Interpreting Deepcode, a learned feedback code

Y. Zhou, N. Devroye, Gy. Turán*, and M. Žefran
University of Illinois Chicago, Chicago, IL, USA
* HUN-REN-SZTE Research Group on AI
{yzhou238, devroye, gyt, mzefran}@uic.edu

2024 IEEE International Symposium on Information Theory (ISIT) | 979-8-3503-8284-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISIT57864.2024.10619390

***Abstract*—Deep learning methods have recently been used to construct non-linear codes for the additive white Gaussian noise (AWGN) channel with feedback. However, there is limited understanding of how these black-box-like codes with many learned parameters use feedback. This study aims to uncover the fundamental principles underlying the first deep-learned feedback code, known as Deepcode, which is based on an RNN architecture. Our interpretable model based on Deepcode is built by analyzing the influence length of inputs and approximating the non-linear dynamics of the original black-box RNN encoder. Numerical experiments demonstrate that our interpretable model – which includes both an encoder and a decoder – achieves comparable performance to Deepcode while offering an interpretation of how it employs feedback for error correction.** [1]

I. Introduction

Although it is known that feedback does not increase the capacity of memoryless channels, it can reduce the coding complexity and improve reliability. Moreover, for finite blocklengths and fixed rates, constructing codes that achieve the smallest bit or block error rate remains open. This has motivated the construction of deep-learned error-correcting codes (DL-ECC) [1]–[8] in which the encoding and decoding functions are parameterized by a (usually) very large number of parameters in a neural network architecture, which are then learned by adjusting these to numerically minimize a loss function. This approach differs markedly from previous feedback coding schemes [9]–[13] which are analytically constructed.

Due to the increasing model complexity, it is difficult to understand how these codes accomplish error correction, and this leads us to perceive learned models as "black boxes". Such an understanding is important to a) build trust in these models, b) identify their weaknesses, and c) reveal how feedback is used, potentially pointing us in the direction (similar in spirit to [14]) of new (possibly non-linear) coding schemes. We aim to open the black box and *interpret* a prominent early example of the learned feedback encoders and decoders: Deepcode [1].

Deepcode is a recurrent neural network (RNN)-based non-linear coding scheme for AWGN channels with passive feedback. Its experimental error performance was shown [1] to outperform that of Schalkwijk-Kailath (SK) [9] and Chance-Love (CL) [12] schemes in the case of passive noisy feedback. More recently, a state propagation-based non-linear feedback code based on RNNs has emerged [8] which encompasses SK, CL, and Deepcode and is especially robust to feedback noise levels; but it is still unclear exactly how this scheme utilizes the feedback symbols for error correction.

The relevant notion of interpretation depends on context. Here we aim to provide an understanding of the deep-learned models similar to the analytically constructed feedback coding schemes. Deepcode [1] was the first to offer a limited interpretation of the underlying functioning of encoders / decoders through scatter-plots and coupling. We significantly expand the understanding of how Deepcode works here.

Some approaches have been proposed to understand RNNs [15]–[17], but none is immediately applicable in this novel DL-ECC setting. The validity of explanations can be controversial as well [18], [19]. Unlike previous classification tasks, we focus on understanding a pair of RNNs, an encoder and a decoder, which are jointly learned in the presence of channel noise, to transmit and decode exponentially many messages. Our prior work [20]–[23] has focused on interpreting TurboAE [24] – a deep-learned forward error-correcting code based on convolutional neural networks (CNNs) placed into a Turbo-code-like architecture with an interleaver, in the absence of feedback. The coding structure of the feed-forward, CNN-based model with an interleaver varied markedly from the RNN and feedback structure of Deepcode, and while some methods carry over, new techniques must be developed for interpretation in the feedback setting.

Contributions. We suggest several tools for understanding how Deepcode works, including simplifying RNN through dimension reduction and pruning, outlier analysis, architecture-based insights, influence length, and nonlinear input-output map approximations. Our interpretation efforts culminate by proposing an interpretable encoder and decoder approximation yielding an understanding[2] of how feedback is used to correct errors in Deepcode, with competitive, and sometimes even superior BER performance in noisy or noiseless feedback.

Notation: subscripts i, j represent time and bit indices, respectively. Vectors of random variables are expressed in bold, with superscripts indicating their lengths. K and N represent the lengths of message bits and codewords, respectively. The coding rate is $r = K/N$. SNR_f and SNR_{fb} represent the forward and feedback channel Signal-to-Noise Ratios, respectively. $\mathbb{R}^n$ represents n-dimensional real vectors. $\mathbb{F}_2$ denotes the finite field with elements 0 or 1 and addition

[1]This work was supported by NSF under awards 1900911, 2217023, and 2240532, and by the AI National Laboratory Program (RRF-2.3.1-21-2022-00004). The contents of this article are solely the responsibility of the authors and do not necessarily represent the official views of the NSF.

