DRAGON: A DRL-based MIMO Layer and MCS Adapter in Open RAN 5G Networks

Qing An
Rice University
USA

Roy Yang
Mavenir Systems Inc.
USA

Kamakshi Sridhar
Mavenir Systems Inc.
USA

Rahman Doost-Mohammady
Rice University
USA

Abstract

In the rapidly evolving field of wireless communication, Multiple Input Multiple Output (MIMO) networks have emerged as a pivotal technology, offering enhanced data rates and spectral efficiency by leveraging multiple antennas at both the transmitter and receiver. The introduction of Open Radio Access Network (O-RAN) architecture has further revolutionized this domain, enabling greater flexibility, scalability, and interoperability through its open interfaces and software-defined approach. This paper presents DRAGON, a novel Deep Reinforcement Learning (DRL)-based framework for joint Layer and Modulation and Coding Scheme (MCS) selection, tailored for downlink single-user MIMO networks under the O-RAN framework. Our approach is designed to be highly scalable, capable of efficiently managing a large number of configuration options in one-shot prediction, including up to 25 MCS and 4 layer choices. The proposed solution has been rigorously evaluated using an O-RAN-based simulation environment, demonstrating up to an 18% performance improvement over the state-of-the-art (SOTA) methods and achieving a best throughput of 87.4% when compared to the collected ground-truth dataset. Furthermore, our method supports real-time prediction, making it viable for practical deployment. In addition to these advancements, we explore the potential integration of our DRL-based solution with real-world platforms and discuss the extension of our approach to handle multi-user (MU) MIMO scenarios, paving the way for broader applications in next-generation wireless networks.

CCS Concepts

• **Networks** → **Network management**.

Keywords

Open RAN, Reinforcement Learning, Modulation and Code Scheme, MIMO

ACM MobiCom '24, November 18–22, 2024, Washington D.C., DC, USA

ACM ISBN 979-8-4007-0489-5/24/11
https://doi.org/10.1145/3636534.3701549

ACM Reference Format:
Qing An, Roy Yang, Kamakshi Sridhar, and Rahman Doost-Mohammady. 2024. DRAGON: A DRL-based MIMO Layer and MCS Adapter in Open RAN 5G Networks. In *The 30th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '24), November 18–22, 2024, Washington D.C., DC, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3636534.3701549

1 Introduction

The increasing demand for higher data rates and improved spectral efficiency in wireless communication systems has driven the adoption of advanced technologies such as Multiple-Input Multiple-Output (MIMO). MIMO systems, by utilizing multiple antennas at both the transmitter and receiver, significantly enhance the capacity and reliability of wireless networks. The ability of MIMO to exploit spatial diversity and multiplexing gain has made it a cornerstone in modern communication standards, including 5G and beyond. In parallel, the Open Radio Access Network (Open RAN) initiative has emerged as a promising approach to revolutionize the traditional Radio Access Network (RAN) architecture. Open RAN promotes the disaggregation of RAN functions, enabling interoperability, flexibility, and innovation by utilizing open interfaces and standardized software-defined components. This paradigm shift not only reduces costs but also accelerates the deployment of new features and services in wireless networks. Within the MIMO framework, the selection of modulation and coding schemes (MCS) and the determination of the number of transmission layers are critical for optimizing network performance. The MCS determines the trade-off between data rate and robustness against channel impairments, while the layer selection in MIMO dictates the number of parallel data streams transmitted, influencing both throughput and reliability. The challenge lies in the dynamic nature of wireless channels, where factors such as interference, mobility, and environmental conditions fluctuate rapidly. Adaptive MCS and layer selection strategies are essential to address these challenges. By dynamically adjusting the MCS and the number of layers in response to real-time channel conditions, adaptive techniques can maximize spectral efficiency and ensure robust communication. In MIMO systems, where the complexity of channel interactions is high, adaptive strategies are particularly valuable as they enable the network to exploit the full potential of the available spatial resources.

