VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI - 590018

Seminar Report

on

”MULTI-LAYER PERCEPTRON NEURAL NETWORKS DECODER FOR LDPC CODES ”

Submitted in partial fulfilment of the requirements for the VIII Semester

Bachelor of Engineering

in

ELECTRONICS AND COMMUNICATION ENGINEERING

For the Academic Year

2020-2021

BY

PUSHKAR PRAMOD WANI

1PE17EC093

UNDER THE GUIDANCE OF

Prof. Bivas Bhattacharya

Assistant Professor, Dept. of ECE, PESIT-BSC

Department of Electronics and Communication Engineering

PESIT - BANGALORE SOUTH CAMPUS

Hosur Road, Bengaluru - 560100

PESIT - BANGALORE SOUTH CAMPUS
HOSUR ROAD, BENGALURU - 560100
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the seminar entitled **"Multi Layer Perceptron Neural Networks Decoder for LDPC Codes "** is a bonafide work carried out by **PUSHKAR PRAMOD WANI** bearing register number **1PE17EC093** in partial fulfilment for the award of Degree of Bachelors (Bachelors of Engineering) in **Electronics and communication Engineering of Visvesvaraya Technological University**, Belagavi during the year **2020-2021**.

Signature of the Seminar Guide	Signature of the HOD
Prof.Bivas Bhattacharya	**Dr. Subhash Kulkarni**
Assistant Professor	**HOD, ECE**

Name of the Examiners	**Signature with Date**
1.	
2.	

Acknowledgements

It is always a pleasure to remind the fine people for their sincere guidance I received in completion of this internship.I would like to express my gratitude to all those who gave me the possibility to complete this internship.

I am very grateful to my Guide **Prof. Bivas Bhattacharya , Assistant Professor** , for providing me with suggestions and helping choosing the topic. His suggestions, instructions and support during this unprecedented times via online have served as the major contributor towards the completion of the work.

Also I would like to express Gratitude to our Honorable Prinicipal **Dr. Subhash Kulkarni, Principal** , PESIT BSC, for his continuous encouragement , management of faculty for assistance and availability for any Queries.

I would like to thank **Dr. Subhash Kulkarni, Professor and Head**, Department of Electronics and Communication Engineering, PESIT BSC, for providing me a moral support and an ample amount of time to prepare for the seminar amidst this compact and constricted timeframe.

Finally I apologize all other unnamed who helped me in various ways to have a good training.

PUSHKAR PRAMOD WANI

ABSTRACT

A very important and near optimum group of the block codes that have been developed in direction of Shannon's theory concept, is low density parity check (LDPC) code. There have been presented different algorithms for decoding of this class of code such as maximum likelihood (ML), bit flipping (BF), a posteriori probability (APP) and sum product (SP) algorithms that the first two of them decode by hard criterion and two others decode by soft criterion. In this article, considering LDPC codes structure and taking advantage of LDPC codes demonstrated by Tanner graph, we have presented a quite new method based on multi layer perceptron (MLP) neural networks that decodes LDPC codes by soft decision manner. This method opens another new way for LDPC and even other block codes decoding. Simulation results display good performance of this method compared to other known algorithms.

Keywords: *LDPC codes, Tanner graph, iterative decoding, MLP neural networks*

Contents

List of Figures

Chapter 1

Introduction

Low density parity check (LDPC) codes were introduced by Gallager in 1962. These codes are a type of block codes, being so considerable because of their near to ideal performance known as Shannon limit. In addition to the construction of LDPC codes, the paper presented an iterative algorithm for decoding LDPC codes. However, the complexity of the algorithm was higher than the power of existent electronic processors.

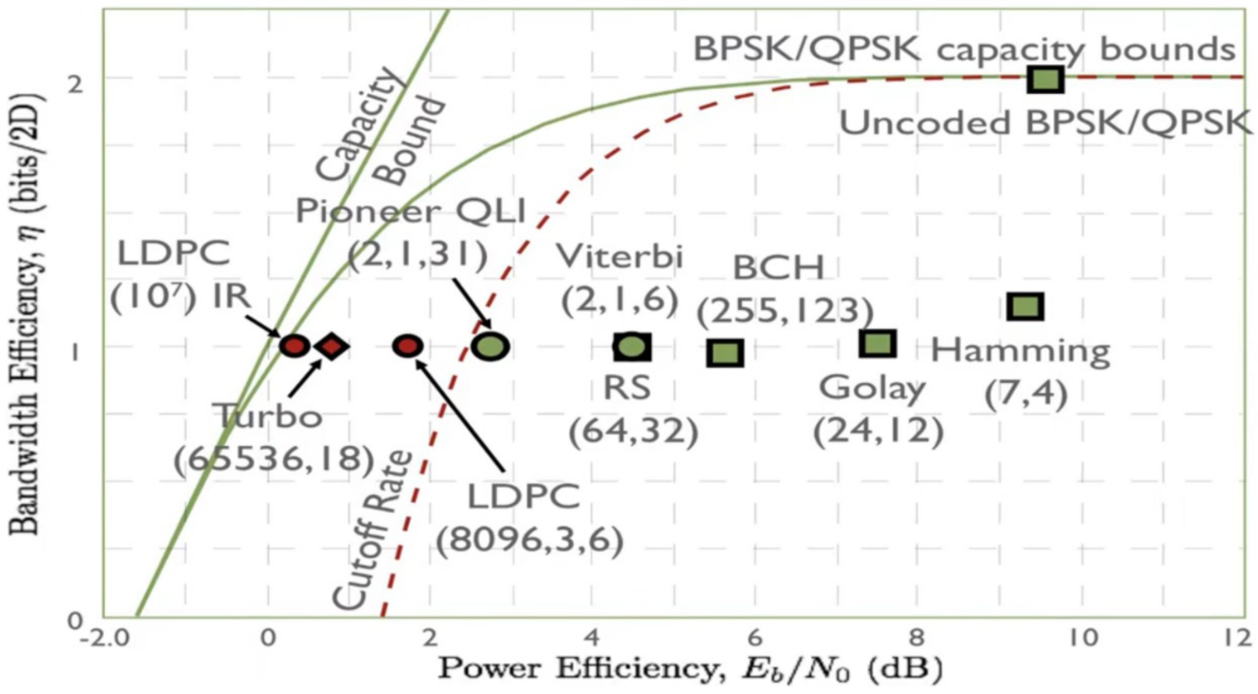


Figure 1.1: Shannon limit

LDPC codes can be illustrated in two forms: the matrix form and the graphical form. The graphical display creates a very useful realization for decoding of this code. This display was introduced first by Tanner in 1982 and is called Tanner graph. Despite Tanner's effort to revive LDPC codes, these codes have been forgotten until 1996, when MacKay and Neal rediscovered LDPC codes as a competitor to turbo codes. A LDPC code is represented with its sparse parity check matrix and the corresponding Tanner graph. LDPC codes can be considered serious competitors to turbo codes in terms of performance and complexity. However, much of the work on LDPC decoder design has been directed towards achieving optimal tradeoffs between complexity and coding gain.

