ARTIFICIAL INTELLIGENCE-BASED ANOMALY DETECTION FOR THE EFFICIENT MANAGEMENT AND SECURITY OF THE FUTURE CELLULAR NETWORKS

BILAL HUSSAIN

PhD

The Hong Kong Polytechnic University

This programme is jointly offered by The Hong Kong Polytechnic University and Xi'an Jiaotong University

2021

The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

Xi'an Jiaotong University

Department of Information and Communications Engineering

Artificial Intelligence-Based Anomaly Detection for the Efficient Management and Security of the Future Cellular Networks

Bilal Hussain

A thesis submitted in partial fulfillment of the requirements for

the degree of Doctor of Philosophy

May 2021

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or writ-ten, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

_____(Signed)

BILAL HUSSAIN (Name of student)

To my parents, for their unwavering love and support;

To my wife, for her steadfastness and encouragement through the toughest of times;

To my son, RAYYAN HUSSAIN, for bringing comfort and joy in our lives during this tiring journey.

Abstract

Existing communication systems are getting more intricate and a morass to manage which is galvanized by ever-increasing network capacity demand, device density, and the data traffic. Network operators in the United States alone deplete over USD 15 billion yearly to handle cellular outages, incurring escalated operational expenditure (OPEX) as a result. Furthermore, congestion in a cell degrades subscriber quality-of-experience and quality-of-service which results in an increased churn rate and subsequently reduced operator's revenue. Besides the management aspect, the security of the cellular network is paramount to prevent cyber-attacks against its infrastructure not just for its primary subscribers but also for the fact that the networks can be exploited as a proxy to attack the connected cyber-physical systems (CPSs). To unlock the full extent of 6G networks, artificial intelligence (AI)-empowerment has paramount potential for efficient network management and preventing cyber-attacks against their infrastructure.

Based on this backdrop, this thesis applies AI techniques to achieve a primary objective of an efficient, scalable, and timely detection of outages and the situation leading towards congestion in a cell under the context of cellular network management. It also aims to achieve a secondary objective of detecting various cyber-attacks towards the availability of cellular network services in the context of cyber-security of cellular infrastructure and CPSs. Cell outages, situation leading towards congestion in a cell, and cyber-attacks are treated as anomalies in

this work and various machine learning (subset of AI) models and data analytic tools are utilized to detect them by leveraging real subscriber data based on call detail records (CDRs) extracted from a 4G LTE-A network.

List of Publications

Journals

1. **B. Hussain**, Q. Du, and J. Zhang, "A Prescriptive Analytics-Based Modular Framework for Proactive Cell Outage and Congestion Detection in Mobile Networks," *IEEE Transactions on Network and Service Management* (Under-review).

2. **B. Hussain**, Q. Du, B. Sun, and Z. Han, "Deep Learning-Based DDoS-Attack Detection for Cyber-Physical System over 5G network," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 860-870, Feb. 2021.

3. **B. Hussain**, Q. Du, A. Imran, and M. A. Imran, "Artificial Intelligence-powered Mobile Edge Computing-based Anomaly Detection in Cellular Networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 4986-4996, Aug. 2020.

4. **B. Hussain**, Q. Du, S. Zhang, A. Imran, and M. A. Imran, "Mobile Edge Computing-Based Data-Driven Deep Learning Framework for Anomaly Detection," *IEEE Access*, vol. 7, pp. 137656-137667, Sept. 2019.

5. **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, Apr. 2018.

Conference Proceedings

1. **B. Hussain**, Q. Du, and P. Ren, "Deep Learning-Based Big Data-Assisted Anomaly Detection in Cellular Networks," *2018 IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, United Arab Emirates, Dec. 2018, pp. 1-6.

2. **B. Hussain**, Q. Du, and P. Ren, "Big data-driven anomaly detection in cellular networks," *2017 IEEE/CIC International Conference on Communications in China (ICCC)*, Qingdao, P. R. China, Oct. 2017, pp. 1-6.

Acknowledgments

All the praises to Almighty Allah who blessed me with this opportunity and enabled me to complete this thesis. All respects to the Holy Prophet Muhammad (May Allah grant peace and honor to him and his family).

Taking this opportunity, I would like to express my gratitude to all the people who have supported, encouraged and guided me through out the course of my PhD studies. I am particularly thankful to my supervisors Prof. Qinghe Du from the Xi'an Jiaotong University and Dr. Jun Zhang from the Hong Kong Polytechnic University for their exceptional guidance, training, and support in terms of extra funding. I am also thankful to Prof. Muhammad Imran from the University of Glasgow, UK and Dr. Ali Imran from the University of Oklahoma, USA for their support and guidance throughout my journey. I feel greatly indebted to the Xi'an Jiaotong University and the Hong Kong Polytechnic University for their generous financial support.

Finally, I would like to thank my family for always standing by me through the hardest of times.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Anomalies in Cellular Networks

6G cellular system is anticipated to realize advanced Internet of everything (IoE) services applied for extended reality, flying vehicles, haptics, and various other application domains [4]. Such services will enable communication between billions of machines and humans. A study by Cisco [5] forecasts the global IP networks-connected device population to exceed thrice the global human population by 2023 with 14.7 billion machine-to-machine (M2M) connections—50% of the global connected devices and connections. This imply a boost in network density and consequently, an upsurge in node failures leading towards outages [6]. An outage occurs due to the malfunctioned soft or physical elements of the network entities and depending on the severity, a base station either halts services (full outage) or performs sub-optimally (partial outage).

In addition to outages, congestion can transpire at an abruptly high traffic scenario (such as, ongoing match in a stadium, congregation, road traffic congestion, etc.) when the allocable resources are inadequate to cater the demand and if necessary efforts are delayed [7, 8]. Such necessary efforts may involve: earmarking extra resources to the ROI [7], unloading traffic to the neighboring

cells [9], and dynamic billing in QoS-enabled cellular networks [10]. This leads to a poor network performance reflected by hefty connection breakdowns and timeouts, and affecting a multitude of customers, which may increase churn rate and hamper the revenue.

Besides the outages and congestion, cyber-security in future networks is of prime importance due to the fact that cyber-physical systems (CPSs)—employed in the vertical industries and critical infrastructures—will depend on the cellular infrastructure for their functionality. A CPS is a large, complex, and networked mixture of sensors, actuators, and computing nodes that monitor and control physical processes [11, 12]. Due to its highly intricate and heterogeneous nature, contributed by both cyber and physical aspects, it has many general and application-specific vulnerabilities that can be exploited by an attacker to perform mischievous acts [11, Sec. IV, V]. Once compromised, cellular networks can be exploited as powerful attack vectors against CPSs. CPS innovations applied in the vertical industries will potentially account for more than USD 82 trillion in economic activity by 2025 and sabotaging a CPS will equate to a significant bump on an economy [13]. Hence, strong measures should be taken such that the cellular network cannot be exploited as proxy to attack CPSs.

Anomaly is an eccentric demeanor and is defined contextually [14]. For example, to execute network optimization, Wang *et al.* [15] specified city scene (highway, tourist area, railway station, etc.) as an anomaly having a peculiar key performance indicators (KPIs) and traits. Karatepe *et al.* [16] identified location-based anomalies pertaining to the users (commuting from one city to another) who have an abnormal movement behavior, to ameliorate system's consistency. A substantial deviation in user quality-of-experience (QoE) from the expected QoE is described as an anomaly in [17], and it is used for network optimization.

In the backdrop of cellular network management, cell outages and congestion are treated as anomalies in this work because they have an abnormal imprint

on subscriber-level activities or traffic of a base station—abnormally high user activity corresponds to a scenario leading towards congestion while an unusually low user activity reflects an outage [3]. Similarly, in cyber-security's context, we consider various attacks (discussed thoroughly in Section 2.2) against cellular networks as anomalies because of the similar reason that they reflect peculiarly on the traffic of a base station.

1.2 Artificial Intelligence and Data Analytics for Anomaly Detection

Due to the proliferating success and transformational effect of artificial intelligence (AI) techniques in various fields, AI has recently gained momentum and popularity among wireless communication researchers from academia and industry alike. Clamor regarding AI-utilization in cellular networks is evident as various standardization bodies, MNOs, and network vendors have initiated efforts and established alliances towards leveraging the technology [18]. Since cellular networks are evolving into extremely complex and heterogeneous systems, AI is one of the sought after technologies for smooth management of network operations, optimization, and resource orchestration to execute 6G-enabled services [19]. 5G networks adopted a conservative approach towards incorporating AI-technologies for anomaly detection [20]; however, AI for anomaly detection is anticipated to play a substantial part in the 6G (or later stages of 5G) networks as AI has more room to grow in the 6G/post-5G era [21].

Big Data [22] (explained in detail in Ch. 2) is similar for AI algorithms as the fuel for a combustion engine. They are spawned at various levels of a mobile communication system including the cell, core network (CN), and subscriber levels (depicted in Figure 1.1). Big Data analytics through sophisticated machine

