KO codes: inventing nonlinear encoding and decoding for reliable wireless communication via deep-learning

Ashok Vardhan Makkuva [*1] **Xiyang Liu** [*2] **Mohammad Vahid Jamali** [3] **Hessam Mahdavifar** [3] **Sewoong Oh** [2] **Pramod Viswanath** [1]

Abstract

Landmark codes underpin reliable physical layer communication, e.g., Reed-Muller, BCH, Convolution, Turbo, LDPC and Polar codes: each is a linear code and represents a mathematical breakthrough. The impact on humanity is huge: each of these codes has been used in global wireless communication standards (satellite, WiFi, cellular). Reliability of communication over the classical additive white Gaussian noise (AWGN) channel enables benchmarking and ranking of the different codes. In this paper, we construct KO codes, a computationaly efficient family of deep-learning driven (encoder, decoder) pairs that outperform the state-of-the-art reliability performance on the standardized AWGN channel. KO codes beat state-of-the-art Reed-Muller and Polar codes, under the low-complexity successive cancellation decoding, in the challenging short-to-medium block length regime on the AWGN channel. We show that the gains of KO codes are primarily due to the nonlinear mapping of information bits directly to transmit symbols (bypassing modulation) and yet possess an efficient, high performance decoder. The key technical innovation that renders this possible is design of a novel family of neural architectures inspired by the computation tree of the **K**ronecker **O**peration (KO) central to Reed-Muller and Polar codes. These architectures pave way for the discovery of a much richer class of hitherto unexplored nonlinear algebraic structures. The code is available at https://github.com/deepcomm/KOcodes.

[*]Equal contribution [1]Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign [2]Paul G. Allen School of Computer Science & Engineering, University of Washington [3]Department of Electrical Engineerign and Computer Science, University of Michigan. Correspondence to: Ashok, Xiyang <makkuva2@illinois.edu, xiyangl@cs.washington.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

1. Introduction

Physical layer communication underpins the information age (WiFi, cellular, cable and satellite modems). Codes, composed of encoder and decoder pairs, are the basic mathematical objects enabling reliable communication: encoder maps original data bits into a longer sequence, and decoders map the received sequence to the original bits. Reliability is precisely measured: bit error rate (BER) measures the fraction of input bits that were incorrectly decoded; block error rate (BLER) measures the fraction of times at least one of the original data bits was incorrectly decoded.

Landmark codes include Reed-Muller (RM), BCH, Turbo, LDPC and Polar codes (Richardson & Urbanke, 2008): each is a linear code and represents a mathematical breakthrough discovered over a span of six decades. The impact on humanity is huge: each of these codes has been used in global communication standards over the past six decades. These codes essentially operate at the information-theoretic limits of reliability over the additive white Gaussian noise (AWGN) channel, when the number of information bits is large, the so-called “large block length" regime. In the small and medium block length regimes, the state-of-the-art codes are *algebraic*: encoders and decoders are invented based on specific linear algebraic constructions over the binary and higher order fields and rings. Especially prominent binary algebraic codes are RM codes and closely related polar codes, whose encoders are recursively defined as Kronecker products of a simple linear operator and constitute the state of the art in small-to-medium block length regimes.

Inventing new codes is a major intellectual activity both in academia and the wireless industry; this is driven by emerging practical applications, e.g., low block length regime in Internet of Things (Ma et al., 2019). The core challenge is that the space of codes is very vast and the sizes astronomical; for instance a rate $1/2$ code over even 100 information bits involves designing 2^{100} codewords in a 200 dimensional space. Computationally efficient encoding and decoding procedures are a must, apart from high reliability. Thus, although a random code is information theoretically optimal, neither encoding nor decoding is computationally efficient. The mathematical landscape of computationally efficient codes has been plumbed over the decades by some of the finest mathematical minds, resulting in two distinct families of codes: *algebraic codes* (RM, BCH – focused on properties of polynomials) and *graph codes* (Turbo, LDPC – based on sparse graphs and statistical physics). The former is deterministic and involves discrete mathematics, while the latter harnesses randomness, graphs, and statistical physics to behave like a pseudorandom code. A major open question is the invention of new codes, and especially fascinating would be a family of codes outside of these two classes.

Our major result is the invention of a new family of codes, called KO codes, that have features of both code families: they are nonlinear generalizations of the Kronecker operation underlying the algebraic codes (e.g., Reed-Muller) parameterized by neural networks; the parameters are learnt in an end-to-end training paradigm in a data driven manner. Deep learning (DL) has transformed several domains of human endeavor that have traditionally relied heavily on mathematical ingenuity, e.g., game playing (AlphaZero (Silver et al., 2018)), biology (AlphaFold (Senior et al., 2019)), and physics (new laws (Udrescu & Tegmark, 2020)). Our results can be viewed as an added domain to the successes of DL in inventing mathematical structures.

A linear encoder is defined by a *generator matrix*, which maps information bits to a codeword. The RM and the Polar families construct their generator matrices by recursively applying the Kronecker product operation to a simple two-by-two matrix and then selecting rows from the resulting matrix. The careful choice in selecting these rows is driven by the desired algebraic structure of the code, which is central to achieving the large *minimum* pairwise distance between two codewords, a hallmark of the algebraic family. This encoder can be alternatively represented by a computation graph. The recursive Kronecker product corresponds to a complete binary tree, and row-selection corresponds to freezing a set of leaves in the tree, which we refer to as a “Plotkin tree", inspired by the pioneering construction in (Plotkin, 1960).

The Plotkin tree skeleton allows us to tailor a new neural network architecture: we expand the algebraic family of codes by replacing the (linear) Plotkin construction with a non-linear operation parametrized by neural networks. The parameters are discovered by training the encoder with a matching decoder, that has the matching Plotkin tree as a skeleton, to minimize the error rate over (the unlimited) samples generated on AWGN channels.

Algebraic and the original RM codes promise a large worst-case pairwise distance (Alon et al., 2005). This ensures that RM codes achieve capacity in the large block length limit (Kudekar et al., 2017). However, for short block lengths, they are too conservative as we are interested in the average-case reliability. This is the gap KO codes exploit: we seek a better average-case reliability and not the minimum pairwise distance.

