Which ML model to choose? Experimental Evaluation for a beyond-5G Traffic Steering case

Ilias Chatzistefanidis*, Nikos Makris*†, Virgilios Passas*† and Thanasis Korakis*†
*Dept. of Electrical and Computer Engineering, University of Thessaly, Greece
†Centre for Research and Technology Hellas, CERTH, Greece
Email: ichatsist@uth.gr, nimakris@uth.gr, vipassas@uth.gr, korakis@uth.gr

***Abstract*—Beyond 5G and future next-generation networks will have to cope with the ever-growing traffic demand for mobile traffic, as well as low-latency communications. Network densification has been long proposed as a solution for augmenting the available wireless links with more technologies, thus enhancing the available capacity for the end-users. Nevertheless, selecting the optimal split of traffic among the available links is not a trivial decision. Machine Learning (ML) approaches can assist in these decisions, by forecasting metrics collected directly from the RAN, towards predicting the near-future performance, and appropriately selecting the split of traffic. In this work, we evaluate a total of 22 different ML models in such a traffic steering use case, towards determining the solution that yields the best results in terms of accuracy of predictions, training time, and computational resources. We use a real-world testbed prototype based on OpenAirInterface to evaluate our contributions, and use realistic mobility datasets for emulating client mobility. Our results show that the different algorithms can present variations in terms of the achievable throughput, but several can substantially improve the offered wireless network capacity.**

I. Introduction

The 5th generation of telecommunication networks (5G) have introduced several novel functionalities that add up to the optimized management and control of the deployed network. Beyond 5G networks are expected to add-up to the flexibility of management, by using standardized interfaces that allow the fine-grained control of the different functions at different layers. Through the definition of the Open RAN (O-RAN) [1] interfaces and messaging exchange schemes, the operator can use applications (usually mentioned as *xApps*) that analyze metrics and statistics collected directly from the RAN, and conclude on the optimal allocation of resources. Application of ML for Artificial Intelligence (AI) driven control of the network is key for the future network management. By accurately forecasting the monitored values upon which the decisions for the allocation is made, the operator can proactively apply decisions for the resource allocation, towards providing better services, meeting the demand, or ensuring that the overall network is energy efficient. The integration of such forecasting methods to the telecommunications network is also reflected in the definition of the 5G Network Data Analytics Function (NWDAF) [2]. Although NWDAF is mainly targeting at analyzing data at the 5G Core Network (CN) side, it is evident that the RAN can also benefit from such approaches in the context of beyond 5G and 6G networks.

The research leading to these results has received funding from the European Horizon 2020 Programme for research, technological development and demonstration under Grant Agreement Number No 101008468 (H2020 SLICES-SC). The European Union and its agencies are not liable or otherwise responsible for the contents of this document; its content reflects the view of its authors only.

Towards pushing the network capacity to meet new performance requirements in terms of bandwidth and latency, heterogeneous ultra-dense networks (UDN) have been proposed [3]. By deploying multiple RANs on non-overlapping spectrum, the operator can augment the network capacity, and select the different paths that can serve each client. Such functionality can be also integrated within the telecommunications network, by converging all the different networks at the Packet Data Convergence Protocol (PDCP) layer [4]. This, in turn, enables the operator to control the steering process in a per-packet basis, while traffic from heterogeneous technologies has a single entry/exit point (i.e. through the UPF).

In this work we blend the two concepts of traffic steering and ML-driven network optimization towards determining which forecasting algorithm is performing best in such a dense scenario. We use a real telecommunications network, with such programmable functionalities, acting as the convergence point at the base station level for multiple heterogeneous technologies. The prototype is based on OpenAirInterface, and the work presented in [5]. On top, we built a scheme for the collection of RAN related metrics, from the 3GPP network, and accordingly we feed different ML algorithms with the aim to determine which one performs better in such an environment. We use different traffic scenarios reflecting real car routes with fluctuating performance for the 3GPP network, and select the percentage of traffic split between different technologies. The main goals of this work are:

- To effectively forecast and infer through ML the estimated QoS of a mobile client, under different mobility patterns.
- To determine the traffic split in UDNs, among the available technologies and the per user estimated performance.
- To conclude on the optimal ML scheme that yields the highest accuracy/precision in measurements for our setup.
- To experimentally evaluate the contributions in a real environment, using realistic datasets and emulated mobility.

