On the Induction of Temporal Structure by Recurrent Neural Networks

MAHMUD SAAD SHERTIL

A thesis submitted in partial fulfilment of the requirements of Nottingham Trent University for the degree of Doctor of Philosophy

November 2014

Acknowledgment

I would like to present my full gratitude to my director of studies, Dr. Heather Powell, for her support, guidance and invaluable advice to achieve the requirements of the thesis. She significantly dedicated and provided necessary critique regarding the research.

I would like also to extend my sincere thanks to my second supervisor, Dr. Jonathan Tepper, I could not have imagined having a better advisor and mentor for my PhD, without his common sense, knowledge, perceptiveness and assistances and his friendly support in many issues during the research.

I would like to dedicate this thesis to the soul of my mother, who I have never seen her and the soul of my father. In addition, many thanks to the beloved family who have taken care of me after I lost my mother.

Heartfelt thanks and deep respect for my wife, who stood by me in both my sad and happy moments. Also, thanks for her understanding and endless love, throughout my studies. Even when I felt panicked or disturbed, she encouraged me and said: Go on! You have to continue in the doctoral task, which is similar to a marathon runner. My kind thanks also, go to my lovely children: "Salam, Salsabeel, Samaher, Mohammed", seeing them, I absorb the power to achieve the best position to make them proud as a father.

I would also like to acknowledge my brothers, sisters and friends for their constant encouragement for everything. I acknowledge all those who have prayed for me, guided me with wisdom, helped me with their kindness, and tolerated me out of their love.

Last but not least, my real thanks and appreciations goes to my colleagues and friends in my native country, Libya and the PhD students in the UK, for their help and wishes for the successful completion of this research.

Abstract

Language acquisition is one of the core problems in artificial intelligence (AI) and it is generally accepted that any successful AI account of the mind will stand or fall depending on its ability to model human language. Simple Recurrent Networks (SRNs) are a class of so-called artificial neural networks that have a long history in language modelling via learning to predict the next word in a sentence. However, SRNs have also been shown to suffer from catastrophic forgetting, lack of syntactic systematicity and an inability to represent more than three levels of centre-embedding, due to the so-called 'vanishing gradients' problem. This problem is caused by the decay of past input information encoded within the error-gradients which vanish exponentially as additional input information is encountered and passed through the recurrent connections. That said, a number of architectural variations have been applied which may compensate for this issue, such as the Nonlinear Autoregressive Network with exogenous inputs (NARX) network and the multi-recurrent network (MRN). In addition to this, Echo State Networks (ESNs) are a relatively new class of recurrent neural network that do not suffer from the vanishing gradients problem and have been shown to exhibit state-of-the-art performance in tasks such as motor control, dynamic time series prediction, and more recently language processing.

This research re-explores the class of SRNs and evaluates them against the state-of-the-art ESN to identify which model class is best able to induce the underlying finite-state automaton of the target grammar implicitly through the next word prediction task. In order to meet its aim, the research analyses the internal representations formed by each of the different models and explores the conditions under which they are able to carry information about long term sequential dependencies beyond what is found in the training data.

The findings of the research are significant. It reveals that the traditional class of SRNs, trained with backpropagation through time, are superior to ESNs for the grammar prediction task. More specifically, the MRN, with its state-based memory of varying rigidity, is more able to learn the underlying grammar than any other model. An analysis of the MRN's internal state reveals that this is due to its ability to maintain a constant variance within its state-based representation of the embedded aspects (or finite state machines) of the target grammar. The investigations show that in order to successfully induce complex context free grammars directly from sentence examples, then not only are a hidden layer and output layer recurrency required, but so is self-recurrency on the context layer to enable varying degrees of current and past state information, that are integrated over time.

Contents

Acknowledgment ii

Abstract iii

Contents v

List of Figures x

List of Tables xiii

Acronyms xvii

Chapter 1 1

1. Introduction 1

1.1 Summary 1

1.2 Problem Statement 2

1.3 Scope of Research 4

1.4 Thesis Outline 6

Chapter 2 8

2. Literature Study 8

2.1 The Nature and Complexity of Language 8

2.1.1 Nativist vs. Empiricist Perspectives 8

2.1.2 Language Complexity and Computation 11

2.2 Connectionist and Statistical Models of Language Acquisition 15

2.2.1 Supervised Connectionist Learning Algorithms 18

2.2.2 Supervised Connectionist Models of Language Acquisition 20

2.3 Limitations of Connectionism 22

2.3.1 Argument against Biological Plausibility 24

2.3.2 Argument against Connectionism for Developmental Cognitive Modelling 25

2.3.3 Learning Deterministic Representations Using a Continuous State Space 26

2.4 Discussion and Conclusion 28

Chapter 3 30
3. Neural Network Architectures 30
3.1. Recurrent Neural Networks 30
3.1.1 Jordan Network 31
3.1.2 Time Delay Neural Recurrent Network (TDNN) 32
3.1.3 Nonlinear Autoregressive Network with Exogenous Input (NARX) 33
3.1.4 Simple Recurrent Networks (SRN) 35
3.1.5 Multi Recurrent Networks (MRN) 37
3.1.6 Long Short Term Memory (LSTM) 39
3.1.7 Echo State Networks (ESNs) 40
3.2. Summary 43
Chapter 4 44
4. Data and Methodology 44
4.1. The Reber Grammar Datasets 44
4.1.1 Symbol Representations 45
4.1.2 The Regular Grammar: Simple Reber Grammar 47
4.1.2.1 Reber Grammar Dataset 48
4.1.3. The Context-Free Grammar: Embedded Reber Grammar (ERG) 49
4.1.3. 1 Symmetrical Sequences 52
4.1.3. 2 Asymmetrical Sequences 53
4.2. Model Fitting 56
4.2.1. Overview 57
4.2.2. Model Selection 58
4.2.2.1 Backpropagation through Time (BPTT) 60
4.2.2.2 Fixed Learning Rate 62
4.2.2.3 Pattern Error- Sensitive Learning Rate 62
4.2.2.4 Taguchi Method and Analysis of Variance (ANOVA) 63

4.2.3. Model Evaluation 65

Chapter 5 67

5. Experimental Results 67

5.1 Learning the Regular Grammar 67

5.2 Learning the Context Free Grammar 71

5.2.1 SRN Using Constant Learning Rate 71

5.2.1.1 Embedded Reber Grammar (Symmetrical Sequences) 72

5.2.1.2 Embedded Reber Grammar (Asymmetrical Sequences) 77

5.2.2 SRN Using Pattern Error-Sensitive 79

5.2.2.1 SRN Results for Asymmetrical Training Tested with Symmetrical and Asymmetrical Sequences 79

5.2.2.2 SRN Results for Symmetrical Training, Tested with Symmetrical and Asymmetrical Sequences 80

5.2.3 Jordan Network 81

5.2.2.3 Summary 84

5.2.4 TDNN 85

5.2.4.1 TDNN results for asymmetrical training tested with symmetrical and asymmetrical sequences 85

5.2.4.2 TDNN results for symmetrical training, tested with symmetrical and asymmetrical sequences 86

5.2.5 NARX 88

5.2.5.1 NARX results for asymmetrical training, tested with symmetrical and asymmetrical sequences 88

5.2.5.2 NARX results for symmetrical training tested with symmetrical and asymmetrical sequences 89

5.2.6 MRNs 90

5.2.5.1 MRN with Noise Injection 95

5.2.7 Conclusion 98

5.2.8 ESNs 99

5.2.6.1 Standard ESN (with Feedback from Output to Reservoir) 99

5.2.6.2 ESN with Jumping Connections 100

5.2.6.3 The performance of the ESN 105

5.2.6.3 The Limitation of ESN 105

5.3 Summary of Results and Discussion 106

Chapter 6 114

6. Understanding the Internal Representations Formed. 114

6.1 Visualisation of the Internal Representations Formed 114

6.1.1 Internal Representations of the SRN 117

6.1.2 Internal Representations of the NARX 124

6.1.3 Internal Representations of the MRN 129

6.1.4 Internal Representations of the ESN 135

6.2 Comparative Analysis of the Internal Representations 140

6.3 Summary of Results and Discussion 145

Chapter 7 147

7. Conclusion & Future Work 147

7.1 Introduction 147

7.2 Training Parameters 149

7.3 Main Conclusion Derived from This Study 149

7.4 Future Research Directions. 152

References 154

Appendixes 175

Appendix A: Some Results of Medium Acceptance Criterion Using SRN 175

Appendix B: Results of Different Networks and Info of Some Dataset 177

Appendix C: Plots of Different PCA Components Using SRN 180

Appendix D: Test Datasets Results of Some of RNNs 182

Appendix E: The Evaluation of Symmetrical Training Dataset 186

List of Figures

Figure 2.1 Block diagram that displays the form of supervised learning 18
Figure 3.1 The Jordan Network (Jordan 1986) 31
Figure 3.2 Shifting Recurrent Network 33
Figure 3.3 NARX network with d_x delayed inputs and d_y delayed outputs (Diaconescu 2008).. 34
Figure 3.4 Simple Recurrent Network 36
Figure 3.5 Architecture of Multi-Recurrent Network 38
Figure 3.6 A LSTM memory block with one cell and its gate units 40
Figure 3.7 ESN Architectures: Solid arrows indicate fixed connections and dashed arrows indicate trained connection. 41
Figure 4.1 The finite-state grammar (FSG) used by Reber 48
Figure 4.2 A complex finite-state grammar involving embedded sequences. 51
Figure 4.3 Embedded Reber grammar biased symbols (Asymmetrical). The numbers above each arc indicate the transition probabilities in the biased form of the grammar 54
Figure 4.4 SRN with noise-injection units 59
Figure 4.5 MRN with noise-injection units 59
Figure 4.6 Schema of the basic idea of Backpropagation Through Time (Jaeger 2002) 61
Figure 6.1 shows the states of the embedded Reber Grammar 116
Figure 6.2 Plots of the two most significant principle components of the hidden layer activations of a asymmetrically trained SRN, presented with three pairs of symbol sequences (in a, b and c respectively) from the ERG. Each pair has the same embedded sequence but different initial symbol so that one is in the lower half (dashed lines) and the other is in the upper half (solid blue lines). The sequences are in the table 6.7: (a) 1, 2; (b) 5, 6; (c) 7, 8 and with respect to principal components PC1 and PC2. (a BPTXSP/BTTXST, b) BPPVVP/BTPVVT, c) BTTSSXST/BPTSSXSP 120
Figure 6.3 SRN: Centroid of each state using asymmetrical sequences (correctly predicted sequences) 121
Figure 6.4 Plots of the two most significant principle components of the hidden layer activations of an asymmetrically trained by SRN, trajectories of two non-identical asymmetrical sequences that were incorrectly predicted by the SRN. (a) The lower

embedded part BPPTVPSP. (b) The upper embedded part BTPTTVVT. With respect to principal components PC1 and PC2. 122
Figure 6.5 SRN: located ranges of state seven for correctly predicted and incorrectly predicted asymmetrical sequences. 124
Figure 6.6 Plots of the two most significant principle components of the hidden layer activations of an asymmetrically trained NARX, (a, b) are the trajectories of the sequence numbered 1,2,3,4 a) BTTXST/BPTXSP and b) BTPVVT/BPPVVP in Table 6.7 which have a length of six symbols. 126
Figure 6.7 Plots of the two most significant principle components of the hidden layer activations of an asymmetrically trained NARX, (a, b) trajectories of two non-identical sequences that are incorrectly predicted a) BPPTTVPXTTVPSP and b) BTPVPXVPXTTVVT 127
Figure 6.8 Plots of the two most significant principle components of the hidden layer activations of an asymmetrically trained NARX. a) Trajectories of 16-length sequences that are incorrectly predicted "BPTXXTTVPXTTVPSP and b) is the correctly predicted sequence of the same length BPPVPXTVPXTTTVVP. 128
Figure 6.9 NARX: Range of state seven for both correctly predicted sequences and incorrectly predicted sequences (failes on the penultimate symbol) 129
Figure 6.10 Plots of the two most significant principle components of the hidden layer activations of an asymmetrically trained by MRN (a) trajectories of the six length sequences BTTXST/BPTXSP (b) BTTXXVVT/BPTXXVVP and (c) BTTSSXST/BPTSSXSP are trajectories of eight length sequences upper and lower embedded. 132
Figure 6.11 Plots of the two most significant principle components of the hidden layer activations of an asymmetrically trained MRN (a) trajectories of 15-length sequence BPTSSSXXTVPXVVP, (b) BPTSSSXXTTTVVP 14-length sequence that is incorrectly predicted using MRN. 133
Figure 6.12 Range of state seven for both correctly predicted and incorrectly predicted penultimate symbols 134
Figure 6.13 Range of sate four, (*) lower embedded sequence correctly predicted, (+) lower embedded sequence unpredicted sequences. 135
Figure 6.14 Plots of the two most significant principle components of the hidden layer activations of an asymmetrically trained ESN. (a) BTTXST/BPTXSP (b)

BTPVVT/BPPVVP (c) BPPVPXTVPXTTTVVP. The trajectories are of sequences that were correctly predicted by the network. 137
Figure 6.15 Plots of the two most significant principle components of the hidden layer activations of an asymmetrically trained ESN. (a) BTTXXVVT. (b) BPTSSXSP. The trajectories are of sequences that were incorrectly predicted by the network in the embedded part. 138
Figure 6.16 ESN: Range of state seven for both correctly and incorrectly predicted penultimate symbols... 139
Figure 6.18 The centroid of the upper and lower embedded grammar of the networks a)SRN, b) NARX, c) MRN, d) ESN 143
Figure 6.19 The Euclidean distances between each corresponding centroid representing the embedded grammar states of the upper and lower embedded grammars (for each model evaluated). 144

List of Tables

Table 2.1 Chomsky hierarchy of grammars 13

Table 4.1 Orthogonal binary vector representations for input patterns. 46

Table 4.2 Orthogonal vector representations for input patterns 47

Table 4.3 Characteristics of the 300000-Symmetrical-embedded-Reber 52

Table 4.4 Characteristics of the 1000 Symmetrical Testing Dataset (Embedded Reber Grammar) 53

Table 4.5 Characteristics of the 300000 Asymmetrical Dataset 55

Table 4.6 Characteristics of the 1000 Asymmetrical Testing Dataset 56

Table 5.1 Elman's SRN: The percentage of the accuracy of the sequences across the whole data set on the training performance for the Reber Grammar 70

Table 5.2 Elman’s SRN: Percentage accuracy of the entire dataset training for the embedded Reber grammar (soft acceptance criterion) 73

Table 5.3 SRN training results on ERG (soft acceptance criterion) 74

Table 5.4 SRN test results using soft acceptance criterion for the ERG 75

Table 5.5 SRN test results for ERG using hard acceptance criterion and binary symbol representations 76

Table 5.6 SRN test results for ERG using hard acceptance criteria and non-binary symbol representations 77

Table 5.7 SRN results of the symmetrical test file with 10 asymmetrical training networks and binary symbol representations using soft acceptance criterion 78

Table 5.8 SRN results of five asymmetrical training nets tested on asymmetrical sequences (pattern error-sensitive learning rate) 79

Table 5.9 SRN results of five asymmetrical training nets tested on symmetrical sequences (pattern error-sensitive learning rate) 80

Table 5.10 SRN results of five testing nets tested on asymmetrical sequences 81

Table 5.11 SRN results of five testing nets tested on symmetrical sequences 81

Table 5.12 Jordan test results for asymmetrical training and asymmetrical test (binary input representations) 82

Table 5.13 Jordan test results for asymmetrical training and symmetrical test (binary input representations) 82

Table 5.14 Jordan results of five testing nets tested in asymmetrical sequences (binary input representations) 83
Table 5.15 Jordan results of five testing nets tested in symmetrical sequences (binary input representations) 83
Table 5.16 Comparing results of SRN and Jordan network (best network performance for asymmetrical training and test sets) 84
Table 5.17 TDNN results of five nets trained with asymmetrical sequences and tested on asymmetrical sequences 85
Table 5.18 TDNN results of five nets trained with asymmetrical sequences and tested on symmetrical sequences 86
Table 5.19 TDNN results of five nets trained with symmetrical sequences and tested on asymmetrical sequences 86
Table 5.20 TDNN results of five nets trained with symmetrical sequences and tested on symmetrical sequences 87
Table 5.21 Part of a test file showing the prediction according to the sequence length 88
Table 5.22 NARX results of five nets trained with asymmetrical sequences and tested on asymmetrical sequences 88
Table 5.23 NARX results of five nets trained with asymmetrical sequences and tested on symmetrical sequences 89
Table 5.24 NARX results of five nets trained with symmetrical sequences and tested on asymmetrical sequences 89
Table 5.25 NARX results of five nets trained with symmetrical sequences and tested on symmetrical sequences 90
Table 5.26 The percentage of correct predictions for different training and test datasets in terms of bias and non-bias sequences for TDNN and NARX networks 90
Table 5.27 Results of different numbers of memory boxes trained with Asymmetrical and tested with Asymmetrical sequences in MRN 94
Table 5.28 Result of Testing the MRN (with 4 memory banks) for sequences longer than in the training set (5 tested at each length) 95
Table 5.29 MRN: The accuracy of asymmetrical training; tested with asymmetrical data (one noise node) 96
Table 5.30 MRN: The accuracy of asymmetrical training; tested with asymmetrical and symmetrical data (one noise unit – seven nodes) 97

Table 5.31 MRN: The accuracy of symmetrical training, tested with asymmetrical and symmetrical data (one noise unit) 98
Table 5.32 Testing different length of the datasets using MRN with unit noise injection 99
Table 5.33 Percentage of the ESN performance using different types of feedback .. 100
Table 5.34 Percentage of prediction accuracy using ESN with jumping connection without settling time 102
Table 5.35 Number of the performance of the ESN that applied on the Taguchi method 102
Table 5.36 the values of the parameters tried 103
Table 5.37 trained ESN with different size of reservoir 105
Table 5.38 Testing different length of the dataset tested on ESN, the minimal length for dataset 40 is 27 then 40 for the dataset 50 etc. 106
Table 5.39 Percentage of correct predictions by trained networks processing training and test datasets. W = whole sequence; E = embedded section; P = penultimate symbol. All network architectures used pattern error- sensitive learning type, binary input representations, learning rate 0.3 and 0.75 momentum. 108
Table 5.40 The best performance of asymmetrically trained networks tested with both dataset types 109
Table 5.41 The best performance of symmetrically trained networks tested with both dataset types 109
Table 5.42 The performance of the networks against the memory 111
Table 5.43 the performance of networks have same number of memory 112
Table 6.1 The models evaluated using asymmetrical sequences for training and testing 114
Table 6.2 Meaning of some terms 116
Table 6.3 Symmetrical and asymmetrical Sequences that have been selected 117
Table 6.4 Results of SRN using asymmetrical sequences and their prediction results 118
Table 6.5 SRN: Centroid of the states for asymmetrical sequences (correctly predicted) 119
Table 6.6 SRN: Range of the states for asymmetrical sequences (Incorrectly predicted sequences) 123

Table 6.7 Sequences results for NARX and the position of the incorrectly predicted symbol 125
Table 6.8 Asymmetrical sequences correctly predicted by MRN 130
Table 6.9 Sequences results for ESN and the position of the unpredicted symbol 136
Table 6.10 The prediction of the unique sequences in the test dataset 140
Table 6.11 Details of the number of each length sequence (unique) correctly predicted by each network 140
Table 6.12 The centroids for each grammar state for each network 141
Table 6.13 Euclidean distance between MRN and the networks 145

Acronyms

AG: Artificial Grammar

AI: Artificial Intelligence

ANN: Artificial Neural Networks

ANOVA: Analysis of Variance

Asym: Asymmetrical Sequences

BP: Backpropagation

BPTT: Backpropagation through Time

CDA: Canonical Discriminant Analysis

CFL: Context Free Language

CSG: Context-Sensitive Grammars

EM: Expectation-Maximisation

ERG: Embedded Reber Grammar

ESNs: Echo State Networks

ESP: Echo State Property

FF-MLP: Feed-Forward Multi-Layered Perceptron

FSM: Finite State Machine

HCA: Hierarchical Cluster Analysis.

HMM: Hidden Markov Models

IL: Implicit Learning

LMS: Last Mean Squared

LSTM: Long Short Term Memory

LTM: Long Term Memory

MDS: Multi-dimensional Scaling

MRN: Multi Recurrent Network

NARX: Nonlinear Autoregressive Network with exogenous

NLP: Natural Language Processing

PAU: Periodically Attentive Units

OAS: Orthogonal Array Selector

PCA: Principle Component Analysis

PDP: Parallel-Distributed Processing

RG: Regular Grammars

RNNs: Recurrent Neural Networks

SD: Standard Deviation

SRN: Simple Recurrent Network

ST: Settling Time

Sym: Symmetrical Sequences

TDNN: Time Delay Neural Network

Chapter 1
1. Introduction

1.1 Summary

The general idea of this project is to explore the capability and limitations of recurrent artificial neural network architectures for learning finite state grammar directly from string or sentence examples. A number of common recurrent neural network architectures will be investigated. The overarching aim is to compare the capabilities of the various architectures and understand their capacities and limitations. As part of this, more established architectures will be compared with a new class of networks, called Echo State Networks (ESNs) (Jaeger, 2001) which are also subsumed in literature under the general term: *reservoir computing*. It will be important to ascertain the characteristics for good architecture design and training regimes for such systems when attempting to model grammar or language induction.

More specifically a grammatical induction task is used to examine and evaluate; a) the capability of Simple Recurrent Networks (SRNs) (i.e. Elman network & Jordan network), b) Nonlinear Autoregressive model process with Exogenous Input (NARX), c) Multi-Recurrent Networks (MRN) and, d) the state-of-the-art Echo State Network (ESN). An investigation is carried out to explore how well these networks have learned the grammar directly from string examples (generated from the target grammar) and also examine the robustness of the learned representations and how they begin to fail as the complexity of the language increases.

The aim of the present study is to review, examine, and develop a way to explore which embedded memory configurations within recurrent neural network architectures, trained with gradient descent learning algorithm, provide the most effective propagation of gradient information for learning simple long-term dependency problems, such as those found in natural language contexts. The six RNN models used in this study have various types of recurrency, so they need to be evaluated individually and a comparison between their performances will be carried out. Recurrency of these networks includes output layer recurrency, as found in Jordan network (Jordan 1986); hidden layer recurrency, as in Elman (Elman 1990); output recurrency with time delay connection, as in time delay neural networks; input output delay connection, as found in non-linear autoregressive model process with exogenous input (NARX)(Gers 2001); and input and output recurrency, as in multi-layer neural networks. It should be pointed out here that these five networks share the same learning algorithm. However, the sixth network considered differs in terms of learning algorithm and recurrency connections; this is the echo state network (ESN). The task presented to these networks is to learn a finite state grammar and this work aims to investigate when these networks begin to fail and how failure occurs as the complexity of the language increases.

In order to accomplish the aim of this research, a number of different feedback connections, hidden units, and state memories were explored to optimise these models which share the same learning algorithm, and compare them against an ESN which does not share the gradient-descent learning limitation. The, the internal representation of these networks are then analysed and evaluated.

1.2 Problem Statement

Recent studies demonstrate that the ability to learn nonlinear temporal dynamic behaviour is a significant factor in the solution of many types of complex problem-solving, such as those found in the practical problem domains of natural language processing, speech processing, adaptive control, time series predication, financial modelling, DNA sequencing etc. (Koskela, Varsta et al. 1998). There are several statistical learning and machine learning techniques, methods and models, which can be applied in order to learn the underlying temporal structure and dynamics of a particular problem. Algorithms and techniques, including recurrent neural networks (RNNs) support vector machines, kernel regression, hidden Markov models, reinforcement learning and Bayesian networks. RNNs are a class of connectionist network whose computational neurons produce activations based on the activation history of the network (Kremer, Kolen 2001).

An RNN has a set of units, each taking a real value in each time step, and has a group of weighted connections joining units together. The input and output units are set according to the problem situation. While the input values are determined by the problem, the output units are computed using the connection weights and the hidden units. Activations from units within these input, hidden, and output layers, are typically fed back as input to the same or previous layers. This forms a complex memory-based system due to cycles in the flow of activation, with the output from one time step informing the input to the next. RNNs have nonlinear dynamics, allowing them to perform in a highly complex manner. In theory, the states of the hidden units can store data over time in the form of a distributed representation and this can be used many time-steps later to predict subsequent input vectors (Sutskever, Hinton 2010). The characteristic of an RNN can be distinguished from its feedforward counterparts by its ability to map sequences of input vectors distributed across time into output vector sequences. In this respect, RNNs can be viewed as vector-sequence transducers. The reason this makes RNNs interesting is that they can be applied to almost any problem with sequential structure, including problems that arise in many natural contexts such as in control, speech, and natural language processing.

RNNs are classified as a type of graphical model, which is an interaction between probability theory and graph theory that are a group of traditional statistical models. These RNNs have units and connections whose values are determined by statistical methods. Other examples of graphical models are Bayesian networks, Gaussian mixture models and Hidden Markov models (Murphy 2001). RNNs play an important role in applied mathematics and engineering, as they provide a set of probabilistic tools for dealing with two problems that occur naturally within these disciplines, namely, uncertainty and complexity.

It is stated (Bengio et al 1994, Gers et al 2003) that the RNNs which use back-propagation through time, are unable to hold long term dependency due to the vanishing time problem. Long term dependency is important because it provides a challenge to natural language processing. Thus, the research will investigate, develop, analyse and evaluate the networks mentioned previous and explore their limitations.

1.3 Scope of Research

Although there are numerous connectionist techniques for processing temporal information, the most widely used is the simplest RNN known as the Simple Recurrent Network (SRN) (Elman 1990). The SRN is a state-based model, similar in complexity to a Hidden Markov Model and represents sequences of information by internal states of neural activation (Cartling 2008). The SRN has proven remarkably useful for temporal problem domains such as natural language processing and in particular, regular languages. Much research has been conducted to illustrate temporal processing in SRNs (Gupta, McAvoy et al. 2000, Deliang, Xiaomei et al. 1996). However, the majority of these SRN-based studies involve processing noise-free binary temporal sequences with orthogonal components or fixed duration feature vectors having low dimensions (Gupta, McAvoy et al. 2000). Furthermore, SRNs have failed to adequately model the combinatorial systematicity of human language appropriately (Farkaš, Crocker 2008). Combinatorial systematicity refers to the ability of the human language faculty to use a relatively small lexicon and few syntactic rules to generate a very large, possibly infinite, number of sentences. It appears that such a faculty must have a neural basis and therefore, any artificial neural network attempting to model human cognition should be able to demonstrate systematicity, although this is still a matter of debate (Farkaš, Crocker 2008).

SRN-based approaches with their gradient descent learning are considered 'standard RNNs' and have been plagued by major practical difficulty (Gers 2001). The gradient of the total output error with respect to previous input quickly vanishes as the time lag between relevant inputs and errors increases (Bengio, Simard et al. 1994). Gers (2003) stated that this is why standard SRNs are unable to learn time lagged information or dependencies exceeding as few as 5-10 discrete time steps among relevant input events and target signals. Other architectures have been developed to attempt to overcome these issues. One is the Long Short Term Memory (LSTM) (Hochreiter, Schmidhuber 1997). LSTM is a gate-based RNN architecture that uses gradient descent-based learning to remember, establish and maintain temporal information over very long times periods (James, 2003). Unfortunately, due to its constant error flow through internal states of memory cells, it exhibits problems similar to those of feedforward networks (which present the entire input string at once).

Over the past five years, there has been interest in another alternative: Echo state networks (ESNs), a type of RNN (Jaeger 2002). ESNs are relatively simple RNNs with the hidden layer consisting of a large collection of processing units, randomly inter-connected, as well as normal connections applied to the input, hidden and output layers. All links have randomly assigned and fixed (untrainable) weights except for those coming from the hidden layer, which are trainable. The appeal of the ESN is the simplicity of its training process, which is reduced to a task of one-short simple linear regression (Jaeger 2002). ESNs have been applied with varying success to numerous problem domains such as behaviour classification (Noris, Nobile et al. 2008); natural language processing (Bickett et al. 2007); speech recognition (Skowronski, Harris 2006); financial forecasting (Lin, Yang et al. 2009); and symbol grounding in robots (Jaeger et al 2002).

Although state-of-the-art performance has been reported for the iterated prediction of noiseless time series data, the usefulness of this is questionable and studies with ESNs for realistic problem domains have revealed the difficulty of creating the reservoir of interconnections in a systematic way for a given problem. It can take the exploration of many reservoir configurations before a solution is found (Binner et al., 2010; Rodan & Tino, 2011). Clearly, there is scope for advancing knowledge concerning the strengths and weaknesses of ESNs for different types of problem and the need for a principled approach to ESN application, appropriate to the problem domain in order to increase their utility.

The major contribution of this research is that it is the first to reveal that sluggish state based representations formed by recurrent memory layers with self-recurrent links help to solve the grammar induction task in a way that is superior to the other architectures examined in this research including the state-of-the-art ESN. A well-accepted approach of clustering hidden unit activation profiles called Principle Component Analysis (PCA) was used to identify why the MRN performs better than the other models assessed. The internal representation formed by the weights of an RNN after training can be revealed by analysing the resulting hidden neuron activations and how they vary with respect to each new input within a sequence (Cartling, 2007). The reason for such performance is not attributed exclusively to the hidden and output recurrency but also to the self-recurrency of units in the context layer and replicating the context layer itself to form memory banks. This collectively implements a complex state space of varying rigidity (also known as sluggish state space). These sluggish state spaces enable the network to generate stable representations of the underlying grammar by maintaining a constant distance between clusters of activation space (representing individual grammatical states) which the other models failed to perform.

1.4 Thesis Outline

This thesis has seven chapters. Since chapter one has already been introduced, this section will provide a content summary for the remaining six chapters:

Chapter 2: Literature Study

This chapter provides a review of the relevant literature in the field of natural language sentences and connectionist models. The main area in the natural language discussed in this chapter is the complexity of language. In particular, this chapter focuses on a Chomsky hierarchy of grammars. In addition, the rest part of this chapter reviews supervised connectionist models of language acquisition and the limitations of connectionism for such tasks.

Chapter 3: Neural Network Approaches

This chapter describes the connectionist methods used in conducting this research. The chapter focuses on the traditional techniques used for language modelling tasks, in particular, for the next-symbol prediction task. The Jordan network, SRNs, MRN and ESN are considered.

Chapter 4: Data and Methodology

This chapter presents the language data sets used in the current study and discusses the model fitting and model selection approaches applied to optimise the different neural networks for language task. This chapter concluded with a summary of connectionist networks that appear to be able to discover the underlying grammar and thus finite-state automata.

Chapter 5: Experimental Results

In this chapter, evaluation and contrast the present work by applying the same experimental framework to run a series of experiments with the different RNNs under review in the work done by Cleereman's et al, 1989 with the Reber grammar. This will be followed by optimising each of the different RNNs in more challenging Embedded Reber grammar and running a series of experiments. It aims at establishing the preferred class of RNN based on the generalisation performance to a wider population of strings and the fewest memory units.

Chapter 6: Internal Representation and Discussion

This chapter discusses the "black box" of the RNN models developed by applying PCA to unit activations of the hidden layer of each of the different networks. PCA helps to provide a 2-d visualisation of the hidden states each model has produced in response to the next symbol prediction task. This in turn, will enable us to trace the trajectories of the movements within state space and ascertain whether such transitions correspond to those found within the Embedded Reber Grammar.

Chapter 7: Conclusion and Future work

This chapter offers conclusions for the presented work. It summarises the key contributions made by this research, stating the preferred model for the language acquisition task and justification for this. It also highlights the limitations of the work and proposes some suggestions for future research.

Chapter 2
2. Literature Study

2.1 The Nature and Complexity of Language

2.1.1 Nativist vs. Empiricist Perspectives

The acquisition of language by children is testimony to the power of the human mind. Many philosophers consider the ability to communicate through verbal or written language as being a hallmark of human intelligence (MacWhinney, 2004). The acquisition of language is still a matter of debate. It raises a question about whether the capabilities of learning language are innate, or whether we solely use the input from the environment to find structure in language.

Nativists believe that infants have an innate capability for acquiring language. Their view is that an infant can acquire linguistic structure with few inputs and that it plays a minor role in the speed and sequence with which they learn language (Aimetti, 2009) . One of the best-known and most influential linguists supporting this theory is Chomsky. His view is based on the idea that learning language is based on a language faculty, a genetically inherited component of the mind which possesses prior knowledge of the language. Some nativists even believe that it is still possible to acquire language without any input. To support this belief, research was carried out with deaf children showing that they automatically developed observations gained from home with limited exposure to language (Chomsky 1959, Lust 2006). Some nativists have hypothesised that depending on the input, for an infant's native language the innate linguistic knowledge is attributed with complex parameters that the infant must set (Gathercole & Hoff, 2007).

On the other hand, empiricists hypothesise that the input contains much more structural information and is not full of errors as nativists suggested. Many researchers have conducted studies showing that young infants use statistical mechanisms to exploit the distribution of patterns heard in speech during the early stages of language development (Hannon & Trehub, 2005). Infants make use of statistical learning procedures, tracking the probability that sounds appear together; thus, segmenting the continuous stream of speech into separate words. Phonological memory is the capacity to store those speech sound sequences; it comes into play as entries in the mental lexicon are created (Gathercole, 2006). In the process of drawing a new word onto its intended reference, children are guided by their ability to make use of socially based inference mechanisms, also, by their cognitive understanding of the world and their prior linguistic knowledge (Hoff 2009).

Linguistics is concerned with investigating human language as a universal part of human behaviour and thinking. It also seeks to understand the common properties of human language. It is a general term used to define the study of language, it covers several areas of studies such as syntax, semantics and pragmatics. Other disciplines that linguistics can draw on are neurobiology, informatics, neuroscience, and computer science. The major objective of linguistic research is to recognise and describe the rules governing the structure of language. Several researchers view the development of an automated language acquisition system that is capable of learning the rules and structure of language on a large scale as something precious, since it infers and manipulates knowledge immediately from the countless surviving databases and other readable media (McQueen, 2005). The ability to model and therefore understand natural language has influenced many applications of Artificial Intelligence (AI), from speech recognition (Beaufays et al, 2001) to translation (Arnold et al, 1993) and natural language understanding (Allen, 1995).

Language contains a set of sentences. These sentences could be finite or infinite depending on the language. Natural languages have infinite sets. Any sentence is a word (string) of one or more symbols for a specific vocabulary of a language. A grammar is a finite and formal specification of a language. A production grammar is a commonly adopted method used to specify formal and natural languages (Grishman, 1986). Moreover, combining identity words in a sentence with the constraints of syntax and semantics, allows for the expression of concepts in a potentially infinite number of forms. Therefore, the combinatorial power of natural language sentences that have similar meanings can be expressed in a potentially infinite number of different ways.

A model that can learn a representation of the derivational rules of English language gives rise to the possibility of a natural language processing system to parse sentences. Several researchers consider the problem of language acquisition to be a paradox. Nativists who believe in the poverty of stimulus theory argue that fragmentary evidence available to language learners is too inconsistent and incomplete to allow the induction of language without some innate tendency towards the acquisition of language. This creates a paradox, since children who grow up in social environments are nearly always able to learn their language. However, children who are isolated from linguistic input do not always acquire a proper language as adults (Jackendoff, 2002). Children who are exposed to linguistic input are able to learn language. Gold's theorem (Gold, 1967) assumed that an infinite grammar could not be learnt with only positive examples due to the problem of overgeneralisation. His theorem and its implications for language acquisition, shape the fundamental principles of modern linguistics. Gold's theorem is the focus of much debate among those who view the process of language acquisition as a learning process (McQueen, 2005).

The resulting paradox of language acquisition and Gold's theorem formed the basis of Chomsky's theory of universal grammar. This debate and the widely misinterpreted theory are concerned with the human innate pre-specification for language acquisition. Johnson (2004) stated that Gold's theory helped the psychological community to become aware of the possibilities for mathematically modelling psychologically relevant aspects of learning and showed that these models can, at least in principle, establish psychologically interesting limitations on possible hypotheses about cognitive activities like language acquisition. Nevertheless, instead of stating that the entire aspects of language from the lexicon to the grammatical rules are innate, the theory of universal grammar states that the brain's language acquisition capacity is innate. The universal grammar theory stipulates that language learners are born with a brain that is functionally pre-disposed to grammar acquisition. Innate knowledge limits the shape of an acquired grammar to that of possible human languages. Moreover, it contains an approach that can select a grammar compatible with the linguistic input.

2.1.2 Language Complexity and Computation

Natural language sentences are not just a linear arrangement of words. These sentences also contain complex structural relationships between words that are usually characterised syntactically. For example, phrase structures, subject-verb agreement and relative clauses. The critical challenge for the mechanisms of natural language processing is to represent and process such structure therefore we may understand of both their theoretical capabilities and their potential to provide a psychological account of natural language parsing.

The traditional interpretation of natural language processing is based on mechanisms of finite automata, in which discrete states and transitions between states specify the temporal processing of the system. Regular grammars (RG) are comparable to finite automata, that is, a regular language can be defined as a regular grammar and by a finite automaton. In addition, the language accepted by any finite automaton is regular. Automata theory states that a finite state machine (FSM) can process a RG. Nevertheless, a language with centre-embedding is at least a Context Free Language (CFL), where at least a push-down automaton is needed (Rodriguez, Wiles, & Elman, 1999). Essentially, a pushdown automaton is a finite state machine that has the additional resource of a memory-like device that is a stack to keep track of the embedding. To simulate a context free language, the connectionism must have something akin to a stack, such as in a pushdown automata; therefore, it can keep track of a state. Then the stack would predict the relationships in temporal data by internally counting the appearance of signals.

(Hochreiter, Schmidhuber 1997) proposed a type of RNN called Long Short-Term Memory (LSTM) that constructed from units the ability of preserving memory for long periods of time. That is, LSTM is an agate-based RNN architecture that uses gradient descent-based learning to remember, establish and maintain temporal information over a long time-period (Hammerton 2003). LSTMs have unlimited recursive productivity unlike the RNNs (Van der Velde, De Kamps 2006). They can process context-free languages such as $a^n b^m B^m A^n$ for arbitrary (n, m); a valid string from the language consists of a number of *as*, followed by a number of *bs*, then *Bs* and finally followed by a number of *As* which is the same as the number of *as*. However, when they handle this case, their capability to treat combinatorial productivity (this refers to the ability to deal with multiplicative growth of the number of possible events or object combinations) is excluded. Moreover, due to its constant error flow through internal states of memory cells, LSTM exhibits problems similar to those of feedforward networks (which represent the entire input string at once).

There is no predefined internal counting mechanism in parallel-distributed processing systems. Thus, they would have to either build one during training or discover another approach that they can use to keep track of the state. Rodriguez et al, 1999 states that in their experiments, neural networks picked the first choice and developed counters in order to predict the state in a deterministic context free grammar (Lubell, 2010).

Generative grammar is the creation of a formal modelling technique and belongs to the study of language within the discipline of computational linguistics (Chomsky, 1959). Organising a set of production rules, these grammars provide a structure that describes all of the legal sentences in a language.

$$G = \{\Sigma, N, P, S\}$$

Equation 2.1 Grammar definition

Equation 2.1 illustrates a formal definition of a grammar. It is composed of four individual elements. First, the $\sum$ symbol represents a set of terminal symbols and consist of elements that cannot be decomposed into sub-elements (words such as “buy” or “drive”). Then, N is a set of non-terminal symbols, which can be broken down and are used to represent phrases and parts of speech (nouns or verbs). Next, P is a set of production rules that specify which non-terminal symbols can be re-written into which terminal symbol. Finally, S is the start symbol. An example, let $G_1 = (\{0,1\}, \{S, T, O, I\}, S, P)$, where P contains the following productions

$$S \rightarrow OT$$
$$S \rightarrow OI$$
$$T \rightarrow SI$$
$$O \rightarrow 0$$
$$I \rightarrow 1$$

As we can see, the grammar G_1 can be used to describe the set $\{0^n1^n|n \geq 1\}$.

Type	Language	Automata
0	Recursively enumerable grammars (General Rewrite)	Unrestricted phrase structure grammars (Turing Machines)
1	Context sensitive grammars (CSG)	Linear-Bounded
2	Context free Grammars (CFG)	Push down automata
3	Regular Grammars	Finite-state grammars (FSG), Deterministic finite automata (DFA)

Table 2.1 Chomsky hierarchy of grammars

The Chomsky hierarchy, Table 2.1, lists the generative grammars and the corresponding automata required to process them, where the different grammars or languages are categorised into four different types according to their complexity. The simplest type of languages are those generated from grammars belonging to type three, which is the class of regular grammars. Regular grammars are defined by a single non-terminal symbol on the left hand side of the production rule and the limit of one terminal symbol on the right. A production rule is a set of rules generated by a human expert or from numerical input-output data using some heuristic means (Mamdani, Østergaard et al. 1983). Therefore, a formal grammar is a set of production rules (such as $S \rightarrow AB$) which convert one string to another through a series of alternatives. In principle, production rules can take any form such as those used in Probabilistic Context-Free Grammars (PCFGs) (Manning, Schütze 1999) and compositional structures (Jin, Geman 2006). In this research, the production rules give rise to regular grammars.

Regular grammars, such as the Reber grammar, (Reber, 1967), are recognised as memory-less grammars because the next valid state in a sequence can constantly be predicted by the current grammatical state and next input word. In other words, knowledge about past symbols in the input sequences is not needed, only the current state of grammar (or finite state automata being used to process it). Context-free grammars are categorised by their production rule, which allows a non-terminal to be replaced by a set of any number of terminals and non-terminals, as well as the same non-terminal that is being replaced. An example of a context-free language rule is $\{a^n b^n | n > 0\}$ which generates language which is not regular. Where a^n and b^nrepresent a string of n-times repetition of $a \ and \ b$ symbols.

The reason is that context-free grammars allow for the replacement of non-terminal symbols with multiple terminal and non-terminal symbols. Therefore, any computational mechanism to process context-free languages must incorporate a memory. Allowing non-terminal symbols to occur inside the right-hand side of other production rules allows embedded structures to occur. Context-free means that a sequence of symbols can be reduced to a single simple term irrespective of the context in which the sequence occurs. More generally, most natural languages can be described by context free grammars (Allen, 1995).

Type one of the Chomsky hierarchy contains the class of context-sensitive grammars (CSG). A CSG is an unrestricted (recursively enumerable) type of grammar in which every production has the form in Equation 2.2. As the equation shows, it can be seen that the non-terminal A can only be replaced by the set of terminal symbols $abcd$ when it is preceded by the terminal symbol, x, and followed by the terminal symbol, y. This enforces the notion of context-sensitivity.

$$\boldsymbol{xAy} \Rightarrow \ \boldsymbol{xabcdy}$$

Equation 2.2 Example of a production rule from a CSG.

Type zero of the Chomsky hierarchy contains the recursively enumerable set of grammars.

Alan Turing, in 1937, invented a formalism referred to as the Turing machine. A language is recursive if a Turing machine accepts it and halts on any input string (a language is recursive if there is a membership algorithm for it). Equation 2.3 illustrates the production rule for recursively enumerable.

$$uXv \rightarrow uwv$$

Equation 2.3 A production rule for a recursively enumerable string

Where$u, v \in (\Sigma \cup V)^*$,$w \in \Sigma^*$ arbitrary, Σ is a finite alphabet, v is auxiliary symbols Unrestricted phrase structure grammars is another name for recursively-enumerable grammars, because either side of their production rules can contain any sequences of terminals and non-terminals. Equation 2.4 shows the production rule of the unrestricted languages.

$$u \rightarrow v$$

Equation 2.4 A production rule for unrestricted language

Where $u, v \in (\Sigma \cup V)^*$ are equivalent to the enumerable languages. This type of grammar give a close approximation to natural language (Chomsky 1959). Research has shown that monkeys are able to learn simple regular grammars. However, they appear incapable of mastering the rules found in unrestricted phrase structure grammars (Fitch, Hauser 2004).

2.2 Connectionist and Statistical Models of Language Acquisition

Connectionist models might have the potential to achieve language acquisition in such a way that humans do. (Marcus 2003) has discussed the importance of several connectionist models that can represent abstract relations between variables. Those were models that do not incorporate operations over variables i.e. a Simple Recurrent Network (SRN) and models that incorporate operations over variables (a SRN trained by an external supervisor). Marcus states that all models that do implement over variables captured the results whereas, a limited number of connectionist models that do not incorporate any sort of actual operation over variables cannot capture the outcomes.

A concern regarding the ability of connectionist models to represent language acquisition is whether explicit rules are necessary to account for complex behaviour or not. This is because generative grammarians have assumed the need for rules in order to account for the patterns established in natural language (Chomsky, Halle 1968). Marcus, Vijayan et al., 1999 assumed that algebraic rules are necessary for connectionist models to explain language acquisition. However, Christiansen and Curtin pointed out that an SRN model of word segmentation can present the relevant data without invoking any rules. However, that research was in the domain of speech segmentation and further studies are needed to focus on other aspects of language (Christiansen, Curtin 1999).

Traditional symbolic linguistic approaches avoid function-based language acquisition by concentrating on describing linguistic performance, utilising sets of rules and exceptions. A model such as top-down cognition endeavours to function backwards from linguistic structure towards human processing mechanisms. Whereas, symbolic models are fully powerful and the inflexibility of the resultant models which in general are described in this way, cannot easily be applied to general purpose linguistic problems (Corrigan and Iverson, 1994). The designer of the system must take into account the inflexibility arising from rules and exceptions of the symbolic approach, and the system designer ignores the possibility where preference is to be learned by the model itself. Therefore, such systems need a new sets of rules and exceptions.

Probabilistic methods have been applied to a wide range of language tasks within cognitive science, such as speech processing, word recognition, probabilistic phonology, acquisition etc. Probabilistic approaches have been applied across language processing, from modelling lexical semantics to modelling processing difficulty. However, it is extremely challenging to integrate these diverse approaches into a unified model of language. Traditionally, several theoretical issues concerned with psycholinguists are re-framed, rather than resolved, by a probabilistic approach. The processing of language acquisition is the key to cognitive science. However, from the viewpoint of this Special Issue (sophisticated probabilistic models), the first stages towards a cognitive science of language engaged in driving out, rather than building on probability. The improvement of sophisticated probabilistic approaches, such as Special Issue (Chater, Manning 2006), throws these issues into a different light. Such models may be classified in terms of symbolic rules and representations. Therefore, grammatical rules may be connected with probabilities of usage, collecting the likely linguistic, not only what is linguistically possible. With this view, probabilistic ideas give rise to symbolic approaches of language (Chater, Manning 2006).

Connectionist and statistical approaches are used to investigate and discover linguistic patterns found within language. Statistical learning entails the discovery of patterns such as probabilities and statistical similarities in sample language input. In principle, this type of learning ranges from supervised learning, similar to that found in operant conditioning (a certain behaviour of learning that leads to punishment or reinforcement), to unsupervised pattern detection, and includes the sophisticated probability learning exemplified in Bayesian models (Romberg, Saffran 2010). Infants are sensitive to the statistical regularities of the world around them and can learn to recognise patterns in the stimuli they are exposed to. This has led to a variety of computational models of early language learning, based on statistical inference. Statistical models, which have had a significant impact on language problem examples are the Inside-Outside algorithm for acquiring syntactic parsing (Baker 1979) and Expectation-Maximisation (EM) for point estimation whose goal is to find a single optimal model and set of parameters (Ma, Ji 1998, Murphy 2001). In addition, there are Hidden Markov Models (HMM) for acquiring context-free grammars (Jagota, Lyngsø et al. 2001), and the Forward-backward procedure used to find the maximum probability of a sequence *S* in a given model. This is useful for training HMM (Tebelskis 1995) etc. This research focuses on the connectionist models introduced below, rather than statistical models.

Since the early 1990's, an alternative approach to cognitive modelling, known, as connectionism, has gained popularity amongst researchers. This practical field of research uses approaches with designs that are biologically inspired and aspire to simulate the operation of the human brain and nervous system. Unlike symbolic models of cognition, connectionism uses a bottom up approach to cognition that models the learning process itself. Connectionist approaches are suitable for language acquisition because they can learn problem solutions directly from examples from the problem domain.

Gold (1967, cited in Johnson, 2004) proved that an infinite grammar could not be learnt from just positive examples alone. This was based on the assumption that successful acquisition would result in a deterministic grammar. Consequently, acquisition could only be called successful if the language learner possessed an exemplary representation of the grammar resulting in no mistakes. Horning (1969) challenged Gold's theory and illustrated that language could be learnt from just positive examples if the language identification criterion uses a stochastic probability of a winner. The stochastic view of grammar induction is also supported by much of the language acquisition research, including U-shaped learning curves (Rumelhart, Hinton et al. 1986). Connectionism has subsequently provided a realistic means with which the theoretical limitations on language acquisition posed by Gold and Horning could be evaluated (Brown 1973).

2.2.1 Supervised Connectionist Learning Algorithms

In connectionist approaches to language modelling, language acquisition is often assumed to require supervised, semi-supervised, unsupervised, reinforcement and active learning algorithms. Although unsupervised and reinforcement learning are more directly related to known learning mechanisms in the brain, the most successful applications of connectionist modelling in cognitive psychology have employed supervised learning. That is because it is more effective at developing internal representations of linguistic phenomena that can support the complex transformations involved in many forms of cognitive processing (Plaut 1999).

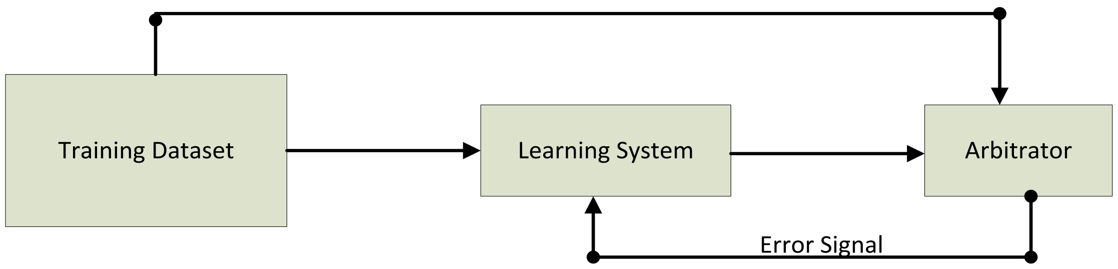


Figure 2.1 Block diagram that displays the form of supervised learning

Figure 2.1 shows a block diagram illustrating supervised learning. In this diagram, the dataset is supervisory training samples. During a supervised learning process, the training input is fed to a learning system. Then, the learning system generates an output, which is then compared with the desired output by an arbitrator that computes the difference between them. The difference, termed error signal in this diagram, is then sent to the learning system as the basis for adjusting the parameters of the learner. The goal of this learning process is to obtain a set of optimal learning system parameters that can minimise the error over the entire training dataset.

Processing is carried out by a number of elements. The elements called *nodes* or *units* have dynamics, which are analogous to simple neurons. Every node receives input from several other nodes, responding to that input according to its activation function, and in transfer exciting or inhibiting other nodes to which it is connected. Details vary between approaches, however, they generally adhere to this universal scheme (Elman 2001).

Rumelhart et al. (1986) error back-propagation algorithm (BP) illustrated how connectionist approaches could be trained to solve non-linear problems. The essential concept behind supervised learning is that the difference between the model's response to an input and the target response, i.e. the "Error", is used to modify the weights in order to shrink the upcoming error. Therefore, the model's individual weights are effectively corrected and reinforced until they reach the correct values that represent the problem. In other words, a network trained using the back-propagation algorithm has its output for the training patterns compared with the required output. The difference between the output and the desired patterns, the error, is used for improving the current weights. At the beginning, the error of the network output is used for computing the weights of the connections to the last layer, then for the weights to the next to the last layer and so on. The error is propagated from the output of the network back to the input layer. The name "backpropagation" (BP) comes from this process. This learning algorithm is not biologically plausible, however it is still commonly used in connectionist learning because it produces a reasonable model for most datasets (Rumelhart, Hinton et al. 2002).

This algorithm has drawbacks: the possibility of converging to local minima instead of the global minimum; temporal instability; and poor scaling properties. Nevertheless, its main problem is the relatively slow rate of convergence, typical for simple gradient descent methods. For these reasons, an enormous number of modifications based on heuristic arguments have been proposed to improve the performance of standard BP. Several of these have been developed in a somewhat specific manner, dominated by the search for speed rather than generalisation and generally evaluated on artificial problems (Alpsan, Towsey et al. 1995). However, the algorithm has some disadvantages such as: during searching for a global minimum, a local minimum where the error derivative of the surface is also zero can be reached and the algorithm can continue there forever. The time to converge the neural network can be very high because the initial weights are randomly defined relative to the final target (de Albuquerque, de Alexandria et al. 2009).

The advantage of supervised learning is that an approach can be trained using just one subset of inputs and desired output pairs for a specific problem. The learning algorithm will construct, from the subset of the training data, an approximate solution to the problem that generalises to unseen samples. Moreover, most supervised connectionist models use distributed representations. In a brain, each stimulus is encoded by a number of varied neurons and each neuron may respond to conjunctions of features that may be present in many different stimuli. For example, each neuron in a visual cortex may react highly to the sight of a specific bar rotated to a particular angle moving at a particular speed in a specific direction. Whereas, another neuron in a sensory cortex, may respond to a touch stimulus on a specific part of the body's surface. However, any stimulus that shares the relevant features of the corresponding neurons will be activated.

(Elman 1995), in his model of the inflection of the English past tense, suggested input features corresponding to speech segments in specific positions; 14 input units correspond to 14 possible beginnings of syllables, six input units correspond to six possible instantiations of the middles of syllables, and 18 input units correspond to 18 possible ends of syllables. For example, the three simultaneous activation units would represent the word "bid", the units correspond to *b* in a beginning syllable position, *i* in a middle of syllables position, and *d* in an end of syllables position. The encoding of other inputs to each of those units would be to participate. This is called distributed representation, an issue of central relevance in the study of cognition in general, and language in particular, which is the nature of the underlying representation of information (Gluck, Myers 2001). A distributed model uses multiple weights to represent each input pattern instead of having single weights holding specific pieces of information. Supervised connectionist models that use distributed representations are usually referred to as parallel-distributed processing models (PDP).

2.2.2 Supervised Connectionist Models of Language Acquisition

Many learning problems in the field of Natural Language Processing (NLP) need supervised training. For example, inducing a grammar from a mass of raw text is a complicated task. However, this becomes much easier when the training sentences are supplemented with their parse trees. Nevertheless, suitable supervised training data may be difficult to obtain (Hwa 2000).

Rumelhart and McClelland (1986) was probably the first study that used NLP models and applied them to language acquisition. Their research involved the acquisition of the markings of the English past tense. Many examples had been given to the network of the form “walk → walked”, the network not only learned to produce the correct past tense for those verbs to which it had been shown, but also generalised to novel irregular verbs (e.g., “begin→ began”). A Multi-Layer Perceptron (MLP) was used to map a representation of the present tense of an English verb onto the equivalent representation of that verb’s past tense (Ševa, 2006). The model showed that the process of abstraction and generalisation of grammatical categories does not need to be based on symbolic rules. It also showed that non-linear change observed in the most developmental processes can be explained through an associative learning mechanism, allowing the extraction of statistical regularities of the domain to be learned. However, this model has been severely criticised for various reasons (Fodor and Pylyshyn, 1988; Pinker and Prince, 1988) relating to its input representation, its incorrect predictions of novel morphological derivations and its artificial training rules. Several studies followed which attempted to overcome the limitations of the Rumelhart et al (1986) model (Marchman and Plunkett, 1996; Jackson *et al* 1996) by, for instance, including a number of realistic training rules.

