Abstract—Wireless communication networks are conventionally designed in model-based approaches through utilizing performance metrics such as spectral efficiency and bit error rate. However, from the perspectives of wireless service operators, network-level performance metrics such as the 5%-tile user data rate and network capacity are far more important. Unfortunately, it is difficult to mathematically compute such network-level performance metrics in a model-based approach. To cope with this challenge, this work proposes a data-driven machine learning approach to predict these network-level performance metrics by utilizing customized deep neural networks (DNN). More specifically, the proposed approach capitalizes on cross-layer information from both the physical (PHY) layer and the medium access control (MAC) layer to train customized DNNs, which was considered impossible for the conventional model-based approach. Furthermore, a robust training algorithm called weighted co-teaching (WCT) is devised to overcome the noise existing in the network data due to the stochastic nature of the wireless networks. Extensive simulation results show that the proposed approach can accurately predict two network-level performance metrics, namely user average throughput (UAT) and acknowledgment (ACK)/negative acknowledgment (NACK) feedback with great accuracy.

Index Terms—Cross-layer information, machine learning, network-level performance.

I. INTRODUCTION

Conventionally, wireless communication networks are designed based on mathematical models that are established with expert experience. Such models usually focus on one-single network layer, for example, the PHY or MAC layer, as it is considered impossible to develop one model to unify information from multiple layers. For instance, it is rather challenging to mathematically characterize both the average user throughput and the package ACK/NACK rate in one unified mathematical model as these two performance metrics measure two vastly different aspects of the network performance. As a result, most existing models can only measure the so-called link-level performance. However, it is much more desirable to investigate the network-level performance by taking into account available information from all network layers.

More specifically, it is critical for network designers to understand network-level performance such as the network capacity, the average user data rate and the 5%-tile user data rate (the data rate of the worst 5% users). The main challenges in modeling these network-level performance stem from the following difficulties. First, the wireless communication network is highly complex with many components and protocols, which renders the whole system analytically intractable. Second, it is difficult to accurately characterize the channel and user behavior using channel transfer functions, user distribution and motion models. Finally, network events are mostly stochastic such as user arrival and traffic load. For these reasons, two existing approaches have been developed to evaluate network performance in the literature. The first approach is to over-simplify the system mathematical model to approximate the network-level performance. However, the performance of such approximation is far from being satisfactory. Alternatively, the other existing approach is to develop network simulators to predict or model the network-level performance. Such an approach has been widely adopted in the wireless communications industry. Despite the large discrepancy between simulation and field test results, network simulators are still the more preferable choice for studying the network-level performance of a large network. However, the development of network simulators is prohibitively expensive and labor-intensive.

In the meantime, powerful machine learning techniques have been recently developed and successfully applied in many engineering areas such as image and linguistic processing. Built upon the ever-increasing computer power and the availability of Big Data, machine learning techniques are characterized by their data-driven approach that is particularly suitable for the data-rich wireless communication networks. However, to our best knowledge, there are only a few existing works on utilizing data from multiple network layers to understand network behaviors and subsequently optimize network design. In the following sections, we first review these related works before summarizing our main contributions.
A. Related Works and Main Contributions

There were early studies exploiting machine learning in radio resource management (RRM) for wireless networks [1, 2]. For instance, power allocation in multi-user interference channels is a classic NP-hard problem due to the combinatorial nature of the problem. With the goal being maximizing the weighted sum rate (WSR), the traditional convex optimization theory can reach solutions that are close to the global optimum using the iterative algorithm namely, weighted minimum mean-squared error (WMMSE). However, the WMMSE algorithm is of high complexity and thus time-consuming [3]. In [4], a five-layer fully connected neural network is built to learn from the resulting solutions of WMMSE via supervised learning. On this basis, the work in [5] adopts negative WSR as the loss function to train an ensemble deep neural network (CNN) to solve the same problem of power allocation, showing a better performance in high SNR regime (>10 dB). However, there is a big impediment hindering the practical implementation of such DNNs, which is the dynamic number of users. It is shown from a theoretical perspective that the graph neural network (GNN) is a powerful solver to combinatorial problems as it is adaptively scalable according to the number of entities [6]. Thus, by using the a GNN to solve the same power allocation problem, the WSR is increased by more than 2% with respect to WMMSE that always finds a local optimum.

The above power allocation methods require accurate channel state information (CSI). However, in many current communication networks, accurate CSI of each user equipment (UE) may not be available, especially in frequency division duplex (FDD) systems. A more common practice is that each UE feeds back its channel quality indicators (CQI) to the base station, and the base station determines the communication scheme based on the CQI [7]. In [8], a machine learning-based solution is studied that uses only accessible communication overhead data such as CQI on the transmit side. Based on a two-cell model, the study applies reinforcement learning to allocate limited transmit power to 10 UEs working in the same frequency band. The work shows that reinforcement learning is superior to the traditional algorithm in terms of the 5%-tile and median UE data rates. Furthermore, the work [9] shows that it is sufficient to effectively adjust the modulation and coding scheme (MCS) selection dynamically when the base station (BS) only has the UE’s CQI feedback.

Apart from power allocation, a remarkable application of deep learning in the PHY layer is the end-to-end learning-based wireless communication system. The basic idea behind is that a communication system is similar to a neural network in the sense that both systems have input and output. By replacing the encoding/modulation and decoding/demodulation module with DNNs (also known as the autoencoder and autodecoder respectively), the whole system can be automatically optimized by unsupervised learning. Interestingly, it has been observed from [10]–[15] that the trained autoencoder works like a conventional channel coder when the BS has redundant bits to represent the messages. In contrast, the trained autoencoder behaviors like modulation when the BS lacks sufficient bits to describe the messages. However, training the neural network requires that the channel model should be able to represent all non-linear properties of the system while maintaining to be differentiable, which is difficult to achieve in real-life systems. To cope with this problem, [16] proposes to use another CNN to approximate the gradient via supervised learning. Finally, the actor-critic reinforcement learning algorithm has been applied to handle user scheduling and content caching at the same time [17], [18].