II. Related Work

ML integration for managing programmable network functions has received strong attention from the research community, as through the definition of programmable interfaces even on the RAN [6], intelligent provisioning of resources is possible. This is expected to play a key role for future beyond 5G and 6G communications, as network programming

ICC 2023 - IEEE International Conference on Communications | 978-1-5386-7462-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICC45041.2023.10279485

decisions can be taken on the fly based on the demand, while meeting the overall user satisfaction and energy efficiency [7].

Relevant literature on traffic steering mainly employ the Quality of Service (QoS) and Quality of Experience (QoE) metrics for determining the optimal split of traffic in UDNs. Authors in [8] present the design approach for a unified traffic steering framework, aiming at the orchestration of traffic steering features for optimal radio resource utilization. In [9], authors employ multi-tier cellular networks that combat performance issues induced by client mobility. A data-driven self-tuning algorithm for traffic steering is proposed to improve the overall QoE in multi-carrier 4G networks. In [10], authors use a QoE-driven traffic sharing algorithm based on mobility load balancing, that equalizes the QoE provided by all network cells. For this purpose, the handover margins between adjacent cells are tuned on a per-adjacency or per-service basis based on QoE measurements collected from the network management system. Authors in [11] develop a multiservice-type based transmission traffic scheduling optimization strategy, in order to integrate non-3GPP networks in the steering concept. Through their contributions, user satisfaction and effective capacity are always better than when using the always-best-connected and fixed-ratio power-allocation. Similarly, authors in [12] consider a similar topology, but approach the problem by considering two different classes of users in the system: single-homed users and multi-homed users. By applying proportional fairness to the network as a whole or to each isolated wireless access network, they improve the achievable throughput and stability conditions by applying a load balancing between accesses, thus increasing the overall network capacity.

Beyond integrating ML into the network, it is crucial to determine which algorithm better fits the under-study data. Therefore, several works focus on comparing different ML approaches. For example, in [13], authors integrate time-series based predictive analytics with the 5G Core and show a comparative study between two Time Series Forecasting Models-AutoRegressive Integrated Moving Average (ARIMA) and Facebook Prophet. In [14], authors compare supervised learning and time series analysis to predict the monthly rush-hour data traffic per cell in a live LTE network. They compare methods including Random Forest, different Neural Networks, Support Vector Regression, Seasonal ARIMA and Additive Holt–Winters. Their results show that supervised learning models outperform time series approaches. Finally, in [15], authors employ ML in a cognitive radio network. The authors investigate various time-series modeling approaches and Recurrent Neural Networks for predicting spectrum occupancy and conclude that the latter can provide higher accuracy.

In this work, we start from a traffic steering problem, with the management taking place at the RAN level, by integrating all the different accesses in a single converged base station unit. Based on the statistics received from the cellular network, we conduct an experimental campaign to determine which ML approach performs better, from a total of 22 different models. Using this approach, we determine which algorithms perform better, and subsequently, apply them to the traffic steering case.

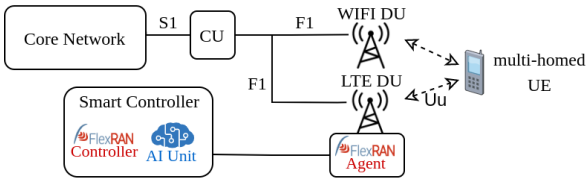


Fig. 1: 5G Disaggregated Topology providing heterogeneous UE connection based on OpenAirInterface and FlexRAN.

III. System Architecture and Methodology

This work evaluates experimentally several supervised ML and deep learning (DL) models when utilized for real-time time-series forecasting (TSF) of the User Equipment (UE) statistics in 5G networks. We work on univariate time-series of Channel Quality Indicator (CQI) values obtained from real car routes. Initially, in Section III-A we describe the network architecture, based on our previous work [5]. Section III-B demonstrates the Distributed Unit (DU) steering scheme, while Section III-C analyzes the process of emulating UE mobility and synthesizing traffic scenarios in the experimental environment. Section III-D provides details on data management and Section III-E presents the utilized ML models.