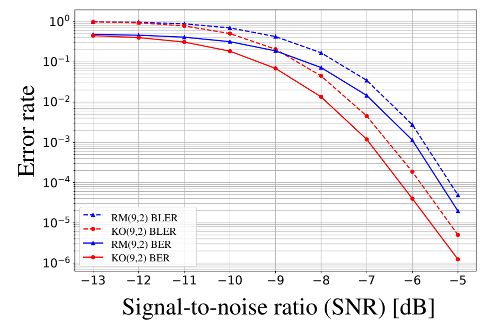


Figure 1. KO$(9, 2)$, discovered by training a neural network with a carefully chosen architecture in §3, significantly improves upon state-of-the-art RM$(9, 2)$ both in BER and BLER. (For both codes, the code block length is $2^9 = 512$ and the number of transmitted message bits is $\binom{9}{0} + \binom{9}{1} + \binom{9}{2} = 55$. Also, both codes are decoded using successive cancellation decoding with similar decoding complexity)

Figure 1 illustrates the gain for the example of RM$(9, 2)$ code. Using the Plotkin tree of RM$(9, 2)$ code as a skeleton, we design the KO$(9, 2)$ code architecture and train on samples simulated over an AWGN channel. We discover a novel non-linear code and a corresponding efficient decoder that improves significantly over the RM$(9, 2)$ code baseline, assuming both codes are decoded using successive cancellation decoding with similar decoding complexity. Analyzing the pairwise distances between two codewords reveals a surprising fact. The histogram for KO code nearly matches that of a random Gaussian codebook. The skeleton of the architecture from an algebraic family of codes, the training process with a variation of the stochastic gradient descent, and the simulated AWGN channel have worked together to discover a novel family of codes that harness the benefits of both algebraic and pseudorandom constructions.

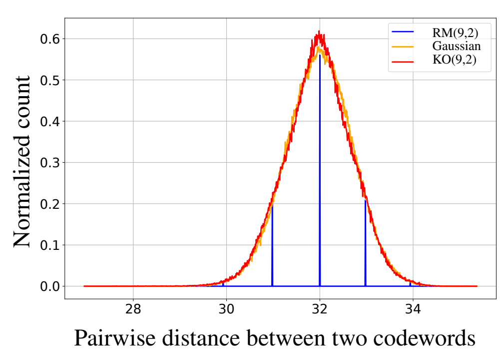


Figure 2. Histogram of pairwise distances between codewords of the KO$(9, 2)$ code shows a strong resemblance to that of the Gaussian codebook, unlike the classical Reed-Muller code RM$(9, 2)$.

In summary, we make the following contributions: We introduce novel neural network architectures for the (encoder, decoder) pair that generalizes the Kronecker operation central to RM/Polar codes. We propose training methods that discover novel non-linear codes when trained over AWGN and provide empirical results showing that this family of non-linear codes improves significantly upon the baseline code it was built on (both RM and Polar codes) whilst having the same encoding and decoding complexity. Interpreting the pairwise distances of the discovered codewords reveals that a KO code mimics the distribution of codewords from the random Gaussian codebook, which is known to be reliable but computationally challenging to decode. The decoding complexities of KO codes are $O(n \log n)$ where n is the block length, matching that of efficient decoders for RM and Polar codes.

We highlight that the design principle of KO codes serves as a general recipe to discover new family of non-linear codes improving upon their linear counterparts. In particular, the construction is not restricted to a specific decoding algorithm, such as successive cancellation (SC). In this paper, we focus on the SC decoding algorithm since it is one of the most efficient decoders for the RM and Polar family. At this decoding complexity, i.e. $O(n \log n)$, our results demonstrate that we achieve significant gain over these codes. Our preliminary results show that KO codes achieve similar gains over the RM codes, when both are decoded with list-decoding. We refer to §B for more details. Designing KO-inspired codes to improve upon the RPA decoder for RM codes (with complexity $O(n^r \log n)$ (Ye & Abbe, 2020)), and the list-decoded Polar codes (with complexity $O(Ln \log n)$ (Tal & Vardy, 2015)) where L is the list size, are promising active research directions, and outside the scope of this paper.

2. Problem formulation and background

We formally define the channel coding problem and provide background on Reed-Muller codes, the inspiration for our approach. Our notation is the following. We denote Euclidean vectors by bold face letters like $\boldsymbol{m}, \boldsymbol{L}$, etc. For $\boldsymbol{L} \in \mathbb{R}^n$, $\boldsymbol{L}_{k:m} \triangleq (L_k, \ldots, L_m)$. If $\boldsymbol{v} \in \{0,1\}^n$, we define the operator $\oplus_{\boldsymbol{v}}$ as $\boldsymbol{x} \oplus_{\boldsymbol{v}} \boldsymbol{y} \triangleq \boldsymbol{x} + (-1)^{\boldsymbol{v}} \boldsymbol{y}$.

2.1. Channel coding

Let $\boldsymbol{m} = (m_1, \ldots, m_k) \in \{0,1\}^k$ denote a block of *information/message bits* that we want to transmit. An encoder $g_\theta(\cdot)$ is a function parametrized by θ that maps these information bits into a binary vector $\boldsymbol{x}$ of length n, i.e. $\boldsymbol{x} = g_\theta(\boldsymbol{m}) \in \{0,1\}^n$. The *rate* $\rho = k/n$ of such a code measures how many bits of information we are sending per channel use. These codewords are transformed into real (or complex) valued signals, called modulation, before being transmitted over a channel. For example, Binary Phase Shift Keying (BPSK) modulation maps each $x_i \in \{0,1\}$ to $1 - 2x_i \in \{\pm 1\}$ up to a universal scaling constant for all $i \in [n]$. Here, we do not strictly separate encoding from modulation and refer to both binary encoded symbols and real-valued transmitted symbols as *codewords*. The codewords also satisfy either a hard or soft power constraint. Here we consider the hard power constraint, i.e., $\|x\|^2 = n$.

Upon transmission of this codeword $\boldsymbol{x}$ across a noisy channel $P_{Y|X}(\cdot|\cdot)$, we receive its corrupted version $\boldsymbol{y} \in \mathbb{R}^n$. The decoder $f_\phi(\cdot)$ is a function parametrized by ϕ that subsequently processes the received vector $\boldsymbol{y}$ to estimate the information bits $\hat{\boldsymbol{m}} = f_\phi(\boldsymbol{y})$. The closer $\hat{\boldsymbol{m}}$ is to $\boldsymbol{m}$, the more reliable the transmission. An error metric, such as Bit-Error-Rate (BER) or Block-Error-Rate (BLER), gauges the performance of the encoder-decoder pair (g_θ, f_ϕ). Note that BER is defined as $\mathrm{BER} \triangleq (1/k) \sum_i \mathbb{P}[\hat{m}_i \neq m_i]$, whereas $\mathrm{BLER} \triangleq \mathbb{P}[\hat{\boldsymbol{m}} \neq \boldsymbol{m}]$.

The design of good codes given a channel and a fixed set of code parameters (k, n) can be formulated as:

$$(\theta, \phi) \quad \in \quad \arg\min_{\theta, \phi} \ \mathrm{BER}(g_\theta, f_\phi) \ , \tag{1}$$

which is a joint classification problem for k binary classes, and we train on the surrogate loss of cross entropy to make the objective differentiable. While classical optimal codes such as Turbo, LDPC, and Polar codes all have *linear* encoders, appropriately parametrizing both the encoder $g_\theta(\cdot)$ and the decoder $f_\phi(\cdot)$ by neural networks (NN) allows for a much broader class of codes, especially non-linear codes. However, in the absence of any structure, NNs fail to learn non-trivial codes and end up performing worse than simply repeating each message bit n/k times (Kim et al., 2018; Jiang et al., 2019b).