Heuristic-based layer number and MCS selection techniques [7, 23] rely on Channel Quality Indicator (CQI) and Rank Indicator (RI) feedback from the UE to the base station. The base station uses a

look-up table to map CQI to MCS and RI to layer number, adapting to channel variations, with some methods incorporating HARQ for finer adjustments. A recent proposal [21] introduced a novel SNR-CQI-MCS mapping table, showing effectiveness in various 5G multi-user scenarios. However, these table-based methods often struggle with limited adaptability to diverse channel conditions. Machine learning (ML) offers a more adaptive approach in wireless communication [2, 3, 5, 6, 11], including MCS and layer selection. Techniques utilizing CQI-RI feedback (CSI-reporting) as inputs for ML models, such as supervised learning (SL)[2, 5, 6] and reinforcement learning (RL)[3, 11], efficiently map channel conditions to MCS-layer choices. An RL-based adaptive CQI and RI estimation for 5G NR [3] introduces an online adaptation algorithm to meet target block error rates (BLER). However, CSI-reporting techniques often suffer from long periodicity and coarse granularity, leading to performance degradation. Recently, a CNN-LSTM-based adaptive modulation and coding (AMC) technique for massive MIMO networks [2] was introduced, utilizing uplink channel matrix to predict MCS in MIMO transmissions by extracting spatial and sequential information through CNN and LSTM models. Nevertheless, using SRS-estimated uplink channel information to predict downlink MCS is unreliable for cell-edge users due to limited power and poor SRS reception. Additionally, like many ML-based layer and MCS prediction methods, this approach focuses on a simplified task with a limited number of prediction choices (e.g., 11 MCS indices). Its effectiveness in scenarios involving both layer number and MCS indices selection remains to be evaluated. In response, we propose DRAGON, a DRL-based joint layer number and MCS adapter for 5G MIMO networks within the O-RAN framework. By utilizing SRS-estimated uplink channel matrices and CQI as auxiliary inputs, DRAGON enhances reliability, even for cell-edge users. The action branching architecture enables the DRL model to manage the high-dimensional joint layer-MCS prediction task. As O-RAN continues to grow, there is also extensive literature on radio resource allocation and management within this framework [8, 10].

The main contributions of the paper are as follows:

- We propose DRAGON, a real-time DRL-based mechanism for joint layer number and MCS prediction.
- The action branching architecture is employed within the Deep Double Q-Network (D3QN) model, enabling it to efficiently manage the high-dimensional task of layer-MCS prediction.
- We conduct an exhaustive evaluation of DRAGON using a realistic dataset, and we compare its performance against state-of-the-art approaches to demonstrate its effectiveness. Furthermore, DRAGON is integrated with a real-world O-RAN-based MIMO platform to validate its practical applicability.

2 Methodology

2.1 Overview of DRAGON

In the conventional framework, UEs utilize the Channel State Information Reference Signal (CSI-RS) to measure CSI feedback, including Channel Quality Indicator (CQI), Precoding Matrix Indicator (PMI), and Rank Indicator (RI). This feedback is then transmitted to the base station via an uplink control/data channel. The base station subsequently maps the received CQI values to specific MCS levels by referencing a predefined lookup table [21]. Additionally, the RI is used to determine the number of independent data streams or layers that can be reliably transmitted and received between the UE and the base station. To improve the accuracy of this mapping, certain approaches also incorporate Hybrid Automatic Repeat Request (HARQ) information (i.e., ACK/NACK) from previous time slots. This inclusion is particularly significant in mobile scenarios, where the sequential dependency of channel conditions plays a critical role in adaptive MCS and layer selection.

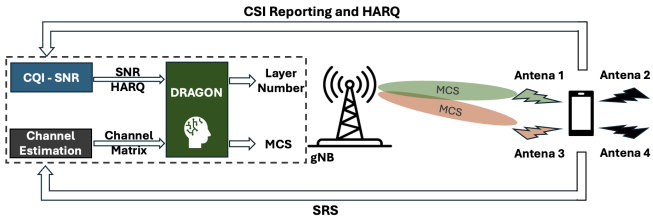


Figure 1: DRAGON framework.

Nevertheless, as indicated in prior studies, CSI reporting typically incurs a delay of up to 9.5 ms and occurs every 8 time slots [1]. In some real-world implementations, CSI reporting may be further delayed, occurring only once every 20 ms. In scenarios where stringent time constraints are present, such feedback-based approaches may struggle to adjust the MCS levels and the number of transmission layers in a timely manner, potentially leading to a degradation in system performance. Moreover, CQI generally quantizes channel quality into 16 levels using 4 bits, resulting in a coarse granularity that complicates precise MCS selection at the base station. Additionally, table-based MCS selection methods may prove ineffective across varying communication scenarios. Therefore, the traditional CQI- and RI-based mechanisms for layer and MCS selection have significant room for improvement.

