Iterative Decoding of Low Density Parity Check Codes

Final year B.Tech Project Report

By

Saptarashmi Bandyopadhyay

Examination Roll:510514006

Eigth Semester 2017-2018

B.Tech. in Computer Science and Engineering

under the Esteemed Guidance of

Dr. Abhik Mukherjee

Department of Computer Science and Technology

Indian Institute of Engineering Science and Technology, Shibpur
Howrah-711103
West Bengal, India

Department of Computer Science and Technology

Indian Institute of Engineering Science and Technology, Shibpur
Howrah-711103
West Bengal, India

ACKNOWLEDGEMENT

I must take this opportunity to place on record my deep sense of respect and gratitude to **Dr. Abhik Mukherjee,** my final year B.Tech project guide for his valuable advice, resourceful guidance, active supervision and constant encouragement without which it would not have been possible to complete this project report.

I also want to express my heartfelt gratitude towards Prof. Sulata Mitra, Head, **Department of Computer Science and Technology, IIEST Shibpur**, and to all the teachers in the department for their guidance and active support.

Date ....

(SAPTARASHMI BANDYOPADHYAY)

Examination Roll:510514006
Eigth Semester 2017-2018
B.Tech. in Computer Science and Engineering

OUTLINE

- Low density parity check codes
- Tanner Graph Representation
- Bit Flipping Algorithm
  - -Multi-Bit Flipping Hard Decision decoding
- Message Passing Algorithm
- Message Passing with Bit Flipping
- A comparison with probabilistic decoding
- Parameters of LDPC Code
- Error Correction
- Encoding using BPSK Modulation and AWGN noise
- Experimentation
- Importance of Message Passing Algorithm
- Deep Learning in LDPC Decoding
- Future Work
- References

Low-density parity-check (LDPC) codes had a renaissance when they were rediscovered in the 1990's. Since then LDPC codes have been an important part of the field of error-correcting codes, and have been shown to be able to approach the Shannon capacity, the limit at which we can reliably transmit information over noisy channels.

Following this, many modern communications standards have adopted LDPC codes. Error-correction is equally important in protecting data from corruption on a hard drive as it is in deep-space communications. It is most commonly used for example for reliable wireless transmission of data to mobile devices. For practical purposes, both encoding and decoding need to be of low complexity to achieve high throughput and low power consumption.

Low-density parity check codes

A low-density parity-check code, or LDPC[1] code, is defined by a parity check matrix

$H \varepsilon \{0,1\}_{mxn}$. H is sparse and has O(n) nonzero elements.

The set of code words C is defined by

$C = \{ x \varepsilon \{0,1\}^n \mid H x^T = 0^T \}$; where 0 is the zero vector and we assume that all vectors are row

vectors.

The matrix H is referred to as the parity-check matrix, since it requires that a code word x have even parity in the so-called parity-check equations.

$s_a = \sum_{i=1}^{n} H_{ai} x_i = 0$; for all a = 1,2, …, m.

Each parity-check equation is often called a checksum or check, and the vector s of sums s_a is called the syndrome. If a check has even parity we say that the check is satisfied and otherwise we say that the check is unsatisfied.

The term ''low density'' signifies a sparse parity check matrix in terms of the number of 1s present in the matrix. Due to this sparseness, LDPC codes possess great advantages in terms of efficient decoder implementation and storage and can approach the Shannon Limit .

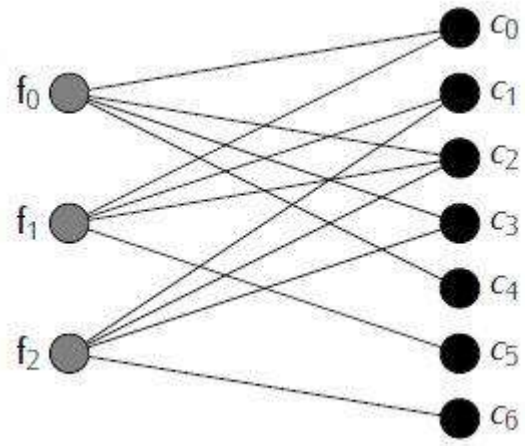
Tanner Graph Representation

The graph representation of a parity-check matrix is more commonly referred to as its Tanner graph. The properties of a tanner graph are as follows :

- The Tanner graph is a bipartite graph with two sets of nodes.
- The first set consists of the check nodes, each corresponding to the rows, or checks, of H.
- The second set consists of the variable nodes, each corresponding to a column of H, or a symbol of a word.
- There is an edge between a check node *a* and a variable node i if and only if $H_{ai} = 1$.

