BLIND CODE IDENTIFICATION USING DEEP NEURAL NETWORKS

by

Cihat Keçeci

B.S., Electrical and Electronics Engineering, Boğaziçi University, 2017

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical&Electronics Engineering
Boğaziçi University

2019

ACKNOWLEDGEMENTS

I would like to thank my supervisor Prof. Hakan Deliç for his insightful guidance and patience throughout my graduate studies. His guidance helped me in all the time of research and writing of this thesis.

I am also grateful to Dr. Ali Rıza Ekti and Dr. Ali Görçin for their guidance and help not only in my research but also in my life in general.

I would like to thank my family, especially my brother Fatih, who have always provided me through moral and emotional support in my life. I am also grateful to my other family members and friends who have supported me along the way.

ABSTRACT

BLIND CODE IDENTIFICATION USING DEEP NEURAL NETWORKS

The non-hierarchical communication paradigm shift in the next generation wireless networks reveals the requirement for the adaptive configuration of communication parameters in order to provide flexibility across different scenarios and changing channel conditions. Hence, automatic identification of communication parameters will play a significant role in 5G and beyond networks. In automatic modulation and coding (AMC) systems, the transmission parameters are changed dynamically in order to adapt to the changing channel conditions. These parameters are signalled through control channels which results in an increase of the resource usage. Blind decoding is fast becoming a key instrument in such systems since it eliminates the need for signaling of parameters. Blind decoding systems are composed of two stages: code identification and decoding with the identified code parameters. There is a considerable amount of research on the application of deep learning (DL) to channel decoding, and it has been shown that DL models are able to learn the code structure. In this thesis, we consider the use of DL for blind code identification, which is a first in this domain to the best of our knowledge. It is possible to design a universal code identification system via deep neural networks due to their independence of the channel code type. The presented approach is applied in detection of widely used convolutional, turbo and polar codes in order to show its capabilities. For each code type, the DL classification models provide not only higher detection rates, but also lower and predictable delay amounts compared to the existing methods. Additionally, an analysis on the required number of codewords for model training is provided.

ÖZET

DERİN YAPAY SİNİR AĞLARI KULLANILARAK KÖR KOD TANIMA

Yeni nesil kablosuz ağlardaki hiyerarşik olmayan iletişim paradigması kayması, farklı senaryolar ve değişen kanal koşulları için iletişim parametrelerinin uyarlanabilir yapılandırılması gerekliliğini ortaya koymaktadır. Bu nedenle haberleşme parametrelerinin otomatik belirlenmesi 5G ve ötesi ağlarda önemli bir rol oynayacaktır. Otomatik modülasyon ve kodlama sistemlerinde, değişen kanal şartlarına uyum sağlayabilmek için gönderici parametreleri dinamik olarak değiştirilmektedir. Bu parametrelerin iletimi kontrol kanalları aracılığıyla yapılmaktadır ve kaynak kullanımında artışa neden olmaktadır. Kör kod çözümü hızlı bir şekilde bu tarz sistemler için kilit bir öğe haline gelmektedir çünkü parametrelerin gönderimine olan ihtiyacı ortadan kaldırmaktadır. Kör kod çözme sistemleri iki aşamadan oluşmaktadır: kodun belirlenmesi ve belirlenen kod parametrelerine göre kod çözümü. Derin öğrenme yöntemlerinin kod çözümünde kullanımı üzerine hatrı sayılır miktarda çalışma bulunmaktadır ve bu çalışmalar derin öğrenme modellerinin kod yapısını öğrenebildiğini göstermiştir. Bu tez çalışmasında, bu konuda yeni bir yaklaşım olarak, kör kod tanımada derin öğrenme tabanlı bir yöntem önerilmektedir. Derin yapay sinir ağları herhangi bir kod parametresine bağımlı olmadığından dolayı evrensel bir kod tanıma sistemi tasarlanabilir. Sunulan yöntem, yetkinliğinin gösterimi için çokça kullanılan katlamalı, turbo ve kutupsal kodların seziminde kullanılmıştır. Her kod tipi için, derin öğrenme tabanlı yöntemler hem yüksek hata performansı, hem düşük ve öngörülebilir gecikme değerleri sağlamaktadır. Ek olarak, model eğitimi için kullanılması gereken kod sözcüğü üzerine bir analiz de sunulmaktadır. Elde edilen sonuçlar derin öğrenme modellerinin bütün kod sözcükleri ile eğitilmediği durumda bile genelleştirebildiğini göstermektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\mathbb{F}_2$	The Galois field of two elements
$\mathbf{G}$	Generator matrix
$g(D)$	Generator polynomial g in $\mathbb{F}_2$
$\mathbf{H}$	Parity check matrix
k	Information length
n	Codeword length
r	Code rate
ν	Constraint length
σ_N^2	Noise variance
$\otimes$	Kronecker product
$\{\cdot\}^T$	Transposition

LIST OF ACRONYMS/ABBREVIATIONS

3GPP	3rd Generation Partnership Project
AMC	Adaptive Modulation and Coding
AWGN	Additive White Gaussian Noise
BCJR	Bahl, Cocke, Jelinek and Raviv
BER	Bit Error Rate
BLER	Block Error Rate
BP	Belief Propagation
BPSK	Binary Phase Shift Keying
CFAR	Constant False Alarm Rate
CNN	Convolutional Neural Network
CRC	Cyclic Redundancy Check
D2D	Device-to-Device
DDG	Data Dependency Graph
DL	Deep Learning
DNN	Deep Neural Network
EM	Expectation Maximization
FEC	Forward Error Correction
GJETP	Gauss-Jordan Elimination Through Pivoting
GLRT	Generalized Likelihood Ratio Test
HDPC	High Density Parity Check
LD	Likelihood Difference
LDPC	Low Density Parity Check
LLR	Log-Likelihood Ratio
LTE	Long Term Evolution
M2M	Machine-to-Machine
ML	Maximum Likelihood
MLP	Multilayer Perceptron
mRRD	Modified Random Redundant Iterative Decoding

MSE	Mean Squared Error
NR	New Radio
PC	Parity Check
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
RNTI	Radio Network Temporary Identifier
ROC	Receiver Operating Characteristic
RS	Reed Solomon
RSC	Recursive Systematic Convolutional
SC	Successive Cancellation
SCL	Successive Cancellation List
SINR	Signal to Interference and Noise Ratio
SNR	Signal-to-Noise Ratio
SON	Self-Organizing Network
SPP	Syndrome Posterior Probability
URLLC	Ultra-Reliable Low-Latency Communication
V2X	Vehicle-to-Everything

1. INTRODUCTION

Error correction codes are widely used in modern communication systems in order to improve error performance. They enable reliable data transmission over noisy channels by adding redundancy to transmitted data. Since the introduction of the channel coding theorem by Shannon [1], a wide range of code families have emerged in pursuit of achieving the channel capacity with a low encoding and decoding complexity. The most remarkable codes include convolutional, turbo, low density parity check (LDPC) codes and, more recently, polar codes. Heterogeneous network architecture of 5G and beyond networks will enable the inter-connection of different networks and devices. Moreover, direct communication technologies such as device-to-device (D2D), machine-to-machine (M2M) and vehicle-to-everything (V2X) will enable the wireless communication nodes to communicate directly in local premises, independent of core networks. Such a non-hierarchical communication paradigm will require the adaptive configuration of communication parameters in order to provide flexibility across different scenarios and changing channel conditions. Hence, automatic identification of communication parameters will play a significant role in 5G and beyond networks. Thus, the blind detection and decoding of the control channel for the selected coding schemes have gained significant attention in the heterogeneous 5G and beyond networks to fulfill the ever-increasing demands and requirements [2].

In adaptive modulation and coding (AMC) systems, modulation and coding parameters are changed dynamically in order to adapt to the dynamic nature of channel states. The active parameters for data channels are sent through control channels, which introduces an overhead for the resources. On the other hand, configurations for the control channels are also becoming adaptive with the evolution of the communication systems, introducing the need for blind identification of the active parameters. In light of the aforementioned reasons, successful blind identification of coding parameters is of utmost importance for future communication networks. There are two approaches in blind code identification: (i) estimation of code parameters among an infinite set, or (ii) detection of the code parameters from a predetermined set of alternatives [3].

The former problem is NP-hard without having any prior information [4]. However, it is possible to assume that the code type is known for most practical cases since it is predefined by the employed communication standard. The latter problem is available in a cooperative context where the transmitter and receiver have agreed upon a set of parameters. Blind identification of a channel code, in the first step, consists of finding the information (input) length k, the codeword (output) length n, and therefore, the code rate $r = k/n$ of the encoder. In addition to the code rate, the generator matrix of the code should also be identified. There are also parameters specific to a code type. For instance, the generator polynomials and the constraint length for a convolutional code, the interleaver parameters for a turbo code, frozen bit indices for a polar code, puncturing pattern for a punctured code should also be estimated. Considering all these variables, statistical signal processing algorithms will not be adequate to solve the existing problems [5].

1.1. Motivation

Machine learning will be instrumental in designing and managing the highly complex 5G and beyond networks. In particular, features such as dynamic air interface, self-organizing network (SON) and network slicing bring challenges for network operation and maintenance. These problems fall into the field of machine learning, which provides new approaches and paves the way to bring significant performance improvements compared to traditional methods [6], hence has recently regained attention in the field of communications. Deep learning (DL) is a family of machine learning methods that uses artificial neural networks for learning multi-level representations and features in hierarchical architectures in pattern recognition problems. Deep neural network (DNN) architecture is inspired by the structure and function of the human brain. DL-based methods have been widely used in a variety of signal processing applications such as computer vision, speech processing, robotics, anomaly detection due to its strong ability of generalization [7]. It provides enormous performance gain in supervised learning tasks compared to other traditional learning algorithms [8]. Artificial neural networks also enable the solution of analytically complex problems by removing the dependency on statistical models. Recent studies show that the DL-based meth-

ods give promising results in a variety of tasks in wireless communication area such as equalizer design, modulation identification, channel decoding [5]. The main contribution of DL-based methods have been in the cases when there is a model uncertainty, e.g., in the case of correlated noise or fading channels. The DNN models are generally used to replace a particular sub-block of a communication system. However, it is also possible to replace even the whole communication system by an autoencoder [9]. Based upon the successful implementations of machine learning algorithms in the wireless communications area, DL can be applied to the blind code identification problems in order to increase the performance of the systems for both parameter estimation and classification tasks.

1.2. Contributions

Existing methods for blind code identification have several drawbacks that undermine the performance of the system. The generated mathematical models are typically based on statistical and algebraic concepts. Most of them have assumptions on coding parameters, message structure as well as channel and noise models. The proposed methods are not universal for all code types since they heavily rely on the encoder parameters and the analytical expressions vary drastically even for a minor change in the parameter. The algebraic blind code identification methods are not generally robust to noisy channel conditions due to their non-deterministic nature. The rank-based methods have higher error performance but the rank behavior of the codes have not been analytically derived for most code types, whereas the DL-based classifier is applicable to any type of code. The log-likelihood ratio (LLR) based heuristic methods have relatively high complexity and not suitable for most practical cases. Channel decoding-based detection methods have high complexity and delay due to their iterative nature, whereas parallel implementation of DNNs reduces the delay. The novel DL-based approach can overcome the aforementioned weaknesses of the existing methods since it does not require any prior information and able to capture hidden representations in the data.

The contributions of this thesis are summarized as follows.

- The existing literature on DL–based channel decoding has proven that DL models are able to learn the code structure [5]. In this thesis, we consider the use of DL for blind identification of channel codes which is a first in this domain to the best of our knowledge. Hence, it is anticipated that the DL approach will also outperform the existing methods in blind code identification literature. Accordingly, the simulation results show that the performance of DL classifier outperforms the existing methods such as the SPP-based [10] and SCL-based [11] methods for convolutional and polar codes, respectively.
- The existing code identification methods are not derived for all code types and most of them are introduced for only a specific code type. On the other hand, deep learning based algorithms have the advantage that they enable construction of universal detection and recognition algorithms for any code type. Hence, DL models offer a universal detection/classification scheme for all code types. In this thesis, we will verify the performance of the deep learning based code identification method for the most used channel code families such as convolutional, turbo and polar codes.
- The algebraic methods in the literature are not robust to noise and channel effects [12]. Analytical derivations for some cases are not available and can be cumbersome [13]. In contrast, the DL-based classifiers are able to learn the code structure under noise and channel conditions and do not require statistical models for noise or channel.
- In order to accomplish the tight latency and reliability requirements for ultra-reliable low-latency communication (URLLC) for 5G and beyond networks, DL-based classifiers provide low and predictable delay amounts by offering one-shot decision [14]. Some of the existing methods perform code detection by trying to decode the received bit stream for each code candidate and make decision based on a cumulative metric. Such methods have high computational complexity. Additionally, those decoding algorithms are iterative and may not meet the low delay requirements introduced by the standards.
- The simulation results show that the presented DL polar code classifier offers a

performance gain for signal-to-noise ratio (SNR) values higher than -4 dB compared to the SCL-based classification method. Since the existing polar code classification methods are based on the metrics originally used for decoding, they have poor performance for classification tasks.

- DL-based channel decoders suffer from the curse of dimensionality, i.e., there are 2^k possible codewords of length n for an (n, k) error correction code, so the number of all possible codewords grows exponentially. In this thesis, an analysis on the impact of the number of training codewords on the performance is provided. The results show that the DL code classifier is able to achieve adequate classification performance even utilizing a portion of the all possible codewords.
- The performance of the proposed method is verified under varying SNR conditions. The simulation results show that the classifier is robust to changes in SNR. The DL-based code classifier trained for polar codes at -4 dB has 80% accuracy at -6 dB and has accuracy higher than 90% for higher SNR values.

1.3. Thesis Outline

The rest of the thesis is organized as follows. A brief background on forward error correction codes is provided in Chapter 2. A background on deep learning is provided and the literature on deep learning based channel decoding is considered in Chapter 3. Blind code identification problem is introduced and the literature on blind code identification using classical approaches is provided in Chapter 4. The system model for the proposed classification method is given in Chapter 5. A deep learning based method for blind code identification is proposed and the performance of the method is verified for widely used convolutional, turbo and polar codes in Chapter 6. Finally, conclusions are drawn in Chapter 7.

2. FORWARD ERROR CORRECTION CODES

Forward error correction (FEC) codes, or channel codes, are used widely in order to improve the error performance of digital communication systems. They enable the detection of bit errors introduced by noise by adding redundancy to transmitted bits. Error correction codes can be categorized as block codes, convolutional codes, turbo codes [15].

A basic communication system with FEC coding is visualized in Figure 2.1. It consists of an encoder and a decoder, which maps the information to codewords and noisy channel outputs back to information respectively.

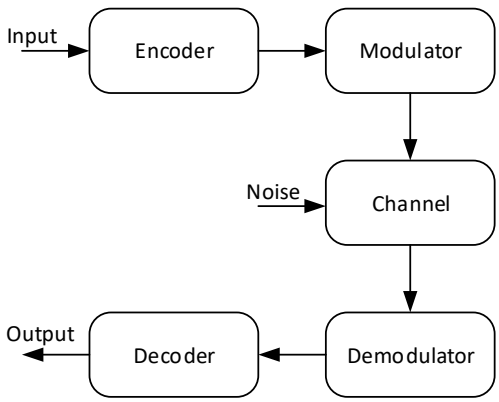


Figure 2.1. A basic communication system with channel coding. Transmitter encodes the information prior to modulation and receiver decodes the noisy codewords after demodulation.