In contrast, DRAGON utilizes the uplink channel matrix, estimated through channel measurements using the Sounding Reference Signal (SRS), along with the SNR measured periodically by the CQI, to predict downlink layer and MCS selection based on reciprocity in TDD mode. In this approach, the channel matrix provides a more granular and timely response compared to the traditional CQI-RI pair, making it a more effective basis for layer and MCS prediction. The CQI-based SNR serves as an auxiliary factor, mitigating potential inaccuracies in SRS-based SNR due to power limitations on the UE side, particularly for UEs located at the cell edge. Additionally, ACK/NACK feedback from the previous TTI is incorporated to further refine predictions, with throughput serving as the performance metric for evaluation. An illustration of DRAGON is provided in Fig. 1. As illustrated, this problem falls within the domain of discrete-action reinforcement learning, where a Markov Decision Process can be used to model the decision-making process. Discrete-action algorithms have been instrumental in many recent advancements in deep reinforcement learning [12, 22, 24]. However, applying these algorithms to high-dimensional action tasks presents challenges, as the combinatorial increase in the number of possible actions with each additional action dimension leads to difficulties in convergence—a phenomenon known as the "curse

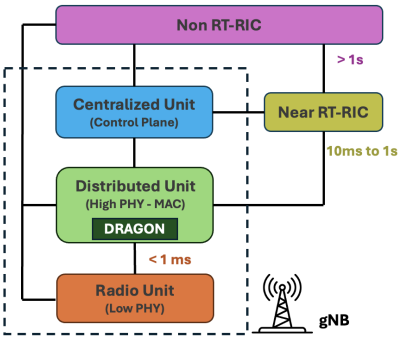


Figure 2: DRAGON in O-RAN framework.

of dimensionality" or "action dimensional disaster" [4]. Given that joint prediction of layer and MCS is a high-dimensional combinatorial problem with hundreds of potential choices, conventional RL algorithms are not well-suited for direct application. To address this challenge, DRAGON adopts an action branching architecture [20] that features a shared decision module followed by multiple network branches, each corresponding to a different action dimension. This architecture enables a linear increase in the number of outputs relative to the number of degrees of freedom, allowing a degree of independence for each action dimension and effectively managing the complexity of the problem.

In O-RAN framework, the key elements include the Radio Unit (RU), Distributed Unit (DU), Centralized Unit (CU), and the RAN Intelligent Controller (RIC). The RU is the front-end of the Open RAN architecture, responsible for the transmission and reception of radio signals over the air interface. The DU handles the real-time and lower-layer tasks, including scheduling (e.g. UE scheduling and layer and MCS selection), HARQ (Hybrid Automatic Repeat Request), and beamforming [15]. Positioned closer to the edge of the network, the DU is crucial for maintaining low-latency communication, making it a vital element in scenarios requiring real-time responsiveness. The CU is responsible for higher-layer protocol processing, such as non-real-time functions and control plane management. Finally, the RAN Intelligent Controller (RIC) adds an additional layer of intelligence to the Open RAN architecture by providing a platform for deploying advanced control and optimization algorithms. DRAGON naturally resides in the DU performing layer and MCS selection like shown in Fig. 2. Because DRAGON leverages a single DRL model to perform one-shot predictions of both the optimal MIMO layer configuration and the appropriate MCS, enabling real-time adaptation to fluctuating channel conditions. This capability is particularly advantageous for deployment in the DU, as it allows DRAGON to seamlessly integrate with existing DU operations without disrupting the functionality of other O-RAN components.

2.2 Action Branching Architecture

As highlighted in §2.1, joint layer and MCS selection is a high-dimensional combinatorial optimization problem, making it challenging for standard discrete-action DRL models to manage due to the curse of dimensionality. The action branching architecture mitigates this issue by splitting the large action space into multiple branches, while retaining a shared decision-making module to reduce the action space size in each branch. Additionally, we employ Dueling Double Deep Q-Network (D3QN) as the foundational algorithm due to its simplicity and effectiveness as a powerful off-policy approach [20].

