2021 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI)

2021 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI) | 978-1-6654-1236-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/SWC50871.2021.00060

Deep Learning Based Link-Level Abstraction for mmWave Communications

Jian Wang, Neeraj Varshney, Jiayi Zhang, David Griffith, and Nada Golmie
Wireless Networks Division, National Institute of Standards and Technology (NIST), USA
Emails: {jian.wang, neeraj.varshney, jiayi.zhang, david.griffith, nada.golmie}@nist.gov

***Abstract*—Link abstraction is an efficient way for predicting link-level performance (e.g., whether the receiver successfully receives packets through communication links). It can be used to support system-level simulation and facilitate link adaption. Nonetheless, it is challenging to quantify the collective impact of modulation, channel coding, interleaving, guard intervals, and other components on error performance of the communication link. In addition, the existing link abstraction schemes assume ideal hardware with perfect channel estimation. To address this issue, we design a deep learning approach to carry out link-level abstraction for millimeter wave (mmWave) communications. Specially, we conduct link-level simulation based on IEEE 802.11ay single carrier (SC) communication to collect training data, and describe the detailed training and prediction procedures in our approach. We further discuss model reuse and adaptation so that the trained model can be applied to different systems in different environments. Through extensive performance evaluation with data collected on 60 GHz mmWave channels from four environments, we show the prediction accuracy of our approach with respect to feature selection, learning model structure, and the amount of training data. We also demonstrate the efficacy of our approach in model reuse in cross-environments and cross-systems.**

***Keywords*-Link abstraction, mmWave communications, machine learning**

I. Introduction

Link abstraction is essential for system-level simulations, as a viable way to predict link-level performance without going through physical layer simulation [1], [2]. It can avoid the need to perform bit-by-bit link-level simulations within system-level simulations, which is often a very time-consuming process. Moreover, with the predicted link-level error performance, link abstraction can be further used to carry out link adaptation based on channel characteristics at the receiver to improve the spectrum efficiency [2].

Nonetheless, as the complex transceiver system consists of a number of components, including channel coding, modulation, interleaving, and guard interval insertion, among others, it is challenging to derive a closed formula, which is capable of approximating relationships between a number of input parameters that characterize the channel at the receiver and the output (i.e., error performance). Even though there are some existing analytical bounds for link abstraction, some approximations and assumptions have been made [1], [3]. Another challenge is that these existing schemes only consider ideal hardware and perfect channel knowledge at the receiver (i.e., no channel estimation error) when deriving the mathematical expressions, leading to too optimistic prediction results.

The most widely adopted link abstraction schemes are based on effective signal-to-noise ratio mapping (ESM) [1]. The ESM-based schemes map a vector of the received subcarrier SNR values to an additive white Gaussian noise (AWGN) channel equivalent signal-to-noise ratio (SNR), which would yield the same packet error rate (PER). Subsequently, a look-up table per MCS is used to convert the AWGN channel equivalent SNR to a PER value [4]. The two most representative ESM-based schemes are effective exponential SNR mapping (EESM) and mean mutual information per coded bit (MMIB) [1], [3], which have been designed for multi-carrier systems such as Orthogonal Frequency Division Multiplexing (OFDM) mode transmission in IEEE 802.11ay. There are some research efforts [1], [3], [5], which extend ESM-based schemes for single carrier (SC) communications. All these ESM-based schemes involve approximations along with assuming perfect channel estimation and ideal hardware. However, in a dynamic environment with various levels of hardware impairments of individual devices and different signal processing algorithms, it is hard to derive an accurate and fixed formula to characterize the link error performance.

To overcome the aforementioned limitations, in this paper, we design a deep neural network (DNN) based approach to carry out link-level error performance prediction based on a number of input features, such as received signal power, signal bandwidth, channel multi-paths, and MCS. As a viable approach to extract insightful knowledge from data, machine learning has been applied extensively to a number of areas, including image processing, natural language process, robotics, and Internet of Things (IoT), among others [6], [7]. Although there are some existing works on applying machine learning to link-level performance [8], [9], to our best knowledge, no existing research efforts have been done to design approaches that are capable of predicting Bit Error Rate (BER) directly on SC based millimeter wave (mmWave) systems.

Our contributions are summarized as follows:

- *A Novel Approach:* We design the DNN based approach to carry out link-level error performance prediction and conduct IEEE 802.11ay link-level simulation to collect training data. We explicitly describe the details of the training and prediction procedures that we have developed. Our approach enables transfer learning, i.e., the model reuse and adaption such that a trained model can be reused in different channel environments and continuously tuned for different wireless systems.

- *Extensive Evaluation:* Based on the 60 GHz mmWave channel data collected under four different environments (i.e., lecture room, data center, open area hotspot, and large hotel lobby), we conduct extensive experiments to demonstrate prediction accuracy of the BER in terms of feature selection, model structure, and the amount of training data. Furthermore, our experimental results demonstrate the re-usability of our trained model in different environments as well as for different systems.

