AGENCE NATIONALE DE LA RECHERCHE
ANR

AI4CODE
AI-aided FEC code design & decoding

ANR, Appel à Projets Générique (AAPG 2021)

AI4CODE Project (ANR-21-CE25-0006)

Deliverable D3.1
Design Space Exploration for ML-Augmented Decoding

Editor:	Valentin Savin (CEA)
Deliverable nature:	Public
Due date:	October 31, 2022
Delivery date:	December 8, 2022
Version:	1.1
Total number of pages:	28 pages
Keywords:	

Abstract
This deliverable reports on the activities carried out within the Task 3.1 of Work Package 3, aimed at establishing a design space exploration for ML-augmented decoding. Accordingly, the deliverable first formalizes several important communication scenarios where the performance or complexity of FEC decoders do not meet the expected requirements and could be improved by learning algorithms, and specifies the targeted improvements and corresponding key performance indicators. It further provides a systematic state of the art review of ML-based FEC decoders, summarizing recent advances in ML-augmented decoding of LDPC, Turbo, Polar, and more general linear block codes. Finally, it establishes a design space exploration for ML-augmented decoding, identifying parameters or technical components that can benefit from ML, and providing guidelines for improved designs of FEC decoders to be explored in the project.

List of Authors

Partner	Author
LAB-STICC/IMTA	Raphaël Le Bidan (raphael.lebidan@imt-atlantique.fr)
	Charbel Abdel Nour (charbel.abdelnour@imt-atlantique.fr)
	Elsa Dupraz (elsa.dupraz@imt-atlantique.fr)
	Catherine Douillard (catherine.douillard@imt-atlantique.fr)
CEA-LETI	Valentin Savin (valetin.savin@cea.fr)
	Valérien Mannoni (valerien.mannoni@cea.fr)
	Joachim Rosseel (joachim.rosseel@cea.fr)
IMS/INPB	Christophe Jégo (christophe.jego@ims-bordeaux.fr)
	Camille Leroux (camille.leroux@ims-bordeaux.fr)
	Romain Tajan (romain.tajan@ims-bordeaux.fr)
	Afaf Alaoui (afaf.alaoui@ims-bordeaux.fr)
IRIT/INP-ENSEEIHT	Charly Poulliat (charly.poulliat@enseeiht.fr)
ETIS/CYU	Inbar Fijalkow (inbar.fijalkow@ensea.fr)
Lab-STICC/UBS	Emmanuel Boutillon (emmanuel.boutillon@univ-ubs.fr)

Contents

Introduction

The aim of the AI4CODE project is to explore and assess how machine learning (ML) techniques can contribute to improvements in coding theory, techniques, and practice. The focus is placed on forward error correction (FEC), and the project is built around the following four inter-related objectives, identified and detailed in the scientific document of the project:

Objective #1: Explore how ML can contribute to improving the state of the art (SoA) in FEC decoding.

Objective #2: Investigate how ML techniques can improve current knowledge and practice in FEC code design.

Objective #3: Learn from the machine.

Objective #4: Develop a general expertise and critical thinking on ML algorithms and their applications to coding theory and practice.

Work Package 3 (WP3) focuses on ML-augmented FEC decoding techniques, thus addressing the first of the above objectives. Leveraging ML tools and algorithms to improve the SoA in FEC decoding has been gaining increasing attention in the last few years. A first major issue that has been identified is to improve the performance of message-passing (MP) decoding at short to moderate block length, with the ultimate purpose of closely approaching maximum-likelihood decoding (MLD) at reasonable cost. ML is also regarded as a promising approach to reduce the complexity of certain decoding algorithms, for example to alleviate the burden of message computation in non-binary LDPC decoders, or to accelerate successive-cancelation list-decoding of Polar codes by better predicting the optimal path and pruning unlikely candidates earlier. Finally, learning may also help design decoders that adapt automatically, online, to unknown or rapidly-varying channel parameters. Encouraging results have been reported in the literature, although a significant gap to MLD performance, at practical cost, still remains.

The Gantt chart of WP3 is represented in Fig. 1. This deliverable reports on the activities carried out within the **Task 3.1**, aimed at establishing a design space exploration for ML-augmented decoding. Accordingly, the deliverable is organized as follows.

Section 1 formalizes the communication scenarios where the performance or complexity of FEC decoders do not meet the expected requirements and could be improved by learning algorithms, and specifies the targeted improvements and corresponding key performance indicators (KPIs).

Section 2 provides a systematic state of the art review of ML-based FEC decoders, summarizing recent advances in ML-augmented decoding of Low-Density Parity-Check (LDPC), Turbo, Polar, and dense (*e.g.*, algebraic) codes.

Section 3 establishes a design space exploration (DSE) for ML-augmented decoding, identifying parameters or technical components that can benefit from ML, and providing guidelines for improved designs of FEC decoders to be explored in the project.

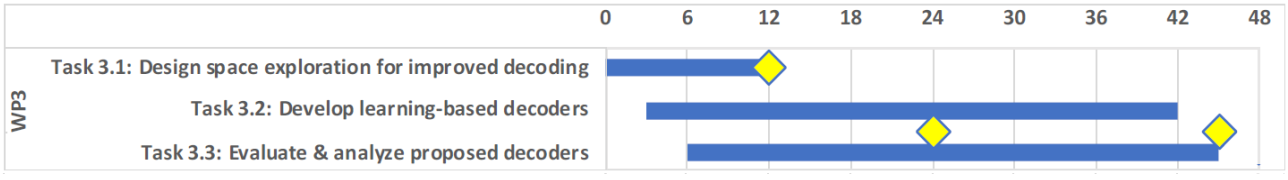

Figure 1: Gantt diagram of WP3

1 Opportunities for ML-Aided Decoding

1.1 Communication Scenarios and Expected Requirements

1.1.1 The construction and decoding of capacity-approaching codes: lessons from the past

In direct line with the pioneering work of Claude Shannon, research in the early days of coding theory was primarily focused on the following two key questions:

- How good are the best codes ?
- How can we design good codes ?

As our theoretical understanding progressed with the development of information theory and algebraic coding theory, it has been progressively realized and recognized that perhaps the major challenge in coding theory was not so much to find good codes than, quoting Berlekamp [1],

- Finding decoding algorithms for the good codes discovered so far ,
- Finding ways of implementing the decoding algorithms.

The situation evolved to the point that coding theorists turned the problem on its head, by first designing the decoding algorithm, thereby making sure that it had manageable complexity, and then looking for channel codes that were good for the selected decoder. In retrospect, the widespread use of convolutional codes may be regarded as the first demonstration of the practical benefits of such a reversed design paradigm[1]

In direct line of this first breakthrough, the invention of turbo codes and the concurrent rediscovery of LDPC codes in the late 90's have taught us two other fundamental and complementary lessons. The first lesson is that it is possible, and not so hard, to design practical codes that can approach channel capacity within tenths of decibels. The second lesson, even more unexpected, is that such a *tour de force* can be achieved with decoding complexity (quasi)-linear in the block length, owing to iterative message-passing decoders combined with proper decoder-matched code design. MLD and its prohitive time complexity, exponential in the message length, can be avoided. The subsequent invention of polar codes and their companion successive cancellation decoder by the mid 2000s has provided additional support to these conclusions. Decoding problem solved. Or is it ?

1.1.2 Challenges raised by short-length codes

The main caveat with the previous result is that is holds true only asymptotically, *e.g.*, under very large code length. In practice, experimental evidence demonstrate that carefully-designed turbo, LDPC, or polar codes can maintain performance close to the theoretical limits down to block length in the order of a thousand bits, sometimes even less. However, the situation changes drastically as we enter the realm of short codes of about a few hundred code bits or less.

The recent increasing interest for short-packet machine-to-machine communications at the heart of Internet of Things (IoT) has revived interest in the design of efficient short codes. Several studies [2–4] provide benchmarks of state-of-the-art as well as off-the-shelf FEC codes in this regime. Interestingly, they show that all coding solutions that can either reach or come very close to the finite-length coding bounds need a near-maximum-likelihood decoder to do so. Such decoder may take the form

[1]Strictly speaking, the invention of convolutional codes has preceded the discovery of the Viterbi algorithm. But it is really the latter that has shed new light on the practical potential of this family of codes, and led to revisit their design to favor codes of short to moderate memory.

of a generic ordered-statistics decoder (OSD) of sufficient order for good algebraic codes such as extended Bose-Chaudhuri-Hocquenghem (BCH) codes, of a successive-cancellation list decoder with a large enough list for polar + cyclic-redundancy-check (CRC) codes, of a sequential decoder for the polarization-adjusted convolutional (PAC) codes, or of a wrap-around Viterbi decoder for tail-biting convolutional codes of high-enough memory. The performance-vs-complexity analysis undertaken in [5, Fig. 3 & 4], [6, Fig. 1], or in [7,8], suggest that exponential decoding complexity is the price to pay to match the theoretical performance of the best short codes. We note though that this can at least partly attributed to a mismatch between the code and the decoder. Indeed, the usual design rules for capacity-approaching LDPC, turbo, or polar codes are for the most part asymptotic in nature. They thus fail to produce codes that perform well at short block length, under the same message-passing decoding algorithms that have been proven so efficient with large blocks. Yet, whether the ultimate performance limits at short block length can be achieved or closely approached, with linear or quasi-linear time complexity, is still an open question. And it seems likely that message-passing decoders should be modified and improved for that purpose, in addition to revisiting the design of the code itself.

