UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

# Design and Development of Artificial Intelligence methods for efficient real-time scheduling of wireless clients in 5G and beyond networks

Diploma Thesis

**Ilias Chatzistefanidis**

**Supervisor:** Athanasios Korakis

Volos 2020

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

# Design and Development of Artificial Intelligence methods for efficient real-time scheduling of wireless clients in 5G and beyond networks

Diploma Thesis

**Ilias Chatzistefanidis**

**Supervisor:** Athanasios Korakis

Volos 2020

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

# Σχεδιασμός και Υλοποίηση αλγορίθμων τεχνητής νοημοσύνης για την χρονοδρομολόγηση σε πραγματικό χρόνο ασύρματων πελατών δικτύων 5ης και πέραν της 5ης γενιάς

Διπλωματική Εργασία

**Ηλίας Χατζηστεφανίδης**

**Επιβλέπων/πουσα:** Αθανάσιος Κοράκης

Βόλος 2020

Approved by the Examination Committee:

| | |
|---|---|
| Supervisor | **Athanasios Korakis**<br>Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly |
| Member | **Antonios Argyriou**<br>Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly |
| Member | **Dimitrios Bargiotas**<br>Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly |

Date of approval: 20-9-2020

# Acknowledgements

Many people helped and supported me through my undergraduate journey to fulfill my studies. I would like to express my deepest gratitude to a group of people that undoubtedly contributed and assisted in my degree completion. It is certain that I could not achieve it without them.

First, I would like to sincerely thank my supervisor, Associate Professor Thanasis Korakis, who supported me and gave me the opportunity to join and learn from such an inspiring research team in NITlab. Moreover, I could not forget mentioning the senior researcher and postdoctoral Nikos Makris, that guided me carefully through the telecommunication research world. I would like to underline that his help was one of the most valuable things for completing my studies. With his immense support, I learned so many things and got involved in numerous state-of-the-art projects. Finally, I would like to deeply thank the emeritus professor Elias N. Houstis that introduced me to the world of machine learning and artificial intelligence and provided me his support with his knowledge and resources.

Additionally, I want to express my gratitude to my friends and the people that were always on my side supporting me on this journey. Every single one knows that certainly has a special place in my heart.

Last but not least, I have no words to express my gratitude and love to my family. By taking a step backward and looking back in time through the past generations, it is undeniable that I would not even come close to where I am today without their love, support, and faith.

## DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Ilias Chatzistefanidis
15-9-2020

# Abstract

Towards enabling innovative and enormous scale applications such as 4K video streaming, Augmented Reality (AR), Virtual Reality (VR), eHealth, Holographic Telepresence, and many more, the urgent need for more powerful telecommunication network architectures is vital. Specifically, the user's data demand is growing more and more daily, leading to the research of more robust cellular network designs beyond 5G. Several approaches for the next generation systems propose integrating machine learning and artificial intelligence techniques through the whole network stack as a critical step towards creating self-organized networks adaptive to the fluctuations and the network conditions at every time. In this work, we design, develop, and evaluate an AI-Driven framework for real-time scheduling of wireless clients between heterogeneous technologies, such as WiFi, 5G, LTE, on a 5G disaggregated RAN architecture.

Building this framework requires two steps-milestones. First, we develop and evaluate a scheduling scheme for the different technologies on a disaggregated architecture based on LTE RAN slicing. Then, we extend this work by incorporating an ML & AI unit to monitor and predict the LTE channel's quality and make scheduling decisions for the technology to use. This way, it ensures enhanced network performance and optimized user Quality of Service (QoS) and Quality of Experience (QoE). To validate and evaluate our work in a real environment, we utilize the NITOS Testbed in Volos, Greece. Precisely, we implement a complete 5G disaggregated architecture utilizing the OpenAirInterface and FlexRAN platforms for the network infrastructure and commercial LTE USB Dongles for the User Equipment (UE). Moreover, we emulate a realistic car route inside the testbed using software-defined radios (SDR) and programmable attenuators. This way, we reproduce LTE channel quality fluctuation data collected from real commercial networks in Volos city. Finally, we compare the experimental results of the network performance when employing the ML & AI unit and when using the default network configuration to determine the efficiency of our framework.

# Περίληψη

Για την υλοποίηση καινοτόμων εφαρμογών τεράστιας κλίμακας, όπως η ροή βίντεο 4K, η Επαυξημένη Πραγματικότητα (AR), η Εικονική Πραγματικότητα (VR), η ηλεκτρονική υγεία, η ολογραφική τηλεπαρουσία και πολλά άλλα, η επείγουσα ανάγκη για ισχυρότερες αρχιτεκτονικές τηλεπικοινωνιακών δικτύων είναι ζωτικής σημασίας. Συγκεκριμένα, η ζήτηση δεδομένων των χρηστών αυξάνεται καθημερινά όλο και περισσότερο, γεγονός που οδηγεί στην έρευνα για πιο ισχυρά σχέδια κυψελοειδών δικτύων πέραν του 5G. Αρκετές προσεγγίσεις για τα συστήματα επόμενης γενιάς προτείνουν την ενσωμάτωση τεχνικών μηχανικής μάθησης και τεχνητής νοημοσύνης σε ολόκληρη τη στοίβα του δικτύου ως ένα κρίσιμο βήμα προς τη δημιουργία αυτοοργανωμένων δικτύων που προσαρμόζονται στις διακυμάνσεις και τις συνθήκες του δικτύου ανά πάσα στιγμή. Στην παρούσα εργασία, σχεδιάζουμε, αναπτύσσουμε και αξιολογούμε ένα πλαίσιο βασισμένο στην τεχνητή νοημοσύνη για τον προγραμματισμό σε πραγματικό χρόνο των ασύρματων πελατών μεταξύ ετερογενών τεχνολογιών, όπως WiFi, 5G, LTE, σε μία 5G διαχωρισμένη αρχιτεκτονική RAN.

Η δημιουργία αυτού του πλαισίου απαιτεί δύο βήματα-ορόσημα. Πρώτον, αναπτύσσουμε και αξιολογούμε ένα σχήμα χρονοπρογραμματισμού για τις διαφορετικές τεχνολογίες σε μια διαχωρισμένη αρχιτεκτονική που βασίζεται στο LTE RAN slicing. Στη συνέχεια, επεκτείνουμε αυτή την εργασία με την ενσωμάτωση μιας μονάδας ML & AI για την παρακολούθηση και πρόβλεψη της ποιότητας του καναλιού LTE και τη λήψη αποφάσεων χρονοπρογραμματισμού για την τεχνολογία που θα χρησιμοποιηθεί. Με αυτόν τον τρόπο, εξασφαλίζεται ενισχυμένη απόδοση δικτύου και βελτιστοποιημένη ποιότητα υπηρεσίας (QoS) και ποιότητα εμπειρίας (QoE) των χρηστών. Για να επικυρώσουμε και να αξιολογήσουμε την εργασία μας σε πραγματικό περιβάλλον, χρησιμοποιούμε το NITOS Testbed στο Βόλο, Ελλάδα. Συγκεκριμένα, υλοποιούμε μια πλήρη διαχωρισμένη αρχιτεκτονική 5G χρησιμοποιώντας τις πλατφόρμες OpeanAirInterface και FlexRAN για την υποδομή δικτύου και εμπορικά LTE USB Dongles για τον εξοπλισμό χρήστη (UE). Επιπλέον, προσομοιώνουμε μια ρεαλιστική διαδρομή αυτοκινήτου μέσα στο δοκιμαστικό περιβάλλον με τη χρήση ραδιοσυχνοτήτων καθορισμένων από λογισμικό (SDR) και προγραμματιζόμενων εξασθενητών. Με αυτόν τον τρόπο, αναπαράγουμε τα δεδομένα διακύμανσης της ποιότητας του καναλιού LTE που συλλέγονται από πραγματικά εμπορικά δίκτυα στην πόλη του Βόλου. Τέλος, συγκρίνουμε τα πειραματικά αποτελέσματα της απόδοσης του δικτύου όταν χρησιμοποιούμε τη μονάδα ML & AI και όταν χρησιμοποιούμε την προεπιλεγμένη διαμόρφωση του δικτύου για να προσδιορίσουμε την αποτελεσματικότητα του πλαισίου μας.

# Table of contents

**Acknowledgements** **ix**

**Abstract** **xi**

**Περίληψη** **xiii**

**Table of contents** **xv**

**List of figures** **xix**

**List of tables** **xxiii**

**Abbreviations** **xxv**

**1 Introduction** **1**

1.1 Motivation and Problem Statement . . . . . 1

1.1.1 Contribution . . . . . 2

1.2 Thesis Synopsis . . . . . 2

**2 Background** **5**

2.1 Introduction to Mobile Telecommunication Systems . . . . . 5

2.2 4G Mobile Networks . . . . . 6

2.2.1 Introduction . . . . . 6

2.2.2 Key Technologies . . . . . 6

2.2.3 LTE Architecture . . . . . 8

2.2.4 LTE Protocol Stack . . . . . 10

2.3 5G Mobile Networks . . . . . 13

2.3.1 Introduction . . . . . 13

2.3.2 Key Technologies . . . . . . . . . . . . . . . . . . . . . . . . . . 14
2.3.3 Functional Split Architecture . . . . . . . . . . . . . . . . . . . . . 16
2.4 Future Networks Beyond 5G . . . . . . . . . . . . . . . . . . . . . . . 18
2.4.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
2.4.2 Application of ML/AI in 6G Networks . . . . . . . . . . . . . . 19
2.5 AI & Machine Learning . . . . . . . . . . . . . . . . . . . . . . . . . . 20
2.5.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
2.5.2 Supervised Learning . . . . . . . . . . . . . . . . . . . . . . . . . 21
2.5.3 Machine Learning Steps . . . . . . . . . . . . . . . . . . . . . . . 22
2.6 Experimental Tools and Methods . . . . . . . . . . . . . . . . . . . . . 23
2.6.1 NITOS Testbed . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
2.6.2 The OpenAirInterface Platform . . . . . . . . . . . . . . . . . . . 26
2.6.3 The FlexRAN Platform . . . . . . . . . . . . . . . . . . . . . . . 26

**3 Framework for Scheduling among Heterogeneous DUs 29**
3.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
3.2 Framework . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30
3.2.1 Initial Topology . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30
3.2.2 Proposed Approach . . . . . . . . . . . . . . . . . . . . . . . . . . 30
3.3 Experiments . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32
3.3.1 Experimental Setup . . . . . . . . . . . . . . . . . . . . . . . . . . 32
3.3.2 Experimental Scenario . . . . . . . . . . . . . . . . . . . . . . . . 33
3.3.3 Experimental Results . . . . . . . . . . . . . . . . . . . . . . . . . 33

**4 AI-Driven Real-time Scheduling between Heterogeneous DUs 35**
4.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
4.2 System Architecture . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36
4.3 Data . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
4.3.1 Attenuation Data . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
4.3.2 Designing Basic Attenuation Scenario . . . . . . . . . . . . . . . 38
4.3.3 Attenuation Data Augmentation . . . . . . . . . . . . . . . . . . . 39
4.3.4 CQI Data Collection . . . . . . . . . . . . . . . . . . . . . . . . . 40
4.3.5 CQI Data Pre-processing . . . . . . . . . . . . . . . . . . . . . . . 42

4.4 Searching and Evaluation of ML & AI Algorithms . . . . . . . . . . . . . 43
4.4.1 Linear Regression . . . . . . . . . . . . . . . . . . . . . . . . . 44
4.4.2 Ridge Regression . . . . . . . . . . . . . . . . . . . . . . . . . 45
4.4.3 Lasso Regression . . . . . . . . . . . . . . . . . . . . . . . . . 47
4.4.4 Elastic Net Regression . . . . . . . . . . . . . . . . . . . . . . 48
4.4.5 Polynomial Regression . . . . . . . . . . . . . . . . . . . . . . 50
4.4.6 ARIMA . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
4.4.7 Support Vector Regression . . . . . . . . . . . . . . . . . . . . 53
4.4.8 k-Nearest Neighbors . . . . . . . . . . . . . . . . . . . . . . . 55
4.4.9 Decision Tree . . . . . . . . . . . . . . . . . . . . . . . . . . . 56
4.4.10 Random Forest . . . . . . . . . . . . . . . . . . . . . . . . . . 58
4.4.11 Bagging Scikit-learn . . . . . . . . . . . . . . . . . . . . . . . 59
4.4.12 XGBoost . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
4.4.13 LightGBM . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61
4.4.14 GBM Scikit-learn . . . . . . . . . . . . . . . . . . . . . . . . . 62
4.4.15 CatBoost . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 62
4.4.16 AdaBoost . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 63
4.4.17 Feedforward Neural Network . . . . . . . . . . . . . . . . . . . 64
4.4.18 Long short-term memory (LSTM) . . . . . . . . . . . . . . . . . 66
4.4.19 Bidirectional LSTM . . . . . . . . . . . . . . . . . . . . . . . 67
4.4.20 Gated Recurrent Unit (GRU) . . . . . . . . . . . . . . . . . . . 68
4.4.21 Convolutional Neural Network (CNN) . . . . . . . . . . . . . . . 68
4.4.22 CNN-LSTM Neural Network . . . . . . . . . . . . . . . . . . . 69
4.4.23 Models Comparison . . . . . . . . . . . . . . . . . . . . . . . . 70
4.5 Experiments . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 72
4.5.1 Experimental Setup . . . . . . . . . . . . . . . . . . . . . . . . 72
4.5.2 Experimental Scenario . . . . . . . . . . . . . . . . . . . . . . 73
4.5.3 Experimental Results . . . . . . . . . . . . . . . . . . . . . . . 74

**5 Conclusion** **77**
5.1 Summary and Conclusions . . . . . . . . . . . . . . . . . . . . . . . . 77

**Bibliography** **79**

# List of figures

2.1 LTE High-level Architecture . . . . . . . . . . . . . . . . . . . . . . . . . 8
2.2 High-level Protocol Architecture . . . . . . . . . . . . . . . . . . . . . . 10
2.3 Protocols used on the air interface and by the fixed network. . . . . . . . . 11
2.4 C-plane Protocol Stack . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
2.5 U-plane Protocol Stack . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
2.6 C-RAN Architecture: BBU-pool and RRHs . . . . . . . . . . . . . . . . . 17
2.7 Functional Splits in LTE Protocol Stack proposed by 3GPP. . . . . . . . . . 18
2.8 CU/DU architecture for the PDCP/RLC split . . . . . . . . . . . . . . . . . 18
2.9 Machine Learning Applications in 6G Networks . . . . . . . . . . . . . . . 19
2.10 Artificial Intelligence Domains . . . . . . . . . . . . . . . . . . . . . . . 21
2.11 Machine Learning Pipeline from data collection to prediction . . . . . . . . 22
2.12 Overall NITOS Architecture . . . . . . . . . . . . . . . . . . . . . . . . 24
2.13 Indoor RF Isolated Testbed . . . . . . . . . . . . . . . . . . . . . . . . . 25
2.14 Outdoor Testbed . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
2.15 Office Testbed . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25

3.1 Heterogeneous and Disaggregated RAN Architecture incorporating FlexRAN . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30
3.2 A Downlink FlexRAN Slice Example . . . . . . . . . . . . . . . . . . . . 31
3.3 The Message Format with the Slicing Values to be sent via F1 Interface. . . 31
3.4 Proposed Approach for Scheduling between WiFi and LTE DUs . . . . . . 31
3.5 PDCP Scheduling Technique for LTE and WiFi DUs . . . . . . . . . . . . 32
3.6 Experimental Setup with NITOS Testbed. . . . . . . . . . . . . . . . . . . 32
3.7 UE’s LTE Interface . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
3.8 UE’s WiFi Interface . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34

3.9 LTE/WiFi Bandwidth Monitoring with FlexRAN Slice Reconfiguration . . 34

4.1 Overall System Architecture incorporating Real-time Scheduling with the ML & AI Unit . . . . . . . . . . . . . . . . . . . . . . . . . . . 36
4.2 Attenuation Data Available from Real Commercial Networks . . . . . . . . 37
4.3 Basic Attenuation Data Scenario including car turn-around and red traffic light. 39
4.4 Attenuation Data Augmentation: Simulating the continuous monitoring of a specific city pathway. Every augmented scenario corresponds to a car that traverses the pathway with a unique driving style. . . . . . . . . . . . . . . 40
4.5 Nitos Testbed Data Collection Architecture . . . . . . . . . . . . . . . . 41
4.6 Collected CQI data. . . . . . . . . . . . . . . . . . . . . . . . . . . . 41
4.7 CQI Data Pre-processing: Sequence Flattening & Sliding Window Techniques . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42
4.8 Time Series Cross-Validation Technique: Every square represents one Fold. This figure shows five iterations where we increase the training set by one Fold and use the following one for validation. In the end, we obtain the mean of all validation errors. . . . . . . . . . . . . . . . . . . . . . . . . . 43
4.9 Linear Regression Generalization Evaluation . . . . . . . . . . . . . . . . 45
4.10 Time Series Cross-Validation Evaluation of Linear Regression Model. . . . 45
4.11 Ridge Regression Grid-Searching of Hyperparameter *alpha*. . . . . . . . . 46
4.12 Ridge Regression Generalization Evaluation with *alpha* = 3 . . . . . . . . 46
4.13 Lasso Regression Grid-Searching of Hyperparameter alpha. . . . . . . . . . 47
4.14 Lasso Regression Generalization Evaluation with *alpha* = 0.1 . . . . . . . 48
4.15 Elastic Net Regression Grid-Searching of Hyperparameters alpha and l1_ratio. 49
4.16 Elastic Net Regression Generalization Evaluation with *alpha* = 0.01 and *l1_ratio* = 0.01 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49
4.17 Polynomial Regression Generalization Evaluation with $2^{nd}$-order Degree . 50
4.18 Polynomial Regression Model Grid-Searching of Hyperparameter *degree* . 50
4.19 ARIMA Hyperparameter Grid-Searching. . . . . . . . . . . . . . . . . . 52
4.20 Seasonal ARIMA Hyperparameter Grid-Searching. . . . . . . . . . . . . 52
4.21 ARIMA (p=2, d=1, q=2) Generalization Evaluation. . . . . . . . . . . . . 52
4.22 One-Dimensional Linear Support Vector Regression . . . . . . . . . . . . 53
4.23 SVR Generalization Evaluation with $C = 10$ . . . . . . . . . . . . . . . . 54

4.24 SVR Model Grid-Searching of Regularization Hyperparameter $C$ . . . . . . 54
4.25 k-NN Generalization Evaluation with $k = 23$ . . . . . . . . . . . . . . . . 56
4.26 k-NN Model Grid-Searching of Regularization Hyperparameter $k$ . . . . . 56
4.27 Decision Tree Model Grid-Searching . . . . . . . . . . . . . . . . . . . . . 57
4.28 Decision Tree Generalization Evaluation with $max\_features = 10$ and $min\_samples\_leaf = 1$ . . . . . . . . . . . . . . . . . . . . . . . . . . 57
4.29 Random Forest Generalization Evaluation with $n\_estimators = 200$ . . . 58
4.30 Random Forest Model Grid-Searching . . . . . . . . . . . . . . . . . . . 58
4.31 Scikit-learn Bagging Generalization Evaluation with $n\_estimators = 200$ . 59
4.32 Scikit-learn Bagging Model Grid-Searching . . . . . . . . . . . . . . . . 59
4.33 XGBoost Generalization Evaluation with $n\_estimators = 10000$ . . . . . 60
4.34 XGBoost Model Grid-Searching . . . . . . . . . . . . . . . . . . . . . . . 60
4.35 LightGBM Generalization Evaluation with $n\_estimators = 1000$ . . . . . 61
4.36 LightGBM Model Grid-Searching . . . . . . . . . . . . . . . . . . . . . . 61
4.37 Scikit learn GBM Generalization Evaluation with $n\_estimators = 1000$ . . 62
4.38 Scikit learn GBM Model Grid-Searching . . . . . . . . . . . . . . . . . . 62
4.39 CatBoost Generalization Evaluation with $n\_estimators = 10000$ . . . . . 63
4.40 CatBoost Model Grid-Searching . . . . . . . . . . . . . . . . . . . . . . . 63
4.41 AdaBoost Generalization Evaluation with $n\_estimators = 10$ . . . . . . . 64
4.42 AdaBoost Model Grid-Searching . . . . . . . . . . . . . . . . . . . . . . . 64
4.43 Feedforward Neural Network Structure . . . . . . . . . . . . . . . . . . . 64
4.44 Feedforward NN Generalization Evaluation . . . . . . . . . . . . . . . . 65
4.45 LSTM Generalization Evaluation . . . . . . . . . . . . . . . . . . . . . . 66
4.46 Bidirectional LSTM Generalization Evaluation . . . . . . . . . . . . . . . 67
4.47 GRU Generalization Evaluation . . . . . . . . . . . . . . . . . . . . . . . 68
4.48 CNN Generalization Evaluation . . . . . . . . . . . . . . . . . . . . . . . 69
4.49 CNN-LSTM Generalization Evaluation . . . . . . . . . . . . . . . . . . . 70
4.50 Overall Comparison between the various ML & AI Model based on the Generalization MAE from the non-augmented data. . . . . . . . . . . . . . . . 71
4.51 Overall Comparison between the various ML & AI Model based on the training time on the augmented data. . . . . . . . . . . . . . . . . . . . . . . 72
4.52 Experimental Topology utilizing six machines on NITOS Testbed . . . . . 73

4.53 UE Experienced Bandwidth, Jitter and Packet Loss without and with the incorporation of the ML & AI Unit . . . . . . . . . . . . . . . . . . . . . . . . 75

# List of tables

2.1 Minimum 5G technical performance requirements (ITU-R) . . . . . . . . . . 15

4.1 Feedforward NN Configuration . . . . . . . . . . . . . . . . . . . . . . . . 65
4.2 LSTM Configuration . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 66
4.3 Bidirectional LSTM Configuration . . . . . . . . . . . . . . . . . . . . . . 67
4.4 GRU Configuration . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 68
4.5 CNN Configuration . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 69
4.6 CNN-LSTM Configuration . . . . . . . . . . . . . . . . . . . . . . . . . . . 70

# Abbreviations

| | |
|---|---|
| 1G, ..., 6G | $1^{st}$ Generation, ..., $6^{th}$ Generation |
| 3GPP | 3rd Generation Partnership Project |
| AI | Artificial Intelligence |
| ANN | Artificial Neural Network |
| APN | Access Point Name |
| ARIMA | Autoregressive Integrated Moving Average |
| ARQ | Automatic Repeat Request |
| Bagging | Bootstrap Aggregation |
| BBU | Baseband Unit |
| BLER | Block Error Rate |
| CBC | Cell Broadcast Centre |
| CDMA | Code Division Multiple Access |
| CNN | Convolutional Neural Network |
| CQI | Channel Quality Indicator |
| CRC | Cyclic Redundancy Check |
| CU | Central Unit |
| DU | Distributed Unit |
| EC | Edge Computing |
| EI | Edge Intelligence |
| EIR | Equipment Identity Registry |
| eMBB | Enhanced Mobile Broadband |
| eNB | evolved Nodes B |
| E-UTRAN | Evolved UMTS Terrestrial Radio Access Network |
| EPC | Evolved Packet Core |
| EPS | Evolved Packet System |

| | |
|---|---|
| ETWS | Earthquake and Tsunami Warning System |
| FDMA | Frequency Division Multiple Access |
| FNN | Feedforward Neural Network |
| GBM | Gradient Boosting Machine |
| GTP-U | GPRS tunneling protocol user part |
| GRU | Gated Recurrent Unit |
| HARQ | Hybrid Automatic Repeat Request |
| HSS | Home Subscriber Server |
| IP | Internet Protocol |
| ISI | Intersymbol Interference |
| LSTM | Long Short-Term Memory |
| LTE | Long-Term Evolution |
| MAC | Medium Access Control |
| MAE | Mean Absolute Error |
| MIMO | Multiple-Input and Multiple-Output |
| ML | Machine Learning |
| MME | Mobility Management Entity |
| mMTC | Massive Machine-Type Communications |
| MSE | Mean Squared Error |
| NITlab | Network Implementation Testbed Laboratory |
| NITOS | Network Implementation Testbed based on Open-Source software |
| OAI | OpenAirInterface |
| OFDMA | Orthogonal Frequency Division Multiple Access |
| PCEF | Policy Control Enforcement Function |
| PCRF | Policy Control and Charging Rules Function |
| PDCCH | Physical Downlink Control Channel |
| PDN | Packet Data Network |
| P-GW | Packet Data Network Gateway |
| QoE | Quality of Experience |
| QoS | Quality of Service |
| PDCP | Packet Data Convergence Protocol |
| RB | Resource Block |

| | |
|---|---|
| RLC | Radio Link Control |
| RNN | Recurrent Neural Network |
| RRC | Radio Resource Control |
| RRH | Remote Radio Head |
| SCTP | Stream Control Transmission Protocol |
| SDN | Software-Defined Networking |
| SDR | Software Defined Radio |
| S-GW | Serving Gateway |
| SRS | Sounding Reference Signal |
| TCP | Transmission Control Protocol |
| TDMA | Time Division Multiple Access |
| UDP | User Datagram Protocol |
| UE | User Equipment |
| UMTS | Universal Mobile Telecommunications System |
| URLLC | Ultra-Reliable and Low-Latency Communications |
| USRP | Universal Software Radio Peripheral |

# Chapter 1

# Introduction

Telecommunication systems have been evolved tremendously over the past few years to fill the users' needs. Specifically, the rise of the internet and the evolution of mobile devices bring a tremendous need for data that prior cellular architectures could not support. For this reason, the telecommunication systems structure is continuously growing, starting from the first generation systems (1G) to currently the fifth generation (5G).

Nevertheless, the digitization and rising of numerous industries results in a massive increase in the data flow now more than ever. Precisely, new applications are coming, including Augmented Reality (AR), Virtual Reality (VR), eHealth, holographic telepresence, among many others. Thus, the need for more robust and powerful systems is growing, leading us to research the next generation of cellular networks beyond 5G.

Many works [1], [2], [3] study the necessary steps towards enabling next-generation systems, highlighting the importance of machine learning and artificial intelligence. In detail, the wide network softwarization creates opportunities for innovations in every component of the network stack. With the integration of ML & AI techniques, the network could become self-organizing to overcome fluctuations and adapt to the various network conditions resulting in enhanced network management, QoS and QoE.

## 1.1 Motivation and Problem Statement

In this thesis, we design, develop, and evaluate an AI-Driven framework for the real-time scheduling of wireless clients. Precisely, we build on top of the disaggregated RAN architecture that exists in 5G networks [4] by implementing a scheduling technique between the heterogeneous wireless network technologies, such as LTE, 5G, WiFi, and more. In real-time, we utilize machine learning methods to predict future network conditions and make scheduling decisions to improve network performance and QoE.

We complete this work on two milestones. First, we develop the scheduling scheme among the heterogeneous technologies, utilizing the OpenAirInterface and the FlexRAN platforms. Then, we extend this work by inserting ML & AI methods to handle the scheduling by real-time monitoring the channel quality. To achieve it, we emulate channel quality patterns based on real car routes in Volos city.

Finally, we validate and evaluate this work by testing it in a real environment. Specifically, we deploy a complete cellular disaggregated network in NITOS Testbed with the help of the OpenAirInterface and FlexRAN platforms utilizing real devices. Moreover, we emulate a realistic car route in the testbed by employing SDRs and programmable attenuators. Finally, we deploy and evaluate our ML & AI unit in real-time, making conclusions about its contribution in enhancing the network performance and user's QoS and QoE.

### 1.1.1 Contribution

The overall thesis contributions are summarized as follows:

- To develop an efficient manner of scheduling the network downlink traffic among heterogeneous technologies.
- To develop a robust ML & AI unit that monitors and predicts the LTE channel quality and also makes efficient scheduling decisions that ensure enhanced network performance.
- To investigate and evaluate numerous machine learning techniques for finding the most appropriate models for real-time LTE channel quality forecasting.
- To evaluate the developed framework in a testbed environment, using real devices in real-time.

