KAN-based Decoder for Message Detection in BMOCZ Communication Systems

Anthony Joseph Perre* and Jack Hyatt†
*Department of Electrical Engineering, University of South Carolina, Columbia, SC, USA
†Department of Computer Science, University of South Carolina, Columbia, SC, USA
e-mail: {aperre, jahyatt}@email.sc.edu

***Abstract*—This paper introduces novel deep learning-based decoders for binary modulation on conjugate-reciprocal zeros (BMOCZ) in communication systems. BMOCZ, a technique that optimizes spectral properties and enhances robustness, has traditionally employed direct zero-testing (DiZeT) decoders. To improve performance, we propose two innovative decoding schemes: a multi-layer perceptron (MLP)-based decoder and a Kolmogorov-Arnold network (KAN)-based decoder. Both approaches leverage advanced machine learning (ML) techniques, with MLPs offering powerful capabilities and KANs providing unique advantages based on the Kolmogorov-Arnold representation theorem. Extensive simulations compare these decoders under additive white Gaussian noise (AWGN) and flat-fading Rayleigh channels. The results demonstrate that both MLP and KAN-based decoders significantly outperform the DiZeT decoder in terms of block error rate (BLER). The MLP-based decoder achieves the best overall performance, while the KAN-based decoder provides competitive BLER and exhibits similar performance at higher E_b/N_0 levels. These findings validate the effectiveness of machine learning-based decoders in improving reliability and accuracy of BMOCZ communication schemes. This work introduces a significant step in the applications of deep learning (DL) in communication systems by highlighting the strengths of both MLP and KAN decoders; specifically, by demonstrating their superior performance over traditional methods, this research opens new pathways for using neural network architectures in modern and future wireless communication networks, including applications in 5G, 6G, and Internet-of-Things (IoT) technologies.**

***Index Terms*—BMOCZ, Decoder, Machine Learning, KAN**

I. Introduction

In the field of IoT, modern communication has many areas that constantly improve. As the number of interconnected devices grows exponentially, so does the demand for communication protocols that can efficiently handle high data throughput, minimize latency, and maintain robustness against interference. Achieving these goals requires innovative approaches to signal design, particularly those that can optimize spectral properties while adhering to stringent performance constraints.

One promising approach in this context involves leveraging the mathematical properties of polynomials used to encode and modulate signals. By carefully designing the zeros of these polynomials, it is possible to shape the spectral characteristics of the signal in ways that enhance its transmission properties. This technique is modulation on conjugate-reciprocal zeros (MOCZ), and leads to the coefficients of the polynomial being transmitted [1]. MOCZ has been studied and improved to be more efficient in [2]. An effective varient of MOCZ is BMOCZ, which combines binary signal encoding with the symmetry and stability offered by conjugate-reciprocal zero placement. BMOCZ as a variant has been studied in recent times. In [3], strategies for decoding are proposed to prevent unintended receivers from decoding themselves. In [4], optimal radii are shown to be different with different decoders, and the performance of decoders is compared.

Currently, in the DL field, MLPs serve as a foundational building block in many architectures. However, in the past several months, a novel DL structure, called KANs, has emerged as an alternative to MLPs [5]. The authors in [5] claim that KANs can outperform MLPs in terms of accuracy with fewer total parameters. Another study in [6] disputes many of the claims made in [5] regarding the advantages of KANs over MLPs; however, the authors of [6] show that KANs do outperform MLP in terms of symbolic formula representation under a fair comparison. Recently, KANs have seen extensive use in various domains such as physics [7] and time series prediction [8], particularly for their increased interpretability and symbolic representation capabilities.

In this paper, we propose using KAN-based decoders for BMOCZ. We compare the performance of different decoders in terms of BLER for a DiZeT decoder, MLP-based decoder, and a KAN-based decoder. We evaluate the performance of these models under both AWGN and flat-fading channels.

Organization: The paper is organized as follows. Section II presents the system model and provides fundamental concepts regarding KANs. Section III outlines the methodology and describes the proposed KAN-based decoder. Section IV shows the BLER performance and compares KANs to MLPs and DiZeT decoders. Section V concludes the paper.

Notation: The set of real numbers is represented by $\mathbb{R}$. The set of complex numbers is represented by $\mathbb{C}$. A symmetric complex normal distribution with zero mean and variance σ^2 is written as $\mathcal{CN}(0, \sigma^2)$.

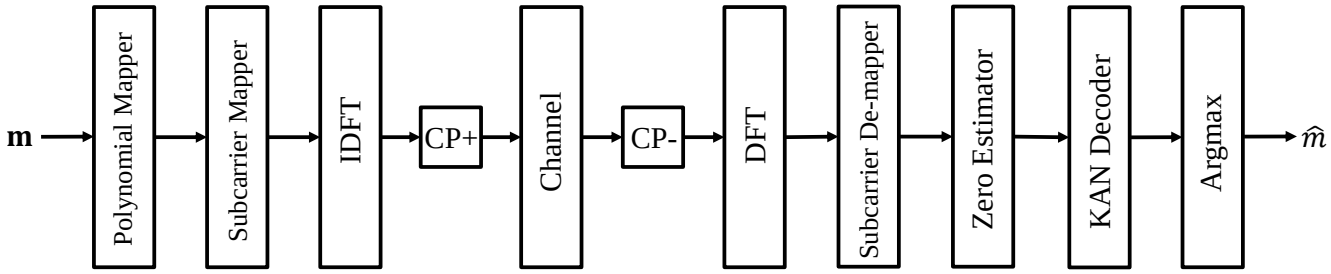

Fig. 1: Single-link OFDM communication scheme with KAN-based decoder.

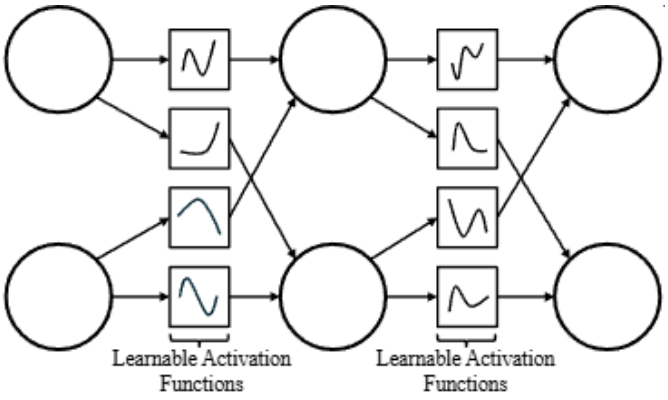

Fig. 2: Diagram of KAN-based network

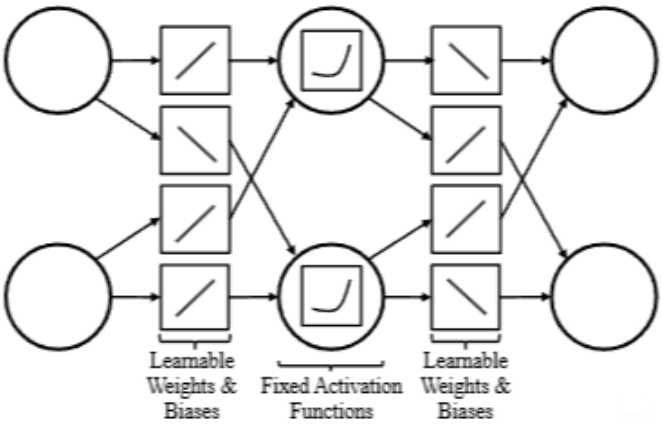

Fig. 3: Diagram of MLP-based network

II. Preliminaries and System Model

In this section, we discuss preliminaries on KAN and provide our system model for BMOCZ.

A. Kolmogorov-Arnold Networks

The structure of KANs is inspired by the Kolmogorov-Arnold representation theorem, which establishes that any multi-variate continuous function can be expressed as the sum of multiple uni-variate continuous functions [9], i.e.,

$$f(x_1, x_2, \ldots, x_d) = \sum_{i=0}^{2d} \Phi_i \left(\sum_{j=1}^{d} \phi_{i,j}(x_j) \right) \ , \tag{1}$$

where $\phi_{i,j} \in [0,1] \to \mathbb{R}$ and $\Phi_i : \mathbb{R} \to \mathbb{R}$. The authors of [5] generalize the inner and outer sums in (1) to accommodate an arbitrary number of layers L such that

$$\mathrm{KAN}(\mathbf{x}) = (\mathbf{\Phi}^{(L)} \circ \mathbf{\Phi}^{(L-1)} \circ \cdots \circ \mathbf{\Phi}^{(2)} \circ \mathbf{\Phi}^{(1)})(\mathbf{x}) \ , \tag{2}$$

where $\mathbf{\Phi}^{(l)}$, $\forall \ell \in \{1, \ldots, L\}$, contains learnable activation functions applied to edges. Here, an edge refers to a connection between an input and an output neuron and performs a transformation on the input. An illustration of a KAN model can be seen in figure 2, showcasing the learnable activation functions. For comparison, we provide a complementary illustration of a typical MLP model in figure 3, also showcasing the learnable weights and fixed activation functions.

It is worth noting that, in MLPs, the edges are linear learnable transformations and non-linear activation functions are fixed. In [5], the authors express $\phi(x)$ as a linear combination of B-splines and sigmoid linear unit (SiLU) activation function. The function $\phi(x)$ in KANs can then be formulated as

$$\phi(x) = w_{\mathrm{b}} \times \mathtt{SiLU}(x) + w_{\mathrm{s}} \times \sum_i \left[c_i \times B_i(x) \right] \ , \tag{3}$$

where $B_i(x)$ is the B-spline basis function consisting of piecewise polynomials of degree p and is scaled by a learnable weight c_i. The parameters w_{b} and w_{s} are also learnable. Each B-spline is defined on a specific grid interval, which is determined from observing the range of input samples.

B. System Model

Consider a single-user communication link where $\mathbf{m} = (m_0, m_1, \ldots, m_{K-1})$ is the binary message vector to be transmitted. For BMOCZ, we map the kth information bit to the zero of a polynomial α_k such that

$$\alpha_k = \begin{cases} r\, \mathrm{e}^{j2\pi \frac{k}{K}} & \text{if } m_k = 1 \\ \frac{1}{r}\, \mathrm{e}^{j2\pi \frac{k}{K}} & \text{otherwise} \end{cases} \ , \tag{4}$$

where r determines the radii of complex-plane circles on which the complex zeros are placed. In BMOCZ, for a given K, the value of r is given by

$$r(K) = \sqrt{1 + \sin(\pi/K)} \ . \tag{5}$$

The transmitted symbols $s_{\text{tx}} \in \mathbb{C}^K$ are the coefficients of the polynomial $p(x)$ defined by α_k, $\forall k \in \{0, \ldots, K-1\}$. After encoding at the transmitter, s_{tx} propagate through a communication channel, where they are distorted by the channel and zero-mean symmetric complex AWGN. Let $h \in \mathbb{C}$ denote the channel coefficient. The received symbols s_{rx} can then be expressed as

$$s_{\text{rx}} = h s_{\text{tx}} + w, \tag{6}$$

for $w \sim \mathcal{CN}(0, \sigma_n^2)$. Under an AWGN channel, we set $h = 1$. For a flat-fading Rayleigh channel, we instead let $h \sim \mathcal{CN}(0, 1/2)$. We consider an orthogonal frequency-division multiplexing (OFDM) communication scheme in this scenario; consequently, we can assume h is constant for a given $\mathbf{m}$. Since h is constant, the position of the received zeros $\hat{\boldsymbol{\alpha}} = (\hat{\alpha}_0, \hat{\alpha}_1, \ldots, \hat{\alpha}_{K-1})$ is not affected by the channel response, which makes channel estimation unnecessary. For the DiZeT decoder, we evaluate the received polynomial at each of the possible conjugate-reciprocal zero pairs and check to see which is closer to zero. In doing so, we can identify which zeros were transmitted, since the polynomial evaluated near each transmitted α_k should be approximately zero. Naturally, finding the transmitted α_k means that we can also identify the value of $\mathbf{m}$. In this work, we propose replacing the DiZeT decoder using a KAN-based decoder, as discussed next.

III. Proposed Methodology

In this section, we propose a KAN-based decoder for BMOCZ, as well as a ML basaed method to determine the zero-mapping procedure. We also discuss our training strategies and general methodology.

A. Encoding Scheme

In BMOCZ, $\mathbf{m}$ is mapped to the zeros of a polynomial using (7), where the optimal r for a given K is determined using (5). In this work, we utilize ML methods to determine the zero-mapping strategy instead. Consider r and a vector of phase values for each bit, i.e., $\boldsymbol{\phi} = (\phi_0, \phi_1, \ldots, \phi_{K-1})$. Here, the kth information bit is mapped to

$$\alpha_k = \begin{cases} r\, \mathrm{e}^{j\phi_k} & \text{if } m_k = 1 \\ \frac{1}{r}\, \mathrm{e}^{j\phi_k} & \text{otherwise} \end{cases} . \tag{7}$$

To determine the values of r and $\boldsymbol{\phi}$, we jointly optimize the encoder and decoder with ADAM. During training, the same loss function is used to adjust the encoder and decoder parameters, as discussed in III-B2.

B. KAN-based Decoding Scheme

Let $\hat{p}(x)$ be a polynomial whose coefficients $\mathbf{c}$ are defined by s_{rx}. To obtain the roots of $\hat{p}(x)$, we put it in monic form, i.e.,

$$\hat{p}(x) = c_0 + c_1 x + \cdots + c_{K-1} x^{K-1} + x^K . \tag{8}$$

TABLE I: MLP-based Decoder Architecture

Layer Type	**Output Shape**
Input Layer	$(2K,)$
Dense + ReLU	$(L_{\text{hidden}},)$
Dense + ReLU	$(L_{\text{hidden}},)$
Dense (Output Layer)	$(2^K,)$

Then, we define the Frobenius companion matrix of the monic polynomial as

$$C(\hat{p}) = \begin{bmatrix} 0 & 0 & \ldots & 0 & -c_0 \\ 1 & 0 & \ldots & 0 & -c_1 \\ 0 & 1 & \ldots & 0 & -c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & -c_{K-1} \end{bmatrix} , \tag{9}$$

where $\det\{xI - C\}$ are exactly the roots of $\hat{p}(x)$, i.e., the values of $\hat{\boldsymbol{\alpha}}$. Next, we map $\hat{\boldsymbol{\alpha}}$ to a separate real-valued vector $\mathbf{s}_{\text{d}} \in \mathbb{R}^{2K}$. Let $\delta(\mathbf{s}_{\text{d}})$ denote the decoder network that maps $\mathbf{s}_{\text{d}}$ to a logits vector $\hat{\mathbf{s}}_{\text{m}} \in \mathbb{R}^{2^K}$. We consider L_{dec} layers at the decoder, where d_l denotes the number of neurons in the lth layer. For an KAN-based neural network, we have

$$\mathbf{a}_{\text{dec}}^{(l)} = \boldsymbol{\Phi}_{\text{dec}}^{(l)}(\mathbf{a}_{\text{dec}}^{(l-1)}) \quad l = 1, \ldots, L_{\text{dec}}$$

$$\boldsymbol{\Phi}_{\text{dec}}^{(l)}(\mathbf{a}_{\text{dec}}^{(l-1)}) = \begin{bmatrix} \phi_{1,1}^{(l)}(a_1^{(l-1)}) & \ldots & \phi_{1,d_{l-1}}^{(l)}(a_{d_{l-1}}^{(l-1)}) \\ \vdots & \ddots & \vdots \\ \phi_{d_l,1}^{(l)}(a_1^{(l-1)}) & \ldots & \phi_{d_l,d_{l-1}}^{(l)}(a_{d_{l-1}}^{(l-1)}) \end{bmatrix} , \tag{10}$$

where $\boldsymbol{\Phi}_{\text{dec}}^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$, $\mathbf{a}_{\text{dec}}^{(0)} = \mathbf{s}_{\text{d}}$, and $\phi_{d_{l-1},d_l}^{(l)}(\cdot)$ is the activation function in the lth layer connecting the d_{l-1}th input neuron to the d_lth output neuron. In this work, we use (3) for learning the activation functions during training for KAN. For an MLP-based neural network, we instead have

$$\mathbf{a}_{\text{dec}}^{(l)} = \sigma_{\text{dec}}^{(l)}(\mathbf{W}_{\text{dec}}^{(l)} \mathbf{a}_{\text{dec}}^{(l-1)} + \mathbf{b}_{\text{dec}}^{(l)}) \quad l = 1, \ldots, L_{\text{dec}} , \tag{11}$$

where $\sigma_{\text{dec}}^{(l)}$ is the element-wise non-linear activation function, $\mathbf{W}_{\text{dec}}^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$, $\mathbf{b}_{\text{dec}}^{(l)} \in \mathbb{R}^{d_l}$, and $\mathbf{a}_{\text{dec}}^{(0)} = \mathbf{s}_{\text{d}}$. For both the MLP and KAN-based decoder, the index of the detected message $\hat{m}$ is computed by

$$\hat{m} = \underset{\hat{\mathbf{s}}_{\text{m}} \in \mathbb{R}^{2^k}}{\arg\max}\ \hat{\mathbf{s}}_{\text{m}} . \tag{12}$$

Here, $\hat{m}$ can be converted to its binary representation as needed; however, only the BLER will be computed in this study, so converting $\hat{m}$ to bits is unnecessary.

1) Model Architecture: A block diagram of an end-to-end OFDM BMOCZ communication scheme using a KAN-based decoder can be seen in Fig. 1. In this study, we consider a 2 layer KAN-based decoder. For experiments, this model is compared to the MLP-based decoder outlined in table I. The MLP-based decoder serves the same purpose as the KAN in

Algorithm 1 Training with noise scheduling

Input: m, B, β, σ^2_{max}, σ^2_{min}, n_{epochs}
Output: Optimized parameters θ_{d},r,$\boldsymbol{\phi}$
$\sigma_n^2 = \sigma^2_{\text{max}}$
for $n = 1$ ***to*** n_{epochs} **do**
  Sample $\{m^{(i)}\}_{i=1}^{B*2^K}$ from all possible $\mathbf{m}$
  $s_{\text{tx}} = \text{Encoder}(m)$
  $s_{\text{rx}} = s_{\text{tx}} + w$, $w \sim \mathcal{CN}(0, \sigma^2)$
  $\hat{\boldsymbol{\alpha}} = \text{zeroEstimator}(s_{\text{rx}})$
  $\mathbf{s}_{\text{d}} \leftarrow \hat{\boldsymbol{\alpha}}$
  $\hat{y} = \delta(\mathbf{s}_{\text{d}})$
  $\mathcal{L} \leftarrow (13)$
  Update θ_{d}, r, and $\boldsymbol{\phi}$ using $\mathcal{L}$ and β
  $\sigma_n^2 = \sigma^2_{\text{max}} - (n/n_{\text{epochs}}) \times (\sigma^2_{\text{max}} - \sigma^2_{\text{min}})$
end

Fig. 1, and replaces the decoder block.

2) Training: To optimize the BLER performance, we train the decoder by using noise-scheduling [10] along with a modified cross-entropy loss function

$$\mathcal{L} = -\sum_{i=1}^{2^k} T_i \log\left(\frac{\exp(l_i)}{\sum_{j=1}^{2^k}\exp(l_j)}\right), \tag{13}$$

where T_i is the true label for the ith class, and l_i and l_j are the logits for the ith and jth decoder output. The model directly outputs logits since it does not have a softmax layer. The Adam optimizer adjusts the decoder parameters θ_{d} to jointly train the decoder. Algorithm 1 outlines this process, where B is the batch size, σ^2_{max} and σ^2_{min} are the noise scheduling range, and β is the learning rate.

IV. Numerical Results

For numerical experiments, we consider decoders for $K = 4$ and $K = 6$ bits. In this study, we consider an MLP with $L_{\text{hidden}} = 150$ for the decoder. The KAN-based decoder replaces MLPs with a two KAN layers at the decoder, where the single hidden layer contains $L_{\text{hidden}} = 150$ neurons. Each activation function in the decoder contains 5 learnable control points c and third-degree polynomial basis functions. The KAN-based decoder and MLP-based decoder are created, tested, and trained in Python using the PyTorch machine learning library. For training each model, The ADAM optimizer is used with a learning rate $\beta = 5 \times 10^{-3}$. We note that $E_{\text{b}}/N_0 = 15.5$ dB is used to compute σ^2_{max} and $E_{\text{b}}/N_0 = 10.5$ dB is used to compute σ^2_{min}. We train using $B = 256$, which means there are 256×2^5 messages transmitted per epoch. For $K = 4$, we train for $n_{\text{epochs}} = 10^6$, while for $K = 6$, we train for $n_{\text{epochs}} = 5 \times 10^5$. All ML models are trained using a GeForce RTX 3070. We compare the performance of the MLP and KAN-based decoders to DiZeT for the same K values, where the gain in dB for different decoders is computed at BLER $= 10^{-3}$.

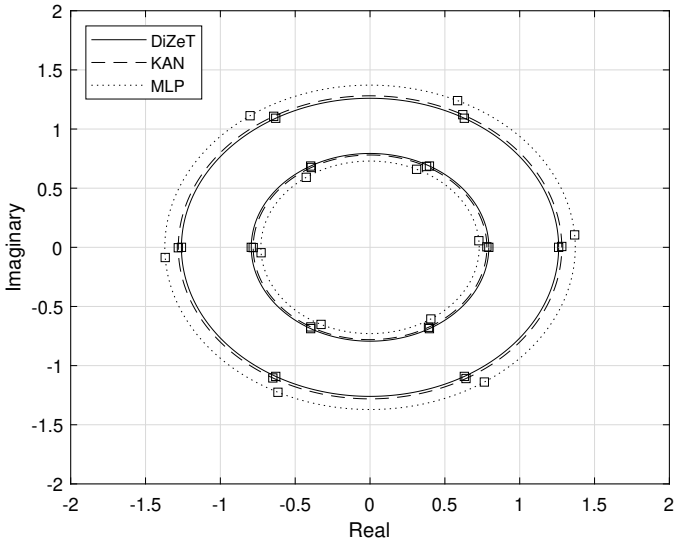

Fig. 4: Encoder zero-mapping for the various schemes

A. Encoder Zero-Mapping

As discussed in section III-A, we learned r and ϕ using the ADAM optimizer. To observe the zero-mapping for each encoder scheme, we plot the constellation seen in Fig. 4. From this plot, we can see that the two concentric circles are farther apart when r is optimized while training the MLP-based decoder as compared to KAN and DiZeT. Interestingly, we see that the value of r for both KAN and DiZeT is almost exactly the same. In all cases, the $\boldsymbol{\phi}$ learned is uniformly distributed on th interval $[0, 2\pi]$, which makes sense given that in this case the average distance between different zeros is maximized. We note that while using KANs, the value of $\boldsymbol{\phi}$ matches the DiZeT decoder almost exactly, which somewhat validates equation 5.

B. BLER in AWGN Channel

To access the BLER for each model, we perform a Monte-Carlo simulation where the E_{b}/N_0 is gradually increased from 0 dB to 14 dB. We run the simulation until 2,500 block errors have been detected for each tested E_{b}/N_0. Fig. 5 shows the BLER curves for the KAN-based decoder, MLP-based decoder, and DiZeT decoder in an AWGN channel. For $K = 4$ and $K = 6$, we see that the KAN and MLP-based decoders perform significantly better in terms of BLER as compared to the DiZeT decoder. For $K = 4$, the MLP-based decoder performs the best, although it appears to level off at higher E_{b}/N_0; specifically, we see that the KAN-based decoder catches up to the MLP. For $K = 5$, we see that the MLP-based decoder and KAN-based decoder more-or-less perform the same.

C. BLER in Flat-fading Rayleigh Channel

We again perform a Monte-Carlo simulation to access the BLER performance for each model; however, we now consider an E_{b}/N_0 range from 0 dB to 40 dB. Fig. 6 shows

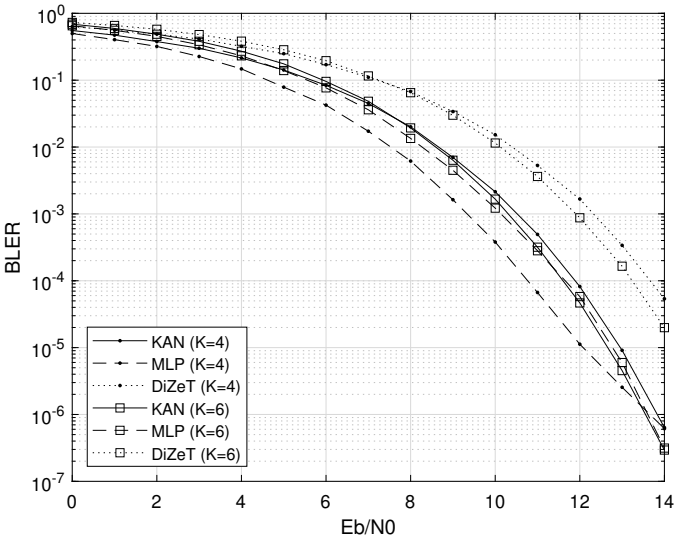

Fig. 5: BLER in AWGN channel for KAN-based decoder, MLP-based decoder, and DiZeT decoder.

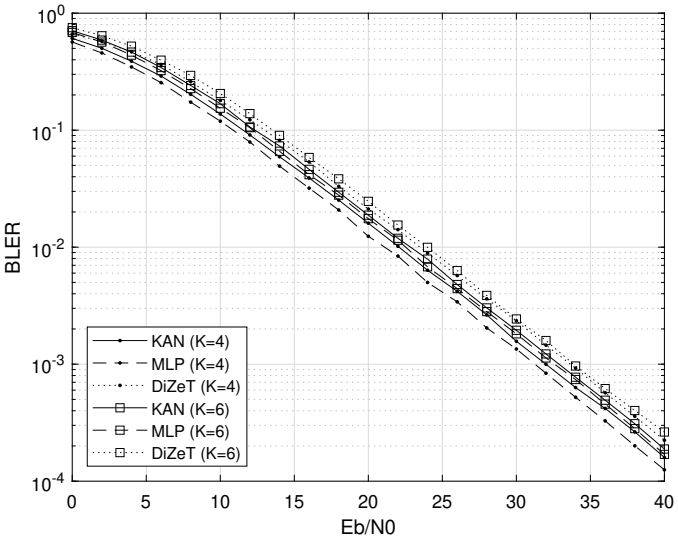

Fig. 6: BLER in flat-fading rayleigh channel for KAN-based decoder, MLP-based decoder, and DiZeT decoder.

the BLER curves for the KAN-based decoder, MLP-based decoder, and DiZeT decoder in a flat-fading rayleigh channel. For $K = 4$ and $K = 6$, we see that the KAN and MLP-based decoders perform better in terms of BLER as compared to the DiZeT decoder; however, we note that the difference is not as pronounced as in the AWGN channel. For $K = 4$, we again see that the MLP-based decoder performs the best, and it does not appear to level off as in the AWGN case. For $K = 5$, we see that the MLP-based decoder and KAN-based decoder perform similarly, which matches expectations set by the AWGN BLER curves in Fig. 5.

D. Discussion of Results

The BLER curves in Fig. 5 and Fig. 6 indicate that both the KAN-based decoder and MLP-based decoder perform better than DiZeT in various scenarios. This is significant as it indicates that ML methods can be used to improve BMOCZ communication systems by reducing error and improving reliability. It is important to note that the increase in performance in the flat-fading channel in figure 6 may not look like a significant increase, but increasing the performance by a dB of 2 or 3 is worthy of highlighting.

For a direct comparison between the ML methods, the figures reveal that the MLP-based decoder consistently achieves better performance than the KAN-based decoder. This may suggest that the architecture and learning capacity of the MLP enable it to model and correct errors more effectively than the KAN-based approach. However, further research exploring a wider variety of internal structures in DL models is necessary. The difference in the number of hidden layers specifically between models is a potential area of interest to study, as the KAN used has 1 hidden layer compared to the 2 hidden layers in the MLP, which gives less parameters for the KAN to work with.

In all, these findings highlight the potential for future research to refine these methods further and explore hybrid or ensemble approaches to maximize decoding efficiency and reliability.

V. Concluding Remarks

In this work, we propose and investigate the usage of MLP and KAN based decoders in OFDM communication schemes. Through simulations, we demonstrate that both ML-based decoders outperform traditional DiZeT decoders in both AWGN channels and flat-fading channels, offering significant reductions in BLER and improving overall system reliability. It is also noteworthy that the MLP-based decoder consistently achieves superior performance compared to the KAN-based decoder. The code and numerical results can be found in the GitHub repository here: BMOCZ-with-KAN-Decoder.

These findings open the door for more research into exploring other DL architectures for OFDM communication schemes, as well as internal structure differences within the same DL model, such as the number of hidden layers. Additionally, extending this framework to other communication paradigms or non-linear channel models could further validate the generalizability and scalability of the proposed decoders. Future areas of research may also include power efficiency between models running OFDM communication schemes, as efficiency is extremely important in relation to IoT.

As communication networks evolve to meet the demands of increasingly complex wireless environments, these findings suggest that deep learning techniques could complement traditional methods in developing more robust and adaptive systems. This potential is particularly relevant for next-generation wireless technologies, such as 5G and 6G networks, where reliability and adaptability are critical for applications ranging from IoT devices to low-latency communications.

REFERENCES

[1] P. Walk, P. Jung, and B. Hassibi, "Short-message communication and fir system identification using huffman sequences," in *2017 IEEE International Symposium on Information Theory (ISIT)*. IEEE Press, 2017, p. 968972. [Online]. Available: https://doi.org/10.1109/ISIT.2017.8006672

[2] A. A. Siddiqui, E. Bedeer, H. H. Nguyen, and R. Barton, "Spectrally-efficient modulation on conjugate-reciprocal zeros (se-mocz) for non-coherent short packet communications," *IEEE Transactions on Wireless Communications*, vol. 23, no. 3, pp. 2226–2240, 2024.

[3] M. Rajiv and U. Mitra, "Securing bmocz signaling: A two layer artificial noise injection scheme," in *2022 IEEE 23rd International Workshop on Signal Processing Advances in Wireless Communication (SPAWC)*, 2022, pp. 1–5.

[4] P. Huggins and A. Sahin, "On the optimal radius and subcarrier mapping for binary modulation on conjugate-reciprocal zeros," 2024. [Online]. Available: https://arxiv.org/abs/2408.10029

[5] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark, "KAN: Kolmogorov-Arnold networks," *arXiv preprint arXiv:2404.19756*, 2024.

[6] R. Yu, W. Yu, and X. Wang, "KAN or MLP: A fairer comparison," *arXiv preprint arXiv:2407.16674*, 2024.

[7] Z. Liu, P. Ma, Y. Wang, W. Matusik, and M. Tegmark, "KAN 2.0: Kolmogorov-Arnold networks meet science," 2024. [Online]. Available: https://arxiv.org/abs/2408.10205

[8] C. J. Vaca-Rubio, L. Blanco, R. Pereira, and M. Caus, "Kolmogorov-Arnold networks (KANs) for time series analysis," *arXiv preprint arXiv:2405.08790*, 2024.

[9] A. K. Kolmogorov, "On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition," *Doklady Akademii Nauk SSSR*, vol. 114, pp. 369–373, 1957.

[10] K. J. Geras and C. Sutton, "Scheduled denoising autoencoders," *arXiv preprint arXiv:1406.3269*, 2014.