Improved Syndrome-based Neural Decoder for Linear Block Codes

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

***Abstract*—In this work, we investigate the problem of neural-based error correction decoding, and more specifically, the new so-called *syndrome-based* decoding technique introduced to tackle scalability in the training phase for larger code sizes. We improve on previous works in terms of allowing full decoding of the message rather than codewords, allowing thus the application to non-systematic codes, and proving that the single-message training property is still viable. The suggested system is implemented and tested on polar codes of sizes (64,32) and (128,64), and a BCH of size (63,51), leading to a significant improvement in both Bit Error Rate (BER) and Frame Error Rate (FER), with gains between 0.3dB and 1dB for the implemented codes in the high Signal-to-Noise Ratio (SNR) regime.**

I. Introduction

The introduction of the beyond-5G and 6G wireless technology standards in recent years has led to an increasing interest in communication tools that enable high reliability with a low processing latency. In this scenario, Machine Learning (ML) quickly gained notoriety as a powerful tool for developing fast, effective, and re-configurable solutions for most components of the digital communication chain, including error correction coding which will be of interest in this work.

Early works in machine learning solutions for error correction coding were presented over thirty years ago [1]–[3], and their interest has increased dramatically ever since. Indeed, recent advances in computer science and computing power have enabled a rapid expansion in the field of Neural Networks (NN) for channel coding and decoding, producing excellent results when applied to short codes [4]–[6].

However, these solutions were promptly faced with the scaling-up problem when dealing with large code sizes. Neural-based decoders being data-driven algorithms, both their performance and generalization capability are highly dependent on the representativity of the training data. Almost optimal performances, close to Maximum A Posteriori (MAP), often require training on the entire codebook whose size is exponential in the information message length. The need to decode large codes with such intractably large training spaces gave rise to new models of scalable neural decoders [7]–[12].

Currently, two main NN decoding approaches can learn to decode while training on a small subset of all possible codewords: graph-based –or model-based– and model-free decoders. Among graph-based solutions, Nachmani *et al.* proposed a generalization of the Belief Propagation (BP) algorithm by assigning trainable weights to the edges of the Tanner Graph that characterizes a particular code [9]. This architecture –applied mainly to High-Density Parity Check (HDPC) codes such as BCH– partially compensates for the effect of short cycles present in the bipartite graph. This approach quickly became dominant, and several variations were implemented [10], [11], [13], including a neural BP algorithm specific to polar codes [12], based on the BP algorithm for polar codes proposed by Arikan [14], [15]. All of these examples display favorable results in terms of Bit Error Rate (BER), but remain nevertheless deeply constrained to the specific structure of the code, namely the shape of its Tanner graph representation. Model-free decoders, on the opposite, usually achieve similar or better performances while keeping a more shallow neural network, and allowing for the use of more powerful ML techniques. The decoder introduced in [7] displays promising BER performances while being able to incorporate a wider range of machine learning technologies [8].

However, previous works –on both approaches– operate on a codeword level, seeking thus to minimize the errors between the sent codeword and the received one, instead of working at a message level. This can result in significant performance degradation, including invalid codewords at the output of the decoder. Furthermore, the introduction of the beyond-5G technology standard has expanded the interest in Arikan's polar codes, which are used for the control channels [14]. Additionally, polar codes are not systematic in their original form, making Bennatan's approach not directly applicable.

For these reasons, in this work we improve the latter approach by simultaneously fulfilling three objectives: developing a full decoder that retrieves the original message; generalizing its application to non-systematic codes; and preserving the single-codeword training property. This is accomplished while improving both bit and frame error rate measures for both polar and BCH codes.

The work is organized as follows: section II presents the problem and introduces some preliminary definitions and tools. In section III, we describe and justify the architecture of our solution, and propose an implementation using Recurrent Neural Networks (RNNs). Simulation results are displayed and interpreted in section IV, and final concluding remarks are given in section V, along with some future lines of work.

Notation: Roman and bold letters (e.g. x and $\boldsymbol{x}$) will be used to denote scalars and column vectors, respectively, whereas capital italic letters (e.g. X and $\boldsymbol{X}$) will represent random variables and vectors. Matrices are represented by non-italic capital letters (e.g. A). Given x a real value, x^b and x^s will respectively denote its hard-decision binary value (i.e. 0 if $x > 0$ and 1 otherwise) and its corresponding Binary Phase Shift Keying (BPSK) mapping, given by $0 \rightarrow +1$, $1 \rightarrow -1$. Operations $\text{bin}(\cdot)$ and $\text{sign}(\cdot)$ are defined accordingly to perform these mappings on scalar or vector values. The Hadamard product between vectors is represented by $\odot$. Finally, $\text{P}(X = x)$ represents the probability of the event $\{X = x\}$, i.e. the random variable X taking the value x.

