Learning Joint Synchronization, Equalization, and Decoding in Short Packet Communications

Master's thesis in Information and Communication Technology

Xi Zhang

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

Learning Joint Synchronization, Equalization, and Decoding in Short Packet Communications

XI ZHANG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2024

Learning Joint Synchronization, Equalization, and Decoding in Short Packet Communications

XI ZHANG

Supervisors: Giuseppe Durisi, Department of Electrical Engineering
Khac-Hoang Ngo, Department of Electrical Engineering
Examiner: Giuseppe Durisi, Department of Electrical Engineering

Master's Thesis 2024
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg

Typeset in LaTeX
Gothenburg, Sweden 2024

Learning Joint Synchronization, Equalization, and Decoding in Short Packet Communications
XI ZHANG
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The rapid evolution of cellular communication technologies necessitates improvements to support emerging applications like autonomous driving and remote medical surgery. Ultra-Reliable Low Latency Communications (URLLC), a key scenario in 5G, demands stringent latency and reliability, with even more rigorous requirements expected in 6G. Traditional communication systems using dedicated preambles for detection, synchronization, and channel estimation is suboptimal for short packet transmissions, highlighting the need for innovative approaches.

This thesis investigates the potential of deep learning (DL) techniques in enhancing short packet communications. By designing an autoencoder-based joint synchronization, equalization, and decoding scheme, the system jointly learns the transmitter and receiver end-to-end for the tasks of synchronization, equalization, and decoding without relying on a dedicated preamble. The objectives include developing an autoencoder-based communication scheme, extending it for joint equalization and decoding, and proposing a joint synchronization, equalization, and decoding scheme under block fading waveform channels.

The findings demonstrate that an end-to-end learning approach using a convolutional neural network-autoencoder (CNN-AE) improves spectral efficiency and reduces overhead in short packet communications while maintaining system reliability. The proposed system, without using dedicated preambles, outperforms the nonasymptotic achievability bound for pilot-assisted transmission systems in terms of block error rate (BLER) at high signal-to-noise ratios (SNRs). This highlights the potential of DL techniques in addressing the challenges of short packet communications in future wireless networks.

Keywords: Short packet communications, Deep learning, Autoencoders, Joint synchronization and decoding.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Giuseppe Durisi and Khac-Hoang Ngo, for their invaluable guidance, insightful feedback and support throughout my journey in completing this master thesis.

I would also like to thank Alireza Bordbar for his suggestions and discussions, and Christian Häger for providing access to the computation resources, which were essential for the completion of this thesis.

The computations of the study were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

Xi Zhang, Gothenburg, August 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

3GPP	3rd generation partnership project
5G	The fifth generation
AE	Autoencoder
ASIC	Application specific integrated circuit
AWGN	Additive white Gaussian noise
BCE	Binary cross-entropy
BER	Bit error rate
BLER	Block error rate
CCE	Categorical cross-entropy
CNN	Convolutional neural network
CNN-AE	Convolutional neural network-autoencoder
CPU	Central processing unit
DL	Deep learning
DSP	Digital signal processing
ELU	Exponential linear unit
FEC	Forward error correction
FPGA	Field-programmable gate array
GPU	Graphics processing unit
LDPC	Low-density parity-check
IoT	Internet of things
ISI	Inter-symbol interference
ML	Machine learning
MSE	Mean squared error
NN	Neural network
PMF	Probability mass function
PSK	Phase shift keying
QAM	Quadrature amplitude modulation
SGD	Stochastic gradient descent
SNR	Signal-to-noise ratio
SISO	Single-input single-output
TurboAE	Turbo autoencoder
URLLC	Ultra-reliable low latency communications

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background

Cellular communication technologies are going through a revolutionary improvement every ten years to support increasing demands of applications, from phone call to videostream and then to the Internet of Things (IoT). The fifth generation (5G) has been commercially deployed and supports various emerging applications like autonomous driving [1], remote medical surgery [2], and intelligent transport systems [3].

Ultra-reliable low latency communications (URLLC), identified as one of the key usage scenarios in 5G, plays an essential role in providing stringent latency and reliability guarantees for mission-critical applications. The 3rd generation partnership project (3GPP) defines that URLLC is expected to provide 99.999% reliability for a single transmission of short packet with an end-to-end latency of less than 1 ms [4]. In 6G, URLLC will enable even more demanding applications, such as tactile internet and augmented reality, with the requirement of even lower latency (25 μs--1 ms) and lower block error rate (BLER) (10^{-5}--10^{-7}) [5].

In URLLC applications, the transmitted messages are rather small. To satisfy such challenging requirements of short packet transmissions, new technical enablers need to be adopted for latency reduction and reliability enhancement [6]. In conventional communication systems, a dedicated preamble is added to the data packet to perform detection, synchronization, and channel estimation. However, in short-packet communications, such preamble-based transmission might be suboptimal due the limited size of the data packet [7], [8].

From an information-theoretic perspective, the results in [8] demonstrate that for a binary-input additive white Gaussian noise (AWGN) channel, performing detection using a dedicated preamble is highly suboptimal in the short-packet regime, even when the length of preamble is optimized. Instead, joint detection and decoding, where the receiver detects the information packet without relying on a dedicated preamble, yields significant gains in terms of the maximum coding rate over the preamble-based detection scheme. Naturally, this leads to the questions: how can we perform joint detection and decoding in practice? How does the performance compare to the theoretical bounds for preamble-based scheme?

Machine learning (ML) techniques are expected to be essential to assist in designing future communication networks. Traditional communication systems divide the transmitter and receiver into different processing blocks and rely on specific channel and system modeling assumptions, that enable tractable mathematically analysis. However, emerging complex communication scenarios are difficult to describe with tractable mathematical models. In this case, machine learning techniques can directly learn and optimize from data and are not constrained by modeling assumptions [9].

In recent years, end-to-end learning of communications systems has become a promising concept to improve the reliability of block-based transmissions [10]. Different from conventional communications systems, this approach represents transmitter and receiver as one deep neural network (NN) that is optimized for an end-to-end performance metric. This can be achieved by interpreting the whole system as an autoencoder which is trained in a supervised manner using stochastic gradient descent (SGD). Such an autoencoder-based system breaks up restrictions in conventional block based signal processing and has shown great benefits in terms of reliability [11], [12]. In [13], Turbo autoencoder (TurboAE) is introduced. It outperforms the state-of-art codes under the AWGN channel model in terms of bit error rate. The authors of [14] propose a convolutional neural network autoencoder (CNN-AE) which approaches the theoretical maximum achievable rate over the AWGN channel.

In this thesis, we investigate the potential of applying deep learning (DL) techniques in short packet transmissions. We focus on designing an autoencoder-based joint synchronization, equalization, and decoding scheme. We aim to compare the end-to-end performance with the theoretical bound for the pilot-assisted transmission system.

1.2 Objectives

During the thesis, we aim to:

- Design a DL-based communication scheme for short packets, and investigate its performance compared with achievability bounds and state-of-art channel codes under an AWGN channel.

- Extend the DL-based scheme to handle both equalization and decoding, then compare the scheme's performance with an achievability bound under a block fading channel.

- Propose a DL-based joint synchronization, equalization, and decoding scheme, and compare both the synchronization and decoding performance with an achievability bound for pilot-assisted transmission scheme under a block fading waveform channel with imperfect synchronization.

1.3 Limitations

The work primarily focuses on the simulations for deep learning applications in physical layer of communication systems. Consequently, hardware implementations have not been considered. Modern communication systems mainly use application-specific integrated circuit (ASIC) and field-programmable gate array (FPGA), while neural networks need to be processed on central processing units (CPUs) and graphics processing units (GPUs). The performance of NN-based systems is limited by the computation and memory capabilities of the hardware.

Another limitation is that practical channels have not been considered in this thesis work. End-to-end learning communications systems require a differentiable channel model, so that gradients can be backpropagated through the channel. However, in practice, the channel is generally a black box, where only inputs and outputs can be observed. Several works have been done for solving this problem in [11], [15], but they are not covered in this thesis.

1.4 Thesis Outline

The thesis is divided into five chapters. Chapter 2 briefly introduces the classical communication system, as well as the basics of deep learning. Chapter 3 presents the considered transmission scenario, and the proposed autoencoder-based scheme, along with the training procedure and parameters. Chapter 4 introduces the performance comparison between the proposed joint synchronization, equalization and decoding scheme and the achievability bound for pilot-assisted transmission in the finite block-length regime. Finally, a brief summary and discussion of the contribution is presented in Chapter 5.

1.5 Notation

We denote scalar random variables by upper case letters, such as X, and their realizations by lower case letters, e.g., x. Bold-faced upper case letters denote random vectors, e.g., $\mathbf{X}$, and their realizations are denoted by lower case letters, e.g., $\mathbf{x}$. The superscripts $(\cdot)^T$, $(\cdot)^H$ and $(\cdot)^*$ denote transposition, Hermitian transposition, and complex conjugation, respectively. We denote the set of real numbers by $\mathbb{R}$ and the set of complex numbers by $\mathbb{C}$. The distribution of a complex Gaussian random variable with mean μ and variance σ^2 is denoted by $\mathcal{CN}(\mu, \sigma^2)$. We write $\log(\cdot)$ to denote the natural logarithm. Probabilities are written as $\mathbb{P}[\cdot]$ and the expectation operation is denoted by $\mathbb{E}[\cdot]$. The notation $\|\cdot\|$ stands for the ℓ^2-norm. For two functions $f(n)$ and $g(n)$, the notation $f(n) = \mathcal{O}(g(n))$ means that $\limsup_{n\to\infty} |f(n)/g(n)| < \infty$.

2

Theory

This chapter provide a brief introduction to the general theory of communication systems and DL.

2.1 Classical Communication System

The objective of a communication system is to transmit information reliably from one point to another. A classical single-input single-output (SISO) communication system includes a transmitter, a channel, and a receiver, as illustrated in Figure 2.1. These blocks are described as follows:

- **Transmitter**: Encodes a message W into a codeword $\mathbf{x} \in \mathbb{C}^n$ and transmits this codeword over n complex-valued channel uses. Here, W is assumed to be drawn uniformly from the alphabet $\left\{1, 2, ..., 2^{nR}\right\}$, where R is the transmission rate. We consider the power constraint $\|\mathbf{x}\|^2 \leq n$.

- **Channel**: Adds propagation distortion to the transmitted signal, specified by a conditional probability mass function (PMF) $P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$.

- **Receiver**: Constructs an estimate $\hat{W}$ of the original message W from the noisy observation $\mathbf{y} \in \mathbb{C}^n$ of the transmitted signal. The reliability is measured by the block error probability $P_e = \Pr\left[\hat{W} \neq W\right]$.

The following subsections will provide a detailed explanation for each parts in the system.

Figure 2.1: Illustration of a simple communication system.

2.1.1 Classical Transmitter

Figure 2.2 shows the basic components of a transmitter in a general communication system. The transmitter feeds the binary representation of its message to a channel encoder, then the coded bits are mapped to the real or complex-valued symbols specified by a modulation format. After that, a dedicated preamble sequence (also called pilot) is added in front of the symbols. The preamble is then used for synchronization and channel estimation at the receiver. Finally, through pulse shaping, both preamble sequence and payload are transformed into a waveform transmitted through the channel. We next describe each component.

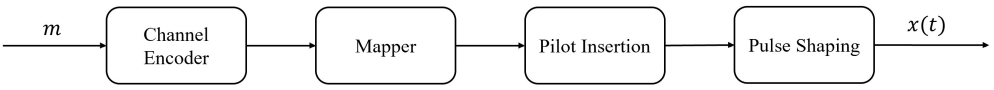

Figure 2.2: Block diagram of a conventional transmitter.

Channel Encoder

The objective of the channel encoder is to introduce redundancy to the transmitted bit sequence so that the receiver can correct errors introduced by the channel. Consider the encoder as a function $f : \left\{1, ..., 2^k\right\} \rightarrow \mathcal{X}^n$ that maps the message $m \in \left\{1, ..., 2^k\right\}$ into a codeword $x^n(m) = [x_1(m), ..., x_n(m)]$, where $\mathcal{X} = \{0, 1\}$ and $x_l(m) \in \mathcal{X}, l = 1, ..., n$. Here, m is drawn uniformly from the message set $\left\{1, ..., 2^k\right\}$, and 2^k is the number of codewords in the codebook $\left\{x^n(1), ..., x^n\left(2^k\right)\right\}$. Each message is described using k bits, the code rate R_c is defined as $R_c = k/n \leq 1$, which represents the spectral efficiency.

