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 D2.1
Specification of the code design problems

Editor:	Charly Poulliat (Toulouse INP)
Deliverable nature:	Public
Due date:	January 31, 2023
Delivery date:	April 14, 2023
Version:	1.0
Total number of pages:	22 pages
Keywords:	

Abstract
This deliverable reports on the activities carried out within the Task 2.1 of Work Package 2 (WP2). WP2 aims at investigating how learning techniques can help come up with new code design paradigms or discover new code constructions, with application to selected communication scenarios that impose challenges on FEC code design that are not fully met with existing methods. Accordingly, the deliverable first discusses the opportunities and challenges for a learning-based approach to channel code design. A review of the current state-of-the-art then follows. Areas for potential improvements are identified for each of the major code families in use in current standards, and will serve as a starting point for the forthcoming studies.

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)
	Mathieu Leonardon (mathieu.leonardon@imt-atlantique.fr)
	Catherine Douillard (catherine.douillard@imt-atlantique.fr)
	Ahmad Ismail (ahmad.ismail@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	Iryna Andriyanova (iryna.andriyanova@ensea.fr)
	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 in FEC decoding.

Objective #2: Investigate how ML 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 2 (WP2) of the AI4CODE project investigates how learning techniques can help come up with new code design paradigms or discover new code constructions with application to selected communication scenarios that place challenges on FEC code design that existing methods can address only partially at best.

The Gantt chart of WP2 is represented in Fig. 1. This deliverable reports on the activities carried out within the **Task 2.1**, aimed at specifying code design problems for which ML can be a game changer. The main objectives are as follows:

- Identify communication problems and opportunities for a learning-based code design approach,
- Review the state-of-the-art (SoA) for learning-based code design methods,
- Identify limitations and weaknesses, and highlight areas of potential improvements

Accordingly, the deliverable is organized as follows.

Section 1 formalizes the communication scenarios where the performance or decoding complexity of existing FEC codes do not meet the expected requirements and could be improved by ML-assisted design, and specifies the targeted improvements/

Section 2 provides a broad state-of-the-art review of ML-assisted FEC design, with an emphasis on recent advances in ML-based design of Low-Density Parity-Check (LDPC), Turbo, Polar, dense (*e.g.*, algebraic) codes and coded modulations (CM).

Section 3 lists the areas for potential improvements have been identified for each code family, and that will be explored in the forthcoming studies related to this WP.

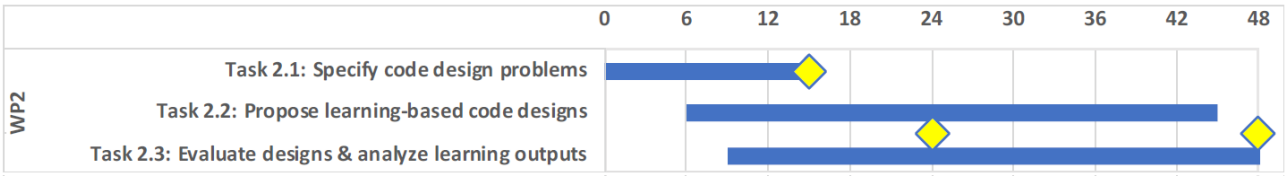

Figure 1: Gantt diagram of WP2

1 Opportunities and challenges for ML-based code design

1.1 Why learning to code ? The many opportunities raised by AI

Recent years have seen a flurry of interest in the field of communications to investigate deep learning methods for physical layer design and resource allocation in communication networks [1, 2]. In particular, learning techniques are expected to be one of the key enablers to accommodate the many envisioned use cases and related requirements of future beyond 5G (B5G) and 6G systems [3].

1.1.1 Incentives and obstacles for trainable communications

At the heart of many of these approaches is the concept of trainable communication systems, which rethink communications system design as a transmitter-receiver joint optimization problem for which a global solution can be learned, rather than as a succession of separate processing block where each block is designed and optimized more or less independently of the others [1, 4]. The trainable approach to communication system design is facilitated by the existence of off-the-shelf, comprehensive dedicated learning libraries, such as the recent Sionna open-source library for rapid prototyping and testing of complex communication system architectures that incorporate neural networks [5]. The fundamental learning architecture at the core of many of these learned communication signals approaches is the auto-encoder (AE) architecture, whose application to this context was first championed by [1]. The holistic trainable approach may not necessarily result in performance improvement over the traditional communication engineering paradigm, as long as the models used to design the individual processing blocks in the separate approach accurately reflect real-world operation conditions. Very often, however, reality happens to deviate substantially from the assumed models, especially in wireless transmissions, resulting in sub-optimal communication systems. Even if a good model is known, the fundamental limits on achievable performance and the coding and signaling techniques to achieve them may not. In that case trainable communication systems are in their rightful place. Also, trainable communication systems may result in integrated transceivers of lower overall complexity, and that can be easily adapted in a data-driven manner. But for as promising as they may look, those potential benefits should not obliterate the respective advantages of the classical, divide-and-conquer engineering approach, which has long proved its worth over the past 70 years. Let us mention only one, but of utmost importance: interpretability in case of misoperation, with the possibility to trace back the origin of the problem down to the faulty signal processing block, and to fix it without compromising inevitably the whole design.

1.1.2 Better matching with the channel

In the context of channel coding, the bulk of the work on ML has focused on using neural decoders to improve the decoding of existing linear codes. This line of research is the main topic of WP3 and an overview of the corresponding state-of-the-art is provided in AI4CODE Deliverable D3.1 [6]. But recent years have also witnessed a rapidly growing interest in the application of learning techniques to the search for new channel codes, see *e.g.* the overview in [7]. What good can ML be for channel code design, given that modern FEC codes such as turbo codes, LDPC or polar codes can already approach capacity within a tenth of decibels and with workable decoders ?

First, the latter only holds true for certain abstract memoryless channel models, that are idealized version of real operating conditions. Modern wireless systems compensate partly for the unavoidable mismatch between the model assumed for code design and the real channel, by modulation and code rate adaptation at the transmitter, and by channel-state information estimation and metric adaptation at the decoder. But this assumes a feedback link and low-latency communications. Similarly, interleaving is used to turn channels with memory into approximate memoryless channels at the code-

word level, but at the cost of latency and memory. Consequently there is a real incentive and stake at developing channel codes that can adapt automatically to unknown noise and fading conditions.

