LIGHTCODE: Light Analytical and Neural Codes for Channels with Feedback

Sravan Kumar Ankireddy[1], Krishna Narayanan[2], Hyeji Kim[1]

[1]University of Texas at Austin
[2]Texas A&M University, College Station

***Abstract*— The design of reliable and efficient codes for channels with feedback remains a longstanding challenge in communication theory. While significant improvements have been achieved by leveraging deep learning techniques, neural codes often suffer from high computational costs, a lack of interpretability, and limited practicality in resource-constrained settings. We focus on designing low-complexity coding schemes that are interpretable and more suitable for communication systems. We advance both analytical and neural codes. First, we demonstrate that POWERBLAST, an analytical coding scheme inspired by Schalkwijk-Kailath (SK) and Gallager-Nakiboğlu (GN) schemes, achieves notable reliability improvements over both SK and GN schemes, outperforming neural codes in high signal-to-noise ratio (SNR) regions. Next, to enhance reliability in low-SNR regions, we propose LIGHTCODE, a lightweight neural code that achieves state-of-the-art reliability while using a fraction of memory and compute compared to existing deep-learning-based codes. Finally, we systematically analyze the learned codes, establishing connections between LIGHTCODE and POWERBLAST, identifying components crucial for performance, and providing interpretation aided by linear regression analysis.**

***Index Terms*—Channels with Feedback, Deep Learning, Channel Coding, Feedback Coding, Finite Block Length Coding**

I. INTRODUCTION

Shannon introduced the concept of feedback channel in [1]. In a channel with feedback, the transmitter *cooperates* with the receiver to improve the probability of successful transmission by utilizing the *feedback* from the receiver. In [1], Shannon assumes a perfect *noiseless* feedback channel with *unit delay* and demonstrates that the availability of feedback at the transmitter does not change the capacity of the resultant forward channel for memoryless channels. Interestingly, while the capacity remains the same, significant improvements in error exponents can be achieved with the help of feedback.

One of the seminal schemes in the noiseless feedback setting has been provided by Shalkwijk and Kailath in [2], [3] (SK) using a simple linear encoding scheme resulting in a doubly exponential error exponent for finite block lengths. In [4], a two-phase variant of the SK scheme, which we refer to as Gallager-Nakiboğlu (GN), was discussed that can lead to further exponential decay of the error, provided a sufficiently good SNR is available for the forward channel. The SK scheme has been enhanced in various ways. The Modulo-SK scheme [5] and schemes by Chance-Love [6] and Mishra et al [7], extends the SK scheme for noisy feedback, while compressed error correction (CEC) [8] and accumulative iterative code (AIC) [9] focus on reducing channel use through continuous error vector compression, using noiseless feedback.

While guaranteeing impressive error exponents, SK and other analytical coding schemes have not been adapted to practical communication systems, as the improvement in performance does not justify the cost incurred in terms of high-numerical precision, increase in the amount of feedback, and large number of rounds of communication. Because of this, analytical feedback coding schemes in practice are limited to automatic repeat request (ARQ) and hybrid-ARQ (HARQ) retransmission schemes, where the receiver provides simple one-bit feedback to indicate success (acknowledgment/ACK) or failure (negative acknowledgment/NACK). Extending ARQ to multi-bit feedback is an interesting research topic; for example, compressed error hybrid ARQ (CE-HARQ) [10] and Griffin et al. [11] propose using full feedback from the receiver to iteratively improve the error vector at the receiver.

Recent advances in deep learning revived the interest in coding for channels with feedback by leveraging the expressive power [12] of deep neural networks. Several works proposed deep learning approaches to improve the performance of channel codes, ranging from augmenting analytical decoders with learnable parameters [13]–[16] to creating novel neural network architectures based neural encoders and decoders [17], [18] and improving the code design using sequential models [19], [20]. For channels with feedback, deep learning techniques were used to design new encoding and decoding schemes that take advantage of the high-capacity feedback channel. Deepcode [21] modeled both encoder and decoder functions as recurrent neural networks (RNNs) to process a bit stream in a sequential manner to directly minimize the end-to-end transmission error over an additive white Gaussian noise (AWGN) channel. Several works further explored this idea of using learning-based approaches for modeling the encoding and decoding operations, which can be broadly classified into the RNN family of codes [22]–[24] and the transformer family of codes [25], [26], discussed in detail in Section V. The current state-of-the-art is generalized block attention feedback (GBAF) [26], which uses self-attention and transformer architecture to perform block coding.

While these state-of-the-art deep-learning-based feedback codes provide tremendous improvements in BLER perfor-

Correspondence to: Sravan Ankireddy & Hyeji Kim (email: {sravan.ankireddy,hyeji.kim}@utexas.edu,
Source code available at: https://github.com/sravan-ankireddy/lightcode

mance, they also come with significant memory and computational costs that may not be supported by the next generation of communication transceivers that operate with limited onboard resources. Therefore, it is desirable to develop lightweight codes that use simple schemes while providing desirable performance. To accomplish a reduced complexity coding scheme, we explore two different directions in this work. First, we review and understand the existing analytical feedback coding schemes to identify the limitations and propose a new feedback coding scheme that provides non-trivial performance improvements over the existing schemes for channels with passive, noiseless feedback. Next, we propose a lightweight deep-learning-based feedback coding scheme that can significantly reduce the complexity compared to neural block-feedback coding schemes by limiting ourselves to a symbol-by-symbol scheme instead of block coding schemes.

Our main contributions are summarized as follows:

- We provide a comprehensive review of the existing analytical and deep-learning-based coding schemes for channels with feedback and identify the limitations of existing approaches (Section III and Section V).
- We propose POWERBLAST, an analytical coding scheme that noticeably improves the performance over Schalkwijk-Kailath and two-phase Gallager-Nakiboğlu schemes (Section IV) and exhibits reliability comparable to state-of-the-art deep-learning-based coding schemes in regions of high-SNR (Section VII).
- We propose LIGHTCODE, a lightweight neural coding scheme that achieves a performance superior to current state-of-the-art feedback coding schemes using $10\times$ fewer parameters and computational complexity (Section VI, Section VII), while maintaining interpretability.
- We analyze the representations learned by the LIGHTCODE encoder using linear regression and draw comparisons with POWERBLAST. We also demonstrate that the relation between encoded symbols and the feedback is highly non-linear in the final rounds, underscoring the need for a deep-learning-based coding scheme (Section VIII).

II. System Model

We consider a feedback coding scheme with an AWGN forward channel and an AWGN feedback channel with noisy passive feedback. The goal is to transmit a message vector $\mathbf{u} \in \{0,1\}^K$ of length K to the receiver using a total of D independent channel uses *i.e.,* the noise across the rounds is independent and identically distributed (i.i.d). This results in an overall coding rate of $R = {}^{K}/_{D}$. In this work, we consider symbol-by-symbol coding, where the block of K bits will be mapped to one symbol. Hence, the terms block and symbol can be used interchangeably, and the number of rounds of communication for a rate ${}^{K}/_{D}$ code is D.

In the first round, the transmitter encodes the message block $\mathbf{u} \in \{0,1\}^K$ to a real-valued output $x_1 \in \mathbb{R}$ and transmits using the forward AWGN channel as

$$x_1 = \phi(\mathbf{u}), \tag{1}$$

$$y_1 = x_1 + n_1, \tag{2}$$

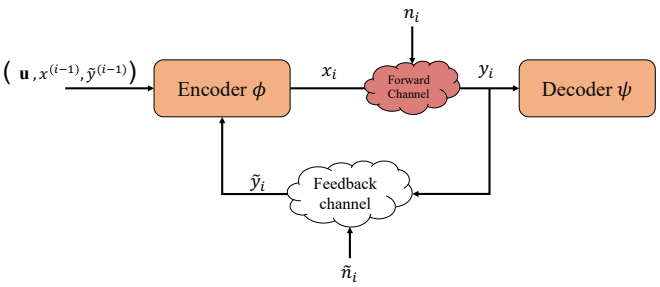


Fig. 1: Illustration of the i^{th} round of communication for channels with feedback. The encoder takes as input the message bits $\mathbf{u}$ and the encoder output from previous rounds $x^{(i-1)}$, concatenated with the feedback from previous rounds $\tilde{y}^{(i-1)}$, to compute x_i.

where ϕ is the encoding function and $n_1 \sim \mathcal{N}(0, \sigma_{ff}^2) \in \mathbb{R}$ denotes the feedforward noise. The receiver then sends the noisy received symbol as feedback using the feedback channel as

$$\tilde{y}_1 = y_1 + \tilde{n}_1, \tag{3}$$

where $\tilde{n}_1 \sim \mathcal{N}(0, \sigma_{fb}^2) \in \mathbb{R}$ denotes the feedback noise.

At round $i > 1$, the encoder ϕ computes the output x_i using the input message bits $\mathbf{u}$ and the encoder outputs from previous $i-1$ rounds $x^{(i-1)} = \{x_1, \ldots, x_{i-1}\}$ and the feedback from previous $i-1$ rounds $\tilde{y}^{(i-1)} = \{\tilde{y}_1, \ldots, \tilde{y}_{i-1}\}$ as

$$x_i = \phi\left(\mathbf{u}, x_1, \ldots, x_{i-1}, \tilde{y}_1, \ldots, \tilde{y}_{i-1}\right). \tag{4}$$

At the end of D rounds of communication, the decoder ψ estimates the transmitted message vector $\hat{\mathbf{u}}$ using the received symbols from all D rounds $\{y_1, y_2, \ldots, y_D\}$ as

$$\hat{\mathbf{u}} = \psi\left(y_1, y_2, \ldots, y_D\right), \tag{5}$$

where ψ is the decoding function and $\hat{\mathbf{u}} \in \{0,1\}^K$. The objective is to design an encoder-decoder pair $\{\phi, \psi\}$ that minimizes the probability of error $\mathbb{P}\{\mathbf{u} \neq \hat{\mathbf{u}}\}$ for a given number of rounds D, under a sum power constraint of $\sum_{i=1}^{D} \mathbb{E}[|x_i|^2] \leq D$.

III. Analytical Coding Schemes for Channels with Noiseless Feedback

In this section, we review analytical coding schemes for channels with noiseless feedback *i.e.,* $\sigma_{fb}^2 = 0$. We begin by reviewing the celebrated Schalkwijk-Kailath coding scheme [2], one of the seminal works in coding for channels with noiseless feedback. We then review a less widely-known scheme by Gallager-Nakiboğlu [4]. This is similar to the SK scheme for the first $D-1$ rounds. Still, it deviates significantly in the last round, tailoring for the transmission of discrete messages, often significantly improving the performance in one round of communication. The Gallager-Nakiboğlu (GN) scheme is typically not considered as a baseline as it exhibits a worse error rate compared to SK in certain regions of SNR. However, in the next section, we introduce a novel analytical coding scheme, building on the SK and GN schemes. This new scheme achieves a significantly lower error rate than the SK or GN schemes and is often comparable to highly complex neural coding schemes in high-SNR regions, defying the conventional

belief that neural coding schemes are much more reliable than analytical ones.

