University of the Witwatersrand

Masters Dissertation

Improving Iterative Soft Decision Decoding of Reed Solomon Codes Using Deep Learning

Author:
Kimberly Ntokozo Nkiwane

A Dissertation submitted in fulfilment of the requirements for the degree of Master of Science

in the

School of Electrical and Information Engineering

November 2024

Declaration

I, **Kimberly Ntokozo Nkiwane**, declare that this Dissertation titled, "**Improving Iterative Soft Decision Decoding of Reed Solomon Codes Using Deep Learning**" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this Dissertation has previously been submitted for a degree or any other qualification at this University or any other institution, has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this Dissertation is entirely my own work.
- I have acknowledged all main sources of help.

Signed: 

Date: 14/11/2024

UNIVERSITY OF THE WITWATERSRAND

Abstract

Engineering and the Built Environment
School of Electrical and Information Engineering

Master of Science

Improving Iterative Soft Decision Decoding of Reed Solomon Codes Using Deep Learning

by Kimberly Ntokozo Nkiwane

Supervisor: Yuval Genga

Telecommunications in the current information age is increasingly dependent on efficient transmission of data through a noisy channel. Therefore, utilizing Forward Error Correction (FEC) in the development of decoding algorithms is an active area of research. This dissertation work focuses on exploiting deep learning techniques and error correction techniques to improve iterative soft decision decoding of Reed Solomon codes (RS).

The parity check matrix of RS codes is characterized by a dense structure. This directly affects the exchange of soft information during the iterative decoding process. Therefore, to counter this issue, a bit-level implementation is utilized with the proposed decoding approach. Furthermore, additional techniques to add sparsity to the parity check matrix are presented in this research work. The proposed method for adding sparsity leverages the cyclical properties of RS codes to add low rate rows to the parity check matrix. This sparse implementation aids with the exchange of soft information during the message passing stage of the proposed iterative decoding process.

The implementation of deep learning techniques to improve iterative soft decision decoders are also presented in this dissertation. The proposed approach makes adjustments to the Neural Belief Propagation (NBP) algorithm for RS codes. The proposed NBP utilizes the sparse implementation presented in this research to improve exchange of soft information. This in turn leads to gains in error

correction performance without further adding complexity which is one of the main advantages of incorporating neural networks in the iterative decoding process.

Additionally, this dissertation proposes a Graph Neural Network (GNN) implementation for iterative soft decision decoding of RS codes. The approach employs the GNN architecture to construct a fully connected graph. This graph represents a message passing algorithm based on the Tanner graph, with trainable weights assigned to the graph nodes. This implementation improves the error correction performance of the proposed iterative soft decision decoder while reducing the number of iterations required to decode the received vector.

I dedicate this to God

For looking over me and protecting me.

I dedicate this to my Parents

For constantly being by my side and supporting me.

I dedicate this to my Brother

For always pushing me to reach for the stars.

Acknowledgements

I would like to express my deepest gratitude to Dr. Yuval Genga for his supervision, patience, and unwavering support throughout this journey. His dedication has been instrumental to my success, and for that, I am immensely grateful.

My sincere thanks also go to the University of Witwatersrand for their generous sponsorship of my studies, providing me with the resources to pursue my academic goals.

To my friends, your encouragement and motivation have been a source of strength. Your belief in me has pushed me to continue, even when challenges arose.

Lastly, my heartfelt appreciation to my family, whose financial assistance and emotional support have been my pillars. I am truly blessed and eternally thankful for having you by my side every step of the way.

Contents

List of Figures

List of Tables

Abbreviations

ABP	**A**daptive **B**elief **P**ropagation
AWGN	**A**dditive **W**hite **G**aussian **N**oise
BCH	**B**ose-**C**haudhuri-**H**ocquenghem
BER	**B**it **E**rror **R**ate
BM	**B**erlekamp **M**assey
BP	**B**elief **P**ropagation
BPSK	**B**inary **P**hase **S**hift **K**eying
ECC	**E**rror **C**ontrol **C**oding
FEC	**F**orward **E**rror **C**orrection
GF	**G**alois **F**ield
GNN	**G**raph **N**eural **N**etwork
GNNBP	**G**raph **N**eural **N**etwork **B**elief **P**ropagation
GS	**G**uruswami-**S**udan
HDPC	**H**igh **D**ensity **P**arity **C**heck
KV	**K**oetter-**V**ardy
LDPC	**L**ow **P**arity **D**ensity **C**heck
MNBP	**M**odified **N**eural **B**elief **P**ropagation
MDS	**M**aximum **D**istance **S**eparable
NBP	**N**eural **B**elief **P**ropagation
SPA	**S**um **P**roduct **A**
SPCM	**S**parse **P**arity **C**heck **M**atrix
NPCM	**N**ormal **P**arity **C**heck **M**atrix
SNR	**S**ignal-to-**N**oise **R**atio
RS	**R**eed **S**olomon

A graph	$\mathscr{G}$
Edge-Set	$\mathscr{E}$
Node-Set	$\mathscr{V}$ and $\mathscr{U}$
Function f with a of domain $\mathbb{A}$ and range $\mathbb{B}$	$f : \mathbb{A} \to \mathbb{B}$
A function of x parametrized by θ	$f(\mathrm{x}; \theta)$
A feature vector F of node x	F_{x}

CHAPTER 1

Introduction

1.1 Introduction

At the center of any modern communication scheme is the need for efficient transmission of soft information transmitted through a noisy channel. In order to enable this, Forward Error Correction (FEC) codes [1] are used to improve exchange of information. FEC codes ensure efficient exchange of information through error detection and correction of received information from a noisy channel. There are several types of FEC codes such as Low Density Parity Check (LDPC) codes [2], Bose-Chaudhuri-Hocquenghem (BCH) codes [3], Reed Solomon (RS) codes [4].

RS codes are a popular class of non-binary linear block codes [4]. They have been used in various applications that range from Quantum computing [5, 6], Internet of Things (IoT) systems [7] and storage systems [8, 9]. They are denoted as (n, k) RS codes where n is defined as the length of the codeword, k is defined as the number of information symbols and $(n-k)$ is the number of redundant symbols. RS codes are classified as Maximum Distance Separable (MDS) codes [10] which means they can correct up to $t = \left\lfloor \frac{n-k}{2} \right\rfloor$ number of errors in a received codeword. Since they are classified as MDS codes, the minimum distance can be defined using the equation $d_{min} \leq n - k + 1$. This property ensures optimal error correction capabilities, as MDS codes can correct the maximum number of symbol errors [10]. Due to their optimal error correction performance, they continue to be an interesting target for research [6, 7, 11, 12].

There are various elements to consider when selecting a compatible decoder. These elements include the decoding complexity and error correction performance. By investigating these elements, we can evaluate the suitability of a decoding algorithm. There are two approaches of decoding RS codes namely hard and soft-decision decoding. Algebraic hard-decision decoding algorithms such as the Berkelamp Massey (BM) [13] work well with RS codes. However, hard-decision decoders incur performance losses when soft information is available [13]. Therefore, soft-decision decoding methods such as Generalized Minimum Distance (GMD) [14] algorithm and Algebraic Soft Decision (ASD) [15] outperform hard-decision decoders. Koetter and Vardy (KV) [16] is a proposed algebraic soft-decision decoder which takes advantage of the algebraic properties of RS codes. This approach significantly outperformed the hard-decision decoder and reached over 2-3 dB gains [15]. However, the complexity becomes prohibitively larger for longer codes [15]. Therefore, other soft-decision decoding methods such as iterative decision decoders have been explored to improve on decoding performance.

Iterative soft-decision decoding continues to be a subject of ongoing research [17–20]. This approach involves modifying the received channel output vector by updating reliability information iteratively. An early attempt on iterative soft-decision algorithms was the Belief Propagation (BP) algorithm popular for decoding LDPC [2]. Due to the good error correction performance with LDPC codes, the BP algorithm was applied on RS codes [21]. A major challenge for this implementation is the dense nature of the parity check matrix of RS codes which directly affects error correction performance. This is because messages passed from the check to variable nodes in the Tanner graph [2] get stuck in pseudo equilibrium points due to the occurrence of small cycles. These small cycles are detrimental for the performance of the decoder and exchange of soft information in the decoding process. Additionally, the algorithm has a relatively high computational complexity.

Jiang et al [22] introduced the Adaptive Belief Propagation (ABP) algorithm. The algorithm works by iteratively adapting the parity check matrix and performing

a Binary Image Expansion on the same matrix (BIE). This in turn results in increased sparsity of the parity check matrix and reduces the occurrence of small cycles that degrade decoding performance. The gains come at the cost of an increased computational complexity. This led to the work in [23] which applied the Sum Product Algorithm (SPA) a version of the BP algorithm to RS codes. This approach improved the overall convergence rate of the iterative decoder but did not tackle the computational complexity vs performance issue. The approach in [24] highlighted how to increase the sparsity of a non-binary parity check matrix then implemented the Q-ary Sum Product Algorithm (QSPA). The approach gave a good error correction performance without further increasing computational complexity. Although, various avenues of iterative decoding of RS codes have been explored, improving the error correction performance of RS codes is still ongoing research.

Deep learning techniques have shown great potential in solving problems in FEC codes [25–27]. Nachmani et al [18] proposed applying deep learning techniques on the BP algorithm for BCH codes and LDPC codes. When compared to the original BP implementation, applying deep learning improved the error correction performance. It is important to highlight that the deep learning approach improved performance without increasing the computational complexity cost of the BP decoder. Zhang et al [17] investigated deep learning approach on Stochastic Shifting Based Iterative Decoding (SSID) for RS codes using BP algorithm. They implemented a neural network decoder by applying pre-trained weights on the nodes of the tanner graph. The variable weights are based on the damping factor to replace the scalar parameters. The proposed approach improved error correction performance compared to the original SSID BP implementation [17]. There was an improvement in the error correction performance as a result of applying deep learning without an increase of computational complexity. Due to deep learning techniques improving iterative soft-decision decoding other deep learning approaches have been explored to further improve the error correction performance gains of iterative soft-decision decoders.

Graph Neural Networks (GNN) [28–30] are a class of neural networks designed to work with data that can be represented in graphs. Graphs are mathematical structures that consist of nodes (vertices) and edges connecting those nodes. GNN have gained significant attention in various fields including IoT systems [31], and Quantum error correction [32]. Crammer et al [33] utilized this background to build a graph structure based on the Tanner graph. The messages exchanged between the check and variable nodes, as well as their respective neighbors, are structured into multiple GNN layers. This implementation was tested with BCH and LDPC codes. The method has significant performance gains with less iterations compared to the artificial neural belief propagation algorithm.

In this research, we investigate the application of error correction and deep learning techniques to improve iterative soft-decision decoding of RS codes. The objective is to construct a sparse parity check matrix that facilitates efficient exchange of soft information during the iterative decoding process. The impact of the sparse implementation on the error correction performance is investigated using the BP algorithm and the artificial neural belief propagation algorithm, which is adapted for RS codes. Existing literature demonstrates that the application of GNN based BP decoders has primarily been implemented on binary codes. Therefore, this work proposes a GNN-based BP decoder adapted for RS codes, which are inherently non-binary. This adaptation is achieved by implementing the decoder on a bit-level, resulting in longer and relatively denser codes compared to BCH codes. The primary aim of this adaptation is to reduce the number of iterations required for decoding the received vector, referred to as the iterative convergence rate and improve error correction performance.

1.2 Problem Statement

RS codes are characterized by a dense parity check matrix which directly affects the performance of soft-input and soft-output iterative decoders. This is due to the increased occurrence of small cycles. Therefore messages sent between edges of the Tanner graph get stuck on pseudo equilibrium points which degrades iterative

decoding performance. Hence, this research work will look into ways of adding sparsity to parity check matrix with the aim of improving the iterative decoding performance.

The artificial neural belief propagation algorithm [17, 18] yields a performance gain compared to the BP decoding algorithm without further increasing computational complexity [18]. This research aims to leverage this approach to improve the error correction performance of soft-decision iterative decoding for RS codes, alongside other error correction techniques. These techniques include investigating the addition of sparsity to the RS parity check matrix. The objective is to improve error correction performance without further increasing complexity.

Furthermore, in order to improve performance of iterative soft-decision decoding while reducing the number of decoding iterations, this work proposes a GNN based BP decoder for RS codes. In literature [33, 34], this implementation results in improved decoding performance while reducing the number of decoding iterations when compared to the artificial neural belief propagation for binary linear block codes. Therefore a similar approach is applied to RS codes in order to replicate the good error correction performance.

1.3 Research Question

The research work will answer the following question:

What error correction and deep learning techniques can be applied to iterative soft-decision decoding of RS codes to improve error correction performance?

1.4 Research Objectives

The main objective is to improve the performance of iterative soft-decision decoding of RS codes using deep learning. It is known that RS codes possess a dense parity check matrix, which directly affects error correction performance. Therefore, this research will focus on :

1. Developing a method to add the sparsity to the parity check matrix for RS codes by leveraging the cyclical properties possessed by this class of codes. This is done with the aim of improving exchange of information during the iterative decoding process.

2. Adaptation of the artificial neural belief propagation for RS codes. This is done with the aim of improving the error correction performance for these codes without adding computational complexity.

3. Developing a GNN model based iterative soft-decision decoder for RS codes. The aim of this is to develop a GNN model for RS codes to improve error correction performance and reduce the number of iterations required to decode the received vector.

1.5 Organization Of Dissertation

The remainder of this dissertation is organized as follows.

Chapter 2 - *Background of Forward Error Correction Principles.* This chapter focuses on the principles behind Forward Error Correction (FEC) techniques and provides a foundation of RS codes used in this research work. In addition, this chapter provides a short overview of the BP algorithm and other advanced deep learning soft-decision iterative decoders for RS codes. Additional background information on neural networks is covered in preparation for later chapters.

Chapter 3 - *Research Methodology.* This chapter covers the process followed to answer the research question. In several sections we present the core work of this research from generation of data to model design and evaluation of the deep learning implementations presented in work.

Chapter 4 - *Improvements to Iterative Soft Decision Decoding for RS codes.* The proposed improvements include increasing sparsity on the RS code parity check matrix. These improvements are tested on the BP algorithm and the artificial

neural belief propagation. The results of the computer simulations implementing these improvements are discussed.

Chapter 5 - *Proposed Graph Neural Network Implementation on the Belief Propagation Algorithm for RS codes*. This chapter presents the GNN implementation on BP algorithm for RS codes. This implementation is tested for different code rates. The performance is compared with the proposed artificial neural belief propagation.

Chapter 6 - *Conclusion and Future Work*. This chapter presents a conclusion of the research done is this dissertation. This summary is based on the results from the simulations.

CHAPTER 2

Background Review: Forward Error Correction Principles

2.1 Introduction

This chapter covers the basics of Forward Error Correction (FEC), and the fundamentals of Reed Solomon(RS) error correction codes. In addition, the principles behind the Belief Propagation (BP) algorithm are introduced along with different techniques used to improve iterative soft decision decoders. Furthermore, this chapter introduces deep learning and how this approach has improved iterative soft decision decoding.

2.2 Forward Error Correction (FEC)

Forward Error Correction (FEC) [35–37] is a method that enables communication systems to detect and correct errors in the data that is transmitted through a noisy channel. The technique appends additional information to the original message before transmitting it over a noisy channel, in order for the receiver to be able to correct errors without retransmission of data. Fig. 2.1 shows the different classes of FEC codes.

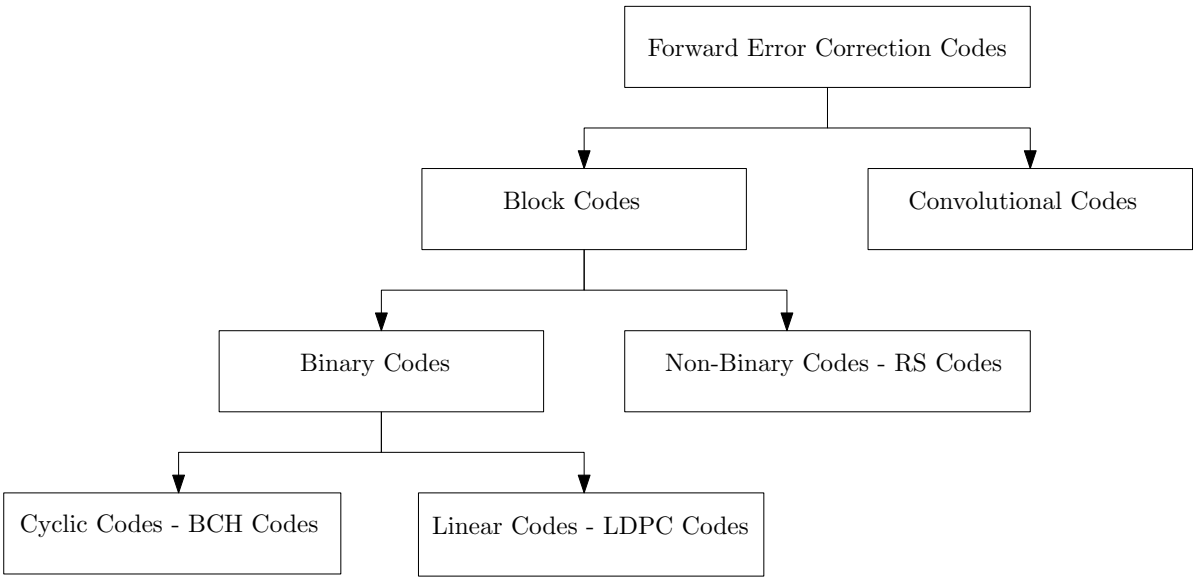


FIGURE 2.1: Classification Of FEC codes

As shown in Fig. 2.1 there are two main categories of FEC codes namely block codes [1] and convolutional codes [1]. The main difference is that block codes split the data into blocks of a fixed size and encode each block, while convolutional codes encode the data continuously as it arrives, taking into account the previous bits in the data stream. Linear block codes are preferred over non-linear codes in this research work due to their advantageous linear properties [1]. Some of the common examples of FEC block codes that will be discussed in this research are Reed Solomon codes (RS) [4], Bose-Chaudhuri-Hocquenghem (BCH) codes [3] and Low Density Parity Check codes (LDPC) [2].

