

TRACTOR: Traffic Analysis and Classification Tool for Open RAN

Joshua Groen, Mauro Belgiovine, Utku Demir, Brian Kim, Kaushik Chowdhury
Institute for the Wireless Internet of Things, Northeastern University, Boston, MA, USA
{groen.j, belgiovine.m, u.demir, br.kim, k.chowdhury}@northeastern.edu

Abstract—5G and beyond cellular networks promise remarkable advancements in bandwidth, latency, and connectivity. The emergence of Open Radio Access Network (O-RAN) represents a pivotal direction for the evolution of cellular networks, inherently supporting machine learning (ML) for network operation control. Within this framework, RAN Intelligence Controllers (RICs) from one provider can employ ML models developed by third-party vendors through the acquisition of key performance indicators (KPIs) from geographically distant base stations or user equipment (UE). Yet, the development of ML models hinges on the availability of realistic and robust datasets. In this study, we embark on a two-fold journey. First, we collect a comprehensive 5G dataset, harnessing real-world cell phones across diverse applications, locations, and mobility scenarios. Next, we replicate this traffic within a full-stack srsRAN-based O-RAN framework on Colosseum, the world’s largest radio frequency (RF) emulator. This process yields a robust and O-RAN compliant KPI dataset mirroring real-world conditions. We illustrate how such a dataset can fuel the training of ML models and facilitate the deployment of xApps for traffic slice classification by introducing a CNN based classifier that achieves accuracy $> 95\%$ offline and 92% online. To accelerate research in this domain, we provide open-source access to our toolchain and supplementary utilities, empowering the broader research community to expedite the creation of realistic and O-RAN compliant datasets.

Index Terms—O-RAN, 5G, Data set generation, Traffic Classification, Network Slicing

I. INTRODUCTION

5G and beyond networks hold the promise of significantly amplified throughput, reduced latency, and a vast increase in connection capacity. However, the pursuit of these objectives often entails competing demands that render a fixed-configuration system incapable of concurrently fulfilling all performance criteria. To address this challenge, 5G introduces the concept of network slicing, comprising three key slices: enhanced Mobile Broadband (eMBB), Ultra Reliable Low Latency Communications (URLLC), and massive Machine Type Communications (mMTC). Network slicing facilitates end-to-end resource allocation tailored to the specific requirements of distinct traffic types. This process involves partitioning a physical network into multiple virtual networks, capitalizing on the growing virtualization trend in 5G networks [1]. Each slice is designed with a distinct purpose: eMBB prioritizes maximum bandwidth, URLLC ensures high reliability and minimal latency, while mMTC excels in accommodating a multitude of concurrent connections.

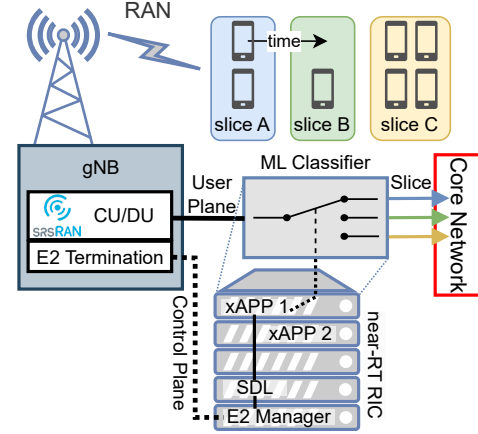


Fig. 1: O-RAN system used in TRACTOR

The Open Radio Access Network (O-RAN) framework supports virtualization as one of its four key architectural principles [2]. Additionally, O-RAN provides inherent support for closed-loop control driven by machine learning (ML), a capability harnessed through RAN Intelligent Controllers (RICs) for the optimization of network performance. O-RAN introduces standardized technical specifications for open interfaces, enabling the collection of Key Performance Indicators (KPIs) to facilitate diverse control and automation actions. In this context, O-RAN stands out as a robust platform for the implementation of network slicing.

