Geometric Constellation Shaping Using Initialized Autoencoders

Amir Omidi, Ming Zeng, Jiachuan Lin†, Leslie A. Rusch
Centre for Optics, Photonics and Lasers (COPL), ECE Department, Laval University, Quebec, Canada
†Huawei Technologies Canada Co., Ltd., ON, Canada
amir.omidi.1@ulaval.ca, (ming.zeng, rusch)@gel.ulaval.ca, jiachuan.lin@huawei.com

***Abstract*—Geometric constellation shaping is a promising technique to boost the transmission capacity of communication systems. Earlier, traditional optimization methods in constellation design lead to several advanced quadrature amplitude modulation (QAM) formats, such as star QAM, cross QAM, and hexagonal QAM. The difficulty in determining decision boundaries limited their use in real systems. To overcome this, machine learning based geometric constellation shaping has recently been proposed, where the detection is done via neural networks. Unfortunately, the resulting constellation shape is often unstable and highly dependent on initialization. In this paper, we use an autoencoder for constellation shaping and detection, with strategic initialization. We contrast initialization with hexagonal QAM and square QAM. We present numerical results showing the hexagonal QAM initialization achieves the best symbol error rate performance, while the square QAM initialization has better bit error rate performance.**

I. Introduction

Constellation shaping is a technique to boost the transmission capacity of communication systems. Existing shaping methods can be broadly classified into two categories, namely geometric constellation shaping and probabilistic constellation shaping, each has its strengths and weaknesses [1], [2]. Considering its relative simplicity, the focus of this paper is on geometric constellation shaping.

Geometric constellation shaping optimizes the locations of the constellation points in the complex plane to achieve more compact constellations. The basic quadrature amplitude modulation (QAM) constellations are the well-known square QAM (SQAM) and rectangular QAM (RQAM), which are used for even and odd power of 2 constellations, respectively. With the help of geometric constellation shaping, various novel QAM constellations have been proposed, such as cross QAM (XQAM) [3], star QAM [4], hexagonal QAM (HQAM) [5]. More precisely, XQAM shifts the corner points in RQAM such that the peak and average energies of the constellation are reduced. Compared with RQAM, XQAM constellation provides at least 1 dB gain. Star QAM comprises of multiple concentric phase shift keying circles, each with an equal number of constellation points and equal phase angles. Star QAM is preferred over SQAM in systems limited by peak power. HQAM targets maximizing the minimum separation between two neighboring points, thus placing the constellation at the center of an equilateral hexagon. HQAM can either be regular or irregular (I-HQAM). Constellations of I-HQAM are set symmetric around the origin to achieve more compact packing [6], [7].

These various QAM constellations are the result of conventional optimization methods for geometric constellation shaping. Machine learning techniques for optimizing constellation shaping [8] have attracted much attention recently for this optimization. In this case, the detection can be performed straightforwardly using neural networks, without the need for calculation and application of complicated decision boundaries. Autoencoders in particular have shown great potential for constellation shaping. They exploit feedforward neural networks to encode binary symbols into in-phase and quadrature (I/Q) components at the transmitter and to decode I/Q components to binary symbols at the receiver [9].

Training of autoencoders is done as an end-to-end communication system. Data passes through two neural networks (an encoder and a decoder neural network) to reproduce the input binary stream at the output with minimum error. In each training step, the difference between transmitted data and the regenerated data is calculated and backpropagated to update the weights of both neural networks to reduce symbol error. The difference between input and output data is called training loss or training cost. The backpropagation algorithm uses a gradient descent optimization to update weights. autoencoders are explained in detail in section II. There are two types of autoencoder structures. The first aims to jointly optimize the constellation as well as the bit mapping; it has binary symbols at the input [10], [11]. However, since the bit mapping is inherently a discrete problem, and poorly suited to machine learning based optimization, the overall performance is unsatisfactory. The second one addresses only the constellation shaping and has one-hot data (symbols) as input. However, even in this case, shaping performance highly depends on initialization. This is most likely attributable to the constellation shaping problem being highly non-convex [12].