The remainder of the paper is organized as follows. In Section II, we introduce link abstraction. In Section III, we describe the IEEE 802.11ay PHY simulation platform which we used to collect training data. In Section IV, we introduce our machine learning based approach to conduct link-level error performance prediction. In Section V, we present the performance evaluation results. Finally, we conclude the paper in Section VI.

II. Link Abstraction

The ESM is the widely used link abstraction technique, which can be generally described as

$$\gamma_{\text{eff}} = -\beta_1 \Phi^{-1}\left(\frac{1}{N}\sum_{n=1}^{N}\Phi\left(-\frac{\gamma_n}{\beta_2}\right)\right), \tag{1}$$

where γ_{eff} is the effective SNR, $\Phi^{-1}(\cdot)$ is the inverse mapping function of $\Phi(\cdot)$, and β_1 and β_2 are the scalar parameters which need to be optimized for each MCS. Depending on the mapping function $\Phi(\cdot)$, several ESM approaches have been proposed in the literature [10], and the most widely used ones are EESM and MMIB.

The EESM and MMIB schemes were originally proposed for multi-carrier systems. In the context of single-carrier systems, post-processing ESM (PPESM) [1] demonstrated good performance for IEEE 802.11ay SC mode. Nevertheless, PPESM is derived based on post processing SNR measured after the frequency domain equalizer (FDE), which is highly dependent on the PHY implementation.

In addition to the conventional ESM-based schemes, there are only a few research efforts that explore machine learning-based schemes to characterize the link-level performance [8], [11]. For example, Carreras *et al.* [8] designed a logistic regression-based scheme to map receiver subcarrier SNRs to an AWGN-equivalent SNR for a multi-carrier system. Chu *et al.* [11] used a traditional EESM approach, but employed DNN to train β_1 and β_2 parameters.

Besides predicting the link-level performance (e.g., BER and PER), another use of link-level abstraction is to carry out the link adaptation that plays an important role in modern wireless communication systems, such as long-term evolution (LTE) and Wireless Local Area Network (WLAN) systems [12]. With link-level adaptation, MCS can be dynamically assigned in order to maximize the spectral efficiency while satisfying the reliability requirements. For example, in the 4G cellular system, the user equipment (UE) reports its channel quality indicator (CQI) back to the base station. CQI indicates that the data rate can be supported under current channel conditions and usually involves block error rate (BLER) prediction. Based on the received CQI value, the base station may assign a new MCS value to the UE in order to keep the BLER below a target value. Similarly, the link adaptation can also be used in WLAN systems between the access point (AP) and the station (STA) [13], [14].

III. IEEE 802.11ay SC PHY Simulation over Quasi-Deterministic mmWave Channel

IEEE 802.11ay [15] is the next generation WLAN standard operating at 60 GHz mmWave band. It supports both OFDM and SC modes for data transmission. In our study, we focus on SC mode[1], and use IEEE 802.11ay SC PHY to collect training data. Although our study focuses on IEEE 802.11ay SC PHY, our methodology is generic and can be extended to IEEE 802.11ay OFDM and any other PHY systems.

A block diagram of IEEE 802.11ay SC PHY is shown in Fig. 1. After a data packet is randomly generated, it will go through scrambling, low-density parity-check (LDPC) encoding, modulation, and interleaving operations. Before the transmission, Inverse Discrete Fourier Transform (IDFT) will be performed and guard interval (GI) will be further applied to deal with the inter-symbol-interference. After going through the wireless channel, a chain of reverse operations, including FDE, is applied to recover the transmitted packet at the receiver. Using the decoded packet, the BER is derived by counting the number of bits in error and dividing the count by the total number of bits transmitted within a packet[2]. These BER values are considered as a part of training data along with their multi-tap channel responses obtained through a Quasi-Deterministic (QD) channel model, as described below.

The QD model, proposed by IEEE Task Group 802.11ay (TGay) [16], uses ray tracing to generate the dominant specular rays in an environment and relies on the QD parameters to statistically recreate the diffused rays. In general, QD model provides a good trade-off between accuracy and

[1]It is worth noting that IEEE 802.11ay supports a number of MCS for different throughput. In particular, SC PHY supports 21 MCS configurations, i.e., MCS 1 to MCS 21.

[2]In this work, the data packet length is set to 4096 bytes.

