Dynamic Modelling of Nonlinear Industrial Processes Using Echo State Network

Kai Liu

A Thesis submitted for the degree of Doctor of Philosophy

School of Engineering

Newcastle University

United Kingdom

March 2022

Abstract

Accurate models are essential for the control and optimisation of industrial processes. Many chemical processes are complex and highly nonlinear and data-driven models need to be capitalised. Artificial neural networks based data-driven models are gaining popularity in chemical engineering area. Echo state network (ESN) is a recurrent neural network with a non-trainable sparse recurrent reservoir and an adaptable readout from the reservoir. Generally, the inputs weight and sparse reservoir connection weights are generated randomly. ESN is a highly effective method for the analysis and prediction of time series data and non-linear data and has been widely used in various research and application areas. However, ESN still has some drawbacks such as: incomprehensible black box properties, the reservoir connection structures and weights require multiple attempts to determine and lack of principled reservoir creation. This has raised the question of deep research of reservoir topology and how to create optimal ESN for modelling of complex chemical processes.

In this thesis, a modular small world reservoir topology for ESN is proposed and it can enhance the dynamic property of the reservoir. Applications to both time series data and chemical process data show that ESN models with the proposed topology perform better than those with randomly generated topology.

To further overcome the drawback of randomly generated reservoir, an adaptive genetic algorithm optimized ESN (AGA-ESN) is proposed. Genetic algorithm (GA) is population based global search and optimization algorithm. An adaptive mechanism is added to adjust the crossover and mutation probability. Four structural parameters of ESN are optimized by GA to promote the modelling performance for the specific modelling tasks.

To overcome the problem of binary coding and discrete searching domain of GA, a continuous and efficient covariance matrix adaptation evolution strategy (CMA-ES) is used to optimize ESN. CMA-ES is a stochastic, derivative-free evolutionary algorithm for difficult non-linear optimization problems in continuous domain. As in AGA-ESN, the ESN reservoir parameters are optimized by CMA-ES. It is shown that CMA-ES models give more accurate predictions than GA-ESN models.

In the final part of the thesis, the ability of ESN on handling multiple inputs data is enhanced by introducing an attention mechanism into ESN. The attention mechanism guarantees the ESN works with the most relevant input data by adding a weighted input scaling vector. It is shown that this modification leads to improved performance on modelling a penicillin fermentation process with complex multiple inputs, the proposed method reduced MSE by 6 times.

Acknowledgement

Throughout the writing of this thesis, I have received a great deal of support and assistance.

I would first like to thank my supervisor, Dr. Jie Zhang, whose expertise was invaluable in formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and brought my work to a high level.

I would like particularly like to acknowledge my teammate, Kazeem Alli, Jeremiah Corrigan, Ruosen Qi, Changhao Zhu and Shengkai Wang, also the visiting academic Kangcheng Wang and Yiming Tian.

I would like to extend my thanks to the staffs and students at the School of Engineering for their help in offering me the resources in my whole PhD period.

I am extremely grateful to my wife Dingming Liu for her love, understanding, help and patience throughout my PhD programme. Also, many thanks to all friends who stayed with me through my 4 years life in the UK.

Last but not least, I wish to thank my parents and all my family members for their support and encouragement throughout my study.

Table of Contents

List of Figures

List of Tables

Nomenclatures

b	bias
$\boldsymbol{C}$	covariance matrix
$diag$	diagonal matrix
f	activate function
E	error
h	tank level
$\boldsymbol{I}$	identity matrix
N	reservoir size
o_i	output of neural networks
p	probability
s_i	hidden state
t	time step
$\boldsymbol{u}$	input data
$\boldsymbol{W}$	reservoir weights matrix
$\boldsymbol{W^{in}}$	input weight
$\boldsymbol{W^{out}}$	output weight
$\boldsymbol{x}$	reservoir state
y	output

Greek letters

α	leak rate
γ	ridge parameter
σ	singular value
ρ	spectral radius

Mathematical operators

$\|\cdot\|$	Euclidean norm
$\odot$	element-wise multiplication

Abbreviations

AGAESN	Adaptative Genetic Algorithm Echo State Networks
AO	Anti-Oja's
ANN	Artificial Neural Networks
BAGNET	Bootstrap Aggregated Network
BP	Backpropagation
BPDC	Back-Propagation Decorrelation Neural Network
BPTT	Backpropagation Through Time
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
CSA	Cumulative Step Length Adaptation
DBN	Deep Belief Network
DLR	Delay Line Reservoir
DLRB	Delay Line Reservoir with Feedback Connections
ER	Erdős–Rényi Model
ESN	Echo State Networks
ESP	Echo State Property
FNN	Feedforward Neural Networks
GA	Genetic Algorithm
LSTM	Long Short-Term Memory
LSM	Liquid State Machines
LVQ	Learning Vector Quantization
MLP	Multilayer Perceptron
MMA	Methyl Methacrylate
MSE	Mean Square Error
MSMESN	Modular Small World Echo State Networks
MSO	Multiple Superimposed Oscillator Time Series
PFC	Prefrontal Cortex
PPC	Parietal Cortex
RBM	Restricted Boltzmann Machine
RC	Reservoir Networks
RLS	Recursive Least Squares
RNN	Recurrent Neural Networks
RTRL	Real-time Recurrent Learning

SGD	Stochastic Gradient Descent
SCR	Simple Cycle Reservoir
SVD	Singular Value Decomposition
SVM	Support Vector Machine

Chapter 1. Introduction

1.1 Research background

The development of chemical industry has a great influence on the development of any country. It involves almost every aspect of national economy, resource development, national defence and people's basic necessities. Chemical industry is the basic industry of national economy and its development is related to the overall situation of national economic development. At the same time, the chemical industry is facing many challenges including three traditional requirements which are safety, economy and stability, and also increasingly needs to meet the requirements of environmental protection, sustainable development and product design. In order to meet these challenges, modern chemical enterprises need to combine traditional chemical technology with modern computer software and hardware technology to establish a more accurate chemical process model. Therefore, the establishment of a more accurate chemical process model has become a constant pursuit of most chemical practitioners.

The development of modern chemical industry has the following characteristics：

1. Increasing level of automation of the production processes.
2. Increasing level of process optimization to meet different demands of the changing market.
3. Increasing level of application on operation optimization and improvement of production conditions in practice.

These new characteristics of the chemical industry lead to higher requirements for the improvement of operating conditions in the production process and the innovation of process monitoring and control. These changes make our research on chemical production more detailed and deeper. Chemical process models are the basis of studying the characteristics, the properties of chemical substances and chemical production process. Establishing a good model has an important role in control, optimization and simulation of the chemical process.

In recent years, the development of computer simulation technology provides a new way and method for chemical process modelling. The development of computer technology leads to the possibility of complex chemical process modelling. Traditional mechanism modelling is difficult to develop and their simplifications may not accurately reflect the reality while some empirical modelling methods brought by computer technology are very valuable. In recent years, the rapid development of neural network, machine learning and other artificial intelligence of new computing methods have a strong role in promoting chemical process modelling research created a new situation of empirical modelling. Computer simulation has played a very important role in process control, fault diagnosis and production evaluation, and the core of computer simulation is the chemical process modelling established on the basis of the accurate model of chemical plant, so modelling accurately has become a problem that must be solved in the development of chemical industry.

Chemical production processes often face raw material variations, production load changes, process changes and so on, resulting in production instability and difficulty on smooth operation and control, in order to make the production stable under optimal operating conditions, accurate chemical process models are required to achieve optimal control strategy.

1.2 Challenges in process modelling

Generally, for real-world problems, first principle models and data-driven models are available, choosing one of these two methods is related mainly to amiable knowledge, goal of simulations and problems approached. First principle models are often engineering design models by mathematic equations, reflecting chemical and physical laws such as mass balance, energy balance, heat transfer relations and so on, which use nonlinear algebraic or differential equations (Rodrigues and Minceva, 2005). For many years, the mechanistic models have been used to model and simulate chemical process, such as secreted protein (Park and Fred Ramirez, 1988), ibuprofen crystal growth and size distribution (Nayhouse et al., 2015), control of polymer quality (Kozub and MacGregor, 1992). However, the traditional methods of solving chemical engineering problems have met their limitations, on the computational point, the chemical and physical based equations are difficult to apply numerical methods to approximate them. Another

disadvantage is the difficulties in developing model, quantifying the phenomenology through mathematical formalism. The accuracy of the numerical calculations proven by errors in the process parameters is also a critical problem, such as the algebraic errors in the calculations, round-off errors, approximation errors and iteration errors (Hoffman and Frankel, 2018). Moreover, in some particular situations, for example, when time is a critical aspect in control modules, the first principle models are not suitable because of high computational time required to generate results. Also, there can be certain elements with unknown or partially unknown parameters in some chemical processes. The parameters can be found by using the experimental measurements from the real system and fitting them to the mathematical model in system identification, in some cases, can be reduced to curve fitting and regression analysis (Fritzson, 2011).

The development of intelligent manufacturing, Industrial Internet of Things (IIOT) and data storage technology has facilitated the collection, review and use of massive amounts of data in various industries, especially the chemical industry because of huge amount data generation. To overcome the difficulties in developing first principle models, data-driven Artificial Neural Networks (ANN) have been widely proposed. Neural networks have been proved to be able to approximate and continuous nonlinear functions (Cybenko, 1989, Girosi and Poggio, 1990, Park and Sandberg, 1991) and have been applied to process modelling and control, such as an augmented recurrent neural network is used to control a fed-batch bioreactor (Tian et al., 2002), modelling a Xylose production (Alvarez et al., 1999), pollution prediction model (Arena et al., 1996), an online model for a semi-regenerative catalytic reformer (Chessari et al., 1994). Normally, ANN for nonlinear process modelling can be separated into two parts: static networks and dynamic networks. Static networks including multi-layer feedforward neural networks and radial basis function networks can provide accurate one-step-ahead predictions and are suitable for short-range predictions (Rój and Wilk, 1998, Chen et al., 1990). Correspondingly, dynamic networks including locally recurrent neural networks (Zhang et al., 1998), globally recurrent neural networks (Su et al., 1992), Elman networks (Elman, 1990) and dynamic filter networks (Morris et al., 1994), are more appropriate for providing multi-step-ahead predictions and building of long-range prediction models.

Chemical process data are characterized by their inherently time-series nature with intrinsic causality. Since the feed-forward networks generation is based on the

combination among the process variables its behaviour is static, thus is not good at being applied in dynamic systems. Effective data analysis through deep learning is very valuable for determining process performance improvement and product quality assurance strategies. If the process system is highly complex, and the data is multi-dimensional, noisy, and dynamic, then modelling based on Echo State Networks (ESN) should be a feasible method for system characterization and performance prediction (Bianchi et al., 2018).

1.3 Motivation

Reservoir computing (RC) refers to a class of new state-space models transition structures with fixed states and adaptive readout of the state space. Echo state networks are a type of reservoir computing which are quite different from classical RNNs in that stability can be maintained if the condition called 'echo state property' is satisfied. An ESN is composed of a reservoir and a linear output layer which maps the reservoir states to the network output with some advantages such as excellent dynamic modelling performance and fast training compared to the conventional RNNs. As the most famous Reservoir Computing (RC), Echo State Network (ESN) has been successfully applied in time-series prediction problem (Jaeger and Haas, 2004, Song et al., 2010), identification of nonlinear dynamic systems (Jaeger, 2002a, Han and Lee, 2013), language modelling (Skowronski and Harris, 2006) and industrial chemical engineering modelling (Wang and Yan, 2014).

However, there are still several challenges prohibiting ESN to become a widely used tool in chemical engineering modelling:

1. The properties and topology of reservoir are poorly understood (Xue et al., 2007).
2. Due to the black-box nature of ESN, the creation of weights and connection structures for reservoir often requires multiple attempts and sometimes even an element of luck (Ozturk et al., 2007).
3. Different reservoir cannot be identified for specific objects (Ozturk and Principe, 2007).

1.4 Aim and objectives

This project aims to develop efficient RNN-based models for complex chemical engineering processes.

The main objectives for achieving the above aim are:

1. Develop the reservoir topology of echo state networks.
2. Optimize the structure of ESN reservoir to improve the model reliability, robustness and generalization capability in chemical engineering processes by applying genetic algorithm and evolution strategy.
3. Combine attention mechanism with ESN to model the complex multiple inputs chemical engineering processes.

1.5 Structure of the thesis

In Chapter 2, the development of artificial neural networks, classification and training algorithms are reviewed. The shortcomings of the traditional ANN training algorithm are described, followed by a detailed introduction to the composition structure and training algorithms of ESN and LSTM.

In Chapter 3, the research of topology construction of reservoir is presented and the modular small world ESN is proposed. The modular reservoir topology is designed to enhance the richness of dynamical properties within the reservoir and improves the performance of ESN.

In Chapter 4, an adaptative genetic algorithm is used to optimize ESN by adaptatively changing the crossover and mutation probabilities. The decision variables in the optimization are the structural parameters of ESN which are reservoir size, leak rate, spectral radius and sparseness density. The proposed method is compared to the genetic algorithm optimized ESN and conventional ESN on the dynamic case studies.

In Chapter 5, a covariance matrix adaption evolution strategy (CMA-ES) based ESN is proposed. The CMA-ES is used to optimize the structure parameters including reservoir size, leak rate and spectral radius to improve the modelling performance. The proposed method is applied to three case studies: modelling a time-series data, modelling tank level in a conic tank, and modelling a fed-batch penicillin fermentation process.

In Chapter 6, an optimized attention mechanism based ESN is proposed. This enables the ESN to have the attentiveness capacity, and the important levels of distinct variables in

input vectors will be addressed in an adaptive manner according to their relevance. In the multiple inputs penicillin fermentation process, the results show that the attention-based methods are better than the conventional ESN and LSTM to make prediction of product concentration, furthermore, after optimizing the key structure parameters by CMA-ES, the proposed network' performance is best among all the compared approaches. This illustrates that the inputs attention score has the similar impact level with the other parameters of ESN as: reservoir size, leak rate and spectral radius. The proposed O-Att-ESN can be used in other multiple input applications as well.

Chapter 7, summarize the conclusions from the research and recommendations for the future work.

1.6 Publications

1. Liu, K., & Zhang, J. (2020). Nonlinear process modelling using echo state networks optimized by covariance matrix adaption evolutionary strategy. *Computers & Chemical Engineering*, *135*, 106730.
2. Liu, K., & Zhang, J. (2021). A Dual-Layer Attention-Based LSTM Network for Fed-batch Fermentation Process Modelling. In *Computer Aided Chemical Engineering* (Vol. 50, pp. 541-547). Elsevier.
3. Liu, K., & Zhang, J. (2020). Modelling a Penicillin Fermentation Process Using Attention-Based Echo State Networks Optimized by Covariance Matrix Adaption Evolutionary Strategy. In *Computer Aided Chemical Engineering* (Vol. 48, pp. 1117-1122). Elsevier.
4. Liu, K., & Zhang, J. (2018, September). Optimization of Echo State Networks by Covariance Matrix Adaption Evolutionary Strategy. In *2018 24th International Conference on Automation and Computing (ICAC)* (pp. 1-6). IEEE.

Chapter 2. Literature review

2.1 Introduction

The human brain is a highly nonlinear, complex parallel computational machine which is organized by complicated internal structure of components named as neurons, so that the brain can perform specific tasks such as controlling and feedback of body temperature, blood pressure, heartbeat, breath and some other more advanced missions including pattern recognition and classification. In an adult brain, there are approximately 20 billion cerebral cortex neurons and each neuron is connected with others by 10 thousand synapsis. The largest computational machine in the world is made by Stanford's Artificial Intelligence Lab with only 1.2 billion neurons (Coates et al., 2013). Although in some certain tasks, the ANN is faster than biological brain, most time the human brain performs many tasks much faster than ANN today but many of the human brain operating mechanism is still unknown. For instance, human vision is an image-processing work, it is the function of visual system to get pictorial information from surrounding environment and give the feedback to the information, and to be specific, recognizing a familiar face of human brain usually takes about 100ms, whereas the much less complex task takes a longer time on a powerful computer. Figure 2.1 shows the complicated neurons structure in the human brain (Haykin, 2010).

Figure 2.1The internal structure of biological brain (Haykin, 2010).

The intelligent information processing of artificial neural networks has led to a gradual expansion of their application areas and many of the problems solved by traditional information processing methods can be well solved by neural networks. However, the development of artificial neural networks has not always been smooth. In 1943, a mathematical algorithm named threshold logic was created as the first model of neural networks, this work described how neurons might work in biologic brain with modeling a simple neural network by electrical circuits (McCulloch and Pitts, 1943). This method

pioneered the neural networks to separate into two ways, one focuses on the research of brain biological procedure and the other is concentrated on artificial intelligence application with neural network. In 1958, the formal neural network was improved by Perceptron theory, a single-layer perceptron which is the simplest Feed Forward Network using gradient descent training was established to classify a continuous values pack into two sides by computing the sum of weighted input singles and getting one of the two possible values out as the classification result (Rosenblatt, 1958). In 1974, two key issues were discovered, firstly the original perceptron was only able to manage linear issues, secondly under the limitation of computer computational ability, large neural networks could not be realized (Minsky, 1974). These two issues caused the research stagnation of neural networks for almost two decades called 'disillusioned years' (Minsky, 1974). An important trigger of calling back the interests in neural networks is backpropagation (BP) algorithm which distributes the error values backwards through the layers and modified the strength at every neurons generated (Werbos, 1990). Based on the backpropagation theory, Multilayer Perceptron (MLP) was generated, which can distinguish nonlinearly separable data and has been proved that it can approximate a continuous function in any closed interval (Rumelhart et al., 1986). From then on, the artificial neural networks development entered a new stage. Echo State Networks (ESN) is proposed in 2001 which is further promotion of artificial neural networks (Jaeger, 2001).

The reminding part of this chapter is organized as follows. Section 2.2 introduces feedforward neural networks. Section 2.3 presents recurrent neural networks. Echo State Network is introduced in detail in Section 2.4, and finally, in Section 2.5, a brief summary of this chapter is given.

2.2 Feedforward Neural Networks

In general, artificial neural networks could be separated into two fundamental classes: feedforward networks and recurrent networks. Further classification of feedforward networks includes single layer feedforward networks and multilayer neurons networks. Refer to layered neural networks, neurons are formed into different layers including input layer, hidden layer(s) and output layer. Generally, there are one input layer and one output layer in a neural network. The input layer is an input vector constituted by input nodes without calculation function and these kinds of nodes just represent input signal values and

send them into next layer directly. The output layer produces the final results and keep them in storage. Between them are single or multiple hidden layers which are composed of hidden nodes. Hidden layers influence connections between the input data and external output results in specific manner. With single or multiple hidden layers, the neural networks can grab higher-dimensional statistical properties from input signal. The term 'feedforward' means the information moves in the forward direction, from input nodes to output neurons through hidden layers without cycles (feedback) or loops. In other words, input signal passes to the first hidden layer (which is the second layer in the network) and, after transformation in first hidden neurons, first hidden layer outputs form the inputs of the second hidden layer, as well as the rest layers of the networks. Figure 2.2 shows a general structure of the multilayer feedforward network with three hidden layers. So far, there are some representative feedforward neural networks including Multilayer Perceptron (MLP), Bootstrap Aggregated Network (BAGNET), Support Vector Machine (SVM) and Learning Vector Quantization (LVQ), etc. (Hofmann et al., 2008). The most typical and widely used feedforward neural network model is Multilayer Perceptron (MLP) mentioned before. MLP follows feedforward direction strictly. In a MLP, there are *K* input units which are $\boldsymbol{u} = (u_1, u_2, \dots, u_K)$, *N* hidden neurons $\boldsymbol{s} = (s_1, s_2, \dots, s_N)$ and *L* output nodes $\boldsymbol{o} = (o_1, o_2, \dots, o_L)$ and $\boldsymbol{W}_u^s$ and $\boldsymbol{W}_s^o$ denote weights from input layer to hidden layer and from hidden layer to output layer, respectively. A typical three layers MLP is shown in Figure 2.3. The input signals to a hidden neuron are the neuron outputs from the previous layer and are weighted and then summed with a bias. For the j^{th} node on the hidden layer, this is:

$$net_j = \sum_{i=1}^{K} W_{u(i,j)}^{s} u_i + b_{1,j} \tag{2.1}$$

Where $W_{u(i,j)}^{s}$ is the weight between input *i* and hidden neuron *j* and $b_{1,j}$ is the bias for the i^{th} hidden node.

This sum then passes through an activation function f, to form the output values of the hidden layer, $(s_1, s_2, \dots, s_N)$.

$$s_j = f(net_j) \tag{2.2}$$

The l^{th} output on the latest layer, o_l, is given by the following equation:

$$o_l = f(\sum_{i=1}^{N} W^o_{s(i,l)} s_i + b_{2,l}) \tag{2.3}$$

Where $W^o_{s(i,l)}$ denotes the weight between the i^{th} hidden node and the l^{th} output node, $b_{2,l}$ is the bias for the l^{th} output node, and f is the activation function, such as linear function, sigmoid function.

Using the backpropagation (BP) algorithm to train the whole MLP is to modify the connection weights and bias. BP starts from the error between the network output o_l and the desired or target output $\hat{o}_l$ (Doya, 1995).

$$E = \sum_{l=1}^{L} \|o_l - \hat{o}_l\|^2 \tag{2.4}$$

where $\|\cdot\|$ represent the Euclidean norm.

In a MLP, all neural weights and bias are initialised randomly and they are adjusted during the training process to find the best weights and bias to minimize the prediction error.

Multilayer feed-forward neural networks have been successfully used in modelling and optimization of chemical processes. For example, mechanistic modelling of polymerization reactors is quite difficult, the difficulties include the large number of kinetic parameters which are hard to determine, the complex and numerous reactions and chemical species occurring simultaneously inside the reactor and the poor understanding of phenomena for mixtures involving polymers. Under particular situation, the free radical polymerization of methyl methacrylate (MMA), associated with gel and glass effects, simple MLPs were used to model the mass reaction viscosity, number average molecular weight, gravimetric average molecular and monomer conversion depending on initial working conditions (Curteanu, 2004).

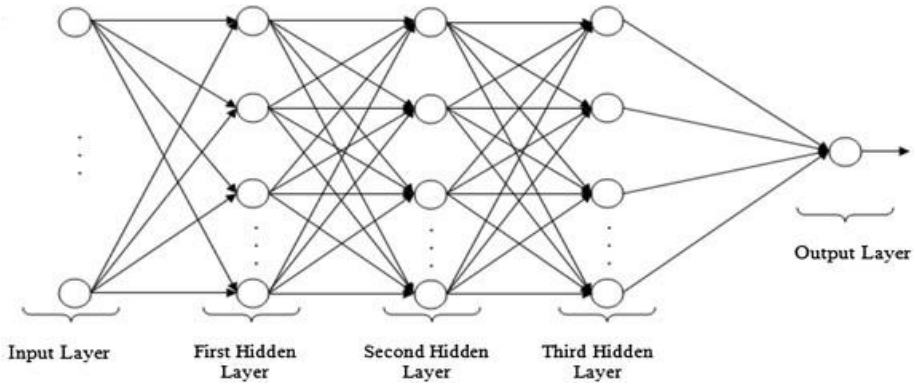


Figure 2.2 Architectural graph of a multilayer network with three hidden layers. (Zeki et al., 2016)

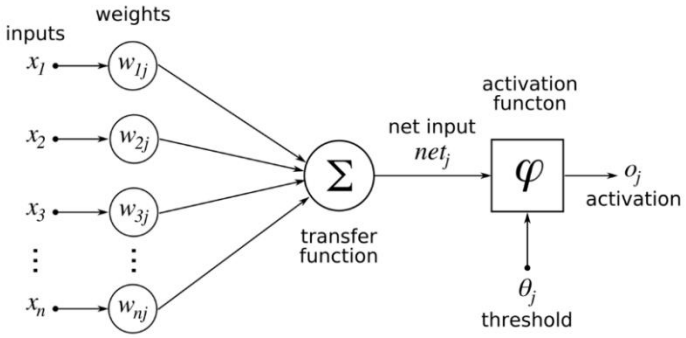


Figure 2.3 A sample graph for Multilayer Perceptron. (Polat, 2017)

2.3 Recurrent neural networks

Recurrent Neural Network (RNN) is a large computational model inspired by biological brain modules. Although RNNs have great theoretical promise, the training of RNNs has always been a problem for the academic community. In RNN, many neurons are connected to each other, just like synapses in the brain, and these connections allow signals to propagate through the network, also the signal can propagate from the output neurons. Figure 2.4 shows the typical structure of RNN, where $\boldsymbol{u}$, $\boldsymbol{W}$ and $\boldsymbol{y}$ denotes the input, weight matrix and output respectively. The recurrent property is reflected in the following two points:

1. RNN can be developed in the absence of an input signal as an automatic transient generator that is connected around recursion and can autonomously maintain an automatic transient signal generator, mathematically, illustrating that RNN can act as a dynamical system while the FNN is simply a function approximator.
2. When driven by an input signal, RNN converts the input signal from past moments into the internal state of the hidden layer internal states, the RNN has dynamic memory and can handle transient information.

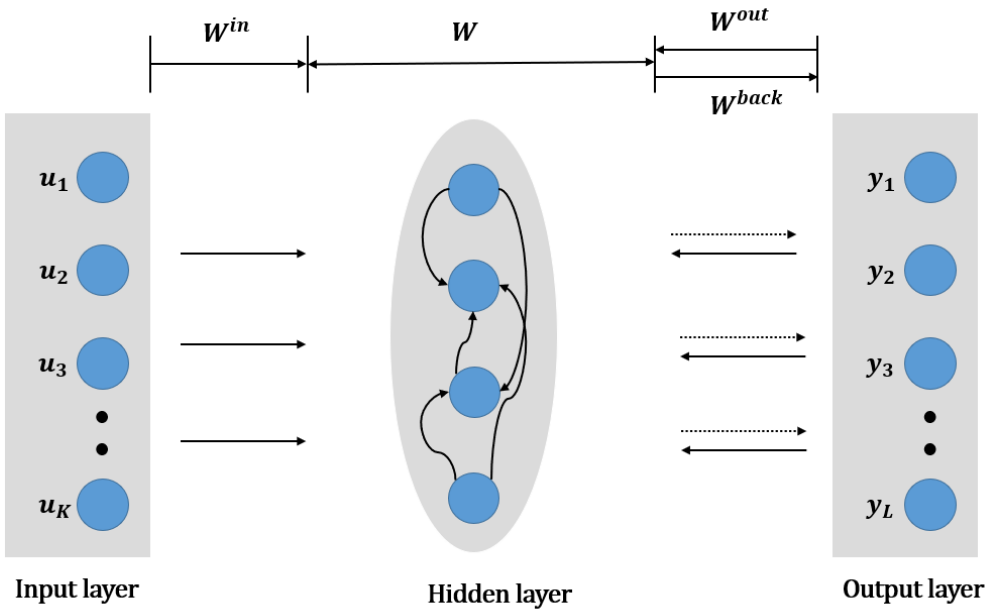


Figure 2.4 Typical structure of RNN.

From a dynamic system perspective, there are two types of RNN. One type of RNN is the symmetric topological RNN with the aim of maintaining the minimum energy, the most typical network for this type is the Hopfield network (Hopfield, 2007). The training algorithm for the Hopfield network is unsupervised learning algorithm and the main functions are memory association, data compression, data distribution and static mode classification. Hopfield network is based on the principles of statistical physics. In contrast, another type of RNN models are mainly characterized by heuristic dynamic updates and direct connection methods. This type of networks is mainly used to implement nonlinear filtering, which converses a specific timing input signal into a class of output timing signals. The mathematical background of this type of RNN is nonlinear dynamical systems, and their standard training algorithm is a supervised learning algorithm. The RNNs discussed in this chapter are all based on the second type of RNNs.

The main reason that RNN is a good solution to the non-linear time series problem is that RNN can be used as a general approximator for arbitrary dynamic system. RNN has already demonstrated its strong research and application value in certain fields. Although many scholars generally believe that RNN has great promise, the application of RNN to dynamic non-linear systems is still very limited. The main reason is that the traditional training algorithm of RNN is based on gradient descent. This type of algorithm has many

inherent drawbacks, mainly in the following three aspects (Pascanu et al., 2013, Zhang et al., 2019, Chen et al., 2020):

1. The gradient descent algorithm aims to reduce the training error cyclically, but during the RNN training process, if the execution is not done properly while using gradient descent, it may lead to problems like gradient vanishing or gradient exploding, which does not guarantee the convergence performance of the RNN.
2. In the gradient descent algorithm, an individual parameter update requires so much computation, and the update of these parameters is often achieved through many iterations. This leads to the problem of long training times and thus prevents the structure of the RNN from being too large.
3. In essence, gradient information decreases exponentially in traditional RNNs, so it is difficult for traditional RNNs to learn data with long-term memory.

A typical Recurrent neural network is consisted to have *K* input neurons, *N* hidden neurons and *L* output nodes. These nodes take discrete time *t* as step length, so the input activation, hidden layer activation and output activation are denoted as $\boldsymbol{u}_t = (u_1, u_2, \dots, u_K)$, $\boldsymbol{s}_t = (s_1, s_2, \dots, s_N)$ and $\boldsymbol{o}_t = (o_1, o_2, \dots, o_L)$, respectively. $\boldsymbol{W}$ and $\boldsymbol{U}$ represents the $N \times N$ and $N \times K$ internal connection matrices. The outputs of hidden layer node and output layer node at time t is:

$$s_t = f(Uu_t + Ws_{t-1}) \tag{2.5}$$

$$o_t = g(Vs_t) \tag{2.6}$$

Where o_t is the network output value, s_t represents hidden state value, u_t is the input value at time step t, g and f represent active functions of output layer and hidden layers respectively, U, W and V are input weights, hidden layer weights and output layer weights, respectively.

Plug Eq 2.5 into Eq 2.6 repeatedly, the following can be obtained:

$$
\begin{aligned}
o_t &= g(Vs_t) \\
&= \mathrm{g}(Vf(Uu_t + Ws_{t-1})) \\
&= \mathrm{g}(Vf(Uu_t + Wf(Uu_{t-1} + Ws_{t-2}))) \\
&= \mathrm{g}(Vf\left(Uu_t + Wf(Uu_{t-1} + Wf(Uu_{t-2} + Ws_{t-3}))\right)) \\
&= \mathrm{g}(Vf\left(Uu_t + Wf(Uu_{t-1} + Wf(Uu_{t-2} + Wf(Uu_{t-3} + \cdots)))\right))
\end{aligned} \tag{2.7}
$$

It can be found from the above equations that the output of RNNs o_t is influenced by previous inputs $u_t, u_{t-1}, u_{t-2}, u_{t-3}$, ... This is why it called recurrent neural network.

In contrast with FNN, RNN has better performance on nonlinear and dynamic problems. There are four traditional RNN training methods which are Backpropagation Through Time (BPTT), Real-time Recurrent Learning (RTRL), Long Short-Term Memory (LSTM), and reservoir networks. These are further explained below.

1. Backpropagation Through Time

   BPTT is an extended algorithm of backpropagation which is used in training feedforward neural networks. The basic principle of BPTT is unfolding the RNN to a multilayer feedforward network in a step-by-step manner under time dimension. Training data is separated by time interval from t_s to t_e, meaning that in this time interval, the multilayer feedforward network is unfolding from $s(t_s)$ to $s(t_e)$. Finally, the total loss gradient is the sum of the loss gradient for each sequence of time (Werbos, 1990):

$$net_t = Uu_t + Ws_{t-1} \tag{2.8}$$

$$s_{t-1} = f(net_{t-1}) \tag{2.9}$$

   Where net_t is the weighted input of a neuron at time t.

$$\frac{\partial net_t}{\partial net_{t-1}} = \frac{\partial net_t}{\partial s_{t-1}} \frac{\partial s_{t-1}}{\partial net_{t-1}}$$

$$\frac{\partial net_t}{\partial s_{t-1}} = W$$

$$\frac{\partial s_{t-1}}{\partial net_{t-1}} = diag[f'(net_{t-1})]$$

$$\frac{\partial net_t}{\partial net_{t-1}} = \frac{\partial net_t}{\partial s_{t-1}} \frac{\partial s_{t-1}}{\partial net_{t-1}} = W \cdot diag[f'(net_{t-1})] \tag{2.10}$$

Where $diag[]$ represents a diagonal matrix and f' is the derivative of f.

$$\begin{aligned}
\delta_k^T &= \frac{\partial E}{\partial net_k} \\
&= \frac{\partial E}{\partial net_t} \frac{\partial net_t}{\partial net_k} \\
&= \frac{\partial E}{\partial net_t} \frac{\partial net_t}{\partial net_{t-1}} \frac{\partial net_{t-1}}{\partial net_{t-2}} \cdots \frac{\partial net_{k+1}}{\partial net_k} \\
&= W diag[f'(net_{t-1})]\, W diag[f'(net_{t-2})] \cdots W diag[f'(net_k)] \delta_t^l \\
&= \delta_t^T \prod_{i=k}^{t-1} W diag[f'(net_i)]
\end{aligned} \tag{2.11}$$

Where δ_k^T is temporal gradient flow-back component, it can be seen that it is a product of the same gradient starting from time T and extending until end of the sequence k. $\beta_f \beta_w$ is the upper bound for the norm of diagonal matrix $diag[f'(net_i)]$ and weight matrix W.

The unfolding process of RNN is shown in Figure 2.5. The difference between BPTT and BP is calculating the locality gradient error in the selecting time interval and then pass the locality gradient error to other layers.

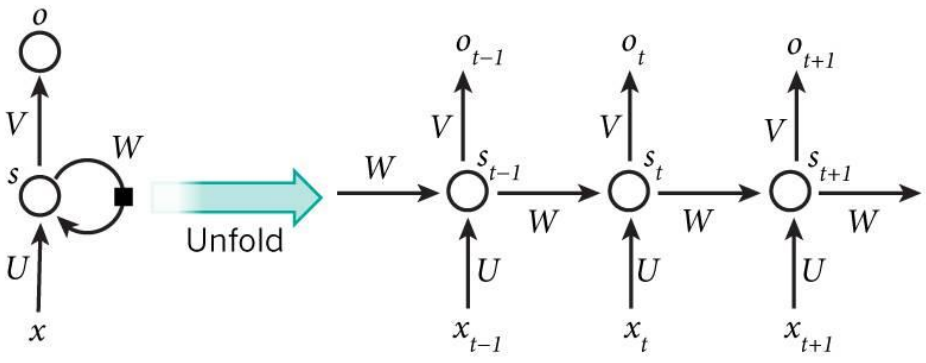


Figure 2.5 Unfold the RNN to multilayer feedforward network.

2. Real-time Recurrent Learning (RTRL)

   RTRL is a feedforward learning method which is opposite to BPTT. The training process calculates the error gradient at time t firstly, then through the recursive

procedure to obtain the derivate coefficient of the error gradient at time $t+1$ (Williams and Zipser, 1989).

3. Long Short-Term Memory (LSTM)

   Long Short Term Memory network (LSTM) is a type of RNN which have ability to learn long-term information. It is first introduced in 1997 and optimized by some researchers in the past years (Hochreiter and Schmidhuber, 1997). It is a kind of gate neural network, using forget gate to determine what information should be going to throw away from the cell and even the scale of information. Figure 2.6 shows a typical LSTM network contains four interacting layers.

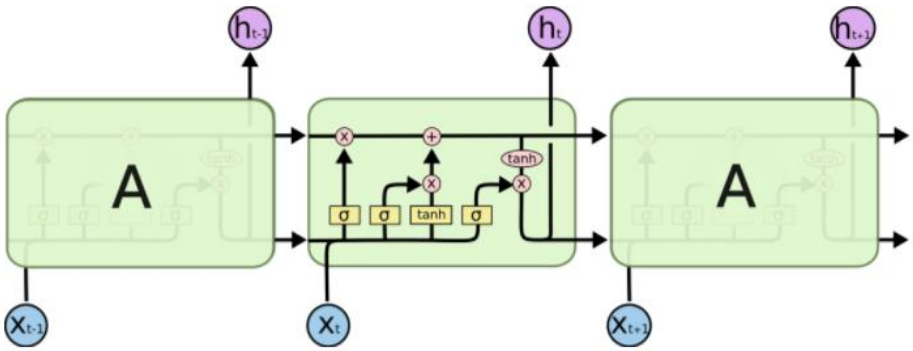


Figure 2.6 A LSTM model contains four interacting layers.

4. Reservoir Networks

   Reservoir computing (RC) refers to a class of new state-space models transition structures with fixed states and adaptive readout of the state space. Reservoir computing is a novel framework for designing and training recurrent neural networks, and it has emerged in the last decade as an alternative to gradient descent methods for training RNNs. The reservoir should be complex enough to capture a large number of reservoir features mapped to the input stream that the output readings can utilize. Echo State Networks (ESNs) (Jaeger, 2001), Liquid State Machines (LSMs) (Maass et al., 2002) and the back-propagation decorrelation neural network (BPDC) (Steil, 2004) are examples of popular RC methods. In RC, the input data is converted to a spatiotemporal pattern in a higher dimensional space. Then, a mode analysis from the spatiotemporal mode is performed during the readout process. The main features of the input weights and the weights in the reservoir are not trained and only the reading

weights are trained with a simple learning algorithm, such as linear regression. This simple and fast training process makes it possible to significantly reduce the computational cost of learning compared to standard RNNs, which is the main advantage of RC (Lukoševičius and Jaeger, 2009).

The role of reservoir in reservoir computing is to nonlinearly transform continuous inputs into high-dimensional spaces, so that features of inputs can be effectively read out by simple learning algorithms. In particular, the application of physics-based RC utilizing reservoirs has attracted more and more interest in many research fields in recent years. Various physical systems and devices are proposed to implement RC. The motivation for physical realization of the reservoir is that the device for achieving rapid information processing has low learning costs. For hardware implementations where training is required, it typically relies on advanced training neural network hardware technologies (Misra and Saha, 2010) and neuromorphic hardware (Hasler and Marr, 2013). In contrast, the physical realization of a reservoir can use various physical phenomena in the real world, because a mechanism is not necessary for training to adapt to changes. In fact, physical RC is another choice for unconventional computing based on new hardware examples (Hadaeghi et al., 2017).

2.4 Echo state networks

2.4.1 Echo state networks

Echo state networks are a type of recurrent neural networks which are quite different from classical RNNs in that stability can be maintained if the condition called ‘echo state property’ is satisfied. An ESN is composed of a reservoir and a linear output layer which maps the reservoir states to the network output. Figure 2.7 shows the original ESN where all input neurons are connected to the reservoir neurons (the hidden layer) which are totally linked with the output layer, while the output neurons can be reconnected backwards with the input neurons and reservoir neurons. The input weights are generated randomly, but the internal weights between reservoir neurons can be created with a sparse connection density which means that internal neurons may not be fully connected to each other but connected sparsely. In addition, the backwards weights for recurrent connections from the output to the reservoir neurons (if necessary) can be generated similarly as the input weights. When an ESN is established, the weights described above will not change

during the training process and only the readout output weights need to be learned. Thus ESN can be built much faster with less computation effort than other RNNS trained by BPTT approach.

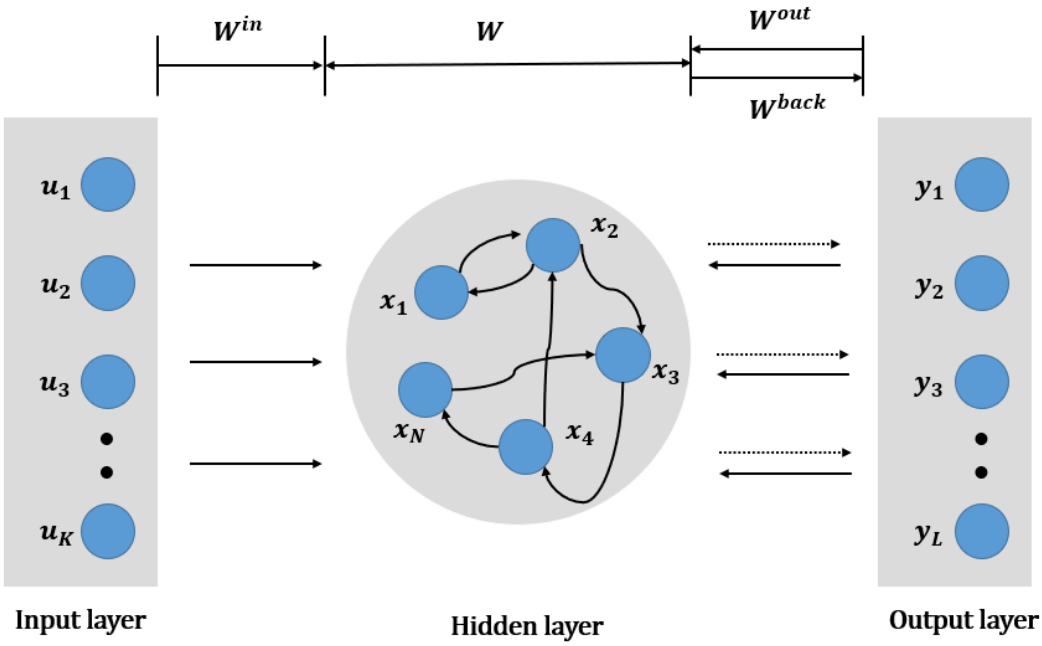


Figure 2.1 Typical structure of an echo stat network (Liu and Zhang, 2020).