## 1.2 Thesis Synopsis

In chapter 2, we provide the essential background knowledge needed to understand the contribution of our work. Precisely, we present the evolution of the telecommunication systems, the structure of the functional split disaggregated architecture, the research direction for future networks beyond 5G, and the machine learning methodologies. Finally, we refer to the experimental environment used for the evaluation of our work. In chapter 3, we design, develop, and evaluate the framework for scheduling between heterogeneous wireless technologies. Chapter 4 extends this work by designing, developing, and evaluating the ML & AI unit that monitors and predicts the channel quality and makes scheduling decisions in real-time. To achieve this, we study and evaluate a large number of machine learning algorithms. Finally, chapter 5 summarizes our work and provides the conclusion and the thoughts for future work.

# Chapter 2

# Background

## 2.1 Introduction to Mobile Telecommunication Systems

In the early 1980s, mobile telecommunication systems were initially launched [5]. The first fully functional systems (1G) relied on analog techniques, being, in this way, unable to utilize the available spectrum. Due to their big size and high cost, mobile devices were exclusively available for corporate usage.

The breakthrough moment was the introduction of 2G systems in the early 1990s [6]. They provide digitization and a more efficient bandwidth/spectrum allocation, applying multiple access schemes, such as FDMA, TDMA, and CDMA. Thus, they improved the voice quality significantly and originated the first data services. Moreover, they brought a semi-global roaming system to advance connectivity internationally.

The third generation (3G) systems were introduced in the years after 2000, supporting broadband and multimedia services. Some of their key features are the IP network technology, the enhancement in voice quality and data rate, global roaming and Quality of Service (QoS).

In the beginning, voice calls dominated the traffic in cellular networks. Nevertheless, the years to come brought an enormous growth of mobile data and massive demand for low latency, mobility, and enhanced QoS. As a result, it was critical to build more robust networks.

## 2.2 4G Mobile Networks

### 2.2.1 Introduction

The 3GPP [7] established the fourth-generation systems as the result of research into the long-term evolution of UMTS. They are referred to as 3GPP Long-Term Evolution (LTE) systems and are widely known as LTE.

LTE appeared to be a promising technology for meeting users' demands for reduced latency and high data rates [6] [8] [9]. It was required to deliver a downlink peak data rate of 100 Mbps and an uplink peak data rate of 50 Mbps. The final system surpassed these criteria, with peak data speeds of 300 Mbps and 75 Mbps, respectively. However, these performances are only possible under idealistic circumstances and are entirely unattainable in any actual scenario. Thus, the spectral efficiency, which reflects the typical capacity of one cell per unit bandwidth, is a better metric. In the downlink, LTE has to enable a spectral efficiency three to four times higher than Release 6 WCDMA (3G) and two to three times greater in the uplink.

Another significant issue is latency, which is highly essential for time-sensitive applications like speech and interactive games. There are two aspects to consider. Firstly, the standards specify that data should move between a mobile phone and a fixed network in less than 5 milliseconds if the air interface is not congested. Secondly, there are two operation modes in mobile phones: an active mode in which they are connecting with the network and a low-power idle one. The standards stipulate that a phone shall go from idle to active status in fewer than 100 milliseconds following a user intervention.

Finally, there are also coverage and mobility specifications. LTE is enhanced for cell ranges up to 5 kilometers, operates with decreased performance up to 30 kilometers, and can support cell sizes up to 100 kilometers. It's also suited for mobile velocities of up to 15 km/h, has valuable performance up to 120 km/h, and can handle speeds up to 350 km/h. Finally, LTE offers a wide range of bandwidths, spanning from 1.4 MHz to a maximum of 20 MHz.

### 2.2.2 Key Technologies

LTE incorporates numerous essential technologies that allow it to deliver all the features mentioned. Some of the most important are worth mentioning.

- **Orthogonal Frequency Division Multiple Access (OFDMA):** LTE utilizes OFDM for the downlink to transmit data over several narrowband carriers instead of distributing one signal over the entire carrier bandwidth. The OFDM symbols are organized into resource blocks, and every user allocates a number of them in the time-frequency grid. The ability of OFDM to cope with extreme channel circumstances without the need for sophisticated equalization filters is its dominant benefit. Furthermore, because of the low symbol rate, inter-symbol interference (ISI) may be avoided.

- **Multiple-Input and Multiple-Output (MIMO)**: MIMO is a technique for increasing the capacity of a radio link by employing multiple transmission and receiving antennas and thus utilizing multipath propagation. MIMO antennas benefit mobile radio systems, with more dependable operation in low signal situations, enhanced spectral efficiency, and higher data rates for individual users. The LTE standard enables several MIMO operation modes and allows dynamic mode switching adapting to various radio situations.

- **Link Adaptation:** LTE optimizes air interface efficiency by selecting the appropriate modulation and channel coding rate based on several essential metrics. Channel Quality Indicator (CQI) is a primary metric that indicates the received SINR and thus the channel quality as experienced by the UE. Moreover, Physical Downlink Control Channel (PDCCH) provides scheduling information to individual UEs when Block Error Rate (BLER) defines the in-sync or out-of-sync indication during radio link monitoring. Finally, Sounding Reference Signal (SRS) gives a more precise estimation of the user's uplink channel.

- **Hybrid Automatic Repeat Request (HARQ)** LTE adopts an enhanced version of *Automatic Repeat Request* (ARQ) called HARQ. When a block of data fails the *Cyclic Redundancy Check* (CRC) in the simple ARQ, the receiver discards it. On the other hand, HARQ keeps any meaningful data in a buffer and continues to the phase of error detection/correction. If the CRC fails, the data is sent again by the transmitter. Now, however, the receiver analyzes the data from the buffer along with the re-transmissions. In this way, the signal energy rises at the receiver, leading to improved performance.

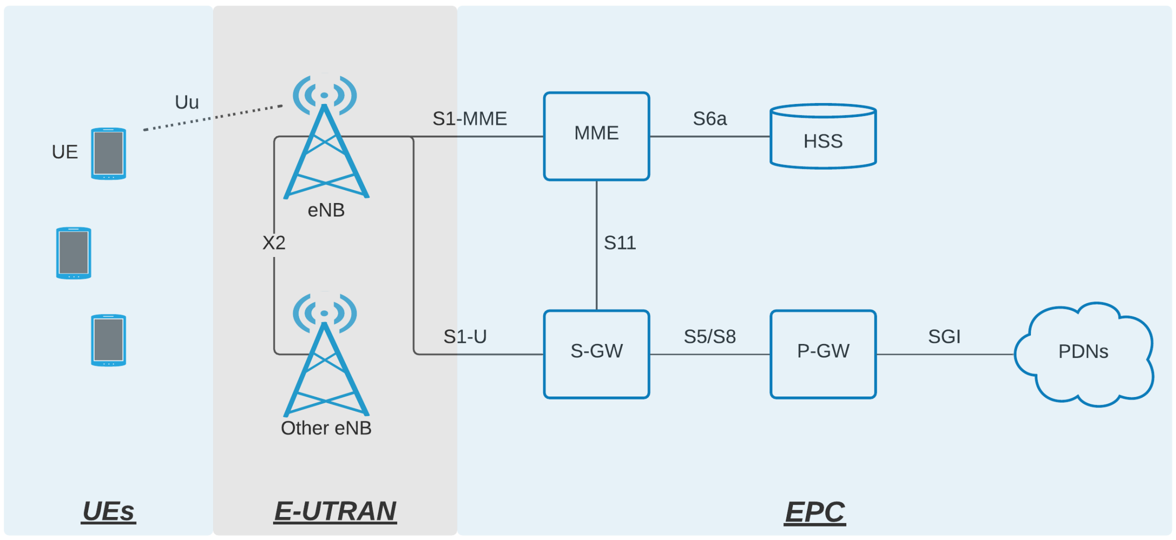


Figure 2.1: LTE High-level Architecture

### 2.2.3 LTE Architecture

The high-level architecture of the Evolved Packet System (EPS), widely known as LTE, consists of three main components as shown in Figure 2.1 [10]. The User Equipment (UE), the Evolved UMTS Terrestrial Radio Access Network (E-UTRAN) and the Evolved Packet Core (EPC).

#### User Equipment (UE)

Any device used by an end-user to communicate is called User Equipment (UE). It could be a smartphone, a laptop computer with a mobile broadband adaptor, or any other gadget.

#### Evolved UMTS Terrestrial Radio Access Network (E-UTRAN)

E-UTRAN responsibility is to handle the radio communication between UE and EPC. It consists of several base stations known as evolved Nodes B (eNB). Each eNB uses the Uu interface to manage a UE in one or many cells specified as its serving eNB. The S1 interface connects to the EPC, while the X2 interface, which employs signaling and packet forwarding during handover, may also connect to adjacent base stations.

An eNB handles UE's downlink/uplink traffic by routing user plane data to S-GW. On the controlling side, it implements radio resource management, for instance, radio bearer control, radio admission/connection mobility control, and dynamic scheduling. Furthermore, it manages low-level operations such as handovers. Finally, ciphering, packet dependable delivery, and header compression are all eNodeB functionalities.

**Evolved Packet Core (EPC)**

The EPC is the LTE network's core element. It consists of several nodes, the most important of which are MME, SGW, PGW, and HSS. They provide various services, including mobility management, authentication, session management, bearer setup, and the application of various Quality of Service levels.

- **Home Subscriber Server (HSS):** HSS is a central database that collects data on all of the subscribers of a network provider. It is one of the few LTE features that has been carried over from previous generation systems. It also has mobility management, call and session creation, user authentication, and access authorization features.

- **Packet Data Network Gateway (P-GW):** The EPC's point of interaction with the outside world is the packet data network gateway (P-GW). Each PDN gateway communicates with many external devices or packet data networks through the SGi interface, such as the network provider's servers, the internet, or the IP multimedia subsystem. An access point name (APN) corresponds to each packet data network. Usually, the network provider utilizes various APNs, one for the internet, one for the IP multimedia subsystem, and more.

- **Serving Gateway (S-GW):** The serving gateway (S-GW) is a high-level router that connects the base station to the PDN gateway. A typical network could include several serving gateways, each responsible for the mobiles in a specific geographic area. Every mobile belongs to a single serving gateway; however, if the mobile goes far enough, the serving gateway may be changed.

- **Mobility Management Entity (MME):** MME manages the mobile's high-level functioning by transmitting messages about concerns like security and data stream management unrelated to radio communications. A typical network may comprise several MMEs, each responsible for a particular geographical area, similar to the serving gateway. Each mobile corresponds to a single MME, known as the serving MME; however, this could change if the mobile travels a long distance. The MME also has control over the other network parts through internal EPC signaling messages.

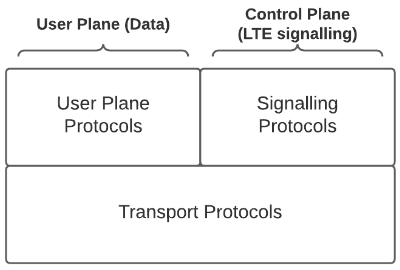


Figure 2.2: High-level Protocol Architecture

- **Policy Control and Charging Rules Function (PCRF):** The Policy Control and Charging Rules Function (PCRF) is not visible in Figure 2.1. However, it is responsible for policy control decision-making and directing the flow-based charging features in the Policy Control Enforcement Function (PCEF), placed in the P-GW.

- **Other components:** There are also components not shown in Figure 2.1 but worth mentioning. The Earthquake and Tsunami Warning System (ETWS) is one of them, and it exploits the Cell Broadcast Centre (CBC), which UMTS previously used. Another one is the Equipment Identity Registry (EIR), which contains the details of lost or stolen mobile phones, was inherited from UMTS too.

### 2.2.4 LTE Protocol Stack

There are two planes in the protocol stack as shown in Figure 2.2. The user plane protocols deal with data relevant to the user, such as data routing within a network. In contrast, the control plane protocols manage signaling messages that are primarily of interest to the network elements, such as signaling communication between two devices. Finally, data and signaling messages are sent from one place to another via the underlying transport protocols.

In LTE, there are two specific types of protocol structures: the air interface and the fixed network protocols, as demonstrated in Figure 2.3. Between the mobile and the eNB is the air interface, also known as the Uu interface. The physical layer comprises the digital and analog signal processing operations that the mobile and eNB employ to exchange data. The data connection layer, or layer 2 of the OSI model, comprises the following three protocols. The medium access control (MAC) protocol manages the physical layer at a basic level, scheduling data transactions between the mobile and the eNB. The radio link control (RLC) protocol keeps the data connection between the two devices running smoothly, for example,

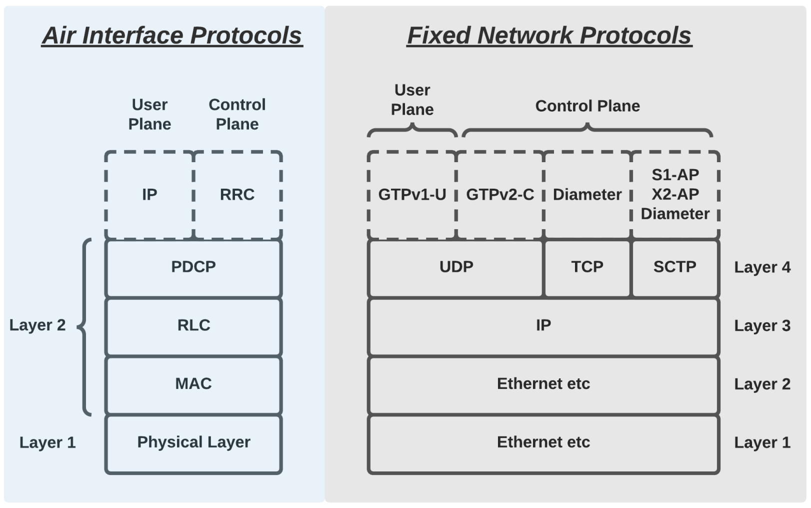


Figure 2.3: Protocols used on the air interface and by the fixed network.

by assuring dependable delivery of data streams that must arrive on time. Last but not least, the packet data convergence protocol (PDCP) performs high-level transport services such as header compression and security.

The fixed network utilizes the OSI layers to transfer information between components of the eNB and EPC. The transport network can use any appropriate protocol for the bottom two layers, for instance, the Ethernet. Then assigns an IP address to each network unit, and the internet protocol (IP) routes data from one network element to the next. A transport layer protocol exists above IP with three different protocols in use. The user datagram protocol (UDP) distributes data packets from one network device to another, whereas the transmission control protocol (TCP) re-sends packets that arrive improperly. TCP is the foundation of the stream control transmission protocol (SCTP), which has additional capabilities making it ideal for signaling message delivery. The user plane always utilizes UDP as its transport protocol for faster data delivery.

The majority of user plane interfaces use the GPRS tunneling protocol user part (GTP-U), which employs tunneling, a mechanism to forward packets from one network node to another.

LTE supports a diverse set of signaling methods. The base station regulates a mobile's radio communications via the air interface using signaling messages specified in the radio resource control (RRC) protocol. An MME uses the S1 application protocol (S1-AP) to govern the base stations within its pool area in a radio access network. Furthermore, the X2 Application Protocol (X2-AP) bridges the communication between two eNBs.

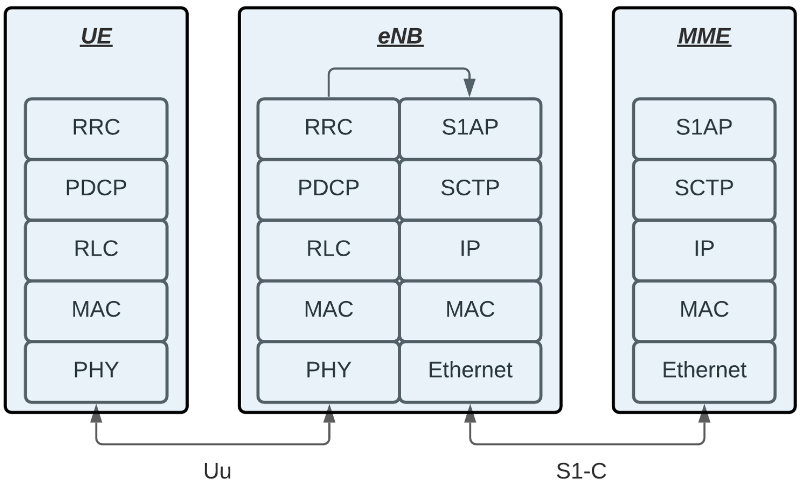


Figure 2.4: C-plane Protocol Stack

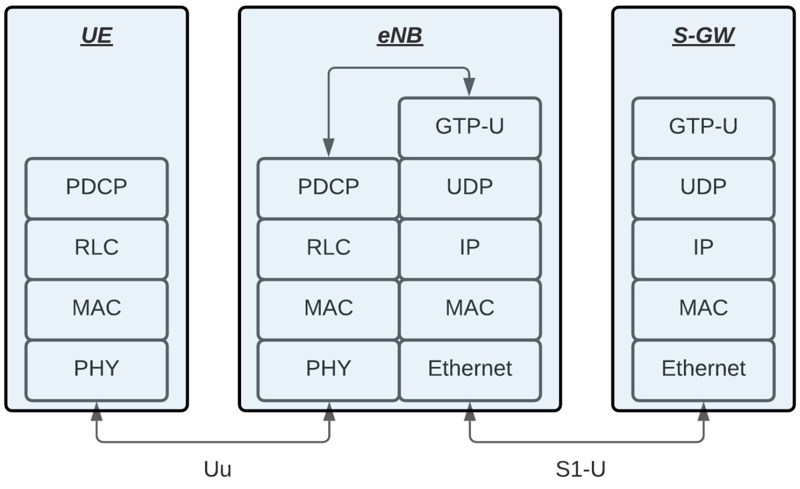


Figure 2.5: U-plane Protocol Stack

The MME uses two protocols at the non-access stratum of the air interface to manage a UE's high-level operation. These protocols are the EPS session management (ESM), which manages internal accounting inside the EPC, and the EPS mobility management (EMM), which regulates the data streams through which a mobile connects with the outside world.

The HSS and MME use the Diameter protocol to interact within the EPC for authentication, authorization, and accounting. Diameter is based on the Remote Authentication Dial In User Service (RADIUS) protocol, an earlier standard.

The majority of the other EPC interfaces employ the GPRS tunneling protocol control portion (GTP-C). This protocol specifies protocols for peer-to-peer communication between the EPC's various components and the management of the GTP-U tunnels mentioned earlier.

Figures 2.4 and 2.5 depict the protocol stack's flows for the communications between

UE/MME and between UE/S-GW accordingly.

## 2.3 5G Mobile Networks

### 2.3.1 Introduction

5G is the next significant step in mobile telecommunication systems due to the increasing demand for higher data rates and reduced latency. Data-hungry mobile devices and the growing demand for sophisticated multimedia applications such as UltraHigh Definition (UHD), 3D video, augmented reality, and immersive experience prompted the need for more resilient networks [11].

Many industry efforts have outlined eight primary needs that 5G technology must meet, and these objectives may be summarized as follows.

- 99.999% availability
- Full coverage (100%)
- 90% less energy consumption
- 1000 times more bandwidth per unit area
- 1–10 Gbps data rate for all nodes of the network
- Up to ten years battery life for low-power mode
- Supporting 10–100 times more connected devices
- 1 ms end-to-end (E2E) loop delay in the network (latency).

The ITU Radiocommunication Sector is one of the three parts of the International Telecommunication Union (ITU) and has stipulated the International Mobile Telecommunications 2020 (IMT-2020) as mobile systems in Resolution ITU-R 56-2. The IMT-2020, according to the ITU-R, will be more advanced than the IMT-2000 and IMT-Advanced. It opens up new avenues for 5G's expanded capabilities, including Enhanced Mobile Broadband (eMBB), Ultra-Reliable and Low-Latency Communications (URLLC), and substantial Machine-Type Communications (mMTC).

- **Enhanced Mobile Broadband (eMBB):** Consumers would enjoy improved speed and reliable service in future mobile networks compared to existing systems based on this use model. The eMBB comprises many scenarios, including wide-area coverage and hotspot scenarios. When considering the wide-area situation, anticipate full coverage, increased mobility, and increased data speeds. The expected data speeds will be in the gigabit per second range. On the contrary, considering the hotspot scenario, high user density and traffic capacity are needed. Still, the mobility requirements in this scenario are just for foot-traveler velocities.

- **Ultra-reliable and low-latency communications (URLLC):** For URLLC, there are various essential requirements in terms of dependability, availability, and latency; for instance, E2E latency should be less than five milliseconds. Smart grids, remote monitoring and control, vehicle-to-everything (V2X), intelligent transport systems, tactile internet applications, and other fields are among the most common URLLC application domains.
- **Massive machine-type communications (mMTC):** The mMTC framework can accommodate many devices with latency-insensitive designs, cheap costs, and extended battery life; for example, it will use millions of low-power sensors and actuators.

ITU-R defines the minimum performance requirements for 5G mobile communication networks. This report includes some key performance indicator (KPI) metrics presented in Table 2.1 for various use scenarios.

### 2.3.2 Key Technologies

5G networks combine several critical technologies to give all of the above characteristics while overcoming the limitations of previous-generation systems. It is worth discussing a few of the most essential.

- **Massive Multiple-Input Multiple-Output (Massive MIMO) Technology:** Massive MIMO is a variant of the MIMO idea that advocates for the employment of larger and more antennas in each base station (100 or more). Massive MIMO's primary goal is to expand the MIMO concept to larger scales, upgrading technologies to help 5G mobile networks achieve more stability, security, and efficiency in terms of energy and spectrum. It has a more considerable power gain resulting in a significant increase in the strength of the received signal. They also improve system throughput when base stations with massive antenna arrays serve more users.

Table 2.1: Minimum 5G technical performance requirements (ITU-R)

| **KPI** | **Model** | **Values** |
|---|---|---|
| Peak data rate | eMBB | DL: 20 Gbps, UL: 10 Gbps |
| Peak spectral efficiency | eMBB | DL: 30 bps/Hz, UL: 15 bps/Hz |
| User experienced data rate | eMBB | DL: 100 Mbps, UL: 50 Mbps (for Dense Urban case) |
| 5% user spectral efficiency | eMBB | DL: 0.3 bps/Hz, UL: 0.21 bps/Hz (for Indoor Hotspot)<br>DL: 0.225 bps/Hz, UL: 0.15 bps/Hz (for Dense Urban)<br>DL: 0.12 bps/Hz, UL: 0.045 bps/Hz (for Rural) |
| Average spectral efficiency | eMBB | DL: 9 bps/Hz/TRxP, UL: 6.75 bps/Hz/TRxP (for Indoor Hotspot)<br>DL: 7.8 bps/Hz/TRxP, UL: 5.4 bps/Hz/TRxP (for Dense Urban)<br>DL: 3.3 bps/Hz/TRxP, UL: 1.6 bps/Hz/TRxP (for Rural) |
| Area traffic capacity | eMBB | DL: 10 Mbps/m2 (for Indoor Hotspot) |
| User plane latency | eMBB, URLLC | 4 ms for eMBB and 1 ms for URLLC |
| Control plane latency | eMBB, URLLC | 20 ms |
| Connection density | mMTC | 1,000,000 devices per km2 |
| Energy efficiency | eMBB | Supporting low energy consumption capability when there is no data |
| Reliability | URLLC | 1–$10^{-5}$ success probability of transmitting a layer 2 protocol data unit of 32 bytes within 1 ms in channel quality of coverage edge |
| Mobility | eMBB | Up to 500 km/h for high-speed vehicular |
| Mobility interruption time | eMBB, URLLC | 0 ms |
| Bandwidth | eMBB | Minimum 100 MHz, up to 1 GHz for higher frequency band operation |

- **Millimeter Wave (mmWave) Systems and mmWave Massive MIMO:** The extremely high frequency (EHF) band, commonly known as the mmWave band, includes the frequency range of 30 to 300 GHz. The super-high frequency (SHF) band covers the frequency range of 3 to 30 GHz. Because the radio waves in these bands have the same propagation characteristics, the frequency range between 3 and 300 GHz s collectively referred to as mmWave bands with 1 to 100 mm wavelengths.

  The fact that mmWave frequencies have such a broad range is an advantage. The spectrum of today's cellular networks below 3 GHz is both rich and constrained. Because of the strong attenuation impact of open space in mmWave communications, they might also reuse the same frequency more often. Furthermore, the antennas in the mmWave frequency have modest physical dimensions, and the mmWave band offers a more secure and private communication medium owing to its restricted transmission space and beam widths.

- **Beamforming Techniques for 5G Mobile Communication Systems:** In mmWave networks, beamforming is a critical strategy for reducing interference and compensating for high channel attenuation. The base stations can use multiplexing to increase data rate and spatial diversity to improve durability by applying beamforming methods. There are numerous communication modes to choose from when setting up a wireless connection, including completely directional, omnidirectional, and semi-directional. Firstly, the base station and user equipment both have a directional structure in a fully directional setup. Additionally, in omnidirectional mode, both the base station and the user equipment have an omnidirectional structure. However, in semi-directional mode, one base station or user equipment has omnidirectional and the other has directional mode. Finally, analog, digital, and hybrid beamforming methods are the three types of beamforming techniques.

### 2.3.3 Functional Split Architecture

The third-generation systems introduced splitting the base station into two parts; the Remote Radio Head (RRH) utilizing all radio capabilities and the Baseband Unit (BBU) employing baseband processing functions. These parts build the so-called fronthaul network. Later, 4G networks originated the idea of Cloud-RAN, where BBUs exist in a central area,

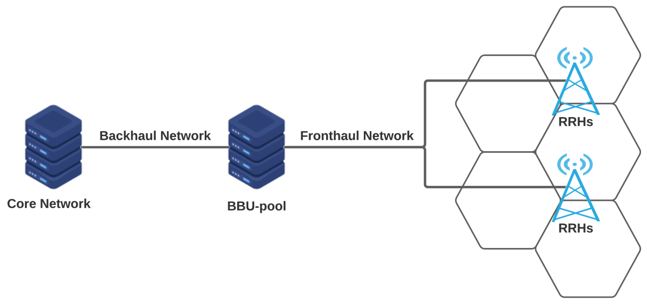


Figure 2.6: C-RAN Architecture: BBU-pool and RRHs

the BBU-pool, as shown in Figure 2.6. This way, it splits the computational processing by shifting some of it to Cloud-based virtual machines.

Cloud-RAN is now a critical part of 5G mobile networks. However, the fronthaul network's capacity requirement is massively high, leaving the opportunity for more advancements. Thus, the research focuses on finding the optimal functional splits. In other words, to indicate how many functions should remain locally at RRH and how many should exist in the cloud. Finally, several proposals [12] reached up to eight functional splits, as illustrated in Figure 2.7.

Many of these proposals, based on much research [4], are not proved beneficial. Thus, studies on real-world experimentation environments, such as Testbeds, focus primarily on PDCP/RLC and MAC/PHY split. They found that the PDCP/RLC split has the most negligible overhead and offers compatibility with various technologies, allowing for increased network capacity.

The PDCP/RLC split architecture consists of two units as shown in Figure 2.8:

- **Central Unit (CU):** It incorporates the processes of the PDCP layer and upwards.
- **Distributed Unit (DU):** It supports the processing of the RLC layer and downwards.

A single CU may manage numerous DUs, forming a one-to-many connection, but each DU has only one CU (one-to-one connection). The DUs implement numerous heterogeneous wireless network technologies including 5G, LTE, WiFi, and more. The F1AP protocol controls the communication over the new F1 interface between the CU and DU. Like the S1-U interface, data traffic is carried through the F1-U interface, wrapped with GTP headers over UDP/IP. Moreover, similar to the S1-C interface, the control plane employs the F1-C interface, which runs via SCTP/IP. Because the separation of base station functions occurs at a

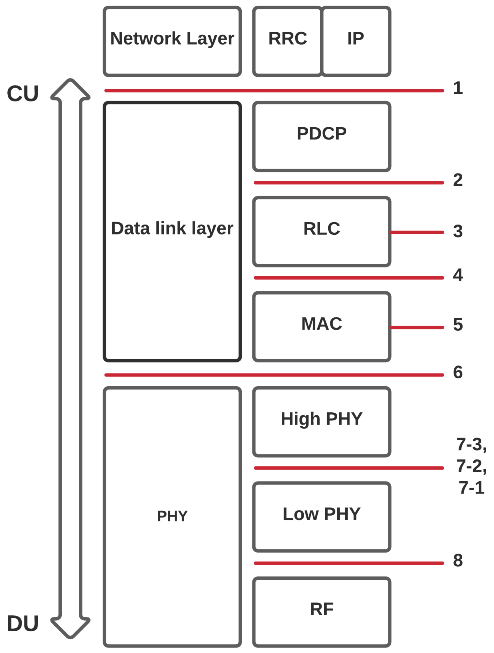


Figure 2.7: Functional Splits in LTE Protocol Stack proposed by 3GPP.

