Contents

List of Figures

List of Tables

Contributors

WARREN J. GROSS, Department of Electrical and Computer Engineering, McGill University, Montréal, Québec, Canada

NGHIA DOAN, Department of Electrical and Computer Engineering, McGill University, Montréal, Québec, Canada

ELIE NGOMSEU MAMBOU, Department of Electrical and Computer Engineering, McGill University, Montréal, Québec, Canada

SEYYED ALI HASHEMI, Department of Electrical and Computer Engineering, McGill University, Montréal, Québec, Canada

Chapter 15

Deep Learning Techniques for Decoding Polar Codes

For past decades, the challenge of designing capacity-approaching codes has been one of the main focuses in digital communications. In this regard, polar codes were introduced as the first class of error correcting codes that provably achieve the capacity for any channel at infinite code length. Recently, deep learning (DL) has shown a great potential in a wide range of applications in digital communications including channel coding for forward error correction (FEC) codes. Therefore, we believe that a literature review on the intersection between DL and FEC codes, especially polar codes, can contribute to the coding community. Organized into four sections, this chapter first provides background and motivation for the use of DL in various FEC schemes used for wireless communication systems. The second section of this chapter introduces polar codes and their traditional decoding algorithms. In the third section, three major DL based approaches for decoding polar codes are presented in terms of decoding performance, algorithm complexity and decoding latency. As the last section of this chapter, the fourth section makes further discussions and conclusions.

15.1. Motivation and Background

Deep learning (DL) [LeCun et al., 2015] has been widely used in digital communications through applications such as channel decoding, prediction, equalization, modulation/demodulation, detection, quantization, compression, and spectrum sensing [Ibnkahla, 2000]. In [O'Shea and Hoydis, 2017], a new approach in communication systems based on DL was introduced, where the entire channel transmission was abstracted as an autoencoder; that is an end-to-end pipeline designed by jointly concatenation of transmitters and receivers in a single process. It has been established that the block error rate (BLER) of traditional communication systems can be improved through this scheme. Furthermore, this idea of end-to-end channel modeling was extended to multiple input multiple output (MIMO) systems. MIMO suffers from interference between channels in a way that obtaining the optimal signal is very difficult. It has been shown that such systems can be represented as multiple input and output generative adversarial nets [Goodfellow et al., 2014] which can be optimized using a customized loss function for the joint model. The challenges and opportunities related to DL for wireless physical layer were presented in [Wang et al., 2017].

In [Farsad et al., 2018], a DL-based approach for joint source-channel coding of a text was proposed. This approach is almost similar to the idea presented in [O'Shea and Hoydis, 2017] where the source is jointly trained with the channel to reduce the transmission distortion. This architecture makes use of a long short-term memory (LSTM)-based model as proposed in [Graves and Schmidhuber, 2005, Hasim et al., 2014] over a binary erasure channel (BEC) to ensure a gain in word error rate (WER) when compared to a separate source-channel word processing. However, the fixed-length word processing scheme does not allow flexibility at the incoming source. Similarly, an end-to-end neural network (NN) system was proposed for communication over the air between two software-defined radios (SDRs) in [Dörner et al., 2018].

The concept of end-to-end NN was also extended for MIMO relays in [Sun and Jing, 2012]. In this work, channel training and coherent decoding under estimated channel error were studied for relay networks, and the setup of a single-antenna as transmitter, R distinct antenna relays, and a single R antenna receiver is considered. Note that many coherent cooperative decoding methods assume a perfect and global channel state information (CSI) which is in reality imperfect especially in cooperative relay networks. Two types of approaches were considered: the decoding where CSI is assumed noise-free and the matched decoding where the noise estimation is evaluated. Simulations demonstrate that at least $3R$ symbol intervals of training are needed for mismatched versus $R+2$ for the matched decoding to achieve a full diversity. In addition, an adaptive decoding was presented to compensate for the complexity in the matched decoding. Several works concerning NN applied to MIMO systems can be found in [Wen et al., 2018, O'Shea and Hoydis, 2017, Samuel et al., 2017].

It was established that DL decoding for linear block codes is equivalent to deriving the maximum energy function of a NN [Bruck and Blaum, 1989]. In order to maximize the energy function of a NN, [Bruck and Blaum, 1989] suggested that decoding of FEC codes can be done through maximizing polynomials over the N-cube for a (N, K) block code, where N is the length and K is the dimension of the code. In a similar logic, [Tallini and Cull, 1995] predicted that the NP-complete problem of receiving an error-free message through a channel can be solved through NN. In [Wu et al., 2002], a neural structure was described as a perceptron with a higher order polynomial as a discriminant function. A $(2^m - 1, 2^m - 1 - m)$ Hamming code was decoded through only $m + 1$ assigned weights on each perceptron, m being a positive integer. This architecture combines two layers, the first is made of a set of parity bits and the second is a linear classifier. This proves that high-order codes such as Bose–Chaudhuri–Hocquenghem (BCH) codes can be learned successfully through a multilayer perceptron (MLP).

In channel coding, DL-based decoders can provide improvements in error probabilities over conventional decoders. In [Nachmani et al., 2018], the conventional belief propagation (BP) decoding algorithm is formalized as a partially connected NN. In addition, by assigning trainable weights to the BP-based NN, neural BP decoders can achieve the same performance as the conventional sum-product BP decoding algorithm with a significantly smaller number of iterations. In [Kim et al., 2018], the sequential decoding of turbo and convolutional codes [Berrou et al., 1993, Viterbi, 1971] was performed through recurrent-neural-network (RNN) based models. It has been demonstrated that the trained RNN architecture can decode these codes over additive white Gaussian noise (AWGN) channel with a performance near that of the maximum a posteriori (MAP) decoding given by Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm and the maximum likelihood (ML) decoding given by Viterbi algorithm for sequential codes. This result was confirmed in [Kim et al., 2018] through the same architecture and extended to codes with larger block lengths. In [Bennatan et al., 2018], a syndrome-based DL technique was proposed to decode linear block codes. For a BCH code of length 127, an 11-layer vanilla MLP and a RNN of four stacks were considered separately to estimate the channel noise rather than the transmitted codeword. The syndrome decoding and soft channel reliabilities were helpful to eliminate the problem of overfitting of the training codeword set. The BCH decoding performances were compared under traditional BP decoding, syndrome-based vanilla MLP, and syndrome-based stacked RNN. Results showed that the syndrome-based stacked RNN performance approached that of the ordered statistics decoding (OSD) of order 2 while outperforming that of the syndrome-based vanilla MLP at the cost of high complexity and latency.

In the following section, we will focus on DL techniques used for polar codes as a case study of DL for FEC codes.