Another approach published in the field of connectionist language acquisition was initiated by Elman (1992). He produced a recursive neural network called the Simple Recurrent Network (SRN) that was able to learn about sequential dependencies between words in sentences. The model attempts to find structure within each class by hierarchically clustering words into similar grammatical types. Representations of past input words within a sentence degrade as the length and complexity of the input sentences increase e.g., including embedded clauses that are comparable to the difficulty that human language users face, for instance.

“The cat that chased the mouse, which ate the cheese, is hungry”

The network weights perform as attractors in a state space, allowing the system to respond sensibly to novel input. The SRN is a Multilayer Perceptron (MLP) with dynamic extension, which uses recurrent connections to feed back the hidden layer activations at the next time step. This recurrence mechanism permits the SRN to process sequences of inputs such that at any given stage after the start of the sequence it has a representation of what has preceded to inform its processing.

An early work done by Tong et al. (2007), investigated the state of the art ESN, (Jaeger, 2001) for language acquisition in the task of learning. The work is a comparison between the ESN and Elman's SRN from 1992. It was a natural choice for experimentation and it was similar to the work done by Elman (they remove the non-recurrent layers that were between the input and hidden and output layers, therefore to achieve distributed representations of the words). Moreover, the work was to determine whether the ESN could perform a language task that requires a significant amount of memory and generalisation. Tong et al showed that ESNs have the ability to learn to be sensitive to grammatical structure. In his work, Elman applied the basic SRN, as well as an SRN using Backpropagation through time (BPTT), to a the task of predicting the next word in a corpus of sentences generated from a context-free grammar, given the previous words. Elman demonstrated that the SRNs were able to learn the internal representations of the network that were sensitive to linguistic processes, which were valuable for the prediction task. Tong et al. (2007) stated that training such internal representations in the ESN is unnecessary to attain comparable performance to the SRNs. His results proposed that they are capable to form internal representations without learning them. However, the results were really on the corpus that Elman designed. This research is used both approaches to examine them.

2.3 Limitations of Connectionism

Several researchers have used connectionist models to investigate a variety of aspects of language acquisition, from inflectional morphology (Rumelhart, Hintont et al. 1986) to grammar induction (Elman 1990). Nevertheless, several researchers in the field of linguistics have criticised the outcomes of this experimentation and examined the applicability of connectionism to language acquisition (Fodor, Pylyshyn 1988, Jackendoff 2002). These arguments have focused on problems such as adaptive generalisation, scalability, biological plausibility and psychological similarity to human learners.

One of the issues of the connectionist model is dealing with combinatorial productivity of language in natural language processing (NLP). The attention of combinatorial productivity is about the ability to handle a very large lexicon, even if there are simple and limited syntactical structures. Strong systematicity is a form of productivity where it is the ability to learn a word in a given sentence frame and then used it in a novel sentence frame (Marcus 1998). In general, strong systematicity refers to the ability to use familiar words in a novel sentence context and/or novel syntactic locations. An example is when a noun is in the object location for training, and then they will be tested in the subject location. It is very difficult for SRNs to achieve this, specifically because SRNs have to learn in the same time the words and their syntactic usage, thus avoiding the distinction between lexicon and rule. Van der Velde et al, 2004. in their work stated that SRNs failed on the test of combinatorial productivity (when words from sentences with the same type were combined, even though all words that were in the sentences appeared in the same syntactic locations as in training sentences). SRNs are limited with the response to recursive productivity, unlike LSTMs (van der Velde, van der Voort van der Kleij, Gwendid T et al. 2004). Moreover, LSTMs cannot deal with any form of combinatorial productivity (humans are very good at this).

In his paper, Noel Sharkey et al. (2000) has raised questions regarding whether SRNs are sufficient for modelling language acquisition. The sensitivity of SRNs to the initial state is one of the questions and they found that SRNs are extremely sensitive to their initial weight configuration. For instance, only one in every forty-five models they used was actually able to solve the given problem at all. That means that SRNs cannot be an empiricist model of language acquisition. In their study, it was shown that initial conditions interact with training sets. This was an unexpected result even from the point of view of a nativist, because it indicates that in order to confirm consistent language acquisition, the linguistic examples that the learner will reveal must be known in advance. Another issue is the extending long-range sequential capabilities of SRNs, which is where the real problem lies in their limitations with respect to embedded sequences. SRNs display a limited ability to deal with embedded sequences, failing to perform a correct prediction. (Noel Sharkey and Jackson 2000).

Increasing the recurrent network layer units of the SRN is increasing the capacity of the network in two ways. The first is called Short-Term Memory (STM), the capacity of the recurrent layer for storage of the input sequences. The second is named long-term memory (LTM), the capacity for storing the training examples, these arise because of more connection weights to be set. In this way, over-fitting accrues because of increasing the LTM capacity. This means that whenever raising the recurrent network layer, the positive effects of increasing the STM will become overridden by the negative effect of increasing the LTM (Frank 2006).

Tong et al. (2007) stated that in the comparison they did between SRN and ESN for learning grammatical structure, the ESN could suffer from cognitively plausible errors. This is the ability for the network to learn systematic language behaviour from exposure to only a small proportion of possible sentences. Related with the training, Jaeger et al. (2012) stated that in order to possess Echo State Property (ESP) (the reservoir state is an echo of the complete input history) spectral radius could be tested more than unity. Therefore, specific weight patterns cannot be lost. He indicated that Frank's, 2006 and their results are insufficient to confirm that ESN are capable of scaling to a large natural language corpus.

2.3.1 Argument against Biological Plausibility

Since the advent of connectionism, there has been continuing arguments concerned with understanding the relationship between ANNs and their biological counterparts. It is stated that these arguments show the field's loss of focus from its founding principle of biological plausibility. MLP (Rumelhart, Hintont et al. 1986) is a connectionist model that is not a realistic model of the structure, learning process or even the singular neurons that are found in biological neural networks.

Recently, connectionist models have been used to model and predict certain aspects of brain function. A criticism of such models, however, has been their dependence on BP (back-propagation). The argument against the BP is that it is considered biologically implausible, since it is based on the error back propagation where the stimulus propagates forward and the difference between the actual and the desired output propagates backwardly. Whereas in the cerebral cortex, a stimulus is generated when a neuron fires across the axon towards its end to make a synapse onto another neuron input (Rosa 2005). With this, connectionist networks are deliberately analogous to neural processes in the brain. Further biologically plausible ANN models are concerned with the connectionist architecture; associated directly with the cerebral cortex biological structure, or focused on the neural features and the signalling between neurons. The focus of such research is to innovate a more proper model concerning the biological properties, structure and functionalities, containing learning processes, of the cerebral cortex, not ignoring its computational performance. The selection of the approaches upon which the proposed description is based takes into account two main criteria (Rosa 2005): the fact that they are considered biologically more realistic and the fact that they deal with intra and inter-neuron signalling in electrical and chemical synapses. In addition to this, the period of action potentials is taken into account. However, in this study ANNs are used as biologically inspired models to solve and investigate language acquisition and are not concerned with simulating cognitive processing accurately.

2.3.2 Argument against Connectionism for Developmental Cognitive Modelling

In the last three decades, connectionist modelling has formed an important approach in the computational study of cognition. It is distinguished by its focus on the essentials of neural computation considering the primitive components, which are the basis for the cognitive level models. Connectionism has been applied to various cognitive abilities such as attention, perception, action, language, concept formation, models of memory and reasoning (Thomas, McClelland 2008). Many of these models attempt to capture adult function, however connectionism is concerned with learning internal representations.

The main struggle for the connectionist theory of the last three decades has been that many ANN models, especially the models constructed upon feed-forward backpropagation, were confirmed to be implausible for both physiological and theoretical reasons (Newell 1994, Lachter, Bever 1988), however these models are still widely used in many applications because of their powerful abilities. BP networks, for example, can approximate almost any function and are easy to train, because any set of psychological data can be presented as a function that maps input to a behaviour, and because a BP network can approximate almost any function. It is therefore not surprising that such networks can model a wide range of psychological and other phenomena.

Adaptive generalisation abilities of connectionism have been explained by Noel Sharkey et al (2000): if connectionist models are to perform functions of human cognition, they must present similar developmental properties to those detected by humans. His grammatical- transfer trials demonstrated that the SRN is incapable of extracting previous grammatical knowledge. These experiments show that if a model trained on a particular grammar is exposed to new lexical items, the training times are poorly affected. These results are at odds with human performance, where language acquisition gets easier as development progresses.

There is a relationship between the ability to perform grammatical-transfer and another undesirable behaviour that happens in gradient-descent based connectionist models. Catastrophic forgetting, described by (French 1992), is the inability of a neural network to retain old information in the presence of new information. Training requires that the trained knowledge must be continually refreshed by cycling through the whole dataset; otherwise, the existing knowledge is forgotten in favour of the new knowledge. Children, however, are capable to learn new knowledge without overwriting the existing knowledge. This makes the connectionist behaviour psychologically implausible (Noel Sharkey et al, 2000).

2.3.3 Learning Deterministic Representations Using a Continuous State Space

The SRN is a connectionist model (Elman 1990) that has been applied to the language acquisition task in the form of grammar induction. The task was to learn simple approximations of natural language, context-free and regular grammar. These trial results proposed that dynamic recurrent networks (DRNs) can learn to mimic finite-state automata. However, other models of connectionism show several fundamental difficulties, which may derive from using a model with a continuous state-space to approximate a discrete problem (McQueen, Hopgood et al. 2005).

Supervised connectionist models have shown the capability to learn simple formal languages and there are ways of overcoming their instability when dealing with long sequences that were not part of their training set (Omlin 2001). SRN (Elman 1990) has shown its ability to partition its state-space into areas that are supposed to approximate the states in a grammar. Nevertheless, their sensitivity to initial conditions can be explained in that each transition between regions of state space will result in a slightly different trajectory, which causes instability when transmitting state trajectories that were not seen during training (McQueen, Hopgood et al. 2005).

This kind of behaviour is one of the characteristics of supervised dynamic connectionist models and can be determined as both a strength and weakness of this class of model. Although this representational power makes the model exceed Deterministic Finite Automata (DFA) and mimic non-deterministic systems, it is a significant disadvantage when attempting to mimic the deterministic behaviour fundamental to deterministic finite automata.

A number of researchers have tried to produce state-space models by using a step-function for the hidden layer units (e.g. Zeng, Goodman et al. 1993). Although the technique eliminates the instability problem, using a non-differentiable function means that the weight update algorithm, which uses the sigmoid function, can only approximate the error signal. This weakens the power of the learning algorithm, in some cases leading to the model learning an incorrect representation of the DFA and increasing training times. (McQueen, Hopgood et al. 2005).

Simple Synchrony Network (SSN) (Henderson, Lane 1998) is a model that overcomes instability in continuous state-space models that operate Temporal Synchrony Variable Binding (TSVB) to encode entities using pulsing binary threshold units. This technique can enhance the power of continuous state space models by providing static building blocks within the ever-changing sea of internal representations.

Representations done by RNNs have revealed some inherent problems with the principle of language learning. According to Kolen, 1994 there are two major problems with extraction of a learned automaton. First, sensitivity to initial conditions results in non-deterministic machines where their trajectories are indicated by the initial state of the network and the dynamic of the state transformation. Secondly, trivial changes in observation strategies can cause one to induce behavioural descriptions from a massive number of computational complexity classes for a single system.

2.4 Discussion and Conclusion

Many linguists define language acquisition as a complex and powerful system that describes and possibly shapes every aspect of human perception (Gordon 2004, Sapir 1929). Nevertheless, the operation of language acquisition itself is a paradox (Jackendoff 2002). Although children seem to be involved in processing their native tongue learning, the linguistic input that they experience appears overly sparse for the acquisition of a grammar. Classical theories of linguistics have therefore supposed a particular level of innate knowledge that constrains language acquisition and provides children with an earlier knowledge of grammatical structure. The apparent intractability of the problem even when considering an automated language acquisition system has engaged researchers to study this field.

Traditional linguists have aligned with modern connectionism by demonstrating that linguistic input has not got as much structure as was previously thought. Connectionist and statistical models to learning are of great relevance to the investigation of language acquisition as they provide a principled conception of the learning process. Second, they also offer potential learning mechanisms for particular aspects of language. Lastly, these models allow inferences concerning the nature and extent of innate knowledge, either in relation to innate learning mechanisms or to innate knowledge per se. Statistical methods have the problem that they cannot be directly implemented as connectionist networks such as nonparametric statistical methods (rank correlation) (Redington, Chater 1998). There is hypothesis that new-borns begin life using statistical processes for simpler problems, for example learning the sound of their native language and building a lexicon, while grammar is learnt by non-statistical learning, later in development. (Seidenberg, MacDonald et al. 2002) assert that statistical learning ends when learning grammar begins. However, this is still a matter of debate and it has proven very difficult to detect this boundary (Aimetti 2009).

In conclusion, adaptive connectionist models are usually capable of detecting statistical regularities in the set of input patterns presented to them, after being suitably trained. They are able to perform in reasonable ways when presented with novel input patterns, based on what they have learned during training. Their abilities, in general called induction, interpolation or generalisation, enable them to operate much more flexibly than systems that depend on explicit, rigid rules.

Chapter 3

3. Neural Network Architectures

In this chapter, a number of Recurrent Neural Network (RNN) models are reviewed. The RNN models reviewed are those commonly used for language modelling tasks and more specifically, for the next-symbol prediction task where the aim is to induce a reliable underlying finite state automaton directly from linear input sequences (or sentence examples).

3.1. Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are usually employed in cognitive science to process symbol sequences that represent natural language information. RNNs are normally adaptations of the classical Feed-Forward Multi-Layered Perceptron (FF-MLP) model to add recurrent connections, which let the network activations to feed back to itself or previous layer as input. This kind of connection, produces internal memory that allows the RNNs to construct dynamic internal representations of temporal order and dependencies that may be present in the data. (Binner, Tino et al. 2010). The units that receive feedback values are referred to as context or state units. In addition, nonlinear activations are assumed to be used; the extension of RNNs is naturally done by universal function approximation properties of FF-MLPs. This means they provide the processing system dynamic properties that are responsive to temporal sequences. Many techniques and methods have been tried using feedback connections (Picton 2000). Moreover, analysis of state space trajectories in connectionism provides new insights into the types of processes that may be considered as learning models to acquire and represent language without reference to traditional linguistic theories (Čerňanský, Makula et al. 2007).

A variety of connectionist models have been studied with a view to modelling natural language processing (language acquisition), although there is no consensus regarding the best architecture to use (e.g. (Cleeremans, Servan-Schreiber et al. 1989, Plunkett, Karmiloff-Smith et al. 1997, Elman 2001, Tong, Bickett et al. 2007). The way in which the layers of an RNN are interconnected determines its structure. Therefore, various RNN topologies have been investigated that have different structures and different learning algorithms. SRNs (Jordan, 1986 and Elman, 1990), Time delay neural networks, nonlinear autoregressive network with exogenous input, Multi Recurrent Networks (MRN), ESN and long short-term memory have been studied.

3.1.1 Jordan Network

(Jordan 1986) describes a network (Figure 3.1) in which the output associated with each state is fed back and combined with the input representing the next state. The neurons in the state (also known as context) units and the output units are expected to be represented as distributed patterns of activation on separate pools of processing units (set of units of simple processing units which communicate by sending signals to each other over a large number of weight connections). The connections from state units to the hidden units act as input for the network architecture. This network implements the output function through weighted connections from the state units to the output units.

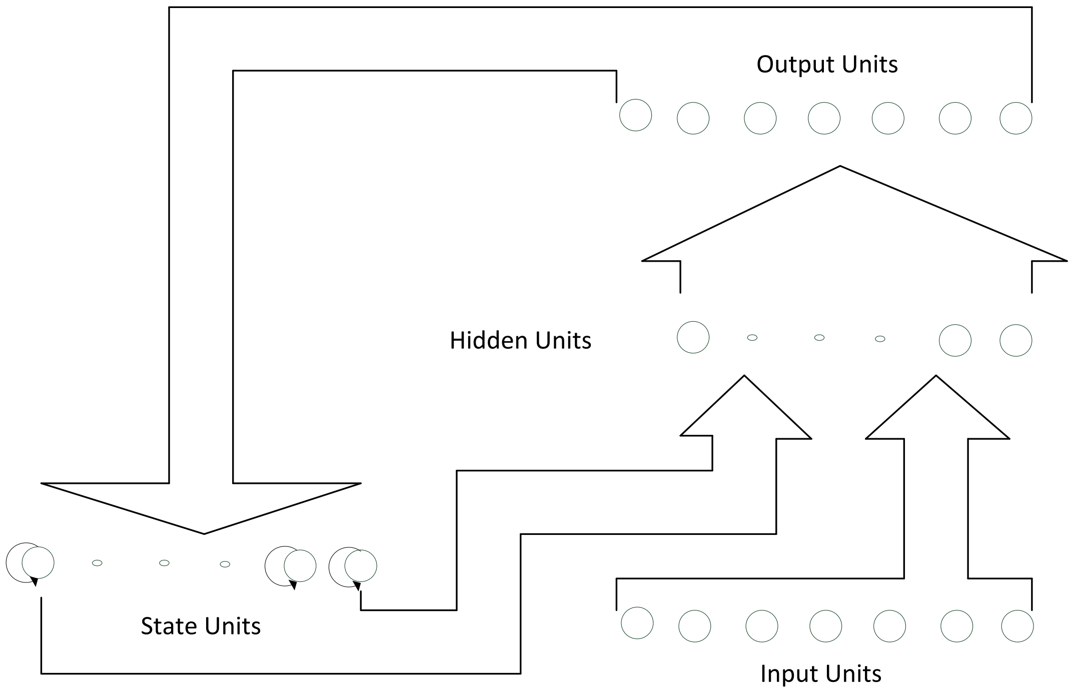


Figure 3.1 The Jordan Network (Jordan 1986)

Hidden units take their input from state units and input units. Each state unit output is derived from a combination of a recurrent connection from the state unit to itself, and from all the output units. Therefore, the current state depends on the previous state and on the previous output. (Jordan 1986). This property, common to all RNNs means that they should be able to exploit information beyond the existing input. However, in practice this cannot really be exploited (Dorffner 1996). If the weight of a connection to a context unit is close to one, the unit (using sigmoid as activation function) saturates very quickly to maximum activation, where additional inputs have less effect. If the weight is very small in comparison to one, the impact of past estimates quickly becomes negligible.

3.1.2 Time Delay Neural Recurrent Network (TDNN)

Another class of dynamic neural network, known as the time delay neural network, is presented by (Waibel 1989). TDNNs depend essentially on a special kind of memory known as "tap delay line", where all recent outputs are buffered at different time steps. These 'delay' connections between the output and input layers provide the network with additional memory (Marques, Souza et al. 2005). Representation of a tap delay connection is illustrated in Figure 3.2. The response of these neural networks in time t is based on the output in times $(t-1),(t-2),\dots,(t-n)$.

The output of this network is a function of the current external input together with outputs as given by:

$$y(t) = f(x(t), y(t-1), y(t-2), \dots, y(t-n)) \qquad (3.1)$$

Where x(t) is the input at time t and y(t-1) is the output at time t-1, n is the maximum adopted time-delay. The activation of the unit f at any time step is calculated as follows:

$$y_i^t = f(\textstyle\sum_{j=1}^{i-1} \sum_{k=0}^{d} y_j^{t-k} . w_{ijk}) \qquad (3.2)$$

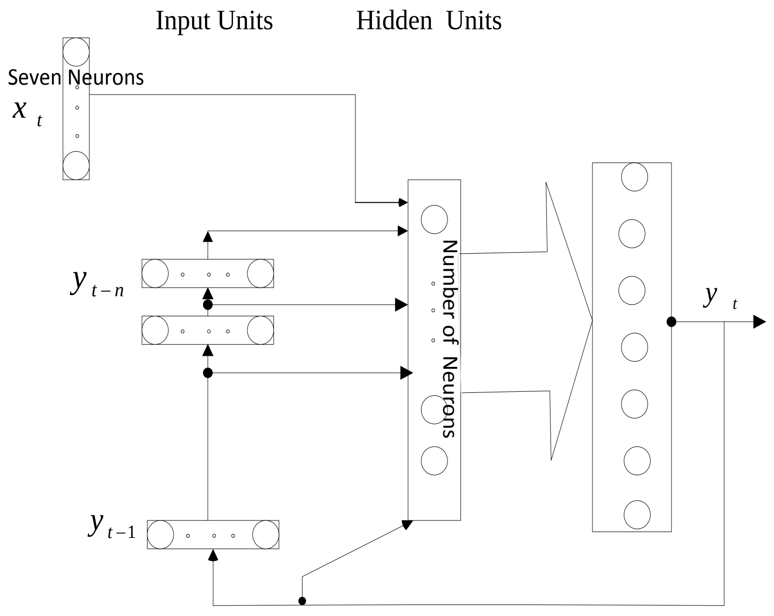


Figure 3.2 Shifting Recurrent Network

Where y_i^t is the output of the unit i at time t and w_{ijk} is the weight to the unit i from the output of the unit j at time t-k.

TDNN has the ability to store temporal information explicitly using time-delayed structures (Wah, Qian 2004). Therefore, the delay mechanism supplies the network with memory to deal with temporal structure by having the previous state stored in a sequence.

3.1.3 Nonlinear Autoregressive Network with Exogenous Input (NARX)

The nonlinear autoregressive model process with exogenous input is a discrete time nonlinear system, which is established to be equivalent to a Turing machine (Menezes Jr, José Maria P, Barreto 2008). The architecture, known as the NARX network, is depicted in Figure 3.3.

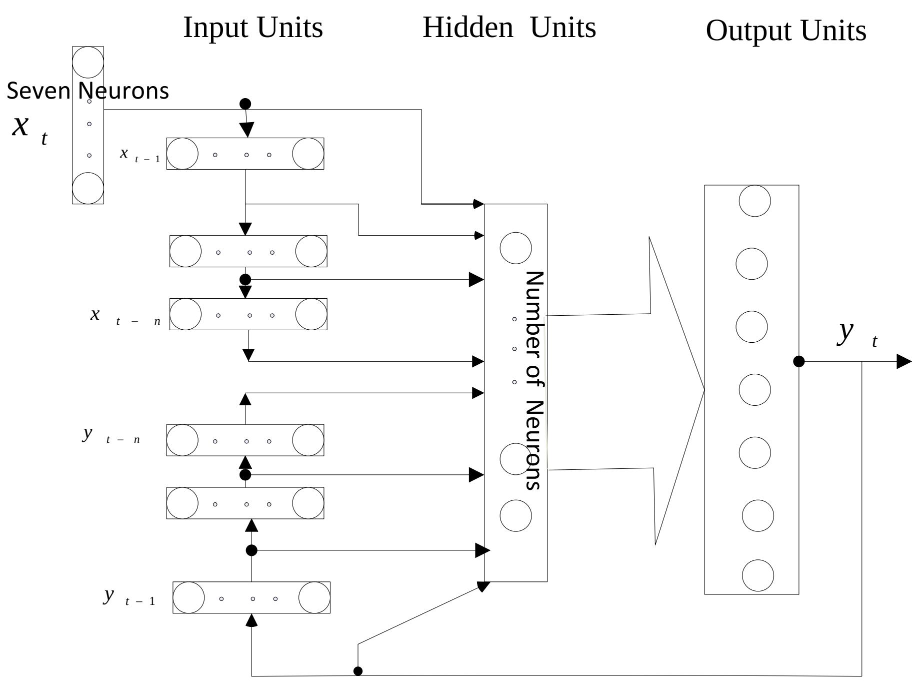


Figure 3.3 NARX network with d_x delayed inputs and d_y delayed outputs (Diaconescu 2008)

NARX is a feedforward neural network with embedded memory such that copies of both the previous inputs and outputs are presented to the hidden layer via two series of time-delayed buffers as used with the TDNN. This makes the network dependent on d_x previous sequence elements and it is identical to using d_x input units being fed with d_x adjacent sequence elements. This input is normally denoted as a time window since it provides a limited view on parts of the sequence. It can also be viewed as a simple way of transforming the temporal dimension into a spatial dimension (Diaconescu 2008).
NARX is an important class of discrete time nonlinear system and can be implemented using the following function:

$$y(t) = f(x(t-1), x(t-2) \dots x(t-d_x), y(t-1), y(t-2) \dots y(t-d_y)) \quad (3.3)$$

Where *x(t)* and *y(t)* are respectively the input and output of the model at time step *t* and the input and output memory orders are denoted by d_x and d_y. The nonlinear mapping function *f* describes the hidden layer mapping of its inputs to its outputs. *y(t)* is the output of the net and returned to input as exogenous to the net. For these features, NARX is described that has a similarity of the dynamical characteristic of a system efficiently and a short-term memory (the dynamical features comes because there is one of the input of NARX is the output of the network) (Jiang, Song 2010). Moreover, it has been shown that this model is good for modelling nonlinear systems such as discrete-time non-linear systems (Chen, Billings et al. 1990), dynamic system identification (Qin, Su et al. 1992) and long-term dependencies (Siegelmann, Horne et al. 1997).

3.1.4 Simple Recurrent Networks (SRN)

Simple Recurrent Network is an artificial neural network as shown in Figure 3.4, where the activations of the hidden units from time t are used as input to the network at time $t+1$. Recurrent connections provide the network with access to its prior state and subsequently the network has the ability to detect and learn temporal relationships within the data. The input units I and hidden units (recurrent layer) R and the output units O are fully connected through the first order weight W^{RI} and W^{OR}, respectively, as in the feedforward multilayer perceptron (MLP). Time delay connections feedback the activities of recurrent (hidden) units $R^{(t)}$ to the context layer, i.e. $C^{(t)} = R^{(t-1)}$. Thus, each recurrent unit is fed by activities of all recurrent units from previous time step through recurrent weights W^{RC} . Previous time step of the recurrent units activate itself and can be understood as an extension of input to the recurrent units. They represent the memory of the network.

Given input symbols in time $t, I^t = (I_1^t, \dots, I_j^t, \dots, I_{|I|}^t)$ and recurrent activities $R^t = (R_1^t, \dots, R_j^t, \dots, R_{|R|}^t)$, the recurrent unit's net input $\hat{R}_i^t$ and output activity R_i^t are computed as

$$\hat{R}_i^t = \sum_j W_{ij}^{RI} I_j^t + \sum_j W_{ij}^{RC} R_j^{t-1} \qquad (3.4)$$

Where

$$R_i^t = f(\hat{R}_i^t) \qquad (3.5)$$

The output unit k computes its net input $\hat{O}_i^t$ as following:

$$\hat{O}_i^t = \sum_j W_{ij}^{OR} R_j^t \qquad (3.6)$$

Where

$$O_i^t = f(\hat{O}_i^t) \qquad (3.7)$$

Where $|I|, |R| and\ |O|$ are the number of inputs, hidden and output units, respectively, and f is the activation function.

(Elman 1990) used the SRN to solve sequence prediction tasks where the network is presented with a sequence of symbols one at a time and is required to predict the next symbol in the sequence at each time step and over all sequences.

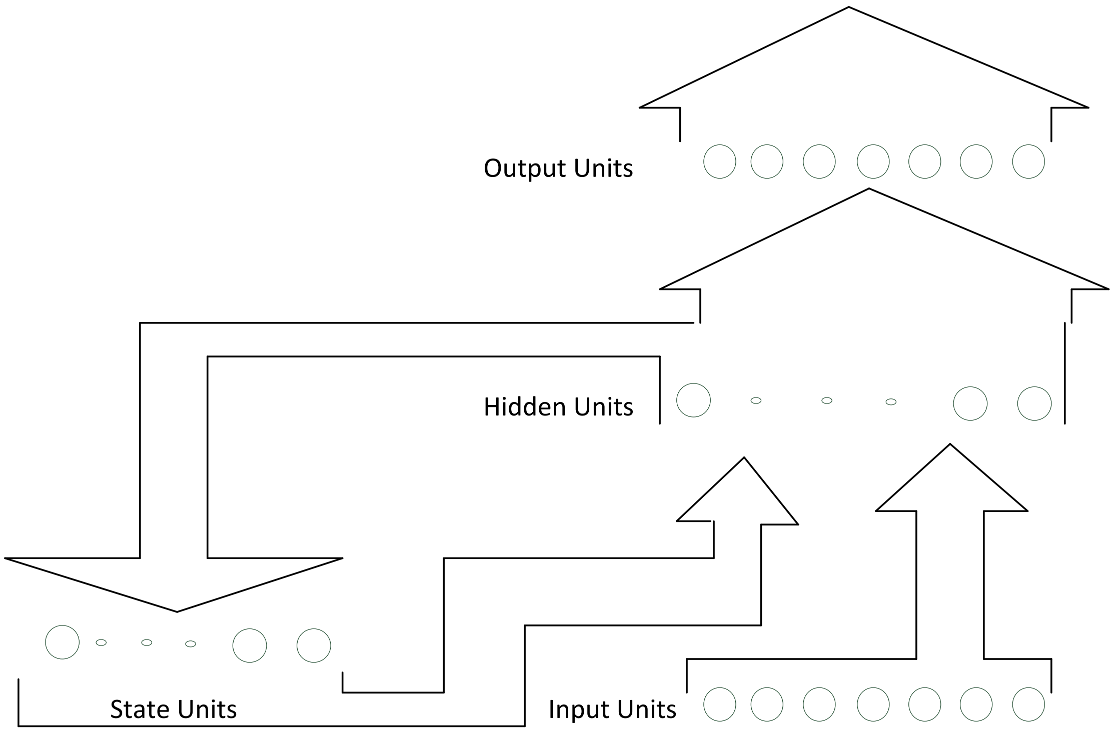


Figure 3.4 Simple Recurrent Network

Elman (1990) introduced the SRN. This structure has the potential to master a large corpus of sequences (the ability to mimic closely a finite state automaton FSA in its behaviour and its state representations) with the limited means of a learning procedure that is local in time totally (Cleeremans, Servan-Schreiber et al. 1989). The mechanism of the simple recurrent network operates as follows: the input sequences are presented to the input layer one element at a time. The purpose of the input layer is to feed the hidden layer through a weight matrix. In addition, the activations of the hidden layer return copies to a context layer after every step, which provides another input to the hidden layer (information about the past). Since the activation of the hidden layer depends on both its previous state (the context unit) and on the current input, SRNs have theoretical capacity to be sensitive to the whole history of the input sequence. Nevertheless, in practice there are limitations restricting the time span of the context information. This has been estimated to be effectively 10 to 15 steps (Stoianov, 2001). Ultimately, the hidden layer neurons output their values through the weight matrix connecting the hidden layer to the output layer. Then, the activation of it is the product of the network.

SRNs have been applied to grammar learning (Elman, 1990), word prediction (Lewis, Elman 2001) and many other problems that require sequential information processing. SRNs have been successful in extracting the fundamental structure of complex embedded sentences. However, their success is dependent on the details of the problem and training. (Elman 1993) discovered that presenting the complete range of sentence structures to the network in the corpus results in a leak of the performance on the complex shapes. For example, SRNs failed to achieve verb agreement across clause boundaries. On the other hand, successful results were obtained in the application of an incremental training technique (Plunkett, Karmiloff - Smith et al. 1997).

3.1.5 Multi Recurrent Networks (MRN)

A further development of the RNN is to employ multiple feedback connections to enhance performance. (Ulbricht 1995) introduced the Multi-Recurrent Network (MRN) architecture that is illustrated in Figure 3.5. The construction provides three levels of feedback allowing recurrent connections from the following:

1. The output layer back to the input layer as established in Jordan networks (1986).
2. The hidden layer back to the input layer, as found in (Elman 1990) SRNs.
3. The connection from the context units within the input layer back to themselves (self-recurrent links).

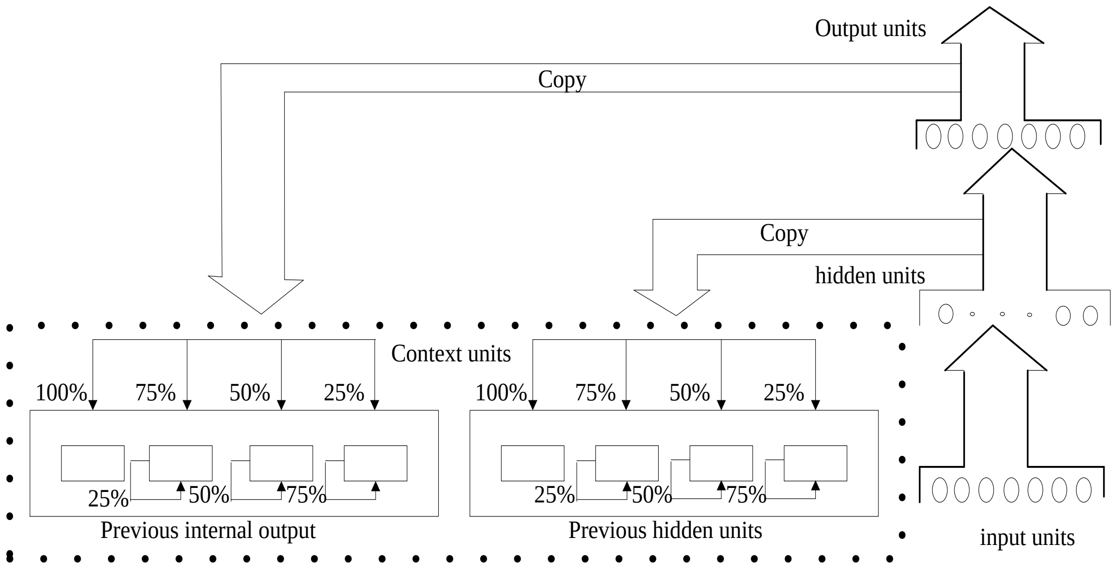


Figure 3.5 Architecture of Multi-Recurrent Network

There are no recurrent or self-recurrent connections from external input units. There are additional banks of context units (memory banks) on the input layer. The number of additional memory banks followed the Ulbricht (1995) report. Four memory banks used φ=4. The context layer represents a flexible memory structure. In the example shown in Figure 3.5, the leftmost banks are 100% copies of the previous output/context and represent short-term memory. The next banks are 75% copies of the output/context with 25% self-recurrent feedback of the previous time step. The third banks are 50% copies of the output/context with 50% self-recurrency. The final banks are 25% copies of the outputs/context with 75% self-recurrent, representing a longer, more rigid memory. Binner (2010) demonstrated that moving beyond the four banks does not lead to enhanced performance. Rather, it is the number of units within each bank that is pivotal to the performance of the network and this can be optimised using the validation set. The MRN architecture can be implemented using the following function:

$$y\ (t+1) = g(f\left(c(t), x(t), W_f(t)\right), W_g(t)) \quad (3.8)$$

Where: $y(t+1)$ indicates the predicted values of the symbol. *x(t)* is the external vector of input variables; *c(t)* is the concatenation of the previous hidden state vector with four delays of varying strength and summation of elements of previous output vector with four delays of varying strength; $W_f(t)$ is the weight matrix connecting the input layer to the hidden layer; $W_g(t)$ is the weight matrix connecting the hidden layer to the output layer; vector function f returns activation vectors from the hidden layer; and function g returns activation vectors from the output layers.

3.1.6 Long Short Term Memory (LSTM)

An RNN called Long Short-Term Memory (LSTM) is described in this section (Hochreiter, Schmidhuber 1997) and is primarily designed for supervised time series learning (Bakker 2001). The main difference between LSTM and the traditional RNNs techniques is its ability to overcome the vanishing gradient problem. That is, the influence of a given input on the hidden layer and thus on the output of the network, either vanishes or blows up exponentially as it cycles around the recurrent connections.

LSTM networks consist of three layers: input, hidden and output layers as shown in Figure 3.6. The main difference between LSTM and traditional RNNs is the hidden layer. The hidden units can contain one or more memory cells which are connected to all cells and gates. In addition, these cells are connected to the output units and the gates are connected to other cells and gates in the hidden units. The memory cell is a linear unit with self-connection that has a weight of value one. The cell maintains its current activation over time when there is no input. The input to the memory cell is passed through a squashing function and gated (multiplied) by the activation of the input gate. Therefore, the input gate controls the flow of activation into the cell. Before the memory cell's output is gated, it is passed through a squashing function by the output gate activation. Therefore, the flow of activation from cells to outputs is controlled by the output gate. The input and output gates learn to open and close in order to allow new information into the cells and allow the cells to influence the output during the training process. (Hammerton 2001, Hammerton 2003).

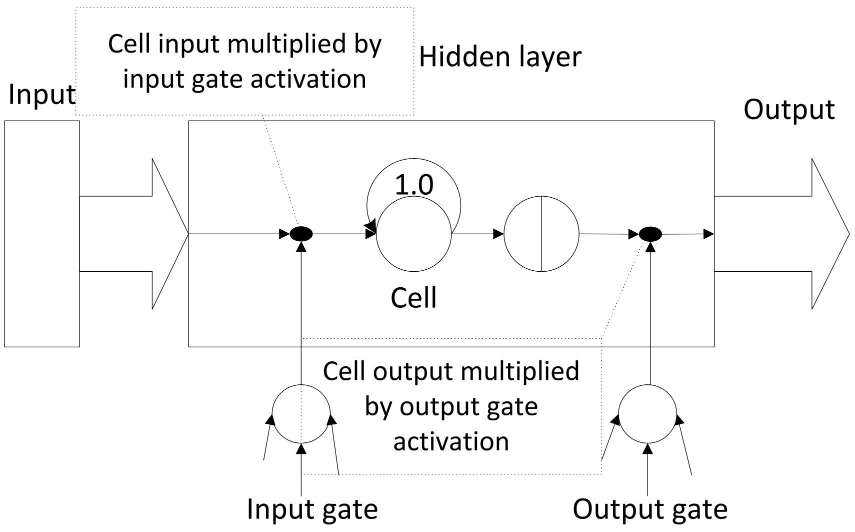

Cell input multiplied by input gate activation
Hidden layer
Input
Output
1.0
Cell
Cell output multiplied by output gate activation
Input gate
Output gate

Figure 3.6 A LSTM memory block with one cell and its gate units

3.1.7 Echo State Networks (ESNs)

Supplemental recurrent neural networks, also known as ESN, were proposed by (Jaeger 2001). ESNs were developed to learn nonlinear systems for prediction tasks. This type of RNN is constructed from the concept that the recurrent dynamic part of RNNs does not need training. Instead, it functions as a non-specific memory; that is why it is called a dynamic reservoir, which makes it keep information about the input sequence by allowing activations to rebound around the recurrent units (Frank 2006). The two architectures for ESNs that are investigated in this research are shown schematically in Figure 3.7. A typical ESN is an RNN in which all the connections (weights) from the input to the hidden units (reservoir or recurrent) and from the output to the reservoir units are fixed. The only trainable weights are from the reservoir nodes to the output units. In the jumping connection architecture, they remove the connections from the output to the reservoir are the only noise version of the current step's inputs (Tong, Bickett et al. 2007). Therefore, the training weights in this model are defined as:

$$V_{out}(t)=W_{out}(V_{in}(t),V_{hidden}(t)).$$

Where $V(t)$ is a vector that indicates the activation of the hidden units at time t. the rest of the architecture is the same as the standard ESN.

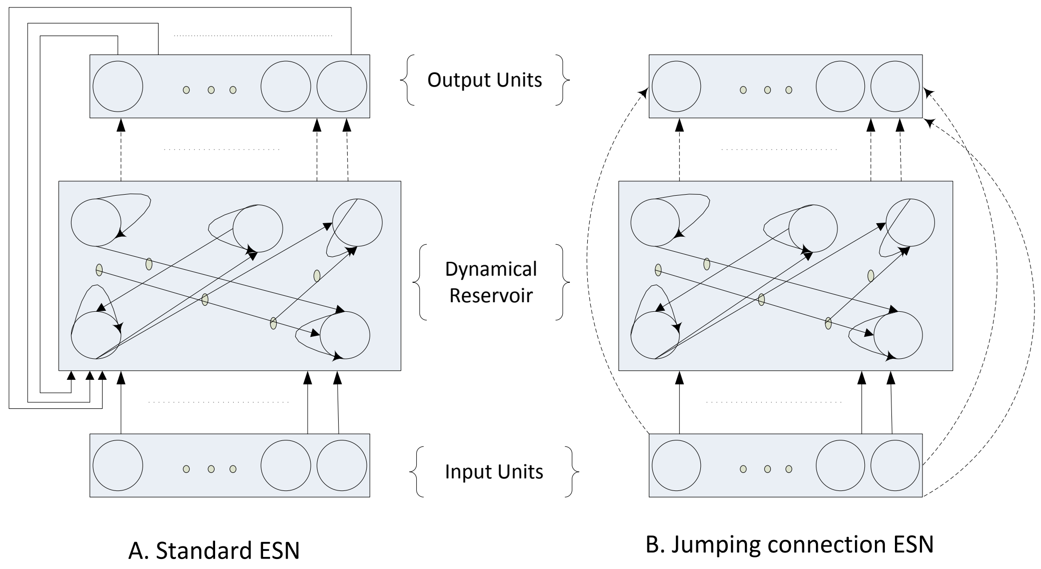


Figure 3.7 ESN Architectures: Solid arrows indicate fixed connections and dashed arrows indicate trained connection.

ESNs use a simple learning algorithm for dynamical systems. It works by training linear readout neurons that combine the signals from a random fixed, excitable "reservoir" network (pseudoinverse method is used). A standard ESN is a simple discrete-time RNN that has three layers: an input layer (units), a recurrent layer also called reservoir, internal or hidden units. It is the core of the ESN structure and the readout layer extracts information from the reservoir. The network is fed each time step t with input vector u_t, which drives the dynamic of recurrent layer, x_t and output vector y_t . An input vector $u(t+1)$ at time step $(t+1)$, with activations of recurrent units, $x(t+1)$, is generally updated according to:

$$x(t+1) = f\left(Wx(t) + W^{in}u(t+1)\right) \qquad (3.9)$$

In addition, output units, y(t+1) are used to extract interesting features from this rich reservoir of dynamics, therefore, only reservoir output connections, W^{Out} are modified during the learning/ training process, the output is computed according to

$$y(t+1) = g(W^{Out} * x(t+1)) \qquad (3.9)$$

Where f is the internal unit's activation function (tansig, sigmoid, etc.),

W, W^{in} *and* W^{Out} are Input-hidden, Hidden-hidden, Output-hidden connection matrices respectively and $y(t)$ is the output of the ESN. The weight matrices are initialised randomly and are kept fixed except for the output matrix, which is adapted through learning. The weights in the reservoir layer are assigned randomly to sparsely and randomly connect neurons. The magnitude of the spectral radius determines the persistence of memory. To scale the initial weights to a desired spectral radius we calculate hidden weights as:

$$W^{in} = \frac{\alpha W'^{in}}{|\lambda_{max}|} \qquad (3.10)$$

Where λ_{max} is the maximum eigenvalue of W'^{in} and α is the spectral radius $0 < \alpha < 1$. This is used to ensure the echo state property (ESP) i.e. that the activation of the reservoir layer forgets asymptotically. Because of this, the history of the input sequence has a decreasing effect on the current activation as new symbols are input to the network. Scaling the weights of the reservoir determines the ESP. This guarantees that the activity of the reservoir, driven by input sequences with comparable history, will converge to close regions of the state space (Rachez, Hagiwara 2012). Settling Time (ST) is measured by the number of iterations allowed in the recurrent layer after its excitation by an input and before the sampling of the output. A smaller ST means that the iteration is short term or truncated and the output is available soon after the input is passed to the reservoir. However, a higher ST indicates more iteration in the network and that the output is delayed (Venayagamoorthy, Shishir 2009).

The ESN method differs from other RNN approaches in having a large number of recurrent neurons (in the order of 50 to 1000 neurons). As previously stated, only synaptic connections from the RNN to the output units are updated i.e. the only connections that are updated are the connections from the recurrent layer (reservoir) to output units. In this investigation two different training approaches were used as mentioned before. The standard ESN by Jaeger and the jumping connection version where there are trained connections from input to output layer as well as from reservoir to output layer, as shown in Figure 3.7 B.

3.2. Summary

This chapter provides an overview of different RNN approaches that might be used for grammar induction. RNNs have been applied to grammar inference such as SRN, MRN, NARX and ESN. The performances of these different architectures for grammar induction has been investigated. Some modifications have been done on these models (e.g. learning rate, number of hidden units, activation functions, etc.). Almost all the networks share the same learning algorithm (back-propagation through time) except for the ESN structure. This research considers several RNN approaches with some modifications in their architectures and parameters to try to enhance their performance. Moreover, SRN, Jordan, TDNN, NARX and MRN are used to evaluate the role of different recurrencies and the role of memory rigidity (in terms of past and current information). This will be discussed in chapter four. This research aims to investigate their limitations and make comparisons between the various RNNs to find the optimal approach for grammar inference.

Chapter 4

4. Data and Methodology

This chapter describes the data sets and methodologies used to fit, select and evaluate the RNN used to model the data. One of the principle aims of this study is to determine the class of RNNs that is able to robustly learn to represent the underlying finite state automata describing the languages used. In this chapter, a number of popular RNNs will be considered.

4.1. The Reber Grammar Datasets

Fundamentally, two different types of data sets are used in this research. The first data set type consists of sentence strings generated from a regular grammar. The purpose here is to ascertain whether the RNNs are able to discover the underlying *finite state automaton* used to generate the data. Finally, the second type consists of sentence strings generated from a context free grammar. The purpose here is to discover whether the RNNs are able to discover the underlying *linear bounded automaton* used to generate the data. Both data sets are based on the popular Reber Grammar (Reber, 1976), which is discussed below.

Many researchers have argued that people can learn complex tasks in distinct ways based on implicit or explicit learning (e.g. Reber 1976, Dienes, Broadbent et al. 1991). The methods are distinguished in two ways: the conditions that elicit them and the type of knowledge that they result in. Implicit Learning (IL) is the learning of the complicated information or data without fully understanding what has been learned (Sun, 2008). IL has been illustrated using a variety of experimental paradigms that differ in the type of indicator used and in the type of implicit knowledge acquired. In the Artificial Grammar paradigm (AG), artificial grammar learning consists of complex rules that determine the sequence of letters producing short strings (Rosas, Ceric et al. 2010).

Artificial grammar learning is the paradigm that has been used largely to investigate the acquisition of implicit knowledge (Reber 1976, Reber, Kassin et al. 1980). The practise contains a comparison of performance on two tests. Initially, participants study sequences of symbols generated by an artificial grammar shown in Figure 4.1. In the first test, participants are asked to distinguish between grammatical and ungrammatical sequences. After that, they may be asked, whether they recognise sequences, whether they can orally report the foundation of their choice, or may be questioned about the nature of the grammar. Reber, 1976 found that participants classified 79% of test sequences in accordance with the rules of grammar. However, participants were incapable of describing those rules. A reason for selecting this grammar is due to its previous widespread use in numerous psychological trials on implicit memory (Reber, Kassin et al. 1980, Perruchet, Pacteau 1990, Rosas, Ceric et al. 2010). This work mirrors those experiments by training the networks and evaluating their environments. The aim of the task is to predict the next symbol following on from previous symbols in a sequence of letters.

4.1.1 Symbol Representations

In language, there are techniques to represent non-numeric data to a neural network. Take for instance representing the four characters a, b, c and d in a network. There are three basic techniques to represent them. First, represent them using one signal and assigning numeric values to the characters, such as $a = 0.0, b = 0.3, c = 0.6 \; and \; d = 0.9$. This method will fail, as it is difficult to interpret the data precisely; if the network cannot provide the decision concerning which of the two outputs is correct, it naturally produces an intermediate value. The second way is to represent them by using two binary signals: $a = [0,0], b = [0,1], c = [1,0] and \; d = [1,1]$. This representation suffers from the same problem as the output of the network could be [0.5, 0.5]. The third technique is using localist mapping. Here, every symbol will be linked to a specific cell in the representation and only one cell in the representation can contain a one. Therefore, to represent the four letter the representation will require $a = [1, 0, 0, 0], b = [0, 1, 0, 0], c = [0, 0, 1, 0], d = [0, 0, 0, 1]$. In this situation, four signals are needed. The problem of the two previous cases will not occur here (Tjongkimsang 1992). This representation does not have the problem of the two previous techniques. If the network produces an intermediate value the candidates that it suggests can unambiguously be pointed at; for example, presume that the network outputs $[0.5, 0.5, 0, 0]$ then it was difficult for the network to choose between *a* and *b*. In addition, all the representation patterns are equally similar. Therefore, the network is unable to recognise non-intended similarities from the representations. The disadvantage of this technique is more training time, especially when there are many symbols to be processed (more units, links and weights). When considering large representations and non-intended similarities problems between patterns, a balance can be found by choosing an intermediate representation size, and letting the network find out meaningful representations of that size. Three methods have been demonstrated for allowing networks to build fixed length representations for symbols (Elman 1990, Blank, Meeden et al. 1992). The task that the network learned to perform with the symbols is then coded in the representations built by the network. Therefore, a different network task will lead to different representations.

The input layer consists of seven neurons for all the networks that have been examined in this research. The activation function used for both hidden and output layers was binary sigmoid; the learning rate was 0.15; and the momentum coefficient was 0.75. These values were arrived at as optimal settings after extensive training on the Reber grammar problem, where a range of values of learning rate and momentum were systematically applied. The symbols represented to the input layer are coded to the network using a localist mapping as shown in Table 4.1. The context unit, will have as input, a copy of the activation function of the hidden layer.

Grammatical pattern	Orthogonal vector
B	1 0 0 0 0 0 0
P	0 1 0 0 0 0 0
S	0 0 1 0 0 0 0
T	0 0 0 1 0 0 0
V	0 0 0 0 1 0 0
X	0 0 0 0 0 1 0
E	0 0 0 0 0 0 1

Table 4.1 Orthogonal binary vector representations for input patterns.

The output layer also consists of seven neurons to represent a letter. A sequence of *n* patterns is coded at each time step t as x(t) symbol and expecting the net to predict the next symbol in the sequence x(t+1). Each pattern consists of two input vectors and one target vector. The target vector is a seven-bit vector representing element t+1 of the sequence. The two input vectors are:

- The hidden unit (i.e. context nodes) at time t-1.
- A seven-bit vector representing input at time t of the sequence.

Another representation was used in the trials. The purpose is to make the representation of the letter more effective; there are no zero inputs so each input will have some significance and thus cause a weight change resulting in learning for all weights. Symbols that are not active are represented by 0.2 and those that are active represented by 0.8. The coding of the symbols represented to the networks is demonstrated in Table 4.2.

Grammatical pattern	Orthogonal vector						
B	0.8	0.2	0.2	0.2	0.2	0.2	0.2
P	0.2	0.8	0.2	0.2	0.2	0.2	0.2
S	0.2	0.2	0.8	0.2	0.2	0.2	0.2
T	0.2	0.2	0.2	0.8	0.2	0.2	0.2
V	0.2	0.2	0.2	0.2	0.8	0.2	0.2
X	0.2	0.2	0.2	0.2	0.2	0.8	0.2
E	0.2	0.2	0.2	0.2	0.2	0.2	0.8

Table 4.2 Orthogonal vector representations for input patterns

4.1.2 The Regular Grammar: Simple Reber Grammar

This study uses Reber's small finite-state grammar (Reber, 1976), shown in Figure 4.1. It is a deterministic finite automaton capable of generating some regular language L; an element or sequence s, where $s \in L$, is generated according to Reber grammar's rule (grammatical). Moreover, other invalid sequences can be generated (ungrammatical) that do not follow the Reber rules.

Finite-state grammars consist of nodes connected by labelled arcs. A grammatical sequence is produced by entering the network through the B 'Begin' node and by moving from node to node until the E 'End' node is reached. Every transition from node to node generates the corresponding letter to the label of the arc linking between these two nodes. The elements of the language are illustrated as follows:

$$L = \{B, P, T, S, X, V, E\}$$

Examples of sequences that can be generated by the grammar are 'BTSXSE' 'BPTVPSE' 'BTSXXTTVVE'.

Two occurrences of the same letter may lead to different nodes. This makes it difficult to predict its successor. In order to perform the task sufficiently, the network has to encode the letter context instead of just identifying the current letter.

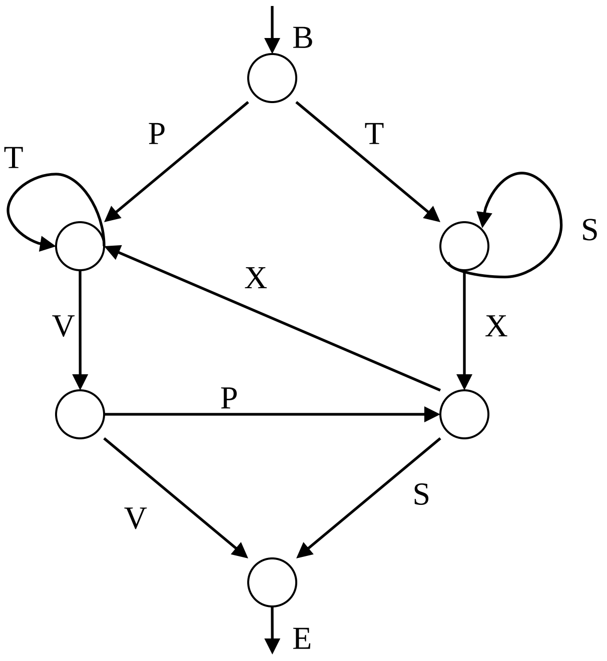


Figure 4.1 The finite-state grammar (FSG) used by Reber

4.1.2.1 Reber Grammar Dataset

Three datasets were randomly generated and examined, the sequences were unbiased sequences, and that is, each transition arc from node to another or itself (self-looping) has the same probability of 0.5. The size of datasets used consisted of 60,000, 100,000 and 1,000,000 randomly generated training sequences with resulting average weighted lengths of 7.9, 7.97, and 7.95, have SD of 3.2, 3.3 and 3.26 respectively. In addition to these, two testing datasets were randomly generated; the first is 33335 grammatical sequences and 200,000 ungrammatical sequences. The grammatical sequences generated from the grammar are shown in Figure 4.1. A sequence starts with the initial symbol 'B' and follows a pattern of two possible symbols selected at random at each node, with a probability of 0.5 for each (symmetrical sequences). Each pattern was then presented sequentially to the networks. The activation of the context units were reset to zero at the beginning of each sequence. After each pattern, the error between the network's prediction and actual successor specified by the sequence was computed and back propagated. All three datasets were generated randomly and the sequence lengths in each ranged from five to 32 patterns.

The weighted mean used for calculating the average length of the strings was as follows. The equation is:

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i} \qquad (4.1)$$

Where, w is the number of times a particular sequence length is repeated and x is the length of a sequence, n is the number of observations (number of sequence lengths). Standard Deviation (SD) has been calculated according to equation 4.2 {the average distance from the mean of the data set to a point}.

$$s = \sqrt{\frac{\sum_{i=1}^{n} w_i \left(x_i - \bar{x}\right)^2}{\sum_{i=1}^{n} w_i}} \qquad (4.2)$$

Where $\bar{x}$= is the weighted mean, x_i= are the observations and w_i = are the weights.

