6-18-2021

Efficient and Accurate Neural Network Based Internal Combustion Engine Modeling and Prediction

Weiying Zeng
University of Windsor

Efficient and Accurate Neural Network Based Internal Combustion Engine Modeling and Prediction

by

Weiying Zeng

A Dissertation
Submitted to the Faculty of Graduate Studies
through the Department of Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy at the
University of Windsor

Windsor, Ontario, Canada

2021

Efficient and Accurate Neural Network Based Internal Combustion Engine Modeling and Prediction

by

Weiying Zeng

APPROVED BY:

March 24, 2021

DECLARATION OF CO-AUTHORSHIP / PREVIOUS PUBLICATION

I. Co-Authorship

I hereby declare that this thesis incorporates material that is a result of joint research, as follows: Chapter 3 to Chapter 5 of the thesis was co-authored with Dr. Mohammed Khalid, Dr. Xiaoye Han, and Dr. Jimi Tjong. In all cases, the key ideas, primary contributions, experimental designs, data analysis, interpretation, and writing were performed by the author. Dr. Khalid contributed to the idea refinement, manuscript revision, and progress supervision. Dr. Han supported experimental design and setup. Dr. Tjong's contribution was primarily through the provision of supervision.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

II. Previous Publication

This thesis includes one original paper that has been previously published and one original paper that is going to be submitted for publication in peer reviewed journals, as follows:

Thesis chapter	Journal publication title/full citation	Publication status
Chapter [3] [4] [5]	Zeng, W., Khalid, M.A., Han, X. and Tjong, J., 2020, "A Study on Extreme Learning Machine for Gasoline Engine Torque Prediction", **IEEE Access**, 8, pp.104762-104774.	Published
Chapter [6] [7]	Zeng, W., Khalid, M.A., Han, X. and Tjong, J., "A Novel Progressive Extreme Learning Machine for System Identification", **Applied Intelligence, Springer**	To be submitted

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as a graduate student at the University of Windsor.

III. General

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis.

ABSTRACT

Traditionally the internal combustion engines and their subsystems are modeled purely based on their physical/mathematical principles. Such modeling techniques usually require deep prior knowledge of the internal combustion engine, which is often too difficult for many non-engine experts. In addition, the modeling process is usually very complicated and time-consuming. In some cases, the models may not be useful for many real-world applications due to oversimplified modeling assumptions. In recent years, with the rise of artificial intelligence technologies, the neural network based internal combustion engine modeling techniques have gained increasing popularity. In contrast to the traditional internal combustion engine modeling approaches, the neural network based methods can create the models directly from the system data instead of from the complicated physical/mathematical equations. This type of approach is easier to handle and often has fewer parameters to tune.

This dissertation presents an extreme learning machine based neural network modeling technique for gasoline engine torque prediction. The technique utilizes a single-hidden layer feedforward neural-network structure that has the potential to approximate any continuous function with high accuracy. To verify the robustness of this technique, over 3300 data points collected from a real-world gasoline engine were used to train and test the model. The data points spanned from 1000 rpm to 4500 rpm engine speed, idle to full engine load, which mirrored the full map of normal engine operating conditions.

The experimental results demonstrate that the created model predicts the gasoline engine torque with high accuracy. Furthermore, this research proposed a weight factor approach to further improve the model accuracy in the desired data regions without modifying the input data set. The model evaluation showed that the weight factor approach could reduce the overall prediction errors in the desired regions significantly. This feature is particularly useful in tuning the performance of the model when the significance of the individual data points varies, or when the distribution of the data points is imbalanced.

Moreover, an innovative form of extreme learning (referred to as progressive extreme learning machine) was proposed and evaluated. It was capable of gradually improving the estimation accuracy with recursions. The new algorithm maintained the random weights generation feature of the traditional extreme learning machine and upheld the training speed advantage over many other competing algorithms. The experimental evaluation results show that progressive extreme learning machine has higher accuracy and superior generalization than many other extreme learning machine based algorithms. Furthermore, its performance was also compared with some nonlinear machine learning algorithms using the publicly available data sets. The experimental evaluation results showed that the progressive learning machine outperformed the support vector regression and had comparable performance with Levenberg-Marquardt Algorithm.

DEDICATION

This thesis is dedicated to my wife Zhenyi, my daughter Lucy, my mother Miliang, my father Shuangjian, and my brother Yidi. Love you all.

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my advisor, Dr. Mohammed Khalid, for his faith and patience with me. His guidance, support, and encouragement are precious during my dissertation work. I am so grateful to be his student and have learned a lot under his supervision. I also like to thank my committee members, Dr. Jonathan Wu, Dr. Esam Abdel-Raheem, and Dr. Imran Ahmad for all the invaluable comments and suggestions on my research. I acknowledge Dr. Fayez Gebali from University of Victoria for his helpful suggestions and corrections on my dissertation. Special thanks to Dr. Xiaoye Han, Dr. Jimi Tjong, and Dr. Ming Zheng for all their supports, direct and indirect.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

Symbols

α	Lagrange multiplier
β	the weight vector connecting the hidden layer and output layer
$\hat{\beta}$	the estimation of the weight vector connecting the hidden layer and output layer
λ	the maximum number of recursions
ξ	the slack variable for Support Vector Regression
σ	the center spread parameter
η	the target accuracy
b	the bias of the hidden layer
c	neuron’s center
f	the activation function
g	the activation function of the hidden nodes
h	the output of a neuron
k	the regularization factor
m	the dimension of the input data
n	the number of training samples
w	the weight vector connecting the hidden layer and the input layer
$\widehat{w}$	the estimation of the weight vector connecting the hidden layer and the input layer
C	the box value for Support Vector Regression
H	the output of hidden layer

$H^{\dagger}$	the Moore-Penrose inverse of matrix H
$\widetilde{N}$	number of neurons
N	number of total data points
X	model input
Y	model output
$\hat{Y}$	the estimation of model output

Abbreviations

AE	Autoencoder
AG-ELM	Adaptive Growth of Hidden Nodes
AOI	Area of Interest
AFR	Air-to-Fuel Ratio
ANN	Artificial Neural Network
AIC	Akaike’s Information Criterion
B-ELM	Bidirectional Extreme Learning Machine
BP	Backpropagation
CFD	Computational Fluid Dynamics
CI-ELM	Convex Incremental Extreme Learning Machine
CNN	Convolutional Neural Network
D-ELM	Dynamic Extreme Learning Machine
DL	Deep Learning
E-ELM	Evolutionary Extreme Learning Machine
EB-ELM	Enhanced Bidirectional Extreme Learning Machine
ECU	Engine Control Unit
ECI-ELM	Enhanced Convex Incremental Extreme Learning Machine
EM-ELM	Error Minimized Extreme Learning Machine

EGR	Exhaust Gas Recirculation
EI-ELM	Enhanced Incremental Extreme Learning Machine
ELM	Extreme Learning Machine
ELM-AE	Extreme Learning Machine Auto-Encoder
ELU	Exponential Linear Unit
EM-ELM	Error Minimized Extreme Learning Machine
EMVT	Electromagnetic Valve Train
GELM-AE	Generalized Extreme Learning Machine Autoencoder
H-ELM	Hierarchical Extreme Learning Machine
HWFET	Highway Fuel Economy Test
IC	Internal Combustion
I-ELM	Incremental Extreme Learning Machine
KKT	Karush-Kuhn-Tucker
LLN	Law of Large Numbers
LMA	Levenberg-Marquardt Algorithm
LMSO	Low-to-Medium Speed Operating
MAF	Mass Air Flow
MLP	Multilayer Perceptron
ML-ELM	Multilayer Extreme Learning Machine

MSE	Mean Square Error
NM	Newton Meter
NNRW	Neural Network with Random Weights
NOx	Nitrogen Oxide
OEM	Original Equipment Manufacturer
OP-ELM	Optimally Pruned Extreme Learning Machine
OEB-ELM	Random Orthogonal Projection Based Enhanced Bidirectional Extreme Learning Machine
OS-ELM	Online Sequential Extreme Learning Machine
P-ELM	Pruned Extreme Learning Machine
PELM	Parallel Extreme Learning Machine
Pr-ELM	Progressive Extreme Learning Machine
PCM	Powertrain Control Module
RBF	Radial Basis Function
RBN	Radial Basis Network
ReLU	Rectified Linear Unit
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
RPM	Revolutions per Minute

RVFL	Random Vector Functional Link
SaE-ELM	Self-Adaptive Evolutionary Extreme Learning Machine
SLFN	Single-hidden Layer Feedforward Neural-Network
SSE	Sum of Squared Error
SVM	Support Vector Machine
SVR	Support Vector Regression
UDDS	Urban Dynamometer Driving Schedule
VCT	Variable Cam Timing

Chapter 1. Introduction

The modeling of the internal combustion (IC) engine and its subsystems is mainly realized through either the model-based or the data-driven approaches [1], [2]. The model-based approach often creates the engine models based on the physical laws of the engine system. Proper system abstractions and assumptions are critical for the accuracy of the model. Though such models have explicit structures, their expressions are usually mathematically complicated. A major disadvantage of this approach is that it requires the users to have a deep understanding of the engine system to build the models. On the other hand, the data-driven approach can create engine models based on the data directly collected from the target system. This method focuses on finding out the functional relations between the input and output data of the modeling object, rather than understanding the complex physics of the system. Among the data-driven automotive IC engine modeling approaches, the neural-network-based models have become increasingly popular in recent years.

1.1 Model-Based Internal Combustion Engine Modeling

The model-based approach is classical and probably the first choice for many people when it comes to IC engine modeling. Starting from the physics principles, with proper abstractions, the system is constructed with mathematical equations that directly describe the engine operational mechanism.

Various models were proposed to simulate the physical, electrical, and chemical processes of the in-cylinder and other parts of engine systems. For instance, Tolou *et al.* presented a semi-predictive model of turbocharged gasoline direct injection engine [3]. The model approximated the combustion heat release with a double-Wiebe function and predicted the cylinder peak pressure by tuning the Wiebe variables. Togun *et al.* proposed a nonlinear mathematical approach to model the engine torque using a recursive least square method, in which the nonlinearity was determined with Hammerstein structure, and the system was identified by studying multi-order linear dynamics [4]. Tan and Reitz presented a spark model with an equation to calculate the propagation rate of the ignition flame [5]. The formation and properties of the flame kernel were also modeled in [6] by Boudier *et al.* Their model results matched a set of typical engine experiment data without parameter adjustments. To investigate the cycle-to-cycle performance variance of gasoline engines, Daw *et al.* proposed a model that showed the autocorrelation between the stochastic fluctuation of the engine parameters and the nonlinear deterministic coupling between successive engine cycles [7]. In addition, some computational simulation approaches were also adopted to model the engine operation. For instance, computational fluid dynamics (CFD) and MATLAB/Simulink modeling technologies were demonstrated [8]–[11].

However, to use the model-based method properly, a comprehensive understanding of the internal combustion engine or its subsystems is required, which limits non-experts' accessibility to such approaches. Moreover, as pointed out in [12], the model-based modeling strategies often fail to work effectively in practice due to the complexity of multivariate coupling and unexpected noise in real-world applications.

1.2 Neural Network Based Automotive IC Engine Modeling

In contrast to the above-mentioned modeling approaches, the data-driven modeling techniques can work well even without a deep understanding of the fundamental physics of the target system. Generally, these techniques can create models directly from the data collected on the target system instead of from the complicated governing equations derived from physical principles. Moreover, the data-driven modeling techniques could function well when multivariate correlation exists. Among many data-driven techniques for the engine-related applications, artificial neural network (ANN) is very popular and has been increasingly studied.

Wu *et al.* proposed a mass air flow (MAF) sensor model on a dual-cam engine using a two hidden layer ANN structure [13]. The simulated results showed good agreement between the dynamometer data and the vehicle test data. Togun and Baysec demonstrated the capability of predicting engine torque and brake fuel specific consumption using neural networks [14]. Cay investigated the applicability of ANN to predict engine performance and exhaust temperature values [15]. Cycle-to-cycle combustion variation was studied by Di Mauro *et al.* in [16], which helped identify the pre-ignition and pre-combustion factors, as well as predict the variation of the indicated mean effective pressure. A neural network based nonlinear predictive control scheme was investigated by Hu *et al.*, in which the coordinated control of throttle and wastegate on a turbocharged gasoline engine was explored [17]. Li *et al.* presented a model for nitrogen oxide and smoke emissions on a diesel engine [18], in which the authors not only discussed how to select the model inputs based on the physical analysis but also showed the insightful comparisons of a range of neural network architectures in terms of model complexity and accuracy. In addition to characterizing engine performance and predicting engine behaviors, neural network models were also used in engine fault diagnostics. Zheng *et al.* revealed how misfire could be detected and categorized under different engine conditions using Elman neural network [19]. Bearing knock fault features were detected by proper ANNs as demonstrated by Chen and Randall [20]. Vibration data were investigated by Ahmed *et al.* with an ANN model that could identify various engine faults and categorize the severity of the faults [21]. Wen *et al.* proposed a bearing fault detection model using a convolutional neural network (CNN) that could autonomously learn the unique features of each type of bearing fault and identify the defective bearings with its fault type [22].

However, the neural network approaches in the aforementioned research papers still need further improvement. Firstly, the neural networks all use a backpropagation algorithm to train the weight of each neuron. In practice, this algorithm may not be easy enough to implement, especially for non-experts. The disadvantages of backpropagation include difficulties in determining the proper learning rate, getting trapped in local minima, prone to over-training, and very time-consuming for most of the applications [23]. Secondly, these neural network models are typically created and tested with about 80 to 130 data points. More data points are required to explore the robustness of the neural network. Thirdly, all the data points are treated equally in the neural network optimization process. However, the weight of each data point needs to be considered differently to meet the requirements of practical engineering considerations. For instance, the gasoline engines usually operate at certain conditions much more often than other conditions [24]. Therefore, a neural network model should have higher accuracy at the more frequently operated engine conditions, even though the accuracy at the less frequently operated engine conditions may have to be sacrificed slightly.

1.3 Thesis Goals

In this dissertation, we explore a novel neural network based approach for automotive IC engine torque modeling. The proposed approach is easy to implement, so it can be quickly adopted by the users, with and without the background knowledge in IC engines. In addition, the proposed approach delivers sufficient accuracy for the intended IC engine applications. It also provides comparable performance with other competing algorithms for the data sets used in different applications other than IC engine torque modeling.

The goals of this research are as follows:

1) Explore a single-hidden layer feedforward neural-network (SLFN) approach for IC engine torque modeling. Propose an improved extreme learning machine (ELM) based methodology for IC engine torque modeling.
2) Conduct comprehensive IC engine mapping tests to collect a large amount of engine mapping data that can cover the normal operating conditions of a real-world IC engine. Over 3300 points are collected as the evaluation data set compared to typically 80 to 130 points employed by other IC engine modeling algorithms.
3) Evaluate the performance of the proposed IC engine torque model with the experimentally collected data set.
4) Investigate the effectiveness of the proposed algorithm. Compare its performance with other popular ELM-based algorithms and the non-ELM algorithms.

1.4 Scope of Work

This section describes the key compositions of the dissertation and identifies the boundary of the work. Necessary background information was covered before introducing each algorithm. Practicality was considered throughout the research. The experiment design and the evaluation criteria of the newly proposed algorithm were application-oriented to assure that this research can help solve real-world engineering problems.

Comprehensive engine mapping experiments were carried out to provide a primary data source to evaluate the proposed algorithms. Over 3300 data points were collected for this research compared with about 80~130 data points used in the research described in Section 1.2. The experiments were conducted at an industry-leading testing facility and in line with the standards of the leading automotive Original Equipment Manufacturers (OEMs). The whole engine map for normal engine operating conditions was covered in the test. With a large number of high-quality data, not only the robustness of the proposed algorithm was tested, but the applicability of the created model was also guaranteed.

Other than many of the aforementioned research papers, in which all the data points were treated uniformly to achieve an overall performance level of the neural network, a weighted optimization was applied to treat the data points with different significance, so the engine model could achieve higher accuracy at certain dedicated data areas. This feature is important for the real-world applications as IC engines usually operate at certain operating conditions at a higher frequency. These operating conditions require higher modeling accuracy.

Further explorations were conducted to seek improvement of the existing ELM algorithm. With this objective, a new ELM-based algorithm, named progressive extreme learning machine (Pr-ELM), was proposed and evaluated against the competing ELM-based algorithms. Regression accuracy and the linear relationship were used to assess the performance of the algorithm.

1.5 Thesis Contributions

This dissertation presents an approach to create a single-hidden layer feedforward neural network based regression model using ELM, which is used for predicting the output torque of a gasoline engine. In addition, an improved ELM-based algorithm is proposed to enhance the accuracy of the existing ELM algorithms. The contributions of this thesis are as follows.

1) Established a procedure to create an engine torque prediction model using ELM approach so that a non-engine expert could also benefit from it.
2) Proposed a weight factor approach to further enhance the model accuracy in the desired data regions based on real-world applications.
3) Proposed an improved ELM algorithm that could increase the accuracy of the traditional ELM significantly.

1.6 Thesis Organization

The remainder of this thesis is organized as follows.

In Chapter 2, various neural network algorithms for SLFN are reviewed. The advantages and disadvantages of each algorithm are highlighted. The ELM based engine modeling algorithm is presented in Chapter 3. Chapter 4 describes the experimental setup and the data acquisition approaches for evaluating the model. The engine test data presented in this chapter are used to verify the proposed IC engine torque model. Chapter 5 presents the experimental evaluation results of the engine torque model with the acquired data. A new ELM based algorithm called Progressive ELM is presented in Chapter 6. Chapter 7 briefly covers the comparison studies between Pr-ELM and other ELM-based algorithms, as well the popular non ELM-based training algorithms. Finally, concluding remarks and future research perspectives are provided in Chapter 8. The organization chart of this thesis is shown in Figure 1-1.

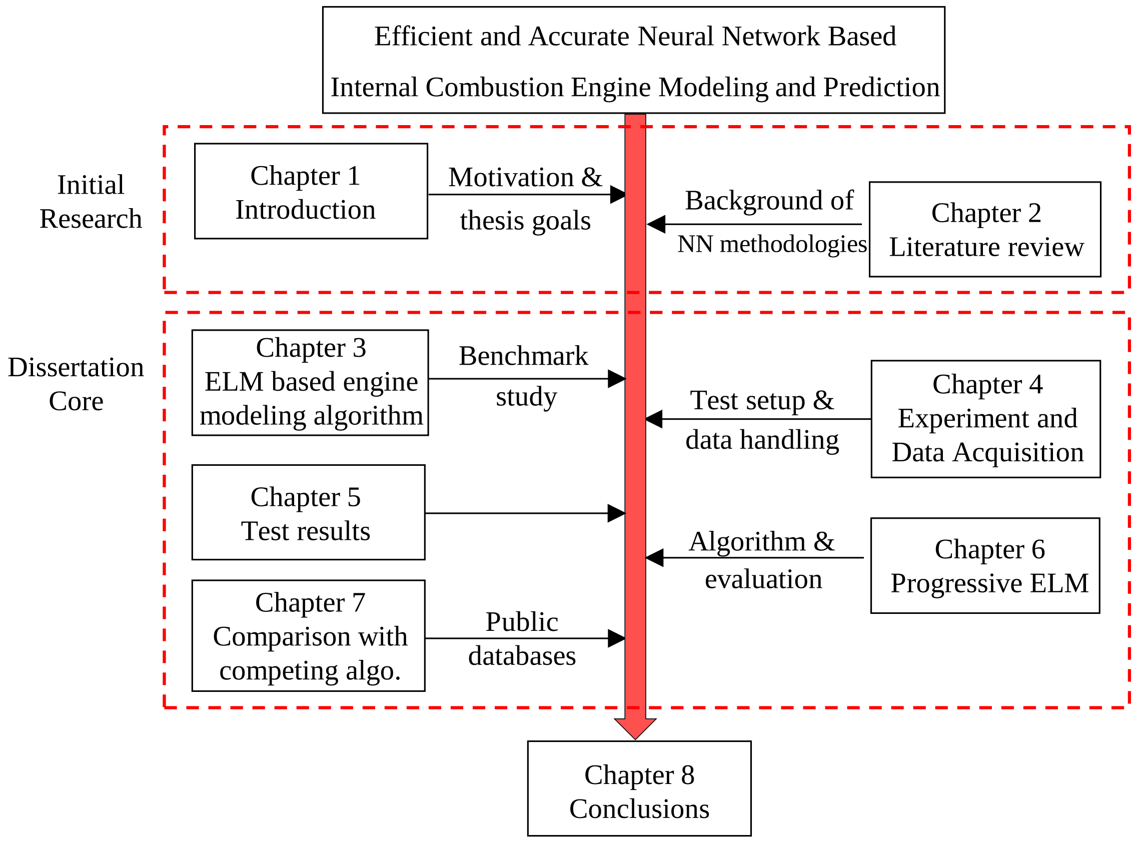

Efficient and Accurate Neural Network Based
Internal Combustion Engine Modeling and Prediction
Initial Research
Chapter 1 Introduction
Motivation & thesis goals
Background of NN methodologies
Chapter 2 Literature review
Dissertation Core
Chapter 3 ELM based engine modeling algorithm
Benchmark study
Test setup & data handling
Chapter 4 Experiment and Data Acquisition
Chapter 5 Test results
Algorithm & evaluation
Chapter 6 Progressive ELM
Chapter 7 Comparison with competing algo.
Public databases
Chapter 8 Conclusions

Figure 1-1 Thesis organization

Chapter 2. Background and Related Work

Neural network based system modeling and identification techniques have provided new perspectives on understanding the behavior of complex systems. Based on the anatomy of the human nervous system and the mechanism of its operation, artificial neural networks are created in an attempt to achieve similar functionalities that a human neural network can do, such as learning adaptation, generalization, massive parallelism, robustness, associative storage of information, and spatiotemporal information processing [25]. Without knowing much physical insight into the target system, neural network based system modeling and identification techniques can infer the relationships of the relevant system parameters from the data that are usually acquired from the target system [26]. Like many system identification models [4], [13], [14], [27], [28], two main issues that need to be solved are the selection of proper neural network architecture (such as model parameters, activation functions, and connection types) and the choice of model training algorithms (such as backpropagation and support vector machine) [29].

In this chapter, the basic anatomy of neuron networks is introduced. Then the popular neural network architectures are presented, followed by the introduction of the mainstream network training algorithms for system identification.

2.1 The Basics of Single Layer Perceptron

The idea behind the artificial neural network is to mimic the anatomy of the human brain neural network and simulate the mechanism of its operation.

The human brain is a vast and extremely sophisticated neural network. An exemplary biological neuron network with only one neuron cell is demonstrated in Figure 2-1. Though the biological sensory organs may be of various types, the input signals are all picked up by the cell dendrites, through which the external input enters a neural system. Then the electrochemical nerve impulse (also called stimulus) is generated in the dendrites and transmitted through axons to the axon terminals (in a unilateral direction). The axon terminals connect to the next one or multiple neurons' dendrites in the synapse and exchange nerve information in it. In a human brain, there are approximately 100 billion neurons [30]. As more neurons joining in the process, the more complicated and abstracted information the neural network can handle.

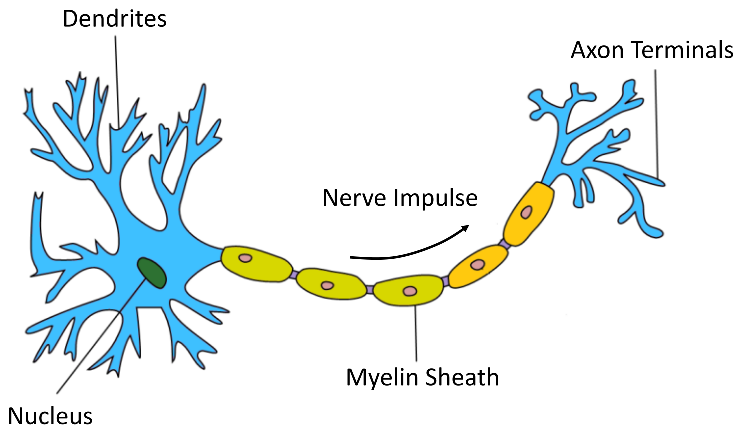


Figure 2-1 Structure of a biological neuron (adapted from Q. Jaroz [31])

In the same analogy, an artificial neuron node can be modeled mathematically as shown in Figure 2-2. The cell receives and transmits information through the input and the output ports, which simulate the dendrites and axon terminals of the biologic neural cell, respectively. For illustration simplicity, only one input port and one output port are presented in the figure. A bias term is also fed into the neuron node to analogize the stochastic noises and all other unknown characteristics that may have contributed to the overall input of the neuron cell. The cell body contains a function f, called the activation function, to imitate the decision-making process of the neuron cell. The output is the outcome of the processed information which is passed to the next connected neuron(s).

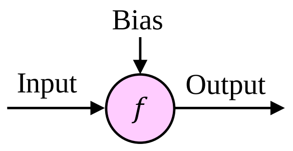


Figure 2-2 Mathematical abstraction of a single neuron

A single layer perceptron can be created accordingly by putting more neurons together. An exemplar neural network is the integrate-and-fire model proposed by McCulloch and Pitts [32]. As illustrated in Figure 2-3, each pink circle represents an input neuron cell, and the blue circle represents an output neuron cell (biases are omitted for simplicity). The pink circles analogize to the information sensory units such as dendrites, the link between the pink and the blue circles represent the information transmitting unit such as the axons, and the blue circle represents the axon terminals. The weight at each path indicates the significance the cell gives to that input path. In this example, the output is a weighted sum of the inputs.

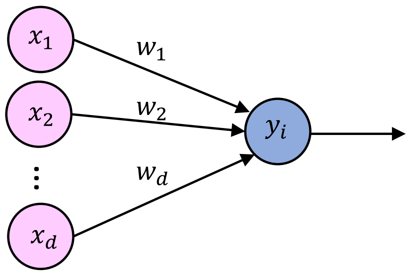


Figure 2-3 Structure of a single-layer artificial neural network

The mathematical equation of this model is shown in (2-1).

$$\hat{y}_i = w_1x_1 + w_2x_2 + \cdots + w_dx_d \tag{2-1}$$

2.2 Popular Activation Functions

There are various types of activation functions, such as sigmoid, arctangent, rectified linear unit, and many other mathematical functions. According to the research by Huang *et al.* [33], the specific function of the biological neural cells is unknown. A rectified linear unit function may be a close form in the sense that it requires a threshold before anything happens, which matches the firing concept of neural activation. In this section, a few popular activation functions are briefly described.