A. Schalkwijk-Kailath coding scheme

The SK scheme considers the problem of transmitting a fixed number of bits on an AWGN forward channel and a noiseless feedback channel. The transmission begins by mapping K bits of information symbols to a single 2^K-ary pulse-amplitude modulation (PAM) symbol Θ from the constellation

$$\Theta \in \{\pm 1\eta, \pm 3\eta, \ldots, \pm(2^K - 1)\eta\}, \tag{6}$$

where $\eta = \sqrt{3/(2^{2K}-1)}$ is the scaling factor to ensure unit power normalization of the PAM constellation. Even though a block of information is transmitted since all the K bits of information are mapped to one PAM symbol, the SK scheme at its core can be considered a symbol-by-symbol feedback scheme. We now describe the SK coding scheme in detail.

In the first round, the uncoded PAM symbol is transmitted after accounting for power constraint P *i.e.,* $x_1 = \sqrt{P}\Theta$ as

$$y_1 = x_1 + n_1, \tag{7}$$

where $n_1 \in \mathcal{N}(0, \sigma_{ff}^2)$ is the noise in the forward AWGN channel. The received symbol y_1 is then sent back to the transmitter noiselessly. For the second round, the transmitter first computes the receiver's estimate of the transmitted symbol based on y_1 as $\hat{\Theta}_1 = \frac{\sqrt{P}y_1}{P+\sigma_{ff}^2}$ using linear minimum mean-square error (LMMSE). In the second round, after receiving y_1 as feedback, the transmitter sends the scaled version of the *error* in the LMMSE estimate from the previous round, i.e., $\epsilon_1 = \hat{\Theta}_1 - \Theta$. This process continues for the remaining rounds. In other words, starting from round $i = 2$, the goal of the transmitter is to transmit the error in estimate from the previous round, $\epsilon_{i-1} = \hat{\Theta}_{i-1} - \Theta$, after scaling appropriately to satisfy the power constraint. The complete algorithm is described in Algorithm 1.

Error analysis. It is shown in [5] that the probability of error for rate K/D SK scheme is given by

$$p_{SK} = 2(1 - 2^{-K})Q\left(\sqrt{\frac{3S(1+S)^{D-1}}{2^{2K}-1}}\right), \tag{8}$$

where S denotes the SNR of the forward AWGN channel on a linear scale. Note that [5] assumes a minimum-variance unbiased estimator (MVUE) at the end of round 1, $\hat{\Theta}_1 = \frac{y_1}{\sqrt{P}}$, for ease of analysis but it is sub-optimal in terms of error in the estimate after round 1.

B. Gallager-Nakiboğlu coding scheme

In [4], Gallager-Nakiboğlu proposed a two-phase scheme that builds on the Elias scheme [27], also closely related to the SK scheme. While the SK scheme considers the problem of transmitting a discrete symbol, Elias studied the problem of transmitting a Gaussian random variable $U \sim \mathcal{N}(0, \sigma^2)$. The strategy for forward and feedback transmissions is similar to SK, where a scaled version of the error in the LMMSE estimate is transmitted in every round $I > 1$. The main difference between the SK and Elias schemes lies in that the SK scheme aims to refine the message itself, while the Elias scheme aims to refine the estimate of noise added in the very first transmission.

Algorithm 1: Schalkwijk-Kailath (SK) coding scheme

Input: Message symbol Θ, number of rounds D, forward noise variance σ_{ff}^2

Round 1: Tx: Power normalization: $x_1 = \sqrt{P}\Theta$;
Forward channel: $y_1 = x_1 + n_1$;
Rx: LMMSE estimate of transmit symbol $\hat{\Theta}_1 = \frac{\sqrt{P}y_1}{P+\sigma_{ff}^2}$;
`/* Tx communicates the error in estimate` $\hat{\Theta}_1 - \Theta$ `over the next` $D-1$ `rounds */`
while $2 \le i \le D$ **do**
  Tx: Compute the error in estimate of previous round $\epsilon_{i-1} = \hat{\Theta}_{i-1} - \Theta$;
  Power normalization: $x_i = \frac{\sqrt{P}}{\sigma_{i-1}}\epsilon_{i-1}$, $\sigma_{i-1}^2 = \mathbb{E}[\epsilon_{i-1}^2]$;
  Forward channel: $y_i = x_i + n_i$;
  Rx: LMMSE estimate of transmit symbol $\hat{\epsilon}_{i-1} = \frac{\sqrt{P}\sigma_{i-1}}{P+\sigma_{ff}^2}y_i$;
  Update the estimate of Θ as: $\hat{\Theta}_i = \hat{\Theta}_{i-1} - \hat{\epsilon}_{i-1}$
Decoding: Map $\hat{\Theta}_D$ to the closest symbol in the 2^K PAM constellation.

GN scheme operates in two phases and relies on the assumption that after a sufficiently large number of rounds of the Elias scheme, the effective SNR for the forward channel shall be adequate for the noise variance to be considered small compared to the distance between the symbols in the PAM constellation. In such a high-SNR regime, a strategy superior to the Elias scheme can be implemented by taking advantage of the discrete nature of the signal. Instead of transmitting the original error vector with respect to 2^K-ary PAM, the integer difference between the PAM index of the estimate and the true PAM symbol is transmitted. We refer to this as *discrete-symbol* scheme. This method results in an error exponent that decreases with an exponential *order*, which increases linearly with the number of rounds. For this work, we assume that the high-SNR region is realized in the final round of communication, which is valid according to the 2-phase strategy described in [4].

We now describe the GN scheme in detail. The first round of GN is simply uncoded PAM, except for the power allocation. In [4], it is shown the optimal power distribution across D rounds is attained by choosing P_1 and P_2 such that $P_1 + (D-1)P_2 = DP$ and $P_1 = P_2 + 1$, P_1 is the power constraint in round 1 and P_2 is the power constraint in remaining rounds. Hence, the transmission in round 1 is given by

$$y_1 = \sqrt{P_1}\Theta + n_1. \tag{9}$$

For the remaining rounds, the goal of the transmitter is to communicate the noise n_1 to the receiver as in the Elias scheme, where the LMMSE estimate at the receiver is improved iteratively, and a maximum likelihood (ML) detection is used at the end of $D-1$ rounds of communication to map the estimate to the original PAM constellation.

Finally, for the last round, GN uses a discrete-symbol scheme suitable for the high-SNR region by transmitting the error in the PAM index U. We follow the assumption from [4] that the high-SNR region guarantees that $U \in \{-1, 0, 1\}$ with high probability. In [4], an ML decoder is used for the ease of analysis across multiple rounds of discrete symbol schemes, which is sub-optimal. However, since we assume only one round of the discrete-symbol scheme in this work, we assume a maximum-a-posteriori (MAP) decoder, which is optimal for the performance. Hence, the final round of GN can be viewed as MAP detection on a constellation $\{-1, 0, 1\}$ with probability distribution $\{p_{GN1}/2, 1-p_{GN1}, p_{GN1}/2\}$, p_{GN1} is the probability of error after $D-1$ rounds. The complete algorithm is described in Algorithm 4 in Appendix, Section C.

Error analysis. The probability of error for a rate K/D Gallager-Nakiboğlu scheme can be computed in two phases. The first phase can be analyzed as one round of uncoded PAM followed by $D-2$ rounds of Elias scheme, for which it is shown in [4] that the probability of error is given by

$$p_{GN1} = 2(1-2^{-K})Q\left(\sqrt{\frac{3(1+S-\frac{1}{D-1})^{D-1}}{2^{2K}-1}}\right), \quad (10)$$

where S denotes the SNR of the forward AWGN channel on a linear scale.

In the second phase, which corresponds to the final round, the discrete integer difference between the PAM index of the decoded message $\hat{M}$ and the index of the true message M is transmitted. Here, the message index $M \in \{0, 1, 2, \ldots, 2^K - 1\}$ is deterministic based on the transmitted symbol $\Theta \in \{\pm 1\eta, \pm 3\eta, \ldots, \pm(2^K-1)\eta\}$ and can be computed using a predetermined mapping, where η is the scaling factor to ensure unit power normalization. Similarly, $\hat{M}$ corresponds to the message index whose corresponding symbol is closest to the estimated transmitted symbol $\hat{\Theta}$.

As the error in the $(\hat{M} - M)$ lies in $\{-1, 0, 1\}$ with probability distribution $\{p_{GN1}/2, 1-p_{GN1}, p_{GN1}/2\}$ based on the assumption from [4], the probability of error in decoding using a MAP decoder can be computed as

$$p_{GN} = 2(1-p_{GN1})Q\left(\gamma\sqrt{S}\right) + p_{GN1}Q\left(\left(\frac{1}{p_{GN1}} - \gamma\right)\sqrt{S}\right), \quad (11)$$

where S is the SNR of the forward AWGN channel on a linear scale, and γ is the detection threshold given by

$$\gamma = \frac{1}{2\sqrt{p_{GN1}}} + \frac{\sqrt{p_{GN1}}}{S}\log\left(\frac{2(1-p_{GN1})}{p_{GN1}}\right). \quad (12)$$

IV. Proposed analytical coding scheme: PowerBlast

In this section, we propose PowerBlast, a hybrid 2-phase scheme, to iteratively refine the LMMSE estimate of the PAM symbol at the receiver and shift to a discrete symbol scheme that takes advantage of the sparsity of the error in the estimate of the PAM index in the final round. Through theoretical analysis and empirical results, we demonstrate that the performance of PowerBlast is better than both SK and GN in several regimes of interest.

For a rate K/D code, PowerBlast begins by mapping K bits of information to a 2^K PAM constellation and transmits the symbols on the forward AWGN channel. The receiver performs an LMMSE estimate and sends the estimate as feedback to the transmitter through the noiseless feedback channel. This continues for the first $D-1$ rounds. In the final round, PowerBlast uses a discrete symbol strategy to send the error in the PAM index of the estimate. The complete algorithm is described in Algorithm 2.

Algorithm 2: PowerBlast coding scheme

Input: Message symbol Θ, number of rounds D, forward noise variance σ_{ff}^2

Round 1: Tx: Power normalization: $x_1 = \sqrt{P}\Theta$;
Forward channel: $y_1 = x_1 + n_1$;
Rx: LMMSE estimate of transmit symbol $\hat{\Theta}_1 = \frac{\sqrt{P}y_1}{P+\sigma_{ff}^2}$;

`/* Tx communicates the error in estimate` $\hat{\Theta}_1 - \Theta$ `over the next` $D-2$ `rounds */`

while $2 \le i \le D-1$ **do**
  Tx: Compute the error in estimate of previous round $\epsilon_{i-1} = \hat{\Theta}_{i-1} - \Theta$;
  Power normalization: $x_i = \frac{\sqrt{P}}{\sigma_{i-1}}\epsilon_{i-1}$, $\sigma_{i-1}^2 = \mathbb{E}[\epsilon_{i-1}^2]$;
  Forward channel: $y_i = x_i + n_i$;
  Rx: LMMSE estimate of transmit symbol $\hat{\epsilon}_i = \frac{\sqrt{P}\sigma_{i-1}}{P+\sigma_{ff}^2}y_i$;
  Update the estimate of Θ as: $\hat{\Theta}_i = \hat{\Theta}_{i-1} - \hat{\epsilon}_{i-1}$

Decoding after round $D-1$: Map $\hat{\Theta}_{D-1} = \frac{\hat{x_1}}{\sqrt{P_1}\eta}$ to the closest symbol in the 2^K PAM constellation.

