IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 9 September 2024, accepted 29 September 2024, date of publication 10 October 2024,
date of current version 24 October 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3477427

APPLIED RESEARCH

On the Design of a Light-Weight Deep Learning Framework for Embedding in 5G Software Modem

WOONGGYU MIN[1], (Student Member, IEEE), SEUNGWOO KANG[1], (Student Member, IEEE), JUYEOP KIM[2], (Member, IEEE), AND OHYUN JO[1], (Member, IEEE)

[1]Department of Computer Science, Chungbuk National University, Cheongju 28644, Republic of Korea
[2]Department of Electronic Engineering, Sookmyung Women's University, Seoul 04310, Republic of Korea

Corresponding authors: Ohyun Jo (ohyunjo@chungbuk.ac.kr) and Juyeop Kim (jykim@sookmyung.ac.kr)

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korea Government [Ministry of Science and ICT (MSIT)] (Development of 5G+ Intelligent Base Station Software Modem) under Grant 2021-0-00165, and in part by the National Research Foundation of Korea (NRF) Grant funded by the Korea Government (MSIT) under Grant 2021R1A2C2095289.

ABSTRACT In this paper, we present a light-weight deep learning framework specifically designed and implemented for embedding in 5G software modems. The framework is developed completely using the C language to operate in real time by being mounted on a software modem. The framework incorporates an imagification process proposed by the authors which can enhance efficient reference signal classification in constrained environments. Imagification is the proposed technique that converts radio signal data, which is sequence data, into image form, processing the data into a structure similar to the RGB color model used in traditional images. This enables the application of convolutional operations to reduce complexity and training time. The performance of the framework is validated using a spec-compliant 5G software modem testbed developed by the authors, achieving up to 99.7% accuracy even at a relatively low SNR of −2.74 dB. These results demonstrate the feasibility of integrating the deep learning framework into a practical 5G software modem. Additionally, we perform hyperparameter optimization to identify the most suitable learning structure for the system. The developed source code is available at Github for public use.

INDEX TERMS 5G, deep learning, software modem, wireless communication.

I. INTRODUCTION

Due to the dynamic nature and complexity of 5G networks, recent research is actively exploring the use of software-defined modems as an alternative to traditional hardware-based solutions [2], [3], [4]. While hardware modems provide efficient communication and performance, they struggle to adapt to evolving standards and incorporate new features, which can hinder network deployment and innovation [5]. Software modems, on the other hand, are more flexible and adaptable, and can be easily updated with new features and performance enhancements through software updates, simplifying network deployment and reducing development time and costs. As 5G networks mature, the shift to software-based solutions is expected to accelerate, bringing agility and cost-effectiveness to the forefront of future communications systems. However, challenges remain with this transition, including processing power demands and potential security vulnerabilities, which will require additional research and development efforts to ensure a smooth transition. This is necessary to ensure that software-defined modems are better able to meet the needs of the network and are reliable in terms of security and performance.

With deep learning's impressive achievements in various fields, there is a growing movement to apply deep learning to

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

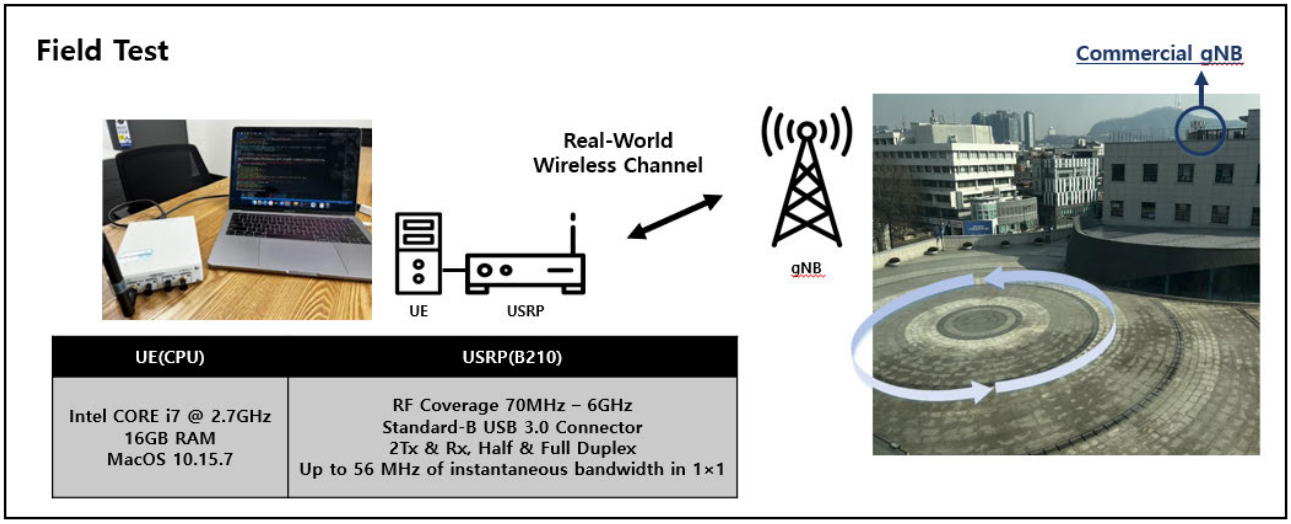

FIGURE 1. Data collection with the software modem.

communication systems such as 5G [6], [7], [8], [9], [10]. One scenario where deep learning technology can be applied in 5G systems is the classification of DMRS (Demodulation Reference Signals). Conventionally, 5G devices must undergo a blind decoding process based on correlation operation to identify the index of the DMRS signal. In the 5G system, DMRS signals can have 8 types of indices and are composed of Zadoff-Chu sequences [11]. Performing blind decoding without any prior information can lead to significant latency in identifying the index of the sequence signal. This is because a thorough investigation must be conducted on all types of sequences. Prior to attempting blind decoding, deep learning can be employed to predict the likely sequence index by learning from the raw data of the received signal in advance. Utilizing datasets collected in real environments, We have previously confirmed through our research that DMRS signal classification using deep learning can be highly accurate [1], [11], [12], [13], [14].

In the previous research, data from real environments was collected and structured into a dataset, which was then trained on a learning server equipped with various Python libraries and high computing power. This approach was undertaken to evaluate whether deep learning technology can indeed improve real-world 5G signal processing. However, in a real 5G system, signal processing needs to happen in real-time. Hence, we still face practical challenges in integrating deep learning technology into real-time 5G systems.

While popular deep learning libraries like Keras provide flexibility and ease of use, they may have limitations in terms of efficiency and processing speed, particularly in resource-constrained embedded environments where high-performance CPU (Central Processing Units) or GPU (Graphics Processing Units) cannot be fully deployed [15]. As a consequence, Python-based Keras models may struggle to meet the real-time demands of practical 5G systems. Moreover, high computational complexity is a major obstacle to applying deep learning to real-time 5G software modems. Therefore, lightweight and efficient deep learning frameworks with low cost and complexity become crucial. Indeed, deep learning frameworks designed for 5G systems must facilitate real-time learning and inference processes, even within resource-constrained environments, ensuring compatibility with existing communication modules and improving interoperability. Beyond aiding resource-constrained systems, such frameworks can offer benefits such as power efficiency and scalability across a broad spectrum of 5G applications.