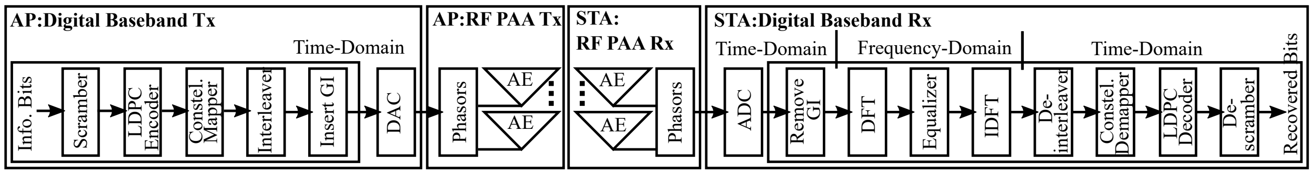
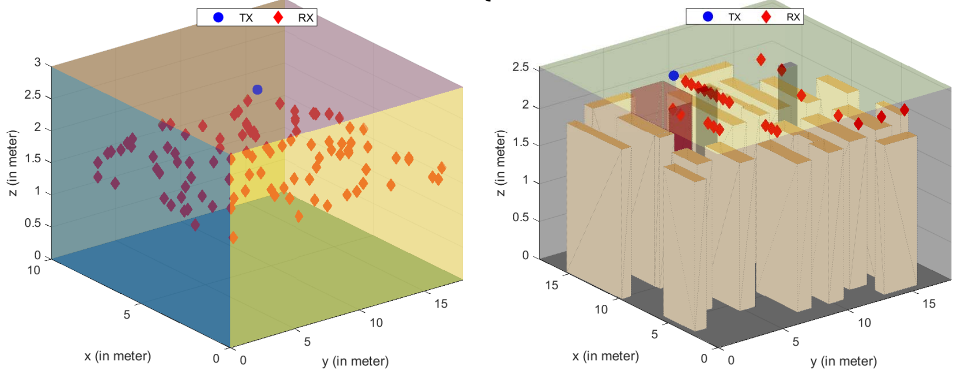


Figure 1: Transceiver diagram of SC PHY mode



(a) Lecture Room (b) Data Center

Figure 2: Lecture room and data center environments

modeling complexity. In our simulation, we use NIST's QD open source software [17] to realize the channel between a transmitter (TX) and a receiver (RX) in two environments, i.e., lecture room of size 10 m × 19 m × 3 m [18] and data center of size 18 m × 18 m × 2.56 m [19], as shown in Fig. 2. In the lecture room, a TX is attached to the ceiling at a height of 2.8 m in the middle of the room and a RX is randomly deployed in the environment at a height of 1.5 m as shown in Fig. 2(a). On the other hand, in the data center, a TX is placed on a rack-top at a height of 2.19 m and a RX is deployed in the environment at a height of 2.19 m as shown in Fig. 2(b). Note that the TX and RX locations in the data center are identical to the ones considered in the measurement campaign [19]. To simulate different channel conditions with random diffuse rays, 100 realizations for each RX location are generated. These propagation rays are then beamformed by the phased array antennas employed at both TX and RX. A 8-by-8 phased array antenna is used in both lecture room and data center environments. Subsequently, a tapped delay line (TDL) fading channel is obtained by sampling the beamformed QD channel based on IEEE 802.11ay SC mode sampling rate, which is used by the PHY simulation. Finally, for a fixed channel realization and a fixed transmit power, a total of 100 packets are transmitted to obtain the average BER value. In order to simulate various SNR conditions, for each channel realization, we vary the transmit power within a fixed range depending on the MCS value [1]. In general, the BER can be described as a function of the TDL fading channel, transmit power, and physical layer algorithms (e.g., MCS).

IV. DNN Based Link-Level Abstraction

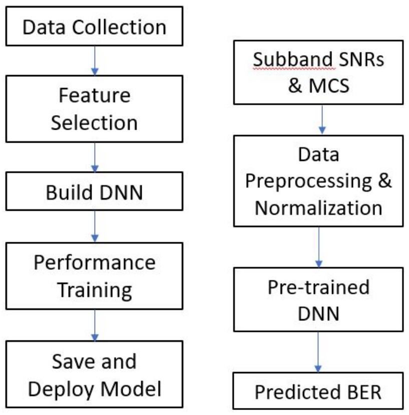
In this section, we propose the DNN based link-level abstraction approach and describe its architecture, training process, prediction process, as well as the model reuse and adaptation in detail.

A. Workflow

Our DNN based approach is motivated by the *Universal Approximation Theorem* [20], which states that a standard feed-forward neural network with an arbitrary number of layers and multiple neurons in each layer can approximate any continuous function to any degree of accuracy when the non-polynomial activation function is used. However, in our case, the output BER may not form a continuous function and could have some discontinuity around zero due to the simulation accuracy (e.g., no arbitrarily small BER can be generated). To address this issue, we only use the collected data with the non-zero target values (i.e., non-zero BER values) for training. Then, by using the Sigmoid function as the activation function at the output layer, the output value will be limited to a number in the range between 0 and 1. In this study, although we focus on predicting the BER value of the communication link, our designed machine learning methodology can be applied to PER prediction as well, with PER value as the training target.