Apart from the problems in PHY layer, [19] studies an RL-based resource block (RB) allocation scheduler, which selects the momentarily best scheduler for each transmission time interval (TTI). As the conventional schedulers always focus on some particular key performance indicator (KPI), the RL based scheduler can flexibly choose the best scheduling rule among the conventional schedulers to achieve customized goals. Alternatively, the work [20] use an RL-based framework to adjust the parameters for the proportional fairness (PF) scheduler, which can also allocate RBs better than conventional schedulers. Using a similar methodology, [21] improves the quality of service (QoS) for an unmanned aerial vehicle (UAV)-based immersive live system. In addition, authors in [22] studied a dense small-cell network and proposed to capitalize on deep Q-learning (DQN) to reduce the end-to-end delay.

There are studies concerning other aspects in RRM. For instance, an actor-critic reinforcement learning is utilized to solve the user allocation problem aiming at more energy-efficient strategies [23], while the work [24] considers the user allocation problem from the handover point of view. Also, the authors in [25] effectively reduce the energy consumption in base station sleeping control with a data-driven method. Furthermore, in [26]–[28], data-driven signal recolonization and modulation classification problems were investigated, showing impressive performance when it is compared with model-driven method.

B. Contributions

Most studies discussed above focus on utilizing information from only one network layer, but they neglect to verify whether a network is predictable or not, especially when it comes to the multi-layer architecture. In contrast with above studies, this work considers a DNN structure to predict the network-level performance by exploiting information from both PHY and MAC layers. In [29], we have made some initial attempts to explore the feasibility of such a CNN structure and achieved some preliminary results. In this work, we will rigorously define the average UE throughput specifically designed for our proposed framework. Furthermore, we will extend our investigations to the prediction of UE average throughput as well as the ACK/NACK feedback. Finally, we will provide in-depth elaboration on the applications of such predictions for network parameter fine-tuning. The main contributions of this work are summarized as follows:

- To our best knowledge, this work is the first successful attempt to demonstrate that it is feasible to accurately
predict network-level performance using DNN by exploiting both PHY and MAC information derived from various network counters of vastly different natures and complex network mechanisms such as out loop link adaptation (OLLA) and proportional fairness (PF) user scheduling. Specifically, we design two DNN structures to predict two important network-level performance metrics, namely the UE average throughput (UAT) and the ACK/NACK outcome of a transmission, respectively;

- The performance prediction is highly data-dependent, and the measured real-world data always has very high randomness, and the performance is affected by complicated factors such as scheduling algorithms, user behaviors and so on. However, accurate labels are essential for training DNNs, and the stochastic nature of wireless communication system causes noisy data. We formulate the noisy data cleaning task as a bi-level optimization problem and propose a robust weighted co-teaching algorithm to circumvent the problem;

- Predicting network-level performance is ultimately about improving QoS/QoE for all users. Leveraging the predictive capability of our trained DNNs, we can depict the MCS landscape for any users, and thus guide the decision-making process during MCS selection. This application is utilized as an example to demonstrate the feasibility of better utilization of the network resource to improve users’ QoS by data-driven methodology.

Note that the aim of this work is to validate the feasibility of network-level performance prediction. However, this work can be extended to other applications. For instance, the link-level model is usually over-simplified for the benefit of expediting the system-level simulation, which is known as the link-to-system mapping (L2SM). The mapping is mainly aimed at providing an outcome of a transport block transmission (TB), i.e. whether the TB is successfully received or not [30]. In the literature, there are many reported results studying L2SM from an information theoretic point of view [31]–[35]. Unfortunately, these studies fail to consider cross-layer information as their cross-layer models become analytically intractable. In this work, we devise a new approach to replace the L2SM module for network simulation.

In the sequel, we will first introduce the wireless network simulator settings in Section II while the network data preparation and the network-level performance prediction tasks are elaborated in Section III. After that, two customized DNN structures are developed in Section IV before two training algorithms are proposed in Section V. Finally, extensive simulation results are shown in Section VI followed by the conclusion given in Section VII.

II. WIRELESS COMMUNICATION NETWORK BASICS

We begin with reviewing some basic concepts of a typical communication system that we are dealing with extensively in this work. Specifically, we consider a single-cell wireless communication network with BS operating in the 2 GHz frequency band in the LTE FDD mode. K resource block groups (RBG) are set to serve downlink UEs. Each RBG consists of several RBs. We consider the downlink transmission scenario in which both the BS and UEs are equipped with two antennas. In contrast to most works in the literature that studied the full-buffer transmission, we consider the bursty traffic mode in which each UE has finite amount of traffic request. Next, we will elaborate on four basic transmission mechanisms that are adopted in our network simulator, namely the OLLA for MCS, the transmission block formation, the proportional fairness user scheduling scheme and the Hybrid Automatic Repeat reQuest and Retransmission (HARQ). It will be clear that it is non-trivial to model all these mechanisms mathematically using the model-based approach.

A. Out Loop Link Adaptation (OLLA)

The LTE protocol allows the UE to suggest an appropriate MCS to be used in the next transmission, which is aimed at achieving a pre-defined block error rate (BLER). To propose such a suggestion, the UE actually selects its desirable MCS by sending back a CQI value as a quantized reference. Typically, each CQI representing a signal-to-noise ratio (SNR) interval is periodically measured and reported. Thus, MCSs are indeed selected by mapping the received instantaneous SNR into its interval. The challenge about the MCS selection is that it cannot be either too aggressive (i.e. too high) or too conservative (i.e. too low). A higher-level selected MCS leads to a larger transport block (TB) size while incurring a higher BLER. In the industry, the MCS achieving a BLER of 10% is commonly adopted to maximize the expected TB size while maintaining a high successful transmission rate.

