DEEPPOLAR: Inventing Nonlinear Large-Kernel Polar Codes via Deep Learning

S Ashwin Hebbar* [1] **Sravan Kumar Ankireddy*** [2] **Hyeji Kim** [2] **Sewoong Oh** [3] **Pramod Viswanath** [1]

Abstract

Progress in designing channel codes has been driven by human ingenuity and, fittingly, has been sporadic. Polar codes, developed on the foundation of Arikan's polarization kernel, represent the latest breakthrough in coding theory and have emerged as the state-of-the-art error-correction code for short-to-medium block length regimes. In an effort to automate the invention of good channel codes, especially in this regime, we explore a novel, non-linear generalization of Polar codes, which we call DEEPPOLAR codes. DEEPPOLAR codes extend the conventional Polar coding framework by utilizing a larger kernel size and parameterizing these kernels and matched decoders through neural networks. Our results demonstrate that these data-driven codes effectively leverage the benefits of a larger kernel size, resulting in enhanced reliability when compared to both existing neural codes and conventional Polar codes. Source code is available at this link.

1. Introduction

Reliable digital communication is a primary workhorse of the information age. To ensure reliable communication over a noisy channel, it is common to introduce redundancy in the transmitted data to enable faithful reconstruction of the message by receivers. This crucial process, known as error correction coding (channel coding), lies at the heart of both wired (Ethernet, cable) and wireless (cellular, WiFi, satellite) communication systems. Over the past seven decades, a significant research thrust has focused on designing reliable codes (consisting of an encoder-decoder pair) that achieve good reliability whilst having an efficient decoder. The canonical setting is one of point-to-point reliable communication over the additive white Gaussian noise (AWGN) channel, and the performance of a code in this setting is its gold standard. The figure of merit can be 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.

The field of coding theory has experienced sporadic yet significant breakthroughs, largely propelled by human ingenuity. Polar codes, invented by Arikan in 2009 (Arikan, 2009), is one of the most profound developments in coding theory, significantly revitalizing the field. Polar codes, a combination of algebraic and graphical coding structures, are the first class of codes with a deterministic construction proven to achieve Shannon capacity. Importantly, this is achieved with low-complexity encoding and decoding. The impact of Polar codes is evident from their integration into the 5G standards within just a decade of their proposal – a remarkably swift timeline considering that it typically takes several decades for new coding methods to be incorporated into cellular standards (3GPP, 2018; Bioglio et al., 2020).

The basic building block of Polar codes is a binary matrix $G = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, called the polarization *kernel*. The generative matrix, which characterizes a linear code, is obtained by several Kronecker products of G with itself. This construction of polar codes gives rise to a remarkable phenomenon known as "channel polarization". This process transforms n views of a binary memoryless channel into n synthetic "bit channels", each with distinct reliability. As the block length n grows asymptotically large, these bit channels become *polarized*, becoming either completely noiseless or completely noisy. Polar encoding proceeds by sending information bits in the noiseless bit channels while the noisy input bits are "frozen" to a known value. Capacity is achieved via the sequential successive cancellation (SC) decoder.

Polar codes and SC decoding are optimal for large asymptotic blocklengths; however, practical finite-length performance is lacking. Recognizing this limitation, recent research has focused on augmenting the polar encoder and improving the decoding performance. Specifically, the concatenation of cyclic redundancy check (CRC) with polar codes improves distance properties. Combining this with successive cancellation list (SCL) decoding markedly enhances decoding performance (Tal & Vardy, 2015; Niu & Chen,

*Lead authors [1]Princeton University [2]University of Texas at Austin [3]University of Washington. Correspondence to: Ashwin Hebbar <hebbar@princeton.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

2012). Consequently, Polar codes have emerged as the state-of-the-art in the short-to-medium block length regime, leading to their inclusion in 5G standards. Nonetheless, SCL decoding introduces considerable decoding complexity and latency, marking a trade-off between performance and computational efficiency.

In parallel, there have been efforts to improve polar encoders. One approach involves modifying the Polar encoding structure. A noteworthy improvement is polarization-adjusted convolutional (PAC) codes (Arıkan, 2019), which use convolutional precoding before polar coding. Remarkably, PAC codes approach finite-length information-theoretic bounds for binary codes in the short block length regime. However, the practical application of PAC codes is limited by their high decoding complexity, necessitating extremely large list sizes for effective decoding.

An alternative idea involves increasing the size of the polarization kernel. Indeed, (Korada et al., 2010) prove that polarization holds for all kernels provided they are not unitary and not upper triangular under any column permutation. Further, one can find large polarization kernels ($\ell \geq 8$) that achieve faster polarization (characterized by the "scaling exponent"). Despite these theoretical advantages, Polar codes with large kernels are not preferred in practice due to their increased decoding complexity. Nevertheless, there is recent work addressing these challenges (Trifonov, 2023) based on specially designed polarization kernels.

In this work, we aim to address the question: How can we automate the search for good codes? Indeed, it is possible by parameterizing and learning both the encoder and decoder using neural networks. However, constructing effective non-linear codes using this approach is highly challenging: it is well-documented in literature that naively parameterizing with off-the-shelf neural architectures often results in performance worse than even repetition codes, as elaborated further in App. F.3. Rather, a more promising approach is to design neural architectures that enable structured redundancy. Specifically, our work delves into the innovative intersection between algebraic coding theory and machine learning by exploring non-linear generalizations of polarization-driven structures. This can be achieved by parameterizing each kernel by a neural network, combining the information-theoretic properties of polarization-driven code structures with the adaptability and learning capabilities of deep learning. This interplay between algebraic coding structures and deep learning is a relatively uncharted territory. Our algorithm builds upon the groundwork laid by (Makkuva et al., 2021), who introduced KO codes as a non-linear generalization of RM codes. Although this work marked a significant leap forward, it is limited by its dependence on Reed-Muller (RM) encoding-decoding schemes, which restricted its applicability across a broader range of rates. Importantly, the architecture of KO codes proved inapplicable for Polar codes, being only scalable upto a (64,7) code. In contrast, our work proposes DEEPPOLAR codes, a non-linear generalization of the encoding and decoding structures of Polar codes (which encompass RM codes as a special case). This allows us to scale seamlessly to various rates and block lengths.

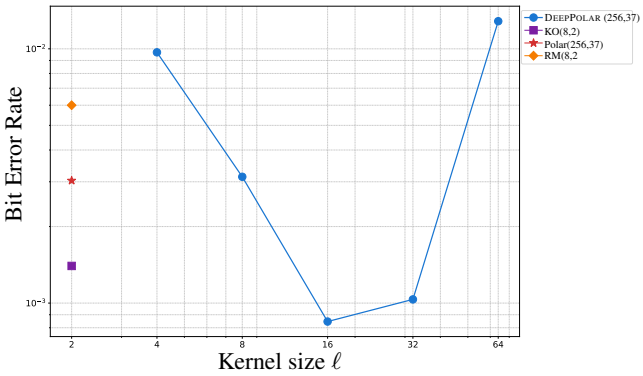


Figure 1. DEEPPOLAR (n=256, k=37) with appropriate kernel sizes (e.g., $\ell = 16, 32$) outperforms classical Reed-Muller, Polar, and state-of-the-art neural KO codes (Makkuva et al., 2021) on AWGN channels with -2dB SNR

The core innovation of DEEPPOLAR lies in its *utilization of larger-size kernels*, which enables the neural network to explore an expansive function space. This is complemented by a matching SC-like neural decoder, which is jointly trained on samples drawn from an AWGN channel. Larger kernel sizes are pivotal for DEEPPOLAR achieving substantial performance improvements on the canonical AWGN channel, as evidenced in Fig. 1 for the case of $n = 256, k = 37$. Through extensive empirical studies, we find that a kernel size $\ell = \sqrt{n}$ is most effective in balancing bias and variance, enabling it to achieve significantly lower bit error rates over the baseline Polar and RM codes, as well as the KO$(8, 2)$ code, SOTA at this block length and rate. Additionally, we design a training curriculum that builds upon the inherent nested hierarchy of polar coding structures to accelerate training. This combination of principled coding-theoretic structures with a targeted training methodology is designed to enable effective generalization across the vast space of messages. In summary, we make the following contributions:

- We propose DEEPPOLAR, a novel generalization of Polar codes via large non-linear NN-based kernels.
- DEEPPOLAR outperforms the classical Polar and Reed-Muller codes, state-of-the-art binary linear codes, as well as KO codes, state-of-the-art ML-based codes, whilst scaling to various code rates.
- We develop a principled curriculum-based training

methodology that allows DEEPPOLAR to generalize to the challenging high SNR regime.

2. Problem formulation and Background

2.1. Channel coding

Channel coding is a technique to add redundancy to the transmission to make it robust against noise added by the communication channel. More precisely, let $\boldsymbol{u} = (u_0, \ldots, u_{k-1}) \in \{0,1\}^k$ denote a block of *information/message* bits that we wish to transmit. A code consists of an encoder and decoder pair. The encoder $g_\phi : \{0,1\}^k \to \{0,1\}^n$ maps these message bits into a binary codeword $\boldsymbol{x}$ of length n, i.e. $\boldsymbol{x} = g(\boldsymbol{u})$. The codewords are mapped to real/complex values by modulation (eg, Binary Phase Shift Keying (BPSK)). The channel, denoted as $W_{Y|X}(\cdot|\cdot)$, corrupts the codeword $\boldsymbol{x}$ to its noisy version $\boldsymbol{y} \in \mathbb{R}^n$. Upon receiving the corrupted codeword, the decoder f_θ estimates the message bits as $\hat{\boldsymbol{u}} = f_\theta(\boldsymbol{y})$. The performance of the code is measured using standard error metrics such as Bit-Error-Rate (BER) or Block-Error-Rate (BLER): $\mathrm{BER}(g_\phi, f_\theta) \triangleq (1/k)\sum_i \mathbb{P}[\hat{u}_i \neq u_i]$, whereas $\mathrm{BLER}(g_\phi, f_\theta) \triangleq \mathbb{P}[\hat{\boldsymbol{u}} \neq \boldsymbol{u}]$.

Figure 2. Channel coding via deep learning

2.2. Polar codes

Polar codes, introduced by Erdal Arıkan (Arikan, 2009), are the first deterministic code construction to achieve the Shannon capacity while maintaining low encoding and decoding complexity. This section formally defines Polar codes and motivates our method.

Polar Encoding. A Polar code can be described by Polar(n,k,$\mathcal{F}$). Here, n is the block length; $n = 2^m$ for some integer m, k is the number of information bits, and $\mathcal{F}$ represents the set of "frozen" bit positions. Typically, the positions corresponding to the noisiest $n-k$ bit channels arising due to polarization are chosen to be frozen. A polar encoder maps information bits $\boldsymbol{u} \in \{0,1\}^k$ to a binary codeword $\boldsymbol{x} \in \{0,1\}^n$. The basic building block of Polar codes is the Plotkin transform: $\{0,1\}^d \times \{0,1\}^d \to \{0,1\}^{2d}$. This mapping for a pair of input bits (u, v) can be represented by the matrix $G_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, transforming (u_0, u_1) into $(u_0 \oplus u_1, u_1)$, where $\oplus$ denotes the XOR operation. Consistent with the coding theory literature, we term such a building block a *kernel*. The encoding matrix for block length $n = 2^m$ is obtained by taking the *Kronecker product* of the base kernel G_2 m times.

