Deep Learning-Aided Coding for the Fading Broadcast Channel with Feedback

Siyao Li, Daniela Tuninetti, and Natasha Devroye
University of Illinois at Chicago, Chicago, IL 60607, USA
Email: {sli210, danielat, devroye}@uic.edu

***Abstract*—We consider the design of practical codes for a symmetric two-user fading Gaussian Broadcast Channel (BC) with feedback. We construct a two-phase coding scheme with the help of deep Neural Networks (NNs) that seeks to optimize the encoder and decoders jointly. Interpreting a communication system as an autoencoder (denoted by AE), we train the AE under various scenarios of noiseless feedback signals. Performance evaluation is presented for Rayleigh distributed channel state, which reveals the existence of a trained NN-based two-phase model that outperforms state-of-the-art codes in the low SNR regime. Considering the availability of feedback signals, we train the AE with different inputs, and observe that feedback consisting of received signals appears to be more beneficial than channel states to boost reliability under the proposed scheme. We provide initial interpretations of the encoding scheme which uses channel state feedback.**

I. Introduction

In the fifth generation (5G) communication systems, the demand for codes with short message lengths and low error rates has significantly increased. Careful design of the physical layer helps in satisfying these requirements. A channel model particularly important in downlink wireless communications, is the Fading Additive White Gaussian Noise Broadcast Channel (F-AWGN-BC), where the channel between the single transmitter or base-station sending signal X, and multiple users is modeled as $Y_u = \sqrt{S_u}X + N_u$ for user u, where N_u is the Additive White Gaussian Noise (AWGN), and S_u is the fading parameter, or Channel State Information (CSI). When the transmitter has independent messages to send to different subsets of users, the capacity region, the largest set of rates for which the probability of error vanishes as the block length increases, captures some of the tension seen in BCs: a single signal must be encoded such that when different quality versions of this signal are received at the users, each can extract their own intended message(s).

Channel Output Feedback (COF) is commonly used in communication systems to send back information about the decoding process as well as the channel dynamics. Although COF does not increase the capacity region of the Point-to-Point (P2P) memoryless channel, it can simplify the transmission schemes and improve the reliability dramatically [1], [2]. COF can strictly enlarge the capacity region of multiuser channels [3]. Several linear coding schemes are proposed for AWGN-BC with COF without fading [3]–[7] and with fading [8], [9].

Recently, Neural Networks (NNs) have started to attract significant attention in the context of wireless communications and networking [10]–[14], since the development of smart devices and mobile applications has significantly increased the autonomy of a wireless network. In particular, Recurrent Neural Network (RNN) based architectures, have been used for learning feedback codes in communication systems leveraging feedback from the system in [15] as well as predicting the channel fading in [16], [17]. [18] demonstrates a Deep Learning approach for training a static AWGN P2P channel with noisy feedback. [19] presents a two-phase coding scheme for the multiuser degraded broadcast channel without feedback, which is able to learn to communicate on this channel using superposition coding. [20] evaluates the performance of an AE for interference channels without feedback.

Contribution: We build on the above mentioned promising results, and focus on using RNN-based strategies to build coding schemes for F-AWGN-BC with COF. We propose a deep learning-aided two-phase scheme and compare it with the state-of-the art coding schemes for F-AWGN-BC without feedback in [21] and with COF in [8], [9]. We train two models: one with feedback of received signal Y_u, the other with both Y_u and S_u. Our trained models suggest that the received signal Y_u plays a more important role on improving the Bit Error Rate (BER) than the channel state S_u at low SNR. We also provide initial interpretations of the codewords constructed by the RNN which uses channel state feedback.

Paper Organization: Section II introduces the F-AWGN-BC system model and the NN used in this work; Section III presents the deep learning-aided AE; Section IV includes the simulation results; Section V concludes the paper.

Notation: In this paper, we represent user indices u and time indices t by subscripts, such as $Y_{u,t}$. Sequences of Random Variables (RVs) are denoted by boldface, such as $\mathbf{Y_u} := [Y_{u,1}, \cdots, Y_{u,N}]^T$. To denote explicitly the dimensionality of a vector, we use superscripts and subscripts in the following manner: $\mathbf{Y}_{u,i}^{j} := [Y_{u,i}, \cdots, Y_{u,j}]$ for $i \leq j$ and $\mathbf{Y}_{u}^{i-1} := [\mathbf{Y}_{u,1}, \cdots, \mathbf{Y}_{u,i-1}]^T$. $\mathbb{R}$ and $\mathbb{R}^n$ represent sets of real scalars and n-dimensional real column vectors, respectively. The notation $[n]$ for $n \in \mathbb{N}$ denotes the set $\{1, 2, \cdots n\}$. We use $\mathbb{R}_{\geq 0}$ and $\mathbb{N}$ to denote the set of nonnegative real numbers and natural numbers. We write $\mathbb{F}(\cdot)$ for cumulative distribution and $\mathbb{P}(\cdot)$ for the probability of a measurable event within the parentheses. $\mathcal{N}(\mu, \sigma^2)$ represents a real Gaussian distribution with mean μ and variance σ^2. We use $\mathrm{sgn}(x)$ to denote the sign function, where $\mathrm{sgn}(x) := 1$ if $x \geq 0$ and $\mathrm{sgn}(x) := -1$ if $x < 0$.

