Evaluation of Neural Demappers for Trainable Constellation in an End-to-End Communication System

Nazmul Islam and Seokjoo Shin*
Dept. of Computer Engineering, Chosun University
Gwangju, 61452 South Korea
nazmul@chosun.kr, *sjshin@chosun.ac.kr (Corresponding author)

***Abstract*— Conventional M-ary Quadrature Amplitude Modulation (M-QAM) constellation designs such as rectangular constellation, are based on mathematical data and estimated channel models to achieve equal probability of error for transmitted bits in communication systems. However, these designs are suboptimal as they are not receptive to practical channel conditions or system performance due to their fixed lattice structure, and their performance degrades with a higher number of bits per symbol. Deep learning (DL) based end-to-end communication systems can be utilized to circumvent these challenges for better overall performance. Such systems are implemented as deep neural network (DNN) autoencoders, where trainable constellations and neural demappers (ND) can be jointly trained to achieve optimum constellation design for a higher data rate communication system. In this study, we evaluate the performance of two NDs that implement trainable constellation design and compared them with two baseline demapping algorithms in an end-to-end communication system. In the analysis, the NDs outperformed the baseline demappers for higher bits per symbol transmission, and trainable constellation design corresponding to 1024-QAM achieved the highest gain of 0.8 dB compared to the baseline demappers.**

Keywords— Constellation, digital communication, end-to-end learning, machine learning, modulation, wireless communication.

I. INTRODUCTION

Traditional communication system designs are block-based, where each block (such as encoding, modulation, channel correction) can be individually analyzed and optimized using stationary and linear mathematical models. However, despite their solid foundation in statistics and information theory, block-based communication systems have suboptimal point-to-point performance and are far from approaching the Shannon limit as they are imperfect with non-linear properties [1]. In the pursuit of approaching Shannon limit and achieving beyond fifth-generation (5G) communication, many researchers have been adapting various Machine Learning (ML) technologies in wireless communication systems due to the recent popularity of Artificial Intelligence (AI). Owing to that, [2] first proposed physical layer (PHY) of wireless communication system as a Deep Learning (DL) architecture, known as the autoencoder (AE) [3]. It is termed as end-to-end learning of communication system which replaces the conventional block-based system with a single Deep Neural Network (DNN) implemented as the transmitter, channel, and receiver, and trained to reproduce the input as the output as shown in Fig. 1. The main drawback of this approach is the need for a mathematical channel model for training, which makes the implementation to real channels particularly challenging. To circumvent this challenge, authors in [4], [5] studied learning of differentiable channel model in the form of a generative adversarial network (GAN) for supervised autoencoder training, however they did not show practical implementations. Authors in [6] proposed alternating training of the AE, where the receiver is trained with supervised learning and the transmitter is trained by reinforcement learning (RL), alternatively. This was later extended in [7] with a noisy feedback system. The authors evaluated the communication system over Additive White Gaussian Noise (AWGN) and Rayleigh fading channels, which respectively showed comparable and improved performance compared to training with perfect feedback. Authors in [8] proposed differentiable iterative demapping and decoding (IDD) in an extensive study of the end-to-end communication system and verified its significant gain using software defined radio (SRD) over the AWGN channel.

Additionally, in scenarios where the channel model is unknown, highly complex, or challenging to optimize, traditional receiver algorithms that rely on mathematical modeling may not be effective as they are based on estimated channel model. Neural demappers (ND) at the receiver have the potential to perform well in these scenarios by employing trainable constellation designs at the mapper as implemented in [6]–[8]. For traditional digital communication systems, various constellation schemes have been developed over the last decades including the M-ary quadrature amplitude modulation (M-QAM) schemes. These modulation schemes use constellation design to map data stream into complex in-phase and quadrature (IQ) planes with different amplitude and phase shift. The constellation designs are usually optimized by adjusting the location of the constellation points (geometric shaping) or probabilities of occurrence (probabilistic shaping). However, these designs become suboptimal as the M-QAM order increases (higher bits per symbol) as they are not receptive to channel conditions or system performance due to their fixed geographic lattice structure. Therefore, implementing a communication system with trainable constellation at the mapper and DNN-based demapper would enable optimizing the constellation design to maximize the mutual information of channel input and output in an end-to-end communication system [8].

