4 September 2025

Model-free Neural Decoding for Bit-Interleaved Coded Modulations

Gastón De Boni Rovella[1], Meryem Benammar[1], Tarik Benaddi[1], Hugo Meric[2]

1. Fédération ENAC ISAE-SUPAERO ONERA, Université de Toulouse

2. CNES

Abstract

In this work, we investigate neural network decoding of short codes, and more specifically, Syndrome-Based Neural Decoding (SBND) for Bit-Interleaved Coded Modulations (BICM). We generalize SBND for BICM regardless of the constellation shape, size, labeling, and Signal-to-Noise-Ratio (SNR). We similarly prove theoretical optimality of the proposed decoder and establish that the single-codeword trainable property holds. Moreover, we introduce and build an iterative SBND (I-SBND), which improves the performance for labelings other than Gray. Additionally, we propose a novel recurrent transformer architecture to implement the neural decoder, which greatly reduces the number of parameters used in the architecture whilst maintaining competitive error performances. We implement several neural architectures for different codes and rates, and compare their performances in error probability and complexity.

Keywords

bit-interleaved coded modulations, channel coding, communication, networking and broadcast technologies, deep learning, iterative decoding, neural networks

Model-free Neural Decoding for Bit-Interleaved Coded Modulations

Gastón De Boni Rovella, *Member, IEEE*, Meryem Benammar, *Member, IEEE*,
Tarik Benaddi, *Member, IEEE*, Hugo Meric *Member, IEEE*

***Abstract*—In this work, we investigate neural network decoding of short codes, and more specifically, Syndrome-Based Neural Decoding (SBND) for Bit-Interleaved Coded Modulations (BICM). We generalize SBND for BICM regardless of the constellation shape, size, labeling, and Signal-to-Noise-Ratio (SNR). We similarly prove theoretical optimality of the proposed decoder and establish that the single-codeword trainable property holds. Moreover, we introduce and build an iterative SBND (I-SBND), which improves the performance for labelings other than Gray. Additionally, we propose a novel recurrent transformer architecture to implement the neural decoder, which greatly reduces the number of parameters used in the architecture whilst maintaining competitive error performances. We implement several neural architectures for different codes and rates, and compare their performances in error probability and complexity.**

***Index Terms*—Channel coding, deep learning, bit-interleaved coded modulations, neural networks, iterative decoding.**

I. Introduction

THE last few decades have been witnesses to an evergrowing interest in machine learning-based, and more specifically *deep learning*-based, solutions as enhancers or substitutes for many components of the communication chain [1], [2], [3]. Channel coding is no exception, it being an essential —and often time-consuming— part of the communication system. While early neural-based solutions for channel coding were presented over thirty years ago [4], [5], [6], a renewed interest was triggered by the work of Gruber et al. [7] and resulted in numerous subsequence contributions (see [8], [9]).

Gruber et al. [7] showed that Deep Neural Networks (DNN) —such as Multi-Layer Perceptrons (MLP)— could fully learn to decode a linear code with close-to-optimal error probability. However, this result remains valid only for very short codes (up to around 20 bits), for which the neural network can be trained over the entire codebook of the code, and is thus able to *memorize* it. However, larger codes present an exponentially growing codebook size that makes this approach intractable, and the network's learning capacity rapidly shrinks. This effect is known as the *curse of dimensionality* and remains one of the biggest challenges pertaining to neural-based decoding.

Related works: Two main approaches in the literature allow for better scalability, i.e., the capacity to extend to codes with codebooks that are too large to be fully explored in training.

G. De Boni Rovella was with the Fédération ENAC ISAE-SUPAERO ONERA, and with TéSA research laboratory, Toulouse, France. He is currently in Facultad de Ingeniería, UdelaR, Montevideo, Uruguay. M. Benammar is with the Fédération ENAC ISAE-SUPAERO ONERA, Université de Toulouse, France. T. Benaddi is with Thales Alenia Space and H. Meric is with CNES, Toulouse, France. E-mails: gdeboni@fing.edu.uy, meryem.benammar@isae-supaero.fr, tarik.benaddi@thalesaleniaspace.com, hugo.meric@cnes.fr. Part of these results were presented at the 2024 IEEE International Conference on Machine Learning for Communications and Networking, Stockholm.

The first approach is the *model-based* approach, introduced by Nachmani et al. [10], which is a neural extension of the Belief Propagation (BP) decoding of Gallager's Low-Density Parity Check (LDPC) codes [11]. This approach consists in assigning weights to the edges of the Tanner graph used by the BP algorithm, and then, training the resulting neural network to minimize the error rate. Due to the iterative nature of the BP algorithm, [10] resorts to a *deep unfolding* procedure, where the Tanner graph is unfolded into a DNN with twice as many layers as iterations employed, and the links between the layers abide by the structure of the code's bipartite graph. This technique has been shown to partially reduce the negative effect related to the presence of short cycles in the Tanner graph, and systematically improve performances of the BP algorithm in dense codes. The main advantage of this technique is that it preserves the message passing symmetry conditions of the original BP, and hence, it suffices to train the network with randomly generated noisy observations of a single codeword, e.g., the all-zero codeword. Subsequent contributions using this approach targetted lower complexity and better performance. In [12], the weights of the edges of the Tanner graph are tied to form a recurrent neural decoder of reduced complexity. In [13], the syndrome is incorporated to the loss function in order to aid the decoder to converge towards a valid codeword. An autoregressive architecture is implemented in [14], which sacrifices the single-codeword training property but increases performances significantly. The authors in [15] apply a post-processing stage based on Ordered Statistics Decoding (OSD), while the model-based decoder was also adapted to factor graphs for decoding polar codes [16]. More examples can be found in [9].

The second approach is the *model-free* approach, introduced by Bennatan et al. in [17] and tailored for transmissions over a Binary-Input Additive White Gaussian Noise (BI-AWGN) channel. In this approach, the proposed decoder seeks to estimate the positions of the Binary Phase-Shift Keying (BPSK) symbols in the modulated codeword that have suffered a change in their sign, or analogously, in the binary domain, a *bitflip*. To this end, the received noisy symbols undergo a pre-processing that extracts their absolute values (reliabilities) and the syndrome associated with their hard decisions and are then fed to a neural-based bitflip estimator which yields soft estimates on the codeword bitflips. Authors in [17] proved that these two elements, namely the reliabilities and the syndrome, are sufficient statistics for the optimal estimation of the bitflip positions and are independent of the transmitted codeword. Hence, the training can be performed using a unique codeword, thus allowing for good scalability of the decoder. A further contribution employing this approach introduced a

transformer-based architecture [18], which was revisited by Park et al. in [19] where two distinct parity-check matrices are used to diversify the information extracted by the neural network, and again in [20], where the authors combine the transformer architecture with they main operational principles of conventional message-passing decoders. Other works have also explored iterative approaches, e.g., employing the decoder several times in a sequential manner [21], or using denoising diffusion models [22].

Motivation and contributions: Nachmani's model-based approach [10] was proposed as an improved BP decoding for short-to-medium length BCH codes, which are high-density parity-check codes and thus prone to short cycles in their Tanner graph. However, BP decoding is well-suited for very large and sparse codes —such as LDPC codes— due to the absence of short cycles in the bipartite graph induced by the parity-check matrix of the code. In contrast, when applied to short or medium-length and dense codes —namely, BCH codes and polar codes—, BP performs poorly compared to Maximum Likelihood (ML) decoding. Even if neural BP can partially alleviate the negative effects of short cycles in the Tanner graph, its impact remains moderate. Thus far, to the best of the authors' knowledge, no model-based implementations display close-to-optimal performances for BCH and polar codes.

The *model-free* approach —referred to as Syndrome-Based Neural Decoding (SBND)— is harder to interpret but has shown competitive performances when applied to Polar or BCH codes, and approaches optimal error rates in some scenarios. However, the work in [17] presented some limitations that can be enhanced to improve its performance. In a previous work by the authors [23], we proposed a system that makes use of a pseudo-inverse of the generator matrix to train the system by only focusing on the information bits of the codeword. This led to an improvement in Bit Error Rate (BER) and Frame Error Rate (FER) performance and rendered the proposed decoder directly applicable to non-systematic codes. Nonetheless, both the proposed decoder in [23] and the work of [17] rely on the assumption of a BPSK modulation to derive sufficient statistics and prove the *single-codeword training property*. Thus, they cannot be straightforwardly applied to higher-order modulations. For this reason, in a subsequent work by the authors [24], we made a proof of concept of the extension to higher order modulations in a Bit-Interleaved Coded Modulation (BICM) framework, where the decoder from [23] was generalized to two higher-order modulations, namely the 8-PSK and the 16-QAM constellations with Gray symbol labeling. However, the proof of optimality of the suggested decoder as well as the training dataset analysis is valid only under high SNR assumptions, and only for 8-PSK and 16-QAM constellations with Gray labeling.

In this work, we generalize the construction of SBND for BICM with arbitrary constellation shapes, sizes, symbol labelings, and SNR. Our contributions are listed as follows.

(i) We generalize the optimality proof of the SBND for BICM irrespective of the constellation and SNR assumptions. The proof follows from introducing a scrambling operation that helps symmetrize the BICM channel. This, in turn, allows us to identify the absolute values (reliabilities) and syndromes of the Log-Likelihood Ratios (LLRs) of the demodulator as sufficient statistics for decoding, proving hence the optimality of the proposed decoding approach.

(ii) With the introduction of scrambling, we show that, unlike in our earlier contribution on the model-free approach for BICM [24], it is still feasible to train on a single codeword, thereby greatly simplifying the construction and analysis of the training dataset.

(iii) To improve the performance of the basic SBND decoder for symbol labelings other than Gray—e.g., *set-partitionning* (SP) and *natural* labelings—, we introduce and build an iterative decoder for BICM, namely I-SBND, which is a modified output SBND that allows to exploit the benefits of iterative demodulation and decoding for such symbol labelings, similarly to [25], [26].

(iv) From a deep-learning perspective, we implement the proposed SBND using Recurrent Neural Networks (RNN) as in [17] and the transformer-based architecture from [18], and propose a recurrent version of the latter that reduces the number of parameters in the network to only a fraction of those of previous solutions, without compromising its performance.

(v) We provide a complexity analysis in terms of the total number of weights and show that it displays similar performances compared to the other architectures, with only 10% of the number of weights in the transformer architecture of [18] and 1% to 5% the number of weights of the RNN [17], depending on the code.

(vi) Finally, we integrate all the examined components and evaluate the decoding performance of all the decoders through various polar and BCH Codes. We demonstrate that our solution presents, arguably, the best complexity-performance trade-off in the literature of SBND decoders.

Structure: The remainder of this work is organized as follows. Section II introduces the channel decoding problem for BICM and establishes the equivalent BICM channel. Section III describes our proposed decoder (SBND), proves its optimality, and discusses the training dataset design. Section IV describes the construction of an iterative decoder (I-SBND) from a modification of the output of our proposed decoder. Section V lists existing neural-based architectures, describes the proposed lightweight recurrent architecture, and gives the complexity analysis of the different solutions. Numerical results are given in Section VI for both the non-iterative and the iterative decoders. Section VII concludes the work.

