Toward More Reliable Deep Learning-Based Link Adaptation for WiFi 6

Mostafa Hussien[1,2], Mohammed F. A . Ahmed[2], Ghassan Dahman[2,3], Kim Khoa Nguyen [1], Mohamed Cheriet [1], and Gwenael Poitau[2,3]
[1] École de technologie supérieure (ÉTS), Univeristy of Québec, Canada
[2] Resilient Machine-learning Institute (ReMI), Montréal, Canada
[3] Ultra Electronics TCS, Montréal, Canada

***Abstract*—The problem of selecting the modulation and coding scheme (MCS) that maximizes the system throughput, known as link adaptation, has been investigated extensively, especially for IEEE 802.11 (WiFi) standards. Recently, deep learning has widely been adopted as an efficient solution to this problem. However, in model failure cases, predicting a higher-rate MCS can result in a failed transmission. In this case, retransmission is required, which largely degrades the system throughput. To address this issue, we formulate the adaptive modulation and coding (AMC) problem as a multi-label multi-class classification problem. The proposed formulation allows more control over what the model predicts in failure cases. In this context, we propose a simple, yet powerful, loss function to reduce the number of retransmissions due to higher-rate MCS classification errors. Since wireless channels change significantly due to the surrounding environment, a huge dataset is generated to cover all possible propagation conditions. However, to reduce training complexity, we train the CNN model using part of the dataset. The effect of different subdataset selection criteria on the classification accuracy is studied. It is shown that some criteria for dataset selection consistently behave better than others. To confirm the performance, we applied the proposed model for adapting the IEEE 802.11ax standard in outdoor propagation scenarios. The simulation results show that the proposed loss function reduces up to 50% of retransmissions compared to traditional loss functions. Finally, we propose an optimal subdataset selection criterion.**

***Index Terms*—Link adaptation, IEEE 802.11ax, Machine learning, Deep learning, WiFi 6**

I. Introduction

To accommodate the ever-increasing growth in throughput demand, developing high-performance wireless systems became more essential. These wireless systems should consider both: the unique features of the different data services and the dynamic and spatiotemporal characteristics of the wireless channels. Therefore, techniques like dynamic resource allocation and link adaptation are incorporated into the different wireless standards in order to support the quality of service (QoS) requirements while serving the increased number of users [1]. Link adaptation represents a key element in determining the system's latency and throughput performance [2]. Fortunately, machine learning (ML) is anticipated to provide viable solutions to the link adaptation challenges in wireless systems [3], [4].

In the literature, the link adaptation problem is formulated as a multiclass classification problem where the class labels represent different modulation and coding scheme (MCS) combinations [4]–[8]. According to this formulation, each data point is allowed to belong to only one class and a supervised ML model can be trained to select the ideal MCS based on the training data. However, supervised models generally, and neural networks (NN) specifically, have a certain level of accuracy [9]. In this case, failing to predict the ideal MSC has unpredictable implications on the system throughput. In fact, predicting a higher-rate MCS will result in a failed transmission and, consequently, a retransmission is required which largely degrades the system throughput. These problems come from the fact that formulating the problem as multiclass classification has no control over what the model can predict in the failure cases. Now the question is, if the model failed to predict the optimal MCS, why we do not train it to predict a suboptimal one?

To answer this question, we model the link adaptation problem, for the first time, as a multi-label multi-class classification. In this modeling, a datapoint is allowed to belong to more than one class at the same time (all the successful MCS in AMC problem). Bases on that, the model learns to predict not only the optimal MCS, but also all suboptimal ones. Such modeling approach gives more control to what the model learns from the training phase and what it can predict in the failure cases. However, we need to enforce the model avoid predicting higher-rate MCs that cause the retransmissions. To solve this issue, we propose a new loss function that adds more penalization to these cases. The proposed loss function reduces the number of retransmissions compared to traditional crossentropy loss function, which widely employed in the literature.

