Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
College of Information Technology
Information Network Department

A Developed 5G Network Channel Quality Indicator Prediction Based On Optimized Hybrid AI Techniques

A Dissertation

Submitted to the Council of the College of Information Technology for Postgraduate Studies of University of Babylon in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Information Technology-Information Networks

By

Karrar Ibrahim Abdulameer Abbas

Supervised by

Prof.Dr. Sattar B. Sadkhan Al-Maliki

2023A.D. **1445 A.H.**

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

نَرْفَعُ دَرَجَاتٍ مَّن نَّشَاءُ ۗ
وَفَوْقَ كُلِّ ذِي عِلْمٍ عَلِيمٌ ﴿٧٦﴾

صَدَقَ اللَّهُ الْعَلِيُّ الْعَظِيمُ

سورة يوسف

Supervisor Certification

I certify that the dissertation entitled (**A developed 5G network channel quality indicator prediction based on optimized hybrid AI techniques**) was prepared under my supervision at the department of Information Networks/ College of Information Technology/ University of Babylon as partial fulfillment of the requirements of the degree of Doctor of Philosophy in Information Technology-Information Networks.

Signature:
Supervisor Name: Prof. Dr. Sattar B. Sadkhan
Date: / /2023

The Head of the Department Certification

In view of the available recommendations, I forward the thesis entitled "**A Developed 5G Network Channel Quality Indicator Prediction Based On Optimized Hybrid AI Techniques**" for debate by the examination committee.

Signature:
Asst. Prof. Dr. Alharith A.Abdullah
Head of Information Networks Department
Date: / /2023

Certification of the Examination Committee

We hereby certify that we have studied the dissertation entitled (**A Developed 5G Network Channel Quality Indicator Prediction Based On Optimized Hybrid AI Techniques**) presented by the student (**Karrar Ibrahim Abdulameer**) and examined him/her in its content and what is related to it, and that, in our opinion, it is adequate with (**Excellence**) standing as a thesis for the degree of Doctor of Philosophy in Information Technology-Information Networks.

Signature:
Name: Wesam S.bhaya
Title: Prof.Dr
Date: / / 2023
(**Chairman**)

Signature:
Name: Yahya Ali Lafta
Title: Assist.Prof
Date: / / 2023
(**Member**)

Signature:
Name: Shamam Fadhil Alwash
Title: Prof.Dr
Date: / / 2023
(**Member**)

Signature:
Name: Ahmed M.Al-Salih
Title: Assist.Prof
Date: / / 2023
(**Member**)

Signature:
Name: Saif Al-Alak
Title: Assist.Prof
Date: / / 2023
(**Member**)

Signature:
Name: Sattar B. Sadkhan
Title: Prof.Dr
Date: / / 2023
(**Member and Supervisor**)

Approved by the Dean of the College of Information Technology, University of Babylon.

Signature:
Name: Dr. Wesam S.bhaya
Title: Professor
Date: / / 2023
(**Dean of Collage of Information Technology**)

Dedication

To my beloved mother,

This journey, this achievement, and every word contained within the following pages bear the indelible imprint of your love, wisdom, and sacrifice. You are my first teacher, my enduring inspiration, and my constant ally. The challenges I faced seemed less daunting, and the victories tasted sweeter, knowing that you believed in me, even when I doubted myself.

It is with immense pride and gratitude that I dedicate this thesis to you, my beloved mother. Your love, patience, and unwavering faith in me have made this accomplishment possible. I am forever indebted to you for shaping me into the person I am today.

Thank you, Mom.

Acknowledgement

All praise be to ALLAH Almighty, who helped me complete this task successfully, and my utmost respect to His last Prophet Mohammad PBUH, and to the hero of Islam born in the heart of Kaaba, Amir-Al-Mo'mineen, Al-Imam Ali Ibn Abi Talib.

To the memory of my late father, I extend my heartfelt appreciation for the indelible mark he left on my life. Though he is no longer with us in person, his love, wisdom, and guidance continue to shape my path. His unwavering belief in my abilities has been a driving force behind the successful completion of this thesis. I am forever grateful for the values he instilled in me and the enduring inspiration he provides.

To my caring Mother, your unconditional love, sacrifice, and endless encouragement have been the cornerstone of my success. Your unwavering support during the highs and lows of this journey has given me the strength to persevere.

To my brothers and sisters, you are my closest confidants, my pillars of support, and my cheerleaders. Your belief in my abilities, your listening ear, and your unyielding encouragement have been a driving force behind my achievements. Growing up with you has been a source of joy and camaraderie that has enriched my life beyond measure.

To my esteemed supervisor **Prof.Dr. Sattar B. Sadkhan**, I express my deepest gratitude for your expert guidance, constant motivation, and constructive feedback, which have been invaluable throughout this research endeavor. Your belief in my potential and your patience have been pivotal in shaping the course of my research.

To my dear friends, thank you for being the anchor in the storm, the voice of reason, and the source of laughter throughout this journey. Your unwavering support and encouragement have made even the most challenging moments bearable.

To my professors, mentors, and colleagues, I express my thanks for engaging in stimulating discussions and providing an intellectually stimulating environment that contributed to the refinement of my ideas and expanded my horizons. Your valuable insights and thoughtful critiques have enriched the quality of this work.

Abstract

In the face of the continuous growth of mobile data traffic and the emergence of new applications beyond 5G, Massive Radio Access Networks (RANs) are expected to deliver superior performance and support many connected devices. Effectively managing these networks presents significant challenges, including efficient mobility management, accurate Channel Quality Indicator (CQI) prediction, and ensuring high Quality of Service (QoS).

This dissertation aims to address these challenges by developing an integrated hybrid system that utilizes Artificial Intelligence (AI) techniques, specifically focusing on a hybrid model that combines the strengths of Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU).

In the first stage of this research, several ML models, including "Artificial Neural Networks (ANN), Deep Neural Networks (DNN), Support Vector Machines (SVM), and LSTM," are implemented and compared for their efficiency in predicting CQI, this forms the basis for the exploration of more accurate models.

In the second stage, we developed and optimized a novel integrated hybrid LSTM-GRU system using two different hyperparameter tuning algorithms - Hyperband and Bayesian Optimization. We selected these specific models and tuning algorithms based on their potential to capture the dynamic and sequential nature of the input data, which is crucial for accurate CQI prediction in a massive RAN scenario.

The research demonstrates that the proposed hybrid LSTM-GRU system, when properly tuned and optimized, performs better predicting CQI than the other ML models. This improvement in CQI prediction accuracy translates into better network management, more efficient resource allocation, and enhanced QoS 5G Massive RANs.

In the initial stage, multiple machine learning models were assessed for CQI prediction in Massive Radio Access Networks. Among these models,

LSTM displayed the best performance with an MSE of 0.811544, RMSE of 0.900858, and MAE of 0.608555, achieving an R2 of 0.883451.

In the subsequent stage, an integrated hybrid LSTM-GRU system was introduced. This model surpassed all others, delivering exceptional results with an MSE of 0.0182, RMSE of 0.1350, MAE of 0.1197, and an impressive R2 of 0.9975. These findings unequivocally establish the hybrid LSTM-GRU system as the most accurate and robust choice for CQI prediction in the context of Massive RANs. Its potential extends to enhancing network management, optimizing resource allocation, and elevating Quality of Service (QoS) in the era of advanced wireless technologies.

Declaration Associated with this Thesis

Some of the works presented in this thesis have been published or accepted as listed below.

Published paper

1- K. Ibrahim and S. B. Sadkhan, "Radio Access Network Techniques Beyond 5G Network: A Brief Overview," *2021 International Conference on Advanced Computer Applications (ACA)*, Maysan, Iraq, 2021, pp. 96-100, doi: 10.1109/ACA52198.2021.9626804.

Accepted paper

2- K. Ibrahim and S. B. Sadkhan, “A Developed Beyond 5G Massive Radio Access Networks Based On Artificial Intelligence Techniques,” *International Conference on Information Technology, Applied Mathematics and Statistics (ICITAMS), Dewaniyah, Iraq,2023.*

List of Contents

List of Tables

List of Figures

Table of Abbreviations

Abbreviation	Description
4G	Fourth generation
5G	Fifth Generation
6G	Sixth Generation
B5G	Beyond fifth generation
ADAM	Adaptive Moment Estimation
AI	Artificial Intelligence
ANN	Artificial Neural Networks
BS	Base Station
CQI	Channel Quality Indicator
CSI	Channel State Information
DNN	Deep Neural Networks
GRU	Gated Recurrent Units
KPI	Key Performance Indicators
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MCS	Modulation and Coding Scheme
ML	Machine Learning
MSE	Mean Squared Error
QAM	Quadrature amplitude modulation
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
RAN	Radio Access Networks
RB	Resource Block

RMSE	Root Mean Squared Error
RMSProp	Root Mean Square Propagation
RNN	Recurrent Neural Network
RSRP	Reference Signal Received Power
RSRQ	Reference Signal Received Quality
RSS	Received Signal Strength
RSSI	Received Signal Strength Indicator
SGD	Stochastic Gradient Descent
SINR	Signal to Interference & Noise Ratio
SNR	Signal To Noise Ratio
SVM	Support Vector Machines
UE	User Equipment
URLLC	Ultra-Reliable and Low-Latency Communications
IoT	Internet of things
NS-3	Network Simulators - 3
BLER	Block Error Rate
NACK	Negative Acknowledgment
UAVs	Unmanned Aerial Vehicles
3GPP	The 3rd Generation Partnership Project
DL	Downlink
eNB	Evolved Node B
gNB	Next Generation Node B
LTE-A	Long Term Evolution Advanced
MAC	Media Access Control Address
PDCP	Packet Data Convergence Protocol

CHAPTER ONE

INTRODUCTION

1.1 Overview

The journey began with the first generation (1G) of mobile networks, which supported voice-only services over an analog system. Then it transitioned into the second generation (2G) that introduced digital technology, supporting voice and low-speed data services. The third generation (3G) brought about high-speed data services, enabling web browsing and other multimedia applications on mobile devices. The fourth generation (4G) further improved these services, offering higher data rates, lower latencies, and enhanced multimedia service capabilities [1].

The advent of the fifth-generation (5G) technology marked a paradigm shift characterized by the proliferation of the Internet of Things (IoT), massive device connectivity, high-speed data rates, ultra-low latency, and improved Quality of Service (QoS) [2]. This generation also witnessed the emergence of many advanced applications, such as autonomous driving, remote surgery, smart cities, and Industry 4.0, which became feasible due to the superior capabilities of 5G [3].

Despite these advancements, the growth in user requirements and the emergence of new applications pose unprecedented challenges that the existing technologies cannot efficiently address. This has necessitated the transition towards the sixth generation (6G) of mobile networks, which is expected to provide features such as sub-millisecond latencies, terabit-per-second speeds, and comprehensive AI integration [4].

As the focus shifts beyond 5G, an essential aspect is the development of Massive Radio Access Networks (RANs) [5]. Researchers expect these networks to support a huge number of devices and handle enormous data traffic while ensuring high-speed communication, ultra-reliability, and low

latency. However, the development of these networks presents several challenges, including efficient use of the radio spectrum, dynamic resource allocation, interference management, and mobility management, among others [6].

Mobility management is particularly crucial in this context. It ensures the seamless transition of user equipment (UE) between different base stations while maintaining continuous and quality communication. Given the high device density and diverse service requirements, efficient mobility management becomes even more critical in a massive RAN[7].

Another critical aspect is the Channel Quality Indicator (CQI). It provides information about the downlink channel condition, vital for efficient resource allocation and scheduling decisions. The efficient use of CQI information in a massive RAN can significantly enhance network performance and QoS [8].

Moreover, meeting the strict QoS requirements for different applications in a massive RAN is challenging. This includes ensuring high data rates for bandwidth-intensive applications, ultra-reliability for mission-critical applications, and low latencies for real-time applications [9] [10].

Integration of Artificial Intelligence (AI) techniques is considered a potential solution in light of these obstacles. AI can provide intelligent and dynamic solutions to many problems associated with developing beyond 5G massive RANs. For instance, it can be used for efficient mobility management, accurate prediction of CQI information, dynamic resource allocation, and enhanced QoS management, among others [8].

In the present research, researchers aim to develop an AI-based model for beyond 5G massive RANs. The proposed model will address the challenges associated with mobility management, efficient use of CQI information, and QoS enhancement. This study will significantly contribute to the evolution of mobile cellular communication. Researchers explore the potential AI-driven wireless communication infrastructure, addressing its challenges and proposing groundbreaking solutions. The study's results could be instrumental in shaping the future direction of wireless networking research and development, potentially transforming the global digital landscape.

To transition into an era 5G and move towards 6G, the requirements for mobile networks have become increasingly stringent and complex. These requirements encompass a broad spectrum of needs, from extremely high data rates and ultra-reliability to ubiquitous connectivity and improved Quality of Service (QoS) [2]. While the current 5G technology has made significant strides in meeting these requirements, it is yet to achieve the full breadth and depth of these needs, necessitating the development of more advanced technologies [8].

The advent of Massive Radio Access Networks (RANs) in 5G and beyond introduces a plethora of challenges. Massive RANs, characterized by a high density of connected devices, resulting in an explosion of data traffic, demanding efficient management of resources and connectivity. Ensuring seamless communication and high-quality service in such a dense and complex network environment is formidable [2].

1.2 Problem Statement

The efficient management of radio resources in the context of Massive Radio Access Networks (RANs) is of paramount importance due to the challenges posed by varying CQI reports from User Equipment (UE). An important issue in these networks is the potential mismatch between CQI reports and the actual channel conditions, leading to significant resource wastage and a reduction in Quality of Service (QoS) [11].

One critical challenge is when UEs report low CQI values despite the network sending large transport blocks. In such scenarios, it is highly probable that UEs are unable to decode the data due to CRC errors, leading to the transmission of Negative Acknowledgments (NACK) to the network. As a result, the network is compelled to retransmit the data, resulting in a wasteful consumption of radio resources [12].

Conversely, when UEs report high CQI values, even when the real channel quality is poor, a similar issue arises. The network responds by sending large transport block sizes based on the reported CQI values, only to encounter the same problem of CRC errors and NACKs. This to leads to resource inefficiencies within the network.

This discrepancy between reported CQI and actual channel quality highlights the need for a more robust and accurate CQI prediction system. The proposed integrated hybrid LSTM-GRU system aims to mitigate this issue by improving the accuracy of CQI prediction, thereby optimizing network resource utilization and enhancing QoS in the evolving landscape of mobile communications.

1.3 Research Questions

The main question in this dissertation is how we can develop an integrated system leveraging AI techniques to enhance the efficiency and performance of beyond 5G Massive Radio Access Networks. From this overarching question, we can derive several sub-questions:

- How can AI techniques be utilized for efficient mobility management 5G Massive RANs?

- How can various machine learning models such as “Artificial Neural Networks (ANN), Deep Neural Networks (DNN), Support Vector Machines (SVM), and Long Short-Term Memory (LSTM) networks” be employed for effective CQI prediction in 5G Massive RANs?

- How can a hybrid LSTM-GRU model be develop to improve the accuracy of CQI prediction in massive RANs?

- How can the Hyperband and Bayesian Optimization algorithms be utilized for practical hyperparameter tuning of the hybrid LSTM-GRU model?

By addressing these sub-questions, the research aims to answer the main question and provide a comprehensive solution for improving the efficiency and performance of beyond 5G Massive RANs using AI techniques.

1.4 The aims of the dissertation

This research aims to design an integrated hybrid LSTM-GRU system to enhance performance beyond 5G Massive Radio Access Networks. The main objectives derived from this aim include:

- Design and implement a hybrid model using LSTM and Gated Recurrent Units (GRU). The performance of this hybrid model in predicting CQI will be compared with the other AI models to determine its efficiency and effectiveness.

- Applying and comparing the efficiency of two different hyperparameter tuning algorithms, Hyperband and Bayesian Optimization, in optimizing the LSTM-GRU hybrid model will provide insights into the most effective approach to hyperparameter tuning for this specific use case.

Through these aims, the dissertation seeks to contribute to the knowledge of AI applications in the evolution of mobile networks, particularly in the context of beyond 5G Massive RANs.

1.5 Objectives of the dissertation

The overarching aim of this research is to design an integrated hybrid Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) system that can enhance the performance of beyond 5G Massive Radio Access Networks. The specific objectives derived from this aim include:

- Design and implement a hybrid model using LSTM and GRU: This hybrid model will be assessed for its capability to predict CQI, and its

performance will be compared with other AI models to evaluate its efficiency and effectiveness.

- Apply both Hyperband and Bayesian Optimization algorithms for hyperparameter tuning: The study aims to utilize both Hyperband and Bayesian Optimization for hyperparameter tuning, and the best-performing tuning approach will be selected.

1.6 Dissertation Contributions

Artificial Intelligence (AI) application in the telecommunication domain is an area of growing interest, particularly in designing and optimizing beyond 5G networks. This dissertation aims to make significant contributions to this emerging field through meticulous research and comprehensive methodology. The essential contributions of this dissertation are as follows:

- Development of a Hybrid LSTM-GRU Model: One of the critical contributions of this dissertation is the development of a hybrid LSTM-GRU model for CQI prediction. This hybrid model is a novel approach aimed at improving the accuracy of CQI prediction, which is a crucial factor in network performance.

- Hyperparameter Tuning: This dissertation contributes to the understanding of hyperparameter tuning by comparing the performance of Hyperband and Bayesian Optimization algorithms in tuning the LSTM-GRU hybrid model. This comparison offers insights into effective hyperparameter tuning methods for such complex models in telecommunications.

The outcomes of this dissertation have the potential to guide the design and implementation of AI solutions for next-generation mobile networks. Additionally, they provide a blueprint for researchers and professionals working on integrating AI techniques in telecommunications, particularly in the context of massive RANs.

1.7 Related Work

We anticipate the forthcoming generation of wireless networks will bring about an entirely new spectrum of experiences and applications. These range from enabling multisensory communications to realizing brilliant cities and facilitating communication in domains beyond terrestrial boundaries, such as in space or underwater. With the emergence of such novel use cases and applications, the focus and requirements for future wireless communication development must be flexibility, support for a massive number of applications, and energy efficiency [2].

As the integral parts of 4G and 5G standards, differential CQI and wideband CQI play significant roles. They facilitate the acquisition of essential channel state information by the base station (BS), thereby crucial for rate adaptation and scheduling. This is achieved without inundating the uplink, demonstrating its importance in network management and performance optimization [13].

1.7.1 CQI Prediction in Communication Environments

A particular study provides significant insights of predicting the CQI. In [14], the authors proposed a deep learning-based approach to predict CQI in vehicular communication. CQI measures the radio channel quality between a vehicle and a base station. The base station uses it to

select the appropriate modulation and coding scheme (MCS) for the downlink transmission. However, the CQI can vary rapidly due to the dynamic nature of vehicular communication, and this can lead to inaccurate CQI feedback, which can, in turn, lead to poor communication performance. The proposed deep learning-based approach uses a long short-term memory (LSTM) network to predict the CQI, as shown in Figure 1.1. LSTM networks are a recurrent neural network well-suited for predicting time series data. The proposed approach was trained on a dataset of CQI measurements collected from a vehicular communication testbed. The results of the experiments show that the proposed system can significantly improve the accuracy of CQI prediction. The authors of [14] provide a valuable reference for our research as it demonstrates the potential of machine learning, especially LSTM networks, in predicting CQI in communication environments. However, our research aims to build upon this work by delving deeper into the LSTM model design and parameter tuning to improve predictive performance.

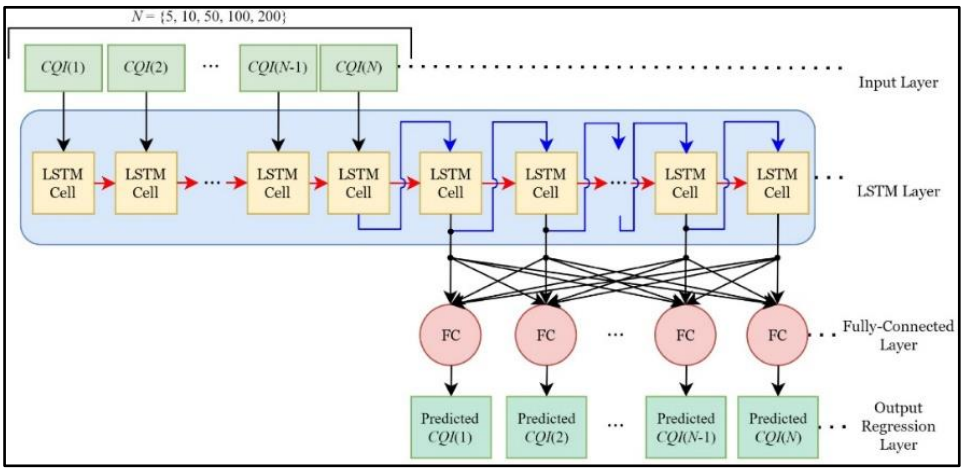

Figure 1.1: LSTM based CQI prediction model for vehicular communication [14].

A machine learning-based solution was proposed in [15] to address the outdated CQI issue caused by delayed feedback processes. The model, implemented on the UE side, considers factors such as the current SINR

and delay length to calculate an updated signal-to-interference-and-noise (SINR) that can be used as a CQI value, as shown in Figure 1.2. What sets this proposal apart is that it is a multi-variable approach that does not require any changes in signaling between the 5G base station (gNB) and the UE or overburdening of the gNB. The effectiveness of this method is proven through its mean squared error (MSE) results obtained from 5G network simulation data.

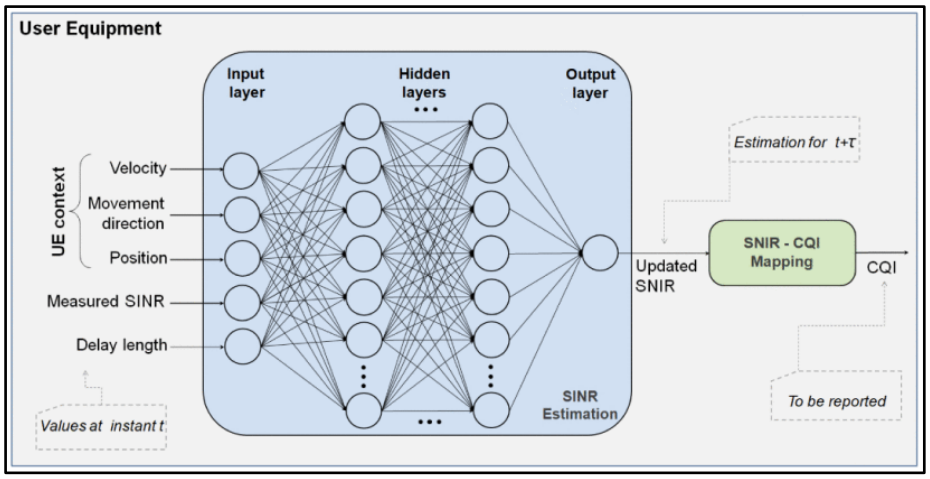

Figure 1.2: Proposed scheme for paper [15].

In the research work articulated in [16], a method predicated on Long Short-Term Memory (LSTM) was developed for CQI prediction. This method incorporated an online training module within the network simulator ns-3, addressing specifically the challenge of feedback delay. The LSTM architecture, a form of recurrent Artificial Neural Network (ANN), was utilized to enhance the precision of CQI predictions. The model operates from the perspective of the Base Station (BS), using the current CQI as an input and generating the future CQI as output, as shown in Figure 1.3. This study fortifies LSTM networks' capabilities in providing accurate CQI predictions, contributing substantial insights pertinent to our current investigation.

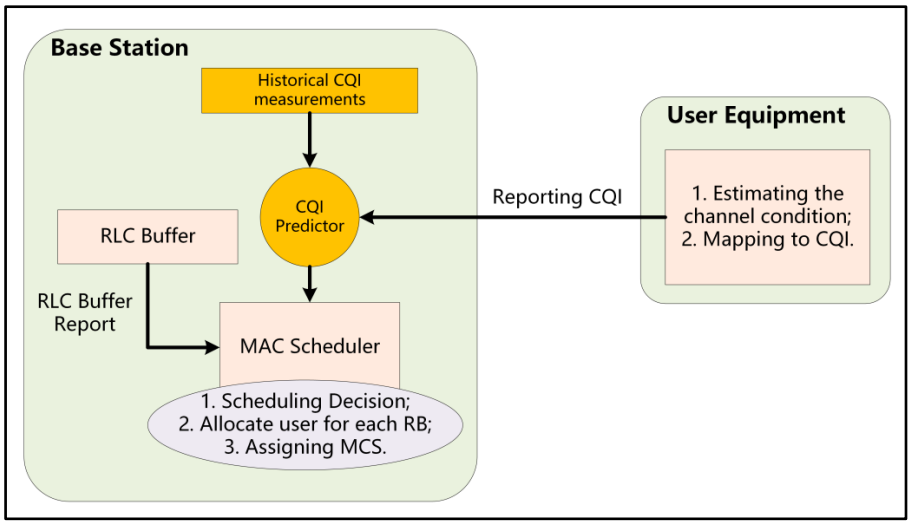

Figure 1.3: CQI prediction module [16].

The challenge of precise downlink adaptation in ultra-reliable and low-latency communications (URLLC) has been addressed by several studies. However, an innovative approach was introduced in [17], where researchers proposed enhancements to the CQI measurement and reporting procedures for 5G New Radio (NR). The goal was to accurately estimate and report the lower percentiles of the user channel quality distribution. The methodology proposed involved a simple yet efficient technique for filtering the channel quality samples, thereby improving the tail SINR performance estimate, as shown in Figure 1.4. Simultaneously, the authors proposed a new CQI reporting format aimed at better-guiding downlink scheduling and link adaptation decisions for small URLLC payloads at the gNB.

Importantly, the proposed enhancements, validated through advanced system-level simulations, demonstrated a significant reduction in latency percentile compared to existing techniques. They facilitated the downlink transmission of small and sporadic URLLC payloads with low Block Error Rate (BLER) constraints, achieving the stringent 1 ms latency

at the 99.999%, a requirement necessitated by industrial vertical applications.

The reference [17] significantly contributes to our understanding of accurate link adaptation in URLLC, thus opening new directions for our research to investigate the impact of improved CQI measurement and reporting procedures on downlink adaptation decisions.

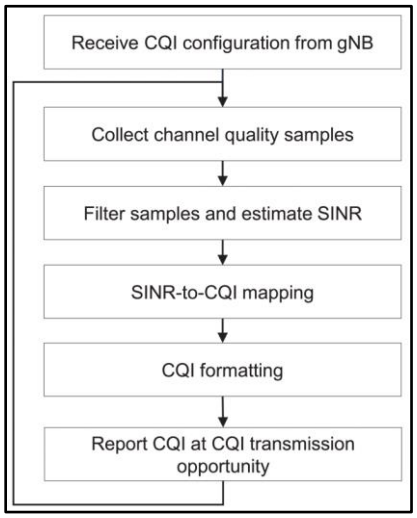

Figure 1.4: CQI measurement and reporting operation at the UE [17].

The authors in [18] present a unique perspective on the challenge of CQI estimation, a critical parameter in communication system design. The primary objective of CQI is to represent the state of the channel, enabling the base station (BS) to adjust the quality of service optimally suited for the given time and location, thus enhancing overall communication efficacy.

Traditionally, CQI has been requested from users. However, estimating it in certain instances is desirable due to the counterproductive nature of requesting CQI from all users. The authors addressed this

difficulty and proposed a solution founded on neural networks that combines the advantages of both CQI requesting and estimation [18].

The study convincingly demonstrated that the proposed neural network architecture surpasses the performance of legacy-based systems. With the ongoing trend of virtualizing radio access network (RAN) architectures and offloading substantial baseband processing to the Cloud near the base station site, the authors envision the Edge Cloud as a suitable host for their proposed neural network solution due to its substantial computational capabilities [18].

This investigation offers a compelling argument for incorporating neural networks in CQI estimation in 5G systems, a perspective that has significant implications for our research. Notably, it highlights the potential of machine learning algorithms in improving network efficiency and performance.

The authors in [19] address a pertinent issue in the rapidly evolving field of Unmanned Aerial Vehicles (UAVs) interacting with cellular networks, such as 5G and beyond, as shown in Figure 1.5. Given the increasing utility of UAVs across various applications, owing to their ease of deployment, low maintenance costs, and high mobility, the concept of a cellular-UAV system is gaining significant momentum. However, the effective implementation of this system necessitates reliable communication between the base station and the UAV, with a well-designed CQI feedback mechanism being a fundamental requisite. The traditional approach in a cellular network requires CQI feedback from users to determine an appropriate Modulation and Coding Scheme (MCS) in a multicast system. However, this method tends to increase essential

feedback as the group size grows, consequently causing more signal overhead. This problem also applies to the cellular-UAV system, necessitating an innovative solution [19].

The authors propose a novel CQI feedback mechanism to address this issue. Their approach is designed not only to reduce signaling overhead but also to enhance spectral efficiency. The efficacy of the proposed mechanism was further demonstrated by simulation results [19].

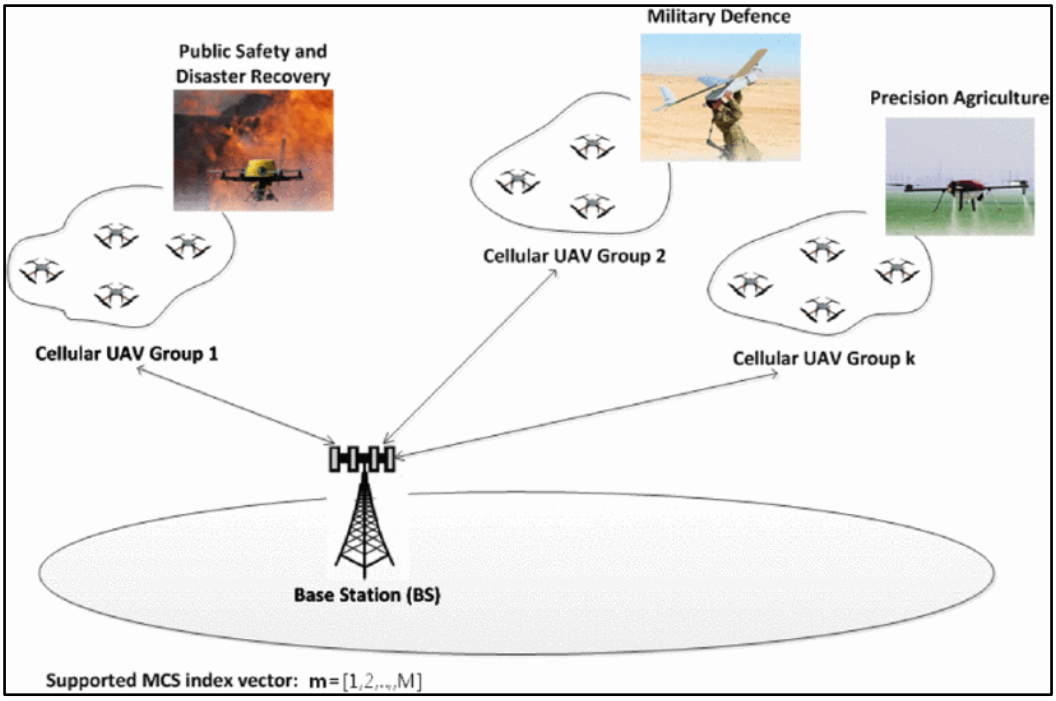

Figure 1.5: System and service model for cellular-UAV [19].

In the study by [20] the authors explored the effects of bias in CQI reporting wireless cellular networks' rate and overall performance. They identified that this bias is connected to the predicted Signal-to-Noise Ratio (SNR) estimation error, derived via the Gaussian Process Regression (GPR) method. To counter this issue, an optimized predicted SNR-CQI mapping was introduced, with an offset adaptively adjusted based on the

estimation accuracy relative to user density. Their findings indicated an enhanced system performance across different user densities.

The primary objective of [20] was to diminish the uplink CQI feedback overhead while adhering to Block Error Rate (BLER) requirements. Any inaccuracies in estimating and reporting the CQI can lead to erroneous Modulation and Coding Scheme (MCS) decisions and incorrect link adaptation, potentially compromising system performance. To mitigate these problems, [20] proposed optimizing the predicted CQI mapping through an adaptive offset, improving system performance. An analytical approximation for the optimal balance was also provided, demonstrating potential rate improvements across various user densities. These insights are integral to our investigation into effectively optimizing CQI in wireless cellular networks.

In [21], the authors propose a method to reduce the CQI feedback overhead in wireless networks by exploiting spatial correlation. CQI feedback is a critical part of the resource allocation process in wireless networks. It allows the base station (BS) to know the radio channel quality between the BS and the user equipment (UE). The BS uses this information to select the appropriate modulation and coding scheme (MCS) for the downlink transmission. However, CQI feedback can be a significant overhead in wireless networks. This is because each UE must send its CQI feedback to the BS, and the BS must then process this feedback. This can lead to a significant amount of signaling overhead, especially in networks with a large number of UEs. The proposed method in [21] exploits spatial correlation to reduce the CQI feedback overhead. Spatial correlation is the phenomenon where the channel quality between two UEs is often similar if they are close, as shown in Figure 1.6. The proposed method uses this

spatial correlation to predict the CQI of UEs that do not send CQI feedback. This prediction uses a Gaussian process regression (GPR) model. GPR is a machine learning model that can be used to predict values based on a set of known values. The proposed method's available values are the CQI feedback from a subset of UEs. Then, the GPR model is then used to predict the CQI of the remaining UEs.

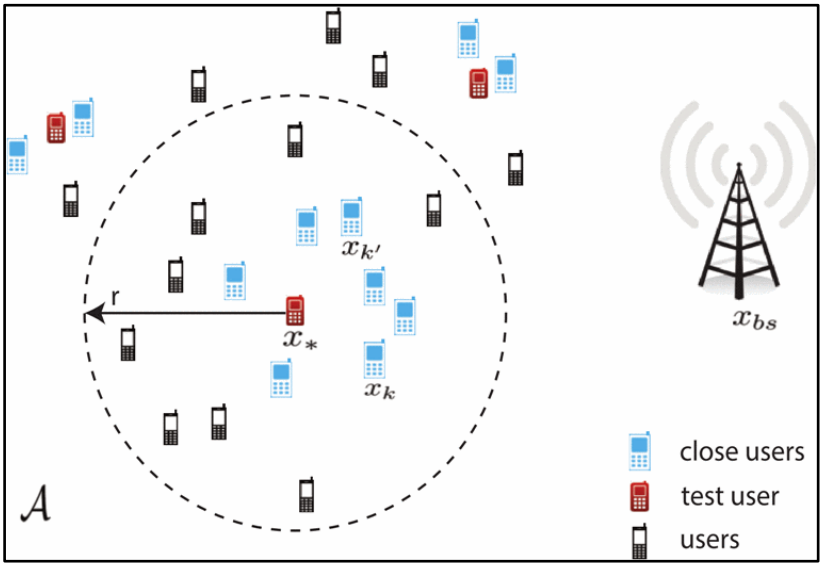

Figure 1.6: System model for paper [21].

In their research, [22] proposed a channel quality estimation methodology grounded on Gaussian process regression (GPR) for predicting users' channel states across varying user mobility profiles. They introduced a dual-control technique for determining the most suitable prediction time for each user to maintain the packet loss rate below a predefined threshold. Their approach capitalized on active learning and the exploration-exploitation paradigm, enabling the controller to choose the subsequent sampling point in time autonomously. This technique restricted the exploration of the control space while achieving optimal performance. The simulation results affirmed the method's efficacy in providing

consistent gain in terms of packet loss rate for users with low and average mobility. However, its effectiveness declined for high-velocity users [22].

Further, the dual-control technique was deployed, and its impact on users' packet loss was examined in a multicell network setting with proportional fair and maximum throughput scheduling mechanisms. Interestingly, this approach led to an overall channel quality signaling reduction exceeding 90% while maintaining packet loss below 5% with maximum throughput schedulers and a signaling reduction of 60% with proportional fair scheduling [22].

1.7.2 Hyperparameter Optimization Techniques

Deep learning has achieved impressive results on many problems. However, it requires a high degree of expertise or a lot of experience to tune hyperparameters effectively, and such manual tuning processes are prone to bias. Furthermore, deep learning differs from other machine learning scenarios in that trying out numerous hyperparameter configurations is not practical. This is primarily due to the time-consuming nature of evaluating each single hyperparameter configuration, which involves training deep neural networks, a process known for its substantial time requirements.

The Hyperband algorithm has emerged as a solution to address hyperparameter optimization challenges in the field of deep learning. It has achieved state-of-the-art performance on various hyperparameter optimization problems. However, one limitation of the Hyperband algorithm is its inability to utilize historical information from previously explored hyperparameter configurations. Consequently, the solutions it finds may be suboptimal.

In response to these challenges, recent research efforts propose a hybrid approach that combines the strengths of the Hyperband algorithm with Bayesian optimization. Bayesian optimization, unlike Hyperband, considers historical information when sampling the next trial configuration. Experimental results, as detailed in [23], demonstrate that this combined approach outperforms other hyperparameter optimization techniques, including the standalone Hyperband algorithm.

In [24], an innovative architecture is proposed, which combines a well-posed stacked sparse AutoEncoder (AE) for feature learning with a Deep Neural Network (DNN) for the classification of network traffic into benign traffic and DDoS attack traffic. The improvements suggested in [24] involve parameter tuning for AE and DNN using appropriately designed techniques to optimize DDoS attack detection, achieving low reconstruction error, mitigating issues such as exploding and vanishing gradients, and constructing a compact network design to prevent overfitting. The effectiveness of proposed approach is corroborated through a comparative analysis against ten state-of-the-art DDoS attack detection approaches.

In [25], the authors present a novel approach for hyperparameter tuning by employing multiobjective optimization for Long-Short Term Memory (LSTM) models. The technique leverages Bayesian optimization-Hyperband (BOHB) with a weighted metric scalarization method. This multiobjective optimization process optimizes the neural network model with equal consideration for two vital objective functions: the F1-score (Macro) and model training time.

To differentiate this research from prior studies, This research consider various characteristics, including input types, output, technique, and goal, which are comprehensively detailed in Table 1.1. Our proposed model is uniquely designed to operate on the gNB, thus eliminating the need for any modifications in the signaling protocol or User Equipment (UE). We employ multiple input types to predict a CQI value. Distinct from many existing studies, our research leverages a hybrid model combining LSTM and GRU networks, particularly for time-series prediction. This unique choice of model considers the input data's dynamic and sequential nature. Incorporating LSTM and GRU elements into a hybrid framework underscores the innovation in our approach and its potential for a more precise and efficient prediction of CQI.

TABLE 1.1: Approaches to Optimize CQI Prediction and Feedback in Related Studies.

