Received 28 August 2023, accepted 4 September 2023, date of publication 11 September 2023,
date of current version 20 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3314167

RESEARCH ARTICLE

Intelligent Index Classification Method Based on Machine Learning for Detection of Reference Signal in 5G Networks

SEUNGWOO KANG[1], (Student Member, IEEE), TAEGYEOM LEE[1], (Student Member, IEEE), JONGSEOK KIM[1], (Student Member, IEEE), A-REUM-SAEM LEE[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, South Korea
[2]Department of Electronics Engineering, Sookmyung Women's University, Seoul 04310, South 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 & Communications Technology Planning & Evaluation (IITP) funded by the Korean Government [Ministry of Science and ICT (MSIT)] through the Development of 5G+ Intelligent Basestation Software Modem under Grant 2021-0-00165, and in part by the National Research Foundation of Korea (NRF) funded by the Korean Government (MSIT) under Grant 2021R1A2C2095289.

ABSTRACT In order to maintain stable communication in 5G wireless networks, the link between a 5G base station and user equipment (UE) should be constantly monitored and adapted to the time-varying wireless channel. The use of UE for seamless information exchange is based on obtaining a target reference signal. The method used to obtain the reference signal involves identifying the index of the reference signal received from the 5G base stations. However, the existing index identification method employed in commercial 5G networks is based on the blind detection method, which is inefficient in terms of time and can cause misdetections. On the other hand, machine learning (ML), which is statistically predictable through data accumulation, can be robust in practical network environments. Taking this into account, we build a dataset consisting of reference signal data collected in a real-world 5G network environment to obtain an optimal machine learning model that is applicable to practical 5G networks. We evaluate a total of 23 index classification models by combining six ML models and three data pre-processing methods. The results of the study represent optimized combinations of ML-based index classifiers and data pre-processing methods. Performance differences between neural network (NN) models and non-NN models are also revealed.

INDEX TERMS Demodulation reference signal, machine learning, neural network, non-neural network, pre-processing.

I. INTRODUCTION

In commercial application services based on 5G wireless networks, ensuring accurate information delivery between senders and receivers is an important challenge due to the weakness of the relatively-high-frequency band. The stability of the signal transmission may be greatly influenced by dynamic changes in communication environments. To overcome this problem, 5G and other mobile networks use reference signals for synchronization, channel measurement, adaptation, and other important functionalities. In the conventional index classification method, user equipment (UE) uses a specific correlation formula to identify the target signal. The UE can obtain similarities between the original demodulation reference signal (DMRS) sequence and the candidate DMRS sequence by calculating the correlation. However, the conventional correlation method is performed based on blind detection, and many calculations are required to check all the indexes of the reference signals and obtain the most similar signal index from the set of various reference signals. Therefore, the conventional index classification approach using the blind detection method may be inefficient

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

in terms of computational power, and misdetections may occur when the original sequence is damaged.

Machine learning (ML) may be an appropriate solution that can reduce the problems of the blind detection method, such as misdetections and the relatively high computational power that is required. ML can perform statistics-based prediction and classification through data accumulation, so the more data these algorithms learn, the more robust they are to random changes. In addition, the blind detection method has a higher uncertainty and requires more calculations due to the large number of candidate DMRS sequence types, but ML-based classification methods can find a target signal index without such limitations.

In this paper, we conduct research to develop an intelligent software modem that can adapt to dynamically changing communication environments through continuous learning. For future intelligent software modems, machine learning models that ensure performance under various signal-to-noise ratio (SNR) conditions are necessary. To investigate the feasibility of applying machine learning models in real communication environments, we construct an outdoor channel environment using universal software radio peripheral (USRP) and generate and collect demodulation reference signal (DMRS). The collected signals are then used for training and evaluation of the index classification model. Even though some previous studies [1], [2] have attempted to apply ML technologies to channel estimation and prediction, these studies still faced practical limitations. These studies derived results using computer simulations based on mathematical assumptions; they were not based on data measured in a real environment. Therefore, results measured in an actual 5G environment are needed to further evaluate the feasibility of using machine learning technologies for 5G networks. We implemented softwarized modem testbeds that are completely compliant with the current 5G standard to validate various ML technologies in real 5G networks and to produce realistic results [3]. The testbeds were used to transmit and receive 5G signals in a real environment and to measure the signals; then, we constructed a dataset based on these real measurements. In addition, [3] has already demonstrated the superiority of ML-based methods over conventional approaches. Building upon these findings, our main focus in this paper is to enhance the analysis of real 5G data using various ML methods. Specifically, we compare the DMRS index classification performances of several ML models using a DMRS sequence dataset collected from the communications between a gNodeB (gNB) and UE in a 5G communication environment built using our softwarized modem testbeds. We also combine three pre-processing methods to create DMRS sequence datasets and use them as training data for ML models.

This study makes several contributions. Firstly, by utilizing practical gNB signals that comply with the 5G standard, we provide new insights into real-world 5G channel estimation scenarios. Secondly, through the analysis and comparison of different approaches in this work, we offer valuable insights into the impact of these methods on index prediction performance, thereby making a significant contribution to the field. Lastly, our comprehensive analysis enhances our understanding of the effects of pre-processing techniques and facilitates informed decisions regarding their practical applicability.

The rest of this paper is organized as follows. We begin with an explanation of index classification. Then, the actual experimental environments and the structure of the DMRS sequence are described, and three pre-processing techniques are presented. Additionally, brief descriptions of the machine learning models used in the experiments and the learning process are provided. Finally, we evaluate the performances of the machine learning models and compare them with the conventional blind decoding method.

II. RELATED WORK

To compensate for the weaknesses in the relatively-high-frequency band of 5G, it is becoming important to find an appropriate index of the reference signal to create a communication channel between the base station and the UE, and research on these major challenges has been growing. In [4], the authors use the measured power in the received beam to improve the latency that the UE requires to detect cells. Furthermore, they propose a beam scheduling algorithm that can optimize beam sets for cell retrieval and reduce the detection latency, as well as a beam coupling algorithm that combines power in several reception directions. In [5] and [6], the authors propose a novel DMRS structure to increase the channel estimation accuracy of UE and DMRS bundling to improve physical uplink shared channel (PUSCH) coverage, respectively.

Blind detection methods are applied when the presence of a signal needs to be detected without prior knowledge of the signal characteristics or channel information; these methods include initial cell search and the channel estimation of the UE [7]. However, blind detection methods are likely to distort the synchronization signal due to noise when the signal-to-noise ratio (SNR) is low [8]. Conventional orthogonal frequency division multiplexing (OFDM) receivers initially estimate the channel state information (CSI) explicitly and then utilize the estimated CSI to detect or recover the transmitted symbols. In contrast, the deep neural network (DNN)-based approach proposed in [9] implicitly estimates the CSI and directly recovers the transmitted symbols. Simulation results demonstrate that DNN approach achieves performance similar to that of the minimum mean square error estimator in symbol detection. The author of [3] propose an ML-based channel learning scheme that can improve cell search and confirm that ML-based methods perform better than conventional correlation-based blind detection methods. In order to solve the problem of high packet losses caused by overlapping transmissions in random access schemes in IoT wireless access networks, [10] proposes a hybrid deep learning-matched filter approach called HybNet. The proposed model demonstrates its ability to enable the detector to operate even in complex environments where a large amount of data can be transmitted simultaneously. In [11], reinforcement learning is applied to beam search models in addition to ML, and the authors propose a blind beamforming solution that can reduce overhead by applying reinforcement learning to traditional beam search models. Numerical experiments indicate that the proposed solution exhibits better data transfer rates than conventional beam sweeping techniques. In [12], the authors propose a deep learning-based beam selection algorithm that utilizes channel state information (CSI) below 6 GHz to select beams. The proposed algorithm exhibits an improved performance in experimental environments using 3D ray tracking simulations and software-defined radios (SDRs).

