Abstract—We present a new deep-neural-network (DNN) based error correction code for fading channels with noisy channel output feedback, called deep SNR-robust feedback (DRF) code. At the encoder, parity symbols are generated by a long short term memory (LSTM) network based on the message as well as past forward channel outputs observed by the transmitter in a noisy fashion. The decoder uses a bi-directional LSTM architecture along with a signal to noise ratio (SNR)-aware attention module. The proposed code overcomes two major shortcomings of the previously proposed DNN-based codes over channels with noisy channel output feedback: (i) SNR-aware attention mechanism at the decoder enables reliable application of the same trained DNN over a wide range of SNR values; and (ii) curriculum training with batch-size scheduling is used to speed up and stabilize training while improving the SNR-robustness of the resulting code. We show that the DRF codes significantly outperform state-of-the-art in terms of both the SNR-robustness and the error rate in additive white Gaussian noise (AWGN) channels with feedback. In fading channels with perfect phase compensation at the receiver, DRF codes learn to efficiently exploit the knowledge of the instantaneous fading amplitude (which is available to the encoder through feedback) to reduce the overhead and complexity associated with channel estimation at the decoder.

Index terms—Communication with feedback, channel coding, LSTM, attention neural networks, curriculum training.

I. INTRODUCTION

Most wireless communication systems incorporate some form of feedback from the receiver to the transmitter. Although feedback does not improve the Shannon capacity of a channel [1], it can significantly boost the reliability at finite block-lengths [2]–[4]. Codes that make full use of feedback can potentially achieve improved performance as predicted in [4]. However, the design of reliable codes for channels with feedback has been a long-standing and notoriously difficult problem. Several coding schemes for channels with feedback have been proposed in the past [2], [5]–[8]; however, known solutions either do not approach the performance predicted in [4], or introduce unaffordable complexity. These schemes are also extremely sensitive to both the precision of the numerical computations and the noise in the feedback channel [3]. It has been proven that with noisy output feedback, linear coding schemes fail to achieve any positive rate [9]. This is especially troubling since all practical codes are linear and linear codes are known to achieve capacity without feedback [10], and boost the error performance significantly in the case of noiseless feedback [2]. For the noisy feedback case, considerable improvements have been achieved using non-linear modulo operations [11].

More recently, some progress has been made by applying machine learning (ML) techniques, where channel decoding is regarded as a classification task, and the encoder and decoder, implemented as deep neural network (DNN) architectures, are jointly trained in a data-driven fashion [12]–[14]. In [12], the authors propose Deepcode for communication with feedback, consisting of a recurrent neural network (RNN) encoder architecture along with a two-layer bi-directional gated recurrent unit (GRU) decoder architecture, which are trained jointly on a dataset of random input/output realizations of the channel. In [13], a convolutional neural network (CNN) encoder/decoder architecture with interleaving is used. In [14], deep extended feedback (DEF) codes are introduced, which improve the error correction capability in [12] by an extended feedback mechanism that introduces longer range dependencies within the code blocks. These DNN-based codes achieve lower error rates in comparison with traditional codes (e.g. Turbo, LDPC, and Polar codes that do not exploit the feedback, as well as the Schalkwijk–Kailath scheme [2] with a low resolution feedback), over an additive white Gaussian noise (AWGN) channel with output feedback at the typical code rate of $r = 1/3$ and relatively short block length of $L = 50$ [12]–[14].

Despite their significant performance, DNN-based codes are very sensitive to the mismatch between the actual channel signal to noise ratio (SNR) and the SNR value that the code is trained for, which limits their application in practical communication systems with time-varying SNR values. In this paper, we propose Deep SNR-Robust Feedback (DRF) codes for fading channels with noisy output feedback, which overcome the above-mentioned limitation of DNN-based channel codes. The DRF encoder transmits a message followed by a sequence of parity symbols, which are generated by a long short term memory (LSTM) architecture based on the message as well as the delayed past forward channel outputs observed by the encoder through a noisy feedback channel. The decoder uses a bi-directional LSTM architecture along with a SNR-aware attention [15], [16] network to decode the message. The major contributions of this paper can be summarized as follows:

- We propose an attention mechanism that enables SNR-aware decoding of the DRF code, thereby considerably improving its robustness in realistic time-varying channels, where there may be a considerable mismatch between the training SNR and the instantaneous channel SNR. For fading channels, in which the instantaneous SNR may be varying on each transmitted codeword (slow fading) or symbol (fast fading), we show that the proposed DRF codes learn to efficiently exploit the Channel State Information (CSI), which is available to the encoder through feedback, and no further improvement is possible by providing the CSI to the decoder.

