TRAINING OF CC4 NEURAL NETWORK USING UNARY AND SPREAD UNARY INPUTS

By

PUSHPA SREE POTLURI

Bachelor of Technology in

Electronics and Communication Engineering

Jawaharlal Nehru Technological University

Kakinada, India

2014

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2015

TRAINING OF CC4 NEURAL NETWORK USING UNARY AND SPREAD UNARY INPUTS

Thesis Approved:

Dr. Subhash C. Kak

Thesis Advisor

Dr. Ramachandra Gupta Ramakumar

Dr. Daniel R. Grischkowsky

ACKNOWLEDGMENTS

I take this opportunity to express my gratitude to everyone who is involved in the successful completion of my thesis.

It has been an honor to have Dr. Subhash Kak as my advisor and research supervisor. I am grateful to him for his advice, support and guidance, not just in the course of my research but also for my graduate career at OSU.

I would also like to thank Dr. Daniel R. Grischkowsky and Dr. Ramachandra Gupta Ramakumar for being my thesis committee members and for providing valuable inputs to my research.

Finally, I am grateful eternally to my family: my mother, father and my brother for their affection, care and support, at all times throughout my life.

Acknowledgements reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

Name: PUSHPA SREE POTLURI

Date of Degree: DECEMBER, 2015

Title of Study: TRAINING OF CC4 NEURAL NETWORK USING UNARY AND SPREAD UNARY INPUTS

Major Field: ELECTRICAL ENGINEERING

Abstract: This thesis presents results on training the corner classification (CC4) feed forward neural networks with the recently proposed spread unary inputs. We show the performance of this network is quite like that of the original CC4 network. The spread unary CC4 network is tested on pattern classification data. It is also used in time series prediction which is illustrated using the Mackey-Glass time series, sunspot predictions and the prediction of d-sequences.

LIST OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

In spite of great advances in computer technology, human beings remain more intelligent than machines at certain tasks [1,2]. The superfast computer can perform complex calculations, but at the same it cannot manage an operation which needs intuitive thinking. For example, to solve a puzzle a lot of intuition and prediction is needed. Humans find solutions by good clues and they can figure out certain things from seemingly unrelated facts that are stored in their memories. Though computers perform some tasks with high accuracy and efficiency, they cannot generalize in situations with much uncertainty as humans do.

In a computer, the chips and few other components are interconnected and structurally organized. The biological neuron is the basic structural unit of the brain. It has 3 parts: soma, dendrites, and the axon. The central unit of the neuron is named as soma or cell body. The cell nucleus is located in soma and it ranges from 3-18 μm in diameter. Dendrites are extensions emanating from the cell body. The output from the nucleus is carried by a thin cable-like projection called the axon which carries electrical impulses from the neuron. The end of the axon splits into fine strands and each strand terminates into a synapse. The synapse connects the axon and dendrites. The dendrite absorbs the chemical which is released by axon. This converts that chemical signal to an electrical signal. This signal finally reaches the central unit of the nucleus [2].

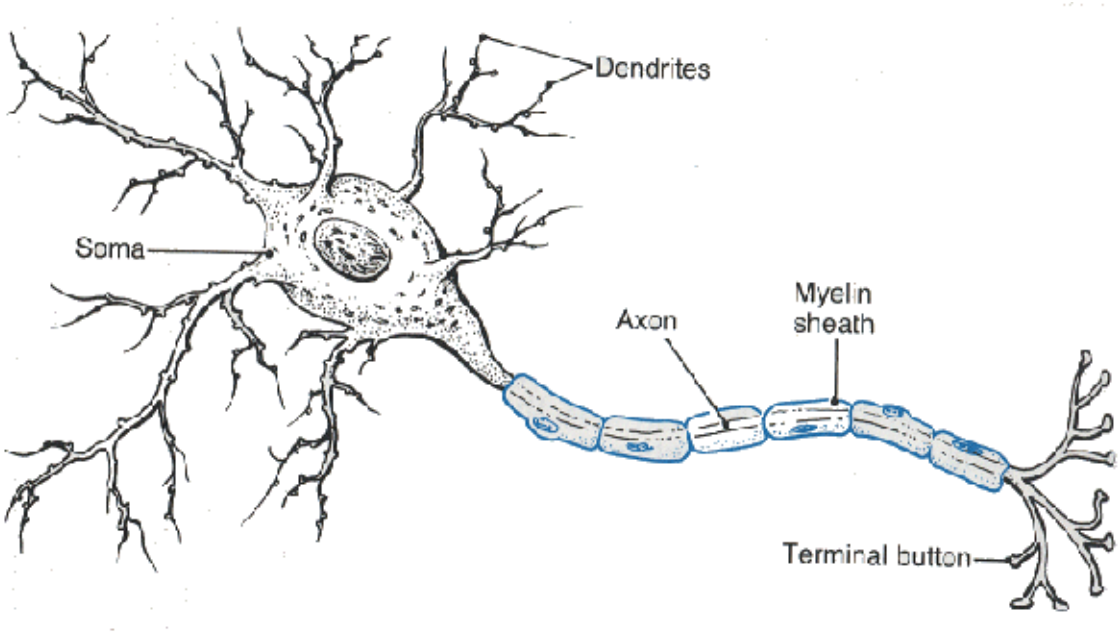


Figure 1**:** Biological Neuron (*Courtesy: Cedar Crest College*)

Artificial Neuron

An artificial neuron is an abstract mathematical model of a biological neuron. The first of its kind was the Threshold Logic Unit which was proposed by Warren McCulloch and Walter Pitts in 1943. In this model, each neuron receives one or more inputs from other neurons (simulating dendrites) and generates an output (axon in biological neuron). The effect of each input line on the neuron is controlled by a synaptic weight (analogous to synaptic strengths in biological neurons). Synaptic weights adapt so that the whole network learns to perform useful computations such as recognizing objects, understanding languages and controlling the body. The weighted sum of these inputs is used as input to the threshold function, also generally known as activation function or transfer function, to generate the output.

Neural networks model the human brain. They perform well various generalization tasks such as pattern recognition, pattern classification, optimization and time series prediction. Artificial neural networks possess a large number of processing elements or nodes/neurons which operate in parallel. Neurons are connected with others by connection link. Each link is associated with weights which contain information about the input signal. Each neuron has an internal state of its own which is a function of the inputs that neuron receives.

CHAPTER II

CC4 APPROACH FOR NEURAL NETWORKS

Artificial Neural Networks (popularly known as ANNs) are a network of unidirectional connections comprising of very simple processing units each of which in turn might have a small amount of memory. Owing to their efficiency and better performance over conventional methods, these are being extensively used in wide areas such as virtual reality, data compression, adaptive control, detection and tracking of moving targets and the like.

The current work is based on the corner classification approach to artificial neural network training of CC4 network, as proposed by Subhash Kak in 1992/1993 and published soon thereafter [3,4]; this work was granted a U.S. patent. While there are other techniques for training a neural network including the back propagation algorithm, these have the limitation of a larger time-consumption and the requirement of substantial training. [5,6,7]

CC4 Approach

This approach is characterized by the classification of training sample outputs to the corners of a multidimensional cube based on the input vectors, as suggested by the name *Corner Classification* [3]. Of the four algorithms that exist in the class, CC4 is the most advanced [8]. The corner classification approach provides instantaneous training and in the basic algorithm the training samples are presented only once as that is sufficient to determine the interconnection

weights. These advantages are of significant value in many applications [9,10,11]. Implementation and applications of CC4 are available in the literature [12,13,14,15,16].

Network Architecture

The CC4 network architecture is built on a three-layered feed forward network of basic binary neurons that use a thresholding function as the activation function. These three layers are: i) Input layer ii) Hidden layer iii) Output layer.

Input layer

For representing the inputs in the input vector, the required number of input neurons equals the sum of input ranges in the input vector. There is one extra neuron, called the *bias neuron*, as the number of input neurons is one more than the number of input elements in a training sample. This extra neuron receives a constant input of 1.

Hidden layer

Each hidden neuron corresponds to a training sample in the training set implying that the hidden neurons and training samples are equal in number. Further, each hidden neuron is connected to all of the input neurons.

Output layer

Neurons equal in number to that of the minimum number of bits required to present an output constitute the outer layer. The output layer is fully connected; each hidden neuron is connected to each and every output neuron so as to affirm membership effectively.

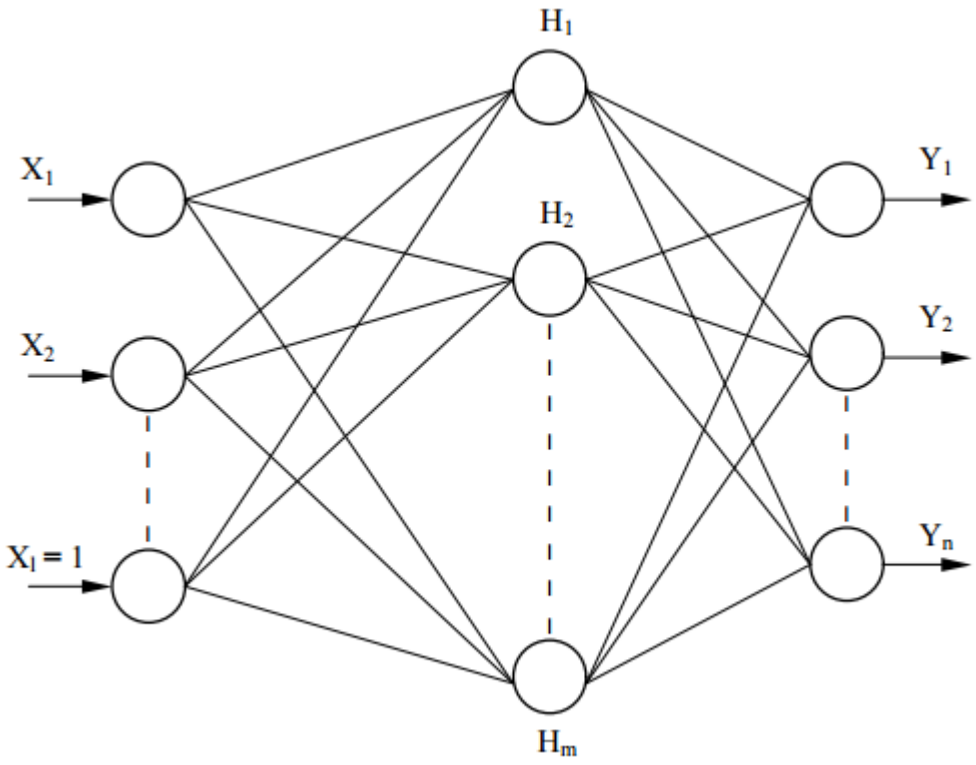


Figure 2: General network architecture for CC4

Functioning of CC4

Two novel ideas which enable a network to learn and generalize form the basis of CC4 algorithm functioning. Learning is the process of assigning weights to connections between the three layers of neurons. This network follows Prescriptive Learning, a process where the input and output vectors of the training samples in the training set are inspected and weights assigned to the links between corresponding neurons.

Secondly, the concept of radius of generalization helps in the classification of input vectors based on the class of stored vectors. This is checked with respect to the hamming distance between a new input vector and any of the stored vectors. Any value less than or equal to the user-specified radius causes the outputs of all such stored vectors to be considered for generating

output of that input vector. Consequently, the number of 1s and 0s in every bit location of the output vector of all these stored vectors is calculated and cumulated. Finally, the output neuron outputs a 1 for a positive result while a 0 for a negative result.

By inspecting the training samples, both input and output weights to the hidden neurons are prescribed. The weight of the link between an input neuron and the corresponding hidden neuron is assigned a 1 or a -1 based on the input neuron receiving 1 or 0 respectively. In a similar manner, outputs in the output vector determine the weight of the link between an output neuron and the corresponding hidden neuron to be set to either a 1 or a -1. However, this is different in the case of the extra neuron, the bias neuron. The number of 1's in the input vector of the training sample is represented by *s* and the weight between the bias neuron and the hidden neuron corresponding to this input vector is set to *r-s+1*, where *r* is the user-specified radius of generalization. The following equation summarizes the weights for the links between input layer and the hidden neurons

$$W_i[j] = \begin{cases} 1 & \text{if } x_i[j] = 1 \\ -1 & \text{if } x_i[j] = 0 \\ r\text{-}s+1 & \text{if } j = n. \end{cases}$$

Weights to the links between the hidden layer neurons and the output neurons is built the same way except for the absence of the extra bias neuron as opposed to that of the input layer. The working of CC4 algorithm can now be understood better by considering *r* to be 0. Now for every training sample presented, the vector combines with the input interconnection weights as the sum of the products of each input element with its corresponding interconnection weight element. The hidden neuron corresponding to the training sample presented receives a +1 input from all the input neurons presented with a binary "1". There are s such input neurons, which receive a +1 input from the training sample since 7 there are s ones in the sample. Since r is zero,

the input to the hidden neuron from the bias neuron is -s + 1 and the net input received by the hidden neuron is s – s + 1 = 1. The other hidden neurons will receive a negative or 0 input because the positions of the +1 and -1 weights don't match the positions of all the binary "1" inputs and the binary "0" inputs respectively. This ensures that only the corresponding hidden neuron fires and produces a binary "1" output since the threshold of the activation function is set as zero. Thus the similar combination of the hidden neuron outputs and the output layer weights produces the desired final outputs. The working of the algorithm can be illustrated by the following examples. The first is the simple XOR example.