A block encoder, as the name suggests, encodes the information sequence in blocks. A block code maps an information block of length k into a codeword of length n. The code rate r is the number of carried information symbols per codeword, given by $r = k/n$. The input-output relationship of a block code is represented by a $k \times n$

generator matrix $\mathbf{G}$

$$\mathbf{G} = \begin{pmatrix} g_{0,0} & g_{0,1} & g_{0,2} & \cdots & g_{0,n-1} \\ g_{1,0} & g_{1,1} & g_{1,2} & \cdots & g_{1,n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{k-1,0} & g_{k-1,1} & g_{k-1,2} & \cdots & g_{k-1,n-1} \end{pmatrix}. \tag{2.1}$$

The information vector $\mathbf{u}$ is encoded to codeword vector $\mathbf{v} = \begin{bmatrix} v_0 & v_1 & \cdots & v_{n-1} \end{bmatrix}$ by the following equation

$$\mathbf{v} = \mathbf{u}\mathbf{G}. \tag{2.2}$$

The dual code of C, denoted by $C^{\perp}$, is the code that spans the subspace orthogonal to the span of C. The $n \times (n-k)$ parity matrix $\mathbf{H}$ of C is the generator matrix of $C^{\perp}$. Each row of the parity matrix is orthogonal to the rows of the generator matrix

$$\mathbf{G}\mathbf{H}^T = 0. \tag{2.3}$$

Hence, a block code C can be described in two ways: a (n, k) code C is (i) the code generated by $\mathbf{G}$, or (ii) the code that is orthogonal to the space spanned by the rows of $\mathbf{H}$

$$\mathbf{v}\mathbf{H}^T = 0. \tag{2.4}$$

Since the generated codewords are transmitted through a noisy channel, the received vector $\mathbf{r}$ has errors and it is in the form

$$\mathbf{r} = \mathbf{v} + \mathbf{e} \tag{2.5}$$

where $\mathbf{e}$ denotes the error vector. The error vector is a sparse vector since the probability of error is low in a typical SNR range of operation. In order to detect and locate

the errors, the decoder computes the so-called syndrome of the received codeword. The syndrome vector $\mathbf{s}$ is given by the following expression

$$\begin{aligned}\mathbf{s} &= \mathbf{r}\mathbf{H}^T && (2.6)\\ &= (\mathbf{v}+\mathbf{e})\,\mathbf{H}^T = \mathbf{v}\mathbf{H}^T + \mathbf{e}\mathbf{H}^T = 0 + \mathbf{e}\mathbf{H}^T && (2.7)\\ &= \mathbf{e}\mathbf{H}^T && (2.8)\\ &= \begin{bmatrix} s_0 & s_1 & \dots & s_{n-k-1}\end{bmatrix}. && (2.9)\end{aligned}$$

Each element of the syndrome vector shows whether a parity check is satisfied or not. For instance, for a parity check given by $s_i = \begin{bmatrix}1\,0\,1\,1\,0\,0\,1\end{bmatrix}$, the codeword vector satisfies

$$v_0 \oplus v_2 \oplus v_3 \oplus v_6 = 0. \tag{2.10}$$

In addition to hard decision bits, some decoding algorithms use soft decisions for the received bits. In such a scheme, the demodulator outputs a soft decision metric for each received bit. Use of soft information improves the performance of decoders. More detailed information on soft decoding can be found in [15]. Soft decoding algorithms give higher coding gain compared to the hard decision decoding algorithms. One commonly used soft metric is the log-likelihood ratio, ℓ, which is defined as

$$\ell_i \triangleq \log\left(\frac{\Pr\left(v_i = 0|\gamma_i\right)\right)}{\Pr\left(v_i = 1|\gamma_i\right)}\right) \tag{2.11}$$

where the γ_i is the ith received symbol. One should also note that the absolute magnitude of the LLR value represents the reliability of the bit. For instance, the binary phase shift keying (BPSK) modulation scheme maps the input bits to symbols as

$$\gamma_i = \begin{cases} 1, & \text{if } v_i = 0 \\ -1, & \text{if } v_i = 1 \end{cases}. \tag{2.12}$$

Therefore, the LLR values for BPSK symbols under additive white Gaussian noise

(AWGN) channel are given by

$$\begin{aligned}\ell_i &\triangleq \log\left(\frac{\frac{1}{\sqrt{2\pi\sigma_N^2}}\exp\left(-\frac{(\gamma_i-1)^2}{2\sigma_N^2}\right)}{\frac{1}{\sqrt{2\pi\sigma_N^2}}\exp\left(-\frac{(\gamma_i+1)^2}{2\sigma_N^2}\right)}\right) \\ &= -\frac{(\gamma_i-1)^2}{2\sigma_N^2}+\frac{(\gamma_i+1)^2}{2\sigma_N^2} \\ &= \frac{2\gamma_i}{\sigma_N^2}\end{aligned} \tag{2.13}$$

where σ_N^2 denotes the noise variance. The LLR value for the sum of two bits is calculated by the boxplus operator

$$\ell_k = \ell_i \boxplus \ell_j \tag{2.14}$$

which is defined by

$$\ell_i \boxplus \ell_j \triangleq \tanh^{-1}\left(\tanh\left(\frac{\ell_i}{2}\right)\tanh\left(\frac{\ell_j}{2}\right)\right). \tag{2.15}$$

In order to enable the calculation for practical systems, the boxplus operator can be approximated by

$$\ell_i \boxplus \ell_j \approx \operatorname{sgn}(\ell_i)\operatorname{sgn}(\ell_j)\min\{|\ell_i|,|\ell_j|\} \tag{2.16}$$

where sgn function is given by

$$\operatorname{sgn}(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0 \end{cases}. \tag{2.17}$$

The syndrome posterior probability (SPP) of the parity check s_i satisfying the

expression

$$v_0 \oplus v_1 \oplus \cdots \oplus v_{p-1} = 0 \tag{2.18}$$

is given by

$$\ell_{s_i} \triangleq \boxplus_{j=0}^{p-1} \ell_j = \ell_0 \boxplus \ell_1 \boxplus \cdots \boxplus \ell_{p-1}. \tag{2.19}$$

Convolutional codes encode the information by a sliding window. Convolutional codes have memory and the number of memory elements is called the constraint length (K) of the code. An example convolutional encoder is shown in Figure 2.2. The sliding window structure of convolutional codes can be modeled as a trellis, which facilitates low complexity decoding algorithms by employing time-invariant trellises. Convolutional codes are widely used in wireless communication systems such as cellular and satellite communications since they enable reliable high data rate applications. More detailed information on convolutional codes and decoding algorithms can be found in [16]. The effective generator matrix of a convolutional code can be represented as a matrix where each element is a polynomial over finite field $\mathbb{F}_2$

$$\mathbf{G}(D) = \begin{pmatrix} g_{0,0}(D) & g_{0,1}(D) & \cdots & g_{0,n-1}(D) \\ g_{1,0}(D) & g_{1,1}(D) & \cdots & g_{1,n-1}(D) \\ \vdots & \vdots & \ddots & \vdots \\ g_{k-1,0}(D) & g_{k-1,1}(D) & \cdots & g_{k-1,n-1}(D) \end{pmatrix}. \tag{2.20}$$

The encoding operation is given by

$$\mathbf{v}(D) = \mathbf{u}(D)\mathbf{G}(D). \tag{2.21}$$

For instance, the generator matrix of a rate-1/2 convolutional code in polynomial

representation [15] is given as

$$G(D) = \left[D^8 + D^6 + D^5 + D^4 + 1 \quad D^8 + D^7 + D^6 + D^5 + D^3 + D + 1 \right] \tag{2.22}$$

and representation in octal base is

$$G(D) = \left[561 \quad 753 \right]. \tag{2.23}$$

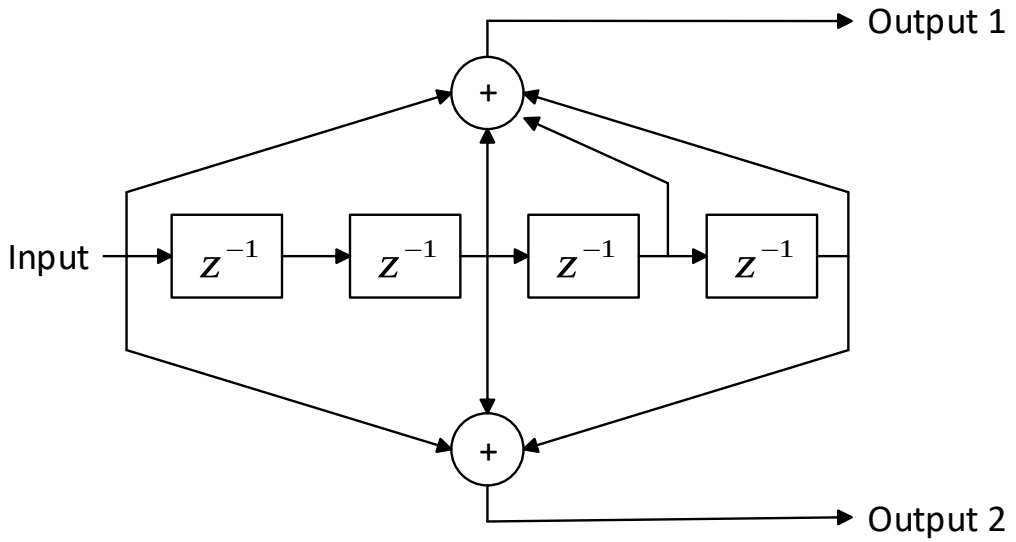


Figure 2.2. An example convolutional encoder. z^{-1} denotes the delay operation.

The performance of channel codes can be enhanced by parallel or serial concatenation of multiple codes. For instance, convolutional codes are concatenated with Reed Solomon (RS) codes for improving the performance in satellite communications [17]. Another approach for concatenation of convolutional codes is turbo coding. Turbo codes are constructed by parallel concatenation of two or more convolutional codes. The interleaved versions of the input are fed into each convolutional encoder in order to provide random-like coding [16]. Turbo codes are used for encoding of data channels in Long Term Evolution (LTE) cellular standard. A typical turbo encoder consists of two

recursive systematic convolutional (RSC) encoders in parallel and a pseudo-random block interleaver, see Figure 2.3.

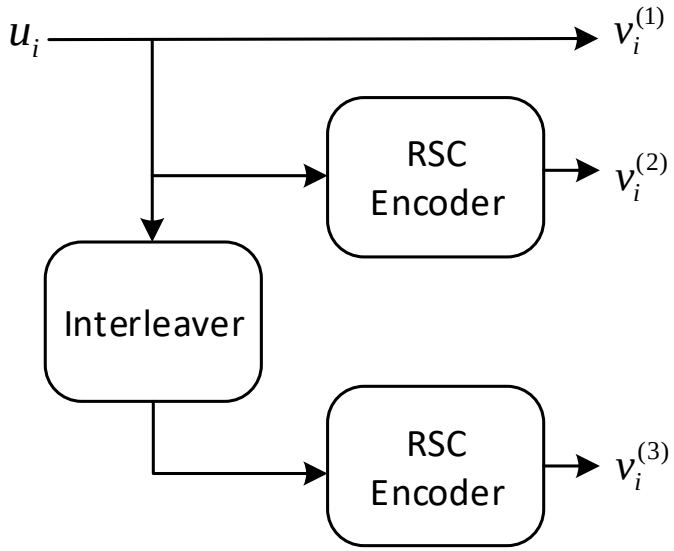


Figure 2.3. A typical (rate-1/3) turbo encoder consists of two RSC encoders and an interleaver.

Another emerging class of channel codes is polar code, which utilizes the phenomenon named channel polarization [18]. Channel polarization corresponds to synthesis of new channels from n independent binary symmetric channels so that the capacities of a portion of the channels converge to 1 while the capacities of the others converge to 0.

The most reliable k channels are used for transmission of information bits, while a fixed value (mostly 0) is transmitted on the remaining $n-k$ channels. Those $n-k$ channels are called the frozen bits. A polar code of length $n = 2^m$ with input length k is denoted as $P(n, k, A_f)$, where A_f stands for the frozen bit set for the code.

The generator matrix for polar codes is constructed by a recursive relationship and has the form

$$\mathbf{G} = \mathbf{F}_2^{\otimes m} \tag{2.24}$$

where the kernel $\mathbf{F}_2$ is given by

$$\mathbf{F}_2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \tag{2.25}$$

and $\otimes$ operator denotes the Kronecker product, defined as

$$A \otimes B = \begin{pmatrix} a_{1,1}B & \dots & a_{1,n}B \\ \vdots & \ddots & \vdots \\ a_{m,1}B & \dots & a_{m,n}B \end{pmatrix}. \tag{2.26}$$

The encoding relationship is

$$\mathbf{v} = \mathbf{w}\mathbf{G} \tag{2.27}$$

where the vector $\mathbf{w}$ consists of k information bits and $n - k$ frozen bits.

Arıkan proposed a recursive decoding scheme for polar codes, called the successive cancellation (SC) decoding [18]. The data dependency graph (DDG) is traversed and decision LLRs are produced for estimation of the bits in this decoding scheme. However, SC decoding is sub-optimal for short code lengths. In addition, SC decoding has error propagation problem due to its sequential nature. The complexity of SC decoding is reduced by the introduction of Simplified SC decoding which prunes the unnecessary nodes in the decoding tree [19]. In order to improve the error performance of polar codes, several approaches are presented in the literature. Instead of making decisions at each decoding stage, successive cancellation list (SCL) decoding considers up to L different candidate paths [20]. The number of decoding paths increases exponentially

in each stage since both cases are considered for each bit value. In order to limit the complexity, the most reliable L paths are selected via a path metric at each stage. Furthermore, an enhancement for polar codes is introduced by concatenation of cyclic redundancy check (CRC) and parity check (PC) codes. The CRC-aided SCL decoding [21] has provided a significant performance boost for polar codes so that it outperformed the state of the art LDPC codes. In addition to the SC decoder and derivatives, the belief propagation (BP) decoding can be used for polar decoding [22]. BP decoding is based on factor graph representation of polar codes. However, the BP decoding for polar codes suffers from higher implementation complexity and a lower throughput than the SC-based decoders [23].

3. DEEP LEARNING

In this section, a brief information on DL is presented. A more comprehensive discussion on DL can be found in [8]. DL methods employ deep neural networks for performing regression and classification tasks. Neural network architecture is inspired by the structure and function of the human brain [24]. DL approach has been widely used in a range of signal processing applications such as computer vision, speech processing, robotics due to its strong ability of generalization.

A deep feed-forward neural network, or a multilayer perceptron, describes a nonlinear mapping between its input and output. A feed-forward network with a single hidden layer is able to approximate a continuous function f on compact subsets of $\mathbb{R}^n$ [25]. It has a layered structure, where each layer consist of interconnection of many neurons, as shown in Figure 3.1. The mapping in each layer generally consists of two steps: a linear transformation defined by a weight and a bias, plus a nonlinear activation function, σ, for constraining the output in a finite interval. Use of an activation function enables the neural network to learn nonlinear relationships between the input and output of functions. The input-output relationship of a single layer can be expressed as

$$\mathbf{y} = \sigma\left(\mathbf{W}\mathbf{x} + \mathbf{b}\right) \tag{3.1}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the input and output vectors and $\mathbf{W}$ and $\mathbf{b}$ denote the weight matrix and the bias vector, respectively. Hence, the relationship between the input and the output of a multilayer perceptron is given by

$$\mathbf{y} = \sigma\Big(\mathbf{W}\sigma\big(\mathbf{W}\cdots\sigma\left(\mathbf{W}\mathbf{x} + \mathbf{b}\right) + \mathbf{b}\big) + \mathbf{b}\Big). \tag{3.2}$$

The first layer in a neural network is called the input layer and the last layer is called the output layer. The rest are the hidden layers, which are not observable

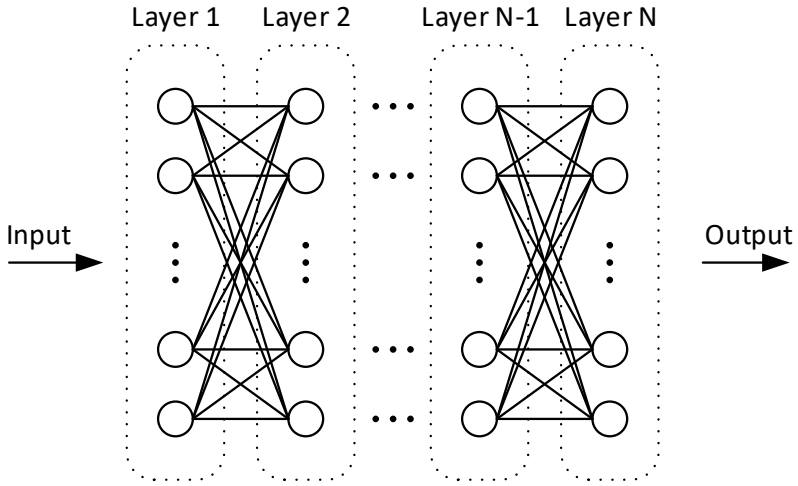


Figure 3.1. Multilayered neural network structure. Circles denote the nodes and lines denote the connections between the nodes of neighboring layers. Connections represent a linear transformation followed by a non-linear activation function.

from input or output. A multilayer perceptron is called feed-forward since there is no feedback connection and the information flows in forward direction. Some of the common activation functions are listed in Table 3.1 such as rectified linear unit (ReLU).