To address these challenges, this research proposes a novel C-based deep learning framework. The framework aims at efficient ground truth signal classification and uses a lightweight CNN (Convolutional Neural Network) architecture along with image-based techniques to perform comparably to existing Python-based libraries in terms of classification accuracy. To this end, we devise an *imagification* technique that can process 5G sequence signals into an image format, and through this, we are able to lighten the learning model. The deep learning framework is implemented using only C language without any additional libraries, making it adaptable for general use in a variety of system environments. As the result, the main contribution of this paper is that we have implemented a C language-based deep learning framework with Imagification, a lightweight preprocessing technique, for use in practical 5G base station systems, and have shown reliable results using real-world testbeds and measured data for performance evaluation. To ensure public access, we uploaded the source code of the C-based deep learning framework to GitHub. The source code is available through the following link: *https://github.com/CBNU-ISL/C-based-DL-Framework*.

II. SYSTEM MODEL AND ENVIRONMENT

A. INTELLIGENT 5G SOFTWARE MODEM

The mobile communication industry has undergone relentless innovation and development [16], [17], [18], [19]. However, communication modems have still been developed the form of hardware. To achieve fast transmission speeds and facilitate integration into mobile devices, it is more advantageous to develop them as ASICs (Application-Specific Integrated Circuits) that operate for a specific purpose, rather than implementing them in software using a general-purpose CPU. The hardware modem in 5G systems plays a crucial role in enabling data communication between mobile devices and networks [20], [21]. It handles data transmission and reception across various frequency bands, supporting high-speed data transfer and low-latency communication. Functions such as frequency conversion, digital signal processing, error detection and correction, and secure data transmission are integrated into the hardware modem.

Despite its pivotal role in driving innovation, hardware modems also present disadvantages from a development perspective [22]. Firstly, high engineering costs pose a significant drawback. The fabrication process is costly, and the development takes a very long time. Components such as high-frequency transceivers, high-performance processors and memory contribute to the substantial production costs. Furthermore, hardware development is less flexible, posing challenges in responding to design errors. The process of adding or modifying detailed functions to configure a specialized network is intricate. Compatibility issues also exist. In regions or countries where 5G infrastructure is not yet widespread, compatibility problems may arise, causing inconvenience for users who travel between different areas.

To address the challenges, there is growing interest in the development of 5G software modems [4], [8]. Software modems offer flexibility and scalability compared to hardware counterparts. They can easily incorporate new features and performance improvements through software updates, enabling faster adaptation to new standards and requirements. Furthermore, software modems can run on various hardware platforms, resolving compatibility issues and supporting diverse mobile devices and network environments. Additionally, leveraging virtualization technology enables efficient resource management.

To this end, we have developed a real-time 5G software modem based on a universal software radio peripheral (USRP) that is fully compliant with the 5G standard. The modem has been operationally tested in a typical CPU environment and successfully verified its ability to synchronize with commercial 5G base stations and send and receive signals. In addition, we have developed and integrated a learning module that works in conjunction with the communication module to enhance the intelligence of the 5G system. These characteristics make this 5G software modem a promising solution to overcome the limitations of 5G hardware modems, providing improved performance and flexibility. Furthermore, since blind decoding is used in signal processing cases in wireless communications, such as Beam Search, Physical Random Access Channel (PRACH), and Sounding Reference Signal(SRS), this approach can be similarly applied to other wireless communication standards [23], [24], [25].

TABLE 1. Experimental parameters for the test environment.

Parameters	Value
Sampling Rate	30720000
Downlink Freqeuncy	3608.79MHz
nr-arfcn	640586
Bandwidth	20MHz
FFT Size	1024
Subcarrier Spacing	30kHz
SSB Period	20ms
L_max	8

TABLE 2. Details of the collected DMRS data.

	Details
SNR	23.37 dB, 9.56 dB, -2.51 dB, -2.74 dB, -2.81 dB, -2.99 dB, -3.13 dB, -3.42 dB, -3.7 dB, -4.11 dB, -4.5 dB, -4.66 dB
Index	0, 1, 2, 3, 4, 5, 6, 7
Number of data	(SNR 12) x (Index 8) x 20,000 = 1,920,000
Shape of data	1 x 288 (144 per In-Phase, Quadrature-Phase)

B. DETAILS OF TEST ENVIRONMENT & DATASET

The USRP(B210) contains RF modules covering frequencies from 70 MHz to 6 GHz and uses a 2 × 2 MIMO structure. The PC associated with the USRP is a PC with typical specifications, and the details of the device are shown in Figure 1. In addition, the parameters of the frame structure were set based on the cell configurations used by 5G gNBs and are shown in Table 1.

For the validation of the framework, real 5G DMRS data collected in a real wireless channel environment by using the 5G software modem testbeds [14], [26]. The details of the collected 5G DMRS data are presented in Table 2. The DMRS data is composed of 8 types of Zadoff-Chu sequences and was collected in a total of 12 SNR(Signal-to-Noise Ratio) environments, ranging from −4.66 dB to 23.37 dB. The sequence data are indexed from 0 to 7. There are 20,000 data samples for each index, for a total of 1,920,000 data samples. Each data sample consists of 144 real elements and 144 imaginary elements.

C. REAL-TIME LEARNING ALGORITHMS FOR LARGE-SCALE EMBEDDED SYSTEMS

Large-scale embedded systems play an important role in providing intelligent and automated capabilities in

