# Scalable Syndrome-based Neural Decoders for Bit-Interleaved Coded Modulations

Gastón De Boni Rovella*†, Meryem Benammar†, Tarik Benaddi‡ , Hugo Meric§
*TéSA Laboratory, Toulouse, France
†ISAE-SUPAERO, Université de Toulouse, France
‡Thales Alenia Space, Toulouse, France
§Centre National d'Études Spatiales, Toulouse, France
Email: {gaston.de-boni-rovella, meryem.benammar}@isae-supaero.fr

***Abstract*—In this work, we introduce a framework that enables the use of Syndrome-Based Neural Decoders (SBND) for high-order Bit-Interleaved Coded Modulations (BICM). To this end, we extend the previous results on SBND, for which the validity is limited to Binary Phase-Shift Keying (BPSK), by means of a theoretical channel modeling of the bit Log-Likelihood Ratio (bit-LLR) induced outputs. We implement the proposed SBND system for two polar codes $(64, 32)$ and $(128, 64)$, using a Recurrent Neural Network (RNN) and a Transformer-based architecture. Both implementations are compared in Bit Error Rate (BER) performance and computational complexity.**

## I. Introduction

With the recent introduction of 5G and beyond technologies, the interest in fast and reliable communication systems has increased in an unprecedented manner. At the same time, the ever-growing computational power has given machine learning a crucial role in future communication systems as a fast and robust alternative to some classical physical layer solutions like channel demodulation and decoding.

Early works in channel decoding [1], [2] promptly faced the *curse of dimensionality*, where the space of valid codewords was too large for a common Deep Neural Network (DNN) to explore and *learn*. To tackle this problem, two main scalable alternatives were proposed: model-based solutions, that directly exploit the structure of the code [3]–[5], and model-free solutions, that do not depend on the code and allow the integration of more sophisticated machine learning techniques [6]–[8]. Model-based solutions are often neural extensions of the Belief Propagation (BP) algorithm that help tackle the negative impact of short cycles in BP decoding. However, implementations for semi-dense or dense codes, such as BCH or Polar codes [9], remain subpar.

A model-free approach, which we denote as Syndrome-Based Neural Decoder (SBND), was introduced more recently by Bennatan *et al.* [6], and has since been implemented using different deep learning techniques [7], [8], [10], [11]. The main idea behind the SBND is to produce a symmetric decoder that does not depend on the codeword and can thus be trained with a unique codeword. Although this provides us with a very promising framework, these works rely extensively on the properties of Binary Phase-Shift Keying (BPSK) and can be easily extended to Quadrature Phase-Shift Keying (QPSK). However, in order to allow for practical implementations, SBND has to be extended to higher-order modulations such as $M$-Quadrature Amplitude Modulation (QAM) and $M$-Phase-Shift Keying (PSK) for arbitrary $M$. In this work, we propose a decoder that can be directly applied to such linear modulation techniques. More particularly, we focus on Bit-Interleaved Coded Modulations (BICM) [12], [13] which, unlike classical coded modulation schemes, present the advantages of allowing the usage of any Forward Error Correction (FEC) code designed for memoryless channels, and of being more robust to burst errors.

The main difference between higher-order modulations and BPSK/QPSK is that the decoder is not directly fed with the channel output, but rather, with bit Log-Likelihood Ratios (bit-LLR) produced by the soft demodulator. Hence, in order to design an SBND for the BICM, we first start by characterizing the channel induced by bit-LLRs for two common modulation schemes. Next, we propose an SBND that extends that of [7], [8], [10], [11] to the case of high-order BICM. Finally, we analyze the performance of two main architectures for the neural-based decoders, namely, RNN-based and transformer-based, and compare their respective complexities. Finally, Section V concludes the work.

The remainder of this work is organized as follows. Section II introduces the system model and some preliminaries on channel modeling for BICM. In Section III, we describe the proposed decoding framework. Then, Section IV presents two possible implementations for the deep learning-based portion of the decoder, compares experimentally their BER performances and analyzes their complexity.

*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 $\mathrm{I}_n$ denotes the $n \times n$ identity matrix. The Hadamard product between vectors is represented by $(.)$. 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 $P_{Z|Y,X} = P_{Z|Y}$. $\mathbb{P}$ (resp. $\mathbb{1}$) denotes the generic probability (resp. indicator) of an event. $[1 : n]$ denotes the set of integers from 1 to $n$ and $\lfloor . \rfloor$ represents the floor function. The xor operation is noted $\oplus$.

## II. System model and preliminaries

### A. Bit-Interleaved Coded Modulations (BICM)

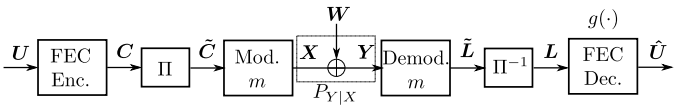


Fig. 1: General system model.