2.3 Linear Block Codes

2.3.1 Construction Of Linear Block Codes

A linear block code is described as a (n,k) code where n is the length of the linear block code and k is the number of information bits. The parity symbols, t, are defined as

$$t = n - k, \tag{2.1}$$

and the rate of the block is defined as a ratio

$$\mathcal{R} = \frac{k}{n}. \tag{2.2}$$

Definition 2.1: Hamming Distance [1] of two codewords belonging to the code c is defined as the difference of the total number of position between c_1 and c_2 of equal length. This distance is denoted as the Hamming distance between c_1 and c_2 belonging to the code c. It is expressed as

$$d_h(c_1, c_2). \tag{2.3}$$

Definition 2.2: Hamming Weight [1] of a codeword, $c = [c_1, ..., c_n]$, is denoted as $w(c)$ and is defined as the number of non zero components of c. The relation between hamming distance and hamming weight is expressed as follows

$$d(c_1, c_2) = w(c_1 - c_2). \tag{2.4}$$

Definition 2.3: The minimum Hamming distance [1] of a linear block code is expressed as the smallest distance between c_1 and c_2

$$d_{min} = min(d_h(c_1, c_2 : c_1, c_2)). \tag{2.5}$$

Definition 2.4: Maximum Distance Separable (MDS) codes [38] are a class of error-correcting codes whose minimum distance for a given block length and code size is represented by

$$d_{min} = n - k + 1. \tag{2.6}$$

These codes possess the ability to correct upto λ number of errors and can be obtained as

$$n - k = 2\lambda, \tag{2.7}$$

where λ is expressed as follows

$$\lambda = \left\lfloor \frac{d_{min} - 1}{2} \right\rfloor, \tag{2.8}$$

and $\lfloor \cdot \rfloor$ is the floor function.

2.3.2 Finite Field Arithmetic

A finite field is often referred to as a Galois field. A Galois field is defined as $GF(q^b)$ where q is a prime number and b is a positive nonzero integer.

Listed below are some of the properties of a finite fields:

1. Closure under two operations known as addition $A + B$ and multiplication $A \times B$, where A and B are defined as an element in the finite field.

2. Adding or multiplying two elements from the finite field results in an element in the same finite field.

3. The additive identity 0 must exist such that $A + 0 = A$ for all the elements in the finite field.

4. The multiplicative identity of an element in the field is such that $A \times 1 = A$ for all the elements

   in the finite field.

5. For every element in the finite field, there is an additive inverse element A in the finite field, and there exists another element B in the finite field where $A + B = 0$. The element B is defined as $-A$.

6. For every element in the finite field, there is an multiplicative inverse element A in the finite field, and there exists another element B in the finite field where $A \times B = 1$. The element B is defined as A^{-1}.

7. The associative, $(A \times B) \times D = A \times (B \times D)$, and commutative, $(A \times B) = (B \times A)$, laws apply for all the elements A, B, D in the finite field.

Definition 2.5: A **Galois Field** is made up of a finite set of elements and arithmetic functions such as addition, multiplication can be performed. The Galois Field is denoted as $GF(q^b)$ where q is a prime number and b is a positive nonzero integer.

Definition 2.6: A **Primitive element** of $GF(q^b)$ is a non zero element β, in the field. That is, the power of the primitive elements β can be used to generate all the nonzero element in the field.

2.3.3 The Generator Matrix (G), Parity Check Matrix (H) and Syndrome Of A Linear Block Code

The generator matrix for an (n, k) linear block code $\mathcal{C}$ is defined by k linear independent vectors that are codewords belonging to the same sub-space as C and the generator matrix has the dimensions $k \times n$. The generator matrix G is used generate codewords $\mathbf{c}$ from a message $\mathbf{M}$ using (2.9)

$$\mathbf{c} = \mathbf{M} \times \mathbf{G}. \tag{2.9}$$

The generator matrix can be represented in systematic form and is expressed as follows

$$\mathbf{G} = [P, I_k], \tag{2.10}$$

where I_k is the $(k \times k)$ identity matrix and P is a $k \times (n - k)$ parity sub-matrix.

The parity check matrix **H** of an (n, k) linear block code is known as the dual code of the generator matrix. Similar to the **G** matrix, the **H** matrix can be represented as a systematic parity check matrix. The matrix is partitioned into an information sub-matrix and a parity sub-matrix. These dimensions are represented as follows:

$$\mathbf{H} = [I_{(n-k)}, P^T]. \tag{2.11}$$

The parity check matrix exists in the dual space with the generator matrix. The parity check matrix is denoted as **H**. The **G** and **H** matrices have to satisfy

$$\mathbf{G} \times \mathbf{H}^T = 0. \tag{2.12}$$

Since G is made up of k linearly independent codewords, a valid codeword satisfies (2.13). This is referred to as the syndrome check S. The syndrome check S is represented as a zero vector of length $(n - k)$

$$\mathbf{s} = \mathbf{c} \ \times \ \mathbf{H}^T = 0. \tag{2.13}$$

If the syndrome **s** is a zero vector of length $(n - k)$ then the syndrome check is satisfied and the codeword is considered valid. If this is not the case then the decoder has detected errors in the received codeword. As seen (2.13), the **H** matrix is essential for the syndrome calculation and is commonly used as a stopping condition during soft decision iterative decoders of RS codes.

2.4 Reed Solomon Codes (RS)

RS codes are a class of non-binary BCH codes which were introduced in 1960 by Irving S.Reed and Gustave Solomon [4]. RS codes are classified as linear cyclic codes and have been used in a number of communication systems due their favorable algebraic properties [6, 23, 39]. They have some advantages that make them

compatible with different decoding schemes. Similar, to other linear block codes, the transmitted message is divided into different blocks as shown in Fig. 2.2.

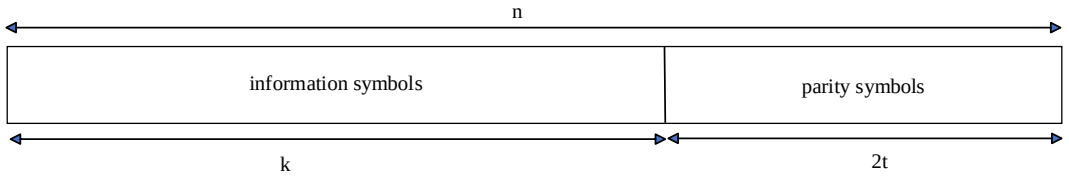


FIGURE 2.2: Reed Solomon (RS) Codeword

Fig. 2.2 illustrates an example of an RS codes with a systematic structure, where the codeword is divided into two blocks, the information symbols and parity symbols. The information block consists of k information symbols and the parity block consists of $n-k$ parity symbols. RS codes consist of the following properties

1. Block length: $n = q^b - 1$,
2. Length of parity check symbols : $t = \left\lfloor \frac{n-k}{2} \right\rfloor$,
3. Length of information symbols : $k = n - 2t$,
4. Minimum Distance : $d_{min} = 2t + 1$.

2.4.1 Generator Polynomial And Parity Check Polynomial

A (n, k) RS code can be defined by the generator polynomial $g(x)$ [37] and the parity check matrix polynomial. The generator polynomial of the RS codes over $GF(q^b)$ of length $n = q^b - 1$ has a degree of $(n-k)$ and can be denoted as

$$g(x) = \prod_{i=k}^{n-1} (x - \beta^{-i}). \tag{2.14}$$

The parity check polynomial $h(x)$ [37] is obtained using similar calculation as that of the generator polynomial $g(x)$. The parity check polynomial $h(x)$ has a degree k and is expressed as

$$h(x) = \prod_{i=0}^{k-1}(x - \beta^{-i}). \tag{2.15}$$

2.4.2 The Generator Matrix And Parity Check Matrix

From (2.15) the generator polynomial $g(x)$ can be rewritten as

$$g(x) = (x - \beta^{-k})(x - \beta^{-(k+1)}) \cdots (x - \beta^{-(n-1)}), \tag{2.16}$$

$$g(x) = g_0 + g_1 x + g_2\, x^2 + \cdots + g_{n-k}\, x^{n-k}. \tag{2.17}$$

Based on the properties of cyclic codes, the generator matrix G of a (n, k) RS code can be created by cyclically shifting coefficients of the polynomial k times to give the form shown in

$$\mathbf{G} = \begin{bmatrix} g_0 & g_1 & g_2 & \cdots & g_{(n-k)} & 0 & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & g_2 & \cdots & g_{(n-k)} & 0 & \cdots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \cdots & g_{(n-k)} & \cdots & 0 \\ \vdots & & & & \vdots & & & \cdots & \vdots \\ 0 & 0 & \cdots & 0 & g_0 & g_1 & g_2 & \cdots & g_{(n-k)} \end{bmatrix}. \tag{2.18}$$

The generator matrix can also be represented in systematic form as follows

$$\mathbf{G} = \begin{bmatrix} P_{1,1} & P_{1,2} & \cdots & P_{1,n-k} & 1 & 0 & \cdots & 0 \\ P_{2,1} & P_{2,2} & \cdots & P_{2,n-k} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ P_{k,1} & P_{k,2} & \cdots & P_{k,n-k} & 0 & 0 & \cdots & 1 \end{bmatrix}, \tag{2.19}$$

where $P_{i,j}$ represents the elements of $GF(q^b)$ in the parity sub-matrix, as defined in (2.10).

Another common representation of the the generator matrix is referred to as a Vandermonde matrix and is shown in

$$\mathbf{G}_{\text{van}} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & \beta^1 & \dots & \beta^{(n-1)} \\ 1 & \beta^2 & \dots & \beta^{2(n-1)} \\ \vdots & \vdots & \dots & \vdots \\ 1 & \beta^{k-1} & \dots & \beta^{(k-1)(n-1)} \end{bmatrix}, \tag{2.20}$$

where β represents the primitive element of the field.

The parity check matrix $\mathbf{H}$ is also obtained in a similar way to that of the generator matrix. Firstly, the equation for the parity check polynomial is expanded from the form shown in (2.15) [37].

$$h(x) = (x - \beta^0\ )(x - \beta^{-1}\ ) \cdots (x - \beta^{-(k-1)}\ ), \tag{2.21}$$

$$h(x) = h_0 + h_1\ x + h_2\ x^2 + \cdots + h_k\ x^k. \tag{2.22}$$

The parity check matrix is then created by shifting the coefficients of $h(x)$ by $(n-k)$ times as shown in (2.23)

$$\mathbf{H} = \begin{bmatrix} h_k & h_{(k-1)} & h_{(k-2)} & \dots & h_0 & 0 & 0 & \dots & 0 \\ 0 & h_k & h_{(k-1)} & h_{(k-2)} & \dots & h_0 & 0 & \dots & 0 \\ 0 & 0 & h_k & h_{(k-1)} & h_{(k-2)} & \dots & h_0 & \dots & 0 \\ \vdots & & & & \vdots & & & \dots & \vdots \\ 0 & 0 & \dots & 0 & h_k & h_{(k-1)} & h_{(k-2)} & \dots & h_0 \end{bmatrix}. \tag{2.23}$$

Similar to the generator matrix, the parity check matrix can also be represented in systematic form as follows

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & P_{1,1}^T & P_{1,2}^T & \dots & P_{1,k}^T \\ 0 & 1 & 0 & \dots & 0 & P_{2,1}^T & P_{2,2}^T & \dots & P_{2,k}^T \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & \dots & 1 & P_{k,1}^T & P_{k,2}^T & \dots & P_{k,k}^T \end{bmatrix}, \tag{2.24}$$

where $P_{i,j}^T$ represents the elements of $GF(q^b)$ in the transposed parity sub-matrix, as defined in (2.11).

Similar to the generator matrix, the parity check matrix can also be represented in the Vandermonde matrix form as follows

$$\mathbf{H}_{\text{van}} = \begin{bmatrix} 1 & \beta^1 & \dots & \beta^{(n-1)} \\ 1 & \beta^2 & \dots & \beta^{2(n-1)} \\ \vdots & \vdots & \dots & \vdots \\ 1 & \beta^t & \dots & \beta^{t(n-1)} \end{bmatrix}. \tag{2.25}$$

2.4.3 Encoding Of Reed Solomon Codes

Let the message vector $\mathbf{m}$ to be defined as

$$\mathbf{m} = [\mathbf{m}_0, \mathbf{m}_1, \dots, \mathbf{m}_{k-1}], \tag{2.26}$$

the message polynomial $M(x)$ is defined as

$$M(x) = M_0 + M_{1x} + ... + M_{k-1}x^{k-1}. \tag{2.27}$$

The encoding process involves multiplication of $M(x)$ with the generator polynomial $g(x)$ as shown in (2.29)

$$g(x) = g_0 + g_{1x} + g_{2x} + ... + g_{n-1}x^{n-k}, \tag{2.28}$$

$$c(x) = M(x) \times g(x). \tag{2.29}$$

2.4.4 Decoding Of Reed Solomon Codes

The problem of effectively decoding RS codes has be investigated for many years [12, 15, 21, 23, 40]. This has led to the implementation of various algorithms and techniques for decoding this class of codes. There are two categories of decoding techniques, namely hard decision and soft decision decoders. Soft decision decoders such as the ones presented in [16, 40–42] have shown significant gain in error detection and correction. In addition, exploration of deep learning techniques [27, 43] has become a point of interest for researchers to further improve error correction performance of iterative soft decision decoding of RS codes.

2.5 Iterative Soft Decision Algorithms

2.5.1 Belief Propagation Algorithm

Popular iterative soft decision algorithms, such as the Belief Propagation (BP) algorithm, also known as the Sum Product algorithm (SPA), are well-known message passing algorithms famous for working with LDPC codes [2]. The BP decoder significantly improved performance for soft decision decoders. This algorithm is characterized as a message passing algorithm due to the exchange of messages through a graph structure. The graph structure is represented by a graphical model commonly referred to as a Tanner graph [44]. The Tanner graph is a representation of both the parity check matrix of a code and its decoding algorithm. The algorithm functions by iteratively exchanging messages between the nodes of the graph, referred to as the variable nodes associated to the columns in the **H** matrix, and check nodes associated to the rows in the **H** matrix. The edges represent the connections between the variable and check nodes which correspond to the non-zero bits in the parity check matrix.

To better illustrate the Tanner graph, take an example of a (7,4) linear block code, the parity check matrix H is expressed as follows

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}. \tag{2.30}$$

Fig. 2.3 illustrates the corresponding Tanner graph representation of the parity check matrix (2.30).

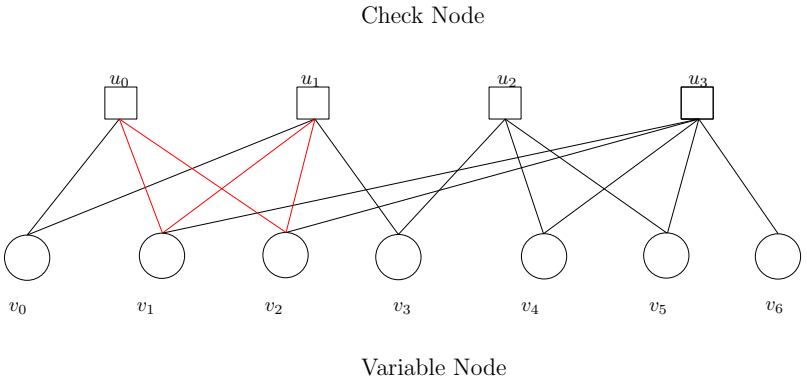


FIGURE 2.3: Tanner Graph Representation

As shown in Fig. 2.3, the two groups of nodes in a Tanner graph are the variable nodes v (column) and the check nodes u (row). The rule for drawing a Tanner graph for a code is that there is an edge connection between check node and variable node if the element $H_{vu} = 1$. Fig. 2.3 also shows short cycles(denoted by the red lines) of length 4 in the Tanner graph which affect the exchange of soft information due to messages getting stuck in pseudo equilibrium points. These short cycles degrade the performance of the BP algorithm when applied for RS codes. This a major issue for iterative decoders like the BP algorithm. There are two versions of the BP algorithm which are known as the probability and log domain BP algorithm. In this research we will work with the log-domain BP algorithm [18, 21, 23] for a fair comparison with our model.

In this research we will work with the log-domain BP algorithm for a fair comparison with our model

Algorithm 1: Log-Domain BP algorithm

Input: The received vector,r, from the channel. The probabilities $P(c_j = 1|r_j)$ and $P(c_j = 0|r_j)$. The maximum number of iterations $\mathcal{T}_{max}$ is set.

Output: Decoded vector $\hat{r}$

. **Initialize:** Obtain the the LLR values $\frac{P(c_j=1|r_j)}{P(c_j=0|r_j)}$ based on the channel. Set

$$L(p_{i,j}^{(1)}) = L_j \text{ for all } (i,j) \text{ with } H_{i,j} = 1$$

.

repeat

- **Check Node Update:** The extrinsic information ,$L(a_{i,j})$ is updated based on the LLR values,

  $L(e_{(i,j)}^{(f)}) = 2\tanh^{-1}\left(\prod_{\grave{j}=i,\grave{j}\neq j,H_{(i,\grave{j})}=1}^{n} \tanh\left(\frac{L(p_{i,j})}{2}\right)\right)$.

- **Variable Node Update:** LLR values L_j are updated based on the extrinsic information,

  $L(p_{i,j})^{(\mathcal{T}+1)} = L_j + \sum_{\grave{i}=1,\grave{i}\neq i,H_{i,j=1}}^{(n-k)} + a_{i,j}^{(\mathcal{T})}$.

- **Calculation Of Total LLR:**

  $L(Total)_j = L_j + \sum_{i=1,H_{i,j}=1}^{(n-k)} La_{i,j}^{(\mathcal{T})}$,

- **Hard Decision Vector $\hat{r}$:**

  $$\hat{r} = \begin{cases} 1 & if L(Total)_j > 0, \\ 0 & if L(Total)_j < 0. \end{cases}$$

- **Syndrome Check**: $S = \hat{r} \times H^T$

- **Stopping Criteria:**

  - If the syndrome check condition $S = 0$ is satisfied the algorithm breaks.
  - If the condition is not met $S \neq 0$ then $\mathcal{T} = \mathcal{T} + 1$

until $\mathcal{T} = \mathcal{T}_{max}$;

2.5.2 Log-Domain Belief Propagation

The log-domain BP algorithm [1] iteratively handles soft information in the form of log-likelihood ratios. The log-domain BP algorithm has two stages namely the variable node update and the check update in order to correct the extrinsic and intrinsic information respectively. Algorithm 1 shows the pseudo code for log-domain BP algorithm for an iteration $\mathcal{T}$.

2.5.3 Binary Image Expansion (BIE)

During the decoding process, the conversion from symbol-level to bit-level is also applied to the parity check matrix, which plays a crucial role in the decoding process. This conversion is accomplished through a process known as Binary Image Expansion (BIE) [37].

BIE involves the process of converting a symbol-matrix into its bit representation. In the context of the parity check matrix of a RS code, the BIE technique utilizes the bit-level representation illustrated in the Fig. 2.4.

Fig. 2.4 shows the bit representation for elements in $GF(2^3)$ and expresses the order of the elements in the field. The BIE approach is employed to construct an $b \times b$ companion matrix [45], where b is the number of bits in the binary representation based on the order of the element in the field as shown in Fig. 2.4. The corresponding companion matrix replaces each corresponding symbol when the code is converted to its bit representation.