SDRs can be used as a general-purpose hardware device only by software replacement. Universal Software Radio Peripheral (USRP) devices, which enable the establishment of a wireless communication network environment at a low price, have made it easier to develop wireless communication applications [13], [14], [15]. Recently, USRP devices have been used as a research tool for developing various applications because they make it possible to build a 5G network environment at the laboratory level without expensive and complex 5G base stations [16], [17], [18], [19], [20]. We validate the DMRS index classification performances of six ML models in different communication environments to build a 5G wireless network environment using these USRP devices and find an optimal ML model.

Compared to the previous studies discussed in this section, we tried to conduct more reliable studies in a real environment. First, a testbed that meets the 5G standard was implemented, and based on this, practical channel data were collected to create a dataset. Efficient machine learning models and pre-processing methods that are suitable for a real environment were developed and analyzed.

III. MACHINE LEARNING-BASED 5G INDEX CLASSIFICATION

One of the main contributions of this paper is to present combinations of ML-based DMRS index classification models and pre-processing methods that are suitable for different channel conditions, as shown in Fig. 1. These models, including the channel-learning models, can be effectively applied as representative scenarios in real-world environments, leveraging the concepts introduced in the channel-learning scheme for synchronization signal block (SSB) index classification [3].

In this section, we describe the DMRS sequence data, the pre-processing methods, and the ML-based models that we applied to our measurements. The DMRS data collection process and the DMRS structure will be described in detail in the first subsection. We generated four training datasets, three of which are pre-processed, and these training datasets are described in the second subsection. For DMRS index classification, four non-neural network (NN) and two NN models are used. The Python-based implementation of the

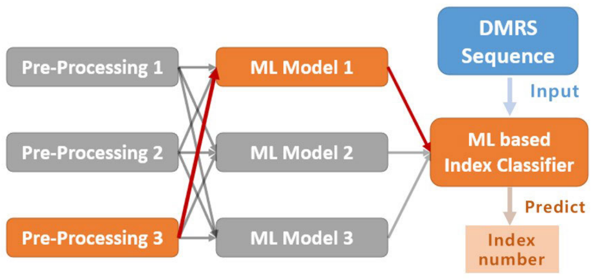

FIGURE 1. Block diagram illustrating ML-based index classification.

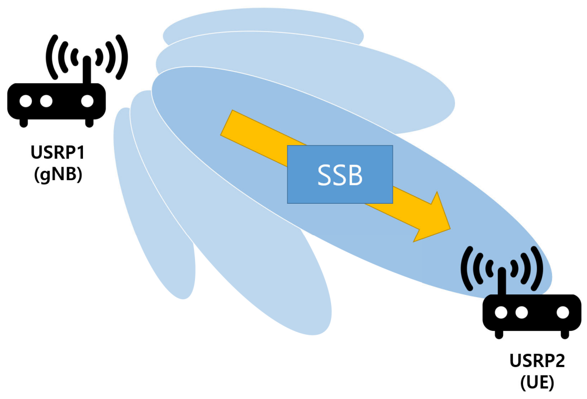

FIGURE 2. The environment in which the DMRS data are collected.

those model can be found in [21]. Note that we define ML models that are not NN-based as non-NN models.

A. DMRS DATA MEASURED IN REAL CHANNEL ENVIRONMENT

We measured and collected DMRS data in realistic environments, as shown in Fig. 2. Two static USRP testbeds are used, with one serving as the gNB and the other as the UE [3]. The gNB creates a beam to cover a single cell's coverage area, and within this coverage area, the UE performs the initial cell selection process. The UE receives the SSB from the gNB, which consists of 4 OFDM symbols and contains 240 subcarriers in each frequency domain. In the first and third OFDM symbols, there are primary synchronization signal (PSS) and secondary synchronization signal (SSS) respectively. After detecting the PSS, the fast Fourier transform (FFT) is performed to detect the SSS. The PSS has 3 candidate sequences, while the SSS has 336 candidate sequences. By utilizing the candidate sequences and indices N_{ID}^{1} (0 2) for PSS and N_{ID}^{2} (0 335) for SSS, we can represent the gNB's PCI (Physical Cell ID) as follows [3]:

$$N_{ID}^{CELL} = N_{ID}^{2} + 3N_{ID}^{1} \tag{1}$$

The last 3 OFDM symbols contain the physical broadcast channel (PBCH) [3]. In the second and fourth symbols, the PBCH is distributed across subcarrier indices 0 to 239. In the third symbol, the PBCH is allocated to subcarrier

TABLE 1. DMRS data collection details.

	Collection Details
SNRs	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
Indices	0, 1,..., 7
Total Number of Sequences	20,000 * 12 (SNRs) * 8 (Indices) = 1,920,000

indices 0 to 47 and 192 to 239. The DMRS is present in the second and fourth symbols on specific subcarrier indices, which are determined by adding v to subcarrier indices $0, 4, 8, \ldots, 236$. In the third symbol, the DMRS is allocated to subcarrier indices $0, 4, 8, \ldots, 44$, and also to subcarrier indices $192, 196, \ldots, 236$, by adding v, where the value of v is obtained by taking the modulo 4 of the N_{ID}^{CELL}.

The parameters used in the testing environment are as follows: The sampling rate is 3072000, the downlink frequency is 3608.79MHz, the NR-ARFCN is 640586, the bandwidth is 20MHz, the FFT size is 1024, the subcarrier spacing is 30kHz, and the SSB transmission period is 20ms [3].

DMRS data are collected evenly; 20,000 pieces of data are collected for each of 12 SNR levels and 8 indices. The details concerning the DMRS data collection are given in Table 1.

The DMRS is a complex signal with 144 elements, and each element can be represented in two-dimensional coordinates with in-phase (I) and quadrature-phase (Q) axes. For a single DMRS, we represent I/Q sequences as $I = \{i_0, i_1, i_2, \ldots, i_{143}\}$ and $Q = \{q_0, q_1, q_2, \ldots, q_{143}\}$, respectively.

B. PRE-PROCESSING METHODS

To investigate the impact of the data pre-processing procedure on the performance of ML models, we applied three methods to DMRS sequences: power averaging, sequence scaling, and principal component analysis (PCA).

1) POWER AVERAGING

Unprocessed data typically have outliers and often cause problems when applied to machine learning due to severe differences in the scales of different values. Therefore, we normalized DMRS sequence elements so that they would have a consistent scale. Through the power-averaging process, the sequence elements are divided by the average power of the signal. The normalized sequence element is expressed by (2):

$$\left(\frac{144 * i_n}{\sum_{k=0}^{143}\sqrt{i_k^2+q_k^2}}, \frac{144 * q_n}{\sum_{k=0}^{143}\sqrt{i_k^2+q_k^2}}\right), \tag{2}$$

where the sequence element's number is $n = 0, 1, \ldots, 143$.

2) SEQUENCE SCALING

Similar to the power-averaging method, the sequence-scaling method is also used to create a consistent scale among signals. However, power averaging only scales based on the power of a single signal; it does not allow sequence elements to be within a specific range. In the sequence-scaling process, sequences are scaled with a mean normalization that ensures that each sequence element is within the range -1 to 1 [22]. Unlike common mean normalization, which scales between features, sequence scaling uses the minimum and maximum values within each sequence. In addition, this method is applied to each of the I/Q values, and a sequence element is expressed as follows:

$$2 * \left(\frac{i_n - \frac{\max(I)+\min(I)}{2}}{\max(I)-\min(I)}, \frac{q_n - \frac{\max(Q)+\min(Q)}{2}}{\max(Q)-\min(Q)}\right), \tag{3}$$

where the sequence element's number is n.

3) PRINCIPAL COMPONENT ANALYSIS

PCA is a technique for converting samples in a high-dimensional space into samples in low-dimensional spaces with no linear association; this technique finds new axes that are orthogonal to each other while preserving the variance of the data as much as possible [23]. In this pre-processing method, the minimum number of new principal components with high explanatory power are extracted to preserve at least 95% of the variance of the original data.

