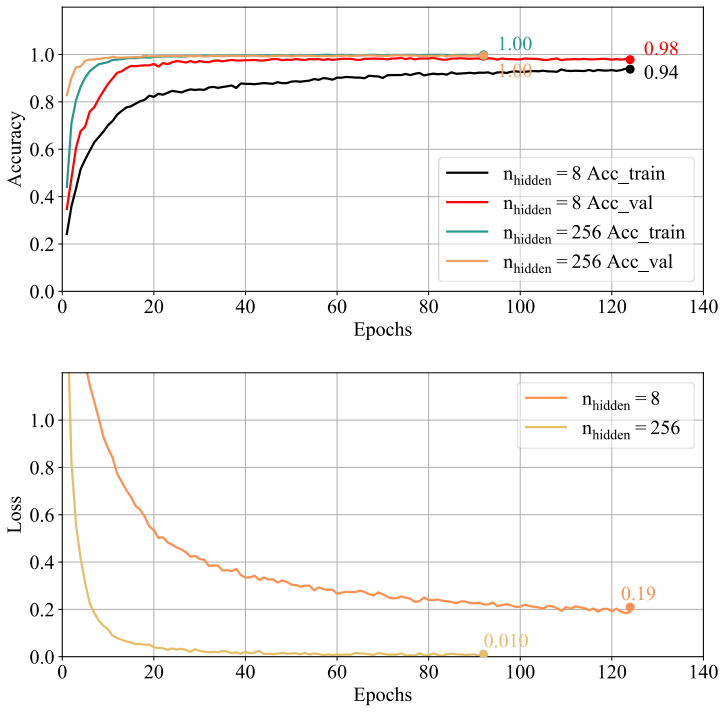


Figure 4.23: Training curves of the NN model with $n_{hidden} = 8$ and $n_{hidden} = 256$

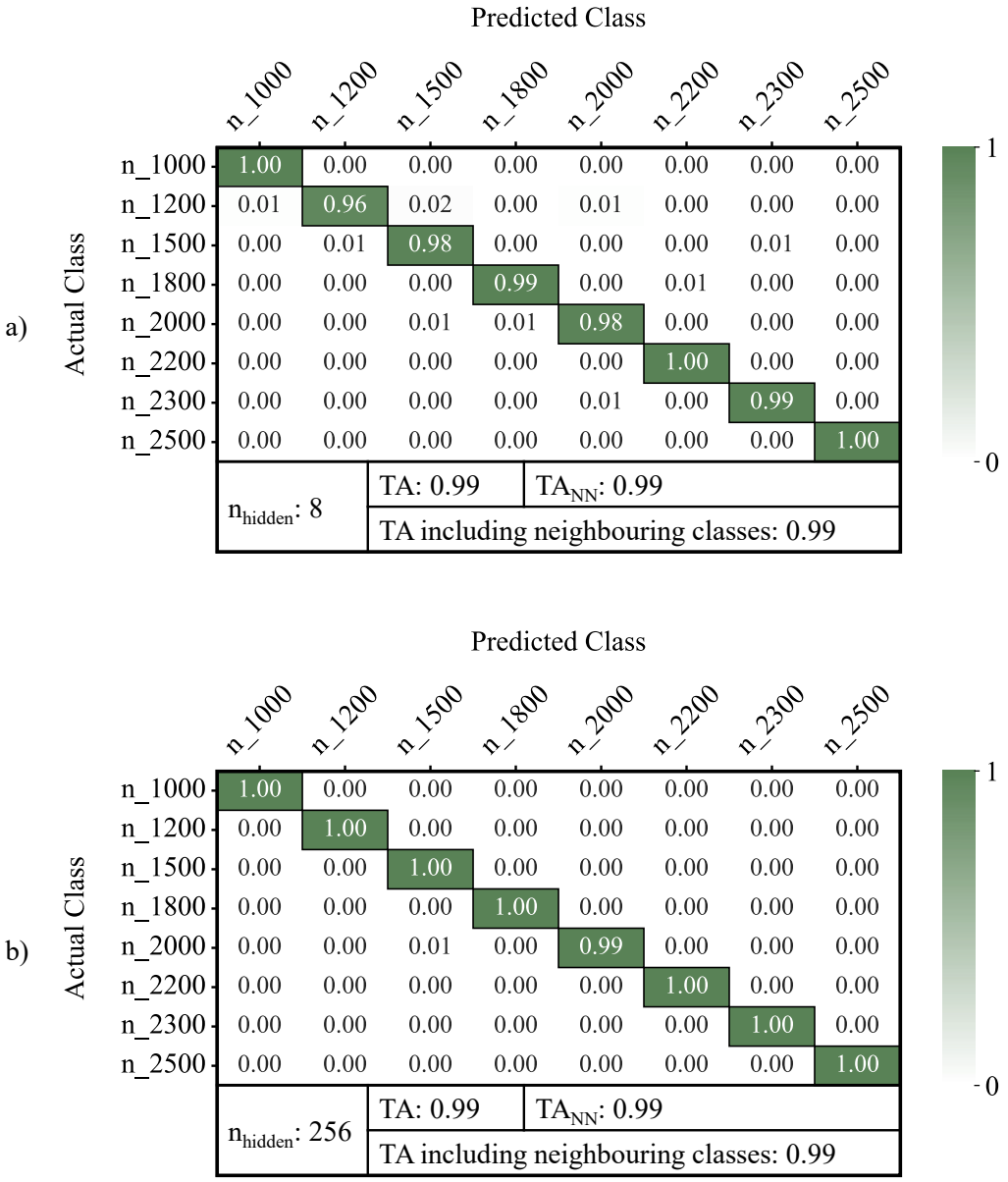


Figure 4.24: Confusion matrix of the NN model with $n_{hidden} = 8$ and $n_{hidden} = 256$

Based on the training curves, if simplicity isn't the priority, the model with 256 hidden nodes provides better training performance, with a 15-second increase in training duration time compared to the model with 8 hidden nodes.

4.2.8 Flow Rate Detection

The preliminary investigations carried out so far have already been investigated on the basis of flow rate classification. The results have shown that simple NNs can classify the flow rate. When focusing solely on the flow rate of the test bench, the task is formulated as a large multiclassification problem. The flow rate interval is set to 2 m^3/h, and the data

is divided into 35 classes. The screening program for finding the simplest NN model with the highest test accuracy is also used. However, for larger classification problems, there is a difference between hyperparameter selection and testing for speed classification. Due to the increased number of classes, a model that is too simple may struggle to learn the complex features required for accurate classification. Therefore, the nodes of the hidden layer in this case are between 64 and 2048. The maximum TA_{NN} under each n_{hidden} is shown in Figure 4.25.

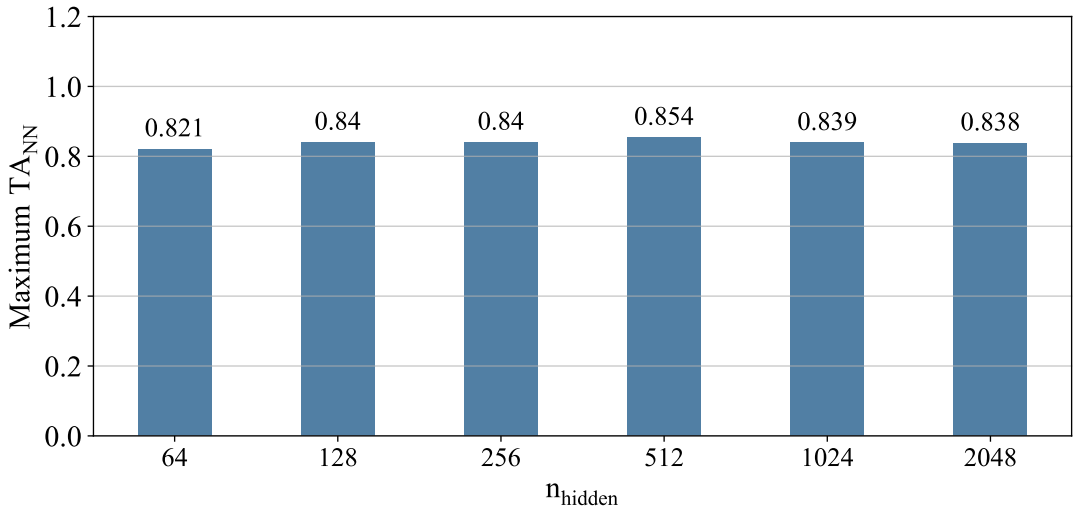


Figure 4.25: Maximum TA_{NN} of the flow rate classification with different n_{hidden}

The maximum TA_{NN} of this case is achieved with 512 hidden nodes. The NN model hyperparameters are list in Table 4.17

Table 4.17: Hyperparameters and training duration of each NN model

n_{hidden}	64	128	256	512	1024	2048
Batch Size	128	256	64	128	64	256
Dropout	0.1	0.3	0.4	0.5	0.4	0.1
Initial η	0.0001	0.0001	0.0002	0.0003	0.0001	0.0003
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam
Training duration (s)	179.65	165.95	133.75	192.43	471.1	153.24

The confusion matrix of the optimal NN model is shown in Figure 4.26.

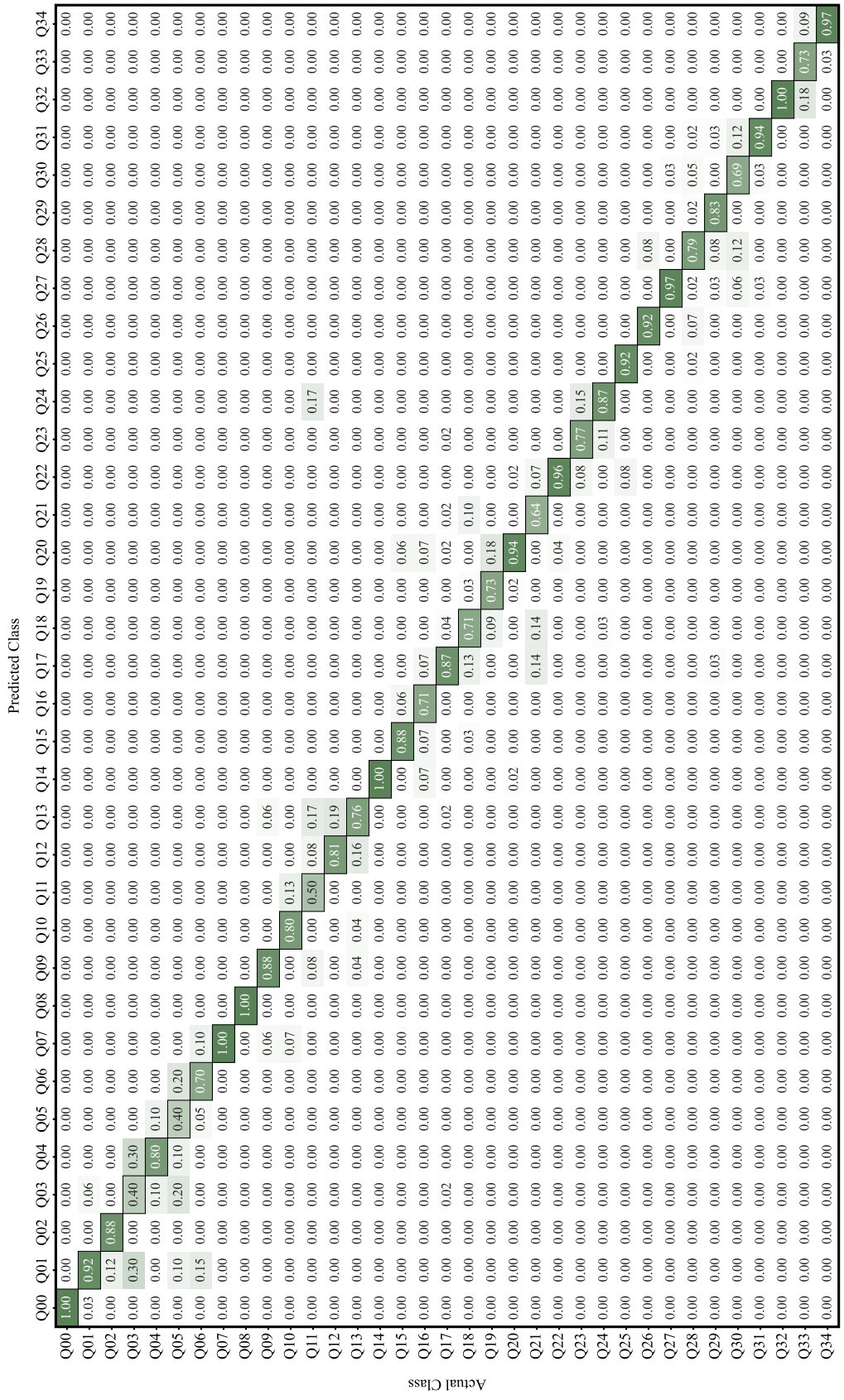


Figure 4.26: The confusion matrix of the optimal NN model for flow rate detection

The TA including neighboring classes achieved 0.915 by this NN model. However, if flow rate detection with a flow rate interval 1 m³/h is conducted at each rotation speed, the accuracy of the test set will vary significantly due to the differing minimum intervals of the flow rate at different rotational speeds. The TA, TA including neighboring classes are shown in Figure 4.27.

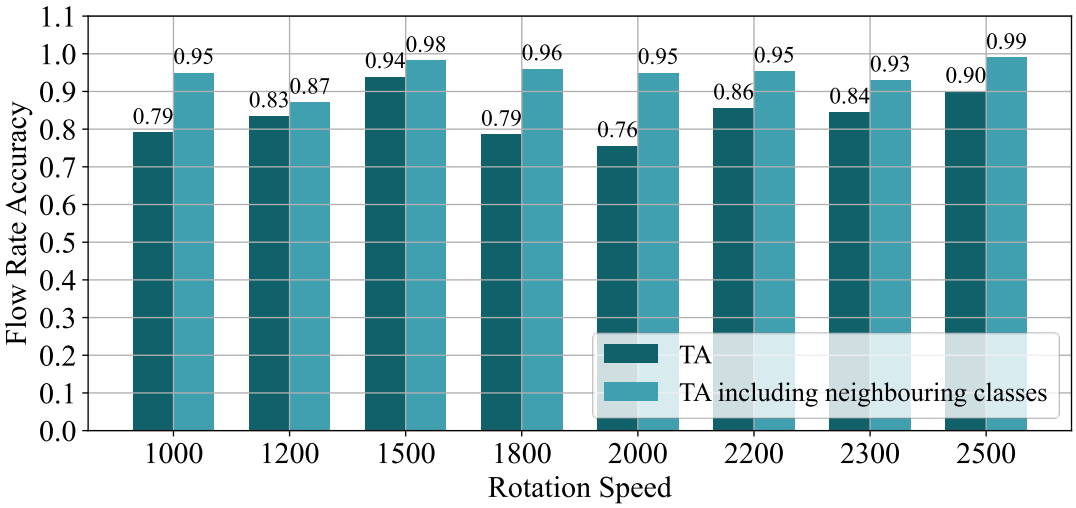


Figure 4.27: Flow rate accuracy under each rotation speed

The hyperparameters of the optimal model at each rotation speed are presented in Table 4.18.

Table 4.18: Test accuracy and hyperparameters of each NN model

Rotation Speed	1000	1200	1500	1800	2000	2200	2300	2500
Batch Size	128	128	128	128	128	128	64	128
Dropout	0.3	0.3	0.3	0.1	0.3	0.3	0.3	0.2
Initial η	0.0001	0.0004	0.0003	0.0002	0.0001	0.0002	0.0004	0.0003
n_{hidden}	256	256	256	32	128	256	64	256
Optimizer	RMSP rop	Adam	Adam	Adam	RMSP rop	RMSP rop	Adam	RMSP rop
Training Duration (s)	9.93	10.61	12.1	13.67	13.79	10.25	7.29	10.2

The confusion matrix at each rotation speed is shown in the Appendixes.