Utilizing this structure, the encoding can be efficiently performed via a recursive coordinate-wise application of the Plotkin transform on a binary tree, called the *Plotkin tree*. To encode a block of message bits $\boldsymbol{u} = (u_0, \ldots, u_{k-1}) \in \{0,1\}^k$, we first embed them into a source message vector $\boldsymbol{m} \triangleq (m_1, \ldots, m_n) = (0, \ldots, u_0, 0, \ldots, u_1, 0, \ldots, u_{k-1}, 0, \ldots) \in \{0,1\}^n$, where $\boldsymbol{m}_{\mathcal{I}} = \boldsymbol{u}$ and $\boldsymbol{m}_{\mathcal{I}^C} = 0$ for some $\mathcal{I} \subseteq [n]$. Since the message block $\boldsymbol{m}$ contains the information bits $\boldsymbol{u}$ only at the indices pertaining to $\mathcal{I}$, the set $\mathcal{I}$ is called the *information set*, and its complement $\mathcal{F} = \mathcal{I}^C$ the *frozen set*. We describe the encoding on a Plotkin tree via a small example, Polar(4,3) - illustrated in Fig. 3(a). Here, $\mathcal{F}$={0}. Consider an input of size $k = 3$, $u = [u_0, u_1, u_2]$. At the input level (depth 1), we freeze m_0, i.e., $m_0 = 0$ and assign $\boldsymbol{u}$ to the remaining positions. Applying the Plotkin transform, $(0, u_0) \to (0 \oplus u_0, u_0)$ and $(u_1, u_2) \to (u_1 \oplus u_2, u_2)$. At the second level, we apply the same operation *coordinatewise* to these vectors *i.e,* $((0 \oplus u_0), (u_1 \oplus u_2)) \to ((0 \oplus u_0) \oplus (u_1 \oplus u_2), u_1 \oplus u_2)$ and $(u_0, u_2) \to (u_0 \oplus u_2, u_2)$. The final encoded vector is the concatenation of the outputs from the second-level nodes *i.e,* $(u_0 \oplus u_1 \oplus u_2, u_0 \oplus u_2, u_1 \oplus u_2, u_2)$. For a general (n, k) polar code, the encoding proceeds similarly up to $m = \log_2 n$ levels.

Polar decoding. The encoded messages are corrupted by a noisy channel $W \triangleq W_{Y|X}$. The successive cancellation (SC) algorithm is one of the most efficient decoders for Polar codes and is optimal asymptotically. The basic principle behind the SC algorithm is to sequentially decode each message bit u_i according to the conditional likelihood given the corrupted codeword y and previously decoded bits $\hat{\boldsymbol{u}}^{(i-1)}$. LLR for i^{th} bit can be computed as

$$L_i = \log\left(\frac{\mathbb{P}(u_i = 0|y, \hat{\boldsymbol{u}}^{(i-1)})}{\mathbb{P}(u_i = 1|(y, \hat{\boldsymbol{u}}^{(i-1)})}\right). \tag{1}$$

SC decoding is described in detail in App. A.

Large kernel Polar codes. Polar codes are optimal (capacity-achieving) at asymptotic blocklengths due to the phenomenon of channel polarization. (Korada et al., 2010) prove that replacing the conventional 2×2 kernel with an $\ell \times \ell$ binary kernel ($\ell > 2$) still results in polarization, provided this matrix is non-singular and not upper triangular under any column permutation. Further, large kernel polar codes that achieve capacity at shorter block lengths compared to conventional polar codes have been found, as indicated by better *scaling exponents* (Fazeli & Vardy, 2014; Fazeli et al., 2020). Notably, the kernel size must be expanded to 8 to surpass Arikan's $G^{\otimes 2}$ kernel, as no size 4 linear kernel offers an improved scaling exponent. These kernels enable better finite-length performance at the cost of increased decoding complexity.

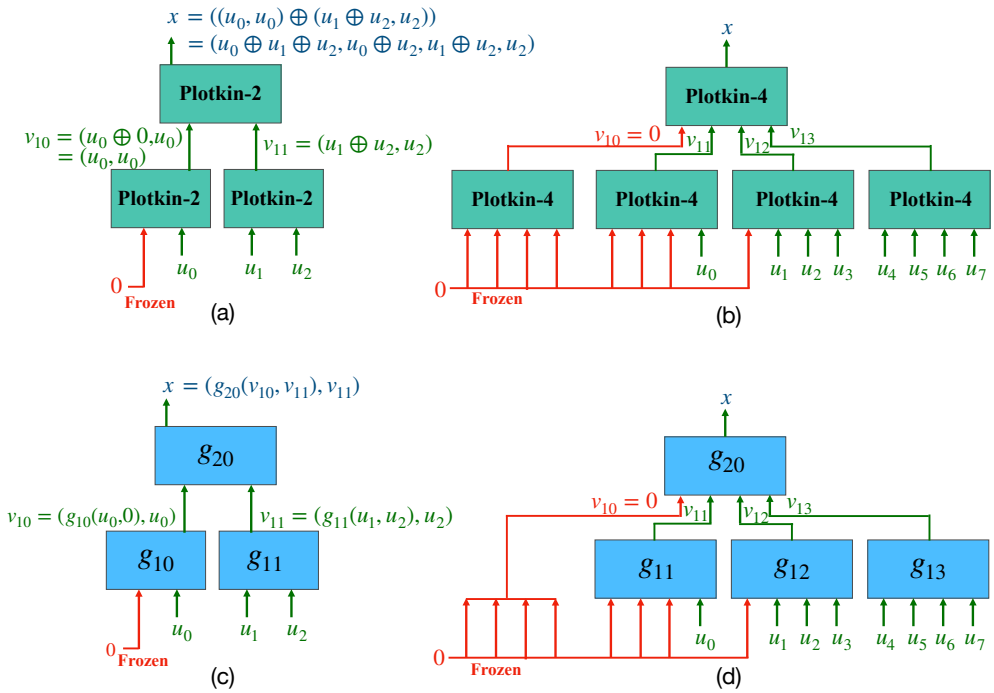


Figure 3. (a) Polar$(4, 3)$ encoding structure using the standard 2×2 kernel. Encoding is performed recursively on the Plotkin tree. (b) Polar$(16, 8)$ encoding using 4×4 kernels. (c) DEEPPOLAR $(4, 3, \ell = 2)$ replaces the xor operation in Plotkin - 2×2 by neural networks. (d) DEEPPOLAR $(16, 8, \ell = 4)$: Scaling the DEEPPOLAR encoding to higher-order kernels enables us to achieve good reliability. We are the first to explore this design space

Large kernel polar encoding and decoding proceeds similarly to conventional Polar codes via an $\ell \times \ell$ kernel. An exemplar kernel with $\ell = 4$ is simply the Kronecker product of G with itself, $G_4 = G \otimes G$. This is a Plotkin transform with 4 inputs, referred to as *Plotkin-4*. We illustrate this via an example of a $(n = 16, k = 8)$ code with kernel size $\ell = 4$ in Fig. 3 . In this example, the message $\boldsymbol{u}$ is input at the information positions $\mathcal{I} = \mathcal{F}^C = \{7, 9, 10, 11, 12, 13, 14, 15\}$. The remaining positions are frozen to 0. The kernel is applied in parallel to groups of $\ell = 4$ bits. Mirroring the conventional polar encoding, we iteratively follow this process at each level of the tree. At the second level, we apply Plotkin-4 coordinatewise to the vectors $(v_{10}, v_{11}, v_{12}, v_{13})$ to obtain the codeword x.

In this work, we design a non-linear generalization of the polar coding and decoding structure by both expanding the kernel size and parameterizing them as neural networks. By expanding the design space to include non-linearity and larger kernels, we aim to discover more reliable codes within the neural Plotkin code family.

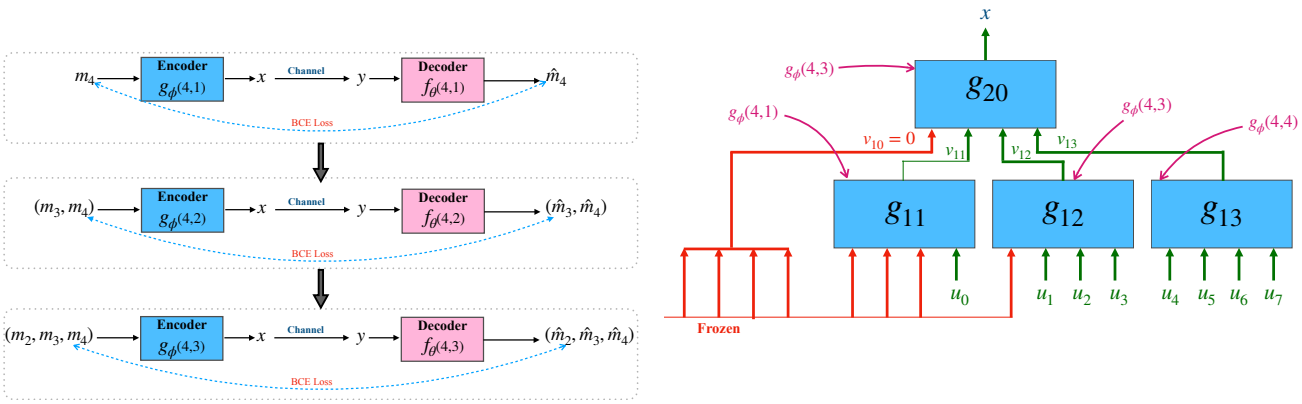
3. DEEPPOLAR codes

3.1. Proposed Architecture

We design DEEPPOLAR codes by generalizing the encoding and decoding structures of large kernel Polar codes. A DEEPPOLAR (n, k, ℓ) code maps bits $\boldsymbol{u} \in \{0, 1\}^k$ to a codeword $\boldsymbol{x} \in \mathbb{R}^n$, using the neural Plotkin tree of kernel size ℓ.

DEEPPOLAR encoder. Building on the foundational encoding structure of Polar codes (Fig. 3a), which employs the recursive application of a binary kernel over a Plotkin tree, we introduce DEEPPOLAR codes as a non-linear extension of this concept. The first major modification in our approach is expanding the conventional 2×2 kernel to a larger $\ell \times \ell$ kernel. This is inspired by the existence of larger binary kernels that outperform the Arikan kernel (App. B.2.)

On the other hand, we generalize the Plotkin transform by replacing each XOR operation with a 2-input NN $g_{d,b}$, where d is the depth in the encoding tree, and b is the index of the NN at depth d. A DEEPPOLAR (n, k, ℓ) inherits the same parameters as a Polar(n, k) code. The encoding structure is analogous to conventional polar codes and is illustrated in Fig. 3(c) - DEEPPOLAR (4,3) with $\ell = 2$. We use the same frozen set $\mathcal{F} = \{0\}$ as the conventional polar code. Applying the neural Plotkin transform, $(0, u_0) \rightarrow v_{10} =$



a)Stage 1 - Curriculum learning for kernels

b)Stage 2 - Initialize encoder and decoder by pretrained kernels

Figure 4. The curriculum to train DEEPPOLAR $(16, 8, \ell = 4)$ proceeds in two phases: (a) In Phase 1, the kernels $(4, i)$ are trained progressively for $i = 1, \cdots, \ell$. (b) In phase 2, we initialize each kernel in the encoder by the respective kernels (and similarly for the decoders). For instance, the kernel g_{11} has one information input bit; we initialize it with a pretrained $g_\phi(4, 1)$. Similarly, g_{20} has three information input bit-groups; we initialize it with a pretrained $g_\phi(4, 3)$.

