

Niklas Savijoki

OPTIMIZING LDPC DECODING WITH MACHINE LEARNING USING LEARNABLE WEIGHTS

A Hybrid Framework Integrating Neural Networks with
Iterative Message-Passing for Enhanced Error-Correction in
5G Systems

Master's thesis
Faculty of Information Technology and Communication Sciences
Examiners: Dr. Jukka Talvitie
Prof. Mikko Valkama
May 2025

ABSTRACT

Niklas Savijoki: Optimizing LDPC Decoding with Machine Learning Using Learnable Weights
Master's thesis
Tampere University
Master's Programme in Information Technology
May 2025

This thesis investigates the optimization of Low-Density Parity-Check (LDPC) decoding through machine learning techniques, with a specific focus on 5G communication systems. Traditional LDPC decoding algorithms face a critical trade-off: Belief Propagation (BP) offers excellent error-correction performance but with high computational complexity, while Min-Sum (MS) reduces complexity at the cost of degraded performance. To address this limitation, a hybrid neural network architecture is proposed that incorporates learnable normalization and offset weights into the message-passing framework of LDPC decoders.

This approach integrates multiplicative (normalization) and additive (offset) weights into the Tanner graph structure, allowing the decoder to adapt to specific channel conditions. A novel weighted binary cross-entropy loss function is developed that prioritizes entire block recovery by assigning different importance to information and parity bits during training. Experimental results demonstrate that ML-assisted decoder consistently outperforms conventional Min-Sum decoding, approaching the error-correction capability of Belief Propagation while maintaining comparable computational efficiency to Min-Sum.

Comprehensive evaluations across different weight configurations (scalar and vector-based) reveal that increased parametrization generally improves decoding performance. Analysis of the impact of iteration counts shows that the optimal number of decoding iterations varies between different code configurations with studied Basegraph 1 case requiring 15 iterations for optimal performance while studied Basegraph 2 case benefits significantly from 20 iterations.

The proposed ML-assisted LDPC decoder represents a significant advancement for 5G systems, offering improved error correction at lower signal strengths while balancing latency and computational requirements. This research contributes to the ongoing evolution of wireless technology by demonstrating how machine learning can effectively enhance traditional decoding algorithms, providing a practical pathway toward more efficient and adaptive error correction in modern communication systems.

Keywords: LDPC decoding, Machine learning, Neural networks, 5G systems, Error correction, Learnable weights

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

TIIVISTELMÄ

Niklas Savijoki: LDPC-Dekoodauksen optimointi koneoppimisella käyttäen opittuja painoarvoja
Diplomityö
Tampereen yliopisto
Tietotekniikan DI-ohjelma
Toukokuu 2025

Tämä tutkielma tarkastelee Low-Density Parity-Check (LDPC) -dekoodauksen optimointia koneoppimismenetelmien avulla, keskittyen erityisesti 5G-viestintäjärjestelmiin. Perinteiset LDPC-dekoodausalgoritmit kohtaavat kriittisen kompromissin: Belief Propagation (BP) -menetelmä tarjoaa erinomaisen virheenkorjauskyvyn, mutta korkean laskennallisen monimutkaisuuden, kun taas Min-Sum (MS) vähentää monimutkaisuutta suorituskyvyn heikkenemisen kustannuksella. Tämän rajoituksen ratkaisemiseksi tutkielmassa esitetään hybridi neuroverkkoarkkitehtuuri, joka integroi opittavia normalisointi- ja siirtymäpainoja LDPC-dekoodereiden viestinvälityскеhykseen.

Lähestymistapa yhdistää multiplikatiiviset (normalisointi) ja additiiviset (siirtymä) painokertoimet Tanner-graafirakenteeseen, mahdollistaen dekooderin mukautumisen tiettyihin kanavaolosuhteisiin. Tutkielmassa kehitetään uudenlainen painotettu binäärinen ristientropiahäviöfunktio, joka priorisoi kokonaisten lohkojen palautumista antamalla eri painoarvot informaatio- ja pariteettibiteille koulutuksen aikana. Kokeelliset tulokset osoittavat, että koneoppimisavusteinen dekooderi suorituu johdonmukaisesti paremmin kuin perinteinen Min-Sum-dekoodaus, lähestyen uskomusten levityksen virheenkorjauskykyä säilyttäen samalla Min-Sumin laskennallisen tehokkuuden.

Kattavat arvioinnit eri painokonfiguraatioiden (skalaari- ja vektoripohjaisten) välillä paljastavat, että lisääntynyt parametrisointi yleensä parantaa dekoodaus suorituskykyä. Iteraatiomäärien vaikutuksen analyysi osoittaa, että optimaalinen dekoodausiteraatioiden määrä vaihtelee eri koodikonfiguraatioiden välillä, jossa tutkittu Basegraph 1 -tapaus vaatii 15 iteraatiota optimaaliseen suorituskykyyn, kun taas tutkittu Basegraph 2 -tapaus hyötyy merkittävästi 20 iteraatiosta.

Ehdotettu koneoppimisavusteinen LDPC-dekooderi edustaa merkittävää edistysaskelta 5G-järjestelmille, tarjoten parantuneen virheenkorjauksen alemmilla signaaliivoimakkuuksilla tasapainottaen samalla viivettä ja laskennallisia vaatimuksia. Tämä tutkimus edistää langattoman teknologian kehitystä osoittamalla, kuinka koneoppiminen voi tehokkaasti parantaa perinteisiä dekoodausalgoritmeja, tarjoten käytännöllisen polun kohti tehokkaampaa ja mukautuvampaa virheenkorjausta nykyaikaisissa viestintäjärjestelmissä.

Avainsanat: LDPC-dekoodaus, Koneoppiminen, Neuroverkot, 5G-järjestelmät, Virheenkorjaus, Opittavat painot

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck -ohjelmalla.

USE OF ARTIFICIAL INTELLIGENCE IN THIS WORK

Artificial intelligence (AI) has been used in generating this work:

- No
- Yes

I hereby declare, that the AI-based applications used in generating this work are as follows:

Application	Version
OpenAI ChatGPT	GPT-4o, GPT-4o mini

Purpose of the use of AI

ChatGPT was used during the writing process to clarify technical concepts, improve the structure and flow of the English text, and assist in paraphrasing draft content into clearer academic language. It was also consulted to suggest alternative section headings and to ensure logical coherence between paragraphs and sections.

Parts of this work, where AI was used

AI assistance was applied throughout the text, as described above. No AI tools were used in the creation of figures or tables. Furthermore, AI was not involved in the generation of Python code for the decoder implementation.

Acknowledgement of risks

I hereby acknowledge, that as the author of this work, I am fully responsible for the contents presented in this thesis. This includes the parts that were generated by an AI, in part or in their entirety. I therefore also acknowledge my responsibility in the case, where use of AI has resulted in ethical guidelines being breached.

PREFACE

This master's thesis was conducted at Tampere University, Faculty of Information Technology and Communication Sciences, during the autumn of 2024 and spring of 2025. The research explores the integration of machine learning techniques with traditional Low-Density Parity-Check (LDPC) decoding algorithms to enhance error correction capabilities in 5G communication systems.

I would like to express my sincere gratitude to my supervisors, Toni Levanen from Nokia and Jukka Talvitie from Tampere University, for their invaluable guidance, expertise, and support throughout this research journey. Their insightful feedback and encouragement have been instrumental in shaping this work.

I am also grateful to Nokia, where I have worked as a trainee for four years alongside my studies. This professional experience has provided me with practical insights that have significantly enriched my academic research. I especially appreciate Nokia for providing the computational resources necessary for conducting the extensive simulations presented in this thesis, as well as for creating an environment where academic pursuits and professional development could flourish side by side.

My deepest appreciation goes to my family for their unwavering support, patience, and understanding during the demanding process of completing this thesis. Their encouragement has been a constant source of motivation.

Finally, I would like to thank my friends and colleagues at Tampere University for the stimulating discussions and collaborative environment that contributed significantly to my academic growth.

Tampereella, 1st May 2025

Niklas Savijoki

CONTENTS

1.	Introduction	1
1.1	Background and Motivation	1
1.2	Problem Statement	2
1.3	Objectives and Contributions	2
1.4	Thesis Organization.	3
1.5	Significance of the Study.	3
2.	Theoretical Background	4
2.1	Communication System Model and Transmission Chain	4
2.1.1	Overview of Digital Communication Systems	4
2.1.2	Transmission Chain Components.	5
2.1.3	Modulation	6
2.1.4	Channel Models	8
2.2	LDPC Code Theory.	10
2.2.1	Overview of Error-Correcting Codes	10
2.2.2	Code Structure and Graph Representation	11
2.2.3	LDPC Code Construction and Variants	13
2.2.4	Decoding Algorithms	14
2.2.5	Performance Metrics and Evaluation	17
2.2.6	LDPC Codes in 5G New Radio (NR)	19
2.3	Machine Learning Foundations	20
2.3.1	Overview of Machine Learning.	20
2.3.2	Neural Network Architectures	21
2.3.3	Supervised Learning and Training Process	23
2.3.4	Loss Functions and Optimization	24
2.4	Integration of Machine Learning and LDPC Decoding	25
2.4.1	Motivation for ML-Assisted LDPC Decoding	25
2.4.2	Hybrid Decoding Architectures.	26
3.	Research Methodology	28
3.1	Methodological Justification	28
3.2	System Design	29
3.3	Training and Evaluation	30
3.4	Implementation Details	33

4. Results and Analysis	35
4.1 Results of Different Weight Combinations	36
4.1.1 BLER and BER for Basegraph 1	36
4.1.2 BLER and BER for Basegraph 2	39
4.2 Training Metrics	39
4.3 Weight Distribution	41
4.4 Results of Different Number of Decoding Iterations	44
4.5 Analysis of Different Weight Combinations	45
4.6 Analysis of Different Decoding Iterations	49
5. Conclusions	51
References.	56

GLOSSARY

Activation function	A function used in neural networks to introduce nonlinearity (e.g., ReLU, sigmoid, or tanh)
AWGN	Additive White Gaussian Noise
Backpropagation	An algorithm for training neural networks by propagating the error gradient backwards through the network
BER	Bit Error Rate
BLER	Block Error Rate
BP	Belief Propagation
CNN	Convolutional Neural Network
FNN	Feedforward Neural Network
LDPC	Low-Density Parity-Check
LLR	Log-Likelihood Ratio
Loss function	A function that measures the error between a model's predicted outputs and the actual outputs
ML	Machine Learning
MS	Min-Sum
MSE	Mean Squared Error
Optimizer	An algorithm used to update neural network weights during training (e.g., Adam or SGD)
Parity-check matrix	A sparse binary matrix that defines the constraints of an LDPC code
QC	Quasi-Cyclic
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
Sionna	A TensorFlow-based open-source library for simulating the physical layer of wireless and optical communication systems
Tanner graph	A bipartite graph representing the structure of an LDPC code, consisting of variable nodes and check nodes

1. INTRODUCTION

Digital communication systems are the foundation of modern information exchange, enabling accurate transmission of data over intrinsically noisy channels. With the expanding demand for higher data rates and better reliability, advanced error-correcting codes have become essential. Among these, Low-Density Parity-Check (LDPC) codes have come up as a popular choice due to their near-capacity performance and efficient iterative decoding algorithms [17].

1.1 Background and Motivation

LDPC codes were originally introduced by Gallager in the 1960s and later rediscovered as an efficient method for achieving performance close to the Shannon limit. Despite having profound error-correcting capabilities, traditional LDPC decoders, like those based on Belief Propagation (BP) or Min-Sum (MS) algorithms, face notable challenges. Specifically, the computational complexity of BP, which uses operations such as hyperbolic functions, can be too excessive in resource-constrained applications, while the simplified MS algorithm usually experiences a degradation in performance.

In parallel, the advance of Machine Learning (ML) has revolutionized many fields by offering data-driven approaches to traditionally deterministic algorithms. Recent research has shown that utilizing ML techniques in the decoding process can help to alleviate some of the drawbacks of conventional LDPC decoders. By learning optimal message-passing rules and adjusting to varying channel conditions, ML-based decoders show potential in reducing complexity while maintaining or even increasing the decoding accuracy [28, 54, 44].

A persistent bottleneck in practical deployments is the decoding algorithm that follows LDPC encoding. Full-precision Belief Propagation (BP) delivers near-optimal performance but involves hyperbolic and logarithmic operations that are expensive on embedded processors; the simplified Min-Sum (MS) rule removes those operations yet suffers a non-negligible performance loss when signal-to-noise ratios are low or code rates are high [46]. This performance-versus-complexity dilemma is especially acute in 5G user equipment, massive-IoT sensors, and other battery-constrained devices that must meet tight latency budgets while operating under varying channel conditions.

Recent research suggests that machine-learning techniques can bridge this gap. By treating the message-passing schedule inside an LDPC decoder as a trainable graph, neural networks have been shown to learn normalisation, offset, or attention weights that compensate for the sub-optimality of MS while keeping its hardware-friendly arithmetic [35, 30]. Empirical studies report noticeably lower BLER and BER across multiple codes and channels, and the trend is reinforced by the growing availability of on-chip AI accelerators in modern SoCs. These insights motivate the present thesis: to integrate learnable weights into the 5G NR LDPC framework and to quantify how much complexity can be saved without surrendering error-correction capability.

1.2 Problem Statement

Traditional LDPC decoding methods rely upon fixed, pre-defined message passing update rules that may not fully utilize the inherent structure of communication channels. This lack of flexibility may lead to suboptimal performance, especially under difficult channel conditions or in applications with demanding computational constraints. The core problem addressed in this thesis is how to integrate machine learning techniques with traditional LDPC decoding to develop a hybrid decoder that utilizes learned parameters and improves performance metrics such as Bit Error Rate (BER) and Block Error Rate (BLER).

1.3 Objectives and Contributions

The main objective of this thesis is to design and implement ML-assisted LDPC decoder that utilizes the strengths of both traditional decoding algorithms and modern machine learning methods. The main contributions of this work include:

- Developing a hybrid decoding architecture that uses trainable parameters into the message-passing framework of LDPC decoders.
- Formulating a supervised learning framework which optimizes these parameters focusing on minimizing decoding errors under different channel conditions.
- Narrowing the performance gap between the near optimal Belief Propagation and more hardware-friendly Min-Sum algorithms.
- Introducing novel modifications in Binary Cross Entropy loss function leading major performance improvements.
- Performing extensive simulations to evaluate the performance of the proposed decoder in terms of BER and BLER.

1.4 Thesis Organization

The remainder of this thesis is organized as follows:

- **Chapter 2: Theoretical Background** – This chapter reviews the fundamentals of digital communication systems, LDPC code theory, and the essential concepts of machine learning that are relevant to the proposed architecture.
- **Chapter 3: Research Methodology** – This chapter describes the design of the ML-assisted LDPC decoder, including the neural network architecture and training process. Also, implementation details of the used computational systems are described.
- **Chapter 4: Results and Analysis** – This chapter presents the simulation results and a detailed analysis of the performance improvements achieved by the proposed decoder. Also, other relevant visuals are represented, such as model behavior during training and weight distributions of the final model.
- **Chapter 5: Conclusions and Future Work** – This chapter summarizes the main findings of the research, discusses their implications, and outlines potential avenues for future research.

1.5 Significance of the Study

The integration of machine learning into LDPC decoding is a promising pathway toward developing more adaptive and efficient error-correcting systems. With the combination of the powerful theoretical framework of LDPC codes with the flexible capabilities of ML, this research aims to bridge the gap between traditional decoding methods and modern communication requirements.

Building on recent advances in machine-learning-assisted decoding, this thesis refines and unifies several complementary ideas into a single LDPC-decoder architecture: learnable normalisation and offset weights are woven into the standard message-passing schedule for 5G quasi-cyclic codes, yielding a compact framework that balances algorithmic flexibility with implementation practicality.

Simulation results show that the approach consistently improves upon conventional Min-Sum while approaching the performance of full Belief Propagation, yet keeps the low-complexity characteristics valued in hardware design. These findings create a constructive starting point for further work—ranging from ASIC/FPGA realisations and channel-adaptive variants to the exploration of similar learning mechanisms for broader classes of error-correcting codes.

2. THEORETICAL BACKGROUND

2.1 Communication System Model and Transmission Chain

This section introduces the fundamental concepts of digital communication systems, explaining the processes used in transmitting data from a sender to a receiver over a noisy channel. It provides the background needed to understand where LDPC codes are applied in the transmission chain.

2.1.1 Overview of Digital Communication Systems

Digital communication systems are vital to the present day civilization, allowing the efficient and stable transmission of information over different media, such as wired networks, wireless channels, and optical fibers. The primary objective of these technologies is to transfer data from a sender to a receiver correctly and efficiently, while handling the occurring noise and other impairments in the used communication channel.

Considering at a high level, there are three primary components in a digital communication system: the transmitter, the communication channel, and the receiver. The transmitter converts the source information into a sufficient format for transmission, the channel functions as the medium through which the information propagates as a signal, and the receiver reconstructs the original information from the received signal. This process consists of several stages, including source encoding, channel encoding, modulation, transmission, demodulation, channel decoding, and source decoding, each of which plays an integral role in the overall performance of the system.

Advancements in digital techniques have revolutionized communication systems by offering developments such as higher data rates, better noise immunity, and the ability to use complex signal processing algorithms. Additionally, digital systems are more scalable and flexible, making it possible to integrate many different services like data, voice, and video into the same network infrastructure. Therefore, digital communication has become the foundation of technologies that extend from mobile telephony and satellite communications to the Internet and multimedia data transmission. [39]

It is essential to understand the basic principles of digital communication systems to ap-

precipitate the role of error correction codes such as LDPC codes in improving system efficiency and reliability. The following sections delve into specific components of the transmission chain and the challenges posed by the communication channel.

2.1.2 Transmission Chain Components

The transmission chain in a digital communication system includes a chain of processing steps that convert the original information into a suitable form for transmission over a physical medium and then back into the original format at the receiver end. Figure 2.1 illustrates the transport channel processing in a 5G communication system, highlighting the steps related to channel coding [14].

A fundamental aspect of this chain is *source encoding*, which compresses the original data by reducing redundancy. This process minimizes the amount of information that is needed to be transmitted without negatively impacting its essential content. Algorithms such as Huffman coding and Lempel-Ziv-Welch are widely used for source encoding in modern communication systems [19]. To enhance the transmission reliability, *channel encoding* adds controlled redundancy into the data. This redundancy allows the receiver to identify and correct errors caused by noise and interference during transmission. Techniques such as Low-Density Parity-Check (LDPC) codes, Turbo codes, and convolutional codes are commonly employed in this context [31]. Once the data is encoded, it undergoes *modulation*, a step where the digital information is mapped onto an analog signal waveform suitable for transmission. To utilize the available bandwidth efficiently, modulation schemes such as Quadrature Amplitude Modulation (QAM) and Phase Shift Keying (PSK) are used [39].

The modulated signals are then transmitted through the physical medium, which can be, for example, wireless channel, copper wire, or optical fiber. The physical constraints of the medium and other external factors can negatively impact the signal quality.

At the receiver end, *demodulation* reverses the modulation process by converting the analog signals back into a digital format. This step extracts the transmitted symbols from the received waveforms and prepares the data for subsequent processing. Following demodulation, *channel decoding* leverages the redundancy introduced during channel encoding to detect and correct errors, restoring the original data bits. At the end, the *source decoding* tries to reconstruct the original information by reversion of the compression performed during source encoding. This ensures that the transmitted data is accurately recovered and is suitable for use in its intended application.