The network parameters $\mathbf{W}$ and $\mathbf{b}$ are optimized by the gradient descent algorithm, which optimizes a loss function by iteratively updating the parameters in the direction of a steepest descent. Typical loss functions used in deep learning applications can be listed as cross-entropy, mean squared error (MSE) and Hinge loss. For instance, cross-entropy loss is given by

$$J = -\sum_{i=1}^{N} p_i \log q_i \tag{3.3}$$

where p_i and q_i denote the probabilities of the ith element of estimated and true values, respectively. Each iteration step of the gradient descent algorithm is named an epoch. An epoch consists of two stages: (i) forward propagation and (ii) backward

Table 3.1. List of widely used activation functions.

Name	**Expression**
ReLU	$\max(0, z)$
Leaky ReLU	$\max(0, z) + 0.01 \min(0, z)$
Parametric ReLU	$\max(0, z) + \alpha \min(0, z)$
Sigmoid	$\frac{1}{1+e^{-z}}$
Tanh	$\frac{1-e^{-2z}}{1+e^{-2z}}$
Softmax	$\frac{e^{z_i}}{\sum_{i=1}^{N} e^{z_i}}$

propagation. In forward propagation stage, the output of the network is calculated using the current weights and the loss is calculated by comparing to the true values. In backward propagation stage, the gradients are calculated for each layer successively from the output layer to the input layer and the weights are updated in the direction of the gradients.

There are also different type of layers in addition to fully connected layer such as convolutional layer and recurrent layer. In a convolutional layer, the matrix multiplication operation is replaced by the convolution operation. The convolutional layer is simply a filter performing convolution along the input data. In a convolutional layer, the connections between the neurons of neighboring layer are sparse, which decreases the complexity of the layer. convolutional neural network (CNN) architecture is widely used in computer vision applications since the input data-sets are generally unstructured [7].

A recurrent layer has also a feedback connection which provides a memory for the layer. A recurrent neural network (RNN) stores the information in context nodes, which gives it the ability to process and learn the data in sequences. Having a memory makes RNN architecture a good candidate for time-series applications, where there is a correlation between the successive time instances. Hence, the RNN architecture is

widely used in natural language processing and anomaly detection applications [8].

3.1. DL-Based Channel Decoding

A considerable amount of literature has been published on the application of DL to channel decoding. These studies generally replace the conventional channel decoders with a DNN model in order to improve the bit error rate (BER) performance of the systems. The proposed channel decoders process the received bits or the LLR values directly or in one-hot encoding scheme. One of the biggest problems in the area of DL is the construction of labeled data-set since it is time-consuming and expensive. Fortunately, labeled data collection for channel coding is an easier task because the data consist of artificial signals. In addition, the DNN models can be trained with noisy versions of the codewords in order to avoid overfitting. Conventional channel decoders generally have an iterative structure leading to a relatively higher latency. DL-based channel decoders have lower latency by offering a one-shot decision property [14]. There are 2^k possible codewords of length n for an error correction code, which means that the data-set grows exponentially. Therefore, the implementation of the DL-based channel decoders are restricted by the dimensionality of channel codes [26].

A Tanner graph of a code with the parity matrix $\mathbf{H}$ is a bipartite graph, which has one vertex for each row of $\mathbf{H}$ (check node) and one vertex for each column of $\mathbf{H}$ (variable node). There is an edge between the ith check node and jth variable node if the element of $\mathbf{H}$ in the ith row and the jth column is non-zero. The BP algorithm updates the nodes of a Tanner graph iteratively in order to produce decisions for a low density code such as an LDPC code. However, the BP decoding algorithm does not perform well for high density parity check (HDPC) codes. An improvement for the BP algorithm using deep learning is proposed in [27]. The proposed method introduces a soft Tanner graph by assigning weights to the edges in the graph and trains them via a neural network. The assigned weights are optimized for different codes by using deep learning methods. The proposed soft Tanner graph structure increases the performance of BP algorithm for HDPC codes. Furthermore, it is possible to train the network by using noisy versions of a single codeword, since the performance of the BP algorithm does not depend on the

transmitted codeword. The proposed soft BP decoder is improved in [28] by using an RNN architecture. This so called BP-RNN architecture improves the performance by reducing the required number of parameters. It also adapts the multiloss concept, which calculates loss for each time step. A variation of the BP-RNN decoder is introduced by employing the modified random redundant iterative decoding (mRRD) algorithm. The BP-RNN decoder is further improved in [29] by using the off-set min-sum algorithm instead of sum-product algorithm. The neural off-set min-sum decoder have a lower computational complexity compared to the original BP-RNN decoder. The min-sum neural decoder is further improved in [30] by introducing a new architecture for the RNN decoder. In [31], a similar approach is applied to polar codes.

An alternative loss function, syndrome loss, for neural decoders is proposed in [32]. The introduced loss function is applicable only to decoder models that output codeword estimates such as ones that are based on the BP decoder structure. In [33], a neural network based decoder is presented. The proposed model performs syndrome decoding by having the syndrome values as an input to the decoder.

A feed-forward neural network is used for channel decoding in [14]. The proposed method is applied for decoding polar codes and random codes of short code lengths. The obtained results show that DNNs are capable of generalizing a subset of codewords to decode the previously unseen codewords. In [34], the plain neural decoder is implemented via different architectures such as multilayer perceptron (MLP), CNN and RNN. In [35], in order to decode codes with larger codeword lengths, decoding is performed for sub-codewords and then combined. This scheme relies on the graph partitioning of polar codes. In [36], a similar approach utilizing graph partitioning is presented. The latter combines the codewords via SC decoding as opposed to BP decoding in [35].

A neural network polar-LDPC decoder is proposed in [37] by appending an indicator bit to the input codewords. The proposed system is able to decode a polar code and an LDPC code with the same input and output lengths. The authors state that a feed-forward neural network performs better than CNNs and RNNs for the given setup.

In addition to the decoding of classical channel codes, channel codes are designed by using RNN encoders and decoders in [38]. The authors state that the designed RNN driven code, named deepcode, have better BER performance compared to the existing codes.

The literature on DL channel decoding have demonstrated that DNNs are able to learn the structure of error correction codes. The capability of DNNs for learning and generalizing the code structure can be extended for identification of code parameters. The DL approach is a promising candidate for blind code identification tasks.

4. BLIND CODE IDENTIFICATION

Blind identification of a channel code is simply drawing inference about the parameters of a channel encoder by observing its output. There are several scenarios that may require blind identification of channel codes:

- The first scenario is formulated as the code detection problem: determine whether a received bit stream is encoded with a certain code. The code detection problem may arise in a blind decoding process. By utilization of blind decoding, a receiver can determine the existence of a payload data without requiring any control signaling.
- The second scenario is the code classification problem: among a predetermined code set, find which of the codes is used to encode a given bit stream. In 5G and future networks, classification of the codes will be crucial for heterogeneous network architecture since the automatic identification of transmitter parameters will play a key role in such systems.
- The third scenario is the code recognition problem: given a bit stream, determine the used code without prior knowledge of a code set. In this scenario, there is no a priori knowledge about the code, e.g., in a cryptanalysis system.

The most general code parameters can be listed as, but not limited to

- information (input) length k,
- codeword (output) length n,
- code rate $r = k/n$,
- generator matrix,
- parity check matrix.

There are also parameters specific to a code type. For instance, the constraint length and the generator polynomials for a convolutional code, the interleaver parameters for a turbo code, frozen bit indices for a polar code, puncturing pattern for a punctured

code should also be estimated.

Blind code detection and recognition tasks can be performed by a classification or a regression model, respectively. Blind recognition of code parameters can also be performed by a classification algorithm. For instance, in order to find the codeword length of a code, classification among a set that consists of codes with all possible codeword lengths can be used. Such a model can decrease the complexity of the models and improve the error performance. A typical layout of the blind code identification systems is shown in Figure 4.1. In the first stage, the code parameters are identified by a code classification algorithm. In the second stage, the codeword is decoded by a decoder configured with the corresponding parameters.

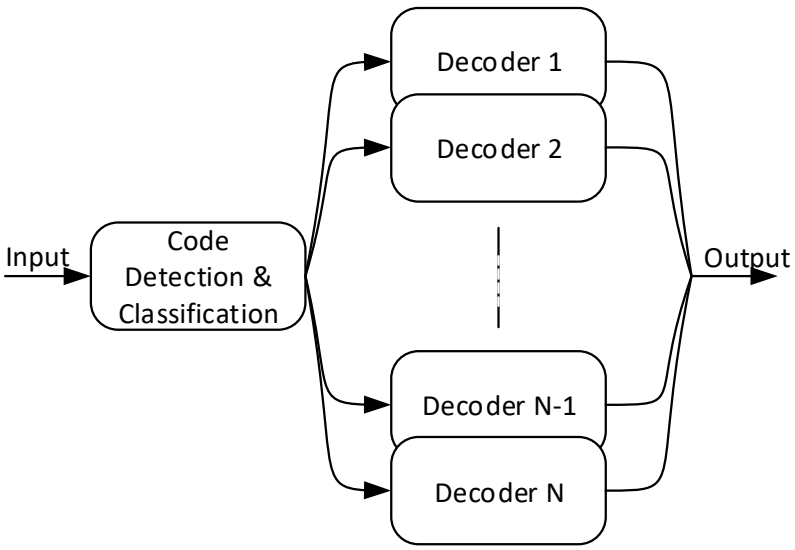


Figure 4.1. A basic blind decoding system consists of a detection/classification stage and parallel decoders with each code in the set. Input is the received bit stream and the output is the decoded bit stream.

The existing literature on blind code identification generally exploits the algebraic structure of the error correction codes. Although there are nonlinear error correction codes, the most widely used codes are linear since they allow for more efficient encoding

and decoding algorithms [15]. Channel encoders for linear codes encode the information according to a linear transformation, represented by a matrix multiplication in a finite field, specifically Galois field. As a result, channel encoder induces a linear dependence between the coded symbols. In other words, codewords belonging to an (n, k) code span a k-dimensional subspace of the vector space of all the n-tuples. The existing methods generally exploit this linear dependence between the code symbols, as well as the dependence between the consecutive codewords for identification of the encoder parameters. In [3], the code detection problem is formulated analytically and two maximum likelihood (ML)-based detection algorithms are proposed as a solution. The algorithms first generate a set of candidates for dual codewords by searching low weight combinations of the codewords, then eliminate the least likely ones by using different detection criteria. The proposed method is improved by adding an a priori iterative decoding stage in [39]. In [40], an analysis on the required number of intercepted bits for the algorithm is provided. The results show that the required number of bits for LDPC codes turns out to be proportional to the logarithm of the codeword length. Although the ML algorithm provides the optimum detection results, it has high computational complexity. An analysis on the computational complexity of ML code detection is provided in [4].

A joint estimation algorithm for interleaver period and code rate is proposed in [41] for the noiseless case. The algorithm constructs an interception matrix by stacking the received bit stream column-by-column. Since there is a linear dependence between the codewords when the number of rows matches the codeword length, the interception matrix becomes rank-deficient. Hence, the codeword length and the code rate can be determined by observing the rank behavior of the matrix for different row lengths. The rank-based estimation algorithm is improved in [42] by shuffling the columns of the interception matrix. The introduced l-randomized rank metric improves the performance of the algorithm in a noisy scenario.

A code detection method in a noisy environment utilizing the syndrome checks is proposed in [43]. A rank-based method for estimating the interleaver period and frame synchronization is proposed in [44]. The proposed method uses Gauss-Jordan

elimination through pivoting (GJETP) algorithm for finding the dual space of the codes. The algorithm performs better when soft information is available. Even though the presented approach is applied to the estimation of interleaver period and frame synchronization, the parity check estimation method can also be used to identify other coding parameters, too. However, the rank behavior of each code differs significantly and there is no general solution to the problem yet. A similar approach can be used for finding the code length and code block synchronization [45]. An improvement for the GJETP-based algorithm is proposed in [46]. The GJETP-based algorithm is used for code classification between block codes, convolutional codes and uncoded bit streams as well as estimation of interleaver parameters. Once the code length is found, the dual codewords can be found by an exhaustive search by utilizing the Chose-Joux-Mitton algorithm [47] or Canteaut-Chabaud information set decoding algorithm [48]. The study in [49] extends the similar problem to non-binary codes. In [50], the rank-based algorithm is considered for linear block codes and generalized to convolutional and turbo codes. A more general solution for rank-based methods is provided in [51]. The method introduces the partial Gaussian elimination algorithm which is performed by searching for the low weight column sums in each step.

Detection of a linear block code or a convolutional code from a predetermined set is considered in [10]. The proposed method calculates SPP for each code in the set as a detection metric. The intuition behind the algorithm is that if a bit stream is encoded by a particular code, it satisfies the syndrome checks of the code. Moreover, the use of LLR values instead of hard decisions for detection metric calculation improves the detection performance. The error performance of the proposed algorithm is further analyzed in [52]. A similar approach using average likelihood difference (LD) instead of average LLR is used in [53]. The SPP based algorithm is considered for the detection of binary and non-binary LDPC codes in [54,55], respectively, when the signal parameters such as signal amplitude and noise variance are not known a priori. Accordingly, the signal parameters are estimated using the expectation maximization (EM) algorithm prior to detection.

A generalized likelihood ratio test (GLRT) based detection method for block codes is proposed in [56]. The optimum threshold for detection is computed by Neyman-Pearson criterion. The Groebner basis is used for estimation of dual codewords in [57]. In [58, 59], the distribution of Euclidean distances between codewords is considered for codeword length estimation. The blind identification of LDPC codes under flat fading channel is considered in [60]. The proposed method uses the EM algorithm for parameter estimation and SPP based method for code detection.

The algebraic properties of convolutional codes are examined and a method for estimation of the generator polynomials is proposed in [61]. The EM algorithm is used for estimation of convolutional code parameters in [12]. The method utilizes the LLR values and applies LLR algebra for the calculation of the objective function and gradients. In [62], it is shown that a multi-order key equation can be used for recognition of a rate-1/2 convolutional code and the equation can be solved by the Euclidean algorithm. The proposed method is further improved by adapting the Euclidean algorithm to a more general model in [63].

The rank-based estimation algorithm utilizing GJETP is used for blind recognition of convolutional codes of rate $(n-1)/n$ in [64]. The proposed method is generalized for rate k/n codes in [65]. The algebraic properties of dual codes are examined and the equivalency of recursive systematic codes to non-recursive non-recursive codes is shown in [66]. The method is generalized for non-binary convolutional codes in [67]. It is adapted to the soft decision scenario by providing a scheme for the construction of the interception matrix in [68]. The proposed rank-based methods does not provide an analytical solution for the relationship between the rank of the interception matrix and the parameters of convolutional codes. The relationships derived in those papers only depend on empirical results. An analytical solution for the rank behavior is presented in [69]. The study considers various encoder-dependent parameters that generate the rank deficiency. Another attempt at providing an analytical solution for the rank problem is given in [13]. The solution formulates the relationship between the code parameters and the rank of the interception matrix.