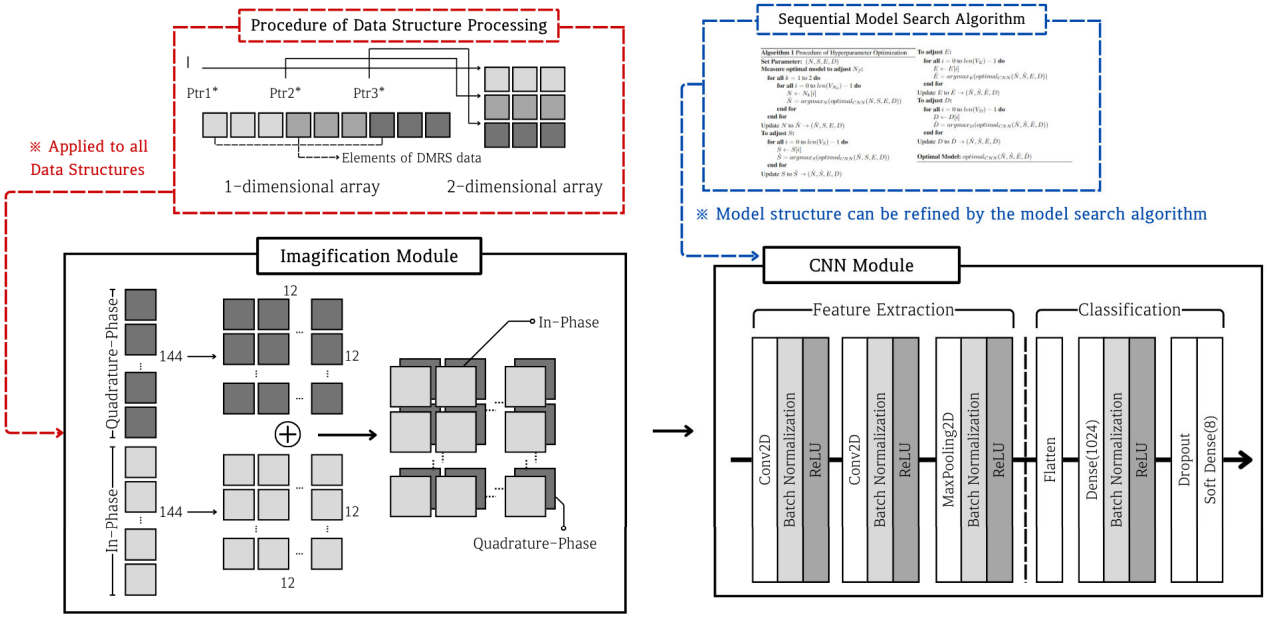

FIGURE 2. Overall architecture of the implemented learning model with Imagification technique.

industry. These systems typically require efficient and rapid decision-making processes because their environments are subject to change and they operate under limited resources and real-time requirements. To operate these systems efficiently, it is essential to respond quickly to changes in the environment and adjust and optimize the system accordingly [5]. As a result, the importance of real-time learning in embedded environments has been emphasized.

Recent research has shown great interest in applying real-time learning algorithms in large-scale embedded systems [27], [28], [29]. Current research focuses on combining deep learning techniques with real-time learning algorithms in such systems. This approach enables the system to quickly adapt to changes in the environment, make quick decisions, and utilize resources efficiently. In addition, real-time learning algorithms continuously process and analyze data to continuously improve system performance. This helps the system adapt and optimize to changing environments over time. The key to real-time learning in embedded environments is lightweight models and efficient arithmetic algorithms. It is also necessary to use the minimum amount of resources required to process and learn from data in real time. To this end, model compression techniques, low-power devices, and efficient algorithms are being researched, and as these technologies are further developed, real-time learning in embedded environments is expected to become more feasible.

To this end, we implemented preprocessing techniques to enable real-time processing of deep learning in embedded environments, simplifying the learning model and implementing the advanced computational capabilities of efficient data structures and frameworks. we devised and implemented a method to optimize data structure and Arithmetical Functions for Real-time Operation through this study and applied it to our Learning Framework. These technical elements reflected in the framework are discussed in detail in the next section.

III. DEEP LEARNING FRAMEWORK FOR 5G SOFTWARE MODEM EMBEDDING

A. IMAGIFICATION: LIGHTWEIGHT LEARNING MODEL

Model lightweighting is essential to ensure the effective operation of deep learning models in diverse environments [30], [31], [32], [33], [34]. Lightweight models are especially necessary in environments with constrained memory and processing capabilities, such as mobile devices and edge devices. Additionally, in the most of real-time applications such as communications and signal processing, making model lightweighting is even more important. Traditional DNN (Deep Neural Network) models which contain fully connected layers exhibit increased computational complexity with an increase in the number of nodes. In contrast, CNN based models adopting convolutional filter operations can be lighter. By using the proposed imagification technique, it is possible to maintain high accuracy of the learning model while reducing computational complexity. This enables the learning framework to operate in general-purpose CPU environments and embedded systems without the need for advanced computational modules like GPU [12], [13].

In this section, we introduce the imagification technique for lightweighting the learning model and describe the structure of the lightweight CNN model used as a classification model [35]. Figure 2 represents the overall process of the learning model using imagification. When Imagification is applied to a complex signal consisting of real and imaginary parts, each phase can be represented as shown in Equation 1.

TABLE 3. Details of CNN module.

layer	Output Shape	Param
Conv2D	(None, 10, 10, 16)	304
Conv2D	(None, 8, 8, 8)	1160
MaxPool2D	(None, 4, 4, 8)	0
Flatten	(None, 128)	0
Dense	(None, 1024)	132096
Dropout	(None, 1024)	0
Dense	(None, 8)	8200

In this formula, i represents each data and n is the length of the sequence. Equation 2 determines the size of the image when converting sequence data into image format. Here, n represents the total number of pixels in the generated image, and N represents the axis length of the image. Consequently, the image generated is in the form of $N \times N$ in this case. However, it is generally extensible to other image formats. Then, Equation 3 is used to transform the sequence data from one-dimensional to two-dimensional matrix form, resembling the channels of the image [12].

$$DMRS\_RG_i[n] = \begin{cases} R_i[n] = I_i[n] \\ G_i[n] = Q_i[n], \end{cases} \quad n \in [0,\ Len(Seq)] \tag{1}$$

$$N = ceil(\sqrt{max(n)}) \tag{2}$$

$$DMRS_{1D-array}Seq[n] \rightarrow DMRS_{2D-array}Seq[n/N][n\%N] \tag{3}$$

Classification Module in Figure 2. illustrates the structure of the learning model implemented through the proposed imagification process, consisting of a feature extraction component and a classification component. Details about the structure of the CNN Module can be found in Table 3. The imagified 5G DMRS data is inputted into the feature extraction component of the model. The feature extraction component comprises convolutional layers and pooling layers, with the structure varying based on the values of the hyperparameters optimized in the subsequent section's experiments.

In deep learning models, the tensor data type is commonly used. This is because tensors effectively handle multidimensional arrays and are optimized for high-speed processing of large-scale operations during model training and inference [36]. The procedure of data structuring Module in Figure 2 illustrates the data structure used in the proposed framework to implement tensor data types. To enable flexible transformation and processing of sampled DMRS data, multi-dimensional pointers are added to arbitrary positions in the array, allowing direct access to indices. This design enhances the speed of operations during shape transformation of the array.

B. STRUCTURE OF C FRAMEWORK USING EFFICIENT ARITHMETIC FUNCTIONS

The proposed C-based deep learning framework is specifically designed and implemented for real-time operation within 5G base station software modems. The learning model consists of several layers, each of which serves as a forward propagation and a background propagation that calculates the error for the true answer using the forward propagation and loss functions that perform the forward operation. Each layer receives the output of the previous layer and a derivative for the next layer. In the framework, a linked-list is used as a data structure for implementing such a deep learning structure.

In real-time learning, shortening the training time of the model is essential. The proposed framework focuses on implementing a lightweight model using image-based techniques to achieve training time reduction through various algorithms. In order to process the main operations of the CNN model, the Im2Col algorithm that converts images into matrices and its inverse, the Col2Im algorithm, were implemented to improve the efficiency. The He/Xavier initialization algorithm was used to initialize the weight of the model [37], [38].