This paper is organized as follows. In the second section, we will present MLP neural networks. Our proposed structure and method of decoding will be explained in the third section. Finally, in the fourth section, we will present computer simulation results for a typical short length LDPC code and compare the performance with SP algorithm.

Chapter 2

Background Knowledge

2.1 Basics of Communication Channel

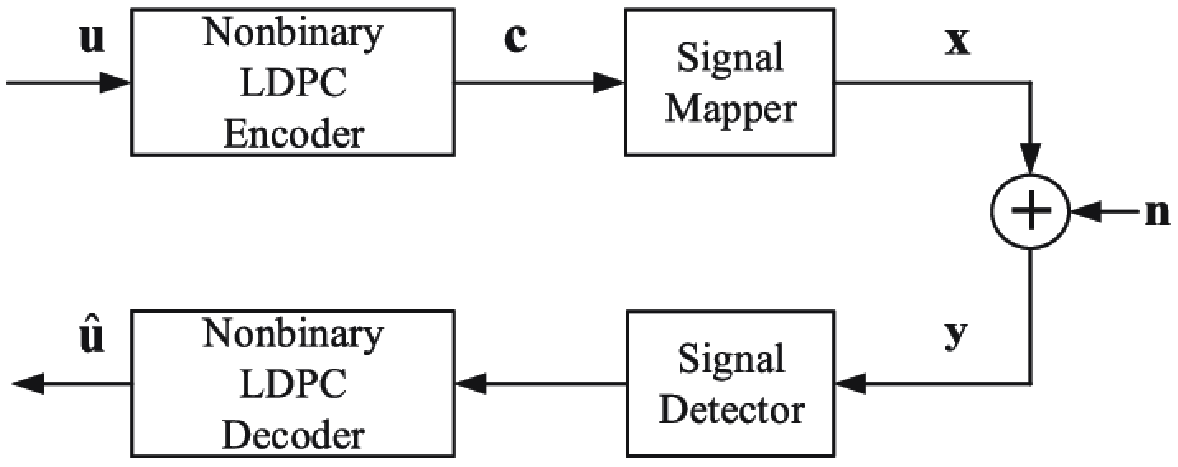


Figure 2.1: Communication channel

2.1.1 Generator matrix

In coding theory, a generator matrix is a matrix whose rows form a basis for a linear code. The codewords are all of the linear combinations of the rows of this matrix, that is, the linear code is the row space of its generator matrix.

$$G = [I_K \ P]$$

2.1.2 Parity Check Matrix

In coding theory, a parity-check matrix of a linear block code C is a matrix which describes the linear relations that the components of a codeword must satisfy. It can be used to decide whether a particular vector is a codeword and is also used in decoding algorithms.

Formally, a parity check matrix, H of a linear code C is a generator matrix of the dual code. This means that a codeword c is in C if and only if the matrix-vector product $Hc^T = 0$.

$$H = [P^T \quad I_{n-k}]$$

2.1.3 Syndrome

Syndrome in coding theory, a symbol vector (ordered set of symbols) generated at an intermediate stage of the decoding algorithm for an error-correcting code. The syndrome depends only on the error pattern and not on the transmitted codeword.

$$S = rH^T$$

2.2 LDPC Codes

LDPC Codes stands for low density parity check codes. They are the codes specified by a parity check matrix H, containing mostly 0's and only a small number of 1's. These codes are two types: binary and non-binary. Also, they are either regular or irregular. In regular case there is equal number of ones in either rows or columns of this matrix.

LDPC codes can be illustrated in two forms: the matrix form and the graphical form. The graphical display creates a very useful realization for decoding of this code. This display was introduced first by Tanner and is called Tanner graph. The Tanner graph is a bipartite graph with

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Figure 2.2: Parity check matrix

two types of nodes called variable nodes and check nodes. The variable nodes can be connected only to check nodes and vice versa. The connections between variable and check nodes is performed in light of matrix H , in such a way that if the columns of matrix H be considered as variable nodes and the rows of that as check nodes, in any location there is a one in the matrix, the related variable and check nodes will be connected.