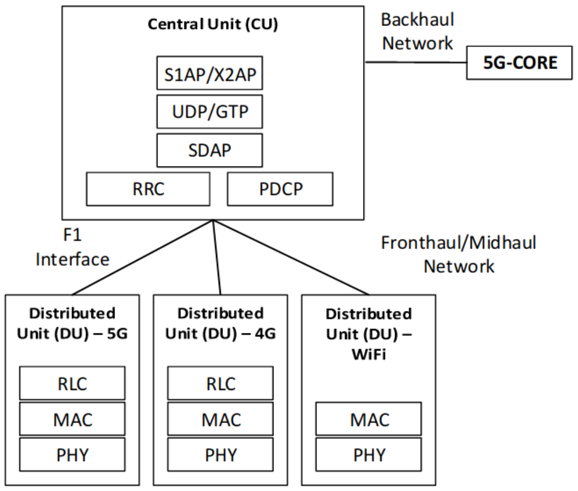

Central Unit (CU)
S1AP/X2AP
UDP/GTP
SDAP
RRC
PDCP
Backhaul Network
5G-CORE
F1 Interface
Fronthaul/Midhaul Network
Distributed Unit (DU) – 5G
RLC
MAC
PHY
Distributed Unit (DU) – 4G
RLC
MAC
PHY
Distributed Unit (DU) – WiFi
MAC
PHY

Figure 2.8: CU/DU architecture for the PDCP/RLC split

higher layer, it enables the incorporation of lower layer splits, resulting in a multi-tier disaggregated design. As such, this interface is known as the midhaul interface.

## 2.4 Future Networks Beyond 5G

### 2.4.1 Introduction

As 5G networks are being deployed, new challenges will emerge. Numerous industries are continuously digitizing, resulting in a massive increase in the data flow. For example, we foresee holographic telepresence, eHealth applications, industry 4.0 with enormous robotics, widespread 3D mobility, augmented reality (AR), and virtual reality (VR) in the future, among other things. Thus, ultra-broadband and ultra-low-latency connection, as well as more efficient wireless communication technologies, are more critical than ever.

Recently, 6G Flagship, an exceptional research program funded by the Academy of Finland, introduced the first 6G White Papers [1], [2], [3]. The research indicates that one of the primary vital missing elements in 5G networks is Machine Learning and AI. Consequently, machine learning and artificial intelligence approaches will probably be critical enablers of future 6G systems.

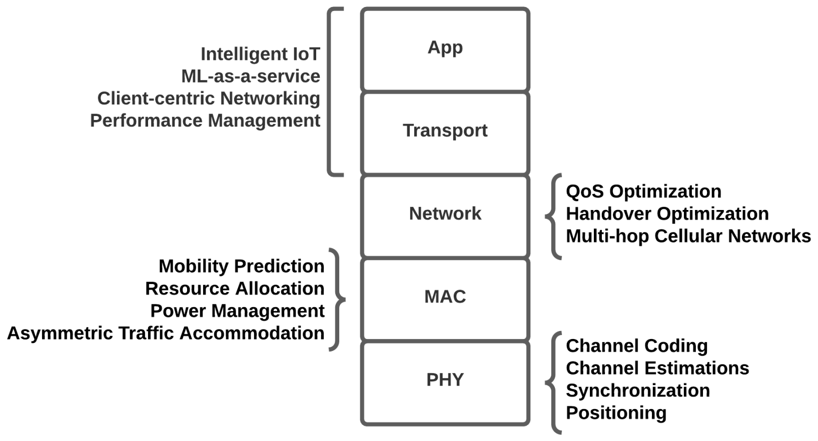


Figure 2.9: Machine Learning Applications in 6G Networks

### 2.4.2 Application of ML/AI in 6G Networks

The applications of ML/AI in the future 6G Networks will be vast, spanning the entire protocol stack, as illustrated in Figure 2.9.

- **Physical Layer** At the physical layer, channel coding combined with deep learning approaches could address various complicated problems. Moreover, auto-encoders promise to optimize performance in synchronization while ML techniques could solve non-line-of-sight (NLOS) multipaths restrictions in positioning. Finally, at channel estimation, online training neural networks could help estimate the channel's conditions accurately.

- **MAC Layer** At the MAC layer, determining users' orientations and mobility is accomplished by deploying a federated echo state network (ESN) prediction algorithm. Furthermore, predictive resource allocation will decrease the latency, enhancing machine-type communications (MTC) performance. In addition, predicting traffic could increase adaptive power-saving, and ML solutions could help alleviate the side effects of asymmetric traffic accommodation.

- **Network Layer** Adopting ML/AI algorithms will optimize the Quality of Service at the Network Layer. Also, centralized Deep Neural Networks could boost handover efficiency while various ML methods could assist in developing advanced Multi-hop cellular networks.

- **App & Transport Layer** Numerous innovative applications will come from integrating ML/AI at the Application and Transport layers, including network performance management automation, ML-aided UAV control, opportunistic data transfer in vehicular networks, ML-as-a-service, Client-centric networking, intelligent IoT, and many more.

Along with the fascinating applications that 5G and beyond technologies enable, new security issues arise. Consequently, ML/AI methods will aid in network security from an end-to-end viewpoint. To begin, it might validate the authenticity of connected peers, resulting in secure information sharing. Moreover, both defensive and attacking mechanisms should be adopted in wireless communications due to wireless medium vulnerability in interference. For instance, defending methods involve cryptography, while attack techniques require the proactive execution of an attack, such as jamming or eavesdropping, to safeguard future communications.

Edge Intelligence (EI) using ML/AI methods is another vital component of enabling 6G networks to support their efficiency, new functionalities, and new services. The devices that produce and use data are frequently positioned at the network's edge, close to the users, and monitored, surveilled, or controlled systems, producing zettabytes of data. The plethora of such resources (high density of base stations and devices in big cities) require big data handling with ML/AI approaches. These resources form a foundation for edge and cloud computing.

Edge computing (EC) is a subset of cloud computing. A portion of the data processing and storage is moved from the cloud to edge network nodes located geographically and logically close to data suppliers and end-users. In this way, it will result in performance gains, traffic enhancement, and new ultra-low latency services.

## 2.5 AI & Machine Learning

### 2.5.1 Introduction

Artificial Intelligence is a method for developing systems that replicate human reasoning, decision-making, problem-solving, and perceiving. Thousands of applications are utilizing it in industry and academics to handle unique challenges. AI is a fast-growing multipurpose technology with seemingly great potential for the future.

Machine Learning is a subfield of AI that uses data to solve problems. These algorithms

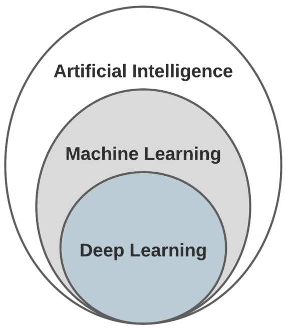


Figure 2.10: Artificial Intelligence Domains

are trained models based on probability theory and linear algebra. ML algorithms utilize our data to learn and complete prediction tasks automatically.

As shown in Figure 2.10, deep learning is a subdomain of machine learning that refers to algorithms that evaluate data the way the human brain does. Specifically, DL applications use a layered structure of algorithms known as artificial neural networks (ANNs). ANNs implement a group of so-called artificial neurons inspired by the biological neural network of the human mind.

There are three fundamental branches of Machine Learning; supervised, unsupervised, and reinforcement learning. Supervised learning involves learning a function that translates input to an output based on examples labeled training data. The validation of the algorithm comes from the generalization error. On the other hand, unsupervised learning algorithms do not need labels and seek patterns on their own. Moreover, reinforcement learning is the ML branch associated with intelligent agents taking actions in an environment. It does not require labeled data since the agents obtain knowledge through exploration and maximizing a reward function. Finally, these branches also have combinations, such as semi-supervised learning and partially reinforcement learning algorithms.

## 2.5.2 Supervised Learning

As previously stated, supervised learning is concerned with acquiring knowledge via the use of labeled data. It consists of two types; regression and classification. The regression objective is to approximate a mapping function between continuous input variables and a continuous output variable (a continuous quantity). On the contrary, classification is the task of predicting a discrete output variable (discrete class label).

Our primary focus in this study is on supervised regression approaches. Numerous so-

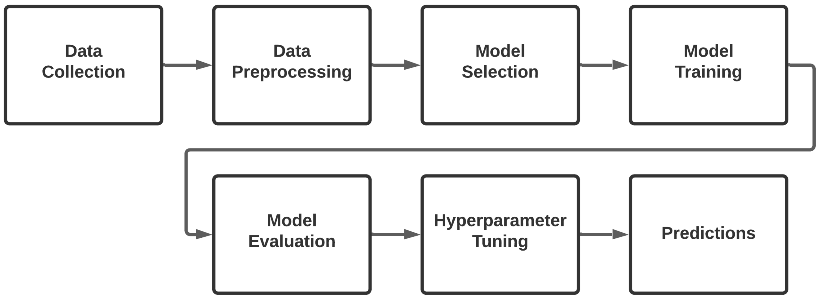


Figure 2.11: Machine Learning Pipeline from data collection to prediction

phisticated algorithms are in use, each of which stands on a different design. Linear models, such as Linear Regression, assume a linear relationship between the independent variables. There are also models supporting non-linear relationships; for instance, Support Vector Machines rely on high dimensional feature space, the hyperplane. Furthermore, as Decision Tree, tree-based models are very robust, utilizing thresholds-driven decisions. Finally, ensemble techniques, for example, XGBoost, employ multiple learning models leading to enhanced performance.

Deep learning algorithms, on the other hand, have a great deal to contribute. Multilayer perceptron (MLP) is an Artificial Neural Network utilizing multiple layers of neurons and non-linear activation functions. Another significant model that critically impacts Time Series Forecasting problems is the long short-term memory (LSTM) recurrent neural network. Its huge advantage is processing entire time sequences of data by employing a cell, an input gate, an output gate, and a forget gate. Moreover, Convolutional Neural Networks (CNNs) have dominated computer vision due to their various building ingredients, including convolutional layers, pooling layers, and fully connected layers. Of course, hybrid models such as CNN-LSTMs dominate today, providing the best of both worlds.

### 2.5.3 Machine Learning Steps

Numerous distinct steps exist in order to construct an effective prediction algorithm. As illustrated in Figure 2.11, the whole procedure consists of data collection, data preprocessing, model selection, model training, model evaluation, hyperparameter tuning, and finally, predictions. It is worth delving a little more into each of them.

- **Data Collection:** Data collection is the technique of collecting and analyzing data on selected variables. This data rely on the corresponding predicting problem, and they are, at this point, in a raw format.

- **Data Preprocessing:** Preprocessing data is a critical step that involves converting raw data to a comprehensible format before use, improving efficiency. This step may include cleaning missing or noisy data, normalization, standardization, dimensionality reduction, and more.

- **Model Selection:** This step determines the most relevant model for the problem at hand. As discussed earlier, there is a wide variety of machine learning techniques. However, appropriate model selection plays an essential role in performance.

- **Model Training:** Here the model is trained on a specific subset of the available data, called the training set. The model adapts to the data by modifying its internal state with various techniques, such as minimizing a cost function, using the Gradient Descent approach.

- **Model Evaluation:** Model evaluation is an essential process in concluding whether the model performs well or not. It relies on data splitting techniques such as training/test/validation split or, a more sophisticated one, k-Fold Cross-Validation. Moreover, validating a regression model includes studying many error metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE).

- **Hyperparameter Tuning** Every model accepts a set of pre-defined arguments, called hyperparameters. Hyperparameter tuning is the method of searching and carefully configuring the appropriate model hyperparameters to enhance overall performance.

- **Predictions** Finally, after all the steps mentioned above, the model is ready to be applied and make predictions.

## 2.6 Experimental Tools and Methods

This section analyzes all the resources and tools used for the experiments. It consists of the NITOS wireless testbed [13] [14], located at the University of Thessaly, Greece, the OpenAirInterface platform [15] used to develop real cellular network components, and the FlexRAN platform [16] utilized to implement scheduling of clients.

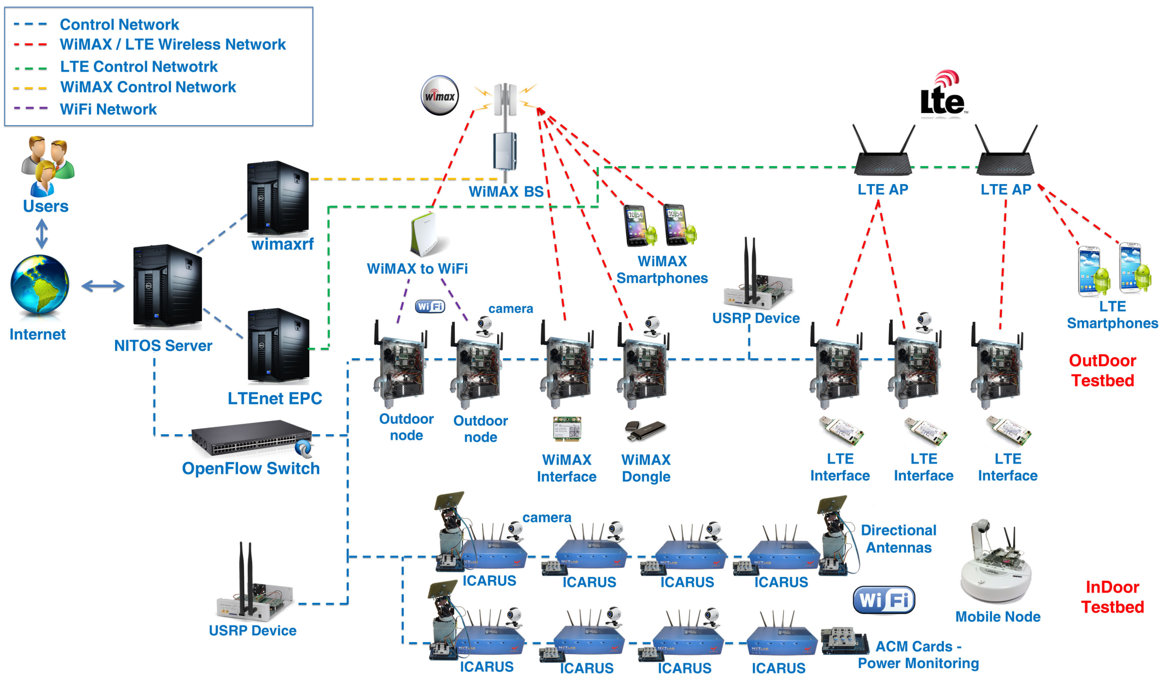


Figure 2.12: Overall NITOS Architecture

### 2.6.1 NITOS Testbed

Since 2007, the NITLAB unit at the University of Thessaly has run the Network Implementation Testbed utilizing Open Source platforms (NITOS), evaluating cutting-edge networking research. The NITOS testbed is one of the most extensive open experimental departments in Europe, allowing users globally to access highly programmable and remotely accessible equipment using the cOntrol and Management Framework (OMF) open-source software, as shown in Figure 2.12. The testbed is integrated into more prominent resource federations, such as OneLab and Fed4FIRE, supporting more heterogeneous resource testings. It consists of three separate deployments: the Indoor RF Isolated Testbed, the Outdoor Testbed, and the Office Testbed, as illustrated in Figures 2.13, 2.14 and 2.15.

Numerous critical and cutting-edge components comprise the NITOS testbed:

- The **wireless experimentation testbed** incorporates 100 well-equipped nodes, featuring multiple wireless interfaces and allowing experimentation with heterogeneous wireless technologies, such as Wi-Fi, WiMAX, LTE, Bluetooth.

- The **Cloud infrastructure** comprises seven HP blade servers and two rack-mounted servers with a combined volume of 272 CPU cores, 800 GB of RAM, and 22 TB of storage. The network is connected using an HP 5400 series modular Openflow switch,

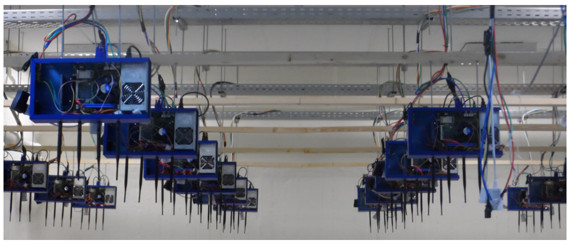


Figure 2.13: Indoor RF Isolated Testbed

Figure 2.14: Outdoor Testbed

Figure 2.15: Office Testbed

supporting 10Gb Ethernet communication between cluster modules and 1Gb between the cluster and GEANT.

- The **wireless sensor network testbed** consists of three components: a manageable testbed installed in UTH's buildings, a city-scale sensor network installed in Volos, and city-scale mobility sensing infrastructure based on volunteer users' bicycles. All sensor platforms are unique and built by UTH; they run on Arduino firmware and communicate via various wireless methods (ZigBee, Wi-Fi, LTE, Bluetooth, IR).

- The **Software Defined Radio (SDR) testbed** is made up of Universal Software Radio Peripheral (USRP) devices connected to the NITOS wireless nodes via USB. USRPs enable the researcher to program numerous physical layer characteristics (for example, modulation), enabling devoted PHY layer or cross-layer study.

- The **Software-Defined Networking (SDN) testbed** comprises numerous switches attached with OpenFlow technology to the NITOS nodes, allowing for experimentation with switching and routing networking protocols. Experiments utilizing OpenFlow technology can be integrated with wireless networking technologies, allowing for more diverse experimental scenarios.

### 2.6.2 The OpenAirInterface Platform

The OpenAirInterface (OAI) wireless technology platform is the first open-source software-based implementation of the LTE system, including the whole protocol stack defined by 3GPP specs. It incorporates developments to the Evolved Packet Core and E-UTRAN. Thus, researchers implement base stations and core networks and attach real UEs, building a complete 4G and beyond infrastructure. It is ideal for cutting-edge experimentation on novel 4G/5G and beyond configurations validating theoretical models and algorithms.

OAI is written in standard C and is distributed as free software under the GNU General Public License provisions(GPLv3). OAI provides a thorough development environment with built-in tools, including realistic emulation, soft monitoring/debugging instruments, protocol analyzer, performance profiler, and a configurable logging system for all layers and channels.

This thesis implements a realistic experimentation environment on the NITOS Testbed using OAI. Additionally, we contribute to the development of OAI's code by incorporating unique methods and methodologies on wireless clients' scheduling described in the following sections.

### 2.6.3 The FlexRAN Platform

FlexRAN is a resilient and programmable open-source implementation of a Software-Defined Radio Access Networks. It comprises two primary components: the FlexRAN Service and Control Plane and the FlexRAN Application Plane.

The FlexRAN service and control plane is hierarchical, consisting of a Real-time Controller (RTC) coupled to several underlying RAN runtimes, one for each RAN module; for instance, one for monolithic 4G eNB, or multiple for a disaggregated 4G and 5G.The RAN runtime environment is an abstraction layer between the RAN module and the RTC and control applications, separating the control and data planes.

FlexRAN protocol enables the communication between the real-time controller and the runtime environment's RAN agent. RAN control applications may exist on top of the RAN runtime or the RTC SDK, enabling the monitoring, control, and coordination of RAN infrastructure. These applications can range from simple real-time monitoring to more complex distributed apps that affect the status of the RAN in real-time, such as a MAC scheduler.

We utilize FlexRAN for the efficient scheduling of wireless clients. Specifically, FlexRAN enables the slicing of base station resources between the clients (UEs). We exploit this technique to schedule clients among different wireless technologies, such as LTE, WiFi, 5G, and more. A FlexRAN slice could include various parameters such as:

- **dl:** A list of configuration items for downlink slices.
- **id:** The slice's unique identifier.
- **percentage:** The number of resource blocks that this slice may utilize, as a percentage of the total bandwidth
- **maxmcs:** The maximum MCS that this slice is permitted to employ.

# Chapter 3

# Framework for Scheduling among Heterogeneous DUs

## 3.1 Introduction

In 5G and beyond networks, efficient scheduling between different wireless technologies is critical. Thus, in the functional split architecture described in section 2.3.3, the CU unit should act as a coordinator by balancing the traffic load among the various DUs. This chapter constructs and evaluates this advanced architecture under realistic experimental situations.

Currently, there are many experimental disaggregated architectures for heterogeneous RANs. Especially, a work [4] has proposed one equipped with both LTE and WiFi DUs under the guidance of a single CU. They build the experimental environment using NITOS Testbed described in section 2.6.1, incorporating the OpenAirInterface for the cellular topology implementation detailed in section 2.6.2.

Our target is to build on top of this work, providing a novel scheduling scheme that utilizes the Flexran Platform illustrated in section 2.6.3. In the following sections of this chapter, we describe, validate and evaluate the framework under the same experimental circumstances.

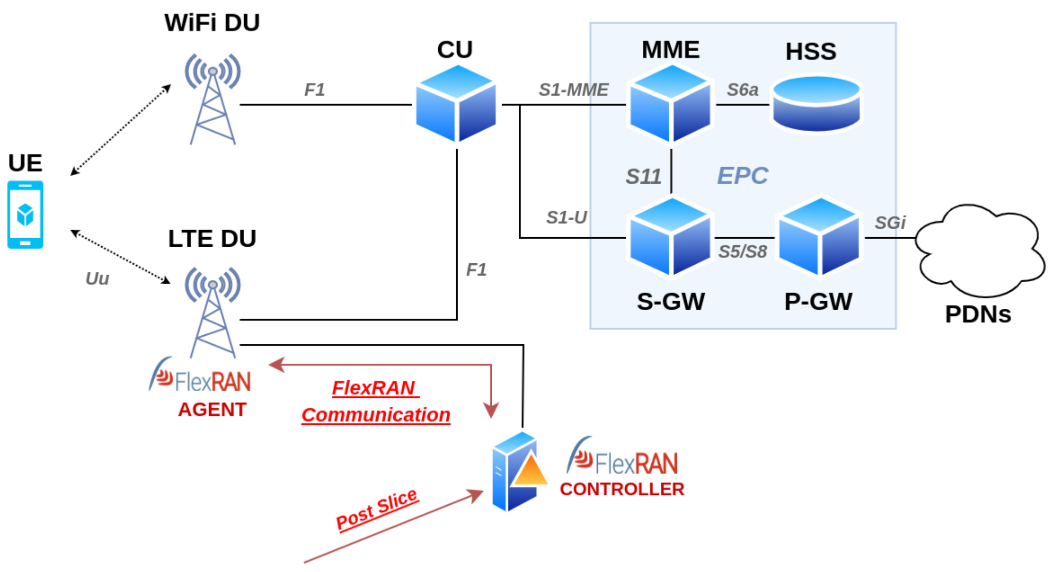


Figure 3.1: Heterogeneous and Disaggregated RAN Architecture incorporating FlexRAN

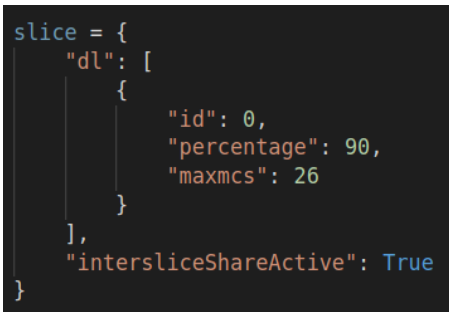
## 3.2 Framework

### 3.2.1 Initial Topology

First of all, the fundamental topology used is illustrated in Figure 3.1. Several components are essential, beginning from the core network (EPC) that connects the RAN architecture with the outer world. Moreover, following the functional split architecture, a single CU unit is connected with one LTE DU and one WiFi DU providing heterogeneous connectivity to a single UE. Last but not least, the FlexRAN controller is an essential instrument communicating with the FlexRAN Agent located in the LTE DU to provide LTE traffic slicing. In this way, the slicing control of the LTE interface is handled with a single slice post to the FlexRAN controller, as shown in Figure 3.1 with the red arrows. For instance, a slice with a percentage value equal to 50 percent will force the LTE DU to provide half of the resource blocks (RBs) available to the specific UE. Consequently, if all RBs were used before the slice post, the link's capacity and bandwidth will now drop to half. An example of a FlexRAN slice is illustrated in Figure 3.2.

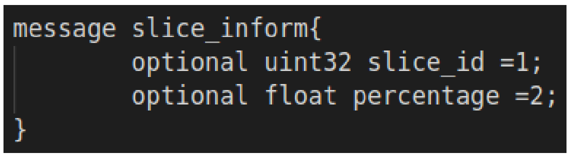
### 3.2.2 Proposed Approach

The proposed approach is heavily based on FlexRAN slicing implementation. The primary goal is the usage of the slice by the CU to schedule traffic based on its percentage value.



```
slice = {
    "dl": [
        {
            "id": 0,
            "percentage": 90,
            "maxmcs": 26
        }
    ],
    "intersliceShareActive": True
}
```

Figure 3.2: A Downlink FlexRAN Slice Example



```
message slice_inform{
        optional uint32 slice_id =1;
        optional float percentage =2;
}
```

Figure 3.3: The Message Format with the Slicing Values to be sent via F1 Interface.

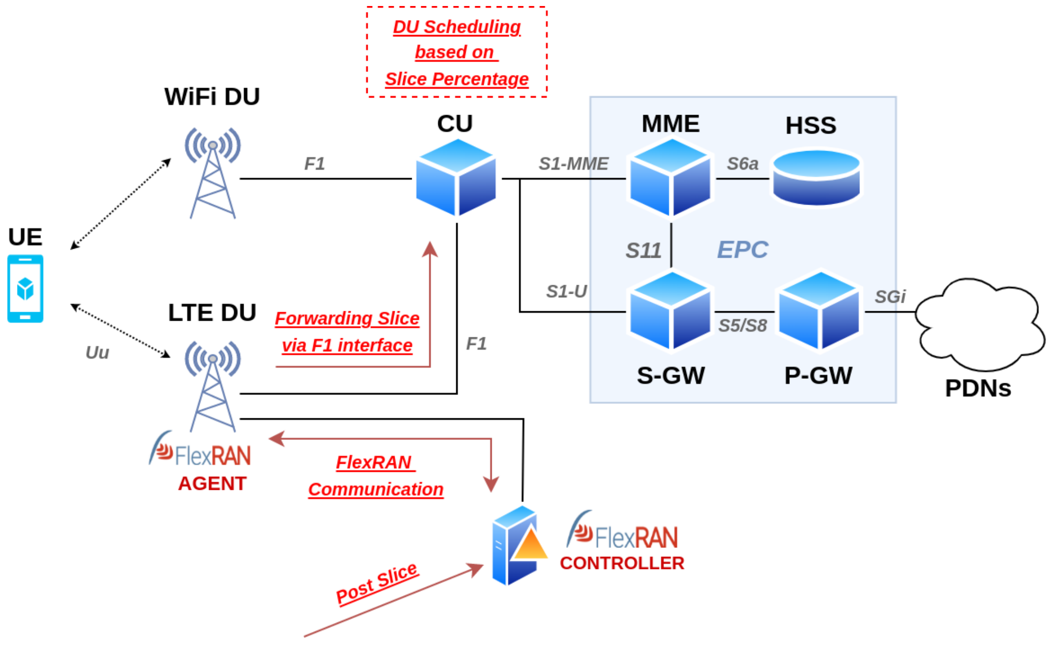


Figure 3.4: Proposed Approach for Scheduling between WiFi and LTE DUs

For this reason, when the slice arrives at the FlexRAN agent, it should be sent directly to the CU, as shown in Figure 3.4.

The full implementation of this communication flow resides on a struct data structure. When the posted slice arrives at the FlexRAN Agent from the Controller, we retrieve it at the MAC layer and save the percentage and id values in a struct. Then, as illustrated in Figure 3.3, we compose a message utilizing these values and forward it to the CU via the F1 interface.