4.1.3. The Context-Free Grammar: Embedded Reber Grammar (ERG)

Embedded Reber Grammar (ERG) (Cleeremans, Servan-Schreiber et al. 1989, Fahlman 1991, Hochreiter, Schmidhuber 1997, Gers 2001) is an extension to simple Reber grammar. Both grammars are in the form *n*L*m*\aL*b*, where *n*, *m*, *a* and *b* are unique strings included in L. Therefore, ERG is much more difficult because the neural network must have to remember the initial sequence for several number of previous time steps; In other words, the last symbol of a valid sequence is determined by the first one and is independent of the sub-sequencing in between. As a result of this, the memory of the past sequence required to predict the ultimate symbol (penultimate) is considerably larger in this case; because the network must observe the initial symbols 'T' or 'P' and must retain this information while processing an embedded sequence or arbitrary length. This is what makes the prediction task much harder.

Consider the problem of number agreement demonstrated by the following two sentences:

The **dog** *that chased the cat* **is** very playful.
The **dogs** *that chased the cat* **are** very playful.

At first glance, information about the head of the sentence does not seem to be relevant for processing the embedded clause itself. Nonetheless, from the perspective of a system that is continuously generating expectations about possible succeeding events, information about the head is relevant within the embedding. For instance, the embedded clauses require different agreement morphemes (chases vs. chase) when the clause is in the present tense, etc. Moreover, even after the same word has been encountered in both cases, expectations about possible successors for that word remain different. There is sufficient empirical evidence to support the claim that human subjects do generate expectations continuously in the course of natural language processing. (Servan-Schreiber, Cleeremans et al. 1989) devised a finite-state grammar, based on the Reber grammar, called the Embedded Reber Grammar, in which the identity of the last letter depends on the identity of the first one.

The ERG used for the neural networks to learn the paths of a transition diagram are depicted in Figure 4.3. A sequence is generated by travelling from the leftmost node to the rightmost. The ERG is contained from two parallel Reber grammar diagrams that raise the number of pathways to make the task more difficult. In the embedded Reber grammar Figure 4.2, the first symbol after B serves as a pointer that uniquely determines the penultimate symbol in a sequence before the last symbol, E. The embedded section between these two symbols comprises letters generated by the Reber grammar. The relevance of the embedded Reber grammar to natural language processing is that such complex embedded clauses occur in natural language and every natural language processing system must have the ability to preserve information about long-distance contingencies in order to ensure that sentences containing embedded sequences are processed correctly (Cleeremans el at. 1989).

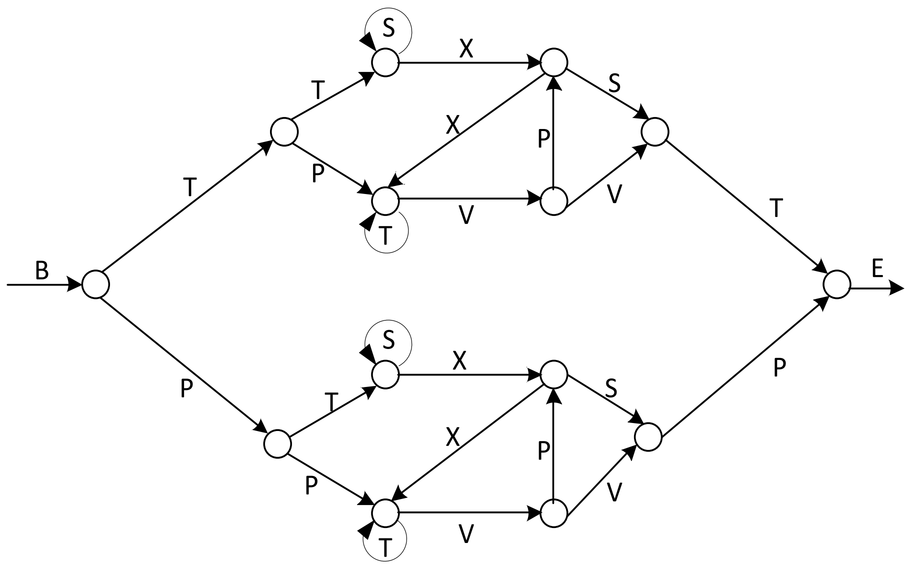


Figure 4.2 A complex finite-state grammar involving embedded sequences.

Two kinds of datasets of ERG generated, biased and unbiased sequences (Symmetrical and Asymmetrical). In symmetrical sequences, the sequences generated randomly as training proceeded with each of the two possible successor symbols at any state being equally likely, probability of 0.5. In addition to this, asymmetrical sequences generated, in this method of generation exaggerates the bias of the training set toward upper part of the embedded, if transition probability is ‘T’, or toward bottom part of the embedded if transitioned to ‘P’. Therefore, a network receives some steady, continuing reinforcement for remembering whether it is in the upper or bottom embedded of the grammar. One of the aims of this research is to investigate which of the training datasets is better to preserve information about the predecessor of the embedded sequences across symmetrical and asymmetrical sequences.

4.1.3. 1 Symmetrical Sequences

This section describes and explains the random datasets generated as unbiased datasets, (Cleeremans, Servan-Schreiber et al. 1989, O'connell 1995) generated 2.4 million letters to present to the network. In this research, 300,000 sequences that are approximately 2.65 million letters randomly generated that is comparable with their work, the maximal length is 27 and the weighted mean is 9.81.and SD 3.03. Equation 4.1 and 4.2 show the formula used to calculate both weighted mean and SD.

Table 4.5 describes the number (frequency) of sequences at each length together with the statistical characteristics of the data set and the total number of patterns. The generation of sequences was illustrated as in the regular grammar: for each time step, the next symbol is generated from a choice of two with equal probability 0.5, except for the last symbol where there is one choice. Therefore, information about the starting symbol is not locally relevant to predicting the successor of any letter in the embedding.

Length	Frequency	Length	Frequency	Length	Frequency	Length	Frequency
6	74815	12	14656	18	1821	24	323
7	55983	13	9665	19	1281	25	116
8	37624	14	6712	20	893	26	24
9	42094	15	4830	21	619	27	1
10	23649	16	3391	22	627		
11	17871	17	2489	23	516		
Number of Sequences			300000	Standard Deviation			3.03
Weighted Mean			9.81	Number of Patterns			2643125

Table 4.3 Characteristics of the 300000-Symmetrical-embedded-Reber

The 300,000 dataset is generated probabilistically, therefore it will include some natural replication of sequences. Another 1,000 sequences test file was generated in the same way but was followed by the removal of naturally replicated sequences in the dataset. The entire sequences are a unique form not replicated as with Cleeremans Servan-Schreiber et al (1989), there is a naturally occurring overlap between the training and test samples of 79.8%. The maximal length for unbiased testing dataset was increased to 36. The weighted mean is 18.52 and the SD 4.4. Table 4.6 illustrates the contents of the dataset.

Length	Frequency	Length	Frequency	Length	Frequency	Length	Frequency
6	4	14	68	22	54	30	5
7	6	15	89	23	29	31	2
8	8	16	100	24	30	32	3
9	14	17	96	25	15	33	1
10	18	18	94	26	9	35	1
11	26	19	82	27	7	36	1
12	38	20	82	28	9		
13	50	21	58	29	1		
Number of Sequences			1000	Standard Deviation			4.4
Weighted Mean			18.52	Number of Patterns			17521

Table 4.4 Characteristics of the 1000 Symmetrical Testing Dataset (Embedded Reber Grammar)

In Figure 4.3 the transition probabilities of the whole arcs were equal and set to be 0.5, giving rise to what are called symmetrical sequences. Figure 4.3 also shows an embedded Reber grammar but the probabilities of a route being selected are biased toward the top arcs for the top embedding, and toward the bottom arcs for the bottom embedding.

4.1.3. 2 Asymmetrical Sequences

The resulting training sequences would contain some indication as to which of the two embedded grammars it belonged to which may help the RNNs to identify the appropriate exit transition.

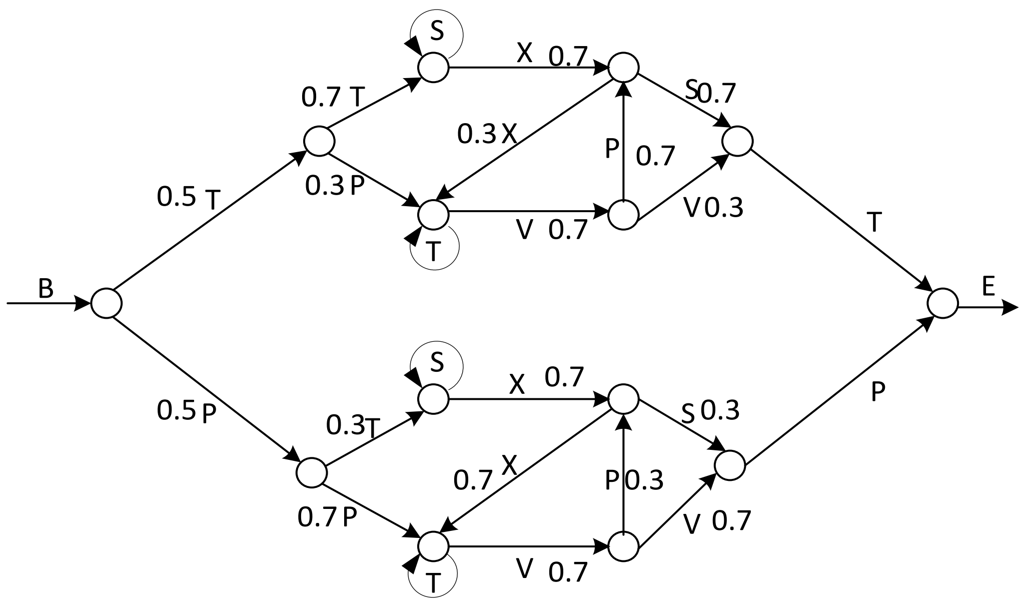


Figure 4.3 Embedded Reber grammar biased symbols (Asymmetrical). The numbers above each arc indicate the transition probabilities in the biased form of the grammar

Therefore, this means that when symbols within an embedded clause relate to the subject (symbols before entry into embedding) then the network has a greater chance of learning to exit the embedding correctly. To illustrate the argument about the embedding more clearly: in natural language we are assuming the following words relate to the subject. For example:

(1) *The cats [when the weather was windy and wet] were hungry.*

In sentence (1), little or no clue is provided within the embedded clause to indicate that the subject is plural. Generating sequences with equiprobable transitions as in sentence 1 are referred to as symmetrical sequence data, and how this is implemented is discussed in detail below.

(2) *The cat [who is lost and has a wet coat] is hungry.*

In sentence (2), clues are provided within the embedded clause to remind the RNN that the subject is singular. Data generated in this way results in asymmetrical sequence data and what will be called an asymmetrical data set. It was therefore decided to generate both symmetrical and asymmetrical data to determine if the asymmetrical data helps the learning of such cross-serial dependencies and to explore this property if it does.

Another 300000 sequences were generated; the sequences are asymmetrical strings where the top sub-grammar was slightly biased towards the top nodes. The probability of the first *T* was 0.7 with 0.3 for the first *P*, 0.7 for the second S with 0.3 for the second *X* and 0.7 for the second *P* with 0.3 for the second *V*. Equally; probabilities in the bottom sub-grammar were biased in the opposite direction. The probability distribution used in the dataset generation is shown in Figure 4.3. In addition in this dataset, there are 21.2% of the sequences not in the training dataset (unique), the percentage of the matchless in unbiased test dataset is 34.8%. The maximal length is 26 with average length 8.78 and SD 2.31. Table 4.5 illustrates the file contents.

Length	Frequency	Length	Frequency	Length	Frequency	Length	Frequency
6	121318	12	8351	18	539	24	58
7	58871	13	5216	19	345	25	22
8	33250	14	3089	20	262	26	10
9	33953	15	2221	21	172		
10	18986	16	1410	22	113		
11	10877	17	839	23	98		
Number of Sequences			300000	Standard Deviation			2.31
Weighted Mean			8.78	Number of Patterns			2334544

Table 4.5 Characteristics of the 300000 Asymmetrical Dataset

Another 1,000 sequence test file was generated in the same asymmetrical way but followed by the removal of repeated sequences in the dataset. The maximum length for testing datasets is 36 and the weighted mean is 18.52 with SD 4.4. Table 4.6 illustrates the contents of the dataset.

Length	Frequency	Length	Frequency	Length	Frequency	Length	Frequency
6	4	12	38	18	116	24	20
7	6	13	50	19	110	25	9
8	8	14	68	20	75	26	5
9	14	15	91	21	53		
10	18	16	107	22	39		
11	26	17	117	23	26		
Number of Sequences			1000	Standard Deviation			3.63
Weighted Mean			17.928	Number of Patterns			16928

Table 4.6 Characteristics of the 1000 Asymmetrical Testing Dataset

4.2. Model Fitting

In this section, techniques used to evaluate and optimise the networks are considered. Six, varied network architectures (Jordan, TDNN, NARX, SRN, MRN and ESN) are compared and their limitations studied. In order to improve the performance of some of the networks, a regularisation method is used for controlling the complexity of the model. It attempts to overcome the over fitting problem by using a flexible model with constraints on the values that model parameters can take, normally through the addition of a penalty term (Zur, Jiang et al. 2009). There is a wide range of techniques for regularising neural network models that have been developed. For instance, Bayesian methods (MacKay 1995), weight elimination (Weigend, Rumelhart et al. 1991), Dropout (Hinton, Srivastava et al. 2012). The regularisation mechanism was used in SRNs, NARX, MRNs and ESN. The reasons behind choosing these networks were their performance on the task compared with the other networks (outcome of the networks), trying to improve the performance and the effect of regularisation on different networks architectures. Also, in this section, a summary of Backpropagation through Time (BPT) is described, together with some learning algorithms and the Taguchi method (Roy 2010). In addition to these, the way in which internal representation of SRN, NARX, MRN and ESN (hidden units) are analysed using principle components analysis is described.

4.2.1. Overview

This section, describes the methods used in the training, testing and understanding of the networks. The size of the input and output layers for all models was seven units (due to the number of symbols that the grammar has) for the next symbol prediction task. A single hidden layer was used in all cases following the networks architectures adopted by the inventors. The hidden layer sizes for the SRN models varied from 3 to 25 hidden units. The reservoir size for the ESN models ranged from 30 to 700 hidden units. The activation functions used here for all the networks except the ESN were *binary sigmoid* for both hidden and output layers. For the ESN the functions were *tanh* and *sigmoid* for reservoir and output layers respectively. The non-linear activation functions allow the networks to solve problems, which are out of reach of linear networks. Therefore, this can introduce a non-linearity system into the network. The range of the random weights for the networks was - -0.3 to 0.3. The context units are initially set to 0.5. The learning rate was 0.15 and the momentum coefficient was 0.75 (excluding for the ESN where they are not applicable). The prediction was considered correct if, for each pattern in a given sequence, the activation for the correct successor symbol was greater than 0.3 or 0.38 used when the dataset uses 'symbol representations orthogonal binary' and 'fraction vectors' respectively. If this criterion was not met, the sequence was considered as rejected for both cases. This will be further explained later in the next chapter.

Each training experiment was repeated 15 times (with the same architecture and training parameters) to explore and account for sensitivity to the initial state determined by the randomly generated starting weights.

Cross validation is a technique used to estimate the prediction error (estimate how well a model has been trained). It is used to try to avoid over-fitting of the training to the data. Over-fitting can arise where there is: a small training set (patterns that in small training set may be spurious and due to noise); noise in the data (the likelihood of spurious patterns that do not reflect actual patterns in the domain increases); and where there are many features in the dataset (each feature provides an opportunity for spurious patterns to show up). In contrast, the datasets used in this study, were without noise.

Cross validation is not therefore used in this study. Since the data applied in this research is the same as the training methodology used by Cleeremans et al 1989 and Bob Cartling 2007.

4.2.2. Model Selection

It is illustrated in much neural network research that adding additional noise to the input data during training can, in some situations, lead to significant improvements in generalisation performance (Bishop 1995, Zur, Jiang et al. 2009, Rifai, Glorot et al. 2011). The noise is used for the regularisation of the network. Various techniques to achieve regularisation of a parametric model, weight decay or output smoothing are used to avoid over-fitting during the training stage of the considered model. According to the Bayesian theory, a number of regularisation techniques correspond to imposing specific prior distribution on model parameters (Rifai, Glorot et al. 2011).

One of the methods of regulation, called noise injection, enhances complex models indirectly by adding noise to the training dataset (Zur, Jiang et al. 2009). According to (Matsuoka 1992) if there is an ambiguous mapping from the input space to the output space it should be smooth. Then the noise injection can enhance the generalisation ability of the learning algorithm. However, if this condition does not accrue, the method might produce a weak and bad generalisation capability. In addition to this, it is used to improve the generalisation performance when the number of input samples is relatively small or heavily contaminated with noise. The purpose in this research is to improve the performance of the networks by adding random noise to the input data during training. Since the best-performing network was, the multi-recurrent network, it was investigated first by adding two types of noise injection: random noise vector onto each input pattern and a random noise neuron injection. Then, these techniques were also used with the SRN and ESN. Figure 4.4 and Figure 4.5 display SRN and MRN using noise injection respectively.

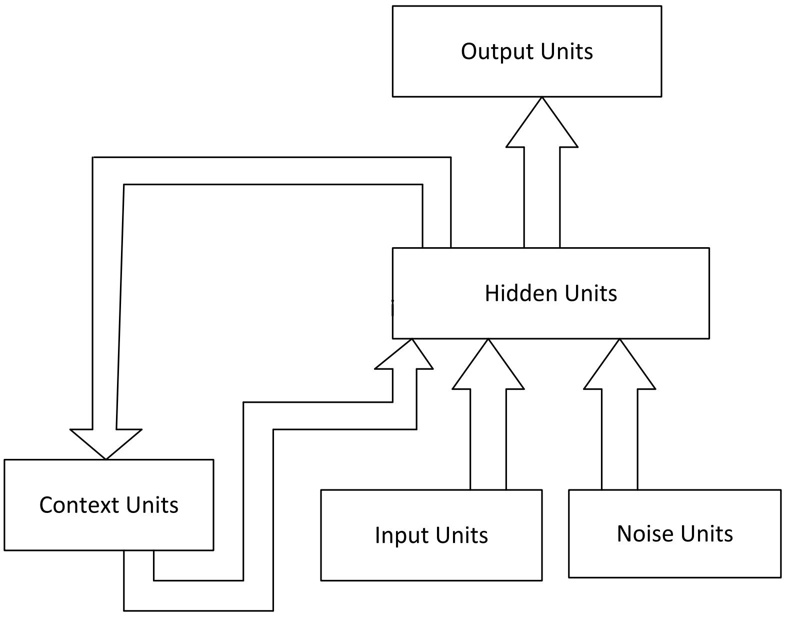


Figure 4.4 SRN with noise-injection units

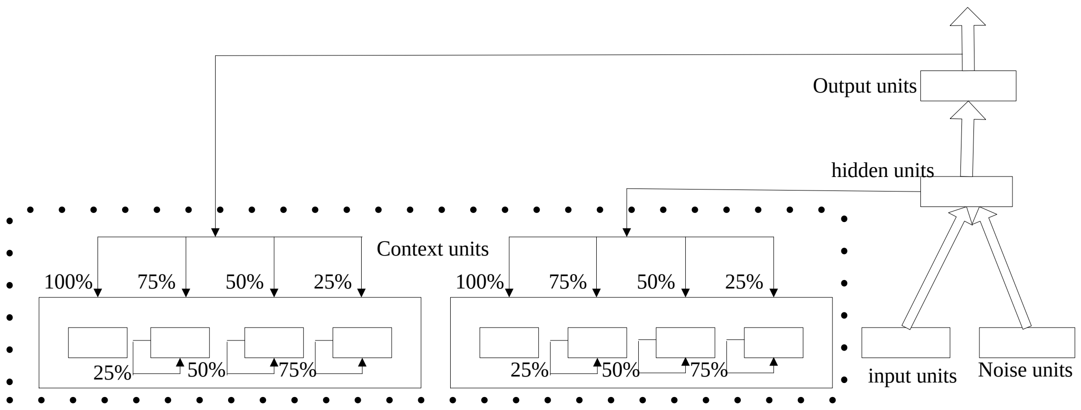


Figure 4.5 MRN with noise-injection units

Two types of noise injection were tried: first, one noise-injection unit was inserted with the input; and second, seven noise-injection units were inserted on the input (one per data input unit). A range of noise levels was used with asymmetrical training sequences since the performance of the networks on asymmetrical sequences is superior to that with symmetrical sequences. These values were in ranges between plus and minus 0.005, 0.01, 0.3, 1, 1.25, 2.5 and 5 for one- noise unit and 0.005, 0.01, 0.03125, 0.25, 0.5, 1, and 1.25 for the seven unit architecture. Real random numbers in these ranges were added to the network as noise, with the input for each symbol entered. Results are illustrated and discussed in the next chapter.

4.2.2.1 Backpropagation through Time (BPTT)

Many learning algorithms can be used with recurrent neural networks. They can be divided into two groups: algorithms working online (Real Time Recurrent Learning, dynamic Backpropagation) and algorithms working offline Backpropagation through time (BPTT). The latter is the most popular method for achieving supervised learning and this is the method used in this work.

The discrete-time RNN learning algorithm entitled BPTT (Rumelhart, McClelland 1985) is described. It is a very powerful procedure. It can be applied to many temporal classification problems; however, it requires considerable computational power to achieve a high level of accuracy (Principe, Kuo et al. 1993, Boné, Crucianu et al. 2002, Smith 2002, Grüning 2007). Figure 4.6 is a schematic of BPTT. At the start, the outputs of the BPTT algorithm are computed for all time steps. Then, the gradient is computed by starting at the last time step and working backwards in time.

BPTT provides a solution to the cycling connections between the nodes in RNNs by stacking identical copies of the RNN and obtaining connections between subsequent copies by redirecting the connections within the network. Whereas the feedforward backpropagation algorithm cannot be directly transferred to RNNs because the error backpropagation pass requires that the connections among the neurons induce no cycle ordering. Unfolding the recurrent network in time is the solution used in the BPTT algorithm; that is stacking identical copies of the recurrent neural network, and redirecting connections among the networks to gain connections between subsequent copies. Therefore, a feedforward network has been obtained, which is adjustable by the backpropagation algorithm.

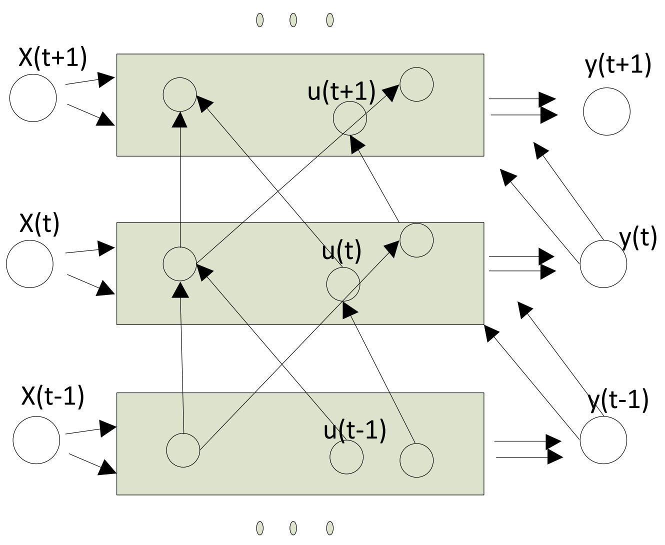


Figure 4.6 Schema of the basic idea of Backpropagation through Time (Jaeger 2002)

The network is trained according to the rule

$$\frac{\partial E}{\partial w_{ij}} = \sum_{t=1}^{T} \delta_i(t) x_j(t-1) x_i(t)(1-x_i(t)) \qquad (4.3)$$

Where

$$\delta_i(t) = -e_i(t) + \sum_j w_{ji} x_j(t+1)(1-x_j(t+1))\delta_j(t+1) \qquad (4.4)$$

Also, $e_i(t)$ is the output error, $x_i(t)$ represent the activations and $\delta_i(t)$ are backpropagated errors. Formulae defined in 4.3 and 4.4 constitute the BPTT algorithm (Prokhorov, Feldkarnp et al. 2002).

The input/ output time series are denoted by:

$x(t) = (x_1(t), \dots, x_n(t))', d(t) = (d_1(t), \dots, d_m(t))'$ t=1… T (4.5)

The BPTT learning algorithm consists of two passes; forward and backward. In the forward pass, the stacked network starting from the first copy to the end of the stack is updated in one training epoch. The input *x(t)* is first read in at each copy *n* and time *t* , then from *x(t)* and *u(t+1)* { and from *y(t-1)*if nonzero w_{ij}^{back} exist} the hidden state *u(n)* is computed. Then, the current copy's output *y(n)* is computed. The minimized error is calculated as follows:

$$E = \sum_{t=1,\dots,T} \|d(t) - y(t)\|^2 = \sum_{t=1,\dots,T} E(t) \qquad (4.6)$$

In the first pass, the outputs of the BPTT algorithm are computed for all time steps. Then, the gradient is calculated by starting at the last time step working backwards in time.

Two types of learning rate were used with BPTT in this research. One was a fixed learning rate and the second a pattern error–sensitive learning rate. The aim of the learning is to minimize the sum of the squared error.

4.2.2.2 Fixed Learning Rate

The learning rate parameter determines how large the weight changes should be. The larger the learning rate, the quicker the adaption of the network as a whole. However, this parameter has a positive or negative effect on network performance. If the learning rate is very large, this damages the generalisation accuracy and slows down training. On the other hand, a very small learning rate requires extra reductions in size and wastes computational resources without any further improvement in generalization accuracy (Wilson, Martinez 2001). There is no fixed rule for how to set the learning rate so as to achieve the best balance between stability and plasticity; the optimal learning rate depends on the particular problem being learned. The fixed learning rate used here is according to the Widrow and Hoff rule and is called Last Mean Squared (LMS) or delta rule, where the weights update according to the following:

$$W^{new} = W^{old} + (\delta * E) \qquad (4.7)$$

4.2.2.3 Pattern Error- Sensitive Learning Rate

A pattern error-sensitive learning rate (Tepper et al. 1995) was used as it was found to aid learning. The principle of this learning rate type is that the learning rate for a given pattern is sensitive to its error within a prescribed range. Therefore, the formula used is as follows.

$$\delta = \frac{\sum_{i=1}^{n} |e_i|}{n} * a + b \qquad (4.8)$$

Where

$e_i = d_i - o_i$

$$f(\delta) = \begin{cases} a, & if\ \delta > a \\ b, & if\ \delta < b \\ \delta, & Otherwise \end{cases}$$

4.2.2.4 Taguchi Method and Analysis of Variance (ANOVA)

Taguchi's theory based on Orthogonal Array Selector (OAS) is usually applied to eliminate a number of experiments that do not have any impact on the process, in order to optimise the quality of the process (Al-Habaibeh, Zorriassatine et al. 2002, Roy 2010). It provides an efficient method on performance variability reduction and is usually used for off-line parametric optimisation control and high performance design (Deng, Fox 2007). The technique is considered here with the aim of reducing the number of experiments needed for optimising the network training parameters by identifying the significant variables affecting the output (Roy 2010).

The alternative is to undertake a full factorial technique to find the contribution of each parameter of the experiment individually (i.e. the independent variables) and computing the dependency of the factor outcomes for each experiment. In Taguchi's method, the dependency is the proportion of contribution found for a parameter by analysing the variance. The dependency of a variable reflects the portion of the total variation detected in an experiment attributed to that factor. The Taguchi methods are based on the statistical analysis of data and offer a simple description of optimisation and analysis of complex systems (Macleod, Dror et al. 1999). The technique used to determine the relative contributions of the factors by comparing their variance is called the analysis of variance (ANOVA).

ANOVA is a statistical method that permits researchers to analyse or dismantle the variations in test outcomes into components that are dependent on the various sources. Therefore, the total experiment variance can be partitioned to different factors and to combinations of different factors. ANOVA uses Taguchi’s method and can be described as a two-stage procedure. Firstly, the total variance of the measured output is computed and possible combinations of factors and the variance due to the individual factors that were studied are computed. Secondly, the variance due to any pair of factors or combination is compared. Consequently, the factor (parameter) that has a more significant effect on the design output can be concluded. To represent the concept, any high dimensional function can be represented as a subset of terms as follows:

$$f(x) = f_0 + \sum_{i=1}^{n} f_i(x_i) + \sum_{i=1}^{n}\sum_{j=i+1}^{n} f_{i,j}(x_i, x_j) \quad + f_{1,2,\ldots\ldots\ldots\ldots\ldots,n}(x) \qquad (4.9)$$

Where n represents the number of inputs, f_0 is a constant and the rest on the right hand side represents functional combinations of input parameters. Then, ANOVA partitions the total variation into its suitable components. The total sum of squares is defined as

$$SS_T = \sum y_i^2 \quad for\ i = 1,2, \ldots n \qquad (4.10)$$

Which can be given as

$$SS_T = SS_m + SS_e \qquad (4.11)$$

Where, $SS_m = nM^2$ and $SS_e = \sum(y_i - M)^2$ are the mean sum of squares and the error sum of squares respectively, and M is the average of the observed data where M is,

$$M = \frac{1}{n}\sum y_i \quad (i = 1,2, \ldots, n)$$

The most distinct disadvantage to the ANOVA procedure is that it’s assumes that the data in the groups are normally distributed. However, moderating the data can overcome this limitation. Another limitation is that it only detects significant difference among cell means, but does not indicate the functional form of the relationship among cell means (Buckless, Frank A 1990).

In this research, the ANOVA method is applied to select the optimal parameters (factors) for the ESN by taking the weight range, connectivity and spectral radius in four different values for each one. There were 64 training settings with these factors, each repeated three times. The results will be in the next chapter. The reason for using this technique with the ESN is that it reduces the time taken for the trainings as this involves a huge dataset that can use an enormous reservoir size. In addition to this, the technique establishes which of the parameters has the highest effect on the performance of the network.

4.2.3. Model Evaluation

The networks were trained to take one symbol at a time and to predict what the next symbol would be. The prediction process forces the network to develop internal representations that encode the relevant grammatical information because the prediction relies on the grammatical structure (Elman 1993). Every network input pattern maps onto a particular point in the hidden unit activation space. Therefore, we learn how these points relate to each other over time, and understand how the network is operating and whether it is analogous to underlying finite state automaton.

Several techniques have been used to deal with the internal representations within trained connectionist systems, to shed more light on what is happening inside the hidden layer, e.g. Hierarchical Cluster Analysis (HCA), Multi-Dimensional Scaling (MDS), Canonical Discriminant Analysis (CDA), Principle Component Analysis (PCA) (Bullinaria 1997). This research will focus on the use of PCA due to its success reported for similar research (Gallagher, Downs 2003, Cartling 2008, Verstraeten 2009).

The internal representations of the networks can play a significant role in solving a problem. With the help of the network structure, learning algorithm etc. the internal representations allow the network to ignore the oppression of a form-based interpretation of the world. The internal representations have similarity structure, which can be a valuable indicator of the meaning, rather than the similarity structure of the bare input (Elman 1993). In this simulation, the networks used various numbers of hidden units to represent a number of different factors, which were related to the task. These need to be able to represent grammar states, the embedded part and grammar symbols. PCA can be used to classify the precise dimensions associated with each factor and was, first introduced by Karl Pearson over a century ago (Pearson 1901). It is a very popular technique and has been used widely in the statistical community, primarily for descriptive but also for inferential purposes (Gallagher, Downs 2003). Moreover, it is a method for identifying patterns in data, and visualising the data, so similarities and differences can be easily seen. It is also used to identify and remove any correlation among problem variables and as an aid to dimensionality reduction.

The main advantage of PCA is its ability to find the patterns in the data and compress it by reducing the number of dimensions, with minimal loss of information (Bullinaria 1997, Smith 2002). $P_{i\alpha}: i = \{1, \ldots, d;\ \alpha = 1, \ldots, n\}$ are the vector components of a set of n points in d dimensional hidden unit activation space and $\langle P_i \rangle$ denotes the mean $P_{i\alpha}$ over all values of α . Therefore,

$$S_{ij} = \sum_{\alpha} (P_{i\alpha} - \langle P_i \rangle)(P_{j\alpha} - \langle P_j \rangle) \qquad (4.12)$$

Where S_{ij} is the standard covariance matrix. It is symmetric and the eigenvectors are given as:

$$\sum_k S_{jk} \Lambda_{ki} = \lambda_i \Lambda_{ji} \qquad (4.13)$$

Which is orthogonal. Λ_{ij} can be used to perform a change of basis, i.e. an axis rotation, as follows:

$$P_{i\alpha}^{\Lambda} = \sum_j \Lambda_{ij}^{-1} P_{j\alpha} \qquad (4.14)$$

The covariance matrix is diagonal and computed as follows

$$S_{il}^{\Lambda} = \sum_j \sum_k \Lambda_{ij}^{-1} S_{ik} \Lambda_{kl} = \lambda_i I_{il} \qquad (4.15)$$

This approach was used to good effect by (Elman 1993) to analyse his sentence processing network and by (Cartling 2008) to visualise the sentence processing.

Chapter 5

5. Experimental Results

The experiments conducted in this chapter will utilise various ANN types to learn a difficult grammar structure in a given corpus. One of the main objectives of this research is to optimise the performance of the SRN and other recurrent networks (Jordan, NARX, TDNN, MRN and ESN) in order to understand the complex grammar represented, namely the Embedded Reber Grammar (ERG). The literature includes limited attempts by Elman (1990), Sharkey et al. (2000). Cleeremans et al. (2008) and others to learn the ERG. These attempts limited the amount of self-looping inside the embedded part of the grammar and they made no comparison between different neural networks architectures. This study, aims to cover these aspects to assess their comparative abilities and limitations and to understand the different representations formed by these different architectures.

5.1 Learning the Regular Grammar

The SRN was given the task of predicting the next symbol in a sequence from a corpus generated using the rules of the regular grammar, the simple Reber Grammar (see 4.1.2). The prediction is considered accurate if, for each pattern in a given sequence, the activation of the output corresponding to one of the two correct successor symbols was greater than 0.3.This is known as a 'soft acceptance criterion'. If this criterion was not met, the sequence was considered rejected. This criterion is the same as that used by (Cleeremans et al. 1989) in which the same learning task was set. The choice of a threshold of 0.3 for the binary representation of symbols (or 0.38 for the 0.2 and 0.8 representation of symbols) is not completely arbitrary. The activation of the output units is related to the frequency with which a particular symbol appears as the successor of a given sequence. The probability of any particular legal successor symbol occurring in the training set is 0.5. Nevertheless, since a momentum term is used in the backpropagation learning procedure, the correct activation of the output units following training was sometimes under 0.5 occasionally as low as 0.3.

The experiments in this section also explore the question raised by (Noel Sharkey, and Stuart Jackson 2000), about whether particular initial starting weights are required to enable the SRN to successfully learn the important aspects of the language, or is the learning success independent of the starting state of the model. To this end, for each SRN configuration, training was repeated 10 times each with different starting states. Table 5.1. Shows the results of repeating 10 experiments with different starting states for networks with various hidden layer sizes and training data set sizes.

The columns represent the 10 trials for each configuration, while the rows list the experiments, which increase in the size of the training set and the hidden nodes. The variability in the results across a row indicates the sensitivity of the network performance to the starting state (weights). The success of training (in terms of sequences learnt) in each scenario rises as the size of the hidden layer increases.

In general, it is preferable when working with neural networks to select an architecture that learns the problem with the minimum number of hidden nodes. This should minimise memorisation and maximise generalisation (Lin & Meador, 1992). The sensitivity of the starting conditions is high for the 60K dataset with fluctuating results for the most of the hidden unit sizes used and also, in the 1M dataset. The SRN configurations trained with 100,000 sequences and 5 hidden units, fully learnt the training set. Therefore, the higher performance of the trained networks was selected for testing.

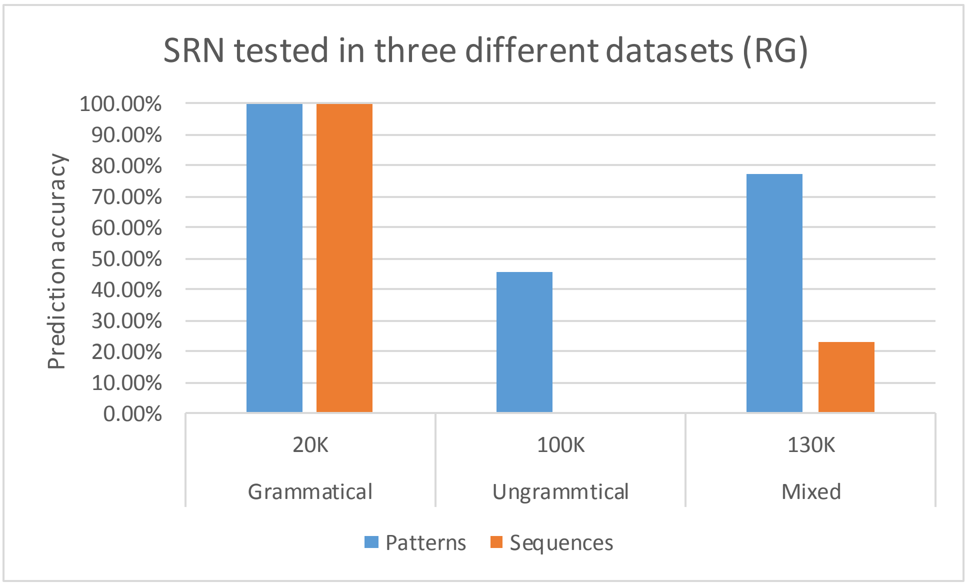
In order to test whether the network would produce similarly good predictions after each pattern, a slight change was made to the testing files so that comparisons could be made with the test files used by (James, McClelland 1988) and ungrammatical sequences were added to the test files. Three different datasets were generated to test the SRNs: i) 20K “20000”grammatical sequences, ii) 130K sequences containing 23.0% grammatical sequences and finally iii), a third file containing 100K sequences, all of which are ungrammatical. The maximum sequence length of these datasets is 50. Approximately 60% of the grammatical dataset sequences were not in the training data.

Graph 5.1 shows the result of the experiments using the test datasets. The blue bars represent the individual symbols that were correctly predicted by the network, whereas the red bars represent the correctly predicted sequences (strings of symbols). The network recognised all of the grammatical sequences in the grammatical test file and in the second test file (mixed sequences). The network was able to differentiate between the grammatical and ungrammatical sequences by predicting the entire grammatical ones and rejecting all of the ungrammatical structure, for the ungrammatical test dataset the network rejected all of the sequences.

As an extra measure and to evaluate the SRN's ability to learn the RG with any sequence length, extremely long sequences (beyond that found in the training set) were therefore presented to the network, and the sequences had a minimum length of 50 symbols and a maximum of 100 symbols. An example of such a sequence is:

'BTXXTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTVPXTTTTTTTT TTTTTTTTTVPXVPXVPXVPXTTTTTTTTTTTTTTTTTTTTTTTVPSE'

The network correctly predicted all of the sequences as 100% indicating the network had learnt to be the perfect model for the underlying regular grammar.



Graph 5.1 Elman's SRN: Testing prediction accuracy on Reber Grammar

Experiments		1	2	3	4	5	6	7	8	9	10
60K Data set	3 H	28.66%	30.90%	48.66%	52.00%	34.51%	51.54%	37.58%	84.35%	35.88%	30.90%
	5 H	84.31%	37.58%	79.27%	75.97%	76.75%	71.76%	70.86%	58.54%	71.82%	90.58%
	7 H	99.18%	87.51%	91.56%	81.71%	100.00%	100.00%	99.36%	79.57%	96.89%	62.88%
	10 H	91.45%	83.11%	93.81%	75.28%	100.00%	100.00%	100.00%	93.62%	100.00%	100.00%
	15 H	100.00%	100.00%	100.00%	93.62%	100.00%	99.20%	93.62%	100.00%	100.00%	93.62%
100K Data set	3 H	60.71%	57.22%	37.32%	65.74%	79.69%	37.32%	51.00%	75.54%	49.91%	37.32%
	5 H	100.00%	96.81%	82.50%	100.00%	37.32%	68.58%	80.98%	93.61%	74.76%	87.36%
	7 H	100.00%	100.00%	87.36%	100.00%	100.00%	100.00%	100.00%	100.00%	73.36%	100.00%
	10 H	99.72%	100.00%	100.00%	84.23%	100.00%	100.00%	100.00%	98.89%	100.00%	93.78%
	15 H	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
1M Data set	3 H	75.05%	51.80%	100.00%	50.78%	50.03%	37.52%	50.03%	76.79%	100.00%	50.03%
	5 H	70.09%	45.56%	84.34%	74.98%	100.00%	87.48%	93.74%	73.35%	59.37%	98.44%
	7 H	100.00%	96.47%	95.32%	100.00%	37.52%	81.22%	42.88%	89.44%	90.61%	95.31%
	10 H	99.92%	98.44%	100.00%	96.15%	87.48%	91.55%	92.54%	100.00%	77.67%	90.61%
	15 H	95.36%	97.90%	99.25%	99.61%	100.00%	99.01%	100.00%	100.00%	100.00%	90.77%

K- Thousands; M – millions; H – number of hidden nodes

Table 5.1 Elman's SRN: The percentage of the accuracy of the sequences across the whole data set on the training performance for the Reber Grammar

5.2 Learning the Context Free Grammar

To further understand the limitation and learning ability of recurrent networks, the Jordan 1986, TDNN, NARX , MRNs and ESN network architectures have, together with the SRN architecture, been investigated and examined with a more complex version of the Reber grammar, known as the Embedded Reber Grammar (ERG), see 4.1.3. A noise injection technique is used in the models to produce a higher accuracy of prediction and evaluate whether this method facilitates leaning in such connectionist models.

5.2.1 SRN Using Constant Learning Rate

The network that has been used is depicted in Figure 3.4. The task for the network is to predict the next symbol in ERG described in section 4.1.3. Each of the embedded parts (upper and lower) of the grammar shown in figure 4.2 is a transition graph similar to the RG. It is a complex task, since the network has to remember the initial starting symbols, T or P, which precede entry into one of the embedded grammars within the overall grammar. This initial 'entry' symbol has to be remembered, as the same symbol is used to correctly exit the embedded grammar (known henceforth as the penultimate symbol). More specifically, T represents entry to/exit from the upper embedded grammar, whilst P represents entry to/exit from the lower embedded grammar. As each of the embedded grammars has some form of recurrency, the number of intervening symbols between the entry/exit symbols can be arbitrarily long and complex. For example, the embedded grammars may include several T's and P's, but only the initial T or P is contingent to the penultimate symbol.

The pattern-error sensitive learning rate (as used by Tepper et al 2002) will be evaluated against a constant learning rate to determine whether improvements reported by the mentioned study also apply to this grammar-learning task. If this proves to be the case, then the pattern-error sensitive learning rate will be used for all other RNN models being considered (that use gradient descent learning). In addition, Hard and Soft Acceptance criteria were evaluated with these models. The optimal results obtained using similar criteria carried out by Cleeremans el al (1989) with the simple grammar have the advantage of using the more stringent criterion. Note that the networks have been trained on symmetrical and asymmetrical sequences as described in sections 4.1.3.1 and 4.1.3.2.

A number of experiments were conducted used the constant learning rate, as with the previous work with the simple Reber grammar. Biased and unbiased datasets were trained and tested to investigate the impact of the learning algorithm on them.

5.2.1.1 Embedded Reber Grammar (Symmetrical Sequences)

In the first experiments, 20,000, 60,000, 100,000, and 1000,000, sequences were generated randomly to generate training sets with weighted mean sequence lengths of, 9.4, 9.5, 8.4, and 9, and standard deviations of, 2.8, 2.9, 1.1, 2 respectively. Networks with different numbers of hidden units (5, 10, 15 and 20) were trained with the various datasets. However, the last dataset (containing 1,000,000 sequences) was trained with just 5 and 10 hidden units due to memory limitations of the computers used. The aim of using these datasets was to investigate how many hidden units are needed to learn the ERG. The input layer and output layer again consisted of seven units, one for each of the seven symbols of the grammar. The symbols were coded to the network as shown in Table 4.1. The activation function used for both hidden and output layers was the binary sigmoid function. The learning rate was 0.15 and the momentum coefficient was 0.75. The learning rate type was constant. The 'Soft Acceptance Criterion', described earlier and used for the RG, were used. If this criterion was not met, the sequence was considered rejected. As before, each model configuration was trained five times, each with different initial starting weights. Table 5.2 shows the results.

(K, M, and H represent thousand, million sequences and Hidden nodes, successively

Experiments		1	2	3	4	5	Max
20K	5 H	78.71%	0.00%	0.00%	25.11%	0.00%	78.71%
	10 H	81.90%	84.34%	65.80%	44.99%	47.54%	84.34%
	15 H	97.76%	98.10%	96.18%	98.31%	98.21%	98.31%
	20 H	70.03%	100.00%	100.00%	99.55%	94.64%	100.00%
60K	5 H	41.78%	11.74%	20.03%	35.49%	0.00%	41.78%
	10 H	98.24%	95.36%	93.23%	96.07%	99.99%	99.99%
	15 H	98.24%	96.07%	70.02%	97.65%	22.61%	98.24%
	20 H	99.99%	99.24%	99.09%	99.24%	99.99%	99.99%
100K	5 H	27.20%	52.78%	0.00%	67.84%	28.50%	67.84%
	10 H	62.20%	92.90%	79.32%	89.10%	75.09%	92.90%
	15 H	91.57%	100.00%	90.24%	91.03%	96.95%	100.00%
	20 H	89.81%	100.00%	92.79%	100.00%	100.00%	100.00%
1M	5 H	20.05%	88.24%	71.18%	50.03%	16.76%	88.24%
	10 H	99.99%	55.36%	25.05%	87.58%	99.99%	99.99%

Table 5.2 Elman's SRN: Percentage accuracy of the entire dataset training for the embedded Reber grammar (soft acceptance criterion)

Table 5.2 shows the percentage accuracy of all the training networks, each network repeated five times. The training is conducted in one iteration. According to the table, the size of the dataset does not affect the performance significantly, whereas the number of hidden units does. The table also shows that a network with more hidden units is more capable of learning the grammar than when it has fewer hidden units. However, the datasets that had a large corpus (large dataset) required a lot of computational time. Although similar best case results were obtained regardless of the size of the dataset, there was significant fluctuation in some cases between the performances of individual networks having different starting conditions (1M, 5H and 100K, 5H). It can be concluded that there is a certain dataset size that can be trained with 15 hidden units to achieve better performance. Therefore, the rest of these experiments in this section have 15 hidden neurons.

Experiment 1: Using Soft Acceptance Criterion

The objective here is to investigate the performance of the SRN by seeing how well it predicted the successor for general sequences from the grammar. As before, the soft acceptance criterion was applied. In addition, the purpose is to ascertain whether the same method applied in the Reber grammar can be successful in the embedded Reber grammar structure, and if not why not. Henceforward, 300K symmetrical sequences will be used which is described as 4.1.3.1. The same criterion that was used in the previous experiments is used here.

The following terminologies are used in the tables:

Embed = percentage of sequences in which each embedded symbol was correctly predicted.

Penult = percentage of sequences in which the penultimate symbol was correctly predicted.

Alternative Penult = percentage from the wrong predictions where it is one of the two possible penultimate symbols (i.e. the network has determined that this is the end of the sequence but not correctly remembered which half the grammar it is in - upper or lower).

Wrong Penult= percentage from the wrong predictions where it is not in the two possible penultimate symbols (i.e. the network has lost track of the grammar within the embedded part so it is not predicting the end of the sequence).

Network	Whole Sequence % Correct	Embed%	Penult%	Incorrect	
				Alternative Penult%	Wrong Penult%
1	88.362	92.629	97.027	2.973	0
2	70.752	70.752	100	0	0
3	67.228	67.228	100	0	0
4	96.454	96.454	100	0	0
5	73.876	87.742	85.205	13.982	0.812

Table 5.3 SRN training results on ERG (soft acceptance criterion)

Table 5.3 shows the results of training for each network. The SRN was unable to predict the embedded part of the grammar perfectly. However, the overall result of the sequences is acceptable, especially network four. In order to determine the networks' abilities to generalise to new sequences, each network was tested by the presentation of 1,000 input sequences. 21.2% of the sequences in the testing set are not in training set (they are unique from the training). The test file was generated randomly but duplicated strings have been removed (unique sequences).

Network	Whole Sequence % Correct	Embed%	Penult%	Incorrect	
				Alternative Penult%	Wrong Penult%
1	77.1	82.1	99.9	0.1	0
2	23.2	23.2	100	0	0
3	70.5	70.5	100	0	0
4	99.9	99.9	100	0	0
5	57.1	72.7	80.7	18	1.3

Table 5.4 SRN test results using soft acceptance criterion for the ERG

Table 5.4 illustrates the test results. It is obvious that the results gained from the test file are sharply lower than the training results. Nonetheless, the result for the final symbol before the end (penultimate) was predicted better compared with the whole of the embedded part in both training and testing. This is surprising as prediction of the penultimate symbol represents the significant challenge of capturing long-term dependency. Network 4 has an excellent performance with the soft acceptance criterion used for the whole sequence to accept the symbols. However, in this grammar, the first symbol after B serves as an indicator, which uniquely determines the penultimate symbol with the embedded section between these two symbols being Reber grammar strings. Following on from this, there is only one correct prediction for the last symbol in each string. Therefore, the method for accepting the penultimate symbol as the correct successor was changed.

Investigating medium acceptance criterion, using Luce ratio (Luce 1963) to mimic to work done by McClelland, 1988, the experiments conducted and the results are in appendix A. The results are poor and are comparable with the models demonstrated.

Experiment 2: Using Hard Acceptance Criterion

The aim of this experiment was to further explore how to find the best acceptance criterion for determining whether the network is correctly predicting the next symbol. The criterion of taking the highest activation function in the output layer as indicating the prediction for the next symbol was tried for evaluating the network performance.

The importance of the penultimate symbol is to investigate how it copes with long-term dependency. Table 5.5 and 5.6 show the testing of two representations of the symbols and the hard acceptance criterion. The training results are in appendix B. These results indicate that the network results are biased to one path of the arc, labelled 'T' in the ERG. This means that the network does not allow the correct selection between the two paths of the grammar. Moreover, the overall result with the second method, which used 0.2 and 0.8 representing the letters, was the best (52.2%) for the whole sequences and 100% for the penultimate where in binary representation the results are fluctuating. In addition to this, the medium acceptance criteria also had poor results: 5.3% for the entire sequences with 89.2% in the embedded. That makes the soft acceptance criteria superior over both methods.

Network	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
1	37.7	82.5	51.2	0	51.2	48.8	0
2	13.2	23.2	51.2	0	51.2	48.8	0
3	37.1	71.7	51.2	0	51.2	48.8	0
4	51.2	99.9	51.2	0	51.2	48.8	0
5	33.2	74.3	50.9	0	50.9	48.6	0
6	45.1	87.8	51.2	0	51.2	48.8	0

Table 5.5 SRN test results for ERG using hard acceptance criterion and binary symbol representations

Using the highest activation of the output units for choosing the successor using binary symbol representation, provides flawless performance for the embedded part.

Network	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
1	52.2	100	52.2	0	52.2	47.8	0
2	52.2	100	52.2	0	52.2	47.8	0
3	52.2	100	52.2	0	52.2	47.8	0
4	52.2	100	52.2	0	52.2	47.8	0
5	49.8	95.2	52.2	0	52.2	47.8	0

Table 5.6 SRN test results for ERG using hard acceptance criteria and non-binary symbol representations

Nevertheless, there were errors in the predictions when the penultimate has been analysed as illustrated in the Table 5.5 and 5.6. Both methods show that the networks are unable to predict which penultimate symbol is correct. They consistently only predict "T". In addition, the training and testing files have been analysed in details as illustrated in appendix B, and this shows that each path has approximately the same frequency of occurrence. Biased sequences have therefore been generated to explore their effect on the network performance.

5.2.1.2 Embedded Reber Grammar (Asymmetrical Sequences)

These experiments concern a dataset of 300,000 asymmetrical sequences randomly generated, in which the two sub-grammars (upper and lower) were slightly biased. Asymmetrical data encodes information about initial state transitions within the embedded grammars. This is presented in chapter four in detail. The upper sub-grammar is biased towards the top nodes. (Probability of the first T was 0.7 vs. 0.3 for the first P; 0.7 for the second S vs. 0.3 for the second X, 0.7 for the second P vs. 0.3 for the second V). Conversely, the probabilities in the bottom sub-grammar were biased in the opposite direction. The average length of a sequence was 8.7 with a standard deviation of 2.3. All of the following experiments apply the soft acceptance criterion, as it is the most reasonable for assessing prediction performance. Testing is done on the networks with the 1,000 symmetrical sequence dataset as previously used.

Experiment 1: SRN results for asymmetrical training and symmetrical test

In this experiment, the network was trained on asymmetrical sequences and tested on symmetrical ones. This will determine whether the asymmetrical sequences have an effect on the network performance, especially in the prediction of the penultimate symbol. The results are shown in Table 5.7.

By using asymmetrical sequences for training, the results show some improvement in recognising the penultimate symbol but at the expense of a very poor performance in the embedded part of the sequence. The network predicts one of the penultimate symbols but the performance regarding this is generally no better than chance. The networks produce poor results for the whole sequences. Similar results were obtained when the representation of symbols with 0.2 and 0.8 are used, and were not significant and some of them are in appendix B. With the use of asymmetrical training data, the SRN has been proven still incapable of learning the long-term dependency as had been shown by Cleeremans et al. (1989) and O'Connell (1995). The next step is to investigate whether other recurrent networks are capable of addressing the shortcoming of the SRN and solve the complexity of the embedded Reber.

Network	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
1	23.8	51.6	50.3	30.1	20.2	49.7	0
2	7.7	15.2	49.9	30.1	19.8	49.8	0.3
3	1.3	2.6	49.9	29.7	20.2	50.1	0
4	1.5	2.7	49.8	29.8	20	49.9	0.3
5	9.8	20.3	49.3	29.5	19.8	50.1	0.6
6	1.9	3.4	50.1	29.9	20.2	49.9	0
7	6.1	12.3	48.2	28.9	19.3	48.8	3
8	1.4	2.4	50.5	20.6	29.9	49.5	0
9	1.9	3.6	50	29.5	20.5	49.8	0.2
10	2.4	4.4	50.1	15.9	34.2	49.9	0

Table 5.7 SRN results of the symmetrical test file with 10 asymmetrical training networks and binary symbol representations using soft acceptance criterion

5.2.2 SRN Using Pattern Error-Sensitive

The results that have been obtained from the SRNs for the embedded grammar were not identical or even similar to the results of Cleeremans and Dienes 2008 and Sharkey 1992. Therefore, more investigation of the problem has been carried out and it was the learning rate type that affects the performance of the networks. Hence, a number of experiments were repeated to study the effect and the performance of the networks. The learning rate used in the next experiment is pattern error-sensitive as described in chapter four. Fifteen networks were trained and five were chosen to assess performance with the testing dataset. Moreover, to limit the research, binary symbol representations were used and the soft acceptance criterion utilised to accept the successful symbol.

5.2.2.1 SRN Results for Asymmetrical Training Tested with Symmetrical and Asymmetrical Sequences

In the experiments, asymmetrical sequences were used in training and two types of 1000 datasets, symmetrical and asymmetrical, were used for testing. The aim was to compare the performance of the network after changing the learning rate type; 15 hidden units are used as with the previous work.