A joint estimation algorithm for convolutional code and block interleaver is presented in [70]. The method uses the algorithm provided in [45] for searching the low weight parity checks. In [71], a method for convolutional code detection utilizing the collision counts is proposed. In [72], a method using Walsh-Hadamard transform is considered for the estimation of convolutional code parameters. The method is able to estimate the parameters of a rate $1/n$ code. An improved version of the method is given in [73] for rate $(n-1)/n$ convolutional codes. The proposed method uses segmented Walsh-Hadamard transform for decreasing the computational complexity of the algorithm. In [74], the block matrix method is introduced for further decreasing the computational complexity of the algorithm. The block matrix method is able to estimate the parameters of a rate k/n convolutional code.

Blind parameter estimation for punctured convolutional codes is considered in [75]. The proposed work determines the parity check matrix, the generator matrix and the puncturing matrix in order. In [76], the ML code detection algorithm introduced by Valembois [3] is used for the estimation of convolutional code parameters in the case that the code is punctured. The presented method finds the codeword length n and the matrices for the mother code and puncturing pattern. A similar method is proposed in [77]. In [78], the rank-based method for convolutional codes [64] is generalized for punctured convolutional codes.

A work on blind identification of turbo codes is given in [79]. The proposed method finds the generator polynomials by the algorithm given in [61]. The author introduces two different methods for the estimation of interleaver parameters; (i) a dynamic trie structure-based method, and (ii) a first-order statistical test-based method for noiseless and noisy channels, respectively. [80] presents a method for estimating the parameters of the second constituent encoder of turbo codes. The proposed method is generalized for rate k/n encoders in [81]. Another estimation algorithm for turbo code interleaver is given in [82]. In [83], a method using the algorithm in [45] is introduced. A rank-based method for turbo code identification that estimates the parameters of the constituent encoders, as well as the interleaver period is given in [84]. A similar method estimating also the puncturing pattern is given in [85]. A least squares based

method for the estimation of the encoder polynomials of a turbo code is introduced in [86]. In [87], the turbo code identification problem is formulated as Bahl, Cocke, Jelinek and Raviv (BCJR) algorithm and the interleaver pattern is found via the forward pass of the BCJR algorithm. In [88], the method is improved by introducing an early stopping criterion. A detection algorithm for turbo codes within a predetermined set is presented in [89]. The EM algorithm is employed for estimation of constituent encoder parameters in [90, 91]. In [92], a Monte Carlo method named Gibbs sampling is used for parameter estimation of the constituent codes of a turbo code. In [93], an identification algorithm for the internal interleaver of a turbo code is introduced. The interleaving pattern is found by using parity checks of the constituent encoders.

A blind detection scheme for polar codes for 5G physical control channel is proposed in [11]. It suggests to send radio network temporary identifier (RNTI) instead of some of the frozen bits. The proposed scheme eliminates the code candidates from a predetermined set in two stages. In the first stage, the received codeword is SC decoded with all code candidates from the set and the path metrics produced by each decoder are used as an elimination criterion. Similarly, the second stage uses the path metrics from the more sophisticated SCL decoder for a final decision. The performance of the proposed scheme is further improved by employing SCL decoder, also in the first stage in [94]. The increase in the complexity is handled by sharing the decoders in both stages. A similar approach using fast-SSC decoder for polar code detection is proposed in [95]. The path metric used in SCL decoding algorithm is modified to increase the code detection performance. Since the number of frozen bits contained in each node type is different, the detection metric can be updated at each type of node differently. For instance, a Rate-0 node is a good indicator for the code type since it consists of only frozen bits, whereas Rate-1 nodes give no information about the code structure since they consist of only information bits. Another blind polar code detection algorithm based on SCL decoding is proposed in [96]. The proposed D-metric is simplified in a way that only the LLRs corresponding to the frozen bits are taken into consideration. In [97], the authors propose a blind detection scheme for 5G New Radio (NR) polar codes. The presented method weighs the path metrics obtained in SCL decoder by the number of frozen bits in order to improve the detection performance. Another

two stage system for blind polar code detection is proposed in [98]. The proposed system calculates the detection metric based on BP decoder for polar codes utilizing an early stopping criteria. Three different early stopping criteria from the BP decoding literature are offered for detection.

5. SYSTEM MODEL

5.1. Neural Network Structure

In this thesis, a feed-forward neural network structure is used in code detection and classification models. Performance of the neural network code identification models may be further improved by using other network structures such as CNN or RNN for a particular channel code type but a feed-forward model is considered in here as the first step. Since the encoding relationships for channel codes are represented by dense matrix operations, the feed-forward neural network architecture seems to be a promising candidate for the code classification tasks. The sparse structure of the CNN architecture may decrease the training and prediction complexity for convolutional codes. However, in order to have a universal code identification method, we will consider a feed-forward neural network.

The performance of DNN models heavily depends on the data-set. If the training data-set is not sufficiently diverse, the model may not be able to learn the general structure and overfit the training data-set. Overfitting occurs when the model achieves such a good fit on the training data that it is not able to generalize well on previously unseen data. In other words, the model learns hidden patterns specific to training data which do not appear in other data. There are several ways to prevent overfitting in neural network models. The best option is to increase diversity by increasing the number of samples in training data-set. However, increasing the amount of training data brings a burden in time, budget and technical aspects. In order to prevent overfitting, a distinct noise sample can be added to the training data in each epoch. It can be achieved by generating the training data-set without noise and adding a non-trainable, noise layer before the input layer of the neural network model, as shown in Figure 5.1(a). This noise layer has no trainable parameters, i.e., its parameters are not optimized by the gradient descent algorithm and adds AWGN with a constant variance to the samples in each epoch. It is also possible to change the noise variance gradually in each epoch but in this thesis, a constant noise variance will be used for an analysis

on the effect of training SNR. Moreover, the code classifier models should be trained with a data-set containing adequate number of codeword combinations in the input. The proposed system model in Figure 5.1 removes the need for the diversity in noise samples in the data-set by adding a distinct noise sample in each epoch. However, the code dimensionality problem is still an issue, i.e., the number of all possible codewords, therefore, the size of the data-set increases exponentially with the input length of the error correction code. The order of the codewords are shuffled in order to prevent overfitting.

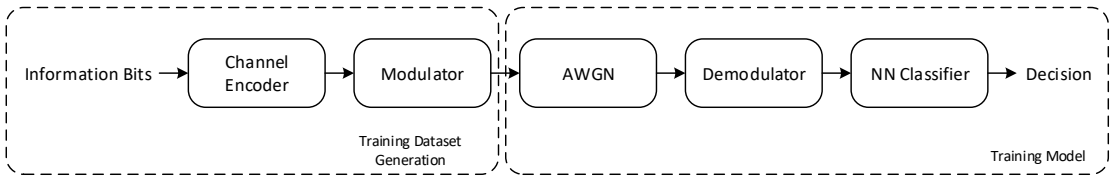
5.2. Model Training

Throughout the thesis, the system model in Figure 5.1 is used for each detection and classification task. The training data-set is generated by the following steps: (i) all possible bit sequences of length-k (information length of the code) are generated, (ii) the generated information vectors are encoded with the corresponding channel encoder(s), (iii) the codewords are BPSK modulated. In training phase, a layer for adding AWGN $\mathcal{N}(0, \sigma_N^2)$ and soft demodulation are appended to the input of the network which are non-trainable. Once the model is trained, the non-trainable layers are removed and the test data-set consists of noisy samples.

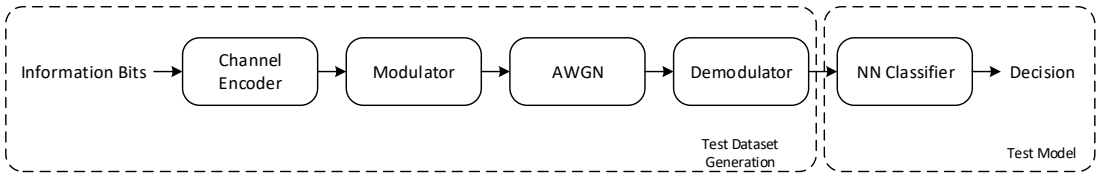
The neural network model used for the classification tasks is a feed-forward neural network. A more sophisticated neural network structure can be adopted for further improvement but it is sufficient to consider this structure for this work. Rectified linear unit (ReLU) function is used as the activation function for each layer except the output layer. ReLU function is given as

$$\sigma_{\mathrm{ReLU}}(z) = \max(0, z). \tag{5.1}$$

In the detection models, the activation function of the output layer is a sigmoid since the model performs binary classification. On the other hand, softmax function is used



(a)



(b)

Figure 5.1. System model for blind code classification for (a) training and (b) test, respectively. Note that the data-set consists of noiseless codewords for training stage and a distinct noise sample is added by the AWGN layer in each epoch.

for the multi-class classification models. The sigmoid function is as follows

$$\sigma_{\text{sigmoid}}(z) = \frac{1}{1+e^{-z}} \tag{5.2}$$

and the softmax function is given by

$$\sigma_{\text{sofmax}}(z) = \frac{e^{z_i}}{\sum_{i=1}^{N} e^{z_i}}. \tag{5.3}$$

One-hot encoding [8] is used for class labels. One-hot encoding is the representation of categorical variables as a binary vector. The length of the vector is equal to the number of classes. The one-hot vector for class c has a 1 in the cth location and 0's in the other locations. Adam optimizer [99] with learning rate 1×10^{-3} is used as an optimizer during training.

Steps of the training data-set generation is as follows:

(i) 2^k information vectors of length k are generated for each code in the set.
(ii) Each information vector is encoded by the corresponding channel encoder.
(iii) 2^k uniformly distributed random vectors of length n are generated for the random class.

In each epoch of the training phase:

(i) Each vector (both encoded and random) is encoded via BPSK modulation.
(ii) The non-trainable AWGN layer adds a distinct noise sample to the BPSK modulated codewords.
(iii) The noisy symbols are demodulated via BPSK soft demodulator, which outputs LLR values.
(iv) Output of the neural network is calculated with the current parameters.
(v) The cross-entropy loss is computed by Equation 3.3.
(vi) Layer parameters are updated sequentially from the last layer to the first layer.

Steps of the test data-set generation is as follows:

(i) Information vectors of length k are generated for each code in the set.
(ii) Each information vector is encoded by the corresponding channel encoder.
(iii) 2^k uniformly distributed random vectors of length n are generated for the random class.
(iv) Each vector (both encoded and random) is encoded via BPSK modulation.
(v) AWGN noise is added to each vector.
(vi) The noisy symbols are demodulated via BPSK soft demodulator, which outputs LLR values.

In the test stage:

(i) Output of the neural network is calculated with the trained parameters.

(ii) Performance metrics such as accuracy is calculated.

The parameters adopted in system simulations are given in Table 5.1. The network parameters are determined experimentally through a grid search and the listed parameters are the ones that give the highest classification accuracy. The columns of the table shows the parameters of rate-1/2 and rate-1/3 convolutional codes, turbo code and polar code, respectively.

Table 5.1. Parameters used for the simulations. $C(n, k, \nu)$ denotes rate-k/n convolutional code with constraint length ν.

	$\mathbf{C(2,1,9)}$	$\mathbf{C(3,1,9)}$	**Turbo**	**Polar**
# of classes	2	2	2	4
sample length	80	120	132	128
# of codewords	2^{18}	2^{16}	2^{16}	2^{16}
# of nodes	128, 64, 32	128, 64, 32	128, 64, 32	512, 256, 128, 64, 32
# epochs	1000	1000	1000	500

The performance of the DNN classifier is tested under detection set-up, i.e., binary classification for convolutional and turbo codes. The code detection problem is a binary decision problem where the alternate hypothesis corresponds to reception of an encoded bit stream and the null hypothesis corresponds to reception of a random bit stream. The data-sets for detection tasks contains two classes: (i) randomly generated codewords and (ii) uncoded random bit streams. In addition to binary classification, the performance of the DL classifier is verified also for multi-class classification of polar codes. In a similar fashion, the data-sets for multi-class classification tasks constructed with codewords from each code class and uncoded random bit streams.

6. BLIND CODE IDENTIFICATION USING DEEP NEURAL NETWORKS

Blind code detection and recognition tasks can be performed by a classification or a regression model, respectively. On the other hand, blind recognition of code parameters can also be performed by a classification algorithm. For instance, in order to find the codeword length of a code, classification among a set consisting of codes with all possible codeword lengths can be used. Such a model can decrease the complexity of the models and improve the error performance. Similarly, code detection tasks can be performed by a recognition algorithm whether the true parameter is found or not. Hence, a DL model designed for code classification, can be used for the identification of any parameter of a error correction code.

The blind code identification methods in the literature are generally based on algebraic and statistical methods. Those methods are proposed for only a single code family and not applicable or not analytically derived for another code type. In contrast, the DL-based classification models are able to learn the structure of any code family if they are trained with a sufficient number of codewords. The algebraic methods are not robust to noise and channel effects, and their performance drops dramatically with increasing noise levels [12]. However, DL-based methods are capable of learning the noise and channel statistics if they are trained properly. The feed-forward neural network structure does not have any feedback connections and iterations. Hence, they can perform classification tasks in a single pass. It provides lower and predictable delay values, which is a crucial parameter for low-latency communication systems.

In this thesis, the code identification problem for the most widely used code families such as convolutional, turbo, and polar codes are considered. The detection problem for convolutional and turbo codes are considered as a binary classification scenario, where there are two hypotheses whether the received bit stream is encoded by a code or it is a random bit stream. The detection problem for polar codes is

considered as a multi-class classification task in which there are three code classes and a random bit stream. General structure of a blind code identification system is given in Figure 6.1. The code classification block receives the demodulator output as an input and decides whether the input bit stream is encoded or not. If the bit stream is encoded, the classifier determines the class of the code among the predetermined set of alternatives, otherwise it is discarded. The bit stream is decoded by the corresponding decoder with respect to the parameters determined by the classifier block.

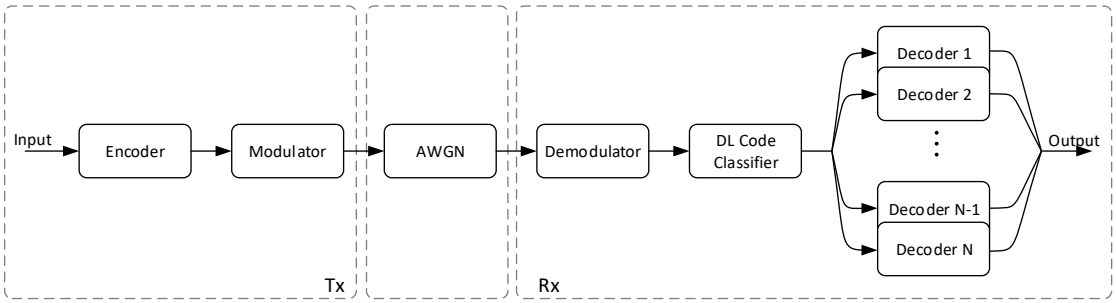


Figure 6.1. Block diagram for blind code detection/decoding. Input is the information vector and the output is the decoded information vector. The DL code classifier block decides the code type and passes the codeword to the corresponding decoder.

Occurrence of a false alarm, which is the decision of the existence of an encoded bit stream whereas the input bit stream is not encoded has different effect from the occurrence of a miss-detection, which corresponds to decision of the uncoded bit stream whereas the input bit stream. The number of false alarms introduced by a blind decoding system can be reduced by addition of a CRC to the encoded data. Recovery of a miss-detection can be accomplished by a different layer by re-transmission of the codeword.

The proposed method utilizes DNNs for blind classification of channel codes. The DNN models are trained with the noisy versions of codewords as described in Chapter 5. Simulation results are provided and discussed by comparing the performance of DL code classifier with existing methods. The performance of the DL-based code classifier

is compared with the SPP-based detection method [10] for convolutional and turbo codes, and the SCL-based detection method [11] for polar codes, respectively. The system model for DL code classifier models for training and test stages is provided in Section 5.2. The training data-set consists of noiseless codeword samples since the noise samples are added in a layer of the DNN model. On the other hand, the test data-set has the noisy codeword samples in order to assess the performance of the classification models.

One example for the use of code identification algorithms is the blind decoding procedure in LTE standard. The receiver block diagram for LTE down-link channels [100, 101] is shown in Figure 6.2. In this process, the channel code parameters are adaptive and configurable. The blind decoding process requires brute-force search of coding parameters by decoding the received block by each decoder in the set and it heavily depends on the CRC checks at the end of the procedure. We can simplify the blind decoding process by adding a blind code detection/classification block before channel decoding step.