Each component has to be carefully designed to maximize total system performance. Also, it's vital to consider several factors including power consumption, bandwidth efficiency, latency and resilience to channel disturbances. The usage of advanced coding

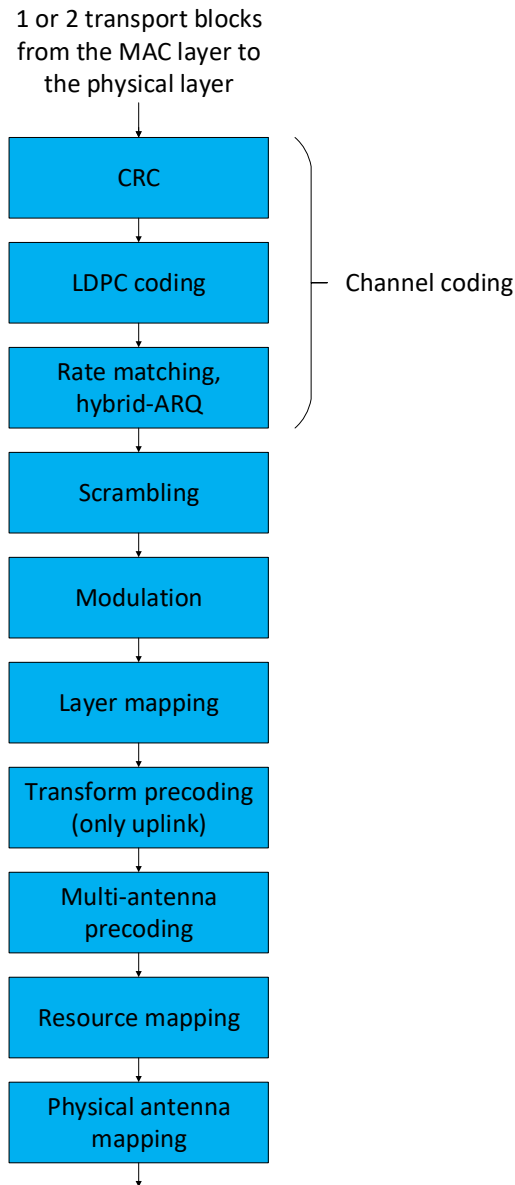


Figure 2.1. Illustration of the transport channel processing in 5G. Steps related to channel coding are highlighted. Adapted from [14].

techniques such as LDPC codes inside the transmission chain will significantly increase the spectral efficiency and system reliability, which are especially critical factors in high-speed communication systems like 5G [14].

2.1.3 Modulation

Modulation is a central part in digital communication systems. It involves transforming the encoded codewords into a suitable format for transmitting over the physical channel. This is achieved by varying different features in a carrier signal, such as amplitude, frequency, or phase. This process provides efficient utilization of available resources and allows

transmitting signals over long distances while minimizing interference and loss. [39, 19]

Digital modulation converts discrete signals onto continuous waveforms, allowing effective transmission through analog channels. Selected modulation technique impacts the system's spectral efficiency, power efficiency, and robustness for channel disturbances. Popular digital modulation schemes include Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK), and Phase Shift Keying (PSK). In the context of this thesis, Quadrature Phase Shift Keying (QPSK) is employed due to its balance between spectral efficiency and robustness to noise. It is also worth noting that QPSK is one of the modulation formats specified for 5G New Radio (NR) [1].

Quadrature Phase Shift Keying (QPSK)

QPSK is a common digital modulation scheme that converts data by modulating the phase of a carrier signal. In QPSK, the signal assumes one of four distinct phase values. Each phase represents a unique pair of bits. This approach doubles the data rate compared to Binary Phase Shift Keying (BPSK) without the need for more bandwidth.

Mathematical Representation The modulated QPSK signal can be mathematically expressed as

$$s(t) = A \cos(2\pi f_c t + \phi_k), \quad (2.1)$$

where $A = \frac{1}{\sqrt{2}}$ is the normalized amplitude of the carrier signal, f_c is the carrier frequency, and ϕ_k is the phase shift corresponding to the k -th symbol, with $\phi_k \in \{\frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4}\}$.

Each phase shift ϕ_k represents a unique pair of bits. The bit pair and corresponding phase mappings are summarized in Table 2.1.

Bit Pair	Phase Shift (ϕ_k)
00	$\frac{\pi}{4}$
10	$\frac{3\pi}{4}$
11	$\frac{5\pi}{4}$
01	$\frac{7\pi}{4}$

Table 2.1. Bit pair to phase shift mapping in QPSK.

In-Phase and Quadrature Components QPSK modulation can also be represented using in-phase (I) and quadrature (Q) components as

$$I(t) = A \cos(\phi_k), \quad (2.2)$$

$$Q(t) = A \sin(\phi_k), \quad (2.3)$$

which allows the modulated signal to be expressed as

$$s(t) = I(t) \cos(2\pi f_c t) - Q(t) \sin(2\pi f_c t). \quad (2.4)$$

This expression points out that QPSK effectively combines two BPSK modulated signals, one the in-phase component and other on the quadrature component.

Constellation Diagram The constellation diagram of QPSK, shown in Figure 2.2, visualizes the four possible symbol states in the I - Q plane. Each point corresponds to a unique symbol defined by its I and Q components.

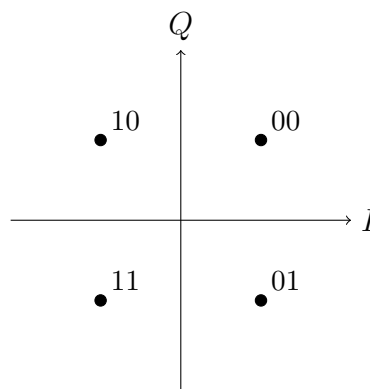


Figure 2.2. Constellation diagram of QPSK modulation showing the mapping of bit pairs to phase shifts.

2.1.4 Channel Models

The signal propagates from the transmitter to the receiver through the medium called communication channel. There various impairments are introduced which can deteriorate the signal, leading to incorrectly received data. These impairments are represented mathematically by channel models, making it possible to design and analyze systems that can combat their effects. [39]

Additive White Gaussian Noise (AWGN) Channel is a fundamental channel model, where the primary channel impairment is naturally occurring thermal noise. There it is modelled as a Gaussian random process which is added to the transmitted signal. [39]

With AWGN, it is assumed that the received signal y_i is the sum of the transmitted signal

x_i and a noise component n_i , which is modeled as a Gaussian random process. This relationship can be expressed as

$$y_i = x_i + n_i, \quad (2.5)$$

where y_i represents the received signal, x_i represents the transmitted signal, and n_i represents the additive noise, modeled as $n_i \sim \mathcal{N}(0, \sigma^2)$, a zero-mean Gaussian random process with variance σ^2 .

Fading Channels arise due to multipath propagation, a common phenomenon occurring in wireless communications. There several copies of transmitted signal arrive at the receiver with different amplitudes and delays. Common fading channel models include Rayleigh and Rician fading. Even though the AWGN channel model provides a foundational scenario that is well suitable for modelling and widely accepted as baseline, real-world systems often face multipath fading and time-varying conditions. For these scenarios, more complex channel models such as Rayleigh or Rician fading may provide a more realistic approach. While the AWGN is essential for early experimentations of code performance with different approaches, the iterative decoding and parity-check structures of LDPC codes are also suitable for more complex environments. Thus, evaluation and understanding of LDPC performance using AWGN channel is a beneficial step for extending their usage into more demanding channels.

The quality of the received signal is often expressed using metrics such as the energy per bit to noise power spectral density ratio E_b/N_0 and the signal-to-noise ratio (SNR). These metrics quantify the impact of noise on the transmitted signal, and are fundamental for evaluating the performance of error correcting codes like LDPC codes. Larger values of (E_b/N_0) and SNR indicate that the signal is stronger and less affected by noise.

The SNR is expressed as the ratio of the received signal power (P_s) to the noise power (P_n) in the system as

$$\text{SNR} = \frac{P_s}{P_n}. \quad (2.6)$$

However, SNR is usually expressed in decibels (dB) using the logarithmic scale as

$$\text{SNR (dB)} = 10 \log_{10} \left(\frac{P_s}{P_n} \right). \quad (2.7)$$

While SNR is a useful metric, it is often preferable to use E_b/N_0 , as it normalizes the signal power by the transmission rate R , providing a rate-compensated measure of performance. Moreover, E_b/N_0 is defined as the energy per information bit E_b divided by the noise power spectral density N_0 as

$$\frac{E_b}{N_0} = \frac{P}{2\sigma^2 R}, \quad (2.8)$$

where P is the average transmitted power, σ^2 is the noise variance and R is the transmission code rate [19].

Similar to SNR, E_b/N_0 is often expressed in decibels as

$$\frac{E_b}{N_0} \text{ (dB)} = 10 \log_{10} \left(\frac{x_n^2}{2\sigma^2 R} \right). \quad (2.9)$$

2.2 LDPC Code Theory

LDPC codes are a class of error-correcting codes which are designed to detect and correct errors occurring during the transmission of data and data storage. These codes were discovered by Robert Gallager in 1962 [17]. Initially, they received limited attention because of computational constraints and lack of suitable hardware. In recent years, however, they have received more research interest due to advancements in computing making them more feasible. LDPC codes can achieve performance close to the Shannon limit with carefully designed code structures and efficient iterative decoding algorithms, especially with large block sizes [32].

LDPC codes are defined using a sparse parity-check matrix \mathbf{H} which contains far fewer ones compared to zeros. This sparsity feature is vital as it allows iterative algorithms to rapidly converge and consume less computational power, enabling a more efficient decoding process.

LDPC codes have become an important part for error correction in modern communication systems because of their efficiency and near-optimal error-correction performance. Their theoretical basis combines graph theory, coding theory and probabilistic approaches which makes them functional for applications ranging from digital communication to data storage. Basics of error-correcting codes provides context for specific structure and exclusive features of LDPC codes, which rely on a sparse parity-check matrix and its graphical representation streamlining the decoding operation. This section delves into the essential concepts, code construction, and different decoding algorithms of LDPC codes, starting with an overview of error-correcting code fundamentals.

2.2.1 Overview of Error-Correcting Codes

Error-correcting codes (ECC) are essential for allowing reliable data transmission over noisy channels. When data is transferred over any channel (e.g., optical fibers, wireless networks or storage systems), it may get corrupted by interference, noise, or other distortions. That would lead to possible errors in the received data. ECCs are designed to handle this problem by encoding the original data into a codeword. That codeword contains redundancy, which allows the receiver to detect and, in many instances, correct

errors without the need for retransmission. This process enables efficient usage of bandwidth, minimizes delays and reduces data loss, making ECC vital to the reliability and performance of modern digital communication systems.

Foundational work in the area of information theory by Claude Shannon introduced the idea of channel capacity. It defines the maximum rate of errorless transmission over a noisy channel. ECCs are crucial for making it possible to approach this theoretical limit (which is often called the Shannon limit) by correcting errors at the limits of channel capacity [45]. This invention was the starting point of ECC research, which led to the work on various codes with different computational complexities and error-correction capabilities. These codes include, for example, convolutional codes, Hamming codes, Reed-Solomon codes, and modern codes such as LDPC and turbo codes [12].

LDPC codes are especially beneficial in scenarios which require both low latency and high reliability compared to other ECCs such as turbo or Reed-Solomon codes. These attributes are possible because of their structured sparsity which enables iterative decoding techniques (such as belief propagation or minsum) that combine computational efficiency and error-correction performance [32]. LDPC codes are widely integrated in standards which require high-speed data transmission, highlighting their importance in modern communication systems. These standards include, for example, WiFi [23], 5G NR [2] and DVB-S2 [47].

LDPC codes are part of codes known as “capacity-approaching codes” which means their performance is close to the Shannon limit when having optimal decoding conditions. This feature makes them a well-suited choice for applications which need high throughput and low error rates, which is why LDPC codes are relevant for researching machine learning-assisted decoding techniques. With ML integration, it is possible to improve accuracy and decoding speed even more.

2.2.2 Code Structure and Graph Representation

Parity-check matrix \mathbf{H} is a binary matrix that defines the linear constraints each codeword must satisfy, ensuring parity-check equations are met. The sparsity of \mathbf{H} is a crucial aspect, making decoding algorithms work efficiently. In \mathbf{H} , each row represents a single parity-check equation that must be fulfilled. The dimensions of \mathbf{H} are based on the codeword length and number of information bits in the codeword. Denote the length of the codeword as n and the number of information source bits in each codeword as k . Then \mathbf{H} is a matrix containing $m = (n - k)$ rows and n columns.

An example of a parity-check matrix is shown in Figure 2.3. This matrix illustrates the sparse structure of LDPC codes, where the majority of entries are 0, and only a few are 1. This sparsity is critical for the efficient decoding algorithm implementation.

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

Figure 2.3. Example of a parity-check matrix for an LDPC code. Each row corresponds to a parity-check equation, and each column represents a codeword bit.

For a particular codeword to be valid, it must fulfill all the parity-check equations (rows in the matrix) at the same time. Mathematically, a valid LDPC codeword \mathbf{c} satisfies

$$\mathbf{H}\mathbf{c}^T = \mathbf{0}, \quad (2.10)$$

where $\mathbf{0}$ is a zero vector.

Another definition closely related to the dimensions of \mathbf{H} is the code rate, which can be expressed as

$$R = \frac{n - m}{n}. \quad (2.11)$$

The code rate can be increased by having fewer parity check equations m relative to the codeword length n , which usually leads to better performance. However, this potentially increases the decoding complexity at the same time.

Tanner graphs are a crucial tool for representing LDPC codes and providing a graphical framework that enhances the understanding of these codes [48]. A Tanner graph is constructed from two different types of nodes: variable nodes and check nodes. Variable nodes represent the codeword bits, whereas check nodes represent the parity-check constraints imposed on these bits. Thus, there is a relationship between the parity-check matrix and Tanner graph. Specifically, the number of check nodes in the graph corresponds to the number of rows in \mathbf{H} , and the number of variable nodes corresponds to the number of columns in \mathbf{H} . Tanner graphs have a bipartite nature which means that vertices are divided into two disjoint sets with no edge connections between nodes in the same set [3]. Thus, in the context of LDPC codes, there aren't any edge connections between variable node and variable node, nor are there any edges connecting check nodes to other check nodes.

Edges in the Tanner graph are defined by the entries of matrix \mathbf{H} . Specifically, there is an edge between check node i and variable node j if the element \mathbf{H}_{ij} is equal to one. Given the fact that \mathbf{H} is a sparse matrix, Tanner graphs related to LDPC codes are also sparse, having relatively few edges compared to the total number of nodes. Figure 2.4 illustrates the Tanner graph corresponding to the parity-check matrix in Figure 2.3.

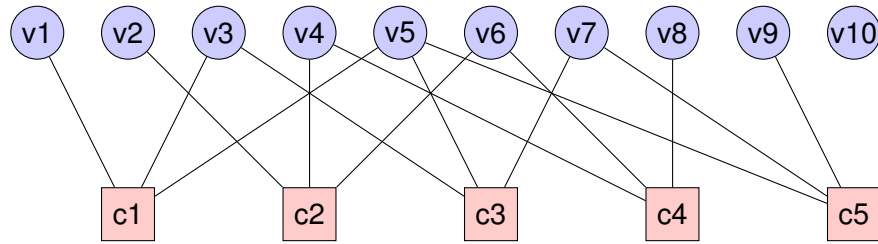


Figure 2.4. Tanner graph representation of the parity-check matrix in Figure 2.3. Variable nodes ($v1-v10$) represent codeword bits, and check nodes ($c1-c5$) represent parity-check equations.

The visual representation of Tanner graphs helps in the understanding of code structure and aids in the decoding algorithm development. During the decoding process, messages are passed along the edges. This message passing forms the basis of iterative decoding algorithms such as belief propagation.

The presence of cycles is an important aspect to consider when designing LDPC codes. Especially, short cycles can reduce the independence of messages during belief propagation, hindering convergence and degrading decoding performance. Therefore, minimizing the number of such cycles is an important aspect when constructing effective LDPC codes. [33]

2.2.3 LDPC Code Construction and Variants

The design of LDPC codes involves construction of the parity-check matrix \mathbf{H} which optimizes decoding performance while simultaneously balancing complexity. LDPC codes are divided into two main branches according to their construction: random and structured codes. Both approaches have their own advantages, and many LDPC code variants have been designed to fulfill unique application scenarios and performance requirements.

Random LDPC codes are constructed by randomly filling the parity-check matrix \mathbf{H} with ones under particular constraints to establish sparsity. The goal is to achieve a specific degree distribution, meaning each row (parity-check equation) and each column (codeword bit) contain some defined number of ones. This degree distribution is the key feature when it comes to the error correction capability of the code.

In [29], it is shown that codes with irregular degree distribution have better performance as they provide improved error-correction efficiency and better weight distribution compared to regular codes. Irregular degree distribution means that some rows and columns have a higher number of ones than others.

In [22], Hu et al. present PEG (Progressive Edge Growth) algorithm which is often used as guidance for design of random LDPC codes. The aim of PEG is to generate codes having a large girth, or the length of the shortest cycle in the corresponding Tanner graph.

Short cycles should be avoided in LDPC code design as they create dependencies in the iterative message-passing algorithms, thus impairing the decoding performance. So, the PEG algorithm seeks to enhance the performance of the code by maximizing cycle lengths.

Randomly constructed LDPC codes may be useful for simulation and research purposes as they are easy to create and allow flexibility in degree distributions. As a drawback, they may lack the structured regularity needed for simpler hardware circuit design. This unpredictability and irregularity can make them less efficient for hardware usage in real-world applications.

As opposed, structured LDPC codes contain specifically designed patterns in the parity-check matrix, which makes them useful for real-world applications where predictability and efficiency are vital. In structured codes, algebraic methods, like cyclic codes or combinatorial block designs, are often used to generate \mathbf{H} [52]. This structured generation leads into highly regular matrices which supports parallel processing during decoding and utilizes efficient hardware implementation.

Quasi-Cyclic (QC) codes are one of the most important classes of structured LDPC codes. These codes construct the full parity-check matrix by aligning cyclically shifted versions of a smaller base matrix. This construction leads to compact storage possibilities and efficient encoding and decoding algorithms. Thus, QC-LDPC codes are a common choice in standards like 5G and WiFi. Additionally, hardware implementation becomes simpler due to cyclic structure because every shift in the code can be implemented with a shift register. [16, 38]

Protograph-based LDPC codes are another alternative which relies on structured code construction. Protographs are smaller graphs used as building blocks when constructing larger Tanner graphs. Lifting is a technique used for expanding the protograph while retaining its preferable properties, like structured sparsity and high girth [50]. These protograph-based LDPC codes are widely utilized in modern communication systems, such as 5G NR and satellite standards (DVB-S2).

2.2.4 Decoding Algorithms

LDPC decoding relies on iterative message passing algorithms. Broadly, these algorithms can be categorized into two main branches: hard decision and soft decision decoding.

In hard decision decoding, the received signal is quantized into binary values before the decoding process begins, meaning that the messages passed between nodes contain only binary bit values (0s and 1s). This quantization simplifies the decoding process and reduces computational complexity per iteration. However, this approach often leads to suboptimal performance, especially in low SNR scenarios. A notable example of a hard

decision decoding algorithm is the Bit-Flipping Algorithm.

Soft decision decoding utilizes probabilistic information derived from the received signal, enabling more informed decision-making about the transmitted bits. Unlike hard decision decoding where messages are binary, soft decision decoding algorithms represent messages using conditional probabilities. This allows more accurate recovery of the original message. [5]

The Log-Likelihood Ratio (LLR) is often used in soft decision decoding to quantify the reliability of bit estimates. The LLR provides an indication of how likely the bit is 0 or 1, given the received signal. Soft decision decoding algorithms, such as Belief Propagation and its variants, generally offer far better performance across a range of SNR levels in the log domain [24]. This thesis focuses on soft decision decoding approaches due to their better performance characteristics.

The Belief Propagation (BP) algorithm, also known as the Sum-Product algorithm, operates on a Tanner graph representing the parity-check matrix. This algorithm proceeds with iterative steps, where each node exchanges messages between its neighbors. The exchanged messages are log-likelihood ratios (LLRs), which represent the logarithmic ratio of the probabilities that a bit is 0 or 1, given the received signal. The magnitude of the LLR reflects the reliability of the message whereas the sign indicates the most probable bit value. These channel LLRs are also the input for the LDPC decoder. The channel LLR for a variable node v_i is given by

$$\mathbf{b}_{v_i} = \log \frac{P(y_i | x_i = 0)}{P(y_i | x_i = 1)}, \quad (2.12)$$

where \mathbf{b}_{v_i} is the input to the LDPC decoder, y_i represents the received noisy signal, and x_i denotes the transmitted bit. [7] [28]

For each connected variable node v_i and check node c_j , the message update function from variable nodes to check nodes is

$$x_{v_i \rightarrow c_j}^{(l)} = b_{v_i} + \sum_{\substack{c_p \rightarrow v_i \\ p \in C(i) \setminus j}} x_{c_p \rightarrow v_i}^{(l-1)} \quad (2.13)$$

where $p \in C(i) \setminus j$ denotes the neighbors of v_i excluding c_j . In other words, each variable node sends all the information it has, except the information check node possesses already. The function from check node to variable node is

$$x_{c_j \rightarrow v_i}^{(l)} = 2 \cdot \tanh^{-1} \left(\prod_{\substack{v_q \rightarrow c_j \\ q \in V(j) \setminus i}} \tanh \left(\frac{x_{v_q \rightarrow c_j}^{(l)}}{2} \right) \right) \quad (2.14)$$

where $\mathbf{q} \in \mathbf{V}(\mathbf{j}) \setminus \mathbf{i}$ denotes the neighbors of \mathbf{c}_j excluding \mathbf{v}_i . Similarly, each check node sends all the extrinsic information it has to neighbor variable nodes. It should be noted that $\tanh(x)$, the hyperbolic tangent function, is defined as $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, and its inverse, $\tanh^{-1}(x)$, represents the inverse hyperbolic tangent function, distinct from $1/\tanh(x)$.