1.1.3 Challenges for the design of hardware-constrained channel decoders

Improving the performance versus complexity trade-off of decoding in the short block-length regime is not the sole challenge ahead related to channel decoders. Standard message-passing decoders such as the max-log-MAP decoder for turbo codes, the Belief Propagation (BP) decoder for LDPC codes, or the successive-cancellation list/flip decoder for polar codes, are well-known for their remarkable performance under reasonable complexity at medium-to-long block-length. Yet the complexity might still be too high, or the algorithm itself inappropriate, for some applications. For instance, these algorithms do not fit well when considering optical communications FEC to be used in the 400Gb/s and now 800Gb/s transceivers for data center interconnection, which have to comply with highly stringent requirements in terms of area, power, and throughput [9]. The usual approach consists of resorting to reduced-complexity approximations of the standard message-passing decoders, for example by replacing the BP algorithm by the min-sum algorithm. This comes with a performance degradation that can be compensated only partly through the use of appropriate ad-hoc parameters, tables, and scaling factors, which, most-of-the time are optimized by trial and error.

Another challenge resides in the design of high-performance hardware-constrained soft-decision decoders that use only two or three bits per message while operating on coarsely-quantized soft-inputs [10]. Devising such complexity- or hardware-constrained decoders requires efficient methods for automated fine-tuning of the decoder so as to minimize the discrepancy with respect to a reference algorithm. This remains an area of continuing research of great practical interest.

1.1.4 The Holy Grail of soft-decision decoding of algebraic codes

Of the billions of chips or devices that decode error-correcting codes every minute across the globe, a significant part of them process hard- rather than soft-information. This is true in particular for the many Reed-Solomon and BCH codes that have found their way in current digital communication and storage systems. As already noted by Berlekamp back in 1987 [1], any progress towards near-maximum-likelihood soft-decision decoding of such powerful algebraic codes may find immediate commercial application at a large scale, given the typical 2-dB additional coding gain afforded by the use of soft inputs. Despite more than fifty years of continuing research and a series of notable achievements towards that goal, it is fair to say that no practical, general solution with close-to-optimal performance has emerged so far, at least for the most ubiquitous codes, the (255, 239 or 223) RS code over GF(256) being the prime example. In particular, considerable efforts have been devoted to leverage on the low-complexity message-passing decoders initially developed for LDPC or polar codes for that purpose. However the dense, random-like structure of their parity-check matrix at the bit-level dramatically hinders the effectiveness of such approaches. Albeit many counter-measures or work-arounds have been suggested, see for example [11] or [12], the long-standing problem of soft-decision

decoding of strong algebraic codes remains largely open.

As a summary, it is clear that finding universal soft-decision decoding algorithms that apply to a broad range of channel codes and showcase close-to-maximum-likelihood performance could not only be beneficial to existing systems but also simplify the chip design for future systems as well.

1.1.5 When model-based approaches meet ground truth

Let us conclude this review of opportunities for ML-aided decoding by noting that all channel decoders are inherently model-based, in that they derive from a particular statistical model assumed for the channel. In practice, however, any real channel deviates from the assumed model, which may entail significant performance loss, especially with decoders operating on soft inputs. Having decoders robust to channel uncertainty is thus highly desirable, especially for short packet wireless communication where the system has to accommodate for a wide range of propagation conditions with limited metadata available for channel estimation [13].

1.2 Targeted Improvements and KPIs

The channel decoding problem can be cast as a classification problem and solved using supervised machine learning. In essence that means replacing the decoder by a deep neural network, capitalizing on its universal function approximation property. However this approach does not scale well with the message length, and it is mainly of interest for coding problems for which an optimal solution, code and/or decoding algorithm, is still lacking. AI4CODE instead promotes a different route, that consists in leveraging on ML techniques to improve, in one way or another, existing message-passing decoders devised for modern channel codes.

From a broad theoretical perspective, WP3 aims at investigating how, and to which extent, ML techniques may contribute to revisit the performance/complexity/latency trade-offs achieved by current message-passing decoders. The general ideal of ML-augmented decoders is to trade online benefits with prior offline training complexity. Practical benefits may take many different forms, most of which fall into the following three categories:

- Learn to decode better
- Learn to decode smarter
- Learn to be robust

Learn to decode better includes all methods that take advantage of ML assistance to approach maximum-likelihood performance with controlled decoding complexity. A prominent class of algorithms that falls into this category is the family of neural BP decoders obtained by deep unfolding of the standard BP algorithm, but it is only one of many possible approaches.

Learn to decode smarter should be understood here as the process of leveraging on ML to make existing decoding algorithms more efficient. The exact definition of efficiency is context-dependent. Depending on the target application and constraints enforced on the decoder design, a higher efficiency could mean reduced computational complexity, higher average throughput, less hardware resources, or lower power consumption.

Learn to be robust refer to the problem of adapting the decoder to uncertain environments. Being data-driven by nature, ML algorithms have the capability to learn the model online, from the input data. As such, they can help recovering from a mismatch between the model assumed by the decoder and the true channel statistics. This applies not only to standard message-passing decoders, but also to neural-aided decoders, which may have been trained under certain specific operating condition, a target SNR for example. Whether or not these neural augmentations will generalize correctly to data never met during the training, and behave as expected on the field, is a very important issue for practical applications.

In the forthcoming WP3 studies, the benefits of learning for the channel decoding problem will be assessed both from the performance and complexity angles.

Performance improvement will be measured either by the bit-error rate (BER) or frame-error rate (FER). Which one is more relevant depends on the context and application. While FER is the relevant metric for wireless transmission systems design, high-capacity optical networks *e.g.* enforce very low BER requirements. Defining proper metrics for estimating decoding complexity is a much more delicate problem, as this highly depends on the implementation target, which may range from software decoding on a general purpose computer or DSP, to high-performance decoding on a dedicated ASIC chip, and on the related design constraints. Hence, rather than imposing *a priori* metrics on which to build comparisons, we prefer to leave the possibility to choose the one that best demonstrates the savings that result from the introduction of learning. Those saving will be measured with respect to some reference decoder for which we already have the numbers for the selected metric. Example of possible KPIs for complexity savings could be the number of iterations, the power consumption, the average throughput, etc.

2 Review of State-of-the-Art ML-Aided Decoding

2.1 LDPC Codes

2.1.1 The many flavors of neural BP

The belief propagation, or BP algorithm, is the reference message-passing decoder for LDPC codes. Augmenting the BP decoder with learning capabilities is one of, if not the earliest successful attempt to bring the two worlds of coding theory and machine learning closer together. We trace it back to the pioneering work of Nachmani, Be'ery and Burshtein in 2016 [14], who where first to recognize that, by unrolling the iterations of the BP and assigning weights to the edges of the unfolded Tanner graph of the code, a layer-wise structure similar to a feed-forward (FF) neural network (NN) architecture naturally emerges. This is the principle of the *weighted BP* decoder, and also the case-study that probably has received the most attention to date, in light of the great expectations raised by the performance improvement observed with respect to classical BP. In the following the term *neural BP* will refer to a more general class of learning-based BP decoders that encompass but is not restricted to *weighted BP*. Trained neural BP is regarded as a promising direction towards a universal decoding solution, capable of overcoming the short-cycle correlation effects that hinder the message-passing decoding performance not only of medium-to-short LDPC codes, but of more general linear block codes having dense parity-check matrices as well. It is fair to say that the highest expectations initially placed on this approach have not been totally fulfilled as of today, as the gap to maximum-likelihood decoding has not been completely bridged. Yet considerable research efforts have been devoted to improving the error-correction performance of neural BP over the last 6 years, and important progress have been made. Some of the main research trends are reviewed below, without any claim to completeness. Most of these studies target the more general problem of decoding arbitrary linear block codes, but may apply equally-well to LDPC codes (see Section 2.1.4 for a more detailed discussion). Section 2.4 describes other learning-based decoding methods developed for general linear codes. They could be used to decode LDPC codes as well, albeit probably not in the most efficient manner.

A large body of investigations has been devoted to the search for more relevant learning architectures than the initial weighted BP decoder of [14], that better specialize to the specific problem of decoding an error-correcting code through message-passing on its Tanner graph. In [15], it is shown that replacing the FF architecture of [14] by a recurrent NN (RNN) can yield substantial reduction in the number of hyper-parameters to train. The simplified architecture of [16] that combines weights sharing across edges and/or iterations with scaling of messages is a complementary strategy to further reduce the number of hyper-parameters to optimize. In [17] expert-domain knowledge is incorporated into the neural BP architecture by enforcing the NN to be invariant to cyclic shifts of its input, in order to specialize the neural decoder to the decoding of cyclic block codes. [18] starts from a highly-redundant parity-check matrix of the code and uses the trained weights to prune the corresponding over-complete Tanner graph by removing check-nodes considered as irrelevant to the decoding. The result is an unrolled graph with a different set of nodes in each successive check node layer, which is equivalent to using different parity-check matrices at each iteration. Pruning-based weighted BP is shown to reduce the gap to MLD compared to standard weighted BP, at the cost of a higher number of weights. In [19] weighted BP is supplemented with a learned bit-guessing process to further approach MLD.

Deep unfolding applies not only to BP, but also to any of its variants. In particular, the normalized-min-sum (NMS) and the offset-min-sum (OMS) are two hardware-friendly approximations of BP that have found widespread use in decoder chips. Both algorithms depend on empirical parameters (messages scaling factor or offsets), to be optimized for improved performance. [15] and [20] have proposed deep-unfolded weighted variants of NMS and OMS decoders, respectively, as a way to improve upon standard NMS and OMS and even surpass standard BP. A major limitation of weighted

BP/NMS/OMS and related neural decoders with edge-specific hyper-parameters, is that the number of edges scales rapidly with the code length, making training impractical for long codes (and a high number of iterations).[21] has described an efficient method of training the weighted NMS and OMS hyper-parameters that overcomes this limitation. One can take advantage also of structure in the code itself to facilitate the realization and training of the neural decoder. For example, in [22] the lifting structure of protograph-based LDPC codes is exploited through weight sharing among edges bundles to facilitate and improve the training of weighted N/OMS.