With these three pre-processing methods, four training datasets are generated, including the original dataset, and we define them as follows:

- D_O: Dataset with no pre-processing,
- D_A: Dataset to which power averaging was applied,
- D_S: Dataset to which sequence scaling was applied,
- D_P: Dataset to which PCA was applied.

C. NON-NEURAL NETWORK MODELS

We validate and compare various learning models for performance analysis, and we refer to learning models that do not have neural network structures as non-NN models. We investigate four types of non-NN models: the decision tree, random forest, LightGBM, and stacking models.

1) DECISION TREE

The decision tree model can be used for both classification and regression in supervised training. Initially, a root node is generated that contains all the data. This model automatically creates child nodes using the rules or conditions of the data. The conditions of a node are determined by the impurity. The impurity I can be expressed as follows:

$$I = -\sum_{c} p(c)log_2p(c), \tag{4}$$

where $p(c)$ is the ratio of the data in space to the data belonging to category c. When I is 1, the impurity is maximum, which means that different data with the same proportions coexist in one space. When I is 0, this means that there is only one piece of data in one space. In addition, in the process of generating the decision tree, the amount of

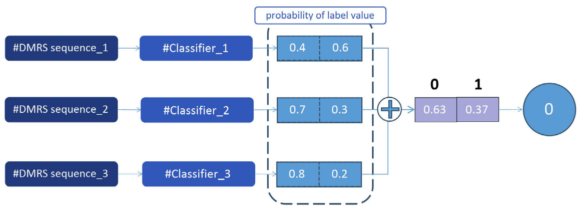

FIGURE 3. Structure of the soft voting model.

information gain is required. The information gain (IG) can be expressed as follows:

$$I_{res} = -\sum_{v} p(v) \sum_{c} p(c|v) log_2 p(c|v), \tag{5}$$

$$IG = I - I_{res}. \tag{6}$$

The IG is the difference between the current impurity and the impurity after division, and it can be expressed using (6). I_{res}, which is calculated using (5), expresses the impurity after division. In order to increase the performance of the classification, the IG must be increased. After the IG is calculated, the decision tree is created by finding a condition for which the IG value is maximized [24].

2) RANDOM FOREST

The random forest algorithm is an ensemble algorithm based on the decision tree, and it has a relatively fast execution speed and a high prediction performance. Additionally, it is a bagging algorithm that creates multiple classifiers and finally determines the output through voting [25]. After multiple decision trees classify data individually, the output of all classifiers is finally determined through the soft voting model.

Fig. 3 shows the structure of the soft voting model. Various classifiers each calculate a probability for the label value. Then, the average is obtained by adding all the corresponding probabilities, and the label value with the highest probability is the final output value [26].

3) LIGHTGBM

Since the existing gradient boosting model (GBM) requires a large amount of computation, efficient packages such as XGBoost, CatBoost, and LightGBM have been developed. In general, gradient boosting uses the level-wise method to create a balanced tree while minimizing the depth of the tree [27]. However, although the level-wise method is good for preventing overfitting, it requires time to balance the tree. In contrast, LightGBM uses the leaf-wise method.

The leaf-wise method increases the depth of the tree by dividing the leaf node, as shown in Fig. 4; this creates an asymmetric tree, and the leaf node has a maximum loss value. Additionally, LightGBM uses the gradient-based one-side sampling (GOSS) algorithm and the exclusive feature bundling (EFB) algorithm. The GOSS algorithm reduces the amount of data to be processed and improves the memory and

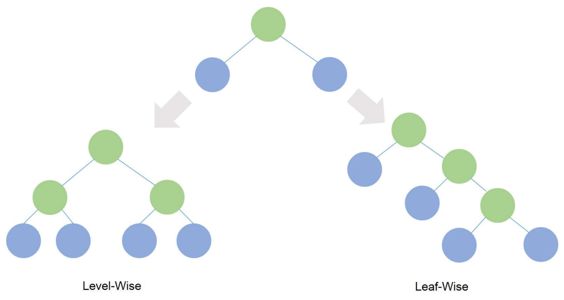

FIGURE 4. Structure of the level-wise and leaf-wise methods.

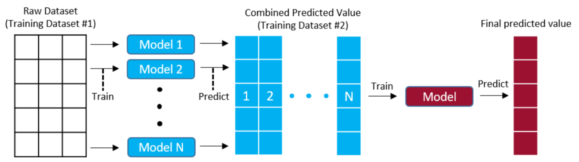

FIGURE 5. Stacking model structure.

processing speed as the amount of data is reduced. The EFB algorithm reduces the amount of feature data, assuming that the high-dimensional data are sparse. Due to the influence of the two algorithms, LightGBM not only minimizes the predictive error loss but also has a fast training time and requires less memory than the original gradient boosting algorithm [28].

4) STACKING

A stacking model is an ensemble model that combines several algorithms to derive predictive results. A stacking model that performs model-specific training and adds predictive label values to generate new data is shown in Fig. 5. The stacking model can be divided into two layers. The first layer is trained on the training data individually and outputs predictions. Then, training data are created using the predicted values, and the final prediction is performed by the second layer with the generated training data. In this paper, the decision tree, random forest, and LightGBM were used as the first layer, and LightGBM was used as the final prediction model [29].

D. NEURAL NETWORK MODELS

We used two NN models for the DMRS classification task. NN models are represented by several layers, which include weight parameters, and they are trained by updating the weights according to the loss using optimizers. The layer configuration of the model depends on which NN is used, and a deep neural network (DNN) and a convolutional neural network (CNN) are applied. As this is an index classification problem, these two models use softmax in the final layer so that they can select the class with the highest numerical value as the predicted class [30]. In addition, we use a categorical cross-entropy loss function and adjust the weights using the adaptive moment estimation (ADAM)

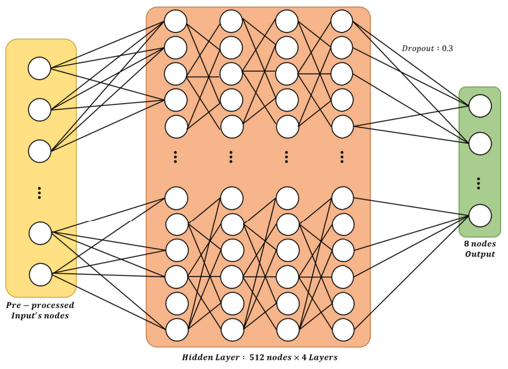
Dropout : 0.3
8 nodes Output
Pre – processed Input's nodes
Hidden Layer : 512 nodes × 4 Layers

FIGURE 6. The structure of the DNN model.

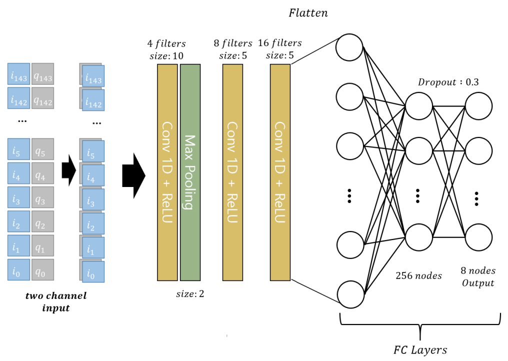

FIGURE 7. The structure of the CNN model with a two-channel input.

optimizer with a learning rate of 0.001 for both models [31], [32]. To determine the parameters for the structure of NN models, such as the number of layers and nodes, we have conducted multiple validations with D_O. During this process, we selected structures with lower complexity while ensuring that they still maintain satisfactory performance. Note that the chosen NN models for our measurements are designed in a general manner, without specific influence from other baselines, in order to compare their performance with other models and ensure a comprehensive evaluation.

1) DNN MODEL

Fig. 6 illustrates the structure of the DNN model used in our experiments. DNN-based models have more than two fully connected (FC) hidden layers, as shown in Fig. 6 [33]. The DNN model used in our experiments consists of four FC hidden layers with 512 nodes for each, and the rectified linear unit (ReLU) activation function is applied before each FC layer [34]. To avoid overfitting, a dropout layer rate of 0.3 is added before the output layer.