However, this mapping rule is not sufficient to robustly compensate the discrepancy between the chosen MCS and the optimal MCS for different UEs. Note that every UE may have its own preference. For instance, aged devices might require relatively lower MCS for the same given channel conditions due to their limited computational power. To cope with this problem, the OLLA algorithm is designed to enable the BS to adaptively update the CQI value q as follows:

$$\tilde{q} = \left\lfloor q + \alpha \right\rfloor,$$  \quad (1)

where $\left\lfloor \cdot \right\rfloor$ is the rounding operator, and $\tilde{q}$ is the offset CQI used for MCS selection. Furthermore, $\alpha$ is the adjustment coefficient dynamically updated every time when an ACK/NACK flag is fed back. Specifically, let $\hat{R}$ denote the ACK/NACK flag with

$$\hat{R} = \begin{cases} 1, & \text{an ACK received,} \\ 0, & \text{a NACK received.} \end{cases}$$  \quad (2)

The adjustment coefficient is updated by OLLA as:

$$\alpha = \begin{cases} \alpha + s_A, & \hat{R} = 1, \\ \alpha + s_N, & \hat{R} = 0, \end{cases}$$  \quad (3)

where the update rate $s_A > 0$ and $s_N < 0$ are usually customized to achieve a specific BLER level $\theta$. For instance, we can set $s_A = 0.01$ and $s_N = -0.09$ to have $\theta = 10\%$. 


B. Transmission Block (TB) Formation

On the BS side, the CQI reported by UEs is one-to-one mapped to a MCS order with unique spectral efficiency (SE), see Appendix for details. We denote by $\mathcal{M}(\hat{q})$ the mapping function from the offset CQI $\hat{q}$ to its SE. Then, the estimated data rate of the $n$-th UE in the $k$-th RBG can be obtained by

$$R_{n,k} = |\mathcal{G}_n(k)| \cdot \mathcal{M}\left(\frac{1}{|\mathcal{G}_n(k)|} \sum_{\ell \in \mathcal{G}_n(k)} \hat{q}_{n,\ell}\right),$$

(4)

where $\mathcal{G}_n(k)$ is the set of all RBs in the $k$-th RBG measured by the $n$-th UE while $|\cdot|$ stands for the cardinality of the enclosed set. Furthermore, $\ell$ and $\hat{q}_{n,\ell}$ are the RB index and the corresponding offset CQI level, respectively. The BS adopts a default MCS when the CQI is not given, which occurs at the beginning of a transmission. According to the estimated data rates, the BS will then allocate each RBG to an appropriate UE while each UE can have more than one RBG. After the allocation is achieved, the BS will re-calculate an MCS for each UE and subsequently, and determine the size of its transmission block $T_n$ in the current TTI:

$$T_n = |\mathcal{G}_n| \cdot \mathcal{M}\left(\frac{1}{|\mathcal{G}_n|} \sum_{\ell \in \mathcal{G}_n} \hat{q}_{n,\ell}\right),$$

(5)

where $\mathcal{G}_n$ the set of all RBs allocated to the $n$-th UE by the RBG allocation process.

C. Proportional Fairness (PF) User Scheduling

To balance the tradeoff between system throughput and fairness among users, an RBG allocation algorithm called proportional fairness was proposed in [36]. In the algorithm, the BS records a PF value for each UE-RBG pair, each RBG is allocated to the UE with the largest PF value defined as follows. The PF value of the UE-RBG pair $(n, k)$ in the $i$-th transmission time interval (TTI), denoted by $\beta_{n,k}[i]$, is defined as

$$\beta_{n,k}[i] = \frac{R_{n,k}[i]}{T_n[i]},$$

(6)

where $T_n[i]$ is the moving average of the historical throughput and given by

$$T_n[i] = (1 - \gamma)T_n[i - 1] + \gamma T_n[i - 1],$$

(7)

where $\gamma$ is a small moving average coefficient, and $T_n[i - 1]$ is the TB size of the $n$-th UE in the $(i - 1)$-th TTI given by (5).

For presentational simplicity, we omit the TTI index in the sequel. Each RBG will be allocated to the UE with the highest PF value in its list, specifically,

$$n^*(k) = \arg \max_n \beta_{n,k}.$$  

(8)

To avoid allocating more RBGs to a UE than it needs, each time when an RBG is allocated to a UE, the system checks if the UE has obtained enough RBGs to convey all remaining data in its buffer. If so, the UE will be removed from the scheduling list. This usually happens when the UE’s buffer is about to be emptied. On the contrary, if a UE fails to acquire any RB, then its effective TB will become zero in the current TTI. As a consequence, its moving average throughput will decrease, which will increase its priority in future RBG allocation.

D. Hybrid Automatic Repeat Request and Retransmission (HARQ)

Next, we briefly review the HARQ process in the downlink transmission. When the TB for a UE is readily to be transmitted from the BS, the data will be shifted from the buffer to the HARQ buffer. The data will stay in the HARQ buffer until it is completely successfully transmitted or dropped. Usually, eight HARQ processes are prepared for each UE. When a UE has a new transmission task, its HARQ process with the smallest possible index is chosen. Eight TTIs after the data transmission is completed, the BS will receive an ACK/NACK flag from the corresponding UE. In the case of ACK, the HARQ process will terminate as the TB has been successfully transmitted. In contrast, a NACK flag will trigger a retransmission. Since the RBGs used to conduct the initial transmission are delegated to conduct the retransmission, these RBGs will be temporarily unavailable for the next user scheduling. The MCS chosen for the retransmission must remain the same to ensure that the same TB can be reloaded to the delegated RBGs again. After five consecutive failed retransmissions of the same HARQ process, the TB will be dropped, which incurs the so-called packet loss.

Fig. 1 illustrates an example in which UE1 started a transmission using HARQ process 2 in TTI 2. After eight TTIs, the BS received a NACK flag, meaning the transmission was failed. Then, the BS initiated a retransmission in TTI 10 using the same HARQ process and RBGs.