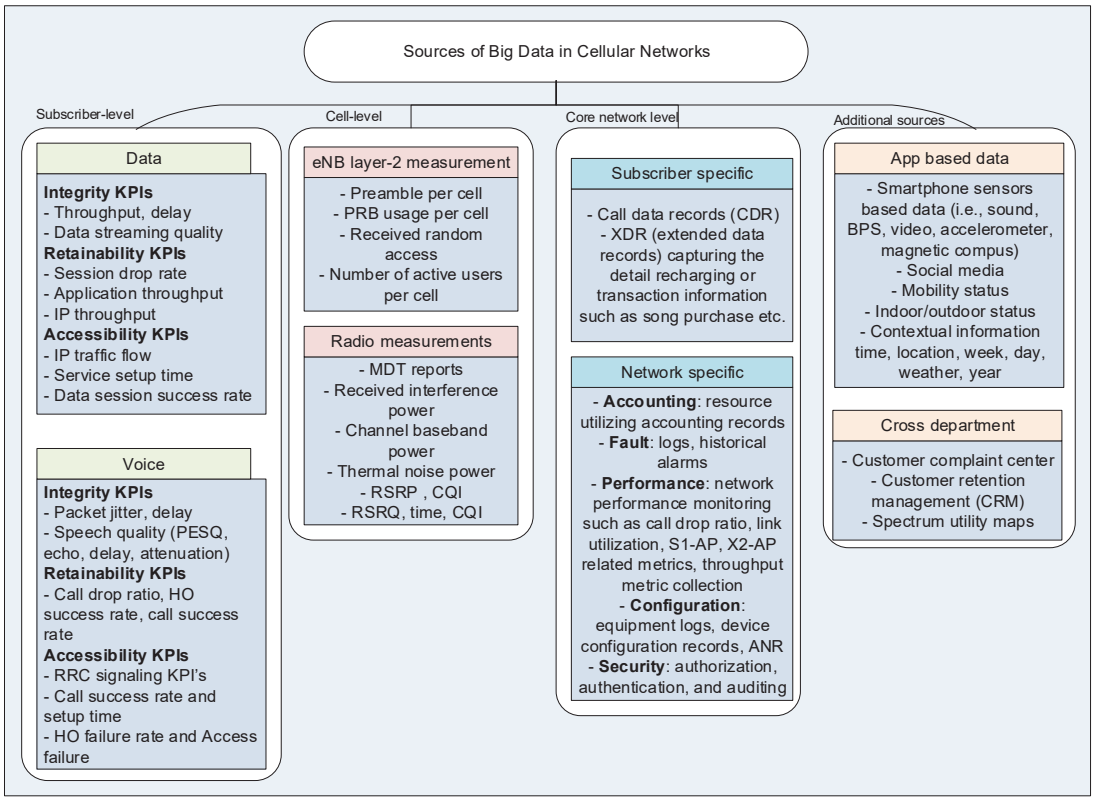

Figure 1.1: Sources of Big Data in Cellular networks [1, Fig. 3].

learning (ML) techniques is projected to be the core breakthrough and an essential component of 6G system which will enable smart network management and optimization [23]. We can exploit the mammoth (big) data for various analytics: Predictive, diagnostic, and prescriptive [24].

1. Predictive analytics assist in forecasting future events like traffic demand, content popularity, resource handiness, etc. by utilizing historical data.

2. Diagnostic analytics empower the network to intelligently identify network outages and performance degradations, determine the root causes, and compensate the affected areas until their restoration—these actions evolve into the functionalities of a self-healing network [8].

3. Prescriptive analytics capitalize on predictions to recommend decisions for solving various problems, e.g., based on the traffic predictions we can design

a more robust and proactive congestion detection system to pre-allocate resources for a region of interest (ROI) where congestion is expected [25].

In this thesis, we present applications of various AI-based techniques for anomaly detection using different data analytic approaches.

1.3 Motivation and the Rising Demand for Anomaly Detection in Communication Networks

According to industry appraisals, cellular networks deplete almost quarter of the total revenue on network operations; out of which, a large chunk is exhausted on the management of cell outages [6]. Hence, the operational expenditures (OPEX) of the mobile network operator (MNO) intensifies. Besides the need to reduce OPEX in cellular networks, industrial Internet of things (IIoT) networks also crave to reduce OPEX as IIoT asset management (identification, troubleshooting, and fixing field-reported problems) account for up to 33 % of the OPEX [26]. Additionally, cellular traffic congestion management is a key challenge that requires meticulous orchestration of network resources. For an effective congestion-control system, precise congestion detection is needed. It is crucial for the QoS-enabled networks offering high QoS-assured services to the consumers [10]. Outages as well as congestion can negatively impact QoE that may in turn raise the churn rate; as irritated customers might be inclined towards switching the mobile operator instead of contacting the customer support center [3].

Towards the security-side, by 2025, CPS advancements pertaining to vertical markets (transportation, energy, manufacturing, eHealth, etc.) are projected to add over USD $82 trillion in the economic growth [13]. Researchers have excogitated 5G technology to be disruptive, playing a critical role in supporting CPSs

with communications and to empower new applications and services in the verticals where it is employed; [27, Fig. 4] demonstrates 5G system's architecture integrated with the vertical industry. As a consequence, vertical infrastructure will have increased dependence on the cellular infrastructure [28], inferred from the white papers [29] presented by 5G infrastructure public private partnership (5G PPP). Industrial applications necessitate the communications infrastructure to support the following rigid demands: low-latency, densely connected devices, ultra-reliability, etc., for which 5G is foreseen to be a suitable candidate [27, 30]. Indeed, services enabling mission-critical and real-time applications will be supported by 5G, such as, smart vehicles' assisted overtaking [28], smart grid's state estimation [31], etc. If CPS is sabotaged, it will significantly hamper the economic growth; therefore, its and the cellular network's security has a paramount importance.

Early detection of anomalies could help reduce OPEX and ultimately improve user QoE and network's quality of service (QoS) because of the lesser network downtime and timely allocation of required additional resources in the ROIs. In the cybersecurity's perspective, timely detection of anomalies could prevent a major dent on critical infrastructure and a compromise on national security. Due to the above-mentioned reasons, popularity of anomaly detection frameworks among the industry and academia is rising and have also motivated this research.

1.4 Scope and Objectives of the Study

Anomaly detection has a broad application domain in which each definition of the anomaly has its own unique flavor depending on the problem in hand [14]. Hence, we will be limiting the scope of our study to mainly outages and congestion as anomalies that lie under the domain of wireless network management. It will be the primary objective to detect them in an efficient, scalable, and timely

manner. Additionally, we will also apply techniques implemented for detecting the above-mentioned anomalies for detecting various cyber-attacks against the cellular infrastructure which belong to the cyber-security domain—this makes our secondary objective.

1.5 Outline of the Thesis

The thesis has been classified into eight chapters. Chapters 3-6 are dedicated to achieve the primary objective (mentioned in the Section 1.4) while Chapter 7 is gearing towards achieving the secondary objective.

In **Chapter 2**, preliminary topics are presented before delving into the proposed frameworks in later chapters. The topics mainly include the explanation about various anomalies considered in this work; a brief description of big data, data science and machine learning; and the elaboration of datasets utilized for anomaly detection followed by the details and visualization of the CDR datasets.

In **Chapter 3**, a semi-supervised machine learning method is presented to extract actionable knowledge out of the dark data (i.e. CDRs) by analyzing the spatiotemporal information on hourly basis. The method is essentially model-based, which assumes the data is distributed according to a Gaussian distribution and that the normal data instances exist in the high probability area while abnormal data instances lie in the low probability area of a statistical model. It generates a normal profile based on the observed pattern of the given data by fitting the model to the given data and then evaluates unseen test data instances with respect to how well they fit the model. Instances which differ significantly from the normal profile are marked as anomalies. The method achieved an overall detection accuracy of about 92%, about 2% higher than the reported accuracy in the state-of-the-art literature.

In **Chapter 4**, the time resolution is shorten from one hour to 10 minutes

for faster detection results and a feed-forward deep neural network (DNN)-based anomaly detector is applied to improve the detection accuracy. Since feature engineering is not necessary for deep learning models, Internet activity is integrated in this chapter which was ignored in the previous one. The chapter takes advantage of Mobile edge computing (MEC) mechanism in which heavy computational works are split among edge servers (ESs) spread across the network. The servers are co-located with the base stations and oversee a small group of base stations. For performance enhancement, the chapter also presents experimentations on various weight initialization, activation functions, regularization, and optimization techniques. The framework achieved 98.8% detection accuracy—a notable improvement that surmount the deficiency of the past studies.

In **Chapter 5**, to expand the computation from detecting anomalies for one base station at a time to 100 or more and to fully integrate MEC paradigm, a novel framework is proposed. It preprocesses raw CDRs to create an image-like volume which is then fed to a deep convolutional neural network (CNN) model. After the training phase, the framework outputs a multilabeled vector identifying anomalous cell(s) in the unseen test (image) data. It is fully compatible with MEC because it is suffice for an edge server to train just one model for the whole population of base stations that are being monitored instead of training a different model for each base station (done in the previous chapter). The chapter presents two CNN models, both having their own pros and cons: the over-the-shelf residual network model with 50 layers and a customized simple CNN model with 5 layers. At the end, the chapter presents some additional experimentation results demonstrating the scalability of our proposed framework. The results manifest the solution can detect anomalies for 100 cells at a time with up to 96% overall test accuracy and with about 7 times lesser training time as compared with the DNN-based model applied for just a single cell anomaly detection.

In **Chapter 6**, prescriptive analytics is performed to achieve proactive anomaly

detection. It proposes a modular framework that first employs a convolutional long short-term memory (ConvLSTM) neural network to forecast the subscriber traffic in a base station 3-hour in advance and then a feed-forward DNN to detect anomalies in the predicted data to alert the MNO. The chapter demonstrates a practical approach on how to integrate both neural networks to achieve the proactiveness. It also presents extensive results: First, preliminary predictions of the forecaster (ConvLSTM neural network) are presented for the chosen three cell IDs which highlights the good, mild, and worst cases. Second, overall prediction results are shown by considering randomly chosen 100 cell IDs with the aim to show how the ConvLSTM performs on the available (Milan) dataset. Third, efforts have been made to elaborate the ways from which the prediction accuracy can be improved and the corresponding results are presented. The results demonstrate the effectiveness of utilizing the proactive approach as it yields over 92% anomaly detection accuracy when the overall number of anomalies in the training set is under 5%.

In **Chapter 7**, an application of Chapter 5's methodology in the cybersecurity domain is presented with a setting having CPSs connected to the cellular network. The chapter proposes a deep CNN-based consolidated framework to provide an early detection of various network availability-targeted attacks that execute a collective distributed denial-of-service (DDoS) attack, orchestrated by numerous malicious devices and controlled by a botmaster. These jeopardized devices separately execute signaling, silent call, SMS spamming, or a blend of these attacks aiming Internet, call, SMS, or a mixture of these services, respectively, that can fracture the connected CPSs' functions. The results demonstrate that the framework can achieve higher than 91% normal and underattack cell detection accuracy.

In **Chapter 8**, concluding remarks and future insights are presented.

Figure 1.2 illustrates progression of our work through the chapters in this

thesis towards achieving the primary and secondary objectives. It also highlights machine learning techniques employed in each chapter.