As regard to the latter point, the design of feedback codes is one of the few areas in channel coding where there is much room left for improvement on closing the gap between theory and practice. Capacity-achieving feedback coding strategies with optimal decay of error-probability with block length have long been known for the archetypal channels of coding theory, such as the binary symmetric channel (BSC) or the additive white gaussian noise (AWGN) channels. However they often rely on unrealistic or impractical assumptions, such as noiseless or unit-time delayed feedback. The assumption of noiseless feedback, in particular, has long been recognized as the Achilles' heel of the information-theoretic study of feedback. As of today, the optimal coding scheme for communicating over an AWGN channel with AWGN noisy feedback is still unknown. On the other hand, it is known that *linear* feedback codes are not sufficient for that purpose [8], calling for more elaborate non-linear coding schemes and sequential retransmission protocols that make an ideal playground for deep learning and reinforcement learning approaches.

1.1.3 Better matching with the decoder

Going back to the classical, feedforward point-to-point communication scenario, another mismatch arises at medium-to-short block length with modern FEC codes such as Turbo, LDPC or Polar codes, namely a (code, decoder)-pair mismatch. It has been recognized and demonstrated multiple times that the performance of such codes under classical message-passing decoding degrades significantly as the block length decreases, see *e.g.* the code benchmarks in [9]. This is not so much the result of weaknesses inherent to the code itself than the symptom of a mismatch between the code design and the message-passing decoder. Part of this mismatch comes from the heuristics and metrics used by most LDPC or Turbo code design methods, which fail to appropriately capture the error-rate performance of iterative decoding at short block length. The convergence of iterative decoding is hard to analyze in this regime, as simplifying assumptions relying on statistical averaging and independence arguments no longer hold for very short codewords. In addition, focusing on the parameters of the code (*e.g.* minimum distance) or of the bipartite graph (*e.g.* trapping sets) that govern iterative decoding performance at high signal-to-noise ratio (SNR) is no longer sufficient. Instead, one has to identify and fix the most critical code or graph weaknesses that hinder performance at low SNR as well, which is much more challenging. The same problem arises with finite-length Polar code. Polar code design most often assumes infinite-length codes under SC decoding. In practice, however, systems such as 5G New Radio use Polar codes of short to moderate block length, supplemented with an outer cyclic-redundancy check (CRC) code, and decoded by means of a successive-cancellation list (SCL) decoder. In this context, ML may prove useful in learning codes that are better matched to (= that perform well when decoded by) a selected target decoder. Learning better (code, decoder)-pairs is a fundamental stake in coding theory, that extends beyond the sole realm of iteratively-decodable FEC codes in the short-block-length regime.

1.1.4 Being more efficient at constructing capacity-approaching codes

Even today, more than 20 years after their discovery, the design of rate- and length-flexible capacity-approaching schemes such as the 4G turbo codes or the 5G LDPC and polar codes remains a blend of art and science that still relies, to a large extent, on intensive computer search driven by expert knowledge. Most design procedures for capacity-approaching codes follow a two-step approach: 1) find an ensemble of codes that is good for the target channel model and decoder, 2) pick up a good finite-length code within this ensemble. While good, computationally-efficient procedures are known for step 1), at least for large code length, step 2) usually requires searching over very large dimensional spaces, precluding systematic exploration. Some form of heuristic optimization or partial search is used instead. In addition, most often, there is no direct relation between the code- or graph-related parameters that are used to guide the search at steps 1 and 2, and the finite-length performance metrics of interest, *e.g.* BER or FER, calling for time-intensive monte-carlo simulation to discriminate

between the list of candidate codes returned by the search. ML may contribute in making existing design methods more efficient by smarter, faster exploration of the space of candidate codes.

1.1.5 Finding new codes

But perhaps the highest demands and expectations concern first and foremost all those coding problems for which optimal coding strategies are still to be found, especially those for which some form of non-linear coding is required. The example was given of the design of noisy feedback codes, but there are a few other communication scenarii for which linear codes cannot provably achieve the best achievable rates, at least by a direct approach. The non-input-symmetric discrete memoryless channel is a single-user example of such scenario [10], whereas the binary adder channel is another example but in the context of multi-user communications [11].

Nowadays, the need for coding extends far beyond its traditional role and place at the physical layer. The rise of network communication paired with the advent of new applications such as cloud storage, distributed learning, caching, or fog computing, prompt the need for novel coding paradigms, for example to recover from nodes failures in distributed storage and computing. Here, usual code parameters such as rate or minimum distance step aside in favour of more relevant application-oriented metrics such as locality, availability, and update efficiency. Beyond ensuring reliable communication or computing, code design has now to address new goals such as maximizing the number of users that can be simultaneously served by the system, or minimizing the expected service time [12]. Little is known, to date, about the optimal achievable trade-off in many such scenarios, and the design of efficient and practical codes for implementation in real systems is a challenging problem and an active area of on-going research in which machine learning may also have an important role to play.

1.1.6 Or improving upon existing ones

It is not because linear codes are sufficient to achieve capacity, for example on discrete memoryless channels, that this forbids the possibility to improve upon them with non-linear codes, in terms of reliability-length trade-off for instance [13, 14]. A major open question is thus whether ML can find better alternatives to the classical algebraic [15] , convolutional [16], and graph-based codes [14] in use in all digital communication and storage systems to date.

Also, there are coding theory areas for which optimal solutions are known but sub-optimal approaches are preferred in practice, since the latter may provide additional benefits that outweigh the loss. The design of signal-space codes by the combination of coding and modulation is a typical example. Albeit not capacity-achieving, the bit-interleaved coded modulation (BICM) paradigm that decouples binary coding and symbol mapping through interleaving has become ubiquitous in wireless systems. To a large extent, this is due to the remarkable simplicity with which one can adapt the code rate by simple adaptation of constellation size and puncturing of code bits. In principle, coded modulation where coding and symbol mapping are jointly optimized can result in better codes, but usually not as flexible as the former. Also, symbol mapping is a non-linear operation that complicates the analysis and design of coded modulation. Consequently, not only can deep learning help improve BICM [4], but it may also facilitate the design of better signal-space codes and related decoders, capable of achieving coding, shaping, and possibly also diversity gains with a generic, unified coding architecture, without sacrificing rate-flexibility. The so-called KO codes recently introduced in [14] and constructed in a recursive manner, reminiscent from the Plotkin construction of Reed-Muller codes, may be a first step in that direction.

