KICS The Korean Institute of Communications and Information Sciences

Available online at www.sciencedirect.com

ScienceDirect

ICT Express 8 (2022) 25–30

www.elsevier.com/locate/icte

Hybrid neural coded modulation: Design and training methods

Sung Hoon Lim[a], Jiyong Han[a], Wonjong Noh[a], Yujae Song[b], Sang-Woon Jeon[c,*]

[a] School of Software, Hallym University, Chuncheon, Republic of Korea
[b] Department of Computer Software Engineering, Kumoh National Institute of Technology, Gumi 39177, Republic of Korea
[c] Department of Electrical and Electronic Engineering, Hanyang University, Ansan, Republic of Korea

Received 30 September 2021; received in revised form 31 December 2021; accepted 30 January 2022
Available online 9 February 2022

Abstract

We propose a hybrid coded modulation scheme which composes of inner and outer codes. The outer-code can be any standard binary linear code with efficient soft decoding capability (e.g. low-density parity-check (LDPC) codes). The inner code is designed using a deep neural network (DNN) which takes the channel coded bits and outputs modulated symbols. For training the DNN, we propose to use a loss function that is inspired by the generalized mutual information. The resulting constellations are shown to outperform the conventional quadrature amplitude modulation (QAM) based coding scheme for modulation order 16 and 64 with 5G standard LDPC codes.

Keywords: Machine learning; Neural networks; Modulation; Channel coding; Generalized mutual information

1. Introduction

Machine learning methods for channel coding is an emerging field which can help overcome many challenging problems in error-correction coding. In particular, end-to-end learning for designing encoders and decoders have been proposed in [1] that utilizes a deep neural network (DNN) autoencoder. An interesting insight of this approach is that the end-to-end structure does not utilize conventional quadrature amplitude modulation (QAM) constellations nor does it make use of the de-facto architecture, bit interleaved coded modulation (BICM) [2,3]. Instead, it is a clean slate approach to find the optimal encoder and decoder pair using a DNN. One benefit of this approach is that the encoder is not restricted to the suboptimal QAM constellation and may learn a better input distribution overall. End-to-end transceiver design utilizing DNNs has been applied in various contexts including additive white Gaussian noise (AWGN) channels [4–6], fast fading [7], intersymbol interference (ISI) channels [8], ultra low-latency [9], and model free design [10]. Other works have also used DNNs focusing on specific components such as decoder design [11–17] and constellation shaping for modulation [18]. Several other approaches to canonical problems have been proposed for feedback channels [19], quantized channel observations [20], joint source–channel coding [21], and the wiretap channel [22]. More recently, theoretical studies on end-to-end design have been given in [23]. While these approaches provide breakthroughs in various situations, one challenge in the end-to-end design approach is that the exponential growth on the number of codewords in code length makes learning end-to-end codes quite difficult for long codes.

In another line of work, hybrid architectures were proposed which consist of a DNN inner code (or modulator) that is concatenated to an outer linear code (e.g. turbo and low-density parity-check (LDPC) codes). For example, hybrid architectures were designed and applied for AWGN with radar interference [24], optical fiber communications [25], one-bit quantized AWGN channels [20], and AWGN channels [26]. The work of [26] also generalizes the decoder for iterative demodulation and decoding (IDD) and gives implementation results on software defined radios. In these approaches, the binary cross entropy (BCE) [27] metric was used to train the DNN which enables the inner code to be compatible with the outer linear code decoder, i.e., the DNN output is in the form of bit-level decoding metrics. An advantage of the hybrid structure approach is that it can benefit from learning a better constellation (shaping gain) while maintaining practical code lengths for error correction performance (coding gain).

* Corresponding author.
E-mail addresses: shlim@hallym.ac.kr (S. H. Lim), 43188@hallym.ac.kr (J. Han), wonjong.noh@hallym.ac.kr (W. Noh), yjsong@kumoh.ac.kr (Y. Song), sangwoonjeon@hanyang.ac.kr (S.-W. Jeon).

Peer review under responsibility of The Korean Institute of Communications and Information Sciences (KICS).

https://doi.org/10.1016/j.icte.2022.01.018

Following this approach, we propose a generic architecture that can take some off-the-shelf linear code (e.g. LDPC codes) and concatenate it with a DNN autoencoder inner-code for modulation. With this goal in mind, we design and train a DNN inner-code that is compatible for the linear channel code decoder, for example, to be compatible with the sum–product algorithm (SPA). Our main contribution in this direction is that we propose an approximated formulation of the generalized mutual information (GMI) [3,28] as a loss function that is tailored for learning the inner DNN encoder and decoder pair. We further outline some useful training techniques that we have learned through extensive evaluations.

