ADDIS ABABA UNIVERSITY
ADDIS ABABA INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Deep Learning-based Cell Performance Degradation Prediction

Submitted by

Betelehem Dagnaw

Supervised by

Dr.-Ing. Dereje Hailemariam

A Thesis Submitted to the School of Electrical and Computer Engineering of Addis Ababa University in Partial Fulfillment of the Requirements for the Degree of Masters of Science in Communication Engineering.

June, 2022
Addis Ababa, Ethiopia

Declaration

I, the undersigned, declare that this thesis is my original work, has not been presented for a degree in this or any other university, and all sources of materials used for the thesis have been fully acknowledged.

Student Name: Betelehem Dagnaw

Signature:_____

Date of submission:_____

This thesis has been submitted for examination with my approval as a university advisor.

Advisor Name: Dr.-Ing. Dereje Hailemariam

Signature:_____

Abstract

In light of rapid developments in the telecommunications sector, there is a growing volume of generated data as well as high customer expectations regarding both cost and Quality of Service (QoS). For Mobile Network Operators (MNOs), the changing dynamics of radio networks pose challenges in coping with the increased number of network faults and outages, which both lead to performance degradation and increased operational expenditures (OPEX). Human expertise is required to diagnose, identify and fix faults and outages. However, the increasing density of mobile cells and diversification of cell types are making this approach less feasible, both financially and technically. In this paper, relying on the power of deep learning and the availability of large radio network data at MNOs, we propose a system that predicts the performance degradation of cells using key performance indicators (KPIs). Data collected from the Universal Mobile Telecommunications Service (UMTS) network of an operator located in Addis Ababa, Ethiopia, is used to build models in the system. The proposed system consists of a multivariate time series forecasting model, which forecasts KPIs in advance. In addition, a cell performance degradation detection model, which detects anomalous records in the KPI data based on the forecasting model outputs. Convolutional Long Short-Term Memory (ConvLSTM) and LSTM Autoencoders are cascaded for prediction and degradation detection. The results show that the system is capable of predicting KPIs with a Root Mean Square Error (RMSE) of 0.896 and a Mean Absolute Error (MAE) of 0.771, and detecting degradation with 98% accuracy. This research can therefore contribute significantly to improving network failure management systems by predicting the impact of upcoming cell performance degradations on network service before they occur. This research can therefore contribute significantly to improving network failure management systems by predicting the impact of upcoming cell performance degradations on network service before they occur.

Keywords: Performance degradation detection, Multivariate forecasting, LSTM, CNN-LSTM, ConvLSTM, Deep Learning Network, UMTS, KPI

Acknowledgements

Praise and gratitude to Almighty God and the Virgin Mary for the blessings and guidance in my life.

It is with the deepest gratitude that I acknowledge and thank my advisor Dr. -Ing. Dereje Hailemariam for his unwavering support, encouragement, and careful guidance throughout this process. Having the opportunity to work with him is an incredible experience. His commitment to his responsibilities is unsurpassed. The completion of this thesis would not have been possible without his guidance, constant follow ups, comments, and suggestions. Thank you very much for being so generous with your time and advice.

I would also like to thank Ms. Bethelhem Seifu and Dr. Tsegamlak Terefe for their valuable comments and time.

Furthermore, I am grateful to Alemu Sisay, who was there for me in my time of need. Without him, it wouldn't be easy. Thank you.

In honor of the courage, the prayers, and the unwavering love they are offering to me, I would also like to thank all my families and friends especially Amsalu, Banchi, Dagmawi, Eskedar, and Habte for their love and friendship.

Last but not least, I would like to thank the University of Gondar and Addis Ababa University, for giving me a chance to complete my Master's studies and those who provide me data for my research.

Contents

List of Figures

List of Tables

List of Acronyms

2G	Second Generation
3G	Third Generation
3GPP	Third Generation Partnership Project
4G	Fourth Generation
AI	Artificial Intelligence
AuC	Authentication Center
BSs	Base Stations
CDR	Call Drop Rate
CN	Core Network
CNMS	Cellular Network Management System
CNN	Convolutional Neural Network
CNN-LSTM	Convolutional Neural Network and Long Short Term Memory
COD	Cell Outage Detection
ConvLSTM	Convolutional LSTM
CS	Circuit Switch
CS CDR	Circuit Switched Call Drop Rate
CS HOSR	Circuit Switched Handover Success Rate
CSSR	Call Setup Success Rate
CSSR HSUPA	Call Setup Success Rate High-Speed Up-link Packet Access
DNN	Deep Neural Network
EIR	Equipment Identity Register
EMSes	Element Management Systems
eNB	Evolved Node B
GAN	Generative Adversarial Networks
GGSN	Gateway GPRS Support Node
GMSC	Gateway MSC
GSM	Global System for Mobile Communication
GSM	Global Systems for Mobile Communications

HLR	Home Location Register
HSDPA	High-Speed Downlink Packet Access
KNNs	K-Nearest Neighbors
KPI	Key Performance Indicators
LSTM	Long Short Term
LSTM	Long Short Term Memory
LSTM	Long Short-Term Memory Networks
LTE	Long Term Evolution
MDT	Minimized Drive Tests
ME	Mobile Equipment
ML	Machine Learning
MNOs	Mobile Network Operators
MSC	Mobile Switching Center
MT	Mobile Terminal
NMS	Network Management System
NOC	Network Operation Center
OPEX	Operational expenditures
OPS	Operation System
OSS	Operation Support System
PS	Packet Switch
PS CDR	Packet Switched Call Drop Rate
PS HOSR	Packet Switched Handover Success Rate
PS RAB	Packet Switched Radio Access Bearer
PS RRC	Packet Switched Radio Resource Control
QoS	Quality of Service
RAN	Radio Access Network
ReLU	Rectified Linear Unit
RNC	Radio Network Controller
RNN	Recurrent Neural Network
RNS	Radio Network Subsystems

ROC	Receiver Operating Curves
RRM	Radio Resource Management
RSRP	Reference Signal Received Power
SGSN	Serving GPRS Support Node
SINR	Signal to Interference Plus Noise Ratio
SON	Self-Organized Networks
SVMs	Support Vector Machines
UE	User Equipment
UMTS	Universal Mobile Telecommunication System
USIM	Universal Subscriber Identity Module
UTRAN	UMTS Terrestrial Radio Access Network
VLR	Visitor Location Register

Chapter 1

Introduction

The rapid growth of mobile technologies has enabled easy access to a wide range of services supported by a variety of technologies. As mobile networks and devices advance, the number of mobile subscribers is increasing dramatically[1]. Increasing base stations to provide enough capacity and coverage is no longer enough in today's world, as subscribers need consistent access anywhere and any time[2]. As a result, mobile operators must constantly monitor their networks and work on quality of service (QoS) assurance to ensure that subscribers receive the best possible service. The changing dynamics of the radio network, on the other hand, present operators with problems in terms of maximizing network efficiency while lowering Operational Expenditures (OPEX) [3]. Globally, increased emphasis on service automation is being driven by competition in liberalized telecommunication markets and subscriber's demand for more complicated services [4]. It's critical to detect and localize faults and outages in the network as fast as possible to prevent the damage they cause. But, traditional schemes of manual network deployment, configuration, optimization, and maintenance are inefficient and incur huge operational expenditures[5]. Operators are working toward self-organized networks (SON) with self-healing functionality to detect and compensate network faults to achieve network monitoring automation[6]. Cell outages and network performance degradation are serious problems in mobile networks. Outages and degradation are traditionally detected manually through alarms and user complaints, which is a time-consuming and error-prone process. Manual and subjective detection of faulty cells from hundreds of different key performance indicator (KPI) measured values is nearly impossible[7]. Thus an automatic detection mechanism need to be implemented. Researchers have adopted different approaches to come up with the solution for system automation.

Machine-learning approaches have been successfully used to detect outages of a cell from indirect evidence such as reports from the user equipment and neighboring cells[8]. Deep-learning algorithms surpassed the performance of many traditional machine learning techniques and

achieved breakthroughs in different domains: e.g, natural language processing[9], spectrum prediction[10], mobile data prediction[11], and computer vision [12]. Deep-learning approach also plays a definitive role in detecting outage cells to improve network QoS and reduce OPEX for the network operators.

1.1 Statement of the problem

While mobile device innovation and usage are growing at a dramatic pace, network technology is progressing at a much slower rate. This disparity results in an inefficient system causing outages and performance degradation in the network system. Cell outage and Performance degradation occurs in cellular networks due to devices malfunctioning, power outages, faulty links, or misconfiguration of parameters during network operation[8]. These performance degradation induce a low QoS that leads to customer dissatisfaction and loss of revenue. The operators use manual methods of detecting outages and performance degradation in the network. However, the manual detection process is labor-intensive, costly, requires adequate expertise, and prone to errors[13].

In summary, the main problems are:

- Detecting outages and cell performance degradation manually is inefficient in terms of cost, maintenance time, and labor and is prone to errors, so it will lead to incorrect analyses and let anomalies remain unfixed.
- Though there are works on detecting outages automatically[14, 15, 16], an end-to-end system showing the logical relationship between network failure reasons and generated alarms has rarely been reported. Therefore, corrective actions are hindered.

So far, cell outage detection depends on drive tests and subscriber complaints to identify the affected cells. Due to increased operational costs, this strategy is quickly becoming unsustainable. Operators must take proactive maintenance actions to ensure QoS and customer satisfaction. Because it is not part of a scheduled program, corrective maintenance will take a lot of time and effort. A system's availability will also decline since services will be disrupted until the repairing or replacing work is completed. Furthermore, the cost of downtime is expensive at peak hours and on critical sites. A fault prediction system would be useful in resolving unexpected service interruptions, ensuring QoS, and thus increasing revenues. Therefore, the goal

of this thesis is to anticipate the upcoming performance degradation of cells that have a vast impact on the network service before they occur. A deep learning-based approach is proposed to predict a cell's performance degradation based on reported key performance indicators.

1.2 Literature Review

Different approaches have been used in literature for detecting, diagnosing, and compensating faults and cell outages in a mobile networks. These approaches range from heuristic solutions based on prior experience and following predefined criteria[17], to a learning based solutions[18]. Following is a discussion of related literature to our work.

In[19] by I. De-La-Bandera et al., Handover statistics were used to detect cell outages. Different scenarios of cell outages are discussed in the Cell Outage Model. Firstly, a cell outage that does not affect the Evolved Node B (eNB) and the eNB generates KPIs from the outage cell. The second instance is an outage that affects eNB and no KPI data available from the Operation support system (OSS). In a third scenario, Cell is not in the outage, but the eNB-OSS connection is down. According to LTE simulator results, the algorithm can detect a cell outage when KPIs from the cell is either available or not. There is a drawback to the algorithm, it can't detect cells in outages with very low traffic.

Asghar et al.[20] have proposed using Pearson's correlation factor to match cells based on the number of active users associated with each cell. According to the algorithm, a cell is considered degraded if its correlation with multiple previously well-correlated cells falls below a certain threshold. According to the authors, the proposed method not only detects degradation but also detects full outages. The performance of this algorithm is, however, highly dependent on correlated cells. Therefore, if multiple correlated cells suffer the same degradation, it may not be detected.

Cell Outage Detection (COD) frameworks based on mobile terminal-assisted data collection are proposed by Zoha et al. in[3]. After collecting UE-reported MDT measurements, the researchers extracted a minimal KPI representation by projecting them to a low-dimensional embedding space. Two types of anomaly detection methods were employed, a local outlier factor-based detector (LOFD) and a one-class support vector machine-based detector (OCSVMD) with embedded measurements. Both algorithms were evaluated and compared. In addition, the

geo-location of each measurement of the COD framework was used to locate the location of the faulty cell. In this study, a fully dynamic LTE simulation tool was used to simulate the LTE network consisting of 27 e-NodeBs and to test the detection performance of both the OCSVMD and the LOFD. Based on the results, OCSVMD was more effective in identifying abnormal measurements than LOFD.

In [21] the sleeping cell problem (where the radio fails without being reported to network management) is addressed using a deep learning algorithm. A deep Recurrent Neural Network (RNN) is used to process the reference signal received power (RSRP) reports detecting degradation in cell radio performance. In this study, a variety of RNN configurations are used to study the effect of hiding layers and conclude that in this case, a single hidden layer achieves the target sensitivity, which leads to highly efficient runtime performance. ROC curves (Receiver Operating Curves) are used to gauge the accuracy of the results. Using Traditional classification techniques for cell outage detection can lead to biased classifiers when the training samples of one class greatly exceed those of other classes.

Zhang et al.,[5], illustrate how generative adversarial neural networks (GAN) can be used to improve data quality in situations where fault data is rare. By combining Generative Adversarial Networks (GAN) with Adaboost, specifically, the proposed approach uses GAN to change the distribution of imbalanced datasets to produce a more balanced distribution, and then Adaboost to classify the calibrated datasets, they showed an improvement in classification performance for imbalanced cell outage data based on several metrics, including Receiver Operating Characteristics (ROC), precision, recall rate, and F-value.

Masood et al. [22] proposed a deep learning technique called deep autoencoder-based sleeping cell detection, which uses the MDT measurement data generated by the user equipment to detect a special case of cell outage that occurs without triggering any alarm. The data consists of neighboring and serving Base stations (BSs) reference signal received power (RSRP) and signal to interference plus noise ratio (SINR) However, this method only takes into account spatial data collected for a single occurrence, resulting in instantaneous detection of sleeping cells. As a result, the identified anomaly may be temporary, with little influence on QoS, and may disappear by the time it is compensated.

In [23], Hussain et al. developed a feed-forward deep neural network (DNN) framework to

detect anomalies in single cells of cellular networks. Real Call Detail Records are pre-processed to obtain feature vectors that correspond to user activities that are accepted as input for the algorithm. The output is a binary number indicating 0 as normal and 1 as an anomaly.

In most papers, machine learning techniques are employed. Machine learning techniques require human input for feature extraction, and as data volumes and velocity increase, they become incapable of dealing with the dimensionality and variety of data [24]. In comparison to machine learning, deep learning has significant advantages. By learning deep feature representations from data, deep learning models can be used to extract discriminative features and potentially eliminate the need for human feature engineering. For this reason, deep learning algorithms are implemented in our work. For outage detection, most techniques analyze spatial data only. This means that the KPIs that are used for outage detection is generated over a set of spatial points for a single time instant. As a result, outages detected by these solutions are instantaneous. It raises the issue of extremely short-lived outages that have little to no effect on subscriber QoE. Our work addresses this issue by considering both the temporal and spatial dimensions of cell data to differentiate between temporary and long-term degradation.

In the majority of the works cited above, the focus is on outage detection, which is a binary classification between outage cells and healthy cells, yet the detection of degraded cells is neglected. In contrast to a full outage, performance degradation does not trigger network alarms and is invisible to operators. Network performance degradation can result in poor QoS, which can increase customer dissatisfaction and churn. Therefore, in addition to detection, our work also involves forecasting the time series KPI data and predicting performance degradation in advance. Barreto et al.[25] suggest that, although simple, using one variable data stream for anomaly detection is not always effective. As a result, rather than using one data stream to train outage detection algorithms as in [26, 27, 28], multivariate data input is considered in our case.

1.3 Objective

1.3.1 General Objective

The general objective of this thesis is to implement deep learning based cell performance degradation detection and prediction by taking UMTS network KPI data as a data source.

1.3.2 Specific Objectives

The specific objectives to be accomplished in this thesis are:

- Identify the KPIs to be used for performance degradation detection.
- Design multivariate, multi-step time series forecasting models which are able to forecast future KPI records;
- Compare and evaluate the performance of the proposed model;
- Select and implement the best deep learning model to detect the degradation of UMTS cells;
- Concatenate the forecasting model with the cell degradation detection model so as to predict the future cell performance degradation;
- Draw conclusions and recommendations based on the findings of the study.

1.4 Methodology

The steps outlined below are followed during this research:

- **Literature review:** Throughout the research, a literature review is conducted to formulate the problem, identify gaps in the prior work, and propose a method.
- **Propose an Approach:** Once a clear problem is formulated, a solution approach is proposed that can address the problem.
- **Data Collection and Preparation:** KPI data from UMTS cells is collected on an hourly basis from performance recording systems. Data is then preprocessed to make it suitable for the model. The Python programming language is used for pre-processing the data-set.

- **Design and Implementation:** Once the approach is selected, the proposed model is designed and trained using the prepared data.
- **Results and Evaluation:** A proposed approach is then tested using the test data.

1.5 Scope

This thesis implements a deep learning algorithm to assess the performance degradation of cells. Due to the availability of data, our work uses UMTS network KPI data record as the only data source. But our work is also applicable to other network technologies too. The scope of the study is limited to the detection and prediction of cell performance degradation. The root cause analysis of the performance degradation, as well as the effect of degraded cells on neighboring sites, are not addressed. Consequently, further investigation could be carried out to uncover the cause and the inter-dependency of the degradation in neighboring sites. In this thesis, only a few deep-learning algorithms are implemented (LSTM, CNN-LSTM, and ConvLSTM). No comparisons with other machine learning algorithms are presented.

1.6 Significance of the Study

The network brings convenience and efficiency to people's life and work, and at the same time, the performance degradation and faults in the network also affect us. Therefore, it is imperative to predict performance degradation and faults in advance. This thesis work tries to give an insight on how deep learning algorithms can be used to predict the performance degradation of network cells. This would help network operators to actively follow the network status and be in position to take actions.

1.7 Contributions

Exploiting the advantage of deep-learning algorithms for cell outage and performance degradation detection is rarely reported in previous works. By taking this fact into account, this research work aimed to implement deep-learning techniques to predict the performance degradation of UMTS cells. Therefore, the main contributions of this research work are as follows:

- Studies the correlation between KPIs to use the least possible number of KPIs.
- Uses a multivariate multi-step time series KPI data for the forecasting task. This increases prediction accuracy.
- Implement deep learning algorithms to predict performance degradation in advance and provides a comparative analysis between used algorithms.
- Present a proactive approach to automate the current manual anomaly detection process

1.8 Organization of the Thesis

The thesis consists of five chapters. The problem statement,literature review, general and specific aims, scope, and contribution of the thesis are all covered in the first chapter. The second chapter focuses on the basic concepts of the UMTS network architecture and the deep learning techniques used. The third chapter presents a data set description and preprocessing techniques applied. A detailed explanation of the results obtained from the conducted sets of experiments is discussed on the fourth chapter. The final chapter concludes the thesis work and suggests future research directions.

Chapter 2

Fundamental Concepts

In this chapter, the fundamental concepts, network architecture and KPIs of the UMTS network are discussed. In addition, a brief overview of deep neural networks which are employed in this thesis work is provided.

2.1 UMTS Network

UMTS is the Third Generation (3G) of mobile cellular systems that was developed by the 3rd Generation Partnership Project (3GPP). The introduction of UMTS was motivated by the desire to provide faster data rates and better voice services than its predecessor, the Second Generation (2G) Global System for Mobile Communications (GSM) [29]. UMTS comprises three functional aspects as a network architecture, which is an infrastructure that delivers services between two endpoints: User Equipment (UE), Radio Access Network (RAN), and Core Network (CN) [30]. The capacity to provide services anywhere and at any time is one of UMTS' most important features. Mobile equipment can be used for communication, entertainment, business, and a variety of other services. The data traffic demand is increasing as the number of smartphones with various application setups rises. This increase forces network operators to devise a new strategy for the network's existing infrastructure. With the cellular industry transitioning from 3G to Fourth Generation (4G), UMTS serves as the foundation for the 3GPP's new Long Term Evolution (LTE) radio technologies (LTE).