Consider a (7,5) RS code, the corresponding parity check matrix as expressed as follows

$$\mathbf{H} = \begin{bmatrix} 1 & 2 & 4 & 3 & 6 & 7 & 5 \\ 1 & 4 & 6 & 5 & 2 & 3 & 7 \end{bmatrix}. \tag{2.31}$$

The BIE [45] works on each symbol based on its position in the H matrix presented in (2.31) as mentioned previously. From [21, 45] the companion matrix is based on the symbol order presented in Fig. 2.4. The companion matrix is constructed by using the bit representation of the reference symbol and the next $b-1$ symbols as

β^0	β^1	β^2	β^3	β^4	β^5	β^6	β^7
1	2	4	3	6	7	5	1
1	0	0	1	0	1	1	1
0	1	0	1	1	1	0	0
0	0	1	0	1	1	1	0

FIGURE 2.4: Binary Representation Of Each Symbol

shown in (2.32). For example, the binary representation of the β^0 and β^1 symbols in the H matrix are expressed as

$$\beta^0 = 1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \beta^1 = 2 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{2.32}$$

A similar approach is done for all the elements and the corresponding symbols in (2.31) to give the matrix shown in (2.33)

$$\mathbf{Hb} = \left[\begin{array}{ccc|ccc|ccc|ccc|ccc|ccc|ccc} 1&0&0&0&0&1&0&1&0&1&0&1&0&1&1&1&1&0&1&0&0 \\ 0&1&0&1&0&0&0&1&1&1&1&1&1&1&0&1&0&0&0&0&1 \\ 0&0&1&0&1&0&1&0&1&0&1&1&1&1&1&1&1&1&1&1&0 \\ -&- \\ 1&0&0&0&1&0&0&1&1&1&0&0&0&0&1&1&0&1&1&1&0 \\ 0&1&0&0&1&1&1&1&0&0&0&1&1&0&0&1&1&1&1&0&0 \\ 0&0&1&1&0&1&1&1&1&1&1&0&0&1&0&0&1&1&1&1&1 \end{array}\right]. \tag{2.33}$$

2.5.4 Adaptive Belief Propagation (ABP) Algorithm

The Adaptive Belief Propagation (ABP) [22, 23] algorithm is an extension of the BP algorithm applied to High Density Parity Check (HDPC) such as RS codes. The algorithm can be applied at both a symbol level and bit-level for RS codes. The results show that the ABP algorithm outperforms other soft decision decoders such as the Koetter and Vardy (KV) algorithm [15]. It is important to note that this research focuses on improving iterative soft decision decoding of RS codes on a bit-level. For this reason, the BP algorithm and its variations will be used to investigate the error correction performance of RS codes.

The bit-level ABP algorithm, a version of the BP algorithm developed for RS codes has the following stages

a) ***Updating of the parity check matrix***

During this stage, the received bit reliabilities undergo sorting from the most reliable to the least reliable bit. These bit reliabilities are expressed as Log Likelihood Ratios (LLR) shown in (2.34) which calculates the probability of the received binary codeword bit cb_i being a 1 or 0. In the equation $L(x_i)$ represents the LLR of the channel for a received codeword y_i. The H matrix is converted to its bit representation using BIE resulting in a matrix with the dimensions $tb \times nb$ where $tb = t \times b$.

$$L(x_i) = log\frac{(cb_i = 1|y_i)}{(cb_i = 0|y_i)}, \tag{2.34}$$

for $1 \leq i \leq nb$. The LLR are fed into the ABP algorithm.

b) ***Perform ABP on every iterations***

For every iteration $\mathcal{T}$, the BP process is carried out until $\mathcal{T}_{max}$ are reached. The $L_{ext}^{(\mathcal{T})}$ represents the transmitted extrinsic information from the LLRs from each bit in the codeword cb_i for the updated parity check matrix $H^{(\mathcal{T})}$. $L^{(\mathcal{T})}$ represents the updated LLRs. The $\acute{i}$ denotes the set of indices in $i = \{nb : H_{tb,nb} = 1\}$ and j runs for $1 \leq j \leq tb$.

$$L_{ext}^{(\mathcal{T})}(cb_i) = \sum_{\substack{i=1 \\ H_{j,i}^{(\mathcal{T})}=1}}^{nb-kb} 2\tanh^{-1}\left(\prod_{\substack{\acute{i}=1 \\ \acute{i}\neq i, H_{j,i}^{(\mathcal{T})}=1}}^{nb} \tanh\left(\frac{L^{(\mathcal{T})}(cb_i)}{2}\right)\right). \tag{2.35}$$

c) ***Hard Decision***

Updating the LLRs values

The LLR values are updated by a factor of α, where $0 < \alpha \leq 1$ is the damping coefficient and $L(cb_i)$ represents the LLRs of the bit .

$$L^{(\mathcal{T}+1)}(cb_i) = L^{(\mathcal{T})}(cb_i) + \alpha L_{ext}^{(\mathcal{T})}(cb_i). \tag{2.36}$$

d) ***Hard Decision***

The value $\hat{cb}_i$ is obtained under the following conditions to determine the

value of cb_i based on its calculated probability. If the conditions are met, the termination criteria is applied if not another iteration is applied.

$$\hat{\mathbf{c}}_\mathbf{i} = \begin{cases} 0 & L(cb_i) > 0 \\ 1 & L(cb_i) \leq 0 \end{cases}. \tag{2.37}$$

e) *Termination Criteria*

The syndrome calculation is as shown below

$$\mathbf{s} = \hat{\mathbf{c}}_\mathbf{i} \times \mathbf{H}^T. \tag{2.38}$$

If the maximum number of pre-set iteration $\mathcal{T}_{max}$ is reached or if $\mathbf{s} = 0$ then the termination criteria is applied based on (2.38).

2.6 Deep Learning

Deep learning is a sub-field of machine learning which is a broader field of artificial intelligence (AI) [46–48]. In deep learning, artificial neural networks, specifically deep neural networks, are employed to model and solve complex problems [49, 50]. Deep learning has also been applied to FEC schemes [25, 43, 51, 52, 52]. Decoding algorithms are essential for reliable communication systems, as they can correct the errors that may occur during transmission. However, traditional decoding algorithms have some limitations, such as high complexity, and sub-optimal performance. In recent years, in order to overcome these challenges, deep learning techniques have been introduced to design and optimize decoding algorithms [26, 53, 54] . By applying deep learning techniques to decoding algorithms, researchers have been able to improve the error correction performance and without further increasing the complexity [18].

2.6.1 Basic Concepts Of Neural Networks

This section will discuss some important basic concepts of neural network principles [55] used in this research work.

- Neural Network (NN) [55] - Neural networks are the building block for deep learning. They are made up of interconnected neurons which are arranged into layers that transform information.
- Multi-Layer Perception (MLP) [56] - The MLP is a type of feed forward artificial neural network which is made up of the fully connected neurons. For instance, the input layer receives the input information, the hidden layer perform certain tasks on the incoming information and final output layer process information from the hidden layers.
- Weight [57] - The weights play a fundamental role as they dictate the strength of connections between neurons across various layers in the neural networks. Through the training phase, these weights undergo adjustments aimed at minimizing prediction errors, thereby enabling the network to learn from data and generate precise predictions.
- Weight Initialization [57] - Weight initialization involves setting the initial model weights. This is essential for model training and convergence.
- Loss Function [57] - The loss function in the neural network measures the performance and the accuracy of the neural network model during the training phase. This research will be utilizing the Binary Cross-Entropy (BCE) denoted as [57]

$$-\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1-y_i) \cdot \log(1-p(y_i)), \tag{2.39}$$

where $p(y_i)$ represents the probability of 1 and $1-p(y_i)$ represent the probability of getting a 0.

- Hyperparameters [58] - These are parameters set before training for fine tuning the training stage for the deep neural network model such as the batch size, learning rate and the number of hidden layers.

- Learning rate [48] - The learning process adjusts the step size during each training epoch to minimize the loss function.

- Batch size - This parameter specifies the number of training samples. The batch size directly influences the stability of the training process.

- ADAM Optimizer [59] - This a type of optimization algorithm utilized for updating the weights.

- Gradient Descent [58] - This is defined as an optimization technique employed to minimize the loss function of the model, thereby adjusting the weights of the model accordingly.

- Xavier/ Glorot Initialization [48] - This method is a form of weight initialization technique, setting initial weights according to a predefined distribution to assist in stabilizing the model's training process.

- Feed-forward Phase [48] - This is defined as the process of feeding data into the neural network model and it propagates forward through the neural network.

- Backpropagation [48] - This is defined as the process of adjusting the model weights of the neural network model by analyzing the error from the training process.

- Activation Function [48] - An activation function is a mathematical function applied to the output of a model, facilitating the learning of complex patterns and aiding in accurate prediction. This research will utilize the sigmoid activation function shown in (2.40) and tanh activation function shown in (2.41)

$$f(x) = \frac{1}{1 + e^{-x}}, \tag{2.40}$$

$$f(x) = \frac{e^{(x)} - e^{(-x)}}{e^{(x)} + e^{(-x)}}. \tag{2.41}$$

2.6.2 Iterative Soft Decision Decoding Using Deep Learning

Deep learning was first applied to linear block codes in the form of an artificial neural network [18, 25, 60]. Nachmani et al [18], first introduced this approach on BCH codes, by applying weights to messages from variable nodes and check nodes, initializing these weights to 1, thereby achieving improved error correction performance without increasing the required computational complexity [18]. Zhang et al [17] extended this approach to RS codes by leveraging the Stochastic Shifting Iterative Decoder (SSID) with a cyclic parity check matrix and variable weights determined by the damping factor, leading to improved error correction performance. The authors in [52] used different forms of neural network structures to investigate how different neural networks improve iterative soft decision decoding.

2.6.3 Architecture Of The Neural Belief Propagation Algorithm

The artificial neural belief propagation algorithm [18] proposed applying weights on the outgoing variable nodes messages. The variable weights are based on the number of non-zero bits in the parity check matrix for $\mathcal{T}$ iterations of the decoding algorithm. This method was applied to BCH codes [18]. This version of the artificial neural belief propagation algorithm is an important foundation for the work done this research. The weight are initialized to 1, therefore the artificial neural belief propagation maintains the same computational complexity [18] as the conventional BP algorithm . The weights are implemented on the variable and check node connection of the Tanner graph. In (2.42) shows the variable node

update [18].

$$x_{i,e} = v, u = tanh\left(\frac{w_i l_v + \sum_{\acute{e}=(v,\acute{u}),u\neq u} w_{i,e,\acute{e}} x_{i-1,\acute{e}}}{2}\right), \tag{2.42}$$

where w represents the weights applied to the variable node. For the messages sent from the variable v to check node u and the edge e connecting the node. For the check node update is computed as follows [18]

$$x_{i,e} = v, u = 2\tanh^{-1}\left(\prod_{\acute{e}=(\acute{v},u),\acute{v}\neq v} x_{i-1,\acute{e}}\right), \tag{2.43}$$

where $x_{i,e}$ represents the check node update for the artificial neural belief propagation algorithm. The output layer computes the final network output after the model is trained. This allows the model to be trained using the sigmoid function.

2.6.4 Graph Neural Networks (GNN)

Graph Neural Networks (GNN) [28, 61] have shown great potential in the representation of various networks such as social networks [62, 63] and computer vision [64]. Leveraging the graph domain, GNNs capture relationships between element inherent in various fields, setting them apart from traditional neural network architectures that process Euclidean data structures [30]. Due to their good performance, GNNs have gained widespread adoption in recent years as a graph analysis model [28, 31, 64]. GNN models have been implemented with modern FEC schemes [33, 34, 65, 66]. They have achieved significant improvements in error detection and correction by representing data as graphs, where each node and its neighbors are taken into account during the training process. An early attempt of the application of GNNs to FEC schemes codes was on BCH codes [33], where a GNN architecture was designed based on iterative message passing decoder. The method integrated the GNN model with a Tanner graph representation [34, 66] of the BP algorithm. The results showed that the GNN based BP

algorithm outperformed the artificial neural belief propagation algorithm in terms of error correction performance and convergence speed. Although the GNN model is relatively more complex, it achieved performance gains over the artificial neural belief propagation model while requiring fewer iterations to decode the received vector.

2.6.4.1 Basic Concept Of Graphs

This section will look at some essential basic concepts of graphs [28] that are vital for this research work .

- Graph [30] - A graph is made up of a group of nodes and a group of edges, where nodes denote the variable node and check nodes while the edge set represents the relationship between the variable node and edge connection.
- Neighborhood [30] - The neighborhood of a specific node is defined as the relationship of that node and other nodes in the graph.
- Node-level tasks [30] - These denote the machine learning tasks related with graph node.
- Edge-level tasks [30] - These denote the machine learning tasks related to the connections between the graph nodes.
- Graph-level task [30] - These refer to the machine learning takes related to the entire graph.
- Graph embedding [30] - This involves representing each node in a graph as a vector, capturing information about the node and the graph structure within this vector representation.
- Feature vector [30] - These refers to characteristics or attributes of the nodes in the graph.

The Table 2.1 below illustrates the basic notation and functions used throughout this dissertation based on [28].

TABLE 2.1: Graph Neural Networks Notations .

Parameter	Notation/Function
A graph	$\mathscr{G}$
Edge-Set	$\mathscr{E}$
Node-Set	$\mathscr{V}$ and $\mathscr{U}$
Function f with a of domain $\mathbb{A}$ and range $\mathbb{B}$	$f : \mathbb{A} \rightarrow \mathbb{B}$
A function of x parametrized by θ	$f(\mathrm{x}; \theta)$
A feature vector F of node x	F_{x}

2.6.4.2 Construction Of The GNN Model

The GNN model is integrated into the message passing BP algorithm. This means that the way the BP algorithm sends messages directly relies on the structure of the graph, and the GNN model is built using information learned from the graph. To combine these two message passing algorithms, the GNN implementation is built on top of the Tanner graph [34, 66]. Fig. 2.5 represents the equivalent GNN representation of the Tanner graph.

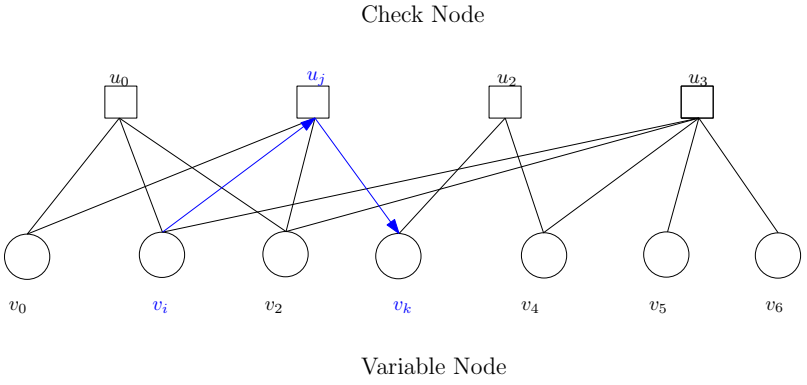


FIGURE 2.5: The Equivalent Graph Representation Of The Tanner Graph

In Fig. 2.5 shows the messages between the check node and variable node denoted in blue based on the GNN structure. Messages are sent between the variable nodes and check nodes. Therefore, the messages based on the Tanner are expressed as follows:

- From Variable Node to Check Node - $m_{v_i} \rightarrow m_{u_j}$
- From Check Node to Variable node - $m_{u_j} \rightarrow m_{v_k}$

This is how the GNN model processes the messages exchanged within the Tanner graph. Afterwards, the graph is constructed taking into account the relationships and connections between edges.

2.6.4.3 Message Passing GNN Based On The Tanner Graph

Based on Fig. 2.5 the message passing in the GNN model based on the Tanner graph representation [33] is expressed as follows:

- Variable Node - For every variable node, denoted as v_i, the group of all variable nodes is represented by $\mathscr{V}$.
- Check Node - For every check node, denoted as u_j, the group of all check nodes is represented by $\mathscr{U}$.
- Variable node neighborhood - $\mathscr{V}(u_j)$ represents the neighborhood of u_j. For example, the group of all the variable nodes v_i connected to check nodes u_j.
- Check node neighborhood - $\mathscr{U}(v_i)$ represents the neighborhood of v_i. For example, the group of all the check nodes u_j connected to the variable nodes v_i.
- Edges/messages - $\mathscr{E}$ represents the edge/messages ($e_{i,j} = (v_i, u_j) \in\rightarrow H_{j,i} = 1$). The group of all edges $e_{i,j} = (v_i, u_j)$ in the graph.
- Node values - Every individual node calculates a F_n dimensional vector, representing the feature vector of the node's dimensionality. Consequently, F_n dimensional vectors are obtained for the nodes $\{h_{v_i}, h_{u_j}\} \in \mathbb{R}^{F_n}$, where each vector corresponds to a variable node v_i and check node u_j. Dimensionality refers to the components in the feature vector that represent the node.
- (Directed) edges values - For every individual directed edge, there is given F_m dimensional value for the message passing from the variable node to the check node, and vice verse as shown in Fig. 2.5. These values are denoted as $\{m_{v_i \rightarrow u_j}, m_{u_j \rightarrow v_i}\} \in \mathbb{R}^{F_m}$. It is important to note that because the messages are computed in different directions, the values are also different.

- Node attributes - These are characteristics given to each node in the graph. These attributes, represented as $\{g_{v_i}, g_{u_j}\} \in \mathbb{R}^{F_{na}}$, are dependent on the direction of the edges connected to the nodes.

- Edge attributes - Each edge in the graph is assigned a specific attribute, denoted as $\{g_{m_{v_i \to u_j}}, g_{m_{u_j \to v_i}}\} \in \mathbb{R}^{F_{ma}}$. These attributes are unique to each edge and vary depending on the direction of the edge. In total $2|\mathscr{E}|$ attribute exist.

2.7 Differences Between GNN And ANN Structure

This section presents a comparison between Artificial Neural Networks (ANNs) and Graph Neural Networks (GNNs) in machine learning. ANNs [59] are adept at processing structured data like images and sequences, while GNNs specialize in analyzing graph-structured data. ANNs utilize feedforward and recurrent structures with gradient-based optimization. In contrast, GNNs operate on graph-based architectures, employing message passing algorithms [64]. Understanding these differences aids in selecting suitable techniques for various machine learning tasks.

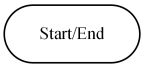
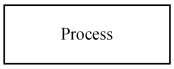
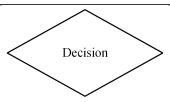
2.8 Flow-Chart Notation

Flow charts [69] are utilized in this dissertation to represent algorithms and data flow. The notation used in this research is summarized in Table 2.2.