Notations: Capital italic letters (e.g. X and $\boldsymbol{X}$) represent random variables and vectors whereas Roman and bold letters (e.g. x and $\boldsymbol{x}$) denote their respective realizations. Matrices are represented by non-italic capital letters (e.g. H) and I_n denotes the $n \times n$ identity matrix. The Hadamard product between vectors is represented by $\odot$. Sets are denoted by calligraphic letters $\mathcal{X}$, and for finite sets, $|\mathcal{X}|$ denotes the cardinality. $P_X(x)$ (resp. $P_{X|Y}(x|y)$) represents the probability distribution (resp. conditional) evaluated in x (resp. (x, y)). Markov chains are denoted $X \leftrightarrow Y \leftrightarrow Z$ to mean $P_{Z|Y,X} = P_{Z|Y}$. $\mathbb{P}(X = x)$ represents the probability of the event $\{X = x\}$, i.e. the

random variable X taking the value x. $\mathbb{1}(e)$ denotes the indicator of the event $\{e\}$, that takes the value 1 if and only if the event $\{e\}$ is true, and 0 otherwise. $[1:n]$ denotes the set of integers from 1 to n and $\lceil\cdot\rceil$ represents the ceiling function. The xor operation is denoted $\oplus$.

II. Preliminaries and problem statement

A. System model and optimal decoder

A random source generates binary data in frames of length k —henceforth referred to as *message* and noted by $\boldsymbol{u}$—, assumed independent and uniformly distributed. This message is then mapped to an n-bit codeword $\boldsymbol{c}$ through an (n,k) linear block code defined by a generator matrix G of size $k\times n$, such that $\boldsymbol{c}=\mathrm{G}^T\boldsymbol{u}$. The coded bits $\boldsymbol{c}$ are then scrambled using a uniform binary vector $\boldsymbol{v}$ to yield $\bar{\boldsymbol{c}}=\boldsymbol{v}\oplus\boldsymbol{c}$[1]. In the BICM scenario, M codewords $\{\bar{\boldsymbol{c}}_i\}_{i=1,\dots,M}$ are grouped into a block, which gets subsequently shuffled by a bit-interleaver, yielding M interleaved codewords $\{\tilde{\boldsymbol{c}}_i\}_{i=1,\dots,M}$. Each codeword is finally mapped through a unit-norm linear modulation scheme of order m, such as PSK or QAM, that yields a modulated codeword $\boldsymbol{x}\in\mathbb{C}^{n'}$ of length $n'=n/m$. The modulated codeword is transmitted through a channel, modeled as an AWGN $\boldsymbol{w}\sim\mathcal{CN}(\boldsymbol{0},\sigma^2\mathrm{I}_{n'})$, such that the received signal writes as $\boldsymbol{y}=\boldsymbol{x}+\boldsymbol{w}$. On reception, a demodulator computes for each received symbol y_j, where $j\in[1:n']$, the LLRs of the m corresponding coded bits given by [2]

$$\tilde{l}_s\triangleq\log\left(\frac{P_{Y|\tilde{C}_s}(y_j|0)}{P_{Y|\tilde{C}_s}(y_j|1)}\right),\ \text{for all } s\in[1:m] \tag{1}$$

where $P_{Y|\tilde{C}_s}$ is the marginal channel relating the s-th coded bit to the received symbol. The shuffled LLR vector $\tilde{\boldsymbol{l}}$ (of size n) is then de-interleaved to obtain $\bar{\boldsymbol{l}}$, which are then de-scrambled by multiplying by $(-1)^{\boldsymbol{v}}$ to yield the LLRs $\boldsymbol{l}$. The obtained LLR vector $\boldsymbol{l}=(l_1,l_2,\dots,l_n)$ is in the original bit order and is fed to the decoder to return an estimation $\hat{\boldsymbol{u}}=\boldsymbol{g}(\boldsymbol{l})$ of the originally transmitted message.

The decoding problem consists in designing a decoder $\boldsymbol{g}(\cdot)$ to minimize either the Bit Error Probability (BEP) or the Frame Error Probability (FEP) defined respectively by

$$P_e^b(\boldsymbol{g})\triangleq\frac{1}{k}\sum_{i=1}^{k}\mathbb{P}(\hat{U}_i\neq U_i)=\frac{1}{k}\sum_{i=1}^{k}\mathbb{P}(g_i(\boldsymbol{L})\neq U_i). \tag{2}$$

$$P_e(\boldsymbol{g})\triangleq\mathbb{P}(\hat{\boldsymbol{U}}^k\neq\boldsymbol{U}^k)=\mathbb{P}(\boldsymbol{g}(\boldsymbol{L})\neq\boldsymbol{U}^k). \tag{3}$$

The BEP is minimized by the so-called bit-Maximum A Posteriori (bit-MAP) decoding rule $\boldsymbol{g}^\star(\cdot)$ defined by

$$u_i^\star=g_i^\star(\boldsymbol{l})\triangleq\underset{u\in\{0,1\}}{\operatorname{argmax}}\ P_{U_i|\boldsymbol{L}}(u|\boldsymbol{l})\ \forall i\in[1:k], \tag{4}$$

while the FEP is minimized by the sequence MAP decoding rule defined by

$$\boldsymbol{u}^{k\star}=\boldsymbol{g}^\star(\boldsymbol{l})\triangleq\underset{\boldsymbol{u}^k\in\{0,1\}^k}{\operatorname{argmax}}\ P_{\boldsymbol{U}^k|\boldsymbol{L}}(\boldsymbol{u}^k|\boldsymbol{l}). \tag{5}$$

[1]The scrambling has no effect on the performance of the system but allows to symmetrize the BICM channel as noted in [27], [28].

[2]For ease of notations, we choose to remove the dependency of the LLRs on the scrambling sequence $\boldsymbol{v}$ throughout the manuscript since it is implicit.

The computation of the bit-MAP decoding rule requires the computation of the k posterior distributions $P_{U_i|\boldsymbol{L}}$ for all $i\in[1:k]$, entailing each one a marginalization over 2^{k-1} sequences $(u_1,\dots,u_{i-1},u_{i+1},\dots,u_k)$. Similarly, the sequence MAP decoder requires exploring 2^k sequences. Both decoders engender, thus, exponential complexity which renders MAP decoding too complex for real-time implementation even for short codes, and simply intractable for larger codes. In this work, we undertake the design of quasi-optimal and low-complexity decoding rules $\boldsymbol{g}(\cdot)$ which are based on deep learning techniques.

B. BICM equivalent channel model

In order to derive optimal decoders for the BICM scenario under investigation, let us characterize the effective channel relating each of the transmitted codewords $\boldsymbol{c}$ to the received LLR $\boldsymbol{l}$, namely the BICM equivalent channel model. Using results from the literature [28], [27], the BICM channel depicted in Figure 1 can be characterized as a *memoryless* channel where each coded bit c sees a mixture of the channels induced by all bit-positions $s\in[1:m]$. More formally, the equivalent BICM channel is given in the following lemma.

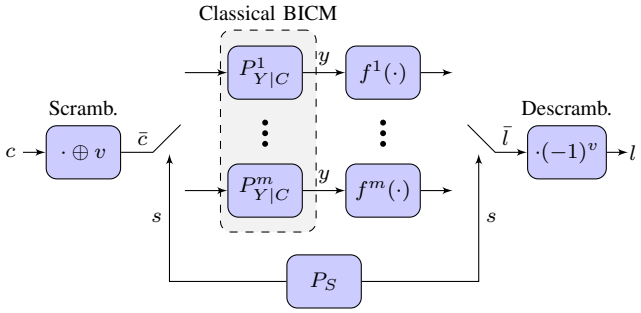

Fig. 1. BICM channel model extended to bit-LLRs outputs and scrambling.

Lemma 1. *The equivalent inner BICM channel $P_{\bar{\boldsymbol{L}}|\bar{\boldsymbol{C}}}$ is memoryless, i.e., for all $\bar{\boldsymbol{l}}\in\mathbb{R}^n$, and $\bar{\boldsymbol{c}}\in\{0,1\}^n$ as*

$$P_{\bar{\boldsymbol{L}}|\bar{\boldsymbol{C}}}(\bar{\boldsymbol{l}}|\bar{\boldsymbol{c}})=\prod_{i=1}^{n}P_{\bar{L}|\bar{C}}(\bar{l}_i|\bar{c}_i), \tag{6}$$

where, the distribution $P_{\bar{L}|\bar{C}}$ is given by

$$P_{\bar{L}|\bar{C}}(\bar{l}|\bar{c})=\frac{1}{m|\mathcal{X}_{\bar{c}}^s|}\sum_{s=1}^{m}\sum_{x\in\mathcal{X}_{\bar{c}}^s}P_{\bar{L}_s|X}(\bar{l}|x), \tag{7}$$

and where $\mathcal{X}_{\bar{c}}^s$ is the set of symbols for which their s-th bit equals $\bar{c}$ (0 or 1). The distribution $P_{\bar{L}_s|X}$ can be easily obtained from $P_{Y|X}$, $\bar{L}_s$ being a deterministic function of Y. Consequently, the overall BICM channel with scrambling is memoryless, i.e., for all $\boldsymbol{l}\in\mathbb{R}^n$, and $\boldsymbol{c}\in\{0,1\}^n$

$$P_{\boldsymbol{L}|\boldsymbol{C}}(\boldsymbol{l}|\boldsymbol{c})=\prod_{i=1}^{n}P_{L|C}(l_i|c_i), \tag{8}$$

where the distribution $P_{L|C}$ is given by

$$P_{L|C}(l|c)=\frac{1}{2}P_{\bar{L}|\bar{C}}(l|c)+\frac{1}{2}P_{\bar{L}|\bar{C}}(-l|1-c), \tag{9}$$

$$= \frac{1}{2m|\mathcal{X}_c^s|} \sum_{s=1}^{m} \left[\sum_{x \in \mathcal{X}_c^s} P_{\bar{L}_s|X}(l|x) + \sum_{x \in \mathcal{X}_{1-c}^s} P_{\bar{L}_s|X}(-l|x) \right] \quad (10)$$

Proof. This result follows from [27, Section 3.4] and [28]. □

In this work, we will not seek a closed-form expression of the BICM equivalent channel distribution $P_{L|C}$. However, the equivalent channel model will prove useful to analyze the optimality of our proposed decoder in Section III-D.

III. Model-free decoding for BICM

In the following, we propose a model-free decoder for higher-order modulations under a BICM scenario. To this end, we start by introducing the principle of model-free decoding, then we present the structure of the proposed decoder, and compare it to the existing state of the art.

A. Model-free decoding for BPSK and QPSK

The model-free decoder introduced in [17] consists in two main ideas, namely i) *estimating bitflips* in the received noisy codewords and ii) identifying *sufficient statistics* to do so.

To this end, a pre-processing stage is added before the decoder, where the received BPSK signal $\boldsymbol{y}$ is divided into its absolute value $|\boldsymbol{y}|$ and its syndrome $\boldsymbol{s}$, defined as $\boldsymbol{s} = \mathrm{H}\boldsymbol{y}^b$, where H denotes the parity-check matrix of the code, $\boldsymbol{y}^b$ represents the hard-decision associated with the received signal $\boldsymbol{y}$. Authors in [17] proved that the two elements $(\mathrm{H}\boldsymbol{y}^b, |\boldsymbol{y}|)$ are sufficient statistics for the optimal estimation of the bitflip pattern $\boldsymbol{e}^b$ where $\boldsymbol{e}^b \triangleq \boldsymbol{c} \oplus \boldsymbol{y}^b$ is a vector containing ones in the flipped positions and zeros everywhere else. The authors in [17] showed that, for a BPSK modulation, the vector $(\mathrm{H}\boldsymbol{y}^b, |\boldsymbol{y}|)$ is independent of the codeword $\boldsymbol{c}$, hence, training can be carried out using noisy observations of a single codeword. Thus, this decoder offers great scalability properties as compared to previous model-free decoders [7].