2.2 UMTS Network Architecture

The UMTS network architecture is categorized into three UE, UTRAN, and CN [31]. The Radio Interface, Uu, connects the UE to the UTRAN; and the CN-UTRAN interface, Iu, connects

the UTRAN to the CN. Figure 2.1 shows the UMTS architecture.

2.2.1 User Equipment

The UE is a key element of the 3G UMTS network architecture. It forms the ultimate interface between the user and Node B [32]. The UE consists of the Mobile Equipment (ME) and the Universal Subscriber Identity Module (USIM). The ME is the single or multimode Mobile Terminal (MT) used for radio communication over the Uu interface. The USIM is a smart card that holds the subscriber identity, performs authentication algorithms, and stores authentication and encryption keys, and information needed at the terminal.

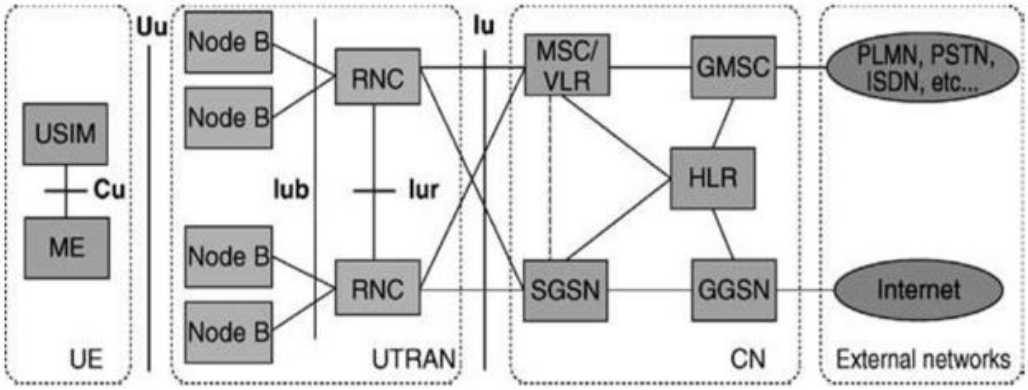


Figure 2.1: UMTS architecture [33].

UMTS Terrestrial Radio Access Network

The UTRAN is made up of many Radio Network Subsystems (RNS). An RNC and Node Bs are included in each RNS. The RNC is connected to the Node Bs via the Iub interface. The Node B's translates data flow between the Iub and Uu interfaces and participates in Radio Resource Management (RRM). It processes the air interface (channel coding, rate adaptation, spreading, synchronization, and power control). The Radio Network Controller (RNC) is in charge of the Node Bs that are connected to it, as well as executing the RRM. The Iur interface allows RNCs to communicate with one another. The RRM ensures power control, packet scheduling, and handover control in the outer loop. Handover, radio coverage, RRM, and control, system access control, security, and privacy are all UTRAN features [31].

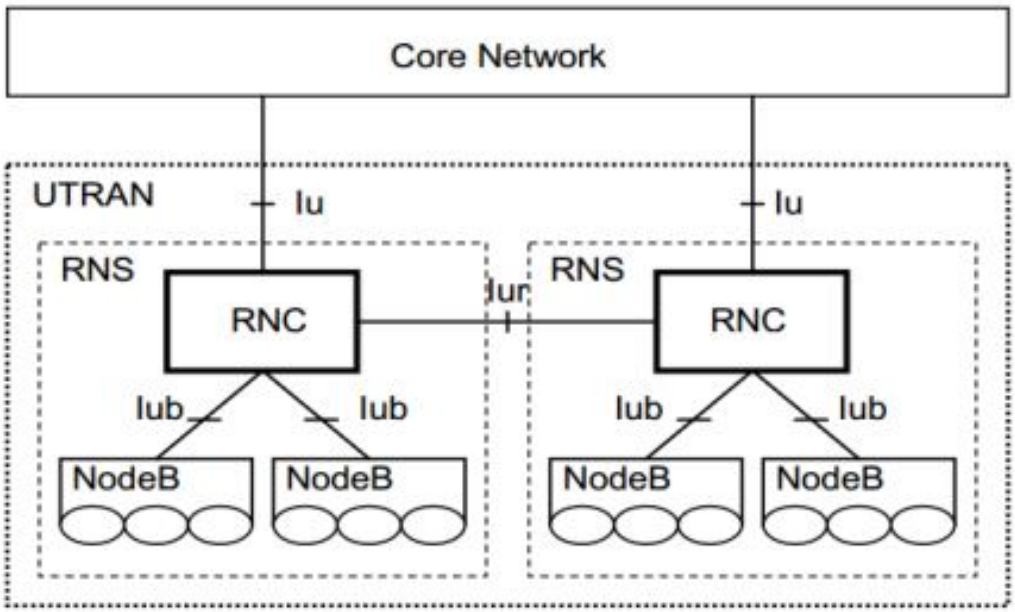


Figure 2.2: UTRAN Architecture [34].

Core Network

The core network is the backbone of the mobile communication system. CN is in charge of transport functions such as call switching, data connection routing to external networks, and mobile user tracking. Transport functions, mobility management, processing subscriber databases with their information, service controlling activities, billing, and so forth are all part of the CN function [31]. The CN aggregates two major domains, Packet Switch (PS) and Circuit Switch (CS) networks. CS is in charge of the public switched telephone network, whereas PS is in charge of switching and routing calls and data to external systems. Mobile Switching Center (MSC), Visitor Location Register (VLR), and Gateway MSC are some of the circuit-switched parts (GMSC). The Serving GPRS Support Node (SGSN) and Gateway GPRS Support Node, on the other hand, are packet-switched components (GGSN). The Mobile Service Switching Center (MSC), the Home Location Register (HLR), the Visitor Location Register (VLR), the Authentication Center (AuC), the Equipment Identity Register (EIR), the Getaway MSC (GMSC), the Serving GPRS Support Node (SGSN), and the Gateway GPRS Support Node are the main components of CN (GGSN) [30].

2.2.2 UMTS Network Interfaces

UMTS network facilitates communication between the above subsystems using different interfaces, some of them are described as follows [31].

- C_u interface: connects the ME to USIM smart card.
- U_u interface: the radio link between Node B and the UE, is called as "air" interface.
- I_u interface: connects the CN to the UTRN, specifically, the IuCS interface connects the circuit-switched network to the UTRAN, and the IuPS interface connects the packet-switched network to the UTRAN
- I_{ub} interface: connects RNC to multiple Node B.
- I_{ur} interface: provides the connection between RNC and allows soft handover between them.

2.2.3 Network Management System

For a network operator who maintains and operates the network, a network management system serves as a human interface with the network. It's essential for network quality management tasks like visualizing the state of a network, including the component communication devices, from many angles. It informs the network operator whether the entire network is up and running or there are any problems [35]. A management system for managing and maintaining these networks is made up of element management systems (EMS) that handle the individual network devices directly and a network management system (NMS) that oversees them all. Figure 2.3 shows conceptual diagram of a mobile network management system. The following are the main functions of a network management system [36].

- Configuration management: configuration management includes the management of network device configuration information and the modification of network device settings. Wireless resource management, station data management, and software upgrades are some of the features it offers. It also includes plug and play, which recognizes when a new device is added to the network and gets the newest firmware automatically[37].
- Performance management: This function checks the traffic and wireless resource condition of network devices to ensure network availability, and alerts if any threshold is exceeded[31].

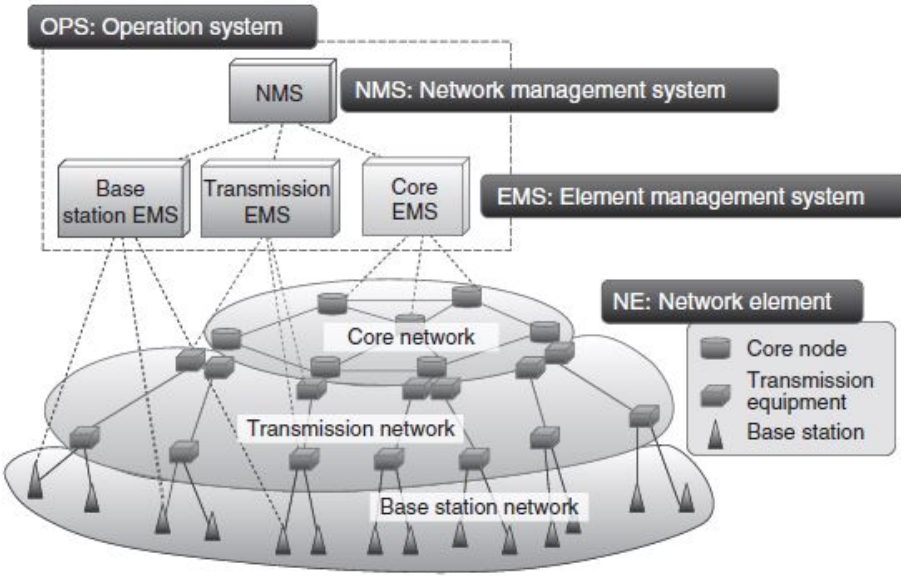


Figure 2.3: Mobile network management [35].

- Security management: Access to resources managed by the network management system is controlled by security management (by user management, access log, etc.) [37].
- Fault management: for a mobile network, fault management serves as a fault monitoring function. It keeps track of the status of the network and notifies when faults happened. This can be determined by monitoring various alarms generated from network elements such as KPI, HO, and MDT data, as well as observed abnormal network behavior [35]. In traditional network management, knowing the state of the overall network and each network element is critical for detecting network faults. Each network element has counters or performance measurements that are used to indicate the state of the network. To detect network element failures, KPIs that integrate individual measurement values into more useful abstract values are employed instead of raw performance measurement [36]. If the observed KPI value exceeds the predefined threshold value, an alarm is triggered, signaling a NE fault. The root cause investigation and data interpretation for an observed behavior are complex, necessitating the use of a human operator to put the data into context.

2.3 Key Performance Indicators

Cellular networks are growing increasingly complicated, and with the complexity, more services are being added, which then means there are more network parameters to be monitored. Operators must constantly monitor network performance to maintain control of these highly dynamic networks[1]. KPIs are the most common measures used to assess process performance as indications of quantitative management and to track progress toward an operator's goals. KPI is a measured statistic that shows how well a network serves its users. KPIs indicate a network's capability level, which may be monitored using a variety of measurements. KPIs are defined by the definition and measurement of key internal network system metrics[38]. According to 3GPP [32], the following are some of the UTRAN KPIs categories.

2.3.1 Accessibility

Accessibility KPIs are used to assess whether users' demanded services are available in a given situation. This also relates to the service's availability when customers needed it[31, 32]. The main parameters for accessibility performance are RRC connection procedure, RAB setup procedures and Call setup success rate.

Radio Resource Control (RRC) setup success rate

RRC Setup Success Rate RRC is the major control signaling protocol between UEs and networks. All procedures regarding the establishment and release of a connection, including paging, are controlled by RRC messages. RRC messages are also used for configuration measurements and reporting during handovers. There are three main and possible reasons for RRC setup failure: no response from RNC, rejection by RNC and no response from UE after RNC responds[32]. The RRC setup success rate is calculated based on the counter at RNC when it receives the UE's RRC connection request [32]. Complete RRC connection setup can be illustrated by the diagram in Figure 2.4.

Number of RRC connection attempt is collected at point A, and the number of successful RRC connection is calculated at point C. Then the RRC setup success rate is calculated as in

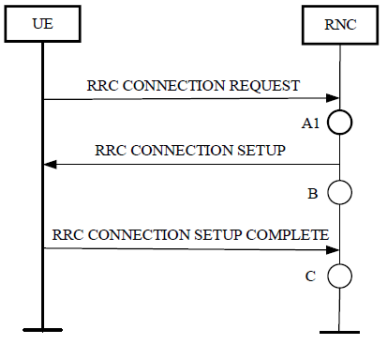


Figure 2.4: RRC connection Setup procedure.

equation 2.1.

$$RRC\ Setup\ Success\ Rate = \frac{RRC\ Connection\ setup\ complete}{RRC\ Connection\ request} * 100\% = \frac{C}{A} * 100\% \quad (2.1)$$

Radio Access Bearer setup success rate (RAB) setup success rate

The RAB protocol establishes a fixed path of communication between user equipment and the core network. Networks build end-to-end QoS connections within radio access bearers. CN initiates the RAB setup procedure[39]. CN sends "RAB ASSIGNMENT REQUEST" to RNC. RNC increments the counter at point A of Figure 2.5 by one when it receives this message. Based on the request, RNC sends UE a "RADIO BEARER SETUP" message. After this, the UE may send a "RADIO BEARER SETUP COMPLETE" or "RADIO BEARER SETUP FAILURE" message to RNC. The counter at point B in Figure 2.5 increments by one if RNC receives a radio setup complete message from the UE.

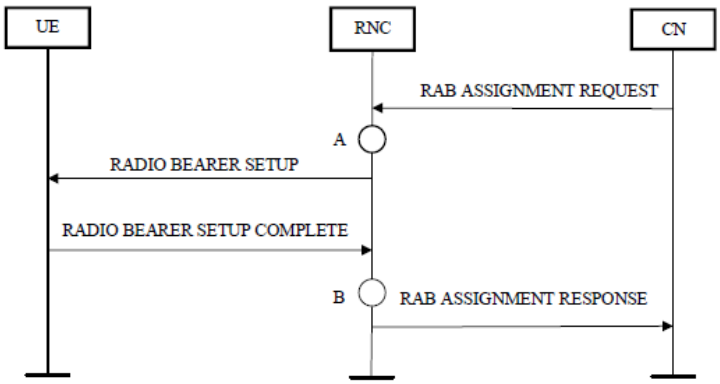


Figure 2.5: RAB setup procedure.

$$RAB\ Setup\ Success\ Rate = \frac{Number\ of\ RAB\ Setup\ Success}{Number\ of\ RAB\ Setup\ Attempt} * 100\% \tag{2.2}$$

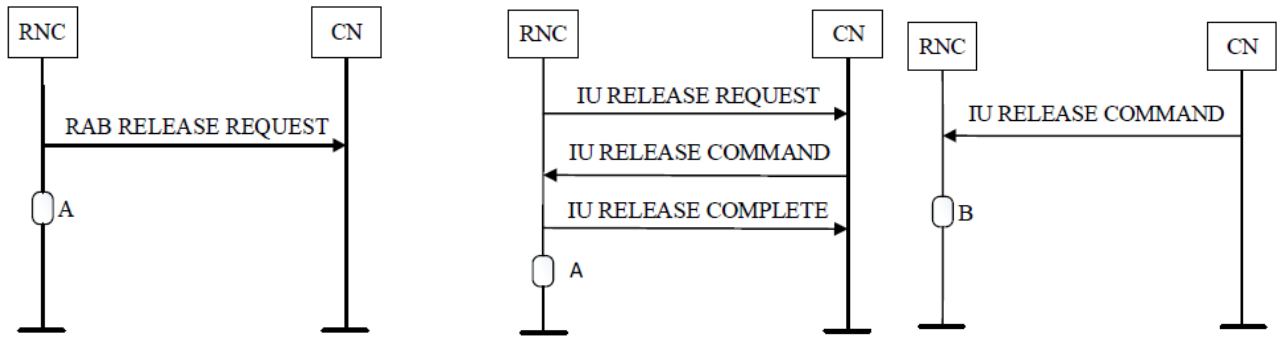
Call setup success rate (CSSR)

Call setup success rate describes the ratio of successful call establishments[32]. It is calculated based on the RRC Connection Establishment Success Rate for call setup as well as the RAB Establishment Success Rate for all RAB types as shown in equation 2.3.

$$CSSR = \frac{Number\ of\ RRC\ Setup\ Success}{Number\ of\ RRC\ Setup\ Attempt} * \frac{Number\ of\ RAB\ Setup\ Success}{Number\ of\ RAB\ Setup\ Attempt} * 100\% \tag{2.3}$$

2.3.2 Retainability

Retainability KPIs measure the cell's ability to hold the user and provide the requested services for the desired duration[32]. A call drop may happen during a transaction for many reasons, including poor coverage, missing neighbor cells, or interference both up and downlink. A call drop occurs when the RNC triggers a release request by sending an "IU RELEASE REQUEST" or "RAB RELEASE REQUEST" message. When the RNC sends an IU or RAB release request to the CN, the counter at point A in Figures 2.6a or 2.6b increments by one. As shown in Figure 2.6c, the counter at point B increments by one when a request is initiated by the CN.



(a) (b) (c)

Figure 2.6: Call Drop Procedures.

2.3.3 Availability

Availability KPI is used to measure the ability of the network to hold and provide the services to the users. The cell availability KPI measures the percentage of time the cell is considered available or if the NodeB can provide RAB service in the cell[32, 39].

2.3.4 Mobility

Mobility KPIs are used to measure the performance of the network and handle the movement of the user while retaining the service. Mobility KPIs include: (Packet Switched Handover success rate and Circuit Switched Handover success rate). Its computation depends on the ratio between the successful handovers over the number of all handover attempts executed by the Node B.

Handover Success Rate

Handover Success Rate KPI measured the rate of successfully transferring an ongoing call or data session from one channel to another in a cellular network. There are different types of handover measurements[31].

- Hard Handover: Hard Handover adopted a 'break before make' policy, in which the first link is broken then the second is re-established. This means that old radio links in the UE must be removed before new radio connections can be established.
- Soft Handover: Soft HO means that radio links are added and removed so that the UE always maintains at least one radio link to the UTRAN (make-before-break). This occurs when a UE is within the overlapped coverage area of two cells.
- Inter-RAT/Inter-system Handover: In this case, the handover is between WCDMA FDD and different systems such as WCDMA TTD or GSM.

Integrity

Integrity KPI's are used to measure the character or honesty of the network to its users such as the downlink throughput, the uplink throughput, and latency that reflect the integrity of the actual services provided to the users. Such KPIs could be measured either hourly or averaged over a day. [32, 39].

