Towards 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 failure cases, predicting a higher-rate MCS can result in a failed transmission. In this case, a retransmission is required, which largely degrades the system throughput. To address this issue, we model the adaptive modulation and coding (AMC) problem as a multi-label multi-class classification problem. The proposed modeling allows more control over what the model predicts in failure cases. We also design 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 has been 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. The proposed model adapts the IEEE 802.11ax communications standard in outdoor scenarios. The simulation results show the proposed loss function reduces up to 50% of retransmissions compared to traditional loss functions.**

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

I. Introduction

Nowadays, dynamic resource allocation and link adaptation techniques have been incorporated into different wireless standards 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 is anticipated to provide viable solutions to the link adaptation challenges in wireless systems [3].

In the literature, the link adaptation problem has been modeled either as a reinforcement learning problem [4], [5], or as a multiclass classification problem where the class labels represent different modulation and coding scheme (MCS) combinations [6]–[10]. According to this modeling, each data point can belong to a single class and a supervised machine learning model can be trained to select the ideal MCS based on the training data. However, supervised models, generally, have a certain level of accuracy [11]. In this case, failing to predict the ideal MCS 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 modeling the problem as a multiclass classification has no control over what the model can predict in failure cases. Now the question is, if the model failed to predict the optimal MCS, can we 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). Therefore, 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 to avoid predicting higher-rate MCSs that may produce retransmissions. To solve this issue, we propose a new loss function that adds more penalization to such cases. The proposed loss function reduces the number of retransmissions compared to traditional crossentropy loss function, which widely employed in the literature. Fig. 1 shows an overview for the proposed system.

As wireless channels vary significantly according to the surrounding environment, a huge dataset is required to cover all possible channel variations. However, it is computationally expensive to utilize all the samples for training. In this work, we examine different selection criteria for the training dataset. The selection criteria are based on the domain-knowledge and our understanding for the nature of 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.

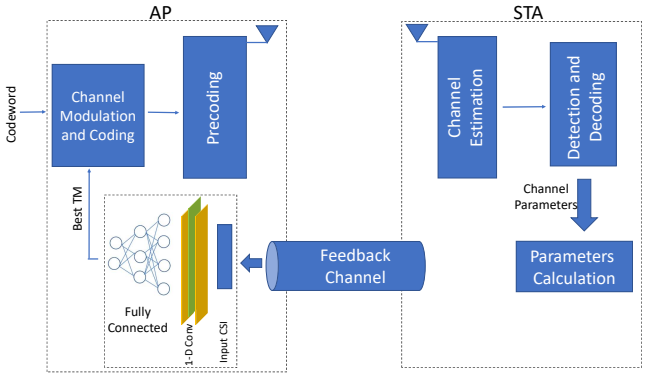

Fig. 1: System Overview.

TABLE I: IEEE 802.11ax bitrate for different single user TMs.

MCS	N_s	Modulation	Coding	20MHz	
				0.8 GI	3.2 GI
0	1	BPSK	1/2	8.6	7.3
1	1	QPSK	1/2	17.2	14.6
2	1	QPSK	3/4	25.8	21.9
3	1	16-QAM	1/2	34.4	29.3
4	1	16-QAM	3/4	51.6	43.9
5	1	64-QAM	2/3	68.8	58.5
6	1	64-QAM	3/4	77.4	65.8
7	1	64-QAM	5/6	86	73.1
8	1	256-QAM	3/4	103.2	87.8
9	1	256-QAM	5/6	114.7	97.5
10	1	1024-QAM	3/4	129	109.7
11	1	1024-QAM	5/6	143.4	121.9

The contributions of this work can be summarized as follows:

- We modeled the problem of AMC as a 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 employed a convolutional neural network (CNN) with an innovative loss function. The proposed model allows to control what transmission parameters combination to predict when failing to predict the optimal one.
- We studied the impact of training subdataset selection criteria on AMC problem and highlighted 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 $\mathbf{MCS}$ and guard intervals, $\mathbf{GI}$, each of them called a transmission mode, ($\mathbf{TM}$). 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 $\mathbf{MCS}$ and $\mathbf{GI}$. We adopt the IEEE 802.11ax standard for single-input single-output system at 0.8 and 3.2 guard intervals with a fixed bandwidth of 20 MHz as shown in table I. 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 $y \subset \{1, 2, \ldots, C\}$ as:

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

where $TX(ch_n, TM_i) = 1$ when transmitting a packet through a channel given by ch_n with transmission configuration given by TM_i is successful, and zero otherwise. From the predicted TMs, we select the TM corresponding to the highest 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 uses the received CSI to adapt the transmission parameters for the next transmission.

B. Datasets Generation

We selected four scenarios with diverse delay dispersion characteristics: urban micro-cell, suburban macro-cell, urban macro-cell and rural macro-cell. Using the *Matlab* WINNER II toolbox [12], for each scenario, 50,000 channels are generated. For each channel, we use the *Matlab* IEEE 802.11ax toolbox to simulate transmitting a packet using all available TMs. We split the generated channels to 80% training and 20% testing.