- We propose a training approach with SNR scheduling and batch-size adaptation. We start the training at low SNR values with a small batch-size, and gradually increase the SNR and the batch-size along the training epochs according to a schedule. The proposed training approach improves the SNR-robustness of the resulting code and speeds up the training. The DRF codes with the proposed training approach not only achieve considerable SNR-robustness, but also improve the error rate over Deepcode [12] roughly by an order of magnitude.

The rest of this paper is organized as follows. In Section II, we present the feedback channel model considered in this paper. In Section III, we provide the DNN architectures for the DRF encoder and decoder. In Section IV, we present our proposed training technique. Section V presents the simulation results, and Section VI concludes the paper.

II. SYSTEM MODEL

Fig. 1 illustrates the canonical fading channel with passive noisy output feedback considered in this paper. Perfect phase compensation...
at the receiver is assumed resulting in a real-valued magnitude fading channel. We have \( y_i = \alpha_i x_i + n_i \), where \( x_i \) and \( y_i \) denote the channel input and output symbols, respectively, \( \alpha_i \) is the channel fading coefficient, \( n_i \) is the independent and identically distributed (i.i.d.) Gaussian noise term, i.e., \( n_i \sim \mathcal{N}(0, \sigma_n^2) \). We will assume that the channel fading coefficient comes from a prescribed distribution. We consider both slow and fast fading scenarios, where the fading coefficient takes random i.i.d. realizations on each symbol, the linear minimum mean square error (LMMSE) estimate of the channel gain is calculated by

\[
\hat{\alpha}_i = \frac{E[|x_i|^2] - \sigma_n^2}{\sigma_n^2} x_i, \quad \text{where the expectation is over the randomness in the noisy feedback symbols.}
\]

We denote the forward and feedback channel SNR values by \( \rho \) and \( \eta \), respectively.

III. ENCODER/DECODER ARCHITECTURES

A major limitation of the existing DNN-based code designs in [12]-[14] is their dependence on the channel SNR. That is, the encoder-decoder pairs are trained jointly for a specific SNR value. This means that, to be able to use these codes in practice, we will have to train and store a different DNN pair for different ranges of SNR values. This significantly limits their practical use in realistic channels with varying SNR. On the other hand, in conventional channel codes, the encoder depends only on the transmit power constraint, and the decoder uses the same decoding algorithm for all SNR values after converting the channel outputs into likelihood values depending on the channel SNR. Accordingly, a major goal of our paper is to implement a similar approach for DNN-based code design. This is achieved in this paper by incorporating an attention mechanism into the decoder of our proposed DRF code. This will allow us to train and store a single DNN, which can be used for all SNR values. Apart from this, we design the DRF code for fading channels with feedback, when the instantaneous channel SNR may change over time. This is different from the previous works that consider the simple AWGN channel with feedback [12]-[14]. Fig. 2 depicts our proposed DRF encoder and decoder architectures for a rate \( r = 1/3 \) code (the architecture could be easily generalized to all rates \( r = 1/q \) with \( q \) being any positive integer greater than 1).