2.4 Deep Learning Algorithms

Deep learning is a subfield of machine learning that emphasizes learning successive layers of increasingly meaningful representations as a new approach to learning representations from data. The word "deep" in deep learning refers to the concept of successive layers of representations, not to any kind of deeper understanding achieved by the technique[40]. The depth of a data model is defined as the number of layers that contribute to a model of data. In deep learning, these layered representations are almost always learned by neural network models, which are arranged in literal layers piled on top of each other. Neural networks are a type of machine learning approach that simulates the learning mechanism in biological organisms[41]. Although the name "neural network" refers to neurobiology, deep-learning models are not brain models. There is no indication that the brain uses learning methods similar to those seen in modern deep-learning models [42]. The aggregation of the basic unit of classical machine learning units, such as least square regression, in such a way that they learn the weights of numerous units to reduce the prediction error, is one property that makes neural networks robust. One thing to keep in mind is that if deep learners and neural networks aren't seen as a whole, the learning algorithms of individual units or subunits aren't that different from machine learning approaches. The learning ability of neural network systems improves as the size of the computational units increase, making them very powerful computational systems capable of learning very complex functions. Similar to the shallow neural network layer, the neural network layer in DNN is divided into three categories input layer, hidden layer, and output layer. DNN has a complex structure with multiple hidden levels. A deep network, according to popular belief, should have at least three hidden layers, whereas a very deep network should have at least ten hidden layers [43]. DNN may learn more complex functional relationships by using several hidden layers. The more hidden layers of the network there are, the higher the accuracy was in specific cases. Figure 2.7 depicts the DNN structure. The features of sample data dictate the number of neurons in the input layer. Each hidden layer has many neurons, which can be calculated using an empirical formula. An activation function transforms the output of each hidden layer into a nonlinear form, and common nonlinear activation functions include sigmoid, Rectified Linear Unit (ReLU), and others. The number of sample labels determines the number of neurons in

the output layer. The neurons in one layer will be connected to neurons in another layer using an interconnection method that is determined by the neural network's architecture[42]. In a fully connected deep neural network structure, for example, each layer's n neuron is intended to be connected to all n+1 layer neurons and all n-1 layer neurons at the same time. However, because input layer neurons are the initial layer, they are not connected to n-1 layers, and output layers are the end layers, they are not connected to another n+1 layer. The output of each neuron in a neural network is determined by the weights applied to each interconnection of neurons. The weights determine how the computed results of the nth layer neuron are propagated to the $(n+1)^{th}$ layer neuron. The entire neural network training procedure entails adjusting the weights of the neural networks by comparing the real sample to the computed output [41].

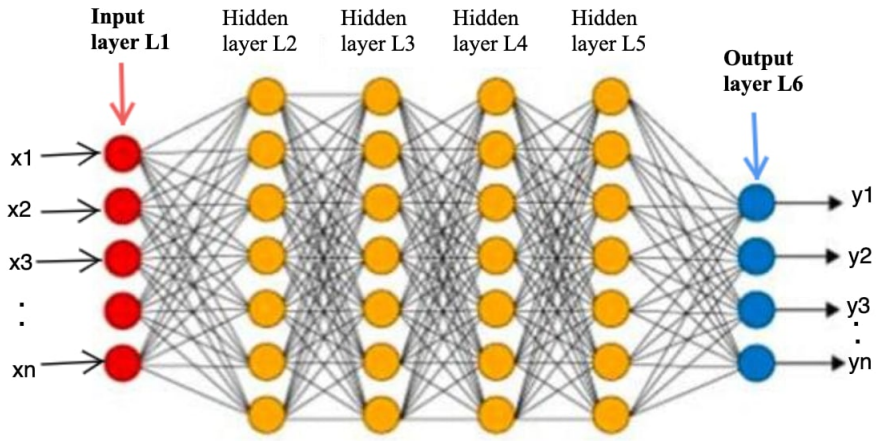


Figure 2.7: Fully connected DNN structure [35].

The input and output of the network are represented by $\{x_1, x_2,...,x_p\}$ and $\{y_0, y_2,...,$and $y_q\}$ respectively. Equation (2.4) below expresses the feature extraction operation of DNN.

$$f(x) = \varphi(w_{ij}^{L} x + b^{L}) \tag{2.4}$$

where:φ is activation function Neural networks must be trained and tested before they can be used to solve issues. Following the completion of the design process, suitable data should be sent to them for training and testing. Sample data, consisting of sample input and output data, will be provided to train neural networks. In addition, the intended level of accuracy that the system must accomplish is specified. After receiving the necessary data, the neural network system adjusts the weight of each interconnection during the training stage by computing the

mean square error of the computed output and the actual output. This technique is repeated iteratively until the system achieves the desired degree of accuracy[41].

Long Short-term Memory Networks (LSTM)

The use of DNN architectures has yielded significant results in a variety of fields, particularly in time-series modeling [44]. This success is largely due to the stacked architecture of these networks, which allows a complex task to be partially addressed in each layer. One variation of DNNs is the Recurrent Neural Networks (RNNs) when unfolded in time. Rather than the hierarchical processing setting observed in deep neural networks, the primary function of the layers of RNNs is to provide some memory.

Long Short-term Memory networks (LSTM) are an extension of recurrent neural networks that are capable of learning long-term dependencies. Small weights are multiplied repeatedly over numerous time steps in a regular RNN, and the gradients drop asymptotically to zero, which is known as the vanishing gradient problem. As a result, LSTM was created to avoid the long-term dependency problem that causes vanishing gradients in traditional RNNs. The LSTM's main innovation is a special unit called the Memory Block or LSTM unit, which has self-connections that store the network's temporal state. As illustrated in Figure 2.8 , LSTM easily addresses vanishing gradient problems by utilizing its four interacting layers within a cell, which varies from ordinary RNN models. By incorporating three gates as shown below, this layer can assist LSTM in learning long-term dependencies and improving the model's performance. LSTM network typically consists of memory blocks, referred to as cells, connected through layers [45]. The information in the cells contained in cell state C_t and hidden state h_t is regulated by mechanisms, and are activated by sigmoid and tanh activation functions. LSTM, therefore, can, conditionally, add or delete information from the cell state. In general, the gates take in, as input, the hidden states from previous time step h_{t-1}, and the current input x_t and multiply them pointwise by weight matrices, W, and a bias b added to the product. Tanh activation is a technique for regulating the values that flow through a network. The tanh function compresses values so that they are always between -1 and 1. Tanh activation is comparable to sigmoid activation. Gates contain sigmoid activation function. Rather than squishing numbers between -1 and 1, it squishes values between 0 and 1, with 0 implying

nothing and 1 implying everything.

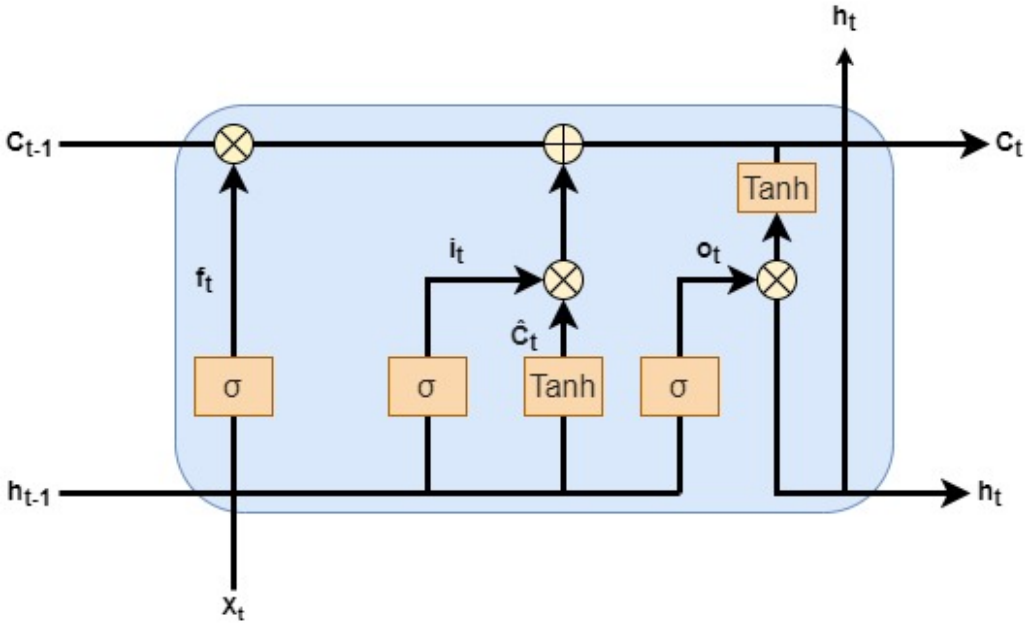


Figure 2.8: Architecture of LSTM models. The LSTM Recurrent Neural Network's architecture is similar to that of an ANN. LSTM hidden units, on the other hand, have connections, which is the RNN's general architecture; however, each hidden unit is responsible for memorizing and forgetting the goal.

An LSTM unit is composed of three gates such as an input gate, an output gate, a forget gate, and a memory cell, which is used to control the flow of information. A memory cell is essentially an accumulator of state information, with numerous self-parameterized controlling gates accessing, writing, and clearing the cell. If the input gate is activated, the data from each new input will be gathered in the cell, and the previous cell state will be forgotten in the process if the forget gate is on and enabled. The final state will propagate to the output gate's final state. A major improvement for LSTM is overcoming the vanishing too quickly from the RNN model [45].

- Input Gate: It determines which of the input values should be used to change the memory. The sigmoid function determines whether to allow 0 or 1 values through. And the tanh function assigns weight to the data provided, determining their importance on a scale of -1 to 1. As shown in equation (2.5).

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ C_t &= \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \end{aligned} \tag{2.5}$$

- The Forget Gate: Takes Previous Long Term Memory (LTM(t-1)) as input and decides on which information should be kept and which to forget.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.6}$$

- Output Gate: The block's input and memory are used to determine the output. The sigmoid function determines whether to allow 0 or 1 values through. And the tanh function determines which values are allowed to pass through 0, 1. And the tanh function assigns weight to the values provided, determining their relevance on a scale of -1 to 1 and multiplying it with the sigmoid output.

$$\begin{aligned} O_t &= \sigma(W_o \cdot [h_t - 1, x_t] + b_o) \\ h_t &= 0_t \star \tanh(C_t) \end{aligned} \tag{2.7}$$

Time-series analysis, natural language processing, and speech processing are just a few examples of where LSTM networks have been used. When hidden layers are stacked to construct deeper recurrent networks, the temporal hierarchy of the sequential data is best retained, much as it is in dense DNNs [46]. Furthermore, if trained on some "clean" data, stacked LSTM networks can be organized to construct an autoencoder that can detect anomalies.

Autoencoders

Autoencoders, which are trained to reconstruct the input at the output by back-propagation, is one of the most desirable types of artificial neural networks for unsupervised learning. They were originally designed for feature extraction and internal representation learning , but they were later expanded to include anomaly detection and non-linear dimensionality reduction. Given some input data, autoencoders are meant to learn a low-dimensional representation. They have two parts: an encoder and a decoder. The encoder learns to map input data to a low-dimensional representation, while the decoder learns to map the low-dimensional representation back to the original input data. By structuring the learning problem in this manner, the encoder network learns an effective "compression" function that translates input data to a prominent lower-dimensional representation, allowing the decoder network to correctly recover the original input data. The reconstruction error, which is the difference (mean squared error) between the

original input and the reconstructed output produced by the decoder, is used to train the model. These architectures have attracted a lot of interest in anomaly detection [47, 48]. They're created to reconstruct the input at the output, with the reconstruction error serving as a metric for detecting anomalous data. Long Short Term Memory (LSTM) networks, which are employed for the encoding and decoding units, are an interesting variation of these architectures. LSTM networks are recurrent neural networks that have demonstrated cutting-edge performance in sequence learning tasks such as anomaly detection [49]. An LSTM-based encoder converts the input sequence into a fixed-length latent vector representation. The decoder, which is another LSTM network, then reconstructs the input sequence using the latent representation. When compared to autoencoders, LSTM-based autoencoders achieve even better outcomes in anomaly detection [50, 51].

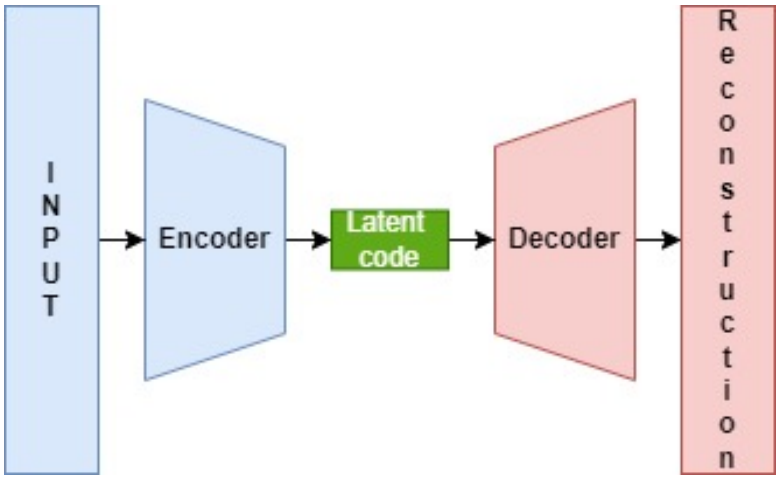


Figure 2.9: Structure of Autoencoder.

Convolutional Neural Network (CNN)

A CNN is feed forward deep neural networks used for processing data with a grid-like topology, including time series data, which can be thought of as a 1D grid, and image data, which can be thought of as a 2D grid [52]. The term "convolutional neural network" signifies that a mathematical operation called convolution is used in the network. As shown in the equation below 2.8, convolution operation involves taking two functions and combining them into a third function that expresses the amount of overlap between one of the functions and a reversed and

translated version of the other.

$$s(t) = (x * w)(t) \quad = \int x(a)w(t-a)da = \sum_{-\infty}^{\infty} x(a)w(t-a) \tag{2.8}$$

When it comes to CNN terminology, the first argument (the function x) to the convolution is often referred to as the input and the second argument (the function w) as the kernel. The output is referred to as a feature map. Convolution is applied to the input data to filter (This filter is also called a kernel, or feature detector) the information and produce a feature map in CNN. The kernel performs matrix multiplication element by element over the input image to perform convolution. In the feature map, the results for each receptive field (the area where convolution occurs) are summarized. Filter sliding continues until the feature map is complete [41]. Figure 2.10 below shows the sliding operation in one dimensional CNN model.

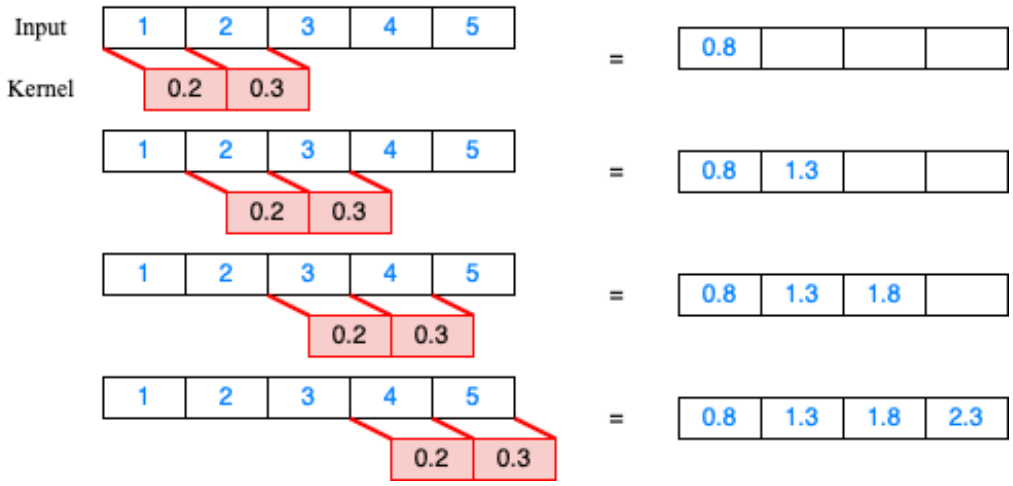


Figure 2.10: Convolution of 1D series with kernel size of (1,2)

The CNN network architecture comprises a convolutional layer, pooling layer, and fully connected layer, as shown in Figure 2.11. Convolutional layers in CNN models capture essential features from input data while pooling layers minimize computing complexity by down sampling the features. A fully connected layer flattens the data, converting it into a single vector that can be used as input data for the next layer. In building a CNN model, there are several hyper parameters to consider, including the number of filters, kernel size, stride size, and padding. Filters extract salient features from data, and kernels specify the width and height of filters. In convolutional algorithms, the stride specifies the amount of movement after each operation, while padding specifies how border problems will be resolved. Convolution is performed only

once for border features without padding [40].

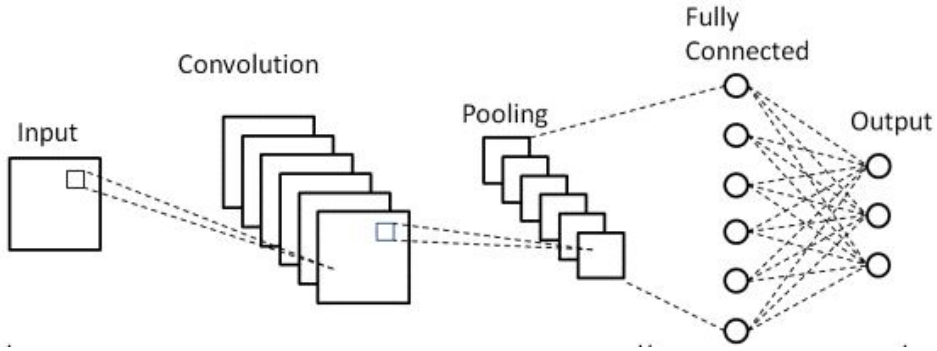


Figure 2.11: CNN Architecture

CNN-LSTM

CNN-LSTM models are combination models that combine CNN and LSTM. Previously, we discussed CNN's capability to automatically learn and extract features from raw sequence data. LSTM network is more efficient at capturing the long-term and short-term effects of temporal features. While CNN models extract salient features from input data sequences, LSTM models connected in tandem interpret and provide output. LSTM models are combination models that combine CNN and LSTM. Figure 2.12 below shows the CNN-LSTM structure.