$(g_{10}(0, u_0), u_0)$ and $(u_1, u_2) \rightarrow v_{11} = (g_{11}(u_1, u_2), u_2)$. At the second level, we apply the kernel g_{20} *coordinate-wise* to these vectors. This coordinatewise operation is the key inductive bias in DEEPPOLAR encoding. The final encoded vector is the concatenation of the outputs from the second-level nodes *i.e,* $(g_{20}(v_{10}, v_{11}), v_{11})$. For a general DEEPPOLAR $(n, k, \ell = 2)$code, the encoding proceeds in a similar way upto $m = \log_2 n$ levels.

Our work synergizes these two directions by employing a non-linear kernel of size $\ell \times \ell$. Fig. 3(d) illustrates DEEPPOLAR $(16, 8, \ell = 4)$, which mirrors the Plotkin-4 encoder for Polar$(16, 8)$. The encoding proceeds similarly, with neural networks $g_{d,b}$ replacing the Plotkin mapping. A power constraint, $\|x\|^2 = n$ is enforced at the output of the encoder. App. E.5 discusses the DEEPPOLAR encoder architecture in detail.

DEEPPOLAR leverages the Polar encoding structure by parameterizing the kernel at each internal node at depth d and bit position b of the Plotkin tree by a neural network $g_{d,b}$. We hypothesize that this innovation unlocks performance gains through the expanded function space afforded by both non-linearization and the increased kernel size. Additionally, this formulation allows us to scale neural Polar codes to larger blocklengths and rates.

DEEPPOLAR-SC decoder. To decode the received codewords corrupted by noise, we design the DEEPPOLAR-SC decoder, a neural generalization of SC decoding. The DEEPPOLAR-SC decoder consists of a decoding tree, consisting of ℓ decoding networks $f_{d,\ell b+j}, j : 0 \rightarrow \ell - 1$, matching every kernel $g_{d,b}$. Notably, each decoding NN $f_{d,\ell b+j}$ is applied coordinatewise and takes as input the LLR from the previous $j - 1$ outputs, along with the ℓ inputs to the kernel. DEEPPOLAR-SC is detailed in App. C.1.

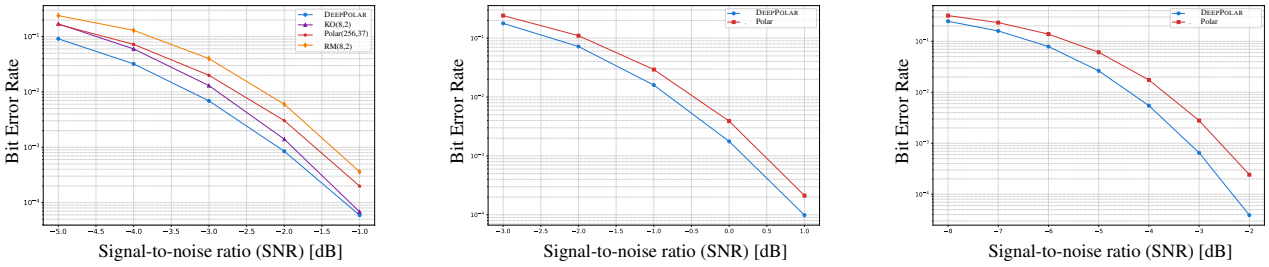
3.2. Training methodology : Curriculum learning.

The primary challenge in training neural channel codes is the astronomical space of potential codewords; the task is to find a noise-robust mapping for each binary string in the k-dimensional Boolean hypercube to a codeword in $\mathbb{R}^n$. This challenge is pronounced even in the short-to-medium blocklength regime we focus on, for instance, training a (256,37) code involves designing 2^{37} codewords in a 256-dimensional space. During training, we encounter a tiny fraction of the message space $(< 1\%)$, making the NN's ability to effectively generalize to unseen messages a pivotal factor in learning effective codes. While principled architectural choices are vital for generalization, the training methodology is equally important. Typically, an end-to-end training strategy would involve minimizing the binary cross entropy loss between the actual message bits and estimated message bits :

$$\mathcal{L} = -\sum_{i=1}^{k} [u_i \log(\hat{u}_i) + (1-u_i) \log(1-\hat{u}_i)] \quad (2)$$

However, as highlighted in Sec. 4.5, direct training often does not generalize well to high SNR scenarios, characterized by infrequent error events.

Curriculum. We address this by introducing a principled two-stage curriculum training procedure that capitalizes on the inherent nested hierarchy of Polar codes. We leverage two key properties of the Polar coding structure: *(1)Hierarchy in k*: A Polar(n, k) subsumes all codewords of lower-rate subcodes Polar$(n, i), 1 \leq i \leq k$. Curriculum training leveraging the hierarchy in k improved both convergence



a) $n = 256, k = 37$ b) $n = 256, k = 64$ c) $n = 256, k = 28$

Figure 5. (a) DEEPPOLAR improves over state-of-the-art KO codes (Makkuva et al., 2021), and RM, Polar codes at n=256,k=37. (b,c) DEEPPOLAR is suitable for a variety of rates and retains gains over Polar, while KO is not suitable for these rates.

speed and reliability in the context of neural polar decoding (Hebbar et al., 2023). In the first stage, we use a similar C2N curriculum to pretrain the parameters for each kernel. We progressively train encoder-decoder pairs for DEEPPOLAR $(n = \ell, k = j, \ell)$ codes, for $j = 1 \to k$ as highlighted in Fig. 4a)), denoted by $g_\phi(\ell, j)$, $f_\theta(\ell, j)$. *(2)Hierarchy in n*: In the Plotkin tree, an $\ell \times \ell$ kernel is applied coordinatewise at each level of the Plotkin tree. In the second stage of our curriculum, we initialize each kernel in DEEPPOLAR (n,k) encoder and decoder with the pre-trained kernels from the first stage, applied at all depths, as illustrated in Fig. 4b). For instance, g_{12} and g_{20}, which have three non-frozen inputs, are initialized by a pre-trained kernel $g_\phi(4, 3)$. Such a systematic and phased training approach significantly enhances the network's learning efficiency and generalizability (Sec. 4.5).

4. Main Results

4.1. Data generation.

We generate synthetic input data for the encoder by randomly sampling from a boolean hypercube *i.e,* $\{0, 1\}^k$. A randomly sampled white Gaussian noise is added to the output of the encoder. During training, the variance of the Gaussian noise is carefully chosen based on the length and rate of the desired channel code.

4.2. Baselines.

DEEPPOLAR extends conventional Polar encoding and decoding structures using neural networks. We evaluate its performance against state-of-the-art learning-based codes, KO codes (Makkuva et al., 2021). Similar to DEEPPOLAR, KO codes replaces elements of Reed-Muller by neural networks to enhance error correction capability. We also compare DEEPPOLAR to traditional Polar codes employing Arikan's kernel and decoded via the successive cancellation (SC) algorithm. Both codes utilize the same frozen positions, determined using the Tal-Vardy method (Tal & Vardy, 2013). We further discuss the impact of frozen position selection in App. B.1. App. B.2 discusses and compares DEEPPOLAR to recent large-kernel polar code constructions.

4.3. Results

DEEPPOLAR codes outperform Polar codes. As highlighted in Fig. 5, DEEPPOLAR codes achieve enhanced performance over traditional Polar codes with SC decoding and Reed-Muller codes with Dumer decoding across a broad range of SNRs in the presence of additive white Gaussian noise (AWGN), in terms of bit error rate (BER). Notably, DEEPPOLAR (256,37,$\ell = 16$) outperforms KO$(8, 2)$ (Fig. 5a)), the state-of-the-art neural code mapping 37 information bits to a length-256 codeword (Makkuva et al., 2021) in the short-to-medium blocklength regime. DEEPPOLAR codes, owing to their Polar-like encoding framework, accommodate a much wider range of rates than KO codes that rely on the algebraic structure of RM codes. This versatility is illustrated in Fig. 5b) and Fig. 5c), where DEEPPOLAR codes outperform Polar codes across diverse rates and SNRs (cf. KO codes and RM codes do not exist for these rates). Further, DEEPPOLAR is robust to non-AWGN deviations, achieving gains over Polar codes on fading channels and radar noise (App. D).

Effect of kernel size. Fig. 6 highlights that scaling the kernel size ℓ is crucial for effective training and DEEPPOLAR outperforming baselines. In case n is not an integer exponent of ℓ - i.e., $\ell^m \leq n < \ell^{m+1}$ and ℓ^m divides n then the kernel size at the root is n/ℓ^m. An increase in kernel size expands the function space the encoder can represent, facilitating learning more robust representations. However, empirical experiments reveal that using $\ell = \sqrt{n}$, which results in an encoding and decoding tree of depth 2, is a good heuristic. Indeed, Fig. 1 highlights significant improvements in performance upto $\ell = 16, 32$, whereas further scaling $\ell = 64$ leads to a decline. This trend indicates a bias-variance tradeoff, suggesting the ideal kernel size strikes a balance between model complexity and its generalization

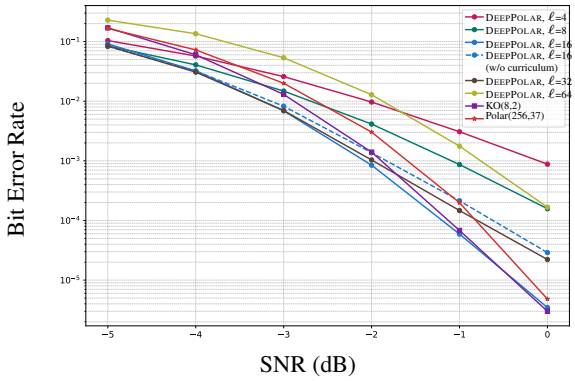


Figure 6. n=256,k=37 : Reliability of DEEPPOLAR improves as the kernel size ℓ is increased. We find that DEEPPOLAR with ℓ=16,32 beats KO codes in BER. Moreover, curriculum training is crucial to improve reliability at high SNRs.

capabilities.

4.4. Interpretation

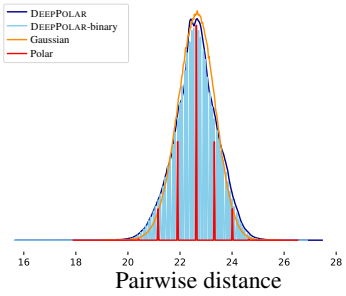
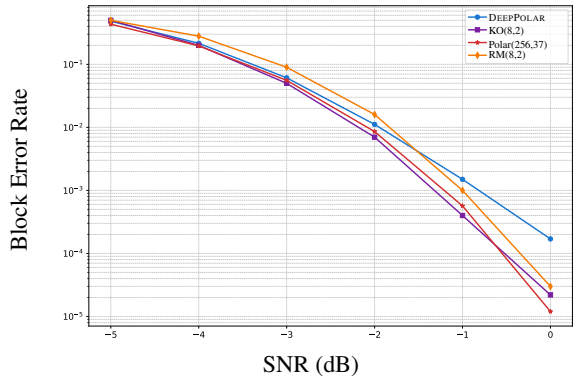


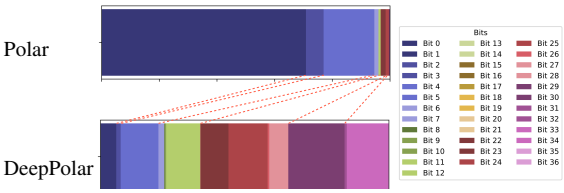
Figure 7. (n=256,k=37,$\ell = 16$): Unlike Polar codes, the distribution of pairwise distribution between codewords of DEEPPOLAR shows a strong resemblance to the Gaussian codebook. DEEPPOLAR-binary code retains a Gaussian-like distance profile. (pdfs and pmfs have been renormalized for better visualization.)