Ref.	Year	Proposal	Problem Addressed	Technique Used	Objective	System Side	Input Parameters	Output
[14]	2022	A deep learning-based approach to predict CQI	Inaccurate CQI feedback and poor communication performance	LSTM	Improve the accuracy of CQI prediction	User equipment (UE)	CQI measurements collected from a vehicular communication	Predicted CQI
[15]	2021	A Machine Learning-based approach with multiple inputs for addressing CQI feedback delay	CQI Feedback delay	MLP Artificial Neural Network	To accurately estimate the SINR considering the CQI feedback delay	User Equipment (UE)	UE velocity, movement direction, position, delay length, and SINR	Predicted SINR and mapped to CQI
[16]	2020	An LSTM-based CQI prediction method with an online training module in ns-3	CQI Feedback delay	LSTM Artificial Neural Network	To enhance the accuracy of CQI prediction	Base Station (BS)	CQI	Predicted CQI
[17]	2020	A CQI reporting scheme aimed at improving Ultra-Reliable Low-Latency Communications (URLLC)	Estimation error	Worst-case estimation	To accurately estimate and report the worst-case SINR conditions	User Equipment (UE)	SINR	CQI
[18]	2020	An ANN-based selector and a Dense ANN-based method for users/subbands selection and CQI estimation	Feedback overhead	Artificial Neural Networks	To reduce the CQI feedback signalling overhead	Base Station (BS)	CQI of selected sub-bands/users	Predicted CQI

[19]	2019	A CQI feedback scheme for Unmanned Aerial Vehicle (UAV) multicast systems	Feedback overhead	Minimum function and fixed channel for CQI feedback per group	To cut down the signaling overhead and enhance spectral efficiency	Base Station (BS)	CQI	CQI
[20]	2018	A Signal-to-Noise Ratio (SNR) prediction scheme based on SNR reported by spatially correlated User equipment (UE)	Feedback overhead	Gaussian Process Regression	To minimize CQI feedback overhead and enhance prediction quality	Base Station (BS)	SNR and spatial correlation of users	Predicted SNR and CQI
[21]	2018	Spatial correlation-based CQI feedback reduction	CQI feedback overhead can be a significant problem in wireless networks	Gaussian process regression (GPR)	Reduce the CQI feedback overhead	Base station (BS)	CQI measurements from a subset of users	Predicted CQIs for the remaining users
[22]	2015	An approach to predict packet loss and CQI featuring personalized prediction windows for users	Feedback overhead	Gaussian Process Regression	To reduce packet loss and constrain the CQI signalling overhead	Base Station (BS)	Packet loss and CQI data	Predicted packet loss and CQI

1.8 Dissertation Outline

The structure of the remaining sections of the dissertation is as follows:

Chapter Two: Offers an in-depth explanation of the critical principles of Machine Learning and Artificial Intelligence, particularly as they pertain to predictive analysis and network optimization. These form the foundation for the methods employed in the research and provide context for the techniques used in CQI prediction and network optimization.

Chapter Three: This chapter is divided into two parts. The first part explains the process of implementing and comparing various machine learning models, namely Artificial Neural Networks (ANN), Deep Neural Networks (DNN), Support Vector Machines (SVM), and Long Short-Term Memory (LSTM) networks, for CQI prediction in massive RANs. The second part presents the development of a novel integrated hybrid LSTM-GRU system for the same task. It describes using two different hyperparameter tuning algorithms, Hyperband and Bayesian Optimization, in optimizing this model.

Chapter Four: Implement the proposed models and techniques on a real-world dataset from 5G RANs. It includes the experimental results and a detailed comparison of the performance of the various models and the hybrid LSTM-GRU system to predicting CQI.

Chapter Five: Compiles the conclusions drawn from this research and proposes potential directions for future work in AI applications beyond 5G networks. This chapter ties together the research findings and suggests how they may influence future developments in this rapidly evolving field.

CHAPTER TWO

THEORETICAL BACKGROUND

2.1 Overview

This chapter presents a deep dive into the theoretical underpinnings that support the structure and functionality of this research, furnishing readers with a robust understanding of the interconnected domains which drive our methodologies and findings. This chapter begins by exploring the essential principles of wireless communication, with particular emphasis on the role and importance of the CQI. Also, will outline how CQI maintains optimal communication quality and explore its mathematical modeling and prediction complexities.

Subsequently, researchers traverse the landscape of artificial intelligence and machine learning, focusing on their role in CQI prediction. Within this sphere, researchers will dissect various machine-learning models utilized in our research.

Through this chapter, researchers will lay a strong theoretical foundation, aiding in comprehending the detailed methodologies and interpretations of results that follow in the subsequent chapters of this thesis.

2.2 Wireless Communication

Wireless communication is an integral part of modern society's connectivity framework, enabling information exchange without needing any physical connection [26]. With the evolution of technology and the subsequent growth in demand for high-speed data transmission, the

efficiency and reliability of wireless communication systems have become paramount [1].

Wireless communication relies on transmitting electromagnetic waves to convey information over a distance without physical cables or wired connections. It involves the utilization of radio frequencies, microwave frequencies, or other parts of the electromagnetic spectrum to transmit and receive data. The process typically requires modulation, encoding, and multiplexing techniques to send and receive information efficiently [27].

2.3 Parameters in Wireless Communication Systems

In wireless communication systems, a multitude of parameters are employed to measure and evaluate the quality of the communication channel meticulously. These parameters provide valuable insights into crucial aspects such as signal strength, signal quality, interference levels, and noise characteristics [28]. Understanding and analyzing these parameters is paramount for optimizing system performance, efficient resource allocation, and ensuring reliable and seamless communication.

Understanding and analyzing these parameters are pivotal in wireless communication systems. They provide the foundation for optimizing network performance, facilitating reliable communication, and supporting a wide range of applications that rely on seamless and efficient wireless connectivity.

2.3.1 Channel Quality Indicator (CQI)

The CQI is a crucial factor in wireless communication systems, particularly in 4G, 5G, and beyond cellular networks, serving as a metric for the communication channel's capacity. In essence, CQI provides a quantitative measure of the channel quality, enabling the base station to optimize resource allocation based on current network conditions [29].

CQI aids in maintaining optimal communication quality by facilitating efficient modulation and coding scheme (MCS) selection, adaptive resource allocation, and advanced receiver design. It helps improve system throughput and fairness among users while reducing latency and packet loss [30]. Furthermore, the accuracy of CQI is essential in 5G and beyond networks, where the ultra-reliability and low latency requirements necessitate precise channel state information.

In a cellular network, multiple users share the available radio resources. The base station needs to allocate these resources, such as frequency bands and modulation schemes, to different users dynamically, based on their channel conditions, as shown in Figure 2.1. The CQI plays a significant role in this process, as it indicates the current channel quality for each user [31].

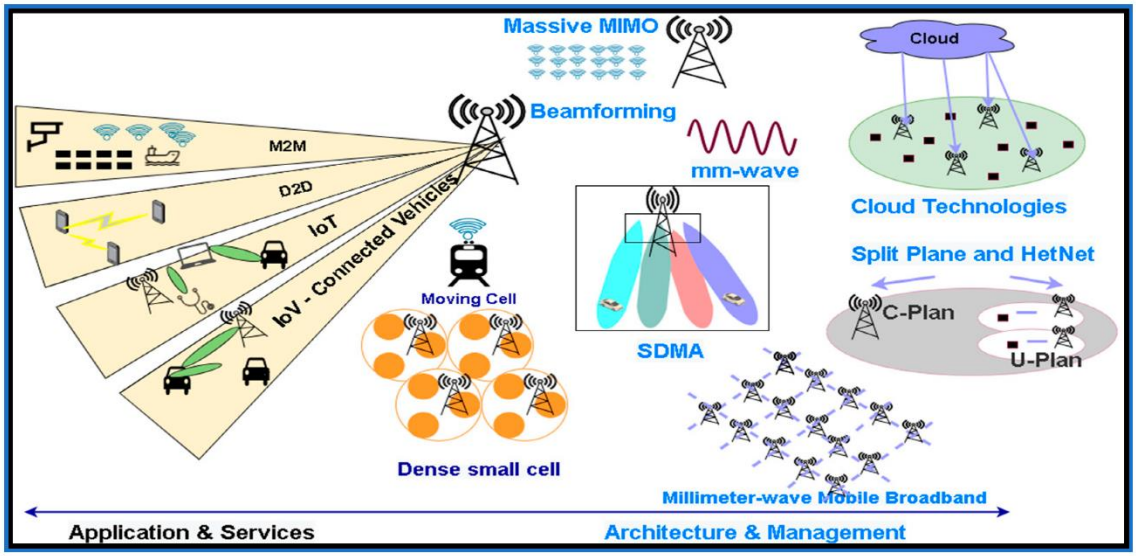

Figure 2.1: Multiple users share the available radio resources [31].

The CQI is typically calculated based on measured parameters, such as the received signal strength (RSS), the signal-to-interference-plus-noise ratio (SINR), or the signal-to-noise ratio (SNR). These parameters are obtained through channel estimation techniques, where the receiver measures the power of the received signal and estimates the interference and noise levels [32].

The calculation of CQI involves transforming the measured parameters into a discrete index representing the quality level. Different wireless communication standards and systems may have specific CQI calculation formulas. However, a common approach is to map the measured parameters to a limited number of CQI index values using look-up tables or mathematical mappings [33].

In addition to its role in adaptive modulation and coding, CQI can also be used for other purposes, such as:

- Power control: The CQI value can be used to adjust the transmit power of the transmitter. This helps to ensure that the received signal power is within the desired range, even if the channel conditions change [34].

- Resource allocation: The CQI value can be used to allocate resources to different users. For example, users with good channel quality can be given more resources, such as a larger bandwidth or a higher MCS [34].

Overall, CQI is an important parameter in wireless communication systems. It is used to indicate the quality of the radio channel and can be used for various purposes, such as adaptive modulation and coding, power control, and resource allocation [16].

Traditional MAC scheduling operated under the premise of a constant channel capacity; a notion that has since been updated for the environments of LTE-A networks. The eNB often determines the modulation and coding scheme (MCS) in these networks based on an anticipated downlink (DL) channel condition. This prediction hinges on the CQI report sent by the UE (illustrated in Figure 2.2). The 3GPP LTE-A provided a reference table indicating the efficiency of each CQI index, which can vary from 1 to 15 and depends on the modulation type (64QAM, 16QAM, QPSK). This is depicted in Table 2.1 [35].

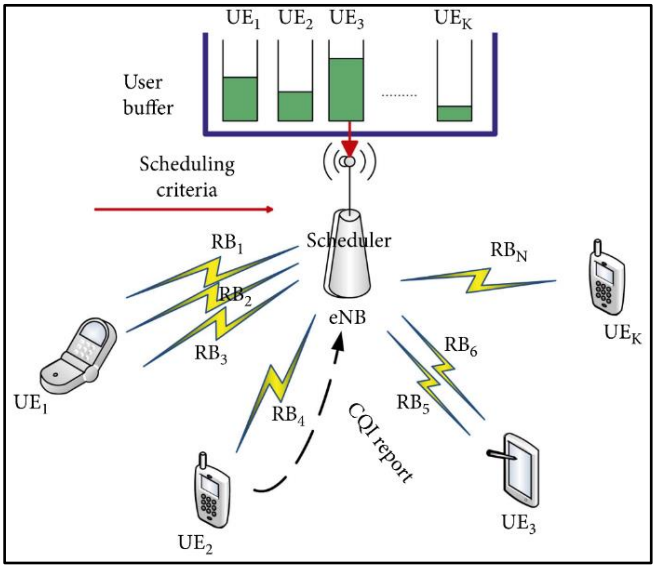

Figure 2.2: Diagram of LTE-A MAC scheduling [16].

TABLE 2.1. CQI table by 3GPP.

CQI index	Modulation	Approximate code rate	Efficiency (bits/RE)
0	No Tx	—	—
1	QPSK	0.076	0.1523
2	QPSK	0.12	0.2344
3	QPSK	0.19	0.3770
4	QPSK	0.3	0.6016
5	QPSK	0.44	0.8770
6	QPSK	0.59	1.1758
7	16QAM	0.37	1.4766
8	16QAM	0.48	1.9141
9	16QAM	0.6	2.4063
10	64QAM	0.45	2.7305
11	64QAM	0.55	3.3223
12	64QAM	0.65	3.9023
13	64QAM	0.75	4.5234
14	64QAM	0.85	5.1152
15	64QAM	0.93	5.5547

However, the CQI computation and feedback process is complex and can be influenced by various factors, including channel propagation conditions, interference levels, and terminal capabilities. These complexities can potentially affect the accuracy of the CQI, thus impeding the effectiveness of resource allocation and, subsequently, the overall network performance [36].

2.3.2 Reference Signal Received Quality (RSRQ)

RSRQ is a key performance indicator for LTE (Long Term Evolution) and newer wireless communication standards. It measures the quality of the received signal [37].

RSRQ is essentially a Reference Signal Received Power (RSRP) ratio to the Received Signal Strength Indicator (RSSI). RSRP and RSSI are essential metrics that measure signal strength but are calculated differently. RSRP measures the average power received from a single reference signal, while RSSI measures the total received wideband power, including all interference and thermal noise [38]. The formula for calculating RSRQ is as follows:

$$RSRQ = N * \left(\frac{RSRP}{RSSI} \right) \quad (2.1)\ [38]$$

In this formula, N represents the number of Resource Blocks of the E-UTRA carrier RSSI measurement bandwidth. The RSRQ measurement bandwidth is typically equal to the RSSI measurement bandwidth.

RSRQ provides a more accurate estimation of signal quality and channel conditions than the simple signal strength measurements provided by RSRP or RSSI. RSRQ is especially useful in situations where interference is a significant factor because it accounts for both the desired signal power (RSRP) and the total received power (RSSI) [37].

RSRQ is measured on a scale from -3 to -19.5 dB—the higher (i.e., less negative) the RSRQ, the better the signal quality. For example, an RSRQ of -10 dB indicates better signal quality than an RSRQ of -15 dB [39].

RSRQ is a critical measure in modern wireless communication systems, providing a comprehensive understanding of signal quality and channel conditions. It helps network operators troubleshoot and optimize their networks for better performance [40].

2.3.3 Reference Signal Received Power (RSRP)

(RSRP) is a metric used in wireless communication, particularly in technologies like Long Term Evolution (LTE), to quantify the average received power of reference signals over a specific bandwidth. In simpler terms, it measures the strength of a cell's reference signal as perceived by the user equipment (UE), such as a mobile phone [41].

RSRP is vital because it directly impacts how well a user's device can communicate with a cell tower (base station). Higher RSRP values correspond to stronger signal strength and, typically, better network performance and data speeds. Conversely, lower RSRP values indicate

weaker signal strength, which might lead to poor network performance [42].

RSRP is measured per resource element and provides the power of a single resource block (RB). A resource block is a unit of data transmission in LTE comprising several resource elements. As such, RSRP considers the power of only the specific reference signals (pilot signals) within a particular frequency bandwidth, not the total power of all signals received by the user equipment [41].

RSRP is measured in dBm (decibels relative to 1 milliwatt). In LTE networks, the RSRP is typically between -44 dBm (excellent) and -140 dBm (very poor). It's important to note that the higher the RSRP value (less negative), the better the signal strength. For example, an RSRP of -75 dBm is stronger and better than an RSRP of -95 dBm [43].

In addition to quantifying signal strength, RSRP is also used in algorithms determining when a cellular device should perform a handover from one cell tower to another. If the RSRP of the connected cell drops below the RSRP of a neighboring cell, the device may decide to disconnect from the current cell and connect to the neighbor [41].

RSRP is a fundamental metric in wireless networks that estimates signal strength, influences the performance of user devices, and impacts critical network decisions, such as handovers [43].

Wireless communication systems rely on several key parameters to evaluate the channel conditions and make informed decisions. The

Received Signal Strength (RSS) measures the power level of the received signals at the receiver. It indicates the signal's intensity and can be affected by distance, obstacles, and interference. Another significant parameter is the Signal-to-Interference-plus-Noise Ratio (SINR), which represents the ratio of the desired signal power to the combined interference and noise power. SINR quantifies the signal quality and provides insights into the system's ability to differentiate the desired signal from other sources of interference. The Signal-to-Noise Ratio (SNR) focuses solely on the ratio of the signal power to the noise power without considering interference from other signals [37].

2.3.4 Splitting Dataset

In the context of neural networks, partitioning the dataset into training and testing subsets is essential for several key reasons:

- Prevent overfitting: Dividing the dataset into separate subsets helps to avoid overfitting by ensuring that the model learns the underlying temporal patterns and dependencies in the training data rather than merely memorizing it [44]. This practice enables the model to generalize to previously unseen data, improving its predictive performance.

- Evaluate model performance: The testing subset offers an impartial means for assessing the model performance. An accurate estimation of its real-world applicability can be determined by examining the model's predictions on data not encountered during the training phase[45].

- Ensure robustness: Segregating the dataset helps verify the model's robustness and ability to adapt to various data distributions. A model that performs well on the training and testing subsets is considered more reliable and less susceptible to fluctuations in data patterns [46].

2.4 Machine Learning Models

In the era of big data, machine learning has emerged as a powerful tool capable of learning from vast amounts of information and making predictions or decisions without being explicitly programmed to perform the task. Machine learning models can identify patterns and learn from data, enabling researchers to make better data-driven decisions. In wireless communication, machine learning models are increasingly being employed for various tasks such as network optimization, resource allocation, and quality prediction [47].

Throughout the progression of this research, a variety of machine learning models have been employed, harnessing their unique predictive capabilities. These models include traditional Artificial Neural Networks (ANNs), Deep Neural Networks (DNNs), and Support Vector Machines (SVMs). Modern models, such as the Long Short-Term Memory (LSTM) networks, have also been utilized. A hybrid model that combines the strengths of LSTM with Gated Recurrent Units (GRUs) was also developed and tested. These models were selected and implemented to understand their unique characteristics and establish the most effective models under different circumstances and for diverse datasets.

2.4.1 Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) are the cornerstone for many modern machine learning applications. Inspired by the biological neural networks in human brains, as shown in Figure 2.3, ANNs represent a class of deep learning models designed to simulate how humans learn from observational data [48].

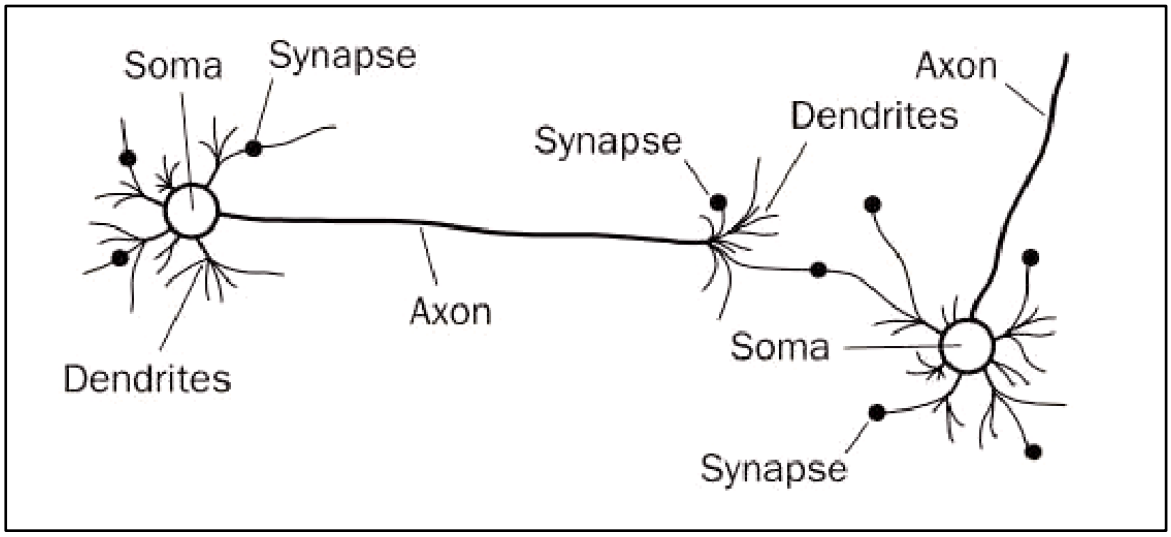

Figure.2.3: Biological neural network [48].

The premise behind the development of ANNs is the emulation of the human brain's powerful computing capability. Our brains comprise a complex network of interconnected neurons that work harmoniously to process information and generate responses. ANNs attempt to mirror this complexity and efficiency, transforming it into a computational model capable of tackling intricate problems that conventional algorithm-based computing cannot solve effectively [49].

An ANN consists of a multitude of layers, including an input layer, one or multiple hidden layers, and an output layer, as shown in Figure 2.4. Each layer comprises numerous interconnected nodes, often referred to as neurons or units. These neurons are the fundamental processing elements of the network, capable of receiving input, processing it, and transmitting it to other neurons within the network [50].

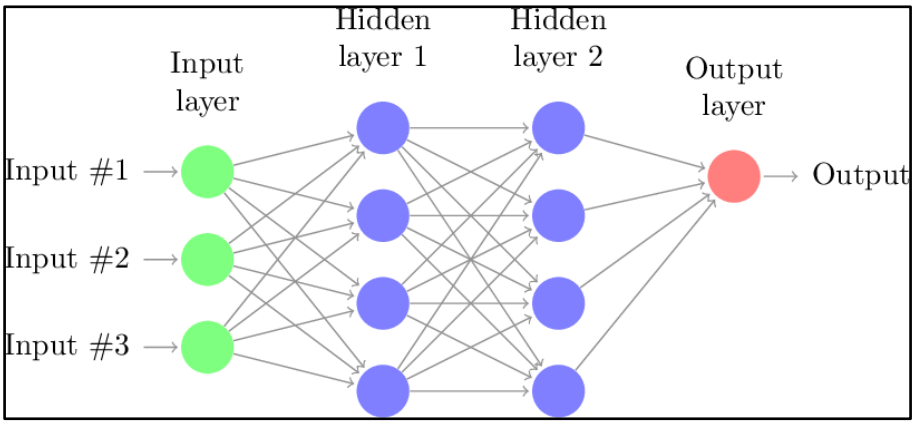

Figure 2.4: Layers in a neural network [50].

ANNs are trained using a process called backpropagation and an optimization algorithm like gradient descent. During training, the initial weights and biases are set randomly. Then the model makes a prediction using the input data. The prediction error is calculated using a loss function, which the model aims to minimize [51].

Backpropagation is used to calculate the gradient of the loss function concerning the weights and biases. Then, gradient descent, or a more sophisticated optimization algorithm, uses this information to adjust the weights and biases to minimize the loss. This training process is performed

iteratively over many epochs (an epoch is one pass through the entire dataset), gradually improving the accuracy of the model's predictions [51].

2.4.2 Deep Neural Networks (DNNs)

As we delve further into the realms of machine learning models, we encounter Deep Neural Networks (DNNs), a more sophisticated and capable variant of the fundamental Artificial Neural Networks (ANNs). DNNs are an evolution of ANNs, developed to tackle more complex problems and learn from more extensive and diverse datasets [52].

Deep Neural Networks, as the name implies, are "deep," meaning they consist of many layers of artificial neurons, also known as nodes. This depth enables DNNs to learn and represent more complex features from the input data than their shallower counterparts. This ability to recognize intricate patterns and subtle relationships in the dataets DNNs apart and makes them especially valuable in tasks where complexity and scalability are crucial [53].

Much like an ANN, a DNN consists of an input layer, multiple hidden layers, and an output layer. However, what sets DNNs apart is the number of hidden layers they contain. While an ANN might include one or two hidden layers, a DNN might have tens or hundreds of hidden layers, as shown in Figure 2.5 [53].

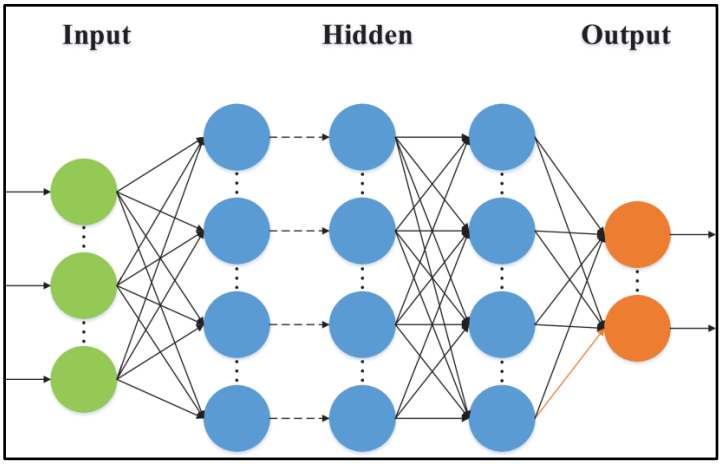

Figure 2.5: Architecture of a DNN [53].

The architecture of DNNs is flexible and can be adapted to suit the specific problem at hand. The number of layers, the number of nodes in each layer, the type of activation function, and many other design choices are part of defining the architecture of a DNN [54].

2.4.3 Support Vector Machines (SVMs)

Support Vector Machines (SVMs) are one of the most robust and conceptually sophisticated machine learning algorithms. A popular choice for classification and regression tasks, SVMs were developed based on statistical learning theory in the 1990s and have since been used in numerous real-world applications [48].

SVMs are particularly favored for their ability to handle high-dimensional data and their versatility in modeling various types of decision boundaries, linear or non-linear. SVMs perform exceptionally well when the problem involves complex but small or medium-sized datasets [55].

The architecture of the SVM does not resemble the layered structure the researchers see in neural networks. Instead, SVMs are fundamentally based on the concept of maximizing the margin to optimally separate the data into different classes [56].

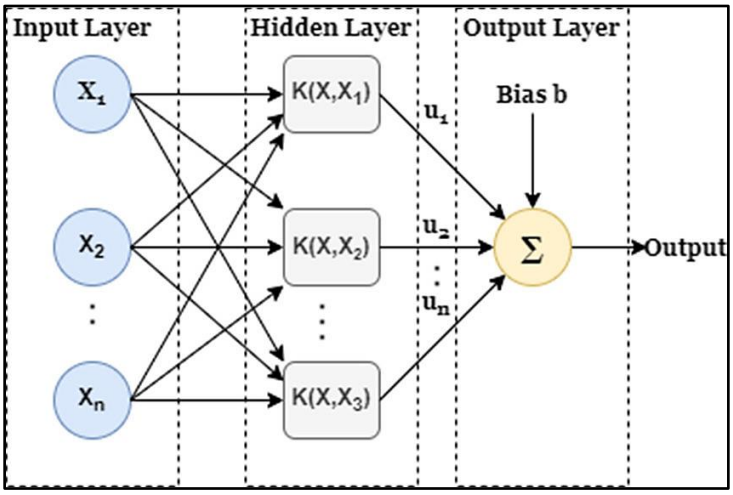

Figure 2.6: Architecture of SVM [56].

Although SVMs were initially designed for linear classification tasks, their true power lies in their ability to handle non-linear classification tasks as well. This is achieved using a technique known as the "kernel trick" [57].

The kernel trick involves mapping the input data to a higher-dimensional space where the data becomes linearly separable. For example, consider a scenario where you have two classes of data points that form concentric circles. In the original 2D space, there's no straight line (a linear boundary) that can separate these two classes. But if we map these 2D data points to a 3D space, they can be separated linearly. Various kernel functions can facilitate this transformation to the higher-

dimensional space, including linear, polynomial, Radial Basis Function (RBF), and sigmoid kernels. Each kernel has its characteristics and use cases [58].

Due to their robustness and versatility, SVMs have found widespread applications across various domains. They've been applied in handwriting recognition [59], image classification [60], sentiment analysis [61], and numerous bioinformatics applications, among others.

Despite being a relatively old algorithm in the machine-learning world, SVMs are far from obsolete. The demand for robust and high-performing algorithms like SVMs remains high with the ever-increasing dimensionality of data in various fields. Furthermore, their compatibility with other machine learning techniques, such as ensemble and deep learning, opens up new avenues for future research and applications. Their use in hybrid models, combined with techniques such as neural networks, can potentially lead to even more powerful and versatile models [62].

2.4.4 Random Forest (RF)

Random Forest is a powerful ensemble learning method that provides solutions to both classification and regression problems. Developed by Leo Breiman in 2001, Random Forest combines numerous decision trees to generate more robust predictions and prevent overfitting, a common issue with single decision tree models. The algorithm leverages the power of "the wisdom of the crowd," where the collective decision

derived from a multitude of models (trees) tends to be more accurate and reliable than that from a single model [63].

Random Forest handles high dimensional spaces efficiently and maintains accuracy even when a large proportion of the data are missing. Furthermore, Random Forest can model non-linear decision boundaries thanks to the flexibility of its base learners and decision trees. It can rank the importance of variables, which is beneficial for interpreting and understanding the underlying data structure [64].

The architecture of Random Forest is quite different from the layered structure found in neural networks and more complex than traditional single decision tree models. A Random Forest model consists of a multitude of decision trees, each constructed independently from a random subset of the data, and each node in each tree is split using the best among a subset of predictors randomly chosen at that node, as shown in Figure 2.7. [65] .

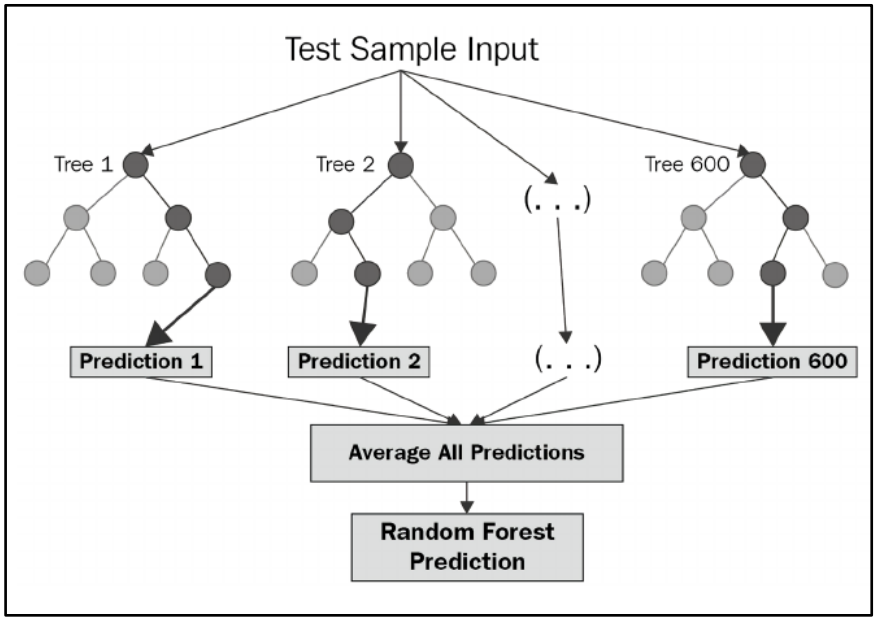

Figure 2.7: Random forest structure [65].

In essence, Random Forest combines the decisions from multiple decision trees to make the final decision. The different trees are supposed to capture various aspects of the data due to the randomness introduced during their construction. When their predictions are aggregated, the collective decision is expected to provide a more comprehensive understanding of the data and, thus, a more accurate prediction [63].

Random Forest's versatility lends itself to numerous applications across various fields. It's used in healthcare for disease detection and diagnosis [66], in finance for predicting stock behaviors and detecting fraudulent transactions [67], in bioinformatics for gene selection and classification [68], and in image processing for object and facial recognition [69].

2.4.5 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) introduced by Sepp Hochreiter and Jürgen Schmidhuber in 1997. The LSTM was designed to overcome the limitations of traditional RNNs, particularly the problem of long-term dependencies and vanishing or exploding gradients [70].

RNN with LSTM have emerged as an effective and scalable model for several learning problems related to sequential data. Earlier methods for attacking these problems have either been tailored toward a specific issue or did not scale to long time dependences. LSTMs, on the other hand, are both general and effective at capturing long-term temporal dependencies [71]. They do not suffer from the optimization hurdles that plague simple recurrent networks (SRNs) and have been used to advance the state-of-the-art for many complex problems. This includes handwriting recognition [72] and generation [73], speech synthesis [74], protein secondary structure prediction, analysis of audio [75], and video data [76] among others.

LSTM models are employed for prediction tasks due to their ability to learn and model long-range dependencies in sequential data effectively. As a RNN type, LSTM models possess unique architectural features that enable them to overcome the limitations of traditional RNNs, particularly in addressing the vanishing gradient problem. This issue arises when gradients in RNNs become too small as they are backpropagated through time, leading to difficulties in learning long-range dependencies. LSTM

models were specifically designed to address this challenge, making them well-suited for a wide range of prediction tasks involving time series or sequential data. The critical advantage of LSTM models lies in their memory cells and gating mechanisms, which allow them to selectively retain or forget information based on the input data. This capability enables LSTM models to maintain a more extended context, capturing complex temporal dependencies and relationships that other models, such as traditional RNNs and feedforward neural networks, might struggle to learn. As a result, LSTM models can achieve superior performance in tasks that require the modeling of long-range dependencies, including time series forecasting [77].

2.4.5.1 LSTM Model Architecture

RNNs have significant issues when there are large gaps between the relevant information in a sequence and the point where it is needed. This is known as the long-term dependency problem, where the network fails to learn to connect the information. In theory, RNNs can handle "long-term dependencies" but fail to remember them [78].

The LSTM architecture introduces a concept called "gates" to address these issues. Each LSTM cell contains three gates: input, forget, and output. The cell uses these gates to control the flow of information, deciding what to keep and discard. The architecture of these cells enables the LSTM to learn from data with long time-effective lags between relevant events [70].

Each LSTM cell consists of a unique internal structure containing input, forget, and output gates and a cell state, as shown in Figure 2.8. These components work together to control the flow of information through the LSTM cell, enabling it to learn and retain long-range dependencies in sequential data [79].

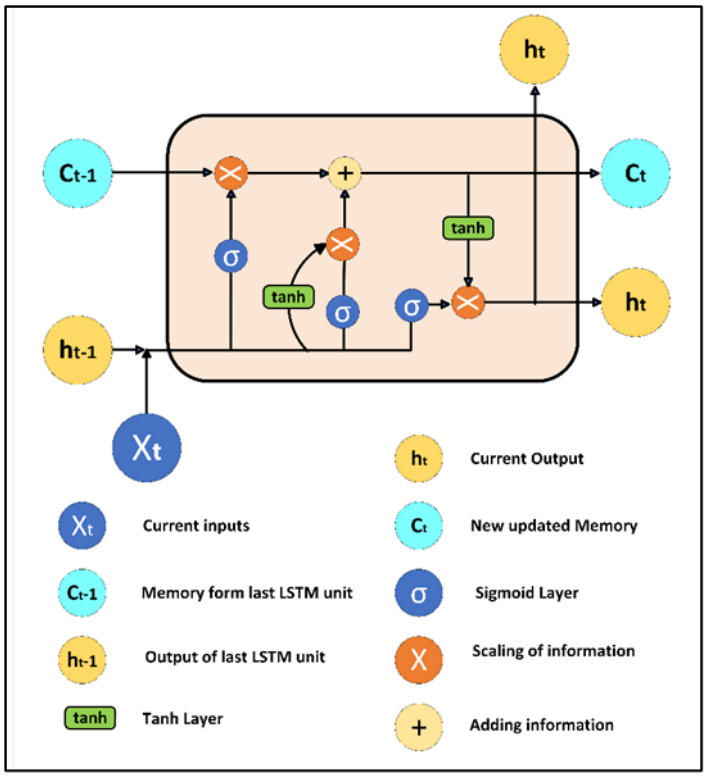

Figure 2.8: Components of an LSTM cell [79].

LSTMs use a gating mechanism that controls the memoizing process. Information in LSTMs can be stored, written, or read via gates that open and close, as shown in Figure 2.9. These gates keep the memory in the analog format, implementing element-wise multiplication by sigmoid ranges between 0-1. Analog, being differentiable, is suitable for backpropagation [80].

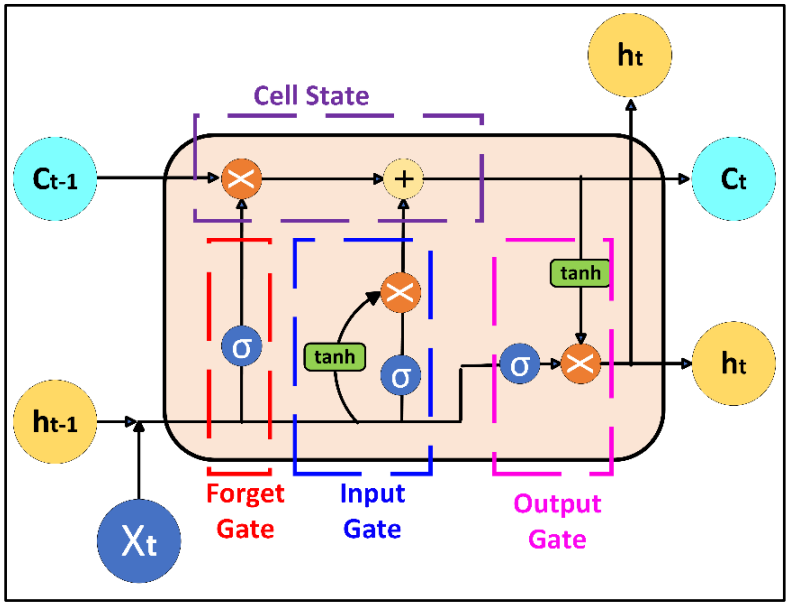

Figure 2.9: Gates of an LSTM cell [80].

The main components of an LSTM cell as shown in Figure 2.8. They:

A. Input gate (i_t): The input gate determines how much of the new input will be considered for updating the cell state, as shown in Figure 2.9. It uses the sigmoid activation function (σ), which squashes the output between 0 and 1, to control the amount of information flow. The input i_t at each time step t will be a vector containing input values. The LSTM cell will process this input to get the output [70]. The input gate equation:

$$i_t = \sigma\left(w_i * [h_{t-1}, x_t] + b_i\right) \quad (2.2)\ [70]$$

Where w_i and b_i are weight matrices and bias for the input gate, respectively, h_{t-1} is the hidden state from the previous time step, and x_t is the input at the current time step.

B. Forget gate (f_t): The forget gate controls how much of the previous cell state (c_{t-1}) should be forgotten. It also uses the sigmoid activation function, as shown in Figure 2.9. To determine the amount of information to be retained or discarded. This is particularly useful in cases where the input data is sequential, and the network needs to learn when to forget old or irrelevant information [81]. The first operation for forget gate is Concatenate the previous hidden state h_{t-1} and the current input x_t as shown in Figure 2.10. The second operation is to multiply a concatenated vector by the forget gate's weight matrix w_f and add the forget gate's bias b_f. Then the sigmoid function to the result, producing the forget gate output f_t. The sigmoid function maps the result to a value between 0 and 1, representing the degree to which each component of the cell state should be forgotten (0 means entirely forget, 1 means retain) [82]. The forget gate equation is:

$$f_t = \sigma\left(w_f * [h_{t-1}, x_t] + b_f \right) \quad (2.3)\ [82]$$

During the training process, the LSTM will learn the appropriate values for the forget gate's weight matrix w_f and bias b_f, allowing it to effectively determine which information to keep or discard from the cell state.

