IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 29 March 2024, accepted 7 May 2024, date of publication 13 May 2024, date of current version 31 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3400367

RESEARCH ARTICLE

Hypernetwork Based Model-Driven Channel Neural Decoding

YUANHUI LIANG[1,2,3], (Member, IEEE), CHAN-TONG LAM[1], (Senior Member, IEEE), QINGLE WU[1,2,3], BENJAMIN K. NG[1], (Senior Member, IEEE), AND SIO-KEI IM[4], (Member, IEEE)

[1]Faculty of Applied Sciences, Macao Polytechnic University, Macau, China
[2]School of Automation and Information Engineering, Sichuan University of Science and Engineering, Yibin 644000, China
[3]Artificial Intelligence Key Laboratory of Sichuan Province, Yibin 644000, China
[4]Macao Polytechnic University, Macau, China

Corresponding author: Chan-Tong Lam (ctlam@mpu.edu.mo)

This work was supported by the Research Funding of Macao Polytechnic University, Macao SAR, China under Project RP/FCA-01/2023.

ABSTRACT Channel decoding algorithms based on model-driven deep learning, also known as channel neural decoding algorithms, have received a lot of attention in recent years. However, the internal parameters and number of layers of the current channel neural decoding algorithm cannot be changed after training. Once changed, retraining of the channel neural decoding network is required. Hypernetwork is a neural network that can generate internal parameters for the main neural network to reduce the training cost of the main neural network and improve the flexibility of the main neural network. In this study, a novel hypernetwork based channel neural decoder is proposed for neural belief propagation algorithms (NBP), including the neural normalized min-sum (NNMS) and neural offset min-sum (NOMS) algorithms. According to the type of information interaction between the hypernetwork and the main decoding network, hypernetwork-based channel neural decoders can be divided into two types: static and dynamic. The internal parameters of the static hypernetwork-based channel neural decoder can be updated as needed without retraining of the main network. In addition to this benefit, the number of layers of the dynamic hypernetwork-based channel neural decoder can also be adjusted. Experimental results show that, compared with the existing NNMS decoding algorithms, the proposed hypernetwork-based NNMS decoding algorithms can achieve better performance on both low-density parity-check (LDPC) and Bose-Chaudhuri-Hocquenghem (BCH) codes.

INDEX TERMS Model-driven, hypernetwork, channel neural decoding, LDPC codes, BCH codes.

I. INTRODUCTION

Channel decoding algorithm is an important part of the communication physical layer and is used to improve the reliability of communication. As deep learning has achieved great success in the fields of computer vision, speech signal processing, and natural language processing, deep learning-based channel decoding algorithms, also called channel neural decoding, have become a hot research topic in recent years. Driven by the application of the Internet of Things and research on sixth-generation (6G) mobile communications, there is an even greater need for low-power, low-latency, and low-complexity decoders for short and moderate codes. Deep learning has been proven to improve the belief propagation (BP) decoding performance of short and moderate codes, which provides a new direction for research on decoders with balanced performance and complexity [1]. Channel neural decoding can be mainly divided into two classes [1], one is model-free (data-driven) deep learning based decoder, and the other is model-driven deep learning based decoder. The model-free deep learning based decoder uses general neural network architectures, such as deep neural networks (DNN), convolution neural network (CNN), recurrent neural network (RNN), etc.,

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Wang.

to decode without using communication expert knowledge, which is a data-driven approach. The authors in [2] used general neural networks to decode random codes and achieve maximum a posteriori (MAP) decoding performance for very short block codes. Due to the shortcoming that neural network (NN)-based decoders are only effective for very short block codes, the authors in [3] proposed to replace the sub-components of the polar decoding algorithm with NN-based decoders. In this way, the scalability of channel decoding based on deep learning is achieved. The authors of [4] and [5] proposed RNN and CNN-based decoders to decode convolutional and turbo codes, respectively. The authors in [6] uses syndrome decoding to eliminate the overfitting problem that often occurs when training codeword sets. The model-free method generally uses fully connected neural network or RNN to implement the iterative process of the decoding algorithm. However, this method lacks interpretability for decoding performance, it is difficult to analyze the reliability of decoding results, and usually requires a large number of trainable parameters.

The model-driven deep learning based decoder is obtained by parametrizing traditional decoding algorithms, where the Tanner graph is unfolded into a neural network in which learning weights are added to each edge. The unfold method was proposed in [7] and has now become a bridge between traditional communication iterative algorithms and model-driven deep learning based communication algorithms. Using the unfold method, the authors in [8] and [9] proposed the neural belief propagation (NBP) algorithm, where the BP algorithm is first unfolded into a neural network, and trainable learning weights are added to every edge in tanner graph. The experimental results show that the performance of NBP decoding algorithm for high-density parity check (HDPC) code exceeds that of BP decoding algorithm. The authors in [10] unfolded offset min-sum (OMS) into the form of a neural network and proposed the neural offset min-sum (NOMS) algorithm, which has lower computational complexity than the NBP algorithm. The authors then applied the syndrome loss to the neural normalized min-sum (NNMS) algorithm [11], thus realizing the unsupervised training of channel neural decoding algorithm. The authors in [12] and [13] then applied the NNMS algorithm to the long low-density parity-check (LDPC) code. The authors in [14] proposed a high-performance neural min-sum (MS) decoding method, which makes full use of the lifting structure of the protograph LDPC (P-LDPC) codes. The authors of [15] proposed a deep learning based decoder for polar codes, and gave the hardware architecture of the basic calculation block in the decoder. The authors of [16] proposed more hardware-friendly OMS and scaling offset min-sum (SOMS) decoders for Polar codes. The differences in hardware implementation between OMS and multiple scaling min-sum (MSMS) decoders are also given. The authors of [17] and [18] proposed a hardware architecture of Syndrome-based DL decoder, which is suitable for implementation in FPGA devices. Channel neural decoding surpasses traditional decoding in terms of decoding performance, but there are still many difficulties in highly mobile time-varying channel environments and noise levels. Once the neural decoder is fixed after training, the learning weights and the number of layers cannot be changed. Once the noise level or the number of layers changes, it needs to be retrained and the stability and optimality of the algorithm can no longer be guaranteed.