Example 1

The following example shows a trained network performing an XOR function. The truth table is shown in Table 1. There are two input elements in each input vector. Thus three input neurons are required including the bias neuron. No generalization is desired here. Hence the radius of generalization r is set as zero. Here all four input vectors are required to train the network. Thus four hidden neurons are used. These hidden neurons are connected to a single output neuron. [6]

Table 1: Inputs and outputs for Example 1; XOR truth table

Inputs		Output
X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

The weights at the input layer and output layer are assigned according to the algorithm. The input vectors are then presented one at a time and the combination of each with the weights causes only one hidden neuron to fire. For the vector (0 1 1), the number of binary ones, s, is 1, without

the bias bit 1. Hence we get the weight vector to the hidden neuron corresponding to the sample as (-1 1 0). Thus the total input to the hidden neuron is (0 * -1) + (1 * 1) + (1 * 0) = 1. The input to the other hidden neurons is either 0 or negative. Similarly the network is trained by presenting the other vectors one after the other and then tested successfully. The network diagram is shown in Figure 3, and Table 2 contains the network parameters obtained during the training.

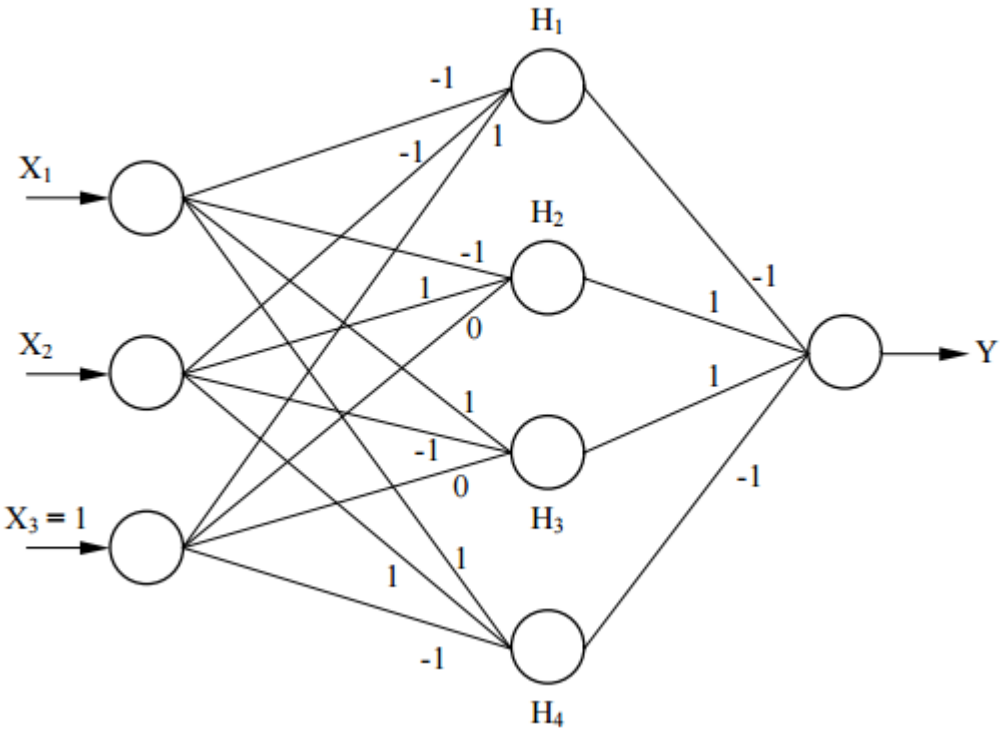


Figure 3: The Network Architecture for Example 1

Table 2: Network Parameters in the input/output mapping of Example 1

							Input to				Output of				Input	Output
Inputs			s	Weights			H_1	H_2	H_3	H_4	H_1	H_2	H_3	H_4	to Y	Y
0	0	1	0	-1	-1	1	1	0	0	-1	1	0	0	0	-1	0
0	1	1	1	-1	1	0	0	1	-1	0	0	1	0	0	1	1
1	0	1	1	1	-1	0	0	-1	1	0	0	0	1	0	1	1
1	1	1	2	1	1	-1	-1	0	0	1	0	0	0	1	-1	0

The standard CC4 network uses unary coding for the input. In this thesis we employ a generalization of unary coding called extended unary coding, which was proposed very recently [17]. The advantage of extended unary coding is that it uses fewer high values of the signal than standard unary coding, and it, therefore, can reduce the power consumption in the hardware designed to do this processing. The main contribution of the thesis is to show that extended unary coding works as well as standard unary coding, although its performance is not identical.

CHAPTER III

TRAINING OF THE NEURAL NETWORKS USING UNARY CODING

Introduction

The simplest way to represent natural numbers is by the base-1 system, called the Unary Number System. Representation of any number *n* in unary coding is done in two ways: *n* ones followed by a zero or *n* zero bits followed by a 1. Other representations use *n-1* ones followed by a zero or with the corresponding number of zeroes followed by a one. Here we use the mapping of the left column of Table 3.

N	Unary code	Alternative code
0	0	0
1	10	01
2	110	001
3	1110	0001
4	11110	00001
5	111110	000001
6	1111110	0000001
7	11111110	00000001
8	111111110	000000001
9	1111111110	0000000001
10	11111111110	00000000001

Table 3. An Example of the Unary Code

The unary number system may also be interpreted as a space coding of numerical information where the location determines the value of the number. In order to denote a specific value, corresponding slot should be marked. This is shown in Figure 4.

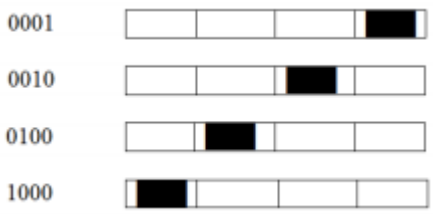


Figure 4. Space representation of numbers

Neural network learning

Fixed length unary coding was used in instantaneously trained neural networks. This enabled learning all points adjacent by Hamming distance to a specific point by learning about that point. It was the uniform property of distance that made instantaneous learning possible since representing the learnt point was straightforward. Once a data point has been recognized, one can achieve generalization by wrapping a region of Hamming radius *r* units around it.

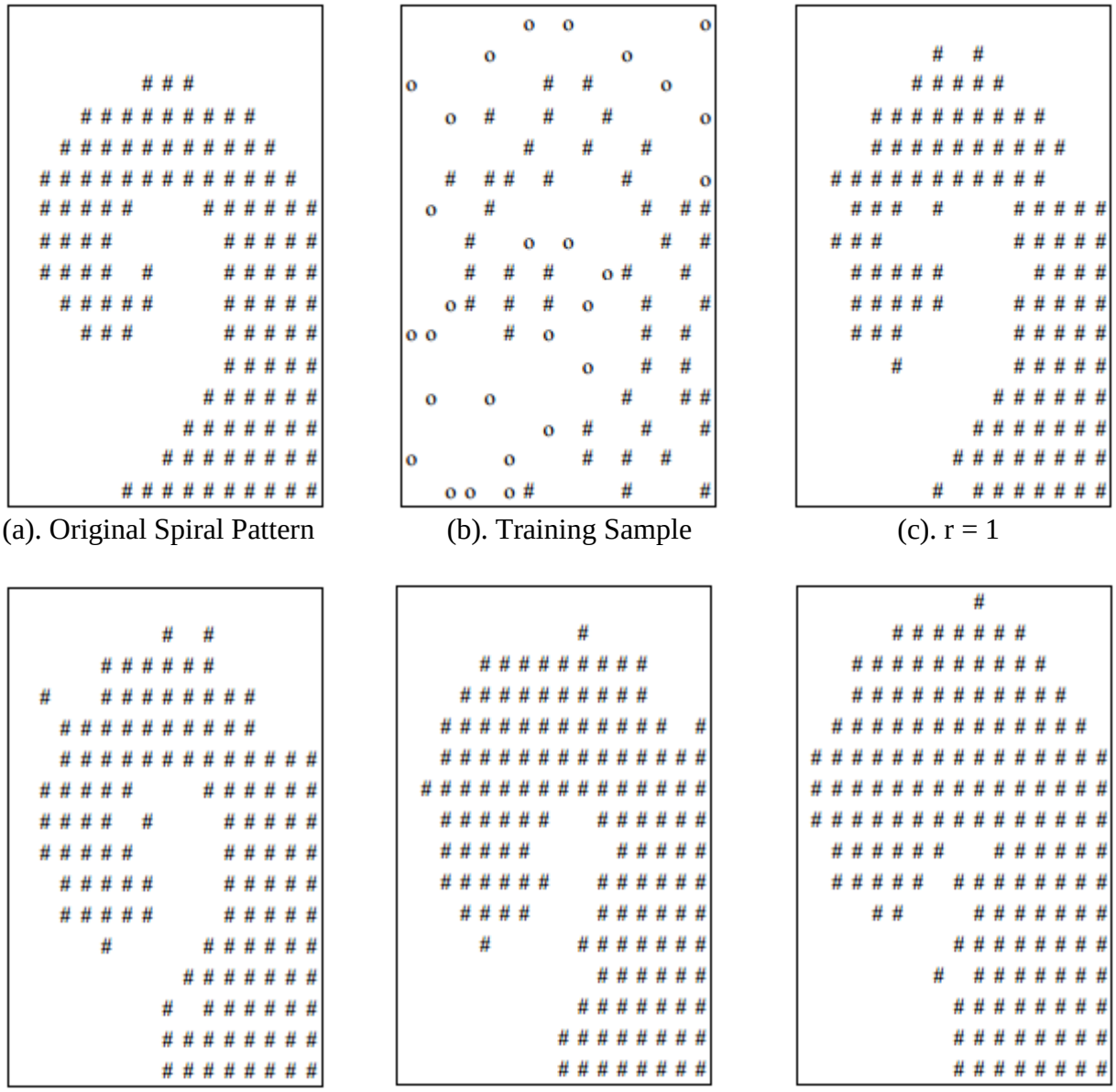
CC4 algorithm used unary coding because it has the advantage of providing an ordered and linear measure of distance, related to the number of ones in the coded sequence. The unary encoding converts numbers 1 through 16 into strings, each of length 16 bits. As an example number 1 is represented by a string that has fifteen 0's followed by a single 1. Similarly, number 2 is represented by fourteen 0's followed by two 1's and so on. Thus the set of numbers from 1 through 16 is represented by strings belonging to the range 0000 0000 0000 0001 to 1111 1111 1111 1111. To represent a point in this pattern, the 16 - bit strings of the row and column coordinates are concatenated together.

To this, another bit that represents the bias is added and the resulting 33-element vector is given as input to the network.

The training samples are randomly selected points from the two regions of the pattern. The samples used here are shown in Figure 5 (b). Points marked "#" are the points from the black region and points marked "o" are points from the white region. A total of 75 points are used for

training. Thus the network used for this pattern classification experiment has 33 neurons in the input layer and 75 neurons in the hidden layer. The output layer requires only one neuron to display a binary “0” or “1”.

After training is done the network is tested for all 256 points in the 16 by 16 area of the pattern as the value of *r* is varied from 1 to 4. The results of different levels of generalization achieved are presented in Figure 5 (c), (d), (e) and (f). It can be seen that as the value of *r* is increased the network tends to generalize more points as belonging to the black region. This over generalization is due to the difference in the density of samples presented from black and white regions during training, the former being greater.