4.2.9 Rotation Speed and Flow Rate Detection

By transforming the output layer of the NN into two layers, both the flow rate and rotation speed corresponding to a specific vibration signal can be identified within a single NN model. The selection program will still yield the NN model with the highest accuracy. However, unlike the single output layer, the accuracy of the NN on the test set is in this case calculated as the average of the accuracies of the two output layers. The tested n_{hidden} is from 64 to 2048 (increases by powers of 2). The NN model with highest average accuracy of 0.909 appears when $n_{hidden} = 2048$. The test accuracy for rotation speed is 0.996 and for flow rate is 0.820. Compared to the accuracy of detecting flow rate and rotational speed separately-where rotational speed achieves 0.998 with $n_{hidden} = 256$ and flow rate achieves 0.854 with $n_{hidden} = 512$-the two output layers NN model becomes more complex and the test accuracy decreases. However, the advantage of this NN model is that after 116.01 seconds of training, both flow rate and rotation speed can be detected at the same time. The hyperparameters of the optimal NN model listed in Table 4.19.

Table 4.19: Hyperparameters and of the optimal NN model with two output layers

	$n_{hidden\text{-}layer}$	optimizer	batch size	dropout	Initial η
Value	2048	RMSProp	128	0.4	0.0001

The confusion matrix of rotation speed detection of the two output layers NN model shown in Figure 4.28.

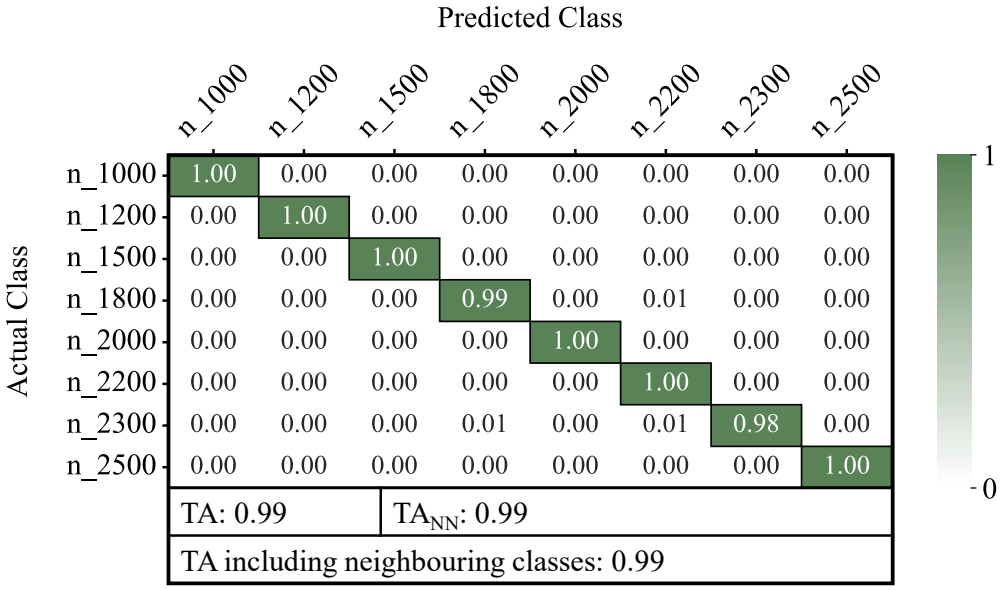


Figure 4.28: The confusion matrix of rotation speed detection of combined NN model

Compared to Figure 4.24 the TA including neighboring classes decreases only 0.001 by detection of class n_2300. The confusion matrix of flow rate detection of combined NN model shown in Figure 4.29.

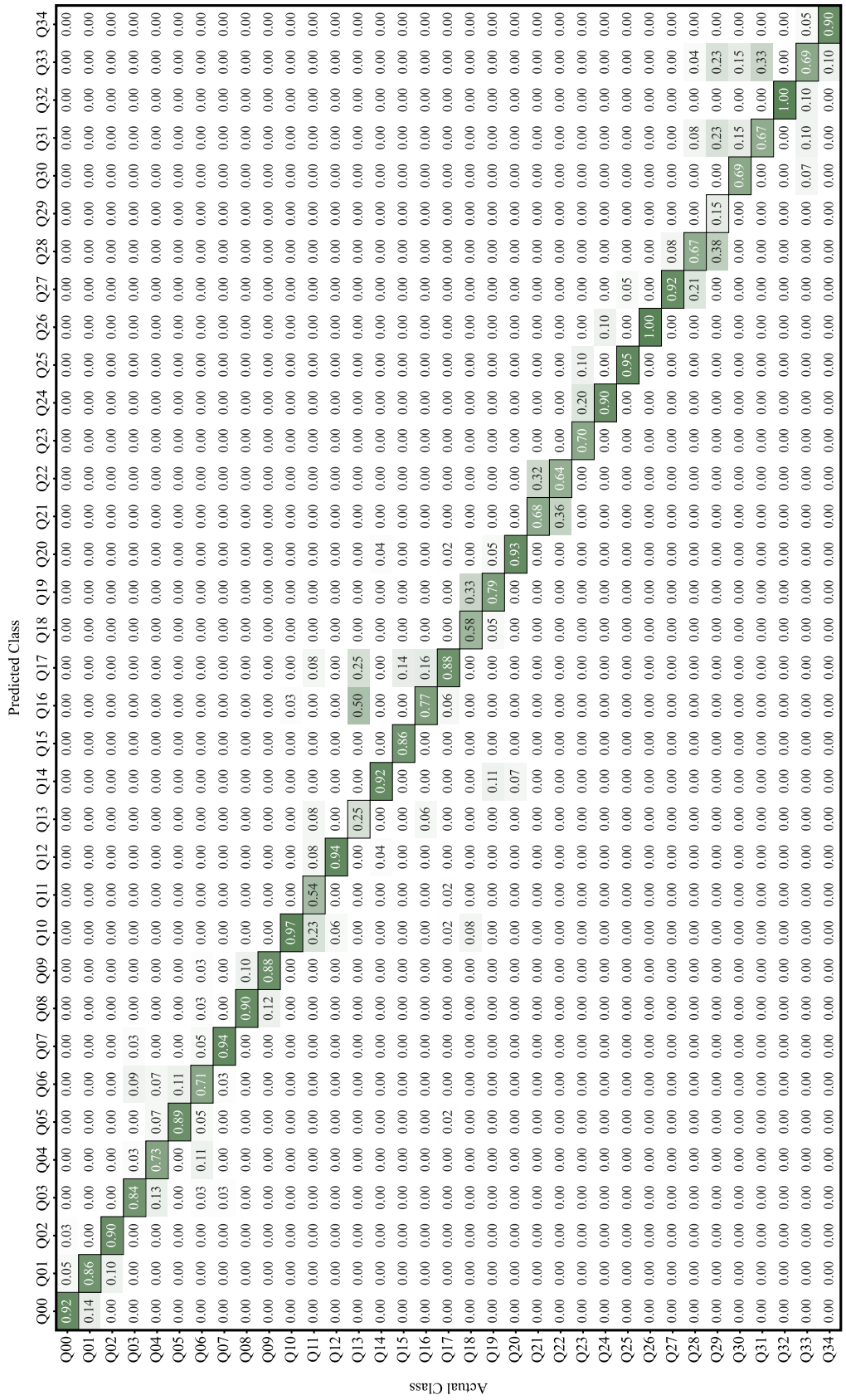


Figure 4.29: The confusion matrix of flow rate detection of combined NN model

The TA including neighboring classes of the flow rate detection achieved 0.898 by this combined NN model, decreases also only 0.017 by individually detection.

Moreover, even with only 64 hidden layer nodes, the average test accuracy of the two output layers NN model can reach 0.893, with 0.995 for rotational speed and 0.791 for flow rate. While it is possible to improve accuracy by experimenting with an increased number of nodes in the hidden layer. However, this study focuses on the simplest NN model. Therefore, exploring more complex NN models is beyond the scope of this dissertation.

In summary, the simple NN classifier has been successfully applied to the operating point detection of an impeller pump on an equipment dependent system. Both of rotation speed and flow rate can be detected in a single simple NN model. In the following, an equipment independent application of the NN classifier is examined.

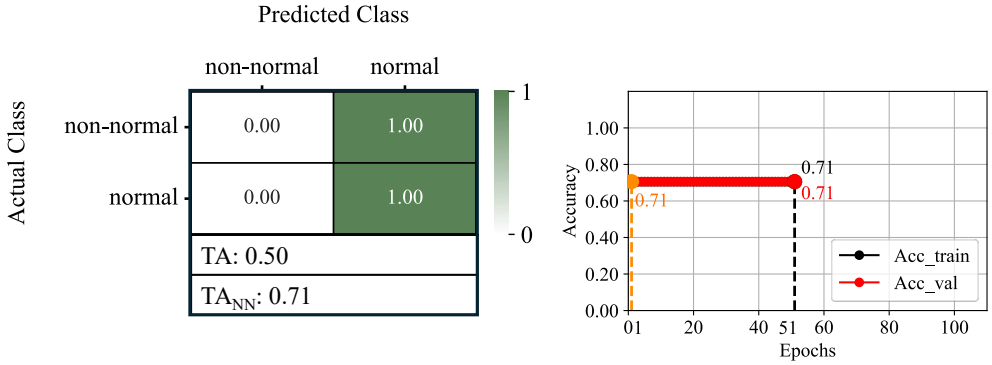
4.3 Experiments Utilizing Time-domain Signals

In this section, experiments were conducted using the same fundamental data as in Section 4.2, with the distinction that time-domain signals were utilized here. Specifically, signals that do not directly reflect vibration frequency were used to differentiate operating points. The selection process for the NN model involved initially training with the optimal frequency-domain signal model to compare accuracy, followed by the use of the screening program to identify the optimal solution for time-domain signals. The experiments were similarly conducted to distinguish between normal and non-normal operating points. Three differently processed time-domain signals will be tested.

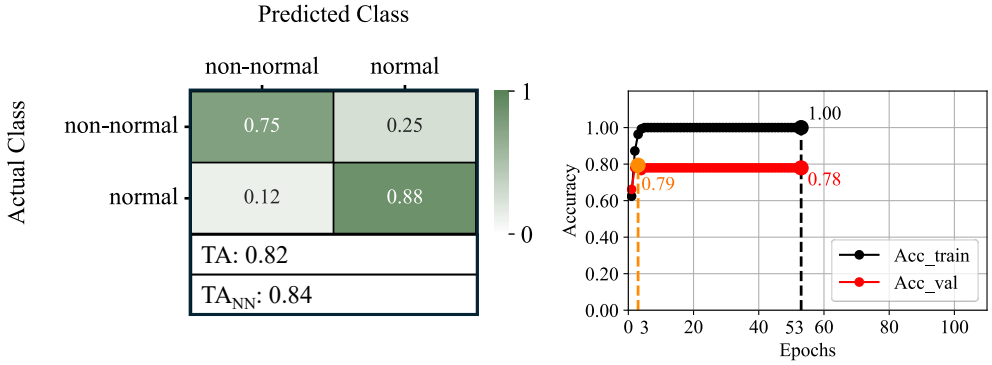
4.3.1 Classification of Normal and Non-normal Operating Points

4.3.1.1 Utilizing Waveform of Signals

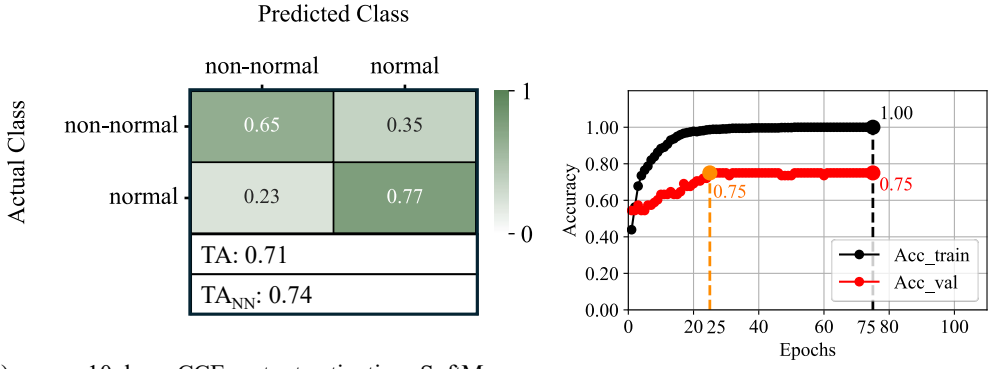
Training is performed directly using waveforms, means that no additional processing is applied to the signals captured by the sensors. The data is fed into the NN as input only after the appropriate classification labels have been added. The results obtained using the same simple NN model as the that used to train the frequency-domain signals are shown in Figure 4.30.



a) n_{hidden}=2, loss=BCE/CCE, output activation=Sigmoid/SoftMax



b) n_{hidden}=10, loss=BCE, output activation=Sigmoid

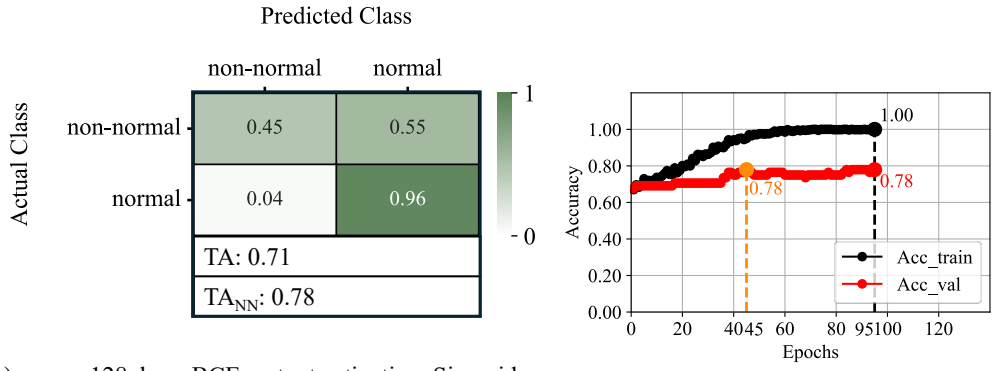


c) n_{hidden}=10, loss=CCE, output activation=SoftMax

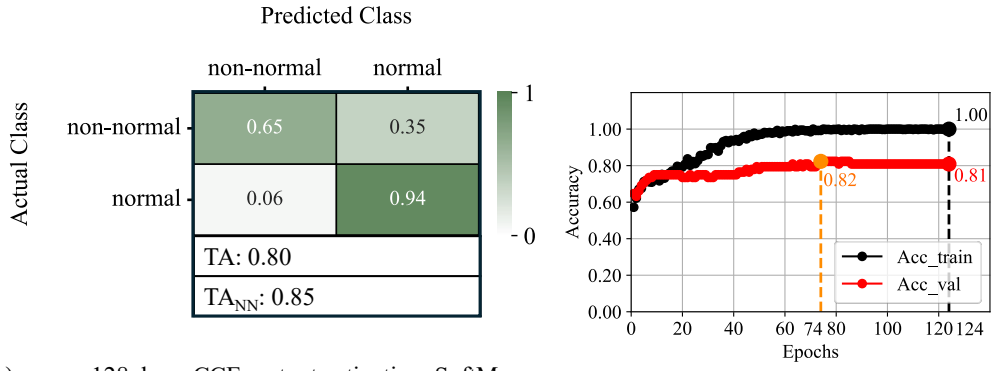
Figure 4.30: Classification of the non-normal and normal operating points with simplest NN model