The scheduling of messages is another degree of freedom in LDPC decoder design that not only shows up implicitly in weighted BP decoders or its variants, but can benefit from learning as well. The DNN that naturally emerges from deep unfolding of several successive BP decoding iterations tacitly assumes a flooding schedule. But hardware-friendly decoders often use a layered-type schedule instead, for it can improve the convergence speed and/or the decoding performance compared to a flooding schedule, when the maximum number of iterations is highly constrained. In layered-type scheduling, nodes in the Tanner graph are updated serially based on the latest messages received from their neighbors. [23, 24] introduce a weighted NMS decoder optimized for the decoding of 5G protograph-based raptor-like codes under a layered node activation schedule. In contrast to classic weighted BP, here the DNN to train is obtained by unrolling the message passing flow within a single iteration of layered decoding. It is also well-known that online adaptation of the scheduling to the reliabilities of the received code bits can outperform a fixed layered schedule. However finding the best sequence of nodes activation can be computationally intensive. [25] has proposed to use reinforcement learning (RL) techniques to learn the best scheduling policy for the efficient decoding of short LDPC codes. The concept has been extended recently to the decoding of medium-length LDPC codes through check node clustering and hierarchical scheduling (nodes within a cluster, and clusters within an iteration) [26]. Whereas [25] compares three different RL algorithms, Q-learning included, only the latter is retained in [26].

All the weighted BP variants discussed so far introduce weights to be trained in the edges and/or in the nodes, but retain essentially the same computation rules than standard BP. A few advanced architectures have considered replacing the standard BP message-passing computation rules at the check and/or variable nodes by learned functions, giving rise to new kinds of neural BP decoders. This approach was first promoted in [27], in the context of finite-alphabet iterative decoding (FAID). FAID decoders are a particular class of LDPC decoders in which the messages belong to some finite alphabet and the message computation rules are designed to improve the error-floor performance of BP by correcting certain problematic error patterns (trapping sets) with low-complexity decoders. In [27] as well as in the following work [28], neural FAIDs are trained to learn parameterized node activation functions that can correct certain specific, detrimental trapping sets. In a similar vein, [29] replaces optimized look-up tables by learned message passing rules in information-bottleneck LDPC decoders, where the goal is to maintain performance close to standard floating-point BP under the constraints of coarsely-quantized messages. Learning better message passing rules has applications beyond the design of low-complexity LDPC decoders. In the hyper-graph network decoder for general linear codes proposed in [30], a fully-connected FF NN is used to learn message computation functions that outperform standard BP. All these approaches fall under the general framework of *graph neural network decoders*, which is a another form of neural BP where the code Tanner graph is regarded as a graph neural network in which node activation functions can be learned. We have already stressed the fact that weighted BP and other related deep unfolding approaches are limited by the fact that the complexity of training and resulting accuracy does not scale well with the block-length and number of iterations. Graph neural network decoders are viewed as a very promising solution to break this barrier [31, 32].

There has been an attempt to apply even more sophisticated and general learning architectures to the decoding of linear codes, for example the Transformer architecture that has proven so useful for natural language processing [33]. However it seems unlikely that such model-free approaches can prove more efficient than model-based approaches for problems such as the decoding of LDPC codes, which have a high degree of structure and symmetries.

2.1.2 Loss function and training methodology

Apart from the learning architecture itself, different loss functions or training methodologies have also been explored. Supervised learning driven by the classical binary cross-entropy (BCE) loss function is a popular choice that has shown good results in training neural BP decoders [15]. However there is no guarantee that this is the best optimization criterion when it comes to minimizing the BER or FER at the trained decoder output. Alternative loss functions have been proposed to arrive at neural decoders that are better trained to correct bit or codeword errors. One of them is the soft-BER loss function proposed in [16]. More recently, [34] introduced another related loss function called negative BER loss. Not only the loss function but also the training methodology can be optimized to arrive at a better trained decoder, and/or reduce the training time and complexity. Active learning inspired training methods specialized to the weighted BP decoding problem have been investigated in [35]. The goal is to generate the training data in an active manner that favors batches of samples that are most informative for the learning task at hand (here the decoding of an error-correction code). In [36], the weighted BP decoder is viewed as a student that is trained to mimic the behavior of a reference expert decoder by means of a proper penalty term introduced in the BCE loss function. The reference teacher decoder should have better BER performance than the student neural BP for the training to produce meaningful results. A simple choice here for the latter can be a standard BP or min-sum decoder a certain amount of iterations ahead of the trained student neural decoder.

2.1.3 Learned BP post-processing techniques

Neural BP is by no means the only way to leveraging on learning techniques to improve the performance of BP decoding of short-to-medium LDPC codes. Many variants of the classical BP already exist to address some of its shortcomings. Most, if not all of them could take advantage of learning to arrive at more efficient decoders. In [37], a DNN fed by the channel LLRs is used to predict the bit flips that are most-likely to lead to successful completion of the BP decoding process, when standard BP gets stuck and fails to decode. A different yet related approach is explored in [38], in which soft-decision decoding is formulated as a (weighted) bit-flipping problem to solve in a sequential manner, one bit flip after the other. Reinforcement learning (standard Q-learning algorithm) is used to learn the best sequence of flips, starting from the syndrome computed for the received word. Learning has also been considered to improve the error-floor performance of LDPC decoders by resolving message-passing decoding failures due to trapping sets or other related problematic structures in the Tanner graph of the code. We already mentioned earlier the learned FAIDs that are designed specifically for that purpose [27, 28]. Alternatively, [39] keeps a standard BP decoder and draws on a DNN to locate, with high probability of success, some or all of the erroneous bits that are part of the trapping set. Those bits are then flipped and decoding is restarted. The DNN is trained on a supervised manner, on the basis of simulated error statistics collected at high SNR.

2.1.4 Neural BP decoding: kill two birds with one stone?

As mentioned in Section 2.1.1, neural BP is regarded as a promising decoding solution, not only for medium-to-short LDPC codes, but for more general linear block codes having dense parity-check matrices as well. First works on leveraging ML to improve the BP decoding performance actually focused on so-called high-density parity-check (HDPC) codes, such as BCH or Reed-Muller (RM) codes. A quick look at the 23 references discussed in Section 2.1.1 and Section 2.1.2, inform us that 7 references [14–17, 20, 35, 36] target[1] HDPC codes only, 9 references [19, 21–24, 26–29] target LDPC codes only, and the remaining 7 references [18, 25, 30–34] target both HDPC and LDPC codes.

Since HDPC codes are defined by higher density bipartite graphs, inducing harmful correlation effects between the exchanged messages, the standard BP decoding yields, unsurprisingly, poor error

[1] We say that a reference *targets* a family of codes, if the error-correction performance of the proposed ML-based decoding solution was assessed for this family.

correction performance. To some extent, this makes the ML's task of improving BP decoding performance much easier in the HDPC case. A solution that is effective for HDPC codes, may be less effective for LDPC codes (*e.g.*, in terms of performance gain between the ML-based and the standard BP). However, in many cases, the error correction performance of a HDPC code under ML-based BP decoding might not be better than that of a properly designed LDPC code under the standard BP decoding. Which does not mean that neural BP decoding misses the target when it concentrates on HDPC codes, since a low-complexity universal decoding solution may have important, far-reaching practical implications.

Another aspect that is worth being emphasized here concerns the number of decoding iterations. The vast majority of the papers reviewed here consider a reduced number of decoding iterations, usually between 5 and 25 iterations. This may be of course motivated by practical considerations, with respect to either the training phase (increasing the number of iterations may also increase the size of the neural network, e.g., in feed-forward architectures) or the use of the trained decoder in a latency-constraint communication system. Yet, getting rid of practical considerations, it is not clear whether considering an increased number of decoding iterations would actually help the training process. In most situations, this seems not be the case. Put differently, what ML really optimizes is more the speed of convergence of the BP decoder than its intrinsic capability to correct errors (especially in case of LDPC codes).

Summarizing, neural BP decoding find applications in decoding both HDPC and LDPC codes. As to the error correction performance (under neural BP decoding), a properly designed LDPC code remains the option of choice, although its intrinsic – i.e., MLD – error-correction capability might be worse than that of a well-chosen, or even random HDPC code. Since we deal with iterative decoders, a distinction is also to be made between speed of convergence and error correction capability under a suitable large number of decoding iterations. All this may suggest that current forms of neural BP decoding are not able to fully exploit the intrinsic error-correction capability of the code, and may call for different forms of ML-based decoding, and/or possibly combined with the use of post-processing techniques.

2.2 Turbo Codes

The literature on the topic of AI techniques for the decoding of turbo codes is not very extensive. Since this deliverable is dedicated to ML-augmented decoding, we have compiled the main studies dedicated to the use of AI to improve the decoding process of convolutional and turbo codes.

2.2.1 Neural decoders for convolutional and turbo codes

N-Turbo decoder

The authors of [40] propose a neural decoder for rate-1/2 recursive systematic convolutional codes (N-RSC) and generalize it to a neural turbo decoder architecture. As shown in Fig. 3, the N-RSC decoder uses an RNN architecture consisting of two layers of bidirectional gated recurrent units (bi-GRU), each followed by batch normalization units, and the output layer is a single fully connected sigmoid unit. It is initialized by pre-trained BiGRU with BCJR input and output to imitate the BCJR algorithm and followed by an end-to-end training until convergence. An N-RSC decoder trained with block length 100, performs the same as the Viterbi or BCJR algorithms, whatever the block size.