(a). Original Spiral Pattern (b). Training Sample (c). r = 1

(d). r = 2 (e). r = 3 (f). r = 4

Figure 5: Results of Spiral Pattern Classification using Unary Coding

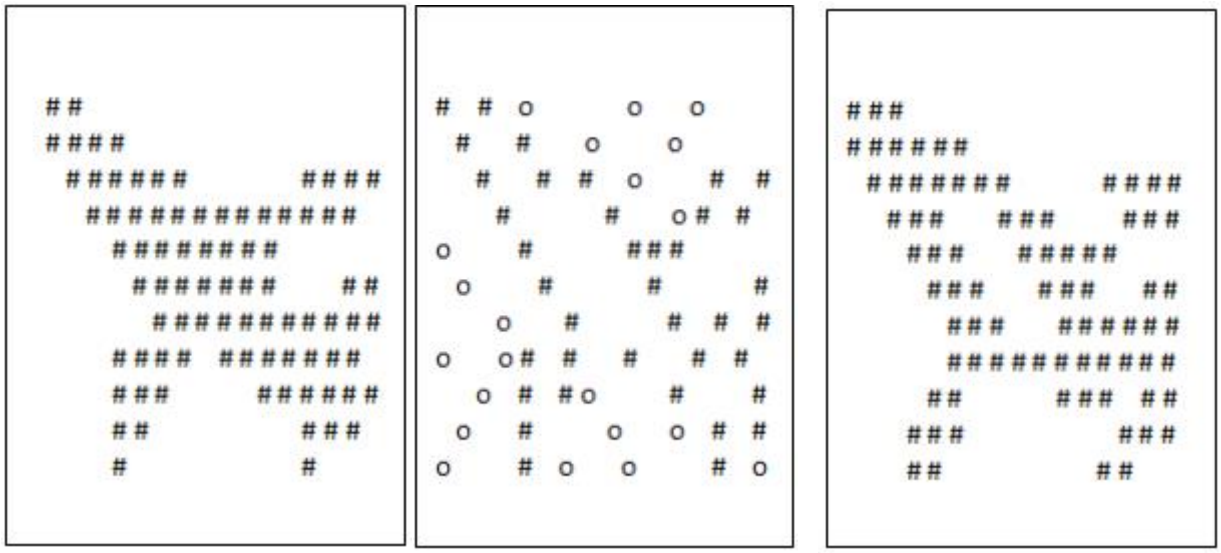
Table 4: No. of points classified/misclassified in the spiral pattern using Unary Coding

	No. of Points			
	r = 1	r = 2	r = 3	r = 4
Classified	237	234	227	207
Misclassified	19	22	29	49

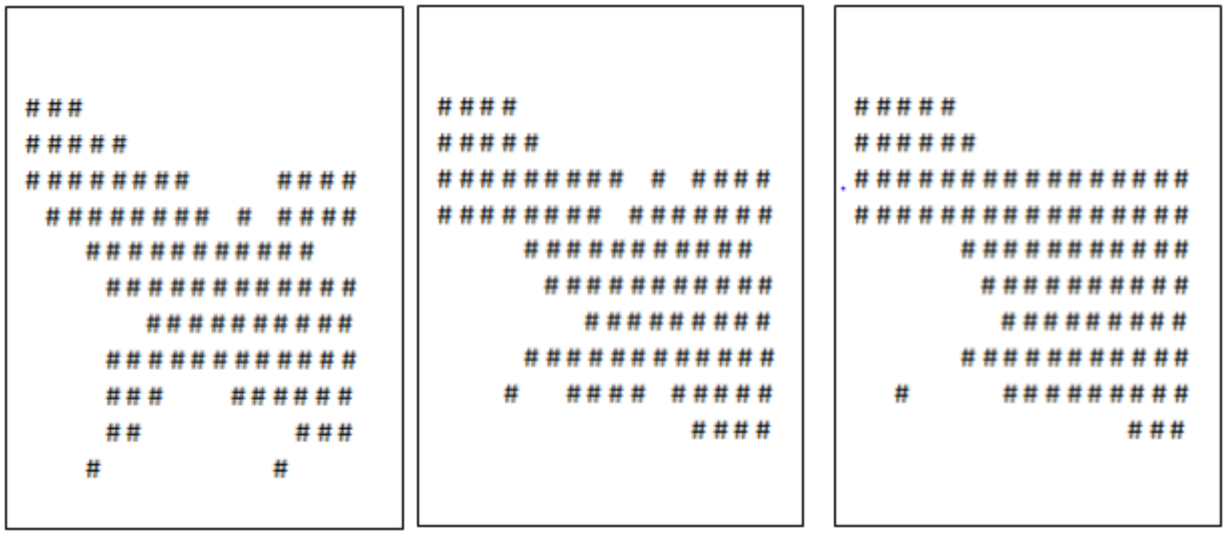
The original plane pattern to show training and performance is shown in Figure 6 (a). The pattern consists of two regions; dividing 11x16 area into a black spiral shaped region, in which a point is represented as 1 and a point in the white region as 0. Any point in this region is represented by row and column coordinates. These coordinates are encoded using 16–bit unary encoding and fed as inputs to the network. The corresponding outputs are 1 or 0, representing the region that the point belongs to. [9]

The training samples are randomly selected points from these two regions of the pattern. Figure 6(b) is an example of a plane pattern used for testing the CC4 network. The points marked "#" represents a positive learnt location while "o" represents a negative learnt location and all blank spaces represent not learnt locations. This matrix is used to train the neural network using the CC4 algorithm.

After the training is done, the network is tested for all 176 points in the 11×16 area of the pattern as the value of *r* is varied from 1 to 4. The results for different values of *r* are shown in the Figures 6(c), (d), (e), and (f). As the value of *r* increases, the network tends to generalize more points belonging to the black region.



(a). Original Plane Pattern (b). Training Sample (c). r = 1

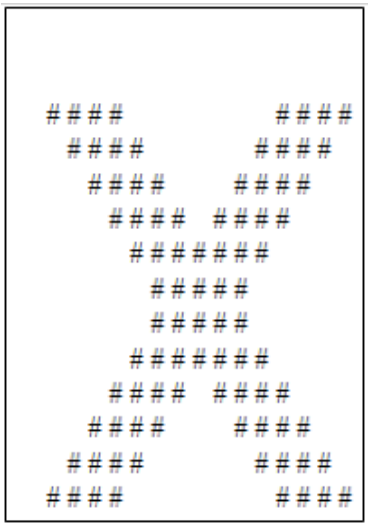


(d). r = 2 (e). r = 3 (f). r = 4

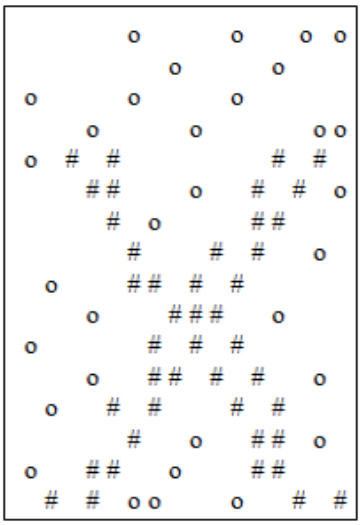
Figure 6: Results of Plane Pattern classification using Unary coding

Table 5: No. of points classified/misclassified in the Plane Pattern using unary coding

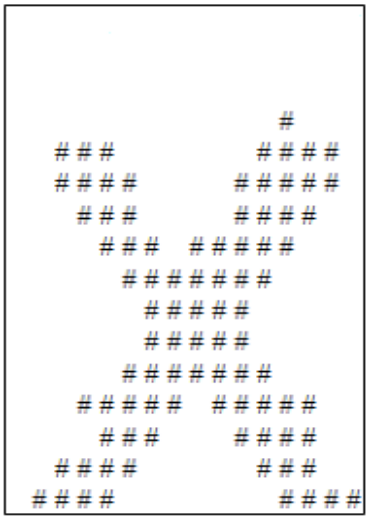
	No. of Points			
	r = 1	r = 2	r = 3	r = 4
Classified	145	154	140	130
Misclassified	31	22	36	46



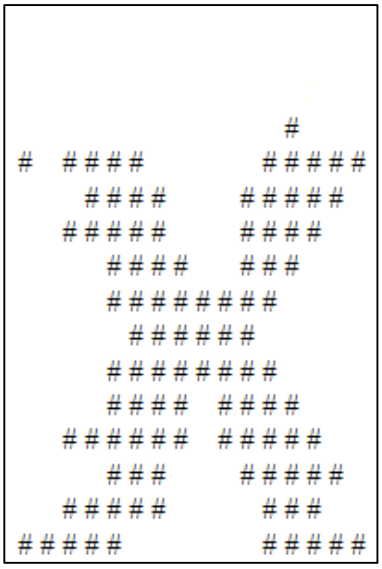
(a). Original Cross Pattern



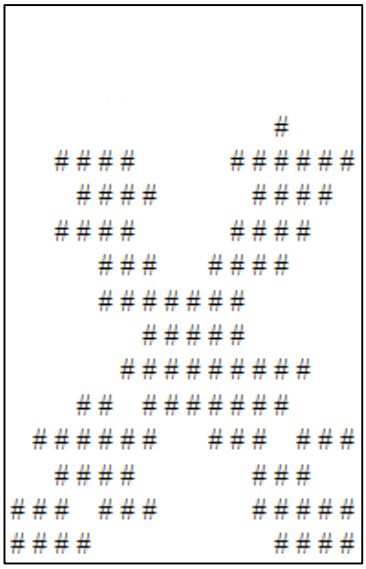
(b). Training Sample



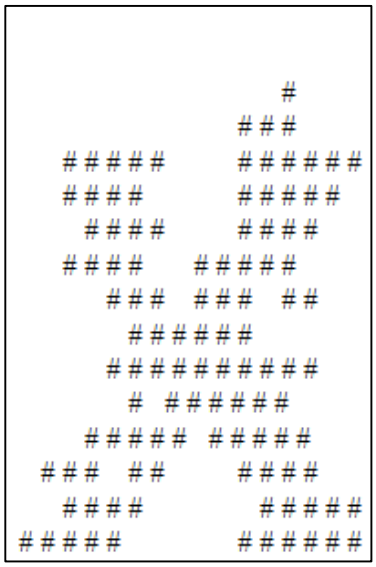
(c). r = 1



(d). r = 2



(e). r = 3



(f). r = 4

Figure 7: Results of Cross Pattern classification using Unary coding

Table 6: No. of points classified/misclassified in the Cross Pattern using unary coding

	No. of Points			
	r = 1	r = 2	r = 3	r = 4
Classified	142	147	138	125
Misclassified	34	29	38	51

CHAPTER IV

TRAINING OF THE NEURAL NETWORKS USING SPREAD UNARY CODING

Introduction

Unary coding is useful but it is redundant in its standard form. Unary coding can also be seen as spatial coding where the value of the number is determined by its place in an array. Motivated by biological finding that several neurons in the vicinity represent the same number, a variant form of unary numeration in its spatial form called spread unary coding is used, in which a single number is represented by several 1s. In this spread unary coding, the number of 1s used is the spread of the code. Spread unary coding is associated with saturation of the Hamming distance between code words [17].

In spread unary coding, for a given spread *k*, the number *n* can be represented by *n-1* 0s followed by *k* 1s.

n: 11… (k times)00… (n-1 times)

Table 7: Spread Unary code for k = 3 for numbers 0 to 10

n	*k* = 3
0	000000000000
1	000000000111
2	000000001110
3	000000011100
4	000000111000
5	000001110000
6	000011100000
7	000111000000
8	001110000000
9	011100000000
10	111000000000

For any two given numbers, *n1* and *n2*, and for spread *k*, let the distance between them be *'d'*. The value of *d* depends on the difference between the two numbers. Specifically, as long as the two numbers differ by a value less than the spread *k*, the distance between them is simply twice the difference between them i.e. *2*(n2-n1)*. The distance is independent of the value of *k*. However, once the numbers differ by a value greater than or equal to *k*, distance between them, now becomes constant at *2k* [17].