2) CNN MODEL

CNN-based models extract the features of inputs for each adjacent section by performing convolutional operations with filters [35]. To effectively extract the features of adjacent regions, we modified the inputs so that they were divided into two channels, which consist of the I and Q units of the sequence, respectively. Note that, unlike other models, the CNN model does not use D_P as an input, because it cannot be represented in an I/Q format. This two-channel input is shown in Fig. 7.

Fig. 7 also shows the structure of the CNN model used in our experiments. Our CNN model has three 1D convolutional layers, which extract the main features of a sequence. Every 1D convolutional layer in the model is followed by a ReLU function and uses a stride of 1 and half-padding. Specifically, the initial layer employs four 10-sized filters, the second uses eight filters of size 5, and the third employs sixteen filters of size 5. To reduce the number of parameters, a max pooling layer with a size of 2 is used. After the convolution process, all parameters are flattened into a one-dimensional form and connected to the next FC layer consisting of 256 nodes. In addition, a dropout layer rate of 0.3 is used to avoid overfitting.

IV. PERFORMANCE EVALUATION

In this section, we evaluate the performance of six ML-based DMRS index classification models for four types of datasets: Four non-NN models are evaluated in the first subsection, and two NN models are evaluated in the next subsection. We measure the average number of decoding attempts needed to identify the correct signal index and compare the ML-based models with the conventional blind decoding method in the subsequent subsection. In the following section, we analyze the computational complexity of each ML model by representing their inference time using Big-O notation. Additionally, we investigate the impact of different pre-processing techniques on the training time of NN-models in the subsequent section. In the last subsection, we comprehensively compare and evaluate all six models.

The entire dataset is divided into training data and test data using a ratio of 8:2, and 20% of the training data are used as validation data to measure the loss value during training.

For all measurements, each model is collectively trained on data with multiple SNR levels and evaluated for each SNR level. For model performance verification, we input test data, which were not used for model training, into the model. We defined the success probability of the index classification as an evaluation metric. For the index x, the classification accuracy S_x is obtained using the number of test data points N_x and the number of correctly classified data points C_x. Using S_x, we define the success probability SP for n indexes as follows:

$$S_x = \frac{C_x}{N_x}, \tag{7}$$

$$SP = \frac{1}{n}\sum_{i=0}^{n-1} S_i. \tag{8}$$

A. NON-NEURAL NETWORK MODELS

Fig. 8 shows the SP values obtained for each pre-processed dataset using four non-NN models in low-SNR cases. When

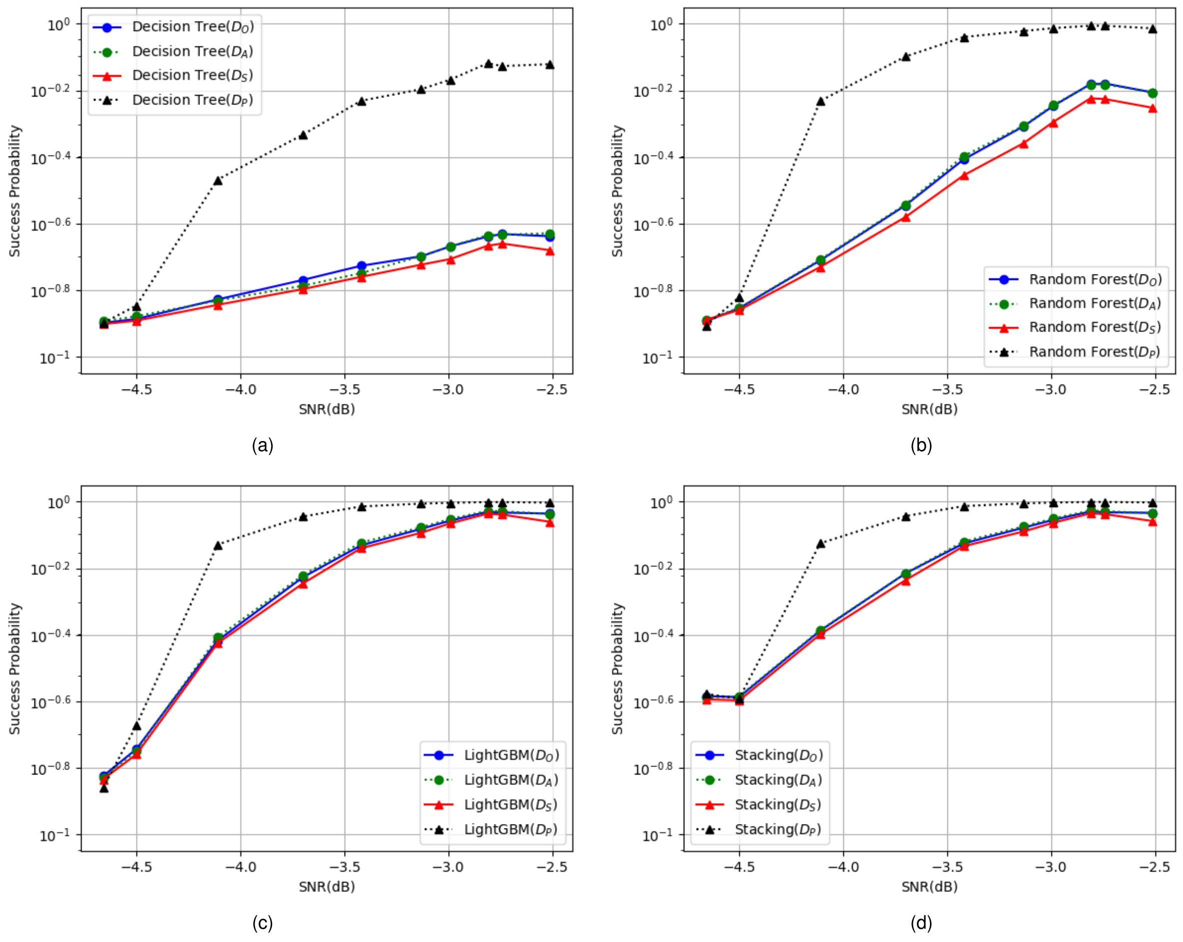

FIGURE 8. Success probability (*SP*) of DMRS index classification for each pre-processing method for non-NN models: (a) Decision tree; (b) random forest; (c) LightGBM; (d) stacking.

the SNR is quite high, all non-NN models except for the decision tree had *SP* values of 1 for all cases, and only the decision tree had an *SP* of about 0.99.

Fig. 8a shows the *SP* of the decision tree for four datasets. At −2.51 dB, the highest SNR shown in the figure, the PCA dataset provides a very high *SP* compared to the other datasets, with an *SP* of about 0.76. Other SNRs also indicate that the PCA dataset resulted in higher *SP* values than the other datasets, and the difference in the *SP* decreased starting at −4.5 dB. All other datasets showed *SP* values of less than 0.3 at −2.51 dB, and it was also found that using the PCA dataset was most appropriate when a decision tree was used. However, the overall results of the decision tree indicate that it is difficult to use in practice.

Fig. 8b shows the *SP* values obtained by the random forest for the four datasets. The random forest obtained the highest *SP* for the PCA dataset when the SNR was −2.51 dB, as shown in Fig. 8a. Likewise, at −2.51 dB, the highest SNR, PCA resulted in the highest *SP*, with an *SP* of about 0.97. However, unlike the results shown in Fig. 8a, when PCA was used, the *SP* remained above 0.9 until the SNR decreased to −3.42 dB. It can be seen that the *SP* of all datasets also decreases as the SNR decreases in all sections, except for SNRs from −2.51 dB to −2.81 dB. In addition, at −4.5 dB and −4.66 dB, the lowest SNRs, all four datasets showed no significant differences in terms of the *SP*. When the random forest is used for index classification, it can be seen that PCA datasets are suitable for multiple SNRs.