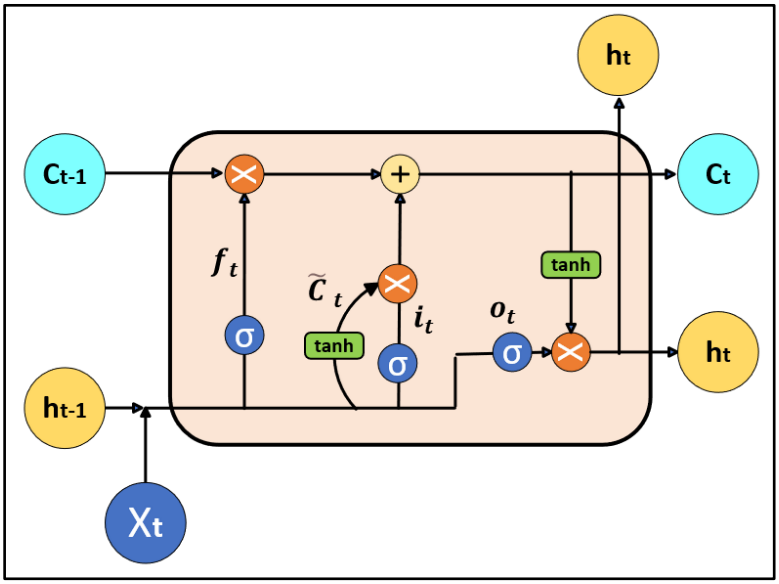

Figure 2.10: Gates operations for LSTM cell

C. Candidate cell state ($\tilde{C}_t$): The candidate cell state is computed using the tanh activation function, which squashes the output between -1 and 1. It's a temporary cell state that combines the new input with the previous hidden state, as shown in Figure 2.9 [83]. The candidate cell state is part of the LSTM responsible for computing potential new information to be added to the cell state based on the current input features.

The first operation for the candidate cell state is to concatenate the previous hidden state h_{t-1} and the current input x_t, then Multiply this concatenated vector by the candidate cell state's weight matrix w_c and add the candidate cell state's bias b_c. In the end, apply the hyperbolic tangent (tanh) activation function to the result, producing the candidate cell state $\tilde{C}$. The tanh function maps the result to a value

between -1 and 1, representing the potential new information to be added to the cell state [83]. The candidate cell state equation is:

$$\tilde{C}_t = tanh\,(w_c * [h_{t-1}, x_t] + b_c) \quad (2.4)\ [83]$$

During the training process, the LSTM will learn the appropriate values for the candidate cell state's weight matrix w_c and bias b_c, allowing it to generate potential new information based on the input features effectively. This candidate cell state is then combined with the previous cell state, modulated by the forget gate and input gate, to create the updated cell state for the current time step [83].

D. Update cell state (c_t): The update cell state step in an LSTM is responsible for updating the cell state c_t by combining the previous cell state c_{t-1}, the output of the forget gate f_t, the candidate's cell state $\tilde{C}_t$, and the output of the input gate i_t as shown in Figure 2.9. This process allows the LSTM to maintain long-term dependencies and learn important features from the input data [83]. We get updated cell state by following these steps [84]:

1. The first step is computing the forget gate output f_t for the current time step. This determines which information from the previous cell state c_{t-1} should be forgotten.

2. The second step is computing the input gate output i_t for the current time step. This determines which information from the candidate cell state $\tilde{C}_t$ should be added.

3. The third step is computing the candidate cell state $\tilde{C}_t$ for the current time step. This represents the potential new information based on the input features at time step t.

4. Then multiply the forget gate output f_t by the previous cell state c_{t-1}.

5. Then multiply the input gate output i_t by the candidate's cell state $\tilde{C}_t$.

6. Last step is adding the results of steps 4 and 5 to obtain the updated cell state c_t.

   The update cell state equation is:

$$c_t = f_t * c_{t-1} + i_t * \tilde{C}_t \quad (2.5)\ [84]$$

During training, the LSTM will learn the appropriate values for the forget gate, input gate, and candidate cell state matrices and biases, allowing it to update the cell state based on the input features effectively. The output gate then uses this updated cell state to generate the hidden state h_t, which can be used to make predictions for the output values.

E. Output gate (o_t): The output gate in an LSTM is responsible for generating the hidden state h_t based on the updated cell state c_t and the input features x_t. It uses the sigmoid activation function to control the information flow [85]. We get the output by following these steps [83]:

1. The first step is concatenate the previous hidden state h_{t-1} and the current input x_t.

2. The second step is multiplying this concatenated vector by the output gate's weight matrix w_o and add the output gate's bias b_o.

3. The third step is applying the sigmoid function to the result, producing the output gate value o_t. The sigmoid function maps the result to a value between 0 and 1, representing the degree to which each component of the cell state should contribute to the hidden state.

4. Then apply the hyperbolic tangent (tanh) activation function to the updated cell state c_t from the previous step. This maps the cell state values to a range between -1 and 1.

5. Last step is multiplying the output gate value o_t. by the result of step 4, obtaining the hidden state h_t for the current time step as shown in Figure 2.11. The output gate equation is:

$$o_t = \sigma\left(w_o * [h_{t-1}, x_t] + b_o\right) \quad (2.6)\ [83]$$

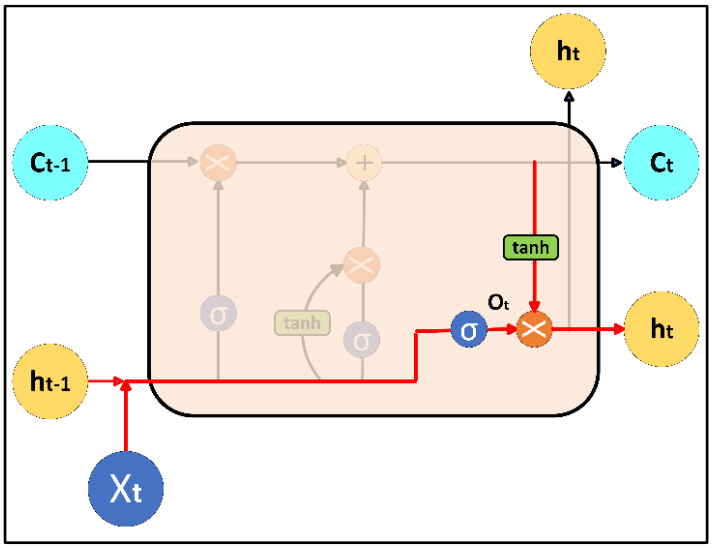

Figure 2.11: Output gate operations for LSTM cell [86].

F. Hidden state (h_t): The hidden state in an LSTM represents the network's current context or "memory" based on the input sequence it has seen. It makes predictions or generates outputs at each time step [87]. The hidden state at time step t is generated based on the updated cell state c_t and the output of the output gate o_t. This hidden state is then used to make predictions for output values. The hidden state is calculated using these steps [83]:

1. The first step is computing the updated cell state c_t by combining the previous cell state, the output of the forget gate, the candidate cell state, and the output of the input gate.

2. The second step computes the output gate value o_t for the current time step. This determines how much of the cell state should contribute to the hidden state.

3. The third step is applying the hyperbolic tangent (tanh) activation function to the updated cell state c_t. This maps the cell state values to a range between -1 and 1.

4. Last step is multiplying the output gate value o_t by the result of step 3, obtaining the hidden state h_t for the current time step.

   The hidden state equation is:

$$h_t = o_t * tanh\ (c_t) \quad (2.7)\ [83]$$

During training, the LSTM will learn the appropriate values for the forget gate, input gate, and candidate cell state matrices and biases, allowing it to update the cell state based on the input features effectively. The output gate then uses this updated cell state to generate the hidden state h_t, which can be used to predict the output values [88].

2.4.6 Gated Recurrent Units (GRUs)

Gated Recurrent Units (GRUs) are a recurrent neural network architecture introduced by Kyunghyun Cho et al. in 2014 as a simplified alternative to Long Short-Term Memory (LSTM) networks. Like LSTMs, GRUs are designed to have more persistent memory, thereby solving the vanishing gradient problem, a significant issue with traditional recurrent neural networks (RNNs). They achieve this by utilizing gating units that modulate the flow of information within the hidden state of the GRU without relying on separate memory cells [89].

The primary advantage of GRUs over LSTMs is their simplicity. They combine the forget and input gates of an LSTM into a single "update gate" and merge the cell state and the hidden state, as shown in Figure 2.12, thereby reducing the complexity of the model. This makes the GRU architecture faster to compute and easier to modify, albeit at the cost of losing some expressive power. Despite this trade-off, GRUs have been shown to perform comparably to LSTMs on various tasks, making them a valuable tool in deep learning [90].

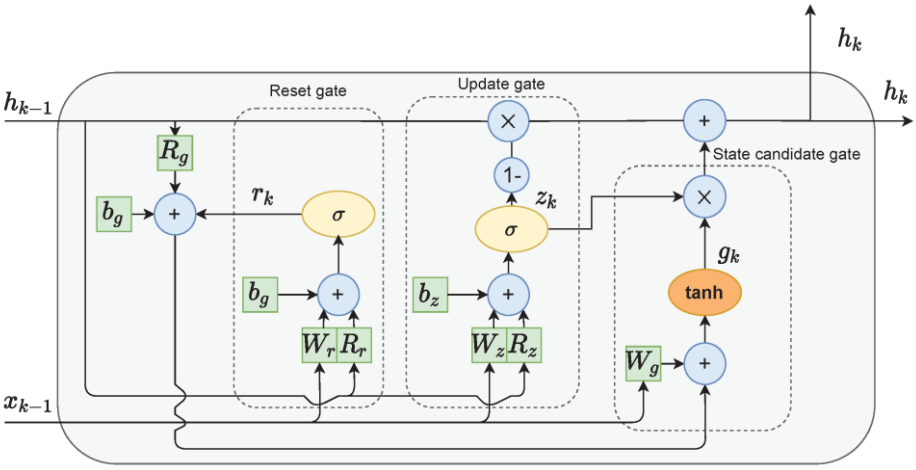

Figure 2.12: The GRU cell structure [90].

2.4.6.1 Operations of GRU

GRUs were designed to tackle the vanishing gradient problem in Recurrent Neural Networks (RNNs) without all the complexity of the Long Short-Term Memory (LSTM) cells. We will delve into the detailed operations of a GRU cell, showcasing how it uses gating mechanisms to control and manage information flow effectively.

GRUs simplify the LSTM architecture by using two gates - the reset and update gates. Both these gates help the GRU to decide what information should be passed to the output, making them crucial for the functioning of the GRU. The reset gate helps the GRU determine how much past information needs to be forgotten, while the update gate helps the model decide how much of the current state should be considered [91].

With fewer tensor operations than LSTMs, GRUs have gained popularity in tasks requiring less computational complexity. Despite having fewer gates, as shown in Figure 2.13, GRUs have shown comparable performance to LSTMs in several tasks, especially when the datasets are not extremely large [92].

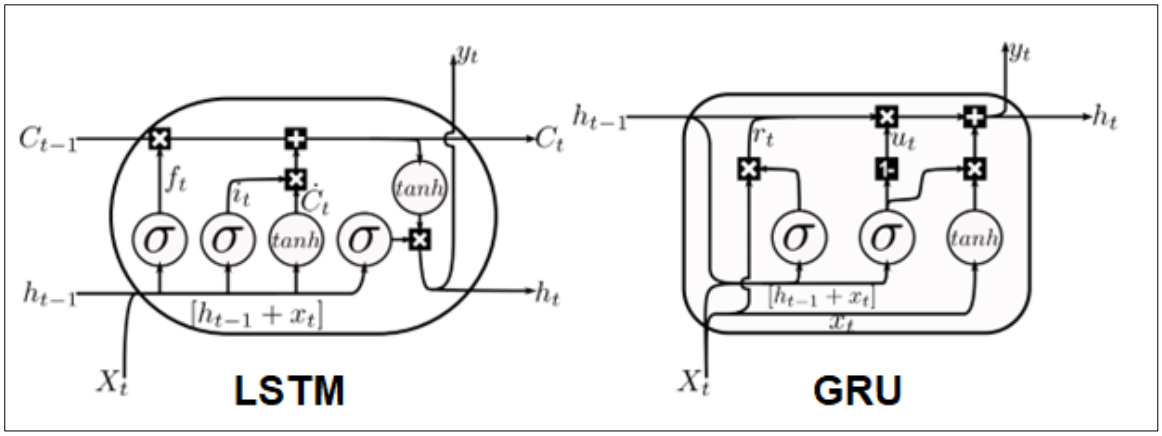

Figure 2.13: General structure of LSTM and GRU units [92].

At each time step, the GRU takes as input the current input vector (x_t) and the hidden state from the previous time step (h_{t-1}) and computes the output as follows:

A. Update gate (Z_t): The update gate determines how much of the previous hidden state to keep and how much new information to add, as shown in Figure 2.14. It is calculated as follows:

$$Z_t = \sigma\left(W^z x_t + U^z h_{t-1} + b^z\right) \quad (2.8)\ [91]$$

The update gate plays a pivotal role in determining how much information from the past h_{t-1} should be carried forward to the next state. It uses the current input x_t and the previous hidden state h_{t-1} and processes them with distinct weight matrices (W^z and U^z, respectively) and a bias term b^z. The result passes through a sigmoid function (σ), ensuring the output between 0 and 1 [93].

Each element in the update gate output can be interpreted as a probability indicating how much of the corresponding element in the previous hidden state will be kept for the next hidden state. A value close to 1 indicates that the GRU cell wants to keep more of that element, while a value close to 0 means it wants to discard more[93].

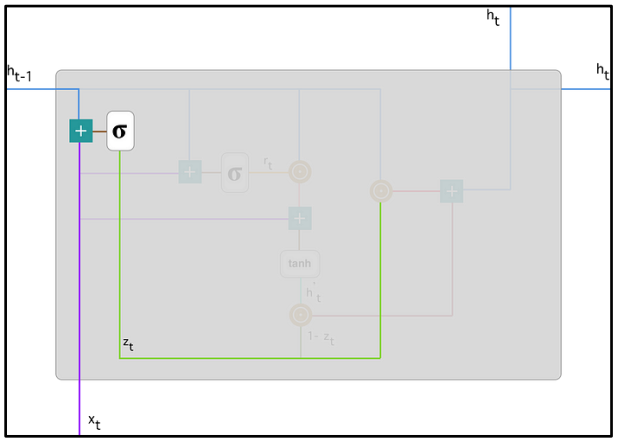

Figure 2.14: The update gate in GRU [90].

B. Reset Gate (r_t): The reset gate helps the model decide how much of the previous hidden state should be forgotten, as shown in Figure 2.15. It is computed as follows:

$$r_t = \sigma\left(W^r x_t + U^r h_{t-1} + b^r\right) \quad (2.9)\ [93]$$

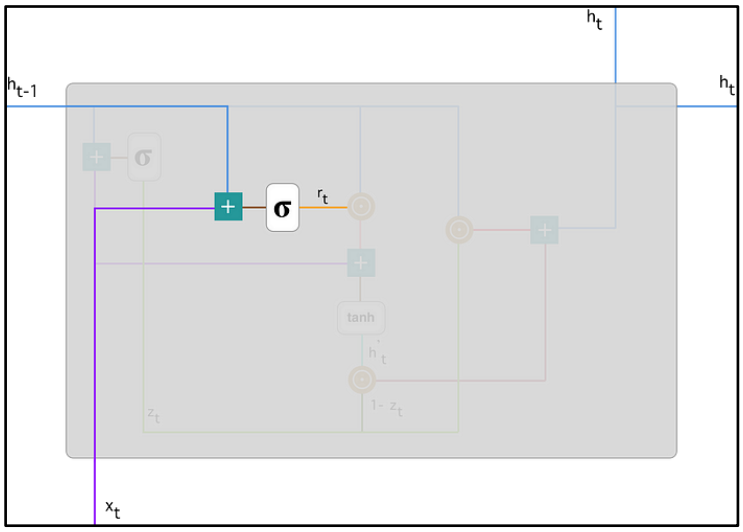

Figure 2.15: The reset gate in GRU [90].

The reset gate works in tandem with the update gate and helps determine how much past information needs to be forgotten. The computation is similar to the update gate, involving the current input and the previous hidden state, with different weight matrices for both (W^r and U^r), and a bias term b^r. The output is passed through the sigmoid function [91].

The reset gate controls how much the past hidden state can influence the candidate hidden state. When the reset gate outputs a vector close to 0, the candidate hidden state will be computed almost independently of

the past hidden state, allowing the GRU cell to forget the past hidden state [94].

C. Candidate Hidden State ($\tilde{h}_t$): The candidate's hidden state represents the new memory to be added to the long-term state. It is computed using the current input, the previous hidden state, and the reset gate:

$$\tilde{h}_t = tanh\,(W^h x_t + U^h\,(h_{t-1} \odot r_t) + b^h) \quad (2.10)\ [94]$$

The candidate hidden state calculation introduces potential new information to be added to the hidden state. It's created using the current input, the reset gate, and the previous hidden state, as shown in Figure 2.16. Unlike the gates, the activation function used here is the hyperbolic tangent (tanh), which outputs values between -1 and 1.

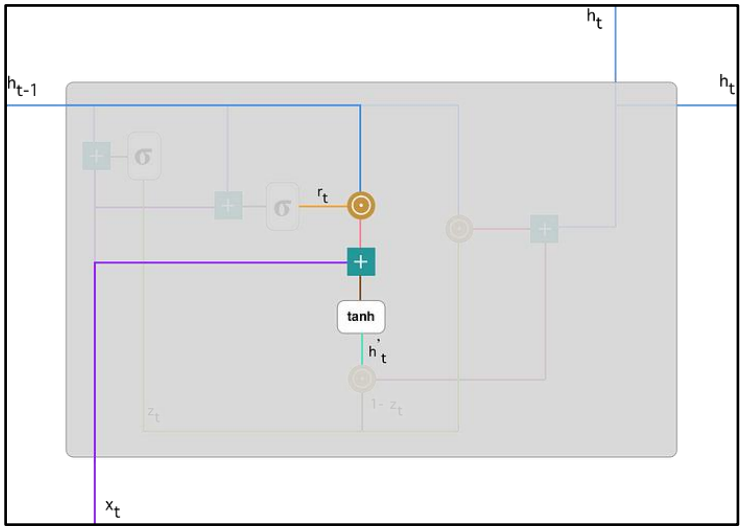

Figure 2.16: The Candidate Hidden State in GRU [90].

The reset gate r_t effectively controls which part of the previous hidden state h_{t-1} will influence the candidate hidden state. When the reset gate values are close to 0, the candidate hidden state will be computed almost solely based on the current input, allowing the cell to forget the previously stored information [91].

The Hadamard product, often denoted as $\odot$, is a mathematical operation that takes two matrices (or vectors) of the same dimensions and produces a third matrix (or vector) which is the element-wise multiplication of the input matrices (or vectors) [95].

The term ($h_{t-1} \odot r_t$) represents the Hadamard product of the reset gate output r_t and the previous hidden state h_{t-1}, is an element-wise multiplication, where each element in the reset gate vector is multiplied by the corresponding element in the hidden state vector [95].

The result is a vector of the same size, where each element results from the multiplication of the corresponding elements in the reset gate and the previous hidden state. This operation allows the reset gate to effectively control the influence each element in the previous hidden state will have on the computation of the candidate hidden state [95].

D. Final Hidden State (h_t): The final hidden state for the current time step is computed as a combination of the previous and candidate hidden states, controlled by the update gate as shown in Figure 2.17.:

$$h_t = (1 - z_t) \odot \tilde{h}_t + z_t \odot h_{t-1} \quad (2.11)\ [95]$$

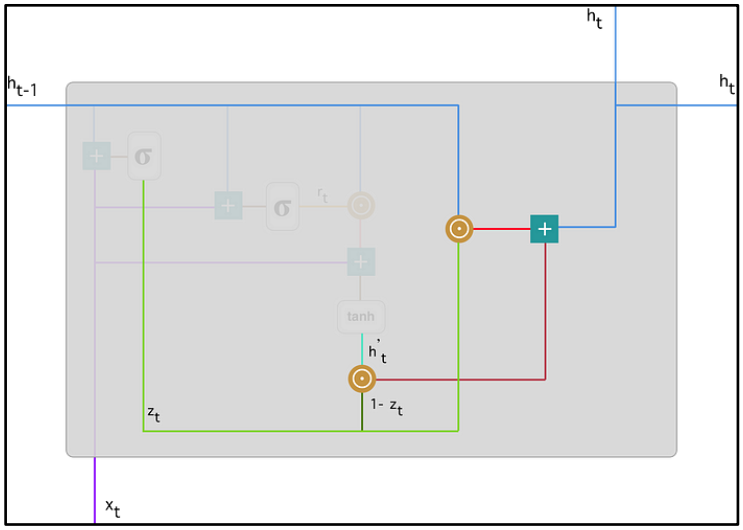

Figure 2.17: The Final Hidden State in GRU [90].

The final hidden state is a weighted average of the previous and candidate hidden states. The weights for this averaging process come from the update gate (z_t) [96].

The term ($1 - z_t$) is an element-wise inversion of the update gate. This means that for every element in the update gate vector that was close to 1 (indicating a preference to keep the corresponding element from the old hidden state), the corresponding element in ($1 - z_t$) will be close to 0, and vice versa. This inverted update gate is then element-wise multiplied by the candidate hidden state $\tilde{h}_t$, effectively "gating" the candidate hidden state. T parts of the candidate hidden state corresponding to a high value in the inverted update gate get passed through, while the parts corresponding to a low value get suppressed [97].

The term of $z_t \odot h_{t-1}$ is part of the equation represents the element-wise multiplication of the update gate z_t and the previous hidden state h_{t-1}. This means that parts of the old hidden state that were deemed important by the update gate (corresponding elements close to 1) get passed through, while parts that were deemed unimportant (corresponding elements close to 0) get suppressed [97].

The final hidden state h_t is obtained by adding these two results together. It's a mix of the gated candidate hidden state and the gated previous hidden state, where the update gate determines the proportions. This mechanism enables the GRU cell to retain long-term dependencies by adapting to pass through or suppress parts of the candidate and previous hidden states [98].

The brilliance of the GRU lies in these equations and how they control and manage information flow. In conjunction with the candidate state, the update and reset gates allow the GRU cell to effectively learn what to keep and discard from the past, making it a powerful tool for tasks involving sequential data.

Despite their relative simplicity, GRUs have been used successfully in various applications. They're used extensively in natural language processing tasks like language modeling [99] [100], machine translation [101], and text generation [102]. They've also been used in speech recognition [103], music generation, and image generation tasks [104].

GRUs represent an important milestone in the evolution of recurrent neural networks. They're a simpler, more efficient alternative to LSTMs that still offer competitive performance, making them a valuable tool for any deep learning practitioner's toolkit.

2.4.7 Activation Functions

Activation functions form the bedrock of Machine Learning (ML) and deep learning models. They introduce the much-needed non-linearity into the output of a neuron, an attribute that allows the model to learn from the complex patterns within data. They decide whether a neuron should be 'fired' or 'activated.' When you consider the composition of a neuron, it receives several inputs, each with its associated weight. These are summed up, and a bias is added. At this junction, the activation function is applied to this sum, determining the final output [105].

Non-linearity is crucial because it allows the model to capture the complexities and varieties of data. Most real-world data is non-linear, so we want our neural network to learn from this non-linear data and perform non-trivial tasks such as language translation, image classification, and more [105].

No matter how many layers we stack in the neural network, without activation functions it will behave the same way as a single layer, i.e., the neural network would essentially act as a linear regression model [106].

As our understanding of neural networks has evolved, so too have the intricacies of activation functions and their variants. It is clear that these

are not simple threshold functions but instrumental in determining the complexity and performance of neural networks. The type of activation function used can dramatically impact the model's ability to learn complex patterns and generalize well from the training data [106].

Numerous activation functions are available, each with its advantages, disadvantages, and ideal use-cases. They range from the simple linear activation function, where the output is proportional to the input, to non-linear processes such as Sigmoid, Tanh (Hyperbolic Tangent), ReLU (Rectified Linear Unit), Leaky ReLU, and Softmax [107].

2.4.7.1 Type Of Activation Function

Linear Activation Function: The Linear activation function is the simplest. As its name suggests, this function takes an input, multiplies it by a constant, and outputs the result [108]. Mathematically, it can be represented as:

$$f(x) = cx \quad (2.12)\ [108]$$

Where 'x' is the input to the function, as shown in Figure 2.18. However, a linear activation function has two major limitations. First, the derivative of a linear function is a constant, so the gradient has no relationship with 'x'. This makes it unhelpful for backpropagation, as the gradients provide no feedback to adjust the weights. Second, a neural network composed of only linear activation functions is still linear, making it incapable of handling complex, nonlinear problems [107].

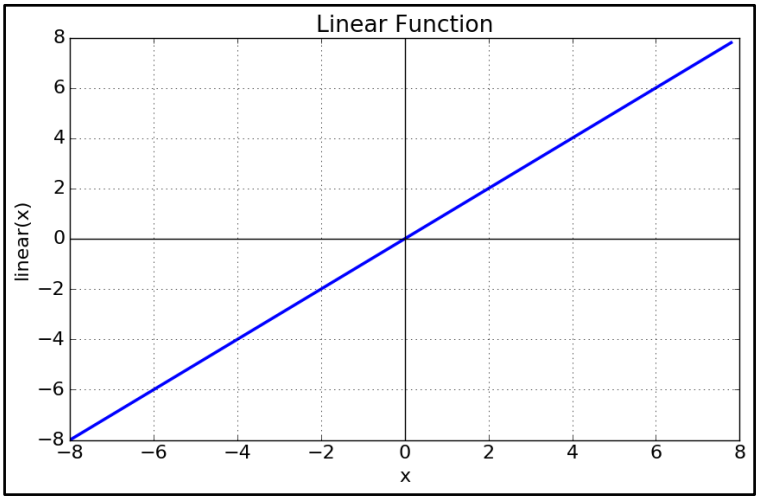

Figure 2.18: linear activation function [107].

Sigmoid Activation Function: The Sigmoid activation function is a type of activation function structured to mimic a simple biological neuron, outputting values between 0 and 1, as shown in Figure 2.19 [109]. Its mathematical formula is given by:

$$f(x) = \frac{1}{(1+e^{-x})} \qquad (2.13)\ [109]$$

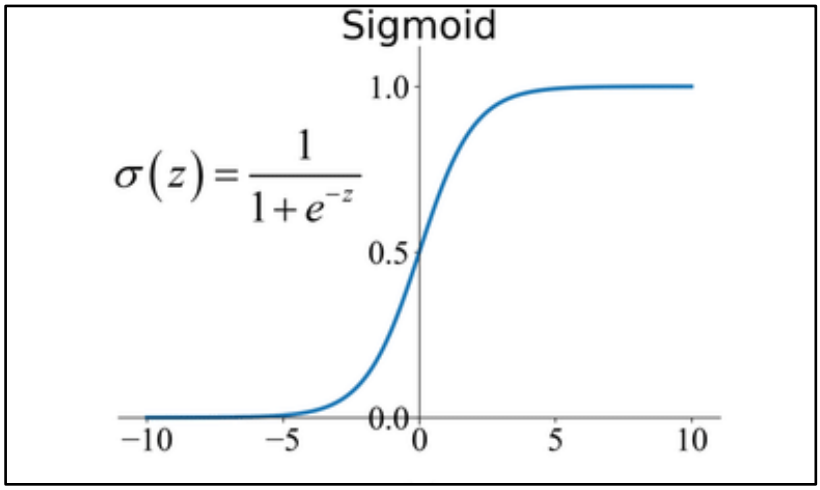

Figure 2.19: Sigmoid Activation Function [110].

This S-shaped curve is characterized by its smooth and continuous nature, which makes it highly suitable for use in neural networks as it enables smooth transitions between activated and non-activated states [111].

Tanh (Hyperbolic Tangent) Activation Function: The hyperbolic tangent (tanh) function is another widely used activation function in neural networks. The tanh function maps input values to an output value within the range of -1 to 1, as shown in Figure 2.20. Its mathematical formula is given by:

$$\sigma(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \qquad (2.14)\ [112]$$

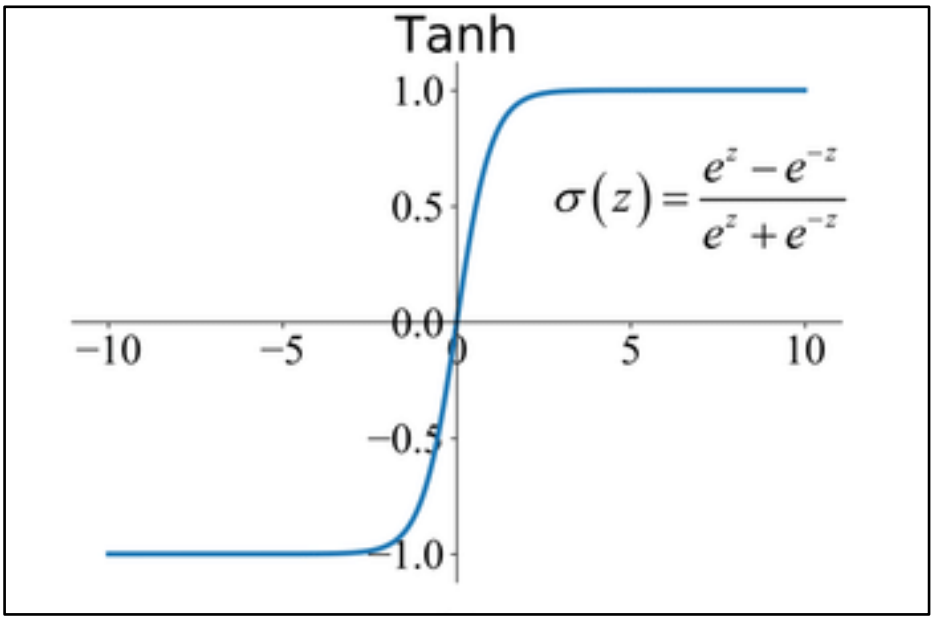

Figure 2.20: Tanh Activation Functions [112].

The tanh function shares some similarities with the sigmoid function, such as being continuous, smooth, and differentiable. However,

the tanh function provides a broader output range and has a zero-centered output, which can lead to faster convergence during the training process [112].

ReLU (Rectified Linear Unit) Activation Function: ReLU stands for Rectified Linear Unit, and it is perhaps the most used activation function when training neural networks due to its computational efficiency [113]. ReLU is defined as:

$$f(x) = max(\,0\,, x\,) \qquad (2.15)\ [113]$$

ReLU provides a very simple non-linearity, as shown in Figure 2.21. When the input is positive, the output is equal to the input. If the input is negative or zero, the outcome is zero. Despite its simplicity, ReLU has been shown to train deep networks efficiently. However, one of its disadvantages is that it can result in dead neurons, i.e., neurons that are not activated and therefore do not learn [114].

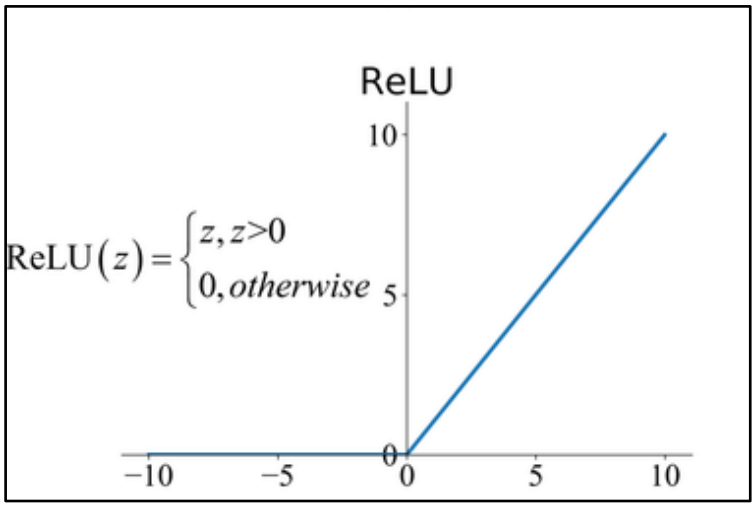

Figure 2.21: ReLU Activation Functions [86].

Leaky ReLU Activation Function: Leaky ReLU is a variation of ReLU that attempts to solve the problem of dead neurons. Instead of defining all negative inputs as zero, Leaky ReLU allows a small, non-zero output for negative input values, as shown in Figure 2.22 [115]. The Leaky ReLU function is represented as:

$$f(x) = max(\, 0.1\, x\, , x\,) \qquad (2.16)\ [115]$$

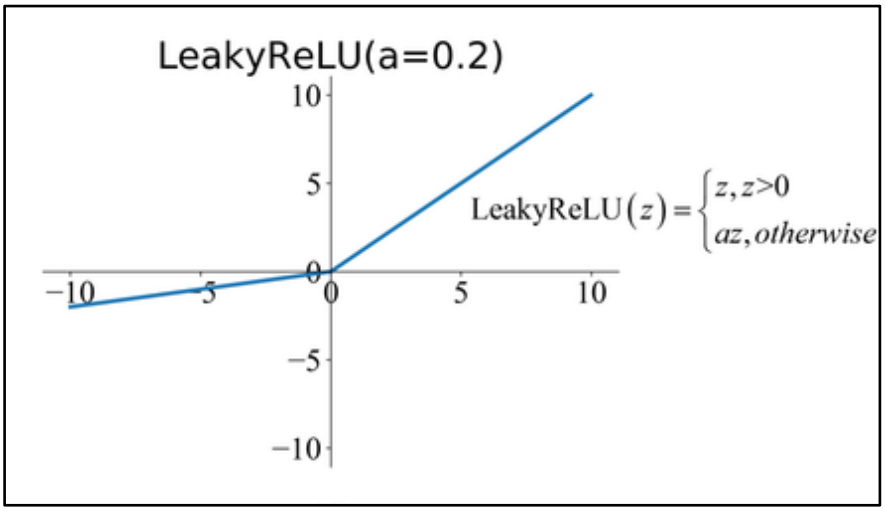

Figure 2.22: Leaky ReLU Activation Function [115].

The slope for the negative region can also be tuned. Leaky ReLU ensures that the neurons never die; they can go into sleep mode for negative input values but can be awakened with positive inputs.

Softmax Activation Function: The Softmax function is often used for the output layer in a multi-class classification neural network. It takes a vector of arbitrary real-valued scores and squashes it to a vector of values between zero and one that sum to one as shown in Figure 2.23 [116]. The softmax function is defined as:

$$f(x_i) = \frac{e^{xi}}{\sum_{j=1}^{n} e^{xj}} \quad (2.17)\ [116]$$

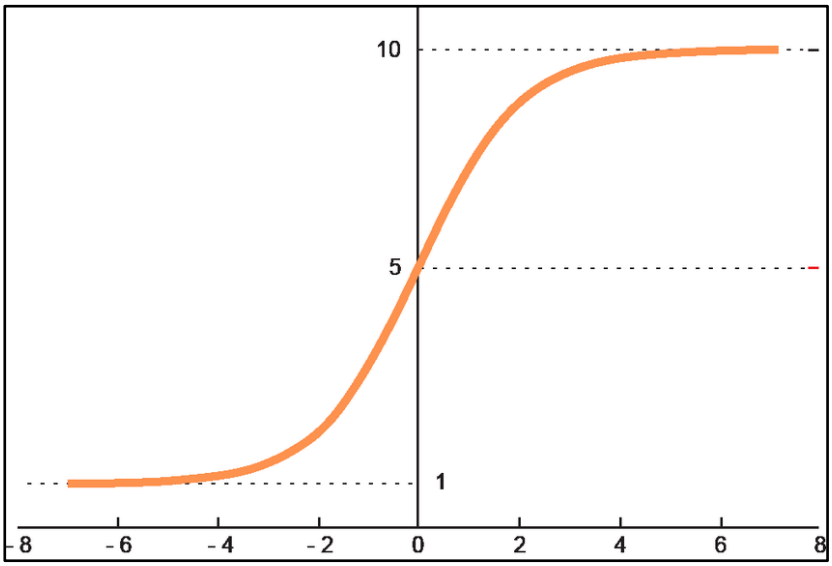

Figure 2.23: Softmax Activation Function [117].

Where, x_i represents the element of the input vector, “ "ith" the element at the "i" position in the sequence, where "i" could be any integer value, “and the denominator is the sum of the exponentials of all the elements of the input vector. The output is a probability distribution where the probabilities sum to 1 [116].

2.5 Optimization Algorithms and techniques

The effectiveness of machine learning models significantly depends on the optimization algorithms and techniques they employ. These techniques guide the iterative refinement of the model parameters to minimize the difference between the model's predictions and the actual output, fine-tune the model structure, and help select the most relevant

features. This section presents a comprehensive overview of this research's critical optimization algorithms and techniques.

2.5.1 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is a widely-used optimization algorithm in machine learning and deep learning for training a wide range of models. Its basis lies in the traditional gradient descent algorithm but with a critical difference that makes it more computationally efficient, especially for large datasets [118].

Gradient Descent [119], in essence, is an iterative optimization algorithm that seeks to find the minimum of a function. In machine learning, this function is usually the loss or cost function, quantifying the error between the model's predictions and the actual data. The 'gradient' in Gradient Descent refers to the derivative of the loss function, pointing toward the steepest ascent. Therefore, the algorithm takes steps in the opposite direction of the gradient to find the minimum of the function.

As an optimization algorithm, gradient Descent [119] has demonstrated efficacy within Deep Learning applications. This algorithm utilizes an iterative method to assess the magnitude of a variable's variation and identifies the minimal achievable value of a convex cost function. Nevertheless, it necessitates using the entire data set as a single batch before adjusting the neural network's weight parameters. Offering a contrasting approach, Stochastic Gradient Descent (SGD), the operations of which are illustrated in Figure 2.24, tackles this problem by modifying

weights at each iteration, deploying a batch size of merely one instance [120].

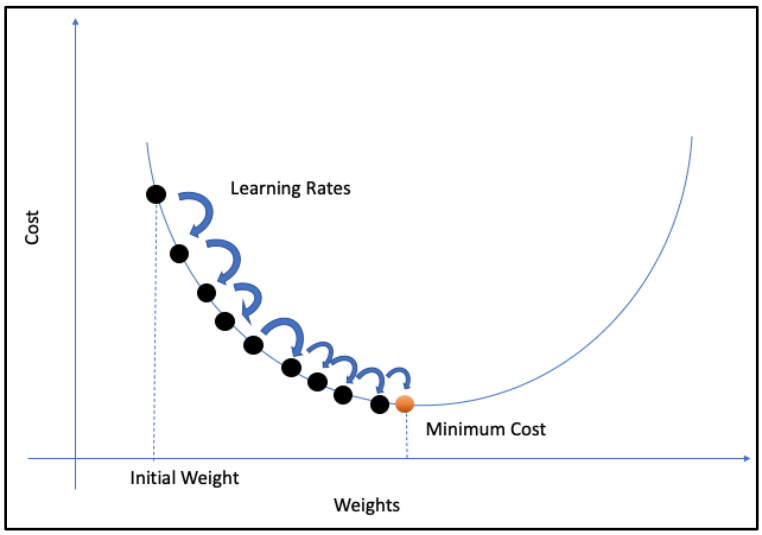

Figure 2.24: Stochastic Gradient Descent [120].

2.5.2 Root Mean Square Propagation (RMSProp)

Root Mean Square Propagation (RMSProp) is an optimization algorithm designed to speed up the training of neural networks. It was first proposed by Geoffrey Hinton in his Coursera lecture "Neural Networks for Machine Learning" [121]

RMSProp is an extension of Stochastic Gradient Descent (SGD), specifically designed to resolve the diminishing learning rates issue in the latter. It does so by maintaining a moving average of the squared gradients for each weight and dividing the learning rate by the square root of its corresponding average. This automatically balances the learning rates, allowing RMSProp to converge faster and with less tuning of the learning rate [122].