Though these approaches are novel and theoretically very appealing, to our best knowledge, there has not been any comparative studies considering various NDs and conventional demapping algorithms that are used in practice. Therefore, in this study we have implemented two conventional demapping algorithms with M-QAM constellation design and two NDs with trainable constellation design for analysis. To begin with, we have performed comparative analysis between the two conventional

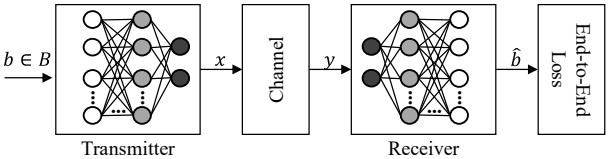

Fig. 1. Trainable end-to-end communication system as AE.

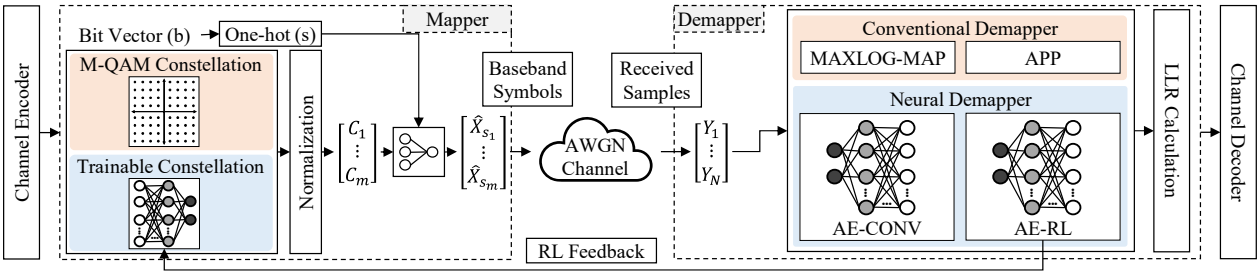

Fig. 2. Illustration of end-to-end communication system with trainable constellation and neural demapper.

demapping algorithms using M-QAM constellation design and later considered them as baseline for the comparative analysis of the two NDs. For the implementation we have used NVIDIA Sionna [9] which is an open-source link-level simulator for communication systems with ML integration and computation. It provides 5G standard modules and supports GPU-accelerated function for ML implementations in wireless communication with practical benchmark results.

The rest of the paper is organized as follows. Section II presents the system architecture for the experiment; section III presents the results and discussion and finally the conclusion and future work.

II. System Model

To transmit complex-valued signals using the conventional M-QAM modulation, a set of constellation points is defined for a specific M-QAM modulation scheme. The $M = 2^n$, is the order of M-QAM where n is the number of bits per symbol. Before transmitting the input data stream over the channel, it is first mapped into constellation points, and the data rate is dependent on the geometric shape of the constellation design and its probability distribution. Each point within the constellation is associated with a unique binary code, with each bit indicating a specific index of the constellation point. The position of the n^{th} constellation point corresponds to its assigned bit label within the array. For the trainable constellation design, the mapper takes a binary tensor as input and maps it to the DNN-based trainable constellation points determined by the Signal to noise Ratio (SNR) based on the binary code. The SNR corresponds to the energy of each information-bit over spectral density ratio of noise power. Fig. 2. shows an end-to-end communication system that utilizes M-QAM constellation and trainable constellation for baseline demappers and NDs, respectively. The simulation parameters of the communication system are summarized in Table. I.