When n_{hidden} = 2, although the network accuracy is 0.71, it can be observed from the confusion matrix (Figure 4.30 a)) that the model is unable to distinguish abnormal working points at all. n_{hidden} = 10, the results with the Sigmoid model are better than those with the SoftMax model, with the Sigmoid model achieving an accuracy of 0.75 for non-normal working points. However, from the accuracy curve during the model training process, it is evident that the model's performance on the validation set is already quite limited, with maximum accuracies of only 0.79 and 0.75, respectively (Figure 4.30 b) and

c)). The result after attempting to train with the more complex model is shown in Figure 4.31.



a) n_{hidden}=128, loss=BCE, output activation=Sigmoid



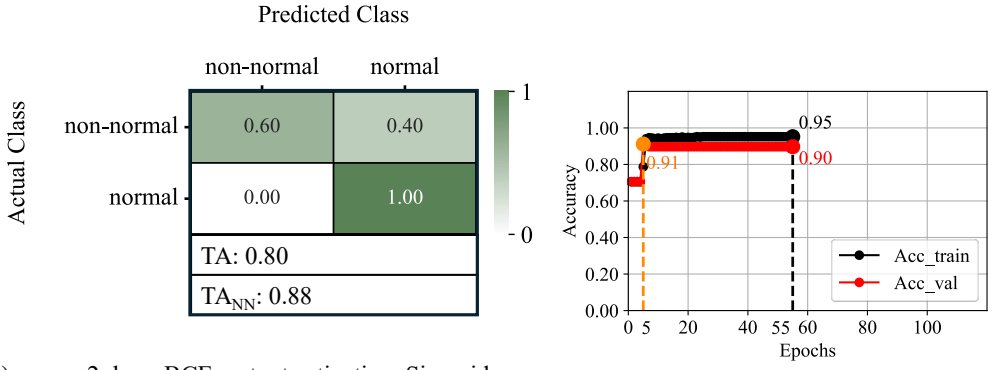
b) n_{hidden}=128, loss=CCE, output activation=SoftMax

Figure 4.31: Classification of the non-normal and normal operating points

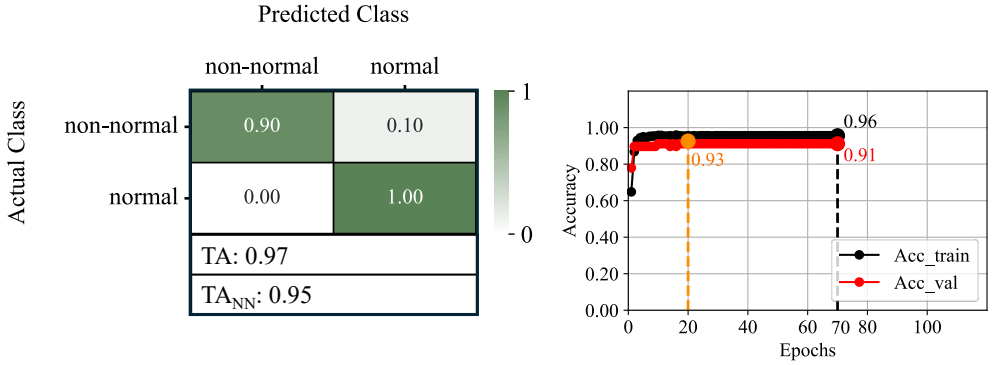
Although the network accuracy of the SoftMax model improved (Figure 4.31 b)), the more complex model did not improve the discrimination of non-normal operating points, as evidenced by the confusion matrix. In contrast, the accuracy of the sigmoid model decreased for non-normal work points (Figure 4.31 a)), with only a 0.06 improvement in accuracy for normal operating points.

4.3.1.2 Utilizing the Peak-to-peak and Root Mean Square Values of the Signals

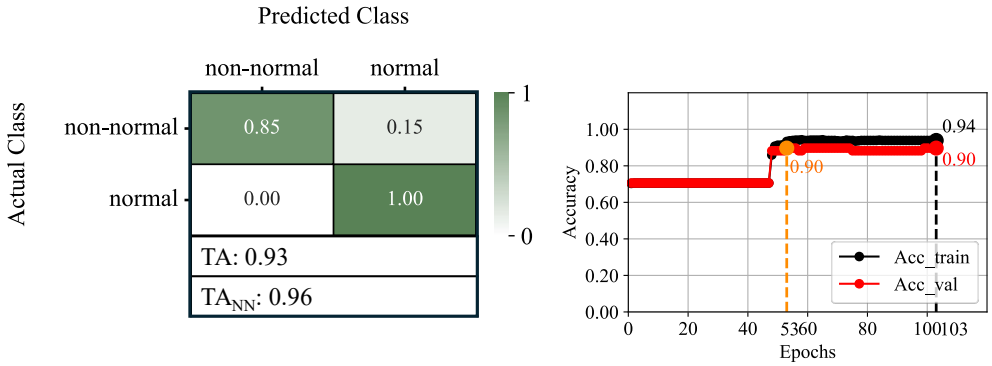
In this case, the PtP and RMS values are extracted from the time-domain signal as features, which means that there are only 2 nodes in the input layer of the NN mode. This significantly reduces both computational and memory requirements. The processed data is quite simplified, and the results obtained using the same NN models are shown in Figure 4.32.



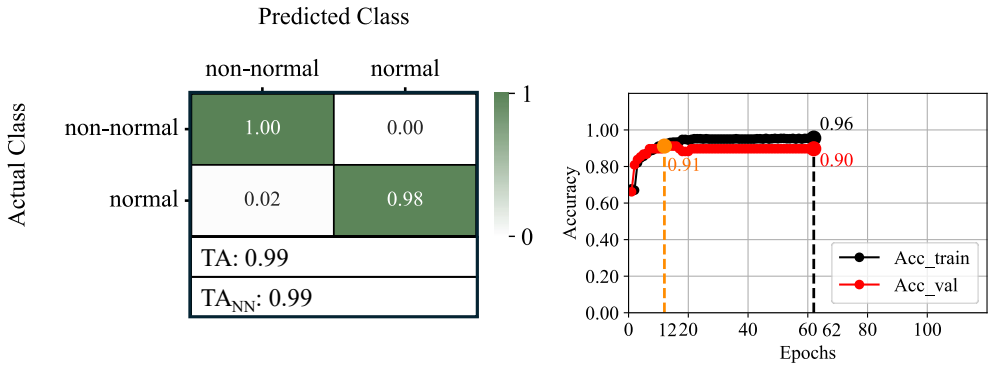
a) n_{hidden}=2, loss=BCE, output activation=Sigmoid



b) n_{hidden}=10, loss=BCE, output activation=Sigmoid



c) n_{hidden}=2, loss=CCE, output activation=SoftMax



d) n_{hidden}=10, loss=CCE, output activation=SoftMax

Figure 4.32: Classification of the non-normal and normal operating points

After applying this signal processing, the simple NNs performed better in classifying normal and non-normal operating points. The results show that, except for n_{hidden} = 2 (Figure 4.32 a) and c)), which did not perform well, the network results for n_{hidden} = 10 are comparable to those obtained with FFT signals, both in terms of accuracy and confusion matrix (Figure 4.32 b) and d)). With increasing network complexity, an accuracy of 1.0 was achieved after 3.79 seconds of training. The results and training curves for n_{hidden} = 128 are shown in Figure 4.33.

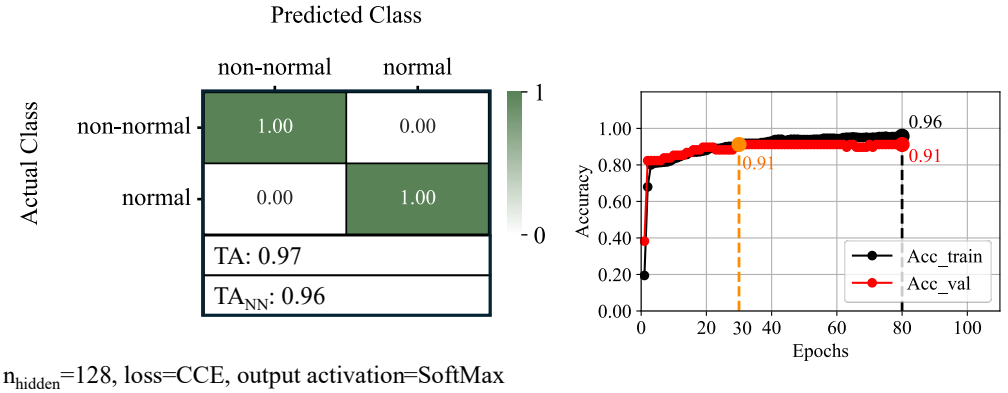


Figure 4.33: Results and training curves of n_{hidden} =128

By reducing the processed signal to just two eigenvalues, which is significantly fewer than the number of FFT components, the training time is greatly shortened. It is observed that the accuracy remains at 1.0, indicating 100% discrimination between normal and non-normal operating points.

4.3.1.3 Utilizing Empirical Mode Decomposition Processed Signal

Performing EMD analysis on the data and extracting multiple IMFs will increase the complexity of the data, resulting in a greater number of eigenvalues, leading to the need for more complex NNs to adequately learn and classify the features within the data. In this section, each IMF of the signal is associated with its corresponding class and fed into NN model training. This approach explores the potential of using a simple NN model to classify the data analyzed through EMD. The results obtained by training different IMFs using the same simple NN models are presented in Figure 4.34.

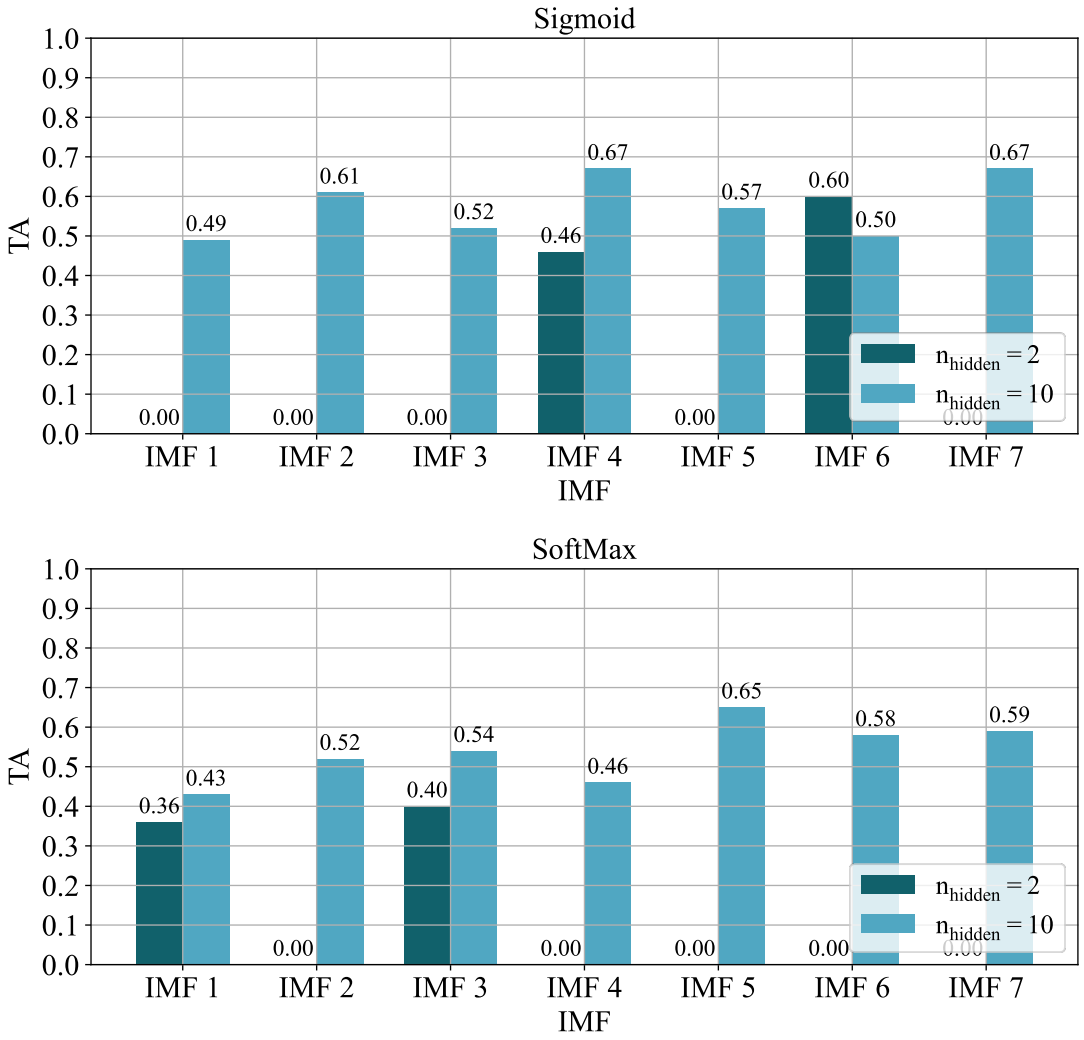
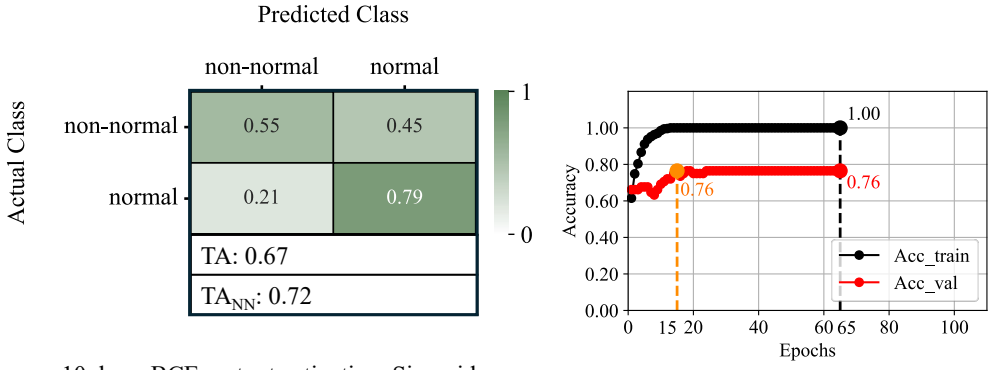


Figure 4.34: TA of the binary classification by training with different IMFs

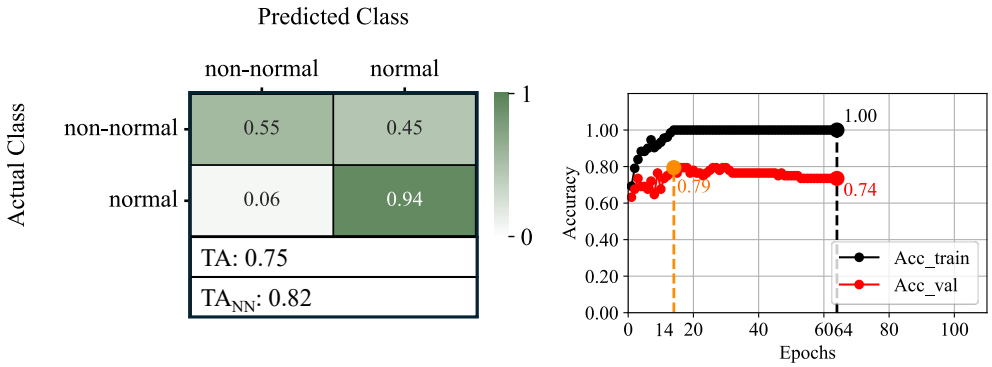
Based on the TA, the data analyzed using EMD demonstrated worse performance when applied to the same simple NN models. The highest TA was achieved by the Sigmoid model with 10 hidden neurons with training data from IMF 4. The results and training curves of this model presents in Figure 4.35.