Although originally designed for a BPSK modulation, the extension of the results of [17] to a QPSK scenario can be done readily by decomposing $\boldsymbol{y}$ into its real and imaginary parts $(\mathrm{Re}(\boldsymbol{y}), \mathrm{Im}(\boldsymbol{y}))$, and then extracting the absolute value and syndrome associated with these two components.

B. Model-free decoding for higher-order BICM

In what follows, we describe our proposed model-free SBND for higher-order modulations of Figure 2.

(i) The deinterleaved LLR sequence $\boldsymbol{l}$ delivered by the demodulator undergoes a pre-processing stage that extracts the syndrome of its associated hard decisions (i.e., $\mathrm{H}\boldsymbol{l}^b$), and the *reliabilities* (i.e., $|\boldsymbol{l}|$).

(ii) A primary estimate of the message $\tilde{\boldsymbol{u}}$ is then obtained through the pseudo-inverse of the hard decisions $\boldsymbol{l}^b$, i.e., $\tilde{\boldsymbol{u}} \triangleq \mathrm{p}_{\mathrm{inv}}(\boldsymbol{l}^b)$, where $\mathrm{p}_{\mathrm{inv}}(\cdot)$ corresponds to a pseudo-inverse of the code $\mathcal{C}$ such that, for every valid codeword $\boldsymbol{c} = \mathrm{G}^T\boldsymbol{u}$, it holds that $\mathrm{p}_{\mathrm{inv}}(\boldsymbol{c}) = \boldsymbol{u}$. If a realization of the LLR vector has suffered a sign change, i.e., $\boldsymbol{l}^b \neq \boldsymbol{c}$, then the auxiliary primary estimate $\tilde{\boldsymbol{u}}$ might also contain errors. Besides, in the special case of a systematic code, a pseudo-inverse corresponds to extracting the systematic bits, i.e., $\mathrm{p}_{\mathrm{inv}}(\boldsymbol{c}) = \mathrm{A}\boldsymbol{c}$ where $\mathrm{A} = \left[\mathrm{I}_k \,|\, \mathbf{0}_{k,n-k}\right]^T$.

(iii) The two inputs $(|\boldsymbol{l}|, \mathrm{H}\boldsymbol{l}^b)$ are fed to the message bitflip estimator, which outputs a vector $\hat{\boldsymbol{e}}_{\boldsymbol{u}}$ with positive values in positions where a bitflip occurred in $\tilde{\boldsymbol{u}}$ and negative values otherwise. The $\mathrm{bin}(\cdot)$ operation converts the real values into binary values $\hat{\boldsymbol{e}}_{\boldsymbol{u}}^b \triangleq \mathbb{1}\big(\hat{e}_{u,i} > 0\big)_{1 \leq i \leq k}$.

(iv) Finally, $\tilde{\boldsymbol{u}}$ is obtained by letting $\hat{\boldsymbol{u}} = \tilde{\boldsymbol{u}} \oplus \hat{\boldsymbol{e}}_{\boldsymbol{u}}^b$.

This decoding layout is a *message-oriented* SBND, and adapted for higher-order modulations. The design and training of the message bitflip estimator will be tackled in the upcoming sections. Let us observe that the proposed decoder is based

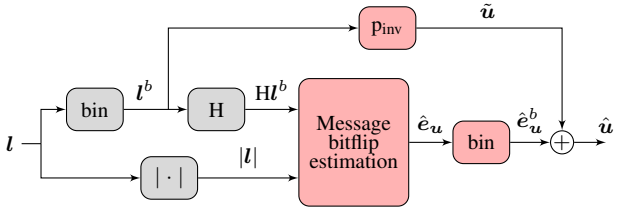

Fig. 2. Proposed architecture for the SBND. The pre-processing stage applied to the LLR vector (in gray) is followed by a decoding stage (in red).

on a *denoising* approach, where the estimator does not output the estimated message $\hat{\boldsymbol{u}}$ directly, but rather an estimate of the *message bitflip* $\boldsymbol{e}_{\boldsymbol{u}}^b$, which is subsequently used to correct the intial hard estimate $\tilde{\boldsymbol{u}}$. Nonetheless, this approach is equivalent to MAP decoding of the message $\boldsymbol{u}$, since we have

$$\underset{\boldsymbol{u} \in \{0,1\}^k}{\mathrm{argmax}}\, P_{\boldsymbol{U}|\boldsymbol{L}}(\boldsymbol{u}|\boldsymbol{l}) = \underset{\boldsymbol{e}_{\boldsymbol{u}}^b \in \{0,1\}^k}{\mathrm{argmax}}\, P_{\tilde{\boldsymbol{U}} \oplus \boldsymbol{E}_{\boldsymbol{u}}^b|\boldsymbol{L}}(\tilde{\boldsymbol{u}} \oplus \boldsymbol{e}_{\boldsymbol{u}}^b|\boldsymbol{l}) \quad (11)$$

$$= \tilde{\boldsymbol{u}} \oplus \underset{\boldsymbol{e}_{\boldsymbol{u}}^b \in \{0,1\}^k}{\mathrm{argmax}}\, P_{\boldsymbol{E}_{\boldsymbol{u}}^b|\boldsymbol{L}}(\boldsymbol{e}_{\boldsymbol{u}}^b|\boldsymbol{l}), \quad (12)$$

where we have used that $\boldsymbol{u} = \tilde{\boldsymbol{u}} \oplus \boldsymbol{e}_{\boldsymbol{u}}^b$, along with the fact that $\tilde{\boldsymbol{u}}$ is a deterministic function of $\boldsymbol{l}$ (pseudo-inverse). Similarly,

$$\underset{u_i \in \{0,1\}}{\mathrm{argmax}}\, P_{U_i|\boldsymbol{L}}(u_i|\boldsymbol{l}) = \tilde{u}_i \oplus \underset{e_{u,i}^b \in \{0,1\}}{\mathrm{argmax}}\, P_{E_{u,i}^b|\boldsymbol{L}}(e_{u,i}^b|\boldsymbol{l}), \quad (13)$$

hence the equivalence of MAP decoding and MAP denoising.

C. Comparison with related works

The herebefore proposed decoder improves on that of [17] and [23] in three different aspects. First, it exploits a previous improvement suggested by the authors in [23] seeking to estimate the bitflips induced by the channel on the message bits $\boldsymbol{u}$ instead of on the whole codeword $\boldsymbol{c}$. The improvements entailed by this feature are two-fold. On the one side, since the objective function to be optimized is the BEP defined as per (2), restricting the loss function to the message $\boldsymbol{u}$ instead of the whole codeword $\boldsymbol{c}$ allows to disregard error events on the $n-k$ parity bits and reduces the decision space dimension from n to k. On the other side, the present decoder structure discards error events in which the obtained *denoised* codeword is not a valid codeword, i.e., does not belong to the code.

Second, the proposed decoder is directly applicable to non-systematic codes, without the need to apply a linear transformation to obtain the estimated message $\hat{\boldsymbol{u}}$ from the estimated codeword $\hat{\boldsymbol{c}}$. The pseudo-inverse introduced in the

proposed decoder is such that the neural network estimates the bitflips present in an artificial variable that has already addressed the codeword-to-message operation.

Third, the proposed decoder generalizes the structures of [17], [18], [23] from a BPSK-specific decoding system to a decoder suited for arbitrary higher-order modulations by considering the LLR $\boldsymbol{l}$ instead of the received noisy symbols $\boldsymbol{y}$. As we show in section III-D, the proposed decoder can achieve optimal decoding for the BICM scenario in the sense that it can approximate the MAP decoder. However, the theoretical analysis of the proposed decoder is more involved than in the case of BPSK as described in [17] and extended in [23].

D. On the optimality of SBND for BICM

In the following, we show that, as in the case of BPSK and QPSK, the SBND inputs $(|\boldsymbol{l}|, \mathrm{H}\boldsymbol{l}^b)$ constitute sufficient statistics for the estimation of the bitflip pattern $\boldsymbol{e}_{\boldsymbol{u}}^b$, thereby confirming the optimality of the proposed decoder structure. Let us first state the following result which allows to characterize the channel distribution $P_{\boldsymbol{L}^b|\boldsymbol{C}}$.

Theorem 1 (Bit-LLRs binary channel model)**.** *For all $\boldsymbol{l}^b, \boldsymbol{c} \in \{0,1\}^n$, the following holds:*

$$P_{\boldsymbol{L}^b|\boldsymbol{C}}(\boldsymbol{l}^b|\boldsymbol{c}) = \prod_{i=1}^{n} P_{L^b|C}(l_i^b|c_i). \tag{14}$$

Besides, the binary channel $P_{\boldsymbol{L}^b|\boldsymbol{C}}$ can be written as

$$\boldsymbol{L}^b = \boldsymbol{C} \oplus \boldsymbol{E}^b \quad s.t. \quad \boldsymbol{E}^b \overset{i.i.d}{\sim} Bern(q), \tag{15}$$

where $\boldsymbol{E}^b$ is independent of $\boldsymbol{C}$ and

$$q \triangleq \frac{1}{2m} \sum_{s=1}^{m} \left(P_{\bar{L}_s^b|\bar{C}}(1|0) + P_{\bar{L}_s^b|\bar{C}}(0|1) \right). \tag{16}$$

Proof. The detailed proof is relegated to Appendix A. □

It is worth noting that, due to the symmetry of the BICM channel obtained by scrambling, the present result holds irrespective of the modulation scheme (QAM, PSK, APSK), to and modulation order m, for all SNRs, and irrespective to the mapping used (binary, gray, set-partiotionning, random). In this regard, Theorem 1 generalizes the authors' previous result in [24] which was valid only 8PSK and 16QAM under Gray mapping and high SNR assumption.

Having characterized the binary channel relating $\boldsymbol{C}$ to $\boldsymbol{L}^b$ as a memoryless Binary Symmetric Channel (BSC), we state the main result that proves the optimality of the proposed decoder.

Theorem 2 (Sufficient statistics)**.** *Considering a BICM scenario and the result of Theorem 1, then for all $\boldsymbol{e}_{\boldsymbol{u}}^b \in \{0,1\}^k$ and $\boldsymbol{l} \in \mathbb{R}^n$, we have that*

$$P_{\boldsymbol{E}_{\boldsymbol{u}}^b|\boldsymbol{L}}(\boldsymbol{e}_{\boldsymbol{u}}^b|\boldsymbol{l}) = P_{\boldsymbol{E}_{\boldsymbol{u}}^b||\boldsymbol{L}|,\mathrm{H}\boldsymbol{L}^b}(\boldsymbol{e}_{\boldsymbol{u}}^b|\,|\boldsymbol{l}|, \mathrm{H}\boldsymbol{l}^b). \tag{17}$$

Hence, it follows that

$$P_{E_{u,i}^b|\boldsymbol{L}}(e_{u,i}^b|\boldsymbol{l}) = P_{E_{u,i}^b||\boldsymbol{L}|,\mathrm{H}\boldsymbol{L}^b}(e_{u,i}^b|\,|\boldsymbol{l}|, \mathrm{H}\boldsymbol{l}^b). \tag{18}$$

Proof. The proof is relegated to Appendix B. □

This result allows us to state that the posterior distribution of the message bitflips $\boldsymbol{e}_{\boldsymbol{u}}^b$ given the received LLR $\boldsymbol{l}$ is indeed equal to the posterior distribution of the bitflips $\boldsymbol{e}_{\boldsymbol{u}}^b$ given only the reliabilities and syndrome $(|\boldsymbol{l}|, \mathrm{H}\boldsymbol{l}^b)$. As such, the quantities $(|\boldsymbol{l}|, \mathrm{H}\boldsymbol{l}^b)$ offer sufficient statistics for estimating $\boldsymbol{e}_{\boldsymbol{u}}^b$. Thus, provided that the message bitflip estimator yields a good approximation of the posterior distribution $P_{\boldsymbol{E}_{\boldsymbol{u}}^b||\boldsymbol{L}|,\mathrm{H}\boldsymbol{L}^b}$, the proposed decoder will be close to the optimal.