The key idea of the RMSProp is to use a decaying average of the squared gradients to normalize the gradient itself. This normalization helps resolve the problem of the rapidly diminishing learning rates in SGD[123].

RMSProp maintains a moving (decaying) average of the squared gradients and divides the learning rate by the square root of this average. This means the algorithm adapts the learning rate to each weight: weights associated with frequently occurring features get a reduced learning rate, while rare features get a higher learning rate [124].

2.5.3 Adaptive Moment Estimation (ADAM)

Adaptive Moment Estimation (ADAM) is a popular and efficient gradient-based optimization algorithm developed specifically for training deep neural networks. It was proposed by Diederik P. Kingma and Jimmy Ba in their paper "Adam: A Method for Stochastic Optimization" at ICLR 2015 [125].

ADAM combines the benefits of two other popular optimization methods, namely Root Mean Square Propagation (RMSProp) and Adaptive Gradient Algorithm (AdaGrad), to handle sparse gradients on noisy problems [125].

The employment of the Adam optimizer within neural network applications has attracted substantial interest due to its capacity to enhance training precision and expedite convergence, outperforming conventional optimization algorithms. The amalgamation of the Adam optimizer and other methods, for instance, the introduction of long short-term memory

(LSTM) with layer normalization, was demonstrated to boost performance [126]. By deploying the Adam optimizer, they enhanced their model's generalization capabilities, leading to more robust and efficient operation for short-term load forecasting in power. This underlines the potential application of the Adam optimizer beyond merely enhancing accuracy and convergence velocity during training, contributing to improved system reliability as a whole. In addition, [127] discovered that utilizing the Adam optimizer led to superior outcomes compared with traditional optimization algorithms when deploying LSTM models. Their study revealed that the integration of this technique enabled them to attain higher prediction precision on real-world datasets, thereby validating its effectiveness. The advantages of the Adam optimizer extend beyond LSTM models; researchers have also reported success using this technique within other neural network architectures [120].

The Key Features of the Adam Optimizer Include:

- Adaptive learning rates: Adam adapts the learning rate for each parameter individually based on the gradients' first-order moments (mean) and second-order moments (variance). This allows for faster convergence and improved robustness to different datasets and model architectures [128].

- Momentum and RMSProp: Adam combines the concepts of momentum and RMSProp (Root Mean Square Propagation) to leverage the advantages of both techniques. Momentum helps the optimizer navigate through plateaus and local minima in the loss landscape, while

RMSProp ensures that the updates are scaled appropriately based on the magnitude of the gradients [129].

- Bias correction: Adam applies bias correction to the estimates of the first-order and second-order moments, which leads to more accurate weight updates, especially in the early stages of training [130].

The Adam Optimizer Equations are as follows:

- Compute the gradients: The gradients of the mean squared error (MSE) loss function concerning the neural networks model's parameters (weights and biases) are computed using backpropagation through time (BPTT). The gradient g_t at time step t represents the direction of the steepest increase in the loss function concerning the model parameters.

$$g_t = \nabla_\theta J(\theta_t) \quad (2.18) [125]$$

- Update the first-order moment (mean) estimate: The first-order moment estimates the mean of the gradients. It is updated by taking a linear combination of the previous moment estimate m_{t-1} and the current gradient g_t. The decay factor β_1 controls the degree of contribution from previous and current gradients.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (2.19) [125]$$

- Update the second-order moment (variance) estimate: The second-order moment estimates the uncentered variance of the gradients. It is updated by taking a linear combination of the previous moment estimate v_{t-1}

and the squared current gradient g_t^2. The decay factor β_2 controls the degree of contribution from previous and current squared gradients.

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (2.20)\ [125]$$

- Apply bias correction to the first-order moment estimate: In the early stages of training, the first-order moment estimate may be biased toward zero. To correct for this bias, divide the first-order moment estimate m_t by $1 - \beta_1^t$, where t is the current time step.

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t} \quad (2.21)\ [125]$$

- Apply bias correction to the second-order moment estimate: Similarly, the second-order moment estimate may also be biased towards zero at the beginning of training. To correct for this bias, divide the second-order moment estimate v_t by $1 - \beta_2^t$.

$$\widehat{v_t} = \frac{v_t}{1-\beta_2^t} \quad (2.22)\ [125]$$

- Update the parameters: Finally, the model parameters are updated by taking a step in the direction of the bias-corrected first-order moment estimate $\widehat{m_t}$, scaled by the adaptive learning rate, which is the ratio of the learning rate $\propto$ to the square root of the bias-corrected second-order moment estimate $\widehat{v_t}$ plus a small constant ε to prevent division by zero.

$$\theta_t + 1 = \theta_t - \propto \frac{\widehat{m_t}}{\sqrt{\widehat{v_t} + \varepsilon}} \quad (2.23)\ [125]$$

During training, the Adam optimizer adjusts the weights and biases of the model based on the gradients of the MSE loss function with respect to these parameters. This ensures that the model learns to map the input features to the target output variable as accurately as possible Figure 2.25 shows a detailed algorithm.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1^{st} moment vector)
$v_0 \leftarrow 0$ (Initialize 2^{nd} moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

Figure 2.25. Adam algorithm for stochastic optimization [125].

2.5.4 Bayesian Optimization Algorithm

Bayesian Optimization is a probabilistic model-based approach to finding the minimum of a function. This function is typically unknown and expensive to evaluate. It is ideal for applications like hyperparameter tuning in machine learning, where assessing a set of hyperparameters can mean training a model, which is time-consuming[131].

The key idea behind Bayesian Optimization is to use a probabilistic model to predict the outcome of unknown inputs and select new inputs to evaluate the function based on the trade-off between exploration and exploitation.

Bayesian Optimization uses a Gaussian Process (GP) as a prior over functions to model the unknown function and updates the GP with the function's observations [132].

Two critical components of Bayesian Optimization are the surrogate model and the acquisition function. The surrogate model, often a Gaussian Process, provides a probabilistic estimate of the unknown function. The acquisition function uses this surrogate model to decide the next query point by balancing exploration (searching in less-explored areas) and exploitation (zooming into regions of known good results) [133].

The Bayesian optimization algorithm is shown in Figure 2.26, representing the training dataset consisting of t –1 observations of function(f). From the description of the algorithm, we can see that the whole algorithm is composed of two parts: Updating the posterior distribution (steps 3 and 4) and maximizing the acquisition function (step 2). As the observations accumulate, the posterior distribution is updated continuously; based on the new posterior, the point where the acquisition function is maximized is found and added to the training dataset. The process is repeated until the maximum number of iterations is reached, or the difference between the current and optimal values obtained so far is less than a predefined threshold. It is noted that Bayesian optimization does

not require the explicit expression of function (f) compared with other optimization methods, such as gradient descent algorithms. Therefore it has a broader range of applications [131].

1: **for** $n = 1, 2, \ldots$ **do**
2: select new $\mathbf{x}_{n+1}$ by optimizing acquisition function α

$$\mathbf{x}_{n+1} = \arg\max_{\mathbf{x}} \alpha(\mathbf{x}; \mathcal{D}_n)$$

3: query objective function to obtain y_{n+1}
4: augment data $\mathcal{D}_{n+1} = \{\mathcal{D}_n, (\mathbf{x}_{n+1}, y_{n+1})\}$
5: update statistical model
6: **end for**

Figure 2.26: Pseudo code Bayesian Optimization Algorithm [131].

2.5.5 Hyperband Optimization Algorithm

Hyperband is a state-of-the-art hyperparameter optimization algorithm extending the Successive Halving (SH) algorithm by introducing an additional "outer loop" to allocate resources to promising configurations more efficiently. It was proposed by Li et al. [134].

The main principle of Hyperband is to allocate more resources to promising hyperparameters and less to those that do not show promise. This is done by evaluating a large number of configurations for a few iterations and promoting only the top-performing fraction to the next round, where they are trained for more iterations. This process is repeated, reducing the number of configurations at each round until only the most promising configurations remain [134].

Hyperband treats the number of iterations as a resource (e.g., the number of epochs in deep learning) and operates in a bandit-based framework, balancing exploration and exploitation. Introducing an outer loop to Successive Halvingxplores a more significant number of configurations with fewer resources and a smaller number of configurations with more resources, providing a more thorough exploration of the configuration space [135].

The Hyperband optimization algorithm can be depicted as shown in Figure 2.27, where it operates on hyperparameters of a function to optimize model training. The algorithm's process can be broken down into two primary segments: Executing the successive halving process (inner loop in steps 3 to 9) and adjusting resource allocation (steps 1 to 2 and step 11) [134].

input : R, η (default $\eta = 3$)
initialization : $s_{\max} = \lfloor \log_\eta(R) \rfloor$, $B = (s_{\max} + 1)R$
1 **for** $s \in \{s_{\max}, s_{\max} - 1, \ldots, 0\}$ **do**
2 $\quad n = \lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \rceil$, $\quad r = R\eta^{-s}$
$\quad$ // begin SuccessiveHalving with (n, r) inner loop
3 $\quad T =$ `get_hyperparameter_configuration`(n)
4 $\quad$ **for** $i \in \{0, \ldots, s\}$ **do**
5 $\quad\quad n_i = \lfloor n\eta^{-i} \rfloor$
6 $\quad\quad r_i = r\eta^i$
7 $\quad\quad L = \{$`run_then_return_val_loss`$(t, r_i) : t \in T\}$
8 $\quad\quad T =$ `top_k`$(T, L, \lfloor n_i/\eta \rfloor)$
9 $\quad$ **end**
10 **end**
11 **return** *Configuration with the smallest intermediate loss seen so far.*

Figure 2.27: Hyperband optimization algorithm [134].

2.5.6 K-Nearest Neighbor Algorithm (KNN)

The k-Nearest-Neighbours (kNN) is a non-parametric classification method, which is simple but effective in many cases. For a data record t to be classified, its k nearest neighbours are retrieved, and this forms a neighbourhood of t. Majority voting among the data records in the neighbourhood is usually used to decide the classification for t with or without consideration of distance-based weighting. However, to apply kNN we need to choose an appropriate value for k, and the success of classification is very much dependent on this value. In a sense, the kNN method is biased by k. There are many ways of choosing the k value, but a simple one is to run the algorithm many times with different k values and choose the one with the best performance [136].

kNN is a simple but effective method for classification and it is convincing as one of the most effective methods on Reuters corpus of newswire stories in text categorization, it motivates us to build a model for kNN to improve its efficiency whilst preserving its classification accuracy as well.

2.6 Data Preprocessing

Approaches for Data Preprocessing are utilized to handle data characterized by suboptimal quality. The methodologies used in Data

Preprocessing can be bifurcated into two primary categories. The initial category involves data purification by eliminating noise, addressing missing data, and removing redundancy [137]. The subsequent category deals with the process of feature engineering, encompassing the generation of technical indicators, normalization of data features, and selection of the most effective features while disregarding the irrelevant ones through feature selection techniques. This research employs most of these techniques from both categories, given their potential to significantly enhance the efficacy of the forecasting process [138].

2.6.1 Z-Score

The Z-score is a statistical measurement that describes a value's relationship to the mean of a group of values. It is measured in terms of standard deviations from the mean. If a Z-score is 0, the data point's score is identical to the mean score. A Z-score of 1.0 would indicate a value of one standard deviation from the mean [139]. The formula for calculating a Z-score:

$$Z = \left(\frac{X-\mu}{\sigma}\right) \quad (2.24)\ [139]$$

Where, Z is the Z-score, X is the value of the element, μ is the population mean, σ is the standard deviation.

2.6.2 Interpolation

Interpolation is a statistical method by which an estimated value is obtained for any point between two known values. Interpolation is used

when we have data at specific points and want to estimate the value of the data at a point within this range [140].

There are several interpolation methods, but the simplest is linear interpolation. The formula for linear interpolation is:

$$y = y_1 + (x - x_1)\frac{(y_2 - y_1)}{(x_2 - x_1)} \quad (2.25)\ [140]$$

Where y is the interpolated value. (x1, y1) and (x2, y2) are the coordinates of the known points. x is the point at which the value needs to be interpolated.

2.6.3 Min-Max Scaling (Normalization)

Min-Max scaling, also referred to as normalization, is a technique used in data preprocessing to standardize the range of independent variables or features of data. It changes the scale of your features while keeping the shapes of their distributions intact [141]. The formula for Min-Max Scaling is:

$$\grave{x} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (2.26)\ [141]$$

The values of $\grave{x}$ will always fall between 0 and 1. This method works by taking the original range of values in the feature and transforming it to a scale of 0 to 1.

2.6.4 Correlation Matrix

The correlation matrix is a table that shows the correlation coefficients between many variables. Each cell in the table displays the correlation between two variables. A correlation matrix is used to summarize data, as input into a more advanced analysis, and as a diagnostic for advanced examinations [142].

In data analysis and data science, a correlation matrix is often used as a preliminary visualization, helping to understand the relationships between different variables in a dataset [142]. We can use it in:

- Feature Selection: Correlation matrices can be used to select features for a machine learning model. Features that are highly correlated with the target variable can be prioritized. Simultaneously, amongst the highly correlated features, one element may be chosen over the other to avoid duplication of similar information known as “multicollinearity”.

- Understanding Trends: Correlation matrices can help understand the trends, patterns, and relationships between variables. This can be helpful in industries like finance, where it's essential to understand the relationship between different commodities or stocks.

- Statistical Analysis: The matrix is also used as an input for advanced statistical analyses like multiple regression, PCA, and factor analysis.

2.7 Evaluation Measures

In data analysis, machine learning, and statistical modeling, evaluation measures are crucial for assessing the performance and effectiveness of models and algorithms. This is especially true in regression tasks, where the goal is to predict continuous numerical values based on input variables [143].

Regression analysis involves understanding the relationship between the dependent variable (the target variable to be predicted) and one or more independent variables (also known as features or predictors). By establishing this relationship, regression models can make accurate predictions or estimate the values of the dependent variable for new, unseen data [144].

Evaluation measures specific to regression tasks provide insights into the regression models' accuracy, precision, and reliability. These measures are essential for understanding how well the models capture the underlying relationships between the input and continuous output variables.

2.7.1 Mean Absolute Error (MAE)

The Mean Absolute Error measures the average magnitude of errors in a set of predictions without considering their direction. It is calculated as the average of the absolute differences between actual and predicted values. MAE is a simple and interpretable metric, directly representing the

average error in the same units as the data. Lower MAE values indicate better model performance [143].

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \quad (2.27)\ [143]$$

2.7.2 Mean Squared Error (MSE)

The Mean Squared Error measures the average squared difference between actual and predicted values. Squaring the errors gives more weight to larger errors, making it more sensitive to outliers than MAE. Lower MSE values indicate better model performance [144].

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \quad (2.28)\ [144]$$

2.7.3 Root Mean Squared Error (RMSE)

The Root Mean Squared Error is the square root of the Mean Squared Error. RMSE measures the average magnitude of the error and has the same units as the data, making it more interpretable than MSE. Like MSE, RMSE is more sensitive to outliers than MAE. Lower RMSE values indicate better model performance [145].

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \quad (2.29)\ [145]$$

2.7.4 R-squared (R^2)

R-squared, also known as the coefficient of determination, measures the proportion of variance in the dependent variable that the independent variables in the model can be explained. R^2 ranges from 0 to 1, with higher values indicating a better fit of the model to the data. An R^2 value of 1 means that the model perfectly explains the variance in the data, while an R^2 value of 0 means that the model does not explain any conflict [146].

$$\text{MSE} = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2} \quad (2.30)\ [146]$$

In supervised machine learning, regression analysis plays a significant role, involving predicting a continuous dependent variable using a set of predictor variables. The distinction between binary classification and regression lies in the target range, with binary classification having only two possible values (typically 0 and 1), while regression allows for multiple target values. Despite the widespread use of regression analysis in machine learning research, a single, universally accepted standard metric for evaluating regression outcomes has not been established. Commonly used metrics include mean square error (MSE), root mean square error (RMSE), mean absolute error (MAE), and mean absolute percentage error (MAPE); however, these metrics share a limitation in that their values can range from zero to positive infinity, making it difficult to interpret their performance in relation to the ground truth distribution [147].

In [147], the authors concentrate on two metrics that yield high scores only when the majority of a ground truth group's elements are accurately predicted: the coefficient of determination (also known as R-squared or R^2) and the symmetric mean absolute percentage error (SMAPE). After discussing their mathematical properties, the authors compare R^2 and SMAPE in various use cases, including two real medical scenarios. The findings reveal that the R-squared metric provides more accurate and informative results than SMAPE without the interpretability constraints of MSE, RMSE, MAE, and MAPE. Consequently, [147] recommends employing R-squared as the standard metric for evaluating regression analyses across all scientific fields.

2.7.5 Mean Absolute Percentage Error (MAPE)

MAPE is a measure of the accuracy of a predictive model expressed as a percentage. It quantifies the average percentage difference between the predicted and actual values [148].

MAPE is calculated as the mean of the absolute percentage differences between the predicted values (Y_p) and the actual values (Y_i), where nn is the number of data points.

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{Y_i - Y_{p,i}}{Y_i} \right| * 100 \qquad (2.31)\ [148]$$

2.7.6 Mean Absolute Scaled Error (MASE)

MASE is a scale-independent error metric that measures the accuracy of a predictive model relative to a naive or benchmark model. It provides a comparison of how well the model performs in relation to a simple model [149].

MASE is calculated as the mean of the absolute errors (e_i) divided by the mean absolute error of the naive model (e_b). Here, nn is the number of data points.

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|e_i|}{e_b} \quad (2.32)[149]$$

Where e_i is the absolute error for each data point, and e_b is the mean absolute error of the naive model.

2.7.7 Explained Variance Score (EV)

The EV score measures how well a model can explain the variance in the observed data. It quantifies the proportion of the variance in the predicted values that the model accounts for [150].

$$\text{EV} = 1 - \frac{Var(Y - Y_p)}{Var\,(Y)} \quad (2.33)[150]$$

Where Y represents the actual values, Y_p represents the predicted values, and Var denotes the variance of a variable. The closer the EV score is to 1, the better the model explains the variance.

2.7.8 Mean Percentage Deviation (MPD)

MPD measures the average percentage deviation of the predicted values from the actual values. It quantifies the directional bias in the predictions, indicating whether the model tends to overestimate or underestimate [151].

MPD is calculated as the mean of the percentage differences between the predicted values (YpYp) and the actual values (YY), where nn is the number of data points.

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^{n} \frac{Y_i - Y_{p,i}}{Y_i} * 100 \qquad (2.34)[151]$$

CHAPTER THREE

THE PROPOSED SYSTEM

3.1 Introduction

This dissertation focused on creating a CQI prediction model specifically aimed at significantly improving the management and performance of Massive RANs in the realm of beyond 5G. The model is architectured to efficiently support a massive quantity of connected devices, a crucial aspect considering the ongoing surge in mobile data traffic and the emergence of novel applications.

This research aims to contribute significantly to the evolution of wireless networks, concentrating on enhancing network management and resource allocation and ensuring superior Quality of Service (QoS). These elements are central to the growth of mobile data traffic and the rise of new applications.

Several methods have been applied in wireless communication and network management to tackle the issue of CQI prediction, including statistical, mathematical, and machine-learning techniques. Nevertheless, a gap remains in formulating models that can accurately predict CQI while accommodating network data's dynamic and sequential nature.

In Figure 3.1, we presented a detailed overview of the first stage of our research, which can be divided into four main aspects. Firstly, data processing is addressed, involving data collection, cleaning, and normalization from RANs, laying the foundation for subsequent model selection and evaluation. The second aspect centers on the initial model selection process, where we explore various machine learning models, including "Artificial Neural Networks (ANN), Deep Neural Networks

(DNN), Support Vector Machines (SVM), and Long Short-Term Memory (LSTM)." Our primary objective here is to identify the best-performing model to ensure accurate CQI prediction. The third aspect involves evaluating the selected models, applying performance metrics to assess their predictive capabilities, and facilitating a comparison of their effectiveness. Lastly, the fourth aspect represents the transition from the initial model selection to the next stage, focusing on taking the best-performing model from the previous stage, which is LSTM, and preparing for the development of an optimized hybrid LSTM-GRU model.

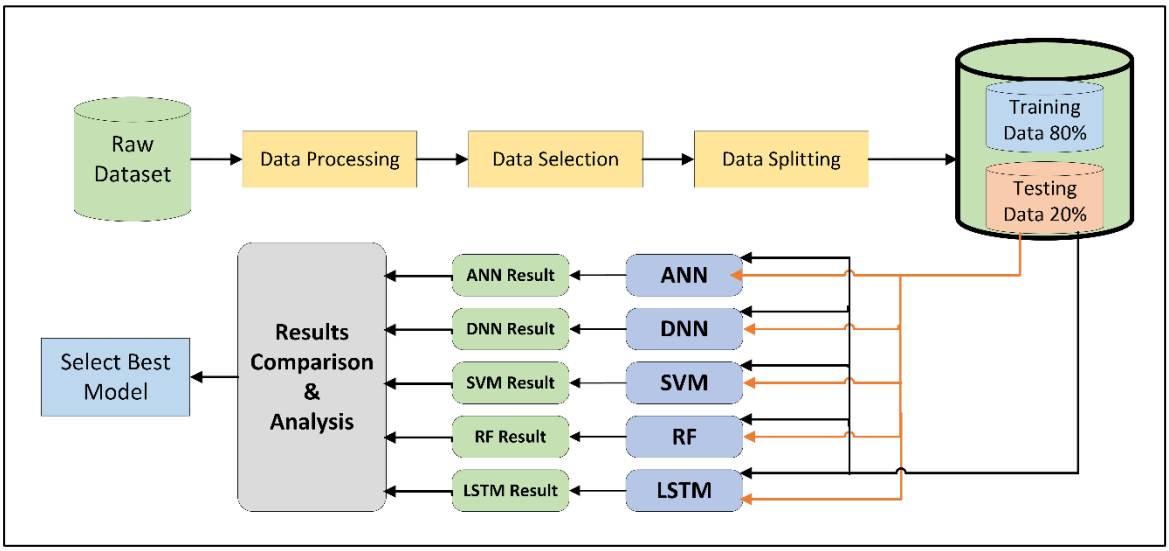

Figure 3.1: first stage for Initial Model Assessment and Selection.

In Figure 3.2, we delve into the second stage of our research, which is dedicated to the development of the proposed system for precise CQI prediction. This stage encompasses key components such as the development and optimization of a hybrid LSTM-GRU model, amalgamating the strengths of LSTM networks and Gated Recurrent Units (GRU) for improved CQI prediction. Hyperparameter tuning is achieved through the utilization of Hyperband and Bayesian optimization

techniques. Subsequently, the trained hybrid LSTM-GRU model's performance is assessed through testing, ensuring its readiness for real-world application. In the final aspect, we compare the performance of the hybrid LSTM-GRU model with the models selected in the first stage (as presented in Figure 3.1), emphasizing the superior predictive capabilities of the hybrid model in CQI prediction.

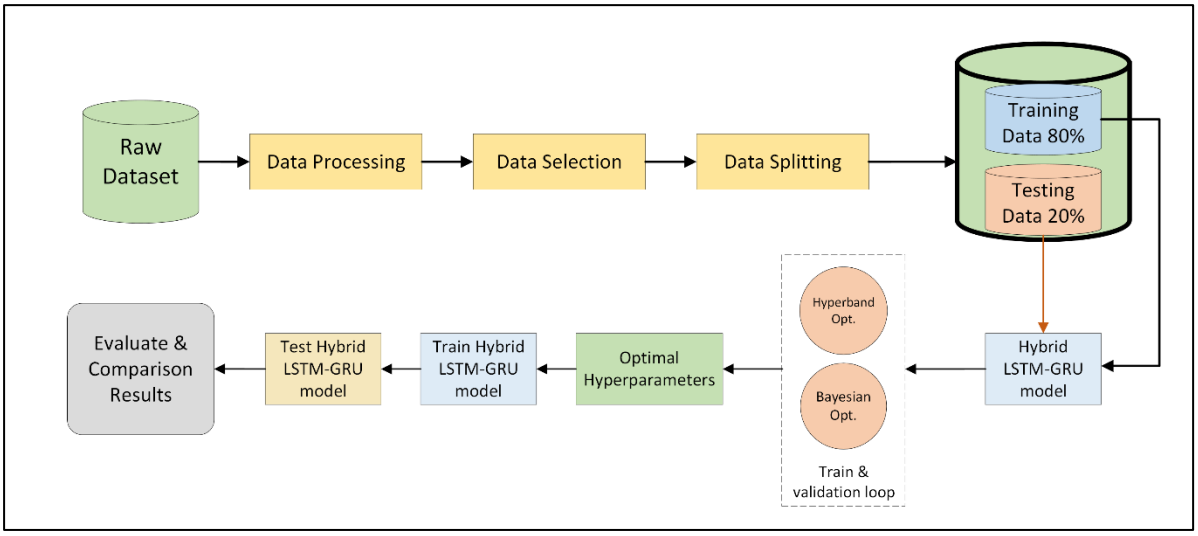

Figure 3.2: Second stage for Hybrid LSTM-GRU Optimization Process.

This dissertation underscores the pivotal role of AI, as opposed to traditional methods, in identifying and selecting the most suitable models for CQI prediction. It places significant emphasis on the design and implementation of the hybrid LSTM-GRU model, which combines the strengths of both LSTM networks and Gated Recurrent Units (GRU), ultimately leading to more accurate and efficient CQI prediction.

3.2 The Proposed System Architecture

Figure 3.1 and 3.2, Portrays the sequence of operations applied in this research to develop a prediction model for CQI in 5G and beyond networks.

- The process starts with a 'Dataset,' the primary information source. This data is collected from various reliable sources and includes many variables related to the 5G network that might influence CQI.

- The next step is 'Data Processing,' which entails cleansing the dataset, handling missing values, and ensuring data consistency. It might also involve normalization or standardization, depending on the nature of the data.

- Following this is the 'Data Selection' phase, where relevant features are selected for model training. Irrelevant or redundant data is eliminated to improve the efficiency of the models.

- The selected data is then divided into the 'Data Splitting' phase, allocating 80% for training the models and 20% for testing their performance.

- In the subsequent phase, Four machine-learning models are trained on the data: ANN, DNN, SVM, and LSTM.

- The individual results of each model - 'The ANN result,' 'DNN result,' 'SVM result,' and 'LSTM result' - are then evaluated and compared.

- The 'Results Comparison and Analysis' stage involves an in-depth analysis of each model's performance to understand its strengths and weaknesses.

- Subsequently, 'The Best Model Result' is identified based on metrics such as MSE, RMSE, MAE, R^2, MAP, MASE, EV, and MPD. This model demonstrates the best performance in predicting the CQI for 5G and beyond networks.

- Once the best model is identified, an 'Enhancement Step' is introduced where the model is improved by introducing a Gated Recurrent Unit (GRU), creating a new 'Hybrid LSTM-GRU Model.' This step aims to enhance the prediction accuracy of the best model.

- The 'Hybrid Model Hyperparameter Tuning' phase follows, where the hybrid model is fine-tuned using two algorithms, namely: Hyperband and Bayesian optimization, to select optimal hyperparameters for the model.

- Next, we move to 'Comparison and Evaluation of Hybrid Model'. Here, the performance of the newly created hybrid LSTM-GRU model is compared and evaluated against the default LSTM model using the same metrics as before (MSE, RMSE, MAE, R^2).

- After evaluating the hybrid model, a thorough 'Hybrid Model Performance Analysis' is conducted to understand its behavior and to ensure that the model is not overfitting or underfitting the data.

- Finally, the last step is 'The Prediction Results', which showcases the performance of the enhanced model in predicting the CQI, thus concluding the entire process.

3.2.1 The Datasets

The first dataset used in this study is the "ElasticMon5G2019" dataset, which was collected and maintained by Eurecom [152]. It consists of 4G and 5G RAN monitoring data captured by the ElasticMon 5G monitoring framework over the FlexRAN platform. The data collection occurred on 11th December 2019, and the dataset was last modified on 28th August 2019.

This dataset is comprised of raw and pre-processed data about 4G/5G MAC, RRC, and PDCP statistics and monitoring information. The raw data was gathered using ElasticMon v0.1, an early version of a monitoring framework designed to extend the FlexRAN 5G programmable platform for Software-Defined Radio Access Networks [152].

The data collection process involves recording data from a single mobile U in five distinct mobility scenarios, with the UE following different movement and distance patterns relative to the base station (eNB).

The dataset is provided in two versions: raw and processed. The raw version, comprises MAC, RRC, and PDCP metric values supplied by the FlexRAN controller. The processed version, adds timestamps to the data, cleans out corrupt/inaccurate metric values, and excludes static values, reducing the metric count per measurement from over 100 to 42.

The raw data was collected from a single Android v8.0 (Oreo) Nexus 6P phone user connected to an eNB. The network used for data collection was based on FlexRAN v2.0 and OpenAirInterface snap packages.

The second dataset used in our model was sourced from the paper titled "Beyond throughput, the next generation: a 5G dataset with channel and context metrics" [153]. The dataset is unique as it captures the throughput, channel, and context information for 5G networks, a feature that hasn't been presented in publicly available datasets prior to this one. The data were obtained from a significant Irish mobile operator.

The data encompasses cellular Key Performance Indicators (KPIs), including channel-related metrics, context-related metrics, cell-related metrics, and throughput information. G-NetTrack Pro, a widely recognized non-rooted Android network monitoring application, generated these metrics. The dataset provides valuable insights into two distinct mobility patterns, static and car, and two application patterns, including video streaming and file download [153].

This dataset is derived from real-time 5G production network data and supplemented with a large-scale multi-cell ns-3 simulation framework designed for 5G. The ns-3 mm-wave network simulator 5G/mm-wave module enables a deeper understanding of the dynamic reasoning for adaptive clients in multi-cell 5G wireless scenarios.

The primary objective of this dataset and its corresponding framework is to provide additional insights, such as competing metrics for users connected to the same cell. This information would otherwise be unavailable from the end-users perspective, especially details of the base station (eNodeB or eNB) environment and scheduling principle.

The dataset provided a rich source of information, assisting in developing a comprehensive model by providing real-world data with varied mobility and application patterns. It allowed for extensive examination of different factors impacting the performance of 5G networks, resulting in a more accurate and versatile model.

3.2.2 Data Pre-processing

In data science, the validity and reliability of the model's output heavily depend on the quality of the input data. Data pre-processing, often considered the most crucial and time-consuming aspect of machine learning, is the foundation for any successful data modeling operation. This stage involves cleaning the data by handling missing or null values, converting data types, and discarding irrelevant information. Furthermore, it encompasses feature engineering, transforming or enriching existing features to maximize the model's performance. This section will elaborate on the various steps involved in the data pre-processing of our study.

Specifically, it will detail how raw data were transformed and prepared to be suitable inputs for the subsequent machine-learning models.

Steps of Data Pre-processing

Input: CSV file

Output: Feature Importance

1: Load DataFrame df from CSV file

2: Convert df['Timestamp'] to datetime format

3: Extract 'Year', 'Month', 'Day', 'Hour' from df['Timestamp']

4: Drop df['Timestamp'] from df

5: Replace '-' with NaN in df

6: Initialize KNN Imputer imp with n_neighbors=3, weights='uniform'

7: Define numeric columns ‘Features’

8: Apply imp on numeric columns of df

9: Define categorical columns

10: Initialize OneHotEncoder ohe with drop='first'

11: Apply ohe on categorical columns of df and add them to df while dropping the original categorical columns

12: Drop any rows in df where 'CQI' is NaN

13: Set y as df['CQI']

14: Drop 'CQI' from df and set it as X

15: Initialize RandomForestRegressor rf with n_estimators=100

16: Fit rf using X and y

17: Get feature importance from rf

18: Return a DataFrame with features and their corresponding importances

According to data pre-processing stage, the first step involves loading the dataset from a CSV file. This data is then processed to convert the 'Timestamp' column into DateTime format as shown in Figure 3.3. Furthermore, additional columns, namely 'Year', 'Month', 'Day', 'Hour', 'Minute', and 'Second' are created to break down the timestamp into more helpful information and convert nominal to numeric data. Once this is

completed, the 'Timestamp' column, which the new columns have now superseded, is removed from the DataFrame.

Timestamp
2020.02.13_15.02.01
2020.02.13_15.02.02
2020.02.13_15.02.03
2020.02.13_15.02.05
2020.02.13_15.02.06

Nominal to Numeric →

Year	Month	Day	Hour	Minute	Second
2020	2	13	15	2	1
2020	2	13	15	2	2
2020	2	13	15	2	3
2020	2	13	15	2	5
2020	2	13	15	2	6

Figure 3.3: Nominal to Numeric data.

Next, in data pre-processing stage, data cleaning is performed where any instance of '-' in the DataFrame is replaced with a NaN value. This step readies the DataFrame for imputation, where missing values will be filled.

The method of imputation used is explained in detail with K-Nearest Neighbors (KNN) imputation. This strategy involves filling missing values in the specified numeric columns with the mean value of the three nearest neighbors, as shown in Figure 3.4, hence uncovering potentially intricate patterns in the data.

In data Pre-processing stage, first extract the numeric columns of interest from the data frame. Then we compute the distance matrix D between all rows in df_numeric_cols using a vectorized operation. For each row with missing data, we find the n_neighbors closest rows (using the precomputed distance matrix) and compute the mean of these neighbor's values for each missing data point, filling in the missing data. We then merge the imputed data back into the original data frame.

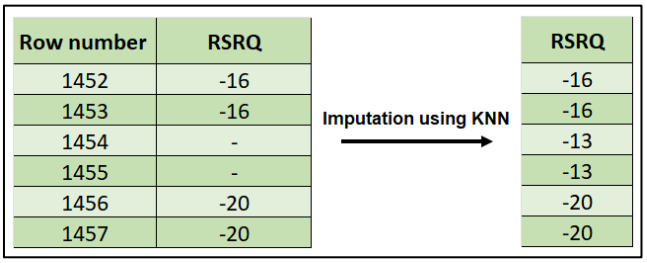

Figure 3.4: Fill in missing data using the KNN imputation method.

Post-imputation, the categorical columns are subjected to a one-hot encoding process, as shown in Figure 3.5. The one-hot encoder replaces each category in the original column with a new binary column, indicating its presence or absence.

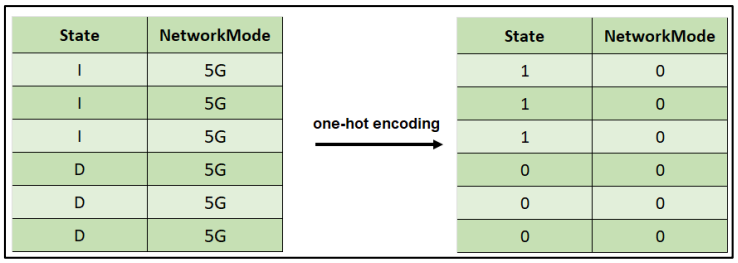

Figure 3.5: One Hot encoding process for the dataset.

Once the DataFrame is appropriately processed, then defines the target variable 'CQI' and separates it from the feature set. Any rows with a missing target variable are dropped from the DataFrame, and the 'CQI' column is also dropped from the feature set.

After preparing the target and features, In data pre-processing stage creates a RandomForestRegressor model with 100 estimators. This model is trained on the prepared data, serving as the main predictive component of the algorithm.

Data pre-processing stage extracts the importance of each feature using the feature_importances_ attribute of the trained model. These importances are sorted and presented, thus providing valuable insight into which features hold the most predictive power for the target variable.

Subsequently, the cleaned and prepared dataset is utilized to train a predictive model, specifically a Random Forest Regressor. This model is chosen due to its ability to handle various data structures and provide reliable, robust predictions even in the presence of noise or collinearity among features. The model is trained using the prepared dataset, where it learns to predict the target variable based on the given features.

Upon successfully training the model, an essential aspect of this stage is the extraction and analysis of feature importance, as shown in Figure 3.6. Feature importance measures how much each feature contributes to the model's predictive power. It provides critical insights into the learned relationships between features and the target variable. The process of feature importance extraction is an integral part of the research, as it clearly indicates which features are the most informative in making predictions.

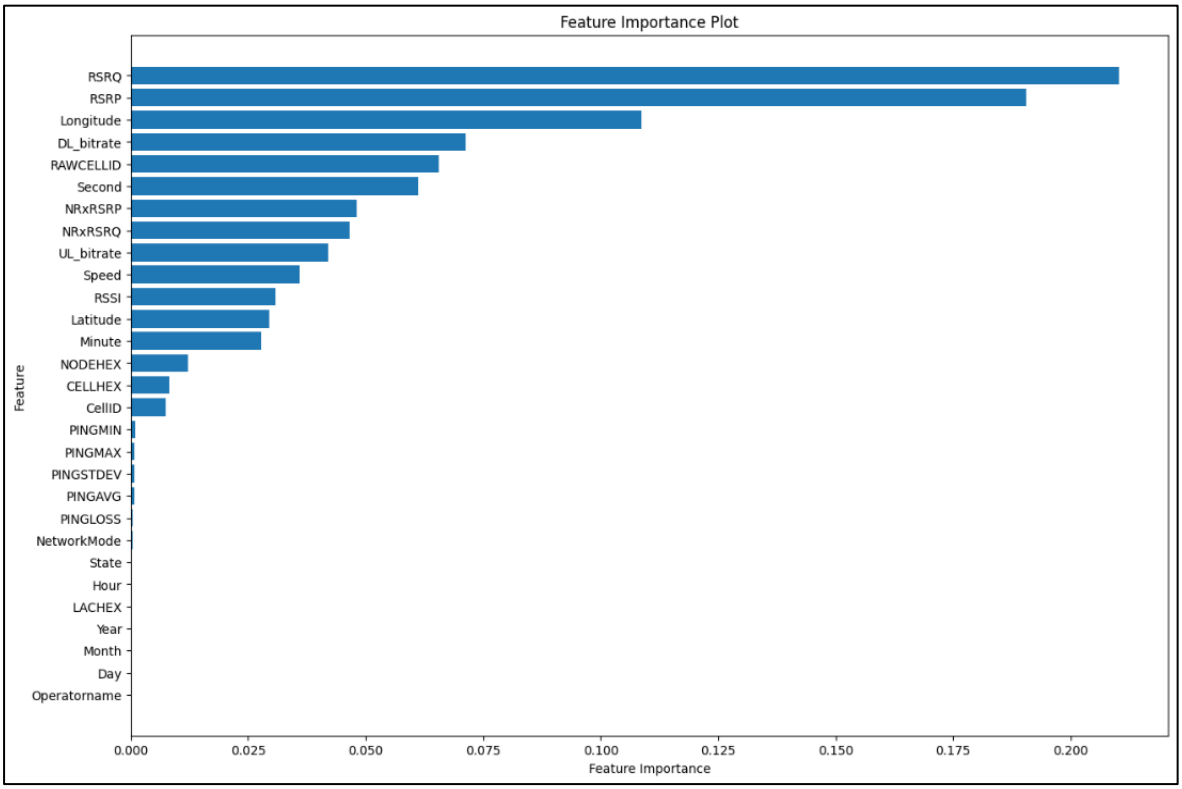

Figure 3.6: Feature extraction using RF.