Hypernetwork, first proposed in [19], uses one network to generate internal parameters for another network to reduce training costs and improve the flexibility of the layers of the main neural network [20]. To increase the adaptability of machine learning (ML)-based multiple input multiple output (MIMO) detection to different channel environments and noise levels, the authors of [21] and [22] introduced hypernetwork into ML-based MIMO detection and proposed HyperMMNet and HyperEPNet, respectively. The authors in [23] proposed the HyperRNN architecture for an end-to-end downlink channel estimation scheme for massive MIMO frequency division duplex (FDD) systems, achieving lower normalized mean square error (NMSE) for channel estimation and higher sum-rate for beamforming. The authors in [24] applies hypernetwork to the channel estimation algorithm in Reconfigurable Intelligent Surface (RIS) aided communication system.

The authors of [25] turned the message graph of algebraic block codes into a graph neural network (GNN) whose learning weights were generated using a hypernetwork. Thus solving the challenge that GNNs for message passing are difficult to train. However, [25] only focuses on the BP decoding algorithm, and does not involve the more hardware-friendly min-sum algorithm. Compared with BP algorithm, NMS and OMS algorithms have lower computational complexity and storage complexity, which are more suitable for hardware implementation [26]. Similarly, compared with the NBP algorithm, NNMS and NOMS algorithms have been widely studied and applied because of their lower computational complexity and storage complexity [9], [10], [12], [13], [14]. NNMS and NOMS are, respectively, unfolded from the NMS and OMS algorithms into the form of neural networks [9]. In recent years, based on the NNMS algorithm, the authors in [27] and [28] proposed a higher performance version NNMS+ algorithm and a lower complexity version based on tensor-train (TT) decomposition TT-NNMS+ algorithm, so that different versions can be selected for application according to needs. Similarly, the authors of [29] proposed a low-complexity neural OMS (NOMS) algorithm based on tensor-ring (TR) decomposition. However, these variants of NNMS and NOMS algorithms require retraining when the learning weights need to be updated or the number of layers needs to be changed, which requires a large training cost. That is, the current channel neural decoding algorithms lack adaptability to changes in the environment and decoding network structure.

In this paper, we propose a novel hypernetwork based channel neural decoder for the neural belief propagation algorithm (NBP), including the neural normalized min-sum (NNMS) and neural offset min-sum (NOMS) algorithms. We divide hypernetworks designed for channel neural decoders into two types, namely static hypernetwork and dynamic hypernetwork. Static hypernetwork can provide learning weights for the entire main network, but cannot adapt to changes in the number of layers of the main decoding network without redesign and redeployment of the hypernetwork. By generating the learning weights for each layer of the main neural decoder through a separate hypernetwork unit, the dynamic hypernetwork can extend channel neural decoders to an arbitrary number of layers without redesign and redeployment. Our main contributions can be summarized as follows:

- We propose the static hypernetwork based channel neural decoding algorithms, including the NNMS algorithm and NOMS algorithm. The hypernetwork generates learning weights for the main channel neural decoding algorithm, avoiding repeated training of the main neural decoder when the internal parameters (learning weights) need to be updated.
- We propose channel neural decoding algorithms, including NNMS algorithm and NOMS algorithm, based on the dynamic hypernetwork, in which a separate hypernetwork unit is used to generate weights for each layer of the main neural decoder. In addition to the advantage that the internal parameters can be updated, the number of layers of the main neural decoding networks can be dynamically expanded without redesign and redeployment of the hypernetworks.
- We perform extensive experiments to validate the proposed algorithms on LDPC, BCH and Hamming codes. The results show that the performance of the proposed hypernetwork-based NNMS decoding algorithm is about $0.1 \sim 0.2$ dB better than the original NNMS decoding algorithm.

The remainder of this paper is organized as follows. In Section II, we provide the preliminaries, including the NNMS algorithm, NOMS algorithm, the NBP algorithm and hypernetwork. In Section III, we describe the static hypernetwork frameworks for channel neural decoding algorithms. We propose dynamic hypernetwork frameworks for channel neural decoding algorithms in Section IV. The experimental results are provided in Section V, followed by the conclusion in Section VI.

II. PRELIMINARIES

A. NNMS DECODING

The layers of the NNMS algorithm can be divided into three types: check node (CN) layer, variable node (VN) layer and output layer. Assume that a codeword $\mathbf{x}$ is transmitted through a communication system with BPSK modulation, the channel noise conforms to Gaussian distribution, and the receiving vector is $\mathbf{y}$. The log-likelihood ratio (LLR) of the v-th bit of the received signal vector can be computed as

$$l_v = log(\frac{P(y_v|x_v = 0)}{P(y_v|x_v = 1)}) = \frac{2y_v}{\sigma^2} \tag{1}$$

where σ^2 is the variance of the channel noise, y_v is the observation value of the v-th value of the received vector.

The decoding process of LDPC codes can be represented by iterative messages passing between VNs and CNs. The message $u^t_{v,c}$ propagated from the VN v to CN c for iteration t is given by

$$u^t_{v,c} = l_v + \sum_{c' \in N(v) \backslash c} u^{t-1}_{c',v} \tag{2}$$

where $N(v)$ represents the set of CNs connected to VN v, $N(v) \setminus c$ represents the set $N(v)$ excluding CN c.

The difference between the calculation of CNs in the NMS algorithm and NNMS algorithm is that a trainable learning weight vector is added, as shown below [12]

$$u^t_{c,v} = \prod_{v' \in M(c) \backslash v} \mathtt{sign}(u^{(t-1)}_{v',c}) \min_{v' \in M(c) \backslash v} \left| \alpha^{(t-1)}_{v',c} \times u^{(t-1)}_{v',c} \right| \tag{3}$$

where $\alpha^{(t-1)}_{v',c}$ is the added trainable learning weight, $u^t_{c,v}$ is the message transmitted from CNs to VNs for iteration t, $u^{t-1}_{v',c}$ is the messages transmitted from VNs to CNs for iteration $t-1$, $M(c)$ is the set of all VNs connected to CN c, $M(c) \setminus v$ is the set $M(c)$ excluding VN v.

Similarly, the difference between the calculation of VNs in the NMS and NNMS algorithms is that a trainable learning weight vector is added, as shown below [12]

$$u^t_{v,c} = l_v + \sum_{c' \in N(v) \backslash c} \beta^{(t)}_{c',v} \times u^{(t)}_{c',v} \tag{4}$$

where $\beta^{(t)}_{c',v}$ is the added trainable learning weight. CN and VN layers constitute the hidden layers of NNMS neural network.