Then the belief propagation algorithm iteratively keeps updating messages until the stopping criterion is met. The decoding process concludes when either all parity-check equations are fulfilled or the maximum number of iterations is achieved. In the end, the output is calculated as

$$x_{v_i}^{(l)} = b_{v_i} + \sum_{\substack{c_p \rightarrow v_i \\ p \in C(i)}} x_{c_p \rightarrow v_i}^{(l-1)} \quad (2.15)$$

The convergence of the BP algorithm depends on the LDPC code structure and the quality of received signal. Variants of BP have been developed where the message update rules are modified. These variants try to balance the trade off between performance and computational complexity.

The min-sum (MS) algorithm is a less complex decoding method as opposed to BP. While the processing of check nodes is identical in both cases, the difference lies in how variable nodes are handled. To avoid the costly computation of the tanh-function, its simple approximation is defined as

$$x_{c_j \rightarrow v_i}^{(l)} = \left(\prod_{\substack{v_q \rightarrow c_j \\ q \in \mathbf{v}(j) \setminus i}} \text{sgn} \left(x_{v_q \rightarrow c_j}^{(l)} \right) \right) \min_{\substack{v_q \rightarrow c_j \\ q \in \mathbf{v}(j) \setminus i}} |x_{v_q \rightarrow c_j}^{(l)}|. \quad (2.16)$$

While MS is computationally less expensive, it introduces some performance loss compared to BP. This trade-off between computational complexity and decoding performance renders it beneficial for use cases with limited resources.

Recently, the usage of neural networks have been explored to enhance LDPC decoding [44]. Neural network based decoders leverage machine learning techniques to model decoding algorithms. These approaches can learn to approximate message passing rules, and have shown promising results in improving decoding performance. For example, Reinforcement Learning and Graph Neural Networks are being studied to further enhance the decoding process.

2.2.5 Performance Metrics and Evaluation

It is crucial to evaluate the performance of LDPC decoders to get an idea of their effectiveness and efficiency in correcting errors. With performance metrics, it is possible to get a quantitative basis for comparing decoding algorithms, such as ML-assisted and conventional approaches. This section introduces the key performance metrics and evaluation methods for LDPC decoders and discusses their importance to this thesis.

One of the most popular metrics for evaluating LDPC decoders is the Bit Error Rate (BER). BER indicates the fraction of incorrectly decoded bits relative to the total number of received bits [9]. BER is defined as

$$\text{BER} = \frac{N_b^{\text{err}}}{N_b^{\text{total}}}, \quad (2.17)$$

where N_b^{err} represents the number of bits received in error, and N_b^{total} represents the total number of bits transmitted. This metric is especially valuable in use cases where the accuracy of individual bits is crucial, as it shows an insight into the reliability of the decoder at the bit level. Mathematically, BER is defined as the ratio of incorrect bits to the total transmitted bits. For reviewing the relationship between BER and channel conditions, it is usually plotted as a function of E_b/N_0 or Signal-to-Noise Ratio (SNR). The resulting curves are typical visualizations in the area. They show clearly the error correction performance of the decoder under different levels of noise. For LDPC decoders, these curves offer useful insights into the connections between decoding algorithms, channel characteristics and code parameters. In this thesis, BER is used as one of the key metrics for evaluating the performance of both classical and ML-assisted decoders.

In addition to BER, the Block Error Rate (BLER) is another crucial metric when evaluating LDPC decoders. As BER focuses on individual bit accuracy, BLER reviews the prevalence of errors at the block level, determining whether entire codeword blocks are decoded correctly. BLER is defined as

$$\text{BLER} = \frac{N_{\text{blocks}}^{\text{err}}}{N_{\text{blocks}}^{\text{total}}}, \quad (2.18)$$

where $N_{\text{blocks}}^{\text{err}}$ represents the number of codeword blocks received in error, and $N_{\text{blocks}}^{\text{total}}$ represents the total number of codeword blocks transmitted. BLER is especially important in communication systems where the correctness of whole data blocks is critical, such as multimedia transmission or packet-switched networks. If even a single bit inside the block is decoded incorrectly, the codeword block is considered to be erroneous. As such, the BLER metric is expressed as the ratio of incorrectly decoded blocks to the total number of transmitted blocks. Similarly to BER, BLER is also usually analyzed as a function of E_b/N_0 or SNR, thus showing a block-level perspective on decoder performance. This thesis

emphasizes BLER as a primary metric, as it aligns closely with real-world prerequisites where error-free block recovery is paramount.

Further, an emerging metric for LDPC decoder performance evaluation is the Bitwise Mutual Information (BMI). BMI indicates the mutual information between the transmitted and received bits after decoding, showing a statistical viewpoint on the ability of the decoder to retain information integrity [13]. BMI is defined as

$$\text{BMI} = \sum_{x \in \{0,1\}} \sum_{y \in \{0,1\}} P(x, y) \log_2 \frac{P(x, y)}{P(x)P(y)}, \quad (2.19)$$

where $P(x, y)$ is the joint probability of the transmitted bit x and the decoded bit y , and $P(x)$ and $P(y)$ are the marginal probabilities of x and y , respectively. BMI is especially valuable as it measures the information-theoretic performance of the decoder, focusing on the quality of the signal reconstruction. Higher BMI values indicate better preservation of mutual information, reflecting the efficacy of the decoder under noise and interference. As opposed to BER and BLER, BMI utilized probabilistic dependencies providing a broad perspective on decoder performance.

Another important aspect to consider is decoding latency. It measures the time to decode a single codeword. Latency is especially critical in real-time applications where low-latency performance is important, such as 5G wireless communication or autonomous vehicle systems [27]. The latency of a decoding algorithm is affected by many causes, including the number of decoding iterations required, the computational complexity of each iteration, and the available hardware used in application. Iterative algorithms, such as BP or MS, often require many passes before error converges, increasing latency. Additionally, computationally expensive operations, like hyperbolic functions or matrix multiplications, negatively impact latency. Hardware constraints, such as processing speed, have also a major impact for latency in practical use cases.

Computational complexity must also be considered when evaluating LDPC decoders [8]. Complexity is affected by the number of operations per iteration and the memory needed for storing intermediate values. Traditional decoding algorithms like belief propagation include complex operations, such as hyperbolic tangent, which can be computationally expensive. For that reason, simplified algorithms like min-sum have been introduced to reduce complexity by approximating certain functions. As a drawback, this often comes at the cost of reduced decoding performance.

It is crucial to perform experiments in well-defined environments to evaluate LDPC decoders correctly. The Additive White Gaussian Noise (AWGN) channel is one of the most commonly used channel models in the research. This channel model corrupts the transmitted signal with Gaussian-distributed noise which is independent and identically distributed across the signal [39, 40]. The straightforwardness of the AWGN model makes it

a universally accepted baseline for performance evaluation. It allows researchers to well isolate the decoder's error-correction performance from other channel impacts. Aforementioned metrics such as BER and BLER are analyzed under AWGN channel for establishing standard performance benchmarks which provide a basis for comparing different decoding approaches and analyzing their relative strengths.

There are also practical considerations playing a significant role in the evaluation process. One important consideration is the stopping criterion in iterative decoding algorithms such as the satisfaction of all parity-check equations or a maximum number of iterations [36]. This selection directly impacts metrics like BLER and latency. Another practical consideration is scalability. It refers to its ability to work persistently across different code rates and lengths. Scalability is especially relevant for the design of adding ML features as the approach should work well with varying code parameters without major performance degradation.

In this thesis, the performance of ML-assisted decoder is compared against classical approaches based on a comprehensive set of metrics including BER, BLER and latency. Also, computational complexity is considered. By reviewing these metrics in detail, this work aims to provide a comprehensive understanding of the benefits and drawbacks of machine learning-assisted LDPC decoding. These evaluations are conducted under regularized conditions using AWGN channel to ensure reproducibility and consistency. The results shown in later parts highlight the potential of ML-assisted decoders to improve error-correction performance, reduce latency and consider computational practicalities, offering beneficial observations into the future of LDPC decoding.

2.2.6 LDPC Codes in 5G New Radio (NR)

The fifth generation mobile communication standard 5G New Radio (NR) uses LDPC codes as the key channel coding approach, being a clear milestone in the usage of LDPC codes in cellular systems. 3GPP TS 38.212 [2] is the standard defining the coding and multiplexing to physical channels in 5G NR. This subsection explores the 5G NR related specific implementation for LDPC codes, highlighting their design considerations, structure, and the reasons for their usage in the standard.

When designing the 5G NR, the 3rd Generation Partnership Project (3GPP) was looking for a channel coding scheme that could support the requirements for enhanced mobile broadband (eMBB), ultra-reliable low-latency communications (URLLC), and massive machine-type communications (mMTC). LDPC codes support excellent error-correction performance and high throughput capabilities. Additionally, LDPC codes are an attractive choice from hardware context. They support lower complexity than the Turbo codes used in LTE when used with higher code rates. [14]

The design of LDPC codes in 5G NR are specifically designed to balance the performance and implementation complexity. They offer a protograph-based approach, allowing scalability and flexibility in code construction. The final parity check matrices are constructed from two base graphs, Base Graph 1 (BG1) and Base Graph (BG2), which are templates defined in standard.

Base Graph 1 (BG1) is utilized for larger block sizes and higher code rates, making it particularly suitable for high-throughput eMBB scenarios. In contrast, Base Graph 2 (BG2) is optimized for smaller block sizes and lower code rates, which makes it ideal for URLLC applications requiring high reliability and low latency. BG2 facilitates efficient decoding even at lower code rates.

The size of these base graphs is increased through a lifting process to get the final parity check matrix. There are 51 different lifting sizes to support a wide range of block sizes. The selection of the lifting factor is done according to a desired code length. In the lifting process, each element in the base graph is replaced with a circularly shifted identity matrix or a zero matrix. These replacements lead to a quasi-cyclic (QC) LDPC code which also allows efficient hardware implementation.

2.3 Machine Learning Foundations

Machine learning (ML) has validated its potential for solving difficult, pattern-based tasks, demonstrating its effectiveness for enhancing traditional decoding processes in communication systems. Unlike traditional approaches, which are determined by predefined rules and iterative processes, ML-based approaches can learn patterns in the data, allowing major gains in error correction performance. This section delves into the foundational principles of ML as used in decoding, including its relevance, neural network architectures used, the supervised learning methods used in training, and the key role of loss functions and optimization strategies.

2.3.1 Overview of Machine Learning

Machine learning tries to solve the problem of how to create computer programs that perform better at some task using past experiences. In other words, it allows computers to perform without explicit instructions with the reliance of patterns and inference from data. ML has proven to have reasonable value in a wide scope of application areas. For example, data mining problems can contain huge databases where valuable implicit conclusions can be made with the help of ML. Also, ML can help with poorly understood domains where humans don't have the knowledge required to create proper algorithms, such as face recognition tasks from images. Additionally, some applications may have to adapt when conditions change. There ML can help, for example, manufacturing process

managers or content creators to observe patterns which would be otherwise difficult to figure out. Machine learning (ML) combines ideas from many different disciplines, such as artificial intelligence, information theory, statistics, probability theory and neurobiology. [34]

Machine learning algorithms are usually divided into three different main categories: supervised learning, unsupervised learning, and reinforcement learning [43]. Regarding this thesis, supervised learning is the most suitable approach for the problem, and thus it's used.

2.3.2 Neural Network Architectures

Neural networks are computational models which are constructed from interconnected processing units termed as neurons or nodes. The structure of neural networks are inspired by the biological structures of the human brain. The networks try to recognize complex relationships and patterns from the data. It is done via adjustments of weights and biases of the connections between neurons through the training process. Several different neural network architectures have been developed for different sorts of problems, each having unique benefits and applications. [20]

In this thesis, the used architecture resembles the Feedforward Neural Network (FNN) architecture. FNNs, also known as Multi-Layer Perceptrons, are the simplest type of neural network. In FNNs data flows from the input layer through one or more hidden layers to the output layer unidirectionally, not having any loops or cycles [6]. Usually each neuron in a particular layer is connected to each neuron in the following layer, composing a fully connected network.

Feedforward neural networks are usually well suited for tasks where static mappings from inputs to outputs are needed, such as regression or classification [18]. During the training process, the network's weights are being updated using algorithms like backpropagation targeting to minimize a predefined loss function [42]. Although having a simple structure, FNNs can model a wide variety of nonlinear relationships with enough hidden layers and neurons. An FNN consists of an input layer, one or more hidden layers, and an output layer. Inside a layer, each neuron calculates a weighted sum of its inputs, adds a bias term, and applies an activation function for introducing nonlinearity into the model [20]. Mathematically, the output of a neuron i in layer l can be expressed as

$$a_i^{(l)} = \phi \left(\sum_{j=1}^{n_{l-1}} w_{ij}^{(l)} a_j^{(l-1)} + b_i^{(l)} \right), \quad (2.20)$$

where $a_i^{(l)}$ represents the output of neuron i in layer l , $w_{ij}^{(l)}$ denotes the weight connecting neuron j in layer $l - 1$ to neuron i in layer l , $b_i^{(l)}$ is the bias term for neuron i in layer l , ϕ

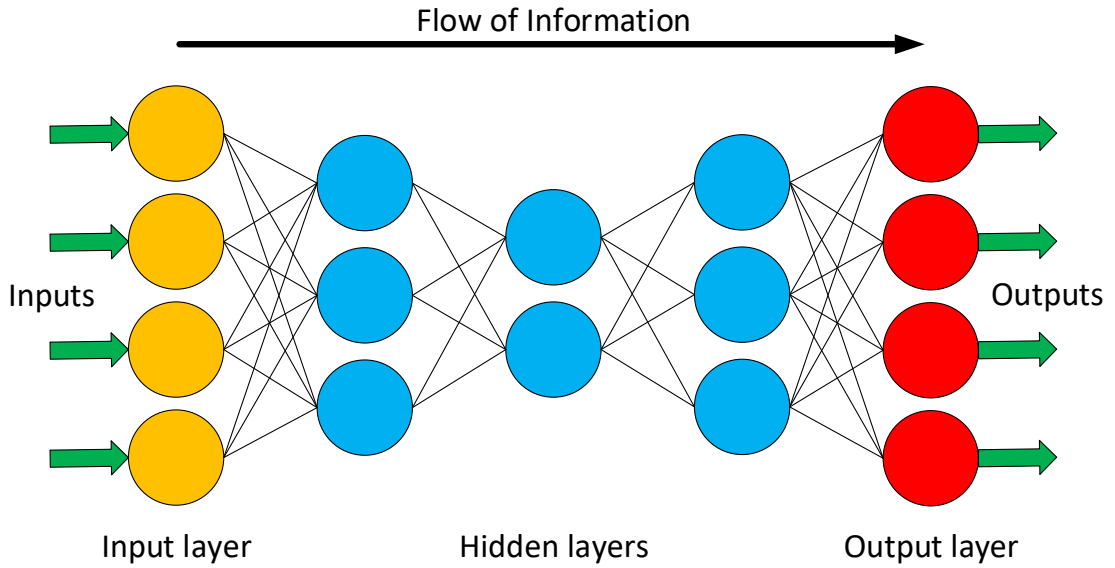


Figure 2.5. Illustration of a feedforward neural network (FNN) architecture, representing the flow of information from the input layer through hidden layers to the output layer. This structure is foundational for mapping inputs to outputs in supervised learning tasks.

is the activation function, and n_{l-1} is the number of neurons in layer $l - 1$.

Activation functions are essential in FNNs because they introduce nonlinearity, enabling the network to learn complex mappings from inputs to outputs [4]. Common activation functions include the sigmoid function, the hyperbolic tangent function, and the Rectified Linear Unit (ReLU). The sigmoid function outputs a value between 0 and 1, defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \quad (2.21)$$

The hyperbolic tangent function, which outputs values in the range of -1 to 1, is given by

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \quad (2.22)$$

Finally, the Rectified Linear Unit (ReLU), which introduces sparsity and alleviates the vanishing gradient problem, is defined as

$$\text{ReLU}(x) = \max(0, x). \quad (2.23)$$

During the training of FNNs, weights and biases are adjusted to minimize a loss function measuring the difference of network's predictions and the true outputs. Often used method for training is the backpropagation algorithm [42]. It computes gradients of the loss function with respect to the network parameters. Then, these gradient values are used to update the parameter values using optimization algorithms, such as stochastic

gradient descent (SGD) or some of its variants [26].

In the backpropagation algorithm, the error from the output layer propagates to the earlier layers, and the chain rule of calculus is applied. Usually, the weight updates are done as

$$w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial w_{ij}^{(l)}}, \quad (2.24)$$

where η denotes the learning rate, and \mathcal{L} represents the loss function.

Overfitting is a major challenge when training FNNs. There the network learns the training data too well, leading the model to fail to generalize to new data [4]. In other words, the difference between test error and training error is too large. Thus, it is very important to validate the performance of the model using a diverse set of test data.

2.3.3 Supervised Learning and Training Process

Supervised learning is an essential part of machine learning, where the model learns mappings from inputs to outputs using labeled datasets [43]. There a set of input-output pairs are used during the training process with the model, and the objective is that the model would infer a function that can correctly predict the output for new, unseen inputs. This approach is crucial for tasks such as regression, classification, and pattern recognition in many different fields.

The training phase involves several key steps, starting from data collection and preprocessing. High-quality data is essential for successful training because the model's ability to create generalizations depends heavily on the applicability of training data. Typically the dataset is divided into three subsets: training, test and validation data sets. In training, a training set is used to optimize the weights of the network. Validation data is used when tuning hyperparameters and preventing overfitting. Whereas, the test set provides an unbiased evaluation at the end for the final model. However, this kind of splitting is not relevant in the context of this thesis as all the data used in training and model evaluation is randomly generated bits "on the fly".

During training, the neural network optimizes its parameters, weights and biases, by trying to minimize a selected loss function which measures the difference of true outputs compared to predicted outputs [4]. The choice of loss function is very important because it has a major impact for learning dynamics and, in the end, for model's performance. Mean squared error (MSE) is a commonly used choice for regression tasks. For classification problems, functions like the cross-entropy loss are preferred.

In ML, optimization algorithms are used to minimize the loss function effectively. Stochastic Gradient Descent (SGD) is a key optimization algorithm that updates the parameters in the opposite direction of the gradient of the loss function with respect to the parameters.

Variants of SGD have been developed, such as Adam [26], RMSProp [51] and Adagrad [15]. These approaches use adaptive learning rates and momentum terms to improve performance and accelerate convergence. They are especially useful in deep networks. In this thesis, the Adam optimization algorithm is used.

A crucial part of training is hyperparameter tuning where optimal settings of parameters, that are not learned directly during training, are searched [4]. These hyperparameters include learning rate, batch size, number of epochs and network architecture (layer counts and number of neurons per layer). Several techniques can be used, such as manual optimization, grid search or random search. In the context of this thesis, hyperparameter optimization is done using grid search. There a small finite set of values to explore are selected. Then, the model is trained for each combination of individual hyperparameters. In the application in this thesis, there aren't very many different parameters so that makes grid search a well suited approach.

In conclusion, the supervised learning and training process of neural networks is a systematic process including data preparation, model optimization, hyperparameter tuning, and performance evaluation. Every step has its own critical role in development of models which not only perform well with training data but also generalize effectively to new, unseen data. This process sets up the development of robust machine learning models in a wide range of use cases, from natural language processing to image recognition and beyond.

2.3.4 Loss Functions and Optimization

The selection of loss function is an important consideration in the design of ML-based decoders because it directly affects the model's learning process. In decoding problems, binary cross-entropy (BCE) is commonly choiced because of its suitability for binary classification problems. BCE measures the error between the ground truth and predicted probabilities for each bit, leading the network to minimize bit-wise prediction errors.

Optimization algorithms have also an important role in training ML-based decoders. Stochastic gradient descent (SGD) and its variants are widely used for loss minimization. One of the variants in particular, Adam, has gained popularity for its adaptive learning rate capabilities and robust convergence performance, making it suitable for difficult, high-dimensional optimization tasks [26].