By mapping 2^k possible information sequences to a larger space of 2^n, the distance between the valid codewords and arbitrary binary sequences increases. This enhances the likelihood of reconstructing the correct codeword from a noisy observation. Channel codes are often referred to as forward error correction (FEC) or error correction codes. Over the years, different types of codes have been invented, starting with classical codes such as Hamming codes and convolutional codes, and evolving to modern coding schemes like low-density parity-check (LDPC) codes [16], and Turbo codes [17]. These modern codes are widely used in current communication systems due to their near Shannon limit performance.

Mapper

After the channel encoder, the mapper maps each codeword to a real or complex-valued symbol specified by a constellation, denoted as $\mathcal{C} = \{c_1, ..., c_M\} \subset \mathbb{C}^M$. For a constellation with M symbols, $\log M$ denotes the number of bits per symbol. Each symbol is defined as $c = c_I + jc_Q$, where c_I represents the in-phase component and c_Q the quadrature component.

Common choices of constellation are phase shift keying (PSK) and quadrature amplitude modulation (QAM). A higher order constellation (larger M) allows more bits per symbol, enhancing spectral efficiency. However, the Euclidean distance between symbols decreases, leading to an increased error probability in symbol decoding under the presence of noise.

Pilot Insertion

Typically, in order to synchronize the transmitted signal and estimate the channel gain, a dedicated preamble sequence is added in front of the payload. The preamble sequence usually has good auto-correlation properties, such as m-sequences [18] and Zadoff-Chu preamble sequences [19].

Pulse Shaping

After mapping the coded bits to symbols of the chosen constellation, pulse shaping filtering is usually employed to transform the symbols into the waveform, in order to limit the bandwidth and reduce the inter-symbol interference (ISI) [20].

Consider a simple case of performing square pulse shaping on transmitted symbols. The square pulse with normalized energy can be expressed as:

$$s(t) = \begin{cases} \frac{1}{\sqrt{t_p}}, & t \in [0, t_p)\,, \\ 0, & \text{otherwise}\,, \end{cases}$$

where t_p denotes the size of the pulses. Subsequently, the transmitted signal can be obtained by convolving the symbols with the square pulse, i.e.,

$$x(t) = \sum_{k=1}^{n_s} x_k s\left(t - (k-1)\, t_p\right),$$

where n_s is the length of the transmitted symbols.

Square pulse shapes are not practical in nowadays communication systems as they require large amount of bandwidth. Common choices of the pulse shapes include sinc shapes, raised-cosine shapes, and root-raised-cosine shapes. Among them, the root-raised-cosine shapes result in a higher spectral efficiency.

2.1.2 Classical Receiver

Figure 2.3 shows a block diagram of a conventional receiver. In this scenario, we consider that the transmitted signal experiences fading propagation with a random

time delay, and both the transmitter and receiver have no prior knowledge of the channel. To reconstruct the transmitted message at the receiver, the received signal needs to be processed by digital signal processing (DSP) blocks, which include synchronization, channel estimation, and equalization.

- **Synchronization**: The receiver needs to synchronize the received signal by estimating the time delay of the start of the signal within the received sequence.
- **Channel estimation**: Channel estimation involves estimating the characteristics of the channel between the transmitter and receiver. After synchronization, the receiver estimates the fading gains from the received signal to compensate for distortions induced by channel subsequently.
- **Equalization**: Equalization is performed to mitigate the effects of ISI and noise introduced by the channel, thereby recovering the transmitted symbols.

These steps can be performed in different sequences depending on the receiver design, and the depicted process does not include all possible techniques in the receiver. In the following subsections, we will briefly review the algorithms from the literature that implement these DSP blocks. For more detailed reviews, we refer to [21].

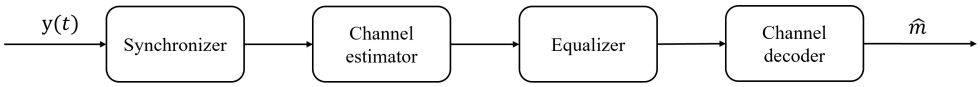

Figure 2.3: Block diagram of a conventional receiver, considering the task of synchronization, equalization, and decoding.

Synchronization

In the first step, the synchronizer needs to estimate the time delay of the received signal. The estimation of the time delay can be achieved using either pilot-assisted or by blind methods. Here we focus on pilot-assisted estimation.

Pilot-assisted synchronization involves embedding a known pilot sequence into the transmitted signal. Upon receiving the signal, the receiver calculates the cross-correlation between the received signal and the pilot sequence. The peak of the cross-correlation result indicates the presence of the pilot sequence in the received signal; thus, the position of the peak corresponds to the estimated time delay. By convolving the received signal with the dedicated pilot sequence and comparing the cross-correlation results to a certain threshold, the beginning of the signal can be

identified, thus achieving synchronization of the received signal. This pilot-assisted method is robust and widely used in practical communication systems.

Channel Estimator

When the channel state information (CSI) is unknown at the receiver, a pilot sequence is typically used for channel estimation. Consider a simple case where the received signal is perfectly synchronized, the signal can be expressed as:

$$\mathbf{y}^{(p)} = H\mathbf{x}^{(p)} + \mathbf{z}^{(p)},$$

where $\mathbf{x}^{(p)}$ is the pilot sequence of the signal, H is the random fading gain, and $\mathbf{z}^{(p)}$ is the complex AWGN. To estimate H, we first derive expression of H as:

$$H = \frac{\left(\mathbf{x}^{(p)}\right)^{\mathsf{H}} \mathbf{y}^{(p)}}{\left\|\mathbf{x}^{(p)}\right\|^2} - \frac{\left(\mathbf{x}^{(p)}\right)^{\mathsf{H}} \mathbf{z}^{(p)}}{\left\|\mathbf{x}^{(p)}\right\|^2}.$$

Therefore, the maximum likelihood estimation (MLE) is applied as:

$$\hat{h} = \arg\min_{\tilde{h}} \left\|\mathbf{y}^{(p)} - \tilde{h}\mathbf{x}^{(p)}\right\|^2 = \frac{\left(\mathbf{x}^{(p)}\right)^{\mathsf{H}} \mathbf{y}^{(p)}}{\left\|\mathbf{x}^{(p)}\right\|^2}.$$

Equalizer

Given the estimated channel gain from channel estimation, the equalizer works to remove the ISI and noise effects from the channel and recover the transmitted symbols. Common digital linear equalizers include zero-forcing (ZF) equalizer, minimum mean square error (MMSE) equalizer.

Consider the following model:

$$\mathbf{y} = \hat{h}\mathbf{x} + \mathbf{z},$$

where $\mathbf{y}$ is the received data signal, $\hat{h}$ is the estimated channel gain, which is assumed to be perfectly estimated $\hat{h} = h$; $\mathbf{x}$ is the transmitted symbols, and $\mathbf{z}$ is the noise vector. The ZF equalizer fully inverses the impact of the channel. It applies the inverse of the channel gain as:

$$G_{\mathrm{ZF}} = \left(h^* h\right)^{-1} h^* = \frac{h^*}{|h|^2},$$

and then, the estimated transmitted symbol can be expressed as:

$$\hat{\mathbf{x}} = G_{\mathrm{ZF}}\mathbf{y} = \mathbf{x} + \frac{h^*}{|h|^2}\mathbf{z}.$$

It is important to note that ZF equalizers ignore the additive noise and may amplify noise, especially in scenarios where the channel gain is small. This can be mitigated by using an MMSE equalizer, which minimizes the mean square error between the output of the equalizer and the transmitted signals.

2.1.3 Channel Models and Fundamental Limits

The channel introduces distortion to the transmitted signal. In this section, we describe two channel models considered in the thesis, and also the fundamental limits on these channels.

AWGN Channel

AWGN channel is a commonly used channel model in communication systems. Consider a discrete-time memoryless AWGN channel given by:

$$\mathbf{y} = \mathbf{x} + \mathbf{z},$$

where $\mathbf{x}$ is the input of the channel (i.e., the transmitted symbols) each with average energy per symbol E_s, each element in $\mathbf{z}$ is independent and identically distributed and drawn from the complex Gaussian distribution with zero mean and variance N_0. The signal-to-noise ratio (SNR) is defined as:

$$\text{SNR} = \rho = \frac{E_s}{N_0}.$$

The channel capacity, as given by Shannon's theorem [22], is

$$C = \log(1 + \rho) \text{ bits/channel use},$$

which indicates the ultimate limit of the information can be transmitted reliably over the channel, as the code length goes to infinity.

Shannon's channel coding theorem states the largest communication rate at which we can transmit messages over a channel with a vanishing error probability for sufficient large blocklength. This theorem proved that the maximum achievable rate for which the error probability $\varepsilon \to 0$ as blocklength $n \to \infty$ is C.

However, for short packet transmissions where the blocklength is relatively small, Shannon's channel capacity might be a loose upper bound on the achievable coding rate. In this scenario, finite blocklength information theory provides a more precise characterization.

Finite Blocklength Information Theory

In finite blocklength regime, achievability bounds (e.g., random coding union bound [23], and random coding union bound with parameter s) and converse bounds (e.g., metaconverse bound [23]) are provided. The achievability bound is an upper bound on error probability indicating the performance that can be achieved by suitable encoding and decoding schemes. In contrast, the converse bound is a lower bound on error probability representing the performance that cannot be outperformed by any choice of encoding and decoding schemes. The computation of both the achievability and converse bounds is difficult. Therefore, asymptotic expansions of both

bounds, such as the normal approximation [23] and the saddlepoint approximation [24], are often used to yield numerical approximation.

The maximal coding rate $R^*(n,\epsilon)$ is defined as the maximum rate that can be achieved for a fixed error probability ϵ and finite blocklength n. For various channels with capacity C, $R^*(n,\epsilon)$ can be characterized as given in [23]:

$$R^*(n,\epsilon) = C - \sqrt{\frac{V}{n}}Q^{-1}(\epsilon) + \mathcal{O}\left(\frac{\log n}{n}\right),$$

where Q^{-1} denotes inverse of Q-function and V is the channel dispersion defined as $V = \rho(2+\rho)/(1+\rho)^2$. The term $\mathcal{O}(\log n/n)$ comprises higher-order terms of order $\log n/n$.

For large blocklength n and small block error rates ϵ, the maximal coding rate $R^*(n,\epsilon)$ approaches channel capacity, i.e., $R^*(n,\epsilon) \approx C$. However, for short blocklength, a more precise approximation can be derived.

Consider the real-valued AWGN channel with noise variance $\sigma^2 = 1$. The transmitted symbols $\mathbf{x}$, with blocklength n, satisfy the power constraint $\|\mathbf{x}\|^2 \leq n\rho$. The normal approximation for $R^*(n,\epsilon)$ as given by [23] can be expressed as:

$$R^*(n,\epsilon) \approx C(\rho) - \sqrt{\frac{V(\rho)}{n}}Q^{-1}(\epsilon) + \frac{\log n}{2n},$$

where $C(\rho)$ and $V(\rho)$ denotes the Gaussian capacity and dispersion respectively:

$$C(\rho) = \log(1+\rho)$$

$$V(\rho) = \rho\frac{(2+\rho)}{(1+\rho)^2}(\log e)^2.$$

The converse bounds, such as meta-converse bound, and the achievability bounds, including the Shannon cone-packing bound, $\kappa\beta$ bound, and Gallager's bound are detailed in [23].

The achievability and converse bounds for the binary-input AWGN (bi-AWGN) channel are detailed in [25]. Given the code parameters and the channel, the BLER can be calculated by these bounds. In this thesis, we will use the achievability bound as the benchmark to evaluated the BLER performance our proposed system.