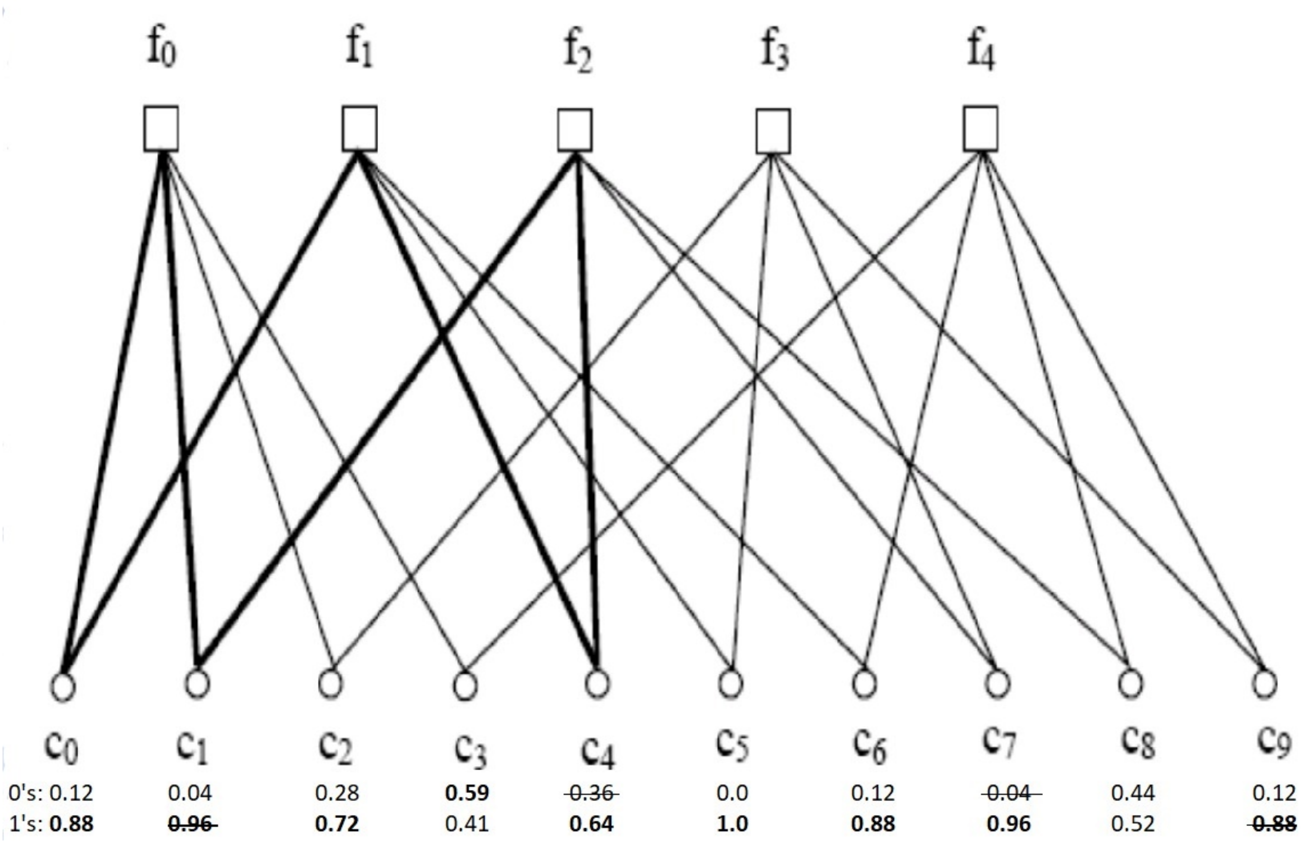

f_0
f_1
f_2
f_3
f_4
c_0
c_1
c_2
c_3
c_4
c_5
c_6
c_7
c_8
c_9
0's: 0.12 0.04 0.28 **0.59** ~~0.36~~ 0.0 0.12 ~~0.04~~ 0.44 0.12
1's: **0.88** ~~0.96~~ **0.72** 0.41 **0.64** **1.0** **0.88** **0.96** 0.52 ~~0.88~~

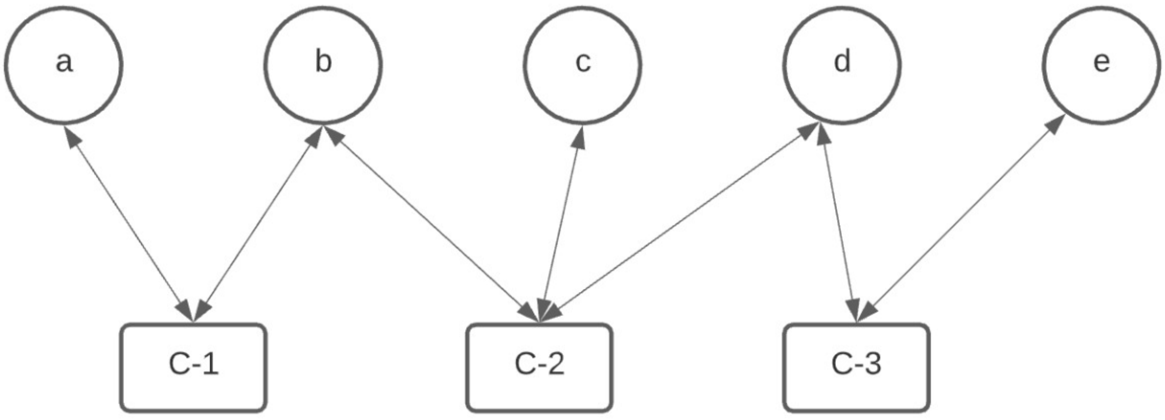
Figure 2.3: Tanner graph

In LDPC Codes number of columns corresponds to Codeword length which intern corresponds to n-bits and each row of the j rows is used to produce parity check sum. In tanner graph cycling back and forth in the loop until $Hc^T = 0$.

2.3 Decoding LDPC Codes

There are various decoding techniques involving Hard decision and Soft decision decoding.
•*Bit flipping*
•*A posteriori decoding*
•*Iterative decoding based on belief propagation or Sum Product Algorithm*

2.3.1 Computational Complexity of Sum Product Algorithm



$$f(a,b,c,d,e) = f_1(a,b) \quad f_2(b,c,d) \quad f_3(d,e) \qquad ---(1)$$

We know that..,

$$\sum_y f(x,y) = f(x) \qquad ---(2)$$

Similarly..,

$$f(b) = \sum_{a,c,d,e} f(a,b,c,d,e) \qquad ---(3)$$

$$f(b) = \sum_{a} f_1(a,b) \ \sum_{c,d} f_2(b,c,d) \ \sum_{e} f_3(d,e) \qquad ---(4)$$

$$\mu_{f_3}(d) = \sum_{e} f_3(d,e) \qquad ---(5)$$

$$\mu_{f_2}(b) = \sum_{c,d} f_2(b,c,d) \ \mu_{f_3}(d) \qquad ---(6)$$

$$\mu_{f_1}(b) = \sum_{a} f_1(a,b) \qquad ---(7)$$

$$f(b) = \mu_{f_1}(b) \ \ \mu_{f_2}(b) \qquad ---(8)$$

Chapter 3

Proposed Neural Network Decoder

3.1 MLP Neural Network

MLP neural network is composed of several layers. The first layer is called input layer and the last one is called output layer. The layers between input and output layers are hidden layers. In this network the outputs of every layer are the inputs of the next layer. In every layer, the outputs are functions of sum of that layer weighted inputs and another parameter called bias.

$$O^l = f^l\left(O^{l-1}W^l + B^l\right)$$

In the above equation, O^l is the l_{th} layer output matrix. Also, W^l and B^l are l_{th} layer weight matrix and bias matrix, respectively. $f^l()$ is a function related to l_{th} layer. It is necessary that this function is so selected that creates the expected amounts of the outputs. On the other hand, being differentiable it is a very important factor for $f^l()$ considering neural network training. In a neural network, weights and biases are adjustable parameters. They can be adjusted based on a set of given data. The process of finding and adjusting the weights and biases of a neural network is called training. The purpose of neural network training is to reduce the Sum Square Error (SSE) function E.