A fundamental question in machine learning for channel coding is thus: how do we design architectures for our neural encoders and decoders that give the appropriate inductive bias? To gain intuition towards addressing this, we focus on Reed-Muller (RM) codes. In §3, we present a novel family of non-linear codes, *KO codes*, that strictly generalize and improve upon RM codes by capitalizing on their inherent recursive structure. Our approach seamlessly generalizes to Polar codes, explained in §5.

2.2. Reed-Muller (RM) codes

We use a small example of $\mathrm{RM}(3,1)$ and refer to Appendix G for the larger example in our main results.

Encoding. RM codes are a family of codes parametrized by a variable size $m \in \mathbb{Z}_+$ and an order $r \in \mathbb{Z}_+$ with $r \leq m$, denoted as $\mathrm{RM}(m,r)$. It is defined by an *encoder*, which maps binary information bits $\boldsymbol{m} \in \{0,1\}^k$ to codewords $\boldsymbol{x} \in \{0,1\}^n$. $\mathrm{RM}(m,r)$ code sends $k = \sum_{i=0}^{r} \binom{m}{i}$

information bits with $n = 2^m$ transmissions. The *code distance* measures the minimum distance between all (pairs of) codewords. Table 1 summarizes these parameters.

Code length	Code dimension	Rate	Distance
$n = 2^m$	$k = \sum_{i=0}^{r} \binom{m}{i}$	$\rho = k/n$	$d = 2^{m-r}$

Table 1. Parameters of a $\mathrm{RM}(m,r)$ code

One way to define $\mathrm{RM}(m,r)$ code is via the recursive application of a *Plotkin construction*. The basic building block is a mapping $\mathrm{Plotkin} : \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}^{2\ell}$, where

$$\mathrm{Plotkin}(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u}, \boldsymbol{u} \oplus \boldsymbol{v}) \,, \tag{2}$$

with $\oplus$ representing a coordinate-wise XOR and $(\cdot,\cdot)$ denoting concatenation of two vectors (Plotkin, 1960).

In view of the Plotkin construction, RM codes are recursively defined as a set of codewords of the form:

$$\mathrm{RM}(m,r) = \{(\boldsymbol{u}, \boldsymbol{u} \oplus \boldsymbol{v}) : \boldsymbol{u} \in \mathrm{RM}(m-1,r), \boldsymbol{v} \in \mathrm{RM}(m-1,r-1)\}, \tag{3}$$

where $\mathrm{RM}(m,0)$ is a repetition code that repeats a single information bit 2^m times, i.e., $\boldsymbol{x} = (m_1, m_1, \ldots, m_1)$. When $r = m$, the full-rate $\mathrm{RM}(m,m)$ code is also recursively defined as a Plotkin construction of two $\mathrm{RM}(m-1,m-1)$ codes. Unrolling the recursion in Eq. (3), a $\mathrm{RM}(m,r)$ encoder can be represented by a corresponding (rooted and binary) computation tree, which we refer to as its *Plotkin tree*. In this tree, each branch represents a Plotkin mapping of two codes of appropriate lengths, recursively applied from the leaves to the root.

Figure 3a illustrates such a Plotkin tree decomposition of $\mathrm{RM}(3,1)$ encoder. Encoding starts from the bottom right leaves. The leaf $\mathrm{RM}(1,0)$ maps m_3 to (m_3, m_3) (repetition), and another leaf $\mathrm{RM}(1,1)$ maps (m_1, m_2) to $(m_1, m_1 \oplus m_2)$ (Plotkin mapping of two $\mathrm{RM}(0,0)$ codes). Each branch in this tree performs the Plotkin construction of Eq. (2). The next operation is the parent of these two leaves, which performs $\mathrm{Plotkin}(\mathrm{RM}(1,1), \mathrm{RM}(1,0)) = \mathrm{Plotkin}((m_1, m_1 \oplus m_2), (m_3, m_3))$ which outputs the vector $(m_1, m_1 \oplus m_2, m_1 \oplus m_3, m_1 \oplus m_2 \oplus m_3)$, which is known as $\mathrm{RM}(2,1)$ code. This coordinate-wise Plotkin construction is applied recursively one more time to combine $\mathrm{RM}(2,0)$ and $\mathrm{RM}(2,1)$ at the root of the tree. The resulting codewords are $\mathrm{RM}(3,1) = \mathrm{Plotkin}(\mathrm{RM}(2,1), \mathrm{RM}(2,0)) = \mathrm{Plotkin}((m_1, m_1 \oplus m_2, m_1 \oplus m_3, m_1 \oplus m_2 \oplus m_3), (m_4, m_4, m_4, m_4))$.

This recursive structure of RM codes (i) inherits the good minimum distance property of the Plotkin construction and (ii) enables efficient decoding.

Decoding. Since (Reed, 1954), there have been several decoders for RM codes; (Abbe et al., 2020) is a detailed survey. We focus on the most efficient one, called *Dumer's recursive decoding* (Dumer, 2004; 2006; Dumer & Shabunov, 2006b) that fully capitalizes on the recursive Plotkin construction in Eq. (3). The basic principle is: to decode an RM codeword $\boldsymbol{x} = (\boldsymbol{u}, \boldsymbol{u} \oplus \boldsymbol{v}) \in \mathrm{RM}(m,r)$, we first recursively decode the left sub-codeword $\boldsymbol{v} \in \mathrm{RM}(m-1,r-1)$ and then the right sub-codeword $\boldsymbol{u} \in \mathrm{RM}(m-1,r)$, and we use them together to stitch back the original codeword. This recursion is continued until we reach the leaf nodes, where we perform maximum a posteriori (MAP) decoding. Dumer's recursive decoding is also referred to as *successive cancellation* decoding in the context of polar codes (Arikan, 2009).