Block-Fading Channel

A block-fading channel is shown in Figure 4.3. Consider the transmission of a sequence of n complex-valued symbols over a SISO memoryless block-fading channel with the number of fading blocks n_b. The received symbols $\mathbf{y}_i, i = 1, ..., n$ can be expressed as:

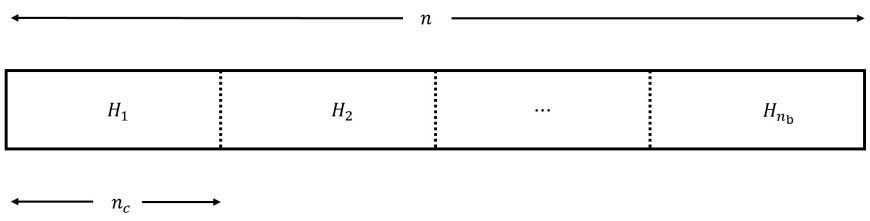

Figure 2.4: A simple model of block fading channel.

$$\mathbf{y}_i = H_j \mathbf{x}_i + \mathbf{z}_i, j = 1, ..., n_b,$$

where $H_j, j = 1, ..., n_b$ denotes the random fading gain for the jth fading block, and $\mathbf{z}_i$ denotes the independent AWGN of the ith block. The coherence time is denoted as n_c. The fading is constant during a block of n_c symbols and independent from block to block. Large number of independent fading blocks increases the channel diversity while few number of blocks increases the chance for experiencing deep fading [26].

In the context of transmission over block-fading channels, it is crucial for the transmitter, receiver, or both to have knowledge of the fading coefficient H. CSI known at the transmitter enables efficient power allocation strategies, such as waterfilling. When CSI is available at the receiver, it facilitates the decoding. In practice, CSI at the receiver is typically obtained by transmitting dedicated pilot sequences, which the receiver uses to estimate the channel. CSI at the transmitter can be acquired by feeding the channel estimates from the receiver back to the transmitter. However, transmitting pilot sequences introduces a rate loss, and establishing a feedback link incurs additional costs.

Two common infinite blocklength performance metrics for communication over fading channels are the ergodic capacity and the outage capacity. The ergodic capacity represents the maximum achievable rate of reliable communication over a fading channel, averaged over all channel states. The outage capacity, on the other hand, characterizes the maximum transmission rate at which the probability of the instantaneous channel capacity falling below this rate does not exceed a specified outage probability ϵ.

Consider the scenario where $n_b = 1$, the outage probability for a given rate R can be expressed as:

$$P_{\text{out}}(R) = \mathbb{P}\left[\log\left(1 + |H|^2 \rho\right) < R\right].$$

The outage capacity C_ϵ is defined as the supremum of all rates R satisfying $P_{\text{out}} \leq \epsilon$. It is given by:

$$C_\epsilon = \sup\left\{R : P_{\text{out}} \leq \epsilon\right\}.$$

The outage capacity C_ϵ implies that, for every realization of the fading coefficient $H = h$, the channel behaves like an AWGN channel with a channel gain $|h|^2$. In this context, communication with an arbitrarily small error probability is achievable for sufficiently large blocklength n if and only if the rate satisfies $R < \log\left(1 + |h|^2 \rho\right)$. However, the work by [27] highlights that the expression $\log\left(1 + |h|^2 \rho\right)$ is meaningful only for sufficient large blocklengths. In the same paper, the maximum coding rate $R^*(n, \epsilon)$ for a given blocklength n and block error probability ϵ is refined to account for finite blocklengths and can be expressed as:

$$R^*(n, \epsilon) = C_\epsilon + \mathcal{O}\left(\frac{\log n}{n}\right),$$

which holds regardless of whether CSI is available at the transmitter, receiver, or both.

The normal approximation to the maximal achievable rate is also provided in [27] for single-antenna case. Additionally, the authors of [28] derive the achievability bounds and converse bounds on maximum coding rate over the multiple-antenna Rayleigh block-fading channel model. Both results provide the accurate performance metrics when blocklength is relatively small.

2.2 Deep Learning Basics

This chapter provides a brief introduction to the general theory behind deep learning and autoencoder.

2.2.1 Neural Networks

Neural networks are adaptive statistical models capable of representing complex functions through the composition of simple operations. Consider a feedforward NN, which is a function $f(\mathbf{r}_0; \boldsymbol{\theta}) : \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$ that maps an input vector $\mathbf{r}_0 \in \mathbb{R}^{N_0}$ to an output vector $\mathbf{r}_L \in \mathbb{R}^{N_L}$ through L iterative processing steps:

$$\mathbf{r}_\ell = f_\ell\left(\mathbf{r}_{\ell-1}; \theta_\ell\right), \ell = \{1, ..., L\},$$

where L is the number of layers, and $f_\ell(\mathbf{r}_{\ell-1}; \theta_\ell) : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}$ is the mapping carried out by the ℓth layer. This mapping depends on both the output vector $\mathbf{r}_{\ell-1}$ from the previous layer and the set of parameters $\boldsymbol{\theta} = \{\theta_1, ..., \theta_L\}$.

A commonly used layer is called dense layer, also known as fully-connected layer. It has the form:

$$f_\ell\left(\mathbf{r}_{\ell-1}; \theta_\ell\right) = \sigma\left(\mathbf{W}_\ell \mathbf{r}_{\ell-1} + \mathbf{b}_\ell\right),$$

where $\mathbf{W}_\ell \in \mathbb{R}^{N^\ell \times N^{\ell-1}}$ is the weight matrix, $\mathbf{b}_\ell \in \mathbb{R}^{N^\ell}$ is the bias vector, and $\sigma(\cdot)$ is an activation function which introduces non-linearity of the output [29]. The set of trainable parameters for this layer is $\theta_\ell = \{\mathbf{W}_\ell, \mathbf{b}_\ell\}$.

A fully-connected NN is a type of neural network where all layers are dense layers. In the fully-connected NN, each neuron in one layer is connected to every neuron in the subsequent layer, allowing for the efficient transmission of information throughout the network. Another popular NN is the convolution neural network (CNN). Compared to fully-connected NNs, CNNs are more efficient and effective for tasks involving structured data [30].

Consider a 2D convolutional layer consisting a set of F trainable filter weights $\mathbf{Q}^f \in \mathbb{R}^{a \times b}$, where $f = 1, ..., F$ and F is the depth of the layer. This layer maps an input matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$ to a feature map $\mathbf{Y}^f \in \mathbb{R}^{n' \times m'}$ according to:

$$\mathbf{Y}^f_{i,j} = \sum_{k=0}^{a-1} \sum_{\ell=0}^{b-1} Q^f_{a-k,b-\ell} X_{1+s(i-1)-k,1+s(j-1)-\ell},$$

where $s \geq 1$ is call stride. It denotes the step size for the convolution, specified by a positive integer. The output size can be calculated as $n' = 1 + \left\lfloor \frac{n+a-2}{s} \right\rfloor$ and $m' = 1 + \left\lfloor \frac{m+b-2}{s} \right\rfloor$. In convolutional layers, the filter slides across an input vector with a certain stride, tying adjacent shifts of the same weights together. Consequently, convolutional layers reduce the model complexity compared to dense layers [9].

2.2.2 Autoencoder

An autoencoder is a unsupervised learning framework designed to learn latent representations by minimizing the reconstruction loss of its input data [31]. An example of an autoencoder is shown in Figure 2.5.

The network consists of two parts: an encoder function that transforms the input data into latent representation $\mathbf{h} = f(\mathbf{x})$ and a decoder that produces a reconstruction $\mathbf{r} = g(\mathbf{h})$, where $\mathbf{h}$ is the latent space that describes a code used to represent the input. Most autoencoders are undercomplete autoencoders, meaning the latent space $\mathbf{h}$ has a smaller dimension than the input data $\mathbf{x}$. Learning an undercomplete representation forces the autoencoder to capture the most essential features of the data [32]. The learning process can be described as minimizing a loss function $L(\mathbf{x}, g(f(\mathbf{x})))$.

2.2.3 Loss Functions

The goal of training a NN is to minimize a chosen loss function. Commonly used loss functions include mean squared error (MSE) loss, binary cross-entropy (BCE) loss, categorical cross-entropy (CCE) loss. The following subsection provides a review of the BCE and CCE, which are used in this thesis.

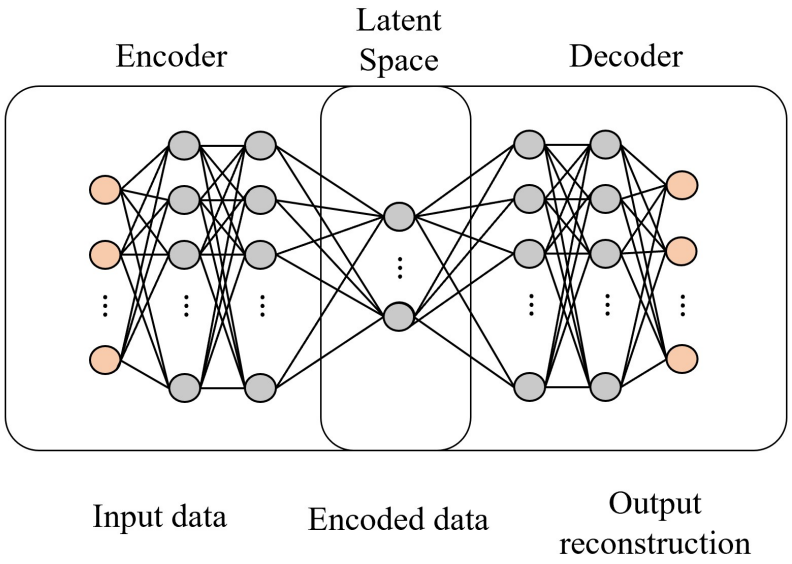

Figure 2.5: An autoencoder composed of two NNs.

Binary Cross-Entropy Loss

BCE loss is commonly used for binary classification tasks. It measures the performance of a classification algorithm whose output is a probability between 0 and 1. In a binary classification task, each class label is denoted by a scalar $s \in \{0, 1\}$. A NN can be designed to output a probability q for a given input r according $q = f(r; \boldsymbol{\theta})$, where q describes the probability of the input belonging to positive class. Here, $\boldsymbol{\theta}$ represents the trainable parameters in the NN.

Given this model and a training dataset $\mathcal{D}$ consisting of $|\mathcal{D}|$ input-output pairs, the BCE loss is defined as:

$$L_{\mathrm{BCE}}(\boldsymbol{\theta}) = -\frac{1}{|\mathcal{D}|} \sum_{(r,s)\in D} s \log\left[f(r; \boldsymbol{\theta})\right] + (1-s) \log\left[1 - f(r; \boldsymbol{\theta})\right],$$

where $f(r; \boldsymbol{\theta})$ is the probability of the input r being in class s and $1 - f(r; \boldsymbol{\theta})$ is the probability of the input r being in the other class $1 - s$.

Categorical Cross-Entropy Loss

CCE is commonly used for multi-class classification problems. In contrast to binary classification tasks, the class label $s \in \{0, 1, ..., C-1\}$ where C is the number of classes. Assume $\mathbf{p}$ is the probability vector associated with the true class label s. The CCE loss can be expressed as:

$$L_{\mathrm{CCE}}(\boldsymbol{\theta}) = -\frac{1}{|\mathcal{D}|} \sum_{(r,s)\in D} \ell_{\mathrm{CE}}(\mathbf{p}, \mathbf{q}),$$

where $\ell_{\mathrm{CE}}(\mathbf{p}, \mathbf{q}) = -\sum_{c=1}^{C} p_c \log q_c$ is the cross entropy, measuring the difference between two distributions $\mathbf{p}$ and $\mathbf{q}$. Here, q_c is the predicted probability for class c and p_c is the true probability (which is 1 for the correct class and 0 otherwise).

2.2.4 Gradient-based Learning

The goal of the training algorithms is to find a good set of parameters $\boldsymbol{\theta}$ that minimize the chosen loss function. This is typically achieved by optimizing the loss function $L(\boldsymbol{\theta})$. The problem can be expressed as:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}),$$

where $\boldsymbol{\theta}^*$ is the optimal parameters that minimize the loss function.