In the other hand, Wireless channels vary significantly according to the surrounding environment. To have realistic results, a huge dataset is constructed to cover all possible channel variations. However, it is computationally expensive to utilize all the samples for training. To train the model, a random selection for part of the dataset is an intuitive method in order to guarantee fair representation for the full spectrum of classified/studied cases. In this work, we examine alternative selection criteria other than the random selection. We also compare the effect on the resulting classification accuracy.

The aforementioned selection criteria are based on the

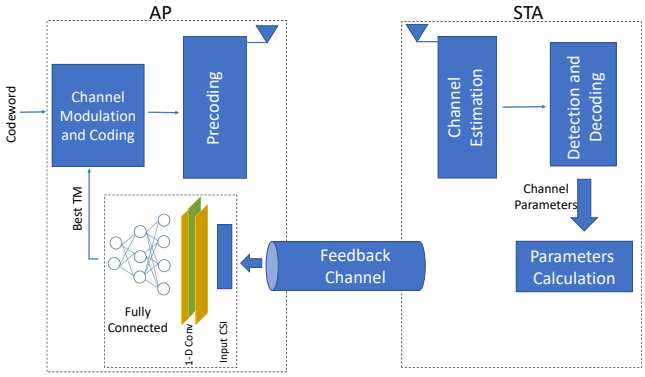

Fig. 1: System Overview.

domain-knowledge and the understanding for the nature of the wireless channels. For orthogonal frequency-division multiplexing (OFDM) based systems, we assume an interference-free, noise-free, single-user, and single-input single-output setup. In this case, the delay dispersion of the channel is the decisive factor on the MCS selection. Hence, instead of randomly selecting the training subdataset, we select the subdataset that comprises a uniform (or as close as possible to a uniform) distribution of the channels delay dispersion behaviors. Given that the *channel dispersion behavior* is not easy to be *fully* characterized, for such selection to take place, we employ well-know criteria characterizing the delay dispersion such as root-mean-square delay spread and window delay spread. We study the effect of the selection criteria on the model accuracy, and the optimal selection criteria is highlighted.

The contributions of this work can be summarized as follows:

- Unlike the literature work, we formulate the problem of AMC as multi-label multi-class classification problem. The model trained to predict all the possible labels for successful transmission (including the optimal MCS and suboptimal ones).
- We employ a convolutional neural network (CNN) with an innovative loss function. The proposed model allows to control what link parameters to use when failing to select the optimal ones. Consequently, it outperforms a CNN with traditional crossentropy function in terms of the retransmission rate.
- We study the impact of training subdataset selection criteria on AMC problem and highlight the corresponding effect in the classification accuracy.

II. Problem Formulation, Dataset Generation, and Training Subdataset Selection

A. Problem Formulation

Assume we have $\mathbf{C}$ different combinations of **MCS** and guard intervals **GI** each of them called a transmission mode (**TM**). The TMs are indexed as $i \in I \subset \mathbb{N}$, where the cardinality of I is the number of available combinations. The index i, thereafter referred to as the class distinctly maps to a combination of **MCS** and **GI**. We adopt the IEEE 802.11ax standard for single-input single-output (SISO) system at 0.8 and 3.2 guard intervals with a fixed bandwidth of 20 MHz. Therefore, in terms of multi-label multi-class classification, link adaptation is the problem of selecting all the class labels, i, to which a certain channel realization belongs. Thus, for a certain channel realization ch_n, the classifier selects all the labels, i, corresponding to all valid transmission modes TM_i. Then, we can express the classifier function as a function F that maps a channel realization ch_n to a set of labels $y \subset \{1, 2, ..., c\}$, where $c \leq C$, such that:

$$F(ch_n) = y = \{i : TX(ch_n, TM_i) = 1\}, \tag{1}$$

where $TX(ch_n, TM_i) = 1$ when transmitting a packet through channel ch_n and with transmission configuration given by TM_i is successful and zero otherwise. From the predicted TMs, we select the one corresponding to the higher data rate. As shown in Fig. 1, a user station (STA) sends the estimated channel state information (CSI) to the access point (AP). The AP then use the received CSI to adapt the transmission link parameters.