Interpreting the encoder. To interpret the encoder, we examine the distribution of pairwise distances between codewords (Fig. 7). Gaussian codebooks achieve capacity and are optimal asymptotically (Shannon, 1948). Remarkably, the distribution of DEEPPOLAR codewords closely resembles that of the Gaussian codebook. This surprising phenomenon has also been observed in the closely related KO codes and is a testament to the potential of the marriage between efficient algebraic code structures and DL.

Interpreting the decoder. In our study, the training loss used is binary cross entropy (BCE), which serves as a surrogate for the bit error rate (BER). However, it is important to recognize that optimizing for BER does not necessarily translate to improved block error rate (BLER), an important



a) BLER performance (n=256,k=37)



b) Distribution of Errors (n=256,k=37)

Figure 8. a) The block error rate performance of DEEPPOLAR codes is subpar. b) The distribution of the first error occurrence across bit positions reveals that this tradeoff is a consequence of a more uniform error distribution, driven by the use of BCE loss, which is a surrogate for BER.

figure of merit in practical systems. Our observations, as indicated in Fig. 8a), show that DEEPPOLAR underperforms in BLER compared to baseline methods (See App. F for more results). The BLER, defined as $\mathbb{P}[\hat{\boldsymbol{u}} \neq \boldsymbol{u}]$, can be expressed as the cumulative probability of bit errors occurring at each position when no errors were made in the previous ones:

$$\mathbb{P}[\hat{\boldsymbol{u}} \neq \boldsymbol{u}] = \sum_{i=1}^{k} \mathbb{P}[\hat{u}_i \neq u_i | \hat{\boldsymbol{u}}_{1:i-1} = \boldsymbol{u}_{1:i-1}] \qquad (3)$$

An analysis of error distribution (Fig. 8b)) reveals notable differences between SC and DEEPPOLAR-SC decoders despite both employing a sequential decoding approach. Due to the effect of channel polarization, most errors in the polar decoding tend to occur at bit position 0, which predominantly drives the BLER. In contrast, errors in DEEPPOLAR decoding are more evenly distributed across various bit positions. This uniformity is a consequence of the BCE loss prioritizing BER over BLER. This trade-off leads to a marginal decrease in BLER performance. Identifying a stable loss function directly targeting BLER, is an interesting direction for future research.

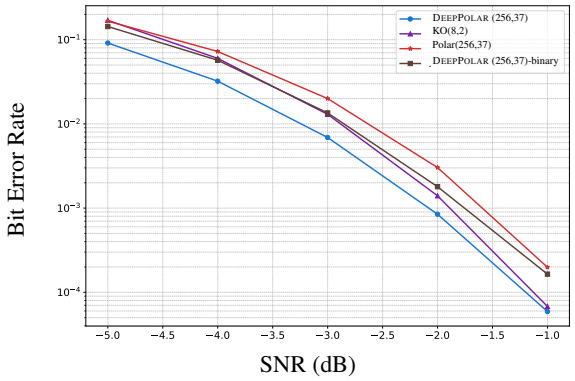


Figure 9. DEEPPOLAR-binary(256,37,ℓ=16) approaches the performance of KO codes

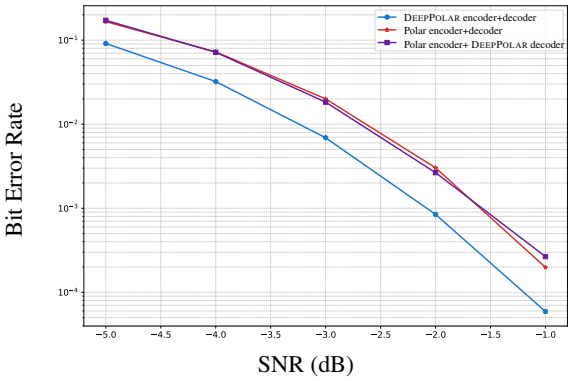


Figure 10. For decoding Polar(256,37) codes, DEEPPOLAR decoder matches the SC decoder.

4.5. Ablation studies

DEEPPOLAR-Binary

Compared to canonical coding schemes, a real-valued coding scheme like DEEPPOLAR offers distinct advantages by integrating modulation and coding schemes. However, practical systems may have hard symbol-level power constraints, necessitating code bits to be integers $x_i \in \{-1, 1\}$. Hence, it is beneficial to have the ability to learn a binary code while maintaining the structure of DEEPPOLAR.

To achieve this, a trained DEEPPOLAR model is fine-tuned with a Straight Through Estimator (STE) (Hubara et al., 2016) and a binarization module, resulting in a binary version, DEEPPOLAR-binary, akin to the methodology in (Jiang et al., 2019b). Fig. 9 highlights that there is a loss incurred with respect to DEEPPOLAR. This underscores the contribution of joint coding and modulation to DEEPPOLAR's performance. Nevertheless, DEEPPOLAR-binary not only surpasses the performance of the traditional Polar code but also closely matches that of KO codes. Additionally, the distance profile of DEEPPOLAR-binary (Fig. 7) indicates it preserves Gaussian codebook-like distance properties, reinforcing the potential of non-linear binary codes derived from polarization-inducing structures. This exploration opens avenues for future research, particularly in interpreting these codes to develop novel, efficient encoders and decoders leveraging non-linear polarization-based methods.

Effect of curriculum. Ensuring reliable performance of neural codes at high SNR levels is a significant challenge, primarily due to the sparse occurrence of error events in this regime. This challenge often leads to a phenomenon known as an *error floor*, where the error rate improvement of a code stagnates beyond a certain SNR (Jamali et al., 2021b). This issue is not exclusive to neural codes, as it's also encountered in classical codes like LDPC and Turbo codes. While training hyperparameters such as SNR scheduling and batch sizes affect high SNR performance - the structure of the code also plays a part. Notably, algebraic codes like polar codes do not suffer from error floors. Driven by this intuition, our curriculum design (Sec. 3.2) reinforces the Plotkin code structure, leading to better generalization. This strategy has been effective, as highlighted in Fig. 6, where implementing a curriculum markedly enhances code performance at higher SNRs.

DeepPolar decoder for Polar codes

DEEPPOLAR codes are a generalization of Polar codes, and inherits its encoding structure (Plotkin tree, frozen set), and the sequential decoding paradigm, and achieves notable gains in BER. To pinpoint the efficacy of the learned code, we fix the Polar encoder and train the DEEPPOLAR decoder to predict the transmitted message. Fig. 10 demonstrates that the DEEPPOLAR decoder matches the performance of SC decoding to decode Polar codes. This is expected, owing to the SC-like parameterization of the DEEPPOLAR decoder. The gains in BER, thus, can largely be attributed to the encoder's non-linear, binary-to-real encoding scheme.

4.6. Computational Complexity

In our study, we introduce a novel non-linear generalization of the Polar code's encoding and decoding structures and achieve substantial improvements in reliability. Another essential aspect of evaluating an algorithm is the *computational complexity*, which has a direct impact on power consumption. Polar codes are favored due to their relatively low-complexity encoding and decoding algorithms. However, in practice, CRC-aided list decoders are used, which substantially increases the decoding complexity. DEEPPOLAR is a non-linear generalization of Polar codes via neural-network-based kernels on the Plotkin tree. The DEEPPOLAR SC-decoder is a kernel-wise sequential decoder inspired by the SC algorithm. For the case of DEEPPOLAR ($n = 256, k = 37, \ell = 16$), the encoder and decoder have

0.1M and 1.6M parameters, respectively, which we did not optimize in this project. Our findings serve as constructive proof that non-linear polar codes with large kernels can outperform the state-of-the-art. Further, we demonstrate the viability of effective binary codes with a large-kernel Polar structure, complete with efficient decoders. This opens avenues for further examination of these codebooks to inform the development of low-complexity solutions. Notably, the complexity of neural codes can be reduced significantly without performance degradation by complexity-aware architecture design (eg, TinyKO (Makkuva et al., 2021)). Our preliminary experiments with DEEPPOLAR-parallel (App. C.2), which reduces the decoder parameter count by $8\times$, have already demonstrated *improved* performance at lower complexity. Further, several techniques are used in practice, like distillation, quantization, and pruning, among others, to reduce the computational overhead of a neural algorithm. This is important and interesting future work beyond the scope of this paper.

5. Related work

The application of machine learning to channel coding has been an active area of research in recent years. The majority of existing works focus on decoding existing codes, aiming to achieve better reliability and robustness, and in some cases, lower decoding complexity (Kim et al., 2018b; Nachmani et al., 2016; Dörner et al., 2017; Vasić et al., 2018; Nachmani & Wolf, 2019; Jiang et al., 2019a; Chen & Ye, 2021; Jamali et al., 2021a; Choukroun & Wolf, 2022a;b; Hebbar et al., 2022; Aharoni et al., 2023). In the context of polar codes, several neural decoders have been proposed (Cammerer et al., 2017b;a; Doan et al., 2018; Hebbar et al., 2023). In another line of work, (Ebada et al., 2019; Liao et al., 2021; Li et al., 2021; Miloslavskaya et al., 2023; Ankireddy et al., 2024a) use deep learning to identify optimal polar code frozen positions without altering the design of encoder and decoder.

In contrast, jointly learning both encoders and decoders is a more challenging problem, and very few works in the literature exist. A common theme is the incorporation of principled coding-theoretic encoding and decoding structures. TurboAE (Jiang et al., 2019b), and follow-up works (Jiang et al., 2020c;a; Chahine et al., 2021; Saber et al., 2022; Chahine et al., 2022a; Wang et al., 2023), use sequential encoding and decoding along with interleaving of input bits, inspired by Turbo codes (Berrou et al., 1993). KO codes (Makkuva et al., 2021) generalize Reed-Muller encoding and Dumer decoding by replacing selected components in the Plotkin tree with neural networks. ProductAE (Jamali et al., 2021b) generalizes two-dimensional product codes to scale neural codes to larger block lengths. In a similar vein, our work generalizes the coding structures of large-kernel Polar codes by using non-linear kernels parameterized by NNs.

Deep learning-based schemes for channels with feedback is another active area of research (Kim et al., 2018a; Safavi et al., 2021; Chahine et al., 2022b; Ozfatura et al., 2022; 2023b;a; Kim et al., 2023; Ankireddy et al., 2024b) Furthermore, breaking the conventional wisdom that neural codes are hard to interpret, (Devroye et al., 2022) derives a closed-form approximation of binarized TurboAE codes via mixed integer linear programming and other techniques. The task of analytically approximating and understanding binarized DEEPPOLAR codes remains a promising subject for future research.

6. Conclusion

In this work, we introduce DEEPPOLAR codes, a new class of non-linear generalizations of large-kernel Polar codes. DEEPPOLAR codes generalize to various rates and block-lengths, and outperform the current state-of-the-art neural codes in BER. The neural architecture mirrors the Polar encoding and decoding structures, which along with a curriculum training methodology is key to improve generalization to unseen messages.