The PDCP layer receives the message on the CU side and stores the values in its struct. Now, the percentage item assists in implementing a scheduling scheme between the DUs in a Round-robin fashion. Precisely, there are one hundred slots available for every round, each one representing one percent. Every DU exploits a slot to transmit a packet to UE. The LTE

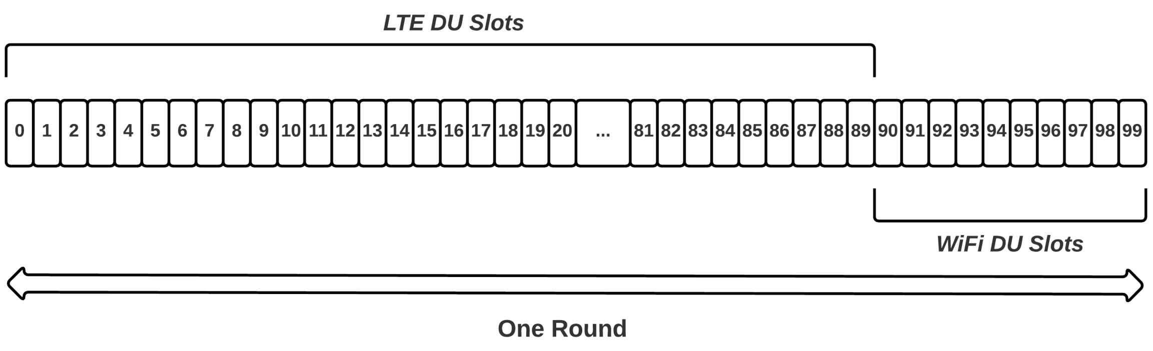


Figure 3.5: PDCP Scheduling Technique for LTE and WiFi DUs

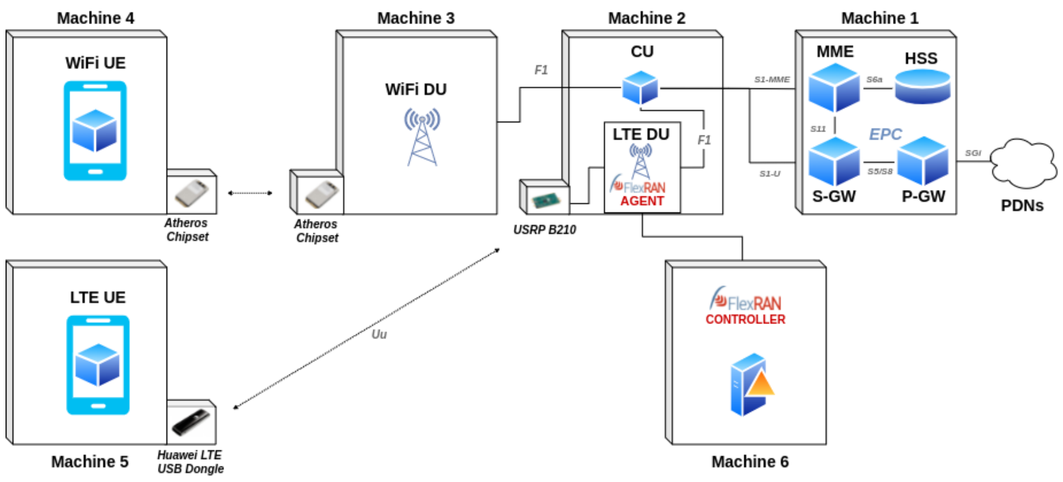


Figure 3.6: Experimental Setup with NITOS Testbed.

DU assigns its slots first, followed by the WiFi DU; for example, if the posted percentage in slice equals 90 percent, the LTE DU will allocate the first 90 slots, while the WiFi DU will employ the remaining 10, as shown in Figure 3.5.

## 3.3 Experiments

### 3.3.1 Experimental Setup

The experiments assist in validating and evaluating the proposed approach in realistic circumstances. Specifically, we use the NITOS Testbed for our experimental topology, depicted in Figure 3.6.

The system architecture employs six NITOS machine nodes. First, one node deploys the core network, including the HSS, MME, and SPGW components. Secondly, a node equipped with a USRP B210 device sets up both CU and LTE DU. We use the LTE implementation of the OpenAirInterface platform, due to its stability and efficiency, compared to the 5GNR implementation at the time of writing. Nevertheless, the solution can be directly projected to the 5G implementation by interchanging the core network components with the 5G ones (HSS/UDM, MME/AMF, SPGW-U/UPF, SPGW-C/SMF) and the disaggregated eNB with a disaggregated gNB. Moreover, there is a WIFI DU node and two nodes implementing the UE; one for the WiFi interface equipped with a Qualcomm Atheros AR9380 chipset and one for the LTE supplied with a Huawei LTE USB Dongle. Finally, we utilize a node for hosting the FlexRAN Controller.

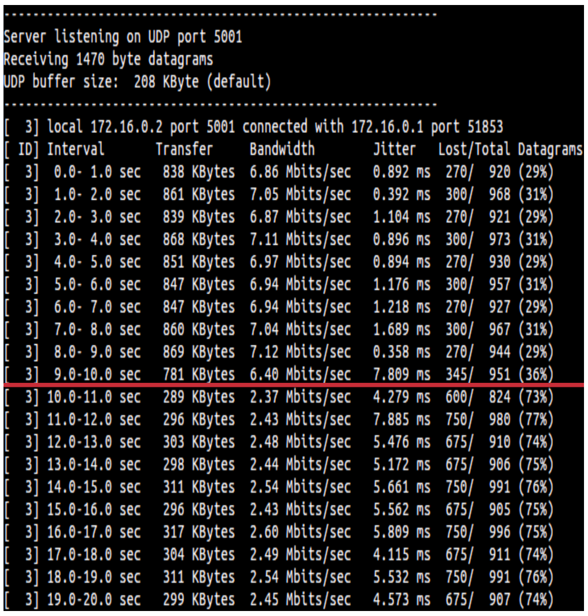
### 3.3.2 Experimental Scenario

The testing scenario is constructed to demonstrate the efficient scheduling between the DUs in real-time. Specifically, we operate the iperf tool for traffic injection and live network performance monitoring. The goal is to send a 10 Mbps downlink traffic from the core network to the UE and observe the differences when posting new slices in FlexRAN Controller.

Initially, the percentage value in the slice is 70 percent, meaning that 70% of the traffic will be scheduled to the LTE UE and the other 30% to the WiFi. At 10 seconds approximately, we post another slice with a percentage value of 25 percent, replacing the previous one. We want to observe the traffic at the LTE interface dropping from 70% to 25% and at WiFi increasing from 30% to 75%.

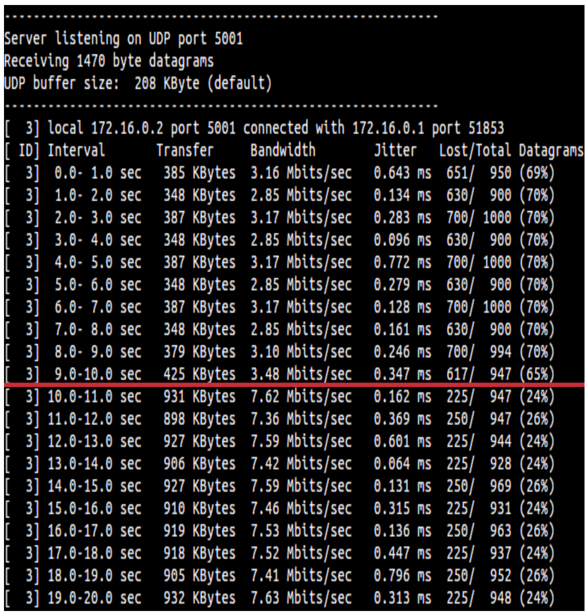
### 3.3.3 Experimental Results

The experimental results come to validate the framework giving the expected outcome. In the beginning, as illustrated in the iperf server (UE) logs in Figures 3.7 and 3.8, the total 10 Mbps traffic is scheduled as expected based on the first slice (with the 70 percent value); that is 7 Mbps (70%) LTE and 3 Mbps (30%) WiFi traffic. At 10 seconds, the red horizontal line represents the post of the new slice with a percentage value of 25 percent. Then, after this time threshold, the LTE bandwidth is dropped to 2.5 Mbps (25%) while the WiFi one increases to 7.5 Mbps (75%), as desired. Figure 3.9 depicts more clearly this scheduling transition.



```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 172.16.0.2 port 5001 connected with 172.16.0.1 port 51853
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3]  0.0- 1.0 sec   838 KBytes  6.86 Mbits/sec   0.892 ms  270/  920 (29%)
[  3]  1.0- 2.0 sec   861 KBytes  7.05 Mbits/sec   0.392 ms  300/  968 (31%)
[  3]  2.0- 3.0 sec   839 KBytes  6.87 Mbits/sec   1.104 ms  270/  921 (29%)
[  3]  3.0- 4.0 sec   868 KBytes  7.11 Mbits/sec   0.896 ms  300/  973 (31%)
[  3]  4.0- 5.0 sec   851 KBytes  6.97 Mbits/sec   0.894 ms  270/  930 (29%)
[  3]  5.0- 6.0 sec   847 KBytes  6.94 Mbits/sec   1.176 ms  300/  957 (31%)
[  3]  6.0- 7.0 sec   847 KBytes  6.94 Mbits/sec   1.218 ms  270/  927 (29%)
[  3]  7.0- 8.0 sec   860 KBytes  7.04 Mbits/sec   1.689 ms  300/  967 (31%)
[  3]  8.0- 9.0 sec   869 KBytes  7.12 Mbits/sec   0.358 ms  270/  944 (29%)
[  3]  9.0-10.0 sec   781 KBytes  6.40 Mbits/sec   7.809 ms  345/  951 (36%)
[  3] 10.0-11.0 sec   289 KBytes  2.37 Mbits/sec   4.279 ms  600/  824 (73%)
[  3] 11.0-12.0 sec   296 KBytes  2.43 Mbits/sec   7.885 ms  750/  980 (77%)
[  3] 12.0-13.0 sec   303 KBytes  2.48 Mbits/sec   5.476 ms  675/  910 (74%)
[  3] 13.0-14.0 sec   298 KBytes  2.44 Mbits/sec   5.172 ms  675/  906 (75%)
[  3] 14.0-15.0 sec   311 KBytes  2.54 Mbits/sec   5.661 ms  750/  991 (76%)
[  3] 15.0-16.0 sec   296 KBytes  2.43 Mbits/sec   5.562 ms  675/  905 (75%)
[  3] 16.0-17.0 sec   317 KBytes  2.60 Mbits/sec   5.809 ms  750/  996 (75%)
[  3] 17.0-18.0 sec   304 KBytes  2.49 Mbits/sec   4.115 ms  675/  911 (74%)
[  3] 18.0-19.0 sec   311 KBytes  2.54 Mbits/sec   5.532 ms  750/  991 (76%)
[  3] 19.0-20.0 sec   299 KBytes  2.45 Mbits/sec   4.573 ms  675/  907 (74%)
```

Figure 3.7: UE's LTE Interface



```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 172.16.0.2 port 5001 connected with 172.16.0.1 port 51853
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3]  0.0- 1.0 sec   385 KBytes  3.16 Mbits/sec   0.643 ms  651/  950 (69%)
[  3]  1.0- 2.0 sec   348 KBytes  2.85 Mbits/sec   0.134 ms  630/  900 (70%)
[  3]  2.0- 3.0 sec   387 KBytes  3.17 Mbits/sec   0.283 ms  700/ 1000 (70%)
[  3]  3.0- 4.0 sec   348 KBytes  2.85 Mbits/sec   0.096 ms  630/  900 (70%)
[  3]  4.0- 5.0 sec   387 KBytes  3.17 Mbits/sec   0.772 ms  700/ 1000 (70%)
[  3]  5.0- 6.0 sec   348 KBytes  2.85 Mbits/sec   0.279 ms  630/  900 (70%)
[  3]  6.0- 7.0 sec   387 KBytes  3.17 Mbits/sec   0.128 ms  700/ 1000 (70%)
[  3]  7.0- 8.0 sec   348 KBytes  2.85 Mbits/sec   0.161 ms  630/  900 (70%)
[  3]  8.0- 9.0 sec   379 KBytes  3.10 Mbits/sec   0.246 ms  700/  994 (70%)
[  3]  9.0-10.0 sec   425 KBytes  3.48 Mbits/sec   0.347 ms  617/  947 (65%)
[  3] 10.0-11.0 sec   931 KBytes  7.62 Mbits/sec   0.162 ms  225/  947 (24%)
[  3] 11.0-12.0 sec   898 KBytes  7.36 Mbits/sec   0.369 ms  250/  947 (26%)
[  3] 12.0-13.0 sec   927 KBytes  7.59 Mbits/sec   0.601 ms  225/  944 (24%)
[  3] 13.0-14.0 sec   906 KBytes  7.42 Mbits/sec   0.064 ms  225/  928 (24%)
[  3] 14.0-15.0 sec   927 KBytes  7.59 Mbits/sec   0.131 ms  250/  969 (26%)
[  3] 15.0-16.0 sec   910 KBytes  7.46 Mbits/sec   0.315 ms  225/  931 (24%)
[  3] 16.0-17.0 sec   919 KBytes  7.53 Mbits/sec   0.136 ms  250/  963 (26%)
[  3] 17.0-18.0 sec   918 KBytes  7.52 Mbits/sec   0.447 ms  225/  937 (24%)
[  3] 18.0-19.0 sec   905 KBytes  7.41 Mbits/sec   0.796 ms  250/  952 (26%)
[  3] 19.0-20.0 sec   932 KBytes  7.63 Mbits/sec   0.313 ms  225/  948 (24%)
```

Figure 3.8: UE's WiFi Interface

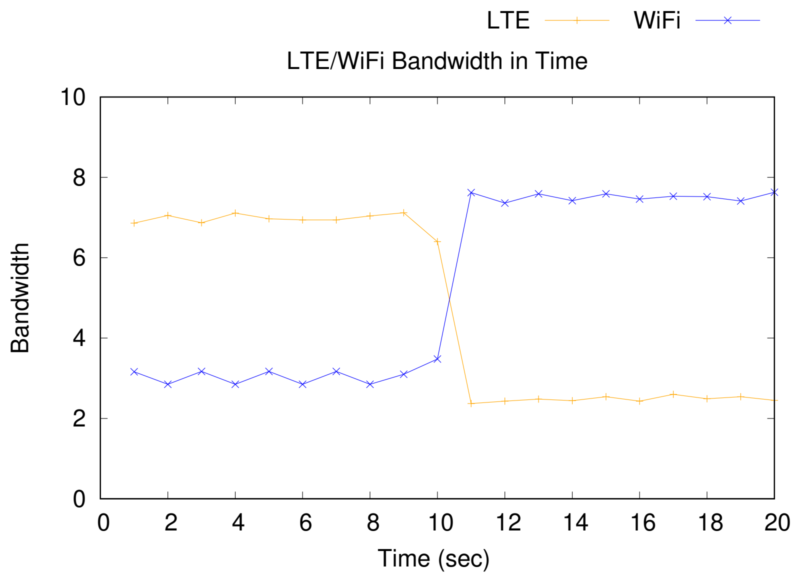


Figure 3.9: LTE/WiFi Bandwidth Monitoring with FlexRAN Slice Reconfiguration

# Chapter 4

# AI-Driven Real-time Scheduling between Heterogeneous DUs

## 4.1 Introduction

Future generation networks set the urgent need for ML and AI applications and methods in every network stack component, as described in section 2.4.2 [1] [2] [3]. As a result, the whole infrastructure will become more flexible, robust, and efficient in handling the massive traffic data while providing minimum latency.

This chapter proposes, validates, and evaluates an AI-Driven framework for scheduling between heterogeneous DUs in 5G disaggregated RAN architectures. We build upon the work of the previous chapter by implementing ML and AI techniques for scheduling the DUs based on the LTE channel quality. Specifically, after continuous LTE link monitoring, the proposed scheme makes scheduling decisions to ensure optimal network conditions and QoE.

The monitoring focuses on the Channel Quality Indicator (CQI). In LTE systems, the CQI is a metric that provides the channel quality to the base station showing the level of modulation and coding the UE could function. Especially, it ranges from 0 to 15 in values. That means from no to 64QAM 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 TBS bits.

The following sections utilize CQI data of car routes from real commercial networks and reproduce them under a controllable and realistic experimental environment. Additionally, we build an ML & AI unit by analyzing the data and evaluating various machine and deep learning models. This unit monitors and forecasts the LTE CQI values continuously in real-

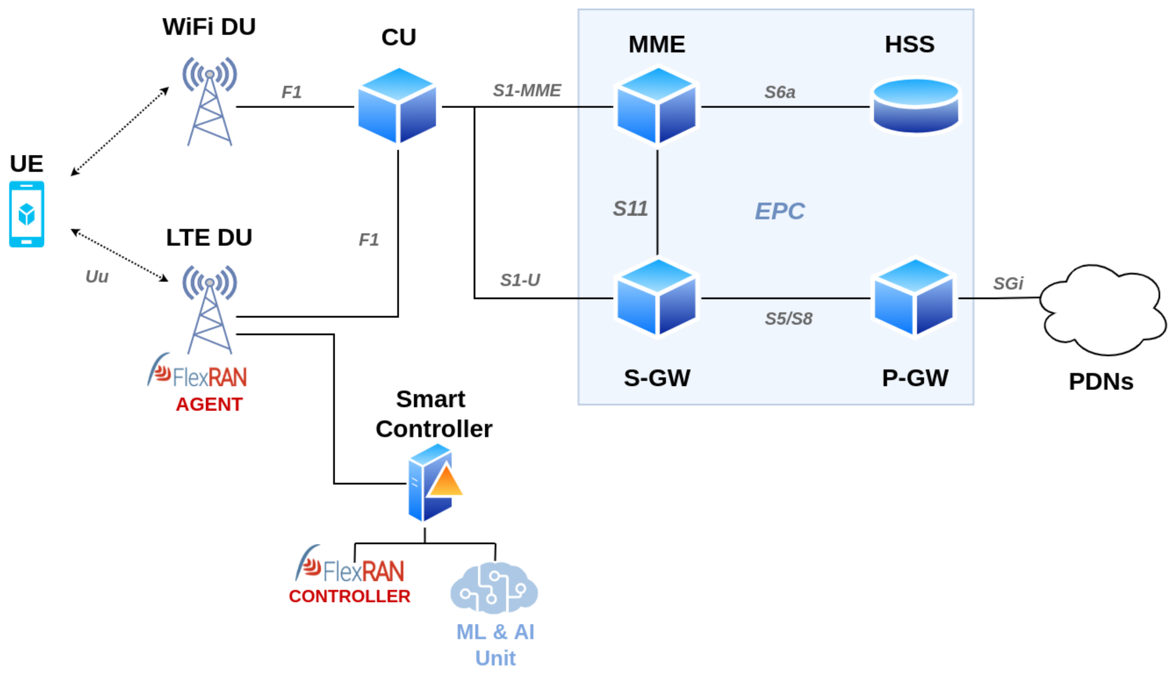


Figure 4.1: Overall System Architecture incorporating Real-time Scheduling with the ML & AI Unit

time. Subsequently, it posts the appropriate slices to the FlexRAN controller based on a slice allocation algorithm to enhance resource management and provide optimal network performance and user experience. In the end, we validate and evaluate the whole framework under the NITOS Testbed in terms of efficiency, QoS, and QoE.

## 4.2 System Architecture

The overall baseline system architecture is depicted in figure 4.1, presenting the incorporation of the ML & AI unit in the RAN disaggregated network of the previous chapter. The main idea is to employ a monitoring scheme on the side of the Flexran controller to sniff the UE's CQI values continuously. Then, machine learning techniques analyze the patterns in the data predicting future CQI values in real-time. This way, the ML & AI unit sends new and more appropriate scheduling configurations (slices) to the FlexRAN controller. Subsequently, the slices arrive at the CU utilizing our communication implementation between the FlexRAN agent and the CU discussed in the previous chapter. Then, the CU continuously reconfigures the DU scheduling among the heterogeneous DUs based on the slice that came from the ML & AI unit, enhancing the overall network performance resulting in a better QoS

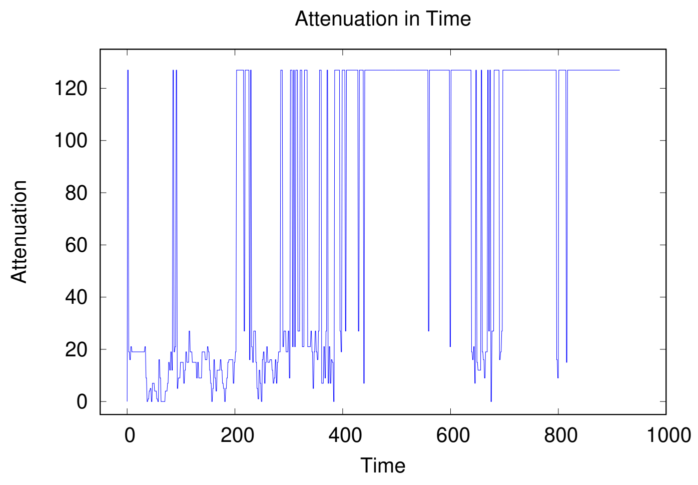


Figure 4.2: Attenuation Data Available from Real Commercial Networks

and QoE.

## 4.3 Data

In this section, we describe the methodology followed from acquiring the data to preparing them for the models. Precisely, we begin by describing the attenuation data used to simulate the car routes. Then, we present the data augmentation step where we enrich the attenuation data to obtain many different car route patterns. Finally, we detail the CQI data collection and pre-processing steps.

### 4.3.1 Attenuation Data

The goal of this work is to study CQI data of real car routes in city Volos. The CQI values vary across different geographical areas of the city as the car is moving. Specifically, it is high in areas with good link conditions, while it drops in regions with poor LTE coverage.

In order to emulate the car routes inside our experimental environment, we need to give the illusion of a moving machine node. To achieve this, we install programmable attenuators on the USRP's outputs. Thus, we control the experienced CQI by modifying the USRP attenuation, giving the sense of motion. In this way, we adjust the CQI values based on our collected data from real car routes in commercial networks.

We possess some attenuation data from real vehicle pathways in Volos, Thessaly, Greece, where our university and testbed are. Figure 4.2 depicts data from a long pathway with approximately one thousand attenuation values. In the beginning, the attenuation values are sufficiently low, corresponding to high CQI, and after the 400th time unit, the attenuation rises to 120, giving low CQI.

### 4.3.2 Designing Basic Attenuation Scenario

This section constructs a basic attenuation scenario based on the available data. It will represent an actual car route, and we will use it in our analysis from now on. Our goal is to demonstrate the ability of our framework to adapt to extreme CQI transitions. In detail, we need a case where the CQI drops from high to low values and also an opposite situation where it rises from low to high values.

As already has been stated, our available attenuation data include the first case where the CQI is initially high and later drops to low values. Now, in order to create a larger scenario that also contains the second case, we use a copy of the available attenuation data. Precisely, we obtain a copy of them, reverse it and append it at the end. This way, we simulate a car turn-around at the end of the pathway to traverse it in the opposite direction. Since the geographical area is the same and the direction is the opposite, we expect the attenuation data for this part to be the same or very similar values as the first part but in a reversed sequence.

Now that we built a larger scenario with both cases, we need to ensure that our model has enough time to prepare and start predicting before the first CQI transition phase in our experiment. Specifically, this phase is approximately at the 400th time unit, and at this point, we could not be sure whether this window is large enough for our models. Thus, it is a good idea to extend it.

We could achieve that by inserting a car-stop at a red traffic light any time inside this window. A regular traffic light cycle is 120 seconds, meaning that the longest waiting time at a red light is 90-120 seconds. So, we assume that the car waits for approximately two minutes at the red light before it continues. To illustrate this with attenuation values, we insert 120 constant values. Understandably, when the car is immovable, it stays in the same geographical area, and as a result, it has a constant attenuation and CQI value.

Finally, after experimentation with these attenuation data in the testbed, we conclude that it is necessary to rescale the values in the range 0-15. Thus, Figure 4.3 illustrates the basic attenuation scenario used to develop the framework in the following sections. In detail, we observe the car waiting at the red light at time values 130-250, the first transition from low to

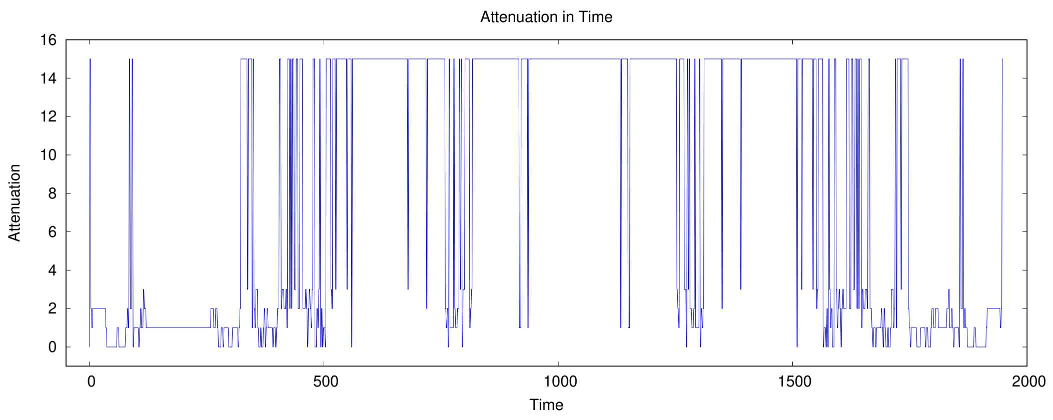


Figure 4.3: Basic Attenuation Data Scenario including car turn-around and red traffic light.

high attenuation at 500 seconds, the turn-around at 1050 seconds, and the second transition from high to low attenuation at 1600 seconds.

### 4.3.3 Attenuation Data Augmentation

This section describes the process for augmenting our data in order to obtain more slightly altered attenuation scenarios based on the basic one. In this way, we simulate many different cars that traverse through the same city pathway. Understandably, only one car route scenario is not enough to build a robust model that generalizes well adapting to new unseen car routes. This procedure is called data augmentation in the literature.

There are many data augmentation techniques for time series data, including adding noise, cropping, scaling, and time-warping, among many others. Also, advanced augmentation methods are Adversarial Training, Neural Style Transfer, and Generative Adversarial Networks (GANs) based augmentation. This thesis utilizes Tsaug [17], a python library for time series augmentation that offers a set of augmentation schemes for time series forecasting that could be stacked in a pipeline of multiple connected augmenters.

After experimentation with the possible augmenters, we conclude to employ a Tsaug pipeline of time-warping, noise, and cropping. As a result, we create multiple scenarios slightly altered from the basic one, representing many different cars. Specifically, this approach simulates the continuous monitoring of a specific city pathway that many vehicles traverse with various velocities and different driving methods. Nevertheless, the same attenuation pattern exists in all the car routes since they drive through the same geographical area. There are only some minor differences - fluctuations based on the driving style and the noise.