Let us consider a BICM communication setting as depicted in Figure 1. In such a setting, an input binary message $\boldsymbol{U}$ of $k$ bits, assumed to be independent and uniformly distributed, is first mapped through an $(n,k)$ linear block FEC encoder into a codeword $\boldsymbol{C}$ of $n$ bits. Let us denote the parity check matrix of this code as H. Then, a perfectly random interleaver $\Pi$, assumed to be known to the receiver as well, shuffles the bits of $\boldsymbol{C}$ into an $n$-bit sequence $\tilde{\boldsymbol{C}}$, which is then passed through a modulator. The complex-valued constellation is denoted by $\mathcal{X}$ and is assumed to be of order $m$ (i.e. $M=2^m$ states). The sequence of $n'=n/m$ symbols $\boldsymbol{X}$ is then fed to an Additive White Gaussian Noise (AGWN). Its output can be modeled as

$$\boldsymbol{Y}=\boldsymbol{X}+\boldsymbol{W},\ \text{s.t. } \boldsymbol{W}\overset{\text{i.i.d}}{\sim}\mathcal{CN}(\boldsymbol{0},\sigma^2\mathrm{I}_{n'}). \tag{1}$$

The demodulator, upon reception of the channel output $\boldsymbol{Y}$, computes the bit-Log Likelihood Ratios (bit-LLR) associated with each bit $C$ given the corresponding received signal $Y$. The resulting bit-LLRs $\tilde{\boldsymbol{L}}$ are then de-interleaved back to the original bit order. Based on the obtained bit-LLRs $\boldsymbol{L}$, the FEC decoder $g(\cdot)$ produces an estimate $\hat{\boldsymbol{u}}$ of the $k$ transmitted bits given by $\hat{\boldsymbol{U}}=g(\boldsymbol{L})$.

In this work, we investigate the design of decoders that minimize the Bit-Error Probability (BEP) defined by

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

### B. BICM equivalent channel models

The main advantage of the BICM communication setting, as opposed to standard (non-interleaved) coded modulation schemes, is that a perfectly random interleaver maps every bit in $\boldsymbol{C}$ evenly to each one of the $m$ bit positions in the constellation mapping. Hence, throughout the transmission, all bits in the sequence $\boldsymbol{C}$ end up experiencing the same channel, which is averaged out over all bit positions.

An equivalent channel model for BICM was introduced in [12] and is given in Figure 2, where $P^s_{Y|C}$ denotes the effective channel distribution experienced by a bit transmitted over a position $s$ in the constellation mapping, $f^s(\cdot)$ denotes the bit-LLRs function (depending on the bit position) and is given for all $y\in\mathbb{C}$ and $s\in[1:m]$ by

$$f^s(y)\triangleq\log\left(P^s_{Y|C}(y|0)\right)-\log\left(P^s_{Y|C}(y|1)\right), \tag{3}$$

and the last operation is simply a decomposition into a hard decision $L^b\triangleq\mathbb{1}(L<0)$ and a reliability measure $|L|$.

In order to motivate our proposed decoder of Section III, we need to characterize three channel distributions $P_{L^b|C}$. To

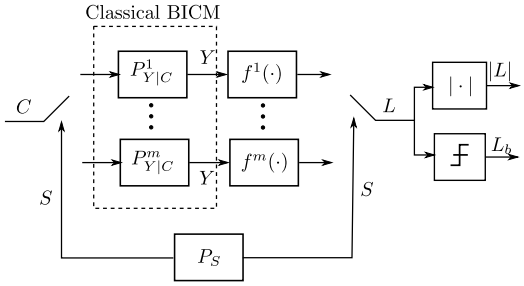


Fig. 2: BICM channel model extended to bit-LLRs.

this end, we will recall the results on $P_{Y|C}$ and $P_{L|C}$, and extend them to model the channel $P_{L^b|C}$.

*1) Channel distributions $P_{Y|C}$ and $P_{L|C}$:* Let us recall the equivalent channel models as of the BICM in Figure 2.

**Lemma 1** (BICM classical channel models [12], [13])**.** *An equivalent channel distribution of a classical BICM setting is given for all $\boldsymbol{y}\in\mathbb{C}^{n'}$, $\boldsymbol{l}\in\mathbb{R}^n$, and $\boldsymbol{c}\in\{0,1\}^n$ by*

$$P_{\boldsymbol{Y}|\boldsymbol{C}}(\boldsymbol{y}|\boldsymbol{c})=\prod_{i=1}^{n}P_{Y|C}(y_{\lfloor i/m\rfloor}|c_i)\ ,\ P_{\boldsymbol{L}|\boldsymbol{C}}(\boldsymbol{l}|\boldsymbol{c})=\prod_{i=1}^{n}P_{L|C}(l_i|c_i)$$

*where, the distributions $P_{Y|C}$ and$P_{L|C}$ are given by*

$$P_{Y|C}(y|c)=\frac{1}{m|\mathcal{X}^s_c|}\sum_{s=1}^{m}\sum_{x\in\mathcal{X}^s_c}P_{Y|X}(y|x), \tag{4}$$

$$P_{L|C}(l|c)=\frac{1}{m|\mathcal{X}^s_c|}\sum_{s=1}^{m}\sum_{x\in\mathcal{X}^s_c}P^s_{L|X}(l|x), \tag{5}$$

*and $\mathcal{X}^s_c$ is the set of symbols for which the $s$-th bit equals $c$.*

*Proof.* The proofs can be found in [13, Section 3.4]. □

Note that the distributions $P^s_{L|X}$, for $s\in[1:m]$, can be easily obtained for BPSK and QPSK modulations; however, they are very challenging to obtain analytically for higher-order more generic modulation schemes. In this work, we do not seek to characterize these distributions in a closed form, but rather, will use them to derive an equivalent binary channel model for the hard decisions on the bit-LLRs.