The typical state updating and $\boldsymbol{W}$ rescaling of ESN are given by Eq 2.12 and Eq 2.13 respectively:

$$\boldsymbol{x}(t) = \boldsymbol{f}\left(\boldsymbol{W^{in}} \cdot \boldsymbol{u}(t) + \boldsymbol{W} \cdot \boldsymbol{x}(t-1) + \boldsymbol{W^{back}} \cdot \boldsymbol{y}\,(t-1) + \boldsymbol{W^{bias}}\right) \tag{2.12}$$

$$\boldsymbol{W} \leftarrow \frac{\boldsymbol{\delta W}}{|\boldsymbol{\theta_{Max}}|} \tag{2.13}$$

In the above equations, $\boldsymbol{u}(t)$, and $\boldsymbol{x}(t)$ represents the input and the reservoir states at time t respectively. In addition, f is the general activation function, it can be $\boldsymbol{sign}()$ or $\boldsymbol{tanh}()$. The weights donated by $\boldsymbol{W^{in}}$, $\boldsymbol{W^{back}}$, $\boldsymbol{W^{bias}}$, and $\boldsymbol{W}$ represent the weights for input, feedback, bias, and reservoir respectively. They are initialized randomly, generally from a normal distribution with zero mean and unit variance. Then $\boldsymbol{W}$ needs to be rescaled by dividing the matrix by a spectral radius (the largest absolute eigenvalue of $\boldsymbol{W}$, $|\boldsymbol{\theta_{Max}}|$) and then multiplying by a spectral radius factor $\boldsymbol{\delta}$ to keep its echo state property. The ESN output can be calculated as follows.

$$\boldsymbol{y}(t) = f^{out}\left(\boldsymbol{W^{inout}} \cdot \boldsymbol{u}(t) + \boldsymbol{W^{out}} \cdot \boldsymbol{x}(t-1) + \boldsymbol{W^{outout}} \cdot \boldsymbol{y}\,(t-1) + \boldsymbol{W^{biasout}}\right) \tag{2.14}$$

$\boldsymbol{y}(t)$ presents output at time step t, $\boldsymbol{f^{out}}$ is output layer activation function, here a linear function is used. The weights denoted by $\boldsymbol{W^{inout}}$, $\boldsymbol{W^{out}}$, $\boldsymbol{W^{outout}}$, $\boldsymbol{W^{biasout}}$ are the input-output, output, output-output and bias-output weights. All the output weights are learned by some linear regression algorithms such as pseudo-inverse, principal component regression (PCR), ridge regression and least angle regression. In this thesis, the output equation is simplified as:

$$\boldsymbol{y}(t) = f^{out}\left(\boldsymbol{W^{out}} \cdot \boldsymbol{x}(t-1)\right) \tag{2.15}$$

For an ESN with K inputs, N reservoir neurons, and L outputs, $\boldsymbol{W^{in}}$ is a $N \times K$ weight matrix, $\boldsymbol{W}$ is a $N \times N$ weight matrix, $\boldsymbol{W^{out}}$ is a $L \times N$ weight matrix and $\boldsymbol{W^{back}}$ is a $N \times L$ weight matrix. A new development of ESN is an addition of the leak rate factor $\boldsymbol{\alpha}$ which can effectively coordinate the dynamic property of intern reservoir. The modification of the original ESN equation is shown in Eq. 2.16 (Lukoševičius and Jaeger, 2009):

$$\begin{aligned}\boldsymbol{x}(t) = (1-\alpha) \cdot \boldsymbol{x}(t-1) + \\ \alpha \cdot f\left(\boldsymbol{W^{in}} \cdot \boldsymbol{u}(t) + \boldsymbol{W} \cdot \boldsymbol{x}(t-1) + \boldsymbol{W^{back}} \cdot \boldsymbol{y}\,(t-1)\right)\end{aligned} \tag{2.16}$$

A typical ESN does not use feedback connections, but for a time-series problem, adding feedback connection can increase the prediction accuracy. However, this structure will complicate the reservoir, increase the calculation cost and reduce whole system stability. During the training process, the reservoir states obtained are collected in a state matrix $\boldsymbol{X}$ which is updated at discrete time step:

$$\boldsymbol{X} = \begin{bmatrix} x^T(1) \\ x^T(2) \\ \vdots \\ x^T(t) \\ \vdots \\ x^T(n) \end{bmatrix} \tag{2.17}$$

and the corresponding target outputs are collected in a target output matrix $\boldsymbol{Y}$:

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}(1) \\ \boldsymbol{y}(2) \\ \vdots \\ \boldsymbol{y}(t) \\ \vdots \\ \boldsymbol{y}(n) \end{bmatrix} \tag{2.18}$$

where n is the number of training samples. The readout matrix should then be obtained by solving a linear regression problem:

$$\boldsymbol{X} \cdot \boldsymbol{W^{out}} = \boldsymbol{Y} \tag{2.19}$$

The least-squares solution (LSM) is the common method to solve $\boldsymbol{W^{out}}$:

$$\boldsymbol{W^{out}} = arg\min_{w} \|\boldsymbol{Xw} - \boldsymbol{Y}\|^2 \tag{2.20}$$

where $\|\cdot\|$ denotes the Euclidean norm, and the readout matrix $\boldsymbol{W^{out}}$ is given by the following:

$$\boldsymbol{W^{out}} = (\boldsymbol{X^T X})^{-1} \boldsymbol{X^T Y} \tag{2.21}$$

ESN can be trained both offline and online by minimizing the given loss function. Mean Square Error (MSE) is the most used method for evaluating the performance of ESN:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y(i) - y_{test}(i))^2 \tag{2.22}$$

where y is the actual output and y_{test} is the desired output and n is the number of data point.

2.4.2 ESN Training Method

2.4.2.1 The offline Training

In the offline training mode, input the training data firstly, and then compute the output weights that minimise the MSE. In general, the training procedure contains the following steps:

1. Randomly create a large reservoir and initialize the input weight matrix $\boldsymbol{W^{in}}$ and reservoir weight matrix $\boldsymbol{W}$, make sure each element in $\boldsymbol{W^{in}}$ and $\boldsymbol{W}$ is following uniform distribution. Run the ESN on the training data set. During training, $\boldsymbol{W^{in}}$ and $\boldsymbol{W}$ are fixed.
2. Dismiss data from initial *washout* period and collect the remaining network states $x(t)$ and arrange row-wise into a matrix $\boldsymbol{X}$, where in case of direct input-output connections, the matrix $\boldsymbol{X}$ collects inputs $s(t)$ as well.
3. The target values from the training set after washout period are collected in a vector $\boldsymbol{Y}$.

4. The output weight matrix $\boldsymbol{W^{out}}$ can be calculated by one of the following four methods:
    1. Singular value decomposition (SVD): perform SVD on an $M \times N$ matrix $\boldsymbol{X}$ as $\boldsymbol{X = PSQ^T}$, where T denotes transpose operation, $\boldsymbol{P, Q}$ denotes $M \times M$ and $N \times N$ orthonormal matrices respectively, and $\boldsymbol{S}$ is an $M \times N$ diagonal matrix containing singular values in descending order $\delta_{11} \geq \delta_{22} \geq \ldots \geq \delta_{NN} \geq 0$. The output weight matrix, $\boldsymbol{W^{out}}$, can be found by solving $\boldsymbol{XW^{out} = Y}$.
    2. Pseudoinverse solution: The output weight matrix $\boldsymbol{W^{out}}$ is computed by multiplying the pseudoinverse of $\boldsymbol{X}$ with $\boldsymbol{Y}$, $\boldsymbol{W^{out} = Y \cdot X^{-1}}$.
    3. Wiener-Hopf solution: The output weight matrix $\boldsymbol{W^{out}}$ is computed by $\boldsymbol{W^{out} = M^{-1}D}$, where $\boldsymbol{M = X^T X}$ is the correlation matrix of the reservoir states and $\boldsymbol{D = X^T Y}$ is the cross-correlation matrix between the states and the desired outputs.
    4. Ridge regression: A regularization method named ridge regression has been shown to be an efficient method to calculate the readout matrix (Dutoit et al., 2009). The ridge regression is a shrinkage method that consists of adding a penalty term proportional to the Euclidean norm of the readout matrix:

$$\boldsymbol{W^{out}} = arg\min_{w} \|\boldsymbol{Xw - Y}\|^2 + \gamma\|\boldsymbol{w}\|^2 \qquad (2.23)$$

    where $\gamma \geq 0$ is the ridge parameter determined on a hold-out validation set. The solution of readout matrix is replaced as following:

$$\boldsymbol{W^{out}} = (\boldsymbol{X^T X} + \gamma \boldsymbol{I}_N)^{-1}\boldsymbol{X^T Y} \qquad (2.24)$$

    where $\boldsymbol{I}_N$ is the identity matrix of size N.

If $\boldsymbol{X}$ is of full rank (number of reservoir units), then SVD, Pseudoinverse, and Wiener-Hopf methods are similar and equivalent. If $\boldsymbol{X}$ is not full rank, then the matrix $\boldsymbol{M = X^T X}$ of the Wiener-Hopf solution is ill-conditioned, the SVD and Pseudoinverse methods are stable. In this thesis, all the training of ESN is offline training and output weights matrix $\boldsymbol{W^{out}}$ is calculated by ridge regression methods.

2.4.2.2 ESN online training method.

Online training is the opposite to offline training and it is a recursive modification of the output weights as the training data inputting. Online training can be achieved by stochastic gradient descent (SGD) (Kountouriotis et al., 2005) and the output weights are modified

by the changes of reservoir states and the output of ESN, the output weights can be computed by:

$$\boldsymbol{W^{out}}(t+1) = \boldsymbol{W^{out}}(t) + \varphi(y(t) - y_t(t)u(t) \tag{2.25}$$

where φ is the learning rate, $u(t)$ is the input, y and y_t denotes the readout output and desired output respectively. The convergence of SGD heavily relies on learning rate α, also the eigenvalues of the cross-correlation matrix influence the performance (Lukoševičius and Jaeger, 2009).

In order to increase the convergence of SGD method. Jaeger (Jaeger, 2002b) posted Recursive Least Squares (RLS) to train ESN online. RLS is an adaptive filter, by minimizing MSE to update the output weights. In RLS, after the initial washout period, the output weights $\boldsymbol{W^{out}}$ are recursively updated at every time step t:

$$k(t) = \frac{\theta(t-1)\boldsymbol{X}(t)}{\boldsymbol{X}^T(t)\theta(t-1)\boldsymbol{X}(t) + \gamma} \tag{2.26}$$

$$\theta(t) = \frac{\theta(t-1) - k(t)\boldsymbol{X}^T(t)\theta(t-1)}{\gamma} \tag{2.27}$$

$$\boldsymbol{W^{out}}(t) = \boldsymbol{W^{out}}(t-1) + k(t)[y(t) - y_t(t)] \tag{2.28}$$

where *k* stands for the innovation vector; θ is the error covariance matrix initialized with large diagonal values. Forgetting parameter, γ, $(0 < \gamma < 1)$, is usually set to a value close to 1. From the equations, it can be found that RLS is a special case of Kalman filter. Compared to SGD, RLS has higher computational complexity and RLS based ESN has fast velocity of convergence, but it may be affected by the unstable data.

2.4.3 Stability analysis

Stability is a basic problem that must be considered and analyzed when using recurrent neural networks. Traditional RNNs, such as Hopfield neural networks, Cohen-Grossberg neural networks, BAM neural networks, need to construct different Lyapunov functional and use LaSalle principle to obtain global stability criterion or exponential stability criterion (Li and Wu, 2010, Alzabut et al., 2020). As the stability conditions of traditional

recursive neural networks are relatively conservative, it is not conducive to the promotion and application of recurrent neural networks (Park et al., 2008, Cao and Li, 2005, Song and Cao, 2007, Forti and Tesi, 1995, Wang and Zou, 2002). Different from the traditional recurrent neural networks, the reservoir connection weights of echo state network are fixed in the training process, and only the output weights of the network are adjusted. Therefore, the echo state network can guarantee the stability of the internal state of the network by setting the initial connection weight of the reservoir, which greatly improves the practicability of the network and provides a new direction for the research of recurrent neural networks.

Echo state network has good network performance due to an important feature - echo state property (ESP). The echo status property indicates that the network is not affected by the initial state. The internal state of the reservoir $\boldsymbol{X}(t)$ is uniquely determined by the history input $u(t), u(t-1), \ldots u(1)$.The internal state reaches asymptotic stability. The stability of network internal state is the basis for the extensive application of echo state networks. The process of proving the stability of echo state network is as follows (Jaeger, 2002b):
Definition 1: assume the activate function f is a tanh function, when the largest singular value $\sigma(\sigma = \sigma_{max}(\boldsymbol{W}))$ of weight matrix of reservoir is less than 1, the echo state property of internal network is stable.

Given two different internal states in an echo state network, which are $\boldsymbol{x}(t)$ and $\boldsymbol{x}(t)\,'$, with the same input $u(t)$:

$$\|\boldsymbol{h}(t+1)\|_2 = \|\boldsymbol{x}(t+1) - \boldsymbol{x}(t+1)\,'\,\|_2$$
$$= \left\| f\left(\boldsymbol{W}^{in}\boldsymbol{u}(t+1) + \boldsymbol{W}\boldsymbol{x}(t)\right) - f\left(\boldsymbol{W}^{in}\boldsymbol{u}(t+1) + \boldsymbol{W}\boldsymbol{x}(t)\,'\right)\right\|_2$$
$$\leq \left\|\left(\boldsymbol{W}^{in}\boldsymbol{u}(t+1) + \boldsymbol{W}\boldsymbol{x}(t)\right) - \left(\boldsymbol{W}^{in}\boldsymbol{u}(t+1) + \boldsymbol{W}\boldsymbol{x}(t)\,'\right)\right\|_2$$
$$= \|\boldsymbol{W}\boldsymbol{x}(t) - \boldsymbol{W}\boldsymbol{x}(t)\,'\|_2$$

$$= \|\boldsymbol{W}(\boldsymbol{x}(t) - \boldsymbol{x}(t)\,')\|_2 \tag{2.29}$$

According to the compatibility of norms:

$$\|\boldsymbol{W}(\boldsymbol{x}(t) - \boldsymbol{x}(t)\,')\|_2 \leq \|\boldsymbol{W}\|_2\|\boldsymbol{x}(t) - \boldsymbol{x}(t)\,'\|_2$$
$$\|\boldsymbol{h}(t+1)\|_2 \leq \|\boldsymbol{W}\|_2\|\boldsymbol{x}(t) - \boldsymbol{x}(t)\,'\|_2$$
$$= \|\boldsymbol{W}\|_2\|\boldsymbol{h}(t)\|_2$$

$$= \sigma(\boldsymbol{W})\|\boldsymbol{h}(t)\|_2 \tag{2.30}$$

Obviously, when the largest singular value $\sigma_{max}(\boldsymbol{W})$ of the reservoir matrix is less than 1 and $t \to \infty$, $\|\boldsymbol{h}(t)\|_2 = 0$.

Based on the above analysis, the following conclusions are posted:

1. The stability of ESN is closely related to the reservoir weight matrix $\boldsymbol{W}$. If the maximum singular value $\sigma_{max}(\boldsymbol{W})$ is less than 1, then the ESN has Echo State Property.
2. The definition is a sufficient condition for the internal state stability of ESN. However, meeting this condition is relatively strict and is often ignored in practical application (Strauss and Tino, 2005). For the convenience of application, as long as the spectral radius of reservoir connection weight matrix is less than 1, i.e. $\rho(\boldsymbol{W}) < 1$, the network is considered to have Echo State Property.
3. Because the reservoir connection weights matrix $\boldsymbol{W}$ will remain unchanged after the initial setting, and during training process the reservoir matrix $\boldsymbol{W}$ will not modulated. Therefore, the training of ESN does not affect the stability of internal networks.

Generally, in order to ensure the ESN has ESP, a reservoir meeting the necessary conditions must be generated, i.e. $\rho(\boldsymbol{W}) < 1$. The reservoir connection weights matrix can be computed:

$$\boldsymbol{W} = \alpha_{\boldsymbol{W}} \left(\frac{\boldsymbol{W}^r}{|\lambda_{max}|} \right) \tag{2.31}$$

$$|\lambda_{max}| = max\{\lambda_i(\boldsymbol{W}^r)\}, i = 1,2, \dots N \tag{2.32}$$

where $\alpha_{\boldsymbol{W}}$ is sparse factor, with $0 < \alpha_{\boldsymbol{W}} < 1$. $\boldsymbol{W}^r$ is a randomly generated sparse matrix, λ_{max} is the maximum eigenvalues of matrix $\boldsymbol{W}^r$, and λ_i is i-th eigenvalues of matrix $\boldsymbol{W}^r$. Adjusting sparse matrix $\boldsymbol{W}^r$ by the sparse factor $\alpha_{\boldsymbol{W}}$, a reservoir connection weight matrix $\boldsymbol{W}$ with spectral radius less than 1 can be obtained, so that the network meets the necessary conditions of Echo State Property.

2.4.4 The parameters of the reservoir

In the practical application of ESN, generating a good reservoir has great influence on the modelling ability. In view of a practical problem, in order to build a good reservoir, the

functions of the reservoir need to be deeply analysed first. Generally, the reservoir has two important functions: nonlinear high-dimensional transformation and memory of input data. In the field of machine learning, there is a similar relationship between echo state network and kernel function method. For time series prediction, the role of the reservoir can be seen as transforming the input signal u_t into the reservoir state through high-dimensional nonlinear transformation $\boldsymbol{x}_t$. For pattern classification tasks, the input signal u_t is linearly indivisible in its original space, after the reservoir transforms the input data into a high-dimensional nonlinear space, it can be linearly separable by the output weight $\boldsymbol{W^{out}}$ (Hermans and Schrauwen, 2012). Another function is that the reservoir has memory function for time - related data. When choosing to use the ESN in practice, the first consideration is the memory function of the reservoir. The model needs to remember the previously learned information, which is an important factor for choosing to use the ESN for time series prediction. There is no need to use an ESN if the model is not required to have memory in a practical problem.

ESN is a data-driven dynamic model, the balance between nonlinear high-dimensional transformation and data memory should be considered when setting the parameters of the reservoir (Verstraeten et al., 2010). The global parameters of the ESN are as follows: the number of neurons in the reservoir n, the sparseness of the reservoir $\alpha_{\boldsymbol{W}}$, the distribution of reservoir connection weight matrix $\boldsymbol{W}$, the spectral radius λ of reservoir connection weight matrix $\boldsymbol{W}$, the scaling ratio α_{in} of input data and the leakage rate γ.

2.4.4.1 Reservoir size and sparseness

If the training data is not too much, selecting too many neurons n in the reservoir will lead to insufficient data for training, which will degrade the performance of the ESN. It is important to find the computational balance for practical problems to be solved using ESN. The lower limit of the dimension n of the reservoir is roughly estimated by considering the number of independent data that the reservoir must remember. Independent data refers to the data that the ESN needs to remember in order to successfully complete its task (Jaeger, 2001). For independent and equally distributed input data, the number of independent data is roughly estimated by multiplying the dimension of the input data by the time step which needed to memorize. In fact, there is often a correlation between time and variables in the input data u_t, which makes it somewhat compressible. In addition, the forgetting curve of

the reservoir is usually not rectangular as forgetting is a gradual process. Therefore, the actual number of neurons n needed by the reservoir can be smaller than the theoretical value.

In an ESN, there is a sparse connection between neurons in a reservoir, which means that the connection weights of a majority of elements in the reservoir would be equal to zero (Jaeger, 2001). In the simulation experiments, it is found that sparse connections may slightly improve the predictive performance of ESN (Li et al., 2012). In practice, if the connection weight matrix of the reservoir is sparse, the sparse reservoir can improve the running speed of the ESN.

The connection weight matrix of the reservoir $\boldsymbol{W}$ is usually set as sparse, and the non-zero elements in the matrix are usually uniformly or normally distributed centred on zero. Different researchers may prefer to use different distributions. Gaussian distribution is widely used to generate the connection weight matrix of reservoir. The uniformly distributed data has boundedness and continuity, so uniform distribution is also widely used. In fact, the two distributions can make the ESN have similar performance, also depending on the settings of other parameters. The input connection weight matrix $\boldsymbol{W}^{in}$ is typically generated with the same type of distribution as $\boldsymbol{W}$.

2.4.4.2 Input scaling and leak rate

A scale factor used for sizing the input connection matrix is a key parameter in the optimization of an ESN. For the input connection matrix $\boldsymbol{W}^{in}$ that follows a normal distribution, the standard deviation can be used as the scale factor.

In order to reduce the number of freely adjustable parameters, it is usual to adjust all the columns of the input matrix $\boldsymbol{W}^{in}$ together using a single scale factor. However, if the first column of a matrix corresponds to the offset input of a neuron, the first column of the matrix $\boldsymbol{W}^{in}$ and the rest of the matrix $\boldsymbol{W}^{in}$ can be adjusted separately. The value range of the input data can be optimized by scaling and transforming the input data. In fact, the same effect can be achieved by adjusting the weights of the actual and biased inputs of the matrix $\boldsymbol{W}^{in}$ by different scale factors.

It can be seen from Eq 2.16 that adjusting the range of input weight matrix $\boldsymbol{W}^{in}$ and reservoir connection weight matrix $\boldsymbol{W}$ separately can make different effect on the current time step input signal $u(t)$ and previous time step $\boldsymbol{x}(t-1)$. On the other hand, it is necessary to consider the respective numbers of neurons in the input matrix and reservoir. The reservoir tends to eliminate the spectrum of the principal component of input data $u(t)$ in reservoir state $\boldsymbol{x}(t)$, so before the data is input into the ESN, the correct representation of the data needs to be selected or pre-processed. For example, if the smaller principal components of the input data $u(t)$ do not carry useful information, they may be removed from the data by principal component analysis before they are entered into the reservoir, where they would otherwise be amplified.

The leak rate in Eq 2.16 is considered as an important parameter to decide the updating speed of reservoir state $\boldsymbol{x}(t)$. A time-continuous differential equation is used to describe the dynamic characteristics of state update of reservoir:

$$\dot{\boldsymbol{x}}(t) = -\boldsymbol{x}(t) + f\left(\boldsymbol{W} \cdot \boldsymbol{x} + \boldsymbol{W}^{in} \cdot u\right) \tag{2.33}$$

Make Euler discretization of $\dot{\boldsymbol{x}}$ in time gives:

$$\dot{\boldsymbol{x}} \approx \frac{\triangle \boldsymbol{x}}{\triangle t} = \frac{\boldsymbol{x}(t+1) - \boldsymbol{x}(t)}{\triangle t} \tag{2.34}$$

Therefore, in the discrete time Eq 2.16, α can be replaced by the sampling interval $\triangle t$, and α can be regarded as the time interval discretized between two continuous time steps. In addition, when the signal changes slowly, setting α works similarly to resample the input and output (Lukoševicius et al., 2006, Schrauwen et al., 2007). The leak rate α can even be adjusted online for the time interval between processing signals (Jaeger et al., 2007a).

In some cases, choosing α is difficult and subjective. This global parameter needs to be obtained by trial-and-error method, especially when the input and output time scales are quite different. When a task needs to model a dynamic system that generates time series on multiple time scales, a feasible approach is to set different leak rates for different inputs, which results in more parameters that need to be optimized (Holzmann and Hauser, 2010).

In addition, Eq 2.16 can also be thought of as a simple digital low-pass filter. Some literature even suggests using more powerful filters for this purpose (Wyffels et al., 2008).

In some cases, a small leak rate can make the reservoir state has slow changing dynamic characteristics and can significantly increase the short-term memory duration of ESN.

2.4.4.3 Spectral radius

The most important global parameter of ESN is considered as the spectral radius of the connection weight matrix of reservoir, which is defined as the absolute value of the maximum eigenvalue of the matrix. The spectral radius can be used to scale the matrix $\boldsymbol{W}$, which changes the width of the distribution of the non-zero elements of the matrix. Generally, the sparse connection weight matrix $\boldsymbol{W}$ is generated randomly, and its spectral radius $\rho(\boldsymbol{W})$ is calculated, then $\boldsymbol{W}$ divided by ρ to generate a matrix with unit spectral radius. The final spectral radius can be easily adjusted to scale the matrix $\boldsymbol{W}$. In order for the ESN to work effectively, the reservoir should satisfy ESP and the reservoir state $\boldsymbol{x}(t)$ should be uniquely determined by input $u(t)$. In other words, for a sufficiently long input $u(t)$, the final reservoir state $\boldsymbol{x}(t)$, should not be related to the initial conditions prior of the input. If the spectral radius of the reservoir is larger, the reservoir state $\boldsymbol{x}(t)$ may have multiple fixed points, periodicity or chaos, and the ESP is violated.

The reservoir may violate the ESP even when the spectral radius is $\rho(\boldsymbol{W}) < 1$, but this is almost impossible to happen in practice. More importantly, for non-zero input $u(t)$, the reservoir has ESP even at spectral radius $\rho(\boldsymbol{W}) > 1$. This can be explained by the large input $u(t)$ pushing the activated neuron away from 0, where the $\tanh$ activate function of the neuron is nonlinear and has a single slope in the region with a small slope, thus reducing the gain of the neuron and the strength of the feedback connection. This means that for non-zero input $u(t)$, the spectral radius $\rho(\boldsymbol{W}) < 1$ is not a necessary condition for the ESP, and the optimal spectral radius value may sometimes be significantly greater than 1.

In practice, the spectral radius value $\rho(\boldsymbol{W}) = 1$ is taken as the initial reference point, and the spectral radius is selected to make the performance of the ESN close to optimization. In general, the spectral radius should be larger for tasks requiring a longer input memory, and smaller where the current output force depends more on the recent input signal. The spectral radius determines how quickly the influence of input on the state of the reservoir disappears over time and how stable the reservoir is.

2.4.5 The optimization and application of ESN

1. The dynamic weights of reservoir

   In the traditional ESNs, the connection weights in the reservoir are fixed after generating. But the randomly generated reservoir weights may not be the most suitable for the specific data and tasks, sometimes choosing initial parameters depends on experience. Thus, dynamic reservoir is an available method to optimize the ESN when processing data is a critical concern.

   The Principle Neuron Reinforcement (PNR) algorithm was proposed to change the strength of reservoir synapses by modifying reservoir connections, so as to achieve the goal of optimizing neuron dynamic weights, and PNR changed the connection according to the output weight of pre-training phase (Fan et al., 2017). The local plasticity rules which allow variety neurons to utilize different parameters to learn local features was proposed, replacing the simple type of plasticity rule in common ESNs, so that the connections between neurons remain global (Wang et al., 2019). A critical echo state network was proposed and the weights of reservoir were embedded by the input matrix weights and output matrix weight (Hajnal and Lőrincz, 2006). An Anti-Oja's (AO) based ESN was proposed where AO modified the connection weights by declining the correlation between neurons, the AO-ESN had richer internal state dynamics and larger diversity between neurons (Babinec and Pospíchal, 2007).

2. Modes of multiple reservoirs

   The common structure of ESN is built on the single reservoir, practically, ESN can be built with multiple reservoirs to achieve better performance. A shared reservoir modular ESN was designed with dividing the neural state into multiple independent output weight modules of subspace, then merging the data belonging to every subspace in one reservoir (Chen et al., 2010). Variational mode decomposition was utilized to allocate the data to different reservoir (Han et al., 2019). A Volterra filter structure with principal component analysis is presented to decrease the number of effective signals transmitted to the output layer (Boccato et al., 2012). Broad-band Echo State Network was proposed to determine reservoir number by limiting the unsupervised learning algorithm of Boltzmann machine (RBM) in order to fully reflect the dynamic characteristics of a class of multivariate time series (Yao and Wang, 2019). A decoupled Echo State Network was proposed which uses lateral

suppression units to represent decoupling of states before input to multiple reservoirs (Xue et al., 2007).

3. Designs of regularization and training phase

   According to the definition of Echo State Networks, during the training process, only output weights need to be learned, so many efficient training methods have proposed. Tikhonov regularization method was proposed to train output weights and the second-order proportional-integral-derivative feedback was applied to minimizes the prediction error (Chew et al., 2015). A random matrix theory based first theoretical performance analysis of the training phase of large dimensional linear ESNs is proposed (Couillet et al., 2016). The variational Bayesian framework is proposed to train ESNs with "delay & sum" output weight adaptation and automatic regularization (Shutin et al., 2012). The unsupervised learning algorithm of ESN was presented by addressing the unclear ground truth problem with evolution strategy, where optimizing a real-valued vector representing the plastic weights of the network (Devert et al., 2007).

   The ill-posed problem usually causes the leak of training samples, especially when the number of training samples less than the size of hidden layer. A hybrid regularized ESN was proposed when a big amount of unknown readout weights, the unbiasedness and sparse $L_{\frac{1}{2}}$ regulation and L_2 regulation were employed, where the L_2 regulation has the ability to shrinking the amplitude of the output weights (Xu et al., 2018). The Laplacian eigenmaps ESN was proposed to calculate output weights by low-dimensional manifold and estimate the reservoir manifold (Han and Xu, 2017). The adaptive Levenberg-Marquardt algorithm based ESN was used to get convergence and stable performance (Jiang et al., 2008). There were over-fitting issue existing widely, to solve this problem, some methods were proposed. A Bayesian Ridge ESN was proposed to avoid over-fitting and automatically identify the network architecture (Yang et al., 2018), with the same purpose, a leave-one-out cross-validation error ESN was proposed (Nguyen et al., 2018). A sparse recursive least squares algorithm ESN with online sequential adaptation was proposed, the network size was controlled by two norm sparsity penalty constraints of output weights (Yang et al., 2019).

   The network training algorithm is another research area to improve the ESN. Good hyper-parameter training is related to good validation. K-fold cross validation scheme was proposed, the time complexity was fixed and not scaling up with K, which

dominates by component (Lukoševičius and Uselis, 2019). To visualize the model system dynamics, a conversion of ESN internal layer output to a lower dimensional space was suggested, meanwhile, to get better generalization capabilities, the enforced regularization constraint was used (Løkse et al., 2017).

2.5 Summary

In this chapter, ANN is introduced in detail, especially the recurrent neural networks. In Sections 2.2 and 2.3, feedforward neural networks and recurrent neural networks are reviewed respectively. In Section 2.4, a special kind of network ESN in RNN is described in detail, along with the structure and training algorithm of the ESN, and the important parameters that affect the learning effect of the ESN.

Chapter 3. A modular reservoir topology for echo state network

3.1 Introduction

The reservoir topology of traditional echo state networks is usually generated using completely random, sparse connections and once generated will not be adjusted. Although ESNs with the traditional reservoir have successfully been applied to some nonlinear modelling problems, the traditional reservoir is not an optimal reservoir. The strong coupling within the reservoir topology of completely random sparse connection causes the states of the reservoir neurons converge, weakening the dynamics of the ESN and leading to poor network performance and poor stability of its performance.

There have been fruitful results on the topology design of ESNs: (1) networks constructed based on complex network theory: small-world and scale-free networks, as the two most famous complex networks, have been successfully introduced into the calculation of ESN reservoir, and experimental results show that these two structures have better topologies in terms of better test results, a wider range of spectral radius and comparable memory capacity to the random structure (Cui et al., 2012, Kawai et al., 2019, Kawai et al., 2017, Deng and Zhang, 2007); (2) low-complexity network based on the deterministic generation algorithms: the traditional random structure generation algorithm, although simple, has a certain randomness, which brings certain difficulties to theoretical research and analysis. To solve this problem, three simple and generative algorithm-determined structures have been proposed: simple cycle reservoir, delay line reservoir, and delay line reservoir with feedback connection (Rodan and Tino, 2011). The results show that, for some tasks, these three structures and the stochastic structure have comparable or even superior performance. Of course, different structures of ESNs have been proposed to solve different types of tasks, such as tree ESN (Gallicchio and Micheli, 2013), the distributed shift register network (Ma et al., 2014), and ESN with multiple sub reservoirs (Qiao et al., 2017). Some of the newer structures are complete reversals of the traditional stochastic structures with increased computational complexity, defeating the original purpose of ESN. However, it is undeniable that the study of the feasibility and application of new ESNs constructed using other topologies is of great theoretical interest.

Biological studies have shown that brain networks have hierarchy reservoir and modular topological characteristics, and the unique network topology can effectively enhance the

information processing capacity and robustness of the brain. In view of the above analysis, this chapter designs the reservoir topology from a structural bionic perspective. A modular reservoir topology with a class brain is proposed, and then a modular ESN (MESN) is constructed. Compared to the traditional completely randomly generated reservoir topology, the proposed reservoir topology is designed to reduce the coupling between neurons, thus enhancing the richness of the dynamical properties within the reservoir, improving the predictive performance of the ESN and its performance stability. It is also closer in function and structure to a real biological neural network.

The remaining part of this chapter is organized as follow. Section 3.2 gives the introduction of the reservoir topology in echo state networks and, in Section 3.3, the problem of random generated reservoir is raised. In Sections 3.4 to 3.6, the proposed modular small world ESN, the experiments to evaluate the proposed network and results are given respectively.

3.2 Topology of Echo State Networks

As the main connection in the ESN's dynamic memory storage, the reservoir plays a crucial role in the overall system performance. If a network topology can be found that can outperform the traditional random network structure, this will not only make a great progress in the theoretical study of ESNs, but also improve the computational efficiency of ESNs and make them more widely available. Therefore, the main focus of this chapter is on the topology of the reservoir network in ESNs. In this chapter, the topology of the reservoir network in ESNs is studied. By introducing complex and low-complexity network topology, it is expected that the modelling accuracy of the ESN can be improved. The network topology is briefly described.

3.2.1 Regular networks

A regular network is "regular" because each node has the same number of edges. The network exhibits high degree of regularity. A general regular network can be represented by a square "regular" lattice, where a square "regular" lattice means that each node in the network is connected to its nearest neighbour. At the same time, regular networks can also be represented in the form of a ring, a tree, etc. These lattices can represent real-world

things whose behaviour is closely related to that of their nearest neighbours (Arden and Hikyu, 1982).

Figure 3.1 shows a graph of a regular network where each node is connected to the four closest nodes. For example, nodes 2, 3, 10 and 9 form a cluster because they are all connected to node 1. This diagram visually shows the short path length and aggregation of the regular network. Two nodes in opposite positions (node 1 and node 6) have a low degree of aggregation and a large path length. While any two of the three nodes that can form a ring (node 1 and node 4) have a high degree of aggregation. This cross-connection enhances the connectivity of the network.

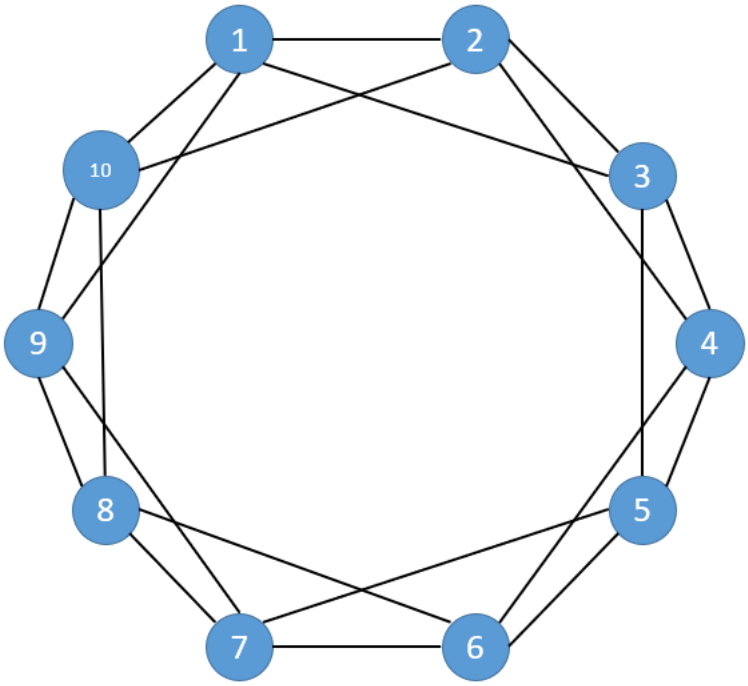


Figure 3.1 Typical structure of regular networks.

3.2.2 Random networks

The opposite topology to the regular network structure is the random network, its original purpose was to investigate probabilistically how the properties of graphs change as the number of connections increases. A random network is a network in which connections between nodes are formed randomly, but the final degree distribution of the resulting network is highly equal (the degree distribution is the distribution of the degrees of the nodes). The most typical random network is the Erdős–Rényi (ER) model (Erdös and

Rényi, 2011), which is a 'natural' based construction method: assume that there are n nodes and that the probability of each pair of nodes being connected is a constant $0 < p < 1$. The resulting network is the ER model network.

Random connections are generally characterised by short paths, where nodes in a network can reach the rest of the nodes via a short path. However, the degree of agglomeration of a random network is low, so there are hardly any particular nodes in a random network that have a high degree of agglomeration. Unlike regular networks, the degrees of nodes in a random network are generally Poisson distributed.

Figure 3.2 shows a graph representing a random network, where the network density is 30%. In this graph, node 1 and node 6, or node 3 and node 8, are randomly generated connections. Each node in the graph always has a shortest path to the remaining nodes.

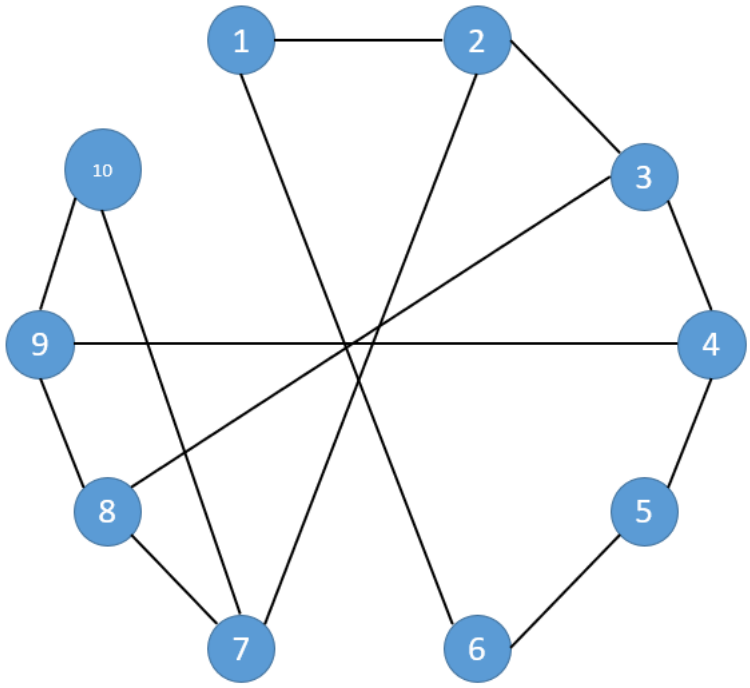


Figure 3.2 Typical structure of random networks (Probability = 0.3).

3.2.3 Small world networks

In the fields of mathematics, physics, and sociology, small-world networks are one type of the mathematical graphs. This type of graph is characterised by a minimum path length: although most of the nodes in the graph are not adjacent to each other, a node can be

reached from any node in just a few steps. A small world network can be a social network, which reflects the phenomenon of strangers meeting each other through people they know in common. If each person is considered to be a node in a small world network, and people know each other, it means that any two nodes in the small world network are connected, then the whole human race or the network of connections formed by a large number of people is a small world network, and this phenomenon in the real world is called the small world phenomenon. In the 1960s, Stanley Milgram conducted a famous experiment, which showed that it took very few transitions to get a letter from one person to another, from one person to a designated, but unknown person (Slater et al., 2006). The experiment is the first direct demonstration of the small-world effect and shows that small-world networks are characterised by a minimum path length, most nodes in the network can be connected by a short path.

The small world was proposed and another characteristic is the degree of agglomeration, it represents the probability that any two nodes are connected to each other by their respective neighbouring nodes (Watts and Strogatz, 1998).