Fig. 3 shows the complete workflow that involves both training and run-time prediction processes.

*1) **Training Process:*** Training process contains five steps: (i) Data collection, (ii) Feature selection, (iii) Building a neural network, (iv) Conducting training process, and (v)



(a) Training (b) Prediction

Figure 3: Machine learning workflow

Storing and deploying model. Each of the steps is described below in detail.

Step 1. Data Collection: This step consists of data collection and data preprocessing. To collect the training data, for a given channel realization, we randomly select a transmit power within a given range and transmit 100 packets to obtain an average BER, which is the training target. To this end, for each BER value, the corresponding TDL channel, transmit power, MCS configuration are saved along with the BER value.

Assuming that the channel is slow fading and the channel is static during a packet transmission, we divide the channel into 512 sub-channels and compute the sub-channel gain by calculating the 512-point Discrete Fourier Transform (DFT) of the TDL channel h_{TDL} as

$$H_{\mathrm{TDL}}(k) = \sum_{l=0}^{L} h_{\mathrm{TDL}}(l) \exp(-j2\pi kl/N), \tag{2}$$

where k is the sub-channel index, l is the tap index, L is the number of taps in the TDL, and N is the number of sub-channels (i.e., 512). Together with the transmit power, the SNR for the kth sub-channel can be computed via,

$$\gamma_k = \frac{P|H_{\mathrm{TDL}}(k)|^2}{N_0}. \tag{3}$$

Here, N_0 is the AWGN power, P is the transmit power. The 512 sub-channel SNR values characterize the channel condition, and can be used as training input.

Step 2. Feature Selection: To characterize the channel between TX and RX, we can use either frequency-domain features such as sub-channel gains as computed in Step 1, as well as the time-domain features such as K factor [3], root-mean-square (RMS) delay spread, maximum delay, and the number of the propagation paths, among others. These time-domain features can be derived directly from the TDL channel h_{TDL}. In this step, we select a set of features to prepare for the training process. Note that as Steps 2, 3, and 4 are iterative processes, we try several sets of training features and select the one giving the best performance. Note that we also compare the performance with or without the time-domain features on top of the frequency-domain features.

As the number of features increases, the DNN architecture becomes more complex and the required training data increases exponentially. In order to maintain a good training performance without requiring a large amount of training data, we first sort the sub-channel SNR values and then sample the SNR value by taking equally spaced elements of the sorted SNR vector.

Step 3. Building Deep Neural Network: We leverage a fully connected multi-layer feed-forward neural network to perform the prediction. The loss function is defined as the mean squared error (MSE) between the predicted error rate and the actual simulated error rate. In terms of activation function, for each hidden layer, the Rectified Linear Unit (ReLU) function is used, and, for the output layer, the Sigmoid function is used because the output is in the range of $[0, 1]$. The optimizing function is using the Adamax algorithm.

Step 4. Perform Training Process: Based on the established DNN with its configuration (e.g., the number of input features, the number of layers, and the number of neurons in each layer), we carry out the training process. We select a minibatch size of 32 and execute the training process (200 epochs as default).

Step 5. Storing and Deploying Model: After the training process is complete, we store and deploy the model. In our study, we build the machine learning model using the Tensorflow and Keras platforms for Python[4]. The model (i.e., training weights and biases) is stored as a model checkpoint at the end of the training. The benefit of a stored checkpoint is that after it is loaded, the model can be continuously trained to improve model accuracy or adapt to a new scenario using additional training data tailored for system dynamics and new environments. We also record the normalization parameters and settings, which is necessary for the model reuse because the same normalization process needs to apply to the new data when we use the pre-trained model.

2) ***Prediction Process:*** Fig. 3(b) illustrates the prediction procedure. Once the channel state information (CSI) and PHY configuration (i.e., MCS) are obtained, data will be

[3] K factor is defined as the ratio of the power of the dominate path to the total power of the remaining paths

[4] Certain commercial equipment, instruments, or materials are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

preprocessed for features extraction. The extracted features should match the training features used to build the model, and these features will go through the same normalization process using stored normalization parameters and settings. The normalized features will go through the deployed model, and BER will be predicted. The prediction can be included in the system-level simulation (e.g., ns-3) to determine whether a packet can be received successfully based on the predicted error rate. The prediction process can also be used for link adaptation.

B. Model Reuse and Adaptation

After we obtain a trained model based on one system, applying it to another system is an important issue, as no two systems or two environments are identical. If the trained model can be reused, it can significantly reduce not only the time taken to collect sufficient data for the training process, but also the time taken to train the model. To address this issue, we design mechanisms for model reuse and adaptation to enable reusing the trained model for different systems as well as for different environments.