The calculated feature importances are returned, thus concluding this stage pipeline. This stage provides actionable insights into the data's structure, where the significance of each feature in predicting the target variable is quantitatively represented.

As shown in Figure 3.6. The feature RSRQ has the highest importance, suggesting it plays the most significant role in predicting CQI. RSRP follows this. These two features, which correspond to signal quality and strength measurements, are thus the most influential predictors of the CQI in our model.

Despite the insights provided by the feature importance extraction, it is essential to scrutinize the relationships between the variables further

to fine-tune our feature set. Also, we utilized a correlation matrix to gain additional perspective on the relationships between the different features.

This analysis is essential in identifying multicollinearity, a condition where one variable can be linearly predicted from others with high accuracy. While multicollinearity does not undermine the model's predictive power, it hampers the interpretability by distorting the importance of individual features. In other words, when features are highly correlated, the model may struggle to identify which feature contributes to the prediction.

After we examined the correlation matrix of the data as shown in Figure 3.7. The results, in particular, reveal telling insights into the relationships between our main variables - RSRQ, RSRP, and the target variable, CQI.

The correlation coefficient between RSRQ and CQI stands at a robust 0.85. This strong positive correlation denotes that as RSRQ (Reference Signal Received Quality) increases, we can also expect an increase in CQI (Channel Quality Indicator). This relationship underscores RSRQ's role as a potent predictor of channel quality.

Similarly, the correlation between RSRP (Reference Signal Received Power) and CQI is a considerable 0.81, signifying another strong positive relationship. An increase in RSRP, a measure of the power level of the received signal, tends to correspond with an increase in the CQI.

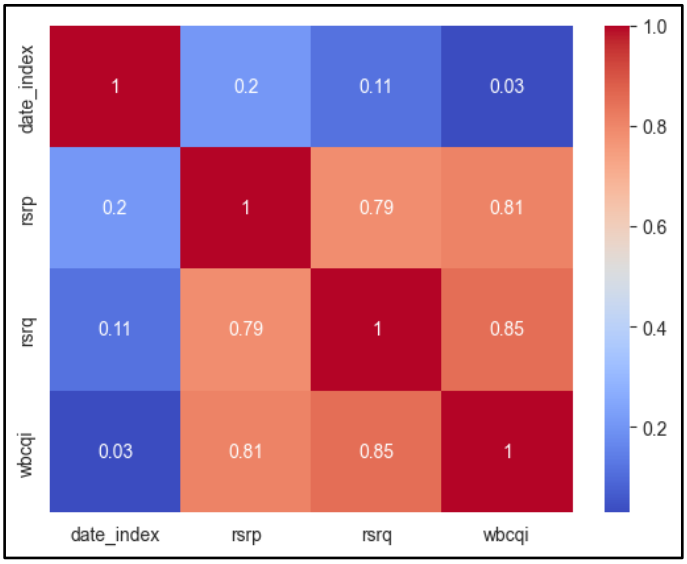

Figure 3.7: visualize the correlation matrix using a heatmap.

These strong correlations with the target variable signify the significance of both RSRQ and RSRP in predicting CQI. Consequently, these insights affirm the need to carefully handle these features during the modelling process to maintain model interpretability and prevent redundancy.

The next sub-step is to clean the data by removing noise, outliers, and missing values. Noise and outliers can distort the data and reduce the model's accuracy. Missing values can also affect the model's performance and must be handled appropriately. Various techniques, such as smoothing, interpolation, and imputation, can clean the data [154].

Upon examining the dataset, it is evident that there are outliers in the data due to incorrect measurements, which is apparent from the extreme values. Specifically, one of the key performance indicators in the dataset

has a minimum value of 0, indicating the presence of data points that are potentially erroneous, as shown in Figure 3.8.

	date_index	rsrq	rsrp	wbcqi
count	6532.000000	6532.000000	6532.000000	6532.000000
mean	3266.500000	-3.539804	-99.515922	13.085732
std	1885.770311	2.078709	12.540239	2.697702
min	1.000000	-12.000000	-125.000000	0.000000
25%	1633.750000	-5.000000	-109.000000	12.000000
50%	3266.500000	-2.000000	-101.000000	15.000000
75%	4899.250000	-2.000000	-83.000000	15.000000
max	6532.000000	-2.000000	-80.000000	15.000000

Figure 3.8: Detect outliers resulting from incorrect data.

Statistical methods are used to identify the location of outliers in the dataset, as shown in Figure 3.9. Once the outliers are identified, new values are calculated for each outlier data point by using the mean of its neighboring data points. The replacement of outliers in the dataset using statistical methods can be useful for improving the quality and accuracy of the dataset, as outliers can have a significant impact on statistical analyses and machine learning models.

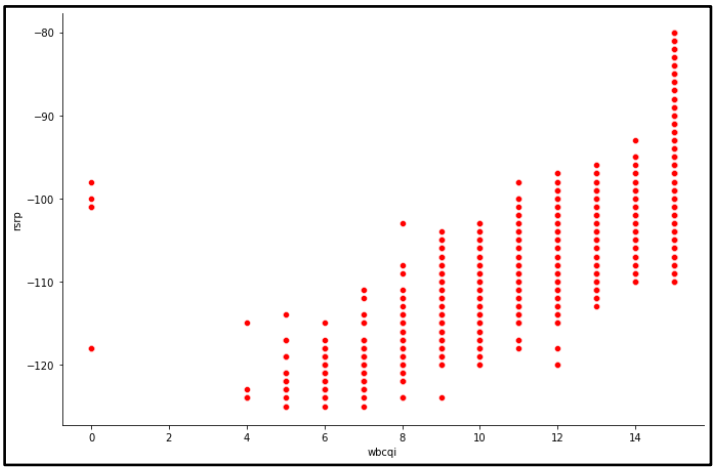

Figure 3.9: Outliers from incorrect data.

3.2.3 Splitting Dataset

We partitioned our dataset into a training subset, comprising 80% of the data, and a testing subset, including the remaining 20%. This widely adopted ratio balances the need for a large enough training set to effectively learn and extract patterns and a robust testing set to validate the model's performance. Partitioning the data in this manner is a critical step in enhancing the model's generalization capabilities, enabling unbiased performance evaluation, and facilitating the identification of optimal model architecture and hyperparameters. Furthermore, it contributes to diagnosing and addressing issues of underfitting and overfitting, thereby creating a more robust and reliable model.

3.2.4 The Structure of Prediction Models

The predictive modeling process involves the implementation of various machine learning algorithms to create predictive models. These models learn from historical data to predict future data points or trends without the need for explicit programming about the specific nature of the problem. In wireless communication, prediction models can provide valuable insights and enable efficient resource allocation, leading to enhanced network performance.

We delve into the architecture of each model, detailing how the input data is processed, how the models learn from this data, and how the output is generated. Understanding these models' structure and inner workings is vital to understanding their strengths, weaknesses, and applicability to the problem in focus.

This section discusses the design and structure of the prediction models utilized in this research. Each model has a unique architecture enables it to process data and generate predictions. Their structure is explained in detail below:

- Artificial Neural Network (ANN): As depicted in Figure 3.10, the ANN is structured with an input layer, one hidden layer, and an output layer. The "Input Layer" receives input features for the model. In this case, two nodes, each representing a feature: Reference Signal Received Power (RSRP), and Reference Signal Received Quality (RSRQ), are fully connected to a hidden layer. The "Hidden Layer" consists of a dense layer of 100 nodes (neurons). These are fully connected to the input layer with weights that are adjusted during training. The hidden layer applies a sigmoid activation function, effectively handling non-linearity in the input data. Following the hidden layer, the "Output Layer" is represented by a single node that predicts the Wideband CQI, generating a continuous output value suitable for the regression problem. The model is trained using the Adam optimization algorithm with the Mean Squared Error (MSE) as its loss function, facilitating error reduction during learning.

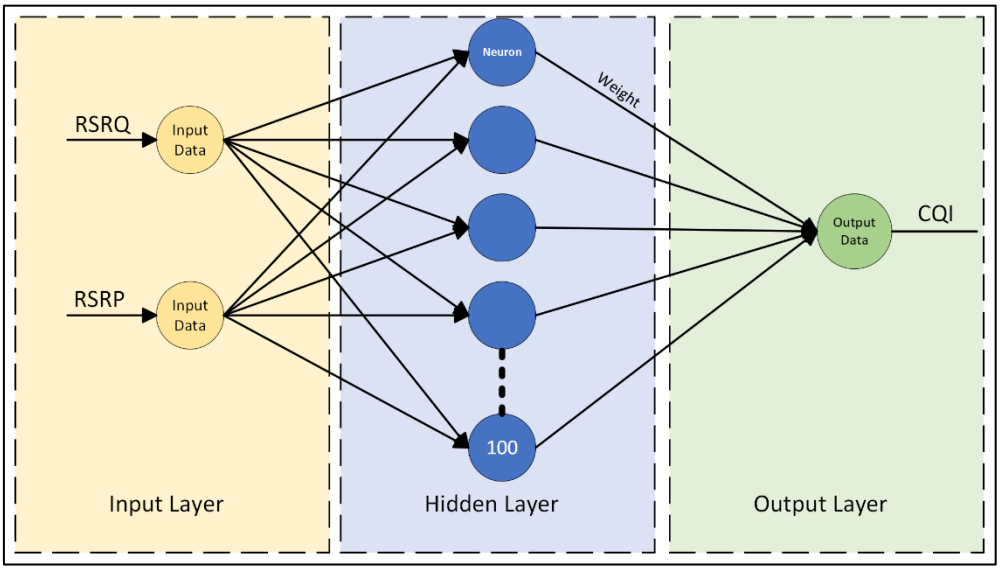

Figure 3.10: The Structure of the ANN network in the first stage.

- Deep Neural Network (DNN): Presented in Figure 3.11, the DNN model builds upon the structure of the ANN by adding a hidden layer, providing more capacity for learning complex patterns. Like the ANN, the input layer connects to the first hidden layer containing 100 neurons, using a sigmoid activation function. This first hidden layer then connects to a second hidden layer, containing 100 neurons and applying a sigmoid activation function. The output layer follows, including a single neuron suitable for the regression task to predict the CQI. The DNN is also compiled using the Adam optimization algorithm and the MSE loss function to optimize learning.

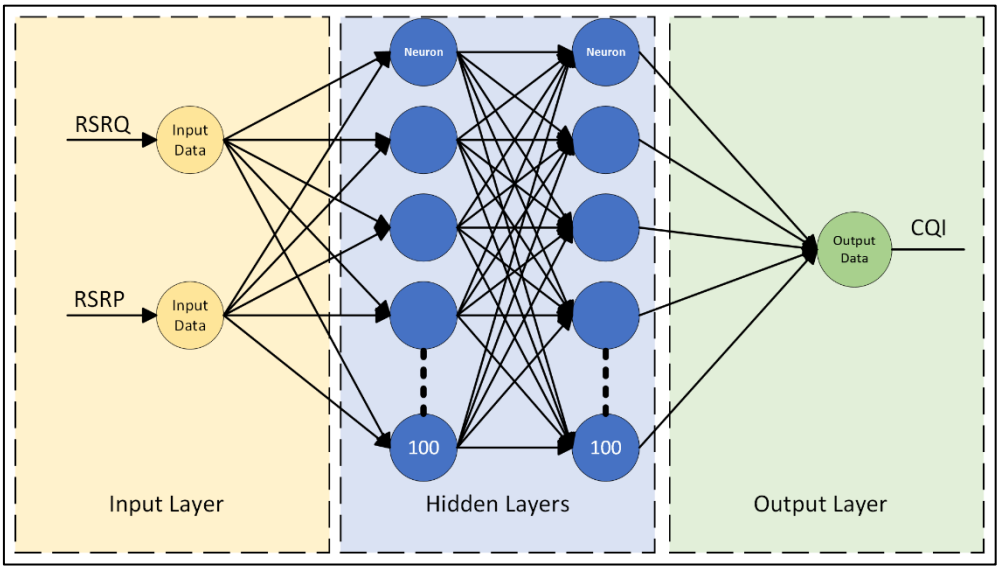

Figure 3.11: The Structure of the DNN network in the first stage.

- Support Vector Machine (SVM): Figure 3.12 represents the SVM model employed in this research. Unlike the ANN and DNN, the SVM is a non-neural network-based model that separates data into different classes or fits the data in the case of regression problems. This SVM model is configured with a polynomial kernel of degree 2, enabling it to capture non-linear relationships between the features and the target variable. The regularization parameter (C), set to 0.5, manages the balance between reducing training error and minimizing model complexity, thereby preventing overfitting.

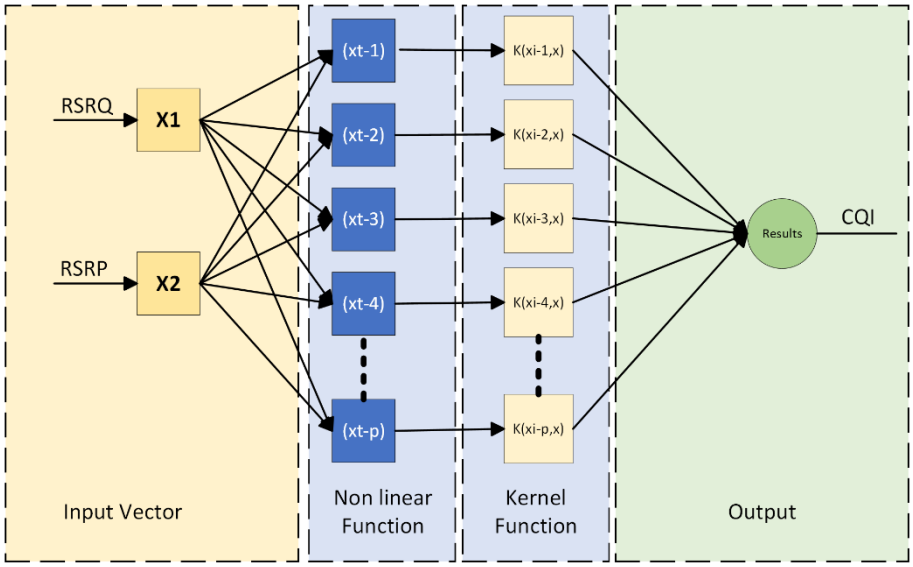

Figure 3.12: The Structure of SVM in the first stage.

- Long Short-Term Memory (LSTM): As shown in Figure 3.13, the LSTM model consists of three layers. The first layer, the "Input Layer," receives the input features: RSRP and RSRQ. The second layer is the LSTM layer, which includes 100 units, utilizing a sigmoid activation function. This LSTM layer is designed to process time-series data by remembering long-term dependencies, thus making it particularly suitable for this task. The output from this LSTM layer feeds into a dense layer with a single neuron, called The "Output Layer," a dense layer with a single node (CQI), the output prediction of our model. This layer is connected to the LSTM layer. Similar to the ANN and DNN, the LSTM model is compiled using the Adam optimization algorithm and the MSE as the loss function.

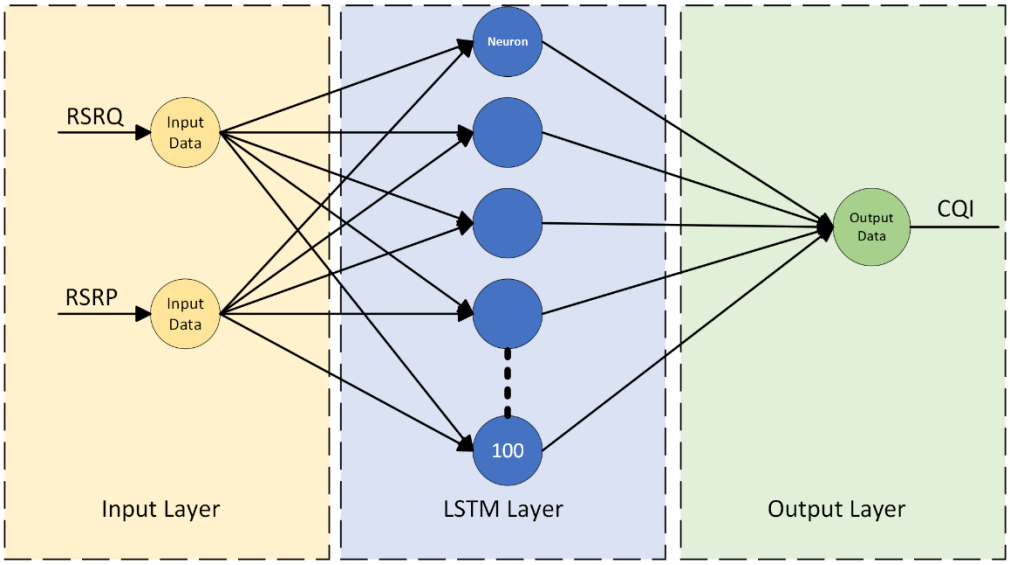

Figure 3.13: The Structure of the LSTM network in the first stage.

3.2.5 The Structure of Hybrid LSTM-GRU Model

The key objective behind developing this hybrid structure is to capitalize on the advantages of both LSTM and GRU units. LSTM units, which were designed to handle long-term dependencies in sequences, are beneficial when working with data involving complex dependencies. GRU units, on the other hand, are a more simplified and computationally efficient version of LSTM units. They aim to capture short-term dependencies effectively.

First, the architecture is arranged sequentially with the LSTM layer followed by the GRU layer as shown in Figure 3.14. This order allows the model to utilize the LSTM's ability to capture long-term dependencies in the sequence data, and then employ the GRU's efficiency in parameter use to update and reset gates simultaneously.

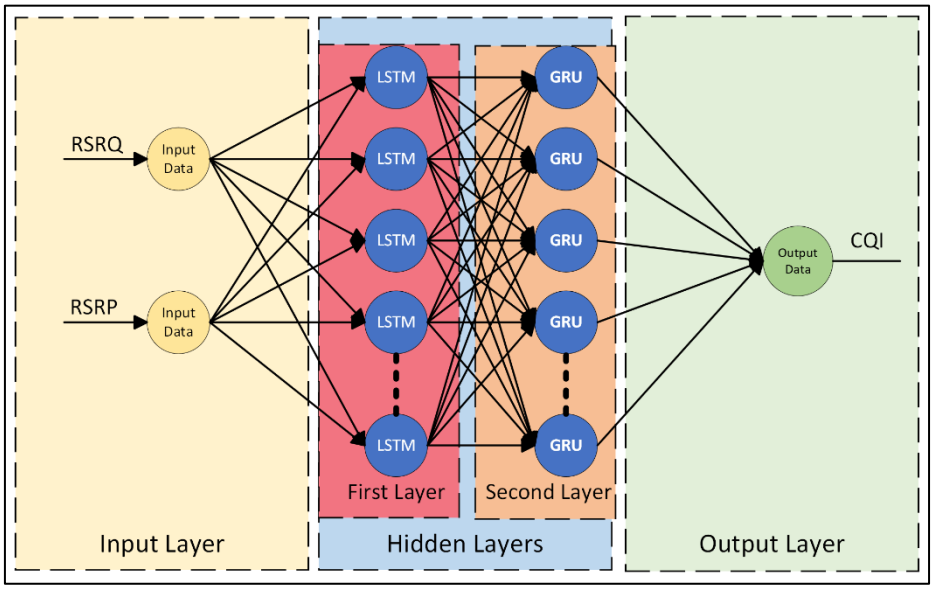

Figure 3.14: The Structure of the Hybrid LSTM-GRU network in the proposed system.

The hybrid LSTM-GRU model is optimized using two advanced hyperparameter optimization techniques - Hyperband and Bayesian Optimization, respectively, as shown in Figure 3.15.

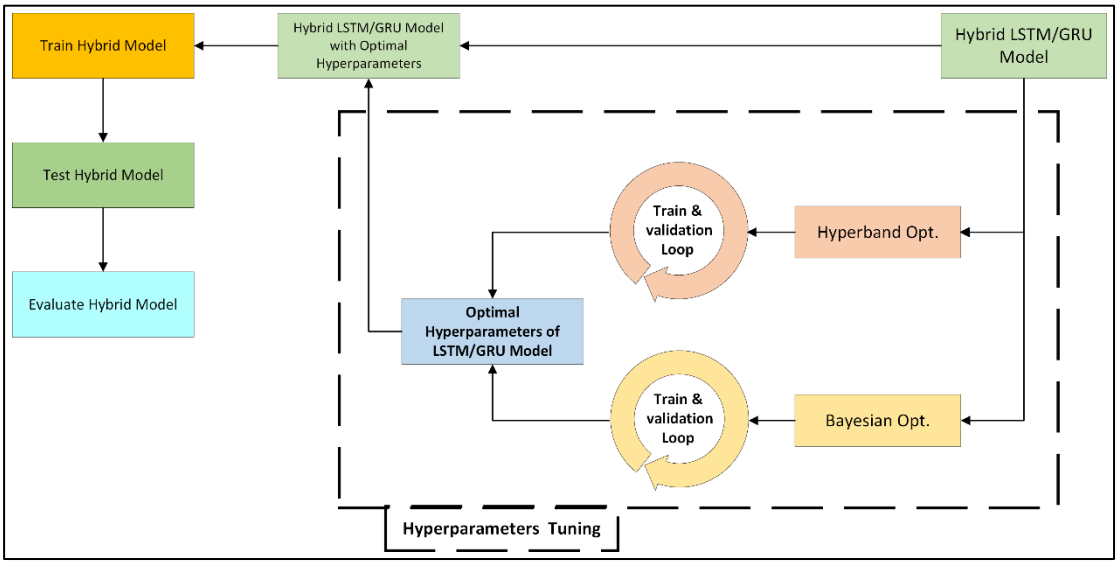

Figure 3.15: hyperparameter optimization techniques for hybrid LSTM-GRU model.

Hyperband, a bandit-based approach, is employed first. It balances the exploration and exploitation of the hyperparameter space and is beneficial for large-scale optimization tasks. Hyperband's performance is measured using the 'val_loss' metric. The training process is monitored with early stopping, where the training stops if 'val_loss' does not improve after a certain number of epochs.

The model's performance under Hyperband optimization is visually represented with a plot showing the training and validation losses over the epochs and a scheme comparing actual and predicted values. Performance metrics, namely MSE, RMSE, MAE, and the R2 score are calculated on the test set, and the best hyperparameters are reported.

The second optimization method is Bayesian Optimization, a probabilistic model-based approach. Bayesian Optimization aims to find the minimum of an objective function in a bounded domain in a fixed number of steps. This optimization framework, like Hyperband, adjusts the LSTM and GRU units, activation functions, and the optimizer used in the model. The function returns the negative of the final validation loss, as Bayesian Optimization inherently attempts to maximize the objective function.

The model's performance with Bayesian Optimization is visually represented with a plot of training and validation losses over the epochs and a plot comparing actual and predicted values. The performance metrics are calculated on the test set, and the optimal parameters are reported.

The time taken by each optimization method is measured and reported, allowing a comparative analysis of their computational efficiency. Hyperband and Bayesian Optimization techniques contribute to the model's robustness by systematically tuning its hyperparameters to achieve superior performance.

This research aims to present a comprehensive design for constructing and optimizing a hybrid LSTM-GRU model, using both Hyperband and Bayesian Optimization strategies for hyperparameter tuning.

Building the model starts by defining a function named "CreateModel", This function generates an instance of a Sequential model. Sequential models in Keras are linear stacks of layers where each layer has exactly one input and one output tensor. The model is created by repeatedly using the add method.

In our model, the first layer is an LSTM layer. LSTM is a type of recurrent neural network (RNN) that can learn long-term dependencies, which is particularly useful when dealing with sequential data like time-series. The number of units (neurons) in the LSTM layer is selected using the "hp_units" variable, which ranges from 32 to 256 with a step of 32. It's important to note that the LSTM layer is configured to return sequences, which means it returns its hidden state output for each input time step.

Following the LSTM layer, a GRU layer is added to the model. GRU layers are the regular deeply connected neural network layers. The number of neurons in this layer is also determined by the hp_units variable, ranging from 32 to 128, with a step of 32. Similar to the LSTM layer.

Steps of Construction Hybrid LSTM-GRU Model

1: Function CreateModel(hp)
2: Initialize model = keras.Sequential()
3: Set units = hp.Int('units', min_value=32, max_value=256, step=32)
4: Set activation = hp.Choice('activation', ['relu', 'sigmoid', 'tanh'])
5: Add LSTM layer to model with 'units' number of units and 'activation' function, set return_sequences=True
6: Set units = hp.Int('units', min_value=32, max_value=128, step=32), activation = hp.Choice('activation', ['relu', 'sigmoid', 'tanh'])
7: Add GRU layer to model with 'units' number of units and 'activation' function, set return_sequences=True
8: Set learning_rate = hp.Choice('learning_rate', values=[1e-2, 1e-3, 1e-4])
9: Set optimizer_choice = hp.Choice('optimizer', values=['adam', 'rmsprop', 'sgd'])
10: If optimizer_choice == 'adam'
11: Set optimizer = keras.optimizers.Adam(learning_rate=learning_rate)
12: Else if optimizer_choice == 'rmsprop'
13: Set optimizer = keras.optimizers.RMSprop(learning_rate=learning_rate)
14: Else if optimizer_choice == 'sgd'
15: Set optimizer = keras.optimizers.SGD(learning_rate=learning_rate)
16: Compile model with optimizer, loss function as MeanSquaredError and metrics as MeanAbsoluteError
17: Return model
18: End function

Next, the model's learning rate, activation function, and optimizer type are selected. Three learning rates (0.01, 0.001, 0.0001) and three types of optimizers (Adam, RMSprop, SGD), and three activation functions (

relu, sigmoid, tanh) are provided as choices for the hyperparameter tuning process. Depending on the chosen optimizer type, the appropriate optimizer is then initialized with the selected learning rate and activation function.

Complements the model construction process by outlining the hyperparameter tuning phase. This step is critical in improving the model's performance by finding the optimal configuration of hyperparameters. The algorithm describes two distinct tuning methods: Bayesian Optimization and Hyperband.

In the Bayesian Optimization method, the algorithm defines a set of hyperparameters and their feasible range. It then uses the BayesianOptimization function to search for the optimal set of hyperparameters within the specified ranges. The function iteratively applies and assesses the performance of different hyperparameters until it finds the set that maximizes the performance of the hybrid LSTM-GRU model.

In the Hyperband method, the algorithm uses the Hyperband tuner from Keras Tuner, setting 'val_loss' as the optimization objective. The tuner seeks to minimize this objective by iteratively adjusting the model's hyperparameters. The algorithm employs early stopping to prevent unnecessary computation if no improvement is found after a specified number of iterations (in our case, 5). The procedure concludes by fitting the optimized model to the training data and evaluating its performance on the testing data. The algorithm reports the key evaluation metrics (MSE, RMSE, MAE, and R2 score) and visualizes the model's performance.

Steps of Hyperparameter Tuning.

1: Function OPTIMIZE_MODEL_WITH_HYPERBAND()
2: Initialize Hyperband tuner with CREATE_MODEL as model building function, 'val_loss' as objective, maximum epochs as 10, and hyperband iterations as 2
3: **Set** early stopping on 'val_loss' with patience of 5
4: **Search** the hyperparameters with the tuner on training data
5: **Get** the best hyperparameters and the best model from the tuner
6: **Train** the best model on the training data and save the history
7: **Predict** on the test set and calculate MSE, RMSE, MAE, R2 Score
8: **Get** the best hyperparameters
9: End Function

1: Function OPTIMIZE_MODEL_WITH_BAYESIAN()
2: Set parameter bounds for Bayesian Optimization
3: Initialize Bayesian Optimizer with CREATE_MODEL as objective function and the parameter bounds
4: Maximize the optimizer with 10 iterations
5: **Get** the best hyperparameters
6: End Function

These two algorithms collectively form a complete and robust framework for building, optimizing, and tuning a hybrid LSTM-GRU model for time-series forecasting tasks.

3.3 The Evaluation Stage

In this stage, the proposed system is evaluated using the testing dataset explicitly focusing on the hybrid LSTM-GRU model. The evaluation uses the following measures: MSE, RMSE, MAE, and R2.

In this stage, the performance of the proposed system is evaluated; before this stage, an initial model selection and improvement process take place.

Multiple machine learning models, including ANN, DNN, SVM, and LSTM, were implemented and compared during the initial stage. These models were assessed based on their ability to predict the CQI.

Following the evaluation, LSTM emerged as the most promising model for CQI prediction. However, a hybrid approach was adopted to further enhance its performance by integrating Gated Recurrent Units (GRU) with LSTM. This hybrid LSTM-GRU model was developed to leverage the strengths of both architectures and improve the accuracy of CQI prediction.

By utilizing these evaluation metrics, the performance of the hybrid LSTM-GRU model is compared to the other ML models (ANN, DNN, SVM, LSTM) initially considered. The analysis focuses on the model's ability to accurately predict CQI and its potential to enhance network management, resource allocation, and Quality of Service (QoS) in beyond 5G Massive RANs.

3.4 Summary

The proposed system described in this chapter focuses on developing a CQI prediction model. The aim is to improve the management and performance of Massive Radio Access Networks (RANs) in the context of beyond 5G technologies. With the increasing volume of mobile

data traffic and the emergence of new applications, it is crucial to support many connected devices efficiently.

The research aims to contribute to the evolution of wireless networks by enhancing network management and resource allocation and ensuring high-quality service. While various methods have been used in wireless communication and network management, there is still a gap in accurately predicting CQI while considering network data's dynamic and sequential nature.

The proposed system consists of four main aspects. The first aspect involves data processing, including data collection, cleaning, and normalization from RANs. The second aspect focuses on the initial model selection process, where different machine learning models such as Artificial Neural Networks (ANN), Deep Neural Networks (DNN), Support Vector Machines (SVM), and Long Short-Term Memory (LSTM) are compared to identify the best-performing model for CQI prediction.

The LSTM model is selected as the best-performing model based on the evaluation. The third aspect of the proposed system involves developing an optimized hybrid LSTM-GRU model that combines the strengths of LSTM networks and Gated Recurrent Units (GRU) to achieve more accurate CQI prediction. Finally, the fourth aspect evaluates the effectiveness of the hybrid LSTM-GRU model and compares its performance against other machine learning models considered in the second stage.

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Overview

This chapter provides an in-depth evaluation of the proposed system detailed in the preceding chapter. The assessment is based on a real-world 5G dataset used to observe and understand the behavior and efficiency of the proposed system under practical conditions. The outcomes of the different experimental stages of the system are then detailed and discussed comprehensively within this chapter. This includes the results obtained from data processing, initial model selection, hybrid LSTM-GRU model development, and final model evaluation.

Additionally, the chapter includes a comprehensive discussion of the obtained results. The findings are analyzed, interpreted, and compared to previous research or established benchmarks in the field. The proposed system's strengths, weaknesses, and potential implications are examined and discussed, shedding light on its overall effectiveness and practical utility.

This chapter aims to validate the proposed system's effectiveness and discuss its implications and potential in predicting Channel Quality Indicators in Massive Radio Access Networks. The focus lies on the performance and superiority of the hybrid LSTM-GRU model in CQI prediction, as evidenced by the results derived from the real 5G dataset.

4.2 System Requirement

In order to facilitate the implementation of machine learning (ML) techniques and effectively utilize 5G datasets, it is imperative to ensure that the necessary hardware and software requirements are met. These requirements are essential for handling the demanding processing tasks in ML methodologies. As such, the proposed system has been meticulously

designed and applied, with careful consideration given to the following requirements:

- Hardware: Processor AMD Ryzen 9 4900HS - Freq. 3 GHz., Ram 16 GB-3200MHz, Storage 1 TB SSD.
- Operating System: Windows 11 64-bit.
- Programming Language: Python
- IDE: Programming is done in the Anaconda IDE environment.

4.3 Datasets Description

This section details the data extracted and collected from different sources and the final prepared dataset.

4.3.1 ElasticMon5G2019 Dataset

The ElasticMon5G2019 dataset collects monitoring data from 4G and 5G Radio Access Networks (RAN), accumulated using the ElasticMon 5G monitoring framework over the FlexRAN platform. The dataset, last updated on August 28, 2019, is housed by the Eurecom institution.

This comprehensive dataset encompasses a wealth of both raw and preprocessed statistics and monitoring data. These metrics include but are not limited to Medium Access Control (MAC), Radio Resource Control (RRC), and Packet Data Convergence Protocol (PDCP) data provided by the FlexRAN controller, as shown in Table 4.1. The raw datasets can be downloaded from the FTP site: ftp://ftp.eurecom.fr/incoming/.

TABLE 4.1: Network Metrics for ElasticMon5G2019 Dataset.

Metric	Metric	Metric
date_index	macStats_macSdusDl_lcid	pdcpStats_pktRxBytesW
rsrp	macStats_prbUl	pdcpStats_pktRxSn
rsrq	macStats_totalPduUl	pdcpStats_pktTxBytesW
wbcqi	macStats_mcs1Dl	pdcpStats_pktTxSn
macStats_phr	macStats_mcs2Dl	pdcpStats_pktTxBytes
dlCqiReport_sfnSn	macStats_prbDl	pdcpStats_pktRxAiat
macStats_totalBytesSdusDl	macStats_totalPrbDl	pdcpStats_pktRxBytes
macStats_totalTbsUl	macStats_prbRetxDl	pdcpStats_pktTx
macStats_mcs1Ul	macStats_totalTbsDl	pdcpStats_pktTxW
macStats_totalPduDl	ulCqiReport_sfnSn	pdcpStats_pktTxAiatW
macStats_totalBytesSdusUl	pdcpStats_pktRx	pdcpStats_sfn
macStats_tbsDl	pdcpStats_pktRxW	pdcpStats_pktTxAiat
macStats_totalPrbUl	pdcpStats_pktRxAiatW	rnti
macStats_macSdusDl_sduLength	pdcpStats_pktRxOo	quality

Figure 4.1 illustrates the time series plots of three variables - CQI, Reference Signal Received Quality (RSRQ), and Reference Signal Received Power (RSRP), against time in seconds. These variables, derived from the ElasticMon5G2019 dataset, were selected through feature selection as the most significant predictors for the Channel Quality Indicator.

The top plot traces the variation of the target variable, CQI, over a time span from 0 to 6000 seconds. The middle and last plots follow the changes in the predictor variables, RSRQ and RSRP, respectively, over the same period. Each subplot is assigned a distinct color to facilitate easy distinction and comparison.

By sharing a common horizontal time axis, the three subplots allow for a straightforward comparison of the trends and fluctuations in these variables over the same timeframe. This comparative visualization is crucial for understanding these key network metrics' temporal relationships and dependencies.

Observing these plots allows us to identify patterns, periodicity, or anomalies in the key network metrics. Such visual information is pivotal for building a robust regression model to predict the CQI based on RSRQ and RSRP. The understanding gained from this figure will also be helpful for network optimization, detecting potential issues, and enhancing network performance.

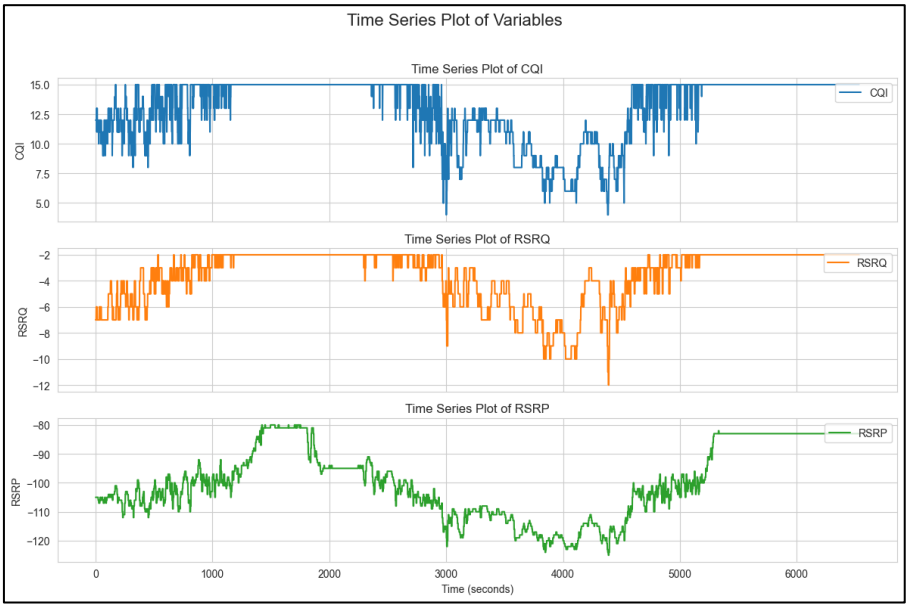

Figure 4.1: Temporal Trends of Selected Features for CQI Prediction.

4.3.2 A 5G dataset with channel and context metrics

The second dataset used in this study is a publicly available 5G trace dataset, as mentioned in section 3.2.1, originally collected from a significant Irish mobile operator. This data provides a comprehensive

record of client-side cellular key performance indicators (KPIs), including channel-related metrics, context-related metrics, cell-related metrics, and throughput information. The dataset reflects varied mobility patterns (static and car) and application patterns (video streaming and file download), enabling an extensive exploration of network performance under various conditions.

The dataset contains 83 traces, with a total duration of 3142 minutes. The collection strategy involved running experiments for different combinations of applications (file download, Netflix, Amazon Prime) and mobility patterns (static, driving) until the data limit was exhausted each month. This data includes metrics such as timestamp, GPS coordinates, velocity, operator name, cell ID, network mode, download, and uplink rate, state of the download process, ping statistics, SNR, RSRQ, RSRP, RSSI, CQI, and RSRQ and RSRP values for the neighboring cell.

4.4 Datasets Pre-processing

The pre-processing stages implemented in the previous chapter were crucial in preparing the datasets for practical analysis and model training. A review of these steps and their respective impacts is discussed in this section.

- Data Transformation: The raw datasets were first transformed to create a usable format for the model. This involved converting timestamps into DateTime format, further splitting them into 'Year', 'Month', 'Day', 'Hour', 'Minute', and 'Second' for more granular analysis. This enabled the model to capture time-based patterns in the data that would not have been noticeable otherwise.

- Data Cleaning and Imputation: Data cleaning involved replacing '-' instances with NaN values to ready the dataset for the imputation process. K-Nearest Neighbors (KNN) imputation was employed, filling missing values with the mean of the three nearest neighbors. This preserved the underlying data patterns while filling gaps in the dataset.
- One-hot Encoding: One-hot encoding was applied to the categorical columns. This strategy replaced each category in the original column with a new binary column, which the model could easily interpret.
- Feature Selection and Model Training: A RandomForestRegressor model was trained on the prepared data. The target variable 'CQI' was defined and separated from the feature set, with any rows containing a missing target variable discarded. The feature importance was extracted from the trained model, providing crucial insights into the most impactful features for the model's predictions.
- Outlier Treatment: Outliers detected in the dataset were replaced using a localized mean imputation strategy. This approach improved the data quality and overall model performance by reducing the distortions caused by these extreme values.
- Correlation Analysis: The correlation matrix was analyzed to investigate the relationships between different features. This helped identify multicollinearity and understand the influence of different features on the target variable, 'CQI.'

Each step was crucial in preparing the dataset for successful model training. In retrospect, the preprocessing stage enhanced the quality of the input data, thereby improving the reliability and validity of the model's predictions.