Fig. 8c shows the *SP* values obtained by the LightGBM model, and it shows a trend similar to that shown in Fig. 8b. Fig. 8c shows that when PCA was used, the *SP* was maintained at a value of at least 0.9 until the SNR decreased to −3.7 dB. In addition, the PCA dataset resulted in higher *SP* values than the other datasets for all SNRs except −4.66 dB. For LightGBM, unlike the decision tree and random forest, all datasets show *SP* values of 0.8 for SNRs up to −3.13 dB, so it is appropriate to use LightGBM in this SNR range.

Fig. 8d shows the *SP* values obtained for the four datasets using stacking. For the stacking model, all of the datasets for all SNRs showed higher *SP* values than they did for the previous non-NN models. In addition, with PCA, the *SP* remained above 0.97 until the SNR decreased to −3.42 dB,

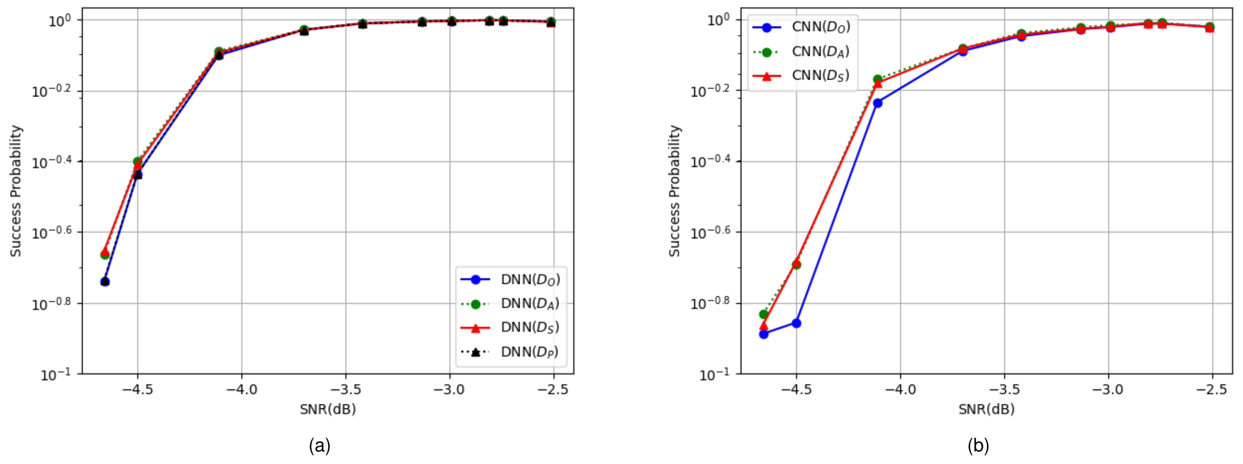

FIGURE 9. Success probability (*SP*) of DMRS index classification for each pre-processing method for NN models: (a) DNN-based model; (b) CNN-based model.

and the *SP* remained above 0.9 until the SNR decreased to −3.7 dB, similar to the results shown in Fig. 8c. In addition, as the SNR decreased, the *SP* also decreased, but for the lowest SNR values (−4.5 dB and −4.66 dB), the *SP* tended to increase slightly, except for the power-averaging dataset.

The non-NN model that is considered most appropriate to use is the stacking model, which has a high *SP* overall. From the dataset perspective, the PCA dataset, which provides a higher *SP* than the other datasets, is considered the most appropriate dataset. If the stacking non-NN model and the PCA dataset are used together, good results are expected when DMRS index classification is performed.

B. NEURAL NETWORK MODELS

In this subsection, we discuss the performance of the NN models for each pre-processing method. The results of the DNN-based model are shown in Fig. 6 and the results of the CNN-based model are shown in Fig. 7. All these experiments were conducted using the popular deep learning framework Keras (version 2.8). During the training of both models, the batch size was 512, and if the loss value converged within 1000 epochs, the training process was terminated early.

Fig. 9 shows the performance of the NN models for each SNR environment. At 23.37 dB and 9.56 dB, which are not shown in Fig. 9, the *SP* is 1 for all pre-processing methods, and the performance decreases starting at −2.51 dB. Meanwhile, in the CNN-based model, as previously mentioned, D_P could not be used to create two channels, so it was not applied.

For all datasets, the DNN-based model shows no significant difference in the *SP* above −3.7 dB, as shown in Fig. 9a. The largest *SP* difference between datasets above −3.7 dB is only about 0.003. However, there is a performance difference for SNRs below this value. Generally, below −4.11 dB, the performance is good for D_A, D_S, D_P, and D_O, and there is no significant *SP* difference between D_P and D_O. Compared with D_O, at −4.11 dB, D_A and D_S improve the *SP* by 0.021 and 0.015, respectively, but D_P only improves the *SP* by 0.014. According to these results, it is beneficial to use data processed with the power-averaging method in a DNN-based DMRS index classifier.

The CNN-based model reveals a pattern similar to that shown by the DNN. As shown in Fig. 9b, D_A and D_S did not show much difference in the *SP*, and when D_O was applied, the lowest *SP* was recorded for all SNRs. At −2.99 dB, D_A and D_S improved by 0.015 and 0.004, respectively, compared to D_O, but at −4.11 dB, they improved by 0.071 and 0.054, respectively. This result also indicates that D_A provides slightly higher *SP* values than D_S in general. Thus, we can reveal that power averaging is the most advantageous pre-processing method for a CNN-based DMRS index classifier according to these results.

C. AVERAGE NUMBER OF ATTEMPTS TO IDENTIFY THE SIGNAL INDEX FOR ML MODELS

We compare the average number of attempts of the ML-based methods with the average number of attempts of the conventional blind decoding method. Herein, the average number of attempts is defined as the average number of decoding operations performed before the correct index is identified. The conventional blind decoding method checks the signal sequences one by one to find the correct signal index. For ML-based methods, we calculate the average number of attempts based on the measured *SP* values.

The conventional blind decoding method involves simple random sampling without replacement. Thus, the probability of finding an index in every trial is equal to $1/N$ for the total number of indices N. We can represent the average number of attempts of the conventional blind decoding method as follows:

$$\frac{\sum_{i=1}^{N} i}{N}, \tag{9}$$

where i represents the number of trials.

TABLE 2. Average number of attempts to detect the correct signal index for 5G.