A. Encoder

Fig. 2a illustrates the encoder architecture. Encoding is a two-phase process: in phase I, vector \( b = [b_1, \ldots, b_K, 0]^T \) consisting of the message bits padded by a zero is transmitted over the channel by an antipodal mapping, i.e., \( e_1 = 2b - 1 \). Zero padding is applied to mitigate the increasing error rate effects on the last few bits of the block as suggested in [12]. During phase II, the encoder uses a 1-layer LSTM network, including 1 LSTM units to generate two sets of parity bits, i.e., \( e_{(1)}^I \) and \( e_{(2)}^I \), based on the observations of channel noise and fading in phase I and the delayed noise and fading in phase II on each of the two sets of parity symbols. We use single directional LSTM units due to the causality constraint enforced by the channel model. The LSTM activation is hyperbolic tangent, i.e., \( \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \). The resulting code block transmitted over the channel is

\[
\begin{align*}
\mathbf{x} &= [\mathbf{x}_I^{(1)}, \mathbf{x}_II^{(1)}, \mathbf{x}_II^{(2)}]^T, \\
\mathbf{x}_I^{(1)} &= \mathcal{P}\{e_{(1)}^I\} = [x_1, x_2, \ldots, x_{K-1}], \\
\mathbf{x}_II^{(1)} &= \mathcal{P}\{e_{(1)}^I\} = [x_{K+2}, x_{K+3}, \ldots, x_{2K+2}]^T, \\
\mathbf{x}_II^{(2)} &= \mathcal{P}\{e_{(2)}^I\} = [x_{2K+3}, x_{2K+4}, \ldots, x_{3K+3}]^T.
\end{align*}
\]

Here, \( \mathcal{P}\{\cdot\} \) denotes a learned power re-allocation layer to balance the error over the whole block as suggested in [12].

The encoder estimates the forward channel from observations of the feedback. It knows the transmitted symbol \( x_i \) and observes the corresponding feedback symbol \( z_i = \alpha_i x_{i-1} + n_{i-1} + m_i \) with a single delay, from which it can estimate the CSI \( \alpha_i \). The estimate of the CSI, denoted by \( \hat{\alpha}_i \), is then input to the encoder. In the fast fading scenario, where the fading coefficient takes random i.i.d. realizations on each symbol, this estimate is calculated by
\[ \hat{\alpha}_i = \frac{x_{i-1} \text{var}(\alpha)}{|x_{i-1}|^2 \text{var}(\alpha) + \sigma_a^2 + \sigma_m^2} z_i + \frac{\sigma_a^2 + \sigma_m^2}{|x_{i-1}|^2 \text{var}(\alpha) + \sigma_a^2 + \sigma_m^2} E[\alpha], \]

where \( E[\alpha] \) and \( \text{var}(\alpha) \) denote the expected value and variance of the fading coefficient, respectively. In a slow fading scenario, the fading coefficient is fixed over the whole codeword, i.e., for the considered rate \( r = 1/3 \) code with a single bit zero padding we have \( \alpha_1 = \cdots = \alpha_{3K+3} = \alpha \). The fading coefficient \( \alpha \) takes random i.i.d. realizations over different codewords, and the transmitter uses the causal vectors \( \hat{z}_i = [z_1, \ldots, z_i]^T \) and \( x_i = [x_1, \ldots, x_i]^T \) to calculate the LMMSE channel estimate as

\[ \hat{\alpha} = \text{var}(\alpha) x_i^T (\text{var}(\alpha) x_i x_i^T + \sigma_a^2 I + \sigma_m^2 I)^{-1} z_i, \]

\[ + E[\alpha](1 - \text{var}(\alpha) x_i x_i^T (\text{var}(\alpha) x_i x_i^T + \sigma_a^2 I + \sigma_m^2 I)^{-1})^{-1} x_i, \]

in which \( I \) is the identity matrix. In (1), (2), knowledge of \( E[\alpha] \) and \( \text{var}(\alpha) \) at the transmitter is assumed.

The causal CSI available at the encoder is fed into the LSTM units to cope with channel uncertainty due to fading. To this end, we concatenate the vector of instantaneous channel fading coefficients in phase I, i.e. \( \alpha_I = [\alpha_1, \ldots, \alpha_K+1]^T \), and the causal fading coefficient in phase II i.e. \( D'_{\alpha_{i+1}} = [0, \alpha_{K+2}, \ldots, \alpha_{2K+1}]^T \), and \( D^{(2)}_{\alpha_{i+1}} = [0, \alpha_{K+3}, \ldots, \alpha_{3K+2}]^T \), and feed into the LSTM units at the encoder (D denotes a single delay, see Fig. 2a). We also provide estimates of the noise in the forward and feedback channels to the encoder, i.e. \( z_i = \alpha_i \odot x_i, \) \( D(z_{\alpha i} = \alpha_i \odot x_i^\alpha) \) and \( D(x_{\alpha i} - \alpha_i \odot x_i^\alpha) \), where \( \odot \) denotes element-wise multiplication.