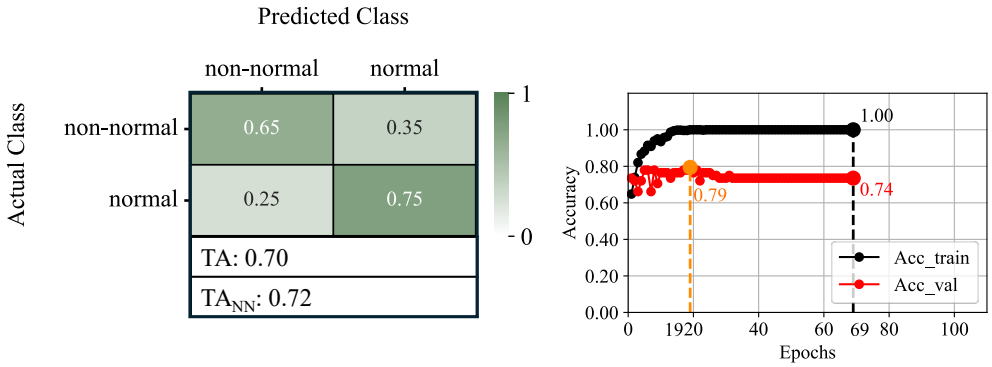
n_{hidden}=10, loss=BCE, output activation=Sigmoid

Figure 4.35: Binary classification by training IMF 4 with simple NN models

The accuracy of 1.0 on the training set indicates that the model performed extremely well on the training set and may have memorized all the training samples. However, it performs poorly on the validation and test sets by the EMD analyzed data. To further explore the potential of using EMD for classification, the complexity of the model was increased, and additional experiments were conducted. The results presented in Figure 4.36.



a) n_{hidden}=1024, loss=BCE, output activation=Sigmoid



b) n_{hidden}=2048, loss=BCE, output activation=Sigmoid

Figure 4.36: Binary classification by training IMF 4 with complex NN models

After increasing the model complexity, the ability to discriminate normal operating points improved by 0.15 when the number of hidden layer nodes increased to 1024, and by 0.25 when the nodes increased to 2048, while the ability to discriminate non-normal remained unchanged or decreased. This indicates that the IMF data itself suffers from insufficient feature partitioning. The NN model was capable of accurately distinguishing classes based on feature values. Therefore, the time-domain data processed by EMD must be further analyzed and transformed to facilitate easier learning of patterns by the NN model.

4.3.2 Rotation Speed and Flow Rate Detection

The rotational speed and the flow rate of the pump can be effectively detected through FFT analysis, as it is reflected in the vibration signal's frequency components. However, time-domain signals do not directly convey frequency information, making it challenging to determine rotational speeds using time-domain data. Due to the poor performance of time-domain signals in distinguishing between normal and non-normal operating points, the value of flow rate interval was increased from 2 m^3/h to 3 m^3/h during this step. The classification of rotational speeds remains unchanged.

4.3.2.1 Utilizing Waveform of Signals

The optimal frequency-domain signal model list in Table 4.19. The training outcomes utilizing the identical NN model on waveform signals are depicted in Figure 4.37, which pertains to rotational speed and in Figure 4.38, which is dedicates to flow rate.

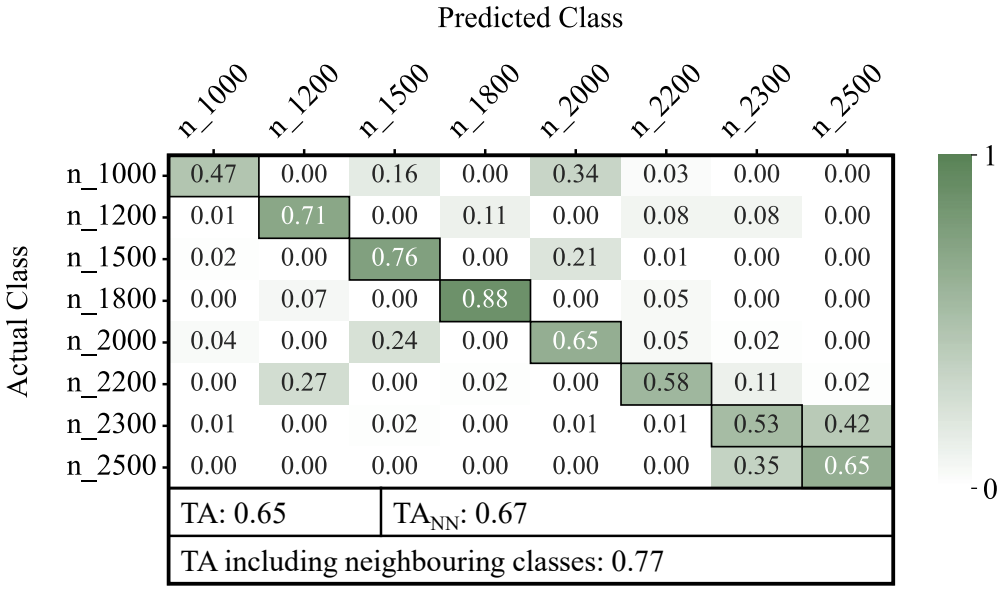


Figure 4.37: Result of rotation speed detection utilizing waveform signals with same NN model as FFT signals

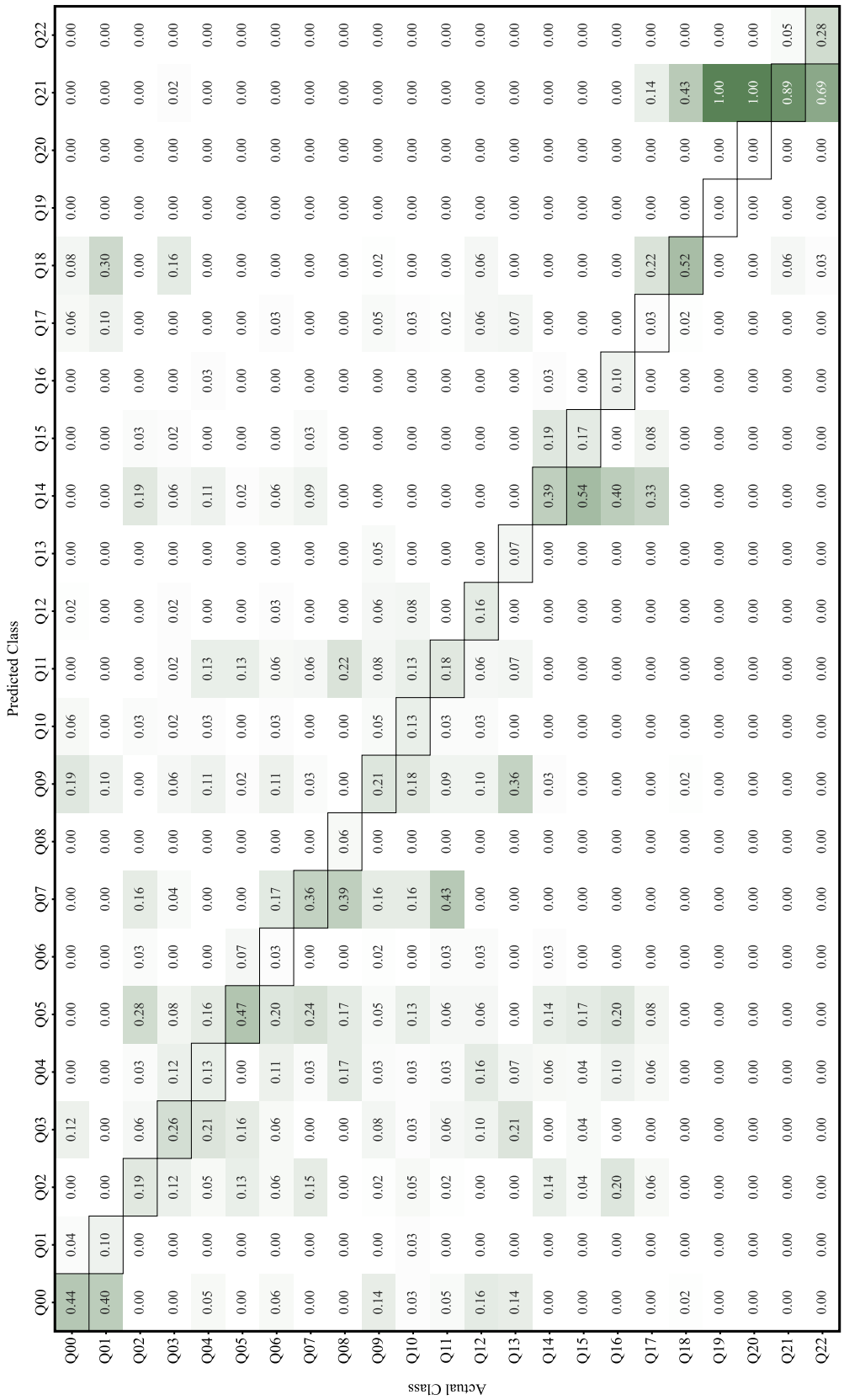


Figure 4.38: Confusion matrix of flow rate detection utilizing waveform signals with same NN model as FFT signals

As might be expected, it is difficult to accurately detects rotation speed and flow rate using the wave form of signals. For rotation speed detection, the accuracy of the NN model is only 0.67 and the TA including neighboring classes achieved only 0.82. In addition, there is a high chance of misclassification between n_2300 and n_2500. And the results of the flow rate detection are even worse. The accuracy of the flow rate detection is only 0.29, the confusion matrix reveals that the accuracy for over 10 classes is nearly negligible, indicating that the model's predictive performance for these classes is minimal. It can be argued that it is not possible to detect flow rate of the test bench in an interval of 3 m^3/h. The best results of after training using the screening program were at n_{hidden} = 2048, a complex single hidden layer NN. However, the accuracies of the two output layers have not improved much either, respectively 0.71 of rotation speed or 0.33 of flow rate.

When the NN model with only one output layer is used to distinguish rotation speed, the accuracy of the NN can be improved to 0.75 at n_{hidden} = 1024 with a training duration time 264.51s. The result is displayed in Figure 4.39.

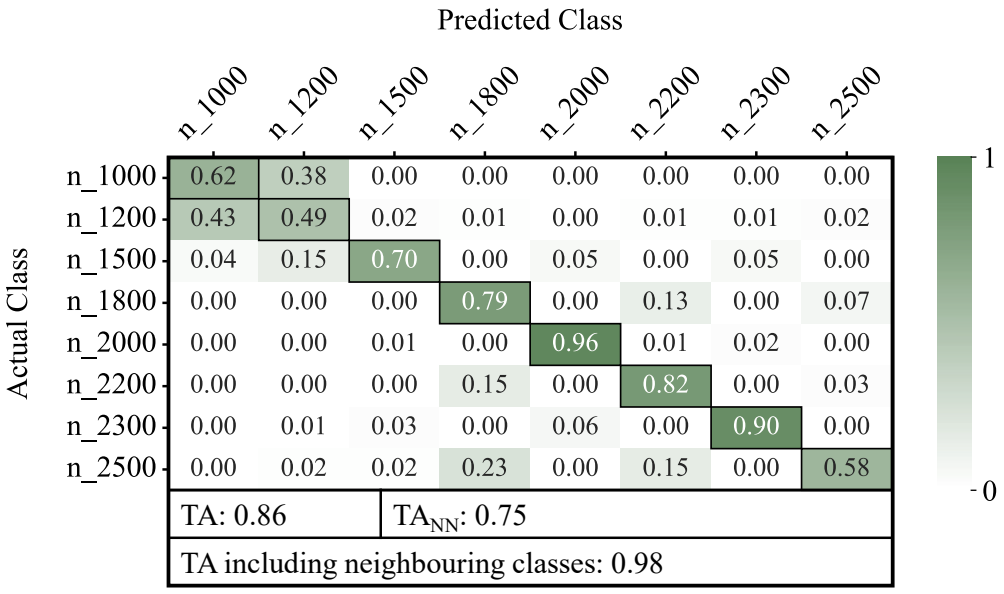


Figure 4.39: Result of rotation speed detection with maximum TA_{NN}

Nonetheless, the n_1000 and n_1200 can easily be confused by the NN model. However, this NN model was able to achieve a TA including neighbouring classes of 0.98, demonstrating the possibility of using the waveform signal for rotation speed detection.

4.3.2.2 Utilizing the Peak-to-peak and Root Mean Square Values of the Signals

The training outcomes utilizing the optimal two output layers NN model of FFT signals on PtP and RMS values of the signals are depicted in Figure 4.40 and Figure 4.41.

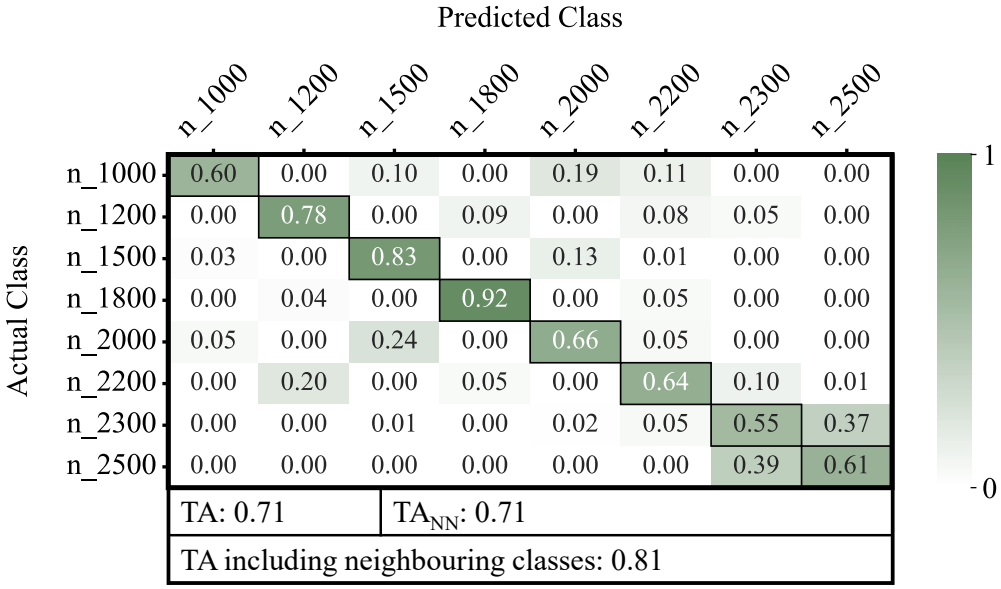


Figure 4.40: Result of rotation speed detection utilizing PtP and RMS values with same NN model as FFT signals