II. Preliminary notions

A. System model

Let us now briefly introduce the framework. A simplified schematic is given in Figure 1. Let $\boldsymbol{u}^b \in \{0,1\}^k$ denote the k-bit message to be transmitted, and $\boldsymbol{x}^b \in \{0,1\}^n$ its associated n-bit codeword through a linear code $\mathcal{C}$. This codeword is mapped to a BPSK vector $\boldsymbol{x} = \boldsymbol{x}^s$, which is transmitted through a symmetric binary-input Additive White Gaussian Noise (AWGN) channel. The received signal $\boldsymbol{y}$ is used as input to the decoder to give an estimate $\hat{\boldsymbol{u}}^b$ of the message $\boldsymbol{u}^b$.

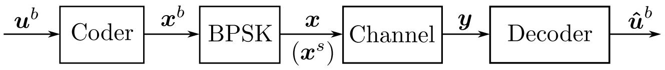

Fig. 1: General system model.

B. Noise model

In the traditional AWGN channel model, the received random signal is expressed as follows:

$$\boldsymbol{Y} = \boldsymbol{X} + \boldsymbol{N}, \tag{1}$$

where $\boldsymbol{X}$ is a random vector of size n that represents the BPSK modulated codeword and $\boldsymbol{N} = (N_1, N_2, ..., N_n)$, such that $\{N_i\}_{1 \leq i \leq n}$ are independent and identically distributed (iid) random variables distributed as $\mathcal{N}(0, \sigma^2)$. In this scenario, the following holds for all $i = \{1, ..., n\}$:

$$\begin{aligned} \text{P}(Y_i^s \neq X_i) &= \text{P}(Y_i^s = 1 | X_i = -1)\text{P}(X_i = -1) \\ &\quad + \text{P}(Y_i^s = -1 | X_i = 1)\text{P}(X_i = 1) \qquad (2) \\ &= \text{P}(N_i > 1)\frac{1}{2} + \text{P}(N_i < -1)\frac{1}{2} \qquad (3) \\ &= \text{P}(N_i > 1) = \text{P}(N_i < -1). \qquad (4) \end{aligned}$$

In this work, in order to motivate the preprocessing of the decoder input $\boldsymbol{y}$, we need to introduce an equivalent multiplicative formulation of the AWGN channel, which we define following [16, Lemma 1] by

$$\boldsymbol{Y} = \boldsymbol{X} \odot \boldsymbol{Z}, \tag{5}$$

where $\boldsymbol{X}$ and $\boldsymbol{Y}$ designate the channel input and output vectors, and $\boldsymbol{Z}$ is a random noise that verifies, $\forall i = \{1, ..., n\}$:

$$\begin{aligned} \text{P}(Z_i = z_i) &= \text{P}(Y_i = z_i | X_i = 1) \qquad (6) \\ &= \text{P}(Y_i = z_i x_i | X_i = 1) \qquad (7) \\ &= \text{P}(Y_i = y_i | X_i = 1). \qquad (8) \end{aligned}$$

Therefore, $Z_i \sim \mathcal{N}(1, \sigma^2)$ for all $i = \{1, ..., n\}$. As a direct consequence of this multiplicative model for the noise, the probability of error simplifies to:

$$\text{P}(Y_i^s \neq X_i) = \text{P}(Z_i < 0). \tag{9}$$

It is easy to observe that the probability of a 0-centered white noise being greater than 1 is the same as a 1-centered white noise of the same power being smaller than 0, i.e., (4) and (9) are equal for a same given variance σ^2.

C. Polar code PC matrix and pseudo-inverse

In the following, we introduce some results relative to polar codes that will be used later in the work. Let $\text{P}_n = \text{F}^{\otimes \log_2 n}$, where $\text{F}^{\otimes k}$ represents the kth Kronecker power of F, and F is Arikan's kernel given by

$$\text{F} \triangleq \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}. \tag{10}$$

Let G be the $k \times n$ generator matrix for a polar code of size n with k information bits, composed of k rows of the matrix P_n, and let $\mathcal{A} \subset \{1, 2, ..., n\}$ denote the indices of these rows. Finally, let V represent the identity submatrix of size $k \times n$, consisting of the k rows of the $n \times n$ identity matrix I_n with indices in $\mathcal{A}$, such that $\text{G} = \text{VP}_n$. Considering that the matrix P_n is an involutory matrix, i.e. $\text{P}_n^{-1} = \text{P}_n$, we introduce the following lemma:

Lemma 1. In the previous scenario regarding a polar code $\mathcal{C}$ generated by G, the two following statements hold:

1) The matrix H of size $n - k \times n$ consisting of the columns of P_n with indices in the complement of the set $\mathcal{A}$, i.e. $\mathcal{A}^c$, is a valid Parity-Check (PC) matrix for the code defined by G.
2) If $\boldsymbol{x}^b = \text{G}^T \boldsymbol{u}^b$, then $\boldsymbol{u}^b = \text{VP}_n^T \boldsymbol{x}^b$, and thus $f(\boldsymbol{x}^b) = \text{VP}_n^T \boldsymbol{x}^b$ yields a possible pseudo-inverse for the (n, k) polar code defined by the generator matrix G.