D3QN. The Deep Q-Network (DQN)[12] represents a fundamental RL model that integrates Q-learning with deep neural networks to approximate the action-value function, enabling agents to make informed decisions based on state spaces. However, both tabular Q-learning and DQN have been shown to suffer from overestimation of action values[22]. This overestimation arises from the fact that the same network is utilized for both action selection and evaluation, leading to overoptimism in the Q-value estimations. To mitigate this issue, [22] introduced the Double DQN (DDQN) algorithm, which employs the policy network for action selection while using the target network for evaluation. Furthermore, the dueling network architecture [24] explicitly separates the representation of state values and state-dependent action advantages into two distinct branches, while sharing a common feature-learning module. These branches are subsequently combined through a specialized aggregation layer to produce an estimate of the action-value function. The dueling network architecture has been shown to enhance policy evaluation, particularly in scenarios involving many similar-valued or redundant actions, thereby enabling faster generalization across large action spaces.

Prioritized Experience Replay Buffer (PERB). The experience replay mechanism allows off-policy reinforcement learning agents to reuse past experiences or demonstrations, thereby enhancing learning efficiency [12]. In the standard DQN algorithm, experience transitions are uniformly sampled from a replay buffer. To further optimize learning, a framework for prioritizing experience was proposed by [19] to replay important transitions—those with a high expected learning progress—more frequently. In our work, we incorporate PERB to enhance neural network performance. We specifically assign higher weights to experience transitions that exhibit higher loss during the learning process and store them in the replay buffer with <state, action, reward, next state> tuples. As a result, these transitions are more likely to be sampled in subsequent iterations.

Action Branching D3QN. Fig. 3 illustrates the architecture of the Action Branching D3QN, which distributes the representation of the value function or policy across multiple network branches while maintaining a shared decision module that encodes a latent representation of the common input state. When a state is input, the shared decision module computes a latent representation, which is subsequently used for evaluating the state value and the factorized state-dependent action advantages in the independent branches. These outputs are then combined via a specialized aggregation layer to produce the Q-values for each action dimension. The factorized Q-values are subsequently queried to generate a joint-action tuple. In the DRAGON model, the input representation is distributed across three branches—one dedicated to layer prediction and two for MCS prediction—reflecting the greater number of potential choices in MCS selection compared to layer selection.

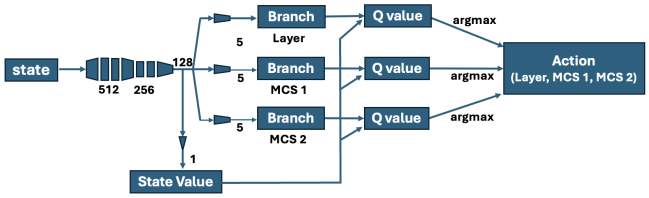


Figure 3: Network architecture of DRAGON. (The number of units of each layer is indicated)

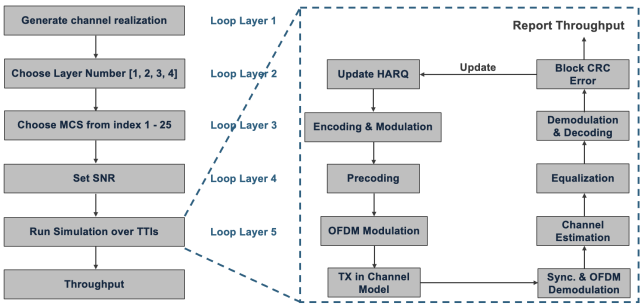


Figure 4: Dataset collection flow.

2.3 Markov Decision Process (MDP) Modeling

We adapt the Action Branching D3QN to formulate and construct a MDP model specifically designed to predict the layer number and MCS in downlink single-user MIMO networks.

State space. We define $s_t := [C_t^{SNR}, HARQ_{t-1}]$ as the state space at TTI t, where C_t^{SNR} indicates flattened channel matrix at TTI t factorized by linear SNR and $HARQ_{t-1}$ indicates the HARQ at TTI $t-1$. Due to the inherent limitations of neural networks in processing complex numbers, we concatenate the real and imaginary components of the channel matrix entries. Since our focus is on wideband MCS selection [1], we do not account for frequency-selective fading and instead utilize an averaged channel matrix across all subcarriers.