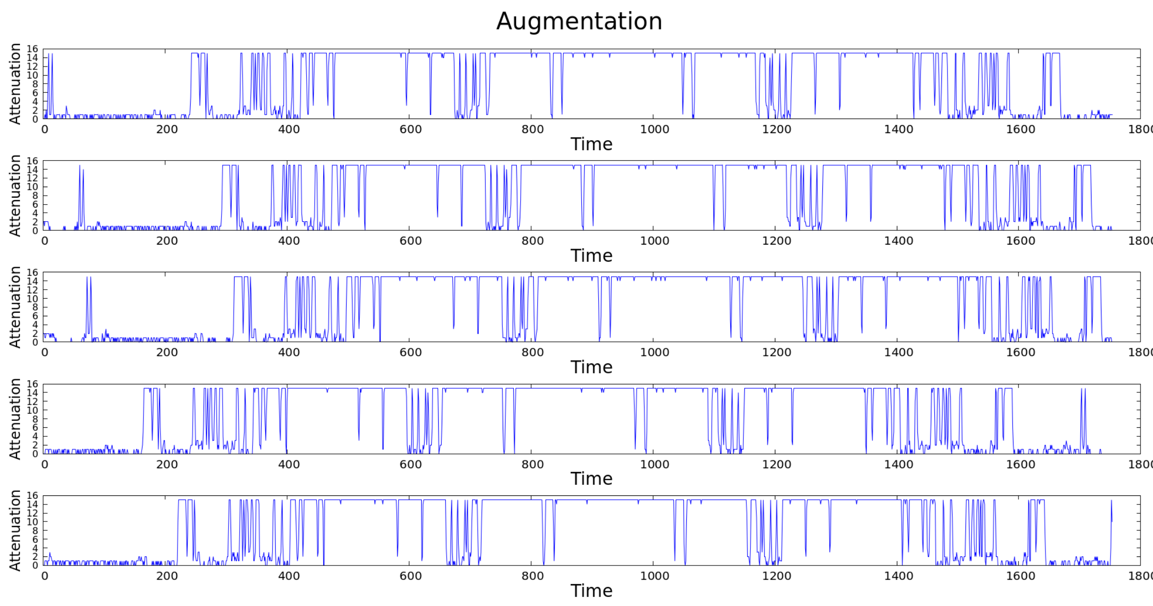


Figure 4.4: Attenuation Data Augmentation: Simulating the continuous monitoring of a specific city pathway. Every augmented scenario corresponds to a car that traverses the pathway with a unique driving style.

Figure 4.4 shows five indicative augmented scenarios representing five different cars driving through the city pathway.

### 4.3.4 CQI Data Collection

This section describes the procedure for collecting CQI data based on the augmented attenuation scenarios mentioned in the previous section. Precisely, we reproduce the car behaviors captured from the city pathway monitoring under a complete infrastructure in the NITOS Testbed. The experimental architecture is similar to that described in the preceding chapter, as seen in Figure 4.5. The only difference is that we do not need the UE's WiFi interface for now since only the LTE CQI data are required.

The FlexRAN Controler plays a vital role in the data collection process. First, we deploy the complete topology of Figure 4.5 and start the experiment. We obtain a random attenuation scenario (a car route) from the augmented data available and start configuring the USRP's attenuation accordingly. Meanwhile, we continuously parse the UE's CQI values with the ML & AI unit hosted on the FlexRAN Controller's node. To achieve this, we periodically send a GET request to the Controller inquiring about the based station's statistics. Then we extract the CQI value from the returned JSON and store it in a CSV. This process continues

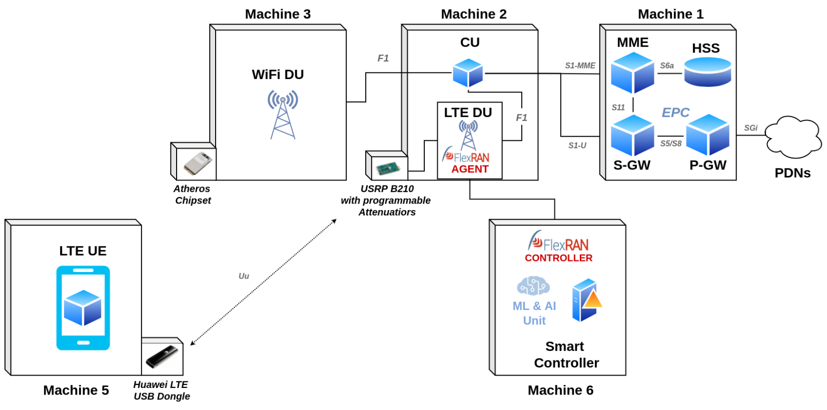


Figure 4.5: Nitos Testbed Data Collection Architecture

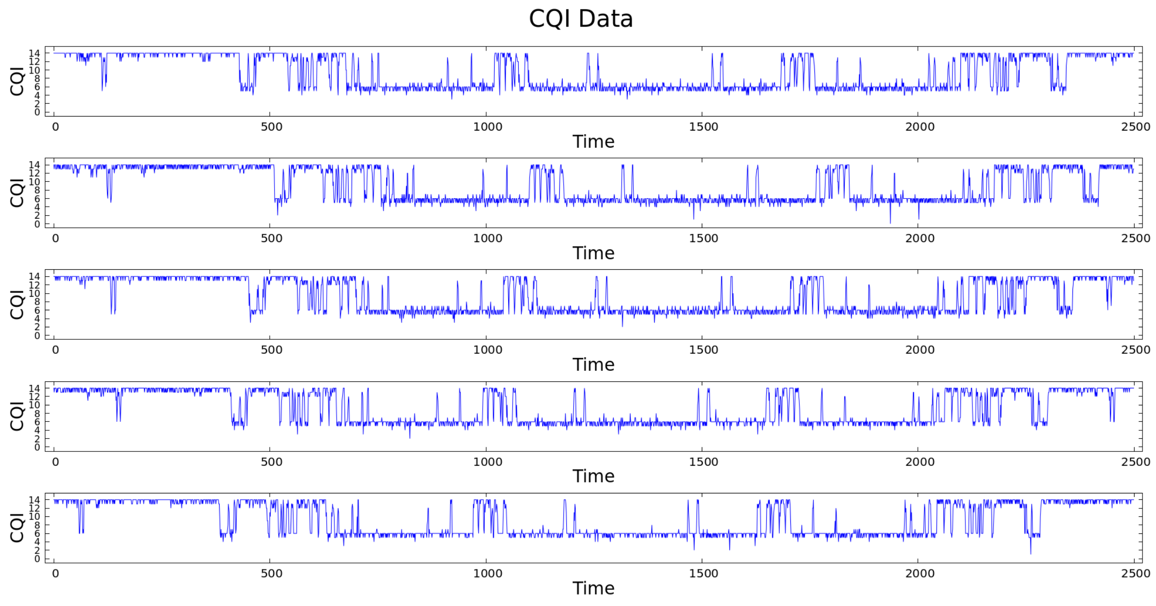


Figure 4.6: Collected CQI data.

until the attenuation scenario is completed (about 2500 CQI values). In this way, we get the CQI data for one scenario corresponding to one car that passes the city pathway.

Repeating this methodology many times, we collect approximately 70 scenarios corresponding to 70 different cars passing the specific road. Finally, we build 70 univariate time-series sequences with a total of 182.000 CQI values. Figure 4.6 depicts five of them, showing slight differences from car to car. Additionally, focusing on the first sequence of the figure, we can spot the first transition when the CQI drops (entering an area with poor coverage) at about 600 seconds and the second phase when the CQI rises (leaving the area with poor coverage) at about 2100 seconds. In this way, we build the final data that represent realistic car behaviors on a specific city road.

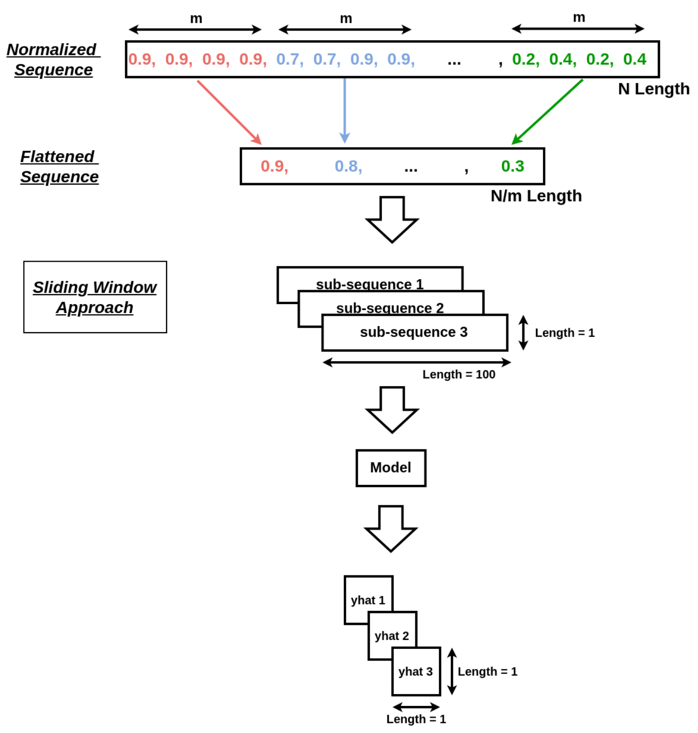


Figure 4.7: CQI Data Pre-processing: Sequence Flattening & Sliding Window Techniques

### 4.3.5 CQI Data Pre-processing

To begin the data pre-processing, we normalize the sequence. Generally, the large inputs in the unscaled data restrain the network's learning and convergence and, in certain situations, prevent it from learning correctly. Our data range is not so large since it is from 0 to 15. Nevertheless, we rescale them to the range [0,1] to boost model training and efficiency.

As a next step, we shrink the data sequences' lengths ($N$), assisting in better data handling by the model. Precisely, we flatten the time sequences by continuously collecting the means for windows with length $m$, as shown in Figure 4.7. In our case, $m$'s optimal value is 4, resulting in flattened sequences with $N/4$ length.

Then, we form the flattened sequences in such a way to feed them appropriately in a model. Specifically, we utilize a sliding window technique transforming the time-series forecasting problem to a supervised one, as illustrated in Figure 4.7. More accurately, we obtain one flattened sequence and split it into several sub-sequences based on a sliding window with length 100. Hence, the model's input is a univariate time-series window with a length 100 corresponding to 400 non-flattened sequence values. At the same time, the output is a single continuous value, representing the predicted CQI as a mean of the following 15 flattened CQI values corresponding to 60 non-flattened values. To summarize, we practically obtain

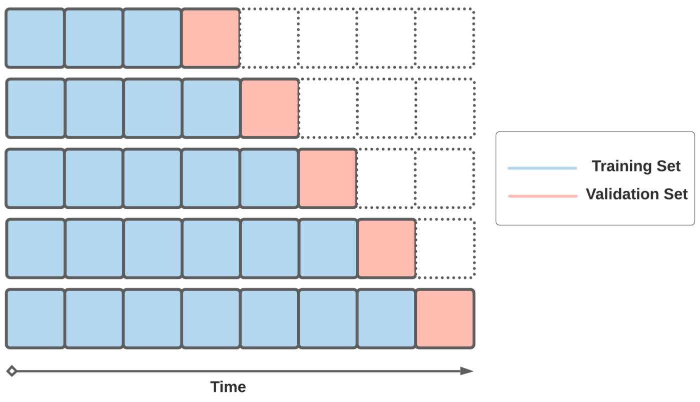


Figure 4.8: Time Series Cross-Validation Technique: Every square represents one Fold. This figure shows five iterations where we increase the training set by one Fold and use the following one for validation. In the end, we obtain the mean of all validation errors.

400 non-flattened values and predict the mean of the next 60 continuously by flattening and feeding them into the model.

## 4.4 Searching and Evaluation of ML & AI Algorithms

This section investigates numerous machine and deep learning techniques to find the best model for fitting the data. Beginning from simple linear algorithms like Linear Regression, we reach to employing complicated neural network architectures such as LSTMs utilizing the Tensorflow-Keras API. We use all the augmented traffic scenarios available as training data while employing the basic non-augmented scenario only for validation. In general, we grid-search and tune the hyperparameters for every algorithm to find its best version and follow a 2-step evaluation technique.

Every tested model has its subsection where we implement and evaluate it. As first-step evaluation, we implement a time-series cross-validation technique based on K-fold cross-validation but designed to keep the time sequence, as shown in Figure 4.8. Specifically, we split the augmented sequences (training data) equally in folds and create two sets, the training, and the validation set. Then, on every iteration, we increase the training set by appending one following Fold, and we use its next one for validation. Finally, we obtain the mean of all validation errors, calculating the final model validation value. From now on, we refer to this step as cross-evaluation.

As a second evaluation step, we utilize the obtained CQI values from the basic non-augmented car route scenario, which was excluded from the training set. We will use these data for the final experiment on the Testbed, validating the model's generalization to unseen data. From now on, we refer to this step as generalization evaluation.

Both these steps are highly essential for the overall model evaluation. The generalization one allows us to measure how well the model finds the pattern in unseen and unaugmented data since it is only trained on the augmented scenarios. It also indicates the model's performance on the data used for the experiment. Finally, the cross-evaluation shows us the model's ability to find the pattern in the augmented data adapting to new unseen fluctuations.

Every model is trained and evaluated utilizing the Google Colab Notebooks using a non-premium subscription with a TPU backend for faster model analysis. We also measure its training time as another important metric for finding the faster one. Thus, we need to keep this configuration in mind when studying the evaluation results for every algorithm.

### 4.4.1 Linear Regression

Linear Regression is a basic algorithm used for regression problems in machine learning assuming a linear relationship between the predictors [18]. The mathematical formula for the simple univariate linear regression with one explanatory variable is:

$$\hat{y_i} = w_0 + w_1 * x_i + \epsilon_i, \tag{4.1}$$

where $\hat{y_i}$ is the dependent variable, $x_i$ is the explanatory variable, $w_0$ is the intercept, $w_1$ is the slope and $\epsilon_i$ is the random noise for instance i. We generally consider the input a dataset with n instances-rows described by p explanatory variables. When p > 1, the multiple linear regression formula is:

$$\hat{y_i} = w_0 + w_1 * x_{i1} + w_2 * x_{i2} + ... + w_n * x_{in} + \epsilon_i \tag{4.2}$$

$$\hat{y_i} = w_0 + \sum_{j=1}^{p}(w_j * x_{ij}) + \epsilon_i \tag{4.3}$$

The $w_1$ can be interpreted as the average effect on $\hat{y_i}$ of a one-unit increase in $x_{i1}$, holding all other predictors fixed.

The model fits the data by calculating all the appropriate $w_j$ and $w_0$ coefficients that minimize a cost function. Specifically, the most common fitting technique is the Least Squares that minimizes the residual sum of squared errors (RSS).

$$RSS = \sum_{i=1}^{n}(y_i - \hat{y_i})^2 = \sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p} w_j * x_{ij} - w_0\right)^2, \tag{4.4}$$

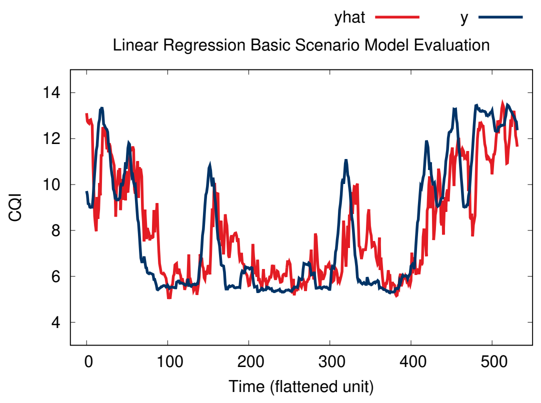


Figure 4.9: Linear Regression Generalization Evaluation

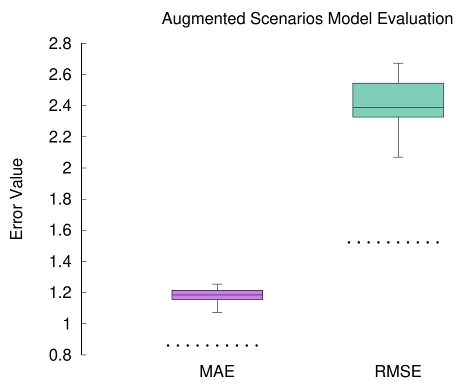


Figure 4.10: Time Series Cross-Validation Evaluation of Linear Regression Model.

where $y_i$ are the real expected values and the $\hat{y}_i$ are the predictions.

In our case, the model's input window forms a multivariate vector with a length of 100, as mentioned in section 4.3.5. Thus, we employ the scikit-learn multiple linear regression implementation for data fitting [19]. Figure 4.9 illustrates the results for the generalization evaluation on the basic non-augmented traffic scenario. The calculated MAE is 1.24, while the MSE is 2.72.

Figure 4.10 shows the cross-evaluation results on the augmented scenarios. Specifically, we implement a 20-Fold time-series evaluation with Folds of 500 values. We repeat it for 10 iterations to acquire enough confidence for the validity of the results. Finally, the distributions of the MAE and MSE are shown in Figure 4.10, with the median cross-evaluation MAE being 1.19, while the MSE is 2.39. Understandably, the model does not fit the data optimally.

On the timing side, the measurements show that the model fits the data quickly. Precisely, the cross-evaluation duration is 70 seconds, while the training-data fitting duration is 0.4 seconds.

### 4.4.2 Ridge Regression

Ridge Regression or L2 Regularization is a regularization technique for Linear Regression. The main drawback of Linear Regression is instability in minimizing the RSS when highly correlated variables exist. Precisely, it can lead to growing large positive $w_j$ coefficient values on some variables and large negative on others, creating a high variance or an overfitting model.

L2 Regularization inserts a penalty term proportional to the squared values of $w_j$ (ex-

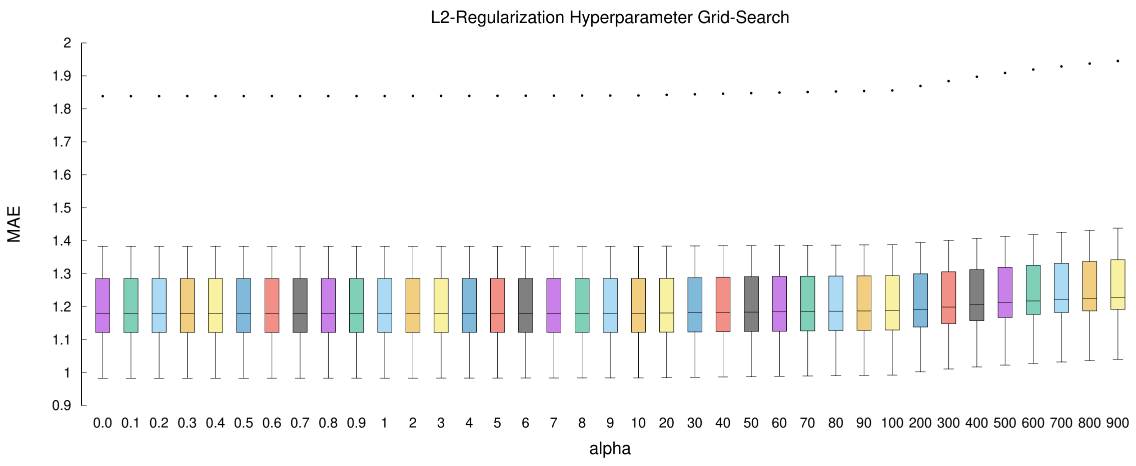


Figure 4.11: Ridge Regression Grid-Searching of Hyperparameter *alpha*.

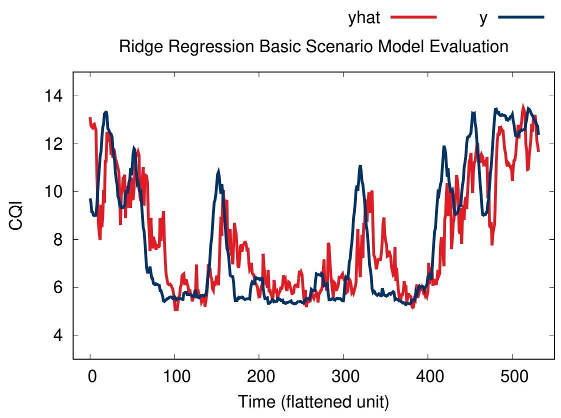


Figure 4.12: Ridge Regression Generalization Evaluation with *alpha* = 3

cluded $w_0$) in the cost function to address this problem:

$$RSS_{L2} = \sum_{i=1}^{n} \left( y_i - \sum_{j=1}^{p} x_{ij} * w_j - w_0 \right)^2 + \lambda \sum_{j=1}^{p} w_j^2, \tag{4.5}$$

This way, it controls the size of the coefficients providing numerical stability and increasing predictive performance.

We implement Ridge Regression utilizing the scikit-learn library [20]. Moreover, we apply a 25-Fold time series cross-evaluation with Folds of 200 values. For finding the optimal regularization hyperparameter $\lambda$ (*alpha* in scikit-learn documentation), we grid-search multiple values from 0 to 900, as shown in Figure 4.11. However, there are no significant MAE changes for almost all tested values except the last large ones, where the MAE rises. Thus, we randomly choose one of the first values and precisely the value 3.

We evaluate the model by calculating the median cross-evaluation MAE to be 1.18, while the MSE is 2.45. Additionally, Figure 4.12 illustrates the results for the generalization eval-

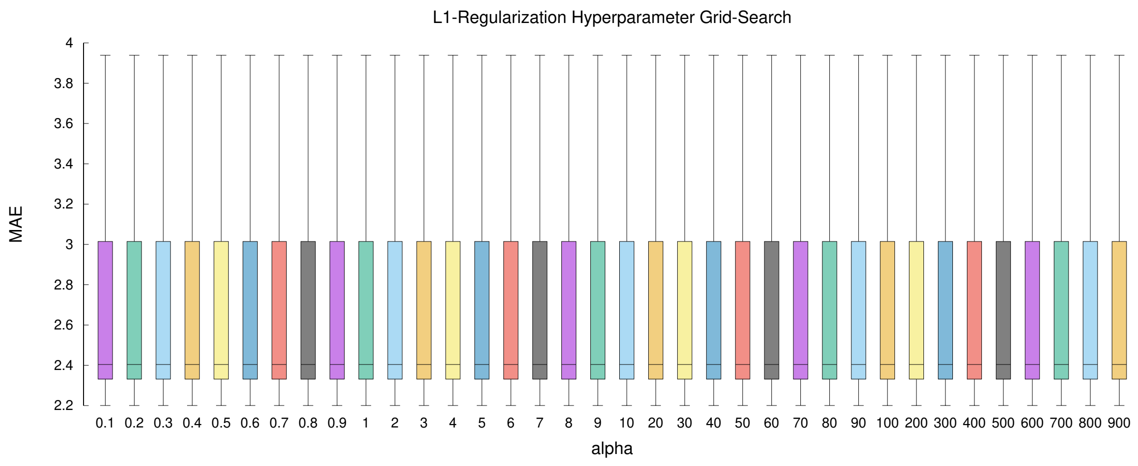


Figure 4.13: Lasso Regression Grid-Searching of Hyperparameter alpha.

uation on the basic non-augmented traffic scenario. The calculated MAE is 1.24, while the MSE is 2.72. As a result, the model does not fit the data efficiently.

On the timing side, the measurements show that the model fits the data fast. Precisely, the cross-evaluation duration is 86 seconds, while the training-data fitting duration with the final hyperparameter is 0.1 seconds.

### 4.4.3 Lasso Regression

Lasso Regression or L1 Regularization is also a regularization technique for Linear Regression that overcomes the disadvantages of L2 Regularization. Precisely, despite the latter's benefits, the L2-norm does not encourage sparsity, resulting in non-zero and ofter similar coefficient values. On the other hand, L1 Regularization provides a regularized feature selection approach by minimizing the irrelevant features resulting, this way, on a low variance non-overfitting model. By replacing the L2 norm with the L1 one in equation 4.5, the Lasso Regression's cost function becomes:

$$RSS_{L1} = \sum_{i=1}^{n} \left( y_i - \sum_{j=1}^{p} x_{ij} * w_j - w_0 \right)^2 + \lambda \sum_{j=1}^{p} |w_j|, \tag{4.6}$$

We implement Lasso Regression utilizing the scikit-learn library [21] and use a 25-Fold time series cross-evaluation with Folds of 200 values. For finding the optimal regularization hyperparameter $\lambda$ (*alpha* in scikit-learn documentation), we grid-search multiple values from 0.1 to 900, as shown in Figure 4.13. However, all tested values give huge MAE values. Thus, we randomly choose the value 0.1.

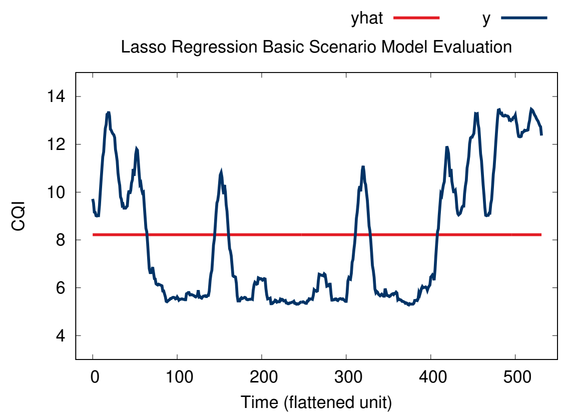


Figure 4.14: Lasso Regression Generalization Evaluation with *alpha* = 0.1

We calculate the median cross-evaluation MAE equal to 2.4, while the MSE is 6.7. Additionally, Figure 4.14 illustrates the results for the generalization evaluation on the basic non-augmented traffic scenario with the *alpha* to be 0.1. The calculated MAE is 2.55, while the MSE is 7.77. We conclude that the model completely fails to fit the data.

On the timing side, the measurements show that the model fits the data fast. Precisely, the cross-evaluation duration is 66 seconds, while the training-data fitting duration with the final hyperparameter is 0.08 seconds.

### 4.4.4 Elastic Net Regression

Elastic Net Regularization is also a regularization technique for Linear Regression proposed to solve Lasso Regression's restrictions [22]. Specifically, one drawback of Lasso is that its feature selection may be unstable since it is too dependent on data. Also, it is unable to perform grouped selection since it only picks a single variable from a group of similar features.

The solution is combining L2 and L1 Regularization to obtain the benefits of both worlds. Thus, Elastic Net's cost function is:

$$RSS_{ElasticNet} = \frac{\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p} x_{ij} * w_j - w_0\right)^2}{2n} + \lambda(\frac{1-a}{2}\sum_{j=1}^{p} w_j^2 + a\sum_{j=1}^{p}|w_j|) \quad (4.7)$$

Here, there are two tunable hyperparameters, the $\lambda$ and the $\alpha$. The $\lambda$ is the regularization strength as in Ridge and Lasso Regression, while the $\alpha$ is the mixing parameter between L2 ($\alpha$=0) and L1 ($\alpha$=1) Regularizations. With the appropriate tuning of the hyperparameters,

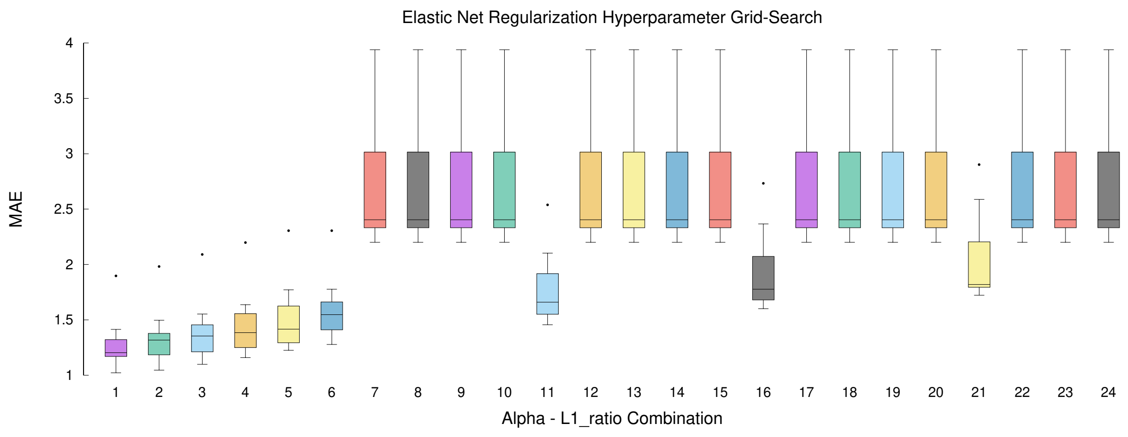


Figure 4.15: Elastic Net Regression Grid-Searching of Hyperparameters alpha and l1_ratio.

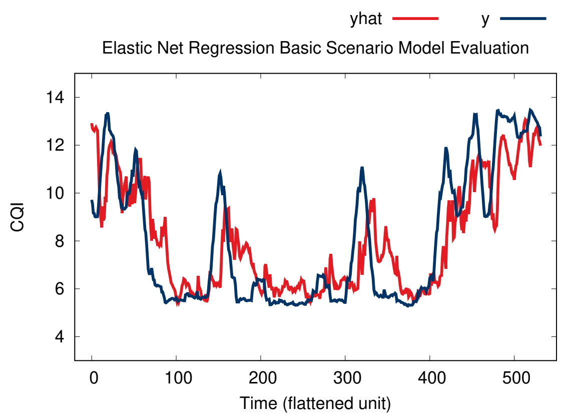


Figure 4.16: Elastic Net Regression Generalization Evaluation with *alpha* = 0.01 and *l1_ratio* = 0.01

we could utilize L1 to generate a sparse model and the quadratic part (L2) to encourage the grouping effect and the L1 feature selection stability.