*2) Bit-LLRs binary channel distribution $P_{L^b|C}$:* In order to motivate the structure of the decoder developed in this work, we need to characterize the channel distribution $P_{L^b|C}$. To this end, let us consider, for instance, the 16-QAM and 8-PSK constellations with Gray labeling shown in Figures 5 and 6.

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

$$P_{\boldsymbol{L}^b|\boldsymbol{C}}(\boldsymbol{l}^b|\boldsymbol{c})=\prod_{i=1}^{n}P_{L^b|C}(l^b_i|c_i). \tag{6}$$

*Besides, for the 8-PSK and 16-QAM under Gray labeling, the channel $P_{L^b|C}$ can be approximated by*

$$\boldsymbol{L^b}=\boldsymbol{C}\oplus\boldsymbol{W}^b\ \ \textit{s.t.}\ \ \boldsymbol{W}^b\overset{\textit{i.i.d}}{\sim}\textit{Bern}(q), \tag{7}$$

*where $\boldsymbol{W}^b$ is independent of $\boldsymbol{C}$ and $q\triangleq\frac{1}{m}\sum_{s=1}^{m}P^s_{L^b|C}(1|0)$.*

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

The present Theorem states that for the 8-PSK constellation under Gray labeling, the channel $P_{L^b|C}$ is in fact a memoryless stationary Binary Symmetric Channel (BSC), as described in (7). As for the 16-QAM channel, we show that this channel model is valid for a wide range of Signal-to-Noise Ratios (SNR), i.e., except for very low SNR. Although we prove this theorem for only these two constellations, which are the most common in practice, the result can be extended to M-PSK and M-QAM constellations following the same lines of the proof.

## III. Proposed system: SBND for BICM

In what follows, we start by stating a result on Maximum A Posteriori (MAP) decoding, then we review existing literature on SBND for the BPSK and QPSK modulations and introduce our proposed solution.

### A. Decoding as a binary noise detection problem

The decoder architecture introduced in this section stems from a first result on the equivalence between decoding and denoising. First, let us recall that, given the channel model in Figure 1, the optimal decoding rule $g(\cdot)$ is the MAP rule, and is given by

$$\hat{\boldsymbol{u}} = g(\boldsymbol{l}) \triangleq \underset{\boldsymbol{u}\in\{0,1\}^k}{\operatorname{argmax}} P_{\boldsymbol{U}|\boldsymbol{L}}(\boldsymbol{u}|\boldsymbol{l}). \tag{8}$$

Let us consider the result of Theorem 1 in (7), and let us define the pseudo-inverse function of the FEC encoder as $p_{inv}(\cdot)$, i.e., $\boldsymbol{u} = p_{inv}(\boldsymbol{c})$. Then exploiting the linearity of $p_{inv}(\cdot)$, we can show that there exists a binary noise sequence $\boldsymbol{W}_u^b$ independent from $\boldsymbol{U}$ such that

$$p_{inv}(\boldsymbol{L}^b) = \boldsymbol{U} \oplus \boldsymbol{W}_u^b. \tag{9}$$

Hence, we rewrite the MAP decoding rule as

$$\underset{\boldsymbol{u}\in\{0,1\}^k}{\operatorname{argmax}} P_{\boldsymbol{U}|\boldsymbol{L}}(\boldsymbol{u}|\boldsymbol{l}) = p_{inv}(\boldsymbol{l}^b) \oplus \underset{\boldsymbol{w}\in\{0,1\}^k}{\operatorname{argmax}} P_{\boldsymbol{W}_u^b|\boldsymbol{L}}(\boldsymbol{w}|\boldsymbol{l}). \tag{10}$$

Hence, MAP decoding of $\boldsymbol{U}$ amounts to MAP detection of the binary noise –or *bit-flip*– sequence $\boldsymbol{W}_u^b$.