B. Datasets Generation

In this work, we selected four scenarios that have diverse delay dispersion characteristics: urban micro-cell, suburban macro-cell, urban macro-cell and rural macro-cell. Using the *Matlab* WINNER II toolbox [10], for each scenario, 50,000 channels are generated. For each channel, using the *Matlab* IEEE 802.11ax toolbox, we simulate transmitting a packet using all TMs and, consequently, the corresponding labels are obtained. We split the generated channels to 80% training and 20% testing.

C. Selection of Training Subdatasets using Different Delay Dispersion Criteria

The training subdatasets will be constructed using two approaches: random selection criteria and different delay-spread-based selection criteria. Based on the random approach, Cases 1 & 2 are identified, and based on the delay-spread approach, Cases 3, 4, & 5 are identified.

1) The random selection criteria (Cases 1 & 2): The random approach is applied in the following two ways:

- Case 1, Random Full Dataset (RandomFD): all data points (i.e., a total of 160,000 data points; 40,000 data point from each of the four scenarios) are used as the training dataset.
- Case 2, Random Partial Dataset (RandomPD): the training subdataset is composed of data points selected randomly and equally from each scenario.

RandomFD represents a reference case where all data points are used for training, and RandomPD is the typical widely-used way of reducing the number of data points through random selection.

2) The delay-spread-based criteria (Cases 3, 4, & 5): The delay-spread-based selection approach is applied to select different training subdatasets each of which has the same number of data points as RandomPD. Instead of being selected in a total random fashion as in the RandomPD, the goal hear is to make the selection such that the data points of the built subdatasets experience the full delay dispersion behaviour of RandomFD in some sense. Using this approach, from the total 160,000 available data points, we select the subdataset points such that the distribution of the delay dispersion metric will be as close as possible to uniform.

Lets assume $RandomFD_i$ to be the ith data point in the RandomFD dataset; $\mathcal{S}(RandomFD_i)$ is its corresponding delay dispersion evaluated based on a specific metric of interest, $\mathcal{S}$; $i = 1, 2, ..., I$ (where I is the total number of data points in RandomFD), and $\min \mathcal{S}(RandomFD)$ & $\max \mathcal{S}(RandomFD)$ are the minimum and maximum obtained delay dispersion values, respectively, among all the data points of RandomFD. We assume the interval $[\min \mathcal{S}(RandomFD)\ ,\ \max \mathcal{S}(RandomFD)]$ to be divided into Z equal disjoint sub-intervals. We define the histogram of $\mathcal{S}(RandomFD)$ as the function that counts the number of delay-spread observations, n_z, that fall into the zth sub-interval, where $z = 1, , 2, ..., Z$, and n_{min} & n_{max} are the minimum and maximum number of observations, respectively, obtained per sub-interval using the full dataset i.e., RandomFD.

Then, our proposed delay-spread-based selection approach can be applied as follows. Select a subdataset from RandomFD such that the selected subdataset has a histogram, m_z, defined as follows.

$$\begin{aligned} \max_{n_{max} \geq x \geq n_{min}} \quad & \sum_{z=1}^{Z} m_z \\ m_z &= min(x, n_z) \\ \text{s.t.} \quad & \sum_{z=1}^{Z} m_z \leq T, \end{aligned} \tag{2}$$

where T is the total number of data points in the selected subdataset.

The value of x determines the maximum number of data points at each of the Z intervals, which results in selecting a subdataset with a histogram that exhibits a tendency toward having a uniform distribution of the delay dispersion behaviour over the $[minFD, maxFD]$ range. The possibilty of ending up with a perfect uniform distribution increases as the number of data points in RandomFD increases.

Based on the applied delay-spread metric (i.e., $\mathcal{S}$), which is our design criterion, we can now define the differences among Case 3, Case 4, and Case 5 of the studied cases.