In the numerical evaluations section, we provide our performance evaluations with our trained DNN inner code concatenated with a 5G standard LDPC code [29] and compare its performance with QAM based BICM systems for modulation order $M = 16$ and $M = 64$.

In the sequel, we define $\mathbb{C}$ and $\mathbb{F}_2$ as the complex and binary field. Random variables are denoted by upper-case letters X, and expected values are noted by $\mathsf{E}[X]$. We define $c^n := (c_1, \ldots, c_n)$, i.e., an n-length sequence (or vector) and will often re-index a sequence c^n by its jth subsequence $c^{(j)} = (c_1^{(j)}, \ldots, c_m^{(j)})$ of length m such that $c^n = (c^{(1)}, \ldots, c^{(N)})$, where $N = n/m$. The length of the subsequence will be noted in the context when needed.

2. System model

2.1. Communication system and channel model

Consider a memoryless channel $(\mathcal{X}, p_{Y|X}, \mathcal{Y})$ which consists of an input alphabet $\mathcal{X}$, a receiver alphabet $\mathcal{Y}$, and a collection of conditional distributions $p_{Y|X}$.

A $(2^{NR}, N)$ code for the channel consists of a message set $[1 : 2^{NR}]$, an encoder which maps each message $\mathsf{m} \in [1 : 2^{NR}]$ to a sequence $x^N(\mathsf{m}) \in \mathcal{X}^N$, and a decoder that assigns estimates $\hat{\mathsf{m}}(y^N) \in \mathcal{M}$ to each received sequence $y^N \in \mathcal{Y}^N$. From the memoryless channel assumption we have $P(y^N|x^N) = \prod_{i=1}^N p_{Y|X}(y_i|x_i)$ where $p_{Y|X}$ is the AWGN channel, i.e.,

$$y_i = x_i + Z_i, \quad i \in [1 : n] \tag{1}$$

where $Z_i \sim \mathcal{CN}(0, \sigma^2)$ and $\mathcal{Y} = \mathcal{X} = \mathbb{C}$. We let $\sigma^2 = 1/\mathsf{SNR}$ where SNR is the signal to noise power ratio and we assume that the input is subject to an average power constraint $\frac{1}{n}\sum_{i=1}^n |x_i|^2 \le 1$. Note that the underlined channel distribution of the AWGN channel is thus given by,

$$p_{Y|X}(y|x; \sigma^2) = \frac{1}{\pi\sigma^2} \exp\left(\frac{-|y-x|^2}{\sigma^2}\right).$$

2.2. Bit interleaved coded modulation

A coded modulation architecture specializes the generic communication system as follows. The messages are considered as bit sequences b^k, e.g., binary expansions of $\mathsf{m} \in \mathcal{M}$. The encoder is specialized into two sub-components, a binary code and a bit-to-symbol mapper (modulator), and the decoder is specialized into two sub-components, symbol-to-bit level log-likelihood ratio (LLR) mapper (demodulator) and a binary channel code decoder. The input alphabet, i.e., the constellation set $\mathcal{X}$ is fixed as a discrete subset of $\mathbb{C}$ of size M, where M is the modulation order and $m = \log(M)$.

Specifically, we denote a length $n = mN$ binary code codebook by $\mathcal{C}$ which maps k binary inputs $b^k \in \mathbb{F}_2^k$ to n binary sequences $c^n(b^k) \in \mathbb{F}_2^n$. The rate of the binary code is $R_{\mathcal{C}} = \frac{k}{n}$. The modulator then maps the jth m-length subsequence $c^{(j)}$ of c^n to a point $x^{(j)} \in \mathcal{X}$ by the function

$$x^{(j)} = \mu(c^{(j)}), \ j = 1, \ldots, N.$$

The theoretical performance of a coded modulation strategy can be measured by the *generalized mutual information (GMI)*[1] in the form of

$$I^{\text{gmi}}(X; Y) = \mathsf{E}\left[\log \frac{q(X, Y)}{\sum_{x' \in \mathcal{X}} P_X(x') q(x', Y)}\right], \tag{2}$$

where $q(x, y)$ is a (symbol-level) decision metric. Note that when $q(x, y) = p_{Y|X}(y|x)$, the GMI is equal to the coded modulation capacity [3]. In this work, we are particularly interested in a bit-level decision metric based coded modulation strategy to be compatible with bit-level decoders (e.g. sum–product algorithm). To this end, we define a generic bit-level metric based demodulator that maps the jth received signal $y^{(j)} \in \mathcal{Y}$ to a set of m bit probabilities by the function

$$\phi(y^{(j)}) = p^{(j)}, \ j = 1, \ldots, N$$

where $p^{(j)} = (p_1^{(j)}, \ldots, p_m^{(j)})$, $p_i^{(j)} \in [0, 1]$ is the demodulated bit probability estimate of $c_i^{(j)}$.