To solve this optimization problem, gradient-based techniques such as stochastic gradient descent (SGD) are often used. In SGD, the parameters $\boldsymbol{\theta}$ are updated iteratively using the gradient of the loss function with respect to $\boldsymbol{\theta}$. At each iteration,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} \tilde{L}(\boldsymbol{\theta}_t),$$

where $\boldsymbol{\theta}_t$ denotes the parameters at iteration t, η is the learning rate, and $\tilde{L}(\boldsymbol{\theta}_t)$ is an approximation of the loss function using a random mini-batch $\mathcal{B}_t$ of the entire training samples $\mathcal{D}$ at iteration t. This mini-batch $\mathcal{B}_t \subset \mathcal{D}$, and it helps to reduce the computational cost. The gradient $\nabla_{\boldsymbol{\theta}} \tilde{L}(\boldsymbol{\theta}_t)$ can be efficiently calculated using back-propagation algorithm [33].

The choice of the learning rate influences the convergence rate, a very large learning rate might cause the algorithm to diverge, while a very low learning rate makes it slow to converge. There are many variants of the SGD proposed to improve the convergence [32], such as the momentum method [34], RMSProp method [35], and Adam optimizer [36]. These optimizers adjust the learning rate during the training based on the gradients, which helps to avoid the local minima and speeding up the convergence.

3
Methods

Autoencoder can be used to assist the design of communication systems. In this chapter, we first introduce a simple AE-based end-to-end learning of communication system and its training procedure. Then, we introduce the proposed CNN-AE based joint synchronization, equalization, and decoding system.

3.1 An AE-based Communication System

Consider a simple communication system as shown in Figure 2.1. From a deep learning perspective, this system can be viewed as a particular type of autoencoder. In this scenario, the encoder acts as the transmitter, learning a representation $\mathbf{x}$ of the message W in a manner robust to the channel impairments. The decoder functions as the receiver, attempting to recover the message from the channel observation $\mathbf{y}$ with a low probability of error. This concept was initially proposed in [9], where both the transmitter and receiver are replaced by multiple dense layers. In this section, we introduce a CNN-AE based scheme, where the transmitter and receiver are comprised of a set of CNN layers. The structure of the CNN-AE is based on the model proposed in [14]. However, we adjust the parameters and number of layers for the purposes of this thesis.

The structure of the CNN-AE is shown in Figure 3.1, where we model four blocks (channel encoder, modulator, demodulator and channel decoder) as CNN blocks and train them jointly in an end-to-end manner. Given an information bit sequence $\mathbf{u} \in \{0,1\}^k$ of length k, the transmitter outputs n complex-valued symbols $\mathbf{x} \in \mathbb{C}^n$. These symbols are then propagated through the channel, resulting in a noisy observation $\mathbf{y} \in \mathbb{C}^n$ of the transmitted signal. The receiver takes this noisy observation and compensates for transmission impairments to output an estimate of the transmitted bit sequence $\hat{\mathbf{u}}$. The goal of the end-to-end learning is to find suitable parameters for the AE such that the transmitter learns a signal representation that is robust to channel impairments, while the receiver learns reliable reconstructions of the transmitted bits from the channel observation. This end-to-end training ensures that the entire communication system is optimized holistically, leading to improved performance in terms of error rates and robustness to noise.

The CNN-AE structure is designed to mimic the blocks of a conventional communication system. Each block of the CNN-AE is represented by a set of 1-dimensional convolutional (Conv1D) layers, where the dimensions are chosen based on the spe-

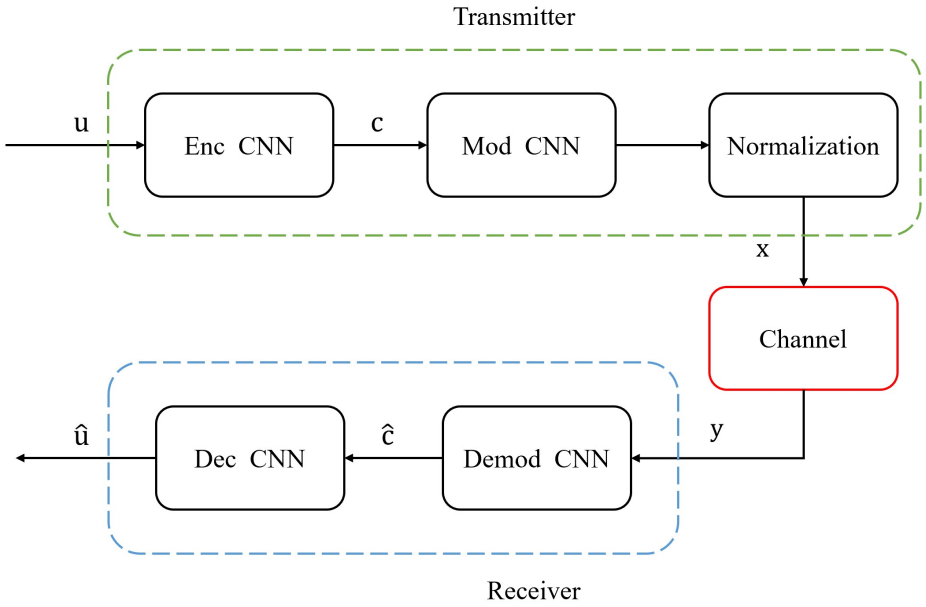

Figure 3.1: The structure of a CNN-AE based communication system, where the traditional transmitter and receiver is replaced by CNNs.

cific function of each block. Compared to fully connected layers, Conv1D layers offer lower complexity and better trainability.

The detailed structure is shown in Table 3.1, we will introduce each block and training procedure in the following subsections.

Transmitter

The transmitter maps a message with k information bits to n complex-valued symbols. The Enc CNN first works as a channel encoder, mapping the bit sequence into n_c coded sequence at a code rate $R_{\text{cod}} = k/n_c$. Then, the Mod CNN functions as a modulator, mapping the n_c coded sequence into n complex-valued symbols with the modulation order $m = n_c/n$. The overall communication rate is overall $R = k/n$ bits per complex channel use.

For simplicity, we set l as the greatest common divisor of n_c and k, thus $k' = k/l$ and $n'_c = n_c/l$. This allows us to interpret the encoding of k bits into n_c coded bits as the encoding of l sub-bits of k' bits into l sub-codewords of n'_c bits. This transformation enables the AE to fit different code rates easily.

- Enc CNN: We apply five Conv1D layers to function as a channel encoder. The first four layers map the information bits into a higher dimensional space, allowing the AE to learn an effective placement of the bit sequence. The final layer maps the sub-codewords down to a lower dimensional space, with the output reshaped into a matrix of size (n, m) for modulation. Convolutional

Layer	Activation	Output dimensions
	Enc CNN	
Conv1D	ELU	$(k,100)$
Conv1D	ELU	$(l,100)$
Conv1D	ELU	$(l,100)$
Conv1D	ELU	$(l,100)$
Conv1D	ELU	(l,n_c')
Reshape		(n,m)
	Mod CNN	
Conv1D	ELU	$(n,100)$
Conv1D	ELU	$(n,100)$
Conv1D	ELU	$(n,100)$
Conv1D	Linear	$(n,2)$
	Demod CNN	
Conv1D	ELU	$(n,100)$
Conv1D	ELU	$(n,100)$
Conv1D	ELU	$(n,100)$
Conv1D	Linear	(n,m)
Reshape		(l,n_c')
	Dec CNN	
Conv1D	ELU	$(l,100)$
Conv1D	ELU	$(l,100)$
Conv1D	ELU	$(l,100)$
Conv1D	ELU	$(l,100)$
Conv1D	Sigmoid	$(k,1)$

Table 3.1: Parameters of the CNN-AE, each Conv1D layer is followed by a batch normalization layer before activation to help the model converge quickly.

operations enable linear coding, while exponential linear unit (ELU) activation functions allow potential non-linear operations as :

$$\mathrm{ELU}\,(z) = \begin{cases} z, & z > 0, \\ e^z - 1, & z \leq 0, \end{cases}$$

The use of ELU typically speeds up the learning and reduces errors [37].

- Mod CNN: We use four Conv1D layers to modulate the n symbols, each with m symbols. The first three layers map m symbols into a higher dimensional space. The final layer maps each of the n modulated 100-dimensional symbols into a 2-dimensional real-valued symbols, representing the real and imaginary components of the transmit symbols.

- Normalization: A non-trainable normalization is added to satisfy the average power constraint $\mathbb{E}\left[\|\mathbf{x}\|^2\right] = n\rho$, where ρ denotes the SNR.

The normalized signal $\mathbf{x}$ is then transmitted over the channel. We assume an AWGN channel is used, i.e., $\mathbf{y} = \mathbf{x} + \mathbf{z}$. The noise $\mathbf{z}$ is an n-dimensional vector of independent and identically distributed complex Gaussian noise with zero mean and unit variance.

Receiver

The receiver takes the channel observation as input and attempts to estimate the transmitted message. The demod CNN functions as a demodulator, and the Dec CNN functions as a channel decoder. Both the demod CNN and Dec CNN are designed in the same manner to reconstruct the transmitted message. Key components include:

- Sigmoid activation function: The Sigmoid function takes a real value as input and outputs a value between 0 and 1. It is expressed as:

$$S(z) = \frac{1}{1 + e^{-z}}.$$

  The Sigmoid outputs can be interpreted as the estimated posterior probabilities of the bits being 0 or 1. The closer the output value is to 0, the more likely the bits is 0, and vice versa.

- Decision: Since the output of the last layer is a posterior probability between 0 and 1, a threshold of 0.5 is applied to ensure the output is a binary vector, representing the estimated transmitted bits.

Training procedure and parameters

We train the CNN-AE by optimizing the total BCE loss between originally transmitted bit sequence $\mathbf{u}$ and the estimated bit sequence $\hat{\mathbf{u}}$ at the receiver output. The training process involves adjusting all trainable parameters in an end-to-end manner using SGD. The detailed training procedure is outlined in Algorithm 1.

The joint training algorithm only works when the channel model is differentiable; otherwise, the gradients cannot be backpropagated through the channel. In [11], the authors propose an alternating training algorithm. In this approach, at each iteration, the receiver is optimized while keeping the transmitter parameters θ_{tx} fixed, and the transmitter is optimized while keeping the receiver parameters θ_{rx} fixed. By following this procedure, the system achieves faster convergence. The details of the alternating training method are provided in Algorithm 2.

The training parameters are listed in Table 3.2. It is important to note that while the BCE loss function optimizes the BER, it does not directly result in optimal BLER. In this thesis, we have chosen to optimize the BCE loss, which is sufficient

Algorithm 1 Training Procedure for CNN-AE

Input: Number of Epoch M, Training Step T, Training SNR $\sigma^2_{\text{min}}, \sigma^2_{\text{max}}$, Training parameter θ_{tx}, θ_{rx}
Output: $\theta_{\text{tx}}, \theta_{\text{rx}}$
for $i \leq M$ **do**
  for $j \leq T$ **do**
    $\sigma^2 \longleftarrow$ generate_SNR ($[\sigma^2_{\text{min}}, \sigma^2_{\text{max}}]$)
    $\mathbf{u} \longleftarrow$ generate_bits ()
    $\mathbf{x} \longleftarrow$ transmit ($\mathbf{u}; \theta_{\text{tx}}$)
    $\mathbf{y} \longleftarrow$ channel ($\mathbf{x}; \sigma^2$)
    $\hat{\mathbf{u}} \longleftarrow$ receive ($\mathbf{y}; \theta_{\text{rx}}$)
    $L_{\text{BCE}} \longleftarrow$ BCE ($\mathbf{u}, \hat{\mathbf{u}}$)
    $\theta_{\text{tx}}, \theta_{\text{rx}} \longleftarrow$ SGD ($[\theta_{\text{tx}}, \theta_{\text{rx}}]$,$L_{\text{BCE}}$)
  end for
end for