Acknowledgement

This research is supported in part by NSF CCF 2312753, ONR N00014-21-1-2388, NSF CNS 2112471, ARO W911NF2310062, ONR N00014-21-1-2379, NSF CNS-2008824, a gift from Intel, and Samsung Research America through 6G@UT center within the Wireless Networking and Communications Group (WNCG) at the University of Texas at Austin. We thank P. Trifonov for his insightful comments.

Impact Statement

This paper presents work whose goal is to advance the field of channel coding using machine learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

3GPP. Multiplexing and channel coding. Technical Specification 38.212, 3rd Generation Partnership Project (3GPP), 2018. Release 15.

Abbasi, F. and Viterbo, E. Large kernel polar codes with efficient window decoding. *IEEE Transactions On Vehicular Technology*, 69(11):14031–14036, 2020.

Aharoni, Z., Huleihel, B., Pfister, H. D., and Permuter, H. H. Data-driven polar codes for unknown channels with and without memory. In *2023 IEEE International Symposium on Information Theory (ISIT)*, pp. 1890–1895. IEEE, 2023.

Ankireddy, S. K., Hebbar, S. A., Wan, H., Cho, J., and Zhang, C. Nested construction of polar codes via transformers. *arXiv preprint arXiv:2401.17188*, 2024a.

Ankireddy, S. K., Narayanan, K., and Kim, H. Lightcode: Light analytical and neural codes for channels with feedback. *arXiv preprint arXiv:2403.10751*, 2024b.

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.

Arıkan, E. From sequential decoding to channel polarization and back again. *arXiv preprint arXiv:1908.09594*, 2019.

Berrou, C., Glavieux, A., and Thitimajshima, P. Near shannon limit error-correcting coding and decoding: Turbo-codes. 1. In *Proceedings of ICC'93-IEEE International Conference on Communications*, volume 2, pp. 1064–1070. IEEE, 1993.

Bioglio, V., Condo, C., and Land, I. Design of polar codes in 5g new radio. *IEEE Communications Surveys & Tutorials*, 23(1):29–40, 2020.

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, 2017a.

Cammerer, S., Leible, B., Stahl, M., Hoydis, J., and ten Brink, S. Combining belief propagation and successive cancellation list decoding of polar codes on a gpu platform. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 3664–3668. IEEE, 2017b.

Chahine, K., Ye, N., and Kim, H. Deepic: Coding for interference channels via deep learning. *arXiv preprint arXiv:2108.06028*, 2021.

Chahine, K., Jiang, Y., Nuti, P., Kim, H., and Cho, J. Turbo autoencoder with a trainable interleaver. In *ICC 2022-IEEE International Conference on Communications*, pp. 3886–3891. IEEE, 2022a.

Chahine, K., Mishra, R., and Kim, H. Inventing codes for channels with active feedback via deep learning. *IEEE Journal on Selected Areas in Information Theory*, 3(3): 574–586, 2022b.

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

Choukroun, Y. and Wolf, L. Denoising diffusion error correction codes. *arXiv preprint arXiv:2209.13533*, 2022a.

Choukroun, Y. and Wolf, L. Error correction code transformer. *arXiv preprint arXiv:2203.14966*, 2022b.

Devroye, N., Mohammadi, N., Mulgund, A., Naik, H., Shekhar, R., Turán, G., Wei, Y., and Žefran, M. Interpreting deep-learned error-correcting codes. In *2022 IEEE International Symposium on Information Theory (ISIT)*, pp. 2457–2462, 2022. doi: 10.1109/ISIT50566.2022.9834599.

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.

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.

Fazeli, A. and Vardy, A. On the scaling exponent of binary polarization kernels. In *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 797–804. IEEE, 2014.

Fazeli, A., Hassani, H., Mondelli, M., and Vardy, A. Binary linear codes with optimal scaling: Polar codes with large kernels. *IEEE Transactions on Information Theory*, 67 (9):5693–5710, 2020.

Gupta, B., Yao, H., Fazeli, A., and Vardy, A. Polar list decoding for large polarization kernels. In *2021 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6. IEEE, 2021.

Hebbar, S. A., Mishra, R. K., Ankireddy, S. K., Makkuva, A. V., Kim, H., and Viswanath, P. Tinyturbo: Efficient

turbo decoders on edge. In *2022 IEEE International Symposium on Information Theory (ISIT)*, pp. 2797–2802. IEEE, 2022.

Hebbar, S. A., Nadkarni, V. V., Makkuva, A. V., Bhat, S., Oh, S., and Viswanath, P. Crisp: Curriculum based sequential neural decoders for polar code family. In *International Conference on Machine Learning*, pp. 12823–12845. PMLR, 2023.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks. *Advances in neural information processing systems*, 29, 2016.

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. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pp. 1088–1093. IEEE, 2021a.

Jamali, M. V., Saber, H., Hatami, H., and Bae, J. H. Productae: Towards training larger channel codes based on neural product codes. *arXiv preprint arXiv:2110.04466*, 2021b.

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. Joint channel coding and modulation via deep learning. In *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–5. IEEE, 2020a.

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*, 2020b.

Jiang, Y., Kim, H., Asnani, H., Oh, S., Kannan, S., and Viswanath, P. Feedback turbo autoencoder. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8559–8563. IEEE, 2020c.

Kim, H., Jiang, Y., Kannan, S., Oh, S., and Viswanath, P. Deepcode: Feedback codes via deep learning. *Advances in neural information processing systems*, 31, 2018a.

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

Kim, J., Kim, T., Love, D., and Brinton, C. Robust nonlinear feedback coding via power-constrained deep learning. *arXiv preprint arXiv:2304.13178*, 2023.

Korada, S. B., Şaşoğlu, E., and Urbanke, R. Polar codes: Characterization of exponent, bounds, and constructions. *IEEE Transactions on Information Theory*, 56(12):6253–6264, 2010.

Li, Y., Chen, Z., Liu, G., Wu, Y.-C., and Wong, K.-K. Learning to construct nested polar codes: An attention-based set-to-element model. *IEEE Communications Letters*, 25 (12):3898–3902, 2021.

Liao, Y., Hashemi, S. A., Cioffi, J. M., and Goldsmith, A. Construction of polar codes with reinforcement learning. *IEEE Transactions on Communications*, 70(1):185–198, 2021.

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

Miloslavskaya, V., Li, Y., and Vucetic, B. Design of compactly specified polar codes with dynamic frozen bits based on reinforcement learning. *IEEE Transactions on Communications*, 2023.

Nachmani, E. and Wolf, L. Hyper-graph-network decoders for block codes. *Advances in Neural Information Processing Systems*, 32:2329–2339, 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.

Niu, K. and Chen, K. Crc-aided decoding of polar codes. *IEEE communications letters*, 16(10):1668–1671, 2012.

Ozfatura, E., Shao, Y., Perotti, A. G., Popović, B. M., and Gündüz, D. All you need is feedback: Communication with block attention feedback codes. *IEEE Journal on Selected Areas in Information Theory*, 3(3):587–602, 2022.

Ozfatura, E., Bian, C., and Gündüz, D. Do not interfere but cooperate: A fully learnable code design for multi-access channels with feedback. In *2023 12th International Symposium on Topics in Coding (ISTC)*, pp. 1–5. IEEE, 2023a.

Ozfatura, E., Shao, Y., Ghazanfari, A., Perotti, A., Popovic, B., and Gündüz, D. Feedback is good, active feedback is better: Block attention active feedback codes. In *ICC 2023-IEEE International Conference on Communications*, pp. 6652–6657. IEEE, 2023b.

Saber, H., Hatami, H., and Bae, J. H. List autoencoder: Towards deep learning based reliable transmission over noisy channels. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pp. 1454–1459. IEEE, 2022.

Safavi, A. R., Perotti, A. G., Popovic, B. M., Mashhadi, M. B., and Gunduz, D. Deep extended feedback codes. *arXiv preprint arXiv:2105.01365*, 2021.

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

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.

Trifonov, P. V. Design and decoding of polar codes with large kernels: a survey. *Problems of Information Transmission*, 59(1):22–40, 2023.

Trofimiuk, G. and Trifonov, P. Window processing of binary polarization kernels. *IEEE Transactions on Communications*, 69(7):4294–4305, 2021.

Vasić, B., Xiao, X., and Lin, S. Learning to decode ldpc codes with finite-alphabet message passing. In *2018 Information Theory and Applications Workshop (ITA)*, pp. 1–9. IEEE, 2018.

Wang, L., Saber, H., Hatami, H., Jamali, M. V., and Bae, J. H. Rate-matched turbo autoencoder: A deep learning based multi-rate channel autoencoder. In *ICC 2023-IEEE International Conference on Communications*, pp. 6355–6360. IEEE, 2023.

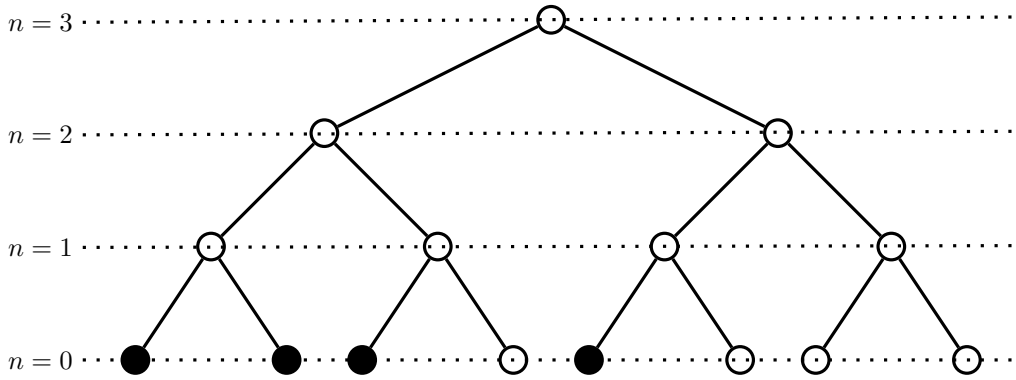


Figure 11. Binary tree structure of (8,4) polar code with stages indicated. The frozen bits (in black) are set to 0. Operations at each node are detailed in Fig. A.

A. Successive Cancellation decoder

Here we look at the Successive Cancellation Decoding algorithm, provided in (Arikan, 2009). As the name suggests, the SC decoding algorithm decodes the bits sequentially, starting with u_0. A frozen bit node is always decoded as 0. And while decoding the u_i, the available bits u_0 to u_0^{i-1} which are represented by the vector u_{i-1} are used according to decode u_i according to the following rule:

$$\hat{u_i} = \begin{cases} 0, & \text{if } i \in \mathcal{I} \text{ and } Pr[y, \hat{u_0}^{i-1}|u_i = 0] \geq Pr[y, u_0^{i-1}|\hat{u_i} = 1] \\ 1, & \text{if } i \in \mathcal{I} \text{ and } Pr[y, \hat{u_0}^{i-1}|u_i = 0] \leq Pr[y, u_0^{i-1}|\hat{u_i} = 1] \\ 0, & \text{if } i \in \mathcal{F} \end{cases} \tag{4}$$

where A is the set of information positions.