During the transmission the data stream is first grouped as bit vector, $b^{(i)}, i = 1, \dots, s$, which are converted to one-hot coding representation s. All the elements of the vector of size M are set as 0, with exception to those whose bit vector index are as binary representation; those are set as 1. The one-hot vector s selects the corresponding symbol, and the bit vector (b) is then mapped to the DNN-based trainable constellation points in the mapper. The DNN in the mapper includes two dense layers of 2^{n+1} units each but with rectified linear activation unit (ReLU) and linear activation functions, respectively. The 2^{n+1} output of the second layer corresponds to the real and imaginary of the 2^n constellation points in the IQ plane. The final layer is the normalization layer to maintain the normalization factor to 1. This ensures that the transmitted signal is optimized for the given channel and the receiver can extract the transmitted symbols with minimal error while maintaining the power constraint. In this manner, b is mapped to the complex baseband symbol and transmitted over AWGN channel. In the receiver with conventional demapping, each received symbol is processed to generate the log-likelihood ratios (LLR), which is fed to the Low-Density Parity Check 5G (LDPC5G) decoder based on belief propagation for decoding [8]. As for the NDs, the training loss is calculated at the receiver using the input bits and the LLR.

TABLE I. Simulation parameters of the communication system.

Parameters	Values
Modulation technique	M-QAM
Modulation order (M)	M ∈ {16, 64, 256, 1024}
Coderate	0.6
Codeword length	1500
Channel Model	AWGN
Demapping Method	LLR
Demapper	Baseline: APP, MAXLOG-MAP ND: AE-CONV, AE-RL

For demapping the M-QAM constellation, the baseline demappers computes LLRs (sometimes referred as hard-decisions) on bits for a tensor of received symbols with conventional dempping technique and gray labeling. For the experiment both Approximate Posteriori Probability (APP) and Maximum a Posteriori Probability Logarithmic (MAXLOG-MAP) based dempping are considered for evaluation. When using APP dempping technique, the i^{th} bit LLR is calculated as:

$$LLR(i) = \ln\left(\frac{\sum_{c\in C_{i,1}} \Pr(c|p)\, exp\left(-\frac{1}{N_0}|y-c|^2\right)}{\sum_{c\in C_{i,0}} \Pr(c|p)\, exp\left(-\frac{1}{N_0}|y-c|^2\right)}\right), \quad (1)$$

where N_0 is the noise spectral density of the channel, y is the received signal and $C_{i,1}$ and $C_{i,0}$ are the sets of constellation points for which i^{th} bit is equal to 1 and 0, respectively. The LLR vector used as a prior knowledge for K bits mapped to a specific constellation point is given by $p = [p_0, \dots, p_{K-1}]$, and $p = 0$ if there is no prior knowledge available. $\Pr(c|p)$ probability distribution of prior knowledge about the constellation symbol c is calculated as:

$$\Pr(c|p) = \prod_{K=0}^{K-1} sigmoid(p_k l(c)_k), \quad (2)$$

where $l(c)_k$ is k^{th} bit of the constellation symbol c, and 0 is replaced by -1 to map the constellation symbols to the IQ plane. For the MAXLOG-MAP dempping algorithm, the received signal is compared to all possible transmitted symbols and the symbol with the maximum a posteriori probability (MAP) is selected as the estimated symbol, which

is calculated as the logarithm of the product of the received signal and the corresponding transmitted symbol. Using the MAXLOG-MAP, the LLR of the i^{th} bit is calculated as:

$$LLR(i) = max_{c\in C_{i,0}}\left(\ln\left(\Pr(c|p)\right) - \frac{|y-c|^2}{N_0}\right) - max_{c\in C_{i,1}}\left(\ln\left(\Pr(c|p)\right) - \frac{|y-c|^2}{N_0}\right). \quad (3)$$

The LLR for both the algorithms has been defined in a manner that makes it equivalent to logits, different to many existing literatures where it is defined as $ln\left(\frac{\Pr(b_i=0|y)}{\Pr(b_i=1|y)}\right)$.