The Im2Col and Col2Im algorithms are used to perform convolutional operations in CNNs, a technique that can significantly reduce training time through a trade-off between spatial and temporal complexity. The convolution operation itself is a relatively simple process of moving filters and performing dot multiplication, but because the images used for training are multidimensional, the number of iterations increases proportionally to the number of dimensions. This leads to an increase in code complexity and time complexity, which we address by converting the multidimensional image to a two-dimensional matrix, effectively reducing the time complexity by performing the convolution operation in a single matrix multiplication.

Figure 3 illustrates how the Im2Col algorithm works, showing that a convolutional operation on an image of size $(m \times m \times C)$ and a filter of size $(n \times n \times C)$ requires nine operations, where m and n are values for the height and width of the imagified data and convolutional filter, and C is the number of dimensions of the imagified data and filter. The data is converted into a two-dimensional arrangement equal to the number divided by the filter and the size of the filter. At the same time, the filter is converted into a two-dimensional arrangement equal to the size of the filter and the number of filters. The image data and filters converted in this way are then processed through the Dot operation. Applying this algorithm increases the space used for training, but reduces the time it takes to train by more than 200 times.

In addition, in deep learning training, the initial weight setting also has a huge impact on the learning time. If the initial weights are set well, the slope of the loss function is calculated appropriately, which makes the value of the loss function fluctuate stably during the learning process, making it easier to control the learning rate and enabling the model to learn more stably. Therefore, we applied the

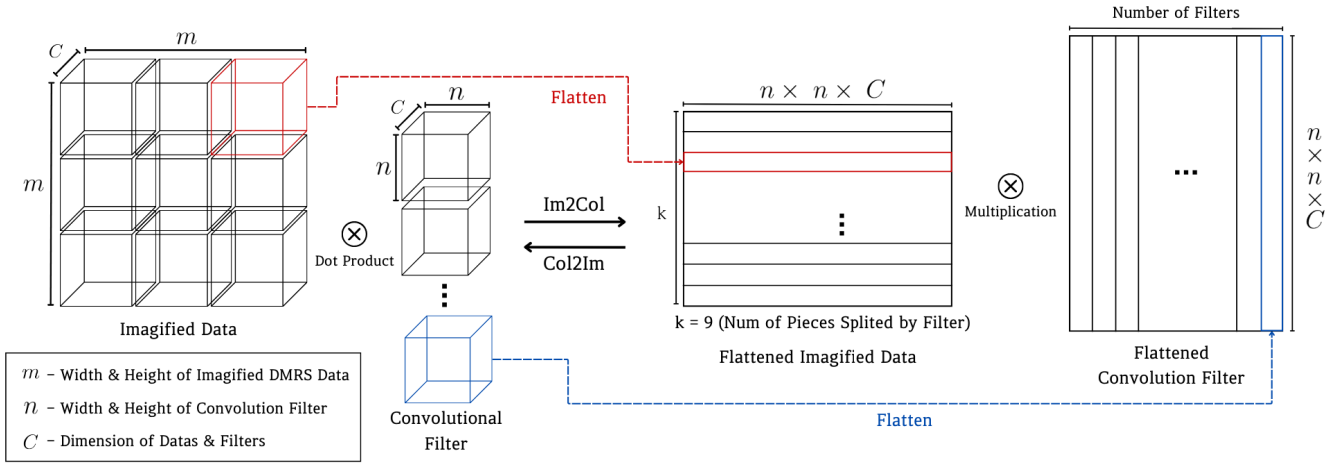

FIGURE 3. A visual representation of the Im2Col and Col2Im algorithms.

Xavier Initialization method [38], which is effective for nonlinear functions, and the He Initialization method [37], which is effective for ReLU activation functions, to shorten the learning time.

Xavier initialization [38] sets weights that take into account the number of inputs and outputs, ensuring that the signals between layers are properly distributed. It is often used with nonlinear activation functions such as Sigmoid or Tanh. On the other hand, He initialization [37] is optimized for ReLU and its variants activation functions, setting the weights proportional to the square root of the inverse of the number of input neurons to avoid problems with gradient vanishing or explosion. These two initialization methods allowed the model to learn faster and more reliably.

IV. EXPERIMENTS & RESULTS

The optimization experiment consists of two parts: an optimization part to analyze the performance of different hyperparameters related to convolutional operations, and a learning rate optimization to find the optimal learning rate. The metrics used to evaluate the performance of the model are divided into precision and average precision, and the formulas for each metric are shown in Equation 4, 5.

$$\tau_j = \frac{\sum_{i=1}^{n_j} I(y_{i,j} = \hat{y}_{i,j})}{n_j} \tag{4}$$

$$\bar{\tau} = \frac{1}{k}\sum_{j=1}^{k} \tau_j \tag{5}$$

The $\bar{\tau}$ is the average of the individual accuracy(τ) obtained from k SNR levels divided by k, which can be expressed as Equation 5. Also, the τ that can be obtained from individual SNR levels can be expressed as Equation 4. In this expression, n_j is the total number of samples in j_{th} SNR level, $y_{i,j}$ is the actual label of the i_{th} sample in j_{th} SNR level, and $\hat{y}_{i,j}$ is the predicted label. The $I(\cdot)$ function is an indicative function that returns 1 if the condition is true and 0 if it is false.

TABLE 4. Details of the Hyperparameters.

HyperParameter	Number of Conv2D Layer(k)	Details(V)
Number of Filter (N)	k=1	(4, None), (8, None), (16, None)
	k=2	(4, 4), (4, 8), (8, 8), (8, 16), (16, 16)
Size of Filter (S)	Decided by previous experiment	(2x2), (3x3), (4x4)
Existence of MaxPooling (E)		w, w/o
Shape of Imagified Data (D)		(6x24), (8x18), (9x16), (12x12), (16x9), (18x8), (24x6)

The hyperparameters for training the model are set as follows: 25 epochs were used for training, and MultiClass CrossEntropy was used as the loss function. Adam Optimizer was selected as the optimization technique, and the batch size was set to 256. The dataset was divided into 7:2:1 for train, validation, and test.

A. CNN HYPERPARAMETER OPTIMIZATION

1) DETAILS OF CNN HYPERPARAMETERS

Table 4 provides a list of hyperparameters used in the experiments and the detailed description of each hyperparameter. These hyperparameters affect the feature extraction process in the convolutional layer and were chosen considering the experimental environment and computational time. The optimization experiments follow the order in the table, and the highest classification accuracy obtained in each experiment is fixed as the hyperparameter value for the next experiment.

2) RESULTS OF CNN HYPERPARAMETER OPTIMIZATION

The experiments were performed on the index classification problem of DMRS, and the classification accuracy defined in the previous subsection was used as an evaluation metric for the SNR-specific hyperparameter experiments. In addition,

TABLE 5. Classification accuracy and average classification accuracy by convolutional layer and number of filters.