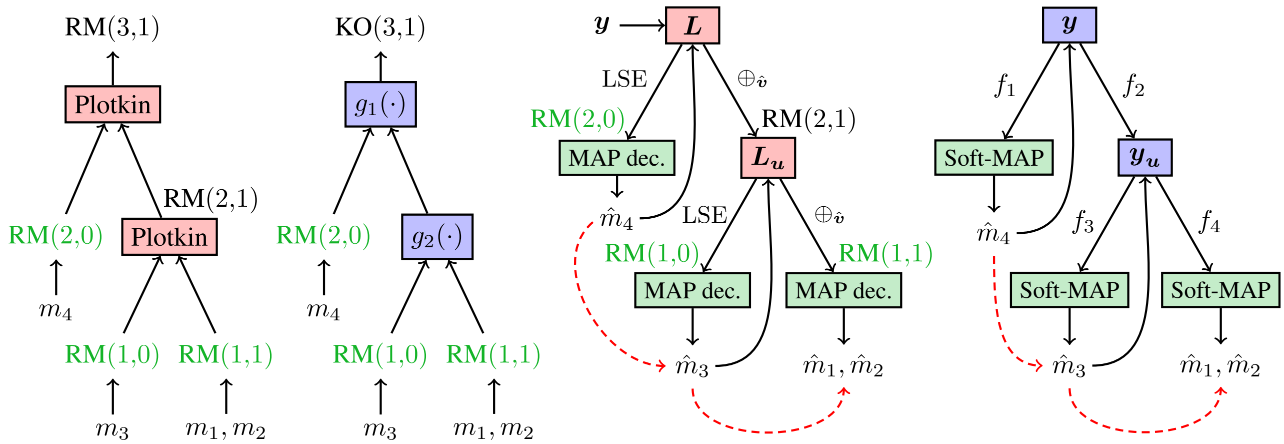
Figure 3c illustrates this decoding procedure for $\mathrm{RM}(3,1)$. Dumer's decoding starts at the root and uses the soft-information of codewords to decode the message bits. Suppose that the message bits $\boldsymbol{m} = (m_1, \ldots, m_4)$ are encoded into an $\mathrm{RM}(3,1)$ codeword $\boldsymbol{x} \in \{0,1\}^8$ using the Plotkin encoder in Figure 3a. Let $\boldsymbol{y} \in \mathbb{R}^8$ be the corresponding noisy codeword received at the decoder. To decode the bits $\boldsymbol{m}$, we first obtain the soft-information of the codeword $\boldsymbol{x}$, i.e., we compute its Log-Likehood-Ratio (LLR) $\boldsymbol{L} \in \mathbb{R}^8$:

$$L_i = \log \frac{\mathbb{P}[y_i | x_i = 0]}{\mathbb{P}[y_i | x_i = 1]}, \quad i = 1, \ldots, 8.$$

We next use $\boldsymbol{L}$ to compute soft-information for its left and right children: the $\mathrm{RM}(2,0)$ codeword $\boldsymbol{v}$ and the $\mathrm{RM}(2,1)$ codeword $\boldsymbol{u}$. We start with the left child $\boldsymbol{v}$.

Since the codeword $\boldsymbol{x} = (\boldsymbol{u}, \boldsymbol{u} \oplus \boldsymbol{v})$, we can also represent its left child as $\boldsymbol{v} = \boldsymbol{u} \oplus (\boldsymbol{u} \oplus \boldsymbol{v}) = \boldsymbol{x}_{1:4} \oplus \boldsymbol{x}_{5:8}$. Hence its LLR vector $\boldsymbol{L_v} \in \mathbb{R}^4$ can be readily obtained from that of $\boldsymbol{x}$. In particular it is given by the log-sum-exponential transformation: $\boldsymbol{L_v} = \mathrm{LSE}(\boldsymbol{L}_{1:4}, \boldsymbol{L}_{5:8})$, where $\mathrm{LSE}(a,b) \triangleq \log((1+e^{a+b})/(e^a+e^b))$ for $a, b \in \mathbb{R}$. Since this feature $\boldsymbol{L_v}$ corresponds to a repetition code, $\boldsymbol{v} = (m_4, m_4, m_4, m_4)$, majority decoding (same as the MAP) on the sign of $\boldsymbol{L_v}$ yields the decoded message bit as $\hat{m}_4$. Finally, the left codeword is decoded as $\hat{\boldsymbol{v}} = (\hat{m}_4, \hat{m}_4, \hat{m}_4, \hat{m}_4)$.

Having decoded the left $\mathrm{RM}(2,0)$ codeword $\hat{\boldsymbol{v}}$, our goal is to now obtain soft-information $\boldsymbol{L_u} \in \mathbb{R}^4$ for the right $\mathrm{RM}(2,1)$ codeword $\boldsymbol{u}$. Fixing $\boldsymbol{v} = \hat{\boldsymbol{v}}$, notice that the codeword $\boldsymbol{x} = (\boldsymbol{u}, \boldsymbol{u} \oplus \hat{\boldsymbol{v}})$ can be viewed as a 2-repetition of $\boldsymbol{u}$ depending on the parity of $\hat{\boldsymbol{v}}$. Thus the LLR $\boldsymbol{L_u}$ is given by LLR addition accounting for the parity of $\hat{\boldsymbol{v}}$: $\boldsymbol{L_u} = \boldsymbol{L}_{1:4} \oplus_{\hat{\boldsymbol{v}}} \boldsymbol{L}_{5:8} = \boldsymbol{L}_{1:4} + (-1)^{\hat{\boldsymbol{v}}} \boldsymbol{L}_{5:8}$. Since $\mathrm{RM}(2,1)$ is an internal node in the tree, we again recursively decode its left child $\mathrm{RM}(1,0)$ and its right child $\mathrm{RM}(1,1)$, which are both leaves. For $\mathrm{RM}(1,0)$, decoding is similar to that of $\mathrm{RM}(2,0)$ above, and we obtain its information bit $\hat{m}_3$ by first applying the log-sum-exponential function on the feature $\boldsymbol{L_u}$ and then majority decoding. Likewise, we obtain the LLR feature $\boldsymbol{L_{uu}} \in \mathbb{R}^2$ for the right $\mathrm{RM}(1,1)$ child using parity-adjusted LLR addition on $\boldsymbol{L_u}$. Finally,



(a) Plotkin encoder (b) KO encoder (c) Dumer's decoder (d) KO decoder

Figure 3. Plotkin trees for RM$(3, 1)$ and KO$(3, 1)$ codes; Leaves are shown in green. Red arrows indicate the bit decoding order.

we decode its corresponding bits $(\hat{m}_1, \hat{m}_2)$ using efficient MAP-decoding of first order RM codes (Abbe et al., 2020). Thus we obtain the full block of decoded message bits as $\hat{\boldsymbol{m}} = (\hat{m}_1, \hat{m}_2, \hat{m}_3, \hat{m}_4)$.

An important observation from Dumer's algorithm is that the sequence of bit decoding in the tree is: RM$(2, 0) \to$ RM$(1, 0) \to$ RM$(1, 1)$. A similar decoding order holds for all RM$(m, 2)$ codes, where all the left leaves (order-1 codes) are decoded first from top to bottom, and the right-most leaf (full-rate RM$(2, 2)$) is decoded at the end.

3. KO codes: Novel Neural codes

We design KO codes using the Plotkin tree as the skeleton of a new neural network architecture, which strictly improve upon their classical counterparts.