For demapping trainable constellation, the NDs calculates the LLR as logits. The ND takes in received samples $y \in C$ and processes them into individual layers for training. It consists of multiple dense layers each with ReLU activation function. The DNN can only work with real-valued data, so the inputs are fed to the network as 3-dimensional vectors:

$$[R(y), I(y), log_{10}(N_0)\,], \quad (4)$$

where, $R(y)$ and $I(y)$ are the real and imaginary inputs of y, respectively. The output of the NDs consists of logits that correspond to the LLR of the number of bits mapped on the constellation point for each dimension of the constellation design, therefore the last layer consists of neurons same as the number of bits per symbol n. For optimization of the end-to-end systems, Stochastic Gradient Descent (SGD) with backpropagation technique is employed [2]. It assumes a differentiable channel model and backpropagates the gradient through the channel to refine the system. Additionally, training algorithm from [10] without assuming differentiable channel is also adapted in the experiment, followed by fine tuning the receiver.

When utilizing conventional AE (AE-CONV) training on the trainable constellation, constellation geometry and bit-labeling are jointly trained on the transmitter side and the ND computes the logits on the coded transmitted bits at the receiver. The ND and trainable constellations are jointly trained using SGD and backpropagation. For the loss function Binary Cross Entropy (BCE) is used which maximizes the achievable information rate [2], [8]. The BCE is a commonly used loss function in end-to-end learning as it is closely linked to the achievable information rate in bit-interleaved coded modulation systems [11], [12]. The Reinforcement Learning AE (AE-RL) training uses the same end-to-end system; however, it stops the gradient after the existing channel to create a non-differentiable channel. Therefore, it jointly trains the transmitter and receiver over the non-differentiable channel where both transmitter and receiver losses are returned [10]. In essence, the transmitter acts as an agent that interacts with the environment (the channel and receiver) through a policy, receiving penalties or rewards that estimate the loss function gradient based on its actions and the current state. Therefore, the AE is trained with only observation and without any prior knowledge through policy learning [6], [10]. The constellation points are trained on the known perturbation from its output which estimates the transmitter gradient weights with respect to the loss function approximation, and the ND is trained using BCE which utilizes conventional backpropagation and SGD. For both ND the output logits of the transmitted bits correspond to the LLR that is directly used to calculate the BCE without any further computations.

III. Result and Discussion

The Bit Error Rate (BER) performance of constellation design corresponding to 16-QAM, 64-QAM, 256-QAM and 1024-QAM using the various demappers presented in section II is shown in Fig. 3. In conventional M-QAM schemes, higher order increases the number of bits per symbol, enabling higher data rates in digital communication systems. However, with higher order M-QAM, the error probability also increases, which leads to an increase in the overall BER [13] as shown in Fig. 3. However, it is also observed that the gain achieved by the NDs over baseline also improves when the number of bits per symbol increases.

From the analysis, the two baseline demappers using M-QAM constellation design achieved comparable performance throughout the experiment. APP exhibited slightly better BER compared to MAXLOG-MAP; however, the difference is insignificant. Considering that APP calculates the full posterior probability distribution of the transmitted bits, whereas MAXLOG-MAP only involves finding the maximum value of the LLR for each bit, APP generally requires more computation compared to MAXLOG-MAP. Therefore, there can be a tradeoff between BER and computation when selecting APP or MAXLOG-MAP dempping algorithm while using M-QAM constellation.

With respect to the DNs, constellation design corresponding to 16-QAM ($n = 4$) has no significant