Traffic Classification Challenges: While significant progress has been made in network slicing (Sec. II-B), the focus has largely been on improving the RAN component (Fig. 1). However, measurements from deployed 5G networks reveal that the primary throughput and delay bottleneck lies not within the RAN but in the second hop of the network [3], [4]—within the LTE Enhanced Packet Core (EPC) or 5G Core Network (CN) highlighted in Fig. 1. Meeting diverse network slice requirements necessitates end-to-end network path optimization and accurate traffic classification [5].

Developing O-RAN compliant traffic slice classification poses several challenges. Neither the traditional 5-tuple flow information (source and destination IP address, source and destination port, and transport layer protocol) nor the complete layer 2 frame are available to the near-real time (near-RT) RIC in an O-RAN system. Further, the KPIs used in the state-of-

the-art 5G classifiers [6]–[8] are neither O-RAN compliant nor based on real 5G traffic.

To tackle these challenges, we introduce **TRACTOR: Traffic Analysis and Classification Tool for Open RAN**. This toolchain leverages O-RAN’s open interfaces for gathering KPIs from the gNB, which serve as inputs to near-RT RICs equipped with ML classifiers for user traffic classification. We intentionally design our entire system to ensure user privacy and system security while achieving accurate network-initiated classification. TRACTOR represents a comprehensive, O-RAN compliant deployment within Colosseum, the world’s largest wireless network emulator featuring hardware-in-the-loop capabilities [9].

Our main contributions are:

- The creation of a pioneering and publicly available dataset of O-RAN compliant KPIs generated from real 5G user traffic, accompanied by all necessary tools for traffic emulation in Colosseum or similar environments.
- The development of a toolset that empowers fellow researchers to efficiently generate datasets of their choice from O-RAN compliant full-stack emulations.
- An evaluation of an ML based traffic slice classifier xApp, both offline and online. This demonstrates the model’s feasibility when deployed in actual systems.
- We release our xApp and all associated tools for public use and to enable replication and extension of our results.

The rest of this paper is structured as follows. We discuss the background and related works in Sec. II. We describe our TRACTOR tool chain in detail in Sec. III. Next, we briefly describe the ML model we deploy as an xApp in Sec. IV and highlight initial offline and online results in Sec. V. We end with a brief discussion and conclusion in Sec. VI.

II. BACKGROUND AND RELATED WORKS

A. O-RAN Principles and Framework

The near-RT RIC seen in Fig. 1 is a core component of the ML-driven control and optimization of the RAN necessary for 5G. The near-RT RIC hosts xApps, microservices that support custom logic to perform RAN management. Open interfaces are another core principle in O-RAN and one of the key interfaces is the E2 interface that connects the gNB and the near-RT RIC. An xApp can receive KPIs from the gNB over the E2 interface, use those KPIs in a pre-trained ML model, and send back control actions.

B. O-RAN Traffic Classification

Traditional IP traffic classification relied on packet inspection, as seen in [10] and [11]. However, these methods fail when packets are encrypted at the network or data-link layer. To overcome this, statistical traffic properties were used to identify applications, which naturally led to the use of ML in traffic classification, including encrypted traffic. Generally, encrypted traffic classification techniques either use traffic flows defined by a 5-tuple (source IP, source port, destination IP, destination port, and transport-level protocol) [12] or the entire encrypted packet [13]–[15] as the input. Both of these

options present a high risk to privacy and security, especially when paired with other information unique to a cellular environment, such as physical location. Furthermore, in an O-RAN system, the near-RT RIC does not have access to the entire packet, encrypted or otherwise.

Bonati *et al.* [16] select the best performing scheduling policy and resource block assignment in each network slice using deep reinforcement learning fed with data generated in the Colosseum emulator. Both UE assignment and the rewards in the DRL agents are based on knowing the traffic slice *a priori*. Weerasinghe *et al.* [17] aims to predict incoming traffic at the LTE base station (eNB) using supervised ML. The authors test the performance of their model using simulated bursty LTE traffic data. Li *et al.* [18] predicts traffic type (among IM, web browsing, and video data) in an upcoming 5 minute period using the previous 3 hours of traffic data in a framework that consists of α -stable models, dictionary learning, and alternating direction method (ADM) using 7 million users’ OTA 2G-4G application layer data. Johnson *et al.* [19] perform network slicing and scheduling on an xApp that is based on srsRAN using policy driven heuristic methods.