E. Training set design

The performance of the proposed decoder depends on the capacity of the message bitflip estimator in Figure 2 to closely approximate the posterior distribution $P_{\boldsymbol{E}_{\boldsymbol{u}}^b||\boldsymbol{L}|,\mathrm{H}\boldsymbol{L}^b}$ (or rather its argmax). In this work, we choose to implement the message bitflip estimator using neural networks since they are well-known and powerful universal approximators [29]. The training loss is set to be the *binary cross-entropy*, which will further ensure that the soft output of the neural network converges to the desired posterior distributions, provided that the training dataset is well designed to prevent overfitting.

Let us now analyze the design of the training dataset. A key implication of the result of Theorem 2 when applied to the case of low-order constellations, namely BPSK and QPSK, is that the quantities $|\boldsymbol{l}|$ and $\mathrm{H}\boldsymbol{l}^b$ are invariant to the transmitted codewords $\boldsymbol{c}$, reducing thus the training set of codewords to only one codeword $\boldsymbol{c}$ (for instance the all-zero codeword) as shown in [17] and [23]. This feature is key to the scaling capability of syndrome-based decoders for BPSK and QPSK since it alleviates the necessity to train over the whole set of all 2^k possible codewords.

For higher-order modulations, while the syndrome $\mathrm{H}\boldsymbol{l}^b$ is independent of the transmitted codeword $\boldsymbol{c}$ in a BICM scenario, the reliabilities $|\boldsymbol{l}|$ are not —they depend on the transmitted symbols $\boldsymbol{x}$. In the authors' previous contribution [24], the proposed decoder could not be trained on only *the all-zero codeword* since this always leads to the same transmitted symbols $\boldsymbol{x}$, irrespective of the interleaving. In this work, since we are resorting to scrambling with the sequence $\boldsymbol{v}$, the transmitted symbols are varied and allow us to explore a vast number of combinations of LLRs even if we transmit only the the all-zero codeword. This leads to a sensible simplification of the training process, and as will be discussed in the numerical analysis section, the vast number is but negligible as compared to the total number of codewords of the code 2^k.

IV. Iterative demodulation and model-free decoding: I-SBND

Iterative decoding plays a crucial role in improving the performance of BICM receivers, especially for symbol mappings other than Gray mapping [25], [26]. Inspired by the construction of iterative receivers based on classical decoders, we propose an iterative decoder for BICM which is based on a MAP-demodulator and a modified SBND structure which we name Iterative SBND (I-SBND). In the following, we start by presenting the receiver structure, we recall the MAP-demodulator formulae, and describe the proposed I-SBND structure.

A. Iterative demodulation and decoding based on SBND

The proposed receiver based on iterative demodulation and decoding of Figure 3 can be described as follows. The

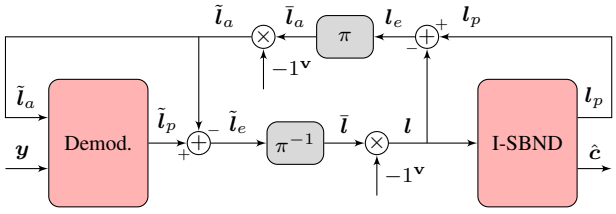

Fig. 3. Proposed architecture for iterative demodulation and decoding.

demodulator receives as input the noisy symbols $\boldsymbol{y}$ and the prior LLRs on the interleaved coded bits $\tilde{\boldsymbol{l}}_a$, defined by

$$\tilde{l}_{a,i} \triangleq \log\left(\frac{P_{\tilde{C}_i}(0)}{P_{\tilde{C}_i}(1)}\right) \text{ for all } i \in [1:n]. \tag{19}$$

It then computes for each received symbol y the posterior LLRs of the m interleaved coded bits associated with it

$$\tilde{l}_{p,s} \triangleq \log\left(\frac{P_{\tilde{C}_s|Y}(0|y)}{P_{\tilde{C}_s|Y}(1|y)}\right), \text{ for all } s \in [1:m]. \tag{20}$$

The extrinsic LLRs of the demodulator $\tilde{\boldsymbol{l}}_e$, where $\tilde{\boldsymbol{l}}_e \triangleq \tilde{\boldsymbol{l}}_p - \tilde{\boldsymbol{l}}_a$ are then de-interleaved, descrambled, and fed to the decoder (I-SBND) to serve as prior LLRs on the coded bits $\boldsymbol{c}$, i.e., $\boldsymbol{l} \triangleq (-1)^{\boldsymbol{v}}\pi^{-1}(\tilde{\boldsymbol{l}}_e)$. When the I-SBND is well designed and trained, it provides the posterior distribution of the coded bits $\boldsymbol{l}_p$, i.e.,

$$l_{p,i} \triangleq \log\left(\frac{P_{C_i|\boldsymbol{Y}}(0|\boldsymbol{y})}{P_{C_i|\boldsymbol{Y}}(1|\boldsymbol{y})}\right), \text{ for all } i \in [1:n] \tag{21}$$

which are then transformed into extrinsic LLRs $\boldsymbol{l}_e \triangleq \boldsymbol{l}_p - \boldsymbol{l}$ on the coded bits. The interleaved and scrambled extrinsic LLRs serve as priors for the demodulator for the next iteration, i.e., $\tilde{\boldsymbol{l}}_a \triangleq (-1)^{\boldsymbol{v}}\pi(\boldsymbol{l}_e)$.

B. Prior-based symbol demodulation

In the following, we describe analytically the demodulator of Figure 4 under the hypothesis of an AWGN channel.

Lemma 2 (Demodulator extrinsic LLRs)**.** *For a received symbol y, the m extrinsic LLRs at the output of the demodulator are given, for all $s \in [1:m]$, by*

$$\tilde{l}_{e,s} = \log \sum_{\substack{\tilde{\boldsymbol{c}}\in\{0,1\}^m \\ \tilde{c}_s=0}} \exp\left[-\frac{1}{\sigma^2}|y - x(\tilde{\boldsymbol{c}})|^2 + \sum_{\substack{v=1 \\ v\neq s}}^{m}(1-\tilde{c}_v)\tilde{l}_{a,v}\right]$$

$$- \log \sum_{\substack{\tilde{\boldsymbol{c}}\in\{0,1\}^m \\ \tilde{c}_s=1}} \exp\left[-\frac{1}{\sigma^2}|y - x(\tilde{\boldsymbol{c}})|^2 + \sum_{\substack{v=1 \\ v\neq s}}^{m}(1-\tilde{c}_v)\tilde{l}_{a,v}\right], \tag{22}$$

where $x(\tilde{\boldsymbol{c}})$ is the symbol associated to the m bits $\tilde{\boldsymbol{c}}$.

Proof. Although classical in iterative decoding [25], [30], a proof is given in Appendix C for self-containedness. □

The implementation of this demodulator can be further simplified by producing only *approximate LLRs*, or by using the *soft-max* operator.

C. Proposed I-SBND structures

To build an interative demodulator and decoder based on SBND, we suggest a modification of the output of the SBND in two aspects, as shown in Figure 4. First, we modify the

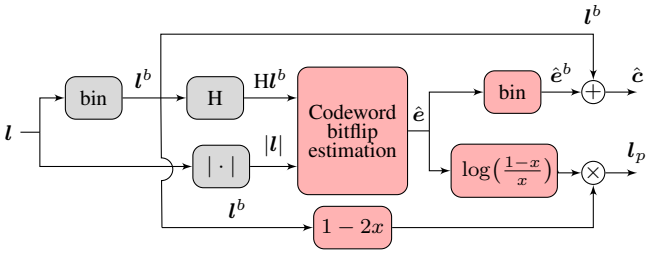

Fig. 4. Proposed structure for I-SBND.

SBND to recover bitflip estimates of the whole codeword $\boldsymbol{c}$, and not of the message $\boldsymbol{u}$, recovering the original scheme of [17]. Second, we modify the output of the SBND to additionally obtain posterior LLRs on the coded bits from the bitflip soft estimates. More specifically, let $\boldsymbol{y}$ be the received symbols, let $\boldsymbol{l}$ be the LLRs at the input of the decoder, and let $\boldsymbol{e}^b$ be the bitflip pattern of the coded bits, i.e.,

$$\boldsymbol{e}^b \triangleq \boldsymbol{l}^b \oplus \boldsymbol{c}, \tag{23}$$

where $\boldsymbol{l}^b$ are the hard decisions associated with the LLRs $\boldsymbol{l}$. The output of the codeword bitflip estimator consist in soft estimates $\hat{\boldsymbol{e}}$ of the bitflip pattern $\boldsymbol{e}^b$, which approximate the posterior distribution given the input of the decoder, i.e.,

$$\hat{e}_i \triangleq P_{E_i^b|\boldsymbol{L}}(1|\boldsymbol{l}) = P_{E_i^b||\boldsymbol{L}|,\mathrm{H}\boldsymbol{L}^b}(1|\,|\boldsymbol{l}|, \mathrm{H}\boldsymbol{l}^b), \tag{24}$$

which follows by using a similar proof to that of Theorem 2. The posterior LLRs on the coded bits can then be obtained from the soft estimates $\hat{\boldsymbol{e}}$ as follows.

Lemma 3 (Coded bits posterior LLRs)**.** *For all $i \in [1:n]$,*

$$l_{p,i} \triangleq \log\left(\frac{P_{C_i|\boldsymbol{Y}}(0|\boldsymbol{y})}{P_{C_i|\boldsymbol{Y}}(1|\boldsymbol{y})}\right) = (1-2l_i^b)\log\left(\frac{1-\hat{e}_i}{\hat{e}_i}\right). \tag{25}$$

Proof. The proof is relegated to Appendix D. □

In the numerical results section, we show that the proposed iterative demodulation and decoding improves the performance of the SBND for symbol mappings other than Gray, for which iterating does not bring any gain.

V. NN-based implementations

In what follows, we introduce a novel recurrent version of the attention-based solution in [18] for the bitflip estimator of Section III-B, aiming to reduce even further the number of parameters in the network to start converging toward a feasibly implementable solution. For this, we start by presenting and implementing the two main architectures in the literature [17], [18]. Then, we introduce our lightweight solution and carry out a complexity analysis that compares the number of parameters that configure each architecture. Finally, we evaluate the decoding performances of each of the architectures and compare the different solutions for several Polar and BCH codes.

A. RNN and ECCT architectures

The first architecture considered to implement the bitflip estimator is the RNN using Gated Recurrent Units (GRU) [31] as in [17], [23]. Its structure is depicted in Figure 5. The RNN is constituted of d_l GRU cells stacked on top of each other, each one with a hidden state size of $\alpha(2n-k)$, where $\alpha \in \mathbb{N}^+$ is a scaling hyperparameter to be selected and (n, k) are the parameters of the code. The network performs T recurrent timesteps before giving an output, which is passed through a dense Feed-Forward Neural Network (FFNN) with a linear activation function that outputs the estimated bitflip $\hat{\boldsymbol{e}}_{\boldsymbol{u}}$.