Our study considers two different evaluation cases: (i) cross-environment prediction and (ii) cross-system prediction. In the cross-environment prediction case, we assume that identical digital transceiver systems (e.g., hardware) are operating in different environments. More specifically, we consider 802.11ay SC PHY with ideal hardware along with perfect channel knowledge in different environments. Within this case, we investigate whether we can apply one well-trained model to environments with different channel characteristics, but with the same hardware. In the cross-system prediction case, we investigate how to apply the pre-trained model obtained from one system to another system with hardware impairment through continuous learning so that the learning model can be further tuned to improve the prediction accuracy. In the next section, we design experimental scenarios and demonstrate the efficacy of model reuse and adaptation.

V. Performance Evaluation

In this section, we describe the performance evaluation that we used to validate the efficacy of our approach. We present the evaluation methodology, followed by the performance evaluation results.

A. Methodology

We leverage IEEE 802.11ay PHY simulation platforms to collect data for 60 GHz mmWave channel under four different environments: lecture room, data center, large hotel lobby, and open area hotspot. Among these four sets of channel data, data for lecture room and data center environments are generated using open-source NIST QD software [17]. Channel models for large hotel lobby and open area hotspot environments are provided by MATLAB WLAN toolbox[4] The TGay channel data is used to provide more variety in the training data that in turn increases the robustness of the trained model. In particular, we use open area hotspot channel data to evaluate the cross-environment prediction performance of the trained model. In addition to a variety of environments, we consider different antenna sizes, resulting in different channel characteristics, to examine the prediction efficacy of the DNN model. In particular, for open area hotspot, a 4-by-4 phased array antenna is used; for large hotel lobby, a 5-by-5 phased array antenna is adopted, while an 8-by-8 phased array antenna is used for both lecture room and data center environments.

The training is conducted using a laptop with Intel Core i7 with CPU speed at 1.8 GHz and RAM size 12 GB. In our evaluation, we adopt a multi-layer feed-forward neural network to perform the prediction. In this network, the size of the input layer is the number of input features, and the size of the output layer is 1 (i.e., the predicted BER value). We use the ReLu function as the activation function for each hidden layer, and the Sigmoid function in the output layer.

For training and testing, we first partition the data into a dataset with zero BER and a dataset with non-zero BER. In a default setup, $80\,\%$ of the data with non-zero BER is randomly selected for training and the remaining $20\,\%$ non-zero BER data is used for testing. Furthermore, within the $80\,\%$ training data, $20\,\%$ is then used for validation purposes during the training process.

In our evaluation, the DNN model is trained, validated, and tested based on simulation data collected for one MCS (i.e., MCS7). Due to the limited space, all results shown in this section are based on MCS7. Note that modulation order and coding rate have also been used as two additional features to build a general model to predict all MCSs for SC mode. The Loss function used in training is MSE, which is defined as $\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_T(i) - y_P(i))^2$. Here, $y_T(\cdot)$ is the true value, $y_P(\cdot)$ is the predicted value, and N is the total number of samples.

B. Evaluation Results

*1) **Learning accuracy versus feature selection:*** The collected channel raw data (i.e., a multi-tap channel impulse response), along with the transmission power need to be preprocessed. We apply 512-point DFT to convert the time-domain tap-delay channel response to its corresponding frequency response at 512 sub-channels, equally spaced in the band occupied by the signal. Due to the curse of dimensionality, the number of training data required grows exponentially with the number of features. Thus, in our case, we only use a subset of all the 512 sub-channel SNRs. In particular, we first sort the 512 sub-channel SNR values in descending order; next, we sample the sub-channel at different sampling intervals (e.g., every 16, 32, 64, 128 samples), and finally we compare their training and validation performances.

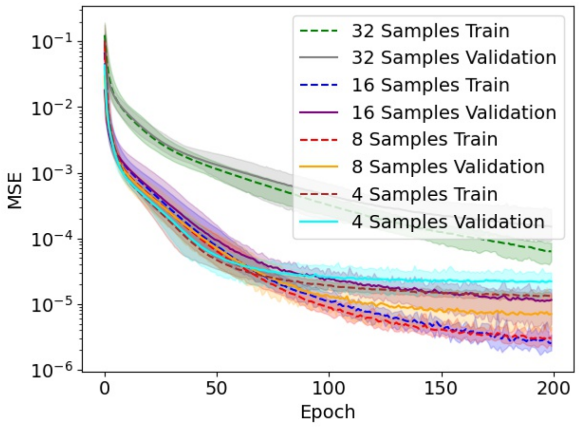


Figure 4: Training accuracy versus number of input sub-channel SNRs with performance range of 10 experiments shown