The calculation of the output layer of the NNMS algorithm directly inherits the output calculation of the NMS algorithm. The soft output message s^t_v of VNs for the maximum number of iterations is given by

$$s^T_v = l_v + \sum_{c' \in N(v)} u^T_{c',v}, \tag{5}$$

$$o_i = \sigma(s^T_v) \tag{6}$$

where $\sigma(x) = (1 + e^{-x})^{-1}$ is the activation function used in the NNMS decoding algorithm and T is the maximum number of iterations.

At the end of the decoding iteration, the final decision can be written as

$$\hat{x}_v = \frac{1 - \mathtt{sgn}(s_v)}{2} \tag{7}$$

where $\mathtt{sgn}$ is a sign function, $\mathtt{sgn}(s_v)$ means the sign of s_v.

There are two loss functions that are often used during the training of the channel neural decoding algorithm, one is the cross-entropy loss function, and the other is the syndrome

loss function. The cross-entropy loss function is defined as follows

$$L(x, o) = -\frac{1}{N}\sum_{i=1}^{N} x_i \log(o_i) + (1 - x_i)\log(1 - o_i). \quad (8)$$

where o_i is the i-th bit of the estimated information output by the NNMS algorithm, and x_i is the i-th bit of the transmitted codeword. The cross-entropy loss function $L(x, o)$ refers to the concept of cross-entropy to measure the difference between the output $\mathbf{o}$ of NNMS decoding and the source codeword $\mathbf{x}$.

Based on NNMS algorithm, the authors in [27] proposed the NNMS+ algorithm, which uses different learning weights for different edges [27]. They replace the trainable learned weight vector $\alpha_{v',c}^{(t-1)}$ in the NNMS algorithm with a trainable learned weight matrix $\mathcal{A}_{v',c}^{(t-1)} \in \mathbb{R}^2$. The calculation of the check layer (3) is updated as follows

$$U_{c,v}^{t} = \prod_{v' \in M(c)\setminus v} \mathtt{sign}(U_{v',c}^{(t-1)}) \min_{v' \in M(c)\setminus v} |\mathcal{A}_{v',c}^{(t-1)} \times U_{v',c}^{(t-1)}| \quad (9)$$

where $U_{c,v}^{t}$ is the calculation of the messages from CNs to VNs for iteration t.

From the calculation of the CN layer of the NNMS algorithm (3) to the calculation of the CN layer of the NNMS+ algorithm (9), the trainable learning weights are changed from a one-dimensional vector to a two-dimensional matrix. The decoding performance of the NNMS+ algorithm has been greatly improved, at the cost of increasing computational complexity. We reduced the computational complexity of the NNMS+ algorithm by using tensor decomposition and joint-way in [27] and [28].

B. NOMS

The model-driven NOMS algorithm is similar to the NNMS algorithm, including an input layer, multiple iterative hidden layers, and an output layer. The hidden layer is composed of a CN layer and a VN layer. In the hidden layer, it is the same except that the multiplication parameter of the CNs layer becomes a subtraction parameter. The CNs layer is given as [12]

$$u_{c,v}^{t} = \max(\min_{v' \in M(c\setminus v)} |u_{v',c}^{t}| - \beta_{v',c}^{t}, 0) \cdot \prod_{v' \in M(c\setminus v)} \mathtt{sign}(u_{v',c}^{t}). \quad (10)$$

where the learning parameter is $\beta_{v',c}^{t}$, which is called the offset factor. $\beta_{v',c}^{t}$ is used to correct the error caused by the OMS decoding algorithm approaching the BP decoding algorithm.

The VNs layer is given as [10]

$$u_{v,c}^{t} = w_{v,c}^{t}(l_v + \sum_{c' \in N(v\setminus c)} u_{c',v}^{t-1}). \quad (11)$$

where the learning parameter is $w_{v,c}^{t}$.

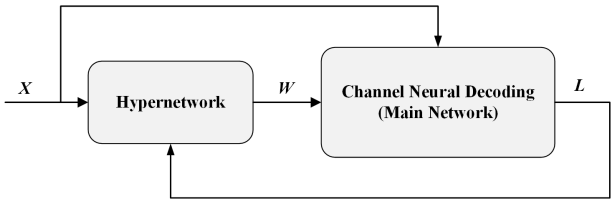


FIGURE 1. Overview of Channel Neural Decoding with Hypernetwork.

Finally, the output of each iteration can be expressed as

$$s_v^t = w_v^t(l_v + \sum_{c' \in N(v)} u_{c',v}^{t}), \quad (12)$$

$$o_i = \sigma(s_v^t) \quad (13)$$

where activation function $\sigma(x)$ and and the cross-entropy loss are the same as NNMS algorithm.

C. HYPERNETWORK

Hypernetwork are neural networks that can generate parameters for the main neural network [20], similar to the relationship between a genotype (the hypernetwork) and a phenotype (the main network) in nature [19]. As shown in Fig. 1, the main network is a neural network that performs learning tasks, essentially learning a parameterized function $g(:, W)$ based on input data to complete the required tasks. The learning parameters W of the main network are provided by a hypernetwork, which is the main function of the hypernetwork. More precisely, the hypernetwork is also a neural network, and its goal is to learn another parameterized function f from the input X to generate W, which are the learning parameters of the main network.

Hypernetwork can increase the adaptability of the main network. Avoid retraining the main network when the internal parameters or the number of network layers need to be changed, which greatly reduces the training cost [19], [22]. In addition, there are some hypernetwork solutions that can improve the performance of the main network [25].

III. STATIC HYPERNETWORK-BASED CHANNEL NEURAL DECODING

Hypernetwork solution consists of two neural networks, one is the main channel neural decoding network [13], and the other is the hypernetwork, which generates learning weights for the main network. The hypernetwork consists of a dense neural network of l layers. We use the `tanh` function as the activation function of the hypernetwork. The input of the hypernetwork is the same as the input of the main network, which is the received log likelihood ratio (LLR) value vector [27]. The hypernetwork is trained using stochastic gradient descent (SGD) under the guidance of the cross-entropy loss function.