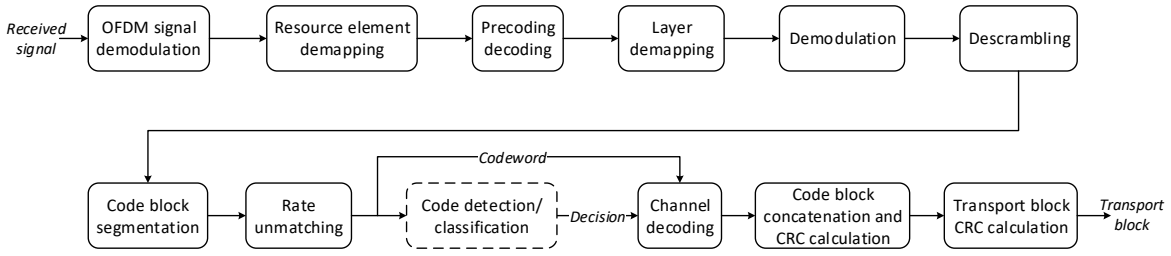


Figure 6.2. The receiver block diagram for LTE down-link channels. The code detection/classification block can be placed before channel decoding in order to perform blind decoding.

Interference in the communications networks could deteriorate the performance of the blind code identification system as well as the channel decoding performance. A survey on the interference mitigation in 3rd Generation Partnership Project (3GPP) LTE networks is provided in [102]. Inter-cell interference is the most common interference type in an LTE network. One of the approaches for reducing the interference

from neighboring cells is to use enhanced frequency reuse techniques.

The signal to interference and noise ratio (SINR) metric is used to measure the overall effect of both interference and noise. The SINR is given by

$$\text{SINR} = \frac{P_{\text{signal}}}{P_{\text{interference}} + P_{\text{noise}}}. \tag{6.1}$$

and SNR is given by

$$\text{SNR} = \frac{P_{\text{signal}}}{P_{\text{noise}}}. \tag{6.2}$$

In general, the SNR and SINR metrics are used interchangeably and the results in this thesis can be adapted into the interference scenario by substituting SNR by SINR.

6.1. Convolutional Code Detection

Convolutional codes are one of the most common code families since they enable low complexity encoding and decoding algorithms. A convolutional encoder, as the name suggests, encodes the input bit stream by the convolution operation. A $C(n, k, \nu)$ code is a convolutional code with input, output and constraint length k, n, ν, respectively. The detection performance of the system is simulated for $C(3, 1, 9)$ and $C(2, 1, 9)$ convolutional codes. The detection problem is considered as a binary classification scenario where the alternate hypothesis is the reception of a convolutionally encoded bit stream whereas the null hypothesis is given as the reception of a uniform random bit stream.

In order to see the effect of the code rate for our code classification model, convolutional codes with equal constraint lengths are selected. The generator matrices of the codes in octal form are given by

$$G_1(D) = \begin{bmatrix} 557 & 663 & 711 \end{bmatrix} \tag{6.3}$$

and

$$G_2(D) = \begin{bmatrix} 561 & 753 \end{bmatrix}. \tag{6.4}$$

The block diagram for the feed-forward neural network used for convolutional code detection is given in Figure 6.3. The LLR values ℓ_i of the received bits are input to the network, where $i = 1, \ldots, n$. The output $c \in \{0, 1\}$ is the decision output of the network, where $c = 1$ and $c = 0$ correspond to the cases that the received bits are encoded and not encoded, respectively. The DNN models are trained according to the system model in Figure 5.1 and system parameters given in Table 5.1. The sample lengths for $C(2, 1, 9)$ and $C(3, 1, 9)$ codes are taken to be 80 and 120, respectively. The number of training codewords are 2^{18} and 2^{16}, respectively. The feed-forward neural network has three layers with 128, 64 and 32 nodes, respectively. The DNN classification model is trained for 1000 epochs. The number of training codewords, layers, nodes and epochs are determined experimentally by observing the detection accuracy values. The number of training epochs can be arbitrarily increased for obtaining higher detection rates without overfitting since the model sees a distinct noise sample in each epoch. However, the performance gain is negligible after a certain number of epochs.

The performance of the DL-based algorithm is compared to the SPP-based algorithm [10] to see the gain in the detection rate. The SPP-based detection method calculates the SPP value for each parity check of the given code as a detection metric. The parity check matrices used for the SPP-based detection algorithm are given as

$$H_1(D) = \begin{bmatrix} 663 & 557 & 0 \\ 711 & 0 & 557 \end{bmatrix} \tag{6.5}$$

and

$$H_2(D) = \begin{bmatrix} 753 & 561 \end{bmatrix}. \tag{6.6}$$

In the SPP-based method, the average SPP value, denoted as $\Gamma(C)$ is calculated, and

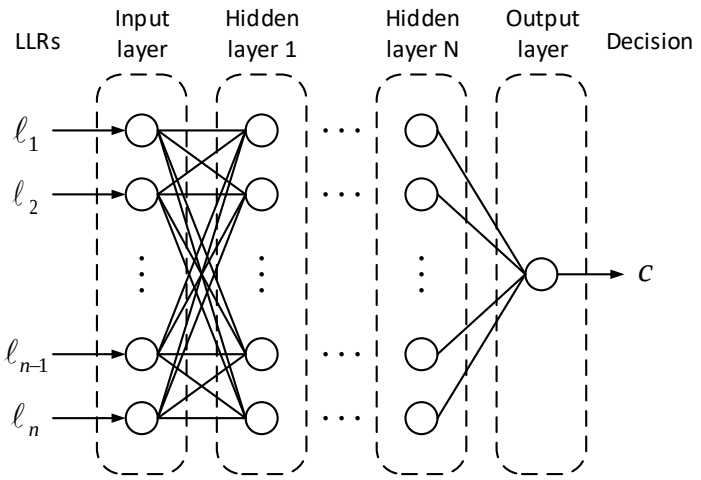


Figure 6.3. The feed-forward neural network model for convolutional code detection. The inputs ℓ_i, $i = 1, \ldots, n$ are the LLR values of the bits. The output $c \in \{0, 1\}$ is the decision whether the input bits are encoded or not.

compared with a threshold for decision

$$\Gamma(C) \underset{H_0}{\overset{H_1}{\gtrless}} \tau_{opt}. \tag{6.7}$$

The probability of false alarm is

$$P_{FA} = \Pr\left(H_1 | H_0\right). \tag{6.8}$$

Accuracy is given by

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}. \tag{6.9}$$

The decision threshold for SPP-based classifier is determined by constant false alarm rate (CFAR) detection, and the false alarm rate is set to 0.01. CFAR detector determines the decision threshold such that the false alarm rate is equal to the target

value. Since there is no closed form equation for the relationship between threshold and false alarm rate, we determined the target thresholds using the bisection method. As seen from Figure 6.4, the performance of the SPP-based method is very poor for both convolutional codes in low SNR regions since the constraint length of the code is relatively long. The DNN model has superior performance even for the codes with a large constraint length. In addition, the performance of both methods are higher for the code with lower rate since it has more parity check relations.

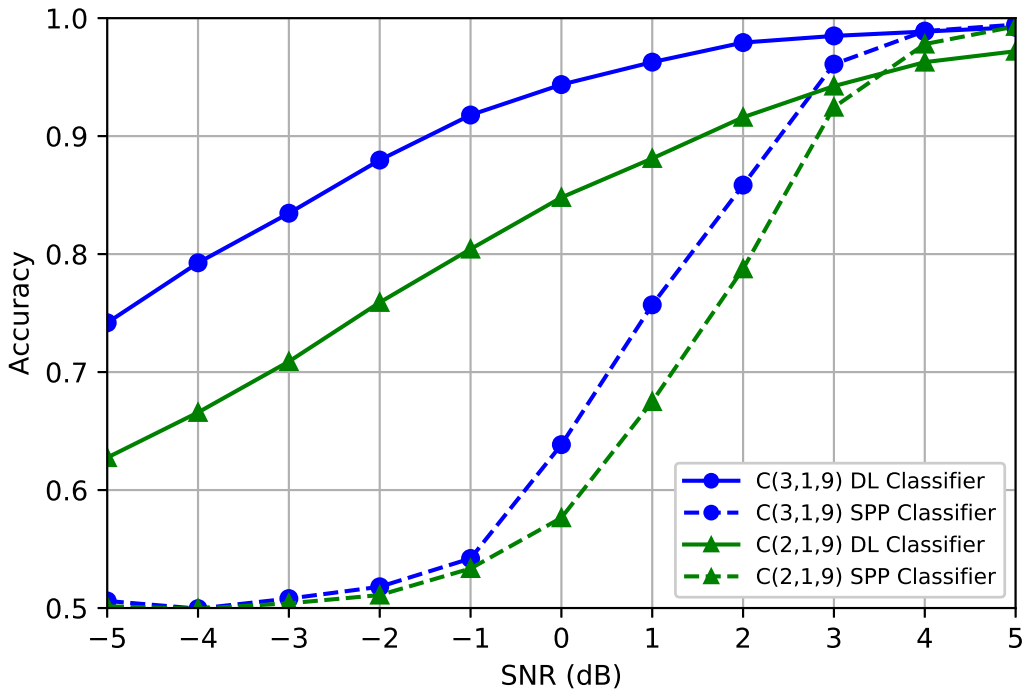


Figure 6.4. Simulation results for convolutional code detection. The number of observed bits are 80 and 120 for codes $C(2,1,9)$ and $C(3,1,9)$, respectively. The number of training codewords are 2^{18} and 2^{16}, respectively.

6.2. Turbo Code Detection

Turbo codes have been the first practical codes to closely achieve the channel capacity. Turbo codes are used for encoding of data channels in 3GPP LTE standard.

LTE turbo encoder is a parallel concatenated convolutional code with a constraint length of 4. It consists of two recursive systematic convolutional encoders and an internal interleaver. The rate of the LTE turbo encoder is 1/3. The transfer function for the constituent codes is given by

$$G(D) = \left[1 \quad \frac{1+D+D^3}{1+D^2+D^3}\right]. \tag{6.10}$$

The LTE turbo code with the internal interleaver length 40 is used for the simulations. Hence, the codeword length is $3 \times (40 + 4) = 132$. The number of training codewords is 2^{16}. The feed-forward neural network has three layers with 128, 64 and 32 nodes, respectively. The DNN classification model is trained for 1000 epochs. The number of training codewords, layers, nodes and epochs are determined experimentally by observing the detection accuracy values. The simulations are performed for the binary classification scenario in which there are two classes of bit streams: a turbo encoded bit stream and a uniform random bit stream. The performance of the DL-based detection model is compared to the SPP-based detection method proposed in [10]. There is no parity check relationship between the outputs of the two constituent codes since the internal interleaver breaks the linear dependency which constitutes a major deficiency for SPP-based method. Thus, the average SPP value is calculated only for the output of the first constituent code. The parity check matrix used for the SPP-based detection algorithm is given as

$$H_{turbo}(D) = \left[1 + D + D^3 \quad 1 + D^2 + D^3\right]. \tag{6.11}$$

The decision threshold for SPP-based detection is set by CFAR detection, and the false alarm rate is selected as 0.01. As clearly seen from Figure 6.5, the DL-based method offers superior performance compared to the previous method. Main reason is that DL-based methods are able to learn the code structure without depending on any code parameters such as density of the parity checks.

In the technical specification document for 3GPP LTE standard [103], it is stated that the block error rate (BLER) for a transport block should be lower than 0.1. The turbo code with the given parameters achieves the mentioned BLER around -3 dB SNR [104]. The SNR values are chosen to be in the -5 dB to 5 dB interval for the simulations. The code detection accuracy for the proposed DL-based method is higher than 90% for SNR values higher than -2 dB. Hence, the DL-based approach is able to satisfy the specifications of the standard.

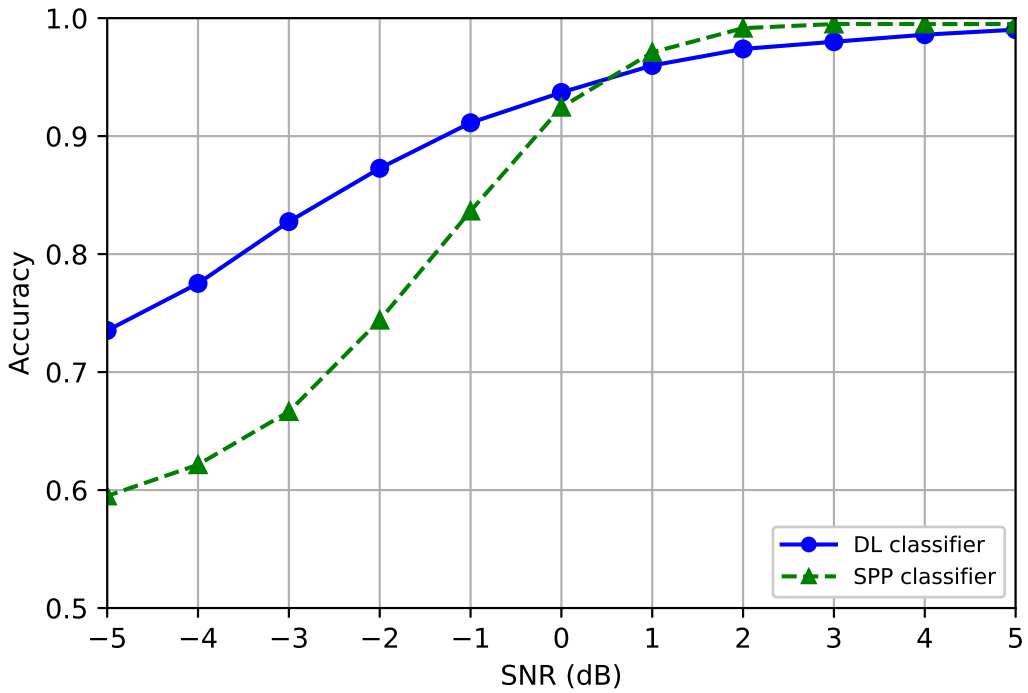


Figure 6.5. Simulation results for turbo code detection. The neural network has three layers with 128, 64, 32 nodes, respectively. The number of observed bits is 132 (one codeword), the number of training codewords is 2^{16} and the number of epochs is 1000.

6.3. Polar Code Classification

In this section, the use of DNNs for polar code classification is considered. Polar codes are a recently emerging code family that is capable of achieving the channel

capacity for discrete memoryless channel, introduced by Arıkan [18]. Polar codes are selected as the coding scheme for the control channels in 3GPP 5G NR standard. Polar codes utilize channel polarization, which corresponds to synthesis of new channels from n independent binary symmetric channels so that the capacities of a portion of the channels converge to 1 while the capacities of the others converge to 0. This phenomenon suggests a novel adaptive approach for channel coding: selecting the most reliable k bits for transmission of information bits and sending a predetermined bit value using the others. Employment of DL based methods instead of the decoding based classifiers can provide benefits for both performance and delay of blind decoding systems.

Polar code classification is performed among four different classes. These classes consist of three different polar codes with different lengths and a random bit stream. The rates of the codes are taken to be constant and equal to 1/8, which is an available code rate in NR standard. The used codes are $P(128, 16)$, $P(64, 8)$, $P(32, 4)$ and the random bit stream is a vector of uniform random variables of length 128. The input length of the classifier is equal to the maximum code length in the code set, which is 128 in this case. In order to have a fair classification, samples from each class have the same length. Therefore, the samples for shorter codes are composed of multiple codewords. For a $P(n, k)$ code, there are 2^k distinct codewords of length n. In this section, the neural network classifier is trained with all possible codewords in order for the network to learn the code structure. The effect of the number of training codewords is discussed in Section 6.4. The number of all possible codewords is 2^{16}.