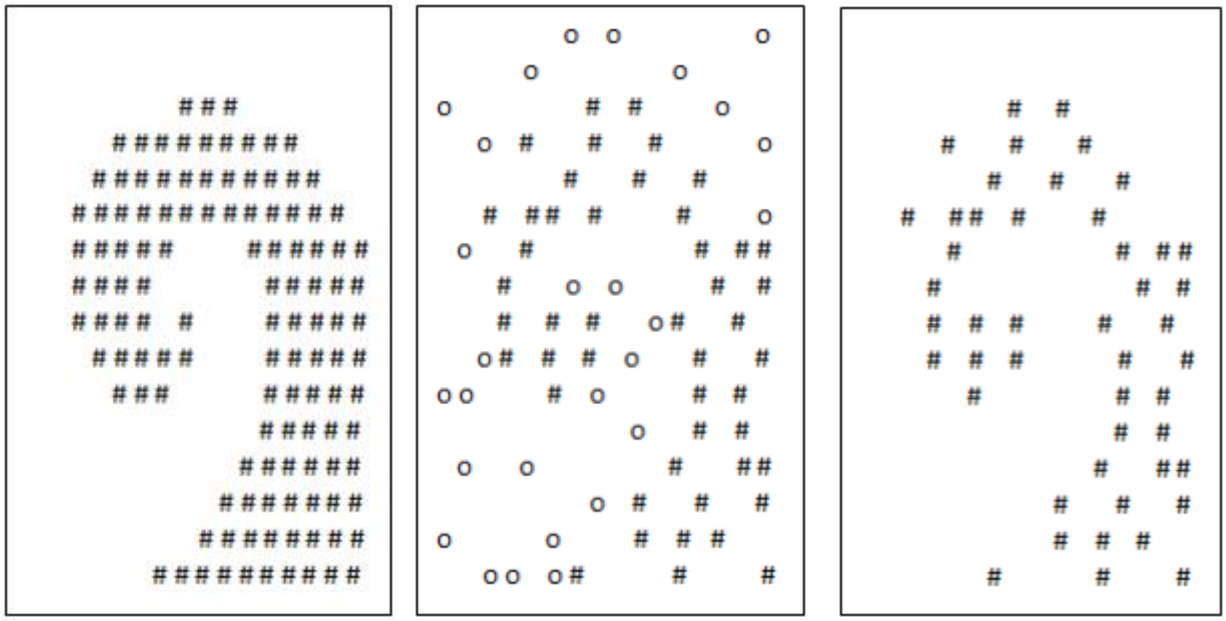
Training of Patterns

The original pattern to show training and performance is shown in Figure 8 (a). The pattern consists of two regions; dividing a 16x16 area into a black spiral shaped region, in which a point is represented as 1 and a point in the white region is represented as 0. Any point in the region is represented by row and column coordinates. These coordinates are encoded using the spread unary coding and fed as inputs to the network. The corresponding outputs are 1 or 0, to denote the region the point belongs to.

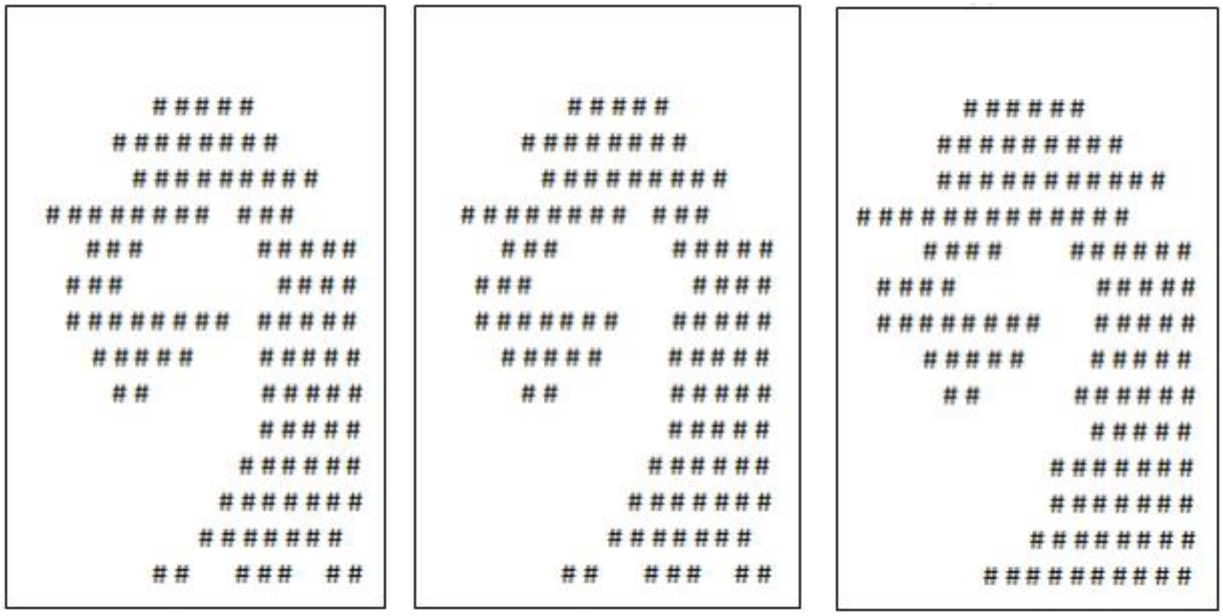
The spread unary code converts numbers 1 to 16 into strings, each of length 18 bits. As an example the number 1 is represented by a string that has fifteen 0s followed by *k* number of 1s in the first place. To represent a point in the pattern, the 18–bit strings of the row and column coordinates are concatenated together. To this, another bit that represents the bias is added and the resulting 37-element vector is given as the input to the network. [9]

The training samples are randomly selected points from the two regions of the pattern. Figure 8(b) is an example of a spiral image pattern used for testing the CC4 network. The points marked "#" represents the location is learnt positive, "o" represents the location is learnt negative and all blank spaces represent the locations are not learnt. This matrix is used to train the neural network using the CC4 algorithm.

After the training is done, the network is tested for all 256 points in the 16×16 area of the pattern as the value of *r* is varied from 1 to 4. The results for different values of *r* are shown in the Figures 8(c), (d), (e), and (f). As the value of *r* increases, the network tends to generalize more points belonging to the black region.



(a). Original Spiral Pattern (b). Training Sample (c). r = 1



(d). r = 2 (e). r = 3 (f). r = 4

Figure 8: Results of Spiral Pattern classification using Spread Unary coding

Table 8: No. of points classified/misclassified in the Spiral Pattern using spread unary coding

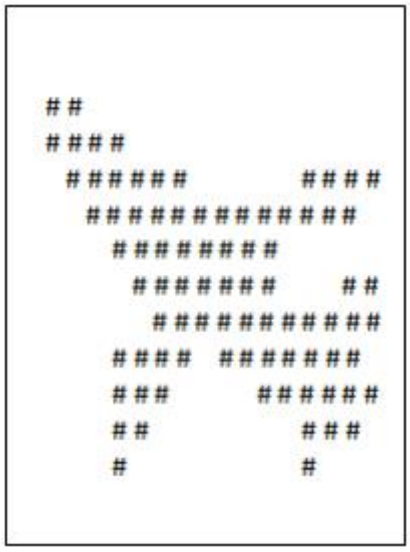
	No. of Points			
	r = 1	r = 2	r = 3	r = 4
Classified	179	236	235	237
Misclassified	77	20	21	19

The original pattern to show training and performance is shown in Figure 9 (a). The pattern consists of two regions; dividing a 11x16 area into a black plane shaped region, in which a point is represented as 1 and a point in the white region is represented as 0. Any point in the region is represented by row and column coordinates. These coordinates are encoded using the spread unary coding and fed as inputs to the network. The corresponding outputs are 1 or 0, to denote the region the point belongs to.

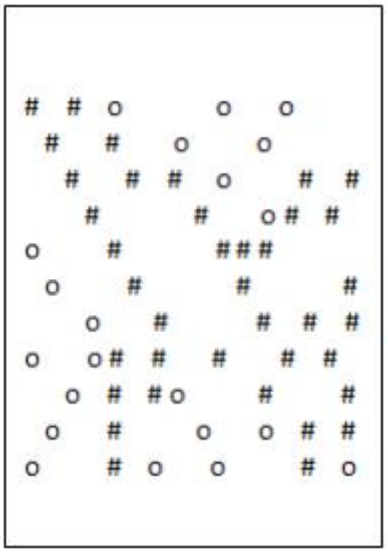
The spread unary code converts numbers 1 to 16 into strings, each of length 18 bits. As an example the number 1 is represented by a string that has fifteen 0s followed by *k* number of 1s in the first place. To represent a point in the pattern, the 18–bit strings of the row and column coordinates are concatenated together. To this, another bit that represents the bias is added and the resulting 37-element vector is given as the input to the network.

The training samples are randomly selected points from the two regions of the pattern. Figure 9(b) is an example of a plane image pattern used for testing the CC4 network. The points marked "#" represents the location is learnt positive, "o" represents the location is learnt negative and all blank spaces represent the locations are not learnt. This matrix is used to train the neural network using the CC4 algorithm.

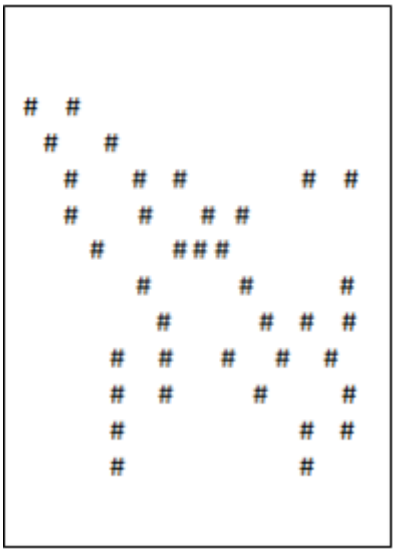
After the training is done, the network is tested for all 176 points in the 11×16 area of the pattern as the value of *r* is varied from 1 to 4. The results for different values of *r* are shown in the Figures 9(c), (d), (e), and (f). As the value of *r* increases, the network tends to generalize more points belonging to the black region.