### B. Previous works on SBND for BPSK and QPSK modulations

Previous work from Bennatan *et al.* [6] proved that, under the assumption of a BPSK modulation and an AWGN channel, there exists a noise sequence $\boldsymbol{W}$ such that

$$\boldsymbol{Y} = \boldsymbol{X}.\boldsymbol{W} \text{ and } \boldsymbol{Y}^b = \boldsymbol{C} \oplus \boldsymbol{W}^b, \tag{11}$$

where $\boldsymbol{Y}^b$ and $\boldsymbol{W}^b$ denote the binary hard decisions of $\boldsymbol{Y}$ and $\boldsymbol{W}$. From this, they showed that the knowledge of the received signal's module $|\boldsymbol{Y}|$ and the hard-decision syndrome $\mathrm{H}\boldsymbol{Y}^b$ is enough to estimate the codeword binary noise $\boldsymbol{W}^b$, i.e.,

$$P_{\boldsymbol{W}^b|\boldsymbol{Y}}(\boldsymbol{w}^b|\boldsymbol{y}) = P_{\boldsymbol{W}^b||\boldsymbol{Y}|,\mathrm{H}\boldsymbol{Y}^b}(\boldsymbol{w}^b|\,|\boldsymbol{y}|,\mathrm{H}\boldsymbol{y}^b). \tag{12}$$

Moreover, since $|\boldsymbol{Y}| = |\boldsymbol{W}|$ and $\mathrm{H}\boldsymbol{Y}^b = \mathrm{H}\boldsymbol{W}^b$, the posterior distribution $P_{\boldsymbol{W}^b|\boldsymbol{Y}}$ does not depend on the transmitted codeword $\boldsymbol{C}$. Hence, we can train a neural network to approximate this posterior using only one codeword, as long as the noise $\boldsymbol{W}$ remains random.

In a previous work from the authors [11], the result of [6] was improved by directly estimating the bit-flips on the information bits rather than on the codewords, for both systematic and non-systematic codes. To this end, we showed that there exists a binary noise sequence $\boldsymbol{W}_u^b$ such that

$$p_{inv}(\boldsymbol{Y}^b) = \boldsymbol{U} \oplus \boldsymbol{W}_u^b, \tag{13}$$

and proved that the posterior can be written as

$$P_{\boldsymbol{W}_u^b|\boldsymbol{Y}}(\boldsymbol{w}_u^b|\boldsymbol{y}) = P_{\boldsymbol{W}_u^b||\boldsymbol{Y}|,\mathrm{H}\boldsymbol{Y}^b}(\boldsymbol{w}_u^b|\,|\boldsymbol{y}|,\mathrm{H}\boldsymbol{y}^b). \tag{14}$$

and is also independent of the message $\boldsymbol{U}$, which still allows single codeword training.

The extension of all these results to QPSK modulation is straightforward. In this work, we seek to generalize syndrome-based decoding to arbitrary higher-order modulations when the decoder is fed with bit-LLRs rather than the channel output.

### C. Proposed solution: SBND for BICM

In the following, we build on the results of [11] and on Theorem 1 to suggest an SBND for higher-order modulations. We start by proving that $|\boldsymbol{L}|$ and $\mathrm{H}\boldsymbol{L}^b$ are sufficient statistics for the detection of $\boldsymbol{W}_u^b$.

**Theorem 2** (Sufficient statistics)**.** *Considering the problem setting and the result of Theorem 1, we have that*

$$P_{\boldsymbol{W}_u^b|\boldsymbol{L}}(\boldsymbol{w}_u^b|\boldsymbol{l}) = P_{\boldsymbol{W}_u^b||\boldsymbol{L}|,\mathrm{H}\boldsymbol{L}^b}(\boldsymbol{w}_u^b|\,|\boldsymbol{l}|,\mathrm{H}\boldsymbol{l}^b). \tag{15}$$

*Proof.* Proof is relegated to Appendix B □

This previous theorem allows us to state that $|\boldsymbol{L}|, \mathrm{H}\boldsymbol{L}^b$ are sufficient to compute the posterior distribution of $\boldsymbol{W}_u^b$. Hence, we suggest an implementation of the SBND for higher-order modulations as shown in Figure 3. Observe that, as opposed to the BPSK and QPSK cases, the reliabilities $|\boldsymbol{L}|$ are not independent of the transmitted symbols, and thus, one cannot train using only one codeword ($\boldsymbol{c} = \boldsymbol{0}$ for instance). This imposes training over randomly generated codewords, in order to ensure variability of the transmitted symbols.

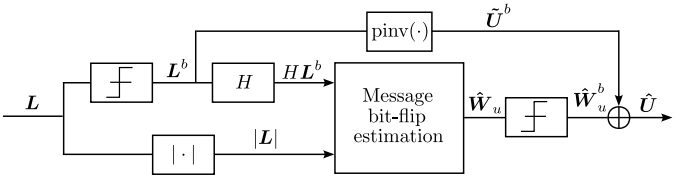


Fig. 3: Suggested SBND for higher-order modulations

## IV. Experiments

In this section, we briefly introduce two possible implementations of the bit-flip estimator of Section III-C, test the proposed decoder with both implementations, and compare both their performances and computational complexities.

| | | | | |
|---|---|---|---|---|
| RNN | $\alpha = 5$ | $T = 5$ | $d_l = 5$ | batch size $= 2^{12}$ |
| Transformer | $d_e = 128$ | $d_h = 8$ | $N = 10$ | batch size $= 2^8$ |

TABLE I: Model and training parameters.

