DELUXE: A DL-Based Link Adaptation for URLLC/eMBB Multiplexing in 5G NR

Yan Huang, *Member, IEEE*, Y. Thomas Hou, *Fellow, IEEE*, and Wenjing Lou, *Fellow, IEEE*

***Abstract*— Ultra-Reliable and Low Latency Communications (URLLC) is an important use case in 5G NR that targets at 1-ms level delay sensitive applications. For fast transmission of URLLC traffic, a promising mechanism is to multiplex URLLC traffic into a channel occupied by enhanced Mobile BroadBand (eMBB) service through preemptive puncturing. Although preemptive puncturing can offer transmission resource to URLLC on demand, it will adversely affect throughput and link reliability performance of eMBB service. To mitigate such an adverse impact, a possible approach is to employ link adaptation (LA) through modulation and coding scheme (MCS) selection for eMBB users. In this paper, we study the problem of maximizing eMBB throughput through MCS selection while ensuring link reliability requirement for eMBB users. We present DELUXE – the first successful design and implementation based on deep learning to address this problem. DELUXE involves a novel mapping method to compress high-dimensional eMBB transmission information into a low-dimensional representation with minimal information loss, a learning method to learn and predict the block-error rate (BLER) under each MCS, and a fast calibration method to compensate errors in BLER predictions. For proof of concept, we implemented DELUXE through a link-level 5G NR simulator with GPU and MathWorks 5G toolbox. Through extensive experiments, we show that DELUXE can successfully choose MCS for eMBB transmissions to maintain the desired link reliability while striving for spectral efficiency. In addition, our implementation can meet the real-time requirement ($<$125 μs) in 5G NR.**

***Index Terms*— 5G NR, eMBB, URLLC, multiplexing, preemptive puncturing, link adaptation, link reliability, MCS selection, deep learning, GPU.**

I. Introduction

5G NR has been developed with the objective of supporting services types with extremely diverse performance requirements under a unified radio interface [2]. Among the service types defined in NR, enhanced Mobile Broad-Band (eMBB) and Ultra-Reliable and Low Latency Communications (URLLC) are two important categories that target at drastically different applications. While eMBB is expected to offer extremely high throughput for human-oriented services (e.g., 4K/8K video streaming and train/airplane communications) [3], URLLC has the goal of achieving an end-to-end latency on 1 ms time scale with a guarantee of ultra-reliability (e.g., with a 99.999% packet success probability) to meet the requirements of mission-critical applications (e.g., industrial IoT and autonomous driving) [4].

With such drastic differences in service requirements, eMBB and URLLC data need to be scheduled and transmitted on very different time scales. An eMBB transmission time interval (TTI) may span one or multiple time slots to achieve a high spectral efficiency, where a time slot consists of 14 OFDM symbols [5]. In contrast, a URLLC packet will be transmitted using a mini-slot in order to meet its latency requirement, where a mini-slot may contain as few as 2 OFDM symbols [6]. As a result, when eMBB and URLLC are dynamically multiplexed on the same channel, the allocation of radio resources becomes very complicated.

Since URLLC traffic is sporadic in nature, it would be inefficient to statically reserve a portion of the channel bandwidth for URLLC transmissions [18]. A more efficient approach is to allow URLLC packets to preempt ("puncture") resources from on-going eMBB transmissions. This novel mechanism, termed "preemptive puncturing", was proposed in 3GPP standards body [7]. With preemptive puncturing, each arrived URLLC packet is immediately scheduled for transmission for the following mini-slot using resources that have already been allocated to eMBB traffic. In the meantime, scheduled eMBB transmissions in those mini-slots will cease. An illustration of this mechanism is given in Fig. 1. For the portions of the eMBB transport blocks (TB) that are corrupted by the URLLC traffic, an indication signal is sent to each eMBB user to specify the time and frequency positions of the punctured resources [17]. This information will be used by eMBB users to improve the LDPC decoding (e.g., for flushing out corrupted data bits).

Although preemptive puncturing can effectively reduce the latency of URLLC, it will lead to a performance degradation for eMBB services in terms of data throughput and link reliability. Here, reliability is measured as the proportion of packets that are successfully decoded on an eMBB link. Specifically, when one or more URLLC packets were puncturing an eMBB transmission, the block-error rate (BLER) at the eMBB receiver increases and the decoding of the received

Manuscript received June 16, 2021; revised September 3, 2021; accepted October 14, 2021. Date of publication November 8, 2021; date of current version December 17, 2021. This research was supported in part by NSF under grant CNS-1617634, Virginia Commonwealth Cyber Initiative (CCI), Virginia Tech Institute for Critical Technology and Applied Science (ICTAS), and the NVIDIA AI Lab (NVAIL) in Santa Clara, CA for its unrestricted gift and equipment donation. An earlier version of this paper was presented at IEEE INFOCOM (Virtual Conference) in May 2021 [1] [DOI: 10.1109/INFOCOM42981.2021.9488790]. *(Corresponding author: Y. Thomas Hou.)*

Yan Huang is with NVIDIA Corporation, Santa Clara, CA 95051 USA (e-mail: huangyan@vt.edu).

Y. Thomas Hou is with the Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061 USA (e-mail: thou@vt.edu).

Wenjing Lou is with the Department of Computer Science, Virginia Tech, Fall Church, VA 22043 USA (e-mail: wjlou@vt.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2021.3126084.

Digital Object Identifier 10.1109/JSAC.2021.3126084

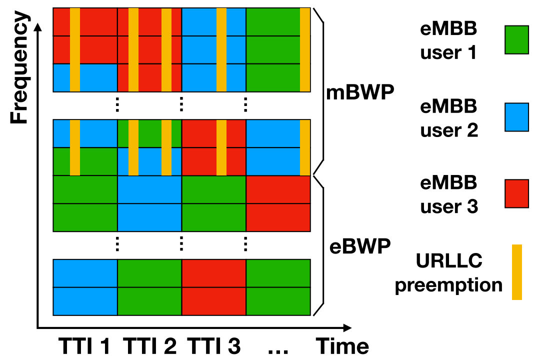
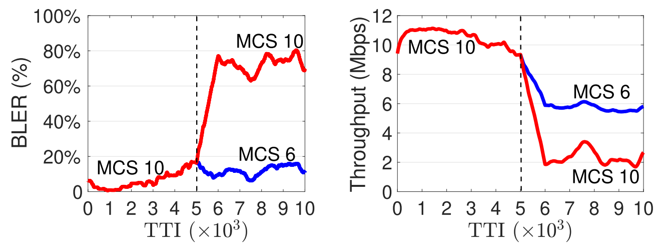


Fig. 1. An illustration of URLLC preemptive puncturing.



(a) BLER (b) Throughput

Fig. 2. BLER and throughput performance of an eMBB link before and after the occurrence of URLLC puncturing under adaptive and non-adaptive MCS configurations.

transmission is more likely to fail. Such an increase of BLER means a deteriorated link reliability (i.e., 1-BLER) and will cause a loss of the achieved eMBB data throughput since each transmission has a higher probability to fail. To mitigate this adverse impact, a promising approach is to design a link adaptation (LA) mechanism for eMBB that is robust under URLLC puncturing. By LA, it means the dynamic adjustment of transmission parameters such as modulation type and code rate based on the instantaneous channel quality and interference on the eMBB link [9].

In this paper, we explore this approach and focus on achieving LA through dynamic selection of modulation and coding scheme (MCS) [16] for each eMBB transmission corresponding to the given channel conditions and URLLC puncturing on its allocated resources. Specifically, there are 29 MCS levels defined in 5G standard [16], with a higher MCS level corresponding to a higher spectral efficiency (bits/s/Hz) but requiring a better channel quality. The problem we will study is to select one among the 29 MCSs for each eMBB transmission so as to maximize eMBB data throughput while guaranteeing the link reliability under the impact of URLLC puncturing.

A. A Motivating Example

To understand how URLLC puncturing impacts the performance of eMBB and how LA can help mitigate such adverse impact, we conduct a link-level simulation study as follows. Consider an eMBB communication link between a base station (BS) and an eMBB user (see Section IX for parameter settings). In each TTI, we allocate a random number of resource blocks (RB) on the channel to this link to mimic RB allocation in a multi-user 5G cell. In Fig. 2, we show the BLER and throughput performance of this link under adaptive and non-adaptive MCS configurations before and after the occurrence of URLLC puncturing. Specifically, before the 5,000-th TTI, there is no URLLC traffic and the MCS is set to level 10 (QPSK modulation and a 0.6631 LDPC code rate). As shown in Fig. 2a and 2b, the average BLER of the link is 6.2% and the average eMBB throughput is 10.41 Mbps before the 5,000-th TTI. Starting from the 5,000-th TTI, we generate URLLC traffic following a Poisson process with an arrival rate of 3 packets/TTI. After the 5,000-th TTI, under a non-adaptive MCS configuration (still using MCS 10 for this link), the average BLER increases significantly to 67.6% and the average throughput drops to 2.36 Mbps. However, if an adaptive MCS configuration is employed, e.g., MCS is adjusted to level 6 (QPSK modulation and a 0.3701 code rate) after the 5,000-th TTI, the average BLER becomes 12.0% and the average eMBB throughput is 5.66 Mbps. Comparing the results in Fig. 2, it is clear that MCS selection is critical to mitigate the impact of URLLC puncturing on eMBB performance.

Although URLLC's adverse impact on eMBB is similar to the interference between communication links in 4G LTE, existing LA methods designed for LTE cannot be applied to address this problem. This is because the behavior of URLLC puncturing in 5G is fundamentally different from that of the interference effects in LTE networks (e.g., the inter-cell interference). Specifically, in LTE, LA is implemented as the adaptive modulation and coding (AMC) component [8], [9]. Under AMC, the MCS for a transmission in a TTI is selected based on the user's channel quality indicator (CQI) report in the prior TTI. Since there is a strong temporal correlation of channel conditions across the TTIs (within the channel coherence time), CQI report for the previous TTI can still be quite useful for the current TTI [8]. But this is hardly true in the case of URLLC's interference to eMBB, as the arrival of URLLC traffic in each TTI is sporadic and lack of correlation. As a result, CQI report on URLLC puncturing for the previous TTI is hardly useful for the current TTI. In other words, LA approaches based on the correlation of channel conditions in consecutive TTIs are no longer useful for the eMBB MCS selection (LA) in the presence of URLLC puncturing. Thus our first objective is to develop a new LA approach that can address the impact of sporadic URLLC puncturing.

In addition to the inability to address the impact from URLLC puncturing, there also exists some well-known limitation with existing CQI-based LA methods. To see this, it is important to understand how CQI methods work. In each TTI, CQI is obtained by reducing a high-dimensional description of the channel conditions on a large number of sub-carriers (SC) to a single metric through some non-linear mapping function, e.g., sum of exponential functions in the well-known exponential effective SNR mapping (EESM) method [10]–[12]. Such a mapping approach, though being simple, suffers from a significant loss of the channel condition information, which leads to performance issues in MCS selection and BLER guarantee [11]–[13]. It has been shown that CQI-based LA may result in over an order of magnitude variation of BLER under different channel realizations [11], [12]. For this reason, our new LA approach will be no longer based on CQI.

B. Main Contributions

The objectives of this paper are twofold. First and foremost, we aim to design an LA method that can maximize eMBB's throughput in the presence of URLLC puncturing, while guaranteeing a link's target BLER for reliability.[1] Second, in the design of this new LA scheme, we want to go beyond the existing CQI methods when handling the information of channel conditions and URLLC traffic. To achieve these objectives, we present DELUXE – a deep-learning-based (DE) link adaptation design (L) for URLLC's multiplexing with eMBB (UXE). The basic idea behind DELUXE is to exploit deep learning (DL) to make reliable and efficient MCS selection based on *both* the *short-term channel condition* information that has strong correlation within channel coherence time, *and* the *long-term URLLC traffic behavior* which does not exhibit any short-term correlation. Specifically, DELUXE uses a deep function approximator (DFA) to make better utilization of information regarding channel conditions and URLLC puncturing than the simple mapping functions employed by the CQI methods.

Although the use of DL for LA and MCS selection in wireless networks has been investigated in previous works (see, e.g., [14], [15]), none of them has addressed the impact of URLLC puncturing on MCS selection. Further, due to the uniqueness of the problem in this paper, none of the existing solutions can be applied to address the problem in this paper.

The main contributions of this paper are summarized as follows:

- DELUXE presents the first successful LA design that can achieve reliable and efficient eMBB transmission in the presence of URLLC puncturing. In each TTI, DELUXE finds the highest possible MCS level for each scheduled eMBB transmission while guaranteeing a given target BLER.
- DELUXE is also the first LA design for dynamic eMBB/URLLC multiplexing that exploits DL. Through the use of a DFA, DELUXE makes MCS selection for eMBB based on both the short-term channel condition information and the long-term URLLC traffic behavior. This capability mitigates the limitation of the existing CQI-based LA methods, which select MCS only based on the short-term channel conditions.
- We propose a mapping to translate the high-dimensional information of channel condition and URLLC puncturing on an eMBB transmission into a low-dimensional representation. Our method can effectively minimize the information loss caused by such translation. This mapping significantly reduces the dimension of input information and allows us to simplify the DFA's multi-layer structure and reduce its execution time (for forward propagation).

[1]One reason for enforcing a target BLER is to avoid too frequent re-transmissions of eMBB TBs which will result in deteriorated latency performance for eMBB services [9].

- We propose a learning method that allows a DFA to learn and predict the BLERs of an eMBB transmission under each MCS level. With this method, the DFA can be learned based on the decoding results from a large number of eMBB transmission samples, without the need to determine the actual BLER of each transmission sample. Moreover, the learned DFA can simultaneously predict the BLERs for all MCS levels through a single forward propagation. To mitigate the potential errors in the BLER predictions, we design a calibration mechanism to make a final tuning for the MCS selection by taking into consideration the recent decoding result history of each eMBB user.
- To ensure the total execution time can meet the real-time requirement in NR (under 100 μs), we employ a hybrid CPU and GPU based architecture to implement DELUXE. In our implementation, the MCS selections for all eMBB transmissions scheduled in a TTI are performed in parallel through efficient matrix operations and single-instruction multiple-data (SIMD) processing.
- We evaluate the performance of DELUXE through an extensive link-level simulation study under practical network settings. Our results show that DELUXE can effectively maintain the target BLER for eMBB under the impact of URLLC puncturing while the state-of-the-art EESM method fails to do so (by a wide margin). Moreover, DELUXE achieves eMBB throughput that is comparable to that with EESM, which is considered the most spectrally efficient LA method for 4G LTE and 5G NR. Finally, our GPU-based implementation of DELUXE can successfully meet the 5G real-time requirement (under 100 μs).

The rest of the paper is organized as follows. In Section II, we review related work on dynamic eMBB/URLLC multiplexing and the applications of DL in wireless communications and networking. In Section III, we describe the system model and the problem that we study in this paper. Section IV presents the architecture and key ideas of DELUXE. Through Section V to VII, we present each of the three stages of DELUXE. In Section VIII, we describe our GPU-based real-time implementation for DELUXE. Section IX presents our performance evaluation of DELUXE. Section X concludes this paper.

II. Related Work

In this section, we review related work in two thrusts that are most relevant to this research: (i) multiplexing of eMBB and URLLC (on the same channel), and (ii) the use of DL for other topics in wireless communications and networking.

A. eMBB/URLLC Multiplexing

Related work on dynamic multiplexing of eMBB and URLLC based on preemptive puncturing includes [18], [31]–[37]. In [18], the authors conducted system-level simulations and confirmed that preemptive puncturing achieves significantly higher spectral efficiency than static bandwidth division under sporadic URLLC packet arrivals.