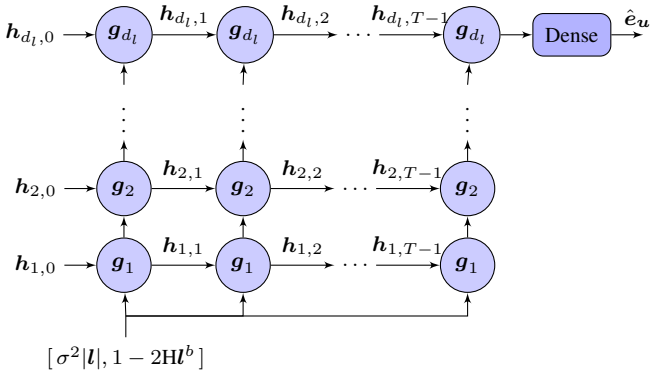

Fig. 5. RNN-based architecture for the noise estimator. The depth of the network is depicted vertically, while its recurrence is unfolded horizontally.

The second architecture we consider is the transformer-based NN proposed by Choukroun et al. in [18] —therein referred to as Error Correction Code Transformer (ECCT)— , inspired by the work of Vaswani et al. [32] and depicted in Figure 6. This network consists of three main stages: (i) an embedding stage, where each component of the input vector is projected into a high-dimensional space of size d_e; (ii) an encoding stage, repeated N times, where a masked Multi-Head Self-Attention (MH-SA) block alternates with two dense FFNNs —an expanding and a contracting one—, with normalization layers added before according to Figure 6; and (iii) a decoder stage, which consists in a normalization layer, a dense layer to squeeze the embedding dimension to 1 and a final dense layer that outputs the estimate $\hat{\boldsymbol{e}}_{\boldsymbol{u}}$. The mask is a symmetric matrix of size $2n-k \times 2n-k$ that contains a one in the position (i, j) if the ith and jth elements of the input vector $[|\boldsymbol{l}|, 1-2\mathrm{H}\boldsymbol{l}^b]$ are *connected*. To achieve this, we start with an identity matrix I_{2n-k}, since every element is connected to itself. Then, for each row $i \in [1 : n-k]$ of the parity-check matrix, for every pair $(j, j') \in [1 : n] \times [1 : n]$ such that $\mathrm{H}_{i,j} = \mathrm{H}_{i,j'} = 1$, we unmask the positions (j, j') and $(n+i, j)$, along with their symmetric elements. The resulting matrix can be seen as an extended adjacency matrix of the Tanner graph, in order to account for all the $2n-k$ input elements. For further details, the reader is referred to [18].

B. Proposed architecture: r-ECCT

It has been stated that a central obstacle in contemporary NN-based channel decoding is the issue of *scalability*. This concept encompasses two key dimensions: (i) the network's capacity to decode larger codes and (ii) its ability to achieve this without resorting to an ever-increasing number of model parameters. The work in [18] tackled this problem, achieving slightly better performances than [17] for a BCH code of size $(127, 64)$ with only a tenth of the number of weights.

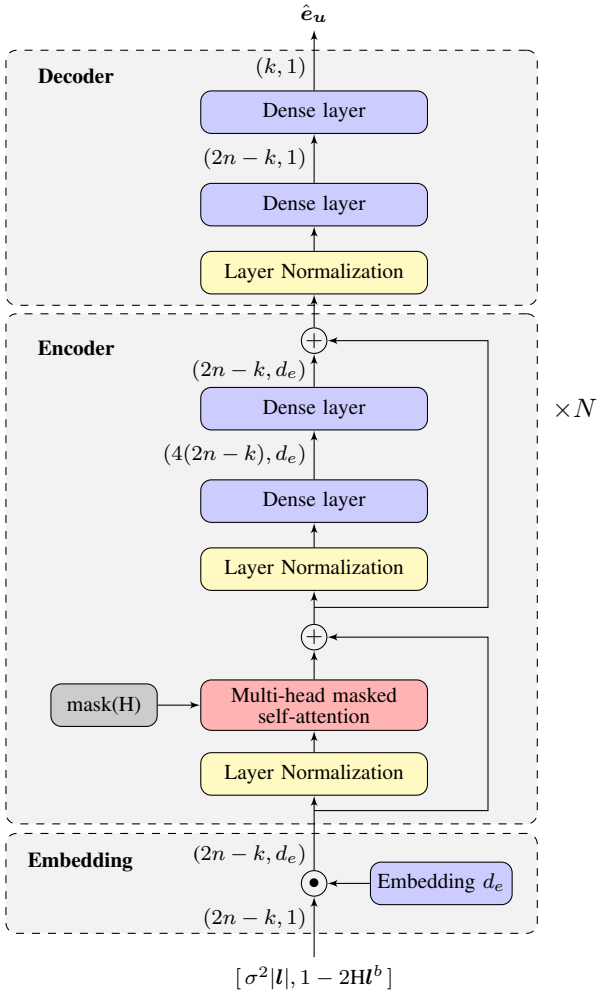

Fig. 6. Transformer-based architecture of [18] for the bitlfip estimator. Relevant vector dimensions are added to the diagram for clarity.

In this section, we propose a recurrent version of the ECCT —henceforth referred to as r-ECCT— that takes a step further in the same direction. As shown in Section V-C, the transformer solution presents a quadratic dependency with the embedding dimension d_e and a linear dependency on the number of concatenated attention blocks N (i.e., encoders). Through simulations, it was observed that a reduction in the embedding dimension often translates into a performance penalty that varies with the size of the code. Therefore, combining the iterative nature of the RNN and the smaller size of the ECCT, we propose removing the linear dependecy on the number of encoders by adopting a recurrent strategy, where the same encoder is employed iteratively for N cycles. The proposed architecture is outlined in Figure 7: after the embedding phase, the data is fed to the encoder block that iterates on itself N times. The resulting encoded vector of size $(2n-k, d_e)$ is fed to the decoder to estimate the bitflip

vector.

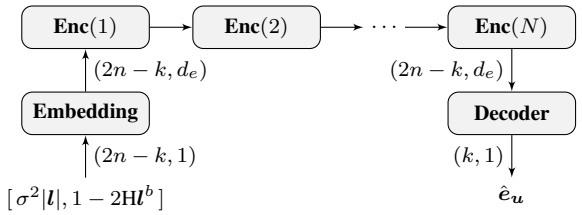

Fig. 7. Recurrent ECCT solution proposed for the bitflip estimator.

C. Complexity analysis

In the following, we characterize analytically the complexity of the three considered NN architectures —i.e., the number of weights— as a function of relevant hyperparameters. For this, we recall the study by the authors [24] and extend it to encompass the new proposed architecture.

1) Recurrent Neural Network: The RNN implemented in our work and in [17] is based on GRU cells. If n_i and n_o represent the dimensions of the input and the output, respectively, we have that the total number of weights for a GRU equals $3(n_0^2 + n_i n_0 + n_0)$ [33]. Moreover, the number of parameters for the final dense layer is given by $n_i n_0 + n_0$. Recalling that each GRU cell consists of $\alpha(2n-k)$ GRUs, and that the network contains d_l GRU cells stacked on top of each other, the total number of weights for the RNN is

$$\mathbb{W}_{\text{RNN}} = \underbrace{3(\alpha^2 r^2 + \alpha r^2 + \alpha r)}_{\text{input GRU layer}} + \underbrace{(d_l - 1)3(\alpha^2 r^2 + \alpha^2 r^2 + \alpha r)}_{d_l - 1 \text{ hidden GRU layers}} + \underbrace{\alpha r k + k}_{\text{dense layer}},$$

where $r \triangleq 2n-k$ is the size of the network's input. Factorizing the expression yields

$$\mathbb{W}_{\text{RNN}} = 3\left((2d_l - 1)\alpha^2 + \alpha\right) r^2 + (3d_l + k)\alpha r + k. \quad (26)$$

It is worth observing that, for a fixed depth d_l and a scaling parameter α, this result implies a network that increases in size predominantly as $\mathcal{O}(r^2)$.

2) Error Correction Code Transformer: The transformer-based architecture is composed of three stages: i) the embedding, which involves rd_e weights, with d_e the embedding dimension; ii) the N encoders, each composed of two normalization layers, an MH-SA, and two dense layers; and iii) the decoder, consisting of a normalization layer and two dense layers. Hence, the number of weights writes as

$$\mathbb{W}_{\text{ECCT}} = N\Big(\underbrace{4d_e}_{\text{norm.}} + \underbrace{4d_e(d_e+1)}_{\text{MH-SA}} + \underbrace{4d_e^2 + 4d_e + 4d_e^2 + d_e}_{\text{encoder dense layers}} \Big) + \underbrace{2d_e}_{\text{norm.}} + \underbrace{d + 1 + rk + k}_{\text{decoder dense layers}} + \underbrace{rd_e}_{\text{embedding}}.$$

By factorization, we get the following expression for the total number of weights in the ECCT:

$$\mathbb{W}_{\text{ECCT}} = 12Nd_e^2 + (13N + r + 3)d_e + (r+1)k + 1, \quad (27)$$

where, for a fixed number of encoders N and an embedding dimension d_e, the value of $\mathbb{W}_{\text{ECCT}}$ is almost independent of the code parameters (n, k).

3) Recurrent ECCT: The proposed solution is rooted in the ECCT architecture, with a novel adaptation involving the recurrent utilization of the encoder for the N iterations. Hence, the number of parameters follows by letting $N = 1$ in (27):

$$\mathbb{W}_{\text{r-ECCT}} = 12d_e^2 + (16 + r)d_e + (r+1)k + 1. \quad (28)$$

The proposed NN is virtually only dependent on the embedding dimension, which is considered fixed in this work. In practice, larger codes will probably require larger embedding dimensions to maintain competitive decoding performances.

In the following, we will show that the RNN, though powerful, relies heavily on a very large number of parameters, and its size grows quickly with the code size. The ECCT removes this dependency on the code size, obtaining competitive performances while keeping a relatively constant number of weights. However, we will see that, with the recurrent approach of the r-ECCT, the size of the ECCT is reduced by a factor of 10, allowing for better computation of the gradient during training and thus obtaining globally better results than the ECCT with considerably lower complexity.

VI. Numerical results

In this section, we present the decoding performance of our proposed decoders from Sections III-B and IV with two different families of codes, namely BCH and polar codes, each with different sizes and code rates[3]. To this end, we implement the three architectures from Section V using TensorFlow [35] and the Keras API [36]. We resort to the *binary cross-entropy* as a training loss function defined for $\boldsymbol{e}$ and $\hat{\boldsymbol{e}}$ as follows:

$$\mathcal{L}_{\text{BCE}}(\boldsymbol{e}, \hat{\boldsymbol{e}}) = \sum_{i=1}^{k} \big(e_i \log(\hat{e}_i) + (1 - e_i)\log(1 - \hat{e}_i)\big), \quad (29)$$

where $\boldsymbol{e}$ is conventionally the true value (or *ground truth*) and $\hat{\boldsymbol{e}}$ denotes the NN-estimated value. The bit-interleaving described in section II-A is carried out in blocks of $M = 512$ codewords. As for the input vector of the NN, i.e., $(|\boldsymbol{l}|, \mathrm{H}\boldsymbol{l}^b)$, we normalize the reliabilities $|\boldsymbol{l}|$ by multiplying them by a factor of σ^2 (see Figures 5, 6 and 7) to render the input range less dependent on the SNR. This normalization operation yields a better generalization capability of the message bitflip estimator for a wide range of SNRs. Regarding the output of the NN, we fix the last dense layer of every architecture to be linear to *mimic* LLR values. As such, a sigmoid function is applied to the estimators' outputs before computing the binary cross-entropy loss. Another possibility would be to select the sigmoid as the final activation function and use a $1/2$-threshold to convert to the binary domain.