[2]An explicit scheme with a small number of learnable parameters and an interpretation of how error-correction is performed at the encoder and decoder.

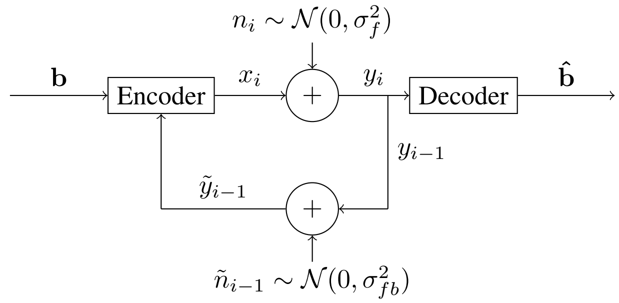

Fig. 1: AWGN channel with passive noisy feedback.

$\bigoplus$. Function $\mathbb{I}(x) = 1$ if the argument is true (non-negative), 0 else.

II. System Model

Fig. 1 shows the AWGN channel with passive noisy feedback. The message bits $\mathbf{b} \in \mathbb{F}_2^K$ are sent through N time steps. At each time instant $i \in \{1, \ldots, N\}$, the forward channel is characterized by $y_i = x_i + n_i$, where $x_i \in \mathbb{R}$ is the transmitted symbol, and $n_i \sim \mathcal{N}(0, \sigma_f^2)$ is the Gaussian noise, independent and identically distributed (iid) across time steps. The receiver sends channel outputs back to the transmitter with one unit delay through the noisy channel: $\tilde{y}_{i-1} = y_{i-1} + \tilde{n}_{i-1}$. Here, $\tilde{n}_{i-1} \sim \mathcal{N}(0, \sigma_{fb}^2)$ is also an iid Gaussian noise.

Encoding functions f_i map the message bits and feedback to transmitted codewords (joint coding and modulation), denoted as $x_i = f_i(\mathbf{b}, \tilde{\mathbf{y}}^{i-1})$. The decoding function g maps the channel outputs to estimated message bits $\widehat{\mathbf{b}} = g(\mathbf{y}^N) \in \mathbb{F}_2^K$. We impose an average power constraint $\frac{1}{N}\mathbb{E}\left[\|\mathbf{x}\|_2^2\right] \leq 1$ where $\mathbf{x} = (x_1, \cdots, x_N)$. Performance is measured through the bit error rate $BER = \frac{1}{K}\sum_{i=1}^{K} \mathbb{P}(b_i \neq \widehat{b}_i)$.

Deepcode [1] is a DL-ECC designed for this AWGN channel with rate 1/3 (Fig. 2). The encoding scheme contains two phases. In the first phase, the K message bits $\mathbf{b}$ under BPSK modulation $\mathbf{c} = 2\mathbf{b} - 1 \in \{-1, 1\}^K$ are transmitted, uncoded, through the channel, yielding the feedback $\tilde{\mathbf{y}} = [\tilde{y}_1, \cdots, \tilde{y}_K]$. The encoder stores noises in the first phase $\mathbf{n} + \tilde{\mathbf{n}} = \tilde{\mathbf{y}} - \mathbf{c} \in \mathbb{R}^K$ for later use. In the second phase, the encoder uses a directional RNN with a $\tanh$ activation function and linear combination layer to sequentially generate $2K$ parity bits $c_{i,1}$ and $c_{i,2}$ where $i \in \{1, \ldots, K\}$. At time instant i, we define the input to the RNN as $\mathbf{P}_i = \left[b_i, n_i + \tilde{n}_i, n_{i-1,1} + \tilde{n}_{i-1,1}, n_{i-1,2} + \tilde{n}_{i-1,2}\right]$ which contains the current message bit (b_i), the noise in the first phase ($n_i + \tilde{n}_i$), and the feedback noises resulting from the transmission of parity bits in the second phase ($n_{i-1,j} + \tilde{n}_{i-1,j} = \tilde{y}_{i-1,j} - c_{i-1,j}$, $j \in \{1, 2\}$). The codewords are denoted as $\mathbf{X}^N = \left[c_1, \cdots, c_K, c_{1,1}, c_{1,2}, c_{2,1}, c_{2,2}, \cdots, c_{K,1}, c_{K,2}\right]$. The decoding scheme uses a two-layered bidirectional gated recurrent unit (GRU) to estimate message bits $\widehat{\mathbf{b}}$ from the noisy codewords $\mathbf{Y}^N$. In what follows, we initially focus on the noiseless feedback case, where $\tilde{n}_i = \tilde{n}_{i-1,1} = \tilde{n}_{i-1,2} = 0$. We later extend our scheme to noisy feedback.