Hyperparameter tuning is another crucial aspect of model training. It requires conducting careful experiments to determine optimal selections for learning rate and other parameters. Lastly, computational considerations must also be addressed in parameter selection and other optimization. Memory efficiency and hardware constraints must be considered to make sure algorithmic implementations are feasible in real-time communication sys-

tems.

By combining these foundational elements, neural network design, supervised learning methodologies, and optimization strategies, ML-assisted decoding has shown significant performance in addressing challenges of classical LDPC decoding. These approaches establish the groundwork for hybrid decoding techniques which combine adaptability of ML with strengths of traditional algorithms.

2.4 Integration of Machine Learning and LDPC Decoding

Integrating machine learning techniques into LDPC decoding is showing a promising way in the pursuit of better computational efficiency and error-correction capabilities. With the combination of both traditional decoding algorithms and ML, hybrid decoders are aiming to beat the limitations in traditional methods. This section presents the motivations for ML-assisted decoding, examines different hybrid decoder architectures, and explains the expected benefits and challenges related to this integration.

2.4.1 Motivation for ML-Assisted LDPC Decoding

The integration of machine learning (ML) with LDPC decoding addresses fundamental limitations in traditional decoding algorithms. Classical approaches such as Belief Propagation (BP) and Min-Sum (MS) represent opposing ends of the performance-complexity spectrum: BP offers excellent error-correction performance but suffers from significant computational complexity due to its hyperbolic tangent functions, while MS reduces computational burden at the cost of suboptimal performance [17, 55, 8]. This trade-off poses significant challenges in contemporary applications such as 5G communications and Internet of Things (IoT) devices, where strict power consumption and latency requirements necessitate decoders that balance efficiency with performance [27]. In particular, the computational complexity of BP makes it impractical in many resource-constrained environments [28].

Machine learning offers a promising alternative by enabling models to learn patterns between transmitted and received bits directly from noisy codewords. Several pioneering approaches have demonstrated the potential of this paradigm. Nachmani et al. [35] enhanced BP by introducing trainable weights within message-passing, improving performance without substantially increasing complexity. Lugosch et al. [30] extended these ideas by adapting the offset min-sum algorithm with additive weights instead of multiplicative ones.

The advantages of ML-assisted decoders extend beyond computational efficiency. These approaches can adapt to varying channel conditions and code structures without comprehensive reconfiguration, making them particularly valuable in dynamic environments

where channel circumstances change rapidly, such as in mobile communication systems [37]. The generalization capabilities of ML models enable decoders to maintain high performance across diverse operating conditions. Moreover, ML techniques can approximate optimal decoders like BP with only a fraction of the computational cost. Recent developments in hardware accelerators for ML applications, including Graphics Processing Units (GPUs) and specialized AI chips, provide infrastructure that supports the deployment of ML-assisted decoders in real-world systems [25], reducing barriers to integrating complex ML models into practical telecommunications applications.

Training data generation for these models requires realistic simulation of channel conditions, typically involving random codeword generation and corruption through channel models such as Additive White Gaussian Noise (AWGN). A critical challenge is determining appropriate noise power: sufficient noise is necessary for the ML approach to learn noise behavior, but excessive noise makes codeword restoration impossible, even with ML assistance. The training process itself involves significant computational effort, particularly with large datasets. Key hyperparameters—including batch size, learning rate, and training iterations—must be carefully selected to optimize the training phase and ensure effective decoder performance in practical applications.

2.4.2 Hybrid Decoding Architectures

Hybrid decoding architectures merge machine learning models with traditional decoding to capture the strengths of both. Several approaches have been researched in the literature, each utilizing ML in different ways during the decoding process. In the context of LDPC decoding, different neural network architectures have been studied, such as feedforward neural networks (FNN) [28], recurrent neural networks (RNN) [41] or convolutional neural networks (CNN) [49].

Feedforward neural networks are usually well suited for tasks where static mappings from inputs to outputs are needed. For example, learnable weights implemented into the min-sum algorithm can fine-tune the relative importance of messages passed between nodes, thus improving the model's performance under varying channel conditions.

One notable approach is the usage of neural networks for strengthening the iterative message-passing process within decoders. With this approach, neural networks are used to adjust the messages passed between nodes in the Tanner graph, essentially learning optimal message update rules. This method preserves the structure of traditional decoders while introducing learnable parameters that will be optimized during training.

In [35], a neural BP decoder is proposed, where multiplicative weights are added to the edges of the Tanner graph. These weights are optimized through training, allowing the decoder to beat the BER performance of standard BP decoding without significantly

adding computational complexity. This work highlights that clear performance gains can be achieved with only slight modifications when guided by machine learning. However, they use high density parity check (HDPC) codes, so the results are not directly comparable with this thesis.

3. RESEARCH METHODOLOGY

3.1 Methodological Justification

The methodological choices in this thesis are driven by the need to bridge the gap between the high-performance but computationally intensive belief-propagation (BP) algorithm and the hardware-friendly yet less accurate min-sum (MS) rule for decoding 5G NR LDPC codes. By embedding machine-learning techniques directly into the message-passing schedule, the work seeks to retain the low-complexity arithmetic of MS while recovering a substantial portion of the coding gain traditionally associated with BP.

To achieve this balance, a hybrid decoder is adopted in which multiplicative (normalization) and additive (offset) weights are attached to the edges of the Tanner graph. These weights, denoted $\alpha_n, \beta_n, \gamma_n$ and $\alpha_o, \beta_o, \gamma_o$, are trained to compensate for the approximations inherent in MS, building on the neural belief-propagation idea of Nachmani *et al.* [35] and its min-sum adaptations by Li *et al.* [28] and Wu *et al.* [54]. Because the underlying Tanner graph and update schedule remain intact, the resulting architecture preserves parallelism and memory locality, making it suitable for practical hardware implementation.

Supervised learning is chosen as the optimisation paradigm since ground-truth code-words are known a priori. The loss function is a weighted binary cross-entropy in which separate coefficients ($\omega_{\text{info}}, \omega_{\text{parity}}$) allow the decoder to prioritise complete block recovery without sacrificing bit-level accuracy. A multiloss strategy accumulates the loss after each decoding iteration, an approach that has proved effective in earlier studies of neural BP variants [35]. Parameter updates are performed with the Adam optimiser, which offers rapid convergence under the non-convex conditions typical of deep-learning problems.

Training and evaluation data are generated on-the-fly with the Sionna physical-layer library, using the 5G LDPC encoder, QPSK modulation, and an additive-white-Gaussian-noise channel. The training signal-to-noise ratio is fixed to a single E_b/N_0 level at which an untrained BP decoder yields a block-error rate of approximately 0.2 (here 3.0 dB for Basegraph 1 and 2.1 dB for Basegraph 2). This operating point provides a rich mixture of correct and erroneous blocks, fostering generalisable weight updates without the need for an exhaustive sweep over the SNR axis.

Hyper-parameters are selected through a small grid search. The final configuration em-

plays a batch size of 500, a learning rate of 1.5×10^{-3} , fifteen decoding iterations, and 1 500 training steps. All experiments run on two NVIDIA A100-SXM4-80GB GPUs in mixed precision, a choice that shortens training times with no detectable loss of numerical fidelity.

In sum, the adopted methodology combines a model-driven decoder structure with data-driven parameter learning. This synergy addresses the thesis objectives of improving LDPC decoding performance under realistic 5G conditions while respecting the latency and power constraints of modern communication hardware.

3.2 System Design

The used neural network architecture is shown in Figure 3.1, where positions of learnable normalization weights (α_n , β_n , and γ_n) and offset weights (α_o , β_o , and γ_o) can be seen. The layer structure is not a fully connected neural network. Instead, neurons are connected based on edges in the Tanner graph of used LDPC code, which are ultimately defined by the parity-check matrix.

Motivation for the used neural network architecture is based on work in [28], where Li et al. convert MS algorithm into a trellis structure, and trainable parameters α , β , and γ are

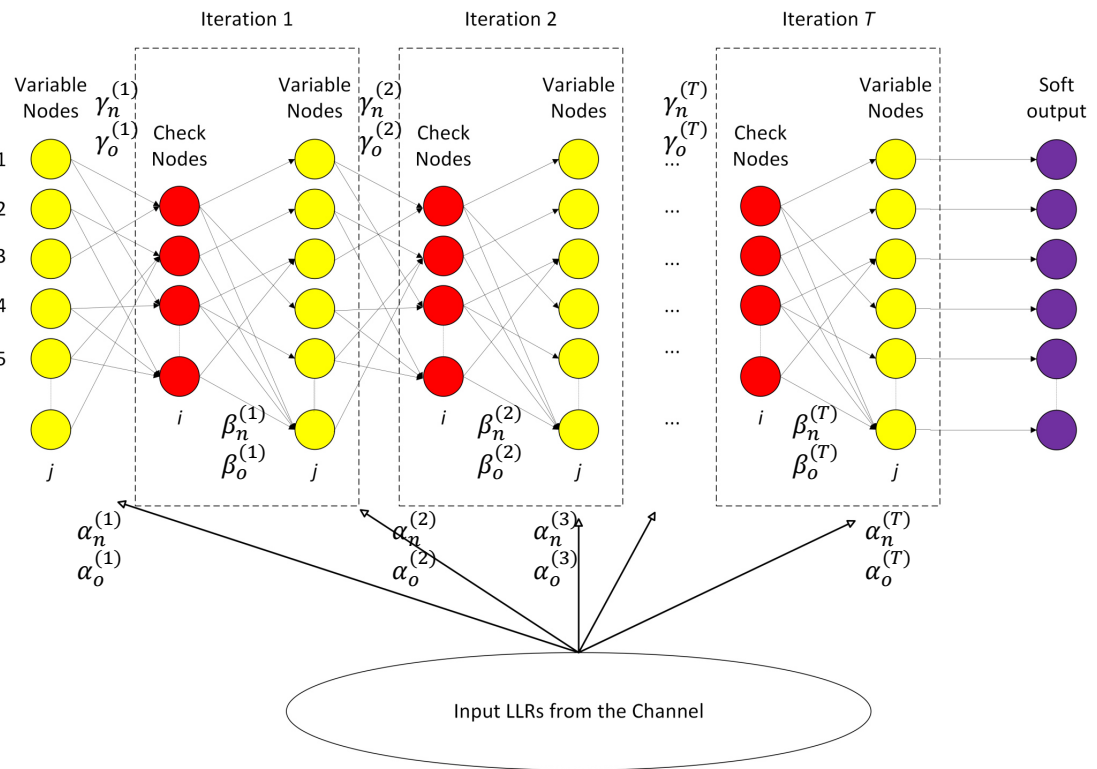


Figure 3.1. Neural network architecture highlighting the positions of learnable normalization and offset weights α , β , and γ within the Tanner graph structure. Adapted from [28].

added on the edges. With trainable parameters, the algorithm is called Neural Normalized Min-Sum (NNMS). Using the notation shown in Figure 3.1, Li et al. use multiplicative weights α_n , β_n , and γ_n in their work. Furthermore, they use less complex LDPC codes, whereas 5G LDPC codes are explored in this thesis.

Additionally, in [54] Wu et al. combine the usage of multiplicative and additive weights in LDPC decoding, leading into a linear approximation (LAMS) algorithm. Also, this idea is expanded in this thesis. Wu et al. add weights to check node updates and channel LLR calculations. Using notation in Figure 3.1, they use weights α_n , α_o , γ_n , and γ_o in their work.

In the context of this thesis, multiplicative and additive weights are called normalization and offset weights, respectively. After applying these weights, message update rule from variable to check node becomes

$$x_{v_i \rightarrow c_j}^{(l)} = \max(\alpha_n^{(l)} b_{v_i} + \alpha_o^{(l)}, 0) + \sum_{\substack{c_p \rightarrow v_i \\ p \in \mathcal{C}(i)/j}} (\beta_n^{(l)} x_{c_p \rightarrow v_i}^{(l-1)} + \beta_o^{(l)}). \quad (3.1)$$

Additionally, message update rule from check to variable node with normalization and offset weights becomes

$$x_{c_j \rightarrow v_i}^{(l)} = \max \left(\left(\prod_{\substack{v_q \rightarrow c_j \\ q \in \mathcal{V}(j)/i}} \text{sgn}(x_{v_q \rightarrow c_j}^{(l)}) \right) \left(\gamma_n^{(l)} \min_{q \in \mathcal{V}(j)/i} |x_{v_q \rightarrow c_j}^{(l)}| + \gamma_o^{(l)}, 0 \right) \right). \quad (3.2)$$

There are several approaches, how this neural network architecture and its weights can be utilized. These different approaches are related to weight formats (scalar value or vector), and weight uniqueness in different iterations (same shared weight in every iteration or unique weight in each iteration).

It is possible to use either single values or vectors as weights. Using single values significantly reduces the total number of weights in the implementation. There are six different weights per iteration: α_n , α_o , β_n , β_o , γ_n , and γ_o . Consequently, for T decoding iterations, the total number of weights is $6 \times T$. If vectors are used as weights, there are i elements in γ_n , and γ_o vectors (number of check nodes). Consequently, there are j elements in α_n , α_o , β_n and β_o vectors (number of variable nodes).

3.3 Training and Evaluation

The selection of the used loss function during training is a very critical choice. In the literature, it is a common choice to use the cross-entropy loss function with ML-assisted LDPC decoders (e.g., [53, 35, 54]). The cross-entropy loss measures the errors between

the decoder's estimate \hat{c} and the transmitted codeword c . It is defined as

$$\mathcal{L}_{\text{CE}}(c, \hat{c}) = -\frac{1}{N} \sum_{i=1}^N [c_i \log(\hat{c}_i) + (1 - c_i) \log(1 - \hat{c}_i)], \quad (3.3)$$

where c_i is the i -th transmitted information bit (ground truth), and \hat{c}_i is the corresponding estimate produced by the decoder. Here, N represents the total number of samples. This loss function penalizes discrepancies between the decoder's estimates and the true transmitted codeword, promoting more accurate predictions.

The starting point of the development for this model was also the regular cross-entropy loss. While BER performance was good with this approach, the BLER performance was relatively poor. This result indicates that the model can correct bits overall well, but it fails to decode entire blocks correctly. This is not desirable since real-world applications require full blocks to be decoded without errors. To address this, a novel approach is used in this thesis, where information bits and parity bits are assigned different weights in the loss calculation. By doing so, it becomes possible to emphasize certain components, making them more dominant in the loss function. The custom weighted binary cross-entropy loss is defined as

$$\mathcal{L}_{\text{WBCE}}(c, \hat{c}) = -(\omega_{\text{info}} \mathcal{L}_{\text{info}} + \omega_{\text{parity}} \mathcal{L}_{\text{parity}}), \quad (3.4)$$

where the loss components for information bits and parity bits are given by

$$\mathcal{L}_{\text{info}} = \frac{1}{N_{\text{info}}} \sum_{i=1}^{N_{\text{info}}} [c_i \log(\hat{c}_i) + (1 - c_i) \log(1 - \hat{c}_i)], \quad (3.5)$$

$$\mathcal{L}_{\text{parity}} = \frac{1}{N_{\text{parity}}} \sum_{j=1}^{N_{\text{parity}}} [c_j \log(\hat{c}_j) + (1 - c_j) \log(1 - \hat{c}_j)]. \quad (3.6)$$

In these equations, N_{info} and N_{parity} denote the number of information bits and parity bits, respectively. The weights ω_{info} and ω_{parity} are applied to the information bits and parity bits, respectively, to control their relative importance in the loss function. The variables c_i and c_j represent the true values of the i -th information bit and j -th parity bit, respectively, while \hat{c}_i and \hat{c}_j are the corresponding estimates produced by the decoder. This weighted approach allows the loss function to prioritize specific parts of the codeword, ultimately improving BLER performance while maintaining good BER performance.

Figure 3.2 illustrates the training process representing all the blocks involved and their relationships. It shows the whole data transmission pipeline from random bit generation

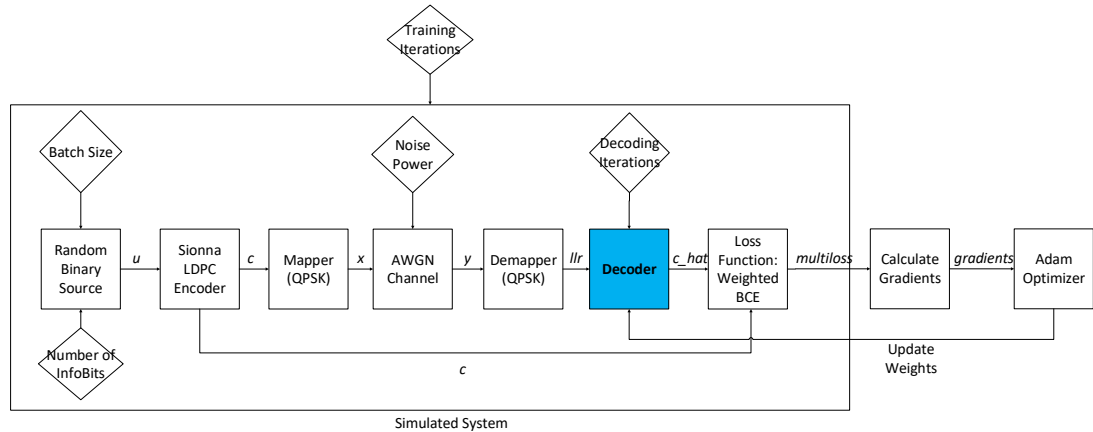


Figure 3.2. Training process representation. Flow chart illustrates all the blocks involved in the training process of the model and their relationships.

leading ultimately into estimated codewords from the decoder. Using these values, the weights of the neural network are optimized. Inside the system, the Sionna library is utilized. It is a TensorFlow-based open-source library for simulating the physical layer of wireless and optical communication systems. It is continually expanded and developed to support 5G and 6G research. [21]

The pipeline starts from a random binary source. It generates batches of random bit sequences u , having a length determined by the number of information bits used in the model. Then these random bit sequences are encoded using Sionna's 5G LDPC encoder, leading into batches of codewords c .

The modulation scheme employed in the pipeline is QPSK (Quadrature Phase Shift Keying). By mapping encoded bit sequences c into QPSK symbols x , the effective symbol rate is halved compared to the bit rate. This reduction arises because QPSK represents two bits per symbol, effectively condensing the information carried by the binary codewords into a more compact form suitable for transmission or further processing. Consequently, this step not only prepares the encoded information for subsequent stages but also optimizes bandwidth usage without compromising the integrity of the data.

Then noise is added to symbols through the AWGN channel. The used noise power strategy must be selected before simulation starts. Selecting the appropriate noise power strategy is a critical step in achieving a well-behaving model. By using too little noise, the model would struggle in real-life scenarios as those have noise and interference present. On the other hand, by using too much noise, the model wouldn't learn the patterns between transmitted data and received data as the received data would be too distorted. Thus, during the model development various approaches were studied to achieve the best possible outcome.

After the channel, the first component in the receiver end is demapper. The demapping

process in the proposed system is performed using the *a posteriori probability* (APP) method, which is implemented using Sionna’s built-in demapper. In this setup, the demapper uses the provided constellation to calculate the Log-Likelihood Ratios (LLRs) for each bit in the received signal. The APP demapper does not directly output the most probable symbol, but instead provides LLRs for each bit in the transmitted codeword.

After demapper, LLR values are passed to the decoder as an input. When defining the decoder, the maximum number of decoding iterations is set. The decoding process continues either until all parity-check equations are fulfilled or until the maximum number of iterations is reached. As an output, the decoder passes the estimations of transmitted codewords \hat{c} .

Then transmitted codewords c and their decoded estimates \hat{c} are passed as an input to the loss function. There weighted binary cross entropy (WBCE) is calculated where information and parity parts have different weights as explained earlier. In [35], multiloss approach in loss calculation is presented by Nachmani et al. With multiloss, the loss value is calculated after each decoding iteration separately, and added to the previous losses cumulatively. After a single decoding round is ready (parity-checks fulfilled or maximum iterations reached), gradients are calculated and the loss is being reset to zero. Alternatively, only the loss of the last iteration could be used. It was studied during the development, but multiloss provided clearly better results.