A. 5G Disaggregated Architecture

Fig. 1 depicts the disaggregated network architecture that supports heterogeneous wireless connection to a multi-homed UE. It is based on the 3GPP Option-2 split of the Radio Access Network (RAN) disaggregating the latter into one Central Unit (CU) and multiple Distributed Units (DUs). 3GPP (LTE) and Non-3GPP (WIFI) DUs are exploited to enforce the heterogeneous UE connection, being guided by the CU. The functional split between the CU and the DUs in the OSI stack takes place at the higher OSI layer 2, among PDCP and Radio Link Control (RLC) layers of the RAN. The CU is equipped with the PDCP and above layers and with the S1 interface to the core, while the DUs include the RLC, lower MAC and PHY layers. The CU and DUs are connected over an ethernet-based fronthaul exchanging traffic through the F1 interface employing the F1 Application Protocol (F1AP). As core and RAN, we use the LTE versions of the OpenAirInterface (OAI) as they were more stable at the experimentation time. However, the proposed implementation can be projected to the 5G NR version by replacing the LTE core components with the 5G ones and the disaggregated eNB with its gNB counterpart. We rely on the FlexRAN slicing communication between the Controller and the Agent to develop a dynamic DU steering mechanism that optimally steers the downlink traffic via the DUs. Therefore, we construct a smart controller by inserting an AI unit at the side of the FlexRAN Controller.

B. DU Steering Mechanism

The slices guide the CU towards appropriately steering the downlink traffic via the DUs in a per-packet basis. In order to apply the FlexRAN slicing mechanism in the disaggregated network, we forward the slice information to the CU and the

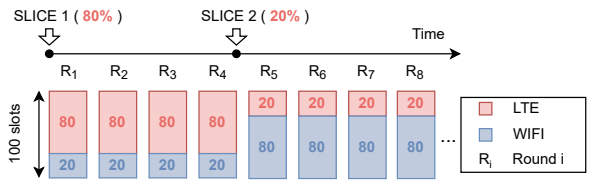


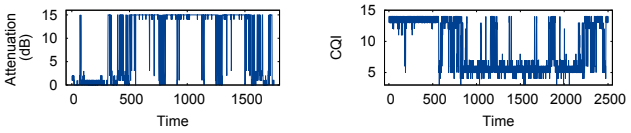
Fig. 2: Round-robin DU scheduler based on slices.

CU schedules the traffic to different DUs with weights (Fig. 2). Specifically, rounds (R_i) of 100 slots are created. In every round, each slot is utilized by one DU to send one packet. At first, the LTE DU assigns its slots based on the *Percentage* value. The remaining slots are used by the WIFI DU. In Fig. 2 the steering begins having a slice with a *Percentage* equal to 80%. Thus, the LTE DU allocates 80 slots, while the WIFI DU uses only 20 in R_1-R_4. Then, the second slice comes with a 20% reversing the number of DU slots. Overall, in R_1-R_4 the 80% of the traffic is steered via the LTE DU and the 20% via the WIFI DU, while in R_5-R_8 the figure is reversed.

Intelligence is inserted with an AI unit at the Controller's side. This unit predicts the LTE link quality and optimally steers the traffic to ensure the best end-user QoE. Noticeably, predictions for the WIFI one are not added yet since this work aims at finding the best ML models for the LTE link's quality forecasting. Thus, the WIFI link maintains high quality in the experiments in order to be used when the LTE quality decreases. This way, we will validate that the AI unit prevents the deterioration of the QoE by switching on time to the WIFI DU. For the LTE link, the unit analyzes the CQI; a metric posted by the UEs to the base station (BS). It is linked with the allocation of the UE modulation and coding schemes and ranges from 0 to 15. This is from BPSK to 64 QAM modulation, from zero to 0.93 code rate, from zero to 5.6 bits per symbol, from less than 1.25 to 20.31 SINR (dB) and from zero to 3840 Transport Block Size bits. Based on CQI forecasting, a CQI threshold is set at 9 (16-QAM, 8.75 SINR). When the CQI is expected to be more than 9, traffic is steered via the 3GPP and otherwise via the WIFI DU.

C. Emulating UE Mobility & Traffic Scenarios

This work analyzes CQI data obtained from cars driving through a specific route in the city of Volos, Greece. Our goal is to reproduce this traffic inside our experimental environment so as to evaluate the efficiency of the AI unit in realistic settings. To emulate UE mobility in the testbed, we install programmable attenuators on the outputs of the Software Defined Radio (SDR). Then, we configure the radio attenuation based on the patterns observed in real cars. Note that increasing the attenuation, decreases the UE's CQI and vice versa. At first, we possess only one basic traffic scenario, which contains radio attenuation values from one car that drove through the city route (Fig. 3a). Every other car that drives through this route is expected to have a similar attenuation pattern since it passes through the same geographical area. Only small variations from car to car are expected, depicting the different velocities, driving styles, road/weather conditions and noise.