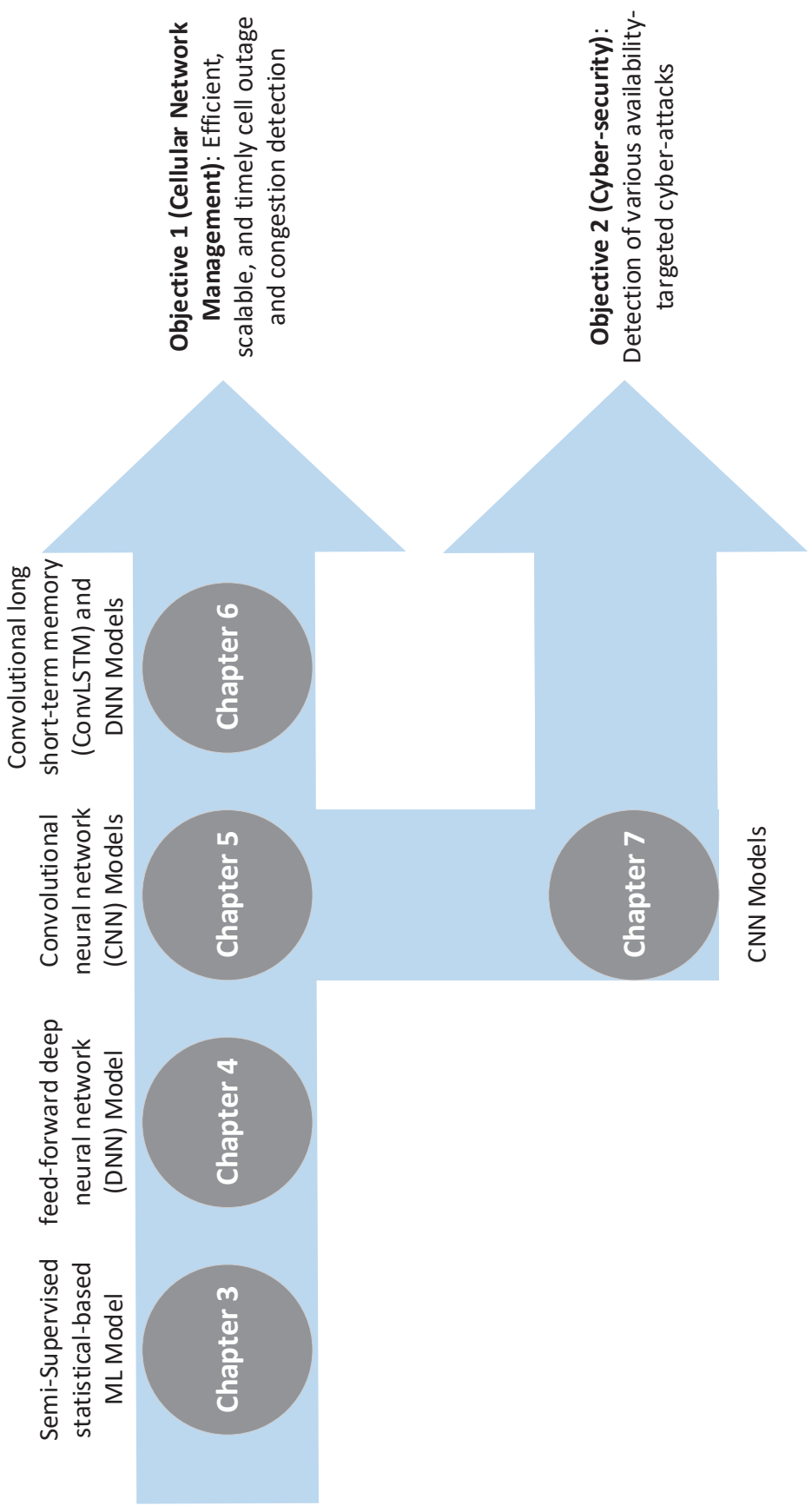

Figure 1.2: Thesis progression.

Chapter 2

Background

2.1 Anomalies Pertaining to Cellular Network Management

2.1.1 Cell Outages

Cellular outage corresponds to a base station providing abnormal services to its users and subsequently compromising the user QoE. It precipitates complete or partial abjection of the base station performance which eventually leads to customer dissatisfaction and conceivably escalates the churn rate. The major challenge, also a hazard, is that it occurs placidly and stays undisclosed from the operation, administration and maintenance (OAM) unit because the bases station still seems to be functional from the network's viewpoint [32]. Network failure detection is a tiresome task in conventional monitoring system and is highly reliant on pre-defined thresholds set for umpteen key performance indicators (KPIs), e.g. access failures, dropped calls, handovers, etc. [33]. Above all, automatic alarm triggering is nonfunctional in such scenarios because there is no way to convey the alarm messages to the network because of the failures explained later in this subsection [34]. The situation is unfolded when numerous complaints are received

from the clients or through manual drive tests, which require up to a few days to discover and even more to remedy [35]. A most recent example of a major large-scale outage is from Rogers (a Canadian telecom provider) [36] during which Call, SMS, and data services were severely affected because of a recent software update that negatively impacted an equipment residing the core network and as a consequence, service deteriorations were experienced all across the country. It took several hours to fix the problem. This in turn escalates OPEX; indeed, MNOs in United States alone consume over 15 billion USD per annum to cope with the cell outages [37].

Classification of Outages

Generally, we can segregate the outages into the classes described below [33], contingent on the extent of the deterioration:

1. **Impaired/degenerated cell:** hauls traffic lesser than the norm, causing degenerated performance. As compared to the other classes, the impact of impaired cell on the services is minimum.

2. **Crippled cell:** represents a severe case with the traffic capacity intensely ablated than the normal conditions.

3. **Catatonic cell:** is the most crucial class which equates to a dead cell. It is completely paralyzed and is incapable to ferry any traffic.

Causes

Outage occurs due to the following types of failures: logical/physical channel breakdowns and hardware malfunctions [32]. Example of the former is a situation when user equipment (UE) is unable to handover to the malfunctioned cell or establish a new connection. The culprit is the exorbitant burden, failed

random access channel (RACH) procedure, or firmware/software trouble at the base station. The situation is only valid for the new UEs while the existing already-connected ones enjoy adequate services. Hence, the sleeping cell eventually transforms into the catatonic class after acting as impaired and then the crippled class [33]. The latter type of failure (hardware malfunctions) arises because of the bidirectional antenna gain failure that converts a cell into a catatonic cell—the main reason being the malfunctioned transmit and receive modules at the base station [38].

Hence, prompt and automatic detection of outage cell is of utmost importance in the current and upcoming cellular systems so that the remedial actions could be timely executed. Such actions may include antenna tilt or reference signal power adjustment in the concerned cell, installment of temporary cell-on-wheels (COWs), dispatching unmanned aerial vehicles (UAVs) to cover the outage area, etc.

2.1.2 Congestion

Congestion can be defined as a situation with heightened traffic but having a relatively lesser throughput to gratify the users' thirst, interrupting performance of the network and ultimately hampering user's QoE [39]. Sudden hike in the base station's traffic can transpire congestion if necessary efforts are delayed [7]. Such necessary efforts may involve: earmarking extra resources to the ROI [7], unloading traffic to the neighboring cells [9], and dynamic billing in QoS-enabled cellular networks [10]. An efficient congestion-control procedure necessitates an accurate congestion exposure and it is crucial for the QoS-enabled networks offering high QoS-assured services to the consumers [10].

2.2 Anomalies Related to Cyber-security of the Cellular Networks in Relation with Cyber-Physical Systems

Various attacks [40], [41, Table 1] compromising the availability, confidentiality, or integrity can be staged against the cellular systems. Denial-of-service (DoS) attack aims at compromising the network resources' availability in an area and has an ability to bulldoze a network: contingent upon various such attacks executed in scattered and coordinated fashion known as distributed denial-of-service (DDoS) attack [40, 41]. As reported by Verizon [42], year 2017 witnessed DDoS attacks as the most recurrent cybersecurity occurrences. They can be machinated as a smoke screen or beachhead for cybersecurity specialists, following which some other intension(s) (for example, data breach) are to be attained [42, 43]: since DDoS attack cannot just severely harm the mobile wireless system and its legitimate subscribers but as a possible side effect, it can also disrupt the connected CPSs' operations which greatly depend on the wireless connectivity. Once the cellular systems are jeopardized, they can be abused as strong attack vectors targeting the CPSs; hence, adequate efforts shall be employed in order for them to avoid being leveraged as proxy to strike the CPSs.

Besides a zero-day threat[1], known vulnerabilities can also be exploited by the malevolent user(s) in mobile networks to organize a DDoS attack—whose mitigation is an open issue in 4G cellular system [40]. To stage a collective DDoS attack, a botmaster (cybercriminal) can leverage a chain of bots called botnet[2] and orchestrate various individual attacks for example signaling [44], silent call [2], and SMS flooding [45] attacks (described in the next subsections) [40, 41, 46].

[1] a vulnerability unknown to the network users, operators, or IT experts; and which is only discovered when they have been publicly exposed.

[2] a superimposed network containing heaps of malware-contaminated user devices capable of receiving instructions from the botmaster and acting on them.

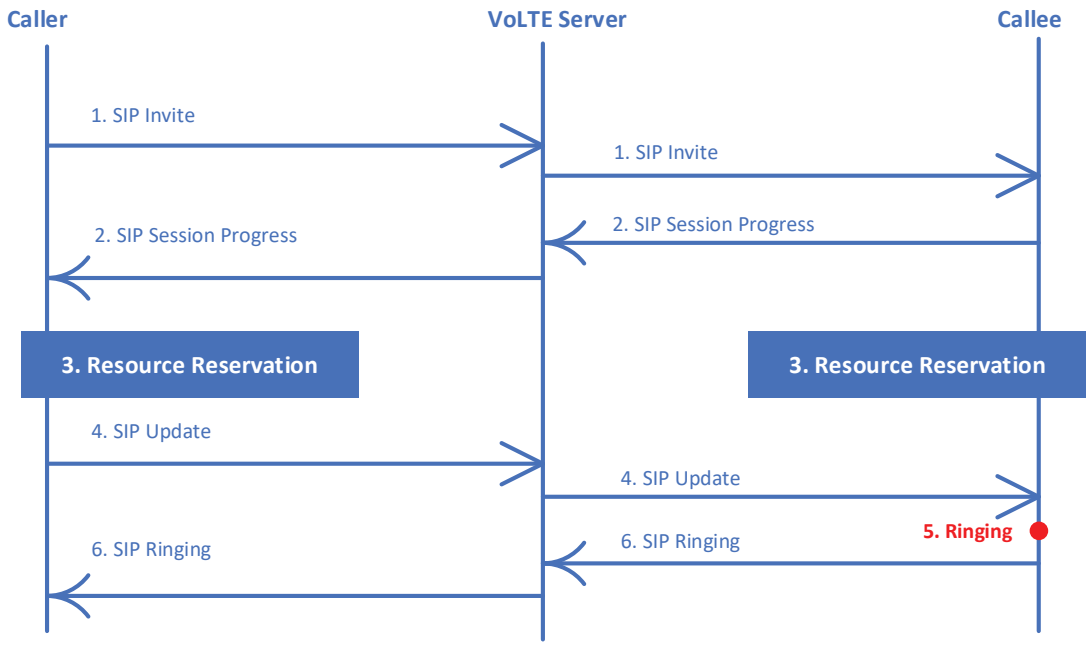

Figure 2.1: An abridged voice over LTE (VoLTE) call establishment procedure adopted from [2, Fig. 4].