We implement Elastic Net Regularization utilizing the scikit-learn library [23] and use a 25-Fold time series cross-evaluation with Folds of 200 values. The hyperparameter $\lambda$ is the *alpha* in the sklearn documentation, and the $\alpha$ is the *l1_ratio*. For finding the optimal regularization hyperparameters, we grid-search multiple combinations for $\alpha$ and *l1_ratio*, as shown in Figure 4.15. The best one is the first with $\alpha$ and *l1_ratio* equal to 0.01.

Using these values, we find that the median cross-evaluation MAE equals 1.2, while the MSE is 2.47. Additionally, Figure 4.16 illustrates the results for the generalization evaluation on the basic non-augmented traffic scenario. The calculated MAE is 1.27, while the MSE is 2.78. In conclusion, the model is not able to fit the data optimally.

On the timing side, the measurements show that the model fits the data fast. Precisely,

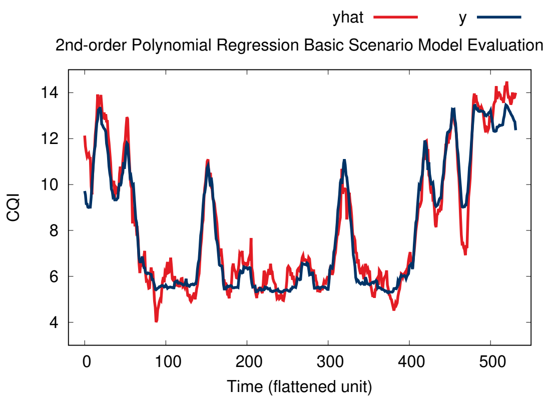


Figure 4.17: Polynomial Regression Generalization Evaluation with $2^{nd}$-order Degree

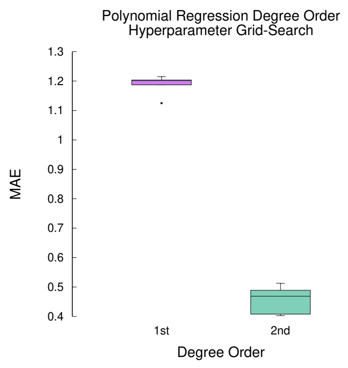


Figure 4.18: Polynomial Regression Model Grid-Searching of Hyperparameter *degree*

the cross-evaluation duration is 106 seconds, while the training-data fitting duration with the final hyperparameters is 0.09 seconds.

### 4.4.5 Polynomial Regression

Polynomial Regression is a technique to fit non-linear data by utilizing polynomial equations [24] [25]. Nevertheless, it is a linear estimation since the coefficients come from a linear regression function $E(y|x)$. Thus, it is considered a special case of Linear Regression. The mathematical formula for a simple univariate polynomial regression is:

$$\hat{y}_i = w_0 + w_1 * x_i + w_2 * x_i^2 + w_3 *_i^3 + ... + w_n * x_i^n, \tag{4.8}$$

$$\hat{y}_i = w_0 + \sum_{k=1}^{n} w_k * x_i^k, \tag{4.9}$$

where $n$ is the order of the polynomial.

Polynomial Regression can also be applied to multiple regression variables, called multivariate polynomial regression. The mathematical formula for a second-order polynomial regression with two regression variables is:

$$\hat{y}_i = w_0 + w_1 * x_{i1} + w_2 * x_{i2} + w_{11} * x_{i1}^2 + w_{22} * x_{i2}^2 + w_{12} * x_{i1} * x_{i2} + \epsilon_i, \tag{4.10}$$

where the $w_0 + w_1 * x_{i1} + w_2 * x_{i2}$ is the linear component, the $w_{11} * x_{i1}^2 + w_{22} * x_{i2}^2$ is the quadratic component and the $w_{12} * x_{i1} * x_{i2}$ is the cross product or interaction component.

We implement Polynomial Regression by utilizing the scikit-learn library and the Linear Regression. We use a 5-Fold time series cross-evaluation with Folds of 1000 values since the polynomial training is more time-demanding from the previous implementations. To find the optimal regularization hyperparameter *degree*, we grid-search two values ($1^{st}$ and $2^{nd}$ degrees) as shown in Figure 4.18. We avoid testing higher-order polynomials because they demand prohibitively huge memory capacity and training time that do not scale.

The results indicate that the optimal polynomial order is 2. We calculate the median cross-evaluation MAE equal to 0.47, while the MSE is 0.39. Additionally, Figure 4.17 illustrates the results for the generalization evaluation on the basic non-augmented traffic scenario. The calculated MAE is 0.53, while the MSE is 0.47. We conclude that the model fits the data with high accuracy, making it a good choice.

On the timing side, the measurements show that the model fits the data sufficiently fast. Precisely, the cross-evaluation duration is 989 seconds (16.5 mins), while the training-data fitting duration with the final hyperparameter is 214 seconds (3.5 mins).

### 4.4.6 ARIMA

ARIMA stands for Autoregressive Integrated Moving Average and is one of the most widely used forecasting models for univariate time series data forecasting. It is a generalization of the simpler AutoRegressive Moving Average incorporating the concept of integration. Its key characteristics are:

- **Autoregression (AR):** An algorithm utilizing the dependent relationship between the observation and several lagged observations. The lag order is specified as the $p$ parameter.
- **Integrated (I):** A method to make the time series stationary by employing the differencing of the observations, for instance, subtracting an observation from another one at a preceding time slot. The degree of differencing is specified as the $d$ parameter.
- **Moving Average (MA):** An algorithm exploiting the relationship between an observation and the residual error of a moving-average algorithm applied to lagged observations. The order of moving average is specified as the $q$ parameter.

ARIMA models are very robust when dealing with the trend in a time series, but they do not support seasonality handling. Thus, there is an extension of the ARIMA supporting seasonality modeling called the SARIMA model.

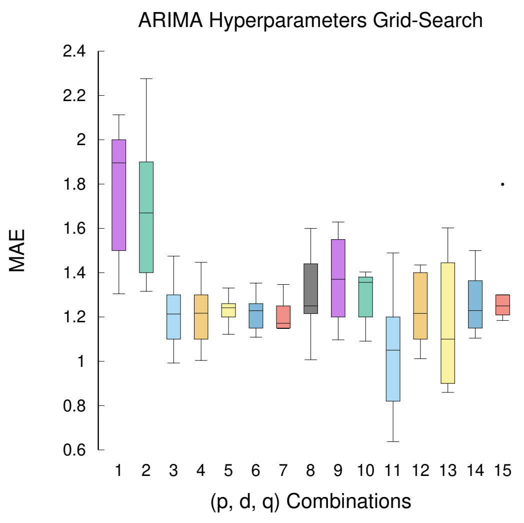


Figure 4.19: ARIMA Hyperparameter Grid-Searching.

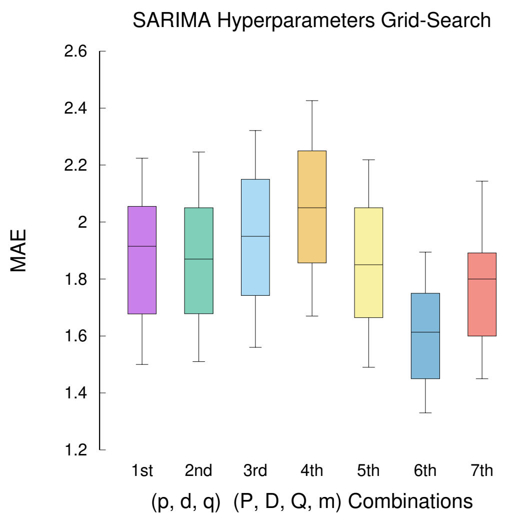


Figure 4.20: Seasonal ARIMA Hyperparameter Grid-Searching.

Seasonal Autoregressive Integrated Moving Average (SARIMA) or Seasonal ARIMA incorporates three new additional hyperparameters ($P$, $D$, $Q$) for the seasonal part, related to the autoregression (AR), the differencing (I), and the moving average (MA). Moreover, it specifies another hyperparameter $m$ for the seasonality interval.

In our implementation, we employ both Seasonal [26] and simple ARIMA [27] to find which fits our data in the best way. To do that, we need to merge all the time series available, creating a large univariate sequence. Then, we flatten it following the same procedure as described in the pre-processing section(4.3.5). After analyzing the autocorrelation plot, we grid-search various combinations both for the ARIMA's ($p$,$d$,$q$) values and for the SARIMA's

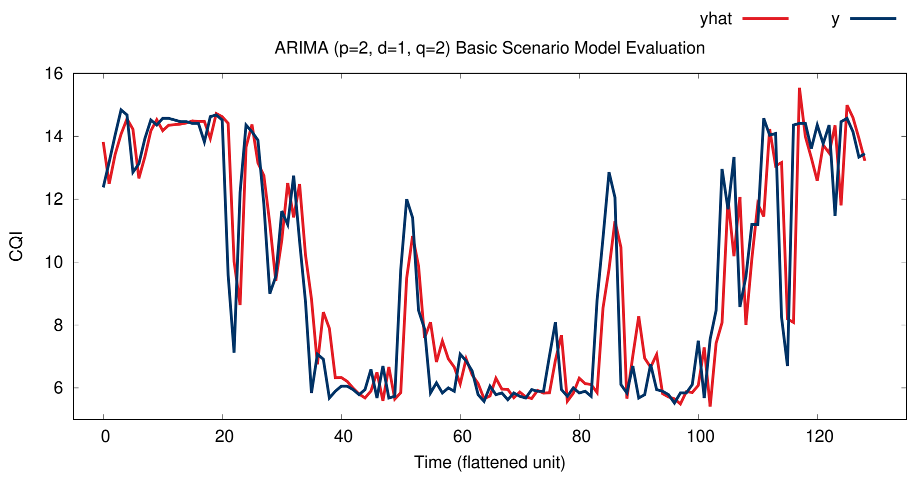


Figure 4.21: ARIMA (p=2, d=1, q=2) Generalization Evaluation.

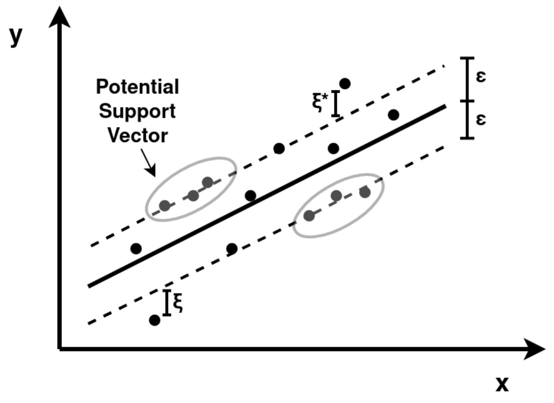


Figure 4.22: One-Dimensional Linear Support Vector Regression

($p$,$d$,$q$) and ($P$, $D$, $Q$, $m$) values, as shown in Figures 4.19 and 4.20, to find the optimal hyperparameters by using a 5-Fold time series cross-evaluation with Folds of 100 values.

The results show that the ARIMA outperforms the SARIMA model in terms of the Mean Absolute Error. Thus, we choose ARIMA's best combination of parameters for fitting the data, including $p$ equals 2, $d$ equals 1, and $q$ equals 2. The calculated median cross-evaluation MAE equals 1.05. Additionally, Figure 4.21 illustrates the results for the generalization evaluation on the basic non-augmented traffic scenario. The calculated MAE is 1.07, while the MSE is 2.64. We conclude that the model does not fit the data optimally.

On the timing side, the measurements show that the model does not fit the data fast. Precisely, the training-data fitting duration is 192 seconds, with the model fitted only in 20% of the training data. That means that the model's fitting could last up to 1000 seconds (16.5 mins) for the complete training set. Moreover, a considerable drawback of ARIMA models for our framework is the need for repeated data fitting before every prediction. This need dramatically slows the prediction process, making it difficult for real-time applications.

### 4.4.7 Support Vector Regression

Support Vector Regression (SVR) is an efficient regression method based on the Support Vector Machines (SVMs) used for classification [28]. It is a generalization of the latter, producing a continuous output value. The univariate one-dimensional approximation function is shown in Figure 4.22, and the mathematical formula is given below:

$$y = \sum_{j=1}^{M} w_j x_j + b, \tag{4.11}$$

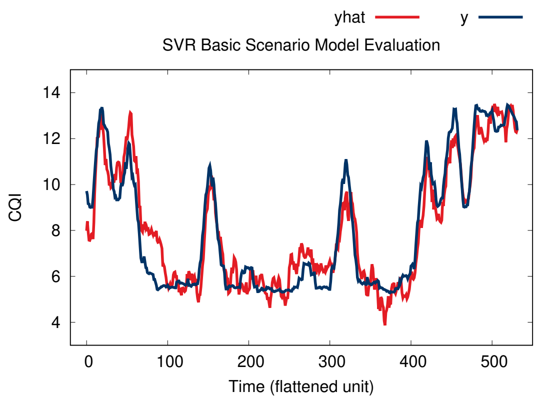


Figure 4.23: SVR Generalization Evaluation with $C = 10$

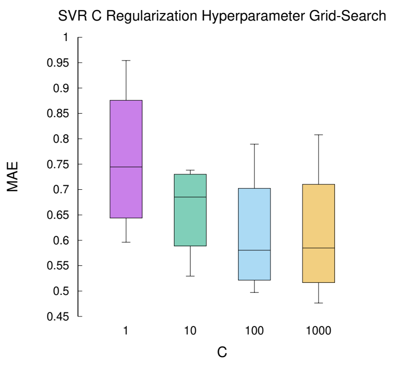


Figure 4.24: SVR Model Grid-Searching of Regularization Hyperparameter $C$

where $y, b \in \mathbb{R}$ and $x, w \in \mathbb{R}^M$. For the multivariate equation, to simplify the formula, include $b$ in the $w$ and increase x by one.

$$y = \begin{bmatrix} w \\ b \end{bmatrix}^T \begin{bmatrix} x \\ 1 \end{bmatrix} = w^T x + b,$$

where $x, w \in \mathbb{R}^{M+1}$.

It is an optimization problem trying to find the narrowest tube centered around the surface while minimizing the prediction error, in other words, the distance between the predictions and the real values. Thus, the optimization function is:

$$min_w \frac{1}{2} \parallel w \parallel^2 \tag{4.12}$$

Implementing a soft-margin approach similar to SVMs, slack variables $\xi$,$\xi^*$ are employed to specify the number of the tolerated outliers. This way, the optimization function becomes:

$$min_w \frac{1}{2} \parallel w \parallel^2 + C \sum_{i=1}^{N} \xi_i + \xi_i^*, \tag{4.13}$$

subject to

$$y_i - w^T x_i \leq \epsilon + \xi_i^*, i = 1, ..., N \tag{4.14}$$

$$w^T x_i - y_i \leq \epsilon + \xi_i, i = 1, ..., N \tag{4.15}$$

$$\xi_i, \xi_i^* \geq 0, i = 1, ..., N \tag{4.16}$$

The $C$ is the regularization parameter giving more weight in minimizing the error.

We implement Support Vector Regression by utilizing the scikit-learn library [29] using a 4-Fold time series cross-evaluation with Folds of 200 values. To find the optimal regularization hyperparameter $C$, we grid-search four values (1, 10, 100, 1000), as shown in Figure 4.24, while employing an *RBF* kernel.

The results indicate that a good choice is $C$ equals 10. We calculate the median cross-evaluation MAE equal to 0.69, while the MSE is 0.59. Additionally, Figure 4.23 illustrates the results for the generalization evaluation on the basic non-augmented traffic scenario. The calculated MAE is 0.73, while the MSE is 0.80. We conclude that the model fits the data with high accuracy, making it a good choice.

On the timing side, the measurements show that the model fits the data fast. Precisely, the cross-evaluation duration is 2887 seconds (48 mins), while the training-data fitting duration with the final hyperparameter is 139.1 seconds (2.5 mins).

### 4.4.8 k-Nearest Neighbors

k-Nearest Neighbors or kNN is a simple algorithm used for classification but could also be applied to regression problems when the data labels are continuous rather than discrete variables. The kNN regression model stores all the training data and make predictions by calculating the average value from the input's k nearest neighbors in the training set. The number of the nearest neighbors (k) used is a hyperparameter and needs to be configured appropriately. kNN calculates the k nearest neighbors utilizing a distance function such as:

- Euclidean Distance:

$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

- Manhattan Distance:

$$\sum_{i=1}^{n}|x_i - y_i|$$

- Minkowski Distance:

$$\left(\sum_{i=1}^{n}|x_i - y_i|^p\right)^{1/p}$$

We implement kNN by utilizing the scikit-learn library [30] using a 10-Fold time series cross-evaluation with Folds of 500 values. We grid-search several values to find the optimal hyperparameter $k$, as shown in Figure 4.26.

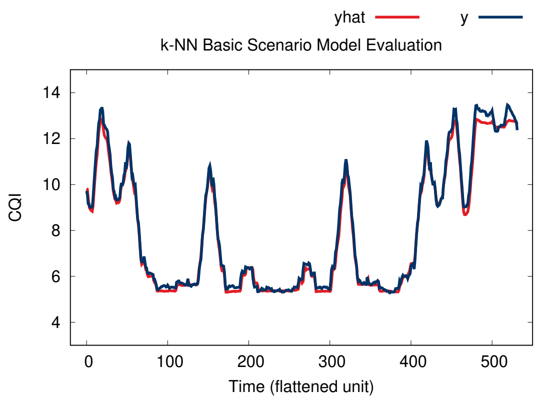


Figure 4.25: k-NN Generalization Evaluation with $k = 23$

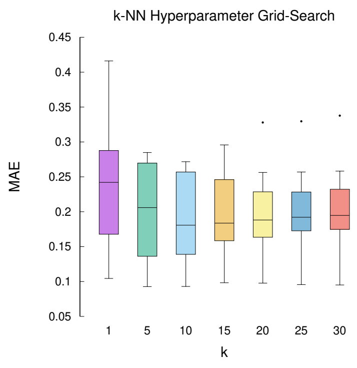


Figure 4.26: k-NN Model Grid-Searching of Regularization Hyperparameter $k$

The results indicate that the optimal choice is k equals 23. We calculate the median cross-evaluation MAE equal to 0.19, while the MSE is 0.18. Additionally, Figure 4.25 illustrates the results for the generalization evaluation on the basic non-augmented traffic scenario. The calculated MAE is 0.175, while the MSE is 0.054. We conclude that the model fits the data with extremely high accuracy, making it an excellent choice.

On the timing side, the measurements show that the model fits the data fast. Precisely, the cross-evaluation duration is 0.47 seconds, while the training-data fitting duration with the final hyperparameter is 0.007 seconds. Nevertheless, these values are misleading since the kNN is a non-parametric algorithm that does not calculate coefficients or weights. On the contrary, it just stores the training data and calculates the distances only at the prediction time. This technique is a considerable drawback when used on real-time applications since the prediction time could be dramatically slowed down depending on the amount of training data.

### 4.4.9 Decision Tree

Decision Tree is one of the most widely used machine learning techniques deployed in classification and regression analysis [31]. A decision tree is a specific type of flowchart structure consisting of several decision-making nodes leading to a final prediction value based on decision thresholds rules.

Regression Decision Tree's target is to predict a continuous variable by combining decision trees and linear regression. Precisely, they receive input features and apply decision and splitting rules based on an efficient recursive partitioning algorithm utilizing a least-

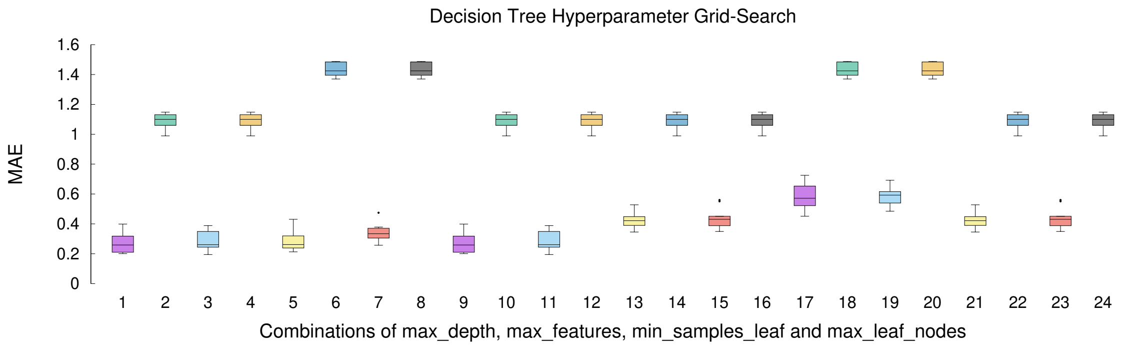


Figure 4.27: Decision Tree Model Grid-Searching

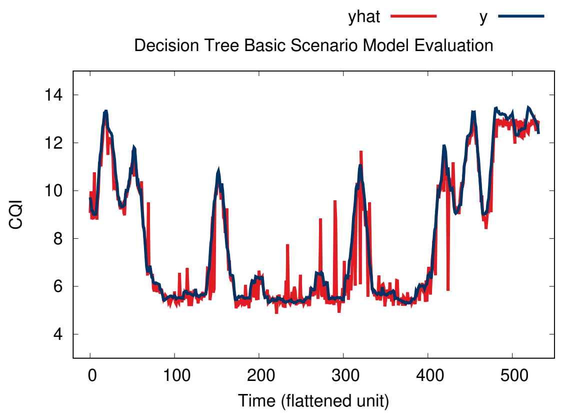


Figure 4.28: Decision Tree Generalization Evaluation with $max\_features = 10$ and $min\_samples\_leaf = 1$

squares criterion. There are various decision tree algorithms for regression, such as CART and CHAID, that assign a constant mean leaf value. Moreover, RETIS and M5 algorithms utilize linear regression models at the leaves.

We implement Decision Tree Regression by utilizing the scikit-learn library [32]. We use a 10-Fold time series cross-evaluation with Folds of 500 values. To find the optimal regularization hyperparameters, we grid-search various combinations of *max_depth*, *max_features*, *min_samples_leaf*, and *max_leaf_nodes*, as shown in Figure 4.27.

The results indicate that the optimal tuning is with *max_features* equals 10, and with *min_samples_leaf* equals 1, configuring all the others to their default value. We calculate the median cross-evaluation MAE equal to 0.26, while the MSE is 0.39. Additionally, Figure 4.28 illustrates the results for the generalization evaluation on the basic non-augmented traffic scenario. We see some prediction fluctuations-bursts at some points, but the overall accuracy seems high. The calculated MAE is 0.378, while the MSE is 0.48. We conclude that the model

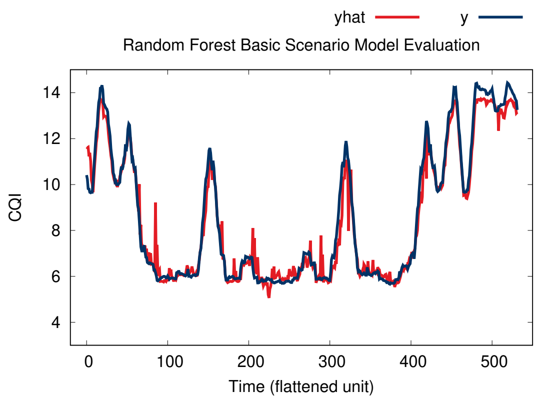


Figure 4.29: Random Forest Generalization Evaluation with $n\_estimators = 200$

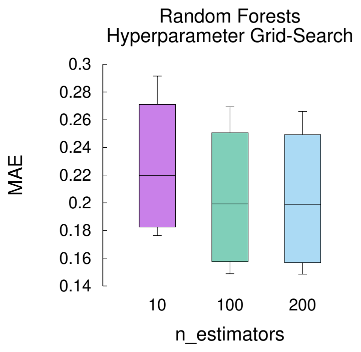


Figure 4.30: Random Forest Model Grid-Searching

fits the data with high accuracy, making it a very good choice.

On the timing side, the measurements show that the model fits the data very fast. Precisely, the cross-evaluation duration is 200 seconds (3.5 mins), while the training-data fitting duration with the final hyperparameters is 0.36 seconds.

### 4.4.10 Random Forest

Random Forests are an ensemble technique utilized for both classification and regression tasks. They are widely used and considered a very robust algorithm since they outperform many other approaches, including Decision Trees. The main principle of Random Forests is using multiple Decision Trees (a forest) at training time. The regression output is the average of all these trees' predictions resulting, this way, in higher precision avoiding the overfitting.

Random Forests utilize the bagging ensemble approach, also called Bootstrap Aggregation, commonly used to reduce the variance in a noisy dataset. Precisely, many subsamples of a training set are chosen with replacement (some data points more than once) forming several training subsets. Subsequently, week learners are trained on them, and finally, the results are aggregated to construct the average output for the regression problems, resulting in better performance.

We implement Random Forest Regression by utilizing the scikit-learn library [33]. We use a 4-Fold time series cross-evaluation with Folds of 500 values. To find the optimal hyperparameter *n_estimators*, we grid-search three values (10, 100, and 200), as shown in Figure 4.30.

The results indicate that the optimal value is 200. We calculate the median cross-evaluation

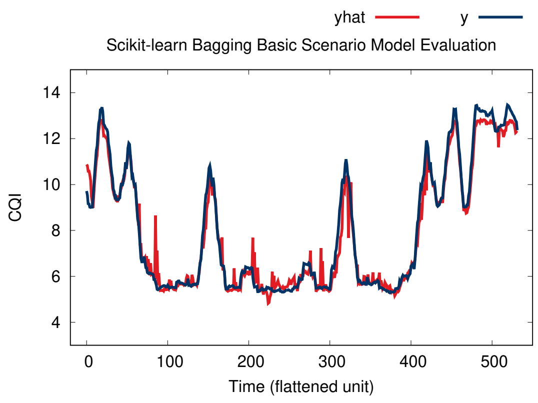


Figure 4.31: Scikit-learn Bagging Generalization Evaluation with $n\_estimators = 200$

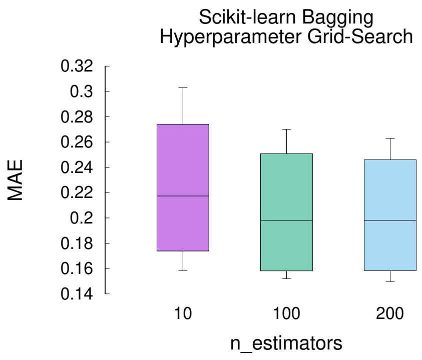


Figure 4.32: Scikit-learn Bagging Model Grid-Searching

MAE equal 0.20, while the MSE is 0.13. Additionally, Figure 4.29 illustrates the results for the generalization evaluation on the basic non-augmented traffic scenario. We see some prediction fluctuations-bursts at some points, but the overall accuracy seems high. The calculated MAE is 0.34, while the MSE is 0.275. We conclude that the model fits the data with high accuracy, making it a good choice.

On the timing side, the measurements show that the model fits the data sufficiently fast. Precisely, the cross-evaluation duration is 2816 seconds (30.5 mins), while the training-data fitting duration with the final hyperparameter is 469 seconds (7.8 mins). Both training and prediction times are significantly affected by the number of estimators. Thus, if necessary, we could also choose lower estimator values since we need a real-time implementation.

### 4.4.11 Bagging Scikit-learn

Scikit-learn library provides its own implementation for the Bootstrap Aggregation scheme [34]. We will deploy it for comparison with the other methods in order to find the best approach for fitting our data. We use a 4-Fold time series cross-evaluation with Folds of 500 values. To find the optimal hyperparameter *n_estimators*, we grid-search three values (10, 100, and 200), as shown in Figure 4.32.