15.2. Decoding of Polar Codes: An Overview

Polar codes are a recent breakthrough in the field of channel coding as they were proved to achieve channel capacity with efficient encoding and decoding algorithms [Arıkan, 2009]. Successive cancellation (SC) and BP decoding algorithms are first introduced to decode polar codes [Arıkan, 2009]. Although SC decoding can provide a low-complexity implementation, its serial nature prevents the decoder to reach a high decoding throughput. In addition, the error-correction performance of SC decoding for short to moderate polar codes does not satisfy the requirements of the fifth generation of cellular mobile communications (5G). SC list (SCL) decoding was introduced in [Tal and Vardy, 2015] to improve the performance of SC decoding. SCL can provide a significant improvement in terms of error probability if the decoder is aided by a cyclic redundancy check (CRC). With these appealing properties, polar codes have been selected to be used in the enhanced mobile broadband (eMBB) control channel of 5G together with a CRC [3GPP, 2018].

Recently, it has been shown that polar codes can also be decoded using off-the-shelf DL decoders, which may lead to a high decoding throughput thanks to their one-shot-decoding property [Cammerer et al., 2017]. In addition, it was observed that by assigning trainable weights to the unrolled factor graph of polar codes, neural BP decoders can provide a significant error-correction performance gain over conventional BP decoding [Nachmani et al., 2018]. Other approaches such as using DL models for channel noise estimation have also demonstrated a great potential of DL techniques when applied to well-established problems in channel coding [Bennatan et al., 2018, Liang et al., 2018].

In this section, we first provide some basic knowledge about polar codes and conventional polar decoders. In the next section, several DL-based decoding algorithms and their variants for polar codes are discussed and followed by a detailed evaluation concerning error-correction performance and decoding latency of state-of-the-art DL-aided decoders for a 5G polar code.

15.2.1. Problem Formulation of Polar Codes

A polar code $\mathcal{P}(N, K)$ of length N with K information bits is constructed by applying a linear transformation to the message word $\boldsymbol{u} = \{u_0, u_1, \ldots, u_{N-1}\}$ as $\boldsymbol{x} = \boldsymbol{u}\boldsymbol{G}^{\otimes n}$ where $\boldsymbol{x} = \{x_0, x_1, \ldots, x_{N-1}\}$ is the codeword, $\boldsymbol{G}^{\otimes n}$ is the n-th Kronecker power of the polarizing matrix $\boldsymbol{G} = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$, and $n = \log_2 N$. The vector $\boldsymbol{u}$ contains a set $\mathcal{A}$ of K information bits and a set $\mathcal{A}^c$ of $N - K$ frozen bits. The locations of the frozen bits are known to both the encoder and the decoder, and their values are set to 0. The codeword $\boldsymbol{x}$ is then modulated and sent through the channel where binary phase-shift keying (BPSK) modulation and AWGN channel model are considered. The soft vector of the transmitted codeword received by the decoder in this setting can be written as

$$\boldsymbol{y} = (\mathbf{1} - 2\boldsymbol{x}) + \boldsymbol{z}, \tag{15.1}$$

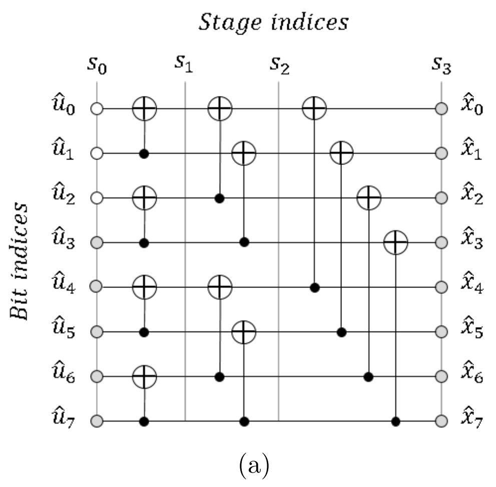
where $\mathbf{1}$ is an all-one vector of size N, and $\boldsymbol{z} \in \mathbb{R}^N$ is the AWGN vector with variance σ^2 and zero mean. In the log-likelihood ratio (LLR) domain, the LLR vector of the transmitted codeword is

$$\mathbf{LLR}_{\boldsymbol{x}} = \ln \frac{\Pr(\boldsymbol{x} = 0|\boldsymbol{y})}{\Pr(\boldsymbol{x} = 1|\boldsymbol{y})} = \frac{2\boldsymbol{y}}{\sigma^2}. \tag{15.2}$$

15.2.2. Successive-Cancellation Decoding

The SC decoding algorithm can be illustrated on a polar-code factor graph as shown in Figure 15.1a where $N = 8$ and $K = 4$. The messages are propagated through the SC processing elements (SCPEs) as shown in Figure 15.1b where $\beta_{t,s}$ denotes a left-to-right message, and $\alpha_{t,s}$ denotes a right-to-left message of the t-th bit index at stage s of the factor graph.

In SC decoding, the right-to-left messages $\alpha_{t,s}$ are soft LLR values and the left-to-right messages



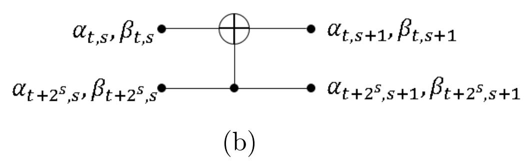

$\alpha_{t,s}, \beta_{t,s}$
$\alpha_{t,s+1}, \beta_{t,s+1}$
$\alpha_{t+2^s,s}, \beta_{t+2^s,s}$
$\alpha_{t+2^s,s+1}, \beta_{t+2^s,s+1}$
(b)

Figure 15.1: (a) Factor graph representation of a polar code with $N = 8$, $K = 4$, and $\{u_0, u_1, u_2\} \in \mathcal{A}^c$, (b) an SC processing element (SCPE).

$\beta_{t,s}$ are hard decision bits. The messages in the factor graph are updated as

$$\alpha_{t,s} = f\left(\alpha_{t,s+1}, \alpha_{t+2^s,s+1}\right), \tag{15.3}$$

$$\alpha_{t+2^s,s} = g\left(\alpha_{t,s+1}, \alpha_{t+2^s,s+1}, \beta_{t,s}\right), \tag{15.4}$$

$$\beta_{t,s+1} = \beta_{t,s} \oplus \beta_{t+2^s,s}, \tag{15.5}$$

$$\beta_{t+2^s,s+1} = \beta_{t+2^s,s}, \tag{15.6}$$

where

$$\begin{aligned} f\left(a,b\right) &= 2\operatorname{arctanh}\left(\tanh\left(\frac{a}{2}\right)\tanh\left(\frac{b}{2}\right)\right) \\ &\approx \operatorname{sgn}\left(a\right)\operatorname{sgn}\left(b\right)\min\left(\left|a\right|,\left|b\right|\right), \end{aligned} \tag{15.7}$$

$$g\left(a,b,c\right) = b + \left(1-2c\right)a, \tag{15.8}$$

and $\oplus$ is the bitwise XOR operation. SC decoding is initialized by setting $\alpha_{t,n} = y_t$ and the decoding schedule is such that the bits are decoded one by one from u_0 to u_{N-1}. At layer 0, the elements of

$\boldsymbol{u}$ are estimated as

$$\hat{u}_t = \begin{cases} 0, & \text{if } u_t \in \mathcal{A}^c \text{ or } \alpha_{t,0} \geq 0, \\ 1, & \text{otherwise.} \end{cases} \tag{15.9}$$