Novelty of Proposed Approach over State-of-the-art: Our approach to slice allocation differs from user equipment (UE) initiated slicing, which often involves manual or negotiated slice assignments, adding complexity and potential overhead [20], [21]. Instead, we focus on network-initiated slicing, allowing the gNB to dynamically allocate traffic flows to the appropriate slice. This approach enhances resource allocation without the need for extensive user involvement. While prior works [6]–[8] explore similar concepts, none of these utilize O-RAN compliant KPIs generated from real 5G traffic.

One of the primary shortfalls of previous ML-based slicing is that the KPIs utilized could identify specific users or user traffic, violating user privacy. In contrast, TRACTOR does not use any identifying KPIs that could be correlated to a specific UE, source, or destination. This ensures user privacy and reduces the threat surface area for malicious attacks. Further, neither the traditional 5-tuple flow information, the entire layer 2 frame, nor the KPIs used in [6]–[8] are available to the near-RT RIC in an O-RAN system. In contrast to these works, TRACTOR is the first work that uses real 5G traces to generate O-RAN compliant KPIs used to perform near-RT traffic slice classification using trained ML models.

III. CREATING AN O-RAN DATASET

To understand real 5G traffic patterns and overcome one of the challenges found in prior work, we create a new and publicly available tool chain and dataset. We implement a software-defined RAN using the srsRAN-based SCOPE framework [22] for both the gNB and one or more UEs, as depicted in Fig. 1. SCOPE extends srsLTE (now srsRAN) version 20.04 by introducing features like an E2 interface and open APIs for real-time gNB reconfiguration, along with additional data collection tools. Our near-RT RIC, part of the CoO-RAN framework [23], utilizes a minimal near-RT RIC

based on the O-RAN Software Community’s near-RT RIC (Bronze release). This near-RT RIC is structured as Docker containers within the CoIO-RAN LXC, providing essential functionalities such as E2 interface support for data collection and control communication with RAN nodes, alongside a sample xApp for collecting fundamental KPIs from the gNB.

We add several key elements to this framework by creating a custom traffic generator and developing the TRACTOR xApp. We also create several supporting utilities as part of the TRACTOR code base. We make this code publicly available to build a framework useful for other researchers interested in traffic classification and slicing.

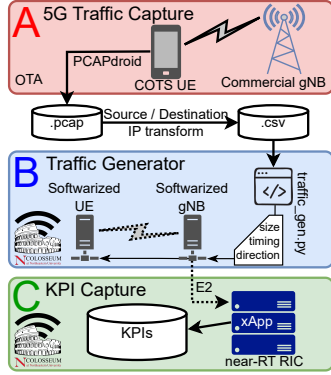


Fig. 2: Process to create an O-RAN compliant dataset.

A. Collecting Real-World 5G User Data

To collect real-world 5G user traffic, we use the open source PCAPdroid [24] Android application on a Google Pixel 6 Pro smartphone and generate packet captures (.pcap files) of user traffic. This is illustrated by block A in Fig. 2. We use a variety of applications for each network slice. For eMBB, we stream videos, browse the Internet, and transfer large files. For URLLC, we conduct both voice phone calls, video chat, and utilize real time AR applications. For mMTC we capture texts and background traffic from all apps when the phone is not actively being used. This is not the typical example of mMTC traffic, such as IoT applications. However, it does fit nicely in the fundamental definition of mMTC because it is low throughput, latency tolerant communication from numerous applications. PCAPdroid provides a custom trailer that adds metadata identifying the phone application to each packet capture. This data is used to ensure captured data is labeled with the correct network slice. This large dataset was collected on multiple different days, in different locations, with different levels of mobility. Table I gives a detailed overview of the parameters used to capture 447 minutes of 5G traffic.