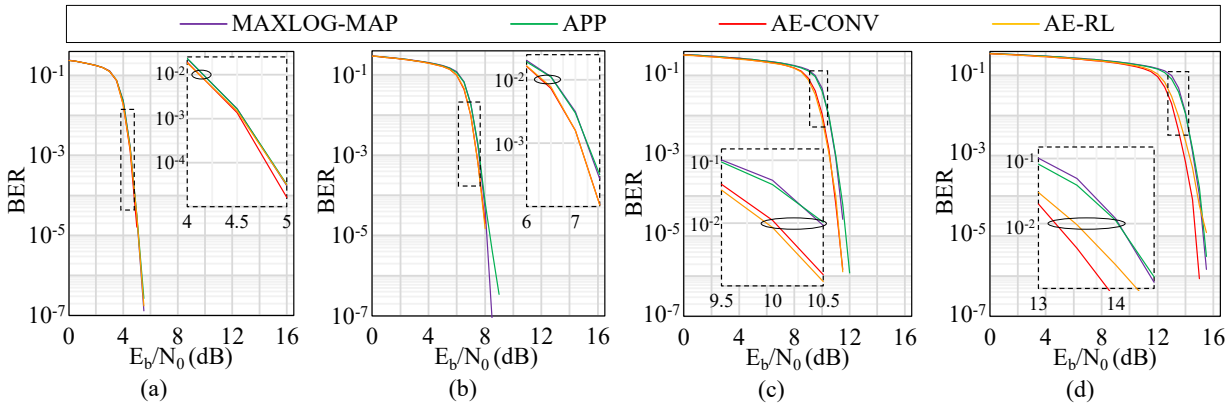

Fig. 3. BER analysis using different demappers corresponding to higher order M-QAM constellation design of (a)16-QAM ($n = 4$), (b) 64-QAM ($n = 6$), (c) 256-QAM ($n = 8$) and (d) 1024-QAM ($n = 10$).

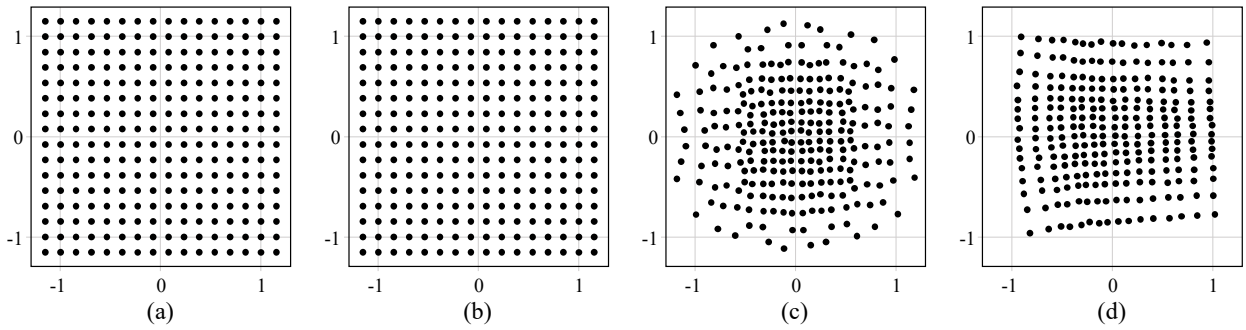

Fig. 4. Visualization of constellation design corresponding to 256-QAM ($n = 8$) for (a) MAXLOG-MAP, (b) APP, (c) AE-CONV and (d) AE-RL demapper.

difference between the learnable and non-learnable demappers, but rest of them show a significant improvement when using learnable demappers. The NDs showed a 0.5 dB gain over the baseline demappers for both trainable constellation design corresponding to 64-QAM ($n = 6$) and 256-QAM ($n = 8$) , the gain further improved by 0.05 dB with AE-RL for the latter. The highest improvement was achieved with trainable constellation design corresponding to 1024-QAM ($n = 10$), where the ND reached 0.8 dB with AE-CONV and 0.6 dB with AE-RL. Among the two DNs, AE-RL performed better with trainable constellation corresponding to 256-QAM and AE-CONV performed better with trainable constellation corresponding to 1024-QAM. The performance of AE-RL degrades with constellation design corresponding to higher number of bits per symbol, which is due to the instability and divergence issues in the feedback loop between the demapper and the channel model [7], [10]. AE-RL can also be affected by the vanishing gradient problem [3] for higher bits per symbol transmission.