$[H]_{(3\times7)}$	c_0	c_1	c_2	c_3	c_4	c_5	c_6
f_0	1	0	1	1	1	0	0
f_1	1	1	1	0	0	1	0
f_2	0	1	1	1	0	0	1

$$f_0 = c_0 + c_2 + c_3 + c_4 = 0$$
$$f_1 = c_0 + c_1 + c_2 + c_5 = 0$$
$$f_2 = c_1 + c_2 + c_3 + c_6 = 0$$



Diagrammatic Representation of the Tanner Graph

Although much of the focus in research of LDPC codes has been on decoding of the codes, encoding is equally important in terms of time complexity. For general LDPC codes, encoding cannot as of today be done in linear time. The biggest contribution to date is that of Richardson and Urbanke (2001b). They show that LDPC codes with a carefully chosen degree distribution can be encoded in linear time. In addition, they show that these codes are good for message-passing decoders in the sense that they can approach the Shannon limit using message passing decoders. Simpler code constructions can also yield linear time encoding but may not be as good for error-correction.

Encoding consists of taking k information bits and adding parity bits such that the information bits and the parity bits together form a codeword. The parity bits are essentially chosen by first setting the information bits in fixed positions of the codeword. Once the information bits have been fixed, the parity bits can be determined by solving a set of linear equations. The naïve method does this by Gaussian elimination. This method is simple but has quadratic time complexity. In the other method presented by Richardson and Urbanke (2001b) where the rows and columns of a parity-check matrix are only permuted such that the system of linear equations we are solving is defined by a nearly triangular matrix. Doing this reduces complexity significantly.

While the state of encoding of LDPC codes is in some aspects a solved problem in practice, the lack of a linear-time encoding algorithm for all types of LDPC codes still leaves something to be desired. The QC-LDPC construction is a convenient construction that has proven itself to work well enough for inclusion in communications standards. In addition, it allows for a compact implicit description of the parity-check matrix. The irregular ensembles of LDPC codes which can be encoded in linear time have the additional benefit of working well with message-passing decoders. However, being able to use arbitrary LDPC codes that can still be encoded in linear time may allow the use of codes which have better error-correcting capabilities.

Decoding of LDPC Codes and Implementation

Low Density Parity Check Codes can be decoded with two types of decoding, primarily,

- Algebraic Decoding
- Iterative Decoding (Probabilistic Decoding)

The common iterative decoding techniques include Bit-Flipping algorithms and Message Passing algorithms.

Bit-Flipping Algorithm

This algorithm attempts to correct a single bit in every iteration.

If multiple bits are allowed to flip in one iteration, then the algorithm is called a Multi-Bit-Flipping Algorithm. In every iteration of this algorithm the code word bit(s) with the highest number of a metric are flipped.

Metric is characterized by the type of algorithm: Hard-Decision and Soft-Decision.

- Hard decision metric = the number of unsatisfied parity check equations.
- Soft decision metric = both the number of unsatisfied and satisfied parity check sums, since reliability information is present.

An algebraic decoding algorithm in this context can only correct up to its random error correcting capability. But due to the sparse nature of H, the BF algorithm may correct error patterns whose number of errors exceeds the error correcting capability of the code.

Multi Bit-Flipping Hard Decision Decoding

1. initialize iteration=0, Failure[i]=0 for the i-th bit, i=0,1,..,n-1 and preset a maximum number of iterations i_{max}.