Network	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
1	53.5	89	60.6	28.6	32	39.4	0
2	60.4	99.2	60.9	36.2	24.7	39.1	0
3	60.7	100	60.7	36.1	24.6	39.3	0
4	61.1	100	61.1	36	25.1	38.9	0
5	54.3	89.8	60.6	35.8	24.8	39.4	0

Table 5.8 SRN results of five asymmetrical training nets tested on asymmetrical sequences (pattern error-sensitive learning rate)

Table 5.8 shows the results of the highest five nets from 15 nets trained on asymmetrical sequences. The maximum result is 61.1% of the whole sequences predicted correctly and the entire embedded part of the grammar predicted correctly. In addition, Table 5.9 shows that, in network four the maximum performance is 50.3% on the symmetrical test file that is superior to the asymmetrical training tested on asymmetrical test dataset. Strong evidence of both tables illustrate that using a pattern error – sensitive learning rate improved the network performance. However, trained unbiased sequences needed to be examined on tested dataset for both biased and unbiased sequences.

Network	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
1	45	88.4	51.1	22.9	28.2	48.9	0
2	49.2	98.1	50.1	28.8	21.3	49.9	0
3	49.9	100	49.9	28.7	21.2	50.1	0
4	50.3	100	50.3	28.6	21.7	49.7	0
5	39.1	78.2	49.8	28.4	21.4	50.2	0

Table 5.9 SRN results of five asymmetrical training nets tested on symmetrical sequences (pattern error-sensitive learning rate)

5.2.2.2 SRN Results for Symmetrical Training, Tested with Symmetrical and Asymmetrical Sequences

Here, the objective is to see the performance of the Elman network when symmetrical sequences were used as training, and testing was with asymmetrical and symmetrical sequences to compare with asymmetrical training. Table 5.10 shows the five nets tested on asymmetrical sequences and Table 5.11 shows on symmetrical sequences. The SRN has been proven still with different methods, approximately incapable of learning the long-term dependency that had been shown by Cleeremans et al. (1989) and O'Connell (1995). However, the network is able to encode information about long-distance contingencies as long as the information about critical past actions is related to each time step for creating predictions about possible alternatives.

Other experiments have been conducted, which is training SRN with noise injection. The network poorly predicted the results, which is not comparable with the results obtained from the SRN without noise injection. The next step is to investigate other recurrent networks to see if they are capable of addressing the shortcomings of the SRN and solve the complexity of the embedded Reber grammar. The following section examines the performance of the Jordan network (1986) applied to this grammar-learning problem.

Network	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
1	54.6	100	54.6	22.5	32.1	45.4	0
2	49.7	100	49.7	14.2	35.5	50.3	0
3	52.7	100	52.7	52.3	0.4	47.3	0
4	47.8	100	47.8	1.4	46.4	52.2	0
5	56.4	100	56.4	47.3	9.1	43.6	0

Table 5.10 SRN results of five testing nets tested on asymmetrical sequences

Network	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
1	50.6	100	50.6	19.6	31	49.4	0
2	50	100	50	13.4	36.6	50	0
3	49.3	100	49.3	48.9	0.4	50.7	0
4	51.2	100	51.2	1.4	49.8	48.8	0
5	54.4	100	54.4	44	10.4	45.6	0

Table 5.11 SRN results of five testing nets tested on symmetrical sequences

5.2.3 Jordan Network

The architecture used here is the same as illustrated in Figure 3.1. It is a three-layer network that has context units connected to the hidden units and the output connected to the context units. Moreover, different types of feedback connections are used; using the activation function of the output, the error from the output and target as feedback.

Jordan results for asymmetrical training, tested with symmetrical and asymmetrical sequences

The same datasets as in the previous section for training and testing were used here. As in the previous work, the network was trained with the asymmetrical sequences; 15 hidden units were used and the feedback was the activation of the output, i.e. standard Jordan. As with the last section, the aim is to compare Jordan and Elman architectures. In this section, the learning rate is always pattern error-sensitive.

Table 5.12 and Table 5.13 illustrate the results tested with asymmetrical and symmetrical sequences respectfully. The results here show enhancement compared with the results obtained from the previous work when a constant learning rate was used; the results are still lower than the results gained from the Elman network.

Network	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
1	28.5	59.6	43.2	18.6	24.6	21.6	35.2
2	9.6	22.2	33	8.7	24.3	22.8	44.2
3	11.5	24.2	50.3	25.4	24.9	33.1	16.6
4	17.6	44.7	41.5	17.1	24.4	24.7	33.8
5	35.1	66.6	46.9	22.3	24.6	31.5	21.6

Table 5.12 Jordan test results for asymmetrical training and asymmetrical test (binary input representations)

Network	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
1	19.2	54.3	33	11.8	21.2	27.5	39.5
2	8.2	21.9	28.1	7.2	20.9	25.3	46.6
3	9.8	22.2	42	20.4	21.6	40	18
4	14	47.4	31	10	21	27.6	41.4
5	25.7	61.5	37	16.6	40.4	37.2	25.8

Table 5.13 Jordan test results for asymmetrical training and symmetrical test (binary input representations)

Jordan results for symmetrical training tested with symmetrical and asymmetrical sequences

The results gained from Table 5.14 and Table 5.15 are the results of testing asymmetrical and symmetrical sequences datasets after obtaining the results from training symmetrical data. The average performance of the network when trained on asymmetrical test sequences is 31.18% while on symmetrical is 28.3% thus showing asymmetrical test as superior. In addition to this, contrary to expectations, these results shows that the Jordan network using symmetrical training has better performance than with asymmetrical training.

Network	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
1	24.9	55.5	41.4	29.7	11.7	47.7	10.9
2	27.1	49.6	54.8	29.9	24.9	45.2	0
3	28.5	54.2	51.1	34	17.1	48.5	0.4
4	43.6	77.3	55.1	26.6	28.5	42.7	2.2
5	31.8	88.6	41.6	20.4	21.2	42.5	15.9

Table 5.14 Jordan results of five testing nets tested in asymmetrical sequences (binary input representations)

Network	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
1	27.1	56.1	45.4	31.3	14.1	48.9	5.7
2	23.7	47.2	51.1	25.2	25.9	48.9	0
3	23.4	47.5	50.3	34.9	15.4	49.5	0.2
4	34.1	69.6	47.9	25.6	22.3	50	2.1
5	33.2	91.5	45.2	22.5	22.7	41.8	13

Table 5.15 Jordan results of five testing nets tested in symmetrical sequences (binary input representations)

5.2.2.3 Summary

These experiments have investigated the performance of the both SRN and Jordan networks with two learning methods: constant learning rate and pattern error-sensitive learning rate. One of the more significant finding to emerge from these experiments is that the pattern error-sensitive learning rate results in better performance compared with the constant learning rate. To limit this study, binary input representation of the input will be used, since there were not significant differences in the results between it and the fractions representation. It also maintains consistency with the same work performed by (James, McClelland 1988, O'connell 1995, McQueen et al. 2005). These findings do not support strong recommendations to take symmetrical or asymmetrical training to set input representation for the networks.

Network	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
SRN	61.1	100	61.1	36	25.1	38.9	0
Jordan	35.1	66.6	46.9	22.3	24.6	31.5	21.6

Table 5.16 Comparing results of SRN and Jordan network (best network performance for asymmetrical training and test sets)

The Jordan network performance (with standard feedback of the output activation to the input) after training with symmetrical data using the pattern error-sensitive learning rate and using binary input representations, was the best achieved for Jordan-based architectures. Table 5.16 shows that the performance of the Jordan architecture is inferior to the Elman architecture for learning this long-term dependency problem. One of the questions that need to be asked about the Jordan network, however, is whether the feedback from the output needs to be graduated by shifting the output instead of feeding the output directly to the input. This means training the time delay neural network to explore its performance.

5.2.4 TDNN

5.2.4.1 TDNN results for asymmetrical training tested with symmetrical and asymmetrical sequences

Time-Delayed Neural Networks (TDNN) were introduced in chapter 3. They represent another alternative architecture to compare and contrast with the previous recurrent architectures. In particular, they allow for the assessment of the importance and format of historical information; the network has direct access to the recent outputs as the network has eight delay boxes from the output i.e. prior output symbols. Training with different numbers of delay boxes was carried out to arrive at this figure, which gave the best results.

Network	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
1	57.6	90	65.5	29.1	36.5	34.5	0
2	56.1	89.2	61.8	24	37.8	38.2	0
3	62.3	93.5	66.3	29.3	37	33.7	0
4	66.6	100	66.6	40.9	25.7	33.4	0
5	57	81.6	69.3	34.7	34.6	30.5	0.2

Table 5.17 TDNN results of five nets trained with asymmetrical sequences and tested on asymmetrical sequences

Both

Table 5.17 and Table 5.18 show the results using the asymmetrical training file and asymmetrical and symmetrical testing files. The results show a slight difference for the asymmetrical compared with the symmetrical test data.

Network	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
1	48.6	86.7	57.2	23.9	33.3	42.8	0
2	47	81.3	55.8	20.6	35.2	44.2	0
3	53.3	90.5	58.4	24.7	33.7	41.6	0
4	56.6	100	56.6	32.9	23.7	43.4	0
5	46.4	74.6	60.5	27.7	32.8	39.4	0.1

Table 5.18 TDNN results of five nets trained with asymmetrical sequences and tested on symmetrical sequences

5.2.4.2 TDNN results for symmetrical training, tested with symmetrical and asymmetrical sequences

The results below are for the same network trained with symmetrical data. The results in Table 5.19 and Table 5.20 show the outcomes when the network is trained on symmetrical results and then tested on asymmetrical and symmetrical datasets. Here it is notable that the network recognises the embedded part of the sequences. Yet the network was less successful at identifying the long dependency part (penultimate symbol).

Network	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
1	53.5	100	53.5	29.7	23.8	46.5	0
2	53.3	100	53.3	16.7	36.6	46.7	0
3	48.3	100	48.3	7.1	41.2	51.7	0
4	53	100	53	30.7	22.3	47	0
5	50.8	100	50.8	17	33.8	49.8	0

Table 5.19 TDNN results of five nets trained with symmetrical sequences and tested on asymmetrical sequences

Network	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
1	48.9	100	48.9	26.7	22.2	51.1	0
2	50.4	100	50.4	11.9	38.5	49.6	0
3	51.5	100	51.5	8.5	43	48.5	0
4	50.3	100	50.3	28.2	22.1	49.7	0
5	51.5	100	51.5	15.5	36	48.5	0

Table 5.20 TDNN results of five nets trained with symmetrical sequences and tested on symmetrical sequences

However, the network architecture struggled to learn the penultimate symbol with both training approaches; the network still failed to recognise all the sequences perfectly. Table 5.21 illustrates the drops in the network prediction when the length of sequences increases. Moreover, it shows that the embedded predictions are right and about half the penultimate predictions are wrong and this percentage is fairly constant and independent of the sequence length.

In Table 5.21:

Correct: number of correctly predict sequences. Wrong: number of unpredicted sequences. Embed: number of the embedded part of the grammar predicted.

Penult: number of sequences that their penultimate predicted correctly.

Length	Network 1 symmetrical test file			
	Correct	Wrong	Correct Embed	Correct Penult
6	2	2	4	2
7	3	3	6	3
8	4	4	8	4
9	7	7	14	7
10	9	9	18	9
11	13	13	26	13
12	20	18	38	20
13	25	25	50	25

Table 5.21 Part of a test file showing the prediction according to the sequence length

5.2.5 NARX

The network used here is illustrated in Figure 3.3. The difference between TDNN and NARX networks is that there are delay boxes representing historical input information on the input to the NARX network (as well as the delay boxes with historical output information as for the TDNN). A number of possibilities were again tried to gain the perfect architecture for this problem. Four boxes were used for the input part and eight boxes used for the feedback from the output. The aim was to investigate the enhancement of the network when there is a history about the input as well as the output.

5.2.5.1 NARX results for asymmetrical training, tested with symmetrical and asymmetrical sequences

The objective is to investigate the network when asymmetrical sequences are presented to the network for training.

The results show in Table 5.22 and Table 5.23 that there is a slight difference between testing on asymmetrical and symmetrical sequences when asymmetrical training has been used, and that the memory boxes from the input did not drastically improve the network performance. However, these results are improved when compared with the results obtained by TDNN.

Network	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
1	70.9	95.7	74	39	35	26	0
2	65.2	91.2	70.5	40.4	30.1	29.5	0
3	69.7	100	69.7	40.1	29.6	30.3	0
4	69.9	95.2	73.8	40.5	33.3	26.2	0
5	65.4	96	67.9	34.3	33.6	32.1	0

Table 5.22 NARX results of five nets trained with asymmetrical sequences and tested on asymmetrical sequences

Network	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
1	60.7	92.5	65.1	32.5	32.6	34.9	0
2	57	93	60.6	33	27.6	39.4	0
3	60.8	100	60.8	30.5	30.3	39.2	0
4	58.9	91	64.8	30.9	33.9	35.2	0
5	56.5	90.3	61.5	30.6	30.9	38.5	0

Table 5.23 NARX results of five nets trained with asymmetrical sequences and tested on symmetrical sequences

5.2.5.2 NARX results for symmetrical training tested with symmetrical and asymmetrical sequences

This experiment was conducted to illustrate when the training is based on symmetrical sequences. The results mirror those from the TDNN with the symmetrical training giving better performance on the embedded part of the sequences but worse performance overall compared with the asymmetrical training. This is because the asymmetrical training seems to aid the learning of the long-term dependency required to predict the penultimate symbol.

Network	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
1	53.3	100	53.3	8.8	44.5	46.7	0
2	53.9	100	53.9	9.6	44.3	46.1	0
3	52.8	100	52.8	5.8	47	47.2	0
4	53.9	100	53.9	10.6	43.3	46.1	0
5	51.7	100	51.7	4	47.7	48.3	0

Table 5.24 NARX results of five nets trained with symmetrical sequences and tested on asymmetrical sequences

Network	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
1	54.8	100	54.8	7.6	47.2	45.2	0
2	54.1	100	54.1	7.2	46.9	45.9	0
3	55.5	100	55.5	6.1	49.4	44.5	0
4	54.4	100	54.4	7.1	47.3	45.6	0
5	55.1	100	55.1	4	51.1	44.9	0

Table 5.25 NARX results of five nets trained with symmetrical sequences and tested on symmetrical sequences

Training dataset	Test dataset	TDNN	NARX
Asym	Asym	66.6%	70.9%
	Sym	56.6%	60.8%
Sym	Asym	53.5%	53.9%
	Sym	51.5%	55.5%

Table 5.26 The percentage of correct predictions for different training and test datasets in terms of bias and non-bias sequences for TDNN and NARX networks

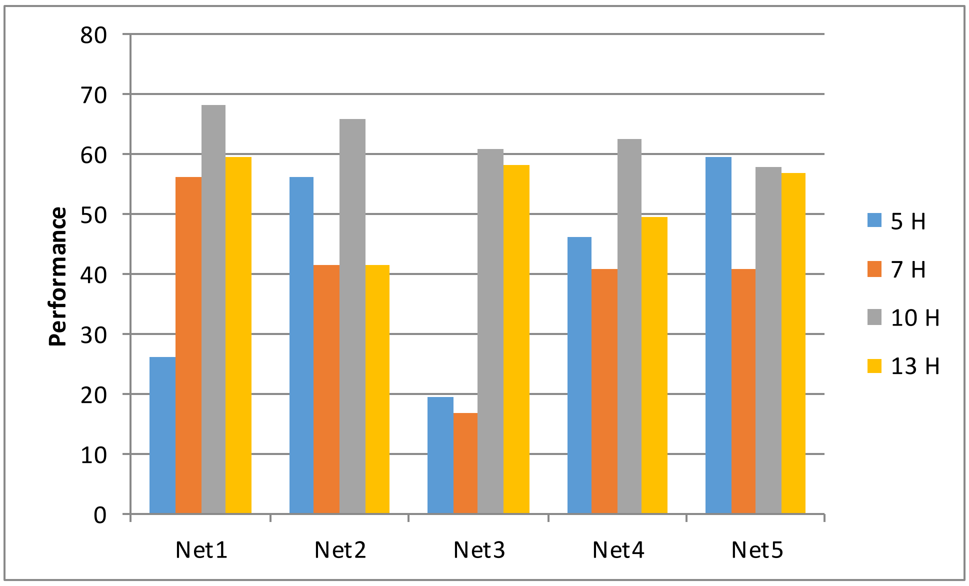
The results shown in Table 5.26 is a comparison between the results obtained from TDNN and NARX show the superiority of NARX over TDNN and furthermore over SRN and Jordan networks. This means that the input delay helped the network slightly to converge to the stable distribution of the grammar. In addition to this work, the NARX was trained with noise injection and the results were poor when compared to the NARX trained without using noise injection.

5.2.6 MRNs

The network used here is shown in Figure 3.5. The critical difference between this network and the previous networks is that there is feedback from both output and hidden units to the input and this is feedback is graduated and includes self-recurrency as described in chapter three. Since the network is different in architecture from the prior networks, networks with different numbers of hidden units were trained to choose the optimal number. All the experiments here use asymmetrical sequences for training the networks. According to the previous results these give superior performance over the symmetrical sequences. Fifteen networks were trained and these are the best five ones with 5, 7, 10 and 13 hidden units and four memory boxes ("banks"). The momentum is 0.75, the initial learning rate is 0.3 and the activation function for both hidden and output units is sigmoid.

Graph 5.2 illustrates the performance of the network when it is trained with different numbers of hidden units. The graph clarifies that 10 hidden units has the highest success compared with the others; the results are in appendix B. However, the results for the embedded sequences are on average still as low as the previous networks (Elman 1990 and Jordan 1986), whereas, NARX has 100% for the embedded part. This shows that there is a trade-off between learning the embedded section and the penultimate symbol. Nevertheless, the MRN needs to be trained with different graduated boxes to optimise this aspect, which may enhance the performance of the network. Also, the performance of the networks were worse this may since the learning rate was a bit high; therefore, the next experiments learning rate is 0.15.

To explore more about the memory boxes of the MRN, the number of the hidden units is set to 10 since the performance with 10 hidden units in the previous experiments has the highest correct prediction for the whole sequence predicted and for the embedded part.

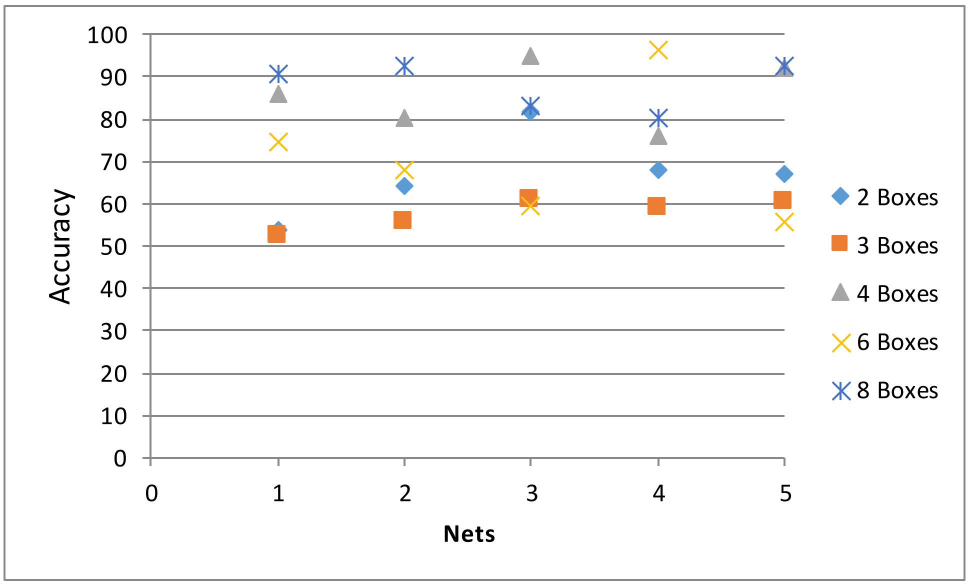


Graph 5.2 Different initial start with different hidden units using MRN network with four memory banks (H: hidden units)

Training with various numbers of memory boxes (banks) was carried out to optimise this parameter: i.e. with 2, 3, 4, 6 and 8. The number of memory banks (ω) relates directly to the degree of granularity at which past and current information is integrated and stored (Binner, et al. 2010). The connection strength of the recurrent link (V_j) from either an output value or a hidden unit activation value to the context unit is $V_j = \frac{1}{\omega} j$ where j=1, 2,…, ω. Lastly, the connection strength of the self-recurrent link Z_i for the context unit C_i is $C_i = \frac{1}{\omega} i$ where i=1, 2,…, ω. Hence, the effect of the memory on the performance of the network can be detected. The training file was of asymmetrical sequences.

Experiment 1: MRN with 2, 3, 4, 6 and 8 banks

Fifteen nets were trained and the best five selected. Table 5.27 shows these results. Graph 5.3 shows that using four memory boxes in the MRN network is the optimal choice. Since the average of the whole sequences predicted correctly for the five networks trained is 85.88% and also the consistency of the result. In addition, (Binner, Tino et al. 2010) state that going beyond this number of boxes does not lead to enhanced performance.



Graph 5.3 Different numbers of memory boxes trained in the MRN

However, the six memory boxes architecture also gives respectable results, as illustrated in the graph. In the next experiments, four boxes were selected for the MRN architecture. Overall comparison between the MRN and the other architectures is considered in section 5.3.

Number of Boxes	Whole Sequence % Correct	Embed %	Penult %	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
2	54	95.7	58.3	50.6	7.7	41.7	0
	64.4	86	73.6	49.7	23.9	26.4	0
	81.6	98.1	83.5	50.9	32.6	16.5	0
	67.9	90.5	77.4	40.5	36.9	22.6	0
	66.9	100	66.9	43	23.9	33.1	0
3	52.4	97.8	55.6	44.7	10.9	44.4	0
	55.5	93.3	60.8	36.1	24.7	39.2	0
	60.7	100	60.7	36	24.7	39.3	0
	59.2	98.2	60.5	35.8	24.7	39.5	0
	60.5	100	60.5	35.8	24.7	39.5	0
4	86.1	100	86.1	49.1	37	13.9	0
	80.2	85.1	94.2	46.5	47.7	5.7	0.1
	95.1	97.9	95.1	51.8	43.3	4.9	0
	76.1	100	76.1	48.6	27.5	23.9	0
	91.9	100	91.9	44.2	47.7	8.1	0
6	74.5	100	74.5	41.3	33.2	25.5	0
	68.1	96.4	69.2	29.3	39.9	30.8	0
	59.6	100	59.6	52.2	7.4	40.4	0
	96.2	100	96.2	50.4	45.8	3.8	0
	55.6	97.2	57.7	52.3	5.4	42.3	0
8	90.6	98.5	91.8	50.5	41.3	8.2	0
	92.6	100	92.6	52.3	40.3	7.4	0
	83.2	93.4	89.4	51.9	37.5	10.3	0
	80.3	98.6	80.6	51.9	28.7	19.4	0
	92.7	100	92.7	46.4	46.3	7.3	0

Table 5.27 Results of different numbers of memory boxes trained with Asymmetrical and tested with Asymmetrical sequences in MRN

Experiment 2: Limitations of the MRN

In these experiments, six sets of sequences of increasing length were used to test the trained MRN to explore the limits of its ability to generalise beyond the training examples and ascertain when its predictions start to fail. The maximum length of training sequence was 26. The tests here extend to sequences of length 120; each file has five sequences. Table 5.28 shows the results. Since the number of sequences of each datasets is five, this demonstrates that the network failed to recognise the sequences when the sequences' lengths reached 120 and the network starts to decline when the sequence length is 50. Whereas, the results when the sequences length is 40, it is approximately the same performance as on the standard test set.

Sequence Length	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
40	80	100	80	40	40	20	0
50	60	100	60	40	20	40	0
60	40	100	40	20	20	60	0
70	40	60	40	20	20	60	0
90	20	20	40	40	0	60	0
120	0	0	60	0	60	40	0

Table 5.28 Result of Testing the MRN (with 4 memory banks) for sequences longer than in the training set (5 tested at each length)

5.2.5.1 MRN with Noise Injection

Two types of injection were conducted: one node and a unit of seven nodes.

Each noise node was fed with a random real number between minus and plus the value chosen. Then, the network was trained 10 times and then the five best performances were selected and tested with asymmetrical sequences. From Table 5.29 it is noticeable that adding noise injection improves the performance of the network. The average results obtained when the noise range was one is a remarkable 90.96%, comparing with most of the ranges, and when the network was tested without noise it was 80.62%.

Using noise node injection

Noise range ±	5	2.5	1.25	1	0.3	0.01	0.005	without noise
Asym Testing	92.5%	64%	92.1%	95.7%	78.7%	72.6%	91.7%	79.5%
	71.8%	98.7%	90.2%	83.7%	78.1%	86.1%	88%	86.7%
	90.9%	72.8%	87.8%	91.1%	91.5%	89.3%	99.2%	65.1%
	79.8%	88.8%	85.4%	96.5%	84.5%	84.3%	69.2%	88.5%
	73.7%	81.3%	93.4%	87.8%	69.1%	87%	78.9%	83.3%
Average	81.74	81.12	89.78	90.96	80.38	83.86	85.4	80.62
SD	8.57	12.09	2.89	4.81	7.43	5.86	10.41	8.35

Table 5.29 MRN: The accuracy of asymmetrical training; tested with asymmetrical data (one noise node)

The average of the correct sequences is 90.96%, nearly 10% above the average performance with the network tested without noise. Although, 99.2% have been obtained in the random noise 0.005 the average of the whole networks tested with the test dataset was 85.4%. Moreover, the standard deviation was 10.41. Therefore, instability of the results is shown in the column of the results that led us to focus on the stability of the results. Increasing the number of noise nodes may provide strength to the network.

Using noise unit injection

In these experiments, seven nodes were injected with the inputs to form a noise unit. After the network was trained 10 times, the best five trained networks were selected. The networks were tested with asymmetrical and symmetrical test datasets. Table 5.30 shows an MRN with a noise unit injected with the input. Many different random noise units were used. On the asymmetrical test dataset, the unit with 0.01 as a random noise range achieved the higher performance comparing with the other values. 93.7% average correct sequences and 2.58 standard division and more than 13% of using the network without noise. On the other hand, the results that have been gained from testing symmetrical sequences shows a drop in the performance of the network. 86.78% is the average success of the network and nearly 3% drop when testing the network without noise.

Range ±	1.25	1	0.5	0.25	0.03125	0.01	0.005	without noise
Asym Testing	98.9%	57.9%	80.1%	76.6%	75.3%	95.4%	93%	79.5%
	84.9%	80.7%	89.4%	70.7%	85.7%	97.7%	19.7%	86.9%
	74.4%	60.4%	90.4%	64.1%	78.4%	90.6%	35.6%	65.1%
	90.3%	74.5%	96.4%	57.7%	79.8%	91.6%	67.8%	88.5%
	94.9%	99.7%	73.9%	95%	87.7%	93.2%	89%	83.3%
Average	88.68	74.64	86.04	72.82	81.38	93.7	61.02	80.66
STD	8.53	15.15	8.00	12.77	4.62	2.58	29.00	8.38
Symm Testing	96.7%	41.3%	72.7%	63.9%	73.3%	89.7%	87.1%	86.1%
	74.4%	71.3%	77.1%	65%	77.4%	93.8%	18.1%	95.1%
	59.2%	53%	84.9%	58%	72.9%	79.7%	25.5%	76.1%
	83.5%	69.9%	91.5%	49%	68.8%	84.7%	59.6%	93%
	88.4%	99.8%	72.6%	84.4%	82.6%	86%	84.1%	91.9%
Average	80.44	67.06	79.76	64.06	75.00	86.78	54.88	88.44
STD	12.84	19.80	7.38	11.65	4.67	4.75	28.74	6.85

Table 5.30 MRN: The accuracy of asymmetrical training; tested with asymmetrical and symmetrical data (one noise unit – seven nodes)

These results have enhanced the network performance which led to further investigation with the network trained with symmetrical sequences and with noise injection for completeness. The range plus or minus 0.01 was chosen as the best noise setting from the previous experiments.

Table 5.31 shows the result of tests on asymmetrical and symmetrical sequence of networks trained on the symmetrical sequences. The results show no improvement in both asymmetrical and symmetrical sequences. The network acts slightly better when tested without noise.

Range ±	0.01	without noise
Asym Testing	48.1%	65.9%
	53%	51.7%
	53.3%	54.2%
	51.3%	49.2%
	52.4%	50%
Average	51.62	54.2
STD	1.89	6.10
Symm Testing	51.6%	69.8%
	49%	51.7%
	49.9%	54.2%
	50.9%	49.2%
	49%	50%
Average	50.08	54.98
STD	1.03	7.61

Table 5.31 MRN: The accuracy of symmetrical training, tested with asymmetrical and symmetrical data (one noise unit)

5.2.7 Conclusion

To conclude, asymmetrical training is superior to symmetrical training, and it also gives the best results if the test data is also asymmetrical. It can be concluded that asymmetrical sequences provide clues within the embedded clauses to help the long-term dependency to be learned.

The limitation of MRN using unit noise injection

The same test datasets used previously are used in this test to explore when the network begins to fail to predict the correct sequences using the best MRN network so far, trained with noise injection. Table 5.32 shows the result of these test datasets with different lengths and have five sequences in each sequence length.

Sequence Length	Whole Sequence%	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
28-40	80	80	100	40	60	0	0
50	80	80	100	40	60	0	0
60	80	80	100	40	60	0	0
70	60	80	80	20	60	20	0
90	60	80	60	0	60	40	0
120	60	80	60	0	60	40	0
150	60	80	60	0	60	40	0
180	60	80	60	0	60	40	0
210	80	100	80	0	80	20	0
240	40	40	80	60	20	20	0
270	20	20	80	60	20	20	0
300	0	0	40	40	0	60	0

Table 5.32 Testing different length of the datasets using MRN with unit noise injection

It shows that the network failed to predict the sequences when their length is above 140 and starts to decline when 70 sequences in length are tested. When the network was trained without noise, results show that the network failed to recognise the sequences of more than 120 and the performance of the network starts to drop when the sequence length reaches 50. This shows that the ability of the network to generalise to unfamiliar sequences beyond the length of the training sequences was enhanced by the addition of noise to the training data.

5.2.8 ESNs

The networks that have been trained in this investigation are shown in Figure 3.7. (a) and (b). They are the standard ESNs by (Jaeger 2002) that have feedback connections from output to the reservoir and also the ESN with jumping connections used by (Tong, Bickett et al. 2007).

5.2.6.1 Standard ESN (with Feedback from Output to Reservoir)

The results from experiments so far clearly show that training with asymmetrical sequences gives better results than training with symmetrical sequences. Therefore, asymmetrical sequences were used to train the ESNs to investigate their performance. Two training variations were used in this architecture: the first was ST with different degrees of settling time; and the second one was without settling time. Altogether, binary symbols were used to represent the input to the network. Various reservoir sizes and different types of feedback were used. However, binary representation showed worse results since the ESP (Echo-state property) is violated for zero input; with a larger input amplitude, and above unity spectral radius may lead to obtaining the ESP (Jaeger 2001, Buehner, Young 2006). Table 5.33 shows some of the results using 30 nodes in the reservoir units and three possible parameters used for feeding back to the reservoir: the output activation itself; the target minus the output (i.e. the error); and the target, i.e. the 'perfect' output. The results demonstrate fluctuations in the performance of the ESN for each experiment. (Tong, Bickett et al. 2007) stated that these are possibly due to the feedback connections, as at any point in time the previous time step's output units are just noise to the current step's input. Therefore, the connections from output to the reservoir have been omitted and trained connections from input to output have been added to the network as illustrated in the next section.

Network Number	Overall Training Performance with Three Different Parameters Fed Back to the Reservoir		
	Output activation	(Target – Output)	Target
1	77.5	35.44	77.89
2	49.29	77.15	78.02
3	77.28	28.68	76.79
4	19.52	78.1	43.21
5	68.57	29.38	78

Table 5.33 Percentage of the ESN performance using different types of feedback

5.2.6.2 ESN with Jumping Connections

The second ESN architecture evaluated has jumping connections. With this architecture, two kinds of datasets were used for training: ordered sequences, these sequences are put in the ascending length order in the file. The second dataset is the biased randomised sequences that were used in the previous experiments. The purpose of adding extra datasets was to explore the effect of training with ordered sequences on the performance of the network. Jaeger 2010 studied the effect of sequence length on the performance of the network and he found that there is a maximal sequence length, which the ESN could stably reproduce, and is a case for investigation.

The results Table 5.34 show improvement in the performance when ordered sequences were used. In these experiments, various datasets have been generated to examine training with random and ordered sequences. The datasets had sequences of lengths: 6 to 12; 13 to 20; and 19 to 25 respectively. They were each then presented either in random order or in ascending order of length. The network parameters were as follows: connectivity 0.85 and spectral radius 0.95. Table 5.34 shows also the average of the 10 networks and it can be indicated from the results that the ESN performance with ordered training sequences is better than with random sequences. On the basis of these findings, ordered sequences are used for training in the next trials.

In addition to these trials, experiments to ascertain optimal values for other training parameters have been conducted. The parameters in question are: the connectivity; the weight range; and the spectral radius. The Taguchi method (Roy 2010) was used to select the optimal parameters as described in chapter four. Each training configuration has been repeated three times. Thus, each configuration has 64 trials repeated three times leading to 192 networks being tested with a reservoir size of 150 nodes (training with a number of reservoir sizes indicated that this produced the best performance; the dataset used has 150,000 ordered sequences). Table 5.35 shows an excerpt of the performance results. The parameters tried are in Table 5.36.

Network Number	Minimal and maximal sequences length and type of the dataset					
	6_12 Random	6_12 Order	13_20 Random	13_20 Order	19_25 Random	19_25 Order
1	56	56	76	60	68	80
2	76	52	72	68	72	80
3	56	80	68	72	80	80
4	88	80	60	72	72	80
5	80	60	68	72	76	72
6	68	88	56	64	76	80
7	68	68	56	68	64	80
8	80	80	64	68	80	80
9	44	84	76	72	80	80
10	68	72	72	68	80	68
Average	68.4	72	66.8	68.4	74.8	78

Table 5.34 Percentage of prediction accuracy using ESN with jumping connection without settling time

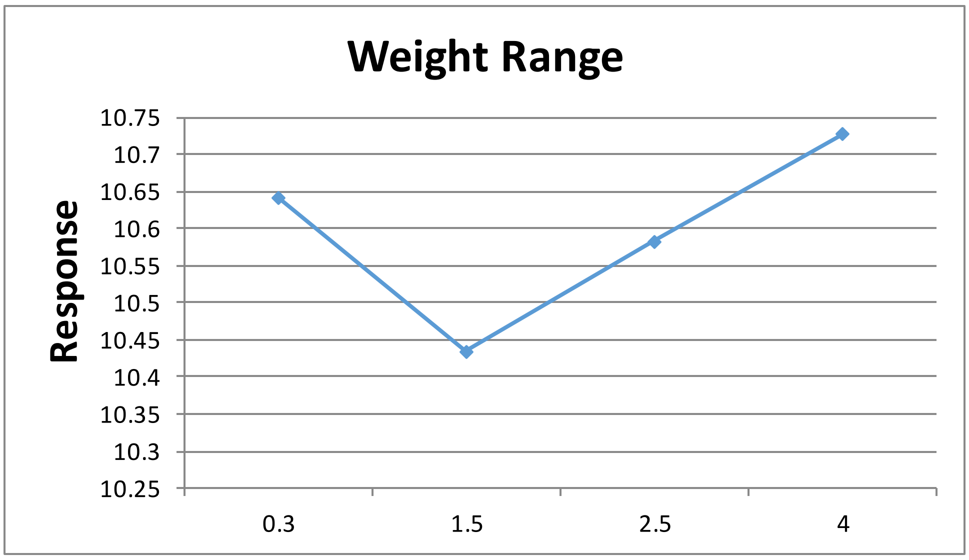
Whole Sequence % Correct		Spectral Radius	Connectivity	Weight Range
1	82.39	0.5	0.5	0.3
2	79.96	0.5	0.5	0.3
3	81.14	0.5	0.5	0.3
4	82.39	0.5	0.5	1.5
5	82.39	0.5	0.5	1.5
6	82.94	0.5	0.5	1.5
7	82.39	0.5	0.5	2.5
8	82.39	0.5	0.5	2.5
9	82.94	0.5	0.5	2.5
10	81.13	0.5	0.5	4
11	82.39	0.5	0.5	4
12	81.13	0.5	0.5	4

Table 5.35 Number of the performance of the ESN that applied on the Taguchi method

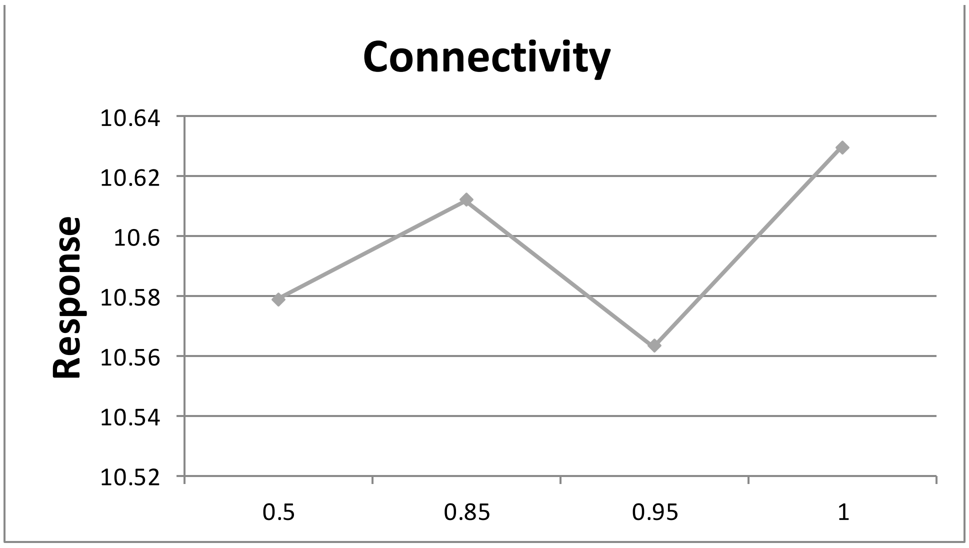
No	spectral radius	connectivity	weight range
1	0.5	0.5	0.3
2	0.75	0.85	1.5
3	1.5	0.95	2.5
4	2	1	4

Table 5.36 the values of the parameters tried

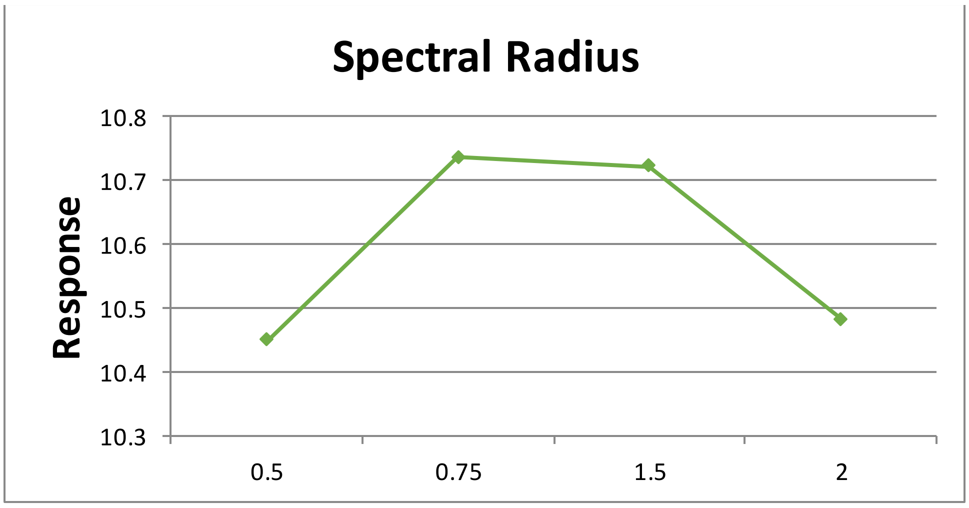
According to the Taguchi method, an analysis of variance (ANOVA) has been applied to these results (this method was described in chapter four). Graph 5.4, It can be seen from the graph that a weight range with 0.3 has more response to the network's performance than the other weight range values chosen. Values over unity were tried here however the value 4 can be ignored since the result is varies and it is over the range of the network output. Graph 5.5 demonstrates the response of performance to the connectivity values. It shows that connectivity with 0.85 is around a peak of responsiveness. Connectivity of 1 is discounted as this implies full connectivity and so would potentially undermine the aspects of the network performance that are predicated on random characteristics. Graph 5.6 determines the effect of spectral radius on the performance of the networks. It can be seen from the graph that the spectral radius value of 0.75 gives rise to the best performance. This is comparable to the results that (Venayagamoorthy, Shishir 2009) gained from their work, which gave a spectral radius value of 0.8. The most striking result to emerge from applying ANOVA is that the average percentage of contribution of spectral radius, connectivity and weight range were 9.71%, 6.48% and 6.05% respectfully. That is, the spectral radius has more effect on the network performance than the connectivity and weight range. In addition, the influence of connectivity on the performance of the network is slightly higher than the weight range.



Graph 5.4 The effect of the chosen weight range on the performane of the network



Graph 5.5 The effect of the connectivity on the performane of the network



Graph 5.6 the effect of the spectral radius on the performane of the network

5.2.6.3 The performance of the ESN

Number of experiments conducted Table 5.37 shows some results from training the ESN with different size of reservoir have been trained. The best performance of the network when network has been tested was 49.9% for unbiased sequences and 60.7% for biased sequences.

Exp	reservoir	correctly predicted	connectivity	Spectral Radius	weight Range
1	343	80.63	0.85	0.75	0.3
2	350	82.51	0.85	0.5	2.5
3	150	84.84	0.85	0.75	2.5
4	150	91.08	0.85	0.75	2.5

Table 5.37 trained ESN with different size of reservoir

5.2.6.3 The Limitation of ESN

Further experiments were carried out; the same long datasets used in MRN are used with ESN to test the network's limitations regarding generalising to long sequences. Table 5.38 depicts the results from the test files. There was fluctuation in the performance of the network over these tested datasets. This will be discussed when investigating the internal representation of the network in the next chapter.

Test set Number	Maximal Length	% Correctly Predicted
1	40	40
2	50	40
3	60	40
4	70	40
5	90	40
6	120	40
7	150	40
8	180	40
9	210	60
10	241	60
11	271	60
12	300	40

Table 5.38 Testing different length of the dataset tested on ESN, the minimal length for dataset 40 is 27 then 40 for the dataset 50 etc.

5.3 Summary of Results and Discussion

In this study, the aim is to attempt to understand better, the abilities of different recurrent architectures to learn to represent and use contextual information when presented with structured sequences of input. Most of the value of the parameters selected here are according to the research that has been done in this field such as in Elman 1990, Cleeremans 1989, Jaeger 2002, Tong 2007 and Cartling 2008. Moreover, these values have been examined in this research.

The results illustrated in Table 5.39 brings together the results from all of the networks investigated when trained with symmetrical and asymmetrical sequences and also tested with symmetrical and asymmetrical datasets. The ESN is not included in the table since it was only trained with asymmetrical data following on from the conclusion that this gave better results.

The table shows the best five trained networks trained for each architecture (Tables such as Table 5.10 and Table 5.11 also give the single best networks). *W* shows the percentage of whole sequences correct in the dataset. From Table 5.39it can be seen that the MRN was superior to all the other networks in performing long-term prediction. For instance, in testing asymmetrical sequences after symmetrical sequence training, the MRN performance reaching 65.9% correctly predicted, whereas, the next best result is the outcome from the SRN at 56.4%, which is nearly 10%, lower than the MRN. Moreover, the MRN acquired 88.5% on the asymmetrical sequences test dataset after training with asymmetrical sequences. This compares with 70.9% acquired from the NARX.

The MRN trained with asymmetrical data was clearly able to detect the long term dependency, achieving a success rate of over 95.1% in five of the symmetrical test trials and over 88.5% in five of the asymmetrical test trials. The other networks on the other hand, generally failed to learn the long-term dependency, largely having only a slightly higher than random chance of predicting the penultimate symbol. Of these, the asymmetrically trained NARX got the highest performance. Another obvious observation is that the results acquired from asymmetrical training are better than that obtained from symmetrical training in all five networks. This indicates that the forced biased sequences do help the networks to learn long-term prediction provided a pattern error- sensitive learning rate is used.

To evaluate the results further, the best performing asymmetrically trained networks tested in both biased and unbiased sequences are shown in Table 5.40. The results for the ESN with asymmetrical training are 91.7% training and 49.9% and 60.7% respectively for unbiased and biased sequences Similarly, Table 5.41 The best performance of symmetrically trained networks tested with both dataset types shows the symmetrical training tested with both datasets.. The information from the tables is also shown graphically in Graph 5.7 and Graph 5.7.

		SRN			Jordan			Jordan Shifting Output (TDNN)			NARX			MRN		
		W	*E*	*P*	*W*	*E*	*P*	*W*	*E*	*P*	*W*	*E*	*P*	*W*	*E*	*P*
Symmetrical Training	Training	62.87	100	62.87	48.58	97.35	49.69	50.23	100	50.23	92.85	100	92.85	61.81	100	61.81
		63.9	100	63.9	47.88	95.7	50.05	50.19	100	50.19	92.34	100	92.34	68.83	100	68.83
		62.38	100	62.38	56.17	88.8	61.39	50.71	100	50.71	92.23	100	92.23	62.48	100	62.48
		61.71	100	61.71	48.98	97.53	50.16	50.16	100	50.16	92.31	100	92.31	65.59	100	65.59
		73.89	100	73.89	51.28	99.61	52.48	50.49	100	50.49	92.33	100	92.33	63.11	100	63.11
	Sym Test	50.6	100	50.6	27.1	56.1	45.4	48.9	100	48.9	54.8	100	54.8	69.8	100	69.8
		50	100	50	23.7	47.2	51.1	50.4	100	50.4	54.1	100	54.1	51.7	100	51.7
		49.3	100	49.3	23.4	47.5	50.3	51.5	100	51.5	55.5	100	55.5	54.2	100	54.2
		51.2	100	51.2	34.1	69.6	47.9	50.3	100	50.3	54.4	100	54.4	49.2	100	49.2
		54.4	100	54.4	33.2	91.5	45.2	51.5	100	51.5	55.1	100	55.1	50	100	50
	Asym Test	54.6	100	54.6	24.9	55.5	41.4	53.5	100	53.5	53.3	100	53.3	65.9	100	65.9
		49.7	100	49.7	27.1	49.6	54.8	53.3	100	53.3	53.9	100	53.9	48.3	100	48.3
		52.7	100	52.7	28.5	54.2	51.1	48.3	100	48.3	52.8	100	52.8	57.7	100	57.7
		47.8	100	47.8	43.6	77.3	55.1	53	100	53	53.9	100	53.9	52.6	100	52.6
		56.4	100	56.4	31.8	88.6	41.6	50.8	100	50.8	51.7	100	51.7	56.9	100	56.9
Asymmetrical Training	Training	90.43	99.53	90.84	76.81	94.91	79.67	94.33	98.94	95.23	99.33	99.85	99.45	99.75	100	99.75
		90.38	99.97	90.4	74.7	92.86	78.39	94.49	99.44	94.82	99.14	99.84	99.26	99.91	99.95	99.91
		90.6	100	90.6	76.88	96.33	79.02	95.25	99.51	95.6	99.16	100	99.16	98.7	100	98.7
		92.91	100	92.91	75.42	89.97	81.86	96.05	100	96.05	99.08	99.66	99.41	99.84	100	99.84
		89.82	98.52	90.76	78.23	91.86	83.64	94.5	97.31	96.85	98.91	99.66	99.22	99.84	100	99.84
	Sym Test	45	88.4	51.1	19.2	54.3	33	48.6	86.7	57.2	60.7	92.5	65.1	86.1	100	86.1
		49.2	98.1	50.1	8.2	21.9	28.1	47	81.3	55.8	57	93	60.6	95.1	97.9	95.1
		49.9	100	49.9	9.8	22.2	42	53.3	90.5	58.4	60.8	100	60.8	76.1	100	76.1
		50.3	100	50.3	14	47.4	31	56.6	100	56.6	58.9	91	64.8	93	100	93
		39.1	78.2	49.8	25.7	61.5	37	46.4	74.6	60.5	56.5	90.3	61.5	91.9	100	91.9
	Asym Test	53.5	89	60.6	28.5	59.6	43.2	57.6	90	65.5	70.9	95.7	74	79.5	100	79.5
		60.4	99.2	60.9	9.6	22.2	33	56.1	89.2	61.8	65.2	91.2	70.5	86.7	93	86.9
		60.7	100	60.7	11.5	24.2	50.3	62.3	93.5	66.3	69.7	100	69.7	65.1	100	65.1
		61.1	100	61.1	17.6	44.7	41.5	66.6	100	66.6	69.9	95.2	73.8	88.5	100	88.5
		54.3	89.8	60.6	35.1	66.6	46.9	57	81.6	69.3	65.4	96	67.9	83.3	100	83.3

Table 5.39 Percentage of correct predictions by trained networks processing training and test datasets. W = whole sequence; E = embedded section; P = penultimate symbol. All network architectures used pattern error- sensitive learning type, binary input representations, learning rate 0.3 and 0.75 momentum.

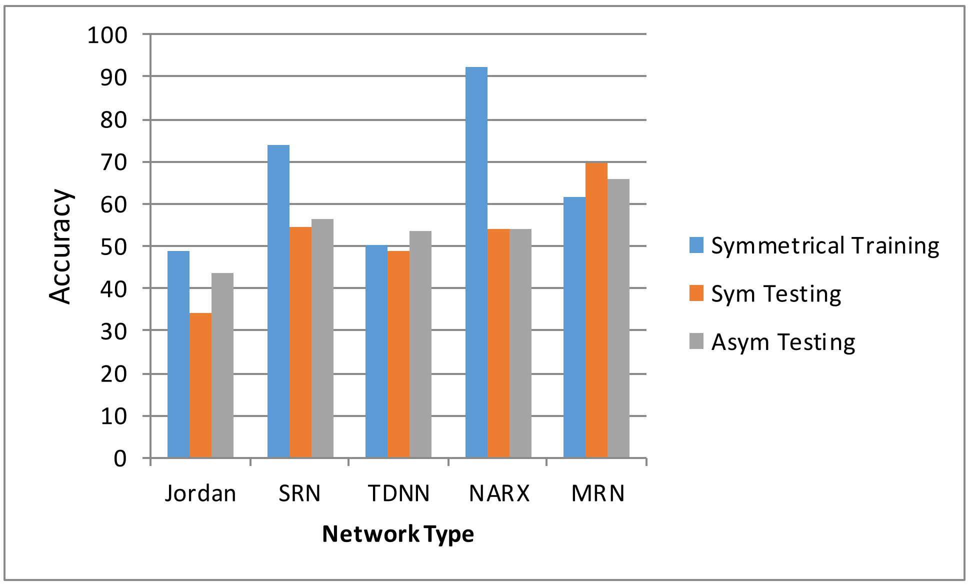
Graph 5.7 and 5.8 clearly show again that the accuracy of the networks is greatest when asymmetrical sequences are presented in training, regardless of whether the testing is with symmetrical or asymmetrical sequences. The average accuracy of all the networks trained asymmetrically is: 63.81% when tested on asymmetrical sequences; and 56.03% when tested on symmetrical sequences. The average accuracy of all the networks trained symmetrically is: 54.66% when tested on asymmetrical sequences and 52.26% when tested on symmetrical sequences.

		Jordan	SRN	TDNN	NARX	MRN	ESN
Asymmetrical	Training	78.23	92.91	96.05	99.33	99.84	91.7
	Sym Test	25.7	50.3	56.6	60.7	93	49.9
	Asym Test	35.1	61.1	66.6	70.9	88.5	60.7

Table 5.40 The best performance of asymmetrically trained networks tested with both dataset types

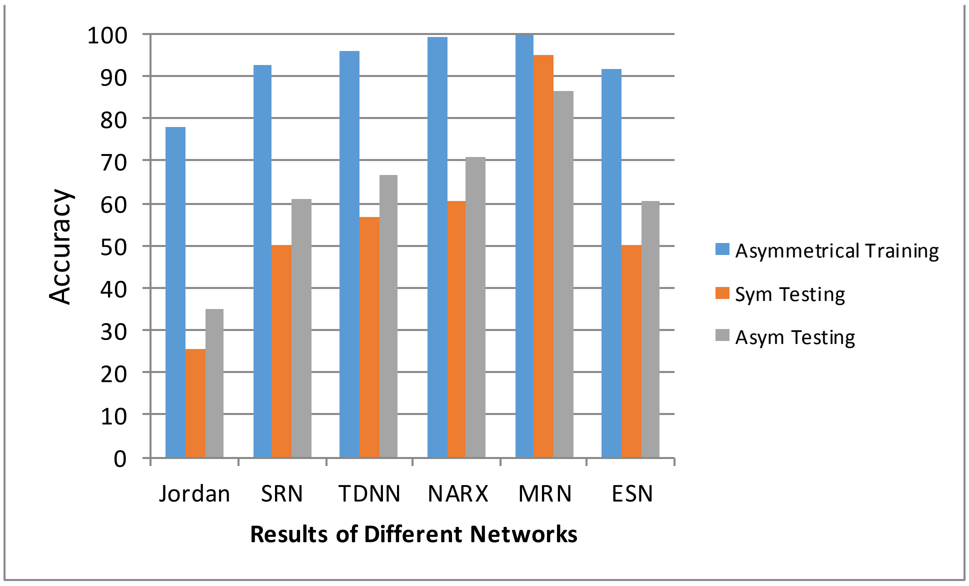
		Jordan	SRN	TDNN	NARX	MRN
Symmetrical	Training	48.98	73.89	50.23	92.34	61.81
	Sym Test	34.1	54.4	48.9	54.1	69.8
	Asym Test	43.6	56.4	53.5	53.9	65.9

Table 5.41 The best performance of symmetrically trained networks tested with both dataset types



Graph 5.7 Performance of best symmetrically trained network for each network type

The bar chart shown in Graph 5.8 illustrates the results of the six networks trained on asymmetrical sequences and tested on biased and unbiased sequences. Overall, most of the networks were more capable of recognising asymmetrical sequences than symmetrical ones. The second observation from the graph is that the MRN is the superior network for the prediction task.