The regular network described earlier has a high aggregation coefficient and its characteristic path length increases linearly with the number of nodes, *N*, in the network. During the conversion from a regular network to a random network, both the path length and the aggregation coefficient actually decrease, reducing to a minimum by the time it becomes a random network. However, this does not mean that large aggregation coefficients are necessarily accompanied by large path lengths, while small path lengths are accompanied by small aggregation coefficients, and small world networks exhibit large aggregation coefficients and small characteristic path lengths. Experiments have shown that the creation of a small number of short cuts can rapidly reduce the characteristic path length with little change in the aggregation coefficient. This is because the creation of a short cut affects the characteristic path lengths of the connected nodes and the path lengths of their neighbours, but has little effect on the aggregation coefficients of the network as a whole. In this way, a small number of short cuts can turn the entire network into a small-world network. Such a small-world network is also generally referred to as the Watts-Strogatz model or the WS model (Watts and Strogatz, 1998), and is the most typical model of a small-world network. Thus it can be said that for regular networks, the feature path length between any two nodes is long, but high degree of agglomeration;

random networks, with short feature path lengths but low agglomeration. Small-world network is a combination of both, with a small feature path length between nodes, close to a random network, and a high degree of agglomeration, comparable to a regular network.

As mentioned earlier, the small world model is a network between a regular network and a random network. It is therefore possible to be based on a fully regular network by disrupting and reconnecting the connections in the network with a certain probability, and then construct a small-world network model. The general construction method is as follows:

1. Start with a regular network. The N nodes in this network are arranged in a positive polygon, with each node being connected to the $2K$ nodes nearest to it, where K is a positive integer much smaller than N and $N \gg K \gg \ln(N) \gg 1$.
2. Select a node as node 1 in the network and number all nodes clockwise, and then sort the connections from each node clockwise as well. Then, the first connection of node 1 will have $0 < p < 1$ probability of being reconnected. The reconnection is done as follows: keep the end of node 1 unchanged and the other end of the connection is randomly replaced by another node in the network, but there is not more than one connection between the two nodes.
3. After reconnecting, do the same for nodes 2 and 3 (if any of these connections have already been reconnected, do not repeat them) until complete the loop.
4. Start again with the second connection at node 1 and repeat steps 2 and 3 until the loop is completed.
5. Start again with node 1 and repeat step 4 until all connections have been performed step 2.

Figure 3.3 shows a small-world network with a ring structure, where node 5 and node 9 are two nodes which are far away from each other but are randomly connected. It is clear from the figure that the overall path length of the small world network is short, this means that most of the nodes are not adjacent to any other nodes, but can reach any of the remaining nodes through a short path, while the degree of agglomeration remains high. It is also clear from Figure 3.3 that the small world network is a special network structure between a random network and a regular network. In such a network, information is

passed quickly and a small change in a few connections can drastically change the performance of the network.

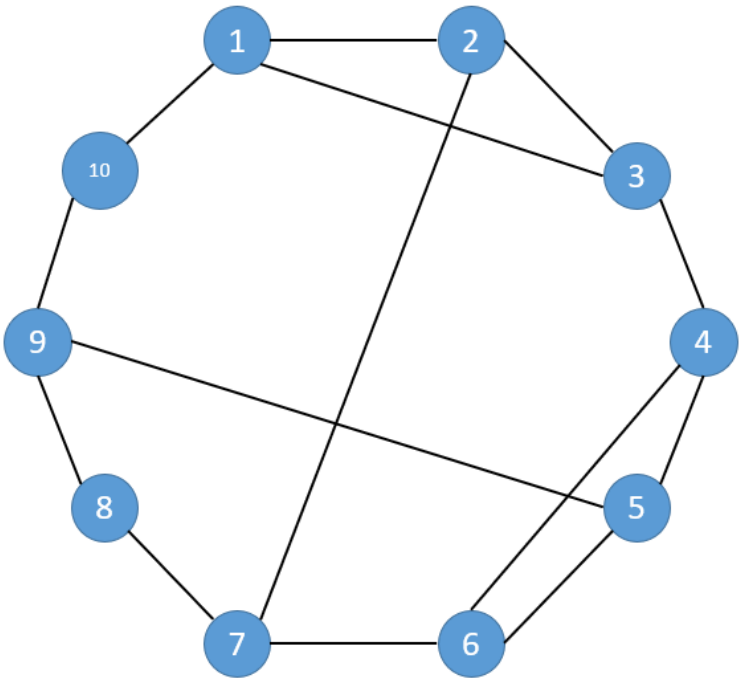


Figure 3.3 Typical structure of small world networks.

3.2.4 Low complexity topology

1. Simple cycle reservoir

A new reservoir templates with fixed topology is a simple cycle reservoir (SCR). In SCR, the units are organized in a cycle. Nonzero elements of W are on the lower sub-diagonal $W_{i+1,i} = r$ and at the upper-right corner $W_{1,N} = r$, where r is the weight of all the feedforward connections. Besides, in case of SCR, all input connections have the same absolute weight value $v > 0$, the sign of each input weight is determined randomly by a random draw from Bernuolli distribution of mean ½. Figure 3.4 shows the basic topology of simple cycle reservoir.

2. Delay line reservoir

The Delay line reservoir (DLR) is composed of units organized in a line. Non-zeros elements are only on the lower sub-diagonal of the reservoir matrix $\boldsymbol{W}$, that is $W_{i+1,i} = r$ for $i = 1 \ldots N - 1$, r is the weight of all the feedforward connections. Figure 3.5 shows the construction of DLR.

3. Delay line reservoir with feedback connections

Delay line reservoir with feedback connections (DLRB) has the same structure as DLR but each reservoir unit is also connected to the preceding neuron. Elements on the lower and upper sub-diagonals are non-zeros, $W_{i+1,i} = r$ and $W_{i,i+1} = b$, where r is the weight of all the feedforward connections and b is the weight of all the feedback connections. Figure 3.6 shows the topology of DLRB.

All these three low complexity topologies belong to regular networks.

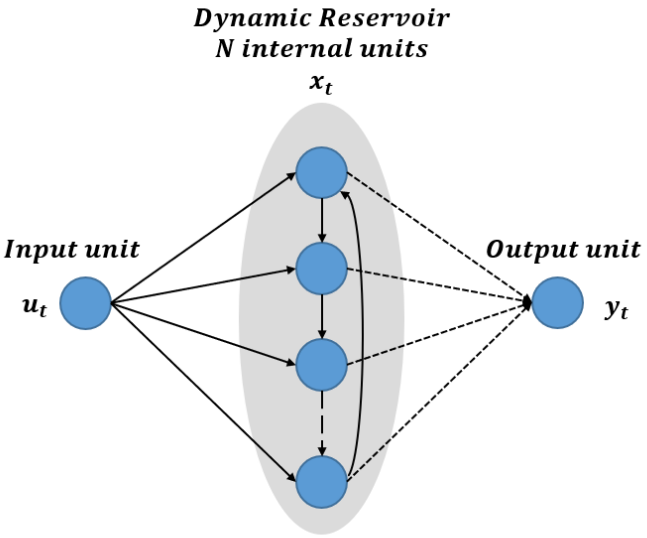


Figure 3.4 Topology of simple cycle reservoir.

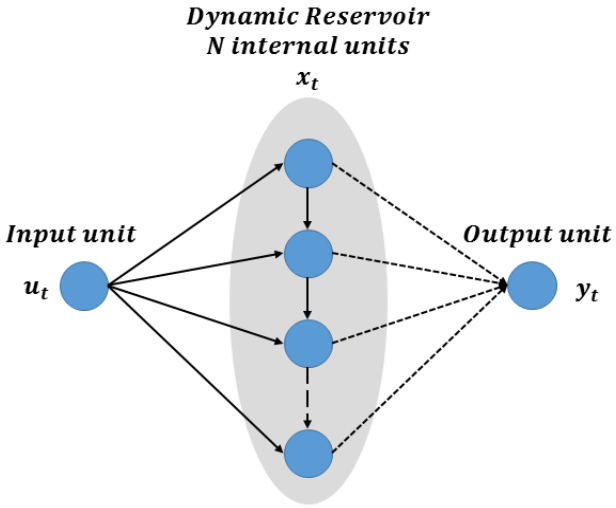


Figure 3.5 Topology of Delay line reservoir.

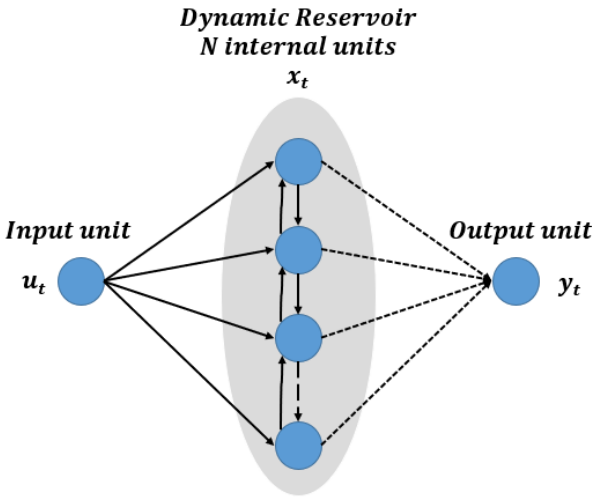


Figure 3.6 Topology of Delay line reservoir with feedback connections.

3.3 Problems with random reservoir topology

The traditional reservoir topology is generated completely randomly, and even when the reservoir size, sparsity and spectral radius are taken to be the same, there can be still differences in the prediction performance of the ESN models. In particular, as the number of reservoir neurons increases, the variability in connectivity within the reservoir becomes more pronounced, resulting in poorer network performance and its stability deteriorates. In order to investigate above problem, Mackey-Glass time series (Mackey and Glass, 1977)

is utilized which the details of this case study is given later. The set of parameters for the ESN models is as follows: reservoir size $N = 500$, spectral radius $\rho = 0.9$ and sparse density $D = 0.1$. Observations were carried out under the constraints of reservoir size, spectral radius and sparse density, 50 randomly generated ESNs were tested for the Mackey-Glass problem.

Figure 3.7 gives the mean squared error of the 50 tests performed on the Mackey-Glass problem by the ESNs. As can be seen from Figure 3.7, the test mean square error variation curve for the 50 randomly generated ESNs for the Mackey-Glass problem fluctuates considerably. There is a difference of three orders of magnitude between the maximum test MSE and the minimum test MSE (from 3.57×10^{-2} to 9.02×10^{-5}). The simulation results illustrate that there is a large variability in the performance of ESNs for solving the same problem, even when the key network parameters are the same. In practice, ESNs often require extensive testing to obtain a better performing network structure, thus limiting the application of the network. Therefore, the traditional stochastic sparse reservoir topology is optimally designed to improve the predictive performance of the ESN and its performance stability, which is beneficial to promote the use of ESN.

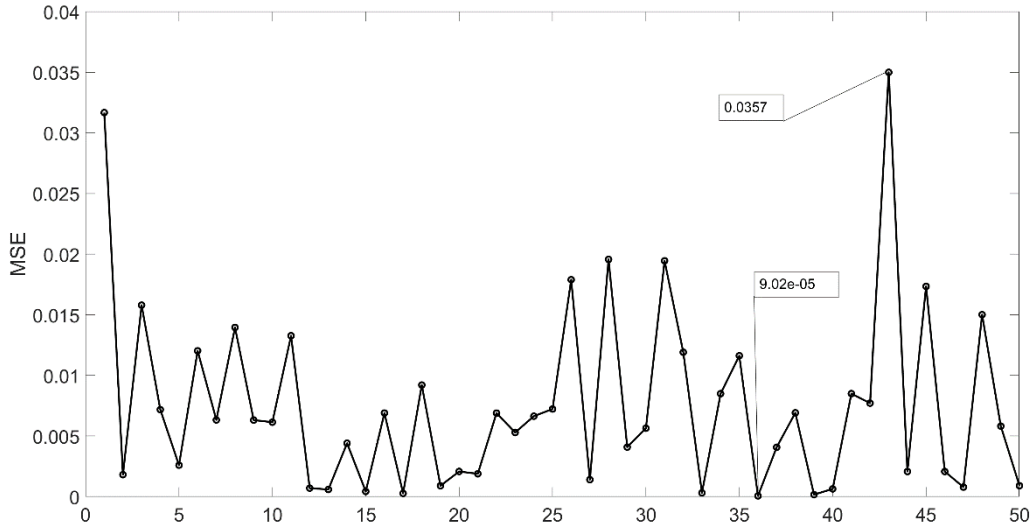


Figure 3.7 MSE of 50 randomly generate ESNs.

3.4 Modular echo state network

From the previous research, the small world topology can enhance the echo state property in echo state network (Kawai et al., 2019), meanwhile, the small world topology can

propagate input signal to the output nodes efficiently, this is similar as the ESN using a real human cortical connectively, but this is not enough. By studying structural brain networks at the macro-level of brain regions and at the micro-level of neurons, it has been found that brain networks have an important topological feature, named modular (Sporns et al., 2005). The small world topology modular-based echo state network is proposed.

3.4.1 Topology of human brain

With the development of brain science, cognitive science, biological anatomy and other disciplines becoming more and more mature, the brain's structure and working mechanisms have gradually been revealed, providing a richer theoretical basis for the study of structural design of neural networks. Figure 3.8 shows a functional separation graph of human brain, it can be found that human brain is divided into different zones depending on its functions. Neurons within the same module generally have similar properties and functions, whereas neurons within different modules have different properties and functions, and neurons between modules have sparsely connected (Sporns et al., 2007). From the previous investigation, the modular structure of brain can enhance the computational ability and robustness of human brain (Kaiser, 2011).

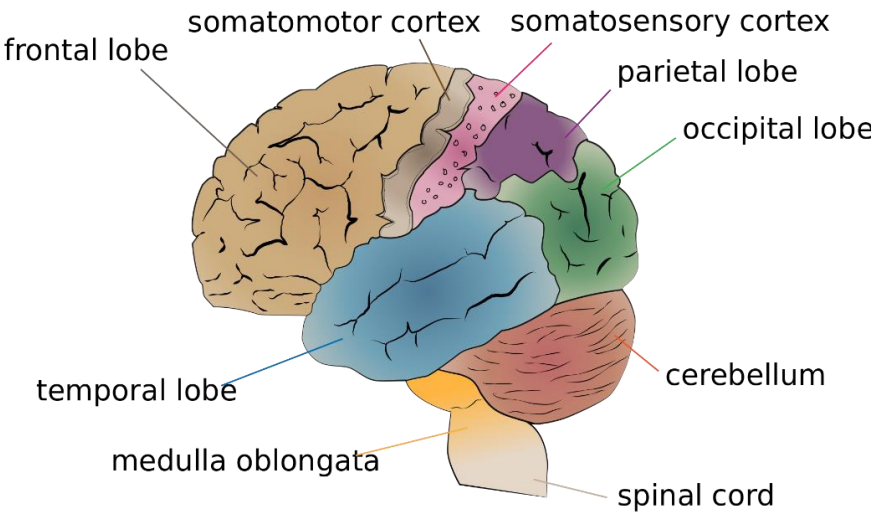


Figure 3.8 Human brain function separation.

Traditional ESNs use a randomly generated reservoir topology with a high degree of coupling between reservoir neurons. The high degree of coupling between reservoir neurons and low heterogeneity limit the network performance and its stability. Therefore, from the perspective that structure determines function perspective, the modular topological features of the brain network were simulated according to the structural

bionics principle (Ma and Chen, 2013), the design of the reservoir topology of the ESN is carried out in this chapter.

3.4.2 Modular echo state network design

A high performance ESN model is designed from a structural bionics perspective and it closely resembles the neuronal connectivity pattern of the brain. By modelling the modular connectivity of neurons at cerebral cortex, a modular reservoir topology of the brain, which weakens the coupling between reservoir neurons and enriches the reservoir to improve the overall performance of the network, is proposed.

The brain-like modular reservoir consists of multiple modules of small world networks, with each neuron having strong connections to neighbouring neurons in the same module and weakly connected to relatively distant neurons, and there are only sparse connections between different modules. This enables partial decoupling of neurons within the reservoir and enriches the dynamics of the network. Figure 3.9 shows a brief graph of MESN with 4 modules.

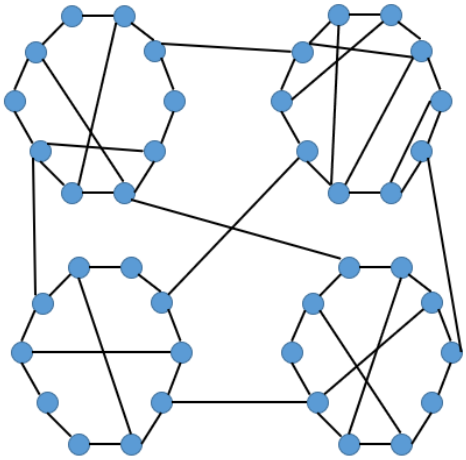


Figure 3.9 The diagram of modular echo state network.

The modular ESN can be established as following:

1. The size of reservoir for MESN is initially set at N based on the actual problem and the module number is m, so the reservoir size N_m of each module is:

$$N_m = \frac{N}{\sqrt{m}} \tag{3.1}$$

2. Generate each module reservoir with Watts-Strogatz small world network model with the parameters N_m, K_i, p_i, $i = 1 \ldots m$, K_i represents that each node in the their module connects with nearest K_i nodes by $N_m \gg K_i \gg \ln(N_m) \gg 1$ and p_i is the probability that each node reconnects with distant nodes in same module, where $0 < p_i < 1$. In order to increase the differences between the modules, K_i and p_i are selected differently in their range. In each sub-reservoir, a regular world of N_m is constructed, where the nodes are arranged in a ring-shaped pattern and each node is connected to its nearest K_i nodes, the total number of connections is $N_m \times K_i$, and then, the sub-reservoir is sparse. Each connection is rewired to another randomly selected node with the probability p_i, and the connection weights are generated by uniform distribution in the range [-0.5,0.5].
3. Sparsely reconnect the nodes from different modules by the similar method using in the WS small world model, the reconnection sparse density is p_r, it remains same among all modules.

3.5 Experiments

In this section, in order to evaluate the modular ESN performance and stability, three modelling cases are used, they are Mackey-Glass time series problem, multiple superimposed oscillator time series, and modelling of a water tank. The fitness function to minimize is the Mean Square Error (MSE) of the ESN during the training process:

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \tag{3.2}$$

where y_i and $\hat{y}_i$ are target value and predicted value at time i respectively, and n is the number of samples.

3.5.1 Mackey-Glass time series prediction

The Mackey-glass time series prediction problem is a widely used benchmark problem in nonlinear dynamic modelling. It is represented by the following equation:

$$\frac{dx(t)}{dt} = \frac{ax(t-\tau)}{1+x(t-\tau)^{10}} - bx(t) \tag{3.3}$$

where $a = 0.2, b = 0.1$ and $\tau = 17$. It has a chaotic attractor when $\tau > 16.8$ (Mackey and Glass, 1977). In this study, 3000 samples are generated by using the 4th order Runge-Kutta

method with random initialization and the sampling time used here is 0.1s. The initial 1000 samples are used as training data, the next 1000 samples are as testing data, the final 1000 samples are selected as unseen validation data. In the training samples, the first 100 samples are initialized to warm up the reservoir when the reservoir is newly-built. The regularization coefficient used in this training process is 1×10^{-8}.

In the Mackey-Glass time series model, the long range predicted time series model is of the following form:

$$\hat{y}(t) = f\big(\hat{y}(t-1)\big) \tag{3.4}$$

where $\hat{y}(t)$ is the predicted value at time step t.
The equations represent long range prediction as the model prediction is recursively used as the model input.

3.5.2 Multiple Superimposed Oscillator time series

The multiple superimposed oscillator time series (MSO) contains multiple sine waves of different frequencies superimposed on each other. MSO problem requires that the state variables in the reservoir of the network can represent different frequency sine waves at the same time (Strauß et al., 2012). Since the conventional ESN uses a completely random reservoir topology and it is prone to overall neuronal synchronization phenomenon. Therefore, the MSO time series prediction problem has been a difficult problem for traditional ESNs to solve. The MSO time series data are generated by summing up several different frequency sine wave functions, which is shown below:

$$y(t) = \sum_{i=1}^{m} \sin(\alpha_i t) \tag{3.5}$$

where *t* is the discrete time and *m* represents the number of summed sine waves. The forecast complexity increases as the number of summed sine waves increases. In this test, an MSO containing five sine waves with different frequency is selected to verify the proposed brain-like hierarchical modular reservoir topology. This MSO is given in Eq (3.6), where $\alpha_1 = 0.2, \alpha_2 = 0.311, \alpha_3 = 0.42, \alpha_4 = 0.51, \alpha_5 = 0.63$.

$$y(t) = \sin(0.2t) + \sin(0.311t) + \sin(0.42t) + \sin(0.51t) + \sin(0.63t) \tag{3.6}$$

In this case study, 2000 samples are generated. The initial 1000 samples are used as the training data, the next 500 samples are used as the testing data, and the final 500 samples

are selected to be the unseen validation data. The dynamic model for MSO time series data is one step prediction:

$$y(t) = f\big(y(t-1)\big) \tag{3.7}$$

3.5.3 Modelling of a water tank

ESN is used here to model a water tank shown in Figure 3.10. An inlet flow and an outlet flow are connected to the tank. The water level in the tank can be controlled by manipulating the inlet water flow rate. The mass balance equation of the tank can be represented as follows (Zhang and Morris, 2000):

$$\frac{dV}{dt} = Q_i - Q_o \tag{3.8}$$

where Q_i, Q_o and V denote the inlet and outlet water flow rate and water volume in the tank, respectively.

The outlet water stream rate, Q_o, is related to the tank level h by the following equation:

$$Q_o = k\sqrt{h} \tag{3.9}$$

where k is a constant value representing the valve opening.

The water volume in the tank is related to the water level h expressed by the following equation:

$$V = \pi h\left[r^2 + \frac{hr}{\tan\theta} + \frac{h^2}{3(\tan\theta)^2}\right] \tag{3.10}$$

where θ represents the angle between tank wall and horizontal plane, and r denotes the tank bottom radius. Summarizing the above equations, the dynamic water level in the tank can be expressed as:

$$\frac{dh}{dt} = \frac{Q_i - k\sqrt{h}}{\pi\left[r^2 + \frac{2hr}{\tan\theta} + \frac{h^2}{(\tan\theta)^2}\right]} \tag{3.11}$$

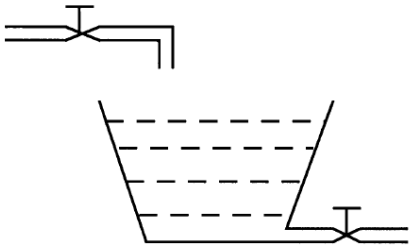


Figure 3.10 A conic water tank.

A simulation program is developed to simulate the water level in the tank. The parameters used in the simulation are $r = 10\text{cm}$, $k = 34.77\text{cm}^{2.5}/\text{s}$ and $\theta = 60^{\circ}$. The sampling time used in the simulation program is 10s. Refer to Eq 3.10, the water level in the tank is clearly not linearly related with inlet flow rate.

In this case study, 2220 samples are generated from simulation. The first 740 samples are used to train the networks, the next 740 samples are testing data and the rest 740 samples are used as the unseen validation dataset. Noise with the distribution of $N(0, 2)$ was added to the input flow rate to represent the effects of measurement noise. The developed nonlinear dynamic model for all methods is of the following form:

$$\hat{y}(t) = f(\hat{y}(t-1), u(t-1))$$

where $\hat{y}$ is the predicted tank level, *u* is the inlet flow rate, and *t* is the discrete time.

3.6 Results

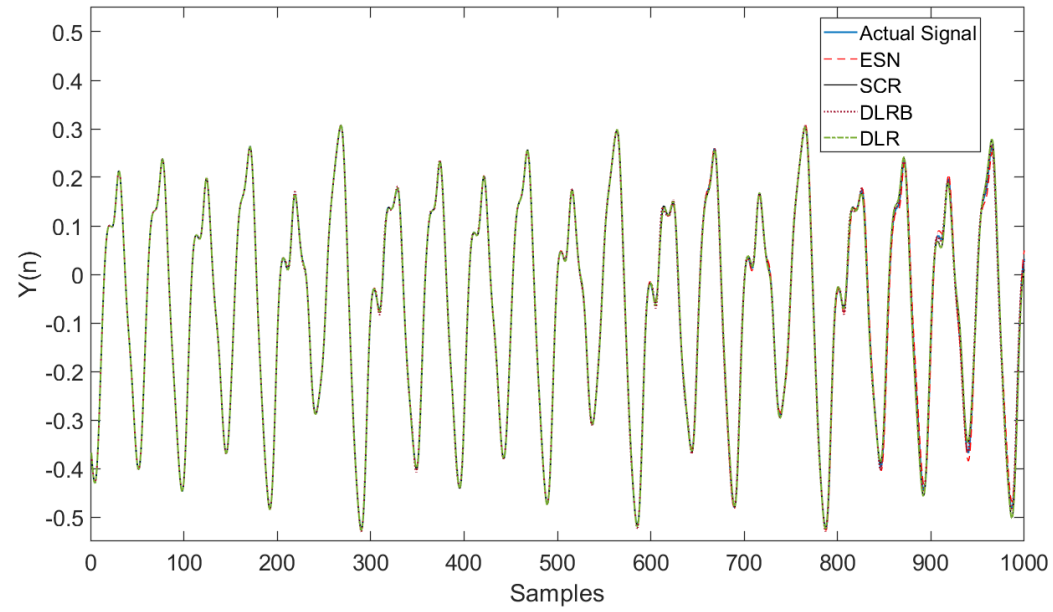
To evaluate the modular ESN, conventional ESN, SCR-ESN, DLR-ESN, DLRB-ESN, small-world ESN are developed to model the cases described above. For all these ESN models, every reservoir node uses the activation function $\tanh(x) = \frac{\sin h(x)}{\cos h(x)} = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$; the input and backwards weights $\boldsymbol{W^{in}}$, $\boldsymbol{W^{back}}$ are fully generated as uniform distribution random values in the range [-0.5,0.5]. For each ESN reservoir topology, 20 ESNs are training with the specific random parameters in the range illustrating below with the training data, then the most accurate model on with the test data is selected. Finally, the developed models are evaluated on the unseen validation data. The following procedures are compared:

- Conventional ESN: In the conventional ESN, reservoir size and spectral radius factor, are selected randomly in the ranges of [1-1000], and [0-1.5], respectively, and the sparse density of reservoir matrix $\boldsymbol{W}$ is fixed to 0.1 and the leak rate remains 0.3.
- For SCR, DLR and DLRB architectures ESNs, the model is defined by reservoir weight r and reservoir size N (for DLRB network, the feedback connection is also needed to specify), and leak rate is fixed to 0.3. So the reservoir size N and reservoir weight r are generated randomly in the ranges of [1-1000] and [0-1], respectively.
- Small world ESN: the small world model reservoir is related to reservoir size N, the number that each node connect with neighbor nodes K and reconnection probability ρ. K is under the rule $N \gg K \gg \ln(N) \gg 1$, N and ρ are in the ranges of [1-1000] and [0-1].
- Modular ESN: In each module, the reservoir is generated by small world network described above, for the nodes reconnection between modules, the probability is p_r, which is in the range [0,1].

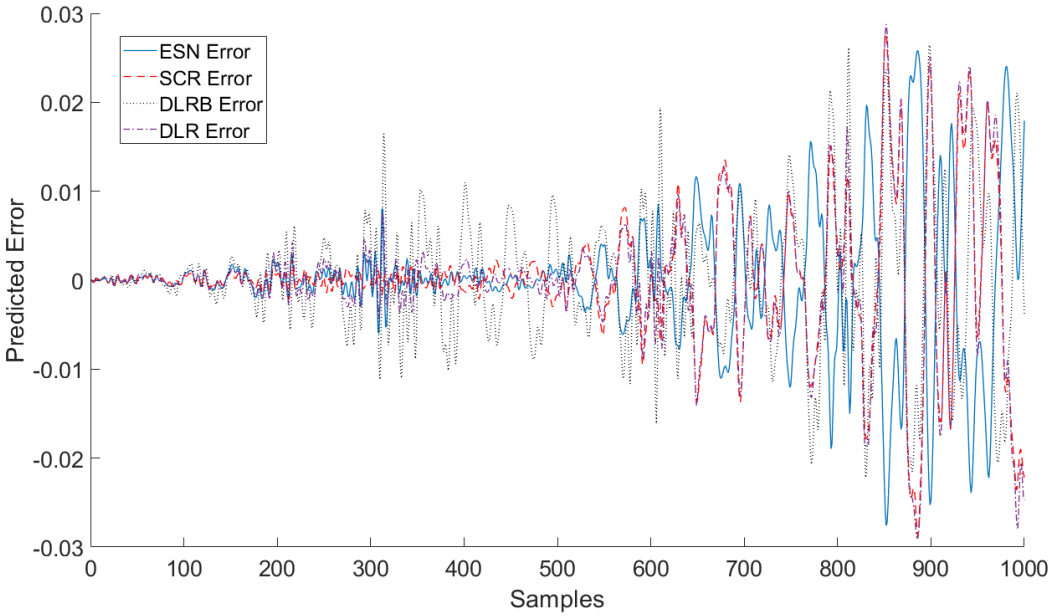
3.6.1 Mackey-Glass time series prediction

Figure 3.11 and Figure 3.12 show the validation results for the best of 20 random generated ESNs selected by test data of each reservoir topology. Figure 3.11(a) shows the actual signal and predicted signal of Conventional ESN, SCR, DLRB and DLR, while Figure 3.11(b) illustrates the prediction errors of the different structure reservoir ESNs. Figure 3.12(a) shows the actual signal and predicted signal of Conventional ESN, modular small world ESN, modular sample cycle reservoir and small world ESN, while Figure 3.12(b) illustrates their prediction errors. From Figure 3.11(a) and Figure 3.12(a), it can be found that all the models are fitted the actual signal in first 500 samples. This means that ESNs have good performance on predicting Mackey-Glass time series data for at least the first 500 samples. After 500 samples, the prediction errors start to increase as shown in Figure 3.11(b) and 3.12(b). From Figure 3.11(b), it can be found that the degree of undulation of all curves (ESN, SCR, DLRB and DLR) are similar except at some peaks point of the DLRB curve. This means there is no order of magnitude difference in the modelling ability of these models. Form Figure 3.12(b), compared to the conventional ESN and small world ESN, the test error curve of MSM-ESN for the Mackey-Glass time series data are smoother and increases gradually as the prediction horizon increases, but it

is always limited to a small range. Therefore, according to Figure 3.11 and Figure 3.12, it can be seen that the modular ESN models have a higher prediction accuracy than the traditional ESN models for the Mackey-Glass chaotic time series, especially the modular small world ESN, the validity of the reservoir topology of proposed modular brain-like structure is further verified.

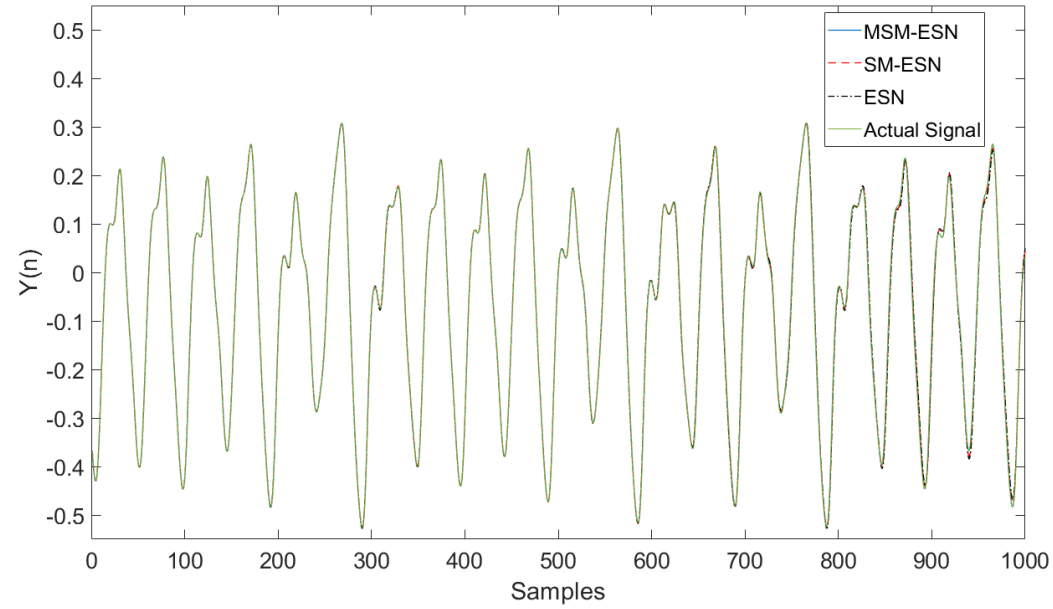


(a)

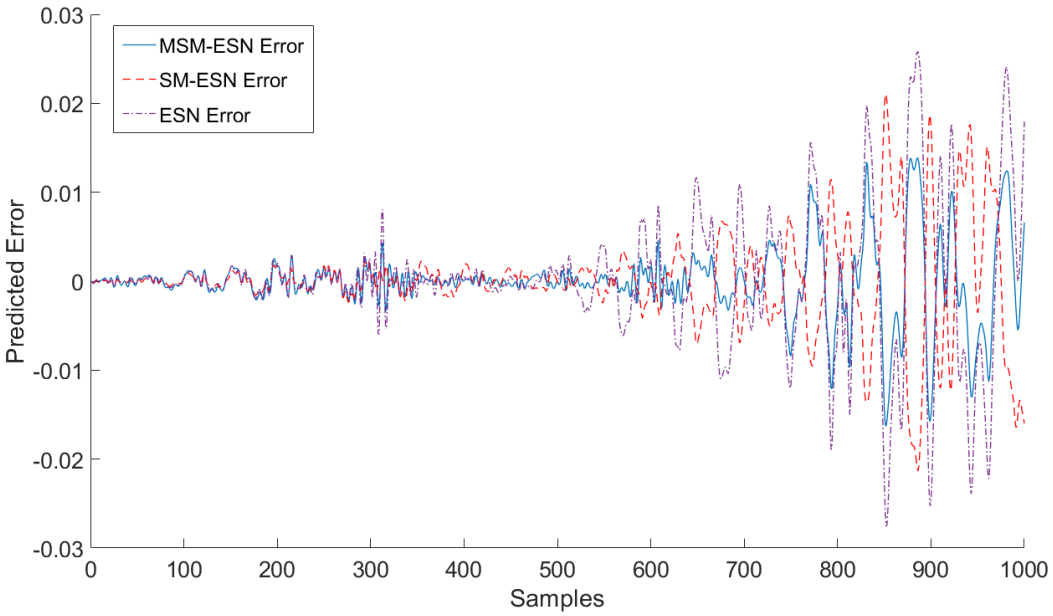


(b)

Figure 3.11 The predictions results of ESN, SCR, DLR and DLRB for Mackey-Glass Time Series data. (a) The actual and predicted signal. (b) The predicted errors.



(a)



(b)

Figure 3.12 The predictions results of MSM-ESN, SM-ESN and ESN for Mackey-Glass Time Series data. (a) The actual and predicted signal. (b) The predicted errors.

To better visualise the dynamics within the reservoir of the MSM-ESN, a randomly selected node in each module of the MSM-ESN reservoir (where $m = 9$) is used to observe the state output of the reservoir nodes. The response curves of the 9 reservoir nodes in the MSM-ESN are given in Figure 3.13. As can be seen in Figure 3.13, all the nodes have been activated and the response curves are similar with the desired signal, this

shows that the nodes within the MSM-ESN modules are able to remember historical information and learn the dynamic properties of the Mackey-Glass time series data well when stimulated by input signals. The simulation results illustrate the excellent dynamic properties of the MSM-ESN model.

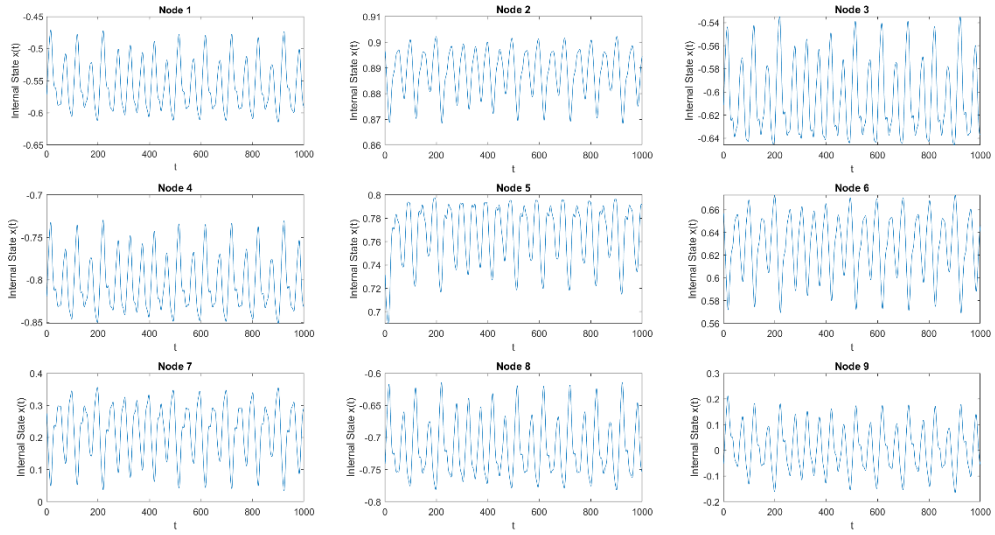


Figure 3.13 Response curves of 9 selected nodes.

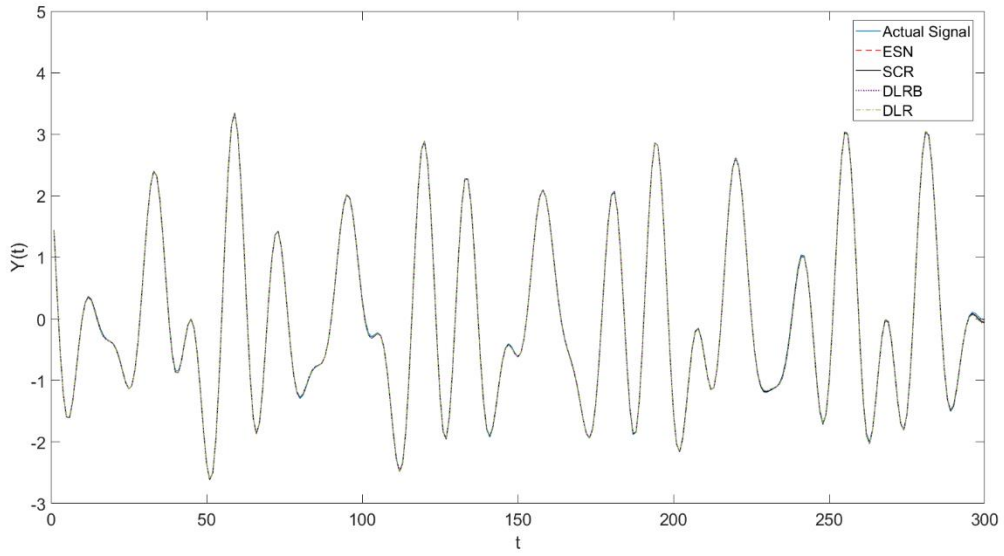
Table 3.1 gives the comparison of the MSM-ESN with the conventional ESN, SCR-ESN, DLR-ESN, DLRB-ESN and small world ESN on the Mackey-Glass time series prediction problems. 20 independent simulations with random reservoir size N and random spectral radius ρ are processed, the average and variance of test MSE are taken for inclusion in Table 3.1. It should be noted that the modular ESN achieve the better prediction capability and the distribution of each independent simulations result is more stable.

Table 3.1 Average and Variance of MSE comparison for Mackey-Glass time series prediction.