- Binary step function

The binary step function is one of the simplest activation functions. It can be used to represent a trigger threshold that determines whether a neuron should fire or not. As plotted in Figure 2-4, the binary step function works much like a transistor: if the input is lower than a certain level, the neuron remains inactive; if the input is greater than a certain level, the neuron is activated. However, the output is invariable with a constant amplitude. The mathematical expression of the binary step function is in (2-2).

$$y = \begin{cases} 0 & \text{if} \quad x < 0 \\ 1 & \text{if} \quad x \geq 0 \end{cases} \tag{2-2}$$

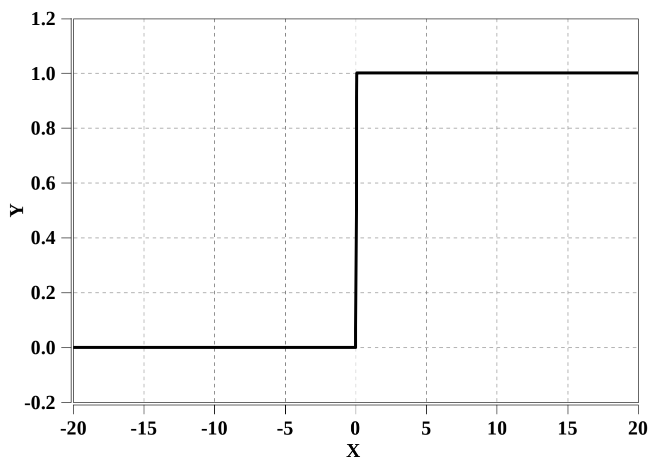


Figure 2-4 Binary step activation function

- Sigmoid function

The sigmoid function is widely used as the activation function in various neural network applications. Its mathematical expression is shown in (2-3). The sigmoidal function is nonlinear and monotonic. It has a bounded output range between 0 and 1 as plotted in Figure 2-5, which makes it a good candidate for probability estimation. In addition, its first-order derivative has a bell-shaped curve, just like a normal distribution curve. Such a property renders it many statistical advantages over other activation functions.

$$y = 1/(1 + e^{-x}) \tag{2-3}$$

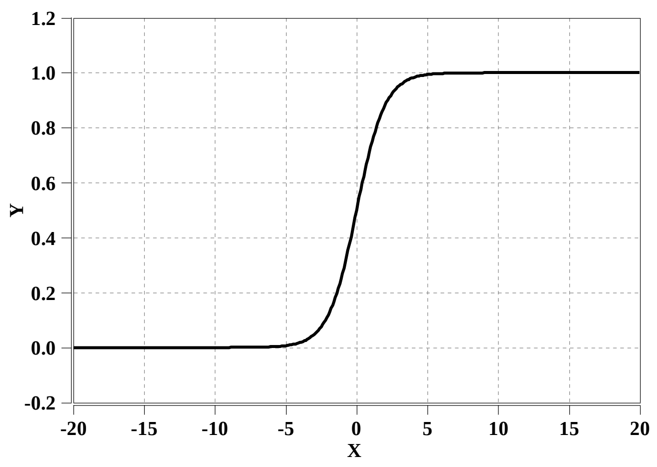


Figure 2-5 Sigmoid activation function

- Arctangent function

The arctangent activation function maps the input to the range (-π/2, π/2) as showing in Figure 2-6. It characterizes a slow rise when x is far from 0 and transits to a sharp growth phase when x approaches to 0. Arctan is monotonic and has symmetric responses across all x ranges. Its mathematical expression is shown in (2-4).

$$y = actan(x) \tag{2-4}$$

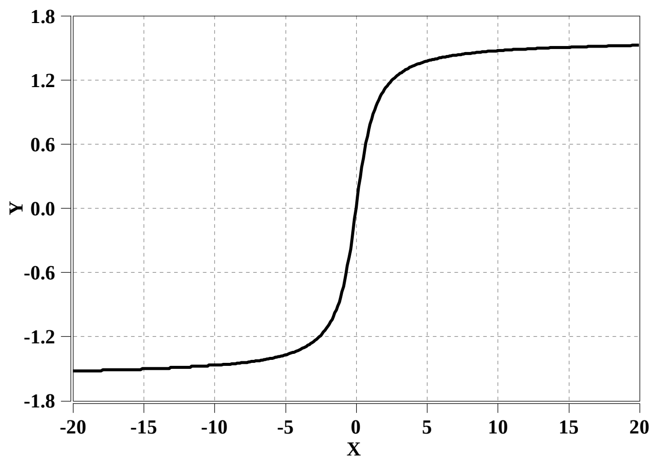


Figure 2-6 Arctangent activation function

- Rectified linear unit function

Rectified linear unit (ReLU) function is a popular non-linear activation function, especially for deep learning applications. As can be seen from (2-5) and Figure 2-7, the neuron fires only to a certain range of the inputs: if the input is negative, its output remains 0; if the input is non-negative, the output is the same value as the input. With such a property, it works perfectly as a “mask” in a neural network to shut down the irrelevant paths without physically changing the network architecture.

$$y = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{if } x \geq 0 \end{cases} \tag{2-5}$$

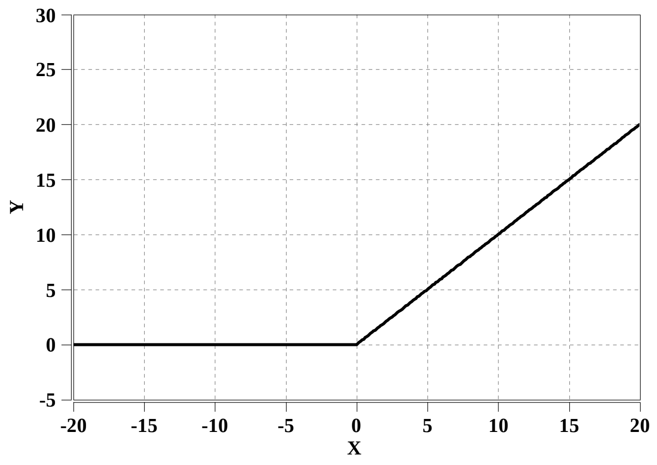


Figure 2-7 ReLU activation function

- Exponential linear unit function

The exponential linear unit (ELU) is a variant of ReLU. When the input is non-negative. ELU has the same as ReLU. However, when the input is negative, ELU smooths slowly until the output approaches $-a$, where it starts to smooth sharply. The equation and plot of

ELU can be found in (2-6) and Figure 2-8, respectively. Unlike ReLU function, ELU can produce negative output and it is often considered as an alternative to ReLU.

$$y = \begin{cases} x & x \geq 0 \\ a(e^x - 1) & x < 0 \end{cases} \tag{2-6}$$

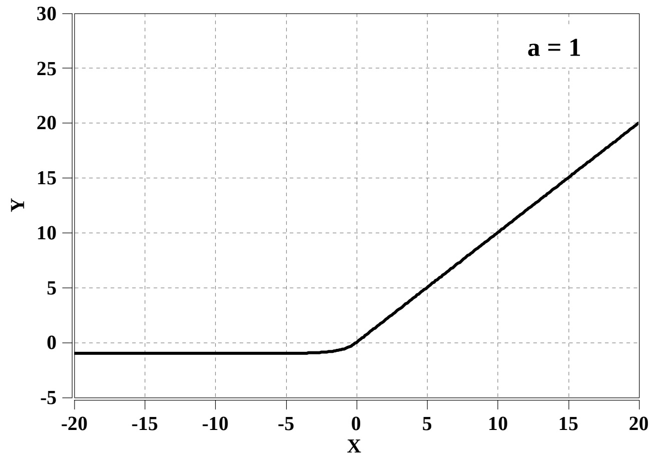


Figure 2-8 Exponential linear unit function

- Hyperbolic tangent function

The hyperbolic tangent function (tanh) maps the input to the range (-1, 1). It is essentially a shifted version of sigmoid function and centers at 0. Therefore, in practice, its nonlinearity is always preferred to the sigmoid nonlinearity [34]. In addition, it also helps in centering the data by bringing the mean value close to 0. The equation and the plot of tanh function are shown in (2-7) and Figure 2-9, respectively.

$$y = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2-7}$$

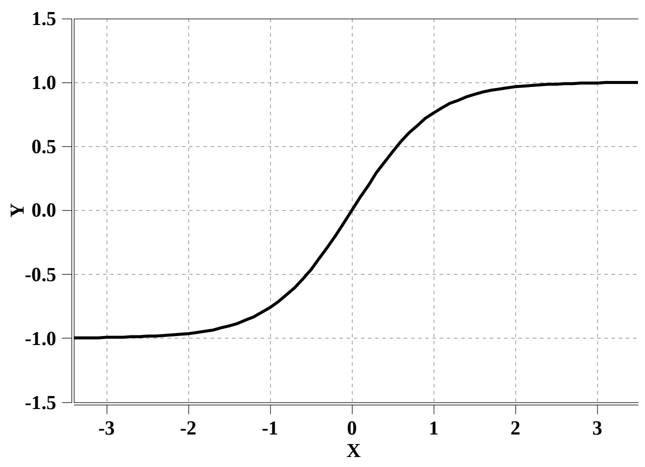


Figure 2-9 Hyperbolic tangent function

From the exemplary functions introduced above, it seems that only the nonlinear functions can be used as neural network activation functions. In fact, the linear functions, such as $y = ax + b$, may also be used as activation functions. However, there are two significant disadvantages.

- As the derivatives of linear function have no relation to the input x, it is impossible to use the popular backpropagation algorithm to optimize the network parameters (because the gradient is the same all over the place regardless of the training data).
- Since the activations are linear, a multi-layer network is virtually the same as a single-layer layer network as all the linear activation functions can be squashed into one single linear function. Because of such attributes, it has limited power to extract deeper features from the data source.

Therefore, the non-linear activation functions are preferred and widely accepted as the activation functions in modern neural networks. However, because the true form of activation function in a biological neuron cell is still unknown, it is impossible to justify which non-linear activation function has the closest form to the true biological activation function of a neuron cell [33]. In general, the choice of proper activation functions depends on the particular network architecture, the available data set, and the network training strategy. In addition, the selected activation functions need to be computationally efficient and easy to handle, as there could easily be hundreds to millions of neurons in a modern network. These neurons have to undergo derivative operations or other complex mathematical operations extensively. Based on empirical observations, the sigmoid function appears to be the most widely used one due to its computational efficiency and mathematical advantages.

2.3 Main Types of Neural Networks

The types of neural networks have grown exponentially since the very first one proposed by McCulloch and Pitts in 1943 [32]. Though different types of neural networks all work essentially in the same principle (resembling the perception of the human nerve system), they come with different capabilities in modeling different systems. Hidden neural nodes and various network structures are used to build different types of neural networks. The hidden nodes are specific functional nodes designed to extract particular features from a given data set and deliver the output in a specific form. The connections between the nodes represent the data flow directions in the network and help form the network architecture. With the hidden neurons and various architectures, a linear regression model can have the potential to describe the complex relationship between input and output data.

- **Feedforward Neural Network**

Feedforward neural network has one of the simplest network architectures. The fundamental characteristic of a feedforward neural network is that its data path is unidirectional, from the input node to the output node. The network can be either single layer (as shown in Figure 2-3), single-hidden layer, or multiple hidden layers. The multiple hidden layer feedforward neural network is also called deep feedforward neural network. The output of the network is the sum of the products of the output of the last hidden layer and their respective weights. An exemplar deep feedforward neural network is shown in Figure 2-10. Though feedforward neural networks have been invented for decades, they only get revived after the invention of the backpropagation algorithm and the development of powerful computers. In practice, feedforward neural network is one of the most widely used network types and most of them are trained using backpropagation based algorithms. Its typical applications include system identification, pattern recognition, and computer vision.

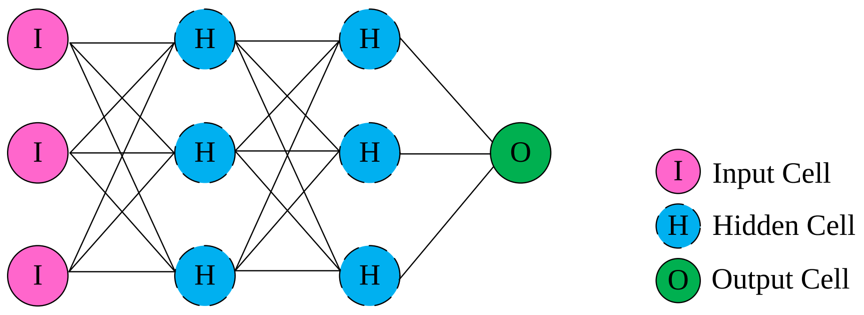


Figure 2-10 Deep feedforward neural network

- **Radial Basis Network (RBN)**

Radial basis network is one of the widely used neural networks for approximation applications, particularly for scattered data interpolation. Unlike other neural networks, RBN uses radial basis function as the activation function. Its cost function is the distance from the data points to the centers [35], [36]. A schematic diagram of a radial basis network is presented in Figure 2-11. The typical applications of radial basis networks are approximation and classification. However, as indicated by Simonenko *et al.* [37], the choice of radial basis network parameter is application dependent, which may pose difficulties in exploiting its potential effectively.

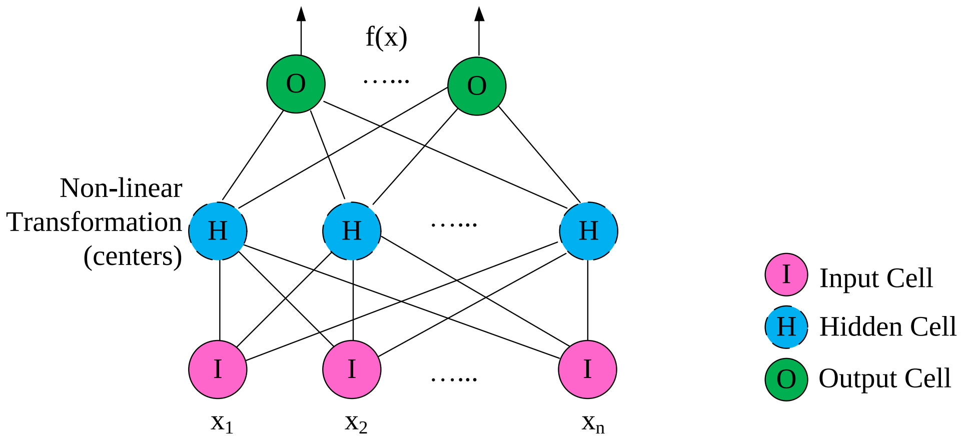


Figure 2-11 Schematic diagram of Radial Basis Network (adapted from [36])

- **Autoencoder (AE)**

Autoencoder is a special network whose output is expected to replicate its input with the least amount of distortion. It plays a fundamental role in unsupervised learning and deep architectures [38]. Hinton *et al.* has proposed an effective way of initializing the weights to allow deep autoencoder networks to reduce the dimensionality of the data without training pattern deformations in [39]. Due to the simple implementation and relatively lower computational cost, autoencoders have been widely used in many applications, such as natural language processing, object detection, biometric recognition, and data analysis [40] [41]. The schematic diagram of a simple autoencoder network is presented in Figure 2-12.

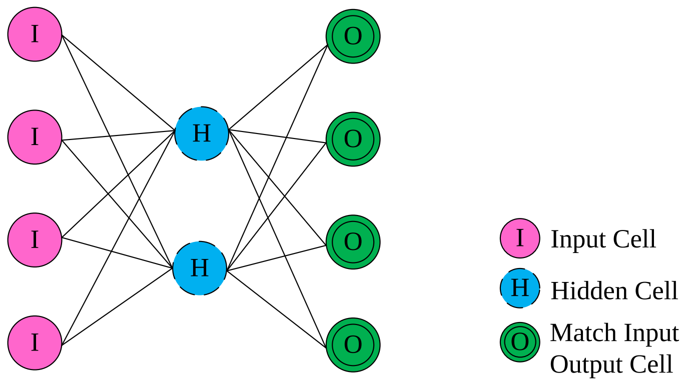


Figure 2-12 Schematic of Autoencoder

- **Recurrent Neural Network (RNN)**

Unlike the feedforward neural networks, RNN contains recurrent cell which receives its own output, or it allows neuron connections from a neuron in one layer to its neurons in its previous layers. As a result, sequential information can be captured. The dynamic temporal behaviors can also be exploited. Based on different activation functions and connection modes, there are various types of RNNs, such as stochastic neural network, bidirectional network, fully recurrent neural network, simple recurrent neural network, neural Turing machines, long short-term memories, and gated recurrent units [42]. Overall, backpropagation through time algorithm is the most widely used algorithm to train RNNs. Compared with the feedforward neural networks, RNNs usually require longer training time as the input value of a neuron may depend on the outputs of a series of neurons in the downstream layers. In practice, RNN is commonly used in speech recognition, queuing theory, mobility pattern prediction, and statistics.

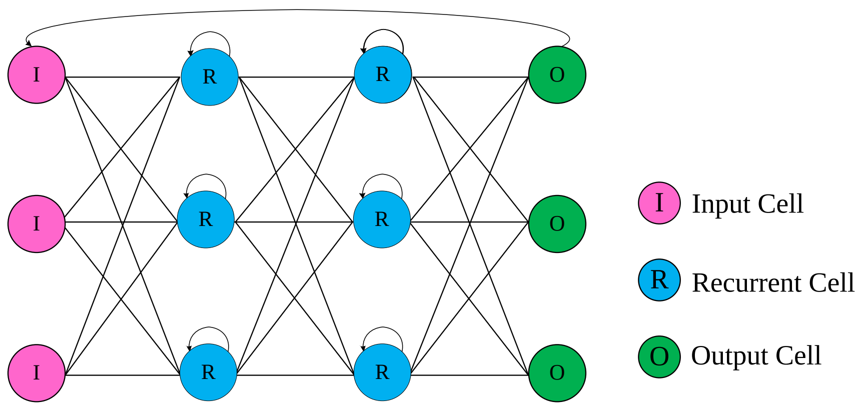


Figure 2-13 Schematic diagram of Recurrent Neural Network

- **Deep Convolutional Network**

Convolutional neural networks are primarily used for image processing but can also be used for other types of input such as audio. A typical use case of CNNs is that the user feeds the network with images and the network classifies the data. As shown in Figure 2-14, the input data are normally convolved with several different kernels to extract different features. They are usually scanned in a raster sequence rather than parsed all at once. The input data are then fed through convolutional layers instead of normal layers. These convolutional layers also tend to shrink as they become deeper, mostly by easily divisible factors of the input. Besides these convolutional layers, they also often feature pooling layers. Pooling is a way to filter out details: a commonly found pooling technique is max pooling. Deep convolutional networks enable the unsupervised construction of hierarchical image representations.

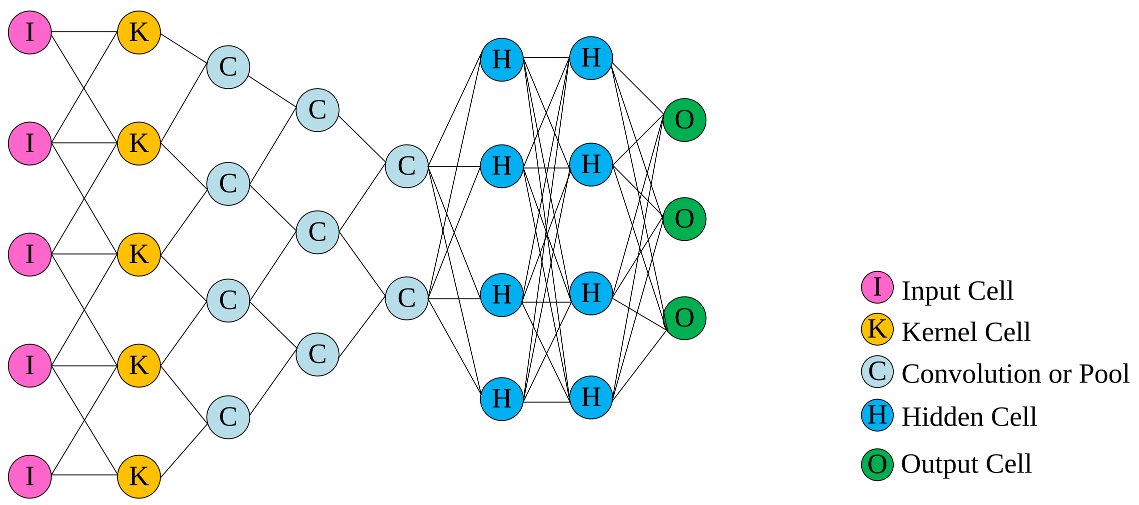


Figure 2-14 Schematic diagram of Deep Convolutional Network

2.4 Neural Network Training Algorithms

Neural network training involves finding the appropriate weights for the neural nodes in the network to approximate the target function by mapping the input to the output with minimum estimation losses. Each neural network training algorithm may have its unique advantages and limitations. Given the same neural network structure, different training approaches may have significant differences in network performance. Therefore, it is vital to select proper algorithms for specific applications. In this section, the popular neural network training algorithms for system identification are presented.

2.4.1 Backpropagation

Backpropagation is one of the most widely used learning algorithms for ANNs [43]. Its applications can be found in various fields, such as system modeling, pattern recognition, sensitivity analysis, and the control of systems over time, among others. Some even think backpropagation is one of the key factors that have brought wide popularity of the powerful multilayer perceptron (MLP) network and created a trend of machine learning over the statistical models [25].

In the standard backpropagation algorithm, the sum of squared error (SSE) is the commonly used cost function[1], which is repetitively derived, with respect to the weights and biases, to gradually approach the proper weights and biases that lead to the minimized error function for a given training data set. Considering an SLFN, the input to the hidden layer is

$$net_i = \sum_{j=1}^{m} w_{(i,j)} x_j + b_i \tag{2-8}$$

The output of the network is

$$\hat{y}_i = \beta^T \cdot f(net_i) \tag{2-9}$$

The sum of squared error of the network is

$$E = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2-10}$$

---

[1] Each cost function may have its unique advantages of being used for a certain application. For instance, Manhattan distance may be a preferred cost function for high data dimensionality. However, the selection of cost functions is not in the scope of this research.

where $x_i = (x_{i1}, x_{i2}, \dots, x_{im}) \in R^m$, m is the dimension of x, β is weight connecting the hidden layer and output layer. f is the activation function, and n is the number of training samples.

In the traditional backpropagation algorithm, the weights and biases are initialized arbitrarily (usually between -0.1 to 0.1), although it is better to initialize them based on the prior information if it is available. The error function is derived repeatedly and backpropagated to find out the gradient with respect to every β_i, w_{ij}, b_i at each data point. The weights and biases are updated along the steepest gradient as showing in (2-11) to (2-13) until the stop condition is met.

$$w^{k+1}(i,j) = w^k(i,j) - \eta \frac{\partial f(E)}{\partial w^k(i,j)} \tag{2-11}$$

$$b^k(i) = b^k(i) - \eta \frac{\partial f(E)}{\partial b^k(i)} \tag{2-12}$$

$$\beta^k(i) = \beta^k(i) - \eta \frac{\partial f(E)}{\partial \beta^k(i)} \tag{2-13}$$

where η is the learning rate.

The learning rate of traditional backpropagation is a relatively small number. Even 0.01 is common in many cases. If the learning rate is too small, the network may converge very slowly. Whereas if the learning rate is too big, the network may oscillate around the optimal value and never approach the optimal value close enough. To increase the learning speed while still remaining good accuracy, some improved backpropagation-based algorithms have been proposed. For instance, Levenberg-Marquardt (LM) algorithm has incorporated Gauss-Newton algorithm with the gradient descent approach, so it can converge fast when the estimated value is far from the expected value, and use a relatively smaller learning rate when the estimated value is close to the expected value.

Many backpropagation algorithms and their improved variants were used in IC engine related applications over the years. Wang *et al*. [44] proposed an approach to estimate nitrogen oxide (NOx) emissions using mutual information and the backpropagation neural network. It was verified that the proposed neural network could reduce absolute deviation and root mean square error by about 15% compared with the static map approach. Najafi *et al*. [45] used the standard backpropagation to create a neural network to analyze the performance and pollutant emissions of a spark-ignition engine operating on ethanol-gasoline blends. It was observed that the ANN model could achieve as high as 0.97 to 1 correlation coefficient and 0.46% to 5.57% mean relative error. A similar study was carried out in [27] to reveal the ANN's potential to offer fast, accurate, and reliable means in prediction or approximation affairs. A scaled conjugate gradient algorithm, a variant of backpropagation, was proposed by Cay [15] to train an SLFN to predict specific fuel consumption, engine power, and exhaust temperature of a gasoline engine. The model was able to predict the engine performance with the R^2 values about 0.99 between the training and test data sets. To improve early diagnosis of the fault on electromagnetic valve train (EMVT), backpropagation neural network was adapted with grey relation analysis to learn the fault pattern and help identify early symptoms of EMVT failure [46].

2.4.2 Neural Network with Random Weights

From a parameter estimation perspective, multilayer perceptron networks tend to have a large number of free parameters. However, a system with a large number of free parameters often requires a large number of data samples to train. The training process often takes a much longer time than the simpler networks. Schmidt *et al*. [47] argued that the weights of the neural network might not be important and do not need to be tuned to very high accuracy. Further research has lead to neural network with random weights (NNRW). It provides a non-iterative approach to solve the ANN training problems that are solved by the traditional BP-based learning approaches. Compared with traditional learning with global tuning such as deep learning with the BP-based method, NNRW can achieve a much faster training speed with acceptable accuracy. In addition, NNRW is easy to implement and its universal approximation capability has been proven in theory [48], [49].

In recent years, NNRW has attracted wide attention and has gone through many interesting developments. A 2D-NNRW algorithm was proposed for facial recognition [50]. It employed left and right projecting vectors to reduce the high dimensional input weight in the hidden layer while still preserved the image matrix structure. It showed improved recognition performance compared to the traditional NNRW approach on the popular datasets. Ramanujan *et al.* [51] dug deep into the NNRW structure and found out that an NNRW contained a subnetwork that had an impressive performance on a given task without modifying the weights. Moreover, they proposed an approach to sort out the impressive subnetwork from the large NNRW. Ramanujan *et al.*'s work was helpful in understanding the optimization and initialization of neural networks. The ranks of the input data and the quality of the random feature mapping in the NNRW were studied in [52]. It revealed that there was a certain threshold of the rank of the input data beyond which the performance of the NNRW increased with the increase of the rank. Moreover, the revealed relationship was independent of the number of hidden neural nodes and the activation functions. In order to evaluate the quality of random feature mapping in the NNRW, a dispersion degree of the matrix information distribution was proposed in this study. It could predict the quality of model initialization prior to model training and greatly improve the efficiency of modeling.

2.4.3 Radial Basis Function Neural Network

The basic RBF neural network is an MLP with three layers: the input layer, the hidden neuron layer, and the output layer. The nodes in each layer are fully connected to the previous layer. It calculates the distance between each input point and its centroid. The activation function is some nonlinear function that operates on the distance. A potential advantage of RBF networks is that it can augment the new training data without retraining the network. The structure of an RBF network is shown in Figure 2-11 and its Gauss function adapted expression is shown in (2-14).

$$h_j = \exp\left(-\frac{\left\|x - c_j\right\|^2}{\sigma_j^2}\right) \tag{2-14}$$

where h_j is the output of the j_{th} neuron, x is the input vector, c_j is the neuron's center and σ_j is the center spread parameter.

Generally, the RBF centers can be selected from the training data sets, determined through clustering analysis [53]. Based on its universal approximation capability, various applications of RBF network have been proposed by researchers. Some of the IC engine related applications are presented as follows.

An RBF neural network trained with recursive least squares method was proposed to simulate the IC engine parameters, such as crankshaft speed, intake manifold pressure, and intake manifold temperature in [54]. Based on the RBF neural network, a model-based predictive control strategy was developed and verified to achieve improved crankshaft speed control results. Wang *et al*. [55] introduced an RBF neural network based dynamic NOx prediction model, which used cylinder pressure as the feedback variable to indicate NOx levels. The experiment demonstrated that cylinder pressures could be properly predicted with acceptable accuracy. As a result, NOx levels were correctly estimated to meet the requirement of diesel engines. A novel two-RBF-network based adaptive inverse model control system was proposed to deal with the strong nonlinear effects in electronic throttle body modeling [35]. The first RBF network was used to identify the sensitivity information of the plant. The second RBF network was utilized as the inverse model controller. The proposed model could achieve successful throttle setpoint tracking within ±0.2° static error even some parameter variations were presented. Bizon *et al*. [56] proposed to use an RBF network to learn the relationship between cylinder pressure and engine block vibration. Then the engine vibration data were used to predict the cylinder pressure. The experiment showed that the relative error of the predicted peak cylinder pressure was less than 4%.

2.5 Summary

This chapter briefly described the background knowledge of neural networks. The network composition, the frequently used activation functions, and a few commonly adopted training algorithms for system identification problems were covered. The neural network based approaches for IC engine related modeling were also reviewed.

Chapter 3. Neural Network Algorithm for IC Engine Torque Modeling

In this research, we explored SLFN based approaches to model the output torque of an IC gasoline engine. The modeling approach needs to be efficient and easy to handle, so it does not require much effort to create to model. It will also benefit the non-experts, who normally do not have much prior knowledge, to create similar IC engine torque models. On the other hand, the modeling approach should be accurate enough so the created models can be used for rapid prototype and cross verification. Based on the empirical experiences, the discrepancy between the estimated and observed output values in an IC engine torque model should be less than 5% to meet such requirements.

As pointed out in [57], there was no generic approach to determine the best neural network architecture, such as the number of hidden neurons and the network layout, for a given problem just based on the problem description. A common approach was to start with a simple neural network structure if no prior data was available, then gradually build up the complexity based on the performance of the created model.

In this chapter, SLFN was selected as the fundamental network type to build the IC engine torque model due to its advantages in terms of structural simplicity and approximation capabilities. A few popular SLFN modeling techniques were briefly compared with their advantages and limitations identified. Eventually, the ELM based approach was chosen to create the engine torque model. In addition to the conventional modeling approach where all the points were equally considered, a weighted ELM model tuning approach was introduced to adapt to the demands of the real-world applications where increased accuracy was expected at certain engine operating conditions.

3.1 Selection of Modeling Methodology

Based on the discussions in Chapter 2, it was obvious that SLFN is one of the simplest network architectures with structural simplicity - only one hidden layer and no feedback from the output. It was also pointed out in previous research [23], [58], [59] that SLFN had the potential to approximate any continuous function with high accuracy. Consequently, SLFN was selected as the network structure to build the IC engine torque model in this research.

In addition to network architecture, the selection of the training algorithm is also critical to the overall performance of a neural network. A range of algorithms has been investigated. Each of them has its advantages and limitations as shown in Table 3-1. For instance, backpropagation is very popular and can be accurate in many cases, but it is slow in training and may suffer from local minima traps. Radial basis function is slow in training and the selection of the basis functions are application dependant. Deep learning is popular in image/video processing and pattern recognition. However, it is over complicated for the IC engine torque modeling application in this dissertation. Among the investigated training algorithms, extreme learning machine has many advantages over other competing ones in terms of algorithm complexity and training speed. In addition, it can yield quite a comparable model accuracy as pointed out in many studies [60]–[62]. Based on the heuristic considerations for speed, accuracy, and simplicity, ELM was selected as the training algorithm.