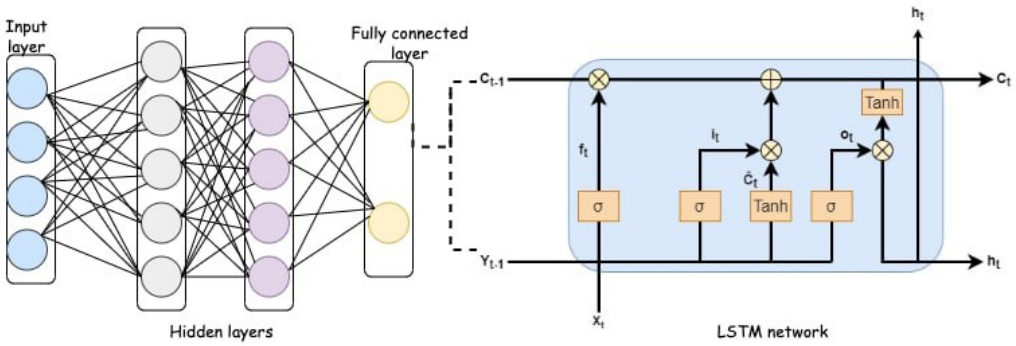


Figure 2.12: CNN-LSTM structure

Convolutional LSTM

LSTM is found to be more efficient to solve the vanishing gradient problem [45]. Incorporating memory units that explicitly allow the network to learn when to "forget" previous hidden states, and when to update hidden states given new information has improved gradient training for LSTM Networks. ConvLSTM combines CNN and LSTM taking the better of the two

worlds to learn the multi-dimensional dependencies in sequential data. In ConvLSTM each LSTM cell has a convolutional operator. The convolutional operation aids in the acquisition of data's spatial features as well as the learning of long-term dependencies, which is an LSTM's distinctive function. Convolution operators have substituted matrix products within LSTM cells in ConvLSTM, allowing the model to easily handle the reading of two-dimensional data like row and column, as well as operate on the spatiotemporal aspect of the dataset by detecting their time dependencies. CNN is a widely known feed-forward network with its capability to extract features from multidimensional input data through a convolution operation of the input with a filter (kernel). Whether it is a 2D layout in the case of an image or a 3D structure

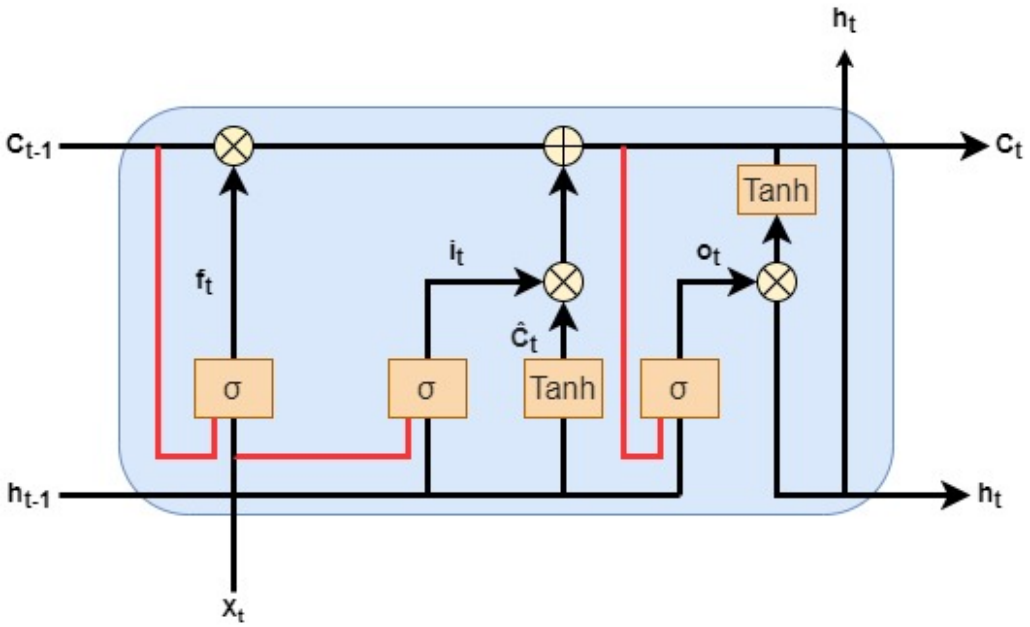


Figure 2.13: ConvLSTM memory cell structure. The red lines show the extra connections discovered in a ConvLSTM cell above an LSTM cell, which arise from the current and previous cell states.

in video frames, its ability to automatically discover relevant contextual and spatial features with a reduced number of parameters makes it more relevant [53]. In ConvLSTM architecture, the inputs to the network are first transformed or convolved with feature extraction parameters (weights) to produce a fixed-length matrix representation. Key equations that define ConvLSTM for a given input tensor x_t are given in Equation (2.9).

$$
\begin{aligned}
i_t &= \sigma(W_x i * X_t + W_h i * H_{t-1} + W_c i \odot \vec{C}_{t-1} + b_i) \\
f_t &= \sigma(W_x f * X_t + W_h f * H_{t-1} + W_c f \odot \vec{C}_{t-1} + b_f) \\
C_t^* &= \tanh(W_h c^* * H_{t-1} + W_x c^* * X_t + b_f) \\
o_t &= \sigma(W_x o * X_t + W_h o * H_{t-1} + W_c o \odot \vec{C}_t + b_o) \\
C_t &= f_t \odot \vec{C}_{t-1} + C^* \\
h_t &= o_t \odot \tanh(\vec{C}_t)
\end{aligned}
\tag{2.9}
$$

Where $\bigodot$ and * denote the Hadamard product and the convolution operator, respectively. it, f_t, o_t, C_t and H_{t-1} are matrices all representing the gating units (input, output, forget), cell unit, and the hidden state, respectively. The weights, W_i, W_f, W_c, W_o, correspond to feature extracting convolutional filter matrices, which are multidimensional arrays in nature, used to transform in the state-to-state and input-to-state convolutional transitions.

Chapter 3

Dataset Description and Preprocessing

In this chapter, the datasets used, the data preprocessing and feature selection steps are briefly discussed.

3.1 Data Collection

The main data source for this thesis is UMTS mobile network data run by an operator. We use UMTS KPI data records as a data source only because of the issue of data availability. However, our work can also be applied to other network technologies. The data is collected and reported to the OSS regularly by a base station (Node-B). To conduct a detailed analysis of cell outage detection, we selected only a few UMTS sites. During the site selection process, we first filter out sites with a zero values for the cell availability KPI. The reason for this is that if the cell availability KPI is zero, the rest of the KPIs will have no reading. Although this might be done on purpose by the network operator for power saving reasons or for any other reasons, we consider sites with zero cell availability to be outages. To make the analysis more efficient, the sites with the fewest missing values are chosen. To conduct this research, two-month KPI data are collected from the UMTS network sites at an hourly granularity, which are 1440 hourly records of each cell for each KPI. The dataset collected consists of around thirty seven (37) KPIs. Since we only need correlated and directly related data to the network degradation problem, we finally selected seven KPIs following some preprocessing. Detailed steps are discussed later in the next sections.

3.2 Data Preprocessing

Data preprocessing greatly affects the performance of a machine learning model [54]. Different activities are performed in this important stage to minimize the effects of noise and to create

clean data for training the model. The activities that are involved in this stage of development concentrate on creating balanced data (in order to avoid skewness), scaling the data down to be contained within a specific range of values, and eliminating irrelevant data (for example, CSV rows that contain null values). This thesis performs the following four types of preprocessing activities before training the model to take advantage of the benefits mentioned above:

3.2.1 Data cleaning

Data cleaning is a procedure that aims to fill the missing values, smooth out noise while identifying outliers, and fix errors in the data [55]. All the collected raw data is not meaningful, instead, it contains several missed and inconsistent values. Thus, data that does not give any relevant information are filtered out. Some parameters are often not logged or missing from actual measurements due to some factors. In our case too, though KPIs are critical to assessing the QoS of a functional wireless network, it is challenging to obtain all measurement parameters with complete details in practice [7]. There are various techniques for avoiding missing values from the final dataset: such as ignoring the tuple value, manually substituting the missing value, or using techniques such as interpolation and replacing the missing value with the constant value using the mean attribute for all samples of the same class as a specific tuple. Since our data is a time series, any missing value disrupts the sequence, so we use interpolation techniques to reduce the impact of missing values. Interpolation is the technique which estimates the value of a function at a point from its values at nearby points. Some of the data cleaning tasks used in this study include eliminating outliers and dealing with missing values. In our work, a linear interpolation technique is applied. This technique is efficient for predicting missing values [56]

3.2.2 Data Normalization

Normalization refers to the process of scaling down the input data to a specific range [54]. The range of the values of the preprocessed data is dependent on the form of the normalization technique applied to the data. The normalization of input data is crucial both to speed up training and testing of the machine learning algorithms as well as to ensure reliable results. There are numerous normalization strategies used in machine learning nowadays Z-score and Min-Max

normalization are often utilized in machine learning approaches. The Z-score standardization uses a standard normal distribution, which sets the mean at zero and scales the data to a standard deviation of 1. This method is useful when the data has a normal distribution (Gaussian distribution). The Min-Max scaler shifts and scales the attributes so that they end up in the range [0, 1] or [-1, 1]. As a result of the influence of outliers while computing the empirical mean and standard deviation, a standardized scaler does not guarantee balanced feature scales when outliers are present. In this thesis, we use Min-Max Normalization, which is formulated using the Equation 3.1.

$$d_n = \frac{d_i - d_{min}}{d_{max} - d_{min}} \tag{3.1}$$

where:

d_n : a normalized version of a datum d_i.

d_i: any datum in a particular column of the dataset.

d_{max} : the maximum value of a particular column of the dataset.

d_{min} : the minimum value of a particular column of the dataset.

Min-max normalization keeps the relations between the original data values unchanged. The cost of having a bounded range is that we will have a further decrease in the standard deviations of the data record, which can further reduce the effect of outliers. Using scikit-learn [57], we use a built-in function to perform min-max normalization before supplying our data to our model by instantiating a normalizer object. After model training, by using an inverse transform of the object instantiated, the normalized records have returned to their original values.

3.2.3 Feature selection and Reduction

Feature selection is a technique for identifying features that provide more information on the tasks at hand and excluding inputs that are unrelated to the outcome (or have a low correlation). It has a significant impact on the deep learning model's success in terms of compressing the amount of data to be used by minimizing or extracting new features, eliminating redundant information, reducing processing time, and minimizing the resources required. If it is not selected wisely, it may diminish the performance of the deep learning model. In this study, correlation-based feature selection is adopted for feature reduction. Correlation is a statistical measurement of the strength of the relationship between two variables. Correlation coefficients

range from -1 (perfectly negative correlation) to +1 (perfectly positive correlation). A zero correlation coefficient indicates no linear relationship between the variables. As explained in the previous section, the dataset collected consists of many KPI values under each KPI categorization. Since using all collected KPIs is not efficient, we use the widely used Pearson linear correlation coefficient [58] to select a better representative KPI from each KPI categorization and lastly for all the selected KPIs. Pearson correlation is defined as which is defined as:

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2(y_i - \overline{y})^2}} \tag{3.2}$$

Where n is the number of samples, and xi and yi are the values of first and second variables for ith sample, respectively.

3.3 Cells KPIs correlation

The thresholds of correlation between KPIs are set at +0.5 for positive correlation and -0.5 for negative correlation. When the calculated correlation coefficient between any two KPIs is above +0.5 or below -0.5, both KPIs are considered tightly correlated. Under Accessibility KPI categorization, the correlation analysis is done for the RAB setup success rate KPI, Call setup success rate, and RRC setup success rate. The following correlation matrix shows the correlation coefficient between KPI's.

(i) The correlation coefficient matrix between RAB setup success rate KPIs.

Figure 3.1a shows the correlation matrix shows that CS RAB setup success rate is uncorrelated with other KPIs while the rest three PS RAB setup success rate, (HSUPA) RAB setup success rate, and High-Speed Downlink Packet Access (HSDPA) RAB setup success rate are highly correlated with almost near to 1 value of correlation coefficient. PS RAB setup success rate is chosen from the three highly correlated KPIs. Therefore CS RAB setup success rate and PS RAB setup success rate are selected as representative KPIs for RAB setup success rate.

(ii) The correlation coefficient matrix between PS RRC Establishment success rate and Cell RRC Attempted Failure Rate:

Figure 3.1b shows the correlation coefficient between PS RRC Establishment success rate

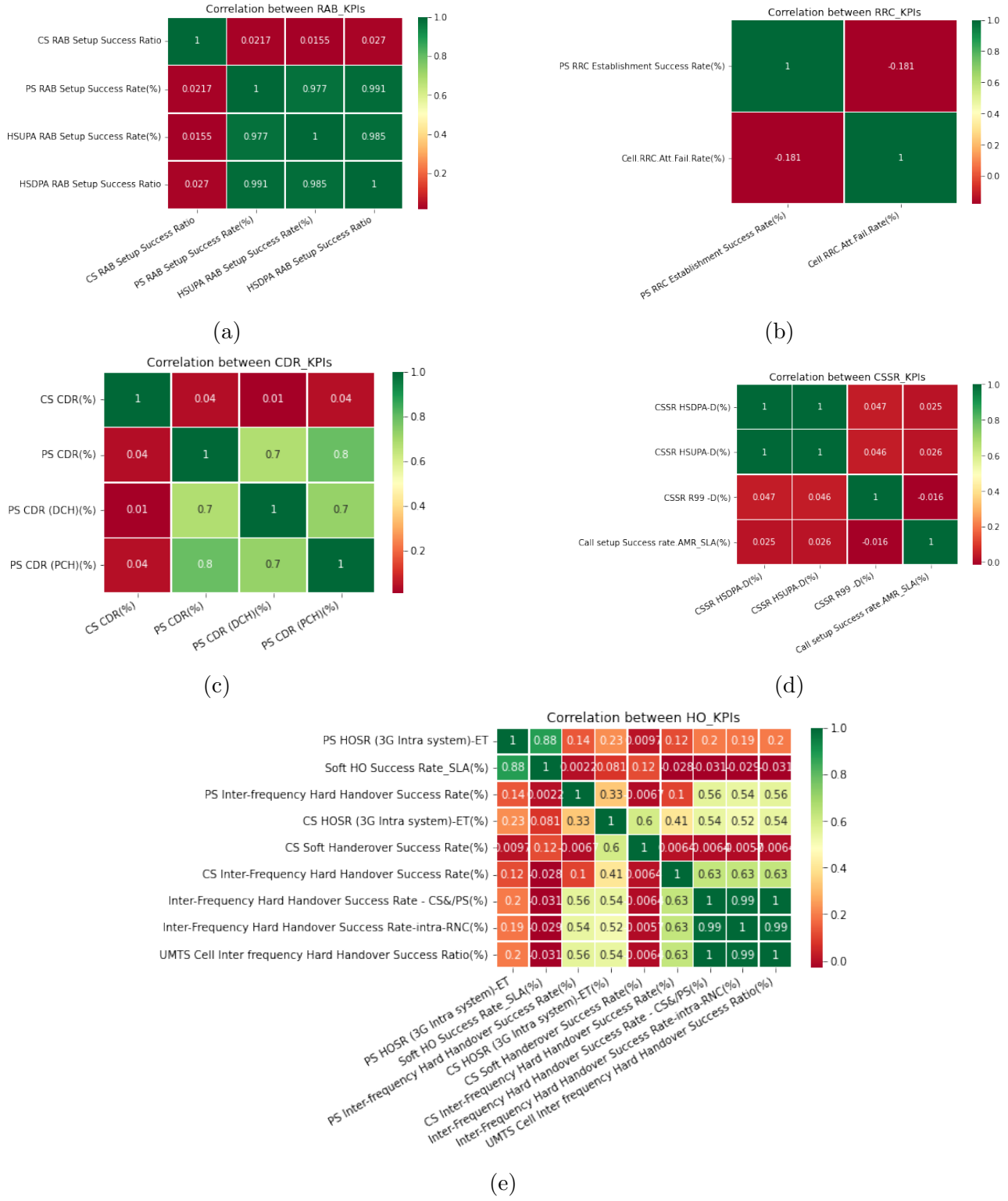


Figure 3.1: Correlation coefficient matrix between between KPIs.

and Cell RRC Attempted Fail Rate. The result showed the two are uncorrelated, so we took both KPIs.

(iii) The correlation coefficient matrix between Call Drop Rate (CDR): Figure 3.1c shows the CDR KPI which is under the Retainablity KPI category. Based on their correlation coefficient Circuit switched and packet switched call drop rates are selected (CS CDR and PS CDR).

(iv) The correlation coefficient matrix between the Call Setup Success Rate (CSSR) KPI: Figure 3.1d shows the correlation coefficient between CSSR. From the correlation matrix, it's shown that CSSR AMR does not correlate with other KPIs but HSDPA, CSSR HSUPA, and CSSR R99 are highly correlated. Therefore we select CSSR AMR and CSSR HSDPA.

(v) The correlation coefficient Matrix between Handover Success Rate KPIs: The same correlation analysis is done for the Mobility KPI category. In this category we got a lot handover related KPIs. The following table shows the correlation coefficient. Figure 3.1e shows the correlation coefficient between the handover KPIs. Based on their correlation coefficient, we select PS HOSR (3G Intra system)-ET, CS HOSR (3G Intra system)-ET, and Inter-Frequency Hard Handover Success Rate (CS &/PS) KPIs.

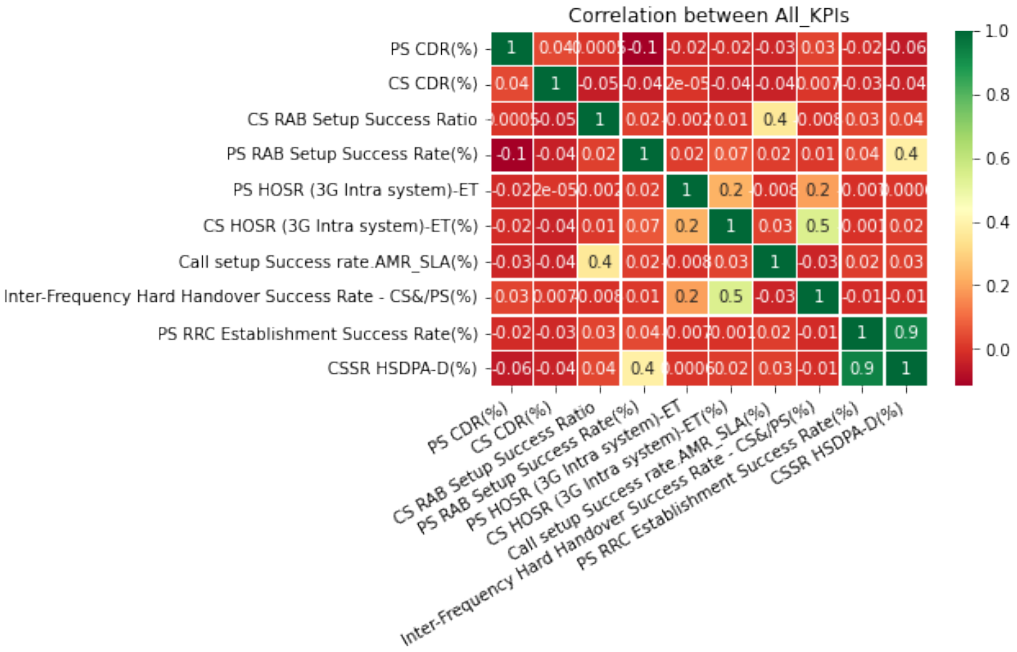


Figure 3.2: The correlation coefficient matrix between all selected KPIs from each category.

From the above correlation analysis, we selected ten KPI's from accessibility, retainablity and

Table 3.1: Selected KPIs.

Measurements	Description
CS CDR	Circuit Switched Call Drop Rate
PS CDR	Packet Switched Call Drop Rate
CSSR	Call Setup Success Rate
PS RAB setup success rate	Packet Switched Radio Access Bearer setup success rate
PS RRC establishment success rate	Packet Switched Radio Resource Control
PS HOSR (3G Intra system)	Packet Switched Handover success rate
CS HOSR (3G Intra system)	Circuit Switched Handover success rate

mobility KPI categories. Still this much KPI is large, so we need further analysis for further reduction. Figure 3.2 shows the correlation between all KPI's from each category. The correlation coefficient matrix shows Inter-Frequency Hard Handover Success Rate - CS and PS is correlated with CS HOSR (3G Intra system)-ET, so we exclude it. CS RAB setup success rate and call setup success rate are correlated, we choose call setup success rate. CSSR HSDPA and PS RRC establishment success rate are highly correlated and PS RRC establishment success rate is selected. From the overall analysis we got seven total features as shown in Table 3.1.

Chapter 4

Experimental Results and Discussion

In this chapter experimental scenarios used to evaluate the proposed approach along with the obtained results are discussed.