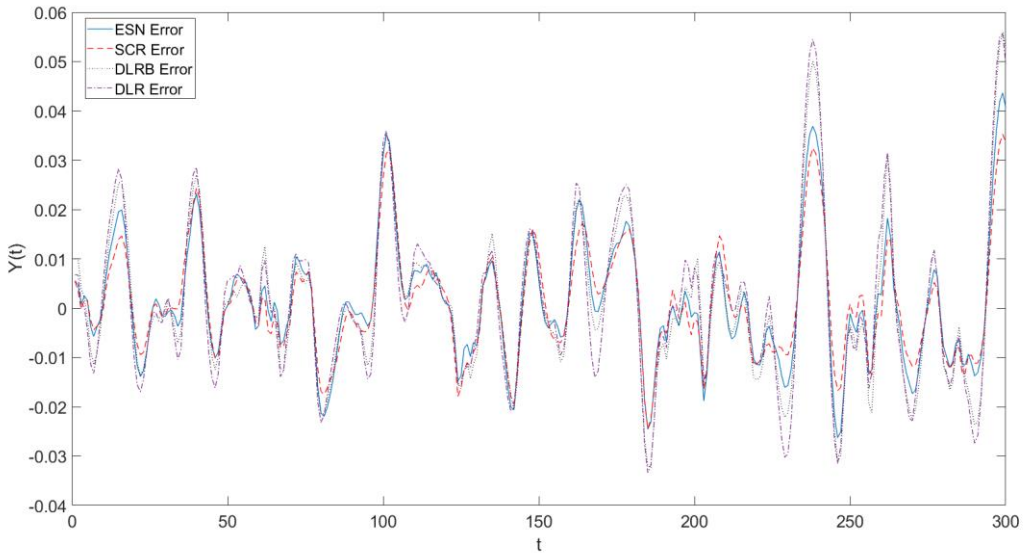
Reservoir Topology	Mean	Var
MSM-ESN	2.31×10^{-4}	4.71×10^{-7}
SM-ESN	9.43×10^{-4}	3.67×10^{-6}
ESN	1.12×10^{-3}	9.02×10^{-6}
SCR	1.45×10^{-3}	1.27×10^{-5}
DLRB	3.64×10^{-3}	3.16×10^{-5}
DLR	2.31×10^{-3}	3.56×10^{-5}

3.6.2 Multiple Superimposed Oscillator time series prediction

In the MSO time series task, to determine the projected accuracy of modular small world ESN, SM-ESN, conventional ESN, SCR-ESN, DLR-ESN and DLRB-ESN are selected for comparation. The modelling results of validation data are shown in Figure 3.14 and Figure 3.15. From Figure 3.14(b), the prediction error of SCR-ESN is smaller than the other topology reservoir. This is different from the result of Mackey-Glass time series data and this shows that SCR topology reservoir has better ability to handle the characteristic of multi-oscillator superpositions than completely random reservoir at similar reservoir size. Figure 3.15(b) shows the prediction errors of MSM-ESN and it can be found that the prediction errors are limited to fluctuations between [-0.01, 0.01] and has remained relatively stable. Figure 3.14 and Figure 3.15 show that the prediction performance of MSM-ESN is better than other compared ESN models. To further test the stability of the network performance, 50 tests were performed on each of the MSM-ESN and SCR-ESN networks and the best performing 20 tests are selected for comparison. The results were compared in Figure 3.16. As can be seen from Figure 3.16, compared with SCR-ESN, the MSM-ESN proposed in this chapter has better prediction accuracy, while the network prediction performance is more stable.

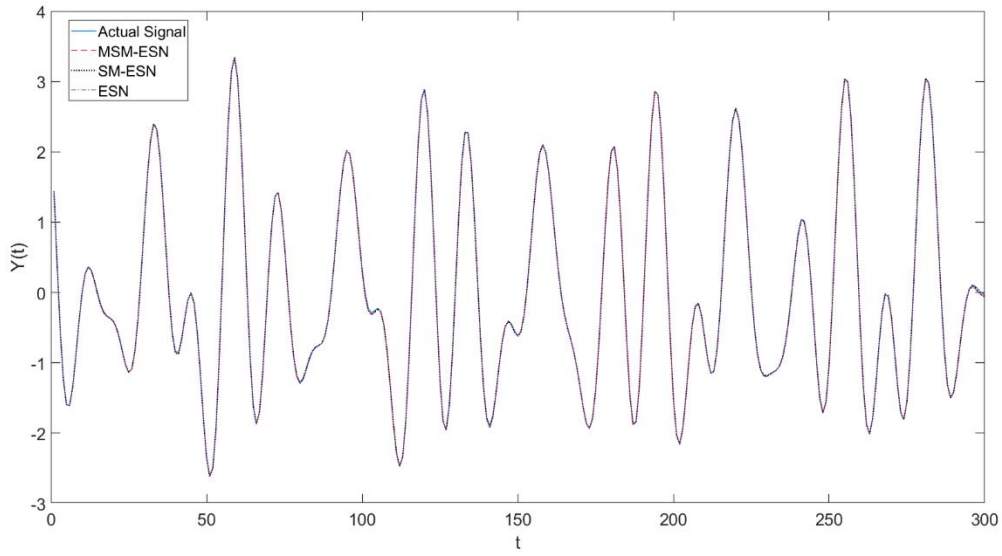


(a)

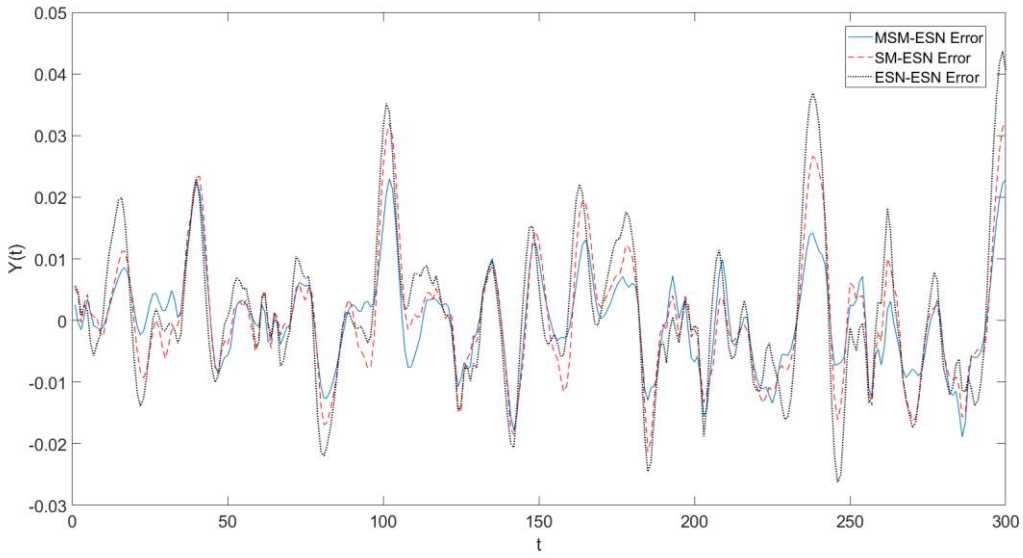


(b)

Figure 3.14 The prediction results of ESN, SCR, DLR and DLRB for MSO time series data. (a) The actual and predicted signals; (b) The predicted errors.



(a)



(b)

Figure 3.15 The prediction results of MSM-ESN, SM-ESN, ESN for MSO time series data. (a) The actual and predicted signals; (b) The predicted errors.

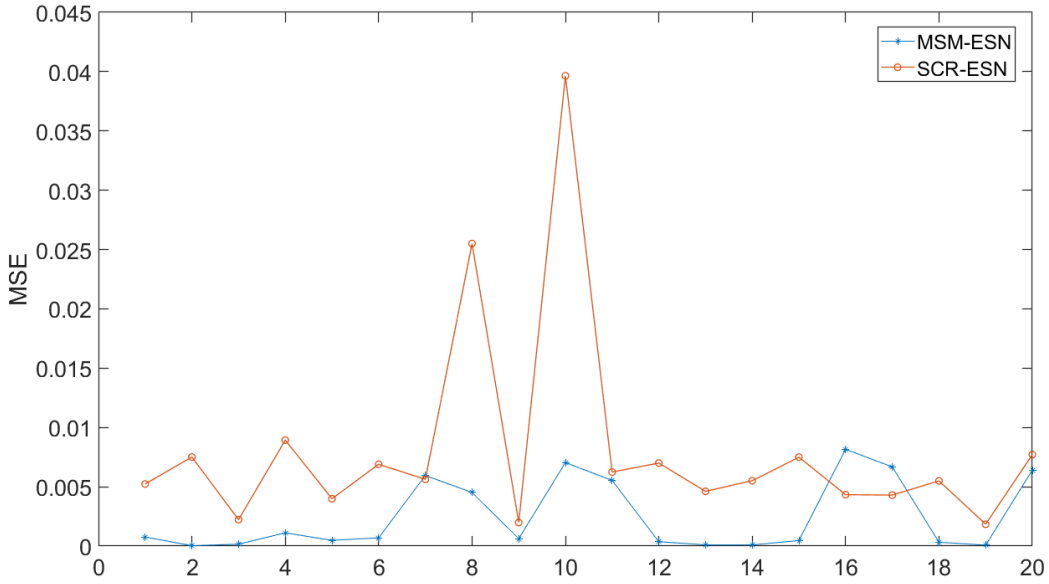


Figure 3.16 The comparison between MSM-ESN and SCR-ESN on MSO time series data.

3.6.3 Modelling of a Water Tank

To investigate the predicted accuracy of modular small world ESN on the water tank liquid level modelling, SM-ESN, SCR-ESN, conventional ESN, DLR ESN and DLRB ESN are used for comparison. Figure 3.17 shows an example of input data which is the water injection flow of tank, it is a step flow with noise. Figure 3.18 and Figure 3.19 show the validation results for the best model of 20 randomly generated ESN with different topology. Figure 3.18(a) shows the actual signal and predicted signal of Conventional ESN, SCR, DLRB and DLR, while Figure 3.18(b) illustrates the error between desired signal and predictions of the above different structure reservoir ESNs. Figure 3.19(a) shows the actual signal and predicted signal of Conventional ESN, modular small world ESN, small world ESN, while Figure 3.19(b) illustrates the errors between desired signal and predictions of above ESNs. From Figure 3.18(a) and Figure 3.19(a), it can be found, in general terms, the ESNs with different topology can model the liquid level accurately, only when the input flow rate start to decrease, the distance between predicted curves and actual signal curve increases. Among all the predictions, the red dashed-line in Figure 3.19(a) is significantly close to the actual signals. In Figure 3.18(b) and Figure 3.19(b), the errors are illustrated more obviously, when the trend of input flow rate changes or the huge change occurs in sudden, the errors increase dramatically, in the other period, the errors are in the range of [-0.5,0.5]. In Figure 3.19(b), at the point where the trend of

injection flow rate decreases, the error curve of MSM-ESN is much smoother than other two error curves. This also illustrates that the robustness of MSM-ESN is higher than other topology.

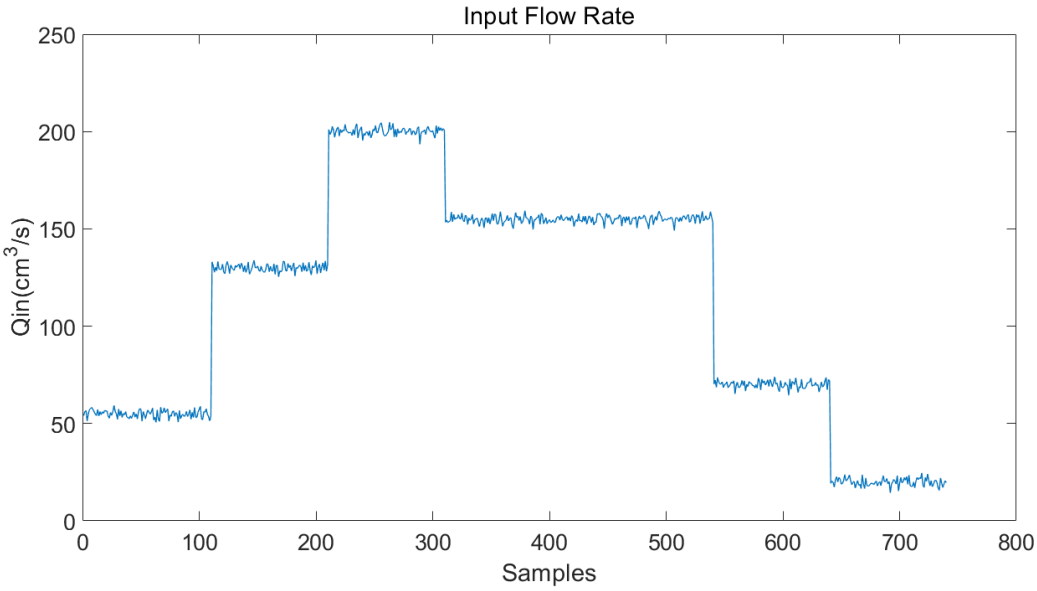
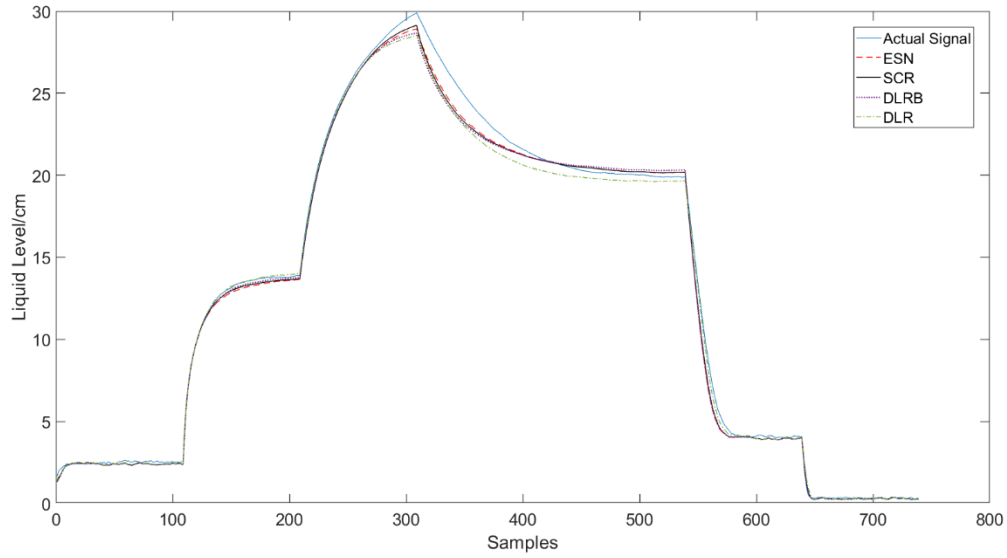
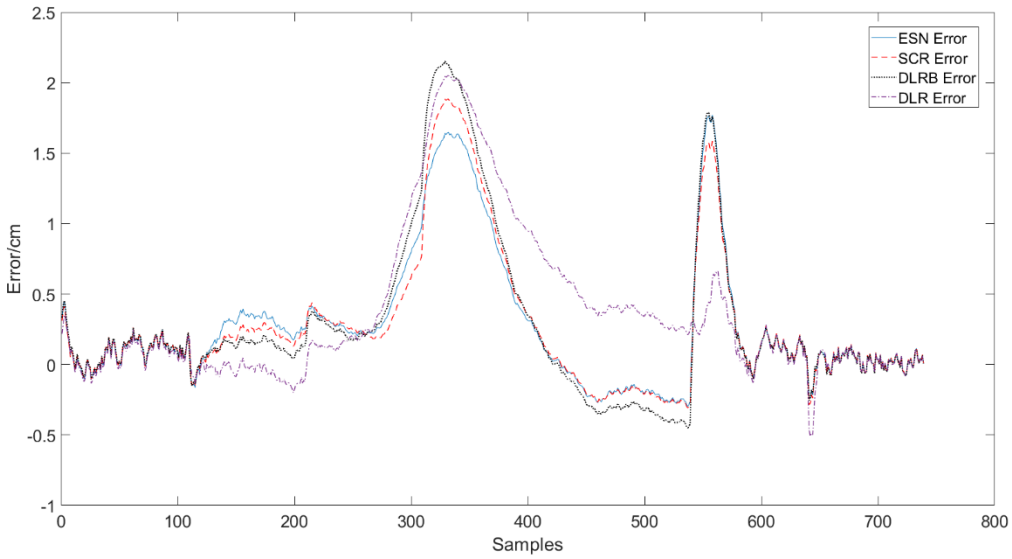


Figure 3.17 Input flow rate of the water tank.

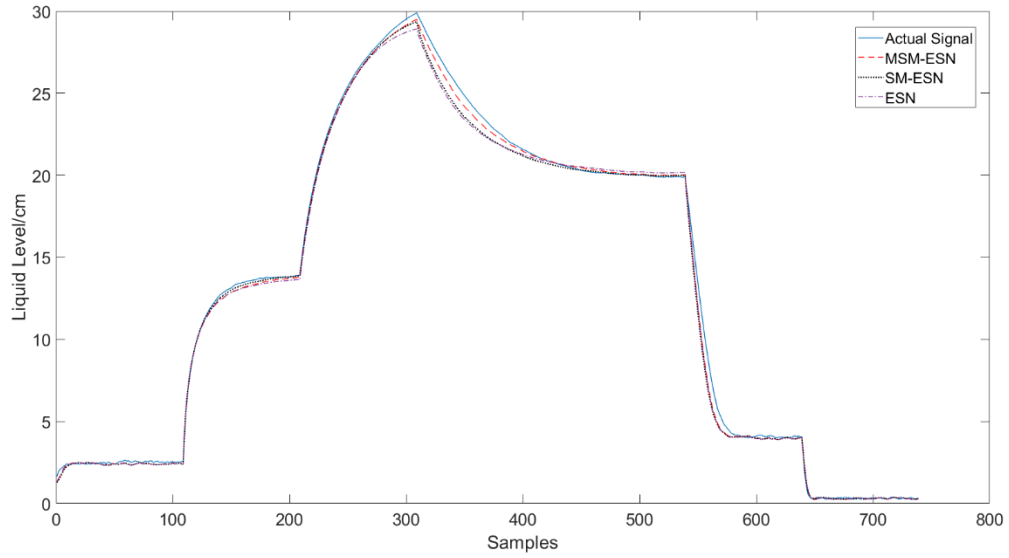


(a)

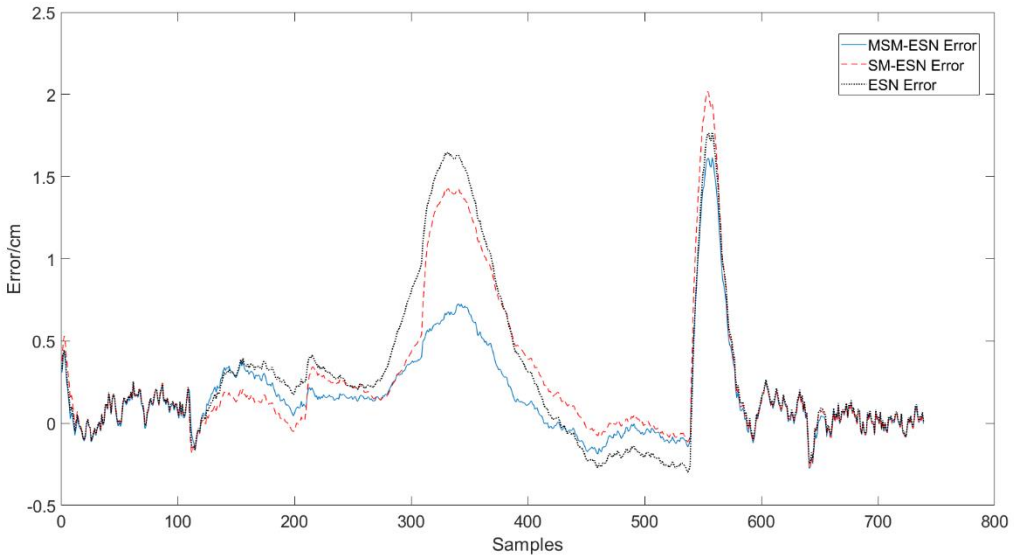


(b)

Figure 3.18 The predictions results of ESN, SCR, DLR and DLRB for water tank model. (a) The actual and predicted signal. (b) The predicted errors.



(a)



(b)

Figure 3.19 The predictions results of MSM-ESN, SM-ESN, ESN for water tank model. (a) The actual and predicted signal. (b) The predicted errors.

To better visualise the dynamics within the reservoir of the MSM-ESN, a randomly selected node in each module of the MSM-ESN reservoir (where $m = 4$) is used to observe the state output of the reservoir nodes. The response curves of the 4 reservoir nodes in the MSM-ESN are given in Figure 3.20. It can be found in the figure that the reservoir node has rapid and sufficient response to the input signal and feedback predicted output signal. The simulation results illustrate the excellent dynamic properties of the MSM-ESN model.

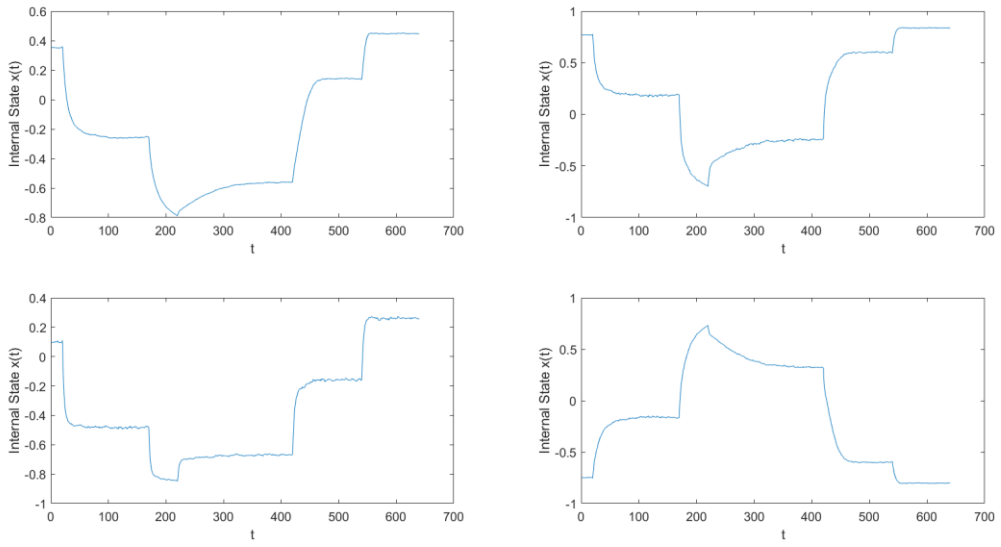


Figure 3.20 Response curves of 4 selected nodes in water tank model.

Table 3.2 gives the comparison of the MSM-ESN with the conventional ESN, SCR-ESN, DLR-ESN, DLRB-ESN and small world ESN on water tank liquid level data. 20 independent simulations with random reservoir size N and random spectral radius ρ are processed, the average and variance of test MSE are taken for inclusion in Table 3.2. It should be noted that the modular ESN achieve the better prediction capability and the distribution of each independent simulations result is more stable, the reason is that modular topology reduces the degree of coupling between reservoir nodes and improves network information processing.

Table 3.2 Average and Variance of MSE comparison of water tank model prediction.

Reservoir Topology	Mean	Var
MSM-ESN	0.3763	0.0655
SM-ESN	0.5325	0.0758
ESN	0.5568	0.0990
SCR	0.6362	0.1243
DLRB	0.8401	0.1498
DLR	0.6428	0.1013

3.7 Conclusions

This chapter analyses the modelling performance of ESN models with the traditional reservoir topologies and their lack of performance stability. From the perspective of structural bionics, a new reservoir topology of ESN, called modular reservoir topology, is proposed by mimicking the brain networks with modular topological features. The modular reservoir topology is designed to achieve partial decoupling of neurons within the reservoir, which enhances the richness of dynamical properties within the reservoir and thus improves the performance of ESN. Based on this, a modular small world ESN (MSM-ESN) is constructed and its performance is verified through extensive modelling experiments. Compared with conventional ESN, sample cycle ESN, delay line ESN and delay line with feedback connection ESN, MSM-ESN solves the Mackey-Glass chaotic time series prediction problem and the MSO problem with better prediction accuracy and stability. In addition, MSM-ESN can also show better information processing capability in the real world industrial simulation problem, laying a theoretical foundation for the application of ESNs to the modelling of complex non-linear dynamic systems.

Chapter 4 Genetic algorithm optimized echo state network

4.1 Introduction

In the last chapter, in order to overcome the problems with randomly generated reservoir topology, a modular small world reservoir topology is proposed, meanwhile, the minimum complexity perspective reservoir topology and small world topology have been tested at same time. However, because all of the aforementioned methods are deterministic and rely heavily on the user's experience and heuristics, suboptimal solutions are expected. In order to further enhance reservoir performance, evolutionary optimization approaches, which are well-known for their capacity to search globally for solutions to complicated problems, can be used to optimise the reservoir.

Evolutionary computation is an artificial intelligence technique that simulates the mechanisms of biological evolution in nature. It starts from a randomly selected set of solutions and iterates and evolves to optimise the current set of solutions until the result is optimal or satisfactory. Evolutionary computation is generally considered to include genetic algorithms, particle swarm optimization (Kennedy and Eberhart, 1995), ant colony optimization (Dorigo et al., 2006), cuckoo search (Yang and Deb, 2009), all of which are intelligent evolutionary methods designed to simulate a characteristic or behaviour of organisms. Genetic algorithms as a representative of evolutionary computing are adaptive probabilistic search techniques based on meritocracy. As a natural computational model with stochastic iterative evolution, the algorithm is robust, practical, efficient and widely applicable, incorporating selection, crossover and mutation mechanisms from natural evolutionary models.

Genetic algorithms and neural networks are important achievements in bionics, although they both simulate the way in which nature processes information, the principles are different. Genetic algorithms are derived from the mechanisms of biological evolution in nature, whereas neural networks abstract and simulate certain basic characteristics of the brain. As a result, there are significant differences in the real-time nature of information processing, by combining the strengths of each, the two methods can mutually enhance each other's capabilities, resulting in a more effective approach to solve problems. The global optimisation capability of genetic algorithms can effectively address the problem of gradient based network training that tends to fall into local minima. However, there are

some disadvantages with genetic algorithms. Firstly, due to the accuracy of the coding adopted in the chromosome, genetic algorithms can only give approximate optimal solutions but not exact values. Secondly, the initial parameters of the genetic algorithm are often chosen on the compromise between population size and convergence speed.

There are three different ways to find a good ESN with evolutionary algorithms: either as a topologically-based evolutionary process or as an evolutionary operator that works with stochastic parameters that create ESNs, or both simultaneously (Ferreira and Ludermir, 2010). The parameters used in the evolutionary algorithm are: reservoir size, leak rate, regularization parameter, spectral radius, activation function, topology and reservoir weights. The search space might be enormous if the evolutionary algorithm acts directly on the reservoir topology or reservoir connection weights. If the evolutionary method operates directly on the structure parameters such as reservoir weights, leak rate, spectral radius, the search space could be significantly smaller. There are several reported studies on using evolutionary algorithm to optimize ESN. A genetic algorithm was used to optimise the influence of spectral radius and settling time on the echo state networks (Venayagamoorthy and Shishir, 2009). A binary particle swarm optimization algorithm was used to optimize the readout weights matrix directly (Wang and Yan, 2015). An evolutionary multi-objective algorithm was used to address the trade-off between identifying dynamic systems with maximum accuracy and minimizing reservoir complexity (Roeschies and Igel, 2010).

Although the global search capability of genetic algorithms has been proven in practice, the problem of "premature convergency" still needs to be improved. In this chapter, to improve the global optimization capability of genetic algorithm and to improve the dynamic performance of ESN, an adaptative genetic algorithm ESN has been proposed. The key parameters of ESN is optimized by the adaptative genetic algorithm to search for more suitable parameters and prevent premature convergency of genetic algorithm.

The reminding parts of the chapter is organised as follows. Section 4.2 gives the detailed introduction to the genetic algorithm including coding, initialize population, selection, crossover, mutation and new proposed adaption method to update the crossover and mutation probabilities. In Section 4.3, the combination of adaptive genetic algorithm and

ESN has been given. In Section 4.4 and Section 4.5, the three modelling cases and the prediction results of the proposed method on these cases are illustrated.

4.2 Genetic Algorithm

A genetic algorithm is a probabilistic search algorithm created by Holland who used a coding technique to act on strings of numbers called chromosomes (Holland, 1992). It is an optimization and search algorithm that simulates the process of biological evolution and natural selection. An important feature of genetic algorithm is that it keeps the population evolving, so that even if an individual loses a useful trait at some point, it will be complimented by other individuals and will continue to develop. It is inherently implicitly parallel, spatially searchable and highly robust.

A genetic algorithm is essentially an iterative algorithm for reproduction, selection and evaluation based on population and genetic manipulation. Before searching, the solution to the problem is mapped to a solution set space, which is a population of a certain number of binary individuals. Starting with these populations representing potential solutions, the problem is to construct a fitness function based on the objective function of the problem, to evaluate the individuals in the population based on the fitness function, and to produce population for the next generation in a way similar to biological reproduction and natual selection. By keeping the population size constant, and through specific genetic manipulations, an organised but random exchange of information takes place, generating populations representing new sets of solutions, which evolve generation by generation, thereby eliminating some of the offspring with low fitness values. In this way, populations continue to inherit good qualities. The new population is more adapted to the environment than its predecessors, gaining individuals with better fitness values and producing increasingly better populations, which move in the direction of the optimal solution.

The genetic algorithm is a search algorithm in which the solution to a problem is represented as a chromosome through an appropriate coding scheme, and then a series of crossover, mutation and selection operations are performed to select individuals based on fitness, resulting in a globally optimal solution. Genetic operations are characterised as follows: Firstly, genetic manipulations are carried out with random disturbances, so that the process of convergence of individuals towards the optimal solution is random. Secondly, the effect of genetic manipulation depends not only on the population size,

coding method, initial population size and fitness function, but also on the probability of carrying out the three basic genetic manipulations: selection, crossover and mutation. Thirdly, the design of the genetic operators depends on the characteristics of the actual problem, and they directly influence the choice of coding method.

4.2.1 Coding

The method of transforming a feasible candidate solution from its solution space to a search space that can be handled by a genetic algorithm is called coding. The arrangement of genes in a genetic chromosome is determined by the coding method and is the first problem to be solved when applying a genetic algorithm. The conversion of the gene structure in the genetic space to data in the solution space establishes a one-to-one relationship between the solution space of the problem and the coding space of the genetic algorithm. The coding is the bridge between the optimisation problem and the algorithm. The difficulty of the algorithm and the accuracy of the solution are determined by the encoding method. The appropriate coding method has a great influence on the search effectiveness and efficiency of the genetic algorithm. Coding is a key step in designing a genetic algorithm, as it directly affects genetic operations such as selection, crossover and mutation. The main coding methods are binary coding, Gray code coding, floating point coding, multi-parameter cascade coding, multi-parameter crossover coding and real number coding (Crispin et al., 2005). In this chapter, binary coding is used.

The binary coding method is the main coding method in genetic algorithms and uses a set of coding symbols consisting of {0,1}. A binary coding symbol string is the genotype of an individual, also known as a chromosome, such as 10010011 as a chromosome with length 8. If a parameter is in the range of $[U_{min}, U_{max}]$ and the length of the binary coding symbol string is l, then $00\ldots0000$ is U_{min}, $11\ldots1111$ is U_{max}, and 00…0001 is corresponding to $U_{min} + |\triangle x|_{min}$ with $|\triangle x|_{min} = \frac{U_{max}-U_{min}}{2^l-1}$ being the coding accuracy. When X is coded as $x_l x_{l-1} \ldots x_2 x_1$, then the decoding equation is:

$$X = U_{min} + \left(\sum_{i=1}^{l} x_i \cdot 2^{l-1}\right) \cdot \frac{U_{max} - U_{min}}{2^l - 1} \tag{4.1}$$

4.2.2 Initialize generation

In genetic algorithms, a random population of individuals is usually used as the initial population, because a random initial population can achieve more traversals. Since the

crossover operation depends strongly on the initial population, and the search speed of a genetic algorithm depends mainly on the crossover operation, the initial population has an important influence on the performance of the genetic algorithm. In general, the most important aspects for the practical application of genetic algorithms are the size of the population and the generation of the population initialization. Population size is the first key factor in the optimisation power of genetic algorithms. The larger the population size, the greater the probability that the genetic algorithm will evolve to a global optimal solution, but this affects the speed of the algorithm. If the size of the population is too small, the individual solutions will not cover the feasible domain of the problem and the probability of finding the optimal solution will be reduced. Theoretically, if the length of chromosome string is l, the coding space contains 2^l individuals, it has been proved that the optimal size of binary coding population is $2^{l/2}$ (Sastry et al., 2005). The initial population can be generated based on a defined population size. In this chapter, the initial population is a randomly generated with uniform distribution.

4.2.3 Selection

The function of selection is to preserve the good genes and select the best individuals from the population that are more adaptable to survive and reproduce. This improves the global convergence of the genetic algorithm and allows it to converge to the global optimum solution. The selection procedure is based on the fitness function and, in genetic algorithms, fitness is used to measure the strengths and weaknesses of each individual in a population. The probability of passing on an individual with a higher fitness to the next generation is higher, while the probability of passing on an individual with a lower fitness to the next generation is lower. Thus the degree of fitness directly affects the magnitude of the chance of survival of each individual in the population. In this chapter, the fitness function is defined as:

$$Fitness_i^n = \frac{1}{MSE_i^n} \qquad n = 1 \dots N \tag{4.2}$$

where i is the generation and n denotes the nth individual in the generation, MSE is the mean square error.

The selection probability can be obtained by following equation:

$$P_i^n = \frac{Fitness_i^n}{\sum_{n=1}^{N} Fitness_i^n} \tag{4.3}$$

In this chapter, roulette-wheel selection is used for all tasks. The roulette selection method calculates the probability of each individual appearing in the offspring based on its fitness value, and randomly selects individuals to form the offspring population based on this probability, hence the method is also known as the fitness proportion method. The starting point of the roulette wheel selection strategy is that the better the fitness value, the greater the probability that an individual will be selected, the roulette selection method is processed as following:

1. Calculate the selection probability P_i^n of each individual in this generation.
2. Calculate the cumulative probability Q_i^n of each individual in the generation.

$$Q_i^n = \sum_{j=1}^{n} Fitness_i^j \qquad j = 1 \dots N \tag{4.4}$$

3. r is randomly generated in the range $[0,1]$, if $r < Q_i^n$, the individual id_i^n will be selected to enter the next generation.

4.2.4 Crossover

The crossover operator plays a central role in genetic algorithms and it is a feature that distinguishes them from other evolutionary algorithms. Crossover usually involves the selection of two individuals from a population with a high fitness and the exchange of one or more bits of the two individuals, or the generation of offspring from one or more parents. Crossover is used to combine new individuals and is the main method of generating new individuals that inherit the basic characteristics of their parents and play a key role in the evolutionary process. By crossover, the search power of the genetic algorithm is improved. The steps in the crossover operation of the binary genetic algorithm are as follows:

1. Pair individuals in the population with two parent individuals selected by the selection operator.
2. Based on the binary code length l, the crossover location are randomly selected from the two selected parents as the intersection fork position.
3. The crossover operation is based on the probability of crossing over P_c, and the two individuals exchange their contents at the crossover position to form two new individuals. The genetic recombination is achieved.

In general, there are several crossover methods: single point crossover, two point crossover and uniform crossover. In this chapter, single point crossover is selected to be crossover operator, because the single point crossover has the lowest probability breaking individual fitness.

Individuals are paired as a crossover target, and a random crossover position is created where the two individuals exchange genetic codes to form two new individuals. The two parents individuals are:

$$A = \{x_1, x_2, \dots, x_{l-1}, x_l\} \tag{4.5}$$

$$B = \{y_1, y_2, \dots, y_{l-1}, y_l\} \tag{4.6}$$

The random crossover position is k, so the offspring individuals are:

$$A' = \{x_1, x_2, \dots, x_k, y_{k+1}, y_{k+2}, \dots, y_{l-1}, y_l\} \tag{4.7}$$

$$B' = \{y_1, y_2, \dots, y_k, x_{k+1}, x_{k+2}, \dots, x_{l-1}, x_l\} \tag{4.8}$$

From the previous research, the crossover probability value is typical in range $[0.5, 1]$ (Srinivas and Patnaik, 1994).

4.2.5 Mutation

In genetic algorithms, mutation operations are introduced to simulate the variation that occurs during the genetic and evolutionary processes of the biological world. The mutation operation in genetic algorithms involves replacing one or more genes at one or more locations in an individual to create new individual. The mutation operation is an auxiliary method of generating new individuals for the genetic algorithm, but it is an essential step in the genetic manipulation. In order to make the genetic algorithm locally searchable and to maintain population variate, mutation has been introduced to overcome the algorithm's tendency to fall into early maturity, and is the most effective way to escape from local convergence. Crossover and mutation work together to complete the global and local search of the space, thus enabling the genetic algorithm to complete the optimization process with good search performance. Basic position mutation is a mutation in the individual string with mutation probability P_m. Generate r corresponding to each position in the individual string, if $r < P_m$, then carry out the mutation operationotherwise, gene is kept unchanged:

1. Set the mutation probability as P_m, generate random number $r_j \in (0,1)$.

2. Generate $A' = \{x'_1, x'_2, \dots, x'_{l-1}, x'_l\}$ from $A = \{x_1, x_2, \dots, x_{l-1}, x_l\}$ using $A' = \{x'_1, x'_2, \dots, x'_{l-1}, x'_l\}$

$$x'_j = \begin{cases} 1 - x_j & r_j < P_m \\ x_j & r_j \geq P_m \end{cases} \quad j \in \{1,2, \dots, l\} \tag{4.9}$$

4.2.6 Adaptative crossover and mutation

Although the global search capability of genetic algorithms has been proven in practice, the problem of premature stopping still needs to be overcome. The phenomenon of premature stopping is manifested in the following ways: Firstly, every individual in the population reaches the same position and stops, causing the population to stop evolving; secondly, the optimal solver keeps being eliminated or destroyed, causing the algorithm to fail to converge. There are several reasons that causes genetic algorithm premature stopping:

1. In the initial stage, the population produces initial individuals.
2. Diverse individuals are eliminated during the selection process, leaving individuals that are genetically similar.
3. Individuals with the same gene cross over continuously, no new individuals are created.
4. Mutation has resulted in individuals with high fitness values, but their numbers are low and they are easily eliminated.

In this chapter, in order to prevent the premature stopping, adaptative genetic algorithm which is improved from previous research by (Srinivas and Patnaik, 1994) is used to optimize the ESN.

Adaptative genetic algorithms change the crossover probability P_c and mutation probability P_m during search according to the fitness of the population, speeding up the convergence of the algorithm while ensuring that the diversity of the population is not compromised. The algorithm can be self-learning and self-searching in a decentralised space. The newly proposed adaptative genetic algorithm make adaptative adjustment by comparing the average fitness of the generation with the maximum and minimum fitness:

$$P_c = \frac{{f_{avg}}^n}{(f_{max} - f_{min})^n + {f_{avg}}^n}(P_{cu} - P_{cl}) + P_{cl} \tag{4.10}$$

$$P_m = \frac{{f_{avg}}^n}{(f_{max} - f_{min})^n + {f_{avg}}^n}(P_{mu} - P_{ml}) + P_{ml} \tag{4.11}$$

where f_{max}, f_{avg} and f_{min} denote the maximum, average fitness and minimum fitness in this generation respectively, $P_{cu}, P_{cl}, P_{mu}, P_{ml}$ are the upper and lower limits of the corssover and mutation probability respectively, $f_{max} - f_{min}$ denotes the overall convergence level, if the difference between f_{max} and f_{min} is small, then the generation tend to converge and increase the P_c and P_m appropriately, and f_{avg} indicates the optimization level of the whole generation. n and μ are the coefficient parameter. When fitness meets a big change, the crossover and mutation probability will keep at a low level to prevent damage to the newly searched individuals, this also ensures the optimizing efficient. When the average fitness value is close to the maximum fitness value, the probability of self-adaptation increases rapidly and the range of probabilities requiring cross-variation is reduced, thus speeding up the completion of evolution. Conversely, the probability of self-adaptation changes less and the need to continue to cross-mutate to generate new individuals.

4.3 Genetic algorithm optimized ESN

The performance of the ESN is quite relative to the reservoir parameter selection. As mentation in Chapter 2, the reservoir size (N), sparseness density (SD), leak rate (a) and spectral radius (SR) are the key parameters in ESN. In this chapter, the individual binary coding form shows in Figure 4.1.

Figure 4.1 The individual binary coding for ESN.

The ESN optimized by genetic algorithm is shown in Figure 4.2, and the optimization steps are as following:

1. Separate the data into 3 parts: training data, testing data and validating data.
2. Initialize the ESN and GA parameter set, make binary coding the ESN to be optimized parameters and generate the initial generation for GA.
3. Calculate the fitness of each individual, select the individuals in the current generation to be passed to the next generation according to the sort of fitness values.
4. Update the crossover probability and mutation probability, then use the individuals in the current generation to create the next generation with mutation and crossover process.

5. Repeat step 3 and step 4 until reaching the maximum iterations or meet the termination conditions.
6. Get the optimized ESN parameters.

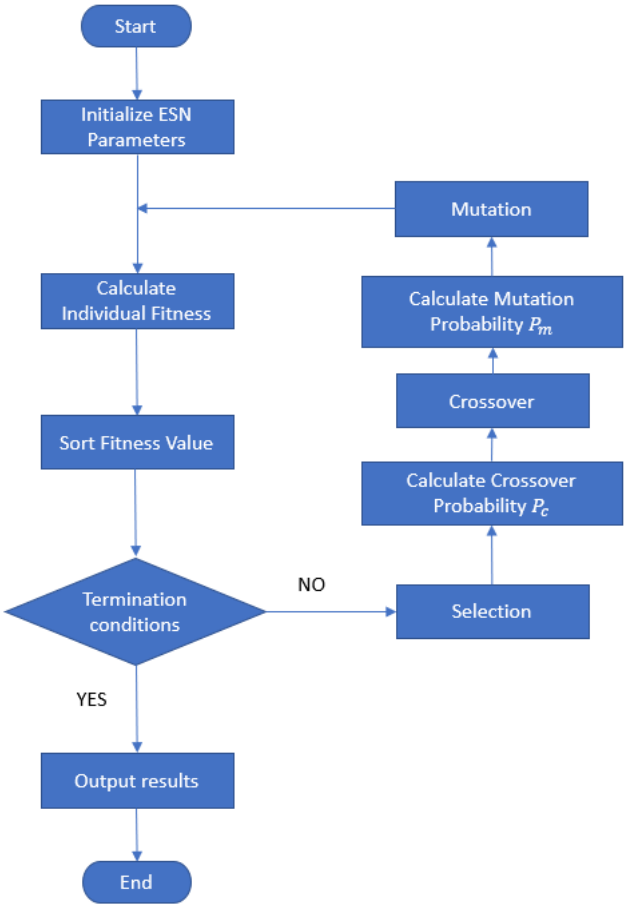


Figure 4.2 The flow chart of GA optimized ESN network.