Method	SNR (dB)											
	23.37	9.56	-2.51	-2.74	-2.81	-2.99	-3.13	-3.42	-3.7	-4.11	-4.5	-4.66
Conventional Blind Decoding	4.5											
Decision Tree (D_O)	1.00	1.01	4.33	4.27	4.34	4.65	4.98	5.32	5.87	6.72	7.69	7.89
Decision Tree (D_A)	1.00	1.01	4.24	4.29	4.30	4.65	4.98	5.60	6.11	6.77	7.54	7.81
Decision Tree (D_S)	1.00	1.01	4.78	4.56	4.63	5.08	5.28	5.74	6.25	6.97	7.77	7.96
Decision Tree (D_P)	1.00	1.00	1.32	1.33	1.31	1.47	1.57	1.70	2.15	2.93	7.00	7.90
Random Forest (D_O)	1.00	1.00	1.60	1.51	1.51	1.76	2.02	2.55	3.50	5.13	7.13	7.81
Random Forest (D_A)	1.00	1.00	1.60	1.52	1.52	1.75	2.02	2.49	3.48	5.09	7.13	7.73
Random Forest (D_S)	1.00	1.00	1.78	1.68	1.67	1.97	2.27	2.85	3.80	5.37	7.23	7.73
Random Forest (D_P)	1.00	1.00	1.03	1.01	1.01	1.03	1.05	1.09	1.25	1.70	6.61	8.05
LightGBM (D_O)	1.00	1.00	1.08	1.07	1.07	1.13	1.20	1.35	1.68	2.61	5.53	6.67
LightGBM (D_A)	1.00	1.00	1.08	1.06	1.06	1.12	1.19	1.32	1.66	2.55	5.59	6.73
LightGBM (D_S)	1.00	1.00	1.14	1.09	1.08	1.16	1.23	1.38	1.76	2.66	5.73	6.81
LightGBM (D_P)	1.00	1.00	1.00	1.00	1.00	1.00	1.01	1.03	1.10	1.34	4.69	7.23
Stacking (D_O)	1.00	1.00	1.07	1.07	1.06	1.13	1.19	1.33	1.64	2.44	3.86	3.83
Stacking (D_A)	1.00	1.00	1.08	1.06	1.06	1.12	1.18	1.31	1.64	2.43	3.84	3.88
Stacking (D_S)	1.00	1.00	1.14	1.08	1.08	1.15	1.22	1.36	1.72	2.50	3.95	3.92
Stacking (D_P)	1.00	1.00	1.00	1.00	1.00	1.00	1.01	1.03	1.10	1.33	3.90	3.77
DNN (D_O)	1.00	1.00	1.01	1.00	1.00	1.00	1.01	1.02	1.07	1.26	2.76	6.12
DNN (D_A)	1.00	1.00	1.01	1.00	1.00	1.00	1.01	1.02	1.07	1.22	2.55	4.58
DNN (D_S)	1.00	1.00	1.01	1.00	1.00	1.00	1.01	1.02	1.07	1.23	2.55	4.52
DNN (D_P)	1.00	1.00	1.01	1.00	1.00	1.00	1.01	1.02	1.06	1.25	2.70	5.36
CNN (D_O)	1.00	1.00	1.05	1.03	1.03	1.05	1.07	1.12	1.26	1.64	7.04	7.91
CNN (D_A)	1.00	1.00	1.05	1.02	1.03	1.04	1.05	1.09	1.21	1.47	4.91	6.78
CNN (D_S)	1.00	1.00	1.05	1.02	1.02	1.05	1.06	1.10	1.21	1.51	4.83	7.29

The ML-based methods involve sampling with replacement, so a different equation is needed to calculate the average number of attempts. When a certain model M has an SP of SP_M, we can represent M's average number of attempts (A_M) as follows:

$$A_M = \sum_{i=1}^{\infty}((1 - SP_M)^{i-1} \cdot SP_M \cdot i), \tag{10}$$

where i represents the number of trials.

From the perspective of the average number of attempts, ML-based methods significantly outperform the conventional method for high SNRs, but their performance decreases rapidly when the SNR is lower than a particular value. Table 2 shows the average number of attempts for all methods, and the results for SNRs under 9.56 dB are shown in Fig. 10. Non-NN methods other than the decision tree at −4.11 dB or higher require an average number of attempts close to 1 when using D_P. NN methods using any pre-processing method require an average number of attempts close to 1 at −4.11 dB or higher. On the other hand, the conventional blind decoding method requires 4.5 attempts on average for all SNRs. This means that ML-based methods require about 3.5 fewer attempts on average than the conventional method at SNRs of −4.11 dB and above. However, when the SNR drops to −4.5 dB, the average number of attempts of the ML-based methods increases rapidly, and at −4.66 dB, most ML-based methods require more attempts than the conventional method on average. In environments with low SNR, noise or interference tends to be more pronounced compared to the signal. In such cases, ML-based models can learn the patterns of noise or interference. This result can be observed in Fig. 10, where the conventional model demonstrates better performance than most of ML models. It is clear that a process that increases the SNR itself, such as signal accumulation, is required to distinguish DMRS indexes under a specific SNR using the ML-based methods. With this process, we can generate a highly efficient ML-based DMRS index classifier.

D. INFERENCE COMPLEXITY ANALYSIS FOR ML MODELS

In the context of 5G network environments, enhancing the prediction accuracy of an inference method is important, but it is also crucial to improve the computational efficiency of the software. Therefore, in this section, we analyze the computational complexity of each model in terms of the operations required for index inference, using the Big-O notation. It is important to note that the specific computational complexity may vary depending on the structure and parameters of each model.Note that the variables in this subsection are distinct from those used elsewhere.

Table 3 represents the inference complexity of each non-NN model using Big-O notation. The decision tree performs class inference by traversing condition nodes up to the tree depth (D). In addition, random forest and LGBM models perform inference by traversing a number of trees

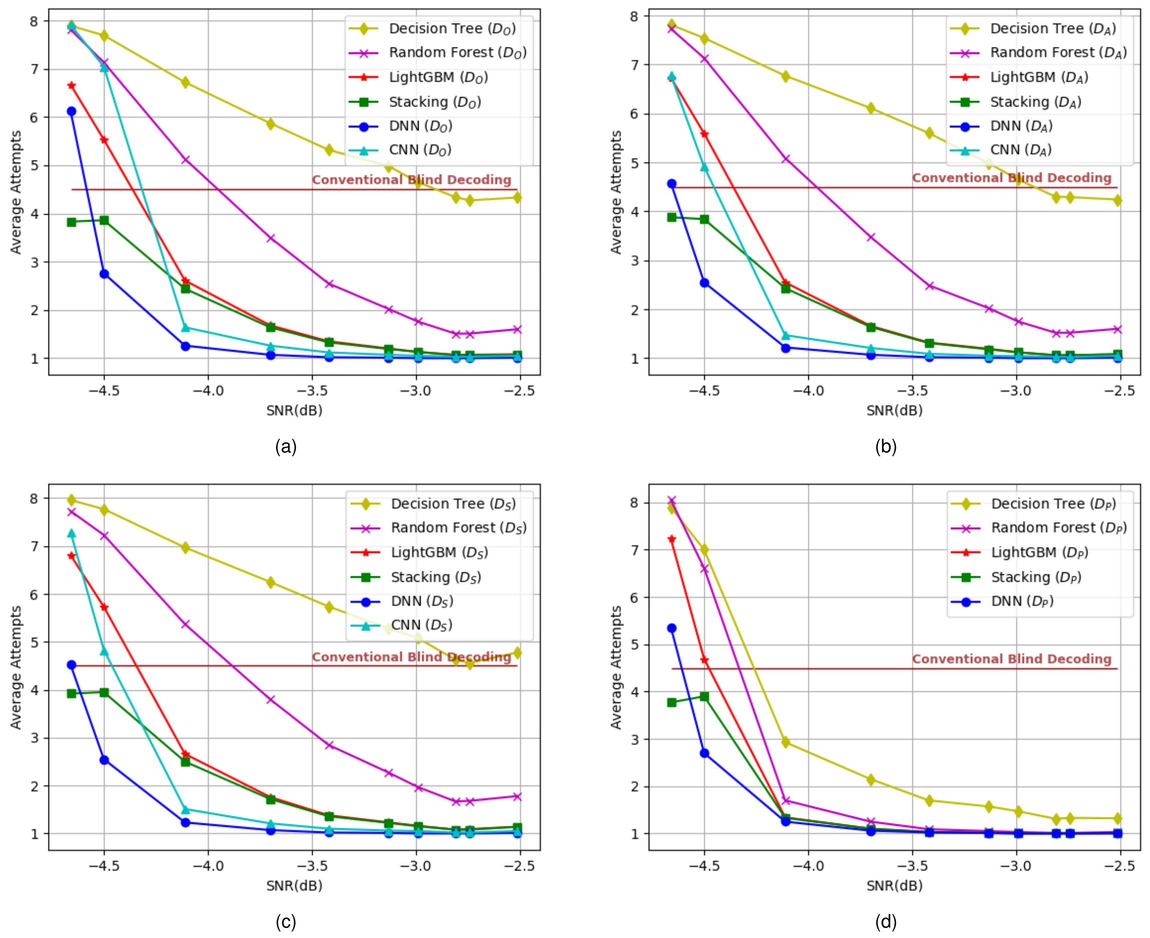

FIGURE 10. The average number of attempts needed to decode the DMRS index for the ML-based methods and conventional blind decoding method for different datasets: (a) D_O; (b) D_A; (c) D_S; (d) D_P.