1.2 Challenges ahead

We have seen that AI may benefit to coding theory in a variety of ways. However this also comes with its own daunting challenges. We'll highlight three of them.

The first one is the curse of code dimension. Direct learning of an (n, k) generic, unstructured block code requires learning 2^k codewords embedded into an n-dimensional space. As the code dimension

k grows, the amount of data needed for accurate training grows also, but exponentially. Being able to scale-up the learning to code parameters relevant for practical applications makes it mandatory to introduce proper structure in the code design, and to leverage upon that structure for the training. For example, focusing on linear codes and learning a generator or parity-check matrix instead of the whole code does help to an extent, but is by no means not sufficient. All learned generic linear codes that we are aware of have dimension $k < 100$ bits. The fact that, even with linear codes, the search space remain formidable, is part of the reason. But the crux of the problem might lie elsewhere: perhaps the learning architecture itself is at stake. Channel codes, especially linear codes, have a high degree of mathematical and topological structure. We would expect to find those structural and symmetry properties reflected somehow in the learning architecture. Yet most work so far have mainly considered off-the-shelf learning models. Maybe the most fundamental problem is to find the right architecture to learn codes.

There is another peculiar feature that differentiate the learning of channel codes from more traditional machine-learning applications. For most code design problems, training data, for example code samples, can be easily generated. Thus constructing a dataset is not an issue. On the other hand, evaluating a sample in the dataset for the training can be. If the code rate and length are fixed, as happens frequently, then the bit or frame error rate after decoding is usually the relevant figure of merit for code design. The problem is that evaluating such metrics may requires minutes, hours, or even days, depending on the operating SNR. It follows that the choice of the right loss function to make the training practical and relevant at the same time is a non-trivial problem [17, 18].

On a very practical level, learning channel codes also raises a number of technical issues, among which differentiability of the models stands out. Standard deep learning software libraries were not designed for binary signals and finite field operations. One may enforce binary weights and signals by some binarization operation, *e.g.* taking the sign. The problem is that most binarization operators are highly discontinuous functions, not differentiable or with a zero derivative almost everywhere but in a few points. Therefore, the usual gradient-descent-based back-propagation techniques cannot be applied as is to update the binary weights. There are known workarounds to this problem, starting from the straight-through estimator [19], but no well-established universal solution so far.

1.3 Scope of the AI4CODE project

The AI4CODE project does not intend to attack head-on all the open questions and challenges raised above. Instead, based on the experience of the project members, the choice has been made to focus, primarily but not exclusively, on the following selected code design problems:

- Improving the match between the modern FEC codes and their companion message-passing decoders at short to medium block length ;
- Finding better heuristics and smarter search procedures to be used in the design of turbo (selection of interleaver and puncturing mask), LDPC (calculation of degree distributions, construction of the graph), and polar codes (selection of the frozen set) ;
- Designing better coded modulation schemes through a joint optimization of the code and modulation ;

All are regarded as ideal playgrounds for experimenting with ML. Most of them are difficult in the sense that we lack a clear theoretical understanding so far. Thus any progress on either of these problems is expected to generate advances that could impact, in turn, the design of communication systems in the short term, by building intelligently on top of existing knowledge and practice.

The central goal of the AI4CODE project is not to replace expert knowledge by black-box algorithms, but ultimately to learn from the machine. All project members have recognized expertise in the design and decoding of error-correcting codes. Thus the driving methodology will be to capitalize upon this expertise and to favour a model-based approach by augmenting our legacy code design methods with learning algorithms wherever relevant. If the machine comes up with new code designs

that outperform the state-of-the-art, then we will inspect and try to interpret the trained solutions in order to infer why they work better, with the intent of obtaining new theoretical hindsight that could ultimately translate into improved design tools. Being able to generate a good code is not sufficient. Practical transmission systems need also some performance guarantees, that can be obtained only by analytical or semi-analytical prediction of error-correction performance. This reinforces the need for introducing structure in ML-based code design, not only to facilitate and scale the learning, but also to obtain more easily interpretable solutions.

2 Review of State-of-the-Art ML-assisted Code Design

The following is a review of the major learning approaches proposed to design better FEC codes. In keeping with the objectives set for the AI4CODE project, particular emphasis will be placed on the FEC code families currently in use in most communication and storage systems, *e.g.* turbo, LDPC, and polar codes. We note that ML has lead also to remarkable recent breakthroughs in the design of feedback codes, first with the Deepcode architecture [20], and more recently with the AttentionCode scheme [21]. But for as interesting and promising as they are, those schemes fall outside the scope of the AI4CODE project, which solely focuses on forward-error correction. Accordingly, learned feedback codes won't receive further attention and discussion here.

2.1 LDPC Codes

The design of short binary LDPC codes proves to be a real challenge in that the usual methods and code- or graph-related metrics used to construct capacity-approaching LDPC codes fail to accurately capture the error-rate performance of iterative decoding at short block length. The problem is that the convergence of iterative decoding is hard to analyze in this regime. This results in codes with performance that degrades with the code length. Surprisingly, AI-aided LDPC code design has received very little attention to date. The bulk of the work has focused on improving the decoding of existing LDPC codes, notably the 5G New Radio LDPC codes (see the state-of-the-art review in Deliverable D3.1 [6]). But it is likely that code design needs to be revisited for that purpose also. To the best of our knowledge, the most relevant and sole contribution to decoder-aware learned LDPC codes is [22] which uses genetic algorithms to design the full, sparse parity-check matrix at once so as to minimize the BLER under a prescribed decoding algorithm and number of iterations. The constructed LDPC code exhibit 0.3 to 0.8 dB additional coding gains over 5G LDPC codes over various channels. More importantly, the outcomes are analyzed to obtain design hints relevant to this regime. But genetic algorithms have more to do with standard optimization than with a real learning approach. The closest related contribution in the literature is [23], which provides an original, comprehensive joint code-decoder design framework to construct linear block codes, not necessarily sparse, having good performance under a specific form of neural belief propagation decoder.

A totally different approach is investigated in [24]. Here, a deep reinforcement learning algorithm is proposed for LDPC code construction, which attempts to transpose the learning techniques used in AlphaGo Zero. The design does not necessarily target short codes, and mainly aims at providing an informed substitute to the classical progressive edge growth (PEG) method. The DNN is trained by the reinforcement learning algorithm, and then used to generate the training data by repeated construction of codes through Monte-Carlo tree search (MCTS) techniques. The learned codes have performance similar to PEG codes, no better, and are obtained at a much more formidable cost in time and memory. Reinforcement learning techniques were also applied to decoder-aware code design in [15], and in a smarter way. However the focus was placed on constructing generic linear block codes with good performance under optimum maximum-likelihood decoding.