In [31], the authors studied a bandwidth allocation problem for eMBB transmissions under linear and nonlinear models for the impact of URLLC puncturing. In [32], the authors considered joint RB allocation for eMBB and scheduling of URLLC puncturing. A one-sided matching and heuristic algorithm was proposed to solve the formulated optimization problem. Subsequently, in [33], the authors employed a penalty successive upper bound minimization method and a transportation model to address the same optimization problem. In [34], scheduling of URLLC puncturing over eMBB resources was studied via a risk-sensitive approach in order to protect eMBB users with low data rates. The work in [35] investigated the use of deep reinforcement learning (DRL) for dynamic multiplexing of eMBB and URLLC with the objective of minimizing the adverse impact of URLLC traffic on eMBB services. In [36], eMBB resource allocation and URLLC preemptive scheduling were jointly optimized and solved through a DRL framework. All these works [18], [31]–[36] did not consider the problem of MCS selection for eMBB under the impact of URLLC puncturing.

MCS selection for eMBB was considered in [37]. In this work, the authors used system-level simulations to evaluate the impact of preemptive URLLC puncturing on eMBB. The well-known outer loop link adaptation (OLLA) algorithm was employed for MCS selection (for eMBB). However, their results showed that eMBB's BLER target could not be satisfied when URLLC puncturing is present.

B. DL for Other Topics in Wireless Communications and Networking

DL and DRL methods have also been extensively applied to address other wireless communications and networking problems in recent years (for a survey, see [38]). This line of research has two main branches. In the first branch, the goal is to custom-design a neural network structure for a specific problem (see, e.g., [39]–[41]). Due to problem-specific nature, a custom-designed neural network structure, in general, cannot be easily extended to solve other problems, such as the MCS selection problem in this paper.

In the second branch, a problem under study is first cast into a DRL framework and then solved by designing a suitable neural network structure (see, e.g., [42]–[44]). The most widely used DRL framework in wireless communications and networking is deep Q-learning (DQL) [25]. Unfortunately, DQL-based approaches are not suitable for solving the MCS selection problem in this paper. This is because Q-learning is a greedy algorithm and therefore, it cannot handle the BLER constraints and ensure feasibility in the decision making process.

III. System Model and Problem Statement

A. System Model

Table I and II list acronyms and notations used in this paper, respectively. Consider a 5G NR macro-cell with its BS located at the cell center and a number of eMBB and URLLC users within the cell's coverage, as shown in Fig. 3. Denote $\mathcal{X}$ as the set of eMBB users in the cell. In the downlink direction, the data traffic of eMBB and URLLC is dynamically multiplexed on the same channel. To focus on the impact of URLLC traffic on eMBB transmission, we assume a full-buffer model for each eMBB user $x \in \mathcal{X}$, i.e., the eMBB buffers at the BS are always nonempty in the downlink direction.

For eMBB transmission, time domain is divided into consecutive TTIs. Scheduling for eMBB transmission is performed for each TTI. Denote t as the time index of a TTI. Typically, a TTI may consist of one or more time slots, with each time slot consisting of 14 OFDM symbols [5]. Denote F as the number of OFDM symbols in a TTI (which is a multiple of 14). Assume that under the fast fading effects, the channel condition for an eMBB link (from the BS to an eMBB user) varies in each TTI.

In the frequency domain, the bandwidth of the operating channel is divided into a large number of SC. Assume that

TABLE I
ACRONYMS

Acronym	Full Name
AMC	Adaptive Modulation and Coding
AWGN	Additive White Gaussian Noise
BLER	Block-Error Rate
BS	Base Station
BWP	Bandwidth Part
CB	Code Block
CDF	Cumulative Distribution Function
CDL	Clustered Delay Line
CPU	Central Processing Unit
CQI	Channel Quality Indicator
DFA	Deep Function Approximator
DL	Deep Learning
DQL	Deep Q-Learning
DRL	Deep Reinforcement Learning
eBWP	exclusive BWP
EESM	Exponential Effective SNR Mapping
eMBB	enhanced Mobile Broad-Band
FEC	Forward Error Correction
GPU	Graphic Processing Unit
LA	Link Adaptation
LDPC	Low-Density Parity-Check code
mBWP	multiplexing BWP
MCS	Modulation and Coding Scheme
MIMO	Multiple-Input and Multiple-Output
MMSE	Minimum Mean Square Error
MSE	Mean Square Error
NLOS	Non-Line-Of-Sight
OFDM	Orthogonal Frequency-Division Multiplexing
OLLA	Outer Loop Link Adaptation
PD-SNR	Post-Detection-SNR
PDF	Probability Density Function
PF	Proportional-Fair
PHY	Physical layer
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
RB	Resource Block
RE	Resource Element
SC	Sub-Carrier
SGD	Stochastic Gradient Descent
SIMD	Single-Instruction Multiple-Data
SNR	Signal-to-Noise Ratio
TB	Transport Block
TDL	Tapped Delay Line
TTI	Transmission Time Interval
URLLC	Ultra-Reliable Low Latency Communications

TABLE II
NOTATION

Symbol	**Definition**
$a_{i,m}(t)$	The binary decoding result for $e_i(t)$ under MCS m
$\mathcal{B}$	The set of all RBs on the operating channel
$\mathcal{B}_i(t)$	The set of RBs allocated to $e_i(t)$
$\boldsymbol{c}_i(t)$	The vector describing the RE-level channel conditions for $e_i(t)$
$\mathcal{E}(t)$	The set of eMBB transmissions scheduled for TTI t
$e_i(t)$	The i-th eMBB transmission in $\mathcal{E}(t)$
F	The number of OFDM symbols in a TTI
I	The number of OFDM symbols in a mini-slot
J	The total number of iterations in the learning phase of Algorithm 1
L	The number of elements in $\boldsymbol{\xi}$
$\mathcal{M}$	The set of MCS levels that can be used for eMBB
M	$= \lvert\mathcal{M}\rvert$, the number of MCS levels in $\mathcal{M}$
$\boldsymbol{p}_i(t)$	The vector specifying the RE-level URLLC puncturing on $e_i(t)$
$\boldsymbol{q}_i(t)$	The vector of $(1 - r_{i,m}(t))$ values under each MCS m for $e_i(t)$
$r_{i,m}(t)$	The BLER for $e_i(t)$ under MCS m
$\boldsymbol{s}_i(t)$	The original input vector for $e_i(t)$
$\tilde{\boldsymbol{s}}_i(t)$	The low-dimensional representation of $\boldsymbol{s}_i(t)$
S	The number of SCs in an RB
T	The total number of TTIs in the data collection phase of Algorithm 1
$\mathcal{X}$	The set of eMBB users in the cell
Z	The update period for β^x in number of eMBB transmissions
β^x	The calibration parameter for eMBB user $x \in \mathcal{X}$
γ	The BLER target for eMBB links
γ_0	The cushion parameter in Algorithm 2
Δ_{D}	The step size for decreasing β^x in Algorithm 2
Δ_{I}	The step size for increasing β^x in Algorithm 2
Θ	The set of trainable parameters in μ's multi-layer structure
λ	The URLLC packet arrival rate
μ	The DFA in the Stage II of DELUXE
$\mu\left(\tilde{\boldsymbol{s}}_i(t)\lvert\Theta\right)$	The output vector of μ given the input $\tilde{\boldsymbol{s}}_i(t)$ and parameters Θ
$\mu_m\left(\tilde{\boldsymbol{s}}_i(t)\lvert\Theta\right)$	The m-th output entry in $\mu\left(\tilde{\boldsymbol{s}}_i(t)\lvert\Theta\right)$
$\boldsymbol{\xi}$	The vector used for generating the low-dimensional input representation $\tilde{\boldsymbol{s}}_i(t)$
ρ	The step size in the learning phase of Algorithm 1
$\upsilon(e_i(t))$	$\in \mathcal{X}$, the receiving eMBB user corresponding to $e_i(t)$

under frequency-selective channel fading, the channel condition for an eMBB link varies on each SC. A contiguous S SCs are grouped into an RB. According to NR standard [5], $S = 12$. An RB is the minimum frequency unit for resource scheduling over the duration of a TTI. Denote $\mathcal{B}$ as the set of all RBs on the channel. For each TTI t, the RB allocation for eMBB transmissions is determined by the BS scheduler in TTI $(t-1)$ following a scheduling algorithm (e.g., proportional-fair (PF) algorithm [20], [21]). Let $\mathcal{E}(t)$ denote the set of eMBB transmissions scheduled for TTI t, where each eMBB transmission is a communication link from the BS to an eMBB user $x \in \mathcal{X}$. For example, $\mathcal{E}(t)$ in Fig. 3 consists of 3 eMBB transmissions: BS to $e_1(t)$, $e_2(t)$, and $e_3(t)$, respectively, where $e_i(t)$ denotes the i-th transmission in $\mathcal{E}(t)$. Denote $\upsilon(e_i(t)) \in \mathcal{X}$ as the receiving eMBB user

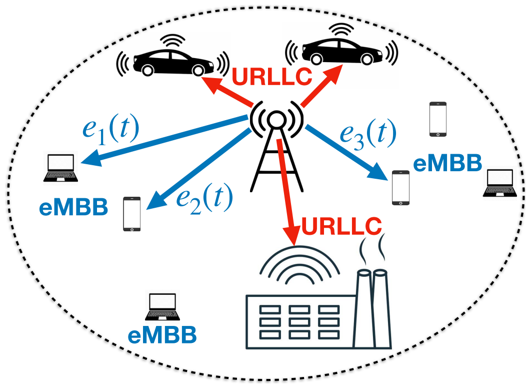


Fig. 3. An NR macro-cell with eMBB/URLLC multiplexing.

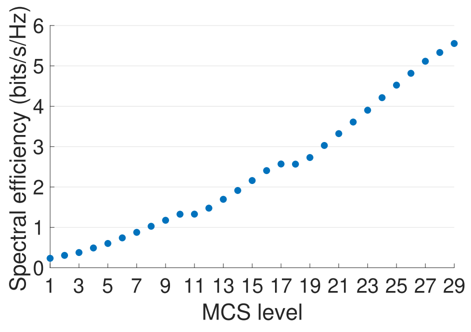


Fig. 4. Spectral efficiencies of the 29 MCS levels defined in NR standard (Table 5.1.3.1-1 in [16]).

corresponding to $e_i(t)$. Denote the set of RBs allocated to transmission $e_i(t)$ as $\mathcal{B}_i(t)$.

Given a scheduled eMBB transmission $\mathcal{E}(t)$ for a given TTI t, one needs to select an MCS for each $e_i(t) \in \mathcal{E}(t)$. Denote $\mathcal{M} = \{1, 2, \cdots, 29\}$ as the set of 29 MCS levels defined in the NR standards [16]. Each MCS represents a combination of a specific modulation type (QPSK, 16-QAM, or 64-QAM) and LDPC code rate (between 0 and 1). The spectral efficiency of each MCS level is shown in Fig. 4. A key constraint associated with MCS is that the same MCS must be used for all the RBs allocated to an eMBB transmission [16]. The main reason behind this constraint is that the benefit of using different MCS levels for different RBs (in terms of throughput gain) cannot justify the excessive control signaling overhead associated with it [9].

For URLLC traffic, we assume that URLLC packets arrive at the BS sporadically following a Poisson process with an arrival rate λ (in number of packets per TTI). Different from eMBB, each URLLC packet is transmitted using a "mini-slot", where a mini-slot contains a much fewer number of OFDM symbols (e.g., 2, 4, or 6 [5]) than a regular time slot (14 symbols). Denote I as the number of OFDM symbols in a mini-slot. A URLLC transmission is scheduled immediately for the next mini-slot following the packet arrival through preemptive puncturing as described in Section I. Specifically, in the frequency domain, a set of contiguous RBs starting from the highest (or lowest) frequency will be punctured in the scheduled mini-slot (see Fig. 1).

Following NR standard [5], we consider a bandwidth part (BWP) based channel division. Under this bandwidth

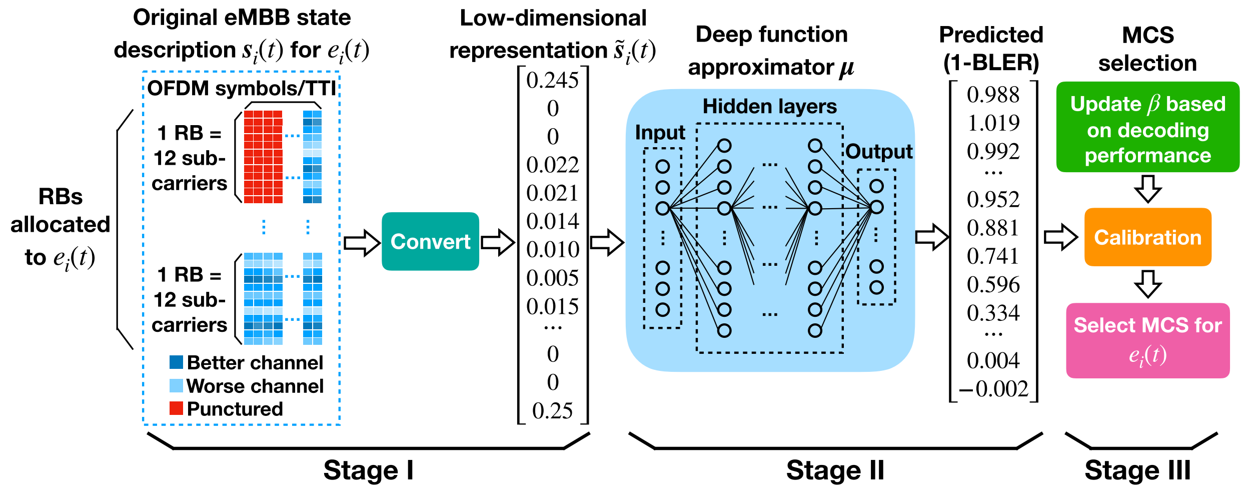


Fig. 5. Architectural overview of DELUXE. This figure illustrates the MCS selection process for a given $e_i(t)$.

division, the operating channel is divided into two non-overlapping frequency parts (or BWPs): the multiplexing BWP (mBWP) and the eMBB exclusive BWP (eBWP), respectively. As shown in Fig. 1, in mBWP, the multiplexing of eMBB and URLLC occurs through preemptive puncturing. In eBWP, the bandwidth is used exclusively for eMBB transmissions. Benefits of such a partition scheme include low control channel overhead for indicating positions of URLLC puncturing and the compatibility with current control channel design in NR standards [17]. In each TTI t, an eMBB transmission $e_i(t)$ may be allocated with RBs from either mBWP, or eBWP, or both. Such an RB allocation decision is done by the eMBB scheduler at the BS and is beyond the scope of this paper.

B. Problem Statement

The problem we want to solve in this paper is to find an optimal MCS level $m_i^*(t)$ for each $e_i(t) \in \mathcal{E}(t)$ so as to maximize throughput while satisfying a given BLER target. Such an MCS selection for each $e_i(t) \in \mathcal{E}(t)$ is based on the following input: i) the code block (CB) length, ii) the channel conditions, and iii) URLLC puncturing on the RBs allocated to $e_i(t)$. More discussion about these three inputs will be given in Section V-A.

Denote γ as the BLER target for eMBB. For 5G, γ is typically set to 10% [47], meaning that the decoding failure probability of an eMBB transmission should not exceed 10%. Then our problem can be stated mathematically as follows:

$$m_i^*(t) = \arg\max_{m \in \mathcal{M}} \ \mathrm{SE}(m), \ \text{subject to} : r_{i,m}(t) \le \gamma, \quad (1)$$

where $\arg\max$ represents the variable value that maximizes the objective function, $\mathrm{SE}(m)$ is the spectral efficiency for MCS level m (see Fig. 4), and $r_{i,m}(t)$ denotes the BLER for $e_i(t)$ when MCS is set to m.

C. Technical Challenges

A major challenge in solving the MCS selection problem in (1) is that it is analytically intractable to determine the BLER $r_{i,m}(t)$ for each $e_i(t)$ under a given MCS m. This is because 5G NR employs LDPC as the forward error correction (FEC) channel code for eMBB transmissions [19]. It is well-known that LDPC decoding performance (in terms of BLER) cannot be modeled using a closed-form mathematical expression. That is, we cannot determine the $r_{i,m}(t)$ values for each $e_i(t)$ even when the perfect information of channel condition and URLLC puncturing on $e_i(t)$ is available. As a result, the problem in (1) cannot be formulated as a closed-form optimization problem where conventional optimization techniques may be applied.