The overall system architecture of the proposed approach is summarized and illustrated in Figure 4.1. In the preprocessing sub-block, we fill the missing records by interpolation and normalize the data to a specific range as already discussed in Section 3.2. The next sub-blocks, the forecasting and performance degradation detection (as anomaly detection) task independently uses the preprocessed KPI data and generate forecasts of future KPI records and identify degraded records respectively. Both tasks can be performed independently as needed or the forecasting output can be applied to the anomaly detection model as input and it can perform anomaly detection tasks on the forecasted KPI records when necessary, resulting in the composite concatenated system that will enable us to predict degradation in performance. The forecasting and anomaly detection blocks are discussed in the following section.

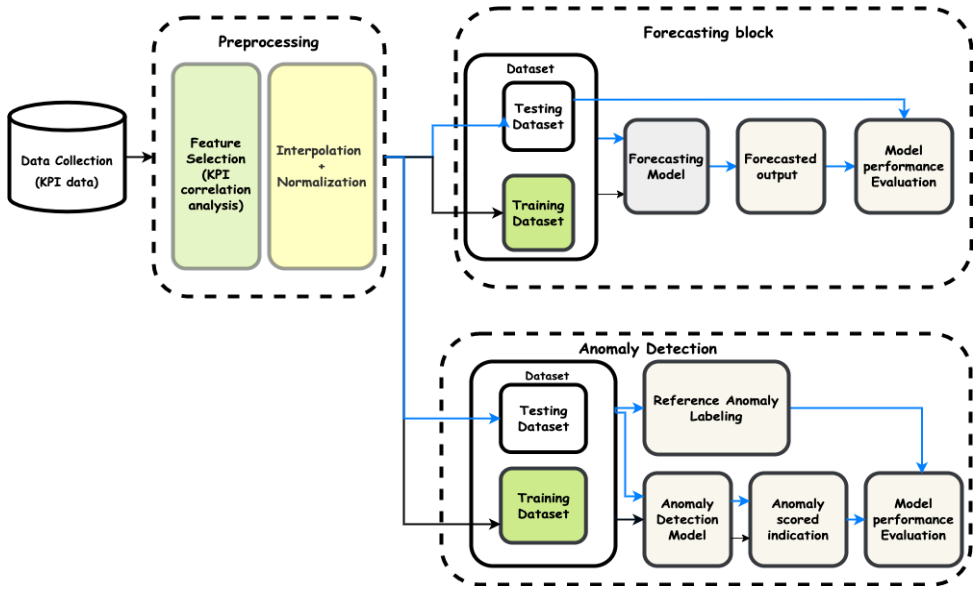


Figure 4.1: System architecture of the proposed approach.

4.1 Forecasting Model

As stated in the introduction part, the main aim of our work is to predict the performance degradation of network cells a head of time. Forecasting the key performance indicators of cells is an important task since the performance of the cells is evaluated by the KPIs. A large amount of KPI data is recorded for each cell at a granularity of one hour, among this KPI some are correlated and some are unrelated. For our work, we choose a few most representative KPIs that would help us to predict the network status. A Multivariate Time Series Forecasting technique is followed to forecast each KPIs by taking the most correlated other KPIs with them. Figure 4.2 below illustrates each of the steps taken to forecast the time series data. The forecasting process begins by choosing correlated KPIs with the target KPIs to be forecasted; filling in missing values; and normalizing the data. The next task is setting the loop back and forecasting range. The model predicts the next six-hour KPI data based on eighteen hour historical dataset. 70% of the data is used for training, while the remaining 30% is used for testing and validation.

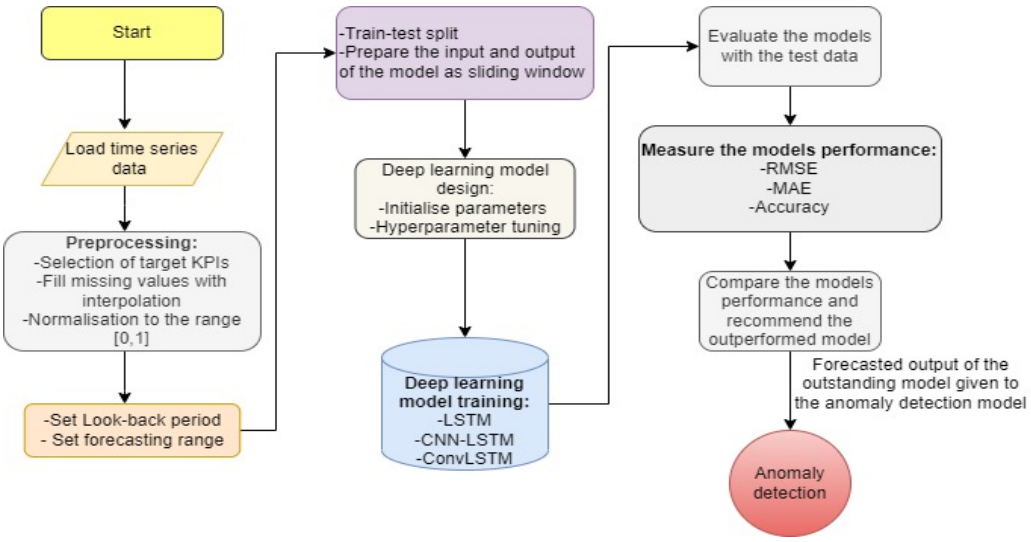


Figure 4.2: Key procedures for the forecasting task.

4.1.1 Multivariate Time Series Forecasting

Multivariate time series refers to a time series that has more than one time dependent variable. That means, unlike univariate forecasting, each variable depends not only on its past values

but also has some dependency on other variables. This dependency of multivariate time series is convenient in modeling interesting inter dependencies and forecasting future values. For forecasting the targeted KPIs, beside the KPI itself, we use the most correlated KPIs we got from the correlation analysis done previously. For each targeted KPIs, correlated KPIs used for forecasting are listed as follows.

1. Circuit Switched (CS) Handover success rate (CS HOSR)
    - CS Soft Handover Success Rate (CS soft HOSR);
    - CS Inter-Frequency Hard HOSR;
    - Inter Frequency Hard Circuit Switched - CS and PS;
2. Packet Switched Call Drop Rate (PS CDR)
    - Packet Switched Call Drop Rate (Dedicated channel) (PS CDR (DCH))
    - Circuit Switched Call Drop Rate (Physical channel) (CS CDR (PCH))
3. Call Setup Success Rate (CSSR)
    - CSSR High-Speed Up-link Packet Access (CSSR HSUPA)
    - CSSR Release 99 (CSSR R99)
4. Packet Switched Random Access Bearer (PS RAB) Setup Success Rate
    - HSUPA RAB Setup Success Rate
    - High-Speed Down-link Packet Access (HSDPA) RAB Setup Success Rate
    - CSSR HSDPA
5. PS Radio Resource Control (RRC) Establishment Success Rate
    - CSSR HSDPA
    - Cell RRC attempted failure rate
6. CS Call Drop Rate (CDR)
    - CSSR HSDPA
    - PS CDR

4.1.2 Model Building

As we already mentioned in the literature review section, different authors discussed that, the deep learning method of forecasting the time series records outperforms the machine learning methods. Taking this into consideration,in this work, we do the performance comparison of

three widely used deep learning methods, LSTM, CNN-LSTM, and ConvLSTM. The internal structure of each used model, as well as their mathematical basis, are discussed in chapter two section 2.4. After the data preprocessing stage is completed, the hyper-parameter of each model is to be tuned to their optimal value as followed in the next section.

Hyper-parameters Tuning

A hyper-parameter is an external variable that controls the learning process. The selection of these hyper-parameters is crucial to learning and predicting the desired task with greater accuracy. Hyper-parameters which affect the training of this model and which can be modified based on the desired task are selected. In this experiment, we explore the different hyper-parameters that are performed on each model to find the optimal hyper-parameters of the models so that this set of hyper-parameters optimize the models performance by minimizing the objective loss function to deliver a better forecast with minimal error.

Table 4.1: LSTM Model candidate hyper-parameters.

Hyperparameters	Value
Number of LSTM cells	[30, 40, 50, 60, 70, 80]
Number of Epoch	[0-100]
Dropout	[0.1, 0.2, 0.3, 0.4]
Learning Rate	[$1e^4$, $1e^3$, $1e^2$]

Table 4.2: CNN-LSTM Model candidate hyper-parameters.

Hyperparameters	**CNN**	**LSTM**
Number of Layers	[2,3,4]	[2,3,4]
Number of kernels	[32,64,128]	–
Number of kernel Size	[3,5,7]	–
Pooling Policy	[Max-Pool, Avg-Pool]	–
Number of LSTM cells	–	[30,50,100]
Number of Steps	–	[$1e^4$, $1e^3$, $1e^2$]
Number of Epoch	[0-100]	

In order to tune hyperparameters, the model is iteratively trained on real data with different combinations of hyperparameters as shown in Table 4.1, 4.2 and 4.3 for all LSTM, CNNLSTM and ConvLSTM models respectively. The best parameters are then selected based on the whole combination of training which gives the least mean average error. After performing the

Table 4.3: convLSTM Model candidate parameters.

Hyperparameters	Value
Number of Layers	[16, 32, 64]
Number of Epoch	[0-100]
Dropout	[0.1, 0.2,0.3,0.4]
Kernel Size	[1,3), (1,5)]
Number of LSTM cells	[30, 50, 80, 100]
Learning Rate	[$1e^4$, $1e^3$, $1e^2$]

tuning process, we summarized the optimal hyper-parameter of the LSTM, CNN-LSTM and ConvLSTM models in Table 4.4, 4.5 and 4.6 respectively.

Table 4.4: LSTM Model tuned parameters.

Hyperparameters	Value
Number of LSTM cells	30
Number of Epoch	49
Dropout	0.3
Learning Rate	0.01

Table 4.5: CNN-LSTM Model tuned parameters.

Hyperparameters	**CNN**	**LSTM**
Number of Layers	2	2
Number of kernels	128	–
Number of kernel Size	3	–
Pooling Policy	Max-Pool	–
Number of LSTM cells	–	50
Number of Steps	–	18

Table 4.6: convLSTM Model tuned parameters.

Hyperparameters	Value
Number of Layers	32
Number of Epoch	87
Dropout	0.3
Kernel Size	(1,5)
Number of LSTM cells	80
Learning Rate	0.0001

Detailed description about each of the methods for performing multivariate time series forecasting is presented in the following section:

I. LSTM Model

Suppose$\{x_t^1, x_t^2, x_t^3, ..., x_t^k\}$ denotes a multivariate time series at the time t where k is the number of variables. An LSTM network is trained using a sequence of data $\{x_1, x_2, ..., x_N\}$, where N is the number of samples. First, the Min-Max Scaler function is used to scale individual observations by the formula:

$$x_{(scaled)}^{(t)} = \frac{x^{(i)} - x_{min}^{(t)}}{x_{max}^{(i)} - x_{min}^{(i)}} \tag{4.1}$$

where $x_{max}^{(i)}$ and $x_{min}^{(i)}$are the maximum and minimum values of x(i) in the data set, respectively. A sliding window of size m, $m < N$, is then employed to train the LSTM. This means m consecutive multi-variate variables are fed into the LSTM at the same time. In order to predict the next value of the characteristic of interest, say $x_*^{(1)}$ we use these $m * k$ inputs.

The following pseudo-code in Algorithm 1 shows the working principle of LSTM model.

Algorithm 1: LSTM based multivariate multistep time series forcasting

Data: Input: A sequence of KPI records $(x_1, x_2, ..., x_N)$,number of epochs E, learning rate λ, batch size B, dropout rate, number of LSTM cells, sliding window size m, number of timesteps to be forcasted n_{out}.

Result: Output: The predicted KPI values $x_{N+n_{out}}^{(1)}$ which is the first n_{out} component of $X_{N+n_{out}}$.

Initialization: parameter of the LSTM model (initial weight and bias).

while $e \leq E$ **do**

  for $i \in (1, ..., B)$ **do**

    $\hat{X}_t^{(1)}, ..., \hat{X}_{t+n_{out}}^{(1)}$. $\leftarrow$*LSTM*$(x_t, ..., x_{t-m+1})$. *Calculate prediction loss,* $L = ||\{\hat{X}_j - X_j\}||$*:*

    Let the predicted error vector e_t *be defined as* $e_j = \hat{X}_j$ - X_j, $j = m+1, ..., N$.

    Optimize the parameter of LSTM based on the loss function L *and* _*back-propagate with learning rate* λ.

  Data: *KPI data to generate the forecasted value* $\hat{X}_{N+n_{out}}$.

  Result: $\hat{X}_{N+n_{out}}^{(1)}$.

At the first window, the sequence $\{x_1, x_2, ..., x_m\}$ from the training data set is input to LSTM and the network can predict the value $\hat{x}_{m+1}^{(1)}$. In the second one, based on the sequence$\{x_2, x_3, ..., x_{m+1}\}$, the LSTM can predict the value $\hat{x}_{m+2}^{(1)}$. This process continues until the windows slide to the end of the training data set. The weights of the LSTM network is trained to minimize the loss function of error prediction:

$$L = \sum_{i=m+1}^{N} (|\hat{x}_i^{(1)} - x_i^{(1)}| \tag{4.2}$$

The performance of the LSTM network is evaluated using the loss metric root mean square error (RSME) and mean absolute error (MAE).

II. CNN-LSTM

CNNs are efficient at extracting spatial features from a large volume of features and selecting the significant ones. CNN is therefore used to extract features (KPIs) that strongly relate to a predictive target. As LSTMs expand with time, they are widely used in time series analysis. A CNN-LSTM-based KPI forecasting model is developed based on the characteristics of CNN and LSTM. After the input data are passed through the convolution layer and pooling layer in the CNN layer, the features of the input data are extracted, and the output data of the CNN layer are calculated through the LSTM layer, and the output value is predicted accordingly.

III. ConvLSTM

The ConvLSTM model combines the advantages of CNN and LSTM for spatiotemporal prediction. While retaining the fundamental structure of LSTMs, it converts all vector products into convolutional operations. With this variation, the model is able to extract spatial features from multiple UMTS KPIs in all time slots efficiently. Here is a brief description of how convLSTM works. First, it take $X(t)$, which is a vector containing all the KPIs from the last k time slots concatenated. The dataset used in our study consists of data aggregated by time slot of one hour. A smaller granularity of time slots would likely lead to better predictions, but would also create more computing and storage load. We set the value of k to 18 in our case because it results in the highest accuracy in prediction. In contrast to LSTM and CNN-LSTM, where we provide a 1D 18 hour time series input, we use a 2D input by distributing the 18 hours onto 3x6 arrays, and instead of getting the temporal relationship between consecutive records, ConvLSTM is able to extract features drawn on both sequential and non-sequential records. Following that, multiple ConvLSTM layers will learn which features contribute most to the target prediction at each time slot from all inputs and also remember the long-term temporal features.

4.1.3 Results

Following the selection of optimal hyper-parameters for each model, we split our data into training, validation, and test sets. In contrast to many deep-learning problems that split training from testing with random selections, time-series data allows training and testing without sacrificing time dependence. Due to the KPI records being time-series data, we used the first 40 days of KPI records as the training set, the next 9 days as the validation set, and the last 9 days as the test set. Then, the training, validation and test sets have been restructured in a way

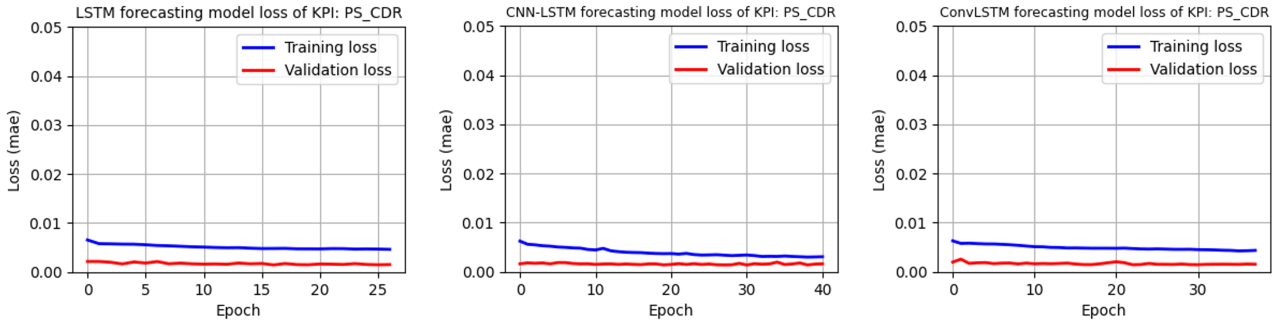
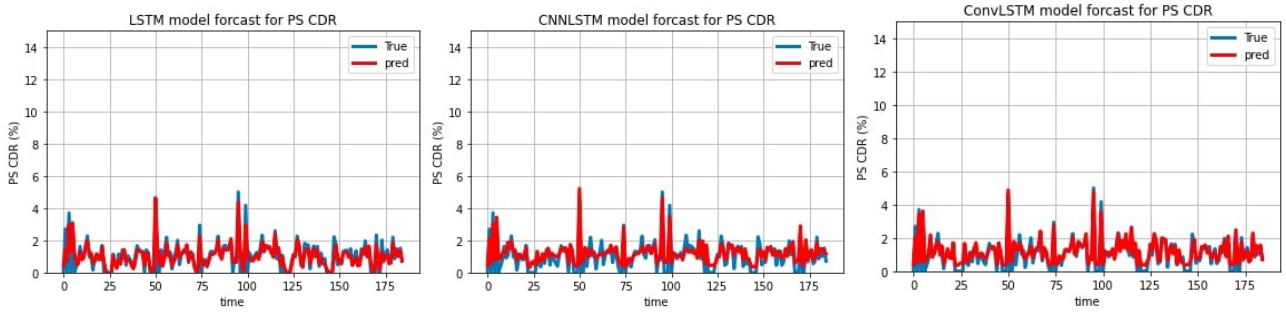


Figure 4.3: Forcasting loss for PS CDR of LSTM, CNN-LSTM and ConvLSTM model.

to accommodate multivariate and multistep forecasting time series to be framed as supervised learning using the sliding window method by taking the first 18 hr record as input and the next 6 hr record as forecast output. After we prepare the data, we fit the training and validation sets of each KPI record to each model, and after a few epochs, the losses on the training and validation sets reduce significantly as can be seen in Figure 4.3 on the PS CDR KPI. In order to avoid repetition in the remaining sections, we will only refer to the PS CDR KPI outputs.



(a) LSTM (b) CNN-LSTM (c) ConvLSTM

Figure 4.4: Actual vs forecasted records of PS CDR for each model.

With regard to each of the seven KPIs, each model is then tested using the test set and the

actual KPI data versus the forecasted KPI data for KPI of PS CDR can be seen in Figure 4.4 illustrating the output of each model.

Based on the obtained results, we observe that the forecasted values are very close to the true values for each of the three models we considered.

Performance Evaluation

In the above assessment, we presented the forecasted vs. actual PS CDR records for each model. In this section, we provide an overall performance evaluation of each model on all KPIs. As part of the performance evaluation metric, RMSE and MAE are used, and Figure 4.5 and 4.6 shows each KPI's RMSE and MAE scores.