Proof. The first statement is easily provable if we consider that computing the matrix product GH^T consists of dot products between a row and a column of P_n with different indices, which is 0 because $\text{P}_n\text{P}_n = \text{I}_n$. The second statement can be proved by expressing the generator matrix G as a function of P_n and V_n,

$$\boldsymbol{x}^b = \text{G}^T \boldsymbol{u}^b = \text{P}_n^T \text{V}^T \boldsymbol{u}^b, \tag{11}$$

and then identifying $\text{P}_n\text{P}_n = \text{I}_n$ and $\text{VV}^T = \text{I}_k$ to find the expression for the uncoded message as a function of the codeword:

$$\boldsymbol{u}^b = \text{VP}_n^T \boldsymbol{x}^b. \tag{12}$$

This completes the proof. □

As previously stated, a function that transforms every valid codeword $\boldsymbol{x}^b$ of a code $\mathcal{C}$ into its corresponding message $\boldsymbol{u}^b$ will be called *pseudo-inverse* and will be expressed as:

$$\mathrm{pinv}(\boldsymbol{x}^b) = \boldsymbol{u}^b. \tag{13}$$

Observe that, given a pseudo-inverse $\mathrm{pinv}(\cdot)$, its application to an invalid codeword may yield an unpredictable result.

III. Syndrome-based neural decoder

In this section, we first present the previous solutions that motivated our work and then introduce the proposed solution and its characteristics.

A. Previous works

Let $\mathcal{C}$ be a linear code generated by a matrix G, and let H be a PC matrix for the code $\mathcal{C}$. It was proven in [7] that knowledge of the syndrome $\mathrm{H}\boldsymbol{y}^b$ and the module of the channel output $|\boldsymbol{y}|$ is sufficient to estimate if a position i has suffered a *bit-flipping* error, that is, $x_i^b \neq y_i^b$ (or equivalently, $x_i^s \neq y_i^s$), without incurring in any intrinsic loss of optimality. Building on this, two main solutions were proposed that achieve the best results among the codes simulated: the syndrome-based estimator in [7] and the error correction transformer in [8]. However, two main issues arise:

1) Most of the work in the literature focuses mostly on bit-wise codeword estimation, where the outputs are in no way restricted to a valid codeword [7], [8], [10], [11].
2) There is no distinction between the information and parity bits when training the system to learn to decode. Boosting the correction capabilities of the information bits to the detriment of the parity bits could provide an increase in information BER and Frame Error Rate (FER) performance.

The decoder in [13] tackled the first point, with rather modest improvements in overall performance. Our system will take it one step further by directly estimating the message $\boldsymbol{u}^b$, and therefore removing the concept of *invalid codewords* altogether. Consequently, the proposed solution will ensure full decoding of the received signal $\boldsymbol{y}$ into a message $\hat{\boldsymbol{u}}^b$, minimizing the error message-wise instead of codeword-wise.

B. Proposed decoder

In this section, we introduce a Syndrome-Based Neural Decoder (SBND) architecture that extends the work of Bennatan *et al.* [7] to a full decoder of the message $\boldsymbol{u}^b$. Hence, we define a new measure of error that assesses messages instead of codewords. Let $\tilde{\boldsymbol{u}}^b$ denote a *noisy* message defined by:

$$\tilde{\boldsymbol{u}}^b \triangleq \mathrm{pinv}(\boldsymbol{y}^b), \tag{14}$$

where the operation $\mathrm{pinv}(\cdot)$ is a pseudo-inverse for the code $\mathcal{C}$, which can be defined as (12) for non-systematic polar codes or vector slicing for systematic codes. Let $\boldsymbol{w}^b$ represent the error vector between the original message $\boldsymbol{u}^b$ and $\tilde{\boldsymbol{u}}^b$:

$$\boldsymbol{w}^b \triangleq \tilde{\boldsymbol{u}}^b \oplus \boldsymbol{u}^b, \tag{15}$$

or equivalently, in its *sign* form,

$$\boldsymbol{w}^s \triangleq \tilde{\boldsymbol{u}}^s \boldsymbol{u}^s. \tag{16}$$

Observe that the so-called noisy message is not actually a signal we receive, but rather the output of a hard-decision decoder that thresholds the vector $\boldsymbol{y}$ to obtain $\boldsymbol{y}^b$ and then inverts it through the function $\mathrm{pinv}(\cdot)$.