III. OBJECTIVES AND DATA PREPARATION

As aforementioned, the objective of this paper is to pioneer network-level performance prediction in a highly complex wireless communication system. In this section, we introduce the network-performance prediction tasks and elaborate on the data preparation. It is worth mentioning that a proprietary network simulator has been employed to generate the raw data used in this work. However, we believe that the proposed techniques are generally applicable to the field network data as well as the simulated data obtained from off-the-shelf network simulators such like NS3 or OPNET.

A. Feature Collection

The state of a UE can be characterized by various features. Table I lists some key features and their definitions used in this work. Particularly, a feature may have more than one counter, e.g. RB_CQI has 50 counters. In this work, we consider in total 268 counters to describe the state of a UE. All these counters, in practice, are available at the BS. The first nine features in Table I are classified as the PHY information (also known as Layer 1). In contrast, the last 11 features are all MAC information (also known as Layer 2). As discussed in Section I, it is considered technically impossible to fuse
information collected from these two layers using the model-based approach. In this work, we take advantage of DNN to exploit the cross-layer information simultaneously without explicitly modeling the information.

B. Tasks and Dataset Construction

Before defining the network-performance prediction tasks, we first introduce the following terminology used in this work:

Definition I: An active UE of some TTI is a UE with a non-empty buffer in the TTI.

Definition II: A scheduled UE is an active UE who is allocated at least one RBG;

Definition III: A target UE is a scheduled UE whose performance is to be predicted;

Definition IV: A parallel UE is an active UE who is not the target UE;

Definition V: The Network snapshot of some TTI consists of features of all active UEs in the TTI.

Now we summarize the tasks as follows:

Task I: Given a network snapshot, the first task is to predict the UAT for a target UE in the next interval $T$. Clearly, the data rate of a target UE is determined by all the complex transmission mechanisms explained in Section II as well as the states of all active UEs. In particular, since every UE competes for the limited radio resources of the network, the UAT of the target UE heavily depends on the state of the parallel UEs. As a result, UAT prediction has to be performed in the network level, in lieu of link level. Finally, the 5%-tile UAT that measures the network fairness can be derived if all UEs’ UATs are found.

Task II: Given a scheduled UE, the task is to predict its ACK/NACK. At the first glance, this task may appear to be related to the link-level performance as ACK/NACK of the UE is independent of the state of parallel UEs, assuming all UEs are allocated to non-overlapping RBGs. However, the ACK/NACK prediction should greatly benefit from the information on BLER and iBLER. Furthermore, if the multi-cell scenario is considered, then the inter-cell interference surely has major impact on the ACK/NACK outcome of a given TB. Thus, it makes more sense to predict the ACK/NACK result for a UE in the network level.

A network snapshot consists of all active UEs, and we collect all the concerned counters of a UE in a vector $x = [c_1, c_2, \ldots, c_M]^T$, where $M$ is the total number of counters. Thus the network snapshot is formatted as an $N \times M$ matrix denoted by $X = [x_1, x_2, \ldots, x_N]^T$, each row corresponding to an active UE, and each column being the same counter measured from different UEs. In particular, $N$ is set to be large enough to cover the maximum number of active UEs in a
network snapshot in most cases. For the case in which the number of active UEs in a TTI is less than \( N \), virtual UEs are inserted into the snapshot to keep the total number of active users fixed at \( N \). Furthermore, the counters of a virtual UE are set to some default values that make the virtual UE easily distinguishable from the real UEs. In addition the target UE is always placed in the first row namely \( x_0 \) while the parallel UEs are placed below the target UE in a randomized order.

We calculate the estimated UAT for the target UE indexed by \( n \) in TTI \( t_0 \) by

\[
y_n[t_0] = \sum_{i=0}^{t_0+\tau} T_n[i] \cdot \hat{R}_n[i] / \min\{\tau, \Delta t\},
\]

where \( \hat{R}_n[i] \) is the ACK/NACK flag of the \( n \)-th UE in TTI \( i \) as defined in (2). Furthermore, \( \Delta t \) stands for the duration from \( t_0 \) to the moment when the \( n \)-th UE successfully receives all its data and \( \tau \) is a predefined time period. If the UE successfully receives all its data within \( \tau \), then the actual transmission interval should be \( \Delta t \) (see Case 2 in Fig. 2); Otherwise, the UAT is defined with the actual data successfully transmitted over the time interval \( \tau \) (see Case 1 in Fig. 2).

The definition in (9) is motivated by the following observations. A UE of a large data buffer to be transmitted may never be able to receive all its data before the simulation time expires. In the worst case, a UE who suffers from poor channel conditions may never be able to successfully receive any packet (i.e. NACK frequently occurs). Thus, the UAT should not be simply defined as the total data buffer size divided by the simulation time period. In contrast, (9) defines UAT in a much finer resolution using the exact amount of successfully transmitted data divided by the actual time elapsed to complete such successful transmissions. For Task I, we can generate multiple sets of training samples by adjusting \( \tau \) from one simulation run, and the label UAT is calculated by (9).

For Task II, training samples are constructed from every scheduled UE in each TTI and the corresponding ACK/NACK flag per transmission is the training label. For the recent wireless networks employing the multi-antenna technology, a UE can receive up to two TBs in the same TTI. Since the ACK/NACK flag is a binary counter, there are four possible outcomes for two TBs. Using a 2-tuple representation with the first element being the outcome for the first TB and the second element for the second TB, the four possible outcomes namely \( \{(N,N), (A,N), (N,A), (A,A)\} \) with \( N \) and \( A \) being NACK and ACK respectively can be one-hot encoded. For the case in which only one TB is used for a UE, the default return for the second TB is NACK.

### C. Data Preprocessing

Before the samples in the dataset are fed into the DNN, they are pre-processed to improve the DNN convergence behaviors.

1) **Normalization:** The counters collected in a network snapshot are of different natures. For instance, the ACK/NACK flag is binary whereas RSRP is a floating-point number and the CQI value is an integer. Therefore, we propose to normalize the values according to the counter type. The maximum and minimum values of each counter type are first found by inspecting a small portion of the sample dataset. After that, all counter values are normalized by their respective maximum and minimum values to the interval \([0, 1]\).