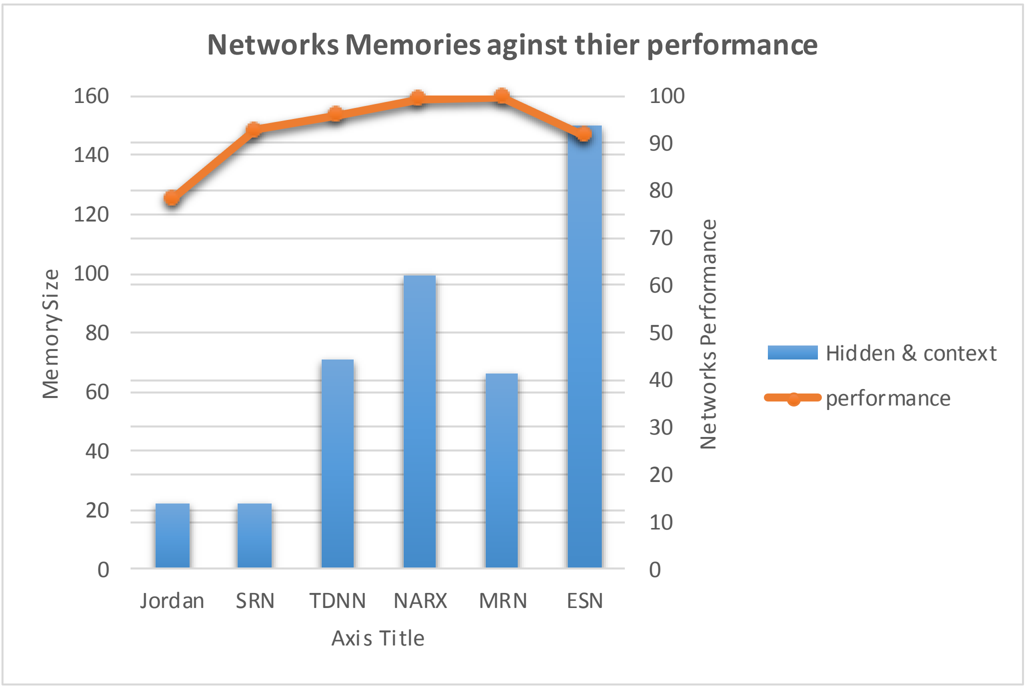


Graph 5.8 The selected symmetrical test network for the five networks using asymmetrical training

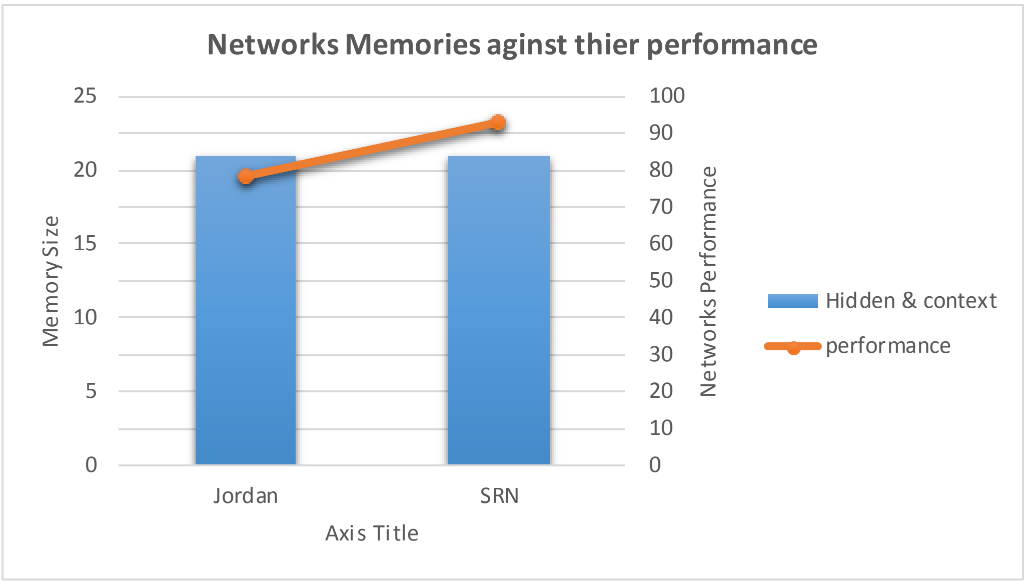
Analysing the results further; the performance of the networks tends to rise when the architecture of the network has more memory and also the associated increase in connections between the layers. Table 5.42 and Graph 5.9 show the performance alongside the networks memory. However, this is not a simple correlation as performance of the networks is also strongly influenced by the architecture. Table 5.43 and Graph 5.10 demonstrate this by showing that with the same memory, two different architectures (Jordan and SRN) perform very differently.

Networks	Jordan	SRN	TDNN	NARX	MRN	ESN
Hidden & context	22	22	71	99	66	150
performance	78.23	92.91	96.05	99.33	99.84	91.7

Table 5.42 The performance of the networks against the memory



Graph 5.9 the performance of the networks against networks memories



Graph 5.10 the effect of networks structures on the networks have same size of memory

Networks	Jordan	SRN
Hidden & context	22	22
performance	78.23	92.91

Table 5.43 the performance of networks have same number of memory

To conclude, these investigations studied various networks given the task of learning a context-free grammar, the Embedded Reber Grammar. The study shows the superiority of the MRN over the other networks studied. Noise injection has enhanced MRN performance by nearly 10%; however, with SRN, NARX and ESN it produced poor results. However, more investigation is required since just unit noise has been conducted with these networks with one range value 0,01. The present study provides additional evidence with respect to the memory size, architecture, parameters and learning algorithm on the performance of the networks. The next chapter investigates the question of why the MRN is superior over the other networks.

The approach taken in Chapter 6 is to consider the internal representations of the networks. Principle Component Analysis (PCA) is used to show that the MRN is able to maintain a higher level of discrimination between the upper and lower embedded sections of the grammar in terms of its internal representations. The difference drops down very low for the poorer performing architectures (SRN and ESN), whereas the NARX and MRN maintain a higher difference between the two sets of states.

Chapter 6

6. Understanding the Internal Representations Formed.

The aim of this chapter is to describe a variety of concrete examples that show the internal representations of SRN, NARX and ESN that have learnt to predict symbols in sequences from the embedded Reber grammar and evaluate them against the MRN. The networks were trained and tested with both biased and unbiased sequences. The results give higher-ranking of MRN performance over the other networks. The outcomes of the networks were presented in chapter five. In order to analyse the internal representations, Principle Component Analysis (PCA) was applied to the weights within the networks. For the SRN, NARX, ESN and MRN architectures this included networks trained on both asymmetrical and symmetrical sequences. However, for the ESN architecture the networks had only been trained using asymmetrical sequences as described previously. Understanding how each network distributes the states of the grammar and a comparison between those networks where conducted to demonstrate which one distributes the data in systematic way and which one is superior. Networks trained with the asymmetrical dataset and tested with the asymmetrical test dataset were considered due to superior performance by MRN and NARX with this these data sets, Table 6.1 illustrates these models and their performance. The evaluation of the symmetrical training dataset is in Appendix E.

Network	Hidden units	Training accuracy	Testing accuracy
SRN	15	92.91%	61.1%
NARX	15	99.33%	70.9%
MRN	10	99.91%	97.7%
ESN	150	91.7%	60.7%

Table 6.1 The models evaluated using asymmetrical sequences for training and testing

6.1 Visualisation of the Internal Representations Formed

So far the underlying representation or 'hypothesis' formed by recurrent networks has been treated as a 'black box'. The hidden units inside these networks express the grammatical knowledge encoded by the weights. Understanding how these units respond to each input symbol over time, may provide the ability to determine whether the networks have formed an adequate representation of the underlying embedded Reber grammar or not. One approach or method to extract the rules mentioned in chapter four, is to try to extract the rules (rules that are informed of "$x_1 = u(x_1), x_2 = u(x_1), \dots x_n = u(x_n)\ then\ P_j$" Where x_i is the input to the network, $u(x_i)$ is one of the value of x_i and P_j is the network's prediction) from the weights, to evaluate the network (Elman 1990, Craven, Shavlik 1994, Setiono, Liu 1995, Bullinaria 1997). By this approach, the trained neural networks can be studied by extracting symbolic rules that describe their classification behaviour (Jacobsson, 2005). Principle component analysis (PCA) has been used to visualise the internal representations of SRN, MRN and ESN to analyse and evaluate the networks. (Cartling 2008) used PCA to investigate the internal representation of SRN on the implicit acquisition of a context free grammar and his results show that a systematic selection of parameters leads to a well organised internal representation of grammatical elements and consequently leads to a better performance. The internal representation refers to the activations of the hidden layer of the network. All the sequences in the training dataset have been studied in a two-dimensional subspace of the internal-representation space that is spanned by all possible combinations of two eigenvectors of the covariance matrix equation (4.3) from those corresponding to largest absolute eigenvalues of the hidden unit used. The steps used to calculate the PCA are:

1. Compute the mean of each internal hidden unit activation.
2. Calculate the variance between each node and its mean.
3. Compute the covariance matrix.
4. Calculate eigenvalue and eigenvectors of the covariance matrix.
5. Choose components and derive a new data set.

The principle components are ordered in descending magnitude, according to the eigenvalues. When the network has been trained, it has acquired a capacity to represent relations between symbols. The primary concern in the network applied context-free language to handle the grammatical elements in the sequences and rules.

The properties of the two datasets are illustrated in Table 4.3 and Table 4.5. All the sequences used in this set have been studied in all the two-dimensional subspaces of the internal representation space that are spanned by all possible combinations of the covariance matrix, Equation (3.4) in chapter four. A number of terms have been used to analyse the sequences that have been selected. Table 6.2 explains the following terms.

Term	Meaning
G_{Sn}	Refers to a particular state, symbol and grammar i.e. G=upper (u) or lower (l), s=input symbol, n=state of the grammar (1 to 7)
F	**False (incorrectly predicted)**
T	True (correctly predicted)
Penult & P	Penultimate symbol of an input sequences
E	Embedded grammar (either upper or lower)

Table 6.2 Meaning of some terms

The terms from Table 6.2 are used to identify symbols in the embedded Reber grammar as depicted in Table 6.2. For example, Uv4 refers to the V symbol generated on the transition from state four to state six in the upper half of the grammar schema.

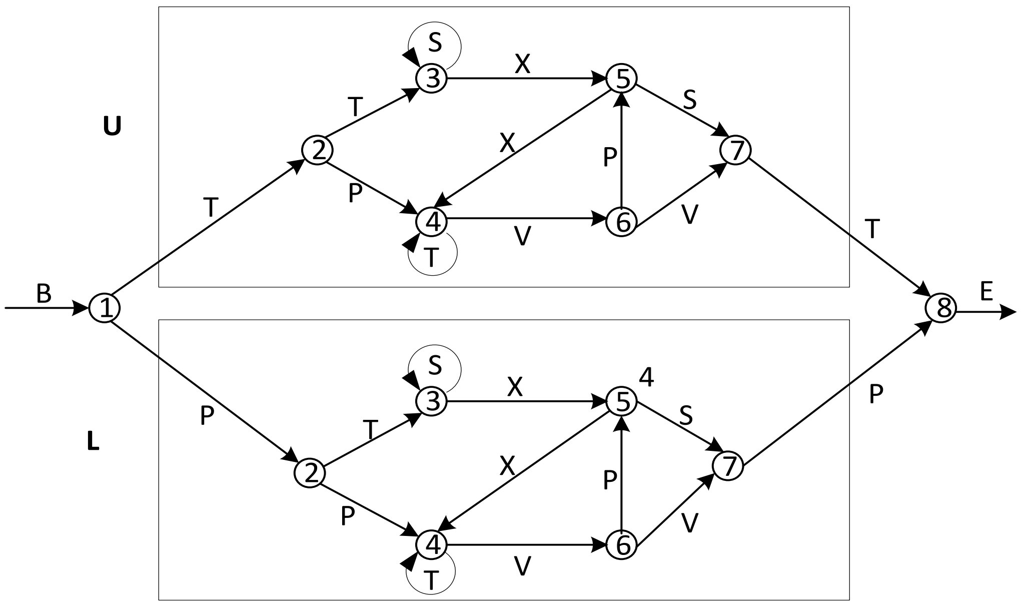


Figure 6.1 shows the states of the embedded Reber Grammar

Four different sequence lengths have been chosen (6, 8, 16 and 26) from the trained dataset, to analyse the internal representations of the networks these sequence were passed each path of the grammar. Table 6.3 shows the sequences that have been selected according to the length of the sequences (from both symmetrical and asymmetrical training sets).

Symmetrical/Asymmetrical Sequences			
Length	No	Embed	Sequences
6	1	U	BTPVVT
	2	L	BPPVVP
	3	U	BTTXST
	4	L	BPTXSP
8	5	U	BPPTTVVP
	6	L	BTPTTVVT
	7	U	BTPTVPST
	8	L	BPPTVPSP
	9	U	BTTSSXST
	10	L	BPTSSXSP
	11	U	BTTXXVVT
	12	L	BPTXXVVP
16	13	BPTXXTTTVPXTTVVP	
26	14	BPPVPXTVPXVPXVPXTVPXVPXVVP	

Table 6.3 Symmetrical and asymmetrical Sequences that have been selected

6.1.1 Internal Representations of the SRN

For this study, the SRN results described in chapter five from training with asymmetrical sequences is investigated; results are shown Table 6.1, which was used to explore the internal representation of the SRN's hidden units. Table 6.4 provides the experimental data results obtained from the SRN. These sequences have been analysed using the PCA method to understand the internal representation formed by the network. Sequences numbered 1, 2, 5, 6, 7 and 8 in the table have been selected to consider the internal representations of the network for correctly predicted sequences. The corresponding trajectories are shown in Figure 6.2 a-c, respectively in the subspace of the internal representation space defined by principle component 1 (PC1) and principle component 2 (PC2). The start point is located in the fourth quarter of bi-dimensional space, for nearly all of the grammar states of 1, 2 and 3; upper and lower located in the second and third quarter of the plane. State four and five are located in the first quarter of the plane while six and seven are in the fourth quarter. The common characteristic of the trajectories of most of the asymmetrical sequences is that they spread on the surface as shown in the Figure 6.2 (a, b). However, when there is a self-looping state three, they are located on the third and fourth quarter of the surface.

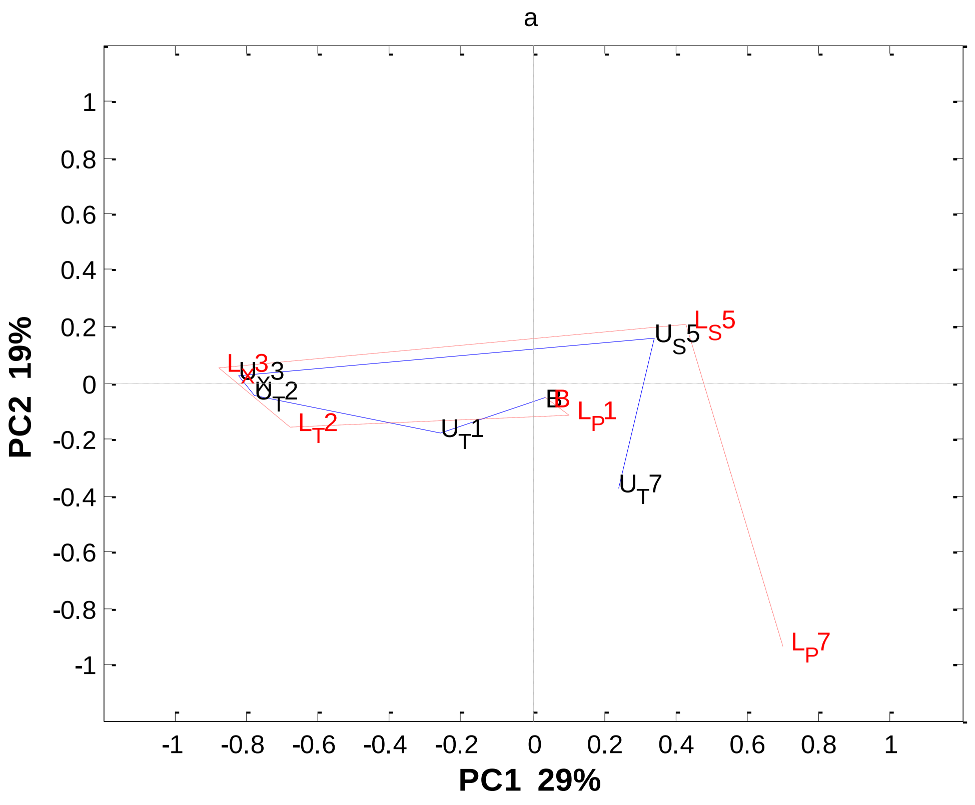
SRN using Asymmetrical sequences					
No	Length	Embed	Sequences	Prediction	Reason for Failure
1	6	U	BTTXST	T	
2		L	BPTXSP	T	
3		U	BTPVVT	T	
4		L	BPPVVP	T	
5	8	U	BTTXXVVT	T	
6		L	BPTXXVVP	T	
7		U	BTTSSXST	T	
8		L	BPTSSXSP	T	
9		U	BTPTVPST	T	
10		L	BPPTVPSP	F	Penult incorrect
11		U	BTPTTVVT	F	Penult incorrect
12		L	BPPTTVVP	T	
13	16		BPTXXTTTVPXTTVVP	T	
14	26		BPPVPXTVPXVPXVPXTVPXVPXVVP	T	

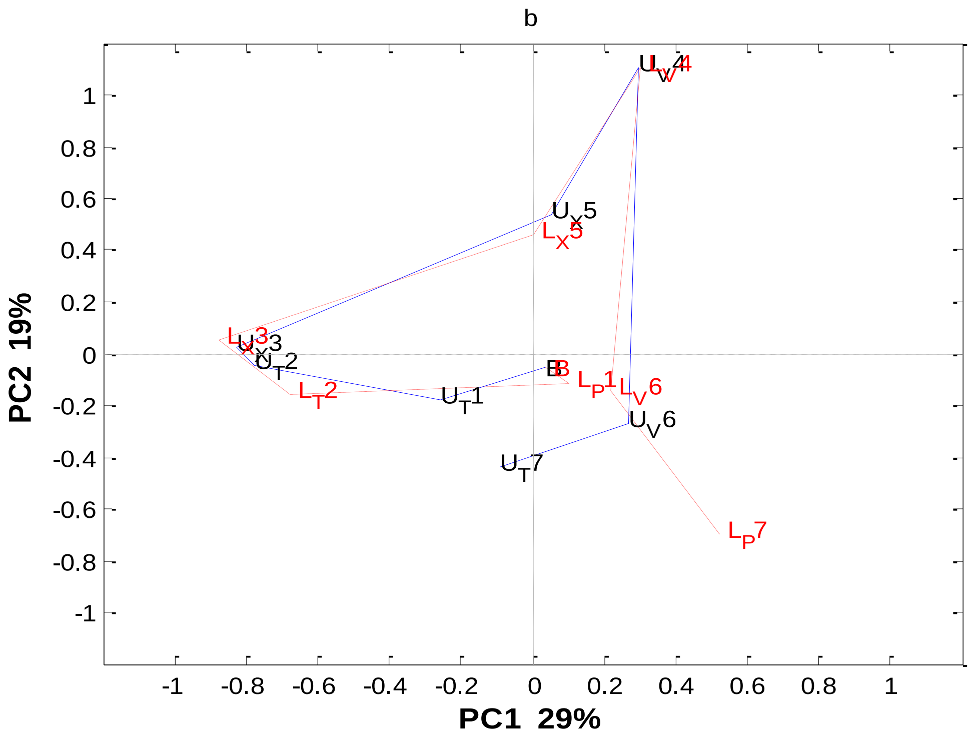
Table 6.4 Results of SRN using asymmetrical sequences and their prediction results

To analyse the results more, a number of correctly predicted sequences have been selected and the centroid of each state calculated. Table 6.5 illustrates the centroid of each sate for both upper and lower embedded part of the sequences.

Embedded	States	Centroid		Embedded	States	Centroid	
		PC1	PC2			PC1	PC2
Upper	1	-0.25646	-0.17579	Lower	1	0.10371	-0.11526
	2	-0.77840	-0.04289		2	-0.36975	-0.12227
	3	-0.59789	-0.23712		3	-0.70120	-0.20574
	4	0.27376	1.00098		4	0.19702	0.73998
	5	-0.14873	0.40160		5	-0.11504	0.23093
	6	0.42873	-0.50940		6	0.37518	-0.34150
	7	-0.11478	-0.41299		7	0.36923	-0.72087

Table 6.5 SRN: Centroid of the states for asymmetrical sequences (correctly predicted)





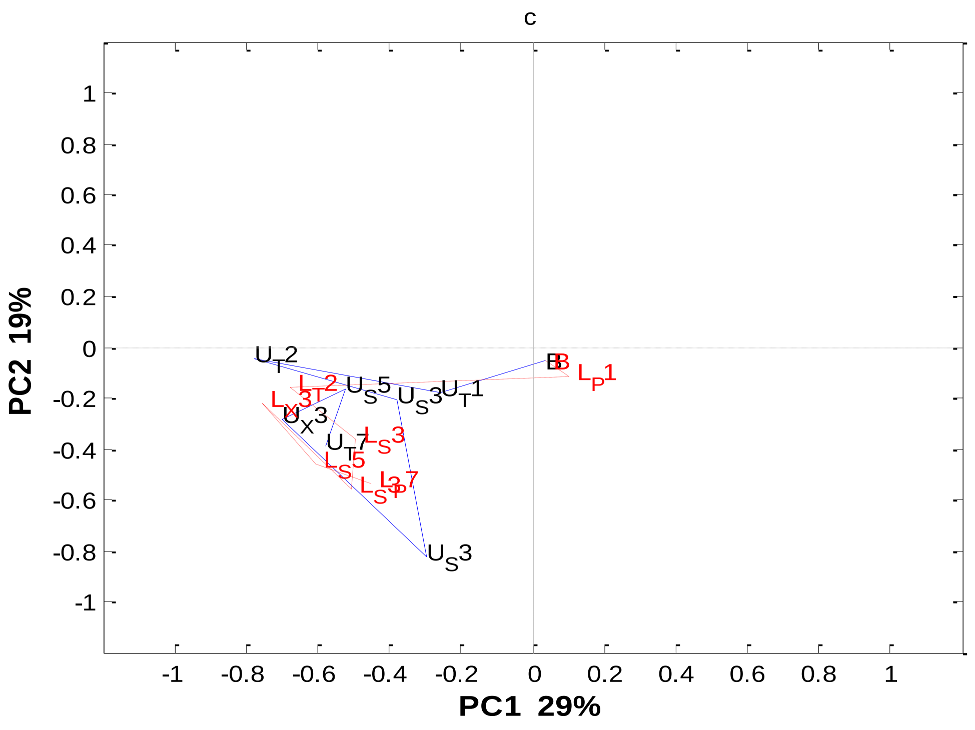


Figure 6.2 Plots of the two most significant principle components of the hidden layer activations of a asymmetrically trained SRN, presented with three pairs of symbol sequences (in a, b and c respectively) from the ERG. Each pair has the same embedded sequence but different initial symbol so that one is in the lower half (dashed lines) and the other is in the upper half (solid blue lines). The sequences are in the table 6.7: (a) 1, 2; (b) 5, 6; (c) 7, 8 and with respect to principal components PC1 and PC2. (a BPTXSP/BTTXST, b) BPPVVP/BTPVVT, c) BTTSSXST/BPTSSXSP

A drawing of the centroid of the states of correctly predicted sequences is shown in Figure 6.3. The figure shows the difference in the distribution of the grammar states over the plane in the coordinates PC1 and PC2. The plots of the different principle component pairs (PC1 and PC2; PC1 and PC3) in all the examples of the SRN, shows that the penultimate symbol is always in the fourth quarter (bottom right) of the plane and the start point is always in the first quarter (top right). Plots of PC2 and PC3 however, establish different trajectories, start and penultimate positions; Appendix C depicts some examples of them. The domain of the upper states on PC1 is distributed from approximately -0.7 to 0.5 and the lower states are approximately between -0.3 to 0.3. In addition to this, there is also, a divergence in the range of the PC2 axis between upper and lower states. Moreover, there is a kind of clustering in each state that can be seen in the figure, since each state is located in a different position in plane. These may explain why the network could differentiate between both embedded parts and memorise long term dependency.

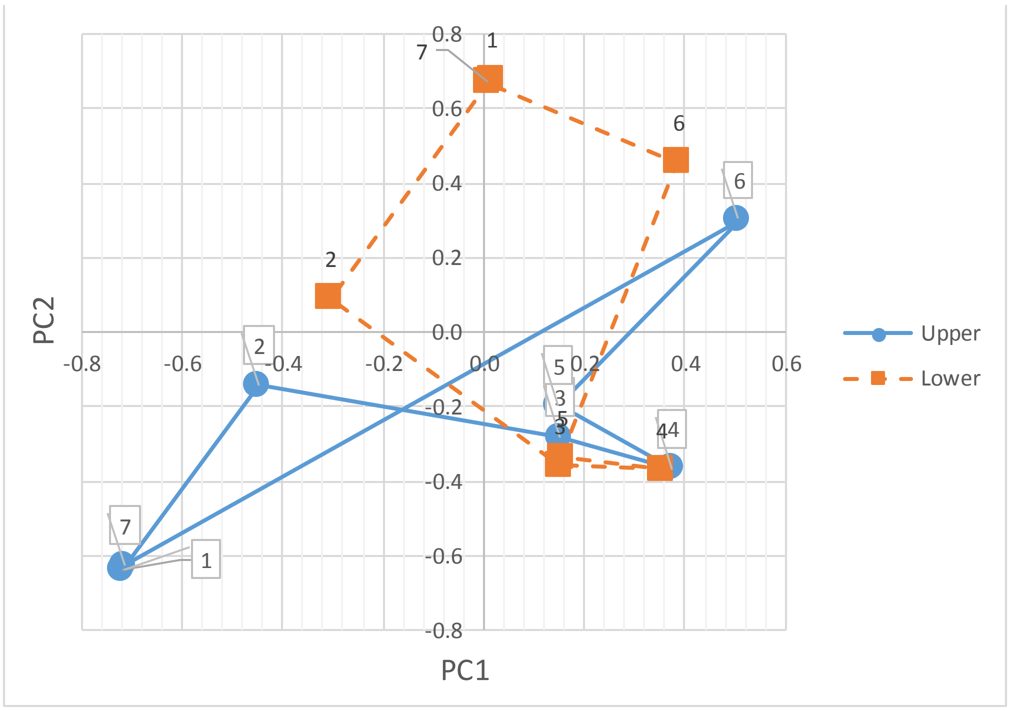
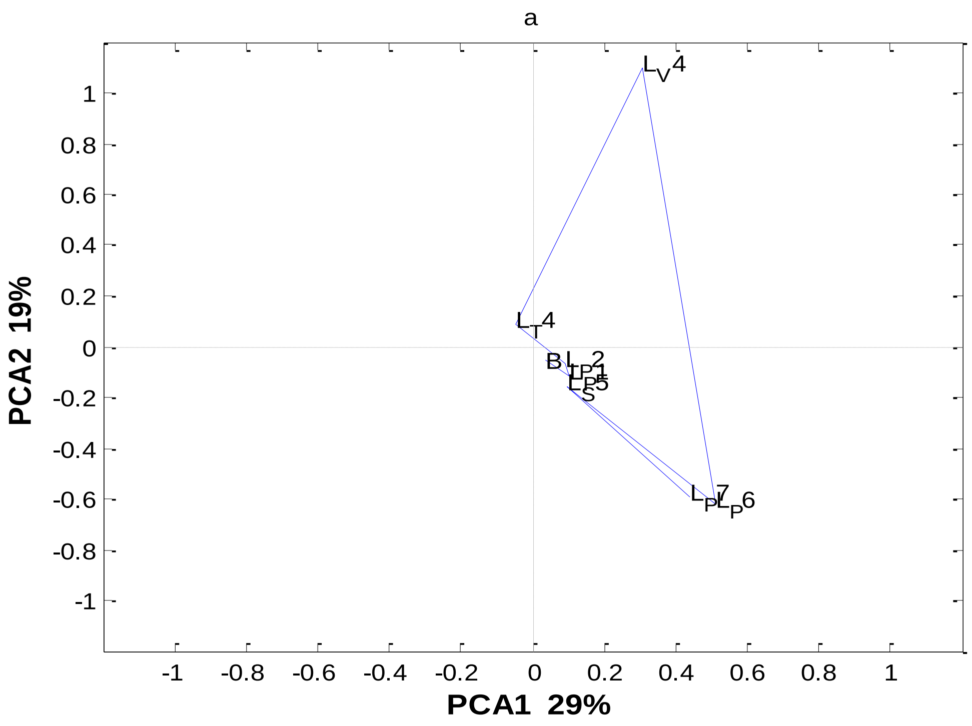


Figure 6.3 SRN: Centroid of each state using asymmetrical sequences (correctly predicted sequences)



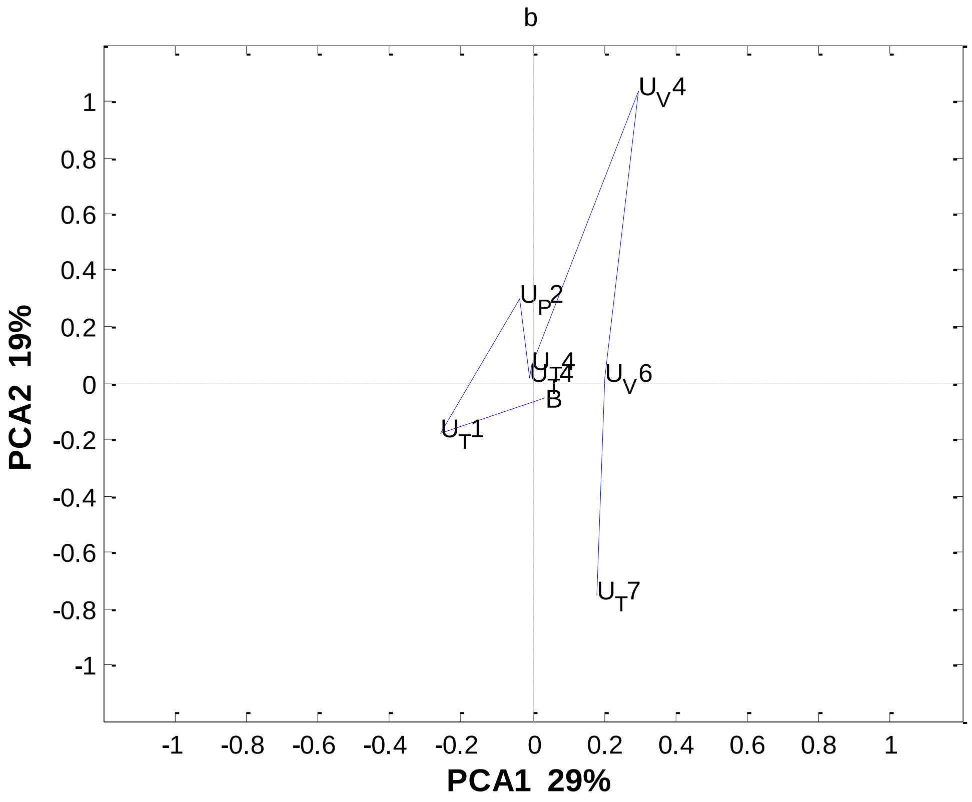


Figure 6.4 Plots of the two most significant principle components of the hidden layer activations of an asymmetrically trained by SRN, trajectories of two non-identical asymmetrical sequences that were incorrectly predicted by the SRN. (a) The lower embedded part BPPTVPSP. (b) The upper embedded part BTPTTVVT. With respect to principal components PC1 and PC2.

To investigate the internal representation when sequences are incorrectly predicted, a number of sequences that have all the grammar states were considered. This was to examine the trajectories through all seven states of both the upper and lower embedded grammars (in the length of 12 to 24 sequences). Figure 6.4 illustrates the lower and upper part of the sequences that have length eight which were predicted incorrectly by the SRN. Both of the sequences have incorrectly predicted symbols. These symbols were the penultimate symbols of the sequences. By drawing the states of the unpredicted sequences, it seems there is a similarity in the distribution of the states on the surface. This may be due to the failure of the sequences being in the penultimate symbol, which makes it difficult to observe the difference between the trajectories of both cases. Thus, twelve correct sequences and incorrect sequences with different lengths have been chosen and their range computed to investigate the range of the state 7 for both cases. Figure 6.5, clearly indicates that there is a distinct distance between final states for correctly and incorrectly predicted sequences (i.e. where PC1 <0.8 appears reserved for incorrect predictions and >=0.8 for correct predictions)

Embedded	States	Range		Embedded	States	Range	
		PC1	PC2			PC1	PC2
Upper	1	0.00009	0.00010	Lower	1	0.00008	0.00017
	2	0.74197	0.34151		2	0.77149	0.09085
	3	0.44686	0.46860		3	0.37745	0.41429
	4	0.30652	1.07506		4	0.34939	1.01875
	5	0.81921	0.92739		5	0.97138	0.61542
	6	0.44090	0.85219		6	0.06009	0.10806
	7	0.54955	0.38318		7	0.08450	0.11046

Table 6.6 SRN: Range of the states for asymmetrical sequences (Incorrectly predicted sequences)

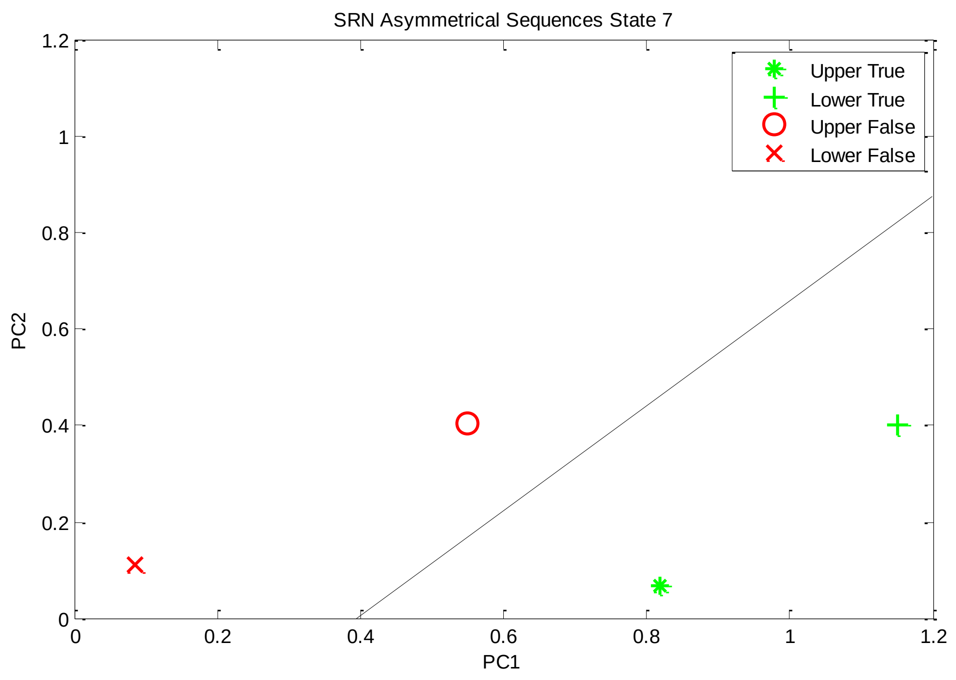


Figure 6.5 SRN: located ranges of state seven for correctly predicted and incorrectly predicted asymmetrical sequences.

The investigation of the internal representation of the SRN has shown that the network distributes the states of the grammar in a systematic way (in a consistent manner). In addition to this, there are ranges where, if state seven is located inside them, the network prediction will fail to recognise the symbols. However, the poor performance of the network needs to be explored.

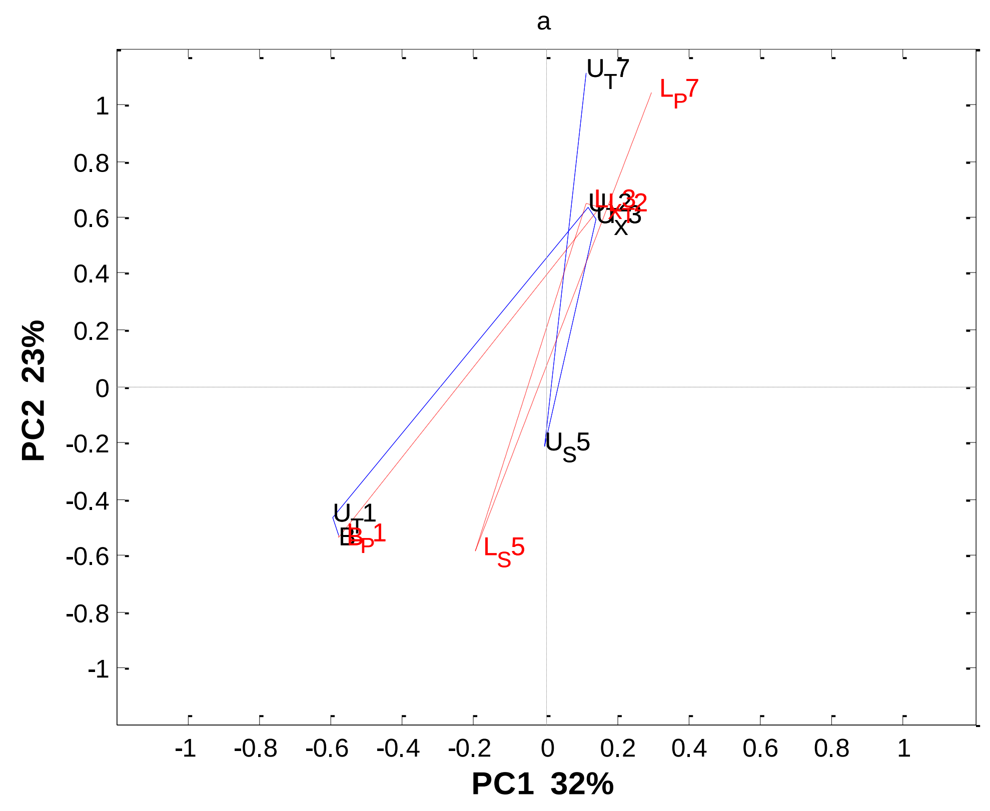
6.1.2 Internal Representations of the NARX

In chapter three a graph of NARX with 15 hidden units, eight feedback boxes from the output unit and four shifted boxes from the input, is depicted. The best performance of the NARX was selected to investigate its internal representation. The network has 15 hidden units and the PCs have been computed for the whole corpora of the training dataset. The same sequences that were used in the previous network to investigate the trajectories of the components are used here as illustrated in Table 6.7. The table shows that the network recognises all the sequences that have been selected except for the sequence with length 26. The trajectories of the sequences are depicted in Figure 6.6. These show how the network represents the sequences in the space. It shows the systematic state of the grammar represented by the hidden units of the network. The trajectories of the grammar states for both upper and lower embedded parts of the grammar are located in approximately the same position in the space for each sequence.

NARX using Asymmetrical sequences					
No	Length	Embed	Sequences	Prediction	Reason for failure
1	6	U	BTTXST	T	
2		L	BPTXSP	T	
3		U	BTPVVT	T	
4		L	BPPVVP	T	
5	8	U	BTTXXVVT	T	
6		L	BPTXXVVP	T	
7		U	BTTSSXST	T	
8		L	BPTSSXSP	T	
9		U	BTPTVPST	T	
10		L	BPPTVPSP	T	
11		U	BTPTTVVT	T	
12		L	BPPTTVVP	T	
13	16		BPTXXTTTVPXTTVVP	T	
14	26		BPPVPXTVPXVPXVPXTVPXVPXVVP	F	Penult incorrect

Table 6.7 Sequences results for NARX and the position of the incorrectly predicted symbol

A number of incorrectly predicted sequences have been selected for experiment to explore the trajectories of the states of these sequences. Prediction by the NARX for all the sequences, failed at the penultimate symbol. Figure 6.7 illustrates the trajectories of sequences that have a length of 14 symbols. The trajectories show systematic distribution of the states within the space. Figure 6.8 (a) depicts trajectories of an incorrectly predicted sequence with sequence length 16 and (b) the trajectories of correctly predicted sequence having the same length. It shows that the trajectories of state seven in the grammar are dissimilar for each sequence. Consequently, several correctly and incorrectly predicted sequences were selected, considering in particular state seven, to visualise the state in two-dimensional space. Figure 6.9 shows the ranges of PC1 and PC2 of state seven for these sequences. The figure illustrates the difference in each case.



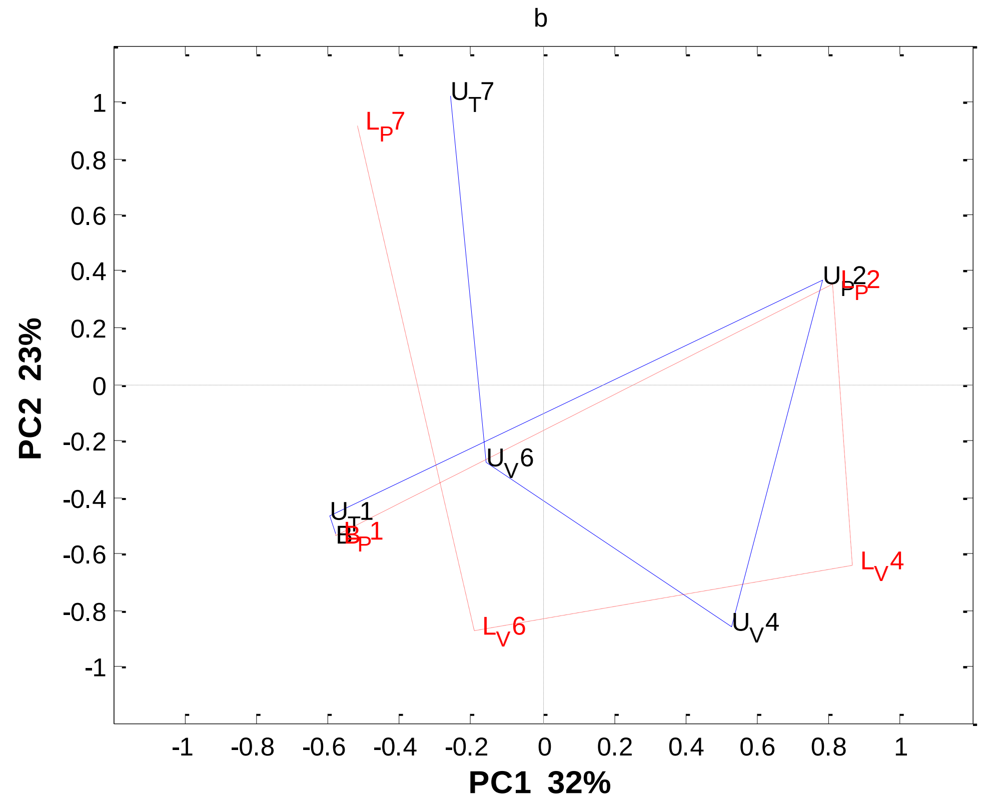
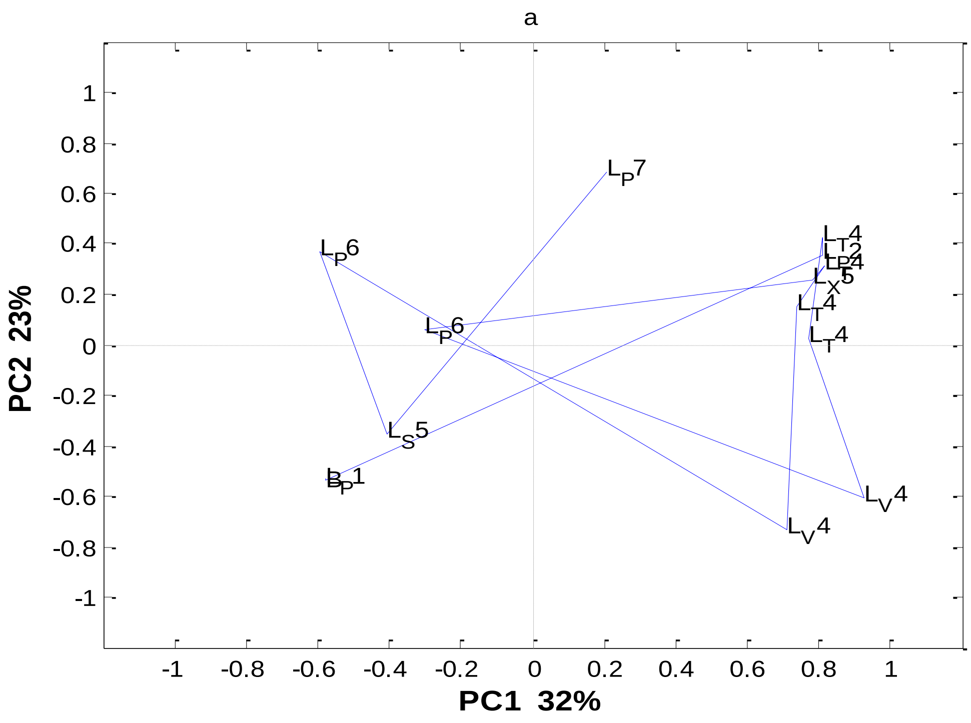


Figure 6.6 Plots of the two most significant principle components of the hidden layer activations of an asymmetrically trained NARX, (a, b) are the trajectories of the sequence numbered 1,2,3,4 a) BTTXST/BPTXSP and b) BTPVVT/BPPVVP in Table 6.7 which have a length of six symbols.



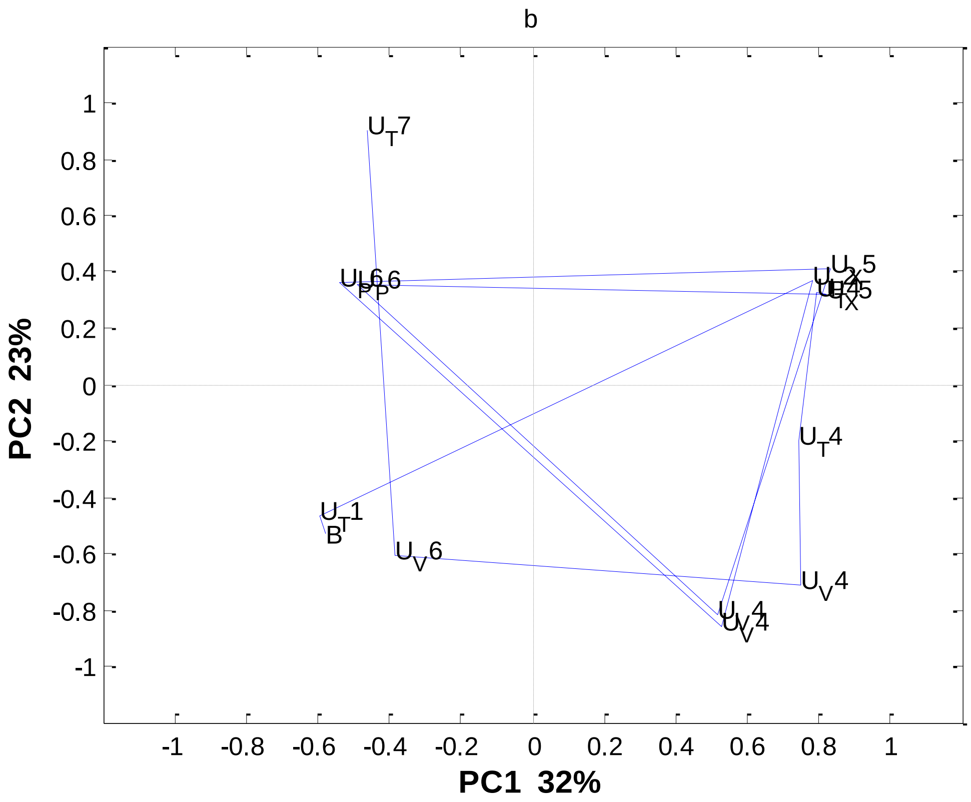
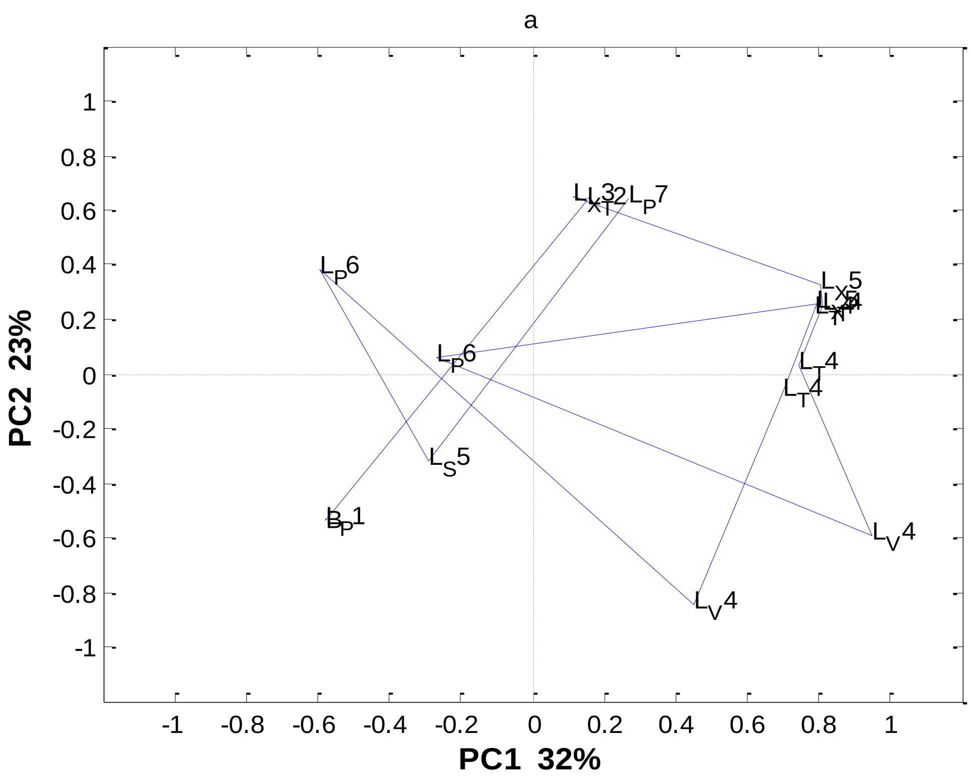


Figure 6.7 Plots of the two most significant principle components of the hidden layer activations of an asymmetrically trained NARX, (a, b) trajectories of two non-identical sequences that are incorrectly predicted a) BPPTTVPXTTVPSP and b) BTPVPXVPXTTVVT



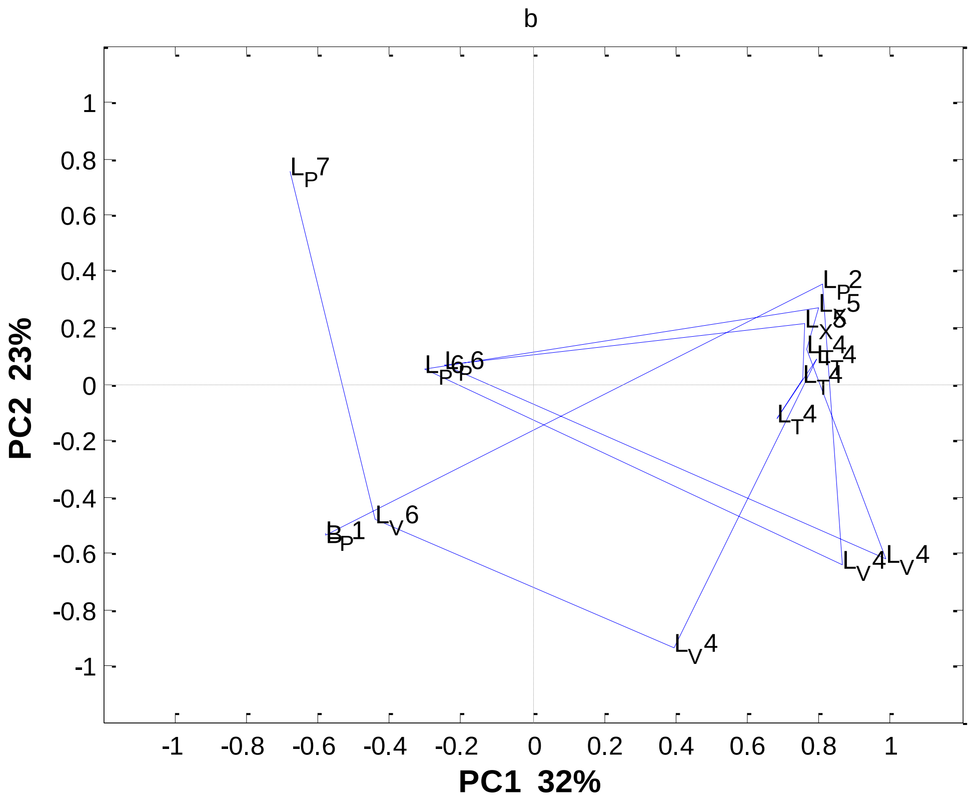


Figure 6.8 Plots of the two most significant principle components of the hidden layer activations of an asymmetrically trained NARX. a) Trajectories of 16-length sequences that are incorrectly predicted "BPTXXTTVPXTTVPSP and b) is the correctly predicted sequence of the same length BPPVPXTVPXTTTVVP.

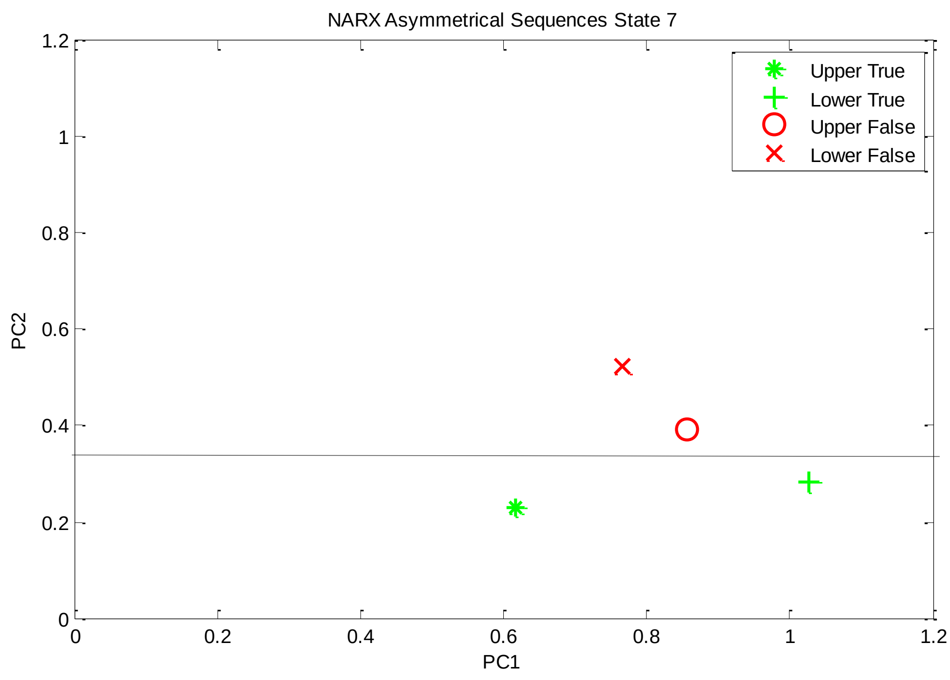


Figure 6.9 NARX: Range of state seven for both correctly predicted sequences and incorrectly predicted sequences (failes on the penultimate symbol)

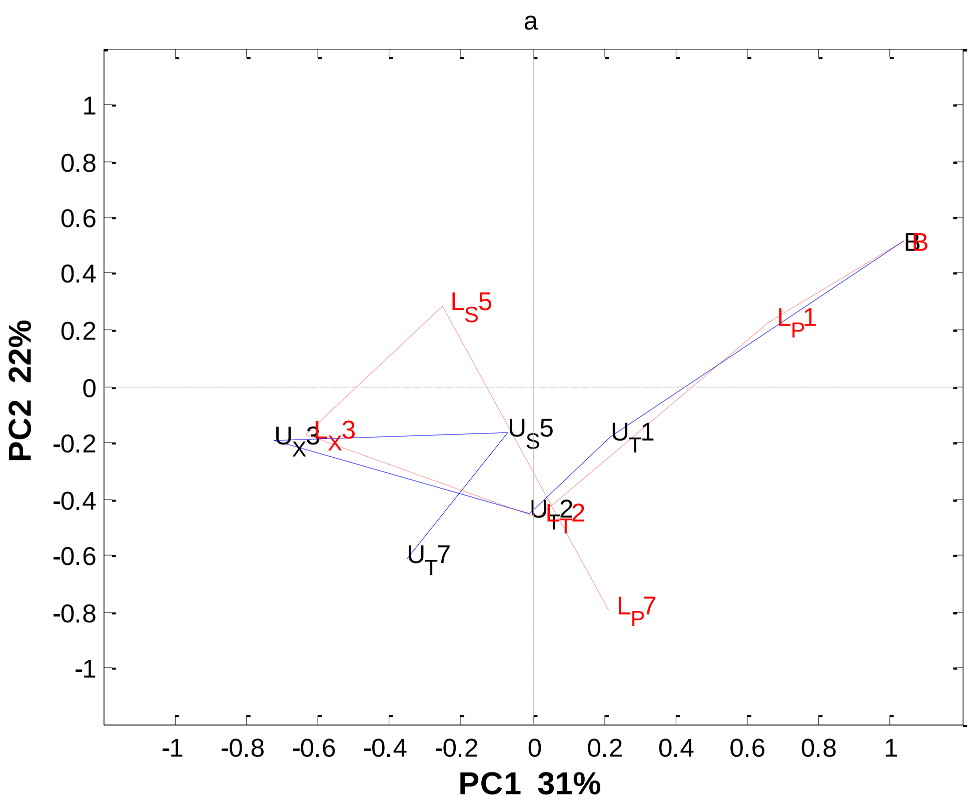
To conclude, the investigation provides an analysis and evaluation of internal representation of the NARX. The trajectories of the states are distributed by the internal representation on the plane in a systematic way, e.g. state one is located in the third quarter for both parts of the grammar and state seven is located in the first and second quarter. There is a kind of clustering that hidden units produce when the predicted symbols of state seven are plotted in a range, which differs from the range where symbols are incorrectly predicted, as illustrated in Figure 6.9. The difference between the trajectories of SRN and NARX are in the locations within PC1-PC2 space. However, there is a small variance between the upper and lower representations in PC1-PC2 space for the embedded sections in the NARX.