`/* High-SNR scheme for final round */`

Round D: Tx: Compute the difference in PAM indices $U = \hat{M} - M$, where $\hat{M}$ and M correspond to the integer index from PAM constellation for $\hat{\Theta}$ and Θ respectively;
Power normalization: $x_D = \frac{\sqrt{P}U}{\sigma_{ff}}$; Transmit: $y_D = x_D + n_D$;
Final Decoding: Use MAP decoder to detect $\hat{U}$ and detect original PAM signal using $\hat{U}$.

We note that PowerBlast can be seen as a combination of SK and GN. Rate K/D PowerBlast can be interpreted as $D-1$ rounds of rate K/D SK scheme with LMMSE estimate, followed by the discrete variant of GN scheme for

the final round, resulting in a non-negligible improvement in performance in certain regions of SNR. The error analysis for the POWERBLAST scheme is provided below.

Theorem 1 (Error Analysis)**.** *The probability of error for rate $^{K}/_{D}$ POWERBLAST scheme is given by*

$$p_{PB} = 2(1-p_{SK})Q\left(\gamma\sqrt{S}\right) + p_{SK}Q\left(\left(\frac{1}{p_{SK}} - \gamma\right)\sqrt{S}\right) \tag{13}$$

where

$$p_{SK} = 2(1-2^{-K})Q\left(\sqrt{\frac{3S(1+S)^{D-2}}{2^{2K}-1}}\right) \tag{14}$$

and γ denotes the detection threshold and S denotes the SNR of the forward AWGN channel on a linear scale, and $Q()$ is the standard Q-function.

Proof. We begin by computing the probability of error for phase 1, which consists of $D-1$ rounds of the SK scheme. For ease of analysis, we follow the same assumption of [5], where the estimator after round one is assumed to be MVUE instead of LMMSE, implying,

$$\hat{\Theta}_1 = \frac{y_1}{\sqrt{P}}. \tag{15}$$

Further, based on Algorithm 1 , we can view the effective channel corresponding to rounds 1 to $D-1$ as one round with effective SNR $S.(1+S)^{D-2}$, $S = {}^{P}/_{\sigma_{ff}^2}$ is SNR for the forward channel in linear scale. It is straightforward to show that the probability of error incurred in transmitting a symbol from unit power normalized 2^K PAM, using an optimal detector, is

$$\begin{aligned} p_e &= 2(1-2^{-K})Q\left(\frac{\sqrt{P}\eta}{\sigma}\right) \\ &= 2(1-2^{-K})Q\left(\sqrt{\frac{3S}{2^{2K}-1}}\right), \end{aligned}$$

where S is the forward channel SNR in linear scale and $Q()$ is the Q-function. Hence, the effective probability of error after $D-1$ rounds of SK is

$$p_{SK} = 2(1-2^{-K})Q\left(\sqrt{\frac{3S(1+S)^{D-2}}{2^{2K}-1}}\right). \tag{16}$$

We now proceed to compute the probability of error for phase two, which is transmitting the difference in integer (message) index over an AWGN channel. The key assumption for the high-SNR region is that the noise variance is small enough (compared to the effective forward SNR) that any errors beyond decoding to adjacent PAM symbols *i.e.,* errors beyond 1 index difference are essentially negligible [4]. Hence, the problem can be rephrased as finding the probability of error to communicate a symbol from the constellation $\{-1,0,1\}$, with probability distribution $\{{}^{p_{SK}}/_{2}, 1-p_{SK}, {}^{p_{SK}}/_{2}\}$ using MAP decoder. This can be computed as

$$p_{PB} = 2(1-p_{SK})Q\left(\gamma\sqrt{S}\right) + p_{SK}Q\left(\left(\frac{1}{p_{SK}} - \gamma\right)\sqrt{S}\right), \tag{17}$$

where γ is the detection threshold and S is the SNR of the forward AWGN channel on a linear scale.

□

POWERBLAST vs. SK and GN schemes. The key difference between POWERBLAST and SK is in the last round of communication, where POWERBLAST shifts to a discrete symbol scheme, as seen in line 13 of Algorithm 2, by taking advantage of the high effective SNR over the initial $D-1$ rounds *i.e.,* the (effective) noise variance is much smaller than the distance between adjacent PAM symbols. Hence, it is beneficial to now transmit the error in the PAM constellation index, which lies in $\{-1,0,1\}$ with a very high probability, resulting in a much lower probability of decoding error.

Further, compared to GN, the key difference in POWERBLAST is the choice of information transmitted after round 1. GN communicates and refines the noise from round 1, n_1, starting round 2, as seen in line 4 of Algorithm 4. In contrast, POWERBLAST transmits and refines the error in LMMSE estimate of Θ, as seen in line 4 of Algorithm 2. It is seen from [5] that the effective SNR (in linear scale) after D rounds is given by $S(1+S)^{D-1}$ for GN and $(1+S)^D-1$ for POWERBLAST. While the difference in effective SNR becomes negligible for large D, it can be significant when operating for a finite number of rounds, resulting in the superior performance of POWERBLAST.

The performance comparison between the error expressions for a general SNR and rate is not straightforward (as they are represented in terms of Q functions). Instead, we limit our comparison to the canonical settings considered in the recent literature on channels with feedback (e.g., [26]), for rates ${}^{3}/_{6}$ and ${}^{3}/_{9}$ and plot the BLER performance in Fig. 2; we observe significant gains of POWERBLAST compared to both SK and GN in terms of the BLER performance.

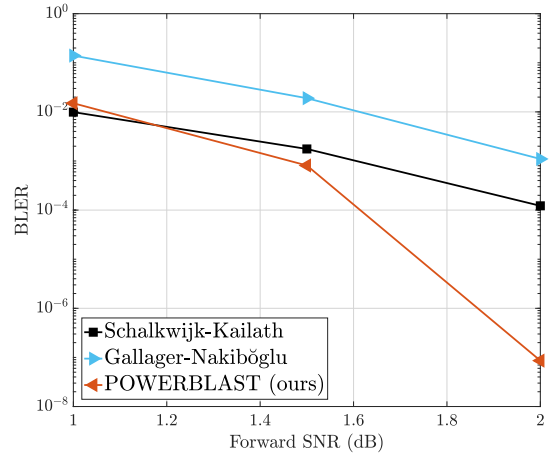
From the results so far, we have demonstrated that POWERBLAST provides the best performance among the analytically tractable solutions for channels with noiseless feedback that are recently considered in the literature (to demonstrate the reliability of deep-learning-based coding schemes [26]). In the coming sections, we consider deep-learning-based coding schemes such as GBAF and show that, surprisingly, the analytical POWERBLAST coding scheme often delivers competing performance. Nevertheless, POWERBLAST still falls short when the SNR is very low; thus, we investigate conditions under which deep-learning-based schemes provide the highest gain. Finally, we propose LIGHTCODE, a lightweight neural coding scheme that achieves state-of-the-art performance with very low complexity.

V. DEEP LEARNING BASED CODING SCHEMES

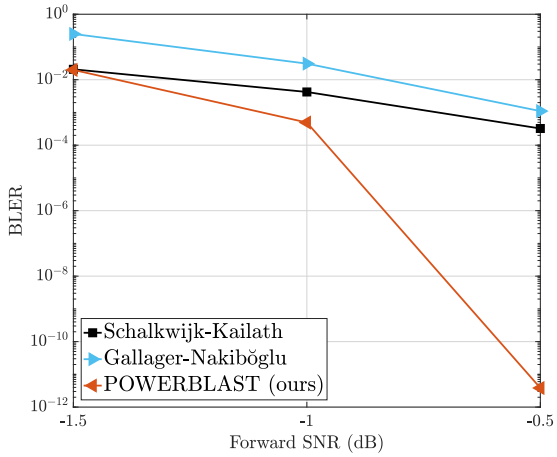
In this section, we review the current state of the learning-based codes for channels with feedback, closely analyze the generalized block attention feedback (GBAF) [26], discuss the shortcomings, and provide motivation for a new learning-based coding scheme we introduce in the next section.

A. RNN families

Deep-learning-based algorithms are capable of modeling complex input-output relations. One of the critical challenges



(a) Rate $R = 3/6$



(b) Rate $R = 3/9$

Fig. 2: By combining the SK and discrete-symbol strategy of the GN scheme, POWERBLAST noticeably improves the BLER performance upon both SK and GN schemes.

in designing codes for channels with feedback is accurately computing the subsequent transmission that minimizes the probability of decoding error, conditioned on the feedback from previous rounds. Existing classical coding schemes employ linear estimators at the receivers and model the dependencies in a linear fashion at the transmitter, which is sub-optimal. Instead, the sequential nature of the feedback from previous can be better leveraged by using deep-learning architectures such as RNNs tailored for processing sequential data. Deepcode [21] demonstrated this advantage in modeling the dependencies across bits and rounds using bi-directional gated recurrent units (GRUs). Several follow-up works investigated the use of other RNN-based architectures such as long short-term memory (LSTMs) [22], [23] and, more recently, Robust Coding [24] was proposed, which combined attention mechanism with bi-directional GRUs to optimize the symbol-by-symbol code design for noisy feedback channels across the rounds. Despite the promising performance, one disadvantage of using RNN-based architecture is the necessity to store the hidden states of the model, which are typically of much higher dimensions than the inputs and demand a lot of memory. Additionally, the sequential and iterative nature of encoding and decoding in RNNs can result in significant latency in the system.

B. Transformer families

In another line of work, self-attention-based transformer architectures have been explored for designing neural feedback codes. AttentionCode [25] introduced the idea of replacing RNN-based architecture with pure attention-based models, resulting in better alignment between a symbol and the corresponding feedback. In other words, AttentionCode can be viewed as Deepcode with transformer architecture. By leveraging the attention mechanism, AttentionCode creates temporal correlations at the transmitter for encoding and exploits these temporal correlations at the receiver for decoding. Further, the inputs to the encoder and decoder transformers are restructured to align each bit with the corresponding noise from multiple rounds as a single column, processed by the self-attention mechanism. More recently, GBAF [26] introduced the idea of performing block coding across the codeblock, in addition to temporal coding across the rounds, resulting in orders of magnitude improvement in performance at extremely low SNRs, explained in detail below.

As illustrated in Fig. 3, the encoding of GBAF is performed across the rounds by causally concatenating the message and feedback symbols from previous rounds and using a series of feature extractors and multi-layer perception (MLP) modules. Additionally, positional encoding (PE) and a self-attention-based transformer encoder layer are deployed to encourage the mixing of information across the symbols within a codeblock, leading to block coding. More specifically, a block of L bits is divided into l sub-blocks of K bits each and first encoded independently using a feature extractor. Next, PE and transformer encoder modules perform cross-symbol coding across the l symbols. Finally, an MLP module is used to encode each symbol. A similar architecture is used at the decoder, but the output dimensions are adjusted accordingly.

For concreteness, in [26], GBAF considers a block size $K = 3$ and codeblock length of $L = 51$ and performs a block coding across $l = 17$ symbols. This is the present state-of-the-art in performance, achieving a BLER of 7×10^{-10} at SNR -1.0 dB for rate $3/9$.