KO encoder. Earlier we saw the design of RM codes via recursive Plotkin mapping. Inspired by this elegant construction, we present a new family of codes, called *KO codes*, denoted as KO(m, r, g_θ, f_ϕ). These codes are parametrized by a set of four parameters: a non-negative integer pair (m, r), a finite set of encoder neural networks g_θ, and a finite set of decoder neural networks f_ϕ. In particular, for any fixed pair (m, r), our KO encoder inherits the same code parameters (k, n, ρ) and the same Plotkin tree skeleton of the RM encoder. However, a critical distinguishing component of our KO(m, r) encoder is a set of encoding neural networks $g_\theta = \{g_i\}$ that strictly generalize the Plotkin mapping: to each internal node i of the Plotkin tree, we associate a neural network g_i that applies a coordinate-wise real valued non-linear mapping $(\boldsymbol{u}, \boldsymbol{v}) \mapsto g_i(\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{R}^{2\ell}$ as opposed to the classical binary valued Plotkin mapping $(\boldsymbol{u}, \boldsymbol{v}) \mapsto (\boldsymbol{u}, \boldsymbol{u} \oplus \boldsymbol{v}) \in \{0, 1\}^{2\ell}$. Figure 3b illustrates this for the KO$(3, 1)$ encoder.

The significance of our KO encoder g_θ is that by allowing for general nonlinearities g_i to be learnt at each node we enable for a much richer and broader class of nonlinear encoders and codes to be discovered on a whole, which contribute to non-trivial gains over standard RM codes. Further, we have the same encoding complexity as that of an RM encoder since each $g_i : \mathbb{R}^2 \to \mathbb{R}$ is applied coordinate-wise on its vector inputs. The parameters of these neural networks g_i are trained via stochastic gradient descent on the cross entropy loss. See §I for experimental detailas.

KO decoder. Training the encoder is possible only if we have a corresponding decoder. This necessitates the need for an efficient family of matching decoders. Inspired by the Dumer's decoder, we present a new family of *KO decoders* that fully capitalize on the recursive structure of KO encoders via the Plotkin tree.

Our KO decoder has three distinct features: (i) Neural decoder: The KO decoder architecture is parametrized by a set of decoding neural networks $f_\phi = \{(f_{2i-1}, f_{2i})\}$. Specifically, to each internal node i in the tree, we associate f_{2i-1} to its left branch whereas f_{2i} corresponds to the right branch. Figure 3d shows this for the KO$(3, 1)$ decoder. The pair of decoding neural networks (f_{2i-1}, f_{2i}) can be viewed as matching decoders for the corresponding encoding network g_i: While g_i encodes the left and right codewords arriving at this node, the outputs of f_{2i-1} and f_{2i} represent appropriate Euclidean feature vectors for decoding them. Further, f_{2i-1} and f_{2i} can also be viewed as a generalization of Dumer's decoding to nonlinear real codewords: f_{2i-1} generalizes the LSE function, while f_{2i} extends the operation $\oplus_{\hat{\boldsymbol{v}}}$. Note that both the functions f_{2i-1} and f_{2i} are also applied coordinate-wise and hence we inherit the same decoding complexity as Dumer's. (ii) Soft-MAP decoding: Since the classical MAP decoding to decode the bits at the leaves is not differentiable, we design a new differentiable counterpart, the *Soft-MAP decoder*. Soft-MAP decoder enables gradients to pass through it, which is crucial for training the neural (encoder, decoder) pair (g_θ, f_ϕ) in an end-to-end manner. (iii) Channel agnostic: Our decoder directly operates on the received noisy codeword $\boldsymbol{y} \in \mathbb{R}^n$ while Dumer's decoder uses its LLR transformation $\boldsymbol{L} \in \mathbb{R}^n$. Thus, our decoder can learn the appropriate channel statistics for decoding directly from $\boldsymbol{y}$ alone; in contrast, Dumer's algorithm requires precise channel characterization, which is not usually known.

4. Main results

We train the KO encoder g_θ and KO decoder f_ϕ from §3 using an approximation of the BER loss in (1). The details are provided in §I. In this section we focus on the second-order KO$(8, 2)$ and KO$(9, 2)$ codes.

4.1. KO codes improve over RM codes

In Figure 1, the trained KO$(9, 2)$ improves over the competing RM$(9, 2)$ both in BER and BLER. The superiority in BLER is unexpected as our training loss is a surrogate for the BER. Though one would prefer to train on BLER as it is more relevant in practice, it is challenging to design a surrogate loss for BLER that is also differentiable: all literature on learning decoders minimize only BER (Kim et al., 2020; Nachmani et al., 2018; Dörner et al., 2017). Consequently, improvements in BLER with trained encoders and/or decoders are rare. We discover a code that improves both BER and BLER, and we observe a similar gain with KO$(8, 2)$ in Figure 4. Performance of a binarized version KO-b$(8, 2)$ is also shown, which we describe further in §D.

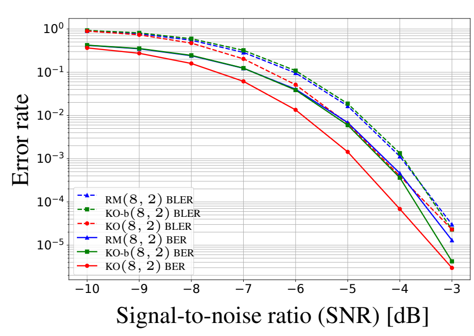


Figure 4. Neural network based KO$(8, 2)$ and KO-b$(8, 2)$ improve upon RM$(8, 2)$ in BER and BLER, but the gain is small for the binarized codewords of KO-b$(8, 2)$ (for all the codes, the code dimension is 37 and block length is 256).

4.2. Interpreting KO codes

We interpret the learned encoders and decoders to explain the source of the performance gain.

Interpreting the KO encoder. To interpret the learned KO code, we examine the pairwise distance between codewords. In classical linear coding, pairwise distances are expressed in terms of the weight distribution of the code, which counts how many codewords of each specific Hamming weight $1, 2, \ldots, n$ exist in the code. The weight distribution of linear codes are used to derive analytical bounds, that can be explicitly computed, on the BER and BLER over AWGN channels (Sason & Shamai, 2006). For nonlinear codes, however, the weight distribution does not capture pairwise distances. Therefore, we explore the distribution of all the pairwise distances of non-linear KO codes that can play the same role as the weight distribution does for linear codes.

The pairwise distance distribution of the RM codes remains an active area of research as it is used to prove that RM codes achieve the capacity (Kaufman et al., 2012; Abbe et al., 2015; Sberlo & Shpilka, 2020) (Figure 5 blue). However, these results are asymptotic in the block length and do not guarantee a good performance, especially in the small-to-medium block lengths that we are interested in. On the other hand, Gaussian codebooks, codebooks randomly picked from the ensemble of all Gaussian codebooks, are known to be asymptotically optimal, i.e., achieving the capacity (Shannon, 1948), and also demonstrate optimal finite-length scaling laws closely related to the pairwise distance distribution (Polyanskiy et al., 2010) (Figure 5 orange).