B. Replaying Traffic in Colosseum

After capturing real-world data, we need a method to replay traffic in an O-RAN system to capture KPIs. To enable this, we use Wireshark to export the packet dissection to a .csv file. Next, we alter the source and destination IP addresses so that IP traffic is between two endpoints. This is especially

necessary for the mMTC traffic where, in reality, the UE is reaching out to a wide range of IPs. These transformation processes are shown between blocks A and B in Fig. 2. While we chose to use real traces from an UE using deployed 5G networks, any .pcap capture can be used as the input to the system. In this way, we enable researchers to use any network packet capture as the input to understand the impact to O-RAN KPIs.

We built a traffic generator tool to replay the traffic between the UE and gNB as shown in block B in Fig. 2. The traffic generator emulates the original traffic by reading the length field of the .csv file and sending a random byte string of the appropriate length at the time indicated by the packet timestamp. We use User Datagram Protocol (UDP) in our traffic emulation because any transport layer functions are already captured in the original traffic. In order to ensure the replay traffic is the same as the original, we subtract 70 bytes of overhead to remove the PCAPdroid trailer and account for the new Ethernet frame, IP header, and UDP header we use. We use the source and destination IP to determine if the UE or gNB should be sending the traffic. We utilize the time stamp to wait the appropriate amount of time before sending the next packet. In this way, the UE and gNB accurately emulate the first hop (cellular RAN) of the original 5G traffic.

We have developed several supplementary utilities related to traffic generation, including multiple UE traffic generation, anomalous traffic generation, and the introduction of arbitrary RF interference. The multiple UE traffic generator operates by replaying a randomly selected trace for each UE participating in the experiment. While our original traces already encompass real network competition, this approach enables researchers to delve deeper into the influence of multiple competing UEs on

Slice	Application	Location	Mobility	Time (min)
eMBB	Chrome, YouTube, One Drive	Residential	Stationary	43.5
eMBB	YouTube	Campus	Stationary	29.0
eMBB	YouTube	Campus	Stationary	17.2
eMBB	Netflix	Campus	Stationary	21.3
eMBB	One Drive	Campus	Stationary	30.6
eMBB	YouTube	Campus	Stationary	4.9
eMBB	Pandora	Campus	Stationary	6.7
eMBB	One Drive	Campus	Stationary	1.1
eMBB	Chrome	Campus	Stationary	5.7
mMTC	background	Mixed	Driving	64.0
mMTC	background	Campus	Walking	11.8
mMTC	background	Campus	Stationary	23.7
mMTC	background	Campus	Stationary	23.9
mMTC	background	Campus	Stationary	16.4
mMTC	background	Campus	Stationary	5.6
mMTC	background	Campus	Stationary	20.9
URLLC	Google Meet	Campus	Walking	57.0
URLLC	Phone, Google Meet	Residential	Walking	5.8
URLLC	Google Meet	Campus	Stationary	8.0
URLLC	Facebook Messenger	Campus	Stationary	21.0
URLLC	Google Meet	Campus	Walking	7.9
URLLC	Google Meet	Campus	Stationary	7.1
URLLC	Google Maps Live View AR	Campus B	Walking	6.5
URLLC	Facebook Messenger	Campus	Stationary	3.9
URLLC	Microsoft Teams	Campus	Stationary	3.5

TABLE I: Detailed break down of real world data capture variables including application used, location, and mobility. For a given traffic slice, each row was collected on a different day.

O-RAN KPIs. The anomalous traffic generation tool offers support for two potential attack scenarios. Firstly, it models a DoS UDP flood attack using a statistical model. Secondly, it facilitates an attack known as “data-hog,” which employs the original traces but increases packet sizes based on a Gaussian distribution. The RF interference tool generates arbitrary waveforms specified by the user on the uplink or downlink channel. This provides a valuable tool for understanding the impact on O-RAN systems as the RF environment becomes ever more congested. These tools serve as starting points for exploring security implications within O-RAN systems.

C. Capturing KPIs

The traffic generation script allows us to replicate the timing, length, and direction of all data sent between the UE and gNB, while completely anonymizing the actual payload within our experimental test bed. Our O-RAN test bed further emulates the channel conditions between the gNB and UE based on measured channel conditions for a real deployed cellular system. This allows us to accurately capture the O-RAN KPIs as if the original communication were taking place in our O-RAN test bed. To the best of our knowledge, this is the first dataset generated with live 5G traffic and accurately replayed in an O-RAN system to capture KPIs.