In Deepcode, zero padding is applied to the last message bit to reduce the error. Codewords are assigned different learned power weighting parameters $\mathbf{w}$ and $\mathbf{a}$ to balance errors. Finally, the codewords are normalized to satisfy the power constraint (Fig. 2). The encoder and decoder are trained jointly to minimize the binary cross-entropy (BCE). *"TensorFlow Deepcode"* will refer to the original Deepcode implementation with $N_h = 50$ hidden states. We implemented our *"Pytorch Deepcode"* based on the TensorFlow Deepcode[3].

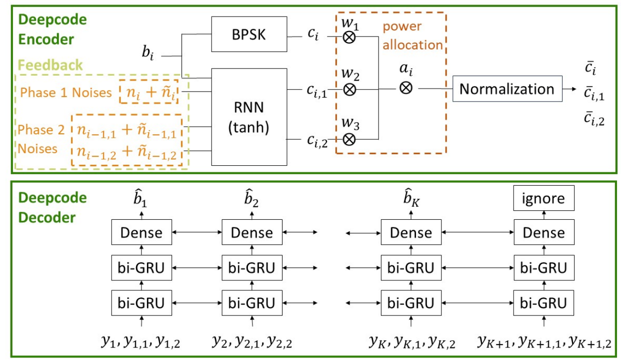

Fig. 2: Deepcode encoder (above) and decoder (below). Here, $i \in \{1, \ldots, K+1\}$ because the message bits are padded with a zero. When $i = 1$, the initial value for phase 2 noises is 0.

III. Model Reduction

Since Deepcode has $N_h = 50$ hidden states, the model becomes relatively complex, with over $65,000$ parameters. This makes direct interpretation challenging. In this section, we perform model reduction (through dimension reduction and pruning) to find a model of much smaller dimension / fewer parameters without sacrificing performance.

A. Dimension Reduction

The term "dimension" here refers to the number of hidden states in both the encoder and decoder. We tried two known model reduction techniques (Appendix A-A of [25]), but their BER performance was much poorer than directly re-training models of smaller dimension. In Deepcode's linear combination with 50 hidden states (which generate parity bits), only around 13 out of the 50 trained weights had significant absolute values; we thus pursued training models of dimension $\leq$ 13 directly using the following technique: we initially trained Deepcode with reduced dimension using a block length of 100 without power allocation and then re-trained with a block length of 50 with power allocation inspired by [4]. The corresponding performance is shown in Fig. 3. The parameters are trained at various forward SNRs, defined as $-10\log_{10}\sigma_f^2$. It turns out that a reduced dimension can approximate the performance of Deepcode with 50 hidden states quite effectively. When the SNR_f is low, larger dimension codes exhibit better performance. Conversely, when the SNR_f is high, smaller dimensions perform better. This may occur because training a large dimension becomes challenging in the presence of a small number of errors (small BER, small BCE). We expand on this in Section V.

[3]Code is available at https://github.com/zyy-cc/Deepcode-Interpretability and https://github.com/hyejikim1/Deepcode.

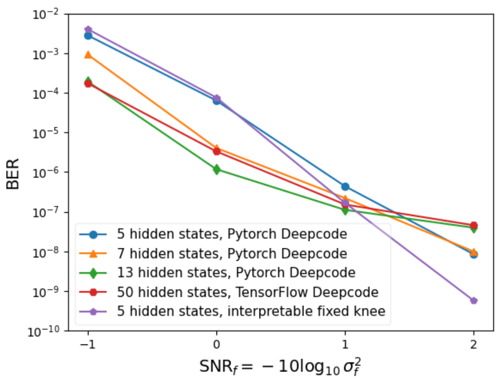

Fig. 3: BER performance vs. forward SNR, different dimensions (number of hidden states in RNNs), noiseless feedback

B. Pruning

We observed that in the trained parameters, some values have significantly smaller magnitude than others. Our next approach was simple: to prune (set to zero) the weights with small absolute values. It turns out that pruning smaller weights (up to a certain amount) has minimal impact on the model's BER performance, while yielding a more compact and efficient model (see Appendix A-B in [25]).