The use of learning techniques to optimize LDPC degree distributions was considered in [25]. The LDPC code design problem is cast as a supervised learning problem by mapping the recursive update equations of density evolution to a recurrent neural network (RNN) architecture, whose trained weights are interpreted as the coefficients of the desired degree distributions. The advantage of this approach over the usual differential evolution method is unclear.

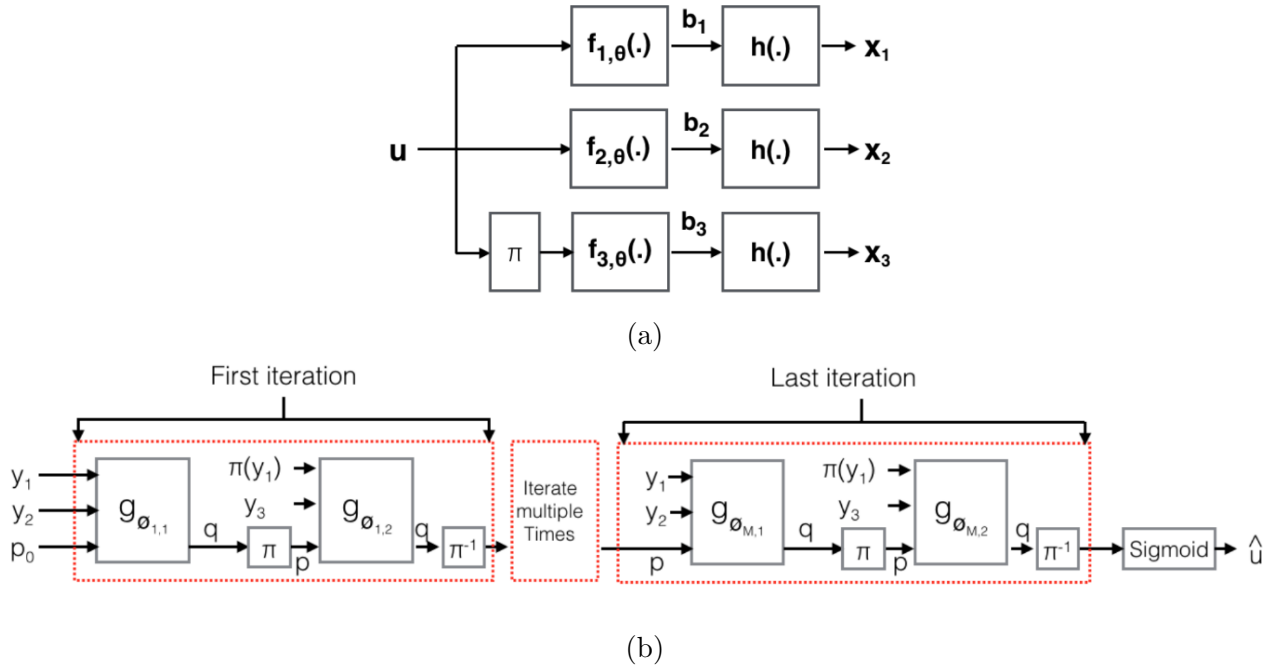

Figure 2.1: : (a) Rate-1/3 TurboAE encoder, (b) Corresponding TurboAE iterative decoder. Both figures are taken from [26] .

2.2 Turbo Codes

Among the papers published in recent years in the field of AI for channel coding, a particular topic emerges: the autoencoder approach [1]. The end-to-end learning concept of the AE approach aims at jointly learning the complete data-link between transmitter and receiver including all required signal processing.

2.2.1 The turbo autoencoder (TurboAE) baseline

The channel encoder and the decoder together can be viewed as an over-complete AE, where the noisy channel in the middle corrupts the hidden representation. However, it was observed that neither convolutional neural networks (CNN) nor recurrent neural networks based AEs can directly approach state-of-the-art performance. Furthermore, they do not scale well and are typically limited by an exponential training complexity.

To overcome these issues, the authors in [26] have been inspired by a successful conventional code construction technique, the turbo coding structure. They have proposed a turbo code inspired neural network structure called TurboAE [26], which is also described in [27]. As described in Fig. 2.1(a), TurboAE is a neural network-based over-complete autoencoder parameterized as CNNs along with pseudo-random interleavers (permutation) and deinterleavers (inverse permutation) inspired by turbo codes.

The training phase is crucial and has the following specificities:

- Very large batch size (>500) to average the channel noise effect,
- The encoder and decoder are trained separately to avoid getting stuck in local optimum,
- The encoder and decoder are trained with several noise levels.

TurboAE is able to outperform classical algorithms in some scenarios (fading channels) in a wide signal-to-Noise Ratio region for short message lengths (up to a few hundred bits) but its coding gain for long block length is smaller than turbo code due to trainability and computation issues.

The inference complexities of TurboAE encoder and decoder were compared with the computational complexities of the canonical Turbo encoder and decoder: a ratio of 17 in favour of the conventional implementation is observed for the encoder and more than 700 for the decoder.

2.2.2 Further improvements to the TurboAE approach

Improvements were later added to this first version of TurboAE. In [28], a novel interleaver was proposed for the TurboAE to address an issue with non-uniformity of BER across decoded bit positions. This paper also proposes enhancements to the training of TurboAE via a novel addition to the loss function. Due to these enhancements, the end-to-end performance of the TurboAE structure is improved with coding gains of several tenths of a dB for the transmission of 100-bit messages. In [29], a penalty function is introduced to make the interleaver trainable and provide an optimization methodology to learn the interleaver together with the rest of TurboAE. The introduction of the trainable interleaver leads to improved reliability on various channels, such as fading channels and bursty noise channels. With a personalized training, the improved TurboAE shows a better performance when compared to the LTE turbo code, but only in the low-to-moderate E_b/N_0 regime. The observed gain varies from channel to channel and can reach up to 1dB at bit error rate values in the range of 10^{-1} to 10^{-2}.