4.4 Experiments

In this section, in order to evaluate the GA optimized ESN performance and stability, three modelling cases are considered and they are Mackey-Glass time series problem, modelling of a water tank, and modelling of a benchmark industrial fed-batch fermentation process. Mackey-Glass time series problems and modelling of a water tank case are introduced in Chapter 3. The loss training function to minimize is also the MSE. The industrial fed-batch fermentation process is introduced in the following.

Fed-batch fermentation is a well-known complicated bioprocess which microbial communities are developed while promoting reproduction and metabolism. Industrial-

scale antibiotics productions are pioneered through the development of deep tank fermentation during the scaling up of penicillin, and this development of deep tank fermentation changes the biotechnology to a billion dollar scale industry. The model simulator used in the research is referred to as IndPenSim, which is from Newcastle University. The code in Matlab R2013b is available to download at www.industrialpenicillinsimulation.com. The simulator was validated using the batch data from several 100,000L fed-batch penicillin fermentations, and it improves on previous fed-batch penicillin simulations as it considers the typical problem encountered on large-scale fermentations, including challenges associated with control of the dissolved oxygen during highly viscous fermentations and controlling key nutrients using delayed off-line measurements. The simulation considers the growth, morphology, metabolic production and degeneration of the biomass during a submerged P.chrysogenum fermentation, and it divided the internal structure of the biomass or hyphae into four separate regions: actively growing regions (A_0), non-growing regions (A_1), degenerated regions (A_3) formed through vacuolation and autolyzed regions (A_4). The component balance of four separate regions on the fermenter are shown below:

Growing regions (A_0):

$$\frac{dA_0}{dt} = \underbrace{r_b}_{branching} - \underbrace{r_{diff}}_{differentation} - \underbrace{\frac{F_{in}A_0}{V}}_{dilution} \tag{4.12}$$

Non-growing regions (A_1):

$$\frac{dA_1}{dt} = \underbrace{r_e}_{extension} - \underbrace{r_b}_{branching} - \underbrace{r_{diff}}_{differentation} - \underbrace{r_{deg}}_{degeneration} - \underbrace{\frac{F_{in}A_1}{V}}_{dilution} \tag{4.13}$$

Degeneration regions (A3):

$$\frac{dA_3}{dt} = \underbrace{r_{deg}}_{degeneration} - \underbrace{r_a}_{autolysis} - \underbrace{\frac{F_{in}A_3}{V}}_{dilution} \tag{4.14}$$

Autolysed regions (A_4):

$$\frac{dA_4}{dt} = \underbrace{r_a}_{autolysis} - \underbrace{\frac{F_{in}A_4}{V}}_{dilution} \tag{4.15}$$

Total biomass (X):

$$X = A_0 + A_1 + A_3 + A_4 \tag{4.16}$$

Product formation (P):

$$\frac{dP}{dt} = \underbrace{r_p}_{production} - \underbrace{r_h}_{hydrolysis} - \underbrace{\frac{F_{in}P}{V}}_{dilution} \tag{4.17}$$

Substrate consumption (s):

$$\frac{ds}{dt} = -\underbrace{\frac{Y_s}{X} r_e}_{extension} - \underbrace{\frac{Y_s}{X} r_b}_{branching} - \underbrace{m_s r_m}_{maintenace} - \underbrace{\frac{Y_s}{P} r_p}_{production} + \underbrace{\frac{F_s C_s}{V} + \frac{F_{oil} c_{oil}}{V}}_{feedin} - \underbrace{\frac{F_{in} s}{V}}_{dilution} \quad (4.18)$$

where $r_b, r_{diff}, r_e, r_{deg}, r_a, r_P, r_h, r_m$ are the rates of branching, differentiation, extension, degeneration, autolysis, product formation, product hydrolysis and maintenance respectively, $Y_{s/X}$ and $Y_{s/p}$ represents substrate yield coefficients of biomass and penicillin, respectively, m_s is the substrate maintenance term, and F_s, F_{oil}, c_s, c_{oil} represents the sugar and soybean oil feed rate and concentrations respectively.

The process starts with a simple phase of the batch process to grow microorganisms and improve cell quality. Thus, the production of penicillin in a supplementary batch treatment with the goal of increasing the rate is induced, while the cell growth is induced at a slower rate. The feed rate of the substrate runs in an open-loop mode during the batch period. The configuration of the bioreactor was consistent with a traditional 100,000L bioreactor.

In this process, five different variables are selected as the inputs of the models, as illustrated in Table 4.1.

Table 4.1 The model input and output variables selected in this study

Input Parameters	Description	Unit
$\boldsymbol{F_{PAA}}$	Phenylacetic acid flow	$L\ h^{-1}$
$\boldsymbol{F_g}$	Aeration rate	$m^3\ min^{-1}$
Pressure	Air head pressure	bar
$\boldsymbol{F_s}$	Sugar flow rate	$L\ h^{-1}$
$\boldsymbol{F_w}$	Water for injection flow rate	$L\ h^{-1}$
Outputs		
P	Penicillin concentration	$g\ L^{-1}$

In this study, 10 batches are generated by using the IndPenSim modular simulator under the nominal operations for model building. Among these data, 8 batches are selected as training data, one batch is set as testing dataset and the remaining batch is used as unseen validation data. The batch time is 100h and the sampling time is 20 minutes. All the batch data used in this study are spread by variable-wise spreading and normalized to [-1, 1]. The model outputs are the predicted values of total biomass concentration and penicillin concentration. In this study, 5 variables are used as model inputs and they are shown in

Figure 4.3. The variables all belongs to manipulated variables which can be manipulated externally and they are phenylacetic acid flow (F_{PAA}), aeration rate (F_g), air head pressure (pressure), sugar flow rate (F_s), and water injection flow rate (F_w). In this model, the outputs is penicillin concentration which is the most important product in this fermentation process.

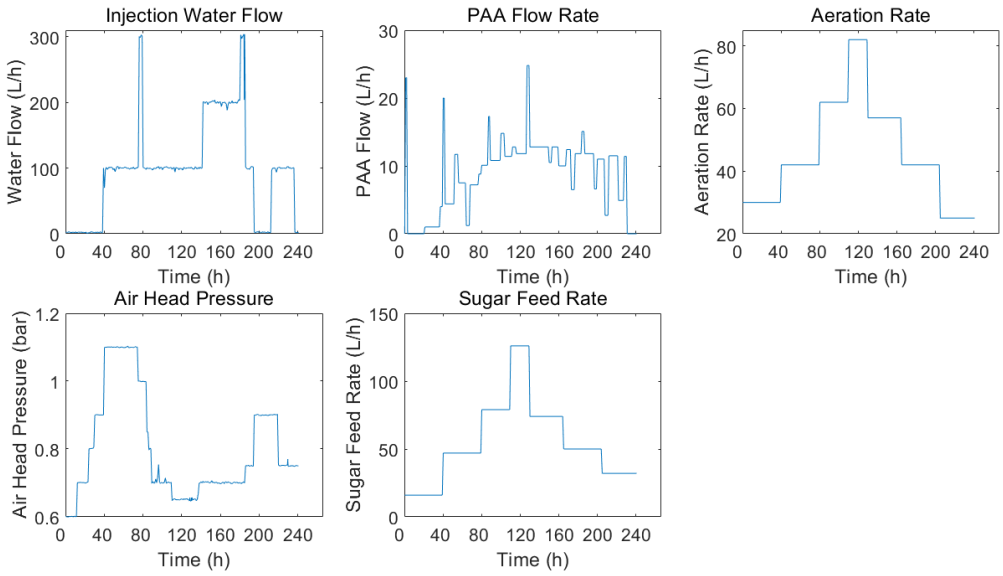


Figure 4.3 MSE distribution when reservoir size and leak rate change.

4.5 Results

To evaluate the adaptative genetic algorithm, conventional ESN, GA-ESN are trained on modelling cases. For all ESN models, every reservoir node has the activation function $\tanh(x) = \frac{\sin h(x)}{\cos h(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$; the input and backwards weights $\boldsymbol{W^{in}}$, $\boldsymbol{W^{back}}$ are fully generated by uniform distribution in the range [-0.5,0.5]. During the optimization process, the ESNs are trained by training data and calculate the fitness with test data. The comparing networks are set as following:

- Conventional ESN: In the conventional ESN, reservoir size and spectral radius factor and leak rate, are selected randomly in the ranges of [1-1000], [0-1.5] and [0-1], respectively, and the sparse density of reservoir matrix $\boldsymbol{W}$ is fixed to 0.1.
- GA-ESN: Because the largest reservoir size is set to 1000, which is close to 2^{10}, so the binary coding length is 10. The maximum generation is 500, according to the best number of individuals in one generation is $2^{l/2}$, so there 30 individuals in each

generation. The fixed crossover probability P_c and mutation probability P_m are 0.75 and 0.04 respectively. The length of crossover and mutation in the binary coding is 2.

- Adaptative GA-ESN: The set of adaptative GA-ESN is same to the GA-ESN except crossover and mutation probability. For Mackey-Glass time series prediction and water tank modelling prediction, the upper P_{cu} and lower P_{cl} limit of crossover probability are 0.9 and 0.5, the upper P_{mu} and lower P_{ml} limit of mutation probability are 0.1 and 0.02, and for penicillin fermentation process, the P_{cl} is set as 0.1.

To illustrate the parameters impacts on the ESN, Figure 4.4 shows the MSE distribution of the prediction results for Mackey-Glass time series data as the reservoir size and leak rate change with spectral radius and sparseness density are fixed, where the transformation range of reservoir size N is [50,1000], the change step size is 50, and variation range of leak rate a is [0.05,1], the transformation step size is 0.05, at the same time, $SD = 0.1$ and $SR = 0.8$.

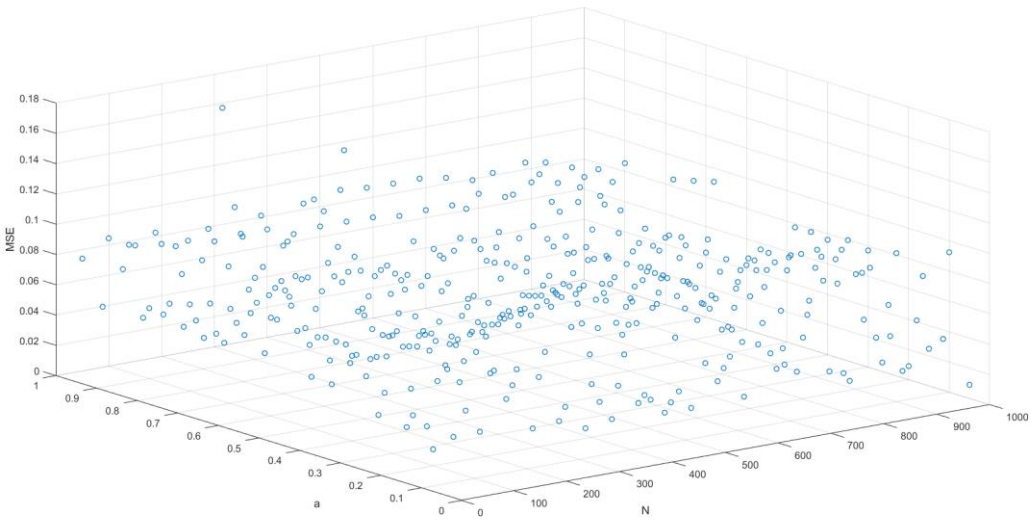


Figure 4.4 MSE distribution with different reservoir sizes and leak rates on Mackey-Glass time series data.

Figure 4.5 shows the MSE distribution of the prediction results for Mackey-Glass time series data when spectral radius and sparseness density change and reservoir size and leak rate are fixed. The $SD = [0.05,1]$, the step size is 0.032 and $SR = [0.05,1.5]$, the step size is 0.05, meanwhile, the reservoir size $N = 500$ and leak rate $a = 0.3$.

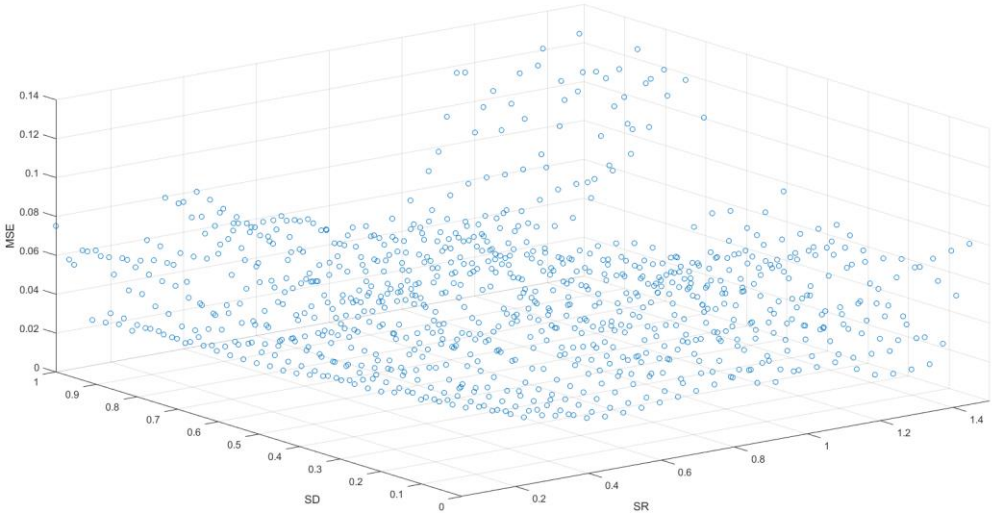


Figure 4.5 MSE distribution when spectral radius and sparseness density change.

From Figure 4.4 and Figure 4.5, it can be found that with varied combination of N, a, SR, SD, the MSE values of randomly generated ESN have large variations, this shows that these structural parameters will significantly affect the performance of the network, it is essential to use genetic algorithm to optimize the parameters.

4.5.1 Mackey-Glass time series prediction

In order to evaluate the adaptative genetic algorithm's performance on the Mackey-Glass time series prediction, traditional genetic algorithm and random echo state network has been applied. In the Mackey-Glass time series model, the long range predicted time series model is of the following form:

$$\hat{y}(t) = f\big(\hat{y}(t-1)\big) \tag{4.19}$$

Figure 4.6 shows the best individual fitness in each iteration of the adaptative genetic algorithm and the traditional genetic algorithm. Figure 4.7 and Figure 4.8 illustrate the variance of adaptative crossover probability and comparison of the maximum, minimum and average fitness in each iteration, respectively. From Figure 4.6, it can be found that the best individual fitness curve of adaptative genetic algorithm decreases more rapidly than that of the traditional genetic algorithm at nearly all iterations, indicating that the optimising convergence speed is greater. Also, it can be found that the best individual fitness curve of GA-ESN is getting flat and the value is becoming unchanged after 90 iterations, oppositely, the AGA-ESN is continually search optimal parameters after 150 iterations and by adjust the crossover probability and mutation probability, the algorithm

finds an superior result after the best individual fitness is keeping fixed for 230 iterations, this indicates that the adaptative genetic algorithm is not easy to fall into local optimum because of the adaptative crossover and mutation operations. From Figure 4.7 and Figure 4.8, it can be found that the best individual fitness updates significantly, the difference between best individual fitness, the average fitness and the minimum fitness in this generation increases, according to the Eq 4.10 and Eq 4.11, the crossover and mutation probability degrade to maintain the new best individuals not to be changed. The optimized MSE of AGA-ESN, ESN and the best MSE of 50 random ESNs on validation data are illustrated in Table 4.2, it can be found that GA search is better than the random experiments, this also indicates that the genetic algorithm is a useful tool for global searching optimization. For applying in the echo state networks, the accuracy of the prediction is improved by avoiding blindness in the selection of parameters.

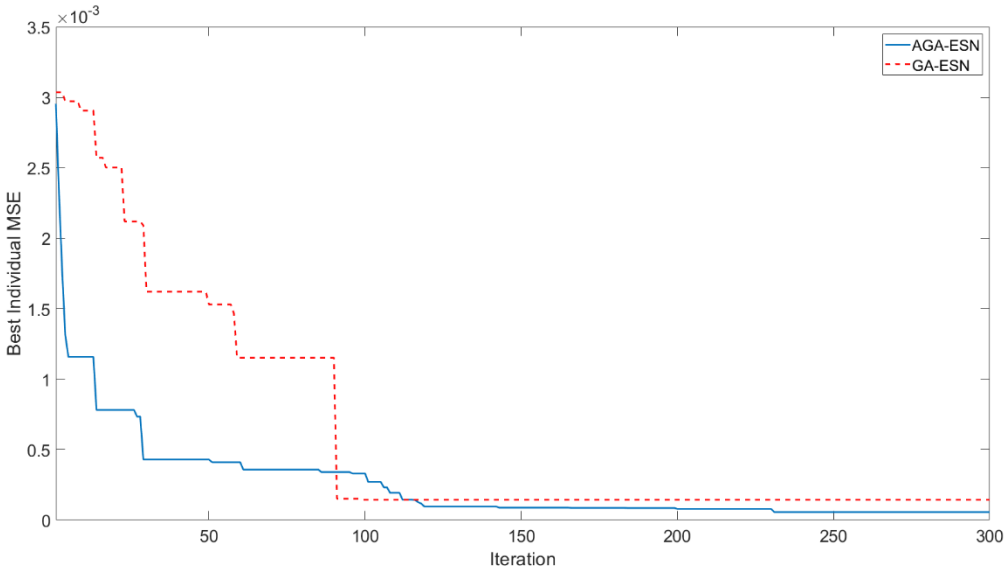


Figure 4.6 Best individual fitness of AGA-ESN and GA-ESN on Mackey-Glass time series prediction.

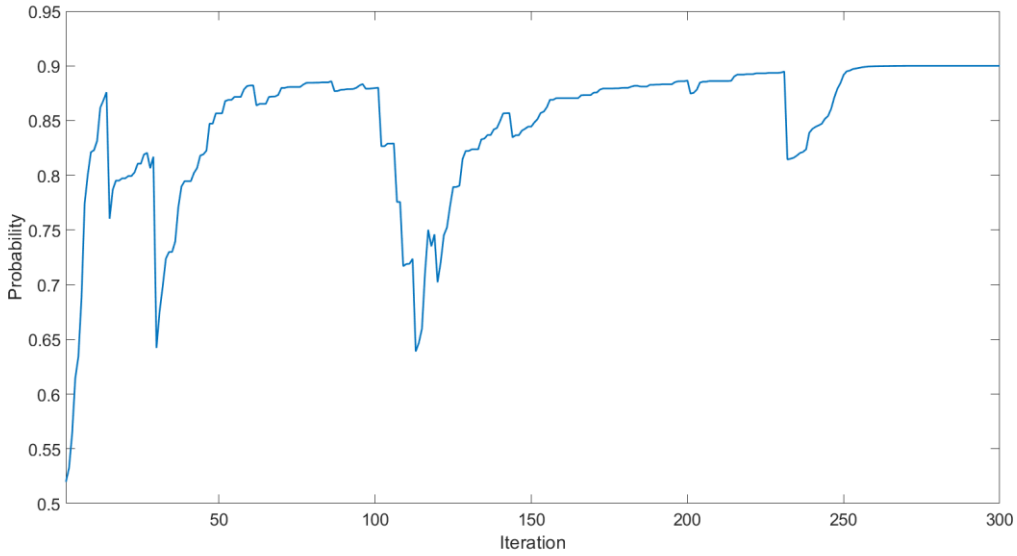


Figure 4.7 The variance of crossover probability for AGA-ESN on Mackey-Glass time series prediction.

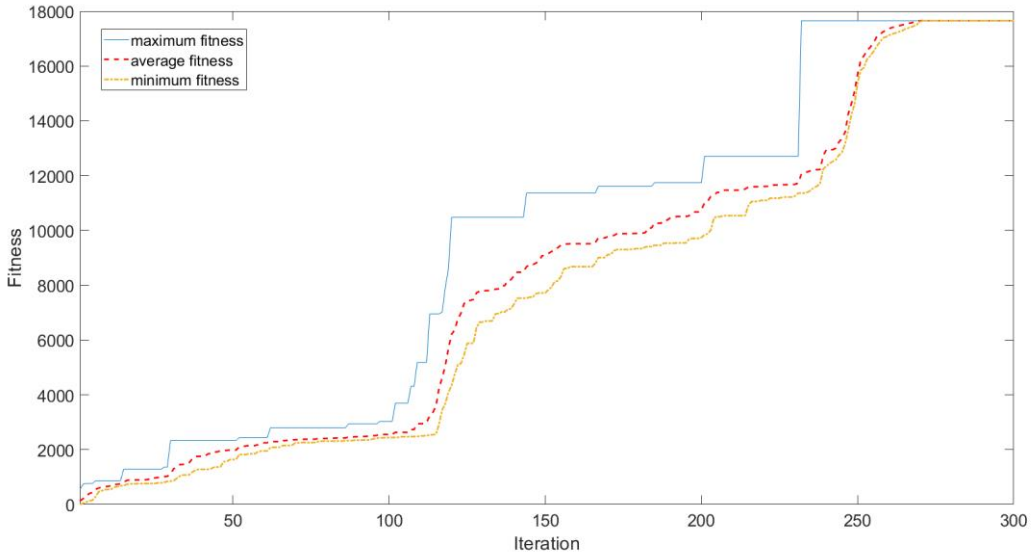


Figure 4.8 The maximum, average, minimum fitness for AGA-ESN.

Table 4.2 MSE on the validation data of Mackey-Glass time series prediction.

Optimizing Method	MSE
AGA-ESN	5.67×10^{-6}
GA-ESN	1.43×10^{-5}
ESN	6.86×10^{-4}

4.5.2 Modelling of water tank

To evaluate the adaption genetic algorithm on optimizing echo state networks for the water tank modelling case, the traditional genetic algorithm ESN and conventional ESN have been used. The case of modelling on the water tank is not so nonlinear as the case of Mackey-Glass time series prediction, the max iteration is setting as 150. The developed nonlinear dynamic model for all methods is of the following form:

$$\hat{y}(t) = f\big(\hat{y}(t-1), u(t-1)\big) \tag{4.20}$$

where $\hat{y}$ is the predicted tank level, u is the inlet flow rate, and t is the discrete time.

Figure 4.9 shows the best individual MSE of test data in each generation, Figure 4.10 and Figure 4.11 illustrate the variance of adaptative crossover probability and comparison of the maximum, minimum and average fitness in each iteration, respectively. From Figure 4.9 and Figure 4.10, it can be found that at the start several iterations, the AGA-ESN and GA-ESN have similar level fitness, because both of them are starting from randomly generated ESNs and the crossover, mutation rate are similar which are around 0.6 and 0.03. From the 7^{th} iteration, with the distribution of individuals in the past generation becoming concentrated, the values of crossover and mutation probability are increasing, until the 20^{th} iteration, the optimization process becomes more efficient than traditional GA-ESM, and at the 40^{th} iteration, the fitness of GA-ESN reaches the same level as that of AGA-ESN. After the 20^{th} iteration, the AGA-ESN falls into the local optimum, which is common occurrence in the genetic algorithm, the means to escape from the local optimum are mutation and crossover with the mutation has a greater impact. It can be seen from Figure 4.10 that although the probability is quite large, the values of the crossover probability and mutation probability are increasing gradually and slowly, which is because of the concentrating of the generation distribution. At around the 70^{th} iteration, the AGA-ESN escapes from the local optimum and fitness evolves twice around 80^{th} iteration, in contrast, GA-ESN stays deep in the local optimum after 43^{th} iteration. When the fitness evolves a big step forward, the probability decreases significantly, this can be found in Figure 4.10. Finally, the AGA-ESN finishes the optimization process around 95^{th} iteration. Also from Figure 4.11, it can be found that maximum fitness change suddenly, at the time, the curves of average fitness and minimum fitness are smoother and gentler, it means that the convergence of distribution in generation individuals is smoother and gentler, finally, the distribution converges, all the individuals in one generation are same. The optimized MSE of AGA-ESN, ESN and the best MSE of 50 random ESNs on validation data in the water

tank modelling case are illustrated in Table 4.3, as for the case with small degree of non-linearity, the performance of conventional random ESN is still a lot of room for improvement, and adaptative genetic algorithm is a powerful global optimization method for searching parameters, it can strength the modelling ability of ESN.

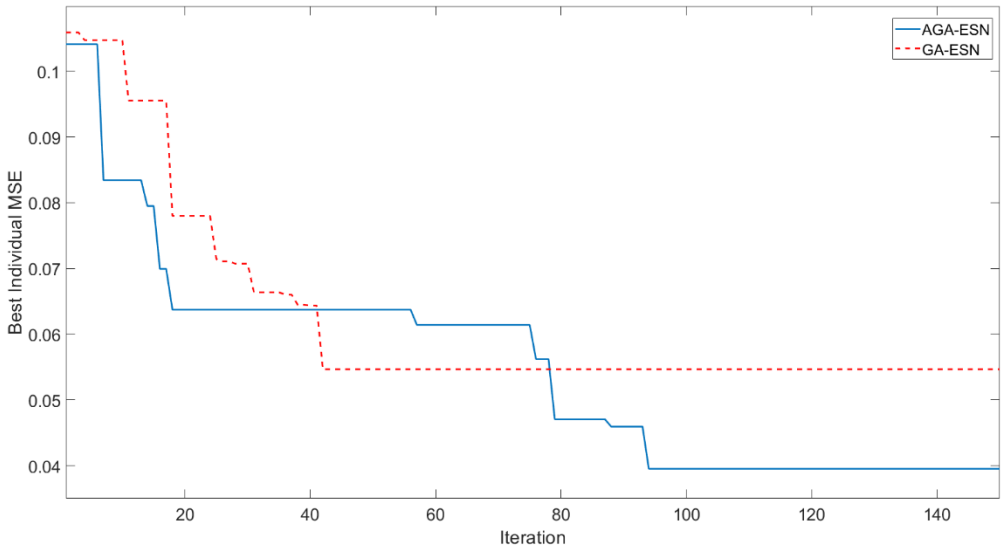


Figure 4.9 Best individual fitness of AGA-ESN and GA-ESN on water tank modelling.

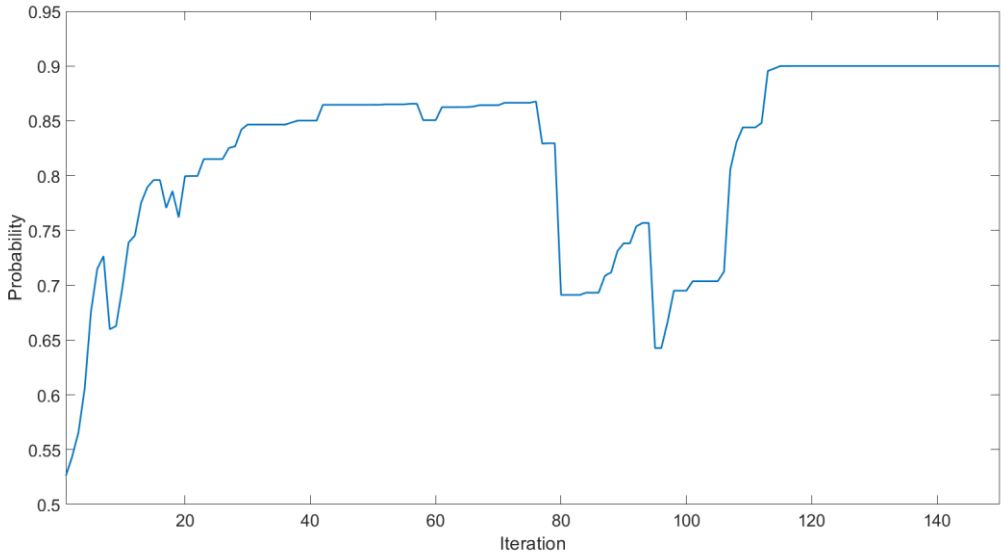


Figure 4.10 The variance of crossover probability for AGA-ESN on water tank modelling.

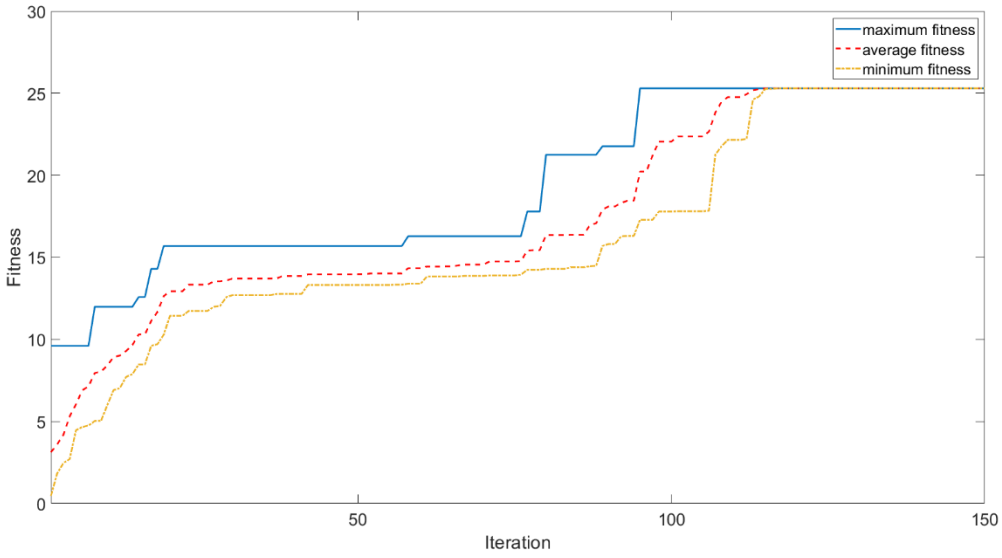


Figure 4.11 The maximum, average, minimum fitness for AGA-ESN on water tank modelling.

Table 4.3 MSE on the validation data of water tank modelling.

Optimizing Method	MSE
AGA-ESN	0.0395
GA-ESN	0.0548
ESN	0.405

4.5.3 Modelling of batch penicillin fermentation

In the InPenSim penicillin fermentation model, GA-ESN and conventional ESN are used to compare with the adaptative genetic algorithm ESN. The maximum iteration number is set at 100. The developed nonlinear dynamic model is of the following form:

$$(\hat{y}_1(t)) = f\big(u_1(t-1), u_2(t-1), u_3(t-1), u_4(t-1), u_5(t-1)\big) \qquad (4.21)$$

Figure 4.12 shows the best individual MSE of test data in each generation. Figure 4.13 and Figure 4.14 illustrate the variance of adaptative crossover probability and comparison of the maximum, minimum and average fitness in each iteration, respectively. From Figure 4.12, it can be found that at the start the convergence speed of AGA-ESN and GA-ESN are similar, this is because the crossover and mutation probability are similar at this time, but GA-ESN falls into a local optimum after the 11^{th} iteration. Because of the adaptative crossover and mutation mechanism, AGA-ESN can continue to discover better individuals. For this non-linear complex fed-batch chemical process, the larger probability variance

intervals lead to more accurate control. After the 40th iteration, the AGA-ESN comes to the optimum individuals. The optimized MSE of AGA-ESN, ESN and the best MSE of 50 random ESNs in the water tank modelling case are illustrated in Table 4.4. This shows that the AGA-ESN have ability to model the complex multiple inputs chemical process.

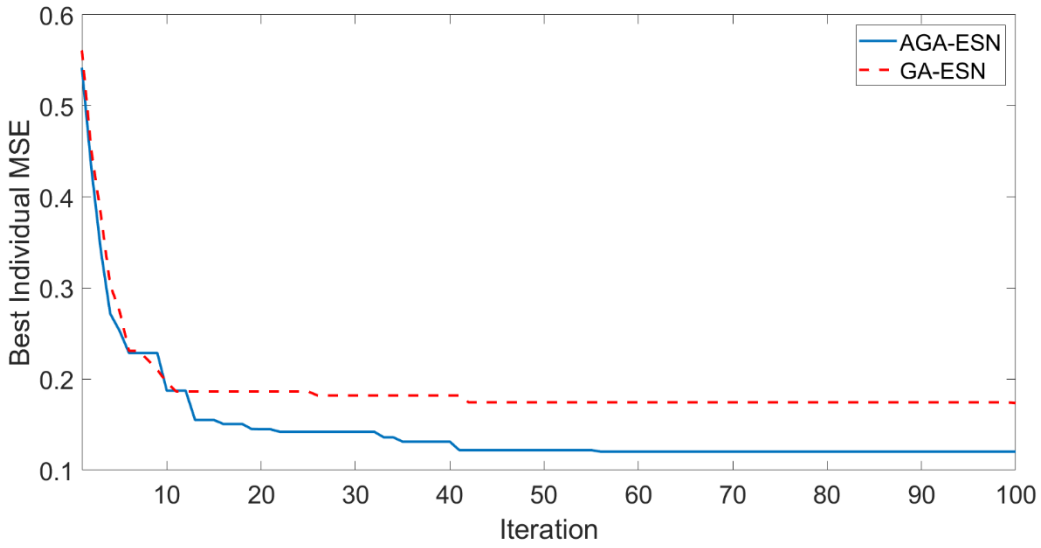


Figure 4.12 Best individual fitness of AGA-ESN and GA-ESN on penicillin fermentation process.

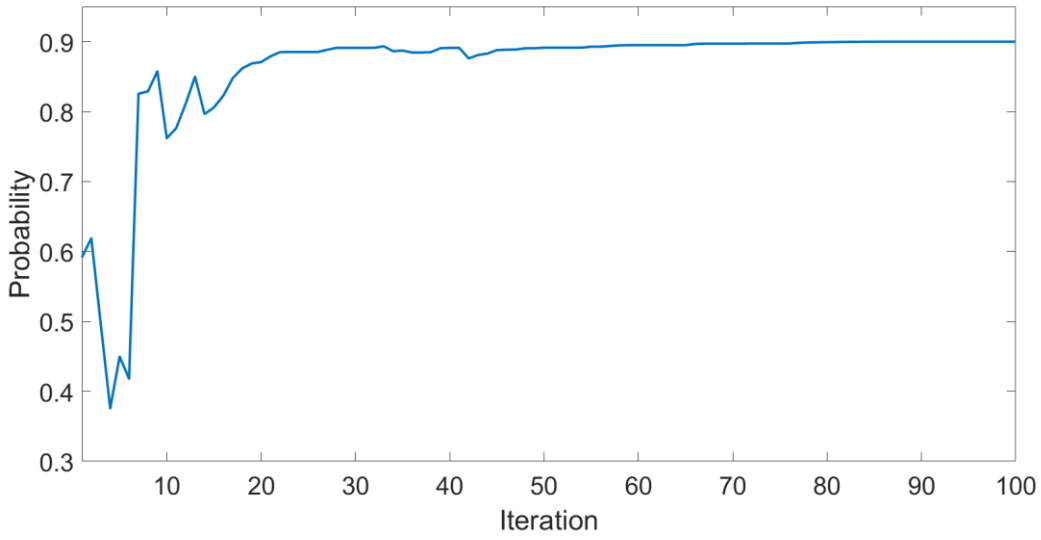


Figure 4.13 The variance of crossover probability for AGA-ESN on penicillin fermentation process.

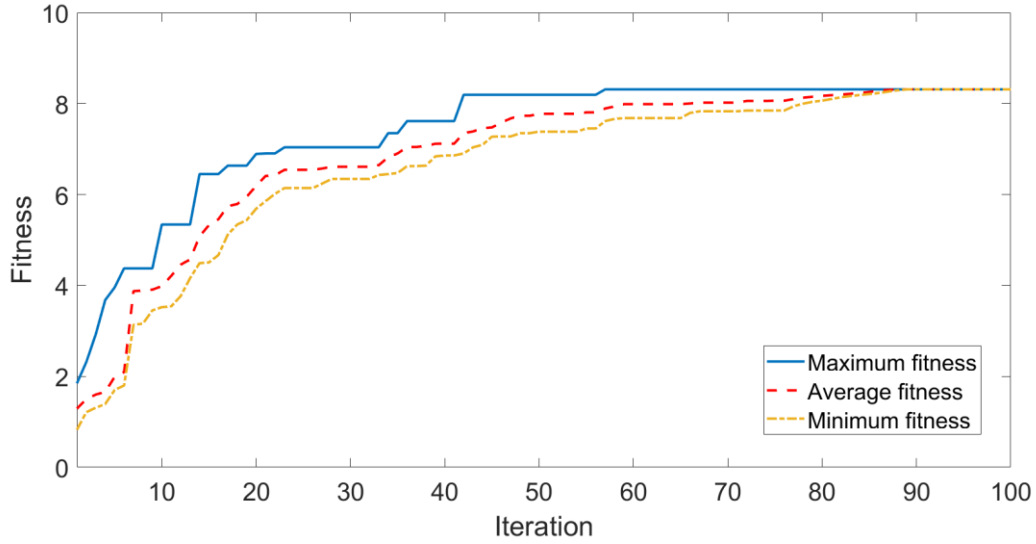


Figure 4.14 The maximum, average, minimum fitness for AGA-ESN on penicillin fermentation process.

Table 4.4 MSE on the validation data of penicillin fermentation process.

Optimizing Method	MSE
AGA-ESN	0.120
GA-ESN	0.185
ESN	0.645

4.6 Conclusions

In this chapter, an adaptative genetic algorithm optimized echo state network is proposed. Four structural parameters of ESN which are reservoir size, leak rate, spectral radius and sparseness density are optimized by adaptative genetic algorithm to improve the performance of ESN. Compared to the traditional genetic algorithm ESN and conventional random ESN, AGA-ESN has shown better performance on long range prediction Mackey-Glass time series data, single input water tank modelling data, and complex multiple input fed-batch penicillin fermentation data. The experiment results show that because of the adaptive crossover and mutation probability mechanism, the optimizing efficiency and the chance that escapes from the local optimum of AGA-ESN are higher.

Chapter 5 Covariance matrix adaptation evolution strategy optimized ESN

5.1 Introduction

In the previous chapter, the structure parameters are optimized by genetic algorithm (GA), and good predicted results are achieved, but GA may still fall into local optimum. Therefore, an improved global search method is needed to better solve the problem of feature subset selection. As optimizing resorvoirs is generally challenging and checking the performance of a resulting ESN is relatively inexpensive, evolutionary optimization techniques such as evolutionary strategy (ES) is a natural strategy for this optimisation task (Lukoševičius and Jaeger, 2009). Evolutionary strategy is a useful global searching method for optimization because they do not possess the limitations found in the traditional methods, thus evolutionary strategies can be used to build a good ESN model. Optimization of ESN using evolutionary strategies can generally be carried out using the following different approaches: optimization the topology of ESN reservoir; optimization of the connection weights; and optimization of the reservoir parameters of ESN. If the optimization method operates on the connection weights directly, the search space should be quite large, moreover, the variance of the performance across different reservoirs with the same spectral radius is still quite substantial, which is clearly undersirable (Schrauwen et al., 2008), so to optimize the ESN reservoir property by optimizing reservoir parameters is a compellent procedure. In an earlier work, several evolutionary algorithm on ESN reservoir optimization have been presented including differential evolution (DE) (Otte et al., 2016), particle swarm optimization (PSO) (Chouikhi et al., 2015) and Evolino (Schmidhuber et al., 2007). Addtionally, other metaheuristic methods were used to optimize the reservoir global parameters and topology (Ferreira and Ludermir, 2010, Ferreira et al., 2013).

In this chapter, a new method for optimizing ESN using covariance matrix adaptation evolution strategy (CMA-ES) is proposed. CMA-ES is an efficient and widely used metaheuristic approach to search optimal regions on complex spaces with the advantage is invariant against order-preserving transformations of the fitness function value and in particular against rotation and translation of the search space (Hansen, 2006). Three global reservoir parameters are optimized by CMA-ES in this chapter and they are reservoir size, spectral radius factor and leak rate. The reservoir size is the number of neurons in the

reservoir layer, for the specific task, the performance varies because of reservoir scale. The obtained internal weights of reservoir needs to be scaled to satisfy the necessary condition for echo state property (ESP) (Jaeger et al., 2007b), spectral radius is the key factor to maintain ESP. Leak rate determines the time scales and status updating scale in the reservoir.

The remaining part of this chapter is organised as follows. Section 5.2 gives a brief introduction of CMA-ES,.In Section 5.3, the CMA-ES optimized ESN is proposed. The application examples and results are given in Section 5.4 and Section 5.5 respectively. Finally, the conclusion is given in Section 5.6.