Remarkably, the pairwise distance distribution of KO code shows a staggering resemblance to that of the Gaussian codebook of the same rate ρ and blocklength n (Figure 5 red). This is an unexpected phenomenon since we minimize only BER. We posit that the NN training has learned to construct a Gaussian-like codebook, in order to minimize BER. Most importantly, unlike the Gaussian codebook, KO codes constructed via NN training are fully compatible with efficient decoding. This phenomenon is observed for all order-2 codes we trained (e.g., Figure 2 for KO$(9, 2)$).

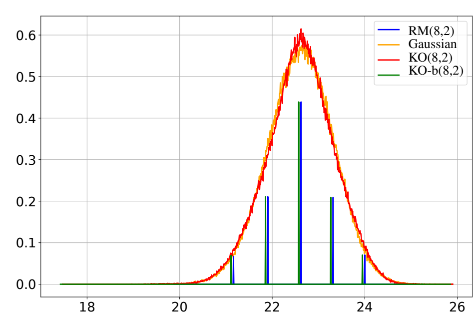


Figure 5. Histograms of pairwise distances between codewords for $(8, 2)$ codes reveal that KO$(8, 2)$ code has learned an approximate Gaussian codebook that can be efficiently decoded.

Interpreting the KO decoder. We now analyze how the KO decoder contributes to the gains in BLER over the RM decoder. Let $\boldsymbol{m} = (\boldsymbol{m}_{(7,1)}, \ldots, \boldsymbol{m}_{(2,2)})$ denote the block of transmitted message bits, where the ordered set of indices $\mathcal{L} = \{(7,1), \ldots, (2,2)\}$ correspond to the leaf branches (RM codes) of the Plotkin tree. Let $\hat{\boldsymbol{m}}$ be the decoded estimate by the KO$(8,2)$ decoder.

We provide Plotkin trees of RM$(8,2)$ and KO$(8,2)$ decoders in Figures 14a and 14b in the appendix. Recall that for this KO$(8,2)$ decoder, similar to the KO$(3,1)$ decoder in Figure 3d, we decode each sub-code in the leaves sequentially, starting from the $(7,1)$ branch down to $(2,2)$: $\hat{\boldsymbol{m}}_{(7,1)} \to \ldots \to \hat{\boldsymbol{m}}_{(2,2)}$. In view of this decoding order, BLER, defined as $\mathbb{P}\left[\hat{\boldsymbol{m}} \neq \boldsymbol{m}\right]$, can be decomposed as

$$\mathbb{P}\left[\hat{\boldsymbol{m}} \neq \boldsymbol{m}\right] = \sum_{i \in \mathcal{L}} \mathbb{P}\left[\hat{\boldsymbol{m}}_i \neq \boldsymbol{m}_i, \hat{\boldsymbol{m}}_{1:i-1} = \boldsymbol{m}_{1:i-1}\right]. \quad (4)$$

In other words, BLER can also be represented as the sum of the fraction of errors the decoder makes in each of the leaf branches when no errors were made in the previous ones. Thus, each term in Eq. (4) can be viewed as the contribution of each sub-code to the total BLER.

This is plotted in Figure 6, which shows that the KO$(8,2)$ decoder achieves better BLER than the RM$(8,2)$ decoder by making major gains in the leftmost $(7,1)$ branch (which is decoded first) at the expense of other branches. However, the decoder (together with the encoder) has learnt to better balance these contributions evenly across all branches, resulting in lower BLER overall. The unequal errors in the branches of the RM code has been observed before, and some efforts made to balance them (Dumer & Shabunov, 2001); that KO codes learn such a balancing scheme purely from data is, perhaps, remarkable.

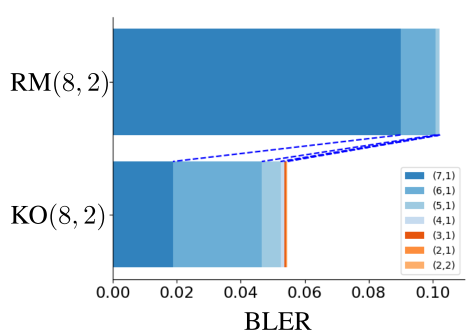


Figure 6. Separating each sub-code contribution in the KO$(8,2)$ decoder and the RM$(8,2)$ decoder reveals that KO$(8,2)$ improves in the total BLER by balancing the contributions more evenly over the sub-codes.

4.3. Robustness to non-AWGN channels

As the environment changes dynamically in real world channels, robustness is crucial in practice. We therefore test the KO code under canonical channel models and demonstrate robustness, i.e., the ability of a code trained on AWGN to perform well under a different channel *without retraining*. It is well known that Gaussian noise is the worst case noise among all noise with the same variance (Lapidoth, 1996; Shannon, 1948) when an optimal decoder is used, which might take an exponential time. When decoded with efficient decoders, as we do with both RM and KO codes, catastrophic failures have been reported in the case of Turbo decoders (Kim et al., 2018). We show that both RM codes and KO codes are robust and that KO codes maintain their gains over RM codes as the channels vary.

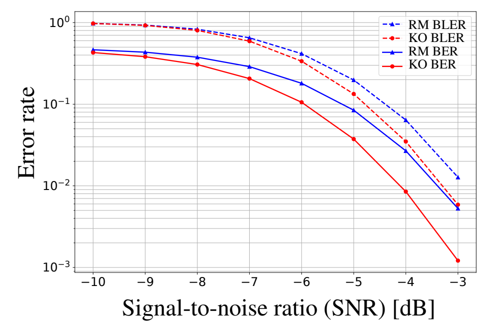


Figure 7. KO$(8,2)$ trained on AWGN is robust when tested on a fast fading channel and maintains a significant gain over RM(8,2).

We first test on a *Rayleigh fast fading channel*, defined as $y_i = a_i x_i + n_i$, where x_i is the transmitted symbol, y_i is the received symbol, $n_i \sim \mathcal{N}(0, \sigma^2)$ is the additive Gaussian noise, and a is from a Rayleigh distribution with the variance of a chosen as $\mathbb{E}[a_i^2] = 1$.

We next test on a bursty channel, defined as $y_i = x_i + n_i + w_i$, where x_i is the input symbol, y_i is the received symbol, $n_i \sim \mathcal{N}(0, \sigma^2)$ is the additive Gaussian noise, and $w_i \sim \mathcal{N}(0, \sigma_b^2)$ with probability ρ and $w_i = 0$ with probability $1 - \rho$. In the experiment, we choose $\rho = 0.1$ and $\sigma_b = \sqrt{2}\sigma$.

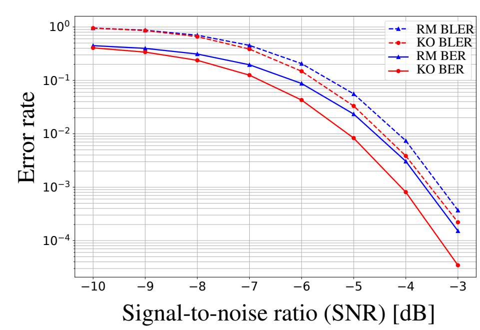


Figure 8. KO$(8,2)$ trained on AWGN is robust when tested on a bursty channel and maintains a significant gain over RM$(8,2)$.