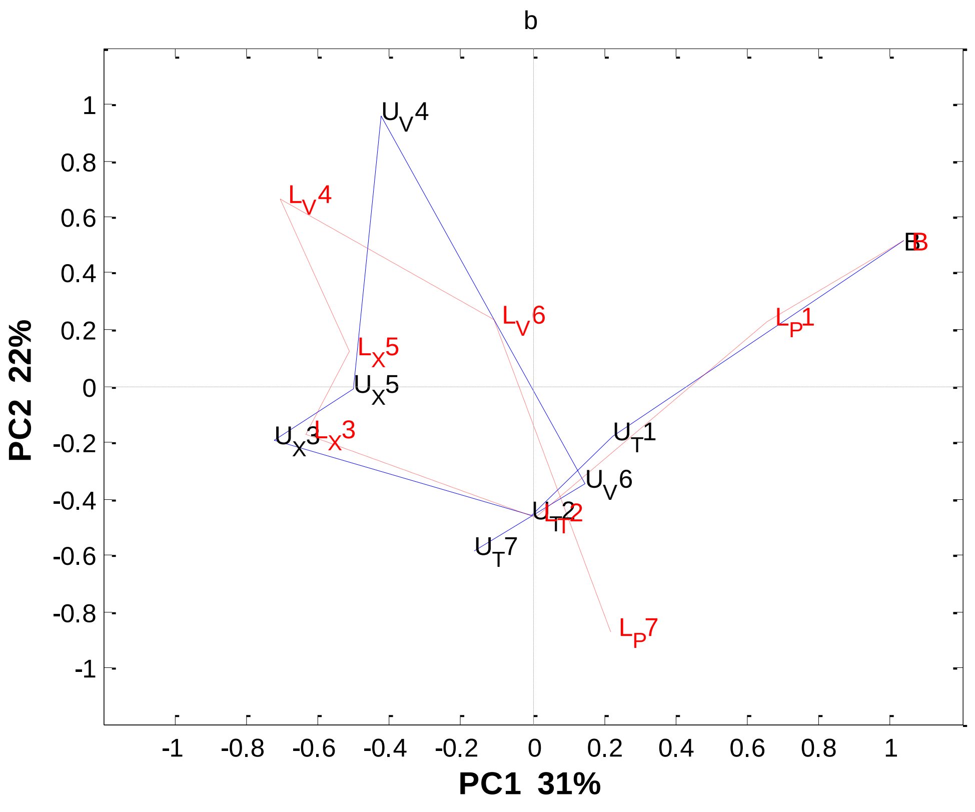
6.1.3 Internal Representations of the MRN

The results analysed in this investigation are taken from the MRN that was shown in chapter three and that produced the results presented in chapter five. The results were acquired from the MRN having ten hidden units and four memory boxes. Table 6.8 shows asymmetrical sequences that were correctly predicted by the MRN. The trajectories of these sequences have been studied to investigate how the internal representation of the MRN represented the nominal states. Figure 6.10 (a, b, c) shows the sequences in the Table 6.8 numbered 1, 2, 5, 6, 7 and 8 respectively. The positions of state seven for sequences generated by traversing the upper embedded section of the grammar are located in the third quarter (bottom left), whereas the same state is in the fourth quarter (bottom right) for sequences generated using the lower embedded section. Additionally, the positions of state one for both upper and lower embedded sections are located in the fourth and first (top right) quarters respectively. The other state trajectories are located in different subspaces of the surface with consistent locations.

No	Length	Embed	Sequences
1	6	U	BTTXST
2		L	BPTXSP
3		U	BTPVVT
4		L	BPPVVP
5	8	U	BTTXXVVT
6		L	BPTXXVVP
7		U	BTTSSXST
8		L	BPTSSXSP
9		U	BTPTVPST
10		L	BPPTVPSP
11		U	BTPTTVVT
12		L	BPPTTVVP
13	16		BPTXXTTTVPXTTVVP
14	26		BPPVPXTVPXVPXVPXTVPXVPXVVP

Table 6.8 Asymmetrical sequences correctly predicted by MRN


a
PC2 22%
PC1 31%
B
B
L_P1
L_S5
U_X3
L_X3
U_S5
U_T1
U_T2
L_T2
U_T7
L_P7


b
PC2 22%
PC1 31%
U_V4
L_V4
L_V6
B
B
L_P1
L_X5
U_X5
U_X3
L_X3
U_T1
U_V6
U_T2
L_T2
U_T7
L_P7

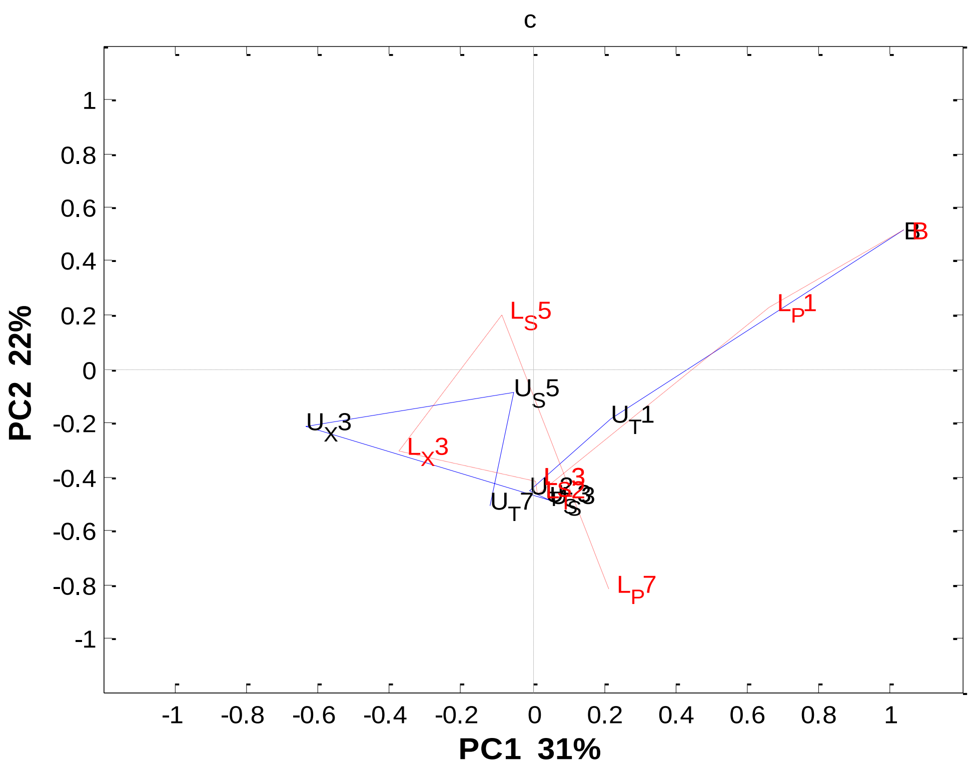
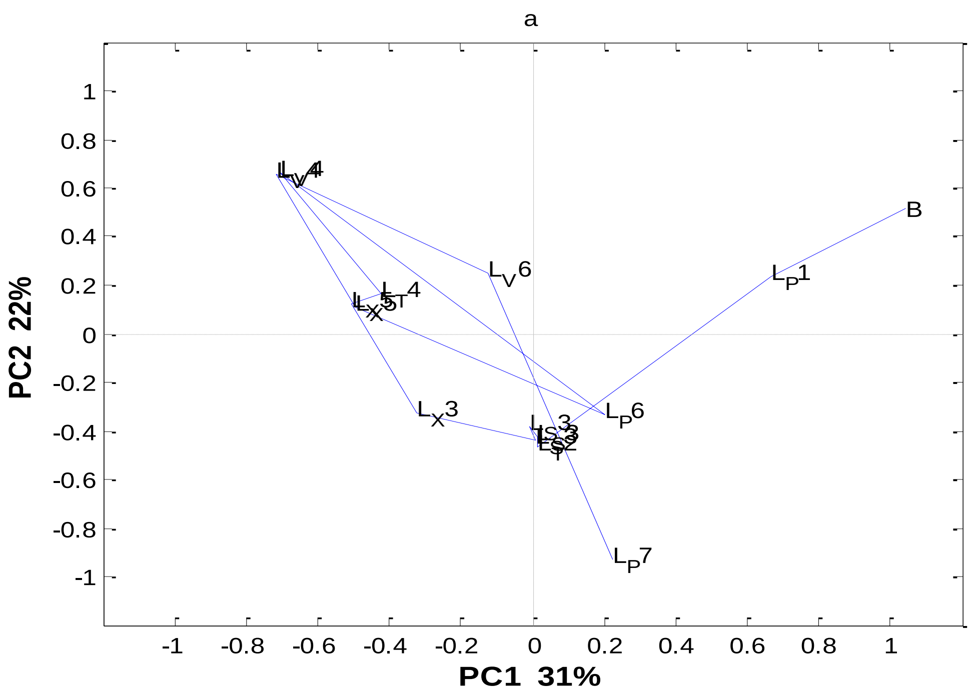


Figure 6.10 Plots of the two most significant principle components of the hidden layer activations of an asymmetrically trained by MRN (a) trajectories of the six length sequences BTTXST/BPTXSP (b) BTTXXVVT/BPTXXVVP and (c) BTTSSXST/BPTSSXSP are trajectories of eight length sequences upper and lower embedded.



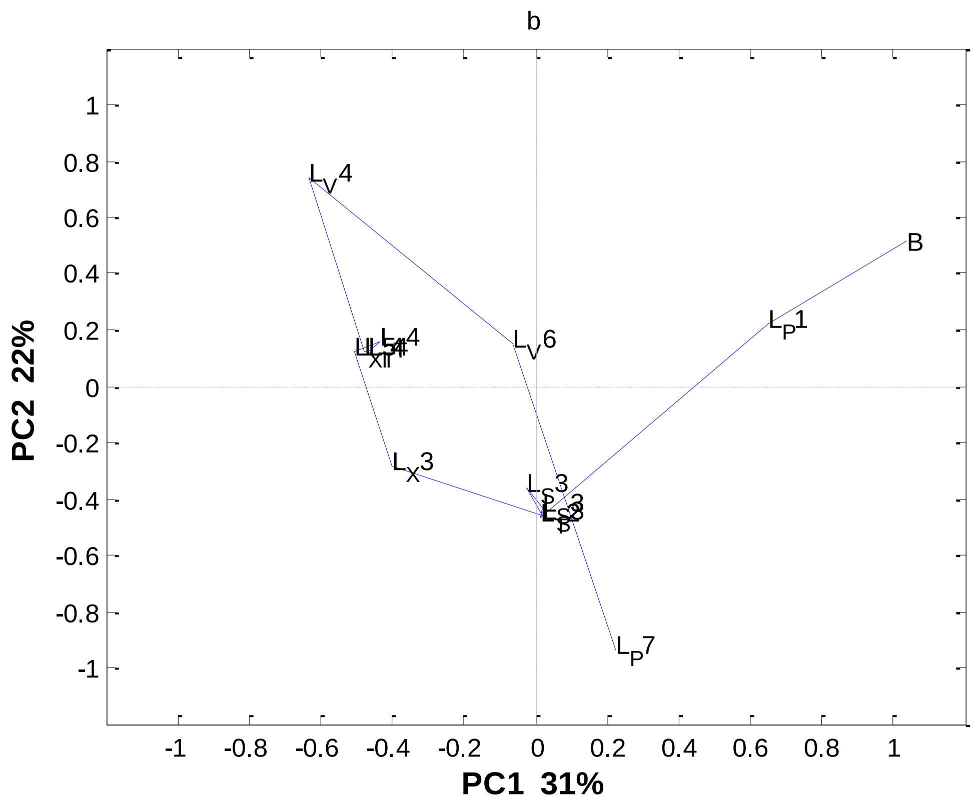


Figure 6.11 Plots of the two most significant principle components of the hidden layer activations of an asymmetrically trained MRN (a) trajectories of 15-length sequence BPTSSSXXTVPXVVP, (b) BPTSSSXXTTTVVP 14-length sequence that is incorrectly predicted using MRN.

Figure 6.11 depicted the trajectories of two different asymmetrical sequence lengths, (a) is 15-length sequence and (b) is the 14-length sequence. The prediction of the MRN was incorrect for both of them, in the embedded part; it was incorrect in the self-looping (state four) for (a) and the penultimate for (b). The other trajectories of the states were located as the correct ones. To investigate the ranges of state seven, four correct and incorrect sequences have been chosen and the ranges displayed for both of them.

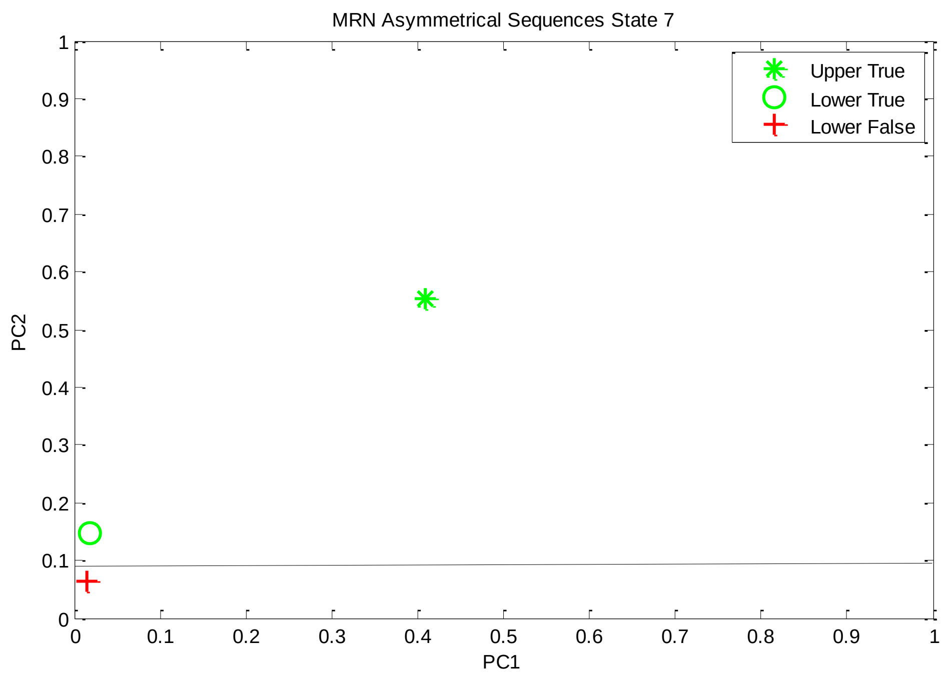


Figure 6.12 Range of state seven for both correctly predicted and incorrectly predicted penultimate symbols

It can be observed from Figure 6.12 that the MRN has ranges of the state seven for correctly predicted and incorrectly predicted symbols, which means that the hidden units organise the states of the grammar. Another example of this is shown in Figure 6.13 where a number of correctly and incorrectly predicted sequences have been selected from the lower embedded part of the grammar. It shows that for state four in the grammar, each range is located in approximately the same place but there is a small distance between them. That shows the difficulties of the hidden units to recognise this state.

This research has shown that, similar to the previous results, the MRN is capable of distributing the states in systematic way. State one of the lower part of the grammar is located in the first quarter and the upper part is in the fourth quarter of the plane. Moreover, state seven lower part is located in the fourth quarter and the upper part is in the third quarter of the plane. The internal representation shows small difference range between upper and lower parts of the grammar in respect of state four, where the network failed in some sequences to predict the state.

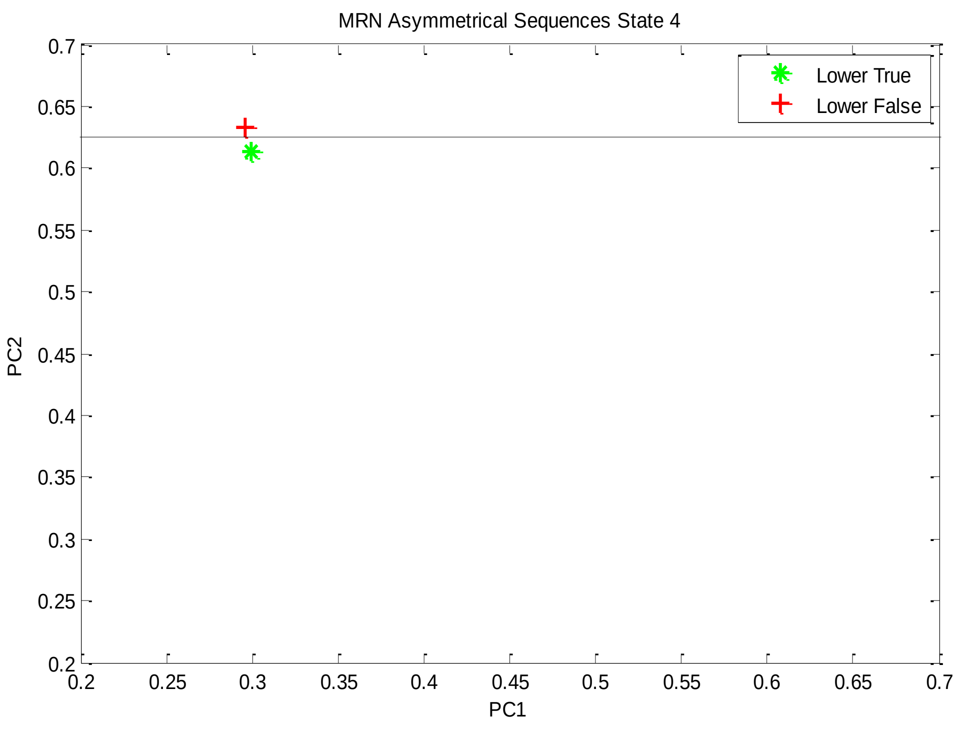


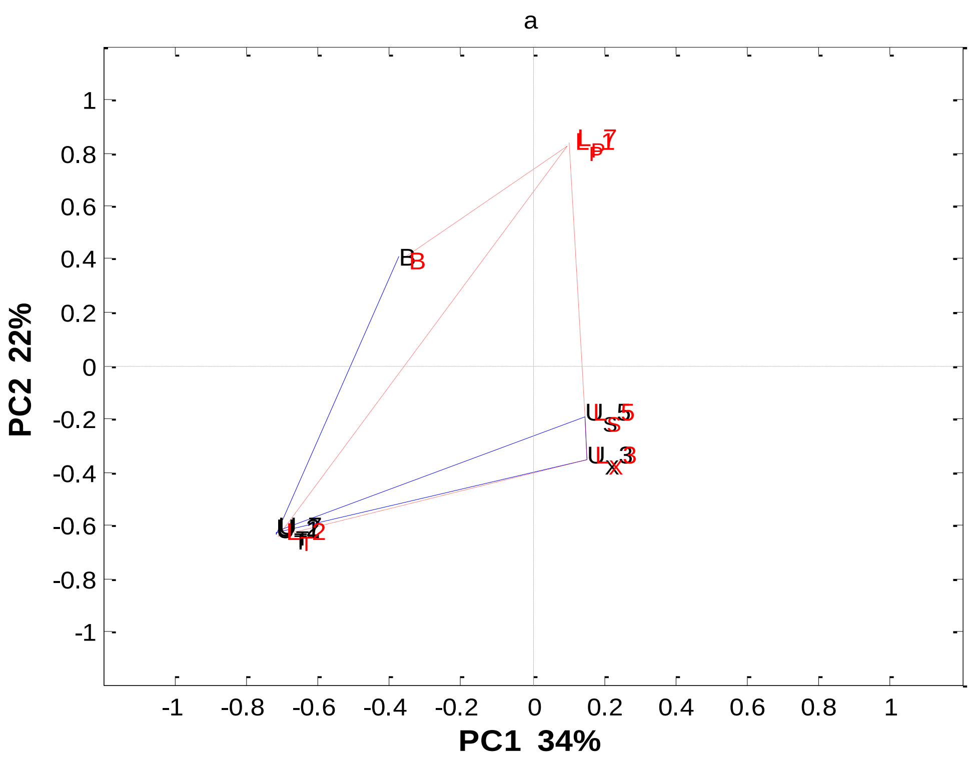
Figure 6.13 Range of sate four, (*) lower embedded sequence correctly predicted, (+) lower embedded sequence unpredicted sequences.

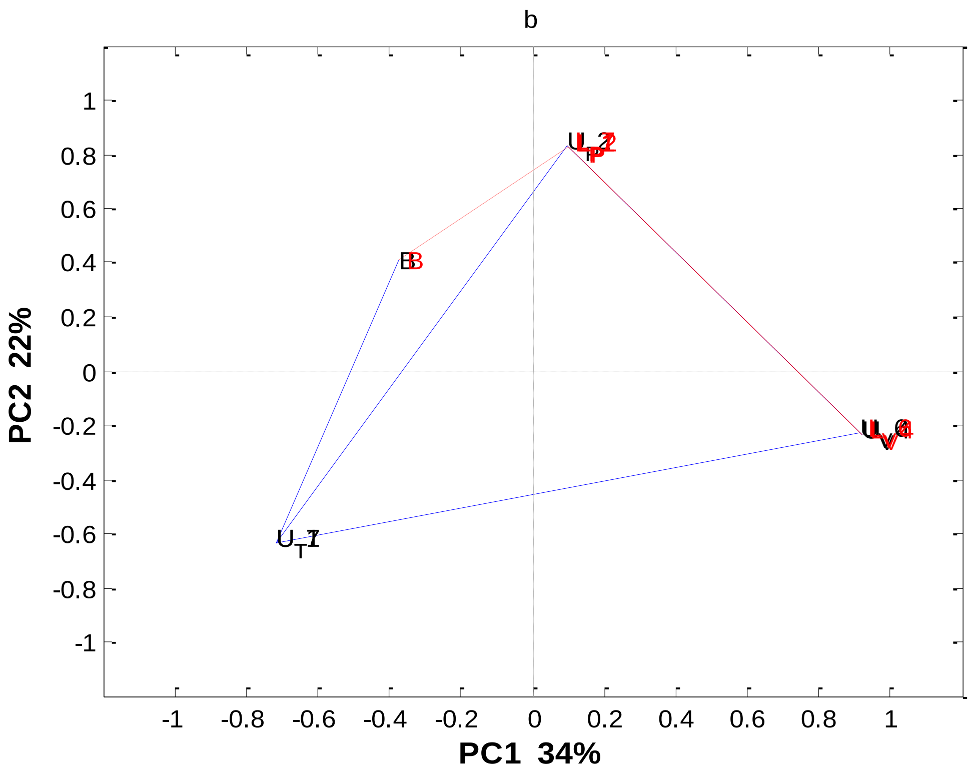
6.1.4 Internal Representations of the ESN

A variety of methods are used to assess the ESN. The data used in this investigation is from the ESN that has jumping connections and was used by Cartling, (2007). It is one of the standard methods described by Jaeger, 2002. The reservoir size of the network is such that it has 150 nodes. The following parameters were applied: 0.75 spectral radius; 0.85 connectivity; and 0.3 weight range. The representation of the embedded Reber grammar rule is investigated here using the 150 hidden units that were used in the training. The same sets of sequences that have been used in the previous network were used here. Table 5.9illustrates these sequences and the results of the ESN when tested with them. The trajectories of sequences numbered in the Table 6.9 1 to 4 and 13 in the (PC1, PC2) subspace are shown in the Figure 6.14. The principle divergence of several trajectories at each state in the sequences with a different route: upper leads to the first quarter and lower leads to the third quarter. The trajectories of the sequences from the SRN and ESN are dissimilar. However, this can explain why the ESN got 100% of the embedded parts of the sequences comparing with SRN and MRN, which were less than this percentage.

ESN using Asymmetrical sequences					
No	Length	Embed	Sequences	Predication	Reason for failure
1	6	U	BTTXST	T	
2		L	BPTXSP	T	
3		U	BTPVVT	T	
4		L	BPPVVP	T	
5	8	U	BTTXXVVT	F	P
6		L	BPTXXVVP	T	
7		U	BTTSSXST	T	
8		L	BPTSSXSP	F	P
9		U	BTPTVPST	T	
10		L	BPPTVPSP	F	P
11		U	BTPTTVVT	F	P
12		L	BPPTTVVP	T	
13	16		BPTXXTTTVPXTTVVP	T	
14	26		BPPVPXTVPXVPXVPXTVPXVPXVVP	T	

Table 6.9 Sequences results for ESN and the position of the unpredicted symbol





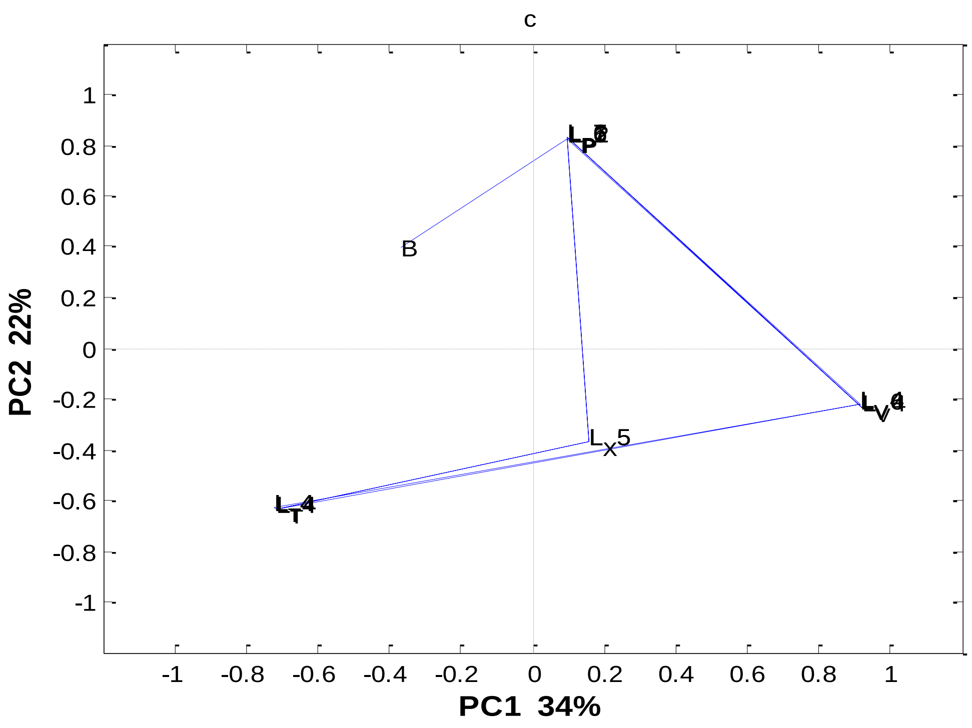
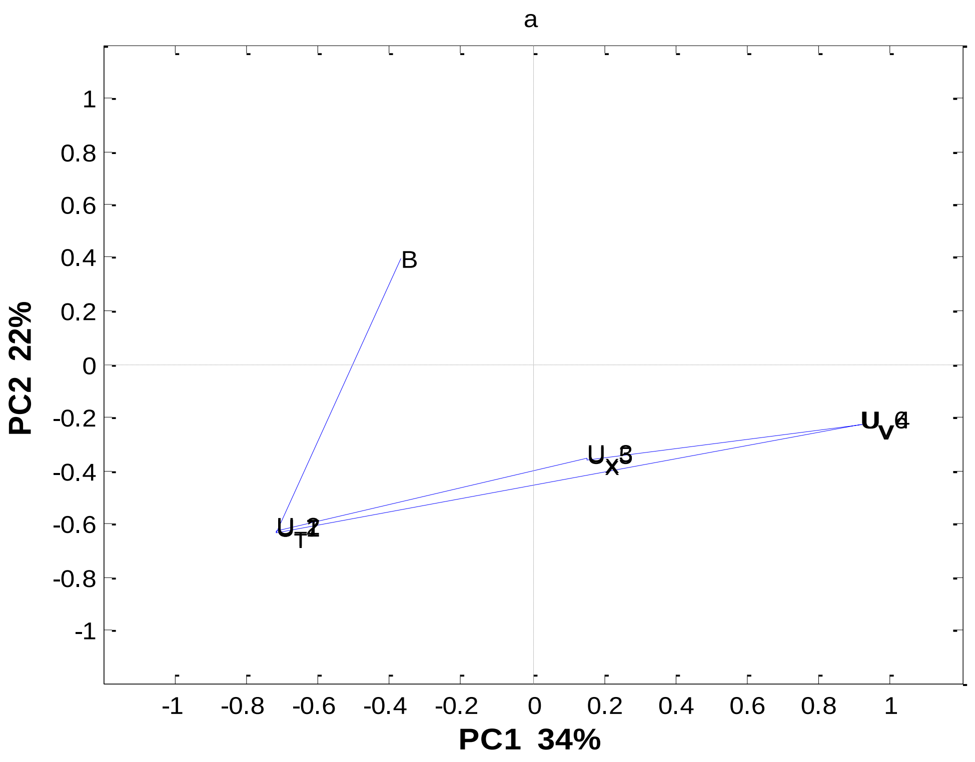


Figure 6.14 Plots of the two most significant principle components of the hidden layer activations of an asymmetrically trained ESN. (a) BTTXST/BPTXSP (b) BTPVVT/BPPVVP (c) BPPVPXTVPXTTTVVP. The trajectories are of sequences that were correctly predicted by the network.



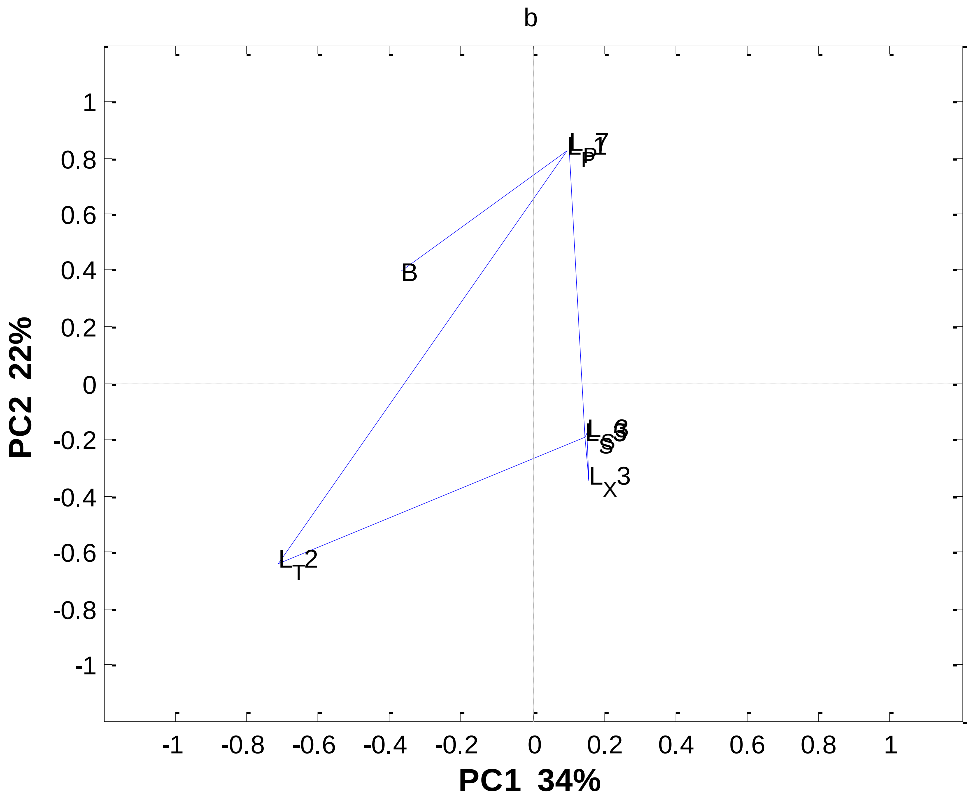


Figure 6.15 Plots of the two most significant principle components of the hidden layer activations of an asymmetrically trained ESN. (a) BTTXXVVT. (b) BPTSSXSP. The trajectories are of sequences that were incorrectly predicted by the network in the embedded part.

To investigate the sequences that were predicted incorrectly, a number of incorrectly predicted sequences have been selected to study how the network organized the penultimate symbols in the state of both the embedded upper and lower parts (since the results of the ESN show the incorrectly predicted symbol is generally the penultimate). Figure 6.15 shows the trajectories of the states of the sequences that have eight symbols. The trajectories of the penultimate symbol for sequences with both upper and lower embedded parts appear to be similar to the trajectories for the correctly predicted symbols. To investigate the dispersion of state seven, the range of this state has been computed for a number of sequences for both correctly and incorrectly predicted sequences. Therefore, comparing between the networks can explain why the MRN is superior over the other networks. Figure 6.16 draws the ranges of state 7 for a numbers of sequences. Asterisk and plus sign symbols represent the correctly predicted sequences and the circle and times signs are the incorrectly predicted ones.

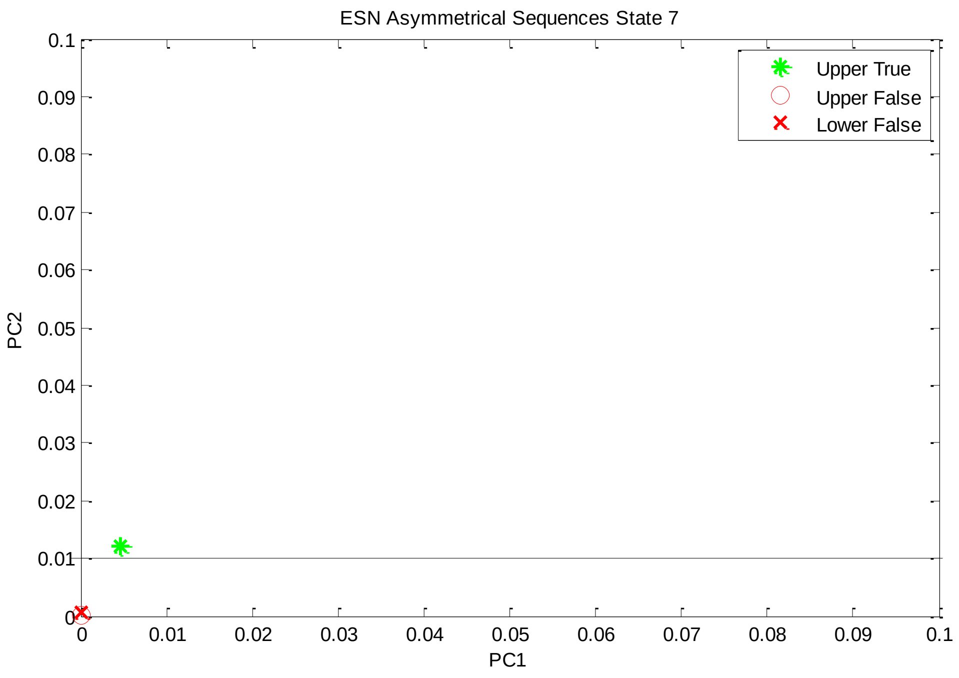


Figure 6.16 ESN: Range of state seven for both correctly and incorrectly predicted penultimate symbols

Figure 6.16shows the ranges when the surface is scaled where the predicted state is in a range differ from the state in the incorrectly predicted sequence. Nevertheless, the distance between them is still small where the previous networks provide a larger distance between them.

6.2 Comparative Analysis of the Internal Representations

Further investigation is required to determine exactly how the networks distribute the states of the grammar; test dataset results after applying PCA have been studied. Appendix D illustrates the results of the networks. The test sequences that have been applied to the PCA are unique sequences (not in the training dataset). 20.2% of the sequences were unique from the trained dataset of 212 sequences.

The prediction of the whole dataset	SRN	MRN	NARX	ESN
	50.94%	95.28%	51.88	48.58%

Table 6.10 The prediction of the unique sequences in the test dataset

Table 6.10 shows the total unique sequences correctly predicted for each network. The MRN is superior over the other networks in the unique sequences. Table 6.11 demonstrates the results of each network in detail according to the sequence length, for the correctly predicted unique sequences. The MRN also has the most successfully predicted sequences when comparing using sequences of increasing length. To investigate why MRN is superior over the networks, internal representation has been studied using PCA.

Length	Frequency	SRN	MRN	NARX	ESN
15	1	0	1	0	0
17	7	4	7	5	3
18	21	9	21	14	8
19	35	11	32	11	13
20	35	19	34	20	18
21	32	17	32	19	17
22	27	17	24	14	17
23	22	14	20	9	9
24	18	9	17	9	10
25	9	5	9	6	0
26	5	3	5	3	3

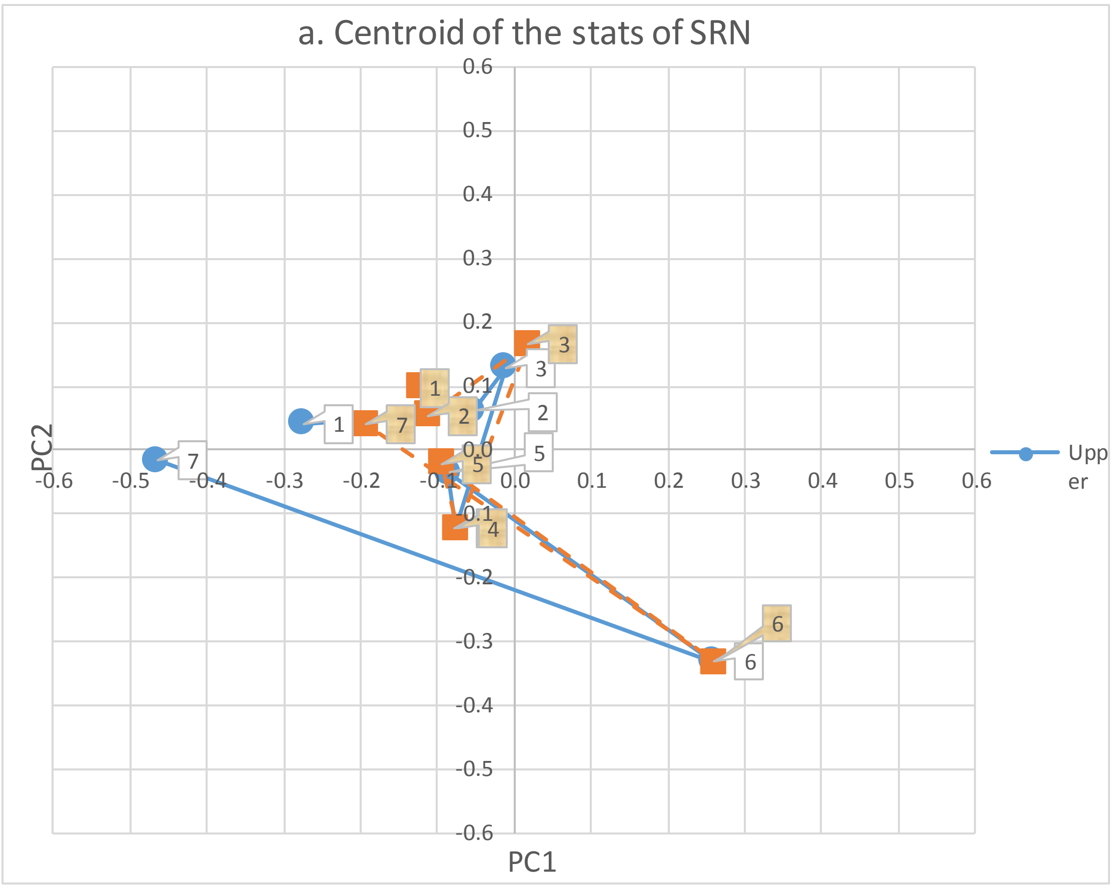
Table 6.11 Details of the number of each length sequence (unique) correctly predicted by each network

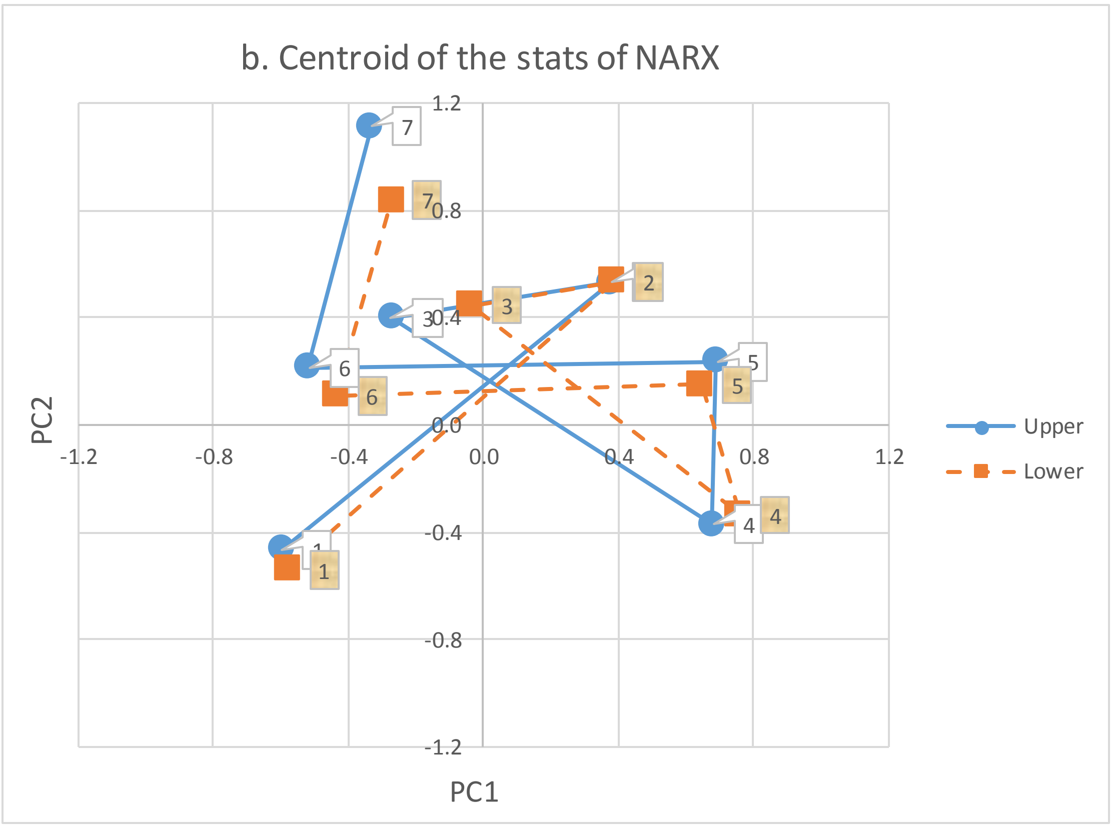
The unique sequences are studied in this investigation, since the rest of the sequences of the dataset are in the training dataset which was investigated in the previous section. To assess the distribution of the states, the centroid of the states was used for the upper and lower embedded grammar, thereby, observing the centre of the mass for each state of the grammar. Table 6.12, shows the centroid of each network for both upper and lower embedded routes through the grammar.

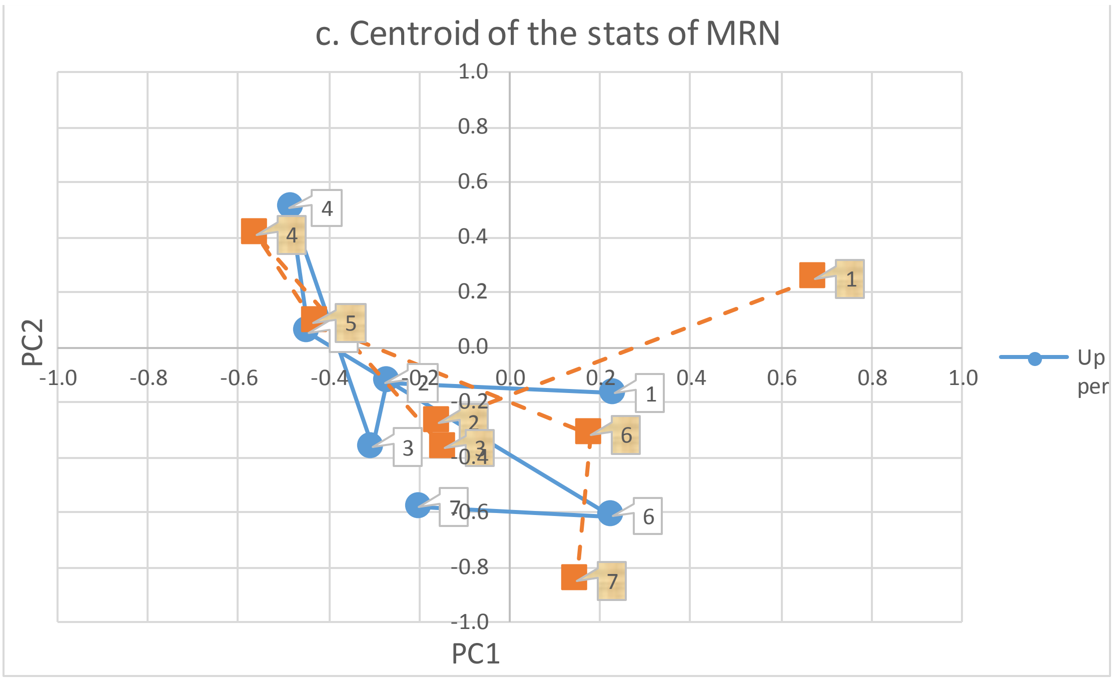
Grammar	States	NARX		SRN		MRN		ESN	
		PC1	PC2	PC1	PC2	PC1	PC2	PC1	PC2
Upper	1	-0.5970	-0.4645	-0.2754	0.0409	0.2321	-0.1679	-0.7176	-0.6375
	2	0.3789	0.5316	-0.0530	0.0621	-0.2708	-0.1281	-0.3976	-0.0541
	3	-0.2707	0.3995	-0.0124	0.1284	-0.3044	-0.3645	0.1507	-0.2720
	4	0.6792	-0.3690	-0.0759	-0.1227	-0.4832	0.5057	0.1835	-0.4095
	5	0.6912	0.2375	-0.0859	-0.0356	-0.4459	0.0523	0.1562	-0.3524
	6	-0.5177	0.2152	0.2595	-0.3311	0.2268	-0.6141	0.2221	0.6675
	7	-0.3321	1.1134	-0.4650	-0.0165	-0.2010	-0.5837	-0.7155	-0.6318
Lower	1'	-0.5776	-0.5398	-0.1220	0.0979	0.6745	0.2480	0.0965	0.8254
	2'	0.3845	0.5352	-0.1128	0.0536	-0.1596	-0.2747	-0.4235	-0.1244
	3'	-0.0346	0.4466	0.0176	0.1659	-0.1451	-0.3670	0.1507	-0.2714
	4'	0.7597	-0.3350	-0.0747	-0.1234	-0.5614	0.4093	0.1542	-0.4168
	5'	0.6473	0.1508	-0.0945	-0.0216	-0.4308	0.0900	0.1561	-0.3508
	6'	-0.4355	0.1098	0.2616	-0.3335	0.1776	-0.3180	0.2145	0.6773
	7'	-0.2703	0.8390	-0.1933	0.0395	0.1468	-0.8502	0.0988	0.8328

Table 6.12 The centroids for each grammar state for each network

The PC1 and PC2 values for all the unique sequences were calculated and the results are plotted in the Figure 6.17. The figure shows the range of the centroid of all the states for each network. The centroid of the SRN states are located approximately in the PC1 range of -0.5 to 0.3, NARX in -0.6 to 0.8, MRN in -0.6 to 0.7 and ESN in -0.8 to 0.3. This gives range of 0.8, 1.4, 1.3 and 1.1 for SRN, NARX, MRN and ESN respectively. This suggests a link may exist between the data distribution and the performance of the networks.


a. Centroid of the stats of SRN
PC1
PC2
Upper
1
2
3
4
5
6
7
-0.6
-0.5
-0.4
-0.3
-0.2
-0.1
0.0
0.1
0.2
0.3
0.4
0.5
0.6


b. Centroid of the stats of NARX
PC1
PC2
Upper
Lower
1
2
3
4
5
6
7
-1.2
-0.8
-0.4
0.0
0.4
0.8
1.2



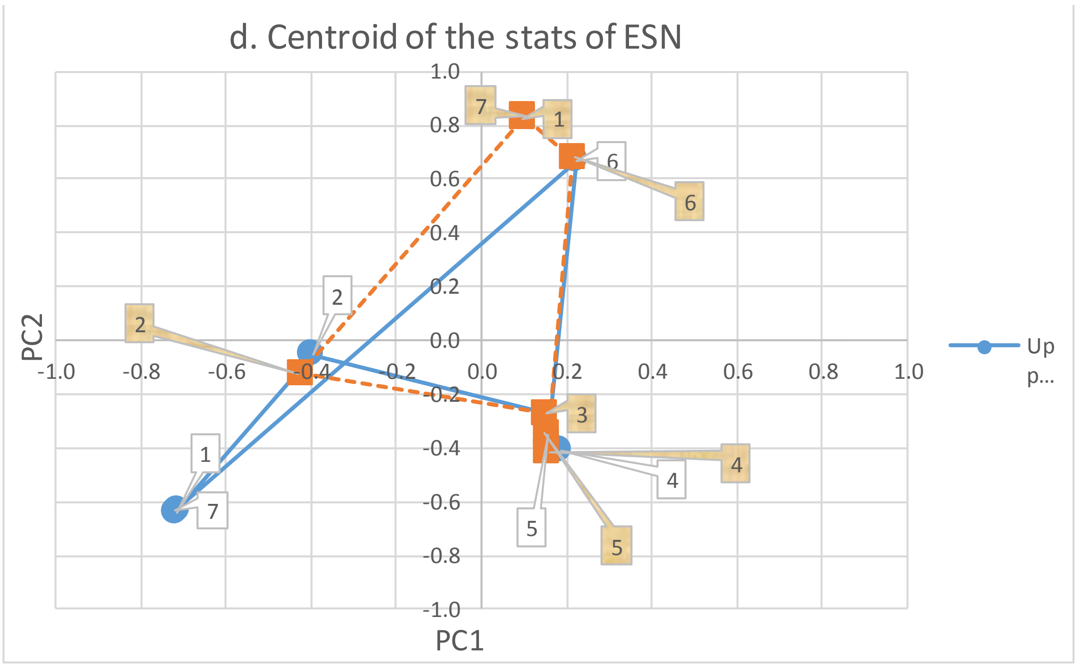


Figure 6.17 The centroid of the upper and lower embedded grammar of the networks a)SRN, b) NARX, c) MRN, d) ESN

Further statistical analysis of the centroid revealed strong links between the distribution of the dataset and the performance of the networks. Figure 6.17 depicted the centroid of each state of the networks represented by the PC1 and PC2. The graphs show the distance variance between sequences generated via the upper and lower routes through the grammar. In addition, it illustrates the range of each network where it shows the difference range represented by the hidden unit. The absolute difference between the centroids taking sequences containing the upper and lower sections of the embedded grammar respectively is given in Figure 6.18. It is apparent from this figure that the Euclidean distance between the states for the upper and lower embedded trajectories varies in these networks; the distance for the MRN is the highest of the networks followed by NARX. This explains why the MRN is superior over the other networks. The MRN is more able to consistently maintain a sufficient distance between corresponding grammar states within the upper and lower sub-grammars. This work contends that this is due to the ability of its sluggish state-based memory to latch onto and maintain information about the entry points (T and P) of the respective sub-grammars throughout the respective sequences.

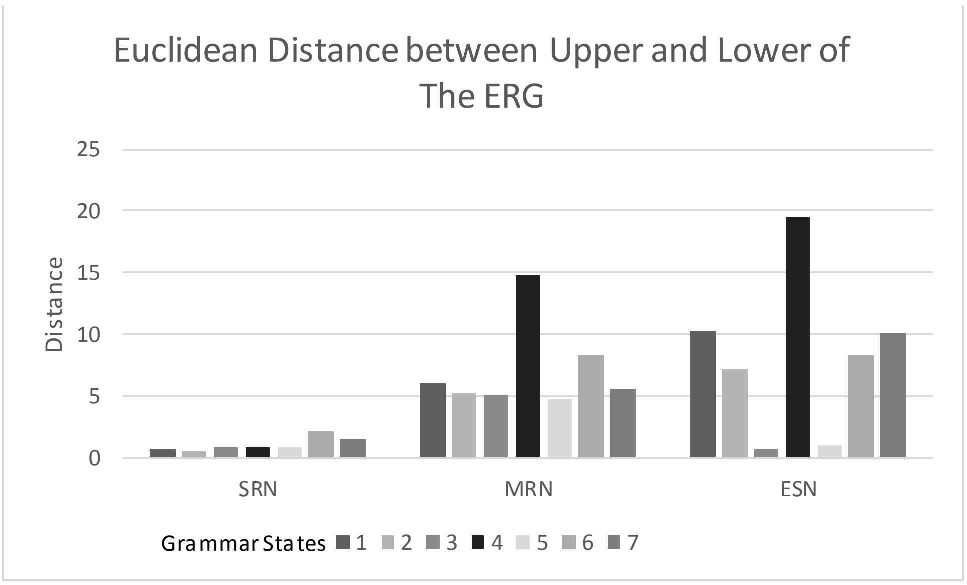


Figure 6.18 The Euclidean distances between each corresponding centroid representing the embedded grammar states of the upper and lower embedded grammars (for each model evaluated).

To explore the order of the networks for their performance, Euclidean distance has been calculated between the MRN and SRN, NARX and ESN. Table 6.1 illustrates the results obtained. The most interesting finding was that the NARX has the closest distance to the MRN at 0.33 followed by the SRN then ESN, which correlates with the performance of the networks.