2. Compute the syndrome of the received vector r. If the syndrome is 0, i.e., all the parity check equations are satisfied, the decoding is stopped and r is declared as the code word.

3. set iteration=iteration+1. Compute the number of failed parity check equations for each bit and store them in Failure[i] for the i-th bit, i=0,1,..,n-1

4. Construct a set S = {j: Failure[j]>= Failure[j'] for all j=0,1,..n-1}, i.e., compute the set S of bits for which the Failure[j] is the largest

5. Flip the bit(s) in S

6. Repeat steps 2 to 5 until the syndrome is 0 or iteration <= i_{max}

Message Passing (MP) Algorithm

Messages are passed between message nodes and check nodes along the edges of the Tanner graph. [3]

The information that is sent from a node v_i to v_j in an MP algorithm should not get influenced by the node v_j i.e., in other words, the data that is being sent to a node must depend only on the other nodes. This type of message is called an "extrinsic message" or "extrinsic information".

The generic message passing structure is as follows :

1.Initialize the decoder by sending the received values of the message nodes to check nodes.

2.In each check node, an extrinsic message for each neighbor variable node is calculated and sent.

3.Each message node calculates a new value for itself depending on the previous received values and the information from the check nodes. The output is then checked by all the parity check equations and if all of them agree, decoding stops. Otherwise, an extrinsic message for each neighbor check node is calculated and sent.

4.Repeat steps 2 and 3 until a maximum number of iterations is reached.

Message Passing with Bit-Flipping:

1.Initialize the decoder by sending the received values of the message nodes to check nodes.

2. The message nodes do bit-flipping based on the number of unsatisfied parity check equations. (Hard Decision Bit Flipping)

3.In each check node, an extrinsic message for each neighbor variable node is calculated and sent to the message nodes.

4.Each message node calculates a new value for itself depending on the previous received values and **bit-flipped value** and the information from the check nodes. The output is then checked by all the parity check equations and if all of them agree, decoding stops. Otherwise, an extrinsic message for each neighbor check node is calculated and sent.

5.Repeat steps 2, 3 and 4 until a maximum number of iterations is reached.

EXPERIMENTATION

An Open source LDPC framework [3] in MATLAB. It has been adapted and coded according to experiments. The encoding framework involving BPSK modulation and addition of AWGN (Additive White Gaussian Noise) from the framework.

Parameters of LDPC Code used

- K(size of input message)=1000
- N(size of received output message with redundancy)=2000
- Parity check Matrix of Dimension 1000*2000.
- In the parity check matrix

- The number of ones in the row = 6
- The number of ones in the column = 3

BPSK for Encoding

BPSK (Binary Phase Shift Keying) Modulation is a digital modulation scheme that conveys data by changing, or modulating, two different phases of a reference signal.

bpskMod = 2*u - 1; where u is the input message.

Addition of AWGN

The properties of AWGN Noise are as follows:

- Additive : The received signal is equal to the transmitted signal plus noise.

  r = v + e

- White noise implies uniform power across the whole frequency band.
- The probability distribution of the noise samples is Gaussian.
- N0 = 1/(exp(EbN0(i)*log(10)/10));
  - Where EbN0 = [0, 0.5 ,1 ,1.5] in the experiment.

Error Correction:

The input message consists of all zeros as any error in the decoded message can be calculated from the relative number of ones compared(x) to the entire error message (w).

Eg in 1 test case in bit flipping.