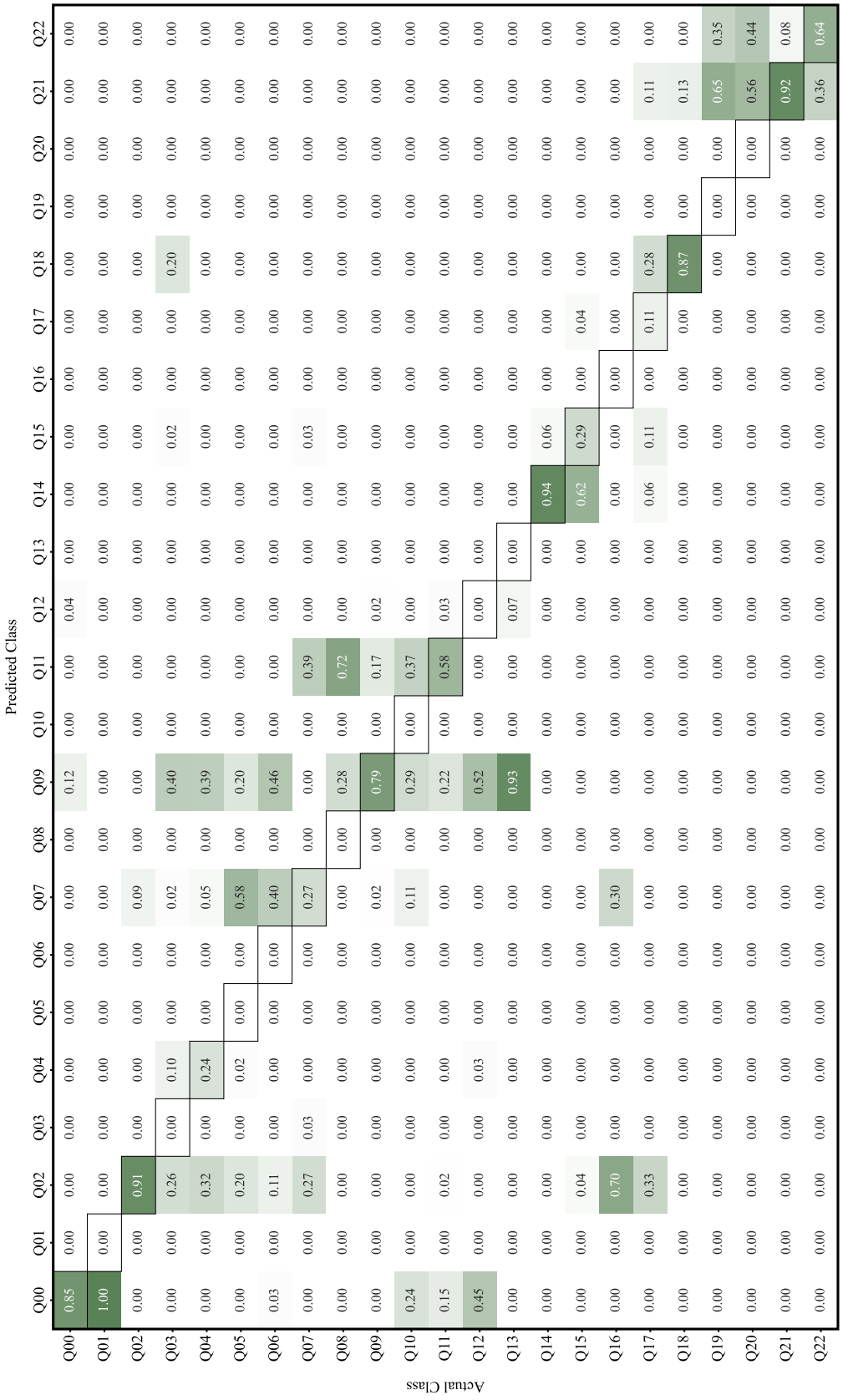


Figure 4.41: Confusion matrix of flow rate detection utilizing PtP and RMS values with same NN model as FFT signals

For rotation speed detection, the accuracy of the NN model is 0.71, the misclassification between n_2300 and n_2500 still exists. The accuracy of the flow rate detection increases to 0.44, however, the confusion matrix reveals that numerous classes exhibit a discriminative capability of 0.

The best results of after training using the screening program were also at n_{hidden} = 2048 with a training time of 449s. The accuracies of the two output layers are respectively 0.77 of rotation speed and 0.52 of flowrate. Incorporating PtP and RMS values into the NN training scheme reduces the computational overhead. However, the temporal dynamics of training indicate that the network requires a greater number of epochs to achieve a superior level of accuracy. From the confusion matrix (Figure 4.42), it is observable that the classification accuracy for the n_2300 class has experienced enhancement relative to the waveform-based result.

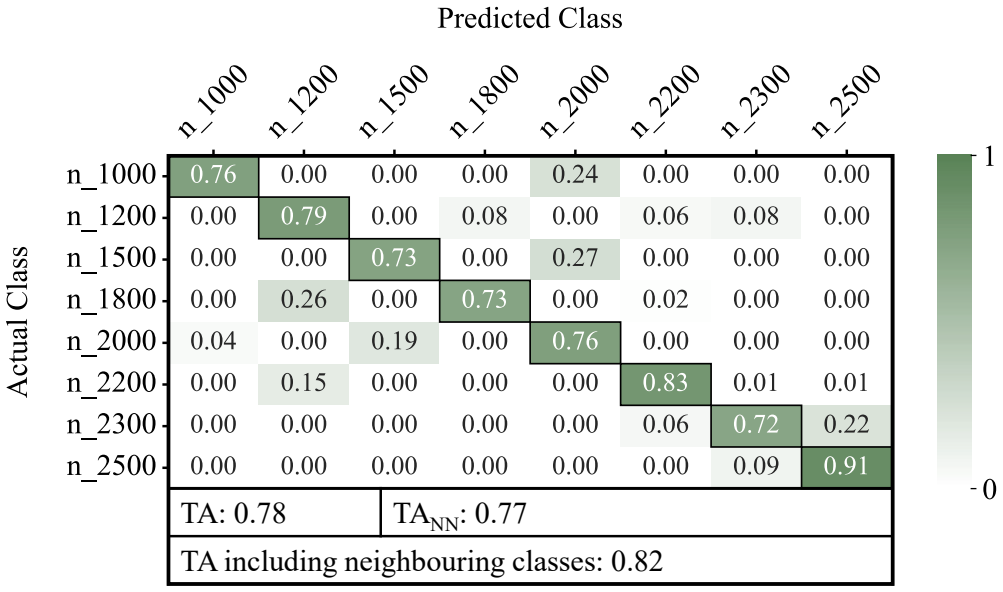


Figure 4.42: Result of rotation speed detection with maximum TA_{NN}

Despite five classes persisting with an accuracy of 0 (Figure 4.43) in the flow rate classification, the network's performance has shown significant improvement compared to the results based on waveform analysis.

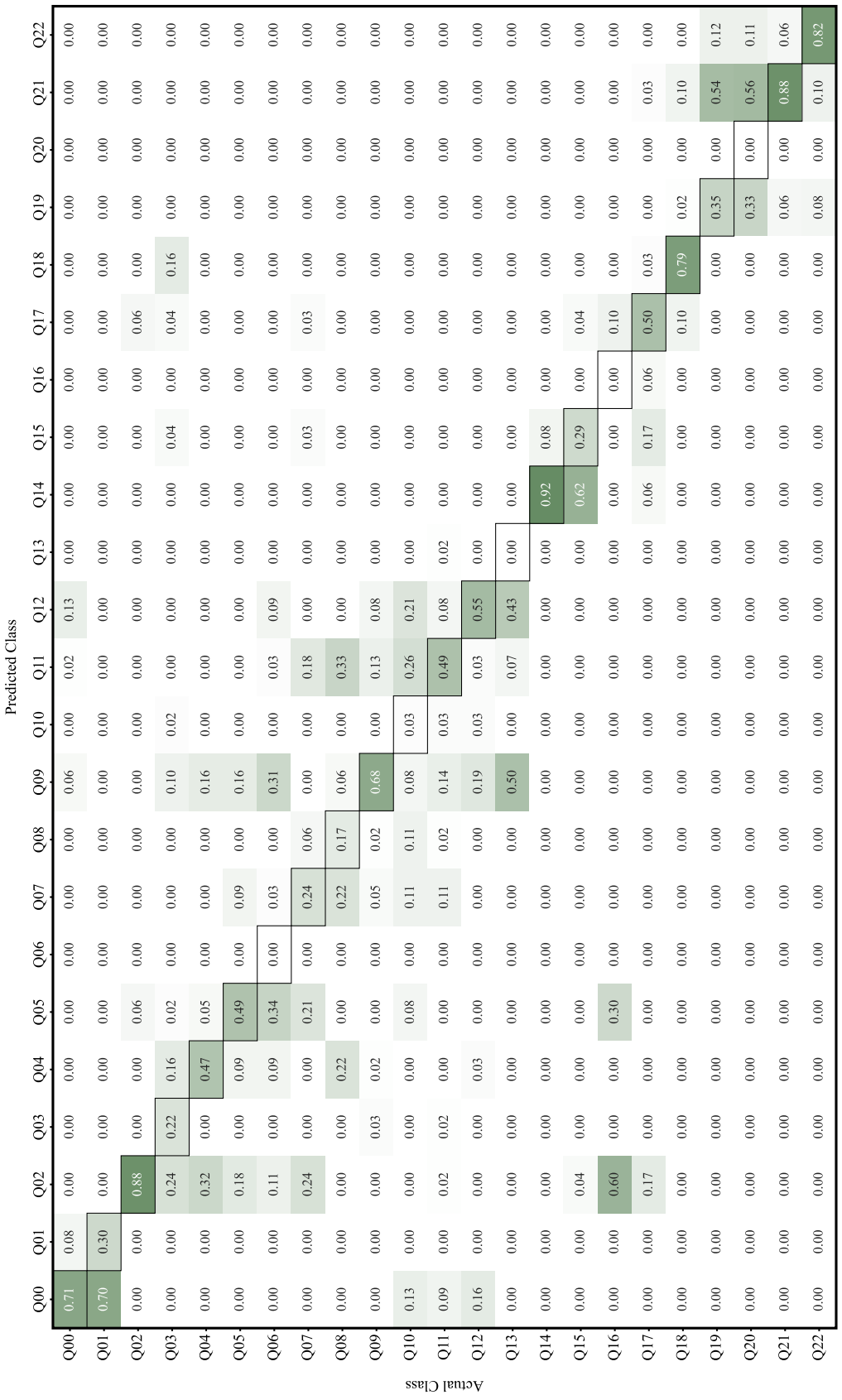


Figure 4.43：Confusion matrix of flow rate detection with maximum TA_{NN}

5 Discussion

The experimental results from the previous section indicate that not only the topology and hyperparameters of the NN influence the performance of the NN classifier, but also the inherent clustering characteristics of the data play a significant role in shaping the results. NNs with a single hidden layer have limited capacity for learning complex nonlinear boundaries. In scenarios where class boundaries are complex or the data distribution is highly non-linear, a single layer architecture may not be sufficient to effectively capture the complex patterns in the data. To clarify the complexity of the classification task, the t-Distributed Stochastic Neighbor Embedding (t-SNE), a dimensionality reduction technique, was employed to visualize the clustering properties of the data in two dimensions [48].

5.1 Impact of the Neural Network Models

In this research, it can be seen a significant interplay between network topology and hyperparameters. Changes in the network architecture often require corresponding adjustments in the hyperparameters to maintain or improve the performance of the NN model. Changes in one aspect can affect the behavior and effectiveness of the other, requiring re-optimization of the hyperparameters to achieve optimal results.

Firstly, adjusting the depth of a NN, such as increasing the number of hidden layers from one to two, introduces additional complexity into gradient propagation and requires a corresponding adjustment in the learning rate. Increasing the number of neurons in the hidden layers increases the learning capacity of the model, but also makes it more susceptible to overfitting. To mitigate this, the dropout rates should be increased accordingly to manage the complexity of the model. However, too high a dropout rate can inhibit the model's ability to learn, resulting in poor performance. Different NN architectures require different optimizers. The results in this dissertation indicate that traditional fully connected networks are more dependent on adaptive learning rate optimizers, such as Adam and RMSProp. The choice of activation function also has a significant impact on the performance of the network. Sigmoid and Tanh functions are more appropriate for simpler networks, while ReLU is more appropriate for complex architectures.

Second, hyperparameter tuning can also influence other parameters, even when the NN topology is fixed. For instance, if the learning rate is adjusted, too small a batch size can lead to unstable training, while too large a batch size can lead to overly smooth training dynamics, resulting in slower convergence. The optimizer determines the method of gradient updating, while the learning rate affects the rate of change during the training process. Certain optimizers exhibit less sensitivity to learning rate adjustments, and overly frequent changes of the learning rate can diminish the optimizer's advantages, adversely affecting the model's convergence. In this study, a larger number of epochs was used in combination with an early stopping strategy. However, if the early stopping strategy is not properly configured, training may be stopped prematurely before the model has fully converged.

In summary, the tuning process of NN models can exhibit a "mutual exclusion" phenomenon, where the network topology and various hyperparameters are not fully independent. To effectively perform hyperparameter optimization, a structured selection process is required. Methods such as grid search, random search, or Bayesian optimization are typically used, along with reliance on experimental results, to identify the optimal combination of hyperparameters for the model.

5.2 Classification of normal and non-normal operating points

The visualization of the datasets still starts by normal and non-normal operating points. Figure 5.1 presents the t-SNE visualization of datasets that have been analyzed using various methods, showcasing the distinct clustering patterns that emerge from each analytical approach.

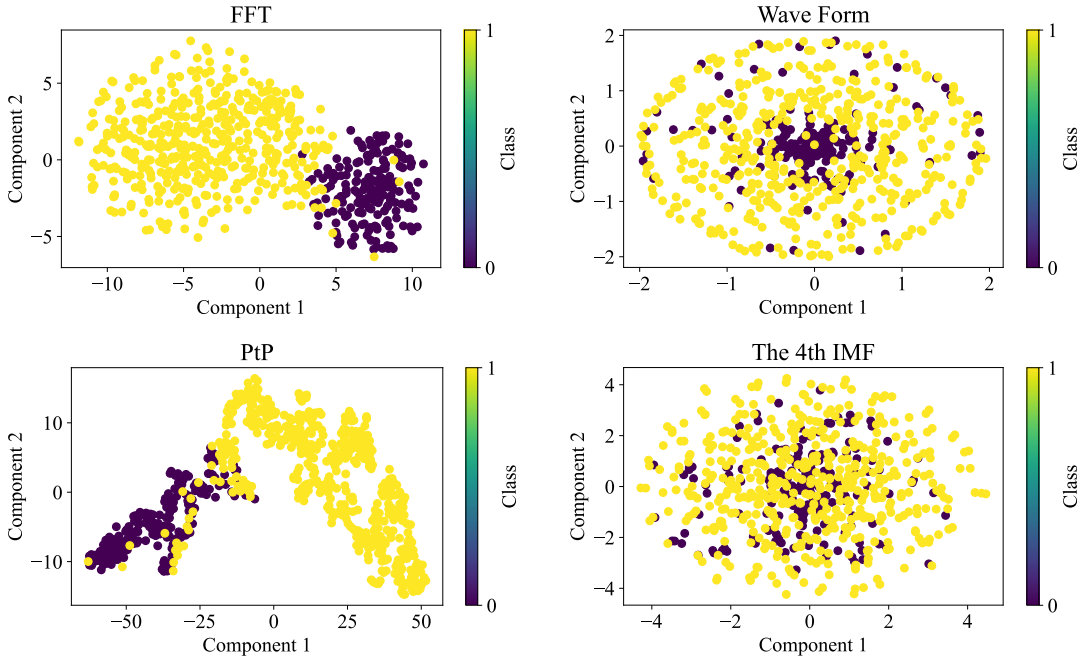


Figure 5.1: The t-SNE of different datasets for binary classification

The NN classifier achieved the best performance on FFT datasets due to the more distinct clustering of classes, as shown in the visualization. The PtP datasets show a reasonable separation between classes. The PtP values provide a simple but informative feature that the NN can use to discriminate between classes to some extent. The classification results using the wave form datasets are less impressive compared to FFT and PtP, with a more overlapping clustering pattern observed. This could mean that the single hidden layer NN was not able to extract the relevant patterns as effectively as with the other datasets. In the IMF datasets, the classes are highly intermixed. The EMD method may not have provided features that are as distinctive or relevant to the classification task at hand, resulting in less accurate separation by the NN.