2) **Virtual UE Padding:** As mentioned, we assume \( N \) active UEs in the system. If the actual number of active UEs in the current TTI is smaller than \( N \), then virtual UEs are inserted into the network snapshot to keep the dimensionality of the DNN input constant. We propose to fill the virtual UEs’ counters with \(-1\) to make them distinguishable from the regular UEs.

3) **UE Shuffling:** The BS usually communicates with multiple UEs in the same TTI. Thus, if we switch the current target UE with another scheduled UE, we will generate a new training sample. Alternatively, we can keep the target UE intact but randomly shuffling the positions of parallel UEs in the snapshot, which can produce more samples of the same target UE. In short, the UE shuffling process enables more efficient usage of the network simulation data.

### IV. NEURAL NETWORK CONFIGURATION

Fig. 3 shows the structure of the DNN for Task I referred to as UATNet that consists of four groups of convolutional layers and one group of fully connected layers. Note that we first use \( 1 \times 5 \) filters to solely extract the features from individual UE, in lieu of the square filters commonly employed in the image processing applications. The fully-connected layers at the end of the DNN are designed to combine the inter-user features. To avoid gradient vanishing or exploding, a batch normalization layer is set before the activation layer with a momentum of 0.99, which is not shown in the figure.

The ACKNet designed for Task II shown in Fig. 4 has a similar structure to UATNet, except that the UATNet takes matrix input while ACKNet takes vector inputs. We use one-hot encoding to represent the four outcomes of the transmission of two TBs.
used to optimize DNN parameters. These noisy labels are particularly detrimental for Task I as UAT is a floating-point number. To cope with this problem, we will have to define robust performance metrics to evaluate the prediction accuracy of our trained UATNet, which is discussed in Section V-B.

For Task II, the problem with the training data is that the four classes occur with very different probabilities as shown in Table III. In other words, we have to deal with the imbalanced data of the four classes, which may cause the ACKNet to prefer some cases to the others. To overcome this problem, we assign weights to each class that is inversely proportional to its sample size.

B. Proposed Weighted Co-Teaching (WCT) Algorithm

In this subsection, we focus on the training algorithm tailored for UATNet to conquer the influence brought by noisy labels. Denote by $X_i$ and $y_i$ the $i$-th observed sample and the corresponding label, respectively. In addition, we use $\theta$ to represent the set of parameters of a DNN. The DNN training tries to find a set of parameters that minimize a loss function, i.e.

$$\min_{\theta} \sum_{i \in \mathcal{D}_{tr}} \mathcal{L}(y_i, \hat{y}_i)$$

(10)

where $\mathcal{D}_{tr}$ is the training set, and $\hat{y}_i = \phi(X_i; \theta)$ is the model prediction for the $i$-th data. Considering the noises, we use the mean absolute percentage error (MAPE) as the loss function, which is defined by

$$\mathcal{L}_{\text{MAPE}}(y_i, \hat{y}_i) = \sum_{i \in \mathcal{D}_{tr}} \frac{|\hat{y}_i - y_i|}{y_i}.$$  \hspace{1cm} (11)

The MAPE loss function defined in (11) scales down the absolute error $|\hat{y}_i - y_i|$ by the magnitude of the label, and more focuses on the relative error. In addition to MAPE, two commonly used loss functions are mean absolute error (MAE) and mean squared error (MSE) loss functions defined as follows:

$$\mathcal{L}_{\text{MAE}}(y_i, \hat{y}_i) = \sum_{i \in \mathcal{D}_{tr}} |\hat{y}_i - y_i|,$$  \hspace{1cm} (12)

$$\mathcal{L}_{\text{MSE}}(y_i, \hat{y}_i) = \sum_{i \in \mathcal{D}_{tr}} (\hat{y}_i - y_i)^2.$$  \hspace{1cm} (13)

MAE and especially MSE are affected by those samples with big absolute errors.
In order to mitigate the negative effects caused by some noisy data, it is necessary to treat each training data with different weights so that the prediction model can focus more on those representative data samples. In other words, we should have a mechanism to identify those “cleaner” data points, and put more attention on them.

In the following, we formulate the data hyper-cleaning task as a bi-level optimization problem, which is given below:

\[
\begin{align*}
\min_{\mathbf{w} \in \mathbb{R}^d} & \quad \ell(\mathbf{w}) := \sum_{i \in \mathcal{D}_{\text{test}}} \mathcal{L}(y_i, \hat{y}_i) = \sum_{i \in \mathcal{D}_{\text{test}}} \mathcal{L}(y_i, \phi(X_i|\mathbf{\theta}^*(\mathbf{w})) \\
\text{s.t.} & \quad \mathbf{\theta}^*(\mathbf{w}) = \arg\min_{\mathbf{\theta} \in \mathcal{M}_j} \sum_{i \in \mathcal{D}_{\text{tr}}} w_i \mathcal{L}(y_i, \phi(X_i|\mathbf{\theta})), \tag{14}
\end{align*}
\]

where \(\mathcal{D}_{\text{test}}\) is the test set and \(\mathbf{w} = \{w_i\}\) is the sample weights. We construct this optimization problem based on the studies [37] and [38] where the authors claim that data should be weighted according to their importance for model training, which could provide better generalization to the neural network model.

To begin the algorithm design, we note that one main intuition provided by the bi-level optimization formulation (14) is that, there will be two entities involved in the training, one determines the weights based on the current model (i.e., the upper layer problem), while the other trains the model based on the fixed set of weights (i.e., the lower layer problem). A rigorous optimization-based method will optimize the weights by taking the gradient \(\nabla \ell(\mathbf{w})\), but such a gradient is difficult to obtain since it will involve \(\nabla \mathbf{\theta}^*(\mathbf{w})\), for which no closed-form solution exists.

There are various studies to find the sample weights. In particular, it has been reported in the literature that an over-trained DNN ends up over-fitting all training samples [39], [40]. Besides, an interesting experiment conducted in [41] shows that in the presence of noisy labels, it is easier for DNNs to find the relationship between the features and labels in the early stage of training before it overfits the data.