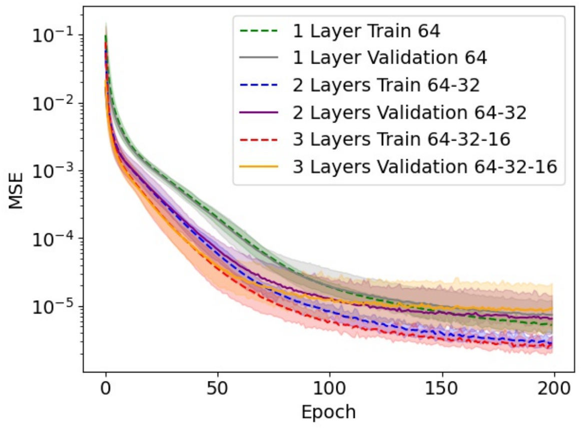


Figure 5: Training accuracy versus number of layers with performance range of 10 experiments shown

Fig. 4 illustrates the training results for the different number of training features. As shown in the figure, the increase in the number of features will not always produce better training results as the number of training data can limit the learning accuracy. In particular, the worse MSE for both training and validation occurs when we use 32 samples. In our case, using 8 samples and 16 samples lead to better results in terms of converging to a smaller MSE value, and 8 samples slightly outperform 16 samples. Thus, in the remaining part of the evaluation, we choose 8 sub-channel SNRs as the input features. Note that, in Fig. 4, the dotted lines are used to demonstrate the average training and validation performance, and the color bands are used to show the performance range of 10 rounds of training.

*2) **Learning accuracy versus DNN structure**:* In this scenario, we evaluate learning performance in terms of the number of hidden layers and the number of neurons at each layer. The training data consists of 3081 samples with non-zero BER out of the total 13 465 collected samples.

Fig. 5 shows the comparison of training results for three DNN structures (i.e., one layer with 64 neurons, two layers with 64 and 32 neurons, and three layers with 64, 32, and 16 neurons, respectively). From the figure, we observe that with the increase of the number of training layers, the training accuracy improves in general. Nonetheless, as the performance improvement from the two-layer structure to the three-layer structure is not significant, we will not go beyond the three-layer structure. Also, considering the three-layered DNN leads to fast convergence performance, in the remaining part of the experiments, we use the three-layered DNN structure with 64, 32, and 16 neurons in the first, second and third layers, respectively.

*3) **Learning accuracy versus number of training data samples**:* In this evaluation, data is collected from all four

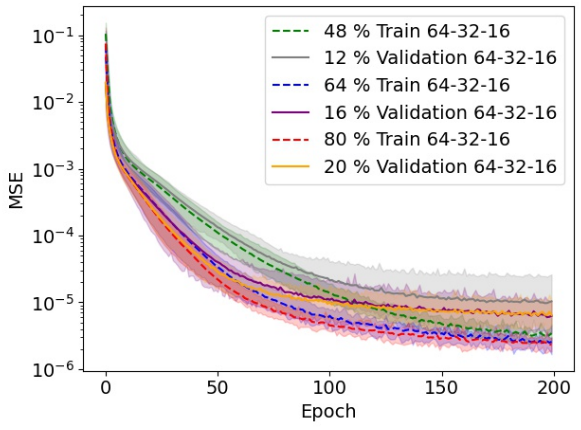


Figure 6: Training accuracy versus number of training samples with performance range of 10 experiments shown

environments: lecture room, open area hotspot, large hotel lobby, and data center. The total number of training samples is 3081, which is split between training and validation. In this experiment, we tend to evaluate how the number of training data samples affects learning accuracy, as shown in Fig. 6. From the figure, we observe that when only 60 % of the data is used to train the DNN, the evaluation curve converges at the higher MSE value than the model trained with more data, such as 80 % or 100 % of the total data. As 80 % and 100 % of the data converge at the same level of MSE of the prediction error, our collected data should be sufficient to train the learning model with this selected DNN structure.

*4) **Cross-environment prediction**:* In this scenario, we evaluate how accurately a learned model from one environment can be applied to another environment with different channel characteristics. To this end, we consider two different environments. One is an indoor environment (i.e., lecture room), and the other is an outdoor environment (i.e., open area hotspot). Fig. 7 illustrates the channel characteristics under these two environments in terms of K-factor and root mean square (RMS) delay spread. In lecture room, we have more channel realizations with stronger dominant path (i.e., larger K-factor) and larger RMS delay spread compared with outdoor open area hotspot due to the high order reflections. We first train a model based on the data collected in the lecture room, and then reuse the same model to predict BER for the open area hotspot environment.

As shown in Fig. 8, the model that has been sufficiently trained in the lecture room can provide good prediction accuracy in the open area hotspot environment, and the prediction MSE is in the order of 10^{-5}. This observation confirms that sub-channel SNR features are capable of capturing the channel characteristics well. It is worth noting that the prediction performance in Fig. 8 is for all channel data, whose corresponding BERs can be either zero or non-zero.