Num of Conv2D Layer(k)	Num of Filter(m, n)	SNR(dB)												Average Accuracy
		23.37	9.56	-2.51	-2.74	-2.81	-2.99	-3.13	-3.42	-3.7	-4.11	-4.5	-4.66	
k=1	(4, None)	**1**	**1**	0.984	**0.997**	**0.996**	**0.992**	**0.988**	0.974	0.934	**0.814**	0.330	0.190	0.850
	(8, None)	**1**	**1**	**0.985**	0.994	**0.996**	**0.992**	**0.988**	**0.977**	**0.939**	0.809	0.338	0.190	0.851
	(16, None)	**1**	**1**	**0.985**	0.995	0.995	0.991	**0.988**	0.975	0.938	0.811	**0.343**	**0.205**	**0.852**
k=2	(4, 4)	**1**	0.999	0.969	0.970	0.985	0.980	0.966	0.944	0.871	0.649	0.267	0.169	0.814
	(8, 8)	**1**	**1**	0.979	0.993	0.989	0.982	0.976	0.959	0.913	0.713	0.307	0.187	0.833
	(16, 16)	**1**	**1**	0.982	0.992	0.993	0.984	0.983	0.968	0.911	0.763	0.312	0.187	0.840
	(4, 8)	0.942	**1**	0.971	0.986	0.984	0.977	0.971	0945	0.862	0.708	0.273	0.173	0.816
	(8, 16)	**1**	**1**	0.976	0.990	0.990	0.985	0.979	0.952	0.909	0.757	0.308	0.184	0.836

TABLE 6. Classification accuracy and average classification accuracy by filter size, with or with out of pooling layer, and the size of the imagified data.

Type of Parameter		SNR(dB)												Average Accuracy
		23.37	9.56	-2.51	-2.74	-2.81	-2.99	-3.13	-3.42	-3.7	-4.11	-4.5	-4.66	
Size of Filter	2x2	**1**	**1**	0.985	**0.995**	**0.996**	**0.992**	**0.988**	0.974	0.936	**0.810**	**0.347**	0.200	**0.852**
	3x3	**1**	**1**	**0.986**	**0.995**	**0.996**	0.990	0.987	0.973	0.935	0.806	0.336	0.203	0.851
	4x4	**1**	**1**	**0.986**	**0.995**	**0.996**	0.991	0.987	**0.975**	**0.937**	0.806	0.339	**0.204**	0.851
Existence of Pooling layer	w	**1**	**1**	0.964	0.984	0.996	0.967	0.965	0.937	0.893	0.734	0.251	0.154	0.818
	w/o	**1**	**1**	**0.985**	**0.995**	**0.995**	**0.991**	**0.988**	**0.975**	**0.938**	**0.811**	**0.343**	**0.205**	**0.852**
Shape of Imagified Data	6x24	**1**	**1**	0.985	0.995	**0.996**	**0.992**	0.987	**0.979**	0.934	0.805	0.344	0.201	0.852
	8x18	**1**	**1**	0.984	0.995	**0.996**	**0.992**	**0.988**	0.975	0.936	0.811	0.340	0.198	0.851
	9x16	0.991	**1**	0.985	0.995	0.995	**0.992**	**0.988**	0.975	0.938	0.811	0.344	0.202	0.851
	12x12	**1**	**1**	0.985	0.995	0.995	0.991	**0.988**	0.975	0.938	0.811	0.343	**0.205**	0.852
	16x9	**1**	**1**	0.984	0.995	0.995	**0.992**	0.987	0.977	0.936	0.811	0.341	0.201	0.852
	18x8	**1**	**1**	**0.987**	**0.996**	**0.996**	**0.992**	0.987	0.977	**0.940**	**0.812**	**0.349**	0.202	**0.853**
	24x6	**1**	**1**	0.986	0.995	**0.996**	0.991	0.987	0.977	0.938	0.810	0.344	0.201	0.852

the average classification accuracy was used to select the optimal hyperparameters to consider the model accuracy in different SNR levels.

Table 5 shows the experimental results as a function of the number of convolutional layers and filters. The results show that when k = 1 and 4 filters, the model achieves up to 99.7% classification accuracy at −2.74 dB. When k = 1 and the number of filters is 4 and 8, the model shows good classification accuracy in the range of −2.74 dB to −4.11 dB, and at the lowest SNR, 16 filters shows 0.015 better classification accuracy compared to other parameters. Also, when looking at the average classification accuracy, 16 filters has the best average classification accuracy, albeit slightly.

It is notable that having one convolutional layer gives better classification accuracy at all SNR levels compared to two convolutional layers. This is likely a result of the nature of the data. Since the shape of the data is very small compared to a typical image, as the number of convolutional layers increases, the features of the data that are learned are lost. As a result, the best average classification accuracy was obtained with 1 convolutional layer and 16 filters, so the feature map parameters are fixed to that value in the following experiments.

Table 6 shows the results of the hyperparameter optimization experiments without the number of filters. The results as a function of filter size show that the parameter value 2 × 2 performs the best at most SNR levels, with an average classification accuracy of 0.852. For most SNR levels, the difference is around 0.001 or less, but for the relatively low SNR levels of −4.11 dB and −4.5 dB, the classification accuracy with the 2 × 2 filter size shows a relatively high difference of more than 0.004 compared to the results with other filter sizes. These results show that the 2 × 2 filter

Algorithm 1 Procedure of Hyperparameter Optimization

Set Parameter: (N, S, E, D)
Measure optimal model to adjust N:
  for all $k = 1$ to 2 **do**
    for all $i = 0$ to $len(V_{N_k}) - 1$ **do**
      $N \leftarrow N_k[i]$
      $\hat{N} = argmax_N(optimal_{CNN}(N, S, E, D))$
    end for
  end for
Update N to $\hat{N} \rightarrow (\hat{N}, S, E, D)$
To adjust S:
  for all $i = 0$ to $len(V_S) - 1$ **do**
    $S \leftarrow S[i]$
    $\hat{S} = argmax_S(optimal_{CNN}(\hat{N}, S, E, D))$
  end for
Update S to $\hat{S} \rightarrow (\hat{N}, \hat{S}, E, D)$
To adjust E:
  for all $i = 0$ to $len(V_E) - 1$ **do**
    $E \leftarrow E[i]$
    $\hat{E} = argmax_E(optimal_{CNN}(\hat{N}, \hat{S}, E, D))$
  end for
Update E to $\hat{E} \rightarrow (\hat{N}, \hat{S}, \hat{E}, D)$
To adjust D:
  for all $i = 0$ to $len(V_D) - 1$ **do**
    $D \leftarrow D[i]$
    $\hat{D} = argmax_D(optimal_{CNN}(\hat{N}, \hat{S}, \hat{E}, D))$
  end for
Update D to $\hat{D} \rightarrow (\hat{N}, \hat{S}, \hat{E}, \hat{D})$

Optimal Model: $optimal_{CNN}(\hat{N}, \hat{S}, \hat{E}, \hat{D})$

size is robust at low SNR levels and can achieve optimal classification accuracy.