(a) Basic Attenuation Scenario (b) Car's Collected CQI series

Fig. 3: Mobility Emulation using Attenuation Scenarios

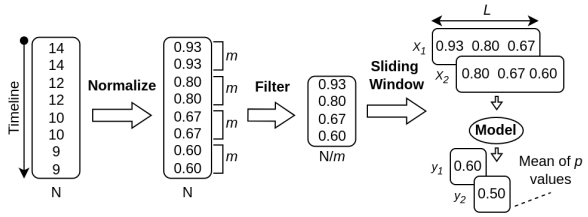


Fig. 4: Supervised Learning Time-series Pre-processing.

Keeping that in mind, we synthesize a lot of car scenarios employing data augmentation techniques using the Python Tsaug library [16]. First, Dynamic Time Warping (DTW) is applied to synthesize scenarios with different car velocities and driving styles. More specifically, the augmented cars randomly change their speed many times during the route (around every 1 km). Moreover, they reach the same places during the route with a time difference that varies from 0 to 30 seconds from the basic car used for the augmentation. Thus, the augmented car scenarios have velocities that vary from 40 km/h to 60 km/h approximately, since the basic car scenario has 50 km/h speed (road's limit). Also, Additive White Gaussian Noise (AWGN) with small variance is randomly added in every scenario representing the effects of different road/weather conditions. Further, many scenarios are cropped and scaled for effective training. Additional but minor noise will be added when we will collect the CQI data in the testbed. This process ensures that a large spectrum of the route's real traffic is captured. Hence, we build efficient AI models, able to generalize avoiding over-fitting and being immune to noise and possible fluctuations.

D. CQI Data Collection & Management

To collect the CQI data, we randomly select one attenuation scenario and execute it in the testbed. Concurrently, the AI unit parses and stores the CQI values utilizing the FlexRAN's REST API. Each CQI is obtained every 0.25 seconds. Consequently, the AI models forecast per 0.25 seconds. Overall, CQI values from 73 cars are collected. Each car has about 2500 CQI values and hence, the total number of collected values is 182500. Fig. 3 shows the basic attenuation scenario (3a), and also a CQI series collected from an attenuation scenario (3b).

Fig. 4 presents the data pre-prosessing. First, the values are rescaled into the range of $[0, 1]$ for efficient training. Then, a filtering algorithm reduces the dataset's size, keeping through the patterns. In specific, we keep the mean of every m indices creating a new series that is m times smaller. After that, the sliding window method is applied forming a supervised learning structure. We crop the filtered series into smaller subseries (X_i) of fixed length, L. Each X_i is the model's input

and is shifted by one index to the future from the previous one (X_{i-1}). Further, each X_i has a y_i pair, which is the label (single value). Noticeably, each y_i could represent either only one CQI value or the mean of multiple (p) values. We prefer the latter option as it gives a clearer picture of the near future and is not affected by fluctuations. The values gathered for each y_i are located after the respective X_i in the dataset. This way, they represent the future of each X_i. The models' optimal m, L and p values are calculated after extensive experimentation. Generally, we need an input window (m, L) that is large enough to capture the patterns and small enough to boost training. After experimenting, we concluded that an efficient window size for this problem is 400 CQI values (before pre-processing). That is about 100 seconds in the near-past. Thus, for the majority of the models, we configure m and L equal to 4 and 100 respectively (except Bi-LSTM that was better with $m = 5$ and $L = 80$). Moreover, p should be large enough to smoothen possible fluctuations and small enough to present an accurate perspective of the near future. We found that a value of 15 for p (after pre-processing) is efficient for the models; that is approx. 15 seconds of the near-future. Overall, each model observes the CQI data of the last 100 seconds, identifies the patterns and forecasts the average CQI for the next 15 seconds.