Another technical challenge associated with the problem in (1) is that its solution must be obtained in real time, i.e., an optimal MCS must be selected for each $e_i(t) \in \mathcal{E}(t)$ within a TTI's duration. In 5G NR, the shortest TTI duration for eMBB is 125 μs under Numerology 3 [5]. This means that the MCS selection problem in (1) must be solved on ~100 μs time scale. Based on our past experience, one must pursue parallel processing in implementation to meet such a stringent real-time requirement.

IV. DELUXE: Architecture and Key Ideas

Given the set of eMBB transmissions $\mathcal{E}(t)$ scheduled for a TTI t, the objective of DELUXE is to determine the optimal MCS $m_i^*(t)$ for each $e_i(t) \in \mathcal{E}(t)$ as described in problem (1). For each $e_i(t)$, we need to provide the following information to DELUXE: i) the length of the eMBB CB in $e_i(t)$, ii) channel conditions on the RBs allocated to $e_i(t)$, and iii) URLLC puncturing on the RBs allocated to $e_i(t)$. The basic idea behind DELUXE is to employ a DFA (a feedforward neural network), denoted by μ, to predict the value of $(1 - r_{i,m}(t))$ for a given $e_i(t)$ under each MCS $m \in \mathcal{M}$. Then based on the predicted $(1 - r_{i,m}(t))$ values, we will be able to find an optimal MCS that solves problem (1).

Fig. 5 shows the overall architecture of DELUXE. There are three stages in DELUXE, i.e., Stage I, II and III. Next, we briefly describe the key ideas in each stage. Technical details of the three stages will be given in Section V, VI and VII, respectively.

Stage I: *Reducing complexity of input data.* If one used the original high-dimensional information for each $e_i(t)$ as an input to μ, a very complex hidden-layer structure in μ would be necessary (e.g., with a series of convolutional, pooling and fully-connected layers) to extract the essential information to predict $(1 - r_{i,m}(t))$ values. This will lead to very high processing/inference time of μ that is far beyond our stringent real-time requirement ($\sim$100 μs). To address this issue, in Section V, we propose an approach to transform the original high-dimensional input information for each $e_i(t)$ into a low-dimensional representation with a minimized information loss.

Stage II: *Learning and prediction.* The design and learning of DFA μ is the core component of DELUXE. The input to μ is a low-dimensional representation of the channel condition and URLLC puncturing information for $e_i(t)$ (obtained in Stage I). The output of μ is a real-valued vector of length $M = |\mathcal{M}|$, with each entry representing the predicted $(1 - r_{i,m}(t))$ for $e_i(t)$ under each MCS $m \in \mathcal{M}$. We employ a fully-connected layer structure for μ (i.e., a neuron within a layer is fully connected to all neurons in the next layer) such that it can offer a strong universal function approximating capability that can meet our need [22]. In Section VI, we will present the details of our learning method for μ based on the decoding results from a large number of eMBB transmission samples.

Stage III: *Calibration and MCS selection.* Our proposed learning method allows μ to predict the $(1 - r_{i,m}(t))$ values for a given $e_i(t)$ under each MCS m. However, due to the nature of a DFA, there may exist errors in these predictions. To mitigate the impact of such errors in MCS selection, in Section VII, we propose a method to dynamically calibrate the $(1 - r_{i,m}(t))$ predictions from μ for each eMBB user based on its recent history of actual decoding results.

V. Reducing Complexity of Input Data

In this section, we describe the operation in Stage I (see Fig. 5), which is to transform the original high-dimensional input data for an eMBB transmission $e_i(t)$ into a low-dimensional representation. We first describe the scope of the input data (Section V-A). Then we propose a method to achieve such transformation (Section V-B).

A. Input Data

In each TTI t, an eMBB transmission $e_i(t) \in \mathcal{E}(t)$ can be described by a vector $\boldsymbol{s}_i(t)$ that contains the following information: (i) the length of the eMBB CB in $e_i(t)$, (ii) the URLLC puncturing on $e_i(t)$, and (iii) the channel conditions across frequency and time on $e_i(t)$. These three types of information determine the BLER that will be achieved by $e_i(t)$. We now discuss each of these inputs in details.

(i) The first component of $\boldsymbol{s}_i(t)$ contains information regarding the length of the eMBB CB in $e_i(t)$. Given the RB allocation $\mathcal{B}_i(t)$, we have the following question: what is the CB length under each of the 29 MCSs?

TABLE III
Determine CB Length for $e_i(t)$ Under Each MCS

MCS	0-9	10-16	17-28
Modulation	QPSK	16-QAM	64-QAM
Bits/RE	2	4	6
Bits/RB	336	672	1,008
CB length	$336\lvert\mathcal{B}_i(t)\rvert$	$672\lvert\mathcal{B}_i(t)\rvert$	$1{,}008\lvert\mathcal{B}_i(t)\rvert$

We can answer this question by constructing Table III. In this table, each of the $M = 29$ MCSs corresponds to a specific modulation type (first row to second row). Under each modulation type, we know exactly how many codeword bits are carried in a resource element (RE),[2] as shown in the third row in the table. Since the number of REs contained in an RB within a TTI is a constant number (i.e., 14 OFDM symbols/TTI $\times$ 12 SCs/RB), the number of codeword bits per RB under each modulation type can be determined (fourth row in the table). This means that the CB length for $e_i(t)$ under each MCS is equal to $|\mathcal{B}_i(t)|$ multiplied by a constant, as shown in the last row of the table. In other words, for each of the 29 MCSs, we can find the CB length through Table III if we know $|\mathcal{B}_i(t)|$.

(ii) The second component of $\boldsymbol{s}_i(t)$ is a vector $\boldsymbol{p}_i(t)$ to specify URLLC puncturing on $e_i(t)$. Since puncturing occurs on the OFDM symbol level (in mini-slots), $\boldsymbol{p}_i(t)$ should have a granularity at the level of RE. In other words, the number of elements in $\boldsymbol{p}_i(t)$ is equal to the total number of REs in $e_i(t)$, with each of $\boldsymbol{p}_i(t)$'s elements indicating whether or not an RE is punctured by URLLC. Given RB allocation $\mathcal{B}_i(t)$, the total number of REs contained in $e_i(t)$ is $|\mathcal{B}_i(t)| \cdot S \cdot F$. Thus, $|\boldsymbol{p}_i(t)| = |\mathcal{B}_i(t)| \cdot S \cdot F$.

However, since the URLLC packet arrivals are sporadic, we do not know how many packets will actually arrive in TTI t when generating the input vector $\boldsymbol{s}_i(t)$ in TTI $t-1$. A reasonable estimate for the number of URLLC packets arrived in each TTI is λ, the Poisson arrival rate of URLLC packets. Further, when generating $\boldsymbol{s}_i(t)$ for each $e_i(t)$, we assume the puncturing of the (expected arrival) URLLC packets occurs in consecutive mini-slots starting from the first in each TTI (within the mBWP). This assumption on puncture position can be justified by noticing that the duration of a TTI is shorter than the channel coherence time, meaning that the channel conditions only experience slight variation across the mini-slots within a TTI. Further, the impact of URLLC puncturing on specific mini-slots in a TTI is spread out uniformly across the eMBB CB by the bit interleaving operation [16].

(iii) The third component of $\boldsymbol{s}_i(t)$ is a vector $\boldsymbol{c}_i(t)$ to describe channel conditions on $e_i(t)$. Similar to $\boldsymbol{p}_i(t)$, $\boldsymbol{c}_i(t)$ should have a granularity at RE level, i.e., the number of elements in $\boldsymbol{c}_i(t)$ should be equal to the total number of REs in $e_i(t)$. That is, $|\boldsymbol{c}_i(t)| = |\mathcal{B}_i(t)| \cdot S \cdot F$.

For each element in $\boldsymbol{c}_i(t)$, we employ *post-detection-SNR* (PD-SNR) as the metric for channel condition. To concretize our discussion, we assume single-user single-layer MIMO

[2]An RE is defined as one OFDM symbol time duration by one SC in frequency [5].

is used for eMBB transmission.[3] Following frequency-major order in the frequency-time RE grid, consider the j-th RE allocated to $e_i(t)$. This corresponds to the j-th entry in $\boldsymbol{c}_i(t)$, which we denote as $c_{i,j}(t)$. Let $\boldsymbol{H}$ and $\boldsymbol{w}$ denote the channel matrix and the precoding vector on this RE. Then with minimum mean square error (MMSE) detector, the PD-SNR (in dB) for this RE is [24]:

$$c_{i,j}(t) = 10 \log_{10} \left[(\boldsymbol{H}\boldsymbol{w})' \boldsymbol{H}\boldsymbol{w} \cdot v_0 \right], \tag{2}$$

where v_0 is the ratio of per-RE signal power over the noise power.

In summary, the input vector $\boldsymbol{s}_i(t)$ is written as:

$$\boldsymbol{s}_i(t) = [|\mathcal{B}_i(t)|;\ \boldsymbol{c}_i(t);\ \boldsymbol{p}_i(t)]\,. \tag{3}$$

There are two issues with $\boldsymbol{s}_i(t)$ in (3). First, the number of RBs $|\mathcal{B}_i(t)|$ allocated to $e_i(t)$ typically varies with TTI t [23]. As a result, the sizes of $\boldsymbol{c}_i(t)$ and $\boldsymbol{p}_i(t)$ both vary with TTI (as both depend on $|\mathcal{B}_i(t)|$). Hence, the dimension of $\boldsymbol{s}_i(t)$ varies in each TTI. Such variability in size makes $\boldsymbol{s}_i(t)$ unsuitable as an input to the DFA μ, which requires a constant input size that is independent of time. Second, the dimension of $\boldsymbol{s}_i(t)$ can be very large. For example, suppose an eMBB transmission $e_i(t)$ is allocated with 20 RBs. Then based on (3), we have $|\boldsymbol{s}_i(t)| = 6,720$ (when a TTI consists of only one time slot). With such a large-sized input, μ needs to have a large and complex layer structure to perform its function (extract the essential information from $\boldsymbol{s}_i(t)$). The inference time under such a large and complex structure will almost certainly violate our real-time requirement ($\sim$100 μs).

B. Transformation to Lower-Dimensional Representation

In this section, we propose a method to transform the original input $\boldsymbol{s}_i(t) = [|\mathcal{B}_i(t)|;\ \boldsymbol{c}_i(t);\ \boldsymbol{p}_i(t)]$ into a lower dimensional representation, denoted by $\tilde{\boldsymbol{s}}_i(t)$, for a given $e_i(t)$. First, we note that $|\mathcal{B}_i(t)|$ is a scalar and there is no room to compress it further. So our focus is to transform vectors $\boldsymbol{c}_i(t)$ and $\boldsymbol{p}_i(t)$ in $\boldsymbol{s}_i(t)$ into their low-dimensional representations, which are denoted by $\tilde{\boldsymbol{c}}_i(t)$ and $\tilde{\boldsymbol{p}}_i(t)$, respectively. Our objective for this transformation is three-fold:

- **(i)** the dimension of $\tilde{\boldsymbol{s}}_i(t)$ must be constant and independent of t;
- **(ii)** $\tilde{\boldsymbol{s}}_i(t)$ must have a much lower dimension compared to $\boldsymbol{s}_i(t)$;
- **(iii)** the loss of information in the transformation should be minimized.

1) Key Idea: The basic idea is to change the information granularity from RE level to something bigger (coarser). In the original input vector $\boldsymbol{s}_i(t)$, we have $\boldsymbol{c}_i(t)$ and $\boldsymbol{p}_i(t)$, both of which have a granularity at the RE level, leading to very large sizes for both vectors. Instead of RE level representation, we propose the following new representation for channel condition. We will use a set of intervals (orders of magnitude fewer than the number of REs) to record the number of entries in $\boldsymbol{c}_i(t)$ and $\boldsymbol{p}_i(t)$ that fall within each interval. Then we use these low-dimensional statistics as $\tilde{\boldsymbol{c}}_i(t)$ and $\tilde{\boldsymbol{p}}_i(t)$ to replace $\boldsymbol{c}_i(t)$ and $\boldsymbol{p}_i(t)$.

[3]The same approach described here can be applied to other MIMO transmission modes and detectors.

One would immediately notice that the above approach not only reduces information granularity, but also loses position information for the channel condition. So the question is: what is the theoretical basis for such a transformation? The answer lies in the bit interleaving operation during PHY-layer signal processing. Recall that after this operation, the codeword bits carried in the REs of $e_i(t)$ are uniformly spread out across the eMBB CB and no longer maintain their positions in the frequency-time RE grid. As a result, the RE position information carried in $\boldsymbol{c}_i(t)$ and $\boldsymbol{p}_i(t)$ is no longer useful and does not need to be maintained.

Such a transformation will meet the first two objectives for $\tilde{\boldsymbol{s}}_i(t)$ (constant size, smaller dimension). In the rest of this section, we will show the details of this transformation and address the third objective (to minimize information loss).

2) Transformation of $\boldsymbol{c}_i(t)$: We first discuss $\boldsymbol{c}_i(t)$. The property of bit interleaving operation allows us to turn away from maintaining per-RE based channel condition information. From (2), we see that the range of PD-SNRs (dB) is $(-\infty, +\infty)$. To guarantee a constant and finite size of the representation $\tilde{\boldsymbol{c}}_i(t)$, we propose to discretize the interval $(-\infty, +\infty)$ into a finite (constant) number of intervals and use the percentages of PD-SNRs in $\boldsymbol{c}_i(t)$ that fall into these intervals to characterize channel condition for eMBB $e_i(t)$.

Specifically, we employ a vector $\boldsymbol{\xi} = [\xi_1, \xi_2, \cdots, \xi_L]$ to divide $(-\infty, +\infty)$ into $(L+1)$ intervals: $(-\infty, \xi_1]$, $(\xi_1, \xi_2]$, $\cdots$, $(\xi_L, +\infty)$.[4] Then the representation $\tilde{\boldsymbol{c}}_i(t)$ is a vector of size $(L+1)$. That is,

$$\tilde{\boldsymbol{c}}_i(t) = [\tilde{c}_{i,1}(t), \tilde{c}_{i,2}(t), \ldots, \tilde{c}_{i,L+1}(t)], \tag{4}$$

where $\tilde{c}_{i,l}(t)$ is the percentage of PD-SNRs from the per-RE based $\boldsymbol{c}_i(t)$ that fall in the l-th interval. More precisely, $\tilde{c}_{i,l}(t)$ is the percentage of PD-SNRs on REs that are *not* punctured by URLLC.

A key question in this discretization process is the following: *For a given L, how to optimize ξ_1, ξ_2, $\cdots$, ξ_L in $\boldsymbol{\xi}$ so that the information loss in this transformation can be minimized?* In the following paragraphs, we address this important question.

3) Minimizing Information Loss: We address the above question (minimizing information loss) by formulating an optimization problem as follows. For the objective function, we use the *entropy* [49] of PD-SNR as the metric to quantify the information loss due to our discretization of $(-\infty, +\infty)$ into $(L+1)$ intervals. Here the PD-SNR in (2) is a random variable for all possible eMBB transmissions in the cell with an *unknown* probability density function (PDF) over $(-\infty, +\infty)$.