II. System Model and Deep Learning Basics

A. System model

In this paper, we consider a two-user real-valued F-AWGN channel with BPSK modulation. The received signal $Y_{u,t}$ for user $u \in [2]$ at time $t \in [N]$ is

$$Y_{u,t} = \sqrt{S_{u,t}} X_t + Z_{u,t} \in \mathbb{R}, \tag{1}$$

where $X_t \in \mathbb{R}$ denotes the transmitted signal, $Z_{u,t} \sim \mathcal{N}(0, \sigma_n^2)$ is the real-valued AWGN with zero mean, power σ_n^2, and $S_{u,t} \in \mathbb{R}_{\geq 0}$ is the channel state of user u with alphabet $\mathcal{S}$. We assume that the RVs (S_1, S_2) are independent and form a memoryless process over time, that the noise variables (Z_1, Z_2) are independent across users and time, and that the input is subject to the average unit power constraint $\mathbb{E}[X^2] \leq 1$. We also assume that $S_{u,t}$, and $Z_{u,t}$ are independent; after each transmission, the instantaneous fading $S_{u,t}$ is known at the receiver side and there exists a noiseless feedback link that transmits one-step delayed information of feedback from the receivers to the transmitter.

A code for the F-AWGN-BC with two receivers is defined as follows. The transmitter must convey K_u information bit $\mathbf{b}_u = (b_{u,1}, \cdots, b_{u,K_u}) \in \{0,1\}^{K_u}$ reliably to user $u \in [2]$. The rate for user $u = 1, 2$ is $R_u = K_u/N$, where N is the number of time slots used to transmit all $K_1 + K_2$ information bits. We distinguish different cases based on the amount of CSI at the transmitter (CSIT):

1) no CSIT: $X_t(\mathbf{b}_1, \mathbf{b}_2, \mathbf{Y}_1^{t-1}, \mathbf{Y}_2^{t-1}), t \in [N]$;
2) only CSIT: $X_t(\mathbf{b}_1, \mathbf{b}_2, \mathbf{S}_1^{t-1}, \mathbf{S}_2^{t-1}), t \in [N]$;
3) COF: $X_t(\mathbf{b}_1, \mathbf{b}_2, \mathbf{S}_1^{t-1}, \mathbf{S}_2^{t-1}, \mathbf{Y}_1^{t-1}, \mathbf{Y}_2^{t-1}),\ t \in [N]$,

where $X_t(\cdot)$ is the encoding function at time t. User $u \in [2]$ estimates $\widehat{\mathbf{b}}_u = \mathrm{dec}_u(\mathbf{Y}_u^N, \mathbf{S}_1^N, \mathbf{S}_2^N)$ for some decoding function dec_u. Accordingly, the coding scheme for F-AWGN-BC without COF discussed in Section IV means $X_t(\mathbf{b}_1, \mathbf{b}_2),\ t \in [N]$. That is, the encoding function only depends on the information bits for two users. The average bit error rate for user u is defined as $\mathrm{BER}_u := \frac{1}{K_u} \sum_{i=1}^{K_u} \mathbb{P}(\widehat{b}_{u,i} \neq b_{u,i})$. For the BC, the probability of error is defined to be the probability the decoded message is not equal to the transmitted message for any user. Similarly, we define the joint average bit error rate as

$$\mathrm{BER} := \frac{1}{\max(K_1, K_2)} \sum_{i=1}^{\max(K_1,K_2)} \mathbb{P}\left(\widehat{b}_{u,i} \neq b_{u,i}, \exists u \in [2]\right). \tag{2}$$

To simplify the analysis, we consider the case where $K_1 = K_2$, i.e., the information bits for the two users are the same.

B. Neural networks

Comprehensive theory of deep learning is presented in [22]. We now will briefly introduce some main ideas and concepts related to this work based on [23].