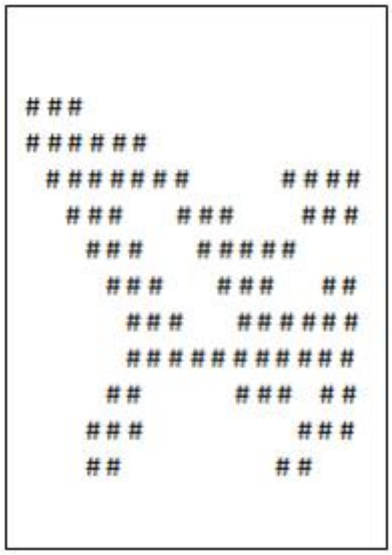
(a). Original Plane Pattern



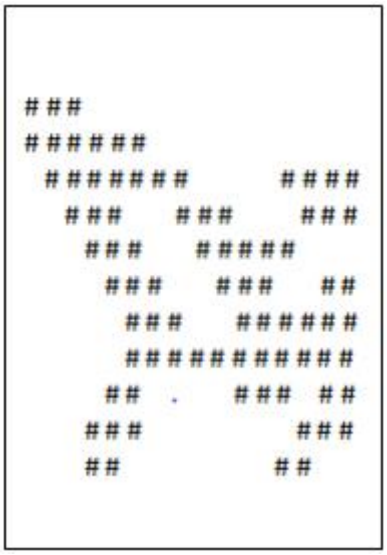
(b). Training Sample



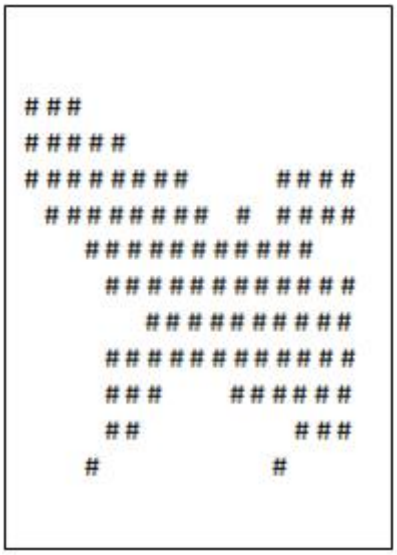
(c). r = 1



(d). r = 2



(e). r = 3

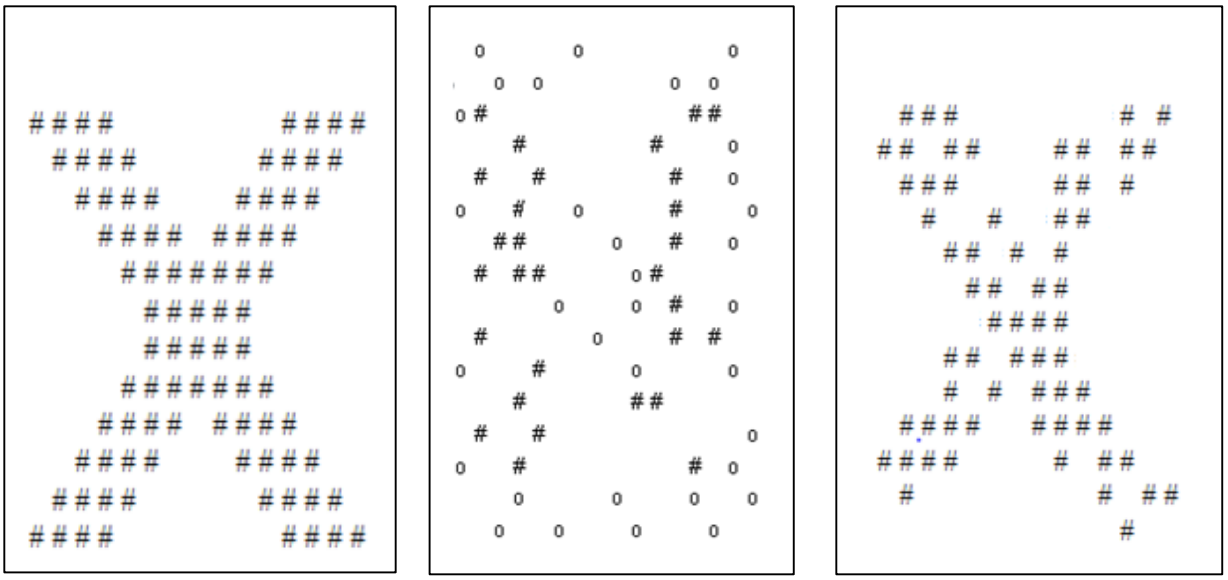


(f). r = 4

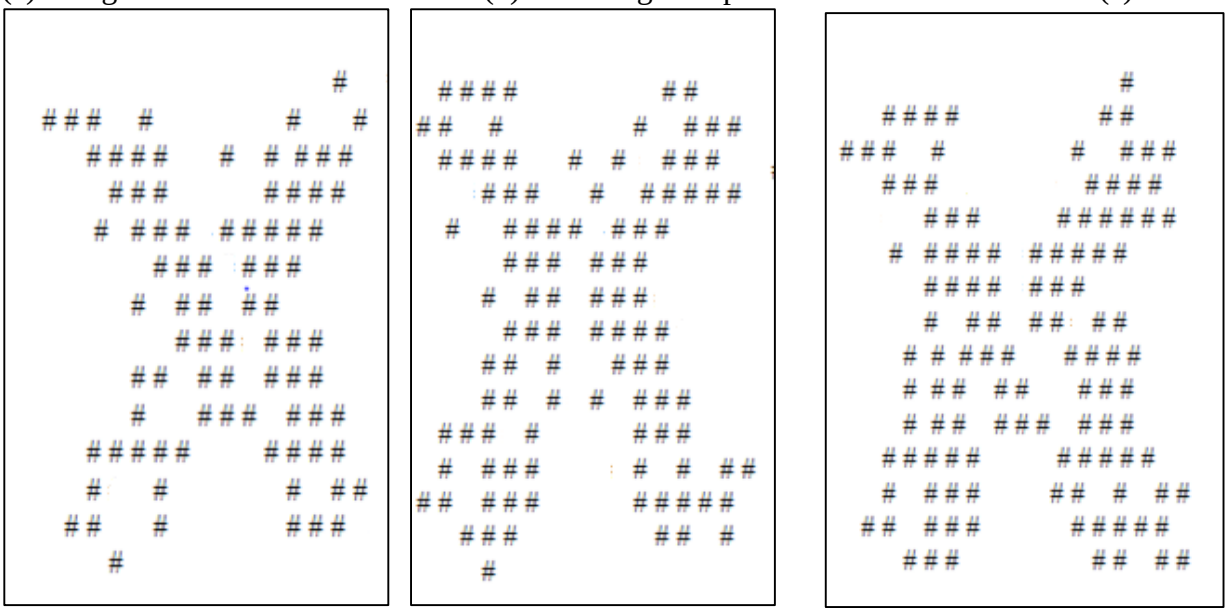
Figure 9: Results of Plane Pattern classification using Spread Unary coding

Table 9: No. of points classified/misclassified in the Plane Pattern using unary coding

	No. of Points			
	r = 1	r = 2	r = 3	r = 4
Classified	117	146	146	132
Misclassified	59	30	30	24



(a). Original Cross Pattern (b). Training Sample (c). r = 1



(d). r = 2 (e). r = 3 (f). r = 4

Figure 10: Results of Cross Pattern classification using Spread Unary coding

Table 10: No. of points classified/misclassified in the Cross Pattern using unary coding

	No. of Points			
	r = 1	r = 2	r = 3	r = 4
Classified	193	179	152	133
Misclassified	63	77	104	123

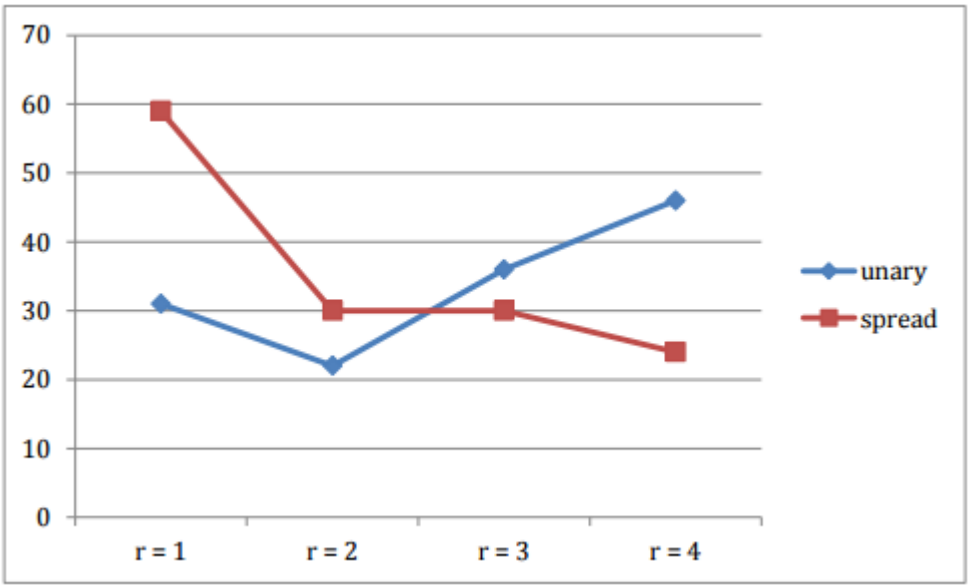


Figure 11: No. of misclassified points for plane patterns using unary vs spread unary

As shown in Figure 11, the number of misclassified points remains stable after the initial drop. This would make spread unary advantageous to use in many situations. Of course, we do not know if some version of CC4 is at the basis of biological learning. If it is, then this might be another reason why Nature has chosen it over other forms of neural network learning.

CHAPTER V

TIME SERIES PREDICTION

The Mackey-Glass time series, originally developed to model white blood cell production as presented by Azoff [18] is commonly used to test the performance of neural networks. The series is a chaotic time series making it an ideal representation of the nonlinear oscillations of many physiological processes. The discrete time representation [8] of the series was used to test the performance of the CC4 algorithm. The discrete time representation of the Mackey-Glass equation is given below: -

$$x(k+1) - x(k) = \alpha x(k-\tau) / \{1 + x\,\gamma\,(k-\tau)\} - \beta x(k)$$

The values of the different parameters in the equation are assigned as follows: -

$$\alpha = 3,\ \beta = 1.0005,\ \gamma = 6,\ \tau = 4$$

Here four samples are required to predict a new point, so $\tau = 4$. Thus the series is started with four arbitrary samples:

$$x(1) = 1.5,\ x(2) = 0.65,\ x(3) = -0.5,\ x(4) = -0.7$$

A series of 200 points within a range -2 to +2 is generated using these samples. Of these 200 points, 180 points are fed to the networks for training known as training set. Then the network is tested using the remaining 20 points known as testing set. In the training and the testing, four consecutive points in the series are given as input and the next point is used as the output. Thus a sliding window of size four is used at each and every step. The total number of sliding windows available in the network is 175, as the number of points used for the training is 180. The first window consists of points 1 to 4 with the 5th as the output whereas the last window consists of points 175 to 178 with the 179th point as the output.

Since four points are required in training or testing, the 16 characters standard unary code words for each of the four inputs are concatenated together. Thus each input vector has 65 elements, where the last element in the vector represents the bias. Unlike the inputs, output points are binary encoded using four bits. This is done to avoid the possibility of generating invalid output vectors that would not belong to the class of expected vectors of the quaternary encoding scheme. Hence 21 neurons are required in the input layer, 175 in the hidden layer (one for each sliding window), and 4 in the output layer.

After the training, the two networks are tested using the same 175 windows to check their learning ability. Then the rest of the windows are presented to the networks to predict future values. The inputs are always points from the original series calculated by the Mackey-Glass equation to avoid an error buildup. The outputs of the network are compared against the expected values in the series. The performance of the CC4 algorithm with unary coding for r varying as 4, 5, 6 and 7 is shown in Figures 12.1, 12.2, 12.3 and 12.4 respectively. Similarly the performance with the spread unary coding for different values of r is presented in the Figures 12.5, 12.6, 12.7 and 12.8. The values of '*r*' are again varied as 4, 5, 6 and 7 respectively.