Denote $H(\xi_l, \xi_{l+1})$ as the entropy of PD-SNR in the interval $(\xi_l, \xi_{l+1}]$, and $P(\xi_l, \xi_{l+1})$ as the probability that the PD-SNR falls within the interval $(\xi_l, \xi_{l+1}]$. Then our optimization

[4]The choice of L depends on the following two considerations: i) L should be small enough (orders or magnitude smaller than $|\boldsymbol{s}_i(t)|$) to meet the 5G real-time requirement; and ii) L should still be large enough so that $\tilde{\boldsymbol{s}}_i(t)$ contains enough information for an optimal MCS selection. In our experimental study, we found that $L = 100$ achieves a good balance between these two considerations.

problem can be formulated as:

OPT-L:

$$\begin{aligned} &\text{minimize } \sum_{l=1}^{L-1} P\left(\xi_l, \xi_{l+1}\right) H(\xi_l, \xi_{l+1}) \,, \\ &\text{subject to } \xi_1 < \xi_2 < \cdots < \xi_L. \end{aligned}$$

There are two problems with OPT-L. First, the PDF of PD-SNR is unknown. As a result, the probabilities $P\left(\xi_l, \xi_{l+1}\right)$ are unknown as well. Second, entropy $H(\xi_l, \xi_{l+1})$, which is a function of the PD-SNR's PDF, cannot be determined in closed-form. To address the problems with $P\left(\xi_l, \xi_{l+1}\right)$ and $H(\xi_l, \xi_{l+1})$ in OPT-L, we propose the following solutions.

Despite that the PDF of PD-SNR is unknown, we can still estimate its distribution through experiments and empirical data. Specifically, we can collect a set $\mathcal{N}$ of PD-SNR samples from all active eMBB links in the cell. Each sample element in $\mathcal{N}$ is the PD-SNR on an RE in an eMBB transmission. Let N denote the number of samples in $\mathcal{N}$ (e.g., $N = 10^6$). Then for an interval $(\xi_l, \xi_{l+1}]$, $P\left(\xi_l, \xi_{l+1}\right)$ can be estimated by calculating the proportion of samples in $\mathcal{N}$ that fall in this interval.

Although the above experiments can give us an estimate for $P\left(\xi_l, \xi_{l+1}\right)$, we still do not have a closed-form PDF for PD-SNR. So for entropy $H(\xi_l, \xi_{l+1})$, we can obtain its upper bound by using a uniform distribution for the PDF of PD-SNR within each interval $(\xi_l, \xi_{l+1}]$. Then we have [49]

$$H(\xi_l, \xi_{l+1}) \leq \log\left(\xi_{l+1} - \xi_l\right).$$

We will use $\log\left(\xi_{l+1} - \xi_l\right)$ as an approximation for $H(\xi_l, \xi_{l+1})$ in the objective function of OPT-L.

With the above estimation for $P\left(\xi_l, \xi_{l+1}\right)$ and upper bound for $H(\xi_l, \xi_{l+1})$ in OPT-L, we propose a greedy algorithm to find a vector $\boldsymbol{\xi} = [\xi_1, \xi_2, \cdots, \xi_L]$ to divide $(-\infty, +\infty)$ into $(L+1)$ intervals as follows.

(i) First, for the N samples from the experiment, we initialize vector $\boldsymbol{\xi}$ such that each of the $(L+1)$ intervals contains the same number of samples, i.e., $N/(L+1)$ samples in each of the intervals $(-\infty, \xi_1]$, $(\xi_1, \xi_2]$, $\cdots$, $(\xi_L, +\infty)$.

(ii) Next, we iteratively update one ξ_l ($l = 2, 3, \cdots, L-1$) in $\boldsymbol{\xi}$ that offers the greatest reduction in the objective value in OPT-L for K iterations (or until no improvement is possible). Specifically, in each iteration, we slightly perturb each ξ_l ($l = 2, 3, \cdots, L-1$) to both the left and right for a certain amount (e.g., such an amount may correspond to an increase or decrease of some fixed number of samples for the underlying intervals). For each perturbation, we calculate the change of objective value in OPT-L. Among the $2(L-2)$ perturbations of ξ_l ($l = 2, 3, \cdots, L-1$), we find the one that offers the maximum reduction in the objective value and update the corresponding ξ_{l^*} with this new value for future iterations.

4) Transformation of $\boldsymbol{p}_i(t)$: Similar to the transformation from $\boldsymbol{c}_i(t)$ to $\tilde{\boldsymbol{c}}_i(t)$, we no longer need to maintain each RE's position information in $\boldsymbol{p}_i(t)$ (due to bit interleaving operation). Instead, we only use a single scalar, denoted by $\tilde{p}_i(t)$ (a reduction from the vector notation $\tilde{\boldsymbol{p}}_i(t)$), to indicate the percentage of REs in $e_i(t)$ that is punctured by URLLC (based on $\boldsymbol{p}_i(t)$).

To better understand this transformation, consider an RE that has been punctured by URLLC. For this RE, its effective PD-SNR (dB) is $-\infty$, since on this RE the useful signal power for eMBB transmission is zero and the inference power increases significantly due to URLLC transmission. In this regard, we can imagine $\tilde{p}_i(t)$ corresponds to a "point" $(-\infty)$ on the leftmost position of the $(L+1)$ intervals ($(-\infty, \xi_1]$, $\cdots$, $(\xi_L, +\infty)$) with a probability $1 - \sum_{k=1}^{L+1} \tilde{c}_{i,k}(t)$.

Based on the above discussions, we have

$$\tilde{p}_i(t) = 1 - \sum_{k=1}^{L+1} \tilde{c}_{i,k}(t). \tag{5}$$

5) Putting Everything Together: We have the following new input representation $\tilde{\boldsymbol{s}}_i(t)$:

$$\tilde{\boldsymbol{s}}_i(t) = [\tilde{B}_i(t); \tilde{\boldsymbol{c}}_i(t); \tilde{p}_i(t)], \tag{6}$$

where $\tilde{\boldsymbol{c}}_i(t)$ is defined in (4), $\tilde{p}_i(t)$ is defined in (5), and

$$\tilde{B}_i(t) = \frac{|\mathcal{B}_i(t)|}{|\mathcal{B}|}.$$

We use $\tilde{B}_i(t)$ instead of $|\mathcal{B}_i(t)|$ to ensure all entries in $\tilde{\boldsymbol{s}}_i(t)$ are normalized within $[0, 1]$.

C. A Case Study

We now use a case study to validate the efficacy of our proposed method to optimize $\boldsymbol{\xi}$ with respect to problem OPT-L. We employ a vector $\boldsymbol{\xi}$ of size $L = 100$ to divide the range $(-\infty, +\infty)$ into 101 intervals: $(-\infty, \xi_1]$, $(\xi_1, \xi_2]$, $\cdots$, $(\xi_{100}, +\infty)$.

As the first step of the method, we collect a number $N = 10^6$ of per-RE PD-SNR samples from 10 eMBB users (see Section IX for parameter settings). During data collection, the 10 eMBB users randomly roam within the cell and the PD-SNRs from all users in each TTI are stored into the sample set $\mathcal{N}$. Then the PD-SNR samples in $\mathcal{N}$ are sorted in ascending order.

After data collection, the elements in $\boldsymbol{\xi}$ are initialized such that all intervals $(-\infty, \xi_1]$, $\cdots$, $(\xi_{100}, +\infty)$ contain the same number of PD-SNR samples in $\mathcal{N}$. We then execute the greedy algorithm described in Section V-B to optimize $\boldsymbol{\xi}$ for $K = 10^4$ iterations. In each iteration, a chosen entry ξ_{l^*} ($l^* \in \{2, \cdots, 99\}$) in $\boldsymbol{\xi}$ that achieves the maximum reduction in the objective value of OPT-L is perturbed by an amount corresponding to 10 PD-SNR samples in the sorted set $\mathcal{N}$.

In Fig. 6, we show the change of information loss (i.e., the change of the objective value of OPT-L) achieved by the proposed greedy algorithm in each iteration. We can see that the change of information loss is always negative, meaning that the information loss can be effectively reduced through the iterations. In particular, the algorithm terminates at iteration 223 because the objective of OPT-L cannot be further reduced by adjusting any entry in $\boldsymbol{\xi}$.

The vector $\boldsymbol{\xi}$ optimized in this case study will be used for our performance evaluation of DELUXE in Section IX.

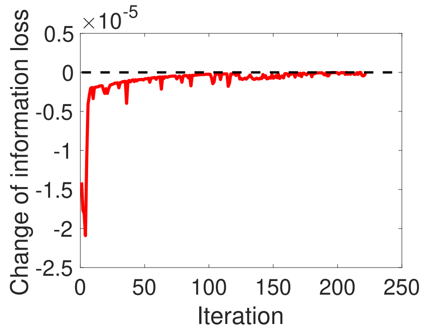


Fig. 6. The change of information loss through the iterations.

VI. Learning and Prediction

This section presents the details of DFA μ, which is Stage II of DELUXE in Fig. 5. Our goal is to obtain an optimized μ that is able to map an input vector $\tilde{\boldsymbol{s}}_i(t)$ into a vector of $(1-r_{i,m}(t))$ predictions for each MCS $m \in \mathcal{M}$, where $r_{i,m}(t)$ is the BLER of $e_i(t)$ under MCS m (see (1)).

A. Motivation

Assume that there exists an *unknown* function, denoted by $f(\cdot)$, that maps each input vector $\tilde{\boldsymbol{s}}_i(t)$ to a vector of *actual* $(1-r_{i,m}(t))$ values for each MCS $m \in \mathcal{M}$. That is,

$$\boldsymbol{q}_i(t) = f(\tilde{\boldsymbol{s}}_i(t)), \tag{7}$$

where $\boldsymbol{q}_i(t)$ is an $M \times 1$ vector, defined as

$$\boldsymbol{q}_i(t) = \begin{bmatrix} 1-r_{i,1}(t) \\ 1-r_{i,2}(t) \\ \cdots \\ 1-r_{i,M}(t) \end{bmatrix} \tag{8}$$

If function f is known, one can directly find the optimal MCS based on the BLER target γ in (1). However, it is analytically intractable to determine f since $r_{i,m}(t)$ cannot be modeled by a closed-form mathematical expression. Further, due to the large input space of $\tilde{\boldsymbol{s}}_i(t)$, we cannot model f empirically through simulations or experiments.

The above difficulties motivate us to employ deep learning to approximate f with a DFA μ. This approach allows us to learn a good approximation for f using a relatively small set of $\tilde{\boldsymbol{s}}_i(t)$ samples from the input space. After the learning phase, μ will be able to provide good predictions for $(1-r_{i,m}(t))$ even when the input vectors are outside the sample set. In other words, the DFA μ is capable of achieving statistical generalization (in terms of serving as a function approximation machine) beyond the sample set used for learning [22].

B. Key Idea

The DFA μ that we use to approximate f is a *feedforward neural network* consisting of an input layer, a number of hidden layers and an output layer, as shown in Fig. 7.[5] Within μ's multi-layer structure, there is a set Θ of trainable

[5]A key consideration in choosing the multi-layer structure of μ is that the inference time of μ (forward propagation from input layer to output layer) should meet the real-time requirement of 100 μs. The multi-layer structure used in our experiments will be presented in Section IX-A.

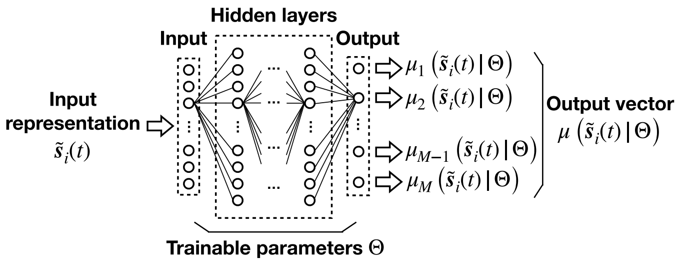

Hidden layers
Input
Output
Input representation $\tilde{s}_i(t)$
$\mu_1(\tilde{s}_i(t)|\Theta)$
$\mu_2(\tilde{s}_i(t)|\Theta)$
$\mu_{M-1}(\tilde{s}_i(t)|\Theta)$
$\mu_M(\tilde{s}_i(t)|\Theta)$
Output vector $\mu(\tilde{s}_i(t)|\Theta)$
Trainable parameters Θ

Fig. 7. Structure of DFA μ.

parameters that determine the mapping from the input to the output of μ. Given an input vector $\tilde{\boldsymbol{s}}_i(t)$, the output of μ is a vector of size M, defined as

$$\mu(\tilde{\boldsymbol{s}}_i(t)|\Theta) = \begin{bmatrix} \mu_1(\tilde{\boldsymbol{s}}_i(t)|\Theta) \\ \mu_2(\tilde{\boldsymbol{s}}_i(t)|\Theta) \\ \cdots \\ \mu_M(\tilde{\boldsymbol{s}}_i(t)|\Theta) \end{bmatrix} \tag{9}$$

where $\mu_m(\tilde{\boldsymbol{s}}_i(t)|\Theta)$ $(m = 1, \cdots, M)$ denotes the m-th output entry in $\mu(\tilde{\boldsymbol{s}}_i(t)|\Theta)$.

Ideally, an optimal set of parameters, denoted by Θ^*, should satisfy:

$$\mathbb{E}\Big[\boldsymbol{q}_i(t) - \mu(\tilde{\boldsymbol{s}}_i(t)|\Theta^*)\Big] = \mathbf{0}, \tag{10}$$

where the expectation $\mathbb{E}[\cdot]$ is with respect to all possible realizations of $\tilde{\boldsymbol{s}}_i(t)$. In this case, $\mu(\tilde{\boldsymbol{s}}_i(t)|\Theta^*)$ can be used as an approximation for $\boldsymbol{q}_i(t)$ for any $e_i(t)$.

Unfortunately, it is impossible to directly learn Θ^* based on (10). This is because for a given input $\tilde{\boldsymbol{s}}_i(t)$, we cannot determine $\boldsymbol{q}_i(t)$ either analytically or empirically.

Nevertheless, (10) points us a direction on how to obtain an approximation for Θ^*. First, we establish a relationship between $(1-r_{i,m}(t))$ (i.e., entries in $\boldsymbol{q}_i(t)$) and the decoding result of $e_i(t)$. Denote $a_{i,m}(t)$ as the binary decoding result for $e_i(t)$ under MCS m as follows:

$$a_{i,m}(t) = \begin{cases} 1, & \text{if decoding of } e_i(t) \text{ is successful,} \\ 0, & \text{otherwise.} \end{cases}$$

Then the decoding success probability for $e_i(t)$ under MCS m is equal to the expectation of $a_{i,m}(t)$, i.e., $\mathbb{E}[a_{i,m}(t)|\tilde{\boldsymbol{s}}_i(t), m]$, where the expectation is conditioned on $\tilde{\boldsymbol{s}}_i(t)$ and m. Since $r_{i,m}(t)$ represents the actual BLER for $e_i(t)$ under MCS m, we have the following relationship between $a_{i,m}(t)$ and $r_{i,m}(t)$:

$$\mathbb{E}[a_{i,m}(t)|\tilde{\boldsymbol{s}}_i(t), m] = 1 - r_{i,m}(t). \tag{11}$$

Then substituting each entry $(1-r_{i,m}(t))$ in $\boldsymbol{q}_i(t)$ with $\mathbb{E}[a_{i,m}(t)|\tilde{\boldsymbol{s}}_i(t), m]$ in (10), we have:

$$\mathbb{E}\Big[\mathbb{E}[a_{i,m}(t)|\tilde{\boldsymbol{s}}_i(t), m] - \mu_m(\tilde{\boldsymbol{s}}_i(t)|\Theta^*)\Big| m\Big] = 0, \tag{12}$$

for each $m \in \mathcal{M}$. But (12) also cannot be used to learn Θ^* since $\mathbb{E}[a_{i,m}(t)|\tilde{\boldsymbol{s}}_i(t), m]$ is unknown (for the same reason as for $\boldsymbol{q}_i(t)$).

Although $\mathbb{E}[a_{i,m}(t)|\tilde{\boldsymbol{s}}_i(t), m]$ in (12) is unknown, $a_{i,m}(t)$ for each TTI is known. So instead of using $\mathbb{E}[a_{i,m}(t)|\tilde{\boldsymbol{s}}_i(t), m]$ in (12), we propose to use $a_{i,m}(t)$ in the learning method.