In the following section, we give a detailed description of our proposed architecture and explain how we specialize (2) for bit-level decoding metrics.

3. Neural network

In this section, we give a detailed description of our proposed architecture for the DNN components μ_θ and ϕ_θ.

A description of the proposed coded modulation system is given in Fig. 1. Note that in the figure, the modulator and demodulator are DNNs specified by parameters θ. Our goal is to find a modulator and demodulator pair $(\mu_\theta, \phi_\theta)$ utilizing a DNN architecture that finds the parameters θ to maximize the GMI. Once the pair $(\mu_\theta, \phi_\theta)$ is fully trained, we treat it as an inner-code that is combined with a binary linear code (e.g. LDPC) as the outer code.

3.1. DNN inner-encoder (modulator)

The neural encoder μ_θ comprises of several layers. The first input layer is a tanh layer, i.e., a fully connected linear layer with tanh activation functions. Similarly, the following hidden layers are rectified linear unit (ReLU) layers. The final two-layers are a vanilla linear layer followed by a normalization layer to satisfy the average power constraint. Recall that the input to the overall DNN-encoder are m-length subsequences

[1] This is a lower bound to the GMI given in [3] with $s = 1$.

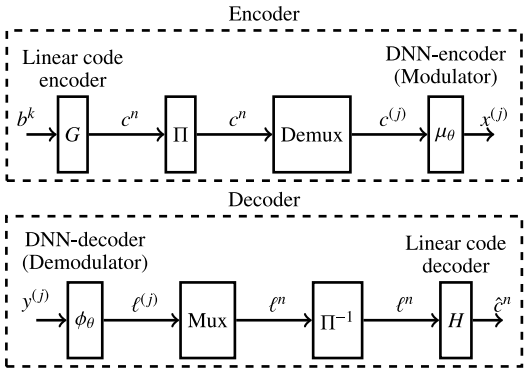

Fig. 1. Proposed coded modulation system architecture. At the encoder, binary information bits b^k are encoded into a binary linear code c^n, interleaved, and demuxed into subsequences of length m $c^{(j)} = \{c_1^{(j)}, \ldots, c_m^{(j)}\}$, which is encoded by the DNN-encoder to send the jth input symbol $x^{(j)}$. The decoding procedure is done in the reverse order, where $\ell^{(j)}$ are the demodulated LLRs (12) of the bits mapped to the jth symbol $x^{(j)}$, and ℓ^n is the multiplexed n length LLRs. In the decoder, the feature mapping function ψ (5) before ϕ_θ and the $p^{(j)}$ to LLR $\ell^{(j)}$ transition after ϕ_θ are omitted for visual clarity.

of a binary codeword $c^{(j)} \in \mathbb{F}_2^m$. The final normalization layer is applied as follows. Define $\tilde{\mathcal{C}}_\mu$ as

$$\tilde{\mathcal{C}}_\mu = \{\tilde{x} : \tilde{x} = \tilde{\mu}_\theta(c^m),\ c^m \in \mathbb{F}_2^m\}, \tag{3}$$

where $\tilde{\mu}_\theta(c^m)$ is the output of the last linear layer, i.e., μ_θ excluding the normalization layer. Let $\tilde{\eta}$ and $\tilde{\sigma}^2$ be the sample mean and variance of $\tilde{\mathcal{C}}_\mu$, respectively. Then, the normalization layer outputs are given by

$$x = \frac{\tilde{x} - \tilde{\eta}}{\tilde{\sigma}}, \tag{4}$$

where x is the output of the normalization layer. Thus, the final normalization layer makes the constellation points satisfy the average power constraint. We note that the number of points in $\tilde{\mathcal{C}}_\mu$ is 2^m and the points are fixed once the parameters are fixed. In the training stage, the sample mean $\tilde{\eta}$ and variance $\tilde{\sigma}^2$ is updated for every parameter update (e.g. for each mini-batch stochastic gradient descent (SGD) update). After training, the normalization layer does not need any update since the parameters are then fixed. Every layer except the final output layers of the encoder and decoder are implemented with batch normalization [30].