2.2.3 Extension of the TurboAE approach

The authors of [30] have extended the TurboAE concept of [26] to propose a new serially concatenated TurboAE neural network structure, as shown in Fig. 2.2. They also propose a pre-training technique that effectively lowers the training time by almost a magnitude. This serial structure shows significant gains compared to the parallel one, but still shows a gap towards the LTE Turbo code. More recently, the same authors have shown that classic trellis decoders such as the BCJR algorithm can be used to decode and also optimize or even learn from scratch CNN-based encoders [31], thereby providing further insight into this type of neural codes.

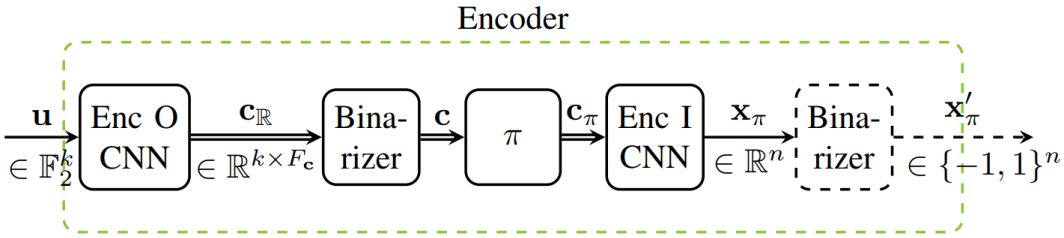

Figure 2.2: : Serial concatenated encoder structure with binarization, taken from [30].

In [32], the TurboAE approach is combined with the DeepCode approach of [20], dedicated to feedback channels. They combine best of both sides by introducing feedback turbo auto-encoder (FTAE), which has block length gain with feedback on block-delayed feedback channel. FTAE shows better performance than conventional turbo codes for bock length up to a few hundred bits. Longer block length performance has still to be improved.

At the same time, the TurboAE concept was combined with feed-forward neural networks for modulation, hence proposing a neural network based coded modulation called TurboAE-MOD [33]. Training is performed in two phases. In the first phase, the encoder and decoder are fixed and only the modulator and the demodulator are trained, via AE training. In the second phase, all four blocks, encoder, modulator, demodulator and decoder, are trained jointly using AE training. The correction performance of TurboAE-MOD is comparable to turbo codes with standard modulations, such as PSK and QAM, on AWGN channel for various spectral efficiencies on moderate block lengths, and it outperforms turbo coded modulations on non-AWGN channels.

2.3 Polar Codes

The polar coding process splits the input vector into two sets, the information set and the frozen set that carries known values, usually defined as zero. Given the encoding rate and the code length, the polar code construction problem aims to decide whether a position in the source vector is a frozen bit or an information bit, in order to provide the best error-correction performance, for a transmission channel and a decoding algorithm in particular. The first polar code construction techniques take advantage of subchannel polarization phenomenon to sort bit channels by reliability and then select the most reliable ones to send information through [34]. However, reliability-based polar code construction does not necessarily provide the best error-correction performance under decoding algorithms other than successive-cancellation (SC) decoding.

Deep learning methods have been recently used as tools to construct polar codes given the nature of the decoder. In [35] a sequential method of constructing polar codes for SCL decoding was introduced. The proposed method is formalized as a maze-game and is solved using reinforcement learning. The game-based polar code constructions were able to outperform the SC standard constructions of [36], for long codes, under both SC and SCL decoding. Other works propose genetic-algorithm-based polar-code constructions. In [15] a focus was made on the SC-based decoders, namely SCL, while [37] and its extension [38] evaluate the BP decoding as well. Apart from decoder types, the major difference between both works lies in inputs to each step of the genetic algorithm. The genetic population in [15] was randomly initialized whereas [37] adopts prior constructions based on the Bhattacharyya parameter introduced in [34] and RM-polar codes [39]. The error-performance results in [15] and [37] were compared to previous works. It turns out that the genetic constructions perform better than those generated in [40], [39] and [41] under SCL decoding. The genetic BP-tailored polar code in [37] shows an error-performance slightly close to the one constructed using the method in [36] under SCL decoding. A graph-based polar code construction method for BP decoding was introduced [42]. The method deduces its trainable neural network from the unrolled BP-decoding graph along iterations and takes into account the channel model. The network was trained over both AWGN and Rayleigh fading channels. The resulting code constructions were compared to 5G polar codes under BP decoding and were proven more effective in terms of error-performance over both channels. Authors in [43] proposed a different graph-based method tailored for the CA-SCL decoding using a bipartite Tanner-like graph. The neural network graph turns out to be generalized to design codes with characteristics other than those seen during training, such as different code lengths, code rates, and channel conditions. The constructions generated can either achieve better or similar error-performance with remarkably lower training complexity in comparison with polar codes in [39], [36], [38] and [35].

In [44], a neural network architecture was built in order to predict the error performance of polar codes under SCL decoding, according to their frozen bit positions. The trained network was further used to generate more frozen bit sequences using Projected Gradient Descent methodology. As a result, the newfound sequences, for long codes, outperform the training data in terms of frame-error-rate. In [45], [46] and [15], attention was placed on nested polar codes which meet communication system requirements regarding memory storage. The construction problem has been modeled as a Markov decision process in [45] and [15]. The reliability sequence was sequentially constructed using a neural network optimized A2C (Advantage Actor Critic) algorithm to maximize the performances overall the nested codes [15]. PPO (proximal policy optimization) algorithm was used in [45], given prior knowledge on optimal polar code constructions generated by the genetic algorithm of [15] for faster convergence. The obtained codes can achieve a comparable, even better, performance compared to the ones of [36] and [40]. In [46], an attention-mechanism-based neural network architecture was adopted. The proposed code constructions outperform 5G nested polar codes as well as the constructions resulted from paper [15] in terms of both error-performance and construction time complexity.

2.4 Coded Modulation

The literature on the topic of AI techniques for the design of coded modulations is far from abundance. Among recent papers, we can mention [47], where a reinforcement learning technique was used to design non-uniform constellations dedicated for non-binary (NB) codes. More specifically, a multi-agent deep Q-learning (DQN) algorithm was used to design a non-uniform 64-QAM constellation to be matched with a NB turbo code (NB-TC) based on one-memory NB convolutional component codes (NB-CC) defined over GF(64).

The application of a non-uniform distribution of the Euclidean distances between constellation symbols can bring its share of improvements for a given NB-CC defined over GF(q) when adapted to the distance spectrum of the code. However, a considerable number of possible constellation shapes should be enumerated in order to find the best non-uniform constellation for a given NB-CC. For large values of q and constellation sizes, this enumeration becomes computationally intractable.