A NN consists of many connected neurons. For a single neuron, the weighted inputs are summed together, with a bias optionally added, and the result is forward to a nonlinear activation function. Commonly used activation functions are the sigmoid, tanh and rectified linear unit, which are respectively defined as

$$g_{\text{sigmoid}} = \frac{1}{1+e^{-x}},\ g_{\tanh} = \frac{e^x - e^{-x}}{e^x + e^{-x}},\ g_{\text{relu}} = \max(0, x).$$

In a feedforward NN, the neurons are arranged in layers without feedback connections. Each layer i with n_i inputs and m_i outputs performs the mapping $\mathbf{f}^{(i)} : \mathbb{R}^{n_i} \to \mathbb{R}^{m_i}$ with the weights and biases of the neurons as parameters. Denote $\mathbf{x}$ the input and $\mathbf{y}$ the output of the NN, the mapping between input and output is defined by a chain of functions depending on the set of parameters $\mathbf{w}, \mathbf{b}$ by $\mathbf{y} = \mathbf{f}(\mathbf{x}; \mathbf{w}, \mathbf{b}) = f^{(L-1)}\left(f^{(L-2)}\left(\cdots f^{(0)}(\mathbf{x})\right)\right)$, where L is the number of layers and is also called depth. The l-th layer is called *dense* or *fully-connected* if $f^{(l)}(x^{(l-1)}; W^{(l)}, b^{(l)}) = g(W^{(l)}\mathbf{x}^{(l-1)} + b^{(l)})$, where $g(\cdot)$ is an activation function and $\mathbf{x}^{(l-1)}$ is the output vector of the previous layer. For feedforward NNs, the NN only handles individual time steps. In a Recurrent Neural Network (RNN), the connected neurons form a direct graph along a temporal sequence. RNNs allow previous outputs to be used as inputs while having hidden states. For a traditional RNN, at each time slot t, the hidden state h_t and the output y_t are expressed as $h_t = g_1(W_{hh}h_{t-1} + W_{hx}x_t + b_h)$ and $y_t = g_2(W_{yh}h_t + b_y)$, where $W_{hh}, W_{hx}, W_{yh}, b_h, b_y$ are coefficients that are shared temporally and g_1, g_2 are activation functions. The vanishing and exploding gradient phenomena are often encountered in the context of traditional RNNs.

An elegant RNN structure, Long Short-Term Memory (LSTM) was designed in 1997 [24] to overcome this issue. It deals with long-term dependency by introducing memory cells in the recurrent hidden layer and multiplicative gates that regulate the information flow. A simplified version of LSTM structure, Gated Recurrent Unit (GRU) was introduced in 2014 [25]. For the sake of space, the mathematical expressions of LSTM and GRU are omitted here, which can be found in [24] and [25] respectively.

To find the optimal parameters of a NN, a specific loss function and a training set of known input-output mappings are required. By gradient descent optimization methods and the backpropagation algorithm [26], parameters minimizing the loss function can be found over the training set. In this work, we use Binary Cross-Entropy (BCE) loss function of the distribution $\mathbb{Q}$ relative to a distribution $\mathbb{P}$ over a given set defined as

$$H_{\mathbb{P}}(\mathbb{Q}) = \tag{3}$$
$$-\frac{1}{N}\sum_{i=1}^{N} \mathbb{P}(b_i)\log(\mathbb{Q}(\hat{b}_i)) + (1 - \mathbb{P}(b_i))\log(1 - \mathbb{Q}(\hat{b}_i)).$$

Based on our setup, $\mathbb{P}(b_i) \in \{0, 1\}$ is the i-th target information bit and $\mathbb{Q}(\hat{b}_i) \in [0, 1]$ is the predicted probability of the target information bit.

III. Deep Learning Aided Two-phase Scheme

With COF, the transmitter encodes current transmitted symbol based on the information bits and previous received

signals. The receiver decodes the information bits after collecting all the symbols it received. According to this recurrent structure, we choose RNN structure as our AE.

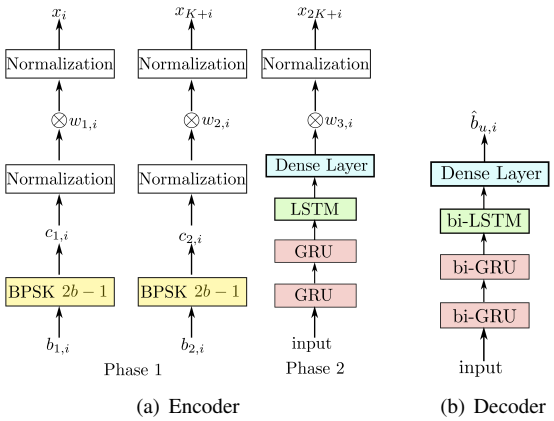
There are some hyperparameters to be selected before constructing a NN, such as the RNN structure, the number of hidden layers, and the number of units of each layer. We focus on differences between deep learning-aided AE and the mathematically derived linear scheme proposed in our previous work [8], [9]. We restrict attention to a fixed set of hyperparameters based on our empirical results. We refer readers to [27] for more research on hyperparameter optimization.