A. Choice of hyperparameters

The hyperparameters involved in the NN architectures were selected following different criteria. For the RNN, the network depth and width —d_l and $\alpha(2n-k)$, respectively— were fixed to be generally consistent with the contribution that proposed this architecture [17]. Regarding the ECCT, the authors in [18]

[3]The BCH is extended to $n = 64$ in order to fit the 16-QAM modulation scheme. All the parity-check matrices, codes, and some trained models employed in these simulations can be found in [34].

explore different sets of parameters (N and d_e) and evaluate their performance. For this paper, the set that displayed the best performances was fixed for all codes. For comparative purposes, we kept the embedding dimension d_e for the r-ECCT, and in section VI-B we analyze the impact of the recurrent factor N in the performance, as it does not change the final number of parameters for the proposed solution. For clarity and reproducibility, the architecture and training hyperparameters employed are summarized in Table I.

B. Non-iterative decoding: SBND

In this section, we display the simulation results for the SBND solution described in section III-B. For this non-iterative version, the 16-QAM with Gray mapping is selected for all codes. As a benchmark, the performances of a Successive Cancellation List (SCL) decoder of list size L are added to the polar codes [37], and an OSD of order 2 for the BCH simulations [38]. For the SCL decoder, we used the algorithm of the Sionna library [39], whereas the OSD was implemented by the authors and added to the repository for reproducibility [34]. Observe that, for most codes, training is carried on only a very small fraction of valid codewords (covering merely $10^{-10}\%$ of the valid codeword space for the largest polar code employed), specially for the transformer-based architectures due to their smaller batch size.

Figures 8 and 9 show the FER performances of each considered architecture for several polar codes with $n \in \{64, 128\}$. As observed in previous contributions [17], [18], [23], larger codes with lower code rates are harder to learn, likely due to a larger number of correctable noise patterns that need to be seen during training. Furthermore, for the polar code of size (128, 64) (see Fig. 9), the RNN architecture exhibits reduced adaptability across a wide range of SNRs. Although all networks were trained at $E_b/N_0 = 5$dB, only the RNN shows difficulties in adapting to the high-SNR regime, likely due to the absence of layer normalization between its layers, which is present in the other architectures. Finally, the proposed lightweight architecture r-ECCT displays competitive error rates compared to the previous architectures, while employing only a fraction of the number of weights.

In Figure 10 we display the FER performances of the SBND on two extended BCH codes of length $n = 64$, along with an OSD of order 2 as benchmark. Therein, BCH codes prove themselves harder to learn, with worsening performances as the code rate decreases, an effect that is also documented in the scientific literature on model-free decoders [17], [18], [22]. Finally, Figure 11 analyzes the effect of the number of iterations in the encoding block of the r-ECCT architecture, where the performance improves with the depth of the network until a certain depth, beyond which it stagnates. Recall that the proposed architecture of Section V-B differs in the ECCT from [18] in that the same encoder block is used iteratively, greatly reducing the number of parameters in memory. This empirical study on the optimal number of iterations of the encoding block was not carried out for the other codes, for which it was fixed at $N = 10$. Further fine-tuning could improve performance and reduce latency by selecting the minimum number of iterations that achieve the lowest possible FERs.

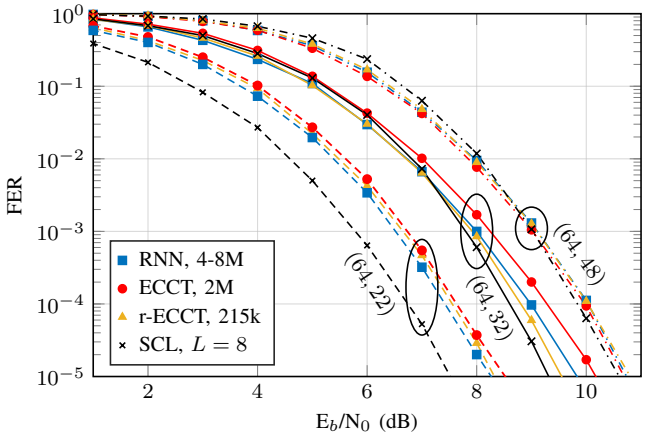

Fig. 8. FER comparison for polar codes $(64, 48)$ (dash-dotted), $(64, 32)$ (solid), and $(64, 22)$ (dashed) under 16-QAM and Gray mapping, for the three considered NN architectures. The approximate number of parameters for each solution is included in the legend.

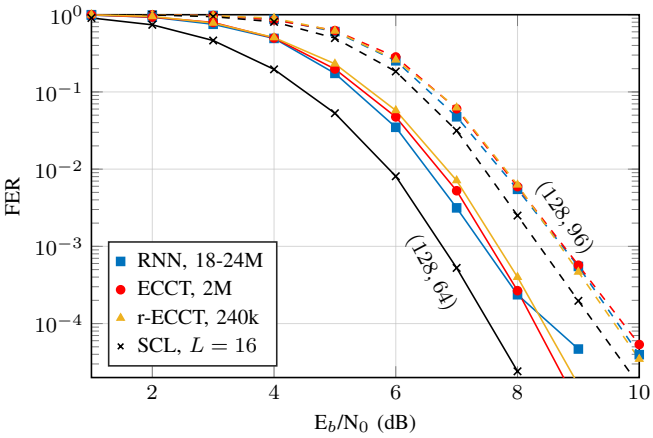

Fig. 9. FER comparison of polar codes $(128, 96)$ (dashed lines) and $(128, 64)$ (solid lines), under 16-QAM and Gray mapping, for the three considered NN architectures. The number of parameters for each solution is included in the legend.

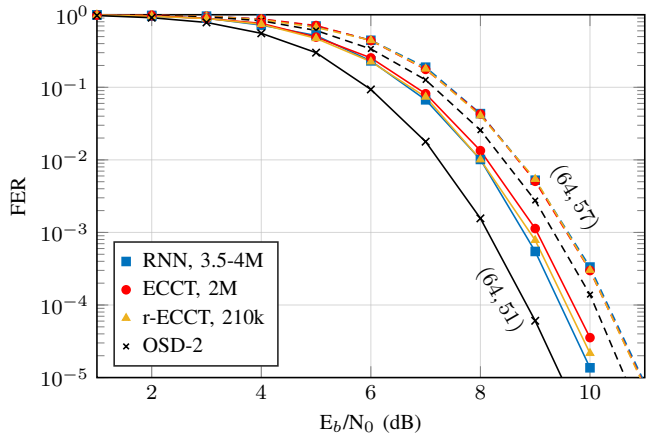

Fig. 10. FER comparison for extended BCH codes $(64, 57)$ (dashed lines) and $(64, 51)$ (solid lines) under 16-QAM, for the three NN architectures.

In general, the r-ECCT matches or slightly outperforms the ECCT, while employing only 10% of the total number of weights. This improvement can be partially attributed to the smaller architecture, which allows for an increase in the training batch size to 2^9, enabling a more precise gradient calculation.

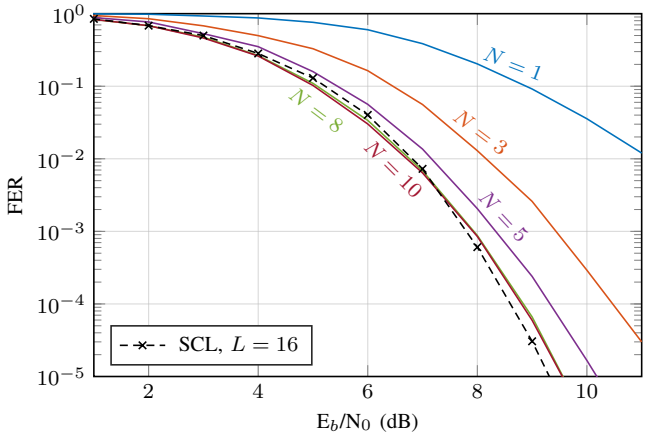

Fig. 11. Effect of the number of iterations N in the encoder block of the r-ECCT, evaluated on a polar code $(64, 32)$ using 16-QAM and Gray mapping. Increasing the value of N improves the performance until it reaches a certain value, after which it stagnates.

C. Iterative demodulation and decoding: I-SBND

In this section, we test the iterative solution proposed in section IV for two different mappings —denominated *natural* and SP [40]—, which are reported in Figure 12. The Gray mapping hitherto employed does not benefit from iterative demodulation and decoding [41], yet its performance is included for reference. Bit-interleaving is carried out in blocks of $M = 512$ codewords. Models are trained for a non-iterative decoder with Gray mapping, and reused for SP and natural mappings for simplicity. Figure 13 shows the FER of the I-SBND applied to a polar code of size (64,32) under the 16-QAM with natural mapping. As a benchmark decoder for the Polar code, we could either use a soft-output SCL decoder [42], or the soft-output OSD decoder introduced in [43]. We chose a soft-output OSD of order 2 for its implementation simplicity and since it is provably quasi-optimal. Additionally, two more practical elements were added to the decoder to decrease latency and enhance performance: (i) the iterations are stopped if the entire interleaving block satisfies the parity-check equations, thus reducing the decoding latency especially for the high-SNR regime; (ii) if the last iteration is reached, the iteration with the least non-zero syndromes is selected as the final output. As shown in [41], iterative decoding provides no performance gain for Gray mapping, whereas it significantly improves the performance of natural mapping for both I-SBND and soft-OSD decoders. Although Gray mapping achieves better performance in the first iteration, natural mapping surpasses it from the second iteration onward. A similar, though more pronounced, trend is observed in Fig. 14, where SP mapping with iterative decoding yields substantial improvements relative to Gray mapping. It is worth noting that in both figures a sudden change in the error-rate slope appears, which is consistent with classical results on iterative decoding for BICM and can be predicted using the Error-Free Feedback bound introduced in [25].

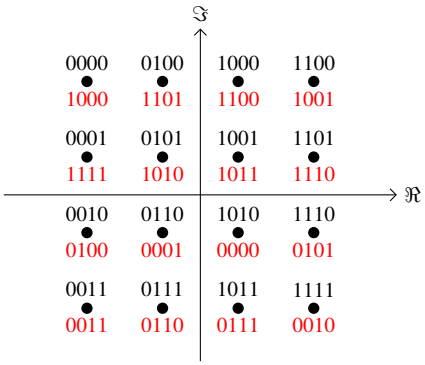

Fig. 12. Natural (in **black**) and SP (in **red**) mappings employed for iterative decoding.

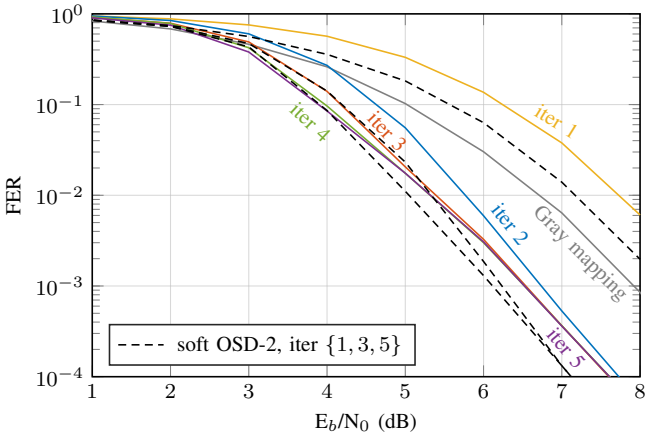

Fig. 13. FER of the I-SBND solution of Section IV implemented with the r-ECCT architecture, under a polar code of size $(64, 32)$, 16-QAM, and natural mapping (Gray mapping also included for reference). Iterations 1, 3 and 5 of the soft-output OSD of order 2 are also included as benchmark.