2.9 Summary

This chapter introduces the fundamental concepts essential for this research. The main aim of this chapter is to explain the core concepts behind FEC, RS codes, iterative soft decision decoding, and the integration of deep learning into iterative

Table 2.2: Flow-Chart Notation

Type	Example	Description
Start/End		Represents the start/end of an algortihm
Process		Describes a single operation specified in the flowchart
Decision		A boolean decision branch

soft decision decoding. Additionally, this chapter reviews related work on improving iterative soft decision decoding of RS codes. It establishes the need for further performance improvements using modern deep learning techniques.

CHAPTER 3

Research Methodology

3.1 Introduction

In this chapter, the methodology employed in this research to improve the performance of iterative soft decision decoding of RS codes. The approach involves applying Artificial Neural Networks (ANN) and Graph Neural Network (GNN) models. A step-by-step implementation of the ANN and GNN models is outlined, detailing the strategies and techniques that are used in the development of the models for the research work.

The research methodology chapter is arranged as follows:

- **Neural Belief Propagation (NBP) Decoder** - This section will look at, the ANN model implementation including the model architecture, model training and model evaluation used in this research. We will refer to this implementation as the proposed Neural Belief Propagation (NBP) decoder.
- **GNN Based BP Decoder** - This section will look at the GNN model implementation, including the development of the graph structure for the BP decoder architecture.

3.2 Proposed NBP Decoder

3.2.1 System Design

The proposed model is based on a similar model in [18]. Fig. 3.1 illustrates the proposed system model for the proposed NBP algorithm.

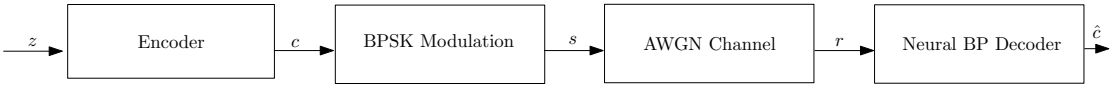


FIGURE 3.1: System Model Of The Proposed NBP Decoder

At the transmitter, inputs z are RS messages. The output from the encoder c is then mapped using BPSK modulation. The BPSK symbols s are transmitted through a AWGN channel. At the output of the AWGN channel, r is received. The log-likelihood ratios are then calculated from r and fed into the proposed NBP decoder similar to [18] which outputs the estimated codeword $\hat{c}$. Each process is described as follows

Encoding And Modulation

The encoding process is executed in two ways :

- All-Zero Codewords - This process involves using a batch of zero codewords of length nb which are fed in the NBP model. This method is explored for a fair comparison with the work in [18].
- RS Encoded Codewords - A batch of random codewords are generated using the RS encoder provided by the Galois library [70]. These codewords are converted to bit-level as discussed in Section 2.6.1. This method is used for comparison with the GNNBP model, which will be discussed in a later section.

The codewords are than modulated using BPSK modulation scheme. Due to computational constraints, the maximum length is set to $n = 31$. Therefore, the maximum bit-level length is $nb = 155$.

Channel Simulation

Channel simulation is achieved using the AWGN Channel . The channel's SNR is measured as $\frac{E_b}{N_0}$ and is varied in 1 dB increments from 1dB to 7 dB.

Generating The LLR Matrix

The received signal r at each SNR are demodulated. This produces the LLRs vector for each codeword in the batch as required by the the NBP decoder.

3.2.2 NBP Model Architecture

As denoted by Nachmani et al [18] the NBP utilizes additional weights on the variable node and check node connection in order to scale the outgoing messages. The proposed neural network takes a similar approach. However, the proposed NBP decoder implements a scalar damping factor α in addition to the variable weights to optimize the weighted messages. Fig. 3.2 shows how the weights are applied to variable and check node connections in the proposed neural network model.

3.2.2.1 Model Training Phase

In order, to construct a NBP model that can predict the transmitted codeword, it is essential to train the model to perform optimally. This section discusses the main components of the model training process such as the loss function, batch size, number of training iterations and optimization method. The SNR values of training sample are an essential part of the model training phase. This is because it affects the performance of the model as it regulates the amount of noise added to the model. Depending on the encoding process utilized the training phase is described as follows:

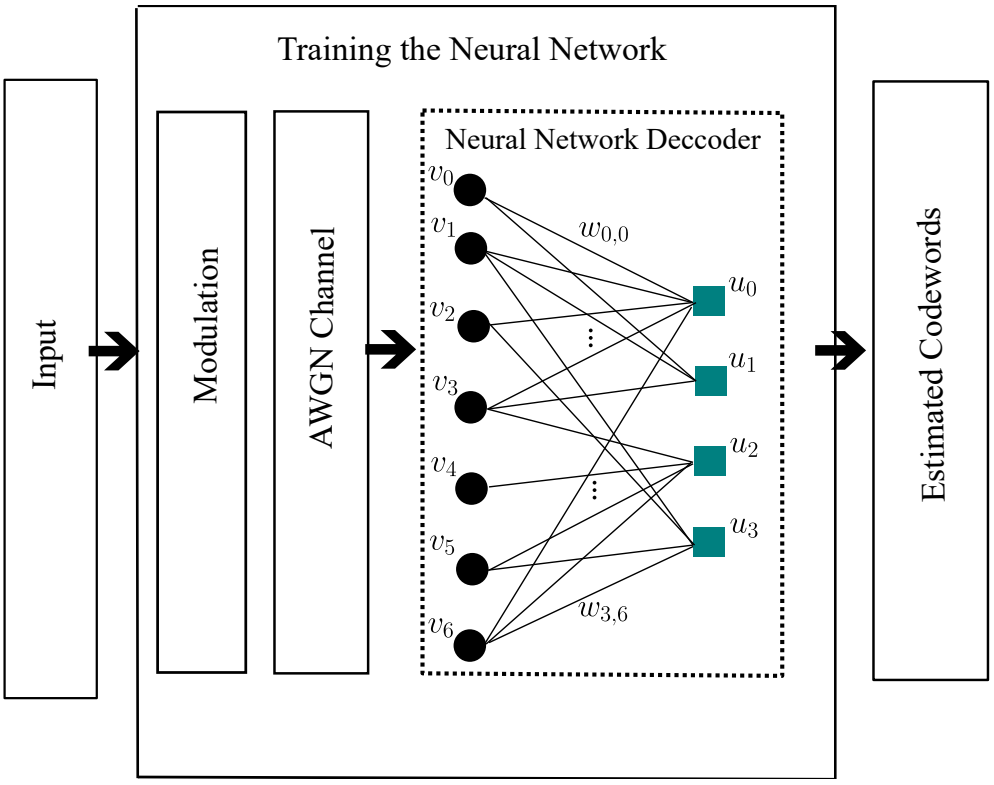


FIGURE 3.2: NBP Model Architecture

- **All-zero codewords** - For the all-zero codewords, the training is established on [18, 71]. A fixed SNR value is used to train the model. The selected model components for training the proposed NBP algorithm are shown in Table 3.1.

- **RS encoded codewords** - For the RS encoded codewords, the training is based on [71]. This specific version of the proposed NBP algorithm is employed to assess the iterative decoding performance of the GNNBP model. Since the GNNBP algorithm does not function well with all-zero codewords [33], it is important for a fair comparison to implement the encoding process for the proposed NBP algorithm. Furthermore, since GNNBP requires extensive training, similar training is applied to the proposed NBP model. This step ensures a fair comparison of error correction performance.

Table 3.1 and Table 3.2 show the training parameters for the proposed NBP algorithm for all zero codewords and RS encoded codewords based on the work done in [18, 33, 34, 71].

Table 3.1: NBP Model All-Zero Codewords Hyperparameters.

Parameter	Hyperparameters
SNR Value	4
Activation Function	tanh
Number Of Iterations	20
Learning Rate	10^{-2}
Batch Size	1000
Batch Size Validation Set	1000
Train iteration	200
LLR clipping	10
SNR Validation	-
Optimization Method	ADAM optimizer
LLR clipping	10

Table 3.2: NBP Model RS Encoded Codewords Hyperparameters Used For Comparison With The GNNBP algorithm.

Parameter	Batch 1	Batch 2	Batch 3
SNR value	5	5	6
Activation Function	tanh		
Number of iterations	20	20	20
Learning Rate	10^{-2}	10^{3}	10^{-3}
Batch Size	2000	2000	2000
Batch Size Validation Set	2000		
Train iteration	300	1000	1000
SNR Validation	7		
Optimization Method	ADAM optimizer		
LLR clipping	10		

3.2.2.2 Model Validation Phase

This section discusses the model validation process for the proposed NBP algorithm. A validation dataset is used to provide an unbiased evaluation of the model's performance on the training dataset while tuning its hyperparameters. The model validation phase utilizes data from the training phase to ensure fine-tuning of the model. Depending on the encoding process utilized the model validation phase is described as follows:

- **All-zero codewords** - The validation process for all-zero codewords is based

on the approach established in [18, 71]. In this process, validation is carried out every 10 training iterations. After each set of iterations, a new batch of 1000 samples with random SNR values is generated, and the model predicts the codewords for this batch. The BCE is then calculated. This validation approach is adjusted to account for the transmission of all-zero codewords. The model components selected for validating the proposed NBP algorithm are shown in Table 3.1.

- **RS encoded codewords** - For the RS encoded codewords, the validation process is based on [71]. As mentioned in Section 3.2.2.1, this version of the NBP algorithm is utilized to evaluate performance of the GNNBP algorithm. We utilize a dataset comprising 2000 samples. The performance of the model is evaluated for a specific SNR, as detailed in Table 3.2.

During the validation phase, data from the training process is used to fine-tune the model, making adjustments to improve its training process. This phase is especially important for optimizing the hyperparameters of the NBP algorithm.

3.2.2.3 Model Testing Phase

This section discusses the model testing, for the proposed NBP algorithm. Detailing how to evaluate model performance using the testing datasets. The dataset is used to provide an unbiased evaluation of model after training and validation phases are completed. The performance of the trained NBP model is evaluated based on the following:

- Testing set - The trained model is evaluated by using a batch size of 10000 codewords.
- Range of SNR values - As mentioned in Section 3.2.4 the SNR affects the performance of the NBP model. Therefore, unlike the model training stage which used a fixed SNR, a range of unseen SNR values are used to test model

how the model performs under different noisy conditions. The SNR range used in the testing stage is from 1dB to 7 dB [18].

Following the simulation, the Bit Error Rate (BER) is measured and plotted against the range of SNR values. The performance of the proposed NBP model is compared to the BP decoder in order to evaluate performance of the proposed decoder.

3.2.3 GNNBP Algorithm Overview

As described in Section 2.6.4.2, the GNN integrates with the Tanner graph to build a model for the BP algorithm. A GNN structure for BCH codes and LDPC codes is proposed in [33]. A summary of the steps are presented below:

1. Log-likehood ratio (LLR) are generated from the received vector. These LLRs initialize the graph nodes and features which will be used to interpret information about the graph.
2. Messages m from the variable nodes to check nodes are exchanged in the graph in both directions.
3. Messages m from the variable nodes and check nodes are updated accordingly.
4. These steps are repeated iteratively until iteration $\mathcal{T}_{max}$ is reached.
5. Hard decision vector of the updated LLRs is computed at the last layer.

The proposed model is based on similar models in [33, 34]. Fig. 3.3 illustrates the proposed system model for the GNNBP decoder.

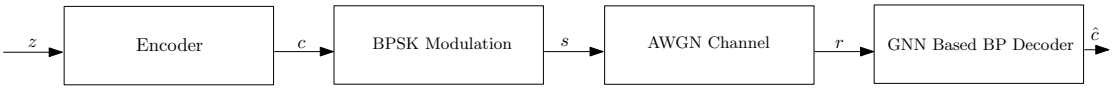


FIGURE 3.3: System Model For The GNNBP decoder

At the transmitter, inputs z are RS messages. The output from the encoder c in then mapped using BPSK modulation. The BPSK symbols s are transmitted through a AWGN channel. At the output of the AWGN channel, r is received. The log-likelihood ratios are then calculated from r and fed in to the GNNBP decoder similar to [33] which outputs the estimated codeword $\hat{c}$.

3.2.3.1 Generation Of Codewords

Due to computational complexity constraints of the GNN, the maximum codeword lengths that are utilized in the research is limited $n = 15$. Therefore the simulation is run with a $GF(16)$ RS codes and the bit-level length is $nb = 60$ respectively.

3.3 Channel Model

This section describes the simulation setup for evaluating the performance of different RS codes in an Additive White Gaussian Noise (AWGN) channel [36]. The channel model is an important aspect of this research, as it determines the noise level and the error probability of the transmitted codewords. The channel capacity C [1] of the AWGN channel is defined as

$$C = \frac{1}{2} \log_2 \left(1 + \frac{E_b}{N_0} \right), \tag{3.1}$$

where E_b is defined as the energy per bit, and N_0 represents the noise spectral density. This is defined as the signal-to-noise ratio (SNR).

In order to evaluate the performance of a communication system at a given SNR level, the transmitted signal or modulated signal is corrupted by additive noise with a certain power. The noise power determines the SNR value in such simulations. In real scenarios, SNR is usually expressed in decibels (dB). The received signal r is expressed as

$$r = s + \mathcal{N}, \tag{3.2}$$

where the transmitted signal is denoted as s and $\mathcal{N}$ represents a white Gaussian noise which is the noise added by the AWGN channel.

3.3.1 Generating Bit Probabilities And Constructing The LLR Matrix Through Binary Phase Shift Keying (BPSK) Modulation

This section describes the approach used to calculate the log-likelihood ratio (LLR) of codewords transmitted using Binary Phase Shift Keying (BPSK) modulation. The work done in this research is on a bit level. Therefore, for bit-level RS codes $nb = n \times b$ where b is the order of field $GF(2^b)$ and n is length of a symbol RS codeword. Let cb be a bit-level RS codeword of length nb.

$$cb = [c_1, c_2, \ldots, c_{nb}]. \tag{3.3}$$

The transmitted signal s is modulated by mapping the codeword cb onto BPSK signal constellation. The BPSK modulation scheme is as follows for $1 < j < nb$, if cb_j=1 it is mapped onto $s_j = 1$ and if the $cb_j = 0$ it is mapped onto $s_j = -1$. The received vector r is obtained after transmitting s (the modulated signal) through an AWGN channel. The received vector r is represented as

$$r = [r_1, r_2, \ldots, r_{nb}]. \tag{3.4}$$

The channel posterior probability that the bit cb_j, given the information in the received signal r_j, can be obtained by rewriting the likelihood function as follows

$$P(cb_j = 1|r_j) = \frac{1}{1 + exp^{-\frac{2r_j}{\sigma^2}}}, \tag{3.5}$$

assuming $P(cb_j = 1) = P(cb_j = 0) = \frac{1}{2}$ the channel posterior probability for the bit $cb_j = 0$ [37, 67], given the information in r_j, can be computed as.

$$P(cb_j = 0|r_j) = 1 - P(cb_j = 1|r_j). \tag{3.6}$$

The channel posterior probabilities can be utilized to compute the Log-Likelihood Ratios. The equation is defined as follows

$$L(cb_j) = \log \frac{P(cb_j = 1|r_j)}{P(cb_j = 0|r_j)}. \tag{3.7}$$

Based on the equations in (3.12), (3.13), we can rewrite the LLR values obtained from the calculation as follows

$$L(cb_j) = \frac{2r_j}{\sigma^2}, \tag{3.8}$$

where σ^2 is defined as the noise variance.

3.4 Simulation Model

A comparative study is conducted to analyze the performance of the decoders under different conditions. The system model shown in Fig. 3.4 will be used for the proposed simulations.

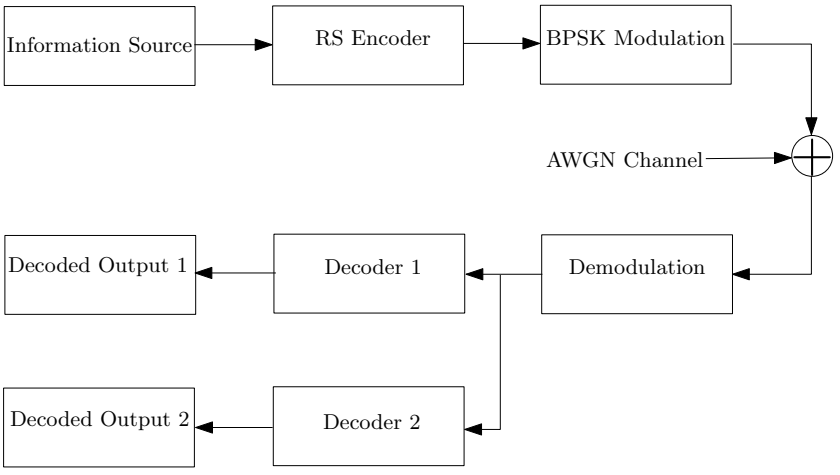


FIGURE 3.4: Simulation Model

The RS codeword is generated from the RS encoder. The codeword is then broken down to its bit representation and transmitted over an AWGN channel using a BPSK modulation scheme. The received vector is then demodulated at the output of the channel and is used to create the LLR vector. The constructed LLR vector is fed into both decoder 1 and decoder 2. The error correction performance is analyzed and compared at the output through the calculation of the Bit Error Rate (BER).

3.5 Simulation Setup

The error correction performance of the simulations in this research are measured in terms of Bit Error Rate(BER) and are tested across a range of different signal-to-noise (SNR) ratios. The simulation model involves generating 10000 RS codewords of length 15 symbols on a bit-level. These codewords are transmitted through an AWGN channel and are modulated with a BPSK modulation scheme [68]. The received signal is decoded at the output of the channel using hard decision detection based on the maximum likelihood. The simulation aims to compare the outcome with the BPSK bit error probability [68]. The BPSK bit error probability equation is expressed as follows

$$\text{Bit Error Probability} = \frac{1}{2} erfc\left(\sqrt{\frac{E_b}{N_0}}\right). \tag{3.9}$$

The BER vs SNR graph obtained from the simulation aligns with the bit error probability. Based on the simulation setup mentioned above the results are shown in Fig. 3.5.

The results confirm that the simulation setup developed for the research is reliable and suitable for running simulations for this research work.