Malwares can disseminate and be injected into these devices by exploiting email attachments, SMS, or other channels [41]. We can realize the botnet's threat severity from [42, Fig. 17], which illustrates global botnet violations in year 2017.

2.2.1 Silent Call Attack

Silent call attack is initiated by capitalizing on voice over LTE (VoLTE)'s[3] elementary design defect in call initialization process, depicted in Figure 2.1. The process sparks off when the VoLTE-supported device calls its prey and numerous messages flow among the caller, callee, gateways, and VoLTE servers. During the process, recourses are reserved for the call prior to the caller sending a "session initiation protocol (SIP) Update" message which ultimately allow the callee's phone to ring—this is where the malicious caller averts the message dispatch to skip the ringing. As a consequence and as resources have already been reserved by the network to execute the call, the calee's phone is obliged to remain in a

[3]VoLTE, a voice solution proposed for 4G LTE network

radio recourse control (RRC) mode that promptly bleeds the battery without the callee's awareness [2]. To demonstrate the attack, Xie *et al.* [47] have developed *VoLTECaller*, an Android application.

2.2.2 Signaling Attack

Signaling attack aims to overburden a CN entity (like a gateway) which deals with RRC-based signaling messages—transported among various network elements to facilitate effective resource orchestration [41]. A mischievous device appeals for a bearer arrangement (called random access) during the attack to transmit data and acquire "Connected" status; after being assigned the resources, it stands by till the timeout and repeatedly performs this action. Colossal signaling messages are generated during this process for the various network units (like UE, e-NodeB, mobility management entity(MME), serving gateway (SGW), and packet data network gateway (PDN-GW)) to execute. In total, bearer activation and deactivation requires 24 messages [48]—the number of messages can significantly amplify as an LTE/LTE-A device can trigger upto 8 bearers.

A similar attack procedure can occur in 3G networks, where a user tries to have a "DCH (dedicated channel)" state assignment [49]. A situation having multiple users synchronously performing the signaling attack can cause a DoS attack, forbidding legitimate users to access the network [48]. Although the above attacks in their corresponding studies are studied in the perspective of defending against harming a single device/user (harm such as draining out device's battery, unwanted billing, or some resource denial of service (DoS) to the user); but if the attacks are considered in a massive scale, it will result in distributed DoS attack choking a cellular network.

2.2.3 SMS Flooding Attack

SMS flooding attack (also known as SMS spamming attack against IP multimedia subsystem (IMS)) counts on the security vulnerabilities emanated from the technology transfer: from 3G circuit-switched (CS)-based networks conveying SMSs through control-plane to 4G IMS and packet-switched (PS)-based networks conveying SMS through data-plane. The main objective is to computationally overburden the IMS server by implanting copious counterfeited SIP/SMS messages amid a SIP session (established during the exchange of SMSs) between IMS server and the device's SMS client [41, 50].

A blend of the above three attacks can be staged in a dispersed and coordinated manner to disrupt availability of a cellular network in a region by leveraging the botnet [41, 46].

2.3 Big Data, Data Science, and Machine Learning

2.3.1 Big Data

Big data is an umbrella term for any collection of datasets so gargantuan or intricate that it becomes challenging to expeditiously process them by utilizing current data management theory and technologies [51]. We can describe them using 5 v's (as depicted in Figure 2.2), which distinguish them from the conventional data [22, 52]: volume (quantity), variety (diversity), velocity (data collection speed), value (valuation of the derived knowledge after data analytics), and veracity (data precision). We can measure the value in the form of shortened network expenditure [4], additional revenue generated, or network improvement,

[4]There are fundamentally two categories of expenses that MNOs endure: Capital expenditure (CAPEX) denoting procurement and upgradation of equipments needed for the network,

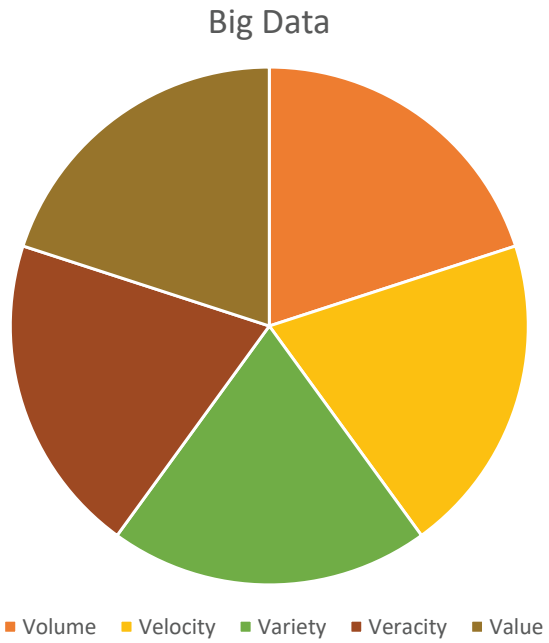

Figure 2.2: 5 V's of Big Data

etc.

2.3.2 Data Science

Data science, also known as data-driven research, is a wide area involving utility of various methodologies for analyzing the mammoth (Big) data to reveal the hidden knowledge and insights they possess in order to answer a particular question or set of questions [51, 53].

2.3.3 Machine Learning

Machine learning (ML) is a subgroup of a wider area of artificial intelligence (AI). It contains researches pertaining to the automatic large-scale (Big) data analytics that enable processing entities to accomplish optimal performance by learning from the example dataset (or past experiences) [54, 55]. Some of the subfields of ML including supervised, semi-supervised, and unsupervised learning

and operational expenditure (OPEX) denoting spendings related to maintenance and management of the network functions [6].

[56], are described below:

1 The aim in **supervised learning**, having a labeled dataset, is to figure out the input and output relationship so that the predicted output given a new input is precise [54].

2 The aim in **unsupervised learning**, having an unlabeled dataset, is to discover the data's latent structure, distribution, and values [51].

3 **Semi-supervised learning**, an amalgam of the above two, relates to the circumstances having inadequate labeled and copious unlabeled data. In such circumstances, semi-supervised learning is favored over the supervised counterpart especially when the data procurement is costly or airy [53].

2.4 Common Datasets Utilized for Anomaly Detection

2.4.1 Minimization-of-drive-test Dataset

Cellular networks can detect anomalies by leveraging various types of data. LTE networks broached minimization-of-drive-test (MDT) [57, 58] measurements to avoid manual drive testing and to dwindle OPEX. Hence, various studies [1, 35, 38, 59, 60] have utilized them for anomaly detection. However, MDT data deplete scarce network resources (communication, computation, and storage). In addition, since MDT measurements are mostly recorded at the user end, precise localization of the associated base station is a major problem for inaccuracy in anomaly detection which leads to high error rates. The above-mentioned limitations of MDT measurements are further elaborated in Section 3.3.

Cell ID	Time stamp[a] (milliseconds)	Country code	Subscriber activities[b]				
			SMS in	SMS out	Call in	Call out	Internet
...	...	...	...	...	...	...	...
3621	1388539800000	39	0.628319	0.274365	0.137755	0.058992	10.355361
3621	1388540400000	0	0.136609				
3621	1388540400000	39	0.374871	0.373152	0.147633	0.059278	15.282042
3621	1388541000000	39	0.265904	0.509475	0.000286	0.118844	12.445180
...	...	...	...	...	...	...	...

[a] Each entry represents beginning of a 10 minutes interval in Unix epoch. For example, 1388539800000 interprets as Wednesday, 01 January 2014, 1:30:00 AM (GMT). We can calculate end of the interval by adding 600000 milliseconds to this value.
[b] Some entries are missing that indicates no activity is recorded for the specified field.

Table 2.1: Sample CDR dataset extracted from the file pertaining to 1st January, 2014.

2.4.2 Call Detail Record Dataset

Due to the shortcomings of MDT dataset, we rather advocate the usage of call detail records (CDRs)—already present in the network for various purposes, primarily for billing [61]. They are extracted from LTE-A's core network. Fig. 2.3 depicts a general LTE-A architecture with network components and standardized interfaces. At the higher level, it mostly comprises user equipment (UE), access network i.e. evolved UMTS terrestrial radio access network (E-UTRAN), and CN. A node known as evolved NodeB (eNodeB or eNB) make up the access network and multiple logical nodes form the CN. These logical nodes include packet data network gateway (PGW), serving gateway (SGW), mobility management entity (MME), gateway mobile location center (GMLC), home subscriber server (HSS), policy control and charging rules function (PCRF), and evolved serving mobile location center (E-SMLC).

CDRs contain valuable user behavior information about the network resource (call, SMS, and Internet) utilization (see Table 2.1 for a sample of raw CDRs) that acts as a proxy to identify anomalies in the cellular networks. Various studies [7, 8, 62–65] have widely employed them for this purpose. This is the reason we assume that an abnormal user traffic behavior can sufficiently emulate the anomalies—abnormally low subscriber activity signals an outage or performance degradation in the cell, and abnormally high activity implies a possible congestion. The similar concept applied in the context of cybersecurity, further elaborated in details in Chapter 7. Hence, we leverage CDRs for anomaly detection throughout this thesis.