A. Autoencoder structure

The two-phase scheme is inspired by our previous work in [28] and the autoencoder structure is inspired by [15]. In this work, we stack LSTM and GRU together aiming to gain better training accuracy and convergence efficiency from both structures [29], [30]. We consider a symmetric F-AWGN-BC with COF, fix the rate $R = 1/3$ and number of information bits $K = 50$ for each user. The overall blocklength is thus $N = 150$.

Encoding: The structure is demonstrated in Fig. 1(a). The encoding process has two phases. Phase 1 consists of $2K = 100$ time slots, where we fix the first 50 time slots to process the information bits destined to user 1 and the second 50 time slots for user 2 (note that the order can be flipped); phase 2 consists of 50 time slots, where we transmit the refinements of the information bits transmitted in phase 1. Specifically, in phase 1, at each time slot, the transmitter first modulates the information bits $b_{u,t},\ u \in [2],\ t \in [K]$ intended for user u by BPSK. Then, the weighted parameters $w_{u,t},\ u \in [2],\ t \in [K]$ are applied to the normalized signals to assign power to each information bit adaptively. Lastly, the weighted signals are normalized to satisfy the power constraint before transmission. In phase 2, at each time slot, the deep learning-aided two-phase scheme takes the input of size 14, which are $b_{1,t}, b_{2,t}$, the information bits to each user, and $Y_{1,t}, Y_{2,t}, Y_{1,K+t}, Y_{2,K+t}, S_{1,t}, S_{2,t}, S_{1,K+t}, S_{2,K+t}$, the COF from both users in phase 1, as well as $Y_{1,2K+t-1}, Y_{2,2K+t-1}, S_{1,2K+t-1}, S_{2,2K+t-1}$, the one-time-slot-delayed COF from both users of phase 2. The input is passed through a deep RNN, consisting of two layers of GRU and one layer of LSTM stacking on top of each other, which means the output of the previous layer is the input of the current layer. This deep RNN is connected to a dense layer followed by a weighted parameter. Similar to phase 1, the generated signals are normalized to satisfy the power constraint before transmission.

Decoding: The receivers start decoding after all information bits are received. Each decoder has the same structure illustrated in Fig. 1(b). The input of the decoder is (Y_u^N, S_u^N), the received signals and channel fading in two phases. The decoder has similar structure to the encoder in phase 2, which consists of two layers of bi-directional GRU, one layer of bi-directional LSTM stacking on top of each other, and a dense layer outputting the probability of the estimated information bit being one.



(a) Encoder (b) Decoder

Fig. 1: Deep learning-aided AE structure

Training: We apply the zero padding technique presented in [15] and train the RNN at a fixed value of SNR= 2 dB. The encoder and decoders are trained jointly in epochs. At the end of each epoch, the gradient of the loss function in (3) is calculated and propagated back over the entire training set and the parameters of the entire NN are updated accordingly using the Adam optimization method [31]. The parameters are updated according to the sum of the BCE loss function of each user in this symmetric setting. Note that the goal of the normalization step in phase 1 before multiplying the weighted parameter is to ease the training. As the channel fading and noise is generated randomly for each time slot, the receivers rarely receive the same signal twice. While we fix the message bits for each user to $K = 50$, we can train on a large set by increasing the number of epochs. The training process is iteratively carried out until the network reaches a certain convergence condition. We use a cyclical scheduler [32] to adjust the learning rate based on the number of epochs. Once the training process completes, the trained encoder and decoders are evaluated using test data based on a range of SNRs.

Note that our proposed scheme can be easily extended to more than two users. Specifically, the COF of all the users is feed back as the input of the encoder, the each user performs the same decoding steps as demonstrated in Fig. 1(b), and the training is evaluated based on the sum of the BCE loss functions of all users. The number of information bits for each user and block length will also be changed accordingly.

B. Computational complexity and discussions

We specify the parameters in Table I. As illustrated in Fig. 1, the encoder in phase 2 has similar structure to the decoder. The GRU (LSTM) and bi-GRU (bi-LSTM) employed in encoder and decoder both have 50 (25) hidden neurons. There are $N_{\text{GRU}} = 2.52 \times 10^4$ and $N_{\text{LSTM}} = 7.7 \times 10^3$ trainable parameters in the RNN of the encoder. Considering the power allocation, dense layer, and decoder parameters, there are 2.09983×10^5 trainable parameters in total. Based on the NN configurations in Table I, the BCE loss function converges after around 2000 epochs.