C. Important open problems

While providing impressive performance and reliability, transformer architecture is computationally expensive. Further, it is well known that the transformer architecture does not scale well to larger sequences. The self-attention mechanism imposes a compute complexity of $O(n^2)$ during training and $O(n)$ during inference, even after using the KV cache, with respect to input length n. Moreover, as the blocklength scales, a memory complexity of $O(n^2)$ prohibits training the algorithm from using a large batch size, which is crucial for attaining a good performance for any deep-learning models.

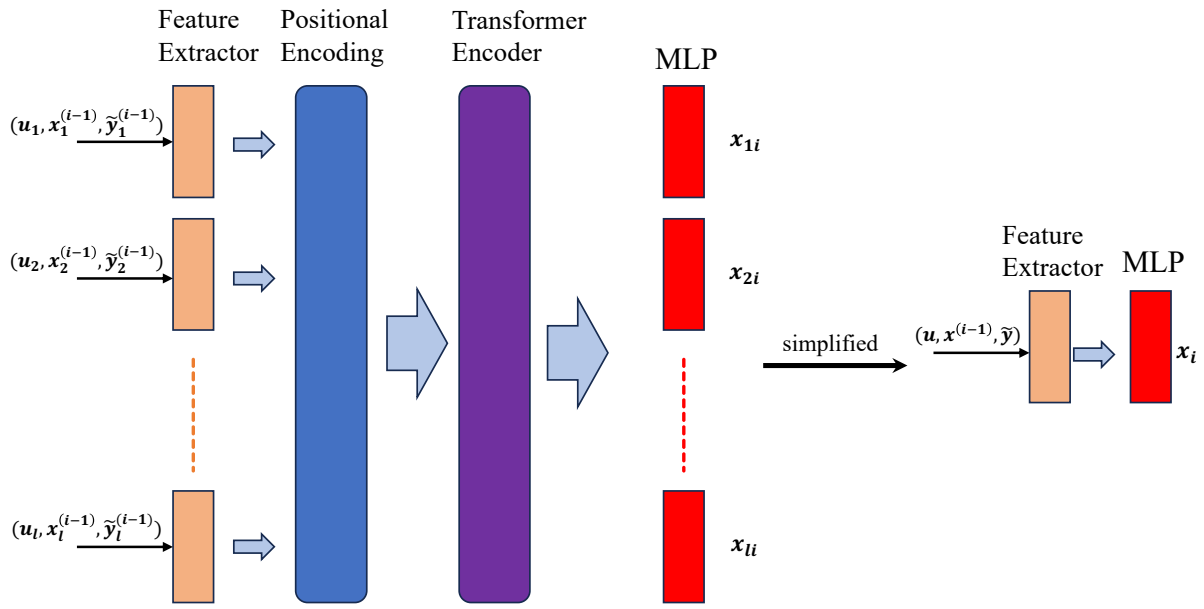


Fig. 3: (Left)Architecture for GBAF: The positional encoding and transformer encoding modules are used for block coding to encourage the mixing of symbols across the positions. (Right): Using a symbol-by-symbol scheme, LIGHTCODE significantly reduces the complexity of encoding and achieves more than 10x reduction in the number of parameters. On the left, we see the architecture for GBAF [26], and on the right, we see the architecture for LIGHTCODE (ours).

An alternative approach would be to design a symbol-by-symbol coding scheme that can be scaled to any blocklength L by encoding K bits at a time independently. Within this context, an important question is: what is the extent of performance degradation compared to neural block coding schemes like GBAF? Moreover, it is necessary to formulate a novel and simplified architecture tailored to symbol-by-symbol processing. Finally, through streamlining the architecture and confining to symbol-by-symbol schemes, can we analyze and interpret the codes learned by deep learning models, discerning the reasons behind their notably superior performance compared to analytical counterparts?

In the coming sections, we answer all these questions. In Section VI, we systematically present LIGHTCODE, a lightweight symbol-by-symbol neural coding scheme, which achieves state-of-the-art BLER performance (Section VII). Further, we analyze GBAF to study the efficacy of block coding by performing a systematic ablation study and also analyze LIGHTCODE to identify the crucial components necessary for achieving ultra-low BLER in Section VIII.

VI. PROPOSED NEURAL CODING SCHEME: LIGHTCODE

Our goal is to design lightweight neural codes without the need for block coding *i.e.,* we limit ourselves to symbol-by-symbol schemes suitable for both noiseless and noisy feedback settings. To this end, we design a lightweight deep-learning-based scheme with $10\times$ fewer parameters compared to existing schemes. Surprisingly, this low-complex solution achieves a performance superior to current state-of-the-art deep-learning-based block-coding schemes. We now present the architecture and training choices crucial to achieving this new state-of-the-art BLER performance.

A. LIGHTCODE: *Architecture*

Our architecture. We split the design of the encoder-decoder architecture into two parts: the feature extractor and the multi-layer perception (MLP) module, as illustrated in Fig. 3 (right). The choice of the feature extractor plays a crucial role in determining the downstream performance. A higher complexity feature extractor can perform better but might not be desirable for practical applications. For LIGHTCODE, after experimenting with various choices, we found that the design illustrated in Fig. 4 gave the optimal trade-off in complexity vs. BLER performance. Further, the output of the feature extractor will be passed to an MLP module. For the encoder, we choose a 1 layer MLP to project the features to a 1 dimensional output. For the decoder, we choose a 2 layer MLP to transform the features into an output of dimension 2^K. The full architecture for encoder and decoder, including the MLP are provided in Appendix B, Fig. 12 and Fig. 13.

Our architecture vs. RNNs. A popular choice for modeling the cross-round relation in feedback coding has been the RNN family of architectures. The sequential nature of the data makes it suitable for RNNs, GRUs, LSTMs, and other similar architectures. While these architectures provided impressive performance, a significant drawback is the necessity to store the hidden states from previous encoding steps, which is a high-dimensional latent that requires a lot of memory. Hence, a feed-forward architecture, which does not need a hidden state, is a better choice for resource-constrained scenarios.

Our architecture vs. transformers. While transformer models overcome the issue of storing hidden states by using positional encoding and self-attention, they also come with great computational complexity. Moreover, recent transformer-based feedback schemes proposed block coding using self-attention, which requires $O(n^2)$ memory and compute with

respect to codeblock length. But as evident from results Section VII, block coding does not seem to provide noticeable gains in the setting under consideration. Hence, we propose a symbol-by-symbol coding scheme that uses a significantly simpler feed-forward architecture.

Complexity. By eliminating the need for block coding and using a short code length, LIGHTCODE avoids the high-complexity transformer module used in GBAF architecture and instead uses a simple feed-forward network. By carefully designing the architecture suitable for a symbol-by-symbol scheme, we achieve a lightweight design with more than $10\times$ reduction in the number of parameters compared to the RNN family of schemes such as Robust Coding and transformer family of block coding schemes such as GBAF. Further, we show in Section VIII-C that this reduction in complexity provides up to $171\times$ higher decoding throughput compared to Robust Coding and up to $10\times$ higher throughput compared to GBAF.

Feature extractor. Fig. 4 depicts the architecture of the feature extractor, which is the backbone for both encoder and decoder models. Compared to the feature extractor used in GBAF, we introduce two changes to improve the performance and reduce the complexity. The first is to add a skip connection, which preserves the prior from input to the feature extractor better, similar to the design of DEEPPOLAR [18]. Further, the hidden dimension is reduced from 64 to 32, decreasing the number of parameters.

Specifically, the input to the feature extractor is processed through a sequence of three linear layers, each with a hidden dimension of 32, and rectified linear unit (ReLU) activation functions are applied between the first two layers. In parallel, we add a skip connection from output of first layer to input of final layer, which reverses the sign of the representation to introduce variance in the information. These outputs are concatenated and passed through the final layer, which generates a 16-dimensional feature representation.

Training. Short code length and lightweight architecture allow for training of LIGHTCODE with an extremely large batch size of 10^5, which is more than $10\times$ larger than the maximum batch size suitable for GBAF codes. We refer to Section VI-B for a detailed discussion of the training details and Section VIII-A for an ablation study on the role of batch size.

We now describe the detailed encoding and decoding procedure below.

Encoding. At round i, the encoder takes as input the original message and any available feedback from previous $i-1$ rounds to compute x_i. Further, after every round, a power reallocation is done across the rounds, similar to Deepcode [21]. This is also based on the theoretical justification that allocating more power to the initial rounds results in an optimal performance [4]. The resulting encoding process at round i is

$$x_i = \alpha_i \phi(\mathbf{u}, \tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_{i-1}, 0, \ldots, 0), \tag{18}$$

where $\tilde{y}_i = x_i + n_i + \tilde{n}_i$ is the feedback from the previous round and α_i is the scaling factor to ensure sum power constraint across the rounds $\sum_{i=1}^{D} \alpha_i^2 = D$. Here, we note that in order to keep the size of the input to the encoder constant, the input is padded with zeros where necessary.

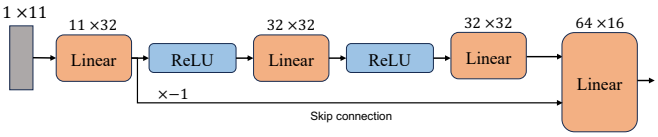


Fig. 4: Feature extractor design for LIGHTCODE for a rate $3/9$ code.

Decoding. At the end of D rounds of transmission, the decoder uses all the received symbols and estimates the original message $\hat{\mathbf{u}}$ as

$$\hat{\mathbf{u}} = \psi(y_1, y_2, \ldots, y_D), \tag{19}$$

where $y_i = x_i + n_i$ is the noisy received symbol in round i.

Here, ϕ and ψ are the encoder and decoder neural networks based on the architecture in Fig. 3. Using this simple symbol-by-symbol scheme, LIGHTCODE archives a performance similar to that of GBAF. It turns out that the computationally intensive self-attention mechanism and the cross-symbol coding are not adding considerable value to GBAF, as will be evidenced by our ablation studies in Section VIII-A.

Choice of feature vectors. Empirically, we observed that the choice of inputs to the feature extractor has a noticeable effect on the BLER performance based on the feedback channel. For noiseless feedback, the input features $(\mathbf{u}, y_1, y_2, \ldots, y_i)$ worked the best. Further, as will be evident from discussions in Section VII-C, the input features $(\mathbf{u}, x_1, x_2, \ldots, x_i, n_1 + \tilde{n}_1, n_2 + \tilde{n}_2, \ldots n_i + \tilde{n}_i)$ gave the best performance for noisy feedback. Apart from this difference in features, the rest of the architecture remains the same for noiseless and noisy feedback scenarios.

Support for multiple rates. One of the limitations of existing symbol-by-symbol coding schemes is the inability to serve a variety of rates *i.e.,* only rates of the form $1/D, D \in \mathbb{Z}^+$ are supported. In order to overcome this, instead of processing 1 bit at a time, LIGHTCODE encodes a block of K bits into 1 symbol and communicates this symbol to the receiver over D rounds. By independently varying K and D any rate of the form K/D where, $K, D \in \mathbb{Z}^+$ can be supported, making the scheme more flexible, while simultaneously reducing the overall latency of the communication by a factor of K. The results for multiple rates for a block length of $K = 3$ are discussed in Table III.