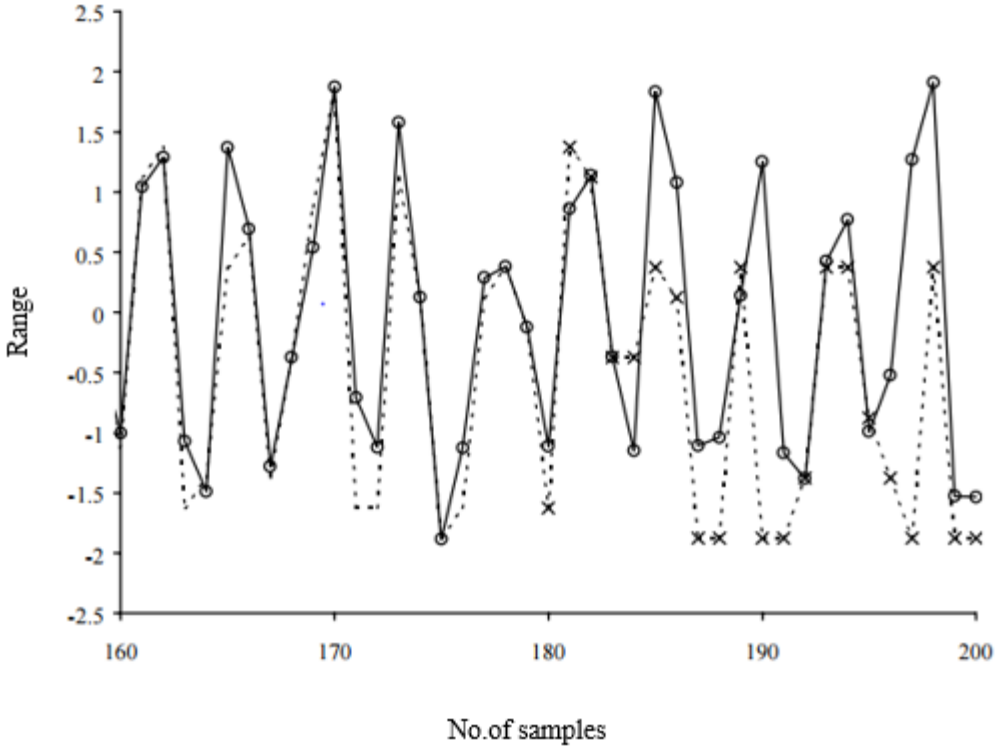


Figure 12.1: Mackey-Glass time series prediction using Unary coding, r = 4

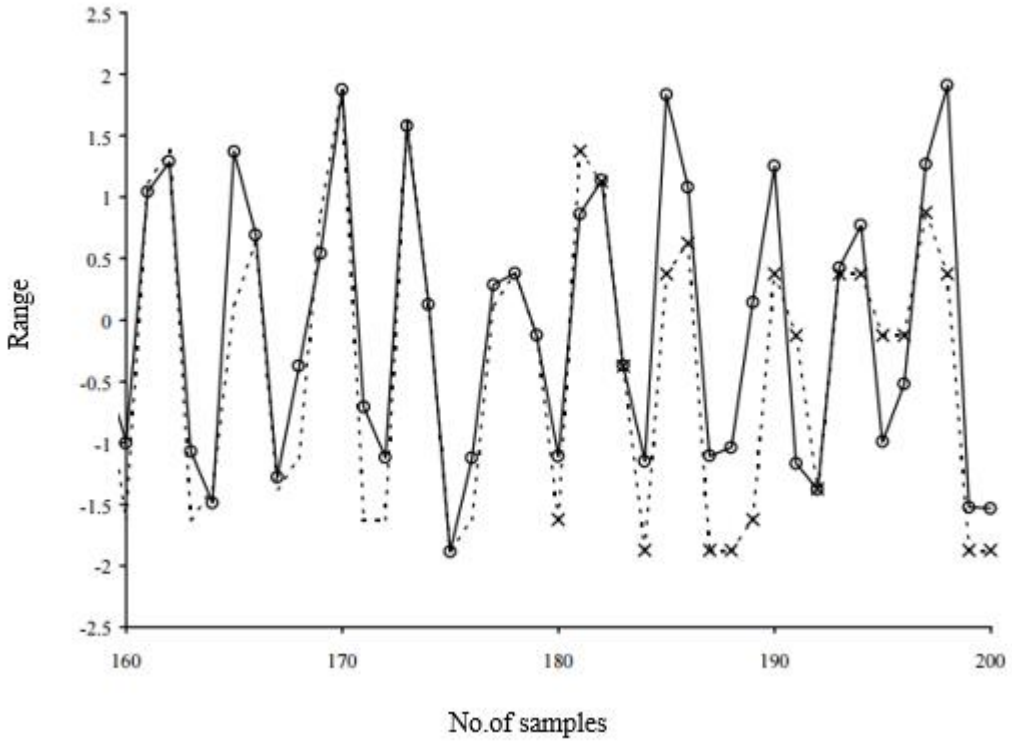


Figure 12.2: Mackey-Glass time series prediction using Unary coding, r = 5

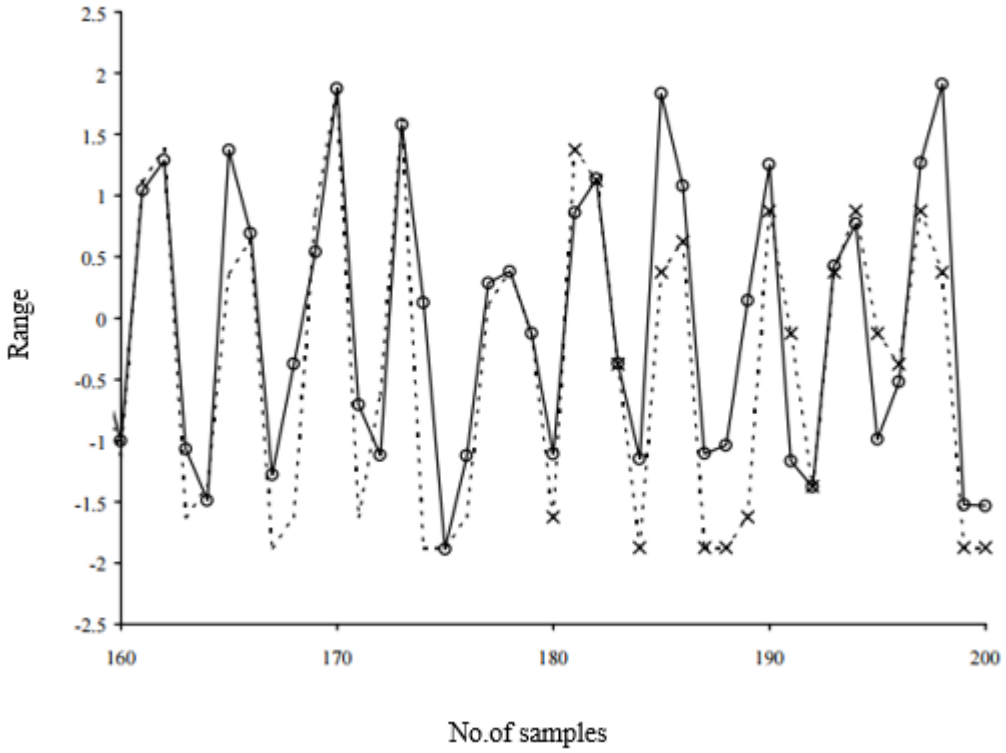


Figure 12.3: Mackey-Glass time series prediction using Unary coding, r = 6

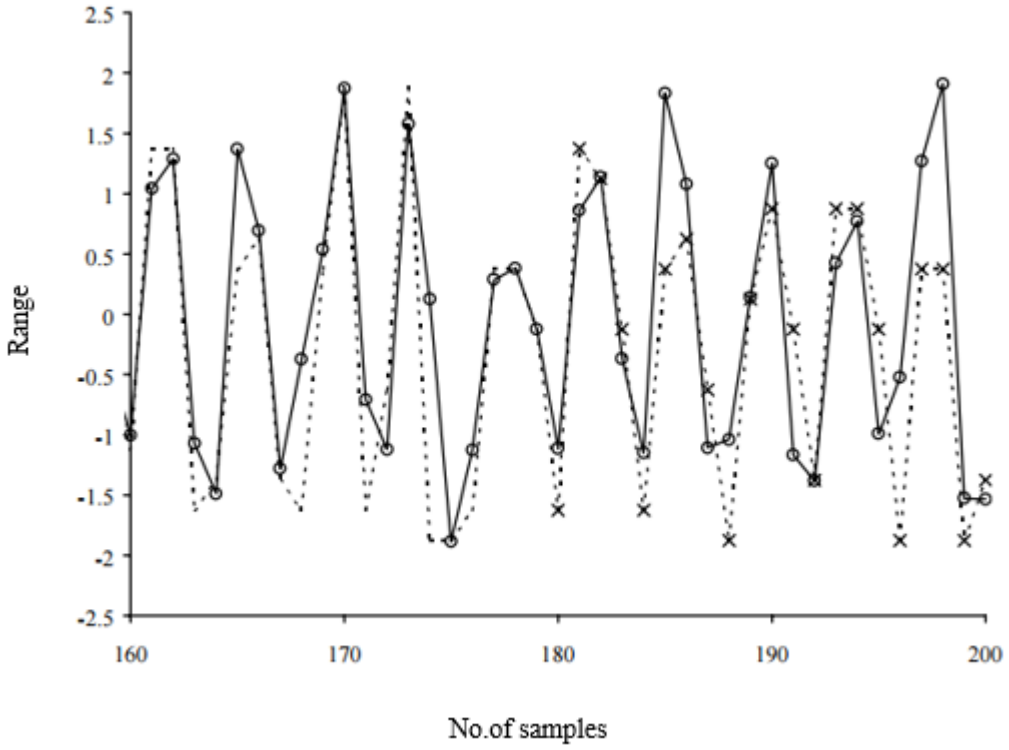


Figure 12.4: Mackey-Glass time series prediction using Unary coding, r = 7

In each of these figures only points 160 to 200 are shown for readability. The solid line represents the original series and the lighter line represents the outputs of the networks designed by the two algorithms. The lighter line from point 160 to 179 shows how well both the networks have learnt the samples for different values of r. The points that are predicted by the networks are represented by "x" on the lighter line. The actual points generated by the Mackey-Glass equation are represented by "o" on the solid line. The first point that is predicted is the point number 180 using the original series points 176, 177, 178 and 179.

The next point that is predicted is 181 using the points 177, 178, 179 and 180. The point number 180, which is used as input here is the original point in the series generated by the Mackey-Glass equation and not the point predicted by the network. Similarly, the last point to be predicted is the point number 200 using the actual points 196 to 199 from the series. The networks always predict one point ahead of time and both the networks predict the turning points in the series efficiently. This ability is of great importance in financial applications, where predicting the turning point of the price movement is more important than predicting the day to day values.

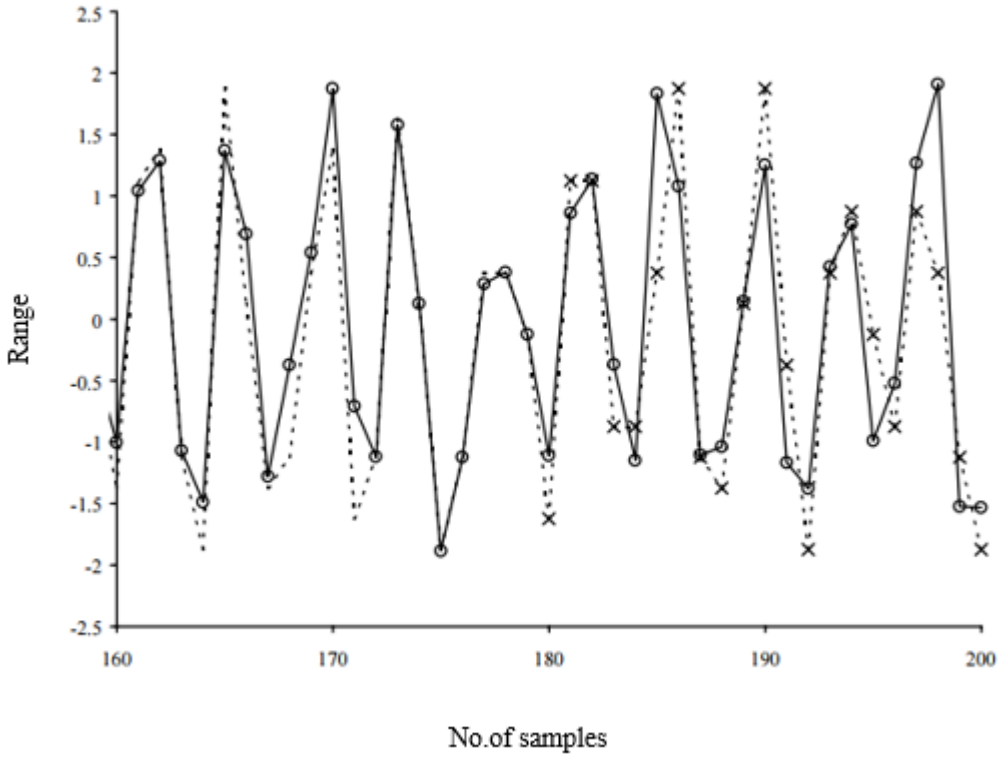


Figure 12.5: Mackey-Glass time series prediction using Spread Unary coding, r = 4

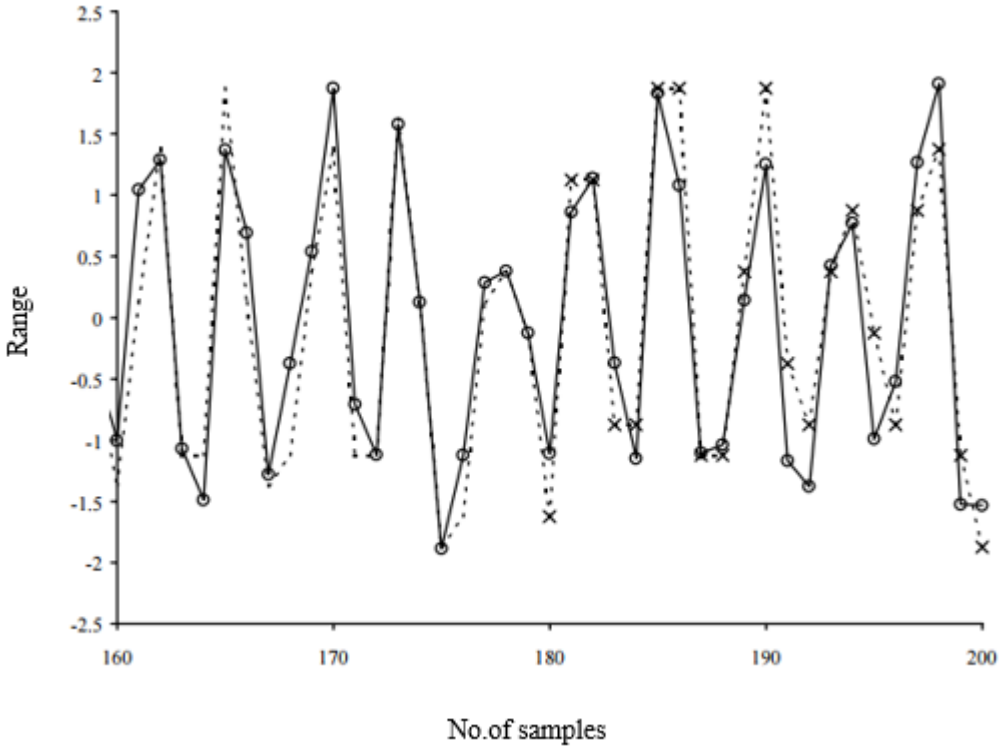


Figure 12.6: Mackey-Glass time series prediction using Spread Unary coding, r = 5