TABLE 3. Inference complexity of each Non-NN model.

Model	Inference Complexity (Big-O Notation)
Decision Tree	$O(D)$
Random Forest, LGBM	$O(E \times D)$
Stacking	$O((3E+1) \times D)$

equal to the number of estimators (E). For the stacking model, it requires passing through the first and second layers. In terms of inference complexity, the first layer includes the three preceding models, while the second layer includes the LGBM model. Therefore, the overall complexity is the sum of the complexities of these four models.

The inference complexity of NN-models varies greatly depending on the configuration of the layers, and therefore, inference is handled at each NN-model's key layers in this part. In DNN models, the commonly used layer is the fully connected (FC) layer, where each node in the current layer is multiplied by its corresponding weights and summed to form a node in the next layer. Consequently, the FC layer requires inference computations proportional to the product of the number of nodes in the current layer (L_c) and the number of nodes in the next layer (L_n).

TABLE 4. Inference complexity of each NN layer.

Layer	Inference Complexity (Big-O Notation)
FC	$O(L_c \times L_n)$
1D Convolutional	$O(S \times C \times F_n \times F_s)$

The CNN models employed in this paper mainly utilize 1D convolutional layers, followed by additional FC layers for classification purposes. The 1D convolutional layer performs convolutional operations by shifting filters one step at a time over the input sequence. Each filter conducts convolutions based on the length (S) and channel count (C) of the input sequence, repeated for the number of filters (F_n) used. Denoting the filter size as $(1 \times F_s)$, the inference complexity of each key layer in the NN model can be expressed using Big-O notation, as presented in Table 4.

In the case of inference with non-NN models, the value of D represents the maximum number of classification conditions that can be represented by a tree, up to 2^D conditions. Consequently, it is generally unnecessary for D

TABLE 5. Average epochs for model training with different pre-processing methods.

Model	D_O	D_A	D_S	D_P
DNN	44.5	24.6	26.5	45.6
CNN	61.2	20.4	23.6	-

to have a large value. In our measurements, the maximum value of D is set to a sufficient value of 24. As a result, even with a large value of E, it can be observed that non-NN models do not require extensive computations in terms of inference.

On the other hand, NN-models generally require more extensive computations in inference compared to non-NN models. DNN models consist of multiple FC layers and typically have hidden layers with a larger number of nodes than the input features. Moreover, the nodes in these layers are fully connected, forming a dense network of connections between the current and next nodes, which increases the computational complexity. While the convolutional layers in CNN models are much less complex than DNN models as they are not fully connected to all the preceding and subsequent elements in the sequence, they still involve relatively higher computational requirements compared to non-NN models. Additionally, CNN models often include an FC layer as the final classifier, further contributing to the overall computational load. In conclusion, non-NN models offer computational advantages over NN models in terms of inference.

E. TRAINING TIME COMPARISON FOR PRE-PROCESSING METHODS ON NN-MODELS

Based on the measurements presented, it is evident that non-NN models exhibit significant superiority when using PCA compared to other pre-processing methods. However, for NN models, it is challenging to determine the most suitable pre-processing method as there is no clear hierarchy. Therefore, this subsection focuses on analyzing the impact of pre-processing methods on training time for NN models. The metric used to evaluate training time is the number of epochs required for training, with the SP of the model converging to its maximum value. Early stopping criteria for model training are defined as no improvement in SP within 20 epochs. The average number of epochs required for training across all SNR levels is used as the comparative metric. When considering the 12 SNR levels and their corresponding training epochs for each model ($e_1, e_2, \ldots, e_{12}$), the metric can be expressed as $\frac{1}{12}\sum_{i=1}^{12} e_i$.

Table 5 presents the average number of epochs required for training each NN model with different pre-processing methods. For the DNN model, the epoch counts were lower in the order of D_A, D_S, D_O, and D_P. Although D_A and D_S showed a slight difference, using D_O and D_P required approximately 20 additional epochs compared to the former. The CNN model exhibited a more pronounced trend compared to the DNN model, particularly with a

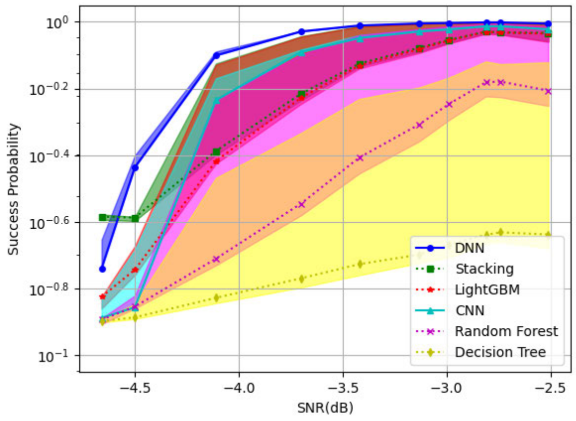

FIGURE 11. Comparison of the performances of all models. The lines show the results of applying D_O to each model, and each area with the same color as a line shows the best and worst performance for the model associated with that line when different pre-processing methods are applied.

threefold increase in training time when using D_O compared to D_A. Overall, from the perspective of training time, the power-averaging method demonstrated superiority among the NN models.

F. COMPREHENSIVE RESULT

In this subsection, we derive a comprehensive result by comparing the performances of all models, which are shown in Fig. 11. Except for the SNR of −4.66 dB, the highest-performing model is the DNN-based model, regardless of the pre-processing method that is used. The models are ranked by performance in the following order: DNN, stacking, LightGBM, CNN, random forest, and decision tree. Overall, among the models used in this experiment, it is most ideal to use a DNN-based model in environments in which an NN model can be implemented. In environments in which NN models are not available, it is appropriate to apply PCA to the available datasets and use them to train the stacking model. On the other hand, it can be seen that the decision tree does not perform well enough to be used in practice; it has an SP of less than 0.9 at −2.51 dB and is not suitable for DMRS index classification. In addition, it is clear that the performance of non-NN models depends greatly on the pre-processing method compared with NN models, so pre-processing methods such as PCA are essential.

V. CONCLUSION

In this work, we first considered the possibility of using ML technologies to perform 5G reference signal identification in practical environments. Practical testbeds with embedded 5G softwarized modems were implemented and used to collect a 5G signal dataset from real communications. Suitable combinations of ML models and pre-processing methods that can be used when ML technology is applied in 5G networks were investigated and verified in practice. This allowed us to demonstrate the feasibility of applying our findings to real-world systems and confirm the potential for performance improvement. In addition, our results can be used to apply an ML-based DMRS index classification model efficiently. In future research, we will propose new ML-based models that are suitable for various real-world environments based on our research findings. Through this, we aim to create robust and generalized models that perform well in practical settings and contribute to the intelligentization of base stations. Furthermore, we acknowledge the importance of hyperparameter tuning in optimizing the performance of ML models. Therefore, we plan to conduct extensive experiments in the future to explore the impact of different hyperparameter settings on the performance of the models. We will also analyze existing studies that can reduce complexity and improve performance, further improving the accuracy and efficiency of the proposed machine learning-based model in real-world 5G environments.

ACKNOWLEDGMENT

(Seungwoo Kang and Taegyeom Lee contributed equally to this work.)

REFERENCES

[1] U. Mutlu and Y. Kabalci, "Deep learning aided channel estimation approach for 5G communication systems," in *Proc. 4th Global Power, Energy Commun. Conf. (GPECOM)*, Jun. 2022, pp. 655–660.

[2] D. Maruyama, K. Kanai, and J. Katto, "Performance evaluations of channel estimation using deep-learning based super-resolution," in *Proc. IEEE 18th Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2021, pp. 1–6.

[3] 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.

[4] S. Kadambar, A. Goyal, and A. K. R. Chavva, "Millimeter wave multi-beam combining algorithm for efficient 5G cell search," in *Proc. IEEE 17th Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2020, pp. 1–6.