B. Training

Our primary region of interest is rate $3/9$ code at SNR -1.0 dB, where our target BLER is $\approx 10^{-9}$. This is the current state-of-the-art performance by GBAF [26]. To achieve such an extremely low error rate, it is important to train and simulate large amounts of samples reliably. Inspired by the high-SNR variant of Gallager-Nakiboğlu [4], we hypothesize that in regions of high-SNR or low errors, a significant benefit to the forward transmission arises from the power reallocation to symbols with non-zero errors. This can only be realized

during the training by considering a large batch and enforcing the power constraint per batch. Moreover, it is well understood that deep-learning models generalize better when trained with large batch sizes. Hence, we consider an extremely large batch size of 10^5. While this is significantly larger than GBAF, which has a batch size of 8192, it is still a smaller number of symbols, considering that GBAF contains 17 symbols per codeblock. In contrast, our scheme consists of only 1 symbol per codeblock. Further, it is possible to train with such a large batch size because of the small number of parameters compared to other deep-learning-based coding schemes. For a fair comparison, we follow a training methodology similar to that of GBAF, as explained below.

Algorithm 3: Training LIGHTCODE

Input: Encoder model ϕ, Decoder model ψ, Block length K, number of rounds D, forward noise variance σ_{ff}^2, feedback noise variance σ_{fb}^2, batch size B, number of epochs E, learning rate `lr`

for $i \leq E$ **do**
- Generate batch of random binary vectors $\mathbf{u} \in \{0,1\}^{K \times B}$
- **for** $i \leq D$ **do**
  - ; `/* Encoding at round` i `*/`
  - **if** $\sigma_{fb}^2 == 0$ **then**
    - $x_i = \phi\left(\mathbf{u}, y_1, y_2, \ldots, y_{i-1}, 0, \ldots, 0\right)$
  - **else**
    - $x_i = \phi\left(\mathbf{u}, x_1, \ldots, x_i, n_1 + \tilde{n}_1, \ldots n_i + \tilde{n}_{i-1}, \ldots, 0\right)$
  - $y_i = x_i + n_i$
- $\mathbf{p_u} = \psi(y_1, y_2, \ldots, y_D)$; `/* Decoding after` D `rounds */`
- Compute the multi-class cross entropy loss $\frac{1}{B}\sum_{j=1}^{B} \mathcal{L}_{\text{CE}}(\mathbf{c_j}, \mathbf{p_{u_j}})$, $\mathbf{c_j}$ is the class index corresponding to j^{th} message vector $\mathbf{u_j}$ and $\mathbf{p_{u_j}}$ is class probability vector after the `SoftMax` layer.
- Clip the gradients to 0.5.
- Update model parameters for ϕ and ψ using `AdamW` optimizer with learning rate `lr`.
- Update the learning rate using `LambdaLR`.

We use `AdamW` optimizer, a stochastic optimization method that modifies the typical implementation of weight decay in `Adam` by decoupling weight decay from the gradient update. We initialize the learning to 10^{-3} and use a `LambdaLR` scheduler with a weight decay of 0.01. Additionally, we clip all the gradient values to 0.5 for numerical stability. Starting with randomly initialized weights, we jointly train the encoder and decoder models on 1.2×10^5 batches. Each input symbol corresponds to K bits, resulting in a 2^K category classification problem for the decoder. Accordingly, we use the multi-class cross entropy (CE) loss to measure the performance as

$$\mathcal{L}_{\text{CE}} = \frac{1}{B}\sum_{i=1}^{B}\left(\sum_{j=0}^{2^K-1} c_{ij}\log p_{ij}\right),$$

where B is the batch size, 2^K is the number of classes , c_{ij} is the true class probability and p_{ij} is the predicted probability of the j^{th} class, for the i^{th} sample.

The hyperparameters used for training the rate $R = 3/9$ code are listed in Table I.

Hyperparameter	Value
Encoder training SNR	-1.0 dB
Decoder training SNR	-1.0 dB
Mini batch size (B)	100,000
Total epochs (E)	120
Batches per epoch	1000
Optimizer	AdamW
Initial learning rate (`lr`)	10^{-3}
Learning rate scheduler	LambdaLR

TABLE I: Hyperparameters for training rate $3/9$ LIGHTCODE.

Once the training is complete, we compute the mean power and standard deviation for the encoded data after every round for a large number of samples, $O(10^6)$, to reliably estimate the mean and standard deviation corresponding to encoder outputs to be used during inference. This is crucial in enforcing the power constraint in expectation. The algorithm for training LIGHTCODE is described in detail in Algorithm 3.

VII. MAIN RESULTS

We begin by comparing the performance of LIGHTCODE with the current state-of-the-art in deep-learning-based coding schemes for noiseless passive feedback setting, GBAF [26] and other schemes, demonstrated in Fig. 5. Next, we discuss a method for extending LIGHTCODE to moderate block-length regimes of up to $L = 51$ and study the performance.

Finally, we look at deep-learning-based coding schemes for noisy feedback settings and how LIGHTCODE can be extended to this setting with minimal changes. We then proceed to compare the BLER performance for the same configuration as before but with a feedback SNR of 20 dB.

A. LIGHTCODE *and* POWERBLAST *vs. existing neural codes for noiseless feedback*

In this section, we evaluate the performance of LIGHTCODE and POWERBLAST and compare them against several existing analytical and deep-learning feedback schemes. For concreteness, we consider the canonical setting of rate $R = 3/9$ with block length $K = 3$ and $D = 9$ rounds of communication on AWGN forward channel and noiseless feedback.

Baselines. Our primary comparison is against GBAF [26], which is the current state-of-the-art for the noiseless passive feedback setting. Additionally, we consider Deepcode [21], DEFC [22], DRFC [23], Attentioncode [25], and Robust Coding [21]. Further, for completeness as well as to understand the relative gains of deep-learning-based coding schemes, we also

compare the performance against NR-LDPC [28], Schalkwijk-Kailath [2], Gallager-Nakiboğlu [4] and POWERBLAST.

Results. In Fig. 5, we compare the BLER performance of rate $3/9$ coding schemes. We train a pair of encoder-decoder models at each SNR point in the plot. LIGHTCODE consistently outperforms the existing deep-learning-based schemes, including GBAF, while utilizing $< 1/10^{\text{th}}$ the number of parameters. Interestingly, these results indicate that POWERBLAST surpasses the performance of all existing schemes, including LIGHTCODE, when an adequately high signal-to-noise ratio is provided, which is -0.5 dB for rate $3/9$. However, the performance of deep-learning codes is significantly better at lower SNRs. Here, we highlight that by utilizing the clean feedback, LIGHTCODE and POWERBLAST achieve an extremely small error rate even when operating at 0.75 dB below the channel capacity. Further, we note that because of the extremely short block lengths considered, it is hard to derive meaningful achievability or converse bounds, such as [29], where blocks lengths of 100 or higher are considered.

The blocklengths considered for the baselines in Fig. 5 are listed in Table II, where $K = 50$ for some of the schemes, which is larger than the length $K = 3$ considered for rest of the schemes including LIGHTCODE. While the BLER comparison against the different blocklengths is unfair, to maintain consistency, we used the same methodology followed in the current state-of-the-art results, including GBAF [26] and Robust Coding [24]. For a fairer comparison, we propose a modular approach for extending LIGHTCODE to larger blocklengths in Section VII-B.

Scheme	Message length K
Deepcode	50
DEFC	50
DRFC	50
AttentionCode	50
Gallager-Nakiboğlu	3
POWERBLAST	3
Robust Coding	3
GBAF	3*
LIGHTCODE	3

TABLE II: Block lengths of different coding schemes.

Multiple rates. The coding rate of LIGHTCODE is determined by two factors, block length K and number of rounds of communication D. We keep the block length constant and vary D to compare the performance against GBAF across multiple rates, as shown in Table III. LIGHTCODE consistently outperforms GBAF across a range of rates $\{3/9, 3/8, 3/7, 3/6, 3/5\}$, while utilizing a fraction of the number of parameters.

Error floor. As observed in Fig. 5, LIGHTCODE exhibits an error floor in the high-SNR region, similar to the existing deep-learning-based coding schemes such as Deepcode and GBAF. We believe a key reason for this behavior is the difficulty of training the decoder in the high SNR region, where the error events are extremely rare. For instance, while training in the regime of BLER 10^{-9}, most of the training batches (batch size 10^5) are error-free and do not provide sufficient guidance for the gradient descent to learn useful information, leading to saturation in performance.

*While GBAF uses a blocklength of 51, the BLER in [26] was reported for the sub-block of length 3.

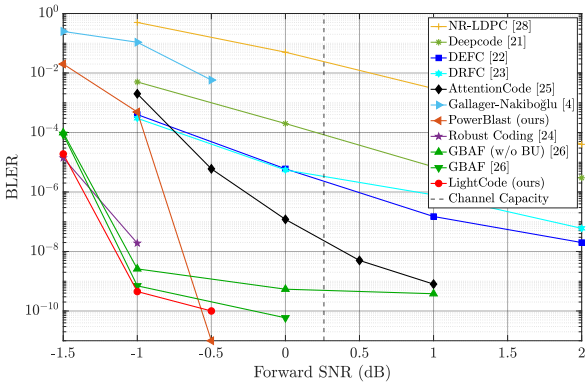


Fig. 5: Noiseless feedback: Performance comparison against existing classical and neural feedback codes for rate $3/9$. POWERBLAST achieves the best performance among existing classical schemes and performs comparable to state-of-the-art neural coding schemes in high-SNR regions. LIGHTCODE achieves superior BLER performance compared to GBAF while utilizing $< 1/10^{\text{th}}$ the number of parameters.

SNR (dB)	Rate	GBAF	POWERBLAST	LIGHTCODE
-1.0	$3/9$	7×10^{-10}	2.8×10^{-4}	$\mathbf{4.5 \times 10^{-10}}$
0.0	$3/8$	6.1×10^{-8}	8.8×10^{-7}	$\mathbf{5.1 \times 10^{-9}}$
1.0	$3/7$	7.5×10^{-8}	1.0×10^{-8}	$\mathbf{1.0 \times 10^{-8}}$
2.0	$3/6$	1.5×10^{-6}	1.5×10^{-8}	$\mathbf{8.3 \times 10^{-7}}$
3.0	$3/5$	8.7×10^{-7}	1.9×10^{-4}	$\mathbf{2.7 \times 10^{-7}}$

TABLE III: BLER performance comparison of LIGHTCODE with GBAF with POWERBLAST for different rates. LIGHTCODE consistently performs better than GBAF while using $< 1/10^{\text{th}}$ the number of parameters.

B. Performance in moderate blocklength regime

We now present a modular way to scale LIGHTCODE to larger block lengths to enable a fair comparison against baselines with longer blocklengths. As a result of the curse of dimensionality, it is well known that the hardness of learning a code increases considerably with an increase in block length. As seen from results in Appendix Section A, directly increasing the blocklength for LIGHTCODE results in a poor BLER performance. GBAF [26] uses a computationally intensive transformer for performing block coding to support a length of 51. To keep the complexity low, LIGHTCODE instead uses a short blocklength K (e.g., $K = 3$) and treats a block of L bits as $l = L/K$ independent sub-blocks. This modular approach of encoding K bits at a time is motivated by ablation studies that demonstrate the negligible benefit of block coding in GBAF, presented in detail in Section VIII-A. Thus, the resulting BLER for a blocklength L LIGHTCODE can be computed as

$$p_L = 1 - (1 - p_K)^l, \tag{20}$$

where p_L is the BLER for blocklength L and p_K is the BLER for blocklength K.