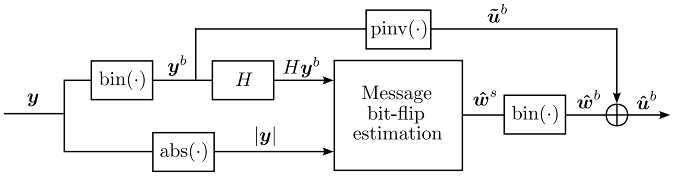

Fig. 2: SBND architecture.

Figure 2 shows the general architecture of the SBND. Essentially, the estimator uses the same inputs as [7], but will now output a vector that indicates the positions of bit-flips in the artificial vector $\tilde{\boldsymbol{u}}^b$, which will be corrected in the final stage to obtain the estimate $\hat{\boldsymbol{u}}^b$.

Theorem 1. Considering the previous structure for estimating the original message $\boldsymbol{u}^b$, the following equation holds:

$$\begin{aligned}&\mathrm{P}(\boldsymbol{U}^b = \boldsymbol{u}^b | \boldsymbol{Y} = \boldsymbol{y}) = \\ &\mathrm{P}(\boldsymbol{W}^s = \boldsymbol{u}^s \tilde{\boldsymbol{u}}^s | \, |\boldsymbol{Z}| = |\boldsymbol{y}|, \mathrm{H}\boldsymbol{Z}^b = \mathrm{H}\boldsymbol{y}^b).\end{aligned} \tag{17}$$

Proof. See Appendix.

This indicates that knowing $\boldsymbol{y}$ and computing the probability distribution of $\boldsymbol{U}^b$ is equivalent to knowing $\mathrm{H}\boldsymbol{y}^b$ and $|\boldsymbol{y}|$ and computing the probability distribution of the random variable $\boldsymbol{W}^s$, which multiplied by the artificial variable $\tilde{\boldsymbol{U}}^s$ yields the estimate $\hat{\boldsymbol{U}}^s$. This extends the previous results [7], [10], [11] to a full decoder architecture, where the output is the estimate of the original message $\boldsymbol{u}^b$, and is independent of the generator matrix –and particularly, whether it is systematic or not.

Finally, observe that Theorem 1 implies that the posterior distribution $\mathrm{P}(\boldsymbol{U}^b = \boldsymbol{u}^b | \boldsymbol{Y} = \boldsymbol{y})$ depends only on the noise random variable $\boldsymbol{Z}$ and is invariant with respect to the transmitted codeword. This enables single-codeword training, as long as the noise remains random throughout the learning process.

C. Noise estimation using RNN

Let us now introduce a possible implementation for the estimator of the bit-flip vector $\hat{\boldsymbol{w}}^s$ using RNNs. Other neural estimators could also be considered in future works to further reduce the number of trainable parameters. The basic architecture is depicted in Figure 3, where D recurrent layers are stacked on top of each other and perform T time steps before producing an output $\hat{\boldsymbol{w}}^s$. Each cell $\boldsymbol{g}_i$, $\forall i = \{1, ..., D\}$ is composed of several Gated Recurrent Units (GRU) [17], and $\boldsymbol{h}_{i,t}$ designates the state vector of the ith GRU cell at time t.

The input is the concatenated vector $(|\boldsymbol{y}|, -2\mathrm{H}\boldsymbol{y}^b + 1)$ of size $2n-k$, where the BPSK mapping has been applied to the syndrome for symmetry. Each cell has $M(2n-k)$ units, where M is a scaling factor hyperparameter. A final dense layer provides a vector output of k elements, confined to the interval

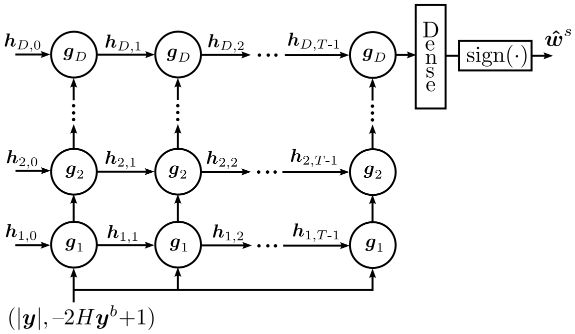

Fig. 3: A RNN implementation of the message bit-flip estimator of Figure 2. $\boldsymbol{h}_{i,t}$ represents the state of the ith GRU cell $\boldsymbol{g}_i$ at the time step t.

$[-1,1]$ via the tanh$(\cdot)$ activation function. A sign operation is added at the output to retrieve the actual sign vector $\hat{\boldsymbol{w}}^s$.