Before applying gradients with the optimizer, they are clipped. Gradient clipping is a regularization technique that prevents individual gradient components from becoming too large, which can lead to unstable training and divergence. This is done using element-wise clipping, which constrains each gradient component within a predefined range:

$$g'_i = \min(\lambda_{\max}, \max(\lambda_{\min}, g_i)), \quad (3.7)$$

where g_i is the original gradient component, g'_i is the clipped gradient component, and λ_{\min} and λ_{\max} define the lower and upper bounds, respectively. This technique ensures numerical stability and prevents extreme gradient values from disrupting the training process.

With this system, Adam optimizer is used in backpropagation after gradients are calculated. Thus, Adam optimizer updates the weights of the model with each training iteration little by little towards minimizing the loss.

3.4 Implementation Details

The experiments for this thesis were conducted on a high-performance computing system equipped with NVIDIA A100-SXM4-80GB GPUs [11]. These GPUs are designed for

demanding artificial intelligence (AI) and high-performance computing (HPC) workloads. Each GPU features 80 GB of high-bandwidth memory (HBM2e) and a maximum power draw of 400 W. The NVIDIA A100 architecture supports mixed precision computing, which is particularly beneficial for accelerating deep learning and scientific computing tasks.

The system utilized for the experiments included eight NVIDIA A100-SXM4-80GB GPUs while two of them were used with the training and experiments. The driver version installed on the system was 535.216.01, and the experiments were conducted using CUDA version 12.2 [10]. Each GPU provides 80 GB of memory, which is ideal for handling large-scale machine learning models and datasets. For the primary experiments in this thesis, two GPUs were utilized. During these runs, one GPU reached a memory usage of 79,529 MiB, which corresponds to approximately 99.4% of its total capacity, while the other GPU showed a memory usage of 423 MiB. These GPUs were dedicated to the computational tasks of the experiments, while other GPUs in the system remained idle or were assigned to unrelated tasks. The use of NVIDIA A100 GPUs enabled the computational performance necessary to implement and optimize the machine learning-assisted LDPC decoder proposed in this thesis. Their high memory capacity and support for CUDA optimization were essential for managing the large-scale computations involved in this research. Training and evaluation of the system are performed using Python with TensorFlow version 2.15.1.

4. RESULTS AND ANALYSIS

In this section, the experimental results of the proposed ML-assisted LDPC decoder are presented. The experiments aim to evaluate its performance compared to traditional decoding algorithms, namely Belief Propagation and Min-Sum, which serve as baseline methods. The baseline results are obtained using the 5G LDPC decoder from the Sionna library [21].

A comprehensive testing environment was developed for this thesis using Python. There, the impact of different parameters can be evaluated to reach the best performing model. The evaluation metrics include bit error rate (BER) and block error rate (BLER). Among these, BLER is particularly relevant for real-world applications, as decoding entire blocks correctly is often critical. The impact of different numbers of decoding iterations is also investigated, highlighting the trade-off between decoding performance and latency. While additional iterations improve performance, they also increase latency, necessitating a balance between the two. Furthermore, experiments explore different approaches to weight representation, as previously explained. Using scalar weights requires less memory but may compromise performance compared to vector-based weights.

Experiments are conducted for both Basegraph 1 and Basegraph 2 configurations. The block size, corresponding to the number of information bits before encoding, is 520. Two code rates, R , are considered to ensure comprehensive evaluation of both basegraphs: $R = 0.8$ for Basegraph 1 and $R = 0.6$ for Basegraph 2. For the Basegraph 1 case, simulations are conducted in the E_b/N_0 range of 2.00 to 3.75 dB. In contrast, for the Basegraph 2 case, the E_b/N_0 range is 0.5 to 2.75 dB. The difference in simulation ranges is due to the varying number of parity bits. When the code rate is $R = 0.8$, fewer error-correcting parity bits are present in the codeword. This reduces the error correction capability, requiring higher-quality signals for successful decoding.

Table 4.1 shows used hyperparameter values in these experiments unless otherwise mentioned. However, some results are also shown with different values, but that is mentioned in those sections.

As previously mentioned, several approaches related to channel noise were studied and experimentally evaluated. A natural starting point was to train the neural network with a wide range of noise power levels. While this approach yielded reasonable results, the

Table 4.1. Hyperparameters used for the ML-assisted LDPC decoder

Parameter	Value
Learning rate	0.0015
Learning rate scheduler	Constant
Loss function	Weighted Binary Cross-Entropy (WBCE)
Loss function weights ($\omega_{\text{info}}, \omega_{\text{parity}}$)	(0.2, 0.8)
Gradient clipping range	[-10, 10]
Batch size	500
Decoding iterations	15
Total training iterations	1500

best performance was achieved by training with a carefully selected single noise power. This outcome can likely be attributed to the relatively low number of parameters in the network compared to, for example, deep convolutional neural networks (CNNs). When the network learns an effective weight configuration under a specific noise condition, these weights tend to generalize well across other noise levels. In contrast, training with varying noise levels may introduce additional complexity, potentially hindering the optimization process and preventing the network from converging to the most optimal weights.

The noise power is expressed in terms of the energy-per-bit to noise power spectral density ratio, E_b/N_0 , measured in decibels (dB). For the experiments conducted in this thesis, the selected noise power levels were $E_b/N_0 = 3.0$ dB for BG1 and $E_b/N_0 = 2.1$ dB for BG2. These specific values were chosen based on empirical observations and were found to yield significant performance improvements.

4.1 Results of Different Weight Combinations

In these experiments, the performance impact of different weight combinations is shown. For normalization-weights, two setups are used: α_n , β_n and γ_n being scalar-based or vector-based. For offset-weights, three approaches are experimented: α_n , β_n , and γ_n being similarly scalar-based or vector-based, but also the exclusion of them is used as one option. This leads to six different models whose BLER and BER performances are plotted in the same figure to easily evaluate their differences. In these experiments, 15 decoding iterations are used. Whereas, the number of training iterations is 1500.

4.1.1 BLER and BER for Basegraph 1

Figure 4.1 shows the BLER performance of different models with a batch size of 500 for $R = 0.8$, leading to Basegraph 1. The results indicate that the proposed ML-assisted

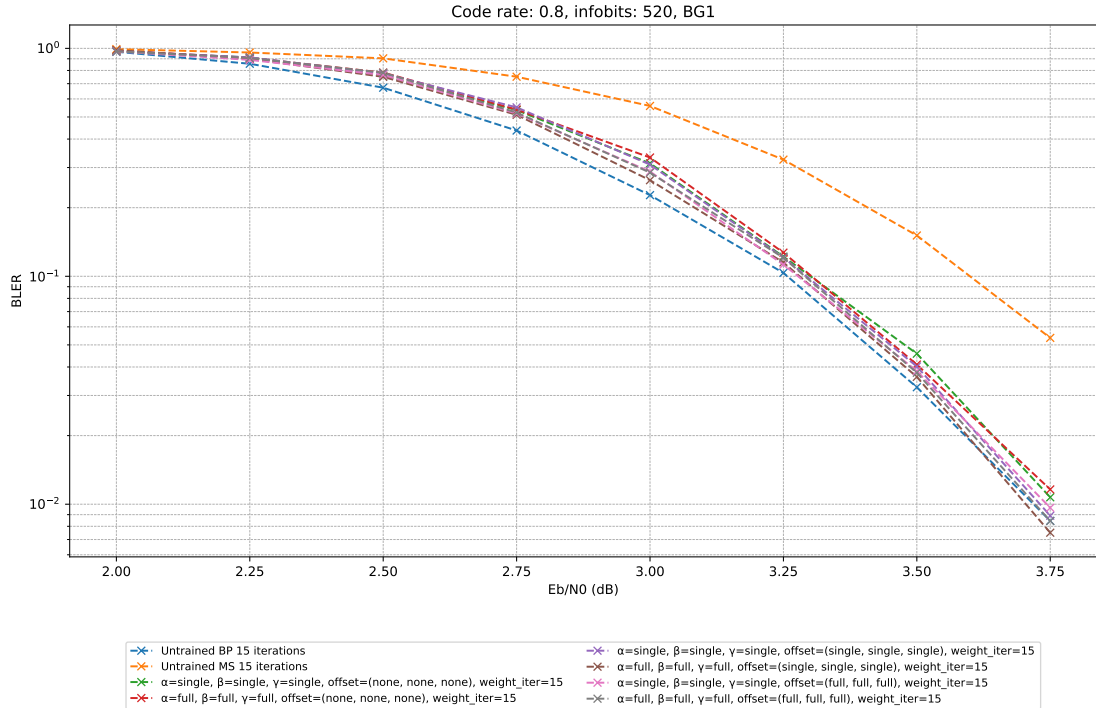


Figure 4.1. BLER performance of different models with batch size 500 for BG1

decoder consistently outperforms MS algorithm, and the performance approaches BP algorithm. In the legend, different models are described. Related to ML-models, the used normalization weight approach is first indicated as α , β , and γ . Then, the used offset weight approach is indicated as *offset*. There *none* indicates that weight is not used in that position, indicating it's 0 for offset weights. Consequently, *single* indicates that a scalar value is used as a weight per iteration. Whereas, *full* indicates that vectors are used as weights, meaning each edge has a unique weight. This same notation is used in all the experimental figures in this thesis. Figure 4.2 extends this analysis to a larger batch size of 5000. While the overall trends remain similar, the increased batch size introduces slight performance improvements with models using more weights.

The BER performance of the models is evaluated in Figures 4.3 and 4.4. In Figure 4.3, the BER results for a batch size of 500 reveal that the ML-assisted decoder achieves significantly lower error rates across the whole E_b/N_0 range compared to the MS, approaching performance of BP. Similarly, Figure 4.4 shows the BER performance for a batch size of 5000, where the proposed decoder maintains its advantage. This analysis reveals that the BER performance with more weights improves even more than BLER, when using larger batch size.

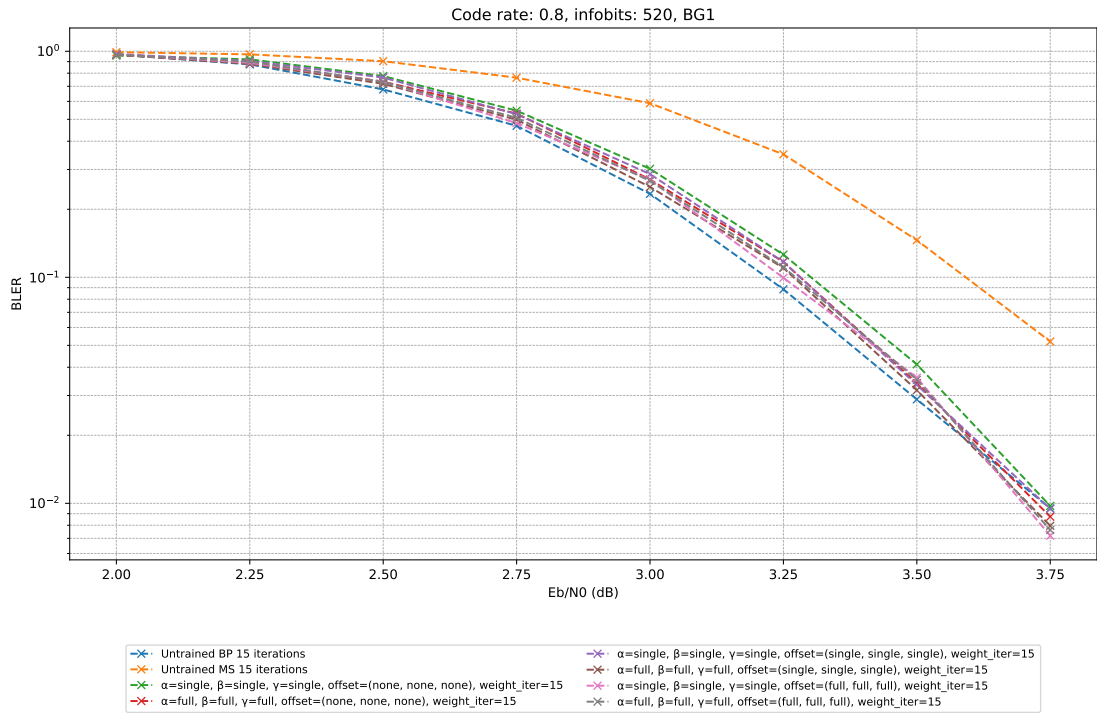


Figure 4.2. BLER performance of different models with batch size 5000 for BG1

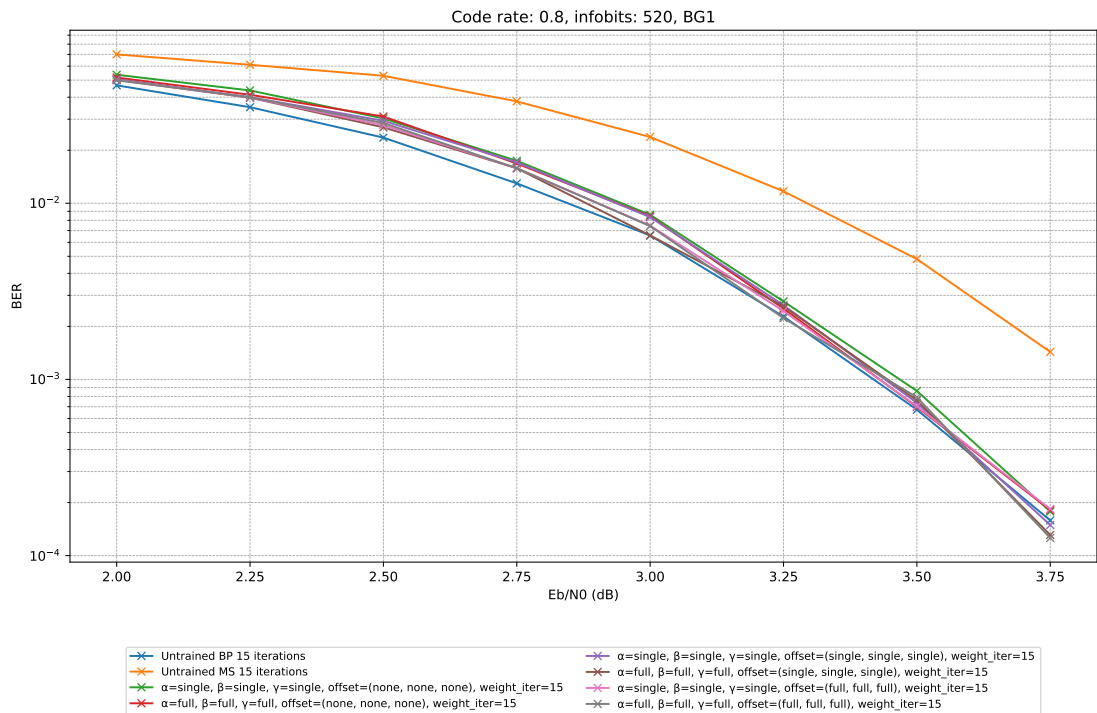


Figure 4.3. BER performance of different models with batch size 500 for BG1

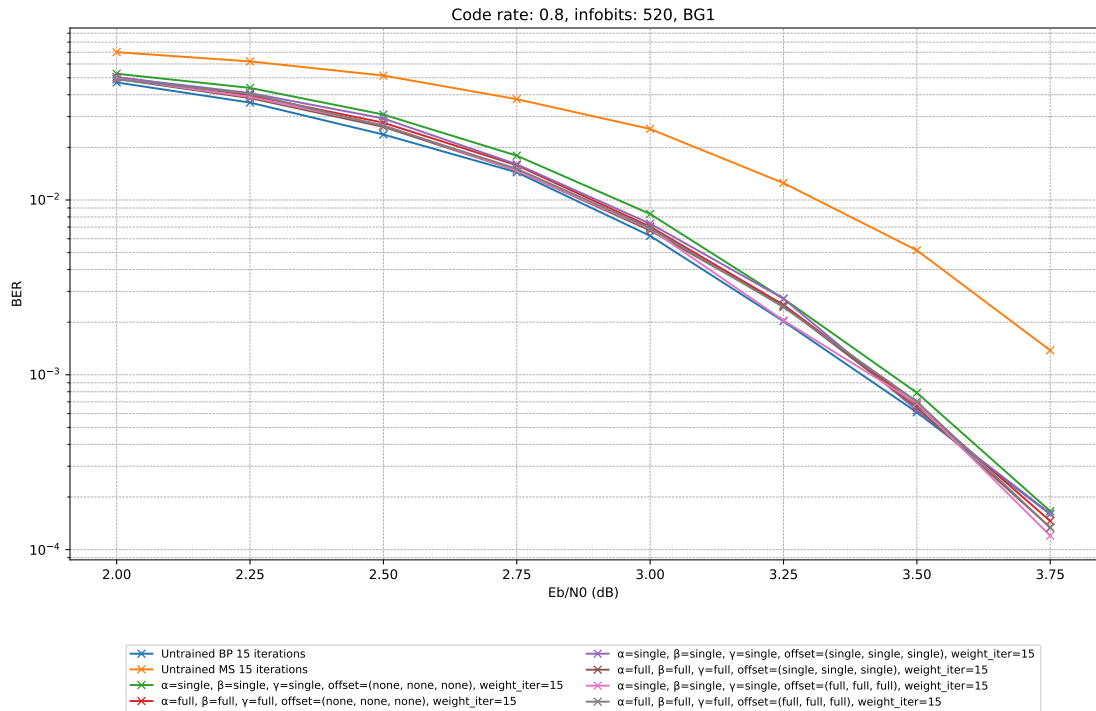


Figure 4.4. BER performance of different models with batch size 5000 for BG1

4.1.2 BLER and BER for Basegraph 2

Similar BLER and BER results with different batch sizes are shown for Basegraph 2 in Figures 4.5, 4.6, 4.7 and 4.8. BLER results show a similar pattern to that with Basegraph 1, models with more weights perform slightly better with more weights. With BER, the performance is slightly different. With Basegraph 2, BER performance doesn't improve when increasing batch size. Overall, the BLER and BER performance differences are relatively small with different batch sizes.

4.2 Training Metrics

During the training process, several metrics were recorded. Most importantly, loss value (which was used in weight optimization) was stored after each iteration. Figure 4.9 shows the loss during training for both BG1 and BG2 when full vectors are used as both normalization and offset weights.

Additionally, the Bit Error Rate (BER) was calculated after each training iteration. The BER values for both BG1 and BG2 with full vector weights are illustrated in Figure 4.10. Similarly, the Bitwise Mutual Information (BMI), another key metric, was computed after each iteration and is shown in Figure 4.11 with full vector weights.