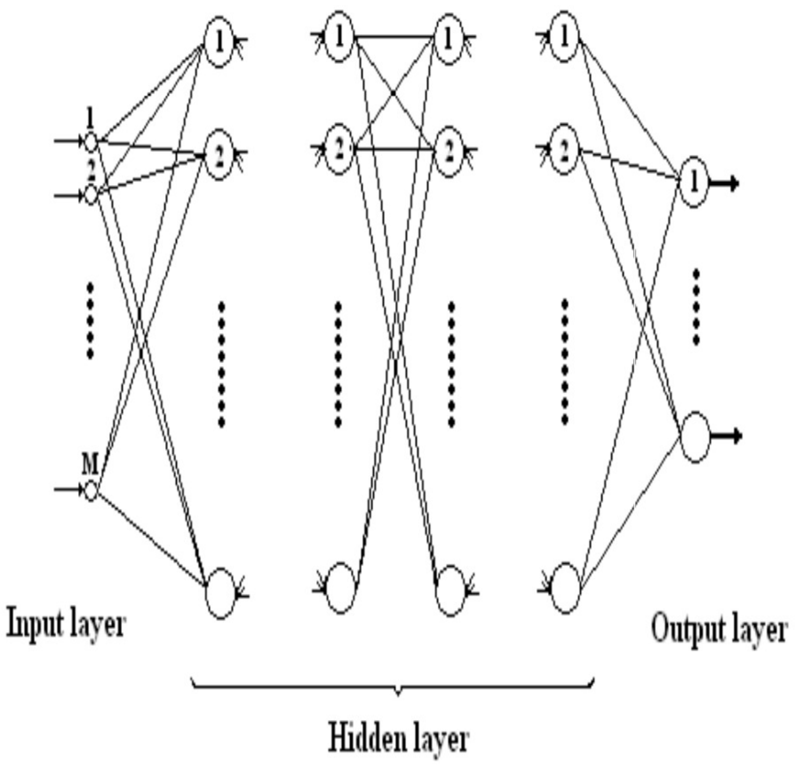


Figure 3.1: MLP Neural Network

$$E = \frac{1}{2}\sum_{i} e_i^2$$

where e_j is the error of j_{th} entry of the output matrix and equals to the difference between the Target value and the Predicted value of the output. Required modifications in the training parameters may be considered using optimization algorithms such as gradient descent algorithm.

$$\Delta W^j = -\eta_{W^j} \frac{\partial E}{\partial W^j}$$

$$\Delta B^j = -\eta_{B^j} \frac{\partial E}{\partial B^j}$$

where both $\eta_{W(j)}$ and $\eta_{B(j)}$ are constant values between 0 and 1, and are called training rate of W^j and training rate of B^j , respectively.

In each iteration the calculated differences are added on the previous amounts of these variables.

$$W^j(n) = W^j(n-1) + \Delta W^j$$

$$B^j(n) = B^j(n-1) + \Delta B^j$$

3.2 Neural decoder structure

The neural decoder is a MLP neural network with two layers. An input layer and an output layer. This network is constructed based on Tanner graph or the parity check matrix of the code. Our decoder carries out the decoding of a binary LDPC code in a quite different process with other existing algorithms. The network input is actually the received vector from channel r . This vector is the code word polluted to the additive noise during the channel and may not be a valid code word.

$$r = c + n$$

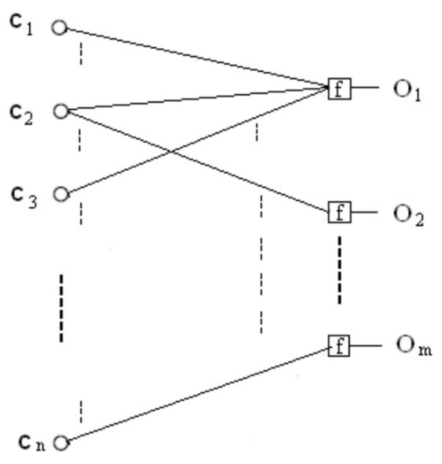


Figure 3.2: The structure of the proposed neural decoder

In this model, layer 1 corresponds to the variable nodes and layer 2 corresponds to the check nodes of Tanner graph. Therefore, if the input of layer 1 is a codeword, then the desired values in the output of layer 2 are zeros. The function of the output layer of the network must be the XOR function in analog conditions.

$$x \oplus y = x(1-y) + y(1-x)$$

However, in practice, the number of code word components, corresponding to the number of received vector components is much more than two. Therefore, XOR function of the related variables can be expressed in the form as shown below

$$\begin{aligned}
&(((s \oplus t) \oplus u) \oplus \ldots) \oplus v \\
&= ((x \oplus u) \oplus \ldots) \oplus v \\
&= (y \oplus \ldots) \oplus v \\
&s \oplus t = x \\
&x \oplus u = y
\end{aligned}$$

The explained XOR function is differentiable, and so it can be used in the training process of the neural network using gradient descent algorithm. The components of the output vector $O = [O_1, O_2, O_3, ..., O_m]$ correspond to the check nodes of Tanner graph. Accordingly, the network parameters are trained so that O tends to zero. The SSE function E of the network is as shown below

$$\begin{aligned}
E = \frac{1}{2}\sum_{j=1}^{m}(e_j)^2 &= \frac{1}{2}\sum_{j=1}^{m}(0 - o_j)^2 \\
&= \frac{1}{2}\sum_{j=1}^{m}(o_j)^2
\end{aligned}$$

The training parameters of our proposed decoder are the received vector components or the inputs of the network. The components change as far as the SSE function E approaches its minimum point. Finally, after training the input vector, the components of the altered vector are mapped to 0 or 1 depending on the minimum Euclidean distance to 0 and 1. The created word is an estimation of the transmitted code word.

Chapter 4

SIMULATION RESULTS

For simulation example, we consider the code introduced in the introductory section by its Tanner graph and parity check matrix. The neural network decoder for this code is illustrated in the figure below.