The results indicate that the optimal value is 200. We calculate the median cross-evaluation MAE equal 0.20, while the MSE is 0.13. Additionally, Figure 4.31 illustrates the results for the generalization evaluation on the basic non-augmented traffic scenario. The calculated MAE is 0.315, while the MSE is 0.24. We conclude that the model fits the data with high

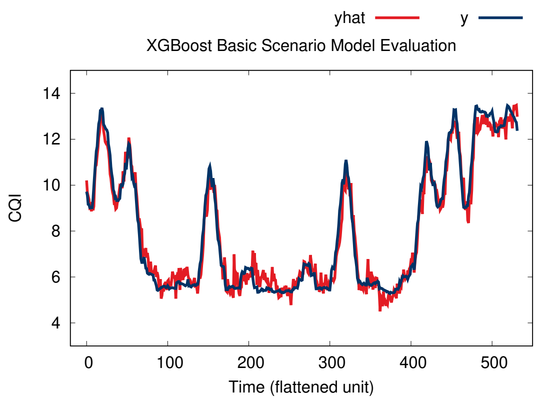


Figure 4.33: XGBoost Generalization Evaluation with $n\_estimators = 10000$

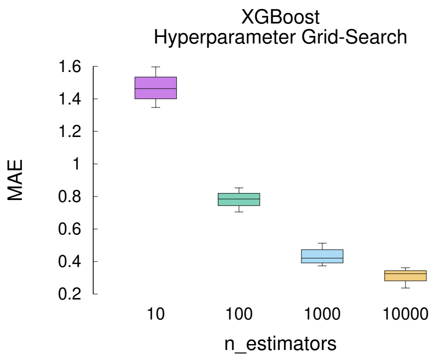


Figure 4.34: XGBoost Model Grid-Searching

accuracy, making it a good choice.

On the timing side, the measurements show that the model fits the data fast. Precisely, the cross-evaluation duration is 3019 seconds (50.5 mins), while the training data fitting duration is 481 seconds (8 mins).

### 4.4.12 XGBoost

XGBoost or Extreme Gradient Boosting is one of the top algorithms in terms of performance and precision [35]. It is an open-source library that implements the gradient boosting algorithm in an efficient manner.

Gradient Boosting is an ensemble approach applied to classification and regression tasks. The main principle is boosting, where an ensemble model is constructed based on decision trees, and new ones are continuously fitted to correct the current prediction error. It also utilizes various differentiable loss functions and gradient descent optimization methods.

We implement XGBoost Regression by utilizing the *xgboost* library [36]. We use a 4-Fold time series cross-evaluation with Folds of 500 values. To find the optimal hyperparameter *n_estimators*, we grid-search four values (10, 100, 1000, and 10000), as shown in Figure 4.34.

The results indicate that the optimal value is 10000. We calculate the median cross-evaluation MAE equal 0.33, while the MSE is 0.20. Additionally, Figure 4.33 illustrates the results for the generalization evaluation on the basic non-augmented traffic scenario. The calculated MAE is 0.384, while the MSE is 0.40. We conclude that the model fits the data with high accuracy, making it a good choice.

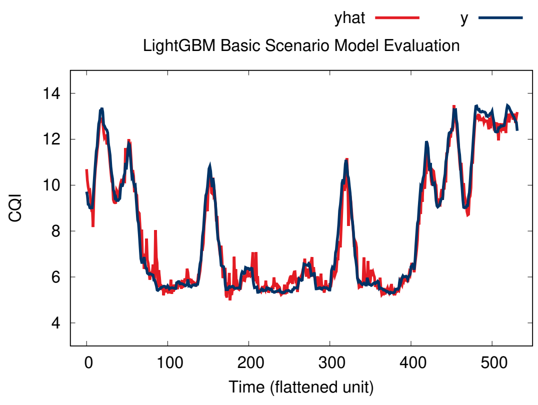


Figure 4.35: LightGBM Generalization Evaluation with $n\_estimators = 1000$

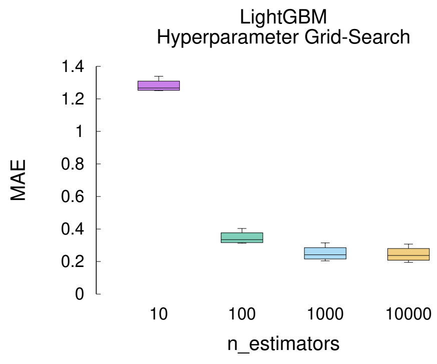


Figure 4.36: LightGBM Model Grid-Searching

On the timing side, the measurements show that the model does not fit the data sufficiently fast. Precisely, the cross-evaluation duration is 4228 seconds (1 hour and 10 mins), while the training-data fitting duration with the final hyperparameter is 954.5 seconds (15.9 mins). Finally, the prediction time is very fast despite the large number of estimators.

### 4.4.13 LightGBM

LightGBM is another popular implementation of the gradient boosting approach [37]. Precisely, it is a tree-based algorithm developed for distributed and faster performance. By incorporating automatic feature selection and also focusing on samples with larger gradients, LightGBM is able to speed up the training and enhance the prediction accuracy.

We implement LightGBM Regression by utilizing the *lightgbm* library. We use a 4-Fold time series cross-evaluation with Folds of 500 values. To find the optimal hyperparameter *n_estimators*, we grid-search four values (10, 100, 1000, and 10000), as shown in Figure 4.36.

The results indicate that the optimal value is 1000. We calculate the median cross-evaluation MAE equal 0.24, while the MSE is 0.13. Additionally, Figure 4.35 illustrates the results for the generalization evaluation on the basic non-augmented traffic scenario. The calculated MAE is 0.319, while the MSE is 0.20. We conclude that the model fits the data with high accuracy, making it a very good choice.

On the timing side, the measurements show that the model fits the data extremely fast. Precisely, the cross-evaluation duration is 654 seconds (11 mins), while the training-data fitting duration is 17 seconds.

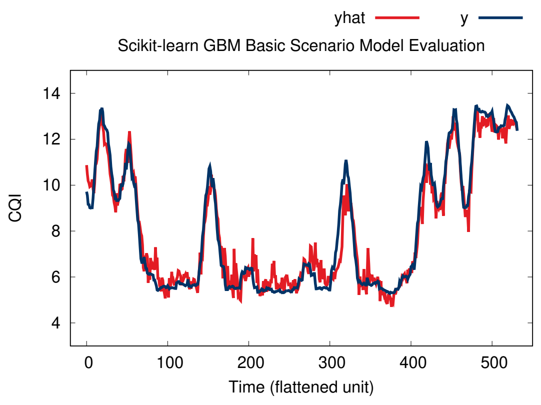


Figure 4.37: Scikit learn GBM Generalization Evaluation with $n\_estimators = 1000$

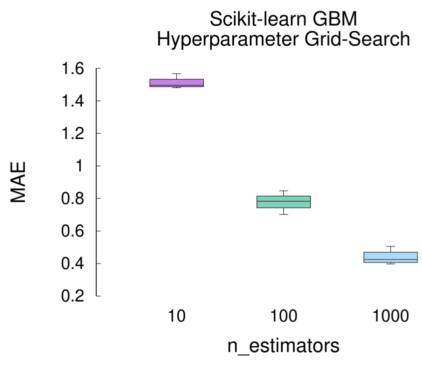


Figure 4.38: Scikit learn GBM Model Grid-Searching

### 4.4.14 GBM Scikit-learn

Scikit-learn library provides its own implementation for the Gradient Boosting scheme [38]. We will deploy it for comparison with the other methods in order to find the best approach for fitting our data. We use a 4-Fold time series cross-evaluation with Folds of 500 values. To find the optimal hyperparameter *n_estimators*, we grid-search three values (10, 100, 1000), as shown in Figure 4.38.

The results indicate that the optimal value is 1000. We calculate the median cross-evaluation MAE equal 0.43, while the MSE is 0.35. Additionally, Figure 4.37 illustrates the results for the generalization evaluation on the basic non-augmented traffic scenario. The calculated MAE is 0.449, while the MSE is 0.37. We conclude that the model fits the data with high accuracy, making it a good choice.

On the timing side, the measurements show that the model does not fit the data fast. Precisely, the cross-evaluation duration is 2463 seconds (41 mins), while the training-data fitting duration is 572 seconds (9.5 mins).

### 4.4.15 CatBoost

CatBoost is another open-source implementation of the gradient boosting approach on decision trees [39]. Its main benefits are great performance without the need for excessive hyperparameter tuning, improved categorical feature handling, scalable GPU support, among many others. We implement CatBoost Regression by utilizing the *catboost* library. We use a 4-Fold time series cross-evaluation with Folds of 500 values. To find the optimal hyperpa-

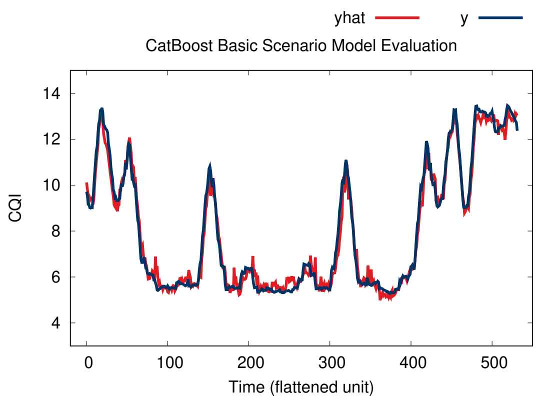


Figure 4.39: CatBoost Generalization Evaluation with $n\_estimators = 10000$

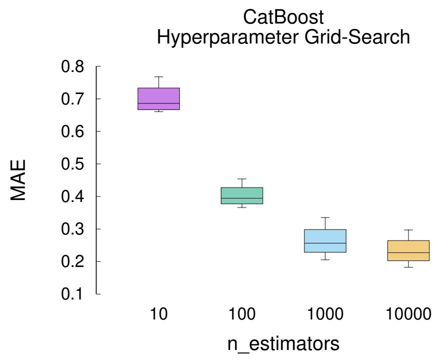


Figure 4.40: CatBoost Model Grid-Searching

rameter *n_estimators*, we grid-search four values (10, 100, 1000, and 10000), as shown in Figure 4.40.

The results indicate that the optimal value is 10000. We calculate the median cross-evaluation MAE equal 0.23, while the MSE is 0.11. Additionally, Figure 4.39 illustrates the results for the generalization evaluation on the basic non-augmented traffic scenario. The calculated MAE is 0.286, while the MSE is 0.14. We conclude that the model fits the data with high accuracy, making it an excellent choice.

On the timing side, the measurements show that the model fits the data fast. Precisely, the cross-evaluation duration is 1234 seconds (20.5 mins), while the training-data fitting duration is 278 seconds (4.5 mins).

### 4.4.16 AdaBoost

AdaBoost or Adaptive Boosting is one of the first algorithms developed following the Boosting scheme. Precisely, in order to build an ensemble model, it utilizes numerous Decision Stumps; small decision trees with only one split (root and two leaves).

We implement AdaBoost Regression by utilizing the scikit-learn library [40]. We use a 4-Fold time series cross-evaluation with a prediction window-fold of 500 values. To find the optimal hyperparameter *n_estimators*, we grid-search three values (10, 100, and 1000), as shown in Figure 4.42.

We choose 10 estimators since the results indicate that the model's error increases as we add more of them. We calculate the median cross-evaluation MAE equal 1.29, while the MSE is 2.25. Additionally, Figure 4.41 illustrates the results for the generalization evaluation on

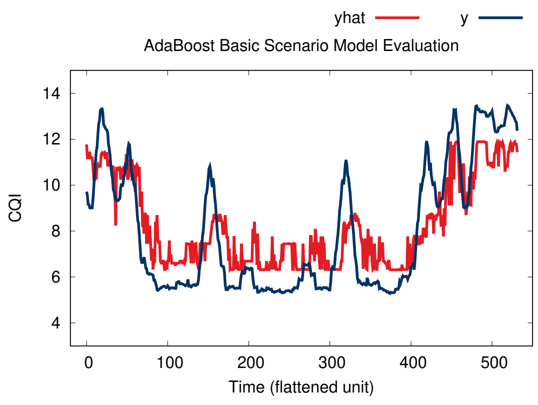


Figure 4.41: AdaBoost Generalization Evaluation with $n\_estimators = 10$

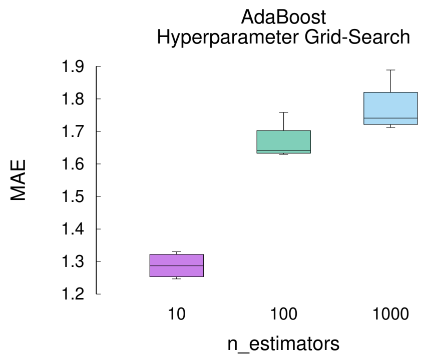


Figure 4.42: AdaBoost Model Grid-Searching

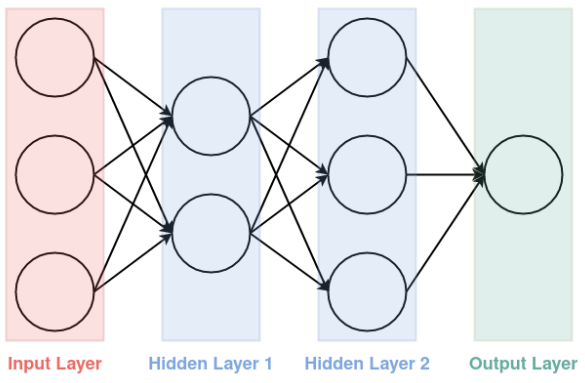


Figure 4.43: Feedforward Neural Network Structure

the basic non-augmented traffic scenario. The calculated MAE is 1.30, while the MSE is 2.29. We conclude that the model fails to fits the data optimally.

On the timing side, the measurements show that the model fits the data fast. Precisely, the cross-evaluation duration is 2144 seconds (36 mins), while the training-data fitting duration is 6 seconds.

### 4.4.17 Feedforward Neural Network

From now on, we apply deep learning techniques to extract the most information possible from our data [41]. Feedforward Neural Networks (FNN) are the first simple Artificial Neural Networks (ANN) that do not form node cycle connections and move forward from one layer to the next.

Figure 4.43 illustrates the structure of NN, which consists of the input, several hidden, and the output layers. In addition, every layer contains numerous neuron nodes that connect with

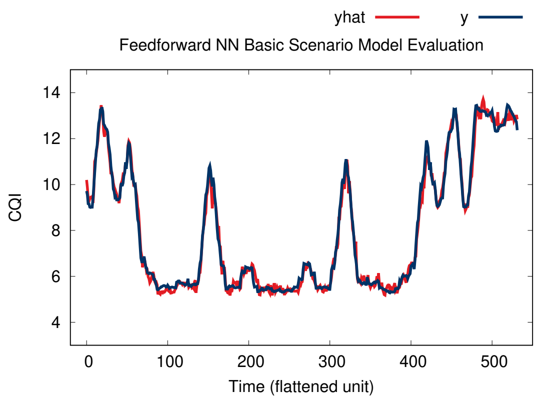


Figure 4.44: Feedforward NN Generalization Evaluation

| Pre-processing | Configuration |
|---|---|
| Input time-steps (non-flattened) | 400 |
| Prediction time-steps (non-flattened) | 60 |
| Flattening window time-steps ($m$) | 4 |
| **Hyper-parameter** | **Configuration** |
| Layers | 2 Hidden Dense + 1 Output Dense |
| Input Shape | 100 (after flattening) |
| Hidden Neurons | 25 |
| Output Neurons | 1 |
| Activation Function | ReLU for Hidden Layers |
| Optimizer | Adam |
| Compile loss | Mean Squared Error |
| Epochs | 640 |
| Batch_size | $2^6$ |

Table 4.1: Feedforward NN Configuration

nodes in prior and following layers with weighted edges. Every neuron calculates its weighted input sum, and whether it is above or below a threshold value based on an activation function, it produces the appropriate output to the next layer.

Feedforward Neural Networks utilize the backpropagation approach to reduce their prediction error. Specifically, the model's output is compared with the actual values, and the error is fed back to the network. It traverses the opposite-backward direction, adjusting the weights utilizing the gradient descent approach. This way, it finds the appropriate weights, enhancing the model's predictive performance.

We implement a Feedforward Neural Network utilizing the Tensorflow-Keras API [42]. After trying various hyperparameter combinations to find the optimal model structure, we pick the values presented in Table 4.1. Since neural networks take a long time for training, we do not implement a time-series cross-evaluation scheme but follow a training-validation evaluation approach.

The calculated training MAE on the augmentation data is 0.02, while the MSE is 0.0011 indicating low bias. On the validation data from the basic non-augmented scenario, the measured MAE is 0.199, while the MSE is 0.069 showing low variance. Figure 4.44 shows the model predictions on the validation data. In conclusion, there is a balance in the bias-variance tradeoff leading to a successful model generalization to unseen data without under-/overfitting. On the timing side, we see that the model's training is completed sufficiently fast. Precisely, we train the model for 640 epochs, and the training finishes after 1141 seconds (19 mins).

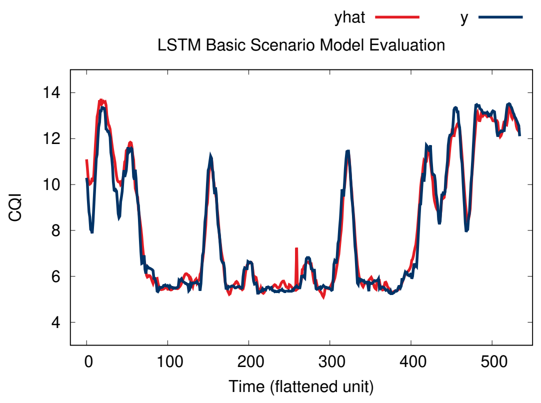


Figure 4.45: LSTM Generalization Evaluation

| Pre-processing | Configuration |
|---|---|
| Input time-steps (non-flattened) | 400 |
| Prediction time-steps (non-flattened) | 50 |
| Flattening window time-steps ($m$) | 4 |

| Hyper-parameter | Configuration |
|---|---|
| Layers | 2 Hidden LSTM + 1 Output Dense |
| Input Shape | (100,1) |
| Hidden Neurons | 25 |
| Output Neurons | 1 |
| Activation Function | ReLU for Hidden Layers |
| Optimizer | Adam |
| Compile loss | Mean Squared Error |
| Epochs | 20 |
| Batch_size | $2^6$ |

Table 4.2: LSTM Configuration

### 4.4.18 Long short-term memory (LSTM)

Long Short-Term Memory (LSTM) networks are a specific type of Recurrent Neural Networks (RNN) widely used for Time Series Forecasting and sequence prediction problems [43]. They are very robust and often outperform other algorithms such as simple Feedforward NNs by employing memory components or conventional RNNs by dealing with the vanishing gradient problem.

The key enabler of their success is their unique design. An LSTM layer comprises numerous memory blocks which contain one or more recurrent memory cells and three units; the input, output, and forget gates. These gates continuously provide write, read, and reset methods to the cell based on the associated weights calculated during the training. This way, the model is able to adapt appropriately to the problem at hand by memorizing values over arbitrary time intervals.

We implement an LSTM Neural Network utilizing the Tensorflow-Keras API. After trying various hyperparameter combinations to find the optimal model structure, we pick the values presented in Table 4.2.

For the model evaluation, we follow a training-validation evaluation approach. The calculated training MAE on the augmentation data is 0.023, while the MSE is 0.0014 indicating low bias. On the validation data from the basic non-augmented scenario, the measured MAE is 0.328, while the MSE is 0.228 showing low variance. Figure 4.45 shows the model predictions on these data. In conclusion, there is a balance in the bias-variance tradeoff leading to a successful model generalization to unseen data without under-/over-fitting. On the timing side, we see that the model's training is completed sufficiently fast. Precisely, we train the

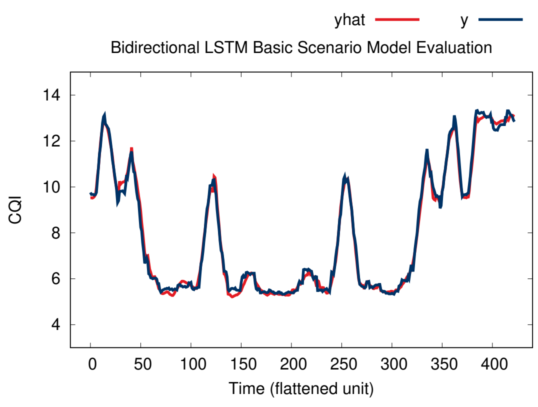


Figure 4.46: Bidirectional LSTM Generalization Evaluation

| Pre-processing | Configuration |
|---|---|
| Input time-steps (non-flattened) | 400 |
| Prediction time-steps (non-flattened) | 70 |
| Flattening window time-steps ($m$) | 5 |

| Hyper-parameter | Configuration |
|---|---|
| Layers | 2 Hidden Bi-LSTM + 1 Output Dense |
| Input Shape | (80,1) |
| Hidden Neurons | 25 |
| Output Neurons | 1 |
| Activation Function | ReLU for Hidden Layers |
| Optimizer | Adam |
| Compile loss | Mean Squared Error |
| Epochs | 55 |
| Batch_size | $2^6$ |

Table 4.3: Bidirectional LSTM Configuration

model for 20 epochs, and the training finishes after 1340 seconds (22 mins).

### 4.4.19 Bidirectional LSTM

Bi-directional LSTMs extend the simple LSTMs providing an additional reversed sequence learning. Precisely, the main idea is incorporating two recurrent layers side-by-side, where the first one handles the original sequence as it is, while the second uses it in the opposite direction. This way, the model could acquire information both from the past and future simultaneously. This technique is utilized in various applications and often enables enhanced predictive performance.

We implement a Bi-directional LSTM Neural Network utilizing the Tensorflow-Keras API. After trying various hyperparameter combinations to find the optimal model structure, we pick the values presented in Table 4.3.

For the model evaluation, we follow a training-validation evaluation approach. The calculated training MAE on the augmentation data is 0.016, while the MSE is 0.0007 indicating low bias. On the validation data from the basic non-augmented scenario, the measured MAE is 0.157, while the MSE is 0.041 showing low variance. Figure 4.46 shows the model predictions on these data. In conclusion, there is a balance in the bias-variance tradeoff leading to a successful model generalization to unseen data without under-/over-fitting. On the timing side, we observe that the model needs much time for training. Precisely, we train the model for 55 epochs, and the training finishes after 3600 seconds (1 hour).

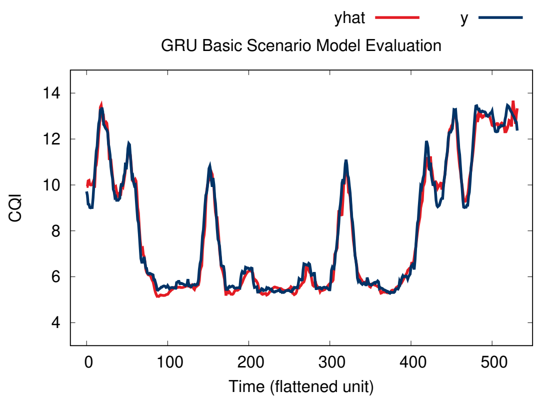


Figure 4.47: GRU Generalization Evaluation

| Pre-processing | Configuration |
|---|---|
| Input time-steps (non-flattened) | 400 |
| Prediction time-steps (non-flattened) | 60 |
| Flattening window time-steps ($m$) | 4 |

| Hyper-parameter | Configuration |
|---|---|
| Layers | 2 Hidden GRU + 1 Output Dense |
| Input Shape | (100,1) |
| Hidden Neurons | 25 |
| Output Neurons | 1 |
| Activation Function | ReLU for Hidden Layers |
| Optimizer | Adam |
| Compile loss | Mean Squared Error |
| Epochs | 55 |
| Batch_size | $2^6$ |

Table 4.4: GRU Configuration

### 4.4.20 Gated Recurrent Unit (GRU)

Gated Recurrent Unit (GRU) is another popular type of Recurrent Neural Network similar to LSTM. It utilizes a gated mechanism to form both long- and short-term memory by incorporating two gates, including the reset and the update gate. Thus, its difference from LSTM is that it does not use an output gate employing, this way, less trainable parameters. This approach often results in a similar performance to the LSTM but with faster training.

We implement a GRU Neural Network utilizing the Tensorflow-Keras API. After trying various hyperparameter combinations to find the optimal model structure, we pick the values presented in Table 4.4.

For the model evaluation, we follow a training-validation evaluation approach. The calculated training MAE on the augmentation data is 0.019, while the MSE is 0.0010 indicating low bias. On the validation data from the basic non-augmented scenario, the measured MAE is 0.256, while the MSE is 0.117 showing low variance. Figure 4.47 shows the model predictions on these data. In conclusion, there is a balance in the bias-variance tradeoff leading to a successful model generalization to unseen data without under-/over-fitting. On the timing side, we observe that the model needs much time for training. Precisely, we train the model for 50 epochs, and the training finishes after 3315 seconds (55 mins).

### 4.4.21 Convolutional Neural Network (CNN)

Convolutional Neural Networks are a specific deep learning technique widely used in the field of computer vision. One of their primary characteristics is the use of kernel filters to

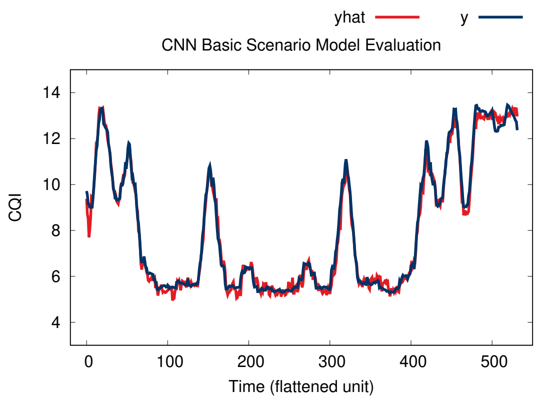


Figure 4.48: CNN Generalization Evaluation

| Pre-processing | Configuration |
|---|---|
| Input time-steps (non-flattened) | 400 |
| Prediction time-steps (non-flattened) | 60 |
| Flattening window time-steps ($m$) | 4 |

| Hyper-parameter | Configuration |
|---|---|
| Layers | Conv1D, MaxPooling1D, Flatten, Dense_1 and Dense_2 |
| Input Shape | (10,10) |
| Conv1D | filters=64, kernel_size=2, activation=ReLU |
| MaxPooling1D Layer | pool_size=2 |
| Dense_1 Layer | 50 Neurons, activation=ReLU |
| Dense_2 Layer | 1 Neuron (output) |
| Optimizer | Adam |
| Compile loss | Mean Squared Error |
| Epochs | 490 |
| Batch_size | $2^6$ |

Table 4.5: CNN Configuration

extract low-level features from the input data by constructing a feature map. Moreover, they utilize pooling layers to extract dominant features, deal with the noise and perform dimensionality reduction resulting, this way, in a more efficient and faster model training. Finally, they also utilize fully connected layers and feed them with the information gathered from prior layers to learn a suitable non-linear function.

We implement a CNN Neural Network utilizing the Tensorflow-Keras API. After trying various hyperparameter combinations to find the optimal model structure, we pick the values presented in Table 4.5.

For the model evaluation, we follow a training-validation evaluation approach. The calculated training MAE on the augmentation data is 0.016, while the MSE is 0.0007 indicating low bias. On the validation data from the basic non-augmented scenario, the measured MAE is 0.210, while the MSE is 0.081 showing low variance. Figure 4.48 shows the model predictions on these data. In conclusion, there is a balance in the bias-variance tradeoff leading to a successful model generalization to unseen data without under-/over-fitting. On the timing side, we observe that the model needs much time for training. Precisely, we train the model for 490 epochs, and the training finishes after 1718 seconds (29 mins).

### 4.4.22 CNN-LSTM Neural Network

A hybrid approach consisting of both CNN and LSTM could result in a more robust and enhanced model. This technique is widely used in many research fields, such as activity recognition, image, and video description. The primary idea is to construct a type of encoder-decoder model architecture. Precisely, the CNN handles the feature extraction procedure, while the LSTM uses the features and analyzes their time sequence to make forecasting.