15.2.3. Successive-Cancellation List Decoding

SCL decoding improves the error-correction performance of SC decoding by running multiple SC decoders in parallel. Instead of using Eq. (15.9) to estimate $\boldsymbol{u}$ as in SC decoding, each bit is estimated considering both its possible values 0 and 1. Therefore, at each bit estimation, the number of candidates doubles. In order to constrain the high complexity of SCL decoding, at each bit estimation a set of only L candidates are allowed to survive based on a path metric which is calculated as [Balatsoukas-Stimming et al., 2015, Hashemi et al., 2016]

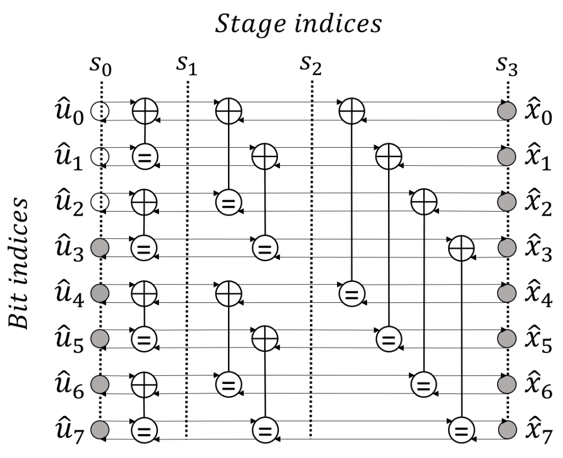
$$\mathrm{PM}_{t_\ell} = \sum_{j=0}^{t} \ln\left(1 + \mathrm{e}^{-(1-2\hat{u}_{j_\ell})\alpha_{j,0_\ell}}\right), \tag{15.10}$$

$$\approx \frac{1}{2}\sum_{j=0}^{t} \operatorname{sgn}(\alpha_{j,0_\ell})\alpha_{j,0_\ell} - (1-2\hat{u}_{j_\ell})\alpha_{j,0_\ell}, \tag{15.11}$$

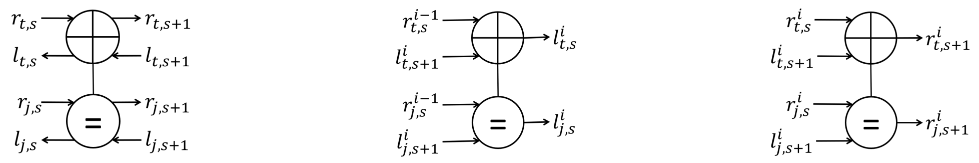
where ℓ is the path index and $\hat{u}_{j_\ell}$ is the estimate of bit j at path ℓ.

15.2.4. Belief Propagation Decoding

Figure 15.2a demonstrates BP decoding on the factor graph representation of $\mathcal{P}(8,5)$. The messages are iteratively propagated through the BP processing elements (BPPEs) [Arıkan, 2010] located in each stage. Each update iteration starts with a right-to-left message pass that propagates the LLR values from the channel (rightmost) stage, to the information bit (leftmost) stage, and ends with the left-to-right message pass which occurs in the opposite order. Figure 15.2b illustrates a BPPE with its corresponding soft messages, where $r_{t,s}$ denotes a left-to-right message, and $l_{t,s}$ denotes a right-to-left message of the t-th bit index at stage s. One can also apply BP decoding on the unrolled polar-code factor graph [Doan et al., 2018b], thus the BP iterations in this setup are performed sequentially. Figure 15.2c and Figure 15.2d illustrate the input and output messages of a BPPE for the right-to-left and left-to-right message updates on an unrolled factor graph, where



(a)



(b) (c) (d)

Figure 15.2: (a) BP decoding on the factor graph of $\mathcal{P}(8,5)$ with $\{u_0,u_1,u_2\} \in \mathcal{A}^c$, (b) a BPPE, (c) a right-to-left message update of a BPPE on an unrolled factor graph, and (d) a left-to-right message update of a BPPE on an unrolled factor graph.

the superscript i denotes the iteration number. The update rule [Arıkan, 2010] for the right-to-left messages of a BPPE is

$$\begin{cases} l^i_{t,s} & = f(l^i_{t,k}, r^{i-1}_{j,s} + l^i_{j,k}), \\ l^i_{j,s} & = f(l^i_{t,k}, r^{i-1}_{t,s}) + l^i_{j,k}, \end{cases} \tag{15.12}$$

and for the left-to-right messages is

$$\begin{cases} r^i_{t,k} & = f(r^i_{t,s}, l^i_{j,k} + r^i_{j,s}), \\ r^i_{j,k} & = f(r^i_{t,s}, l^i_{t,k}) + r^i_{j,s}, \end{cases} \tag{15.13}$$

where $j = t + 2^s$, $k = s + 1$.

The BP decoding performs a predetermined $I_{\max}$ update iterations where the messages are propagated through all BPPEs in accordance with Eq. (15.12) and Eq. (15.13). Initially, for $0 \leq t < N$ and $\forall i \leq I_{\max}$, $l^i_{t,n}$ are set to the received channel LLR values $\mathbf{LLR}_{\boldsymbol{x}}$, and $r^i_{t,0}$ are set to the

LLR values of the information and frozen bits as

$$\mathbf{LLR}_{\mathcal{A}\cup\mathcal{A}^c} = \begin{cases} 0, & \text{if } u_t \in \mathcal{A}, \\ +\infty, & \text{if } u_t \in \mathcal{A}^c. \end{cases} \tag{15.14}$$

All the other left-to-right and right-to-left messages of the BPPEs at the first iteration are set to 0. After running $I_{\max}$ iterations, the decoder makes a hard decision on the LLR values of the t-th bit at the information bit stage to obtain the estimated message word as

$$\hat{u}_t = \begin{cases} 0, & \text{if } r_{t,0}^{I_{\max}} + l_{t,0}^{I_{\max}} \geq 0, \\ 1, & \text{otherwise.} \end{cases} \tag{15.15}$$

15.3. Deep Learning-Based Decoding for Polar Codes

As mentioned in the above, this section is devoted to the use of DL in decoding polar codes with emphasis on the off-the-shelf DL decoders and DL-aided decoders by addressing their working principles, algorithm details, and performance evaluations.

15.3.1. Off-the-Shelf Deep Learning Decoders for Polar Codes

In [Gruber et al., 2017], it was shown that a MLP decoder can generalize structured codes, e.g. polar codes, more effectively than random codes, where the MAP performance was obtained for structured codes but not for random codes. However, the considered code length in [Gruber et al., 2017] is only limited to 16. A more detailed investigation was carried out in [Lyu et al., 2018], where a comparison was performed between different off-the-shelf network models including MLP, convolutional neural network (CNN), and RNN, for polar codes. The RNN model outperforms the MLP and CNN models in terms of error-correction performance with the cost of highest decoding complexity. On the other hand, the CNN model provided a performance gain over the MLP model with a higher computational time. It was also observed in [Lyu et al., 2018] that the code length impacts the fitting (underfitting versus overfitting) of the deep NN and each type of NN has a saturation code length, which is related to the learning capabilities of the model.