Based on this observation, two interesting studies succeeded in handling the noisy label issue in the classification problem. The first study [42] proposed to use a curriculum learning algorithm to capture the patterns and prevent the DNN from tracing the noise in the samples. In particular, a StudentNet that makes the final predictions is trained under the supervision of a pre-trained or pre-defined MentorNet (Fig. 5). To achieve so, the MentorNet must be capable of telling the accurate labels from the noisy ones. The second study [43] proposed a co-teaching algorithm in which two twin DNNs are used and they choose training samples for each other if the predictions meet the labels.

The insight derived from [42], [43] suggests that the samples that have smaller training errors in the early stage have higher value to be learned from, as their labels are more likely to be close to the ground truth. Inspired by this insight, we propose the weighted co-teaching (WCT) algorithm as shown in Algorithm 1. In WCT, we design two UATNets and allow them to exchange their training samples. Rather than simply

---

**Algorithm 1: Weighted Co-Teaching (WCT).**

1. **Initialize:** Fetch the training dataset \(\mathcal{D}_n = \{X_i, y_i\}\), and initialize two identical DNNs with parameters \(\mathbf{\theta}_A^{(0)}, \mathbf{\theta}_B^{(0)}\), and set iteration index \(n = 0\).
2. Train both DNNs with a batch of training samples, and get \(\mathbf{\theta}_A^{(1)}, \mathbf{\theta}_B^{(1)}\) before updating \(n \leftarrow n + 1\).
3. while \(n < N\) do
4. Shuffle the parallel UEs of each \(X_i \in \mathcal{D}_n\).
5. Divide \(\mathcal{D}_n\) into \(M\) equal-size groups with random order.
6. for \(j = 1 : M\) do
7. Get the \(j\)th group \(\mathcal{M}_j\) from \(\mathcal{D}_n\).
8. Use DNN \(\mathbf{\theta}_A^{(n)}\) to make predictions upon \(\mathcal{M}_j\), and get \(\{\hat{y}_i = \phi(X_i, \mathbf{\theta}_A^{(n)})\} \forall X_i \in \mathcal{M}_j\).
9. Compute weights \(\mathbf{w}_A = \{\max\{1 - \frac{\max_{y \in \mathcal{Y}} |y - \hat{y}_i|}{\max_{y \in \mathcal{Y}} |y - \hat{y}_j|}, 0\}\} \forall X_i \in \mathcal{M}_j\).
10. Train \(\mathbf{\theta}_A^{(n+1)} \leftarrow \mathbf{\theta}_A^{(n)}\) with weighted training samples \((\mathcal{M}_j, \mathbf{w}_A)\).
11. Use DNN \(\mathbf{\theta}_B^{(n+1)}\) to make predictions, and get \(\{\hat{y}_i = \phi(X_i, \mathbf{\theta}_B^{(n+1)})\} \forall X_i \in \mathcal{M}_j\).
12. Compute weights \(\mathbf{w}_B = \{\max\{1 - \frac{\max_{y \in \mathcal{Y}} |y - \hat{y}_i|}{\max_{y \in \mathcal{Y}} |y - \hat{y}_j|}, 0\}\} \forall X_i \in \mathcal{M}_j\).
13. Train \(\mathbf{\theta}_A^{(n+1)} \leftarrow \mathbf{\theta}_A^{(n)}\) with weighted training samples \((\mathcal{M}_j, \mathbf{w}_B)\).
14. end for
15. \(n \leftarrow n + 1\).
16. end while

---

Fig. 5. Algorithms against noisy labels: Curriculum Learning (left), Co-teaching (right) for classification problem, and proposed Weighted Co-teaching (WCT) for regression problem.
passing training samples to each other, each UATNet weights the samples according to its weighting rule before sending the samples to the other UATNet. Specifically, the weighting rule is designed to make the impact of the samples consistent with their the accuracy of predictions.

Apart from some metrics like MAPE and MSE, we also consider the following mean relative error percentage (MREP) to measure the accuracy of our predictions, with $S$ being the size of test set:

$$\text{MREP} = \frac{1}{S} \sum_{i=1}^{S} \frac{|\hat{y}_i - y_i|}{\hat{y}_i + y_i}.$$  

MREP will scale the errors to some values between 0 and 1. Furthermore, MREP only represents the relative distance between the label and prediction without assuming that the labels are the ground truth, which differentiates MREP from MAPE.

The complexity of the training algorithm comes from the forward and backpropagation and the number of iterations. The propagation computation mainly relies on the DNN configuration and here we denote the complexity in forward/backpropagation by $C_F$ and $C_B$ mostly due to matrix multiplications. For a given batch with size $S_b$, we need to run forward propagation $S_b$ times but backpropagation once. There are actually two stopping criteria for the training algorithm: i) reaching a fixed number of iterations and ii) when the validation loss has stopped decreasing for the last $N_{stop}$ iterations. For the first stopping criterion, we calculate the complexity as $O(N(S_b C_F + S_b C_B))$, where $S_b$ is the cardinality of the training set. For the second one, it is equivalent to set a small threshold $\varepsilon$ and stop the training when parameter changes is below $\varepsilon$. In such situation, the number of iterations should be at $\log (1/\varepsilon)$ level, and $N_{stop} \propto 1/\varepsilon$, so we compute the complexity as $O(\log N_{stop}(S_b C_F + S_b C_B))$.

VI. SIMULATION RESULTS

In this section, we evaluate the performance of our designed DNNs and training algorithms for both Task I and Task II. In addition, we use MCS selection as an example to demonstrate the potential applications of the proposed DNNs.

The simulated network setup follows the descriptions in Section II. In addition, the simulator randomly initializes the buffer size between 0.5 Kbytes and 3 Mbytes for each UE upon its arrival into the network. In addition, the noise power density is set at $-174$ dBm/Hz while the transmit power for each RB is fixed at 18 dBm. The data are collected from running network simulations for 13 times and each lasts 20 seconds. We use data generated from 10 runs to build our training set of around 400,000 samples. Then, we randomly extract 6,000 samples from each remaining run to form our test set.