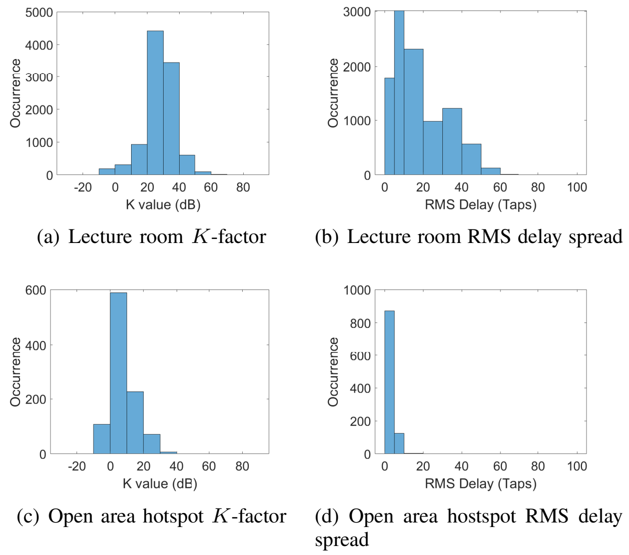

(a) Lecture room K-factor (b) Lecture room RMS delay spread (c) Open area hotspot K-factor (d) Open area hostspot RMS delay spread

Figure 7: Channel characteristics

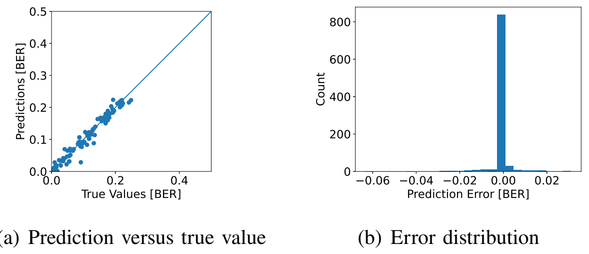

(a) Prediction versus true value (b) Error distribution

Figure 8: Cross-environment prediction

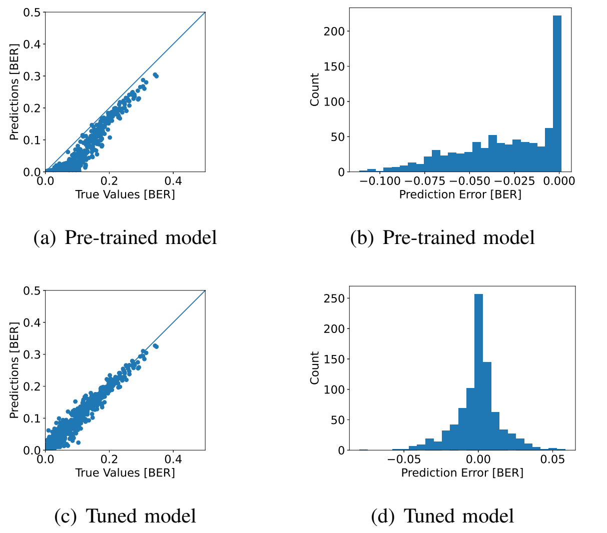

(a) Pre-trained model (b) Pre-trained model (c) Tuned model (d) Tuned model

Figure 9: On-line learning

5) ***Cross-system prediction:*** In this scenario, we evaluate how accurately a learned model from one system can be applied to another system with different hardware. In particular, we investigate the prediction performance when applying the pre-trained model based on ideal hardware to a system with hardware impairments. To emulate such hardware impairment, carrier frequency offset (CFO) of 1.2×10^6 Hz is introduced. As such, the sub-channel SNR is estimated at the receiver from the received pilot signal. As shown in Figs. 9(a) and 9(b), if the learned model from one system is directly applied to another system without further tuning, prediction bias will be observed (i.e., the predicted value is always better than the actual value), since the model is trained using ideal hardware. To overcome this issue, we should deploy continuous learning to further tune the learned model according to the new system. As shown in Figs. 9(c) and 9(d), we observe that after less than 20 epochs, the model is converged, and the prediction bias is corrected. Note that if the training model is not reused, it generally takes a longer time (at least 100 epochs) to achieve the same prediction performance, not even considering the time to collect data. Thus, with the pre-trained model, the convergence time can be significantly reduced.

VI. Summary

In this paper, we have studied link-level abstraction and designed a novel DNN based approach to accurately predict the BER of mmWave communication links. In particular, we have introduced IEEE 802.11ay link-level simulations to collect training data and presented the detailed procedures of learning and prediction processes. We have carried out extensive experiments on the data collected on 60 GHz mmWave channels in four diverse environments and demonstrated the BER prediction accuracy when varying the number of input features, DNN structure, and the amount of training data. We have designed scenarios and conducted experiments to show that the trained model can be reused in different environments and systems. In future work, we plan to extend our DNN based approach to other scenarios and communication systems (e.g., MIMO communications).