From a complexity viewpoint, the complexity of all RSC decoders (Viterbi, BCJR, neural decoder) is linear in the number of information bits (block length). The number of multiplications is quadratic in the dimension of hidden states of GRU (200) for the proposed neural decoder, and the number of encoder states (4) for Viterbi and BCJR algorithms, resulting, in a first approach, in a factor of 50 in favour of the Viterbi/BCJR algorithms.

In the neural turbo decoder structure (N-Turbo), the N-RSC decoders are replaced by RNN-based N-BCJR decoders. N-BCJR decoder are modified N-RSC decoders that can take bit-wise prior distribution at their input. The training is carried out at a fixed SNR. The proposed N-Turbo meets

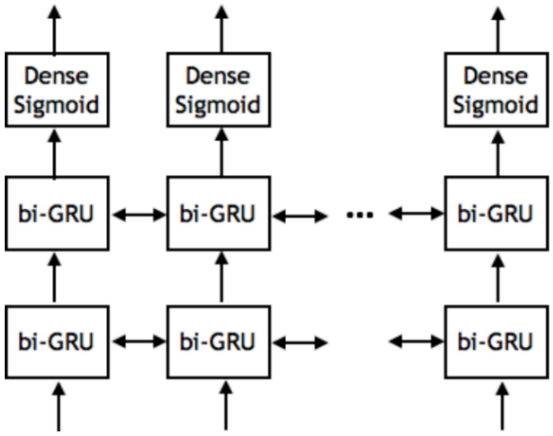

Figure 2.1: : N-RSC, neural decoder for RSC codes, taken from [40].

the performance of a conventional turbo decoder for block length 100 and 1000, and in some cases, performs slightly better.

The paper outlines the following advantages for neural decoders: the neural network can learn a decoding algorithm even if the channel does not yield to a clean mathematical analysis and the neural decoder does not require estimating the channel noise to decode. Furthermore, it seems to be more robust to non-Gaussian noise than a conventional turbo decoder using the BCJR algorithm. On the other hand, this decoding method requires a large amount of data to train numerous parameters.

DeepTurbo

The neural decoder proposed in [40] was then improved in [41] under the name DeepTurbo. The soft-in soft-out (SISO) decoding blocks are still bi-GRUs but unlike in the previous study, DeepTurbo does not use any BCJR knowledge, it doesn't share weight across different iterations and more information is passed from one stage to the other. As a result, the authors of [41] claim that DeepTurbo lowers the floor for short blocks compared to a conventional turbo decoder and it adapts better to non-AWGN channels than conventional turbo decoders and the neural turbo decoders of [40]. However, for longer blocks, further training is required.

TurboNet and its pruned version, TurboNet+

An alternative and competitor to DeepTurbo is called TurboNet [42]. TurboNet is a model-driven deep learning (DL) architecture for turbo decoding that combines DL with the traditional max-log maximum a posteriori (max-Log MAP) algorithm. The original iterative decoding structure is unrolled and the max-log-MAP algorithm is parameterized. Each iteration is replaced by a DNN decoding unit, as shown in Fig. 2.2. The conventional SISO decoders using max-log-MAP algorithm are replaced by two subnets based on neural max-log-MAP algorithm, as described in Fig. 2.3 and Fig. 2.4.

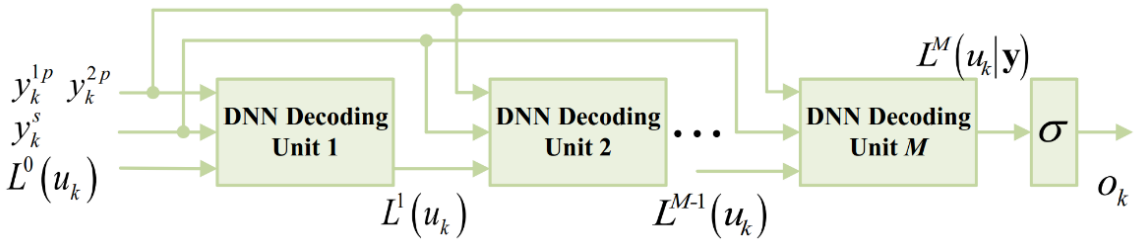

Figure 2.2: : TurboNet architecture, taken from [42].

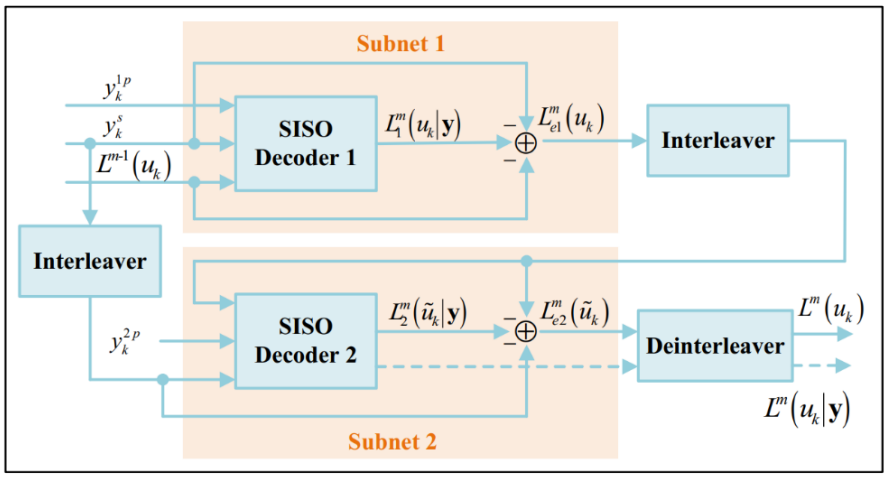

Figure 2.3: : DNN decoding unit in the TurboNet architecture, taken from [42].

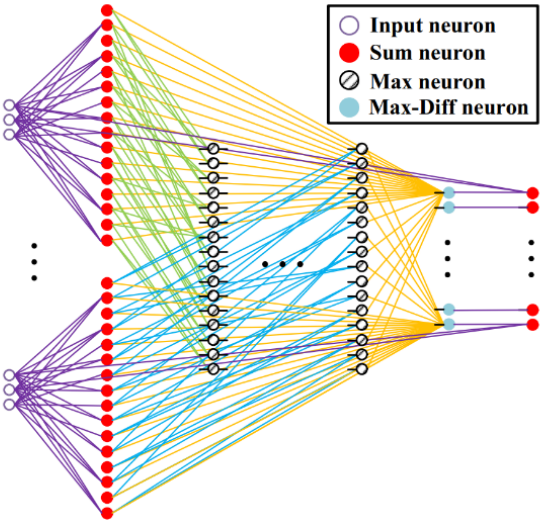

Figure 2.4: : Subnet architecture based on neural max-log-MAP algorithm, taken from [42].

The resulting TurboNet decoder exhibits better performance compared with the traditional max-log MAP algorithm for turbo decoding with different code rates (i.e., 1/2 and 1/3) and contains considerably fewer parameters compared with the neural BCJR decoder proposed in [40]. TurboNet is trained at a special signal-to-noise ratio (SNR) and its performance is similar or even slightly better than that of the max-log-MAP algorithm at a wide range of SNRs.

The same team later improved the TurboNet structure as described in [43], where trainable weights were introduced to the max-log-MAP algorithm and optimized through supervised learning. To further reduce the computational complexity and training cost of TurboNet, the network was pruned in order to significantly reduce the number of parameters without sacrificing error-correction performance. The pruned DNN decoder is called TurboNet+. The TurboNet and TurboNet+ decoders were validated with over-the-air experiments.

TinyTurbo

A refinement of the TurboNet+ approach was recently proposed in [44]. The proposed solution, called TinyTurbo, has many similarities with TurboNet+, but with the following key differences: the weights are entangled across bit positions and the optimal set of scaling weights is directly learnt from the data using an end-to-end loss function. From an error correction perspective, the resulting scaling

weights yield equal or better reliability results than TurboNet+ and successfully generalize across various channels, block lengths, code rates, and code trellises. As for complexity, TinyTurbo has a lower number of parameters than TurboNet+ and this number is independent of the block length. For comparison purpose, TinyTurbo has 18 trainable weights, independently of the block length, to be compared with the 720 weights of TurboNet+ for block length 40 and 3600 for block length 200, in a Gaussian channel. The TinyTurbo algorithm was also tested over the air and the experiments carried out have shown that TinyTurbo is robust to multipath fading.

2.2.2 AI-assisted decoding

AI-assisted early termination of iterative decoding

In the earliest publications we found on this topic [45, 46], it is shown that a neural network can be trained to monitor the cross entropy of the outputs of component decoders in a turbo decoder and to accurately predict the presence of errors in the decoded data at the end of the decoding process. A similar network can be trained to predict, early in the decoding process, whether or not the completion of the decoding process will result in erroneous data, allowing for early termination of decoding and a reduction in computational complexity. In both cases, the neural network can be used to trigger retransmission requests in a hybrid-ARQ protocol. This can replace the use of a cyclic-redundancy-check code.

A more recent paper proposes an early termination method for turbo and LDPC iterative decoders using a machine learning based algorithm that takes the soft decision (LLR) based metric as the input at an iteration, and predicts the decoding status of the received codeword at that iteration [47]. A fully connected deep neural network (DNN) is used, with an input layer of the same length as the message, three hidden layers and an output layer of length three. For the LTE turbo code and the extended pedestrian A channel model, the proposed algorithm outperforms the well-known hard decision aided early termination methods in reducing the average number of iterations by 25% to 57%, depending on the modulation and coding scheme (MCS) index.