3.2. DNN inner-decoder (demodulator)

The DNN-decoder (demodulator) has the following structure. First, the input of the DNN-decoder is formulated by a feature mapping function applied on the received signal y. Upon receiving the jth channel output $y^{(j)}$, the received symbol is mapped to the logarithm of the channel distribution $p_{Y|X}$ as an M dimensional vector, i.e.,

$$\psi(y^{(j)}, \sigma^2) = [\ln p_{Y|X}(y^{(j)}|x; \sigma^2) : x \in \mathcal{C}_\mu]. \tag{5}$$

The DNN-decoder input $\psi(y^{(j)}, \sigma^2)$ is passed through some ReLU layers, and the final output layer is given by a sigmoid layer with m output units representing the m bit probabilities

$$\phi_\theta(y^{(j)}) = [p_1^{(j)}, \ldots, p_m^{(j)}], \tag{6}$$

where $p_i^{(j)} \in [0, 1]$ and we define $\phi_\theta(y^{(j)})$ as a shorthand notation for $\phi_\theta(\psi(y^{(j)}, \sigma^2))$.

3.3. Loss function

For the loss function, we use an approximate variant of (2). To this end, we first approximate (2) by

$$\hat{I}^{\text{gmi}}(X; Y) := \mathsf{E}\left[\log \frac{q(X, Y)}{\sum_{x' \in \mathcal{C}_\mu} P_X(x') P_{Y|X}(Y|x')}\right] \tag{7}$$

$$= M + \mathsf{E}\left[\log \frac{q(X, Y)}{\sum_{x' \in \mathcal{C}_\mu} P_{Y|X}(Y|x')}\right] \tag{8}$$

$$= M + \mathsf{E}\left[\log \frac{\prod_{i=1}^m q(C_i, Y)}{\sum_{x' \in \mathcal{C}_\mu} P_{Y|X}(Y|x')}\right], \tag{9}$$

where we define the metric $q(x, y) = \prod_{i=1}^m q(c_i, y)$. Note that we have defined the symbol level metric $q(x', Y)$ in the denominator of (2) by its optimal value $P_{Y|X}$ in (7) and we further choose the symbol level decision metric $q(x, y)$ as a product of bit-level metrics $\prod_{i=1}^m q(c_i, y)$, which is a function of the output of the DNN-decoder. We note that the marginalization in the denominator of (9) is a function of the constellation points $\mathcal{C}_\mu$. Also, we treat the bit-level decision metric $q(c_i, y)$ to be estimates of $p(y|c_i)$ in product form to be compatible with linear code decoders, in particular, the sum–product algorithm.

In the following, we explain how we integrate the metric (9) with our proposed architecture. Firstly, the channel input symbols are chosen as the outputs of the DNN encoder given by

$$x = \mu_\theta(c_1, \ldots, c_m),$$

which results in $p_X(x) = 1/M$. For the decoding metric mapping, recall that our DNN decoder outputs are given by $\phi_\theta(y^{(j)})$ defined in (6). In our proposed architecture, the decision metric is defined by

$$q_\theta(c_i, y) = \begin{cases} p_i & \text{if } c_i = 1 \\ 1 - p_i & \text{otherwise.} \end{cases} \tag{10}$$

We note that the metric $q_\theta(c_i, y)$ is a function of the DNN parameters θ since p_i is the output of the overall DNN. When it is clear in the context, we will omit the subscript θ and use $q(c_i, y)$ for brevity.

The final step is to approximate $\hat{I}^{\text{gmi}}(X; Y)$ by its sample mean given as

$$\mathcal{L}(\theta) = M + \sum_{j=1}^N \frac{1}{N} \log \frac{\prod_{i=1}^m q_\theta(c_i^{(j)}, y^{(j)})}{\sum_{x' \in \mathcal{C}_\mu} P_{Y|X}(y^{(j)}|x')}, \tag{11}$$

where $c_i^{(j)}$ is the ith bit of the jth input symbol, i.e., $x^{(j)} = \mu_\theta(c_1^{(j)}, \ldots, c_m^{(j)})$ and $y^{(j)}$ is the channel output of the jth input symbol $x^{(j)}$. The DNN parameters θ are trained to maximize (11).

As a final note in this section, we point out some differences between the binary cross-entropy (BCE) loss [27] often used for binary classification and the loss given in Eq. (9). In the Appendix, we show that the BCE loss is equal to a specialized GMI (up to a constant difference) when the metric is defined as an estimate of $p(c_i|y)$ while assuming that the bit probabilities are conditionally independent. We note that the conditional independence assumption is not true in general due to the linear encoding structure. Further improvements to account for the linear structure in the demodulation stage can be done by integrating iterative decoding and demodulation (IDD) as in [26]. In our approach, we define our metric as an approximation of $p(y|c_i)$ in formulating the GMI which results in an additional term represented by the numerator inside the expectation of (9). We note that the numerator term is a function of the DNN modulation constellation points which in turn makes it a function of the DNN parameters. Our proposed loss function explicitly utilizes the marginal $p(y)$ in the loss function and produces estimates of $p(y|c_i)$ that is directly compatible with SPA. Performance comparison on the BCE loss and our proposed metric are given in Section 4.