TABLE I: NN configuration.

Parameters	Values
Rate	$\frac{1}{3}$
Batch size	200
Information bit size	50
Loss function	BCE
GRU input size (encoder)	14
GRU hidden neurons	50
GRU layer number	2
LSTM input size (encoder)	50
LSTM hidden neurons	25
LSTM layer number	1
Dense + sigmoid output size	1
bi-GRU input size (decoder)	6
bi-LSTM input size (decoder)	100

Since deep learning for communications is still a relatively new field, little is known about optimal data representations, loss functions, and training strategies. Here we do not claim any optimality of our approach. There are many other possibilities to construct and train a NN for communication systems. For example, binary signals can be represented by binary, modulated symbols, integers or one-hot vectors. [15] showed that using binary information bits as input for P2P Gaussian channels can significantly reduce BER comparing with the state-of-the-art codes; [19] encoded messages to each user as a one-hot vector and demonstrated that the autoencoder learns to use superposition coding over the degraded BC. In addition, one can select different NN structures with different number of layers and hidden neurons. The selection of training SNR is not obvious as well. In terms of generalization, it would be desirable for a learned system that can operate at any SNR, regardless at which SNR it was trained. However, we have observed that this is in general not the case. Training at low SNR does not fit the high SNR regimes and vice versa. Since we choose the binary representation and treat the decoding process as a classification problem, the BEC loss function is a common choice. While for alternate data representations, the choice of loss function is less clear.

IV. Numerical Evaluations

In this section, we set rate $R_1 = R_2 = \frac{1}{3}$ and illustrate the performance of the deep learning-aided two-phase scheme and compare it with the state-of-the art coding schemes for F-AWGN-BC without feedback in [21] and with COF in [8], [9]. We take a Rayleigh distribution of each channel state with distribution function $\mathbb{F}(q, \sigma_s) = 1 - e^{-q^2/(2\sigma_s^2)}$, where $q \in [0, \infty)$ and σ_s is the scale parameter.

As shown in Fig. 2, we use solid lines to present the joint BER of two users defined in (2) and evaluate different SNRs by fixing the scale parameter $\sigma_s = 1$ and varying the noise power σ_n^2. With unit transmission power constraint, the average SNR is $2/\sigma_n^2$. Since we consider a symmetric channel and rate setting, the BER of each user is approximately the same. Thus, we only plot the BER of one of the users with dashed lines. Specifically, the deep learning-aided AE with COF introduced in Section III is demonstrated in red, the linear coding scheme with COF presented in [8], [9] is plotted in blue, and the LDPC based scheme without COF in [21] is illustrated in black. Note that for the linear coding scheme,

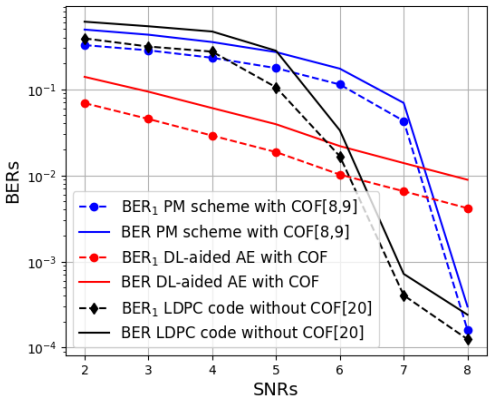


Fig. 2: Coding schemes performance comparison.