Euclidean Distance			
Cross state	MRN vs SRN	MRN vs NARX	MRN vs ESN
1	0.049	0.050	0.142
2	0.028	0.202	0.065
3	0.178	0.107	0.162
4	0.038	0.091	0.005
5	0.253	0.115	0.159
6	0.282	0.164	1.212
7	0.167	0.094	0.318
all states	0.456	0.335	1.284

Table 6.13 Euclidean distance between MRN and the networks

6.3 Summary of Results and Discussion

The strong relationship between SRN, NARX, MRN and ESN has been reported in the literature. However, they differ from their architecture and training process. This study provides information about the internal representation of these networks. The high performance of the MRN in this research is traced to a well-organized internal representation of the grammatical elements. A number of architectures of networks facilitate an improved resolution of the internal representation, this is discussed above in terms of the intervening layers in each network architecture. The internal representation of the SRN in both biased and unbiased datasets, shows that the trajectories of the grammatical states are distributed in a systematic way which illustrates the important role of the hidden units in the network. Similar trajectories were acquired from the MRN with respect to their quarter positions in the PC1-PC2 plane. It seems significant that the trajectories mirror each other for the same embedded sections but from the upper and lower paths for the MRN and SRN, but for the ESN they look rather different. This is because the ESN had different learning algorithms compared with the rest of the networks where they use BPTT and all the connections were trainable. Another important finding was the constant variance between the upper and lower parts of the sequences represented by the internal representation of the MRN, which explain its superiority over the networks. The present study, however, makes several noteworthy contributions to connectionism. For the networks studied, their performance relies on how the internal representations of the networks maintains a constant variance between upper and lower part of the grammar. Investigation of the hidden units found that there is a kind of clustering represented by the hidden layer that helps the networks to recognise.

The internal representation, as explored by a principal component analysis of the hidden unit activities for entire networks, is shown to organize the states of the embedded Reber grammar. However, they differ when they organise state seven (the penultimate state) of the grammar; the ESN has difficulties in distinguishing between penultimate symbols of the grammar this may due to learning algorithm of the network.

Chapter 7

7. Conclusion & Future Work

7.1 Introduction

This thesis has focused on one of the principle problems in artificial intelligence, a problem that is still subject to ongoing research, despite approximately half a century of investigation by numerous researchers. Language acquisition is a complex problem to many linguists such that they consider it a paradox and a NP-complete problem. It is a perspective that consequently denies the possibility of an automated solution (Jackendoff 2002).

Nevertheless, nowadays, NLP systems are a key area of interest in the field of connectionism and much work has been conducted on how linguistic representations and descriptions can be used for processing. That is the focus of computational linguistics and NLP. This is formed by such models. The creation of an automated language acquisition system for natural language would be a revolutionary discovery because the complexity of its syntax and morphology is difficult to parse etc. One of the most significant debates within this field is that of empiricists, who argue that the brain has the neurological basis to discover an automated language acquisition model directly from exposure to naturally occurring sentences. The research shows that a particular class of connectionist networks, the multi-recurrent network, provides evidence in support of the empiricist's hypothesis, where other classes of connectionist network do not.

A principle objective of this research is to determine the class of RNNs that is able to robustly learn to represent the underlying pushdown automata that adequately describes the important characteristics considered essential for natural language acquisition, such as being able to establish and maintain cross-serial dependencies. This research has investigated several RNNs that are diverse in their network architectures and with variation in some cases in the learning algorithm. The study investigated the most common types of RNN models applied to language modelling tasks and variants thereof. In particular, the study applied the 'next symbol prediction' task for the Embedded Reber Grammar to a set of SRNs trained with back-propagation through time (BPTT), namely Elman's SRN, the Jordan net, NARX and the MRN. Although each of these networks shared the vanishing gradients problem associated with BPTT, they differed in their architecture and therefore how they may be able compensate, if at all, for the limitations of the learning algorithm. The performance of each of these networks was then contrasted against each other and to that of the current 'state-of-the-art' RNN, the ESN, which is not known to suffer from the vanishing gradient issue and has an echo-state property which allows for effective learning of temporal problems. A process of noise injection was also applied and evaluated to ascertain whether or not this optimised performance. Also, to support the networks in their task, asymmetrical data sets have been generated that gave statistical indications as to which embedded grammar was being entered into and therefore a bias towards the correct transition with which to exit the embedded aspect to reach the final state of the grammar.

The investigations found that not all RNNs examined were able to correctly model the grammar and process the associated long term dependencies. This Chapter therefore summarises the key findings and contribution of the research with respect to the type of RNN most able to correctly learn the grammar and generalise to the wider population of sequences. In addition, it provides insight into *why* a particular class of RNN was consistently better and then highlights important areas for future research.

Simulations with a class of MRNs using 4 memory banks and 10 hidden nodes per bank, showed that they were able to successfully learn 100% of the training sequences (with a maximum sequence length of 26 symbols). The MRN tested with the asymmetrical dataset was clearly able to detect the dependency, achieving a success rate of over 95.1% in five of the symmetrical test trials and over 88.5% in five of the asymmetrical test trails. However, the MRN's performance begins to significantly degrade as the sequence length increases above 60 symbols and fails entirely when the sequence length exceeds 100 symbols.

The performance of the MRN was achieved by using noise injection to the network training, however the SRN, NARX and ESN failed to improve their outcomes when the same method was used. A number of modifications were carried out to enhance the results of the networks (varying architectures, parameters, learning algorithm etc.).

7.2 Training Parameters

This section illustrates the parameters that are used to obtain the results with respect to the different networks. The hidden units used for SRN, Jordan, TDNN, and NARX were 15 and for MRN it was 10, the learning rate for those networks were 0.15. This was after testing a number of values; the momentum was 0.75 and the weight range was 0.3. The leaning type that enhanced the performance of the networks is “pattern error–sensitive” learning rate. The ESN has 150-reservoir size, 0.75 spectral radius, connectivity of 0.85 and weight range of 0.3. All networks were trained in one epoch.

7.3 Main Conclusion Derived from This Study

The investigations presented in this thesis are a novel attempt to answer research questions, both from a practical and a theoretical point of view. In particular, to evaluate the efficacy of an embedded memory architecture consisting of recurrent and self-recurrent units used in variants of the SRNs against the vanishing gradient problem associated with the gradient-descent learning algorithm. This research assesses the efficacy of this memory mechanism for the SRNs and ESN using a popular complex grammar induction task. Based on the results obtained from this research, it can be concluded that:

The investigations have studied the effect of the size of the hidden units on the performance of the networks. There is a limit of memory size above which the network outcomes start to decay. In addition, the investigation for the MRN found that going above four memory banks does not lead to enhanced performance. Rather, it is the number of neurons in the boxes that is important for the performance of the network and this is determined using the validation set. This is as stated by (Ulbricht 1994, Binner, Tino et al. 2010) and so it is confirmation of their work.

This research evaluated the stability of different network architectures to attain the ability to process long term dependency between different clauses within a sentence. This has been a key challenge for artificial neural networks such as the SRN and ESN. An artificial grammar, which replicates this problem was used to investigate a range of recurrent network architectures with gradient descent based learning such as SRN, NARX and MRN. Although, they have the same learning algorithm, they have differences in their architectures in terms of their natural feedback. These were compared with the new type of recurrent network called ESN with a one shot learning scheme and it is shown to be incapable to store this type of information to a high degree of accuracy. It is apparent that what neural networks need to represent in terms of a state machine is the linear boundary automata which is a finite state machine with a push-down stack. Therefore, it needs to be able to learn to represent the computational model, whenever the computational model needs to learn to represent a linear bounded automata to automate in order to process a grammar that has long-term dependency.

One aspect of the current study was to determine whether training with asymmetrical rather than symmetrical sequences could help the network learning. Asymmetrical sequences enhanced the learning of the networks, compared with symmetrical sequences.

In order to understand the quality of the state-based representations formed, the internal representation of each of the networks using PCA has been analysed to provide some visualisation of the trajectories through state space as a sequence is processed and therefore to ascertain whether this followed the underlying grammar, and if so, how robustly. PCA was performed on SRN, NARX, MRN and ESN because of their higher performance among the other networks.

It was noted that all models analysed had formed meaningful representations of the underlying grammar, where centroids within PCA space represented different states of the grammar. For SRN, MRN and ESN there was a clear mirroring effect within PCA space between the state space trajectories for the upper and lower embedded grammars. However, only the MRN was able to maintain sufficient distance between the representations of the two embedded grammars in order to consistently exit the embedded part to the correct exit point (grammar state 7). What do we mean by sufficient distance? Well, the research analysis showed that only the MRN maintained a constant variance between the distances of the centroids (grammar states) across all states of the embedded grammars. The NARX, SRN and ESN were all unable to form a constant variance between states. This strongly suggests that although the additional graded-state memory mechanism of the MRN requires additional weights, this is countered by the resulting quality and stability of the state-based representations formed i.e. it was able to more fully learn the training sequences and form hypotheses that provided robust generalisation to statistically biased and unbiased data.

The challenge of the long-term dependency meant that the embedded parts are correctly predicted while the long term dependency part is incorrectly memorised. The best architecture and learning algorithm to capture the long term dependency was the MRN in terms of the degree to which it learnt the problem and the degree to which it generalised to unique sequences beyond those found in the training set. In contrast, other networks such as ESN memorised or learned the embedded grammar but failed to adequately capture the long term dependency and therefore, to actually incorporate the embedded part correctly.

Throughout this research, the training method and training parameters were investigated to optimise these networks. Some key findings are:

Noise injection has enhanced the MRN performance by approximately 10% compared with when used without it. The range of the value of noise that provides the MRN with stability is ±0.01. Implementing this technique to the other networks did not enhance the networks; however, more investigations need to be conducted since more noise values need to be studied. The best values of the spectral radius and connectivity and the range weight were as mentioned previously. These values were arrived out using the ANOVA method and a range of training results with the ESN.

The results of this investigation show that the performance of the SRN and MRN are quite sensitive to the initial starting conditions comparing with the ESN. In addition to this, however, SRN, NARX and MRN have a similar learning algorithm; they have different feedback connections whereas the ESN is completely different in terms of learning algorithm and connections feedback.

Although in some instances in some paths of the grammar, some SRN and ESN have learned particular paths very well, in terms of the problem as a whole and across all paths of the grammar, the MRN is more constant in being able to correctly recognise sequences that are longer than and/or different from those in the training dataset, which shows a strong level of generation.

Analysing the internal representation of the networks using PCA demonstrated the systematic learning of different networks, particularly where sequences were the same in the embedded. This provided a mirroring between sequences in terms of those with upper and lower embedded sections which then diverged at state seven and these were compared using. The MRN captured the knowledge of the embedded grammar since it distributed the variance of the centroid for both upper and lower parts of the grammar in a constant way. This attribute was seen to a lesser extent in the NARX, followed by the SRN and then the ESN.

7.4 Future Research Directions.

Further investigations, in which future works could proceed, are listed below:

Although the MRN was superior over the other networks, further investigation is required to better understand the representations formed. For example, do MRNs allow for systematicity of language and structure? Can so-called ‘deep neural networks’ offer further representational power to the MRN? Since the grammar induction task involves learning grammar whose structure is potentially unknown, the model must be able to correctly predict the grammar to a reliable degree.

This work provides support for further exploration of the MRN for modelling human sentence processing and the associated computational machinery neutrally implemented within the brain (e.g. registers, counter functions, variable binding through temporal synchrony of neuronal firing). In particular, it would be interesting to investigate whether the generalization performance of the MRN when processing centre-embedded clauses is akin to that of human working memory when processing similar grammatical structures, building on the work of Cowan, 2001 and contrasting the MRN representations with those of the LSTM. Also, as the number of memory banks within the MRN, and thus the degree of granularity to which it integrates and stores past and current information, has to be pre-determined, how the number of memory banks can be automatically determined or learnt for a given prediction task could be explored.

Additional work also needs to be conducted to compare the MRN with long short term memory (LSTM) both from a theoretical and also an experimental point of view, to assess the LSTM against the MRN and ESN. Although LSTM has well counting facilities, to what extent language acquisition requires more complex machinery than counters and stacks could be assessed.

References

Aimetti, G., 2009. Modelling early language acquisition skills: Towards a general statistical learning mechanism. *In: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop,* Association for Computational Linguistics, pp. 1-9.

Al-Habaibeh, A., Zorriassatine, F. and Gindy, N., 2002. Comprehensive experimental evaluation of a systematic approach for cost effective and rapid design of condition monitoring systems using Taguchi's method. *Journal of Materials Processing Technology,* 124 (3), 372-383.

Allen, J. (1995) *NATURAL LANGUAGE UNDERSTANDING, 2nd edition, CA,* The Benjamin/Cummings Publishing Company Inc.

Alpsan, D., Towsey, M., Ozdamar, O., Tsoi, A.C. and Ghista, D.N., 1995. Efficacy of modified backpropagation and optimisation methods on a real-world medical problem. *Neural Networks,* 8 (6), 945-962.

Arnold, D., Balkan, L., Meijer, S., Humphreys, R. and Sadler, L. (1993) *Machine Translation: An Introductory Guide*. Manchester, Blackwell publishers.

Baker, J.K., 1979. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America,* 65, S132.

Bakker, B., 2001. Reinforcement Learning with Long Short-Term Memory. *In: NIPS,* pp. 1475-1482.

Batchelder, E.O., 2002. Bootstrapping the lexicon: A computational model of infant speech segmentation. *Cognition,* 83 (2), 167-206.

Beaufays, F., Bourlard, H., Franco, H. and Morgan, N. (2001) NEURAL NETWORKS IN AUTOMATIC SPEECH RECOGNITION. *In: M. Arbib (ed) The Handbook of Brain Theory and Neural Networks, 2nd Edition*. Bradford Books.

Becerikli, Y., Konar, A.F. and Samad, T., 2003. Intelligent optimal control with dynamic neural networks. *Neural Networks,* 16 (2), 251-259.

Bengio, Y., Simard, P. And Frasconi, P., 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on,* **5**(2), pp. 157-166.

Bengio, Y., Simard, P. and Frasconi, P., 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on,* 5 (2), 157-166.

Binner, J.M., Tino, P., Tepper, J., Anderson, R., Jones, B. and Kendall, G., 2010. Does money matter in inflation forecasting? *Physica A: Statistical Mechanics and its Applications,* 389 (21), 4793-4808.

Bishop, C.M., 1995. Training with noise is equivalent to Tikhonov regularization. *Neural Computation,* 7 (1), 108-116.

Blank, D.S., Meeden, L.A. and Marshall, J.B., 1992. Exploring the symbolic/subsymbolic continuum: A case study of RAAM. *The Symbolic and Connectionist Paradigms: Closing the Gap,* 113, 148.

Boné, R., Crucianu, M. and Asselin de Beauville, J., 2002. Learning long-term dependencies by the selective addition of time-delayed connections to recurrent neural networks. *Neurocomputing,* 48 (1), 251-266.

Brown, R., 1973. *A first language: The early stages.* Harvard U. Press.

Buehner, M., and Young, P., 2006. A tighter bound for the echo state property. *IEEE Transactions on Neural Networks,* 17 (3), 820-824.

Bullinaria, J., 1997. Analyzing the internal representations of trained neural networks. *Neural Network Analysis, Architectures and Algorithms,* , 3-26.

Cammarota, M., Bevilaqua, L.R.M., Rossato, J.I., Ramirez, M., Medina, J.H. and Izquierdo, I., 2005. Relationship between short- and long-term memory and short- and long-term extinction. *Neurobiology of Learning and Memory,* 84 (1), 25-32.

Cartling, B., 2008. On the implicit acquisition of a context-free grammar by a simple recurrent neural network. *Neurocomputing,* 71 (7-9), 1527-1537.

Čerňanský, M., Makula, M. and Beňušková, Ľ., 2007. Organization of the state space of a simple recurrent network before and after training on recursive linguistic structures. *Neural Networks,* 20 (2), 236-244.

Chalup, S.K., and Blair, A.D., 2003. Incremental training of first order recurrent neural networks to predict a context-sensitive language. *Neural Networks,* 16 (7), 955-972.

Chater, N., and Manning, C.D., 2006. Probabilistic models of language processing and acquisition. *Trends in Cognitive Sciences,* 10 (7), 335-344.

Chen, S., Billings, S. and Grant, P., 1990. Non-linear system identification using neural networks. *International Journal of Control,* 51 (6), 1191-1214.

Chen, T., Lin, K.H.C. and Soo, V., 1997. Training recurrent neural networks to learn lexical encoding and thematic role assignment in parsing Mandarin Chinese sentences. *Neurocomputing,* 15 (3-4), 383-409.

Chomsky, N., 1959. On certain formal properties of grammars. *Information and Control,* 2 (2), 137-167.

Chomsky, N., and Halle, M., 1968. The sound pattern of English. New York, NY: Harper and Row.

Christiansen, M.H., and Curtin, S.L., 1999. The power of statistical learning: No need for algebraic rules. *In: Proceedings of the 21st annual conference of the Cognitive Science Society,* Citeseer, pp. 119.

Cleeremans, A., and Dienes, Z., 2008. Computational models of implicit learning. *Cambridge Handbook of Computational Psychology,* , 396-421.

Cleeremans, A., Destrebecqz, A. and Boyer, M., 1998. Implicit learning: news from the front. *Trends in Cognitive Sciences,* 2 (10), 406-416.

Cleeremans, A., Servan-Schreiber, D. and McClelland, J.L., 1989. Finite state automata and simple recurrent networks. *Neural Computation,* 1 (3), 372-381.

Conway, C.M., Bauernschmidt, A., Huang, S.S. and Pisoni, D.B., 2010. Implicit statistical learning in language processing: Word predictability is the key. *Cognition,* 114 (3), 356-371.

Corrigan, R. and Iverson, G. (eds) (1994) *The reality of linguistic rules.* Amsterdam, Benjamins.

Cowan, N., 2001. The magical number 4 in short-term memory: A reconsideration of mental storage capacity. Behavioral and Brain Sciences, 24:87–185.

Cramer, B., 2007. Limitations of current grammar induction algorithms. *In: Proceedings of the 45th annual meeting of the ACL: student research workshop,* Association for Computational Linguistics, pp. 43-48.

Craven, M., and Shavlik, J.W., 1994. Using Sampling and Queries to Extract Rules from Trained Neural Networks. *In: ICML,* Citeseer, pp. 37-45.

D'Ulizia, A., Ferri, F. and Grifoni, P., 2011. A survey of grammatical inference methods for natural language learning. *Artificial Intelligence Review,* 36 (1), 1-27.

De Albuquerque, Victor Hugo C, de Alexandria, A.R., Cortez, P.C. and Tavares, J.M.R., 2009. Evaluation of multilayer perceptron and self-organizing map neural network topologies applied on microstructure segmentation from metallographic images. *NDT & E International,* 42 (7), 644-651.

Deliang, W., Xiaomei, L. And Ahalt, S.C., 1996. On temporal generalization of simple recurrent networks. *Neural Networks,* **9**(7), pp. 1099-1118.

Deliang, W., Xiaomei, L. and Ahalt, S.C., 1996. On temporal generalization of simple recurrent networks. *Neural Networks,* 9 (7), 1099-1118.

Deng, R., and Fox, M.D., 2007. Parametric optimization for EPGVF snake using ANOVA and Taguchi method. *In: Bioengineering Conference, 2007. NEBC'07. IEEE 33rd Annual Northeast,* IEEE, pp. 108-109.

Diaconescu, E., 2008. The use of NARX neural networks to predict chaotic time series. *WSEAS Transactions on Computer Research,* 3 (3), 182-191.

Dienes, Z., 1992. Connectionist and memory-array models of artificial grammar learning. *Cognitive Science,* 16 (1), 41-79.

Dienes, Z., Altmann, G.T.M. and Gao, S., 1999. Mapping across domains without feedback: A neural network model of transfer of implicit knowledge. *Cognitive Science,* 23 (1), 53-82.

Dienes, Z., Broadbent, D. and Berry, D.C., 1991. Implicit and explicit knowledge bases in artificial grammar learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition,* 17 (5), 875.

Doey, T., 2008. Child Language Acquisition and Growth. *Journal of the Canadian Academy of Child and Adolescent Psychiatry,* 17 (3), 163.

Dorffner, G., 1996. Neural networks for time series processing. *In: Neural Network World,* Citeseer.

Dror, G., MacLeod, C. and Maxwell, G., Training artificial neural networks using Taguchi methods.

Du, K.-., 2010. Clustering: A neural network approach. *Neural Networks,* 23 (1), 89-107.

Elman, J.L., 1990. Finding structure in time. *Cognitive Science,* 14 (2), 179-211.

Elman, J.L., 1993. Learning and development in neural networks: The importance of starting small. *Cognition,* 48 (1), 71-99.

Elman, J.L., 1995. Language as a dynamical system. *Mind as Motion: Explorations in the Dynamics of Cognition,* , 195-223.

Elman, J.L., 2001. Connectionism and language acquisition. *Language Development: The Essential Readings,* , 295-306.

Frank. A. Buckless and S. P. Ravenscroft, "Contrast coding: A refinement of ANOVA in behavioral analysis," *Accounting Review*, pp. 933-945, 1990.

Fred, Karlsson, “Working Memory Constraints on Multiple Center-Embedding Definition of Center-Embedding,” *Traffic*, pp. 2045–2050, 2007.

Fahlman, S.E., 1991. The recurrent cascade-correlation architecture.

Farkaš, I. And Crocker, M.W., 2008. Syntactic systematicity in sentence processing with a recurrent self-organizing network. *Neurocomputing*, **71**(7-9), pp. 1172-1179.

Farkaš, I., and Crocker, M.W., 2008. Syntactic systematicity in sentence processing with a recurrent self-organizing network. *Neurocomputing*, 71 (7-9), 1172-1179.

Feldman, J., and Howell, S.R., PDP and Structured Connectionism: An Integrated View on Language Acquisition.

Fitch, W.T., and Hauser, M.D., 2004. Computational constraints on syntactic processing in a nonhuman primate. *Science*, 303 (5656), 377-380.

Fodor, J.A., and Pylyshyn, Z.W., 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28 (1), 3-71.

Frank, S.L., 2006. Learn more by training less: systematicity in sentence processing by recurrent networks. *Connection Science*, 18 (3), 287-302.

French, R.M., 1992. Semi-distributed representations and catastrophic forgetting in connectionist networks. *Connection Science*, **4**(3-4), pp. 365-377.

French, R.M., 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3 (4), 128-135.

Gabrijel, I., and Dobnikar, A., 2003. On-line identification and reconstruction of finite automata with generalized recurrent neural networks. *Neural Networks*, 16 (1), 101-120.

Gallagher, M., and Downs, T., 2003. Visualization of learning in multilayer perceptron networks using principal component analysis. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on,* 33 (1), 28-34.

Gathercole SE. Nonword repetition and word learning: The nature of the relationship. *Applied Psycholinguistics* 2006;27(4):513-543. Gers, F.A., and Schmidhuber, E., 2001. LSTM recurrent networks learn simple context-free and context-sensitive languages. *Neural Networks, IEEE Transactions on,* 12 (6), 1333-1340.

Gers, F., 2001. Long Short-Term Memory in Recurrent Neural Networks. *Lausanne, EPFL.*

Gers, F.A., and Schmidhuber, E., 2001. LSTM recurrent networks learn simple context-free and context-sensitive languages. *Neural Networks, IEEE Transactions on,* 12 (6), 1333-1340.

Girosi, F., Jones, M. and Poggio, T., 1995. Regularization theory and neural networks architectures. *Neural Computation,* 7 (2), 219-269.

Gluck, M.A., and Myers, C.E., 2001. *Gateway to memory: An introduction to neural network modeling of the hippocampus and learning.* The MIT Press.

Gold, E. (1967) Language Identification in the Limit. *Information and Control,* 16, 447-474.

Gopalsamy, B.M., Mondal, B. and Ghosh, S., 2009. Taguchi method and ANOVA: An approach for process parameters optimization of hard machining while machining hardened steel. *Journal of Scientific & Industrial Research,* 68 (8), 686-695.

Gordon, P., 2004. Numerical cognition without words: Evidence from Amazonia. *Science,* 306 (5695), 496-499.

Goschke, T., and Bolte, A., 2007. Implicit Learning of Semantic Category Sequences: Response-Independent Acquisition of Abstract Sequential Regularities. *Journal of Experimental Psychology: Learning, Memory, and Cognition,* 33 (2), 394-406.

Grishman, R. (1986). Computational Linguistics: an introduction. Cambridge University Press.

Grüning, A., 2007. Elman backpropagation as reinforcement for simple recurrent networks. *Neural Computation,* 19 (11), 3108-3131.

Gupta, L., and McAvoy, M., 2000. Investigating the prediction capabilities of the simple recurrent neural network on real temporal sequences. *Pattern Recognition,* 33 (12), 2075-2081.

Gupta, L., Mcavoy, M. And Phegley, J., 2000. Classification of temporal sequences via prediction using the simple recurrent neural network. *Pattern Recognition,* **33**(10), pp. 1759-1770.

Hammerton, J., 2001. Clause identification with long short-term memory. *In: Proceedings of the 2001 workshop on Computational Natural Language Learning-Volume 7,* Association for Computational Linguistics, pp. 22.

Hammerton, J., 2003. Named entity recognition with long short-term memory. *In: Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4,* Association for Computational Linguistics, pp. 172-175.

Hannon, E. E. & Trehub, S. E. (2005), 'Tuning in to musical rhythms: Infants learn more readily than adults', *PNAS* **102**(35), 12639-12643.Henderson, J., and Lane, P., 1998. A connectionist architecture for learning to parse. *In: Proceedings of the 17th international conference on Computational linguistics-Volume 1,* Association for Computational Linguistics, pp. 531-537.

Henrik Jacobsson, Rule extraction from recurrent neural networks: a taxonomy and review, Neural Comput. 17 (2005) 1223–1263.

Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R.R., 2012. Improving neural networks by preventing co-adaptation of feature detectors. *ArXiv Preprint arXiv:1207.0580.*

Hochreiter, S., and Schmidhuber, J., 1997. Long short-term memory. *Neural Computation,* 9 (8), 1735-1780.

Hoff, E., 2009. Language development at an early age: learning mechanisms and outcomes from birth to five years. *Encyclopedia on Early Childhood Development,* , 1-5.

Horning, J. (1969) *A study of grammatical inference*. PhD thesis, Stanford University, California.Hwa, R., 2000. Sample selection for statistical grammar induction. *In: Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13,* Association for Computational Linguistics, pp. 45-52.

J. A. P�rez-Ortiz, F. A. Gers, D. Eck, and J. Schmidhuber, "Kalman filters improve LSTM network performance in problems unsolvable by traditional recurrent nets," *Neural Networks*, vol. 16, no. 2, pp. 241–250, 2003.

Jackendoff, R., 2002. *Foundations of language: Brain, meaning, grammar, evolution.* Oxford University Press, USA.

Jackson, D., Constandse, R. and Cottrell, G. (1996) Selective attention in the acquisition of the past tense. *In: Proceedings of the 18th Annual Conference of the Cognitive Science Society*. Hillsdale, NJ. p. 183-188.

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.

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

Jaeger, H., 2003. Adaptive nonlinear system identification with echo state networks. *Networks,* 8, 9.

Jaeger, H., 2007. Discovering multiscale dynamical features with hierarchical echo state networks. *Jacobs University Bremen, Tech.Rep.*

Jaeger, H., Lukoševičius, M., Popovici, D. and Siewert, U., 2007. Optimization and applications of echo state networks with leaky- integrator neurons. *Neural Networks*, 20 (3), 335-352.

Jagota, A., Lyngsø, R.B. and Pedersen, C.N., 2001. Comparing a hidden Markov model and a stochastic context-free grammar. *In:* Comparing a hidden Markov model and a stochastic context-free grammar. *Algorithms in Bioinformatics.* Springer, 2001, pp. 69-84.

James, D.S.S.A.C., and McClelland, L., 1988. Encoding Sequential Structure In Simple Recurrent Networks. CMU-CS-88- 183. Carnegie Mellon University, Computer Science Department, Pittsburgh.

James. Hammerton, "Named entity recognition with long short-term memory," in *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003-Volume 4,* 2003, pp. 172-175.

Jiang, C., and Song, F., 2010. Forecasting chaotic time series of exchange rate based on nonlinear autoregressive model. *In: Advanced Computer Control (ICACC), 2010 2nd International Conference on,* IEEE, pp. 238-241.

Jin, Y., and Geman, S., 2006. Context and hierarchy in a probabilistic image model. *In: Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on,* IEEE, pp. 2145-2152.

Johnson, K., 2004. Gold's Theorem and Cognitive Science*. *Philosophy of Science,* 71 (4), 571-592.

Jonker, J., 2007. Grammar induction and PP attachment disambiguation.

Jordan, M., 1986. Attractor dynamics and parallelism in a sequential connectionist machine. *In: Proceedings of 9th Annual Conference of Cognitive Science Society,* pp. 531-546.

Kalman, B.L., and Kwasny, S.C., 1997. High performance training of feedforward and simple recurrent networks. *Neurocomputing,* 14 (1), 63-83.

Khaw, J.F., Lim, B. and Lim, L.E., 1995. Optimal design of neural networks using the Taguchi method. *Neurocomputing,* 7 (3), 225-245.

Kolen, J.F., 1994. Fool's gold: Extracting finite state machines from recurrent network dynamics. *Advances in Neural Information Processing Systems,* , 501-501.

Kolen, J.F., and Kremer, S.C., 2001. *A field guide to dynamical recurrent networks.* John Wiley & Sons.

Kolen, J.F., and Pollack, J.B., 1990. Back Propagation is Sensitive to Initial Conditions. *In: NIPS,* pp. 860-867.

Koskela, T., Varsta, M., Heikkonen, J. and Kaski, K., 1998. Temporal sequence processing using recurrent SOM. *In: Knowledge-Based Intelligent Electronic Systems, 1998. Proceedings KES'98. 1998 Second International Conference on,* IEEE, pp. 290-297.

Lachter, J., and Bever, T.G., 1988. The relation between linguistic structure and associative theories of language learning—A constructive critique of some connectionist learning models. *Cognition,* 28 (1), 195-247.

Lari, K., and Young, S.J., 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech & Language,* 4 (1), 35-56.

Lawrence, S., Giles, C.L. and Fong, S., 2000. Natural language grammatical inference with recurrent neural networks. *Knowledge and Data Engineering, IEEE Transactions on,* 12 (1), 126-140.

Lewis, J.D., and Elman, J., 2001. Learnability and the statistical structure of language: Poverty of stimulus arguments revisited. *In: Proceedings of the 26th annual Boston University conference on language development,* Citeseer, pp. 359-370.

Lin, T. & Meador,J.,1992. Classification-accuracy monitored backpropagation, Pp. 1553-1556.

Lin, T., Horne, B.G., Tino, P. and Giles, C.L., 1996. Learning long-term dependencies in NARX recurrent neural networks. *Neural Networks, IEEE Transactions on,* 7 (6), 1329-1338.

Lin, X., Yang, Z. and Song, Y., 2009. Short-term stock price prediction based on echo state networks. *Expert System with Applications, 36 (3, Part 2), 7313-7317.*

Lo, Y., and Tsao, C., 2002. Integrated Taguchi method and neural network analysis of physical profiling in the wirebonding process. *Components and Packaging Technologies, IEEE Transactions on,* 25 (2), 270-277.

Lubell-Doughtie, P., 2010. Using Echo State Networks to Count without a Counter.

Luce, R.D., 1963. Detection and recognition. *Handbook of Mathematical Psychology,* 1, 103-189.

LukošEvičIus, M., and Jaeger, H., 2009. Survey: Reservoir computing approaches to recurrent neural network training. *Computer Science Review,* 3 (3), 127-149.

Lust, B., 2006. *Child language.* Cambridge University Press.

M. H. Tong, A. D. Bickett, E. M. Christiansen, and G. W. Cottrell, "Learning grammatical structure with Echo State Networks.," *Neural Netw.*, vol. 20, no. 3, pp. 424–32, Apr. 2007.

Ma, S., and Ji, C., 1998. Fast training of recurrent networks based on the EM algorithm. *Neural Networks, IEEE Transactions on,* 9 (1), 11-26.

Mabbutt, S., Picton, P., Shaw, P. and Black, S., 2012. Review of Artificial Neural Networks (ANN) applied to corrosion monitoring. *In: Journal of Physics: Conference Series,* IOP Publishing, pp. 012114.

MacKay, D.J., 1995. Probable networks and plausible predictions-a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems,* 6 (3), 469-505.

Macleod, C., Dror, G. and Maxwell, G., 1999. Training artificial neural networks using Taguchi methods. *Artificial Intelligence Review*, 13 (3), 177-184.

MacWhinney, B (2004) A multiple process solution to the logical problem of language acquisition. *Journal of Child Language*, 31, 883-914.

Mamdani, E., Østergaard, J. and Lembessis, E., 1983. Use of fuzzy logic for implementing rule-based control of industrial processes. *In:* Use of fuzzy logic for implementing rule-based control of industrial processes. *Advances in Fuzzy Sets, Possibility Theory, and Applications.* Springer, 1983, pp. 307-323.

Manning, C.D., and Schütze, H., 1999. *Foundations of statistical natural language processing.* MIT press.

Marcus, G.F., 1998. Can connectionism save constructivism? *Cognition*, 66 (2), 153-182.

Marcus, G.F., 1999. Language acquisition in the absence of explicit negative evidence: can simple recurrent networks obviate the need for domain-specific learning devices? *Cognition*, 73 (3), 293-296.

Marcus, G.F., 2003. *The algebraic mind: Integrating connectionism and cognitive science.* The MIT Press.

Marcus, G.F., Vijayan, S., Rao, S.B. and Vishton, P.M., 1999. Rule learning by seven-month-old infants. *Science*, 283 (5398), 77-80.

Mareschal, D., and Shultz, T.R., 1996. Generative connectionist networks and constructivist cognitive development. *Cognitive Development*, 11 (4), 571-603.

Marques, F., Souza, L.F., Rebolho, D., Caporali, A., Belo, E. and Ortolan, R., 2005. Application of time-delay neural and recurrent neural networks for the identification of a hingeless helicopter blade flapping and torsion motions. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 27, 97.

Matsuoka, K., 1992. Noise injection into inputs in back-propagation learning. *Systems, Man and Cybernetics, IEEE Transactions on*, 22 (3), 436-440.

McQueen, T., Hopgood, A.A., Allen, T.J. and Tepper, J.A., 2005. Extracting finite structure from infinite language. *Knowledge-Based Systems,* 18 (4-5), 135-141.

Medsker, L.R., and Jain, L.C., 2000. *Recurrent neural networks design and applications.* USA: CRC Press LLC.

Menezes Jr, José Maria P, and Barreto, G.A., 2008. Long-term time series prediction with the NARX network: An empirical evaluation. *Neurocomputing,* 71 (16), 3335-3343.

Murphy, K., 2001. An introduction to graphical models. *Rap.Tech.*

Naseem, T., Chen, H., Barzilay, R. and Johnson, M., 2010. Using universal linguistic knowledge to guide grammar induction. *In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing,* Association for Computational Linguistics, pp. 1234-1244.

Newell, A., 1994. *Unified theories of cognition.* Harvard University Press.

Noel Sharkey, Amanda Sharkey and Stuart Jackson, 2000. Are SRNs Sufficient for Modeling Language Acquisition? *In:* Are SRNs Sufficient for Modeling Language Acquisition? Oxford University Press, 2000, pp. 33-54.

Noris, B., Nobile, M., Piccini, L., Berti, M., Mani, E., Molteni, M., Keller, F., Campolo, D. and Billard, A.G., 2008. P2.104 Gait analysis of autistic children with Echo State Networks. *Parkinsonism & Related Disorders,* 14 (Supplement 1), S70-S70.

O'connell, T.C., 1995. *Using Periodically Attentive Units to Extend the Temporal Capacity of Simple Recurrent Networks.*

Oja, E., 1982. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology,* 15 (3), 267-273.

Omlin, C., 2001. Understanding and explaining DRN behaviour. *A Field Guide to Dynamical Recurrent Networks.*

Palmer-Brown, D., Tepper, J.A. and Powell, H.M., 2002. Connectionist natural language parsing. *Trends in Cognitive Sciences,* 6 (10), 437-442.

Papadatou-pastou, M., 2011. Are connectionist models neurally plausible? A critical appraisal. *Encephalos,* 48 (1), 5-12.

Pearson, K., 1901. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science,* 2 (11), 559-572.

Pérez-Ortiz, J.A., Gers, F.A., Eck, D. and Schmidhuber, J., 2003. Kalman filters improve LSTM network performance in problems unsolvable by traditional recurrent nets. *Neural Networks,* 16 (2), 241-250.

Perruchet, P., 2008. Implicit Learning. *In:* John H. Byrne, ed., *Learning and Memory: A Comprehensive Reference.* Oxford: Academic Press, 2008, pp. 597-621.

Perruchet, P., and Pacteau, C., 1990. Synthetic grammar learning: Implicit rule abstraction or explicit fragmentary knowledge? *Journal of Experimental Psychology: General,* 119 (3), 264.

Perruchet, P., and Pacton, S., 2006. Implicit learning and statistical learning: one phenomenon, two approaches. *Trends in Cognitive Sciences,* 10 (5), 233-238.

Peterson, G.E., St Clair, D., Aylward, S.R. and Bond, W.E., 1995. Using Taguchi's method of experimental design to control errors in layered perceptrons. *Neural Networks, IEEE Transactions on,* 6 (4), 949-961.

Picton, P., 2000. *Neural networks.* New York: Palgrave.

Pinker, S. and Prince, A. (1988). On language and connectionism: Analysis of a parallel distributed processing model of language acquisition. *Cognition,* 28, 73-193.

Plaut, D.C., 1999. Connectionist modeling. In A. E. Kazdin (Ed.), Encyclopedia of psychology.

Plunkett, K. and Marchman, V. (1996) Learning from a connectionist model of the English past tense. *Cognition,* 61, 299-308.

Pothos, E.M., 2007. Theories of Artificial Grammar Learning. *Psychological Bulletin,* 133 (2), 227-244.

Principe, J.C., Kuo, J.M. and de Vries, B., 1993. Backpropagation through time with fixed memory size requirements. *In: Neural Networks for Signal Processing [1993] III. Proceedings of the 1993 IEEE-SP Workshop,* IEEE, pp. 207-215.

Prokhorov, D.V., Feldkarnp, L. and Tyukin, I.Y., 2002. Adaptive behavior with fixed weights in RNN: an overview. *In: Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on,* IEEE, pp. 2018-2022.

Qin, S., Su, H. and McAvoy, T.J., 1992. Comparison of four neural net learning methods for dynamic system identification. *Neural Networks, IEEE Transactions on,* 3 (1), 122-130.

Rachez, A., and Hagiwara, M., 2012. Augmented Echo State Networks with a feature layer and a nonlinear readout. *In: Neural Networks (IJCNN), The 2012 International Joint Conference on,* IEEE, pp. 1-8.

Rajasekaran, S., and Pai, G.V., 2003. *Neural Networks, Fuzzy Logic and Genetic Algorithm: Synthesis and Applications (WITH CD).* PHI Learning Pvt. Ltd.

Reber, A.S., 1976. Implicit learning of synthetic languages: The role of instructional set. *Journal of Experimental Psychology: Human Learning and Memory,* 2 (1), 88-94.

Reber, A.S., Kassin, S.M., Lewis, S. and Cantor, G., 1980. On the relationship between implicit and explicit modes in the learning of a complex rule structure. *Journal of Experimental Psychology: Human Learning and Memory,* 6 (5), 492.

Reber, P.J., 2002. Attempting to model dissociations of memory. *Trends in Cognitive Sciences,* 6 (5), 192-194.

Redington, M., and Chater, N., 1998. Connectionist and statistical approaches to language acquisition: A distributional perspective. *Language and Cognitive Processes,* 13 (2-3), 129-191.

Rifai, S., Glorot, X., Bengio, Y. and Vincent, P., 2011. Adding noise to the input of a model trained with a regularized objective. *ArXiv Preprint arXiv:1104.3250.*

Rodan, A., and Tino, P., 2011. Minimum complexity echo state network. *Neural Networks, IEEE Transactions on,* 22 (1), 131-144.

Rodriguez, P., Wiles, J., & Elman, J. (1999). A Recurrent Neural Network that Learns to Count. *Connection Science*, *11*(1), 5–40. doi:10.1080/095400999116340.

Rohde, D.L.T., and Plaut, D.C., 1999. Simple recurrent networks can distinguish non-occurring from ungrammatical sentences given appropriate task structure: reply to Marcus. *Cognition,* 73 (3), 297-300.

Romberg, A.R., and Saffran, J.R., 2010. Statistical learning and language acquisition. *Wiley Interdisciplinary Reviews: Cognitive Science,* 1 (6), 906-914.

Rosa, J.L.G., 2005. Biologically Plausible Artificial Neural Networks. Two-hour tutorial at IEEE IJCNN 2005 - International Joint Conference on Neural Networks, Montréal, Canada, July 31, 2005. Available at http://ewh.ieee.org/cmte/cis/mtsc/ieeecis/contributors.htm.

Rosas, R., Ceric, F., Tenorio, M., Mourgues, C., Thibaut, C., Hurtado, E. and Aravena, M.T., 2010. ADHD children outperform normal children in an artificial grammar implicit learning task: ERP and RT evidence. *Consciousness and Cognition,* 19 (1), 341-351.

Roy, R., 2010. *A primer on the Taguchi method.* Society of Manufacturing Engineers.

Rumelhart, D.E., and McClelland, J.L., 1985. *On learning the past tenses of English verbs.* Institute for Cognitive Science, University of California, San Diego.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J., 1985. *Learning Internal Representations by Error Propagation.*

Rumelhart, D.E., Hinton, G.E. and Williams, R.J., 2002. Learning representations by back-propagating errors. *Cognitive Modeling,* 1, 213.

Rumelhart, D.E., Hintont, G.E. and Williams, R.J., 1986. Learning representations by back-propagating errors. *Nature,* 323 (6088), 533-536.

Sapir, E., 1929. The status of linguistics as a science. *Language,* , 207-214.

Seidenberg, M.S., MacDonald, M.C. and Saffran, J.R., 2002. Does grammar start where statistics stop? *Science,* 298 (5593), 553-554.

Servan-Schreiber, D., Cleeremans, A. and McClelland, J.L., 1989. *Encoding Sequential Structure in Simple Recurrent Networks.*

Servan-Schreiber, D., Cleeremans, A. and McClelland, J.L., 1991. Graded state machines: The representation of temporal contingencies in simple recurrent networks. *Machine Learning,* 7 (2-3), 161-193.

Setiono, R., and Liu, H., 1995. Understanding neural networks via rule extraction. *In: IJCAI,* Citeseer, pp. 480-485.

Ševa, N. (2006). Exploring the facilitating effect of diminutives on the acquisition of Serbian noun morphology, (August).

Shaw, A.M., Doyle, F.J. and Schwaber, J.S., 1997. A dynamic neural network approach to nonlinear process modeling. *Computers & Chemical Engineering,* 21 (4), 371-385.

Siegelmann, H.T., Horne, B.G. and Giles, C.L., 1997. Computational capabilities of recurrent NARX neural networks. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on,* 27 (2), 208-215.

Skowronski, M.D., and Harris, J.G., 2006. Minimum mean squared error time series classification using an echo state network prediction model. *In: Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on,* IEEE, pp. 4 pp.-3156.

Smith, L.I., 2002. A tutorial on principal components analysis. *Cornell University, USA,* 51, 52.

Stoianov, I.P., 2001. *Connectionist Lexical Processing.*

Sun, R., 2008. Introduction to computational cognitive modeling. *Cambridge Handbook of Computational Psychology,* , 3-19.

Sutskever, I., and Hinton, G., 2010. Temporal-kernel recurrent neural networks. *Neural Networks,* 23 (2), 239-243.

Tebelskis, J., 1995. *Speech Recognition using Neural Networks.*

Tepper, J.A., Powell, H. and Palmer-Brown, D., 1995. Integrating symbolic and subsymbolic architectures for parsing arithmetic expressions and natural language sentences. *In:* Integrating symbolic and subsymbolic architectures for parsing arithmetic expressions and natural language sentences. *Neural Networks: Artificial Intelligence and Industrial Applications.* Springer, 1995, pp. 81-84.

Tepper, J.A., Powell, H.M. and Palmer-Brown, D., 2002. A corpus-based connectionist architecture for large-scale natural language parsing. *Connection Science,* 14 (2), 93-114.

Thomas, M.S., and McClelland, J.L., 2008. Connectionist models of cognition. *The Cambridge Handbook of Computational Psychology,* , 23-58.

Tjongkimsang, E., 1992. A connectionist representation for phrase structures. *Univ.Twente, Connectionism and Natural Language Processing p 53-55(SEE N 94-20242 05-63).*

Tomasello, M., and Tomasello, M., 2009. *Constructing a language: A usage-based theory of language acquisition.* Harvard University Press.

Tong, M.H., Bickett, A.D., Christiansen, E.M. and Cottrell, G.W., 2007. Learning grammatical structure with echo state networks. *Neural Networks,* 20 (3), 424-432.

Trevor. Hastie, Robert. Tibshirani and Friedman, J.J.H., 2001. *The elements of statistical learning.* Springer New York.

Tsai, J., Chou, J. and Liu, T., 2006. Tuning the structure and parameters of a neural network by using hybrid Taguchi-genetic algorithm. *Neural Networks, IEEE Transactions on,* 17 (1), 69-80.

Ulbricht, C., 1995. Multi-recurrent networks for traffic forecasting. *In: Proceedings Of The National Conference On Artificial Intelligence,* John Wiley & Sons LTD, pp. 883-883.

Van der Velde, F., and De Kamps, M., 2006. Neural blackboard architectures of combinatorial structures in cognition. *Behavioral and Brain Sciences,* 29 (1), 37-69.

van der Velde, F., van der Voort van der Kleij, Gwendid T and de Kamps, M., 2004. Lack of combinatorial productivity in language processing with simple recurrent networks. *Connection Science,* 16 (1), 21-46.

Venayagamoorthy, G.K., and Shishir, B., 2009. Effects of spectral radius and settling time in the performance of echo state networks. *Neural Networks,* 22 (7), 861-863.

Verstraeten, D., 2009. Reservoir Computing: computation with dynamical systems.

Victor Chow, K., Denning, K.C., Ferris, S. and Noronha, G., 1995. Long-term and short-term price memory in the stock market. *Economics Letters,* 49 (3), 287-293.

Wah, B.W., and Qian, M., 2004. Constraint-Based Neural Network Learning for Time Series Predictions. *In:* Constraint-Based Neural Network Learning for Time Series Predictions. *Intelligent Technologies for Information Analysis.* Springer, 2004, pp. 409-431.

Waibel, A., 1989. Modular construction of time-delay neural networks for speech recognition. *Neural Computation,* 1 (1), 39-46.

Wan, L., Zeiler, M., Zhang, S., Cun, Y.L. and Fergus, R., 2013. Regularization of neural networks using dropconnect. *In: Proceedings of the 30th International Conference on Machine Learning (ICML-13),* pp. 1058-1066.

Wang, Y., Picton, P., Turner, S. and Attenburrow, G., 2011. Predicting leather handle like an expert by artificial neural networks. *Applied Artificial Intelligence,* 25 (2), 180-192.

Washington, DC: American Psychological Association. Plunkett, K., Karmiloff-Smith, A., Bates, E., Elman, J.L. and Johnson, M.H., 1997. Connectionism and developmental psychology. *Journal of Child Psychology and Psychiatry,* 38 (1), 53-80.

Weckerly, J and Elman, J. (1992) A PDP approach to processing center-embedded sentences. *In: Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ. Lawrence Erlbaum Associates. p. 414-419.

Weigend, A.S., Rumelhart, D.E. and Huberman, B.A., 1991. Generalization by weight-elimination with application to forecasting.

Werbos, P.J., 1990. Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE,* 78 (10), 1550-1560.

Wilson, D.R., and Martinez, T.R., 2001. The need for small learning rates on large problems. *In: Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on,* IEEE, pp. 115-119 vol. 1.

Yang, W., and Tarng, Y., 1998. Design optimization of cutting parameters for turning operations based on the Taguchi method. *Journal of Materials Processing Technology,* 84 (1), 122-129.

Yildiz, I.B., Jaeger, H. and Kiebel, S.J., 2012. Re-visiting the echo state property. *Neural Networks.*

Yildiz, I.B., Jaeger, H. and Kiebel, S.J., 2012. Re-visiting the echo state property. *Neural Networks.*

Zeng, Z., Goodman, R.M. and Smyth, P., 1993. Learning finite state machines with self-clustering recurrent networks. *Neural Computation,* 5 (6), 976-990.

Zur, R.M., Jiang, Y., Pesce, L.L. and Drukker, K., 2009. Noise injection for training artificial neural networks: A comparison with weight decay and early stopping. *Medical Physics,* 36, 4810.

Appendixes

Appendix A: Some Results of Medium Acceptance Criterion Using SRN

Medium Acceptance Criterion

In order to deal with the situation where there is only one right answer for the penultimate symbol in the embedded Reber grammar (as opposed to two at each stage of the Reber grammar) a medium acceptance criterion was applied. The aim was to investigate the performance of the SRN on the embedded Reber grammar using the Luce ratio (Luce 1963) approach to assessing correctness. It is calculated by dividing a given output unit's activation value of all output units. This kind of measurement is a common method used in psychology to model the strength of response tendency among a finite set of alternatives (James, McClelland 1988). It has also quantified the prediction accuracy of the network and it is produced comparable predication accuracy endorsement rates. In these experiments, the same training file was used for ten networks. Table 1 illustrates the best five training results. The effect of the medium acceptance criterion needs to be investigated with this representation before experiments using the second representation (0 & 1) described in chapter four. 10 networks trained and the table shows the five most successful. The network has not learned enough of the embedded structure, contrary with the results obtained from the previous method. In addition, the results of the penultimate dropped compared with the previous result.

Even after training the network with the second representations of the symbols (0.2 & 0.8). The network has not learned as the first representation it was poor results.