The result according to the filter size shows that the parameter value 2×2 shows the best classification accuracy in most SNR levels and the best performance with the average classification accuracy of 0.852. However, it can also be seen that the average classification accuracy by filter size has a value of 0.001 or less. From this result, we can see that the average classification accuracy tends to decrease as the size of the filter increases, but the difference in accuracy is very small, suggesting that the size of the filter does not significantly affect the classification accuracy in this experiment.

Next, the results from the existence of the MaxPooling layer show that the classification accuracy is higher in the case of no pooling layer in all SNR levels, and the average accuracy is 0.034 higher. Since this experiment converts signal data into images for classification, the input data is small compared to typical images. In addition, the pooling layer can reduce the parameters and computation of the network by gradually reducing the spatial size of the representation, but there is a loss of information equal to the reduced spatial size. As a result, this happens because the data

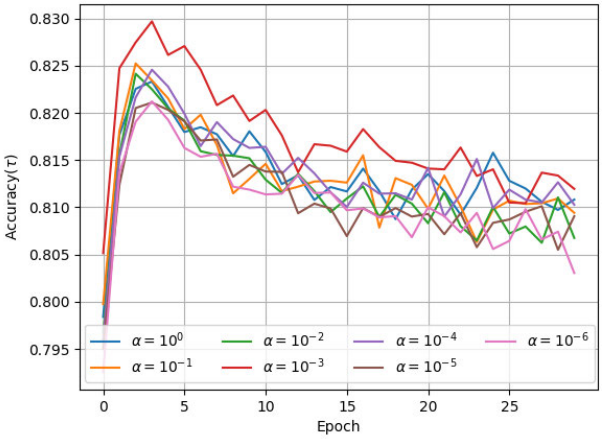

FIGURE 4. Comparison of convergence rates according to Learning Rate values in −4.11 dB.

that is reduced by the imaging technique passes through the pooling layer and loses information.

The results based on the image shape show that the best classification accuracy is obtained for the 18×8 shape. At most SNR levels, the shape of 18×8 performs well, and at -3.7 dB to -4.5 dB, it shows a classification accuracy improvement of up to 0.009 or more compared to other values. However, the difference in the average classification accuracy in the experiment is insignificant at 0.002, indicating that the filter size parameter has a small impact on the experiment. The optimal CNN model determined as a result is a model with one convolutional layer and 16 filters, a filter size of 2×2, and no pooling layer with an imaging size of 18×8.

B. LEARNING RATE REFINEMENT

1) DETAILS OF LEARNING RATE

Various refinement experiments were conducted to find the suitable learning rate as additional experiments. The learning rate is also an important hyperparameter that determines how much the weight values are updated when a deep learning model is trained. Setting an appropriate learning rate helps the model converge efficiently in the right direction. Different learning rate values represent a trade-off between learning speed and stability. For example, selecting a high learning rate can lead to increased weight updates, making convergence unstable. In this case, the model may not find the optimal solution and diverge or cause excessive vibration. Conversely, selecting a low learning rate could slow down the update rate, consequently delay the learning process, and risk taking more time than necessary.

Therefore, it is very important to find a suitable learning rate value for the learning model. In this experiment, various values from 10^{-6} to 10^{0} were used to systematically evaluate the effect of learning rates. By searching for the learning rate within this range, we aim to find a point where the model can

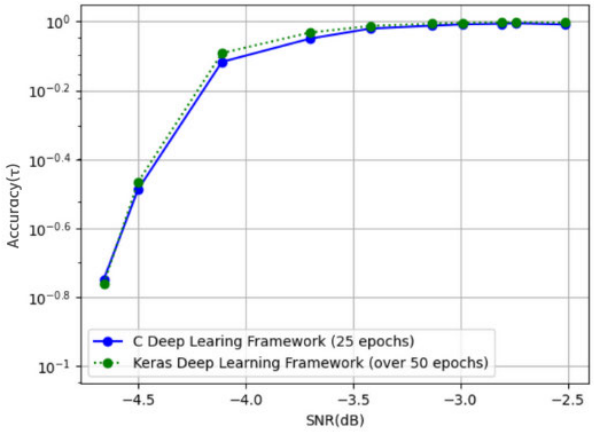

FIGURE 5. Comparison graph of classification accuracy for DL models across different frameworks.

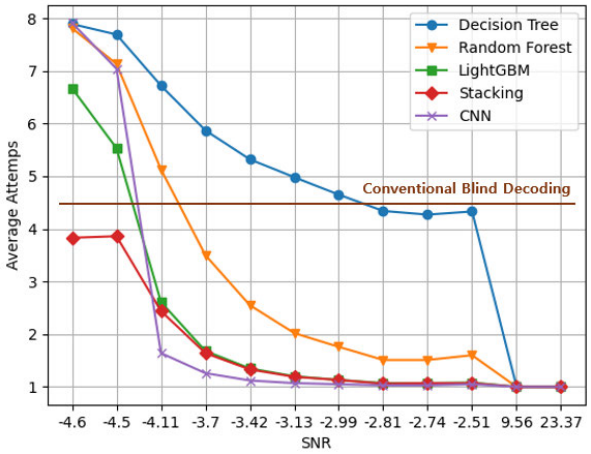

FIGURE 6. Comparison of the number of average attemps across different learning algorithms.

maintain stable performance while converging quickly in the following subsection.

2) RESULTS OF LEARNING RATE OPTIMIZATION

Figure 4 compares convergence speeds and classification accuracy at −4.11 dB for different learning rates. While all values generally converge within 3 to 4 epochs, the learning rate of 10^{-3} achieves a similar classification accuracy to the other values by the first epoch. Notably, the learning rate of 10^{-3} demonstrates the fastest initial convergence and the highest performance, with classification accuracy steadily increasing with each epoch and ultimately reaching a peak accuracy of 0.83. Based on these results, we have determined that a learning rate of 10^{-3} is the optimal value for our learning model. This optimal learning rate allows for both rapid training and high classification accuracy, thereby significantly enhancing the real-time processing capabilities of embedded systems.

C. COMPARISON OF DEEP LEARNING MODEL PERFORMANCE ACROSS FRAMEWORKS

We compare our deep learning framework with state-of-the-art models that utilize public libraries such as Keras to provide user-friendly features. The Keras model used in the comparison provides may have many useful features for easy algorithm validation, but requires high computational resources and significant computing power. Figure 5 shows the classification accuracy achieved by the two types of deep learning models.