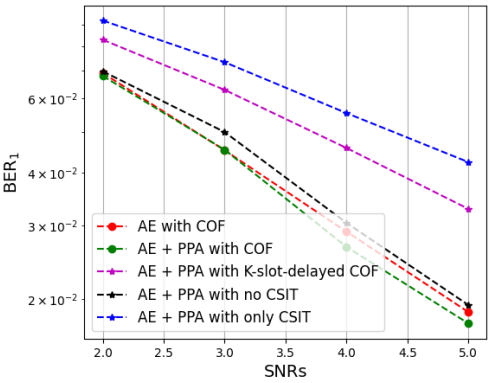
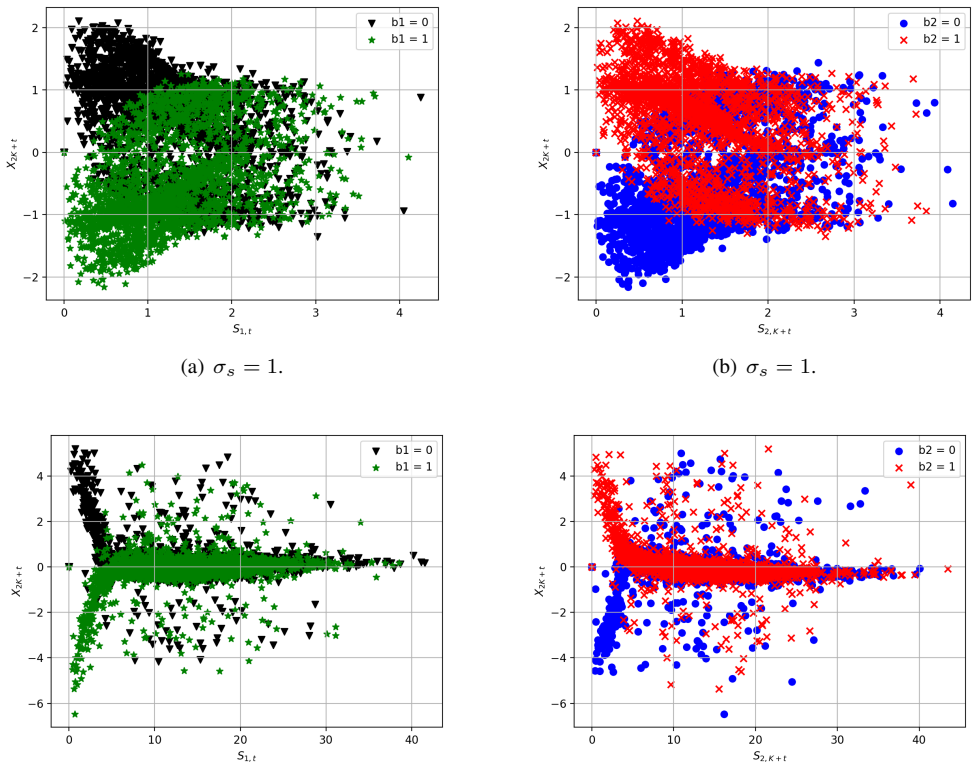


Fig. 3: Deep learning-aided two-phase scheme performance comparison.

the achievable rate $R_u = \frac{1}{N} \log M_u$ where $N \in \mathbb{N}$ is the number of iteration round and $M_u \in \mathbb{N}$ is the number of total messages destined to user u. Each user updates an estimate for the message at every iteration round $t \in \mathbb{N}$ and as the number of iterations increase, the BER decreases. For a fixed rate, N increases leading to M_u increases as well. The penalty is that more information bits are needed to represent each message which enlarges the block length. The LDPC scheme plotted here is based on the regular LDPC code. The BERs of the LDPC scheme reduces significantly when SNR $>$ 5dB and the performance can be further improved when employing irregular LDPC code. We notice that the deep learning-aided AE works well for low SNRs. At high SNR, it does not perform as well as the linear or LDPC codes without feedback, and should in principle be re-trained at this SNR. This is a drawback of such deep learning-based scheme, which is the generalization challenge.

In Fig. 3, we explore the benefit of a trainable Phase Power Allocation (PPA) weight and take different types of delayed feedback signals into consideration. Since the BER of each user has similar trend and slightly better performance than the joint BER, we only plot the joint BER in Fig. 3 to get a clear view. The red lines are from Fig. 2 and are used as the benchmark. Instead of fixing the power for each phase as some constant, we add a trainable PPA parameter and plot the BERs in green. Let the average power constraint $P = 1$. The training result shows that the RNN tends to allocate power $P_1 = 0.976$ to the "raw" information bits intended for each



(a) $\sigma_s = 1$. (b) $\sigma_s = 1$.

(c) $\sigma_s = 10$. (d) $\sigma_s = 10$.

Fig. 4: Codewords in phase 2 with respect to the channel fading in phase 1.