D. *Standardization of the PC matrix*

Let S denote a $n_1 \times n_2$ matrix of full row rank, with $n_1 \leq n_2$. S is said to be in its standard form when it can be expressed as follows:

$$\mathrm{S} = (\mathrm{I}_{n_1} \mid \mathrm{S}_r), \tag{18}$$

only by swapping columns and where S_r designates a $n_1 \times n_2 - n_1$ matrix. It was observed during training that using a PC matrix in its standard form resulted in a more smoothly-decaying loss and better accuracy of estimation. For this reason, [18, Theorem 5.5] was applied on the PC matrix H in order to make I_{n_1} a submatrix of H. Columns swaps are omitted so as to preserve the same row-generated vector space. Further research on the influence of H in the SBND is being carried out. Finally, let us observe that only the PC matrix is modified, and with it the syndrome probability distribution. The code $\mathcal{C}$ and its associated matrix G remain unaltered.

IV. Experiments

A. *Training and testing*

In this section, we implement, train, and evaluate the SBND previously proposed. We will use Google's TensorFlow library [19] and the Keras API [20]. Training data are generated *on the fly* in batch sizes of size 4096, using the all-1 message $\boldsymbol{u}^b = (1,1,...,1)^T$ along with an AWGN with a variance set to meet a normalized signal-to-noise ratio $E_b/N_0 = 3\text{dB}$. The network is composed of $D = 5$ layers, i.e., 5 GRU cells stacked on top of each other, and each unit performs $T = 5$ time steps. Training is carried out using the Adam optimizer [21] with a learning rate of 10^{-3}. Table I shows a summary of the hyperparameters used for training. Let us observe that the same values were used for all the decoders implemented, showing robustness to the choice of parameters and suggesting the possibility of fine-tuning for further performance improvements.

The loss function L used in training is a scaled binary cross-entropy, modified to admit input values in the range $[-1,1]$,

$$L(\boldsymbol{w}_s, \hat{\boldsymbol{w}}_s) = \mathcal{H}\left(\frac{1-\boldsymbol{w}_s}{2}, \frac{1-\hat{\boldsymbol{w}}_s}{2}\right), \tag{19}$$

Parameter	Symbol	Value
Scaling factor	M	6
Time steps	T	5
Network depth	D	5
Batch size	-	2^{12}
Training E_b/N_0	-	3dB
Learning rate	μ	10^{-3}

TABLE I: Model and training parameters

where, for two vectors $\boldsymbol{a} = \{a_i\}_{1\leq i\leq k}$ and $\hat{\boldsymbol{a}} = \{\hat{a}_i\}_{1\leq i\leq k}$,

$$\mathcal{H}(\boldsymbol{a}, \hat{\boldsymbol{a}}) = \sum_{i=1}^{k}\Bigg(a_i\log(\hat{a}_i) + (1-a_i)\log(1-\hat{a}_i)\Bigg). \tag{20}$$

To estimate the BER and FER for a given E_b/N_0, a Monte Carlo simulation is carried out, with a stopping criterion of 300 frame errors and with a minimum of 10^4 frames sent. The SBND is employed on two polar codes of rate $1/2$, with $n \in \{128, 64\}$ and $k \in \{64, 32\}$, and a BCH code of $n = 63$ and $k = 51$[1]. For each of the three codes, our solution is compared with the best among the previously introduced model-free solutions for those particular codes [7], [8] which we reproduced ourselves.

BER and FER are evaluated in a message-wise sense, as opposed to the codeword-wise approach of previous works. Given that [7] and [8] estimate the codeword $\boldsymbol{x}^b$, we perform a pseudo-inverse to retrieve the estimated message $\hat{\boldsymbol{u}}^b$ for non-systematic codes. For systematic codes, this comes down to extracting the systematic bits of the estimated codeword.

B. *Simulation results and complexity*

Figure 4 shows the BER and FER performances for two polar codes of codeword size 128 and 64 and code rate $1/2$, respectively, through an AWGN channel. As expected, the focus on information bits results in an improvement in information BER for both polar codes. However, the most valuable gain appears in the FER curves: directly estimating messages and hence removing invalid codewords as a possible output leads to a major performance enhancement. Our system is very close to the performance of the Order Statistics Decoder (OSD) [22] of order 2 for the polar code of size 64 and greatly narrows down the gap for the code of size 128, and this for only a small fraction of the processing time required for the OSD. The performance gap for the $(128, 64)$ polar code illustrates, however, the predicament of exploring a highly dimensional syndrome space, which is one of the main limitations to be tackled in the future.

Regarding complexity for the $(64, 32)$ polar code decoder, the SBND has 6.3M weights for a 5-layer deep RNN whereas [8] has only 1.2M weights but spread through a 15-layer network, resulting in a larger decoding latency. For the $(128, 64)$

[1] All of the parity-check matrices are taken from RPTU's website: https://rptu.de/en/channel-codes.