5.3 Rotation Speed and Flow Rate Detection

The t-SNE visualization presented in Figure 5.2 offers a graphical representation of the datasets utilized for the detection of rotation speeds.

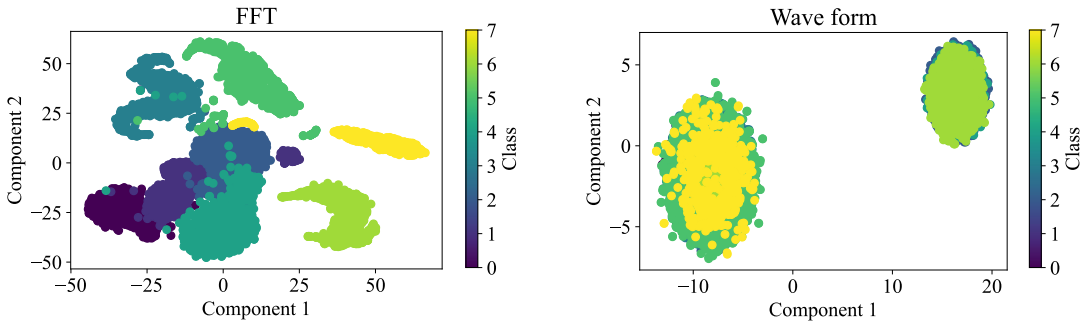


Figure 5.2: The t-SNE of different datasets for rotation speed classification

The FFT-derived data exhibits a more structured pattern. This indicate that the frequency domain features extracted through FFT are effective in distinguishing between different operating points of the datasets. The waveform-derived data appears to have a less distinct clustering pattern, with points seemingly more intermixed. This might suggest that the raw waveform features are less discriminative or that the class boundaries are more ambiguous when using time-domain waveforms directly. Nevertheless, as demonstrated by the results presented in Section 4.3.2, despite the waveform's clustering performance exhibiting considerable overlap among a multitude of data points, the existence of well-defined aggregations within the dataset still offers the possibility of effective classification with a NN. This indicates that, with suitable feature engineering and model tuning, the single hidden layer NN is capable of discerning the subtle differences between classes, leveraging the discernible patterns that exist within the aggregated clusters.

The t-SNE visualization for flowrate classification (Figure 5.3) effectively demonstrates the challenges faced by a single hidden layer NN when attempting to discern the flowrates of test bench using wave form, even after increasing the flow rate interval from 2 m^3/h to 3 m^3/h.

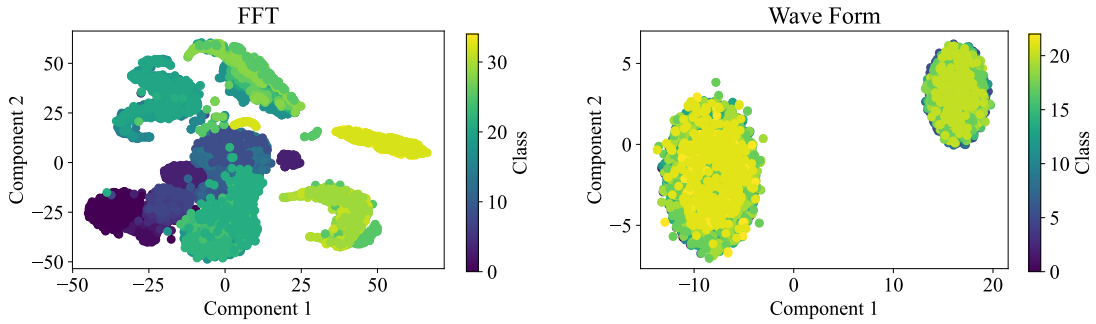


Figure 5.3: The t-SNE of different datasets for flow rate classification (△Q = 2 m^3/h FFT and △Q = 3 m^3/h Wave Form)

The t-SNE of wave form indicates a data distribution characterized by numerous scattered points with extensive overlap. This pattern suggests that the waveform datasets lack the necessary discriminative features for the NN to effectively classify flow rates. As evidenced in the results presented in Section 4.3.2, the inherent limitations of simplistic network architectures become evident when attempting to capture the subtleties necessary for accurate classification. To improve classification performance, it is essential to utilize NNs with deeper architectures to more effectively capture and model the complex relationships within the data.

6 Summary

6.1 Research

In this dissertation, the applicability of simple NN classifiers for detecting the operating points of centrifugal pumps using vibration signals was investigated. The research began with the construction of a pump loop system to serve as the experimental test bench, which included assembling the necessary hardware components and writing the measurement and control software. The hardware setup consisted of a pump loop system equipped with acceleration sensors (ADXL345) mounted on the pump casing to capture vibration signals. Additionally, two sets of measurement and control programs were developed using MATLAB and Python to collect and manage the vibration signal data. These signal data were then used as input data for training and testing the NN models. The collected data consisted of both time-domain signals and frequency-domain. A low-cost hardware setup, priced under €60, was used to ensure the reliability of the data.

This affordable yet effective system provided consistent signal acquisition, demonstrating that reliable data can be achieved even with minimal resources, the focus was placed on exploring how NN classifiers could be employed with minimalistic setups, highlighting the accessibility of the approach.

Following the data collection phase, a model selection program was developed in Python to systematically explore different NN architectures, consisting of an input layer, up to two hidden layers, and output layers, using the TensorFlow framework. The training and evaluation processes, begin with the development of a structured program to assess NN performance across different test series. The experiments utilizing frequency-domain signals focus on assessing the capabilities of single measurement points, recognizing minimal flow rate differences, and evaluating the influence of dataset settings, hidden layers, and hyperparameters on NN accuracies. Through this systematic approach, different NN configurations were evaluated, and their performance was compared across several tasks: classification of normal versus non-normal operating points, rotation speeds detection and flow rate detection. The second half of the experiments shifts focus to time-domain signals, where similar classifications were performed. The discussion of the results synthesizes the findings from both signal domains, comparing their effectiveness in pump operating point detection.

6.2 Conclusion

The results conducted in this dissertation revealed several key insights:

Dataset Construction and Sensor Optimization:

Initial experiments with single-point measurement models demonstrated rapid convergence, achieving an accuracy of 0.99 within 10 epochs. This suggests that, under certain conditions, simplicity in sensor placement can be advantageous. However, the aggregation of data from multiple sensor points did not improvements in model performance. Consequently, sensor position Point 4 was selected for its optimal balance between accuracy and computational efficiency for subsequent analyses. The study also delved into the minimum detectable flow rate difference ($\triangle Q$), establishing that the network's discriminative capabilities remained robust even at $\triangle Q = 1\ m^3/h$.

NN Topologies and Hyperparameters:

The investigation into the impact of NN topologies and hyperparameters on classification accuracy revealed that the selection of the optimizer, learning rate, and batch size is critical, depending on the specific task and available computational resources. To streamline this process, a selection program was developed to automatically identify the optimal NN models for subsequent detection tasks. This program allows for the customization of topologies and hyperparameters to meet various experimental needs. However, certain elements were kept consistent throughout all models. The ReLU activation function was chosen for its computational efficiency and its ability to alleviate issues related to the vanishing gradient problem. For multi-class classification tasks, the hidden layer utilized the SoftMax activation function, paired with the categorical cross-entropy loss function, ensuring robust and accurate performance.

Frequency-Domain Signals based Detection:

In terms of classification between normal and non-normal operating points utilizing frequency-domain signal, the simplest NN model achieved 0.97 accuracy within 22 epochs, with the more complex models demonstrating even better performance. For rotation speed detection, a hidden layer with eight nodes offers a balance between model simplicity, higher accuracy (0.99) and reduced training time (15.74s). For flow rate detection in a complete range of the test bench, using a NN with 512 hidden nodes was

performed with a flow rate interval of 2 m³/h, resulting in a test duration of 192.43 seconds and a TA of 0.85. When including neighboring classes, the TA improved to 0.92. Furthermore, when the flow rate interval was reduced to 3 m³/h and detection was conducted at each rotation speed, the NN classifier achieved a maximum TA of 0.9 at 2500 rpm, with TA including neighboring classes reaching 0.99. For rotation speed and flow rate detection at the same time, the simple NN model achieved an average accuracy of 0.89, with rotation speed detection accuracy reaching 0.99.

Time-Domain Signals based Detection:

Time-domain signals, although theoretically containing all necessary information, presented some limitations in direct classification by simple NN models. However, both time-domain waveform and PtP of the signals were effective in distinguishing between normal and non-normal operating points and rotation speeds. The classification accuracy for these tasks was comparable to that of frequency-domain signals. Time-domain signals showed their usefulness in detecting operating points. However, challenges remained in accurately detecting flow rates, with the accuracy reaching only 0.5 at its best. This underscores the potential of time-domain data, especially for simpler tasks, while also indicating that improving the depth and complexity of the NN could be beneficial for more detailed analyses and achieving higher performance in more challenging tasks.

Hardware Setup and Practicality:

An essential aspect of this research was demonstrating that effective pump condition detection could be achieved with a low-cost hardware setup. The entire hardware system for vibration signal acquisition was constructed for less than €60, emphasizing the accessibility of this approach. Despite the minimal investment, the system provided consistent and reliable signal data, proving that resource constraints do not necessarily limit the potential for accurate classification when using simple NN models.

In summary, this research confirms the viability of using simple NN classifiers for centrifugal pump operating point detection, especially when using frequency-domain signals. The findings suggest that even with a minimalistic setup and basic NN models, accurate detection of rotation speeds and flow rates at the same time is possible. However, the results also indicate that further refinement and more complex feature processing may be necessary when working with time-domain signals to achieve comparable performance.

The outcomes of this study provide a solid foundation for future research in equipment condition monitoring using affordable and accessible NN-based approaches.

6.3 Outlook

Building upon the conclusions of this dissertation, future research should be divided into two primary areas. First, improving NN classifiers by exploiting deeper and more complex architectures, such as CNNs and RNNs, can significantly enhance feature extraction from time-domain signals. Exploring self-supervised and multi-task learning approaches could further improve the model's ability to capture intricate signal patterns and dependencies, thereby boosting the accuracy of operating point detection. Extending these classifiers to include early fault detection and diagnosis would greatly increase the value of the system in predictive maintenance strategies. Additionally, the transferability of this method to other domains, such as different types of rotating machinery or even non-mechanical applications, opens the door to broader applications. Developing intuitive user interfaces for these systems will ensure that operators without extensive technical expertise can efficiently interact with and fully benefit from this advanced technology.

In parallel, practical applications should focus on developing cost-effective and efficient real-time monitoring systems using devices like the Raspberry Pi. The Raspberry Pi, functioning as a microcomputer, is capable of executing TensorFlow Lite. Although its computational power may not suffice for training or deploying very complex NN models, it efficiently handles the execution of pre-trained simple classifiers offloaded from a computer. This makes the Raspberry Pi a practical solution for implementing operating point detection systems in centrifugal pumps, offering a cost-effective approach to real-time monitoring and diagnostics. Optimizing deep learning models for edge computing would allow more complex models to be deployed directly on such hardware, minimizing reliance on PC-based processing. Real-time data streaming and analysis systems could provide instant feedback, improving proactive maintenance capabilities. Additionally, studies into the energy efficiency of NN models running on microcomputers could contribute to the development of sustainable monitoring systems that minimize power consumption while maintaining high performance. Finally, studies on the long-term reliability and stability of these systems in industrial environments are essential to ensure trustworthiness and robustness in critical applications.

Bibliography

[1] B. Nikhil, Fundamentals of Deep Learning, O'Reilly Media Inc., 2017.

[2] K. Wang, Y. Zhang and J. Li, A Review of Research on Pump Fault Diagnosis, Pump Technology, 2007.

[3] D. J. Ewins, Modal Testing: Theory, Practice and Application, vol. 22, Research Studies Press Ltd., 2000, pp. 683-723.

[4] Q. Yu and X. Han, Principle and Application of Modern Mechanical Fault Diagnosis Based on Vibration Analysis, BeiJing: Science Press, 2010.

[5] P. Henriquez, J. B. Alonso, M. A. Ferrer and C. M. Travieso, "Review of Automatic Fault Diagnosis Systems Using Audio and Vibration Signals," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 44, no. 5, pp. 642-652, 2014.

[6] J. Quinlan, "Induction of Decision Trees," *Machine learning,* no. 1, p. 81–106, 1986.

[7] C. Licht, Entwicklung eines Betriebspunktmonitoringsystems für Kreiselpumpen mit Hilfe der Signale eines Beschleunigungssensors auf Basis des Maschinellen Lernens, Kaiserslautern: Verlag Technische Universität Kaiserslautern, 2016.

[8] D. Huhn, Störungsfrüherkennung an Wellendichtungslosen Pumpen durch Bauteilintegrierte Sensorik, Kaiserslautern: Verlag Technische Universität Kaiserslautern, 2001.

[9] D. Kollmar, Störungsfrüherkennung an Kreiselpumpen mit Verfahren des Maschinellen Lernens, Kaiserslauter: Verlag Technische Universität Kaiserslautern, 2002.

[10] K. R. Nuglisch, Entwicklung eines Anlagenunabhängigen Störungsfrüherkennungssystems für Pumpen auf der Basis des Maschinellen Lernens, Kaiserslautern: Verlag Technische Universität Kaiserslautern, 2006.

[11] M. Kiggen, Störungsfrüherkennung an Kreiselpumpen mit Hilfe Bauteilintegrierter Sensorik und Verfahren der Explorativen Datenanalyse, Verlag Technische Universität Kaiserslautern, 2007.

[12] J. Feng and M. Böhle, "Feasibility Study for the Application of a Neural Network for Operating Condition Detection of a Centrifugal Pump," *Journal of Physics: Conference Series,* vol. 1909, no. 012071, 2021.

[13] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016.

[14] M. Chui, J. Manyika and M. Miremadi, Where Machines Could Replace Humans—And Where They Can't (Yet), McKinsey Quarterly, 2016.

[15] T. H. Davenport and R. Ronanki, Artificial Intelligence for the Real World, Harvard Business Review, 2018.

[16] P. Gangsar and R. Tiwari, "Signal-Based Condition Monitoring Techniques for Fault Detection and Diagnosis of Induction Motors: A State-of-the-Art Review," *Mechanical Systems and Signal Processing,* vol. 144, no. 106908, pp. 205-219, 2020.

[17] T. Miller, "Explanation in Artificial Intelligence: Insights from the Social Sciences," *Artificial Intelligence,* vol. 267, pp. 1-38, 2019.

[18] F. M. Johnson and K. Ashby, Materials and Design: The Art and Science of Material Selection in Product Design (3rd ed.), Butterworth-Heinemann, 2013.

[19] D. S. Nau, M. Ghallab and P. Traverso, Automated Planning: Theory and Practice, Morgan Kaufmann, 2004.

[20] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[21] R. Liu, B. Yang, E. Zio and X. Chen, "Artificial Intelligence for Fault Diagnosis of Rotating Machinery: A Review," *Mechanical Systems and Signal Processing,* vol. 108, pp. 33-47, 2018.

[22] D. A. Waterman, A Guide to Expert Systems, Pearson Education, 1986.