2.4.3 Datasets Utilized in this Work

In this study, we utilize CDR datasets which are spatiotemporal in nature and are made available by Telecom Italia [66]. They are geo-referenced and are based

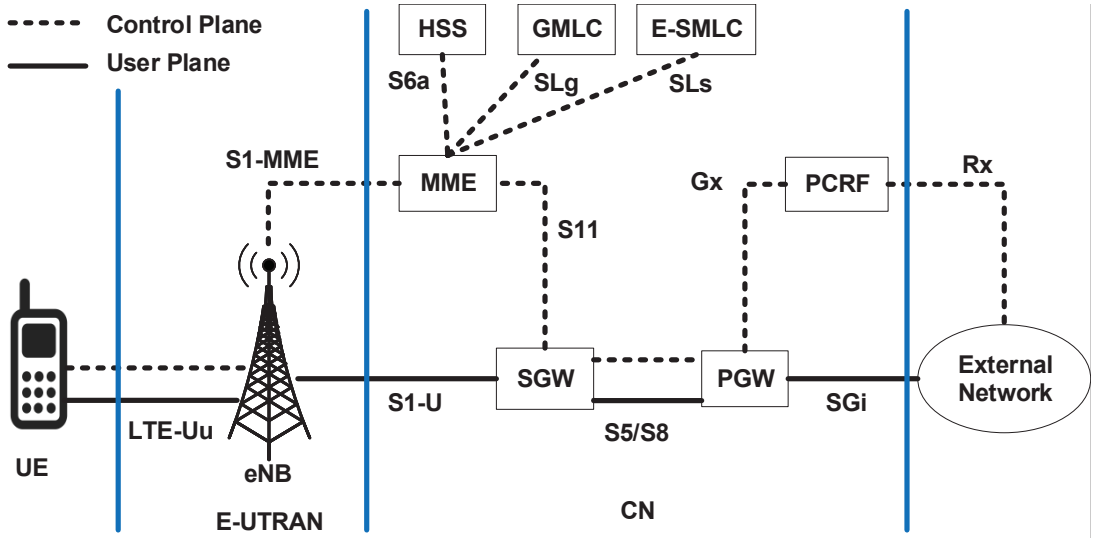

Figure 2.3: LTE-A Architecture.

on the subscriber activities from Milan and Trentino in Italy. Temporally, both datasets are split into 62 files each representing a recording day from 1st November 2013 to 1st January 2014. For each day, they contain subscriber activity logs recorded for every 10-min duration (timestamp). Each log entry contains the following information:

1. Cell/Grid ID containing the identification of the associated cell/grid during which the activity (voice, SMS or Internet) values were recorded.

2. Timestamp, containing starting value (in milliseconds and in Unix epoch format) of the 10 minute recording period,

3. Country code,

4. Received SMS (henceforth referred as SMS in) activity,

5. Sent SMS (henceforth referred as SMS out) activity,

6. Inbound calls (henceforth referred as Call in) activity,

7. Outbound calls (henceforth referred as Call out) activity, and

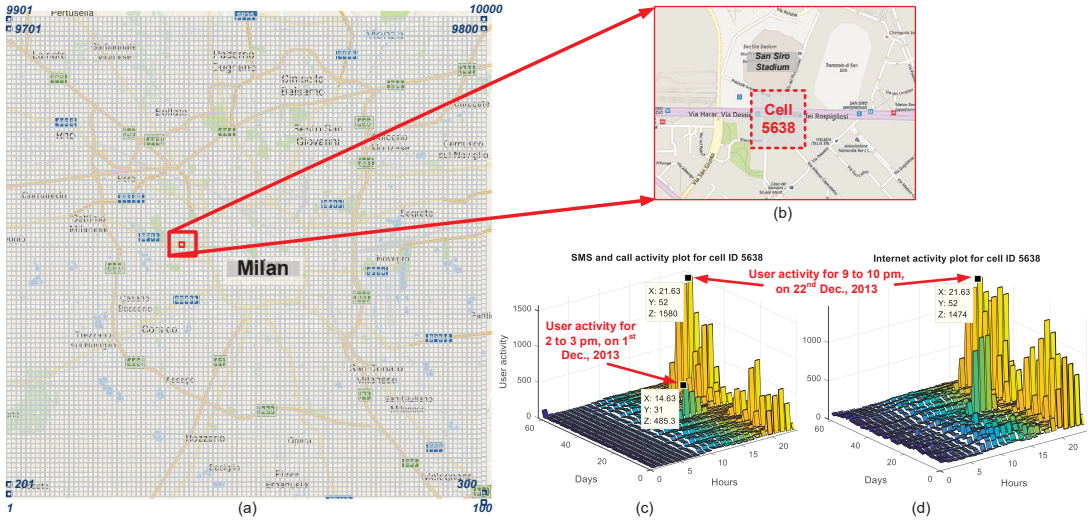

Figure 2.4: Visualization of Milan dataset. **(a)** 10,000 subgrids are overlaid with Milan's map, each subgrid having a side length of 0.235 km. **(b)** Cell ID 5638 located near the San Siro stadium is highlighted **(c)** and **(d)** display spatial distribution of cell ID 5638's SMS and call, and Internet activities, respectively.

8. Internet usage.

The activities (list items 5−8) measure the communication level of the subscribers with the base station having the designated Grid ID over a 10-min interval. The dataset conceals the precise activity values for privacy reason and anonymize them by giving values proportional to the real ones [67]. For instance, the larger the number of SMS or calls made by the subscribers, the larger is the activity of the SMS or calls made, respectively.

Milan Grid

Spatially, CDR data from Milan consists of more than 319 million user-activity logs (with each file, out of 62, containing an average of 5.15 million logs) for a 100 × 100 grid (known as Milan grid). This makes a total of 10,000 cells spread across the city. Each cell is a square sub-grid and has a 235 m length and the data cover a total area of 23.5 km^2. Figure 2.4 illustrates the 10,000 grids overlaid on the Milan's map.

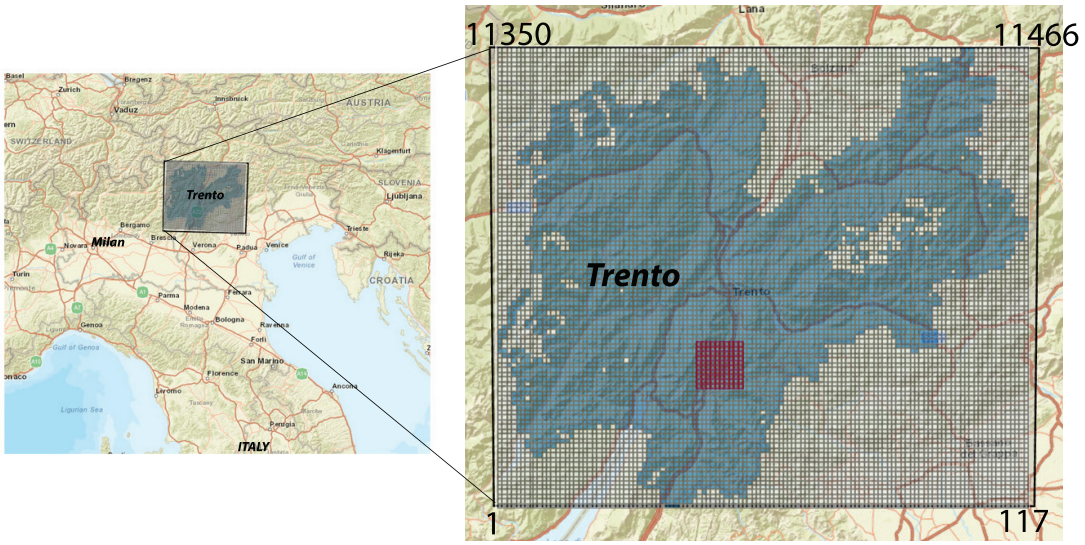

Figure 2.5: Spatial description of the Trentino dataset: Trentino grid and the Italy's map are superimposed by utilizing the GPS coordinates **(left)**. For reader's clarity and understanding of the covered area, we enlarge Trentino grid **(right)**.

For dataset visualization, we choose grid ID 5638 and highlight it in Figure 2.4(a) and (b). We display aggregated SMS and call (both inbound and outbound) activities in Figure 2.4(c), as they have the identical measurement scale; while Figure 2.4(d) demonstrates the Internet activity. The annotations emphasizing abnormal traffic surges on 1^{st} and 22^{nd} December, 2013 corresponds to the football contests happening at that time [62, Fig. 7(a)] and [7, Table 1].

Trentino Grid

Spatially, CDR data representing the Trentino's subscriber activities consist of more than 171.4 million records for a total of $6,259$ grid IDs. On average, there are 2.76 million records in each file. The data covers a 117×98 (Trentino) grid [66] with a single unit's side length of about 1 km. Figure 2.5 delineates the superimposition of the grid and the Trentino's map generated by utilizing the actual GPS coordinates.

2.5 Performance Metrics

Various widely-used performance metrics are utilized in the thesis. We start with elaborating the confusion matrix [68], which is based on the following components:

- T_{+ve} *(true positive)*: number of classifier (anomaly detector)-labeled abnormal instances that are verified by the ground-truth data to also be abnormal.
- T_{-ve} *(true negative)*: number of classifier-labeled normal instances that are verified by the ground-truth data to also be normal.
- F_{+ve} *(false positive)*: number of classifier-labeled abnormal instances that are actually normal (verified by the ground-truth data).
- F_{-ve} *(false negative)*: number of classifier-labeled normal instances that are actually abnormal (verified by the ground-truth data).

We further present additional metrics build on top of the confusion matrix:

- *Precision* is the fragment of the positive instances which are actually positive. It is given as follows:

$$Precision = \frac{T_{+ve}}{F_{+ve} + T_{+ve}}, \tag{2.1}$$

  It shows trustworthiness of the utilized model i.e. when the model predicts/classifies a test case to be an anomaly, it is more likely to be one. In the cellular network management's perspective, low precision value implies that there would be too many false positives and a significant amount of OPEX, in terms of sending technicians to the faulty sites which are actually not faulty (in case of outage detection) and utilizing extra resources in an ROI where in reality there is no need (in case of detection of high user traffic activity), would be wasted.