In the following sections, *"reduced Deepcode (N_h)"* refers to the model obtained by retraining from scratch with N_h hidden states in PyTorch. This process includes pruning the encoder and retraining the decoder. To ensure validity, we generate scatter plots illustrating the relationship between parity bits and phase 1 noises (Appendix E of [25]), demonstrating consistency with Deepcode [1]. We focus on the simplest reduced Deepcode (5) later.

IV. Opening the Black Box

In this section, we open the black box of Deepcode's encoder by looking first at the influence length of the learned encoder RNNs based on different input perturbations, and then dig further to understand the actual function learned through the use of scatter plots and regression.

A. Influence Length

For a specific input β, we define the expected L_1 difference of parity bits at time instant i as $L_{i,\beta,\Delta} = \mathbb{E}\|f(\mathbf{P}_i) - f(\mathbf{P}_i^{(\beta,\Delta)})\|_1$. Here, $\mathbf{P}_i$ is the RNN input, and the vector $\mathbf{P}_i^{(\beta,\Delta)}$ is obtained by perturbing the element β in the vector $\mathbf{P}_i$ by Δ. E.g., flipping the message bit results in $\mathbf{P_i}^{(b_i,\Delta)} = \left[b_i \bigoplus 1, n_i, n_{i-1,1}, n_{i-1,2}\right]$, where $\Delta = 1 \in \mathbb{F}_2$ or perturbing one of the noise components gives $\mathbf{P}_i^{(n_i,\Delta)} = \left[b_i, n_i - \Delta, n_{i-1,1}, n_{i-1,2}\right]$, for $\Delta \in \mathbb{R}$ the perturbation.

The *influence length* of input β captures the number of time steps over which a change in the input affects the parity bits. Given a small value δ (set at 0.05) and a perturbation of the input at position t (randomly chosen as $t = 5$), the influence length is defined as:

$$\mathcal{L}_{\beta,\Delta} = \sum_{i=t}^{K} \mathbb{I}\left(L_{i,\beta,\Delta} > \delta \max_{k\in\{t,\dots,K\}} L_{k,\beta,\Delta}\right).$$

For each model, $\mathcal{L}_{b_i,\Delta}$ (Appendix B Table. IV [25]) remains constant, while the influence length for the noise (Table. III of Appendix B [25]) increases with Δ, but eventually levels off. Together with BER plots, we conclude that longer influence lengths are effective in addressing rare events when the noises become extremely large. Based on our experimental maximum influence lengths, it seems unnecessary to use 50 hidden states (as in the original Deepcode) to achieve comparable influence lengths and similar BER performance. As the noises follow a Gaussian distribution, the probability of exceeding $3\sigma_f$ is low. Consequently, we establish our interpretation based on small Δ deviations $\mathcal{L}_{b_i,\Delta} = \mathcal{L}_{n_i,\Delta} = 2$ and $\mathcal{L}_{n_{i,1},\Delta} = \mathcal{L}_{n_{i,2},\Delta} = 1$.

B. Nonlinear Dynamics

We now further peel open the black box and look at the specific learned RNN functions directly. The RNN of Deepcode's encoder (original or reduced) is a discrete nonlinear dynamical system $\mathbf{h}_i = \tanh(\mathbf{W_{hp}}\mathbf{P}_i + \mathbf{W_{hh}}\mathbf{h}_{i-1} + \mathbf{b})$ where $\mathbf{W_{hp}}, \mathbf{W_{hh}}, \mathbf{b}$ are the learned parameters. After pruning, the learned RNN parameters in the reduced Deepcode (5) have a special structure:

$$h_{i,p} = \begin{cases} q_{p,1}(b_i, n_i), & \text{if } p \in \{1,2,3\} \\ q_{p,2}(n_{i-1,1}, n_{i-1,2}, \mathbf{h}_{i-1}), & \text{if } p \in \{4,5\} \end{cases}$$

where $h_{i,p}$ represents the p-th element of hidden state $\mathbf{h}_i$. In particular, each of the 5 hidden states is either a non-recurrent $q_{p,1}$ function only of the message bits and phase 1 noise, or is a recurrent $q_{p,2}$ function only of the phase 2 noises and past hidden state. The parity bits are linear combinations of the hidden state elements $c_{i,j} = \sum_{p=1}^{3} \alpha_{p,j} q_{p,1}(b_i, n_i) + \sum_{p=4}^{5} \alpha_{p,j} q_{p,2}(n_{i-1,1}, n_{i-1,2}, \mathbf{h}_{i-1})$, where $\alpha_{p,j}$ are the learned coefficients. We represent the non-recurrent component of $c_{i,1}$ as q_1 and that of $c_{i,2}$ as q_1'.