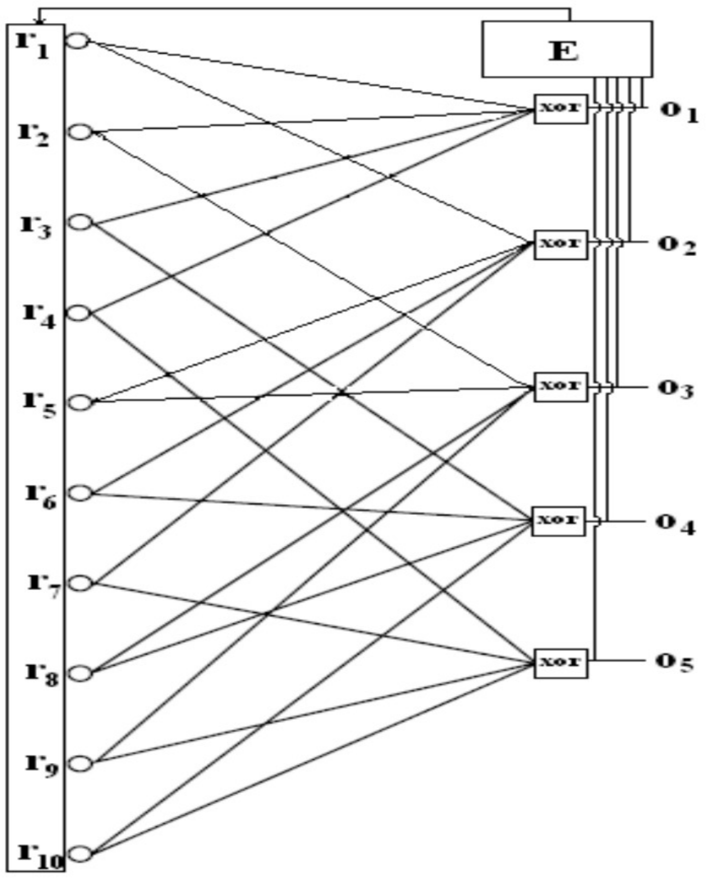


Figure 4.1: MLP Neural Network Decoder

In each iteration, the decoder performs XOR functions at each check node. Then, after sum square error calculation, it inserts the necessary changes to the input components. This operation is carried out until function E decreases sufficiently. For instance when SNR =11,$\eta = 0.008$, in E verses Number of iteration's graph it is observable how amount of E decreases as a result of the iterations increase for decoding process. One of the most important criterions is number of iterations. In BER verses Number of iteration's graph you can see BER results considering different number of iterations for $\eta = 0.008$ and SNR =11dB. However, it is better that number of iterations depends on network error or E . We can insert a threshold for E . We consider $E \leq 0.2$ and the iterations number equal to 200. Consequently, if the amount of E be less than 0.2, the decoding process will be stopped. Finally, the simulation results are given and compared with SPA in Figure 4.4 for different amounts of $\eta = 0.008, 0.02, 0.04$ and SNR = $(11-14)$dB , where for each SNR , we have inserted 2000000 code words polluted to additive white gausian noise (AWGN) and calculated bit error rates.

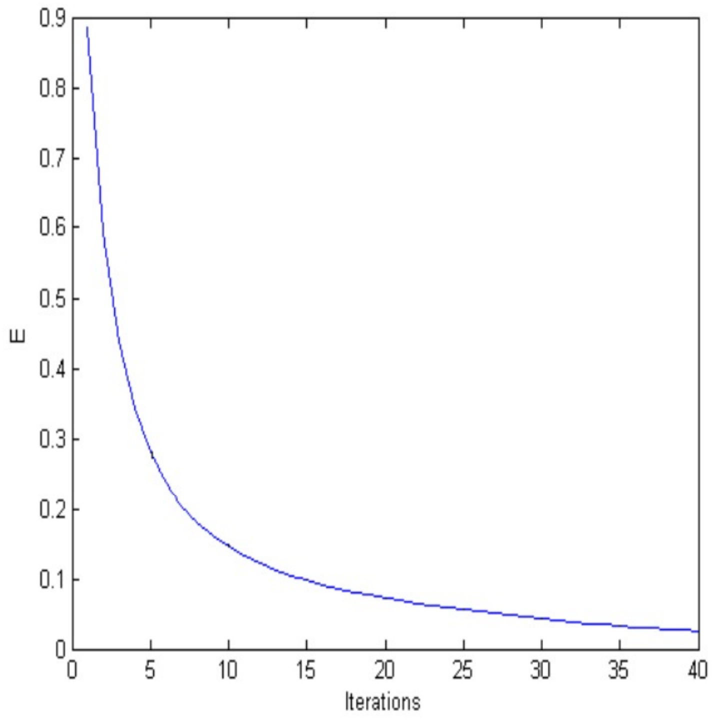


Figure 4.2: E verses Number of iterations

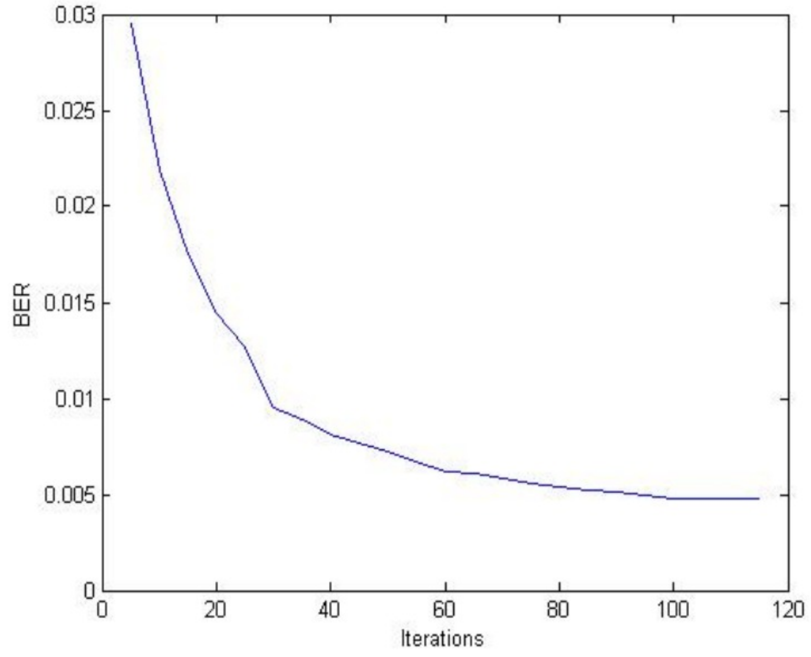
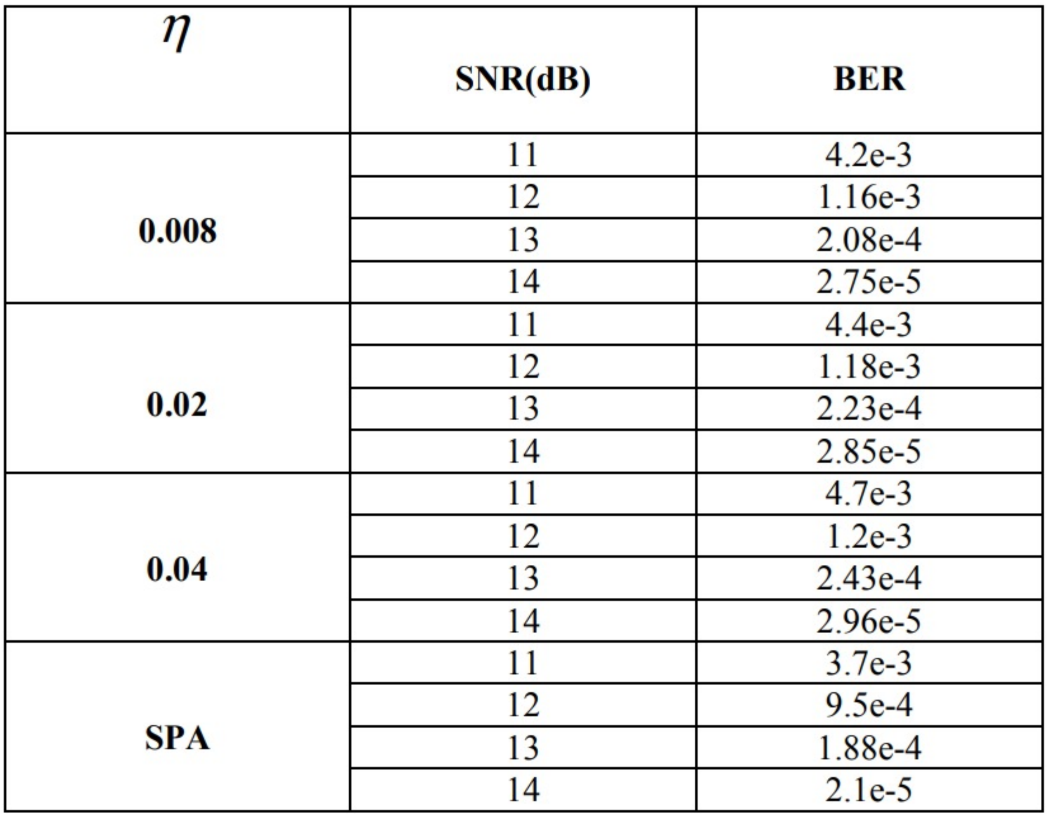