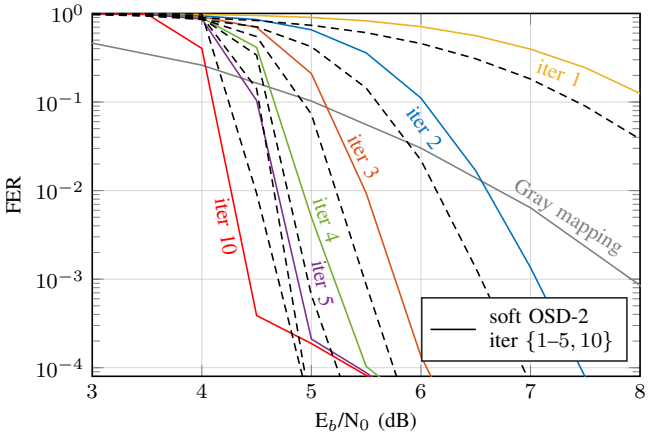

Fig. 14. FER of the I-SBND solution of Section IV implemented with the r-ECCT architecture, under a polar code of size $(64, 32)$, 16-QAM, and SP mapping (Gray mapping also included for reference). Iterations 1-5 and 10 of a soft-output OSD of order 2 are also included as benchmark.

Code	Code rate	Modulation	Training E_b/N_0	RNN	ECCT	r-ECCT	$\mathbb{W}_{\text{RNN}}$	$\mathbb{W}_{\text{ECCT}}$	$\mathbb{W}_{\text{r-ECCT}}$
BCH $(64, 57)$	0.89	16-QAM	6dB	$\alpha = 5$ $d_l = 5$ batch size $= 2^{12}$	$d_e = 128$ $N = 10$ batch size $= \{2^8, 2^9\}$		3.3M	approx. 2M	approx. 200k
BCH $(64, 51)$	0.80						3.9M		
Polar$(64, 48)$	0.75						4.4M		
Polar $(64, 32)$	0.50		5dB				6.4M		
Polar $(64, 22)$	0.34						7.8M		
Polar $(128, 96)$	0.75		6dB				17.8M		
Polar $(128, 64)$	0.50		5dB				25.5M		
binary cross-entropy loss | learning rate $= 1e^{-3}$ | Adam optimizer [44] | Gray mapping | trained until convergence, max of $1e^6$ batches									

TABLE I
SUMMARY OF THE SIMULATIONS' HYPERPARAMETERS WITH THE APPROXIMATE NUMBER OF WEIGHTS FOR EACH NN ARCHITECTURE.

VII. CONCLUSIONS

In this work, we addressed the emerging challenge of applying machine learning-based solutions to next-generation communication systems, and more particularly, at the channel decoding level. We have introduced a series of contributions aimed at advancing these solutions to be not only high-performing but also feasible for implementation in realistic scenarios. The recurrent transformer-based network greatly reduces the number of weights with virtually no penalty loss for most cases. Additionally, the proposed iterative demodulation-decoding approach allows for a much steeper performance gain with BICM and modulations other than Gray, whilst avoiding retraining the network specifically for the iterative case. The resulting performances are very promising and competitive compared to the classical close-to-optimal decoders. Future research may focus on developing a model-free decoder capable of adapting to various code rates without requiring separate training for each case, thereby significantly reducing overall complexity and enhancing adaptability. Finally, the SBND approach is thus far limited to the codes sizes hereby considered, and future works may attempt to tackle larger codes and study the limits of scalability more thoroughly.

REFERENCES

[1] I. Ahmad, S. Shahabuddin, H. Malik, E. Harjula, T. Leppanen, L. Loven, A. Anttonen, A. H. Sodhro, M. Mahtab Alam, M. Juntti, A. Yla-Jaaski, T. Sauter, A. Gurtov, M. Ylianttila, and J. Riekki, "Machine Learning Meets Communication Networks: Current Trends and Future Challenges," *IEEE Access*, vol. 8, pp. 223 418–223 460, 2020.

[2] M. E. Morocho-Cayamcela, H. Lee, and W. Lim, "Machine Learning for 5G/B5G Mobile and Wireless Communications: Potential, Limitations, and Future Directions," *IEEE Access*, vol. 7, pp. 137 184–137 206, 2019.

[3] N. Ye, S. Miao, J. Pan, Q. Ouyang, X. Li, and X. Hou, "Artificial Intelligence for Wireless Physical-Layer Technologies (AI4PHY): A Comprehensive Survey," *IEEE Transactions on Cognitive Communications and Networking*, pp. 1–1, 2024.

[4] G. Zeng, D. Hush, and N. Ahmed, "An Application of Neural Net in Decoding Error-Correcting Codes," in *IEEE International Symposium on Circuits and Systems*. IEEE, 1989.

[5] J. Bruck and M. Blaum, "Neural Networks, Error-Correcting Codes, and Polynomials over the Binary n-Cube," *IEEE Transactions on Information Theory*, vol. 35, no. 5, pp. 976–987, 1989.

[6] J. Yuan, V. Bhargava, and Q. Wang, "An Error Correcting Neural Network," in *Conference Proceeding IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*. IEEE, 1989.

[7] T. Gruber, S. Cammerer, J. Hoydis, and S. ten Brink, "On Deep Learning-Based Channel Decoding," in *2017 51st Annual Conference on Information Sciences and Systems (CISS)*. IEEE, mar 2017.

[8] K. Niu, J. Dai, K. Tan, and J. Gao, "Deep Learning Methods for Channel Decoding: A Brief Tutorial," in *2021 IEEE/CIC International Conference on Communications in China (ICCC)*. IEEE, Jul. 2021.

[9] H. Lim Meng Kee, N. Ahmad, M. Azri Mohd Izhar, K. Anwar, and S. X. Ng, "A Review on Machine Learning for Channel Coding," *IEEE Access*, vol. 12, pp. 89 002–89 025, 2024.

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

[11] R. G. Gallager, *Low-Density Parity-Check Codes*. The MIT Press, 1963.

[12] 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*, vol. 12, no. 1, pp. 119–131, feb 2018.

[13] L. Lugosch and W. J. Gross, "Learning from the Syndrome," in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*. IEEE, oct 2018.

[14] E. Nachmani and L. Wolf, "Autoregressive Belief Propagation for Decoding Block Codes," 2021.

[15] J. Rosseel, V. Mannoni, I. Fijalkow, and V. Savin, "Decoding Short LDPC Codes via BP-RNN Diversity and Reliability-Based Post-Processing," *IEEE Transactions on Communications*, vol. 70, no. 12, pp. 7830–7842, Dec. 2022.

[16] W. Xu, Z. Wu, Y.-L. Ueng, X. You, and C. Zhang, "Improved Polar Decoder Based on Deep Learning," in *2017 IEEE International Workshop on Signal Processing Systems (SiPS)*. IEEE, oct 2017.

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

[18] Y. Choukroun and L. Wolf, "Error Correction Code Transformer," 2022.

[19] S.-J. Park, H.-Y. Kwak, S.-H. Kim, S. Kim, Y. Kim, and J.-S. No, "How to Mask in Error Correction Code Transformer: Systematic and Double Masking," 2023.

[20] S.-J. Park, H.-Y. Kwak, S.-H. Kim, Y. Kim, and J.-S. No, "CrossMPT: Cross-attention Message-Passing Transformer for Error Correcting Codes," 2024.

[21] E. Kavvousanos and V. Paliouras, "An Iterative Approach to Syndrome-based Deep Learning Decoding," in *2020 IEEE Globecom Workshops*. IEEE, Dec. 2020.

[22] Y. Choukroun and L. Wolf, "Denoising Diffusion Error Correction Codes," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=rLwC0_MG-4w

[23] G. De Boni Rovella and M. Benammar, "Improved Syndrome-based Neural Decoder for Linear Block Codes," in *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*. IEEE, Dec. 2023.

[24] G. De Boni Rovella, M. Benammar, T. Benaddi, and H. Meric, "Scalable Syndrome-based Neural Decoders for Bit-Interleaved Coded Modulations," in *2024 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)*. IEEE, May 2024, pp. 341–346.

[25] X. Li, A. Chindapol, and J. Ritcey, "Bit-interleaved coded modulation with iterative decoding and 8 psk signaling," *IEEE Transactions on Communications*, vol. 50, no. 8, pp. 1250–1257, 2002.

[26] A. G. i Fabregas, A. Martinez, G. Caire *et al.*, "Bit-interleaved coded modulation," *Foundations and Trends® in Communications and Information Theory*, vol. 5, no. 1–2, pp. 1–153, 2008.

[27] A. Alvarado, "On Bit-interleaved Coded Modulation with QAM Constellations," 2008. [Online]. Available: https://core.ac.uk/download/pdf/70574676.pdf

[28] G. Caire, G. Taricco, and E. Biglieri, "Bit-Interleaved Coded Modulation," *IEEE Transactions on Information Theory*, vol. 44, no. 3, pp. 927–946, May 1998.

[29] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, Jan. 1989.

[30] M. Tüchler and A. C. Singer, "Turbo equalization: An overview," *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 920–952, 2011.

[31] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," 2017.

[33] R. Dey and F. M. Salem, "Gate-variants of Gated Recurrent Unit (GRU) neural networks," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, Aug. 2017.

[34] G. De Boni Rovella, "Github repository," https://github.com/gastondeboni/Syndrome_Based_Neural_Decoding, 2024.

[35] M. A. et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[36] F. Chollet *et al.*, "Keras," 2015. [Online]. Available: https://keras.io

[37] I. Tal and A. Vardy, "List Decoding of Polar Codes," in *2011 IEEE International Symposium on Information Theory Proceedings*. IEEE, Jul. 2011.

[38] M. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Transactions on Information Theory*, vol. 41, no. 5, pp. 1379–1396, 1995.

[39] J. Hoydis, S. Cammerer, F. Ait Aoudia, M. Nimier-David, L. Maggi, G. Marcus, A. Vem, and A. Keller, "Sionna," 2022, https://nvlabs.github.io/sionna/.

[40] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Transactions on Information Theory*, vol. 28, no. 1, pp. 55–67, 1982.

[41] S. ten Brink, "Designing iterative decoding schemes with the extrinsic information chart," *AEU-Archiv fur Elektronik und Ubertragungstechnik*, vol. 54, 01 2000.

[42] P. Yuan, K. R. Duffy, and M. Médard, "Soft-output successive cancellation list decoding," *IEEE Transactions on Information Theory*, vol. 71, no. 2, pp. 1007–1017, 2025.

[43] M. Fossorier and S. Lin, "Soft-input soft-output decoding of linear block codes based on ordered statistics," in *IEEE GLOBECOM 1998 (Cat. NO. 98CH36250)*, vol. 5, 1998, pp. 2828–2833 vol.5.

[44] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 2014.