In this paper we use an unsupervised end-to-end learning communication system realized with autoencoders [9] to minimize the bit error rate (BER) and symbol error rate (SER) in an additive white Gaussian noise (AWGN) channel by geometric shaping of the constellation. We propose the use of Gray-

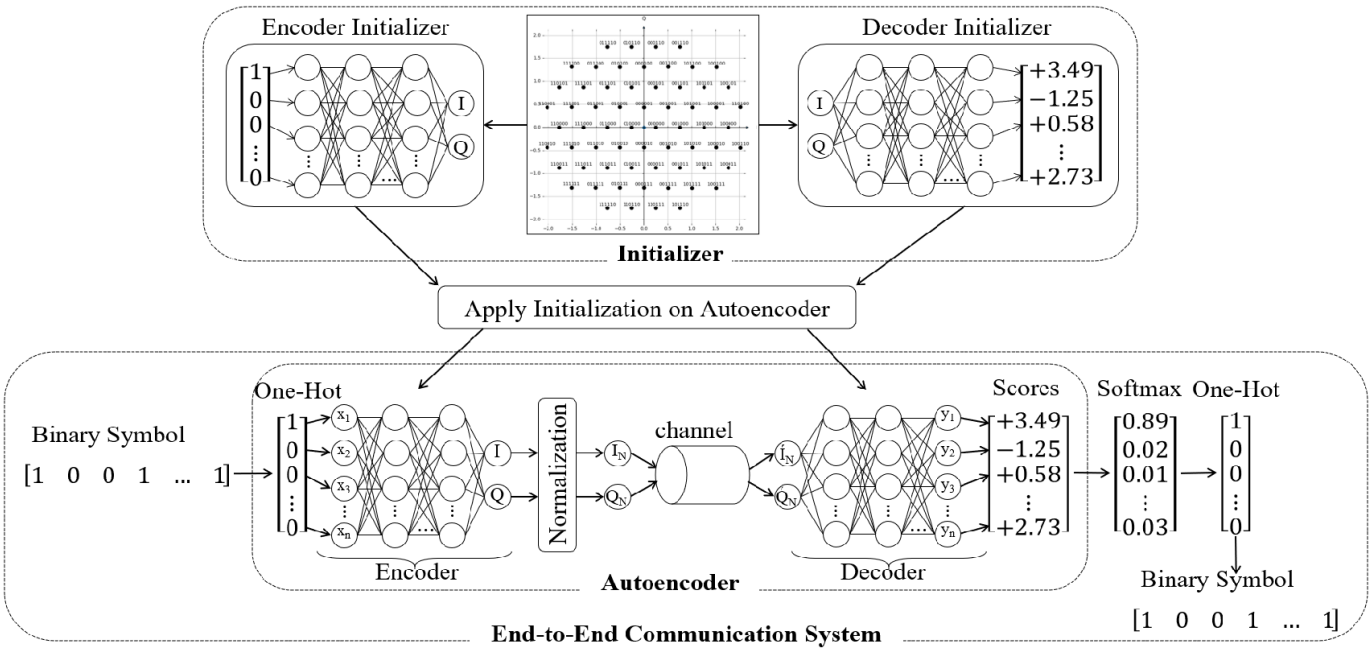

Fig. 1: end-to-end communication system realized with autoencoders

coded SQAM and I-HQAM for initialization. Our simulations show that I-HQAM-based initialization achieves the best SER performance in all considered solutions, while SQAM-based initialization has better BER performance when the signal noise ratio (SNR) is below 12 dB.

The rest of this paper is organized as follows. Section II describes the autoencoder structure we adopt, its hyperparameters, and the initialization step. Section III presents our simulation results. Finally, section IV concludes the paper.

II. AUTOENCODERS WITH INITIALIZATION

A. *autoencoders*

An autoencoder is an unsupervised machine learning method; it is a feedforward neural network in which the input is compressed into a lower dimension at an intermediate layer known as the code layer. The data is reconstructed in the output layer from this coded representation. A version of the autoencoder structure adapted to communications is presented in the *autoencoder* block of Fig. 1. The channel and power normalization are not standard to autoencoders, but rather introduced to simulate end-to-end communications. Mapping from inputs to code is accomplished with the encoder unit of the autoencoder, while de-mapping from code to outputs is accomplished with the decoder unit. This dimension reduction and reconstruction is similar to what happens in QAM modulation, where each input symbol is coded into in-phase and quadrature (I/Q) components and the reverse is done at the receiver by decoding received IQ components into binary symbols.