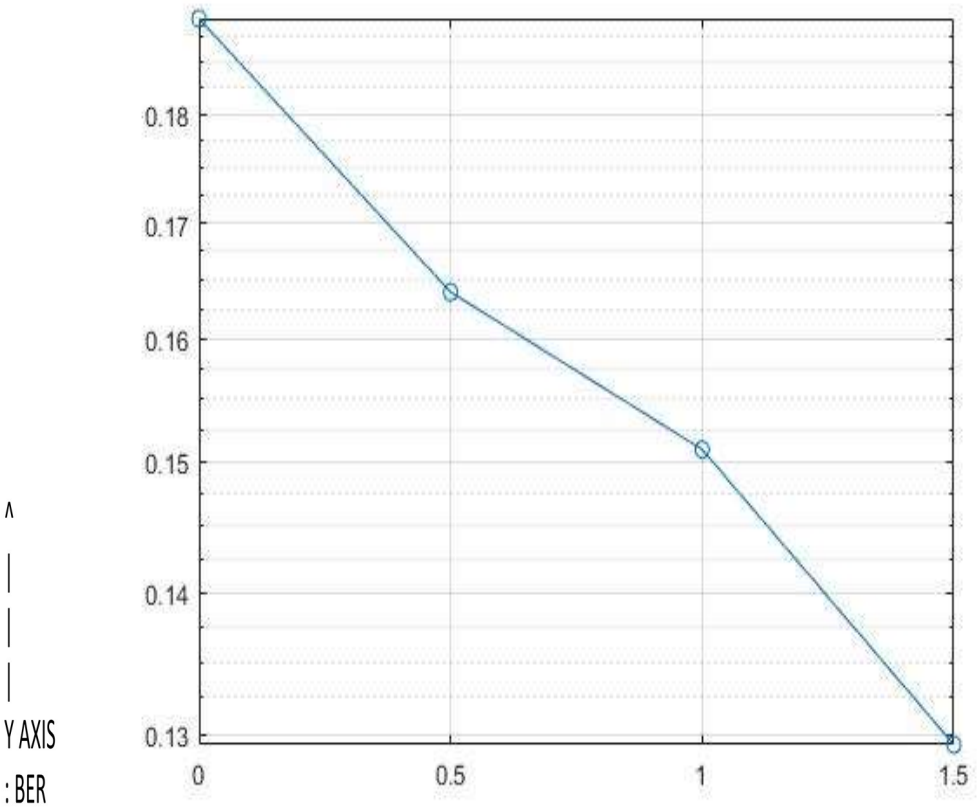
- Number of non zeros after decoding 203
- The bit error rate should be 203/2000.
- The bit error rate is 0.1015000.

Standard Deviation and Mean for Bit flipping

Noise (in dB)	0	0.5	1	1.5
Standard Deviation	0	1.025482	0.7696541	0.5626998
Mean	1.892500	0.9143750	0.6209000	0.4492875

Implication of the above data implies the bit error rate values over individual noise is spread around the mean.

- The overall standard deviation : 0.02506546
- The overall mean : 0.1583625
- 10 Test cases are considered and then averaged over.



X AXIS : EbNo ---------->

Bit error rate to noise in Bit flipping

Error Correction in Message Passing

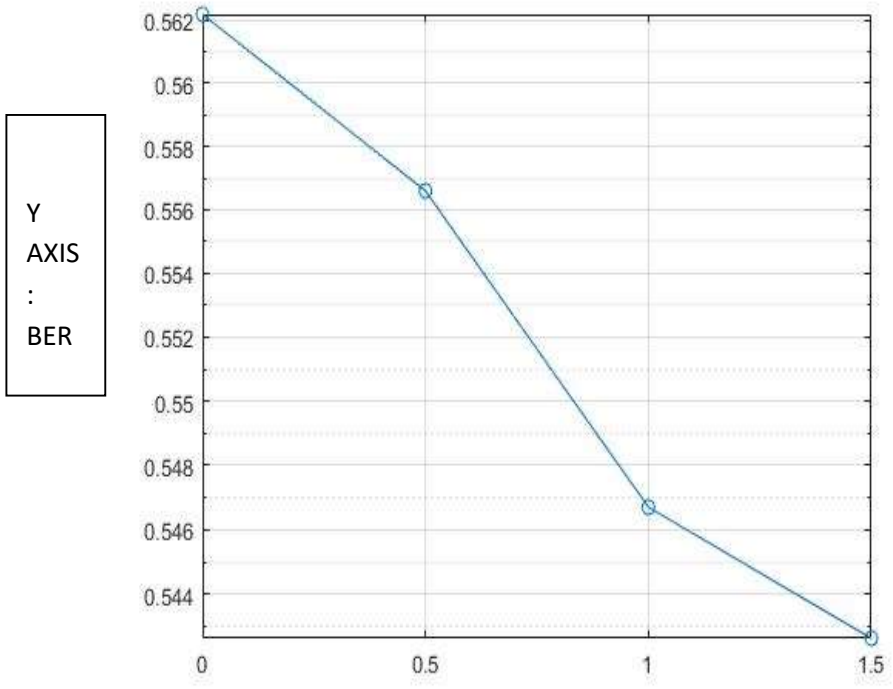
- Number of non zeros after decoding 1059
- The bit error rate should be 1059/2000.
- The bit error rate is 0.5295000