For all the aforementioned off-the-shelf decoders, the networks are formalized to solve a multi-category classification problem where the correct codewords are used as the training labels, and the corresponding values of $\mathbf{LLR}_x$ are used as the network's input. Normally the size of the DL decoders scales with the size of the codeword and the natural architecture of the DL models in use. Finding the network parameters or weights is done by back-propagation [LeCun et al., 2015] with various optimization methods such as ADAM [Kingma and Ba, 2014] or RMSPROP [Hinton et al.]. Polar decoding under off-the-shelf DL decoders is carried out by performing the inference phase of the trained DL models, given the channel LLR values.

The main problem associated with off-the-shelf DL decoders when applied to polar codes or other linear block codes is *the curse of dimensionality* [Gruber et al., 2017], which states that the number of required training samples scales exponentially with the code lengths. To overcome this issue, a scaling approach was introduced in [Cammerer et al., 2017] constraining the DL decoders to only work with sub-codes with small code sizes. Specifically, a partitioned NN (PNN) decoder for a polar code of size 128 was proposed. The considered polar code was divided into smaller sub-blocks and the partitioned DL decoders are trained individually so that the performance obtained for each sub-block was close to that of MAP decoding. However, the bit-error-rate (BER) performance of the integrated system is only similar to that of SC decoding. It is worth mentioning that the latency of the proposed decoder can be reduced as parallel computations can be exploited for the DL decoders thanks to their one-shot-decoding property.

In [Doan et al., 2018a], a neural decoder was introduced on the basis of the partitioning idea of [Cammerer et al., 2017]. The proposed neural SC (NSC) in [Doan et al., 2018a] preserves the same decoding performance in terms of BER and frame error rate (FER) as that of PNN with a decoding latency improved by 42.5% for a polar code of length 128 and rate 0.5.

15.3.2. Deep Learning-Aided Decoders for Polar Codes

15.3.2.1. Neural Belief Propagation Decoders

In contrast to off-the-shelf DL decoders, another approach is to exploit *domain knowledge* to design DL-aided decoders. In [Nachmani et al., 2018], a deep neural BP decoder is proposed to improve conventional BP decoding where trainable weights are assigned to the edges of the unrolled factor graph. The deep network in this case is constrained to be a partially connected NN and its inference functions resemble the operations of conventional BP decoding. This idea was first evaluated on high density parity check (HDPC) codes in [Nachmani et al., 2018] and then on polar codes in [Doan et al., 2018b]. It was observed that the trainable weights help the conventional BP decoding to mitigate the detrimental effects of the code's short cycles, which are often found in practical linear codes.

Another problem associated with BP decoding is the costly sum-product (SP) algorithm [Ryan and Lin, 2009]. Instead, a low complexity min-sum (MS) algorithm is used in practical applications [Ryan and Lin, 2009]. However, the MS algorithm also introduces decoding errors due to its poor estimation compared to the SP algorithm. With the objective of tackling this challenge, neural offset min-sum (NOMS) decoding was proposed in [Lugosch and Gross, 2017]. The NOMS algorithm trains offset parameters and uses them to correct the MS approximation. It should be noted that NOMS decoding only requires additions which makes this decoder particularly attractive for hardware implementation.

In [Nachmani et al., 2018], the architectures from [Nachmani et al., 2018] and [Lugosch and Gross, 2017] were changed to resemble an RNN by reusing the weights at each iteration; this reduces their complexity as fewer parameters are needed. Furthermore, an RNN architecture based on a successive relaxation technique was constructed which further improved the proposed RNN-like neural BP decoders.

In [Xu et al., 2017], a neural normalized min-sum (NNMS) decoder was developed, which adapts the idea in [Nachmani et al., 2018] for the case of polar codes. NNMS also uses a multiplicative weight to correct the min-sum approximation. This setup can be scaled to large size polar codes while still maintaining a low decoding latency and complexity.

It was shown in [Ren et al., 2015] that the CRC capability is only used as an early stopping criterion with incremental error-correction performance improvement for BP decoding of polar codes. In [Doan et al., 2018b], by assigning trainable weights to the CRC-Polar concatenated graph, the proposed decoder has shown a performance gain of 0.5 dB over the conventional CRC-aided BP at the FER of 10^{-5}, for a 5G polar code of length 128. The authors in [Doan et al., 2018b] also derived a general neural BP decoder architecture specified for polar codes. Figure 15.3 illustrates

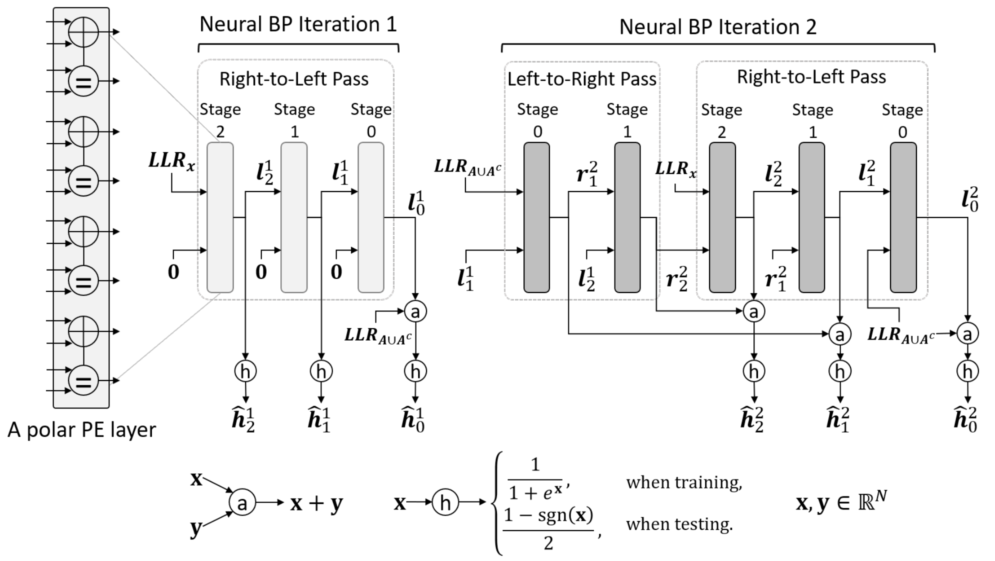


Figure 15.3: A neural BP decoder architecture with 2 iterations for $\mathcal{P}(8,5)$

an example of this architecture where the weights are shared between each neural BP decoding iteration. The weight assignment schemes of state-of-the-art neural BP decoders when applied to the BPPE update functions in Eq. (15.12) and Eq. (15.13) are summarized as follows.