[5] G. Noh, B. Hui, J. Kim, H. S. Chung, and I. Kim, "DMRS design and evaluation for 3GPP 5G new radio in a high speed train scenario," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, Dec. 2017, pp. 1–6.

[6] Z. Guo, D. Chen, and Y. Yuan, "5G NR uplink coverage enhancement based on DMRS bundling and multi-slot transmission," in *Proc. IEEE 20th Int. Conf. Commun. Technol. (ICCT)*, Oct. 2020, pp. 482–486.

[7] H. Sun, E. Viterbo, and R. Liu, "Significance-test based blind detection for 5G," *IEEE Trans. Veh. Technol.*, vol. 71, no. 7, pp. 7957–7962, Jul. 2022.

[8] H. Sun, E. Viterbo, and R. Liu, "Efficient blind detection scheme based on simplified decoding of polar codes," *IEEE Wireless Commun. Lett.*, vol. 10, no. 4, pp. 864–868, Apr. 2021.

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

[10] K. Dakic, B. Al Homssi, M. Lech, and A. Al-Hourani, "HybNet: A hybrid deep learning-matched filter approach for IoT signal detection," *IEEE Trans. Mach. Learn. Commun. Netw.*, vol. 1, pp. 18–30, 2023.

[11] V. Raj, N. Nayak, and S. Kalyani, "Deep reinforcement learning based blind mmWave MIMO beam alignment," *IEEE Trans. Wireless Commun.*, vol. 21, no. 10, pp. 8772–8785, Oct. 2022.

[12] M. S. Sim, Y.-G. Lim, S. H. Park, L. Dai, and C.-B. Chae, "Deep learning-based mmWave beam selection for 5G NR/6G with Sub-6 GHz channel information: Algorithms and prototype validation," *IEEE Access*, vol. 8, pp. 51634–51646, 2020.

[13] O. Holland, H. Bogucka, and A. Medeisis, *Opportunistic Spectrum Sharing and White Space Access: The Practical Reality*. Hoboken, NJ, USA: Wiley, 2015.

[14] M. N. O. Sadiku and C. M. Akujuobi, "Software-defined radio: A brief overview," *IEEE Potentials*, vol. 23, no. 4, pp. 14–15, Oct. 2004.

[15] T. B. Welch and S. Shearman, "Teaching software defined radio using the USRP and LabVIEW," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2012, pp. 2789–2792.

[16] A. Kumar and S. Noghanian, "Wireless channel test-bed for DSRC applications using USRP software defined radio," in *Proc. IEEE Antennas Propag. Soc. Int. Symp. (APSURSI)*, Jul. 2013, pp. 2105–2106.

[17] M. El-Hajjar, Q. A. Nguyen, R. G. Maunder, and S. X. Ng, "Demonstrating the practical challenges of wireless communications using USRP," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 194–201, May 2014.

[18] J. Liang, H. Chen, and S. C. Liew, "Design and implementation of time-sensitive wireless IoT networks on software-defined radio," *IEEE Internet Things J.*, vol. 9, no. 3, pp. 2361–2374, Feb. 2022.

[19] F. Peng, S. Zhang, S. Cao, and S. Xu, "A prototype performance analysis for V2V communications using USRP-based software defined radio platform," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Nov. 2018, pp. 1218–1222.

[20] J. Muslimin, A. L. Asnawi, A. F. Ismail, and A. Z. Jusoh, "SDR-based transceiver of digital communication system using USRP and GNU radio," in *Proc. Int. Conf. Comput. Commun. Eng. (ICCCE)*, Jul. 2016, pp. 449–453.

[21] (2023). *ML-Based DMRS Singal Analysis*. [Online]. Available: https://github.com/SeungwKang/ML-based-DMRS-Signal-Analysis

[22] V. Safak, "Min-mid-max scaling, limits of agreement, and agreement score," 2020, *arXiv:2006.12904*.

[23] H. Abdi and L. J. Williams, "Principal component analysis," *WIREs Comput. Statistic*, vol. 2, no. 4, pp. 433–459, Jul./Aug. 2010.

[24] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Trans. Syst. Man, Cybern.*, vol. 21, no. 3, pp. 660–674, May/Jun. 1991.

[25] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, Dec. 2002.

[26] A. Singh, N. Thakur, and A. Sharma, "A review of supervised machine learning algorithms," in *Proc. 3rd Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, Mar. 2016, pp. 1310–1315.

[27] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. Int. Conf. Mach. Learn.*, vol. 96, 1996, pp. 148–156.

[28] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 3146–3154.

[29] O. Sagi and L. Rokach, "Ensemble learning: A survey," *WIREs Data Mining Knowl. Discovery*, vol. 8, no. 4, Jul. 2018, Art. no. e1249.

[30] Y. Wu, J. Li, Y. Kong, and Y. Fu, "Deep convolutional neural network with independent softmax for large scale face recognition," in *Proc. 24th ACM Int. Conf. Multimedia*, Oct. 2016, pp. 1063–1067.

[31] Y. Ho and S. Wookey, "The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling," *IEEE Access*, vol. 8, pp. 4806–4813, 2020.

[32] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl, "On empirical comparisons of optimizers for deep learning," 2019, *arXiv:1910.05446*.

[33] S. K. Dhull and K. K. Singh, "ECG beat classifiers: A journey from ANN to DNN," *Proc. Comput. Sci.*, vol. 167, pp. 747–759, Jan. 2020.

[34] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2017, *arXiv:1710.05941*.

[35] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Networks*, vol. 3361, no. 10. Cambridge, MA, USA: MIT Press, 1995, p. 1995.

SEUNGWOO KANG (Student Member, IEEE) received the B.S. degree in computer science from Chungbuk National University, Cheongju, South Korea, in 2021, where he is currently pursuing the M.S. degree with the Department of Computer Science. His current research interests include time series prediction and deep learning-based communication systems.

TAEGYEOM LEE (Student Member, IEEE) received the B.S. and M.S. degrees in computer science from Semyung University, Chungbuk, South Korea, in 2018 and 2020, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Science, Chungbuk National University, Cheongju, South Korea. His current research interests include deep learning-based 5G mobile communication systems, reinforcement learning-based medium access, multi-task learning for communication systems, and future wireless applications. He was a recipient of best paper awards, in 2021, fall conferences of the Korean Institute of Communications and Information Sciences.

JONGSEOK KIM (Student Member, IEEE) received the B.S. degree in computer science from Chungbuk National University, Cheongju, South Korea, in 2023. In his undergraduate, he was a Research Assistant with the Information System Laboratory under the supervision of Prof. Ohyun Jo. His current research interests include deep learning-based communications systems and real-time configurable systems.

A-REUM-SAEM LEE (Student Member, IEEE) received the B.S. degree in computer engineering from Chungbuk National University, Cheongju, South Korea, in 2012, where she is currently pursuing the M.S. degree with the Department of Computer Science. Her current research interest includes deep learning-based communications systems.

JUYEOP KIM (Member, IEEE) received the B.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2004 and 2010, respectively. He is currently an Assistant Professor with the Department of Electronics Engineering, Sookmyung Women's University, Seoul, South Korea. From 2010 to 2011, he was with the KAIST Institute IT Convergence Research Center in charge of research for 5G cellular systems. 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 the Korea Railroad Research Institute in charge of research and development for LTE-railway (LTE-R), public safety LTE (PS-LTE), and railway IoT solutions. 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 the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2005, 2007, and 2011, respectively. He is currently an Associate Professor with the Department of Computer Science, Chungbuk National University, Cheongju, South Korea. 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 and from August 2017 to February 2018, he was an Assistant Professor with the Department of Electrical Engineering, Korea Military Academy. He has authored or coauthored more than 70 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 numerous recognitions, including the Gold Prize in Samsung Annual Award, Most Creative Researcher of the Year Award, Best Mentoring Award, Major Research and Development Achievement Award, and Best Improvement of Organization Culture Award.