The proposed autoencoder structure in this paper uses one-hot encoded vectors as the input of the encoder. Some research have used binary symbols instead of one-hot encoded vectors. Inputting binary symbols to the autoencoder can jointly optimize bit mapping and constellation shaping. However, the overall performance is unsatisfactory. To show this, some constellations generated using binary inputs are presented in Fig. 2(a, b, c). The constellation points are highly asymmetric and the distances between some constellation points are very small. Note that the asymmetry in the constellations resulted from binary inputs can also be seen in the constellation presented in Fig. 2 of [11]. Since the goal of geometric constellation shaping is to maximize the minimum distance between any two constellation members for a given average transmitted power [13], the resulting performance is unsatisfactory. In contrast, Fig. 2(d) presents the results for the same autoencoder structure when one-hot encoded vectors are input. We see that the constellation points are quite symmetric and the minimum distance is much larger. Therefore we take binary input sequences and convert them into one-hot encoded vectors at the input of the encoder. One-hot encoded vectors at the output of the receiver are similarly converted to their binary sequence at the receiver. The one-hot form of a symbol is the vector

$$v_{e_i}[j] = \begin{cases} 1 & \text{if} \quad i = j \\ 0 & \text{otherwise} \end{cases} \qquad j = 0, ..., M-1 \qquad (1)$$

where e_i is the i^{th} symbol, $v_{ei}[j]$ is the j^{th} entry of the vector v related to the symbol e_i, and M is the number of symbols. The generated one-hot encoded vector is fed into the encoder to pass through its layers and generate IQ components.

We normalize the average power of the symbols to one to ensure the power efficiency of the resulting constellation. The normalized modulated signals presented with I_N and Q_N in Fig. 1 pass through the AWGN channel. At the receiver side,

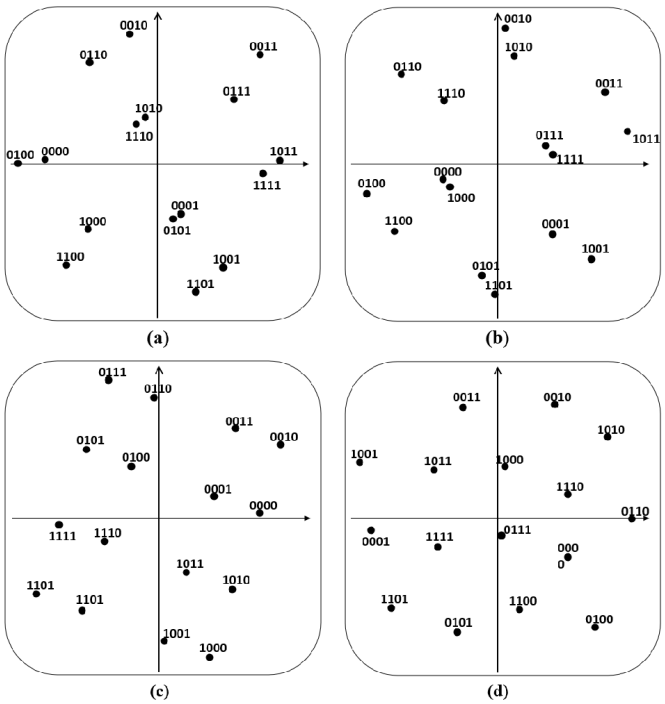

Fig. 2: Some 16 QAM constellations generated with autoencoders using (a, b, c) binary inputs and (d) one-hot encoded vector inputs.

noisy symbols (shown as I'_N and Q'_N in Fig. 1) will be used to reconstruct the original one-hot encoded vector with the help of the autoencoder decoder unit of . The output of the decoder is a vector of real numbers that are called scores. Examples of different vectors mentioned in this section are presented in Fig. 1. To extract the one-hot vectors from the scores, the following softmax function is applied.

$$S(r_i) = \frac{e^{r_i}}{\sum_{j=1}^{n} r_j}, \tag{2}$$

$$\sum_{i=1}^{n} S(r_i) = 1, \tag{3}$$

where r_i is the i^{th} entry of the scores vector and n is the length of the one-hot encoded vector. softmax converts the scores into probabilities which sum to 1; each entry in the softmax vector denotes the probability of being 1 for that entry.