AI-assisted decoding in trellis-based receivers

A very recent contribution was published about how to learn a trellis diagram in real-time using an artificial neural network (ANN) trained by a pilot sequence [48]. This approach, termed as the online learning trellis diagram (OLTD), requires neither the channel state information (CSI) nor statistics of the noise, and can be incorporated into the classic BCJR algorithm. Unfortunately, no performance result is given in [48] for the application of the OLTD method to turbo codes.

Meta learning for neural decoders

The introduction of meta learning to strengthen the adaptive property of neural decoders so that only minimal re-training is necessary was recently addressed in [49]. The proposed meta learning framework can be applied to neural turbo decoders. It operates in two steps: (a) it firstly performs a meta training phase by learning on a wide range of archetypal tasks, and then (b) during the meta testing phase enables learning new tasks faster, while consuming less adaptation data than learning from scratch. Supervised meta learning has a natural connection to adaptive decoder design, as different channels can be considered as different tasks in the meta learning framework.

2.2.3 Joint symbol detection and error correction with neural networks

The work presented in [50] proposes a BCJR receiver for joint symbol detection and channel decoding, called BCJRNet (see Fig. 2.5). The proposed approach is a hybrid one: instead of completely replacing the functional blocks with neural networks, this approach combines the advantages of both well-designed algorithms and neural networks, which can complement the defects of each other.

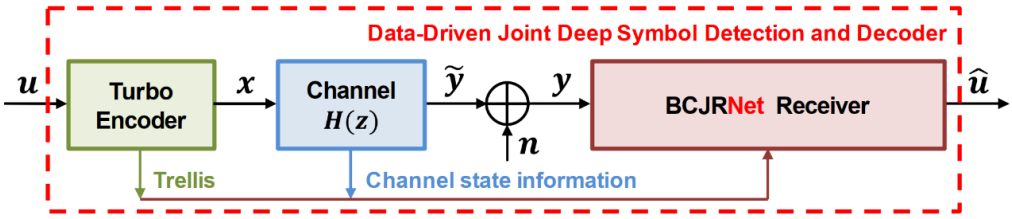

Figure 2.5: : BCJRNet receiver with joint symbol detection and channel decoding, taken from [50].

In the proposed BCJRNet receiver, the calculation of the branch probability is realized in a data-driven manner, and then the acquisition of branch probability in both the forward and backward recursions is carried out by inferencing the well-trained model. As illustrated in Fig. 2.6, rectified linear units (ReLU) are used by the first two layers and a softmax activation is used at the output layer to normalize the estimated branch probability.

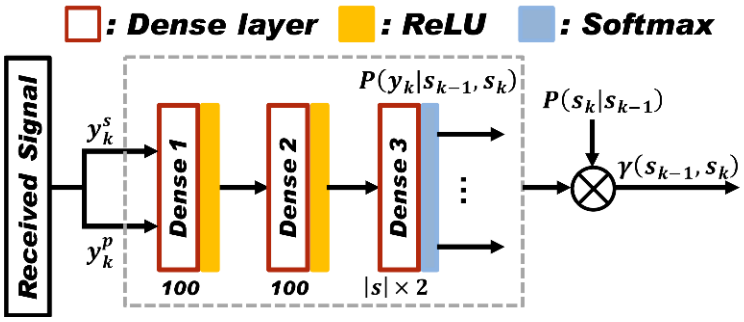

Figure 2.6: : Branch probability calculation for the proposed BCJRNet receiver, taken from [50].

The performance of the resulting receiver was compared with the one of a DeepTurbo of [41] applying 2-layer bidirectional RNN to replace the SISO block of BCJR algorithm. 1 dB gain was observed over DeepTurbo SISO block and the proposed BCJRNet receiver is shown to be more robust to CSI uncertainty than a conventional BCJR receiver.

2.2.4 Conclusion and takeaways

Several publications tackled the use of AI and ML techniques for the decoding of turbo codes and turbo coded modulations. In the prior art and up to now, it has been reported that AI-based solutions applied to jointly decode both component codes do not represent viable alternatives. Instead, a turbo processing approach with information exchange between AI-based component decoders was applied. One notable takeaway is the high additional complexity of the proposed schemes compared to the conventional BCJR algorithm. Given the complexity of the BCJR algorithm itself and the comparable performance achieved with more complex AI-based alternatives, the added complexity may be considered prohibitive to widespread adoption of these techniques. However, a special case can be made for applications targeting non-Gaussian channels, where some of these alternatives did show improved performance. That said, for a fair comparison with BCJR, the results should have also included BCJR-based decoders with soft input that benefits from improved Channel State Information (CSI) through learning techniques. The absence of this benchmark raises the question of the additional benefit from improved CSI on the one hand and from AI-based decoding on the other.

This analysis of prior art advocates for the study and proposal of reduced-complexity decoding algorithm alternatives with limited performance impact prior to the use of AI-based/augmented variants. Indeed, the study of such viable reduced complexity alternatives seems to be a prerequisite for an efficient introduction and use of AI techniques in this context.

2.3 Polar Codes

Polar codes are proven to be a channel-capacity achieving codes under successive cancellation (SC) decoding [51]. However, infinite-length polar codes are required to reach such performance. In order to improve the decoding performance for short codes, other decoding algorithms have been introduced, mainly divided into two classes: the SC-based decoders, and the BP-based decoders. Besides, deep learning techniques have recently been investigated to improve the error-performance as well as the time complexity of multiple decoders.

Let us consider first how ML can be used to improve BP-based decoding of polar codes. The min-sum algorithm induces lower error-performance when substituted to the sum-product algorithm. Thus, a neural offset min-sum (NOMS) algorithm was introduced in [20]. It uses multiple offset parameters that are optimized applying a neural network and shows similar error-performance as the neural sum-product algorithm in [14] with low training complexity. The same approach of the latter paper was presented in [52] and [15]. The multiple scaled BP decoder was trained over the unrolled polar decoding graph in [52] whereas [15] uses a RNN architecture. The results show better error-performance than the original BP decoder under fewer iterations, especially when the polar encoding graph is concatenated to a CRC graph [53]. In [54], a partitioned neural network decoder (PNN) was proposed. The PNN decoder takes advantage from the regular algebraic structure of polar codes. Thus, it partitions some stages of the polar encoding graph into independent neural network sub-graphs and connects them to the remaining stages using BP decoding. This approach reduces significantly decoding time complexity while showing similar performance as BP decoding. Similarly, the partitioned neural network sub-decoders can be connected to the remaining stages of the graph using SC decoding [55]. This reduces the latency introduced by BP decoding iterations, required to achieve an error-performance similar to SC algorithm. It has been shown that the proposed decoder outperforms the SC, BP, and PNN decoders by up to 89% in terms of time steps number. In [56], a RNN-based neural network sub-decoders are used instead of the fully-connected ones. The LSTM architecture improves the sub-decoder performances in terms of latency and BER. Furthermore, a pruning method was introduced to lower the decoding time complexity. The error-performance results have shown that the LSTM-based decoder outperforms the two previous partitioned ones. In [57], the NSC decoder of [55] is trained over a free space optical turbulence channel using tanh-modified LLR as trainable data. Over the FSO channel, the proposed decoder approaches the SC decoding error-performance while taking up to 25% less time steps.

Successive cancellation flip decoder performs further SC decoding attempts when the CRC verification fails. Each new decoding attempt proceeds to flipping the least reliable bit according to its corresponding LLR value. Dynamic successive cancellation flip (DSCF) decoding algorithm was introduced in [58] as a multiple bit-flipping approach based on a probabilistic metric that takes the SC sequential aspect into account. Neural DSCF, unlike DSCF, uses an additive parameter to approximate the logarithmic and exponential flipping metric, which is optimized using machine learning techniques. NDSCF has shown to be as performant as the ideal DSCF algorithm with an average complexity approaching the SC decoding. The work [59] is an extension of the previous paper. It uses a different training approach to optimize the additive parameter at each error order individually. In order to drive the parameter at a given error rate, only the frames that do not pass the CRC check during the previous execution are used which limits considerably the number of training samples. In [60], a flipping algorithm for SCL decoding is proposed. The bit-flipping metric is expressed depending on a correlation matrix that is optimized as a machine learning problem. The results of the proposed decoder outperforms the SCL decoding and are identical to [58] while requiring less operations. In [61], a LSTM network is used to construct the first erroneous bit-flipping set in a increasing order of reliability. Moreover, if the sequence still fail the CRC verification after the first flipping attempts, a RL-based method is used to find the best multi-flipping strategy. The proposed ML-MSCF decoder outperforms the DSCF decoder [58] at high SNR and shows a lower time complexity. In [62], a double LSTM-based SC flipping decoder is proposed to locate the first erroneous bits and construct the one-bit flipping set. Moreover, for the sequences that reach the maximum number of one-bit flipping and

still fail the CRC verification, a method based on channel reliability is used for multi-bit flipping. The proposed DLSTM-based SCF decoder has better error-correction performance than the ML-MSCF and DSCF decoders and approaches the performance of CA-SCL (L=8). The decoding complexity turns out to be lower than SCL (L=4) and DSCF. The flipping algorithm was considered for BP decoding as well [63], [64]. It uses a convolutional neural network whose training data are the BP polar decoding LLRs to output the flipping probabilities for each bit-position. [64] extends the work in [63] to multi-bit flipping attempts using a tree-based model. It enables to achieve an error-performance gain up to 0.5 dB.