Net	Whole Sequence % Correct	Embed%	Penult%	Incorrect	
				Alternative Penult%	Wrong Penult%
1	68.05	98.45	68.56	31.43	0
2	68.14	98.33	68.98	31.01	0
3	63.02	93.81	67.56	32.34	0
4	49.65	99.25	50.24	49.75	0
5	49.46	97.7	50.08	49.91	0

Table 1 SRN training results in learning the embedded Reber grammar with medium acceptance criterion applied

Net	Whole Sequence % Correct	Embed%	Penult%	Incorrect	
				Alternative Penult%	Wrong Penult%
1	1.1	94.7	1.1	0.4	98.5
2	5.3	89.2	5.6	4.4	90
3	3.8	78.3	4.1	3.3	92.6
4	0.1	93	0.1	0	99.9
5	0.6	85.2	0.7	0.5	98.8

Table 2 SRN test results using five test networks trained on the embedded Reber grammar with medium acceptance criterion applied

Appendix B: Results of Different Networks and Info of Some Dataset

Net	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
1	46.89	95.58	50.08	0	50.08	49.9	0
2	37.65	80.65	50.08	0	50.08	49.91	0
3	38.59	63.7	50.08	0	50.08	49.91	0
4	50.08	97.75	50.08	0	50.08	49.9	0
5	39.5	89.14	50.08	0	50.08	49.91	0
6	44.94	93.05	49.99	0	49.99	50	0

Table 1 SRN training results for embedded Reber grammar using hard acceptance criteria and binary symbol representations

Net	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
1	50.08	100	50.08	0	50.08	49.91	0
2	50.08	100	50.08	0	50.08	49.91	0
3	50.08	100	50.08	0	50.08	49.91	0
4	50.08	100	50.08	0	50.08	49.91	0
5	50.08	100	50.08	0	50.08	49.91	0

Table 2 SRN training results for embedded Reber grammar using hard acceptance criteria and non-binary symbol representations

Training file 300000 sequences

Sequences Start with	Number Of Sequences	Percentage
T	149994	49.99%
TT	75893	25.29%
TP	74101	24.70%
P	150006	50.00%
PP	76048	25.34%
PT	73958	24.65%

Testing file 1000 sequences

Sequences Start with	Number Of Sequences	Percentage
T	512	51.20%
TT	293	29.30%
TP	219	21.90%
P	488	48.80%
PP	202	20.20%
PT	286	28.60%

Net	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
1	7.1	14.5	51	29.2	21.8	49	0
2	2.3	4.1	52	29.2	22.8	48	0
3	5.8	12.3	51	29.2	21.8	49	0
4	0.7	1.7	50	29.1	20.9	49	0
5	5.4	10.2	51	29.2	21.8	49	0

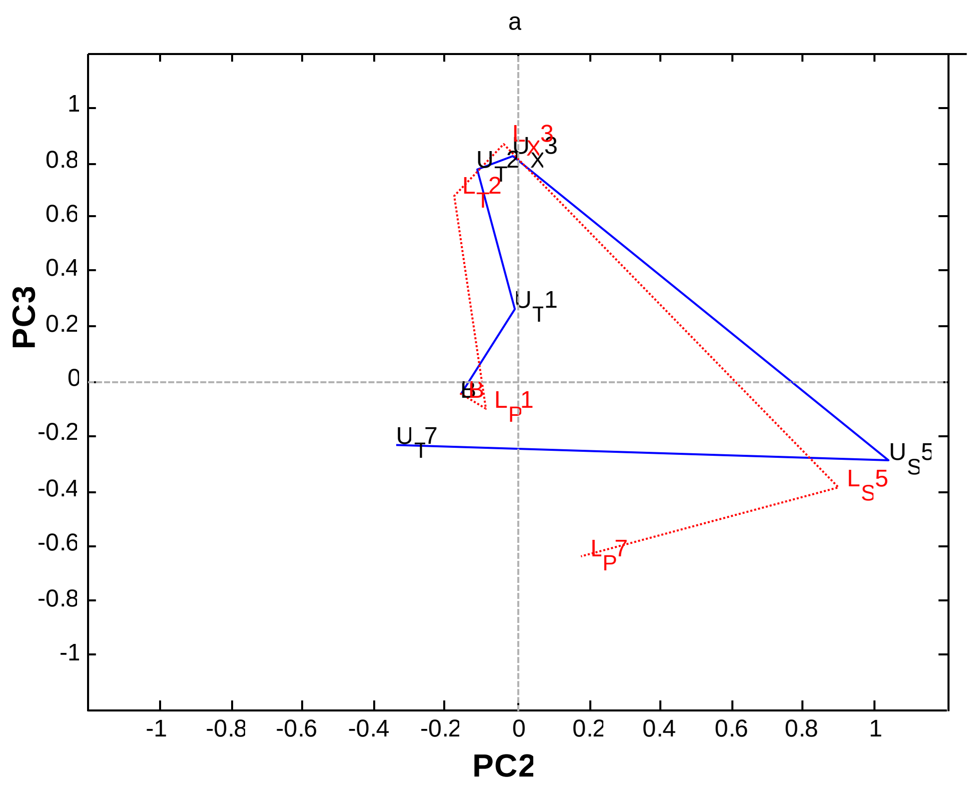
Table 3 SRN results of the symmetrical test file with ten asymmetrical training networks and non-binary symbol representations using soft acceptance criteria

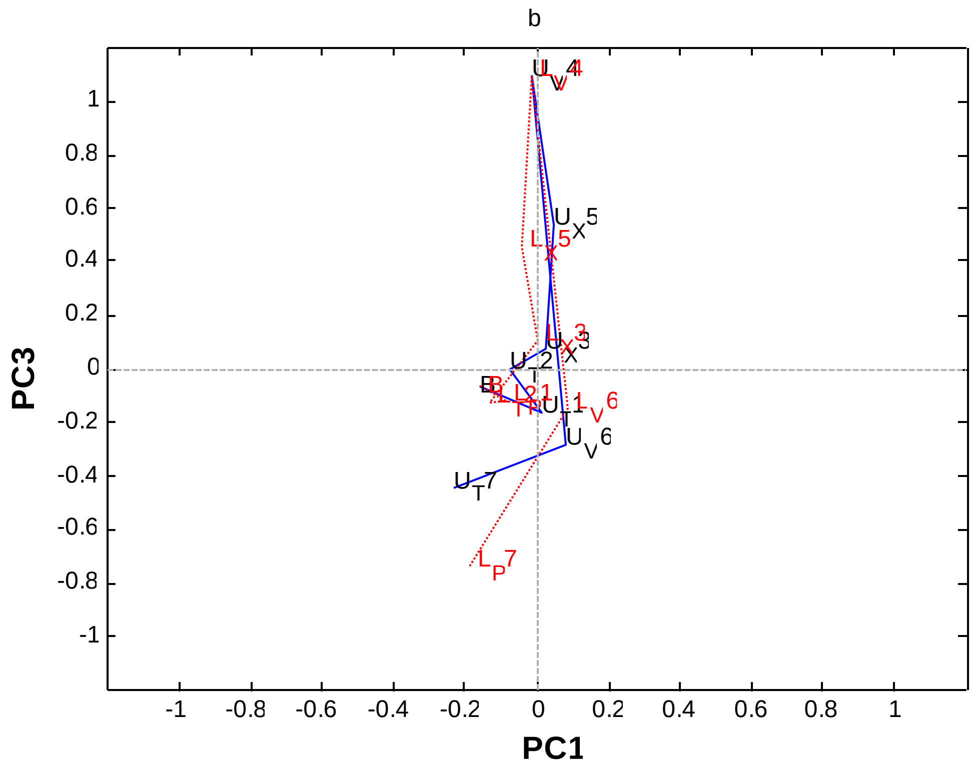
Hidden Unit	Whole Sequence % Correct	Embed%	Penult%	Penultimate		Incorrect	
				P%	T%	Alternative Penult%	Wrong Penult%
5	26.12	61.3	97.08	47.65	49.42	2.92	0
	56.36	57.86	91.45	49.49	41.96	8.54	0
	19.55	39.93	85.27	45.35	39.92	10.63	4.08
	46.27	50.42	90.14	45.27	44.87	9.85	0
	59.69	65.23	91.02	43.16	47.85	8.97	0
7	56.22	56.23	98.8	49.3	49.49	1.19	0
	41.54	41.54	96.89	49.16	47.73	3.1	0
	16.99	34.11	98.6	49.47	49.13	1.39	0
	40.86	40.86	88.16	47.83	40.33	11.83	0
	41.03	41.03	99.17	49.53	49.64	0.54	0.28
10	68.23	69.5	95.95	48.5	47.45	3.76	0.28
	65.96	66.17	98.72	48.64	50.08	0.99	0.28
	60.77	60.77	99.48	49.91	49.56	0.51	0
	62.6	62.6	99.89	49.81	50.08	0.106	0
	57.93	58.02	94.6	45.09	49.5	5.39	0
13	59.5	61.74	99.41	49.33	50.08	0.58	0
	41.56	41.56	99.98	49.91	50.06	0.014	0
	58.25	58.25	99.05	48.97	50.08	0.94	0
	49.7	49.7	87.08	46.88	40.19	12.91	0
	56.92	56.92	98.67	49.91	48.75	1.32	0

Table 4 MRN Training Performance on the Embedded Reber Grammar using various numbers of hidden units

Appendix C: Plots of Different PCA Components Using SRN

These graphs show the different trajectories when using PC2 and PC3; PCA1and PC3





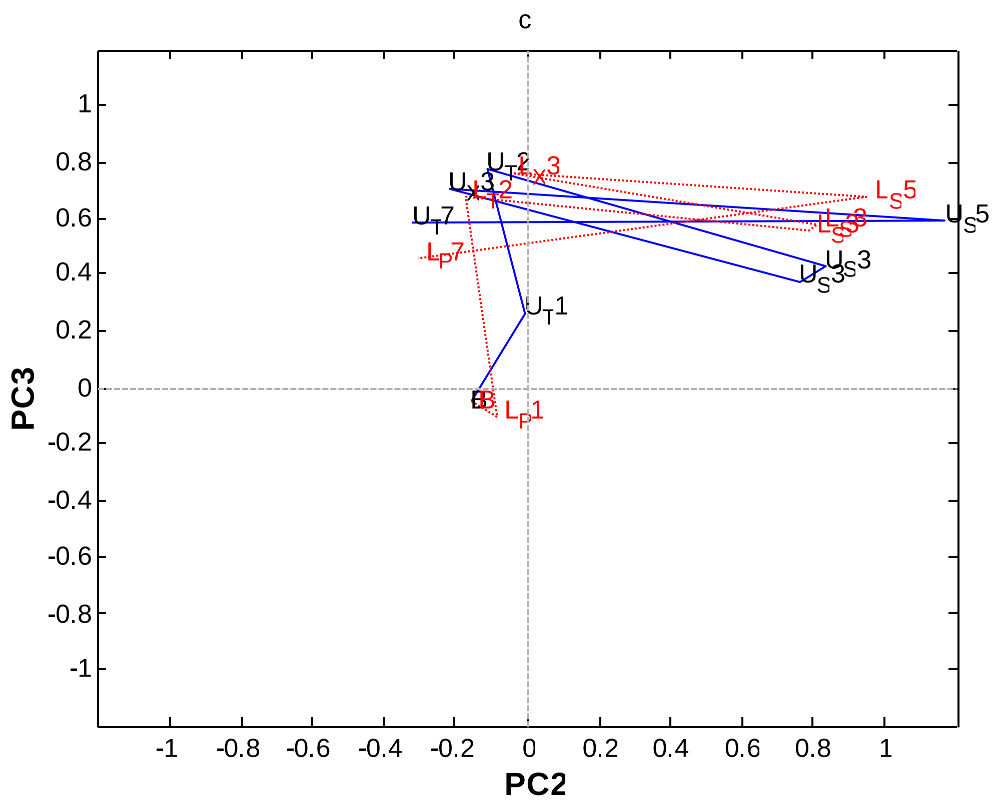


Figure: Plots of the different principle components of the hidden layer activations of a asymmetrically trained by SRN, presented with three pairs of symbol sequences (in a, b and c respectively) from the ERG. Each pair has the same embedded sequence but different initial symbol so that one is in the lower half (dashed lines) and the other is in the upper half (solid blue lines). a BPTXSP/BTTXST, b) BTTXXVVT/BPTXXVVP, c) BTTSSXST/BPTSSXSP

Appendix D: Test Datasets Results of Some of RNNs

MRN has been tested with 1000 sequences, which were asymmetrical sequences.
The result of the file was
Number of Patterns: 16928
Ratio of convergent Patterns: 99.8641%
Number of Sequences: 1000
Ratio of convergent Sequences: 97.7%
The ratio of convergent embedded: 98.2%
Ratio of convergent final: 99.5%
Ratio of Incorrect final but within the alternative: 0.5%
Ratio of Incorrect final: 0%
Number of the correct path T: 477. The percentage of it: 47.7%
Number of the correct path P: 518. The percentage of it: 51.8%

Sequence Length	Total Sequences	Correct	Wrong	Embedded	Final
6	4	4	0	4	4
7	6	6	0	6	6
8	8	8	0	8	8
9	14	14	0	14	14
10	18	18	0	18	18
11	26	26	0	26	26
12	38	37	1	37	38
13	50	48	2	48	50
14	68	66	2	66	68
15	91	90	1	90	91
16	107	105	2	105	107
17	117	114	3	114	117
18	116	114	2	115	115
19	110	107	3	107	110
20	75	74	1	74	75
21	53	53	0	53	53
22	39	36	3	38	37
23	26	24	2	25	25
24	20	19	1	20	19
25	9	9	0	9	9
26	5	5	0	5	5

SRN has been tested with 1000 sequences, which were asymmetrical sequences.

Number of Patterns: 16928
Ratio of convergent Patterns: 97.324%
Number of Sequences: 1000
Ratio of convergent Sequences: 56%

The ratio of convergent embedded: 98.2%
Ratio of convergent final: 56.5%
Ratio of Incorrect final but within the alternative: 43.5%
Ratio of Incorrect final: 0%
Number of the correct path T: 127. The percentage of it: 12.7%
Number of the correct path P: 438. The percentage of it: 43.8%

Sequence Length	Total Sequences	Correct	Wrong	Embedded	Final
6	4	3	1	4	3
7	6	4	2	6	4
8	8	6	2	8	6
9	14	6	8	14	6
10	18	11	7	18	11
11	26	13	13	25	13
12	38	19	19	37	20
13	50	24	26	50	24
14	68	35	33	67	35
15	91	42	49	89	43
16	107	61	46	105	61
17	117	64	53	115	64
18	116	71	45	114	71
19	110	61	49	106	62
20	75	44	31	75	44
21	53	32	21	52	33
22	39	28	11	37	29
23	26	17	9	26	17
24	20	11	9	20	11
25	9	5	4	9	5
26	5	3	2	5	3

ESN has been tested with 1000 sequences, which were asymmetrical sequences.
Number of Sequences: 1000
Ratio of convergent Sequences: 60.7%
The ratio of convergent embedded: 100%
Ratio of convergent final: 60.7%
Ratio of Incorrect final but within the alternative: 39.3%
Ratio of Incorrect final: 0%
Number of the correct path T: 248 the percentage of it: 24.8%
Number of the correct path P: 359 the percentage of it: 35.9%

Sequence Length	Total Sequences	Correct	Wrong	Embedded	Final
6	4	4	0	4	4
7	6	6	0	6	6
8	8	4	4	8	4
9	14	7	7	14	7
10	18	9	9	18	9
11	26	13	13	26	13
12	38	19	19	38	19
13	50	25	25	50	25
14	68	34	34	68	34
15	91	48	43	91	48
16	107	65	42	107	65
17	117	75	42	117	75
18	116	79	37	116	79
19	110	74	36	110	74
20	75	48	27	75	48
21	53	36	17	53	36
22	39	29	10	39	29
23	26	12	14	26	12
24	20	12	8	20	12
25	9	5	4	9	5
26	5	3	2	5	3

NARX has been tested with 1000 sequences, which were asymmetrical sequences.

Number of Patterns: 16928
Ratio of convergent Patterns % : 98.1864
Number of Sequences: 1000
Ratio of convergent Sequences % : 70.9
Ratio of convergent embedded % : 95.7
Ratio of convergent final % : 74
Ratio of Incorrect final but within the alternative % :26
Ratio of Incorrect final % :0
Number of the correct path T :350 the percentage of it : %35
Number of the correct path P: 390 the percentage of it: %39
Sequence Length: Total Sequence Correct Wrong Embedded Final

Sequence Length	Total Sequences	Correct	Wrong	Embedded	Final
6	4	4	0	4	4
7	6	6	0	6	6
8	8	8	0	8	8
9	14	14	0	14	14
10	18	18	0	18	18
11	26	26	0	26	26
12	38	36	2	38	36
13	50	41	9	47	43
14	68	54	14	68	54
15	91	66	25	86	70
16	107	76	31	102	81
17	117	80	37	112	83
18	116	79	37	109	84
19	110	67	43	103	71
20	75	44	31	70	48
21	53	36	17	51	36
22	39	23	16	35	27
23	26	12	14	26	12
24	20	10	10	20	10
25	9	6	3	9	6
26	5	3	2	5	3

Comparison between SRN, MRN and ESN

The prediction of the whole dataset	SRN	MRN	NARX	ESN
	56%	97.7%	70.9%	60.7%

The networks predicted the whole sequence length correctly from the test file.

Length	SRN	MRN	NARX	ESN
6 and 7	X	√	√	√
8 to 11	X	√	√	X
12 to 20	X	X	X	X
21	X	√	X	X
22 to 24	X	X	X	X
25 and 26	X	√	X	X

Appendix E: The Evaluation of Symmetrical Training Dataset

Internal Representation using Symmetrical Training Dataset.

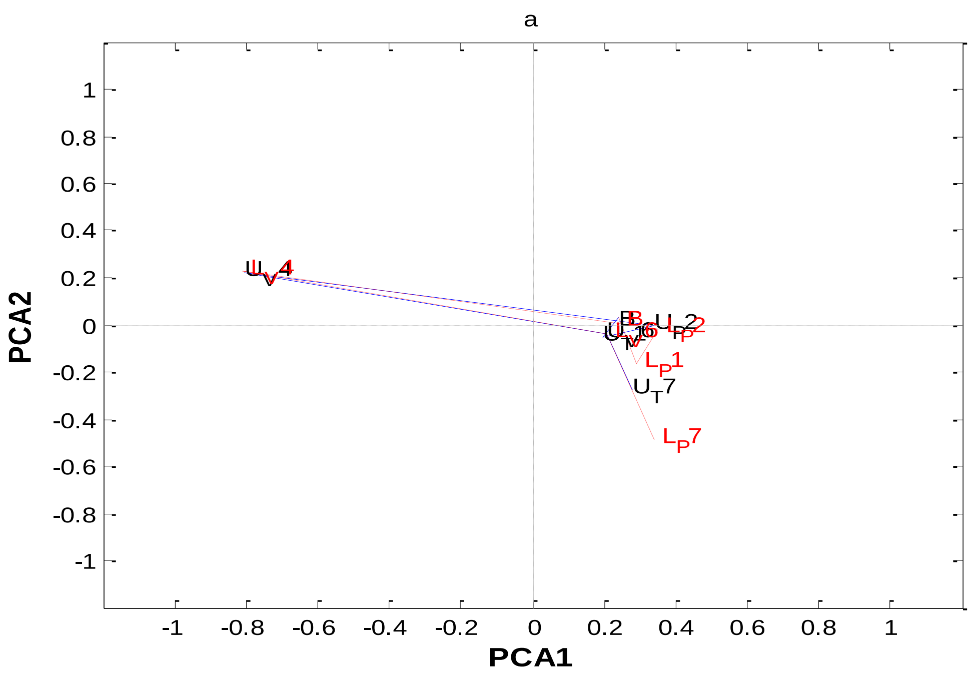
SRN Trained with Symmetrical Sequences

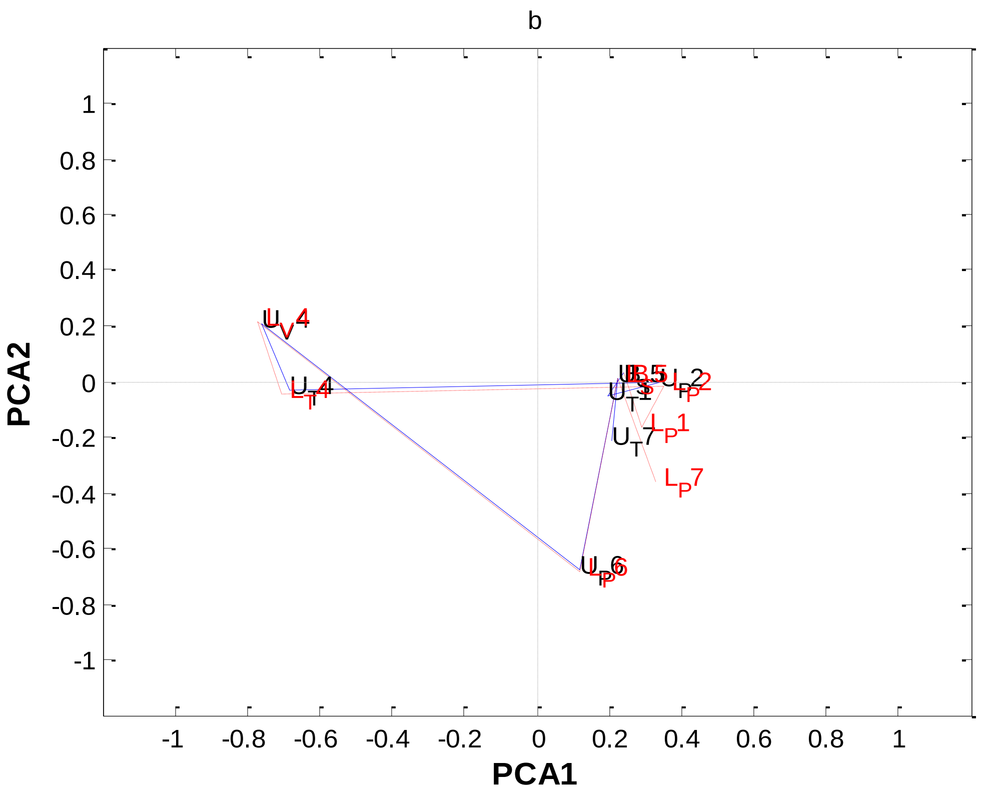
Fourteen symmetrical sequences are selected; Table 1 shows the sequences and their results after the SRN has been trained. Three of the sequences were not correctly predicted with the error being at the penultimate symbol. Although different grammatical aspects may be distinguished in different subspaces, the subspace spanned by the eigenvectors corresponding to the first and the second largest absolute eigenvalue of the covariance matrix, that defines the two principal components labelled PC1 and PC2, respectively, which considered as being the most likely to hold significant information. Sequence length and correctly predicted symmetrical sequences are studied first. Some of these sequences are numbered in the Table 1: 9, 10; 11 and 12 and have lengths of eight symbols, and the others are mentioned in the title in the figure specified.

SRN using Symmetrical sequences					
No	Length	Embed	Sequences	Prediction	Reason for Failure
1	6	U	BTTXST	T	Penult
2		L	BPTXSP	F	
3		U	BTPVVT	T	
4		L	BPPVVP	T	
5	8	U	BTTXXVVT	T	
6		L	BPTXXVVP	F	Penult
7		U	BTTSSXST	F	Penult
8		L	BPTSSXSP	T	
9		U	BTPTVPST	T	
10		L	BPPTVPSP	T	
11		U	BTPTTVVT	T	
12		L	BPPTTVVP	T	
13	16		BPTXXTTTVPXTTVVP	T	
14	26		BPTXXTVPXVPXTVPXVPXVPXTVVP	T	

Table 1 Sequences results for SRN and the position of the unpredicted symbol

The trajectories of some of these sequences in the PC1/PC2 subspace are shown in Figure 1. The labels on the graph represent the states of the grammar for the symbol transition indicated using the labelling scheme described above. The trajectories divergence slightly for each pair of sequences, however the similarity between the trajectories in each pair where the embedded section is the same, and the difference between the pairs from each other is striking. These sequences were all predicted correctly by the SRN. The figure shows a very slight divergence from the same starting point in both embedded states with the upper and lower trajectories. However, where the sequences are different (the first symbol after the start marker (T/P) and the penultimate symbol) have noticeable distances between the upper and embedded symbols. It is worth noting that in the internal representations of the SRN, for all the three pairs of the sequences. In the Figure 1 b and c the positions of state four are positioned in the second and third quarters for each pathway (V or T) and in a, the state four located V in the second quarter. N.B. Quartiles are defined here as first (top-right); second (top-left); third (bottom-left) and fourth (bottom-right).





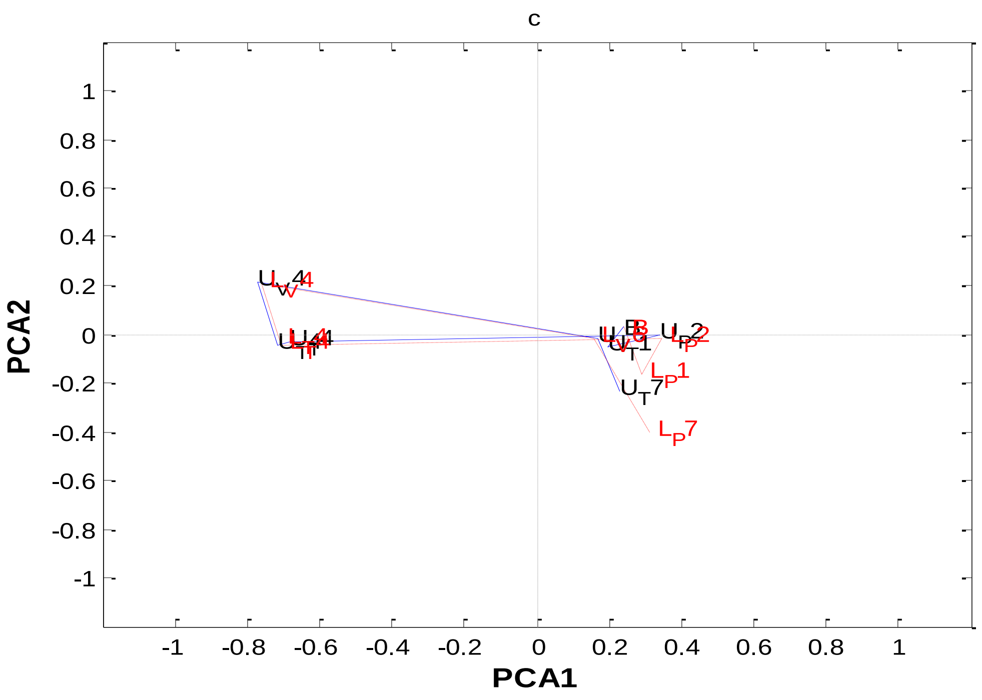


Figure 1 Plots of the two most significant principle components of the hidden layer activations of a symmetrically trained SRN, presented with three pairs of symbol sequences (in a, b, and c respectively) from the embedded Reber Grammar. Each pair has the same embedded sequence but has different initial symbols so that one is in the lower half (dashed red lines) and the other is in the upper half (solid blue lines). The sequences are: a) BTPVVT/BPPVVP (b) BTPTVPST/BPPTVPSP (C) BTPTTVVT/BPPTTVVP.

To investigate more about the states, a number of different sequences, correctly predicted by the SRN, were selected. The minimum and maximum for both PC1 and PC2 were computed for their states values (same numbers of each state where computes). The distribution of each state can be measured by ascertaining its mean and range (calculated by finding the difference between the highest value of the PC of each state and its lowest value). Table 2 and Table 3 provide the results obtained from calculating the range and the mean of the sequences states respectively. Figure 2 presents the results of the state's ranges. It shows that the variability (range) of both principle components of the activation vectors is consistent whether the symbol is in the upper section or the lower, i.e. for state 5, PCA 1 is very variable for both upper and lower grammar sequences whereas PCA 2 is much less so; this position is reversed for state 6.

Embedded	States	Range		Embedded	States	Range	
		PC1	PC2			PC1	PC2
Upper	1	0.00003	0.00006	Lower	1	0.25414	0.81043
	2	0.14819	0.48126		2	0.00003	0.00012
	3	0.53903	0.14291		3	0.17509	0.48815
	4	0.11884	0.26484		4	0.82923	0.29743
	5	1.00753	0.28692		5	0.10413	0.27049
	6	0.09155	0.70294		6	0.94316	0.12750
	7	0.25414	0.81043		7	0.08950	0.67967

Table 2 SRN: The range of number of sequences with respect to their states for each PC1 and PC2

Embedded	States	Mean		Embedded	States	Mean	
		PC1	PC2			PC1	PC2
Upper	1	0.197	-0.051	Lower	1	0.290	-0.166
	2	0.399	0.187		2	0.418	0.178
	3	-0.096	0.391		3	0.212	0.520
	4	-0.729	0.064		4	-0.752	0.111
	5	-0.376	-0.051		5	-0.347	0.037
	6	0.165	-0.299		6	0.148	-0.288
	7	0.292	-0.408		7	0.343	-0.511

Table 3 SRN: The mean of number of sequences with respect to their states for each PC1 and PC2

Some incorrect sequences were selected to compare with the above, which were correctly predicted sequences. In addition, the error for the SRN was in the penultimate symbol (the embedded part was correctly predicted) therefore, the emphasis will be to consider the penultimate part of the sequence, to understand the reason for the error.

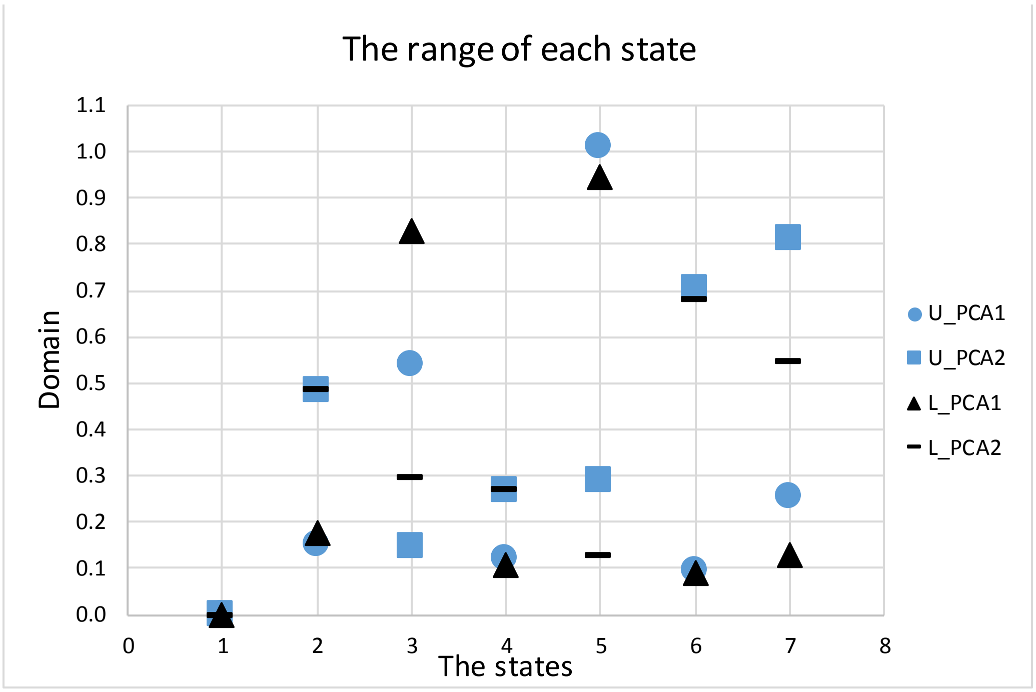
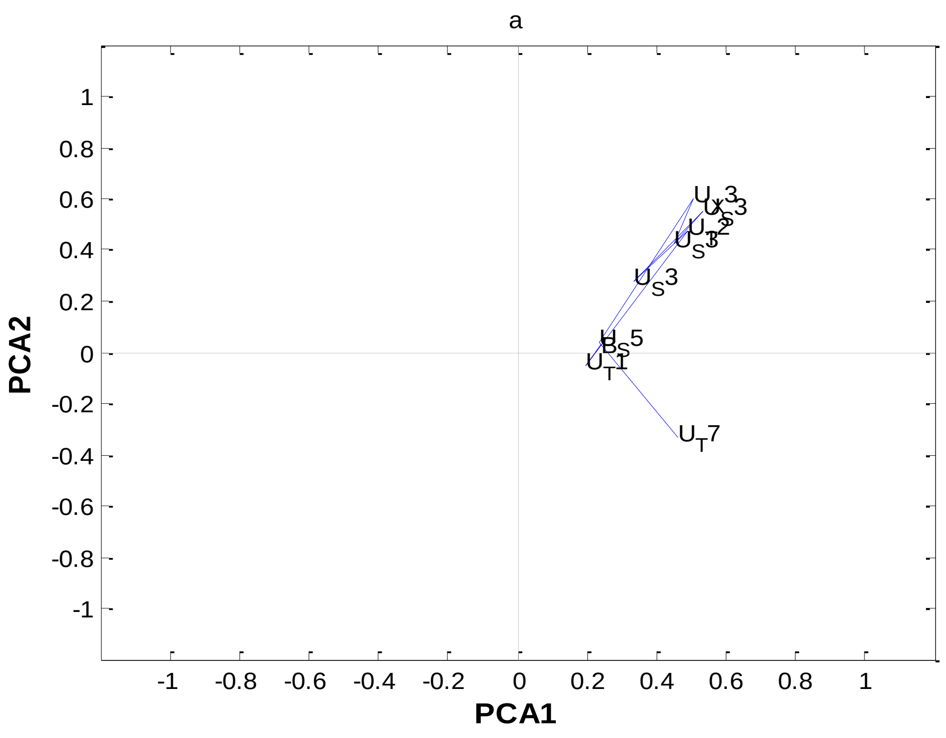


Figure 2 SRN: The range of number of sequences with respect to their states (correctly predicted sequences)



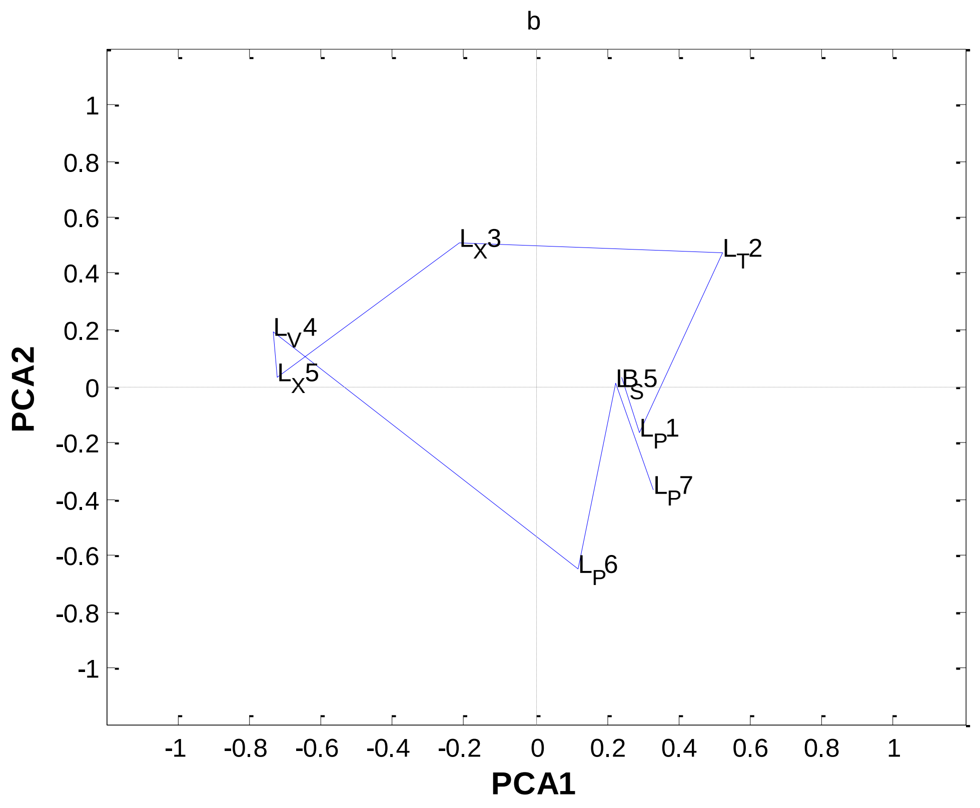


Figure 3 Plots of the two most significant principle components of the hidden layer activations of a symmetrically trained SRN, (a) BTTSSSXST, (b) BPTXXVPSP. Are incorrectly predicted sequences have 9-length symbols.

Figure 4 presents the results obtained from incorrectly predicted sequences: a and b are the upper and lower sequences respectively of nine symbols in length. The remarkable feature from this visualisation of the two principle components for these sequences with incorrect penultimate predictions, is that for most of the trace they are different from each other and indeed traverse different quadrants.

As shown in Figure 5 the range of the state seven are diverged from the results that shown in the Figure 4, in respect of the range between PC1 and PC2 in lower embedded for unpredicted sequences, increases to 0.24 comparing with the correctly predicted sequences and the upper decreased to 0.1. To study state seven more, the ranges of state seven of both correctly and incorrectly predicted sequences have been plotted to grasp where the internal representation of the network located each type of sequence.

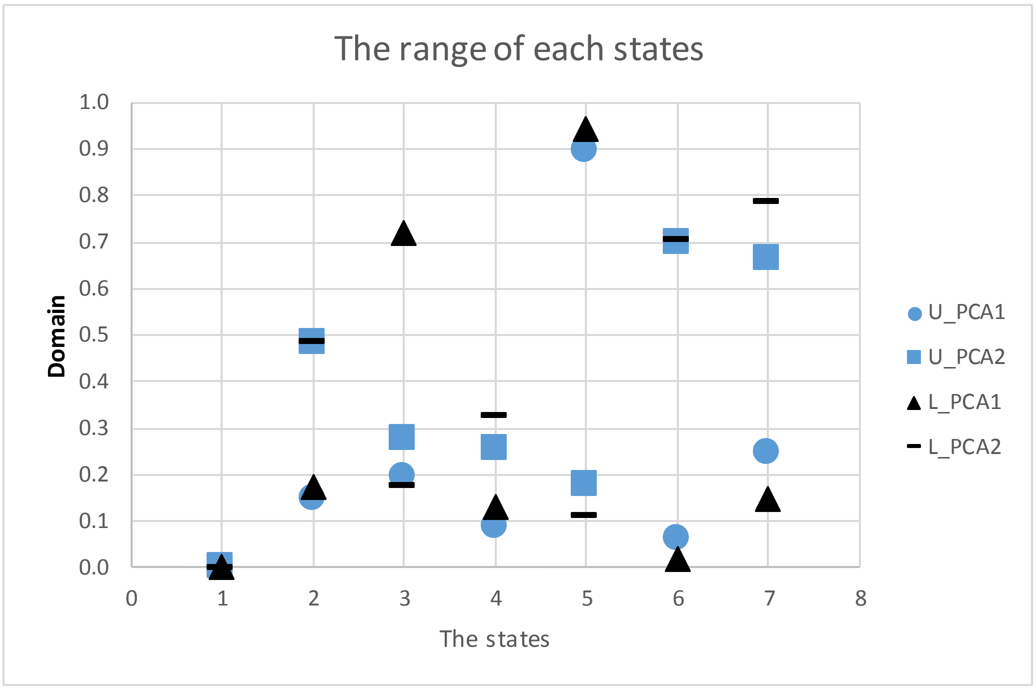


Figure 4 SRN: The range of number of sequences with respect to their states (incorrectly predicted sequences)

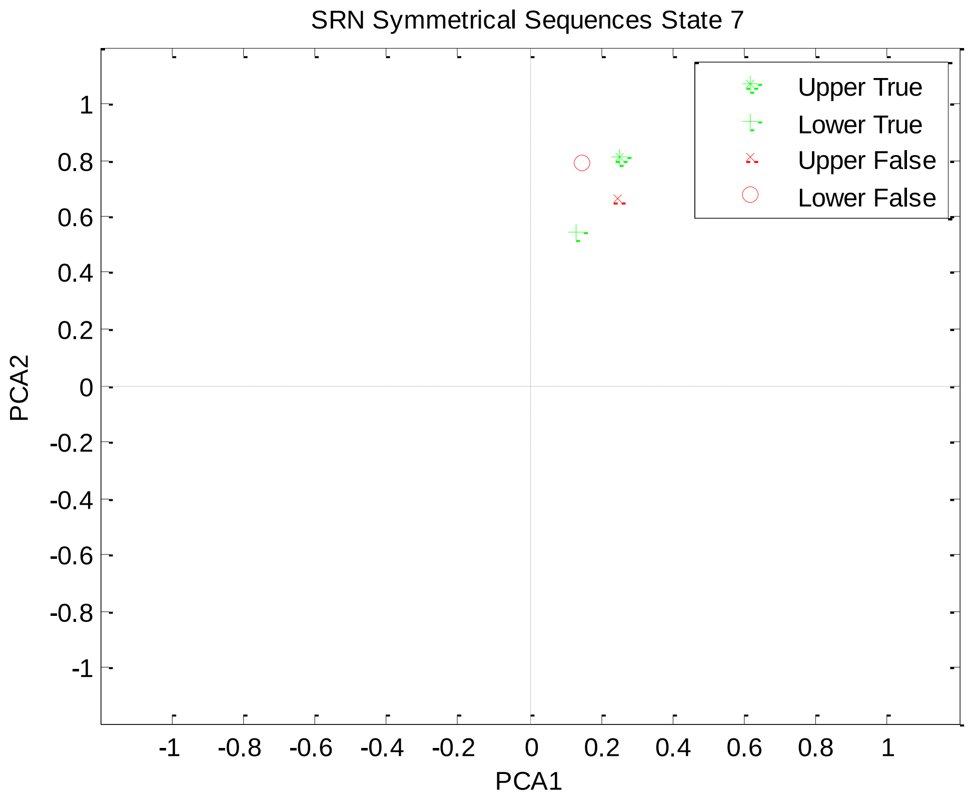


Figure 5 SRN: the range of state seven taken from number of sequences, predicted and unpredicted sequences.

Figure 5 shows the range of states seven for both upper and lower embedded sequences of the correctly predicted sequences that coloured in green are located in the first quarter. Incorrectly predicted sequences, were also located in the first quarter but in different position from the correct ones. This may because of PC1 was not sufficiently developed since the network performs poorly

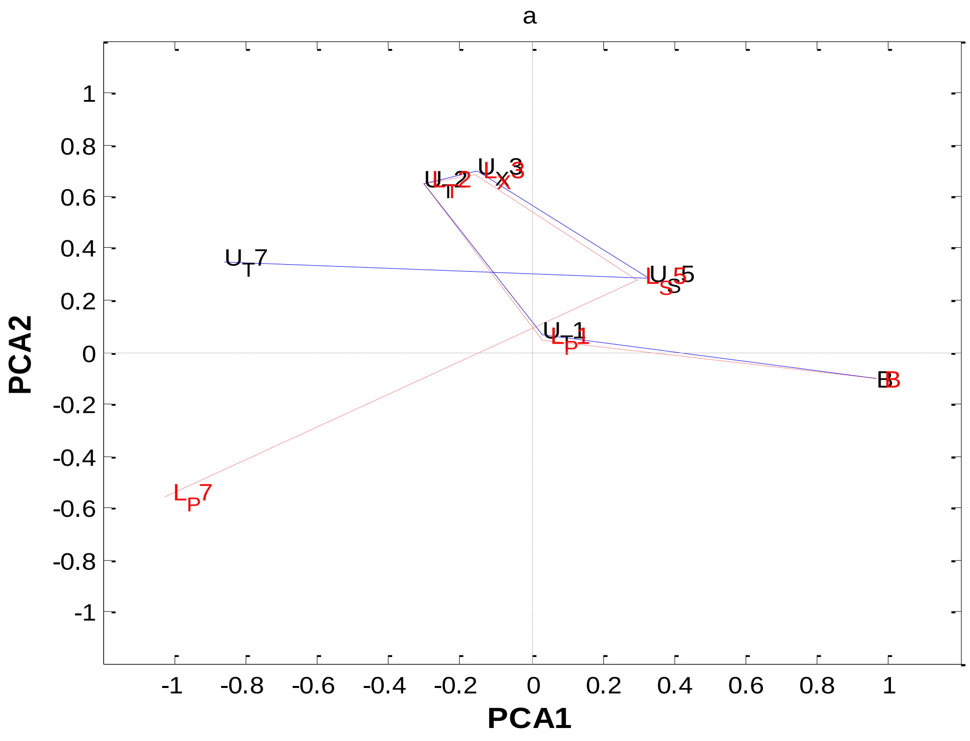
MRN Trained with Symmetrical Sequences

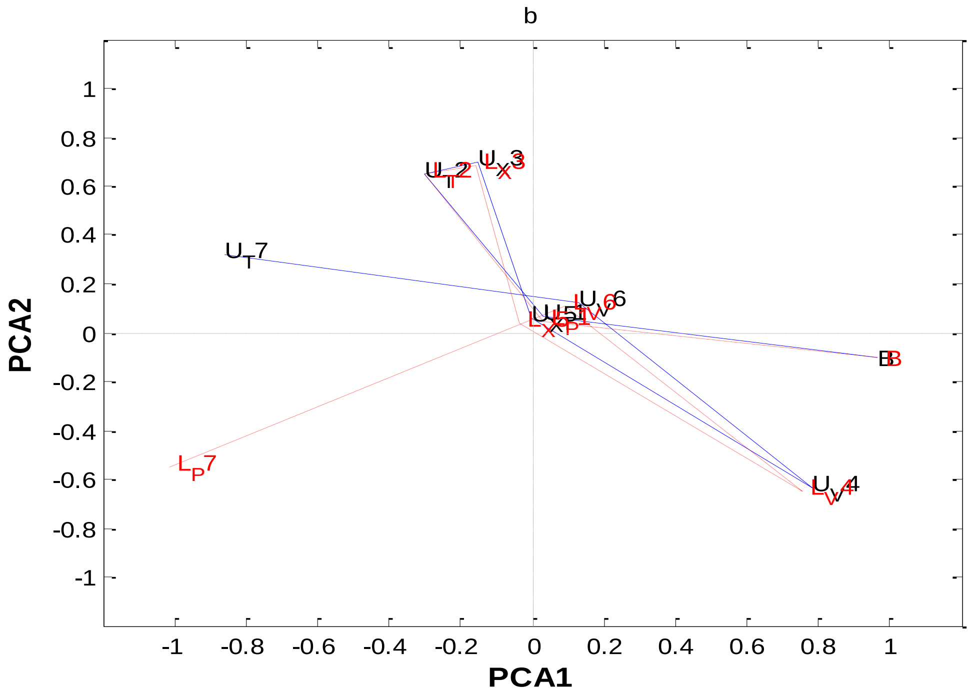
The sequences selected when investigating the SRN are also used here. Table 4 illustrates the results have been obtained from MRN when noise injection was used in the training; fourteen sequences of various lengths were selected. PCA has been applied on the training dataset to visualise the internal representation of the MRN and study how the network organized the grammar. A number of correctly predicted sequences have been studied to grasp the trajectories of their states.

MRN using Symmetrical sequences					
No	Length	Embed	Sequences	Predication	Reason for Failure
1	6	U	BTTXST	T	
2		L	BPTXSP	T	
3		U	BTPVVT	F	Penult
4		L	BPPVVP	T	
5	8	U	BTTXXVVT	T	
6		L	BPTXXVVP	T	
7		U	BTTSSXST	T	
8		L	BPTSSXSP	T	
9		U	BTPTVPST	T	
10		L	BPPTVPSP	T	
11		U	BTPTTVVT	F	Penult
12		L	BPPTTVVP	T	
13	16		BPTXXTTTVPXTTVVP	T	
14	26		BPTXXTVPXVPXTVPXVPXVPXTVVP	T	

Table 4 Sequences results for MRN and the position of the unpredicted symbol

Figure 6 depicts the trajectories of symmetrical sequences in the PC1/ PC2 component subspace. The principle divergence of the two trajectories is at the penultimate symbol comparing the upper or lower parts of the grammar. (a) Shows the sequences 1 and 2 of length six in Table 4 i.e. upper and lower sequences; solid and dashed lines respectively.





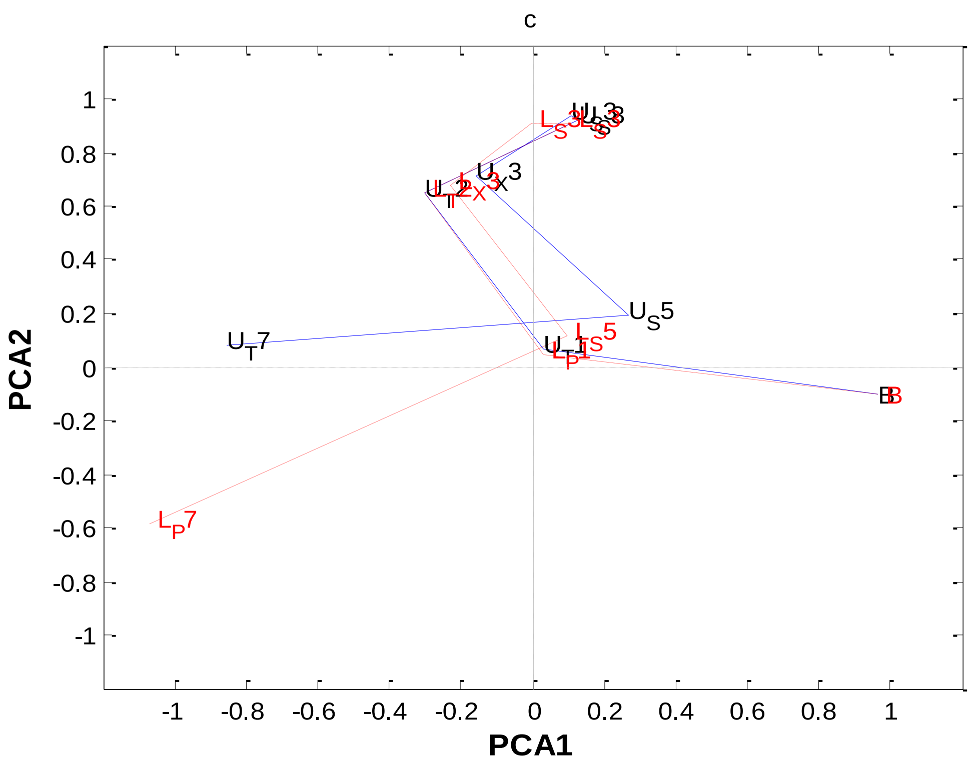


Figure 6 Plots of the two most significant principle components of the hidden layer activations of a symmetrically trained MRN, trajectories of the three identical symmetrical sequences that were correctly predicted, the sequences are: a) BTTXST/BPTXSP, b) BTTXXVVT/BPTXXVVP and c) BTTSSXST/BPTSSXSP.

Figure 6 (b, c) shows sequences of length eight. The location of all the states for both upper and lower part except for state seven are located in the same quarters for each pair. However, location of state seven for the upper half of the grammar is located in the second quarter and for the lower half is located in the third quarter. This seems to show a clear representation by these two principal components of the difference at the point where the sequence symbols differ (the penultimate symbol) but a limited amount of memory of the difference held by them prior to this (memory of the divergence at the start of the sequence). To investigate more about the states, the range of each state has been computed for several sequences (five different values for each state).

Table 5 shows the range of each state for the upper and lower parts of the sequences. Figure 7 depicts this information graphically. Figure 7 shows that the variability (range) of both principle components of the activation vectors is consistent whether the symbol is in the upper section or the lower. A striking observation to emerge from the figure is that each state range is located differently in subspace showing systematic nature of the internal representation of the MRN.

Embedded	States	Range		Embedded	States	Range	
		PC1	PC2			PC1	PC2
Upper	1	0.00004	0.00007	Lower	1	0.00007	0.00009
	2	0.01660	0.78226		2	0.01630	0.78415
	3	0.28531	0.23711		3	0.33777	0.23399
	4	0.99502	0.59741		4	1.06641	0.63145
	5	0.32853	0.22116		5	0.42364	0.30967
	6	0.54501	0.30817		6	0.50981	0.29276
	7	0.08226	0.26876		7	0.05489	0.03401

Table 5 Range of the each state with MRN for symmetrical sequences (predicted sequences)

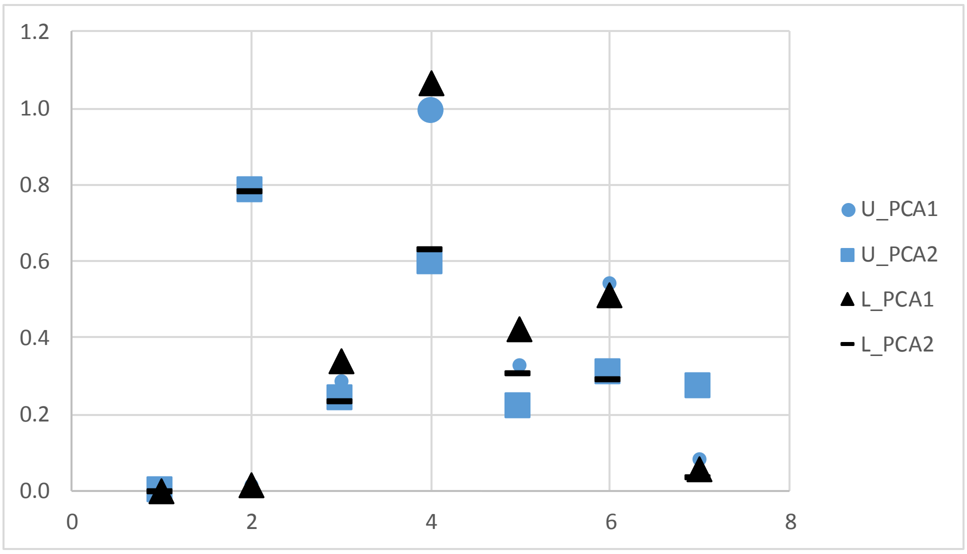
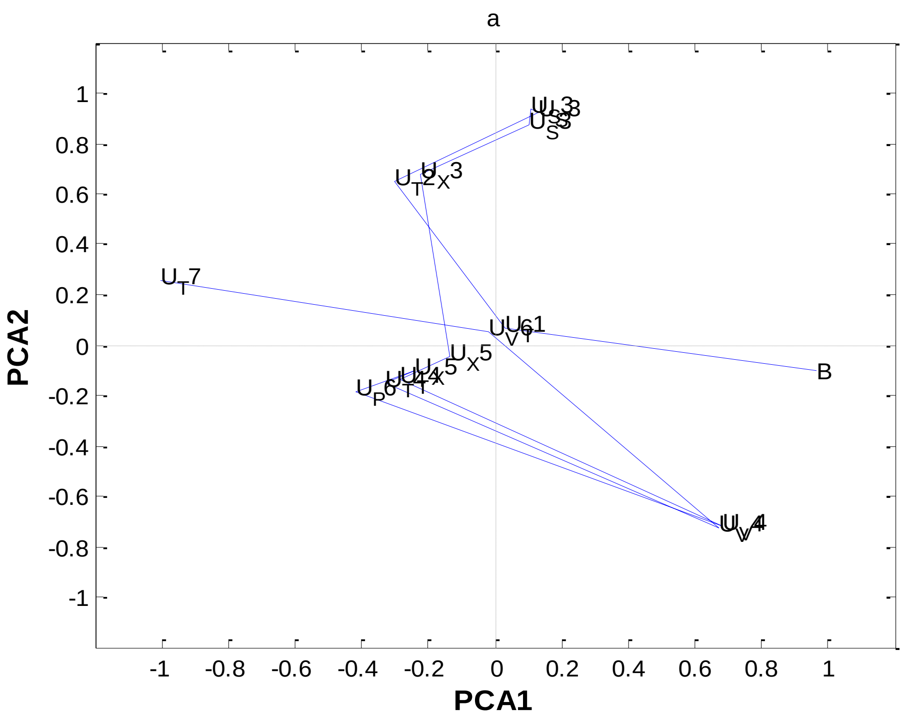


Figure 7 The range of each sate by MRN using symmetrical sequences

Now turn to the incorrectly predicted sequences to observe where the network locates the states and the difference with the correctly predicted ones. Figure 8 shows the trajectories of the upper and lower sequences that have been incorrectly predicted by the MRN and also for the penultimate symbol for both parts. The graph depicts the similarity of the states compared with the correctly predicted sequences. To investigate the difference between the trajectories of state seven for both the predicted and unpredicted sequences, the ranges of state seven have been calculated for both of them. Figure 9 illustrates the range of state seven in four sequences: upper and lower paths through the grammar with correctly and incorrectly predicted penultimate symbols. The figure shows that the states for the correctly predicted upper and lower sequences are located in the same part of the subspace while the incorrectly predicted ones have a larger distance between them although they are in the same quarter.



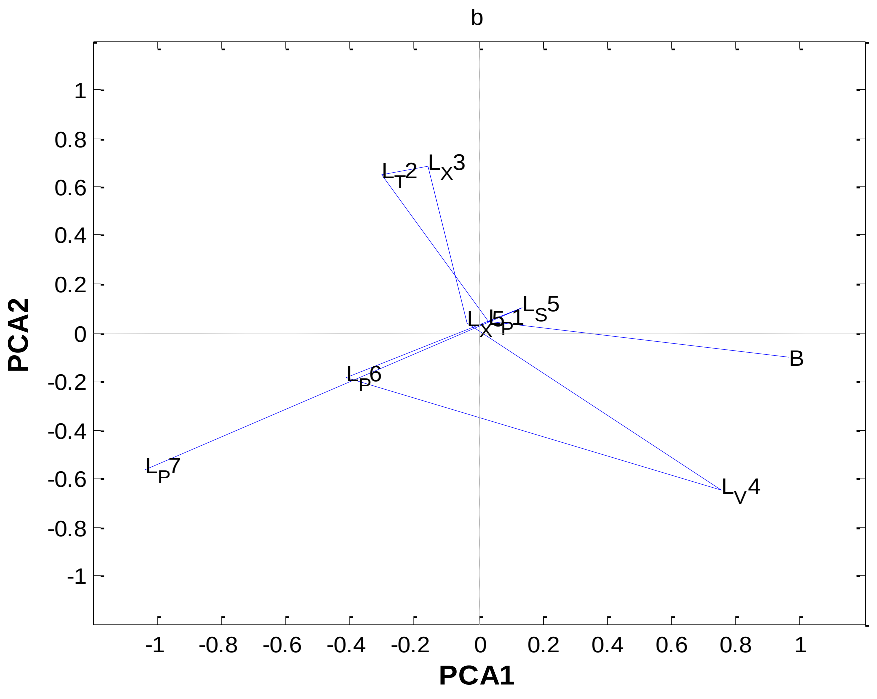


Figure 8 Plots of the two most significant principle components of the hidden layer activations of a symmetrically trained MRN, incorrectly predicted sequences by MRN (a) BTTSSSXXTVPXTVVT has 16-length symbols and upper sequence, (b) BPTXXVPSP has 9-length of symbols and in the lower part sequence.

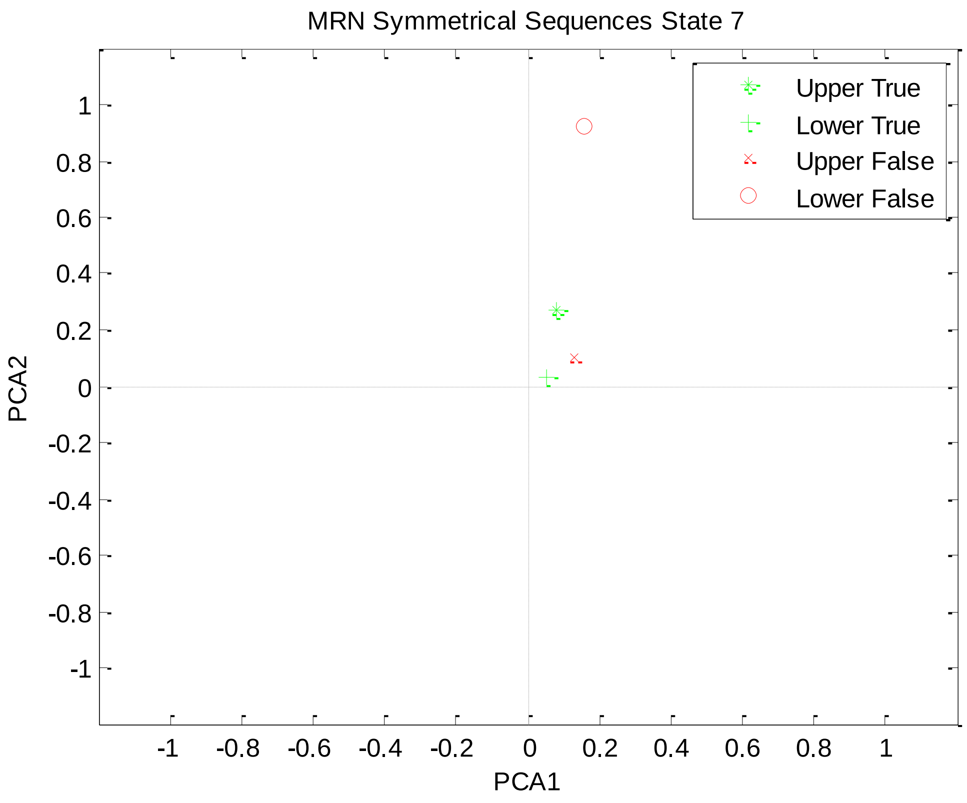


Figure 9 MRN: located ranges of the state seven for predicted and unpredicted symmetrical sequences.

A possible explanation is that the difference in the state seven between the predicted and unpredicted penultimate is a results from the poorly performance of the network.