For the AWGN case where \( \alpha_1 = \alpha_2 = \cdots = \alpha_K = \alpha \), the corresponding inputs are omitted to avoid unnecessary complexity.

### B. Decoder

Fig. 2b illustrates the DRF decoder consisting of a two-layer LSTM architecture (each including \( K + 1 \) LSTM units) and a SNR-aware fully connected attention network. At the decoder, we use bi-directional LSTM layers to exploit long range forward and backward dependencies in the received code block. The phase I and II received signals are concatenated at the decoder and fed to the bi-directional LSTM layers. Each LSTM layer is followed by batch normalization. Similarly to the encoder, the LSTM activation is hyperbolic tangent while the output activation is sigmoid. The bi-directional LSTM layers extract features from the noisy received signals, which are then used for efficient decoding. Note that we use LSTM layers at both the encoder and the decoder, which, according to our observations, considerably reduce the error rate in comparison with simple RNN and GRU layers used in [12]. This is because LSTM layers can better learn long-range dependencies by avoiding the gradient vanishing problem in training long RNN layers [17].

### C. SNR-Aware Attention

Another novelty in our decoder architecture is the SNR-aware attention module. An attention mechanism is a vector of importance weights to measure the correlations between a vector of inputs and the target to be predicted. Attention weights are calculated as a parameterized attention function with learnable parameters [15], [16]. We use a two-layer fully connected (FC) attention at the DRF decoder. The idea is to let the attention layers learn how much each bi-LSTM output features should be weighted according to the SNR. By means of the attention module, we explicitly provide the noise standard deviation to the decoder, which enables learning codes that are capable of adaptation to the channel SNR, which in turn allows to use the same trained encoder/decoder weights over a wide range of channel SNR values. Here, the standard deviations of the forward and feedback channel noise are obtained through link-level estimation. The number of attention weights determines the number of neurons at the last FC layer, i.e., \( 2HK \), where \( H \) is the length of the LSTM hidden state (i.e., \( H = K \) here) and is multiplied by 2 because the LSTM layer is bi-directional. The total number of FC attention layers and the number of neurons in each intermediate layer are hyperparameters optimized numerically for the best performance.

### IV. TRAINING DRF CODES

We denote the \( i \)’th training sample by \( S_i = \{b_i, \alpha_i, n_i, m_i\} \), which consists of a random realization of the message \( b_i \), the corresponding realization of the channel fading coefficient \( \alpha_i \), and the forward and feedback noise realizations, \( n_i \) and \( m_i \), respectively. We denote the encoder and decoder functions by \( f(\cdot; \theta) \) and \( g(\cdot; \psi) \), where \( \theta \) and \( \psi \) are the trainable encoder and decoder parameters. We have, \( \hat{b}_i = g(\alpha_i f(S_i; \theta) + n_i; \psi) \). To train the model, we minimize

\[ \mathcal{L}(\theta, \psi, \mathcal{B}) = -\frac{1}{|\mathcal{B}|} \sum_{S_i \in \mathcal{B}} l(\hat{b}_i, b_i; \theta, \psi), \]

where \( \mathcal{B} \) is a batch of samples, \( l(\hat{b}_i, b_i; \theta, \psi) \) is the binary cross entropy loss given by

\[ l(\hat{b}_i, b_i; \theta, \psi) = \sum_{k=1}^{K} [b_i]_k \log_2(1 - [\hat{b}_i]_k) + (1 - [b_i]_k) \log_2([\hat{b}_i]_k), \]

and \([b_i]_k\) and \([\hat{b}_i]_k\) denote the kth bit of the message and its estimate.