To capture KPIs, we employ the TRACTOR xApp, which retrieves requested KPIs from the gNB every 250ms over the E2 interface. This xApp uses our ML model for online traffic slice classification. Simultaneously, we record all the available KPIs for offline training. These KPIs are stored in a .csv file and are part of our publicly accessible dataset. The process is depicted in block C of Fig. 2. Additionally, we provide a tool for automated IPsec configuration over this E2 interface, as elaborated in [25], which is the first open-source solution for configuring O-RAN compliant IPsec over the E2 interface, facilitating swift O-RAN system deployment and systematic performance analysis.

D. Pre-processing KPIs

In our O-RAN setup, we have access to 31 KPIs that cover various low-level performance metrics and include some identifiers like IMSI, RNTI, and slice ID. Before inputting these KPIs into our ML model, we conduct preprocessing to remove KPIs that contain unique identifying information and certain administrative data, such as slice assignments and scheduling policies, to ensure user privacy and confidentiality. Additionally, we exclude KPIs like received signal strength indicator (RSSI) that lack values in our Colosseum emulation, reducing input dimensions without loss of information. The resulting dataset for model training consists of 17 carefully selected KPIs, detailed in Table II. A comprehensive list of all KPIs and their descriptions is available in our public dataset.

The second major step is to trim some of the time from the beginning and the end of the KPI capture. While the replay script exactly replicates the original capture, there are periods of time before and after the replay script runs when KPIs are still being captured. Further, within some slices there can

be large periods of no traffic. We had to manually inspect the KPIs and remove these periods of silence for training purposes only.

KPI name	Description
dl_mcs	Downlink modulation and coding
dl_n_samples	Number of download samples in previous 250 ms
dl_buffer_bytes	Downlink queue length in bytes
tx_brat_downlink_Mbps	Downlink bitrate in Mbps
tx_pkts_downlink	Downlink number of packets transmitted in previous 250 ms
dl_cqi	Downlink channel quality indicator
ul_mcs	Uplink modulation and coding
ul_n_samples	Uplink number of samples in previous 250 ms
ul_buffer_bytes	Uplink queue length in bytes
rx_brat_uplink_Mbps	Uplink bitrate in Mbps
rx_pkts_uplink	Uplink number of packets recieved in previous 250 ms
rx_errors_up_perc	Uplink percent of packets with errors in previous 250 ms
ul_sinr	Uplink signal to interference and noise ratio
phr	UE power head room
sum_reqsted_prbs	Sum of the resource blocks requested in previous 250 ms
sum_granted_prbs	Sum of the resource blocks granted in previous 250 ms
ul_turbo_iters	Uplink turbo encoding

TABLE II: TRACTOR uses 17 O-RAN compliant KPIs. None of these KPIs expose uniquely identifiable information.

IV. MACHINE LEARNING FOR TRAFFIC CLASSIFICATION

To demonstrate the the potential of the TRACTOR toolset, we train and deploy an xApp that uses a ML model, shown in Fig 3, to perform traffic slice classification.

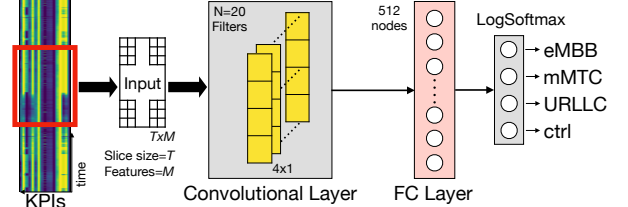


Fig. 3: ML model used for traffic classification in the TRACTOR framework.

A. Data Preprocessing

We form our input by stacking T consecutive KPIs sampled by the gNB, highlighted on the left side of Fig. 3. Each KPI feature set is formed by $M = 17$ traffic metrics listed in Table II. Thus, when stacked, we obtain a $T \times M$ 2D input, representing a snapshot of how such traffic indicators evolve over a $T \times 250$ ms time span. In other words, we use a sliding window with the size of $T \times 250$ ms, making a new prediction at every 250 ms. We collect traffic metrics in Colosseum for the three main traffic categories, i.e. eMBB, URLLC, and mMTC, and include an additional class of samples, denoted as *control* (ctrl), or “silent”, class to identify the portions of traffic where no application data is being exchanged between registered UEs and gNB. Note that depending on the UE’s activities, control class traffic might be found in any of the three traffic categories and therefore we consider *ctrl* as a fourth meta-class that can be used to identify idle users and applications. Adding this traffic slice will benefit users and service providers by providing a separate slice with minimum resources allocated to keep connections alive and preserve resources for the other slice types.