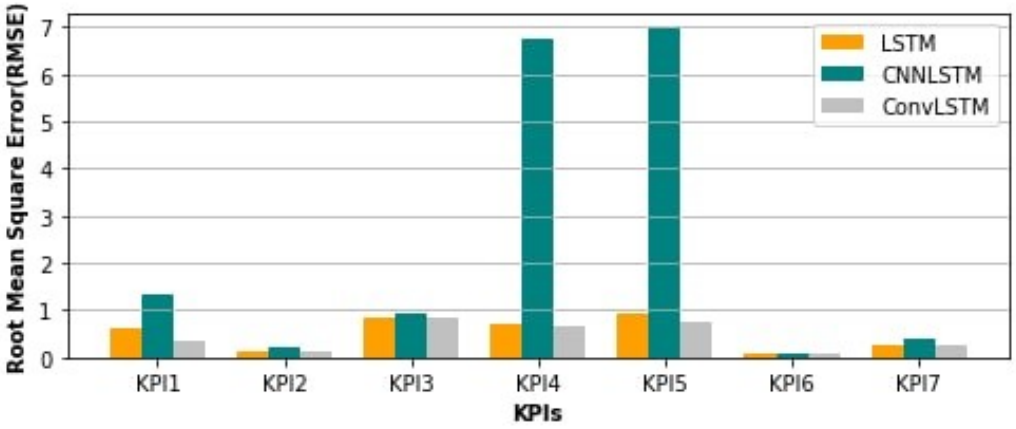


Figure 4.5: Root Mean Squared Error (RMSE) value of models for each KPIs.

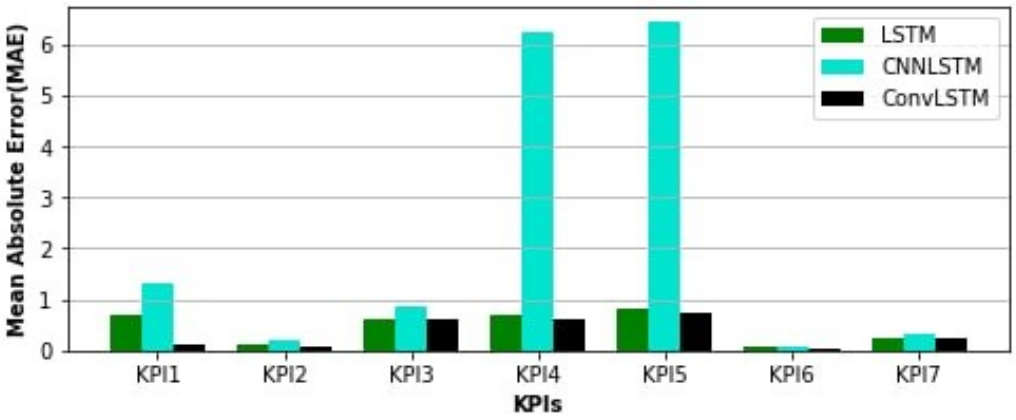


Figure 4.6: Mean Absolute Error (MAE) value of models for each KPIs.

Where: KPI1, KPI2, KPI3, KPI4, KPI5, KPI6, KPI7 are CS HOSR, PS HOSR, PS CDR, Call

setup Success rate, PS RRC Establishment Success Rate, CS CDR, PS RAB Setup Success Rate respectively.

Based on the three deep learning models considered in this study with the evaluation metrics, as it can be seen in Figure 4.5 and further confirmed by the MAE score as in Figure 4.6, the LSTM model has a minimal RMSE error compared to the CNN-LSTM model, which implies LSTM can provide a better forecasting performance on all KPIs. This is because in the CNN-LSTM model, the input time series data first applied to CNN networks and its output is then flattened before it is applied to the LSTM network. Thus, in this case when applying to CNN networks, the spatial information of the data can be captured by the CNN network but the time series nature of the data will be lost, and it will be treated as a series. Eventually, the LSTM network that comes after the CNN network can't perform a good forecasting as the temporal feature variation can't be extracted and leads to the high forecasting error. In contrast, the third model, ConvLSTM, has a smaller RMSE score (less than 1 for all KPIs) and outperforms the two earlier networks. Having this better performance is due to the fact that ConvLSTM uses convolutional networks without sacrificing the time series nature of the data and the system can therefore exploit both spatial and temporal variations of features.

So, in this experiment an independent multivariate time series forecasting model is implemented and tested on all targeted KPIs. As such, when necessary, a forecast of KPI measures can be made ahead of time independently, and the forecast can further be assessed to determine whether performance has been degraded using an anomaly detection model as it briefly described in the following section.

4.2 Anomaly Detection

In this experiment, the cell performance degradation problem is formulated as an anomaly detection problem. So, LSTM Autoencoder model with dynamic thresholding framework is proposed and implemented for anomaly detection. Figure 4.7 shows the steps followed by the anomaly detection model.

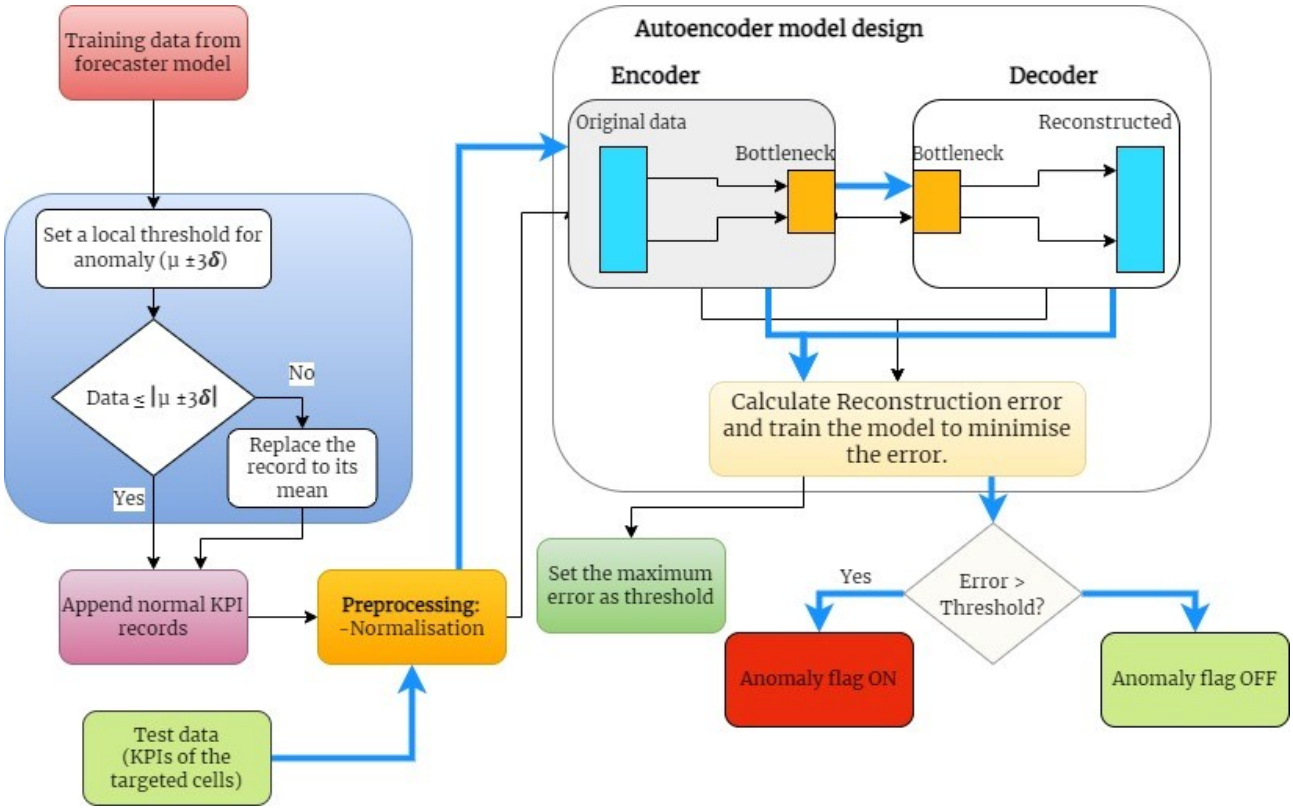


Figure 4.7: Anomaly Detection Model.

4.2.1 Training Dataset Preparation

As the model is supposed to be trained by normal (anomaly-free) data, the training dataset is checked for anomalies. According to [7] a KPI data sample that deviates one standard deviation σ from the average KPI value is 68.2% anomaly, and data sample that deviates $3 * \sigma$ from the average is 99.7% anomaly. In our work KPI data samples that are $3 * \sigma$ away from the average KPIs values is be labeled as anomalies. For labeling the training dataset to normal and anomaly data, we need to set a threshold value for each time series KPIs. But a static threshold can't be used to label time series data. Therefore, for all seven of our selected KPIs, we use dynamic thresholding. We do this by setting up a sliding window and calculating the mean and standard deviation for each data point. After that, we calculate the upper and lower thresholds using $(\mu \pm 3 * \sigma)$. A sliding window of size k is created, which moves one time step at a time. As a result, we calculate the means and standard deviations of each data point in the sliding windows, and then we calculate the lower and upper thresholds accordingly. After threshold settings have been established, the training dataset is labeled by comparing it with the upper and lower threshold. While our model must be trained with normal data, since

this is a time series dataset, we cannot discard anomaly points from our training set. In their place, we substituted the record mean value, and then the training data became normal. After normalizing and cleaning the training dataset, the LSTM-Autoencoder model will be trained on the normalized and cleaned training dataset. Figure 4.8 and 4.9 shows a sample KPI training dataset before and after removal of the anomalies. Each points are compared with the dynamic threshold value and those who are $3 * \sigma$ away from the average KPIs are labeled as anomaly and trimming the data to the mean value and the normal training dataset is found.

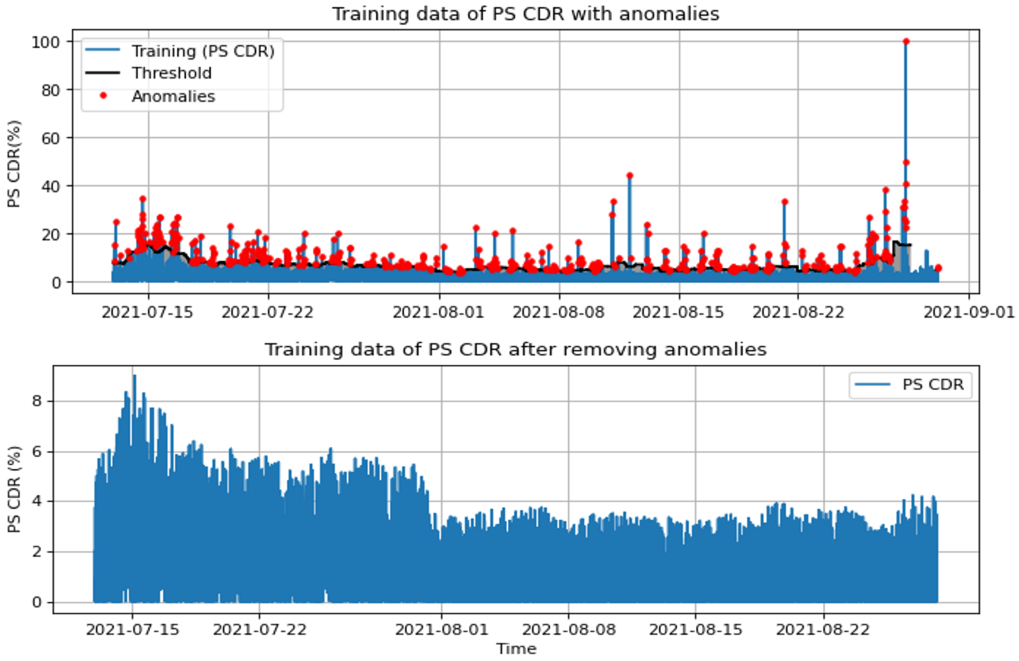


Figure 4.8: Training Dataset of PS CDR before and after anomaly removal.

4.2.2 Model Training

After preparing training dataset, we build the anomaly detection model. For anomaly detection, we use LSTM-Autoencoder model. The working principles and hyperparameter tuning is discussed in the next section.

LSTM-Autoencoder

The Autoencoder is an unsupervised neural network that learns the best encoding-decoding scheme from data. It consists of an input layer, an output layer, an encoder neural network, a decoder neural network, and a latent space. As data is fed into the network, the encoder

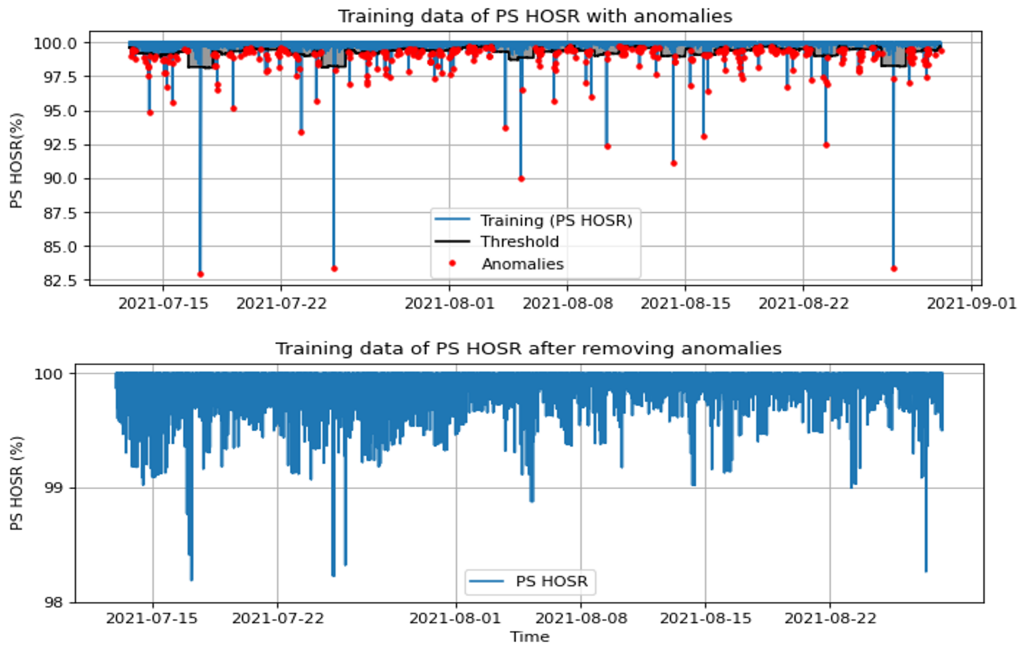


Figure 4.9: Training Dataset of PS HOSR before and after anomaly removal.

compresses the data into the latent space, and the decoder decompresses the encoded representation into the output layer. In order to update the weights of the network, the encoded and decoded output is compared with the initial data, and the error is propagated through the architecture. Specifically, given the input $x \in R^m$, the encoder compresses x to obtain an encoded representation $z = e(x) \in R^n$. After this, the decoder reconstructs the representation to produce the output $\hat{x} = d(z) \in R^m$. The reconstruction error of the autoencoder network is then calculated by taking the difference between the input (x) and output ($\hat{x}$) of the network using MSE metrics as in Equation (4.3). Taking this reconstruction error as an objective function of the model as described in [59], [50], and the model is trained to minimize this objective function using Adam optimizer, which is an adaptive learning rate optimization algorithm.

$$L = \frac{1}{2}\sum_{x} ||x - \hat{x}||^2 \tag{4.3}$$

Autoencoders are not just meant to copy the input to the output. The autoencoder is forced to learn the most salient features by constraining the latent space to have a dimension smaller than the input, i.e. $n < m$. Hence, an important feature in the design of the autoencoder is that it reduces data dimensions while keeping the major information of the structure. There are different types of autoencoders. The LSTM autoencoder makes both the encoder and the

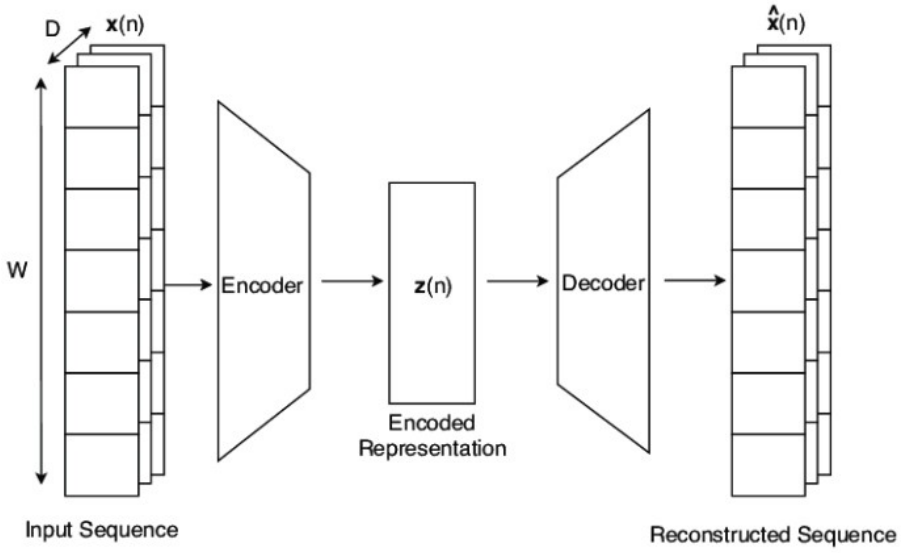


Figure 4.10: An illustration of a LSTM Autoencoder network.

decoder a LSTM network. The ability of LSTMs to learn patterns in long sequences of data makes them suitable for time series forecasting or anomaly detection. LSTM cells capture temporal relationships between variables in multivariate data, i.e., they are used to detect temporal dependencies. The following table shows the results of tuning the hyperparameters for the LSTM autoencoder model to get the best performance

Hyperparameters	Value
Number of Layers	3
Number of Epoch	100
Dropout	0.3
Kernel Size	(1,5)
Number of LSTM cells	Encoder: (32,16,4) Decoder: (4,16,32)
Optimizer	Adaptive moment estimation (ADAM)
Activation function	rectified linear unit (ReLU)
Loss Function	Mean Squared Error Loss
Early Stopping	Patience (10)
Learning Rate	0.0001

Table 4.7: LSTM Autoencoder tuned parameters.

The next step is reshaping the input data into a suitable format to the model. The training and test dataset are normalized with a min-max scaler. The following is the pseudocode of the multivariate time series anomaly detection using LSTM Autoencoder.

Algorithm 2: Anomaly Detection

Data: Input: A sequence of KPI records $(x_1, x_2, ..., x_N)$,number of epochs E, learning rate λ, batch size B, dropout rate, number of LSTM cells.

Result: Output: The classified KPI records as Normal or Anomaly.

Initialization: parameter of the LSTM autoencoder (initial weight and bias).

for $e \in E$ **do**

  for $i \in \{1, ...B\}$ **do**

    $\hat{X}_j = (\hat{x}_t, ..., \hat{x}_{t-m+1}) \leftarrow$LSTM autoencoder$(\mathrm{X}_j = x_t, ..., x_{t-m+1})$, $j = m+1, ..., N$;

    Calculate prediction loss, $L = ||\hat{X}_j - X_j||$;

    Let the predicted error vector e_t be defined as $e_j = \hat{X}_j - X_j, j = m+1, ...N$;

    Optimize the parameter of LSTM autoencoder based on the loss function L and back propagate with learning rate λ;

Predict the training set to the model;

Calculate and capture the maximum reconstruction loss;

Set threshold $T_e = L_max$;

Predict the test data and calculate the prediction loss L_{test};

Compare the loss L_{test} with the threshold T_e;

Label the test data based on;

x_i is anomaly.