- NNMS-RNN [Nachmani et al., 2018]

$$\begin{cases} l_{t,s}^i &= w_0 f(l_{t,k}^i, w_1 r_{j,s}^{i-1} + w_2 l_{j,k}^i), \\ l_{j,s}^i &= w_4(w_3 f(l_{t,k}^i, r_{t,s}^{i-1})) + w_5 l_{j,k}^i, \end{cases} \tag{15.16}$$

$$\begin{cases} r_{t,k}^i &= w_6 f(r_{t,s}^i, w_7 l_{j,k}^i + w_8 r_{j,s}^i), \\ r_{j,k}^i &= w_{10}(w_9 f(r_{t,s}^i, l_{t,k}^i)) + w_{11} r_{j,s}^i, \end{cases} \tag{15.17}$$

- NOMS [Lugosch and Gross, 2017]

$$\begin{cases} l_{t,s}^{i} &= \operatorname{sgn}(l_{t,k}^{i}) \operatorname{sgn}(r_{j,s}^{i-1} + l_{j,k}^{i}) \max\left(0, \min\left(|l_{t,k}^{i}|, |r_{j,s}^{i-1} + l_{j,k}^{i}|\right) - w_0\right), \\ l_{j,s}^{i} &= \operatorname{sgn}(l_{t,k}^{i}) \operatorname{sgn}(r_{t,s}^{i-1}) \max\left(0, \min\left(|l_{t,k}^{i}|, |r_{t,s}^{i-1}|\right) - w_3\right), \end{cases} \tag{15.18}$$

$$\begin{cases} r_{t,k}^{i} &= \operatorname{sgn}(r_{t,s}^{i}) \operatorname{sgn}(l_{j,k}^{i} + r_{j,s}^{i}) \max\left(0, \min\left(|r_{t,s}^{i}|, |l_{j,k}^{i} + r_{j,s}^{i}|\right) - w_6\right), \\ r_{j,k}^{i} &= \operatorname{sgn}(r_{t,s}^{i}) \operatorname{sgn}(l_{t,k}^{i}) \max\left(0, \min\left(|r_{t,s}^{i}|, l_{t,k}^{i}|\right) - w_9\right) + r_{j,s}^{i}, \end{cases} \tag{15.19}$$

- NNMS [Xu et al., 2017]

$$\begin{cases} l_{t,s}^{i} &= w_0 f(l_{t,k}^{i}, r_{j,s}^{i-1} + l_{j,k}^{i}), \\ l_{j,s}^{i} &= w_3 f(l_{t,k}^{i}, r_{t,s}^{i-1}) + l_{j,k}^{i}, \end{cases} \tag{15.20}$$

$$\begin{cases} r_{t,k}^{i} &= w_6 f(r_{t,s}^{i}, l_{j,k}^{i} + r_{j,s}^{i}), \\ r_{j,k}^{i} &= w_9 f(r_{t,s}^{i}, l_{t,k}^{i}) + r_{j,s}^{i}, \end{cases} \tag{15.21}$$

where $w_m \in \mathbb{R}$ $(0 \leq m \leq 11)$ are the trainable weights.

Optimizing the weights of the neural BP decoders, as depicted in Figure 15.3, can be done through back-propagation in order to minimize the following objective function

$$Loss = \sum_{i=1}^{I_{\max}} \sum_{s=0}^{n-1} H_{\text{CE}}(\hat{\boldsymbol{h}}_s^i, \boldsymbol{h_s}), \tag{15.22}$$

where H_{CE} is the cross-entropy function, and $\boldsymbol{h_s}$ is the correct hard value vector at stage s of the polar code factor graph which is obtained from the training samples. In the decoding phase, only the hard estimated values at stage 0 of the polar code factor graph, i.e. $\hat{\boldsymbol{h}}_0^i$ $(1 \leq i \leq I_{\max})$ is required to obtain the decoded message bits.

15.3.2.2. Joint Decoder and Noise Estimator

In [Liang et al., 2018] a CNN-based noise estimator is used to remove interference noise between channels. The noise estimator is then coupled with a conventional BP decoder. Although the proposed iterative denoising-decoding approach in [Liang et al., 2018] showed a significant performance gain in the case of strong correlations between channels, when there is no correlation, this approach only achieved a negligible error probability gain over the conventional BP decoder.

A multiplicative noise model was introduced in [Bennatan et al., 2018] based on the decoding syndrome. This noise model has two advantages compared to that of the additive noise-estimation model used in [Liang et al., 2018]. First, it reduces the regression problem of the noise model in [Liang et al., 2018] to a classification problem, i.e. only learn to estimate the sign of the noise instead of the actual noise value, which is more feasible to learn given the high dimensional state of the model inputs. Second, the multiplicative noise-estimation model also preserves the code symmetry conditions [Richardson and Urbanke, 2008], allowing for the use of all-zero codewords during training.

Figure 15.4 depicts a joint decoder-noise estimator approach derived from [Liang et al., 2018, Bennatan et al., 2018], where the noise estimator is a DL model such as NN, CNN, and RNN. Unlike the off-the-shelf DL decoders, which try to directly predict the message words given the channel LLRs, the joint decoder and noise estimator approach only utilizes the off-the-shelf DL models to denoise the channel LLR values, while the main decoding algorithm is still carried out by conventional BP decoding.

The iterative decoding algorithm in Figure 15.4 starts with the first decoding attempt by running the conventional BP decoding given the channel input $\mathbf{LLR}_x$. If the estimated codeword $\hat{\boldsymbol{x}}$ and the estimated message word $\hat{\boldsymbol{u}}$ do not satisfy the G-matrix based termination condition [Yuan and Parhi, 2013], the channel LLR values will be denoised and followed by another BP decoding attempt. Given the syndrome of the conventional BP decoding algorithm, $\hat{\boldsymbol{x}}\boldsymbol{H}^{\boldsymbol{T}}$, where $\boldsymbol{H}$ is the parity-check matrix of polar codes, and the absolute values of the channel LLR values, $|\mathbf{LLR}_x|$, the DL-based noise estimator predicts the channel noise by estimating its sign values, $\hat{\boldsymbol{q}}$. The channel LLR values are then updated by flipping the signs at certain positions predicted by the noise estimator, which results in the denoised channel LLR values, $\mathbf{LLR'}_x$. Another BP decoding attempt is then carried

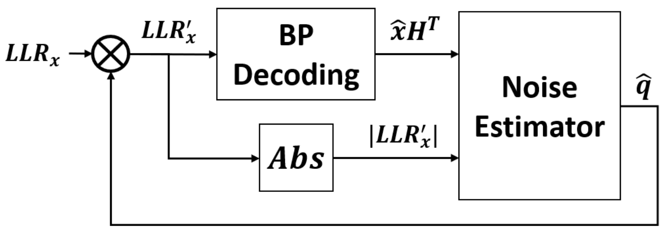


Figure 15.4: A joint BP decoder-DL noise estimator as proposed in [Liang et al., 2018], the input of the noise estimator is the syndrome of the estimated codeword and the magnitude of the estimated channel LLRs [Bennatan et al., 2018].

out given the denoised LLR values. Finally, the decoding is terminated if the mentioned termination condition is satisfied or a predetermined maximum number of decoding attempts is reached.