We preprocess input slices in the dataset by normalizing each KPI feature individually, in order to keep all values in the dataset within $[0, 1]$ interval. We then randomize the order of slices and allocate 80% of input samples to train the model, while retaining 20% for testing purposes that the model does not see during training. After training and testing datasets are defined, we obtain 111.6K training samples and 27.9K testing samples, equally distributed among the main 3 traffic classes and control traffic class.

B. Model Training

To train our model, we generate a dataset by slicing the KPIs in groups of T consecutive time samples for each KPI. The proposed model is composed of a single 2D CNN layer with 20 kernels using ReLU activation, followed by 1 fully connected (FC) hidden layer of 512 neurons with ReLU activation function and the LogSoftmax output layer. The kernel size of spatial filters in the convolutional layer is chosen to be 4×1 , since we want local patterns to be learned at each individual feature value as they are not spatially correlated on the KPI features dimension. The extracted pattern is then passed on to the following hidden layer and finally used to classify the incoming KPIs traffic pattern. The optimizer used is Adam and learning rate chosen starts from 10^{-3} and decreases by a factor 10 when a plateau in the cross entropy loss is found, until a minimum value of 10^{-5} is reached. We train our model for a total of up to 350 epochs, though we enable early stopping.

V. PERFORMANCE EVALUATION

In this section, we present the results of our O-RAN traffic classifier. We evaluate our classifiers in both offline and online setting, i.e. deployed as an xApp. We generate all the evaluation data in Colosseum as described in Sec. III in order to validate the feasibility of our model and analysis. Our metric of success is traffic classification accuracy, which we evaluate across multiple temporal input sizes (T).

A. Offline TRACTOR Accuracy Evaluation

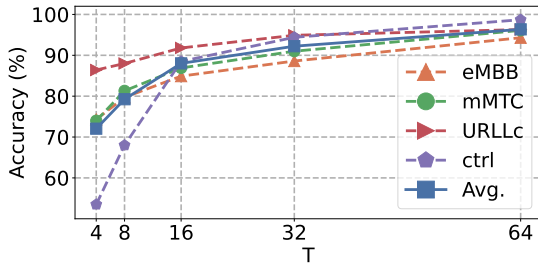


Fig. 4: Average offline CNN-based classifier performance for different input slice sizes.

In order to test the classifier performance and highlight the challenges of our proposed approach in a realistic O-RAN setting, we train different versions of the proposed model for input slice sizes of $T = \{4, 8, 16, 32, 64\}$, which correspond to classification time granularity of $\{1, 2, 4, 8, 16\}$ seconds,

respectively. Fig. 4 depicts the CNN-based classification accuracy over chosen values of T and shows how this parameter influences the classification performance. It is clear that the classifier performance is best when longer periods of traffic metrics are observed by the classifier, reaching an overall accuracy of $\sim 95\%$ for $T = 64$, i.e. 16s worth of sampled KPIs.

B. Online TRACTOR Accuracy Evaluation

We also evaluate the performance of TRACTOR deployed as an xApp in an O-RAN compliant system, shown in Fig. 1, to validate the results from the offline trained model. The xApp is deployed inside the near-RT RIC as a Python script, following the implementation details in Sec. III-C. The xApp constructs the input to the classifiers by stacking newly received KPIs on the previous $T - 1$, and then queries the classifier model in order to provide the classification output. The classifier output identifies which slice the UE should be allocated to at the classification moment and is reported back to the gNB to perform slice assignment.