Algorithm 2 Alternating Training Procedure for CNN-AE

Input: Number of Epoch M, Training Step T_{TX},T_{RX}, Training SNR $\sigma^2_{\text{min}}, \sigma^2_{\text{max}}$, Training parameter θ_{tx}, θ_{rx}
Output: $\theta_{\text{tx}}, \theta_{\text{rx}}$
for $i \leq M$ **do**
  for $j \leq T_{\text{TX}}$ **do**
    set_trainable($\theta_{\text{tx}}, \theta_{\text{rx}}$) = [True, False]
    $\sigma^2 \longleftarrow$ generate_SNR ($[\sigma^2_{\text{min}}, \sigma^2_{\text{max}}]$)
    $\mathbf{u} \longleftarrow$ generate_bits ()
    $\mathbf{x} \longleftarrow$ transmit ($\mathbf{u}; \theta_{\text{tx}}$)
    $\mathbf{y} \longleftarrow$ channel ($\mathbf{x}; \sigma^2$)
    $\hat{\mathbf{u}} \longleftarrow$ receive ($\mathbf{y}$)
    $L_{\text{BCE}} \longleftarrow$ BCE ($\mathbf{u}, \hat{\mathbf{u}}$)
    $\theta_{\text{tx}} \longleftarrow$ SGD (θ_{tx}, L_{BCE})
  end for
  for $j \leq T_{\text{RX}}$ **do**
    set_trainable($\theta_{\text{tx}}, \theta_{\text{rx}}$) = [False, True]
    $\mathbf{x} \longleftarrow$ transmit ($\mathbf{u}; \theta_{\text{tx}}$)
    $\mathbf{y} \longleftarrow$ channel ($\mathbf{x}; \sigma^2$)
    $\hat{\mathbf{u}} \longleftarrow$ receive ($\mathbf{y}; \theta_{\text{rx}}$)
    $L_{\text{BCE}} \longleftarrow$ BCE ($\mathbf{u}, \hat{\mathbf{u}}; \theta_{\text{rx}}$)
    $\theta_{\text{tx}}, \theta_{\text{rx}} \longleftarrow$ SGD ($[\theta_{\text{tx}}, \theta_{\text{rx}}]$,$L_{\text{BCE}}$)
  end for
end for

Parameter	Value
Loss	BCE
Epoch	100
Batch size	500
Training vectors	10^6
Optimizer	Adam
Learning rate	0.001

Table 3.2: Hyperparameters for the training of CNN-AE under the AWGN channel.

for achieving the BLER performance necessary for our comparative analysis. It is worth mentioning that in [38], several alternative loss functions are proposed that aim for BLER-optimal decoding.

3.2 An AE-based Joint Synchronization, Equalization, and Decoding System

We now move forward to consider transmitting short packets over a memoryless block-fading waveform channel with an unknown delay. In this scenario, the receiver needs to handle the task of synchronization, equalization, and decoding. We propose a CNN-AE-based joint synchronization, equalization, and decoding system. Instead of relying on the dedicated pilots, the proposed system uses the holistic message from the channel outputs to synchronize and decode the transmitted signal. This approach not only results in higher spectral efficiency but also allows for the possibility of shorter messages compared to systems that use dedicated pilots. The system follows the same setup in [39], but discards the use of conventional pilots. The following subsections will provide a detailed introduction to the CNN-AE-based scheme and its training procedure.

Transmitter

The transmitter part, depicted in Figure 3.2, consists of the trainable CNN blocks and the non-trainable parts. At first, the Enc CNN encodes the information bits $\mathbf{b} \in \{0,1\}^k$ into a coded sequence $\mathbf{c} \in \mathbb{R}^{n_c}$, which is real-valued. Then the Mod CNN maps the coded sequence $\mathbf{c}$ to the complex-valued symbol sequence $x \in \mathbb{C}^n$. In order to transmit the short message through the block fading channel with unknown delay, we process the symbol sequence with following steps.

First, we split the symbol sequence $\mathbf{x} \in \mathbb{C}^n$ into n_b sub-sequences $\{\mathbf{x}_\ell\}_{\ell=1}^{n_b}$ of length n_s

$$\mathbf{x}_\ell = [x_{1,\ell}, ..., x_{n_s,\ell}] \in \mathbb{C}^{n_s},$$

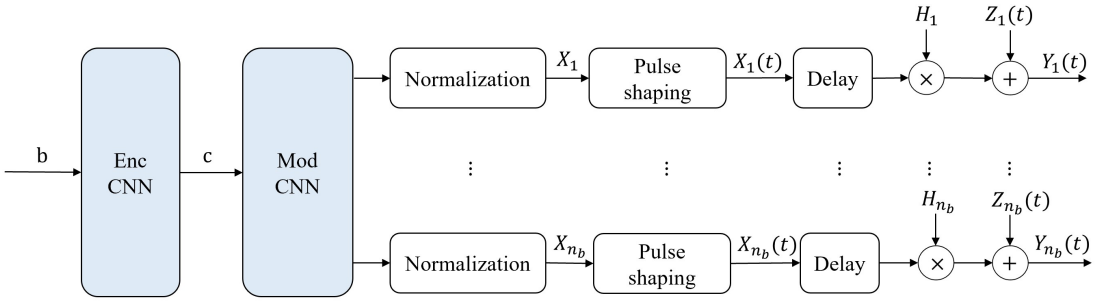

Figure 3.2: Block diagram of the transmitter part of the CNN-AE-based system model, the blue blocks indicates the trainable parts.

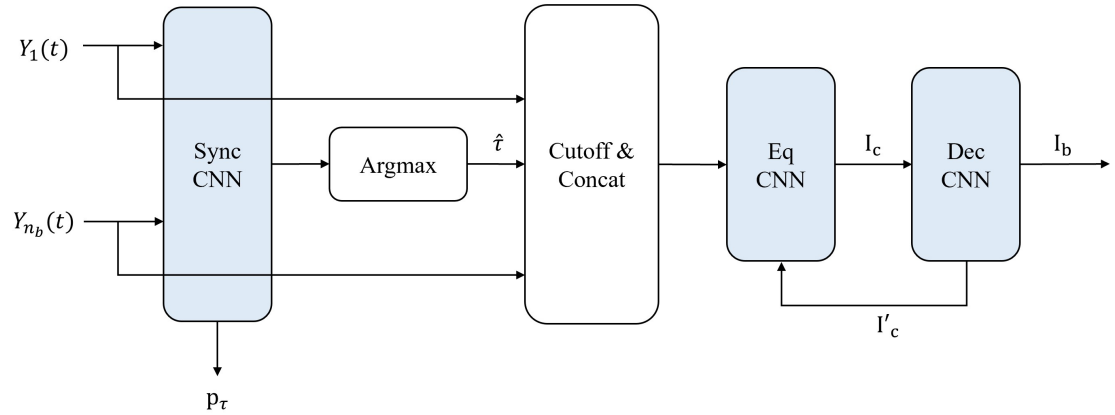

Figure 3.3: Block diagram of the receiver part of the CNN-AE-based system model.

where $\ell = 1, ..., n_b$, n_b denotes the number of fading blocks, each sub-fading blocks contains n_s complex-valued channel uses.

Then we apply power normalization for ℓth sub packet according to

$$\mathbb{E}\left[\|\mathbf{x}_\ell\|^2\right] = n_s\rho,$$

where ρ denoted the SNR.

In order to form the continuous-time signal, we add pulse shaping block. Consider using square pulse with normalized energy

$$s_{t_p}(t) = \begin{cases} \frac{1}{\sqrt{t_p}}, & t \in [0, t_p)\,, \\ 0, & \text{otherwise}\,, \end{cases}$$

where t_p denotes the period of the pulse, determined by the upsampling rate N and sampling interval t_s as $t_p = Nt_s$. The signal for ℓth sub packet can be expressed as:

$$x_\ell(t) = \sum_{k=1}^{n_s} x_{k,\ell} s_{t_p}\left(t - (k-1)\,t_p\right).$$

The continuous-time signal then can be transmitted over the fading channel with an unknown delay. The received signal for ℓth fading block is:

$$Y_\ell(t) = H_\ell x_\ell(t-\tau) + Z_\ell(t),$$

where H_ℓ denotes the random complex gain for the ℓth fading block, following the $\mathcal{CN}(0,1)$ distribution; τ denotes the time delay, which we consider a simple case that each sub packet experiences the same delay; τ can be treated as uniform distributed in $[0, \tau_{\max}]$; $Z_1(t), ..., Z_{n_b}(t)$ are independent additive white Gaussian noise with power spectral density N_0.

Receiver

The receiver structure is depicted in Figure 3.3. Unlike the previous CNN-AE scheme designed for the AWGN channel, the proposed structure includes a Sync CNN for synchronization, as well as a Demod CNN and Dec CNN for equalization and decoding. We next introduce each block in detail.

First, the Sync CNN estimates the starting position of the data signal from the received signal $Y_1(t), ..., Y_{n_b}(t)$. The length of the received signal for each sub-packet is $n' = n_s t_p + \tau_{\max}$. Since the received signal is complex-valued, it is divided into 2 real-valued signals, corresponding to the real and imaginary parts. The Sync CNN maps the received signal of each sub-packet into a higher dimensional space and outputs the probability distribution of $\tau_{\max}$ possible outcomes using the softmax activation function

$$\text{softmax}(\boldsymbol{\tau})_i = \frac{e^{\tau_i}}{\sum_{j=1}^{\tau_{\max}} e^{\tau_j}}.$$

Here, $\mathbf{p}_\tau$ is a $\tau_{\max}$-dimensional probability vector with all entries between 0 and 1. $p_i = \text{softmax}(\boldsymbol{\tau})_i$ represents the probability of the estimated delay $\hat{\tau} = i$. The sum of the probabilities $\sum_{i=1}^{\tau_{\max}} p_i = 1$.

Then the time delay is estimated according to $\hat{\tau} = \arg\max \mathbf{p}_\tau$. We remove the estimated delay from each sub-packets and concatenate all sub-packets together, resulting in a signal $\mathbf{y}(t)'$ of length nt_p, which is then used for equalization and decoding.

The following CNN blocks function in an iterative fashion, as shown in Figure 3.4. First, the EQ CNN performs equalization with inherent channel gain estimation. The output of the EQ CNN, denoted as $\mathbf{I_c}$, can be interpreted as the prior information provided to the Dec CNN. Subsequently, the Dec CNN calculates the posterior of the transmitted bit sequence $\mathbf{I_b}$, and sends the extrinsic information $\mathbf{I_c}' = \mathbf{I_b} - \mathbf{I_c}$ back to the EQ CNN that can be used as prior for equalization at next iteration. After a sufficient number of iterations, the estimated bits can be calculated from $\mathbf{I_b}$ using the sigmoid function: sigmoid($\mathbf{I_b}$).

The size of both the prior information and extrinsic information is (n, F), where F represents the information feature size. The number F indicates the amount of in-

Layer	Activation	Output dimensions
	Enc CNN	
Conv1D	ELU	$(k,200)$
Conv1D	ELU	$(k,200)$
Conv1D	ELU	$(k,200)$
Conv1D	ELU	$(k,200)$
Conv1D	ELU	$(k,\lfloor n/k \rfloor)$
Reshape		$(n,1)$
	Mod CNN	
Conv1D	ELU	$(n,200)$
Conv1D	ELU	$(n,200)$
Conv1D	ELU	$(n,200)$
Conv1D	ELU	$(n,200)$
Conv1D	Linear	$(n,2)$
	Sync CNN	
Conv1D	ELU	$(n_b,n',100)$
Conv1D	ELU	$(n_b,n',100)$
Conv1D	ELU	$(n_b,n',100)$
Conv1D	ELU	$(n_b,n',100)$
Conv1D	ELU	$(n_b,n',1)$
Flatten		$(n_b n',)$
Dense	Softmax	$(\tau_{\max})$
	EQ CNN	
Conv1D	ELU	$(n,200)$
Conv1D	ELU	$(n,200)$
Conv1D	ELU	$(n,200)$
Conv1D	ELU	$(n,200)$
Conv1D		(n,F)
	Dec CNN	
Conv1D	ELU	$(n,200)$
Conv1D	ELU	$(n,200)$
Conv1D	ELU	$(n,200)$
Conv1D	ELU	$(n,200)$
Conv1D		(n,F)
	Dec CNN Last Iteration	
Flatten		$(nF,)$
Dense	Sigmoid	$(k,1)$

Table 3.3: Parameters of the CNN-AE-based joint synchronization, equalization, and decoding system.