The multiagent DQN algorithm proposed in [48] was used to jointly optimize the positions of all constellation symbols. It involves training multiple DQN agents in parallel in order to jointly optimize a single total reward called the team reward. In the study case involving codes over GF(64) and a 64-QAM constellation, 64 DQN learning agents were applied in parallel to maximize the minimum Euclidean distance of the NB-CC.

Monte Carlo simulations carried out in Gaussian channel showed that the DQN-assisted design of the 64-QAM constellation improves the floor of the corresponding turbo coded modulation by one decade without significantly penalizing its convergence threshold.

3 Summary of the Planned Contributions

3.1 Decoder-aware short LDPC code design

The heuristics used by most practical LDPC code design methods aim at optimizing code or bipartite graph parameters that only partially reflect the performance of iterative decoding at short length. AI4CODE intends to push one step further the decoder-aware design approach of [37] by investigating at a broader level how ML can help come up with short, structured LDPC codes that are better suited to iterative decoding at short to very-short block length. A first step could be to start from an initial bipartite graph either constructed at random or produced by one of the usual LDPC code design method, and use learning techniques, for example reinforcement learning, to progressively transform this initial graph into a new graph having better performance under a prescribed iterative decoding algorithm. Our primary goal is to learn how to design short LDPC codes which are good for BP decoding or for any of its improvement to be developed within the project. The best bipartite graphs produced by learning will be compared to the graphs obtained by conventional designs, with the aim of gaining theoretical hindsight about LDPC code design at short length. The underlying motivation is to characterize LDPC code ensembles, e.g. identify special code structures or degree distributions, that are good for iterative decoding in this regime. Learned joint optimization of the code design and neural BP decoder will also be investigated in a second step.

3.2 Design of Polar codes optimized for both decoding and implementation performance

The construction of polar codes consists in selecting the location of frozen bits. The original polar codes were devised assuming an SC decoder with an infinite code length N. However, the polarization phenomenon grows slowly with N, and the SC decoder, despite its low complexity, does not seem to be a practical solution, especially if N has to be very large ($N > 2^{15}$).

The alternative SCL decoder brings some performance improvements at some complexity cost. This algorithm can be seen as an approximate Maximum-Likelihood Decoder (MLD) in the sense that if the list size $L = 2^K$, all the possible codewords would be evaluated in terms of distance to the received vector. Unlike the SC decoder, the SCL decoder benefits from codes with good distance properties. Some minimum distance optimizations were proposed in order to improve the decoding performance. The PAC codes, and the dynamic frozen bits polar codes are two instances of these improved constructions. However, unlike SC decoding, even if these codes are optimized for distance, the construction methods do not actually include the decoder model in the optimization process. The challenge here is to find construction methods that would include the decoder model but also some implementation parameters in such a way that the polar code is optimized for both the decoding performance and the implementation efficiency. To reach this goal, we propose to start from the work in [44]. The proposed method is based on the prediction of the FER performance of a polar code assuming a given decoder configuration(SCL with $L = 8$). It is shown that a well defined neural network can predict the performance of a frozen bit set under SCL. The neural network can also provide a frozen bit set given a target FER. The generated codes actually reach better decoding performance than the learnt codes. The learning process is agnostic with the type of decoder or the list size. The NN can be used to predict the performance of any polar decoder as long as it is used to generate the learning data set.

As a first step, we want to change the learning set to a better one. Actually, in [44], the polar codes data set is built from density evolution methods. The codes in the data set do not have state-of-the-art performance. It would be interesting to apply the method to more recent constructions that have improved performance. We hope that the NN will then be able to generate some even better codes.

This would answer to the question : "Can NN improve the state-of-the-art polar codes constructions ?"

Another interesting aspect of the method is that it includes the decoder in the learning process. One can change the floating-point SCL decoder used for the FER performance to a more practical decoder with limited list size, quantized LLRs. This will bring the code construction closer to implementation perspectives. We could also take into account the decoding tree structure in order to generate codes that are more "prunable" and thus less complex to decode.

Finally, on a longer perspective, we would investigate other types of decoders such as BP or permutation based decoders. In the latter case, it would be interesting to generate efficient automorphisms of a given polar code or alternatively devise a polar code given a set of hardware-friendly automorphisms.

3.3 Joint interleaving and puncturing learning

The coding solutions in applications such as Internet of Things (IoT) are expected to use short frame sizes, low code rates while achieving excellent performance under low SNR. For traditional Turbo code construction methods, coding theoretic measures do exist to predict code performance and to assist code design (i.e. EXIT charts, distance spectrum). However, these do not yield conclusive results for smaller frame sizes on one side and are extremely complex to apply for large frame sizes on the other. Indeed for small frame sizes, there is no known code parameter that captures accurately the convergence behavior under iterative decoding. For large frame sizes, impulse-based methods, classically used for distance spectrum estimate, become extremely time and resource consuming to apply for obtaining accurate results. Both these facts advocate resorting to extensive Monte-Carlo simulations for code design.

For Turbo Codes, an interleaver construction constrained by puncturing patterns and applying protographs has been proposed by members of the AI4CODE team [49]. This layered construction answered implementation constraints for the first flexible Turbo decoder with a throughput of more than 100 Gb/s [50] developed in the context of the H2020 EPIC project. Obtained results confirmed the strengths of such a construction as well as the need to tailor the interleaver design to the considered decoding algorithm.

By using Machine/Deep Learning methods, AI4CODE intends to bridge the gap between a structured methodical construction of the code and the current trial and error approach based on a-posteriori evaluation of code properties. Using Reinforcement Learning (RL) and genetic algorithms, AI4CODE will build upon the experience and results gathered in EPIC, focusing on layered code constructions for small frame sizes of a few hundred bits. We will start by evaluating the suitability of Machine Learning techniques and genetic algorithms for AI-aided layered Turbo code interleaver design. This will be followed by identifying suitable known joint interleaver and puncturing parameters as input for the most promising AI-methods. As evidenced by preliminary results obtained in the EPIC project, maintaining the connection with the decoding algorithm is critical, since sub-optimal decoding has an increased impact on the convergence behavior of the designed codes. The efficiency of the design procedure will be assessed through the classification of the designed codes into poor and good performance categories. This classification will resort to the evaluation of performance through Monte Carlo simulations or through the evaluation of some code design metrics such as the minimum distance when possible.