- Case 3, **r**oot-**m**ean-**s**quare delay spread Partial Dataset (rmsPD). In this case, the training dataset is selected using the delay-spread metric defined as the normalized second-order moment of the delay profile of the channels.
- Case 4, **w**indow (40%) delay spread Partial Dataset (W40%PD). In this case, we characterize the delay dispersion using the delay window parameter which is defined as "the length of the middle portion of the power delay profile containing a certain percentage of the total power found in that impulse response" (p. 4, [11]). Here we use the 40% as our design criterion.
- Case 5, **w**indow (70%) delay spread Partial Dataset (W70%PD). In this case, we use the same definition of the delay dispersion metric as in Case 4; however, here we use the window that contains 70% of the power of the delay profile.

III. Proposed Deep-Learning Approach for AMC

The convolutional neural networks (CNNs) have showed superior performance in different domains including computer vision, natural language processing, speech synthesis, etc [12]. One main advantage of CNNs is its proven capabilities in processing raw data. This advantage eliminates the burdens of data pre-processing. Inspired by this, we propose a CNN-based approach for AMC in IEEE 802.11ax. While all the work in the literature treated the problem as multi-class classification [13], this is the first work to tackle the problem as a multi-label multi-class classification. In this case, each channel realization can belongs to more than one class which means that a packet is successfully transmitted via this channel with more than one TM configurations.

A. CNN Model

The proposed deep convolutional neural network (DCNN) includes convolutional layers, average pooling layers, and fully connected layers. Particularly, the first hidden layer is a convolutional layer with 20 filters. The second hidden layer is a convolutional layer of 32 filters. The following layer is an average pooling layer with pool size of 4. Then, another convolutional layer is added with 64 filters. After that, an average pooling layer with pool size 2 is utilized. The fourth convolutional layer consists of 32 filters. Then, an average pooling layer with pool size 2 is utilized. For the all convolutional layers, every filter has a size of 10 × 2, with ReLU activation, $F(x) = \max(x, 0)$. After the 4 convolutional layers, there are 2 fully connected layers. The first fully connected layer contains 50 and C neurons respectively where C is the number of available TMs. Since one CSI can belong to many classes at the same time, we use *Sigmoid* activation function (3) in the output layer to approximate the multinomial distribution of the class labels. To relieve the effect of overfitting, an $l2$ regularizer is added to the last two layers.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

Adam optimizer [14] is adopted to train the model along with our customized loss function (section IV). The DCNN is trained for 1000 epochs with batch size of 128. After training the DCNN, it is deployed for predicting the appropriate TM.

B. Dataset Description

Consider a labeled dataset consisting of pairs of x and y. In this case, x represents different CSI in different selection cases described in subsection II-C. The label vector y is a vector in $\{0,1\}^C$ where C is the number of the available TMs (i.e., the same as the number of the available classes). If the i^{th} position in the label vector of the j^{th} data instance is set to one, this indicates that a transmission over a channel with CSI equal to j^{th} CSI in the dataset using the i^{th} transmission mode will result in a successful transmission. In the same way, 0 indicates a failed transmission. In our experiments, the label vector is 24^{th} dimensional vector representing the different available combinations of MCS and GI.

C. Evaluation Metrics

To evaluate the proposed model in the context of communication systems efficiency, we applied two system-specific evaluation metrics, namely, data-rate loss (DRL) and number of retransmissions (NR). We define δ as:

$$\delta = R(\overline{TM}_i) - R(TM_i), \tag{4}$$

where $R(\cdot)$ is a function that maps a TM to the data rate associated with this TM, TM_i is the optimal TM given in the dataset, and $\overline{TM}_i$ is the predicted TM. For positive values of δ this means that the model predicts higher-index TM than the optimal one. This implicitly incurs a retransmission. The number of retransmissions is given by NR metric. While negative values of δ implies that the model predicts suboptimal TM, which leads to a rate loss. The difference between the data rates of TM_i and $\overline{TM}_i$ is given by DRL.