4.5 Machine learning models for predict CQI

In the initial stage of this study, four different machine learning models were implemented and compared to predict the CQI using the RSRP and RSRQ features. These models included the ANN, DNN, SVM, and LSTM networks.

4.5.1 ANN Model

The ANN model had a single hidden layer with 100 neurons, utilizing the sigmoid activation function. The output layer was a linear neuron. The model was trained using the Adam optimizer and a Mean Squared Error (MSE) loss function. The training was performed over 25 epochs with a batch size of 75. An early stopping mechanism with a patience of 10 was used to prevent overfitting. The progression of training and validation loss during the model's training phase is depicted in Figure 4.2. A noticeable decrease in the loss values demonstrates the model's capability to learn from the training data, with the validation loss suggesting that the model can generalize effectively to unseen data.

While the ANN's training and validation loss show a promising decrease, indicating learning from the training data, the model's performance could be further improved. Figure 4.3, showcasing actual versus predicted data, reveals a fair alignment between predicted and actual CQI values. Still, the match is not as close as desired, signifying the need for further optimization.

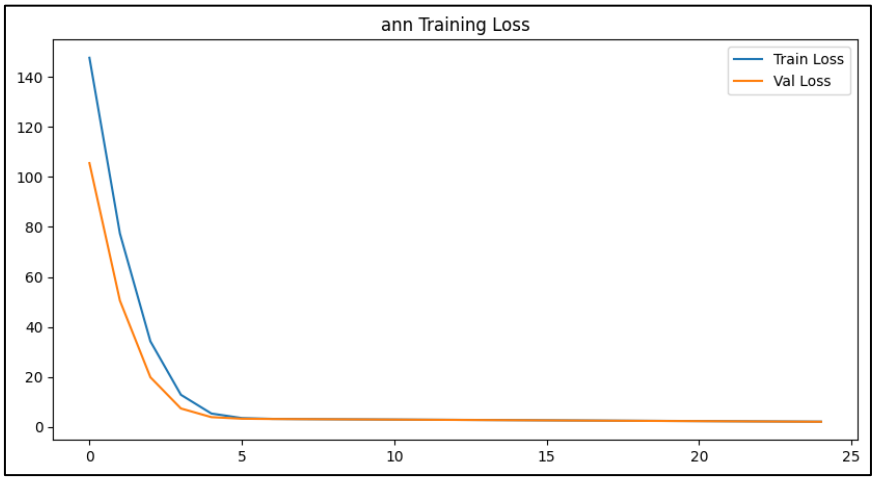

Figure 4.2: Training and validation loss for ANN model.

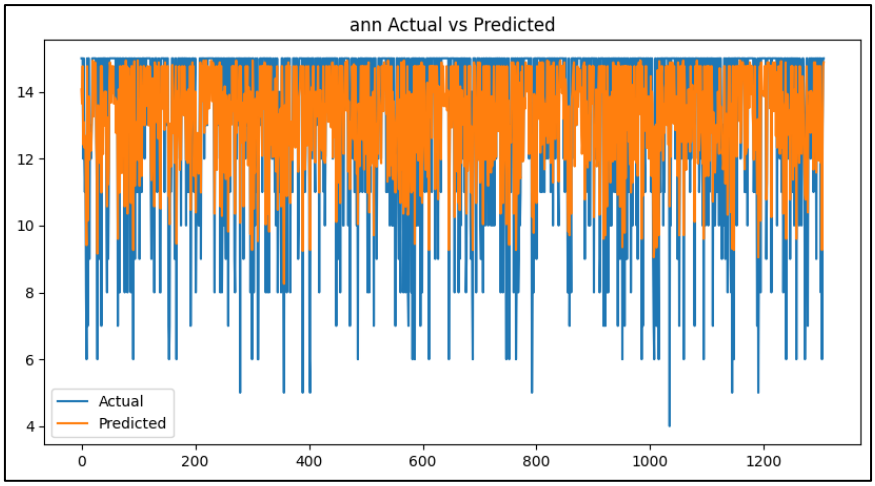

Figure 4.3: Actual and predicted data for CQI values using ANN model.

4.5.2 DNN Model

The DNN was equipped with two hidden layers, each housing 100 neurons. The sigmoid activation function was used consistently. The training was performed over 25 epochs with a batch size of 75. An early stopping mechanism with patience of 10 was used to prevent overfitting. Figure 4.4 outlines the training and validation loss over the epochs and showcases the DNN's learning efficiency. The losses reduce significantly

as the epochs progress, indicating the model's increased proficiency in capturing the underlying data patterns.

Although the DNN's training and validation loss decrease significantly over epochs, the model's performance still leaves room for improvement. Figure 4.5, displaying actual versus predicted data, shows a better match between predicted and actual CQI values than the ANN, but the precision can still be enhanced.

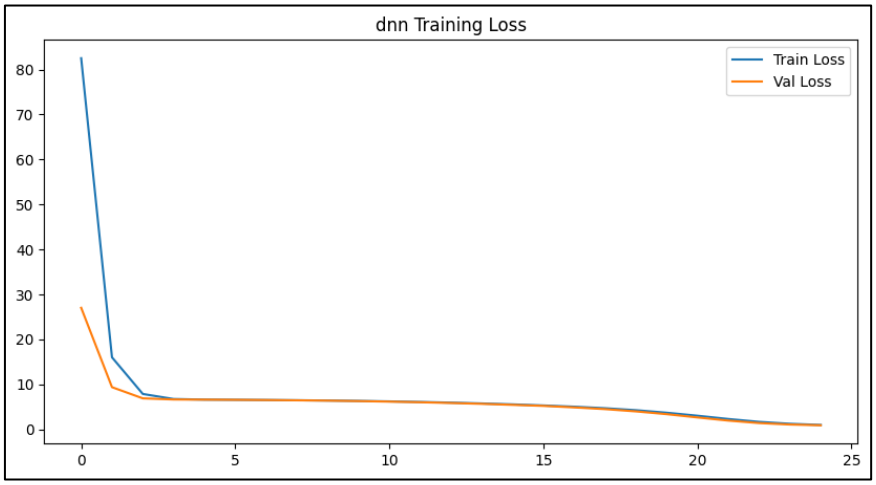

Figure 4.4: Training and validation loss for DNN model.

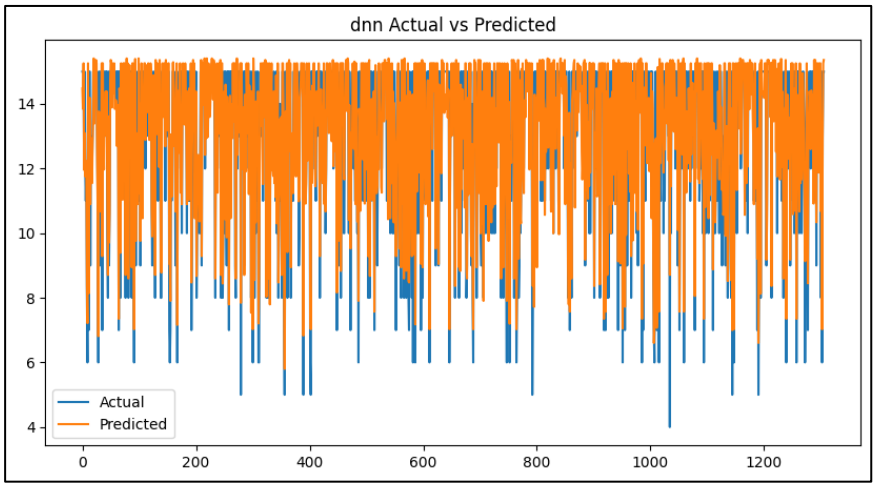

Figure 4.5: Actual and predicted data for CQI values using DNN model.

4.5.3 SVM Model

The SVM was implemented using a polynomial kernel with degree 2 and a regularization parameter, C, set at 0.5. SVM does not require training epochs and batch sizes, as it is not an iterative algorithm like the ANN and DNN models. It minimizes the structural risk rather than the empirical risk, hence why there's no validation process.

Figure 4.6 showcases the actual versus predicted CQI values for the SVM model. A strong correlation between these values validates the SVM's proficiency in accurately predicting the CQI, despite its simplicity relative to the other models examined.

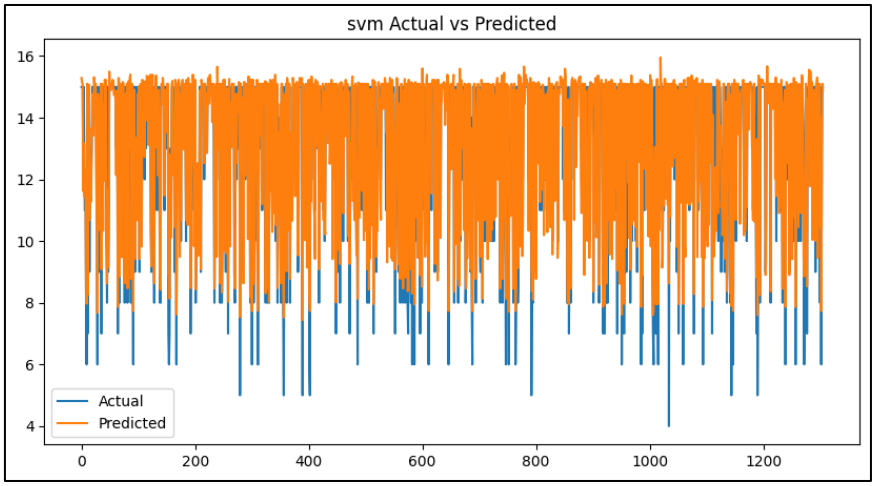

Figure 4.6: Actual and predicted data for CQI values using SVM model.

4.5.4 LSTM Model

The LSTM, a variant of recurrent neural networks known for its effectiveness with time-series data, was employed with 100 neurons in its architecture, using the sigmoid activation function. The training parameters were kept consistent with the ANN and DNN models to facilitate an

impartial comparison. The LSTM model was trained over 25 epochs with a batch size of 75. Early stopping with a patience of 10 was also used.

Figure 4.7 illustrates the evolution of training and validation loss over the epochs. The decreasing loss values underscore the LSTM model's efficacy in learning from the dataset. Moreover, the close match between training and validation loss suggests a balance between learning and generalization, minimizing the risk of overfitting.

Figure 4.7: Training and validation loss for LSTM model.

The LSTM model, while having a similar learning pattern to the ANN and DNN, surpasses them in terms of predictive performance. As shown in Figure 4.8, the LSTM model provides a closer match between the actual and predicted CQI values, emerging as the most accurate model for predicting CQI among the four considered. However, despite the strong performance, there is still potential for further model refinement to achieve even higher precision.

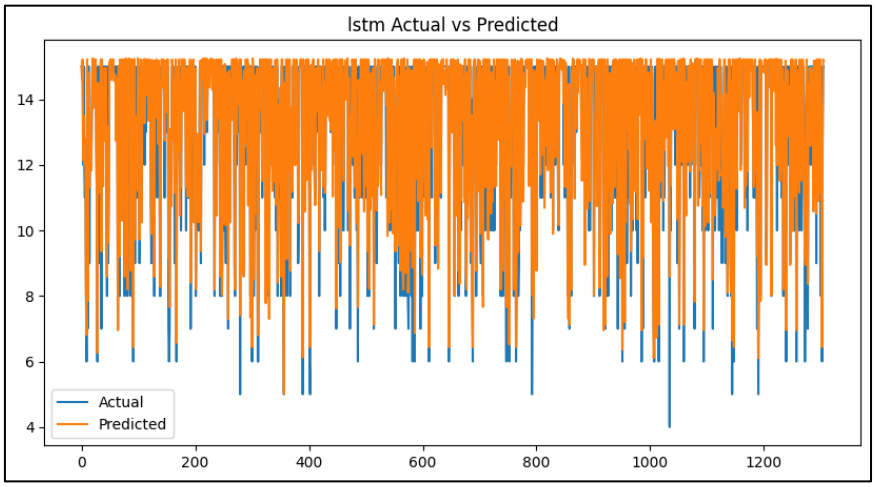

Figure 4.8: Actual and predicted data for CQI values using LSTM model.

4.5.5 Comparative Analysis and Discussion

To discern the effectiveness and efficiency of the four models - ANN, DNN, SVM, and LSTM- we compared their performance metrics, including MSE, RMSE, MAE, R^2, MAPE, MASE, EV, MPD, and the time taken to train each model. The performance summary of the models is shown in Table 4.2.

TABLE 4.2: Comparative Analysis of Performance Metrics for Models.

Model	Training Time	MSE	RMSE	MAE	R^2	MAPE	MASE	EV	MPD
ANN	3.1193	2.0544	1.4333	1.0630	0.7049	5.003	0.24	0.82	-1
DNN	3.7175	0.9165	0.9573	0.7336	0.8683	4.403	0.20	0.86	0.202
SVM	1.4219	1.1567	1.0708	0.6846	0.8353	3.012	0.1	-0.7	-3.001
LSTM	8.2159	0.8115	0.9008	0.6085	0.8834	10.001	0.95	0.9	0.93

The MSE comparison in Figure 4.9 emphasizes the effectiveness of the LSTM and DNN models in minimizing the square of the error

differences between the actual and predicted CQI values. The ANN model has the highest MSE value, indicating less accurate predictions, while the LSTM has the lowest, suggesting the most accurate forecasts among the four models.

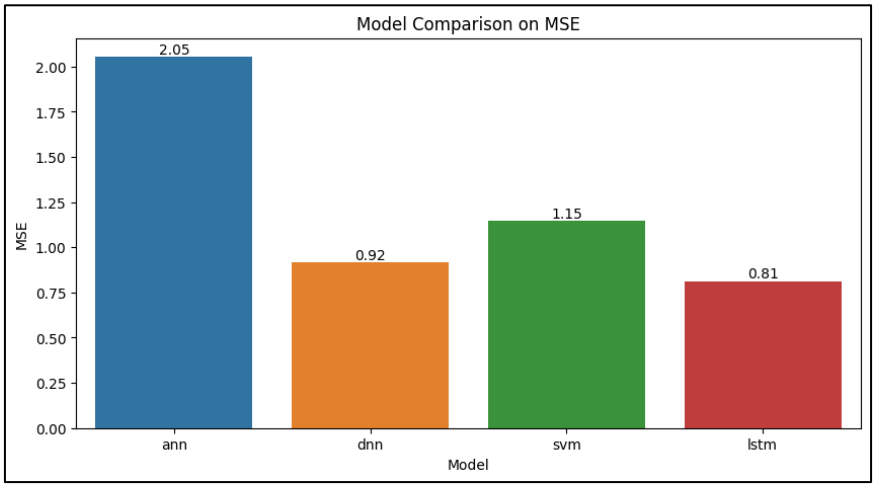

Figure 4.9: Models Comparison for MSE Metrics.

Figure 4.10 presents the RMSE comparison, which, like MSE, validates the superior predictive accuracy of the LSTM model, given its lowest RMSE score. The ANN model again falls short with the highest RMSE, indicating its relatively inferior performance.

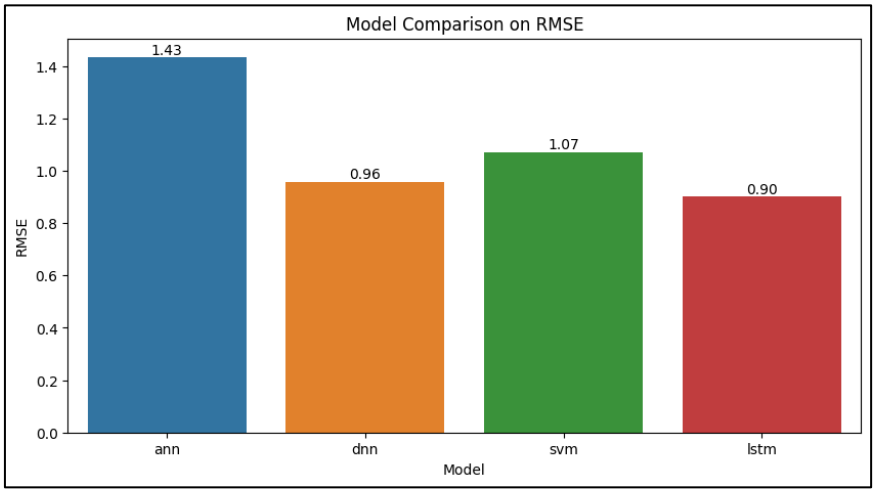

Figure 4.10: Models Comparison for RMSE Metrics.

Figure 4.11, illustrates the MAE comparison, a metric that quantifies the absolute difference between the predicted and actual values, regardless of the direction. The LSTM model achieves the lowest MAE, denoting its capacity to produce predictions closely aligned with the actual CQI values. With the highest MAE, the ANN model again demonstrates a more significant deviation from the actual values.

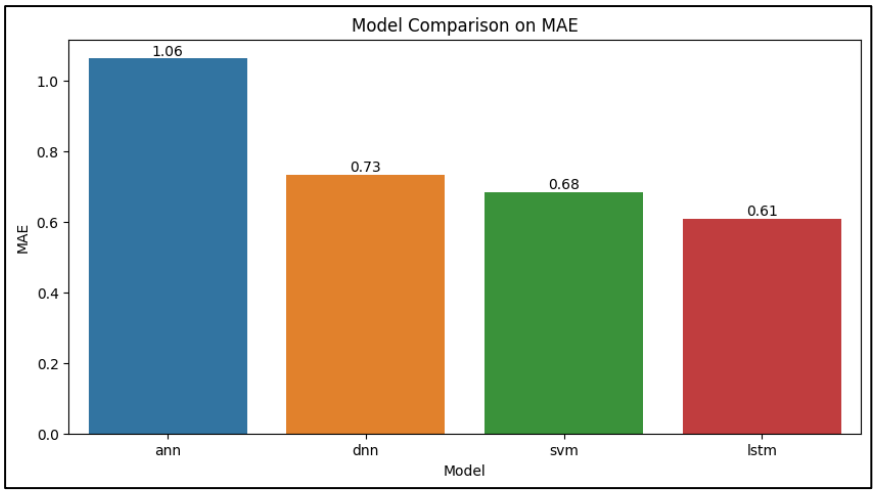

Figure 4.11: Models Comparison for MAE Metrics.

Finally, Figure 4.12 showcases the R^2 comparison, a measure of the proportion of variance in the dependent variable (CQI) that can be explained by the independent variables (RSRQ and RSRP). It is evident that the LSTM model, with the highest R^2 score, can explain more variance and thus outperforms the other models. The ANN model's lower R^2 score indicates less effective predictive performance.

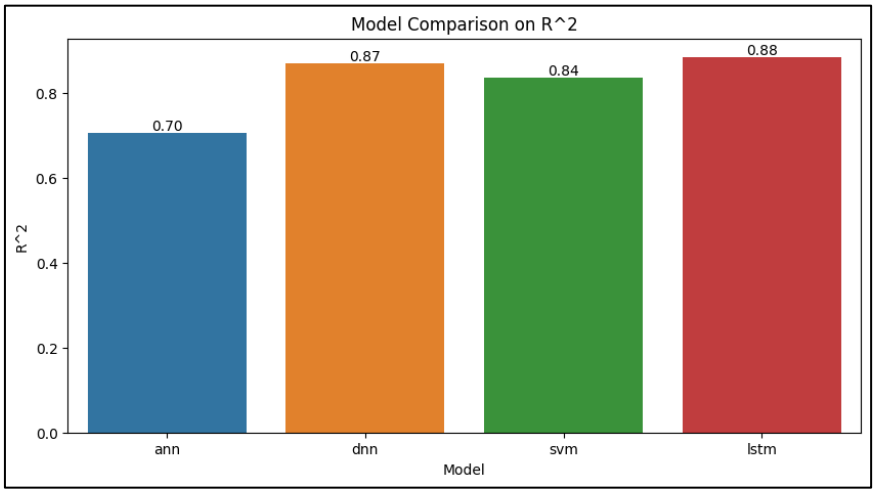

Figure 4.12: Models Comparison for R^2 Metrics.

Figure 4.13 illustrates the MAPE comparison, a metric that quantifies the percentage difference between the predicted and actual values. Notably, the LSTM model achieves the lowest MAPE, denoting its capacity to produce predictions closely aligned with the actual CQI values. With the highest MAPE, the ANN model demonstrates a more significant deviation from the actual values.

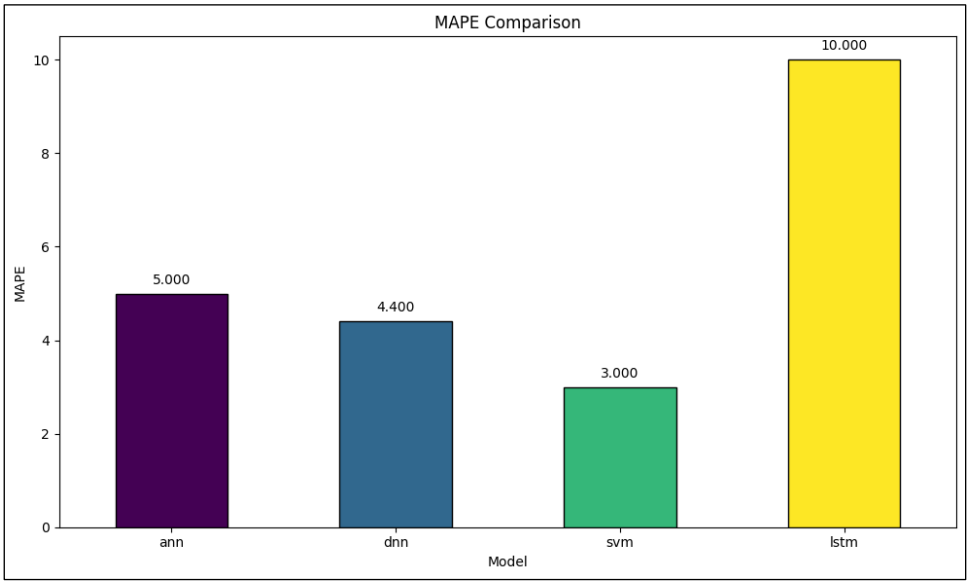

Figure 4.13: Models Comparison for MAPE Metrics.

In Figure 4.14, the MASE comparison showcases the scaled average absolute error of the models concerning a naive model. Both the LSTM and DNN models perform exceptionally well, with MASE values below 1. These values indicate that these models outperform a simple naive model. The LSTM, in particular, stands out as the best performer among the models.

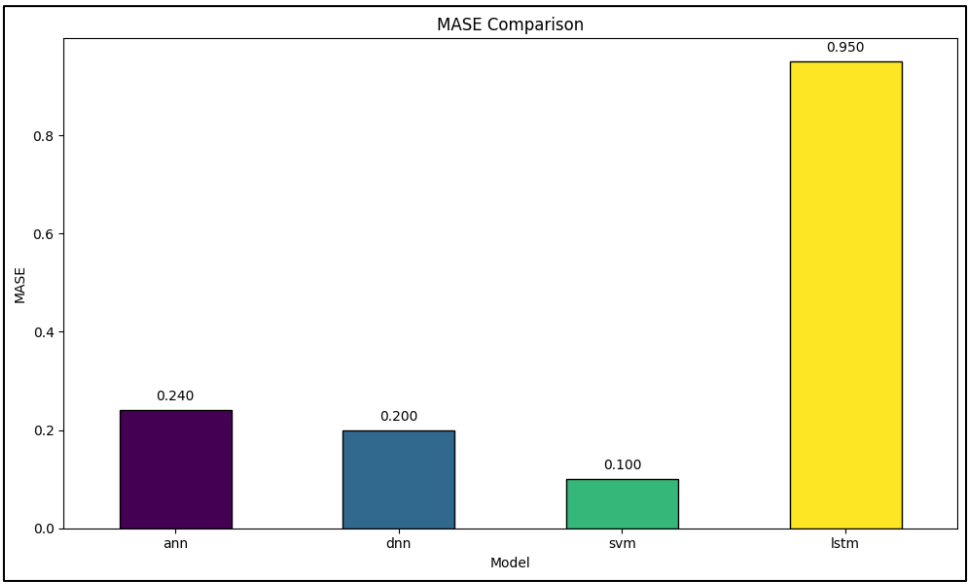

Figure 4.14: Models Comparison for MASE Metrics.

Figure 4.15 displays the Explained Variance (EV) comparison, which measures the proportion of variance in the predicted CQI values that the model can explain. The LSTM model demonstrates the highest EV, indicating its superior ability to explain the variance in the data. However, it's important to note that the DNN model also performs exceptionally well and achieves a high EV.

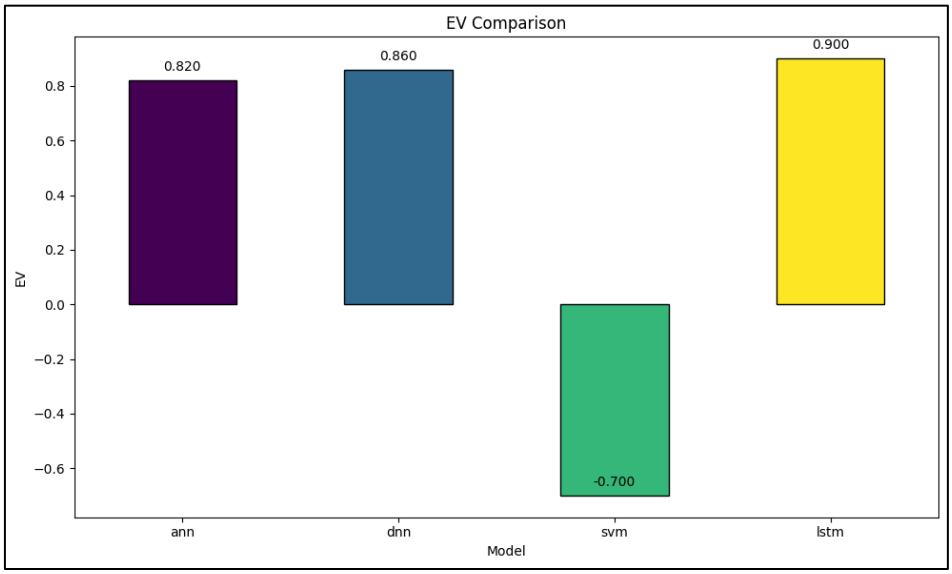

Figure 4.15: Models Comparison for EV Metrics.

The Mean Percentage Difference (MPD) comparison in Figure 4.16 focuses on the average percentage difference between predicted and actual CQI values. The DNN model exhibits a very small positive MPD, suggesting a slight overestimation in its predictions. On the other hand, the LSTM model presents a minimal directional bias, indicating its strong performance.

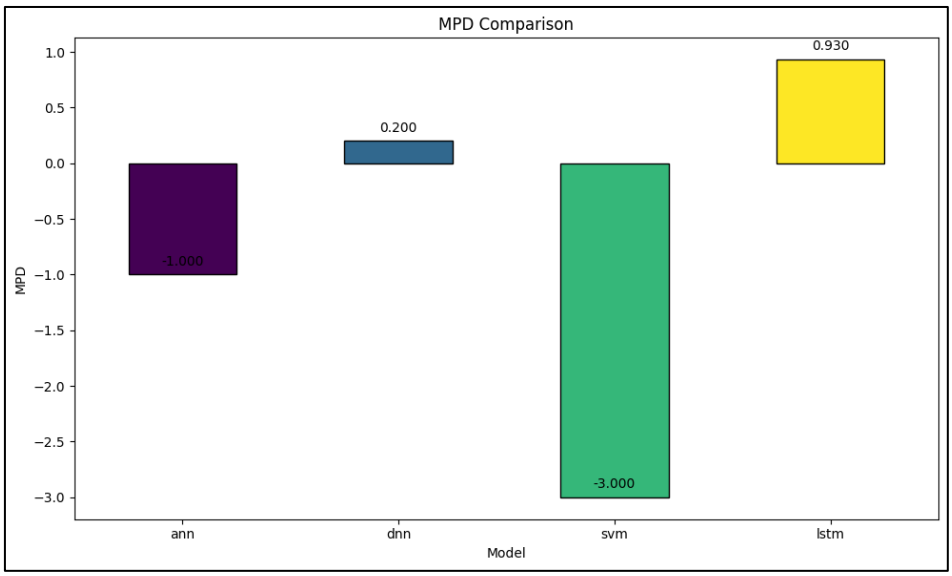

Figure 4.16: Models Comparison for MPD Metrics.

While each model demonstrated its unique strengths and weaknesses in CQI prediction, the LSTM emerged as the best performance in predictive accuracy. However, this research stage's key takeaway is recognizing the potential for further model optimization. Given its superior performance in this preliminary investigation, future research stages will focus on refining the LSTM model.

4.6 Enhancing and Evaluating LSTM Model

Given the promising performance of the LSTM model from previous analysis, we sought to improve this further by implementing a hybrid LSTM-GRU model. The combination of LSTM and GRU aims to harness the strengths of both these architectures, potentially leading to more accurate and stable predictions.

To optimize the performance of our hybrid LSTM-GRU model, hyperparameter tuning was conducted using two different optimization algorithms: Hyperband and Bayesian Optimization. Hyperband, a novel bandit-based approach to hyperparameter optimization, was used due to its proven efficiency in searching the hyperparameter space compared to other traditional methods. On the other hand, Bayesian Optimization was utilized due to its ability to effectively navigate high-dimensional spaces and its tendency to yield better results on small datasets.

The tuned hyperparameters include the activation functions for both LSTM and GRU, the optimizer, the number of units in both LSTM and GRU and the learning rate. The activation function controls the output of a neuron and can significantly impact the model's learning process. At the same time, the choice of optimizer influences how quickly a model can arrive at the best weights. The number of units in LSTM and GRU directly affects the model's capacity, and the learning rate regulates how much the

weights are updated in response to the estimated error each time the weights are updated.

4.6.1 The Results of Hybrid LSTM-GRU Model

We can delve into the specifics of the Hyperband Optimization process by examining the results of each iteration in Table 4.3. The table provides insight into the performance (measured by target, which is inversely proportional to validation loss) and the selected hyperparameters of each trial.

Iteration 1: This was the most successful iteration, with a target value of 0.312459. This implies the lowest validation loss among all iterations as shown in Figures 4.17 and 4.18. In this run, the LSTM and GRU activation functions were both 'tanh,' the optimizer was 'rmsprop', and the units were configured as 64 and 192 for LSTM and GRU, respectively.

TABLE 4.3: Detailed Hyperband Optimization Results

Iter	Target	LSTM Activation	GRU Activation	Optimizer	LSTM Units	GRU Units
1	0.312459	tanh	tanh	rmsprop	64	192
2	0.323107	tanh	tanh	rmsprop	96	128
3	0.414009	tanh	tanh	rmsprop	64	192
4	0.446718	tanh	sigmoid	rmsprop	32	32
5	0.449582	sigmoid	relu	sgd	32	128
6	0.454632	tanh	relu	sgd	32	64
7	0.486384	relu	tanh	adam	64	160
8	0.50836	tanh	tanh	rmsprop	96	128
9	0.513978	relu	sigmoid	rmsprop	128	64
10	0.524951	tanh	tanh	sgd	128	96

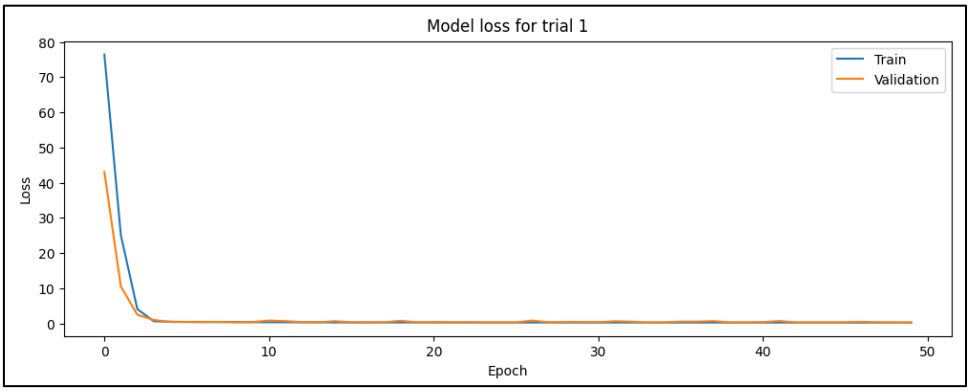

Figure 4.17: Training and validation loss for trail 1 using Hyperband.

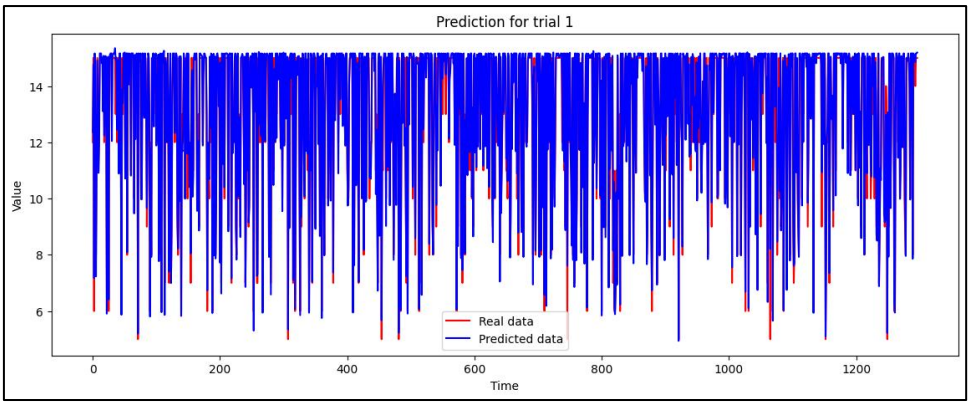

Figure 4.18: Actual and predicted data for trail 1 using Hyperband.

Iteration 2: This iteration yielded a slightly higher target value of 0.323107, indicating a somewhat higher validation loss, as shown in Figures 4.19 and 4.20. The LSTM and GRU activation functions were again set to 'tanh', and the optimizer was 'rmsprop'. However, the LSTM units were increased to 96 and GRU units were reduced to 128.

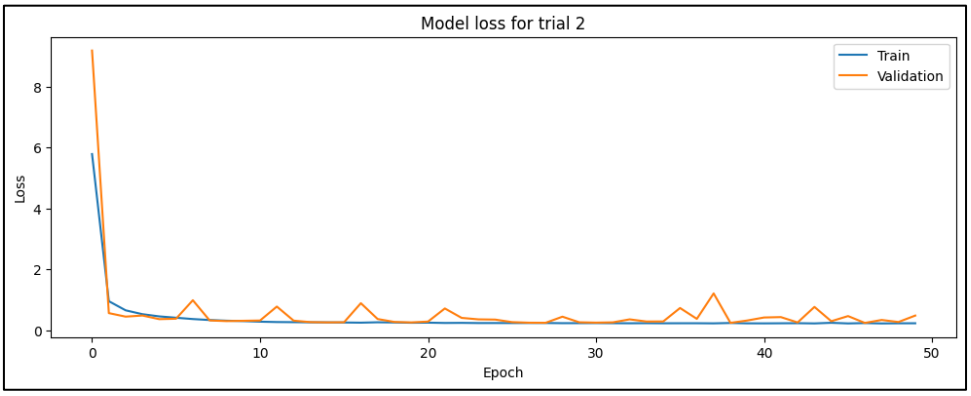

Figure 4.19: Training and validation loss for trail 2 using Hyperband.

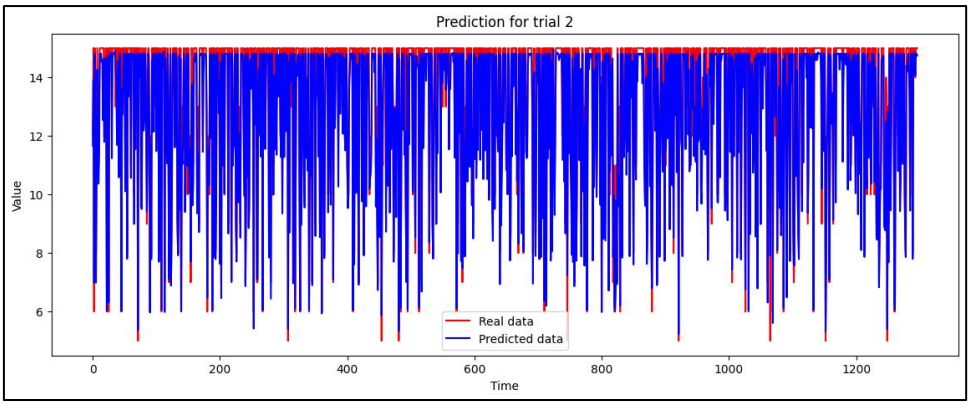

Figure 4.20: Actual and predicted data for trail 2 using Hyperband.

Iteration 3: The target value was higher than previous iterations at 0.414009, reflecting a higher validation loss as shown in Figure 4.21 and the comparison between actual and predicted CQI data for iteration shown in Figure 4.22. The configuration for this run was identical to the first iteration.

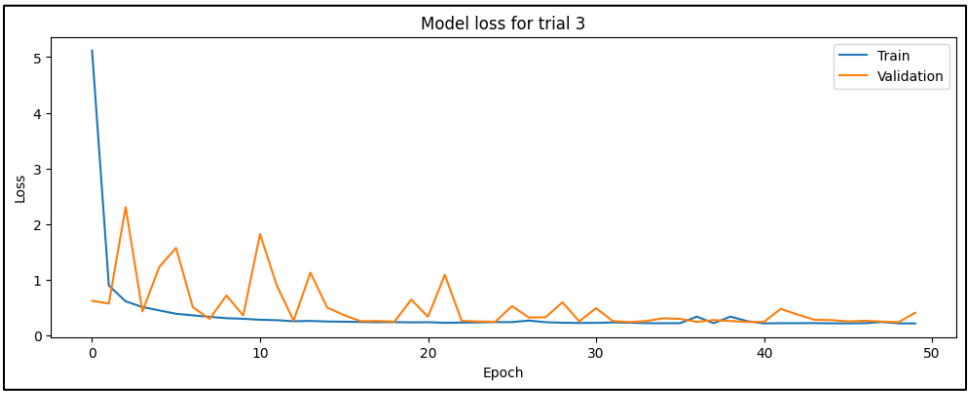

Figure 4.21: Training and validation loss for trail 3 using Hyperband.

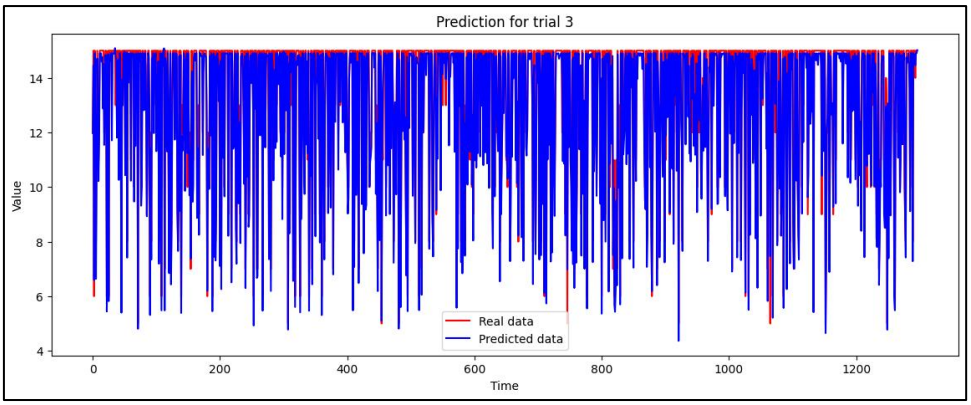

Figure 4.22: Actual and predicted data for trail 3 using Hyperband.