The probability vector produced by the softmax function can now be used to regenerate one-hot encoded vector. We replace the maximum probability in the vector with the value 1 and other values with 0. Following this, the one-hot encoded vector can be converted to the binary symbol as the output of the end-to-end communication system. Note that, the last two steps are not part of the learning process. These steps are only used after the learning process is completed and we use the trained network for decoding the transmitted message. In the learning process, after generating the scores, training loss is calculated using these scores. Next, the gradient of the loss function is found and used to update the weights of the autoencoder with the backpropagation algorithm. The loss function employed in our simulation is the cross-entropy loss, which has the softmax function embedded, and the gradients are calculated with the RMSprop optimizer. The CrossEntropyLoss function in Pytorch (i.e., the deep learning framework used in our simulation) is defined as

$$l(\vec{x}, \vec{y}) = -\sum_{i} x_i \log S(y_i), \tag{4}$$

where, $\vec{x}$ is the transmitted one-hot encode vector and $\vec{y}$ is the vector of scores generated by the decoder. The autoencoder hyperparameters are presented in Table I.

TABLE I: Hyper-parameters of the autoencoder

Hyper-parameter	Value
Number of hidden layers	1
Number of neurons in hidden layer	64
Learning rate	0.001
Activation function	CELU
Optimizer	RMSprop
Batch size	256
Loss function	CrossEntropyLoss

B. Initialization

As noted in the introduction, constellation shaping is a highly non-convex problem [12], and thus, the system performance is strongly dependent on the initialization. Indeed, it has been shown that random initialization is quite poor constellations [11]. To address this, we propose to use Gray-coded SQAM (Fig. 4(a)) and Gray-coded I-HQAM (Fig. 4(c)) for initialization. After choosing the initialization constellation and prior to training the autoencoder in the presence of noise, we construct an feedforward neural network with the same layer structure (i.e., same hidden layers, same number of neurons in each layer and consequently the same weights) as the encoder initializer at the left side of the initializer block in Fig. 1. We train the encoder to produce the selected initialization constellation at its output (the constellation presented in the initializer block of Fig. 1).

We create another feedforward neural network with the same layer structure as the decoder initializer at the right side of the initializer block in Fig. 1. We train the decoder of the initializer to produce the appropriate one-hot encoded vector for the selected initialization constellation (the constellation presented in the initializer block of Fig. 1). As there is no noise in the initializer, the decoding is error free. The weights of these two separate feedforward neural network, are copied to the weights of the encoder and decoder of the autoencoder, respectively. The flowchart shown in Fig. 3 demonstrates the initialization and training processes.

After training, the corresponding optimized constellations related to the two initialization methods under E_b/N_0 = 15 dB are shown in Fig. 4, namely Fig. 4(b) for 64 I-HQAM and