The constellation design corresponding to 256-QAM ($n = 8$) using different demappers is shown in Fig. 4. AE-CONV demapper exhibits a cluster of constellation points that only differ by approximately one bit position in association to the neighboring constellation points. While this pattern may lead to a weaker reliability of the bit position concerning the least significant bit position, it improves the reliability of other bit positions and the decision region concerning the constellation points, thus optimizing the overall achievable information rate [8]. AE-RL attempts to maintain a regular shape while trying to achieve optimal performance, however the overall decision region shrinks, and with higher number of bits per symbol this may lead to higher BER compared to other learned and non-learned demapper. APP and MAXLOG-MAP both retain constellations with perfect square lattice.

IV. Conclusion and Future Work

In this study, we have analyzed the BER performance of NDs for trainable constellation. Specifically, we have considered two NDs and compared them with conventional demapper with M-QAM constellation as baseline. The main objective was to evaluate the performance of NDs on trainable constellation and conduct a comparative analysis with practical baseline. The analysis has shown that both the NDs outperformed the baseline for bits per symbol corresponding to higher order M-QAM. This indicates that, though overall BER increases with higher bit per symbol, NDs with trainable constellation can be utilized for communication services where high data-rate is indispensable. Furthermore, simpler MAXLOG-MAP demappers can be used when using conventional M-QAM modulation, as both baseline demappers achieved comparable results even though they differ in computational complexity.

For the future work, we are interested in exploiting different DL algorithms for the trainable constellation design and various NDs to further improve the performance of the communication system utilizing high bits per symbol.

Acknowledgement

This research is supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2018R1D1A1B07048338).

Reference

[1] I. Ahmad and S. Shin, “Channel model for end-to-end learning of comm. systems: A survey,” *ArXiv Prepr. ArXiv220403944*, 2022.

[2] T. O’shea and J. Hoydis, “An introduction to deep learning for the physical layer,” *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 4, pp. 563–575, 2017.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[4] T. J. O’Shea, T. Roy, and N. West, “Approximating the void: Learning stochastic channel models from observation with variational generative adversarial networks,” presented at the 2019 International Conference on Computing, Networking and Communications (ICNC), IEEE, 2019, pp. 681–686.

[5] H. Ye, G. Y. Li, B.-H. F. Juang, and K. Sivanesan, “Channel agnostic end-to-end learning based communication systems with conditional GAN,” presented at the 2018 IEEE Globecom Workshops (GC Wkshps), IEEE, 2018, pp. 1–5.

[6] F. A. Aoudia and J. Hoydis, “End-to-end learning of communications systems without a channel model,” presented at the 2018 52nd Asilomar Conference on Signals, Systems, and Computers, IEEE, 2018, pp. 298–303.

[7] M. Goutay, F. A. Aoudia, and J. Hoydis, “Deep reinforcement learning autoencoder with noisy feedback,” presented at the 2019 International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT), IEEE, 2019, pp. 1–6.

[8] S. Cammerer, F. A. Aoudia, S. Dörner, M. Stark, J. Hoydis, and S. Ten Brink, “Trainable communication systems: Concepts and prototype,” *IEEE Trans. Commun.*, vol. 68, no. 9, pp. 5489–5503, 2020.

[9] J. Hoydis *et al.*, “Sionna: An open-source library for next-generation physical layer research,” *ArXiv Prepr. ArXiv220311854*, 2022.

[10] F. A. Aoudia and J. Hoydis, “Model-free training of end-to-end communication systems,” *IEEE J. Sel. Areas Commun.*, vol. 37, no. 11, pp. 2503–2516, 2019.

[11] G. Böcherer, “Principles of coded modulation,” 2018.

[12] F. A. Aoudia and J. Hoydis, “End-to-end learning for OFDM: From neural receivers to pilotless communication,” *IEEE Trans. Wirel. Commun.*, vol. 21, no. 2, pp. 1049–1063, 2021.

[13] N. Islam and S. Shin, “Robust Deep Learning Models for OFDM-Based Image Communication Systems in Intelligent Transportation Systems (ITS) for Smart Cities.” *Electronics,* vol. 12, no. 11, p. 2425, May 2023, doi: 10.3390/electronics12112425.