[23] T. Wei, S. Yang and H. Liu, "Integration of Expert Systems and PID Control for Oil Pump Devices," *Journal of Mechanical Engineering,* vol. 12, no. 57, pp. 345-358, 2021.

[24] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," *In Proceedings of the ACM International Conference on Multimedia (MM '14),* vol. 5, no. 34, pp. 675 - 678, 11 2014.

[25] M. Wang, T. Lin, K. Jhan and S. Wu, "Abnormal Event Detection, Identification and Isolation in Nuclear Power Plants Using LSTM Networks," *Progress in Nuclear Energy,* vol. 140, no. 103928, 2021.

[26] M. Khan, T. Liu and F. Ullah, "A New Hybrid Approach to Forecast Wind Power for Large Scale Wind Turbine Data Using Deep Learning with TensorFlow Framework and Principal Component Analysis," *Energies,* vol. 12, no. 12, p. 2229, 2019.

[27] J. F. Gülich, Kreiselpumpen: Handbuch für Entwicklung, Anlagenplanung und Betrieb, Springer-Verlag, 2010.

[28] C. A. Walshaw, "Mechanical Vibrations With Applications," New York, Halsted Press, 1984, p. 21.

[29] J. M. Vance, F. . Y. Zeidan and B. . G. Murphy, Machinery Vibration and Rotordynamics, John Wiley & Sons, 2010.

[30] I. O. f. Standardization, "ISO 20816-1:2016(en) Mechanical vibration-Measurement and evaluation of machine vibration," ISO, 2016.

[31] S. H. Mneney, An Introduction to Digital Signal Processing: A Focus on Implementation, River Publishers, 2009.

[32] K. W. Bonfig and Z. Liu, Virtuelle Instrumente und Signalverarbeitung: Zustandsüberwachung und Diagnose an Maschinen mit LabView by Karl Walter Bonfig, VDE-Verlag, 2004.

[33] J. V. d. Vegte, Fundamentals of Digital Signal Processing, Pearson Education Asia limited and Publishing House of Electronics Industry, 2002.

[34] K.-D. Kammeyer and K. Kroschel, Digitale Signalverarbeitung, Teubner, 2009.

[35] N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Shih, Q. Zheng, N.-C. Yen, C. C. Tung and H. H. Liu, "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences,* vol. 454, no. 1971, pp. 903-995, 1998.

[36] J. Awrejcewicz, Numerical Simulations of Physical and Engineering Processes, IntechOpen, 2011.

[37] K. Suzuki, Artificial Neural Networks : Methodological Advances and Biomedical Applications, IntechOpen, 2011.

[38] J. Patterson and A. Gibson, Deep Learning, USA: O'Reilly Media, 2017.

[39] R. Kruse, C. Borgelt, C. Braune, F. Klawonn, C. Moewes and M. Steinbrecher, Computational Intelligence, Springer Vieweg, 2015.

[40] A. Géron, Hands-on machine learning with Scikit-Learn and TensorFlow, O'Reilly UK Ltd., 2017.

[41] R. Sebastian and V. Mirjalili, Python Machine Learning, Second Edition: Machine Learning and Deep Learning with Python, Scikit-Learn, and TensorFlow, Packt Publishing, 2017, pp. 807-814.

[42] A. Y. LeCun, L. Bottou, G. B. Orr and K.-R. Müller, "Efficient BackProp," in *Neural Networks: Tricks of the Trade*, Berlin, Heidelberg, Springer, 2012, pp. 9-48.

[43] G. James, D. Witten, T. Hasitie and R. Tibshirani, An Introduction to Statistical Learning: with Applications in Python, Springer, 2023.

[44] A. EI-Shahat, Advanced Applications for Artificial Neural Networks, IntechOpen, 2018.

[45] V. F. Veen, "The Asimov Institute," 14 09 2016. [Online]. Available: https://www.asimovinstitute.org/neural-network-zoo/. [Accessed 08 07 2024].

[46] T. Hastie, R. Tibshirani and J. Friedma, The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.), Springer, 2009.

[47] K. P. Murphy, Machine Learning : A Probabilistic Perspective, MIT Press, 2012.

[48] S. Guido and A. Müller, Introduction to machine learning with Python: A guide for data scientists, O'Reilly Media, 2016.

[49] F. Chollet, "Keras: The Python Deep Learning library," 2015. [Online]. Available: https://keras.io/. [Accessed 16 07 2024].

[50] J. M. Hughes, Arduino: A Technical Reference, O'Reilly Media, Inc., 2016.

[51] "ARDUINO.CC," 15 09 2021. [Online]. Available: https://docs.arduino.cc/hardware/mega-2560/. [Accessed 22 07 2024].

[52] "YwRobot Studio Wiki," YwRobot, [Online]. Available: http://wiki.ywrobot.net/. [Accessed 22 07 2024].

[53] R. Pi, "Raspberry Pi," Raspberry Pi , [Online]. Available: https://www.raspberrypi.org. [Accessed 23 07 2024].

[54] "Data Sheet ADXL345," 26 10 2015. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/adxl345.pdf. [Accessed 24 07 2024].

Appendixes

A Calibration of the Control and Measurement Program

Table 1: Voltage valve of the test bench

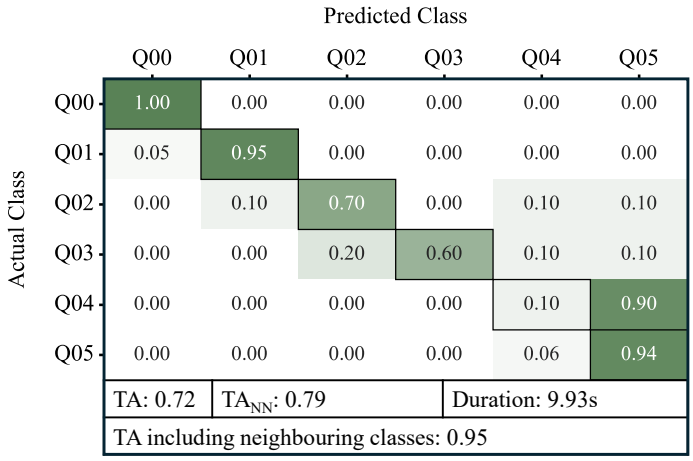
Program							
No.	n [rpm]	Voltage-Pin	Valve	Q [V]	P1 [V]	P2 [V]	M [V]
1	2516.545	3.60214976	100%	2.78335	0.677012	1.016785	2.159665
	1994.083	3.27421751		2.198925	0.835859	0.684588	1.734035
	1496.465	2.96187861		1.60329	0.953242	0.42522	1.291625
	1195.38	2.77289731		1.043393	1.033457	0.217226	0.850929
	1000.697	2.6507009		0.617791	1.06761	0.10712	0.52395
2	2516.545	3.60214976	90%	2.711875	0.690942	1.07307	2.163
	1994.083	3.27421751		2.162265	0.8493	0.725155	1.72996
	1496.465	2.96187861		1.57478	0.962855	0.452347	1.29089
	1195.38	2.77289731		1.02672	1.03845	0.220702	0.848811
	1000.697	2.6507009		0.608831	1.07403	0.110052	0.522891
3	2516.545	3.60214976	80%	2.59604	0.729146	1.217255	2.16569
	1994.083	3.27421751		2.07356	0.863636	0.805067	1.72882
	1496.465	2.96187861		1.51336	0.973363	0.49104	1.28796
	1195.38	2.77289731		0.983219	1.044043	0.240632	0.851146
	1000.697	2.6507009		0.583985	1.07535	0.116325	0.524927
4	2516.545	3.60214976	70%	2.343925	0.800343	1.46033	2.1564
	1994.083	3.27421751		1.871375	0.912339	0.964728	1.730855
	1496.465	2.96187861		1.36836	0.998697	0.568182	1.28886
	1195.38	2.77289731		0.891278	1.049313	0.275063	0.850711
	1000.697	2.6507009		0.530629	1.076	0.128136	0.523887
5	2516.545	3.60214976	60%	1.95715	0.885386	1.78918	2.16683
	1994.083	3.27421751		1.569565	0.964321	1.16373	1.736315
	1496.465	2.96187861		1.153065	1.02297	0.673428	1.29513
	1195.38	2.77289731		0.750896	1.068103	0.318073	0.8512
	1000.697	2.6507009		0.442815	1.082925	0.143044	0.525253
6	2516.545	3.60214976	50%	1.475805	0.983383	2.071115	2.15347
	1994.083	3.27421751		1.176685	1.011	1.344575	1.73876
	1496.465	2.96187861		0.868524	1.05474	0.760264	1.2957
	1195.38	2.77289731		0.559384	1.077465	0.348974	0.852884
	1000.697	2.6507009		0.332356	1.082355	0.156159	0.528104
7	2516.545	3.60214976	40%	1.360705	1.008065	2.131965	2.16227
	1994.083	3.27421751		1.07454	1.031525	1.388075	1.736075
	1496.465	2.96187861		0.804497	1.06134	0.782747	1.294475
	1195.38	2.77289731		0.520772	1.07576	0.358749	0.849218

	1000.697	2.6507009		0.310606	1.085045	0.153715	0.524438

Table 2: Display valve of the test bench

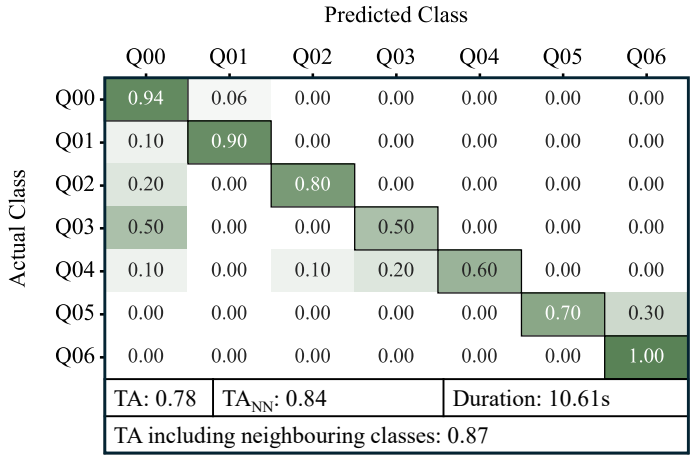
Display							
Num	n[rpm]	Valve	Q [m^3/h]	P1 [bar]	P2 [bar]	M [N·m]	Q [m^3/h]
1	2500	100%/ 97.6%	105.65	-28450	103700	21.205	5.53
	2000		84.09	-16000	69800	13.815	2.89
	1500		62.17	-5900	42050	7.95	1.24
	1000		40.785	1450	22075	3.745	0.385
	620		24.405	5035	11935	1.615	0.1
2	2500	90%/ 87.6%	104.685	-27750	106350	21.215	5.51
	2000		83.15	-15750	71800	13.82	2.905
	1500		61.52	-5650	43100	7.96	1.245
	1000		40.32	1550	22550	3.74	0.38
	620		24.05	5015	12060	1.61	0.1
3	2500	80%/ 78.3%	99.075	-24500	120400	21.155	5.505
	2000		79.075	-13300	80450	13.78	2.875
	1500		58.605	-4575	47800	7.9	1.235
	1000		38.385	2000	24575	3.735	0.385
	620		21.53	5380	12425	1.555	0.095
4	2500	70%/ 68.2%	91.24	-20200	139500	20.745	5.41
	2000		72.23	-10700	93300	13.485	2.82
	1500		53.67	-2550	54950	7.76	1.21
	1000		35.09	2860	27500	3.65	0.375
	620		21.02	5690	13800	1.6	0.1
5	2500	60%/ 58.8%	77.06	-12600	169500	19.77	5.145
	2000		61.74	-5575	111700	12.92	2.705
	1500		45.635	-75	65300	7.4	1.16
	1000		29.91	3990	31800	3.49	0.36
	620		17.795	5975	15200	1.52	0.095
6	2500	50%/ 48.6%	58.33	-4300	197200	17.625	4.595
	2000		46.62	-316.667	129366.7	11.47333	2.398333
	1500		34.61667	2960	74683.33	6.596667	1.036667
	1000		22.55	5283.333	35550	3.083333	0.32
	620		13.435	6430	16475	1.335	0.08
7	2500	39%	52.4	-2100	206000	16.85	4.37
	2000		41.775	1150	134400	10.96	2.295
	1500		31.195	3775	76400	6.23	0.985
	1000		20.275	5650	36200	2.875	0.295
	620		12.315	6600	16700	1.27	0.08

B Confusion Matrix of Flow Rate Detection at Each Rotation Speed



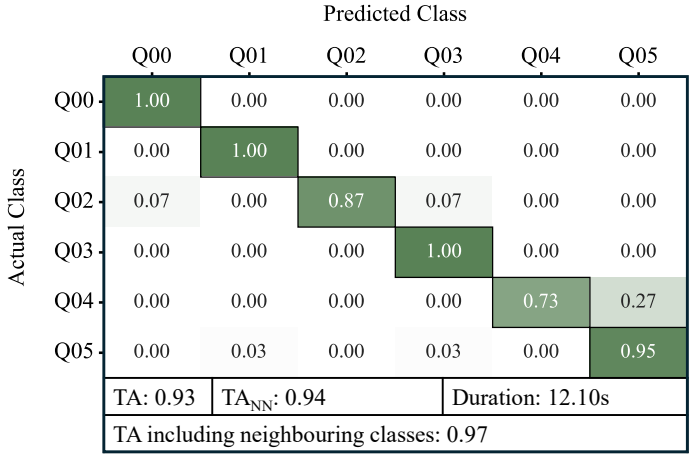
Classes	Flow rate [m³/h]	Sample count
Q 00	30	100
Q 01	32	203
Q 02	33	100
Q 03	34	101
Q 04	35	101
Q 05	36	303

Figure 1: Flow rate detection at n = 1000 rpm



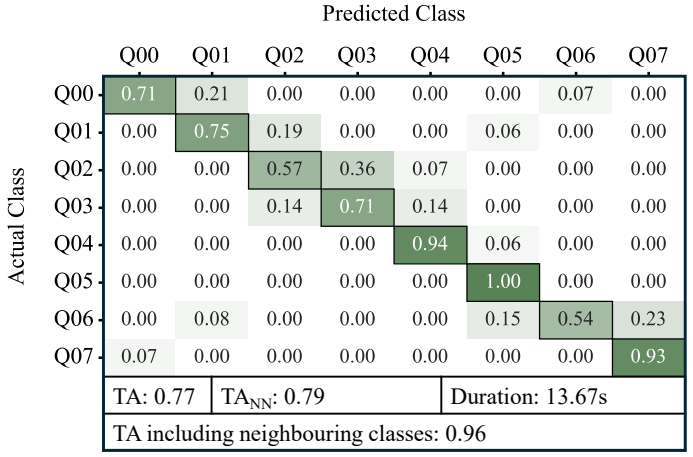
Classes	Flow rate [m³/h]	Sample count
Q 00	34	155
Q 01	37	101
Q 02	39	103
Q 03	41	101
Q 04	43	101
Q 05	44	100
Q 06	45	304