We use stochastic gradient descent (SGD) for training, where the vector of all trainable parameters \( \phi^T = [\theta^T, \psi^T] \) is optimized in an iterative manner

\[ \phi^{(t)} = \phi^{(t-1)} - \mu_t \nabla_{\phi} \mathcal{L}(\phi^{(t-1)}, \mathcal{B}^{(t)}), \]

where \( t \) is the iteration index, \( \mu_t > 0 \) is the learning rate, and \( \mathcal{B}^{(t)} \) is a random batch from the dataset.

To ensure that the model is trained with many random realizations of the data and noise, we generate and use a new random set of samples in each epoch. We denote the dataset used in the u’th training epoch by \( D^u = \{S_i\}_{i=1}^{D^{(u)}} \), where \( |D^u| = \zeta |B^u| \), \( \zeta \) is a constant and \( |B^u| \) is the batch-size for the u’th epoch. Training DNNs with SGD, or its variants, requires careful choice of the training parameters (e.g., learning rate, batch-size, etc.).

#### A. Batch-size Adaptation

In training machine learning models, a static batch-size held constant throughout the training process forces the user to resolve a tradeoff. Small batch sizes are desirable since they tend to achieve faster convergence. On the other hand, large batch sizes offer more data-parallelism, which in turn improves computational efficiency and scalability. However, for the specific channel encoder/decoder training task a significantly larger batch size is necessary not only due to the data-parallelism benefits, but also because after a few training steps, the error rate and consequently the binary cross entropy loss (3) becomes very small, typically \( 10^{-3} \sim 10^{-7} \) for the range of SNR values considered here. Hence, to get a statistically accurate estimate of such a small loss value, and consequently, an accurate estimate of the gradient update in (5), the batch-size must be very large (typically \( \sim 10000 \) samples here).

We here propose an adaptive batch size scheme tailored for training a DNN-based channel encoder and decoder pair. In this scheme, we train the model starting from a small batch-size \( |B^1| \), and multiply the batch size by a factor of \( \kappa > 1 \) whenever the cross entropy loss does not decrease by a factor of \( \lambda \) in two consecutive epochs, until we reach a maximum batch-size of \( B_{\text{max}} \). The maximum batch-size is constrained by the memory resources available to our training
platform. We hence train with a sequence of batch-sizes, \(|B_1| \leq |B_2| \leq \cdots \leq |B_U| \leq B_{\text{max}}\), where \(U\) is the total number of epochs. Starting from a smaller batch size enables a faster convergence during initial epochs. We increase the batch size whenever trapped around a minimum due to insufficiency of the batch size to achieve an accurate estimate of the gradient. The proposed batch-size adaptation stabilizes and speeds up the training process.

\section*{V. NUMERICAL EVALUATIONS}

In this section, we evaluate the performance of the proposed DRF codes and provide comparisons with previous works. In all the simulations, we use 10^8 random samples to achieve a reliable estimate of the error rate. Each sample includes a random realization of the message \(b\), and the corresponding random realizations of forward and feedback channels. We set \(K = 50, L = 153\), and use the Adam optimizer. The values of the hyperparameters are: \(U = 15, |B_1| = 1000, B_{\text{max}} = 16000, \zeta = 100, \lambda = 2, \kappa = 2\).

\subsection*{A. AWGN Channel}

We first consider a static AWGN channel, i.e. \(\alpha_i = 1, \forall i\). We show the robustness of the proposed DRF codes to a mismatch between the training and the actual channel SNR values, and provide BLER comparisons with existing conventional and feedback channel codes. We show that the DRF codes outperform the benchmark low density parity check (LDPC) codes adopted for the fifth generation new radio (5G NR) [20], by three orders of magnitude and the previously trained Deepcode [12] by an order of magnitude. In Fig. 2a, we compare the BER curves for the two cases with and without CSIR for both fast and slow Rayleigh fading channels depending on the availability coefficient \(\alpha_i\). Depending on the wireless environment, the CSI coefficient \(\alpha_i\) may follow various statistics. We adopt the Rayleigh fading channel model with an average power of \(\Omega = 2\sigma^2\).