Permuted factor graph structures yet another way to enhance polar-code error-performance. It has been proven in [65] that one single graph can be used to perform decoding on all the required permutations, simply by permuting the bits of the original decoder graph. As there is a considerable number of possible permutations, [66] uses a reinforcement learning method, referred to as a multi-armed bandit problem, in order to select the most effective ones. The decoding over the selected permutations was applied to CRC-Polar concatenated codes [53] and has shown a significant error-performance improvement.

2.4 Dense Codes

Most of ML-based methods that have been described in section 2.1.1 were initially devised to cope with low-complexity BP decoding of high-density parity-check codes, referred to as *dense codes*. In particular, BCH codes or Reed-Solomon codes (using their binary representations) were the first codes that have been considered to illustrate all the potential of neural BP based decoding strategies [15][35] to improve BP decoding performance, towards MLD. Since then, several strategies have been developed to improve performance for some specific class of codes, often inspired from existing regular BP strategies that are recast within the neural BP framework. For example, in [67], a deep learning strategy is applied to BP decoding of redundant Tanner graph for which redundant representation are generated based on some columns permutations of the initial parity-check matrix that maps the code onto itself, defining the so-called *automorphism group*. These permutations that are selected during the decoding process can be chosen randomly, or pre-determined carefully according to some given metrics[67]. The corresponding BP decoding process can be then unfolded and neural BP like optimization can be performed to optimize the weights of the corresponding neural network. Another strategy has been proposed by [68]. They use ML to train a neural network that enables to predict the performance of successful decoding based on the received codeword for the different possible permutations. From this prediction, a permutation is selected and BP decoding is performed. For even denser codes, such as Reed-Solomon codes, specific strategies have been also proposed. In [69], the authors combine neural BP with the stochastic shifting decoding strategy that exploits cyclic structure of RS codewords. Finally, [70] have investigated on adversarial neural networks based decoders that have shown promising results for both BCH and RS codes.

For another generic route to near-MLD of linear block codes, we may cite the syndrome-based approach of [71], in which a DNN is trained to predict the most-likely coset leader given the (binary) syndrome computed from the hard-decision sequence and the bit reliabilities. Similar to some of the previously mentioned decoding strategies, the proposed ML-aided syndrome-based decoder include a pre-processing step that takes advantage of the automorphism group of the code to permute the code bits in a way that boosts the DNN decoder performance.

3 Design Space Exploration for Improved ML-Aided Decoding

3.1 Design Space Exploration Matrix

Table 3.1 provides an overview of the established design space exploration for ML-aided decoding. It incorporates the targeted improvements and corresponding KPIs from Section 1, while informed by the latest developments in ML-aided FEC decoding reviewed in Section 2, and also builds upon the specific research experience and interests of the consortium. It therefore provides guidelines for the technical contributions that will be carried out in the project, which are outlined in Section 3.2.

It is worth noticing that the performance and complexity KPIs are strongly correlated. Thus, one KPI cannot be considered as the only means to evaluate a proposed solution, without any consideration of the other. For each targeted improvement, we therefore consider in Table 3.1 a primary KPI and a secondary one. This means that we target an improvement of the primary KPI, possibly with an acceptable degradation (within practical limits) of the second.

Also, evaluating the performance/complexity trade-off of an ML-augmented decoder requires not only assessing how modifications of the learning architecture and hyper-parameters may affect the KPIs, but also appraising the overhead induced by the learning augmentation relative the complexity of the vanilla decoder. Particular care will be paid to this question within the planned studies as it will condition latter the practicality of the proposed solutions.

Finally, for the case of Turbo codes, more gains are expected to come from a decoding-aware optimization of the encoder (rather than the optimization of the decoding algorithm itself). Thus, the consortium found more relevant to concentrate their effort on the ML-aided construction of Turbo codes (WP2). This is also in line with what has been anticipated in the scientific document of the project.

Table 3.1: Design Space Exploration Matrix

Code	Targeted improvement	Learning[(1)]	KPI[(2)]
LDPC	Enhancing the ML-aided BP decoding performance for binary LDPC codes	DL, RL	BER, FER Complexity
LDPC	Improving robustness to channel uncertainty or mismatch at runtime	ML, DL	BER, FER Complexity
LDPC	Optimizing hyper-parameters for BP or gradient descent based decoders	RL, DL	BER, FER Complexity
Polar	Developing reduced-complexity SC decoders for non-binary Polar codes	DL	Complexity BER, FER
Polar	Developing enhanced ML-aided SCL decoders for Polar Codes	RL	Complexity BER, FER
Polar Dense	Developing enhanced ML-aided non-binary BP decoders	DL	BER, FER Complexity

[(1)] DL = Deep-Learning, ML = Meta-Learning, RL = Reinforcement Learning.
[(2)] Primary KPI is shown in black, secondary KPI is shown in gray.

3.2 Outline of the Planned Contributions

3.2.1 Developing enhanced ML-aided BP decoders for binary LDPC codes

The BP decoding algorithm for binary LDPC codes has very low complexity, only linear in the block length, but falls short of releasing the full error-correction capability of short to very short binary LDPC codes. Many variations on the BP algorithm have been proposed to bridge this gap, such as using redundant graphs, decoding over multiple graph representations in parallel, using reliability-based pre/post-processing, or adding noise to the received word. As discussed in Chapter 2, the use of DNNs for iterative decoding has also received much attention recently, albeit with a focus primarily placed on codes defined by high-density bipartite graphs. The central idea consists in unfolding the iterations of the BP algorithm into a deep neural network architecture whose weights and biases can be trained to improve performance, giving rise to the so-called neural or weighted BP framework. Many subsequent refinements have followed, and other approaches have been explored as well. However, for now, there is still margin for improvement as for a typical short LDPC code such as the $(128, 64)$ code recommended for deep space telecommand systems, neural BP decoders can recover at best only about half of the gap to MLD, with an overall complexity comparable to or higher than standard BP. We therefore propose to investigate how to come closer to MLD of short binary LDPC with ML-augmented message-passing decoders, and at which cost in terms of computational complexity.

A first research lead will consist in specializing the decoder to difficult error events. Such error events depend on the code and the decoding algorithm, e.g., error events corresponding to trapping/absorbing sets under message passing decoding, and can be characterized by a careful analysis of both. As a unique decoder might not be able to deal with all such error events efficiently, we propose to group error events in "difficulty classes", and to specialize the training set, thus the trained decoder, according to the difficulty class. We can then optimize a bunch of decoders, which can be run either in parallel or sequentially, in order to approach MLD. A syndrome check or an extra cyclic-redundancy-check code can then be used to select the decoded message.

Rather than resorting to a bunch of "specialists", a second approach consists in calling on a single, more generalist ML-driven BP decoder that adapts the decoding process, online, to the received word. The idea is to improve upon the neural BP decoder by combining several of the refinements proposed to approach MLD with standard BP, such as flipping or saturating certain bits flagged as unreliable, or selecting a proper decoding graph (received word permutation). ML can then be used to learn and predict the perturbation that is most appropriate at each decoding step in order to maximize the chances of correct decoding. The neural BP weights and hyper-parameters of the ML-driven decoder augmentation can be trained separately or jointly, and both cases will be studied.

3.2.2 Improving robustness to channel uncertainty or mismatch at runtime

As pointed out in Section 1.1, channel decoders are built for a particular statistical model of the channel. The decoding metrics, and possibly the decoder architecture also, are matched to the channel. Thus channel decoding fundamentally follows a model-based approach. The problem, however, is that channel models are only an approximation of the reality. A real channel always deviates from the assumed model. This may entail some performance loss, which can be significant for decoders operating on soft inputs. The degradation can be even more severe for ML-augmented decoders such as the improved neural BP decoders to be developed in Sub-Section 3.2.1. Such neural decoders generally rely on many hyper-parameters that have been optimized offline with respect to a certain channel model and target SNR, without guarantee that the resulting decoder will generalize well online to noise and fading statistics never met during training.

Here we propose to investigate how to design robust decoders for binary LDPC codes, that use online learning to automatically adapt to unknown channel parameters, directly from the received data. In the case of message-passing decoders, model dependency only appears in the log-likelihood ratio (LLR) front-end that precedes the decoder and is in charge of computing soft-decisions from the channel output. Inspired by [72], a first line of action will be to study how to augment the LLR

computation front-end with meta-learning capabilities to obtain robust standard LDPC decoders that can automatically adapt to unknown SNR or fading statistics. In a second time, the proposed approach will be extended to improved neural BP decoders for binary LDPC codes, so that they can adapt and generalize well to unseen channels. One of the major challenges here is to adapt online and fast, as neural-based decoders may have a significant number of hyper-parameters, requiring, in principle, an expensive retraining phase based on a large amount of data. Solutions along the lines of those promoted in [49, 73], that take advantage of some channel-state information combined with meta-learning concepts and domain-adaptation techniques to minimize the retraining cost, will be explored and assessed.

3.2.3 Developing reduced-complexity SC decoders for non-binary Polar codes

A non-binary polar code works on the same principle as a binary polar code, except that operation are done on $\mathrm{GF}(q)$, $q = 2^p$, instead of $\mathrm{GF}(2)$. It has been shown in [74] that the decoding of the non-binary parity checks of a polar code can be simplified, without significant degradation of the error-correction performance. Such a simplification was obtained by reducing the size of the non-binary alphabet, which has been tuned by empirical trials. Here, we propose to apply machine learning to develop a generic optimization method, in which the size of the non-binary alphabet may vary depending on the particular position of the corresponding node in the decoding graph. The result of ML-based optimization will give us useful insights to increase the hardware efficiency of non-binary polar decoders.