3.4. Concatenation with linear codes

Once the DNN is trained, we can combine it with outer linear codes. For ease of presentation, we will focus on LDPC codes, however, the structure can be combined with any binary linear code with efficient soft decoding algorithms. Let $c^n \in \mathcal{C}$ be a linear code codeword and assume $n = Nm$. The codewords are interleaved and demultiplexed into m-bits $c^{(j)} = (c_1^{(j)}, \ldots, c_m^{(j)})$. The jth subsequence is encoded by the DNN-encoder $\mu_\theta(c^{(j)})$, then sent through the channel to receive $y^{(j)}$, and finally decoded at the DNN-decoder $p^{(j)} = \phi_\theta(\psi(y^{(j)}, \sigma^2))$. The output of the decoder $p^{(j)}$ are then converted to decoding metrics $q(c_i^{(j)}, y^{(j)})$, $i = 1, \ldots, m$ (10) for training. For channel code decoding, the DNN outputs are translated to log-likelihood ratios (LLRs) by

$$\ell_i^{(j)} = \ln \frac{1 - p_i^{(j)}}{p_i^{(j)}}, \quad j = 1, \ldots, N, \; i = 1, \ldots, m. \tag{12}$$

Then, the LLRs are multiplexed into n-sequences and is decoded by the linear code decoder, e.g., sum–product algorithm.

4. Training methods and numerical evaluations

The evaluations in this section have been implemented using the Pytorch [31] framework. In our evaluations, we use the 5G standard LDPC codes [29] with BG=1 and $Z_c = 24$, rate 1/2 LDPC codes which translate to length 1104 (1056 after puncturing) bit codewords and use the sum–product decoding algorithm with 50 iterations for decoding of the outer code.

Training of the DNN inner code is done independent of the linear channel code. That is, at training stage, c^m are simply generated randomly and independently. For testing, we use the encoded bits from the linear code. We use a two-stage training process which we will refer to as the first stage training (pretraining) and the second stage training. The training parameters for each training steps are summarized in Table 1.

Table 1
DNN training parameters for 1st stage and 2nd stage training.

Parameters	$M = 16$	$M = 64$
Enc. hidden units	[16, 64, 32]	[64, 128, 128, 128]
1st stage dec. hidden units	[128]	[128]
2nd stage dec. hidden units	[128]	[64, 128]
Training SNR	7 dB	11.5 dB
1st stage batch size	$M \times 20$	$M \times 20$
2nd stage batch size	$M \times 1600$	$M \times 1600$
Number of samples	$M \times 4800$	$M \times 3200$
Learning rate	$0.1 \sim 0.001$	$0.1 \sim 0.001$
1st stage optimizer	AdamW (0.01)	AdamW (0.2)
2nd stage optimizer	Adam	Adam
Activation functions	tanh, ReLU	tanh, ReLU

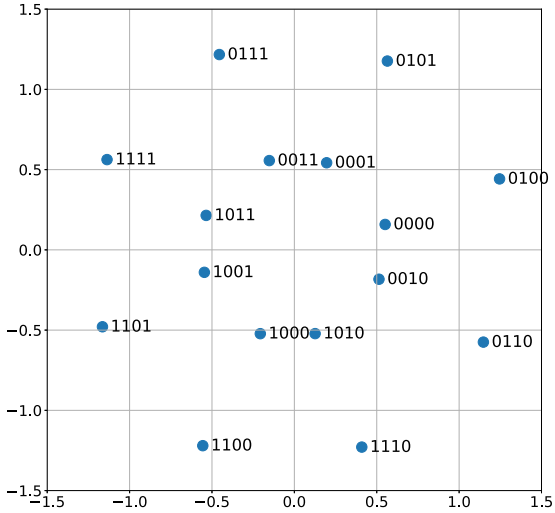

Fig. 2. Learned constellation $\mathcal{C}_\mu$ for $M = 16$ with GMI maximization.

In the first stage of training, we keep the decoding structure simple compared to the 2nd stage training, we have a relatively low batch size, and use the AdamW [32] optimizer with L_2 regularization with regularization coefficients given in the parenthesis of Table 1. These choice of hyper parameters help the optimizer escape local minima or saddle points to find a good initial constellation shape for further precise training at the second stage.

The second stage of training is used to fine-tune the DNN and train a better decoder via transfer learning. Firstly, we exchange the decoder with a randomly initialized larger set of decoding layers (while keeping the pretrained encoder). We train both the encoder and decoder parameters with a larger batch size and use the Adam optimizer without regularization.