It is worth noticing that the training samples of the noise estimator depicted in Figure 15.4 only include the erroneous syndromes after the first BP decoding, i.e. when $\hat{\boldsymbol{x}}\boldsymbol{H}^{\boldsymbol{T}}$ is a nonzero vector, and the corresponding absolute values of the channel LLRs. The label $\boldsymbol{q}$, $\boldsymbol{q} \subset \{-1, 1\}^N$, is used as the correct output label, where $q_j = -1$ indicates a flip at the j-th element of the channel LLRs, while $q_j = 1$ indicates there is no change required for the j-th element.

15.3.3. Evaluation

In this section, we provide a performance comparison in terms of FER for various state-of-the-art DL-aided BP decoders when applied to a polar code. In addition, the FER performance of conventional decoders including BP [Arıkan, 2010] and SCL [Balatsoukas-Stimming et al., 2015, Hashemi et al., 2016] decoding are also plotted. The examined polar code has a code length of 64, with 32 information bits and is selected for the eMBB control channel of 5G [3GPP, 2018]. We denote BP$I_{\max}$, where $I_{\max} \in \{5, 30\}$, as the conventional min-sum BP decoder with $I_{\max}$ iterations, and SCL32 [Tal and Vardy, 2015] as the SCL decoder with a list size of 32. All the neural BP decoders considered in this section contain 5 unrolled BP iterations.

We also examine a joint decoder and channel equalizer decoding system by exploiting the idea proposed in [Liang et al., 2018, Bennatan et al., 2018]. We denote the joint decoding systems as BP5-MLP-BP5 and BP5-LSTM-BP5, where MLP and LSTM refer to the noise estimator models using fully connected NNs and stacked LSTM networks, respectively. Note that the network architectures and parameters of the DL-based noise estimators are adopted from [Bennatan et al., 2018].

As all the DL-aided decoders considered in this section satisfy the code symmetry conditions [Richardson and Urbanke, 2008], only all-zero codewords are required for training. The training data is obtained for various E_b/N_0 values where $E_b/N_0 \in \{1, 2, 3, 4, 5, 6, 7, 8\}$. At each E_b/N_0 value, 10^5 all-zero codewords are obtained using BPSK modulation and AWGN channel model. All neural BP-based decoders are trained for 100 epochs while BP5-MLP-BP5 and BP5-LSTM-BP5 are trained for 1000 epochs. The minibatch size is set to 320 and RMSPROP [Hinton et al.] is used as the optimization algorithm for training. Keras [Chollet et al., 2015] and Tensorflow [Abadi et al., 2016] are used as our DL frameworks. During testing, each decoder decodes at least 10^4 random codewords to obtain at least 50 frames in errors at each E_b/N_0 value.

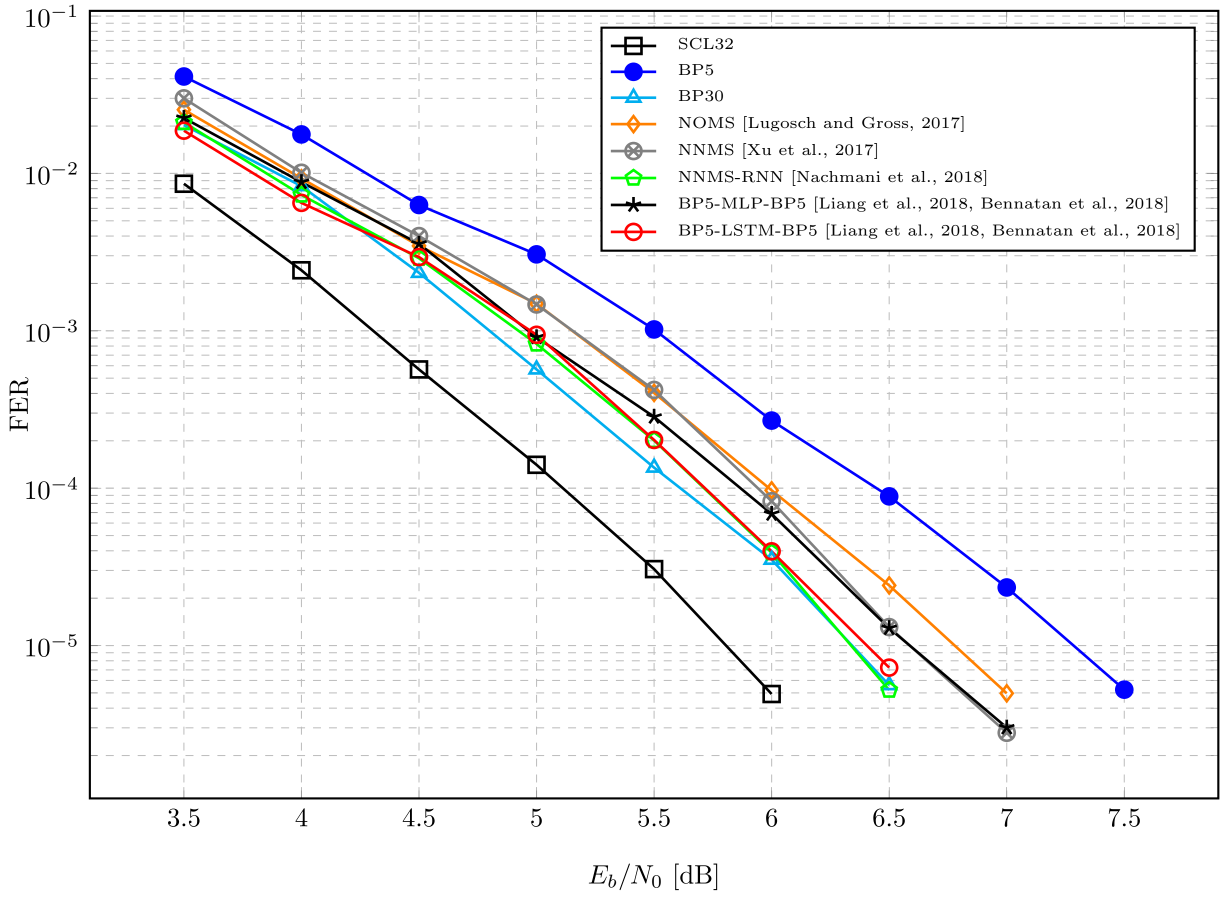


Figure 15.5: FER performance of various decoders for $\mathcal{P}(64, 32)$ selected for 5G.

Figure 15.5 illustrates the FER performance of the above mentioned decoders. Table 15.1 summarizes the error-correction performance gains of all the decoders in Figure 15.5 with respect to the baseline BP5 decoder at a target FER of 10^{-5}. As observed from Table 15.1, NOMS provides a gain of 0.5 dB compared to the baseline BP5 decoder, while NNMS and BP5-MLP-BP5 both have a gain of 0.7 dB. On the other hand, NNMS-RNN and BP30 have the same error-correction performance, which is around 1.0 dB better than that of the baseline BP5, while BP5-LSTM-BP5 is slightly worse than NNMS-RNN. It is worth mentioning that the best neural BP decoder, NNMS-RNN, is still 0.5 dB away from SCL32.