3.2.4 ML-aided SCL decoding of Polar Codes

The Successive Cancellation List (SCL) proposed in [75] relies on L SC decoding paths. At each step of the SCL decoding, only the L most likely paths are kept, the others are discarded. Hence, the SCL decoder fails either if the most likely path is not the correct one (in this case the maximum likelihood decoder would also fail), or if the correct path is pruned at some decision step. We propose to use machine learning (ML) to improve the path selection problem. There are two possible strategies for improvement:

- ML enhanced SCL decoder could locally increase the list size when required and hence eventually avoid the correct path to be removed from the list.
- ML could be used to reduce the size of the list by removing uninteresting paths whenever it is possible.

In addition, a recent approach [76] has shown that deep neural networks can be used to predict the mapping between Frame Error Rate and the frozen bits pattern. Once this deep neural network is trained, Projected Gradient Descent is then used to build frozen bit sequences for a given frame error rate target. We propose to extend the method proposed in [76] for designing good frozen bits strategies for the ML enhanced SCL decoder. This part of the work will cover both WP2 and WP3.

3.2.5 ML-augmented non-binary BP decoding

ML-augmented non-binary BP decoding can be considered from two different points of view.

First, we can consider the ML augmented decoding of codes defined on higher field orders and try to extend existing methods for the binary case. If neural based iterative decoding has been widely addressed for binary error correcting codes, especially for medium to dense parity-check matrices, few studies have considered the application of neural BP like algorithms to non-binary coding schemes using the ground field representation. For dense codes such as Reed-Solomon codes, neural BP decoding is usually applied on the binary image representation in order to apply different kind of neural BP decoding strategies that are directly inspired by decoding strategies used for the regular BP decoding. Thus, one open issue is to derive vector neural BP like strategies based on the nonbinary parity check matrix and the corresponding unfolded Tanner graph. Another strategy is to exploit the sparse graph

representation as given by [77], for codes for which this representation exists, which has some connections to the polar codes factor graph. Capitalizing on strategies proposed for deep-learning based BP decoding of polar codes, we could try to derive efficient methods that exploit this representation. We can also try to improve performance of short non binary LDPC codes, beyond ultra-sparse codes which have been shown to have poor codeword weight supports, despite their good performance under BP decoding.

Second, it has been shown in [78] that non binary decoding on higher order group extension of binary dense codes helps to bridge the gap between the binary BP decoding of these codes and the ML performance. One can think of designing ML-augmented decoding algorithms that consider optimized unfolded non binary decoding of dense codes on the corresponding equivalent non binary graph (or on a modified version of it).

4 General Conclusion

The last 6 years or so have witnessed a growth of interest and research efforts towards applying deep learning techniques to channel coding problems in general, and to the specific task of channel decoding in particular. Yet, as of today, it is fair to recognize that the major breakthrough is still pending, and that we are still in search of that *unreasonable effectiveness* shown by deep learning in other fields, that make it so popular and invaluable now[79].

This deliverable pursued a threefold objective:

- To specify targeted improvements and corresponding KPIs, for communication scenarios where the performance or complexity of FEC decoders do not meet the expected requirements and could be improved by ML-based algorithms.
- To provide a systematic review of the recent literature on ML-based FEC decoders, summarizing the latest advances in ML-augmented decoding of LDPC, Turbo, Polar, and more general linear block codes.
- To establish a design space exploration for ML-augmented decoding, providing guidelines for improved designs of FEC decoders to be developed in the project.

Accordingly, a number of technical contributions have been identified for the remaining of the project, which, simplifying, fall within one of the following two categories.

- Developing enhanced ML-aided FEC decoders, or improving their robustness to channel uncertainty or mismatch at runtime. The main focus here is on the error-correction performance, while ensuring that complexity considerations are also properly taken into account.
- Developing reduced complexity decoders for non-binary codes. The main focus here is on reducing the decoding complexity, while ensuring negligible or minimal degradation of the error-correction performance.

Bibliography

[1] E. Berlekamp, R. Peile, and S. Pope, "The application of error control to communications," *IEEE Communications Magazine*, vol. 25, no. 4, pp. 44–57, 1987.

[2] J. Van Wonterghem, A. Alloum, J. J. Boutros, and M. Moeneclaey, "On short-length error-correcting codes for 5G-NR," *Ad Hoc Networks*, vol. 79, pp. 53–62, 2018.

[3] M. C. Coşkun, G. Durisi, T. Jerkovits, G. Liva, W. Ryan, B. Stein, and F. Steiner, "Efficient error-correcting codes in the short blocklength regime," *Physical Communication*, vol. 34, pp. 66–79, 2019.

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

[5] C. Yue, V. Miloslavskaya, M. Shirvanimoghaddam, B. Vucetic, and Y. Li, "Efficient decoders for short block length codes in 6G URLLC," *arXiv preprint arXiv:2206.09572*, 2022.

[6] B. Lian and F. R. Kschischang, "Sequential decoding of short length binary codes: Performance versus complexity," *IEEE Communications Letters*, vol. 25, no. 10, pp. 3195–3198, 2021.

[7] A. Fazeli, A. Vardy, and H. Yao, "List decoding of polar codes: How large should the list be to achieve ML decoding?" in *2021 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2021, pp. 1594–1599.

[8] M. C. Coşkun and H. D. Pfıster, "An information-theoretic perspective on successive cancellation list decoding and polar code design," *IEEE Transactions on Information Theory*, vol. 68, no. 9, pp. 5779–5791, 2022.

[9] D. Truhachev, K. El-Sankary, A. Karami, A. Zokaei, and S. Li, "Efficient implementation of 400 Gbps optical communication FEC," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 1, pp. 496–509, 2020.

[10] P. Mohr, G. Bauch, F. Yu, and M. Li, "Coarsely quantized layered decoding using the information bottleneck method," in *ICC 2021-IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6.

[11] J. Jiang and K. R. Narayanan, "Iterative soft-input soft-output decoding of Reed–Solomon codes by adapting the parity-check matrix," *IEEE Transactions on Information Theory*, vol. 52, no. 8, pp. 3746–3756, 2006.

[12] I. Dimnik and Y. Be'ery, "Improved random redundant iterative HDPC decoding," *IEEE Transactions on Communications*, vol. 57, no. 7, pp. 1982–1985, 2009.

[13] G. Liva, G. Durisi, M. Chiani, S. S. Ullah, and S. C. Liew, "Short codes with mismatched channel state information: A case study," in *2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2017, pp. 1–5.

[14] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *IEEE Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2016, pp. 341–346.

[15] 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, 2018.

[16] M. Lian, F. Carpi, C. Häger, and H. D. Pfister, "Learned belief-propagation decoding with simple scaling and SNR adaptation," in *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2019, pp. 161–165.

[17] X. Chen and M. Ye, "Cyclically equivariant neural decoders for cyclic codes," *arXiv preprint arXiv:2105.05540*, 2021.

[18] A. Buchberger, C. Häger, H. D. Pfister, L. Schmalen, and A. G. i Amat, "Pruning and quantizing neural belief propagation decoders," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 1957–1966, 2020.

[19] ——, "Learned decimation for neural belief propagation decoders," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 8273–8277.

[20] L. Lugosch and W. J. Gross, "Neural offset min-sum decoding," in *IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 1361–1365.

[21] L. Wang, S. Chen, J. Nguyen, D. Dariush, and R. Wesel, "Neural-network-optimized degree-specific weights for LDPC MinSum decoding," in *2021 11th International Symposium on Topics in Coding (ISTC)*. IEEE, 2021, pp. 1–5.

[22] J. Dai, K. Tan, Z. Si, K. Niu, M. Chen, H. V. Poor, and S. Cui, "Learning to decode protograph LDPC codes," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 1983–1999, 2021.

[23] N. Shah and Y. Vasavada, "Neural layered decoding of 5G LDPC codes," *IEEE Communications Letters*, vol. 25, no. 11, pp. 3590–3593, 2021.

[24] B. A. Jayawickrama and Y. He, "Improved layered normalized min-sum algorithm for 5G NR LDPC," *IEEE Wireless Communications Letters*, vol. 11, no. 9, pp. 2015–2018, 2022.

[25] S. Habib, A. Beemer, and J. Kliewer, "Belief propagation decoding of short graph-based channel codes via reinforcement learning," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 2, pp. 627–640, 2021.

[26] ——, "RELDEC: Reinforcement learning-based decoding of moderate length LDPC codes," *arXiv preprint arXiv:2112.13934*, 2021.

[27] B. Vasić, X. Xiao, and S. Lin, "Learning to decode LDPC codes with finite-alphabet message passing," in *IEEE Information Theory and Applications Workshop*, 2018, pp. 1–9.

[28] X. Xiao, N. Raveendran, B. Vasić, S. Lin, and R. Tandon, "FAID diversity via neural networks," in *2021 11th International Symposium on Topics in Coding (ISTC)*. IEEE, 2021, pp. 1–5.

[29] M. Stark, J. Lewandowsky, and G. Bauch, "Neural information bottleneck decoding," in *2020 14th International Conference on Signal Processing and Communication Systems (ICSPCS)*. IEEE, 2020, pp. 1–7.

[30] E. Nachmani and L. Wolf, "Hyper-graph-network decoders for block codes," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[31] S. Cammerer, J. Hoydis, F. A. Aoudia, and A. Keller, "Graph neural networks for channel decoding," *arXiv preprint arXiv:2207.14742*, 2022.

[32] K. Tian, C. Yue, C. She, Y. Li, and B. Vucetic, "A scalable graph neural network decoder for short block codes," *arXiv preprint arXiv:2211.06962*, 2022.