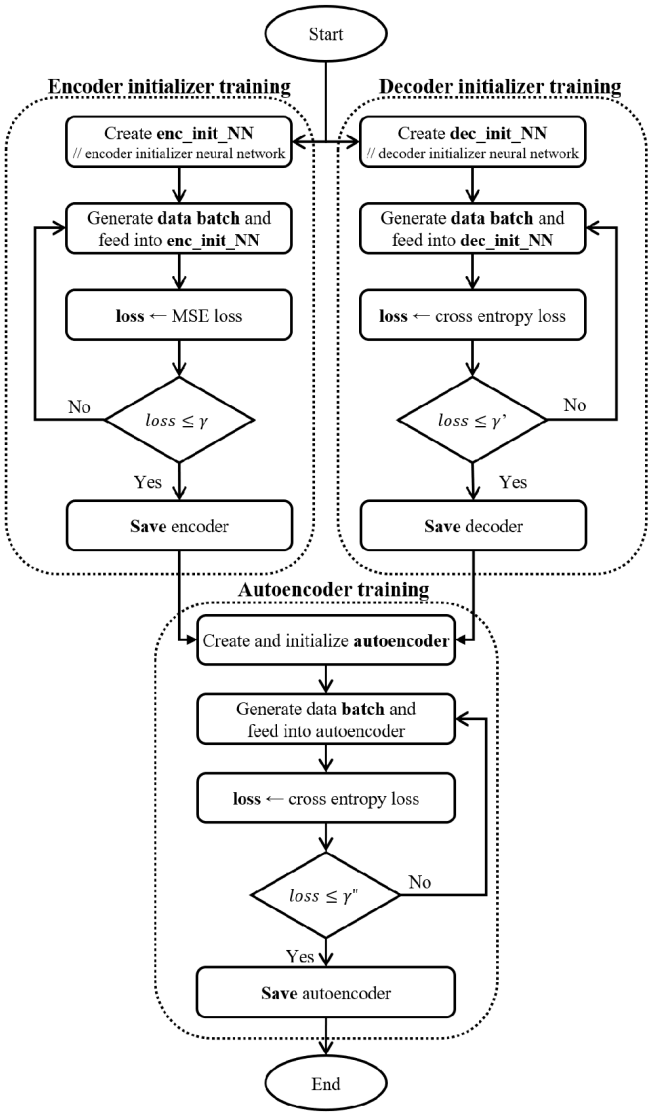

Fig. 3: autoencoder initialization and training process

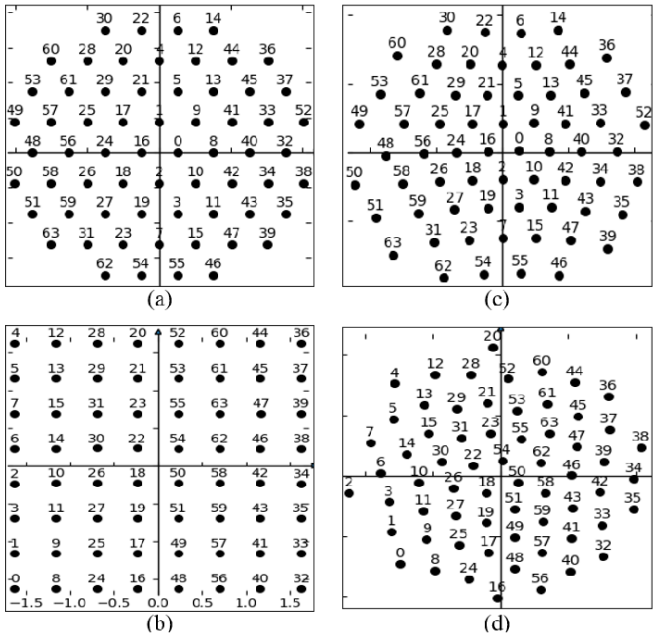

Fig. 4: Left column presents initializer constellations (I-HQAM (a), SQAM (b)) and right column presents autoencoder results for mentioned initializers respectively.

Fig. 4(d) for 64 SQAM. We observe that the overall structure for 64 I-HQAM is well preserved, while that for 64 SQAM changes drastically. This is not surprising as I-HQAM already has very compact packing, and thus, only minor improvement can be obtained. In contrast, the SQAM packing is quite loose; this loose geometry makes the Gray-coded bit mapping easy to produce. The looseness also leads to substantial improvement in constellation point locations, but may reduce bit mapping performance. Note that the shaping results under other E_b/N_0 values may differ slightly.

III. SIMULATION RESULTS AND DISCUSSION

In this section we evaluate the performance of the proposed initialized autoencoder. The simulation is performed by training the autoencoder to optimize 64 QAM over an AWGN channel with the initialization based on 64 SQAM and 64 I-HQAM. We use python for the programming language and Pytorch for the deep learning framework. The BER and SER are estimated using a Monte-Carlo simulation, counting at least 100 errors.