Although the accuracy of the proposed framework is slightly lower than the Keras-based learning model in the range of −4.5 dB to −3.0 dB, the average difference is less than about 0.05, indicating very small performance gap, which means that the two frameworks maintain almost the same level of prediction accuracy. Moreover, the proposed framework evaluated its performance using only 25 epochs, half of the epochs used by the Keras-based learning model, which demonstrates that reliable classification is possible with less training time and computational resources. These results show that the proposed framework can work effectively even in limited resource environments. Specifically, the proposed framework can perform almost as well as a Keras-based learning model, while operating with less training data and computational cost. This shows that the proposed framework can be an efficient and practical choice.

Consequently, we have demonstrated that the proposed framework exhibits a slight performance degradation compared to Keras. However, this difference is minimal, indicating that our framework can work reliably and efficiently even in situations with limited training resources and time. This clearly shows that the proposed framework can provide a high level of classification performance even under high system constraints.

We can also observe that in most SNR environments, the index classification is possible using fewer trials for decoding compared to the Conventional Blind Decoding method. Figure 6 shows the average number of trials required to decode the DMRS index for different ML-based methods and the conventional blind decoding method in various SNR environments. It can be seen that from −23.37 dB to −4.11 dB, all ML methods have a lower average number of trials compared to the conventional blind decoding method. At −4.11dB, it can be shown that classification with CNN has the lowest average number of trials compared to the other ML learning models. These results show that preprocessing with Imagification and the use of CNN models can effectively learn complex features and perform more accurate index classification based on them. In particular, the fact that the method using CNN outperforms other ML models even in low SNR environments such as −4.11 dB shows that the proposed framework can provide reliable decoding performance in various communication environments.

V. CONCLUSION

In this paper, we developed a deep learning framework that applies a lightweight preprocessing imagification technique

to the blind decoding process of a real-time 5G system. For performance evaluation, we built a real 5G wireless channel environment using USRP(B210) and commercial gNBs, and validated the deep learning framework using data collected from the built environment. The framework was developed using the complete C language to ensure that the learning model is sufficiently capable of running in a general system environment, and the applied imaging technique showed that it can extract spatial features even from sequence data converted into images. Finally, hyperparameter optimization experiments were performed to derive the optimal model structure. In conclusion, the construction and experimentation of a real system environment confirmed the feasibility of utilizing deep learning techniques in real 5G systems and other applications, opening up new possibilities. However, the current system has the disadvantage of poor parallel processing capability in a CPU-only environment, which leads to long learning time. To compensate for this, further development aims to develop a model that can be trained and processed in real time on top of the system with reduced learning time.

ACKNOWLEDGMENT

An earlier version of this paper was presented in part at the AAAI-24 Conference DOI: 10.1609/aaai.v38i21.30461.

REFERENCES

[1] S. W. Kang and O. Jo, "Multivariate time-series imagification with time embedding in constrained environments (student abstract)," in *Proc. AAAI Conf. Artif. Intell.*, 2024, vol. 38, no. 21, pp. 23535–23536.

[2] K. Enhos, D. Unal, E. Demirors, and T. Melodia, "Software-defined distributed SIMO system for underwater communication," in *Proc. IEEE Int. Conf. Commun.*, vol. 33, May 2023, pp. 2179–2184.

[3] M. Jeong, S. W. Choi, and J. Kim, "On the design of an OFDMA-based software-defined modem: A CFO estimation perspective," *ICT Exp.*, vol. 10, pp. 184–193, Feb. 2023.

[4] C. Luo, B. Ma, F. Chen, Q. Guan, H. Yu, and F. Ji, "Multicore-processor based software-defined communication/network platform for underwater Internet of Things," *Sensors*, vol. 19, no. 23, p. 5168, Nov. 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/23/5168

[5] E. Batzolis, E. Vrochidou, and G. A. Papakostas, "Machine learning in embedded systems: Limitations, solutions and future challenges," in *Proc. IEEE 13th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Mar. 2023, pp. 345–350.

[6] H. Huang, S. Guo, G. Gui, Z. Yang, J. Zhang, H. Sari, and F. Adachi, "Deep learning for physical-layer 5G wireless techniques: Opportunities, challenges and solutions," *IEEE Wireless Commun.*, vol. 27, no. 1, pp. 214–222, Feb. 2020.

[7] W. Kim, Y. Ahn, J. Kim, and B. Shim, "Towards deep learning-aided wireless channel estimation and channel state information feedback for 6G," *J. Commun. Netw.*, vol. 25, no. 1, pp. 61–75, Feb. 2023.

[8] B. Casasole, L. Bonati, S. D'Oro, S. Basagni, A. Capone, and T. Melodia, "QCell: Self-optimization of softwarized 5G networks through deep Q-learning," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2021, pp. 1–6.

[9] C. Lv and Z. Luo, "Deep learning for channel estimation in physical layer wireless communications: Fundamental, methods, and challenges," *Electronics*, vol. 12, no. 24, p. 4965, Dec. 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/24/4965

[10] H. N. Fakhouri, S. Alawadi, F. M. Awaysheh, I. B. Hani, M. Alkhalaileh, and F. Hamad, "A comprehensive study on the role of machine learning in 5G security: Challenges, technologies, and solutions," *Electronics*, vol. 12, no. 22, p. 4604, Nov. 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/22/4604

[11] S. Kang, T. Lee, J. Kim, and O. Jo, "Performance improvement using 5G DMRS data pre-processing for deep learning based index classification model," in *Proc. Symp. Korean Inst. Commun. Inf. Sci.*, 2022, pp. 175–176.

[12] J. Kim, S. Kang, and O. Jo, "Lightweight data processing scheme based on machine learning for 5G DMRS index classification," *Asia–Pacific J. Convergent Res. Interchange*, vol. 9, no. 11, pp. 91–101, Nov. 2023. [Online]. Available: http://apjcriweb.org/content/vol9no11/9.html

[13] J. Kim, S. Kang, J. Kim, and O. Jo, "Enhancing performance for 5G DMRS signals classification usingmulti-channel based imagification," in *Proc. Korean Inst. Commun. Sci. Conf.*, 2023, pp. 1541–1542.

[14] S. Kang, T. Lee, J. Kim, A.-R.-S. Lee, J. Kim, and O. Jo, "Intelligent index classification method based on machine learning for detection of reference signal in 5G networks," *IEEE Access*, vol. 11, pp. 100810–100822, 2023.

[15] R. Conlin, K. Erickson, J. Abbate, and E. Kolemen, "Keras2c: A library for converting Keras neural networks to real-time compatible C," *Eng. Appl. Artif. Intell.*, vol. 100, Apr. 2021, Art. no. 104182.

[16] G. Liu and D. Jiang, "5G: Vision and requirements for mobile communication system towards year 2020," *Chin. J. Eng.*, vol. 2016, pp. 1–8, Apr. 2016.

[17] A. Kumar and M. Gupta, "A review on activities of fifth generation mobile communication system," *Alexandria Eng. J.*, vol. 57, no. 2, pp. 1125–1135, Jun. 2018.

[18] S. Chen and J. Zhao, "The requirements, challenges, and technologies for 5G of terrestrial mobile telecommunication," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 36–43, May 2014.