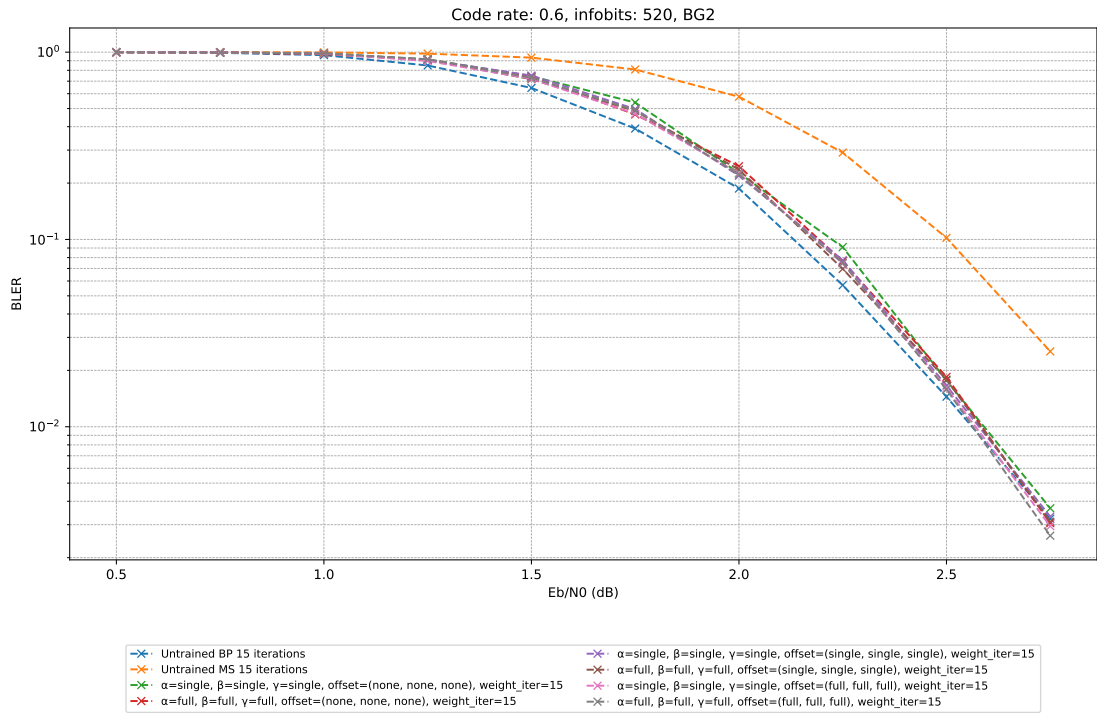


Figure 4.5. BLER performance of different models with batch size 500 for BG2

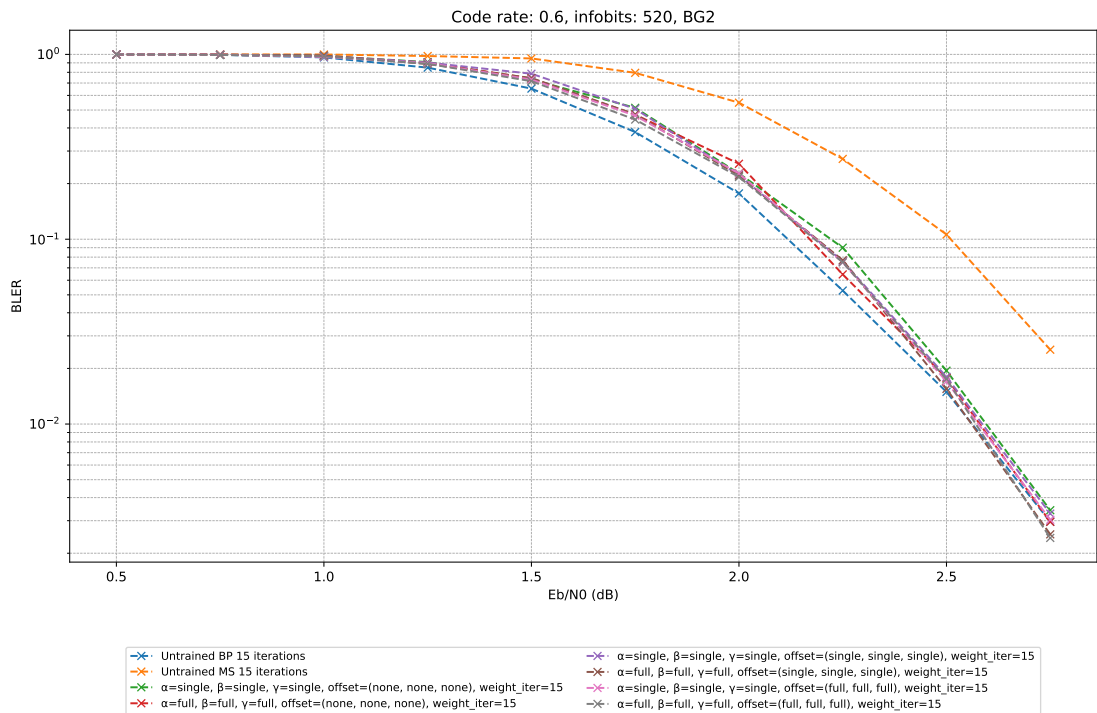


Figure 4.6. BLER performance of different models with batch size 5000 for BG2

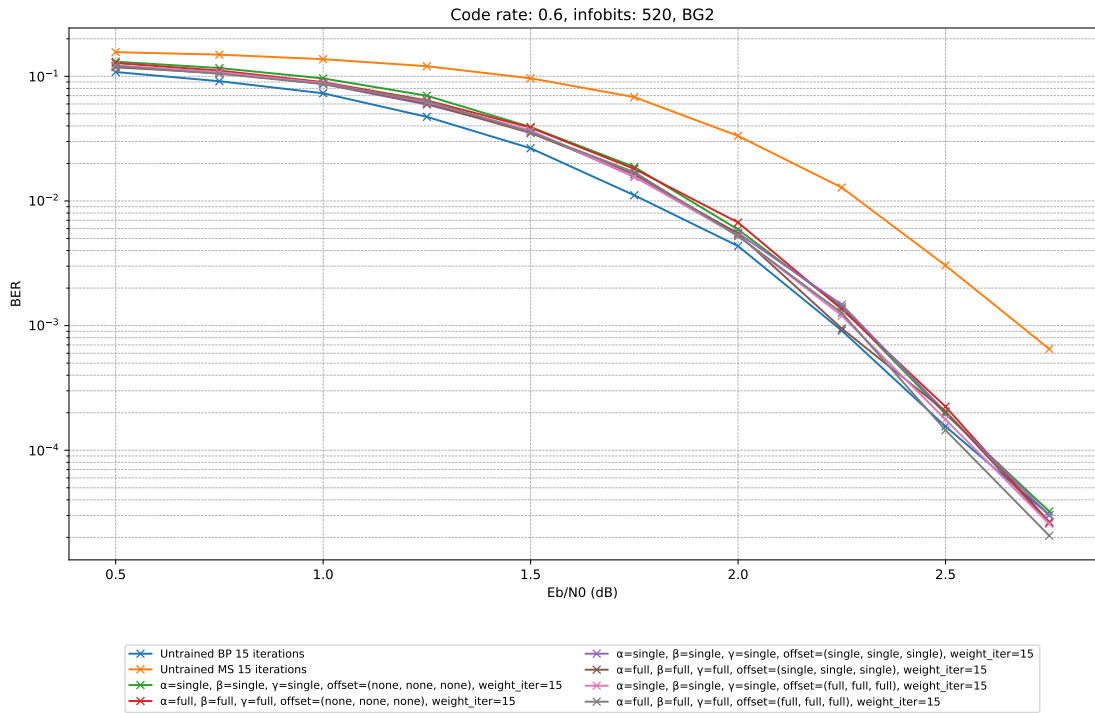


Figure 4.7. BER performance of different models with batch size 500 for BG2

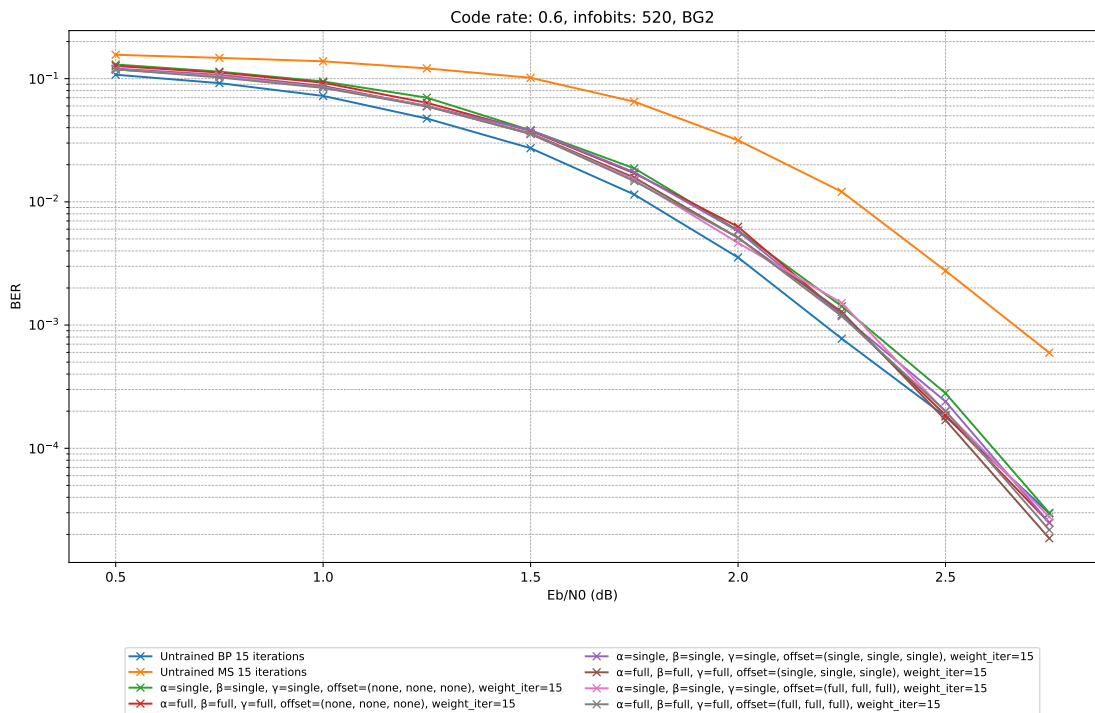


Figure 4.8. BER performance of different models with batch size 5000 for BG2

4.3 Weight Distribution

The weight distribution plays a significant role in the performance of the ML-assisted LDPC decoder. This section provides a detailed analysis of the weight distributions for

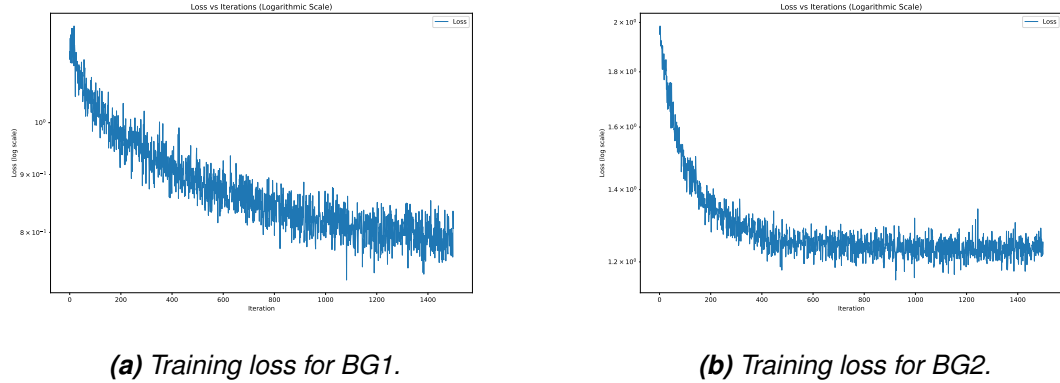


Figure 4.9. Loss during training with full weights.

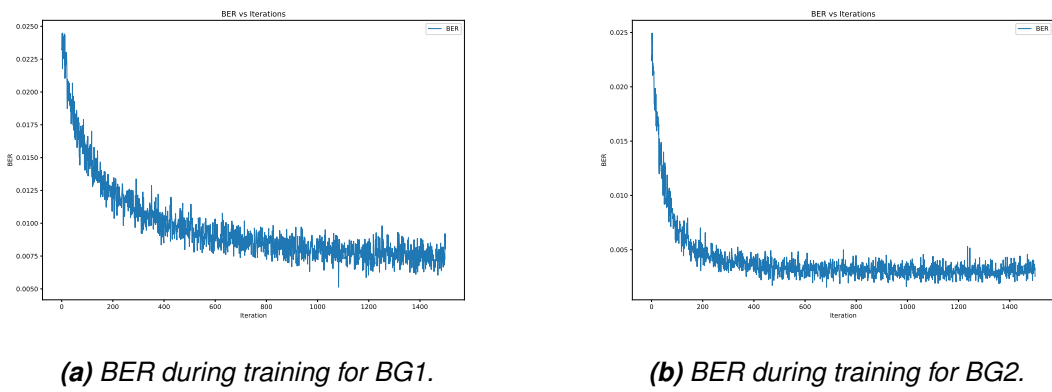


Figure 4.10. BER during training with full weights.

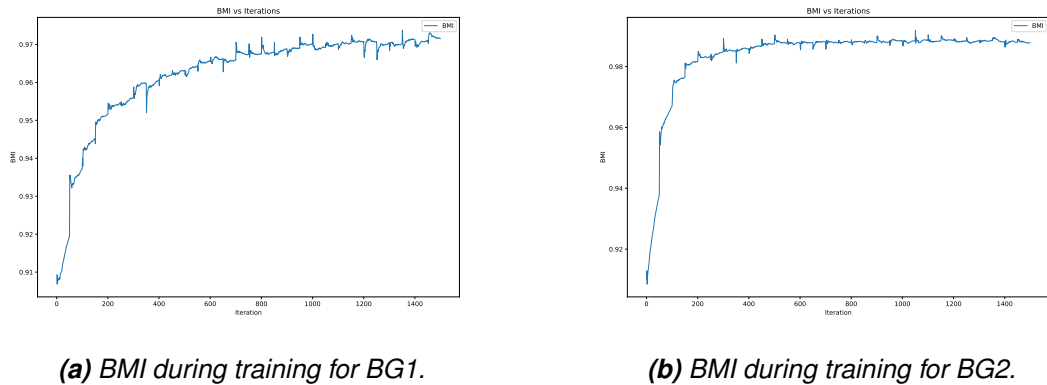


Figure 4.11. BMI during training with full weights.

both vector-based (*full*) and scalar-based (*single*) weights. The distributions are visualized to illustrate the differences between these approaches and their impact on decoding performance. It is not feasible to clearly visualize all the weights for all the experiments in the context of this thesis. However, both approaches (vector-based or scalar-based weights) are explored. With these approaches, different weight types are being selected to be visualized.

For vector-based weights, histograms are used to show the distribution of weights across all decoding iterations. These histograms capture the variation in weight values for each edge in the decoding graph.

Figure 4.12 shows the histogram of the *beta-normalization* weights for Basegraph 1 (BG1). It highlights how these weights are distributed after training, providing insight into the adaptation of the decoder during the learning process. Similarly, Figure 4.13 presents the histogram of *beta-offset* weights for BG1, further illustrating the distribution of vector-based weights and their role in enhancing decoding performance. The weight initialization is clearly visible from these figures. At the beginning of the decoding, there aren't such clear patterns in data. The decoder performs poorly at first, and the behavior is relatively random. For that reason, the majority of weights are initialized values at first, 1 for normalization-weights and 0 for offset-weights. As the decoding progresses, there starts to be a learnable pattern in the decoder outputs and ground truth values. Thus, weights are being optimized such that those adjust the passed messages optimally. Interestingly, the shape of the histograms looks a lot like a normal distribution. Another finding is that the great majority of normalization-weights are less than one, with some caveats. It means that its impact is to reduce the original message value most of the time. Whereas, offset-weights divide surprisingly evenly around zero, meaning that some of the messages are adjusted upwards and some downwards.

For scalar-based weights, a different visualization approach is used to capture their evolution over decoding iterations. As an example, gamma-normalization and gamma-offset weights are explored. In these plots, the x-axis represents the iteration number, while the y-axis indicates the scalar weight value. This representation highlights how weights evolve during the decoding process. Figure 4.14 demonstrates the evolution of *gamma-normalization* weights for BG1 with scalar-based weights. Furthermore, Figure 4.15 illustrates the evolution of *gamma-offset* weights for BG1.

With gamma-normalization weights, the purpose of weight adjustment can be seen to be similar to that with beta-normalization above. During all the iterations, the weight value is less than 1, meaning weights are adjusted downwards. Another similar pattern is related to weight magnitude during iterations. Also in this case, at the beginning of the decoding, the learnable pattern in the data during training is not visible. Thus, two of the first weights will be adjusted the least (closest to value 1). With offset-weights the pattern is a lot different compared to beta-offset with vector-based weights above. Now all the weights are negative, meaning they adjust messages downwards. Also interestingly, the behavior of weight values is relatively linearly decreasing, without much deviation.

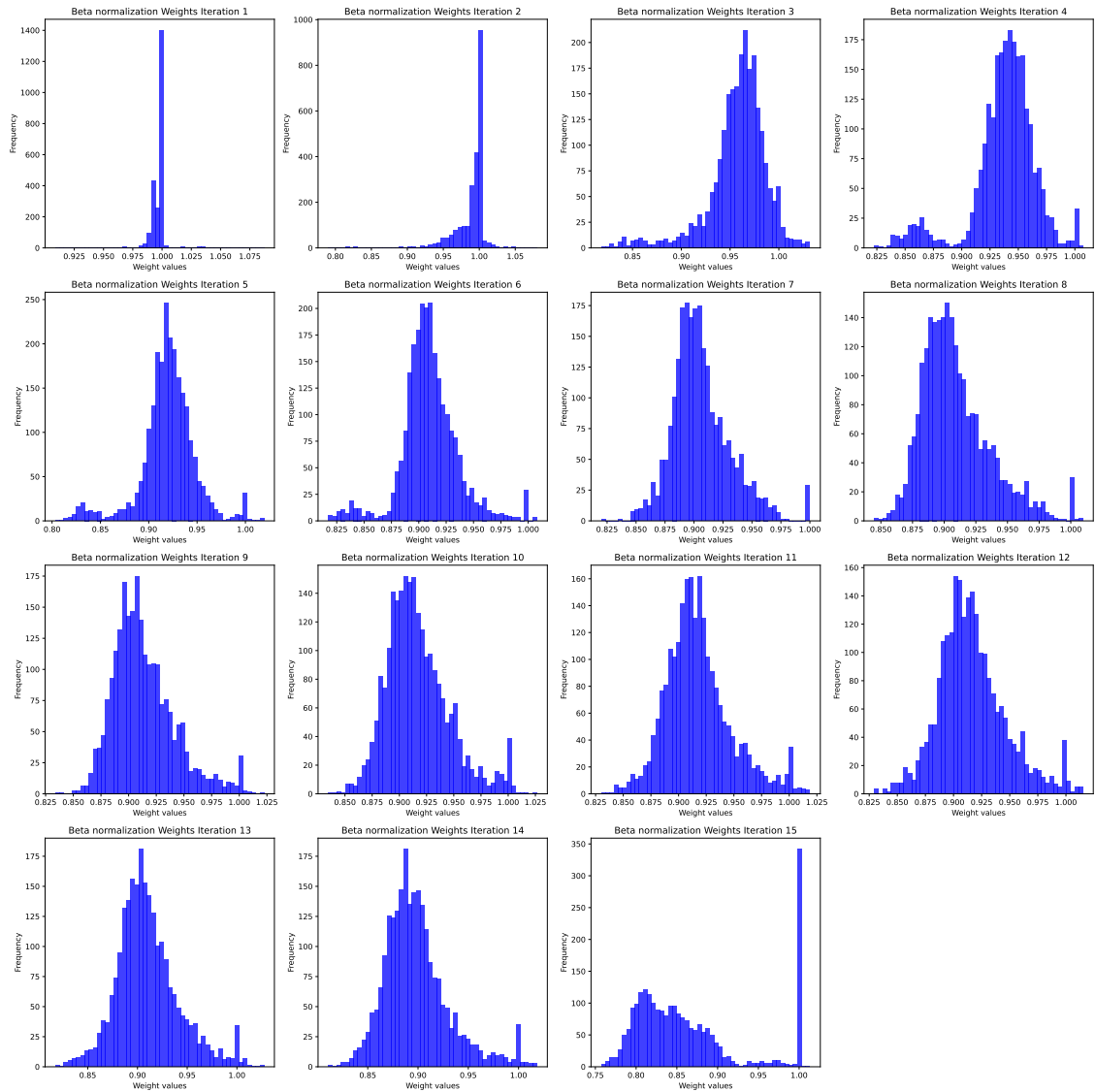


Figure 4.12. Histogram of beta-normalization weights for BG1 using full vector-based weights.

4.4 Results of Different Number of Decoding Iterations

The impact of the number of decoding iterations on BLER and BER performance is evaluated using full weights for both Basegraph 1 (BG1) and Basegraph 2 (BG2). For BG1, the results are illustrated in Figures 4.16 and 4.17, showing how performance improves as the number of iterations increases. Similarly, for BG2, Figures 4.18 and 4.19 provide the corresponding results.

Figure 4.16 highlights the improvements in BLER with additional iterations for BG1, demonstrating the trade-off between latency and decoding accuracy. The corresponding BER improvements are shown in Figure 4.17, emphasizing the reduction in bit errors as iterations increase.