### A. Possible architectures and training

Several DNN-based architectures can be considered for the bit-flip estimator. Due to their favorable performances and substantial differences in terms of complexity, the RNN estimator of [6] and the transformer-based estimator of [7] were implemented, employing the message-wise approach of [11]. The main difference between these architectures lies in the number of weights that comprise each network and the number of operations needed to run each solution (see Section IV-C). For a detailed description of these two solutions, refer to [6], [7], [11].

Regarding the RNN-based estimator, it is a Gated Recurrent Unit (GRU)-based RNN in which each GRU cell [14] is composed of $\alpha(2n-k)$ GRU units, with $\alpha$ an arbitrary scaling parameter. The RNN consists of $d_l$ layers –i.e. $d_l$ stacked GRU cells– and each unit performs $T$ time steps, with a final dense layer with a linear activation that outputs $\hat{\boldsymbol{w}}_u$ of size $k$. Training is carried out with a batch size of $2^{12}$.

As for the transformer architecture, it is determined by three hyperparameters: the embedding dimension $d_e$, the number of heads $d_h$ in the multi-head attention mechanism, and the number of encoder layers $N$ that are connected before the output dense layers. The large number of operations performed in the forward pass of the transformer architecture imposes a significantly smaller batch size, which is set to $2^8$.

Both architectures have a final output dense layer with a linear activation that produces $\hat{\boldsymbol{w}}_u$, which is then thresholded to obtain $\hat{\boldsymbol{w}}_u^b = \mathbb{1}(\hat{\boldsymbol{w}}_u > 0)$. Training and testing were carried out using Google's TensorFlow library [15] and the Keras API [16], using the Adam optimizer [17] with a learning rate of $\mu = 10^{-3}$ and a binary cross-entropy loss function. For both systems, codewords are generated using an AWGN of normalized SNR $E_b/N_0 = 5$dB. Important parameters are reported in Table I.

### B. Results

Both of the considered architectures were applied to the decoding of two rate-$1/2$ polar codes, namely the $(128, 64)$ and $(64, 32)$ polar codes[1]. An Ordered Statistics Decoding (OSD) algorithm is also added as a near-optimal decoding benchmark, along with an ML bound that records an error only when the OSD encounters a decoding failure *and* the obtained codeword has a higher probability than the transmitted one.

The results are displayed in Figure 4. Regarding the $(64, 32)$ polar code, the RNN-based decoder presents a decoding performance that is very close to the OSD, and surpasses the transformer architecture for all the considered $E_b/N_0$. A similar conclusion can be drawn for the $(128, 64)$ polar code, except this time, the gap between the neural-based and near-optimal solutions is more significant, especially for low-to-medium $E_b/N_0$ regions. It is worth mentioning that for the $(128, 64)$ polar code, during training, both solutions have seen at most $10^{-8}\%$ of valid codewords, proving that the models are very much able to learn a proper decoding rule only by seeing a very small fraction of the data.

In the next section, we analyze the complexity involved in each solution and compare them accordingly.

### C. Complexity analysis

In this section, we study the complexity of each of the RNN-based and Transformer-based implementations of the decoder, both at the training and inference.

As shown in [18], the number of parameters (or *weights*) in a GRU unit –including biases, which were not included in the original work [14]– is equal to $3(n_o^2 + n_i n_o + n_o)$, where $n_i$ and $n_o$ depict the number of inputs and outputs, respectively. In the case of a dense layer, the number of weights is given by $n_i n_o + n_o$. Applying these results to the RNN architecture gives the expression for the total number of weights of the RNN-based decoder for an $(n, k)$ linear code:

$$\mathbb{W}_{\text{RNN}} = 3(2d_l - 1)\alpha^2 r^2 + 3(r + d_l + {}^{k}/_{3})\alpha r + k, \quad (16)$$

where $r \triangleq 2n - k$ is the size of the input vector given by $[\,|\boldsymbol{l}|, \mathbf{H}\boldsymbol{l}^b\,]$. For the codes and hyperparameters selected, the following approximation holds to within a $0.5\%$ error margin,

$$\mathbb{W}_{\text{RNN}} \approx 3((2d_l - 1)\alpha^2 + \alpha)r^2, \quad (17)$$

yielding a total number of weights that increases as $\mathcal{O}(r^2)$ for a fixed network depth $d_l$ and scaling factor $\alpha$.

Regarding the transformer architecture, the number of weights is computed with respect to the embedding dimension $d_e$, the input size $r$, and the number of encoder layers $N$:

$$\mathbb{W}_{\text{T}} = 12Nd_e^2 + (13N + r + 3)d_e + (r + 1)k + 1. \quad (18)$$

Similarly to the previous case, an approximation can be made to within an error margin of $2\%$ for both studied codes:

$$\mathbb{W}_{\text{T}} \approx 12Nd_e^2 + 13Nd_e. \quad (19)$$

Let us observe that, for the block sizes considered, the dominant terms do not contain the code parameters $(n, k)$, producing a network structure that does not depend on the code size as strongly as the RNN architecture. Observe, however, that there is a hidden dependence on the block length if we consider that larger codes may require larger embedding spaces and potentially more encoder layers. These values were kept the same throughout both implementations and hence, the number of parameters was almost exactly the same.