Table 3-1 Qualitative comparison of popular training algorithms

Training Algorithm	Key Advantage	Key Disadvantage
Backpropagation	Relatively small residual error	Slow training, local minima, prone to overtraining
Radial Basis Function (RBF)	Works well with discrete data	Slow training, application dependent basis function
Support Vector Regression (SVR)	Good for low dimensional data regression and classification	Slow training, less accurate for high dimensional data
Deep Learning	Prevalent in image and video processing, Pattern recognition	Overly complex for the current application, does not perform well for small data size, prone to over fitting
ELM	Extremely fast learning speed, competitive accuracy, simple handling	Unexploited benefit of weight tuning

3.2 Extreme Learning Machine Based Neural Network Approximation

As one of the powerful neural network techniques, ELM has extended applications in system identification, classification, pattern recognition, and deep learning. Though there are critics about the nature of random weight generation in ELM [63], its popularity keeps growing among the research communities over the years. To the best of the author's knowledge, though there are some automotive-related ELM applications, its capability in gasoline engine torque modeling has not been thoroughly explored.

3.2.1 Single-Hidden Layer Feedforward Neural Network Approximation

Single hidden-layer feedforward neural network has a very simple structure, which makes it much easier to be dealt with than many other complicated artificial neural networks (ANN)[2]. As pointed out in [23], [58], [59], an SLFN with at most $\widetilde{N}$ hidden neurons and with almost any nonlinear activation function can learn N distinct observations with zero error, where $\widetilde{N}$ is the number of hidden neurons and N is the number of distinct observations. It implies that it has the potential for accurate approximation. The description of a generic SLFN approximation problem is as follows.

For N arbitrary distinct samples (x_i, y_i), where $x_i = [x_{i1}, x_{i2}, \dots, x_{im}]^T \in R^m$ and $y_i = [y_{i1}, y_{i2}, \dots, y_{ik}]^T \in R^k$, considering the standard model for SLFN approximation with $\widetilde{N}$ nodes, the relation between the input and output data can be modeled as shown in (3-1),

$$y_j = \sum_{i=1}^{\widetilde{N}} \beta_i g_i(w_i \cdot x_j + b_i), \quad j = 1,2,\dots,N \tag{3-1}$$

where m is the dimension of the input data, k is the dimension of the output data, $w_i = [w_{i1}, w_{i2}, \dots, w_{im}]$ is the weight vector connecting the i_{th} hidden node and the input neurons, $\beta_i = [\beta_{i1}, \beta_{i2}, \dots, \beta_{ik}]^T$ is the weight vector connecting the i_{th} hidden node and the output neurons, and b_i is the bias of the ith hidden neuron. A structure of a general SLFN with fully connected nodes is illustrated in Figure 3-1. The term $w \cdot x$ is the inner product of w_i and x_j, and g is the activation function of the i_{th} hidden neuron. In practice,

[2] Artificial neural network and neural network will be interchangeably used in this dissertation.

g can be any nonlinear continuous function, such as sine, sigmoid, hyperbolic tangent, or radial basis function. Different neurons can have different g functions as well. However, for the sake of simplicity and complying with common practice, the same g functions will be applied to all the neurons in this research. The term $g_i(\cdot)$ represents the data feature extracted by the i_{th} neuron. The compact version of (3-1) can be written as:

$$Y = H\beta \tag{3-2}$$

where

$$H = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \cdots & g(w_{\tilde{N}} \cdot x_1 + b_{\tilde{N}}) \\ \vdots & \vdots & \vdots \\ g(w_1 \cdot x_N + b_1) & \cdots & g(w_{\tilde{N}} \cdot x_N + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}} \tag{3-3}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times k} \tag{3-4}$$

$$Y = \begin{bmatrix} y_1^T \\ \vdots \\ y_N^T \end{bmatrix}_{N \times k} \tag{3-5}$$

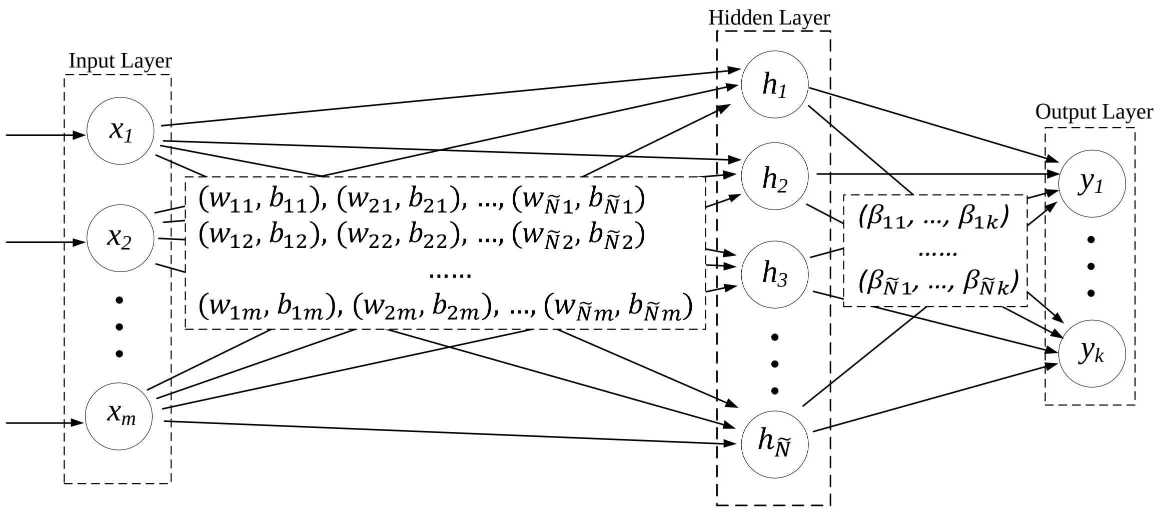

Input Layer
Hidden Layer
Output Layer
x_1
x_2
x_m
h_1
h_2
h_3
$h_{\widetilde{N}}$
y_1
y_k
$(w_{11}, b_{11}), (w_{21}, b_{21}), \ldots, (w_{\widetilde{N}1}, b_{\widetilde{N}1})$
$(w_{12}, b_{12}), (w_{22}, b_{22}), \ldots, (w_{\widetilde{N}2}, b_{\widetilde{N}2})$
……
$(w_{1m}, b_{1m}), (w_{2m}, b_{2m}), \ldots, (w_{\widetilde{N}m}, b_{\widetilde{N}m})$
$(\beta_{11}, \ldots, \beta_{1k})$
……
$(\beta_{\widetilde{N}1}, \ldots, \beta_{\widetilde{N}k})$

Figure 3-1 Basic structure of an SLFN

Let $\hat{\beta}$ be the estimation of β, $\widehat{w}$ be the estimation of w, $\hat{b}$ be the estimation of b, and $\hat{Y}$ be the estimation of Y. If the number of neurons equals the number of samples (generally H is a full rank square matrix in such a case), there exists a perfect estimation of β, which is $\hat{\beta} = H(\widehat{w}, x, \hat{b})^{-1} * Y$, to make $\|\hat{Y} - Y\| = 0$. However, in practice, the number of neurons is often much less than the number of samples ($\widetilde{N} \ll N$). As a result, there may not exist a perfect estimation of $\hat{\beta}$ that makes $\|\hat{Y} - Y\| = 0$. Therefore, the target of the SLFN approximation is to find out the appropriate estimations of $\hat{\beta}$, $\widehat{w}$, and $\hat{b}$, which can minimize the cost of the estimation E as shown in (3-6).

$$E = \|\hat{Y} - Y\| = \min_{\widehat{w}_i, \hat{b}_i, \hat{\beta}_i} \|H(\widehat{w}_i, \hat{b}_i) \cdot \hat{\beta}_i - Y\| \tag{3-6}$$

3.2.2 Extreme Learning Machine Approximation for SLFN

Since the gradient-based approximation method, such as backpropagation, has several prominent disadvantages as pointed out in [23], a different algorithm was preferred in this research. Though there are some debates about the random weight generation feature of ELM in the machine learning community [63], researchers have revealed that ELM has superior training speed and comparable accuracy over many other popular SLFN training algorithms, such as RBF, SVM, and AdaBoost [61]. However, its application in gasoline engine torque prediction has not been explored.

Among many ANN training algorithms, a neural network with random weights (NNRW) is appealing due to its advantages in network simplicity, faster training speed, and competitive accuracy [64]. Compared with other NNRWs, such as Random Vector Functional Link (RVFL) [48] and standard feed-forward neural network with random weights (the Schmidt's method) [47], ELM outshines due to the advantages such as setting the bias of the output node to zero, transforming different hidden nodes to one unified form, and generating random node parameters prior to knowing the training data [65]. In addition, ELM can be extended to multiple hidden layer architecture and has a strong potential for big data analytics [64], [66].

Therefore, in this research, an ELM based SLFN gasoline engine torque model was explored and assessed. The uniqueness, as well as the competitiveness, of the traditional ELM approach, largely lies in the fact that the weights and biases of the hidden neurons do not need to be tuned. With prefixed $\hat{w}$ and $\hat{b}$ (which are usually randomly generated numbers between 0 and 1), the approximation problem stated in (3-6) can be simplified as (3-7), in which only β needs to be optimized.

$$E = \left\|\hat{Y} - Y\right\| = \min_{\hat{\beta}_i}\left\|H\hat{\beta}_i - Y\right\| \tag{3-7}$$

The minimal norm least square estimation of β is as the following [23]:

$$\hat{\beta} = H^{\dagger}Y \tag{3-8}$$

where $H^{\dagger}$ is the Moore-Penrose inverse of matrix H. The unbiased least square estimation of the matrix is:

$$H^{\dagger} = \begin{cases} (H^T H)^{-1} H^T, & if\ H^T H\ is\ nonsingular \\ H^T (HH^T)^{-1}, & if\ HH^T\ is\ nonsingular \end{cases} \tag{3-9}$$

However, according to the ridge regression theory [67], the variance of the above unbiased least square estimation can be large so the estimation may be far from the true value when multicollinearity exists. In addition, when HH^T (or $H^T H$) is not invertible, there can be no unique solution of $\hat{\beta}$. Therefore, further regulation of $\hat{\beta}$ is needed. In order to improve the stability of the estimation, a small positive real number k is added to all the diagonal elements of the correlation matrix.

As a result, the ridge regression estimation of β is:

$$\widehat{\beta^*} = \mathrm{H}^{\dagger}\mathrm{Y}$$

$$= \begin{cases} (kI + H^T H)^{-1} H^T Y, & if\ H^T H\ is\ nonsingular \\ H^T (kI + HH^T)^{-1} Y, & if\ HH^T\ is\ nonsingular \end{cases} \tag{3-10}$$

Consequently, the estimated output of the network becomes:

$$\hat{Y} = H\widehat{\beta^*} \tag{3-11}$$

The variance of the ridge regression estimator is

$$Var(\widehat{\beta^*}) = \begin{cases} \sigma^2 Z(H^T H)^{-1} Z^T, & if\ H^T H\ is\ nonsingular \\ \sigma^2 Z(HH^T)^{-1} Z^T, & if\ HH^T\ is\ nonsingular \end{cases} \tag{3-12}$$

where

$$Z = \begin{cases} (kI + H^T H)^{-1}, & if\ H^T H\ is\ nonsingular \\ (kI + HH^T)^{-1}, & if\ HH^T\ is\ nonsingular \end{cases} \tag{3-13}$$

The squared bias of the ridge regression estimator is

$$Bias^2(\widehat{\beta^*}) = k^2 \beta^T Z^{-1} \beta \tag{3-14}$$

From the above equations, the total variance of $\widehat{\beta^*}$ is a monotonically decreasing function with respect to k, while the total squared bias of $\widehat{\beta^*}$ is a monotonically increasing function with respect to k. As shown in Figure 3-2. with the k value approaching to zero, the total variance decreases sharply, though the squared bias would increase. According to the existence theorem, these properties lead to a conclusion that there always exists such a k value that makes the mean square error (MSE) of $\widehat{\beta^*}$ smaller than the MSE of $\hat{\beta}$, as indicated by the red dotted line in the figure. Therefore, $\widehat{\beta^*}$ is a closer estimation of the true model parameter β. Compared with the gradient descent based algorithms, this method almost only requires a matrix inversion operation to determine the unsolved model parameters. Therefore, the training speed of ELM is significantly faster than many of its competitors [23], [62].

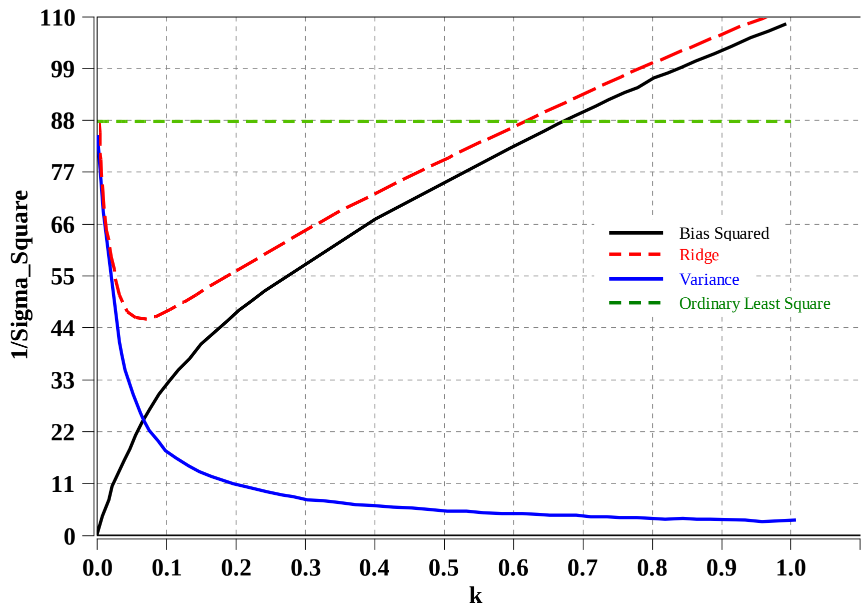


Figure 3-2 Ridge Regression Mean Square Function (adopted from [67])

3.2.3 Determine the k Value for Regression Problem

As pointed out in the previous section, the small number k in (3-10) needs to be determined for better regression performance. The best k is proved to be a positive value very close to zero, but it is also impossible to calculate it out mathematically, as it ultimately depends on the unknown parameter being estimated [67], [68]. To find a closer estimation of the true k, several approaches have been proposed, such as ridge trace method [67], Akaike's information criterion (AIC) [69], and Adjusted R^2 [70], [71]. However, each of these approaches alone has some unique disadvantages. For instance, ridge trace may be too subjective so that it is difficult to be generalized for the new data set. AIC can be good in selecting the best model among all candidates, but it requires the best model to be well established among the candidate models, which is often impractical [72]. Adjusted R^2 is useful for evaluating parameters, but it is incapable of model selection.

In this research, a ten-fold cross-validation approach incorporated with ridge regression was used to determine a proper k value which could produce an error-minimized estimation of β. This method had two main benefits: it delivered a deterministic result, so the process could be computerized without human interference; the result was statistically representative, so it had higher chances to fit the complete targeted data set. The key steps of the approach are listed as follows.

1) The training data set was randomly divided into ten equal (or almost equal if there was residue) partitions, such as $X = (X_1, X_2, \dots, X_{10})$.
2) For each $n = 1,2,3, \dots ,10$, the n_{th} partition was set as the validation data set and all the rest data was taken as the training data set.
3) For each potential k value, $\widehat{\beta^*}$ was computed using the training data set according to (3-10), then $\hat{y}$ was calculate according to (3-11).
4) The distance between the estimated $\hat{y}$ and the expected y was calculated using the validation data set according to (3-15),

$$E(\hat{y} - y)_k = sqrt(\sum_{i=1}^{m} (\hat{y}_i - y_i)_k^2 \,/m) \tag{3-15}$$

where m is the size of the testing data set, k is the partition number.

5) Steps 3) and 4) were repeated for all the data partitions. The cross-validation distance was calculated for the overall data set following (3-16).

$$CV\_E(\hat{y} - y)_k^{(10)} = \sum\nolimits_{n}^{10} E(\hat{y} - y)_k \tag{3-16}$$

6) The k value which had the minimum overall cross-validation distance was chosen, as showing (3-17)

$$k = \underset{k}{argmin}\, CV\_E(\hat{y} - y)_k^{(10)} \tag{3-17}$$

3.2.4 Weighted ELM Regression for Engine Model

Though the ELM method described in the above sections could optimize ordinary linear regression by reducing the overall distance between the estimated and expected values, it might not be well suited for the applications where higher accuracy was preferred at certain points. As indicated in (3-7), since the optimization target was to minimize the squared accumulative distance of all the points in the data set, each point would contribute an equal weight to the squared accumulative distance. Therefore, the optimization might be affected by the availability and distribution of the data samples. The outcome might not suit certain practical IC engine applications.

For instance, as pointed out in [24], though an engine map spreads over a wide range of operating conditions, a typical light-duty vehicle engine actually spends most of the time at the low to medium speed operating (LMSO) conditions in the standard vehicle and fuel emission tests, such as the United States Environmental Protection Agency’s Urban Dynamometer Driving Schedule (UDDS) and Highway Fuel Economy Test (HWFET), as shown in Figure 3-3. The blue crosses are the engine speed and torque conditions during the tests when periodic snapshots are taken. Based on the observation, it is desirable that an engine regression model could have higher accuracy at or near these operating points, even though the accuracy at other less frequently operating points may have to be compromised slightly. In the past, a few researchers proposed a weight matrix on ELM algorithm to solve imbalanced sampling and multiclass classification problems [61], [73], [74]. However, to the best of the authors' knowledge, there was no exploration of using weight regulated ELM to enhance a regression model only at certain data areas without changing the input data sets. Hence, this study proposed the approach of implementing weight regulated ELM on an SLFN linear regression model to serve such a purpose.

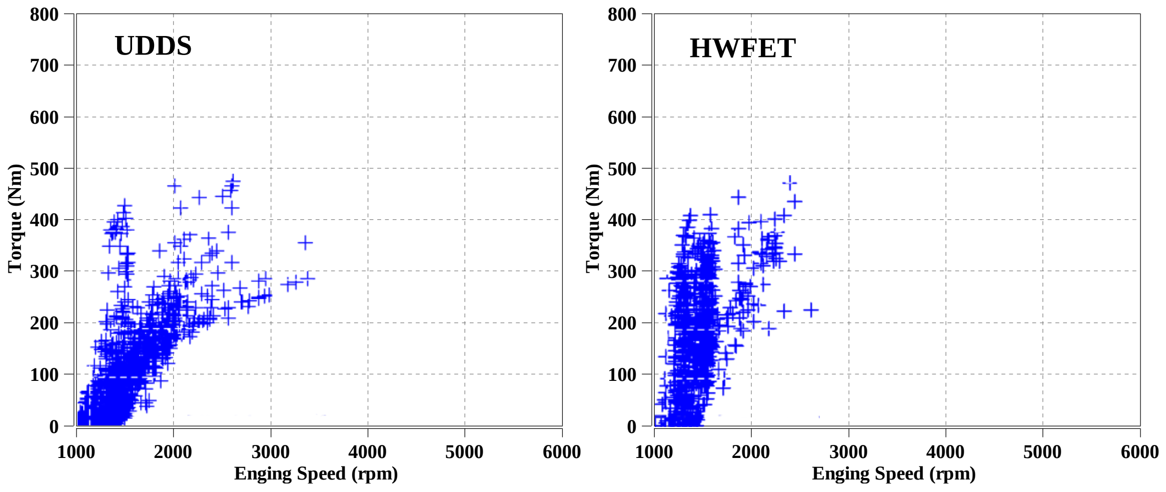


Figure 3-3 Engine speed-load operating points visited by the simulated conventional vehicle over each drive cycle. The blue crosses represent the engine state at 1s intervals (adapted from [24]).

By introducing an $N \times N$ diagonal matrix $W = diag[w_1, w_2, \cdots, w_N]^T$, where N is the number of data samples, each data point is assigned an adjustable weight factor w_i>0, where i is the position of the input data point. By default, w_i equals 1 at every point, which means no weight is added at that point. Whereas at the point where higher accuracy is desired, w_i should increase accordingly. Based on the study by Zong *et al.*, the optimization target with the weighted function becomes [73]:

Minimize: $$L_{P_{ELM}} = \frac{1}{2}\|\beta\|^2 + \frac{1}{2}kW\sum_{i=1}^{N}\|\xi_i\|^2 \tag{3-18}$$

Subject to: $$\begin{cases} h(x_i)\beta = y_i - \xi_i \\ \xi_i = y_i - \widehat{y_i} \end{cases} \tag{3-19}$$

where $\xi_i$is the training error with respect to the training sample x_i.

A single Lagrangian function is possible to be defined in such a way that all the constraints and the objective function are combined. According to Karush-Kuhn-Tucker (KKT) theorem, the equivalent optimization target is [73], [75],

$$L_{P_{ELM}} = \frac{1}{2}\|\beta\|^2 + \frac{1}{2}kW\sum_{i=1}^{N}\|\xi_i\|^2 - \sum_{i=1}^{N}\alpha_i(h(x_i)\beta - y_i + \xi_i) \tag{3-20}$$

where α_i is the Lagrange multiplier.

By setting the partial derivatives of (α_i, ξ_i, β) to zero, the optimal condition of KKT is as follows.

$$\begin{cases} \dfrac{\partial L_{P_{ELM}}}{\partial \beta} = 0 \\ \dfrac{\partial L_{P_{ELM}}}{\partial \xi} = 0 \\ \dfrac{\partial L_{P_{ELM}}}{\partial \alpha_i} = 0 \end{cases} \quad \Rightarrow \quad \begin{cases} \beta = \sum_{i=1}^{N}\alpha_i h(x_i)^T = H^T\alpha_i \\ \alpha_i = kW\xi_i \\ h(x_i)\beta - y_i + \xi_i = 0 \end{cases} \tag{3-21}$$

As a result, the solution to the optimal condition is:

$$\widehat{\beta^*} = \begin{cases} H^T(kI + WHH^T)^{-1}WY, & \text{when N is small} \\ (kI + H^TWH)^{-1}H^TWY, & \text{when N is large} \end{cases} \tag{3-22}$$

Once $\widehat{\beta^*}$ is determined, the model is considered as trained and ready for testing. Then the predicted model output data can be calculated according to (3-11).

3.3 Summary

This section presented an ELM-based approach to building a single-hidden layer feedforward neural network for system modeling. The model adopted random weight generation at the hidden neural nodes but analytically calculated the weights that connect the hidden neurons and the output neurons. Ten-fold cross-validation was utilized to stabilize the model performance. In addition, a weighted approach was utilized which enabled the model to have further enhanced accuracy at or near certain desired data points.

Chapter 4. IC Engine Experiment Setup and Data Acquisition

In this chapter, a comprehensive IC engine mapping experiment is presented. The test covered the normal operating conditions of the tested engine, with the engine speed ranging from 1000 rpm to 4500 rpm, and the engine torque ranging from idle to full load. The test was conducted using facilities that were built in complying with the industry-leading engine test requirements. The test procedure also strictly followed the production engine test standard. Over 3300 high-quality data points were collected in the experiment. The number of data points in this study greatly surpasses the number of data points that were used in the experiments reported in previous research.

4.1 Experiment Design and Data Acquisition Target

In order to verify the proposed modeling approach and evaluate its performance for IC engine torque modeling, a comprehensive gasoline engine mapping test was designed. Unlike previous research studies [13]–[17], [44], [76], where the data set was relatively small (about 80~130 points), over 3300 data points, which covered all the normal operating points of the tested engine, were collected in this experiment. The experiment was conducted at an industry-leading engine testing facility. The test conditions were in line with the stringent industrial test standards. To ensure the quality of data, the data was collected while the engine was running at steady states. The data was also properly averaged to reduce the effects of possible disturbance during engine operation.

4.2 Engine Dynamometer Test Cell Setup

The model was validated on an 8-cylinder, 4-stroke, dual-equal variable camshaft timing (VCT) gasoline engine. The experiment was performed at an industry-leading engine dyno testing facility. The diagram of the key test cell setup is shown in Figure 4-1.

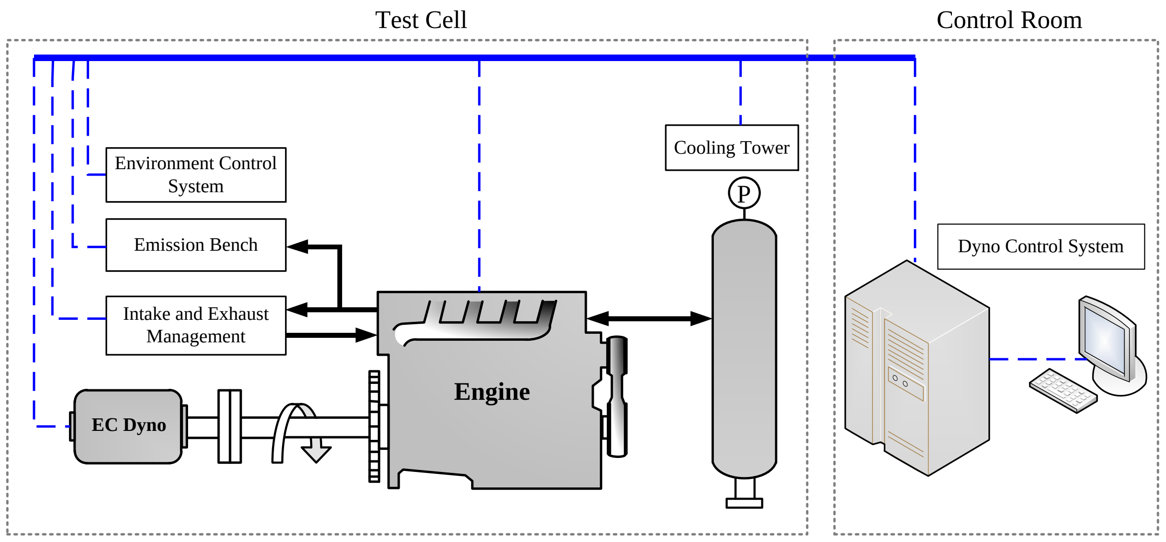


Figure 4-1 Test system diagram

The sophisticated dyno control system was the command center that controlled the dyno operation, cell equipment function, and test data logging. The data from the test cell measurement and the engine control unit (ECU) was recorded. The engine was connected to a high-performance Eddy Current dynamometer (as showing in Figure 4-2) which was capable of measuring torque within ±0.5% error along the full scale and maintaining rotation speed within ±0.1% error at the maximum revolving speed [77]. A heat exchange cooling tower was used to maintain the engine coolant out temperature and pressure at 90.5±2.8 ℃and 145±3.4 kPa, respectively. The maximum engine oil sump temperature was capped at 90.5±2.8 ℃ by an external oil cooler heat exchanger throughout the whole engine test map. The environment control system regulated the ambient temperature and supplied fresh air to maintain proper environment air quality. The combustion air was circulated by the intake and exhaust management system. Fresh air was blown vertically by an intake air pipe connected from the air conditioning unit to the inlet of the engine air box. The temperature and humidity of intake air were maintained at 23.8±2.8 ℃ and 7.9±0.7 g_H_2O/kg_Air, respectively. The emission gases were sampled by the emission bench for composition analysis. The ECU was interfaced with VISION 5.0 software [78], through which the ECU parameters were monitored and controlled in real-time. An Ethernet-based communication protocol (ASAM ASAP3) was used to exchange information between the dyno controller and VISION 5.0 software, through which the ECU parameters were updated to the dyno control system at a frequency of 20 Hz.

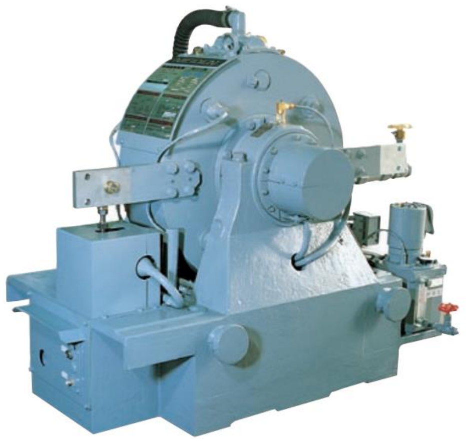


Figure 4-2 Eddy current dynamometer

4.3 Test Matrix and Control Parameters

The aim of the engine experiment was to map the engine operation with fine steps and collect genuine data during the whole process. The collected data should be comprehensive and accurate so that it can serve as a trusted representative set for the neural network to build the corresponding engine model. Additionally, such a data set should serve as solid real-world evidence to support the practical utility of the created engine model.