IV. Proposed Customized Loss

A. Why we need a customized loss

The traditional loss function used in multi-label multi-class classification problems is crossentropy (5).

$$\mathbf{CE}(y,\hat{y}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i), \tag{5}$$

where C is the total number of classes, which equals to the dimension of y. We can see that the function in (5) treats all wrong predictions equally which is not relevant to the considered AMC problem. We can see that equation (5) pushes the model toward learning the true distribution of class labels. Although this is the ultimate goal of any classification problem, in some cases we aim to stress on certain type of errors (false positives or false negative).

Definition .1. The labels of a class i denoted as Y_i. The positive instances of Y_i denoted as Y_i^+ and the negative instances as Y_i^-. Also, the predicted labels for a class i is $\bar{Y}_i$. The positive instances of $\bar{Y}_i$ denoted as $\bar{Y}_i^+$ and the negative instances as $\bar{Y}_i^-$.

Definition .2. Given a classifier f, the false positive, and false negative are defined as:

$$fp(f) = \sum_{j=0}^{|\bar{Y}_+|} 1 : j \in Y_-$$

$$fn(f) = \sum_{j=0}^{|\bar{Y}_-|} 1 : j \in Y_+$$

In the problem under consideration, a false positives in higher-rate MCS may lead to a retransmission, which is very costly in terms of bandwidth utilization. However, a false negative indicates selecting a lower TM, which it can be tolerated more than a retransmissions. For this reason, we aim to design a loss function that emphasis on the false positive errors more than false negatives.

B. Proposed Loss

We propose a new customized loss function that adds more penalization to false positive predictions. Since the proposed loss function emphasis on false positives, we named it Crossentropy+, $\mathbf{CE}_+$. The new loss given by:

$$\mathbf{CE}_+(y,\hat{y}) = \mathbf{CE}(y,\hat{y}) + \phi(y,\hat{y}), \tag{6}$$

where $\mathbf{CE}(y,\bar{y})$ is the traditional crossentropy given in (5) and $\phi(y,\hat{y})$ is an extra penalization term for the false positive predictions given by:

$$\phi(y,\hat{y}) = \beta \times \sum_{i=0}^{C} (y_i - 1)^2 \times \hat{y}_i, \tag{7}$$

where C is the total number of classes and β is a weight term added to control the credit assigned for the traditional crossentropy term and the newly added term. Setting β to a high value may lead the model to predict $\hat{y} = \{0\}^C$ vector which minimizes the second term and completely ignores the first term. In the other hand, if we set $\beta \leq 1$, the model may ignore it and learns parameters that minimize only the first term of (6). We set $\beta = 1.3$ for all the experiments in this work. However, in the future, we can learn a value for β to meet different QoS requirements (may be different for a WiFi public network or for a 5G URLLC network).

V. Simulation Results

We organize this section into two subsections: the prediction results of the CNN model using the different proposed delay-spread-based subdataset selection criteria, and the improved prediction results achieved when incorporating the proposed loss function.

A. Results of AMC using CNN

To figure out the effect of the training set size, we trained the model with varying set size, namely, 10K, 20K, 30K, 40K, and 50K channels, for each selection criterion. We also consider a larger *RandomFD* dataset. For each training set, we test the model using three different scenarios, namely,