C. Selection of Training Subdatasets using Different Delay Dispersion Criteria

The training subdatasets are 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 points from each of the four scenarios) are used for training.
- 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. Unlike RandomPD, the data points of the built subdatasets are selected to represent the full delay dispersion behaviour of RandomFD. 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 i^{th} 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 z^{th} 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.

Our proposed delay-spread-based approach to select a subdataset from RandomFD given a histogram, m_z, is 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, [13]). 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 [3]. 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.

A. CNN Model

The proposed deep convolutional neural network (DCNN) includes convolutional layers, average pooling layers, and fully-connected layers. Typically, the first hidden layer is a convolutional layer with 20 filters. The second hidden layer is a convolutional layer of 32 filters, followed by an average pooling layer with pool size of 4. Then, another convolutional layer is added with 64 filters followed by an average pooling layer with pool size of 2. A convolutional layer consisting of 32 filters is added, followed by an average pooling layer with pool size of 2. 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 fully-connected layers contain 50 and C neurons respectively, where C is the number of available TMs. Since one channel can belong to many classes at the same time, we used *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.

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

For training the model, an *Adam* optimizer [14] is adopted 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 TMs.

B. Dataset Description

Consider a labeled dataset consisting of pairs of x and y where 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 number of 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. A positive value of δ means predicting TM with a rate higher than the optimal one. This implicitly incurs a retransmission. The number of retransmissions is given by NR metric. A negative value 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 classifier, in some cases we aim to emphasis on certain type of errors (false positives or false negative).

Definition 1. The label vector of a data instance i denoted as y_i. The set of positive indices in y_i denoted as $y_i^+ = \{j : y_i(j) = 1\}$, the set of negative indices denoted as $y_i^- = \{j : y_i(j) = 0\}$ where $y_i(j)$ is the j^{th} index in the vector y_i and $j \in \{1, 2, \ldots, C\}$.

Definition 2. The predicted label vector for a data instance i is $\hat{y}_i$. The set of predicted positive indices in $\hat{y}_i$ is denoted by $\hat{y}_i^+ = \{j : \hat{y}_i(j) = 1\}$, and the set of predicted negative indices is $\hat{y}_i^- = \{j : \hat{y}_i(j) = 0\}$ where $\hat{y}_i(j)$ is the j^{th} index in the vector $\hat{y}_i$ and $j \in \{1, 2, \ldots, C\}$.

Definition 3. Given a classifier f, the false positive, and false negative are defined as:

$$fp(f) = \sum_{j \in \hat{y}_i^+} 1 : j \in y_i^-$$

$$fn(f) = \sum_{j \in \hat{y}_i^-} 1 : j \in y_i^+$$

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

B. Proposed Loss

We propose a new customized loss function that adds more penalization on 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 false positive predictions given by:

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

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 large 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 than for a 5G URLLC network).

V. Experimental Results

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

A. Results of AMC using DCNN Model

To evaluate 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, suburban macro-cell (C1), urban macro-cell (C2) and rural macro-cell (D1).

Fig. 2 shows the percentage of retransmissions to the total data points in each test scenario. We can see that, among the different selection criteria, W40%PD obtained 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. This figure also shows that RandomPD and rmsPD training subdatasets always obtain higher retransmission percentage compared to W40%PD and W70%PD. We observe that the performance is largely improved with increasing the size of training dataset. However,

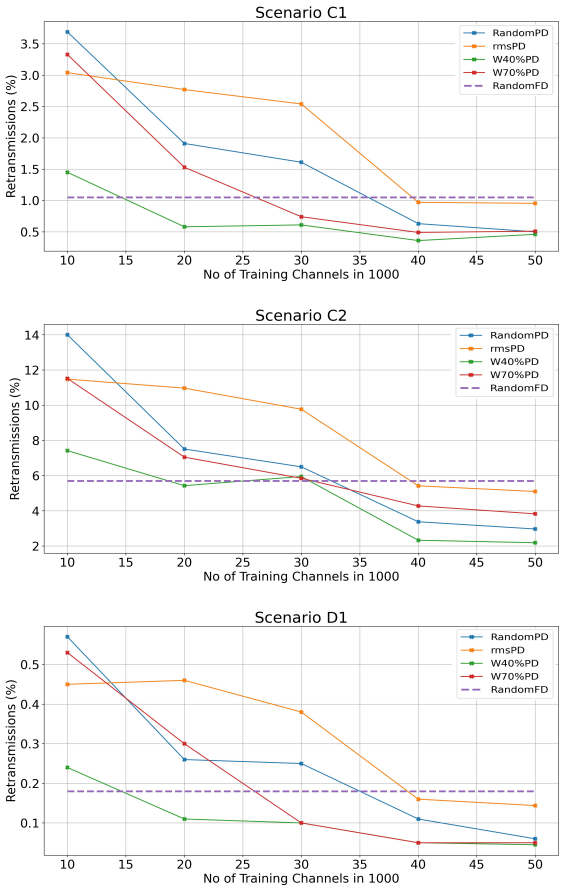

Fig. 2: The percentage of retransmissions in each test-scenario.