Figure 4.3: BER considering different number of iterations



η	SNR(dB)	BER
0.008	11	4.2e-3
	12	1.16e-3
	13	2.08e-4
	14	2.75e-5
0.02	11	4.4e-3
	12	1.18e-3
	13	2.23e-4
	14	2.85e-5
0.04	11	4.7e-3
	12	1.2e-3
	13	2.43e-4
	14	2.96e-5
SPA	11	3.7e-3
	12	9.5e-4
	13	1.88e-4
	14	2.1e-5

Figure 4.4: Results for different η and comparison with SPA

Chapter 5

Conclusion

In this paper, a new method of decoding for LDPC codes based on MLP neural networks with decoding process in a soft decision manner is developed. Whereas previous known decoders use probabilistic metrics, MLP based decoder doesn't need to calculate any probability. It performs a good result with decreased decoding complexity.

Chapter 6

REFERENCES

a. Karami, A. R., Ahmadian Attari, M., Tavakoli, H. (2009). Multi Layer Perceptron Neural Networks Decoder for LDPC Codes. 2009 5th International Conference on Wireless Communications, Networking and Mobile Computing.

b. https://ieeexplore.ieee.org/document/5303382

Multi Layer Perceptron Neural Networks Decoder for LDPC Codes

A. R. Karami[1] , M. Ahmadian Attari[2] , H. Tavakoli[3]
Faculty of electrical engineering of K.N.Toosi university
Tehran, Iran
Ali.r.karami@ee.kntu.ac.ir [1], m_ahmadian@kntu.ac.ir [2], tavakoli@ee.kntu.ac.ir [3]

***Abstract*— A very important and near optimum group of the block codes that have been developed in direction of Shannon's theory concept, is low density parity check (LDPC) code. There have been presented different algorithms for decoding of this class of code such as maximum likelihood (ML), bit flipping (BF), a posteriori probability (APP) and sum product (SP) algorithms that the first two of them decode by hard criterion and two others decode by soft criterion. In this article, considering LDPC codes structure and taking advantage of LDPC codes demonstrated by Tanner graph, we have presented a quite new method based on multi layer perceptron (MLP) neural networks that decodes LDPC codes by soft decision manner. This method opens another new way for LDPC and even other block codes decoding. Simulation results display good performance of this method compared to other known algorithms.**

Keywords- ***LDPC codes, Tanner graph, iterative decoding, MLP neural networks***

I. INTRODUCTION

Since Shannon's theorem on theoretical bounds on performance of error correction coding in his 1948 paper [1], many plans have been presented for error correcting codes and their decoding algorithms. However, none of them have reached the ideal performance until 1993 when turbo codes were discovered by Berrou, Glavieux and Thitimajshima [2] and then in 1996 when Mackey and Neal reintroduced LDPC codes in [3,4] the code were first offered by Gallager [5] in 1962. The adjective low density means that there are a few ones in the check parity matrix of the code. These codes are two types: binary and non binary. Also, they are either regular or irregular. In regular case there is equal number of ones in either rows or columns of this matrix.

LDPC codes can be illustrated in two forms: the matrix form and the graphical form. The graphical display creates a very useful realization for decoding of this code. This display was introduced first by Tanner in [6] and is called Tanner graph. The Tanner graph is a two partite graph with two types of nodes called variable nodes and check nodes. The variable nodes can be connected only to check nodes and vice versa. The connections between variable and check nodes is performed in light of matrix H, in such a way that if the columns of matrix H be considered as variable nodes and the rows of that as check nodes, in any location there is a one in the matrix, the related variable and check nodes will be connected by an edge. Figure 1 illustrates the Tanner graph for a typical matrix H:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

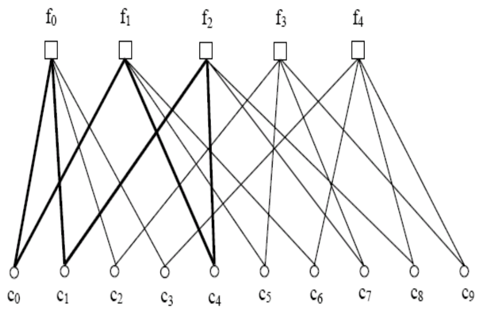


Figure 1. Tanner graph for (10, 5, 3) LDPC code.

An important parameter that must be regarded in the Tanner graph, is its girth. The girth is the length of the shortest closed loop that begins from a node and terminates at the same node. Such a loop of length 6 is illustrated in Fig.1 by thick lines. Normally, the longer girth in the Tanner graph leads to a more powerful code. Therefore, avoiding the girth of length 4 is essential. This means that every two columns or rows of matrix H must not be common at more than one non-zero element.

This paper is organized as follows. In the second section, we will present MLP neural networks. Our proposed structure and method of decoding will be explained in the third section. Finally, in the fourth section, we will present computer simulation results for a typical short length LDPC code and compare the performance with SP algorithm.