E. Algorithms

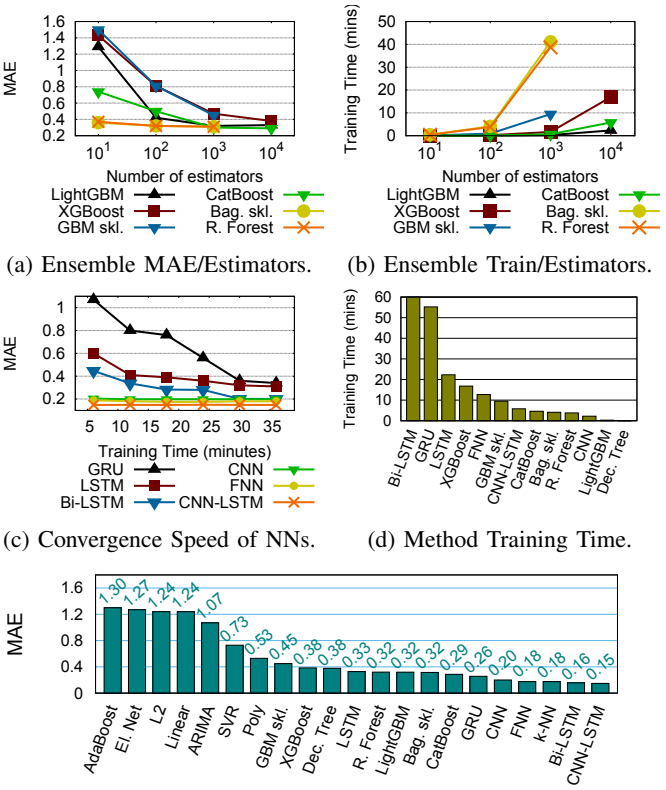
A plethora of AI methods is evaluated so as to find the best ones in terms of accuracy and convergence speed. First, Linear Regression is used, testing also regularization methods, such as L1, L2 and Elastic Net. Then, Polynomial, ARIMA, Support Vector (SVR), k-Nearest Neighbors (kNN) and Decision Tree Regression methods are used. For ARIMA, we did not use the last step of the pre-processing (sliding window) as it is not designed for supervised learning. Moreover, ensemble techniques are used, namely Bootstrap Aggregation (Bagging) and Gradient Boosting. For the former, Random Forest and the scikit-learn (sklearn) implementation of the Bagging method are used. Regarding the latter, we employ the XGBoost, LightGBM, sklearn GBM, CatBoost and AdaBoost Regression. Further, deep learning models are studied starting with a Feed-forward Neural Network (FNN). Then, we work with Recurrent Neural Networks (RNNs), including Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). We also test Bidirectional LSTMs (Bi-LSTM), Convolutional Neural Networks (CNN) and a hybrid CNN-LSTM model.

IV. Evaluation

Towards evaluating the AI unit, we assess the AI model performance in IV-A, while in IV-B we experimentally validate the unit's ability to optimize the QoE in the testbed.

A. Models Evaluation

Time-Series Cross Validation, a method similar to K-fold, is used to evaluate the model's error. The pre-processed data (45339 X_i, y_i samples) are split into several folds of equal size (500 samples). There are two sets; the training and the



(a) Ensemble MAE/Estimators. (b) Ensemble Train/Estimators.

(c) Convergence Speed of NNs. (d) Method Training Time.

(e) *MAE comparison of all algorithms.*

Fig. 5: Machine learning models' evaluation and comparison.

test set. Initially, we insert several serial folds (30000 samples - data of approx. 50 cars) to the training set following the timeline. At each iteration (i), the model is trained on the training set and is evaluated on the validation set (next fold on the timeline) calculating the generalization error on unseen data. In the next i, the training set is extended by one fold and the next one is used for a new validation. In the end, we calculate the mean of all validations folds (data from approx. 23 cars) as the final generalization error. As evaluation metric, we use the *Mean Absolute Error (MAE)* that calculates the sum of absolute errors between the forecasting ($\hat{y_i}$) and the labels (y_i) divided by the sample size (n):

$$MAE = \frac{\sum_{i=1}^{n} |\hat{y_i} - y_i|}{n}$$

Noticeably, MAE is scale-dependent and for the evaluation, we use $\hat{y_i}$, y_i in the original CQI scale, $(0, 15]$. Thus, the error is easily understood. For instance, a MAE of 2 is poor as the forecasting deviates by a mean of 2 CQI values. The ultimate goal is to find the models with the lowest MAE and training time. The former is necessary for accurate predictions, while the latter is essential for online-training implementations. There, the models are continuously updated with new data to capture new traffic patterns. To find the best models' hyper-parameters, the determining question is *"What is the right trade-off between training time and MAE?"*. Generally, more training time means better MAE. However, we should find the right balance for efficient resource management. Fig. 5 presents the model evaluation. All models are trained using

a non-subscription Tensor Processing Unit (TPU) in Google Colab Notebooks. Fig. 5a and 5b compare the ensemble methods when using a different number of estimators based on their achieved MAE and training time respectively. Fig. 5c shows the NNs' convergence speed. Then, after concluding on the models' structure (Tables I, II), Fig. 5e compares all methods based on MAE. The best ones (MAE below 0.5) are compared in Fig. 5d for their training time.