NNMS and NOMS decoding algorithms have a very good balance between complexity and performance, which makes them very promising to be applied to communication systems. The position of learning weights in these two

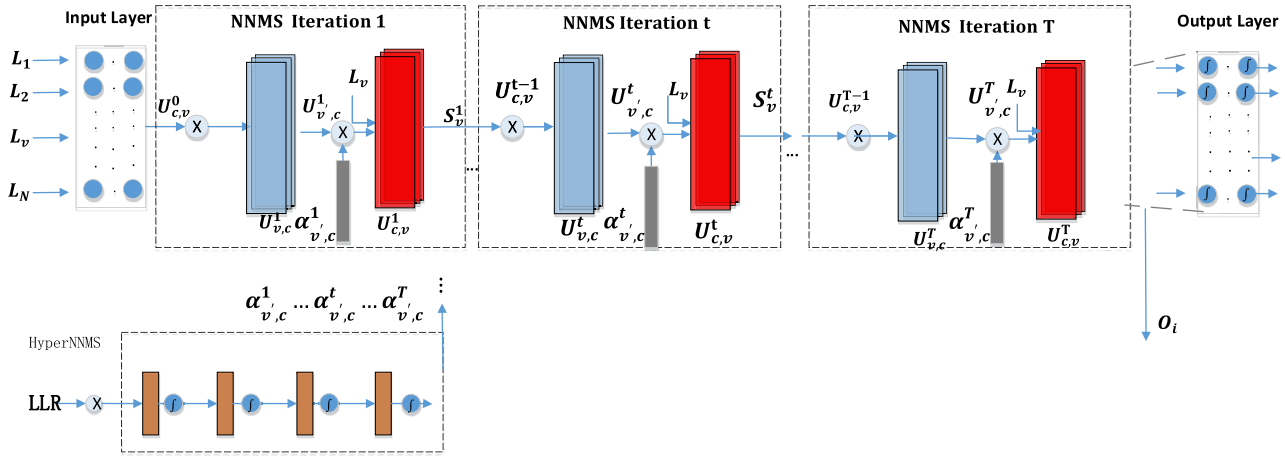


FIGURE 2. Static Hypernetwork based Channel Neural Decoding.

decoding algorithms and their role in decoding are obviously different. This is the reason why we choose NNMS and NOMS decoding algorithms as the main decoding network.

A. STATIC HYPERNETWORK-BASED NNMS DECODING

In the hypernetwork based NNMS decoding algorithm, hypernetwork provides learning weights for the entire NNMS decoding network. Let the hypernetwork be represented by a parameterized function f. In (3), the learning weights generated by the static hypernetwork can be expressed as

$$u_{c,v}^{t} = \prod_{v' \in M(c)\backslash v} \mathrm{sign}(u_{v',c}^{(t-1)}) \min_{v' \in M(c)\backslash v} \left| f(:, W_{\alpha}^{(t-1)}) \times u_{v',c}^{(t-1)} \right| \tag{14}$$

where $W_{\alpha}^{(t-1)}$ are the learning weights of the static hypernetwork f to generate the learning weights $\alpha_{v',c}^{(t-1)}$. Note that the weights are initialized as Gaussian random numbers with zero mean and unit variance.

As shown in Fig. 2, the static hypernetwork generates all required learning weights $\{\alpha_{v',c}^{1}, \ldots, \alpha_{v',c}^{t}, \ldots, \alpha_{v',c}^{T}\}$ for the entire main NNMS network at once, which makes it unnecessary to retrain the NNMS network when the learning weights of the NNMS decoding network need to be changed. Because in the static hypernetwork based channel neural decoding algorithm, the optimal learning weights can be obtained by training the hypernetwork.

B. STATIC HYPERNETWORK-BASED NOMS DECODING

In the hypernetwork based NOMS decoding algorithm, hypernetwork provides learning weights for the NOMS decoding network, which exist in the update equations of the check nodes. In (10), the learning weights generated by the static hypernetwork can be expressed as

$$u_{c,v}^{t} = \max(\min_{v' \in M(c\backslash v)} |u_{v',c}^{t}| - f(:, W_{\beta}^{(t-1)}), 0) \cdot \prod_{v' \in M(c\backslash v)} \mathrm{sign}(u_{v',c}^{t}). \tag{15}$$

where $W_{\beta}^{(t-1)}$ are the learning weights of the static hypernetwork f to generate the learning weights $\beta_{v',c}^{t}$. Similarly, hypernetwork can generate all required learning weights for the NOMS decoding network at once.

Compared with the original channel neural decoder, the channel neural decoder based on the static hypernetwork has an additional l-layer dense neural network, which requires more storage resources and higher computational complexity. However, the advantage of using hypernetwork is that it can reduce the cost of training channel neural decoders. On the other hand, the cost of complexity can also be reduced through model compression methods [27], [28].

IV. DYNAMIC HYPERNETWORK-BASED CHANNEL NEURAL DECODING

The static hypernetwork can generate learning weights for the entire main network, avoiding retraining the main network when the learning weights of the channel neural decoding network need to be changed. However, the size of the learning weight vectors output by the hypernetwork is fixed. Once the number of layers of the main network changes, the output dimension of the hypernetwork needs to be changed, resulting in the need to redesign and redeploy a suitable hypernetwork [19], [22]. In order to solve this issue, We propose a dynamic hypernetwork solution, in which the hypernetwork dynamically generates learning weights for the main network, which can vary online across layers.

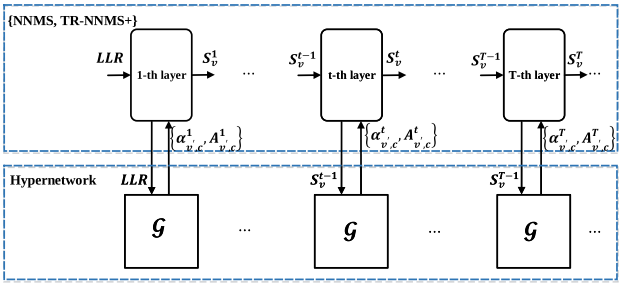


FIGURE 3. Dynamic Hypernetwork based NNMS Algorithm.

A. DYNAMIC HYPERNETWORK-BASED NNMS DECODING

In (3), the learning weights generated by the dynamic hypernetwork can be expressed as

$$u_{c,v}^{t} = \prod_{v' \in M(c)\backslash v} \mathrm{sign}(u_{v',c}^{(t-1)}) \min_{v' \in M(c)\backslash v} \left| f^{(t-1)}(:, W_\alpha) \times u_{v',c}^{(t-1)} \right| \tag{16}$$

where $f^{(t-1)}(:, W_\alpha)$ is the dynamic hypernetwork for NNMS decoding algorithm at the t-th iteration, W_α are the learning weights of the hypernetwork.