The probability calculations are computationally easier and less prone to round-off errors in the log domain. Hence, we consider LLRs instead of probabilities to avoid numerical overflows. LLR for t^{th} bit is defined as:

$$L^{(i)}(y, \hat{u_0}^{i-1}) = log \frac{Pr[y, \hat{u_0}^{i-1}|u_i = 0]}{Pr[y, u_0^{i-1}|\hat{u_i} = 1]}$$

Hence, the decision rule changes as follows:

$$\hat{u_i} = \begin{cases} 0, & \text{if } i \in \mathcal{I} \text{ and } L^{(i)}(y, \hat{u_0}^{i-1}) \geq 0 \\ 1, & \text{if } i \in \mathcal{I} \text{ and } L^{(i)}(y, \hat{u_0}^{i-1}) \leq 0 \\ 0, & \text{if } i \in \mathcal{F} \end{cases} \tag{5}$$

The binary tree structure of a polar code can be exploited to simplify the successive cancellation decoding. The binary tree has $n = \log_2 N + 1$ stages with numbering from $s = 0, \ldots, n$. Each stage s contains 2^s nodes with each node corresponding to 2^{n-s} bits.

In order, traversal of the tree is done to perform the successive cancellation decoding. At each node, messages are passed as shown:

Each node passes LLR corresponding LLR values, namely α, to the child nodes and sends the estimated hard bits at the sage, namely β, o the parent node. The left and right messages, α_i^l and α_i^r are calculated as:

$$\alpha_i^l = \ln\left(\frac{1 + e^{\alpha_i + \alpha_{i+2^{n-s-1}}}}{e^{\alpha_i} + e^{\alpha_{i+2^{n-s-1}}}}\right) \tag{6}$$

$$\alpha_i^r = \alpha_{i+2^{n-s-1}} + (1 - 2\beta_i^l)\alpha_i \tag{7}$$

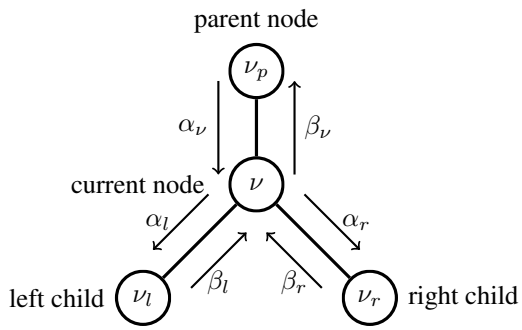


Figure 12. SC decoding update rules for each node

We define two functions to perform these operations, namely f and g, defined as:

$$\begin{aligned} f(\alpha_1, \alpha_2) &= \ln\left(\frac{1 + e^{\alpha_1+\alpha_2}}{e^{\alpha_1} + e^{\alpha_2}}\right) \\ g(\beta, \alpha_1, \alpha_2) &= (1 - 2\beta)\alpha_1 + \alpha_2 \end{aligned} \tag{8}$$

But the f function is computationally expensive, and hence, we approximate it to a hardware-friendly version using min-sum approximation as follows:

$$f_{minsum}(\alpha_1, \alpha_2) = \textit{sign}(\alpha_1)\textit{sign}(\alpha_2)\textit{min}(|\alpha_1|, |\alpha_2|) \tag{9}$$

where *sign* gives the sign of input and *min* gives the minimum of the two inputs.

The algorithm starts from the root node of the tree, which is level $n + 1$, and traverses to the leaf node, which is level 0. For each node, the following set of operations occurs.

1. If the current node has a left child that was not visited, calculate α_l and move to the left child.
2. If the current node has a right child that was not visited, calculate α_r and move to the right child.
3. If both the messages from child nodes are available, calculate β and move to the parent node.

Once the leaf node is reached, decisions are made based on the sign of the corresponding LLR using the binary quantizer function h as:

$$\beta_\nu = h(\alpha_n u) \tag{10}$$

where h is defined as:

$$h(\alpha) = \begin{cases} 0, & \text{if } \alpha \geq 0 \\ 1, & \text{else} \end{cases}$$

B. Polar codes

B.1. 5G Polar codes

Polar codes are conceptualized by channel polarization, i.e., the recursive application of the kernel results in bit-channels that are either noiseless or very noisy. However, at the practical blocklengths that we consider in this paper, there exist bit channels that are partially polarized. Consequently, the selection of frozen positions affects the reliability of polar codes. Although DEEPPOLAR does not exhibit a clear polarization effect—due to its operation in the real field and its non-recursive kernel application—the choice of frozen positions still significantly impacts the effectiveness of the learned code.

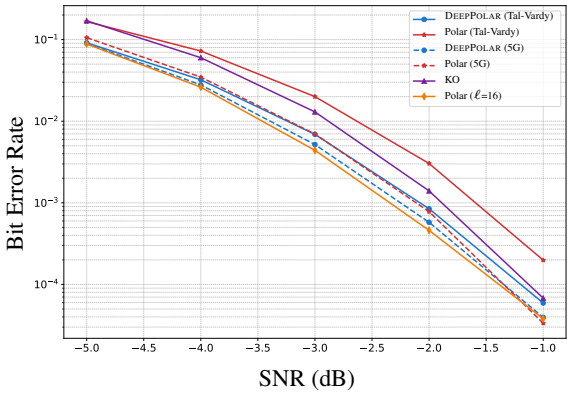


Figure 13. n=256,k=37 : The choice of frozen positions impacts the performance of DEEPPOLAR. While using 5G frozen positions improves DEEPPOLAR, its gains over the 5G-Polar code are modest. DEEPPOLAR is a preliminary effort towards an automated search for large-kernel codes; it matches the best performing Polar code with $\ell = 16$ (Trofimiuk & Trifonov, 2021)

In the paper, we consider polar codes constructed via the Tal-Vardy method as our baseline. This choice is justified by the fact that DEEPPOLAR utilizes the Tal-Vardy rate profile, ensuring a fair comparison. However, we realize that the universal reliability sequence specified in the 5G standards can be used to find a better frozen set. Fig. 13 depicts that DEEPPOLAR trained using the 5G frozen positions shows non-trivial improvement. However, the performance gains compared to the 5G-Polar code are modest. Exploring joint optimization of rate-matching strategies (such as freezing or precoding) along with training DEEPPOLAR presents an intriguing avenue for future research.

B.2. Large kernel Polar codes

The SC decoding operation for larger kernel Polar codes ($\ell > 2$) is similar to the original polar codes. Iteratively, the decoding equation can be expressed as

$$W_m^{(i)}((u_0^{(i-1)}, u_i)|y_0^{(n-1)}) = \frac{W_m^{(i)}(y_0^{n-1}, u_0^{i-1}|u_i)}{2W(y_0^{n-1})},$$

which is the probability for path u_0^i given channel output y_0^{n-1}.

While these operations are similar to conventional polar codes, the complexity of iteratively computing the probabilities increases exponentially with respect to ℓ as $O(2^\ell)$, making it prohibitively expensive for very large kernels. However, recent work has shown methods towards efficient decoding of a class of large-kernel polar codes (Abbasi & Viterbo, 2020; Gupta et al., 2021; Trofimiuk & Trifonov, 2021).

DEEPPOLAR generalizes the conventional polar codes, and makes two notable modifications. First, it expands the kernel size to $\sqrt{n}$. Second, it introduces neural networks to parameterize the encoding and decoding functions for each kernel, which are then learned through training. The exhaustive search across large kernel spaces to improve upon Arikan's original construction is infeasible. Nonetheless, recent advancements have been made by limiting this space and employing more efficient decoding methods (Trifonov, 2023). DEEPPOLAR represents an initial attempt to automate the search for effective large-kernel polar codes.

Here, we consider an optimized construction of a kernel of size $\ell = 16$ (Trofimiuk & Trifonov, 2021), with SC decoding and compare it with DEEPPOLAR code of $\ell = 16$. Fig. 13 shows that DEEPPOLAR marginally falls short of this optimized construction. An exciting future direction would be to directly augment these codes using our methodology, akin to how KO codes augmented the Reed Muller code structure.

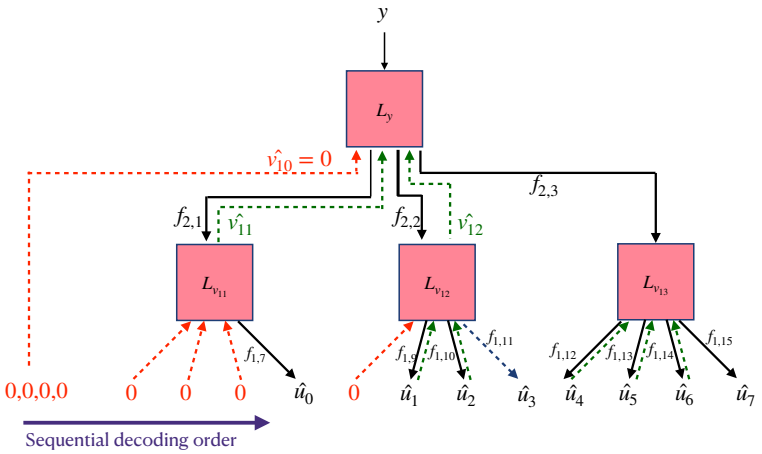


Figure 14. DEEPPOLAR (n=16,k=8,ℓ=4) : DEEPPOLAR-SC decoder decodes each bit sequentially - each edge operation in the decoding tree is parameterized by a neural network $f_{d,b}$

C. DEEPPOLAR Decoders

C.1. DEEPPOLAR-SC Decoder

Decoding of DEEPPOLAR for large kernels is depicted in Fig. 14 for DEEPPOLAR (16,8,$\ell = 4$). The frozen positions are not decoded and are directly set to 0, including the full of the first kernel. Beginning with the first information position, $\hat{u}_0$ is estimated based on the received values $\boldsymbol{y}$ using sub-kernel $f_{1,7}$. Now, the decoding of the two kernels is complete and sent to the parent node to begin the decoding of kernel 3. Using the received values $\boldsymbol{y}$ and $\hat{u}_0$, sub-kernel $f_{1,9}$ estimates the next non-frozen position $\hat{u}_1$. Next, using $\boldsymbol{y}$ and $\{\hat{u}_0, \hat{u}_1\}$, sub-kernel $f_{1,10}$ decodes the next non-frozen position $\hat{u}_2$. This process is continued until kernel-3 is decoded completely, after which the decoding of kernel-4 starts in the same manner and continues until all bits are estimated.

To summarise, successive cancellation decoding of DEEPPOLAR involves sequentially decoding one kernel of size ℓ at a time. Decoding the kernel is again performed in a successive fashion, where each bit is decoded by the corresponding component decoder or sub-kernel.

C.2. DEEPPOLAR-parallel decoder

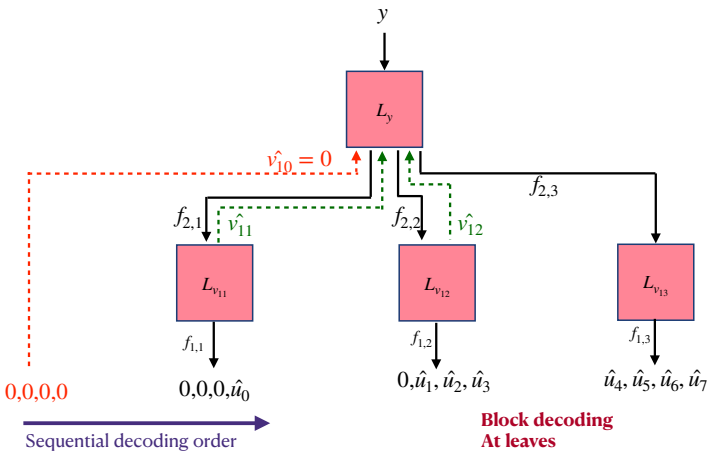


Figure 15. DEEPPOLAR (n=16,k=8,ℓ=4) : DEEPPOLAR-parallel decoder retains the same decoding structure as DEEPPOLAR-SC, but uses a one-shot parallel decoder at the leaves.