TABLE I: Neural Networks Configuration

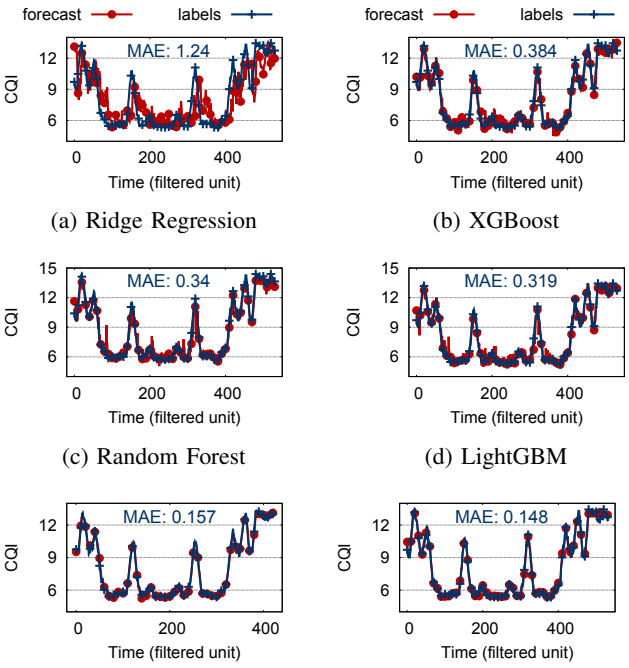
	Layers	Input	Hidden Layers	Epochs
GRU	2 GRU + Dense	(1, 100)	25 units per layer	51
LSTM	2 LSTM + Dense	(1, 100)	25 units per layer	19
Bi-LSTM	2 Bi-LSTM + Dense	(1, 80)	25 units per layer	52
FNN	2 Dense + output Dense	(1, 100)	25 units per layer	381
CNN	Conv1D + MaxPooling1D + Flatten + Dense + Dense (output)	(10, 10)	*filters=64, kernel size=2, pool size=2, 50 Dense units*	64
CNN-LSTM	Conv1D + MaxPooling1D + Flatten + RepeatVector + 2 LSTM + Dense (output)	(10, 10)	*filters=64, kernel size=3, pool size=2, repetition factor=1, 25 units per LSTM layer*	129

TABLE II: Configuration of Ensemble Methods.

	LightGBM	XGBoost	GBM skl.	CatBoost	Bag skl.	R. Forest
Estimators	10^3	10^4	10^3	10^4	10^2	10^2

Fig. 5a and 5b show that when the ensemble methods use more estimators, the MAE decreases but the training time increases. Noticeably, bagging techniques (Bag. sklearn, R. Forest) converge with fewer estimators than the boosting ones (Fig. 5a), but generally need more training time to handle the same estimators (Fig. 5b). This time increases dramatically with 10^3 and more estimators. Regarding boosting, LightGBM and CatBoost outperform XGBoost and GBM sklearn having lower MAE and training time in all estimators. Furthermore, CNN, FNN and CNN-LSTM (Fig. 5c) converge very fast in low MAE in contrast to GRU, LSTM and Bi-LSTM. Based on this, we conclude on the final structure of the NNs and ensemble methods in Tables I, II. Notably, NNs use $ReLU$ activations and the $Adam$ optimizer.

The results in Fig. 5e show that some methods perform poorly (MAE $>$ 1), such as Adaboost, Linear, L1, L2 (not on Fig. as it has MAE of 2.55), Elastic Net and ARIMA. SVR and polynomial perform better having MAE lower than 1. Ensemble methods and LSTM have excellent performance ranging from 0.45 to 0.29 MAE. Finally, k-NN and the other NNs show outstanding results reaching MAE values from 0.26 to only 0.15. Notably, Fig. 5d shows that most NNs need more training time than ensemble methods to reach these MAE values. Surprisingly, FNN, CNN-LSTM and CNN have similar training demands with the ensemble methods, but outperform them in terms of MAE. Importantly, these results should be studied when also considering the model configuration in Tables I, II. Generally, we configure the majority of the models in the most resource-efficient manner. For instance, we stop the LSTM training at approx. 22 minutes to save training resources since we do not observe large improvements afterwards. However, if we are willing to prioritize MAE, we could let the training continue. In some cases, we opted for the latter option to observe the results. For example, we could stop the Bi-LSTM's training earlier as it reached a sufficient MAE lower than 0.5. Instead, we let the training continue for up to 1 hour and hence, the model reaches an outstanding MAE of only 0.16. GRU's training could also be stopped earlier with an accuracy cost. This way, we present the impact of different decisions in the MAE vs. training time trade-off.