In the following we present our numerical evaluations. In Figs. 2 and 3, we show examples of trained constellation points $\mathcal{C}_\mu$ for $M = 16$ and $M = 64$, respectively. Notice that the constellation points have round edges compared to QAM constellations resulting in better shaping gains. It is interesting to note that the constellations seem to have structures with "Gray mapping" like labels, for example, the most significant

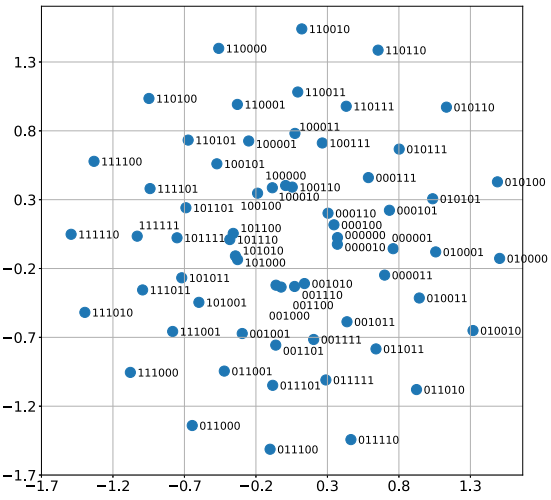

Fig. 3. Learned constellation $\mathcal{C}_\mu$ for $M = 64$ with GMI maximization.

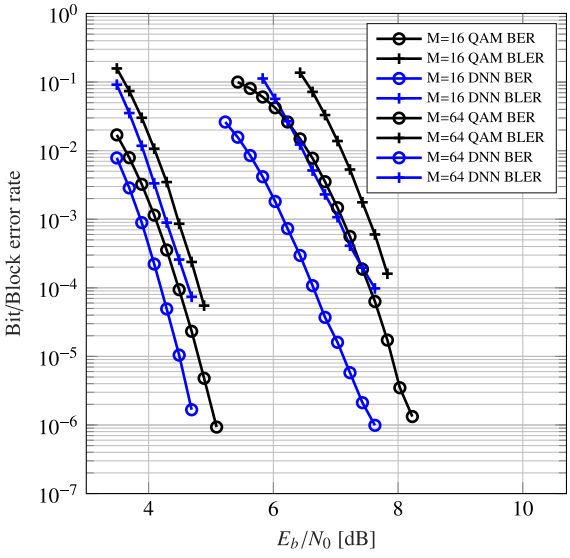

Fig. 4. Performance evaluations of block error and bit error rates for $M = 16$ and $M = 64$. The plots compare our hybrid DNN strategy which maximizes GMI with a standard QAM coding scheme. The plots are results from the constellations given in Figs. 2 and 3.

bits (MSB) are distinguished by the left-half and the right-half of the plane. In Fig. 4, we compare the bit-error rate (BER) and block-error rate (BLER) of our hybrid coded modulation strategy with the conventional QAM based BICM strategy. For hybrid coded modulation strategy provides approximately 0.3 dB gain and 1 dB gain over the conventional QAM based strategy for $M = 16$ and $M = 64$, respectively.

Next, to highlight the effectiveness of our proposed loss function Eq. (11) and training method, we compare the performance of two trained hybrid architectures, one using the GMI loss function Eq. (11) and one trained with the conventional BCE metric similar to the approach in [26]. The simulation environment and base codes are equivalent to Fig. 4. In Fig. 5 we can see that the performance of our proposed training method has the best overall performance while the BCE loss

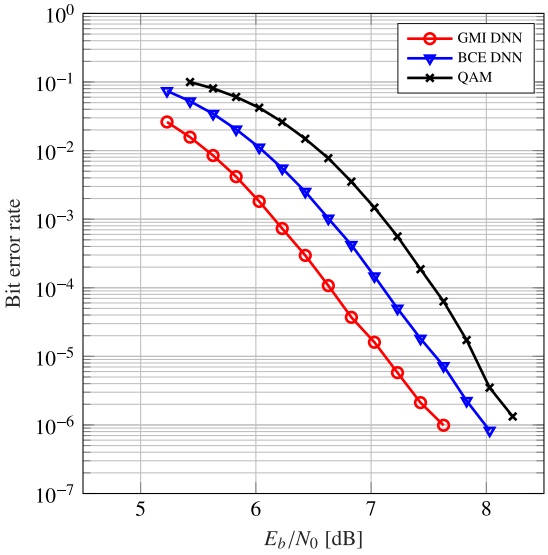

Fig. 5. Evaluation and comparison between GMI loss in (11) and BCE for $M = 64$.

function approach still learns a better constellation compared to the conventional QAM based scheme.