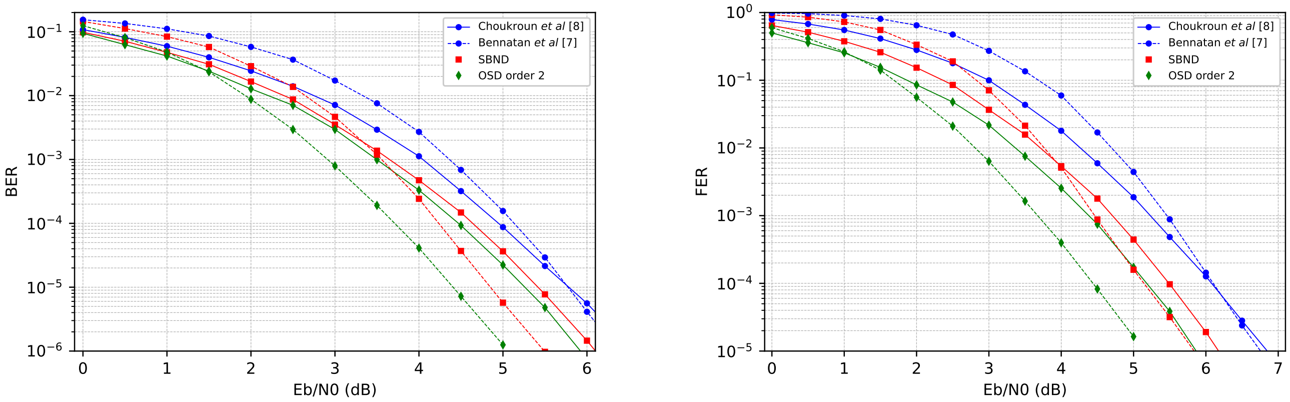

Fig. 4: Error rate studies for two polar codes of block lengths 64 and 128 and code rate 1/2. The continuous lines and the dotted lines represent the $(64, 32)$ and $(128, 64)$ polar codes, respectively.

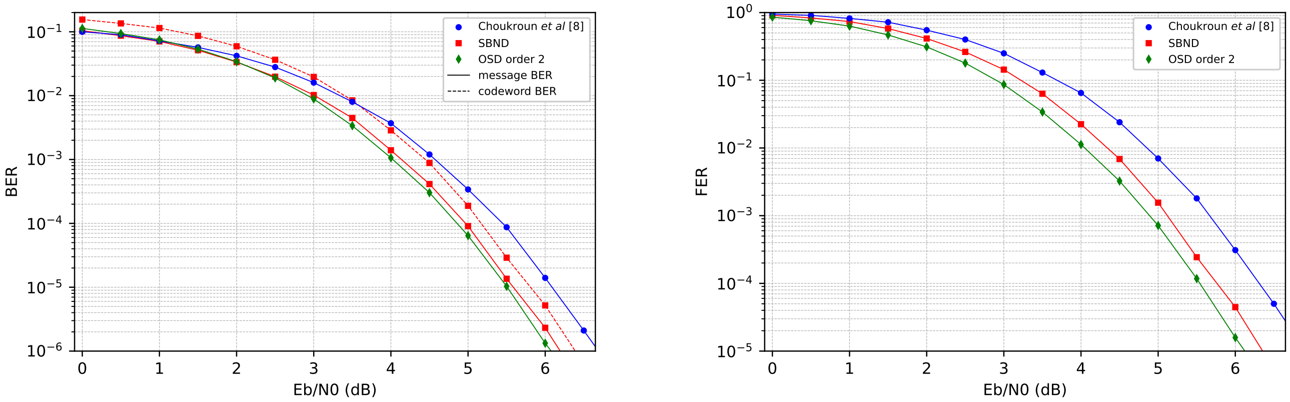

Fig. 5: Error rate studies for a $(63, 51)$ BCH code. Continuous lines represent message-to-message error rates and dotted lines depict codeword-to-codeword error rates. Codeword and message BER are the same for Choukroun *et al* [8].

polar code, the SBND and [7] are very similar in number of weights, with approximately 25.5M each. [8] reduced the number of weights by a factor of 10 but with 1dB to 1.5dB loss in BER performance and a 23-layer deep network.

In Figure 5, our method is applied to the BCH(63, 51) and compared again with the attention-based system in [8]. Here we also display the codeword-to-codeword BER along with the information BER. Observe that, in the codeword sense, our solution is not better than the transformer for all the E_b/N_0 ranges but surpasses it in the low-noise regime. Nevertheless, on message-to-message BER, the SBND represents an improvement of approximately 0.5dB with respect to [8]. The same goes for the FER, paying particular attention to the considerable proximity to the OSD of order 2.

For this code, the SBND includes 3.9M parameters in 5 layers, whereas [8] has 1.2M parameters over a total of 15 layers. A deeper analysis on complexity and latency is left for a subsequent specific study.