4.3.1 Test Points and Data Acquisition

As pointed out in [79], an engine map should describe the behavior of the engine in terms of engine Revolutions-Per-Minute (RPM), Load, Air-to-Fuel Ratio (AFR), Exhaust Gas Recirculation (EGR), Spark Advance Angle, and Variable Cam Timing (VCT). The output should include power, torque, and regulated emission gases. It is necessary to point out that though the previous studies mentioned in Section 1.2 have created models for the engine or its subsystem, the number of data points that were used (approximately 80-130 points) might not be sufficient to describe a realistic engine map. As indicated in Figure 4-3, with the increase of the number of independent control variables, the size of the engine control map would rise exponentially. For instance, if an engine control map is as simple as a speed–load map only, and the numbers of breakpoints of the engine speed and load are m and n, respectively, then the size of the engine control map is $m \times$n. If VCT is an added control parameter and its breakpoint is k, then the size of the engine control map will grow to $m \times n \times k$. If EGR is another added control parameter that has h breakpoints, then the size of the engine control map shall grow to $m \times n \times k \times h$. As the total size of the engine map keeps growing, the complexity and the number of points to characterize the engine would consequently increase drastically. So as the data used to create the map. Therefore, a large amount of data is usually needed when it comes to evaluating a neural network based modern IC engine modeling methodology. In addition, it also helps to certify that the created engine models would be representative of the real-world engine over a wide range of operating conditions.

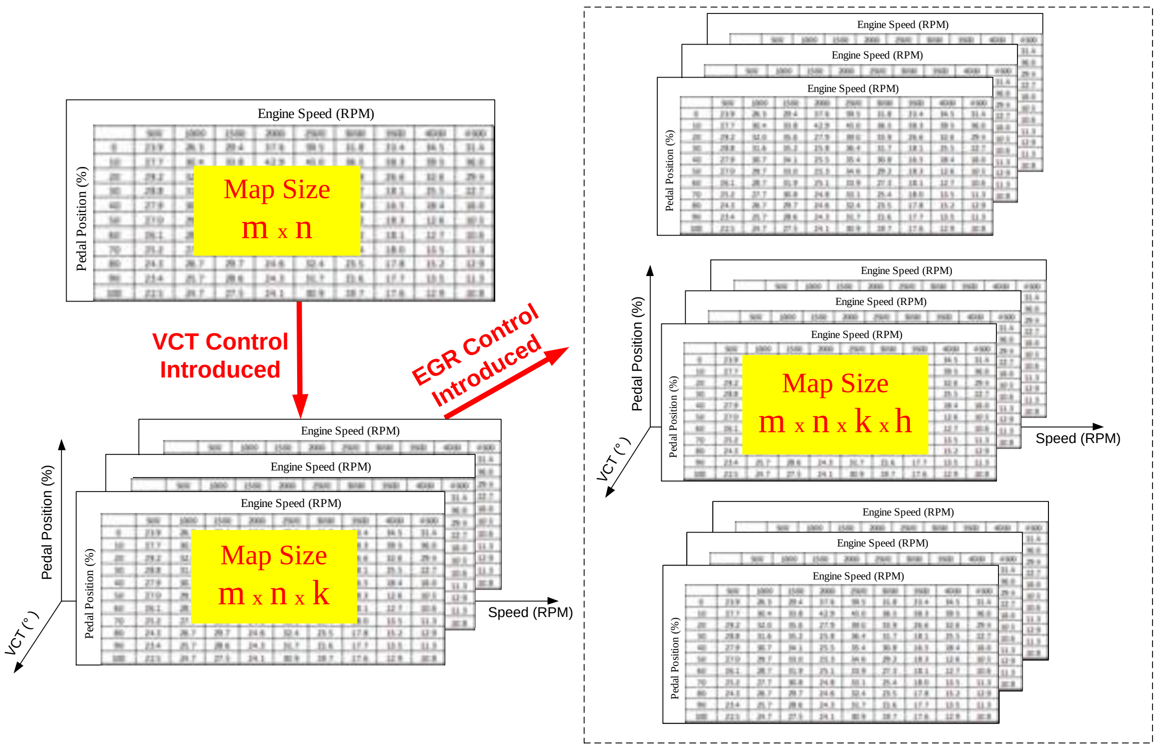


Figure 4-3 Illustration of the exponential rise of the map size and complexity as the control parameters increase (the table contents are blurred for proprietary reasons)

Therefore, in this research, a complete engine mapping test was carried out to provide a comprehensive data set for engine modeling and to test the generalization of the created model. A combination of speed, load, VCT, and spark sweeps was designed for this experiment. The sweeps were comprehensive enough to cover all the normal operating conditions of this engine.

4.3.2 Range of Controlling Parameters

The range and step length of each control parameter are shown in Table 4-1.

Table 4-1 Range of control parameters

ECU Parameter	Sweep Range	Step Size
Engine Speed	1000 - 4500 rpm	500 rpm
Pedal position	0 - 100%	10%
VCT	0 - 50 deg	5 deg
Spark timing	borderline – 4 deg before borderline	1 deg

The data was collected for constant throttle spark timing sweeps, during which RPM, pedal position (which was equivalent to load in this application), and VCT were fixed. AFR was calibrated to remain at a stoichiometric ratio wherever possible and enriched to suppress knock and achieve maximum torque at high loads. Spark advance angle was varied at each condition from borderline to borderline-4 degree crank angle at a step length of 1 degree crank angle. In order to collect only the steady-state data, the test was scheduled to stay at each running condition for 3 minutes for the engine to stabilize. A data point was logged at 10 seconds to the end of each step. The logged value was the average of the data sampled in the previous 30 seconds. Then spark timing sweeps continued and the data was collected at each combination of RPM, pedal, and VCT. By following such a sweeping sequence, the transition disturbance between different steps was reduced to a minimum.

Overall, it took more than 10,000 minutes (over 166 hours) to complete the engine mapping test. The total number of the collected data samples was 3356[3], which substantially surpassed the number of data samples reported in the ANN models in Section 1.2. All the dyno cell measurements and the ECU data were sampled at a 10 Hz rate by the dyno control system.

4.4 Model Parameter Selection and Data Partition

The focus of this study was to create a model that was able to predict gasoline engine output torque from a given engine input data set. The predicted values were compared with the measured values to assess the model's prediction capability. The evaluation metrics were root mean square error (RMSE) and correlation coefficient. Over 300 parameters were recorded at each point during the engine mapping process. These parameters included the test environmental parameters, cell control parameters, powertrain control module (PCM) parameters, and inferred parameters. Only some of the parameters were needed to model a gasoline engine. Taking the engineering sense into consideration (assuming the prior knowledge was available to the user), the parameters that directly affected the in-cylinder combustion process were selected, such as VCT, spark advance angle, and engine speed. Considering from a non-engine expert perspective (assuming no prior knowledge was available to the user), the parameters that were less relevant to the engine combustion process, such as barometric pressure (a redundant parameter in this case) and exhaust gas

[3] Some duplicated points are removed. For instance, at some high load conditions, the engine load has already reached maximum while pedal has not reached 100% yet. In such cases, duplicated points will occur.

temperature (a result of combustion rather than a cause of it), were also included. A list of the model input/output parameters and their statistics in the training and the testing data sets is shown in Table 4-2.

All the collected data points were randomly partitioned as the training data set, validation data set and the testing data set at a ratio of 8:1:1 (or very close to 8:1:1). The training data set was used to train the model, whereas the validation data set was used to validate the model and tune the model hyperparameter (such as *k*). The handling of the training and validation data sets together completed the model training process, which eventually delivered a determined engine torque model. The testing data set was kept “unseen” (isolated) during the whole training process. They were used to provide an unbiased evaluation of the determined model.

As a result, there were 2685 data points in the training data set, 335 data points in the validation data set, and 336 data points in the test data set.

Table 4-2 Model input/output parameters and statistics

Model Parameters	I/O Type	Training Data Set			Validation Data Set			Testing Data Set		
		Min	Mean	Max	Min	Mean	Max	Min	Mean	Max
Engine Torque (Nm)	Output	55.3	323.7	671.6	54.6	321.3	635.6	55	326.7	657.4
Engine Speed (rpm)	Input	999	2658	4504	999	2728	4504	999	2586	4504
Intake Manifold Pressure (kPa)	Input	24.9	71.4	100.3	24.9	70.4	100.3	25	71	100.3
Barometric Pressure (kPa)	Input	98.2	99.7	100.9	98.2	99.7	100.9	98.2	99.8	100.9
Intake Air Temperature (℃)	Input	17.7	24.2	35.5	19.3	24.2	35.3	19.4	24.2	34.6
VCT (deg)	Input	0	25	50	0	24	50	0	23	50
Spark Advance Angle (deg)	Input	0	28	55	6.5	28	55	5.75	27.9	55
Lambda (-)	Input	0.84	0.96	1	0.84	0.95	1	0.84	0.96	1
Exhaust Gas Temperature (℃)	Input	362.5	712.2	938.7	363.1	716.4	929.3	398.3	707.3	934.2
Exhaust Gas Pressure (kPa)	Input	1.7	5.6	13.6	1.9	5.7	13.5	2.1	5.5	13.6

4.5 Summary

This section described an IC engine mapping test setup and the data acquisition approach for the test. The designed test provided real-world engine data to evaluate the proposed modeling algorithm. Engine dyno setup, test procedures, and data partitions were explained. The engine test was conducted in line with industry-leading standards so as to guarantee the completeness and accuracy of the collected data. Over 3300 engine mapping points were collected. To the best of the author’s knowledge, no previous research studies used such a scale of engine mapping data for neural network based IC engine torque modeling. The scale and quality of the test data not only demonstrated the capability of the proposed modeling methodology in handling a large amount of data but also proved that the created IC engine torque model was applicable in real-world scenarios.

Moreover, parameter selection and data partition schemes for the neural network were discussed. In order to make sure the proposed IC engine torque model had good generalization ability, the data was randomly partitioned in an 8:1:1 ratio across the training, validation, and testing data sets. In addition to the IC engine torque-related parameters, some irrelevant parameters were also included in the model input data. Such arrangement simulated the use of non-experts who did not have much engine-related knowledge to support parameter selection. It could also help evaluate the robustness of the proposed algorithm when irrelevant inputs were involved in the modeling.

Chapter 5. Experimental Evaluation of IC Engine Torque Model

The evaluation results of IC engine torque model were described in this chapter. The model was presented in Chapter 3. The experimental data collected using the setup described in Chapter 4 was used to train and evaluate the model. The model output torque was compared with the experimental data to assess the model performance. The evaluation metrics used were root mean square error (RMSE) and correlation coefficient in this case. Both the weighted and unweighted modeling approaches were presented in this chapter.

5.1 Neural Network Training and Performance Evaluation

5.1.1 Data Preparation

As pointed out in [80], [81], the empirical initialization of the weights and biases in the random weight networks with the non-iterative training algorithms, such as ELM, may not always lead to optimal performance. Therefore, a good selection of the initialization range is required to achieve desirable performance. In addition, since many of the activation functions (such as sine or sigmoid function) have periodic or asymptotic output ranges, it is preferable that given the randomized weights and biases, the output of the neuron response should avoid such ranges, so the change of the input values can be reflected sufficiently on the output.

In this research, the activation function was a sigmoid function. All the input and output parameters (such as those listed in Table 4-2) were normalized to [0, 1], where 0 corresponded to 95% of their minimum values and 1 corresponded to 105% of their maximum values, respectively. Such normalization can create some room for redundancy in case the estimated output values fell slightly out of the range of the training values. The randomization range of the hidden neuron weights and the biases were both selected as [-1, 1] after parameter tuning. The considerations in the selection of the model initialization parameters are discussed in detail in Section 5.1.3.1. For this application, such settings would make the overall hidden neuron inputs reside in [-3.6, 2.9], which was a suitable input range to avoid the asymptotic output range of the activation function. After model computation, the output data was restored to their original scales. The network structure with data normalization and restoration is illustrated in Figure 5-1. The training data set was used to train the network in order to extract the data feature. The test data set was used to provide unbiased evaluations of the model once the model was determined.

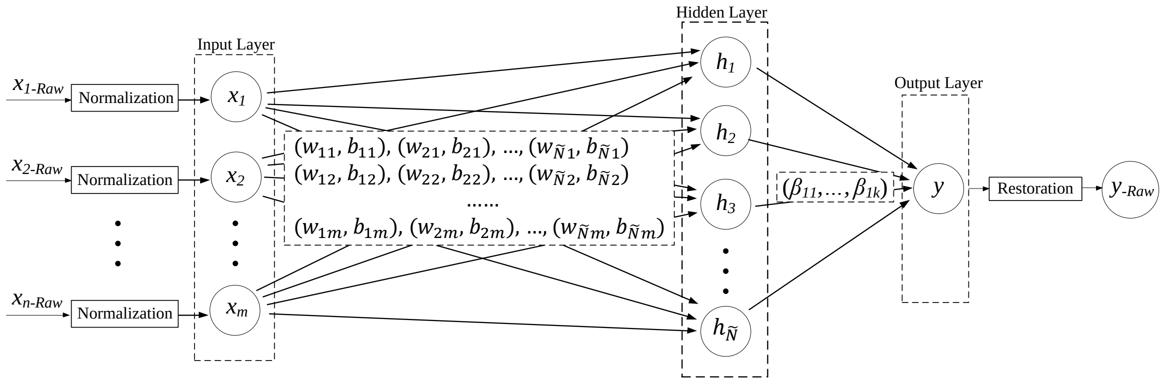

Hidden Layer
Input Layer
Output Layer
$x_{1\text{-}Raw}$
$x_{2\text{-}Raw}$
$x_{n\text{-}Raw}$
Normalization
Normalization
Normalization
x_1
x_2
x_m
$(w_{11}, b_{11}), (w_{21}, b_{21}), \ldots, (w_{\tilde{N}1}, b_{\tilde{N}1})$
$(w_{12}, b_{12}), (w_{22}, b_{22}), \ldots, (w_{\tilde{N}2}, b_{\tilde{N}2})$
......
$(w_{1m}, b_{1m}), (w_{2m}, b_{2m}), \ldots, (w_{\tilde{N}m}, b_{\tilde{N}m})$
h_1
h_2
h_3
$h_{\tilde{N}}$
$(\beta_{11}, \ldots, \beta_{1k})$
y
Restoration
$y_{\text{-}Raw}$

Figure 5-1 Network structure with data normalization and restoration schemes

To finalize the model structure, the number of hidden neurons also needs to be determined. Traditionally, the manual trial-and-error approach is used to find an acceptable number of hidden nodes. However, this approach is time-consuming and cannot always guarantee that the optimal number of hidden nodes exists in the experimental range. In this research, random orthogonal projection based enhanced bidirectional ELM (OEB-ELM) [82] was adopted to find the optimal number of hidden neurons. The key steps of this approach are briefly covered in Section 5.1.3.2. Compared with B-ELM [83] and EB-ELM [84] algorithms, this approach led to better generalization and a more compact network structure. Five sets of the orthogonalized initialization matrices were evaluated at each odd hidden node step to search for the appropriate initialization set that could deliver the minimum residual error. In order to explore how the change of the number of hidden neurons would affect the performance of the model, sweeps of the number of hidden neurons versus the normalized RMSE were conducted on the training data set and validation data set. The normalization is performed by rescaling all the RMSE values relative to the one when the number of hidden nodes equals one. The residual error exit condition was not considered in the sweep so that the full curve trend was revealed. As shown in Figure 5-2, when the number of hidden neurons reached 36, the normalized RMSEs of both data sets started to converge, with both values smaller than 0.1. Based on this discovery, the number of hidden neurons was determined as 36 for this model.

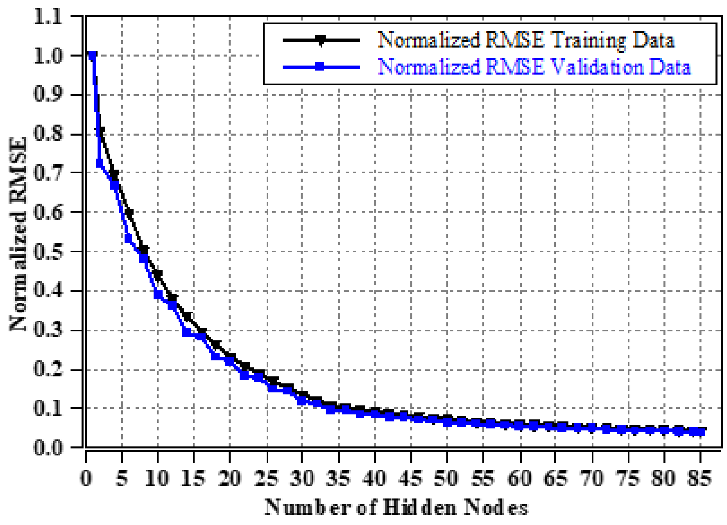


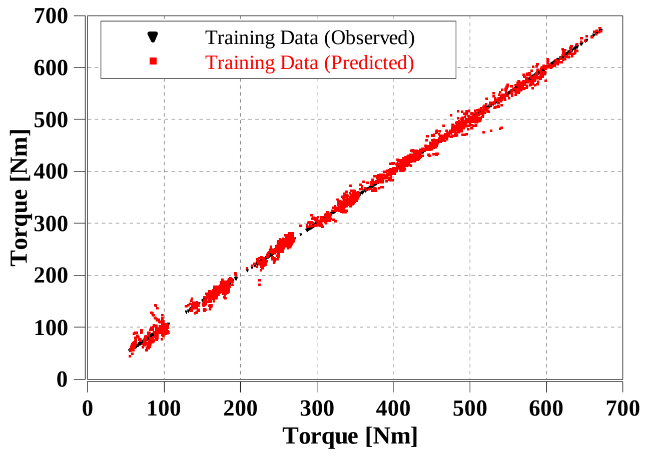
Figure 5-2 Number of hidden neurons vs. normalized RMSE

5.1.2 Experimental Evaluation Results

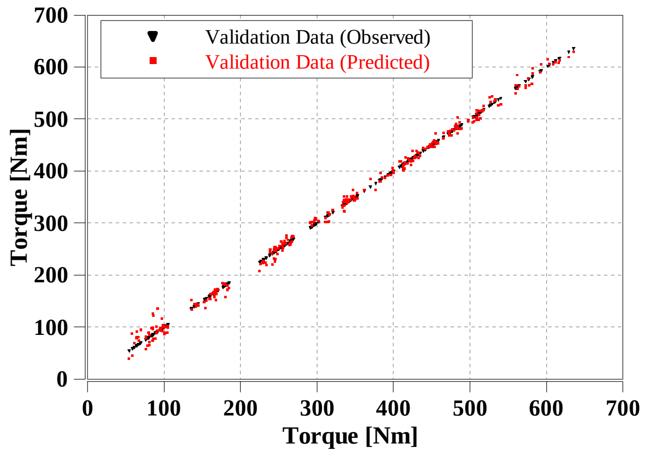
The training data set was firstly fed into the neural network with the above-determined model parameters to learn the features of the training data. The performance of the trained model was then assessed by checking the RMSE of the validation data set with respect to the regulation factor k. As described in Section 3.2.3, a deterministic regulation factor k was derived from the ten-fold cross-validation approach. If the RMSE of the validation data set was within 5% of the range of RMSE of the training data, the regulation factor k would be deemed as acceptable and the model would be evaluated with the test data set. Otherwise, the factor k would have to be re-calculated. In this gasoline engine torque model, the optimal k was determined to be 5.007e-6 through this approach.

The evaluation results of the training data set, the validation data set, and the testing data set are presented in Figure 5-3 (a), (b), and (c), respectively. The observed data and the model predicted data are overlaid in the figures, in which the black triangles represent the observed data; the red squares represent the predicted data. It can be seen that the predicted data are closely clustered around the observed data along the 45-degree line in all the data sets. This pattern indicates that the model predicted values have a very high degree of accuracy, and the model has good generalization ability across different data sets.

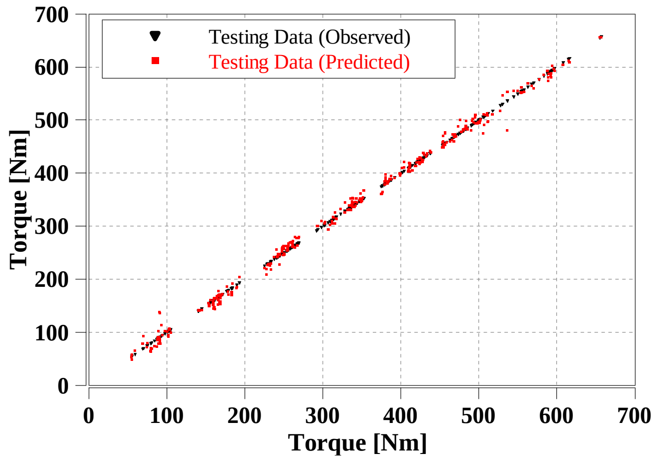
Specifically, the RMSE was 8.75 Newton Meter (Nm) and the correlation coefficient was 0.9983 between the predicted and observed values of the training data set as shown in Figure 5-3 (a). The RMSE was 9.18 Nm and the correlation coefficient was 0.9983 between the predicted and observed values of the validation data set as shown in Figure 5-3 (b). The RMSE was 9.29 Nm and the correlation coefficient was 0.9981 between the predicted and observed values of the testing data set as shown in Figure 5-3 (c). The results showed good generalization across the training, validation, and testing data sets. Considering the wide torque distribution in the data sets and according to the practical engineering requirements (as stated in Chapter 3), the results were considered as within acceptable range. A summary of the model evaluation results is presented in Table 5-1.



(a) Training data set



(b) Validation data set



(c) Testing data set

Figure 5-3 Torque model regression results

Table 5-1 Summary of the model evaluation results

	No. of Points	RMSE (Nm)	Corr. Coef
Training Data	2685	8.75	0.9983
Validation Data	335	9.18	0.9983
Testing Data	336	9.29	0.9981

5.1.3 Parameter Selection Considerations

5.1.3.1 Initialization Range

As mentioned in Section 5.1.1, all the input/output parameters were normalized to [0, 1] with 0 and 1 corresponding to 95% and 105% of their minimum and maximum values, respectively. Both the weights and biases of hidden neurons were initialized randomly in [-1, 1] under the uniform distribution. Three primary considerations led to such initialization settings, listed as follows.

1) The input parameters were normalized to [0, 1] to reduce the chances that a large-value parameter would dominate the total response of the neurons. For instance, in our case engine speed ranged from 1000 to 4500 rpm, whereas exhaust gas pressure resided between 2 to 12 kPa. Such a normalization range effectively prevented the response of the exhaust gas pressure from being dwarfed by the response of the engine speed.
2) Setting 95% and 105% of the minimum and maximum values corresponding to 0 and 1 (rather than traditionally just setting the minimum and maximum values to 0

and 1) offered some room for redundancy in data reconstruction, in case the estimation of the unseen values fell slightly out of the range of the training values.

3) Setting the randomization interval of both the weights and biases to [-1,1] was to make sure that the overall input $(wX + b)$ of the activation function (sigmoid function) fell in such a range that the change of the input values (X) would be reflected sufficiently in the neuron response.

Inspired by the research studies in [80], [81], [85], more trials were conducted with different initialization ranges of the weights and biases of the model. The goal was to explore a possible better initialization range or to justify the current initialization range for the engine torque model. Each of the initialization combinations had 1000 repetitions. The averaged result is reported in Table 5-2.

Table 5-2 Experimental results of weights and biases randomization

Randomization			Mean (of 1000 trials)		
Set	Weights Range	Biases Range	RMSE_Training	RMSE_Validation	RMSE_Testing
1	[-1, 1]	[0, 1]	8.8642	8.6269	10.2115
2	[-5, 5]	[0, 1]	17.6299	17.8968	18.0910
3	[-10, 10]	[0, 1]	28.4095	28.9260	28.2529
4	[-20, 20]	[0, 1]	38.6620	40.0870	39.8914
5	**[-1, 1]**	**[-1, 1]**	**8.8983**	**8.6049**	**9.0448**
6	[-5, 5]	[-1, 1]	17.5118	17.5713	18.5456
7	[-10, 10]	[-1, 1]	28.4811	28.3062	29.2314
8	[-20, 20]	[-1, 1]	38.8517	40.7123	38.6112
9	[-1, 1]	[0, 5]	9.1710	9.9599	9.3129
10	[-5, 5]	[0, 5]	18.1878	19.1947	18.9728
11	[-10, 10]	[0, 5]	28.8576	30.2735	29.3644
12	[-20, 20]	[0, 5]	38.9344	39.1467	39.6945
13	[-1, 1]	[0, 10]	13.1210	12.7263	12.9518
14	[-5, 5]	[0, 10]	20.9542	23.0103	20.5373
15	[-10, 10]	[0, 10]	30.6106	31.0789	30.7309
16	[-20, 20]	[0, 10]	39.6459	39.7117	40.3064

As can be seen in the table, multiple acceptable randomization intervals were available for this model. For instance, set 1, 5, and 9 all delivered comparably small RMSEs.

In addition, the ranges of the overall input ($wX + b$) for Set 5 and Set 6 were plotted, as shown in Figure 5-4, to explore what may have caused the difference between the RMSEs. As can be seen in the figure, the overall input range with set 5 initialization was [-3.6, 2.9], whereas the overall input range with set 6 initialization was [-17.5, 18.3]. Provided the activation function was sigmoid, a significant part of the overall output with set 6 initialization was suppressed, which could lead to crippled feature extraction of the neurons. This observation also partially explained the better initialization of set 5.

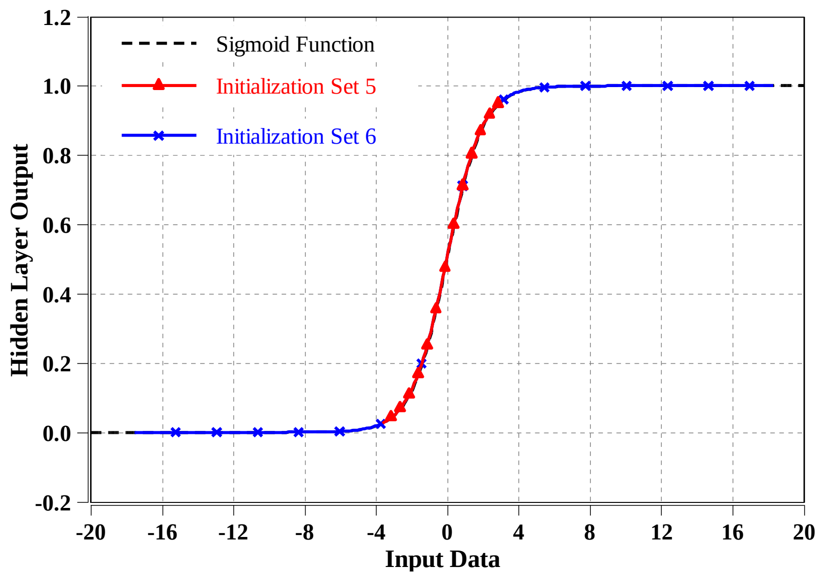


Figure 5-4 Hidden layer output range with different initializations

Based on the above analysis, [-1, 1] was chosen as the initialization intervals for both the weights and biases of the hidden neurons in this research.

5.1.3.2 Number of Hidden Neurons

Inspired by the research in [49], [82]–[84], a random orthogonal projection based enhanced bidirectional extreme learning machine was adopted as the approach to determine the number of hidden neurons in this research. The key steps of the approach are as follows:

1) The OEB-ELM algorithm was executed using the training data set and validate the performance on the validation data set.

2) A sweep from 1 to 85 hidden nodes was conducted. In order to grasp the full trend, no exit condition on the residual error was set for the sweep. When the number of hidden neurons was odd, five sets of random initializations of the weights and biases matrices were generated. Then the matrices were orthogonalized using the Gram-Schmidt orthogonalization method.
3) The residual error between the estimated torque and the observed torque was calculated and stored temporarily for each number of hidden nodes. When the number of hidden neurons was odd, only the set of the orthogonalized matrices that delivered the minimum residue error were retained as the initialization parameters for that node.
4) Totally 1000 such sweeps (repeat Step 2 and Step 3) were conducted to make sure the results would be in line with the true population values according to the Law of Large Numbers (LLN).
5) The average return of the sweeps was calculated. The normalized results of the training data set and validation data set were plotted.

In this research, the number of the hidden nodes was selected as the smallest number that could reduce the residue error by 90% (e.g., normalized RMSE = 0.1), which was 36 nodes in this case as shown in Figure 5-2. Compared with the traditional method, this approach was more objective. The process could be automated as well.