Threshold setting

In this process, LSTM autoencoder models are replicated as KPI sizes, with each copy being fitted to a normal (anomaly free) training set for each KPI. As soon as the training is complete, the same training set is predicted for each copy of the model, and we capture the reconstruction error through mean squared error.

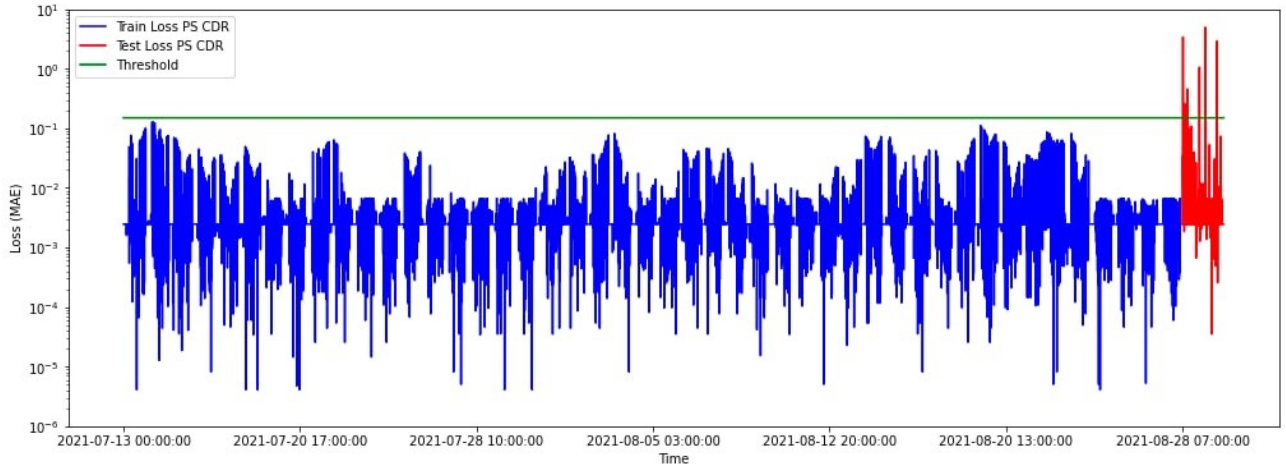


Figure 4.11: Train and Test reconstruction error distribution of PS CDR.

Figure 4.13, illustrates the distribution of KPI losses from the graphs. From these graphs,

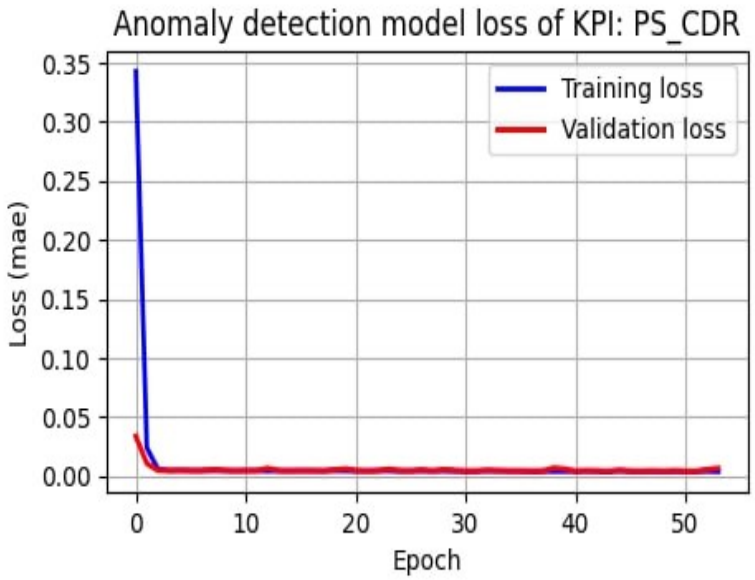


Figure 4.12: Anomaly Detection model loss of PS CDR.

we determine the maximum error for each KPI and use this value as a threshold loss for the comparison with normality. Each model is then fitted with an individual test set of each KPI, then the resulting test loss is compared with the threshold value. In general, the test sets are a combination of anomaly and normal data. Accordingly, when the test set is applied to the trained model, the loss of anomaly records is expected to be greater than the threshold. In Figure 4.11, we show the training and test loss of the Packet Switched Call Drop Rate (PS CDR) KPI. The plot in red represents the error in test data reconstruction (test loss), while the plot in blue represents the error in training. We will be able to identify the anomalous points in the test set by looking at the threshold (shown in the green line). So, those test sets with a loss value that exceeds the threshold loss will be deemed as anomalous, while the rest are deemed normal.

Reference Dataset for performance Evaluation

Since unsupervised learning is the approach we propose, there are no labels needed for the model input. However, once we have trained our model with the normal data and tested it with the test data, a performance evaluation must be performed with reference data. In order to achieve this, we used a linear sliding window for dataset labeling. We made a reference dataset for all seven KPIs. Figure 4.14 below shows a sample Time Series Data Set Labeling for a single cell PS CDR KPI data.

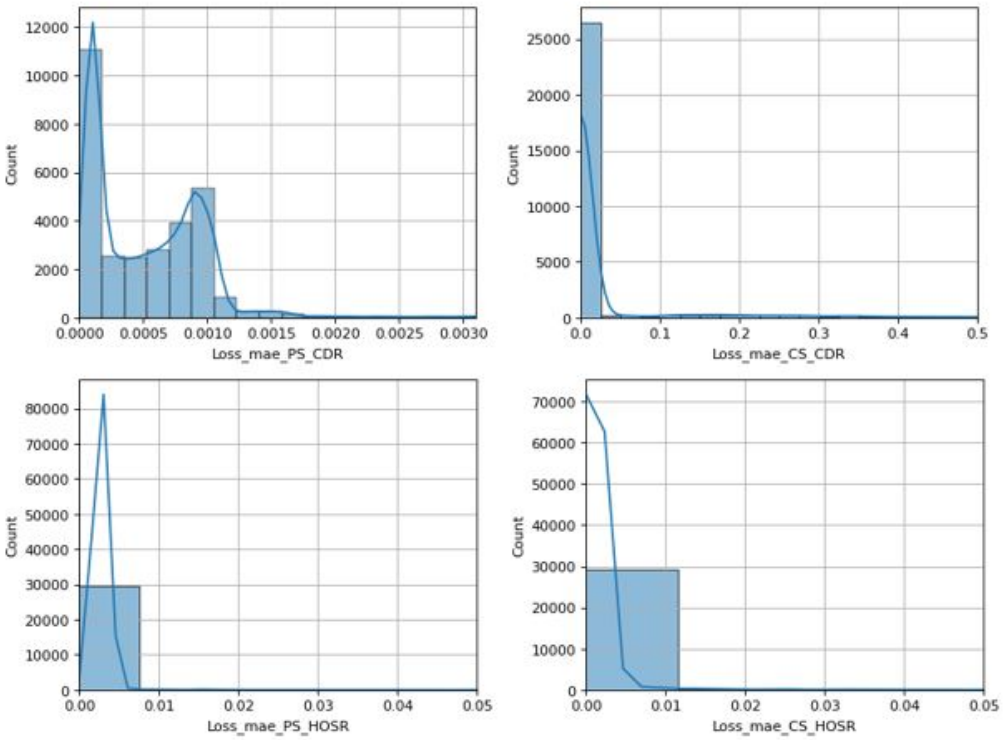


Figure 4.13: KPI's Loss Distribution.

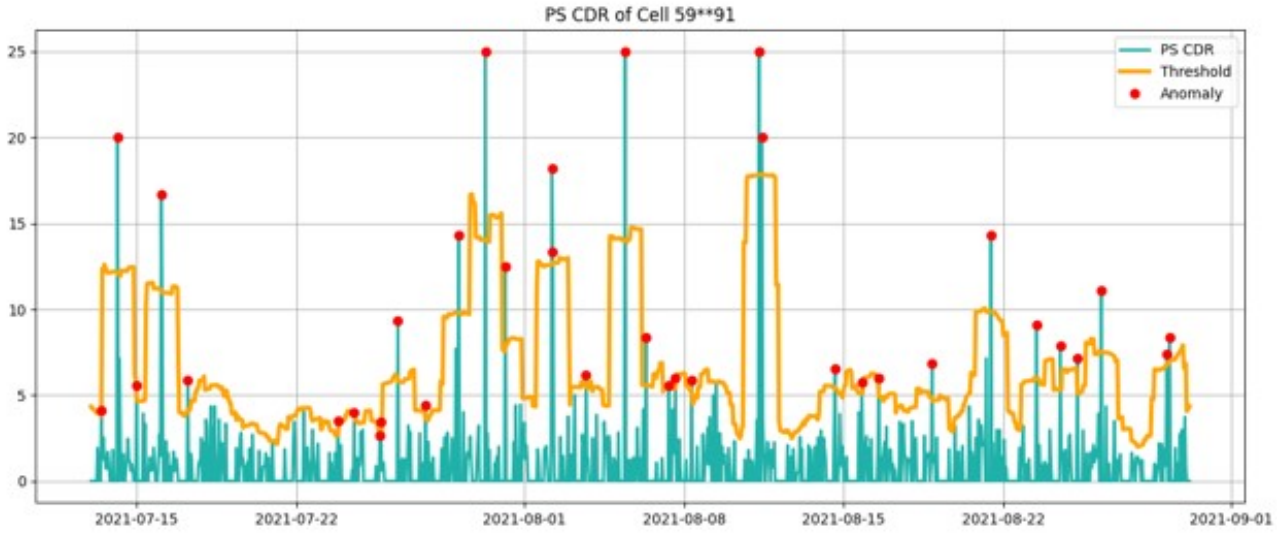


Figure 4.14: Time Series Data Set Labeling of PS CDR.

4.2.3 Results

By training the model with anomaly-free training data, reconstruction loss is computed, and the maximum reconstruction loss is set as our threshold loss value; then, for the test data with

a loss value that is greater than the threshold loss, the data is deemed as anomalous. The same procedure is followed for the all KPIs and for any test set. As a result, we only analyze a single site test set of 12 cells in this work. However, our system is designed to detect anomalies within any input data, regardless of its size.

Our model does the anomaly detection iteratively for all 12 cells for each KPIs by comparing them with the threshold value we set for each KPIs. Figure 4.15 shows the anomaly detected in each KPIs on a single cell. As can be seen in the plot, each KPI value with a anomaly point is shown with a red dot and as such, our model performs the detection task nearly perfectly, and those anomalous points and the date of the anomaly for each cell is then used as an important flag for network operators to react against these network degradation.

Moreover, we carry out a task that counts how many of the seven KPIs are flagged as anomalies simultaneously. Figure 4.16 illustrates the plot of anomalous KPIs count versus time versus cells. As the plot shows, the data recorded on June 18 has the most anomalous KPIs across all 12 cells and it can also be seen that $Cell12$ has more than two anomalous KPIs simultaneously. Accordingly, if at any time a cell has a higher number of anomalous KPIs, the cell needs to give a higher priority to taking measures, since as most KPIs are in an anomalous state, the performance of the cell can suffer and it will help the network operators to take corrective measures based on the priority level assigned to each cell.

Figure 4.17 shows the confusion matrix of the proposed approach for the PS CDR KPI of two cells. For visualization purposes, we selected only two cells while our model can perform on any (n) target cells via loop function. The result shows that for cell one, 1113 are truly classified as normal out of 1121 total values, while 22 are truly detected as anomaly out of 35 anomaly points. Table 4.8. also shows the confusion matrix for the rest KPIs. Due to a lack of reference or true data from the experts for the performance evaluation purpose, we use the reference data that we made using a sliding window technique. The result on a confusion matrix seems low value but it's not because of the model inaccuracy instead the deviation is due to the used reference data. As shown in Figure 4.15, the proposed model is capable of detecting anomalies accurately. Having the true reference data will help improve the performance.

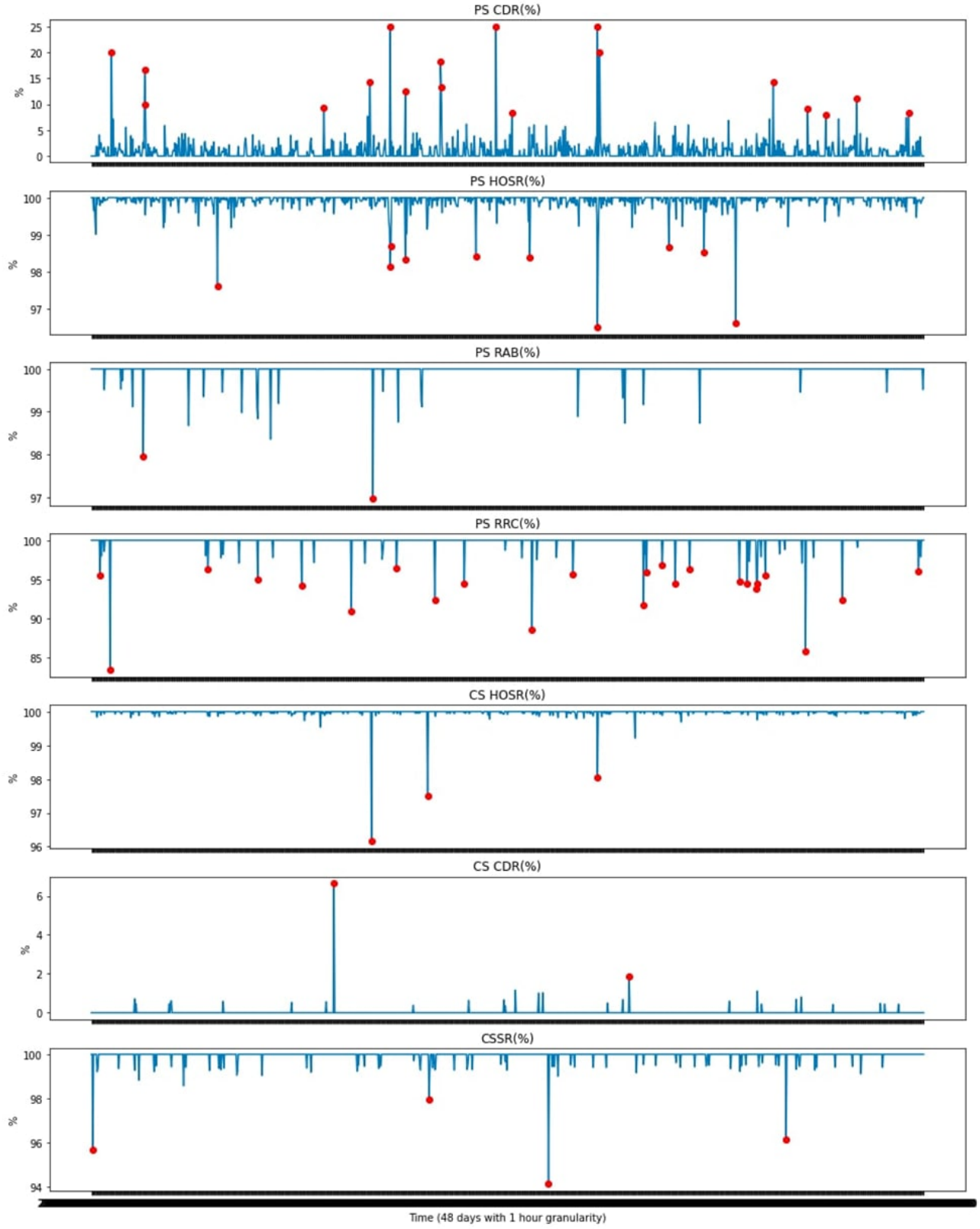


Figure 4.15: Anomaly Detection of a single cell for all KPIs.

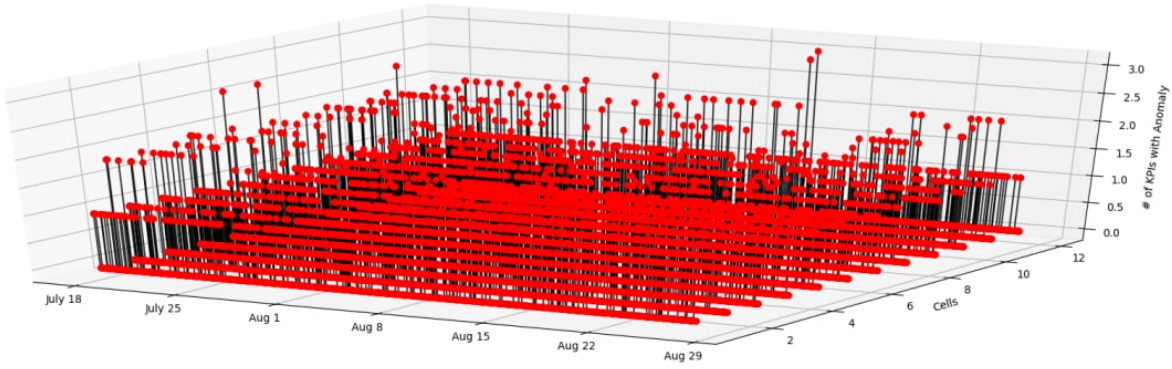
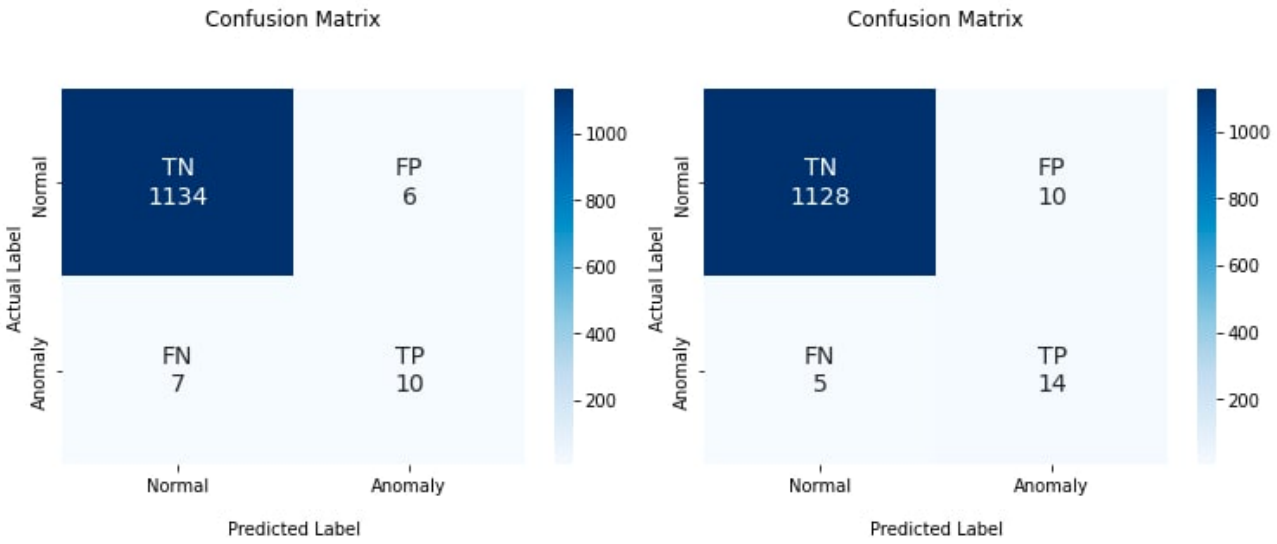


Figure 4.16: Number of Anomaly detected for each cell in a single site.