4.4. Complexity of KO decoding

Ultra-Reliable Low Latency Communication (URLLC) is increasingly required for modern applications including vehicular communication, virtual reality, and remote robotics (Sybis et al., 2016; Jiang et al., 2020). In general, a KO(m, r) code requires $O(n \log n)$ operations to decode which is the same as the efficient Dumer's decoder for an RM(m, r) code, where $n = 2^m$ is the block length. More precisely, the successive cancellation decoder for RM$(8, 2)$ requires 11268 operations whereas KO$(8, 2)$ requires 550644 operations which we did not try to optimize for this project. We discuss promising preliminary results in reducing the computational complexity in §C, where KO decoders achieve a computational efficiency comparable to the successive cancellation decoders of RM codes.

5. KO codes improve upon Polar codes

Results from §4 demonstrate that our KO codes significantly improve upon RM codes on a variety of benchmarks. Here, we focus on a different family of capacity-achieving landmark codes: *Polar codes* (Arikan, 2009).

Polar and RM codes are closely related, especially from an encoding point of view. The generator matrices of both codes are chosen from the same parent square matrix by following different row selection rules. More precisely, consider a RM(m, r) code that has code dimension $k = \sum_{i=0}^{r} \binom{m}{i}$ and blocklength $n = 2^m$. Its encoding generator matrix is obtained by picking the k rows of the square matrix $\boldsymbol{G}_{n\times n} := \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{\otimes m}$ that have the largest Hamming weights (i.e., Hamming weight of at least 2^{m-r}), where $[\cdot]^{\otimes m}$ denotes the m-th Kronecker power. The Polar encoder, on the other hand, picks the rows of $\boldsymbol{G}_{n\times n}$ that correspond to the most reliable bit-channels (Arikan, 2009).

The recursive Kronecker structure inherent to the parent matrix $\boldsymbol{G}_{n\times n}$ can also be represented by a computation graph: a complete binary tree. Thus the corresponding computation tree for a Polar code is obtained by freezing a set of leaves (row-selection). We refer to this encoding computation graph of a Polar code as its *Plotkin tree*. This Plotkin tree structure of Polar codes enables a matching efficient decoder: the *successive cancellation* (SC). The SC decoding algorithm is similar to Dumer's decoding for RM codes. Hence, Polar codes can be completely characterized by their corresponding Plotkin trees.

Inspired by the Kronecker structure of Polar Plotkin trees, we design a new family of KO codes to strictly improve upon them. We build a novel NN architecture that capitalizes on the Plotkin tree skeleton and generalizes it to nonlinear codes. This enables us to discover new nonlinear algebraic structures. The KO encoder and decoder can be trained in an end-to-end manner using variants of stochastic gradient descent (§A).

In Figure 9, we compare the performance of our KO code with its competing Polar$(64, 7)$ code, i.e., code dimension $k = 7$ and block length $n = 64$, in terms of BER. Figure 9 highlights that our KO code achieves significant gains over Polar$(64, 7)$ on a wide range of SNRs. In particular, we obtain a gain of almost 0.7 dB compared to that of Polar at the BER 10^{-4}. For comparison we also plot the performance of both codes with the optimal MAP decoding. We observe that the BER curve of our KO decoder, unlike the SC decoder, almost matches that of the MAP decoder, convincingly demonstrating its optimality.

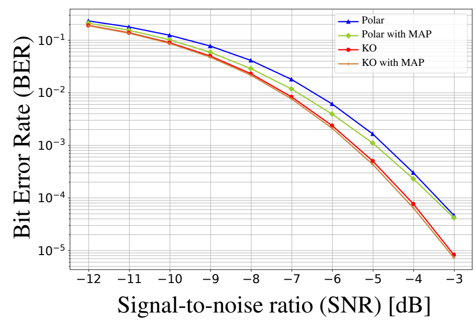


Figure 9. Neural network based KO code improves upon the Polar$(64, 7)$ code when trained on AWGN channel. KO decoder also matches the optimal MAP decoder.

We also observe similar improvements for BLER (Figure 11, §A). This successful case study with training KO (encoder, decoder) pairs further demonstrates that our novel neural architectures seamlessly generalize to codes with an underlying Kronecker product structure.

6. Related work

There is tremendous interest in the coding theory community to incorporate deep learning methods. In the context of channel coding, the bulk of the works focus on decoding known linear codes using data-driven neural decoders (Nachmani et al., 2016; O'shea & Hoydis, 2017; Dörner et al., 2017; Nachmani et al., 2018; Kim et al., 2018; Jiang et al., 2019a); even here, most works have limited themselves to small block lengths due to the difficulty in generalization (for instance, even when nearly 90% of the codewords of a rate $1/2$ Polar code over 8 information bits are exposed to the neural decoder (Gruber et al., 2017)).

On the other hand, very few works in the literature focus on discovering *both* encoders and decoders; the few which do, operate at very small block lengths (O'Shea et al., 2016; O'shea & Hoydis, 2017). One of the major challenges here is to jointly train the (encoder, decoder) pairs without getting stuck in local optima as the losses are non-convex. In (Jiang et al., 2019b), the authors employ clever training tricks to learn a novel autoencoder based codes that outperform the classical Turbo codes, which are sequential in nature. In contrast, here we focus on the generalizations of the Kronecker operation that underpins the RM and Polar family.

RM and Polar codes have seen active research, especially on improving decoding using neural networks: (Xu et al., 2017; Cammerer et al., 2017; Doan et al., 2018; Carpi et al., 2019). (Ebada et al., 2019) proposes a method to learn the frozen indices of a Polar code together with the weights in the belief propagation (BP) factor graph that improves upon the classical BP decoding. However, BP decoder is suboptimal for Polar codes; in comparison, we strictly improve upon their natural SC decoder (Figure 9).

7. Conclusion

We introduce KO codes that generalize the recursive Kronecker operation crucial to designing RM and Polar codes. Using the computation tree (known as a Plotkin tree) of these classical codes as a skeleton, we propose a novel neural network architecture tailored for channel communication. Training over the AWGN channel, we discover the first family of *non-linear* codes that are not built upon any linear structure. KO codes significantly outperform the baseline Polar and RM codes under similar successive cancellation decoding architectures, which we call Dumer's decoder for the RM codes. The pairwise distance profile reveals that KO code combines the analytical structure of algebraic codes with the random structure of the celebrated random Gaussian codes.

References

Abbe, E., Shpilka, A., and Wigderson, A. Reed–muller codes for random erasures and errors. *IEEE Transactions on Information Theory*, 61(10):5229–5252, 2015.