Finally, for a fair comparison against schemes with block length $K = 50$, we refer to the results in Fig. 6 where

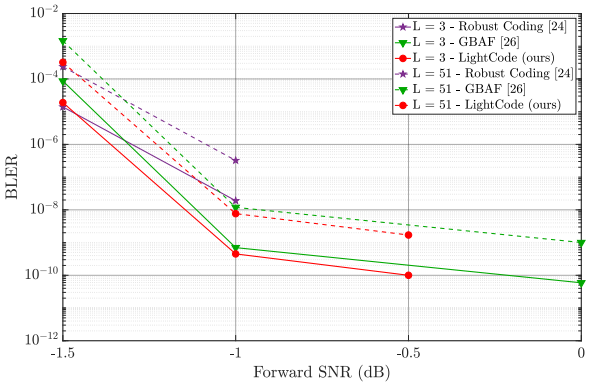


Fig. 6: By independently encoding sub-blocks of length $K = 3$, LIGHTCODE provides a flexible way to encode for any blocklength L, providing a steady trade-off in BLER performance vs blocklength. Even at a block length of 50, LIGHTCODE significantly outperforms all baseline methods shown in Fig. 5, which utilize a maximum block length of 51.

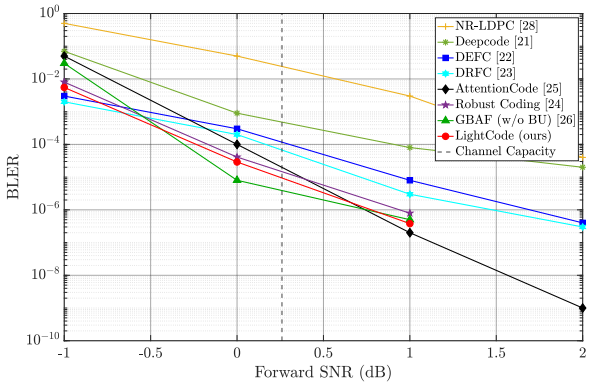


Fig. 7: Noisy feedback: Performance comparison against existing neural feedback codes for rate $3/9$ and feedback SNR = 20 dB. LIGHTCODE achieves BLER performance comparable to GBAF while utilizing only $1/10^{\text{th}}$ the number of parameters.

we compute the BLER for blocklength $L = 51$ for LIGHTCODE using the modular approach presented above. At a forward SNR of -1.0 dB, LIGHTCODE with blocklength 51 has a BLER of 7×10^{-9} which is significantly smaller than Deepcode, DEFC, DRFC, and AttentionCode that use a block length of 50, demonstrating the superior performance of LIGHTCODE even at moderately longer blocklengths. We note that the BLER p_L approaches 1 for very large values of L, but we restrict our study to the relatively short block length regime of $L < 300$, where the performance is still superior to the baselines as demonstrated in Fig. 6. Further potential performance improvements at longer blocklengths may be achievable through a concatenated coding scheme that employs an outer block code, as explored in [6]; we leave this as a direction for future work.

C. Coding for Channels with Noisy Feedback

Finally, we now consider the case of channels with noisy feedback *i.e.,* $\sigma_{fb}^2 > 0$. While the assumption of noiseless feedback is easier to study, most practical feedback channels suffer from noise even when the receiver sending the feedback operates at high power. Because of this limitation, neither the SK nor GN schemes perform well. To address this, [6] introduces a linear feedback scheme that is implemented as an inner code to a concatenated code, which was found to be asymptotically optimal within the linear family of codes under AWGN forward channel [30]. More recently, [7] proposed using dynamic programming to improve the performance, which turns out to be a generalized version of SK. However, despite these improvements, the linearity of these schemes severely limits the performance that can be achieved.

Recalling the architecture of deep-learning-based schemes introduced in Section V, one of the interesting properties of these codes is their robustness to noise in the feedback channel. By taking advantage of the general architecture, injecting noise into the feedback channel during training is straightforward to make the encoder-decoder robust. Hence, learning-based schemes can help design a practically realizable class of codes that can be trained in a data-driven fashion and can provide gains in noiseless and noisy feedback settings.

For training LIGHTCODE on noisy feedback channels, empirically, we found that selecting the input features as the previous encoder outputs x_1 and the cumulative noise $n_i + \tilde{n}_i$ separately works better than directly passing the feedback $\tilde{y}_i$ from previous rounds, which was the choice for noiseless feedback. Except for the change in input feature, the rest of the architecture and training details remain the same as the noiseless feedback setting. In Fig. 7, we compare the performance of rate $3/9$ LIGHTCODE against existing schemes, for a feedback SNR of 20 dB. We see that LIGHTCODE exhibits similar performance compared to GBAF while still utilizing only a fraction of the number of parameters.

VIII. ANALYSIS

As evident from results in Section VII, symbol-by-symbol neural coding schemes such as LIGHTCODE can achieve performance comparable to block coding schemes such as GBAF. Consequently, we examine GBAF and LIGHTCODE in greater detail to systematically analyze their architecture, training process, and performance, with the aim of identifying the key factors contributing to their performance. Furthermore, we compare the memory and computational complexity of LIGHTCODE with existing schemes to quantify the gains. Finally, we provide an interpretation of the encoded representations of LIGHTCODE.

A. Ablation studies on GBAF

We investigate the contribution of the self-attention mechanism to the performance of GBAF by performing an ablation study to understand the compute vs performance trade-off better. First, the self-attention mechanism is disabled, and it is observed that this has no significant effect on the BLER performance. Next, both self-attention and positional encoding blocks are disabled, and the performance remains approximately the same. These results, provided in Table IV, demonstrate a surprising observation that the self-attention and PE

modules, which are responsible for cross-symbol block coding, contribute only marginally to the performance of GBAF. This brings into question the value of block encoding and motivates us to find simpler designs that perform only symbol-by-symbol coding while providing competing performance.

SNR (dB)	GBAF	GBAF (no attn)	GBAF (no attn, no PE)
-1.5	9.8e-5	7.5e-5	2.6e-5
-1.0	2.6e-9	1.2e-9	2.7e-9
0.0	5.4e-10	6.6e-10	8.1e-10

TABLE IV: Effect of block coding on performance of GBAF for rate $3/9$ code with noiseless feedback. Positional encoding and self-attention modules have no noticeable effect on BLER performance.

B. Scaling laws: batch size

In section Section VII, we have demonstrated that it is possible to achieve performance comparable to GBAF with fewer parameters and lower compute complexity. We also hypothesized in Section VI-B that it is important to train the encoder using a very large batch size to accurately capture the statistics of the distribution so that the available power can be optimally allocated to the symbols with error at the receiver while reducing the power to the remaining symbols in a batch. In deep learning literature, it is well known that increasing the batch size can noticeably improve the performance of the neural network model [31]. To better understand the effect of batch size on LIGHTCODE, we perform a systematic study by training LIGHTCODE with different batch sizes. As noted in Fig. 8, at a batch size of 1.5×10^3, LIGHTCODE has a BLER of 3.7×10^{-9}, similar to that of GBAF (w/o BU). However, the BLER drops to 4×10^{-10}, outperforming GBAF when the batch size increases to 5×10^4 and beyond. We would like to note that for each batch size, hyperparameters such as learning rate have been optimized, and the model is trained until saturation to ensure the best possible performance.

Remark 1. Using a lightweight network with a small number of parameters makes it feasible to train with a very large batch size, resulting in a significantly lower error rate.

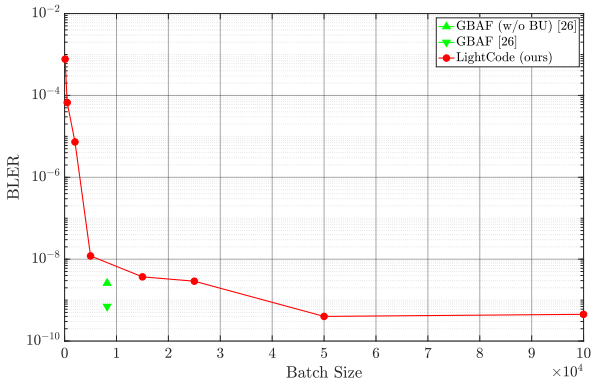


Fig. 8: BLER performance of LIGHTCODE with with respect to training batch size for rate $3/9$ code at SNR -1.0 dB with noiseless feedback. Performance improves significantly with respect to batch size, surpassing the performance of GBAF at batch size $> 5 \times 10^4$.

C. Complexity and Throughput

We compare the total number of parameters in the encoder and decoder for a rate $3/9$ LIGHTCODE against GBAF and Robust Coding within Table V. LIGHTCODE reduces the parameter count by more than $10\times$ compared to GBAF and Robust Coding, resulting in notable savings in memory.

Scheme	Total # parameters
GBAF [26]	9.1×10^4
Robust Coding [24]	8.6×10^4
LIGHTCODE (ours)	$\mathbf{7.3 \times 10^3}$

TABLE V: Rate $3/9$ LIGHTCODE requires $< 1/10^{\text{th}}$ the number of parameters compared to GBAF and Robust Coding.

To compare the throughputs, we measure the inference time. We define the encoder throughput T_E as the number of message blocks or symbols encoded per second and the decoder throughput T_D as the number of message blocks or symbols decoded per second. Formally, for a rate K/D code, the throughput can be defined as

$$T_{EK} = (K/D)T_E \quad \text{(bits/sec)}, \tag{21}$$

$$T_{DK} = (K/D)T_D \quad \text{(bits/sec)}. \tag{22}$$

We measure the throughput in CPU mode on a `AMD Ryzen Threadripper PRO 5975WX 32-Cores` processor and in GPU mode using an `NVIDIA GeForce RTX 4090`, using a batch size of $10^5 - 10^7$. LIGHTCODE provides up to $10\times$ higher encoding and decoding throughput than GBAF. Further, LIGHTCODE provides up to $171\times$ higher decoding throughput in CPU mode and up to $10\times$ higher throughput in GPU mode compared to Robust Coding, providing significant gains in latency, as shown in Table VI.

We note here that comparing throughputs for classical schemes against deep learning-based schemes is not straightforward and depends heavily on the implementation. Our implementation of LIGHTCODE uses `PyTorch` libraries, whereas POWERBLAST and other classical schemes were implemented using `NumPy` libraries. Further, a significant computational bottleneck for the classical schemes is the necessity for demodulation after each round, the speed of which is heavily influenced by the implementation methodology. Hence, we restrict our comparison to the family of deep learning codes, where the latency primarily originates from the architecture complexity and the total number of parameters, and a fair comparison is feasible. However, it is clear that the total number of numerical operations in classical schemes is significantly lower than that of deep-learning-based schemes, and we acknowledge that the throughput of classical schemes can be significantly higher.