Our goal is to learn a set of parameters $\hat{\Theta}^*$ to approximate the optimal Θ^*. Specifically, our learning method will ensure that $\hat{\Theta}^*$ satisfies:

$$\mathbb{E}\Big[a_{i,m}(t) - \mu_m(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*)\Big] = 0, \tag{13}$$

where the expectation is taken with respect to $e_i(t)$, MCS m, and $a_{i,m}(t)$. For (13), we have:

$$\begin{aligned}
&\mathbb{E}\Big[a_{i,m}(t) - \mu_m(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*)\Big]\\
&= \mathbb{E}\Big[\mathbb{E}\big[a_{i,m}(t)|\tilde{\boldsymbol{s}}_i(t), m\big] - \mu_m(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*)\Big]\\
&= \mathbb{E}\Big[1 - r_{i,m}(t) - \mu_m(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*)\Big]\\
&= 0.
\end{aligned} \tag{14}$$

Comparing (12) and (14), we can see that Θ^* is more accurate than $\hat{\Theta}^*$ in terms of predicting $(1 - r_{i,m}(t))$. This is because Θ^* guarantees the prediction accuracy for *every* MCS $m \in \mathcal{M}$. In contrast, $\hat{\Theta}^*$ only ensures that the average gap between $(1 - r_{i,m}(t))$ and $\mu_m(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*)$ over *all* MCSs is zero. But as Θ^* cannot be obtained in practice, we will find $\hat{\Theta}^*$ instead and use it as an approximation for Θ^*.

In the rest of this section, we will present the details of our learning method.

C. Some Details

Since the learning for DFA μ is best performed through gradient-descent approaches [22], we propose to learn parameters $\hat{\Theta}^*$ by solving the following mean-squared error (MSE) minimization problem instead of the equation in (13):

$$\text{minimize } G(\Theta) = \mathbb{E}\left[\big(a_{i,m}(t) - \mu_m\left(\tilde{\boldsymbol{s}}_i(t)|\Theta\right)\big)^2\right], \tag{15}$$

where the expectation is with respect to all possible realizations of $e_i(t)$, MCS m, and $a_{i,m}(t)$.

To see why parameters $\hat{\Theta}^*$ as characterized in (13) can be obtained through solving the MSE minimization problem (15), consider the gradient of $G(\Theta)$ with respect to Θ:

$$\begin{aligned}\nabla_\Theta G(\Theta) = \mathbb{E}\Big[&-2\big(a_{i,m}(t) - \mu_m\left(\tilde{\boldsymbol{s}}_i(t)|\Theta\right)\big)\\
&\cdot\nabla_\Theta \mu_m\left(\tilde{\boldsymbol{s}}_i(t)|\Theta\right)\Big].\end{aligned} \tag{16}$$

Assume that $\hat{\Theta}^*$ is the solution to (15). Then it must satisfy $\nabla_\Theta G(\Theta)|_{\Theta=\hat{\Theta}^*} = \mathbf{0}$. In (16), $\nabla_\Theta \mu_m\left(\tilde{\boldsymbol{s}}_i(t)|\Theta\right)$ is a vector of the gradient of $\mu_m\left(\tilde{\boldsymbol{s}}_i(t)|\Theta\right)$ with respect to the parameters in Θ and cannot be forced to be zero. Thus we must have

$$\mathbb{E}\Big[a_{i,m}(t) - \mu_m(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*)\Big] = 0,$$

which is exactly (13).

We now propose a gradient-descent method to learn $\hat{\Theta}^*$. This method consists of a *data collection* phase and a *learning* phase. The data collection phase is to gather a large set of eMBB transmission samples from the cell and store them into a replay buffer. The learning phase is to learn $\hat{\Theta}^*$ through iterative gradient descent based on the collected eMBB samples.

The data collection phase involves a total of T TTIs (e.g., $T = 10^5 \sim 10^6$). In each TTI $t = 1, \cdots, T$, an eMBB transmission sample $(\tilde{\boldsymbol{s}}_t, m_t, a_t)$ (notation is reduced for the ease of exposition) is obtained and stored into the replay buffer. In particular, the MCS m_t for each eMBB transmission sample is chosen from $\mathcal{M}$ following a uniform distribution. Such a uniformly random MCS selection is to ensure that all MCSs in $\mathcal{M}$ can be thoroughly explored during data collection.

Algorithm 1 Data Collection and Learning

```
 % Data Collection:
 1: Empty replay buffer;
 2: for $t = 1, 2, \ldots, T$ do
 3:     Receive eMBB state representation $\tilde{\boldsymbol{s}}_t$;
 4:     Select an MCS $m_t$ from $\mathcal{M}$ following a uniform distribution;
 5:     Transmit eMBB and receive its decoding result $a_t$ via feedback;
 6:     Store sample tuple $(\tilde{\boldsymbol{s}}_t, m_t, a_t)$ in the replay buffer;
 7: end for
 % Learning Phase:
 8: Initialize $\Theta_0$ with random parameters;
 9: for $j = 1, 2, \cdots, J$ do
10:     Randomly choose a mini-batch of $K$ samples from the replay buffer;
11:     Update the parameters $\Theta_j$ using $\text{SG}(\Theta_{j-1})$ following (18);
12: end for
13: return $\hat{\Theta}^* = \Theta_J$.
```

After data collection is completed, we start the learning phase to determine $\hat{\Theta}^*$. The learning phase involves a total of J iterations, where J should be sufficiently large (e.g., $J = 10^5 \sim 10^6$) to achieve good performance. Denote Θ_j as the set of parameters obtained in the j-th iteration. Before the first iteration, we initialize parameters Θ_0 randomly. For the j-th iteration ($j = 1, 2, \cdots, J$), we update parameters Θ_j by taking a small step along the gradient $\nabla_\Theta G(\Theta)|_{\Theta=\Theta_{j-1}}$ given in (16).

Since it is intractable to determine the expectation in $\nabla_\Theta G(\Theta)|_{\Theta=\Theta_{j-1}}$, we perform the *stochastic gradient descent (SGD)* [22] for each update of Θ_j. Specifically, in each iteration, we randomly choose a set of K (e.g., $K = 32$) samples $(\tilde{\boldsymbol{s}}_k, m_k, a_k)$, $k = 1, \cdots, K$ from the replay buffer to approximate $\nabla_\Theta G(\Theta)|_{\Theta=\Theta_{j-1}}$ using the following sample gradient $\text{SG}(\Theta_{j-1})$:[6]

$$\begin{aligned}\text{SG}(\Theta_{j-1}) = \frac{1}{K}\sum_{k=1}^{K}\Big[&-2\big(a_k - \mu_{m_k}\left(\tilde{\boldsymbol{s}}_k|\Theta_{j-1}\right)\big)\cdot\\
&\cdot\nabla_\Theta \mu_{m_k}\left(\tilde{\boldsymbol{s}}_k|\Theta\right)\big|_{\Theta=\Theta_{j-1}}\Big].\end{aligned} \tag{17}$$

Then we update Θ_j as follows:

$$\Theta_j = \Theta_{j-1} + \rho \cdot \text{SG}(\Theta_{j-1}), \tag{18}$$

where ρ denotes the step size (or learning rate), e.g., $\rho = 10^{-3}$. After the J learning iterations, we set $\hat{\Theta}^* = \Theta_J$. A pseudo

[6]The purpose of random sampling is to break the temporal correlations among the samples and improve the stability of learning [25], [26].

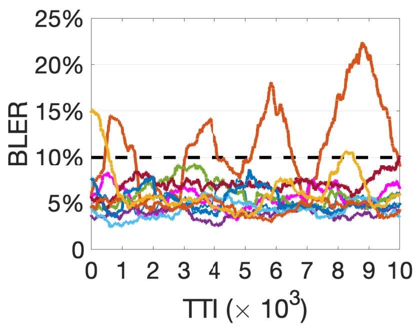


Fig. 8. BLER performance of 10 eMBB users without calibrating μ's approximation error.

code for the data collection and learning phases is given in Algorithm 1.

The time complexity of Algorithm 1 is analyzed as follows. The data collection phase of Algorithm 1 has T iterations. In each iteration, steps in line 3 (receive eMBB state representation) and 4 (randomly select MCS) have $O(1)$ complexity. The step in line 5 has the highest computational complexity as it requires the execution of the entire 5G transmitter and receiver signal processing chains (at both the UE and BS). The complexities of 5G transceiver chains are implementation-specific and a general big-O asymptotic complexity is not available. Therefore, one cannot offer an overall asymptotic time complexity for the data collection phase in big-O form.

The learning phase of Algorithm 1 has J iterations. In each iteration, the step in line 10 (choose a mini-batch of K samples) has a $O(K)$ complexity. The step in line 11 updates the parameters in DFA μ by using the sample gradient $\mathrm{SG}(\Theta_{j-1})$ computed by (17). The computation of $\mathrm{SG}(\Theta_{j-1})$ requires executing both forward-propagation and back-propagation of DFA μ. The time complexities for forward-propagation and back-propagation of a DFA only depend on the number of layers in its structure and the number of neurons per layer. Therefore, with a fixed layered structure of μ, the time complexity of the step in line 11 is $O(K)$. In summary, the time complexity of the learning phase is $O(K \cdot J)$.

VII. Calibration and MCS Selection

A. Motivation

For each eMBB transmission $e_i(t)$, the DFA μ learned in Section VI provides us an approximation $\mu(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*)$ for the $(1 - r_{i,m}(t))$ values in $\boldsymbol{q}_i(t)$ (see (8)). However, due to the nature of DFA, there likely exists approximation error in $\mu(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*)$. As a result, if we directly use $(1-\mu_m(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*))$ to substitute $r_{i,m}(t)$ in MCS selection (based on (1)), the actual BLER performance of the eMBB link may not meet the target γ.

As an example, Fig. 8 shows the BLER curves of 10 eMBB users (see Section IX for parameter settings),[7] where the MCS is selected for each $e_i(t)$ based directly on $(1-\mu_m(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*))$ $(m = 1, \cdots, M)$ and a BLER target $\gamma = 10\%$. As we see in the figure, there are two users' BLER curves (orange and yellow) that violate the target γ during certain periods. This is because during these periods, μ underestimates their $r_{i,m}(t)$ for each MCS and as result, MCS selection based on such $(1 - \mu_m(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*))$'s cannot meet the 10% BLER target.

[7]The BLER curves in Fig. 8 are computed using a sliding window with a size of 1,000 TTIs.

The results in Fig. 8 demonstrate that it is necessary to include a calibration mechanism to address the approximation error of μ. In the rest of this section, we describe our proposed calibration mechanism.

B. Key Idea

For a given $e_i(t)$, the approximation error in each of μ's output entry $\mu_m(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*)$ is:

$$\alpha_m(\tilde{\boldsymbol{s}}_i(t)) = 1 - r_{i,m}(t) - \mu_m(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*).$$

But it is impossible to calculate the approximation error $\alpha_m(\tilde{\boldsymbol{s}}_i(t))$ because we do not know $r_{i,m}(t)$.

Since we cannot find $\alpha_m(\tilde{\boldsymbol{s}}_i(t))$ for a given $\tilde{\boldsymbol{s}}_i(t)$, one may try to find an average approximation error over all possible $\tilde{\boldsymbol{s}}_i(t)$. Denote this average approximation error as α_m. Then we have:

$$\begin{aligned}\alpha_m &= \mathbb{E}\Big[1 - r_{i,m}(t) - \mu_m(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*)\Big|m\Big]\\ &= \mathbb{E}\Big[\mathbb{E}\left[a_{i,m}(t)|\tilde{\boldsymbol{s}}_i(t), m\right] - \mu_m(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*)\Big|m\Big]\\ &= \mathbb{E}\Big[a_{i,m}(t) - \mu_m(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*)\Big|m\Big]\end{aligned}$$

where the second equality follows (11), and the third equality follows the law of total expectation.

Unfortunately, it is not feasible to estimate and calibrate such per-MCS (m) based approximation errors. This is because there is simply not enough decoding results ($a_{i,m}(t)$'s) available for each MCS $m \in \mathcal{M}$ in the desired time scale of our interest. For example, over certain period, there may be no user present in the cell that has strong enough channel quality to support higher MCS levels (e.g., MCS 17-28 with 64-QAM modulation (Table 5.1.3.1-1 in [16]). As a result, we cannot accurately estimate α_m's in a timely fashion.

Although it is infeasible to estimate approximation error for each m, it is possible (and practical) to develop a per-user based approach to calibrate the average approximation error. Specifically, we introduce a calibration parameter β^x for each eMBB user x. Recall that $\upsilon(e_i(t))$ is the receiving eMBB user corresponding to $e_i(t)$ (see Section III). Then we use β^x to calibrate all μ's output entries $\mu_m(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*)$ $(m = 1, 2, \cdots, M)$, if $x = \upsilon(e_i(t))$. The goal is to set β^x properly so that the following mean squared approximation error is minimized for each user x:

$$y^x = \mathbb{E}\Big[\mathbb{E}\Big[\Big(1 - r_{i,m}(t) - \big(\mu_m(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*) + \beta^x\big)\Big)^2\Big|m\Big]\Big]. \tag{19}$$

In (19), the inner expectation (conditioned on a specific MCS m) is with respect to all possible realizations of $\tilde{\boldsymbol{s}}_i(t)$ for user x, and the outer expectation is with respect to all MCS m.

With this calibration, the calibrated BLER prediction $\tilde{r}_{i,m}(t)$ for a given $e_i(t)$ and MCS $m \in \mathcal{M}$ is:

$$\tilde{r}_{i,m}(t) = 1 - \left(\mu_m(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*) + \beta^x\right), \tag{20}$$

where $x = \upsilon(e_i(t))$. Then the MCS selection strategy in (1) becomes

$$\hat{m}_i^*(t) = \arg\max_{m \in \mathcal{M}} \text{SE}(m), \text{ subject to} : \tilde{r}_{i,m}(t) \leq \gamma, \tag{21}$$

where $\hat{m}_i^*(t)$ denotes the MCS level selected for $e_i(t)$.

C. Optimize β^x

The optimization of β^x with the objective of minimizing y^x in (19) can be performed through an iterative gradient descent approach. Specifically, the gradient of y^x with respect to β^x can be obtained as

$$\begin{aligned}\frac{dy^x}{d\beta^x} &= \mathbb{E}\Big[\mathbb{E}\Big[-2\Big(1 - r_{i,m}(t) \\ &\quad - \big(\mu_m(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*) + \beta^x\big)\Big)\Big|m\Big]\Big] \\ &= \mathbb{E}\Big[\mathbb{E}\Big[-2\Big(\mathbb{E}\left[a_{i,m}(t)|\tilde{\boldsymbol{s}}_i(t), m\right] \\ &\quad - \big(\mu_m(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*) + \beta^x\big)\Big)\Big|m\Big]\Big] \\ &= \mathbb{E}\Big[\mathbb{E}\Big[-2\Big(a_{i,m}(t) - \big(\mu_m(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*) + \beta^x\big)\Big)\Big|m\Big]\Big],\end{aligned} \tag{22}$$

where the second equality follows (11), and the third equality follows the law of total expectation. We use the sign function $\text{sgn}(\cdot)$ to indicate whether $\frac{dy^x}{d\beta^x}$ is positive or negative:

$$\text{sgn}(\frac{dy^x}{d\beta^x}) = \begin{cases} 1, & \text{if } \frac{dy^x}{d\beta^x} > 0, \\ 0, & \text{if } \frac{dy^x}{d\beta^x} = 0, \\ -1, & \text{otherwise.} \end{cases}$$

Then in each iteration, we update β^x as follows:

$$\beta^x \leftarrow \beta^x - \text{sgn}(\frac{dy^x}{d\beta^x}) \cdot \Delta, \tag{23}$$

where Δ denotes the step size.

We now show how to incorporate this iterative gradient descent into our 5G setting. Specifically, we show how to implement (23) as a simple additive increase/decrease (AID) mechanism to periodically update β^x for each eMBB user $x \in \mathcal{X}$.