Standard Deviation and Mean for Message passing

Noise (in dB)	0	0.5	1	1.5
Standard Deviation	0	3.538221	2.833406	2.435675
Mean	5.622000	3.064100	2.195267	1.773000

Implication of the above data implies the bit error rate values over individual noise is spread around the mean.

- The overall standard deviation : 0.008957434
- The overall mean : 0.552037
- 10 Test cases are considered and then averaged over.



Error Correction for message passing with bit flipping

- Number of non zeros after decoding 183
- The bit error rate should be 183/2000.
- The bit error rate is 0.09100000.

Standard Deviation and Mean for message passing with bit flipping

Noise (in dB)	0	0.5	1	1.5
Standard Deviation	0	0.8211985	0.6011355	0.3847342
Mean	1.581500	0.7388250	0.4920333	0.3284250

Implication of the above data implies the bit error rate values over individual noise is spread around the mean.

- The overall standard deviation : 0.02815596
- The overall mean : 0.1248000
- 10 Test cases are considered and then averaged over.

^

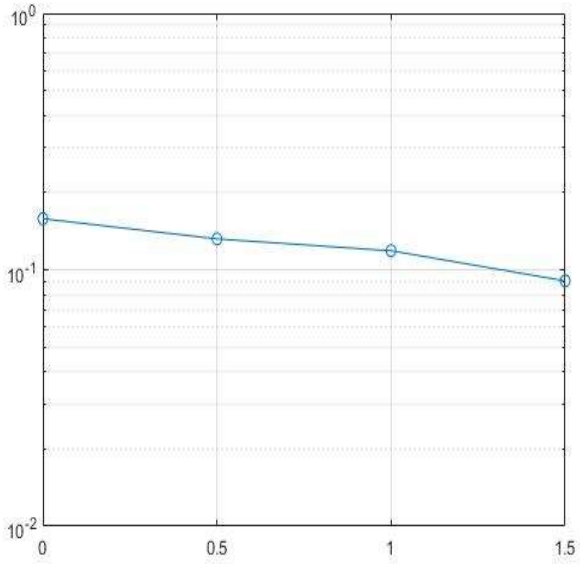
|

|

|

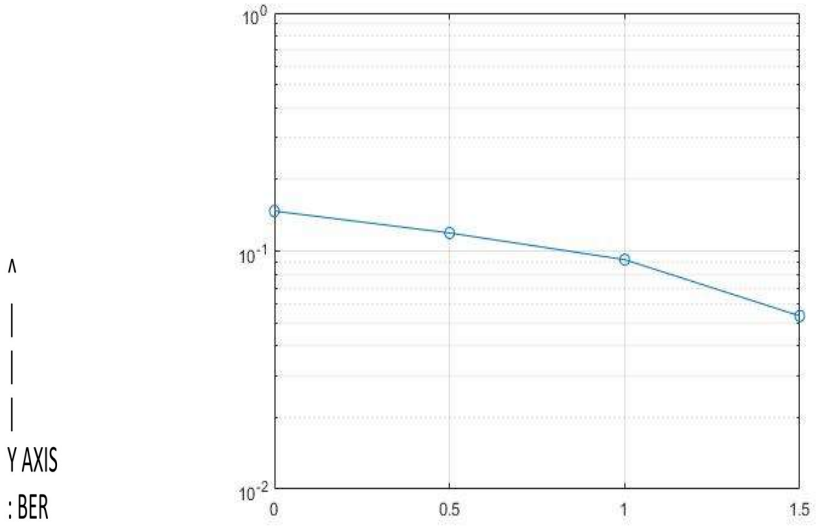
Y AXIS

: BER



X AXIS : EbNo ---------->

Bit error rate to noise for message passing with bit flipping



Comparison with probabilistic decoding

Importance of Message Passing Decoding Algorithm :