SCL32	BP30	NOMS	NNMS	NNMS-RNN	BP5-MLP-BP5	BP5-LSTM-BP5
1.5	1.0	0.5	0.7	1.0	0.7	0.9

Table 15.1: Performance gain in dB when compared with BP5 at FER=10^{-5}

The decoding latency in terms of time steps for a polar code of size N under BP decoding with $I_{\max}$ iterations can be calculated as [Arıkan, 2010]

$$\mathcal{T}_{\mathrm{BP}I_{\max}} = 2I_{\max}\log_2(N). \tag{15.23}$$

As the unrolled factor graph of the NOMS, NNMS, and NNMS-RNN decoders are equivalent to that of a traditional BP decoder with 5 iterations, their decoding latency can also be calculated by Eq. (15.23). For BP5-MLP-BP5 and BP5-LSTM-BP5 decoders, their decoding latency in time steps is the sum of the time steps consumed by two successive BP decoders with 5 iterations, and a deep NN with a depth of 5. Therefore, their decoding latency can be calculated as [Liang et al., 2018]:

$$\mathcal{T}_{\mathrm{BP}I_{\max}\text{-MLP/LSTM-BP}I_{\max}} = 4I_{\max}\log_2(N) + \mathit{Depth}_{\mathrm{MLP/LSTM}}. \tag{15.24}$$

On the other hand, the SCL32 decoder of [Balatsoukas-Stimming et al., 2015] requires $(2N+K-2)$ time steps.

Figure 15.6 illustrates the decoding latency in time steps for all the neural decoders considered in Figure 15.5. It should be noted that by assigning trainable weights to the factor graph of polar codes, NNMS-RNN with 5 iterations is able to achieve the same error-correction performance of BP30, which also results in a saving of 300 time steps. In addition, the decoding latency of BP5-MLP-BP5 and BP5-LSTM-BP5 is 65 time steps greater than that of NNMS-RNN.

Table 15.2 gives a detailed comparison in the number of weights required by different DL-aided BP decoders in Figure 15.5. Although it is demonstrated in Figure 15.5 that off-the-shelf deep

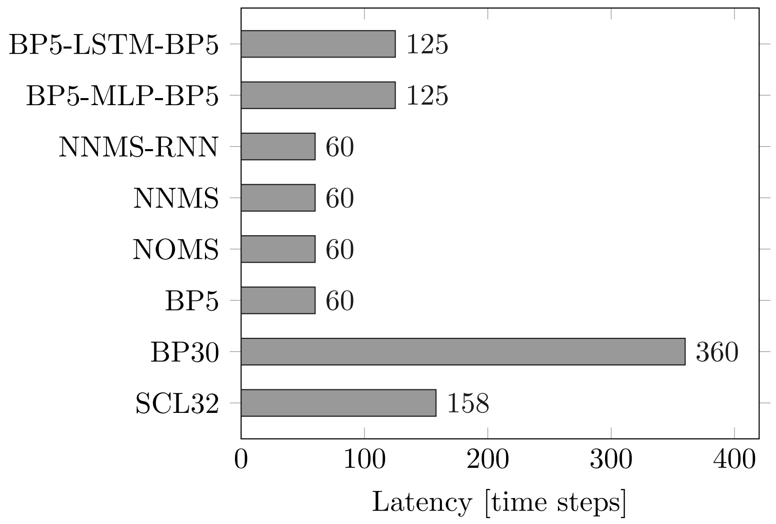


Figure 15.6: Latency comparison of various decoders for $\mathcal{P}(64, 32)$ selected for 5G.

networks are able to estimate channel noise, this approach shows inefficiency since relatively large DL models are required for the task. On the contrary, by incorporating the conventional BP decoding algorithm to define a constrained network model, NNMS-RNN can provide a reasonable error probability while only requiring a small number of weights compared to those of BP5-MLP-BP5 and BP5-LSTM-BP5. Furthermore, although NOMS and NNMS only require 33.33% of the number of weights consumed by NNMS-RNN, the smaller number of weights results in a considerable error-correction performance loss as observed in Figure 15.5.

NOMS	NNMS	NNMS-RNN	BP5-MLP-BP5	BP5-LSTM-BP5
288	288	864	5403712	3446976

Table 15.2: Number of trainable parameters required for different DL-aided BP decoders.

15.4. Conclusions

In this chapter, we have discussed a wide range of fruitful applications of digital communications where deep learning (DL) can play a vital role. We provided an overview on DL techniques with a focus on forward error correction (FEC) codes and examined state-of-the-art DL-aided decoders for polar codes as our case study. It was demonstrated that off-the-shelf DL decoders can reach a maximum-a-posteriori (MAP) decoding performance for short code lengths and that they enable parallel executions thanks to their one-shot-decoding property. However, for longer code lengths, of-the-shelf DL decoders require a training dataset which scales exponentially with the code length. This issue becomes the main challenge of those decoders in practical applications. On the other hand, domain knowledge can be exploited to design DL-aided decoders as demonstrated by various neural belief propagation (BP) decoders. It was shown that neural BP decoders can obtain significant decoding performance gains over their conventional counterpart, while maintaining the same decoding latency.

Future research on applying DL techniques to FEC can be carried out in various directions, such as designing jointly-trained systems of customized DL-aided decoders and neural channel-noise estimators for various non-linear communication channels. In addition, the sequential decoding of linear block codes such as polar codes can be suitably formalized as a reinforcement learning (RL) problem, thus greatly enabling the applications of state-of-the-art RL algorithms to FEC. Other approaches may include the use of DL techniques as optimization methods for well-defined problems of conventional decoding algorithms, whose solutions are obtained based on an approximation or a massive Monte-Carlo simulation.

Bibliography

3GPP. Multiplexing and channel coding (Release 10) 3GPP TS 21.101 v10.4.0. Oct. 2018. URL `http://www.3gpp.org/ftp/Specs/2018-09/Rel-10/21_series/21101-a40.zip`.

M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, et al. Tensorflow: A system for large-scale machine learning. *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, pages 265–283, 2016.

E. Arıkan. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Transactions on Information Theory*, 55(7):3051–3073, July 2009. ISSN 0018-9448. doi: 10.1109/TIT.2009.2021379.

E. Arıkan. Polar codes: A pipelined implementation. *Proceedings of the 4th International Symposium on Broadband Communications*, pages 11–14, 2010.

A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg. LLR-based successive cancellation list decoding of polar codes. *IEEE Transactions on Signal Processing*, 63(19):5165–5179, Oct 2015. ISSN 1053-587X. doi: 10.1109/TSP.2015.2439211.

A. Bennatan, Y. Choukroun, and P. Kisilev. Deep learning for decoding of linear codes - a syndrome-based approach. *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 1595–1599, June 2018. ISSN 2157-8117. doi: 10.1109/ISIT.2018.8437530.