References

[1] N. Varshney, J. Zhang, J. Wang, A. Bodi, and N. Golmie, "Link-level abstraction of ieee 802.11ay based on quasi-deterministic channel model from measurements," in *IEEE VTC 2020-Fall*, 2020, pp. 1–7.

[2] M. Mezzavilla, M. Miozzo, M. Rossi, N. Baldo, and M. Zorzi, "A lightweight and accurate link abstraction model for the simulation of lte networks in ns-3," in *Proceedings of the 15th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWiM '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 55–60. [Online]. Available: https://doi.org/10.1145/2387238.2387250

[3] F. Jiang, Y. Xue, B. Jiang, and S. Jin, "Average effective SNR mapping in LTE-A uplink," in *7th International Conference on Communications and Networking in China*, 2012, pp. 403–407.

[4] J. Wang and R. Rouil, *BLER Performance Evaluation of LTE Device-to-Device Communications*. NIST Interagency/Internal Report (NISTIR), National Institute of Standards and Technology, Gaithersburg, MD, 2016-11-09 2016.

[5] Samsung, "3GPP TSG-RAN WG1 meeting #43: Simulation methodology for EUTRA uplink: SC-FDMA and OFDMA," 3GPP, Tech. Rep. R1-051352, Nov. 2005. [Online]. Available: https://www.3gpp.org/ftp/tsg_ran/WG1_RL1/TSGR1_43/Docs/R1-051352.zip

[6] W. Hatcher and W. Yu, "A survey of deep learning: Platforms, applications and emerging research trends," *IEEE Access*, vol. PP, pp. 1–1, 04 2018.

[7] J. Pang, Y. Huang, Z. Xie, Q. Han, and Z. Cai, "Realizing the heterogeneity: A self-organized federated learning framework for iot," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3088–3098, 2021.

[8] A. Carreras Mesa, M. C. Aguayo-Torres, F. J. Martin-Vega, G. Gómez, F. Blanquez-Casado, I. M. Delgado-Luque, and J. Entrambasaguas, "Link abstraction models for multicarrier systems: A logistic regression approach," *International Journal of Communication Systems*, vol. 31, no. 1, p. e3436, 2018, e3436 dac.3436.

[9] R. C. Daniels, C. M. Caramanis, and R. W. Heath, "Adaptation in convolutionally coded mimo-ofdm wireless systems through supervised learning and snr ordering," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 1, pp. 114–126, 2010.

[10] K. Brueninghaus, D. Astely, T. Salzer, S. Visuri, A. Alexiou, S. Karger, and G.-A. Seraji, "Link performance models for system level simulations of broadband radio access systems," in *2005 IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 4. IEEE, 2005, pp. 2306–2311.

[11] E. Chu, J. Yoon, and B. C. Jung, "A novel link-to-system mapping technique based on machine learning for 5G/IoT wireless networks," *Sensors*, vol. 19, no. 5, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/5/1196

[12] G. Ku and J. M. Walsh, "Resource allocation and link adaptation in lte and lte advanced: A tutorial," *IEEE Communications Surveys Tutorials*, vol. 17, no. 3, pp. 1605–1633, 2015.

[13] C. S. Park and S. Park, "Implementation of a fast link rate adaptation algorithm for wlan systems," *Electronics*, vol. 10, no. 1, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/1/91

[14] R. C. Daniels, C. M. Caramanis, and R. W. Heath, "Adaptation in convolutionally coded MIMO-OFDM wireless systems through supervised learning and snr ordering," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 1, pp. 114–126, 2010.

[15] *Part 11: Wireless LAN (MAC) and (PHY) Specifications–Amendment 2: Enhanced Throughput for Operation in License-Exempt Bands Above 45 GHz*, IEEE Std. IEEE P802.11ay/D5.0, 2019.

[16] A. Maltsev, A. Pudeyev, Y. Gagiev *et al.*, "Channel models for IEEE 802.11ay," Tech. Rep. 11-15/1150r9, 2017.

[17] A. Bodi, S. Blandino, N. Varshney, J. Zhang, T. Ropitault, M. Lecci, P. Testolina, J. Wang, C. Lai, and C. Gentile, "A quasi-deterministic (Q-D) channel implementation in MATLAB software," https://github.com/wigig-tools/qd-realization, 2020.

[18] C. Lai, R. Sun, C. Gentile, P. B. Papazian, J. Wang, and J. Senic, "Methodology for multipath-component tracking in millimeter-wave channel modeling," *IEEE Transactions on Antennas and Propagation*, vol. 67, no. 3, pp. 1826–1836, 2019.

[19] C. Gentile, P. B. Papazian, R. Sun, J. Senic, and J. Wang, "Quasi-deterministic channel model parameters for a data center at 60 ghz," *IEEE Antennas and Wireless Propagation Letters*, vol. 17, no. 5, pp. 808–812, 2018.

[20] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," vol. 6, p. 861–867, Jan. 1993.