Low SNR (Signal to Noise Ratio) in case of minimum distance based decoding is a problem especially in deep space communication. The information from the Neighbours are used which incorporates the minimum distance concept too, showing a better SNR. Instead of an overall minimum distance, it is important to see how to achieve maximum likelihood decoding based on only those nodes that influence . This is what such a decoder tries to achieve.

Deep Learning to LDPC Decoding

A novel deep learning method for improving the belief propagation algorithm has been proposed in Nachmoni et. al.. The method generalizes the standard belief propagation algorithm by assigning weights to the edges of the Tanner graph. These edges are then trained using deep learning techniques. A well-known property of the belief propagation algorithm is the independence of the performance on the transmitted codeword. A crucial property of our new method is that our decoder preserved this property. Furthermore, this property allows us to learn only a single codeword instead of exponential number of codewords. Improvements over the belief propagation algorithm are demonstrated for various high density parity check codes.

Error correcting codes for channel coding are used in order to enable reliable communications at rates close to the Shannon capacity. A well-known family of linear error correcting codes are the low-density parity-check (LDPC) codes [10]. LDPC codes achieve near Shannon channel capacity with the belief propagation (BP) decoding algorithm, but can typically do so for relatively large block lengths. For high

density parity check (HDPC) codes [11], [12], [13], [14], such as common powerful algebraic codes, the BP algorithm obtains poor results compared to the maximum likelihood decoder [15]. In this work the focus is on HDPC codes to demonstrate how the BP algorithm can be improved. The naive approach to the problem is to assume a neural network type decoder without restrictions, and train its weights using a dataset that contains a large amount of code words. The training goal is to reconstruct the transmitted codeword from a noisy version after transmitting over the communication channel.

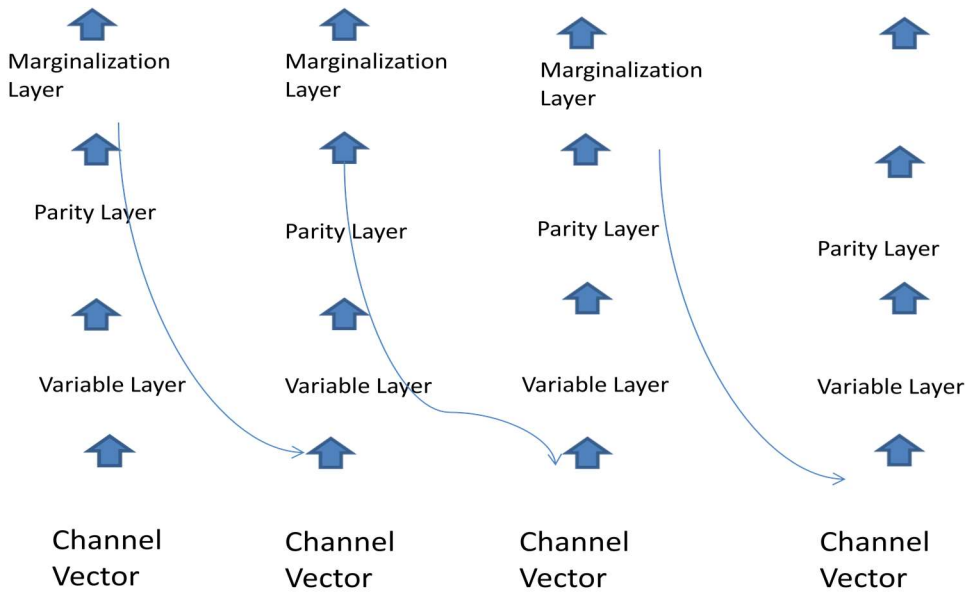
Unfortunately, when using this approach the decoder is not given any side information regarding the structure of the code. In fact it is even not aware of the fact that the code is linear. Hence it is required to train the decoder using a huge collection of codewords from the code, and due to the exponential nature of the problem, this is infeasible.