C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo-codes. *Proceedings of ICC '93 - IEEE International Conference on Communications*, 2:1064–1070 vol.2, May 1993. doi: 10.1109/ICC.1993.397441.

J. Bruck and M. Blaum. Neural networks, error-correcting codes, and polynomials over the binary n-cube. *IEEE Transactions on Information Theory*, 35(5):976–987, Sept 1989. ISSN 0018-9448. doi: 10.1109/18.42215.

S. Cammerer, T. Gruber, J. Hoydis, and S. t. Brink. Scaling deep learning-based decoding of polar codes via partitioning. *IEEE Global Communication Conference*, pages 1–6, December 2017. doi: 10.1109/GLOCOM.2017.8254811.

F. Chollet et al. Keras. `https://keras.io`, 2015.

N. Doan, S. A. Hashemi, and W. J. Gross. Neural successive cancellation decoding of polar codes. *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5, June 2018a. ISSN 1948-3252. doi: 10.1109/SPAWC.2018.8445986.

N. Doan, S. A. Hashemi, E. N. Mambou, T. Tonnellier, and W. J. Gross. Neural belief propagation decoding of CRC-Polar concatenated codes. *arXiv preprint arXiv:1811.00124*, 2018b.

S. Dörner, S. Cammerer, J. Hoydis, and S. t. Brink. Deep learning based communication over the air. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):132–143, Feb 2018. ISSN 1932-4553. doi: 10.1109/JSTSP.2017.2784180.

N. Farsad, M. Rao, and A. Goldsmith. Deep learning for joint source-channel coding of text. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2326–2330, April 2018. ISSN 2379-190X. doi: 10.1109/ICASSP.2018.8461983.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems 27*, pages 2672–2680, 2014.

A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602 – 610, 2005. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2005.06.042. IJCNN 2005.

T. Gruber, S. Cammerer, J. Hoydis, and S. t. Brink. On deep learning-based channel decoding. *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6, March 2017. doi: 10.1109/CISS.2017.7926071.

S. A. Hashemi, C. Condo, and W. J. Gross. A fast polar code list decoder architecture based on sphere decoding. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 63(12): 2368–2380, Dec 2016. ISSN 1549-8328. doi: 10.1109/TCSI.2016.2619324.

S. Hasim, W. S. Andrew, and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. *Fifteenth annual conference of the international speech communication association (INTERSPEECH)*, 2014.

G. Hinton, N. Srivastava, and K. Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. URL `https://cs.toronto.edu/ tijmen/csc321/slides/lecture_slides_lec6.pdf`.

M. Ibnkahla. Applications of neural networks to digital communications–a survey. *Signal processing*, 80(7):1185–1215, 2000. doi: 10.1016/s0165-1684(00)00030-x.

H. Kim, Y. Jiang, R. Rana, S. Kannan, S. Oh, and P. Viswanath. Communication algorithms via deep learning. *International Conference on Learning Representations (ICLR)*, 2018.

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

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, May 2015.

F. Liang, C. Shen, and F. Wu. An iterative BP-CNN architecture for channel decoding. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):144–159, Feb 2018. ISSN 1932-4553. doi: 10.1109/JSTSP.2018.2794062.

L. Lugosch and W. J. Gross. Neural offset min-sum decoding. *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 1361–1365, June 2017. ISSN 2157-8117. doi: 10.1109/ISIT.2017.8006751.

W. Lyu, Z. Zhang, C. Jiao, K. Qin, and H. Zhang. Performance evaluation of channel decoding with deep neural networks. *2018 IEEE International Conference on Communications (ICC)*, pages 1–6, May 2018. ISSN 1938-1883. doi: 10.1109/ICC.2018.8422289.

E. Nachmani, 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*, 12(1):119–131, February 2018. ISSN 1932-4553. doi: 10.1109/JSTSP.2017.2788405.

T. J. O'Shea and J. Hoydis. An introduction to deep learning for the physical layer. *IEEE Transactions on Cognitive Communications and Networking*, 3(4):563–575, Dec 2017. ISSN 2332-7731. doi: 10.1109/TCCN.2017.2758370.

Y. Ren, C. Zhang, X. Liu, and X. You. Efficient early termination schemes for belief-propagation decoding of polar codes. *IEEE 11th International Conference on ASIC*, pages 1–4, Nov 2015. doi: 10.1109/ASICON.2015.7517046.

T. Richardson and R. Urbanke. *Modern coding theory.* Cambridge university press, 2008.

W. Ryan and S. Lin. *Channel codes: classical and modern.* Cambridge University Press, 2009.

N. Samuel, T. Diskin, and A. Wiesel. Deep MIMO detection. *2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5, July 2017. ISSN 1948-3252. doi: 10.1109/SPAWC.2017.8227772.

S. Sun and Y. Jing. Training and decodings for cooperative network with multiple relays and receive antennas. *IEEE Transactions on Communications*, 60(6):1534–1544, June 2012. ISSN 0090-6778. doi: 10.1109/TCOMM.2012.050912.110380.

I. Tal and A. Vardy. List decoding of polar codes. *IEEE Transactions on Information Theory*, 61 (5):2213–2226, May 2015. ISSN 0018-9448. doi: 10.1109/TIT.2015.2410251.

L. G. Tallini and P. Cull. Neural nets for decoding error-correcting codes. *IEEE Technical Applications Conference and Workshops. Northcon/95. Conference Record*, pages 89–, Oct 1995. doi: 10.1109/NORTHC.1995.485019.

A. Viterbi. Convolutional codes and their performance in communication systems. *IEEE Transactions on Communication Technology*, 19(5):751–772, October 1971. ISSN 0018-9332. doi: 10.1109/TCOM.1971.1090700.

T. Wang, C. Wen, H. Wang, F. Gao, T. Jiang, and S. Jin. Deep learning for wireless physical layer: Opportunities and challenges. *China Communications*, 14(11):92–111, Nov 2017. ISSN 1673-5447. doi: 10.1109/CC.2017.8233654.

C. Wen, W. Shih, and S. Jin. Deep learning for massive MIMO CSI feedback. *IEEE Wireless Communications Letters*, 7(5):748–751, Oct 2018. ISSN 2162-2337. doi: 10.1109/LWC.2018.2818160.

J. L. Wu, Y. H. Tseng, and Y. M. Huang. Neural network decoders for linear block codes. *International Journal of Computational Engineering Science*, 3(3):235–256, 2002. doi: 10.1142/S1465876302000629.

W. Xu, Z. Wu, Y. Ueng, X. You, and C. Zhang. Improved polar decoder based on deep learning. *2017 IEEE International Workshop on Signal Processing Systems (SiPS)*, pages 1–6, Oct 2017. ISSN 2374-7390. doi: 10.1109/SiPS.2017.8109997.

B. Yuan and K. K. Parhi. Architecture optimizations for bp polar decoders. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2654–2658, May 2013. ISSN 1520-6149. doi: 10.1109/ICASSP.2013.6638137.