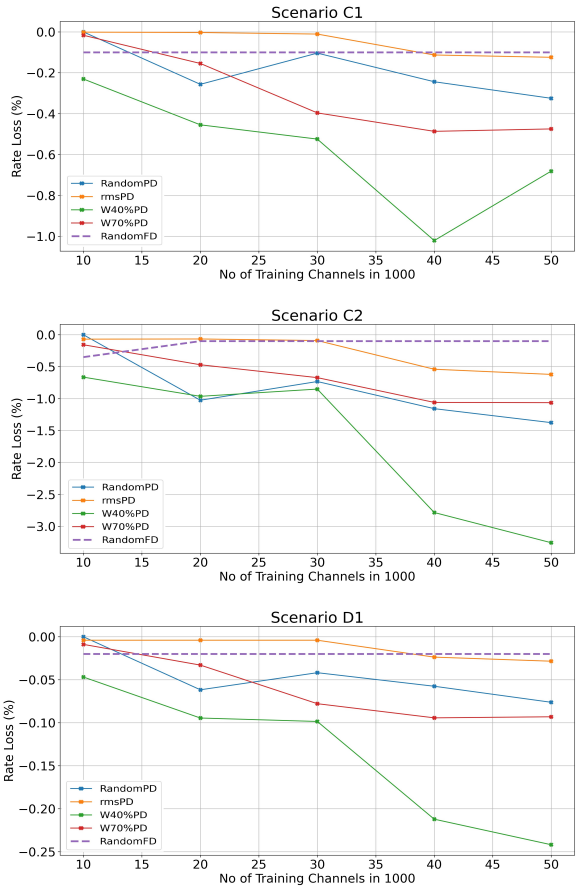

Fig. 3: The percentage of data-rate loss in each test-scenario.

a little or no improvement has been recorded when the size increases from 40K to 50K. According to VC-dimension theorem [15], this saturation happens when the number of training data points reaches a threshold, N_{vc}, after which adding more data points does not improve the learning anymore.

Fig. 3 shows the percentage of data rate loss obtained using the DCNN-model with different training subdataset selection criteria. As explained in section IV, 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 noting that since the overall system performance is decided by both: rate loss and retransmission rate, 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 DCNN approach obtained near-optimal TM selection. However, we can further improve the model performance by introducing the proposed loss function as described 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, activation, regularizer, optimizer, and learning-rate).

The results of training the model using the two loss functions are shown in Table II. The table shows the number of retransmissions in scenario C2. We selected this test scenario since it has the largest percentage of retransmissions compared to 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 obtained more than 50% improvement over traditional crossentropy in some cases.

Table II 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 that of traditional crossentropy. Given

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

	RandomPD		rmsPD		W40%PD		W70%PD	
Percentage of Retransmission								
	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

that the model capacity is the same, this can be explained by the fact that reducing the false positives may result in increased false negatives. However, depending on the specifications of the used communication system (specifically the cost of retransmissions compared to rate loss), varying the value of β in (7) provides a wide range of fine-tuning to meet different performance requirements.

VI. Conclusion

A convolutional neural network framework for adaptive modulation and coding (AMC) in IEEE 802.11ax has been presented. We modeled the problem of AMC as a multi-label multi-class problem. The results showed that traditional loss functions are limited in solving such problem. We proposed a new loss function that increases the reliability of the adaptation framework. The proposed loss function proved to outperform the traditional crossentropy function. We also studied the impact of subdataset selection on the model performance. Empirically, we concluded that window delay 40% subdataset selection criterion along with the proposed loss function give the best throughput/reliability compromise.

Acknowledgment

The authors thank Mitacs and Ciena for supporting this research in the IT13947 grant.

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] V. Saxena, J. Jaldén, J. E. Gonzalez, M. Bengtsson, H. Tullberg, and I. Stoica, "Contextual multi-armed bandits for link adaptation in cellular networks," in *Workshop on Network Meets AI & ML*, 2019.

[5] F. B. Mismar, B. L. Evans, and A. Alkhateeb, "Deep reinforcement learning for 5g networks: Joint beamforming, power control, and interference coordination," *IEEE Transactions on Communications*, vol. 68, no. 3, pp. 1581–1592, 2019.

[6] M. Elwekeil and et al., "Deep convolutional neural networks for link adaptations in MIMO-OFDM wireless systems," *IEEE Wireless Communications Letters*, vol. 8, no. 3, pp. 665–668, 2018.

[7] R. Karmakar and et al., "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.

[8] Z. Dong and et al., "Machine learning based link adaptation method for MIMO system," in *PIMRC*, 2018.

[9] L. Li and et al., "An SDN agent-enabled rate adaptation framework for WLAN," in *ICC*, 2019.

[10] 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.

[11] 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.

[12] 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.

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

[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.