The block diagram for the feed-forward neural network used for polar code detection is given in Figure 6.6. The LLR values ℓ_i of the received bits are input to the network, where $i = 1, \ldots, n$. The outputs of the neural network are the probabilities c_j for each class, where $j = 1, \ldots, m$ and m is the number of code classes in the set. The neural network has 5 hidden layers with 512, 256, 128, 64, 16 nodes, respectively. The neural network model is trained for 500 epochs. The number of epochs can be further increased without overfitting the training data-set but does not add any performance gain beyond this value.

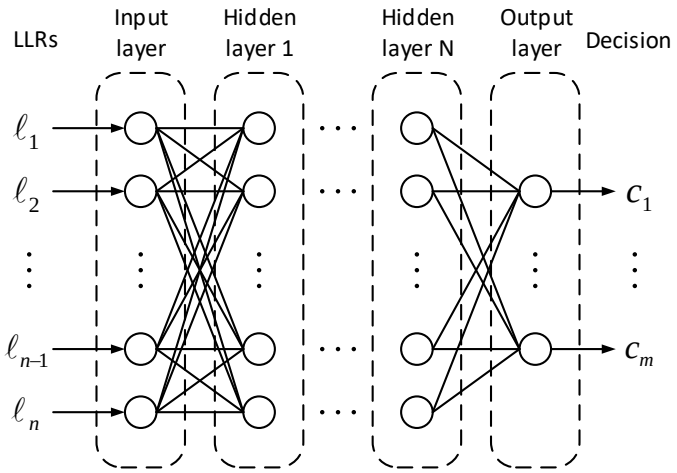


Figure 6.6. The feed-forward neural network model for polar code classification. The input ℓ_i, $i = 1, \ldots, n$ is the input LLR values output from the BPSK demodulator. The outputs c_j, $j = 1, m$ are the decision probabilities for m classes.

The performance of the DL-based classifier is compared to the existing SCL-based classifier. The SCL classifier is based on the SCL detection method, which utilizes the path metric calculated by each SCL decoder for the codes in the set [11]. Arıkan proposes a recursive decoding scheme for polar codes, called SC decoding. The DDG is traversed and decision LLRs are produced for estimation of the bits in this decoding scheme. SC decoding is sub-optimal for short code lengths, and it suffers from error propagation due to its sequential nature. Instead of making decisions at each decoding stage, SCL decoding considers up to L different candidate paths. The number of decoding paths increases exponentially in each stage since both cases are considered for each bit value. In order to limit the complexity, the most reliable L paths are selected via a path metric at each stage.

The classification accuracy of the DL polar code classifier is shown in Figure 6.7. For the low SNR values, the DNN model is not able to learn the code structure due to the dominant noise. As a result, the performance of the SCL-based classifier is better in the low SNR regions since it has a prior knowledge about the code structure. On

the other hand, the performance of the DL-classifier is superior for -4 dB and higher SNR values.

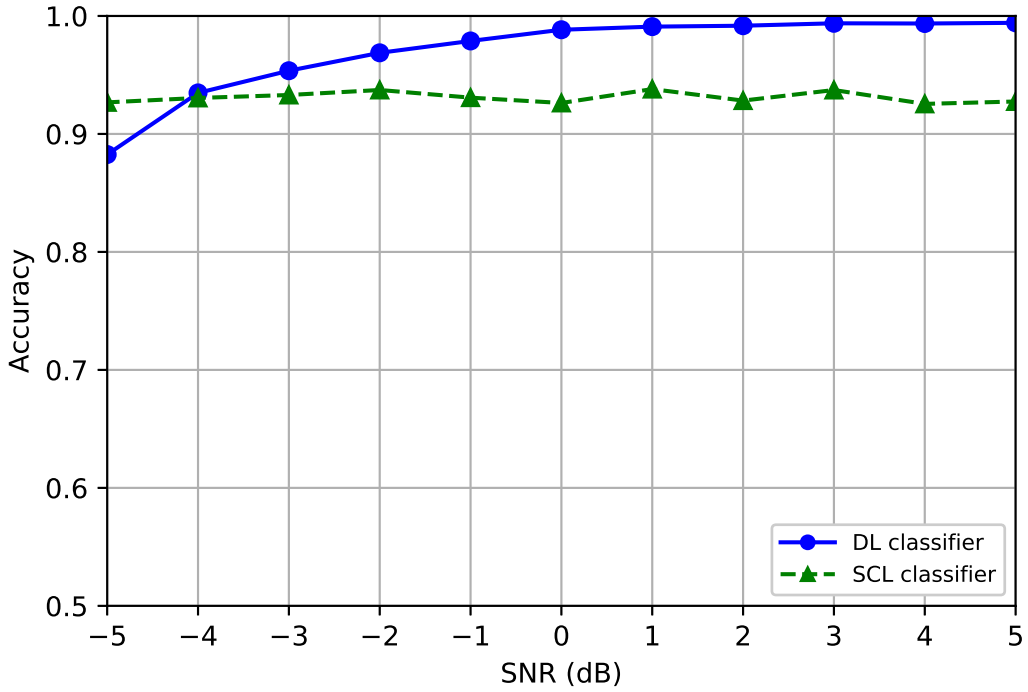


Figure 6.7. Simulation results for polar code classification. There are four classes which can be listed as $P(128, 16)$, $P(64, 8)$, $P(32, 4)$ and random bit stream. The number of observed bits is 128, the number of training codewords is 2^{16}.

6.4. Effect of the Number of Training Codewords

In this section, the capabilities of the neural network in generalizing the code structure is assessed by showing the network only a part of all possible codewords in training phase. The capability of generalizing through the codewords becomes very important for the classification of longer codes since it may not be feasible to train the DL-based classifiers with all possible codewords. On the other hand, a small subset of all possible codewords of a code may represent a different code of a lower dimension. Accordingly, if the network does not see a sufficient number of codeword combinations

in training, there is a possibility that the model learns a different code structure. In order to assess the effect of the number of training samples, the neural network classifier is trained by showing only a portion of all codewords. The classification accuracy of the DL-based classifier trained with a portion of the codewords are given in Figure 6.8. In each simulation, a portion of the codewords are selected randomly. In Figure 6.8, $\gamma\%$ means that $\gamma\%$ of all possible 2^{16} codewords are used for training.

For smaller training sets (lower than 5%), the accuracy values have high variance among different selection of codewords, so the simulations are repeated for different selection of training sets and the results are averaged. As clearly seen from the Figure 6.8, while it is able to generalize even with 10% of the codewords, the classification performance drops drastically for less than 5% of the codewords. Hence, it is possible to train a DNN code classification model by using only 10% of all codewords in expense of a negligible performance drop, which results in a great reduction in computational complexity.

6.5. SNR Robustness

In this section, SNR robustness of the neural network code classifier is considered. Classification accuracy of the model is expected to be the highest when training and test SNRs are matched and decrease when the test SNR deviates from the training SNR. The classifier model can be trained for each SNR value in the interval for obtaining best results. However, the inaccuracies of the SNR estimators and channel fading effects introduce deviations in the estimated SNR. Hence, the classifier models should be robust to changes in SNR. In order to assess the SNR robustness of the proposed method, for each training SNR, the model is tested for varying SNR values. In addition to training at a fixed SNR value, the model is also trained by randomly selecting SNR values from [-5, 5] dB interval for each codeword in each epoch. The performance of the code classifier is shown in Figure 6.9. Each curve represents the test results for a model trained for a fixed SNR except the last curve, which shows the results when the model is trained for all SNR values. The simulation results show that the classifier is robust to changes in SNR. The model trained at -4 dB has 80% accuracy at -6 dB and

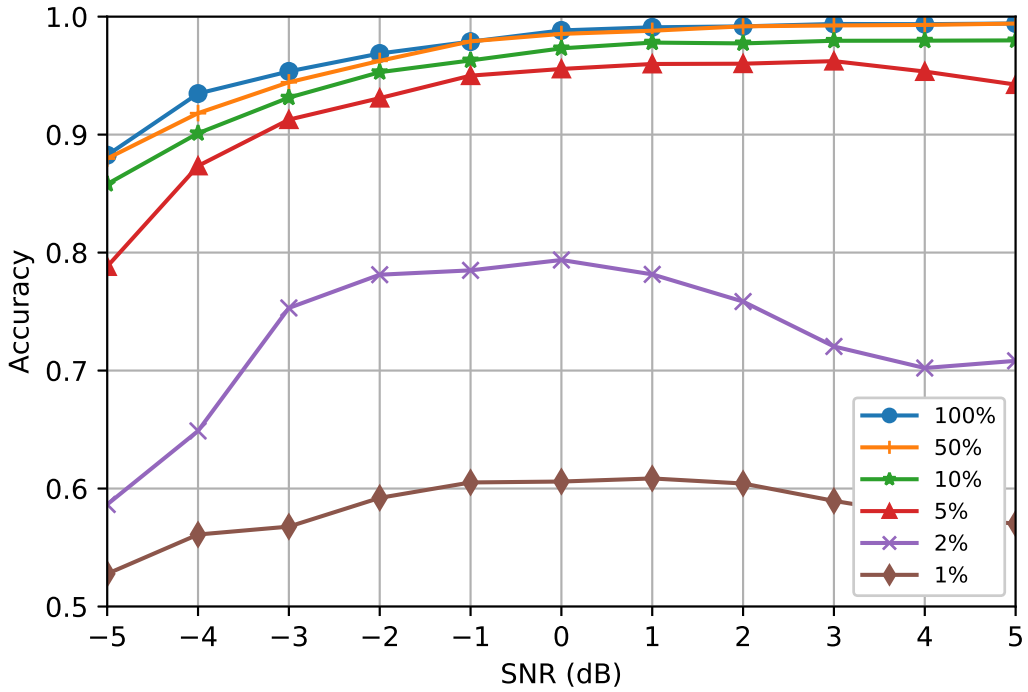


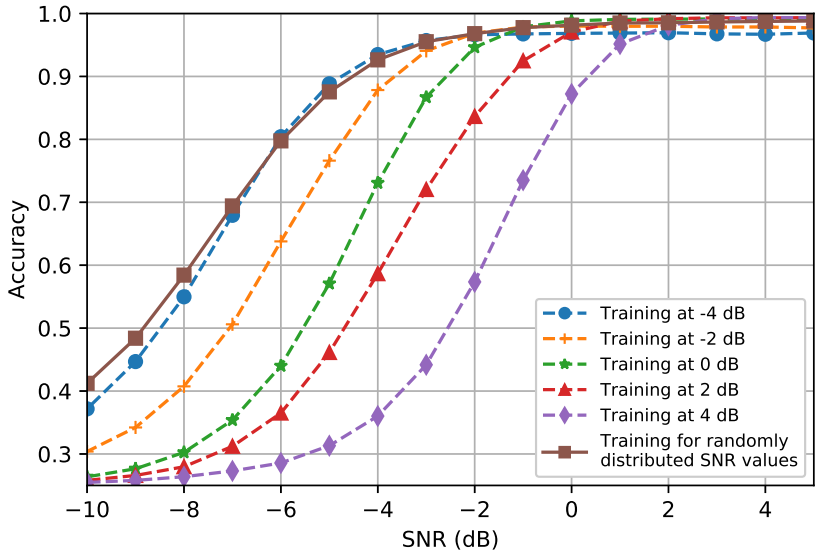
Figure 6.8. Simulation results for polar code classification when the classifier is trained with a portion of all possible codewords. $\gamma\%$ means that $\gamma\%$ of all possible 2^{16} codewords are used for training.

has accuracy higher than 90% for higher SNR values. It also states that it is possible to train a single model at -4 dB for operation in the given SNR interval in expense of a negligible loss in the accuracy. It is also possible to train the model by using all SNR values in the interval by randomly varying the SNR value for each codeword in each epoch. Using all SNR values for the training set increases the SNR robustness of the classifier. However, the addition of codeword samples with lower SNR values deteriorates the performance of the classifier in the high SNR region. Hence, in order to increase the SNR robustness of the classifier, it is advised to use all SNR values in the interval.

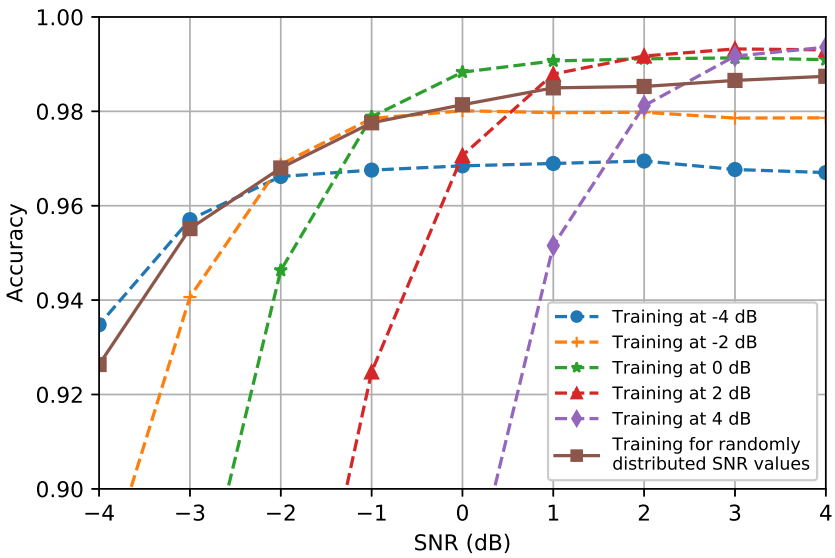
6.6. Effect of False Alarm Rate

In the previous sections, accuracy is used as a performance metric. In this section, in order to show the effect of the false alarm rate to the detection rate of the classifiers, detection rates of the DL-based classifier and the SPP-based classifier compared for different false alarm rates. The detection rates and the false alarm rates of the DL-based classifier for each SNR is given in Figure 6.10. The DL-based classifier is tested at the same SNR as the training SNR. The detection rates of the SPP-based classifier for false alarm rates 0.01, 0.1, and 0.3 are also plotted as a benchmark. As seen from the figure, the false alarm rate of the DL-base classifier changes from 0.36 to 0.01 for SNR values from -5 dB to 5 dB, respectively. For each SNR value, the detection rate of the DL-based classifier is higher than the SPP-based classifier for the same false alarm rate.

The Receiver Operating Characteristic (ROC) curves for both methods is provided in Figure 6.11. The figure shows the ROC curves for -4 dB, 0 dB, and 4 dB SNR values. The simulation results show that the DL-based classifier provides higher detection rates for a fixed false alarm rate for -4 dB and 0 dB SNR values, whereas the SPP-based classifier performs slightly better for 4 dB SNR. However, the performance difference is negligible at 4 dB.



(a)



(b)

Figure 6.9. (a) Simulation results for different training SNR values except the last curve, which represents the results when the model is trained for all SNR values in the interval [-5,5] dB. (b) The zoomed version for the higher SNR region.

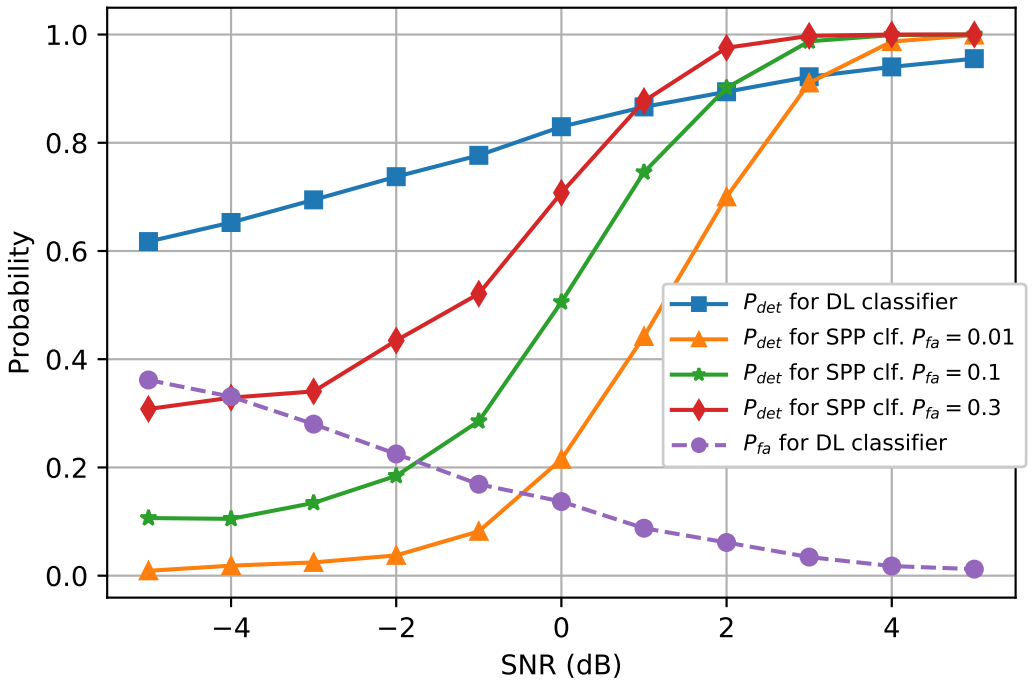


Figure 6.10. Comparison of the convolutional code detection probabilities for $C(2, 1, 9)$ for the DL-based classifier and the SPP-based classifier with different false alarm probabilities.