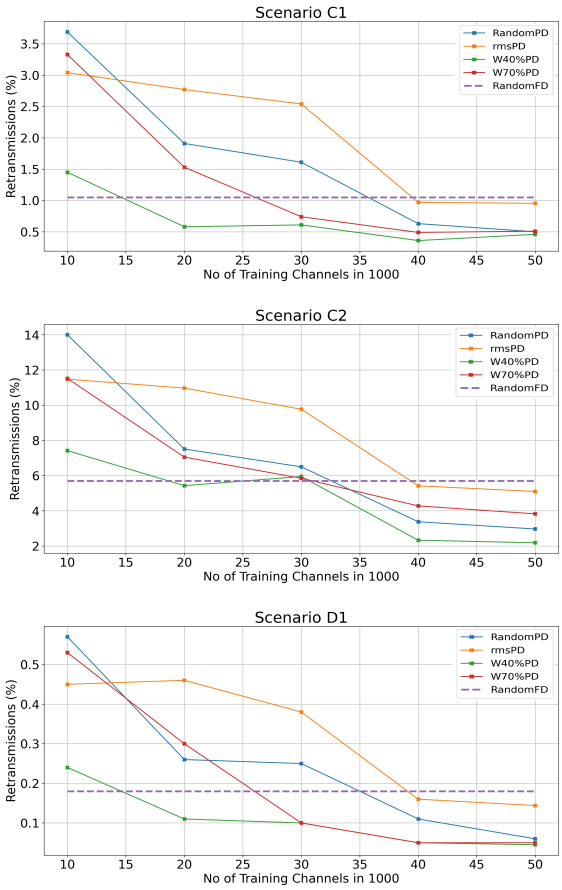

Fig. 2: The percentage of retransmissions in each test-scenario.

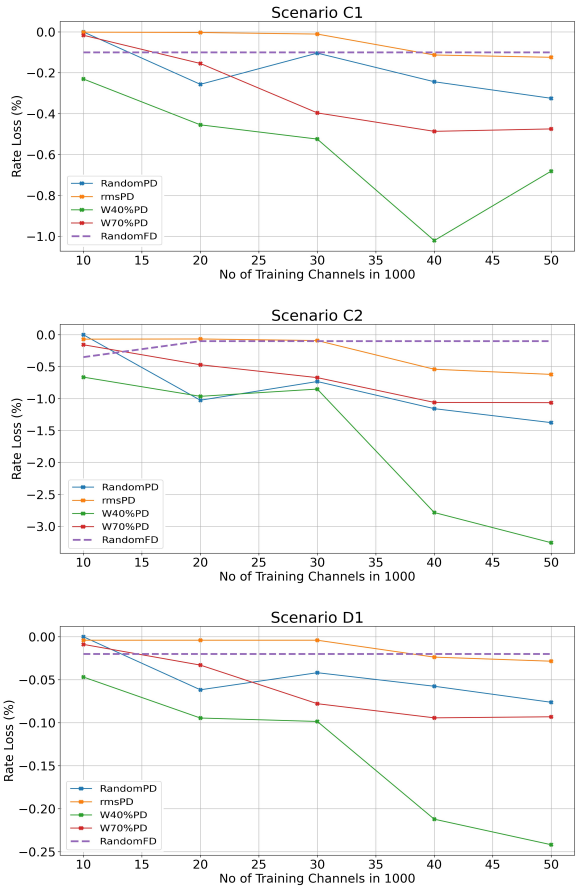

Fig. 3: The percentage of data-rate loss in each test-scenario.

suburban macro-cell (C1), urban macro-cell (C2) and rural macro-cell (D1). The urban micro-cell is not included in the analysis because all studied algorithms achieve almost the same results.

Fig. 2 shows the percentage of retransmissions to the total data points in each test scenario. We can notice that, in terms of the different selection criteria, W40%PD obtains the best performance in all the test scenarios. Also note that for all criteria, scenario D1 obtained higher retransmission rate compared to both C1, and C2. Also, this figure shows that RandomPD and rmsPD training subdatasets always obtain higher retransmission percentage compared to W40%PD and W70%PD. We can also notice that the performance is greatly improves with increasing the size of training dataset. However, a little or no improvement has been recorded with increasing the size from 40K to 50K. Vapnik–Chervonenkis (VC) dimension theorem [15] can explain this saturation behavior. According to VC-dimension theorem, a model keep learning better with more training data points, up to certain number, N_{vc}, after which the model capacity reached a saturation point and adding more data points does not improve the learning anymore.