II. MLP NEURAL NETWORKS

As shown in Figure 2, MLP neural network is composed of several layers. The first layer is called input layer and the last one is called output layer. The layers between input and output layers are hidden layers. In this network the outputs of every layer are the inputs of the next layer. In every layer, the outputs are functions of sum of that layer weighted inputs and another parameter called bias.

$$O^l = f^l\left(O^{l-1}W^l + B^l\right) \quad (1)$$

In the above equation, O^l is the lth layer output matrix. Also, W^l and B^l are lth layer weight matrix and bias matrix, respectively. $f^l(\ )$ is a function related to lth layer. It is necessary that this function is so selected that creates the expected amounts of the outputs. On the other hand, being differentiable it is a very important factor for $f^l(\ )$ considering neural network training.

This network may be trained in two forms: instructive manner and non instructive manner. In the instructive manner the network is trained for the inserted inputs. But in the second form the structure of an input block and its mathematical relations can be modeled by a constant neural network. The training is performed in successive iterations. Each iteration is performed in a time interval that the network parameters are trained once.

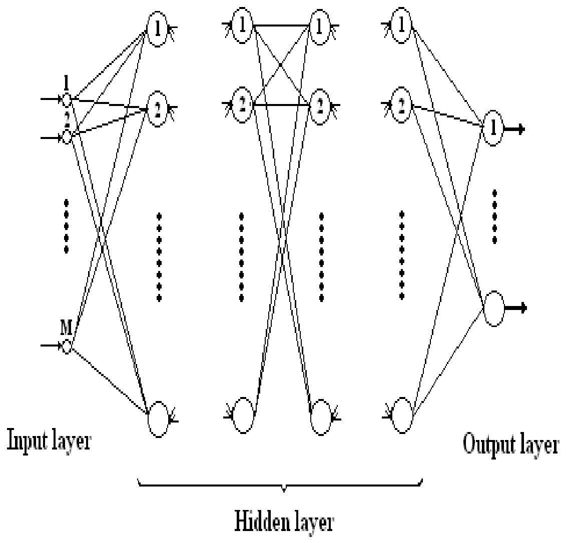


Figure 2. MLP neural network structure.

Instructing parameters are usually each layer weights and biases. Also the training rates can be adjusted. These parameters are trained such that to minimize the sum square error function E, the difference between the network outputs and the desired outputs.

$$E = \frac{1}{2}\sum_i e_i^2 \quad (2)$$

where, e_i is each output cell error.

One of the very conventional ways for minimizing E is gradient descent algorithm.

In each iteration the jth layer weights W^j, and biases B^j variations are calculated from the following equations:

$$\Delta W^j = -\eta_{W^j}\frac{\partial E}{\partial W^j} \quad (3)$$

$$\Delta B^j = -\eta_{B^j}\frac{\partial E}{\partial B^j} \quad (4)$$

where η_{W^j} is W^j learning rate and η_{B^j} is B^j learning rate.

In each iteration the calculated differences are added on the previous amounts of these variables.

$$W^j(n) = W^j(n-1) + \Delta W^j \quad (5)$$

$$B^j(n) = B^j(n-1) + \Delta B^j \quad (6)$$

III. Neural Decoder Structure

The neural decoder is a MLP neural network with two layers. An input layer and an output layer. This network is constructed based on Tanner graph or the parity check matrix of the code. Figure 3 illustrates the presented neural decoder diagram which looks like a Tanner graph.

Our decoder carries out the decoding of a binary LDPC code in a quite different process with other existing algorithms. The network input is actually the received vector from channel r. This vector is the code word polluted to the additive noise during the channel and may not be a valid code word.

$$r = c + n \quad (7)$$

The most important problem in neural decoder is employing a proper error function. If the code is a binary code, the relation between the code components at each check node in the Tanner graph actually will be an XOR function. Employed function in the network output layer must be such that to be able to simulate XOR function in binary condition. One of the functions that can carry out this simulation in binary and continuous conditions is of the form:

$$x \oplus y = x.(1-y) + y.(1-x) \quad (8)$$

However, under actual and practical conditions that the number of these code components is much more at each check node, performs XOR function may be performed by equation (9).

$$\begin{aligned} &(((s\oplus t)\oplus u)\oplus \ldots)\oplus v \\ &=((x\oplus u)\oplus \ldots)\oplus v \\ &=(y\oplus \ldots)\oplus v \\ &s\oplus t=x \\ &x\oplus u=y \end{aligned} \qquad (9)$$

This description is differentiable, so it can be used for the neural network training by gradient descent algorithm.

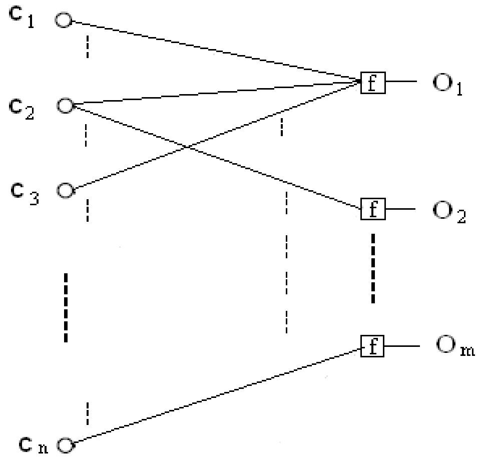


Figure 3. Neural decoder with two layers

The network outputs Os in Figure 3 actually are the same as check nodes outputs of Tanner graph. Consequently, the network must be trained such that the amounts of these outputs intend to 0, because the desired values of the outputs are 0. The sum square error function of our neural decoder will be

$$E=\frac{1}{2}\sum_{i=1}^{m}e_i^2=\frac{1}{2}\sum_{i=1}^{m}O_i^2 \qquad (10)$$

where $e_i=(0-O_i)$ and m is number of check nodes.

In our neural network, the training parameters are the received vector components or the network inputs. The received vector components change so that to minimize the error function E. Finally, the changed vector components are mapped to one or zero depending on their distance to one or zero. Then, the decoded word will be a good estimation of the transmitted code word.

Equation (11) shows that how the received vector components variations are calculated.

$$\begin{aligned} \Delta C_j &= -\eta\frac{\partial E}{\partial C_j} \\ &= -\eta\sum_{i=1}^{m}\left(\frac{\partial E}{\partial e_i}\frac{\partial e_i}{\partial O_i}\frac{\partial O_i}{\partial C_j}\right) \\ &= -\eta\sum_{i=1}^{m}\left(e_i\frac{\partial O_i}{\partial C_j}\right) \end{aligned} \qquad (11)$$

where, $\frac{\partial O_i}{\partial C_j}$ is comfortably calculated considering the output function explanation.