Different from the static hypernetwork based NNMS decoding algorithm in which the hypernetwork generates all learning weights for the entire main network at once, in the proposed dynamic hypernetwork based NNMS decoding algorithm, a hypernetwork unit is used to realize information interaction with the main network at each layer. As shown in Fig. 3, the hypernetwork dynamically provides the learning weight vectors $\{\alpha_{v',c}^{t}, (t = 1, \ldots, T)\}$ or the learning weight tensors $\{\mathcal{A}_{v',c}^{t}, (t = 1, \ldots, T)\}$ to the NNMS network at each layer, and the NNMS network provides the information $\{S_v^{t-1}, (t = 1, \ldots, T)\}$ required for calculation to the hypernetwork. Each $\mathcal{G}$ is a hypernetwork unit consisting of a l-layer dense neural network. Different from the static hypernetwork based NNMS decoding algorithm, each hypernetwork unit in the dynamic hypernetwork based NNMS decoding algorithm can share learning weights across layers. In addition, the network structure of each unit in the dynamic hypernetwork solution is the same. Repeated iterations of a unit can be used to implement multiple unit structures. Therefore, when the number of NNMS network layers needs to change, there is no need to redesign and redeploy the hypernetwork solution [19], [22].

B. DYNAMIC HYPERNETWORK-BASED NOMS DECODING

The learning weights for the NOMS decoding algorithm in (10) can be generated by the dynamic hypernetwork as

$$u_{c,v}^{t} = \max(\min_{v' \in M(c\backslash v)} |u_{v',c}^{t}| - f^{(t-1)}(:, W_\beta), 0) \cdot \prod_{v' \in M(c\backslash v)} \mathrm{sign}(u_{v',c}^{t}). \tag{17}$$

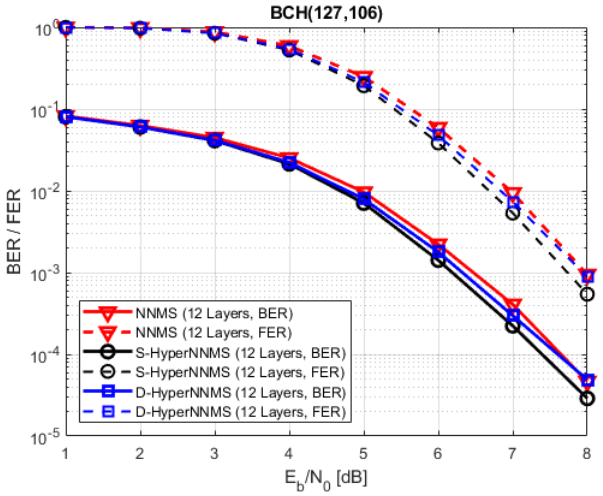


FIGURE 4. BER and FER Performance of the HyperNNMS Algorithms on BCH Code (127,106).

where $f^{(t-1)}(:, W_\beta)$ is the dynamic hypernetwork for NOMS decoding algorithm at the t-th iteration, W_β are the learning weights of the hypernetwork. Similarly, the number of layers of the dynamic hypernetwork-based NOMS decoding algorithm can be changed without redesigning and redeploying the hypernetwork solution.

V. EXPERIMENTAL RESULTS

In this section, we choose three different codes, including BCH, Hamming and LDPC, to verify the performance of our proposed hypernetwork based channel neural decoding algorithms. For the convenience of comparison with the NNMS and NOMS decoding algorithms, we choose the LDPC code used in [12], [13], [27], and [28]. The code rate $r = 0.75$ and the code length $N = 576$. The batch size B for the parallel decoding process, during the inference phase, is set to 120. Tensorflow 2.0 is used as the DL framework for the simulation.

In the simulation experiment, the static hypernetwork is a four-layer deep neural network. There are 32 neurons in every layer. The dynamic hypernetwork consists of multiple units, which are implemented by multiple iterations of a unit, where each unit is a four-layer deep neural network. The *tanh* function and cross entropy are used as activation function and loss function respectively. Randomization is used during the initialization of the algorithm. In the following experiments, for the static and dynamic hypernetworks based channel neural decoding algorithms, only the hypernetwork is trained to generate learning weights for the main decoding network. When the experimental results are shown in the following figures, for convenience, we let S-HyperNNMS and D-HyperNNMS represent the NNMS decoding algorithm based on static and dynamic hypernetworks, respectively. S-HyperNOMS and D-HyperNOMS represent NOMS algorithms based on static and dynamic hypernetworks, respectively.

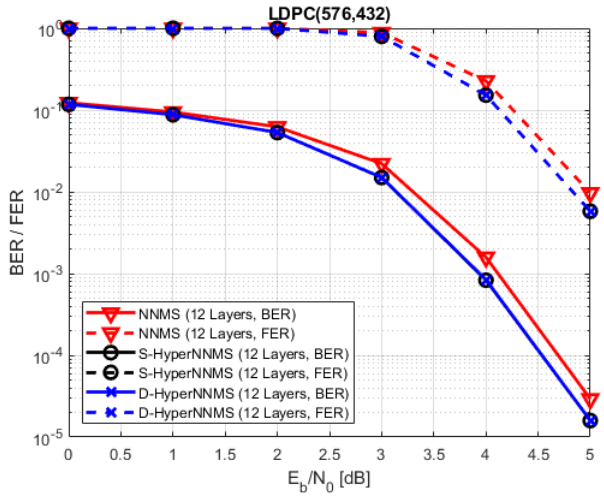


FIGURE 5. BER and FER Performance of the HyperNNMS Algorithms on LDPC Code (576,432).

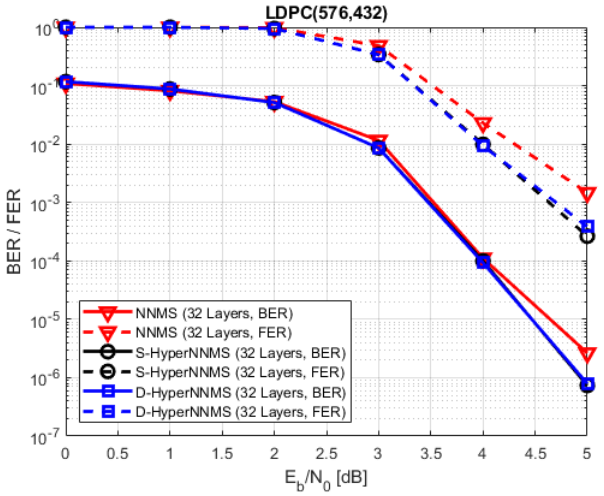


FIGURE 6. BER and FER Performance of the HyperNNMS Algorithms on LDPC Code (576,432).