Approximating the non-recurrent terms: Inspired by Deepcode [1], for the non-recurrent terms, we consider the scatter-plot (uniform message bits, phase 1 noises Gaussian) of functions q_1 and q_1' (Fig. 7 of the Appendix C in [25]). They show that functions q_1 and q_1' appear piecewise linear (PWL) for each of the two possible message bits $b_i = 0$ and $b_i = 1$ (the original TensorFlow Deepcode has a very similar shape), and hence these functions may be well approximated by two segments. We will use the indicator function $\mathbb{I}$ to represent the segments by adjusting the slopes later.

Understanding the recurrent terms through outlier analysis: Unlike the non-recurrent terms, both $h_{i,4}$ and $h_{i,5}$ depend on previous inputs. Consequently, we visualize the trajectory of the hidden states over time. We plot the black box function over Gaussian noises and true bits. From Fig. 4, we observe that $h_{i,4}$ predominantly takes the value of 1, while $h_{i,5}$ is mostly -1. Notably, $h_{i,4}$ has outliers when the past message bit is 0 (similarly $h_{i,5}$ has outliers when the bit is 1), and the noises are unusually large. In most cases, the values of $h_{i,4}$ and $h_{i,5}$ cancel each other out when forming the parity bits. However, outliers significantly impact both parity bits, signaling and enabling error correction, as further interpreted in Section V.

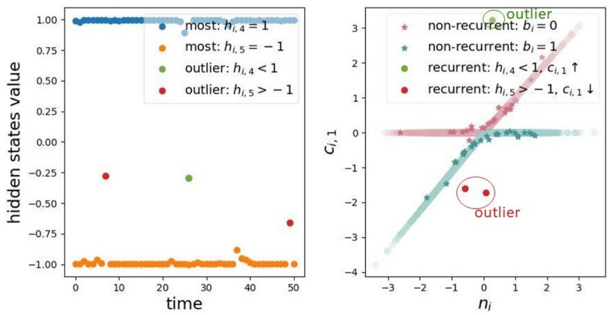

Fig. 4: Outlier values of hidden states (left) and their impact on parity bit $c_{i,1}$ (right). The outliers cause deviations in the parity bits from the regular values in the right figure.

V. Interpretable Model of a Deepcode-like Encoder and Decoder

Having built up an understanding of the learned encoder in the last sections, we now present an interpretable model built on the reduced models. By *interpretable* model we mean succinct non-linear expressions for the encoder f and decoder g that yield BER that closely resembles that of Deepcode. We consider the reduced Deepcode (5) that captures the essential dynamics of Deepcode. The ideas carry over but become more complex as the number of hidden states increases. The interpretable models confirm that the encoder tries to use the feedback to assist the decoder in error correction.

A. Encoder Interpretation

1) Reduced Deepcode (5): We suggest the following basic interpretable model with 5 hidden states:

$$c_i = 2b_i - 1 \text{ (Deepcode's phase 1 – uncoded transmission)}$$

non-recurrent — outlier analysis

$$c_{i,1} = e_1 n_i \mathbb{I}(-(2b_i - 1)n_i) - e_2 h_{i,4} - e_2 h_{i,5} \quad (1)$$

$$c_{i,2} = - e_1 n_i \mathbb{I}(-(2b_i - 1)n_i) - e_2 h_{i,4} - e_2 h_{i,5} \quad (2)$$

where e_1 and e_2 are learned coefficients.

Analysis: In the "red" portion, the parity bits $c_{i,1}$ and $c_{i,2}$ are energy efficient: if the transmitted message bit is $b_i = 0$, and the noise in the first phase added to b_i is negative, then the parity bits will not send new information about n_i (not needed as a binary detection would yield the correct estimated bit). Otherwise, the parity bits contain a scaled version of n_i. The case is similar for $b_i = 1$ and positive phase 1 noises. A similar interpertation was obtained in [1]. What is new with respect to the limited interpretation of [1] are the "blue" portions in (1) and (2). From our earlier studies, we know that feedback noises affect the outputs for 1 time unit in reduced Deepcode (5), i.e. $\mathcal{L}_{n_{i,1},\Delta} = \mathcal{L}_{n_{i,2},\Delta} = 1$. Therefore, $h_{i,4}$ and $h_{i,5}$ convey information from the last time step, as follows:

$$\text{If } b_{i-1} = 0, \text{ then } h_{i,5} = -1 \text{ and}$$
$$h_{i,4} = \tanh\left(-k_1 n_{i-1} + k_2 n_{i-1,1} - k_3 n_{i-1,2} + k_4\right)$$
$$\text{If } b_{i-1} = 1, \text{ then } h_{i,4} = 1 \text{ and}$$
$$h_{i,5} = \tanh\left(-k_1 n_{i-1} + k_2 n_{i-1,1} - k_3 n_{i-1,2} - k_4\right)$$

where k_1, k_2, k_3, and k_4 are learned coefficients.