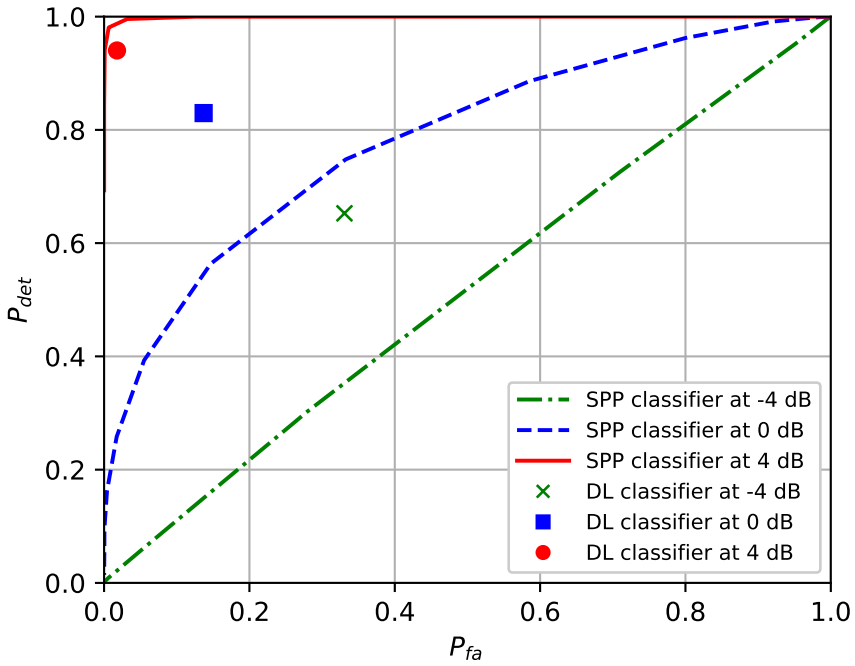


Figure 6.11. Comparison of the receiver operation characteristics for the DL-based classifier and the SPP-based classifier for convolutional code detection for $C(2, 1, 9)$.

The 3σ confidence interval of the DL-based classifier trained with all SNR values is shown in Figure 6.12, where σ denotes the standard deviation of the accuracy values. The simulations are performed for 100 test batches with 1000 codeword samples in each batch. The 3σ interval contains the 99.7% of the values of a normal distribution [105]. As seen from the figure, the classifier provides reliable accuracy results for changing test data.

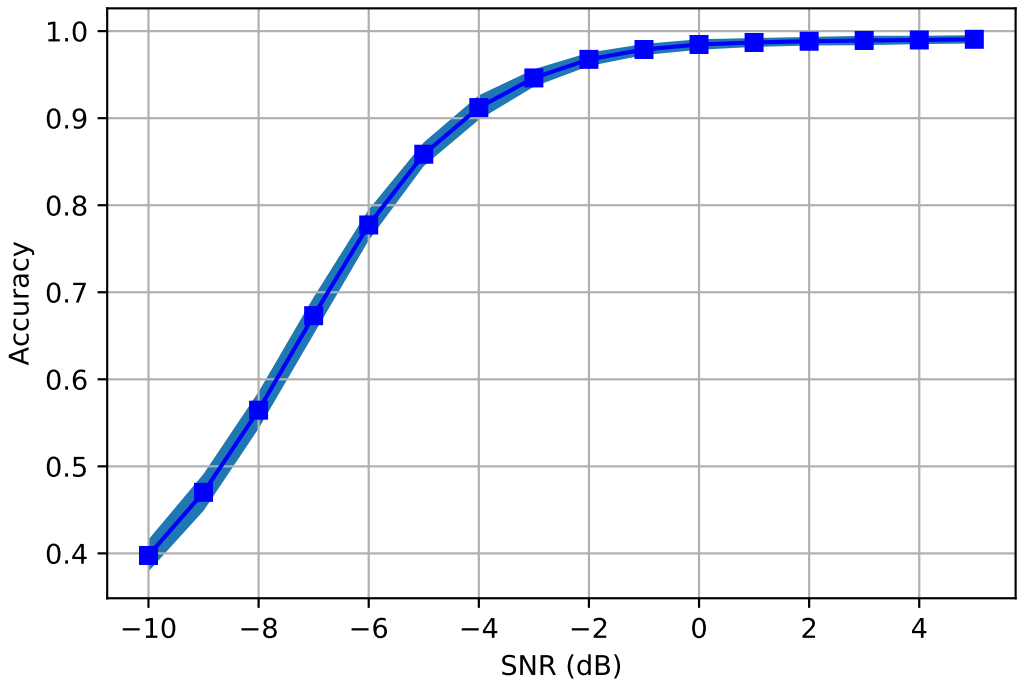


Figure 6.12. Classification accuracy for polar code classification with the 99.7%, i.e., 3σ confidence interval shown. The solid line represents the mean accuracy and the shaded area shows the confidence interval.

The number of hidden layers for the neural network model is determined by a grid search by selecting the parameters that provides the highest accuracy values. The classification accuracy for different number of hidden layers is shown in Figure 6.13 when the DL-based classifier is trained for all SNR values in the interval. The simulations are performed by fixing all network parameters except the number of layers and the number of nodes in each layer. The number of nodes for the neural networks with

3, 5, 7, 10 layers are (512, 256, 128), (512, 256, 128, 64, 32), (512, 256, 128, 128, 64, 64, 32), (512, 256, 256, 128, 128, 64, 64, 32, 32, 16), respectively. The performance of each model are close to each other. As seen from Figure 6.13, the performance decreases slightly from the shallowest network to the deepest network due to the fact that a deeper neural network needs more training epochs since it has more trainable parameters. Taking both the training duration and the accuracy into consideration, it is advisable to choose the neural network with 3 layers.

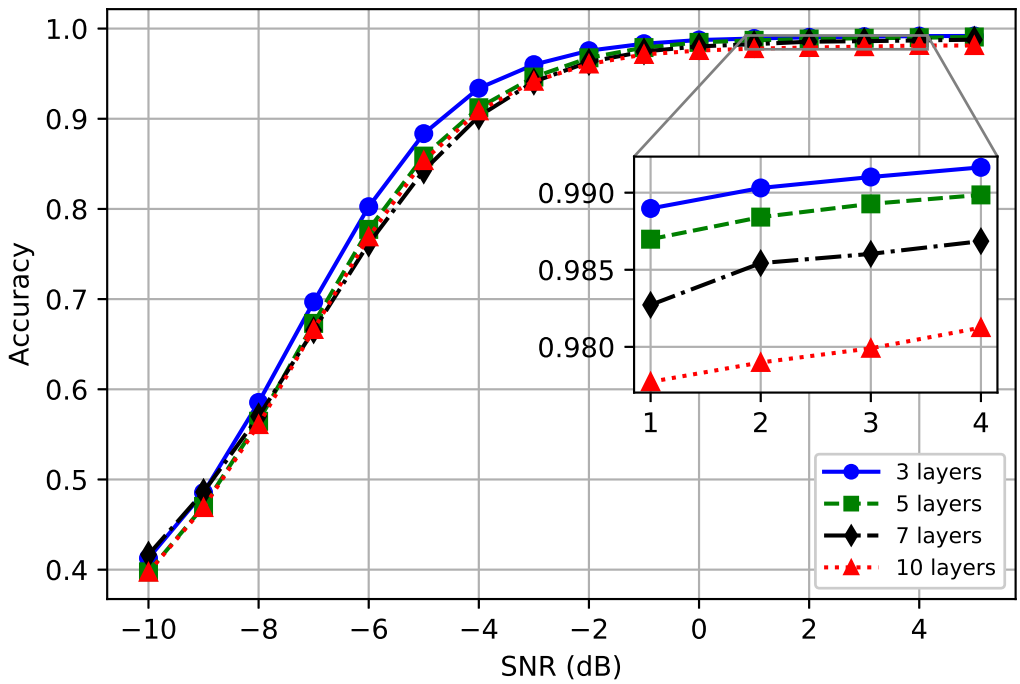


Figure 6.13. Comparison of the accuracy values for different number of hidden layers.

6.7. Computational Complexity

The computational complexity of a feed-forward neural network in the prediction stage is lower than in the training stage since it performs only forward propagation. The training stage consists of two stages: forward and back propagation. In the training stage, the computational complexity of the feed-forward neural network depends

on the parameters such as the number of nodes in a layer, the number of layers, the number of the samples in the data-set, the input length and the number of epochs. The prediction stage performs only a forward propagation. If there is a single input sample, the complexity of the feed-forward neural network depends only on the input length and the number of nodes in the layers. Each layer in the network performs a linear transformation followed by a nonlinear activation function. The linear transformation represented by the multiplication of a matrix with a vector has the computational complexity $O(nm)$, where n denotes the input length and m is the number of nodes in the following hidden layer. The activation function is applied element-wise, which has the complexity $O(m)$. Since the number of nodes in the following layers are generally chosen to be less than the previous layers, the terms coming from the following layers are omitted. Therefore, the worst case complexity of the feed-forward neural network in the prediction stage is $O(nm + m) = O(nm)$. The worst case complexity of the polar SCL decoder in terms of the input length n and the list size L of the decoder is $O(Ln \log n)$ [20]. Hence, the neural network code classifier have a comparable asymptotic computational complexity to the SCL decoder for polar codes. There are also parallel implementations of the neural network models, which further decrease the run-time of the algorithms. However, the sequential nature of the polar SCL decoder does not offer a parallel implementation.

7. CONCLUSION

In this thesis, the utilization of DL models for blind code identification is considered. This approach eliminates the dependence on code parameters due to learning capability of DNNs, and applicable to all code types. The performance of the proposed universal detection method is assessed for widely used convolutional, turbo and polar codes. The presented method enables the parallel implementation since it is based on a DNN, which can meet the low latency requirements of communication networks. A feed-forward neural network architecture is used for classification tasks. The simulation results show that even a simple DNN model offers better accuracy compared to the existing methods. In order to prevent the DNN models from overfitting the training data-set, a distinct noise sample is added in each epoch. Additionally, an analysis on the effect of the training data-set size to the performance is also provided. The simulation results show that the DL model is able to generalize when not all of the codewords are shown in the training stage. The performance of the classifier is satisfactory even when only 10% of the codewords are used for training. The SNR robustness of the method is assessed by training and testing the model for varying SNR values. The simulation results show that the classifier is robust to slight changes in SNR. The model trained at -4 dB has 80% accuracy at -6 dB and has accuracy higher than 90% for higher SNR values. Therefore, it is possible to use a model trained at a single SNR for classification of a wider range of SNR values while the decrease in the classification performance is not significant. In addition to using a single training SNR, the DL classifier can be trained for all SNR values in the interval. However, addition of codeword samples with lower SNR values deteriorates the performance for higher test SNRs. All in all, the proposed approach is envisioned to be a good replacement for the code identification systems.

In future research, we will investigate the performance of the DNN code classifier model which can be improved by employing different DNN architectures for different channel code types. On the other hand, more sophisticated hybrid models can be developed with the aid of code structure for decreasing the number of codeword samples

necessary for training. The performance of the DL code classifier can be verified under correlated noise and fading channel conditions.

REFERENCES

1. Shannon, C. E., "A mathematical theory of communication", *Bell System Technical Journal*, Vol. 27, No. 3, pp. 379–423, 1948.

2. Hui, D., S. Sandberg, Y. Blankenship, M. Andersson and L. Grosjean, "Channel coding in 5G New Radio: a tutorial overview and performance comparison with 4G LTE", *IEEE Vehicular Technology Magazine*, Vol. 13, No. 4, pp. 60–69, Dec 2018.

3. Valembois, A., "Detection and recognition of a binary linear code", *Discrete Applied Mathematics*, Vol. 111, No. 1-2, pp. 199–218, 2001.

4. Balatsoukas-Stimming, A. and A. Filos-Ratsikas, "On the computational complexity of blind detection of binary linear codes", *arXiv preprint arXiv:1806.01050*, 2018.

5. Wang, T., C.-K. Wen, H. Wang, F. Gao, T. Jiang and S. Jin, "Deep learning for wireless physical layer: Opportunities and challenges", *China Communications*, Vol. 14, No. 11, pp. 92–111, 2017.

6. Qin, Z., H. Ye, G. Y. Li and B.-H. F. Juang, "Deep learning in physical layer communications", *IEEE Wireless Communications*, Vol. 26, No. 2, pp. 93–99, 2019.

7. LeCun, Y., Y. Bengio and G. Hinton, "Deep learning", *Nature*, Vol. 521, No. 7553, p. 436, 2015.

8. Goodfellow, I., Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2016.

9. O'Shea, T. and J. Hoydis, "An introduction to deep learning for the physical layer", *IEEE Transactions on Cognitive Communications and Networking*, Vol. 3,

No. 4, pp. 563–575, 2017.

10. Moosavi, R. and E. G. Larsson, "A fast scheme for blind identification of channel codes", *IEEE Global Communications Conference*, pp. 1–5, 2011.

11. Condo, C., S. A. Hashemi and W. J. Gross, "Blind detection with polar codes", *IEEE Communications Letters*, Vol. 21, No. 12, pp. 2550–2553, 2017.

12. Dingel, J. and J. Hagenauer, "Parameter estimation of a convolutional encoder from noisy observations", *IEEE International Symposium on Information Theory*, pp. 1776–1780, 2007.

13. Soteh, A. G. and H. K. Bizaki, "On the analytical solution of rank problem in the convolutional code identification context", *IEEE Communications Letters*, Vol. 20, No. 3, pp. 442–445, 2016.

14. Gruber, T., S. Cammerer, J. Hoydis and S. ten Brink, "On deep learning-based channel decoding", *51st Annual Conference on Information Sciences and Systems*, pp. 1–6, 2017.

15. Lin, S. and D. J. Costello, *Error Control Coding, Second Edition*, Prentice-Hall, Inc., 2004.

16. Johannesson, R. and K. S. Zigangirov, *Fundamentals of Convolutional Coding*, Vol. 15, John Wiley & Sons, 2015.

17. Moon, T. K., *Error Correction Coding: Mathematical Methods and Algorithms*, John Wiley & Sons, 2005.

18. Arikan, E., "Channel polarization: A method for constructing capacity-achieving codes", *IEEE International Symposium on Information Theory*, pp. 1173–1177, 2008.

19. Niu, K., K. Chen, J. Lin and Q. Zhang, "Polar codes: Primary concepts and

practical decoding algorithms", *IEEE Communications Magazine*, Vol. 52, No. 7, pp. 192–203, 2014.

20. Tal, I. and A. Vardy, "List decoding of polar codes", *IEEE International Symposium on Information Theory*, pp. 1–5, 2011.

21. Niu, K. and K. Chen, "CRC-aided decoding of polar codes", *IEEE Communications Letters*, Vol. 16, No. 10, pp. 1668–1671, 2012.

22. Arikan, E., "A performance comparison of polar codes and Reed-Muller codes", *IEEE Communications Letters*, Vol. 12, No. 6, pp. 447–449, 2008.

23. Hussami, N., S. B. Korada and R. Urbanke, "Performance of polar codes for channel and source coding", *IEEE International Symposium on Information Theory*, pp. 1488–1492, 2009.