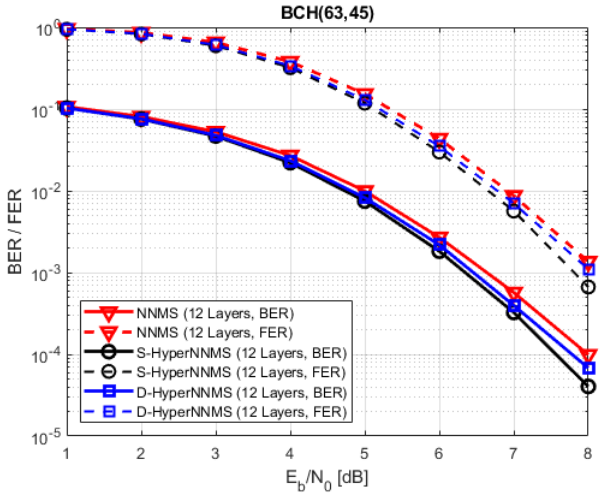


FIGURE 7. BER and FER Performance of the HyperNNMS Algorithms on BCH Code (63,45).

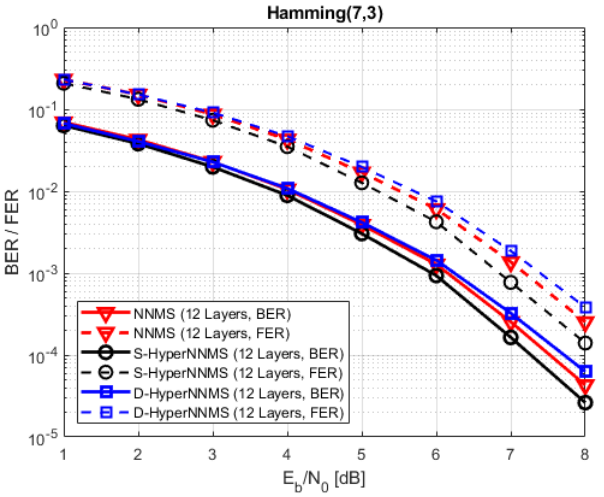


FIGURE 8. BER and FER Performance of the HyperNNMS Algorithms on Hamming Code (7,3).

A. PERFORMANCE OF HYPERNETWORK-BASED NNMS ALGORITHM

Fig. 4 shows the bit error rate (BER) and frame error rate (FER) performance comparison of the proposed static hypernetwork based NNMS decoding algorithm, dynamic hypernetwork based NNMS decoding algorithm and NNMS algorithm for BCH code. The code length of BCH is 127 bits, and the information bit is 106 bits. The number of iterations of the decoding algorithm is set to 5, because there are input and output layers, which is equivalent to a 12-layer NNMS decoding network. It can be seen from Fig. 4 that in the low SNR region, the performance of the static hypernetwork based NNMS decoding algorithm is almost the same as that of the NNMS algorithm. As the SNR increases, the performance of the static hypernetwork based NNMS decoding algorithm outperforms the NNMS algorithm. In both low and high SNR regions, dynamic hypernetwork based NNMS decoding algorithm performs about the same as that of the NNMS algorithm. In the middle SNR region, dynamic hypernetwork based NNMS decoding slightly outperforms the NNMS algorithm. This means that based on the NNMS algorithm, static hypernetwork based NNMS decoding and dynamic hypernetwork based NNMS decoding algorithms can not only avoid retraining of the main network, but also slightly improve BER and FER performance.

In addition, we also conducted experiments on LDPC codes, as shown in Fig. 5 and Fig. 6. In Fig. 5, the NNMS algorithm has almost the same performance as the HyperNNMS algorithm at low SNR. With the increase of SNR, the performance of static hypernetwork based NNMS decoding and dynamic hypernetwork based NNMS decoding algorithms exceed that of NNMS algorithm. Compared with

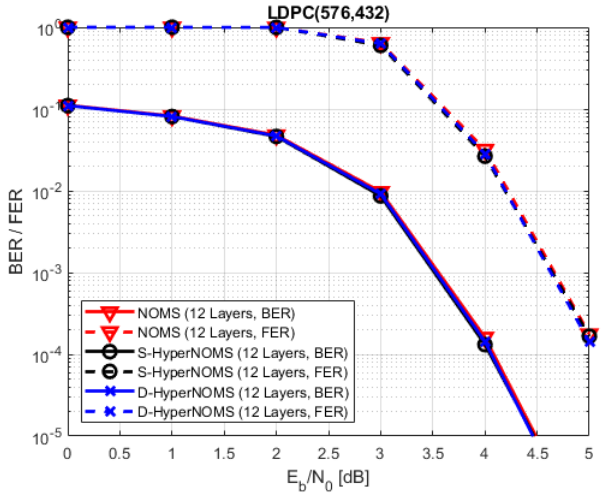


FIGURE 9. BER and FER Performance of the HyperNOMS Algorithms on LDPC Code (576,432).

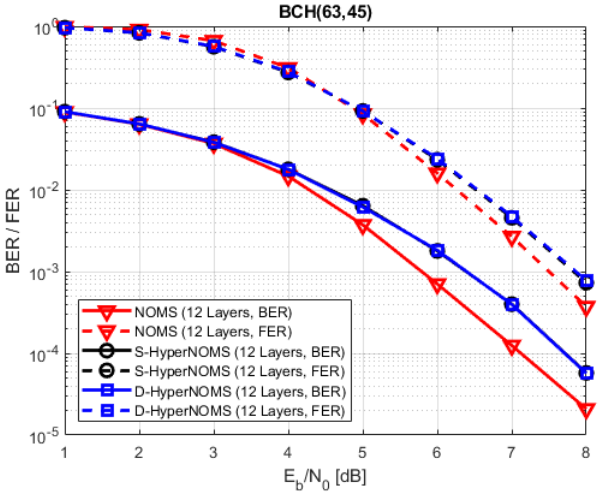


FIGURE 11. BER and FER Performance of the HyperNOMS Algorithms on BCH Code (63,45).

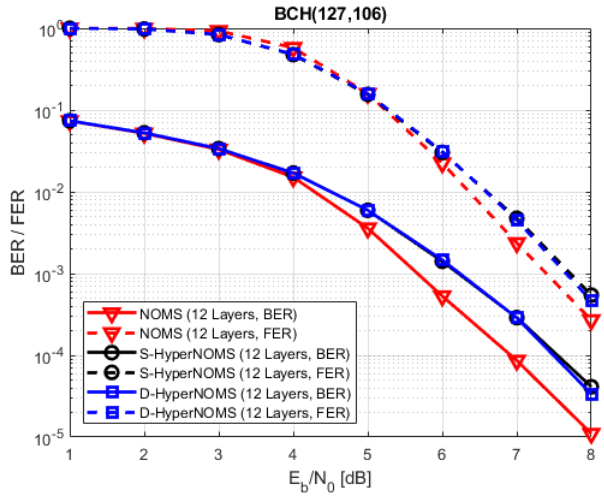


FIGURE 10. BER and FER Performance of the HyperNOMS Algorithms on BCH Code (127,106).