Analysis of the blue portion: The blue portion serves as the error correction for the previous bit, and its value depends on the phase 1 and phase 2 noises from the last time step.

- *Without Outlier*: In small noise scenarios (as depicted in Fig. 5, yellow for $h_{i,4}$ or purple for $h_{i,5}$), $h_{i,4}$ (1) and $h_{i,5}$ (-1) cancel each other out in the parity functions.
- *With Outlier*: For purple values in the left and yellow on the right of Fig. 5 we have outliers:
  - $h_{i,4}$: outliers (value < 1) occur when the message bit $b_{i-1} = 0$, phase 1 noise n_{i-1} is too positive, and phase 2 noise $n_{i-1,1}$ is too negative, or $n_{i-1,2}$ is too positive.
  - $h_{i,5}$: the situation is analogous to above, with $b_{i-1} = 1$, and with the noise signs and magnitudes reversed.

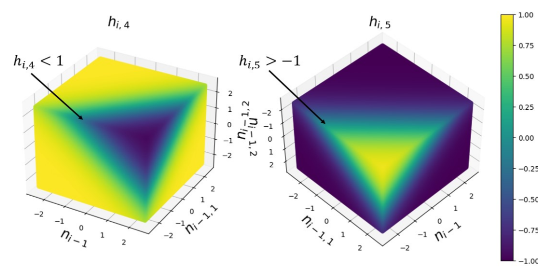

Fig. 5: Outlier analysis of the "blue" portions in (1) and (2).

2) Reduced Deepcode (7): The "enhanced" interpretable encoder (for $N_h = 7$ hidden states) is designed to address more rare events and improve overall performance relative to the $N_h = 5$ interpretable encoder. It introduces two more hidden states $h_{i,6}$ and $h_{i,7}$ to match the longer observed influence lengths. The details are in Appendix D in [25].

B. Decoder Interpretation

In the decoding process, we construct a simple non-linear code for 5 hidden states using the symmetry of the parity bits. Our decoder estimates the message bits as:

$$\begin{aligned} O_{i,j} &= \tanh(d_{j,1}\tilde{c}_i - d_{j,2}(\tilde{c}_{i,1} - \tilde{c}_{i,2}) \\ &\quad - d_{j,3}(\tilde{c}_{i+1,1} + \tilde{c}_{i+1,2}) + d_{j,4}) \\ &= \tanh(d_{j,1}\bar{c}_i + d_{j,1}n_i - d_{j,2}(\bar{c}_{i,1} - \bar{c}_{i,2}) \\ &\quad - d_{j,2}(n_{i,1} - n_{i,2}) - d_{j,3}(\bar{c}_{i+1,1} + \bar{c}_{i+1,2}) \\ &\quad - d_{j,3}(n_{i+1,1} + n_{i+1,2}) + d_{j,4}) \end{aligned} \quad (3)$$

eliminate n_i — error correction

$$\hat{b}_i = \begin{cases} 0 & \text{if } D_i < 0.5, \\ 1 & \text{if } D_i \geq 0.5. \end{cases} \quad D_i = \sigma\left(\sum_{j=1}^{N_l=5} l_j O_{i,j}\right) \quad (4)$$

where σ represents the sigmoid function, and all $\mathbf{d}$ and $\mathbf{l}$ values are learned parameters, and post power-allocation normalized parity bits are $\bar{c}_i$, $\bar{c}_{i,1}$ and $\bar{c}_{i,2}$. A linear combination of $N_l = 5$ decoding results is used in the final decision.

Here, we assume the positivity of learned parameters in the encoder and decoder, except for the $d_{j,4}$ bias; other cases are addressed in symmetrically equivalent codes.

Analysis: the "orange" ("teal") portion is used to eliminate phase 1 (2) noise in (1) and (2).