24. Minsky, M. and S. A. Papert, *Perceptrons: An Introduction to Computational Geometry*, MIT Press, 2017.

25. Hornik, K., "Approximation capabilities of multilayer feedforward networks", *Neural Networks*, Vol. 4, No. 2, pp. 251–257, 1991.

26. Wang, X.-A. and S. B. Wicker, "An artificial neural net Viterbi decoder", *IEEE Transactions on Communications*, Vol. 44, No. 2, pp. 165–171, 1996.

27. Nachmani, E., Y. Be'ery and D. Burshtein, "Learning to decode linear codes using deep learning", *54th Annual Allerton Conference on Communication, Control, and Computing*, pp. 341–346, 2016.

28. Nachmani, E., E. Marciano, D. Burshtein and Y. Be'ery, "RNN decoding of linear block codes", *arXiv preprint arXiv:1702.07560*, 2017.

29. Lugosch, L. and W. J. Gross, "Neural offset min-sum decoding", *IEEE International Symposium on Information Theory*, pp. 1361–1365, 2017.

30. Nachmani, E., E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein and Y. Be'ery, "Deep learning methods for improved decoding of linear codes", *IEEE Journal of Selected Topics in Signal Processing*, Vol. 12, No. 1, pp. 119–131, 2018.

31. Xu, W., Z. Wu, Y.-L. Ueng, X. You and C. Zhang, "Improved polar decoder based on deep learning", *IEEE International Workshop on Signal Processing Systems*, pp. 1–6, 2017.

32. Lugosch, L. and W. J. Gross, "Learning from the syndrome", *arXiv preprint arXiv:1810.10902*, 2018.

33. Bennatan, A., Y. Choukroun and P. Kisilev, "Deep learning for decoding of linear codes-a syndrome-based approach", *IEEE International Symposium on Information Theory*, pp. 1595–1599, 2018.

34. Lyu, W., Z. Zhang, C. Jiao, K. Qin and H. Zhang, "Performance evaluation of channel decoding with deep neural networks", *IEEE International Conference on Communications*, pp. 1–6, 2018.

35. Cammerer, S., T. Gruber, J. Hoydis and S. ten Brink, "Scaling deep learning-based decoding of polar codes via partitioning", *IEEE Global Communications Conference*, pp. 1–6, 2017.

36. Doan, N., S. A. Hashemi and W. J. Gross, "Neural successive cancellation decoding of polar codes", *IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications*, pp. 1–5, 2018.

37. Wang, Y., Z. Zhang, S. Zhang, S. Cao and S. Xu, "A unified deep learning based polar-LDPC decoder for 5G communication systems", *10th International Conference on Wireless Communications and Signal Processing*, pp. 1–6, 2018.

38. Kim, H., Y. Jiang, S. Kannan, S. Oh and P. Viswanath, "Deepcode: Feedback codes via deep learning", *Advances in Neural Information Processing Systems*,

pp. 9436–9446, 2018.

39. Cluzeau, M., "Block code reconstruction using iterative decoding techniques", *IEEE International Symposium on Information Theory*, pp. 2269–2273, 2006.

40. Cluzeau, M. and J.-P. Tillich, "On the code reverse engineering problem", *IEEE International Symposium on Information Theory*, pp. 634–638, 2008.

41. Burel, G. and R. Gautier, "Blind estimation of encoder and interleaver characteristics in a non cooperative context", *International Conference on Communications, Internet and Information Technology*, pp. 275–280, 2003.

42. Barbier, J., G. Sicot and S. Houcke, "Algebraic approach for the reconstruction of linear and convolutional error correcting codes", *International Journal of Applied Mathematics and Computer Sciences*, Vol. 2, No. 3, 2006.

43. Chabot, C., "Recognition of a code in a noisy environment", *IEEE International Symposium on Information Theory*, pp. 2211–2215, 2007.

44. Sicot, G., S. Houcke and J. Barbier, "Blind detection of interleaver parameters", *Signal Processing*, Vol. 89, No. 4, pp. 450–462, 2009.

45. Cluzeau, M. and M. Finiasz, "Recovering a code's length and synchronization from a noisy intercepted bitstream", *IEEE International Symposium on Information Theory*, pp. 2737–2741, 2009.

46. Choi, C. and D. Yoon, "Enhanced blind interleaver parameters estimation algorithm for noisy environment", *IEEE Access*, Vol. 6, pp. 5910–5915, 2018.

47. Chose, P., A. Joux and M. Mitton, "Fast correlation attacks: An algorithmic point of view", *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 209–221, 2002.

48. Canteaut, A. and F. Chabaud, "A new algorithm for finding minimum-weight

words in a linear code: application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511", *IEEE Transactions on Information Theory*, Vol. 44, No. 1, pp. 367–378, 1998.

49. Zrelli, Y., R. Gautier, E. Rannou, M. Marazin and E. Radoi, "Blind identification of code word length for non-binary error-correcting codes in noisy transmission", *EURASIP Journal on Wireless Communications and Networking*, Vol. 2015, No. 1, p. 43, 2015.

50. Barbier, J. and J. Letessier, "Forward error correcting codes characterization based on rank properties", *International Conference on Wireless Communications & Signal Processing*, pp. 1–5, 2009.

51. Carrier, K. and J.-P. Tillich, "Identifying an unknown code by partial Gaussian elimination", *Designs, Codes and Cryptography*, Vol. 87, No. 2-3, pp. 685–713, 2019.

52. Moosavi, R. and E. G. Larsson, "Fast blind recognition of channel codes", *IEEE Transactions on Communications*, Vol. 62, No. 5, pp. 1393–1405, 2014.

53. Yu, P., H. Peng and J. Li, "On blind recognition of channel codes within a candidate set", *IEEE Communications Letters*, Vol. 20, No. 4, pp. 736–739, 2016.

54. Xia, T. and H.-C. Wu, "Novel blind identification of LDPC codes using average LLR of syndrome a posteriori probability", *IEEE Transactions on Signal Processing*, Vol. 62, No. 3, pp. 632–640, 2014.

55. Xia, T. and H.-C. Wu, "Blind identification of nonbinary LDPC codes using average LLR of syndrome a posteriori probability", *IEEE Communications Letters*, Vol. 17, No. 7, pp. 1301–1304, 2013.

56. Yardi, A. D. and S. Vijayakumaran, "Detecting linear block codes in noise using the GLRT", *IEEE International Conference on Communications*, pp. 4895–4899,

2013.

57. Vijayakumaran, S., "Identifying block codes using groebner bases", *IEEE International Conference on Communications*, pp. 4424–4430, 2015.

58. Bonvard, A., S. Houcke, M. Marazin and R. Gautier, "Order statistics on minimal Euclidean distance for blind linear block code identification", *IEEE International Conference on Communications*, pp. 1–5, 2018.

59. Bonvard, A., S. Houcke, R. Gautier and M. Marazin, "Classification based on Euclidean distance distribution for blind identification of error correcting codes in non-cooperative contexts", *IEEE Transactions on Signal Processing*, 2018.

60. Liu, Y., F. Wang, J. Zhang, B. Ai and Z. Zhong, "Blind identification of LDPC codes in multipath fading channel via expectation maximization", *IEEE Global Communications Conference*, pp. 1–6, 2018.

61. Filiol, E., "Reconstruction of convolutional encoders over GF (q)", *IMA International Conference on Cryptography and Coding*, pp. 101–109, 1997.

62. Wang, F., Z. Huang and Y. Zhou, "A method for blind recognition of convolution code based on euclidean algorithm", *International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1414–1417, 2007.

63. Xie, H., X.-m. Chai, F.-h. Wang and Z.-t. Huang, "A method for blind identification of rate 1/2 convolutional code based on improved Euclidean algorithm", *IEEE 11th International Conference on Signal Processing*, Vol. 2, pp. 1307–1310, 2012.

64. Marazin, M., R. Gautier and G. Burel, "Dual code method for blind identification of convolutional encoder for cognitive radio receiver design", *IEEE Global Communications Conference Workshops*, pp. 1–6, 2009.

65. Marazin, M., R. Gautier and G. Burel, "Blind recovery of k/n rate convolutional encoders in a noisy environment", *EURASIP Journal on Wireless Communications and Networking*, Vol. 2011, No. 1, p. 168, 2011.

66. Marazin, M., R. Gautier and G. Burel, "Some interesting dual-code properties of convolutional encoder for standards self-recognition", *IET Communications*, Vol. 6, No. 8, pp. 931–935, 2012.

67. Zrelli, Y., M. Marazin, E. Rannou and R. Gautier, "Blind identification of convolutional encoder parameters over GF (2m) in the noiseless case", *20th International Conference on Computer Communications and Networks*, pp. 1–5, 2011.

68. Jing, Z., H. Zhiping, S. Shaojing and Z. Yimeng, "Blind identification of convolutional codes in soft-decision situations", *International Journal of Modern Communication Technologies and Research*, Vol. 2, No. 4, 2014.

69. Zrelli, Y., R. Gautier, M. Marazin, E. Rannou and E. Radoi, "Focus on theoretical properties of blind convolutional codes identification methods based on rank criterion", *International Conference on Communications*, pp. 353–356, 2012.

70. Tixier, A., "Blind identification of an unknown interleaved convolutional code", *IEEE International Symposium on Information Theory*, pp. 71–75, 2015.

71. Bellard, M. and J.-P. Tillich, "Detecting and reconstructing an unknown convolutional code by counting collisions", *IEEE International Symposium on Information Theory*, pp. 2967–2971, 2014.

72. Liu, J., X.-J. Wang and X.-Y. Zhou, "Blind recognition of convolutional coding based on Walsh-Hadamard transform", *Dianzi Yu Xinxi Xuebao(Journal of Electronics and Information Technology)*, Vol. 32, No. 4, pp. 884–888, 2010.

73. Wang, F., H. Xie and Z. Huang, "Blind reconstruction of convolutional code based on segmented Walsh-Hadamard transform", *Journal of Systems Engineering and*

Electronics, Vol. 25, No. 5, pp. 748–754, 2014.

74. Huang, L., W. Chen, E. Chen and H. Chen, "Blind recognition of k/n rate convolutional encoders from noisy observation", *Journal of Systems Engineering and Electronics*, Vol. 28, No. 2, pp. 235–243, 2017.

75. Lu, P., S. Li, Y. Zou and X. Luo, "Blind recognition of punctured convolutional codes", *Science in China Series F: Information Sciences*, Vol. 48, No. 4, pp. 484–498, 2005.

76. Cluzeau, M. and M. Finiasz, "Reconstruction of punctured convolutional codes", *IEEE Information Theory Workshop*, pp. 75–79, 2009.

77. Côte, M. and N. Sendrier, "Reconstruction of convolutional codes from noisy observation", *IEEE International Symposium on Information Theory*, pp. 546–550, 2009.

78. Marazin, M., R. Gautier and G. Burel, "Algebraic method for blind recovery of punctured convolutional encoders from an erroneous bitstream", *IET Signal Processing*, Vol. 6, No. 2, pp. 122–131, 2012.

79. Barbier, J., "Reconstruction of turbo-code encoders", *Digital Wireless Communications VII and Space Communication Technologies*, Vol. 5819, pp. 463–474, 2005.

80. Marazin, M., R. Gautier and G. Burel, "Blind recovery of the second convolutional encoder of a turbo-code when its systematic outputs are punctured", *Military Technical Academy Review*, Vol. 19, No. 2, pp. 213–232, 2009.

81. Barbier, J. and E. Filiol, "Overview of turbo-code reconstruction techniques.", *IACR Cryptology ePrint Archive*, Vol. 2009, p. 68, 2009.

82. Côte, M. and N. Sendrier, "Reconstruction of a turbo-code interleaver from noisy

observation", *IEEE International Symposium on Information Theory*, pp. 2003–2007, 2010.

83. Cluzeau, M., M. Finiasz and J.-P. Tillich, "Methods for the reconstruction of parallel turbo codes", *IEEE International Symposium on Information Theory*, pp. 2008–2012, 2010.

84. Naseri, A., O. Azmoon and S. Fazeli, "Blind recognition algorithm of turbo codes for communication intelligence systems", *International Journal of Computer Science*, Vol. 8, No. 6, 2011.

85. Teimouri, M. and A. Hedayat, "Parameter estimation of turbo code encoder", *Advances in Electrical Engineering*, Vol. 2014, 2014.

86. Yu, P., J. Li and H. Peng, "A least square method for parameter estimation of RSC sub-codes of turbo codes", *IEEE Communications Letters*, Vol. 18, No. 4, pp. 644–647, 2014.

87. Tillich, J.-P., A. Tixier and N. Sendrier, "Recovering the interleaver of an unknown turbo-code", *IEEE International Symposium on Information Theory*, pp. 2784–2788, 2014.

88. Yu, P., H. Peng and J. Li, "Early stopping for interleaver recovering of turbo codes", *arXiv preprint arXiv:1605.05173*, 2016.

89. Pei, R., Z. Wang, Q. Xiao and L. Quan, "Blind identification for turbo codes in AMC systems", *IEEE International Conference on Communication Software and Networks*, pp. 43–47, 2016.

90. Debessu, Y. G., H.-C. Wu and H. Jiang, "Novel blind encoder parameter estimation for turbo codes", *IEEE Communications Letters*, Vol. 16, No. 12, pp. 1917–1920, 2012.

91. Debessu, Y. G., H.-C. Wu, H. Jiang and S. Y. Chang, "Blind encoder parameter estimation for turbo codes", *IEEE Global Communications Conference*, pp. 4233–4237, 2012.

92. Yu, P., J. Li and H. Peng, "Gibbs sampling based parameter estimation for RSC sub-codes of turbo codes", *6th International Conference on Wireless Communications and Signal Processing*, pp. 1–5, 2014.

93. Tao, J., M. Diao and Z. Chen, "An interleaver estimation algorithm for turbo-code based on conformity of parity-check equation", *IEEE Asia-Pacific Conference on Antennas and Propagation*, pp. 336–340, 2018.

94. Condo, C., S. A. Hashemi, A. Ardakani, F. Ercan and W. J. Gross, "Design and implementation of a polar codes blind detection scheme", *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2018.

95. Giard, P., A. Balatsoukas-Stimming and A. Burg, "Blind detection of polar codes", *IEEE International Workshop on Signal Processing Systems*, pp. 1–6, 2017.

96. Ren, Y., F. Shu, L. Li, Z. Zhang, X. You and C. Zhang, "A novel D-metric for blind detection of polar codes", *IEEE International Workshop on Signal Processing Systems*, pp. 106–111, 2018.

97. Sun, H., R. Liu, K. Tian, B. Dai and B. Feng, "A novel blind detection scheme of polar codes", *IEEE Communications Letters*, pp. 1–1, 2019.

98. Giard, P., A. Balatsoukas-Stimming and A. Burg, "On the tradeoff between accuracy and complexity in blind detection of polar codes", *IEEE 10th International Symposium on Turbo Codes & Iterative Information Processing*, pp. 1–5, 2018.

99. Kingma, D. P. and J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

100. 3GPP TS 36.211, *Evolved Universal Terrestrial Radio Access (E-UTRA); Physical Channels and Modulation*, Rel. 15.

101. 3GPP TS 36.212, *Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and Channel Decoding*, Rel. 15.

102. Kwan, R. and C. Leung, "A survey of scheduling and interference mitigation in LTE", *Journal of Electrical and Computer Engineering*, Vol. 2010, 2010.

103. 3GPP TS 36.213, *Evolved Universal Terrestrial Radio Access (E-UTRA); Physical Layer Procedures*, Rel. 15.

104. Shirvanimoghaddam, M., M. S. Mohammadi, R. Abbas, A. Minja, C. Yue, B. Matuz, G. Han, Z. Lin, W. Liu, Y. Li *et al.*, "Short block-length codes for ultra-reliable low latency communications", *IEEE Communications Magazine*, Vol. 57, No. 2, pp. 130–137, 2018.

105. Ramachandran, K. M. and C. P. Tsokos, *Mathematical Statistics with Applications in R*, Elsevier, 2014.