A. Task I: UAT Prediction

In this subsection, we focus on the results of UAT predictions where $\tau$ is set to be one second (1000 TTI). First we show the influence of loss functions. In particular we train three UATNets using MSE, MAE and MAPE as the loss function, respectively. Fig. 6 shows the cumulative distribution function (CDF) of the REP performance of the three UATNets, based on the same test set. As shown, MAPE generally has smaller REP values which indicates that it is more resilient to the noise in our data. The conjecture is that MSE and MAE amplify the large noise and entails performance degradation. In the remaining experiments of Task I, we will utilize MAPE as our training loss function.

The training loss of the proposed WCT is depicted in Fig. 7. As two DNNs are initialized and interacted, there are two curves. We uniformly divide the whole training set into ten groups, and let the two DNNs compute the weights of the next training group for each other. The fluctuation of the curves arises from the change of training groups and weights. In our experiment, both UATNets tend to converge after about 15 epochs.

In Fig. 8, sixty randomly selected samples from the test set and their labels are shown, where the solid red line is the prediction made by the proposed UATNet and the dotted grey line is the label extracted from the raw simulated data. Inspection of Fig. 8 suggests that our prediction matches well with the labels. It is observed that the DNN is able to accurately predict the UAT in the presence of many complicated network mechanisms such as OLLA and HARQ that are difficult to model mathematically. Furthermore, the CDF of all labels in the test set and predictions are presented in Fig. 9, which also supports the high accuracy of the predictions in statistics.

Note that even though the dotted line shows the simulation results, it does not stand for the ground truth due to the noisy label problem. That being said, we provide different metrics to measure how close our predictions is from the labels, and compare the performance of different training algorithms.

Table II compares three training algorithms, namely the proposed WCT, the standard basic training (BT) that uses one UATNet, and the Self-Weighting (SW) shown in Algorithm 2 which uses the trainer itself to generate the sample weights.
The results are collected over 50 experiments with random UATNet initializations, wherein each experiment we test the performance derived from the three training algorithms in different performance metrics. The metric “CORR” stands for the Pearson correlation coefficient between the predictions and the labels. The table records the mean value of the metrics except REP_std that is the standard deviation of the 50 REP values. A lower REP_std means that the performance of the resulting UATNet is more stable. As indicated by the table, the proposed WCT algorithm outperforms other two algorithms in terms of MSE, CORR, REP, and REP_std.

### Algorithm 2: Self-Weighting (SW).

1. **Initialize:** Fetch the dataset $D_{tr} = \{X_i, y_i\}$, and initialize a DNN with parameters $\theta^{(0)}$ and set iteration index $n = 0$.
2. Train the DNN with a batch of training samples, and get $\theta^{(1)}$, $n \leftarrow n + 1$.
3. **while** $n < N$ **do**
4. Shuffle the parallel UEs of each $X_i \in D$.
5. Get a batch of training samples $M_n$ from $D$.
6. Use DNN $\theta^{(n)}$ to make predictions upon $M_n$, and get $\{\hat{y}_i = \phi(X_i, \theta^{(n)}) | \forall X_i \in M_n\}$.
7. Compute weights $w = \left\{ \max\left(1 - \frac{y_i - \hat{y}_i}{y_i}, 0\right) \right\}_{i \in M_n}$.
8. Train $\theta^{(n+1)} \leftarrow \theta^{(n)}$ with weighted training samples $(M_n, w)$.
9. $n \leftarrow n + 1$.
10. **end while**

### Task II: ACK/NACK Prediction

Next, we present the results of the task of ACK/NACK prediction. The outcomes of ACK/NACK are highly related to the feature MCS. In order to cover the feature space of MCS, we disable the OLLA function of the simulator and randomly choose an MCS for each UE to build the dataset of Task II.

Recalling that each sample can have maximum two TBs, four classes are possible, namely $\{(N, N), (A, N), (N, A), (A, A)\}$. If only one TB is used, then the default return for the unused TB is NACK. In most cases, UE’s channel condition only allows it to transmit one TB, and random MCS selection mostly leads to a NACK., in our training set is different, which causes the common imbalanced-label problem in classification. To cope with this problem, data of each class is weighted before being fed into the ACKNet, where the weight of each class is inversely proportional to its size.

Fig. 10 shows that the training loss and validation loss decrease and converge after about 15 epochs, where the loss function is the standard categorical cross entropy. Furthermore, The resulting accuracy for both training and validation is above 94% after 40 epochs as shown in Fig. 11. Noth that...
the vibrations in the loss curve occurs when the two DNN exchange training samples and update weights of each sample.

The resulting confusion matrix of our prediction on the test set is shown in Table III. 57.94% of the test samples in the (N, N) class. In each box in Table III, the number on the top shows the total samples in the class while the number below shows the percentage of prediction that falls into the class. For instance, there are 9829 samples whose labels and predictions are both (N, N), occupying 94.24% of all the samples with (N, N) labels. The overall accuracy across the diagonal entries is 95.23%.

C. Application: MCS Selection

In this section, we demonstrate a potential application of Task I, which is that we can use a well-trained UATNet to predict the data rate for any given MCS, to facilitate the MCS selection. We disable the OLLA algorithm and let each UE hold a random MCS till the end of its transmission. Based on this setting, a new dataset can be acquired and a UATNet can be trained via WCT. Then, we randomly pick a sample from the test set, and vary the target UE’s MCS from its minimum value 1 to its maximum value 29 and adjust the TB size accordingly, forming 29 modified copies of the sample. Using the WCT-trained UATNet to predict the UATs upon the modified snapshots gives us the MCS landscapes shown in Fig. 12. The red star on the graphs is the original sample given by the network simulator. Clearly, in these four examples, the stars are mostly on the curves of the predicted MCS landscapes.