3.4 Joint design of code and modulation

For IoT applications, Non-Binary Turbo Codes (NB-TC) over Galois Fields GF(q) have been shown to have considerable potential to outperform the binary case [51] thanks to their excellent performance for short frame sizes. Moreover, they can be mapped directly to the corresponding modulation symbols of higher order modulation schemes, hence profiting from an improved association between the code and the constellation. In fact, the cardinality of the used Galois field offers a new degree of freedom compared with binary codes. Inspired by the intra-symbol permutation introduced for double-binary

TCs, previous works from AI4CODE team found a transformation function that, when applied to the input data of one of the component encoders lowers the error floor of NB-TCs without entailing any penalty to their convergence threshold [51].

In other recent works, we proposed a new approach to optimize non-uniform constellations for NB codes, targeting improved performance of the underlying coded modulation [47]. In particular, deep Q-learning algorithms were applied to solve the intractable problem of enumerating all possible candidate high-order constellations. Results have shown that non-uniform constellations designed in this manner significantly improve the error correcting performance asymptotically. The two approaches of AI-aided non-uniform constellation design and symbol transformation for NB-TC shall be combined in a joint design during this work to improve the performance of the joint code/constellation design. Now tailored for the NB-TC, the resulting constellation is expected to improve performance of the coded modulation, especially at low error rates and high coding rates.

3.5 Design of LDPC codes for deep unfolded non linear coded modulation

We will consider code nonlinear modulation schemes involving LDPC codes for which iterative decoding is mandatory to operate close to the fundamental limits (typically orthogonal or quasi-orthogonal modulations). In this context, we intend to design a deep-unfolded coded architecture which consists in alternating some SISO detection layers with some SISO channel decoding layers. This can be seen as the design of a fully unrolled serially concatenated coded modulation scheme. If detection layer will be considered of only one type, we will address the design of the channel code layers to be used in this unrolled architecture. Quantized scheme could be also envisioned.

4 General Conclusion

The advent of deep learning and the continuous improvements in large-scale computing capabilities over the past decade have fuelled a resurgence of interest in applying machine learning techniques to a variety of communication problems. The availability of off-the-shelf learning software packages now makes it possible to parameterize communication systems and algorithms or even replace them by generic, black-box deep neural networks, and to train them to perform at least as well as the state-of-the-art. Application to channel code design and decoding is no exception to this general trend. Yet the application of ML to channel coding is still in its early stages, and its potential benefits for the field far from being sufficiently investigated and understood.

This deliverable aimed at specifying channel code design problems for which ML could be a game changer. After discussing the opportunities for machine learning in coding theory, a broad state-of-the-art review of ML-assisted FEC design has been provided, with an emphasis on recent advances in ML-based design of LDPC, Turbo, Polar, algebraic codes, and also coded modulations. Areas for potential improvements have been identified for each code family, and will be the subject of forthcoming studies in this WP. The main driver underlying those studies is to learn from the machine, and eventually come up with new theoretical hindsight to refine and improve existing code design methods and tools.

Bibliography

[1] 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, 2017.

[2] Z. Qin, H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep learning in physical layer communications," *IEEE Wireless Communications*, vol. 26, no. 2, pp. 93–99, 2019.

[3] C. She, C. Sun, Z. Gu, Y. Li, C. Yang, H. V. Poor, and B. Vucetic, "A tutorial on ultrareliable and low-latency communications in 6g: Integrating domain knowledge into deep learning," *Proceedings of the IEEE*, vol. 109, no. 3, pp. 204–246, 2021.

[4] S. Cammerer, F. A. Aoudia, S. Dörner, M. Stark, J. Hoydis, and S. Ten Brink, "Trainable communication systems: Concepts and prototype," *IEEE Transactions on Communications*, vol. 68, no. 9, pp. 5489–5503, 2020.

[5] J. Hoydis, S. Cammerer, F. A. Aoudia, A. Vem, N. Binder, G. Marcus, and A. Keller, "Sionna: An open-source library for next-generation physical layer research," *arXiv preprint arXiv:2203.11854*, 2022.

[6] AI$_4$CODE Project (ANR-21-CE25-0006), "Design space exploration for ml-augmented decoding," Deliverable D3.1, Dec. 2022.

[7] H. Kim, S. Oh, and P. Viswanath, "Physical layer communication via deep learning," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 5–18, 2020.

[8] Y.-H. Kim, A. Lapidoth, and T. Weissman, "The gaussian channel with noisy feedback," in *2007 IEEE International Symposium on Information Theory*. IEEE, 2007, pp. 1416–1420.

[9] 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.

[10] R. J. McEliece, "Are turbo-like codes effective on nonstandard channels?" *IEEE Information Theory Society Newsletter*, vol. 51, no. 4, pp. 1–8, 2001.

[11] G. Liva and Y. Polyanskiy, "On coding techniques for unsourced multiple-access," in *2021 55th Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2021, pp. 1507–1514.

[12] F. Kazemi, E. Karimi, E. Soljanin, and A. Sprintson, "A combinatorial view of the service rates of codes problem, its equivalence to fractional matching and its connection with batch codes," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 646–651.

[13] P.-N. Chen, H.-Y. Lin, and S. M. Moser, "Nonlinear codes outperform the best linear codes on the binary erasure channel," in *2015 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2015, pp. 1751–1755.

[14] A. V. Makkuva, X. Liu, M. V. Jamali, H. Mahdavifar, S. Oh, and P. Viswanath, "Ko codes: inventing nonlinear encoding and decoding for reliable wireless communication via deep-learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 7368–7378.

[15] L. Huang, H. Zhang, R. Li, Y. Ge, and J. Wang, "Ai coding: Learning to construct error correction codes," *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 26–39, 2019.

[16] Y. Jiang, H. Kim, H. Asnani, S. Kannan, S. Oh, and P. Viswanath, "Learn codes: Inventing low-latency codes via recurrent neural networks," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 207–216, 2020.

[17] N. A. Letizia and A. M. Tonello, "Capacity-driven autoencoders for communications," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 1366–1378, 2021.

[18] R. Wiesmayr, G. Marti, C. Dick, H. Song, and C. Studer, "Bit error and block error rate training for ml-assisted communication," *arXiv preprint arXiv:2210.14103*, 2022.

[19] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[20] H. Kim, Y. Jiang, S. Kannan, S. Oh, and P. Viswanath, "Deepcode: Feedback codes via deep learning," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 194–206, 2020.