Fig. 3 shows the percentage of data rate loss due to deploying the CNN-model with different training subdataset selection criteria. Recall from section IV that a data rate loss happens when the model predicts a false negative in the index of the ideal TM. The figure shows an inverse trend between the retransmission rate and the data rate loss. However, it is worth to note that since the overall system performance is decided by both: rate loss and retransmission rate, then it is more likely to tolerate a reasonable rate loss rather than repeated retransmissions. We can see that W40%PD, which results in the best performance in terms of retransmissions, obtained around -3.1% rate loss in the worst case (scenario C2). Based on these observations, we can conclude that training a model based on W40%PD gives the best performance in the retransmission with acceptable rate loss. Also the proposed CNN approach obtained near-optimal TM selection. However, we still can improve the performance of the proposed model by adapting the loss function as we will see in the next subsection.

B. The Performance of the Proposed Loss-Function

To evaluate the performance of the proposed loss function, we trained a model with traditional crossentropy and our proposed loss functions. To obtain fair comparison, we used the same model capacity in the two cases. We also fixed all other hyperparameters (e.g., the same number of epochs, initialization, or optimizer).

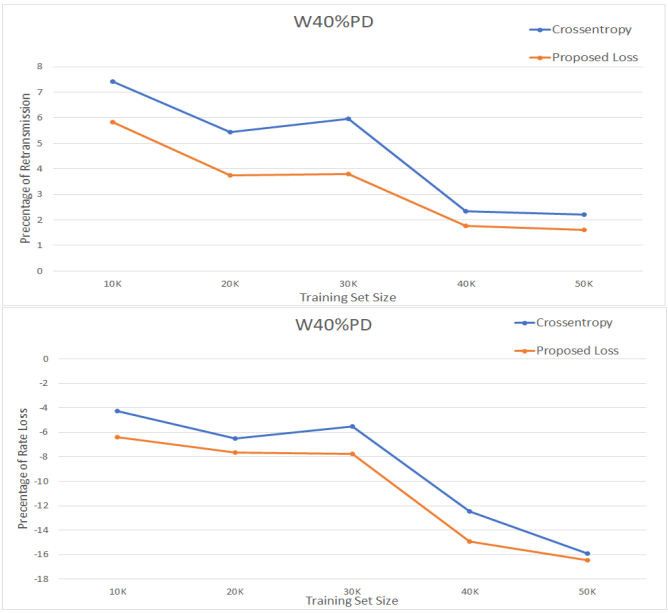

Fig. 4: The percentage of retransmissions and rate loss for W40%PD in scenario C2 for models trained with traditional crossentropy and our proposed loss function (6).

TABLE I: Percentage of retransmission and rate loss for models trained with classical crossentropy loss function (CE) and the proposed loss function (PLoss).

Percentage of Retransmission								
	RandomPD		rmsPD		W40%PD		W70%PD	
	CE	PLoss	CE	PLoss	CE	Ploss	CE	Ploss
10K	14.00	**14.00**	11.48	**10.93**	7.42	**5.84**	11.52	**9.26**
20K	7.51	**6.74**	10.79	**7.47**	5.43	**3.75**	7.05	**6.36**
30K	6.50	**3.72**	9.77	**9.06**	5.95	**3.80**	5.85	**4.40**
40K	3.38	**3.03**	**5.24**	5.26	2.33	**1.76**	4.28	**3.88**
50K	6.63	**3.47**	9.70	**3.57**	2.20	**1.62**	3.98	**3.64**
Percentage of Rate Loss								
10K	0.0	0.0	0.48	0.52	4.29	6.40	1.12	2.82
20K	5.38	6.91	0.51	1.20	6.49	7.68	3.63	4.46
30K	4.76	6.39	0.75	1.06	5.52	7.76	4.69	6.71
40K	7.61	8.81	4.38	4.03	12.46	14.94	7.91	5.75
50K	3.42	5.21	1.01	5.60	15.88	16.46	7.60	8.75

The results of training the model using the two loss functions are shown in Table I. This table shows the number of retransmissions in scenario C2. We selected this test scenario since it has the largest percentage of retransmissions compared with the other scenarios as shown in Fig. 2. We can see that the proposed loss function has largely reduced the number of retransmissions under all selection criteria and dataset sizes. The proposes loss function was capable of obtaining more than 50% proposed improvement over traditional crossentropy in some cases.