Action space. Due to DRAGON's three action branches, the output of the neural network is a vector containing three entries. The first entry corresponds to layer number selection, while the remaining two entries pertain to MCS prediction. To maintain a consistent action bin size across all branches [20], the number of action bins is set to five, with each vector entry being an integer ranging from 0 to 4. Consequently, DRAGON supports the selection of up to four layer numbers (with the fifth bin reserved as a dummy) and 25 MCS levels, effectively covering nearly all options specified in the 5G standard [1]. We define the action space at TTI t as $a_t := [\alpha_t, \beta_t, \gamma_t]$. Layer selection at TTI t: $l_t = \alpha_t$ if $\alpha_t < 4$ else $\alpha_t - 1$ and MCS selection at TTI t: $mcs_t = \beta_t \times 5 + \gamma_t$.

Reward. Our primary objective in layer and MCS selection is to maximize system throughput. To achieve this, we employ the throughput achieved by a single user as the reward. Specifically, if a NACK is received, retransmission occurs in the subsequent TTI, resulting in a throughput of zero at the current moment.

3 Performance Evaluation

In this section, we evaluate DRAGON through a comprehensive assessment. To obtain ground truth data, we perform an exhaustive search over all possible configuration combinations, generating a dataset using the Matlab 5G toolbox [9]. This dataset is subsequently used to train our DRL model and optimal solution can be acquired from it to serve as an upper bound during performance evaluation. Additionally, we implement a heuristic-based approach and a ML-based SOTA method as baselines for comparison with DRAGON. This evaluation underscores our model's efficacy in selecting the optimal MCS and layer configuration.

3.1 Experiment Setup

To generate the dataset, we configured the MATLAB 5G NR Physical Data Shared Channel (PDSCH) Throughput library to simulate a single-cell, 32 (base station antennas) × 4 (UE antennas) single-user MIMO channel. The simulation was set with a 3.5 GHz carrier frequency, 20 MHz bandwidth, and 30 kHz subcarrier spacing, resulting in a total of 51 physical resource blocks (PRBs). To emulate future deployment scenarios, we generated channels under the clustered delay line (CDL)-C model, representative of an Urban Macro Non-Line-of-Sight (NLoS) environment. We also modeled a mobile UE with pedestrian speed, introducing varying channel conditions across different frames. Furthermore, the number of layers and MCS were configurable within the library, with HARQ and retransmission processes enabled. In total, we generated channel matrices for 30K transmission frames for training and testing. We employed MCS Table 2 for PDSCH from 3GPP TS 38.214 [1], encompassing modulation schemes such as 4-QAM, 16-QAM, 64-QAM, and 256-QAM, with LDPC code rates ranging from 0.11 to 0.92. Our study focused on MCS indices 1 to 25, aligning with the two MCS branches detailed in §2.2. MCS 0, 26, and 27 were excluded from our model due to their infrequent selection in real-world applications, attributable to their intolerable bit error rates or inadequate spectral efficiency. System throughput was adopted as the performance metric, with datasets collected for per-frame throughput corresponding to each MCS-layer combination. Specifically, we conducted multiple evaluation rounds per channel, testing all available MCS indices and layer numbers under varying SNRs. The optimal MCS and layer for a given channel scenario and SNR were defined as those achieving the highest throughput. Fig. 4 illustrates the dataset collection process, which includes five nested loops: channel realization, layer selection, MCS selection, SNR setting, and throughput collection. This exhaustive search over all MCS-layer combinations allowed us to determine the throughput of any selection under specific channel conditions and SNR, providing comprehensive guidance for DRL model training and establishing a performance upper bound during evaluation.

The dataset is divided into training and test sets with an 8:2 ratio for training the ML model presented in §2.2. Model training is conducted on an NVIDIA Tesla T4 server [13]. As depicted in Fig.3, the architecture consists of 7 hidden layers in the shared representation module, 3 hidden layers for each of the 3 branches, and one for the state value. ReLU is employed as the activation

Table 1: Experiment Setting and Training Hyper-parameters

Parameter	Value
System Carrier Frequency	3.5 GHz
System Bandwidth	20 MHz
Frame Duration	1 ms
Channel Model	CDL-C
MCS Table	38.214 - Table 5.1.3.1 [1]
Number of BS Antennas	32
Number of UE Antennas	4
Batch Size	128
Learning Rate	1e-4
Optimizer	SGD
Episodes	700

function following each hidden layer. The model is trained using the SGD optimizer [18] in PyTorch [14]. A summary of the relevant simulation parameters is provided in Table 1.