5. Conclusion

In this paper, we propose a hybrid BICM architecture that combines binary linear codes with DNN based inner-codes. We formulate a GMI inspired loss function and design the architecture to be compatible with conventional linear codes and soft decoding algorithms. The inner DNN based code offers shaping gain compared to standard QAM based approaches while maintaining coding gains from practical length codes resulting in overall better error correcting performance. Moreover, we provide some useful training methods for optimizing the DNN. Numerical results show that the proposed hybrid approach outperforms the QAM based BICM architectures which can provide gains for future high-order modulation communication systems.

Some interesting future research directions would be to extend the framework to fading channels, multiple antennas, and multi-user channels.

CRediT authorship contribution statement

Sung Hoon Lim: Proposal of the main idea, Development of the detailed proposed scheme. **Jiyong Han:** Numerical simulations, Discussion of numerical results. **Wonjong Noh:** Paper writing, Reference survey. **Yujae Song:** Numerical simulations, Discussion of numerical results. **Sang-Woon Jeon:** Overall paper organization, Performance comparison and analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research was supported by the Hallym University Research Fund, 2019, Republic of Korea (HRF-201910-012).

Appendix

Consider the joint distribution $p_{C^m,X,Y}(c^m, x, y) = p(c^m)p(x|c^m)p(y|x)$ where $p(c^m) = 1/2^m$, $c^m \in \mathbb{F}_2^m$, $p(x|c^m)$ is the bit-to-symbol mapping distribution and $p(y|x)$ is the channel distribution. We define a decision metric in conditional distribution form as

$$\tilde{q}(x|y) \coloneqq \prod_{i=1}^{m} \tilde{q}(c_i|y). \tag{13}$$

Then, specializing the GMI function with the metric (13), we have

$$\begin{aligned} I^{\text{gmi}}(X;Y) &= H(X) + \mathsf{E}\left[\log \tilde{q}(X|Y)\right] \\ &= H(X) + \mathsf{E}\left[\log \prod_{i=1}^{m} \tilde{q}(C_i|Y)\right] \\ &= H(X) + \sum_{i=1}^{m} \mathsf{E}\left[\log \tilde{q}(C_i|Y)\right] \\ &= H(X) + \sum_{i=1}^{m} \mathsf{E}\left[\sum_{c_i} p(c_i|Y) \log \tilde{q}(c_i|Y)\right] \\ &= H(X) - \sum_{i=1}^{m} \text{BCE}(p(c_i|Y), \tilde{q}(c_i|Y)), \end{aligned}$$

where the expectation is with respect to $p_{C^m,X,Y}$ and $\text{BCE}(p, q)$ is the binary cross entropy function. Thus, minimizing the BCE is equivalent to maximizing the GMI with decision metric in (13).