Table I also shows the percentage of rate loss for each training set size. We can see that the rate loss using our proposed loss function is larger than using traditional crossentropy. Given that the model capacity is constant, this can be explained by the fact that reducing the false positives may result in increasing the false negatives. However, depending on the specifications of the used communication system (specifically the cost of retransmission compared to rate loss), varying the value of β in (7) provides a wide range of trade-off for performance selection.

VI. Conclusion

A convolutional neural network framework for adaptive modulation and coding (AMC) is presented. The proposed framework is validated for adapting IEEE 802.11ax in outdoor scenarios. We model the problem of AMC, for the first time, as a multi-label multi-class problem to predict the best available transmission modes. We showed that traditional loss functions are limited in solving such problem. We proposed a new loss function that reduces the number of retransmissions while increasing the likelihood of selecting the ideal modulation and coding scheme (MCS). The proposed loss function proved to outperform the traditional crossentropy function. Empirically, we showed that best throughput is obtained by applying a window delay 40% subdataset selection criterion.

References

[1] W. Xu, H. Zhou, H. Wu, F. Lyu, N. Cheng, and X. Shen, "Intelligent link adaptation in 802.11 vehicular networks: Challenges and solutions," *IEEE Communications Standards Magazine*, vol. 3, no. 1, pp. 12–18, 2019.

[2] H. Shariatmadari, Z. Li, M. A. Uusitalo, S. Iraji, and R. Jäntti, "Link adaptation design for ultra-reliable communications," in *ICC*, 2016.

[3] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017.

[4] R. Karmakar, S. Chattopadhyay, and S. Chakraborty, "Intelligent MU-MIMO user selection with dynamic link adaptation in IEEE 802.11 ax," *IEEE Transactions on Wireless Communications*, vol. 18, no. 2, pp. 1155–1165, 2019.

[5] M. Elwekeil, S. Jiang, T. Wang, and S. Zhang, "Deep convolutional neural networks for link adaptations in MIMO-OFDM wireless systems," *IEEE Wireless Communications Letters*, 2018.

[6] Z. Dong, J. Shi, W. Wang, and X. Gao, "Machine learning based link adaptation method for MIMO system," in *PIMRC*, 2018.

[7] L. Li, G. Oikonomou, M. Beach, R. Nejabati, and D. Simeonidou, "An SDN agent-enabled rate adaptation framework for WLAN," in *ICC*, 2019.

[8] F. Blanquez-Casado, M. d. C. A. Torres, and G. Gomez, "Link adaptation mechanisms based on logistic regression modeling," *IEEE Communications Letters*, vol. 23, no. 5, pp. 942–945, 2019.

[9] J. Jagannath, N. Polosky, D. O'Connor, L. N. Theagarajan, B. Sheaffer, S. Foulke, and P. K. Varshney, "Artificial neural network based automatic modulation classification over a software defined radio testbed," in *ICC*, 2018.

[10] Y. d. J. Bultitude and T. Rautiainen, "IST-4-027756 WINNER II D1. 1.2 V1. 2 WINNER II channel models," *EBITG, TUI, UOULU, CU/CRC, NOKIA, Tech. Rep., Tech. Rep*, 2007.

[11] ITUR-R, "Recommendation ITU-R p.1407-6: multipath propagation and parameterization of its characteristics."

[12] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[13] J. Nie, Y. Zhang, Z. He, S. Chen, S. Gong, and W. Zhang, "Deep hierarchical network for automatic modulation classification," *IEEE Access*, vol. 7, pp. 94 604–94 613, 2019.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[15] C. M. Bishop *et al.*, *Neural networks for pattern recognition*. Oxford University Press, 1995.