[19] H. Taha, P. Vári, and S. Nagy, "Adjacent-channel compatibility analysis of international mobile telecommunications downlink and digital terrestrial television broadcasting reception in the 470–694 MHz frequency band using Monte Carlo simulation," *Electronics*, vol. 13, no. 3, p. 575, Jan. 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/3/575

[20] J.-W. Kim, Y.-J. Bang, Y.-O. Park, I. Kim, and T. J. Kim, "Physical layer modem implementation for mmWave 5G mobile communication," *J. Korean Inst. Commun. Inf. Sci.*, vol. 41, no. 1, pp. 51–57, Jan. 2016.

[21] F. Balteanu, "RF front end module architectures for 5G," in *Proc. IEEE BiCMOS Compound Semiconductor Integr. Circuits Technol. Symp. (BCICTS)*, Nov. 2019, pp. 1–8.

[22] Y. Chen, J.-C. Guey, C. Hwang, P. Liao, G. Sébire, W. Wu, and W. Yang, "5G modem design challenges," in *Proc. 5G Technol., 3GPP Evol. 5G-Adv.*, 2024, pp. 399–429.

[23] D. Li, X. Yang, A. Hu, F. Zhou, and O. A. Dobre, "LTE device radio frequency fingerprints blind extraction based on temporal-frequency domain PRACH signals," *IEEE Trans. Veh. Technol.*, vol. 72, no. 10, pp. 13229–13242, May 2023.

[24] L. Shen, Y.-D. Yao, H. Wang, and H. Wang, "Blind decoding based on independent component analysis for a massive MIMO uplink system in microcell Rician/Rayleigh fading channels," *IEEE Trans. Veh. Technol.*, vol. 65, no. 10, pp. 8322–8330, Oct. 2016.

[25] 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.

[26] J. Y. Han, O. Jo, and J. Kim, "Exploitation of channel-learning for enhancing 5G blind beam index detection," *IEEE Trans. Veh. Technol.*, vol. 71, no. 3, pp. 2925–2938, Mar. 2022.

[27] J. Park, P. Aryal, S. R. Mandumula, and R. P. Asolkar, "An optimized DNN model for real-time inferencing on an embedded device," *Sensors*, vol. 23, no. 8, p. 3992, Apr. 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/8/3992

[28] Z. Chen, J. Chen, G. Ding, and H. Huang, "A lightweight CNN-based algorithm and implementation on embedded system for real-time face recognition," *Multimedia Syst.*, vol. 29, no. 1, pp. 129–138, Feb. 2023.

[29] M. Lopez-Montiel, U. Orozco-Rosas, M. Sánchez-Adame, K. Picos, and O. H. M. Ross, "Evaluation method of deep learning-based embedded systems for traffic sign detection," *IEEE Access*, vol. 9, pp. 101217–101238, 2021.

[30] Y. Chen, B. Zheng, Z. Zhang, Q. Wang, C. Shen, and Q. Zhang, "Deep learning on mobile and embedded devices: State-of-the-art, challenges, and future directions," *ACM Comput. Surveys*, vol. 53, no. 4, pp. 1–37, Jul. 2021.

[31] Z. Wang, J. Zhang, Z. Zhao, and F. Su, "Efficient YOLO: A lightweight model for embedded deep learning object detection," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops (ICMEW)*, Jul. 2020, pp. 1–6.

[32] H. W. Lee and S. W. Choi, "A light-weighted machine learning approach to channel estimation for new-radio systems," *Electronics*, vol. 12, no. 23, p. 4740, Nov. 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/23/4740
[33] B. Xing, W. Wang, J. Qian, C. Pan, and Q. Le, "A lightweight model for real-time monitoring of ships," *Electronics*, vol. 12, no. 18, p. 3804, Sep. 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/18/3804
[34] S. Boubaker, F. S. Alsubaei, Y. Said, and H. E. Ahmed, "Lightweight cryptography for connected vehicles communication security on edge devices," *Electronics*, vol. 12, no. 19, p. 4090, Sep. 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/19/4090
[35] A. P. Hermawan, R. R. Ginanjar, D.-S. Kim, and J.-M. Lee, "CNN-based automatic modulation classification for beyond 5G communications," *IEEE Commun. Lett.*, vol. 24, no. 5, pp. 1038–1041, May 2020.
[36] Y. Panagakis, J. Kossaifi, G. G. Chrysos, J. Oldfield, M. A. Nicolaou, A. Anandkumar, and S. Zafeiriou, "Tensor methods in computer vision and deep learning," *Proc. IEEE*, vol. 109, no. 5, pp. 863–890, May 2021.
[37] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.
[38] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.

WOONGGYU MIN (Student Member, IEEE) received the B.S. degree from the Department of Computer Science, Chungbuk National University, Chungju, South Korea, in 2024, where he is currently pursuing the M.S. degree with the Department of Computer Science. He is also with the Information System Laboratory, Chungbuk National University. His current research interests include deep learning and future wireless communication systems.

SEUNGWOO KANG (Student Member, IEEE) received the B.S. and M.S. degrees in computer science from Chungbuk National University, Cheongju, South Korea, in 2021 and 2023, respectively. His current research interests include time series prediction and deep learning-based communication systems.

JUYEOP KIM (Member, IEEE) received the B.S. and Ph.D. degrees in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2004 and 2010, respectively. From 2010 to 2011, he was with the Institute IT Convergence Research Center, KAIST, in charge of research for 5G cellular system. From 2011 to 2013, he was with Samsung Electronics, in charge of development and commercialization for 2G/3G/4G multimode mobile modem solution. From 2014 to 2018, he was with Korea Railroad Research Institute, in charge of research and development for LTE-Railway (LTE-R), public safety LTE (PS-LTE), and railway IoT solutions. He is currently an Assistant Professor with the Department of Electronics Engineering, Sookmyung Women's University, Seoul, South Korea. His current research interests include the applied wireless communications, including mission critical communications, the Internet of Things, and software-defined modems.

OHYUN JO (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering from Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2005, 2007, and 2011, respectively. From April 2011 to February 2016, he was with Samsung Electronics, in charge of research and development for future wireless communication systems, applications, and services. From March 2016 to July 2017, he was a Senior Researcher with the Electronics and Telecommunications Research Institute. From August 2017 to February 2018, he was an Assistant Professor with the Department of Electrical Engineering, Korea Military Academy. He is currently an Associate Professor with the Department of Computer Science, Chungbuk National University, Cheongju, South Korea. He has authored or co-authored more than 90 papers and holds more than 170 registered and filed patents. His research interests include machine learning applications, next generation mobile communication systems, military communications, the Internet of Things, future wireless solutions/applications/services, and embedded communications ASIC design. During his appointment with Samsung, he was a recipient of several recognitions, including the Gold Prize in Samsung Annual Award, the Most Creative Researcher of the Year Award, the Best Mentoring Award, the Major Research and Development Achievement Award, and the Best Improvement of Organization Culture Award.