5.2 Covariance matrix adaptation evolution strategy

The covariance matrix adaption evolution strategy (CMA-ES) is a well-established evolutionary algorithm for real-valued optimization with many successful applications (Hansen and Ostermeier, 2001). The main advantages of CMA-ES lie in its invariance properties, which are achieved by carefully designed variation and selection operators and its efficient adaptation of the mutation distribution. The CMA-ES is invariant against order-preserving transformations of the fitness function value and in particular against rotation and translation of the search space.

In the CMA-ES, the population of new search offspring $\boldsymbol{x} \in \mathbb{R}^n$ is generated by sampling a multivariate normal distribution (Hansen et al., 2003):

$$\boldsymbol{x}_k^{(g+1)} \sim \boldsymbol{m}^{(g)} + \sigma^{(g)} \cdot \aleph\left(0, \boldsymbol{C}^{(g)}\right) \; for\; k = 1, \dots, \gamma \tag{5.1}$$

Where $\boldsymbol{x}_k^{(g+1)}$ denotes the kth offspring at the generation $g+1$; $\boldsymbol{m}^{(g)}$ is the mean value of the search distribution at generation g; $\aleph\left(0, \boldsymbol{C}^{(g)}\right)$ is a multivariate normal distribution with zero mean and covariance matrix $\boldsymbol{C}^{(g)}$; and $\sigma^{(g)}$ is the step-size, a scaling parameter at generation g. There are several advantages of covariance matrix $\boldsymbol{C}^{(g)}$:

1. $\boldsymbol{C}$ is a diagonal matrix.
2. $\boldsymbol{C}$ is a positive semi-definite matrix.
3. All eigenvalues are non-negative real numbers.
4. All eigenvalues are orthogonal.

$$\boldsymbol{C} = \boldsymbol{B}^T \boldsymbol{D}^2 \boldsymbol{B} = [b_1\ b_2\ \dots\ b_n] \begin{bmatrix} \lambda_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n^2 \end{bmatrix} [b_1\ b_2\ \dots\ b_n]^T \tag{5.2}$$

$$\boldsymbol{C}^{\frac{1}{2}} = \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{B} \tag{5.3}$$

where $\boldsymbol{B}$ is matrix of $\boldsymbol{C}$ eigenvectors as row vectors. $\boldsymbol{D}$ is a diagonal matrix with $\boldsymbol{C}$ eigenvectors as row vectors.

After those γ individuals have been created, they are evaluated on the objective function which is the mean squared errors (MSE) of the ESN and sorted according to their objective function values. To implement CMA-ES, $\boldsymbol{m}^{(g+1)}$, $\sigma^{(g+1)}$, and $\boldsymbol{C}^{(g+1)}$ need to be updated. The new mean $\boldsymbol{m}^{(g+1)}$ of the search distribution is generated using truncation selection by choose $\mu < \gamma$ out of γ offsprings.

$$\boldsymbol{m}^{(g+1)} = \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\gamma}^{(g+1)} \tag{5.4}$$

$$\sum_{i=1}^{\mu} w_i = 1\,,\ w_1 \geq w_2 \geq \cdots \geq w_\mu > 0 \tag{5.5}$$

$$\mu_{eff} = \left(\frac{\|\boldsymbol{w}\|_1}{\|\boldsymbol{w}\|_2}\right)^2 = \frac{\|\boldsymbol{w}\|_1^2}{\|\boldsymbol{w}\|_2^2} = \frac{\left(\sum_{i=1}^{\mu}|w_i|\right)^2}{\sum_{i=1}^{\mu} w_i^2} = \frac{1}{\sum_{i=1}^{\mu} w_i^2} \tag{5.6}$$

where $w_{i=1\ldots\mu} \in \mathbb{R}_{>0}$, positive weight coefficients for recombination. For $w_{i=1\ldots\mu} = 1/\mu$, Eq(5.4) calculates the mean value of μ selected points. Usually, $\mu_{eff} \approx \frac{\gamma}{4}$ indicates a reasonable setting of w_i.

The final equation for updating $\boldsymbol{m}$ is:

$$\boldsymbol{m}^{(g+1)} = \boldsymbol{m}^{(g)} + c_m \sum_{i=1}^{\mu} w_i \left(\boldsymbol{x}_{i:\gamma}^{(g+1)} - \boldsymbol{m}^{(g)}\right) \tag{5.7}$$

where $c_m \leq 1$ is the learning rate, usually set to 1.

The parameter $\sigma^{(g)}$ controls the overall scale of the distribution. It is separated from the covariance matrix, so the step size can be changed more quickly than the change of full covariance matrix. A larger step size will result in faster parameter updates. To assess whether the current step size is appropriate, CMA-ES constructs an evolution path by summing successive sequences of moving steps. The cumulative step length adaptation is shown in Figure 5.1. By comparing this path with the path length generating in a randomly chosen (meaning that each step is uncorrelated) state, $\sigma^{(g)}$ can be updated as following:

$$\boldsymbol{p}_\sigma^{(g+1)} = (1 - c_\sigma)\boldsymbol{p}_\sigma^{(g)} + \sqrt{c_\sigma(2 - c_\sigma)\mu_{eff}}\,\boldsymbol{C}^{(g)^{-\frac{1}{2}}}\frac{\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)}}{\sigma^{(g)}} \tag{5.8}$$

where $\boldsymbol{p}_\sigma^{(g)}$ is the conjugate evolution path at generation g. $c_\sigma < 1$, $\frac{1}{c_\sigma}$ is the backward time horizon of the evolution path. $\sqrt{c_\sigma(2 - c_\sigma)\mu_{eff}}$ is a normalization constant.

To updated $\sigma^{(g)}$, compare $\left\|\boldsymbol{p}_\sigma^{(g+1)}\right\|$ with its expected length $E\|\aleph(0, \boldsymbol{I})\|$:

$$\ln\sigma^{(g+1)} = \ln\sigma^{(g)} + \frac{c_\sigma}{d_\sigma}\left(\frac{\left\|\boldsymbol{p}_\sigma^{(g+1)}\right\|}{E\|\aleph(0, \boldsymbol{I})\|} - 1\right) \tag{5.9}$$

$$\sigma^{(g+1)} = \sigma^{(g)} exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\left\|\boldsymbol{p}_\sigma^{(g+1)}\right\|}{E\|\aleph(0,\boldsymbol{I})\|} - 1\right)\right) \tag{5.10}$$

ere $d_\sigma \approx 1$, damping parameter, scales the change magnitude of $\ln\sigma^{(g)}$. The selection of c_σ, d_σ and $E\|\aleph(0,\boldsymbol{I})\|$ is based on in-depth investigations of the algorithm.

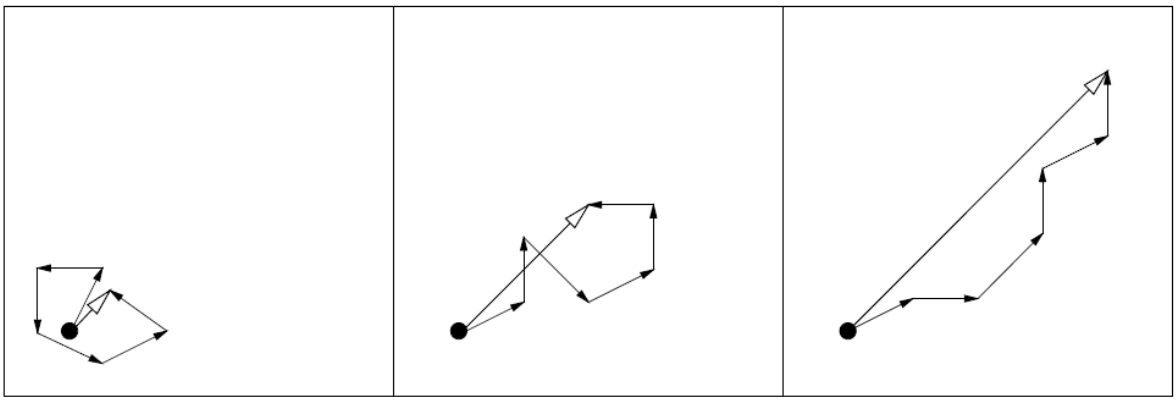


Figure 5.1 Three evolution paths of six steps from different selection situations.

For updating the covariance matrix $\boldsymbol{C}^{(g+1)}$, the principle is to increase the variance of successful searching directions. This means to increase the search probability in these directions, the final covariance matrix adaptation combines $\boldsymbol{rank-one-update}$ with $\boldsymbol{rank-\mu-update}$:

$$\boldsymbol{p}_c^{(g+1)} = (1-c_c)\boldsymbol{p}_c^{(g)} + h_\sigma\sqrt{c_c(2-c_c)\mu_{eff}}\,\frac{\boldsymbol{m}^{(g+1)}-\boldsymbol{m}^{(g)}}{\sigma^{(g)}} \tag{5.11}$$

$$\boldsymbol{C}^{(g+1)} = \left(1-c_1-c_\mu\right)\boldsymbol{C}^{(g)} + c_1\boldsymbol{p}_c^{(g+1)}\boldsymbol{p}_c^{(g+1)^T} + \cdots$$
$$c_\mu\sum_{i=1}^{\mu} w_i\left(\frac{\boldsymbol{m}^{(g+1)}-\boldsymbol{m}^{(g)}}{\sigma^{(g)}}\right)\left(\frac{\boldsymbol{m}^{(g+1)}-\boldsymbol{m}^{(g)}}{\sigma^{(g)}}\right)^T \tag{5.12}$$

where c_1, c_c, c_μ, c_σ, d_σ are some empirical parameters usually determined by the dimension of the problem (number of optimized parameters), *n*.

The information from the entire population is used efficiently by $\boldsymbol{rank-\mu-update}$, and the information of correlations between generations is exploited by using the evolution path of $\boldsymbol{rank-one-update}$. The covariance matrix adaptation does not explicitly control the overall scale of the distribution, which is the step size. The covariance matrix adaption increases or decreases the scale only in a single direction for

each selected step or it decreases the scale by fading out old information by a given, non-adaptive factor. To control the step-size, the method used in CMA-ES can be applied independently of the covariance matrix update and is denoted as cumulative step length adaptation (CSA)

It has been illustrated by experiments that the covariance matrix $\boldsymbol{C}^{(g)}$ is similar to the inverse of the Hessian matrix of the problem at the optimum point. In the procedure of CMA-ES, the number of offspring γ is the most important parameter that must be fitted with the ruggedness of the fitness function. From experimental tests, $\gamma = 4 + 3\ln n$ is usually adopted (Kämpf and Robinson, 2009).

5.3 Optimizing ESN by using CMA-ES

This chapter proposes using CMA-ES to optimize the structure parameters of ESN for nonlinear process modelling. Figure 5.2 shows the flow chart of the proposed algorithm. The procedure for optimizing ESN by using CMA-ES can be summarized as follows:

1. Data for model building are divided into three sets: training data, testing data, and unseen validation data, and then they are normalized to have zero mean and unit variance.
2. Establish an ESN with random N, α and ρ in the range based on sufficient internal units as default. The activation function used here in the hidden layer (reservoir) is $f = \tanh$ and the input weights and reservoir weights are generated randomly.
3. Train the established ESN with training data using ridge regression. The ridge parameter is selected to suit the data set, normally through cross-validation.
4. Optimize the ESN by Covariance Matrix Adaption Evolution Strategy. The MSE on the testing data is used as objective function. The optimization objective is to upgrade the values of N, α and ρ to minimize the MSE on the testing data. Repeat CMA-ES to optimize ESN until convergence is achieved. The best CMA-ES-ESN of validation performance is determined.
5. Evaluate the model on the unseen validation data.

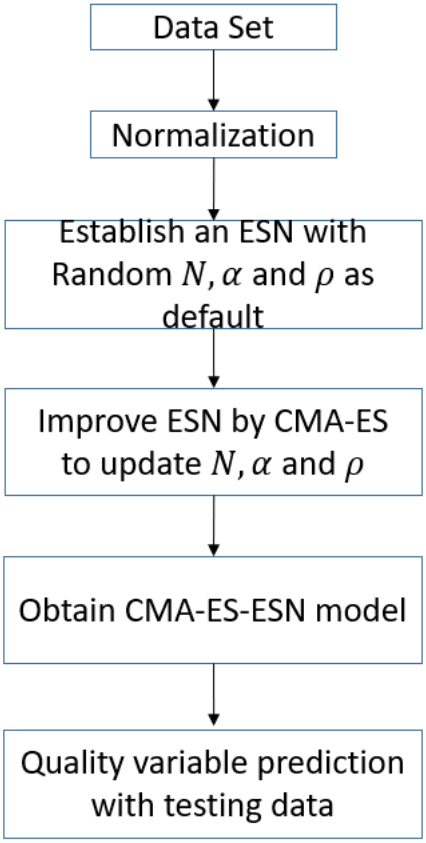


Figure 5.2 The optimization diagram of CMA-ES-ESM model.

5.4 Application examples

In this section, three modelling case studies are introduced to test the performance of CMA-ES-ESN. The three case studies are Mackey-Glass time series prediction, modelling of a water tank, and modelling of a benchmark industrial fed-batch fermentation process, which are introduced in the previous chapters. To evaluate the effectiveness of CMA-ES-ESN, the results are compared with the original ESN, GA-ESN, long short-term memory networks and feedforward neural networks.

The fitness function to minimize is the MSE of the ESN on the training data:

$$MSE = \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N} \tag{5.13}$$

5.5 Results

The prediction performance for ESN models and the hybrid models on the three applications are provided in this section. The following procedures are compared:

- ESN: For all ESN model and hybrid ESN model, every reservoir node has the activation function $\tanh(x) = \frac{\sin h(x)}{\cos h(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$; the input and backwards weights $\boldsymbol{W^{in}}$, $\boldsymbol{W^{back}}$ are fully generated by uniform distribution in the range [-0.5,0.5], and similarly the internal weights $\boldsymbol{W}$ are created by uniform distribution in the range [-0.5,0.5] with different sparse density and reservoir size. In the standard ESN, three structural parameters, reservoir size, leak rate and spectral radius factor, are selected randomly in the ranges of [1-1000], [0-1] and [0-1.5], respectively, finally, generate 50 ESNs with different random seed, and time consumption of standard ESN is sum of 50 times.

- CMA-ES-ESN: In CMA-ES-ESN, the three structural parameters to be optimized are reservoir size, leak rate and special radius with the ranges of [1-1000], [0-1] and [0-1.5], respectively, and according to $\gamma = 4 + 3\ln n$, when $n = 3$, γ is found to be 7, i.e. the population size is 7, and initial search step length is set at 0.2. The stopping criterion varies from application to application, and the maximum number of iterations is reached according to $max\ stopeval\ number = 1000 * n^2$, which is 9000 in this case or if the minimal fitness value stops decreasing for a consecutive 40 iterations.

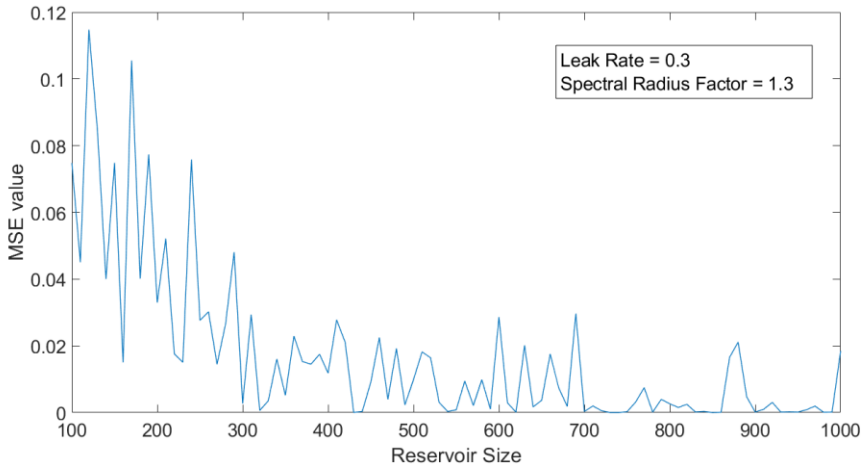
- GA-ESN-RWS: In this paper, the selection method used in GA is roulette wheel selection, the maximum number of generations is 500, in each generation there are 10 individuals, crossover and mutation rates are selected as 0.6 and 0.03 respectively.

- FNN: The feedforward multilayer neural network is generated by Neural Time Series Toolbox from MATLAB® with Levenberg-Marquardt training algorithm. The feedforward neural network has one hidden layer with 20 neurons determined by cross validation and is trained for 600 epochs (iterations).

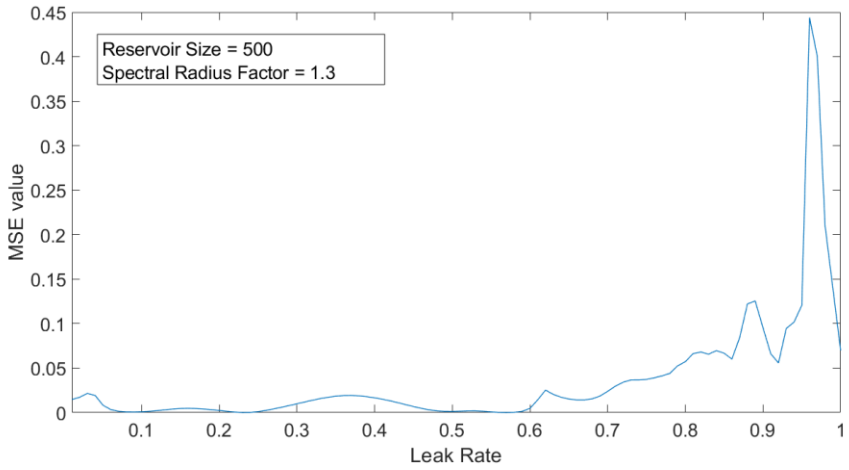
- LSTM: Long short-term memory network is established by Deep Learning Toolbox from MATLAB®. Specify the LSTM layer to have 100 hidden units, set the solver to 'adam' and train the network for 500 epochs. To prevent the gradients from exploding, set the gradient threshold to 1. Specify the initial learn rate 0.005 and drop the learn rate after 100 epochs by multiplying by a factor of 0.2.

5.5.1 Mackey-Glass Time Series Prediction

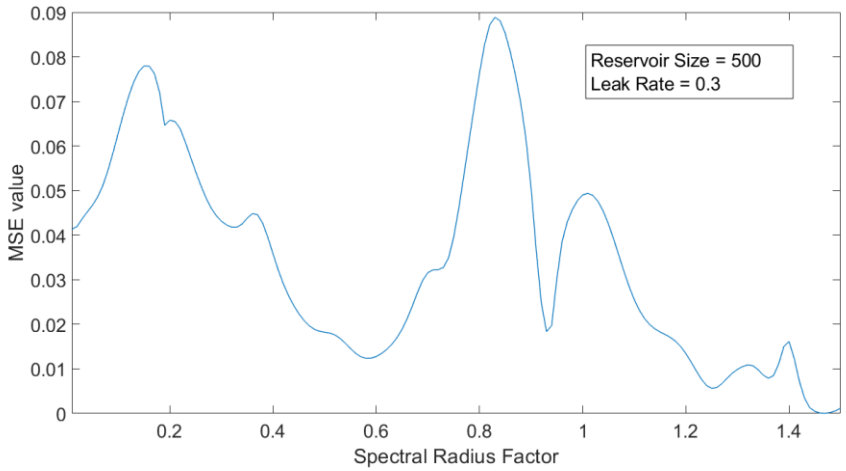
The influence of the three ESN structure parameters considered in this study on ESN model performance is illustrated in Figure 5.3. These figures are generated during the model validation process when modelling the Mackey-Glass Time Series data. The Y axis is the MSE value of training error and X axis represents three ESN structure parameters, which are reservoir size, leak rate and spectral radius factor with corresponding ranges, [100-1000], [0-1], and [0-1.5]. In Figure 5.3(a), because of sparse distribution of neurons in the reservoir, small reservoir size cannot keep accurate and stable performance of the model. It can be seen from Figure 5.3(b) that the influence of leak rate and spectral radius factor on the ESN performance is more continuous and curves are smoother. In contrast, reservoir size causes more tremendous impact on ESN performance. Figure 5.3(c) shows that there are several local minimum and, overall, the networks with factor values larger than 1 give better performance. Figure 5.3 indicates the need of optimization of ESN structure parameters.



(a) Reservoir size



(b) Leak Rate



(c) Spectral Radius Factor

Figure 5.3 Impact of structure parameters on ESN performance, (a). reservoir size, (b). leak rate, (c). spectral radius factor.

In the Mackey-Glass time series model, the long range predicted time series model is of the following form:

$$\hat{y}(t) = f\big(\hat{y}(t-1)\big) \tag{5.14}$$

The LSTM model is of the form:

$$\hat{y}(t) = f\big(\hat{y}(t-1), \hat{y}(t-2)\big) \tag{5.15}$$

where $\hat{y}(t)$ is the predicted value at time step t.

The equations represent long range prediction as the model prediction is recursively used as the model input.

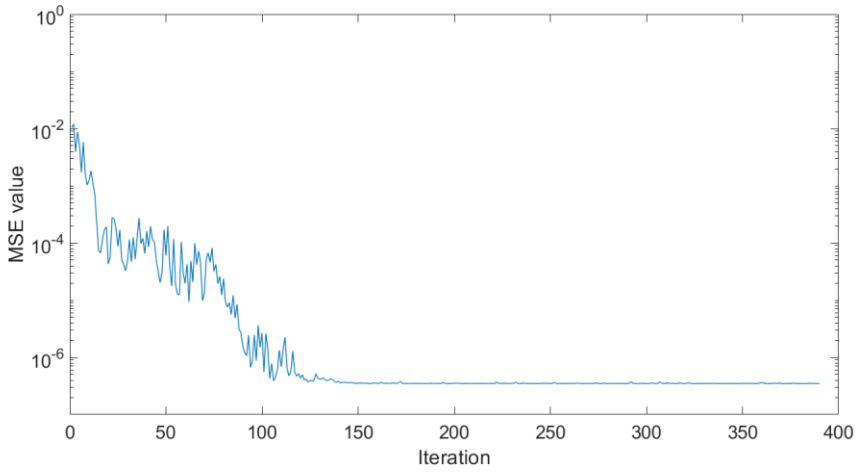
Figure 5.4(a) shows the variation of fitness values (MSE), while Figure 5.4(b) illustrates the fluctuation of three ESN structural parameters, reservoir size, leak rate and spectral radius factor, Figure 5.4(c) shows the distribution of MSE of ESN during optimization process. From Figure 5.4(a), it can be seen that in first 150 iterations, although there are some oscillations in the fitness values, the tendency of fitness drops dramatically, and stabilize after about 150 iterations. Figure 5.4(b) shows the means of reservoir size, leak rate and spectral radius factor at different iterations. It can be found that after 150 iterations, the mean values are almost constant indicating convergence. Figure 5.4(c) illustrates the distribution of ESN fitness during CMA-ES optimization process and it can be seen that the fitness (MSE of ESN) is concentrated around 10^{-6}. The actual values, LSTM predictions and CMA-ES-ESN predictions on the unseen validation dataset are plotted in Figure 5.5 where the prediction errors are shown as well. The MSE is 3.4×10^{-7} under the selected parameters for 1000 samples, and for the first 500 validation data samples, the MSE is 7.4×10^{-8}. It can be seen from Figure 5.5(a) and Figure 5.5(b) that LSTM prediction errors increase when the signals come to peak or bottom, the CMA-ES-ESN is under the similar situation but its prediction errors are much smaller than those of LSTM. It can be seen from Figure 5.5(c) that, in the first 611 samples of the unseen validation data, the multi-step-ahead predictions are quite close to the actual data, and starting from the 611th sample, the prediction error increases dramatically. This is because the memory capacity in the reservoir under the specific structural parameters is limited

and, under this situation, the memory capacity is around 500 steps and can be measured by pseudo-Lyapunov exponent (Verstraeten et al., 2007).

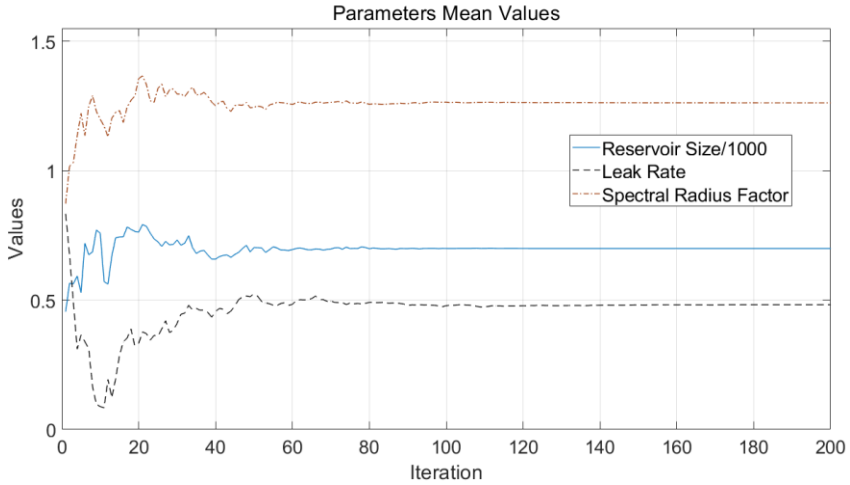
The prediction results and time consumptions of different models on the validation dataset are shown in Table 5.1, GA-ESN-RWS with roulette wheel selection procedure overcomes the randomness of the original ESN but the process is easy to get into local optimum when one fitness is much smaller than others in one iteration. The MSE of LSTM is slightly larger than that of standard-ESN, in contrast, the FNN error surpasses other four methods significantly. Regarding to time consumptions, CMA-ES is a faster convergence algorithm than GA-RWS, moreover, LSTM with back-propagation-through-time is computation and time consumption method. These results show that ESN has captured the nonlinear dynamic relationship to make better long range predictions on time series modelling tasks, also prove that CMA-ES is a powerful optimization tool for ESN.

Table 5.1 Model performance on the Mackey-Glass time series.

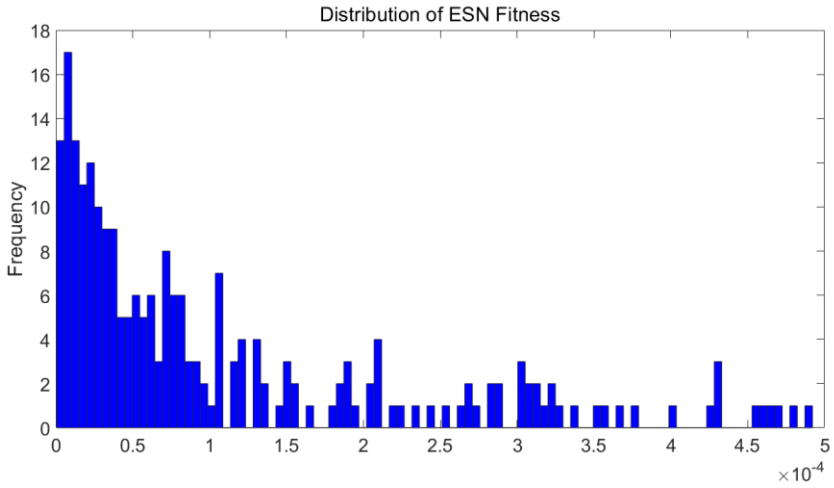
Approaches	MSE	Time(s)
GA-ESN-RWS	2.7×10^{-6}	383.1
CMA-ES-ESN	3.4×10^{-7}	162.8
Standard-ESN(50)	2.5×10^{-4}	24.6
LSTM	8.4×10^{-4}	218.2
FNN	0.052	4.5



(a)

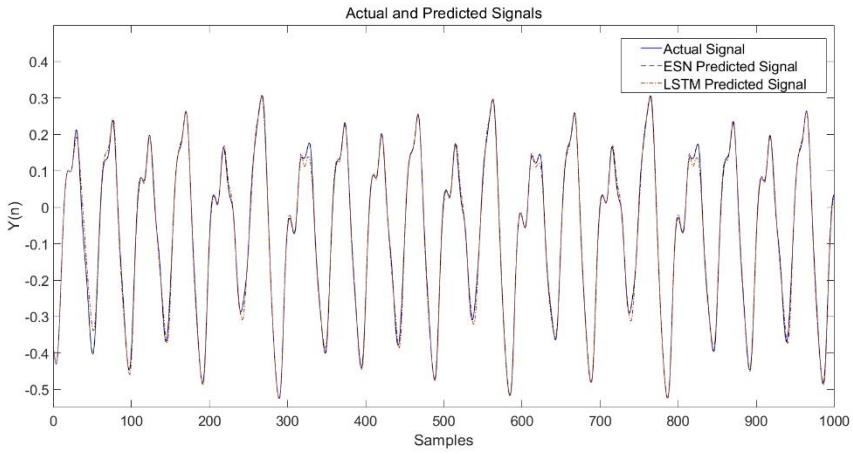


(b)

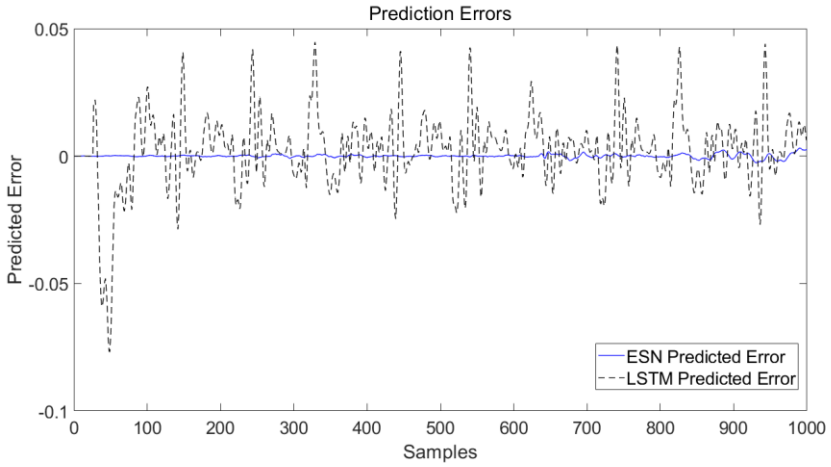


(c)

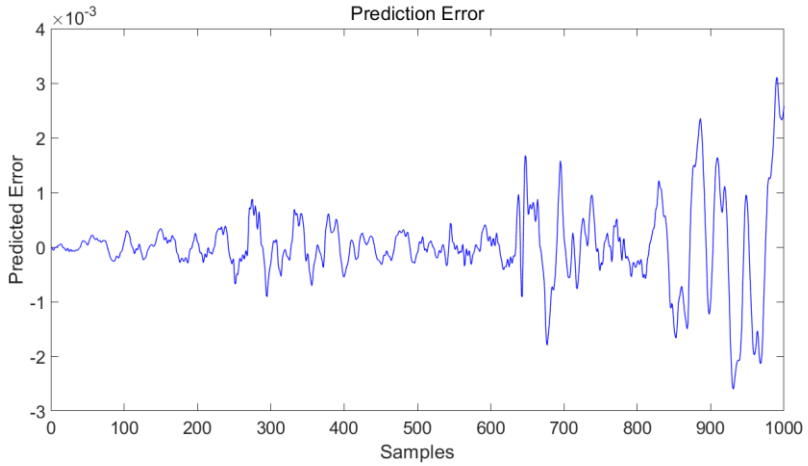
Figure 5.4 The variations of parameters of CMA-ES-ESN, (a) Values of fitness, step-Length, fitness fluctuation. (b) Distribution mean of reservoir size, leak rate and spectral radius factor, (c) Distribution of fitness for CMA-ES-ESN.



(a)



(b)



(c)

Figure 5.5 The prediction results of CMA-ES-ESN and LSTM for Mackey-Glass Time Series Data. (a) The actual and predicted signal. (b) The prediction errors. (c) The prediction error of CMA-ES-ESN.

5.5.2 Modelling of a Water Tank

In this case study, 2220 samples are generated from simulation. The first 740 samples are used to train the networks, the next 740 samples are testing data and the rest 740 samples are used as the unseen validation dataset. Noise with the distribution of $N(0, 2)$ was added to the input flow rate to represent the effects of measurement noise. The developed nonlinear dynamic model for all methods is of the following form:

$$\hat{y}(t) = f\big(\hat{y}(t-1), u(t-1)\big) \tag{5.16}$$

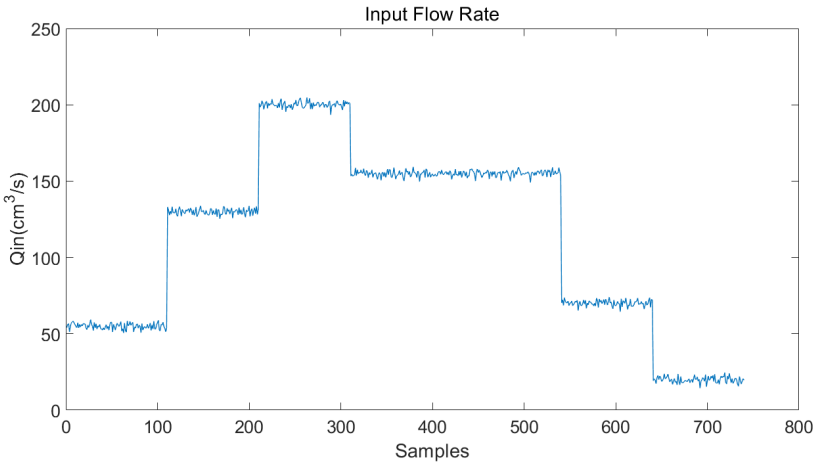
where $\hat{y}$ is the predicted tank level, u is the inlet flow rate, and t is the discrete time.

Figure 5.6 shows the long-range predictions of CMA-ES-ESN and LSTM comparing with the actual tank level on the unseen validation dataset, as well as the prediction errors and input signals. The MSE on the validation dataset after CMA-ES optimization is 0.0289 and the MSE on the testing dataset is 0.0157. It can be seen from Figure 5.6(c) that the error increases when the input signal, which is the inlet flow rate, changes suddenly for both CMA-ES-ESN and LSTM models, but the optimized ESN predictions get close to the actual signal more rapidly than LSTM predictions. This means optimized echo state networks have good stability and robustness on long-range predictions when the signal changes dramatically with one input and one previous output.

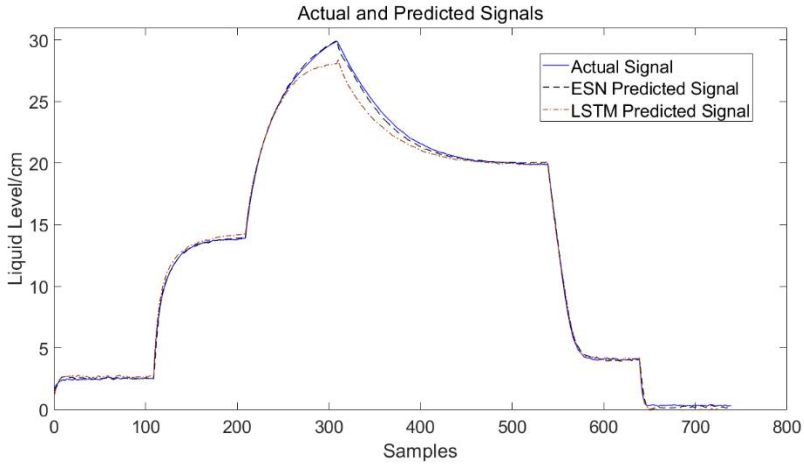
Table 5.2 shows the prediction MSE and time consumption of different methods. The MSE of CMA-ES-ESN is still the best among all results, besides, comparing to the FNN predicted performance of first application, FNN MSE on validation data is acceptable on this water tank level model, this also illustrates that for pure time-series data, RNN is the better choice.

Table 5.2 Model performance on validation data of Water Tank Level.

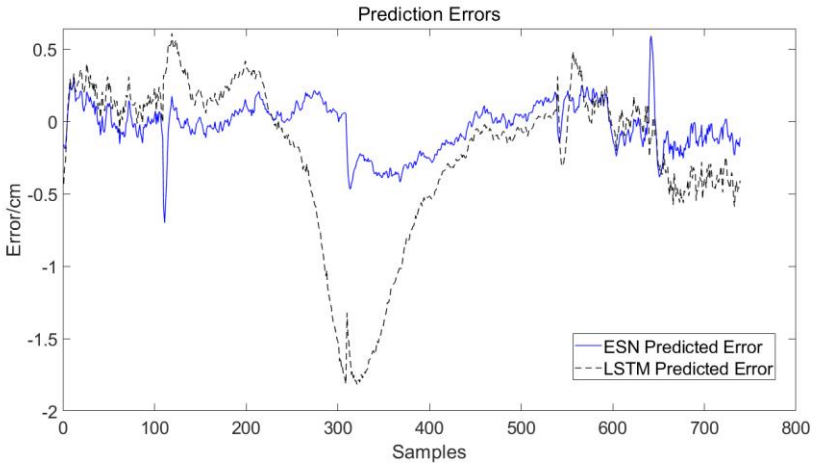
Approaches	MSE	Time(s)
GA-ESN-RWS	0.0997	294.3
CMA-ES-ESN	0.0289	36.8
Standard-ESN(50)	0.4179	4.2
LSTM	0.4195	133.9
FNN	0.8074	6.7



(a)



(b)



(c)

Figure 5.6 The predictions from CMA-ES-ESN for modelling of a water tank. (a) Input flow rate. (b) The actual and predicted tank level. (c) The prediction errors.

5.5.3 IndPenSim model

In this study, 5 variables are used as model inputs and they are shown in Figure 5.7. The variables all belong to manipulated variables which can be manipulated externally and they are phenylacetic acid flow (F_{PAA}), aeration rate (F_g), air head pressure (pressure), sugar flow rate (F_s), and water injection flow rate (F_w). In this model, the outputs include penicillin concentration which is the most important product in this fermentation process, biomass concentration and substrate concentration in the batch.

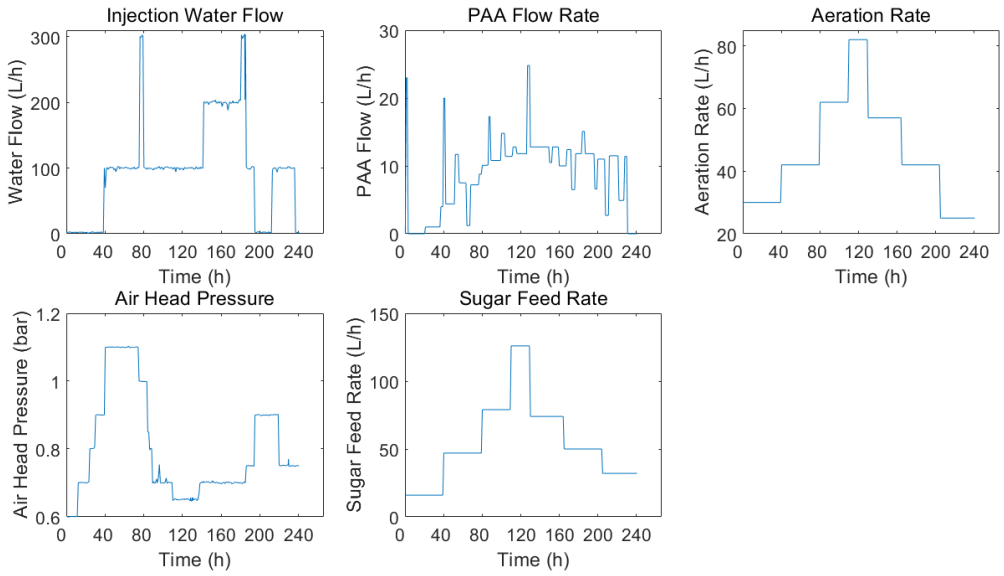


Figure 5.7 Model input variables.