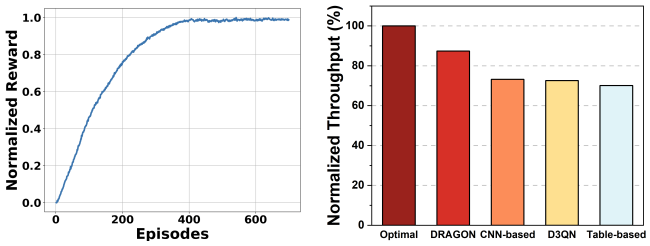
3.2 Benchmarks

To rigorously assess the performance of DRAGON, we compare it against a SOTA CNN-based approach [2] and a heuristic table-based method, which serve as baselines. Additionally, the performance upper bound is derived directly from the collected dataset.

CNN-based Method: Convolutional Neural Networks (CNNs) are widely utilized in classification problems due to their ability to automatically learn complicated feature representations from raw data, making them particularly effective in tasks involving image and signal processing. A state-of-the-art approach has applied a CNN-LSTM architecture for MCS selection, demonstrating significant performance improvements. Using the open-source code provided by the authors [17], we extend the model to include both layer number selection and MCS selection, where the best layer and MCS indices are used as labels for training. To maintain the integrity of the original approach, we preserve the neural network size as specified in the reference work.

Table-based Method: The table-based method is a traditional technique employed in the 5G standard, yet its performance is notably influenced by factors such as 5G numerology, the number of user spatial streams, propagation conditions, and traffic characteristics. In [21], a novel SNR-CQI-MCS mapping table is introduced and evaluated in the context of 5G multi-user scenarios that include audio, video, and gaming traffic patterns, demonstrating a performance improvement of approximately 35% compared to state-of-the-art mapping tables. To make it comparable with DRAGON, we incorporated RI-based layer selection into the method.

D3QN-based Method: This benchmark was conducted as an ablation study in which we removed the action branching from proposed design, relying solely on the D3QN model. As outlined in §2.1, the action branching architecture is central to our design, enabling the discrete-action DRL model to effectively manage the challenges posed by high-dimensional action spaces. Disabling the action branching compromises model performance in the layer-MCS prediction task with high-dimensional action requirements.



(a) (b)

Figure 5: (a) Reward progression during training and (b) Performance Comparison between DRAGON and benchmarks.

Optimal Method: The last benchmark we designed is to achieve an optimal solution by performing an exhaustive search over all possible MCS-layer combinations during the dataset collection phase. By thoroughly exploring the entire solution space, the method ensures that the most effective MCS-layer pair is identified for each channel condition, thereby guaranteeing optimal performance. When applied during evaluation, this method provides a reliable performance upper bound, offering a reference point against which other approaches can be compared.

3.3 Experiment Results

3.3.1 Model Training and Convergence. We constructed a comprehensive dataset comprising 30,000 frames, partitioned into 24,000 samples for training and 6,000 for testing. The DRAGON model was trained for 700 epochs, with 24,000 iterations per epoch. Convergence of the DRL model typically occurred around the 400th epoch shown in Fig. 5a. Throughout training, we employed the epsilon-greedy algorithm to balance exploration and exploitation by either selecting random actions or using learned actions that maximize reward. Epsilon, which represents the probability of choosing random actions, was initially set to 1 and gradually decreased to zero over 200 epochs.

3.3.2 Performance Results. We evaluate DRAGON's performance on a test dataset in comparison to other benchmark models. For CNN-based and D3QN-based approaches, we utilize the test dataset for inference and derive the corresponding throughput based on the predicted layer and MCS indices. In the table-based method, we map the recorded CQI to MCS indices and the RI to the layer number to estimate throughput. The comparison results of normalized throughput are presented in Fig. 5b. DRAGON achieves 87.4% of the best throughput, outperforming other benchmarks due to its ability to effectively manage high-dimensional combinatorial challenges, thanks to the action branching architecture. While the CNN-based method demonstrated near-optimal performance in[2], it was limited to a simpler prediction task with 11 MCS indices. However, as the classification task expands exponentially to include layer selection and all MCS indices, a significant performance degradation is observed. Moreover, the table-based method shows the poorest performance among the benchmarks, primarily due to its limited adaptability to varying channel conditions. Quantitatively, the action branching architecture delivers an impressive 15% throughput improvement over the pure D3QN model.