Figure 5(a) presents the SER vs. E_b/N_0 performance between the proposed initialized autoencoder and conventional SQAM and I-HQAM. For both SQAM and I-HQAM, the proposed initialized autoencoder can achieve better SER performance when compared with its counterpart. This shows the effectiveness of employing machine learning for constellation shaping. More precisely, a performance gain of 0.18 dB and 0.39 dB can be achieved for I-HQAM and SQAM, respectively. Such a difference in performance gain is consistent with constellation shapes in Fig. 4. Moreover, when compared with conventional SQAM, the constellation initialized I-HQAM can achieve a performance gain as large as 0.47 dB.

Figure 5(b) presents the BER vs. E_b/N_0 performance for the considered schemes. We see that the proposed scheme initialized with I-HQAM outperforms conventional I-HQAM for any given E_b/N_0 value, again validating employing machine learning for constellation shaping.

For SQAM there exists a trade-off between enhancement in packing and bit mapping. There is on average among nearest neighboring points a larger minimum Euclidean distance (for a given power), but a smaller Hamming distance. Accordingly, performance of the proposed scheme initialized with SQAM versus the conventional SQAM depends on the E_b/N_0. From Fig. 5(b), we see that for low E_b/N_0 values the deterioration in bit mapping is dominant, and thus, conventional SQAM obtains slightly better BER performance than its machine learning based counterpart. For higher E_b/N_0 values, however, the opposite is observed. Note that the same trade-off also leads

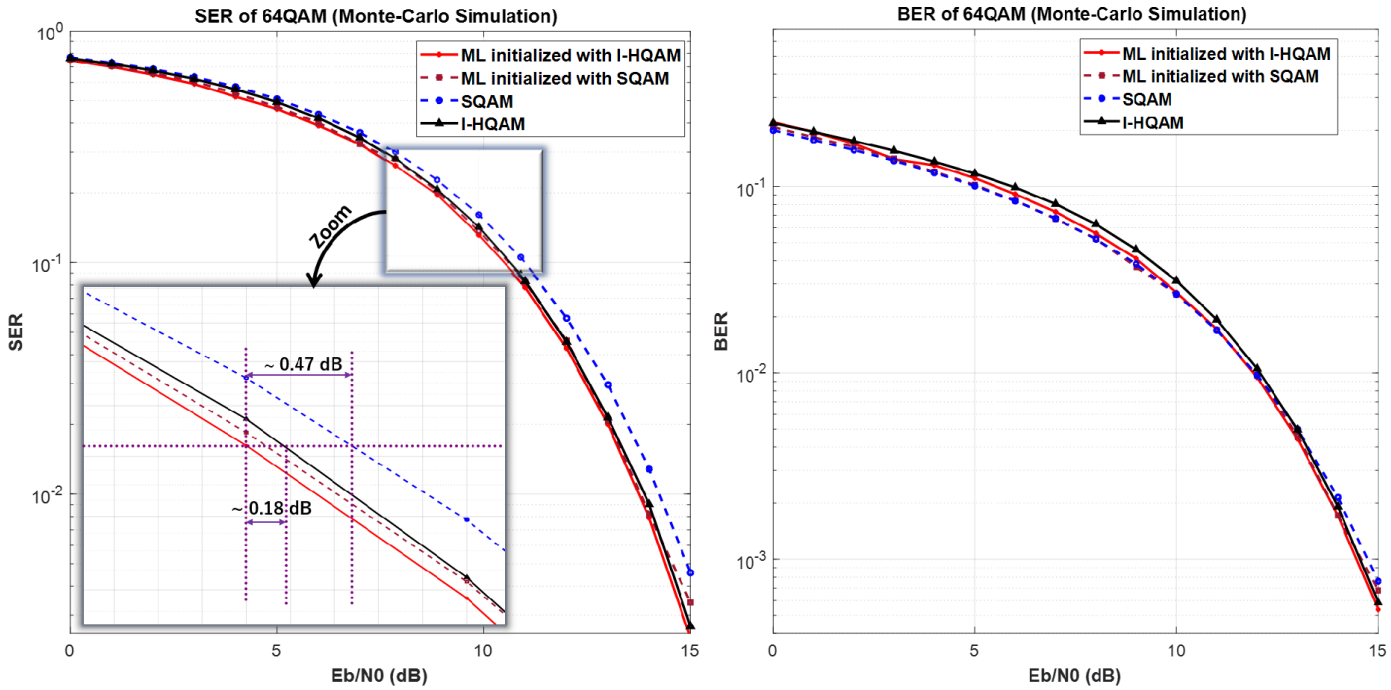

Fig. 5: SER and BER performance comparison between the proposed initialized autoencoder and conventional SQAM and I-HQAM when E_b/N_0 increases; a) SER versus E_b/N_0, and b) BER versus E_b/N_0.

to the crossing between results for the autoencoder initialized with I-HQAM and results for the autoencoder initialized with SQAM.