5.5.3.1 Recurrent neural network model vs fedforward neural network model

In this part, prediction performance comparison between recurrent neural networks and feedforward neural networks conduct on complex fermentation process is illustrated. ESN and LSTM represent RNNs and one hidden layer FNN is compared. The unoptimized ESNs used in this part are all set as reservoir size being 400, leak rate being 0.9, sparse density being 0.1 and spectral radius factor being 0.9. The ESN model computes two outputs with one reservoir at the same time, which means the network has 5 inputs and 2 outputs. The developed nonlinear dynamic model is of the following form:

$$\begin{pmatrix}\hat{y}_1(t)\\ \hat{y}_2(t)\end{pmatrix} = f\big(u_1(t-1), u_2(t-1), u_3(t-1), u_4(t-1), u_5(t-1)\big) \quad (5.17)$$

While, the FNN and LSTM models are in the form given by Eq(5.18) and Eq(5.19) respectively:

$$\hat{y}_1(t) = f_1\big(u_1(t-1), u_2(t-1), u_3(t-1), u_4(t-1), u_5(t-1)\big) \tag{5.18}$$

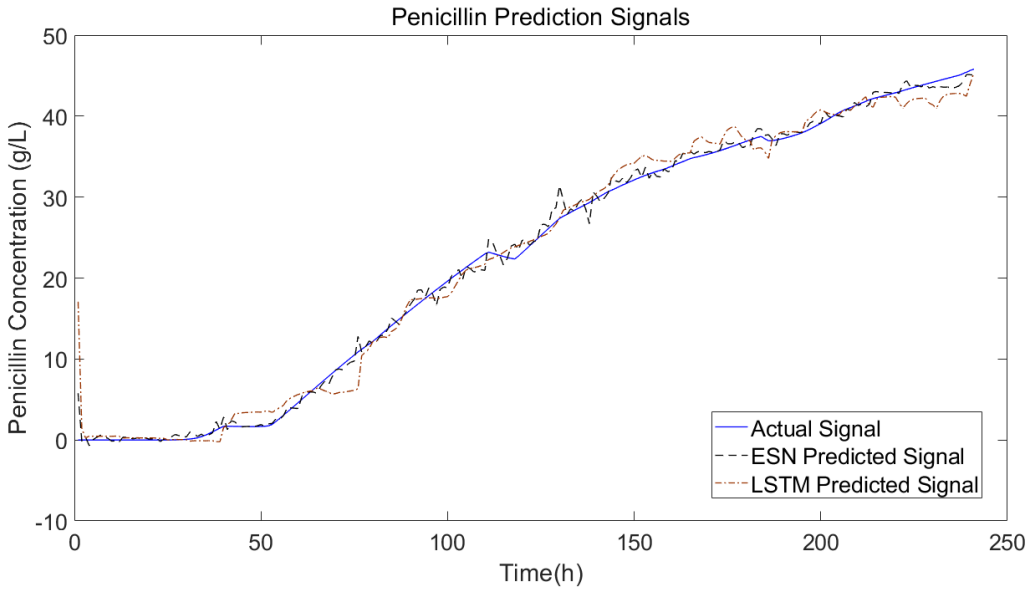
$$\hat{y}_2(t) = f_2\big(u_1(t-1), u_2(t-1), u_3(t-1), u_4(t-1), u_5(t-1)\big) \tag{5.19}$$

where u_1, u_2, u_3, u_4, u_5 represent injection water flow, PAA flow rate, aeration rate, air head pressure and suger feed rate, respectively, and $\hat{y}_1, \hat{y}_2$ are penicillin concentration and biomass concentration respectively.

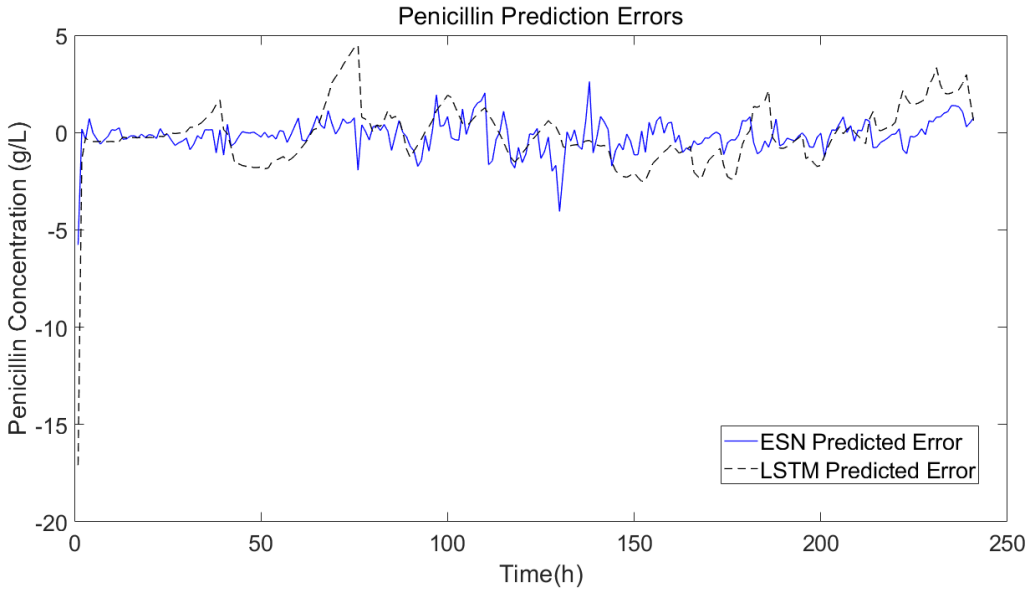
Because of the worse prediction results of FNN, the prediction signals are not shown in the figures. The long-range predictions of penicillin and biomass concentration on the unseen validation batch using the unoptimized ESN and LSTM are shown in Figure 5.8(a) and 5.9(a) respectively. The plots of prediction errors for biomass and penicillin concentrations of different scenarios are shown in Figure 5.8(b) and 5.9(b) respectively. The MSE of these models are shown in Table 5.3.

Table 5.3 MSE Comparison between unoptimized ESN, LSTM and FNN.

	Penicillin Prediction MSE	Biomass Prediction MSE
Unoptimized ESN	0.6840	0.1047
LSTM	0.7921	0.1582
FNN	267.63	4.428

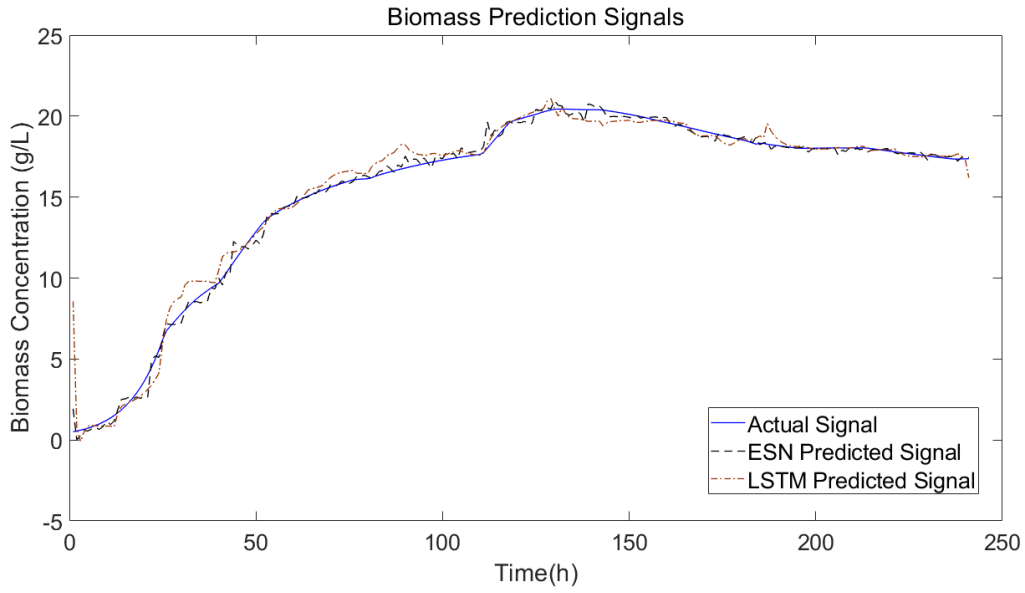


(a) Penicillin Prediction

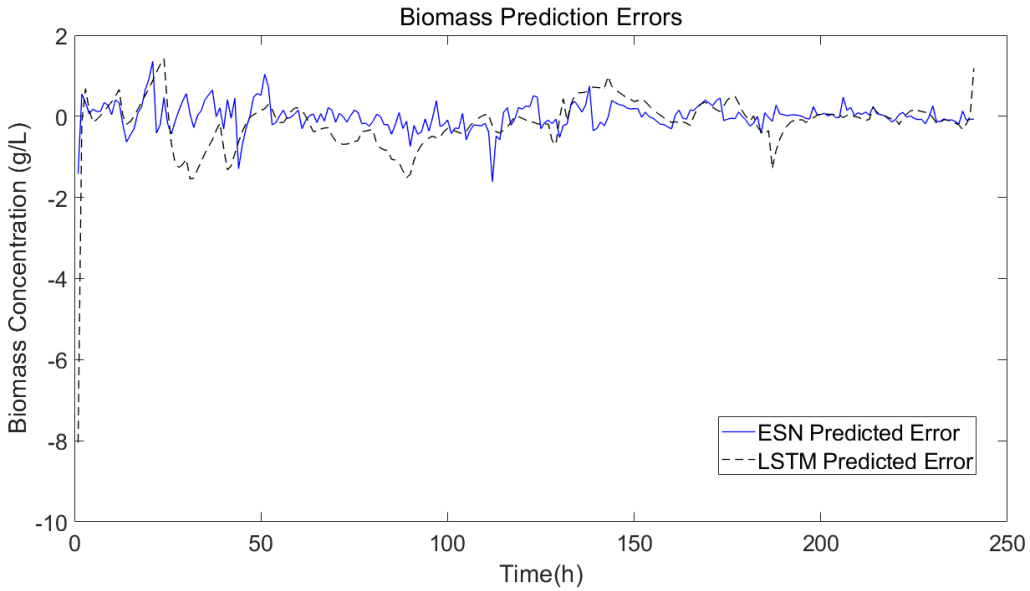


(b) Penicillin Predicted Error

Figure 5.8 Prediction results and error of ESN and FNN for Penicillin. (a) Penicillin concentration (b) Penicillin predicted error.



(a) Biomass Prediction



(b)Biomass Predicted Error

Figure 5.9 Prediction results and error of ESN and FNN for biomass (a) Biomass concentration (b) Biomass predicted error.

From Table 5.3, it can be found that the MSE of penicillin prediction by FNN is quite big, which means that the feedforward neural network is not a good choice to make prediction of penicillin concentration. As to the prediction result of biomass, the FNN prediction result is better than that of penicillin, but the error is still much bigger than that of ESN and LSTM. Comparing to FNN prediction results, both results of ESN and LSTM are much accurate, and also the predicted ability of unoptimized ESN and LSTM is similar,

the ESN result is moderately better than LSTM. Another advantage of ESN is the training consumption is much less than LSTM. The error of penicillin concentration is larger than biomass concentration because the penicillin generation process is more complicated but biomass concentration is related with sugar flow rate. The unoptimized ESN and LSTM performance is better than feedforward networks on this task, but the optimization process on ESN is still necessary.

5.5.3.2 Optimizing ESN model with CMA-ES

Echo state networks have been shown to work well with nonlinear complex fed-batch chemical process, but with the randomly generated reservoir weights and input connection weights, the ESN may not achieve satisfactory performance. Figure 5.10 shows the prediction fitness distribution of 500 randomly generated ESNs on validation dataset with the range of parameters given in Table 5.4. It can be seen that the number of fitness which is less than 1 is around 250 and most of fitness gather in the range from 0 to 10. In order to dig more potential of ESN using in this task, CMA-ES is used to optimize the structure of ESN. The ranges of parameters which need to be optimized and CMA-ES initial step size is also given in Table 5.4.

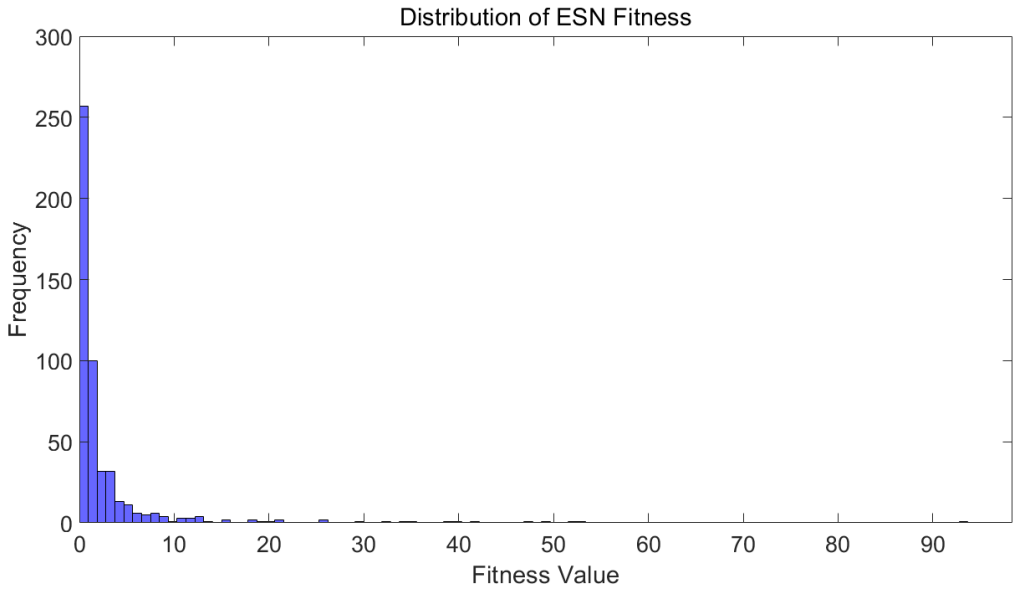
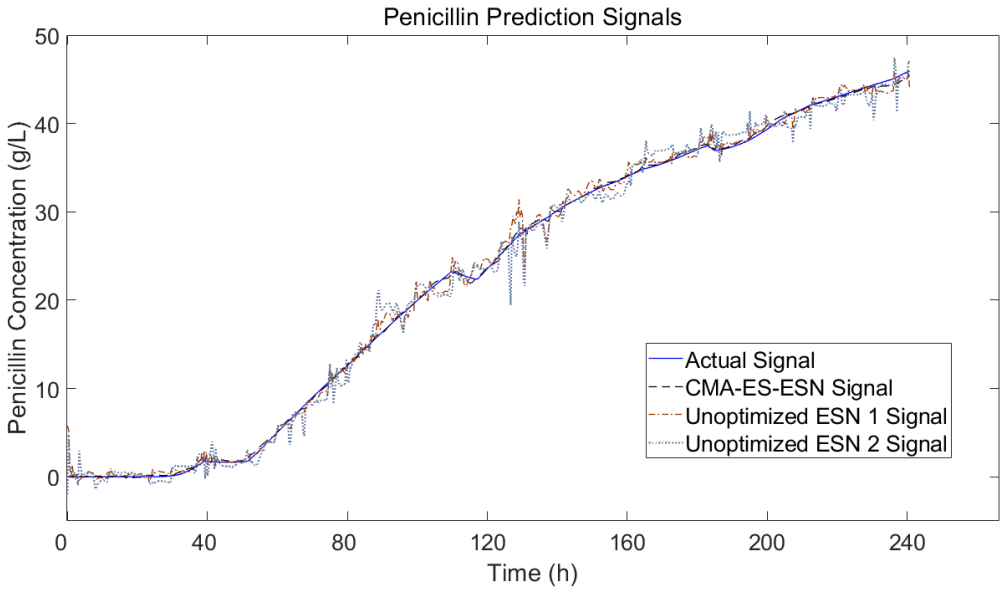


Figure 5.10 Distribution of ESN fitness.

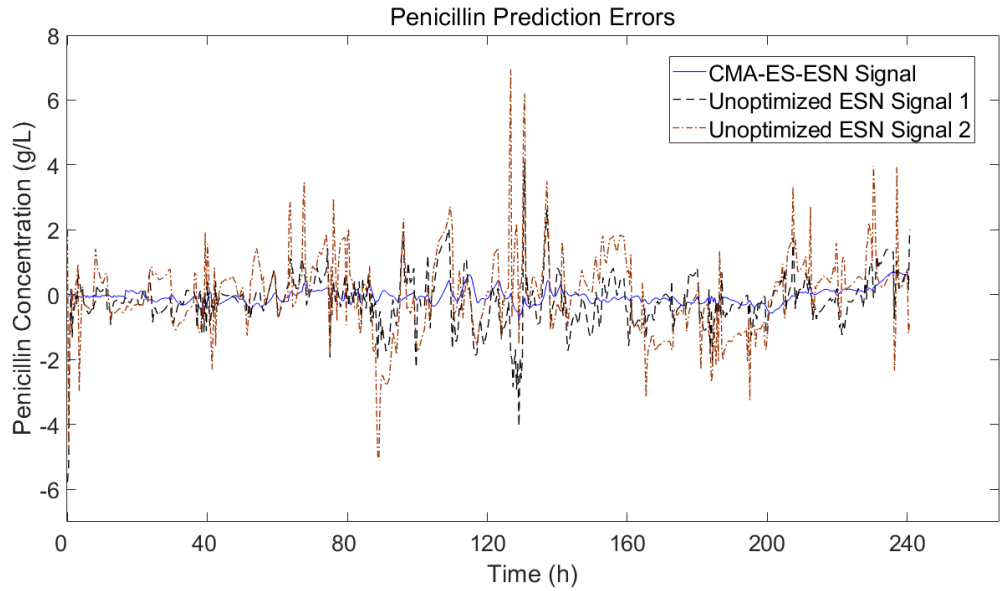
Table 5.4 Ranges of ESN parameters and CMA-ES initial step size.

Range of reservoir size	100~1000
Range of leak rate	0~1
Range of spectral radius factor	0~1.5
CMA-ES initial step size	0.2

The ESNs optimized by CMA-ES algorithm and GA-RWS algorithm are compared with 2 normal randomly generated sparse reservoir ESN. The trend plots of penicillin concentration and biomass prediction error are shown in Figure 5.11(a) and 5.11(b), while Figure 5.12(a) and 5.12(b) display biomass concentration prediction and prediction error plot. The mean square errors of all these ESNs are indicated in Table 5.5.

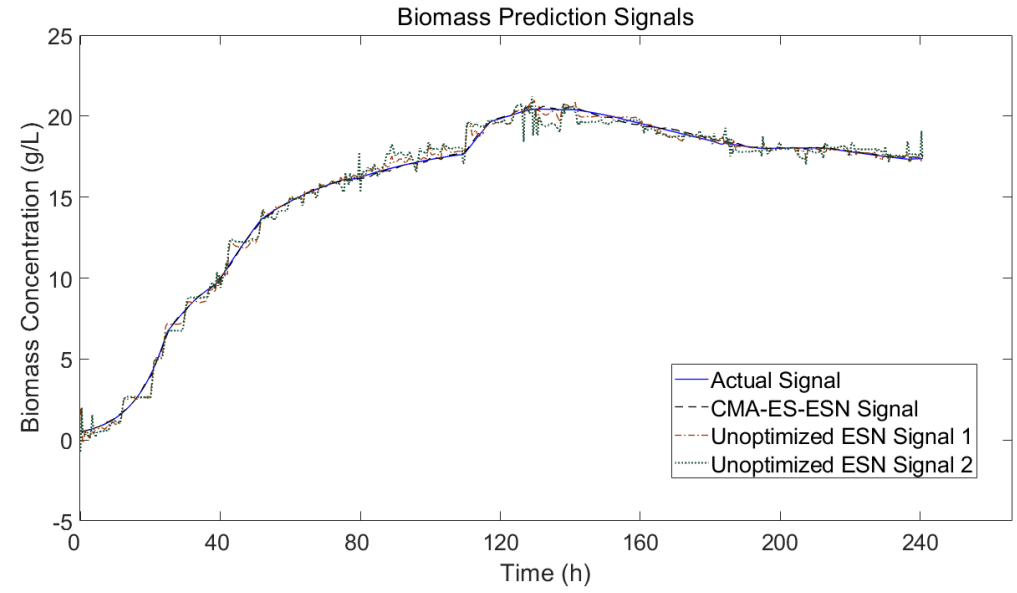


(a)Penicillin Prediction

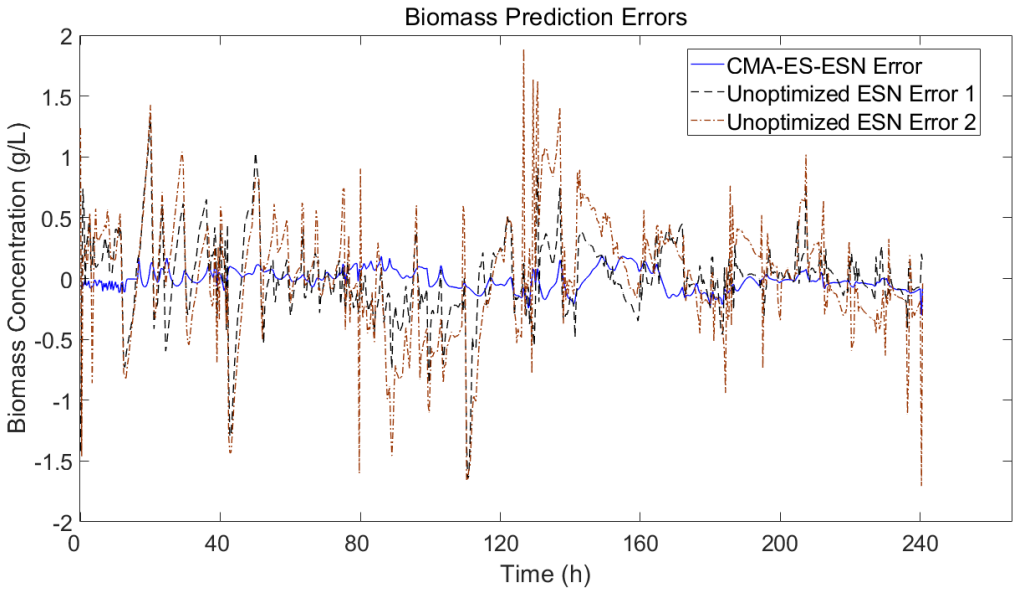


(b)Penicillin prediction error

Figure 5.11 Prediction results and error of CMA-ES-ESN and unoptimized ESN on validation data. (a) Penicillin concentration (b) Penicillin prediction error.



(a) Biomass Prediction



(a) Biomass prediction error

Figure 5.12 Prediction results and error of CMA-ES-ESN and unoptimized ESN on validation data. (a) Biomass concentration (b) Biomass prediction error.

Table 5.5 MSE of different echo state networks.

	Penicillin Concentraion	Biomass Concentraion
CMA-ES-ESN	0.05548	0.005841
GA-ESN-RWS	0.1247	0.0176
Unoptimized ESN 1	0.6840	0.1047
Unoptimized ESN 2	1.5149	0.2499

Table 5.6 Time consumption on penicillin fermentation task.

	Time(s)
CMA-ES-ESN	1270
GA-ESN-RWS	2312
LSTM	1460

Table 5.5 shows the MSE of predictions for penicillin and biomass concentration and Table 5.6 shows time consumption. From the plots shown in Figure 5.11 and Figure 5.12 and the MSE values given in Table 5.5, comparing with randomly unoptimized ESNs, the CMA-ES-ESN with sparse reservoir performs superiorly in modelling the fed-batch penicillin cultivation process. The prediction performance on biomass concentration is even better than penicillin concentration which is 10 times better than unoptimized ESN models. Therefore, the proposed method is a better choice for modelling the fed-batch penicillin process. The proposed method would be also suitable for other batch processes.

5.6 Conclusions

In this chapter, a method which uses the covariance matrix adaption evolution strategy (CMA-ES) for initialization of echo state networks is proposed. ESN has been shown to be a rapid, efficient and accurate dynamic system modelling algorithm, moreover, the CMA-

ES algorithm has been successfully utilized for optimization in complex space. The ESN structural parameters including reservoir size, leak rate and spectral radius factor are optimized via CMA-ES. Modelling a nonlinear time series, a conic water tank, and a fed-batch penicillin fermentation process are used to test the ESN. Comparing with the original ESN, LSTM, GA-ESN, CMA-ES-ESN has shown the ability to improve the model prediction accuracy, moreover, regarding to time consumption results of different methods, CMA-ES is proved to be a fast convergence algorithm. When modified topology reservoir cannot optimize ESN performance significantly, covariance matrix adaption evolution strategy is a suitable method to build reservoir stucture in ESN by working with different models.

Chapter 6 An echo state networks with attention mechanism

6.1 Introduction

Multi-input high-dimensional models are encountered in many disciplinary applications such as finance, chemical and biological applications, such as the fed-batch penicillin fermentation process used in Chapter 4 and Chapter 5. For models with multiple inputs, not all input variables have significant effect on the output variables. Some input variables have a great impact on the output variables, whereas some inputs have no contribution to the output variables. Thus, it is possible to eliminate inputs that have no effect on the model outputs. Moreover, the fluctuations in the model output variables are not only related to fluctuations in the input variables, but also affected by the interactions between the input variables. It is possible to eliminate the fluctuations in the output by adjusting the interactions between the inputs. By examining the contribution of the input variables to the output variables, the output variables can be better modelled. In this chapter, an attention-based ESN is proposed to adjust the scale of inputs to improve the network ability on modelling tasks involving multiple inputs.

The remaining part of this chapter is organized as follows. Section 6.2 gives the introduction of the attention mechanism and in Section 6.3, the CMA-ES optimized attention-based ESN is proposed. In Section 6.4 and Section 6.5, the experiments to evaluate the proposed network and results are given, respectively.

6.2 Attention mechanism

In everyday life, our brains receive a great amount of input through our various senses, which are of different types and carry different semantic meanings, and our brains are able to process them in an organised manner. In this process, attentional mechanisms play a crucial role in the complex cognitive functions of the human brain, enabling the brain to consciously process and respond to information through the perception of associations between information and events, and the selection of information. The human attentional mechanism is a way for the human perceptual system to selectively focus on feedback from local stimuli and is part of the human cognitive process. Human attention increases sensitivity to input signals at the cellular synaptic level, thereby sharpening the accuracy of the input signal, selectively increasing the transmission of important signals and reducing the noise level within them, with the aim of reshaping neural sensations. Among the mechanisms of human attention, the visual system is the most typical. Signals from the

physical world (e.g. colour, shape, direction of movement, etc.) are transmitted through the visual pathway to the higher prefrontal cortex (PFC) and posterior parietal cortex (PPC) in the brain, creating a salience map that directs the brain's attention to the more salient stimulus signals. The brain modulates the information in the visual pathway according to the goal of the task and the knowledge learned in the past, and when attention is modulated to the salience area, the prefrontal lobe controls eye movements through the superior colliculus, allowing us to focus on the more salient stimulus signals. Thus, attentional mechanisms allow our brain to focus on task-relevant information by filtering it in a task-oriented manner, thus increasing the efficiency of information processing and use (Katsuki and Constantinidis, 2014).

Similar to the biological neuron system in the human brain, the artificial neural network has a network capacity limit and is limited in its ability to store information. In general, the amount of information that a neural network can store is proportional to the number of neurons and the complexity of the neural network. Therefore, storing more information requires more neurons and a more complex network structure, which not only multiplies the computational complexity, but also makes the neural networks training much more difficult. In the face of information overload, machine learning has also taken a cue from the way the human brain processes information, using an attention mechanism that allows neural networks to selectively accept and process information, improving their ability to process, analyse and understand large-scale data. Artificial intelligence algorithms have achieved impressive results at the perceptual level by fitting statistical information to induction, more intelligent machine learning algorithms still require more powerful logical inference and biased induction. Thus, attentional mechanisms are the core technology needed to enable artificial intelligence systems to operate consciously. In fact, different data types, different objects and different task goals have different functional requirements for attention mechanisms. This requires an in-depth knowledge and understanding of the mechanics of different attentional approaches, and an appropriate modelling approach to the design and use of attentional mechanisms for different application requirements. Although attentional mechanisms have already brought significant improvements to many of the learning tasks mentioned above, there is still much room for improvement in the efficiency, performance and scope of neural network-oriented attentional mechanisms compared to attentional mechanisms in human cognitive functions.

The study of attentional mechanisms has a long history of development, with the main ideas dating back to as early as 1990s. Deepmind combines the attention mechanism with recurrent neural networks, the application to image classification tasks has achieved good performance and has attracted much attention to attentional mechanisms (Mnih et al., 2014). Then the attention mechanism has been introduced into Natural Language Processing (NLP), joining with translation and alignment, which extends the range of applications of attention mechanism (Bahdanau et al., 2015). In 2017, self-attention has been introduced in the application of machine translating problem (Vaswani et al., 2017).

The attention mechanism is an explicit attention method that obtains attention weights by predicting the importance of each component of the data to the task optimisation goal, and then uses the attention weights to explicitly enhance the important components of the data (or features) and suppress the components of the data (or features) that are not relevant to the task optimisation goal.

Attention mechanisms have been widely used in a variety of information processing and analysis tasks, including verbal and visual signals (Maruf et al., 2019, Maidhof and Koelsch, 2011, Chen et al., 2017), and have greatly improved the non-linear representational capabilities of neural networks and the abstraction of high-level semantics. The selective attention mechanism takes the data itself as input, uses the neural network learning to generate an attention mask as a prediction of the importance of each part of the data, and then uses this attention mask to enhance or suppress the feature.
The attention mask $a(\boldsymbol{u})$ can be obtained according to the following equation,:

$$a(\boldsymbol{u}) = F_{norm}\left(F_{score}(\boldsymbol{u}, \boldsymbol{x})\right) \tag{6.1}$$

where $\boldsymbol{u}$ denotes the input matrix and $\boldsymbol{x}$ is internal state of reservoir, and F_{score} denotes the function transformation corresponding to the attention model, F_{norm} denoting the normalization operation.

For the learned attention mask $a(\boldsymbol{u})$, it is usually multiplied by the corresponding element of the original signal $\boldsymbol{u}$ to select the information in the original signal, a process that is formulated as follows:

$$\widetilde{\boldsymbol{u}} = a(\boldsymbol{u}) \odot \boldsymbol{u} \tag{6.2}$$

where $\tilde{\boldsymbol{u}}$ is the signal after enhancement or suppression of original signal x and $\odot$ denotes element-wise multiplication. The attention mask $a(\boldsymbol{u})$ is also divided into soft attention and hard attention. When the attention mask is taken continuously values within a certain range, this is called soft attention mechanism, it enhances and suppresses different components of the original signal. When $a(\boldsymbol{u})$ takes the value of 0 or 1, remove or retain the different components of the original signal, this is hard attention mechanism.

For the backpropagation-based networks, such as normal RNNs and LSTM, F_{score} is usually a learnable function, implemented by a neural network, that learns the degree of attention to different components of the input $\boldsymbol{u}$. To calculate the score function F_{score} between the input infromation $\boldsymbol{u}$ and the internal state of networks $\boldsymbol{x}$, there are several methods that can be used (Luong et al., 2015):

$$F_{score}(\boldsymbol{u},\boldsymbol{x}) = \begin{cases} \boldsymbol{u}\cdot\boldsymbol{x} & dot \\ \boldsymbol{u}^T\boldsymbol{W}_a\boldsymbol{x} & general \\ \boldsymbol{v}_\alpha^T\cdot\tanh(\boldsymbol{W}_a[\boldsymbol{u}:\boldsymbol{x}]) & concat/additive \end{cases} \tag{6.3}$$

The normalisation operation F_{norm} limits the learned range of attention weights, which can be implemented by functions such as SoftMax, Sigmoid, and Tanh. It can be normalised in different dimensions depending on the task.

In this chapter, the general score method is used to calculate the relationship between input data and internal state data. Because in ESN the only learnable weight is output weight, $\boldsymbol{W}_{out}$ can not be updated through the training, $\boldsymbol{W}_a$ is updated by the CMA-ES.
In this chapter, Sigmoid function is used as F_{norm} to implement the attention mechanism:

$$Sigmoid(x) = \frac{1}{1+e^{-x}} \tag{6.4}$$

The variable-length alignment attention vector $\boldsymbol{a}_t$ can be obtained by:

$$\begin{aligned} \boldsymbol{a}_t &= align(\boldsymbol{u},\boldsymbol{x}) \\ &= Sigmoid(F_{score}(\boldsymbol{u},\boldsymbol{x}) \\ &= Sigmoid(\boldsymbol{u}^T\boldsymbol{W}_a\boldsymbol{x}) \end{aligned} \tag{6.5}$$

6.3 The Attention-based CMA-ES-ESN

In this chapter, CMA-ES is used to optimize the structure parameters of ESN which are reservoir size, leak rate and spectral radius with attention mechanism for fed-batch

complex penicillin fermentation process modelling. Figure 6.1 shows the flow chart of the proposed algorithm. The procedure can be summarized as follows:

1. Data for model building are divided into three sets: training data, testing data, and unseen validation data, and then they are normalized to have zero mean and unit variance.
2. Establish an ESN with random N, α and ρ in the range based on sufficient internal units as default. The activation function used here in the hidden layer (reservoir) is $f = \tanh$ and the input weights and reservoir weights are generated randomly.
3. Train the established ESN with training data using ridge regression. Optimize the Atten-ESN by CMA-ES. The MSE on the testing data is used as objective function. The optimization objective is to upgrade the values of N, α and ρ to minimize the MSE on the testing data.
4. Establish ESN with optimized N, α and ρ.
5. Train the optimized ESN with training data to get the internal reservoir weights $\boldsymbol{x}$.
6. Optimized the attention weighted matrix $\boldsymbol{W}_\alpha$ coming from input data $\boldsymbol{u}$ and internal reservoir matrix $\boldsymbol{x}$, then the variable length attention mask $\alpha(\boldsymbol{u})$ is generated and updated the input data $\tilde{\boldsymbol{u}}$ is achieved.
7. Test the optimized Atten-ESN (O-Atten-ESN) on the unseen validation data.

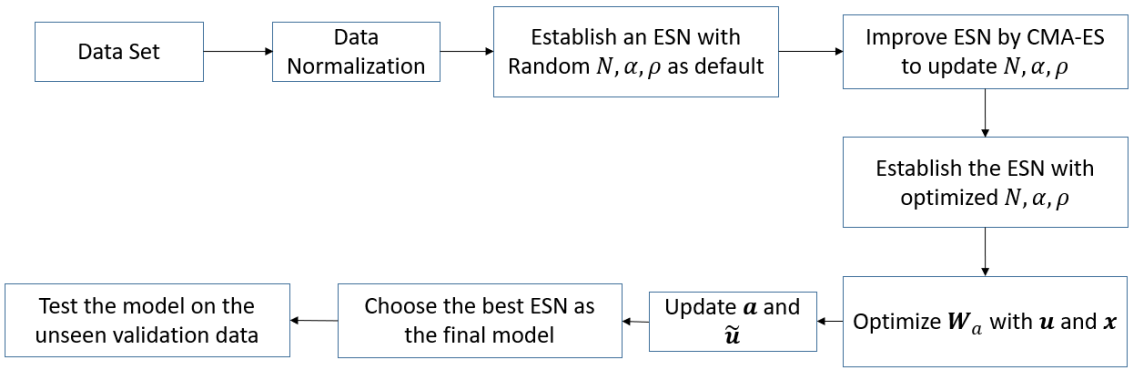


Figure 6.1 Graphical illustration of the proposed approach.

6.4 Experiments

The benchmark industrial penicillin fermentation simulator, IndPenSim (Goldrick et al., 2015), is used to produce simulated process operation data. In this study, 10 benches of data generated by IndPensim are used in model development. Among these data, 8 batches

are used as training data, one batch is used as testing data, and the final batch is used as the unseen validation data. In this chapter, the penicillin concentration is taken as the target output and 14 controllable and monitoring variables are used as model inputs. The manual control variables are aeration rate, sugar feed rate, injection water flow rate, air head pressure, nitrogen flow rate, Phenylacetic acid flow rate and Soybean oil flow rate, meanwhile, the automatic control variables are acid/base flowrate and heating/cooling water flowrate, the pH, temperature and dissolved oxygen concentration are including in on-line measurement variables. Figure 6.2 shows the examples of 14 inputs variables and Table 6.1 shows input variables. The fitness function to minimize is the MSE of the ESN on the training data.

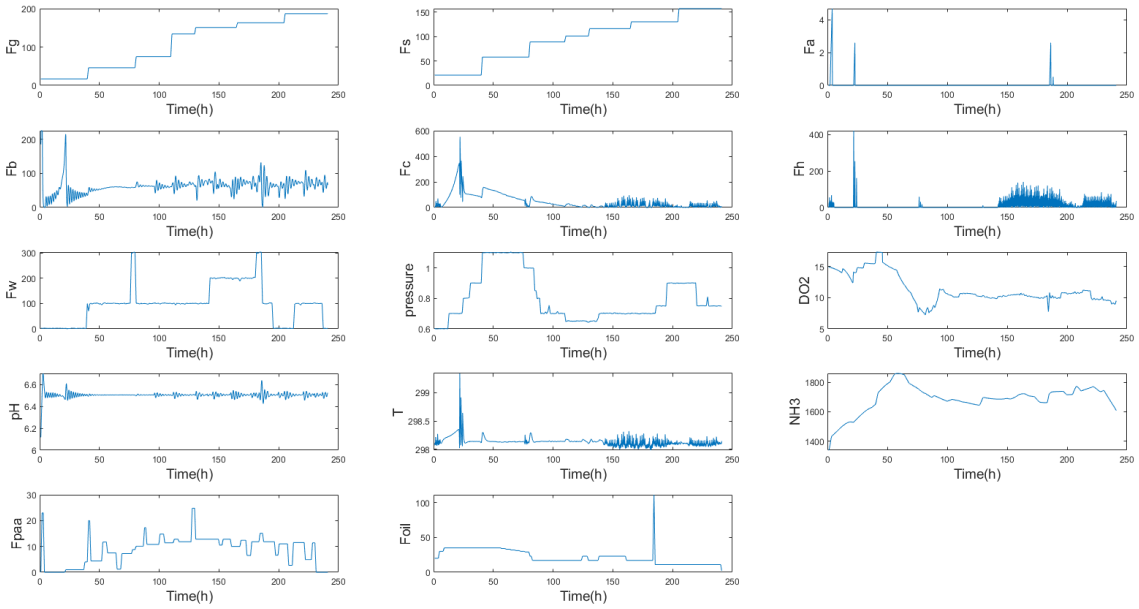


Figure 6.2 The trend plots of 14 inputs variables in penicillin process.

Table 6.1 The model input and output variables selected in this study.

NO	Input Parameters	Description	Unit
1	F_g	Aeration rate	$m^3\ min^{-1}$
2	F_s	Sugar flow rate	$L\ h^{-1}$
3	F_a	Acid flow rate	$L\ h^{-1}$
4	F_b	Base flow rate	$L\ h^{-1}$
5	F_c	Cooling water flow rate	$L\ h^{-1}$
6	F_h	Heating water flow rate	$L\ h^{-1}$
7	F_w	Water for injection flow rate	$L\ h^{-1}$
8	Pressure	Air head pressure	bar
9	DO_2	Dissolved oxygen concentration	$mg\ L^{-1}$
10	pH	pH	
11	T	Temperature	K
12	NH_3	NH_3 concentration	$g\ L^{-1}$
13	F_{paa}	Phenylacetic acid flow	$L\ h^{-1}$
14	F_{oil}	Soybean oil flow rate	$L\ h^{-1}$

6.5 Results

In order to evaluate the CMA-ES optimized attention-based ESN's (O-Att-ESN) performance on complex multiple input variables penicillin fermentation modelling, non-optimized attention-based ESN (A-ESN), conventional ESN, long short-term memory network, and attention long short-term memory network are applied, the following procedures are compared:

- ESN: For all ESN model and hybrid ESN model, every reservoir node has the activation function $\tanh(x) = \frac{\sin h(x)}{\cos h(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$; the input and backwards weights $\boldsymbol{W^{in}}$, $\boldsymbol{W^{back}}$ are fully generated by uniform distribution in the range [-0.5,0.5], and similarly the internal weights $\boldsymbol{W}$ are created by uniform distribution in the range [-0.5,0.5] with different sparse density and reservoir size. In the conventional ESN, three structural parameters, reservoir size, leak rate and spectral radius factor, are selected

randomly in the ranges of [1-1000], [0-1] and [0-1.5], respectively. 50 ESNs with different random seeds were generated and select the best model among the 50 models.

- O-ESN: In O-ESN, the three structural parameters to be optimized are reservoir size, leak rate and special radius with the ranges of [1-1000], [0-1] and [0-1.5], respectively, and according to $\gamma = 4 + 3\ln n$, when $n = 3$, γ is found to be 7, i.e. the population size is 7, and initial search step length is set at 0.2. The stopping criterion is when the maximum iteration 9000 is met or if the minimal fitness value stops decreasing for a consecutive 40 iterations.
- O-Att-ESN: In O-ATT-ESN, the CMA-ES optimized setting is same as O-ESN, the updated input data $\widetilde{\boldsymbol{u}}$ replaces the original input data $\boldsymbol{u}$.
- Att-ESN: In Att-ESN, the reservoir size, leak rate and special radius are as same as the best conventional ESN, after the optimization, the updated input data $\widetilde{\boldsymbol{u}}$ replaces the original input data $\boldsymbol{u}$.
- LSTM: Long short-term memory network is established by Deep Learning Toolbox from MATLAB®. The LSTM layer has 100 hidden units, the solver is set to 'Adam', and the network is trained for 500 epochs. To prevent the gradients from exploding, the gradient threshold is set to 1. The initial learn rate was 0.005 and was dropped after every 100 epochs by multiplying a factor of 0.2.
- Att-LSTM: The attention-based LSTM is also established by Deep Learning Toolbox from MATLAB®. The attention function computes the attention scores according to the 'general' scoring as well, and the attention score is calculated by SoftMax function. The other settings are same as LSTM.