Complexity vs. BLER trade-off: While analytically quantifying the complexity of deep learning-based channel coding schemes with respect to target BLER would be very interesting, it is highly non-trivial and a widely open problem. Thus, we instead conduct an empirical study to analyze how the complexity of LIGHTCODE scales with target BLER. Empirically, we observed that reducing the complexity of the decoder while maintaining the same complexity for the encoder provided the best trade-off in performance vs complexity. In this

Scheme	Enc (CPU)	Dec (CPU)	Enc (GPU)	Dec (GPU)
GBAF [26]	1.6×10^5	1.7×10^6	4.4×10^8	3.3×10^9
Robust Coding [24]	2.0×10^5	7.6×10^4	4.1×10^8	2.4×10^9
LIGHTCODE (ours)	$\mathbf{1.5 \times 10^6}$	$\mathbf{1.3 \times 10^7}$	$\mathbf{2.9 \times 10^9}$	$\mathbf{3.1 \times 10^{10}}$

TABLE VI: Throughput (symbols/sec) comparison. Rate $3/9$ LIGHTCODE achieves up to $10\times$ higher decoding throughput compared to GBAF and up to $171\times$ higher decoding throughput in CPU mode compared to Robust Coding.

experiment, we vary the target BLER and empirically find the required encoder-decoder dimensions for a rate $3/9$ code at a forward SNR of -1.0 dB and a noiseless feedback channel. The results for the same are shown in Table VII, demonstrating an almost linear degradation of BLER in log scale with the number of parameters.

BLER	Enc dimension	Dec dimension	Total # params
4.5×10^{-10}	32	32	7.3×10^3
3.4×10^{-9}	32	16	4.7×10^3
5.2×10^{-9}	32	12	4.3×10^3
2.8×10^{-8}	32	8	3.9×10^3

TABLE VII: Complexity vs BLER trade-off for rate $3/9$ code at a forward SNR of -1.0 dB and noiseless feedback. BLER (in log scale) degrades almost linearly with a decrease in the number of parameters.

D. *Interpretation of* LIGHTCODE

By independently encoding sub-blocks of length $K = 3$ and limiting to a symbol-by-symbol strategy, LIGHTCODE allows for better interpretability and analysis of the learned encoder representations compared to block coding schemes such as GBAF. By analyzing the power allocation and relation between encoder output and feedback from previous rounds, we draw connections between LIGHTCODE and POWERBLAST.

1) Power distribution: A key contributing factor to the superior performance of POWERBLAST compared to SK is the discrete-symbol scheme in the final round of communication. In the high-SNR regime, we are only interested in the error in the PAM index of the decoded symbol with respect to the original symbol; this will result in a sparse distribution where most of the samples are 0. Thus, a majority of the available power is naturally allocated to the symbol locations with non-zero error. Surprisingly, we find similar behavior for LIGHTCODE towards the final rounds of communication where the error is sparse.

To test this hypothesis, we choose a moderately sparse error regime for ease of analysis. In Fig. 9, we plot the power distribution of the encoder output in round 7 for rate $3/9$ code at SNR -1.0 dB and noiseless feedback by randomly sampling 50 symbols. On the X-axis, we see the sample number, and on the Y-axis, we plot the magnitude of error in the integer PAM index of the estimate after round 6 and compare it against the magnitude of encoder output in round 7. Note that a difference in index of 1 corresponds to a magnitude of 2 in the un-normalized PAM from Eqn. 6. It is evident from Fig. 9 that the highest power is allocated to the symbols with error in the estimate.

2) Interpreting the encoder: In Fig. 10, we plot the output of the encoder in round 2 with respect to the feedback in round 1. The encoder is approximately transmitting a linearly scaled version of noise from round 1. Interestingly, for the symbols

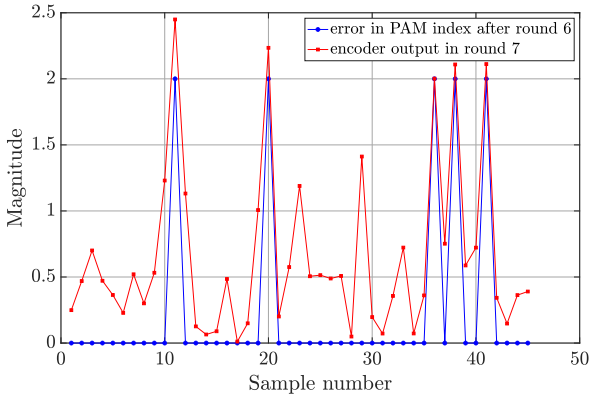


Fig. 9: LIGHTCODE allocates more power to the symbols with error in estimate from the previous round, improving the overall probability of decoding.

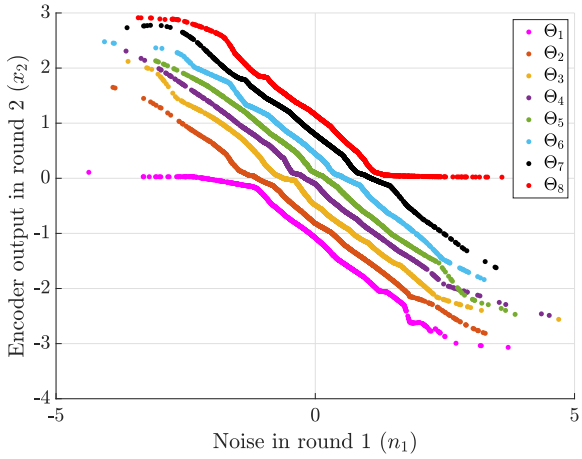


Fig. 10: Output of encoder in round 2 (x_2) shows that the encoder is directly compensating for the noise experienced in round 1 (n_1).

on the boundary of the constellation embeddings, the encoder does not need to transmit any data when the noise favors the ground truth, unlike in the SK scheme, where all noise needs to be corrected. For instance, consider Θ_1, mapped to the symbol at the boundary on the left, where a negative noise in round 1 is favored. And Θ_8 is mapped to the symbol at the boundary on the right, where a positive noise in round 1 is favored. In such scenarios, the transmit power saved here can be reallocated to other symbols, improving the overall decoding error, which is only possible because of the non-linear activation functions.

Visualization of the encoder output beyond round 2 is difficult as the number of inputs to the encoder increases linearly with the number of rounds of communication. Alternatively, we test the linear dependency between x_{i+1} and

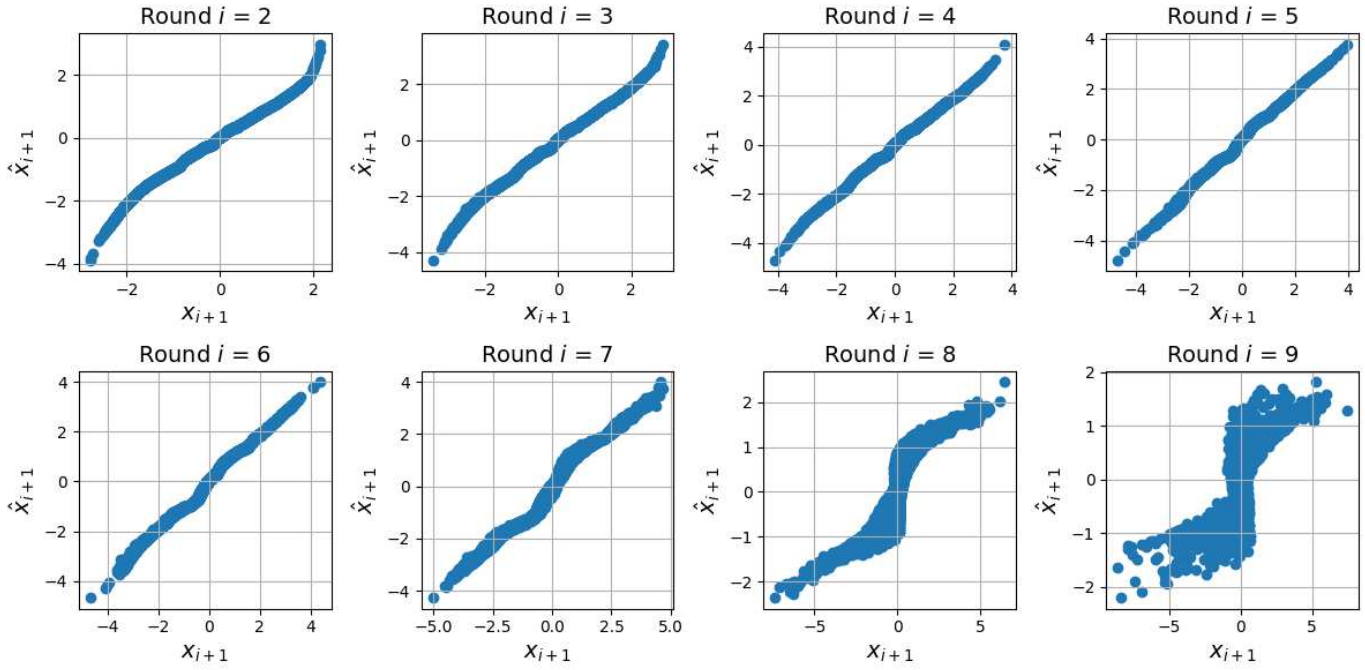


Fig. 11: Comparison of linear approximation (Y-axis) vs true encoder output (X-axis). As the rounds progress, the linearity of the relation between encoder output and the feedback from previous rounds breaks, making it harder for analytical schemes to perform well.

$\{x_1, \ldots, x_i, n_1, \ldots, n_i\}$ by performing a linear regression as

$$\hat{x}_{i+1} = \sum_{j=1}^{i} \alpha_j x_j + \beta_j n_j + c, \tag{23}$$

where α_j and β_j are regression coefficients and c is the intercept found using `LinearRegression` toolbox in `sklearn` over 10^6 samples. To make the analysis tractable, we build one linear regression model for each symbol in the PAM constellation. In Fig. 11, we plot the predicted output using linear regression vs. true encoder output when the input PAM symbol index is $\Theta = 1$, for rounds $j = 2$ to 9. It is evident that as the round number increases, the relation becomes highly non-linear, and hence, classical schemes such as SK and GN as other linear schemes [6], [7] fail to model these dependencies.

We note that recent results in [32], [33] show the possibility of using post-hoc interpretability techniques to analyze the deep-learning-based codes and learn analytical approximations for the learned codes. Making progress in this direction is an interesting avenue for future work.

IX. Conclusion

In this work, we address the problem of designing lightweight coding schemes for channels with feedback. First, we propose an analytical scheme, PowerBlast, that can be viewed as a combination of Schalkwijk-Kailath and Gallager-Nakiboğlu schemes. Using a hybrid strategy and taking advantage of the discrete nature of the signal in the final round, PowerBlast noticeably outperforms both SK and GN, providing a performance that competes with current deep learning schemes in regions of high-SNR.

Next, we propose a lightweight deep-learning-based scheme, LightCode, that can achieve state-of-the-art BLER performance while using less than $1/10^{\text{th}}$ the parameters and compute complexity compared to existing deep-learning-based schemes. By limiting to a symbol-by-symbol strategy and carefully designing the feature extractor using skip connections, combined with a training strategy that uses a very large batch size of 10^5, LightCode achieves a BLER up to $\sim 10^{-10}$.

Additionally, with the help of systematic ablation studies, we have demonstrated that the self-attention module in the transformer-based GBAF code has very little effect on the BLER performance, demonstrating the limited benefit of block coding in this regime.

Further, we interpret the LightCode to show that power distribution in the sparse error regime of LightCode is similar to that of PowerBlast, where a majority of the available power is allocated to the symbols with error. Finally, we also perform a linear regression on the encoder outputs and the feedback from previous rounds. Our findings show that although the early stages of communication exhibit a linear relationship, it becomes non-linear towards the later stages, underscoring the importance of employing deep-learning-based non-linear coding techniques to attain optimal performance in regions of extremely low SNR.