Abbe, E., Shpilka, A., and Ye, M. Reed-muller codes: Theory and algorithms, 2020.

Alon, N., Kaufman, T., Krivelevich, M., Litsyn, S., and Ron, D. Testing reed-muller codes. *IEEE Transactions on Information Theory*, 51(11):4032–4039, 2005.

Arikan, E. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Transactions on information Theory*, 55(7):3051–3073, 2009.

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

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

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

Dörner, S., Cammerer, S., Hoydis, J., and Ten Brink, S. Deep learning based communication over the air. *IEEE Journal of Selected Topics in Signal Processing*, 12(1): 132–143, 2017.

Dumer, I. Recursive decoding and its performance for low-rate reed-muller codes. *IEEE Transactions on Information Theory*, 50(5):811–823, 2004.

Dumer, I. Soft-decision decoding of reed-muller codes: a simplified algorithm. *IEEE transactions on information theory*, 52(3):954–963, 2006.

Dumer, I. and Shabunov, K. Near-optimum decoding for subcodes of reed-muller codes. In *Proceedings. 2001 IEEE International Symposium on Information Theory*, pp. 329. IEEE, 2001.

Dumer, I. and Shabunov, K. Soft-decision decoding of reed-muller codes: recursive lists. *IEEE Transactions on information theory*, 52(3):1260–1266, 2006a.

Dumer, I. and Shabunov, K. Soft-decision decoding of reed-muller codes: recursive lists. *IEEE Transactions on information theory*, 52(3):1260–1266, 2006b.

Ebada, M., Cammerer, S., Elkelesh, A., and ten Brink, S. Deep learning-based polar code design. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 177–183. IEEE, 2019.

Gruber, T., Cammerer, S., Hoydis, J., and ten Brink, S. On deep learning-based channel decoding. In *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6. IEEE, 2017.

Jamali, M. V., Liu, X., Makkuva, A. V., Mahdavifar, H., Oh, S., and Viswanath, P. Reed-Muller subcodes: Machine learning-aided design of efficient soft recursive decoding. *arXiv preprint arXiv:2102.01671*, 2021.

Jiang, Y., Kannan, S., Kim, H., Oh, S., Asnani, H., and Viswanath, P. Deepturbo: Deep turbo decoder. In *2019*

IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), pp. 1–5. IEEE, 2019a.

Jiang, Y., Kim, H., Asnani, H., Kannan, S., Oh, S., and Viswanath, P. Turbo autoencoder: Deep learning based channel codes for point-to-point communication channels. In *Advances in Neural Information Processing Systems*, pp. 2758–2768, 2019b.

Jiang, Y., Kim, H., Asnani, H., Kannan, S., Oh, S., and Viswanath, P. Learn codes: Inventing low-latency codes via recurrent neural networks. *IEEE Journal on Selected Areas in Information Theory*, 2020.

Kaufman, T., Lovett, S., and Porat, E. Weight distribution and list-decoding size of reed–muller codes. *IEEE transactions on information theory*, 58(5):2689–2696, 2012.

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

Kim, H., Oh, S., and Viswanath, P. Physical layer communication via deep learning. *IEEE Journal on Selected Areas in Information Theory*, 2020.

Kudekar, S., Kumar, S., Mondelli, M., Pfister, H. D., Şaşoğlu, E., and Urbanke, R. L. Reed–muller codes achieve capacity on erasure channels. *IEEE Transactions on information theory*, 63(7):4298–4316, 2017.

Lapidoth, A. Nearest neighbor decoding for additive non-gaussian noise channels. *IEEE Transactions on Information Theory*, 42(5):1520–1529, 1996.

Ma, Z., Xiao, M., Xiao, Y., Pang, Z., Poor, H. V., and Vucetic, B. High-reliability and low-latency wireless communication for internet of things: challenges, fundamentals, and enabling technologies. *IEEE Internet of Things Journal*, 6(5):7946–7970, 2019.

Nachmani, E., Be'ery, Y., and Burshtein, D. Learning to decode linear codes using deep learning. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 341–346. IEEE, 2016.

Nachmani, E., Marciano, E., Lugosch, L., Gross, W. J., Burshtein, D., and Be'ery, Y. Deep learning methods for improved decoding of linear codes. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):119–131, 2018.

O'Shea, T. J., Karra, K., and Clancy, T. C. Learning to communicate: Channel auto-encoders, domain specific regularizers, and attention. In *2016 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pp. 223–228. IEEE, 2016.

O'shea, T. and Hoydis, J. An introduction to deep learning for the physical layer. *IEEE Transactions on Cognitive Communications and Networking*, 3(4):563–575, 2017.

Plotkin, M. Binary codes with specified minimum distance. *IRE Transactions on Information Theory*, 6(4):445–450, 1960.

Polyanskiy, Y., Poor, H. V., and Verdú, S. Channel coding rate in the finite blocklength regime. *IEEE Transactions on Information Theory*, 56(5):2307–2359, 2010.

Reed, I. A class of multiple-error-correcting codes and the decoding scheme. *Transactions of the IRE Professional Group on Information Theory*, 4(4):38–49, 1954.

Richardson, T. and Urbanke, R. *Modern coding theory*. Cambridge University Press, 2008.

Sason, I. and Shamai, S. Performance analysis of linear codes under maximum-likelihood decoding: A tutorial. 2006.

Sberlo, O. and Shpilka, A. On the performance of reed-muller codes with respect to random errors and erasures. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1357–1376. SIAM, 2020.

Senior, A. W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Žídek, A., Nelson, A. W., Bridgland, A., et al. Protein structure prediction using multiple deep neural networks in the 13th critical assessment of protein structure prediction (casp13). *Proteins: Structure, Function, and Bioinformatics*, 87(12):1141–1148, 2019.

Shannon, C. E. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Sybis, M., Wesolowski, K., Jayasinghe, K., Venkatasubramanian, V., and Vukadinovic, V. Channel coding for ultra-reliable low-latency communication in 5g systems. In *2016 IEEE 84th vehicular technology conference (VTC-Fall)*, pp. 1–5. IEEE, 2016.

Tal, I. and Vardy, A. How to construct polar codes. *IEEE Trans. Inf. Theory*, 59(10):6562–6582, 2013.

Tal, I. and Vardy, A. List decoding of polar codes. *IEEE Transactions on Information Theory*, 61(5):2213–2226, 2015.

Udrescu, S.-M. and Tegmark, M. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.

Welling, M. Neural augmentation in wireless communication, 2020.

Xu, W., Wu, Z., Ueng, Y.-L., You, X., and Zhang, C. Improved polar decoder based on deep learning. In *2017 IEEE International workshop on signal processing systems (SiPS)*, pp. 1–6. IEEE, 2017.

Ye, M. and Abbe, E. Recursive projection-aggregation decoding of reed-muller codes. *IEEE Transactions on Information Theory*, 66(8):4948–4965, 2020.