On top of that, the database needs to reflect the variability due to the noisy channel. In order to overcome this issue, the proposed approach in Nachmoni et al. is to assign weights to the edges of the Tanner graph that represent the given linear code, thus yielding a “soft” Tanner graph. These edges are trained using deep learning techniques. A well-known property of the BP algorithm is the independence of the performance on the transmitted codeword. A major ingredient in this new method is that this property is preserved by our decoder. Thus it is sufficient to use a single codeword for training the parameters of our decoder.

> Deep learning methods can improve the BP decoding of HDPC codes using a weighted BP decoder. [5]
>
> Belief Propagation algorithm is formulated as a neural network and it is shown that it can improve the decoding by 0.9dB in the high SNR regime. It is sufficient to train the neural network decoder using a single code word (e.g., the all-zero code word), since the architecture guarantees the same error rate for any chosen transmitted code word.
>
> An improved neural network architecture [6] has been considered for research motivation that achieves similar results to [5] with less parameters and reduced complexity. The main difference compared to [5] is that the offset min-sum algorithm is used instead of the sum-product algorithm, thus eliminating the need to use multiplications.

Later, Lugosch & Gross [12] proposed an improved neural network architecture that achieves similar results to [11] with less parameters and reduced complexity. The main difference was that they use the min-sum algorithm instead of the sum product algorithm. Gruber et al. [13] proposed a neural net decoder with an unconstrained graph (i.e., fully connected network) and show that the network gets close to maximum likelihood results for very small block codes, N = 16. Also, OShea & Hoydis [14] proposed to use an autoencoder as a communication system for small block code with N = 7. In this work we modify the architecture of [11] to a recurrent neural network (RNN) and show that it can achieve up to 1:5dB improvement over the belief propagation algorithm in the high SNR regime. The advantage over the feed forward architecture of [11] is that it reduces the number of parameters. We also investigate the performance of the RNN decoder on parity check matrices with lower densities and

fewer short cycles and show that despite the fact that we start with reduced cycle matrix, the network can improve the performance up to 1:0dB. The output of the training algorithm can be interpreted as a soft Tanner graph that replaces the original one. State of the art decoding algorithms of short to moderate algebraic codes, such as [15], [7], [10], utilize the BP algorithm as a component in their solution. Thus, it is natural to replace the standard BP decoder with our trained RNN decoder, in an attempt to improve either the decoding performance or its complexity. In this work we demonstrate, for a BCH(63,36) code, that such improvements can be realized by using the RNN decoder in the mRRD algorithm.



Recurrent Neural Network Architecture with unfold 4 which corresponds to 4 full BP iterations

Future Work:

- An Implementation of the neural network architecture in decoding oflow density parity check codes has been considered.
- The algorithms will be extended to irregular LDPC codes
- The number of cases will be increased for further experimentation.
-

References:

[1] R.G. Gallager, “Low Density Parity Check Codes,” MIT Press,Cambridge,1963.

[2] B.M.J. Leiner, “LDPC Codes – a Brief Tutorial”, April,2005

[3] https://sites.google.com/site/bsnugroho/ldpc

[4] E. Nachmani, Y. Be’ery, and D. Burshtein, “Learning to decode linear codes using deep learning,” in 54’th Annual Allerton Conf. On Communication, Control and Computing, September 2016, arXiv preprint arXiv:1607.04793.

[5] L. Lugosch and W. J. Gross, “Neural offset min-sum decoding,” in 2017

IEEE International Symposium on Information Theory, June 2017, arXiv preprint

arXiv:1701.05931.

[6] Mikale Simberg, “Linear-time encoding and decoding of low-density parity-check codes”, Aalto University, Master’s Thesis, 2015