Figure 2: Flow rate detection at n = 1200 rpm



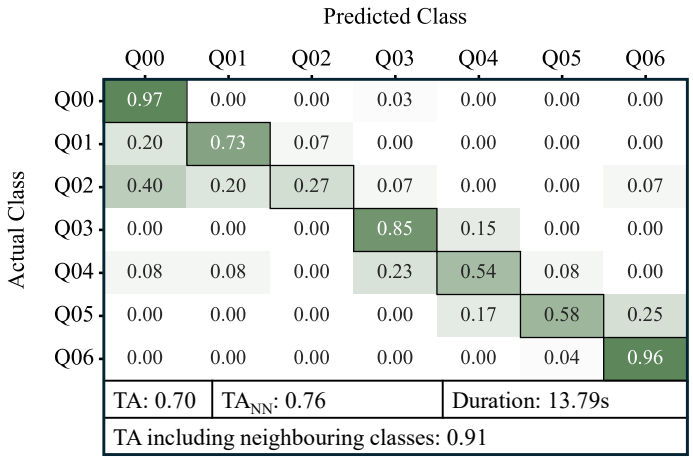
Classes	Flow rate [m³/h]	Sample count
Q 00	47	203
Q 01	51	167
Q 02	54	149
Q 03	56	120
Q 04	57	106
Q 05	58	400

Figure 3: Flow rate detection at n = 1500 rpm



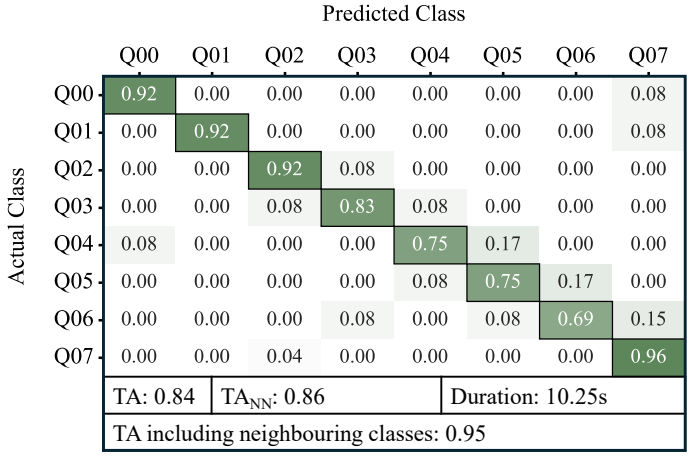
Classes	Flow rate [m³/h]	Sample count
Q 00	61	142
Q 01	64	163
Q 02	65	141
Q 03	67	138
Q 04	70	155
Q 05	72	114
Q 06	73	128
Q 07	74	276

Figure 4: Flow rate detection at n = 1800 rpm



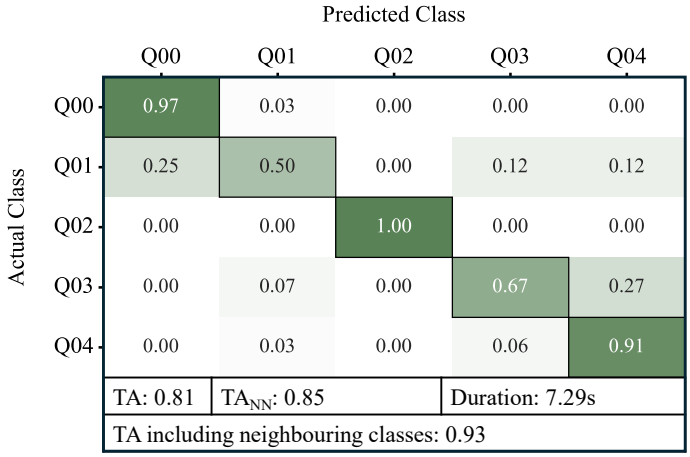
Classes	Flow rate [m³/h]	Sample count
Q 00	67	284
Q 01	70	154
Q 02	74	144
Q 03	77	134
Q 04	79	133
Q 05	80	123
Q 06	81	258

Figure 5: Flow rate detection at n = 2000 rpm



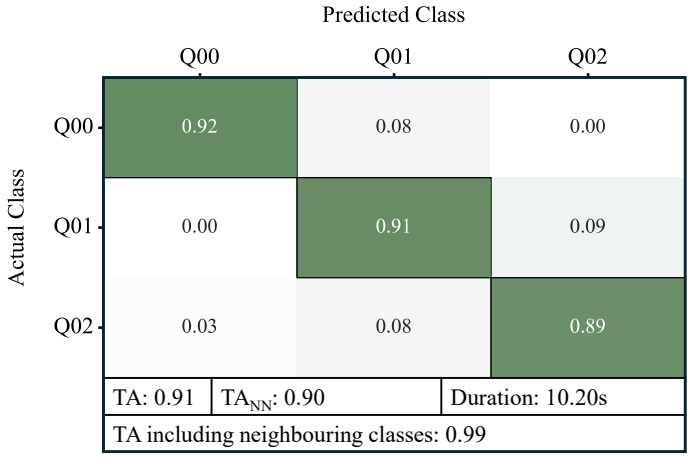
Classes	Flow rate [m³/h]	Sample count
Q 00	67	122
Q 01	73	118
Q 02	78	114
Q 03	82	122
Q 04	86	116
Q 05	88	123
Q 06	90	133
Q 07	91	256

Figure 6: Flow rate detection at n = 2200 rpm



Classes	Flow rate [m³/h]	Sample count
Q 00	87	350
Q 01	89	162
Q 02	91	105
Q 03	93	155
Q 04	94	330

Figure 7: Flow rate detection at n = 2300 rpm



Classes	Flow rate [m³/h]	Sample count
Q 00	101	124
Q 01	103	112
Q 02	104	363

Figure 8: Flow rate detection at n = 2500 rpm

C List of the Program Codes

No.	Program	Language	Platform
1	Control and measurement program (MATLAB)	MATLAB	PC, Arduino
2	Control and measurement program (Python)	Python	Raspberry Pi
3	NN training and evaluation program	Python	PC
4	Independent vibration signal acquisition program	Python	PC / Raspberry Pi

List of Figures

Figure 2.1: Representation of components of an impeller pump....8

Figure 2.2: Three characteristic curves of a pump....9

Figure 2.3: The influence of a rotation speed variation on the H - Q curve....11

Figure 2.4: The system characteristic curve due to valve control....11

Figure 2.5: Process of an analog-to-digital conversion....16

Figure 2.6: Original time-domain vibration signal in two different classes....17

Figure 2.7: Statistical analysed vibration signal in two different classes....18

Figure 2.8: EMD Process....19

Figure 2.9: EMD of the vibration signal in two different classes....20

Figure 2.10: Frequency-domain vibration signal in two different classes....23

Figure 2.11: Data recording, data processing and classification....23

Figure 2.12: The basic unit of a biological brain neuron [40]....26

Figure 2.13: Working principle of an artificial neuron....27

Figure 2.14: Activation functions....28

Figure 2.15 SoftMax function....30

Figure 2.16: Layer structure of typical NNs....30

Figure 2.17: Model of a NN....33

Figure 2.18: Iteration process of a NN....36

Figure 2.19: Different architectures of NNs [45]....41

Figure 2.20: Multi-class classification strategies....43

Figure 2.21: NN classifier training flow chart....44

Figure 2.22: Data labelling for NN model training....45

Figure 2.23: Parallel computation on multiple CPUs/GPUs/servers [40]....47

Figure 3.1: 3D-representation of the test bench of JHU....50

Figure 3.2: Drive components of the test pump of JHU.....51

Figure 3.3: Control and measuring box based on Arduino MEGA 2560.....54

Figure 3.4: Control board pins assignment (Arduino) 55

Figure 3.5: Control box panels diagram (Arduino).....57

Figure 3.6: Control and measuring box base on Raspberry Pi 58

Figure 3.7: Control board pins assignment (Raspberry Pi) 59

Figure 3.8: Control box front panel diagram (Raspberry Pi).....60

Figure 3.9: Vibration signal acquisition of the test pump 60

Figure 3.10: Flow chart of the control and measurement software 64

Figure 3.11: Graphical user interface of the control program (MATLAB).....65

Figure 3.12: Graphical user interface of the control program (Python).....67

Figure 3.13: Settings of automatic measurement and control program 68

Figure 3.14: Vibration signal at Q = 104 m^3/h, n = 2500 rpm.....70

Figure 3.15: Data structure of a processed dataset.....71

Figure 3.16: NN topology of the test process 72

Figure 4.1: Accuracy per epoch of the NN model 76

Figure 4.2: Confusion matrix with absolute classes division 77

Figure 4.3: Confusion matrix with relative classes division.....78

Figure 4.4: Sensor positions.....80

Figure 4.5: Simplified training results of the sensor positions 82

Figure 4.6: Training results of different $\triangle Q$.....85

Figure 4.7: Accuracy of the NN model with different dataset height 87

Figure 4.8: Training results of the total dataset 88

Figure 4.9: Accuracy of the NN model with different dataset length 89

Figure 4.10: Effect of n_{hidden} on TA_{NN} 90

Figure 4.11: Epochs to reach the maximum TA_{NN} at different n_{hidden}.....91

Figure 4.12: Effect of n_{hidden} in 2 hidden layers on TA_{NN} 91

Figure 4.13: Training curve of different batch sizes 92

Figure 4.14: Effect of batch size on TA_{NN} 93

Figure 4.15: Training curve using activation function of ReLU and Tahn 93

Figure 4.16: Effect of initial learning rate on TA_{NN} 94

Figure 4.17: Training curve by initial learning rate at 0.001 and 0.002 95

Figure 4.18: Effect of different optimizer on TA_{NN} 95

Figure 4.19: Effect of different dropout rate on TA_{NN} 96

Figure 4.20: Training curve by dropout rate at 0 and 0.4 96

Figure 4.21: Classification of the non-normal and normal operating points 98

Figure 4.22: Maximum TA_{NN} of the rotation speed classification with different n_{hidden} 99

Figure 4.23: Training curves of the NN model with $n_{hidden} = 8$ and $n_{hidden} = 256$ 100

Figure 4.24: Confusion matrix of the NN model with $n_{hidden} = 8$ and $n_{hidden} = 256$ 101

Figure 4.25: Maximum TA_{NN} of the flow rate classification with different n_{hidden} 102

Figure 4.26: The confusion matrix of the optimal NN model for flow rate detection 103

Figure 4.27: Flow rate accuracy under each rotation speed 104

Figure 4.28: The confusion matrix of rotation speed detection of combined NN model 106

Figure 4.29: The confusion matrix of flow rate detection of combined NN model 107

Figure 4.30: Classification of the non-normal and normal operating points with simplest NN model 109

Figure 4.31: Classification of the non-normal and normal operating points 110

Figure 4.32: Classification of the non-normal and normal operating points 111

Figure 4.33: Results and training curves of n_{hidden} =128 112

Figure 4.34: TA of the binary classification by training with different IMFs 113

Figure 4.35: Binary classification by training IMF 4 with simple NN models 114

Figure 4.36: Binary classification by training IMF 4 with complex NN models 114

Figure 4.37: Result of rotation speed detection utilizing waveform signals with same NN model as FFT signals.....116

Figure 4.38: Confusion matrix of flow rate detection utilizing waveform signals with same NN model as FFT signals 117

Figure 4.39: Result of rotation speed detection with maximum TA_{NN} 118

Figure 4.40: Result of rotation speed detection utilizing PtP and RMS values with same NN model as FFT signals 119

Figure 4.41: Confusion matrix of flow rate detection utilizing PtP and RMS values with same NN model as FFT signals 120

Figure 4.42: Result of rotation speed detection with maximum TA_{NN} 121

Figure 4.43： Confusion matrix of flow rate detection with maximum TA_{NN} 122

Figure 5.1: The t-SNE of different datasets for binary classification 125

Figure 5.2: The t-SNE of different datasets for rotation speed classification 126

Figure 5.3: The t-SNE of different datasets for flow rate classification ($\triangle Q = 2$ m^3/h FFT and $\triangle Q = 3$ m^3/h Wave Form).....127

List of Tables

Table 2.1: Open-source deep learning libraries [40] 48

Table 3.1: Electrical equipment list of the test bench 52

Table 3.2: The electrical components in the control box (Arduino) 56

Table 3.3: Specifications of ADXL345 [54] 62

Table 3.4: Pin mapping of the program 65

Table 3.5：Experiment settings of the test bench 68

Table 3.6：Flow rate ranges at each rotation speed 69

Table 3.7 Data settings 71

Table 3.8: Hyperparameter settings for experiment 73

Table 4.1: Nomenclature of the experiments and evaluations 75

Table 4.2: Dataset and NN settings for single measurement point investigation 80

Table 4.3: NN settings for single measurement point investigation 80

Table 4.4: Classes details for sensor positions determination 80

Table 4.5: Dataset settings for single measurement point investigation 83

Table 4.6: NN settings for single measurement point investigation 83

Table 4.7: Classes for minimum detectable flow rate difference 84

Table 4.8: Dataset settings for investigation the minimum size of the training dataset 86

Table 4.9: NN settings for investigation the minimum size of the training dataset 86

Table 4.10: Classes details for examination the minimum size of the total data 86

Table 4.11: Dataset settings for investigate the hidden layer effect 89

Table 4.12: NN settings for investigate the hidden layer effect 90

Table 4.13: Classes details of non-normal and normal 97

Table 4.14: Dataset settings for non-normal and normal classification 97

Table 4.15: NN settings for non-normal and normal classification 97

Table 4.16: Hyperparameters and training duration of each optimal NN model.....100

Table 4.17: Hyperparameters and training duration of each NN model 102

Table 4.18: Test accuracy and hyperparameters of each NN model.....104

Table 4.19: Hyperparameters and of the optimal NN model with two output layers ...105

Publications and Conferences

J. Feng and M. Böhle. Feasibility study for the application of a neural network for operating condition detection of a centrifugal pump. In 18th International Symposium on Transport Phenomena and Dynamics of Rotating Machinery, November 23-26, 2020. Online, Journal of Physics: Conference Series, vol. 1909, no. 012071, 2021.

T. Bian, X. Shen, **J. Feng**, and B. Wang. Numerical Investigation on the Secondary Flow Control by Using Splitters at Different Positions with Respect to the Main Blade, Fluid Dynamics & Materials Processing, vol.17, no.3, 014902, 2021.

J. Feng, T. Bian, Q. Han and B. Wang. A Numerical Investigation on influence of blade-ring- width of the fan on the characters of the Aerodynamic noise, Fluid Dynamics & Materials Processing, vol.19, no.1, 018275, 2023.

J. Feng, Z. Li, S. Zhang, C. Bao, J. Fang, Y. Yin, B. Chen, L. Pan, B. Wang and Y. Zheng. A Microimage-Processing-Based Technique for Detecting Qualitative and Quantitative Characteristics of Plant Cells, Agriculture, vol.13, no.9, 1816, 2023.

List of Supervised Student Thesis

P. Lehner	Entwicklung von Steuerungssoftware und Messprogrammen für einen Pumpenprüfstand zur Anwendung von neuronalen Netzen	PA	2019
W. Hong	Machbarkeitsstudie für die Anwendung eines Neuronalen Netzes zur Betriebspunkterkennung einer Kreiselpumpe	MA	2019
C. Xing	Anwendung des neuronalen Netzes zur Analyse von Druckverlusten in Rohrleitungen	MA	2019
L. Kunz	Regression neural networks for extended condition monitoring based on pumps vibration patterns	SA	2020
J. Tu	Entwicklung von Steuerungssoftware für einen Pumpenprüfstand basierend auf Raspberry Pi	BA	2020

Curriculum Vitae

Personal Data

Name: Jun Feng

E-mail: j.feng@mv.rptu.de

Education

04/2011	-	12/0215	Master in the University of Wuppertal
09/2006	-	07/2010	Bachelor in Wenhua College-Huazhong University of Science and Technology

Work Experience

10/2016	-	Present	Research Assistent Jianghan University-Technical University Kaiserslautern Fluid Mechanics and Fluid Machinery Research Center Jianghan University
10/2014	-	04/2015	Student Assistent Faculty of Theoretical Electrical Engineering The University of Wuppertal