4 Integration with O-RAN Framework

O-RAN is a modular and open architecture that consists of key components such as the CU, DU, RU, and RIC to enable flexible, software-defined control and optimization of network resources. As illustrated in §2.1, the model is intended to reside in the DU of O-RAN and interact with other essential modules such as the UE scheduler and beamforming components. However, the real-world O-RAN framework-based simulator is implemented in C++, while our trained model is in PyTorch. To bridge this gap, we developed an inference module in C++ and integrated it with the real-world O-RAN-based simulator. Additionally, we employed TorchScript [16] to convert the trained model into a C++-compatible format that can be loaded by the simulator. After rigorous testing with the same test dataset, consisting of approximately 6,000-frame channel realizations, the integrated model demonstrated performance comparable to the results presented in §3. Notably, the model achieved a one-shot prediction time of 0.85 ms for layer and MCS selection, satisfying the stringent latency requirements of 5G systems, which mandate a sub-1ms latency.

5 Discussion

The DRAGON framework, initially developed for single-user MIMO networks within the O-RAN architecture, can be seamlessly extended to multi-user MIMO (MU-MIMO) scenarios, where inter-user interference plays a crucial role. By replacing the raw channel matrix with channel gain information and incorporating inter-user correlations into the state space, DRAGON remains applicable without modifying the action space or reward function. This extension leverages DRAGON's capacity to handle high-dimensional tasks, ensuring adaptability to more complex network scenarios, including power allocation. This flexibility enhances DRAGON's relevance to practical 5G and beyond networks in the O-RAN framework.

6 Conclusion

In this paper, we introduce DRAGON, a DRL-based approach for selecting the optimal layer number and MCS in single-user MIMO networks within the O-RAN framework. Utilizing only the uplink channel matrix, SNR, and HARQ feedback, our method predicts the best downlink configuration. We collected a comprehensive dataset through realistic simulations and validated our model on a real-world O-RAN MIMO platform. Our results show that DRAGON outperforms state-of-the-art ML and heuristic methods.

Acknowledgments

The majority of this work is completed during Qing's internship in Mavenir. We thank Mavenir for providing the experiment platforms and valuable comments during the course of work. This work is supported in part by the NTIA Public Wireless Supply Chain Innovation Fund and National Science Foundation Grant CNS-2016727.

References

[1] 3GPP. 2018. *TSG RAN; NR; Physical Layer Procedures For Data.* Technical Specification (TS) 38.214 Version 15.2.0. 3rd Generation Partnership Project (3GPP).

[2] Qing An, Mehdi Zafari, Chris Dick, Santiago Segarra, Ashutosh Sabharwal, and Rahman Doost-Mohammady. 2023. ML-Based Feedback-Free Adaptive MCS Selection for Massive Multi-User MIMO. In *2023 57th Asilomar Conference on Signals, Systems, and Computers*. 157–161. https://doi.org/10.1109/IEEECONF59524.2023.10476866

[3] Abdulrahman Baknina and HyukJoon Kwon. 2020. Adaptive CQI and RI Estimation for 5G NR: A Shallow Reinforcement Learning Approach. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*. 1–6. https://doi.org/10.1109/GLOBECOM42002.2020.9348049

[4] Liang Chen, Fanglei Sun, Kai Li, Ruiqing Chen, Yang Yang, and Jun Wang. 2021. Deep Reinforcement Learning for Resource Allocation in Massive MIMO. In *2021 29th European Signal Processing Conference (EUSIPCO)*. 1611–1615. https://doi.org/10.23919/EUSIPCO54536.2021.9616054

[5] Robert C. Daniels, Constantine M. Caramanis, and Robert W. Heath. 2010. Adaptation in Convolutionally Coded MIMO-OFDM Wireless Systems Through Supervised Learning and SNR Ordering. *IEEE Transactions on Vehicular Technology* 59, 1 (2010), 114–126. https://doi.org/10.1109/TVT.2009.2029693