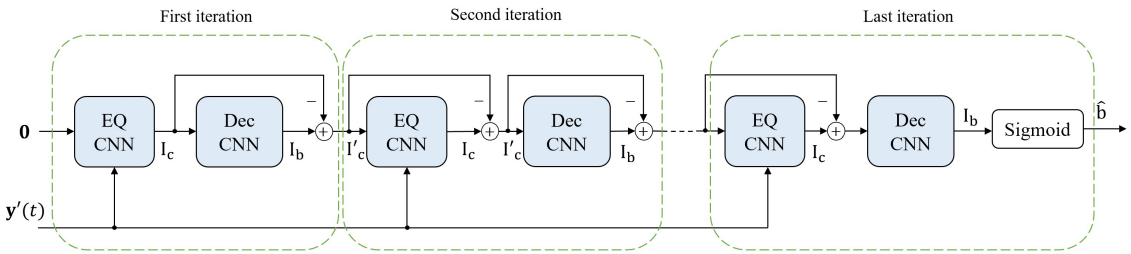

Figure 3.4: The iteration steps of the equalization and decoding at the receiver.

formation exchanged between the EQ CNN and Dec CNN per codeword. Compared to sequential structure, the iterative process results in faster convergence.

Table 3.3 shows the structure of each layer in the system. Compared to previous configurations, more filters are applied to each Conv1D layer to allow the system to better capture and process the intricate patterns of the signal.

Training procedure and parameters

We train the CNN-AE by optimizing the BCE loss between the transmitted bit sequence $\mathbf{b}$ and the estimated bit sequence $\hat{\mathbf{b}}$ at the receiver's output. In parallel, the Sync CNN outputs a prediction $\mathbf{p}_\tau$ for the time delay. The CCE is calculated using the prediction $\mathbf{p}_\tau$ and the one-hot representation of the true delay τ. Thus, we define the total loss as a sum of both synchronization loss and decoding loss,

$$L = L_{\text{BCE}} + \alpha \cdot L_{\text{CCE}}.$$

The hyperparameter α can be adjusted to balance the decoding and synchronization performance of the system.

The main hyperparameters are given in Table 3.4 and the training process is shown in Algorithm 3.

Algorithm 3 Training Procedure for CNN-AE

Input: Number of Epoch M, Training Step T, Training SNR $\sigma^2_{\text{min}}, \sigma^2_{\text{max}}$, Training Parameter θ_{enc}, θ_{dec}, θ_{sync}, Loss Weight α
Output: θ_{enc}, θ_{dec}, θ_{sync}

```
for i ≤ M do
  for j ≤ T do
    τ ⟵ generate_delay ([0, τ_max))
    σ² ⟵ generate_SNR ([σ²_min, σ²_max])
    b ⟵ generate_bits ()
    x ⟵ transmit (b; θ_enc)
    for x_i ∈ x do
      x_i ⟵ normalization (x_i)
      x_i(t) ⟵ pulse_shaping (x_i)
      y_i(t) ⟵ channel (x_i(t), τ, σ²)
    end for
    for y_i ∈ y do
      p_τ ⟵ synchronization (y_i, θ_sync)
    end for
    L_CCE ⟵ CCE (p_τ, τ; θ_sync)
    τ̂ ⟵ argmax (p_τ)
    for y_i ∈ y do
      y_cutoff ⟵ cutoff (y_i, τ̂)
    end for
    y_DEC ⟵ concat (y_cutoff)
    b̂ ⟵ decode (y_DEC; θ_dec)
    L_BCE ⟵ BCE (b, b̂; θ_dec)
    L ⟵ L_BCE + α · L_CCE
    θ_enc, θ_sync, θ_dec ⟵ SGD ([θ_enc, θ_sync, θ_dec], L)
  end for
end for
```

Parameter	Value
Loss	BCE,CCE
α	0.01
F	20
Decoder iteration	6
Batch size	500 - 1000
Optimizer	Adam
Learning rate	10^{-4} - 10^{-5}
Training SNR	2.0 - 20.0 dB

Table 3.4: Hyperparameters for the training of CNN-AE-based system.

4
Results

In this chapter, we discuss the performance of our AE-based communication systems described in the previous chapter. First, we consider the CNN-AE system under a AWGN channel and a block-fading channel and compare its performance with state-of-the-art channel codes in terms of BER and BLER. Next, we consider the proposed CNN-AE-based joint synchronization, equalization, and decoding system, and compare the synchronization and decoding performance with the achievability bound for pilot-assisted system. The following sections provide a detailed illustration.

4.1 Performance of CNN-AE

First, we evaluate the end-to-end performance of proposed CNN-AE system as described in Chapter 3.1. We consider transmitting $k = 64$ information bits within a short packet of $n = 128$ channel uses over a memoryless AWGN channel and a Rayleigh block-fading channel. The communication rate is $R = k/n = 1/2$ bits per channel use.

4.1.1 Performance of CNN-AE under AWGN Channel

Figure 4.1 and 4.2 presents the simulated BER and BLER performance of following schemes under the AWGN channel:

- Baseline system : The system employs 5G compliant LDPC codes combined with BPSK modulation, simulated using the Sionna library [40]. An LDPC code with a code rate $R_c = 1/2$ is utilized. Specifically, the code uses base graph 2 with a lifting factor of 11. At the receiver, a boxplus-phi belief propagation decoder is applied with 20 iterations for decoding.

- Proposed CNN-AE : The system adheres to the structure detailed in Table 3.1 and is trained and tested over the same range of SNRs for 10^6 blocks.

We also plot the normal approximation on BLER as a function of the SNRs for $R = 1/2$ and $n = 128$ under real-valued AWGN channels. The simulation is done with the help of SPECTRE toolbox [41].

The results demonstrate that the proposed CNN-AE scheme exhibits comparable performance to the baseline system under the AWGN channel. At low SNRs, the

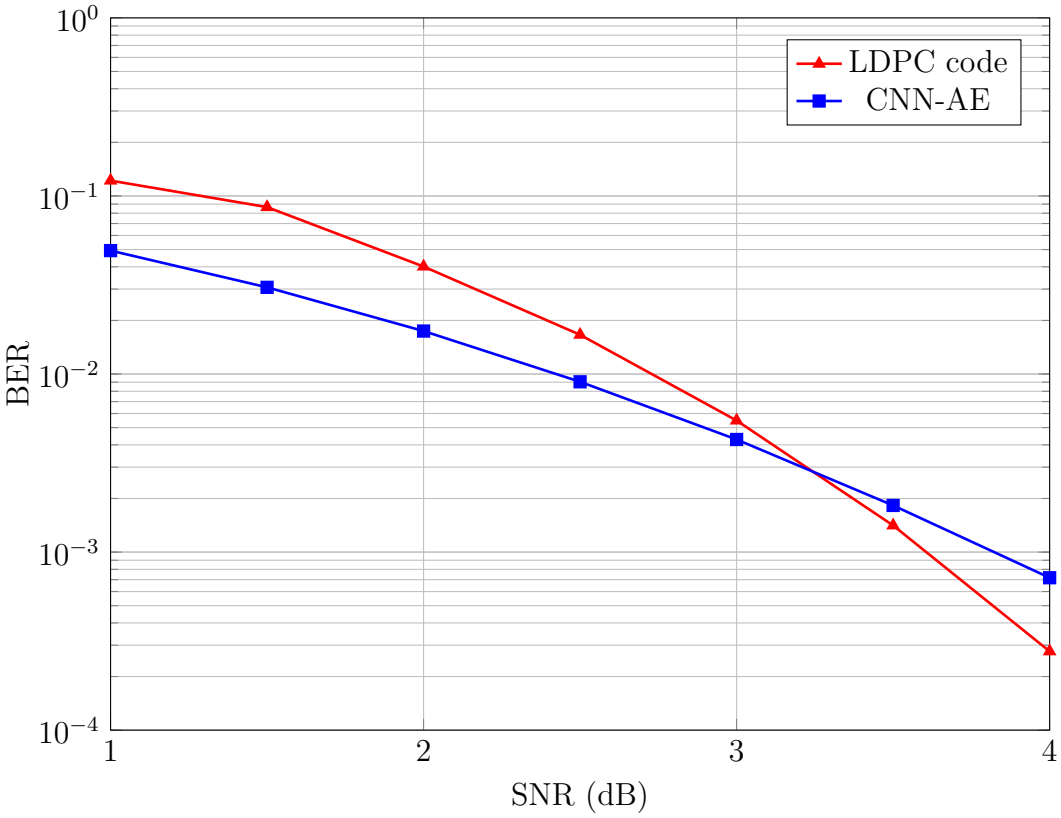

Figure 4.1: Simulated BER under AWGN channel with $k = 64, n = 128$.

CNN-AE scheme outperforms the baseline system, demonstrating superior performance in terms of both BER and BLER. Notably, the CNN-AE achieves a more significant reduction in BER than BLER when compared to the baseline system. However, at higher SNRs, the baseline system surpasses the CNN-AE, resulting in a more rapid decline in both BER and BLER. Both systems show a performance gap relative to the normal approximation for BLER. For the following performance comparison, the focus will be on BLER comparisons instead.

4.1.2 Performance of CNN-AE under Block-fading Channel

We consider the transmission over a Rayleigh memoryless block-fading channel, where neither the transmitter nor the receiver has any prior knowledge of the CSI. The number of fading blocks is $n_b = 4$, and the channel gains of each fading block $H_\ell, \ell = 1, ..., n_b$ are independently distributed according to a complex Gaussian distribution $\mathcal{CN}(0, 1)$.

Figure 4.3 illustrates the simulated BLER performance of following schemes under the block-fading channel:

- Achievability bound : This refers to the nonasymptotic achievability bound on maximum coding rate over Rayleigh block-fading channels, under the assumption that the receiver lacks prior knowledge of the CSI. This bound is

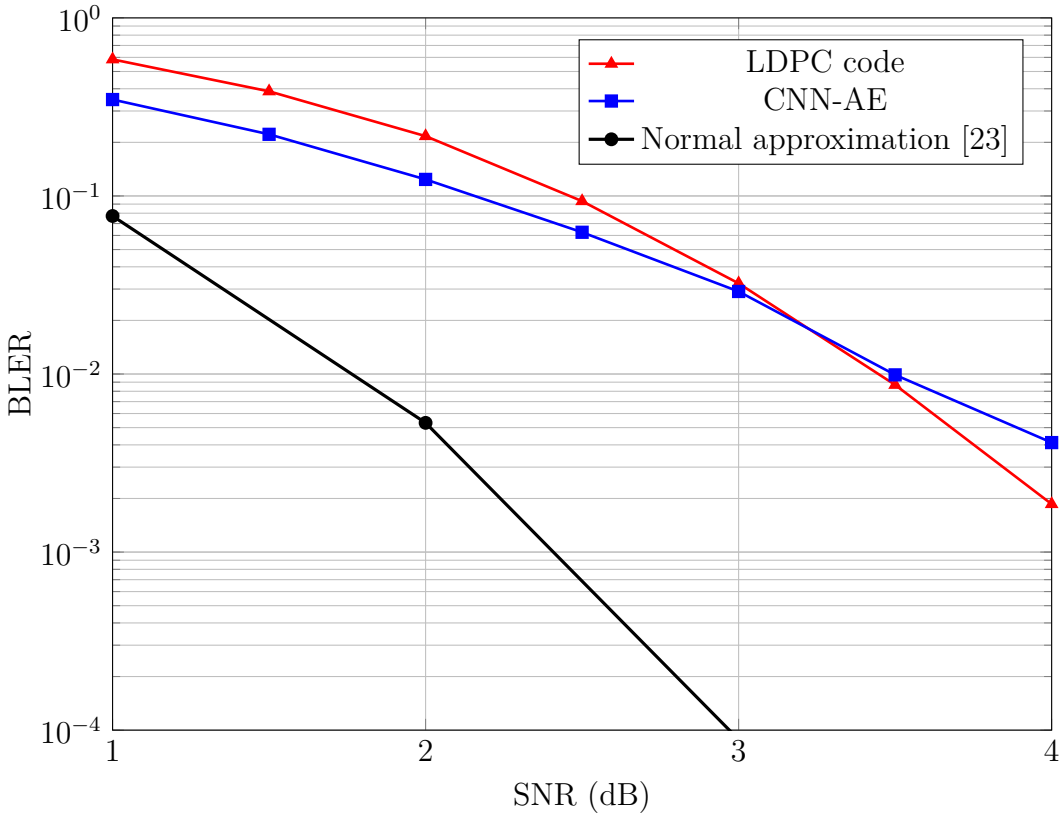

Figure 4.2: Simulated BLER under AWGN channel with $k = 64, n = 128$.

proposed in [28] and is simulated using the SPECTRE toolbox [42].