(a) Ridge Regression (b) XGBoost (c) Random Forest (d) LightGBM (e) Bi-LSTM (f) CNN-LSTM

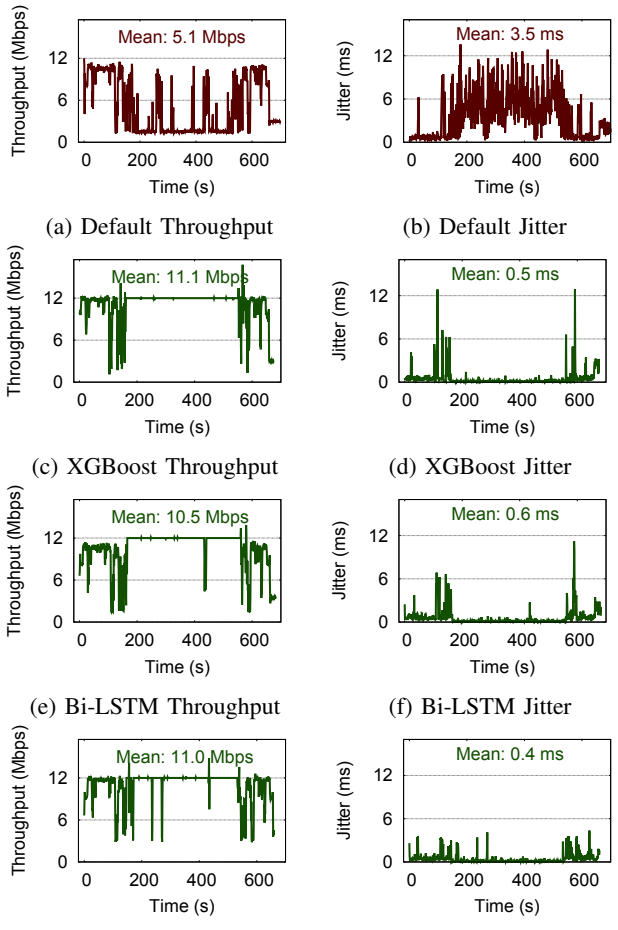
Fig. 6: CQI forecasting performance on Testbed.

Based on our results, we recommend using NNs for this problem as they are more resilient to noise and reach lower MAE values. As shown in Table I, we recommend using at least two hidden layers in NNs to capture non-linear patterns in data. We propose a combination of a Convolutional (CNN) with a Recurrent Neural Network (RNN), such as CNN-LSTM, as the most efficient solution. This forms an Encoder-Decoder architecture, with the CNN implementing the feature extraction, noise and dimensionality reduction procedures, while the LSTM finds the patterns using memory components. This way, a robust model with superb accuracy and training performance is built. Regarding the ensemble models, LightGBM and CatBoost are very promising solutions as they have great accuracy with little training time.

B. Experimental Evaluation

On the testbed, several models are evaluated on a specific car scenario to have a common ground for evaluation. At each experiment, one model is integrated into the AI unit to forecast the car's CQI values. Each model is trained in the complete training set with the data of the augmented scenarios (45339 samples). Moreover, the car scenario used on the testbed is the basic non-augmented one that was excluded from the training data. Fig. 6 presents the real-time CQI forecasting ($\hat{y_i}$) of the models, compared with the labels (y_i) during the exper-

iments, in order to better understand the impact of different MAE values. Moreover, Fig. 7 demonstrates the experienced (a) throughput and (b) jitter during the experiments. More specifically, Fig. 7a and 7b show the default steering, where the AI unit is not employed, steering the downlink traffic only via the LTE DU. Hence, the WIFI link is not exploited leading to poor end-user QoE in a major part of the route. In contrast, the AI unit correctly identifies the CQI patterns in all the other experiments (Fig. 7) and optimally steers the traffic via the DUs. Specifically, it predicts the two moments when the traffic needs to be redirected to the other DU. The first moment is at approx. 160 seconds when the LTE link's quality begins to deteriorate. Then the AI unit redirects the traffic on time towards the WIFI DU preventing the QoE drop. Moreover, the unit predicts that at approx. 550 seconds the LTE quality starts to increase and hence, redirects the traffic back to the LTE DU. This way, throughput is maximized and jitter is minimized. Overall, the selected algorithms achieve great results at matching CQI patterns and guiding the AI-unit to maximize the user QoE.