[21] Y. Shao, E. Ozfatura, A. Perotti, B. Popovic, and D. Gunduz, "Attentioncode: Ultra-reliable feedback codes for short-packet communications," *arXiv preprint arXiv:2205.14955*, 2022.

[22] A. Elkelesh, M. Ebada, S. Cammerer, L. Schmalen, and S. Ten Brink, "Decoder-in-the-loop: Genetic optimization-based ldpc code design," *IEEE access*, vol. 7, pp. 141 161–141 170, 2019.

[23] G. Larue, L.-A. Dufrene, Q. Lampin, H. Ghauch, and G. R.-B. Othman, "Neural belief propagation auto-encoder for linear block code design," *IEEE Transactions on Communications*, vol. 70, no. 11, pp. 7250–7264, 2022.

[24] M. Zhang, Q. Huang, S. Wang, and Z. Wang, "Construction of ldpc codes based on deep reinforcement learning," in *2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 2018, pp. 1–4.

[25] E. Nisioti and N. Thomos, "Design of capacity-approaching low-density parity-check codes using recurrent neural networks," *arXiv preprint arXiv:2001.01249*, 2020.

[26] Y. Jiang, H. Kim, H. Asnani, S. Kannan, S. Oh, and P. Viswanath, "Turbo autoencoder: Deep learning based channel codes for point-to-point communication channels," in *2019 33rd Conference on Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada, Dec. 2019.

[27] H. Kim, S. Oh, and P. Viswanath, "Physical layer communication via deep learning," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 5–18, 2020.

[28] H. Yildiz, H. Hatami, H. Saber, Y.-S. Cheng, and J. H. Bae, "Interleaver design and pairwise codeword distance distribution enhancement for turbo autoencoder," in *2021 IEEE Global Communications Conference (GLOBECOM)*. Madrid, Spain: IEEE, Dec. 2021.

[29] K. Chahine, Y. Jiang, P. Nuti, H. Kim, and J. Cho, "Turbo autoencoder with a trainable interleaver," in *2022 IEEE International Conference on Communications (ICC)*. Seoul, Korea: IEEE, Aug. 2022, pp. 3886–3891.

[30] J. Clausius, S. Dörner, S. Cammerer, and S. ten Brink, "Serial vs. parallel turbo-autoencoders and accelerated training for learned channel codes," in *2021 11th International Symposium on Topics in Coding (ISTC)*, Montreal, Canada, Sep. 2021.

[31] J. Clausius, M. Geiselhart, and S. t. Brink, "Optimizing serially concatenated neural codes with classical decoders," *arXiv preprint arXiv:2212.10355*, 2022.

[32] Y. Jiang, H. Kim, H. Asnani, S. Oh, S. Kannan, and P. Viswanath, "Feedback turbo autoencoder," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Barcelona, Spain: IEEE, May 2020, pp. 8559–8563.

[33] Y. Jiang, H. Kim, H. Asnani, S. Kannan, S. Oh, and P. Viswanath, "Joint channel coding and modulation via deep learning," in *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. Atlanta, GA, USA: IEEE, May 2020, pp. 1–5.

[34] 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.

[35] Y. Liao, S. A. Hashemi, J. M. Cioffi, and A. Goldsmith, "Construction of polar codes with reinforcement learning," *IEEE Transactions on Communications*, vol. 70, no. 1, pp. 185–198, 2021.

[36] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6562–6582, 2013.

[37] A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Genetic algorithm-based polar code construction for the awgn channel," in *SCC 2019; 12th International ITG Conference on Systems, Communications and Coding*. VDE, 2019, pp. 1–6.

[38] A. Elkelesh, M. Ebada, S. Cammerer, and S. Ten Brink, "Decoder-tailored polar code design using the genetic algorithm," *IEEE Transactions on Communications*, vol. 67, no. 7, pp. 4521–4534, 2019.

[39] B. Li, H. Shen, and D. Tse, "A rm-polar codes," *arXiv preprint arXiv:1407.5483*, 2014.

[40] P. Trifonov, "Efficient design and decoding of polar codes," *IEEE Trans. on Comm.*, vol. 60, no. 11, pp. 3221–3227, 2012.

[41] G. He, J.-C. Belfiore, I. Land, G. Yang, X. Liu, Y. Chen, R. Li, J. Wang, Y. Ge, R. Zhang *et al.*, "Beta-expansion: A theoretical framework for fast and recursive construction of polar codes," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–6.

[42] M. Ebada, S. Cammerer, A. Elkelesh, and S. Ten Brink, "Deep learning-based polar code design," *57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2019.

[43] Y. Liao, S. A. Hashemi, H. Yang, and J. M. Cioffi, "Scalable polar code construction for successive cancellation list decoding: A graph neural network-based approach," *arXiv preprint arXiv:2207.01105*, 2022.

[44] 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.

[45] L. Huang, H. Zhang, R. Li, Y. Ge, and J. Wang, "Reinforcement learning for nested polar code construction," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[46] Y. Li, Z. Chen, G. Liu, Y.-C. Wu, and K.-K. Wong, "Learning to construct nested polar codes: An attention-based set-to-element model," *IEEE Communications Letters*, vol. 25, no. 12, pp. 3898–3902, 2021.

[47] R. Klaimi, S. Weithoffer, and C. A. Nour, "Improved non-uniform constellations for non-binary codes through deep reinforcement learning," in *2022 IEEE 23rd International Workshop on Signal Processing Advances in Wireless Communication (SPAWC)*, 2022, pp. 1–5.

[48] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. F. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning," *CoRR*, vol. abs/1706.05296, 2017. [Online]. Available: http://arxiv.org/abs/1706.05296

[49] R. Garzón-Bohórquez, C. Abdel Nour, and C. Douillard, "Protograph-based interleavers for punctured turbo codes," *IEEE Transactions on Communications*, vol. 66, no. 5, pp. 1833–1844, 2018.

[50] S. Weithoffer, O. Griebel, R. Klaimi, C. A. Nour, and N. Wehn, "Advanced hardware architectures for turbo code decoding beyond 100 gb/s," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, 2020, pp. 1–6.

[51] R. Klaimi, "Study of non-binary turbo codes for future communication and broadcasting systems," Ph.D. dissertation, 2019, phD Thesis at Mathematical and Electrical department of IMT Atlantique. [Online]. Available: http://www.theses.fr/2019IMTA0141/document