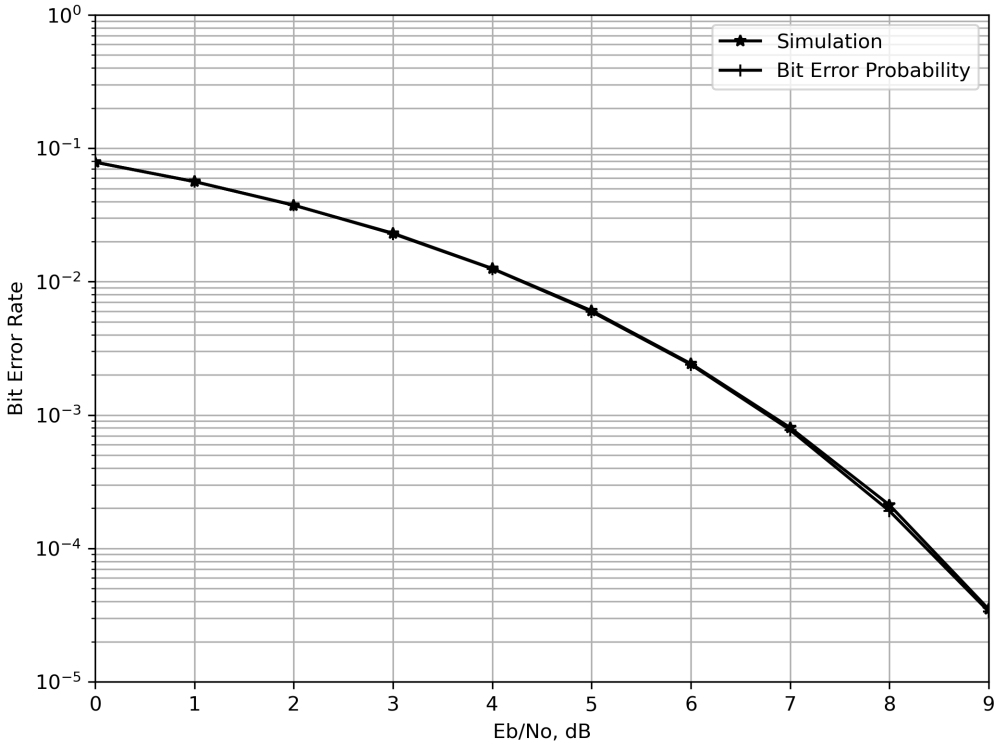


FIGURE 3.5: Uncoded Simulation

3.6 Software And Libraries

This research work utilizes the Python library [73] throughout the dissertation because of the versatility of the libraries such as Pandas, TensorFlow [74], Numpy and Galois library [70] . The Galois library plays a crucial role in generating random codewords in the Galois field, the encoding process and parity check matrices used in this research. For the development and implementation of the different deep learning models, the TensorFlow library and Sionna library [71] is utilized.

3.7 Summary

In this chapter, the proposed NBP model and GNNBP model are presented. Specifically, the model structure and model architecture for the NBP model is

discussed in the chapter. The main contribution is detailing the model training, model evaluation for NBP implementation.

CHAPTER 4

Techniques To Improve Iterative Soft Decision Decoding Of Reed Solomon Codes

4.1 Introduction

This chapter focuses on implementing improvements to iterative soft decision decoding of RS codes. The primary contribution of the chapter is in adding sparsity to parity check matrix, thereby improving exchange of soft information and consequently leading to good error correction performance. This approach is tested with both the Neural Belief Propagation (NBP) algorithm and Belief Propagation (BP) algorithm for high-rate Reed Solomon (RS) codes.

4.2 Proposed Sparsity Implementation On The Reed Solomon Parity Check Matrix

This section will delve into the proposed method for adding sparsity to the RS parity check matrix. After that, the results of the proposed implementation on the NBP and BP algorithms are presented. Thereafter, the simulation results for the comparative study between the normal RS parity check matrix and the proposed sparse parity check matrix are presented.

The following abbreviations will be used in this chapter:

- SPCM - Sparse parity check matrix.
- NPCM - Normal parity check matrix.

- NBP - Neural belief Propagation.

4.2.1 Problems With Iterative Decoding On High Density Parity Check (HDPC) Codes

The sparse nature of the LDPC codes has shown to improve error correction performance with the BP algorithm. Hence sparsity is essential for improved decoding performance. However High Density Parity Check (HDPC) codes struggle due to the dense parity check matrix. This is because the HDPC codes which can be represented as a Tanner graph as illustrated in Section 2.5.1. HDPC codes suffer from an occurrence of small cycles in the Tanner graph. These cycles led to messages getting stuck in pseudo equilibrium points which degrade decoding performance.

Different approaches have been explored to tackle the challenge of HDPC matrices, these approaches include the following works [21, 23, 75, 76]. A few of these techniques used to enable this implementation include Binary Image Expansion (BIE) for RS codes. In this section, we propose a method to add sparsity to the parity check matrix in order to improve error correction performance of different iterative soft decision decoders.

4.2.2 Proposed Sparse Parity Check Matrix Implementation

The proposed technique to add sparsity to the parity check matrix takes advantage of the cyclical properties of RS codes discussed in Section 2.4.2. The approach involves the addition of low weight parity check equations to the H matrix that match the minimum distance of RS codes. Thereafter, cyclically shifting the rows of the H matrix one place to the right n times. Similar to the approach in Section 2.4.2.

Given a (15,7) RS code, the corresponding coefficients of the $h(x)$ are presented based on the parity check polynomial equation presented in Section 2.3

$$h(x) = x^7 + 9x^6 + 9x^5 + 12x^8 + 8x^3 + 6x^2 + 9x + 10. \tag{4.1}$$

The coefficients derived from the $h(x)$ are presented as

$$h_{coefficient} = [1, 9, 9, 12, 8, 6, 9, 10]. \tag{4.2}$$

.

The coefficients are cyclically shifted as follows based on (2.22) in Section 2.3.2 to give

$$H = \begin{bmatrix} 10 & 9 & 6 & 8 & 12 & 9 & 9 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 9 & 6 & 8 & 12 & 9 & 9 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 9 & 6 & 8 & 12 & 9 & 9 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 9 & 6 & 8 & 12 & 9 & 9 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 9 & 6 & 8 & 12 & 9 & 9 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 & 9 & 6 & 8 & 12 & 9 & 9 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10 & 9 & 6 & 8 & 12 & 9 & 9 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 9 & 6 & 8 & 12 & 9 & 9 & 1 \end{bmatrix} \tag{4.3}$$

The proposed sparse implementation further cyclically shifts the rows of the H matrix one place to the right n times to give

$$H = \begin{bmatrix} h_k & h_{(k-1)} & h_{(k-2)} & \dots & h_0 & 0 & 0 & \dots & 0 \\ 0 & h_k & h_{(k-1)} & h_{(k-2)} & \dots & h_0 & 0 & \dots & 0 \\ 0 & 0 & h_k & h_{(k-1)} & h_{(k-2)} & \dots & h_0 & \dots & 0 \\ \vdots & & & & \vdots & & & & \vdots \\ 0 & 0 & \dots & 0 & h_k & h_{(k-1)} & h_{(k-2)} & \dots & h_0 \\ \vdots & & & & \vdots & & & & \vdots \\ h_k & h_{(k-1)} & \dots & h_0 & 0 & 0 & \dots & 0 & h_k \end{bmatrix}. \tag{4.4}$$

Based on (4.4) the sparse parity check matrix is given as

$$H = \begin{bmatrix}
10 & 9 & 6 & 8 & 12 & 9 & 9 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 10 & 9 & 6 & 8 & 12 & 9 & 9 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 10 & 9 & 6 & 8 & 12 & 9 & 9 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 10 & 9 & 6 & 8 & 12 & 9 & 9 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 10 & 9 & 6 & 8 & 12 & 9 & 9 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 10 & 9 & 6 & 8 & 12 & 9 & 9 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 10 & 9 & 6 & 8 & 12 & 9 & 9 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 9 & 6 & 8 & 12 & 9 & 9 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 9 & 6 & 8 & 12 & 9 & 9 \\
9 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 9 & 6 & 8 & 12 & 9 \\
9 & 9 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 9 & 6 & 8 & 12 \\
12 & 9 & 9 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 9 & 6 & 8 \\
8 & 12 & 9 & 9 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 9 & 6 \\
6 & 8 & 12 & 9 & 9 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 9 \\
9 & 6 & 8 & 12 & 9 & 9 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10
\end{bmatrix} \tag{4.5}$$

The next step in the proposed implementation is to transform the H matrix into a bit-level representation using BIE. The application of BIE to the parity check matrix further increases the sparsity of the H matrix.

This proposed implementation is compared with a parity check matrix represented in the Vandermonde matrix form as shown in (2.25) under Section 2.4. 2. In this work, the matrix is referred to as the Normal Parity Check Matrix (NPCM). This is chosen based work done in [21, 23]. Similar to the proposed sparse implementation, the matrix undergoes BIE to increase the sparsity as the matrix is relatively dense.

4.3 Performance Of The Belief Propagation Based On Sparsity Implementation

4.3.1 Bit Error Rate Performance Analysis

The simulation results presented in this section investigate the sparse implementation with the BP algorithm for high-rate RS codes. RS codes are transmitted through an AWGN channel using a BPSK modulation scheme. The received vector is demodulated and fed into the BP algorithm. Then the results are compared to evaluate how adding sparsity affects the decoding performance.

4.3.1.1 SPCM Implemented With BP Algorithm vs NPCM Implemented With BP Algorithm For A (15,11) RS Code

The results of the simulations comparing the error correction performance of the BP algorithm with the proposed sparse parity check matrix are presented in this section. The BP decoder is executed by using a batch of encoded codewords for a (15,11) RS code.

From Fig. 4.1, it can be seen that the SPCM implemented with the BP decoder is able to yield a performance gain compared to the NPCM implementation. At a BER value of 10^{-4} the SPCM yields a gain of about 0.61 dB. The outcome of the simulation supports the hypothesis in Section 4.2.1 that, adding sparsity to the RS parity check matrix results in improved exchange of soft information which leads to improved error correction performance.

4.3.1.2 SPCM Implemented With BP Algorithm Vs NPCM Implemented With BP Algorithm For A (31,27) RS Code

The proposed technique is tested with the (31,27) RS code to investigate the error correction performance for a higher code rates. Fig. 4.2 shows the error correction

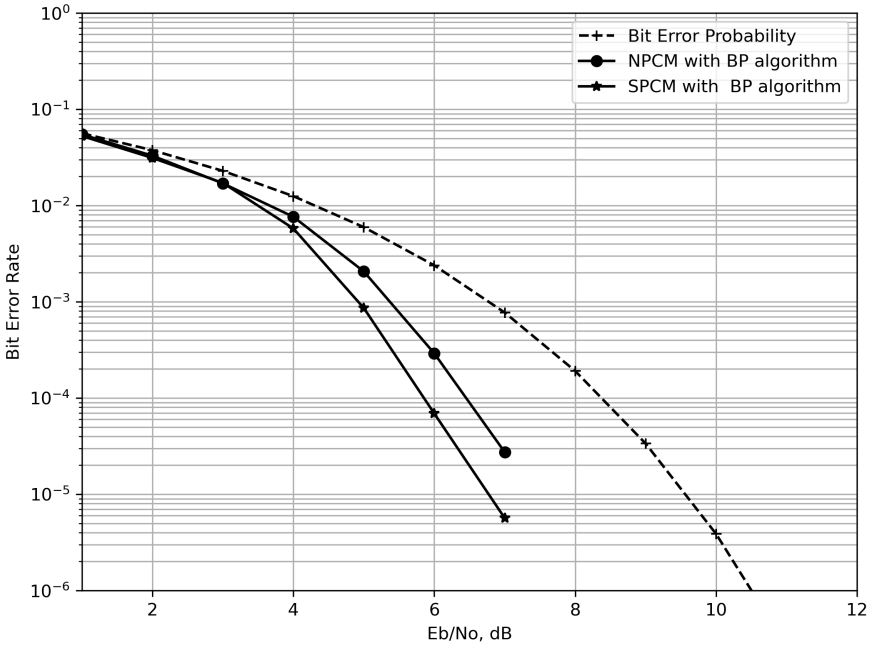


FIGURE 4.1: Performance Comparison Of The SPCM Implemented With BP Vs The NPCM Implemented With BP For A (15,11) RS Encoded Codewords

performance of the (31,27) RS when the proposed technique is applied for encoded codewords. Fig. 4.2 shows the (31,27) RS code BP results with random codewords.

From Fig. 4.2, it can be observed that there is a slight gain when the sparse parity check matrix implemented for the BP algorithm. Similar to the simulation for the (15,11) RS, a batch size of random RS codewords are transmitted through the channel and the received vector is decoded by the BP decoder. At BER 10^{-4}, the SPCM implemented with the BP algorithm achieves a gain of 0.19 dB over the NPCM implemented with the BP algorithm. Working with longer high-rate codes, the density of the parity check matrix often increases due to the higher minimum weight value. A higher density matrix limits the ability to efficiently exchange soft information during iterative decoding.

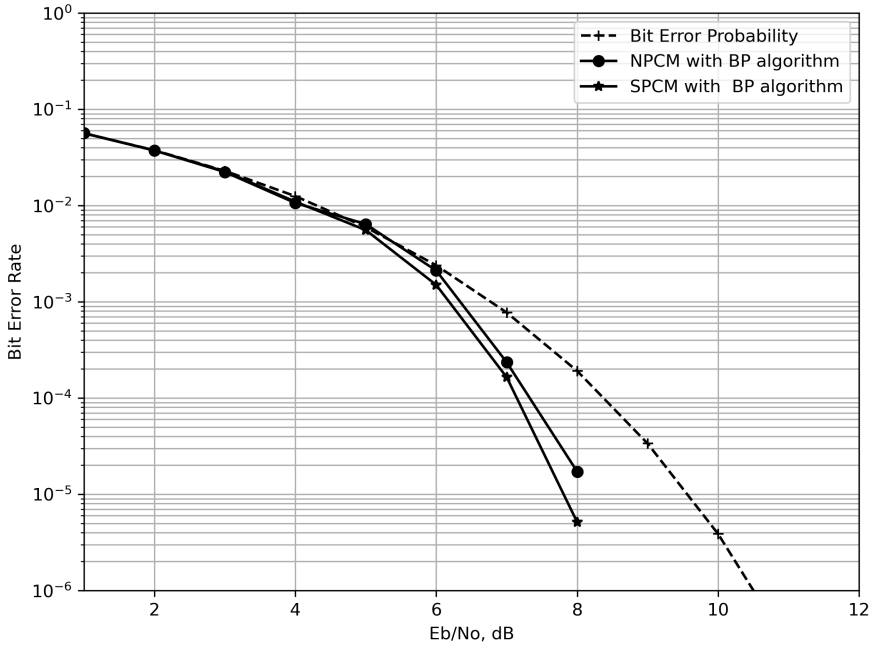


FIGURE 4.2: Performance Comparison Of The SPCM Implemented With BP Vs The NPCM Implemented With BP For (31,27) RS Encoded Codewords

4.4 Performance Of The Neural Belief Propagation Based On Sparsity Implementation

This section presents the performance analysis of the NBP algorithm for RS codes. In this implementation, the NBP algorithm is adapted to work with RS codes by incorporating a scalar damping factor and trainable weights based on the edge connections in the parity check matrix, similar to [18]. The proposed implementation employs a scalar damping factor of $\alpha = 0.5$ to improve error correction performance without increasing the computational complexity similar to [18]. During the training process, the weights are adjusted to improve error correction performance compared to the BP decoder. The NBP decoder is trained using the process defined in Section 3.2.2.1, which is summarized in Fig. 4.3.

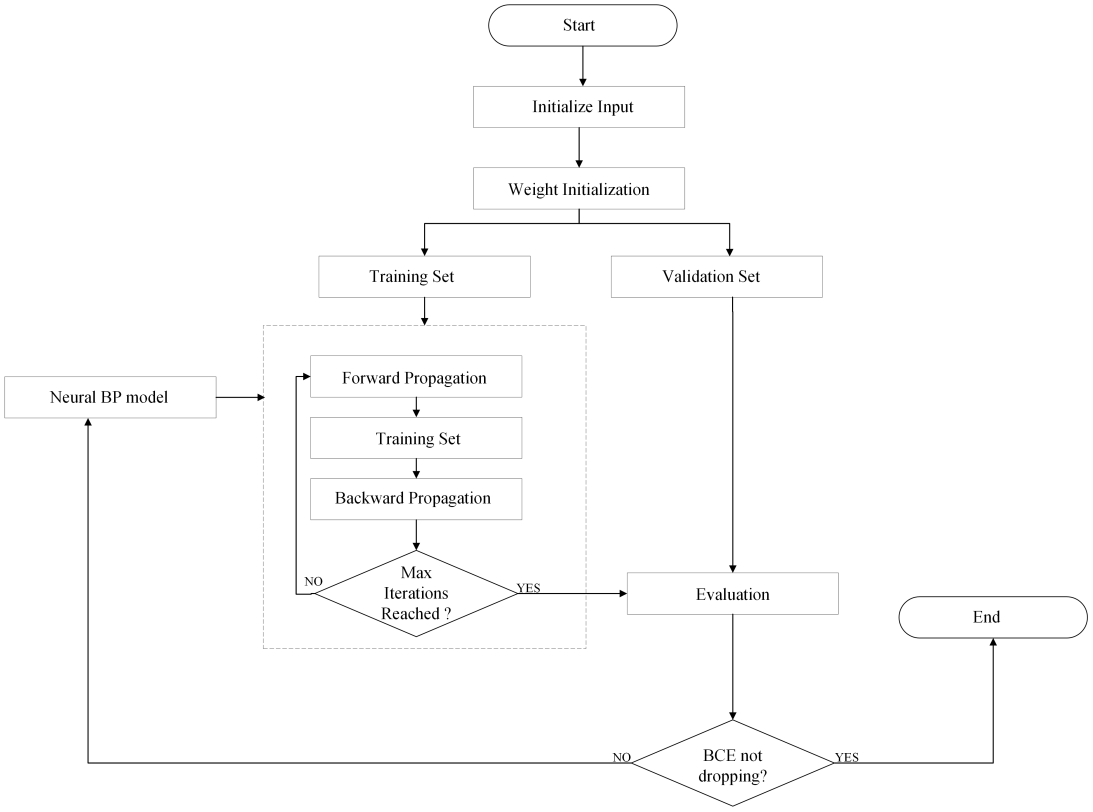


FIGURE 4.3: NBP Decoder Training Process

4.4.1 Bit Error Rate Performance Analysis

In this section, experiments are tested on the proposed NBP algorithm to investigate how the structure of the parity check matrix affects error correction performance of the model. The results are measured in terms of the BER of the NBP algorithm. The results presented are based on the all-zero codeword method discussed in Section 3.2.1 for a similar to the work done in [18].

4.4.1.1 NBP Algorithm Vs BP Algorithm For A (15,11) RS Code

The simulation results presented in this section investigate the sparse implementation with the NBP algorithm for high-rate RS codes. Therefore, simulation sets for running with the SPCM implementation and without the SPCM implementation are discussed. The initial results compare the performance of NBP algorithm and

BP algorithm. These tests are conducted on high-rate (15,11) RS codes. Fig. 4.4 illustrates the performance of the NBP algorithm.