Since the gradient $\frac{dy^x}{d\beta^x}$ cannot be determined analytically, we perform SGD using the recent transmission results of each user x to update β^x periodically. Specifically, at the BS, we maintain a buffer $\mathcal{Q}^x$ for each user x. When the BS receives a decoding result $a_{i,m}(t)$ for a transmission $e_i(t)$, the scheduler pushes the result into the corresponding user $\upsilon(e_i(t))$'s buffer $\mathcal{Q}_{\upsilon(e_i(t))}$. When a new decoding result arrives at the buffer and the buffer is full, the oldest decoding result in the buffer will be pushed out.

For each user x, β^x is updated after every Z transmissions (e.g., $Z = 10$), which correspond to Z decoding results in $\mathcal{Q}^x$.

Algorithm 2 Calibration and MCS Selection

```
1: Initialization: For each eMBB user $x$, set $d^x = 0$, $\beta^x = 0$;
 %MCS selection
2: upon event an $e_i(t)$ is scheduled for transmission do
3:     $x = \upsilon(e_i(t))$;
4:     Obtain input representation $\tilde{\boldsymbol{s}}_i(t)$ as (6);
5:     Obtain the prediction $\mu(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*)$;
6:     Obtain the calibrated BLER $\tilde{r}_{i,m}(t)$ based on $\beta^x$ and (20)
7:     Select MCS $\hat{m}_i^*(t)$ based on (21);
8:     Transmit $e_i(t)$ using MCS $\hat{m}_i^*(t)$;
 %$\beta^x$ adjustment
9: upon event a decoding result $a_{i,m}(t)$ is received do
10:    $x = \upsilon(e_i(t))$;
11:    Push $a_{i,m}(t)$ into the buffer $\mathcal{Q}^x$;
12:    Counter $d^x \leftarrow d^x + 1$;
13:    if $(d^x \bmod Z) = 0$ then
14:        Update $\beta^x$ following (25);
15:    end if
```

Whenever there are Z new results in $\mathcal{Q}^x$, denoted by a_z^x, $z = 1 \cdots, Z$, we compute the sample gradient as follows:

$$\begin{aligned}\text{SY}^x &= \frac{1}{Z}\sum_{z=1}^{Z}[-2\left(a_z^x - (1-\gamma)\right)], \\ &= 2(1 - \gamma - \frac{1}{Z}\sum_{z=1}^{Z} a_z^x) \\ &= 2(1 - \gamma - A^x(Z)),\end{aligned}$$

where $(1-\gamma)$ is used to approximate $\left(\mu_m(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*) + \beta^x\right)$ in (22) for each of the Z transmissions, and $A^x(Z)$ denotes $\sum_{z=1}^{Z} a_z^x/Z$, i.e., the mean of the most recent Z decoding results in $\mathcal{Q}^x$. The use of $(1-\gamma)$ as an approximation follows from (20) and (21), where an MCS m is chosen corresponding to the largest $\tilde{r}_{i,m}(t)$ that is less than γ. Then β^x is updated as follows:

$$\beta^x \leftarrow \beta^x - \text{sgn}(1 - A^x(Z) - \gamma) \cdot \Delta\ . \tag{24}$$

In (24), $(1 - A^x(Z))$ is the empirical decoding failure rate among the Z transmissions.

However, if (24) is directly used to update β^x, β^x will experience frequent fluctuations, bringing BLER of each user frequently exceeding target γ. This is because the sgn function is extremely sensitive: when the decoding failure rate is slightly lower than γ, which is perfectly fine in practice, β^x will still be forced to increase. As a result, the decoding failure rate of the subsequent transmissions will increase immediately, leading to frequent violations of target γ.

To address this undesirable sensitivity in (24), we introduce a parameter γ_0 as a cushion when assessing current decoding failure rate and adjusting β^x. Instead of adjusting β^x when the decoding failure rate falls below γ, we will wait and make adjustment only when the decoding failure rate falls below $(\gamma - \gamma_0)$.

Furthermore, we propose to use two different step sizes, Δ_{D} and Δ_{I}, for the decrease and increase of β^x during adjustment, respectively. In particular, to ensure a fast response in the event that a user's decoding failure rate goes beyond the target γ, we set $\Delta_{\text{D}} \geq \Delta_{\text{I}}$.

Based on the above discussion, we have the following improved adjustment algorithm for β^x:

$$\beta^x \leftarrow \begin{cases} \beta^x - \Delta_{\text{D}}, & \text{if } 1 - A^x(Z) > \gamma, \\ \beta^x + \Delta_{\text{I}}, & \text{if } 1 - A^x(Z) < \gamma - \gamma_0. \end{cases} \tag{25}$$

D. Summary

A pseudo code summarizing the MCS selection and β^x adjustment for each eMBB user $x \in \mathcal{X}$ is given in Algorithm 2.

In Algorithm 2, the time complexity of the MCS selection process is $O(1)$ given that the complexity for forward propagation of DFA μ is only dependent on the number of layers in its structure and the number of neurons per layer, which are fixed during cell operation. The time complexity of β^x adjustment is also $O(1)$ since the parameter Z is a constant.

VIII. Real-Time Implementation

In Sections V to VII, we described in detail the three stages of DELUXE to select an MCS for an eMBB transmission $e_i(t)$. In each TTI, there are usually multiple eMBB transmissions scheduled, i.e., $|\mathcal{E}(t)| > 1$. Given that the shortest duration of a TTI for eMBB is 125 μs (under NR numerology 3), the implementation problem we need to address is to ensure the MCS selection for all $e_i(t) \in \mathcal{E}(t)$ scheduled for a TTI t can be completed within 100 μs time scale.

To address this problem, we propose a hybrid CPU and GPU-based implementation as follows:

Step (i) The input representation vectors $\tilde{\boldsymbol{s}}_i(t)$ for all $e_i(t) \in \mathcal{E}(t)$ are generated by the CPU and then transferred to the GPU.

Step (ii) We use a GPU to compute $\mu(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*)$ for all $e_i(t) \in \mathcal{E}(t)$ concurrently (more details to follow). The motivation of employing a GPU for this step is to exploit parallelism to reduce the computation time. Specifically, the computations of $\mu(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*)$ for all $e_i(t) \in \mathcal{E}(t)$ can be done in parallel through matrix operations. This can be handled much more efficiently by a GPU than a CPU.

Step (iii) Based on the computed $\mu(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*)$ values, MCS selections for all $e_i(t) \in \mathcal{E}(t)$ are performed in parallel on the GPU following (21). Then the result of these selected MCSs is transferred back to the CPU. The update of β^x for each eMBB user x is performed on the CPU and the β^x values are sent to the GPU after each update.

We now describe Step (ii) in detail, i.e., how to compute $\mu(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*)$ for all $e_i(t) \in \mathcal{E}(t)$ concurrently using parallel threads in a GPU. Recall that μ is a fully-connected neural network consisting of an input layer, a number of hidden layers, and an output layer. Denote $\boldsymbol{W}_{(j,j+1)}$ as the weight matrix for the connections from the neurons in the j-th layer to the neurons in the $(j+1)$-th layer. For instance, $\boldsymbol{W}_{(1,2)}$ is the weight matrix for the connections from the input layer to the first hidden layer. The element in the k-th row and the l-th column of the matrix $\boldsymbol{W}_{(j,j+1)}$ is the weight on the connection from the k-th neuron in the j-th layer to the l-th neuron in the $(j+1)$-th layer. Further, denote $\boldsymbol{\pi}_j$ as a vector of the bias added to each neuron in the j-th layer, and $\phi(\cdot)$ as the non-linear activation function (e.g., rectified non-linearity) used for the hidden layers.

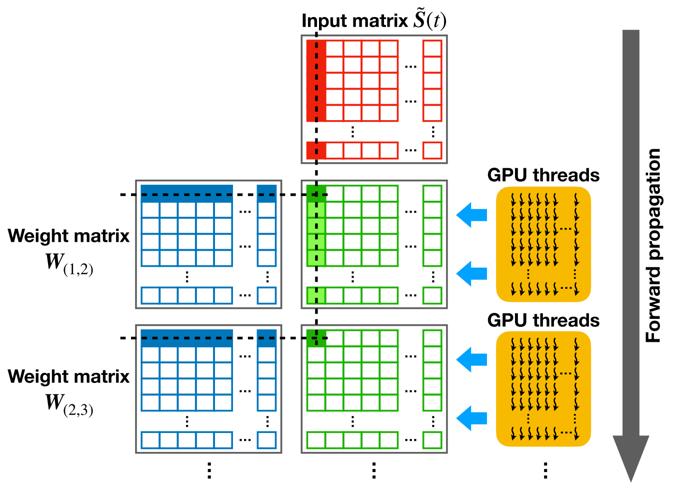

Input matrix $\tilde{\boldsymbol{S}}(t)$
Weight matrix $\boldsymbol{W}_{(1,2)}$
Weight matrix $\boldsymbol{W}_{(2,3)}$
GPU threads
GPU threads
Forward propagation

Fig. 9. Concurrent forward propagation for all $e_i(t) \in \mathcal{E}(t)$.

Consider a specific $e_i(t) \in \mathcal{E}(t)$. The first step to compute $\mu(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*)$ for $e_i(t)$ is to feed the vector $\tilde{\boldsymbol{s}}_i(t)$ into the input layer of μ. Then the forward propagation from the input layer to the first hidden layer is computed as: $\phi(\boldsymbol{W}_{(1,2)} \cdot \tilde{\boldsymbol{s}}_i(t) + \boldsymbol{\pi}_2)$. Subsequently, the forward propagation to the second hidden layer is computed as: $\phi(\boldsymbol{W}_{(2,3)} \cdot \phi(\boldsymbol{W}_{(1,2)} \cdot \tilde{\boldsymbol{s}}_i(t) + \boldsymbol{\pi}_2) + \boldsymbol{\pi}_3))$. Such computations proceed through the remaining layers in μ until we obtain the final output $\mu(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*)$ for $e_i(t)$.

The above forward propagation process for all $e_i(t) \in \mathcal{E}(t)$ can be implemented concurrently through matrix operations leveraging GPU parallel processing. An illustration for this concurrent forward propagation is given in Fig. 9. Specifically, we combine the vectors $\tilde{\boldsymbol{s}}_i(t)$ for all $e_i(t) \in \mathcal{E}(t)$ into a matrix: $\tilde{\boldsymbol{S}}(t) = [\tilde{\boldsymbol{s}}_1(t); \tilde{\boldsymbol{s}}_2(t); \cdots]$, whose number of columns is equal to $|\mathcal{E}(t)|$. Then the forward propagation from the input layer to the first hidden layer for all $e_i(t) \in \mathcal{E}(t)$ can be computed as: $\phi(\boldsymbol{W}_{(1,2)} \cdot \tilde{\boldsymbol{S}}(t) + \boldsymbol{\Pi}_2)$, where $\boldsymbol{\Pi}_2$ is a matrix defined as $\boldsymbol{\Pi}_2 = [\boldsymbol{\pi}_2; \boldsymbol{\pi}_2; \cdots]$. We employ a two-dimensional array of GPU threads to complete this computation, as shown in Fig. 9. Each thread first computes the inner product of a row in $\boldsymbol{W}_{(1,2)}$ and a column in $\tilde{\boldsymbol{S}}(t)$, and then adds the product (a scalar) with the corresponding bias element in $\boldsymbol{\Pi}_2$. Finally, the non-linear activation $\phi(\cdot)$ is applied to the result.

After all threads finish computing $\phi(\boldsymbol{W}_{(1,2)} \cdot \tilde{\boldsymbol{S}}(t) + \boldsymbol{\Pi}_2)$ (a matrix), the forward propagation to the second hidden layer is computed similarly as: $\phi(\boldsymbol{W}_{(2,3)} \cdot \phi(\boldsymbol{W}_{(1,2)} \cdot \tilde{\boldsymbol{S}}(t) + \boldsymbol{\Pi}_2) + \boldsymbol{\Pi}_3)$. Again, we use an array of GPU threads to do this computation. Following the same token, the forward propagation through the subsequent layers in μ proceeds until we obtain an output matrix $[\mu(\tilde{\boldsymbol{s}}_1(t)|\hat{\Theta}^*); \mu(\tilde{\boldsymbol{s}}_2(t)|\hat{\Theta}^*); \cdots]$, whose columns are the vectors $\mu(\tilde{\boldsymbol{s}}_i(t)|\hat{\Theta}^*)$ for all $e_i(t) \in \mathcal{E}(t)$.

TABLE IV
NETWORK SETTINGS

Carrier frequency	4 GHz
NR numerology	Numerology 1 (30 kHz SC spacing)
System bandwidth	20 MHz, 50 RBs in total mBWP: 25 RBs in high frequency range eBWP: 25 RBs in low frequency range
Cell radius	500 meters
eMBB settings	1 TTI = 1 time slot (14 OFDM symbols) BLER target $\gamma = 10\%$ 29 MCS levels from Table 5.1.3.1-1, [16] Full buffer traffic model
URLLC settings	1 mini-slot = 2 OFDM symbols QPSK modulation with a 1/3 LDPC code rate Poisson arrivals of 50-byte packets Each packet transmitted using 25 RBs in mBWP
Fading channel	CDL-C/TDL-C with 300 ns RMS delay spread [29]
Path-loss model	3D UMa NLOS [29]
Channel estimation	Ideal channel estimation
Antenna config.	4 Tx antennas at the BS 2 Rx antennas at each user
BS Tx power	46 dBm
Noise floor	-91.9 dBm

Based on this implementation, the execution time of DELUXE only depends on the size of input data L and the number of eMBB transmissions $|\mathcal{E}(t)|$ in a TTI t, which correspond to the numbers of rows and columns in the matrix $\tilde{\boldsymbol{S}}(t)$.

IX. PERFORMANCE EVALUATION

In this section, we conduct an experimental study to evaluate the performance of DELUXE. We organize this section as follows. In Section IX-A, we describe our experimental platform and system settings. In Section IX-B, we use a case study to validate the efficacy of DELUXE in solving the MCS selection problem in (1). In Section IX-C, we further evaluate the performance of DELUXE under varying parameters to further understand its behavior.

A. Experimental Platform and Settings

1) Platform: All experiments are completed on an NVIDIA DGX station with an Intel Xeon E5-2698 v4 CPU and an NVIDIA Tesla V100 GPU. Programming on GPU is based on NVIDIA CUDA v10.2 [27] with C++. Data communication between CPU and GPU is through a PCIe 3.0 X16 slot with default configuration.

For all experiments, we built a link-level 5G NR simulator that supports dynamic eMBB/URLLC multiplexing based on URLLC puncturing. The PHY layer signal processing functions and the multi-path channel fading models (CDL and TDL) are provided by MathWorks 5G Toolbox [28].

2) Network Settings: We consider an NR cell consisting of a BS and a varying number of eMBB users that are randomly located within the cell's coverage. Table IV list key parameter settings for the NR cell. Since the carrier frequency is 4 GHz, we employ NR numerology 1 (30 kHz SC spacing) for the channel [51]. Thus, for 20 MHz channel bandwidth, we have 50 RBs. To decouple the impact of a scheduler algorithm on the performance of DELUXE, we assume in each TTI t, an RB is randomly allocated to the eMBB users following a uniform distribution. An RB can only be allocated to one eMBB user while an eMBB user may be allocated with multiple RBs. The target BLER for eMBB is set to $\gamma = 10\%$.

We assume URLLC traffic in the cell follows a Poisson arrival process with a rate of λ (packets per TTI). Following NR standards [5], we set 2 OFDM symbols per mini-slot, a packet size of 50 bytes, QPSK modulation and a 1/3 LDPC code rate for URLLC transmission. Under this setting, each URLLC packet is transmitted using 25 RBs in a mini-slot. Accordingly, we set mBWP (see Fig. 1) to 25 RBs.