IV. Conclusion

We compared results for two initializations (SQAM and I-HQAM) of an end-to-end communication system autoencoder for constellation shaping. The performance of the proposed schemes was evaluated via simulation. We have shown that an autoencoder initialized with I-HQAM yields a constellation with the best SER performance; this leads to a 0.47 dB gain over conventional SQAM. This clearly demonstrates the potential of employing machine learning techniques for constellation shaping. We observed that a more advanced bit mapping scheme is required for the proposed schemes to maintain the SER gain for BER. This will be an interesting topic for our future research.

Acknowledgment

This research was supported by the Huawei/NSERC Industrial Research Chair, 537311-18 and by NSERC through its Discovery Grant.

References

[1] A. I. A. El-Rahman and J. C. Cartledge, "Multidimensional geometric shaping for QAM constellations," in *European Conference on Optical Communication (ECOC)*, Sep. 2017, pp. 1–3.

[2] J. Cho and P. J. Winzer, "Probabilistic constellation shaping for optical fiber communications," *Journal of Lightwave Technology*, vol. 37, no. 6, pp. 1590–1607, Mar. 2019.

[3] S. Colonnese, G. Panci, S. Rinauro, and G. Scarano, "High SNR performance analysis of a blind frequency offset estimator for cross QAM communication," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Mar. 2008, pp. 2825–2828.

[4] W. T. Webb, "QAM: the modulation scheme for future mobile radio communications?" *Electronics Communication Engineering Journal*, vol. 4, no. 4, pp. 167–176, Aug. 1992.

[5] P. K. Singya, P. Shaik, N. Kumar, V. Bhatia, and M.-S. Alouini, "A survey on design and performance of higher-order qam constellations," 2020.

[6] S. Park and M. Byeon, "Error performances of 64-ary triangular quadrature amplitude modulation in awgn channel," in *2007 IEEE 65th Vehicular Technology Conference - VTC2007-Spring*, April 2007, pp. 1752–1755.

[7] M. Abdelaziz and T. A. Gulliver, "Triangular constellations for adaptive modulation," *IEEE Transactions on Communications*, vol. 66, no. 2, pp. 756–766, Feb 2018.

[8] F. N. Khan, Q. Fan, J. Lu, G. Zhou, C. Lu, and A. P. Tao Lau, "Applications of machine learning in optical communications and networks," in *2020 Optical Fiber Communications Conference and Exhibition (OFC)*, March 2020, pp. 1–91.

[9] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, Dec 2017.

[10] M. Schaedler, S. Calabrò, F. Pittalà, G. Böcherer, M. Kuschnerov, C. Bluemm, and S. Pachnicke, "Neural network assisted geometric shaping for 800 Gbit/s and 1 Tbit/s optical transmission," in *2020 Optical Fiber Communications Conference and Exhibition (OFC)*, 2020, pp. 1–3.

[11] R. T. Jones, M. P. Yankov, and D. Zibar, "End-to-end learning for GMI optimized geometric constellation shape," in *45th European Conference on Optical Communication (ECOC 2019)*, Sep. 2019, pp. 1–4.

[12] K. Gümüş, A. Alvarado, B. Chen, C. Häger, and E. Agrell, "End-to-end learning of geometrical shaping maximizing generalized mutual information," in *Optical Fiber Communications Conference and Exhibition (OFC)*, March 2020, pp. 1–3.

[13] G. Foschini, R. Gitlin, and S. Weinstein, "Optimization of two-dimensional signal constellations in the presence of Gaussian noise," *IEEE Transactions on Communications*, vol. 22, no. 1, pp. 28–38, January 1974.