In order to test the accuracy of our proposed models in unseen conditions, we collect a new set of traffic pattern transmissions for each class (except *ctrl*, which would be naturally included in all the traces) and replay such patterns on Colosseum between the gNB and UE. This generates a new set of KPIs for testing that are different from the KPIs we used for our training. To fairly evaluate the accuracy of proposed classifiers in recognizing the correct traffic slice in test settings, we process the classifier outputs by defining an Idle Traffic Removal (ITR) heuristic that we apply to the incoming KPIs in order to exclude known idle portion of traffic and re-adjust the total number of classification samples when computing the accuracy. We evaluate how the classifier performs on the filtered inputs, again considering all configurations of slice length $T = \{4, 8, 16, 32, 64\}$ in Fig. 5.

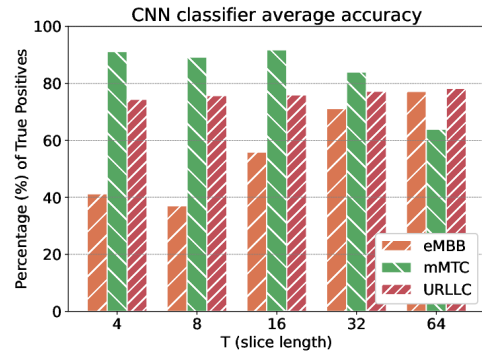


Fig. 5: Average accuracy as percentage (%) of correct classifications for each test in online testing.

Fig. 5 shows that, while longer context provided by larger values of T generally helps the classification accuracy, there is no one-size-fits-all solution. This highlights the challenges posed by the problem of real time traffic type classification in fully O-RAN compliant systems tackled in this work.

VI. DISCUSSION AND CONCLUSION

One of the challenges we identified early on is the vast amount of potential applications and use cases for modern UEs. We chose to use specific exemplar types of traffic, detailed in Sec. III-A, for our initial exploration. While we do not cover all possible use cases in this work, we believe using select cases of real traffic is a better starting point than using statistical based traffic generation. In general, the majority of internet traffic is inherently bursty and unpredictable, as observed in most of our collected traffic.

To meet the complex demands of 5G and beyond cellular networks, we introduce **TRACTOR** (Traffic Analysis and Classification Tool for Open RAN), showcasing the viability of automated traffic classification and slice allocation. We offer public access to our toolset, along with O-RAN compliant KPIs generated from a real-user 5G traffic dataset. Initial ML models validate TRACTOR's capacity for automated, user traffic classification achieving accuracy $> 95\%$ offline and $> 92\%$ online. This work underscores the potential for future networks to excel across multiple performance metrics, catering to individual user requirements while preserving confidentiality and security.

By developing TRACTOR as an entirely open-source and software-based full stack emulation system, we have laid the foundation for further research in several areas. These areas include: the ability to generate O-RAN KPIs from existing or new traffic captures, studying user quality of experience after adding the feedback mechanism to drive traffic slicing, adjusting the KPI reporting frequency to optimize TRACTOR's time scale, studying the impact of multiple competing UEs, and studying the impact of deploying ML models to proactively predict future traffic slices.

ACKNOWLEDGMENT

This article is based upon work partially supported by U.S. National Science Foundation under grant CNS-2112471.