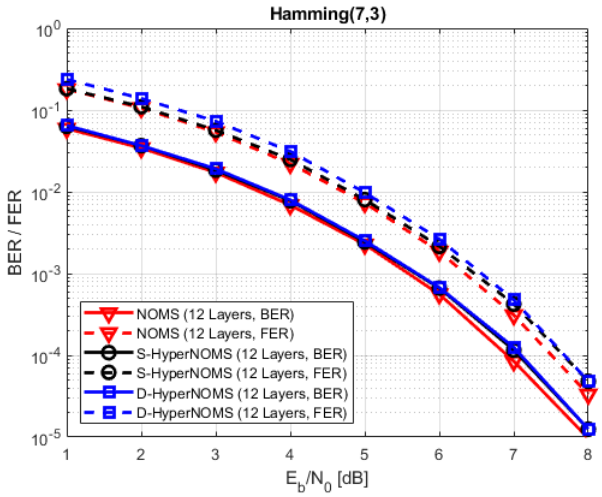


FIGURE 12. BER and FER Performance of the HyperNOMS Algorithms on Hamming Code (7,3).

the case where the decoding network has 12 layers, when the decoding network has 32 layers in Fig. 6, the FER performance advantage of the HyperNNMS algorithm is greater than that of the NNMS algorithm.

We also conducted experiments on BCH (63,45) code and Hamming (7,3) code, and the results are shown in Fig. 7 and Fig. 8. It can be seen from the Fig. 7 and Fig. 8 that with the increase of the SNR, the performance of the static hypernetwork based NNMS decoding algorithm is more and more superior to that of the NNMS algorithm, whether it is a BCH code or a Hamming code. For Hamming code, the performance of dynamic hypernetwork based NNMS decoding is slightly lower than that of the NNMS algorithm. The performance of dynamic hypernetwork based NNMS decoding on Hamming codes is slightly lower than that of NNMS algorithm. For BCH code, the performance of dynamic hypernetwork based NNMS decoding is between that of static hypernetwork based NNMS decoding algorithm and NNMS algorithm.

B. PERFORMANCE OF HYPERNETWORK-BASED NOMS ALGORITHM

The HyperNOMS decoding algorithm has also been verified on LDPC, BCH and Hamming codes, and the results are shown in Fig. 9 to Fig. 12. As can be seen from Fig. 9, the decoding performance of HyperNOMS is almost the same as that of the original NOMS, unlike the performance of HyperNNMS that exceeds the original NNMS algorithm. As for the performance on BCH codes, the performance of both dynamic and static HyperNOMS decoding algorithms has been significantly degraded, which exceeds 0.5dB in the high SNR area, as shown in Fig. 10 and Fig. 11. Finally, the performance results of HyperNOMS on Hamming codes are shown in Fig. 12, which is slightly lower than the

performance of the original NOMS. The reason is that the training of hypernetwork does not make the learning weights of the main NOMS decoding network reach the optimal state.

VI. CONCLUSION

The static hypernetwork based channel neural decoding algorithms have been proposed, which can avoid retraining of the main network when the learning weights of the main decoding network need to be changed. We have also proposed the dynamic hypernetwork based channel neural decoding algorithms, which allows the number of layers of the main decoding network to be changed when needed without redesign and redeployment the hypernetwork. We validated the proposed hypernetwork based NNMS and NOMS decoding algorithms using LDPC, BCH, and Hamming codes. Experimental simulation results show that the performance of the hypernetwork-based NNMS decoding algorithm is slightly better than the original NNMS decoding algorithm for BCH and LDPC codes. The performance of the hypernetwork-based NOMS decoding algorithm has experienced some degradation.

REFERENCES

[1] Y. Choukroun and L. Wolf, "Error correction code transformer," 2022, *arXiv:2203.14966*.

[2] T. Gruber, S. Cammerer, J. Hoydis, and S. T. Brink, "On deep learning-based channel decoding," in *Proc. 51st Annu. Conf. Inf. Sci. Syst. (CISS)*, 2017, pp. 1–6.

[3] S. Cammerer, T. Gruber, J. Hoydis, and S. T. Brink, "Scaling deep learning-based decoding of polar codes via partitioning," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, Dec. 2017, pp. 1–6.

[4] H. Kim, Y. Jiang, R. Rana, S. Kannan, S. Oh, and P. Viswanath, "Communication algorithms via deep learning," 2018, *arXiv:1805.09317*.

[5] Y. Jiang, S. Kannan, H. Kim, S. Oh, H. Asnani, and P. Viswanath, "DEEPTURBO: Deep turbo decoder," in *Proc. IEEE 20th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Jul. 2019, pp. 1–5.

[6] A. Bennatan, Y. Choukroun, and P. Kisilev, "Deep learning for decoding of linear codes–A syndrome-based approach," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 1595–1599.

[7] J. R. Hershey, J. Le Roux, and F. Weninger, "Deep unfolding: Model-based inspiration of novel deep architectures," 2014, *arXiv:1409.2574*.

[8] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *Proc. 54th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2016, pp. 341–346.

[9] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 119–131, Feb. 2018.

[10] L. Lugosch and W. J. Gross, "Neural offset min-sum decoding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 1361–1365.

[11] L. Lugosch and W. J. Gross, "Learning from the syndrome," in *Proc. 52nd Asilomar Conf. Signals, Syst., Comput.*, Oct. 2018, pp. 594–598.

[12] Q. Wang, S. Wang, H. Fang, L. Chen, L. Chen, and Y. Guo, "A model-driven deep learning method for normalized min-sum LDPC decoding," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Jun. 2020, pp. 1–6.

[13] Q. Wang, Q. Liu, S. Wang, L. Chen, H. Fang, L. Chen, Y. Guo, and Z. Wu, "Normalized min-sum neural network for LDPC decoding," *IEEE Trans. Cognit. Commun. Netw.*, vol. 9, no. 1, pp. 70–81, Feb. 2023.