5.2 Weighted Regression Approach and Results

5.2.1 Data Preparation

The performance of the above-established model can be tuned to deliver even higher estimation accuracy at certain data points by accepting slightly bigger errors at other data points. This feature is desirable when the significance of the data points varies, or when the distribution of the samples is imbalanced across the whole data set. For instance, as pointed out in Section 3.2.4, the LMSO points in the engine operating map might deserve higher accuracy as they were the most often run points during the standard vehicle and fuel economy test schedules, such as UDDS and HWFET. To demonstrate how the weighted approach worked, five representative LMSO points ($LMSO_i, i = 1,2, \dots ,5$) were selected as the central points, around which five special areas of interest (AOI) were defined. In this research, the representative LMSO points were selected as 1000rpm@15%load, 1500rpm@30%load, 1500rpm@50%load, 2000rpm@40%load, and 2000rpm@60%load. The size of each AOI was ±50 rpm and ±4% load. The five representative points and their respective AOIs are plotted in Figure 5-5. For convenience, the remainder of the paper addressed all the points in a single AOI by the representative point. Each representative $LMSO_i$ could be assigned a unique weight factor $\widetilde{w}_i$ ($\widetilde{w}_i > 1$). Then, a square diagonal weight matrix could be defined as $W = diag(w_1, w_2, \dots , w_N)$, where N was the number of the training data points. In such a way, each training data point DP_t would have a corresponding adjustable weight w_t at the t_{th} diagonal position of matrix W, where $t = 1,2, \dots , N$. If a DP_t fell in the AOI of an $LMSO_i$, then the t_{th} diagonal weight element of matrix W was set as $w_t = \widetilde{w}_i$. Otherwise, $w_t = 1$, which meant no accuracy enhancement was applied to that data point.

In this manner, it was possible to customize the weight factor for each data point and best adapt this approach to the specific needs of different applications. To simplify the demonstration, the same weight factor was applied to all the representative LMSO point, which meant $\widetilde{w}_1 = \widetilde{w}_2 = \widetilde{w}_3 = \widetilde{w}_4 = \widetilde{w}_5$. Once the weight matrix W was determined, the torque regression model derived from Section 5.1 could be re-trained with the weight matrix to deliver more accurate estimations for the points in the AOIs.

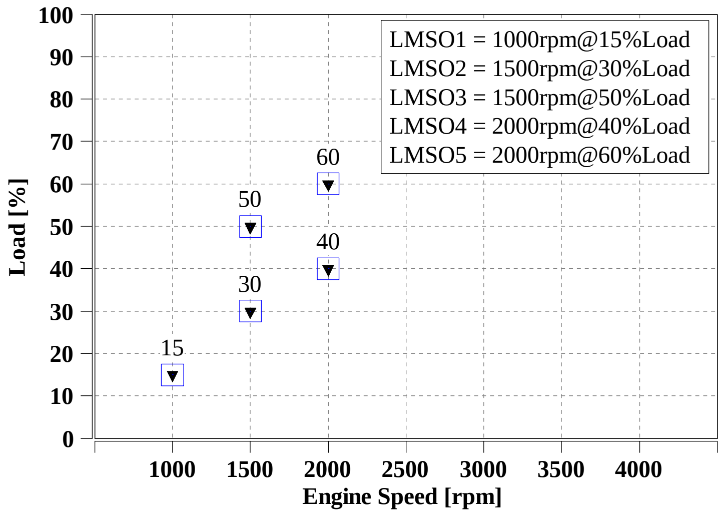


Figure 5-5 Selected LMSO points in a torque and engine speed map

In order to compare the performance between the weighted regression and the non-weighted regression, the same training, validation, and testing data sets as specified in Section 4.4 were reused in this experiment. In addition, the same neuron initialization and regulation factor specified in Section 5.1.2 were also reused. The evaluation metrics were the RMSE and correlation coefficient of the LMSO points before and after applying the weight matrix. The number of LMSO points and the non-LMSO points in the training, validation, and testing data sets are listed in Table 5-3.

Table 5-3 The composition of the training and the testing data sets

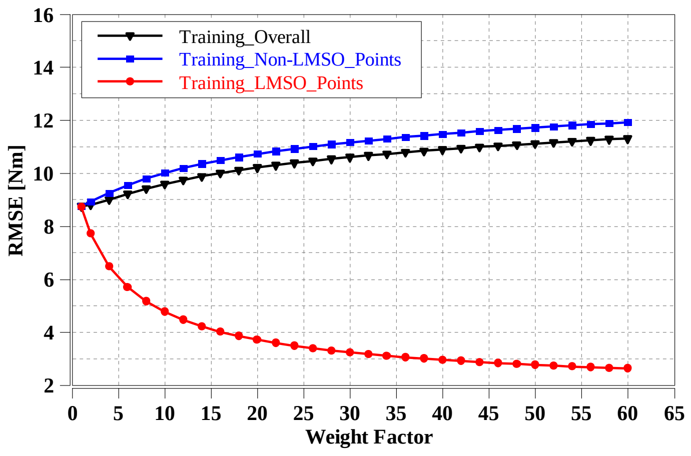
	Training Data Set	Validation Data Set	Testing Data Set
Total Number	2685	335	336
LMSO Points	285	37	34
Non-LMSO Points	2400	298	302

5.2.2 Weighted Regression Results

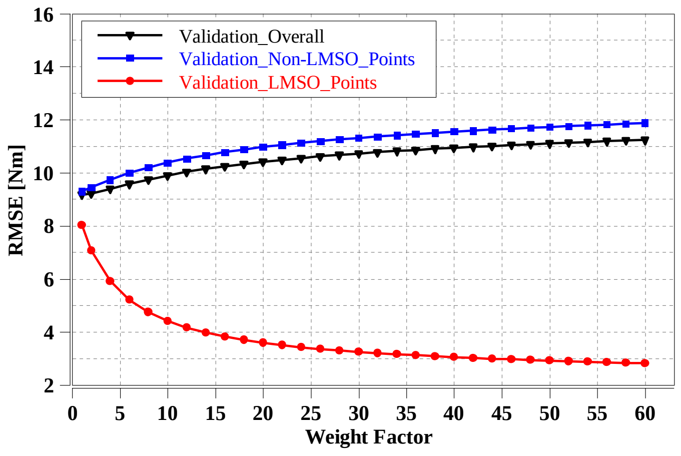
In order to grasp the trend of how the weight factor would affect RMSE and the correlation coefficient, sweeps of the weight factor were firstly conducted on the training, validation, and testing data sets. The results are shown in Figure 5-6 (a), (b), and (c), respectively. The red-dotted line represents the RMSE of the LMSO points after applying the weight factor; the blue-squared line represents the RMSE of the non-LMSO points after applying the weight factor; the black-triangled line represents the overall RMSE of all the data points.

As can be seen in figures, the RMSE of the LMSO points was reduced notably as the value of weight factor increased, which indicated that the model prediction accuracy at these points has been improved greatly. On the other hand, the bigger weight factor had also caused greater RMSE on the non-LMSO points as well as the overall data set. However, the accuracy gain on the LMSO points outweighed the accuracy loss on the non-LMSO points significantly in this experiment. For example, when the weight factor was 60, the RMSE of the LMSO points could be reduced by about 6 Nm, while the RMSE of the other points only increased by less than 3 Nm. In addition, the same data trend was observed across the training, validation, and testing data sets, which indicated the model had good generalization ability across different data sets.

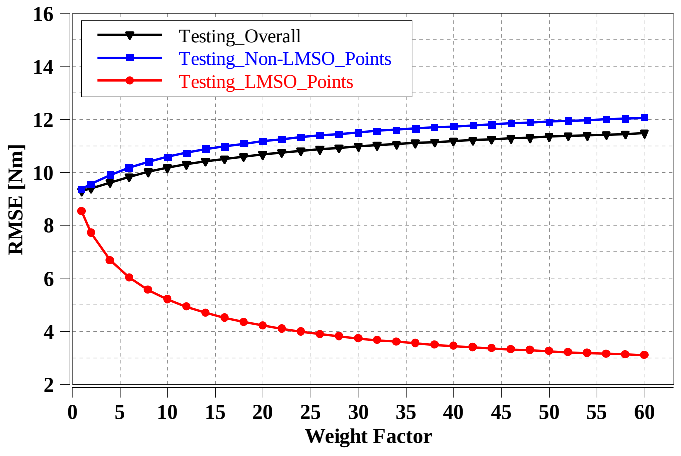
Similar patterns were observed on the correlation coefficient as well. The results are shown in Figure 5-7 (a), (b), and (c), respectively. The red-dotted line represents the correlation coefficient of the LMSO points; the blue-squared line represents the correlation coefficient of the non-LMSO points; the black-triangled line represents the overall RMSE of all the data points. It was observed that significant rises of the correlation coefficients were exhibited on the LMSO points in training, validation, and testing data sets as the weight factor increased, The rise of correlation coefficient indicates that the observed data and predicted data are increasingly related. In the meantime, the correlation coefficients of the non-LMSO points and the overall data set only suffered from very limited drops. In this experiment, the gain of the correlation coefficient was about 0.003, whereas the loss was less than 0.001 for both data sets when the weight factor reached 60.



(a) Training data set

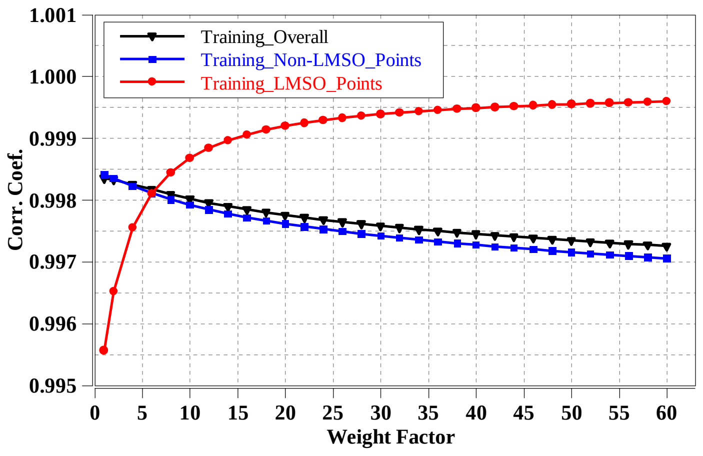


(b) Validation data set

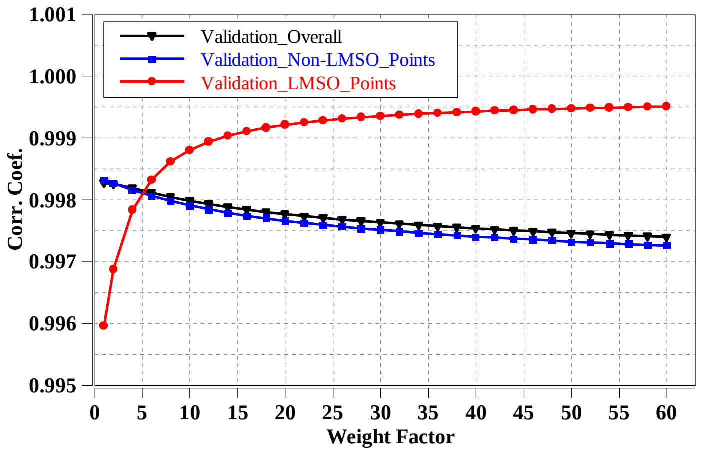


(c) Testing data set

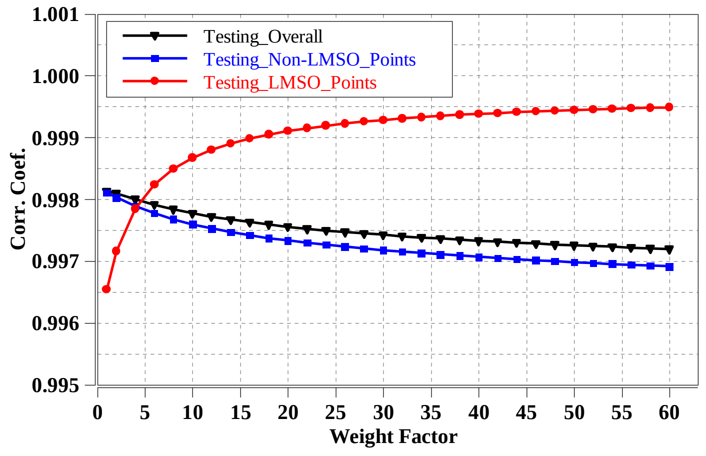
Figure 5-6 Impact of the change of weight factor on RMSE



(a) Training data set



(b) Validation data set

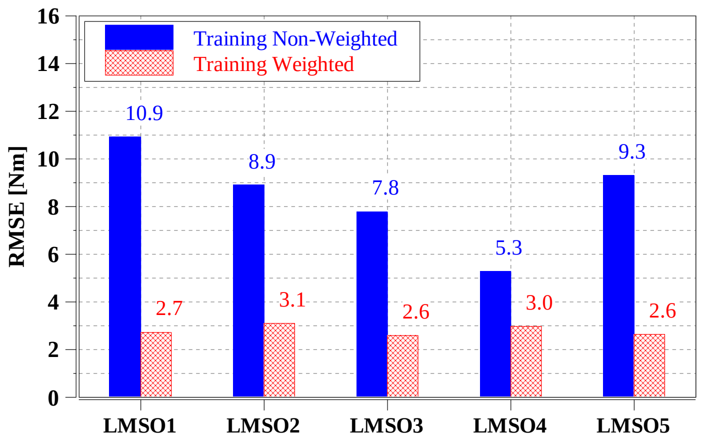


(c) Testing data set

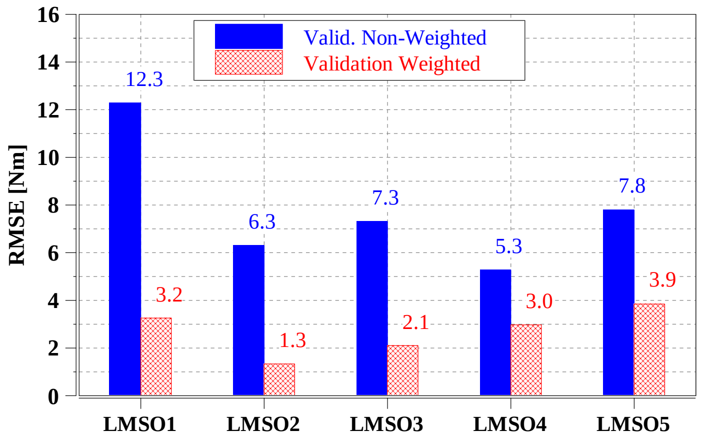
Figure 5-7 Impact of the change of weight factor on the correlation coefficients

When $w = 1$ (which meant no weight was applied), the overall RMSEs were 8.75 Nm, 9.18 Nm, and 9.29 Nm for the training, validation, and testing data sets, respectively; the overall correlation coefficients were 0.9983, 0.9983, and 0.9981 for the training, validation, and testing data sets, respectively. These numbers matched the results of the torque regression model demonstrated in Section 5.1.2, which indicated that the non-weighted approach was a special case of the weighted approach with $w = 1$. The weighted and non-weighted approaches could essentially be unified as one general approach. Another interesting observation was that when w continued to grow, the marginal benefits, such as the decreasing RMSE and the increasing correlation coefficient on the LMSO points, started to diminish. On the other hand, the side effects, such as the rising RMSE and the decreasing correlation coefficient on the non-LMSO points, kept expanding. This trait revealed that the optimal weight w should be determined based on the trade-off between the two aspects according to the specific applications.

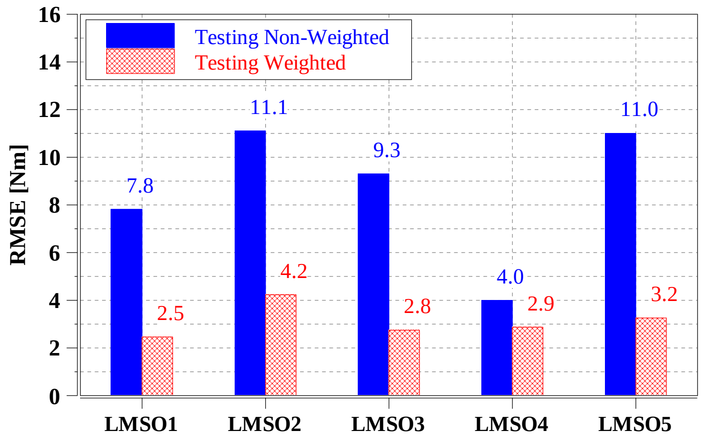
In addition to considering all the LMSO points as a whole, it was worth exploring how much improvement was achieved in each LMSO area with the introduction of the weight factor. In order to do so, comparisons of the same LMSO areas before and after applying the weight factor were also conducted in this research. Note that in Figure 5-6 and Figure 5-7 when the weight factor reached about 30, the benefit margins of increasing weight factor, such as reduced RMSE and increased correlation coefficient, started to dwindle. As a result, $w = 30$ was chosen in this example. As demonstrated in Figure 5-8, the introduction of the weight factor (e.g., $w = 30$) could reduce the RMSE by approximately 2~8 Nm in all the LMSO areas across the training, validation, and testing data sets, though the reduction quantity might vary from one area to another. Similar patterns could also be observed on the correlation coefficients as shown in Figure 5-9. The correlation coefficients of all the LMSO areas increased after applying the weight factor. In general, about 0.1~0.2 increase was observed. The extent of enhancement could be different among the training, the validation, and the testing data sets. An increase in correlation coefficient was observed where a bigger RMSE reduction occurred, which indicated that the introduction of the weight factor did enhance the regression accuracy of that LMSO area.



(a) Training data set

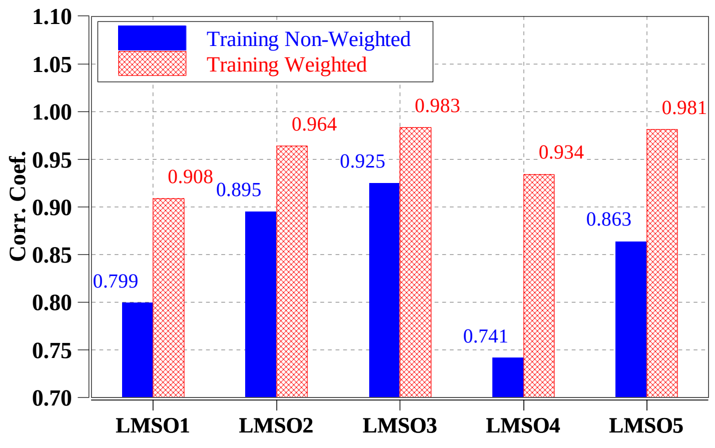


(b) Validation data set

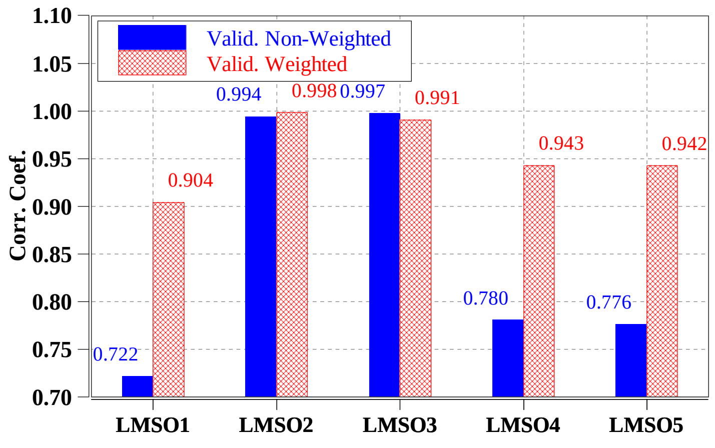


(c) Testing data set

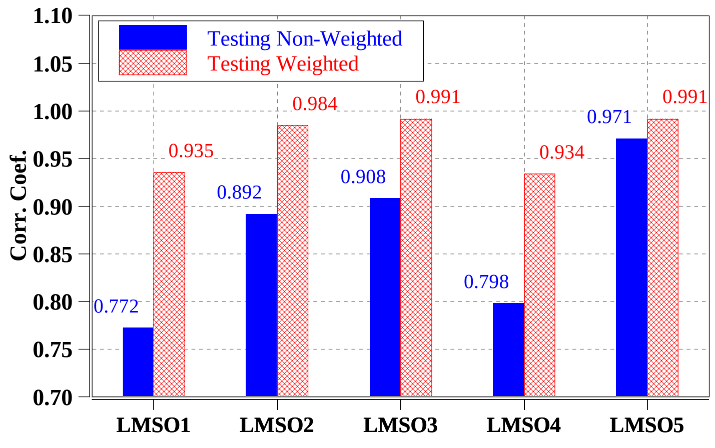
Figure 5-8 RMSE comparison of the individual LMSO areas with w = 30



(a) Training data set



(b) Validation data set

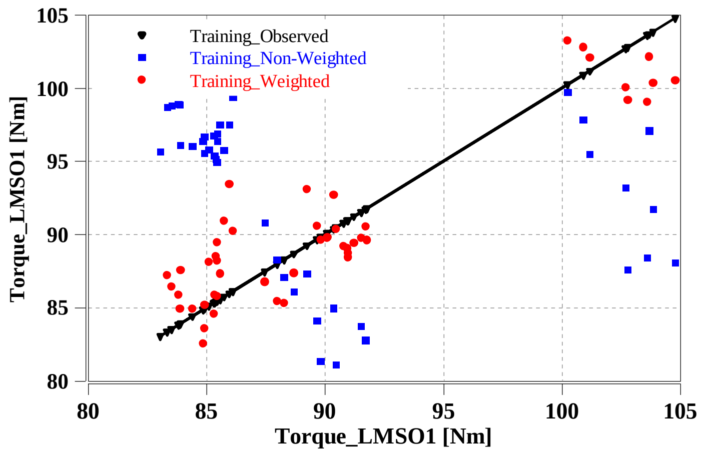


(c) Testing data set

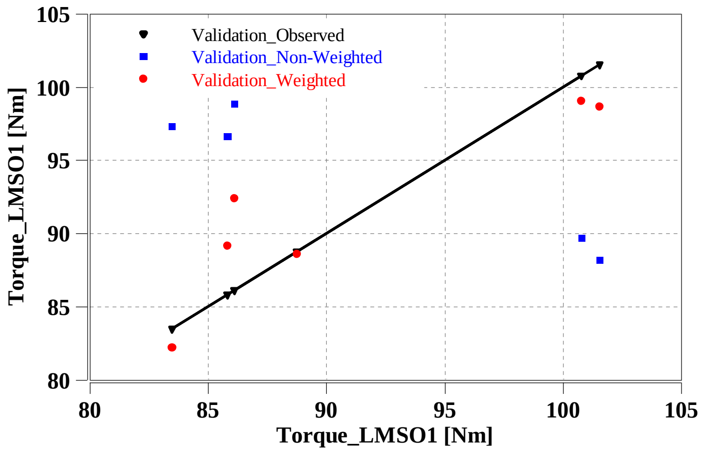
Figure 5-9 Corr. Coef. comparison of the individual LMSO areas with w = 30

These observations provided very helpful information to improve the regression model. Firstly, the inconsistency of enhancement among different LMSO areas indicated that a unified weight factor did not suit all the data regions. The quantitative difference could help determine better-customized weight factors for the individual data regions. Secondly, the inconsistent behavior among the training, validation, and testing data sets implied that the data partition might be imbalanced across the data sets. Therefore, the feature learned in one data set might not suit other data sets in the same way. The quantitative difference indicated the distribution discrepancy, which would help adjust the use of the available data to deliver a better generalization of the model.

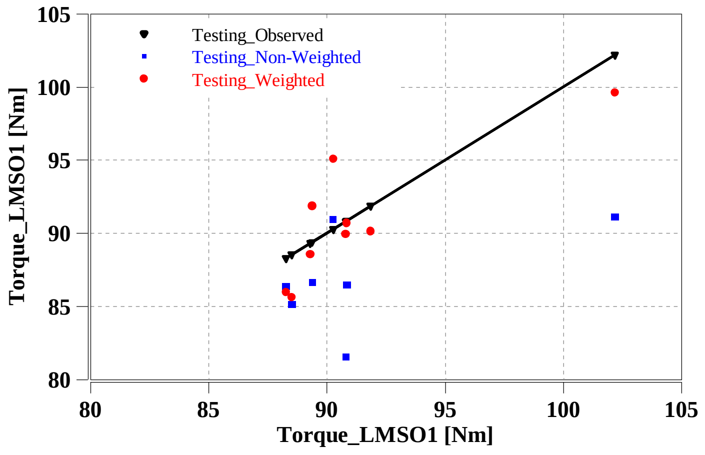
In order to visualize the enhancement brought by the weight factor on the individual data points, the comparison of the observed data, non-weighted regression data, and the weighted regression data are illustrated in Figure 5-10. For the sake of convenience, only LMSO1 points were presented. As can be seen in the figure, the weighted regression points were closer to the observed points than the non-weighted regression points (e.g., the weighted regression data were closer to the 45-degree line than the non-weighted regression points). This matched the finding in the above section that the weighted regression data had a smaller RMSE and higher correlation coefficient than the non-weighted regression data.



(a) Training data set



(b) Validation data set



(c) Testing data set

Figure 5-10 Comparison of regression results of the LMSO1 points with w = 30

5.3 Summary

This chapter presented an ANN approach to predict gasoline engine output torque with high accuracy using an ELM based single-hidden layer feedforward neural network. Ten-fold cross-validation was adopted to stabilize the model. Totally 3356 real-world engine experiment points were collected for this model, in which 2685 and 335 experiment points were used to train and validate the model, respectively. The remaining 336 collected data points were kept unseen from the model to provide unbiased tests of the model. The results demonstrated that the model could predict the engine torque with high accuracy. The RMSE between the observed torque and the predicted torque was about 9 Nm, which was about 2.7% of the mean torque value. In addition, this research also proposed a weight factor approach to further enhance the model accuracy in the designated data regions. The experiment showed that the RMSE in these regions could be further reduced by approximately 6 Nm. Moreover, by exploring the quantitative improvement in the desired regions individually, this research also revealed that optimal overall improvement could be achieved by customizing the weight factor for each desired region accordingly.

Chapter 6. Progressive Extreme Learning Machine

In this chapter, the popular ELM-based algorithms are reviewed. The principles of each algorithm are briefly explained and their limitations are also pointed out. To improve the accuracy of the traditional ELM, a new ELM-based algorithm is proposed. The same experimental data collected in Chapter 4 is used to train and evaluate the proposed algorithm for IC engine torque model. The performance comparison between the proposed algorithm and the traditional ELM is also presented.

6.1 Review of the ELM-Based Techniques

Since its introduction in the 2000s, ELM has attracted attention from the machine learning community and gained increasing popularity among researchers, thanks to many of its advantages, such as compact network structure, fast learning speed, and competitive accuracy. One exceptional characteristic of ELM is its random weight generation feature (also called random feature mapping), which not only reduces the network training time tremendously but also benefits the generalization of the algorithm ([33], [60]). However, by adopting this feature, the potential to improve network accuracy through tuning the weight of the hidden neuron layer is also limited. The traditional ELM algorithm approximates the target function by only calculating the weight between the hidden layer and the output layer to minimize the sum of squared estimation error. Depending on the specific data sets, if the number of hidden nodes is far less than the number of available data points, the traditional ELM may not generate a result as good as that of the gradient-descent approaches.

In the past years, many variants of ELM had been proposed to improve its performance, particularly its accuracy. A majority of them were realized through either incremental or pruning approaches to analytically determine the weights and biases of the hidden neuron layer, or the weights between the hidden neuron layer and the output layer.

For instance, incremental ELM (I-ELM [49]) was proposed to approximate the target function by gradually adding random hidden neurons to the network while analytically calculating the weights between the newly added neurons and the output nodes. Enhanced random search based incremental ELM (EI-ELM [86]) was proposed to improve the quality of the random hidden neurons by generating multiple sets of hidden nodes at each learning step and picking the one that leads to the smallest residue error. Therefore, this approach tended to have better regression accuracy than I-ELM. To improve the convergence rate of I-ELM, convex incremental ELM (CI-ELM [87]) was proposed by utilizing the convex optimization concept and allowing the weights of the existing hidden nodes to be adjusted when a new node was added to the network. Enhanced convex incremental ELM (ECI-ELM [88]) further improved the generalization and convergence speed of CI-ELM by adopting a random search method on the introduced hidden neuron.

Bidirectional ELM (B-ELM [83]) was another type of incremental network. B-ELM did not only analytically decide the weight between the hidden layer and the output layer, but also propagated the error to determine the proper weights and biases of the hidden layer. In addition, unlike I-ELM and its variants, B-ELM only had to analytically determine the weights of the even number of neurons while still adopting random weight generation for the odd number of neurons. It was proved in theory that B-ELM had a faster learning speed and potentially more compact network size than I-ELM. To further improve B-ELM, enhanced bidirectional ELM (EB-ELM [84]) and random orthogonal projection based enhanced bidirectional ELM (OEB-EM [82]) were proposed consecutively. EB-ELM enhanced the quality of random parameter generalization by proposing a few candidates during the initialization steps, then choosing the best one that leads to maximum error reduction. OEB-ELM further optimized the convergence speed of EB-ELM by adopting the improved enhanced random search and orthogonal projection method. In addition, it was able to integrate B-ELM and EB-ELM into a unified form as its non-orthogonal variants.