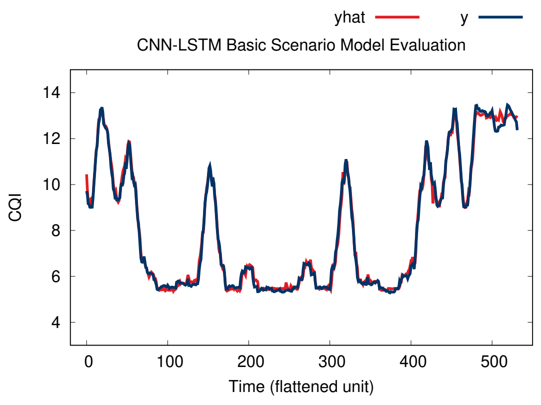


Figure 4.49: CNN-LSTM Generalization Evaluation

| Pre-processing | Configuration |
|---|---|
| Input time-steps (non-flattened) | 400 |
| Prediction time-steps (non-flattened) | 60 |
| Flattening window time-steps ($m$) | 4 |

| Hyper-parameter | Configuration |
|---|---|
| Layers | Conv1D, MaxPooling1D, Flatten, RepeatVector, LSTM_1, LSTM_2, Dense |
| Input Shape | (10,10) |
| Conv1D | filters=64, kernel_size=3, activation=ReLU |
| MaxPooling1D Layer | pool_size=2 |
| RepeatVector Layer | 1 |
| LSTM_1 & LSTM_2 Layers | 25 Neurons, activation=ReLU |
| Dense Layer | 1 Neuron (output) |
| Optimizer | Adam |
| Compile loss | Mean Squared Error |
| Epochs | 70 |
| Batch_size | $2^6$ |

Table 4.6: CNN-LSTM Configuration

We implement a CNN-LSTM Neural Network utilizing the Tensorflow-Keras API. After trying various hyperparameter combinations to find the optimal model structure, we pick the values presented in Table 4.6.

For the model evaluation, we follow a training-validation evaluation approach. The calculated training MAE on the augmentation data is 0.017, while the MSE is 0.0010 indicating low bias. On the validation data from the basic non-augmented scenario, the measured MAE is 0.148, while the MSE is 0.038 showing low variance. Figure 4.49 shows the model predictions on these data. In conclusion, there is a balance in the bias-variance tradeoff leading to a successful model generalization to unseen data without under-/over-fitting. On the timing side, we observe that the model completes the training extremely fast. Precisely, we train the model for 70 epochs, and the training finishes after 350 seconds (6 mins).

## 4.4.23 Models Comparison

Towards finding the optimal ML & AI model for our data, we have investigated numerous algorithms with different designs and have seen both bad and excellent results. To better understand the overall evaluation, we gather all the MAE values from the generalization evaluation of every model in Figure 4.50. The values are sorted in a descending order based on their validation MAE on the non-augmented generalization data (basic car route scenario that was excluded from the training set).

To begin, the Lasso Regression model has completely failed to fit the data with a MAE value of 2.55. Moreover, some models find a way to fit the data but not optimally, including the AdaBoost, the Elastic Net, the Ridge Regression, the Linear Regression, and the ARIMA

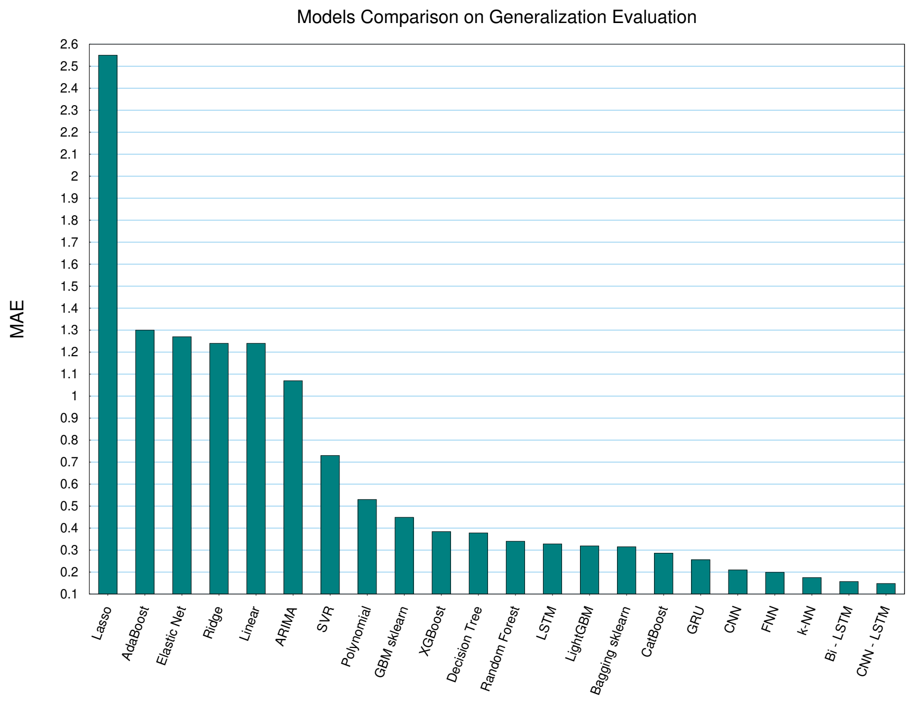


Figure 4.50: Overall Comparison between the various ML & AI Model based on the Generalization MAE from the non-augmented data.

with MAE values over 1. Then, it starts getting more interesting as the MAE drops fast under the 1, with the Support Vector Regressor (SVR) to achieve a MAE of 0.73 and the Polynomial Regression a MAE of 0.53. Subsequently, the tree-based models and the neural networks achieve the best results having MAE values from 0.449 with skicit-learn GBM down to just 0.148 with CNN-LSTM neural network.

Understandably, every method that achieved a generalization MAE under 0.4 is well-suited for our data. Nevertheless, the training time also plays an essential role in extending the framework to support an online-training scheme since we need fast model training. For this reason, in figure 4.51, we gather and compare the training times of the best models (under 0.4 MAE), excluding the non-trainable methods like the k-NN. The results indicate that one of the fastest models is the CNN - LSTM since it outcompetes the other neural networks and converges really fast with 350 seconds (under 6 mins) training time.

Thus, we choose the CNN-LSTM to incorporate in our framework in order to build an ML & AI unit for real-time scheduling of the DUs. The following section presents our experimentation utilizing the CNN-LSTM under a real Testbed.

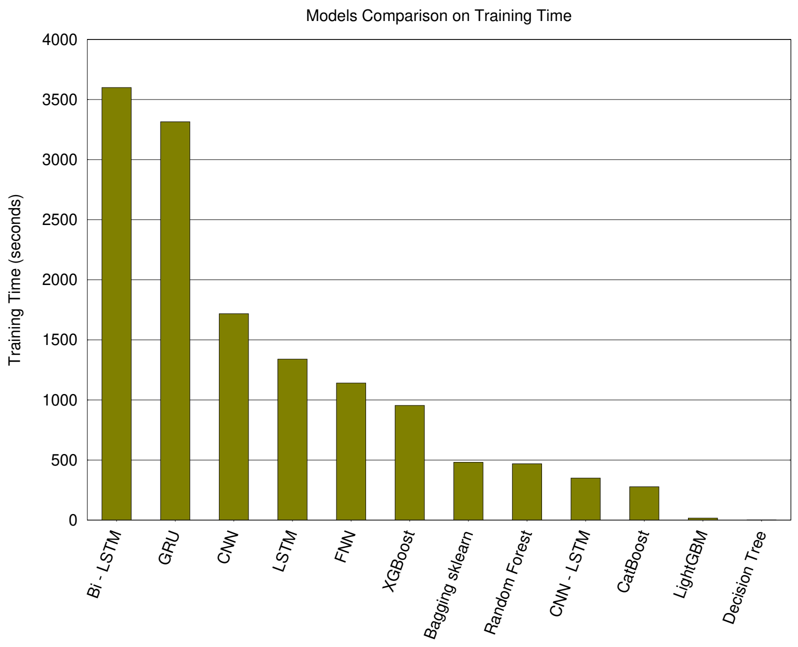


Figure 4.51: Overall Comparison between the various ML & AI Model based on the training time on the augmented data.

## 4.5 Experiments

The experiments assist in validating and evaluating the proposed framework in realistic circumstances. Specifically, we use the NITOS Testbed for our experimental topology, emulating car traffic by deploying the basic non-augmented car route scenario. Subsequently, we monitor the network performance in terms of bandwidth, jitter, and lost packets for two deployments; one with the default behavior without our ML & AI unit and one incorporating it. Then we compare the results to validate that our algorithm works as planned and evaluate its ability in enhancing the UE's QoE.

### 4.5.1 Experimental Setup

Figure 4.52 depicts the experimental topology in Nitos Testbed with all the necessary components. We have already seen and discussed it in the previous chapter. To begin, in machine one, the core network is deployed, including the HSS, MME, S-GW, and P-GW components. In machine two, we run the CU and LTE DU processes utilizing a USRP B210 for the Air communication through the Uu interface between the LTE DU and the UE's LTE interface (Huawei LTE USB Dongle on machine five). The programmable attenuators are installed on the output of the SDR (USRP) to handle its attenuation based on the basic non-

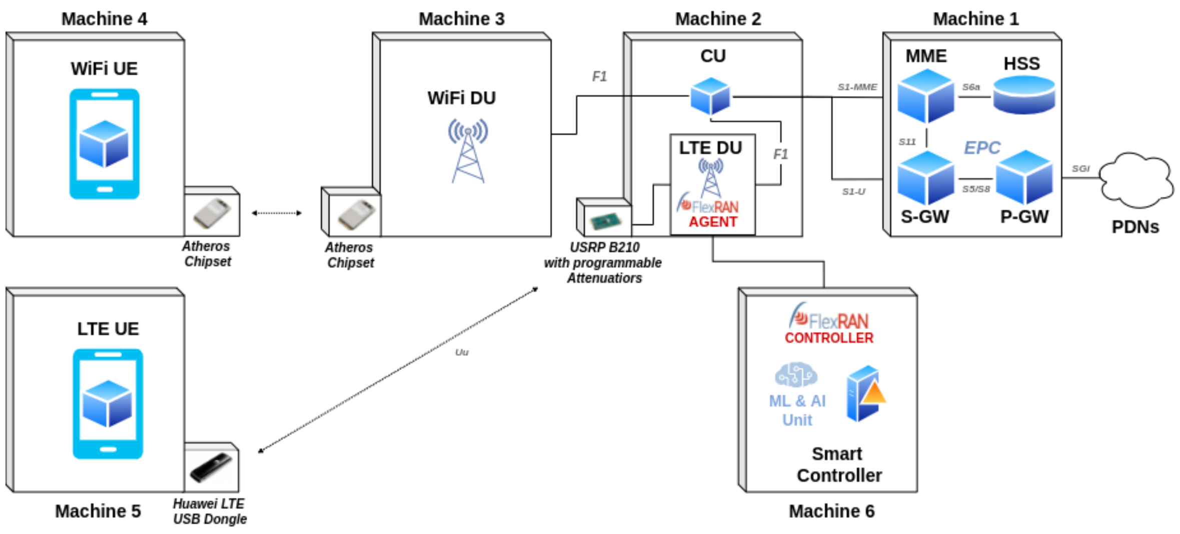


Figure 4.52: Experimental Topology utilizing six machines on NITOS Testbed

augmented car route scenario emulating, this way, a moving UE node (car route). Moreover, the WiFi DU segment exists on a separate node (machine three) communicating with the UE's WiFi interface on machine four utilizing an ad-hoc WiFi link through Qualcomm Atheros AR9380 chipsets on both sides.

Finally, we employ the Flexran controller with the ML & AI unit on machine six that incorporates the CNN-LSTM neural network for the live predictions. Based on the model's forecastings, the unit sends new, more appropriate slices in near-real-time (under 0.25 seconds) to the Flexran Controller. Subsequently, the controller communicates with the agent hosted on the LTE DU, and then the agent sends the slice to the CU based on the developed approach described in the previous chapter adjusting the DU scheduling.

## 4.5.2 Experimental Scenario

The experimental evaluation scenario is constructed to demonstrate the optimization of the network traffic as also the enhanced QoE on the UE side with the incorporation of our ML & AI unit on top of the default network configuration. We run the experiment in two phases, one with the default network behavior without the ML & AI unit and another one utilizing it to handle the real-time slicing allocation.

Generally, when we run the first phase without the ML & AI unit, we assume that there is no monitoring and predicting mechanism, and thus the traffic flows only through the LTE DU.

On the contrary, the ML & AI unit continuously monitors and reconfigures the slicing in the second phase. Specifically, we assume that there is always an optimal WiFi channel quality and that our primary goal is to send the whole traffic through the LTE link when there are good link conditions. In this way, the ML & AI unit will use the WiFi link only when the LTE CQI is predicted to be very low in order to prevent a poor network connection. Precisely, it will redirect the traffic from the LTE to the WiFi link when the forecasted CQI is under the value of 9 (16 QAM modulation). Understandably, the unit will forward the downlink traffic back to only the LTE link when the predicted CQI is equal to or more than 9.

Moreover, for network monitoring and traffic generation, we operate the iperf tool. On every phase, we send 14.5 Mbps downlink traffic from the core network to the UE and collect the monitoring statistics, including the bandwidth, the jitter, and the lost packets. Finally, we compare the results from both phases and evaluate our framework's performance.

### 4.5.3 Experimental Results

The experimental results show that the ML & AI Unit handles the slicing allocation procedure appropriately, enhancing the overall network performance and the Quality of Experience (QoE) of the UE. In Figure 4.53, we illustrate the results of both the experiment with the default slicing behavior (left figures) and the one with the employment of the ML & AI Unit (right figures). Precisely, on the default network configuration, the UE experiences on average 7.4 Mbps throughput, 1.7 ms jitter, and 41% packet loss. On the other hand, with the ML & AI integration, the UE experiences 13.3 Mbps throughput, 0.8 ms jitter, and 7% packet loss.

The significant point to observe is that the ML & AI unit predicts the first CQI drop at approximately 175 seconds. Then it reconfigures the slice in order to redirect all the downlink traffic only through the WiFi DU, which has optimal channel quality. Subsequently, it predicts at about 530 seconds that the CQI will be raised at higher values, meaning that the LTE channel quality will be improved. As a result, it reconfigures the slicing allocation in order to redirect all the traffic back to only the LTE DU.

In conclusion, the ML & AI unit is able to monitor and predict the CQI values in near-real-time with great precision. In addition, it reconfigures the DU scheduling based on a slicing allocation algorithm enhancing the network performance and the UE's QoE.

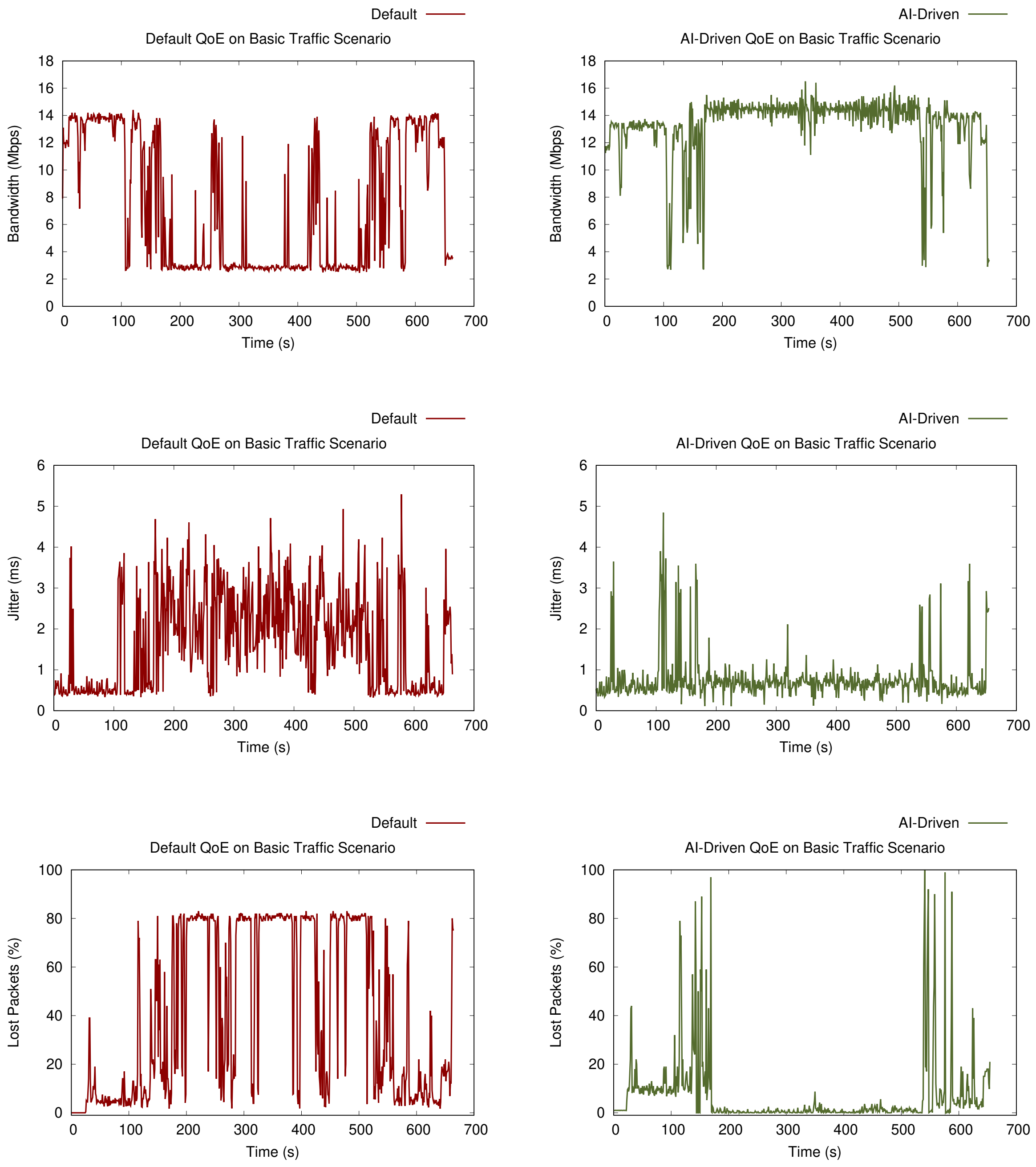


Figure 4.53: UE Experienced Bandwidth, Jitter and Packet Loss without and with the incorporation of the ML & AI Unit

# Chapter 5

# Conclusion

In this section, we summarize our work presenting the milestones, the conclusions, and the thesis contribution. Finally, we share our thoughts on extending this work in the future.

## 5.1 Summary and Conclusions

In this thesis, we studied the evolution of the cellular telecommunication networks from the very beginning with the 1G to future networks beyond 5G. Specifically, we learned about the urgent need for more powerful systems to handle the tremendous users' data demand that came with the evolution of the internet and mobile devices. Many architectures were analyzed to understand our experiments and their significance, including LTE and 5G disaggregated architecture. Moreover, we explored the nature of machine learning and artificial intelligence as also their role in the future of telecommunication networks.

Understandably, this thesis contributes to the research for the upcoming generations of cellular networks to create more flexible and adaptive AI/Data-driven systems by integrating intelligence. Precisely, we begin by developing a scheduling scheme with the CU acting as a coordinator utilizing the FlexRAN's slicing technique. On top of that, we insert an ML & AI unit on the controller's side after analyzing deeply various machine learning algorithms. This unit continuously monitors, predicts, and posts appropriate slices in real-time to optimally manage the available resources, resulting in enhanced network throughput and user's QoE.

In our analysis, we concluded that the optimal deep learning model for LTE CQI forecasting is a CNN-LSTM with extremely low MAE and training time. Subsequently, we utilized it in an emulated car mobility scenario for managing the scheduling of the downlink traffic among the WiFi and LTE DUs. The results indicate enormous efficiency resulting in enhanced user experience and network performance. Nevertheless, all deep learning and tree-based algorithms show sufficiently high precision and could be integrated into the ML & AI unit for excellent performance.

In the future, we could extend this framework by collecting more data and creating an online-training approach to update the model constantly in order to be consistent and flexible with new network patterns to come. Moreover, WiFi channel quality monitoring will be added. This way, the CU will better understand the overall network conditions and apply more sophisticated scheduling decisions by utilizing all the available slicing percentages. For instance, it will not redirect the traffic only to one DU, but it will split it through both DUs to enhance network performance even more.

# Bibliography

[1] Peltonen E., Bennis M., Capobianco M., Debbah M., Ding A., Gil-Castiñeira F., Jurmu M., Karvonen T., Kelanti M., Kliks A., Leppänen T., Lovén L., Mikkonen T., Rao A., Samarakoon S., Seppänen K., Sroka P., Tarkoma S., and Yang T. *6G WHITE PAPER ON EDGE INTELLIGENCE* [white paper]. (6g research visions, no. 8). In *http://urn.fi/urn:isbn:9789526226774*, University of Oulu, 2020.

[2] Taleb T., Aguiar R. L., Yahia I. G. B., Chatras B., Christensen G., Chunduri U., Clemm A., Costa X., Dong L., Elmirghani J., Yosuf B., Foukas X., Galis A., Giordani M., Gurtov A., Hecker A., Huang C.-W., Jacquenet C., Kellerer W., and … Zorzi M. *White Paper on 6G Networking* [white paper]. (6g research visions, no. 6). In *http://urn.fi/urn:isbn:9789526226842*, University of Oulu, 2020.

[3] Ali S., Saad W., and Steinbach D. (Eds.). . *White Paper on Machine Learning in 6G Wireless Communication Networks* [white paper]. (6g research visions, no. 7). In *http://urn.fi/ urn:isbn:9789526226736*, University of Oulu, 2020.

[4] Nikos Makris, Christos Zarafetas, Pavlos Basaras, Thanasis Korakis, Navid Nikaein, and Leandros Tassiulas. Cloud-based convergence of heterogeneous rans in 5g disaggregated architectures. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6, 2018.

[5] Opeoluwa Tosin Eluwole, Nsima Udoh, Mike Ojo, Chibuzo Okoro, and Akintayo Johnson Akinyoade. From 1g to 5g, what next? In *IAENG International Journal of Computer Science, 45:3, IJCS_45_3_06*, Aug. 2018.

[6] Christopher Cox. *An Introduction to LTE: LTE, LTE-Advanced, SAE, VoLTE and 4G Mobile Communications*. John Wiley & Sons, Ltd, second edition, 2014.

[7] 3GPP - A Global Initiative. The mobile broadband standard. [Online]. https://www.3gpp.org/. Access Date: 30-Jan-2022.

[8] Per Beming, Lars Frid, Göran Hall, Peter Malm, Thomas Noren, Magnus Olsson, and Göran Rune. Lte-sae architecture and performance. In *Ericsson Review No. 3*, 2007.

[9] Ghassan A. Abed, Mahamod Ismail, and Kasmiran Jumari. The evolution to 4g cellular systems: Architecture and key features of lte-advanced networks. In *IRACST – International Journal of Computer Networks and Wireless Communications (IJCNWC), ISSN: 2250-3501 Vol. 2, No. 1*, 43600 UKM Bangi, Selangor, Malaysia, 2012.

[10] Tutorialspoing - Learn LTE, [Online]. https://www.tutorialspoint.com/lte/lte_network_architecture.htm. Access Date: 30-Jan-2022.

[11] Ersan Kabalci and Yasin Kabalci. *Smart Grids and Their Communication Systems*. Springer Nature Singapore Pte Ltd, 2019.

[12] Line Maria Pyndt Larsen, Aleksandra Checko, and Henrik Lehrmann Christiansen. A survey of the functional splits proposed for 5g mobile crosshaul networks. *I E E E Communications Surveys and Tutorials*, 21(1):146 – 172, 2018.

[13] Nikos Makris, Christos Zarafetas, Spyros Kechagias, Thanasis Korakis, Ivan Seskar, and Leandros Tassiulas. Enabling open access to lte network components; the nitos testbed paradigm. In *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, pages 1–6, 2015.

[14] NITOS - Network Implementation Testbed using Open Source platforms. [Online]. https://nitlab.inf.uth.gr/NITlab/. Access Date: 27-Jan-2022.

[15] Navid Nikaein, Mahesh K. Marina, Saravana Manickam, Alex Dawson, Raymond Knopp, and Christian Bonnet. Openairinterface: A flexible platform for 5g research. *SIGCOMM Comput. Commun. Rev.*, 44(5):33–38, oct 2014.

[16] X. Foukas, N. Nikaein, M. M. Kassem, M. K. Marina, and K. Kontovasilis. Flexran: A flexible and programmable platform for software-defined radio access networks. In *Proceedings of the 12th ACM CoNEXT*, ACM, 2016.

[17] Tsaug - Python Package for Time Series Augmentation, [Online]. https://tsaug.readthedocs.io/en/stable/. Access Date: 28-Jan-2022.

[18] Mark Schmidt. Least squares optimization with l1norm regularization. 01 2005.

[19] The sklearn LinearRegression, [Online]. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html. Access Date: 30-Jan-2022.

[20] The sklearn Ridge, [Online]. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html. Access Date: 30-Jan-2022.

[21] The sklearn Lasso, [Online]. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html. Access Date: 30-Jan-2022.

[22] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[23] The sklearn ElasticNet, [Online]. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html. Access Date: 30-Jan-2022.

[24] Ijomah Maxwell Azubuike. Second order regression with two predictor variables centered on mean in an ill conditioned model. *International Journal of Statistics and Applications*, 9(4):101–110, 2019.

[25] Priyanka Sinha. Multivariate polynomial regression in data mining: Methodology, problems and solutions. *International Journal of Scientific and Engineering Research*, 4, 12 2013.

[26] The statsmodels SARIMAX, [Online]. https://www.statsmodels.org/dev/generated/statsmodels.tsa.statespace.sarimax.SARIMAX.html. Access Date: 30-Jan-2022.

[27] The statsmodels ARIMA, [Online]. `https://www.statsmodels.org/dev/generated/statsmodels.tsa.arima.model.ARIMA.html`. Access Date: 30-Jan-2022.

[28] Mariette Awad and Rahul Khanna. *Efficient Learning Machines Theories, Concepts, and Applications for Engineers and System Designers*. Apress, Berkeley, CA, 1 edition, 2015.

[29] The sklearn SVR, [Online]. `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html`. Access Date: 30-Jan-2022.

[30] The sklearn KNeighbors Regressor, [Online]. `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html`. Access Date: 30-Jan-2022.

[31] Lior Rokach and Oded Maimon. *Data Mining with Decision Trees Theory and Applications*. World Scientific Publishing Co. Pte. Ltd., 5 Toh Tuck Link, Singapore 596224, 2nd edition, 2014.

[32] The sklearn DecisionTree Regressor, [Online]. `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html`. Access Date: 30-Jan-2022.

[33] The sklearn Random Forest Regressor, [Online]. `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html`. Access Date: 30-Jan-2022.

[34] The sklearn Bagging Regressor, [Online]. `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingRegressor.html`. Access Date: 30-Jan-2022.

[35] Jason Brownlee. *Gradient Boosting Trees With XGBoost and scikit-learn*. Self-published, `https://machinelearningmastery.com/xgboost-with-python/`, 1.1 edition, 2018.

[36] XGBoost Documentation, [Online]. `https://xgboost.readthedocs.io/en/stable/`. Access Date: 30-Jan-2022.

[37] LightGBM Documentation, [Online]. `https://lightgbm.readthedocs.io/en/latest/`. Access Date: 30-Jan-2022.

[38] The sklearn GradientBoosting Regressor, [Online]. `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html`. Access Date: 30-Jan-2022.

[39] CatBoost is a high-performance open source library for gradient boosting on decision trees, [Online]. `https://catboost.ai/`. Access Date: 30-Jan-2022.

[40] The sklearn AdaBoost Regressor, [Online]. `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostRegressor.html`. Access Date: 30-Jan-2022.

[41] Jason Brownlee. *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in python*. Self-published, `https://machinelearningmastery.com/deep-learning-for-time-series-forecasting/`, 1.4 edition, 2018.

[42] Keras - A deep learning API written in Python, running on top of the machine learning platform TensorFlow. [Online]. `https://keras.io/about/`. Access Date: 30-Jan-2022.

[43] Jason Brownlee. *Long Short-Term Memory Networks With Python Develop Deep Learning Models for your Sequence Prediction Problems*. Self-published, `https://machinelearningmastery.com/lstms-with-python/`, 1.0 edition, 2017.