A major drawback of SC decoding and its variants like SCL decoders is the latency overhead due to sequential decoding. This drawback is inherently present in the DEEPPOLAR-SC decoder. One method to break this latency barrier, whilst reducing the computational complexity of the decoder is to replace the ℓ SC-style NN decoders for a kernel at the lowest depth, by a single 3-layer FCNN, with hidden size 64, that decodes ℓ bits in one shot, as depicted in Fig. 15. We refer to this as DEEPPOLAR-parallel. This resulted in an immediate reduction of 8x in parameter count compared to DEEPPOLAR-SC while achieving the same reliability, as highlighted in Fig. 16.

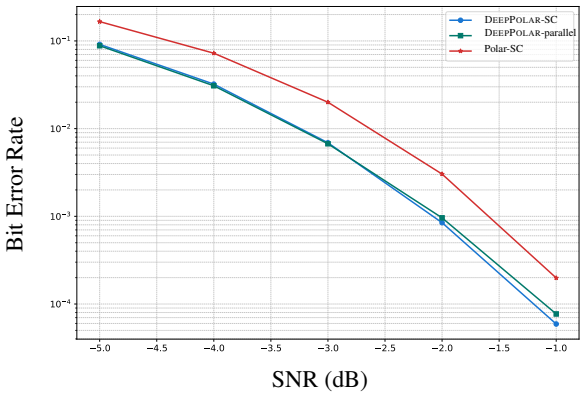


Figure 16. n=256,k=37 : DEEPPOLAR-parallel, a lower complexity decoder constructed by replacing the SC-style decoding at the leaves by a bit-MAP-style decoding, achieves better reliability than DEEPPOLAR-SC, whilst suffering lower latency

These preliminary results hint at the existence of low-complexity architectures that provide comparable performance at a much lower complexity. This is a promising direction of future exploration.

D. Robustness to non-AWGN channels

Traditionally, the canonical setting for the design and evaluation of codes has predominantly focused on AWGN channels due to the relative simplicity of closed-form performance analysis. However, practical communication channels often deviate significantly from the AWGN model. Ideally, both the code and decoder should be robust to various channels or adaptable to the encountered channel conditions. In this section, we demonstrate that DEEPPOLAR codes, which are trained on AWGN channels, exhibit *robustness* when tested on non-AWGN channels *without retraining*. Further, in cases where the target channel model significantly diverges from AWGN noise, data-driven codes like DEEPPOLAR can achieve substantial performance improvements through fine-tuning with a small number of steps on the target channel.

First, we consider Rayleigh fast-fading channel with AWGN noise. Rayleigh fading is considered a suitable model for signal propagation in tropospheric and ionospheric environments, as well as for the effect of heavily built-up urban areas on radio signals. Mathematically, the channel model can be approximated as

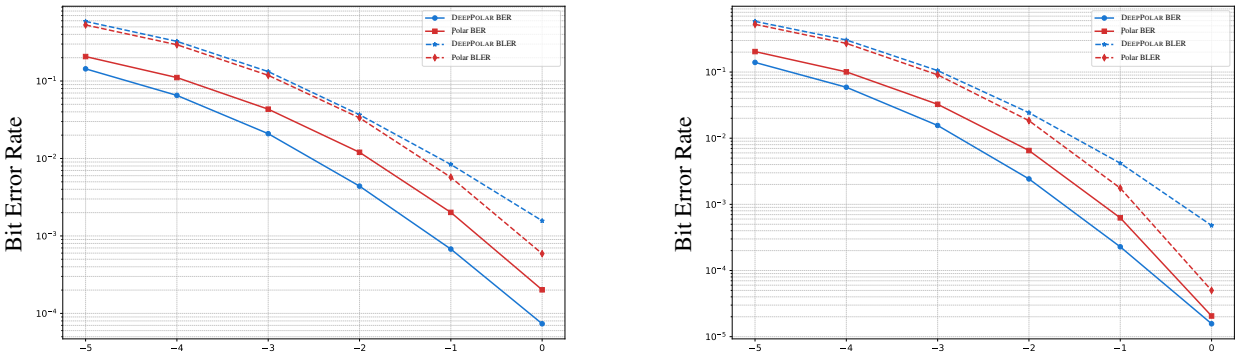
$$y_i = h_i x_i + z_i,$$

where $h \sim \mathcal{N}(0,1)$ is the fading coefficient and $z_i \sim \mathcal{N}(0,\sigma^2)$ is the Gaussian noise. Note that the fading coefficient changes for every symbol in the codeword, making it a much worse channel than the AWGN channel. As seen from Fig. 17a), the DEEPPOLAR code demonstrates a gain of up to 0.3 dB compared to Polar$(256, 37)$, demonstrating the robustness of the learned code.

Next, we consider a bursty noise channel. Bursty noise, also known as popcorn noise, can be described mathematically as

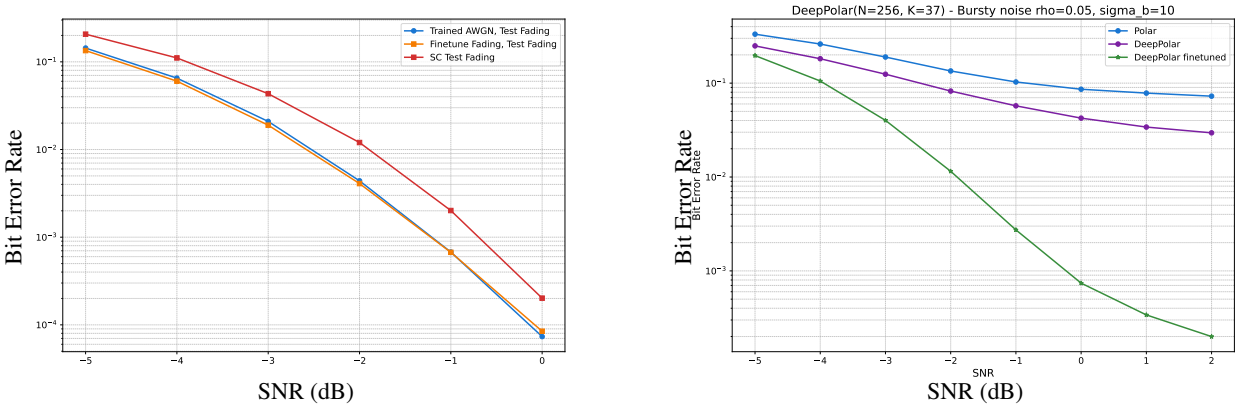
$$y_i = x_i + z_i + w_i,$$

where $z_i \sim \mathcal{N}(0,\sigma^2)$ is the Gaussian noise and $w_i \sim \mathcal{N}(0,\sigma_b^2)$ with probability ρ and $w_i = 0$ with probability $1-\rho$ is the bursty noise. For our experiment, we choose $\rho = 0.1$ and $\sigma_b = \sqrt{2}\sigma$. As seen from Fig. 17b), the DEEPPOLAR code demonstrates a gain of up to 0.25 dB compared to Polar$(256, 37)$ in the presence of bursty noise.



a)$(n = 256, k = 37)$, Rayleigh fast fading channel b)$(n = 256, k = 37)$, Bursty noise channel

Figure 17. Robustness to channel variations: DEEPPOLAR $(256, 37, \ell = 16)$ trained on AWGN channel maintains the gains over Polar$(256, 37)$ when tested on Rayliegh fast fading and bursty noise channels.



a)$(n = 256, k = 37)$, Rayleigh fast fading channel b)$(n = 256, k = 37)$, Bursty noise channel

Figure 18. Adaptivity: Performance of DEEPPOLAR $(256, 37, \ell = 16)$ can be improved by adapting it to target channels. On channels such as Rayleigh fast fading, we do not see extra gains over the base decoder trained on AWGN. However, on large bursty noise we see a pronounced improvement

Under larger degradation, for instance a bursty channel with $\rho = 0.05$ and burst power σ_b= 10, DeepPolar maintains its robustness. Still, the performance can be greatly enhanced by fine-tuning the decoder specifically for the target channel model, as demonstrated in Fig. 18b). This adaptivity to unseen channel models is a key advantage of neural codes and decoders, such as DeepPolar, compared to classical codes.

E. Experimental details

The complete source code is provided at: https://www.github.com/hebbarashwin/deeppolar

E.1. Training algorithm

We follow a training strategy similar to KO codes (Makkuva et al., 2021), where the encoder and decoder networks are trained in an alternating optimization, described below.

Algorithm 1 Training algorithm for DEEPPOLAR (256,37,ℓ=16)

Initialize encoder and decoder parameters
for E epochs **do**
  for T_{dec} steps **do**
    Generate B random message vectors $\boldsymbol{u}$
    Simulate AWGN channel with SNR_{dec}
    Freeze the encoder g and update the decoder network f to minimize the BCE loss $L(g, f)$ using Adam and learning rate lr_{dec}.
  end for
  for T_{enc} steps **do**
    Generate B random message vectors $\boldsymbol{u}$
    Simulate AWGN channel with SNR_{enc}
    Freeze the decoder f and update the encoder network g to minimize the BCE loss $L(g, f)$ using Adam and learning rate lr_{enc}.
  end for
end for

E.2. Training SNR and number of steps

Choosing the right SNR is critical to avoid local optima during training. Choosing a very low SNR will result in the noise dominating the training data, and as a result, the training may not converge. On the contrary, choosing a very high SNR will not provide sufficient error events for the model to learn from. Through empirical testing, we find that a training SNR of 0 dB for the encoder and -2 dB for the decoder works best. Intuitively, this can be interpreted as learning a good decoder being much harder than learning a good encoder, as the decoder has to work with noisy data, whereas the encoder always takes clean input.

Continuing on this theme, we also observe that training the encoder for fewer iterations is sufficient compared to the decoder. In our experiments, we train the encoder 10x fewer steps than the decoder.

High SNR performance. Obtaining gains over classical codes in the high SNR regime is challenging - this stems from several factors : Firstly, the error events become exceedingly sparse in the high SNR regime, leading to noisy gradient estimates and unstable training. Secondly, in the high SNR regime, the error rate is primarily governed by the code's minimum distance. While DeepPolar exhibits remarkable distance properties (closely resembling the Gaussian codebook - Fig. 7), optimizing the minimum distance is challenging given the exponentially large space of codewords.

Nevertheless, high SNR performance can be boosted by finetuning the trained models at substantially high batch sizes. For instance, we finetune the model using a batch size of 200K (a 10x increase from the training phase), at -1dB using a learning rate of 10^{-5}.

E.3. Large batch size

Having a large batch size during training is desirable for stable training. Larger batch sizes will capture the statistics of the underlying distribution more accurately, improving the accuracy of normalization using the mean and variance of the batch. Moreover, a larger batch size also reduces the noise in the gradients computed. In our experiments, we considered a batch size of up to 20000 and further used gradient accumulation techniques to simulate an even larger batch size.