For multi-path channel fading, we consider two widely used channel models: the *tapped delay line* (TDL) model and the *clustered delay line* (CDL) model [29]. TDL model contains power, delay and Doppler spectrum information for the multi-path channel taps. CDL is a spatial extension of TDL that involves more detailed information such as the departure and arrival angles for each cluster of channel taps. We choose CDL-C and TDL-C channel profiles that are constructed for non-line-of-sight (NLOS) channel conditions.

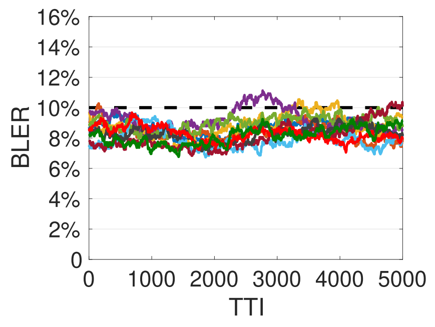
3) DELUXE Settings: We now describe the setting for each of the three stages of DELUXE (see Fig. 5) as follows.

For Stage I, we employ the vector $\boldsymbol{\xi}$ that is optimized in Section V-C.

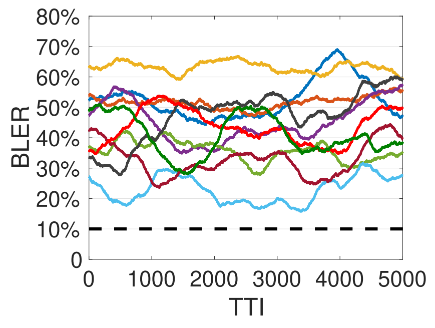
For Stage II, the DFA μ has two fully-connected hidden layers between its input and output layers, which contain 256 and 128 neurons, respectively. We use rectified non-linearity as the activation function for the neurons in the two hidden layers. To learn parameters $\hat{\Theta}^*$ for μ (refer to Algorithm 1), we use $T = 8 \times 10^5$ TTIs to collect data from 10 eMBB users that randomly roam within the cell. Random RB allocation and random MCS selection are employed for scheduling transmissions to the 10 users. Specifically, we collect data for 2×10^5 TTIs under URLLC arrival rate $\lambda = 1$, 2, 3 and 4, respectively. The replay buffer size is set to 8×10^5, which is sufficient to store all collected data. The learning phase involves $J = 4 \times 10^6$ iterations. We use the Adam algorithm [30] with a learning rate of $\rho = 2.5 \times 10^{-4}$ for learning. We set a mini-batch to $K = 64$ samples for each learning iteration.

For Stage III, we set $Z = 10$, $\Delta_{\text{I}} = 2\%$, $\Delta_{\text{D}} = 2.5\%$ and $\gamma_0 = 3\%$.

4) Benchmarking: For performance comparison, we also implemented the EESM method as described in [10], [45] in our 5G simulator. EESM is a popular CQI-based LA method for wideband OFDMA systems such as 4G LTE and 5G NR [46]–[48]. The basic idea of EESM is to map a vector of sub-carrier SNRs that are experienced by a codeword into a single *effective SNR* value through exponential functions. Then this effective SNR is used to find the best MCS to meet a given BLER target γ based on the BLER performance curve under each MCS that are measured under additive white Gaussian noise (AWGN) channels. In the absence of URLLC puncturing, EESM can effectively maintain link reliability.

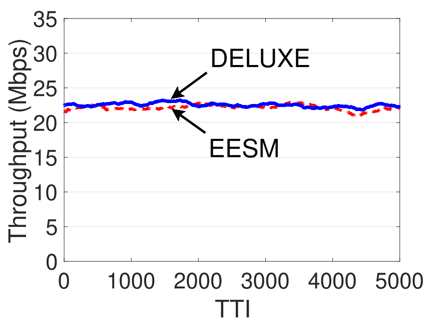


(a) DELUXE

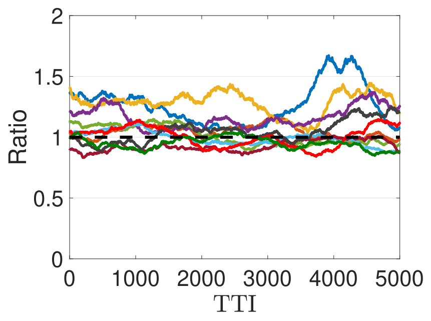


(b) EESM

Fig. 10. BLER performance under (a) DELUXE and (b) EESM.



(a) Sum of throughput of 10 eMBB users achieved by DELUXE and EESM.



(b) Normalized throughput of each user under DELUXE with respect to that under EESM.

Fig. 11. Throughput comparison of DELUXE and EESM.

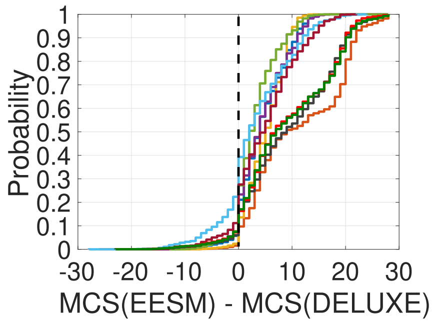


Fig. 12. CDF of the MCS differences between EESM and DELUXE for each of the 10 users.

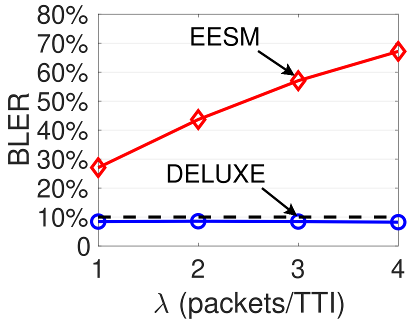
B. A Case Study

We now use a case study to evaluate the performance of DELUXE. In this study, we consider $|\mathcal{X}| = 10$ eMBB users in the cell and a URLLC traffic load $\lambda = 2$ packets/TTI. CDL-C channel model is employed in the simulations. In each TTI, DELUXE and EESM experience the same channel conditions, RB allocation to eMBB users, and URLLC packet arrivals.

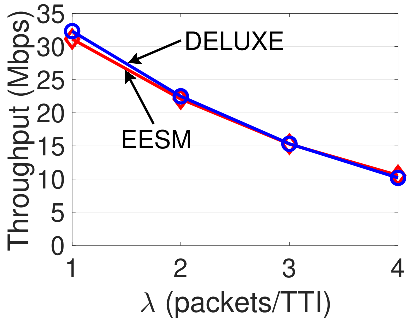
In Fig. 10(a) and (b), we show the BLER performance for the 10 eMBB users over $5,000$ TTIs under DELUXE and EESM, respectively. In both figures, BLER values are calculated using a sliding window with a size of $1,000$ TTIs. In Fig. 10(a), we see that DELUXE is able to keep BLER for each user under $\gamma = 10\%$ most of the time. Although several users' BLERs occasionally exceed 10%, they quickly fall back under $\gamma = 10\%$, thanks to the adaptive β^x-based calibration to the MCS selection (Stage III of DELUXE). In contrast, Fig. 10(b) shows that under EESM, all users' BLERs exceed (violate) the 10% target threshold throughout 5,000 TTIs. The main reason in EESM's failure is that the effective SNR metric used in EESM cannot capture the impact of URLLC puncturing on each eMBB transmission.

In Fig. 11(a) and (b), we show the throughput performance of the 10 eMBB users under DELUXE and compare it to EESM. Note that throughput here only refers to throughput that is *successfully* decoded at a receiver. Fig. 11(a) shows the sum of throughput of the 10 eMBB users achieved by the two schemes. Clearly, there is little difference between the two. Further, in Fig. 11(b), we examine the throughput of each individual eMBB user's throughput under DELUXE by normalizing it with respect to that under EESM. Again, the normalized throughput for each user is close to 1, indicating that throughput at per user level is also comparable under the two schemes. These results confirm that while meeting the target $\gamma = 10\%$ BLER threshold, DELUXE can still achieve comparable eMBB throughput as that under EESM, which is not constrained by the BLER constraints. Note that EESM is the state-of-the-art LA method used in 4G LTE and 5G NR and has been well regarded as the most effective method to maintain link reliability while maximizing spectral efficiency (in the absence of URLLC puncturing) [10], [45].

To see how MCS selection differs under the two schemes, we show the cumulative distribution functions (CDFs) of difference of MCS levels between EESM and DELUXE for the 10 eMBB users in Fig. 12. In this figure, when MCS difference is positive (negative), it means that the MCS level selected by EESM is higher (lower) than that selected by DELUXE. We can see that for each of the 10 users, EESM has a very high probability to select MCS levels higher than DELUXE. This corroborates with the results in Fig. 10(a) and (b), which show that BLER under EESM is

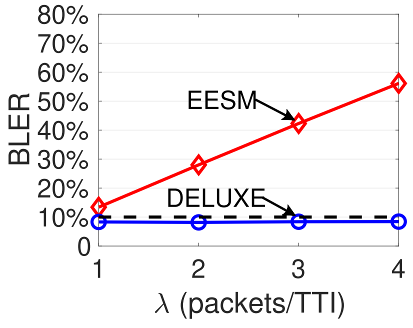


(a) BLER vs. λ

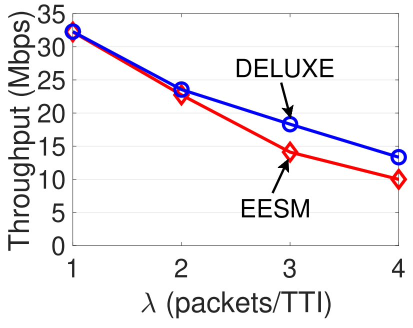


(b) Sum throughput vs. λ

Fig. 13. BLER and sum throughput performance under varying λ under DELUXE and EESM. Multi-path channel model is CDL-C.



(a) BLER vs. λ



(b) Sum throughput vs. λ

Fig. 14. BLER and sum throughput performance under varying λ under DELUXE and EESM. Multi-path channel model is TDL-C.

much higher than that under DELUXE. On the other hand, although EESM leads to high BLERs, it is able to deliver a larger amount of information bits per successfully decoded TB. Such a tradeoff leads to a competitive throughput performance for EESM (similar to that under DELUXE) as shown in Fig. 11(a). However, EESM cannot provide a guarantee to BLER that is required in 5G NR standards.

Finally, the total execution time of DELUXE for 10 eMBB is 90.6 μs, which meets the 5G real-time requirement (~100 μs for numerology 3). The total execution time of DELUXE includes the computation time at both CPU and GPU, and the time cost for data transferring between CPU and GPU.

C. Varying Network Parameters

In this section, we examine the behavior of DELUXE by varying different parameters such as URLLC traffic arrival rate λ, multi-path channel models, and the number of eMBB users in the cell.

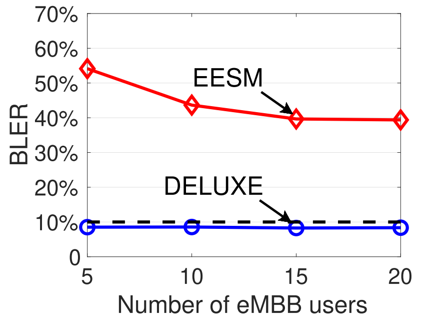
1) Varying URLLC Traffic Load: We vary URLLC traffic arrival rate λ for the same 10 eMBB users in the cell. We assume CDL-C multi-path channel fading model. In Fig. 13(a), we plot the average BLER among the 10 users over $5,000$ TTIs achieved by our DELUXE algorithm. We can see that the average BLER under DELUXE always stays below the $\gamma = 10\%$ target when λ increases. Also shown in Fig. 13(a) is the average BLER among the same 10 users under the EESM algorithm. Clearly, BLER under EESM exceeds (violates) the 10% target and increases with λ.

In Fig. 13(b), we plot the average sum of throughput of the 10 eMBB users under DELUXE over $5,000$ TTIs. Each point in the figure is the sum of each of the 10 eMBB user's average throughput over the $5,000$ TTIs. We can see the sum of eMBB throughput under DELUXE decreases as λ increases. This is because the impact of preemptive puncturing on eMBB is more profound as λ increases. Also shown in Fig. 13(b) is the sum eMBB throughput under EESM. We see that DELUXE's sum eMBB throughput is comparable to EESM, which is consistent to what we have seen in Fig. 11(a).

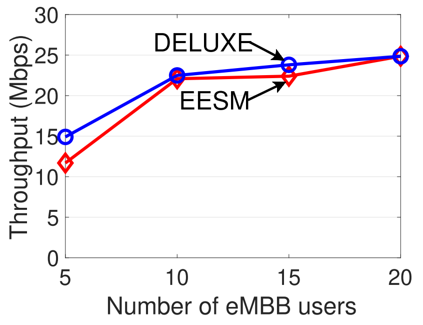
2) Varying Channel Models: We now change the multi-path channel model in the last experiment from CDL-C to TDL-C repeat the same process (i.e., examining DELUXE's performance for varying λ). In Fig. 14(a), we find that average BLER under DELUXE still stays below the $\gamma = 10\%$ target, despite a change of channel model. Also shown in Fig. 14(a) is the average BLERs under EESM. Comparing to that in Fig. 13(a) for EESM, the average BLER performs better under TDL-C model. However, it still exceeds (violates) the 10% target threshold significantly when $\lambda \geq 2$.

From Fig. 14(a), we can see that as λ increases, EESM's BLER appears to increase linearly. This is rather interesting but unfortunately we do not have a concrete explanation to support such linear relationship. To fully understand EESM's BLER as a function of URLLC packet arrival rate, one would need a rigorous model of LDPC decoding behavior under the impact of preemptive puncturing. But it is well-known that LDPC decoding behavior cannot be modeled explicitly with a closed-form expression. Therefore, we will leave the exploration of such relationship for future work.

In Fig. 14(b), we plot the sum of average throughput of the 10 eMBB users under DELUXE. The observations are similar to that for Fig. 13(b) and we omit to repeat the discussion.



(a) BLER vs. number of eMBB users



(b) Sum throughput vs. number of eMBB users

Fig. 15. BLER and sum throughput performance under varying number of eMBB users under DELUXE and EESM.

3) Varying Number of Users: We vary the number of eMBB users $|\mathcal{X}|$ in the cell and evaluate the performance of DELUXE. We assume CDL-C multi-path channel model and $\lambda = 2$. In Fig. 15(a), we plot the average BLER of DELUXE over $5,000$ TTIs under varying number of eMBB users. We see that DELUXE successfully stays below the 10% BLER target while the average BLER of EESM exceeds (violates) the 10% threshold substantially. In Fig. 15(b), we show the sum of average throughput over the $5,000$ TTIs achieved by DELUXE. Again, DELUXE is able to achieve comparable throughput to that by the EESM scheme under different number of users. Note that the throughput of DELUXE increases with the number of users. This is because as the number of user increases, the average number of RBs allocated to each user decreases. Since each user can only choose one MCS for all its allocated RBs, the fewer number of RBs allocated to him, the few number of joint channel conditions needs to be considered. Therefore, with fewer number of RBs per users, it is more likely to choose a higher MCS level for a user (under a random RB allocation).

4) Execution Time of DELUXE: We now investigate the timing performance of DELUXE for MCS selection. Note that the execution time of DELUXE is independent of all other link and PHY layer components (i.e., RB allocation, modulation/demodulation, coding/decoding, among others). So instead of running the full link-level NR simulator (for all link and PHY layer components) as in the previous experiments (which is very time consuming), it is sufficient to run only the DELUXE component.

Recall in Section VIII, the execution time of DELUXE only depends on the size of input data L and the number of eMBB transmissions $|\mathcal{E}(t)|$ in a TTI t. Since $L = 100$ is fixed, we vary the number of eMBB transmissions per TTI and study the timing performance of DELUXE. For a given number of eMBB transmissions per TTI, we randomly generate the input data $\tilde{S}(t)$ to feed DELUXE (see Fig. 9).

TABLE V
EXECUTION TIME OF DELUXE. RESULTS ARE AVERAGED OVER 10^5 TTIS

# of transmissions	10	20	30	40	50
Time (μs)	90.8	91.2	103.5	104.2	107.5

Table V shows the execution time of DELUXE under different numbers of eMBB transmissions per TTI (10 to 50). In all cases, DELUXE's execution time is under 125 μs and meets the real-time requirement in 5G (under Numerology 3).