In order to exploit the benefit of adjusting the weights of the existing nodes (which were usually unchanged in many incremental ELMs), ELM with adaptive growth of hidden nodes (AG-ELM [89]) was proposed. It utilized a method to adaptively evaluate the weight matrix whenever a new hidden node was introduced. Based on the analysis, the existing networks might be replaced by an updated one with better performance and generalization ability. Error minimized ELM (EM-ELM [90]) allowed one or multiple hidden nodes to be added at each incremental step, so the output weights could be incrementally updated. Through such arrangements, the computational complexity of EM-ELM was significantly lower than the traditional ELM.

In contrast to the incremental approaches, where the hidden nodes were incrementally added to the network, the pruning approaches worked by removing the irrelevant nodes to enhance the performance of the network. For instance, pruned ELM (P-ELM [91]) started optimizing with a large number of hidden nodes, then calculating the relevance of each hidden node in regards to its contribution to the reduction of the overall residual error. Based on the results, it could trim the less relevant neurons to reduce the network size and increase the overall estimation accuracy. Optimally pruned ELM (OP-ELM [92]) was an upgrade of P-ELM. It used multi-response sparse regression algorithm to sort the neurons by their usefulness, then utilized the leave-one-out validation approach to prune the irrelevant neurons. The experimental results had shown that OP-ELM could achieve a similar level of accuracy as SVM within a very short computational time. Though the pruning approaches can achieve a more concise network structure, they usually have two typical problems. Firstly, the output accuracy highly depends on the settings of the optimization threshold, according to which the less desirable nodes are pruned. Secondly, the training time may be significantly longer than the traditional ELM. It is also strongly related to the initial size of the network from which the algorithm starts to trim the irrelevant nodes.

In addition to the above-mentioned variants, there are other advancements of ELM that are noteworthy. ELM with kernels [61] utilized a kernel function to abstract the hidden layer activation function so that the activation function could be unknown to the users. Though such a method appeared to generate much better regression accuracy than the traditional ELM, it also used a much larger feature mapping size. The author found out that by increasing the number of hidden neurons of the traditional ELM to make it have the same feature mapping size as ELM with kernels, the traditional ELM may also achieve comparable accuracy as the ELM with kernels. Additionally, the selection of different kernel functions may also lead to significantly different network performance. By analyzing the universal approximation capabilities of ELM, dynamic ELM (D-ELM) was proposed in [93], in which Zhang *et al*. proved that D-ELM could approximate any Lebesgue p-integrable function provided the activation function was Lebesgue p-integrable. To deal with a large amount of data, parallel ELM (PELM [94]) was proposed. It used parallel programming techniques to significantly reduce the computational time of ELM while still maintaining the regression accuracy. In case the input data were only available in sequence, online sequential ELM (OS-ELM [95]) provided the means to process the training data one-by-one or chunk-by-chunk (with either fixed or varying chunk sizes). When a new data point or data chunk arrives, OS-ELM only had to calculate a part of the hidden layer output matrix and update the network output weight according to the recursive least square solution [96]. Self-adaptive evolutionary ELM (SaE-ELM [97]) was proposed to address the limitations of its predecessor - evolutionary ELM (E-ELM [98]). In E-ELM, the mutation strategy was manually selected. However, different vector generation strategies might have highly different performance. In contrast, SaE-ELM could adaptively determine the network parameters and deliver a more proper differential evolution strategy.

As deep learning (DL) has attracted increasing attention due to its excellent performance in pattern recognition and computer vision applications, ELM's potential for DL was also exploited to improve the learning efficiency of DL. For instance, Kasun *et al.* proposed ELM auto-encoder (ELM-AE [99]) that could map features based on the singular values in an unsupervised manner. By integrating it with multilayer ELM (ML-ELM [73]) in the same fashion as the deep neural networks, the overall network had a similar performance as the deep Boltzman machine. To regulate the data manifold of the autoencoder, a generalized ELM autoencoder (GELM-AE [100]) was proposed. By adding a regularization term to the objective function, GELM-AE forced the output of similar samples to be close in the mapped space, so more proper features could be extracted for clustering. As pointed out by Tang *et al.*, due to the shallow structure of ELM, it might not be effective for the natural signals, such as images and videos. To deal with this issue, a hierarchical ELM (H-ELM) was proposed in [60]. It was constructed in a multilayer manner and trained with ELM-based algorithm. The experimental results showed that H-ELM could achieve high-level representation and outperform the accuracy of the traditional ELM.

6.2 Proposed Methodology

It can be seen from the aforementioned studies that, though the ELM-based algorithms have unparalleled training speed and good generalization compared with the BP-based algorithms, they still have difficulties in reaching comparable regression accuracy for certain applications, especially when the number of hidden nodes is far less than the number of available data points. Inspired by the work done by Tang *et al.* in [60], a recursive ELM-based learning framework (referred to as Pr-ELM) was proposed in this thesis to improve the accuracy of ELM. This research only focused on system identification applications. In the proposed algorithm, the estimated results were used to expand the input data set to further tune the model recursively, until the predetermined error threshold or the maximum number of recursions was reached. As a result, the estimated result would approach the expected values progressively.

The proposed approach differs from the incremental cascade ELM [101] by the fact that the number of hidden nodes in Pr-ELM is fixed. It also differs from the research in [102], [103] by the fact that Pr-ELM uses the result of the output layer to enhance the model input data. To verify the performance of the proposed algorithm, a gasoline engine torque model was created using the same data collected in the experiment described in Section 4.3. The results are compared with the model presented in Section 5.1.

The novelties of the proposed algorithm are as follows:

1) It outperforms the conventional ELM in terms of accuracy (RMSE) in system modeling applications.
2) It retains the advantages of the conventional ELM in terms of algorithm simplicity and generalization ability.
3) It supports automatic tuning as no human intervention is needed to find the optimal parameters.
4) It preserves the potential to work with other ELM algorithms, such as B-ELM and EM-ELM, to further enhance the overall accuracy.

6.2.1 Structure of the Proposed Algorithm

The structure of the proposed algorithm is shown in Figure 6-1. Given a training data set with arbitrary distinct samples (x_i, y_i), where $i = 1,2,\dots,N$, the target accuracy η and the maximum number of recursions λ, Pr-ELM is firstly initialized as the conventional ELM with randomly generated input weights w and biases b, as shown in (6-1).

$$Y = g(wX + b)\beta \tag{6-1}$$

where

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mN} \end{bmatrix} \tag{6-2}$$

$$w = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ w_{\tilde{N}1} & w_{\tilde{N}2} & \cdots & w_{\tilde{N}m} \end{bmatrix} \tag{6-3}$$

$$b = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1N} \\ b_{21} & b_{22} & \cdots & b_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ b_{\tilde{N}1} & b_{\tilde{N}2} & \cdots & b_{\tilde{N}N} \end{bmatrix} \tag{6-4}$$

m is the dimension of the input data, k is the dimension of the output data, $\tilde{N}$ is the number of hidden neurons, N is the number of data samples, g is the activation function of the hidden nodes, and $g(\cdot) = H$ is the output of the hidden layer.

Similar to the traditional ELM, the weights β between the hidden neural nodes and the output neural nodes is calculated through Moore-Penrose pseudo inverse as shown in (3-10). The hidden layer maps the m x N dimensional input data onto the $\tilde{N}$ x N dimensional space.

Unlike the traditional ELM, if the output error (measured by RMSE) is greater than the predefined value η, or the recursion is less than the predefined value λ, the output matrix will be appended to the input matrix, expanding its dimension to $(m + k)$ x N. As the number of hidden nodes is not changed, the hidden layer is still mapping the input data to an $\tilde{N}$ x N dimensional space. To maintain the simplicity and generalization ability of the conventional ELM, the input weights $w^*$and biases $b^*$of the newly formed input matrix are still randomly generated. The mathematical expressions of the expanded input matrix, the new input weights, and the new biases are shown in (6-5), (6-6), and (6-7), respectively. The weights between the hidden neural nodes and the output neural nodes, β^*, is calculated according to (3-10) as in the conventional ELM.

$$X^* = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mN} \\ y_{11} & y_{12} & \cdots & y_{1N} \\ \vdots & \vdots & \vdots & \vdots \\ y_{k1} & y_{k2} & \cdots & y_{kN} \end{bmatrix} \tag{6-5}$$

$$w^* = \begin{bmatrix} w_{11}^* & w_{12}^* & \cdots & w_{1m}^* & w_{1,m+1}^* & \cdots & w_{1m+k}^* \\ w_{21}^* & w_{22}^* & \cdots & w_{2m}^* & w_{2,m+1}^* & \cdots & w_{2,m+k}^* \\ \vdots & \vdots & & \vdots & \vdots & \cdots & \\ w_{\tilde{N}1}^* & w_{\tilde{N}2}^* & \cdots & w_{\tilde{N}m}^* & w_{\tilde{N},m+1}^* & \cdots & w_{\tilde{N},m+k}^* \end{bmatrix} \tag{6-6}$$

$$b^* = \begin{bmatrix} b_{11}^* & b_{12}^* & \cdots & b_{1N}^* \\ b_{21}^* & b_{22}^* & \cdots & b_{2N}^* \\ \vdots & \vdots & \vdots & \vdots \\ b_{\tilde{N}1}^* & b_{\tilde{N}2}^* & \cdots & b_{\tilde{N}N}^* \end{bmatrix} \tag{6-7}$$

The error of the newly derived output data will be re-evaluated. If it is still greater than the predefined value η or the iteration is less than the predefined value λ, such iteration will continue until the exit condition is fulfilled. As a result, the estimated output data shall progressively approach the expected values until saturation.

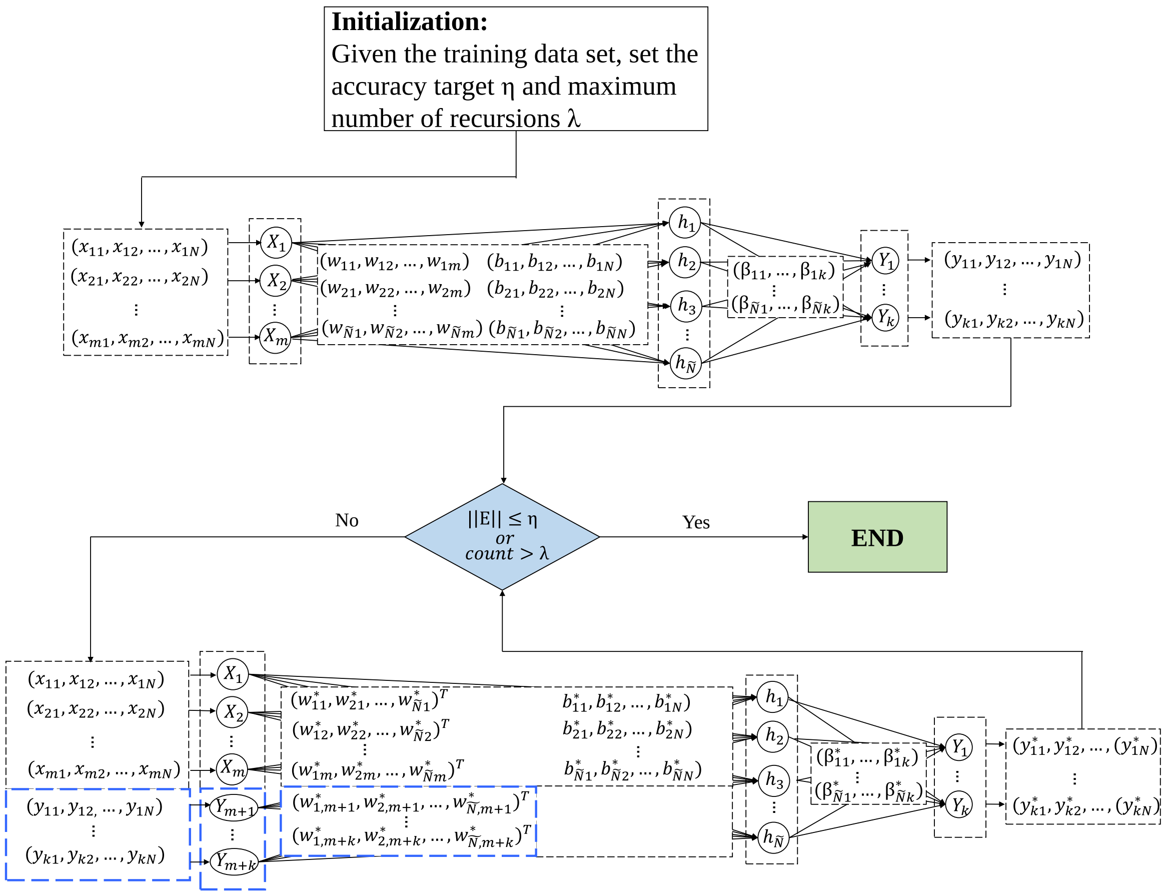

Initialization:
Given the training data set, set the accuracy target η and maximum number of recursions λ
No
||E|| ≤ η or count > λ
Yes
END

Figure 6-1 Structure of the proposed algorithm

6.2.2 Pseudo-code of the Pr-ELM Algorithm

The pseudo-code of the proposed Pr-ELM algorithm is summarized in the ***Algorithm***.

Algorithm Progressive ELM Algorithm

For N arbitrary distinct samples (x_i, y_i), where $x_i = [x_{i1}, x_{i2}, \dots, x_{im}]^T \in R^m$ and $y_i = [y_{i1}, y_{i2}, \dots, y_{ik}]^T \in R^k$, consider the standard ELM approximation with $\tilde{N}$ hidden nodes,

```
Step 1) **Initialization:** Set the acceptable RMSE threshold as $\eta$ and the maximum number of iterations as $\lambda$.
Step 2) **Training the network:**
o  Normalize the data samples to [0, 1], where 0 corresponds to 95% of their minimum values and 1 corresponds to 105% of their maximum values, respectively.
o  Randomly generate the weight of the neuron $w_{\tilde{N}\times m}$ and bias $b_{\tilde{N}\times N}$, and calculate the hidden node output matrix $H_{\tilde{N}\times N}$. It should be assured that the $H_{\tilde{N}\times N}$ does not fall into the periodic or asymptotic range of the given activation function. Otherwise, the normalization range of the data samples may need to be adjusted.
o  Calculate the weight vector $\beta_{\tilde{N}\times k}$ and find out the estimated model output $\hat{Y}$
**while**    RMSE of $(\hat{Y}, Y) \geq \eta$ or the number of iterations $\leq \lambda$    **do**
    o  Append the estimated $\hat{Y}$ to the training data set to form a new training data set as $[X\ \hat{Y}]$.
    o  Randomly generate the weights and biases according to the size of the input data. Then compute the hidden node output matrix $H^*_{\tilde{N}\times N}$.
    o  Calculate the weight vector $\beta^*_{\tilde{N}\times k}$ and the estimated model output $\widehat{Y^*}$
**end while**
o  Restore the model output to the original scale
```

6.3 Data Preparation and Model Evaluation

In this section, a gasoline engine torque model was created with the proposed algorithm. The model was trained and tested with data collected from the engine mapping experiment described in Chapter 4. The evaluation metrics were RMSE and correlation coefficient. The performance difference between the proposed model and the traditional ELM model was highlighted.

6.3.1 Data Preparation

In order to minimize the effect of data discrepancies, the collected experimental data were randomly partitioned into the training data set and the testing data set, at a ratio of 8:2 (or very close to 8:2). As a result, there were 2685 data points in the training data set and 671 data points in the testing data sets. The model input parameters were the ones that might directly affect the engine combustion process, such as VCT and spark advance angle. The parameters that had less impact on the engine combustion process, such as the barometric pressure and exhaust gas temperature, were also included in the model input. A list of the model input/output (I/O) parameters and their statistics in the training and the testing data sets are shown in Table 6-1.

Table 6-1 Model input/output parameters and statistics

Model Parameters	I/O	Training Data Set			Testing Data Set		
		Min	Mean	Max	Min	Mean	Max
Engine Torque (Nm)	Output	55.6	323.7	671.6	55.3	323.9	670.3
Engine Speed (rpm)	Input	999	2662	4504	999	2602	4504
Intake Manifold Pressure (kPa)	Input	23.8	71.2	100.1	22.3	71.4	100.7
Barometric Pressure (kPa)	Input	98.2	99.7	100.9	98.2	99.8	100.9
Intake Air Temperature (℃)	Input	17.7	24.2	35.5	19.3	24.3	35.3
VCT (deg)	Input	0	25	50	0	24	50
Spark Advance Angle (deg)	Input	0	27	55	0	28	55
Lambda (-)	Input	0.84	0.96	1.0	0.87	0.96	1.0
Exhaust Gas Temperature (℃)	Input	369.2	712.5	937.6	370.8	707.6	927.3
Exhaust Gas Pressure (kPa)	Input	2.1	5.6	11.9	2.1	5.4	11.9

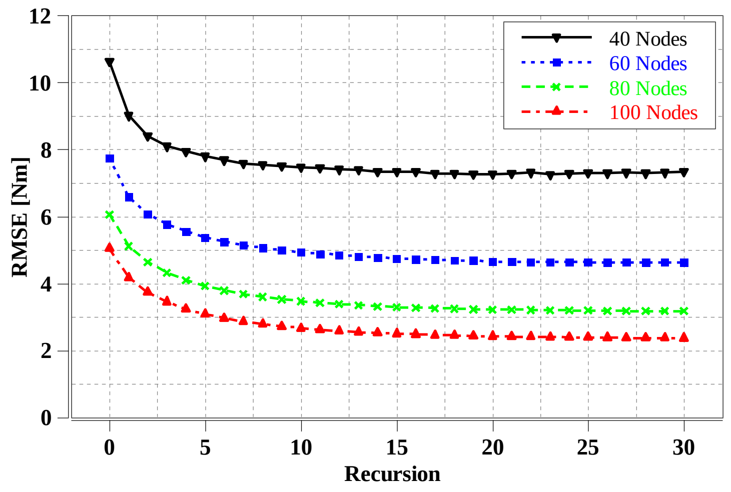
As shown in the table, the model I/O parameters differed a lot numerically. Therefore, data normalization was required to avoid the neural response being dominated by the large numerical parameters. In this experiment, all the input/output parameters were normalized to [0, 1], where 0 corresponded to 95% of their minimum values and 1 corresponded to 105% of their maximum values, respectively. Such normalization also created some room for redundancy in case the estimated output value fell slightly outside of the range of the input value. After neural network computation, the output parameters needed to be restored to their original scales and reviewed in the backdrop of the specific engineering application. The randomization range of the hidden neural weights and biases were both [-1, 1]. Sigmoid function was selected as the activation function for this application. Such initialization settings made the inputs of the activation function reside in the range of [-3.7, 3.1], and the outputs stayed away from the asymptotic range of the activation function output. Therefore, each input value would have a unique output response and the change of input value would also be reflected sufficiently in the neural response.

To reveal the impact of the number of hidden nodes on the model performance, 40, 60, 80, and 100 hidden nodes were used to build the model, respectively. In addition, no RMSE exit condition (as indicated in the ***Algorithm***) was applied to this experiment. The algorithm ran up to 30 recursions to demonstrate the effect of recursions on the model performance. It was noted that the model size of this application was not very big, so the quantization noise was generally not a concern.

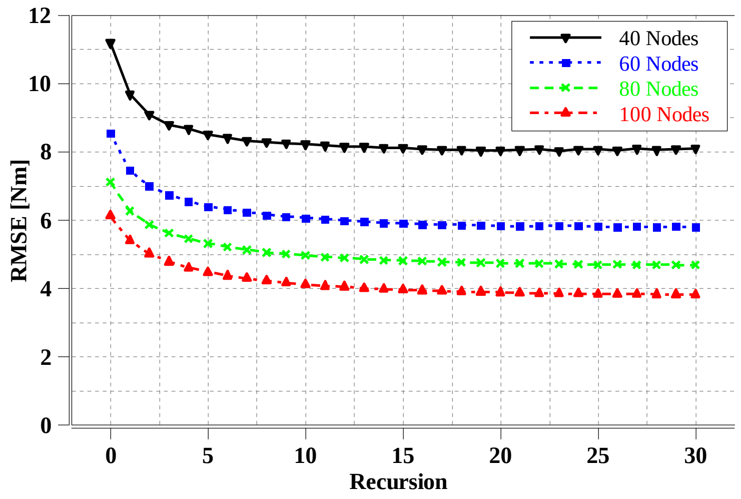
6.3.2 Experimental Evaluation Results

In order to minimize possible disturbances, the average result from 200 repeated execution of the algorithm was reported. The authors had verified that this number of trials was appropriate to make sure that the mean of the trials would represent the mean of the population based on LLN theorem. As can be seen in Figure 6-2, the proposed algorithm reduced the RMSE of the engine torque model output significantly over a wide number of hidden neural nodes. The key observations are summarized as follows.

1) For any given number of hidden nodes, Pr-ELM could gradually reduce the output RMSE with the increased number for recursions.
2) For any given number of recursions, a smaller output RMSE could be achieved by increasing the number of hidden nodes.
3) The reduction in RMSE was mainly achieved within the first 5 recursions. After about 15 recursions, the RMSE reduction began to saturate.
4) Compared with the traditional ELM, which corresponds to the 0 recursion, Pr-ELM could further minimize the RMSE by approximately 30% to 40% for any given number of hidden nodes.
5) Consistent performance was observed between the training and the testing data sets.



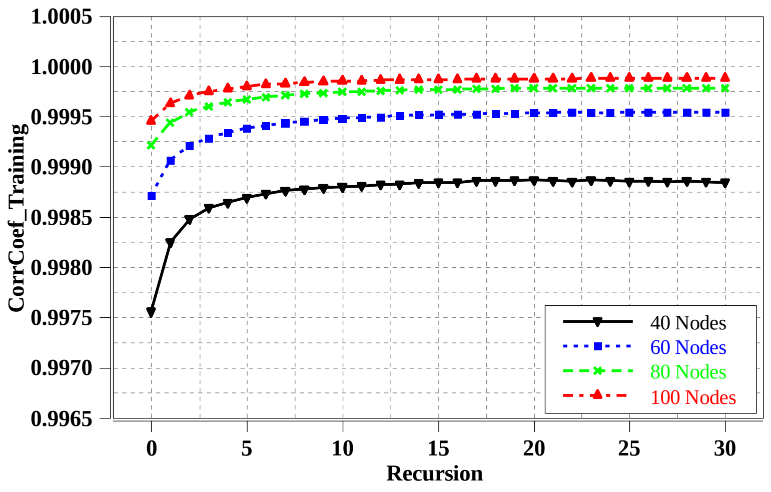
(a) Training data set



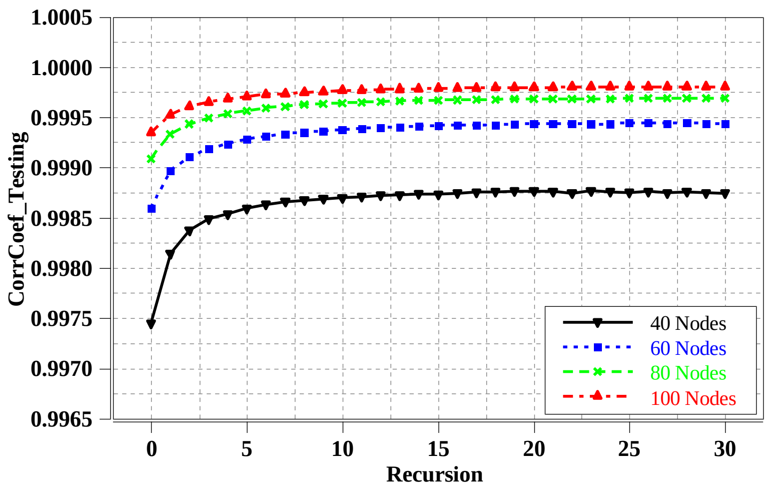
(b) Testing data set

Figure 6-2 Impact of the number of recursions on RMSE

Similar patterns were observed on the model output correlation coefficient. As shown in Figure 6-3, with an increase of recursions, significant rises of the correlation coefficients were exhibited for all the cases with a different number of hidden nodes. In addition, as the recursion continued to grow, the correlation coefficients plateaued, which indicated stable convergence of the algorithm. Moreover, good generalization was also observed between the training and testing data sets.



(a) Training data set



(b) Testing data set

Figure 6-3 Impact of the number of recursions on the correlation coefficient

As observed in the above figures, when the recursion reached 15, the benefit of Pr-ELM began to saturate in this experiment. Therefore, a comparison between the traditional ELM and Pr-ELM (with recursion =15) was made in Table 6-2. As shown in the table, Pr-ELM had substantially better RMSE than the traditional ELM for various numbers of hidden nodes, for both the training and the testing data sets. The average reduction of RMSE of Pr-ELM was about 30% to 40%. Similarly, a significant increase of correlation coefficients was observed on Pr-ELM for all the compared groups.

Table 6-2 Comparison between traditional ELM and Pr-ELM (recursion = 15)

Hidden Nodes		RMSE		Correlation Coefficient	
		Training Data Set	Testing Data Set	Training Data Set	Testing Data Set
40	Traditional ELM	10.6	11.2	0.9976	0.9975
	Pr-ELM	7.3	8.1	0.9988	0.9987
60	Traditional ELM	7.7	8.5	0.9987	0.9986
	Pr-ELM	4.7	5.9	0.9995	0.9994
80	Traditional ELM	6.1	7.1	0.9992	0.9991
	Pr-ELM	3.3	4.8	0.9998	0.9997
100	Traditional ELM	5.0	6.1	0.9995	0.9993
	Pr-ELM	2.5	3.9	0.9999	0.9998

6.4 Summary

This chapter presented an IC engine torque model created with Pr-ELM algorithm. The data acquired from the engine mapping test conducted in Chapter 4 was used to train and test the model. The experimental evaluation results showed that Pr-ELM significantly increased model accuracy in comparison to the traditional ELM approach. Meanwhile, the Pr-ELM still retained the random initialization feature. In this research, Pr-ELM outperformed the traditional ELM by providing 30% to 40% RMSE reduction. The experiment also showed good generalization between the training and testing data sets.

Chapter 7. Comparison of Pr-ELM with Competing Algorithms

In this chapter, the proposed Pr-ELM algorithm was experimentally evaluated and compared with other algorithms. Firstly, the performance of the proposed algorithm was compared with a range of the ELM-based algorithms. The same engine mapping data acquired in Chapter 4 was used to create and evaluate the IC engine torque models. The evaluation metrics were RMSE, correlation coefficient, and the training time. Secondly, the performance of the proposed Pr-ELM algorithm was evaluated against other non-ELM based algorithms, such as Levenberg Marquardt Algorithm (LMA) and Support Vector Regression (SVR) algorithms. In this case, the IC engine mapping data and the data sets from the University of California, Irvine, Machine Learning Repository [104] were used to evaluate the algorithms.

7.1 Comparison with Competing ELM-Based Algorithms

In this section, experimental results are presented that compare the effectiveness of the proposed Pr-ELM against the popular ELM-based algorithms, such as EI-ELM, EB-ELM, EM-ELM, and OP-ELM. These algorithms are the improved versions of their ELM predecessors. They are more efficient than classical ELM and other constructive neural networks [33].

The following measures were taken to ensure that the comparison among different algorithms was carried out under the same conditions.

1) The same real-world engine mapping data acquired from the experiment in Chapter 4 was applied to all the algorithms. The same I/O parameter normalizing scheme as described in Section 5.1 was also reused.
2) The number of hidden nodes was set to be the same in all algorithms. Specifically, 50 and 100 hidden nodes were used in this experiment for comparison purposes.
3) The weights and biases of the hidden neurons were randomly generated between [-1, 1]. The sigmoid function was selected as the activation function for all the algorithms.
4) All the experiments were conducted in Matlab 2017b running on the same Windows 10 machine with a 16-GB RAM and an Intel i7-6700 CPU @ 3.4 GHz processor.
5) Two hundred trials were conducted with each algorithm and the averaged result of each algorithm was reported. The authors had verified that this number of trials was appropriate to reflect the true mean of the population in this application according to the LLN theorem.

In addition, some algorithm-specific settings also need to be highlighted. In this experiment, 20 sets (k=20) of weights and biases were randomly generated at each corresponding increment step of EI-ELM and EB-ELM. The maximum number of neurons allowed was set to 55 and 105 respectively for OP-ELM algorithm. The step length of pruning was set to one. Only the results from the trials that stopped at 50 nodes or 100 nodes were accepted as valid results of OP-ELM. The number of recursions of the Pr-ELM was set to 30 which was in line with the settings in Chapter 6.

The performance of the algorithms was evaluated by using the following metrics: RMSE, the correlation coefficient, and the neural network training time. The experimental results of 50 and 100 hidden nodes are shown in Table 7-1 and Table 7-2, respectively. It can be seen that the proposed Pr-ELM had the lowest RMSE and the highest correlation coefficient among all the listed algorithms for both the training data set and the testing data set.