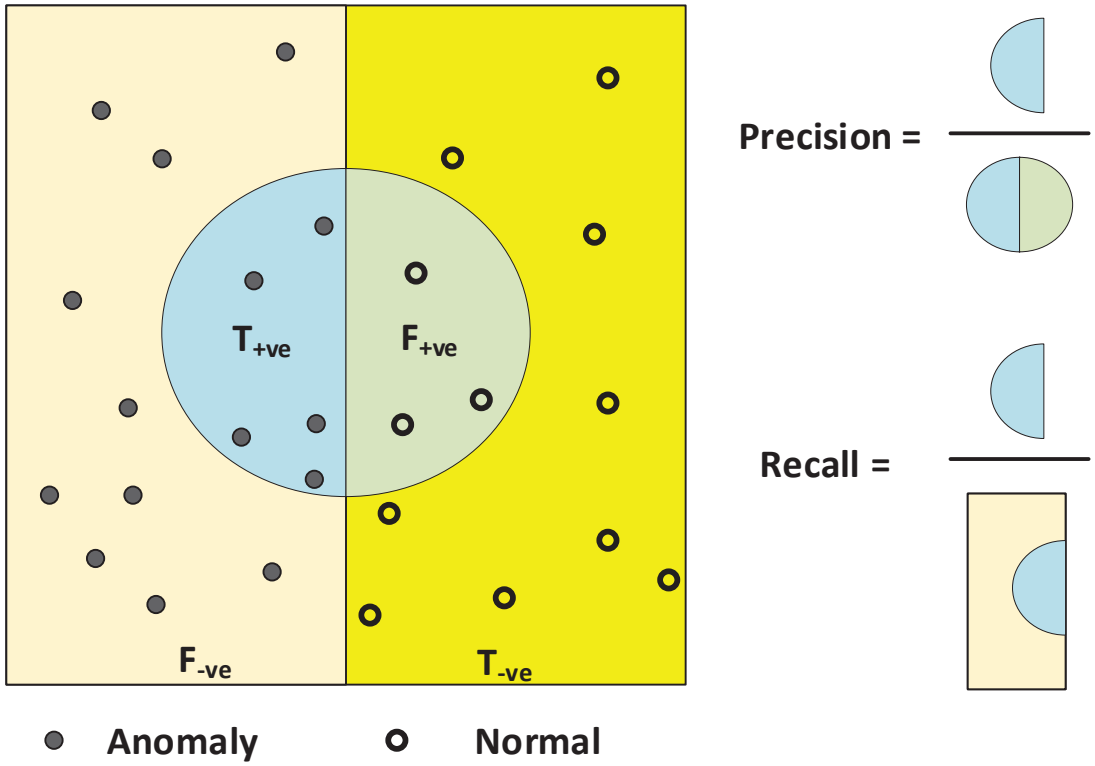

Figure 2.6: Precision and recall.

- *Recall* is the fragment of T_{+ve} instances out of the population of all positive instances and is given as follows:

$$Recall = \frac{T_{+ve}}{F_{-ve} + T_{+ve}}, \tag{2.2}$$

  Figure 2.6 illustrates the above metrics. High recall implies that our model can capture a large fraction of anomalies i.e. it has a wide breadth.

- F_1 *score* is the harmonic mean of the previous metrics (i.e. precision and recall). It describes the model performance in a single number and is given as follows:

$$F_1 = 2\,\frac{Recall \times Precision}{Recall + Precision}. \tag{2.3}$$

- *Accuracy* is the degree of success for the classifier (anomaly detector) and is

given as follows:

$$Accuracy = \frac{T_{+ve} + T_{-ve}}{T_{-ve} + T_{+ve} + F_{-ve} + F_{+ve}}, \tag{2.4}$$

- *Error rate* is the fraction of falsely classified instances and is given as follows:

$$Error\ rate = \frac{F_{+ve} + F_{-ve}}{T_{-ve} + T_{+ve} + F_{-ve} + F_{+ve}} = 1 - Accuracy, \tag{2.5}$$

- *FPR*, false positive rate, is defined as the fragment of F_{+ve} instances out of the population of all negative instances and is given as follows:

$$FPR = \frac{F_{+ve}}{T_{-ve} + F_{+ve}}, \tag{2.6}$$

2.6 Challenges in the Applications of AI Techniques in the Anomaly Detection

There are a few challenges related to the application of AI techniques for the detection of outages, congestion, and various attacks towards the availability of network services. They are mainly related to the data preprocessing. In the context of our work, each chapter applies a different AI technique and hence requires the Data to be preprocessed so that they can be acceptable for the applied model. The original data need to be carefully handled since they contain empty spaces which we fill with 0 to avoid processing error at the later stages. Additionally, for Chapter 7, in which we applied two different DL models, the preprocessing needs a careful consideration so that the data are appropriate for both models to work in sync.

Beside the preprocessing, another major challenge we face in this research is the shortage of labeled dataset. Milan and Trentino dataset lack labels and hence

we utilize a statistical technique to generate synthetic labels.

2.7 Conclusion

This chapter has presented preliminary topics before starting the proposed frameworks in later chapters. The topics mainly included the explanation about various anomalies considered in this work; a brief description of big data, data science and machine learning; and the elaboration of datasets utilized for anomaly detection followed by the details and visualization of the CDR datasets.

Chapter 3

Design, Analysis and Performance Characterization of Semi-Supervised Statistical-Based Cell Outage and Congestion Detection Framework

3.1 Motivation for Utilizing Semi-Supervised Technique for Anomaly Detection

The available CDR datasets utilized in this work are unlabeled. Since anomalies as compared with the normal instances are lesser in practice, using supervised learning techniques becomes unfeasible as they demand labeled dataset for the purpose of classification. In addition, as an anomaly can occur due to multi-fold reasons contingent upon the application domain, it is usually costly for the project managers/researchers to acquire data which represent every kind

of anomaly. Semi-supervised or unsupervised ML techniques can provide the solution under such circumstances and can discover abnormalities, depending on whether the available data is partially-labeled or unlabeled, respectively [14].

3.2 Overview and Contributions

In this chapter, we apply a semi-supervised statistical-based machine learning method to detect network's suspicious demeanor by analyzing the spatiotemporal information (i.e. CDRs based on Milan grid) on hourly basis. The method is essentially model-based, which assumes the data is distributed according to a Gaussian distribution i.e. $x \sim \mathcal{N}(\mu, \sigma^2)$ with mean μ and variance σ^2 as parameters [69] and that the normal data instances exist in the high probability area while abnormal data instances lie in the low probability area of a statistical model [14]. It generates a normal profile based on the observed pattern of the given data by fitting the model to the given data and then evaluates unseen test data instances with respect to how well they fit the model. Instances which differ significantly from the normal profile are marked as anomalies [70].

Following are the main contributions of this work:

1. A new approach for anomaly detection in wireless networks is proposed which is independent of any key performance indicator (KPI) and rather requires the already-available user-specific data (CDRs). As a consequence, outage irrespective of the kind of failure which caused it can be detected.

2. A consolidated method is presented which also detects an anomaly corresponding to the soared subscriber activity leading towards congestion, besides the sleeping cell.

3. Instead of detecting anomalies in the past 1 week data (which was done in Parwez *et al.* [7]), this method can successfully identify anomalies in the

past one hour by utilizing data science and machining learning tools to leverage Big Data containing past 2 months user activity data.

3.3 Literature Survey

The problem of anomaly detection is an old one as it depends on the overall context before defining the anomaly; this makes anomaly detection a broad area and therefore it has been investigated in many works [16, 71, 72]. Various ML algorithms have been employed in the literature based on unsupervised, supervised, and semi-supervised learning methods involving a variety of techniques like knowledge-based or soft computing-based, clustering, classification, and statistical techniques [73].

To discover collective anomalies, Plessis *et al.* [71] presented an unsupervised detection method utilized for a chain of captured events from the mobile network. They simulated mobile devices behavior using a few days of data extracted from a commercial LTE network to determine abnormal patterns and detect anomalies precipitated due to (1) user and network device problems, and (2) development of abnormal and abrupt events like earthquake.

The researchers in [72] proposed a semi-supervised framework for fuzzy classification-based anomaly detection in self-organizing networks (SONs). Problems related to capacity and coverage bring forth by lofty interference from the adjoining base stations are equated as anomalies. Principal component analysis (PCA) is employed to convert the simulator data (constructed from terminal measurements and KPIs) into a lower dimension form.

A knowledge-based, specifically a rule-based, technique is proposed by Karatepe *et al.* [16] in which location-based anomalies are identified by analyzing CDRs of the intercity traveling users. Anomalies can be caused by the network-side misconfiguration of position information or the incorrect mappings of associated

location features in an IT system. The technique incorporates subscriber device's traveling velocity to set a pre-defined rule for the technique to utilize.

The above-mentioned works deal with a broad range of anomalies while the following works treat outage or congestion situation as an anomaly. Detection of catatonic cell caused by hardware malfunction is the focus in articles [32, 35, 38, 74, 75]. Mueller *et al.* [74] employed an algorithm that uses neighbor cell list (NCL) reporting of mobile devices for anomaly detection. However, the algorithm yields high false alarm rate which questions the practicality of the algorithm in real-world applications.

K-means clustering, an unsupervised technique, is applied by Chernogorov *et al.* [32] for anomaly detection after reducing the dataset dimensionality using diffusion maps. Likewise, k-nearest neighbors (K-NN) and local outlier factor (LOF) algorithms are employed by Zoha *et al.* in [38] along with one-class support vector machine (OCSVM) algorithm in their elongated work [35], after reducing the dataset dimensionality using multidimensional scaling (MDS) method to identify the anomalies. The authors used various KPIs extracted through MDT measurement dataset for the purpose. Additionally, they utilized the geographical data linked with the measurements to locate the outage cell. But, a major disadvantage of periodic measurements[1] is they deplete ample user and network resources [33]. Moreover, the cogency of the acquired location is questionable because location information is not present in every MDT measurement and for the samples in which it is present, the localization accuracy is not adequate to pinpoint the source base station [58]. Research by Turkka *et al.* [75] also shares the above-mentioned limitations.

In contrast to the above works, RACH (instead of hardware) failure is the center of attention of the studies described below. Chernogorov *et al.* in [76] and

[1]There are 2 kinds of MDT measurements: logged and immediate. Immediate MDT measurements are used in [35, 38], constituting 2 measurement modes: event-triggered and periodic [33].