The final number of weights for each decoder is included in Fig. 4. The transformer solution has a clear advantage in that the number of weights remains essentially the same as the input size increases, due to the embedding layer of fixed dimension $d_e = 128$. However, the RNN is much more shallow than the transformer: it is composed of $d_l = 5$ recurrent layers, whereas the transformer has the embedding layer, $N = 10$

[1]The parity-check matrices were taken from the channel code database in https://rptu.de/en/channel-codes.

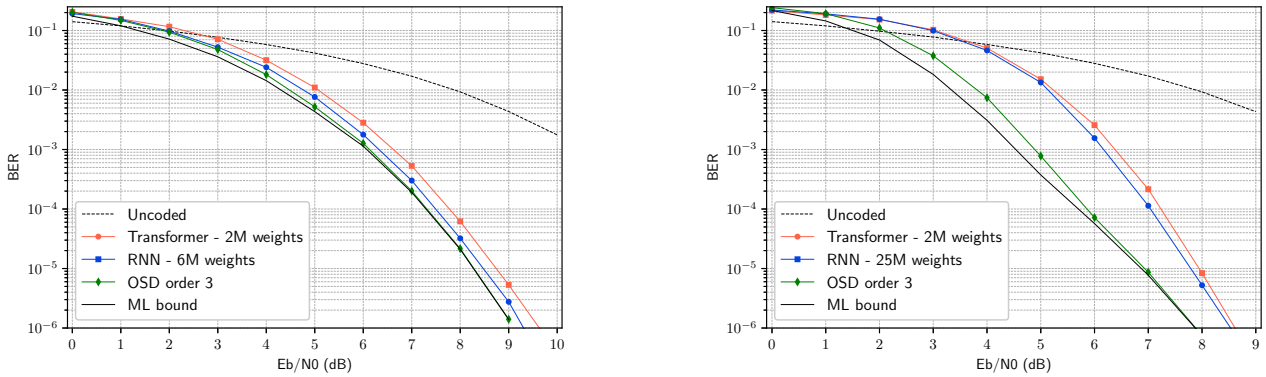


Fig. 4: Error rate studies for two rate $1/2$ polar codes: $(64, 32)$ (left) and $(128, 64)$ (right).

encoders –each consisting of two batch normalization layers, an attention layer and a dense layer– and two final dense layers with batch normalization. Additionally, as shown in [19], each self-attention mechanism entails a computational complexity of $\mathcal{O}(r^2 d_e)$, which significantly increases the inference time with respect to the RNN. Over several tests, the RNN decoded approximately $8$ times faster than the transformer for the $(64, 32)$ polar code and $25$ times faster for the $(128, 64)$ code[2].

## V. Conclusion

In this work, we have introduced a complete framework for decoding linear block codes under a BICM setting. For this purpose, we have deduced an equivalent bit-LLR binary channel model for the 8-PSK and 16-QAM modulation techniques. Next, an SBND for higher-order modulation is proposed, which takes as input the bit-LLRs instead of the BPSK-modulated signal as previous works [6], [7]. Finally, two possible architectures were implemented and compared, both in terms of performance and computational complexity.

## Appendix A
## Proof of Theorem 1

First, note that since that 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 $\boldsymbol{l}_b, \boldsymbol{c} \in \{0,1\}^n$, $P_{\boldsymbol{L}^b|\boldsymbol{C}}(\boldsymbol{l}^b|\boldsymbol{c}) = \prod_{i=1}^n P_{L^b|C}(l_i^b|c_i)$.

To proceed with the proof, we will first prove that $P_{L^b|C}(0|0) = P_{L^b|C}(1|1)$. Consider the following result

$$P_{L^b|C}(0|0) = \frac{1}{m}\sum_{s=1}^{m} P_{L^b|C,S}(0|0,s) \tag{20}$$

$$\overset{(a)}{=} \frac{1}{m|\mathcal{X}_0^s|}\sum_{s=1}^{m}\sum_{x\in\mathcal{X}_0^s} P_{L^b|X,S}(0|x,s) \tag{21}$$

$$= \frac{1}{m|\mathcal{X}_0^s|}\sum_{s=1}^{m}\sum_{x\in\mathcal{X}_0^s}\int_{\mathbb{C}} P_{L^b,Y|X,S}(0,y|x,s)dy \tag{22}$$

$$\overset{(b)}{=} \frac{1}{m|\mathcal{X}_0^s|}\sum_{s=1}^{m}\sum_{x\in\mathcal{X}_0^s}\int_{\mathcal{Y}_0^s} P_{Y|X,S}(y|x,s)dy \tag{23}$$

$$\overset{(c)}{=} \frac{1}{m|\mathcal{X}_0^s|}\sum_{s=1}^{m}\sum_{x\in\mathcal{X}_0^s}\int_{\mathcal{Y}_0^s} P_{Y|X}(y|x)dy, \tag{24}$$

where we define the decision region $\mathcal{Y}_0^s \triangleq \{y \in \mathbb{C}, f^s(y) > 0\}$, and where $(a)$ follows the unformity of the constellation, while $(b)$ follows from the Markov chain $C \leftrightarrow (X,S) \leftrightarrow L^b$, and $(c)$ from the Markov chain $S \leftrightarrow X \leftrightarrow Y$.