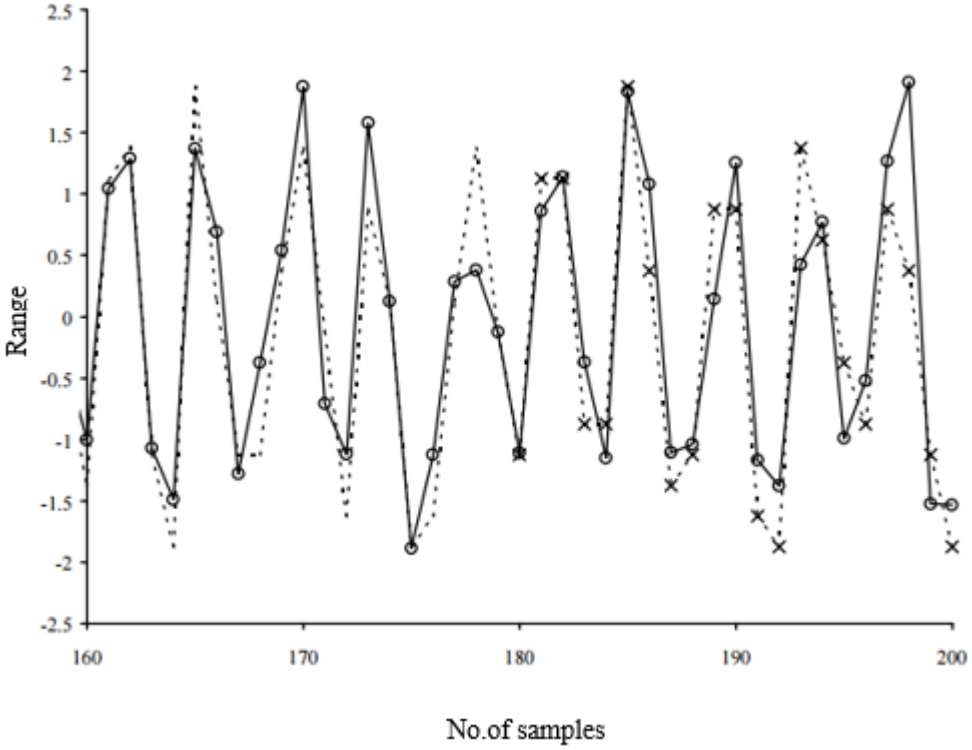


Figure 12.7: Mackey-Glass time series prediction using Spread Unary coding, r = 6

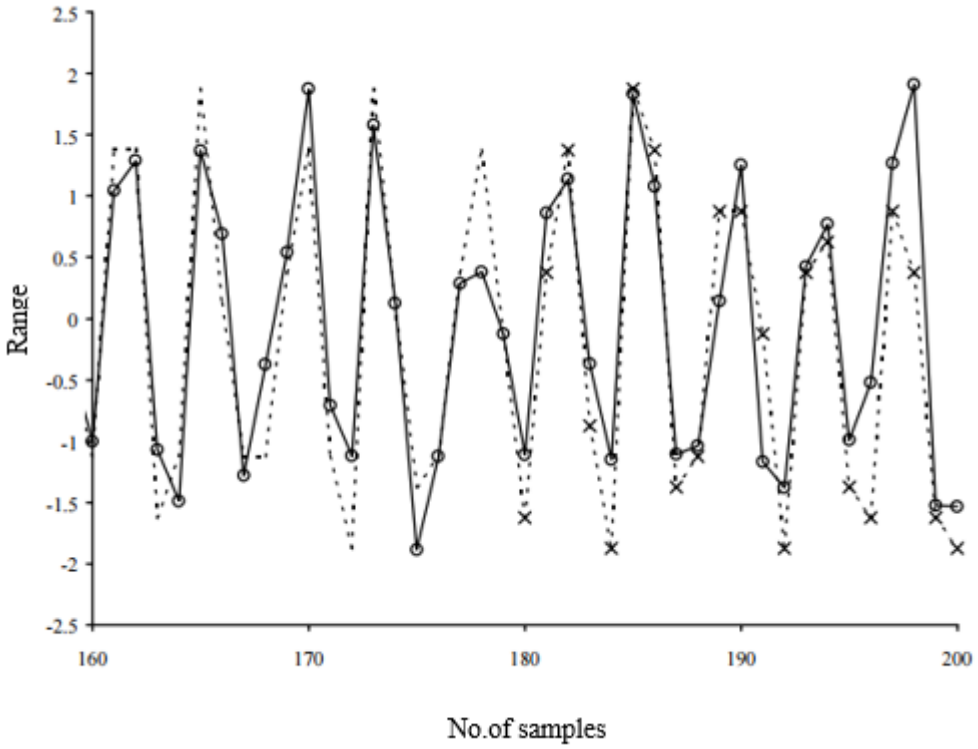


Figure 12.8: Mackey-Glass time series prediction using Spread Unary coding, r = 7

CHAPTER VI

SUNSPOT NUMBERS

Introduction

The mysterious cycles in sunspot numbers has intrigued many investigators, including geologists, astronomers, climatologists, economists, historians, and statisticians [19, 20]. Sunspots are high-intensity electromagnetic flares of solar radiation of largely unknown and unpredictable causes. They have major effects on various terrestrial phenomena; an example is the long-range weather prediction. Sun-weather relationships in the stratosphere are well understood and sunspot numbers have to be taken into account in telecommunications and interplanetary flight. Moreover, sunspot variations are proving important in quaternary chronology and paleoclimatology.

In the past half-century many statisticians have tried to provide a functional relationship between the sunspot numbers. As addressed by Izenman (1985), "... The sunspot numbers have been shown to contain idiosyncrasies that suggest, quite strongly, that the underlying statistical mechanism by which they are generated is non-linear, non-stationary, and non-Gaussian, and as such they are used primarily as a yardstick to compare and judge new statistical modeling and forecasting methods." [18]

Even though sunspot number prediction via statistical methods has provided promising results, new and traditional techniques are being explored with the hope of further improving the results.

In a neural network, which is a directed graph with weighted edges, each node in the network is capable of simple nonlinear processing that typically involves calculating a weighted sum of its inputs and then passing it through a nonlinear function. A series of experiments were performed in which a neural network was trained to follow the curves depicting variations in sunspot numbers. The trained network was used to predict the sunspot data for "future" periods on which the network had not been trained.

Network Training

In each training step an input value, i.e., a normalized sunspot number of some year (average taken from daily sunspot activity), is presented to the network. The network is asked to predict the next value in the sunspot number sequence. The weights between any two nodes are modified in the same way as that in the CC4 algorithm.

The sunspot data taken from Sunspot Index and Long term Solar Observations (SILSO) [19] is used for this experiment. In this experiment only the annual average sunspot numbers are processed, although half yearly data are also available and have been studied in statistical literature. Suppose that the training set consists of a data recorded over several years, let’s say 36 years and assume that I am predicting one value in advance:

1. Partitioning my data into training and validation sets.
2. Taking the first 24 points as training set and remaining as validation set.
3. Create a neural network layout using 3 values as inputs and predicting the fourth value.
4. So a neural network with an input layer consisting of 3 nodes and an output layer containing one node is designed.
5. And for the hidden layer couple of nodes are required.

6. Create the training patterns: Each training pattern will be four values, with the first three corresponding to the input nodes and the last one defining what the correct value is for the output node. For example, if your training data are values

$$x_1, x_2 \ldots, x_{24}$$

then

pattern1: $x_1, x_2 \ldots, x_4$

pattern2: $x_2, x_3 \ldots, x_5$

…

pattern21: $x_{21}, x_{22}, x_{23}, x_{24}$

7. Train the neural network on these patterns

8. Test the network on the validation set (months 25-36): Here you will pass in the three values the neural network needs for the input layer and see what the output node gets set to. So, to see how well the trained neural network can predict month 32's value you'll pass in values for months 29, 30, and 31

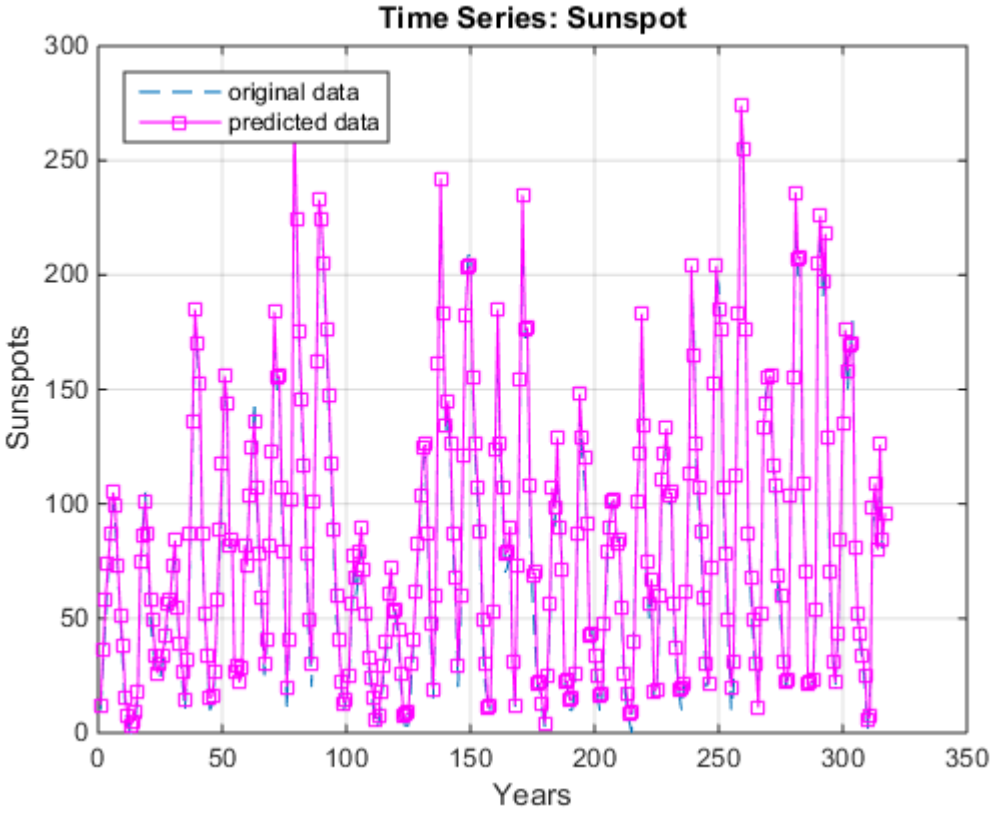


Figure 13. Sunspot Number Predictions in time period 1700-2015

CHAPTER VII

D- SEQUENCES

Introduction

Decimal sequences are obtained when a number is represented in a decimal form in a base r and they may terminate, repeat or be aperiodic. For a certain class of decimal sequences of $1 / q$, q prime, the digits spaced half a period apart add up to $r - 1$, where r is the base in which the sequence is expressed. These sequences are periodic and their randomness properties are checked only in one period. Decimal sequences are known to have good cross correlation properties and they can be used in applications involving pseudorandom sequences. [21,22]

These sequences are categorized into two types based on their definition:

1).The expansion of {1/ q} in base r (non-binary) in this case is given by:

$$a_i = [r^i \bmod q] \bmod r$$

Where q is the prime number and r is the base.

Example 1: Consider the sequence of {1/37} in base 10,

So, the sequence $a_i = [10^i \bmod 37] \bmod 10$

$= [10\ 26\ 1] \bmod 10$

$= [0\ 6\ 1]$

Since the base is 10, the digits in the expansion of the sequence are 0, 1 up to 9. The period of this sequence is 3 after which the digits repeat.

2).This is a case in which the expansion of *{1/ q}* in base r is given by:

$$a_i = [r^i \bmod q] \bmod s$$

Where r is the non-binary base and s = 3.

Example 1: Consider the sequence of {1/13} base 7.

According to the above definition, this sequence can be expanded as:

$$a_i = [7^i \bmod 13] \bmod 3$$

$$= [7\ 10\ 5\ 9\ 11\ 12\ 6\ 3\ 8\ 4\ 2\ 1] \bmod 3$$

$$= [1\ 1\ 2\ 0\ 2\ 0\ 0\ 0\ 2\ 1\ 2\ 1]$$

Time-Series Prediction

Here we consider the prediction of the next d-sequences using the neural networks. As in the earlier material, we first partition the data series into three disjoint sets: the *training set*, the *validation set*, and the *test set*. The network is trained (e.g., with the corner classification algorithm) directly on the training set, its *generalization ability* is monitored on the validation set, and its ability to predict is measured on the test set. A network's generalization ability indirectly measures how well the network can deal with unforeseen inputs, in other words, inputs on which it was not trained.