V. Conclusion

In this work, we have presented an extension of the system introduced by Bennatan *et al.* in [7]. This solution was inspired by the idea of restricting the system's output exclusively to valid codewords and finally developed into a full decoder that produces an estimation of the message.

Theorem 1 provided the mathematical basis and the possibility of training the system with only one codeword and random noise. We then presented a possible implementation of the message bit-flip estimator using RNNs and commented on the standardization of the PC matrix for a smoother training process. Finally, the simulation results showed considerable decoding improvements with respect to the solutions in [7] and [8], of between 0.3dB and 1dB in BER and between 0.5dB and 1dB in FER.

For future works, the main challenge remains the scalability of the system to larger codes. Machine learning solutions seem to be limited by both the codeword and the syndrome: poorer performances on larger codes with lower code rates indicate a difficulty in properly learning over high dimensional spaces with growing sizes of input and output. Notwithstanding, the experimental results of this work suggest a very powerful solution for medium block lengths, and more advanced machine learning techniques –e.g. inspired by computer vision and natural language processing– could allow for more precise learning over a high dimensional space without resorting to ever-growing deep neural networks.

Acknowledgements

The present work was funded by the Centre National d'Études Spatiales (CNES) and Thales Alenia Space (TAS) and the authors would like to thank Hugo Meric from CNES and Tarik Benaddi from TAS for their helpful comments and suggestions regarding this work.

Appendix

This section will provide proof for Theorem 1 of section III-B. Let us start by recalling the two claims of Lemma 1 in [7], regarding the framework of section II-A:

1) There exists a matrix A with dimensions $k \times n$ such that $\mathrm{A}\boldsymbol{x}^b = \boldsymbol{u}^b$ for all possible $\boldsymbol{u}^b \in \{0,1\}^k$ and its corresponding $\boldsymbol{x}^b$ through the code $\mathcal{C}$, that is, $f(\boldsymbol{x}^b) = \mathrm{A}\boldsymbol{x}^b$ is a pseudo-inverse for $\mathcal{C}$.
2) Given a matrix $\mathrm{B} = [\mathrm{H}^T, \mathrm{A}^T]$, then B has full column rank and is thus injective.

To these results, we add and prove the following lemma:

Lemma 2. For the random vectors $\boldsymbol{U}^s$, $\boldsymbol{Y}$ and $\tilde{\boldsymbol{U}}^s$ defined as in sections II-A and III-B, the events $\mathcal{E}_1 = \{\boldsymbol{U}^s = \boldsymbol{u}^s | \boldsymbol{Y} = \boldsymbol{y}\}$ and $\mathcal{E}_2 = \{\boldsymbol{U}^s\tilde{\boldsymbol{U}}^s = \boldsymbol{u}^s\tilde{\boldsymbol{u}}^s | \boldsymbol{Y} = \boldsymbol{y}\}$ are equivalent.

Proof. Considering that $\boldsymbol{Y} = \boldsymbol{y}$ and that $\tilde{\boldsymbol{u}}^s$ is a deterministic function of $\boldsymbol{y}$, it is trivial that $\mathcal{E}_1$ implies $\mathcal{E}_2$. Additionally, since $\tilde{\boldsymbol{u}}^s\tilde{\boldsymbol{u}}^s = 1$, then $\boldsymbol{U}^s\tilde{\boldsymbol{U}}^s\tilde{\boldsymbol{U}}^s = \boldsymbol{U}^s$. Therefore, the event $\mathcal{E}_2$ allows to unequivocally restore $\boldsymbol{U}^s$, and thus implying $\mathcal{E}_1$. □

With these results, we can proceed to prove Theorem 1.