user in phase 1 and $P_2 = 1.058$ to the codewords in phase 2. Herein, we use "raw" to indicate that the information bits are not coded by the RNN and only modulated by BPSK followed by a power allocation parameter. Next, we consider the case where the received signals $Y_{1,t}, Y_{2,t}$ and channel state $(S_{1,t}, S_{2,t})$ experience longer delay and fix the delay to be equal to K time slots with BERs illustrated in magenta. The input size of the RNN in phase 2 becomes 10 and the PPA parameters are $P_1 = 1.036$ and $P_2 = 0.928$. This indicates that the encoder should assign slightly more power to the "raw" information bits to resist the channel noise when timely feedback is not available. The BERs in black presents the case where CSIT is not available but one-slot-delayed received signals $(Y_{1,t-1}, Y_{2,t-1})$ from two users are available at time slot t. The input size of the RNN in phase 2 becomes 8. Here the PPA parameters are similar to the AE with COF and $P_1 = 0.981$ and $P_2 = 1.038$. The BERs in blue demonstrates the case where only one-slot-delayed CSIT $(S_{1,t-1}, S_{2,t-1})$ from two users are available at time slot t. The input size of the RNN in phase 2 is still 8 and the PPA parameters are $P_1 = 1.07$ and $P_2 = 0.86$. Comparing the black, green and blue dashed lines in Fig. 3, it is interesting to notice that sending back the delayed received signals Y_u is more beneficial than the delayed CSIT S_u in terms of reliability. Considering these PPA parameters, we claim that the RNN tends to allocate slightly more power to the "raw" information bits when the signals feeding back to the system are not useful enough for the following encoding process under the training SNR= 2 dB and Rayleigh distribution with $\sigma_s = 1$.

We also notice that by fixing the noise power σ_n^2, as the scale parameter σ_s grows, the SNR increases. Thus, most of the signals transmitted in phase 1 can be amplified which leads to the BERs falling off. As the CSI knowledge at the transmitter is crucial in real-world wireless communication systems, we would like to investigate how the RNN makes use of the delayed CSIT in different scale cases. In the following, we continue with the model that only feeds back the delayed CSIT to the transmitter and compare the mechanisms of the RNN by fixing $\sigma_n^2 = 1.26$ and evaluating $\sigma_s = 1$ and 10. We train the AE based on average SNR $= 2$ dB for Fig. 4(a), 4(b) and SNR $= 22$ dB for Fig. 4(c), 4(d) separately and obtain models that give $\text{BER}_u = 0.09$ and $\text{BER}_u = 2.0 \times 10^{-5}$ respectively. Fig. 4(a) and 4(b) present the codewords in phase 2 with respect to the channel fading of user 1 in the first half of phase 1 and the channel fading of user 2 in the second half of phase 1 when $\sigma_s = 1$; Fig. 4(c) and 4(d) present the case where $\sigma_s = 10$. The codewords in phase 2 are the refinement of the information bits transmitted in phase 1, which are illustrated by different colors and markers in Fig. 4.

By Fig. 4, the RNN learns to i) allocate little power to the information bits that were erased in phase 1 (i.e., $S_{u,t} = 0$); ii) allocate more power to the information bits that experienced dramatic fading in phase 1 (i.e., $S_{u,t} \to 0$); iii) allocate less power to the information bits that were enhanced by the channel state in phase 1 (i.e., $S_{u,t} > 1$).

We trained the AE with a range of σ_s and observed a somewhat linear relationship between the channel state in phase 1 and the coded signals in phase 2. That is, the codewords in phase 2 are mapped to numbers approximated by linear functions $X_{1,2K+i} \approx \text{sgn}(1-2b_{1,i})(\gamma - \alpha S_{1,i})$ and $X_{2,2K+i} \approx \text{sgn}(2b_{2,i}-1)(\gamma - \alpha S_{2,K+i})$ for $S_{u,i} \in [0, \beta]$. The RNN distinguishes the users by changing the sign of the linear function. For $S_{u,i} \in [\beta, \infty]$, the codewords in phase 2 are mapped to numbers almost symmetrically along zero. In Fig. 4(a) and 4(b), $\gamma \approx 2, \alpha \approx \frac{4}{3}$, and $\beta \approx 1.5$; in Fig. 4(c) and 4(d), $\gamma \approx 5, \alpha \approx 1$, and $\beta \approx 5$. Based on the trained $\sigma_s = \{1, 2, \cdots, 15\}$, we noticed that $\gamma \in [2, 5]$, $\alpha \in [0.5, 1.5]$ and $\beta \in [1.5, 7]$. As σ_s grows, γ and β increase slowly. α is some number varying around 1. Also, as the distribution of the channel state becomes flatter, the linear relationship becomes more obvious. The relationship between γ, α, β and σ_s is still under investigation.

V. Conclusion

This paper presented a two-phase AE empowered by a deep RNN integrating GRU and LSTM for the symmetric F-AWGN-BC with COF. Comparisons with the state-of-the-art codes reveal competitive BER performance in the low SNR regime, although generalization to the high SNR regime remains a challenge as well as higher order modulations. We believe that this is the beginning of the investigations into deep learning for fading BCs with feedback. Our approach can provide potential performance improvements in terms of reliability and interesting insight about "good" communication schemes (i.e., power allocation and codewords construction) in scenarios where the optimal schemes are unknown (i.e., F-AWGN-BC with different types of feedback signals). We have identified that the delayed received signals are more helpful than the delayed CSIT for the encoding process in the proposed scheme and also observed that the codewords constructed by the AE in phase 2 have some linear relationship with respect to the Rayleigh distributed channel state. However, there are still a great number of open problems to be explored in the future, such as taking noisy feedback and imperfect CSI into consideration.