In addition, we find that the MCS landscapes drawn by our UATNet are in a “bell shape”. This shape is reasonable as an excessively large MCS incurs frequent NACK while a conservatively small MCS results in under-utilization of the

![Fig. 10. ACKNet loss on training process.](image)

![Fig. 11. ACKNet accuracy on training process.](image)

![Fig. 12. UATNet-predicted MCS landscape.](image)

<table>
<thead>
<tr>
<th>Labels</th>
<th>Predictions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(N, N)</td>
</tr>
<tr>
<td>(N, N)</td>
<td>9829</td>
</tr>
<tr>
<td></td>
<td>94.24%</td>
</tr>
<tr>
<td>(A, N)</td>
<td>149</td>
</tr>
<tr>
<td></td>
<td>2.48%</td>
</tr>
<tr>
<td>(N, A)</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0.49%</td>
</tr>
<tr>
<td>(A, A)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0%</td>
</tr>
</tbody>
</table>
allocated RBG. Both cases incur UAT performance degradation. It is worth emphasizing that Fig. 12 is the first illustration of the MCS landscapes reported in the literature. Empowered with these MCS landscapes, we are capable of designing the optimal MCS for UEs.

Besides UATNet, ACKNet can also predict the UAT from another perspective as it can predict the probability of the outcome of ACK/NACK in terms of different MCS orders. Let $T_n(m)$ denote the effective TB size of the $n$-th UE under a particular MCS of order $m$. It can be estimated by:

$$T_n(m) = P(\bar{\mathcal{R}} = 1 | \text{MCS} = m) \cdot T_n, \quad (16)$$

where $P(\bar{\mathcal{R}} = 1 | \text{MCS} = m)$ is the conditional probability of ACK when an MCS of order $m$ is employed, and $T_n$ is the TB size given in (5). In Fig. 13, we use ACKNet to predict $P(\bar{\mathcal{R}} = 1 | \text{MCS} = m)$ for $m$ from 1 to 29 and substitute them to (16) to obtain the data rate for four UEs.

The red dashed line in Fig. 13 indicates the MCS chosen by the OLLA algorithm in the simulator, which represents the optimal MCS corresponding to the highest expected data rate (with BLER equal to 10%). Fig. 13 confirms that the optimal MCS derived by ACKNet is reasonably close to the actual optimum found by the network simulator, although this estimated rate is a naive approximation for any RBG allocation, assuming that the network states remains the same in the next one second.

Finally, we put the two MCS landscapes predicted by UATNet and ACKNet for the same snapshot together in Fig. 14. The first observation is that the optimal MCS values chosen by the two DNNs are not identical. Interestingly, the optimal MCS chosen by UATNet is smaller than ACKNet. This means that the optimal MCS for maximizing the effective TB size is larger than that for maximizing the UAT. This is because ACKNet only predicts the outcome of a single transmission based on the physical conditions of the UE, while UATNet predicts the longer-term performance considering all the transmission schemes.

Thus, the BS prefers the optimal MCS given by ACKNet, which maximizes the effective TB size for each scheduled UE, and leads to higher network-level throughput. In contrast, a UE may prefer the optimal MCS given by UATNet as the MCS maximizes its individual UAT. Using a smaller MCS, the UE can remain competitive in the next round of PF scheduling according to 6. Clearly, such a UE decision is selfish at the sacrifice of the network performance. Fortunately, UEs in a centralized network cannot choose their own MCS orders.

VII. CONCLUSION

In this paper, we have demonstrated the first DNN capable of predicting the network-level performance of a wireless communication system by exploiting information from both PHY and MAC layers. More specifically, we have proposed two novel DNN structures, UATNet and ACKNet, to predict two network-level performance metrics, namely user average throughput for a target UE and the ACK/NACK feedback of a TB. In particular, a weighted curriculum training (WCT) algorithm has been developed to alleviate the impact of noisy labels. Extensive results have confirmed that UATNet can accurately predict the resulting UAT while ACKNet can achieve an impressive accuracy rate of 95%. Finally, we have demonstrated that the newly proposed UATNet and ACKNet can be utilized to find the optimal MCS value by computing the MCS landscapes for a given UE.

Source code and simulation data used in this work are available on Github at https://github.com/LSCSC/Network-level-Performance-Prediction.
REFERENCES


Qi Cao received the bachelor’s degree in electronic communication engineering from the University of Liverpool, Liverpool, U.K., in 2013, the M.S. degree in communications and signal processing from Imperial College London, London, U.K., in 2014, and the Ph.D. degree from the School of Information and Control Engineering of China University of Mining and Technology, Xuzhou, China, in 2018. He joined the Chinese University, Hong Kong, Shenzhen as a Postdoctoral Research Associate in 2019. His research interests include MIMO wireless communications, machine learning and artificial intelligent (AI)-driven network optimization.

Man-On Pun (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 2006. He was a Postdoctoral Research Associate with Princeton University, Princeton, NJ, USA, from 2006 to 2008. He is currently an Associate Professor with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen (CUHKSZ), China. Prior to joining CUHKSZ in 2015, he held research positions with Huawei (USA), Mitsubishi Electric Research Labs (MERL) in Boston and Sony in Tokyo, Japan. His research interests include AI Internet of Things (AIoT) and applications of machine learning in communications and satellite remote sensing. Prof. Pan was the recipient of best paper awards from IEEE VTC’06 Fall, IEEE ICC’08 and IEEE Infocom’09. He was an Associate Editor for IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS in 2010–2014. He is the founding chair of the IEEE Joint SPS-ComSoc Chapter, Shenzhen.

Yi Chen received the B.S. degree in communication engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2007 and the Ph.D. degree in information engineering from The Chinese University of Hong Kong, Hong Kong, in 2012. She is currently a Research Assistant Professor with the School of Science and Engineering, the Chinese University of Hong Kong, Shenzhen, Hong Kong. She is also a Research Scientist with the Shenzhen Research Institute of Big Data. Her research interests include wireless communication, resource allocation, and machine learning.