In this complex multiple input variables penicillin fermentation modelling problem, the one-step predicted model is of the following form:

$$\hat{y}_1(t) = f(u_1(t-1), u_2(t-1), u_3(t-1), u_4(t-1), u_5(t-1), u_6(t-1), u_7(t-1), u_8(t-1), u_9(t-1), u_{10}(t-1), u_{11}(t-1), u_{12}(t-1), u_{13}(t-1), u_{14}(t-1)) \quad (6.6)$$

The penicillin concentration prediction results of Att-ESN, Att-LSTM, ESN and LSTM are presented in Figure 6.3 and prediction errors of these methods are displayed in Figure 6.4, moreover, the results of O-Att-ESN, O-ESN and Att-ESN are shown in Figure 6.5 and the corresponding errors are illustrated in Figure 6.6. The validation MSE values of corresponding methods are shown in Table 6.2. It can be found in the Figure 6.3 that

comparing to the predictions of conventional ESN and LSTM, the predictions of Att-ESN and Att-LSTM are closer to target values. Both in the ESN and LSTM curves, there are significant offsets from the desired signal, in the same part of Att-ESN and Att-LSTM curves, the differences to the desired signal are much smaller, this also can be observed from Figure 6.4, the peaks of error curves for Att-ESN and Att-LSTM methods are in the similar shape with those of ESN and LSTM, but peaks of the attention-based methods are significantly smaller than those of conventional networks. From above, it can be seen that the attention mechanism can take out some relevant inputs signals to increase their impact and otherwise, the irrelevant variables are weakened the influence. From Figure 6.5, it can be found that the O-Att-ESN predictions are closest to the truth values, especially in the first 50 hours. The optimized ESNs are flatter than unoptimized ESN. In Figure 6.6, there is similar situation as observed in Figure 6.4, the shape of peaks are similar among the three networks, but those of optimized networks are smaller. This epitomizes that the on the basis of the attention mechanism, the CMA-ES can optimize the ESN further and both attention mechanism and optimized structure parameter can are beneficial for final predication signals.

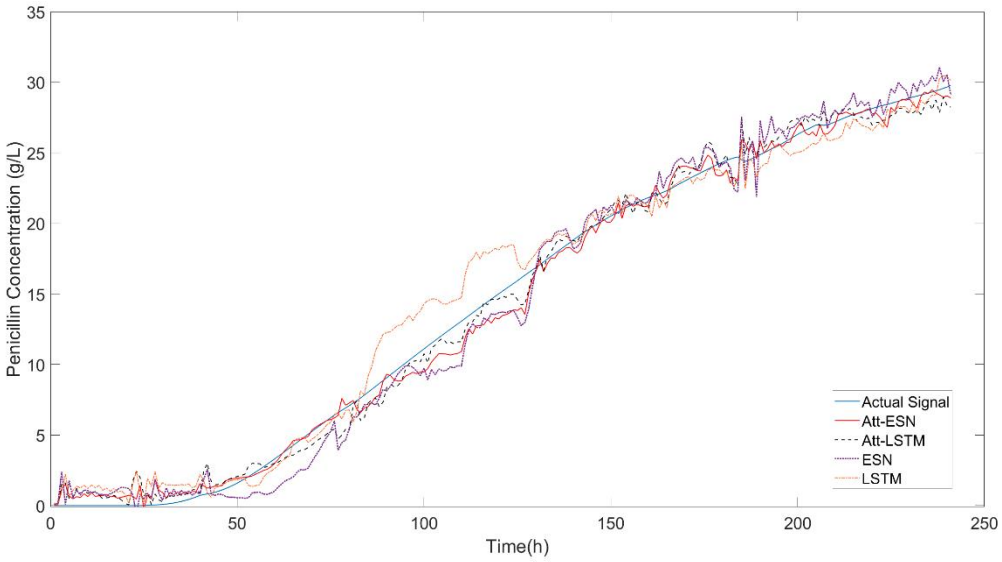


Figure 6.3 The prediction of penicillin concentration by Att-ESN, Att-LSTM, ESN and LSTM.

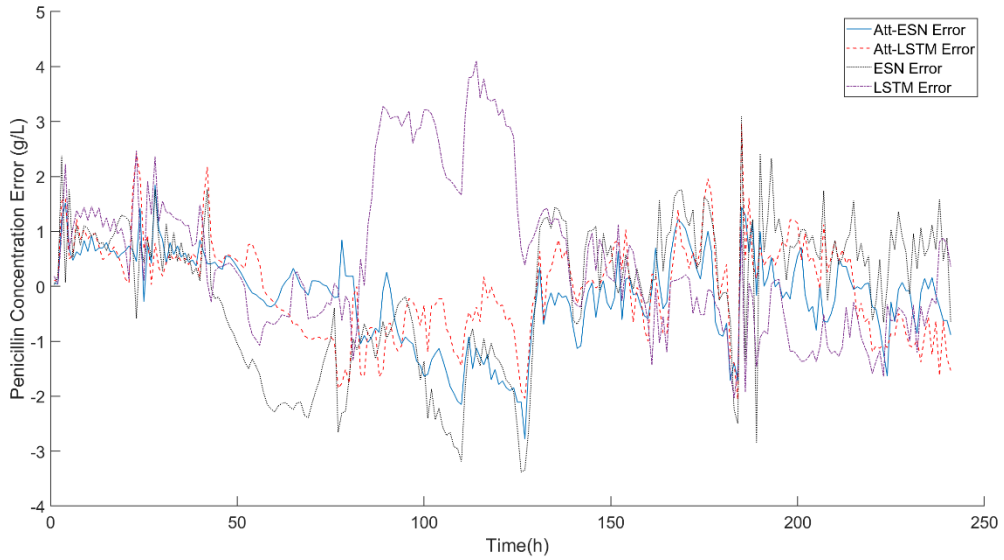


Figure 6.4 The prediction error of penicillin concentration by Att-ESN, Att-LSTM, ESN and LSTM.

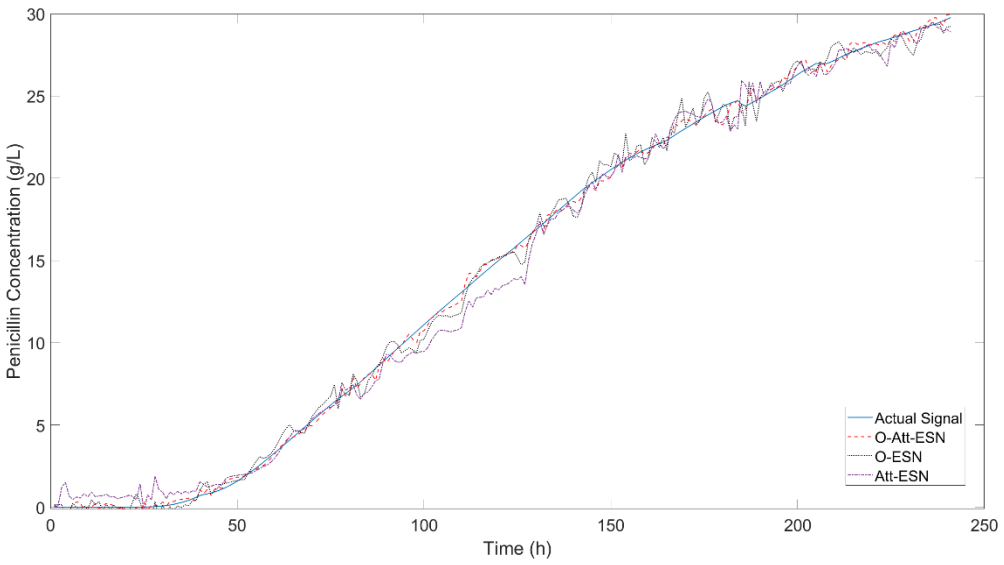


Figure 6.5 The prediction of penicillin concentration by O-Att-ESN, O-ESN, Att-ESN.

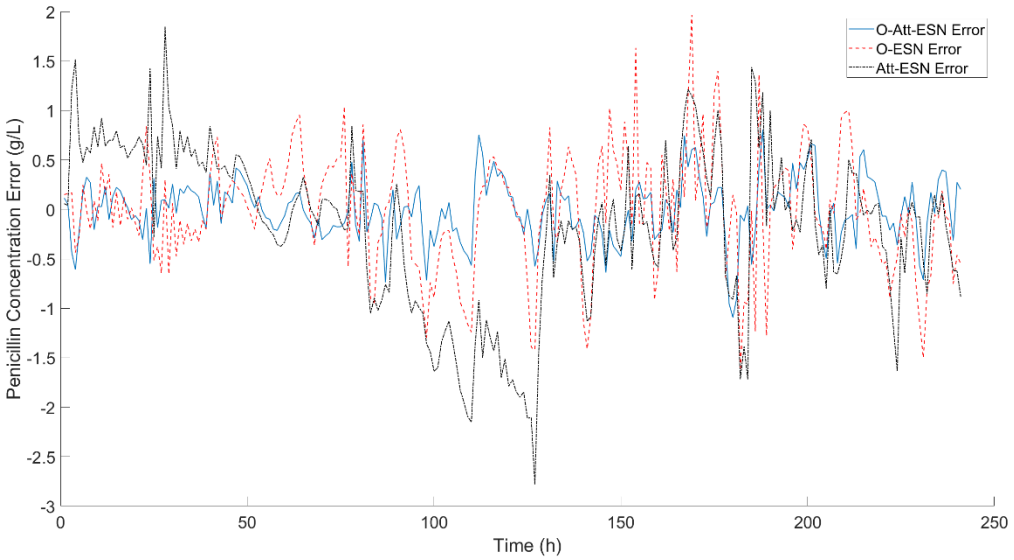


Figure 6.6 The prediction error of penicillin concentration by O-Att-ESN, O-ESN, Att-ESN.

Table 6.2 The MSE of validation data on penicillin fermentation process.

Modelling Methods	MSE
O-Att-ESN	0.095
O-ESN	0.342
Att-ESN	0.6868
Att-LSTM	0.791
ESN	1.976
LSTM	2.214

6.6 Conclusions

In this chapter, an optimized attention mechanism based ESN has been proposed and this enables the ESN to have the attentiveness capacity, and the important levels of distinct variables in input vectors will be addressed in an adaptive manner according to their relevance. In the multiple inputs penicillin fermentation process, the results show that the attention-based methods give more accurate predictions than the conventional ESN and LSTM. Furthermore, after the key structure parameter optimized by CMA-ES, the proposed network' performance is best of all, this illustrates that the inputs attention score has the similar impact level with the other parameters of ESN as: reservoir size, leak rate

and spectral radius. The proposed O-Att-ESN can be used in other multiple input applications as well.

Chapter 7. Conclusions and recommendations for future works

7.1 Conclusions

The Reservoir Computing (RC) is a novel and simple machine learning idea that has emerged in recent years. Echo State Network (ESN) is one of the simplest and most effective RC, in which the reservoir is a recursive structure of large scale and composed of a large number of sparse neuron connections, the reservoir can handle nonlinear systems and time series signals very well. The ESN is a class of dynamic models, and its process of handling input data can be separated into two parts: firstly, the dynamic reservoir processes input data as a complex non-linear dynamic filter, it converts the low-dimensional input data into high-dimensional hidden state data, secondly, a simple linear regression is performed on the transformed high-dimensional hidden state data. Although ESN has yielded good results in both academic and applied fields in recent years, the current research on ESN has addressed the problems which impedes the development such as: difficult to understand reservoir properties, black-box reservoir properties, lack of principled ESN design, difficulty in determining the size of the reservoir, etc. To address these problems, this thesis has proposed several methods to improve the ESN's performance on nonlinear process modelling, including the small world based modular reservoir topology, which can enhance the ESN's prediction accuracy and robustness, the adaptative genetic algorithm optimized ESN can reinforce the modelling capacity on specific cases, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) optimized ESN can search the structure parameters in a continuous domain with a rapid convergence speed, and attention mechanism based ESN to handle the complex multiple input data.

In response to the traditional reservoir topology, from the perspective of structural bionics, a new reservoir topology of ESN, called modular reservoir topology, is proposed by mimicking the brain networks with modular topological features. The modular reservoir topology is designed to achieve partial decoupling of neurons within the reservoir, which enhances the richness of dynamical properties within the reservoir and thus improves the performance of ESN. Based on this, a modular small world ESN (MSM-ESN) is constructed and its performance is verified through extensive modelling experiments. From the results, the proposed reservoir topology is shown to have better predictive performance than the conventional ESN.

In traditional randomly generated ESN, a good performance ESN is always created by experience or continuous trial, in order to solve these problems, an adaptative genetic algorithm optimized echo state network is proposed. Genetic algorithm has been proved to be a robust, practical, efficient and widely applicable method for global optimization problems. Four structural parameters of ESN which are reservoir size, leak rate, spectral radius and sparseness density are optimized by genetic algorithm to overcome the disadvantage of randomly generated ESN. Further. Compared to the traditional genetic algorithm with fixed crossover and mutation probabilities, the proposed adaptative genetic algorithm can adjust the crossover and mutation probabilities by the fitness distribution of each generation. From the experiments results, genetic algorithm optimized ESN has dramatical advantages on prediction accuracy to the conventional random ESN. Furthermore, adaption mechanism based genetic algorithm has higher optimizing efficiency and more chance to escape form the local optimum.

Although genetic algorithm can optimize the ESN, because of the coding strategy, it still faces the problems that the search domain is discrete and large computation cost comes from the big population size. To solve these problems, covariance matrix adaptation evolution strategy is proposed to optimize the ESN on the selected structural parameters. The CMA-ES is a well-established evolutionary algorithm for real-valued optimization with many successful applications. The main advantages of CMA-ES lie in its invariance properties, which are achieved by carefully designed variation and selection operators and its efficient adaptation of the mutation distribution. The CMA-ES is invariant against order-preserving transformations of the fitness function value and in particular against rotation and translation of the search space. From the experiments results, the predicted accuracy of CMA-ES optimized ESN is better than the genetic algorithm optimized ESN and the time consumption of computation of CMA-ES-ESN is much less than GA-ESN.

Finally, in multiple inputs ESN modelling problems, the input scale parameters can influence the ESN modelling performance, to solve this problem, attention mechanism based ESN is proposed. Attention mechanism has gained a lot of attention in recent years and it has been proved to be a powerful tool in many applicants. Attention mechanism is used in the ESN to rescale the input scale parameters to endow different weights on input data. From the experiment results, the attention mechanism can enhance the predicted

accuracy on multiple inputs modelling, meanwhile, the input scale parameters of ESN has same level influence comparing with the structure parameters.

7.2 Future works

This thesis proposed some optimizations on echo state networks and applied the optimized ESNs on some time series data prediction and industrial process modelling. However, there still are lots works need to be done in the future:

1. Further theoretical studies on the stability of echo state networks are needed. At the theoretical level, the pioneer of ESN, Jaeger (Jaeger, 2002b), gave a general sufficient condition for the stability of echo-state networks which is the spectral radius of reservoir is less than 1, but this condition is still not specific enough for the particular applications. In the study of theories and methods of network stability analysis, stability conditions for the design of network structures should be gave and improve the learning theory of echo-state networks.

2. Deep learning (networks with deep structure) is gradually becoming a hot research topic today and combining echo state networks with other deep learning models such as deep belief network, convolutional neural network to improve the network structure and learning algorithm of the echo state network and further enhance the performance of the network is required, so as to build a combined model with better prediction performance that better fits the actual process.

3. Echo state networks have been widely used in problems such as nonlinear time series prediction, speech recognition, and Echo state networks should be further applied in different fields such as industrial process control, fault detection and diagnosis, and environmental protection.

References

ALVAREZ, E., RIVEROL, C. & NAVAZA, J. 1999. Control of chemical processes using neural networks: implementation in a plant for xylose production. *ISA transactions,* 38**,** 375-382.

ALZABUT, J., TYAGI, S. & MARTHA, S. 2020. On the stability and Lyapunov direct method for fractional difference model of BAM neural networks. *Journal of Intelligent & Fuzzy Systems,* 38**,** 2491-2501.

ARDEN & HIKYU, L. 1982. A Regular Network for Multicomputer Systems. *IEEE Transactions on Computers,* C-31**,** 60-69.

ARENA, P., BAGLIO, S., CASTORINA, C., FORTUNA, L. & NUNNARI, G. A neural architecture to predict pollution in industrial areas. Proceedings of International Conference on Neural Networks (ICNN'96), 1996. IEEE, 2107-2112.

BABINEC, Š. & POSPíCHAL, J. Improving the prediction accuracy of echo state neural networks by anti-Oja's learning. International Conference on Artificial Neural Networks, 2007. Springer, 19-28.

BAHDANAU, D., CHO, K. & BENGIO, Y. 2015. Neural Machine Translation by Jointly Learning to Align and Translate.

BIANCHI, F. M., LIVI, L. & ALIPPI, C. 2018. Investigating Echo-State Networks Dynamics by Means of Recurrence Analysis. *IEEE Transactions on Neural Networks and Learning Systems,* 29**,** 427-439.

BOCCATO, L., LOPES, A., ATTUX, R. & VON ZUBEN, F. J. 2012. An extended echo state network using Volterra filtering and principal component analysis. *Neural Networks,* 32**,** 292-302.

CAO, J. & LI, X. 2005. Stability in delayed Cohen Grossberg neural networks: LMI optimization approach. *Physica D Nonlinear Phenomena,* 212**,** 54-65.

CHEN, J., ZHANG, H., HE, X., NIE, L., LIU, W. & CHUA, T.-S. 2017. Attentive Collaborative Filtering: Multimedia Recommendation with Item- and Component-Level Attention. *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval.* Shinjuku, Tokyo, Japan: Association for Computing Machinery.

CHEN, S., BILLINGS, S., COWAN, C. & GRANT, P. 1990. Non-linear systems identification using radial basis functions. *International Journal of Systems Science,* 21**,** 2513-2539.

CHEN, W.-B., MA, Q.-L. & PENG, H. Shared reservoir modular echo state networks for chaotic time series prediction. 2010 International Conference on Machine Learning and Cybernetics, 2010. IEEE, 2439-2443.

CHEN, X., WU, S. Z. & HONG, M. 2020. Understanding gradient clipping in private SGD: A geometric perspective. *Advances in Neural Information Processing Systems,* 33**,** 13773-13782.

CHESSARI, C., MCKAY, B., AGAMCNNONI, O., BARTON, G. & ROMAGNOLI, J. The Application of Neural Networks in the Development of an On-line Model for a Semi-Regenerative Catalytic Reformer. World Congress on Neural Networks, 1994. 173-178.

CHEW, J. Y., KURABAYASHI, D. & NAKAMURA, Y. 2015. Echo state networks with Tikhonov regularization: optimization using integral gain. *Advanced Robotics,* 29**,** 801-814.

CHOUIKHI, N., AMMAR, B., ROKBANI, N., ALIMI, A. M. & ABRAHAM, A. A Hybrid Approach Based on Particle Swarm Optimization for Echo State Network

Initialization. 2015 IEEE International Conference on Systems, Man, and Cybernetics, 9-12 Oct. 2015 2015. 2896-2901.
COATES, A., HUVAL, B., WANG, T., WU, D., CATANZARO, B. & ANDREW, N. Deep learning with COTS HPC systems. International conference on machine learning, 2013. PMLR, 1337-1345.
COUILLET, R., WAINRIB, G., SEVI, H. & ALI, H. T. Training performance of echo state neural networks. 2016 IEEE Statistical Signal Processing Workshop (SSP), 2016. IEEE, 1-4.
CRISPIN, A., CLAY, P., TAYLOR, G., BAYES, T. & REEDMAN, D. 2005. Genetic Algorithm Coding Methods for Leather Nesting. *Applied Intelligence,* 23**,** 9-20.
CUI, H., LIU, X. & LI, L. 2012. The architecture of dynamic reservoir in the echo state network. *Chaos: An Interdisciplinary Journal of Nonlinear Science,* 22**,** 033127.
CURTEANU, S. 2004. Direct and inverse neural network modeling in free radical polymerization. *Central European Journal of Chemistry,* 2**,** 113-140.
CYBENKO, G. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems,* 2**,** 303-314.
DENG, Z. & ZHANG, Y. 2007. Collective behavior of a small-world recurrent neural system with scale-free distribution. *IEEE Transactions on neural networks,* 18**,** 1364-1375.
DEVERT, A., BREDECHE, N. & SCHOENAUER, M. Unsupervised learning of echo state networks: A case study in artificial embryogeny. International conference on artificial evolution (evolution artificielle), 2007. Springer, 278-290.
DORIGO, M., BIRATTARI, M. & STUTZLE, T. 2006. Ant colony optimization. *IEEE computational intelligence magazine,* 1**,** 28-39.
DOYA, K. 1995. Supervised learning in recurrent networks. *Handbook of brain theory and neural networks***,** 796-800.
DUTOIT, X., SCHRAUWEN, B., VAN CAMPENHOUT, J., STROOBANDT, D., VAN BRUSSEL, H. & NUTTIN, M. 2009. Pruning and regularization in reservoir computing. *Neurocomputing,* 72**,** 1534-1546.
ELMAN, J. L. 1990. Finding structure in time. *Cognitive science,* 14**,** 179-211.
ERDöS, P. & RéNYI, A. 2011. On the evolution of random graphs. *The Structure and Dynamics of Networks.* Princeton University Press.
FAN, H.-T., WANG, W. & JIN, Z. Performance optimization of echo state networks through principal neuron reinforcement. 2017 International Joint Conference on Neural Networks (IJCNN), 2017. IEEE, 1717-1723.
FERREIRA, A. A. & LUDERMIR, T. B. Evolutionary strategy for simultaneous optimization of parameters, topology and reservoir weights in Echo State Networks. The 2010 International Joint Conference on Neural Networks (IJCNN), 18-23 July 2010 2010. 1-7.
FERREIRA, A. A., LUDERMIR, T. B. & DE AQUINO, R. R. B. 2013. An approach to reservoir computing design and training. *Expert Systems with Applications,* 40**,** 4172-4182.
FORTI, M. & TESI, A. 1995. New conditions for global stability of neural networks with application to linear and quadratic programming problems. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications,* 42**,** 354-366.
FRITZSON, P. 2011. *Introduction to modeling and simulation of technical and physical systems with Modelica,* John Wiley & Sons.
GALLICCHIO, C. & MICHELI, A. 2013. Tree Echo State Networks. *Neurocomputing,* 101**,** 319–337.
GIROSI, F. & POGGIO, T. 1990. Networks and the best approximation property. *Biological cybernetics,* 63**,** 169-176.

GOLDRICK, S., ŞTEFAN, A., LOVETT, D., MONTAGUE, G. & LENNOX, B. 2015. The development of an industrial-scale fed-batch fermentation simulation. *Journal of biotechnology,* 193**,** 70-82.

HADAEGHI, F., HE, X. & JAEGER, H. 2017. *Unconventional Information Processing Systems, Novel Hardware: A Tour D'Horizon*, IRC-Library, Information Resource Center der Jacobs University Bremen.

HAJNAL, M. A. & LŐRINCZ, A. Critical echo state networks. International Conference on Artificial Neural Networks, 2006. Springer, 658-667.

HAN, M. & XU, M. 2017. Laplacian echo state network for multivariate time series prediction. *IEEE transactions on neural networks and learning systems,* 29**,** 238-244.

HAN, S. I. & LEE, J. M. 2013. Fuzzy echo state neural networks and funnel dynamic surface control for prescribed performance of a nonlinear dynamic system. *IEEE Transactions on Industrial Electronics,* 61**,** 1099-1112.

HAN, Y., JING, Y., LI, K. & DIMIROVSKI, G. M. 2019. Network traffic prediction using variational mode decomposition and multi-reservoirs echo state network. *IEEE Access,* 7**,** 138364-138377.

HANSEN, N. 2006. The CMA Evolution Strategy: A Comparing Review. *In:* LOZANO, J. A., LARRAñAGA, P., INZA, I. & BENGOETXEA, E. (eds.) *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms.* Berlin, Heidelberg: Springer Berlin Heidelberg.

HANSEN, N., MüLLER, S. D. & KOUMOUTSAKOS, P. 2003. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation,* 11**,** 1-18.

HANSEN, N. & OSTERMEIER, A. 2001. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation,* 9**,** 159-195.

HASLER, J. & MARR, H. B. 2013. Finding a roadmap to achieve large neuromorphic hardware systems. *Frontiers in neuroscience,* 7**,** 118.

HAYKIN, S. 2010. *Neural networks and learning machines, 3/E,* Pearson Education India.

HERMANS, M. & SCHRAUWEN, B. 2012. Recurrent Kernel Machines: Computing with Infinite Echo State Networks. *Neural Computation,* 24**,** 104-133.

HOCHREITER, S. & SCHMIDHUBER, J. 1997. Long short-term memory. *Neural computation,* 9**,** 1735-1780.

HOFFMAN, J. D. & FRANKEL, S. 2018. *Numerical methods for engineers and scientists,* CRC press.

HOFMANN, T., SCHöLKOPF, B. & SMOLA, A. J. 2008. Kernel methods in machine learning. *The annals of statistics,* 36**,** 1171-1220.

HOLLAND, J. H. 1992. Genetic algorithms. *Scientific american,* 267**,** 66-73.

HOLZMANN, G. & HAUSER, H. 2010. Echo state networks with filter neurons and a delay&sum readout. *Neural Networks,* 23**,** 244-256.

HOPFIELD, J. J. 2007. Hopfield network. *Scholarpedia,* 2**,** 1977.

JAEGER, H. 2001. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report,* 148**,** 13.

JAEGER, H. 2002a. Adaptive nonlinear system identification with echo state networks. *Advances in neural information processing systems,* 15.

JAEGER, H. 2002b. *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach*, GMD-Forschungszentrum Informationstechnik Bonn.

JAEGER, H. & HAAS, H. 2004. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science*.

JAEGER, H., LUKOŠEVIČIUS, M., POPOVICI, D. & SIEWERT, U. 2007a. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural networks,* 20**,** 335-352.

JAEGER, H., MAASS, W. & PRINCIPE, J. 2007b. Special issue on echo state networks and liquid state machines. *Neural Networks,* 20**,** 287-289.

JIANG, F., BERRY, H. & SCHOENAUER, M. 2008. *Unsupervised Learning of Echo State Networks: Balancing the Double Pole*.

KAISER, M. 2011. A tutorial in connectome analysis: topological and spatial features of brain networks. *Neuroimage,* 57**,** 892-907.

KäMPF, J. H. & ROBINSON, D. 2009. A hybrid CMA-ES and HDE optimisation algorithm with application to solar energy potential. *Applied Soft Computing,* 9**,** 738-745.

KATSUKI, F. & CONSTANTINIDIS, C. 2014. Bottom-up and top-down attention: different processes and overlapping neural systems. *Neuroscientist,* 20**,** 509-21.

KAWAI, Y., PARK, J. & ASADA, M. 2019. A small-world topology enhances the echo state property and signal propagation in reservoir computing. *Neural Networks,* 112**,** 15-23.

KAWAI, Y., TOKUNO, T., PARK, J. & ASADA, M. Echo in a small-world reservoir: Time-series prediction using an economical recurrent neural network. 2017 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob), 2017. IEEE, 126-131.

KENNEDY, J. & EBERHART, R. Particle swarm optimization. Proceedings of ICNN'95-international conference on neural networks, 1995. IEEE, 1942-1948.

KOUNTOURIOTIS, P., OBRADOVIC, D., GOH, S. L. & MANDIC, D. P. Multi-step forecasting using echo state networks. EUROCON 2005-The International Conference on" Computer as a Tool", 2005. IEEE, 1574-1577.

KOZUB, D. J. & MACGREGOR, J. F. 1992. Feedback control of polymer quality in semi-batch copolymerization reactors. *Chemical Engineering Science,* 47**,** 929-942.

LI, D., HAN, M. & WANG, J. 2012. Chaotic time series prediction based on a novel robust echo state network. *IEEE Transactions on Neural Networks and Learning Systems,* 23**,** 787-799.

LI, Y. & WU, H. 2010. Global stability analysis in Cohen–Grossberg neural networks with delays and inverse Hölder neuron activation functions. *Information Sciences,* 180**,** 4022-4030.

LIU, K. & ZHANG, J. 2020. Nonlinear process modelling using echo state networks optimised by covariance matrix adaption evolutionary strategy. *Computers & Chemical Engineering,* 135**,** 106730.

LøKSE, S., BIANCHI, F. M. & JENSSEN, R. 2017. Training Echo State Networks with Regularization Through Dimensionality Reduction. *Cognitive Computation,* 9**,** 364-378.

LUKOŠEVIČIUS, M. & JAEGER, H. 2009. Reservoir computing approaches to recurrent neural network training. *Computer Science Review,* 3**,** 127-149.

LUKOŠEVICIUS, M., POPOVICI, D., JAEGER, H., SIEWERT, U. & PARK, R. 2006. Time warping invariant echo state networks. *International University Bremen, Technical Report*.

LUKOŠEVIČIUS, M. & USELIS, A. 2019. *Efficient Cross-Validation of Echo State Networks*.

LUONG, M.-T., PHAM, H. & MANNING, C. 2015. Effective Approaches to Attention-based Neural Machine Translation.

MA, Q.-L. & CHEN, W.-B. 2013. Modular state space of echo state network. *Neurocomputing,* 122**,** 406-417.

MA, Q., CHEN, W., WEI, J. & YU, Z. 2014. Direct model of memory properties and the linear reservoir topologies in echo state networks. *Applied Soft Computing,* 22**,** 622-628.

MAASS, W., NATSCHLäGER, T. & MARKRAM, H. 2002. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation,* 14**,** 2531-2560.

MACKEY, M. C. & GLASS, L. 1977. Oscillation and chaos in physiological control systems. *Science,* 197**,** 287-289.

MAIDHOF, C. & KOELSCH, S. 2011. Effects of selective attention on syntax processing in music and language. *J Cogn Neurosci,* 23**,** 2252-67.

MARUF, S., MARTINS, A. & HAFFARI, G. 2019. *Selective Attention for Context-aware Neural Machine Translation*.

MCCULLOCH, W. S. & PITTS, W. 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics,* 5**,** 115-133.

MINSKY, M. 1974. A framework for representing knowledge. MIT, Cambridge.

MISRA, J. & SAHA, I. 2010. Artificial neural networks in hardware: A survey of two decades of progress. *Neurocomputing,* 74**,** 239-255.

MNIH, V., HEESS, N., GRAVES, A. & KAVUKCUOGLU, K. 2014. Recurrent models of visual attention. *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2.* Montreal, Canada: MIT Press.

MORRIS, A., MONTAGUE, G. & WILLIS, M. 1994. Artificial Neural Networks-Studies in-Process Modeling and Control. *Chemical engineering research & design,* 72**,** 3-19.

NAYHOUSE, M., TRAN, A., KWON, J. S.-I., CROSE, M., ORKOULAS, G. & CHRISTOFIDES, P. D. 2015. Modeling and control of ibuprofen crystal growth and size distribution. *Chemical Engineering Science,* 134**,** 414-422.

NGUYEN, H. M., KALRA, G., JUN, T. J. & KIM, D. 2018. A Novel Echo State Network Model Using Bayesian Ridge Regression and Independent Component Analysis: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part II.

OTTE, S., BUTZ, M. V., KORYAKIN, D., BECKER, F., LIWICKI, M. & ZELL, A. 2016. Optimizing recurrent reservoirs with neuro-evolution. *Neurocomputing,* 192**,** 128-138.

OZTURK, M. C. & PRINCIPE, J. C. 2007. An associative memory readout for ESNs with applications to dynamical pattern recognition. *Neural Networks,* 20**,** 377-390.

OZTURK, M. C., XU, D. & PRINCIPE, J. C. 2007. Analysis and design of echo state networks. *Neural computation,* 19**,** 111-138.

PARK, J. & SANDBERG, I. W. 1991. Universal approximation using radial-basis-function networks. *Neural computation,* 3**,** 246-257.

PARK, J. H., PARK, C. H., KWON, O. M. & LEE, S. M. 2008. A new stability criterion for bidirectional associative memory neural networks of neutral-type. *Applied Mathematics and Computation,* 199**,** 716-722.

PARK, S. & FRED RAMIREZ, W. 1988. Optimal production of secreted protein in fed-batch reactors. *AIChE Journal,* 34**,** 1550-1558.

PASCANU, R., MIKOLOV, T. & BENGIO, Y. On the difficulty of training recurrent neural networks. International conference on machine learning, 2013. PMLR, 1310-1318.

POLAT, A. Hidden Bank Charges Amidst Ethics, Transparency, and Regulation: Case of Rebate Programs of International Banks. ICPESS (International Congress on Politic, Economic and Social Studies), 2017.

QIAO, J., LI, F., HAN, H. & LI, W. 2017. Growing Echo-State Network With Multiple Subreservoirs. *IEEE Transactions on Neural Networks and Learning Systems,* 28**,** 391-404.

RODAN, A. & TINO, P. 2011. Minimum Complexity Echo State Network. *IEEE Transactions on Neural Networks,* 22**,** 131-144.

RODRIGUES, A. E. & MINCEVA, M. 2005. Modelling and simulation in chemical engineering: Tools for process innovation. *Computers & Chemical Engineering,* 29**,** 1167-1183.

ROESCHIES, B. & IGEL, C. 2010. Structure optimization of reservoir networks. *Logic Journal of the IGPL,* 18**,** 635-669.

RóJ, E. & WILK, M. 1998. Simulation of an absorption column performance using feed-forward neural networks in nitric acid production. *Computers & chemical engineering,* 22**,** S909-S912.

ROSENBLATT, F. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review,* 65**,** 386.

RUMELHART, D. E., HINTON, G. E. & WILLIAMS, R. J. 1986. Learning representations by back-propagating errors. *nature,* 323**,** 533-536.

SASTRY, K., GOLDBERG, D. & KENDALL, G. 2005. Genetic Algorithms. *In:* BURKE, E. K. & KENDALL, G. (eds.) *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques.* Boston, MA: Springer US.

SCHMIDHUBER, J., WIERSTRA, D., GAGLIOLO, M. & GOMEZ, F. 2007. Training Recurrent Networks by Evolino. *Neural Comput.,* 19**,** 757–779.

SCHRAUWEN, B., DEFOUR, J., VERSTRAETEN, D. & CAMPENHOUT, J. V. The introduction of time-scales in reservoir computing, applied to isolated digits recognition. International Conference on Artificial Neural Networks, 2007. Springer, 471-479.

SCHRAUWEN, B., WARDERMANN, M., VERSTRAETEN, D., STEIL, J. J. & STROOBANDT, D. 2008. Improving reservoirs using intrinsic plasticity. *Neurocomputing,* 71**,** 1159-1171.

SHUTIN, D., ZECHNER, C., KULKARNI, S. R. & POOR, H. V. 2012. Regularized variational bayesian learning of echo state networks with delay&sum readout. *Neural Computation,* 24**,** 967-995.

SKOWRONSKI, M. D. & HARRIS, J. G. Minimum mean squared error time series classification using an echo state network prediction model. 2006 IEEE International Symposium on Circuits and Systems, 2006. IEEE, 4 pp.-3156.

SLATER, M., ANTLEY, A., DAVISON, A., SWAPP, D., GUGER, C., BARKER, C., PISTRANG, N. & SANCHEZ-VIVES, M. V. 2006. A Virtual Reprise of the Stanley Milgram Obedience Experiments. *PLOS ONE,* 1**,** e39.

SONG, Q. & CAO, J. 2007. Impulsive Effects on Stability of Fuzzy Cohen–Grossberg Neural Networks With Time-Varying Delays. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* 37**,** 733-741.

SONG, Y., LI, Y., WANG, Q. & LI, C. Multi-steps prediction of chaotic time series based on echo state network. 2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010. IEEE, 669-672.

SPORNS, O., HONEY, C. J. & KöTTER, R. 2007. Identification and Classification of Hubs in Brain Networks. *PLOS ONE,* 2**,** e1049.

SPORNS, O., TONONI, G. & KöTTER, R. 2005. The Human Connectome: A Structural Description of the Human Brain. *PLOS Computational Biology,* 1**,** e42.

SRINIVAS, M. & PATNAIK, L. M. 1994. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics,* 24**,** 656-667.

STEIL, J. J. Backpropagation-decorrelation: online recurrent learning with O (N) complexity. 2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541), 2004. IEEE, 843-848.
STRAUß, T., WUSTLICH, W. & LABAHN, R. 2012. Design Strategies for Weight Matrices of Echo State Networks. *Neural computation,* 24.
SU, H. T., MCAVOY, T. J. & WERBOS, P. 1992. Long-term predictions of chemical processes using recurrent neural networks: A parallel training approach. *Industrial & engineering chemistry research,* 31**,** 1338-1352.
TIAN, Y., ZHANG, J. & MORRIS, J. 2002. Optimal control of a fed-batch bioreactor based upon an augmented recurrent neural network model. *Neurocomputing,* 48**,** 919-936.
VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A., KAISER, L. & POLOSUKHIN, I. 2017. Attention Is All You Need.
VENAYAGAMOORTHY, G. K. & SHISHIR, B. 2009. Effects of spectral radius and settling time in the performance of echo state networks. *Neural Networks,* 22**,** 861-863.
VERSTRAETEN, D., DAMBRE, J., DUTOIT, X. & SCHRAUWEN, B. Memory versus non-linearity in reservoirs. The 2010 international joint conference on neural networks (IJCNN), 2010. IEEE, 1-8.
VERSTRAETEN, D., SCHRAUWEN, B., D'HAENE, M. & STROOBANDT, D. 2007. An experimental unification of reservoir computing methods. *Neural Networks,* 20**,** 391-403.
WANG, H. & YAN, X. 2014. Reservoir computing with sensitivity analysis input scaling regulation and redundant unit pruning for modeling fed-batch bioprocesses. *Industrial & Engineering Chemistry Research,* 53**,** 6789-6797.
WANG, H. & YAN, X. 2015. Optimizing the echo state network with a binary particle swarm optimization algorithm. *Knowledge-Based Systems,* 86**,** 182-193.
WANG, L. & ZOU, X. 2002. Zou, X.F.: Exponential Stability of Cohen-Grossberg Neural Networks. Neural Networks 15, 415-422. *Neural networks : the official journal of the International Neural Network Society,* 15**,** 415-22.
WANG, X., JIN, Y. & HAO, K. 2019. Evolving local plasticity rules for synergistic learning in echo state networks. *IEEE Transactions on Neural Networks and Learning Systems,* 31**,** 1363-1374.
WATTS, D. J. & STROGATZ, S. H. 1998. Collective dynamics of 'small-world' networks. *Nature,* 393**,** 440-442.
WERBOS, P. J. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE,* 78**,** 1550-1560.
WILLIAMS, R. J. & ZIPSER, D. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation,* 1**,** 270-280.
WYFFELS, F., SCHRAUWEN, B., VERSTRAETEN, D. & STROOBANDT, D. Band-pass reservoir computing. 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), 2008. IEEE, 3204-3209.
XU, M., HAN, M., QIU, T. & LIN, H. 2018. Hybrid regularized echo state network for multivariate chaotic time series prediction. *IEEE transactions on cybernetics,* 49**,** 2305-2315.
XUE, Y., YANG, L. & HAYKIN, S. 2007. Decoupled echo state networks with lateral inhibition. *Neural Networks,* 20**,** 365-376.
YANG, C., QIAO, J., AHMAD, Z., NIE, K. & WANG, L. 2019. Online sequential echo state network with sparse RLS algorithm for time series prediction. *Neural Networks,* 118**,** 32-42.
YANG, C., ZHU, X., AHMAD, Z., WANG, L. & QIAO, J. 2018. Design of Incremental Echo State Network Using Leave-One-Out Cross-Validation. *IEEE Access,* PP**,** 1-1.

YANG, X.-S. & DEB, S. Cuckoo search via Lévy flights. 2009 World congress on nature & biologically inspired computing (NaBIC), 2009. Ieee, 210-214.
YAO, X. & WANG, Z. 2019. Broad echo state network for multivariate time series prediction. *Journal of the Franklin Institute,* 356**,** 4888-4906.
ZEKI, A., ABUBAKAR, A. & CHIROMA, H. 2016. An intermediate significant bit (ISB) watermarking technique using neural networks. *SpringerPlus,* 5**,** 1-25.
ZHANG, J., HE, T., SRA, S. & JADBABAIE, A. 2019. Why gradient clipping accelerates training: A theoretical justification for adaptivity. *arXiv preprint arXiv:1905.11881*.
ZHANG, J., MORRIS, A. & MARTIN, E. 1998. Long-term prediction models based on mixed order locally recurrent neural networks. *Computers & chemical engineering,* 22**,** 1051-1063.
ZHANG, J. & MORRIS, A. J. 2000. Long range predictive control of nonlinear processes based on recurrent neuro-fuzzy network models. *Neural computing & applications,* 9**,** 50-59.