[33] Y. Choukroun and L. Wolf, "Error correction code transformer," *arXiv preprint arXiv:2203.14966*, 2022.

[34] R. Dong, F. Lu, Y. Dong, and H. Yan, "The negative BER loss function for deep learning decoders," *IEEE Communications Letters*, 2022.

[35] I. Be'Ery, N. Raviv, T. Raviv, and Y. Be'Ery, "Active deep decoding of linear codes," *IEEE Transactions on Communications*, vol. 68, no. 2, pp. 728–736, 2019.

[36] E. Nachmani and Y. Be'ery, "Neural decoding with optimization of node activations," *IEEE Communications Letters*, 2022.

[37] H. Lee, Y.-S. Kil, M. Y. Chung, and S.-H. Kim, "Neural network aided impulsive perturbation decoding for short raptor-like LDPC codes," *IEEE Wireless Communications Letters*, vol. 11, no. 2, pp. 268–272, 2021.

[38] F. Carpi, C. Häger, M. Martalò, R. Raheli, and H. D. Pfister, "Reinforcement learning for channel coding: Learned bit-flipping decoding," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2019, pp. 922–929.

[39] S. Han, J. Oh, K. Oh, and J. Ha, "Deep-learning for breaking the trapping sets in low-density parity-check codes," *IEEE Transactions on Communications*, vol. 70, no. 5, pp. 2909–2923, 2022.

[40] H. Kim, Y. Jiang, R. Rana, S. Kannan, S. Oh, and P. Viswanath, "Communication algorithms via deep learning," *arXiv preprint arXiv:1805.09317*, May 2018.

[41] Y. Jiang, S. Kannan, H. Kim, S. Oh, H. Asnani, and P. Viswanath, "Deepturbo: Deep turbo decoder," in *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. Cannes, France: IEEE, Jul. 2019, pp. 1–5.

[42] Y. He, J. Zhang, C.-K. Wen, and S. Jin, "TurboNet: A model-driven DNN decoder based on max-log-MAP algorithm for turbo code," in *2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS)*. Singapore: IEEE, Aug. 2019, pp. 1–5.

[43] Y. He, J. Zhang, S. Jin, C.-K. Wen, and G. Y. Li, "Model-driven dnn decoder for turbo codes: Design, simulation, and experimental results," *IEEE Transactions on Communications*, vol. 68, no. 10, pp. 6127–6140, 2020.

[44] S. A. Hebbar, R. K. Mishra, S. K. Ankireddy, A. V. Makkuva, H. Kim, and P. Viswanath, "Tinyturbo: Efficient turbo decoders on edge," in *2022 IEEE International Symposium on Information Theory (ISIT)*. Espoo, Finland: IEEE, Jun. 2022, pp. 2797–2802.

[45] M. E. Buckley and S. B. Wicker, "A neural network for predicting decoder error in turbo decoders," *IEEE Communications Letters*, vol. 3, no. 5, pp. 145–147, 1999.

[46] ——, "The design and performance of a neural network for predicting turbo decoding error with application to hybrid ARQ protocols," *IEEE Transactions on Communications*, vol. 48, no. 4, pp. 566–576, 2000.

[47] A. Gunturu, A. Agrawal, A. K. R. Chavva, and S. Pedamalli, "Machine learning based early termination for turbo and LDPC decoders," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*. Nanjing, China: IEEE, Mar. 2021, pp. 1–7.

[48] J. Yang, Q. Du, and Y. Jiang, "Neural network-assisted receiver design via learning trellis diagram online," *IEEE Transactions on Communications*, 2022.

[49] Y. Jiang, H. Kim, H. Asnani, and S. Kannan, "Mind: Model independent neural decoder," in *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. Cannes, France: IEEE, Jul. 2019, pp. 1–5.

[50] W.-C. Tsai, C.-F. Teng, H.-M. Ou, and A.-Y. A. Wu, "Neural network-aided bcjr algorithm for joint symbol detection and channel decoding," in *2020 IEEE Workshop on Signal Processing Systems (SiPS)*. Coimbra, Portugal: IEEE, Oct. 2020, pp. 1–6.

[51] 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, 2009.

[52] 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, 2017, pp. 1–6.

[53] N. Doan, S. A. Hashemi, E. N. Mambou, T. Tonnellier, and W. J. Gross, "Neural belief propagation decoding of CRC-polar concatenated codes," *IEEE International Conference on Communications (ICC)*, p. 6, 2019.

[54] S. Cammerer, T. Gruber, J. Hoydis, and S. Ten Brink, "Scaling deep learning-based decoding of polar codes via partitioning," in *GLOBECOM 2017-2017 IEEE global communications conference*. IEEE, 2017, pp. 1–6.

[55] N. Doan, S. A. Hashemi, and W. J. Gross, "Neural successive cancellation decoding of polar codes," in *2018 IEEE 19th international workshop on signal processing advances in wireless communications (SPAWC)*. IEEE, 2018, pp. 1–5.

[56] J. G. Guiping Li, Xiuhua Hu, "A recurrent neural network-based succession cancellation for polar decoder," *International Journal of Innovative Computing, Information and Control*, 2021.

[57] J. Fang, M. Bi, S. Xiao, H. Yang, Z. Chen, Z. Liu, and W. Hu, "Neural successive cancellation polar decoder with tanh-based modified LLR over FSO turbulence channel," *IEEE Photonics Journal*, vol. 12, no. 6, pp. 1–10, 2020.

[58] L. Chandesris, V. Savin, and D. Declercq, "Dynamic-scflip decoding of polar codes," *IEEE Transactions on Communications*, vol. 66, no. 6, pp. 2333–2345, 2018.

[59] N. Doan, S. A. Hashemi, F. Ercan, T. Tonnellier, and W. J. Gross, "Neural successive cancellation flip decoding of polar codes," *Journal of Signal Processing Systems*, vol. 93, no. 6, pp. 631–642, 2021.

[60] S. A. Hashemi, N. Doan, T. Tonnellier, and W. J. Gross, "Deep-learning-aided successive-cancellation decoding of polar codes," *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, 2019.

[61] B. He, S. Wu, Y. Deng, H. Yin, J. Jiao, and Q. Zhang, "A machine learning based multi-flips successive cancellation decoding scheme of polar codes," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*. IEEE, 2020, pp. 1–5.

[62] J. Cui, W. Kong, X. Zhang, D. Chen, and Q. Zeng, "Dlstm-based successive cancellation flipping decoder for short polar codes," *Entropy*, vol. 23, no. 7, p. 863, 2021.

[63] C.-F. Teng, A. K.-S. Ho, C.-H. D. Wu, S.-S. Wong, and A.-Y. A. Wu, "Convolutional neural network-aided bit-flipping for belief propagation decoding of polar codes," *Journal of Signal Processing Systems*, 2021.

[64] C.-F. Teng and A.-Y. A. Wu, "Convolutional neural network-aided tree-based bit-flipping framework for polar decoder using imitation learning," *IEEE Transactions on Signal Processing*, 2021.

[65] N. Doan, S. A. Hashemi, M. Mondelli, and W. J. Gross, "On the decoding of polar codes on permuted factor graphs," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–6.

[66] W. J. G. Nghia Doan, Seyyed Ali Hashemi†, "Decoding polar codes with reinforcement learning," *GLOBECOM 2020-2020 IEEE Global Communications Conference*, 2020.

[67] B. Liu, Y. Xie, and J. Yuan, "A deep learning assisted node-classified redundant decoding algorithm for BCH codes," *IEEE Transactions on Communications*, vol. 68, no. 9, pp. 5338–5349, 2020.

[68] 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.

[69] W. Zhang, S. Zou, and Y. Liu, "Iterative soft decoding of reed-solomon codes based on deep learning," *IEEE Communications Letters*, vol. 24, no. 9, pp. 1991–1994, 2020.

[70] H. T. Nguyen, S. Bottone, K. T. Kim, M. Chiang, and H. V. Poor, "Adversarial neural networks for error correcting codes," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 01–06.

[71] 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, 2018, pp. 1595–1599.

[72] R. Li, O. Bohdal, R. Mishra, H. Kim, D. Li, N. Lane, and T. Hospedales, "A channel coding benchmark for meta-learning," *arXiv preprint arXiv:2107.07579*, 2021.

[73] M. Benammar, E. D. C. Gomes, and P. Piantanida, "CSI-aided robust neural-based decoders," in *2021 11th International Symposium on Topics in Coding (ISTC)*. IEEE, 2021, pp. 1–5.

[74] F. Cochachin, L. Luzzi, and F. Ghaffari, "Reduced complexity of a successive cancellation based decoder for NB-polar codes," in *2021 11th International Symposium on Topics in Coding (ISTC)*, 2021, pp. 1–5.

[75] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2213–2226, 2015.

[76] M. Léonardon and V. Gripon, "Using deep neural networks to predict and improve the performance of polar codes," in *2021 11th International Symposium on Topics in Coding (ISTC)*, 2021, pp. 1–5.

[77] J. Yedidia, "Sparse factor graph representations of Reed-Solomon and related codes," in *DIMACS Workshop on Algebraic Coding*, Dec. 2003. [Online]. Available: https://www.merl.com/publications/TR2003-135

[78] A. Goupil, M. Colas, G. Gelle, and D. Declercq, "FFT-based BP decoding of general LDPC codes over abelian groups," *IEEE Trans. on Communications*, vol. 55, no. 4, pp. 644–649, 2007.

[79] T. J. Sejnowski, "The unreasonable effectiveness of deep learning in artificial intelligence," *Proceedings of the National Academy of Sciences*, vol. 117, no. 48, pp. 30 033–30 038, 2020.