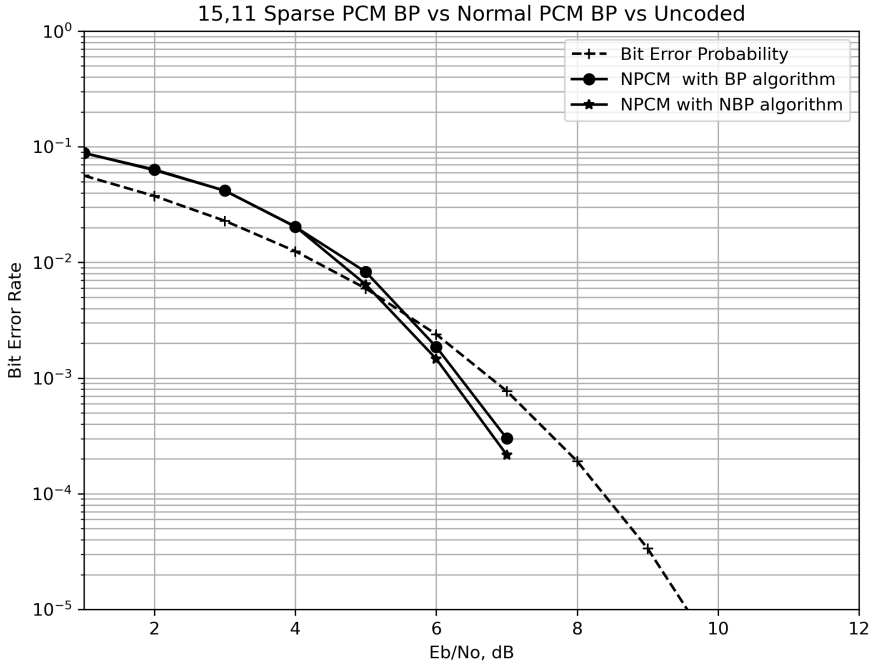


FIGURE 4.4: Performance Comparison Of The NBP Vs BP For A (15,11) RS Code

As shown in Fig. 4.4, the NBP algorithm yields a gain in terms of error correction performance when compared to the BP algorithm for a (15,11) RS code. At a BER of 10^{-3} the NBP exhibits a gain of 0.13 dB over the BP algorithm. This is due to the implementation of weights that adjust during training resulting in improve error correction performance. The performance gain is seen at higher SNR values compared to lower SNR values.

In order to further improve on the error correction performance the proposed SPCM approach is implemented with the NBP algorithm. The results are shown in Fig. 4.5.

As seen in Fig. 4.5, the proposed SPCM approach led to an improvement in the error correction performance of the NBP algorithm when compared to the BP algorithm. At a BER of 10^{-3} NBP algorithm when the proposed SPCM is utilized yields a gain of 0.38 dB. The error correction performance gains for the NBP are

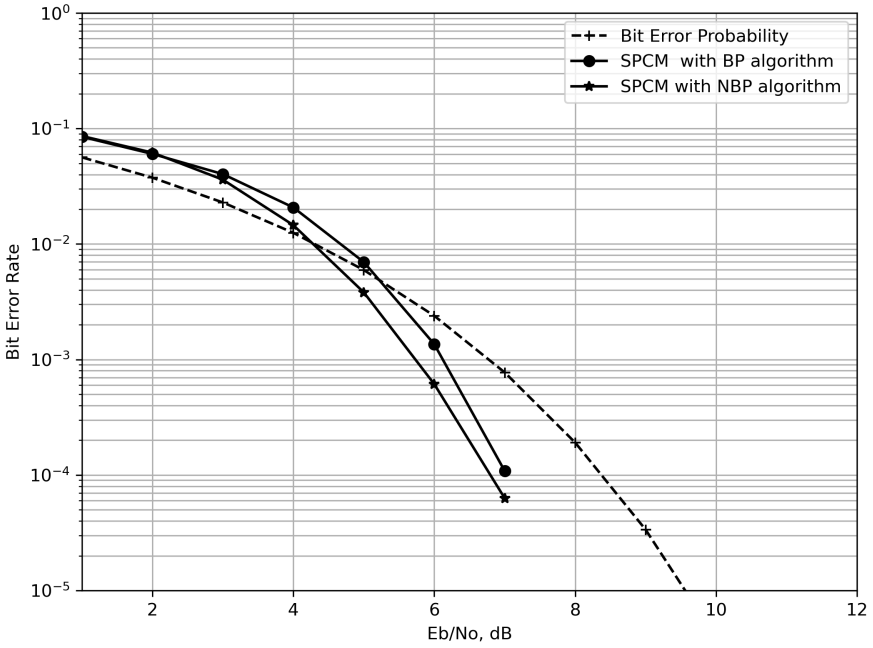


FIGURE 4.5: Performance Comparison Of The SPCM Implemented With NBP Algorithm Vs SPCM Implemented With BP Algorithm For A (15,11) RS Code

more apparent with the SPCM due to the improved exchange of soft information during the iterative decoding process enabled by the sparsity in the matrix.

An additional simulation is carried out to test the performance of the SPCM implemented with the NBP and the NPCM implemented with NBP in order to highlight the error correction gains of using the SPCM implementation. The results for running this simulation with a (15,11) RS code are shown in Fig. 4.6.

From Fig. 4.6 it can be seen that the error performance of the SPCM implemented with the NBP outperforms the NPCM implemented with NBP. At a BER of 10^{-3} the SPCM yields a gain of 0.46 dB over the NPCM implemented with the NBP. This further confirms the hypothesis presented in Section 4.2.1 that utilizing a SPCM leads to improved iterative decoding performance for RS codes with the NBP algorithm.

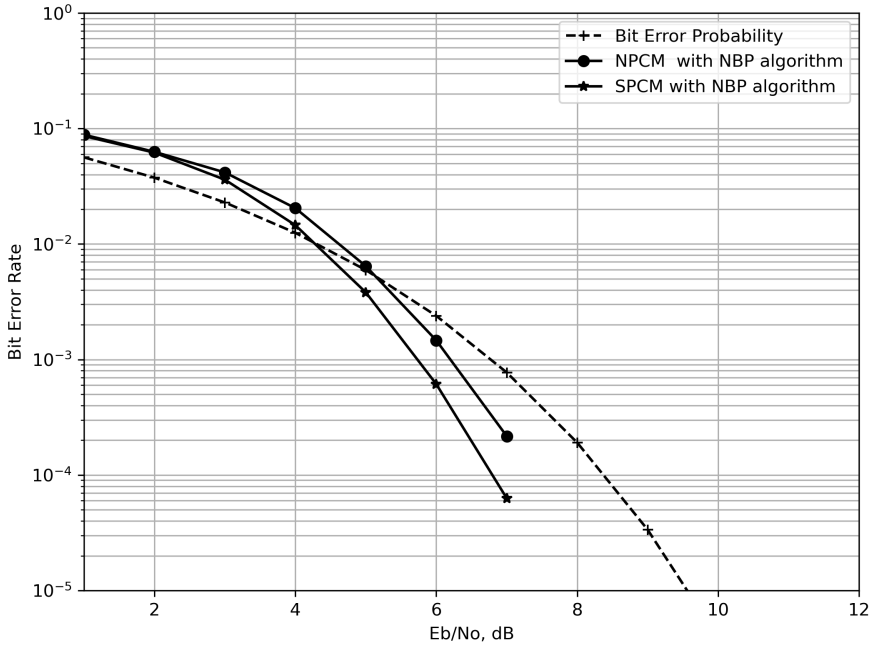


FIGURE 4.6: Performance Comparison Of The SPCM Implemented With NBP Algorithm Vs NPCM Implemented With NBP Algorithm For A (15,11) RS Code

4.4.1.2 NBP Algorithm Vs BP Algorithm For A (31,27) RS Code

The simulation results presented in this section investigate the sparse implementation with the NBP algorithm for high-rate RS codes. Therefore, simulation sets for running with the SPCM implementation and without the SPCM implementation are discussed. The initial results compare the performance of NBP algorithm and BP algorithm. These tests are conducted on high-rate (31,27) RS codes. Fig. 4.7 illustrates the performance of the NBP algorithm.

As shown in Fig. 4.7, the NBP algorithm performs well compared to the BP algorithm for a (31,27) RS code. At a BER of 10^{-3} the NBP has a slight gain of 0.05 dB over the BP algorithm. This is due the relatively dense structure of the high-rate (31,27) RS codes. Therefore limits the decoders ability to efficiently exchange soft information during the decoding process.

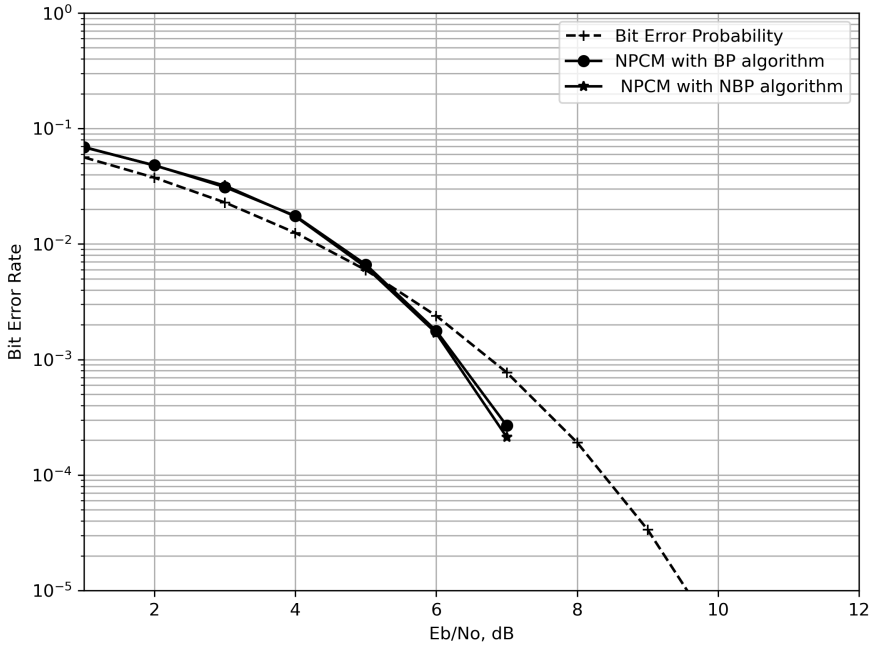


FIGURE 4.7: Performance Comparison Of The Neural BP Vs BP For A (31,27) RS Code

In order to further improve on error correction performance, the proposed SPCM is applied to the NBP algorithm. The results for this approach are shown in Fig. 4.8.

As seen in Fig. 4.8, the proposed SPCM led to an improved error correction performance of the the NBP algorithm when compared to the BP algorithm. At a BER of 10^{-3} SPCM implemented with NBP achieves a gain of 0.10 dB over the SPCM implemented with BP algorithm. The performance is attributed to the use of the SPCM which improves exchange of soft information. This leads to a gain in error correction performance for the (31,27) RS code when working with the SPCM.

An additional simulation is carried out to test the performance of the SPCM implemented with the NBP and the NPCM implemented with NBP in order to highlight the error correction gains of using the SPCM implementation. The results for running this simulation with a (31,27) RS code are shown in Fig. 4.9.

From Fig. 4.9, it can be seen that the error correction performance of the SPCM implemented with the NBP outperforms the NPCM implemented with NBP. At a

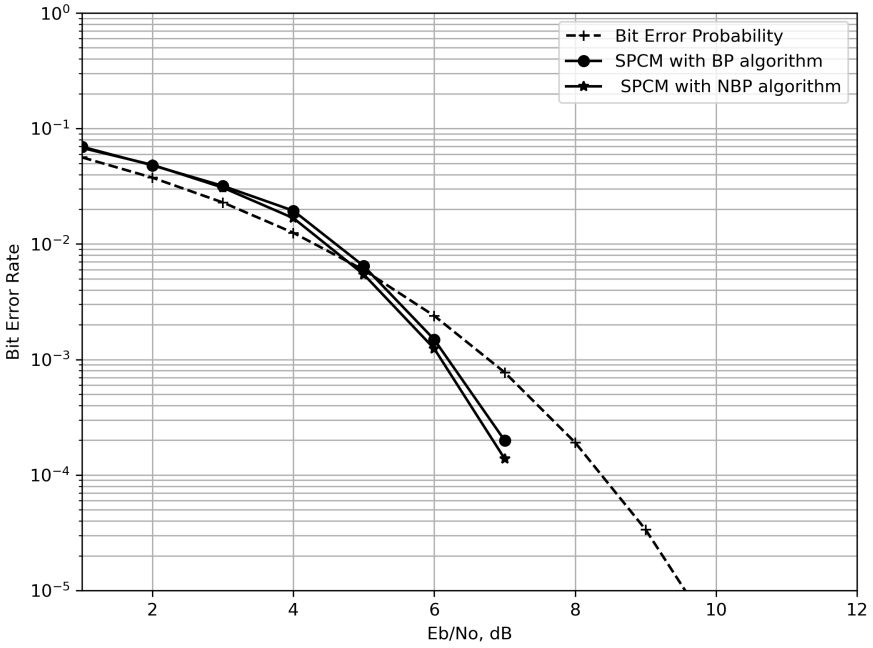


FIGURE 4.8: Performance Comparison Of The SPCM Implemented With NBP Algorithm Vs SPCM Implemented With BP Algorithm For A (31,27) RS Code

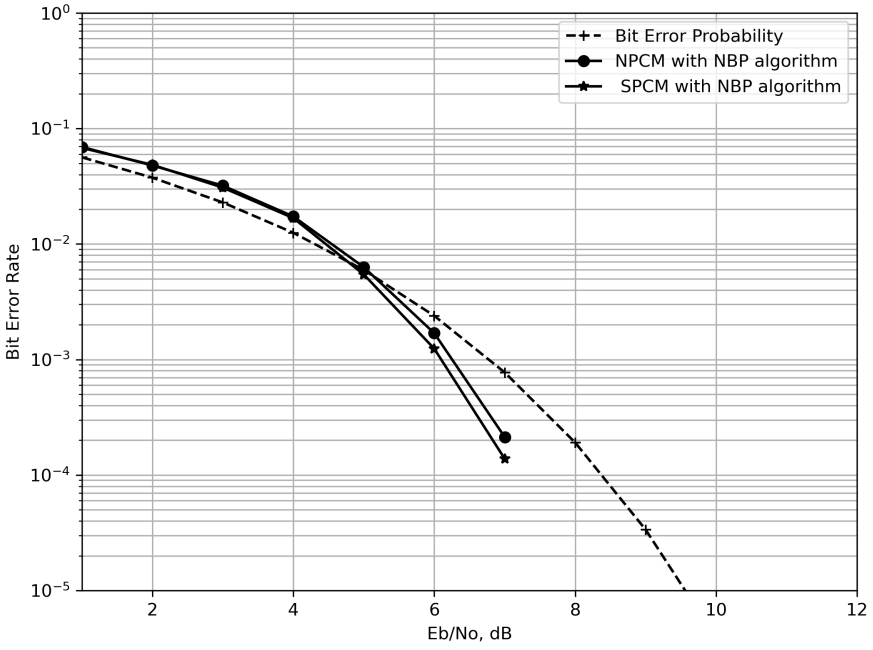


FIGURE 4.9: Performance Comparison Of The SPCM Implemented With NBP Algorithm Vs NPCM Implemented With NBP Algorithm For A (31,27) RS Code

BER of 10^{-3} the SPCM yields a gain of 0.16 dB over the NPCM implemented with the NBP. These results validate that adding sparsity to the parity check matrix improve exchange of information which leads to error correction gains.

4.5 Conclusion

The simulation results, which evaluate the performance of iterative soft decision decoders using the proposed parity check matrix are outlined. This methodology is implemented within both the BP and NBP algorithms adapted for high-rate RS codes. The results indicate a improvement in error correction performance. This improvement is attributed to the efficient exchange of soft information during the iterative decoding process facilitated by the incorporation of additional sparsity into the parity check matrix.

CHAPTER 5

Chapter 5 Graph Neural Network Based Belief Propagation Algorithm For Reed Solomon Codes

5.1 Introduction

This chapter proposes a suitable Graph Neural Network (GNN) based on the Belief Propagation (BP) algorithm for Reed Solomon (RS) codes. A detailed description of the algorithm's implementation on RS codes is presented. Finally, the performance of the GNN model for RS codes is presented. The chapter analyses the results of the proposed model compared to other iterative soft decision decoders.

5.2 Adaption Of The GNN Algorithm To RS Codes

This section discusses the adaptation of the GNNBP algorithm for RS codes, with a focus on the implementation process. The codeword generation is addressed in Section 3.2.3.1.

5.2.1 GNN Model Design

The GNN model design is chosen based on the work done in [33, 34]. However, the model proposed in this research is adapted to work with RS codes. From Chapter 2, it is known that the GNNBP model is represented by the Tanner graph. Therefore, based on the functions explained in Section 2.5.4 the model design is shown in Fig. 5.1.