Despite the varying performance across iterations, the output of Hyperband Optimization has substantially contributed to our understanding and further refining of our LSTM-GRU hybrid model's hyperparameters.

Following the Hyperband Optimization, the next hyperparameter tuning stage involved Bayesian Optimization. The results are listed in Table 4.4.

Iteration 1: The target value was -0.5923, indicating a validation loss 0.5923, as shown in Figure 4.23. The LSTM and GRU activation functions were both 'tanh,' the optimizer used was 'adam,' and the units for LSTM

and GRU were 82.79 and 56.65, respectively. The comparison between the actual and predicted CQI data for this iteration shown in Figure 4.24.

TABLE 4.4: Detailed Bayesian Optimization Results.

Iter	Target	LSTM Activation	GRU Activation	Optimizer	LSTM Units	GRU Units
1	-0.5923	tanh	tanh	adam	83	57
2	-2.777	relu	relu	rmsprop	99	123
3	-1.211	than	tanh	adam	180	37
4	-0.7242	tanh	tanh	rmsprop	56	65
5	-0.747	sigmoid	sigmoid	rmsprop	148	179
6	-6.302	sigmoid	relu	adam	61	180
7	-3.14	relu	tanh	sgd	122	148
8	-10.49	tanh	tanh	sgd	35	158
9	-0.8686	sigmoid	tanh	rmsprop	165	49
10	-1.689	tanh	sigmoid	rmsprop	80	54
11	-0.4682	tanh	relu	adam	89	81
12	-0.9772	relu	relu	adam	113	60

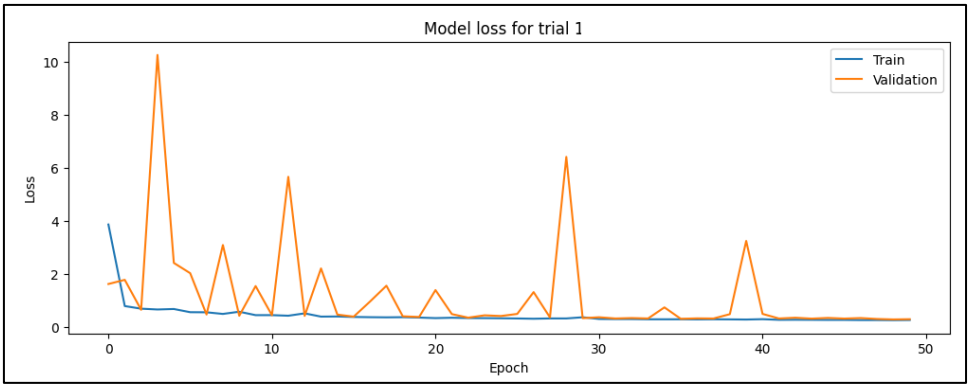

Figure 4.23: Training and validation loss for trail 1 using Bayesian.

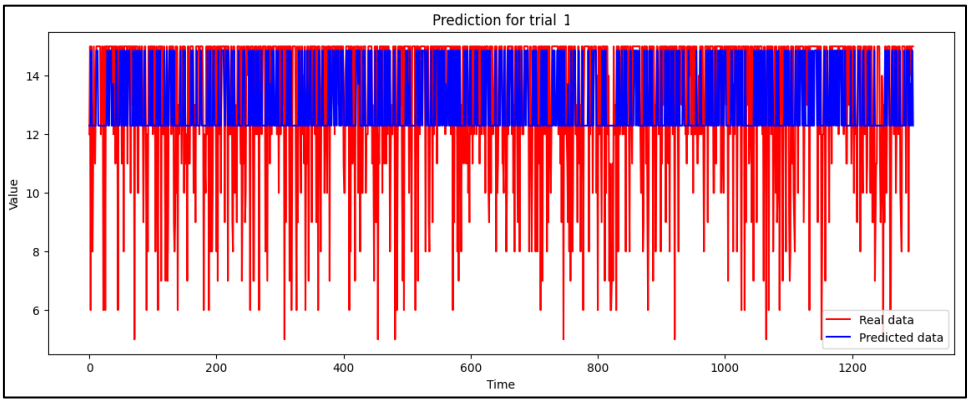

Figure 4.24: Actual and predicted data for trail 1 using Hyperband.

Iteration 2: This iteration had a significantly higher validation loss with a target value of -2.777, as shown in Figure 4.24. The LSTM and GRU activation functions were 'relu,' the optimizer used was 'rmsprop,' and the units for LSTM and GRU were 98.66 and 122.5, respectively.

Iteration 3: The target value was -1.211, corresponding to a validation loss 1.211. The activation functions for LSTM and GRU were 'tanh,' and the optimizer used was 'adam.' The LSTM units were set at 179.5, while the GRU units were set at 36.6.

4.6.2 Evaluating Hybrid LSTM-GRU Model

Following the comprehensive hyperparameter optimization processes using Hyperband and Bayesian Optimization, a final hybrid LSTM-GRU model was constructed and trained with the optimal parameters identified. Specifically, the model employed 'relu' as the activation function for both LSTM and GRU layers. The 'Adam' optimizer was used for the training process, and the model contained 89 units for the GRU layer and 81 units for the LSTM layer.

The model's performance was evaluated using several metrics: MSE, RMSE, MAE, and R2. These metrics comprehensively understand the model's prediction accuracy and error magnitude.

The results of the evaluation were promising, as shown in Figure 4.25. The model achieved an MSE of 0.0182, reflecting the average squared difference between the predicted and actual CQI values. It also obtained an RMSE of 0.1350, offering another perspective on the model's prediction error by placing more weight on more significant errors. The MAE was 0.1197, indicating the average absolute difference between predicted and actual CQI values. The R2 value was imposed at 0.9975, suggesting that the model's predictions could explain approximately 99.7% of the variance in the actual CQI values. This high R2 score underscores the model's excellent performance and ability to predict the CQI accurately.

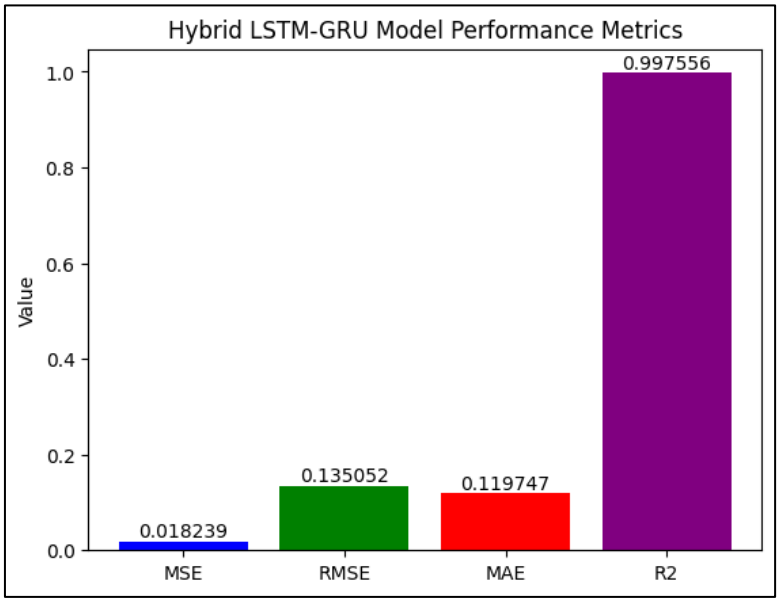

Figure 4.25: The Performance Metrics of the Hybrid LSTM-GRU Model.

During the training process, the model's performance improvement over time was tracked and visualized for training and validation Loss. This is depicted in Figure 4.26, showcasing the learning curve of the Hybrid LSTM-GRU model. The figure demonstrates the consistent reduction in training and validation loss across epochs, which indicates the model's learning progression and the convergence of the training process.

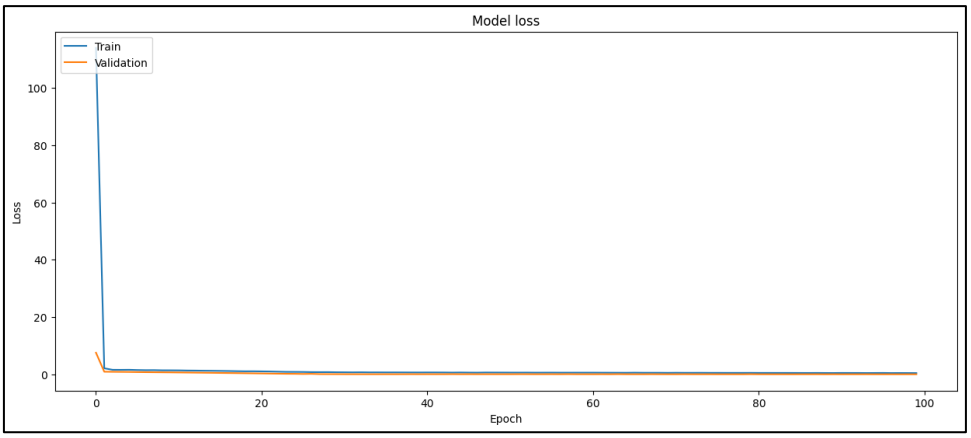

Figure 4.26: Training and validation loss using LSTM-GRU Model with Optimal Val.

Furthermore, the model's predictive capability is visually demonstrated in Figure 4.27; this scatter plot illustrates the close alignment between the actual and predicted CQI values, emphasizing the model's predictive solid performance. The close alignment between the two data sets is a visual testament to the model's ability to predict CQI based on the provided input features accurately.

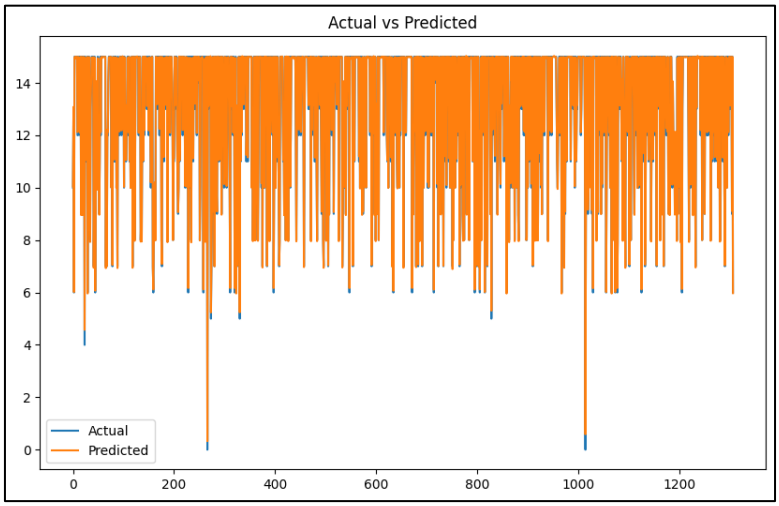

Figure 4.27: Actual and predicted data for Hybrid LSTM-GRU Model.

Starting with Hyperband optimization followed by Bayesian Optimization to refine the hyperparameters further, we determined the best parameters for the LSTM and GRU activation functions, optimizer, LSTM units, and GRU units.

The model's training was then carried out using these optimized parameters. Using both LSTM and GRU layers allowed the model to extract and remember both short-term and long-term dependencies in the data, which is crucial for accurate CQI prediction.

Overall, the optimization and training process led to the developing of a robust and accurate model capable of predicting CQI, a fundamental aspect of wireless communication systems. This model can potentially aid in improving the quality and reliability of these systems, enhancing user experience and network efficiency.

4.6.3 Comparative Analysis of Model Performance

Table 4.5 shows a comprehensive comparison of the performance of various AI models that were employed in the first stage for the prediction of CQI values. The models under consideration include ANN, DNN, SVM, LSTM, and our custom-designed Hybrid LSTM-GRU model, which has undergone rigorous hyperparameter tuning and optimization to enhance its predictive capabilities. The figure illustrates several key evaluation metrics, including MSE, RMSE, MAE, R2, MAPE, EV, and MPD, which provide valuable insights into the accuracy and reliability of the model's predictions. Of particular note is the Hybrid LSTM-GRU model, a product of our extensive efforts in hyperparameter tuning and optimization. This model stands out with remarkable performance, boasting a remarkably low MSE, RMSE, and MAE, while also achieving an exceptionally high R2, MAPE, MASE, EV, and MPD score. These results highlight the efficacy of our model in delivering highly accurate predictions of CQI values, a crucial metric in telecommunications.

In contrast, the conventional models, including ANN, DNN, SVM, and LSTM, display comparatively higher errors and lower R2, MAPE, MASE, EV, and MPD values, indicating a less accurate prediction of CQI values. Our Hybrid LSTM-GRU model's superior performance reinforces the benefits of customized models fine-tuned to the specific task.

TABLE 4.5: Comparative Analysis of Models Performance.

Model	MSE	RMSE	MAE	R^2	MAPE	MASE	EV	MPD
ANN	2.0544	1.4333	1.0630	0.7049	5.003	0.24	0.82	-1
DNN	0.9165	0.9573	0.7336	0.8683	4.403	0.20	0.86	0.202
SVM	1.1567	1.0708	0.6846	0.8353	3.012	0.1	-0.7	-3.01
LSTM	0.8115	0.9008	0.6085	0.8834	10.001	0.95	0.9	0.93
Hybrid LSTM-GRU	0.018	0.135	0.120	0.997	20.00	0.997	1.00	0.996

Figure 4.28 compares the Mean Squared Error (MSE) for different models, including ANN, DNN, SVM, LSTM, and the Hybrid LSTM-GRU model. MSE quantifies the average squared difference between predicted and actual values, with lower values indicating better predictive accuracy. Notably, the Hybrid LSTM-GRU modelexcels with the lowest MSE (0.0182), demonstrating its outstanding predictive accuracy when compared to other models.

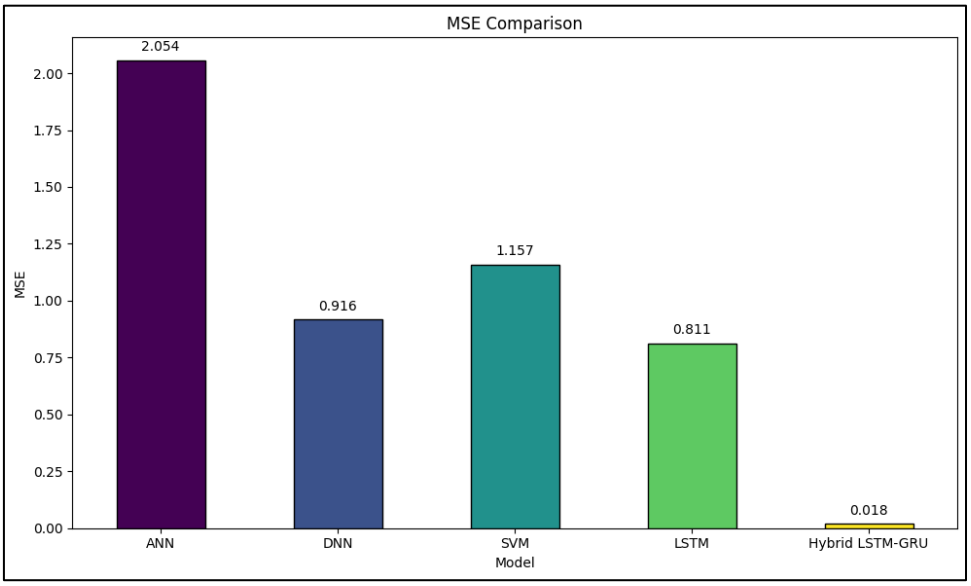

Figure 4.28: comparison of different AI models with our model using MSE.

Figure 4.29 showcases the Root Mean Squared Error (RMSE) for the same set of models. RMSE represents the standard deviation of prediction errors, with lower values indicating a more precise prediction. Once again, the Hybrid LSTM-GRU model shines with the lowest RMSE (0.1350), highlighting its superior predictive accuracy.

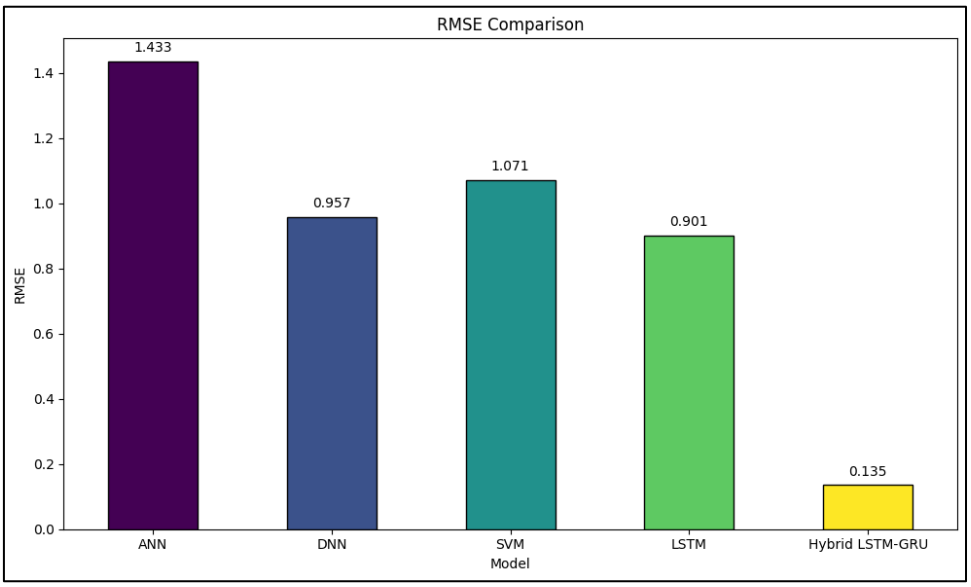

Figure 4.29: comparison of different AI models with our model using RMSE.

In Figure 4.30, we compare the Mean Absolute Error (MAE) among the models. MAE reflects the average absolute difference between predicted and actual values. The Hybrid LSTM-GRU model outperforms other models with the lowest MAE (0.1197), showcasing its exceptional predictive capability.

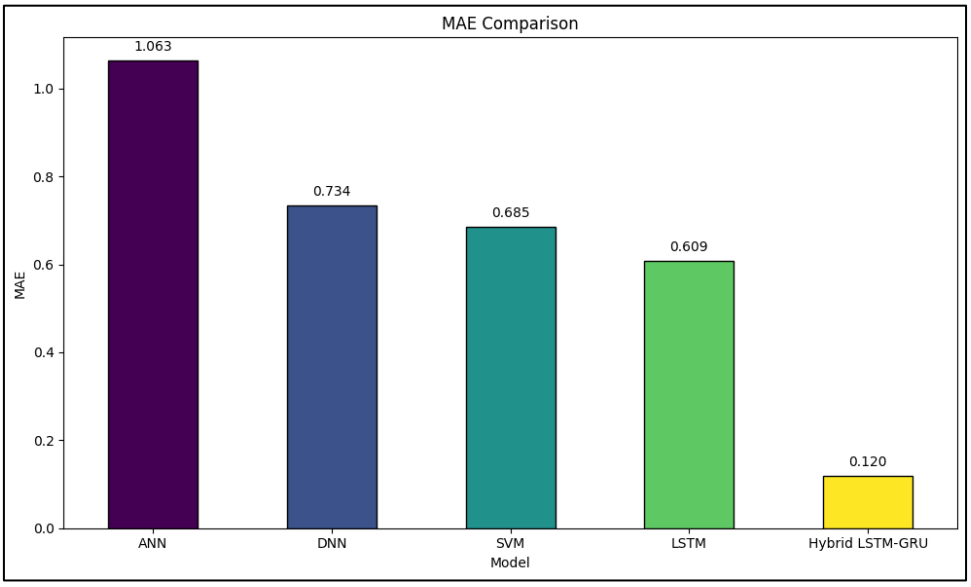

Figure 4.30: comparison of different AI models with our model using MAE.

Figure 4.31, presents the R-squared (R2) values for each model, where R2 measures the proportion of the variance in the dependent variable that can be explained by the independent variables. A higher R2 indicates a better model fit. The Hybrid LSTM-GRU model exhibits the highest R2 (0.997), signifying an excellent fit to the data.

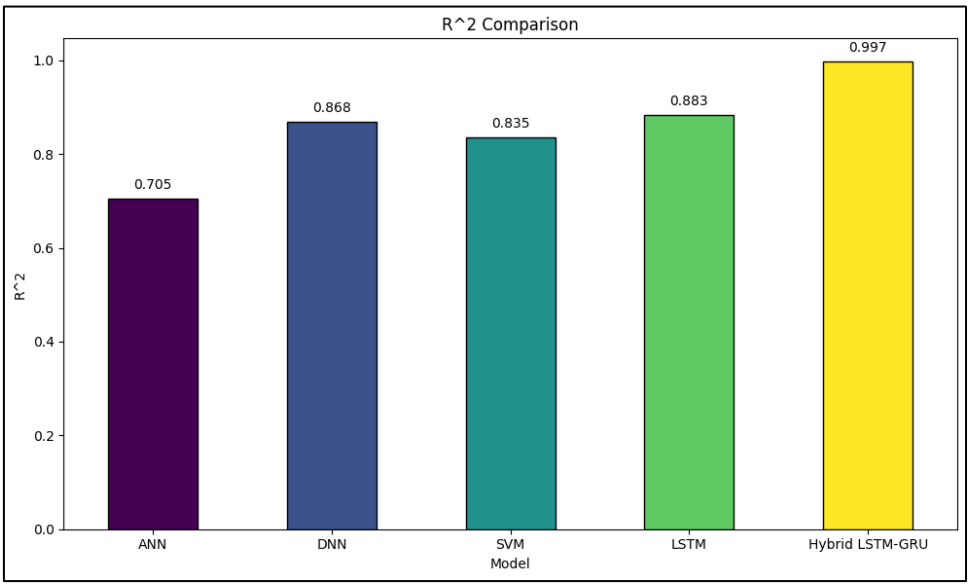

Figure 4.31: comparison of different AI models with our model using R2.

Figure 4.32, illustrates the Mean Absolute Percentage Error (MAPE) for each model, which measures prediction accuracy as a percentage error. While the Hybrid LSTM-GRU model has a higher MAPE of 20%, it is important to note that this is offset by its superior performance in other metrics.

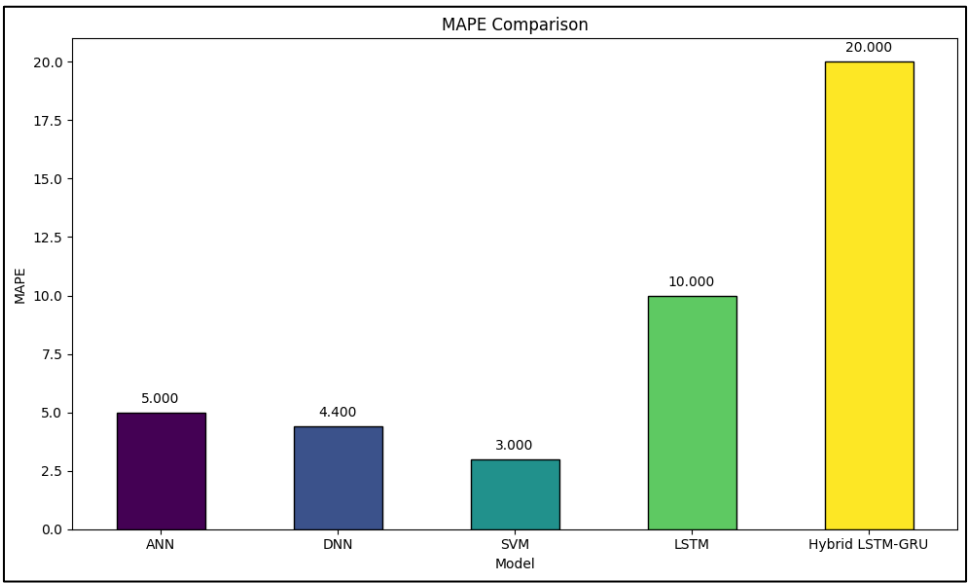

Figure 4.32: comparison of different AI models with our model using MAPE.

In Figure 4.33, we compare the Mean Absolute Scaled Error (MASE) for the models. MASE evaluates predictive accuracy relative to a naïve benchmark. The Hybrid LSTM-GRU model stands out with a MASE of 0.997, indicating strong predictive performance relative to the benchmark.

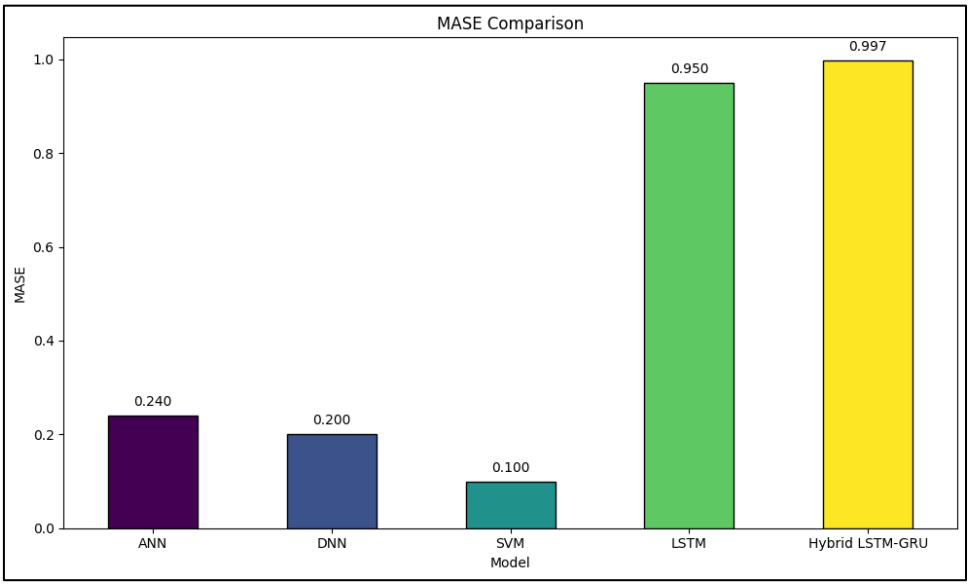

Figure 4.33: comparison of different AI models with our model using MASE.

Figure 4.34, shows the explained variance (EV) values for each model, representing the proportion of the total variance that the model explains. The Hybrid LSTM-GRU model excels with the highest EV (0.9998), indicating excellent explanatory power.

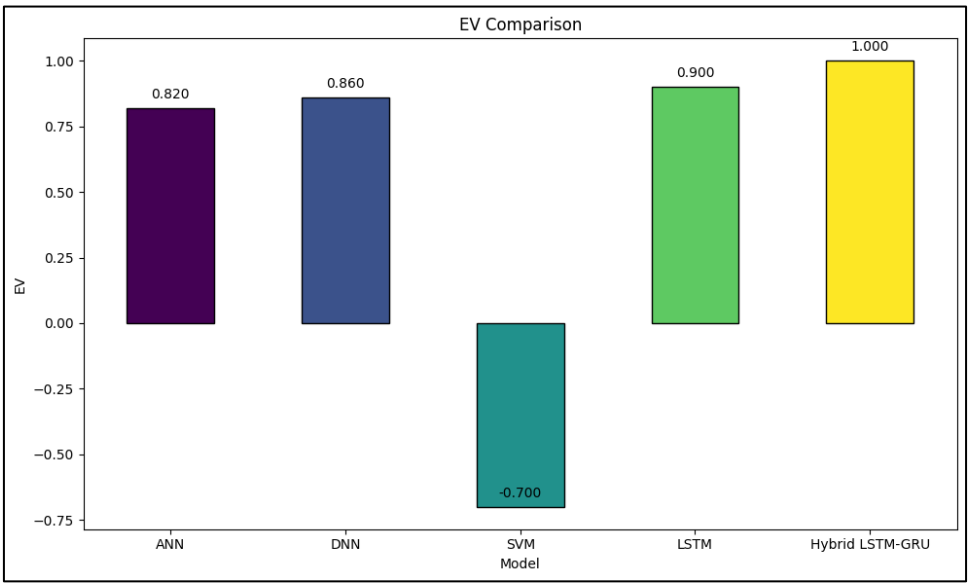

Figure 4.34: comparison of different AI models with our model using EV.

Figure 3.35, compares the Mean Percentage Difference (MPD) for the models, which measures the average percentage difference between predicted and actual values. The Hybrid LSTM-GRU model stands out with an MPD of 0.996, showcasing its proximity to actual values.

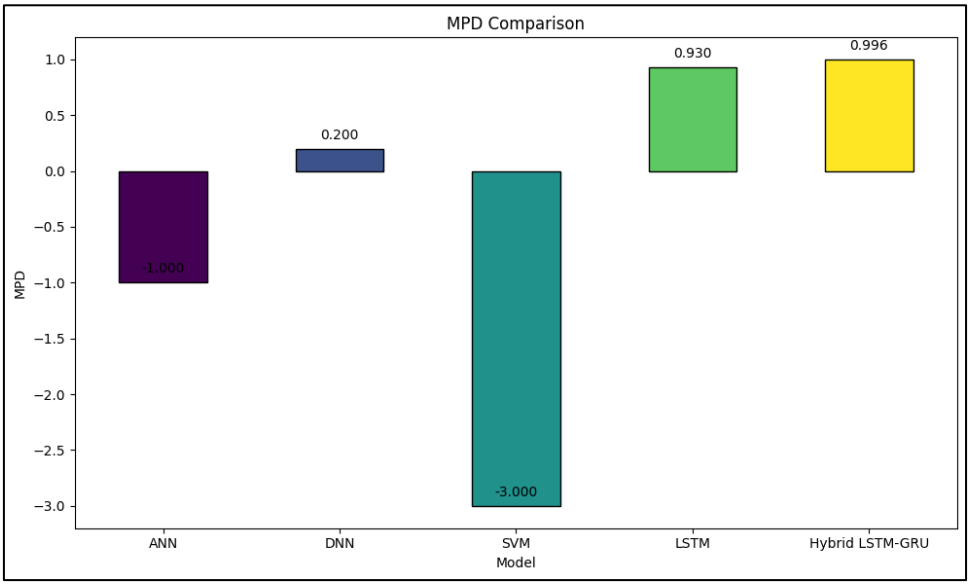

Figure 4.35: comparison of different AI models with our model using EV.

4.7 Summary

This chapter thoroughly examines and discusses the proposed system's performance when tested under varying parameters using a real-world 5G dataset.

This comprehensive evaluation covers every stage of the proposed system, including data processing, the initial model selection, the development of the optimized hybrid LSTM-GRU model, and the final evaluation of the model's effectiveness.

The results of these experiments validate the proposed system's efficiency and effectiveness in predicting Channel Quality Indicators (CQI) for Massive Radio Access Networks (RANs).

CHAPTER FIVE

CONCLUSION AND FUTURE WORKS

5.1 Overview

The journey in this dissertation was driven by the profound need for efficient and accurate prediction of the CQI beyond 5G Massive Radio Access Networks (RANs). The continuous growth in mobile data traffic and the ever-evolving landscape of applications have ushered in a new era of telecommunication networks that demand advanced solutions to tackle emerging challenges. As a result, exploring the potential of Artificial Intelligence (AI) and Machine Learning (ML) techniques in addressing these needs was at the core of this research.

The final chapter synthesizes the essential findings and insights drawn from the study and lays out a roadmap for future investigations in this exciting field. Reflecting on the research process, we consider the research question and objectives set out at the start of the journey and subsequently evaluate how effectively these have been met through the different stages of the research.

5.2 Conclusion

This dissertation presents a rigorous study on the implementation of AI techniques, specifically a hybrid model combining LSTM and GRU, to predict CQI and enhance the performance of beyond 5G Massive Radio Access Networks (rans).

- In the first stage of this research, an extensive comparative analysis was conducted using various AI models, including ANN, DNN, SVM,

and LSTM. The LSTM model emerged as the superior model for CQI prediction.

- The decision to combine LSTM and GRU into a hybrid model in the second stage of the research was based on the need to leverage their complementary strengths. While LSTM networks demonstrated strong performance in handling long-term dependencies in the data, the GRU's more straightforward structure and faster computation time added value to the hybrid model, making it a more efficient and effective system for CQI prediction.

- The hybrid LSTM-GRU model, when optimized using the Hyperband and Bayesian Optimization algorithms for hyperparameter tuning, showed a significant improvement in CQI prediction accuracy compared to the other ML models. This finding underlines the potential of integrating advanced AI techniques in managing and optimizing 5G Massive RANs.

5.3 Limitation

Data Limitations: The study used a specific dataset from a 5G network. While this dataset was sufficient for our analysis, it may not represent all types of beyond 5G RANs. The performance of the proposed models might vary under different network conditions or with different datasets.

Model Limitations: The research focused on a hybrid LSTM-GRU model for CQI prediction. However, other hybrid AI models and

techniques were not investigated in this study. The performance of these models might be different or even better in certain aspects.

Tuning Limitations: While Hyperband and Bayesian Optimization algorithms were used for hyperparameter tuning of the hybrid LSTM-GRU model, other hyperparameter tuning methods might provide different results. Additionally, the tuning was based on a specific set of hyperparameters, and it's possible that other hyperparameters could further improve the model's performance.

Computational Limitations: Implementing and testing the ML models, and notably the hybrid LSTM-GRU model, require significant computational resources. This could limit the practical applicability of the proposed system in resource-constrained scenarios.

Temporal Limitations: Technology and demands in wireless communication are rapidly evolving. Therefore, the proposed solutions might need to be adapted or updated to remain effective.

5.4 Future Work

While this dissertation focused on 5G Massive RANs, the proposed methods and findings could be applied to other types of networks and communication systems. Exploring these potential applications would broaden the scope and impact of the research.

Given the dynamic nature of wireless networks, it is crucial to continuously monitor and update the ML models to adapt to the changing network conditions and requirements. Future work could focus on

developing adaptive ML systems that can automatically update and optimize themselves based on real-time network data.

This dissertation significantly contributes to the field of telecommunications by demonstrating the potential of AI techniques, particularly a hybrid LSTM-GRU model, in predicting CQI and enhancing network performance. It provides a strong foundation for future research and development in AI applications beyond 5G Massive RANs and other telecommunications systems.

REFERENCES

References

[1] F. Salahdine, T. Han, and N. Zhang, “5G, 6G, and Beyond: Recent advances and future challenges,” *Annals of Telecommunications*, Jan. 2023, doi: 10.1007/s12243-022-00938-3.

[2] W. Saad, M. Bennis, and M. Chen, “A Vision of 6G Wireless Systems: Applications, Trends, Technologies, and Open Research Problems,” *IEEE Netw*, vol. 34, no. 3, pp. 134–142, May 2020, doi: 10.1109/MNET.001.1900287.

[3] Y. Yuan, Y. Zhao, B. Zong, and S. Parolari, “Potential key technologies for 6G mobile communications,” *Science China Information Sciences*, vol. 63, no. 8, p. 183301, Aug. 2020, doi: 10.1007/s11432-019-2789-y.

[4] W. Tong and P. Zhu, Eds., *6G: The Next Horizon*. Cambridge University Press, 2021. doi: 10.1017/9781108989817.

[5] A. Dogra, R. K. Jha, and S. Jain, “A Survey on Beyond 5G Network With the Advent of 6G: Architecture and Emerging Technologies,” *IEEE Access*, vol. 9, pp. 67512–67547, 2021, doi: 10.1109/ACCESS.2020.3031234.

[6] S. Rajoria and K. Mishra, “A brief survey on 6G communications,” *Wireless Networks*, vol. 28, no. 7, pp. 2901–2911, Oct. 2022, doi: 10.1007/s11276-022-03007-8.

[7] J. Zhang, E. Björnson, M. Matthaiou, D. W. K. Ng, H. Yang, and D. J. Love, “Prospective Multiple Antenna Technologies for Beyond 5G,” Sep. 2019.

[8] A. I. Salameh and M. El Tarhuni, “From 5G to 6G—Challenges, Technologies, and Applications,” *Future Internet*, vol. 14, no. 4, p. 117, Apr. 2022, doi: 10.3390/fi14040117.

[9] A. T. Z. Kasgari and W. Saad, “Model-Free Ultra Reliable Low Latency Communication (URLLC): A Deep Reinforcement Learning Framework,” in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, IEEE, May 2019, pp. 1–6. doi: 10.1109/ICC.2019.8761721.

[10] N. Ge, C. Jiang, and J. Lu, “A Systematic Perspective on Communication Innovations Toward 6G,” *Engineering*, May 2023, doi: 10.1016/j.eng.2023.03.011.

[11] B. Haryo Prananto, Iskandar, and A. Kurniawan, “O-RAN Intelligent Application for Cellular Mobility Management,” in *2022 International Conference on ICT for Smart Society (ICISS)*, IEEE, Aug. 2022, pp. 01–06. doi: 10.1109/ICISS55894.2022.9915221.

[12] A. Mourad, R. Yang, P. H. Lehne, and A. de la Oliva, “Towards 6G: Evolution of Key Performance Indicators and Technology Trends,” in *2020 2nd 6G Wireless Summit (6G SUMMIT)*, IEEE, Mar. 2020, pp. 1–5. doi: 10.1109/6GSUMMIT49458.2020.9083759.

[13] V. Kumar and N. B. Mehta, “Exploiting Correlation Between Wideband and Differential CQIs for Adaptation and Feedback,” in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, IEEE, Dec. 2020, pp. 1–6. doi: 10.1109/GLOBECOM42002.2020.9348155.

[14] J. Kim and D. S. Han, “Deep learning-based channel quality indicators prediction for vehicular communication,” *ICT Express*, May 2022, doi: 10.1016/j.icte.2022.05.002.

[15] A. Balieiro, K. Dias, and P. Guarda, “A Machine Learning Approach for CQI Feedback Delay in 5G and Beyond 5G Networks,” in *2021 30th Wireless and Optical Communications Conference (WOCC)*, IEEE, Oct. 2021, pp. 26–30. doi: 10.1109/WOCC53213.2021.9603019.

[16] H. Yin, X. Guo, P. Liu, X. Hei, and Y. Gao, “Predicting Channel Quality Indicators for 5G Downlink Scheduling in a Deep Learning Approach,” Aug. 2020.

[17] G. Pocovi, A. A. Esswie, and K. I. Pedersen, “Channel Quality Feedback Enhancements for Accurate URLLC Link Adaptation in 5G Systems,” in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, IEEE, May 2020, pp. 1–6. doi: 10.1109/VTC2020-Spring48590.2020.9128909.

[18] S. K. Vankayala and K. G. Shenoy, “A Neural Network for Estimating CQI in 5G Communication Systems,” in *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, IEEE, Apr. 2020, pp. 1–5. doi: 10.1109/WCNCW48565.2020.9124744.

[19] K. Park, J. Rhee, M.-H. Shin, and W. Hur, “A Novel CQI Feedback Channel for Cellular UAV System,” in *2019 International Conference on Advanced Technologies for Communications (ATC)*, IEEE, Oct. 2019, pp. 84–88. doi: 10.1109/ATC.2019.8924562.