Next, we use a common simplification of the bit-LLRs, commonly known as *approximate LLRs*, to write that:

$$l = \log\Big(\sum_{x\in\mathcal{X}_0^s} P_{Y|X}(y|x)\Big) - \log\Big(\sum_{x\in\mathcal{X}_1^s} P_{Y|X}(y|x)\Big) \tag{25}$$

$$\approx \log\Big(\max_{x\in\mathcal{X}_0^s} P_{Y|X}(y|x)\Big) - \log\Big(\max_{x\in\mathcal{X}_1^s} P_{Y|X}(y|x)\Big) \tag{26}$$

$$= \frac{1}{\sigma^2}\big(\min_{x\in\mathcal{X}_1^s}||y-x||^2 - \min_{x\in\mathcal{X}_0^s}||y-x||^2\big). \tag{27}$$

Hence, we can write that for all $y \in \mathcal{C}$ and all $s$,

$$l^b = \mathbb{1}\big(\min_{x\in\mathcal{X}_0^s}||y-x||^2 \geq \min_{x\in\mathcal{X}_1^s}||y-x||^2\big). \tag{28}$$

Let us then consider the 8-PSK constellation with Gray mapping of Figure 5. Taking into account the simplification in (28), we give in Figure 5 the decision regions $\mathcal{Y}_0^s$ for $s \in [1:3]$. Let us now consider the case $s = 1$. We have

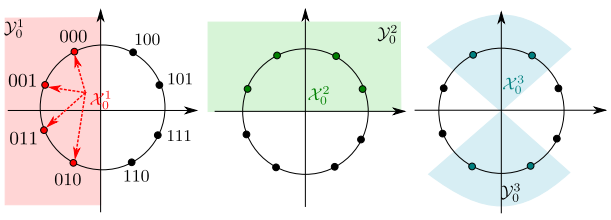


Fig. 5: Decision regions of the 8-PSK constellation

that $\mathcal{X}_0^1 = (\mathcal{X}_1^1)^\star$ and $\mathcal{Y}_0^1 = (\mathcal{Y}_1^1)^\star$. Thus, by exploiting one of

[2]It must be factored in that empirical values are deeply reliant on the software implementation, hardware characteristics, and its parallel computing capabilities. Nonetheless, the obtained decoding latencies are consistent with the expected results.

the symmetries of the normal Gaussian probability distribution $P_{Y|X}(y|x) = P_{Y|X}(y^\star|x^\star)$, one can easily prove that

$$\sum_{x\in\mathcal{X}_0^1}\int_{\mathcal{Y}_0^1} P_{Y|X}(y|x)dy = \sum_{x\in\mathcal{X}_1^1}\int_{\mathcal{Y}_1^1} P_{Y|X}(y|x)dy. \quad (29)$$

Similar results can be proved for $s = 2$ by noticing that $\mathcal{X}_0^2 = -\mathcal{X}_1^2, \mathcal{Y}_0^2 = -\mathcal{Y}_1^2$ and using the property $P_{Y|X}(-y|-x)$. Finally, for $s = 3$, note that $\mathcal{X}_0^3 = \mathcal{X}_1^3 e^{j\pi/2}$ and $\mathcal{Y}_0^3 = \mathcal{Y}_1^3 e^{j\pi/2}$ along with the symmetry $P_{Y|X}(ye^{j\phi}|xe^{j\phi}) = P_{Y|X}(y^\star|x^\star)$ for all $\phi$ yields the same result as (29). Hence, recalling that $|\mathcal{X}_0^s| = |\mathcal{X}_1^s|$, it follows that

$$P_{L^b|C}(0|0) = \frac{1}{m|\mathcal{X}_0^s|}\sum_{s=1}^{m}\sum_{x\in\mathcal{X}_0^s}\int_{\mathcal{Y}_0^s} P_{Y|X}(y|x)dy \quad (30)$$

$$= \frac{1}{m|\mathcal{X}_1^s|}\sum_{s=1}^{m}\sum_{x\in\mathcal{X}_1^s}\int_{\mathcal{Y}_1^s} P_{Y|X}(y|x)dy \quad (31)$$

$$= P_{L^b|C}(1|1). \quad (32)$$

Hence, for the 8-PSK, the channel $P_{L^b|C}$ is a binary symmetric channel, with crossover probability $q$ given by

$$q = P_{L^b|C}(1|0) = \frac{1}{m}\sum_{s=1}^{m} P^s_{L^b|C}(1|0). \quad (33)$$

Concerning the 16 QAM, we represent in Figure 6 the decision regions $\mathcal{Y}_1^s$ for $s \in [1:4]$. It can be easily seen