[33] proposed a N-gram analysis (a data-mining technique)-based semi-supervised model for the purpose by analyzing event-triggered MDT measurements. Minor component analysis (MCA) technique is utilized by the model to reduce the dataset dimensions and K-NN anomaly score algorithm for anomaly detection. The authors utilized different mapping techniques at the post-processing phase for localizing the outages by using the abnormal MDT measurements; however, the location accuracy is dubious.

Additionally, Parwez et al. [7] leveraged CDRs-based big data (Milan dataset) to detect network's abnormal conduct (anomaly) in terms of abrupt hike in traffic activity which can initiate congestion. K-means and Hierarchical (unsupervised) clustering algorithms were employed for this purpose and the proposed solution yielded 90% accuracy. However, time-efficiency of the proposed method is a major concern as it was utilizing past week data for the anomaly detection which might loose the practical utility; as, by the time framework detects the anomaly, MNO would have already gotten the information from the irritated customers.

This chapter attempts to overcome some of the above-mentioned limitations of various works with the following salient features:

1. Since this work utilizes CDRs instead of MDT measurements, it provides a lighter solution as unlike the latter the former are already present in the network and hence save the precious network resources.

2. As MDT measurements face issues related to localization inaccuracy which leads to misclassification of anomalous cells, CDRs don't have such limitations by design as they are generated in the core network rather than at the user end. Each CDR log contains Cell ID which indicates the identification of the base station which is linked with the recorded activity values. We utilize Cell IDs to easily and accurately locate the anomalous cell in this research. Our method basically processes the traffic activities in a chosen Cell

ID and contrast them with the previous activity values at the same hour to detect an anomaly; this makes localization of individual user equipments irrelevant.

3. Since we avoid MDT measurements containing various KPIs which are leveraged to detect different types of outage cells, as observed in this literature survey section; our method is general and can detect outage cell irrespective of the failure which initiated it.

4. In practice, outage stays undisclosed for several hours to a few days in existing wireless networks [35]. Our proposed method detects and pinpoints its location in an hour instead of taking a week (as was done in Parwez et al. [7]).

5. Our work is dual-purpose as it not just detects outage but also a situation leading towards congestion i.e. traffic activity hike.

3.4 Anomaly Detection Framework Design

3.4.1 Data Preprocessing

Below-mentioned are the reasons why the raw CDRs can't be directly used by our algorithm, which raises a demand for their preprocessing [53]:

1. The algorithm will generate error if data with empty entries are passed on. Milan dataset contain high volume of such instances.

2. Original CDRs are split up into 62 files, each pertaining to a single day.

3. There are some needless parameters in the dataset which are not required for this work, for example, "Internet" activity and "Country code".

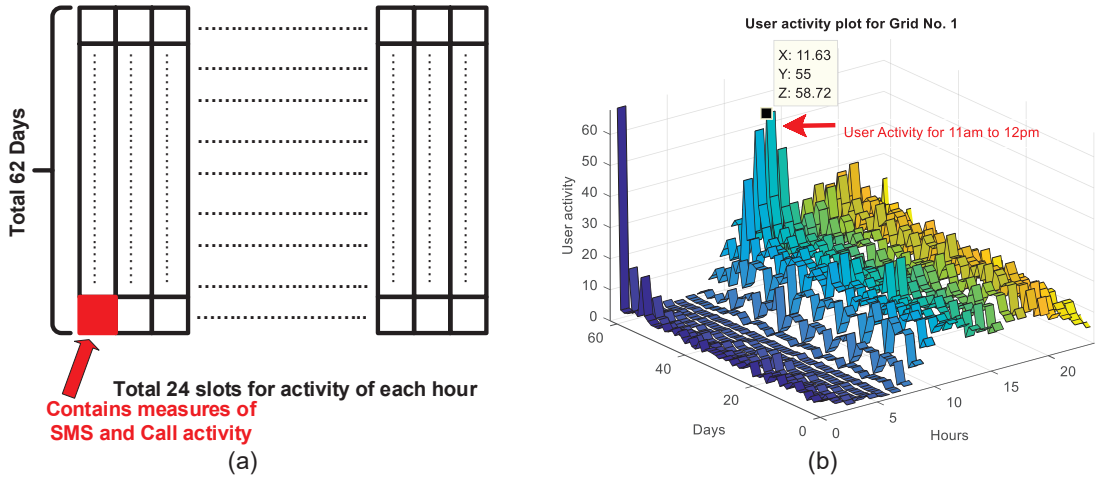

Figure 3.1: Dataset visualization: (a) Original CDR dataset represented as a 62×24 matrix. Each unit representing the activity (SMS and Call) value registered in an hour. (b) Ribbon diagram for Grid 1's traffic activity.

Following are preprocessing steps (also given in Step 1 of Algorithm 1) which ensure the final form of the data is compatible with our proposed algorithm [51]:

1. Cleansing: We impute 0 in the empty activity entries.

2. Combination. Since SMS and Call activities share the same scale [66], we add them to create a single item; henceforth referred as "activity". We then aggregate activities in 10-minute timestamps to convert them into activities collected for every hour. Lastly, we compose a 62×24 dataset, as shown in Figure 3.1(a). It represents hourly activities for 24 hours in 62 days for which the data is available.

3. Transformation. For a selected grid ID and an hour, we constitute a 62-row vector which represents activities of 62 days. Although any hour can be chosen, for this work we choose 11am to 12pm (a rush hour) for further proceedings.

We utilized MATLAB version 2017a for preprocessing as it efficiently handled the available dataset.

3.4.2 Splitting the Dataset

We split the dataset as follows:

1. *Training set*, containing 70% i.e. 44 examples of the total 62, is the set from which our Gaussian-based algorithm learns by analyzing the normal traffic behavior. These examples are unlabeled.

2. *Cross-validation (CV) set* and *Test set*, each contain 15% i.e. 18 examples of the total 62 and are labeled. To determine the optimal threshold value, we employ CV set and the test set is employed for anomaly detection since it contains unseen instances by the algorithm.

The chosen dataset splitting ratio is a common practice, for example, as done in [77]. We assume the original dataset mostly contains normal instances since it is unlabeled and provided by the network operator itself; however some abnormal instances can also be presented indicating high traffic activity which is actually confirmed by the results in [7]. Our semi-supervised ML-based algorithm requires a few labeled (both normal and abnormal) instances. In practice, labeled data can be generated by utilizing information about the past anomalous occurrences in the network for which outage or congestion occurred due to abrupt user activities.

In this connection, the CV set must contain small amounts of those anomalies for its function and hence are artificially created, for example, by manually inserting low and high user activity values. Similarly, many artificial examples are also injected into the test set and are labeled as anomalies if their value deviates more than 2σ (standard deviation) from the mean of training set examples. An instance at 0 is also considered which correspond to a catatonic cell condition having no user activity and is labeled as an anomaly. The injection of instances in the test set is performed in order to demonstrate the performance of our algorithm by applying various performance metrics that requires labeled test set, discussed later in this section.

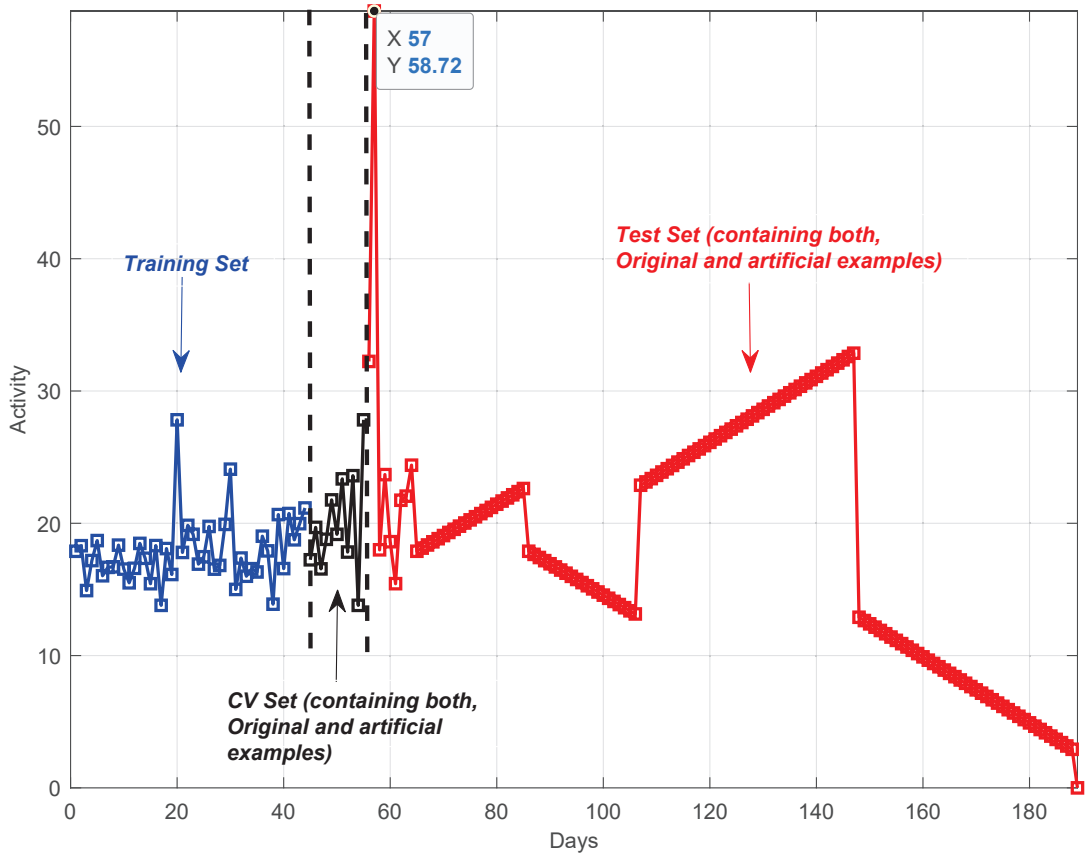

Figure 3.2: Illustration of complete dataset used for anomaly detection in Grid 1 for 1100 to 1200 hours.

The test set instances can be considered as the cases that resemble what our algorithm will be asked to process in the future. In general, given a new instance $x_{test}^{(i)}$, the main purpose of the algorithm is to find if it is normal or anomalous and thus, it can be treated as a binary classification problem. Figure 3.2 illustrates the complete dataset including training, CV and test set, used for Grid 1 for 11am to 12pm.