D. Summary of Results

The advantage of EESM is that it is a simple and straightforward mapping function that can translate high-dimensional channel condition information on a large number of SCs into a single CQI metric that can be used to determine an MCS level. The downside of EESM is that it suffers from a significant loss of channel condition information, which leads to performance issues in MCS selection and BLER guarantee. As shown in the above results, EESM results in a violation of the target BLER by a large margin under the impact of URLLC preemptive puncturing. Thus EESM is not suitable for dynamic eMBB/URLLC multiplexing based on puncturing in 5G.

In contrast, our DELUXE design employs a DFA μ in MCS selection that can make better use of dynamic channel condition and URLLC puncturing information than EESM. Although the forward propagation of DFA μ requires a higher execution time, our GPU-based implementation of DELUXE presented in Section VIII ensures that our MCS selection method can meet the real-time requirement on $\sim$100 μs time scale in 5G NR. In addition, the learning process of DFA μ can be done entirely offline, which will not bring in any time overhead to real-time MCS selection. Our evaluation results confirm that DELUXE can meet a target BLER for eMBB under the impact of URLLC puncturing while achieving throughput performance no worse than EESM.

X. CONCLUSION

In this paper, we investigated dynamic multiplexing of eMBB and URLLC through preemptive URLLC puncturing. We studied the problem of maximizing the throughput of eMBB (through MCS selection) while ensuring link reliability meets target requirement. We proposed DELUXE – a DL-based solution to the eMBB MCS selection under URLLC puncturing. The design of DELUXE includes: (i) a novel mapping to translate high dimensional information (channel condition and URLLC puncturing) on an eMBB code block into a low-dimensional representation with minimal information loss, (ii) a learning method to learn and predict the BLERs of an eMBB transmission under each MCS level based on the decoding results of eMBB training data, (iii) a fast calibration mechanism to offset potential prediction error based on recent decoding results, and (iv) a hybrid CPU and GPU-based implementation of DELUXE that can meet the real-time requirement in 5G NR. We implemented DELUXE on a 5G NR link-level simulator. Through extensive experimental results, we found that DELUXE can successfully maintain the desired link reliability for eMBB services while EESM fails to do so. Further, achieved throughput by DELUXE (via MCS selection) is comparable to EESM, which is considered the most spectral efficient LA approach for 4G LTE and 5G NR.

REFERENCES

[1] Y. Huang, Y. T. Hou, and W. Lou, "A deep-learning-based link adaptation design for eMBB/URLLC multiplexing in 5G NR," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2021, pp. 1–10.

[2] *Study on Scenarios and Requirements for Next Generation Access Technologies*, document TR 38.913, Version 15.0.0, 3GPP, Jul. 2018. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2996

[3] *Minimum Requirements Related to Technical Performance for IMT-2020 Radio Interface(s)*, document ITU-R M.2410-0, Nov. 2017. [Online]. Available: https://www.itu.int/dms_pub/itu-r/opb/rep/R-REP-M.2410-2017-PDF-E.pdf

[4] *Study on Physical Layer Enhancements for NR Ultra-Reliable and Low Latency Case (URLLC)*, document TR 38.824, Version 16.0.0, 3GPP, Mar. 2019. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3498

[5] *NR; Physical Channels and Modulation*, document TS 38.211, Version 16.2.0, 3GPP, Jul. 2020. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3213

[6] *Study on New Radio Access Technology; Radio Interface Protocol Aspects*, document TR 38.804, Version 14.0.0, 3GPP, Mar. 2017. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3070

[7] *Final Report of*, TSG RAN WG1 Meeting #88 Version 1.0.0, 3GPP, Athens, Greece, Feb. 2017. [Online]. Available: https://www.3gpp.org/ftp/TSG_RAN/WG1_RL1/TSGR1_88/Report/

[8] G. Ku and J. M. Walsh, "Resource allocation and link adaptation in LTE and LTE advanced: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 3, pp. 1605–1633, 3rd Quart., 2015.

[9] S. Sesia, I. Toufik, and M. Baker, *LTE-The UMTS Long Term Evolution: From Theory to Practice*. New York, NY, USA: Wiley, 2009.

[10] J. Francis and N. B. Mehta, "EESM-based link adaptation in point-to-point and multi-cell OFDM systems: Modeling and analysis," *IEEE Trans. Wireless Commun.*, vol. 13, no. 1, pp. 407–417, Jan. 2014.

[11] S. Simoens, S. Rouquette-Léveil, P. Sartori, Y. Blankenship, and B. Classon, "Error prediction for adaptive modulation and coding in multiple-antenna OFDM systems," *Signal Process.*, vol. 86, no. 8, pp. 1911–1919, Aug. 2006.

[12] T. L. Jensen, S. Kant, J. Wehinger, and B. H. Fleury, "Fast link adaptation for MIMO OFDM," *IEEE Trans. Veh. Technol.*, vol. 59, no. 8, pp. 3766–3778, Oct. 2010.

[13] R. C. Daniels, "Machine learning for link adaptation in wireless networks," Ph.D. dissertation, Dept. Elect. Comput. Eng., UT Austin, Austin, TX, USA, Dec. 2011. [Online]. Available: http://hdl.handle.net/2152/ETD-UT-2011-12-4509

[14] Z. Dong, J. Shi, W. Wang, and X. Gao, "Machine learning based link adaptation method for MIMO system," in *Proc. IEEE 29th Annu. Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Bologna, Italy, Sep. 2018, pp. 1226–1231.

[15] L. Zhang, J. Tan, Y.-C. Liang, G. Feng, and D. Niyato, "Deep reinforcement learning-based modulation and coding scheme selection in cognitive heterogeneous networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 6, pp. 3281–3294, Jun. 2019.

[16] *NR; Physical Layer Procedures for Data*, document TS 38.214, Version 16.2.0, 3GPP, Jul. 2020. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3216

[17] *NR; Physical Layer Procedures for Control*, document TS 38.213, Version 16.2.0, 3GPP, Jul. 2020. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3215

[18] C.-P. Li, J. Jiang, W. Chen, T. Ji, and J. Smee, "5G ultra-reliable and low-latency systems design," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Oulu, Finland, Jun. 2017, pp. 1–5.

[19] *NR; Multiplexing and Channel Coding*, document TS 38.212, Version 15.5.0, 3GPP, Mar. 2019. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3214

[20] S.-B. Lee, S. Choudhury, A. Khoshnevis, S. Xu, and S. Lu, "Downlink MIMO with frequency-domain packet scheduling for 3GPP LTE," in *Proc. IEEE 28th Conf. Comput. Commun. (INFOCOM)*, Rio de Janeiro, Brazil, Apr. 2009, pp. 1269–1277.

[21] Y. Huang, S. Li, Y. T. Hou, and W. Lou, "GPF: A GPU-based design to achieve ∼100 μs scheduling for 5G NR," in *Proc. ACM MobiCom*, New Delhi, India, Oct./Nov. 2018, pp. 207–222.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[23] F. Capozzi, G. Piro, L. A. Grieco, G. Boggia, and P. Camarda, "Downlink packet scheduling in LTE cellular networks: Key design issues and a survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 678–700, 2nd Quart., 2013.

[24] M. A. Albreem, M. Juntti, and S. Shahabuddin, "Massive MIMO detection techniques: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3109–3132, 4th Quart., 2019.

[25] V. Mnih *et al.*, "Playing Atari with deep reinforcement learning," in *Proc. Workshop Deep Learn. NIPS*, 2013, pp. 1–9.

[26] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–14.

[27] NVIDIA. *CUDA C++ Programming Guide V11.5.0*. [Online]. Available: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

[28] MathWorks. (2019). *5G Toolbox Documentation*. [Online]. Available: https://www.mathworks.com/help/5g/

[29] *Study on Channel Model for Frequencies From 0.5 to 100 GHz*, document TR 38.901, Version 15.0.0, 3GPP, Jun. 2018. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3173

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–15.

[31] A. Anand, G. De Veciana, and S. Shakkottai, "Joint scheduling of URLLC and eMBB traffic in 5G wireless networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Honolulu, HI, USA, Apr. 2018, pp. 1970–1978.

[32] A. K. Bairagi, M. S. Munir, M. Alsenwi, N. H. Tran, and C. S. Hong, "A matching based coexistence mechanism between eMBB and uRLLC in 5G wireless networks," in *Proc. ACM/SIGAPP Symp. Appl. Comput. (SAC)*, Limassol, Cyprus, Apr. 2019, pp. 2377–2384.

[33] A. K. Bairagi *et al.*, "Coexistence mechanism between eMBB and uRLLC in 5G wireless networks," *IEEE Trans. Commun.*, vol. 69, no. 3, pp. 1736–1749, Mar. 2021.

[34] M. Alsenwi, N. H. Tran, M. Bennis, A. K. Bairagi, and C. S. Hong, "EMBB-URLLC resource slicing: A risk-sensitive approach," *IEEE Commun. Lett.*, vol. 23, no. 4, pp. 740–743, Apr. 2019.

[35] Y. Huang, S. Li, C. Li, Y. T. Hou, and W. Lou, "A deep-reinforcement-learning-based approach to dynamic eMBB/URLLC multiplexing in 5G NR," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6439–6456, Jul. 2020.

[36] M. Alsenwi, N. H. Tran, M. Bennis, S. R. Pandey, A. K. Bairagi, and C. S. Hong, "Intelligent resource slicing for eMBB and URLLC coexistence in 5G and beyond: A deep reinforcement learning based approach," *IEEE Trans. Wireless Commun.*, vol. 20, no. 7, pp. 4585–4600, Jul. 2021.

[37] K. I. Pedersen, G. Pocovi, and J. Steiner, "Preemptive scheduling of latency critical traffic and its impact on mobile broadband performance," in *Proc. IEEE 87th Veh. Technol. Conf. (VTC Spring)*, Porto, Portugal, Jun. 2018, pp. 1–6.

[38] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Artificial neural networks-based machine learning for wireless networks: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3039–3071, 4th Quart., 2019.

[39] T. Yu, X. Wang, and A. Shami, "UAV-enabled spatial data sampling in large-scale IoT systems using denoising autoencoder neural network," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1856–1865, Apr. 2019.

[40] H. He, C.-K. Wen, S. Jin, and G. Y. Li, "Deep learning-based channel estimation for beamspace mmWave massive MIMO systems," *IEEE Wireless Commun. Lett.*, vol. 7, no. 5, pp. 852–855, Oct. 2018.

[41] M. Chen, W. Saad, C. Yin, and M. Debbah, "Echo state networks for proactive caching in cloud-based radio access networks with mobile users," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 3520–3535, Jun. 2017.

[42] J. Cui, Y. Liu, and A. Nallanathan, “Multi-agent reinforcement learning-based resource allocation for UAV networks,” *IEEE Trans. Wireless Commun.*, vol. 19, no. 2, pp. 729–743, Feb. 2020.

[43] O. Naparstek and K. Cohen, “Deep multi-user reinforcement learning for distributed dynamic spectrum access,” *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 310–323, Jan. 2019.

[44] J. Zhu, Y. Song, D. Jiang, and H. Song, “A new deep-Q-learning-based transmission scheduling mechanism for the cognitive Internet of Things,” *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2375–2385, Aug. 2018.

[45] R. Giuliano and F. Mazzenga, “Dimensioning of OFDM/OFDMA-based cellular networks using exponential effective SINR,” *IEEE Trans. Veh. Technol.*, vol. 58, no. 8, pp. 4204–4213, Oct. 2009.

[46] W. Anwar, K. Kulkarni, N. Franchi, and G. Fettweis, “Physical layer abstraction for ultra-reliable communications in 5G multi-connectivity networks,” in *Proc. IEEE 29th Annu. Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Bologna, Italy, Sep. 2018, pp. 1–6.

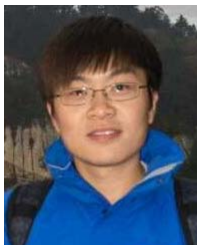
[47] S. Lagen, K. Wanuga, H. Elkotby, S. Goyal, N. Patriciello, and L. Giupponi, “New radio physical layer abstraction for system-level simulations of 5G networks,” in *Proc. IEEE Int. Conf. Commun. (ICC)*, Dublin, Ireland, Jun. 2020, pp. 1–7.

[48] V. Kumar and N. B. Mehta, “Exploiting correlation with wideband CQI and making differential feedback overhead flexible in 4G/5G OFDM systems,” *IEEE Trans. Wireless Commun.*, vol. 20, no. 4, pp. 2579–2591, Apr. 2021.

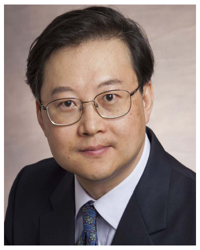
[49] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Hoboken, NJ, USA: Wiley, 2006.

[50] *Feasibility Study on New Services and Markets Technology Enablers; Stage 1*, document TR 22.891, Version 14.2.0, 3GPP, Sep. 2016. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2897

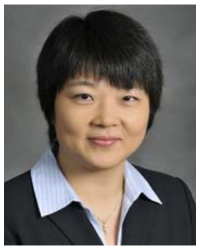
[51] “Making 5G NR a reality,” Qualcomm, San Diego, CA, USA, White Paper, Dec. 2016. [Online]. Available: https://www.qualcomm.com/media/documents/files/whitepaper-making-5g-nr-a-reality.pdf



Yan Huang (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from the Beijing University of Posts and Telecommunications in 2012 and 2015, respectively, and the Ph.D. degree in electrical engineering from Virginia Tech in 2020. He is currently a Senior System Software Engineer at NVIDIA Corporation, Santa Clara, CA, USA. He holds two U.S. and international patents. His research interests include GPU-accelerated real-time optimizations and machine learning for wireless communications and networking. During his Ph.D. study at Virginia Tech, he was awarded a Pratt Scholarship in 2019 and a Bindi Prasad Scholarship in 2020.



Y. Thomas Hou (Fellow, IEEE) is currently a Bradley Distinguished Professor of electrical and computer engineering at Virginia Tech, Blacksburg, VA, USA, which he joined in 2002. He has published over 300 papers in IEEE/ACM journals and conferences. His papers were recognized by nine best paper awards from IEEE and ACM. He holds six U.S. patents. He authored/coauthored two graduate textbooks: *Applied Optimization Methods for Wireless Networks* (Cambridge University Press, 2014) and *Cognitive Radio Communications and Networks: Principles and Practice* (Academic Press/Elsevier, 2009). His current research focuses on developing innovative solutions to complex science and engineering problems arising from wireless and mobile networks. He is also interested in wireless security. He was named as an IEEE Fellow for contributions to modeling and optimization of wireless networks. He was a member of the IEEE Communications Society Board of Governors. He was the Steering Committee Chair of IEEE INFOCOM Conference. He was/is on the editorial boards of a number of IEEE and ACM TRANSACTIONS and journals. He was a Distinguished Lecturer of the IEEE Communications Society.



Wenjing Lou (Fellow, IEEE) is currently the W. C. English Endowed Professor of computer science at Virginia Tech. Her research interests cover many topics in the cybersecurity field, with her current research interests focusing on wireless network security, trustworthy AI, blockchain, and security and privacy problems in the Internet of Things (IoT) systems. She is a Highly Cited Researcher by the Web of Science Group. She is a Steering Committee Member of IEEE INFOCOM and IEEE TRANSACTIONS ON MOBILE COMPUTING. She served as a Program Director for the U.S. National Science Foundation (NSF) from 2014 to 2017. She received the Virginia Tech Alumni Award for Research Excellence in 2018. She received the INFOCOM Test-of-Time Paper Award in 2020. She was the TPC Chair for IEEE INFOCOM 2019 and ACM WiSec 2020. She was the Steering Committee Chair for IEEE CNS Conference from 2013 to 2020.