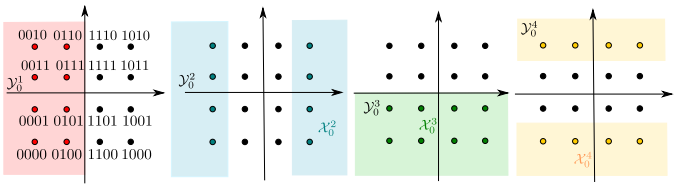


Fig. 6: Decision regions of the 16-QAM constellation

that, for $s = 1$ and $s = 3$, using the symmetries of $\mathcal{X}_c^s$ and of the Gaussian distribution, one can write that

$$\sum_{x\in\mathcal{X}_0^s}\int_{\mathcal{Y}_0^s} P_{Y|X}(y|x)dy = \sum_{x\in\mathcal{X}_1^s}\int_{\mathcal{Y}_1^s} P_{Y|X}(y|x)dy. \quad (34)$$

However, for $s = 2$ and $s = 4$, the previous equality does not hold in the sense that the integration over $\mathcal{Y}_1^s$ entails a larger clipping of the Gaussian distribution than the integration over $\mathcal{Y}_0^s$. However, if the SNR is not too low, SNR $\geq 0$ dB for a normalized 16-QAM constellation, one can show that both integrations yield the same probability. The proof follows then similarly to the 8-PSK case. □

## 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 $\boldsymbol{w}_u^b \in \{0,1\}^k$ and $\boldsymbol{l} \in \mathbb{R}^n$

$$P_{\boldsymbol{W}_u^b|\boldsymbol{L}}(\boldsymbol{w}_u^b|\boldsymbol{l}) = P_{\boldsymbol{W}_u^b||\boldsymbol{L}|,\boldsymbol{L}^b}(\boldsymbol{w}_u^b||\boldsymbol{l}|,\boldsymbol{l}^b) \quad (35)$$

$$= P_{\boldsymbol{W}_u^b||\boldsymbol{L}|,\mathrm{H}\boldsymbol{L}^b,\mathrm{A}\boldsymbol{L}^b}(\boldsymbol{w}_u^b||\boldsymbol{l}|,\mathrm{H}\boldsymbol{l}^b,\mathrm{A}\boldsymbol{l}^b), \quad (36)$$

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 (7), we have that

$$\mathrm{A}\boldsymbol{L}^b = \mathrm{A}\boldsymbol{C} \oplus \boldsymbol{W}_u^b = \boldsymbol{U} \oplus \boldsymbol{W}_u^b. \quad (37)$$

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

$$P_{\boldsymbol{W}_u^b||\boldsymbol{L}|,\mathrm{H}\boldsymbol{L}^b,\mathrm{A}\boldsymbol{L}^b}(\boldsymbol{w}_u^b||\boldsymbol{l}|,\mathrm{H}\boldsymbol{l}^b,\mathrm{A}\boldsymbol{l}^b) = P_{\boldsymbol{W}_u^b||\boldsymbol{L}|,\mathrm{H}\boldsymbol{L}^b}(\boldsymbol{w}_u^b||\boldsymbol{l}|,\mathrm{H}\boldsymbol{l}^b)$$

which completes the proof. □

## References


[1] 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.

[2] T. O'Shea and J. Hoydis, "An Introduction to Deep Learning for the Physical Layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, dec 2017.

[3] 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.

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

[5] 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.

[6] A. Bennatan, Y. Choukroun, and P. Kisilev, "Deep Learning for Decoding of Linear Codes - A Syndrome-Based Approach," 2018.

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

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

[9] E. Arikan, "Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073, jul 2009.

[10] A. Caciularu, N. Raviv, T. Raviv, J. Goldberger, and Y. Be'ery, "perm2vec: Attentive Graph Permutation Selection for Decoding of Error Correction Codes," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 79–88, 2021.

[11] G. De Boni Rovella and M. Benammar, "Improved Syndrome-based Neural Decoder for Linear Block Codes," 2023 IEEE Global Communications Conference. [Online]. Available: https://www.tesa.prd.fr/documents/26/improved_syndrome-based_neural_decoder_for_linear_block_codes.pdf

[12] G. Caire, G. Taricco, and E. Biglieri, "Bit-interleaved coded modulation," *IEEE Transactions on Information Theory*, vol. 44, no. 3, pp. 927–946, 1998.

[13] A. Alvarado, *On bit-interleaved coded modulation with QAM constellations*. Chalmers Tekniska Hogskola (Sweden), 2008.

[14] K. Cho, B. Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," 06 2014.

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

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

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

[18] 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)*, 2017, pp. 1597–1600.

[19] A. Vaswani *et al.*, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.