- Proposed CNN-AE : The system adheres to the structure detailed in Table 3.1 and is trained and tested over the same range of SNRs for 10^6 blocks.

- Baseline system : The system employs 5G compliant LDPC codes with QPSK modulation at the transmitter and ZF equalizer at the receiver, where the CSI is assumed to be known.

The achievability bound serves as a relatively tight benchmark on the error probability for any transmission scheme under a memoryless Rayleigh fading channel, particularly when no CSI is available at the receiver. Compared to the baseline system, CNN-AE exhibits better performance at higher SNRs, particularly from 11 dB onwards. It is important to note that the baseline system benefits from prior knowledge of the CSI; hence, no pilot sequence is required, and no rate loss is incurred. In contrast, the CNN-AE not only maintains performance but also enhances spectral efficiency. Nonetheless, a performance gap remains when compared to the achievability bound. In the following section, we will evaluate the performance of the enhanced CNN-AE structure, as detailed in Chapter 3.2, in a more complicated scenario.

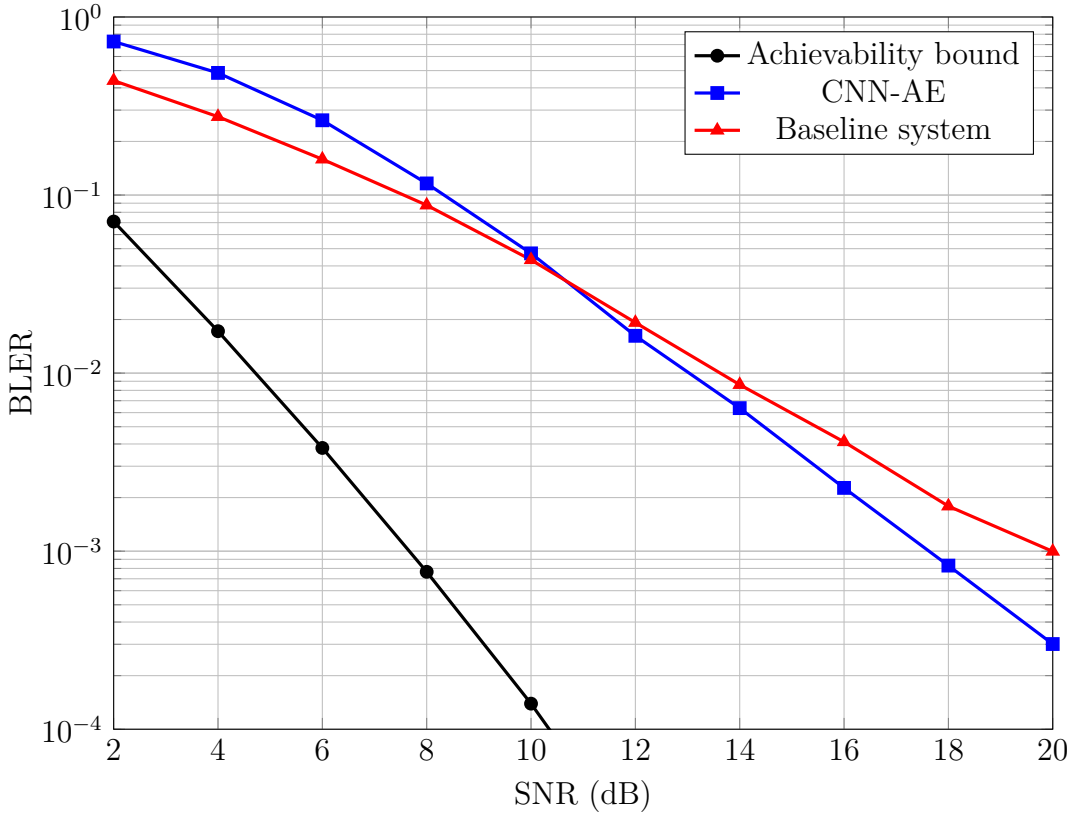

Figure 4.3: Simulated BLER under the block-fading channel with $n_b = 4, k = 64, n = 128$.

Parameter	Value
information bits: k	80
blocklength: n	144
number of fading blocks: n_b	4
upsampling rate: N	5
maximum time delay : $\tau_{\max}$	12

Table 4.1: Parameters for simulation.

4.2 Performance of CNN-AE-based Joint Synchronization, Equalization, and Decoding System

In this section, we consider the short packet transmission over a SISO memoryless block-fading waveform channel with an unknown delay. The benchmark is proposed in [39]. The proposed CNN-AE-based joint synchronization, equalization, and decoding scheme is detailed in Chapter 3.2. We evaluate the performance of the proposed scheme in terms of normalized mean square error (NMSE) for delay estimation error and BLER for decoding.

The selected parameters for the simulation are shown in Table 4.1. The transmission rate is $R = 80/144 = 0.556$ bit per complex channel use. We assume the channel gains $H_\ell, \ell = 1, ..., n_b$ are generated independently from $\mathcal{CN}(0, 1)$. The time delay τ for each sub-block is assumed to be uniformly distributed in $[0, \tau_{\max}]$.

4.2.1 Synchronization Performance

To evaluate synchronization error, we consider the metric NMSE for delay estimation, which is defined as:

$$\text{NMSE} = \mathbb{E}\left[(\tau - \hat{\tau})^2 / t_{\text{p}}^2\right].$$

Here, t_{p} is the period of the pulses used for pulse shaping. With the assumption that the sampling interval is 1 second, t_{p} coincides with the upsampling rate N.

The benchmark is a pilot-assisted system detailed in [39]. Both the benchmark and proposed system aim to handle synchronization tasks at the receiver. While the benchmark applies ML estimation for joint synchronization and channel estimation using pilot assistance, the proposed CNN-AE-based system utilizes the entire received signal for synchronization.

Figure 4.4 presents the simulated results for the benchmark and CNN-AE-based system. It is evident that the CNN-AE-based system outperforms the ML estimation used in the pilot-assisted scheme. This superior performance is attributed to the CNN-AE's ability to perform delay estimation over the entire observation of the

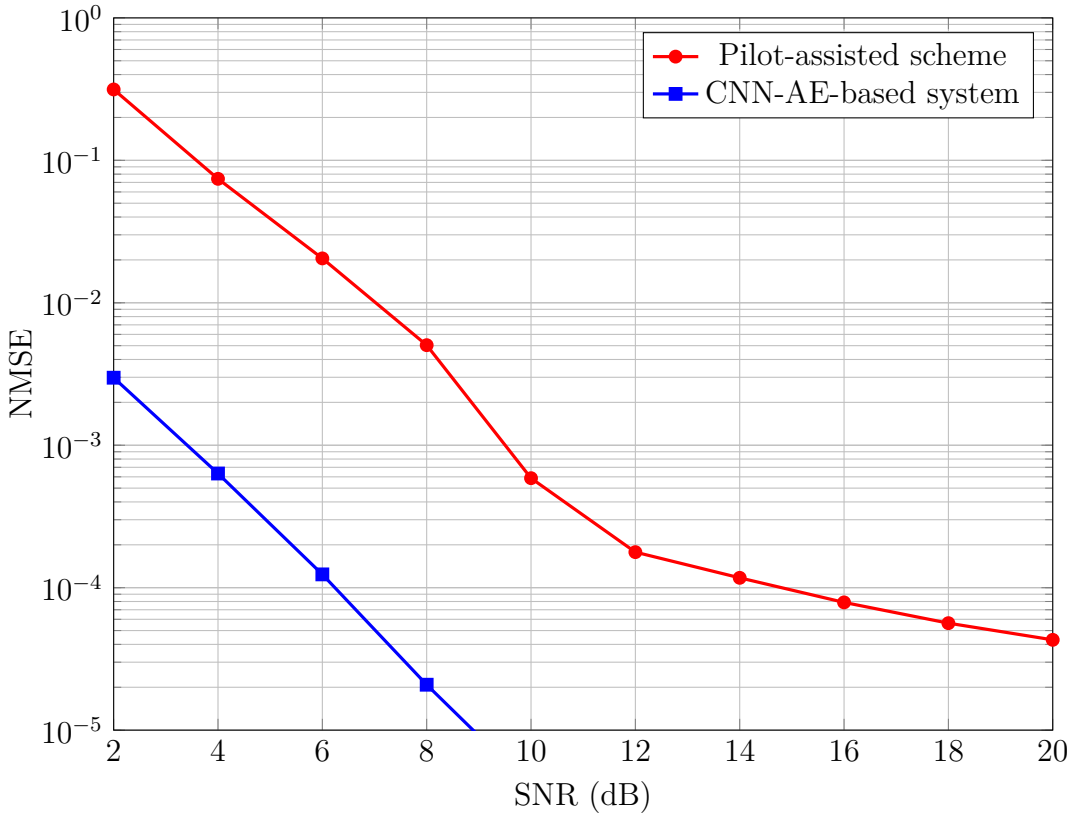

Figure 4.4: Synchronization error comparison.

received signal, as opposed to the pilot-assisted scheme, which relies solely on the pilot sequence. The CNN-AE-based system can learn latent information from the received signal without requiring prior information, enhancing its synchronization performance.

4.2.2 Decoding Performance

After synchronizing the received signal, the receiver needs to equalize the signal and estimate the transmitted bit sequence. The benchmark proposed in [39] developed an RCUs bound on error probability for pilot-assisted transmission systems. The numerical results of the achievability bound are efficiently computed using the saddlepoint approximation.

We evaluate the BLER performance for the benchmark and CNN-AE-based system as shown in Figure 4.4. The simulated results indicate that while the CNN-AE-based system underperforms compared to the benchmark at SNRs lower than 10 dB, it surpasses the benchmark at higher SNRs, which are more practical operational ranges. Specifically, the CNN-AE-based system outperforms the benchmark by 1.7 dB when the BLER is 10^{-3}. Our proposed scheme jointly learns the channel gains and estimates the transmitted bit sequences without relying on any known sequence assistance. At lower SNRs, synchronization performance significantly influences decoding performance, requiring the training process of the CNN-AE to carefully

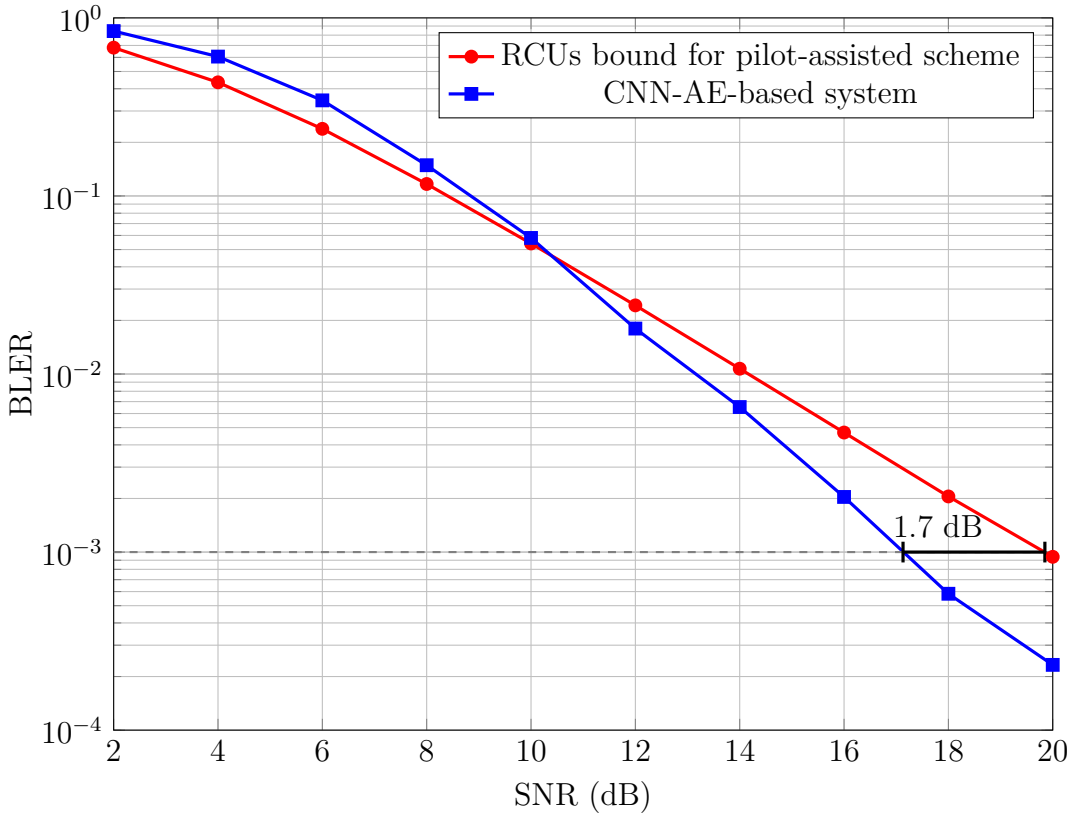

Figure 4.5: Achievable BLER comparison.

balance both tasks. However, at higher SNRs, synchronization performance has minimal impact on decoding, making it easier to improve decoding performance.