$$
\begin{aligned}
&\mathrm{P}(\boldsymbol{U}^b = \boldsymbol{u}^b | \boldsymbol{Y} = \boldsymbol{y}) = \mathrm{P}(\boldsymbol{U}^s = \boldsymbol{u}^s | \boldsymbol{Y} = \boldsymbol{y}) \\
&\overset{(a)}{=} \mathrm{P}(\boldsymbol{U}^s\tilde{\boldsymbol{U}}^s = \boldsymbol{u}^s\tilde{\boldsymbol{u}}^s |\, \boldsymbol{Y} = \boldsymbol{y}) \\
&\overset{(b)}{=} \mathrm{P}(\boldsymbol{W}^s = \boldsymbol{u}^s\tilde{\boldsymbol{u}}^s |\, |\boldsymbol{Z}| = |\boldsymbol{y}|, \boldsymbol{Y}^b = \boldsymbol{y}^b) \\
&\overset{(c)}{=} \mathrm{P}(\boldsymbol{W}^s = \boldsymbol{u}^s\tilde{\boldsymbol{u}}^s |\, |\boldsymbol{Z}| = |\boldsymbol{y}|, \mathrm{B}\boldsymbol{Y}^b = \mathrm{B}\boldsymbol{y}^b) \\
&\overset{(d)}{=} \mathrm{P}(\boldsymbol{W}^s = \boldsymbol{u}^s\tilde{\boldsymbol{u}}^s |\, |\boldsymbol{Z}| = |\boldsymbol{y}|, \mathrm{H}\boldsymbol{Y}^b = \mathrm{H}\boldsymbol{y}^b, \mathrm{A}\boldsymbol{Y}^b = \mathrm{A}\boldsymbol{y}^b) \\
&\overset{(e)}{=} \mathrm{P}(\boldsymbol{W}^s = \boldsymbol{u}^s\tilde{\boldsymbol{u}}^s |\, |\boldsymbol{Z}| = |\boldsymbol{y}|, \mathrm{H}\boldsymbol{Z}^b = \mathrm{H}\boldsymbol{y}^b, \mathrm{A}\boldsymbol{X}^b \oplus A\boldsymbol{Z}^b = \mathrm{A}\boldsymbol{y}^b) \\
&\overset{(f)}{=} \mathrm{P}(\boldsymbol{W}^s = \boldsymbol{u}^s\tilde{\boldsymbol{u}}^s |\, |\boldsymbol{Z}| = |\boldsymbol{y}|, \mathrm{H}\boldsymbol{Z}^b = \mathrm{H}\boldsymbol{y}^b, \boldsymbol{U}^b \oplus \mathrm{A}\boldsymbol{Z}^b = \mathrm{A}\boldsymbol{y}^b) \\
&\overset{(g)}{=} \mathrm{P}(\boldsymbol{W}^s = \boldsymbol{u}^s\tilde{\boldsymbol{u}}^s |\, |\boldsymbol{Z}| = |\boldsymbol{y}|, \mathrm{H}\boldsymbol{Z}^b = \mathrm{H}\boldsymbol{y}^b). \qquad (21)
\end{aligned}
$$

The first equation is trivial. To obtain (a), we used Lemma 2. In (b), we used the definition of $\boldsymbol{W}^s$ and decomposed the variable $\boldsymbol{Y}$ into its module and sign, where $|\boldsymbol{Y}| = |\boldsymbol{Z}|$ by (5). In (c) and (d), the second claim of Lemma 1 was employed. In (e), we expressed $\boldsymbol{Y}^b$ as $\boldsymbol{X}^b \oplus \boldsymbol{Z}^b$, and exploited the validity of the codeword $\boldsymbol{X}^b$:

$$\mathrm{H}\boldsymbol{Y}^b = \mathrm{H}(\boldsymbol{X}^b \oplus \boldsymbol{Z}^b) = \mathrm{H}\boldsymbol{Z}^b. \qquad (22)$$

The pseudo-inverse $\mathrm{A}\boldsymbol{X}^b = \boldsymbol{U}^b$ was employed to obtain (f). Finally, (g) made use of the following result:

$$
\begin{aligned}
\boldsymbol{W}^b &= \boldsymbol{U}^b \oplus \mathrm{A}\boldsymbol{Y}^b && (23) \\
&= \boldsymbol{U}^b \oplus \mathrm{A}(\boldsymbol{X}^b \oplus \boldsymbol{Z}^b) && (24) \\
&= \boldsymbol{U}^b \oplus \boldsymbol{U}^b \oplus \mathrm{A}\boldsymbol{Z}^b && (25) \\
&= \mathrm{A}\boldsymbol{Z}^b \ \perp \ \boldsymbol{U}^b \oplus \mathrm{A}\boldsymbol{Z}^b, && (26)
\end{aligned}
$$

where $\perp$ indicates independence between two random variables and $U_i^b \sim \mathrm{Ber}(1/2)\ \forall i = \{1, ..., k\}$. Given that $\boldsymbol{U}^b \oplus \mathrm{A}\boldsymbol{Z}^b$ is independent of $\boldsymbol{W}^b$ –and thus of $\boldsymbol{W}^s$–, it can be removed from the conditional probability expression. □

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] J. Seo, J. Lee, and K. Kim, "Decoding of Polar Code by Using Deep Feed-Forward Neural Networks," in *2018 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, mar 2018.

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

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

[9] 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.

[10] 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.

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

[12] 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.

[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. 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.

[15] ——, "Systematic Polar Coding," *IEEE Communications Letters*, vol. 15, no. 8, pp. 860–862, aug 2011.

[16] T. Richardson and R. Urbanke, "The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 599–618, 2001.

[17] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches," 2014.

[18] R. Hill, *A First Course in Coding Theory*. Oxford University Press, USA, 1990.

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

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

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

[22] 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.