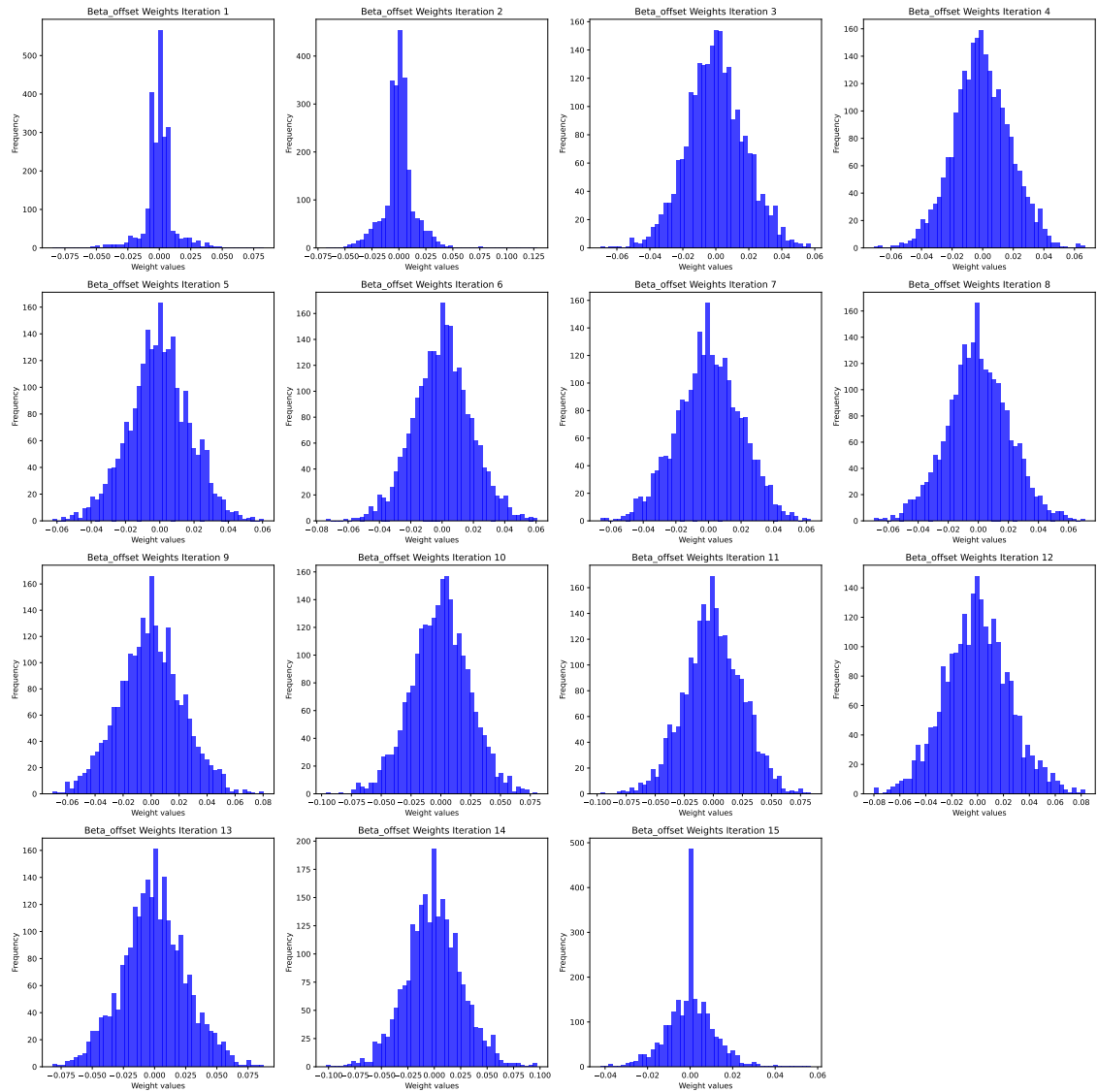


Figure 4.13. Histogram of beta-offset weights for BG1 using full vector-based weights.

For BG2, Figure 4.18 reveals a similar pattern in BLER performance, where additional iterations enhance decoding success rates. The BER performance, detailed in Figure 4.19, follows a comparable trend, with notable improvements in bit error reduction with more iterations.

4.5 Analysis of Different Weight Combinations

When reviewing presented BLER and BER experiment results, it's beneficial to put more emphasis on BLER evaluations, as it's more relevant with real-world applications. However, BER is also a relevant metric in the industry and algorithmic development, so those results are also shown.

For both BG1 and BG2, BLER and BER plots show that using more weights lead to better results. The difference between different models is not major, but the pattern of improving

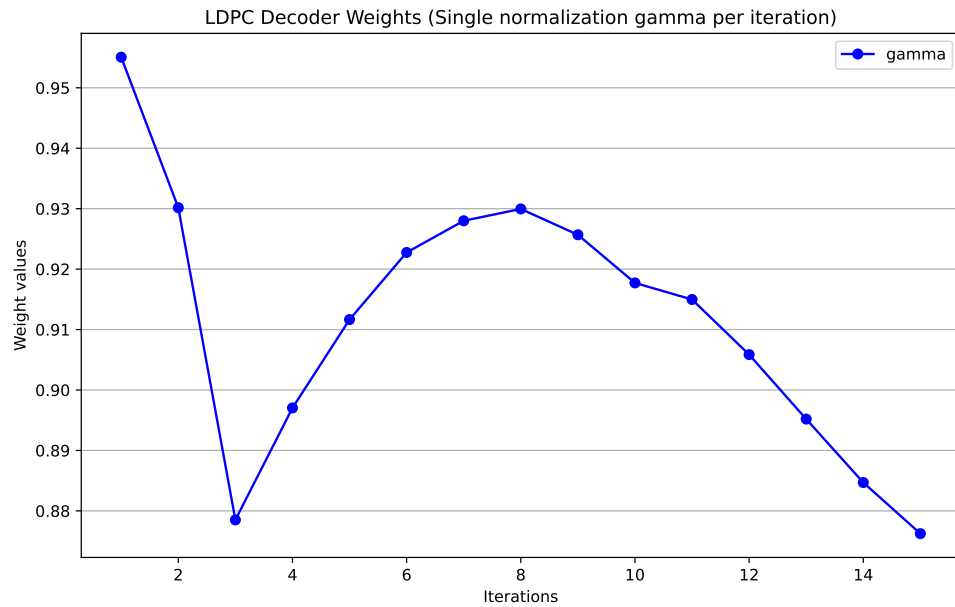


Figure 4.14. Evolution of gamma-normalization weights for BG1 using single scalar-based weights.

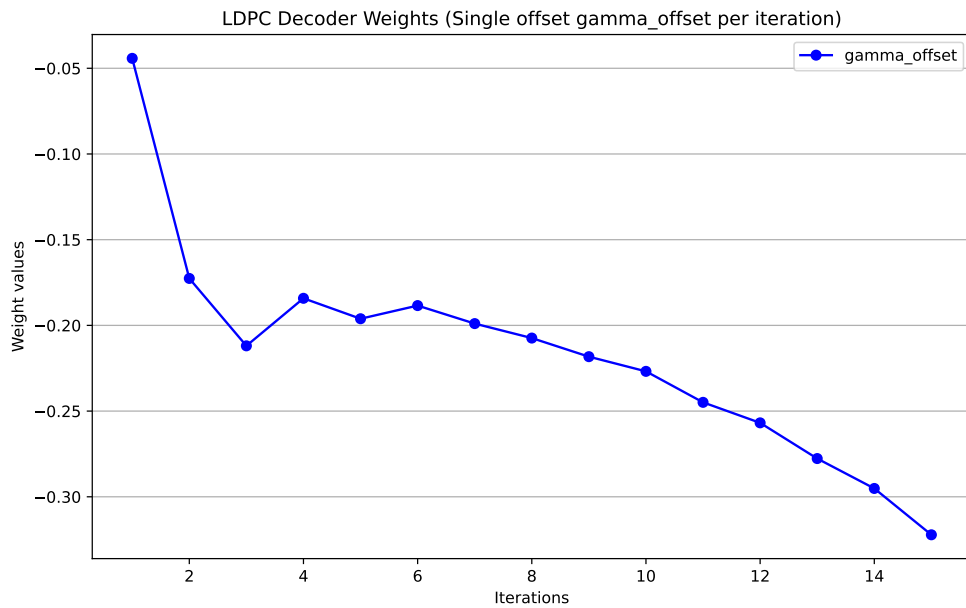


Figure 4.15. Evolution of gamma-offset weights for BG1 using single scalar-based weights.

performance with more weights is clearly visible. Generally, the model with scalar values as normalization weights and no offset weights performs the poorest. Interestingly, the differences with models using more weights are so small that it's not feasible to determine the absolutely best performing model. Also, random data is used in the experiments, so

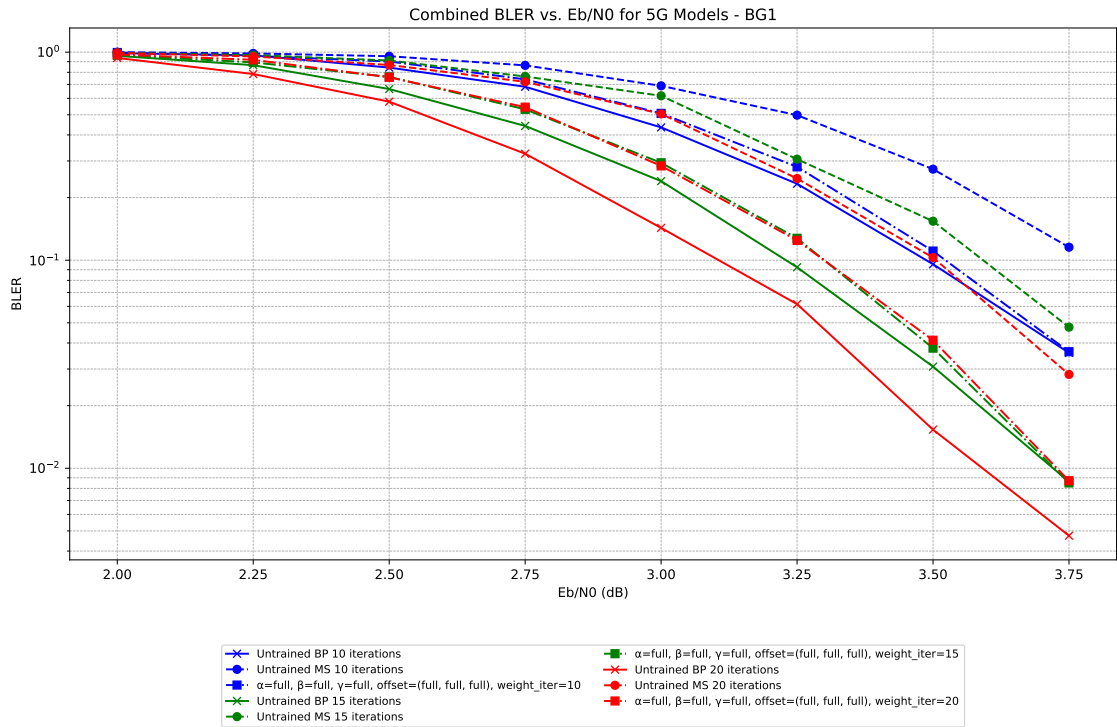


Figure 4.16. BLER performance with full weights using different number of decoding iterations for BG1

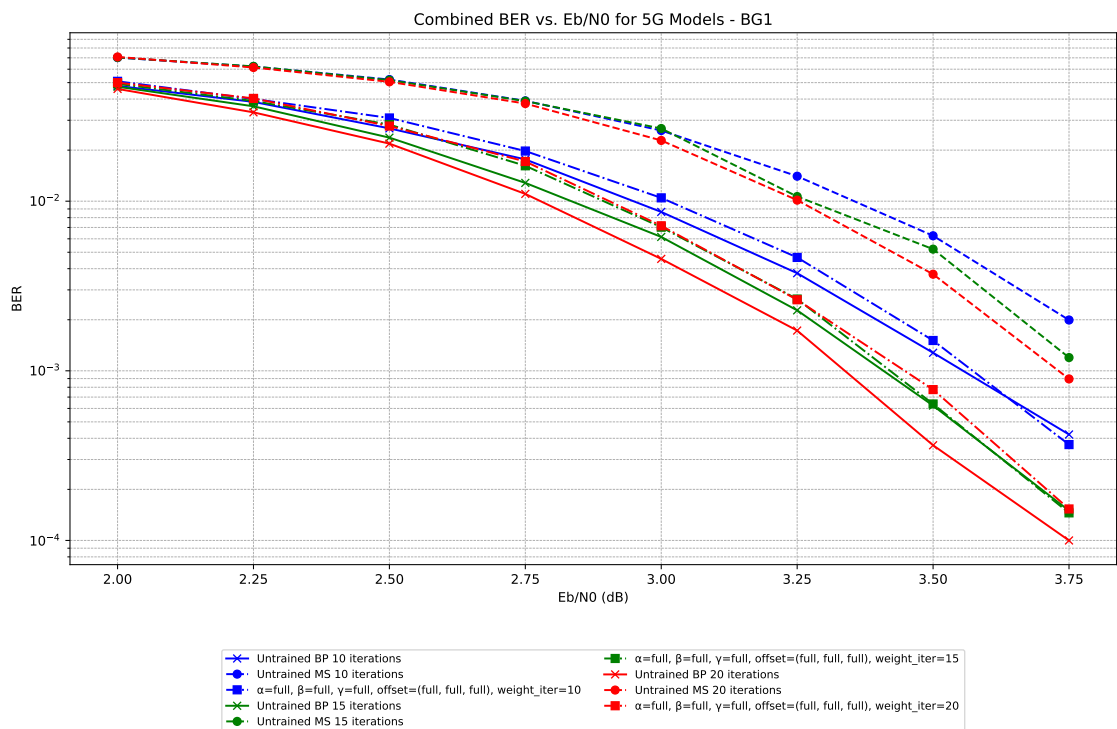


Figure 4.17. BER performance with full weights using different number of decoding iterations for BG1

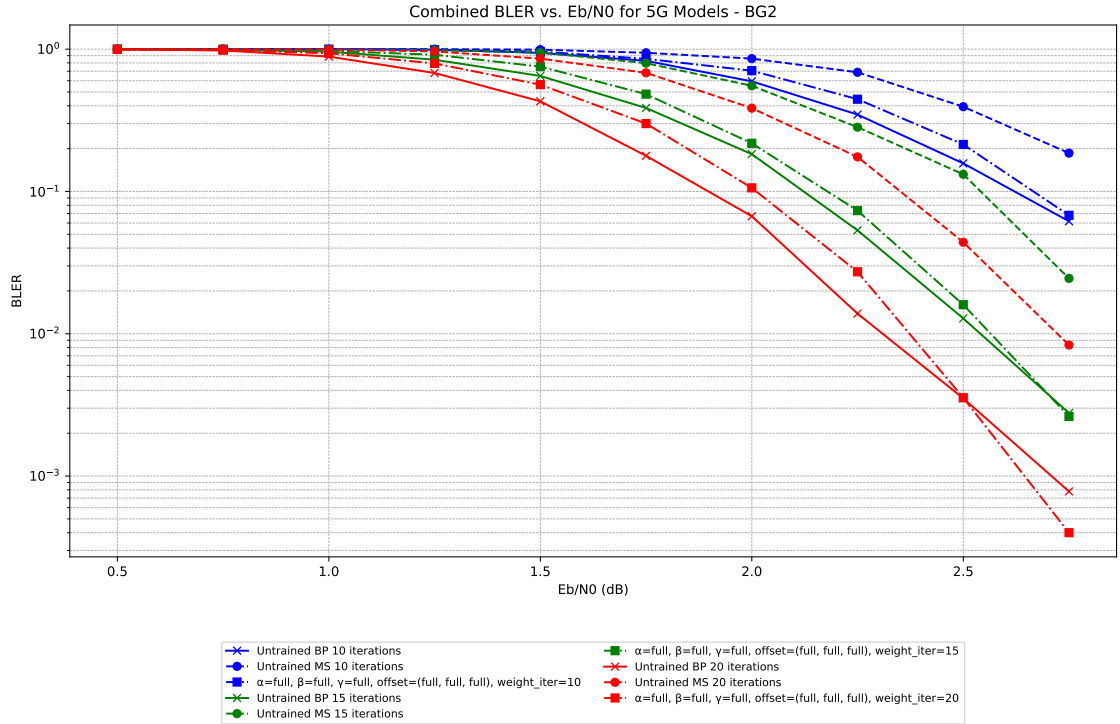


Figure 4.18. BLER performance with full weights using different number of decoding iterations for BG2

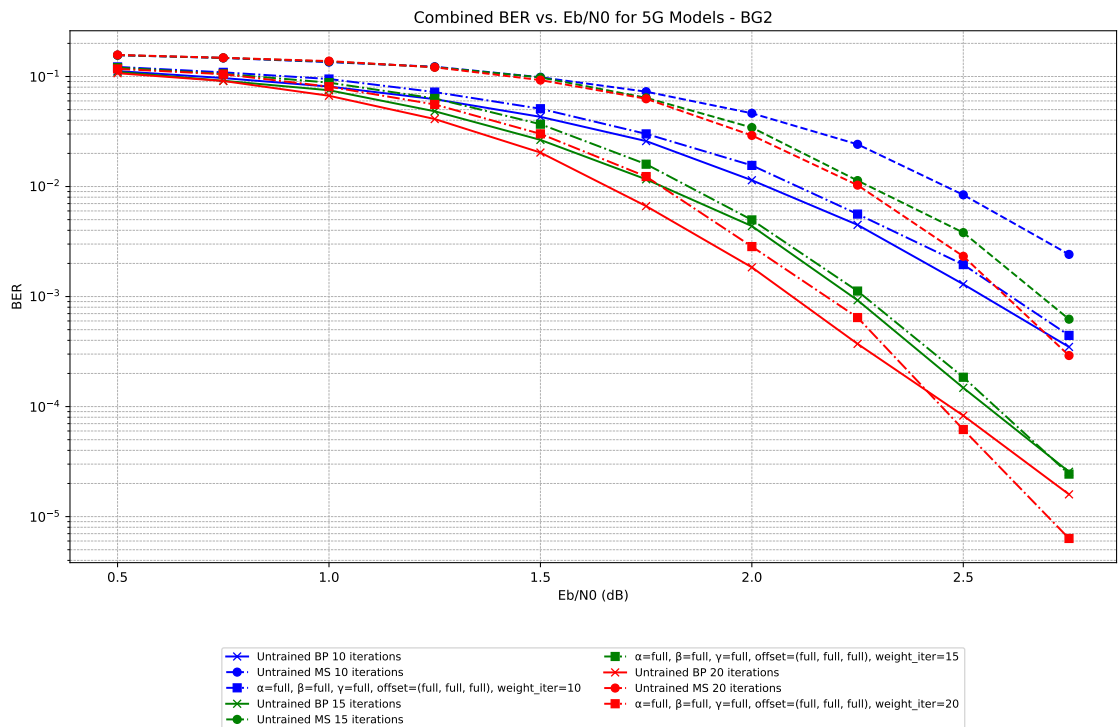


Figure 4.19. BER performance with full weights using different number of decoding iterations for BG2

that can create some variation to results. If it's possible to use the model using the most weights, that is a good choice as it clearly doesn't at least lose to other models. However, if resources are constrained, clearly some reduced model can also be used.

Experiments with batch size show that increasing it makes the performance slightly better overall. The increase is more visible with BG1. With BG1, the improvements occur within both BLER and BER evaluations. With BG2, BLER improvement occurs mainly with higher E_b/N_0 , around 2.5 dB. When evaluating BER with BG2, the increase in batch size doesn't have a clearly visible impact. Larger batch size uses more data during training which makes it more computationally heavy. If resources are constrained, clearly using a batch size value of 500 provides good results, and makes the model well-performing. But if possible, 5000 would be a slightly better batch size compared to 500, if using 1500 training iterations. Another option would be to use a batch size of 500 with more training iterations, and increase the amount of used data that way.

Another important observation from the results concerns the choice of noise power. As previously mentioned, the selected noise power levels were $E_b/N_0 = 3.0$ dB for BG1 and $E_b/N_0 = 2.1$ dB for BG2. For BG1, the block error rate (BLER) of the untrained BP decoder at $E_b/N_0 = 3.0$ dB is approximately 0.2. Similarly, for BG2, the BLER of the untrained BP decoder at $E_b/N_0 = 2.1$ dB is slightly below 0.2. These observations suggest a practical guideline for training new models: an effective starting point is to first evaluate the BLER of an untrained BP decoder and then select a training noise level that corresponds to a BLER close to 0.2. However, this noise level serves only as an initial reference, and further experimentation and optimization will be necessary to achieve the best possible performance.

4.6 Analysis of Different Decoding Iterations

Figure 4.16 shows how BLER performance increases with BG1 using more decoding iterations. Improved performance is clearly visible with all three algorithms: BP, MS and ML-assisted. With 10 iterations, ML-assisted decoding performs very closely compared to optimal BP decoding. With 15 iterations, the gap is slightly larger. However, the gap narrows as E_b/N_0 increases. With 20 iterations, the gap between ML-assisted and BP is the largest of these three iteration counts. Still, significant performance gains are achieved compared to MS algorithm. It's an interesting observation that the performance of ML-assisted decoding increases significantly when moving from 10 iterations to 15 iterations. However, when moving from 15 iterations to 20 iterations, the improvement is much smaller.