[20] S. Homayouni, S. Schwarz, M. K. Mueller, and M. Rupp, “CQI Mapping Optimization in Spatial Wireless Channel Prediction,” in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, IEEE, Jun. 2018, pp. 1–5. doi: 10.1109/VTCSpring.2018.8417470.

[21] S. Homayouni, S. Schwarz, M. K. Mueller, and M. Rupp, “Reducing CQI Feedback Overhead by Exploiting Spatial Correlation,” in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, IEEE, Jun. 2018, pp. 1–5. doi: 10.1109/VTCSpring.2018.8417549.

[22] A. Chiumento, M. Bennis, C. Desset, L. Van der Perre, and S. Pollin, “Adaptive CSI and feedback estimation in LTE and beyond: a Gaussian process regression approach,” *EURASIP J Wirel Commun Netw*, vol. 2015, no. 1, p. 168, Dec. 2015, doi: 10.1186/s13638-015-0388-0.

[23] J. Wang, J. Xu, and X. Wang, “Combination of Hyperband and Bayesian Optimization for Hyperparameter Optimization in Deep Learning,” *Computer Vision and Pattern Recognition*, Jan. 2018.

[24] A. Bhardwaj, V. Mangat, and R. Vig, “Hyperband Tuned Deep Neural Network With Well Posed Stacked Sparse AutoEncoder for Detection of DDoS Attacks in Cloud,” *IEEE Access*, vol. 8, pp. 181916–181929, 2020, doi: 10.1109/ACCESS.2020.3028690.

[25] R. A. Andhika Viadinugroho and D. Rosadi, “A weighted metric scalarization approach for multiobjective BOHB hyperparameter optimization in LSTM model for sentiment analysis,” *Inf Sci (N Y)*, vol. 644, p. 119282, Oct. 2023, doi: 10.1016/j.ins.2023.119282.

[26] M. Majid *et al.*, “Applications of Wireless Sensor Networks and Internet of Things Frameworks in the Industry Revolution 4.0: A Systematic Literature Review,” *Sensors*, vol. 22, no. 6, p. 2087, Mar. 2022, doi: 10.3390/s22062087.

[27] I. F. Akyildiz, A. Kak, and S. Nie, “6G and Beyond: The Future of Wireless Communications Systems,” *IEEE Access*, vol. 8, pp. 133995–134030, 2020, doi: 10.1109/ACCESS.2020.3010896.

[28] A. L. Imoize, F. Udeji, J. Isabona, and C.-C. Lee, “Optimizing the Quality of Service of Mobile Broadband Networks for a Dense Urban Environment,” *Future Internet*, vol. 15, no. 5, p. 181, May 2023, doi: 10.3390/fi15050181.

[29] H. Asplund *et al.*, “3GPP Physical Layer Solutions for LTE and the Evolution Toward NR,” in *Advanced Antenna Systems for 5G Network Deployments*, Elsevier, 2020, pp. 301–350. doi: 10.1016/B978-0-12-820046-9.00008-3.

[30] N. Han, I.-M. Kim, and J. So, “Lightweight LSTM-Based Adaptive CQI Feedback Scheme for IoT Devices,” *Sensors*, vol. 23, no. 10, p. 4929, May 2023, doi: 10.3390/s23104929.

[31] M. A. Kamal, H. W. Raza, M. M. Alam, M. M. Su’ud, and A. binti A. B. Sajak, “Resource Allocation Schemes for 5G Network: A Systematic Review,” *Sensors*, vol. 21, no. 19, p. 6588, Oct. 2021, doi: 10.3390/s21196588.

[32] X. Chen, H. Yi, H. Luo, H. Yu, and H. Wang, “A novel CQI calculation scheme in LTE\LTE-A systems,” in *2011 International Conference on Wireless Communications and Signal Processing (WCSP)*, IEEE, Nov. 2011, pp. 1–5. doi: 10.1109/WCSP.2011.6096767.

[33] S. M. Abd El-atty, D. N. Skoutas, and A. N. Rouskas, “Reducing CQI Signalling Overhead in HSPA,” *Research Letters in Communications*, vol. 2008, pp. 1–5, 2008, doi: 10.1155/2008/982805.

[34] Z. Du, S. Wang, H. Li, X. You, and Y. Shi, “Control plan design,” in *5G NR and Enhancements*, Elsevier, 2022, pp. 579–620. doi: 10.1016/B978-0-323-91060-6.00011-8.

[35] Y.-T. Mai and C.-T. Yang, “A Hybrid Network User Satisfaction-Based Downlink Scheduling in LTE-A Network,” *Math Probl Eng*, vol. 2022, pp. 1–17, May 2022, doi: 10.1155/2022/6342802.

[36] M. P. Mota, D. C. Araujo, F. H. Costa Neto, A. L. F. de Almeida, and F. R. Cavalcanti, “Adaptive Modulation and Coding Based on Reinforcement Learning for 5G Networks,” in *2019 IEEE Globecom Workshops (GC Wkshps)*, IEEE, Dec. 2019, pp. 1–6. doi: 10.1109/GCWkshps45667.2019.9024384.

[37] F. Afroz, R. Subramanian, R. Heidary, K. Sandrasegaran, and S. Ahmed, “SINR, RSRP, RSSI and RSRQ Measurements in Long Term Evolution Networks,” *International Journal of Wireless & Mobile Networks*, vol. 7, no. 4, pp. 113–123, Aug. 2015, doi: 10.5121/ijwmn.2015.7409.

[38] R. Musabe, V. M. Ushindi, A. Mugisha, M. Emmanuel, G. Eugene, and G. Bajpai, “Toward Mobility Parameter Planning for 3GPP LTE in Rwanda,” in *Proceedings of the 3rd International Conference on Smart City Applications*, New York, NY, USA: ACM, Oct. 2018, pp. 1–6. doi: 10.1145/3286606.3286790.

[39] I. Petrut, M. Otesteanu, C. Balint, and G. Budura, “HetNet handover performance analysis based on RSRP vs. RSRQ triggers,” in *2015 38th International Conference on Telecommunications and Signal Processing (TSP)*, IEEE, Jul. 2015, pp. 232–235. doi: 10.1109/TSP.2015.7296258.

[40] V. Raida, M. Lerch, P. Svoboda, and M. Rupp, “Deriving Cell Load from RSRQ Measurements,” in *2018 Network Traffic Measurement and Analysis Conference (TMA)*, IEEE, Jun. 2018, pp. 1–6. doi: 10.23919/TMA.2018.8506494.

[41] G. M. Putra, E. Budiman, Y. Malewa, D. Cahyadi, M. Taruk, and U. Hairah, “4G LTE Experience: Reference Signal Received Power, Noise Ratio and Quality,” in *2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT)*, IEEE, Apr. 2021, pp. 139–144. doi: 10.1109/EIConCIT50028.2021.9431853.

[42] T. Kim, K. Ko, I. Hwang, D. Hong, S. Choi, and H. Wang, “RSRP-Based Doppler Shift Estimator Using Machine Learning in High-Speed Train Systems,” *IEEE Trans Veh Technol*, vol. 70, no. 1, pp. 371–380, Jan. 2021, doi: 10.1109/TVT.2020.3044175.

[43] T. Ngenjaroendee, W. Phakphisut, T. Wijitpornchai, P. Areeprayoonkij, and T. Jaruvitayakovit, "Deep Learning-based Reference Signal Received Power Prediction for LTE Communication System," in *2022 37th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*, IEEE, Jul. 2022, pp. 888–891. doi: 10.1109/ITC-CSCC55581.2022.9895098.

[44] P. Safayenikoo and I. Akturk, "Weight Update Skipping: Reducing Training Time for Artificial Neural Networks," *IEEE J Emerg Sel Top Circuits Syst*, vol. 11, no. 4, pp. 563–574, Dec. 2021, doi: 10.1109/JETCAS.2021.3127907.

[45] K. M. Kahloot and P. Ekler, "Algorithmic Splitting: A Method for Dataset Preparation," *IEEE Access*, vol. 9, pp. 125229–125237, 2021, doi: 10.1109/ACCESS.2021.3110745.

[46] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning (still) requires rethinking generalization," *Commun ACM*, vol. 64, no. 3, pp. 107–115, Mar. 2021, doi: 10.1145/3446776.

[47] M. A. S. Sejan, M. H. Rahman, B.-S. Shin, J.-H. Oh, Y.-H. You, and H.-K. Song, "Machine Learning for Intelligent-Reflecting-Surface-Based Wireless Communication towards 6G: A Review," *Sensors*, vol. 22, no. 14, p. 5405, Jul. 2022, doi: 10.3390/s22145405.

[48] A. Kollár, "Betting models using AI: A review on ANN, SVM, and Markov Chain," 2021.

[49] J. Zou, Y. Han, and S.-S. So, "Overview of Artificial Neural Networks," 2008, pp. 14–22. doi: 10.1007/978-1-60327-101-1_2.

[50] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, Nov. 2018, doi: 10.1016/j.heliyon.2018.e00938.

[51] Hecht-Nielsen, "Theory of the backpropagation neural network," in *International Joint Conference on Neural Networks*, IEEE, 1989, pp. 593–605 vol.1. doi: 10.1109/IJCNN.1989.118638.

[52] K.-C. Chen and Y.-S. Liao, "A Reconfigurable Deep Neural Network on Chip Design with Flexible Convolutional Operations," in *2022 15th IEEE/ACM International Workshop on Network on Chip Architectures (NoCArc)*, IEEE, Oct. 2022, pp. 1–5. doi: 10.1109/NoCArc57472.2022.9911283.

[53] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Artificial Neural Networks-Based Machine Learning for Wireless Networks: A Tutorial," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3039–3071, 2019, doi: 10.1109/COMST.2019.2926625.

[54] Y. Zhang, Y. Xie, Y. Zhang, J. Qiu, and S. Wu, “The adoption of deep neural network (DNN) to the prediction of soil liquefaction based on shear wave velocity,” *Bulletin of Engineering Geology and the Environment*, vol. 80, no. 6, pp. 5053–5060, Jun. 2021, doi: 10.1007/s10064-021-02250-1.

[55] M. Abedi, G.-H. Norouzi, and A. Bahroudi, “Support vector machine for multi-classification of mineral prospectivity areas,” *Comput Geosci*, vol. 46, pp. 272–283, Sep. 2012, doi: 10.1016/j.cageo.2011.12.014.

[56] V. Kadam, S. Kumar, A. Bongale, S. Wazarkar, P. Kamat, and S. Patil, “Enhancing Surface Fault Detection Using Machine Learning for 3D Printed Products,” *Applied System Innovation*, vol. 4, no. 2, p. 34, May 2021, doi: 10.3390/asi4020034.

[57] R. Ravinder, Y. Ramadevi, and K. V. N. Sunitha, “Anomaly Detection using Feature Selection and SVM Kernel Trick,” *Int J Comput Appl*, vol. 129, no. 4, pp. 31–35, Nov. 2015, doi: 10.5120/ijca2015906823.

[58] M. N. Murty and R. Raghava, “Kernel-Based SVM,” 2016, pp. 57–67. doi: 10.1007/978-3-319-41063-0_5.

[59] H. Zan, M. Wang, and M. Sun, “Aerial Handwriting Recognition Based on SVM,” in *2018 International Symposium on Computer, Consumer and Control (IS3C)*, IEEE, Dec. 2018, pp. 250–253. doi: 10.1109/IS3C.2018.00070.

[60] M. Yang, S. Cui, Y. Zhang, J. Zhang, and X. Li, “Data and Image Classification of Haematococcus pluvialis Based on SVM Algorithm,” in *2021 China Automation Congress (CAC)*, IEEE, Oct. 2021, pp. 522–525. doi: 10.1109/CAC53003.2021.9727433.

[61] R. N. Satrya, O. N. Pratiwi, R. Y. Farifah, and J. Abawajy, “Cryptocurrency Sentiment Analysis on the Twitter Platform Using Support Vector Machine (SVM) Algorithm,” in *2022 International Conference Advancement in Data Science, E-learning and Information Systems (ICADEIS)*, IEEE, Nov. 2022, pp. 01–05. doi: 10.1109/ICADEIS56544.2022.10037413.

[62] P. Alekhya, P. M. Reddy, M. Chiranjeevi, and V. S. Dhuli, “An Efficient Brain Tumor Classification using CNN and SVM Models,” in *2023 IEEE 12th International Conference on Communication Systems and Network Technologies (CSNT)*, IEEE, Apr. 2023, pp. 371–376. doi: 10.1109/CSNT57126.2023.10134705.

[63] L. Breiman, “Random Forests,” *Mach Learn*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.

[64] T.-N. Do, P. Lenca, S. Lallich, and N.-K. Pham, “Classifying Very-High-Dimensional Data with Random Forests of Oblique Decision Trees,” 2010, pp. 39–55. doi: 10.1007/978-3-642-00580-0_3.

[65] J. M. Rudd and H. “Gene” Ray, “An Empirical Study of Downstream Analysis Effects of Model Pre-Processing Choices,” *Open J Stat*, vol. 10, no. 05, pp. 735–809, 2020, doi: 10.4236/ojs.2020.105046.

[66] P. Patil, N. Yaligar, and S. M. Meena, “Comparision of Performance of Classifiers - SVM, RF and ANN in Potato Blight Disease Detection Using Leaf Images,” in *2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, IEEE, Dec. 2017, pp. 1–5. doi: 10.1109/ICCIC.2017.8524301.

[67] T. Mehta, P. Devi K K, S. Kumar M, G. Sharma, A. Thatikonda, and L. Kipkirui, “Analyzing Portfolio of Biotech Companies and Predicting Stock Market using Machine Learning,” in *2022 2nd International Conference on Intelligent Technologies (CONIT)*, IEEE, Jun. 2022, pp. 1–7. doi: 10.1109/CONIT55038.2022.9847963.

[68] T. Mohiuddin, S. Naznin, and P. B. Upama, “Classification and Performance Analysis of Cancer Microarrays Using Relevant Genes,” in *2021 5th International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*, IEEE, Nov. 2021, pp. 1–6. doi: 10.1109/ICEEICT53905.2021.9667822.

[69] S. Susan Jacob, “An Efficient Scheme for Facial Expression Recognition using Random Forest and CNN,” in *2022 Third International Conference on Intelligent Computing Instrumentation and Control Technologies (ICICICT)*, IEEE, Aug. 2022, pp. 905–910. doi: 10.1109/ICICICT54557.2022.9917782.

[70] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[71] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, “LSTM: A Search Space Odyssey,” *IEEE Trans Neural Netw Learn Syst*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017, doi: 10.1109/TNNLS.2016.2582924.

[72] D. Castro, B. L. D. Bezerra, and M. Valenca, “Boosting the Deep Multidimensional Long-Short-Term Memory Network for Handwritten Recognition Systems,” in *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, IEEE, Aug. 2018, pp. 127–132. doi: 10.1109/ICFHR-2018.2018.00031.

[73] A. Graves, “Generating Sequences With Recurrent Neural Networks,” Aug. 2013.

[74] Y. Fan, Y. Qian, F.-L. Xie, and F. K. Soong, “TTS synthesis with bidirectional LSTM based recurrent neural networks,” in *Interspeech 2014*, ISCA: ISCA, Sep. 2014, pp. 1964–1968. doi: 10.21437/Interspeech.2014-443.

[75] E. Marchi, G. Ferroni, F. Eyben, L. Gabrielli, S. Squartini, and B. Schuller, "Multi-resolution linear prediction based features for audio onset detection with bidirectional LSTM neural networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, May 2014, pp. 2164–2168. doi: 10.1109/ICASSP.2014.6853982.

[76] J. Donahue *et al.*, "Long-term Recurrent Convolutional Networks for Visual Recognition and Description," Nov. 2014. doi: 10.21236/ADA623249.

[77] L. Ren, J. Dong, X. Wang, Z. Meng, L. Zhao, and M. J. Deen, "A Data-Driven Auto-CNN-LSTM Prediction Model for Lithium-Ion Battery Remaining Useful Life," *IEEE Trans Industr Inform*, vol. 17, no. 5, pp. 3478–3487, May 2021, doi: 10.1109/TII.2020.3008223.

[78] Z. Wu, J. Gao, Q. Li, Z. Guan, and Z. Chen, "Make aspect-based sentiment classification go further: step into the long-document-level," *Applied Intelligence*, vol. 52, no. 8, pp. 8428–8447, Jun. 2022, doi: 10.1007/s10489-021-02836-y.

[79] K. Smagulova and A. P. James, "Overview of Long Short-Term Memory Neural Networks," 2020, pp. 139–153. doi: 10.1007/978-3-030-14524-8_11.

[80] Z. Kong, Y. Cui, Z. Xia, and H. Lv, "Convolution and Long Short-Term Memory Hybrid Deep Neural Networks for Remaining Useful Life Prognostics," *Applied Sciences*, vol. 9, no. 19, p. 4156, Oct. 2019, doi: 10.3390/app9194156.

[81] S. Oh, K. Jang, J. Kim, and I. Moon, "Online state of charge estimation of lithium-ion battery using surrogate model based on electrochemical model," 2022, pp. 1447–1452. doi: 10.1016/B978-0-323-95879-0.50242-3.

[82] Y. Guo, X. Cao, B. Liu, and K. Peng, "El Niño Index Prediction Using Deep Learning with Ensemble Empirical Mode Decomposition," *Symmetry (Basel)*, vol. 12, no. 6, p. 893, Jun. 2020, doi: 10.3390/sym12060893.

[83] R. DiPietro and G. D. Hager, "Deep learning: RNNs and LSTM," in *Handbook of Medical Image Computing and Computer Assisted Intervention*, Elsevier, 2020, pp. 503–519. doi: 10.1016/B978-0-12-816176-0.00026-0.

[84] I. Pisa, I. Santin, A. Morell, J. L. Vicario, and R. Vilanova, "LSTM-Based Wastewater Treatment Plants Operation Strategies for Effluent Quality Improvement," *IEEE Access*, vol. 7, pp. 159773–159786, 2019, doi: 10.1109/ACCESS.2019.2950852.

[85] J. M. Navarro, R. Martínez-España, A. Bueno-Crespo, R. Martínez, and J. M. Cecilia, "Sound Levels Forecasting in an Acoustic Sensor Network Using a Deep Neural Network," *Sensors*, vol. 20, no. 3, p. 903, Feb. 2020, doi: 10.3390/s20030903.

[86] L. Feng, “Predicting Output Responses of Nonlinear Dynamical Systems With Parametrized Inputs Using LSTM,” *IEEE J Multiscale Multiphys Comput Tech*, vol. 8, pp. 97–107, 2023, doi: 10.1109/JMMCT.2023.3242044.

[87] K. A. Koparanov, K. K. Georgiev, and V. A. Shterev, “Lookback Period, Epochs and Hidden States Effect on Time Series Prediction Using a LSTM based Neural Network,” in *2020 28th National Conference with International Participation (TELECOM)*, IEEE, Oct. 2020, pp. 61–64. doi: 10.1109/TELECOM50385.2020.9299551.

[88] T. Gao, Y. Chai, and Y. Liu, “Applying long short term momory neural networks for predicting stock closing price,” in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, IEEE, Nov. 2017, pp. 575–578. doi: 10.1109/ICSESS.2017.8342981.

[89] K. Cho *et al.*, “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation,” Jun. 2014.

[90] K. Zarzycki and M. Ławryńczuk, “LSTM and GRU Neural Networks as Models of Dynamical Processes Used in Predictive Control: A Comparison of Models Developed for Two Chemical Reactors,” *Sensors*, vol. 21, no. 16, p. 5625, Aug. 2021, doi: 10.3390/s21165625.

[91] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,” Dec. 2014.

[92] R. Cahuantzi, X. Chen, and S. Güttel, “A comparison of LSTM and GRU networks for learning symbolic sequences,” 2021. doi: 10.1007/978-3-031-37963-5_53.

[93] W.-C. Chuang, W.-J. Hwang, T.-M. Tai, D.-R. Huang, and Y.-J. Jhang, “Continuous Finger Gesture Recognition Based on Flex Sensors,” *Sensors*, vol. 19, no. 18, p. 3986, Sep. 2019, doi: 10.3390/s19183986.

[94] P. Kumar and S. Suresh, “DeepTransHAR: a novel clustering-based transfer learning approach for recognizing the cross-domain human activities using GRUs (Gated Recurrent Units) Networks,” *Internet of Things*, vol. 21, p. 100681, Apr. 2023, doi: 10.1016/j.iot.2023.100681.

[95] R. Wang, M. Panju, and M. Gohari, “Classification-based RNN machine translation using GRUs,” Mar. 2017.

[96] M. Zulqarnain, S. Abd, R. Ghazali, N. Mohd, M. Aamir, and Y. Mazwin, “An Improved Deep Learning Approach based on Variant Two-State Gated Recurrent Unit and Word Embeddings for Sentiment Classification,” *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 1, 2020, doi: 10.14569/IJACSA.2020.0110174.

[97] A. Dutta, S. Kumar, and M. Basu, “A Gated Recurrent Unit Approach to Bitcoin Price Prediction,” *Journal of Risk and Financial Management*, vol. 13, no. 2, p. 23, Feb. 2020, doi: 10.3390/jrfm13020023.

[98] A. Gu, C. Gulcehre, T. Le Paine, M. Hoffman, and R. Pascanu, “Improving the Gating Mechanism of Recurrent Neural Networks,” Oct. 2019.

[99] B. Athiwaratkun and J. W. Stokes, “Malware classification with LSTM and GRU language models and a character-level CNN,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, Mar. 2017, pp. 2482–2486. doi: 10.1109/ICASSP.2017.7952603.

[100] J. Singh, S. Sharma, and B. J, “Natural Language Processing based Machine Translation for Hindi-English using GRU and Attention,” in *2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, IEEE, May 2022, pp. 965–969. doi: 10.1109/ICAAIC53929.2022.9793214.

[101] A. Mahmud, Md. M. Al Barat, and S. Kamruzzaman, “GRU-based Encoder-Decoder Attention Model for English to Bangla Translation on Novel Dataset,” in *2021 5th International Conference on Electrical Information and Communication Technology (EICT)*, IEEE, Dec. 2021, pp. 1–6. doi: 10.1109/EICT54103.2021.9733595.

[102] W. Akbar, M. Inam Ul Haq, A. Soomro, S. Muhammad Daudpota, A. Shariq Imran, and M. Ullah, “Automated Report Generation: A GRU Based Method for Chest X-Rays,” in *2023 4th International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, IEEE, Mar. 2023, pp. 1–6. doi: 10.1109/iCoMET57998.2023.10099311.

[103] M. Liu, Y. Wang, Z. Yan, J. Wang, and X. Xie, “Robust Speech Recognition based on Multi-Objective Learning with GRU Network,” in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, IEEE, Nov. 2019, pp. 181–185. doi: 10.1109/APSIPAASC47483.2019.9023325.

[104] U. Rawat and S. Singh, “Automatic Music Generation: Comparing LSTM and GRU,” in *2022 3rd International Conference on Intelligent Engineering and Management (ICIEM)*, IEEE, Apr. 2022, pp. 693–698. doi: 10.1109/ICIEM54221.2022.9853112.

[105] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, “Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark,” Sep. 2021.

[106] A. Farzad, H. Mashayekhi, and H. Hassanpour, “A comparative performance analysis of different activation functions in LSTM networks for classification,” *Neural Comput Appl*, vol. 31, no. 7, pp. 2507–2521, Jul. 2019, doi: 10.1007/s00521-017-3210-6.

[107] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, “Activation functions in deep learning: A comprehensive survey and benchmark,” *Neurocomputing*, vol. 503, pp. 92–108, Sep. 2022, doi: 10.1016/j.neucom.2022.06.111.

[108] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, “Activation functions in deep learning: A comprehensive survey and benchmark,” *Neurocomputing*, vol. 503, pp. 92–108, Sep. 2022, doi: 10.1016/j.neucom.2022.06.111.

[109] S. Narayan, “The generalized sigmoid activation function: Competitive supervised learning,” *Inf Sci (N Y)*, vol. 99, no. 1–2, pp. 69–82, Jun. 1997, doi: 10.1016/S0020-0255(96)00200-9.

[110] J. Feng, X. He, Q. Teng, C. Ren, H. Chen, and Y. Li, “Reconstruction of porous media from extremely limited information using conditional generative adversarial networks,” *Phys Rev E*, vol. 100, no. 3, p. 033308, Sep. 2019, doi: 10.1103/PhysRevE.100.033308.

[111] P. Ranjan, P. Khan, S. Kumar, and S. K. Das, “log-Sigmoid Activation-Based Long Short-Term Memory for Time Series Data Classification,” *IEEE Transactions on Artificial Intelligence*, pp. 1–12, 2023, doi: 10.1109/TAI.2023.3265641.

[112] V. K. and S. K., “Towards activation function search for long short-term model network: A differential evolution based approach,” *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 6, pp. 2637–2650, Jun. 2022, doi: 10.1016/j.jksuci.2020.04.015.

[113] B. Ding, H. Qian, and J. Zhou, “Activation functions and their characteristics in deep neural networks,” in *2018 Chinese Control And Decision Conference (CCDC)*, IEEE, Jun. 2018, pp. 1836–1841. doi: 10.1109/CCDC.2018.8407425.

[114] D. Stursa and P. Dolezel, “Comparison of ReLU and linear saturated activation functions in neural network for universal approximation,” in *2019 22nd International Conference on Process Control (PC19)*, IEEE, Jun. 2019, pp. 146–151. doi: 10.1109/PC.2019.8815057.

[115] S. Nasiri, J. Helsper, M. Jung, and M. Fathi, “DePicT Melanoma Deep-CLASS: a deep convolutional neural networks approach to classify skin lesion images,” *BMC Bioinformatics*, vol. 21, no. S2, p. 84, Mar. 2020, doi: 10.1186/s12859-020-3351-y.

[116] A. Kagalkar and S. Raghuram, “CORDIC Based Implementation of the Softmax Activation Function,” in *2020 24th International Symposium on VLSI Design and Test (VDAT)*, IEEE, Jul. 2020, pp. 1–4. doi: 10.1109/VDAT50263.2020.9190498.

[117] F. Es-Sabery, A. Hair, J. Qadir, B. Sainz-De-Abajo, B. Garcia-Zapirain, and I. Torre-Diez, “Sentence-Level Classification Using Parallel Fuzzy Deep Learning Classifier,” *IEEE Access*, vol. 9, pp. 17943–17985, 2021, doi: 10.1109/ACCESS.2021.3053917.

[118] A. Pandey, P. Pinky, and S. Kumar, “Localization Using Stochastic Gradient Descent Method in a 5G Network,” in *2018 15th IEEE India Council International Conference (INDICON)*, IEEE, Dec. 2018, pp. 1–6. doi: 10.1109/INDICON45594.2018.8987064.

[119] S. Ruder, “An overview of gradient descent optimization algorithms,” Sep. 2016.

[120] B. Ghosh, I. K. Dutta, A. Carlson, M. Totaro, and M. Bayoumi, “An Empirical Analysis of Generative Adversarial Network Training Times with Varying Batch Sizes,” in *2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, IEEE, Oct. 2020, pp. 0643–0648. doi: 10.1109/UEMCON51285.2020.9298092.

[121] T. Tieleman and G. Hinton, “RMSProp adaptive learning,” *Coursera: Neural networks for machine learning*, 2012.

[122] R. Zaheer and H. Shaziya, “A Study of the Optimization Algorithms in Deep Learning,” in *2019 Third International Conference on Inventive Systems and Control (ICISC)*, IEEE, Jan. 2019, pp. 536–539. doi: 10.1109/ICISC44355.2019.9036442.

[123] X. Wu, R. Ward, and L. Bottou, “WNGrad: Learn the Learning Rate in Gradient Descent,” Mar. 2018.

[124] M. C. Mukkamala and M. Hein, “Variants of RMSProp and Adagrad with Logarithmic Regret Bounds,” Jun. 2017.

[125] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” Dec. 2014.

[126] Z. Wang *et al.*, “The Power of Short - term Load Algorithm Based on LSTM,” *IOP Conf Ser Earth Environ Sci*, vol. 453, no. 1, p. 012056, Mar. 2020, doi: 10.1088/1755-1315/453/1/012056.

[127] M. K. Kadhim and A. K. Hassan, “Towards Intelligent E-Learning Systems: A Hybrid Model for Predicatingthe Learning Continuity in Iraqi Higher Education,” *Webology*, vol. 17, no. 2, pp. 172–188, Dec. 2020, doi: 10.14704/WEB/V17I2/WEB17023.

[128] Q. Tong, G. Liang, and J. Bi, “Calibrating the adaptive learning rate to improve convergence of ADAM,” *Neurocomputing*, vol. 481, pp. 333–356, Apr. 2022, doi: 10.1016/j.neucom.2022.01.014.

[129] Y. Zhou, M. Zhang, J. Zhu, R. Zheng, and Q. Wu, “A Randomized Block-Coordinate Adam online learning optimization algorithm,” *Neural Comput Appl*, vol. 32, no. 16, pp. 12671–12684, Aug. 2020, doi: 10.1007/s00521-020-04718-9.

[130] J. S. John, “AdamD: Improved bias-correction in Adam,” Oct. 2021.

[131] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, “Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimizationb,”

Journal of Electronic Science and Technology, vol. 17, no. 1, pp. 26–40, 2019, doi: https://doi.org/10.11989/JEST.1674-862X.80904120.

[132] J. Gonzalvez, E. Lezmi, T. Roncalli, and J. Xu, "Financial Applications of Gaussian Processes and Bayesian Optimization," Mar. 2019.

[133] N. Loka, I. Couckuyt, F. Garbuglia, D. Spina, I. Van Nieuwenhuyse, and T. Dhaene, "Bi-objective Bayesian optimization of engineering problems with cheap and expensive cost functions," *Eng Comput*, Jan. 2022, doi: 10.1007/s00366-021-01573-7.

[134] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization," Mar. 2016.

[135] S. Han, H. Eom, J. Kim, and C. Park, "Optimal DNN architecture search using Bayesian Optimization Hyperband for arrhythmia detection," in *2020 IEEE Wireless Power Transfer Conference (WPTC)*, IEEE, Nov. 2020, pp. 357–360. doi: 10.1109/WPTC48563.2020.9295590.

[136] V. G. Kowti, "Stock Price Prediction using LSTM and KNN Algorithms," *Int J Res Appl Sci Eng Technol*, vol. 11, no. 9, pp. 1591–1595, Sep. 2023, doi: 10.22214/ijraset.2023.55894.

[137] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, and F. Herrera, "A survey on data preprocessing for data stream mining: Current status and future directions," *Neurocomputing*, vol. 239, pp. 39–57, May 2017, doi: 10.1016/j.neucom.2017.01.078.

[138] S. García, S. Ramírez-Gallego, J. Luengo, J. M. Benítez, and F. Herrera, "Big data preprocessing: methods and prospects," *Big Data Anal*, vol. 1, no. 1, p. 9, Dec. 2016, doi: 10.1186/s41044-016-0014-0.

[139] A. Curtis, T. Smith, B. Ziganshin, and J. Elefteriades, "The Mystery of the Z-Score," *AORTA*, vol. 04, no. 04, pp. 124–130, Aug. 2016, doi: 10.12945/j.aorta.2016.16.014.

[140] T. Blu, P. Thevenaz, and M. Unser, "Linear Interpolation Revitalized," *IEEE Transactions on Image Processing*, vol. 13, no. 5, pp. 710–719, May 2004, doi: 10.1109/TIP.2004.826093.

[141] S. G. K. Patro and K. K. sahu, "Normalization: A Preprocessing Stage," *IARJSET*, pp. 20–22, Mar. 2015, doi: 10.17148/IARJSET.2015.2305.

[142] Z. Burda, A. Görlich, A. Jarosz, and J. Jurkiewicz, "Signal and noise in correlation matrix," *Physica A: Statistical Mechanics and its Applications*, vol. 343, pp. 295–310, Nov. 2004, doi: 10.1016/j.physa.2004.05.048.

[143] T. O. Hodson, "Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not," *Geosci Model Dev*, vol. 15, no. 14, pp. 5481–5487, Jul. 2022, doi: 10.5194/gmd-15-5481-2022.

[144] T. O. Hodson, T. M. Over, and S. S. Foks, “Mean Squared Error, Deconstructed,” *J Adv Model Earth Syst*, vol. 13, no. 12, Dec. 2021, doi: 10.1029/2021MS002681.

[145] T. Chai and R. R. Draxler, “Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature,” *Geosci Model Dev*, vol. 7, no. 3, pp. 1247–1250, Jun. 2014, doi: 10.5194/gmd-7-1247-2014.

[146] H. Piepho, “A coefficient of determination (R^2) for generalized linear mixed models,” *Biometrical Journal*, p. bimj.201800270, Apr. 2019, doi: 10.1002/bimj.201800270.

[147] D. Chicco, M. J. Warrens, and G. Jurman, “The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation,” *PeerJ Comput Sci*, vol. 7, p. e623, Jul. 2021, doi: 10.7717/peerj-cs.623.

[148] W. Chen, G. Han, H. Zhu, L. Liao, and W. Zhao, “Deep ResNet-Based Ensemble Model for Short-Term Load Forecasting in Protection System of Smart Grid,” *Sustainability*, vol. 14, no. 24, p. 16894, Dec. 2022, doi: 10.3390/su142416894.

[149] P. Kumari, V. Goswami, H. N., and R. S. Pundir, “Recurrent neural network architecture for forecasting banana prices in Gujarat, India,” *PLoS One*, vol. 18, no. 6, p. e0275702, Jun. 2023, doi: 10.1371/journal.pone.0275702.

[150] R. Che and A. A. Motsinger-Reif, “A New Explained-Variance Based Genetic Risk Score for Predictive Modeling of Disease Risk,” *Stat Appl Genet Mol Biol*, vol. 11, no. 4, Jan. 2012, doi: 10.1515/1544-6115.1796.

[151] P. Kumari, V. Goswami, H. N., and R. S. Pundir, “Recurrent neural network architecture for forecasting banana prices in Gujarat, India,” *PLoS One*, vol. 18, no. 6, p. e0275702, Jun. 2023, doi: 10.1371/journal.pone.0275702.

[152] B. Koksal, R. Schmidt, X. Vasilakos, and N. Nikaien, “CRAWDAD eurecom/elasticmon5G2019.” IEEE Dataport, 2022. doi: 10.15783/c7-s58c-qn61.

[153] D. Raca, D. Leahy, C. J. Sreenan, and J. J. Quinlan, “Beyond throughput, the next generation: a 5G dataset with channel and context metrics,” in *Proceedings of the 11th ACM Multimedia Systems Conference*, New York, NY, USA: ACM, May 2020, pp. 303–308. doi: 10.1145/3339825.3394938.

[154] M. Roondiwala, H. Patel, and S. Varma, “Predicting stock prices using LSTM,” *International Journal of Science and Research (IJSR)*, vol. 6, no. 4, pp. 1754–1756, 2017.

الخلاصة

في مواجهة النمو المستمر لحركة البيانات المتنقلة وظهور تطبيقات جديدة تتجاوز الجيل الخامس، من المتوقع أن تقدم شبكات الوصول الراديوي الضخمة أداءً فائقًا وتدعم العديد من الأجهزة المتصلة .تمثل الإدارة الفعالة لهذه الشبكات تحديات كبيرة، بما في ذلك إدارة التنقل الفعالة والتنبؤ الدقيق بمؤشر جودة القناة وضمان الجودة العالية للخدمة .تهدف هذه الأطروحة إلى معالجة هذه التحديات من خلال تطوير نظام هجين متكامل يستخدم تقنيات الذكاء الاصطناعي، مع التركيز بشكل خاص على نموذج هجين يجمع بين نقاط قوة شبكات الذاكرة الطويلة قصيرة المدى و وحدات التراكم المشبكة.

في المرحلة الأولى من هذا البحث، تم تنفيذ العديد من النماذج التعلم الآلي، بما في ذلك الشبكات العصبية الاصطناعية والشبكات العصبية العميقة وآلات الدعم البياني والذاكرة القصيرة الطويلة، وتمت مقارنتها من حيث كفاءتها في توقع مؤشر جودة القناة. وهذا يشكل أساساً لاستكشاف نماذج أكثر دقة.

في المرحلة الثانية، قمنا بتطوير وتحسين نظام متكامل هجين جديد يستخدم شبكات الذاكرة القصيرة ووحدات التراكم المشبكة بمساعدة خوارزميات ضبط معلمات مختلفة و هما هايبرباند والبحث البياني. قمنا باختيار هذه النماذج المحددة وخوارزميات ضبط المعلمات بناءً على إمكانية التقاط الطبيعة الديناميكية والتسلسلية للبيانات المدخلة، وهو أمر بالغ الأهمية لتوقع مؤشر جودة القناة بدقة في سيناريو الوصول الراديوي .

يُظهر البحث أن النظام المقترح هجين الذاكرة القصيرة ووحدة التراكم المشبكة، عند تصحيحه وتحسينه بشكل مناسب، يؤدي بشكل أفضل في توقع مؤشر جودة القناة مقارنة بالنماذج الأخرى من تعلم الآلة. يُظهر هذا التحسن في دقة توقع مؤشر جودة القناة في إدارة الشبكة بشكل أفضل، وتخصيص الموارد بشكل أكثر كفاءة وتحيين جودة الخدمة لشبكات الوصول الراديوي للجيل الخامس.

في المرحلة الأولى، تم تقييم العديد من النماذج لتوقع مؤشر جودة القناة في شبكات الوصول الراديوي الضخمة. من بين هذه النماذج، أظهرت الذاكرة القصيرة الطويلة أفضل أداء بقيم MSE بنسبة 0.811544 وRMSE بنسبة 0.900858 وMAE بنسبة 0.608555، مع تحقيق R2 بنسبة 0.883451.

في المرحلة التالية، تم تقديم نظام متكامل هجين لذاكرة القصيرة الطويلة ووحدة التراكم المشبكة. أسفر هذا النموذج عن تفوق على جميع النماذج الأخرى، مقدمًا نتائج استثنائية بقيم MSE بنسبة 0.0182 وRMSE بنسبة 0.1350 وMAE بنسبة 0.1197، وR2 الرائع بنسبة 0.9975. هذه النتائج تؤكد بوضوح أن نظام الذاكرة القصيرة الطويلة ووحدة التراكم المشبكة هو الخيار الأكثر دقة وصلابة لتوقع مؤشر جودة القناة في سياق شبكات الوصول الراديوي الضخمة. إمكانياته تمتد لتحسين إدارة الشبكة، وتحسين تخصيص الموارد، في عصر التقنيات اللاسلكية المتقدمة.

جمهورية العراق
وزارة التعليم العالي و البحث العلمي
جامعة بابـــل
كلية تكنولوجيا المعلومات
قسم شبكات المعلومات

تطوير شبكة الجيل الخامس للتنبوء بمؤشر جودة القناة بالاعتماد على تقنيات الذكاء الاصطناعي الهجينة المُحسّنة

أطروحة مقدمة الى
مجلس كلية تكنولوجيا المعلومات - جامعة بابل كجزء من متطلبات
نيل درجة دكتوراه فلسفة في تكنولوجيا المعلومات / شبكات
المعلومات

من قبل
كرار ابراهيم عبدالاميرعباس

بإشراف
ا.د ستار بدر سدخان المالكي

1445 A.H. **2023 A.D.**