[6] Zhijie Dong, Junchao Shi, Wenjin Wang, and Xiqi Gao. 2018. Machine Learning Based Link Adaptation Method for MIMO System. In *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. 1226–1231. https://doi.org/10.1109/PIMRC.2018.8580924

[7] A. Durán, M. Toril, F. Ruiz, and A. Mendo. 2015. Self-Optimization Algorithm for Outer Loop Link Adaptation in LTE. *IEEE Communications Letters* 19, 11 (2015), 2005–2008. https://doi.org/10.1109/LCOMM.2015.2477084

[8] Claudio Fiandrino, Leonardo Bonati, Salvatore D'Oro, Michele Polese, Tommaso Melodia, and Joerg Widmer. 2023. EXPLORA: AI/ML EXPLainability for the Open RAN. *Proc. ACM Netw.* 1, CoNEXT3, Article 19 (nov 2023), 26 pages. https://doi.org/10.1145/3629141

[9] The MathWorks Inc. 2022. *MATLAB version: 9.13.0 (R2022b).* Natick, Massachusetts, United States. https://www.mathworks.com

[10] Woo-Hyun Ko, Ushasi Ghosh, Ujwal Dinesha, Raini Wu, Srinivas Shakkottai, and Dinesh Bharadia. 2023. Demo: EdgeRIC: Delivering Realtime RAN Intelligence. In *Proceedings of the ACM SIGCOMM 2023 Conference* (New York, NY, USA) *(ACM SIGCOMM '23)*. Association for Computing Machinery, New York, NY, USA, 1162–1164. https://doi.org/10.1145/3603269.3610867

[11] Joao P. Leite, Paulo Henrique P. de Carvalho, and Robson D. Vieira. 2012. A flexible framework based on reinforcement learning for adaptive modulation and coding in OFDM wireless systems. In *2012 IEEE Wireless Communications and Networking Conference (WCNC)*. 809–814. https://doi.org/10.1109/WCNC.2012.6214482

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[13] NVIDIA Corporation. 2018. NVIDIA Tesla T4 GPU. https://www.nvidia.com/en-us/data-center/tesla-t4/. Available at https://www.nvidia.com/en-us/data-center/tesla-t4/.

[14] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. arXiv:1912.01703 [cs.LG]

[15] Michele Polese, Leonardo Bonati, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. 2023. Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges. *Commun. Surveys Tuts.* 25, 2 (apr 2023), 1376–1411. https://doi.org/10.1109/COMST.2023.3239220

[16] PyTorch Contributors. 2020. TorchScript. https://pytorch.org/docs/stable/jit.html. Accessed: 2024-08-29.

[17] GitHub repository. 2024. ML-Based Feedback-Free Adaptive MCS Selection for Massive Multi-User MIMO. https://github.com/QingRice/ML-Based-Feedback-Free-Adaptive-MCS-Selection-for-Massive-Multi-User-MIMO

[18] Sebastian Ruder. 2017. An overview of gradient descent optimization algorithms. arXiv:1609.04747 [cs.LG]

[19] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized Experience Replay. arXiv:1511.05952 [cs.LG] https://arxiv.org/abs/1511.05952

[20] Arash Tavakoli, Fabio Pardo, and Petar Kormushev. 2019. Action Branching Architectures for Deep Reinforcement Learning. arXiv:1711.08946 [cs.LG] https://arxiv.org/abs/1711.08946

[21] Abitha K Thyagarajan, Priyeash Balasubramanian, Vydeki D, and Karthik M. 2021. SNR-CQI Mapping for 5G Downlink Network. In *2021 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob)*. 173–177. https://doi.org/10.1109/APWiMob51111.2021.9435258

[22] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double Q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.

[23] Sander Wahls and H. Vincent Poor. 2013. Link adaptation for BICM-OFDM through adaptive kernel regression. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 5136–5140. https://doi.org/10.1109/ICASSP.2013.6638641

[24] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. 2016. Dueling Network Architectures for Deep Reinforcement Learning. arXiv:1511.06581 [cs.LG] https://arxiv.org/abs/1511.06581