\subsection*{1) SNR-Robustness:}

We first compare the BLER of the proposed DRF codes with and without the attention module, when there is a mismatch between the actual channel SNR, \(\rho\), and the SNR used for training, \(\hat{\rho}\). Here, we train the codes with batch-size adaption but for a specific SNR value (i.e., without SNR scheduling). The SNR mismatch is defined as \(\Delta\rho = \rho - \hat{\rho}\). The results are depicted in Fig. 3, where we plot BLER versus \(\Delta\rho\) for \(\rho = -1, 0, 1\) dB. This figure shows that without the SNR-aware attention module at the decoder, the BLER is very sensitive to the SNR mismatch. In this case, a negative SNR mismatch (i.e., training SNR is higher than the actual channel SNR), can significantly degrade the BLER by orders of magnitude. The BLER is less sensitive to a positive mismatch but still roughly an order of magnitude BLER degradation is observed if there is \(\Delta\rho = +3\) dB mismatch between the training and test SNR values. This figure shows that DRF codes are significantly more robust to both positive and negative SNR mismatch due to the SNR-aware attention layers added to the decoder.

\subsection*{2) Comparison with Previous Works:}

In Fig. 4, we compare the performance of DRF codes with NR LDPC [20], Deepcode [12], and the DEF code [14]. We plot the BLER values achieved for each code for the forward channel SNR values in the range \([-1, 2]\) dB when (a) the feedback is noiseless (\(\eta = \infty\)), and (b) the feedback SNR is \(\eta = 20\) dB. The blue curve reports the BLER for the RNN-based Deepcode architecture proposed in [12]. According to this figure, the proposed DRF codes reduce the BLER by almost three orders of magnitude in comparison with NR LDPC and an order of magnitude in comparison with Deepcode [12]. Note that for the Deepcode and DEF code, we have trained and used a different DNN for each of the four SNR points. However, for the DRF code, we have used a single DNN for all the SNR points, which is trained using our proposed SNR scheduling approach. Hence, in comparison with the state-of-the-art DEF code, DRF code achieves SNR-robustness with no significant performance degradation in the noiseless feedback case. When the feedback is noisy, DRF code also outperforms DEF code.

\section*{B. Fading Channel}

In this subsection, we consider fading channels with feedback as depicted in Fig. 1. Depending on the wireless environment, the CSI coefficient \(\alpha_i\) may follow various statistics. We adopt the Rayleigh fading channel model with an average power of \(\Omega = 2\sigma^2\). In Fig. 5, we compare the resulting BER curves for DRF codes over both fast and slow Rayleigh fading channels depending on the availability of CSI at the receiver (CSI). With CSI, the decoder first performs LMMSE channel compensation on the received symbols, i.e., \(\hat{y}_i = \frac{-\bar{y}_i}{|\alpha_i|^2 + \sigma^2}\), and then uses \(\hat{y}_i\) as input to the bi-directional LSTM units for decoding. Note that the encoder is the same as depicted in Fig. 2a for both cases. Fig. 5a exhibits the BER curves for the noiseless feedback case (\(\eta = \infty\)), and Fig. 5b for the noisy feedback case at \(\eta = 20\) dB. For a fair comparison, we use the exact value of \(\alpha_i\) (not its estimate) both at the encoder and decoder. The curves show similar performance for the two cases with and without CSI for both
channel estimation at the receiver can be reduced. A desirable property, which means that with the proposed DRF codes, we showed that DRF codes can learn to efficiently use the knowledge of the instantaneous channel fading (available to the encoder through feedback) to reduce the overhead and complexity associated with channel estimation at the receiver.

VI. CONCLUSIONS

We proposed a DNN-based error correction code for fading channels with output feedback, called the DRF code. It is shown that the DRF code significantly improves over the previously proposed DNN-based codes in terms of the error rate as well as robustness to varying SNR values for AWGN channels with feedback. Over fading channels, we showed that DRF codes can learn to efficiently use the knowledge of the instantaneous channel fading (available to the encoder through feedback) to reduce the overhead and complexity associated with channel estimation at the receiver.

REFERENCES