E.4. Hyperparameter choices for DEEPPOLAR (256,37)

The hyperparamters used for training DEEPPOLAR (256,37,$\ell = 16$) are listed in Table 1.

E.5. Architecture for DEEPPOLAR (256,37)

E.5.1. ENCODER

The encoder is a collection of kernels of size $\ell = 16$, each of which is modeled by a neural network g. The encoder kernel g is responsible for encoding ℓ inputs. The architecture for g is given as follows:

Hyperparameter	Value
Batch size (B)	20,000
Encoder training SNR	0 dB
Decoder training SNR	-2 dB
Total epochs (T)	2,000
Encoder steps per epoch (T_{enc})	20
Decoder steps per epoch (T_{dec})	200
Encoder learning rate (l_{enc})	10^{-4}
Decoder learning rate (l_{dec})	10^{-4}
Optimizer	Adam

Table 1. Hyperparameters used in model training for DEEPPOLAR (256,37,$\ell = 16$)

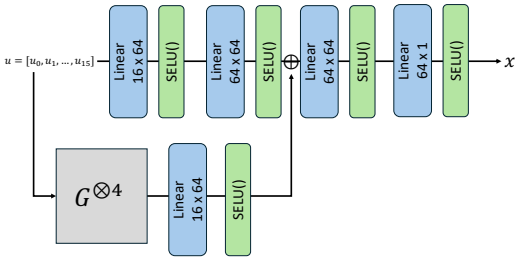


Figure 19. Each kernel g at the encoder is parameterized by a fully connected network of 3 hidden layers and a hidden dimension of 64. The features corresponding to the Plotkin transform of size ℓ is provided via a skip connection

A crucial design choice is to make the polar-encoded features available to the encoding network via a skip connection. DEEPPOLAR codes rely on the encoding and decoding structures of Polar codes, and the features corresponding to polar codes are indeed informative. Further, since it is harder for NNs to learn multiplicative policies, this side information proves to be useful in the learning process.

However, it is noteworthy that the learned DEEPPOLAR codes, as well as DEEPPOLAR-binary codes do not resemble polar codes, both in the mapping as well as the code distance spectrum.

E.5.2. DECODER

The encoder is a collection of kernels of size $\ell = 16$, each of which is modeled by a neural network f. The decoder kernel f contains $\ell = 16$ sub-networks f_i, each responsible for decoding the kernel's i^{th} position. The architecture for f_i is given as follows:

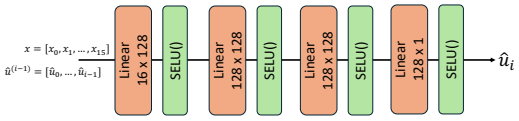


Figure 20. Each kernel f_i at the decoder is parameterized by a fully connected network of 3 hidden layers and a hidden dimension of 128.

F. Additional results

F.1. Block Error Rate

The figures of merit for any channel code are the Bit Error Rate (BER), and Block Error Rate (BLER). Since these metrics are non-differentiable, we need to find a differentiable surrogate. While the binary cross-entropy loss (used in the paper) acts as a stable surrogate loss function to optimize the BER, identifying such a surrogate loss for the BLER is an unsolved problem. (Often, optimizing for BER results in good BLER (eg - Fig. 21b). However, this is not guaranteed).

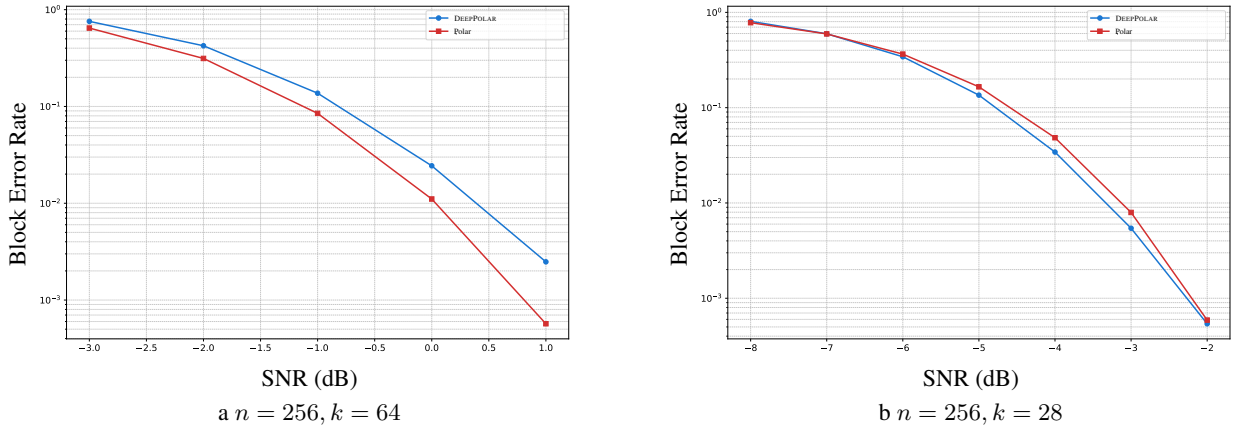


Figure 21. (a) DEEPPOLAR achieves sub-optimal BLER compared to Polar codes: this is an artifact of the BCE training objective, which is a surrogate for the BER

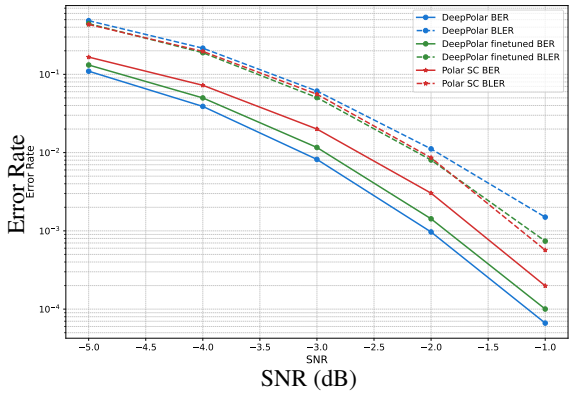


Figure 22. Finetuning a trained DEEPPOLAR model on a BLER surrogate objective trades off BER gains for BLER.

As preliminary work, we consider several potential loss functions to optimize the BLER, including $L_1 = 1 - \Pi_{i=1}^{n} \sigma(u_k L_k)$, where L_k is the logit corresponding to the kth bit, and $L_2 = \text{LogSumExp}(x_1^n)$, where $x_k = \text{BCE}_k$.

However, both these losses lead to unstable training. For example, the product loss L_1 tends to show vanishing gradients. Nevertheless, fine-tuning a model initially trained on BCE loss to optimize the BER, on BLER loss L_1, yields improved BLER performance, demonstrated in Fig. 22 This strategy, however, introduces a tradeoff in BER gains - since the BER and BLER objectives are not fully aligned. With the right training objectives, neural codes provide flexibility in optimizing the figures of merit that we are interested in.

F.2. Model capacity ablation

DEEPPOLAR-SC consists of a total of 100K+1.6M parameters; each kernel is an NN of depth 3, with hidden sizes of 64 and 128, respectively, at the encoder and decoder. We use the notation $(e = 64, d = 128)$ to represent this model. Generally, overparameterized models are easier to train. Since model size is an important practical consideration, we study the effect of model size on reliability. We consider the encoder and decoder networks with kernels with hidden sizes e and d respectively with depth 3. We train the networks via kernel pretraining and curriculum learning, as outlined in Sec. 3.2. As shown in Table 2, Fig. 23, there is a noticeable degradation in performance when small models, i.e., $(e = 8, d = 8)$, $(e = 16, e = 16)$, are used to parameterize each kernel. However, $(e = 16, e = 32)$, and specifically $(e = 32, h = 32)$ with 160K parameters

Enc hidden size	# Parameters in Enc	Dec hidden size	# Parameters in Dec	BER at SNR -1.0 dB
8	5K	8	16K	9e-3
16	12K	16	44K	7.4e-4
16	12K	32	133K	2.1e-4
32	33K	32	133K	8.3e-5
64	100K	128	1.6M	5.9e-5

Table 2. Performance comparison as a function of the number of parameters in the encoder and decoder.

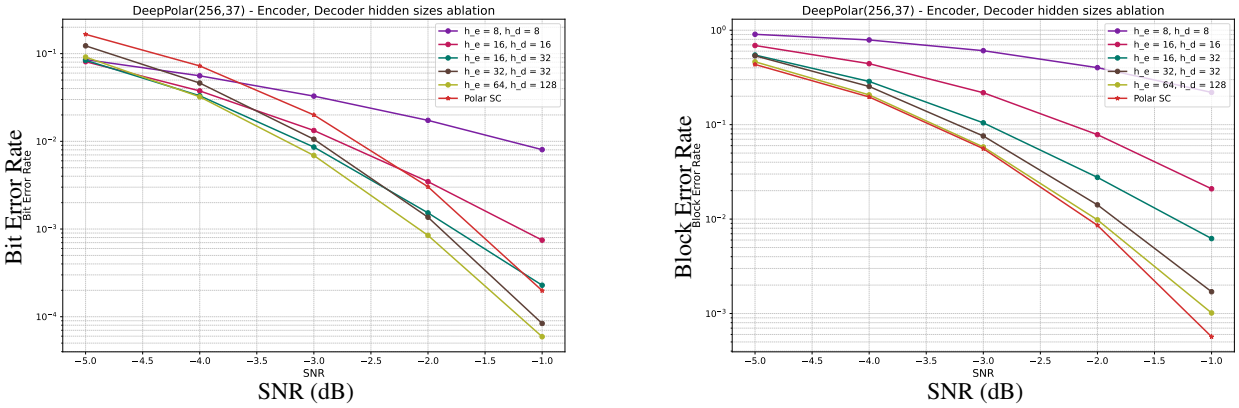


Figure 23. Performance as a function of model size with pretraining and curriculum training: DEEPPOLAR $(256, 37, \ell = 16)$

(10x reduction), outperforms SC in terms of BER and approaches the performance of $(e = 64, d = 128)$. These observations can be attributed to the dimension of the input, which is the kernel size 16.

F.3. Fully connected neural code - lacks generalization.

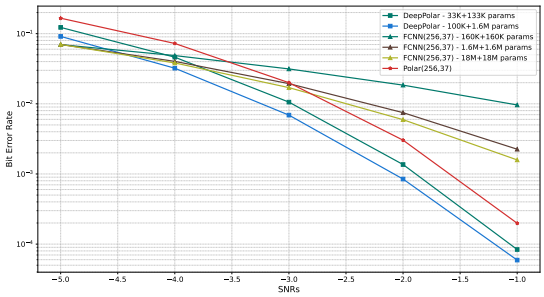


Figure 24. Codes learned via Fully connected NNs fail to generalize to unseen messages due to the absence of strong inductive biases.

While it might seem intuitive that non-linear codes would give better results compared to linear codes, the construction of good non-linear codes is highly non-trivial. To illustrate the difficulty, we train a dense, fully connected encoder-decoder pair using the same training methodology as DEEPPOLAR (Sec. E). As shown in Fig. 24, despite this network having ten times more parameters than DEEPPOLAR, its performance is substantially inferior to the successive cancellation (SC) decoding of polar codes. This is an effect of the model's poor generalization in the absence of good inductive biases, a phenomenon well-documented in existing literature, for example, see (Jiang et al., 2020b). Thus, neural architectures that add redundancy in a structured way are essential for learning a good code.