(a) Cell 1 (b) Cell 2

Figure 4.17: Confusion Matrix for PS CDR Anomaly Detection.

4.3 Performance degradation prediction

The forecasting task and anomaly detection task have been discussed separately up to this point, but we need to look for a composite concatenated model that combines both tasks. The purpose of this section is to test forecasting a single day's KPI information and finding anomalies within the forecasted record. Figure 4.18 below summarizes how the process is carried out. Based on the figure, we used a forecasting procedure for the upcoming KPI for 1 day (24 hours). To do this, we chose the PS CDR KPI as our sample KPI and used our previously trained ConvLSTM model to forecast the last 24 hours' data. The forecasted 24 hour PS CDR records and the actual 24 hour PS CDR records are shown in Figure 4.19. In the next step, the forecasting model's output at point ① is given to the anomaly detection model,

Table 4.8: Confusion Matrix of two cells for all KPIs.

	KPI	TN	FN	TP	FP
Cell 1	KPI1	1134	7	10	6
	KPI2	1117	1	20	19
	KPI3	1127	2	16	12
	KPI4	1126	5	12	14
	KPI5	1121	0	30	6
	KPI6	1131	2	12	12
	KPI7	1114	5	20	18
Cell 2	KPI1	1113	9	22	13
	KPI2	1135	2	11	9
	KPI3	1122	3	15	17
	KPI4	1118	2	23	14
	KPI5	1113	3	31	10
	KPI6	1122	5	17	13
	KPI7	1135	2	13	7

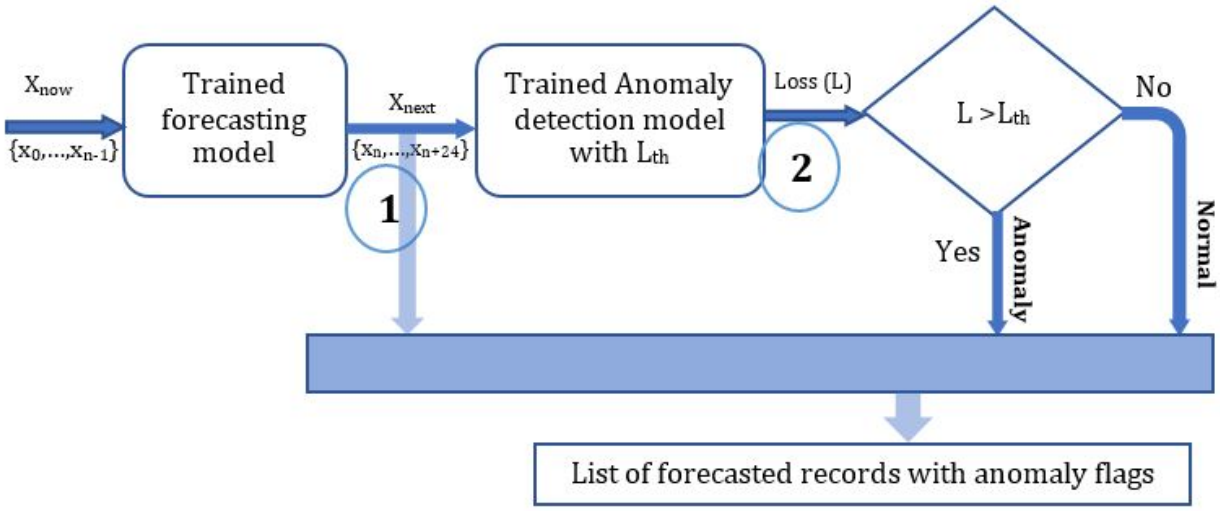


Figure 4.18: Anomaly Prediction architecture.

which then reconstructs the given input and generates reconstruction loss at point ②. As a result of this stage, the generated loss is compared to the threshold loss that has already been set at the training stage, and the forecasted data is classified as normal or abnormal based on the threshold value. An illustration of the final labeled output is provided in Figure 4.20.

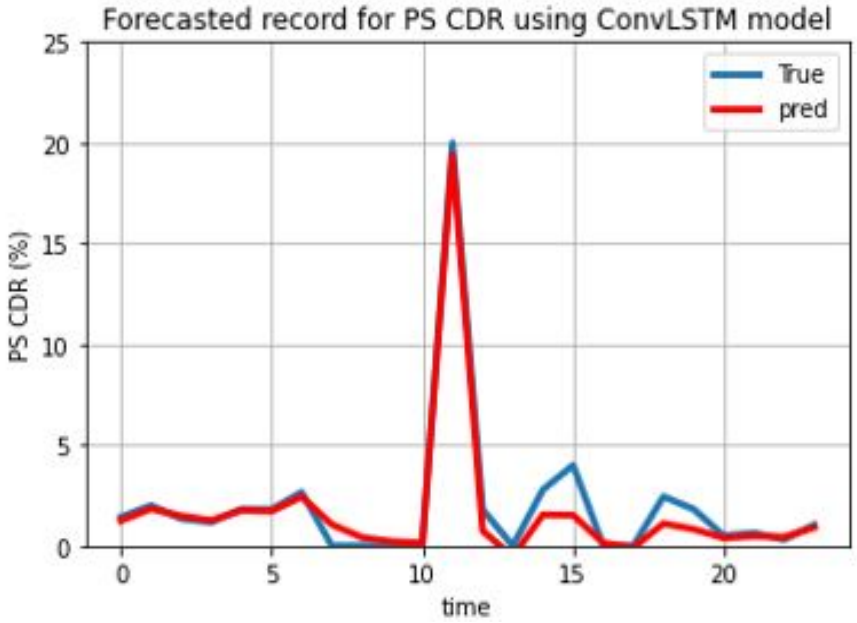


Figure 4.19: Forecasted record of PS CDR

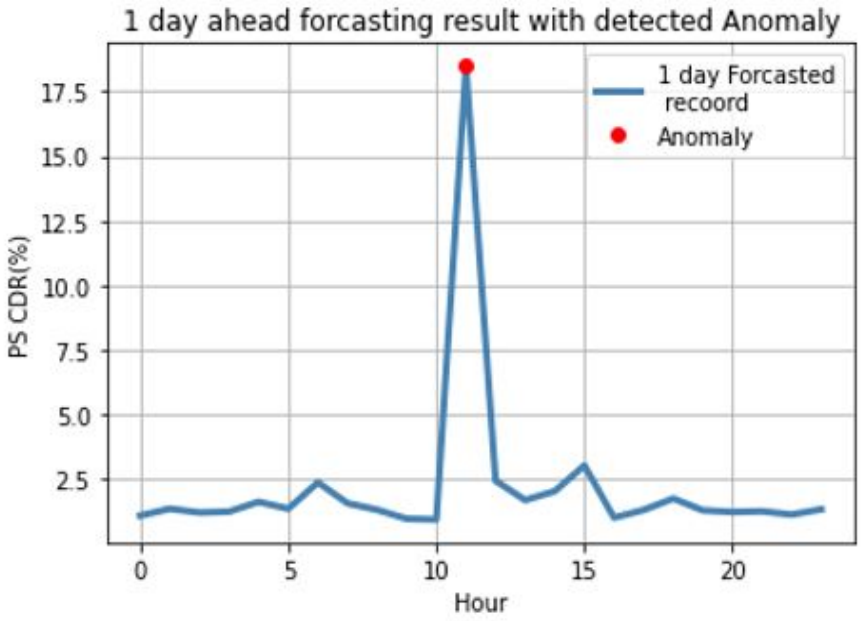


Figure 4.20: Detected Anomalies on the forecasted PS CDR data.

Chapter 5

Conclusions and Future work

This chapter summarizes the results and conclusions of the thesis work. Additionally, future work recommendations are discussed.

5.1 Conclusions

In this work, we propose a deep learning based cell performance degradation prediction using UMTS network KPIs data record as a source data. To achieve this, two different models, multivariate multi-step time series forecasting model and cell performance degradation detection model were designed. LSTM, CNN-LSTM, and ConvLSTM deep learning techniques are used for time series data forecasting. The performance evaluation based on RMSE and MAE shows the ConvLSTM based multivariate forecasting performance has smaller prediction error, hence it outperforms the other two models and this is due to the fact that ConvLSTM uses convolutional networks without sacrificing the time series nature of the data and the system can therefore exploit both spatial and temporal variations of features. LSTM Autoencoder is used for the Cell performance degradation detection, that detects degradation by determining whether the reconstruction error exceeds a certain threshold or not. Forecasting and performance degradation models can both be used independently. Furthermore, by concatenating the forecasting and anomaly detection models, we are able to detect performance degradation in advance. This work can make a significant contribution on improving the performance of network failure management systems to anticipate the upcoming performance degradation of cells that have a vast impact on the network service before they occur and give possibility to improve the service quality and customer satisfaction.

5.2 Future Work

The results of our study show that deep learning algorithms shows a promising result in performance degradation prediction. We believe further investigating additional techniques and additional data could further help to improve the performance of prediction scheme. Taking this into consideration, the following research works are planned to be conducted in the future:

- We suggest investigating different approach to come up with a more robust performance degradation prediction model by applying advanced techniques.
- We recommend investigating the root cause analysis for the fault, degradation and cell outage.

References

[1] A. Zoha, A. Imran, A. Abu-Dayya, and A. Saeed, "A machine learning framework for detection of sleeping cells in lte network," in *Proceedings of the Machine Learning and Data Analysis Symposium*, 2014.

[2] P. Szilágyi and S. Nováczki, "An automatic detection and diagnosis framework for mobile communication systems," *IEEE transactions on Network and Service Management*, vol. 9, no. 2, pp. 184–197, 2012.

[3] A. Zoha, A. Saeed, A. Imran, M. A. Imran, and A. Abu-Dayya, "Data-driven analytics for automated cell outage detection in self-organizing networks," in *2015 11th International Conference on the Design of Reliable Communication Networks (DRCN)*. IEEE, 2015, pp. 203–210.

[4] T. Tsvetkov, S. Nováczki, H. Sanneck, and G. Carle, "A configuration management assessment method for son verification," in *2014 11th International Symposium on Wireless Communications Systems (ISWCS)*. IEEE, 2014, pp. 380–384.

[5] T. Zhang, K. Zhu, and D. Niyato, "A generative adversarial learning-based approach for cell outage detection in self-organizing cellular networks," *IEEE Wireless Communications Letters*, vol. 9, no. 2, pp. 171–174, 2019.

[6] A. Asghar, H. Farooq, and A. Imran, "Self-healing in emerging cellular networks: Review, challenges, and research directions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1682–1709, 2018.

[7] D. Kakadia, J. Yang, and A. Gilgur, "Network performance and fault analytics for lte wireless service providers," in *Network Performance and Fault Analytics for LTE Wireless Service Providers*. Springer, 2017, pp. 1–14.

[8] C. M. Mueller, M. Kaschub, C. Blankenhorn, and S. Wanke, "A cell outage detection algorithm using neighbor cell list reports," in *International Workshop on Self-Organizing Systems*. Springer, 2008, pp. 218–229.

[9] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *ieee Computational intelligenCe magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[10] B. S. Shawel, D. H. Woldegebreal, and S. Pollin, "Convolutional lstm-based long-term spectrum prediction for dynamic spectrum access," in *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.

[11] D. Hailemariam, "Spatiotemporal mobile data traffic prediction using convolutional long short-term memory: The case of addis ababa, ethiopia," Ph.D. dissertation, Addis Ababa University, 2019.

[12] J. Patterson and A. Gibson, *Deep learning: A practitioner's approach.* " O'Reilly Media, Inc.", 2017.

[13] W. Feng, Y. Teng, Y. Man, and M. Song, "Cell outage detection based on improved bp neural network in lte system," 2015.

[14] O. Onireti, A. Zoha, J. Moysen, A. Imran, L. Giupponi, M. A. Imran, and A. Abu-Dayya, "A cell outage management framework for dense heterogeneous networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 4, pp. 2097–2113, 2015.

[15] S. Bekele, "Cell outage detection through density-based local outlier data mining approach: In case of ethio telecom umts network," *no. November*, 2018.

[16] T. Mezgebo, "Anomaly detection of lte cells using knn algorithms: The case of addis ababa," *no. December*, 2019.

[17] Q. Liao, M. Wiczanowski, and S. Stańczak, "Toward cell outage detection with composite hypothesis testing," in *2012 IEEE International Conference on Communications (ICC)*. IEEE, 2012, pp. 4883–4887.

[18] A. Zoha, A. Saeed, A. Imran, M. A. Imran, and A. Abu-Dayya, "A learning-based approach for autonomous outage detection and coverage optimization," *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 3, pp. 439–450, 2016.

[19] I. de-la Bandera, R. Barco, P. Munoz, and I. Serrano, "Cell outage detection based on handover statistics," *IEEE Communications Letters*, vol. 19, no. 7, pp. 1189–1192, 2015.

[20] M. Z. Asghar, R. Fehlmann, and T. Ristaniemi, "Correlation-based cell degradation detection for operational fault detection in cellular wireless base-stations," in *International Conference on Mobile Networks and Management*. Springer, 2013, pp. 83–93.

[21] D. Mulvey, C. H. Foh, M. A. Imran, and R. Tafazolli, "Cell coverage degradation detection using deep learning techniques," in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2018, pp. 441–447.

[22] U. Masood, A. Asghar, A. Imran, and A. N. Mian, "Deep learning based detection of sleeping cells in next generation cellular networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 206–212.

[23] B. Hussain, Q. Du, and P. Ren, "Semi-supervised learning based big data-driven anomaly detection in mobile wireless networks," *China Communications*, vol. 15, no. 4, pp. 41–57, 2018.

[24] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, "Deep learning for smart manufacturing: Methods and applications," *Journal of manufacturing systems*, vol. 48, pp. 144–156, 2018.

[25] G. A. Barreto, J. C. M. Mota, L. G. M. Souza, R. A. Frota, and L. Aguayo, "Condition monitoring of 3g cellular networks through competitive neural models," *IEEE Transactions on Neural Networks*, vol. 16, no. 5, pp. 1064–1075, 2005.

[26] G. Ciocarlie, U. Lindqvist, K. Nitz, S. Nováczki, and H. Sanneck, "On the feasibility of deploying cell anomaly detection in operational cellular networks," in *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE, 2014, pp. 1–6.

[27] G. F. Ciocarlie, U. Lindqvist, S. Nováczki, and H. Sanneck, "Detecting anomalies in cellular networks using an ensemble method," in *Proceedings of the 9th international conference on network and service management (CNSM 2013)*. IEEE, 2013, pp. 171–174.

[28] S. Novaczki and P. Szilagyi, "Radio channel degradation detection and diagnosis based on statistical analysis," in *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*. IEEE, 2011, pp. 1–2.

[29] J. Sanchez and M. Thioune, *UMTS*. John Wiley & Sons, 2013.

[30] P. A. Ochang and P. J. Irving, "Evolutionary analysis of gsm, umts and lte mobile network architectures," *World Scientific News*, no. 54, pp. 27–39, 2016.

[31] H. Holma and A. Toskala, *WCDMA for UMTS: Radio access for third generation mobile communications*. john wiley & sons, 2005.

[32] T. ETSI, "132 450 v9. 1.0 (2010-07), digital cellular telecommunications system (phase 2+); umts; network architecture (3gpp ts 23.002 version 3.6.0 release 1999), 2002," *Search in*, 2002.

[33] H. Holma and A. Toskala, *WCDMA for umts: hspa evolution and lte*. john Wiley & sons, 2007.

[34] M. Bauer, P. Schefczik, M. Soellner, and W. Speltacker, "Evolution of the utran architecture," 2003.

[35] K. Kawamura, M. Murata, S. Higuchi, and F. Kurokochi, "Management system for mobile networks," *FUJITSU Sci. Tech. J*, vol. 48, no. 1, pp. 47–53, 2012.

[36] T. Anttalainen, *Introduction to telecommunications network engineering*. Artech House, 2003.

[37] H. Kaaranen, A. Ahtiainen, L. Laitinen, S. Naghian, and V. Niemi, *UMTS networks: architecture, mobility and services*. John Wiley & Sons, 2005.

[38] T. ETSI, "132 450 v8. 0.0 (2009-08), umts; lte; telecommunication management; key performance indicators (kpi) for e-utran): Definitions, european telecommunication standards institute, 2009," *Search in*, 2010.

[39] R. Kreher, *UMTS performance measurement: a practical guide to KPIs for the UTRAN environment*. John Wiley & Sons, 2006.

[40] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[41] C. C. Aggarwal *et al.*, "Neural networks and deep learning," *Springer*, vol. 10, pp. 978–3, 2018.

[42] F. Chollet, *Deep learning with Python*. Simon and Schuster, 2021.

[43] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[44] M. Längkvist, L. Karlsson, and A. Loutfi, "A review of unsupervised feature learning and deep learning for time-series modeling," *Pattern Recognition Letters*, vol. 42, pp. 11–24, 2014.

[45] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[46] M. Hermans and B. Schrauwen, "Training and analysing deep recurrent neural networks," *Advances in neural information processing systems*, vol. 26, 2013.

[47] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 665–674.

[48] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau, "Autoencoder-based network anomaly detection," in *2018 Wireless Telecommunications Symposium (WTS)*. IEEE, 2018, pp. 1–5.

[49] P. Malhotra, L. Vig, G. Shroff, P. Agarwal *et al.*, "Long short term memory networks for anomaly detection in time series," in *Proceedings*, vol. 89, 2015, pp. 89–94.

[50] H. Nguyen, K. P. Tran, S. Thomassey, and M. Hamad, "Forecasting and anomaly detection approaches using lstm and lstm autoencoder techniques with the applications in supply chain management," *International Journal of Information Management*, vol. 57, p. 102282, 2021.

[51] M. Said Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Network anomaly detection using lstm based autoencoder," in *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, 2020, pp. 37–45.

[52] M. V. Valueva, N. Nagornov, P. A. Lyakhov, G. V. Valuev, and N. I. Chervyakov, "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation," *Mathematics and Computers in Simulation*, vol. 177, pp. 232–243, 2020.

[53] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," *Advances in neural information processing systems*, vol. 28, 2015.

[54] D. Sharma and P. Chandra, "Efficient fault prediction using exploratory and causal techniques," in *2018 Second World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*. IEEE, 2018, pp. 193–197.

[55] I. H. Witten, E. Frank, M. A. Hall, C. J. Pal, and M. DATA, "Practical machine learning tools and techniques," in *Data Mining*, vol. 2, no. 4, 2005.

[56] A. Gnauck, "Interpolation and approximation of water quality time series and process identification," *Analytical and bioanalytical chemistry*, vol. 380, no. 3, pp. 484–492, 2004.

[57] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[58] P. Schober, C. Boer, and L. A. Schwarte, "Correlation coefficients: appropriate use and interpretation," *Anesthesia & Analgesia*, vol. 126, no. 5, pp. 1763–1768, 2018.

[59] B. Sharma, P. Pokharel, and B. Joshi, "User behavior analytics for anomaly detection using lstm autoencoder-insider threat detection," in *Proceedings of the 11th International Conference on Advances in Information Technology*, 2020, pp. 1–9.