- *Phase 1 Noises*: The "orange" portion in (3) eliminates the noises added in the first phase by subtracting the current parity bits.
- *Phase 2 Noises*: The "teal" portion in (3) eliminates phase 2 noises by adding future parity bits. Without outliers, the sum is about 0. With outliers, error correction is demonstrated in Table I:
  - $h_{i+1,4}(< 1)$: Consider the scenario where the message bit $b_i = 0$; we want $O_{i,j}$ to be negative for correct detection. The outliers in $h_{i+1,4}$ will push the decoding value of $O_{i,j}$ to a negative value:

    $$-d_{j,2}(n^{\downarrow}_{i,1} - n^{\uparrow}_{i,2}) \uparrow \quad -d_{j,3}(\bar{c}^{\uparrow}_{i+1,1} + \bar{c}^{\uparrow}_{i+1,2}) \downarrow$$

  - $h_{i+1,5}(> -1)$: For $b_i = 1$, the situation is reversed, and $O_{i,j}$ needs to be positive for correct decoding.

    $$-d_{j,2}(n^{\uparrow}_{i,1} - n^{\downarrow}_{i,2}) \downarrow \quad -d_{j,3}(\bar{c}^{\downarrow}_{i+1,1} + \bar{c}^{\downarrow}_{i+1,2}) \uparrow$$

b_i	n_i	$n_{i,1}$	$n_{i,2}$	$h_{i+1,4}$	$h_{i+1,5}$	$\bar{c}_{i+1,j}$	$O_{i,j}$
0	+	−	+	< 1	-1	↑	↓
1	−	+	−	1	> -1	↓	↑

TABLE I: Error Control

C. Training

Our interpretable encoder and decoder, based on 5 hidden states in equations (1), (2), (3), and (4), along with power allocation, have parameters ($\mathbf{e}$, $\mathbf{k}$, $\mathbf{d}$, $\mathbf{w}$, $\mathbf{a}$) that we learn rather than use to approximate Deepcode's outputs. During training, we focus on a block length of $K = 50$. There are a total of 43 parameters in our interpretable model, with 12 of them ($\mathbf{w}$ and $\mathbf{a}$) associated with power allocation. We optimize all parameters to minimize BCE at different forward SNRs.

Table. II presents the BER performance when matching different interpretable / original encoders / decoders of Deepcode. The results indicate that the interpretable encoder and decoder perform comparably to the reduced code. The BER performance of the interpretable model, based on 5 hidden states across different SNR_f, is shown in Fig. 3. Our interpretable model exhibits better performance than the original Deepcode at high SNR_f but experiences decreased performance at low SNR_f due to what we believe is its limited influence length for capturing unusual sequences of noise events (e.g. longer sequences of phase 2 noises being unusually large). Moreover, we experimentally demonstrate that our encoder with 7 hidden states improves the BER performance by an order of magnitude, making it comparable to Tensorflow Deepcode (Table. V in Appendix D [25]). We are working on developing an $N_h = 7$ interpretable decoder for our $N_h = 7$ encoder.

Encoder	Decoder	BER SNR_f 0	BER SNR_f 2
original	original	$6.375e-05$	$8.417e-09$
interpret.	original	$7.616e-05$	$1.300e-09$
original	interpret.	$7.498e-05$	$5.350e-09$
interpret.	interpret.	$7.595e-05$	$5.694e-10$

TABLE II: BER Performance of interpretable models of dimension 5 (noiseless feedback).

D. Equivalent Codes

The interpretable encoder and decoder, based on 5 hidden states as discussed earlier, provides one particular example of possible codes. There are eight distinct equivalent combinations involving (1) positive or negative signs of components in the parity bits, and (2) the value selections in $h_{i,4}$ and $h_{i,5}$, which can be either 1 or -1 (details are in Appendix F of [25]). We verified (Appendix F Table. VI) that all 8 interpretable models yield comparable performance in terms of BER.

E. Noisy Feedback

After analyzing noiseless feedback, we expand our study to include noisy feedback. In the case of noisy feedback, reduced Deepcode (5) has a structure similar to that in the noiseless feedback, and all interpretation steps follow:

- PWL approximation: from the scatter points we observe that the knee point shifts as the feedback SNR (SNR_{fb}) decreases (Appendix C Fig. 8 [25]). Based on these observations, we extend our interpretable model to incorporate varying knee points and modify the non-recurrent part of the encoder $(n_i + \tilde{n}_i)\mathbb{I}(-(2b_i - 1)(n_i + \tilde{n}_i))$ slightly:

  $$\text{If } b_i = 0, (n_i + \tilde{n}_i + \lambda_1)\mathbb{I}(-(2b_i - 1)(n_i + \tilde{n}_i + \lambda_1))$$
  $$\text{If } b_i = 1, (n_i + \tilde{n}_i - \lambda_2)\mathbb{I}(-(2b_i - 1)(n_i + \tilde{n}_i - \lambda_2))$$

  where λ_1 and λ_2 are learned parameters.
- Outlier analysis: The number of outliers increases as the feedback SNR decreases. (Appendix G Fig. 13 [25]).