References

[1] 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, April 1966.

[2] 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, April 1966.

[3] L. Ozarow and S. Leung-Yan-Cheong, "An achievable region and outer bound for the Gaussian broadcast channel with feedback (Corresp.)," *IEEE Transactions on Information Theory*, vol. 30, no. 4, pp. 667–671, July 1984.

[4] S. R. Bhaskaran, "Gaussian broadcast channel with feedback," *IEEE Transactions on Information Theory*, vol. 54, no. 11, pp. 5252–5257, Nov 2008.

[5] L. V. Truong and H. Yamamoto, "Posterior matching for Gaussian broadcast channels with feedback," *arXiv e-prints*, p. arXiv:1404.2520, Apr 2014.

[6] N. Elia, "When bode meets Shannon: control-oriented feedback communication schemes," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1477–1488, 2004.

[7] S. B. Amor and M. Wigger, "Linear-feedback MAC-BC duality for correlated BC-noises, and iterative coding," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2015, pp. 1502–1509.

[8] S. Li, D. Tuninetti, and N. Devroye, "The fading Gaussian broadcast channel with channel state information and output feedback," in *2020 IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 1474–1479.

[9] ——, "A control-theoretic linear coding scheme for the fading Gaussian broadcast channel with feedback," in *2021 IEEE International Symposium on Information Theory (ISIT)*, 2021, pp. 1–6.

[10] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5g," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 74–80, 2014.

[11] S. Bi, R. Zhang, Z. Ding, and S. Cui, "Wireless communications in the era of big data," *IEEE Communications Magazine*, vol. 53, no. 10, pp. 190–199, 2015.

[12] T. O'Shea, T. Erpek, and T. Clancy, "Deep learning based MIMO communications," *ArXiv*, vol. abs/1707.07980, 2017.

[13] C. Nguyen, H. Dinh Thai, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," 10 2018.

[14] Q. Mao, F. Hu, and Q. Hao, "Deep learning for intelligent wireless networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 20, pp. 2595–2621, 2018.

[15] H. Kim, Y. Jiang, S. Kannan, S. Oh, and P. Viswanath, "Deepcode: Feedback codes via deep learning," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 194–206, 2020.

[16] W. Jiang and H. D. Schotten, "Deep learning for fading channel prediction," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 320–332, 2020.

[17] R.-F. Liao, H. Wen, J. Wu, H. Song, F. Pan, and L. Dong, "The Rayleigh fading channel prediction via deep learning," *Wireless Communications and Mobile Computing*, vol. 2018, pp. 1–11, 07 2018.

[18] M. Goutay, F. A. Aoudia, and J. Hoydis, "Deep reinforcement learning autoencoder with noisy feedback," 2019.

[19] E. Stauffer, A. Wang, and N. Jindal, "Deep learning for the degraded broadcast channel," *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pp. 1760–1763, 2019.

[20] D. Wu, M. Nekovee, and Y. Wang, "Deep learning-based autoencoder for m-user wireless interference channel physical layer design," *IEEE Access*, vol. 8, pp. 174 679–174 691, 2020.

[21] P. Berlin and D. Tuninetti, "LDPC codes for fading Gaussian broadcast channels," *Information Theory, IEEE Transactions on*, vol. 51, pp. 2173 – 2182, 07 2005.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[23] T. Gruber, S. Cammerer, J. Hoydis, and S. t. Brink, "On deep learning-based channel decoding," in *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, 2017, pp. 1–6.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[25] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014.

[26] D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986.

[27] J. Bergstra, R. Bardenet, B. Kégl, and Y. Bengio, "Algorithms for hyper-parameter optimization," 12 2011.

[28] S. Li, D. Tuninetti, and N. Devroye, "On the capacity region of the layered packet erasure broadcast channel with feedback," in *IEEE International Conference on Communications (ICC)*, May 2019, pp. 1–6.

[29] C. Jiang, Y. Chen, S. Chen, Y. Bo, W. Li, T. Wenxin, and J. Guo, "A mixed deep recurrent neural network for mems gyroscope noise suppressing," *Electronics*, vol. 8, p. 181, 02 2019.

[30] R. Cahuantzi, X. Chen, and S. Güttel, "A comparison of lstm and gru networks for learning symbolic sequences," 2021.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[32] L. N. Smith, "Cyclical learning rates for training neural networks," 2017.