References

[1] T. O'Shea, J. Hoydis, An introduction to deep learning for the physical layer, IEEE Trans. Cogn. Commun. Netw. 3 (4) (2017) 563–575.
[2] G. Caire, G. Taricco, E. Biglieri, Bit-interleaved coded modulation, IEEE Trans. Inf. Theory 44 (3) (1998) 927–946.
[3] A. Martinez, A. G. i Fabregas, G. Caire, F.M.J. Willems, Bit-interleaved coded modulation revisited: A mismatched decoding perspective, IEEE Trans. Inf. Theory 55 (6) (2009) 2756–2765.
[4] S. Dörner, S. Cammerer, J. Hoydis, S. Ten Brink, Deep learning based communication over the air, IEEE J. Sel. Top. Signal Process. 12 (1) (2018) 132–143.
[5] Y. Jiang, H. Kim, H. Asnani, S. Kannan, S. Oh, P. Viswanath, Turbo autoencoder: Deep learning based channel codes for point-to-point communication channels, 2019, Preprint available at https://arxiv.org/abs/1911.03038.
[6] H. He, S. Jin, C.-K. Wen, F. Gao, G.Y. Li, Z. Xu, Model-driven deep learning for physical layer communications, IEEE Wirel. Commun. 26 (5) (2019) 77–83.
[7] S. Park, O. Simeone, J. Kang, Meta-learning to communicate: Fast end-to-end training for fading channels, in: Proc. IEEE International Conference On Acoustics, Speech And Signal Processing, ICASSP, 2020, pp. 5075–5079.
[8] Y. Zhang, H. Wu, M. Coates, On the design of channel coding autoencoders with arbitrary rates for ISI channels, IEEE Wirel. Commun. Lett. (Early Access) (2021).
[9] Y. Jiang, H. Kim, H. Asnani, S. Kannan, S. Oh, P. Viswanath, LEARN codes: Inventing low-latency codes via recurrent neural networks, IEEE J. Sel. Areas Inf. Theory 1 (1) (2020) 207–216.
[10] F.A. Aoudia, J. Hoydis, Model-free training of end-to-end communication systems, IEEE J. Sel. Areas Commun. 37 (11) (2019) 2503–2516.
[11] E. Nachmani, E. Marciano, L. Lugosch, W.J. Gross, D. Burshtein, Y. Be'ery, Deep learning methods for improved decoding of linear codes, IEEE J. Sel. Top. Signal Process. 12 (1) (2018) 119–131.
[12] O. Shental, J. Hoydis, "Machine LLRning": Learning to softly demodulate, in: Proc. IEEE Globecom Workshops, 2019, pp. 1–7.
[13] T. Koike-Akino, Y. Wang, D.S. Millar, K. Kojima, K. Parsons, Neural turbo equalization: Deep learning for fiber-optic nonlinearity compensation, J. Lightwave Technol. 38 (11) (2020) 3059–3066.
[14] Y. He, M. Jiang, X. Ling, C. Zhao, Robust BICM design for the LDPC coded DCO-OFDM: A deep learning approach, IEEE Trans. Commun. 68 (2) (2020) 713–727.
[15] N. Shah, Y. Vasavada, Neural layered decoding of 5G LDPC codes, IEEE Commun. Lett. 25 (11) (2021) 3590–3593.
[16] J. Dai, K. Tan, Z. Si, K. Niu, M. Chen, H.V. Poor, S. Cui, Learning to decode protograph LDPC codes, IEEE J. Sel. Areas Commun. 39 (7) (2021) 1983–1999.
[17] E. Nachmani, L. Wolf, Autoregressive belief propagation for decoding block codes, 2021, Preprint available at https://arxiv.org/abs/2103.11780.
[18] M. Stark, F. Ait Aoudia, J. Hoydis, Joint learning of geometric and probabilistic constellation shaping, in: Proc. IEEE Globecom Workshops, 2019, pp. 1–6.
[19] H. Kim, Y. Jiang, S. Kannan, S. Oh, P. Viswanath, Deepcode: Feedback codes via deep learning, IEEE J. Sel. Areas Inf. Theory 1 (1) (2020) 194–206.
[20] E. Balevi, J.G. Andrews, Autoencoder-based error correction coding for one-bit quantization, IEEE Trans. Commun. 68 (6) (2020) 3440–3451.
[21] Y.M. Saidutta, A. Abdi, F. Fekri, Joint source-channel coding over additive noise analog channels using mixture of variational autoencoders, IEEE J. Sel. Areas Commun. 39 (7) (2021) 2000–2013.
[22] K.-L. Besser, P.-H. Lin, C.R. Janda, E.A. Jorswieck, Wiretap code design by neural network autoencoders, IEEE Trans. Inf. Forensics Secur. 15 (2020) 3374–3386.
[23] N. Weinberger, Generalization bounds and algorithms for learning to communicate over additive noise channels, IEEE Trans. Info. Theory (2021) 1–36.
[24] F. Alberge, Deep learning constellation design for the AWGN channel with additive radar interference, IEEE Trans. Commun. 67 (2) (2019) 1413–1423.
[25] R.T. Jones, M.P. Yankov, D. Zibar, End-to-end learning for GMI optimized geometric constellation shape, in: Proc. European Conference On Optical Communication, 2019, pp. 1–4.
[26] S. Cammerer, F.A. Aoudia, S. Dörner, M. Stark, J. Hoydis, S. Ten Brink, Trainable communication systems: Concepts and prototype, IEEE Trans. Commun. 68 (9) (2020) 5489–5503.
[27] K.P. Murphy, Machine Learning: A Probabilistic Perspective, second ed., MIT Press, Cambridge, MA, 2006.
[28] N. Merhav, G. Kaplan, A. Lapidoth, S. Shamai Shitz, On information rates for mismatched decoders, IEEE Trans. Inf. Theory 40 (6) (1994) 1953–1967.
[29] 3GPP, Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification, Technical specification (TS), 3rd Generation Partnership Project (3GPP).
[30] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proc. International Conference On Machine Learning, 2015, pp. 448–456.
[31] A. Paszke, et al., PyTorch: An imperative style, high-performance deep learning library, in: Proc. Advances In Neural Information Processing Systems 32, 2019, pp. 8024–8035.
[32] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, in: Proc. International Conference On Learning Representations, 2019, pp. 1–8.