The BER performance is shown in Appendix G Fig. 14 [25]. With noisy feedback, the interpretable model with varying knee points performs as well as the 5 hidden states Deepcode and slightly outperforms the interpretable model with fixed knee points. This suggests that the varying knee points adapt to the feedback noises and attempt to mitigate their impact.

VI. Conclusions

We presented an interpretable model for the RNN-based Deepcode. We demonstrated the impact of feedback on decoding through outliers and showed that our interpretable model performs comparably to the original Deepcode with significantly fewer parameters, both in noiseless and noisy feedback scenarios. Notably, it outperforms Deepcode at high forward SNR. However, at low SNR_f, our interpretable model exhibits reduced performance, suggesting that the short influence length of our interpretable model limits error correction. Future research may focus on exploring the interpretation of models with longer influence lengths, and on identifying an algorithm for the construction of (analytical) feedback codes with a given influence length.

References

[1] 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.

[2] 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.

[3] M. B. Mashhadi, D. Gunduz, A. Perotti, and B. Popovic, “Drf codes: Deep snr-robust feedback codes,” *arXiv preprint arXiv:2112.11789*, 2021.

[4] Y. Shao, E. Ozfatura, A. Perotti, B. Popovic, and D. Gündüz, “Attentioncode: Ultra-reliable feedback codes for short-packet communications,” *IEEE Transactions on Communications*, 2023.

[5] 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.

[6] H. Kim, Y. Jiang, S. Kannan, S. Oh, and P. Viswanath, “Deepcode and modulo-sk are designed for different settings,” *arXiv preprint arXiv:2008.07997*, 2020.

[7] A. Ben-Yishai and O. Shayevitz, “Simple modulo can significantly outperform deep learning-based deepcode,” *arXiv preprint arXiv:2008.01686*, 2020.

[8] J. Kim, T. Kim, D. Love, and C. Brinton, “Robust non-linear feedback coding via power-constrained deep learning,” *arXiv preprint arXiv:2304.13178*, 2023.

[9] 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.

[10] O. Shayevitz and M. Feder, “Optimal feedback communication via posterior matching,” *IEEE Transactions on Information Theory*, vol. 57, no. 3, pp. 1186–1222, 2011.

[11] 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.

[12] 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.

[13] 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.

[14] M. Lian, C. Häger, and H. D. Pfister, “What can machine learning teach us about communications?” in *2018 IEEE Information Theory Workshop (ITW)*. IEEE, 2018, pp. 1–5.

[15] D. Sussillo and O. Barak, “Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks,” *Neural computation*, vol. 25, no. 3, pp. 626–649, 2013.

[16] A. Karpathy, J. Johnson, and L. Fei-Fei, “Visualizing and understanding recurrent networks,” *arXiv preprint arXiv:1506.02078*, 2015.

[17] Y. J. Choe, J. Shin, and N. Spencer, “Probabilistic interpretations of recurrent neural networks,” *Probabilistic Graphical Models*, 2017.

[18] S. Jain and B. C. Wallace, “Attention is not explanation,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Volume 1 (Long and Short Papers)*, 2019, pp. 3543–3556.

[19] S. Wiegreffe and Y. Pinter, “Attention is not not explanation,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019*, 2019, pp. 11–20.

[20] 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.

[21] A. Mulgund, R. Shekhar, N. Devroye, G. Turán, and M. Žefran, “Evaluating interpretations of deep-learned error-correcting codes,” in *2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2022, pp. 1–8.

[22] N. Devroye, A. Mulgund, R. Shekhar, G. Turán, M. Žefran, and Y. Zhou, “Interpreting training aspects of deep-learned error-correcting codes,” in *2023 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2023, pp. 2374–2379.

[23] A. Mulgund, N. Devroye, G. Turán, and M. Žefran, “Decomposing the training of deep learned turbo codes via a feasible map decoder,” in *2023 12th International Symposium on Topics in Coding (ISTC)*. IEEE, 2023, pp. 1–5.

[24] Y. Jiang, H. Kim, H. Asnani, S. Kannan, S. Oh, and P. Viswanath, “Turbo autoencoder: Deep learning based channel codes for point-to-point communication channels,” *Advances in neural information processing systems*, vol. 32, 2019.

[25] Y. Zhou, N. Devroye, G. Turán, and M. Žefran, “Interpreting Deepcode, a learned feedback code – extended ArXiv version,” Apr. 2024. [Online]. Available: https://arxiv.org/pdf/2404.17519