IV. Simulation Results

For simulation example, we consider the code introduced in the introductory section by its Tanner graph and parity check matrix. The neural network decoder for this code is illustrated in Figure 4.

In each iteration, the decoder performs XOR functions at each check node. Then, after sum square error calculation, it inserts the necessary changes to the input components. This operation is carried out until function *E* decreases sufficiently. For instance when $SNR=11,\eta=0.008$, in Figure 5 it is observable how amount of *E* decreases as a result of the iterations increase for decoding process.

One of the most important criterions is number of iterations. In Figure 6 you can see BER results considering different number of iterations for $\eta=0.008$ and $SNR=11dB$. However, it is better that number of iterations depends on network error or E. We can insert a threshold for E. We consider $E\leq 0.2$ and the iterations number equal to 200. Consequently, if the amount of *E* be less than 0.2, the decoding process will be stopped. Finally, the simulation results are given and compared with SPA in TABLE I for different amounts of $\eta=0.008,0.02,0.04$ and $SNR=(11-14)dB$, where for each SNR, we have inserted 2000000 code words polluted to additive white gausian noise (AWGN) and calculated bit error rates.

Conclusions

We have introduced a new method of decoding for LDPC codes based on MLP neural networks with decoding process in a soft manner. Whereas previous known decoders use probabilistic metrics, our decoder doesn't need to calculate any probability. It performs a good result with decreased decoding complexity. It is a simple prototype neural decoder and we are working to improve its performance using other learning algorithms.

ACKNOWLEDGMENT

This research has been supported by Iran Telecommunication Research Centre (ITRC).

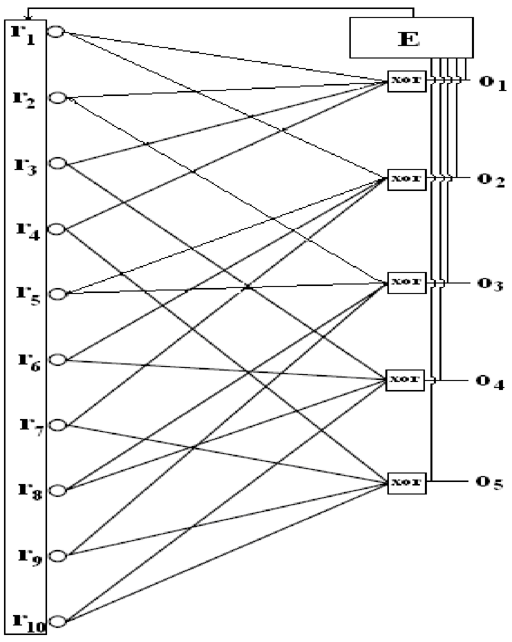


Figure 4. A MLP decoder for the code of Fig 1.

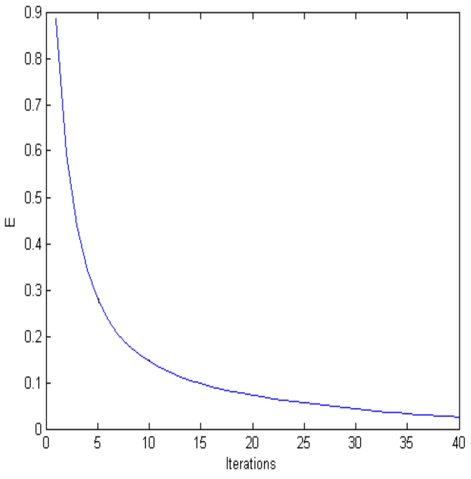


Figure 5. E diagram

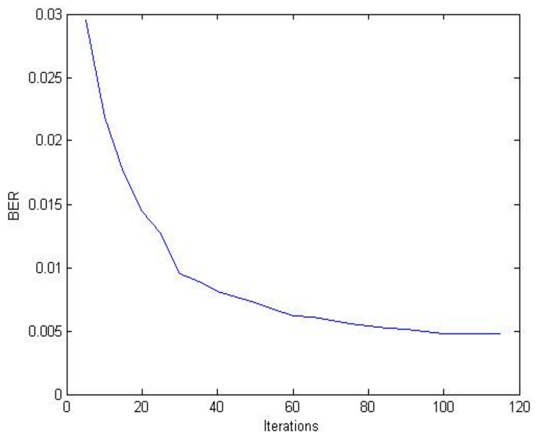


Figure 6. BER results considering different number of iterations

TABLE I. SIMULATION RESULTS

η	**SNR(dB)**	**BER**
0.008	11	4.2e-3
	12	1.16e-3
	13	2.08e-4
	14	2.75e-5
0.02	11	4.4e-3
	12	1.18e-3
	13	2.23e-4
	14	2.85e-5
0.04	11	4.7e-3
	12	1.2e-3
	13	2.43e-4
	14	2.96e-5
SPA	11	3.7e-3
	12	9.5e-4
	13	1.88e-4
	14	2.1e-5

REFERENCES

[1] C. E. Shannon, “A mathematical theory of communication,” *Bell Syst. Tech. J.*, vol. 27, pp. 379--423, 623--656, July and October 1948.

[2] C. Berrou, A. Glavieux and P. Thitimajshima, “Near Shannon limit error-correcting coding and decoding: turbo codes,” *Proc. IEEE International Conference on Communications,* Geneva, Switzerland, vol. 2, pp. 1064--1070, May 1993.

[3] D.J.C. MacKay and R. M. Neal, “Near Shannon limit performance of low density parity check nodes,” *Electron. Lett.,* vol. 33, no. 6, March 13, 1997.

[4] D.J.C. MacKay and R. M. Neal, “Good error-correcting codes based on very sparse matrices,” available at http://www.interference.phy.cam.ac.uk/mackay/CodesGallager.html

[5] R. G. Gallager, “Low-density parity-check codes,” *IRE Trans. Inf. Theory*, vol. **IT**-8, no. 1, pp. 21--28, January 1962.

[6] L. M. Tanner, “A recursive approach to low complexity codes,” *IEEE Trans. Inf. Theory,* vol. 27, no. 5, pp. 533--547, 1981.

[7] W. E. Ryan, “An introduction to LDPC codes,” in CRC Handbook for Coding and Signal Processing for Recording Systems (B. Vasic, ed.) CRC Press, 2004.

[8] S. Haykin *Neural Networks: A comprehensive Foundation.* Upper Saddle River, NJ: Prentice Hall, 1994.