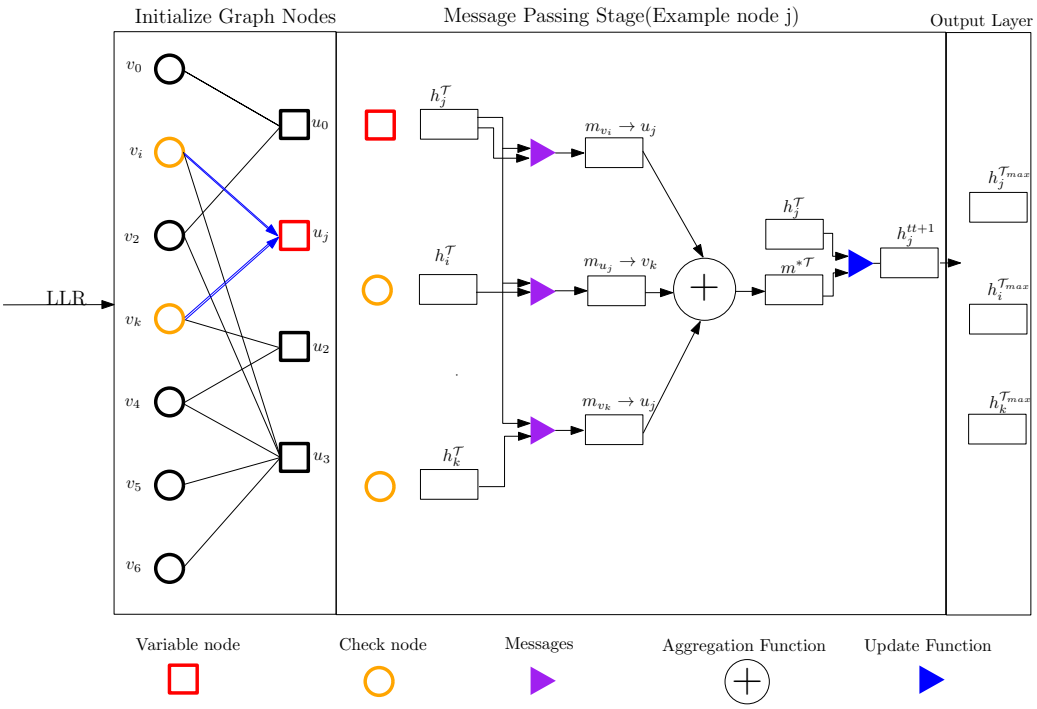


FIGURE 5.1: Architecture of GNN Decoder

Stage 1: Initialization - For an input graph shown in Fig. 5.1, the GNN generates a vector $\mathbf{h}_i$ that stores hidden states and initializes the feature states of the graph. The hidden states $\mathbf{h}_i$ store the variable and check node states as well as the feature states. The initial state for the check node is given as $\mathbf{h}_u = 0$ and for the variable node is given as $\mathbf{h}_{v_i} = (1 - \alpha) \times (L_{ch,i} \times w)$ where w is defined as the input embedding and α is the damping factor used for improve decoding performance. This embedding is utilized during training and transforms the LLR ($L_{ch,i}$) vector from the channel into a denser representation(network) suitable for processing by the GNN. The input dimensions correspond to node values and the batch size.

Stage 2: Message Passing Phase - The messages passing phases interprets messages sent from the variable to check nodes and vice versa. These messages are summed up and the computed node embedding is updated. Multi-Layer Perceptrons (MLP) play a crucial role in this phase as they gather and process node/edge features. This allows them to capture information about the relationships between these elements, ultimately improving the model's performance. The message passing phase is made up of three sections

- Message Function - Message function takes account of the messages from the check nodes and variable node respectively.These are defined as $\{\mathbf{m}_{v_i \to u_j}, \mathbf{m}_{u_j \to v_i}\} \in \mathbb{R}^{F_m}$. Therefore processing the related message in the neighborhood.

- Update Function - There are two types of updates namely the node update and the edge update. The updates function combines the current node states using the aggregate function based on the information the neighborhood. The message passing phase occurs for $\mathcal{T}$ iteration.

- Aggregate Function - The messages computed by the message function from the variable and check node updates. Thereafter, the aggregate function sums the compiled messages.

Stage 3: Output Layer - The output layer computes the final estimate of the predicted codeword.

5.2.2 Model Training Phase

The chosen model training for the proposed GNNBP algorithm is based on the work done in [33, 34]. The training of the model training process is essential and in turn directly affects the overall model performance.

The chosen loss function Binary Cross Entropy (BCE) measures the binary-cross entropy between the true and predicted codewords. The chosen loss function is motivated by [33, 34]. The work shows that the BCE loss function is suitable for this work and leads to better performance.

Previous work [33, 34, 65, 66] in developing the GNN model for channel decoding of linear block codes conducted multiple experiments in order to compile suitable hyperparameters for the GNN model [72]. Taking this into account, we selected the most suitable hyperparameters that would work well with the GNN model specifically for RS codes based on the work done on BCH codes. This is because RS codes are a sub-class the BCH codes [33], the chosen hyperparameters are shown in Table 3.3.

TABLE 5.1: GNN Model Hyperparameters.

Parameter	Batch 1	Batch 2	Batch 3
Activation	tanh		
Number of iteration	8		
Feature vector(nodes) - F_n	20		
Feature vector(messages) - F_m	20		
Number of hidden units MLP	40		
Number of MLP units	2		
Aggregation Function	8		
LLR clipping	20		
Batch Size	256		
Learning rate	10^{-3}	10^{-4}	10^{-5}
Train iteration	35000	500000	500000

The GNN is trained mostly with the ADAM optimization method. Therefore the training phase is computed based on the hyperparameters shown in the Table 3.3. The following steps denote the the specific message passing stages for the GNN based Tanner graph as shown in the GNN model in Fig. 5.1. Based on the work done in [33, 34].

Step 1: The updated message value from the variable node v_i to check node u_j is computed as

$$\mathbf{m}_{v_i \to u_j} = f^{F_m}([\mathbf{h}_{v_i} || \mathbf{h}_{u_j} || g_{m_{v_{i \to u_j}}}], \theta_{\mathbf{m}_{v \to u}}), \tag{5.1}$$

where $\theta_{\mathbf{m}_{v \to u}}$ denotes the parameter to be trained for the node update. Fully connected Multi-layer perceptrons (MLP) are utilized to expand the dimensions. The significance of using MLPs is that information from the neighboring nodes can be processed for model accuracy.

Step 2 : The updated message value from the check node u_j to the variable node v_i are expressed as follows

$$\mathbf{m}_{u_j \to v_i} = f^{F_m}([\mathbf{h}_{u_j} || \mathbf{h}_{v_i} || g_{m_{u_{j \to v_i}}}], \theta_{m_{u \to v}}), \tag{5.2}$$

where $\theta_{\mathbf{m}_{u \to v}}$ denotes the another group of shared parameters to be trained.

Step 3 : For the variable node v_i the updated node value is computed as

$$\mathbf{h}_{v_i}^{'} = f^{F_n}([\mathbf{h}_{v_i}|| \bigoplus_{u_j \in \mathcal{U}(v_i)} \mathbf{m}_{u_j \to v_i} || g_{v_i}], \theta_v), \tag{5.3}$$

where θ_v is a trainable parameter to be shared for the variable node update function and $\bigoplus$ is defined as the message aggregation function. A mean operator for $\bigoplus$ is utilized.

Step 4: For the check node u_j the updated node values is computed as

$$\mathbf{h}_{u_j}^{'} = f^{F_n}([\mathbf{h}_{u_j}|| \bigoplus_{u_j \in \mathcal{U}(u_j)} \mathbf{m}_{v_i \to u_j} || g_{u_j}], \theta_u), \tag{5.4}$$

where θ_u is a trainable parameter to be shared for the check node update function.

Step 5: Hard decision of the estimated bits which is computed in the last layer as shown in Fig. 5.1

$$L_i = Q^{\mathcal{T}} h_{v_i}, \tag{5.5}$$

$$\hat{x} = \begin{cases} 1 & \text{if } L_i > 0 \\ 0 & \text{otherwise} \end{cases}, \tag{5.6}$$

where Q represents the output embedding derived from a last layer which is a dense network and L_i is the updated LLRs.

5.2.3 Selection Of The Damping Coefficient

The proposed GNNBP algorithm is adapted for RS codes. The proposed approach involves using a scalar damping factor of $\alpha = 0.8$ to scale the trainable parameters of the GNNBP algorithm.

5.2.4 Model Validation Phase

This section discusses the model validation process for the proposed GNNBP algorithm. A validation dataset is used to provide an unbiased evaluation of the model's performance on the training dataset while tuning its hyperparameters. The model validation phase utilizes data from the training phase to ensure fine-tuning of the model.

After every 1000 training iterations, a new batch of 10,000 samples is generated with random SNR values. The model predicts codewords for this batch, and the BCE is calculated. To evaluate performance, the log-likelihood ratios are thresholded to generate hard decisions, which are compared to the true codewords. The BER, which measures the incorrectly predicted codeword bits. The loss and BER are logged to monitor the model's validation performance over time.

5.2.5 Model Testing Phase

This section discusses the model testing, for the GNNBP algorithm. Detailing how to evaluate model performance using the testing datasets. The performance of the trained GNNBP model is evaluated based on the following:

- Testing set - The trained model is evaluated by using a batch size of 10000 random codewords.
- Range of SNR values - As mentioned in Section 3.2.4 the SNR affects the performance of the GNN based BP model. Therefore, a range of different SNR values are used to trained model in order to evaluate how the model performs under different noisy conditions. The SNR range used in the validation stage is from 1dB to 12dB.

Following the simulation, the Bit Error Rate (BER) is measured and plotted against the range of SNR values. The performance of the GNN based BP model is

compared to the neural BP decoder in order to evaluate performance of the GNN based BP decoder.

5.3 Performance Of The GNNBP Algorithm Vs NBP Algorithm

The proposed GNN implementation is tested with the RS codes of length (15,9), (15,11), (15,13). The reason behind utilizing these codes is to investigate how the GNN performs with different high-rate RS codes based on GF(16) constraints. The error correction performance is measured using the Bit Error Rate (BER) and the number of iterations needed to decode the received vector by the GNN implementation, in comparison to the NBP algorithm. It is noted that the proposed Sparse Parity Check Matrix (SPCM) described in Chapter 4 is not applied to the GNNBP decoder. This is because the larger parity check matrix presents additional computational complexity due to more nodes in the Tanner graph passing messages with all their neighboring nodes which is resource intensive.

5.3.1 Bit Error Rate Performance Analysis

5.3.1.1 GNNBP Vs NBP Algorithm For A (15,9) RS Code

The results of the simulation comparing the GNN model and the NBP algorithm for the (15,9) RS codes can be seen in Fig. 5.2. The result shows an improvement in error correction performance. This highlights the effectiveness of the proposed algorithm in improving performance iterative soft decision of RS codes.

From Fig. 5.2 it can be seen that the GNNBP algorithm yields a performance gain of 0.48 at a BER of 10^{-6} when compared to the NBP algorithm.The GNN structure has an advantage over the NBP because the GNN implementation takes account of all the nodes and edges in the messages passing algorithm. This characteristic of the GNN implementation leads to an improvement in error correction performance gain.

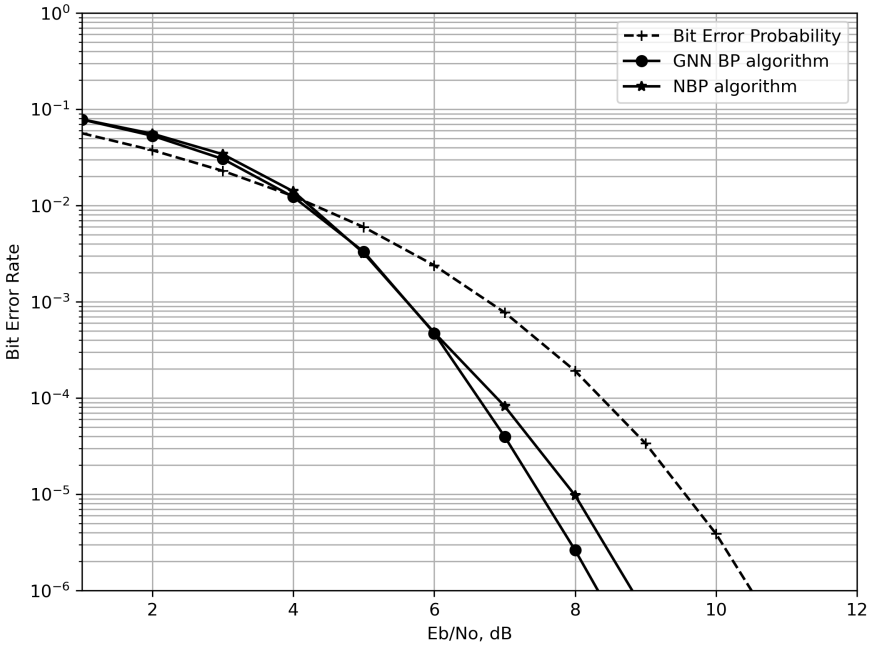


FIGURE 5.2: Performance Comparison Of GNNBP Algorithm And NBP Algorithm For A 15,9 RS Code

5.3.1.2 GNNBP Vs NBP Algorithm For A (15,11) RS Code

A similar experiment simulation is performed for a (15, 11) RS code. This is carried out in order to investigate how the GNN BP decoder performs with higher RS codes. The results for this simulation with regards to BER are presented in Fig. 5.3.

From Fig. 5.3 it is observed that there is a gain for the GNNBP decoder when compared to the NBP decoder. At a BER of 10^{-6} the proposed decoder yields a gain of 0.54 dB over NBP decoder. The performance of the GNNBP improves with the (15,11) RS code. This is attributed to the graph structure of the GNN implementation as mentioned previously. Therefore, resulting in performance gains for the (15,11) RS code. The GNNs implementation captures soft information about the relationships between variable nodes and check node between all the neighboring nodes which leads to the improvement of the model's performance.

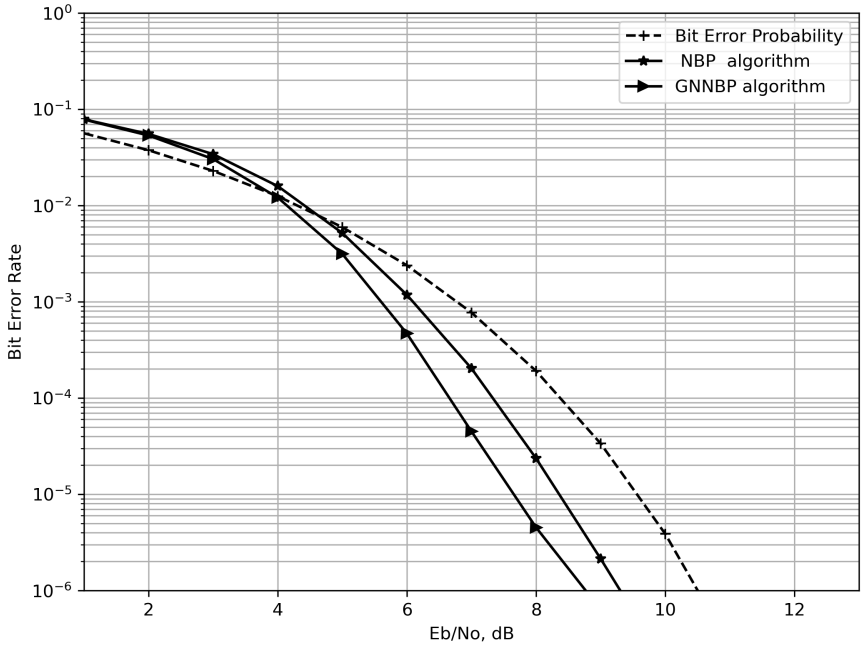


FIGURE 5.3: Performance Comparison Of GNNBP Algorithm And NBP Algorithm For A 15,11 RS Code

5.3.1.3 GNNBP Vs NBP Algorithm For A (15,13) RS Code

An additional simulation is performed for a (15, 13) RS code. The results can be seen in Fig. 5.4.

From Fig. 5.4 it is observed that there is a gain for the GNNBP decoder when compared to the NBP decoder. At a BER of 10^{-6} the proposed decoder yields a gain of 0.84 dB over NBP decoder. From this result, we can infer that the performance of the GNN for RS codes of length 15 improves as the rate of the code increases.

5.3.2 Average Iteration Count

The performance of the algorithm can be evaluated by measuring the number of iterations it takes to reach a correct predicted codeword. These simulations were ran alongside experiments in Fig. 5.2, Fig. 5.3, Fig. 5.4. In Fig. 5.5, the average

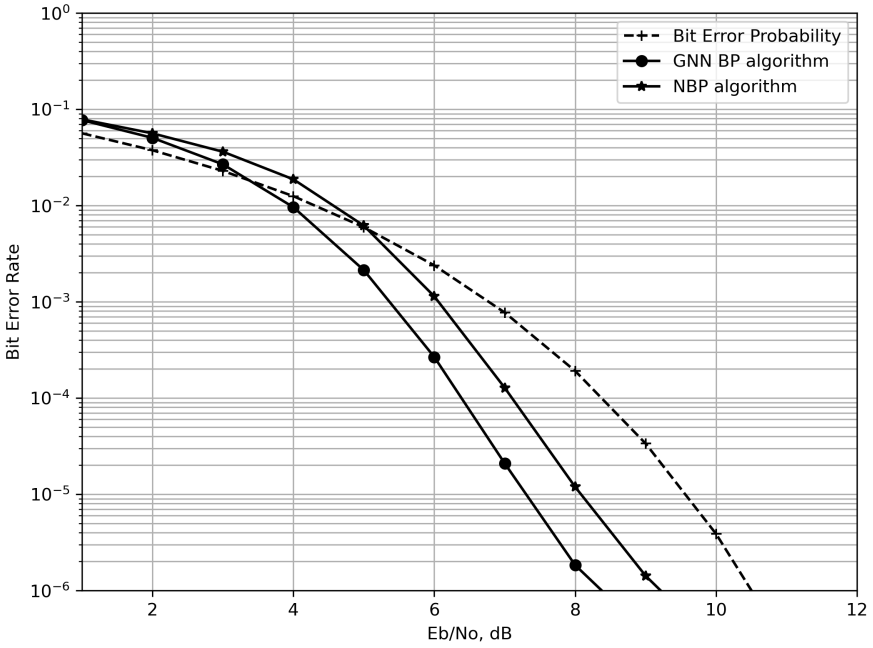


FIGURE 5.4: Performance Comparison Of GNNBP Algorithm And NBP Algorithm For A 15,13 RS Code

number of iterations for the GNN implementation in comparison with the NBP implementation is presented. The GNNBP algorithm converges faster to a valid codeword during iterative decoding when compared to the NBP algorithm. The convergence of the GNNBP algorithm occurs within 8 iterations for higher SNR values, whereas the NBP algorithm requires 20 iterations. This reduction in the number of iterations holds significance, particularly considering that the GNNBP is more computationally expensive than the NBP for a single iteration. This is because the GNNBP uses a large number of MLP network aggregation message in each iteration.

5.4 Conclusion

This chapter presents the performance of the GNNBP algorithm for RS codes. The key idea is to build a graph that represents the BP algorithm Tanner graph, which allows for efficient message passing between variable nodes and factor nodes. The damping factor is implemented to further improve the performance of the proposed

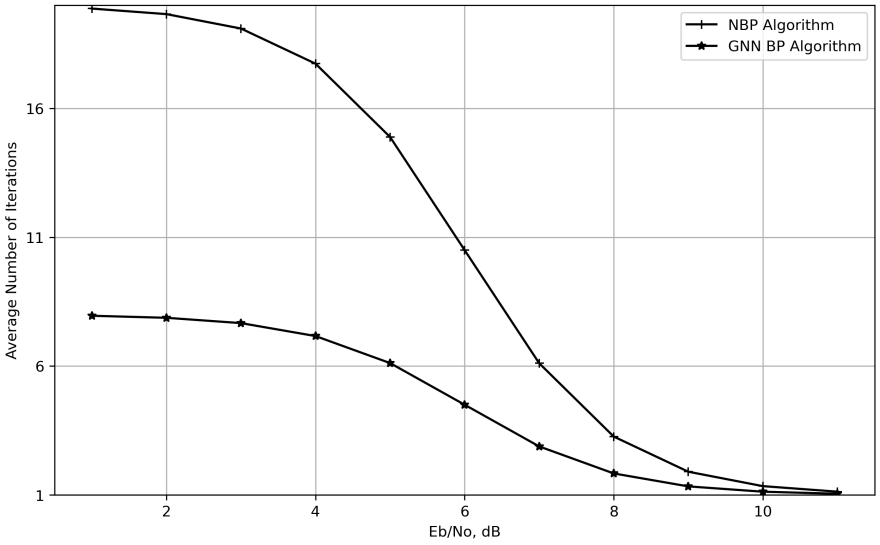


FIGURE 5.5: Average Number Of Iteration

implementation. The BER performance of the GNNBP algorithm for 8 iterations exceeds the BER performance of NBP algorithm for 20 iterations. Therefore, improving the iterative soft decision decoding of RS codes using deep learning techniques.