Specifically, if the number of hidden nodes was 50, the RMSE of Pr-ELM was about 6 Nm, which was approximately 35% less than that of EM-ELM, 40% less than that of OP-ELM, 78% less than that of EB-ELM, and 84% less than that of EI-ELM. Similarly, the correlation coefficient of Pr-ELM was over 0.9992, which was also better than that of all the other competing algorithms. The training time of Pr-ELM was about 0.67 second. Though it was only slightly better than EM-ELM, it was over 6 times faster than that of EB-ELM and OP-ELM, and about 12 times faster than the training time of EI-ELM under the evaluation conditions.

If the number of hidden nodes was 100, the RMSE of Pr-ELM could reach about 3 Nm, which was approximately 50% better than that of EM-ELM, 50% better than that of OP-ELM, 80% better than that of EB-ELM, and more than 90% better than that of EI-ELM under the evaluation conditions. Similarly, the correlation coefficient of Pr-ELM was over 0.9998, which was higher than that of all the other competing algorithms. The training time of Pr-ELM was about 1.6 seconds, which was about 2 times faster than EM-ELM, over 4.5 times faster than EB-ELM and OP-ELM, and about 10 times faster than EI-ELM under the evaluation conditions.

Table 7-1 Performance comparison with 50 hidden nodes

	Training			Testing	
	RMSE (Nm)	Corr. Coef.	Time (s)	RMSE (Nm)	Corr. Coef.
EI-ELM (k=20)	44.8	0.9573	8.11	44.7	0.9576
EB-ELM (k=20)	27.7	0.9832	4.45	27.9	0.9831
EM-ELM	9.0	0.9983	0.70	9.6	0.9981
OP-ELM	9.4	0.9981	4.19	9.9	0.9980
Pr-ELM	5.8	0.9993	0.67	6.5	0.9992

Table 7-2 Performance comparison with 100 hidden nodes

	Training			Testing	
	RMSE (Nm)	Corr. Coef.	Time (s)	RMSE (Nm)	Corr. Coef.
EI-ELM (k=20)	36.5	0.9712	16.4	36.5	0.9712
EB-ELM (k=20)	27.6	0.9835	1.06	27.6	0.9835
EM-ELM	5.0	0.9995	2.64	5.0	0.9995
OP-ELM	5.0	0.9995	10.15	5.0	0.9995
Pr-ELM	2.4	0.9999	1.91	2.4	0.9999

Though the experimental evaluations showed impressive performance of Pr-ELM over the other ELM-based algorithms, it is worth pointing out that the number of hidden nodes was a pre-determined value in Pr-ELM. Unlike the other compared algorithms, Pr-ELM was incapable of determining the network architectures.

7.2 Comparison with LMA and SVR

In this section, the performance of Pr-ELM was further compared with the popular non-linear neural network based system identification algorithms, such as LMA and SVR. Comprehensive experimental results are presented following a brief description of the LMA and SVR algorithms. The IC engine mapping data set (collected in Chapter 4) and the data sets from the University of California, Irvine, Machine Learning Repository [104] were both used to evaluate the algorithms.

7.2.1 Brief description of LMA and SVR algorithms

7.2.1.1 Backpropagation Levenberg-Marquardt Algorithm

Levenberg-Marquardt Algorithm (LMA) is a popular backpropagation-based neural network training algorithm that is usually recommended as the first alternative to the conventional backpropagation algorithm [105]. It combines two minimization methods: the gradient descent method and the Gauss-Newton method [106]–[108]. A brief description of the algorithm is as follows.

The optimization target of the system identification problem is to minimize the error function:

$$f(x) = \frac{1}{2}\sum_{j=1}^{m}(y_j - \hat{y}_j)^2 = \frac{1}{2}\sum_{j=1}^{m} r_j{}^2(x) \qquad (7\text{-}1)$$

where $x = (x_1, x_2, \ldots, x_n)$ is the input data vector, r_i is the residue of each estimation. It is assumed that $m > n$, which means the number of data points is greater than the dimension of input parameters. Set the Jacobian matrix as

$$J(x) = \begin{bmatrix} \frac{\partial r_1(x)}{\partial x_1} & \frac{\partial r_1(x)}{\partial x_2} & \cdots & \frac{\partial r_1(x)}{\partial x_N} \\ \frac{\partial r_2(x)}{\partial x_1} & \frac{\partial r_2(x)}{\partial x_2} & \cdots & \frac{\partial r_2(x)}{\partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r_m(x)}{\partial x_1} & \frac{\partial r_m(x)}{\partial x_2} & \cdots & \frac{\partial r_m(x)}{\partial x_N} \end{bmatrix} \tag{7-2}$$

The first and second order derivative of the error functions are:

$$\nabla f(x) = \sum_{j=1}^{m} r_j(x) \nabla r_j(x) = J(x)^T r(x) \tag{7-3}$$

$$\nabla^2 f(x) = J(x)^T J(x) + \sum_{j=1}^{m} r_j(x) \nabla^2 r_j(x) \tag{7-4}$$

The second order derivative is called Hessian matrix ($\nabla^2 f(x)$). As $\nabla^2 r_j(x)$ or the residual $r_j(x)$ are small, the Hessian matrix can be simplified as showing in ((7-5).

$$H = \nabla^2 f(x) = J(x)^T J(x) \tag{7-5}$$

Therefore, a combination of simple gradient descent and the Gauss-Newton optimization approach can be realized [109], in which the second order derivative is approximated by the operation of Jacobian matrix. In order to avoid the estimation bouncing around the local minima, when the estimated value is far from the target value, the weight descends along the steepest slope direction, which is the traditional backpropagation approach as shown in ((7-6).

$$x_{i+1} = x_i - \lambda \nabla f(x_i) \tag{7-6}$$

When the estimated value is close to the targeted value, the weight descends along the curvature of the slop by solving the equation $\nabla f(x) = 0$. The update rule for the Gauss-Newton method is set up by expanding $\nabla f(x)$ with Taylor Series and omitting the residues over second order, as shown in ((7-7).

$$x_{i+1} = x_i - (\nabla^2 f(x_i))^{-1} \nabla f(x_i) \tag{7-7}$$

As pointed out earlier, the Hessian matrix can be approximated with the Jacobian as shown in ((7-5). The main advantage of this technique is its shorter convergence time. It is complementary to the simple gradient descent algorithm. The unified expression of LMA is shown in ((7-8),

$$x_{i+1} = x_i - (H + \lambda I)^{-1} \nabla f(x_i) \tag{7-8}$$

where H is the Hessian matrix at x_i. The parameter λ is multiplied by a factor (usually 10) when a step leads to an increase of the error function $f(x)$. The parameter λ is divided by the same factor when a step results in a decrease of the error function $f(x)$. It is worth noting that when λ is large, the algorithm becomes the steepest descent algorithm. Whereas when λ is small, the algorithm becomes Gauss-Newton method to reduce the influence of the gradient descent.

7.2.1.2 Support Vector Regression

Support vector regression (SVR) is an extended application of the support vector machine SVM algorithm. SVM works by finding the maximum margin to separate the hyperplane to correctly classify the training data. Whereas the standard SVR uses an ε-sensitive region to characterize the cost function, within the region the estimation error is not penalized [110]. The simple mathematical formulation of SVR is described as follows.

For a linear regression case, given a finite training data set (x_1, y_1), (x_2, y_2), …, (x_N, y_N), where $x_i = [x_{i1}, x_{i2}, \dots, x_{im}]^T \in R^m$ and $y_i = [y_{i1}, y_{i2}, \dots, y_{ip}]^T \in R^p$, the multivariate regression target function is estimated as (7-9),

$$\hat{y}_i = < w, x > + b = \sum_{j=1}^{m} w_j * x_{ij} + b, \qquad i = 1,2, \dots, N \tag{7-9}$$

where N is the number of data points, m and p are the dimensions of the input data and output data, respectively. $\hat{y}_i$ is the estimation of the observed true model output y_i. w is the weight of the input matrix. Minimized norm value of $||w||^2$ is desired in order to flatten the estimation surface as much as possible.

The loss of the estimation is defined as e_i, which is the absolute difference between the estimated and the observed values[4] in this case. The loss is supposed to within the ε-

[4] It is worth noting that linear loss function is used in this example. However, other types of residuals may also be used, such as quadratic loss function or Huber loss function.

sensitive region as shown in (7-10). If the estimation loss of a point is within the **ε** region, there is no penalty for that point. Otherwise, the outlier should be penalized.

$$e_i = |y_i - \hat{y}_i| < \varepsilon \qquad (7\text{-}10)$$

However, the condition may not hold for all the points. In order to solve this issue, slack variables ξ_i and ξ_i^* are added to each data point to "soften" the restriction, as shown in Figure 7-1. Therefore, the cost function of the SVR becomes:

Minimize: $$\mathcal{L} = \frac{1}{2} * \left||w|\right|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \qquad (7\text{-}11)$$

Subject to: $$\begin{cases} y_i - \hat{y}_i \le \varepsilon + \xi_i \\ \hat{y}_i - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* > 0 \end{cases} \qquad (7\text{-}12)$$

where C is a positive box value that penalizes the estimations outside of the **ε** region.

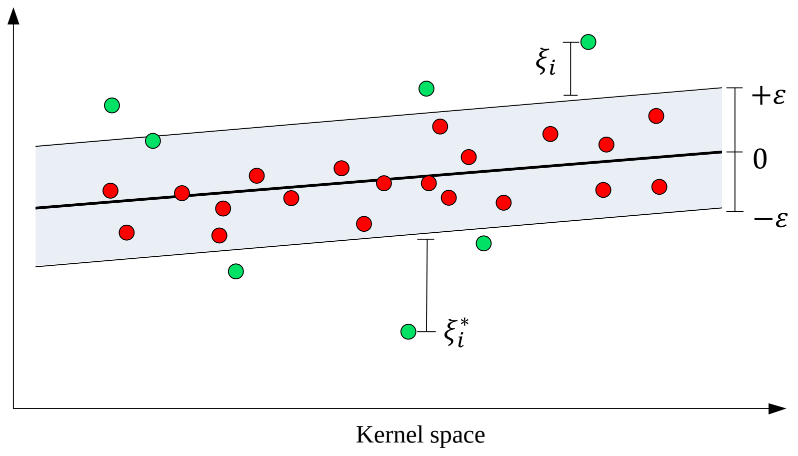


Figure 7-1 Linear ε-insensitive loss function

The optimization problem is usually solved using Lagrange dual formulation. The nonnegative multipliers a_i and a_i^* are introduced at each estimated point.

Minimize: $$\mathcal{L} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(a_i - a_i^*)(a_j - a_j^*)x_i' x_j + \varepsilon\sum_{i=1}^{N}(a_i + a_i^*) + \sum_{i=1}^{N} y_i(a_i^* - a_i) \tag{7-13}$$

Subject to: $$\begin{cases} \sum_{i=1}^{N}(a_i - a_i^*) = 0 \\ 0 \le a_i \le \mathrm{C} \\ 0 \le a_i^* \le \mathrm{C} \end{cases} \tag{7-14}$$

Once all a_i and a_i^* are determined, the weight can be described as a linear combination of the data samples, as shown below:

$$w = \sum_{i=1}^{N}(a_i - a_i^*)x_i \tag{7-15}$$

Hence, the regression function can be constructed as:

$$\hat{y}_i = \sum_{i=1}^{N}(a_i - a_i^*) < x_i, x > +b \tag{7-16}$$

Subject to: $$\begin{cases} a_i(\varepsilon + \xi_i - y_i + wx_i + b) = 0 \\ a_i^*(\varepsilon + \xi_i^* + y_i - wx_i - b) = 0 \\ \xi_i(C - a_i) = 0 \\ \xi_i^*(C - a_i^*) = 0 \end{cases} \tag{7-17}$$

From the above equations, it is observed that w does not need to be calculated explicitly in SVR models as the target function is estimated through the inner products of the input data. In addition, the linear SVR model can also be extended to the non-linear case by replacing the inner product $< x_i', x >$ with a nonlinear kernel function $G(x) =< g(x_i)', g(x) >$. The commonly used kernel function include Gaussian, RBF, and Polynomial functions. As a result, the final estimation function with kernel is formulated as follows [111].

$$\hat{y}_i = \sum_{i=1}^{N} (a_i - a_i^*) < g(x_i), g(x) > +b \tag{7-18}$$

Subject to:
$$\begin{cases} a_i(\varepsilon + \xi_i - y_i + \hat{y}_i) = 0 \\ a_i^*(\varepsilon + \xi_i^* + y_i - \hat{y}_i) = 0 \\ \xi_i(\mathrm{C} - a_i) = 0 \\ \xi_i^*(C - a_i^*) = 0 \end{cases} \tag{7-19}$$

7.2.2 Evaluation Configurations and Results

In this section, the performances of Pr-ELM, LMA, and SVR were compared on the gasoline engine mapping data set (collected in Chapter 4) and eleven benchmark data sets obtained from University of California, Irvine, Machine Learning Repository [104]. The experimental data sets contain a wide variety of applications, data samples, and data attributes. The specification of these data sets is provided in Table 7-3

Table 7-3 Specification of the data sets

Name	Training data	Testing data	Attributes
IC Engine Mapping Data	2685	671	12
Airfoil Self-noise	1202	301	5
Abalone	3342	835	8
Servo Data	134	33	4
Concrete Strength	824	206	8
Combined Cycle Power Plant	7654	1914	4
Energy Efficiency (heating)	614	154	8
Energy Efficiency (cooling)	614	154	8
Wine Quality (Red)	1279	320	11
Wine Quality (White)	3918	980	11
Gas Turbine Emission (CO)	29386	7347	9
Yacht Hydrodynamics	246	62	6

The following measures were taken to preprocess the data sets and configure the modeling parameters.

1) The same data partition and I/O parameters normalizing scheme were applied to all the data sets as described in Chapter 6. The RMSEs are reported in the same units as the original data sets. The number of hidden nodes was set to 50 and 100, respectively, in the experiment for comparison purposes.
2) The weights and biases of the hidden neurons were randomly generated between [-1, 1]. The activation function was selected as a sigmoid function for Pr-ELM. The number of recursions was set as 30, to be in line with the configurations in Section 7.1.
3) All the experiments were conducted in Matlab 2017b running on the same Windows 10 machine with a 16-GB RAM and an Intel i7-6700 CPU @ 3.4 GHz processor.

4) The Matlab built-in functions for LMA and SVR regressions were used to create LMA and SVR system models. Specifically, for LMA, the activation function was tansig, the decrease μ was 0.1, and the minimum performance gradient was 1e-7. Whereas Gaussian kernel function, default box constraint, and sequential minimal optimization solver were used for SVR.
5) The encoded attributes in the data sets were expanded to separate binary attributes based on the number of encoded types. For instance, in the energy efficiency data set, the orientation of the buildings was coded as 2, 3, 4, and 5. However, these numbers could not be directly used in the model training process, as in reality there should be no scaling relationship among the orientations of a building. Therefore, in our experiment, such encoded data attributes were separated into different binary attributes in order to eliminate the improper bias of numerical relationship in the training process. An example of such modifications of the energy efficiency data set is presented in Table 7-4. However, it is worth noting that such an expansion of the number of attributes is not reflected in data set specification as shown in Table 7-3.
6) Two hundred trials were conducted on each algorithm and the averaged results of each algorithm were reported. The authors verified that this number of trials was appropriate to make sure that the mean of the trials would represent the mean of the population based on LLN theorem.

Table 7-4 Data attribute modification for energy efficiency data set

Original Attribute	Modified Attribute			
Orientation	Orietation1	Orietation2	Orietation3	Orietation4
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1

The evaluation metrics of the experiment were training time, testing RMSE, and testing the correlation coefficient. As shown in Table 7-5, when the number of hidden nodes was 50, Pr-ELM could obtain smaller testing RMSE than SVR across all data sets except the Wine Quality (red) data set. However, the difference was minor for the data set. Pr-ELM could achieve a similar level of testing RMSE as LMA for most of the experimental data sets. Its RMSE on the Yacht Hydrodynamics data set was even slightly smaller than that of LMA. Additionally, Pr-ELM achieved very similar performance as LMA in terms of correlation coefficient. Moreover, Pr-ELM had significantly shorter training time than the other algorithms. Specifically, it was over 100 times faster than LMA and more than 20 times faster than SVR in data training under the experimental conditions.

Similar patterns were observed when the number of hidden nodes was 100. As shown in Table 7-6, improved RMSE and correlation coefficient were exhibited on Pr-ELM and LMA compared with the 50 hidden node cases for most of the data sets. Likewise, Pr-ELM also demonstrated similar performance as LMA with respect to RMSE and correlation coefficient. Meanwhile, Pr-ELM maintained a significant advantage in training time over the competitors. However, it was also observed that Pr-ELM and LMA had worse RMSE in the 100 hidden node case than in the 50 hidden node case on some data sets, such as the Yacht Hydrodynamics. One of the reasons for this discrepancy could be training overfitting. Such a problem also implied that the experimental configuration may not be appropriate for all the data sets. Parameter tuning might be required to adapt to the needs of specific applications. In addition, it was noted that the performance of SVR had no apparent changes, which was a piece of evidence that the presented results were the true representations of the population mean values based on the LLN theorem.

Table 7-5 Comparison of Pr-ELM, LMA, and SVR (50 hidden nodes)

Data sets	Pr-ELM			LMA			SVR		
	Testing RMSE	Testing CorrCoef	Training Time	Testing RMSE	Testing CorrCoef	Training Time	Testing RMSE	Testing CorrCoef	Training Time
IC Engine Mapping Data	6.60	0.9992	0.05	2.38	0.9999	40.65	11.25	0.9974	0.24
Airfoil Self-noise	2.76	0.9161	0.03	2.68	0.9207	2.11	3.98	0.8157	0.13
Abalone	2.17	0.7454	0.06	2.19	0.7400	2.68	2.25	0.7362	1.42
Servo Data	0.60	0.91146	0.01	1.56	0.0.6093	0.56	1.48	0.4750	0.03
Concrete Strength	6.13	0.9307	0.01	6.35	0.9269	1.18	7.86	0.8839	0.10
Combined Cycle Power Plant	4.07	0.9712	0.12	3.96	0.9727	5.50	4.21	0.9692	4.93
Energy Efficiency (heating)	1.74	0.9850	0.01	0.79	0.9969	2.88	2.79	0.9616	0.08
Energy Efficiency (cooling)	2.36	0.9690	0.01	1.66	0.9847	2.41	2.97	0.9501	0.09
Wine Quality (red)	0.69	0.5655	0.02	0.69	0.5667	1.37	0.65	0.6031	0.23
Wine Quality (white)	0.72	0.5884	0.06	0.72	0.5881	3.20	0.73	0.5714	1.88
Gas Turbine Emission (CO)	1.23	0.8411	0.46	1.21	0.8454	28.34	1.30	0.8192	89.90
Yacht Hydrodynamics	0.79	0.9987	0.01	1.34	0.9950	0.93	7.37	0.9320	0.04

Table 7-6 Comparison of Pr-ELM, LMA, and SVR (100 hidden nodes)

Data sets	Pr-ELM			LMA			SVR		
	Testing RMSE	Testing CorrCoef	Training Time	Testing RMSE	Testing CorrCoef	Training Time	Testing RMSE	Testing CorrCoef	Training Time
IC Engine Mapping Data	4.03	0.9998	0.11	2.72	0.9999	134.97	11.30	0.9974	0.24
Airfoil Self-noise	6.06	0.7571	0.05	3.11	0.8954	4.04	4.00	0.8147	0.13
Abalone	3.55	0.5321	0.13	2.23	0.7315	10.09	2.25	0.7374	1.41
Servo Data	0.99	0.7946	0.02	2.03	0.5176	1.62	1.50	0.4701	0.03
Concrete Strength	6.37	0.9239	0.04	6.97	0.9122	3.25	7.83	0.8827	0.10
Combined Cycle Power Plant	23.22	0.6486	0.28	3.95	0.9728	10.32	4.19	0.9693	4.89
Energy Efficiency (heating)	1.19	0.9928	0.03	1.24	0.9923	8.68	2.79	0.9618	0.08
Energy Efficiency (cooling)	2.12	0.9745	0.03	2.17	0.9739	7.61	2.99	0.9495	0.09
Wine Quality (red)	1.20	0.3667	0.06	0.72	0.5455	5.59	0.64	0.6065	0.23
Wine Quality (white)	0.86	0.4983	0.15	0.73	0.5747	12.12	0.73	0.5672	1.86
Gas Turbine Emission (CO)	1.49	0.7782	1.15	1.22	0.8465	99.47	1.31	0.8192	88.26
Yacht Hydrodynamics	1.33	0.9949	0.02	2.64	0.9800	2.40	7.25	0.9308	0.04

7.3 Summary

This chapter presented the experimental evaluation and comparison results of the proposed Pr-ELM algorithm. Pr-ELM was compared with other ELM-based algorithms using the IC engine mapping data set. In addition, Pr-ELM was also compared with LMA and SVR algorithms using the IC engine mapping data and eleven publicly available benchmark data sets. The results showed that in addition to the dominant advantage in training time, Pr-ELM had a similar performance as LMA and outperformed SVR in terms of RMSE and correlation coefficient under the evaluation conditions.

Chapter 8. Conclusion and Future Work

8.1 Dissertation Summary

In this dissertation, we explored the SLFN-based approaches for IC engine torque modeling. An ELM-based approach was proposed to create the gasoline engine torque model. Rigorous and comprehensive engine mapping tests were conducted to collect the application data and verify the performance of the created IC engine torque model. Over 10000 minutes of engine dynamometer test were conducted, and more than 3300 high-quality data points were acquired. The experimental evaluation results demonstrated that the proposed approach can create the IC engine torque model with high accuracy. In addition, the approach was extended to a weight-tuning feature, with which the model performance could be tuned based on practical needs of the real-world applications without modifying the available data sets.

Based on the analyses of the popular ELM-based algorithms, a new approach named progressive ELM (Pr-ELM) was proposed. The random feature mapping attribute of the traditional ELM was inherited. The estimated model outputs were used to supplement the input data set for the model, which in turn helped reduce the model estimation error with regard to RMSE. Further experimental evaluation results showed that Pr-ELM could significantly improve the performance of the traditional ELM and outperform the competing ELM-based algorithms in terms of model output RMSE and correlation coefficient.

More evaluations were done with eleven publicly available data sets from the University of California, Irvine, Machine Learning Repository. The performance of Pr-ELM was further compared with the popular and robust non-ELM based algorithms, such as LMA and SVR. In addition to its unparalleled advantage in training speed, the experimental evaluation results showed that Pr-ELM had a comparable performance as LMA in terms of RMSE and correlation coefficient under the experimental conditions. Pr-ELM also demonstrated better performance than SVR in terms of RMSE and correlation coefficient for most of the cases.

8.2 Principal Contributions

The principal contributions of this dissertation are as follows.

1) We established a procedure to create the IC engine torque model using the ELM-based neural network techniques. The modeling approach is efficient and requires minimal prior knowledge of IC engines for its implementation, so even the non-engine experts can also benefit from the approach.
2) Complete engine mapping experiments were conducted in line with the industry-leading engine test standards. Over 3300 high-quality data points were employed to evaluate the IC engine torque model. Such scale and quality of the engine experiment surpassed the previously reported studies and strongly supported the practicality of the proposed model.
3) Rigorous evaluations showed that the created model could predict IC engine output torque over the full engine operating map with high accuracy. The prediction RMSE was about 2.7% of the population mean. The proposed weighted modeling approach effectively enabled the further enhancement of model accuracy.

4) The proposed Pr-ELM strengthened the performance of the existing ELM-based algorithm for system identification applications. Experimental evaluations with the IC engine mapping data and various publicly available data sets showed that Pr-ELM performed significantly better than the competing ELM-based technologies in terms of model accuracy. Pr-ELM also demonstrated comparable performance with LMA and better performance than SVR for different data sets under the evaluation conditions.
5) The proposed approach advanced the techniques for rapid IC engine torque modeling. The created model can provide reliable references for real-world IC engine tests and cross verifications. Based on the evaluation results, the proposed approach can save approximately up to 20 percent of the engine mapping time.

8.3 Additional Remarks and Future Directions

The ELM-based neural network modeling approaches have provided new perspectives for neural network system modeling. Unlike the gradient descent based algorithms, the ELM-based approaches adopt random feature mapping. Therefore, they usually have unparalleled training speed and very good generalization. However, they also appear to perform less accurately, especially when the number of hidden neurons is far less than the number of training data points.

Based on the extensive evaluation of the ELM-based algorithms, we found that for a given data set, the accuracy of the model output largely depends on the output of the hidden layer in the ELM models. If the outputs of the hidden layer are the same, the ELM-based algorithms tend to have very comparable accuracy with the BP-based ones when both are optimized for minimum SSE. As a result, the focus of improving the ELM-based algorithms should be on the improvement of random feature mapping.

The following directions may be of interest in future research.

1) Studying the effectiveness of Pr-ELM on a wider category of applications, such as classification, pattern recognition, and deep learning.
2) Exploring the impact of the initialization process on the performance of the ELM-based algorithms [64]. Since the quality of the feature layer output has a direct effect on the overall performance of the ELM models, it is valuable to conduct a thorough study of this topic.
3) Exploring innovative approach to find the close-form estimation of an ill-conditioned matrix. Based on the studies of the previous ELM-based algorithms, the weight of the hidden layer is often optimized by calculating the inverse of the input matrix using the Moore-Penrose pseudo inverse. However, due to the ill-formed nature of the input matrix, such pseudo inverse is one-sided. In other words, when the pseudo inverse is multiplied with the input matrix from the other side, the result may be far from an identity matrix. As a result, the output of the model may not be the expected value. Therefore, it is desired to have a close-form estimation of the inverse of the input matrix. Possible approaches may include tuning the pseudo inverse of the input matrix by adding linear combinations of the vectors from its null space, or approximating the activation function through accurate linearization.

4) Investigating the opportunity of applying the proposed Pr-ELM on top of other NN algorithms, ELM, and non-ELM based, in order to extract deep data features and improve the overall performance of the algorithms.

5) Exploring the opportunity of utilizing the neural network based IC engine model as the digital twin [112] in engine testing and cross-validation. In the era of industry 4.0, physical objects and virtual twins are growing together. Taking the advantage of the ELM algorithms, the digital twin model may be seamlessly integrated with the IC engine model. The data can be exchanged in real-time and a change made to one end would automatically lead to a change in the other end accordingly. In this regard, the ANN coupled digital twins may open up new possibilities in early fault diagnosis and more efficient and powerful product analysis [113], [114].

REFERENCES

[1] Q. Tan, “Model-Guided Data-Driven Optimization and Control for Internal Combustion Engine Systems,” *Electronic Theses and Dissertations*, 2018, [Online]. Available: https://scholar.uwindsor.ca/etd/7625.

[2] D. Jung and C. Sundstrom, “A Combined Data-Driven and Model-Based Residual Selection Algorithm for Fault Detection and Isolation,” *IEEE Transactions on Control Systems Technology*, vol. 27, no. 2, pp. 616–630, 2019, doi: 10.1109/TCST.2017.2773514.

[3] S. Tolou, R. T. Vedula, H. Schock, G. Zhu, Y. Sun, and A. Kotrba, “Combustion Model for a Homogeneous Turbocharged Gasoline Direct-Injection Engine,” *Journal of Engineering for Gas Turbines and Power*, vol. 140, no. 10, p. 102804, 2018, doi: 10.1115/1.4039813.

[4] N. Togun, S. Baysec, and T. Kara, “Nonlinear Modeling and Identification of a Spark Ignition Engine Torque,” *Mechanical Systems and Signal Processing*, vol. 26, pp. 294–304, 2012, doi: 10.1016/j.ymssp.2011.06.010.

[5] Z. Tan and R. D. Reitz, “An Ignition and Combustion Model Based on the Level-Set Method for Spark Ignition Engine Multidimensional Modeling,” *Combustion and Flame*, vol. 145, no. 1–2, pp. 1–15, 2006, doi: 10.1016/j.combustflame.2005.12.007.

[6] P. Boudier, S. Henriot, T. Poinsot, and T. Baritaud, “A Model for Turbulent Flame Ignition and Propagation in Spark Ignition Engines,” *Elsevier*, vol. 24, no. 1, pp. 503–510, 1992.

[7] C. S. Daw, C. E. A. Finney, J. B. Green, M. B. Kennel, J. F. Thomas, and F. T. Connolly, “A Simple Model for Cyclic Variations in a Spark-Ignition Engine,” *SAE Transactions,* 1996, p. 962086, doi: 10.4271/962086.

[8] S. G. Poulos and J. B. Heywood, “The Effect of Chamber Geometry on Spark-Ignition Engine Combustion,” *SAE transactions*, pp. 1106–1129, 1983.