5
Conclusion

This chapter summarizes the work conducted in this thesis and outlines some potential ideas for future work.

In this thesis, we applied end-to-end learning of physical-layer communications and evaluated the performance of CNN-AE-based joint synchronization, equalization, and decoding system under short packet communications. Unlike conventional communication systems that use dedicated preambles for synchronization and equalization, our proposed system performs joint synchronization, equalization, and decoding without the use of dedicated preambles and prior information. This approach is more suitable in URLLC scenarios as it greatly improves the spectral efficiency and reduces the overhead in short packet regime. Compared to the nonasymptotic achievability bound for pilot-assisted transmission systems, our proposed system results in a lower BLER performance in the high SNR range under block-fading waveform channels.

5.1 Future Work

While this thesis has explored the use of CNN-AE in short packet communications, several important aspects and potential improvements merit further research:

- Exploring CNN-AE and Turbo-AE structures: While the proposed system is based on CNN-AE, another structure called Turbo-AE also demonstrates good performance under AWGN and fading channels [13], [43]. Turbo-AE takes advantage of Turbo codes, utilizing interleavers and deinterleavers at both the transmitter and receiver. Further exploration of the Turbo-AE structure is warranted. Additionally, combining the strengths of both CNN-AE and Turbo-AE could potentially yield a hybrid model with superior performance.

- Optimizing the training procedure: The computational complexity of the training procedure for the proposed system is high. The selection of hyperparameters is currently based on empirical fine-tuning, which can be cumbersome and may not generalize well to other scenarios. Future research should focus on developing more efficient training algorithms and hyperparameter optimization techniques. Methods such as reinforcement learning could be explored to streamline the training process and improve the system's adaptability to different conditions.

- Extending to more practical system setup: Our proposed system considers the case where all fading blocks are synchronous, experiencing the same time delay. A more complex scenario involves different fading blocks experiencing different random time delays, which has also been evaluated in our benchmark [39]. In such cases, the synchronization component of our proposed system should be updated.

By addressing these aspects, we can further improve the performance and applicability of CNN-AE-based joint synchronization, equalization, and decoding system in short packet communications.

Bibliography

[1] Hamidreza Bagheri, Md Noor-A-Rahim, Zilong Liu, Haeyoung Lee, Dirk Pesch, Klaus Moessner, and Pei Xiao. 5G NR-V2X: Toward Connected and Cooperative Autonomous Driving. *IEEE Communications Standards Magazine*, 5(1):48–54, 3 2021.

[2] Georgia Kolovou, Sharief Oteafy, and Periklis Chatzimisios. A Remote Surgery Use Case for the IEEE P1918.1 Tactile Internet Standard. *IEEE International Conference on Communications*, 6 2021.

[3] Ali Gohar, Gianfranco Nencioni, Omar Khyam, and Xuejun Li. The Role of 5G Technologies in a Smart City: The Case for Intelligent Transportation System. *Sustainability 2021, Vol. 13, Page 5188*, 13(9):5188, 5 2021.

[4] Rashid Ali, Yousaf Bin Zikria, Ali Kashif Bashir, Sahil Garg, and Hyung Seok Kim. URLLC for 5G and Beyond: Requirements, Enabling Incumbent Technologies and Network Intelligence. *IEEE Access*, 9:67064–67095, 2021.

[5] Harsh Tataria, Mansoor Shafi, Andreas F. Molisch, Mischa Dohler, Henrik Sjoland, and Fredrik Tufvesson. 6G Wireless Systems: Vision, Requirements, Challenges, Insights, and Opportunities. *Proceedings of the IEEE*, 109(7):1166–1199, 7 2021.

[6] Zexian Li, Hamidreza Shariatmadari, Bikramjit Singh, and Mikko A. Uusitalo. 5G URLLC: Design challenges and system concepts. *Proceedings of the International Symposium on Wireless Communication Systems*, 2018-August, 10 2018.

[7] Alexandru Sabin Bana, Kasper Floe Trillingsgaard, Petar Popovski, and Elisabeth De Carvalho. Short Packet Structure for Ultra-Reliable Machine-Type Communication: Tradeoff between Detection and Decoding. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2018-April:6608–6612, 9 2018.

[8] Alejandro Lancho, Johan Ostman, and Giuseppe Durisi. On Joint Detection and Decoding in Short-Packet Communications. *Proceedings - IEEE Global Communications Conference, GLOBECOM*, 2021.

[9] Timothy O'Shea and Jakob Hoydis. An Introduction to Deep Learning for the Physical Layer. *IEEE Transactions on Cognitive Communications and Networking*, 3(4):563–575, 12 2017.

[10] Sebastian Dorner, Sebastian Cammerer, Jakob Hoydis, and Stephan Ten Brink. Deep Learning Based Communication over the Air. *IEEE Journal on Selected Topics in Signal Processing*, 12(1):132–143, 2 2018.

[11] Faycal Ait Aoudia and Jakob Hoydis. End-to-End Learning of Communications Systems Without a Channel Model. *Conference Record - Asilomar Conference on Signals, Systems and Computers*, 2018-October:298–303, 7 2018.

[12] Alexander Felix, Sebastian Cammerer, Sebastian Dorner, Jakob Hoydis, and Stephan Ten Brink. OFDM-Autoencoder for End-to-End Learning of Communications Systems. *IEEE Workshop on Signal Processing Advances in Wireless Communications, SPAWC*, 2018-June, 8 2018.

[13] Yihan Jiang, Hyeji Kim, Himanshu Asnani, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath. Turbo Autoencoder: Deep learning based channel codes for point-to-point communication channels. *Advances in Neural Information Processing Systems*, 32, 2019.

[14] Nourhan Hesham, Mohamed Bouzid, Ahmad Abdel-Qader, and Anas Chaaban. Coding for the Gaussian Channel in the Finite Blocklength Regime Using a CNN-Autoencoder. *2023 IEEE International Black Sea Conference on Communications and Networking, BlackSeaCom 2023*, pages 15–20, 2023.

[15] Ognjen Jovanovic, Metodi P. Yankov, Francesco Da Ros, and Darko Zibar. Gradient-Free Training of Autoencoders for Non-Differentiable Communication Channels. *Journal of Lightwave Technology*, 39(20):6381–6391, 10 2021.

[16] R. G. Gallager. Low-Density Parity-Check Codes. *IRE Transactions on Information Theory*, 8(1):21–28, 1962.

[17] Claude Berrou, Alain Glavieux, and Punya Thitimajshima. Near SHANNON limit error-correcting coding and encoding: Turbo-codes (1). *IEEE International Conference on Communications*, pages 1064–1070, 1993.

[18] Solomon W. Golomb and Guang Gong. Signal design for good correlation: For wireless communication, cryptography, and radar. *Signal Design for Good Correlation: For Wireless Communication, Cryptography, and Radar*, 9780521821049:1–440, 1 2005.

[19] David C. Chu. Polyphase Codes with Good Periodic Correlation Properties. *IEEE Transactions on Information Theory*, 18(4):531–532, 1972.

[20] Huseyin Arslan. Wireless communication signals : a laboratory-based approach.

[21] John G. Proakis. Digital communications. 2001.

[22] C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

[23] Yury Polyanskiy, H. Vincent Poor, and Sergio Verdú. Channel coding rate in the finite blocklength regime. *IEEE Transactions on Information Theory*, 56(5):2307–2359, 5 2010.

[24] Alfonso Martinez and Albert Guillén I Fàbregas. Saddlepoint approximation of random-coding bounds. *2011 Information Theory and Applications Workshop, ITA 2011 - Conference Proceedings*, pages 257–262, 2011.

[25] Mustafa Cemil Coşkun, Giuseppe Durisi, Thomas Jerkovits, Gianluigi Liva, William Ryan, Brian Stein, and Fabian Steiner. Efficient error-correcting codes in the short blocklength regime. *Physical Communication*, 34:66–79, 6 2019.

[26] Raymond Knopp and Pierre A. Humblet. On coding for block fading channels. *IEEE Transactions on Information Theory*, 46(1):189–205, 1 2000.

[27] Wei Yang, Giuseppe Durisi, Tobias Koch, and Yury Polyanskiy. Quasi-static multiple-antenna fading channels at finite blocklength. *IEEE Transactions on Information Theory*, 60(7):4232–4265, 2014.

[28] Giuseppe Durisi, Tobias Koch, Johan Östman, Yury Polyanskiy, and Wei Yang. Short-Packet Communications over Multiple-Antenna Rayleigh-Fading Channels. *IEEE Transactions on Communications*, 64(2):618–629, 12 2014.

[29] Siddharth Sharma, Simone Sharma, and Anidhya Athaiya. ACTIVATION FUNCTIONS IN NEURAL NETWORKS. *International Journal of Engineering Applied Sciences and Technology*, 4:310–316, 2020.

[30] Y Le Cun and Yann Le Cun. Generalization and Network Design Strategies. 1989.

[31] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. *Proceedings of the 25th International Conference on Machine Learning*, pages 1096–1103, 2008.

[32] Deep Learning - Ian Goodfellow, Yoshua Bengio, Aaron Courville - Google Books.

[33] Paul J. Werbos. Backpropagation Through Time: What It Does and How to Do It. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[34] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature 1986 323:6088*, 323(6088):533–536, 1986.

[35] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning, 5 2013.

[36] Diederik P. Kingma and Jimmy Lei Ba. Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 12 2014.

[37] Djork Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 11 2015.

[38] Reinhard Wiesmayr, Gian Marti, Chris Dick, Haochuan Song, and Christoph Studer. Bit Error and Block Error Rate Training for ML-Assisted Communication. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2023-June, 2023.

[39] A Oguz Kislal, Madhavi Rajiv, Giuseppe Durisi, Senior Member, Erik G Ström, and Urbashi Mitra. Is Synchronization a Bottleneck for Pilot-Assisted URLLC Links? 1 2024.

[40] Jakob Hoydis, Sebastian Cammerer, Fayçal A¨ıt, A¨ıt Aoudia, Avinash Vem, Nikolaus Binder, Guillermo Marcus, and Alexander Keller. Sionna: An Open-Source Library for Next-Generation Physical Layer Research. 3 2022.

[41] gdurisi/fbl-notes: Transmitting short-packet over wireless channels—an information-theoretic perspective.

[42] yp-mit/spectre: SPECTRE: Short packet communication toolbox.

[43] Jannis Clausius, Sebastian Dorner, Sebastian Cammerer, and Stephan Ten Brink. Serial vs. Parallel Turbo-Autoencoders and Accelerated Training for

Learned Channel Codes. *2021 11th International Symposium on Topics in Coding, ISTC 2021*, 2021.

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY