Received 7 November 2024; revised 31 January 2025; accepted 24 February 2025. Date of publication 3 March 2025; date of current version 17 March 2025.

Digital Object Identifier 10.1109/OJCOMS.2025.3547261

Joint Demapping of QAM and APSK Constellations Using Machine Learning

ARWIN GANSEKOELE[1,2,3], ALEXIOS BALATSOUKAS-STIMMING[4] (Member, IEEE), TOM BRUSSE[5], MARK HOOGENDOORN[3], SANDJAI BHULAI[2], AND ROB VAN DER MEI[1,2]

[1]Department of Stochastics, Centrum Wiskunde & Informatica, 1090 GB Amsterdam, The Netherlands
[2]Department of Mathematics, Vrije Universiteit Amsterdam, 1081 HV Amsterdam, The Netherlands
[3]Department of Computer Science, Vrije Universiteit Amsterdam, 1081 HV Amsterdam, The Netherlands
[4]Department of Electrical Engineering, Eindhoven University of Technology, 5612 AZ Eindhoven, The Netherlands
[5]Ministry of Defense, 2500 ES The Hague, The Netherlands

CORRESPONDING AUTHOR: A. GANSEKOELE (e-mail: arwin.gansekoele@cwi.nl)

ABSTRACT As telecommunication systems evolve to meet increasing demands, integrating deep neural networks (DNNs) has shown promise in enhancing performance. However, the trade-off between accuracy and flexibility remains challenging when replacing traditional receivers with DNNs. This paper introduces a novel probabilistic framework that allows a single DNN demapper to demap multiple QAM and APSK constellations simultaneously. It is demonstrated that the framework can exploit hierarchical relationships in families of constellations. The consequence is that we need fewer neural network outputs to encode the same function without an increase in Bit Error Rate (BER). The simulation results confirm that the framework approaches the optimal demodulation error bound under an Additive White Gaussian Noise (AWGN) channel for multiple constellations. Under 3GPP-compliant OFDM fading channels, it is as accurate as a neural receiver operating on just one modulation type. Thereby, the framework addresses multiple important issues in practical neural receiver design. These include improvements in computational efficiency, a reduction in memory overhead, and an improved adaptability in dynamic environments.

INDEX TERMS Communication systems, machine learning, symbol demapping.

I. INTRODUCTION

DIGITAL communications has proven to be an essential technology in the era of information. A common communications pipeline consists of many components that need to match on both the sender and receiver sides. These include components such as filters, coding, and modulators. Recently, the use of machine learning (ML) has been investigated and has already been included in the 5G standard. DNN approaches, specifically, replace (parts of) the communications pipeline with a neural network. One approach is to replace the entire sender-receiver pipeline with a DNN [1], [2], [3], [4], [5], [6], [7], [8], [9]. Doing so allows the DNN to adapt the type of constellation used based on the channel conditions seen during training. However, since the entire pipeline is modeled by one DNN, the modularity of the pipeline is destroyed. Modularity can be hugely beneficial when designing practical systems as it simplifies the reuse of components in a setting where space is limited. That is why a second line of work opts to perform only channel estimation and equalization using DNNs [10], [11], [12], [13], [13], [14], [15], [16], [17], [18], [19], [20], [21]. As channel estimation and equalization are traditionally performed with data-driven techniques, replacing these components with a DNN is intuitive. However, assumptions on the output distribution of the DNN have to be made in this case. That is why a third line of work proposes to also add the demodulation component to the DNN, along with equalization and estimation [22], [23], [24], [25], [26], [27], [28].

Digital demodulation techniques generally assume that the received I/Q samples are perturbed by white Gaussian noise. This assumption becomes unnecessary by including the demodulator as part of the receiver design. This inclusion can potentially improve performance. The work of [22] demonstrated that this approach can reduce the number of pilots necessary, while achieving a performance similar to

that when the state channel information is known. However, this approach introduced a new problem. Traditional demappers for digital communications are flexible in the settings they can use. For example, if needed, the constellation type or bit mapping can be adjusted for every individual data block with traditional demappers. Flexibility in bit mappings allows receivers to operate with senders from previous generations, while supporting multiple constellation types is imperative for adaptive bandwidths.

Without further adjustments, it becomes necessary to train a new DNN receiver for every mapping and constellation type. If the implementation choice is made to keep all sets of parameters in memory, the memory usage scales linearly with the number of mappings and constellation types. On the other hand, loading parameter sets as required results in significant computational delays. Both options are impractical and unnecessary, as the main receiver task of channel estimation is shared between constellations. What makes these inefficiencies even more problematic is that naively applying DNNs already does not meet the low-power and high-throughput requirements of modern communication systems. While [22] propose a scheme to model all variants of the 4^n-QAM, their scheme does not support different bit mappings. It also achieves worse performance after combining multiple types of different 4^n-QAM constellations.

To address these issues, this work proposes a framework for building deep neural receivers that can demap multiple types of constellation. The modularity of the approach makes it applicable to any neural receiver that operates on some form of frequency and/or amplitude keying. The framework enables features present in traditional receivers, such as flexible symbol-to-bit mappings and multiple demodulation types. It does so while maintaining the gain enabled by neural receivers. The core contributions of the work as follows can be characterized as follows.

1) An extension to existing deep receivers is proposed that allows them to have flexible symbol-to-bit mappings without loss of performance.
2) The hierarchical relationship from [22] is generalized to arbitrary constellations. In addition, a hierarchical approach for APSK is proposed that can reduce the memory requirement to model families of constellations.
3) To the best of our knowledge, this work is the first to construct a deep neural receiver that can demap different constellations simultaneously with (close to) no loss in performance. This claim is empirically validated.

This paper is an extended version of [29], which was originally presented at the ICMLCN 2024 conference and was included in the proceedings. The current version extends the conference version as follows. First, a proof is included for a claim in the original work. A proof was included that the framework is strictly more general than a neural receiver that directly predicts bit LLRs. Second, a scaling factor was proposed to correct for an issue that occurs when demapping multiple different 4^n-QAM over fading channels. Third, a link with automatic modulation recognition (AMR) is presented. The extended version demonstrates how the framework can improve the performance of existing AMR systems. Finally, the evaluation suite is expanded to include 3GPP-compliant OFDM receivers.

II. RELATED WORKS

The idea of using DNNs to perform channel estimation and demapping has been studied over the years [30], [31]. [32], [33], [34] provide a vision on the role that AI could play in 6G. Reference [33] specifically identified three steps: AI replacing individual components in the sender-receiver pipeline, AI replacing multiple components at once, and finally, AI replacing the entire pipeline and/or designing the pipeline itself. The core benefit of AI they envision is the ability to design systems in a data-driven manner. A lot of development time goes into building systems that are modular and effective for different cases. The use of artificial intelligence could alleviate these issues by automating (parts of) pipeline design. This paper contributes to their vision by introducing modularity into a pipeline that is entirely data-driven.

Many preliminary works followed the first approach laid out in this vision. In the work of [10], the authors built a neural network demodulator for BPSK, BASK, and BFSK and demonstrated its potential compared to classical approaches. In similar work [11], they built a BPSK demodulator using a 1D-CNN. Furthermore, in the work by [12], they used a CNN to demodulate FSK under AWGN and Rayleigh fading. These works were still fairly simple and only addressed relatively simple constellations. In work by [14], the authors proposed DetNet for detection in OFDM systems. This is one of the first works to apply a DNN to OFDM system detection. The work of [15] was among the first to perform over-the-air demodulation. They used DBN-SVM and AdaBoost-based models to do so. Reference [16] introduced the use of a CNN + RNN-based network for demodulation. The work of [13], [17] was among the first to perform channel estimation and equalization using a DNN. They used a super-resolution-based DNN to do so. The work of [18] proposed following up the DNN denoiser by conventional LS for improved demodulation. The work of [20] investigated the use of graph neural networks for channel decoding. Work by [21] built on top of traditional LMMSE systems using a 1D-CNN to predict second-order statistics and the channel aging effect. Reference [35] used approximate message passing with graph neural networks for MIMO detection. All of these works introduced various components of the signal demodulation pipeline, such as improved modeling in increasingly complex and realistic channel models. They provide essential steps towards the development of fully end-to-end neural receiver systems. The development of the

framework relied on these works and/or the framework can be directly applied to these works. For example, DetNet can be extended with this framework to incorporate the properties discussed in this work. Furthermore, work by [36] proposed a dual-path network that takes into account the characteristics of the channel to improve classification. Their approach is similar to the effect of this work on automatic modulation recognition.

The work of [17], [37] was among the first to approach both channel estimation and demodulation with a neural OFDM receiver pipeline to improve performance. These lines of work share similarities with the detection networks described before. The work of [22] proposed DeepRX, which is a deep receiver that is able to approach the correction performance of correcting with the ground truth channel state information in SIMO OFDM systems. They achieved this by incorporating knowledge of modern neural network design to improve receiver capacity. The work of [25] extended DeepRX to a MIMO setting and the work of [38] investigated training parts of the sender to learn beamforming jointly with the channel estimation. Further work on MIMO [24] proposed the use of a GNN to extend works such as DeepRX to a MIMO setting with arbitrary sets of users. Further improvements by [27] proposed data augmentation strategies to improve the training of deep receivers. Other approaches include [28] who investigated power-efficient deep OFDM receivers, work by [23] that built a neural receiver on time-domain OFDM samples, and work by [26]. who used a metalearning approach to quickly adapt a classifier to new channels. The presented framework builds on these works specifically by making these deep receivers more flexible. Our approach can be added as a module to all of these works. Doing so adds to the properties that are discussed in this work.

Many authors have also already begun investigating the possibility of replacing the entire communication pipeline with learnable components. The preliminary work by [1] first proposed envisioning the sender-receiver pipeline as an autoencoder (AE). By doing so, they showed it possible to perform geometric constellation shaping as a side-effect of optimizing channel capacity directly. The work of [2] validated this approach in an over-the-air setting using two software-defined radios (SDR). They achieved BER scores close to conventional systems and showed that the constellations were realistic. The work of [4] extended these systems to modulation in the frequency domain in the form of OFDM. The work of [6] adds to this by jointly learning optimal pilot patterns and a channel estimator for MIMO. Although all of these works assume a differentiable channel model, the work by [3] proposed a method that allows model-free end-to-end learning that also works with non-differentiable channel models. The work of [7] built on this by demonstrating that end-to-end learning of the sender and receiver allows for pilotless communication without loss of performance. Reference [8] arrived at similar conclusions in their work. In a similar work, [9] learned a transmit and receive filter, constellation geometry, and associated end-to-end bit labeling. They achieved rates similar to those of an OFDM system under 3GPP-compliant channel models. An important step that still needs to be taken is the inclusion of different modes in these end-to-end pipelines. Adding the module presented in this work to these pipelines could enable constellation switching in a manner that respects predefined hierarchical constraints.

III. METHODOLOGY

The symbol demapping or demodulation process is the process of extracting information from some information-carrying signal. Given a sequence of received IQ samples $\mathbf{y}$, extraction of the underlying set of information bits $\mathbf{b}$ is desired. This is done by generating a set of bit estimates $\hat{\mathbf{b}}$ based on $\mathbf{y}$. These estimates can be hard or soft; in the latter case, a log-likelihood ratio (LLR) is computed for each bit that a decoder can use to refine the decision. This process could be performed with a DNN f with a set of parameters θ, in which case the LLR can be defined as

$$\log \frac{P(b_{ij}=1 \mid \mathbf{y},\theta)}{P(b_{ij}=0 \mid \mathbf{y},\theta)} = \left(f_\theta^{(LLR)}(\mathbf{y})\right)_{ij}. \tag{1}$$

The probability $P(b_{ij}=1 \mid \mathbf{y},\theta)$ refers to the probability that bit ij is a 1. The indices ij refer to the i-th IQ sample in the sequence and the j-th bit in the bit string. This DNN predicts the log-likelihood ratio of all bits in the information sequence. This formulation can be optimized by using a sigmoid function followed by a binary cross-entropy loss function. It can then be trained by applying some form of gradient descent on this loss quantity. Note that minimal assumptions have been made on the underlying data distribution. This allows for the finding of an optimal strategy for channel estimation, equalization, and demodulation.

A core issue with this approach is the lack of flexibility. All complexity is abstracted within the architecture and parameters defined by the set of parameters θ. As a DNN is a highly non-linear function, there is no straightforward way to change, for example, the bit mapping or the constellation used in demodulation.

A. MAPPING INDEPENDENCE

Mapping independence is proposed as the ability to change the association between constellation points and the bit assignment. This concept used to be trivially true in traditional receiver design. Previously, demapping was simply a module in a fully modular pipeline. When directly predicting the bit LLRs with a DNN, it becomes less obvious how to change the constellation and association between constellation points. The constellation and association of points with the bit string is hardcoded in the neural network weights. Other than a few special cases, it is necessary to build a new DNN receiver if a different association between constellation points and bit string is desired. Hence why this work defines (bit) mapping independence as a desirable property for practical DNN receivers.

When demodulation is performed without a DNN, the log-likelihood ratio of each bit can be computed exactly based on the constellation points. In this procedure, half of the constellation points are associated with a bit value of 1 and the other half with a bit value of 0 for each bit in the information string. The LLR can then be computed on the basis of the distances of the received sample to each constellation point and their bit assignment as

$$\log \frac{P(b_{ij}=1 \mid \mathbf{y})}{P(b_{ij}=0 \mid \mathbf{y})} = \log\left(\frac{\sum_{\substack{k=1 \\ (\mathcal{M}_b(\mathbf{s}_k))_j=1}}^{|S|} \exp\left(-\frac{1}{N_o}|y_i - s|^2\right)}{\sum_{\substack{k=1 \\ (\mathcal{M}_b(\mathbf{s}_k))_j=0}}^{|S|} \exp\left(-\frac{1}{N_o}|y_i - s_k|^2\right)}\right). \quad (2)$$

Here, c_{y_i} is the constellation point that corresponds to the i-th I/Q sample of the sequence $\mathbf{y}$. Variable s_k is defined as the k-th constellation point that belongs to a set S of possible constellation points given a certain constellation. In the case of 16-QAM, the set S has 16 constellation points, for example. The mapping $\mathcal{M}_b$ was also defined as a function that maps a symbol to a bit string. The outcome of this function can be indexed, which is done in the form of $(\int_{\|})_j = 1$. This means that the j-th bit of the bit string to which symbol s_k is mapped should be 1.

This form of demapping is optimal when it is assumed that the I/Q samples have a Gaussian distribution centered at their ground-truth constellation point. This approach also has a bit mapping that can be easily adjusted within the module. When a different bit assignment is needed under this formulation, only one thing needs to be changed. This is the assignment of which constellation points should be 1 and which 0. To achieve complete mapping independence, it is necessary to have an explicit probability measure over all possible constellation points. To achieve this in a DNN-based approach, the constellation point probabilities can be directly modeled as

$$P(c_{y_i} = s_k \mid \mathbf{y}, \theta) = \left(f_\theta^{(\text{symbol})}(\mathbf{y})\right)_{ik}. \quad (3)$$

Given the probability distribution over the constellation points, the association between the constellation points and the bit predictions can be defined. Note that the formulation has been restricted to bit mappings that are bijective for brevity's sake. This means that every constellation point maps to exactly one-bit string and that each bit string that is mapped is unique. The bit probability can then be defined as

$$P(b_{ij}=1 \mid \mathbf{y}, \theta) = \sum_{\substack{k=1 \\ (\mathcal{M}_b(\mathbf{s}_k))_j=1}}^{|S|} P(c_{y_i} = s_k \mid \mathbf{y}, \theta). \quad (4)$$

Equation (4) utilizes that the probability $P(b_{ij}=1 \mid \mathbf{y}, \theta)$ corresponds to the probability that $\mathbf{s}_k$ should be demapped as one of the symbols that maps to bit j. Thus, it can be computed by adding the constellation point probabilities of points that map to 1 at the index j in the bit string. An explicit term for the probability of each constellation point has been derived. Thus, a mapping-independent formulation is presented. To demodulate with a different mapping, one can change $\mathcal{M}_b$ to a desired new mapping $\hat{\mathcal{M}}_b$. The appropriate probabilities are then directly obtained without adjusting the DNN weights. The described procedure performs an exact computation of the LLR. However, mapping independence can also be achieved with an approximate LLR. This can be computed, for example, by taking the maximal probabilities of constellation points corresponding to a bit value of 1 and 0. Doing so potentially reduces system performance while allowing higher throughput. These effects would be more noticeable the larger the constellation is.

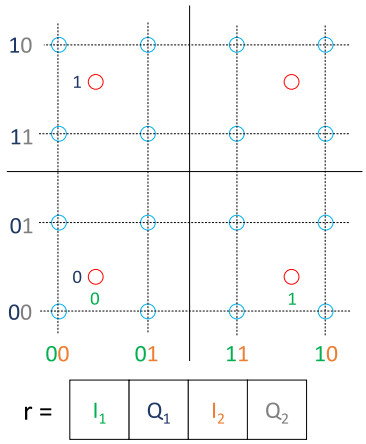

FIGURE 1. An example of QPSK and 16QAM overlaid. The representation is determined by counting from left to right and from bottom to top in Gray code. Afterward, interleave the in-phase and quadrature components to get a hierarchy. The representation values match the position in the representation by color.

B. QAM REPRESENTATION

Although the property of mapping independence is important, it does not address the hierarchical structures in many of the constellation families. Figure 1 depicts the hierarchical structure. Here, a QPSK constellation is overlaid by a 16-QAM constellation. There are two important observations here. First, there is an identical symmetry component for both QPSK and 16-QAM. For QPSK, only two bits are needed to encode the location of a point; one bit is used to determine the sign of the real axis, and the other is used to determine the sign of the imaginary axis. As such, it is not necessary to encode a probability for each individual symbol. Two binary probabilities, one for each sign, are sufficient to compute all symbol probabilities. Second, there are shared decision boundaries between QPSK and 16-QAM. The bit mappings depicted here allow us to define the first two bits of the 16-QAM constellation. These bits correspond to the sign of the real and imaginary components. Thus, these bits can be shared with the QPSK constellation, an observation made previously by [22]. Having a form of binary representation thus makes sense in the context of 4^n-QAM constellations.

Therefore, a form of shared binary representation that respects the hierarchical relationship between different QAM constellations is proposed. This representation is referred to as r and a mapping $\mathcal{M}_r$ is defined. This mapping maps a constellation point to the binary representation r. The binary representation can be defined in a manner similar to [22] and Figure 1. The neural network is then defined as

$$\log \frac{P\big(r_{ij}=1 \mid \mathbf{y}, \theta, r_{i1}, \ldots, r_{i(j-1)}\big)}{P\big(r_{ij}=0 \mid \mathbf{y}, \theta, r_{i1}, \ldots, r_{i(j-1)}\big)} = \Big(f_\theta^{(\text{repr})}\big(\hat{\mathbf{x}}\big)\Big)_{ij}. \quad (5)$$

Note that no assumption on the independence of each representation bit is made. It is also not assumed that $\mathcal{M}_r$ is bijective. It could be desirable to have a binary representation in which elements of the representations are dependent on prior elements in the representation. The formulation in (5) is the final formulation that is used for all experiments. To map the representation bits to a bit string, the symbol probabilities are first computed as

$$P\big(c_{y_i}=s_k \mid \mathbf{y}, \theta\big) = \prod_{j=1}^{R} P\big(r_{ij}=(\mathcal{M}_r(s_k))_j \mid \mathbf{y}, \theta, r_{i1}, \ldots, r_{i(j-1)}\big). \quad (6)$$

Here, R is introduced as the total size of the representation. The symbol probabilities can be computed by sequentially multiplying all the elements in the representation for a given symbol. For the resulting receiver, a sequence of defined formulas can be used. The sequence of (4), (5), and (6) is used. Consequently,all variants of 4^n-QAM can be modeled with this formulation. It also incorporates mapping independence, improving over [22]. For 4^n-QAM, the special case where $\mathcal{M}_r$ is an extension of $\mathcal{M}_b$ occurs. For this case, the following theorem holds assuming an exact LLR computation.

Theorem 1: $\mathcal{M}_b$ is extended by $\mathcal{M}_r \rightarrow P(b_{ij}=1 \mid y, \theta) = P(r_{ij}=1 \mid y, \theta, r_{i1}, \ldots, r_{i(j-1)})$

Proof: $P(b_{ij}=1 \mid \mathbf{y}, \theta) = \sum_{\substack{k=1 \\ (\mathcal{M}_b(\mathbf{s}_k))_j=1}}^{|S|} P(c_{y_i}=s_k \mid \mathbf{y}, \theta)$

$= \sum_{\substack{k=1 \\ (\mathcal{M}_b(\mathbf{s}_k))_j=1}}^{|S|} \prod_{l=1}^{R} P(r_{il} = (\mathcal{M}_r(s_k))_l \mid \mathbf{y}, \theta, r_{i1}, \ldots, r_{i(l-1)})$

$= \sum_{\substack{k=1 \\ (\mathcal{M}_b(\mathbf{s}_k))_j=1}}^{|S|} \prod_{l=1}^{B} P(r_{il} = (\mathcal{M}_b(s_k))_l \mid \mathbf{y}, \theta, r_{i1}, \ldots, r_{i(l-1)})$

$= \sum_{\substack{k=1 \\ (\mathcal{M}_b(\mathbf{s}_k))_j=1}}^{|S|} P(r_{ij}=(\mathcal{M}_b(s_k))_j \mid \mathbf{y}, \theta, r_{i1}, \ldots, r_{i(j-1)}) * \prod_{\substack{l=1 \\ l\neq j}}^{B} P(r_{ij}=(\mathcal{M}_b(s_k))_l \mid \mathbf{y}, \theta, r_{i1}, \ldots, r_{i(l-1)})$

$= P(r_{ij}=1 \mid \mathbf{y}, \theta, r_{i1}, \ldots, r_{i(j-1)}) \prod_{\substack{l=1 \\ l\neq i}}^{B} P(r_{il}=0 \mid \mathbf{y}, \theta, r_{i1}, \ldots, r_{i(j-1)}) + P(r_{il}=1 \mid \mathbf{y}, \theta, r_{1l}, \ldots, r_{i(j-1)})$

$P(b_{ij}=1 \mid \mathbf{y}, \theta) = P(r_{ij}=1 \mid \mathbf{y}, \theta, r_{i1}, \ldots, r_{i(j-1)})$ ■

To describe the proof in text, the first two steps are applications of the definitions of Eq. (4) and Eq. (6). In Eq. (8), the hypothesis that $\mathcal{M}_b$ is extended by $\mathcal{M}_r$ is applied. Then, in Eq. (9), the case where bit l is the same as bit j is factored. The sum is then moved inward, resulting in the product becoming a product on terms that all evaluate to 1. The term that was originally moved out of the product sum becomes $P(r_{ij}=1 \mid \mathbf{y}, \theta, r_{i1}, \ldots, r_{i(j-1)})$. This is a result of moving the sum inward. This step also concludes the proof. This result implies that the formulation for 4^n-QAM is a generalization of direct modeling of LLRs with a neural network. The framework should thus be as accurate as models directly optimized on the LLR. However, it has mapping independence. As a side note, non-square constellation can also be added to the representation. This can be done by appending all and not including them in the hierarchical structure. Doing so gives a shared representation for 4^n-QAM and separate representations for the other QAM constellations.

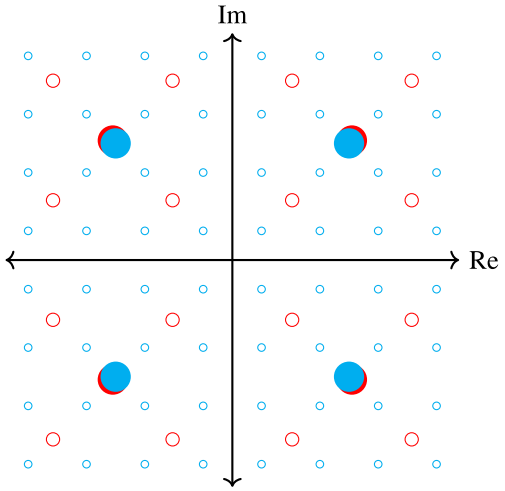

FIGURE 2. Comparison of 16-QAM and 64-QAM. The red circles represent 16-QAM with the large red circle being the average for each quadrant. We presented the same in green for 64-QAM. While the averages for both constellations are quite similar, they are not exactly equal. This causes a small reduction in BER when masking is performed.

a) Scaling: An issue exists with the masking hierarchy found by [22]. The concept of this hierarchy is that 16-QAM can be considered as an extension of QPSK, 64-QAM as an extension of both 16-QAM and QPSK, etc. However, looking at Fig. 1, there is a deviation in this hierarchy visible. For 16-QAM to be a proper extension of QPSK, it is expected that the constellation points of QPSK align with those of 16-QAM. An appropriate alignment based on the given decision boundaries would be as follows. First, take the average of the four constellation points with positive real and imaginary components of 16-QAM. Then, align this average with the QPSK point having a positive real and imaginary component.

However, this is not achieved when normalizing a signal based on the average power. The averaged points in the 16-QAM constellation have the same angle as the QPSK constellation points but a slightly smaller amplitude. This is not an issue when only 16-QAM and QPSK are combined, as the decision boundaries between points still align perfectly. However, this causes problems when including a third constellation from this family. Although the alignment

TABLE 1. The multiplication factors used for different QAM constellations.

	QPSK	16-QAM	64-QAM	256-QAM
γ	1	1.118	1.146	1.152

between 64-QAM and 16-QAM is much better than between QPSK and 16-QAM, there is a discrepancy. This has been illustrated in Fig. 2. This discrepancy becomes an issue in a multi-demapper that demaps, e.g., four different types of 4^n-QAM constellations. This demapper then achieves worse BER rates than a neural receiver that only has to demap one type.

Note that if the signal power of a received signal is normalized, the constellations can be mapped to any desired space through multiplication. Thus, an appropriate hierarchy for 4^n-QAM can be constructed with a scaling factor. This ensures that the underlying demapping problem is on the same scale for all of these constellations. Given the set S^{4^n-QAM} that contains all complex-valued constellation points for a certain constellation, define the set of $S_+^{4^n-\text{QAM}} = \{s \in S^{4^n-\text{QAM}} \mid Re(s) > 0 \ \ \& \ \ Im(s) > 0\}$. Then, one can compute the scaling factor for a constellation as

$$\gamma = \frac{\sum_{c \in S_+^{4^n-\text{QAM}}} c}{\sum_{s \in S_+^{4^n-\text{QAM}}} s}. \tag{7}$$

Using this formula, the appropriate scaling factors for the 4^n-QAM constellations are presented in Table 1. By multiplying a received normalized signal by these factors, the received signal can be lifted back into the same space. The hierarchical approach with a scaling factor is only applicable to 4^n-QAM constellations.

C. APSK REPRESENTATION

The advantage of separating $\mathcal{M}_r$ and $\mathcal{M}_b$ and defining $\mathcal{M}_r$ as non-bijective mapping is not immediately apparent for 4^n-QAM. For square QAM constellations, a masking-based approach can encode all of these with a bijective bit mapping. However, constellation groups generally do not allow solutions where $\mathcal{M}_r$ is bijective. Thus, elements of ∇ are not necessarily independent of each other. Furthermore, it is not always practical to assume that the bit mapping is the same. Different standards may have different bit mappings that can be decoded in a similar way. An important example of this is the family of circular APSK constellations.

These types of constellation are common in the DVB-S2 standard [39]. They come in various configurations and shapes, but can often be defined on the basis of a few aspects. These include the number of amplitude levels, the number of phase levels, the distance between each amplitude level, and the phase offset per amplitude level. For convenience, from now on the different levels of amplitude shift are referred to as *circles*. Note that no assumption has to be made that the outer circle has an amplitude of 1. Similarly to traditional receivers, the only assumption necessary is that the amplitude level of each circle is consistent.

Algorithm 1 Get APSK Representation of Constellation

Require: none
Ensure: Returns updated representations for QAM, circle, and family bits

Initialize $q \leftarrow$ zeros(representation_size $\times$ #symbols)
Determine the quadrant in which each point lies
for each point **do**
  if point in bottom-left quadrant **then**
    Assign quadrant bits 00 in q
  else if point in top-left quadrant **then**
    Assign quadrant bits 10 in q
  else if point in bottom-right quadrant **then**
    Assign quadrant bits 01 in q
  else
    Assign quadrant bits 11 in q
  end if
end for
Number the circles from innermost to outermost
for each point **do**
  Compute the binary value with $\lfloor current\_circle * 2^{\lceil \log_2(total\_circles) \rceil} / (num\_circles - 1) \rfloor$
  Assign the Gray-coded binary representation of the circle in q
end for
for each quadrant and circle **do**
  Extract all points within the circle and quadrant
  Determine their order based on their (counter-)clockwise angle. The top-right and bottom-left have a clockwise order
  Number them using: $\lfloor current\_point * 2^{\lceil \log_2(total\_points) \rceil} / (total\_points - 1) \rfloor$
  Gray-code these values and assign them to the symbols in q
end for
Return q

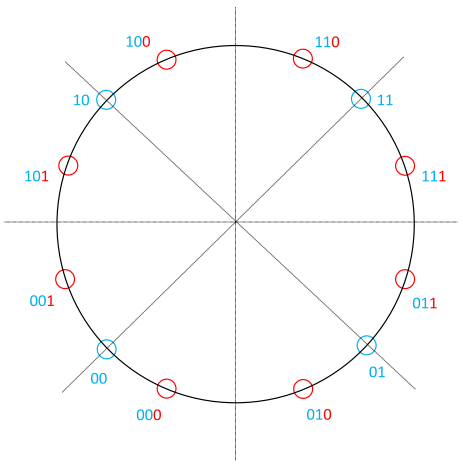

FIGURE 3. A QPSK (blue) and offset 8-PSK (red) constellation overlaid. The latter constellation hierarchically relates to the former, as by adding one more bit (red) to the two bits (blue), the quadrants can be split in two.

For many of the constellations in the DVB-S2(x) standard, there are a few commonalities. The circles in many APSK constellations have a phase offset of π/M where M is the number of constellation points on that circle. The simplest instances of this family are BPSK and QPSK. To better illustrate this family, QPSK overlaid with offset 8-PSK was drawn in Fig. 3. A potential Gray mapping for both was

included. The figure shows that the QPSK points split the offset 8-PSK points exactly through the middle for each quadrant.Interestingly, the real and imaginary axis lines form decision boundaries for both constellations. For QPSK, they form the set of optimal hard decision boundaries. For offset 8-PSK, the lines through the QPSK points have to be added. The hard-decision boundaries between QPSK and offset 8-PSK thus overlap. This means that only a demapper for offset 8-PSK is needed to demap both QPSK and offset 8-PSK. It is thus clear that these two constellations form a hierarchy.

This hierarchy can be used to build an efficient and more general representation for APSK. The relationship in Fig. 3 holds when the number of points in the circle is a power of two starting at four. However, many APSK constellations have a number of points from these different from those. Nevertheless, they still obey this hierarchical relationship as long as the phase offset is π/M. Note that this phase offset can thus differ per circle, as each circle can have a different number of points. The points that form a hierarchy follow $4 \times 2^n \times d$ where $n = 0, 1, 2, \ldots$ and $d = 1, 3, 5, \ldots$. This is equivalent to the set of all odd numbers. Intuitively, the hierarchies are formed with an odd number d at the base. Every factor of 2^n then forms a set in which the constellation points are split again in half. For example, when there are 12 points on a circle, the decision boundaries are perfectly contained by a circle with 24 points. As these circles do not allow for a perfect binary representation a non-bijective dependent mapping is necessary.

The process of generating the binary representation per circle is divided into two steps. First, the bits belonging to a quadrant are determined. The bits belonging to *family* are then determined. Here, family refers to the value of d that applies to the number of points in a circle. For example, a circle with 12 points is part of the 3-family. The general procedure is described in Algorithm 1. In each quadrant and for every circle, points are ordered based on their angle and then Gray coded. Doing so ensures that the representation has the smallest error possible. There are two important details to the labeling process. First, for only three points, they are not labeled as $0, 1, 2$ but as $0, 1, 3$. This procedure ensures that the labeling respects the hierarchical relationship that was established. Second, the top right and bottom left quadrants are labeled clockwise. This ensures that neighboring points in different quadrants have the same representation bits. The quadrant bits themselves correspond to the representation of QPSK. This representation is common in all 4^n-QAM constellations and most APSK constellations. These bits can then be combined with the 4^n-QAM representation. This is done by computing then separately for further efficiency. These quadrant bits are computed on the basis of the sign of the real and imaginary components.

One more aspect is the configuration of the amplitudes per circle. APSK constellations often consist of multiple circles, with each circle having a different amplitude. Although the outermost circle in a normalized APSK constellation usually has an amplitude of around 1, the inner circles can vary depending on the configuration. The optimal decision boundaries for the circles differ on the basis of the configuration. The assumption is made that there is a known set of configurations, such as inthe DVB-S2x standard. For each configuration, a separate set of circle bits is determined. It is important not to share the circle configuration. The network becomes less capable of demapping when the circle bits are shared.

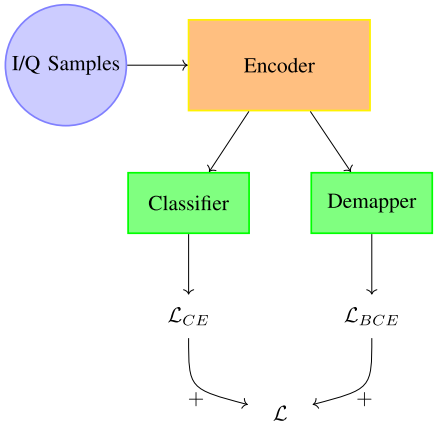

FIGURE 4. The structure of a multi-branch AMR and Demapper model.

Finally, some APSK constellations do not follow any of the structures discussed above. A simple example of this is 8-PSK. As 8-PSK has a phase offset of 0, the decision boundaries do not form a hierarchy. A solution to include these constellations is to extend $\mathcal{M}_r$ with $\mathcal{M}_b$ directly. By doing so, no loss of performance for these constellations is guaranteed by Theorem 1. Thus, any arbitrary constellation can be included in the framework. It is also important to note that the framework does not consider phase ambiguity. As with other receivers, some mechanic is needed to address phase ambiguity. Resolving phase ambiguity does not affect the framework, however. If the framework is unable to resolve phase ambiguity, a standard neural receiver is unlikely to be able to resolve it, also.

D. EXTENSION TO AUTOMATIC MODULATION RECOGNITION

As neural demapping has now been extended to simultaneously demap multiple constellations to increase parameter efficiency, it must be asked whether this task can be further extended. To do so, the link between the simultaneous demapping of multiple types of constellation and multi-task learning is drawn. The shared challenges in channel estimation, equalization, and demapping are used to more efficiently demap multiple constellations. In that context, it can be seen that the pipeline could be extended to other tasks. It is observed that many tasks in communication are rendered trivial in the absence of complex impairments, implying that the most important task that a neural network must perform is the identification and correction of channel impairments.

For that purpose, automatic modulation recognition (AMR) was added to the pipeline. AMR is an interesting fit for inclusion, as the framework's approach already allows demapping multiple types of constellations simultaneously. Thus, it is fairly straightforward to include AMR as well. This can be done using a shared representation learned for both tasks. Combining AMR with simultaneous demapping would be a complementary change. Making the change removes the need for the assumption that the modulation type has been communicated. One issue is that the scaling factor previously discussed cannot be included. This scaling factor violates a fundamental assumption of AMR, as the modulation type must be known. This limits the demapping performance of the model. Note that it is not required to perform the QAM masking approach used. One could treat 4^n-QAM in the same way as the other QAM constellations. This increases memory use and complexity, but removes the need for the scaling factor.

A schematic of the combined model has been included in Fig. 4. The model consists of three components: the encoder, the classifier, and the demapper. The exact architectures of these models can be decided upon on the basis of the setting. In principle, the encoder contains most of the model parameters and has the primary task of correcting channel impairments. The demapper is a small model that first translates the corrected samples to our representation space and then translates them to bit LLRs. Finally, the classifier aggregates all symbols in the transmission and then provides a probability over the set of possible modulation types. For jointly training these models, the BCE loss over the bit LLRs is combined with the cross-entropy (CE) score of the modulation type labels. Note that all models discussed so far can be considered within this framework by leaving out either the classifier or the demapper.

IV. EXPERIMENTAL SETUP

To validate the framework, experiments were run using simulations. The simulations were performed using Sionna [40]. The experiments were first ran in an AWGN setting to validate the approach. Second, simulations were run over an OFDM fading channel to validate whether the conclusions transfer to a more realistic setting.

A. AWGN

To evaluate the performance of the method, the data was simulated over a simple AWGN channel. This was done to better understand how the method behaves in various combinations of constellations. As an AWGN channel allows for a simple optimal hard-decision demodulation rule, it was used as a bound to benchmark the method. The simulation pipeline is depicted in Fig. 5.

To perform our experiments, a simple DNN receiver was used as depicted in Table 2. The encoder takes the raw I/Q samples and consists of three hidden layers of size 256 each, followed by a ReLU activation function and a batch normalization layer. This model was chosen because it has sufficient capacity to approach the BER lower bound for the neural receiver. The DemapHead module then receives the output from the encoder and returns either the bit LLRs directly for the standard neural receiver or the representation bits for the multi-neural receiver. The ClassifierHead is only included when the model performs AMR. It first performs global average pooling over the time dimension and then passes the output through a one-layer deep neural network.

The rest of the hyperparameters are detailed in Table 3. These settings are included for reproducibility; the code will be released on GitHub. For the AWGN channel, the noise was sampled uniformly at random between -5 and 20 E_b/N_0. As we perform simulations, every sequence is new. The number of symbols per block refers to the number of I/Q samples encoded at each step. Coding is done with a 5G compliant LDPC de / encoder with a code rate of $1/2$ [41]. For training, the coding module is not included as

TABLE 2. AWGN experimental configuration.

Layer Type	Configuration
Encoder	
Dense	Filters: 256, Activation: ReLU
BatchNormalization	
Dense	Filters: 256, Activation: ReLU
BatchNormalization	
Dense	Filters: 256, Activation: ReLU
BatchNormalization	
ClassifierHead	
GlobalAveragePooling1D	
Dense	Units: 256, Activation: ReLU
Dense	Units: Number of Classes
DemapHead	
Dense	Filters: 256, Activation: ReLU
BatchNormalization	
Dense	Units: Output Size

TABLE 3. AWGN system parameters.

Parameter	Value
Seeds	42 to 44
Batch size	32
Number of Epochs	150
Batches per Epoch	500
Optimizer	AdamW
Learning Rate	0.004
Weight Decay	0.01
#Symbols per block	512
Filter Type	Square Root Raised Cosine
Span in Symbols	10
Samples per Symbol	4
Roll Off Factor	0.25
SNR Range	-5 dB to 20 dB

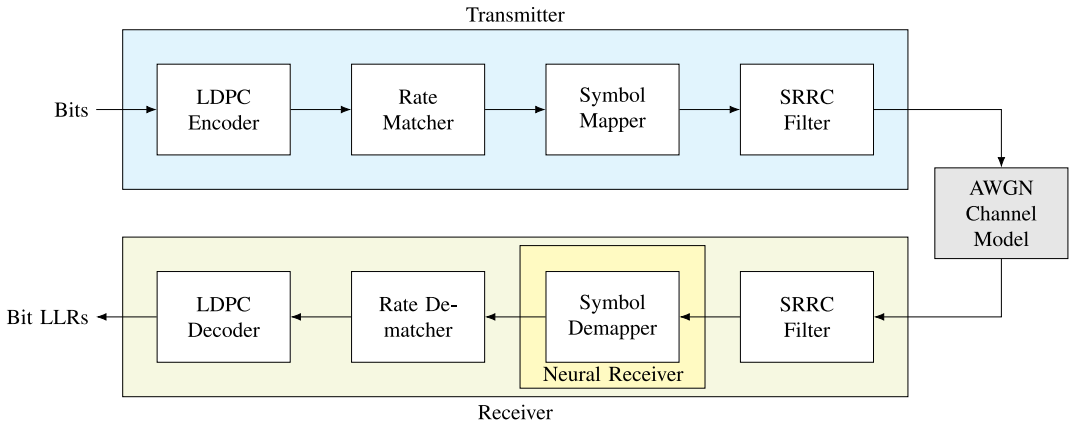

FIGURE 5. The AWGN link-level simulation that was used for part of the experiments.

it does not affect the optimal decision boundary. The batch size depends on the number of constellation types included. For the baseline neural receiver, a batch size of 32 is used. For the multi-receiver, 32 is multiplied by the total number of constellations incorporated during training. In addition to the optimization settings mentioned in the table, we use a scheduler that reduces the learning rate by a factor of 10 in epochs 100 and 125. As new data is simulated every batch, there is no relevant dataset size. The only impairment used in the AWGN channel is the Gaussian noise element.

Each sequence has a different SNR, and thus the DNN demapper should learn how to correct for the noise level. The APSK constellations are generated based on the DVB-S2x specification [39]. These constellations are specified by their configuration and code rate. For example, the constellation 64-APSK-7/9 is the APSK constellation with 64 constellation points associated with the code rate 7/9 under this specification. Different code rates generally refer to a different circle radius in the APSK constellations, but may also refer to entirely different layouts. A diverse subset of the 118 constellations is taken for the experiments. For example, there are many variants of 16-APSK with a configuration of $4 + 12$. This means that the inner circle contains 4 points and the outer circle 12 points. Including all of them is not necessarily as interesting as including an entirely different configuration.

To evaluate the method, the following three systems are compared.

1) A baseline approach that computes the bit LLRs based on the squared Euclidean distance of received symbols to every constellation point. As this approach is equal to the MAP-estimator assuming a Gaussian distribution per constellation point, this approach is optimal under AWGN. That is why we included this approach to demonstrate the achievable lower bound within our setup.
2) A neural demapper that directly predicts the bit LLRs. This single receiver allows us to evaluate how a receiver that only has to demap a single constellation performs comparatively.
3) The multi-demapper approach that we trained on multiple constellations. The constellations on which it is trained are QPSK, 16-QAM, 64-QAM, 256-QAM, 8-PSK, 16-APSK-2/3, 16-APSK-100/180, 32-APSK-2/3, 64-APSK-7/9, 64-APSK-128/180, 256-APSK-124/180.

TABLE 4. OFDM system parameters.

Parameter	Value
Number of Epochs	150
Batches per Epoch	500
Subcarrier Spacing (Hz)	30,000
Number of subcarriers	128
Number of OFDM Symbols	14
Number of RX Antenna	2
Number of TX Antenna	1
Pilot Pattern	Kronecker
Pilot Symbol Indices	2, 11
Training CDL Model	C
Evaluation CDL Model	D

B. OFDM

Note that a neural receiver is redundant for an AWGN channel, as its primary objective is correcting channel impairments. A simple SIMO OFDM simulation pipeline is used to further evaluate the method. A schematic for the pipelines is given in Fig. 6. Four different pipelines are compared for the OFDM setting.

1) For the first baseline, perfect channel state information is assumed. This means that ground-truth fading effects are used for equalization. This baseline served as a reference to what is achievable.
2) For the second baseline, LS estimation based on the pilot symbols is performed. These estimates are passed to the equalization module.

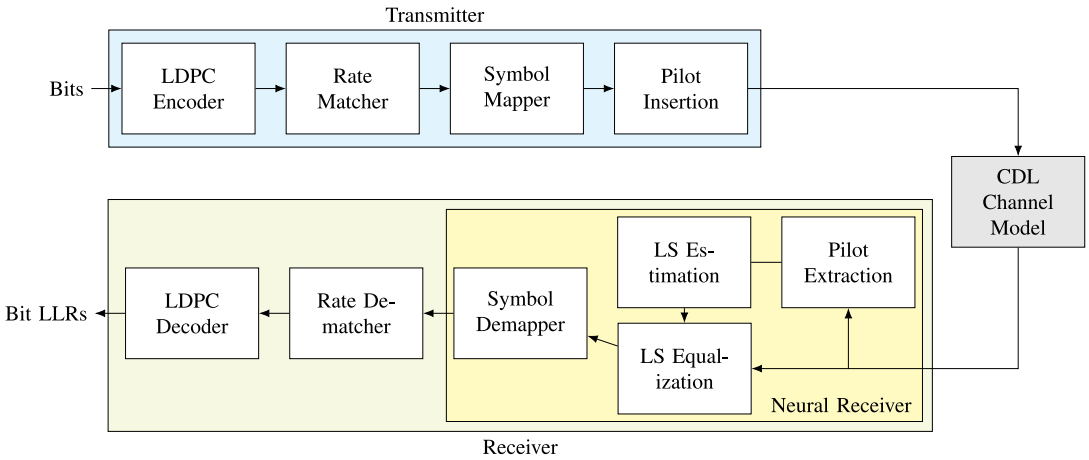

FIGURE 6. The SIMO OFDM pipeline that is used for part of the experiments.

3) The single neural receiver is the DeepRX [22] model. It takes the received I/Q samples, the ground-truth pilot symbols, and the LS estimation based on these pilot symbols as input.
4) The multi-neural receiver. It is built upon DeepRX and extends it with the representation module.

The OFDM system parameters for our method can be found in Table 4. Similar neural network settings were used as they worked well for OFDM too. Thus, each receiver is also trained for 150 epochs using the AdamW optimizer with a learning rate of 0.004. At epochs 100 and 125, the learning rate is reduced by a factor of 10. The receiver has a dual antenna setup with antenna patterns as in the 3GPP TR 38.901 specification. We have opted for a SIMO pipeline as it is sufficiently simple to allow isolating the important factors while still being interesting to evaluate a neural multi-receiver. The combination of a CDL-C for training and a CDL-D model for evaluation was chosen. These channel models are sufficiently general to draw conclusions for SIMO systems. A different channel model for training and evaluation is chosen to ensure that the approaches generalize.

In each training batch, a set of fading channels is sampled from the given CDL-C configuration. The channel coefficients are simulated using this configuration. These coefficients are then multiplied element-wise with the data to be transmitted. Finally, Gaussian noise is generated based on a batch of uniform random SNR values between -5 and 20 E_b/N_0. The noise samples are then added to the batch to simulate the OFDM samples received. As simulations are used, there is again no concept of a dataset size or training/test split. Every generated block is different from every other block. Frame synchronization is assumed to have been accomplished beforehand. This is fair, as this assumption is made for all methods compared. The experiments were run on a compute node that has a partition with 18 cores of an Intel Xeon Platinum 8360Y CPU, 128GB of RAM, and an NVIDIA A100 GPU with 40GB of VRAM.

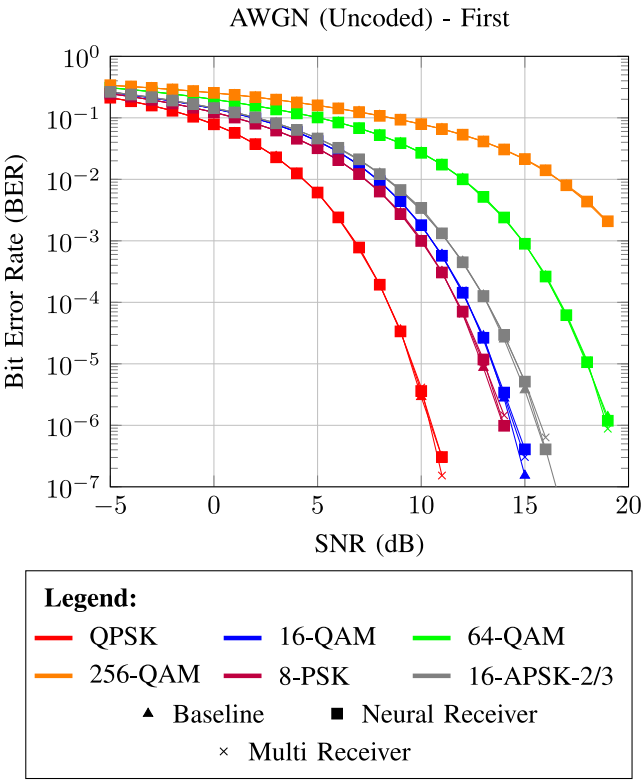

FIGURE 7. BER performance of different methods under AWGN as a function of SNR. This figure contains the first set of six modulations.

Simulations are often sufficient for generalizable conclusions. Real-world demapping datasets are often limited in size and diversity. An alternative is to use the DeepMIMO dataset [42]. This is a highly realistic ray-traced set of scenarios that allows sampling channel coefficients.

V. SIMULATION RESULTS

A. AWGN

First, the methods were compared for AWGN. In Fig. 7, the BER across the entire evaluated SNR range has been

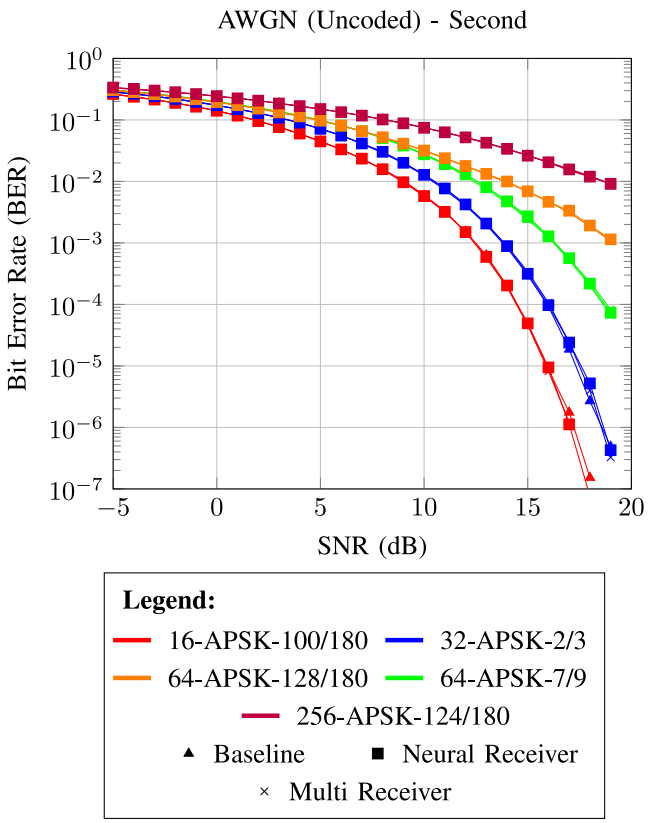

FIGURE 8. BER performance of different methods under AWGN as a function of SNR. This figure contains the second set of five modulations.

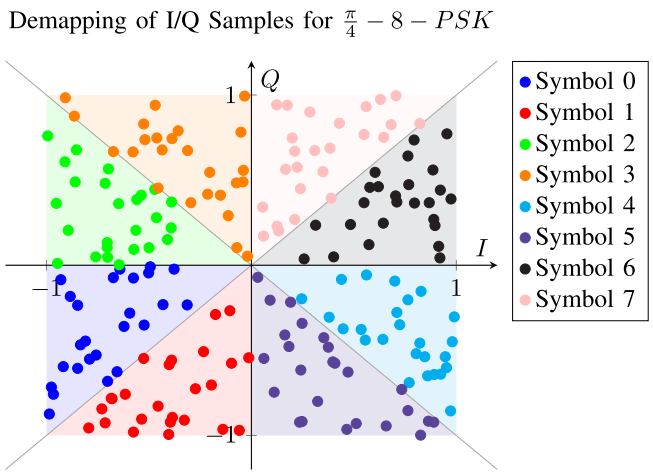

FIGURE 9. Distribution of I/Q samples based on the learned representation. The position is given, and the point color is the model prediction. The background is a visual aid to depict the ground truth. The bits associated with QPSK and the first $1-$ *family APSK* bit are depicted here.

depicted for 6 different constellations. These constellations contain the four constellations that share the QAM bits, i.e., QPSK, 16-QAM, 64-QAM, and 256-QAM. It can be observed that for all of these constellations, both the neural and multi-receiver achieve a performance similar to the baseline. For QPSK, this implies that no performance is lost by sharing the representation with other constellation types. The QPSK constellation is one of the most interesting in this evaluation, as it shares its representation bits with every constellation other than 8-PSK. However, this has not hampered the optimization of this constellation type. The 256-QAM constellation again shares many of its representations with other methods, but it also has many unique representation bits. Furthermore, 256-QAM is one of the most complex constellation types in the setup. It is also relatively hard to fit, since the neural receiver needs to learn to distinguish many more symbols. Again, the approach achieves a performance similar to that of both the baseline and the neural receiver. Similarly, the performance of both the neural receiver and multi-receiver approaches the baseline for both 16-QAM and 64-QAM. One of the smaller *APSK* settings is also depicted in this figure. This constellation was depicted because it does not share many representation bits with other approaches. It is also one of the constellation types for which there is no bijective mapping for the constellation points on the outer circle. However, there does not seem to be a major performance difference between the standard neural receiver and multi-receiver. Finally, the results for $8-PSK$ are presented. Here, it is observed that the performance of the multi-receiver is approximately equal to the baseline and neural receiver. It is indicated by this that, without bit sharing, a similar performance can be achieved.

Second, the remaining five included in Fig. 8 were depicted. These five include three APSK constellations that share all family bits. These include 16-APSK-100/180, 64-APSK-128/180, and 256-APSK-124/180. These have $8+8$, $16+16+16+16$, and $32+32+32+32+32+32+32+32$ as configurations, respectively. It can be observed that these constellations all achieve similar performance to their respective baselines when modeled separately. Consequently, the proposed bit-sharing representation seems valid under AWGN. It can also be seen that the other two constellations achieve a performance similar to their respective baseline.

In general, it is thus observed that the multi-receiver is as accurate under AWGN as the neural receiver that only has to demap a single constellation. This conclusion seems to hold for all included constellations. This indicates that, under AWGN, there is no reason to constrain the set of possible constellations to just one. Using the presented approach, multiple different types of constellation can be included without loss of performance.

a) Learned Representation: To better understand what the method learns, a range of I/Q samples was generated and passed through the multi-receiver. The range of I/Q samples consists of a set of evenly spaced points between the corner points $\{(-1,-1),(1,-1),(-1,1),(1,1)\}$. For each of these samples, the symbol label was determined based on the hard-decision bit labels. For example, if a bit has a hard-demapping of 11, the colour associated with the symbol that demaps to 11 was given. The first representation bit of the $1-$ *family* of the *APSK* representation was also added. According to the intuition of the approach, this representation bit should

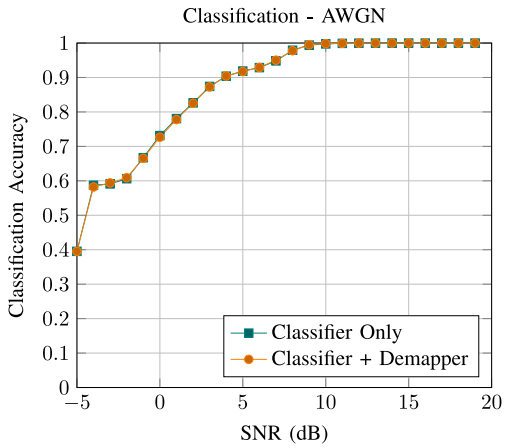

FIGURE 10. AMR performance of different methods under AWGN as a function of SNR.

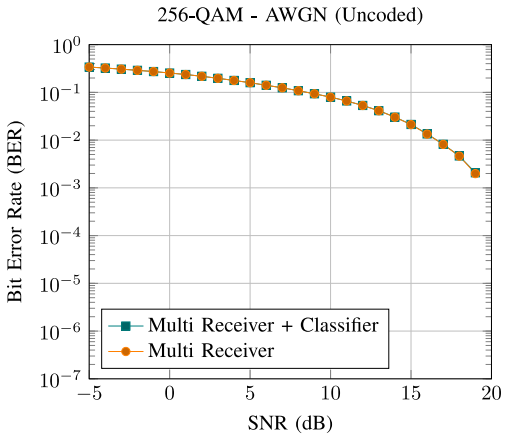

FIGURE 11. BER performance of different methods under AWGN for 256-QAM as a function of SNR.

add the diagonal decision boundaries defined through the points $(-1,-1)$ to $(1,1)$ and $(-1,1)$ to $(1,-1)$. The result of this is depicted in Fig. 9. Although some points around the edges may be classified incorrectly, the points are overall correctly classified based on their position. These results give an idea of what the approach learns. The expected, optimal decision boundaries seem to be the same as the ones learned by the neural multi-receiver. This result underlines the validity of the approach.

b) Demapping + Classification: Finally, the classification head was included in the approach. The multi-receiver was trained with both the demapping head and the classification head simultaneously. To compare, a model with just a classification head was also trained. This experiment allows evaluating whether the tasks of classification and demapping can be performed jointly. The results of the AMR approach are shown in Fig. 10. It can be seen that the accuracy of both is similar. This indicates that the classification task is not hindered by the demapping task performed. To further validate this finding, the BER of the combined model is compared with the original $256-QAM$ multi-demapper

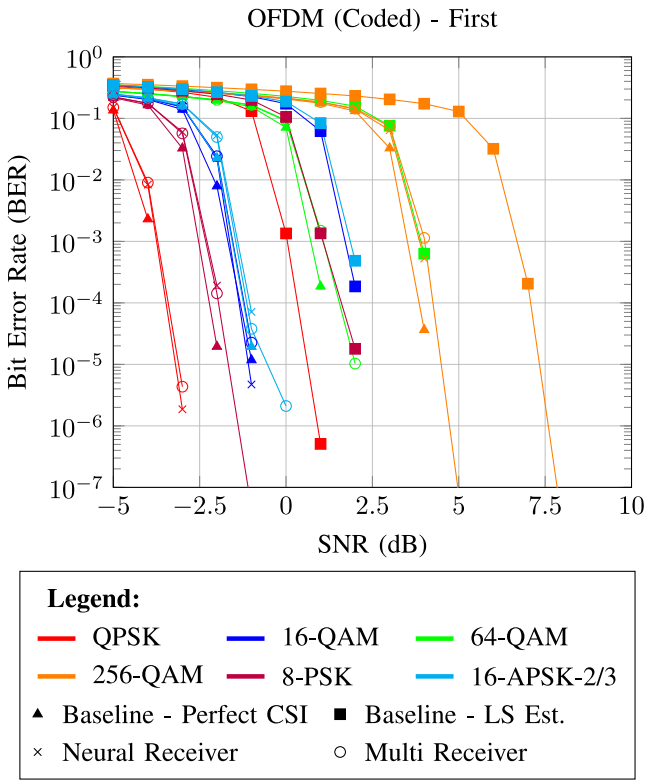

FIGURE 12. Coded BER performance of different methods under OFDM as a function of SNR. This figure contains the first set of six modulations.

results in Fig. 11. Here, the performance is again equivalent. This indicates that the tasks of AMR and bit demapping do not hinder each other. These results demonstrate that it is possible to jointly learn a neural classifier and demapper with a shared representation under AWGN without loss of performance.

B. OFDM PIPELINE

To get a better idea of the practical utility of this work, the framework was also evaluated in a set of four SIMO OFDM pipelines. These pipelines were previously described in the experimental setup.

a) BER Performance: First, the BER results of the pipelines for various modulation types were evaluated. In Fig. 12, the results for the first half of the constellations included in the experimental setup are depicted again. For all constellations, it can be observed that none is as good as the baseline under perfect CSI. This is to be expected, as the perfect CSI baseline under simulation is optimal. The neural receiver and the multi-receiver also beat the LS-estimation significantly, which underlines that an NN-based approach is complementary for both pipelines.

The first four important constellations to consider are the group of QPSK, 16-QAM, 64-QAM, and 256-QAM. Together, these form an interesting family, as they all directly extend each other. Both the neural receiver and multi-receiver achieve relatively similar performance for all four of these constellations. Slight discrepancies between the neural

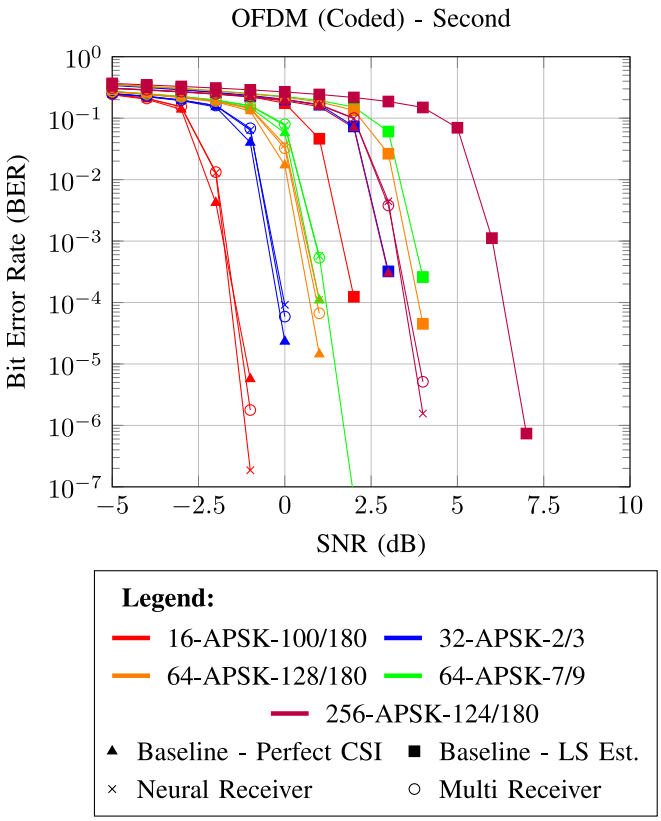

FIGURE 13. Coded BER performance of different methods under OFDM as a function of SNR. This figure contains the second set of five modulations.

receiver and multi-receiver can be observed. Generally, the neural receiver achieves a lower BER when there is a difference. At an SNR of 4, the neural receiver for 256-QAM achieves a clearly lower BER than what we observe for the multi-receiver. Similarly, it can be observed that at an SNR of −3, the neural receiver is also slightly better than the multi-receiver for QPSK. Overall, the BER curve is fairly similar for these constellations. Note that the constellations QPSK, 16-QAM, 64-QAM, and 256-QAM share their representation bits but retain a similar performance as separately trained models. Even under a complicated channel model, masking is able to approach the performance of the single neural receiver for the QAM constellations. Another interesting observation is that the performance of 8-PSK is highly similar as well for both the neural receiver and multi-receiver. As the constellation does not share bits with any other constellations, it demonstrates that adding more general constellations does not harm the BER of these constellations.

In Fig. 13, the BER curves for the remaining constellations are shown. The most important set of constellations to consider here is 16-APSK-100/180, 64-APSK-128/180 and 256-APSK-124/180 as they all share the 1 family for every circle. The performance for these constellations is again similar for both the neural receiver and multi-receiver. This implies that the APSK representation sharing feature also works under a realistic channel model. Furthermore, 32-APSK-2/3 and 64-APSK-7/9 share representation bits for some circles and do not share them for others. However, their performance is also similar for both the neural receiver and multi-receiver. Overall, while there are slight performance discrepancies at some SNR points, the model is overall similarly accurate between both the multi-receiver and neural receiver. This implies that the approach is also valid under OFDM with a realistic channel model.

TABLE 5. Throughput measurements for different systems and modulation methods in Mb/s on an A100 GPU.

Method	Single	Multi
QPSK	4.37	4.25
8-PSK	6.47	6.27
16-QAM	8.46	8.13
16-APSK-2/3	8.41	8.17
16-APSK-100/180	8.44	8.13
32-APSK-2/3	10.05	9.66
64-QAM	11.65	10.98
64-APSK-7/9	11.66	10.99
64-APSK-128/180	11.63	10.97
256-QAM	14.71	12.38
256-APSK-124/180	14.71	12.35

b) Throughput: To get an idea of what the overhead of the approach could be, the throughput values are presented in Table 5. These values entail the time it took between receiving the symbols and computing the bit LLRs. This process was repeated 5 times with 1000 MC iterations each. The minimum time of these repeats were taken. The minimum time is commonly used as it is the least noisy estimate of the average run-time of code. The bitrate was then calculated by dividing the number of demapped bits by the time it took to process them. The experiments were run on a compute node that has a partition with 18 cores of an Intel Xeon Platinum 8360Y CPU, 128GB of RAM, and an NVIDIA A100 GPU with 40GB of VRAM.

In the table, it can be seen that the throughput difference is minimal for methods with a smaller number of symbols. However, for 256 symbols, the performance difference becomes more noticeable. The evaluation of symbol probabilities for 256 symbols can be resource-intensive. Note that the assumption was made here that all neural models are already loaded into memory. While this may be realistic for the multi-model, it may prove less realistic for the standard neural receiver models. Thus, it is shown that the performance impact of using the approach under pessimistic assumptions is only slight.

c) Demapping + AMR: Finally, the addition of a classification head to DeepRX is evaluated. To form a classification head, the last residual block of DeepRX was taken and duplicated for the classification head. A global average pooling layer and a dense layer were added at the end to arrive at the classification model. Again, training the

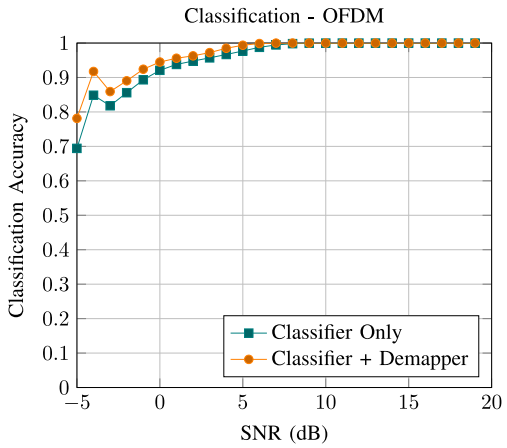

FIGURE 14. AMR performance of different methods under OFDM as a function of SNR.

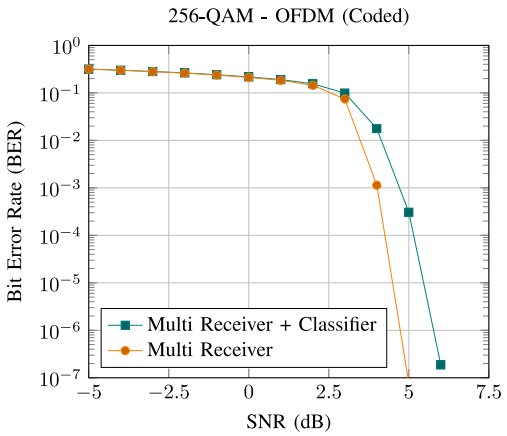

FIGURE 15. BER performance of different methods under OFDM for 256-QAM as a function of SNR.

classifier only was compared with the combination of the classifier and demapper.

The results were drawn in Fig. 14. Surprisingly, the combination of a classifier and demapper is not just as accurate but significantly more accurate than training the classifier. This would imply that the task of demapping helps in learning the task of classification. This result is consistent with the work by [36]. However, they have multiple loss functions that operate directly on the underlying data. The approach presented here can perform this process fully end-to-end. Intuitively, these results make sense, as the learning task becomes more structured when demapping is included. The AMR task needs to deal with at least two types of variance: the variance caused by the channel and the variance caused by the symbol transitions. When demapping is included in the objective, the latter becomes given.

To validate the demapper of the combined model, it is compared with the previously evaluated demapper in Fig. 15. Here, it can be seen that the combined model is less effective than the model that just performs demapping. It seems that the cost of augmenting the classification performance under OFDM is that the demapping performance is worse. In conclusion, the classification head under the shared model performs better than the model purely trained with a classification objective. This indicates that the joint demapper can provide additional information for AMR approaches. This allows them to more quickly converge to a more effective AMR approach without increasing the complexity of the downstream model. This work thus also presents a contribution to AMR.

VI. CONCLUSION

We proposed a framework that allows us to jointly train on and generalize to multiple types of constellations. In this framework, we include mapping independence and the ability to use hierarchical relationships in families of constellations. We found that it is not only possible to train a model jointly on multiple constellations, but that such a method approaches the hard-decision BER bound under AWGN. It also performs similarly to neural receivers under both AWGN and SIMO OFDM channels. Furthermore, we found that combining AMR and demapping results in better AMR accuracies under OFDM.

Thus, the framework introduced opens up the possibility of efficiently modeling families of constellations in the context of DNNs. This is an important step in making DNN receivers sufficiently adaptable in dynamic environments. It also shows the promise of combining various tasks in the communication pipeline. The framework is modular and applicable to any deep learning receiver pipeline. As the approach is implemented as a module applied to every symbol individually, it is also applicable to multi-user MIMO scenarios without further extension. Thus, this work addresses an important issue in the flexibility of neural receivers. The framework applies to many related works, as the approach can be modularized and attached to any neural receiver that outputs bit LLRs. Due to its modular nature, the work can be easily extended to larger datasets and more complex architectures. Future work includes extending the work to MIMO, beamforming, and real-world datasets. Future work could also look at approaches to reducing the exact computation. This is necessary for scaling to very large constellations, such as, e.g., 4096-QAM.

REFERENCES

[1] 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, Dec. 2017.

[2] S. Dörner, S. Cammerer, J. Hoydis, and S. T. Brink, "Deep learning based communication over the air," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 132–143, Feb. 2018.

[3] 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, Nov. 2019.

[4] A. Felix, S. Cammerer, S. Dörner, J. Hoydis, and S. T. Brink, "OFDM-Autoencoder for end-to-end learning of communications systems," in *Proc. IEEE 19th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, 2018, pp. 1–5.

[5] H. Xie, Z. Qin, G. Y. Li, and B.-H. Juang, "Deep learning enabled semantic communication systems," *IEEE Trans. Signal Process.*, vol. 69, pp. 2663–2675, Apr. 2021.

[6] X. Ma and Z. Gao, "Data-driven deep learning to design pilot and channel estimator for massive MIMO," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5677–5682, May 2020.

[7] F. A. Aoudia and J. Hoydis, "End-to-end learning for OFDM: From neural receivers to pilotless communication," *IEEE Trans. Wireless Commun.*, vol. 21, no. 2, pp. 1049–1063, Feb. 2022.

[8] D. Korpi, M. Honkala, and J. M. J. Huttunen, "Deep learning-based pilotless spatial multiplexing," in *Proc. 57th Asilomar Conf. Signals, Syst., Comput.*, 2023, pp. 1025–1029.

[9] F. A. Aoudia and J. Hoydis, "Waveform learning for next-generation wireless communication systems," *IEEE Trans. Commun.*, vol. 70, no. 6, pp. 3804–3817, Jun. 2022.

[10] X. Lin, R. Liu, W. Hu, Y. Li, X. Zhou, and X. He, "A deep convolutional network demodulator for mixed signals with different modulation types," in *Proc. IEEE 15th Int. Conf. Dependable, Auton. Secure Comput., 15th Int. Conf. Pervasive Intell. Comput., 3rd Int. Conf. Big Data Intell. Comput. Cyber Sci. Technol. Congr. (DASC/PiCom/DataCom/CyberSciTech)*, 2017, pp. 893–896.

[11] M. Zhang, Z. Liu, L. Li, and H. Wang, "Enhanced efficiency BPSK demodulator based on one-dimensional convolutional neural network," *IEEE Access*, vol. 6, pp. 26939–26948, 2018.

[12] A. S. Mohammad, N. Reddy, F. James, and C. Beard, "Demodulation of faded wireless signals using deep convolutional neural networks," in *Proc. IEEE 8th Annu. Comput. Commun. Workshop Conf. (CCWC)*, 2018, pp. 969–975.

[13] M. Soltani, V. Pourahmadi, A. Mirzaei, and H. Sheikhzadeh, "Deep learning-based channel estimation," *IEEE Commun. Lett.*, vol. 23, no. 4, pp. 652–655, Apr. 2019.

[14] N. Samuel, T. Diskin, and A. Wiesel, "Learning to detect," *IEEE Trans. Signal Process.*, vol. 67, no. 10, pp. 2554–2564, May 2019.

[15] H. Wang et al., "Deep learning for signal demodulation in physical layer wireless communications: Prototype platform, open dataset, and analytics," *IEEE Access*, vol. 7, pp. 30792–30801, 2019.

[16] T. Wu, "CNN and RNN-based deep learning methods for digital signal demodulation," in *Proc. Int. Conf. Image, Video Signal Process.*, 2019, pp. 122–127.

[17] H. Ye, G. Y. Li, and B.-H. Juang, "Power of deep learning for channel estimation and signal detection in OFDM systems," *IEEE Wireless Commun. Lett.*, vol. 7, no. 1, pp. 114–117, Feb. 2018.

[18] E. Balevi, A. Doshi, and J. G. Andrews, "Massive MIMO channel estimation with an untrained deep neural network," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2079–2090, Mar. 2020.

[19] Y. Wang, H. Lu, and H. Sun, "Channel estimation in IRS-enhanced mmWave system with super-resolution network," *IEEE Commun. Lett.*, vol. 25, no. 8, pp. 2599–2603, Aug. 2021.

[20] S. Cammerer, J. Hoydis, F. A. Aoudia, and A. Keller, "Graph neural networks for channel decoding," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, 2022, pp. 486–491.

[21] M. Goutay, F. A. Aoudia, J. Hoydis, and J.-M. Gorce, "Machine learning for MU-MIMO receive processing in OFDM systems," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2318–2332, Aug. 2021.

[22] M. Honkala, D. Korpi, and J. M. J. Huttunen, "DeepRx: Fully convolutional deep learning receiver," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 3925–3940, Jun. 2021.

[23] Z. Zhao, M. C. Vuran, F. Guo, and S. D. Scott, "Deep-waveform: A learned OFDM receiver based on deep complex-valued convolutional networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2407–2420, Aug. 2021.

[24] S. Cammerer et al., "A neural receiver for 5G NR multi-user MIMO," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, 2023, pp. 329–334.

[25] D. Korpi, M. Honkala, J. M. Huttunen, and V. Starck, "DeepRx MIMO: Convolutional MIMO detection with learned multiplicative transformations," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2021, pp. 1–7.

[26] T. Raviv, S. Park, O. Simeone, Y. C. Eldar, and N. Shlezinger, "Online Meta-learning for hybrid model-based deep receivers," *IEEE Trans. Wireless Commun.*, vol. 22, no. 10, pp. 6415–6431, Oct. 2023.

[27] T. Raviv and N. Shlezinger, "Data augmentation for deep receivers," *IEEE Trans. Wireless Commun.*, vol. 22, no. 11, pp. 8259–8274, Nov. 2023.

[28] J. Pihlajasalo et al., "Deep learning OFDM receivers for improved power efficiency and coverage," *IEEE Trans. Wireless Commun.*, vol. 22, no. 8, pp. 5518–5535, Aug. 2023.

[29] A. Gansekoele, A. Balatsoukas-Stimming, T. Brusse, M. Hoogendoorn, S. Bhulai, and R. van der Mei, "A machine learning approach for simultaneous demapping of QAM and APSK constellations," 2024, *arXiv:2405.09909*.

[30] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2224–2287, 3rd Quart., 2019.

[31] B. Ozpoyraz, A. T. Dogukan, Y. Gevez, U. Altun, and E. Basar, "Deep learning-aided 6G wireless networks: A comprehensive survey of revolutionary PHY architectures," *IEEE Open J. Commun. Soc.*, vol. 3, pp. 1749–1809, 2022.

[32] Z. Qin, H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep learning in physical layer communications," *IEEE Wireless Commun.*, vol. 26, no. 2, pp. 93–99, Apr. 2019.

[33] J. Hoydis, F. A. Aoudia, A. Valcarce, and H. Viswanathan, "Toward a 6G AI-native air interface," *IEEE Commun. Mag.*, vol. 59, no. 5, pp. 76–81, May 2021.

[34] X. You et al., "Towards 6G wireless communication networks: Vision, enabling technologies, and new paradigm shifts," *Sci. China Inf. Sci.*, vol. 64, pp. 1–74, Jan. 2021.

[35] H. He, X. Yu, J. Zhang, S. Song, and K. B. Letaief, "Message passing meets graph neural networks: A new paradigm for massive MIMO systems," *IEEE Trans. Wireless Commun.*, vol. 23, no. 5, pp. 4709–4723, May 2024.

[36] S. Hanna, C. Dick, and D. Cabric, "Signal processing-based deep learning for blind symbol decoding and modulation classification," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 82–96, Jan. 2022.

[37] X. Gao, S. Jin, C.-K. Wen, and G. Y. Li, "ComNet: Combination of deep learning and expert knowledge in OFDM receivers," *IEEE Commun. Lett.*, vol. 22, no. 12, pp. 2627–2630, Dec. 2018.

[38] J. M. J. Huttunen, D. Korpi, and M. Honkala, "DeepTx: Deep learning Beamforming with channel prediction," *IEEE Trans. Wireless Commun.*, vol. 22, no. 3, pp. 1855–1867, Mar. 2023.

[39] A. Morello and V. Mignone, "DVB-S2: The second generation standard for satellite broad-band services," *Proc. IEEE*, vol. 94, no. 1, pp. 210–227, Jan. 2006.

[40] J. Hoydis et al., "Sionna: An open-source library for next-generation physical layer research," 2023, *arXiv:2203.11854*.

[41] "3rd generation partnership project; technical specification group radio access network; requirements for evolved UTRA (E-UTRA) and evolved UTRAN (E-UTRAN); (Release 7), Version 7.3.0," 3GPP, Sophia Antipolis, France, Rep. TR 25.913, 2006.

[42] A. Alkhateeb, "DeepMIMO: A generic deep learning dataset for millimeter wave and massive MIMO applications," in *Proc. Inf. Theory Appl. Workshop (ITA)*, 2019, pp. 1–8.