CHAPTER 6

Conclusion And Future Work

6.1 Conclusion

The main objective of this research is to develop a decoding approach that uses various FEC and deep learning techniques to enhance the decoding performance of Reed Solomon (RS) codes. The approach is based on iterative soft decision decoding algorithms for RS codes on the bit-level. The bit-level implementation is achieved by applying binary image expansion (BIE) to the RS parity check matrix.

In chapter 4, improvements to iterative soft decision decoding of RS codes are presented. The techniques are used to investigate the objectives presented in Chapter 1. One of the techniques involve increasing sparsity to the parity check matrix through addition of low weight parity check equations to the H matrix that match the minimum distance of RS codes. Then cyclically shifting the element n times. The technique led to a relatively sparse parity check matrix, facilitating improved exchange of soft information and consequently better error correction performance. This method is tested across the Belief Propagation (BP) algorithm and the Neural Belief Propagation (NBP) algorithms for RS code of lengths (15,11), (31,27) respectively. The simulation results show that the sparse approach helps improve error correction performance of the iterative soft decision decoders for RS codes.

In Chapter 4, the NBP decoder adapted for RS codes was introduced. Another objective was explored to improve the error correction performance of this adapted NBP decoder. This improvement involved the use of a scalar damping factor, resulting in improved error correction performance when compared to the BP

decoder. The approach is based on the work done in [18]. To ensure a fair comparison for RS codes, all-zero codewords are utilized during the decoding process. As demonstrated in Chapter 4, this approach resulted in a slight improvement in Bit Error Rate (BER) performance without introducing additional computational complexity [18].

Chapter 5 introduces a GNN implementation aimed at improving the error correction performance of iterative soft-in and soft-out decoding of RS codes while reducing the of iterations. GNNs [28] are a type of neural networks that can handle graph structured data. The GNNBP model was previously applied to BCH codes and binary LDPC codes [33, 34]. In this research, we adapt the model to function with RS codes, which are non-binary codes. This adaptation was achieved by working on a bit-level, resulting in longer relatively more dense codes when compared to BCH codes. To counter these challenges, a scalar damping factor is applied to further improve the error correction performance of the GNN for RS codes. Integration of the Tanner graph [34] with the GNN model facilitates the development of the GNNBP implementation. The network structure is designed for codewords of length GF(16), as the GNN model is computational complex. The proposed implementation is compared with the proposed NBP algorithm. The results show that the GNN implemention yields gains for error correction performance. Despite its complex structure, the GNN implementation requires fewer iterations, reducing the compounded complexity associated with running the BP operation for each iteration.

6.2 Future Work

The work done in this research focused on the implementation of error correction and deep learning techniques to improve iterative soft decision decoding of RS codes. These techniques resulted in an error correction performance gain, however there are still some improvements that can be implemented.

For the sparse implementation the approach can be applied on a symbol-level parity check matrix for RS codes. Since the symbol-level parity check matrix is dense this approach can improve exchange of soft information at a symbol-level. This proposed approach can be applied to other iterative soft decision decoding algorithm. As seen in Chapter 4 the sparse implementation results in better error correction performance. However, the algorithm does not add complexity.

The GNNBP implementation outperforms the NBP implementation with less number of iteration. However, the model uses a large number of MLP networks to aggregate messages in each iteration. This results in a larger computational cost. Research carried out in [77] proposes a method to reduce the complexity of the GNNBP implementation for BCH codes. A similar approach can be investigated for iterative soft decision decoding of RS codes. This would in turn improve on the error detection and error correction while reducing the computational complexity for RS codes.

References

[1] B. Kamali, "Error Control Coding," *IEEE Potentials*, vol. 14, no. 2, pp. 15–19, 1995.

[2] W. E. Ryan *et al.*, "An Introduction to LDPC Codes," *CRC Handbook for Coding and Signal Processing for Recording Systems*, vol. 5, no. 2, pp. 1–23, 2004.

[3] G. Forney, "On Decoding BCH Codes," *IEEE Transactions on information theory*, vol. 11, no. 4, pp. 549–557, 1965.

[4] I. S. Reed and G. Solomon, "Polynomial Codes Over Certain Finite Fields," *Journal of the society for industrial and applied mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[5] M. Grassl, W. Geiselmann, and T. Beth, "Quantum Reed-Solomon Codes," in *International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes*. Springer, 1999, pp. 231–244.

[6] P. Gimenez, D. Ruano, and R. San-José, "Entanglement-assisted quantum error-correcting codes from subfield subcodes of projective Reed–Solomon codes," *Computational and Applied Mathematics*, vol. 42, no. 8, Nov. 2023. [Online]. Available: http://dx.doi.org/10.1007/s40314-023-02506-4

[7] I. A. Shah, F. A. Malik, and S. A. Ahmad, "Enhancing Security in loT based Home Automation using Reed Solomon Codes," in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*. IEEE, 2016, pp. 1639–1642.

[8] C. Hsu, I. Reed, and T. Truong, "Use of the RS Decoder as an RS Encoder for Two-way Digital Communications and Storage Systems," *IEEE transactions on circuits and systems for video technology*, vol. 4, no. 1, pp. 91–92, 1994.

[9] A. Levina, I. Kuzmin, and S. Taranov, "Reed Solomon Codes and its Application for Cloud Storage System," in *Third International Conference on Future Generation Communication Technologies (FGCT 2014).* IEEE, 2014, pp. 46–49.

[10] K. Lebed, H. Liu, and J. Luo, "Construction of MDS Self-dual Codes over Finite Fields," *Finite Fields and Their Applications*, vol. 59, pp. 199–207, 2019.

[11] M. El-Khamy, R. J. McEliece, and J. Harel, "Performance Enhancements for Algebraic Soft Decision Decoding of Reed-Solomon Codes," in *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.* IEEE, 2004, p. 421.

[12] A. Duggan and A. Barg, "Performance Analysis of Algebraic Soft-Decision Decoding of Reed–Solomon Codes," *IEEE Transactions on Information Theory*, vol. 54, no. 11, pp. 5012–5018, 2008.

[13] J.-I. Park, K. Lee, C.-S. Choi, and H. Lee, "High-Speed Low-Complexity Reed-Solomon Decoder using Pipelined Berlekamp-Massey Algorithm," in *2009 International SoC Design Conference (ISOCC).* IEEE, 2009, pp. 452–455.

[14] G. Forney, "Generalized Minimum Distance Decoding," *IEEE Transactions on Information Theory*, vol. 12, no. 2, pp. 125–131, 1966.

[15] R. Koetter and A. Vardy, "Algebraic Soft-Decision Decoding of Reed–Solomon Codes," *IEEE Transactions on Information Theory*, vol. 49, no. 11, pp. 2809–2825, 2003.

[16] J. Xing, L. Chen, and M. Bossert, "Low-Complexity Koetter-Vardy Decoding of Reed-Solomon Codes using Module Minimization," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–6.

[17] W. Zhang, S. Zou, and Y. Liu, "Iterative Soft Decoding of Reed-Solomon Codes Based on Deep Learning," *IEEE Communications Letters*, vol. 24, no. 9, pp. 1991–1994, 2020.

[18] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to Decode Linear Codes Using Deep Learning," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2016, pp. 341–346.

[19] J. Xing, M. Bossert, S. Bitzer, and L. Chen, "Iterative Decoding of Non-binary Cyclic Codes using Minimum-Weight Dual Codewords," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 333–337.

[20] F. Shayegh and M. R. Soleymani, "Efficient Iterative Techniques for Soft Decision Decoding of Reed-Solomon Codes," *IEEE Transactions on Communications*, vol. 59, no. 2, pp. 428–436, 2010.

[21] B. Kamali and A. H. Aghvami, "Belief propagation decoding of Reed-Solomon codes; A Bit-Level Soft Decision Decoding Algorithm," *IEEE transactions on broadcasting*, vol. 51, no. 1, pp. 106–113, 2005.

[22] J. Jiang and K. R. Narayanan, "Iterative Soft Decision Decoding of Reed Solomon Codes Based on Adaptive Parity Check Matrices," in *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.* IEEE, 2004, p. 261.

[23] J. Jiang and K. Narayanan, "Iterative Soft-Input Soft-Output Decoding of Reed–Solomon Codes by Adapting the Parity-Check Matrix," *IEEE Transactions on Information Theory*, vol. 52, no. 8, pp. 3746–3756, 2006.

[24] V. B. Wijekoon, H. Dau, and E. Viterbo, "Iterative Decoding of Reed-Solomon Codes based on Non-binary Matrices," in *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2019, pp. 1082–1086.

[25] A. Bennatan, Y. Choukroun, and P. Kisilev, "Deep Learning for Decoding of Linear Codes - A Syndrome-Based Approach," in *2018 IEEE International Symposium on Information Theory (ISIT)*, 2018, pp. 1595–1599.

[26] E. Kavvousanos, V. Paliouras, and I. Kouretas, "Simplified Deep-Learning-based decoders for linear block codes," in *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2018, pp. 769–772.

[27] E. Nachmani, E. Marciano, D. Burshtein, and Y. Be'ery, "RNN Decoding of Linear Block Codes," *ArXiv*, vol. abs/1702.07560, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:18262730

[28] L. Wu, P. Cui, J. Pei, L. Zhao, and X. Guo, "Graph Neural Networks: Foundation, Frontiers and Applications," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 4840–4841.

[29] D. B. West *et al.*, *Introduction to Graph Theory*. Prentice hall Upper Saddle River, 2001, vol. 2.

[30] Z. Liu and J. Zhou, *Introduction to Graph Neural Networks*. Springer Nature, 2022.

[31] G. Dong, M. Tang, Z. Wang, J. Gao, S. Guo, L. Cai, R. Gutierrez, B. Campbel, L. E. Barnes, and M. Boukhechba, "Graph Neural Networks in IoT: A Survey," *ACM Trans. Sen. Netw.*, vol. 19, no. 2, apr 2023.

[32] M. Lange, P. Havström, B. Srivastava, V. Bergentall, K. Hammar, O. Heuts, E. van Nieuwenburg, and M. Granath, "Data-driven decoding of quantum error correcting codes using graph neural networks," 2023.

[33] S. Cammerer, J. Hoydis, F. A. Aoudia, and A. Keller, "Graph Neural Networks for Channel Decoding," in *2022 IEEE Globecom Workshops (GC Wkshps)*, 2022, pp. 486–491.

[34] V. G. Satorras and M. Welling, "Neural Enhanced Belief Propagation on Factor Graphs," in *International Conference on Artificial Intelligence and Statistics*, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:211988929

[35] X. Yang and Y. Wang, "A Brief Look at Forward Error Correcting codes," in *2011 IEEE 2nd International Conference on Software Engineering and Service Science*. IEEE, 2011, pp. 465–468.

[36] J. C. Moreira and P. G. Farrell, *Essentials of Error-Control Coding*. John Wiley & Sons, 2006.

[37] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes*. Elsevier, 1977, vol. 16.

[38] T. Hurley, "MDS codes over finite fields," *arXiv preprint arXiv:1903.05265*, 2019.

[39] J. Bellorado, A. Kavcic, M. Marrow, and L. Ping, "Low-Complexity Soft-Decoding Algorithms for Reed-Solomon Codes—Part II: Soft-input soft-output Iterative Decoding," *IEEE Transactions on Information Theory*, vol. 56, no. 3, pp. 960–967, 2010.

[40] Y. Jing, W. Zhang, H. Wang, Y. Chang, and Y. Liu, "Improved Adaptive Belief Propagation Decoding of Reed-Solomon Codes with SPC Codes," *IEEE Communications Letters*, 2022.

[41] Y. Genga, O. Oyerinde, and J. Versfeld, "Improved Iterative Convergence Rate for Soft-Decision Bit-Level Reed-Solomon Decoders using Information Set Decoding," in *2019 IEEE 2nd Wireless Africa Conference (WAC)*. IEEE, 2019, pp. 1–5.

[42] S. Scholl, S. K. Haider, and N. Wehn, "An Efficient Soft decision Reed-Solomon Decoder for Moderate Throughput," in *2016 18th Mediterranean Electrotechnical Conference (MELECON)*. IEEE, 2016, pp. 1–6.

[43] F. Liang, C. Shen, and F. Wu, "An Iterative BP-CNN Architecture for Channel Decoding," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, p. 144–159, Feb. 2018. [Online]. Available: http://dx.doi.org/10.1109/JSTSP.2018.2794062

[44] X. Hu and E. Eleftheriou, "Binary Representation of Cycle Tanner-Graph GF (2/sup b/) Codes," in *2004 IEEE International Conference on Communications (IEEE Cat. No. 04CH37577)*, vol. 1. IEEE, 2004, pp. 528–532.

[45] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge university press, 2012.

[46] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination press San Francisco, CA, USA, 2015, vol. 25.

[47] P. Geetha *et al.*, "Comprehensive Overview of the Opportunities and Challenges in AI," in *2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS)*. IEEE, 2023, pp. 420–423.

[48] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, p. 85–117, Jan. 2015. [Online]. Available: http://dx.doi.org/10.1016/j.neunet.2014.09.003

[49] J. Brownlee, *Deep Learning for Natural Language Processing: Develop Deep Learning Models for Your Natural Language Problems*. Machine Learning Mastery, 2017.

[50] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 8599–8603.

[51] T. Gruber, S. Cammerer, J. Hoydis, and S. t. Brink, "On Deep Learning-Based Channel Decoding," in *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, 2017, pp. 1–6.

[52] W. Lyu, Z. Zhang, C. Jiao, K. Qin, and H. Zhang, "Performance Evaluation of Channel Decoding with Deep Neural Networks," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[53] D. Artemasov, K. Andreev, P. Rybin, and A. Frolov, "Soft-Output Deep Neural Network-Based Decoding," in *2023 IEEE Globecom Workshops (GC Wkshps)*, 2023, pp. 1692–1697.

[54] K. Andreev, A. Frolov, G. Svistunov, K. Wu, and J. Liang, "Deep Neural Network Based Decoding of Short 5G LDPC Codes," in *2021 XVII International Symposium" Problems of Redundancy in Information and Control Systems"(REDUNDANCY)*. IEEE, 2021, pp. 155–160.

[55] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[56] E. B. Baum, "On the Capabilities of Multilayer Perceptrons," *Journal of complexity*, vol. 4, no. 3, pp. 193–215, 1988.

[57] N. Buduma, N. Buduma, and J. Papa, *Fundamentals of Deep Learning.* " O'Reilly Media, Inc.", 2022.

[58] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, 2020.

[59] J. D. Kelleher, *Deep Learning.* MIT press, 2019.

[60] E.Nachmani, E.Marciano, L. Lugoschand W.J .Gross, and D. Y. Be'ery, "Deep Learning Methods for Improved Decoding of Linear Codes," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, p. 119–131, Feb. 2018. [Online]. Available: http://dx.doi.org/10.1109/JSTSP.2017.2788405

[61] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666651021000012

[62] W. Fan, Y. Ma, Q. Li, Y. He, Y. E. Zhao, J. Tang, and D. Yin, "Graph Neural Networks for Social Recommendation," *The World Wide Web Conference*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:67769538

[63] Y. Wu, J. Yang, X. Zhou, L. Wang, and Z. Xu, "Exploring Graph-aware Multi-View Fusion for Rumor Detection on Social Media," 2022.

[64] M. Krzywda, S. Lukasik, and A. H. Gandomi, "Graph Neural Networks in Computer Vision - Architectures, Datasets and Common Approaches," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, Jul. 2022. [Online]. Available: http://dx.doi.org/10.1109/IJCNN55064.2022.9892658

[65] K. Tian, C. Yue, C. She, Y. Li, and B. Vucetic, "A Scalable Graph Neural Network Decoder for Short Block Codes," in *ICC 2023 - IEEE International Conference on Communications*, 2023, pp. 1268–1273.

[66] Z. Zhang, M. H. Dupty, F. Wu, and F. Wu, "Factor Graph Neural Networks," *ArXiv*, vol. abs/2308.00887, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:227276486

[67] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge university press, 2008.

[68] M. Viswanathan, *Digital Modulations using Python*. Mathuranathan Viswanathan, 2019.

[69] M. Montalbano, "Tables, flow charts, and program logic," *IBM Systems Journal*, vol. 1, no. 1, pp. 51–63, 1962.

[70] M. Hostetter, "Galois: A performant NumPy extension for Galois fields," 11 2020. [Online]. Available: https://github.com/mhostetter/galois

[71] J. Hoydis, S. Cammerer, F. Ait Aoudia, A. Vem, N. Binder, G. Marcus, and A. Keller, "Sionna: An Open-Source Library for Next-Generation Physical Layer Research," *arXiv preprint*, Mar. 2022.

[72] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A Research Platform for Distributed Model Selection and Training," *ArXiv*, vol. abs/1807.05118, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:49741140

[73] G. Van Rossum and F. L. Drake, "Python library reference," 1995.

[74] N. Shukla and K. Fricklas, *Machine Learning with TensorFlow*. Manning Greenwich, 2018.

[75] C. Leroux, S. Hemati, S. Mannor, and W. J. Gross, "Stochastic Chase Decoding of Reed-Solomon Codes," *IEEE Communications Letters*, vol. 14, no. 9, pp. 863–865, 2010.

[76] Y. Genga, O. O. Oyerinde, and J. Versfeld, "Iterative Soft-Input Soft-Output Bit-Level Reed-Solomon Decoder Based on Information Set Decoding," *SAIEE Africa Research Journal*, vol. 112, no. 2, pp. 52–65, 2021.

[77] Q. Wu, B. Ng, C. T. Lam, X. Cen, Y. Liang, and Y. Ma, "Shared Graph Neural Network for Channel Decoding," *Applied Sciences*, vol. 13, p. 12657, 11 2023.