When using real-world data that is observed (instead of artificially generated), it may be difficult or impossible to partition the data series into three disjoint sets. The reason for this is the data series may be limited in length and/or may exhibit nonstationary behavior for data too far in the past (i.e., data previous to a certain point are not representative of the current trend).

The time series prediction of d-sequences is performed by dividing the data obtained by calculating the d-sequence of a prime number *q* with base *r*. The data obtained is divided into three sets: training data set, validation data set and test set.

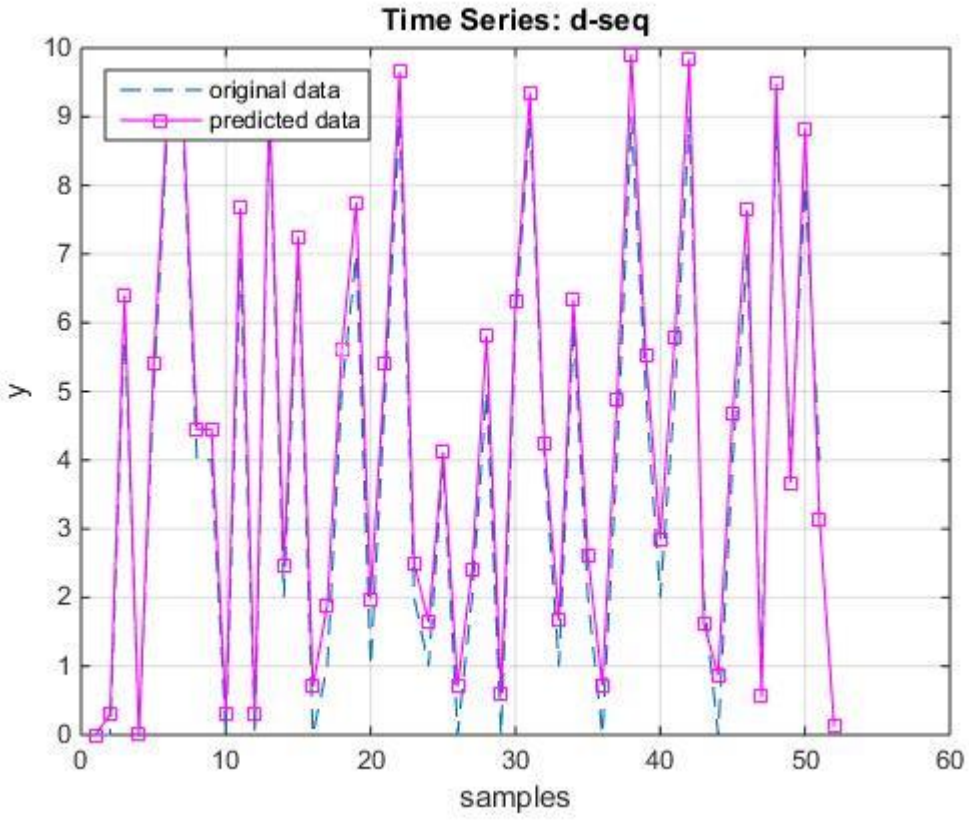


Figure 14. Prediction for the prime number 487

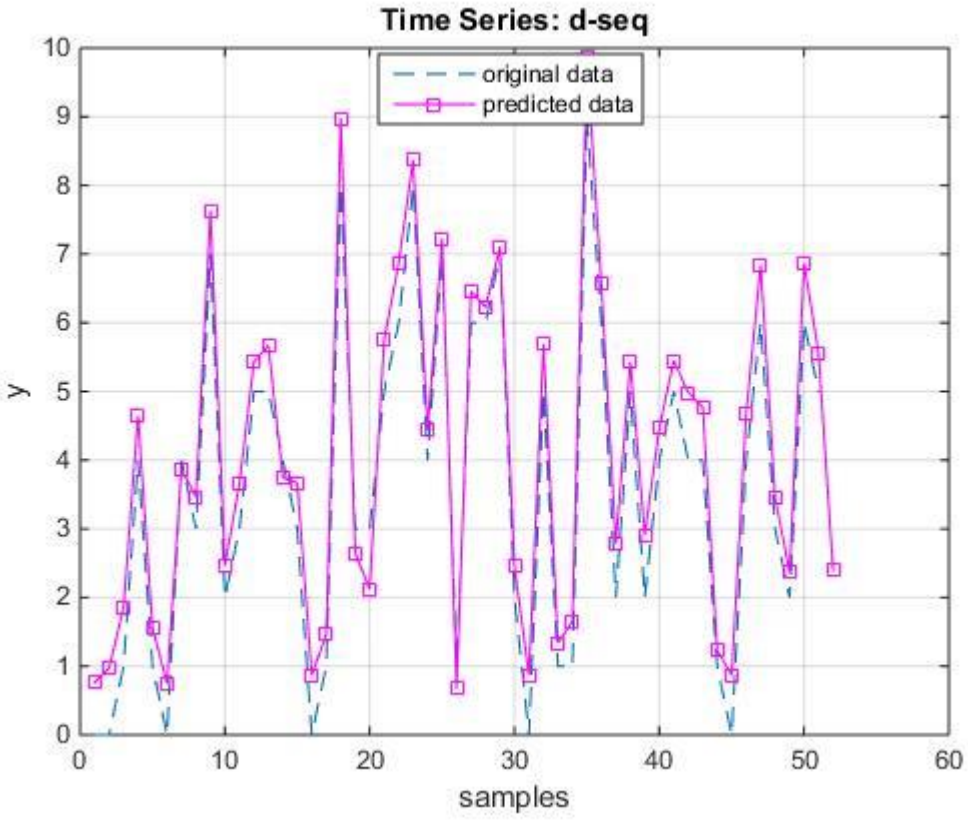


Figure 15. Prediction for the prime number 709

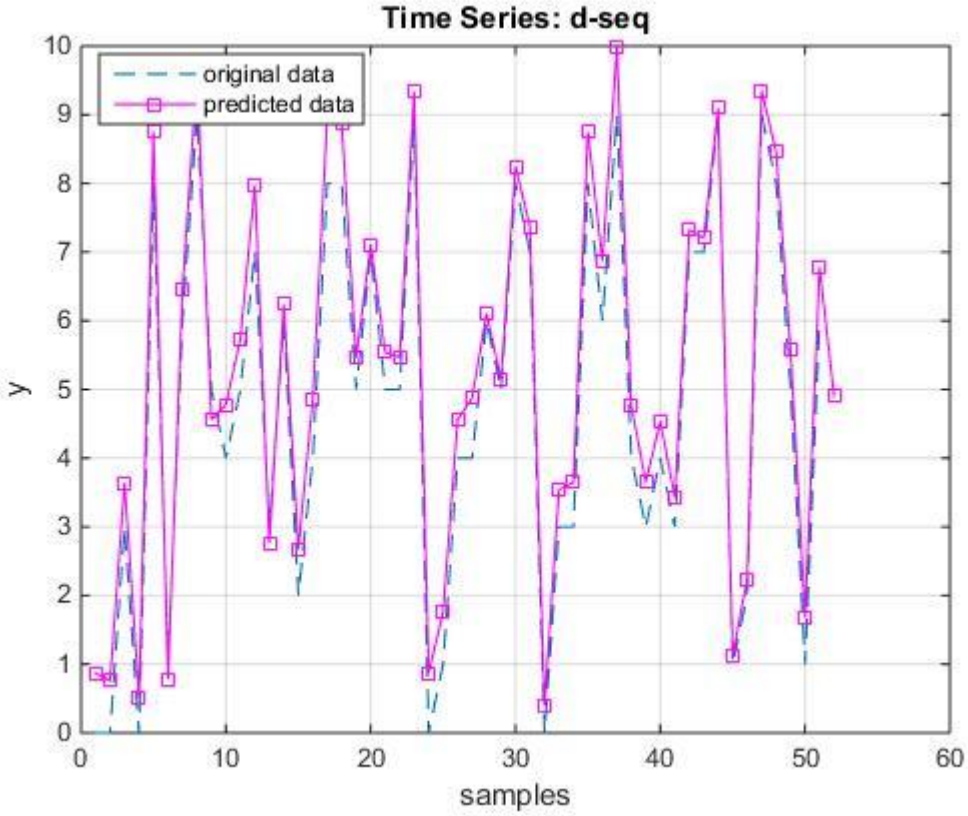


Figure 16. Prediction for the prime number 987

CHAPTER VII

CONCLUSION

In this thesis we employ a generalization of unary coding called extended unary coding, which was proposed very recently. The advantage of extended unary coding is that it uses fewer high values of the signal than standard unary coding, and it, therefore, can reduce the power consumption in the hardware designed to do this processing. The main contribution of the thesis is to show that extended unary coding works as well as standard unary coding, although its performance is not identical.

REFERENCES

1. S. Kak, "Artificial and biological intelligence", *ACM Ubiquity,* vol. 6, No. 42, pp. 1-20, 2005.
2. S. Kak, The Architecture of Knowledge. New Delhi, Motilal Banarsidass, 2004.
3. S. Kak, "On training feedforward neural networks", Pramana J. Physics, vol. 40, pp. 35-42, 1993.
4. S. Kak, "New algorithms for training feedforward neural networks", Pattern Recognition Letters, vol. 15, pp. 295-298, 1994.
5. P. Raina, "Comparison of learning and generalization capabilities of the Kak and the back propagation algorithms", *Information Sciences*, vol. 81, pp. 261-274, 1994.
6. S. Kak, "On generalization by neural networks", *Information Sciences*, vol. 111, pp. 293-302, 1998.
7. S. Kak, The three languages of the brain: quantum, reorganizational, and associative. In Learning as Self-Organization, K. Pribram and J. King (editors). Lawrence Erlbaum Associates, Mahwah, NJ, 1996, pp. 185-219.
8. K.-W. Tang and S. Kak, "A new corner classification approach to neural network training", *Circuits Systems Signal Processing*, Vol.17, No.4, 1998, Pp. 459-469.
9. S. Kak, "A class of instantaneously trained neural networks", *Information Sciences*, vol. 148, pp. 97-102, 2002.
10. S. Kak, "Faster web search and prediction using instantaneously trained neural networks," *IEEE Intelligent Systems*, vol. 14, pp. 79-82, November/December 1999.
11. A. Ponnath, Instantaneously Trained Neural Networks. Computer Research Repository, arXiv:abs/cs/0601129, 2006.

12. Z. Zhang et al., TextCC: New feed forward neural network for classifying documents instantly. Lecture Notes in Computer Science, Volume 3497, Jan 2005, Pages 232 – 237.
13. S. Kak, Y. Chen, L. Wang, Data mining using surface and deep agents based on neural networks. 16th Americas Conference on Information Systems. Lima, Peru, August 2010.
14. J. Zhu and G. Milne, Implementing Kak neural networks on a reconfigurable computing platform. In FPL 2000, LNCS 1896, R.W. Hartenstein and H. Gruenbacher (eds.), SpringerVerlag, 2000, p. 260-269.
15. A. Shortt, J. G. Keating, L. Moulinier, C. N. Pannell , Optical implementation of the Kak neural network. Information Sciences vol. 171, pp. 273-287, 2005.
16. Z. Jihan, P. Sutton, An FPGA implementation of Kak's instantaneously-trained, fastclassification neural networks. Proceedings of the 2003 IEEE International Conference on Field-Programmable Technology (FPT), December 2003.
17. S. Kak, Generalized unary coding. Circuits, Systems and Signal Processing, 2015.
18. E. M. Azoff, Neural network time series forecasting of financial markets, John Wiley and Sons, New York 1994.
19. http://www.sidc.be/silso/ data for sunspot prediction.
20. M. Li, K. Mehrotra, C.K. Mohan, and S. Rank, "Forecasting Sunspot Numbers Using Neural Networks". Electrical Engineering and Computer Science Technical Reports. Paper 67, 1990.
21. S. Kak and A. Chatterjee, On decimal sequences. IEEE Transactions on Information Theory, IT-27: 647 – 652, 1981.

VITA

PUSHPA SREE POTLURI

Candidate for the Degree of

Master of Science

Thesis: TRAINING OF CC4 NEURAL NETWORK USING UNARY AND SPREAD UNARY INPUTS

Major Field: Electrical Engineering

Biographical:

Education:
Completed the requirements for the Master of Science in Electrical Engineering at Oklahoma State University, Stillwater, Oklahoma in December, 2015.

Received Bachelor of Technology degree in Electronics and Communication Engineering from Jawaharlal Nehru Technological University, Kakinada, India in May, 2014.

Experience:
Research Assistant to Dr. Subhash Kak, OSU Stillwater, OK.
Nov 2014 – May 2015