REFERENCES

- [1] S. Zhang, "An Overview of Network Slicing for 5G," *IEEE Wireless Communications*, vol. 26, no. 3, pp. 111–117, 2019.
- [2] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges," *IEEE Communications Surveys & Tutorials*, 2023.
- [3] A. Narayanan, X. Zhang, R. Zhu, A. Hassan, S. Jin, X. Zhu, X. Zhang, D. Rybkin, Z. Yang, Z. M. Mao *et al.*, "A variegated look at 5G in the wild: performance, power, and QoE implications," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021, pp. 610–625.
- [4] D. Xu, A. Zhou, X. Zhang, G. Wang, X. Liu, C. An, Y. Shi, L. Liu, and H. Ma, "Understanding operational 5G: A first measurement study on its coverage, performance and energy consumption," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 479–494.
- [5] Cisco, "QoS: Classification Configuration Guide, Cisco IOS XE 17," Cisco, Tech. Rep., April 2020.
- [6] A. Thantharate, R. Paropkari, V. Walunj, and C. Beard, "Deepslice: A deep learning approach towards an efficient and reliable network slicing in 5G networks," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. IEEE, 2019, pp. 0762–0767.
- [7] R. K. Gupta and R. Misra, "Machine learning-based slice allocation algorithms in 5G networks," in *2019 International Conference on Advances in Computing, Communication and Control (ICAC3)*. IEEE, 2019, pp. 1–4.
- [8] S. Khan, A. Hussain, S. Nazir, F. Khan, A. Oad, and M. D. Alshehri, "Efficient and reliable hybrid deep learning-enabled model for congestion control in 5G/6G networks," *Computer Communications*, vol. 182, pp. 31–40, 2022.
- [9] L. Bonati, P. Johari, M. Polese, S. D'Oro, S. Mohanti, M. Tehrani-Moayyed, D. Villa, S. Shrivastava, C. Tassie, K. Yoder *et al.*, "Colosseum: Large-scale wireless experimentation through hardware-in-the-loop network emulation," in *2021 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*. IEEE, 2021, pp. 105–113.
- [10] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE communications surveys & tutorials*, vol. 10, no. 4, pp. 56–76, 2008.
- [11] Y. Li, B. Liang, and A. Tizghadam, "Robust Online Learning against Malicious Manipulation and Feedback Delay With Application to Network Flow Classification," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 8, pp. 2648–2663, 2021.
- [12] G. Aceto, D. Ciunzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning," in *2018 Network traffic measurement and analysis conference (TMA)*. IEEE, 2018, pp. 1–8.
- [13] M. Lotfollahi, M. Jafari Siavoshani, R. Shirali Hossein Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.
- [14] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," in *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, 2016, pp. 407–414.
- [15] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: An overview," *IEEE communications magazine*, vol. 57, no. 5, pp. 76–81, 2019.
- [16] L. Bonati, S. D'Oro, M. Polese, S. Basagni, and T. Melodia, "Intelligence and learning in O-RAN for data-driven NextG cellular networks," *IEEE Communications Magazine*, vol. 59, no. 10, pp. 21–27, 2021.
- [17] T. N. Weerasinghe, I. A. Balapuwaduge, and F. Y. Li, "Supervised learning based arrival prediction and dynamic preamble allocation for bursty traffic," in *IEEE INFOCOM 2019-IEEE conference on computer communications workshops (INFOCOM WKSHPS)*. IEEE, 2019, pp. 1–6.
- [18] R. Li, Z. Zhao, J. Zheng, C. Mei, Y. Cai, and H. Zhang, "The learning and prediction of application-level traffic data in cellular networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3899–3912, 2017.
- [19] D. Johnson, D. Maas, and J. Van Der Merwe, "NexRAN: Closed-loop RAN slicing in POWDER-A top-to-bottom open-source open-RAN use case," in *Proceedings of the 15th ACM Workshop on Wireless Network Testbeds, Experimental evaluation & Characterization*, 2022, pp. 17–23.
- [20] V. K. Choyi, A. Abdel-Hamid, Y. Shah, S. Ferdi, and A. Brusilovsky, "Network slice selection, assignment and routing within 5G networks," in *2016 IEEE Conference on Standards for Communications and Networking (CSCN)*. IEEE, 2016, pp. 1–7.
- [21] C.-Y. Hsieh, T.-J. Tan, J.-C. Chen, and C.-E. Wei, "5G Mutually Exclusive Access to Network Slices by Adaptively Prioritized Subset Algorithm," in *ICC 2021-IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6.
- [22] L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "SCOPE: An open and softwareized prototyping platform for NextG systems," in *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, 2021, pp. 415–426.
- [23] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "CoO-RAN: Developing machine learning-based xApps for open RAN closed-loop control on programmable experimental platforms," *IEEE Transactions on Mobile Computing*, 2022.
- [24] E. Faranda, "PCAPdroid." [Online]. Available: <https://emanuele-f.github.io/PCAPdroid/>
- [25] J. Groen, B. Kim, and K. Chowdhury, "The Cost of Securing O-RAN," in *IEEE International Conference on Communications (ICC)*, 2023.