[14] J. Dai, K. Tan, Z. Si, K. Niu, M. Chen, H. V. Poor, and S. Cui, "Learning to decode protograph LDPC codes," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 1983–1999, Jul. 2021.

[15] W. Xu, Z. Wu, Y.-L. Ueng, X. You, and C. Zhang, "Improved polar decoder based on deep learning," in *Proc. IEEE Int. Workshop Signal Process. Syst. (SiPS)*, Oct. 2017, pp. 1–6.

[16] B. Dai, R. Liu, and Z. Yan, "New min-sum decoders based on deep learning for polar codes," in *Proc. IEEE Int. Workshop Signal Process. Syst. (SiPS)*, Oct. 2018, pp. 252–257.

[17] E. Kavvousanos and V. Paliouras, "Optimizing deep learning decoders for FPGA implementation," in *Proc. 31st Int. Conf. Field-Programmable Log. Appl. (FPL)*, Aug. 2021, pp. 271–272.

[18] E. Kavvousanos and V. Paliouras, "A low-latency syndrome-based deep learning decoder architecture and its FPGA implementation," in *Proc. 11th Int. Conf. Modern Circuits Syst. Technol. (MOCAST)*, Jun. 2022, pp. 1–4.

[19] D. Ha, A. Dai, and Q. V. Le, "HyperNetworks," 2016, *arXiv:1609.09106*.

[20] V. Kumar Chauhan, J. Zhou, P. Lu, S. Molaei, and D. A. Clifton, "A brief review of hypernetworks in deep learning," 2023, *arXiv:2306.06955*.

[21] M. Goutay, F. A. Aoudia, and J. Hoydis, "Deep hypernetwork-based MIMO detection," in *Proc. IEEE 21st Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, May 2020, pp. 1–5.

[22] J. Zhang, C.-K. Wen, and S. Jin, "Adaptive MIMO detector based on hypernetwork: Design, simulation, and experimental test," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 65–81, Jan. 2022.

[23] Y. Liu and O. Simeone, "Learning how to transfer from uplink to downlink via hyper-recurrent neural network for FDD massive MIMO," *IEEE Trans. Wireless Commun.*, vol. 21, no. 10, pp. 7975–7989, Oct. 2022.

[24] W.-C. Tsai, C.-W. Chen, C.-F. Teng, and A.-Y. Wu, "Low-complexity compressive channel estimation for IRS-aided mmWave systems with hypernetwork-assisted LAMP network," *IEEE Commun. Lett.*, vol. 26, no. 8, pp. 1883–1887, Aug. 2022.

[25] E. Nachmani and L. Wolf, "Hyper-graph-network decoders for block codes," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11.

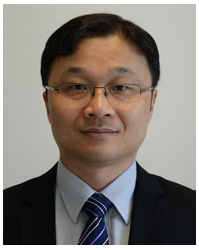
[26] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.

[27] Y. Liang, C.-T. Lam, and B. K. Ng, "A low-complexity neural normalized min-sum LDPC decoding algorithm using tensor-train decomposition," *IEEE Commun. Lett.*, vol. 26, no. 12, pp. 2914–2918, Dec. 2022.

[28] Y. Liang, C.-T. Lam, and B. K. Ng, "Joint-way compression for LDPC neural decoding algorithm with tensor-ring decomposition," *IEEE Access*, vol. 11, pp. 22871–22879, 2023.

[29] Q. Wu, B. K. Ng, Y. Liang, C.-T. Lam, and Y. Ma, "Application of tensor decomposition to reduce the complexity of neural min-sum channel decoding algorithm," *Appl. Sci.*, vol. 13, no. 4, p. 2255, Feb. 2023.

YUANHUI LIANG (Member, IEEE) received the B.A.Sc. degree in mathematics from Northwest Normal University, in 2010, and the M.A.Sc. degree in communication from Southwest Jiaotong University, in 2012. He is currently pursuing the Ph.D. degree in computer applied technology with the Faculty of Applied Sciences, Macao Polytechnic University. From 2012 to 2018, he was an Algorithm Engineer with industry, focusing on the field of communication physical-layer algorithms design. His main research interests include channel neural decoding, the application of tensor decomposition in communication, and machine learning in communications.



CHAN-TONG LAM (Senior Member, IEEE) received the B.Sc. (Eng.) and M.Sc. (Eng.) degrees from Queens University, Kingston, ON, Canada, in 1998 and 2000, respectively, and the Ph.D. degree from Carleton University, Ottawa, ON, Canada, in 2007. He is currently an Associate Professor with the Faculty of Applied Sciences, Macao Polytechnic University, Macau, China. From 2004 to 2007, he participated in the European Wireless World Initiative New Radio (WINNER) Project. His research interests include mobile wireless communications, digital signal processing, machine learning in communications, and computer vision in smart cities. He is a member of the IEEE Communications Society.

QINGLE WU received the B.A.Sc. degree in mathematics from Northwest Normal University, in 2010, and the M.A.Sc. degree in cryptography from Southwest Jiaotong University, in 2013. She is currently pursuing the Ph.D. degree in computer applied technology with the Faculty of Applied Sciences, Macao Polytechnic University. Her main research interests include channel neural decoding algorithm and low-complexity technologies in communications.

BENJAMIN K. NG (Senior Member, IEEE) received the B.A.Sc., M.A.Sc., and Ph.D. degrees in engineering science and electrical engineering from the University of Toronto, in 1996, 1998, and 2002, respectively. From 2005 to 2009, he was with Radiospire Networks Inc., Boston, MA, USA, where he was a Senior Communications Engineer, focusing on UWB and millimeter wave technologies. He joined Macao Polytechnic University, Macau, China, in 2010, where he is currently an Associate Professor with the Faculty of Applied Sciences. His research interests include wireless communications and signal processing, with an emphasis on MIMO, NOMA, and machine learning technologies.

SIO-KEI IM (Member, IEEE) received the degree in computer science and the master's degree in enterprise information systems from the King's College, University of London, U.K., in 1998 and 1999, respectively, and the Ph.D. degree in electronic engineering from the Queen Mary University of London (QMUL), U.K., in 2007. He gained the position of a Lecturer with the Computing Program, Macao Polytechnic Institute (MPI), in 2001. In 2005, he became the Operations Manager of MPI-QMUL Information Systems Research Center, jointly operated by MPI and QMUL, where he carried out signal processing work. He was promoted to a Professor with Macao Polytechnic Institute, in 2015. He was a Visiting Scholar with the School of Engineering, University of California at Los Angeles (UCLA), and an Honorary Professor with the Open University of Hong Kong.