(a) Default Throughput (b) Default Jitter

(c) XGBoost Throughput (d) XGBoost Jitter

(e) Bi-LSTM Throughput (f) Bi-LSTM Jitter

(g) CNN-LSTM Throughput (h) CNN-LSTM Jitter

Fig. 7: Experimental validation for traffic steering

V. Conclusion

In this work, we present a novel traffic-steering approach and attempt to shed light on which ML approach yields the best results in forecasting future QoS metrics of the end users. We evaluated a total of 22 different ML models, towards determining which one performs better in terms of accuracy, training time and convergence speed. Given their performance, we conclude that CNN-LSTMs seem to provide a good-tradeoff between the produced errors and training time needed, though other approaches can yield results with similar MAE but high computing times (e.g. stacked LSTMs) or higher MAE but with significantly less training times (e.g. Forest-based methods). The results are applied in the traffic steering case through a novel architecture, that is able to enhance the total perceived QoE for multi-homed end-users. The tools and data contributions of this paper are also available in [17].

References

[1] A. Garcia-Saavedra and X. Costa-Pérez, "O-RAN: Disrupting the Virtualized RAN Ecosystem," *IEEE Communications Standards Magazine*, vol. 5, no. 4, pp. 96–103, 2021.

[2] 3GPP, "3GPP TS 33.521 V17.2.0 (2022-06); 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 5G Security Assurance Specification (SCAS); Network Data Analytics Function (NWDAF) (Release 17) ," 2022.

[3] S. Andreev, V. Petrov, M. Dohler, and H. Yanikomeroglu, "Future of Ultra-Dense Networks Beyond 5G: Harnessing Heterogeneous Moving Cells," *IEEE Communications Magazine*, vol. 57, no. 6, pp. 86–92, 2019.

[4] 3GPP, "3GPP TS 36.360 V14.0.0 (2017-03), 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); LTE-WLAN Aggregation Adaptation Protocol (LWAAP) specification (Release 14)," 2017.

[5] N. Makris, C. Zarafetas, P. Basaras, T. Korakis, N. Nikaein, and L. Tassiulas, "Cloud-Based Convergence of Heterogeneous RANs in 5G Disaggregated Architectures," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[6] M. Dryjański, Kułacz, and A. Kliks, "Toward Modular and Flexible Open RAN Implementations in 6G Networks: Traffic Steering Use Case and O-RAN xApps," *Sensors*, vol. 21, no. 24, 2021.

[7] X. Lin, "An overview of 5G advanced evolution in 3GPP release 18," *arXiv preprint arXiv:2201.01358*, 2022.

[8] M. Dryjanski and M. Szydelko, "A unified traffic steering framework for LTE radio access network coordination," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 84–92, 2016.

[9] C. Gijón, M. Toril, S. Luna-Ramírez, and M. Luisa Marí-Altozano, "A Data-Driven Traffic Steering Algorithm for Optimizing User Experience in Multi-Tier LTE Networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 10, pp. 9414–9424, 2019.

[10] M. L. Marí-Altozano, S. Luna-Ramírez, M. Toril, and C. Gijón, "A QoE-Driven Traffic Steering Algorithm for LTE Networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11 271–11 282, 2019.

[11] X. Ba, L. Jin, Z. Li, J. Du, and S. Li, "Multiservice-Based Traffic Scheduling for 5G Access Traffic Steering, Switching and Splitting," *Sensors*, vol. 22, no. 9, 2022.

[12] G. Dandachi, S. E. Elayoubi, T. Chahed, N. Chendeb, and H. Jebalia, "Comparing Resource Allocation Schemes in Multi-Homed LTE/WiFi Access Networks," in *2015 IEEE 82nd VTC-Fall*, 2015, pp. 1–6.

[13] P. Chakraborty, M. Corici, and T. Magedanz, "A comparative study for Time Series Forecasting within software 5G networks," in *2020 14th International Conference on Signal Processing and Communication Systems (ICSPCS)*, 2020, pp. 1–7.

[14] C. Gijón, M. Toril, S. Luna-Ramírez, M. L. Marí-Altozano, and J. M. Ruiz-Avilés, "Long-Term Data Traffic Forecasting for Network Dimensioning in LTE with Short Time Series," *Electronics*, vol. 10, 2021.

[15] A. Agarwal, A. S. Sengar, and R. Gangopadhyay, "Spectrum Occupancy Prediction for Realistic Traffic Scenarios: Time Series versus Learning-Based Models," *Journal of Communications and Information Networks*, vol. 3, no. 2, pp. 35–42, 2018.

[16] "Tsaug - Time Series Augmentation," [Online], https://github.com/arundo/tsaug.

[17] I. Chatzistefanidis, N. Makris, V. Passas, and T. Korakis, "UE Statistics Time-Series (CQI) in LTE Networks," 2022. [Online]. Available: https://dx.doi.org/10.21227/ec7p-xq38