Acknowledgment

This work was partly supported by ARO Award W911NF2310062, ONR Award N000142412542, NSF Award CNS-2008824, and the 6G@UT center within the Wireless Networking and Communications Group (WNCG) at the University of Texas at Austin.

REFERENCES

[1] C. Shannon, "The zero error capacity of a noisy channel," *IRE Transactions on Information Theory*, vol. 2, no. 3, pp. 8–19, 1956.

[2] J. Schalkwijk and T. Kailath, "A coding scheme for additive noise channels with feedback–I: No bandwidth constraint," *IEEE Transactions on Information Theory*, vol. 12, no. 2, pp. 172–182, 1966.

[3] J. Schalkwijk, "A coding scheme for additive noise channels with feedback–II: Band-limited signals," *IEEE Transactions on Information Theory*, vol. 12, no. 2, pp. 183–189, 1966.

[4] R. G. Gallager and B. Nakiboğlu, "Variations on a theme by Schalkwijk and Kailath," *IEEE Transactions on Information Theory*, vol. 56, no. 1, pp. 6–17, 2009.

[5] A. Ben-Yishai and O. Shayevitz, "Interactive schemes for the AWGN channel with noisy feedback," *IEEE Transactions on Information Theory*, vol. 63, no. 4, pp. 2409–2427, 2017.

[6] Z. Chance and D. J. Love, "Concatenated coding for the AWGN channel with noisy feedback," *IEEE Transactions on Information Theory*, vol. 57, no. 10, pp. 6633–6649, 2011.

[7] R. Mishra, D. Vasal, and H. Kim, "Linear coding for AWGN channels with noisy output feedback via dynamic programming," *IEEE Transactions on Information Theory*, 2023.

[8] J. M. Ooi and G. W. Wornell, "Fast iterative coding techniques for feedback channels," *IEEE Transactions on Information Theory*, vol. 44, no. 7, pp. 2960–2976, 1998.

[9] A. G. Perotti, B. M. Popovic, and A. R. Safavi, "Accumulative iterative codes based on feedback," *arXiv preprint arXiv:2106.07415*, 2021.

[10] S. K. Ankireddy, S. A. Hebbar, Y. Jiang, P. Viswanath, and H. Kim, "Compressed Error HARQ: Feedback communication on noise-asymmetric channels," in *2023 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2023, pp. 1160–1165.

[11] J. Griffin, P. Yuan, P. Popovski, K. R. Duffy, and M. Médard, "Code at the receiver, decode at the sender: Grand with feedback," in *2023 IEEE Information Theory Workshop (ITW)*. IEEE, 2023, pp. 341–346.

[12] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein, "On the expressive power of deep neural networks," in *international conference on machine learning*. PMLR, 2017, pp. 2847–2854.

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

[14] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 119–131, 2018.

[15] S. K. Ankireddy and H. Kim, "Interpreting neural min-sum decoders," in *ICC 2023-IEEE International Conference on Communications*. IEEE, 2023, pp. 6645–6651.

[16] S. A. Hebbar, R. K. Mishra, S. K. Ankireddy, A. V. Makkuva, H. Kim, and P. Viswanath, "TinyTurbo: Efficient Turbo Decoders on Edge," in *2022 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2022, pp. 2797–2802.

[17] S. A. Hebbar, V. V. Nadkarni, A. V. Makkuva, S. Bhat, S. Oh, and P. Viswanath, "CRISP: Curriculum based Sequential neural decoders for Polar code family," in *International Conference on Machine Learning*. PMLR, 2023, pp. 12 823–12 845.

[18] S. A. Hebbar, S. K. Ankireddy, H. Kim, S. Oh, and P. Viswanath, "DeepPolar: Inventing Nonlinear Large-Kernel Polar Codes via Deep Learning," in *International Conference on Machine Learning*. PMLR, 2024, pp. 18 133–18 154.

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

[20] S. K. Ankireddy, S. A. Hebbar, H. Wan, J. Cho, and C. Zhang, "Nested Construction of Polar Codes via Transformers," in *2024 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2024, pp. 1409–1414.

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

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

[23] M. B. Mashhadi, D. Gunduz, A. Perotti, and B. Popovic, "DRF codes: Deep SNR-robust feedback codes," *arXiv preprint arXiv:2112.11789*, 2021.

[24] J. Kim, T. Kim, D. Love, and C. Brinton, "Robust non-linear feedback coding via power-constrained deep learning," in *International Conference on Machine Learning*. PMLR, 2023, pp. 16 599–16 618.

[25] Y. Shao, E. Ozfatura, A. Perotti, B. Popovic, and D. Gündüz, "Attention-code: Ultra-reliable feedback codes for short-packet communications," *IEEE Transactions on Communications*, 2023.

[26] E. Ozfatura, Y. Shao, A. G. Perotti, B. M. Popović, and D. Gündüz, "All you need is feedback: Communication with block attention feedback codes," *IEEE Journal on Selected Areas in Information Theory*, vol. 3, no. 3, pp. 587–602, 2022.

[27] P. Elias, "Channel capacity without coding," in *Lectures on Communication System Theory*, E. Baghdady, Ed. New York: McGraw Hill, 1961, quarterly progress report, MIT Research Laboratory of Electronics, Oct 15 1956.

[28] Huawei-HiSilicon, "Performance evaluation of ldpc codes for nr embb data," 3GPP RAN1 meeting 90, 3GPP, Sophia Antipolis, France, Aug 2017, [Online]. Available: https://www.3gpp.org/dynareport?code=TDocExMtg--R1-90--17073.htm.

[29] Y. Polyanskiy, H. V. Poor, and S. Verdú, "Channel coding rate in the finite blocklength regime," *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2307–2359, 2010.

[30] M. Agrawal, D. J. Love, and V. Balakrishnan, "An iteratively optimized linear coding scheme for correlated Gaussian channels with noisy feedback," in *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2011, pp. 1012–1018.

[31] Y. Bahri, E. Dyer, J. Kaplan, J. Lee, and U. Sharma, "Explaining neural scaling laws," *arXiv preprint arXiv:2102.06701*, 2021.

[32] N. Devroye, N. Mohammadi, A. Mulgund, H. Naik, R. Shekhar, G. Turán, Y. Wei, and M. Žefran, "Interpreting deep-learned error-correcting codes," in *2022 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2022, pp. 2457–2462.

[33] Y. Zhou, N. Devroye, G. Turán, and M. Žefran, "Interpreting Deepcode, a learned feedback code," in *2024 IEEE International Symposium on Information Theory (ISIT)*, 2024, pp. 1403–1408.

APPENDIX A
LONGER BLOCKLENGTHS FOR LIGHTCODE

While the input and output dimensions for LIGHTCODE can be increased to support learning longer blocklengths, converging to a good solution is harder because of the curse of dimensionality. In Table VIII, we provide the performance of LIGHTCODE trained for $K = 6$ at different complexities of the encoder-decoder at forward SNR = -1.0 and noiseless feedback.

BLER	Enc dimension	Dec dimension	**Total # params**
3.7×10^{-4}	32	32	7.3×10^{3}
7.2×10^{-5}	128	64	1.09×10^{5}
2.4×10^{-6}	128	128	4.1×10^{5}

TABLE VIII: BLER for K=6 at forward SNR of -1.0 dB and noiseless feedback.

We begin by comparing the performance of LIGHTCODE for $K = 6$ while maintaining the same encoder-decoder dimension used for $K = 3$, which is 32. From Fig. 5, the BLER of LIGHTCODE trained with $K = 3$ at a forward SNR of -1.0 dB and noiseless feedback is 4.5×10^{-10}. Hence, by transmitting 6 bits as two sub-blocks, the BLER would be given by $1 - (1 - 4.5 \times 10^{-10})^2 = 9 \times 10^{-10}$. But, as seen from Table VIII, the BLER performance for LIGHTCODE trained with $K = 6$ and a hidden dimension of 32 is orders of magnitude higher, 3.7×10^{-4}. Next, we increase the hidden dimensions to 128, increasing the total number of parameters by $100\times$, and the performance is still much worse than LIGHTCODE trained with $K = 3$.

APPENDIX B
ENCODER DECODER ARCHITECTURE

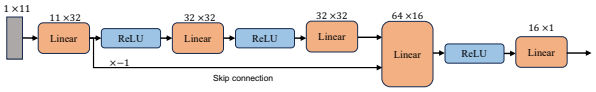


Fig. 12: Encoder

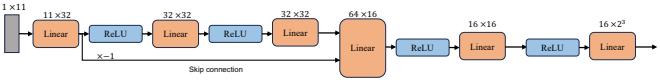


Fig. 13: Decoder

APPENDIX C
ALGORITHM FOR GALLAGER-NAKIBOĞLU

Algorithm 4: Gallager-Nakiboğlu coding scheme

Input: Message symbol Θ, number of rounds D, noise variance σ_{ff}^2

Round 1: Tx: Power normalization: $X_1 = \sqrt{P_1}\Theta$;
Transmit: $Y_1 = X_1 + N_1$;
Rx: Send Y_1 as feedback to Tx ;
Round 2: Tx: Power normalization: $X_2 = \frac{\sqrt{P_2}U_2}{\sigma_2}$;
$U_2 = N_1$ and $\sigma_2 = \sigma_{ff}$;
Transmit: $Y_2 = X_2 + N_2$;
Rx: Compute the LMMSE estimate of transmit symbol $\mathbb{E}[U_2|Y_2] = \frac{\sigma_2\sqrt{P_2}Y_2}{1+P_2}$;
Update the estimate as $\hat{X}_1 = X_1 - \mathbb{E}[U_2|Y_2]$
`/* Tx sends the error in estimate` $\epsilon_2 = \mathbb{E}[U_2|Y_2] - U_2$ `over` $D-2$ `rounds */`
while $3 \leq i \leq D-1$ **do**
  Tx: Compute the error in estimate of previous round $U_i = \mathbb{E}[U_{i-1}|Y_{i-1}] - U_{i-1}$;
  Power normalization: $X_i = \frac{\sqrt{P_2}U_i}{\sigma_i}$, $\sigma_i^2 = \frac{\sigma_{i-1}^2}{1+S_{i-1}}$;
  Transmit: $Y_i = X_i + N_i$;
  Rx: Send feedback as LMMSE estimate of transmit symbol $\mathbb{E}[U_i|Y_i] = \frac{\sigma_i\sqrt{P_2}Y_i}{1+P_2}$;
  Update the estimate as $\hat{X}_1 = X_1 - \sum_{j=2}^{j=i}\mathbb{E}[U_j|Y_j]$
Decoding after round $D-1$: Map $\hat{\Theta}_{D-1} = \frac{\hat{X}_1}{\sqrt{P_1}\eta}$ to the closest symbol in the 2^K PAM constellation.
`/* High-SNR scheme for rounds` D `*/`
Round D: Tx: Compute the difference in PAM indices $U = \hat{M} - M$, where $\hat{M}$ and M correspond to the integer index from PAM constellation for $\hat{\Theta}$ and Θ respectively;
Transmit: $Y_D = X_D + N_D$;
Final Decoding: Use ML decoder to detect $\hat{U}$ and detect original PAM signal using $\hat{U}$.