BER analysis with BG1 in Figure 4.17 looks slightly different compared to BLER plot. In that figure, the MS algorithm with all iterations performs the poorest. With 10 iterations, the performance of ML-assisted decoding looks very promising as its performance is

almost as good as BP. With 15 and 20 iterations, the gap becomes slightly bigger, but the gains are still significant compared to MS. Interestingly, the performance of ML-assisted decoding is almost the same with 15 and 20 iterations during almost the whole evaluation range. Only at the highest values, 20 iterations beats 15 iterations.

BLER performance with BG2 in Figure 4.18 shows some interesting differences compared to the BG1 experiment. 10 iterations performs poorest in this experiment with all algorithms, whereas BP and ML-assisted with 15 iterations were able to beat MS with 10 iterations using BG1. Secondly with BG2, the performance of ML-assisted decoding improves significantly when increasing decoding iterations from 15 to 20 iterations. With BG1, this improvement was only slightly visible with higher E_b/N_0 values. Additionally, BP algorithm shows larger performance increases when increasing iterations.

BER analysis with BG2 in Figure 4.19 shows some clear patterns. MS algorithm performs poorest of all. When increasing decoding iterations from 10 to 15, the performance improves more than when moving from 15 to 20 iterations. After that, three main patterns are visible. These are BP and ML-assisted decoding with each experimented iteration count performing very closely with improving performance together with increasing iterations forming three sections to the plot.

In conclusion, analysis of different decoding iterations showed two important findings. With BG1, it would be the most suitable to use 15 decoding iterations as the performance gains with 20 iterations were very minor. More decoding iterations doesn't worsen the results, but it adds unnecessary latency without improving accuracy. Secondly, the BG2 case showed major performance gains when moving from 15 to 20 iterations. It clearly is beneficial to use more iterations with this setup.

In summary, every evaluated experiment showed significant performance gains with ML-assisted decoding when compared to baseline MS decoding. So, the usage of learnable weights is beneficial with all iterations.

5. CONCLUSIONS

This thesis presented an innovative approach to LDPC decoding by integrating machine learning techniques with traditional decoding algorithms. By incorporating learnable weights into the message-passing framework of LDPC decoders, significant performance improvements were achieved compared to conventional Min-Sum decoding while approaching the performance of the more computationally expensive Belief Propagation algorithm.

Summary of Contributions

The primary contributions of this research are various and address several challenges in modern error-correction systems. First, a hybrid neural network architecture was successfully developed that applies both normalization weights $(\alpha_n, \beta_n, \gamma_n)$ and offset weights $(\alpha_o, \beta_o, \gamma_o)$ into the edges of the Tanner graph structure of LDPC codes. This architecture preserves the fundamental message-passing principles of traditional decoders while introducing learnable parameters that can adapt to specific channel conditions and code structures.

Then, a novel weighted binary cross-entropy loss function was implemented that assigns different importance to information bits and parity bits during training. The weighted approach allows the decoder to prioritize the correct decoding of entire blocks, which is particularly relevant for practical applications where full block recovery is essential.

In addition, a comprehensive evaluation of different weight combinations was conducted, including scalar-based and vector-based approaches. Achieved results consistently demonstrated that increased weight parametrization generally leads to better decoding performance. This finding provides valuable understanding into the trade-offs between model complexity and decoding accuracy, offering practical guidelines for implementation in resource-constrained environments.

Moreover, a detailed analysis on the impact of decoding iterations on performance was performed, revealing that the optimal iteration counts differ between evaluated Basegraph 1 and Basegraph 2 configurations. For Basegraph 1 configuration, 15 iterations provide an optimal balance between performance and computational complexity, with minimal improvements observed when increasing to 20 iterations. In contrast, Basegraph 2 con-

figuration showed significant performance gains when increasing from 15 to 20 iterations, suggesting that the optimal iteration count is code-dependent.

Finally, a practical guideline for selecting training noise power based on target BLER performance was developed. Conducted experiments indicate that optimal training occurs at noise levels where untrained BP decoders achieve approximately 0.2 BLER. This finding provides a straightforward and effective starting point for training similar ML-assisted decoders in different scenarios. Then more finetuning can occur after reviewing achieved results using that noise power.

Key Findings

Experimental results revealed several important findings that contribute to the understanding of ML-assisted LDPC decoding. Most significantly, the ML-assisted decoder consistently outperformed the Min-Sum algorithm across all tested test setups. This performance improvement was substantial, significantly narrowing the performance gap between MS and BP decoders. The ability to achieve near-BP performance with computational complexity closer to MS represents a promising opportunity, addressing the critical trade-off between decoding accuracy and implementation feasibility.

Regarding weight implementation strategies, vector-based weight implementations generally showed better performance than scalar-based implementations. This performance advantage comes at the cost of increased model complexity and computational requirements, presenting an important consideration for practical deployments. The specific performance gains varied across different noise levels and code configurations, but the trend remained consistent.

The inclusion of offset weights alongside normalization weights provided noticeable performance improvements. The combination of vector-based normalization and offset weights achieved the best results, suggesting that both multiplicative and additive adjustments to message values contribute meaningfully to decoding performance.

The performance analysis across different numbers of decoding iterations provided valuable practical insights. For Basegraph 1, performance improvements between 15 and 20 decoding iterations were minimal, suggesting 15 iterations as an optimal point for balancing performance and latency. For Basegraph 2, significant gains were observed when increasing from 15 to 20 iterations, highlighting the importance of code-specific optimization. These findings can guide implementation decisions, helping to minimize unnecessary computational overhead while maintaining desired error-correction capabilities.

Practical Implications

The ML-assisted LDPC decoder developed in this thesis carries significant practical implications for 5G systems and beyond. The improved error correction performance enables more reliable communication at lower signal strengths, potentially extending coverage areas and reducing power requirements for mobile and IoT devices. This advantage is particularly relevant in challenging environments with limited signal quality or in applications requiring high reliability.

By achieving performance closer to BP with computational complexity closer to MS, the proposed decoder offers an attractive trade-off for resource-constrained devices and applications requiring low latency. This balance is especially important for applications such as autonomous vehicles, industrial automation, and real-time multimedia communications, where both reliability and speed are critical factors.

The adaptability of the neural network approach represents another significant practical advantage. The decoder can be optimized for specific channel conditions, code structures, or application requirements through appropriate training. This flexibility allows for targeted performance improvements in specific deployment scenarios, potentially enabling better overall system optimization than one-size-fits-all approaches.

The findings related to weight distributions and training strategies provide practical guidelines for implementing similar ML-assisted decoders in real-world systems. The observed patterns in weight values after training offer insights into how message values should be adjusted during decoding, potentially informing simpler, rule-based modifications to conventional decoders when full neural network implementation is not feasible.

Furthermore, the weighted loss function approach demonstrated in this thesis could be extended to other error-correction applications beyond LDPC codes. The principle of assigning different importance to different bits based on their role in the codeword structure could inform training strategies for other ML-assisted coding schemes, potentially leading to broader improvements in communication system performance.

Limitations and Future Work

Despite the promising results achieved in this research, several limitations and directions for future work can be identified. Current evaluations focused primarily on the AWGN channel model, which, while foundational, represents an idealized scenario. Future work should extend to more complex channel models, including fading channels and those with burst errors, to evaluate performance under more realistic conditions. Such evaluations would provide more comprehensive insights into the practical viability of the ML-assisted decoder in diverse deployment scenarios.

The hardware implementation aspects of the ML-assisted decoder require further investigation. Detailed assessments of memory requirements, power consumption, and throughput would be necessary to ensure practical viability in real-world systems. The translation from floating-point weights used in training to fixed-point representations suitable for hardware implementation represents a particular challenge that warrants dedicated research.

More sophisticated neural network architectures could potentially yield further performance improvements. While the current approach focuses on simple weight adjustments to the message-passing algorithm, more complex network structures could capture additional patterns in the decoding process. Convolutional neural networks, for instance, might effectively leverage spatial relationships in the Tanner graph, while recurrent neural networks could better model the iterative nature of LDPC decoding.

Reinforcement learning approaches represent an intriguing alternative to the supervised learning methodology employed in this thesis. By framing decoding as a sequential decision-making process, reinforcement learning could potentially enable the decoder to adapt more dynamically to changing channel conditions. This approach might be particularly valuable in mobile or rapidly evolving communication environments.

The current research evaluated the ML-assisted decoder with specific code rates and block sizes. A broader evaluation with varying code parameters would further validate the generalizability of the approach. Understanding how the performance benefits scale with different code lengths and rates would provide valuable insights for standardization and implementation decisions. While this work focused specifically on 5G LDPC codes, the methodology could be extended to other error-correcting code families. Polar codes, for instance, represent another important component of the 5G standard and might benefit from similar ML-assisted decoding approaches. Additionally, exploring the applicability of these techniques to future communication standards beyond 5G could ensure the long-term relevance of this research direction.

The training process itself presents opportunities for optimization. Techniques such as transfer learning might allow pre-trained decoders to be efficiently adapted to new channel conditions or code parameters with minimal additional training. Similarly, quantization-aware training could help address the challenges of hardware implementation by incorporating quantization effects during the training process rather than applying them afterward.

Final Remarks

The integration of machine learning techniques with traditional LDPC decoding represents a promising pathway toward more efficient and adaptive error correction in modern communication systems. This thesis has demonstrated that learnable weights in

the message-passing framework can significantly enhance decoding performance while maintaining reasonable computational complexity. The improvements achieved over conventional Min-Sum decoding highlight the potential of hybrid approaches that combine established algorithmic principles with data-driven optimization.

The weighted binary cross-entropy loss function introduced in this work represents a particularly valuable contribution, demonstrating how domain knowledge about the structure and purpose of error-correction codes can inform machine learning methodologies. This cross-disciplinary approach, combining insights from coding theory with techniques from deep learning, exemplifies the potential for innovation at the intersection of these fields.

As communication systems continue to evolve, demanding ever-higher data rates, lower latencies, and greater reliability, hybrid approaches that combine the theoretical foundation of coding theory with the adaptability of machine learning will become increasingly valuable. The challenges of future communication standards, including higher frequencies, denser networks, and more diverse application requirements, will likely necessitate more sophisticated error-correction techniques than those currently employed.

The ML-assisted LDPC decoder presented in this work offers a concrete step toward addressing these challenges, bridging the gap between conventional algorithms and the requirements of next-generation communication systems. By demonstrating practical performance improvements with implementable complexity, this research contributes to the ongoing evolution of wireless technology and the broader field of error-correction coding.

In conclusion, this thesis has shown that machine learning can effectively enhance traditional LDPC decoding by learning optimal message-passing parameters. The comprehensive evaluation of different weight configurations, training strategies, and decoding parameters provides valuable insights for both researchers and practitioners in the field. As communication systems continue to advance, approaches that combine the strengths of traditional algorithms with the adaptability of machine learning will play an increasingly important role in enabling reliable, efficient, and high-performance error correction.

REFERENCES

- [1] 3rd Generation Partnership Project (3GPP). *TS 38.211: NR; Physical channels and modulation*. Technical Specification (TS) 38.211. Release 18. 3GPP, Mar. 2025.
- [2] 3rd Generation Partnership Project (3GPP). *TS 38.212: NR; Multiplexing and channel coding*. Technical Specification (TS) 38.212. Release 18. 3GPP, Mar. 2025.
- [3] Armen S Asratian, Tristan MJ Denley, and Roland Häggkvist. *Bipartite graphs and their applications*. Vol. 131. Cambridge university press, 1998.
- [4] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep learning*. Vol. 1. MIT press Cambridge, MA, USA, 2017.
- [5] Namrata P Bhavsar and Brijesh Vala. “Design of hard and soft decision decoding algorithms of LDPC”. In: *International Journal of Computer Applications* 90.16 (2014), pp. 10–5.
- [6] Christopher M Bishop. “Neural networks for pattern recognition”. In: *Clarendon Press google schola 2* (1995), pp. 223–228.
- [7] Andres I Vila Casado, Miguel Griot, and Richard D Wesel. “Informed dynamic scheduling for belief-propagation decoding of LDPC codes”. In: *2007 IEEE International Conference on Communications*. IEEE. 2007, pp. 932–937.
- [8] Jinghu Chen et al. “Reduced-complexity decoding of LDPC codes”. In: *IEEE transactions on communications* 53.8 (2005), pp. 1288–1299.
- [9] Walter S. Ciciora. *Modern cable television technology : video, voice and data communications*. eng. 2nd ed. The Morgan Kaufmann Series in Networking. Amsterdam ; Elsevier/Morgan Kaufmann Publishers, 2004. ISBN: 1-281-12045-6.
- [10] NVIDIA Corporation. *CUDA Toolkit Documentation*. Accessed: 2024-12-23. 2024. URL: <https://docs.nvidia.com/cuda/>.
- [11] NVIDIA Corporation. *NVIDIA A100 Tensor Core GPU Architecture*. Accessed: 2024-12-23. 2020. URL: <https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf>.
- [12] Daniel J. Costello and G. David Forney. “Channel coding: The road to channel capacity”. eng. In: *Proceedings of the IEEE* 95.6 (2007), pp. 1150–1177. ISSN: 0018-9219.
- [13] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. eng. New York: Wiley, 1991. ISBN: 0-471-06259-6.
- [14] Erik Dahlman, Stefan Parkvall, and Johan Skold. *5G/5G-Advanced: The New Generation Wireless Access Technology*. eng. Third edition. Chantilly: Elsevier Science Technology, 2023. ISBN: 9780443131738.

- [15] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive subgradient methods for on-line learning and stochastic optimization.” In: *Journal of machine learning research* 12.7 (2011).
- [16] Marc P.C. Fossorier. “Quasi-cyclic low-density parity-check codes from circulant permutation matrices”. eng. In: *IEEE transactions on information theory* 50.8 (2004), pp. 1788–1793. ISSN: 0018-9448.
- [17] Robert Gallager. “Low-density parity-check codes”. In: *IRE Transactions on information theory* 8.1 (1962), pp. 21–28.
- [18] Trevor Hastie. *The elements of statistical learning: data mining, inference, and prediction*. 2009.
- [19] Simon Haykin. *Digital communication systems*. eng. 1st ed. Hoboken, NJ: Wiley, 2013. ISBN: 9780471647355.
- [20] Simon. Haykin. *Neural networks : a comprehensive foundation*. eng. 2nd ed. Delhi: Pearson Education, 1999. ISBN: 81-7808-300-0.
- [21] Jakob Hoydis et al. “Sionna: An Open-Source Library for Next-Generation Physical Layer Research”. In: *arXiv preprint* (Mar. 2022).
- [22] Xiao-Yu Hu, E. Eleftheriou, and D.M. Arnold. “Regular and irregular progressive edge-growth tanner graphs”. eng. In: *IEEE transactions on information theory* 51.1 (2005), pp. 386–398. ISSN: 0018-9448.
- [23] IEEE. *IEEE Standard for Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. IEEE Std 802.11ac-2013. Dec. 2013.
- [24] Rinu Jose and Ameenudeen Pe. “Analysis of hard decision and soft decision decoding algorithms of LDPC codes in AWGN”. In: *2015 IEEE International Advance Computing Conference (IACC)*. IEEE. 2015, pp. 430–435.
- [25] Norman P Jouppi et al. “In-datacenter performance analysis of a tensor processing unit”. In: *Proceedings of the 44th annual international symposium on computer architecture*. 2017, pp. 1–12.
- [26] Diederik P Kingma. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).
- [27] Ao Li et al. “Low latency LDPC hard-decision algorithm for 5G NR”. eng. In: *IET circuits, devices systems* 14.2 (2020), pp. 229–234. ISSN: 1751-858X.
- [28] Guangwen Li and Xiao Yu. “A recipe of training neural network-based LDPC decoders”. In: *arXiv preprint arXiv:2205.00481* (2022).
- [29] M.G. Luby et al. “Efficient erasure correcting codes”. eng. In: *IEEE transactions on information theory* 47.2 (2001), pp. 569–584. ISSN: 0018-9448.
- [30] Loren Lugosch and Warren J Gross. “Neural offset min-sum decoding”. In: *2017 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2017, pp. 1361–1365.

- [31] David MacKay. "Information Theory, Inference, and Learning Algorithms". eng. In: *IEEE transactions on information theory* 50.10 (2004), pp. 2544–2545. ISSN: 0018-9448.
- [32] David JC MacKay and Radford M Neal. "Near Shannon limit performance of low density parity check codes". In: *Electronics letters* 33.6 (1997), pp. 457–458.
- [33] David JC MacKay, Simon T Wilson, and Matthew C Davey. "Comparison of constructions of irregular Gallager codes". In: *IEEE Transactions on Communications* 47.10 (1999), pp. 1449–1454.
- [34] Tom M. Mitchell. *Machine learning*. eng. McGraw-Hill series in computer science. New York: McGraw-Hill, 1997. ISBN: 0-07-042807-7.
- [35] Eliya Nachmani, Yair Be'ery, and David 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.
- [36] Ali Naderi et al. "Delayed stochastic decoding of LDPC codes". In: *IEEE Transactions on Signal Processing* 59.11 (2011), pp. 5617–5626.
- [37] Timothy O'shea and Jakob Hoydis. "An introduction to deep learning for the physical layer". In: *IEEE Transactions on Cognitive Communications and Networking* 3.4 (2017), pp. 563–575.
- [38] T. Okamura. "Designing LDPC codes using cyclic shifts". eng. In: *IEEE International Symposium on Information Theory - Proceedings*. IEEE, 2003, pp. 151–. ISBN: 9780780377288.
- [39] John G. Proakis and Massoud. Salehi. *Digital communications*. eng. 5th ed. Boston, MA: McGraw-Hill, 2008. ISBN: 978-0-07-295716-7.
- [40] Tom Richardson and Rüdiger Urbanke. *Modern Coding Theory*. eng. 1st ed. Cambridge: Cambridge University Press, 2008. ISBN: 9780521852296.
- [41] Joachim Rosseel et al. "Decoding short LDPC codes via BP-RNN diversity and reliability-based post-processing". In: *IEEE Transactions on Communications* 70.12 (2022), pp. 7830–7842.
- [42] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.
- [43] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. eng. Global edition. Prentice Hall series in artificial intelligence. Harlow: Pearson Education, Limited, 2016. ISBN: 9781292153964.
- [44] Nemin Shah and Yash Vasavada. "Neural layered decoding of 5G LDPC codes". In: *IEEE Communications Letters* 25.11 (2021), pp. 3590–3593.
- [45] Claude E Shannon and Warren Weaver. *The Mathematical Theory of Communication*. eng. 1st ed. Champaign: University of Illinois Press, 1963. ISBN: 9780252725463.
- [46] J Shrinidhi et al. "Modified Min Sum Decoding Algorithm for Low Density Parity Check Codes". eng. In: *Procedia computer science* 171 (2020), pp. 2128–2136. ISSN: 1877-0509.

- [47] ETSI Standard. "Digital Video Broadcasting (DVB), Second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications (DVB-S2)". In: *European Telecommunications Standards Institute (ETSI) EN 302.307* (2014), p. V1.
- [48] R Tanner. "A recursive approach to low complexity codes". In: *IEEE Transactions on information theory* 27.5 (1981), pp. 533–547.
- [49] Sivarama Prasad Tera et al. "CNN-Based Approach for Enhancing 5G LDPC Code Decoding Performance". In: *IEEE Access* (2024).
- [50] Jeremy Thorpe. "Low-density parity-check (LDPC) codes constructed from protographs". In: *IPN progress report* 42.154 (2003), pp. 42–154.
- [51] Tijmen Tieleman. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude". In: *COURSERA: Neural networks for machine learning* 4.2 (2012), p. 26.
- [52] B. Vasic and O. Milenkovic. "Combinatorial constructions of low-density parity-check codes for iterative decoding". eng. In: *IEEE transactions on information theory* 50.6 (2004), pp. 1156–1176. ISSN: 0018-9448.
- [53] Qing Wang et al. "A model-driven deep learning method for normalized min-sum LDPC decoding". In: *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE. 2020, pp. 1–6.
- [54] Xiaoning Wu, Ming Jiang, and Chunming Zhao. "Decoding optimization for 5G LDPC codes by machine learning". In: *IEEE Access* 6 (2018), pp. 50179–50186.
- [55] Jianguang Zhao, F. Zarkeshvari, and A.H. Banihashemi. "On implementation of min-sum algorithm and its modifications for decoding low-density Parity-check (LDPC) codes". eng. In: *IEEE transactions on communications* 53.4 (2005), pp. 549–554. ISSN: 0090-6778.