[9] G. Fontana and E. Galloni, “Variable Valve Timing for Fuel Economy Improvement in a Small Spark-Ignition Engine,” *Applied Energy*, vol. 86, no. 1, pp. 96–105, 2009, doi: 10.1016/j.apenergy.2008.04.009.

[10] J. M. Luján, C. Guardiola, B. Pla, and P. Bares, “Estimation of Trapped Mass by in-Cylinder Pressure Resonance in HCCI Engines,” *Mechanical Systems and Signal Processing*, vol. 66–67, pp. 862–874, 2016, doi: 10.1016/j.ymssp.2015.05.016.

[11] R. W. Weeks and J. J. Moskwa, “Automotive Engine Modeling for Real-Time Control Using MATLAB/SIMULINK,” *SAE Transactions*, pp. 295-309, 1995, doi: 10.4271/950417.

[12] G. Jiang, P. Xie, H. He, and J. Yan, “Wind Turbine Fault Detection Using a Denoising Autoencoder With Temporal Information,” *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 1, pp. 89–100, 2018, doi: 10.1109/TMECH.2017.2759301.

[13] B. Wu *et al.*, “Using Artificial Neural Networks for Representing the Air Flow Rate through a 2.4 Liter VVT Engine,” *SAE transactions*, 2004, doi: 10.4271/2004-01-3054.

[14] N. K. Togun and S. Baysec, “Prediction of Torque and Specific Fuel Consumption of a Gasoline Engine by Using Artificial Neural Networks,” *Applied Energy*, vol. 87, no. 1, pp. 349–355, 2010, doi: 10.1016/j.apenergy.2009.08.016.

[15] Y. Cay, “Prediction of a Gasoline Engine Performance with Artificial Neural Network,” *Fuel*, vol. 111, pp. 324–331, 2013, doi: 10.1016/j.fuel.2012.12.040.

[16] A. Di Mauro, H. Chen, and V. Sick, “Neural Network Prediction of Cycle-to-Cycle Power Variability in a Spark-Ignited Internal Combustion Engine,” *Proceedings of the Combustion Institute*, vol. 37, no. 4, pp. 4937–4944, 2019, doi: 10.1016/j.proci.2018.08.058.

[17] Y. Hu, H. Chen, P. Wang, H. Chen, and L. Ren, “Nonlinear Model Predictive Controller Design Based on Learning Model for Turbocharged Gasoline Engine of Passenger Vehicle,” *Mechanical Systems and Signal Processing*, vol. 109, pp. 74–88, 2018, doi: 10.1016/j.ymssp.2018.02.012.

[18] H. Li, K. Butts, K. Zaseck, D. Liao-McPherson, and I. Kolmanovsky, “Emissions Modeling of a Light-Duty Diesel Engine for Model-Based Control Design Using Multi-Layer Perceptron Neural Networks,” SAE Technical Paper 2017-01-0601, 2017, doi: 10.4271/2017-01-0601.

[19] T. Zheng, Y. Zhang, Y. Li, and L. Shi, “Real-Time Combustion Torque Estimation and Dynamic Misfire Fault Diagnosis in Gasoline Engine,” *Mechanical Systems and Signal Processing*, vol. 126, pp. 521–535, 2019, doi: 10.1016/j.ymssp.2019.02.048.

[20] J. Chen and R. B. Randall, "Intelligent Diagnosis of Bearing Knock Faults in Internal Combustion Engines Using Vibration Simulation," *Mechanism and Machine Theory*, vol. 104, pp. 161–176, 2016, doi: 10.1016/j.mechmachtheory.2016.05.022.

[21] R. Ahmed, M. El Sayed, S. A. Gadsden, J. Tjong, and S. Habibi, "Automotive Internal-Combustion-Engine Fault Detection and Classification Using Artificial Neural Network Techniques," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 1, pp. 21–33, 2015, doi: 10.1109/TVT.2014.2317736.

[22] L. Wen, X. Li, L. Gao, and Y. Zhang, "A New Convolutional Neural Network-Based Data-Driven Fault Diagnosis Method," *IEEE Transactions on Industrial Electronic.*, vol. 65, no. 7, pp. 5990–5998, 2018, doi: 10.1109/TIE.2017.2774777.

[23] G. Huang, Q. Zhu, and C. Siew, "Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks," in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, Budapest, Hungary, 2004, vol. 2, pp. 985–990, doi: 10.1109/IJCNN.2004.1380068.

[24] Z. Gao *et al.*, "Drive Cycle Simulation of High Efficiency Combustions on Fuel Economy and Exhaust Properties in Light-Duty Vehicles," *Applied Energy*, vol. 157, pp. 762–776, 2015, doi: 10.1016/j.apenergy.2015.03.070.

[25] S. Shanmuganathan, "Artificial Neural Network Modelling: An Introduction," in *Artificial Neural Network Modelling*, vol. 628, S. Shanmuganathan and S. Samarasinghe, Eds. Cham: Springer International Publishing, 2016, pp. 1–14.

[26] J. Sjoberg, H. Hjalmarsson, and L. Ljung, “Neural Networks in System Identification,” *IEEE Control systems magazine*, vol. 10, pp. 31-35, 1994.

[27] M. K. Deh Kiani, B. Ghobadian, T. Tavakoli, A. M. Nikbakht, and G. Najafi, “Application of Artificial Neural Networks for the Prediction of Performance and Exhaust Emissions in SI Engine Using Ethanol- Gasoline Blends,” *Energy*, vol. 35, no. 1, pp. 65–69, 2010, doi: 10.1016/j.energy.2009.08.034.

[28] Y. Çay, I. Korkmaz, A. Çiçek, and F. Kara, “Prediction of Engine Performance and Exhaust Emissions for Gasoline and Methanol Using Artificial Neural Network,” *Energy*, vol. 50, pp. 177–186, 2013, doi: 10.1016/j.energy.2012.10.052.

[29] S. Lu and T. Basar, “Robust Nonlinear System Identification Using Neural-Network Models,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 9, no. 3, pp. 407–429, 1998, doi: 10.1109/72.668883.

[30] S. Herculano-Houzel, “The Remarkable, Yet Not Extraordinary, Human Brain as a Scaled-Up Primate Brain and Its Associated Cost,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 109, no. Supplement 1, p. 10661, 2012, doi: 10.1073/pnas.1201895109.

[31] Q. Jarosz, “Single Neuron Cell Hand-tuned.” https://commons.wikimedia.org/wiki/File:Neuron_Hand-tuned.svg (accessed Nov. 20, 2020).

[32] W. S. McCulloch and W. Pitts, “A Logical Calculus of the Ideas Immanent in Nervous Activity,” *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[33] G. Huang, G.-B. Huang, S. Song, and K. You, “Trends in Extreme Learning Machines: A Review,” *Neural Networks*, vol. 61, pp. 32–48, 2015, doi: 10.1016/j.neunet.2014.10.001.

[34] “CS231n Convolutional Neural Networks for Visual Recognition.” https://cs231n.github.io/neural-networks-1/ (accessed Jan. 23, 2021).

[35] Y. Xiaofang, W. Yaonan, S. Wei, and W. Lianghong, “RBF Networks-Based Adaptive Inverse Model Control System for Electronic Throttle,” *IEEE Transactions on Control Systems Technology*, vol. 18, no. 3, pp. 750–756, 2010, doi: 10.1109/TCST.2009.2026397.

[36] S. Elanayar V.T. and Y. C. Shin, “Radial Basis Function Neural Network for Approximation and Estimation of Nonlinear Stochastic Dynamic Systems,” *IEEE Transactions on Neural Networks and Learning Systems,* vol. 5, no. 4, pp. 594–603, 1994, doi: 10.1109/72.298229.

[37] S. Simonenko, V. Bayona, and M. Kindelan, “Optimal Shape Parameter for the Solution of Elastostatic Problems with the RBF Method,” *Journal of Engineering Mathematics*, vol. 85, no. 1, pp. 115–129, 2014, doi: 10.1007/s10665-013-9636-7.

[38] P. Baldi, “Autoencoders, Unsupervised Learning, and Deep Architectures,” in *Proceedings of ICML workshop on unsupervised and transfer learning*, pp. 37–49, 2012.

[39] G. E. Hinton, “Reducing the Dimensionality of Data with Neural Networks,” *Science*, vol. 313, no. 5786, pp. 504–507, 2006, doi: 10.1126/science.1127647.

[40] G. Dong, G. Liao, H. Liu, and G. Kuang, “A Review of the Autoencoder and Its Variants: A Comparative Perspective from Target Recognition in Synthetic-Aperture Radar Images,” *IEEE Geoscience and Remote Sensing Magazine*, vol. 6, no. 3, pp. 44–68, 2018, doi: 10.1109/MGRS.2018.2853555.

[41] J. Schmidhuber, “Deep Learning in Neural Networks: An Overview,” *Neural Networks*, vol. 61, pp. 85–117, 2015, doi: 10.1016/j.neunet.2014.09.003.

[42] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, “Artificial Neural Networks-Based Machine Learning for Wireless Networks: A Tutorial,” *IEEE Communications Surveys and Tutorials*, vol. 21, no. 4, pp. 3039–3071, 2019, doi: 10.1109/COMST.2019.2926625.

[43] P. J. Werbos, “Backpropagation through Time: What It Does and How to Do It,” *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990, doi: 10.1109/5.58337.

[44] G. Wang, O. I. Awad, S. Liu, S. Shuai, and Z. Wang, “NOx Emissions Prediction Based on Mutual Information and Back Propagation Neural Network Using Correlation Quantitative Analysis,” *Energy*, vol. 198, p. 117286, 2020, doi: 10.1016/j.energy.2020.117286.

[45] G. Najafi, B. Ghobadian, T. Tavakoli, D. R. Buttsworth, T. F. Yusaf, and M. Faizollahnejad, “Performance and Exhaust Emissions of a Gasoline Engine with Ethanol Blended Gasoline Fuels Using Artificial Neural Network,” *Applied Energy*, p. 10, 2009.

[46] T. Guo, S. Chang, Z. Chen, H. Huang, and J. Xu, “Fault Monitoring and Diagnosis of Actuators in Electromagnetic Valve-Train Based on Neural Networks Optimization Algorithm,” *IEEE Access*, vol. 7, pp. 110616–110627, 2019, doi: 10.1109/ACCESS.2019.2933881.

[47] W. F. Schmidt, M. A. Kraaijveld, and R. P. W. Duin, “Feedforward Neural Networks with Random Weights,” in *Proceedings., 11th IAPR International Conference on Pattern Recognition.*, The Hague, Netherlands, 1992, vol. Vol II, pp. 1–4.

[48] B. Igelnik and Yoh-Han Pao, “Stochastic Choice of Basis Functions in Adaptive Function Approximation and the Functional-Link Net,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 6, no. 6, pp. 1320–1329, 1995, doi: 10.1109/72.471375.

[49] G. Huang, L. Chen, and C. Siew, “Universal Approximation Using Incremental Constructive Feedforward Networks with Random Hidden Nodes,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 17, no. 4, pp. 879–892, 2006, doi: 10.1109/TNN.2006.875977.

[50] J. Lu, J. Zhao, and F. Cao, “Extended Feed Forward Neural Networks with Random Weights for Face Recognition,” *Neurocomputing*, vol. 136, pp. 96–102, 2014, doi: 10.1016/j.neucom.2014.01.022.

[51] V. Ramanujan, M. Wortsman, A. Kembhavi, A. Farhadi, and M. Rastegari, “What’s Hidden in a Randomly Weighted Neural Network?,” in *2020 IEEE/CVF Conference*

on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, Jun. 2020, pp. 11890–11899, doi: 10.1109/CVPR42600.2020.01191.

[52] W. Cao, L. Hu, J. Gao, X. Wang, and Z. Ming, “A Study on the Relationship between the Rank of Input Data and the Performance of Random Weight Neural Network,” *Neural Computing and Applications*, vol. 32, no. 16, pp. 12685–12696, 2020, doi: 10.1007/s00521-020-04719-8.

[53] Y. Wu, H. Wang, B. Zhang, and K.-L. Du, “Using Radial Basis Function Networks for Function Approximation and Classification,” *ISRN Applied Mathematics*, vol. 2012, pp. 1–34, Mar. 2012, doi: 10.5402/2012/324194.

[54] S. W. Wang, D. L. Yu, J. B. Gomm, G. F. Page, and S. S. Douglas, “Adaptive Neural Network Model Based Predictive Control of An Internal Combustion Engine with A New Optimization Algorithm,” *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 220, no. 2, pp. 195–208, 2006, doi: 10.1243/095440706X72754.

[55] J. Wang, Y. Zhang, Q. Xiong, and X. Ding, “NOx Prediction by Cylinder Pressure Based on RBF Neural Network in Diesel Engine,” in *2010 International Conference on Measuring Technology and Mechatronics Automation*, Changsha City, China, 2010, pp. 792–795, doi: 10.1109/ICMTMA.2010.621.

[56] K. Bizon, G. Continillo, E. Mancaruso, and B. M. Vaglieco, “Towards On-Line Prediction of the In-Cylinder Pressure in Diesel Engines from Engine Vibration Using Artificial Neural Networks,” SAE Technical Paper 2013-24-0137, 2013.

[57] D. Stathakis, “How Many Hidden Layers and Nodes?,” *International Journal of Remote Sensing*, vol. 30, no. 8, pp. 2133–2147, 2009, doi: 10.1080/01431160802549278.

[58] G. Huang and H. A. Babri, “Upper Bounds on the Number of Hidden Neurons in Feedforward Networks with Arbitrary Bounded Nonlinear Activation Functions,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 9, no. 1, pp. 224–229, 1998, doi: 10.1109/72.655045.

[59] N. J. Guliyev and V. E. Ismailov, “A Single Hidden Layer Feedforward Network with Only One Neuron in the Hidden Layer Can Approximate Any Univariate Function,” *Neural Computation*, vol. 28, no. 7, pp. 1289–1304, 2016, doi: 10.1162/NECO_a_00849.

[60] J. Tang, C. Deng, and G.-B. Huang, “Extreme Learning Machine for Multilayer Perceptron,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 4, pp. 809–821, 2016, doi: 10.1109/TNNLS.2015.2424995.

[61] G. Huang, H. Zhou, X. Ding, and R. Zhang, “Extreme Learning Machine for Regression and Multiclass Classification,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 513–529, 2012, doi: 10.1109/TSMCB.2011.2168604.

[62] G. Huang, Q. Zhu, and C. Siew, “Extreme Learning Machine: Theory and Applications,” *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006, doi: 10.1016/j.neucom.2005.12.126.

[63] L. P. Wang and C. R. Wan, “Comments on "The Extreme Learning Machine,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 19, no. 8, pp. 1494–1495, 2008, doi: 10.1109/TNN.2008.2002273.

[64] W. Cao, X. Wang, Z. Ming, and J. Gao, “A Review on Neural Networks with Random Weights,” *Neurocomputing*, vol. 275, pp. 278–287, 2018, doi: 10.1016/j.neucom.2017.08.040.

[65] G. Huang, “An Insight into Extreme Learning Machines: Random Neurons, Random Features and Kernels,” *Cognitive Computation*, vol. 6, no. 3, pp. 376–390, 2014, doi: 10.1007/s12559-014-9255-2.

[66] X. Wang, R. Wang, and C. Xu, “Discovering the Relationship Between Generalization and Uncertainty by Incorporating Complexity of Classification,” *IEEE Transactions on Cybernetics.*, vol. 48, no. 2, pp. 703–715, 2018, doi: 10.1109/TCYB.2017.2653223.

[67] A. E. Hoerl and R. W. Kennard, “Ridge Regression: Biased Estimation for Nonorthogonal Problems,” *Technometrics*, vol. 12, no. 1, p. 55, 1970, doi: 10.2307/1267351.

[68] G. H. Golub, M. Heath, and G. Wahba, “Generalized Cross-Validation as a Method for Choosing a Good Ridge Parameter,” *Technometrics*, no. 21(2), pp. 215–223, 1979.

[69] W. Pan, “Akaike’s Information Criterion in Generalized Estimating Equations,” *Biometrics*, vol. 57, no. 1, pp. 120–125, 2001, doi: 10.1111/j.0006-341X.2001.00120.x.

[70] K. O. Kvålseth, “Cautionary Note About R^2,” *The American Statistician*, vol. 39, no. 4, pp. 279–285, 1985.

[71] O. Harel, “The Estimation of R^2 and Adjusted R^2 in Incomplete Data Sets Using Multiple Imputation,” *Journal of Applied Statistics*, vol. 36, no. 10, pp. 1109–1118, 2009, doi: 10.1080/02664760802553000.

[72] K. P. Burnham, D. R. Anderson, and K. P. Burnham, *Model selection and multimodel inference: a practical information-theoretic approach*, 2nd ed. New York: Springer, 2002.

[73] W. Zong, G.Huang, and Y. Chen, “Weighted Extreme Learning Machine for Imbalance Learning,” *Neurocomputing*, vol. 101, pp. 229–242, 2013, doi: 10.1016/j.neucom.2012.08.010.

[74] K. Li, X. Kong, Z. Lu, L. Wenyin, and J. Yin, “Boosting Weighted ELM for Imbalanced Learning,” *Neurocomputing*, vol. 128, pp. 15–21, 2014, doi: 10.1016/j.neucom.2013.05.051.

[75] R. Fletcher, *Practical methods of optimization*, 2nd ed. Chichester, West Sussex England: John Wiley & Sons, Ltd, 2013.

[76] K. Atashkari, N. Nariman-Zadeh, M. Gölcü, A. Khalkhali, and A. Jamali, “Modelling and Multi-Objective Optimization of A Variable Valve-Timing Spark-Ignition Engine Using Polynomial Neural Networks and Evolutionary Algorithms,” *Energy Conversion and Management*, vol. 48, no. 3, pp. 1029–1041, 2007, doi: 10.1016/j.enconman.2006.07.007.

[77] Meidensha Corporation, “Automotive Testing Systems Specification,” https://www.meidensha.com/products/automobile/prod_01/prod_01_03/index.html [Accessed Dec. 14, 2020].

[78] “ATI Accurate Technologies, “VISION Calibration and Data Acquisition Software,” https://www.accuratetechnologies.com/Products/VISIONSoftware [Accessed Dec. 14, 2020].

[79] T. Holliday, A. J. Lawrance, and T. P. Davis, “Engine-Mapping Experiments: A Two-Stage Regression Approach,” *Technometrics*, vol. 40, no. 2, pp. 120–126, 1998, doi: 10.1080/00401706.1998.10485194.

[80] W. Cao, J. Gao, Z. Ming, and S. Cai, “Some Tricks in Parameter Selection for Extreme Learning Machine,” *IOP Conference Series: Materials Science and Engineering.*, vol. 261, p. 012002, Oct. 2017, doi: 10.1088/1757-899X/261/1/012002.

[81] M. Li and D. Wang, “Insights into Randomized Algorithms for Neural Networks: Practical Issues and Common Pitfalls,” *Information Sciences*, vol. 382–383, pp. 170–178, 2017, doi: 10.1016/j.ins.2016.12.007.

[82] W. Cao, J. Gao, X. Wang, Z. Ming, and S. Cai, “Random Orthogonal Projection Based Enhanced Bidirectional Extreme Learning Machine,” in *Proceedings of ELM 2018*, vol. 11, J. Cao, C. M. Vong, Y. Miche, and A. Lendasse, Eds. Cham: Springer International Publishing, 2020, pp. 1–10.

[83] Y. Yang, Y. Wang, and X. Yuan, “Bidirectional Extreme Learning Machine for Regression Problem and Its Learning Effectiveness,” *IEEE Transactions on Neural*

Networks and Learning Systems., vol. 23, no. 9, pp. 1498–1505, 2012, doi: 10.1109/TNNLS.2012.2202289.

[84] W. Cao, Z. Ming, X. Wang, and S. Cai, "Improved Bidirectional Extreme Learning Machine Based on Enhanced Random Search," *Memetic Computing*, vol. 11, no. 1, pp. 19–26, 2019, doi: 10.1007/s12293-017-0238-1.

[85] W. Cao, M. J. A. Patwary, P. Yang, X. Wang, and Z. Ming, "An Initial Study on the Relationship Between Meta Features of Dataset and the Initialization of NNRW," in *2019 International Joint Conference on Neural Networks (IJCNN)*, Budapest, Hungary, 2019, pp. 1–8, doi: 10.1109/IJCNN.2019.8852219.

[86] G. Huang and L. Chen, "Enhanced Random Search Based Incremental Extreme Learning Machine," *Neurocomputing*, vol. 71, no. 16–18, pp. 3460–3468, 2008, doi: 10.1016/j.neucom.2007.10.008.

[87] G. Huang and L. Chen, "Convex Incremental Extreme Learning Machine," *Neurocomputing*, vol. 70, no. 16–18, pp. 3056–3062, 2007, doi: 10.1016/j.neucom.2007.02.009.

[88] W. Wang and R. Zhang, "Improved Convex Incremental Extreme Learning Machine Based on Enhanced Random Search," in *Unifying Electrical Engineering and Electronics Engineering*, vol. 238, S. Xing, S. Chen, Z. Wei, and J. Xia, Eds. New York, NY: Springer New York, 2014, pp. 2033–2040.

[89] R. Zhang, Y.Lan, G.Huang, and Z. Xu, "Universal Approximation of Extreme Learning Machine With Adaptive Growth of Hidden Nodes," *IEEE Transactions on*

Neural Networks and Learning Systems, vol. 23, no. 2, pp. 365–371, 2012, doi: 10.1109/TNNLS.2011.2178124.

[90] G. Feng, G. Huang, Q. Lin, and R. Gay, "Error Minimized Extreme Learning Machine With Growth of Hidden Nodes and Incremental Learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 20, no. 8, pp. 1352–1357, 2009, doi: 10.1109/TNN.2009.2024147.

[91] H. Rong, Y. Ong, A. Tan, and Z. Zhu, "A Fast Pruned-Extreme Learning Machine for Classification Problem," *Neurocomputing*, vol. 72, no. 1–3, pp. 359–366, 2008, doi: 10.1016/j.neucom.2008.01.005.

[92] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, "OP-ELM: Optimally Pruned Extreme Learning Machine," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 21, no. 1, pp. 158–162, 2010, doi: 10.1109/TNN.2009.2036259.

[93] R. Zhang, Y. Lan, G.-B. Huang, Z.-B. Xu, and Y. C. Soh, "Dynamic Extreme Learning Machine and Its Approximation Capability," *IEEE Transactions on Cybernetics.*, vol. 43, no. 6, pp. 2054–2065, 2013, doi: 10.1109/TCYB.2013.2239987.

[94] Q. He, T. Shang, F. Zhuang, and Z. Shi, "Parallel Extreme Learning Machine for Regression Based on Mapreduce," *Neurocomputing*, vol. 102, pp. 52–58, 2013, doi: 10.1016/j.neucom.2012.01.040.

[95] N. Liang, G. Huang, P. Saratchandran, and N. Sundararajan, "A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks," *IEEE*

Transactions on Neural Networks, vol. 17, no. 6, pp. 1411–1423, 2006, doi: 10.1109/TNN.2006.880583.

[96] E. K. P. Chong and S. H. Żak, *An Introduction to Optimization*, 2nd ed. New York: Wiley, 2001.

[97] J. Cao, Z. Lin, and G.-B. Huang, "Self-Adaptive Evolutionary Extreme Learning Machine," *Neural Processing Letters*, vol. 36, no. 3, pp. 285–305, 2012, doi: 10.1007/s11063-012-9236-y.

[98] Q. Zhu, A. K. Qin, P. N. Suganthan, and G. Huang, "Evolutionary Extreme Learning Machine," *Pattern Recognition*, vol. 38, no. 10, pp. 1759–1763, 2005, doi: 10.1016/j.patcog.2005.03.028.

[99] L. L. Kasun, H. Zhou, B. Huang, and C. M. Vong, "Representational Learning with ELMs for Big Data," *IEEE Intelligent Systems*, p. 5, 2013.

[100] K. Sun, J. Zhang, C. Zhang, and J. Hu, "Generalized Extreme Learning Machine Autoencoder and a New Deep Neural Network," *Neurocomputing*, vol. 230, pp. 374–381, 2017, doi: 10.1016/j.neucom.2016.12.027.

[101] Y. Wan, S. Song, and G. Huang, "Incremental Extreme Learning Machine Based on Cascade Neural Networks," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, Kowloon Tong, Hong Kong, Oct. 2015, pp. 1889–1894, doi: 10.1109/SMC.2015.330.

[102] A. Oakden, “Cascade Networks and Extreme Learning Machines,” MSc Thesis, Canberra: Australian National University, 2014.

[103] T. Gedeon and A. Oakden, “Extreme Learning Machines with Simple Cascades:,” in *Proceedings of the 5th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, Colmar, Alsace, France, 2015, pp. 271–278, doi: 10.5220/0005539502710278.

[104] D. Dua and C. Graff, “UCI Machine Learning Repository,” 2019. https://archive.ics.uci.edu/ml/index.php (accessed Feb. 14, 2021).

[105] The MathWorks, Inc., “Levenberg-Marquardt Backpropagation” https://www.mathworks.com/help/deeplearning/ref/trainlm.html;jsessionid=52ca70dd85a61850337047e46469 (accessed Dec. 05, 2020).

[106] A. Reynaldi, S. Lukas, and H. Margaretha, “Backpropagation and Levenberg-Marquardt Algorithm for Training Finite Element Neural Network,” in *2012 Sixth UKSim/AMSS European Symposium on Computer Modeling and Simulation*, Malta, Malta, 2012, pp. 89–94, doi: 10.1109/EMS.2012.56.

[107] G. Lera and M. Pinzolas, “Neighborhood based Levenberg-Marquardt algorithm for neural network training,” *IEEE Transaction on Neural Networks*, vol. 13, no. 5, pp. 1200–1203, 2002, doi: 10.1109/TNN.2002.1031951.

[108] Dundee Conference on Numerical Analysis and G. A. Watson, Eds., *Numerical analysis: proceedings of the biennial conference, Dundee, 1977*. Berlin: Springer, 1978.

[109] M. T. Hagan and M. B. Menhaj, “Training Feedforward Networks with the Marquardt Algorithm,” *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 989–993, 1994.

[110] M. Awad and R. Khanna, *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*. Apress Media, LLC, 2015.

[111] S. Salcedo-Sanz, J. L. Rojo-Álvarez, M. Martínez-Ramón, and G. Camps-Valls, “Support Vector Machines in Engineering: An Overview: Support Vector Machines in Engineering,” *WIREs Data Mining and Knowledge Discovery*, vol. 4, no. 3, pp. 234–267, 2014, doi: 10.1002/widm.1125.

[112] R. Wagner, B. Schleich, B. Haefner, A. Kuhnle, S. Wartzack, and G. Lanza, “Challenges and Potentials of Digital Twins and Industry 4.0 in Product Design and Production for High Performance Products,” *Procedia CIRP*, vol. 84, pp. 88–93, 2019, doi: 10.1016/j.procir.2019.04.219.

[113] M. Schluse, M. Priggemeyer, L. Atorf, and J. Rossmann, “Experimentable Digital Twins—Streamlining Simulation-Based Systems Engineering for Industry 4.0,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1722–1731, 2018, doi: 10.1109/TII.2018.2804917.

[114] A. Fuller, Z. Fan, C. Day, and C. Barlow, “Digital Twin: Enabling Technologies, Challenges and Open Research,” *IEEE Access*, vol. 8, pp. 108952–108971, 2020, doi: 10.1109/ACCESS.2020.2998358.

VITA AUCTORIS

Name:	Weiying Zeng
Place of Birth:	Changsha, China
Year of Birth:	1984
Education:	China Agricultural University, Beijing, China
	(2003-2007) B.Sc.
	Peking University, Beijing, China
	(2006-2009) B.Econ.
	China Agricultural University, Beijing, China
	(2007-2009) M.Sc. Vehicle Engineering
	University of Windsor, Windsor, Ontario, Canada
	(2013-2021) Ph.D. Electrical Engineering

LIST OF PUBLICATIONS

Publications completed during my registration as a doctoral graduate student at the University of Windsor, ON, Canada

Published Journal papers:

[1] **Zeng, W.**, Khalid, M.A. and Chowdhury, S., 2016, "In-vehicle networks outlook: Achievements and challenges", *IEEE Communications Surveys & Tutorials*, *18*(3), pp.1552-1571.

[2] **Zeng, W.**, Khalid, M.A., Han, X. and Tjong, J., 2020. A Study on Extreme Learning Machine for Gasoline Engine Torque Prediction. *IEEE Access*, *8*, pp.104762-104774.

To be submitted Journal paper:

[1] **Zeng, W.**, Khalid, M.A., Han, X. and Tjong, J., "A Novel Progressive Extreme Learning Machine for System Identification", Applied Intelligence, Springer

Conference paper:

[1] **Zeng, W.**, Khalid, M. and Chowdhury, S., 2015, May. A qualitative comparison of FlexRay and Ethernet in vehicle networks. In *2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE)* (pp. 571-576). IEEE.