APPENDIX A
PROOF OF THEOREM 1

First, note that since each L^b is a deterministic function of L, and given that $P_{\boldsymbol{L}|\boldsymbol{C}}$ is a memoryless channel, then so is $P_{\boldsymbol{L}^b|\boldsymbol{C}}$, i.e., for all $\boldsymbol{l}_b, \boldsymbol{c} \in \{0,1\}^n$:

$$P_{\boldsymbol{L}^b|\boldsymbol{C}}(\boldsymbol{l}^b|\boldsymbol{c}) = \prod_{i=1}^{n} P_{L^b|C}(l_i^b|c_i). \tag{30}$$

Let us then prove that $P_{L^b|C}(0|0) = P_{L^b|C}(1|1)$.

$$P_{L^b|C}(0|0) = \int_{\mathbb{R}^+} P_{L|C}(l|0)\ dl, \tag{31}$$

$$\overset{(a)}{=} \frac{1}{2}\int_{\mathbb{R}^+} \left[P_{\bar{L}|\bar{C}}(l|0) + P_{\bar{L}|\bar{C}}(-l|1)\right]\ dl \tag{32}$$

$$\overset{(b)}{=} \frac{1}{2}\int_{\mathbb{R}^-} \left[P_{\bar{L}|\bar{C}}(-l|0) + P_{\bar{L}|\bar{C}}(l|1)\right]\ dl \tag{33}$$

$$= \int_{\mathbb{R}^-} P_{L|C}(l|1)\ dl = P_{L^b|C}(1|1), \tag{34}$$

where (a) follows from the symmetry of the channel $P_{L|C}$ given in (9), and where (b) follows from the change of variable $l \to -l$. Hence, since the channel $P_{\boldsymbol{L}^b|\boldsymbol{C}}$ is memoryless and $P_{L^b|C}(0|0) = P_{L^b|C}(1|1)$, an equivalent physical model of the channel is a BSC with crossover probability q given by

$$q \triangleq P_{L^b|C}(1|0) \tag{35}$$

$$= \frac{1}{2}\int_{\mathbb{R}^-} P_{\bar{L}|\bar{C}}(l|0)\ dl + \frac{1}{2}\int_{\mathbb{R}^-} P_{\bar{L}|\bar{C}}(-l|1)dl \tag{36}$$

$$= \frac{1}{2}\int_{\mathbb{R}^-} P_{\bar{L}|\bar{C}}(l|0)\ dl + \frac{1}{2}\int_{\mathbb{R}^+} P_{\bar{L}|\bar{C}}(l|1)\ dl \tag{37}$$

$$= \frac{1}{2}P_{\bar{L}|\bar{C}}(1|0) + \frac{1}{2}P_{\bar{L}|\bar{C}}(0|1) \tag{38}$$

$$= \frac{1}{2m}\sum_{s=1}^{m}\left[P_{\bar{L}_s^b|\bar{C}}(1|0) + P_{\bar{L}_s^b|\bar{C}}(1|0)\right], \tag{39}$$

where $P_{\bar{L}_s^b|\bar{C}}$ can be obtained from $P^s_{Y|C}$ as in Lemma 1. □

APPENDIX B
PROOF OF THEOREM 2

Let us start with two properties of an (n,k) block code. i) The pseudo-inverse of a code $p_{inv}(\cdot)$ is defined by a $k \times n$ matrix A such that $\mathrm{A}\boldsymbol{c} = \boldsymbol{u}$; ii) the matrix $\mathrm{B} = [\mathrm{H}^T, \mathrm{A}^T]$, where H is the parity matrix of the code, is full rank, thus, invertible. Next, we can write for $\boldsymbol{e}_{\boldsymbol{u}}^b \in \{0,1\}^k$ and $\boldsymbol{l} \in \mathbb{R}^n$:

$$P_{\boldsymbol{E}_{\boldsymbol{u}}^b|\boldsymbol{L}}(\boldsymbol{e}_{\boldsymbol{u}}^b|\boldsymbol{l}) = P_{\boldsymbol{E}_{\boldsymbol{u}}^b||\boldsymbol{L}|,\boldsymbol{L}^b}(\boldsymbol{e}_{\boldsymbol{u}}^b|\,|\boldsymbol{l}|,\boldsymbol{l}^b) \tag{40}$$

$$= P_{\boldsymbol{E}_{\boldsymbol{u}}^b||\boldsymbol{L}|,\mathrm{H}\boldsymbol{L}^b,\mathrm{A}\boldsymbol{L}^b}(\boldsymbol{e}_{\boldsymbol{u}}^b|\,|\boldsymbol{l}|,\mathrm{H}\boldsymbol{l}^b,\mathrm{A}\boldsymbol{l}^b), \tag{41}$$

where we have used the fact that $\mathrm{B} = [\mathrm{H}^T, \mathrm{A}^T]$ is invertible. Next, recalling the result of Theorem 1 in (15), we have that

$$\mathrm{A}\boldsymbol{L}^b = \mathrm{A}\boldsymbol{C} \oplus \boldsymbol{E}_{\boldsymbol{u}}^b = \boldsymbol{U} \oplus \boldsymbol{E}_{\boldsymbol{u}}^b. \tag{42}$$

Since, $\boldsymbol{U}$ is i.i.d following a Bern(0.5) distribution, and since $\boldsymbol{E}_{\boldsymbol{u}}^b$ is independent of $\boldsymbol{U}$, then $A\boldsymbol{L}^b$ is Bern(0.5) and is independent of $\boldsymbol{E}_{\boldsymbol{u}}^b$. Hence, it follows that

$$P_{\boldsymbol{E}_{\boldsymbol{u}}^b||\boldsymbol{L}|,\mathrm{H}\boldsymbol{L}^b,\mathrm{A}\boldsymbol{L}^b}(\boldsymbol{e}_{\boldsymbol{u}}^b||\boldsymbol{l}|,\mathrm{H}\boldsymbol{l}^b,\mathrm{A}\boldsymbol{l}^b) = P_{\boldsymbol{E}_{\boldsymbol{u}}^b||\boldsymbol{L}|,\mathrm{H}\boldsymbol{L}^b}(\boldsymbol{e}_{\boldsymbol{u}}^b||\boldsymbol{l}|,\mathrm{H}\boldsymbol{l}^b).$$

Finally, marginalizing both the RHS and LHS with respect to the $k-1$ variables $(E_{u,1}^b, \ldots, E_{u,i-1}^b, E_{u,i+1}^b, \ldots, E_{u,k}^b)$ for all $i \in [1:k]$, concludes the proof.

APPENDIX C
PROOF OF LEMMA 2

Let y be a received symbol, and let $s \in [1:m]$ be the index of one of the m bits corresponding to it. We start first by writing the posterior LLR at the output of the demodulator. To this end, note first that

$$P_{\tilde{C}_s|Y}(0|y) = \sum_{\substack{\tilde{\boldsymbol{c}}\in\{0,1\}^m \\ \tilde{c}_s=0}} P_{\tilde{\boldsymbol{C}}|Y}(\tilde{\boldsymbol{c}}|y) \tag{43}$$

$$= \alpha(y) \sum_{\substack{\tilde{\boldsymbol{c}}\in\{0,1\}^m \\ \tilde{c}_s=0}} P_{\tilde{\boldsymbol{C}}}(\tilde{\boldsymbol{c}}) P_{Y|\tilde{\boldsymbol{C}}}(y|\tilde{\boldsymbol{c}}) \tag{44}$$

$$= \alpha(y) \sum_{\substack{\tilde{\boldsymbol{c}} \in \{0,1\}^m \\ \tilde{c}_s = 0}} \prod_{v=1}^{m} P_{\tilde{C}_v}(\tilde{c}_v) P_{Y|\tilde{\boldsymbol{C}}}(y|\tilde{\boldsymbol{c}}) \tag{45}$$

$$= \alpha(y) P_{\tilde{C}_s}(0) \sum_{\substack{\tilde{\boldsymbol{c}} \in \{0,1\}^m \\ \tilde{c}_s = 0}} \prod_{\substack{v=1 \\ v \neq s}}^{m} P_{\tilde{C}_v}(\tilde{c}_v) P_{Y|\tilde{\boldsymbol{C}}}(y|\tilde{\boldsymbol{c}}) \tag{46}$$

$$= \alpha(y) P_{\tilde{C}_s}(0) \sum_{\substack{\tilde{\boldsymbol{c}} \in \{0,1\}^m \\ \tilde{c}_s = 0}} \prod_{\substack{v=1 \\ v \neq s}}^{m} P_{\tilde{C}_v}(\tilde{c}_v) P_{Y|X}(y|x(\tilde{\boldsymbol{c}})) \tag{47}$$

where $\alpha(y) \triangleq 1/P_Y(y)$, and $x(\tilde{\boldsymbol{c}})$ is the symbol associated to the m bits $\tilde{\boldsymbol{c}}$, and where we have used that $\tilde{\boldsymbol{C}} \leftrightarrow X \leftrightarrow Y$ is a valid Markov chain, and that the bits $\tilde{\boldsymbol{c}}$ are independent (BICM assumption). Hence, we obtain that

$$\frac{P_{\tilde{C}_s|Y}(0|y)}{P_{\tilde{C}_s|Y}(1|y)} = \frac{P_{\tilde{C}_s}(0)}{P_{\tilde{C}_s}(1)} \frac{\displaystyle\sum_{\substack{\tilde{\boldsymbol{c}} \in \{0,1\}^m \\ \tilde{c}_s = 0}} \prod_{\substack{v=1 \\ v \neq s}}^{m} P_{\tilde{C}_v}(\tilde{c}_v) P_{Y|X}(y|x(\tilde{\boldsymbol{c}}))}{\displaystyle\sum_{\substack{\tilde{\boldsymbol{c}} \in \{0,1\}^m \\ \tilde{c}_s = 1}} \prod_{\substack{v=1 \\ v \neq s}}^{m} P_{\tilde{C}_v}(\tilde{c}_v) P_{Y|X}(y|x(\tilde{\boldsymbol{c}}))}.$$

Next, by using the AWGN channel assumption, by writing that

$$P_{\tilde{C}_v}(\tilde{c}_v) = \frac{\exp\big[(1-c_v)\tilde{l}_{a,v}\big]}{1+\exp(\tilde{l}_{a,v})}, \tag{48}$$

and by simplifying the $1 + \exp(\tilde{l}_{a,v})$ terms in both the denominator and numerator, the result follows. □

APPENDIX D
PROOF OF LEMMA 3

Let us start by writing that, for all $i \in [1:n]$,

$$P_{C_i|\boldsymbol{Y}}(0|\boldsymbol{y}) = P_{E_i^b \oplus L_i^b|\boldsymbol{Y}}(0|\boldsymbol{y}) = P_{E_i^b|\boldsymbol{Y}}(l_i^b|\boldsymbol{y}), \tag{49}$$

where we have used the fact that L_i^b is a deterministic function of the random vector $\boldsymbol{Y}$. Hence, we can write that

$$\log\left(\frac{P_{C_i|\boldsymbol{Y}}(0|\boldsymbol{y})}{P_{C_i|\boldsymbol{Y}}(1|\boldsymbol{y})}\right) = \log\left(\frac{P_{E_i^b|\boldsymbol{Y}}(l_i^b|\boldsymbol{y})}{P_{E_i^b|\boldsymbol{Y}}(l_i^b \oplus 1|\boldsymbol{y})}\right) \tag{50}$$

$$= (1 - 2\boldsymbol{l}_i^b) \log\left(\frac{1-\hat{e}_i}{\hat{e}_i}\right) \tag{51}$$

where we used the fact that $\hat{e}_i \triangleq P_{E_i^b|\boldsymbol{Y}}(1|\boldsymbol{y})$. □