3.4.3 Semi-Supervised Statistical Based Anomaly Detector

After splitting the dataset into training, CV, and test sets, the algorithm estimates the distribution parameters, mean $\mu_1 \ldots\ldots \mu_n$ and variance $\sigma_1^2 \ldots\ldots \sigma_n^2$,

Algorithm 1 Semi-Supervised Statistical Based Anomaly Detection Method

Inputs: $GRID\ ID$: Identification of the current grid.
h: Selected hour for analysis.
$CDR\ Dataset$: Files containing user activities of SMS and Calls (both inbound and outbound), and Internet in addition to other entities mentioned in Section 2.4.3.
Output: Anomalies detected in grid $GRID\ ID$ for h hour.
Method:

1. For grid $GRID\ ID$, do the following steps for data preprocessing:
   1.1. Fill missing entries in CDR with 0.
   1.2. Add all the user activities (except internet) to form a single entity.
   1.3. Aggregate the single entity for the 10 minute timestamps to make it for one hour, for a total of 24 hours and for each day i.e. for every file.
   1.4. Construct a 62×24 matrix representing the user activity for 24 hours for 62 days.
   1.5. Extract user activity for h hour and form a row vector containing activity of each day.
2. Split the row vector into training, CV and test sets.
   2.1. Add few artificial anomalies into the CV and test sets as mentioned in Section 3.4.2.
3. Using training set, calculate μ_j and σ_j^2 from Eq. 3.1 and 3.2 respectively.
4. Calculate $p(x_{train}^{(i)})$ from Eq. 3.3.
5. Using CV set, select *best* ϵ (variable initialized 0 and outside the following *for* loop) which maximizes F_1 score in Eq. 2.3, by implementing the following:
   5.1. Calculate $p(x_{CV}^{(i)})$ from Eq. 3.3 using old parameters.
   5.2. **for** various values of ϵ
        Mark anomalies if $p(x_{CV}^{(i)}) < \epsilon$.
        Compute *TP, FP, TN* and *FN* as per description given in Section 3.4.3.
        Compute *Precision, Recall and* F_1 *score* from Eq. 2.1, 2.2 and 2.2, respectively.
        if current F_1 *score* $>$ *best* F_1 *score* (variable initialized 0 and outside this *for* loop)
            Update *best* F_1 *score* to current F_1 *score*.
            Set the current ϵ to *best* ϵ.
        end
   end
6. Calculate $p(x_{test}^{(i)})$ using Eq. 3.3 using old parameters.
7. Mark x_{test} as anomaly if $p(x_{test}) <$ *best* ϵ, normal if $p(x_{test}) \geq$ *best* ϵ.
8. Mark x_{train} as anomaly if $p(x_{train}) <$ *best* ϵ, normal if $p(x_{train}) \geq$ *best* ϵ.

from the training set $\{x^{(1)}, \ldots\ldots, x^{(m)}\}(x^{(i)} \in \mathbb{R}^n)$ using the following equations:

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}, \tag{3.1}$$

$$\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)} - \mu_j)^2, \tag{3.2}$$

where, j is the index of the feature, n is the total number of features and m is the total number of examples. Since our dataset is one-dimensional i.e. containing single-entry user activity, j and n are equal to one. Then, we can apply and adjust the general statistical anomaly-detection approach invented by the machine-learning research community [30] to our specific problem in cellular networks. The specific scheme for our framework is summarized in Algorithm 1. The algorithm fits Gaussian model on the training set by computing probability density estimation $p(x_{train}^{(i)})$ for each training example:

$$\begin{aligned} p(x^{(i)}) &= \prod_{j=1}^{n} p(x_j^{(i)}; \mu_j, \sigma_j^2) \\ &= \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\sigma_j}}\ exp(-\frac{(x_j^{(i)} - \mu_j)^2}{2\sigma_j^2}) \end{aligned} \tag{3.3}$$

We are interested to identify the anomalous examples which are more likely to have a very low probability. This can be done by selecting a threshold ϵ based on CV set $\{(x_{CV}^{(1)}, y_{CV}^{(1)}), ..., (x_{CV}^{(m_{CV})}, y_{CV}^{(m_{CV})})\}$, where the label $y = 1$ corresponds to an anomalous example and $y = 0$ corresponds to a normal example. For each CV example, density estimation $p(x_{CV}^{(i)})$ is computed by using Eq. 3.3 with old parameters. Using all these probabilities $\{p(x_{CV}^{(1)}), ..., p(x_{CV}^{(m_{CV})})\}$ and corresponding ground truth labels $\{y_{CV}^{(1)}, ..., y_{CV}^{(m_{CV})}\}$, we run an iterative process where a confusion matrix, elaborated in Section 2.5, is computed for many different values of ϵ. Note, as the CV set is labeled, the anomalous examples in the CV set is supposed to represent past occurrences of a network at which there was an actual anomaly. Hence, we term the labels of CV set as ground truth labels in this paper.

Using confusion matrix, *Precision*, *Recall*, and F_1 *score* are subsequently calcu-

lated. The value of ϵ (denoted as *best* ϵ in Algorithm 1), which corresponds to the highest F_1 *score* is selected. Finally, given test set $\{(x_{test}^{(1)}, y_{test}^{(1)}), \ldots..., (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})\}$, the algorithm computes density estimation $p(x_{test}^{(i)})$ by using Eq. 3.3 with old parameters and uses selected value of threshold ϵ as a dividing line to differentiate a corresponding normal instance from an anomaly which can formally be expressed as follows:

$$x_{test} = \begin{cases} Anomalous, & \text{if } p(x_{test}) < \epsilon \\ Normal, & \text{if } p(x_{test}) \geq \epsilon \end{cases} \tag{3.4}$$

The algorithm also repeats the above for the training set to detect anomalous instances in the past data for calibration, explained in the next section.

3.4.4 Performance Metrics

We employ labeled test set and utilize prediction *Accuracy, Error rate, FPR, F_1 score, Precision, and Recall* (elaborated in Section 2.5) for our algorithm's performance analysis.

3.5 Experimental Results and Discussion

It can be examined from Figure 3.3 that the algorithm has successfully traced unusual network behaviors i.e. anomalies, in the test as well as in the training set, which are represented by red diamond. The test instances that significantly deviate from the training data (which is comprised mainly of normal instances), are marked as anomalies by the algorithm. The detected anomalies on the right and left hand side of the figure corresponds to the surge in the traffic activity and the outage cell, respectively, in Grid 1 from 11am to 12pm. The values in the middle are normal instances having an inbound and outbound traffic flow as per norm. The anomalous instance on the right hand side having user activity value

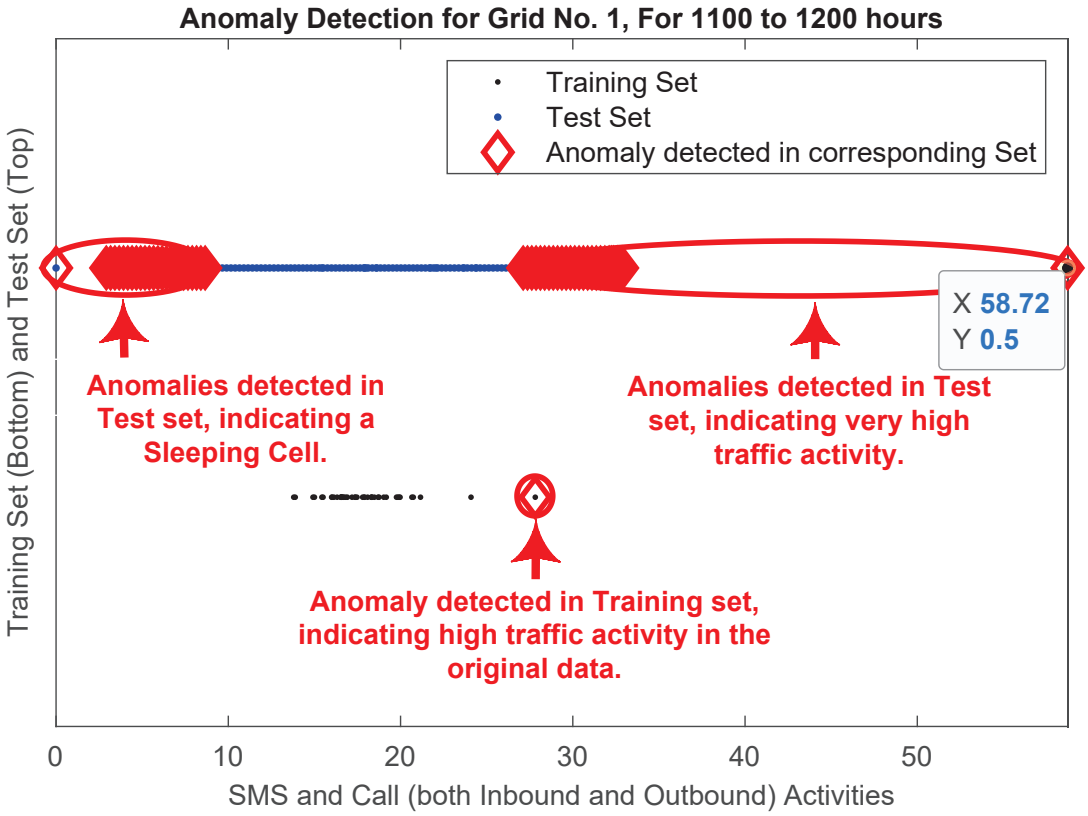

Figure 3.3: Anomaly Detection for Grid 1, for User Activity between 11am and 12pm

of 58.72 can also be seen as an abnormality in Figure 3.1(b). The algorithm also marked a test instance having 0 user activity as anomaly, denoting a catatonic cell in Figure 3.3. The rest of the marked anomalies on the left hand side of the figure denotes a crippled cell.

In addition, the algorithm also detected unusual network behavior in the training set and marked it an anomaly. The value of that anomaly can be seen as being considerably diverged from the majority of the data. As the training set consists of real instances, having measure of user traffic activity from 1^{st} November till 14^{th} December 2013 (i.e. 42 days data), the detected anomaly in training set corresponds to a real situation where the traffic activity is unusual.

In this connection, we compare our results with the ones presented by the authors in [7], for calibration. They reported anomalies in grids 5638 to 5640

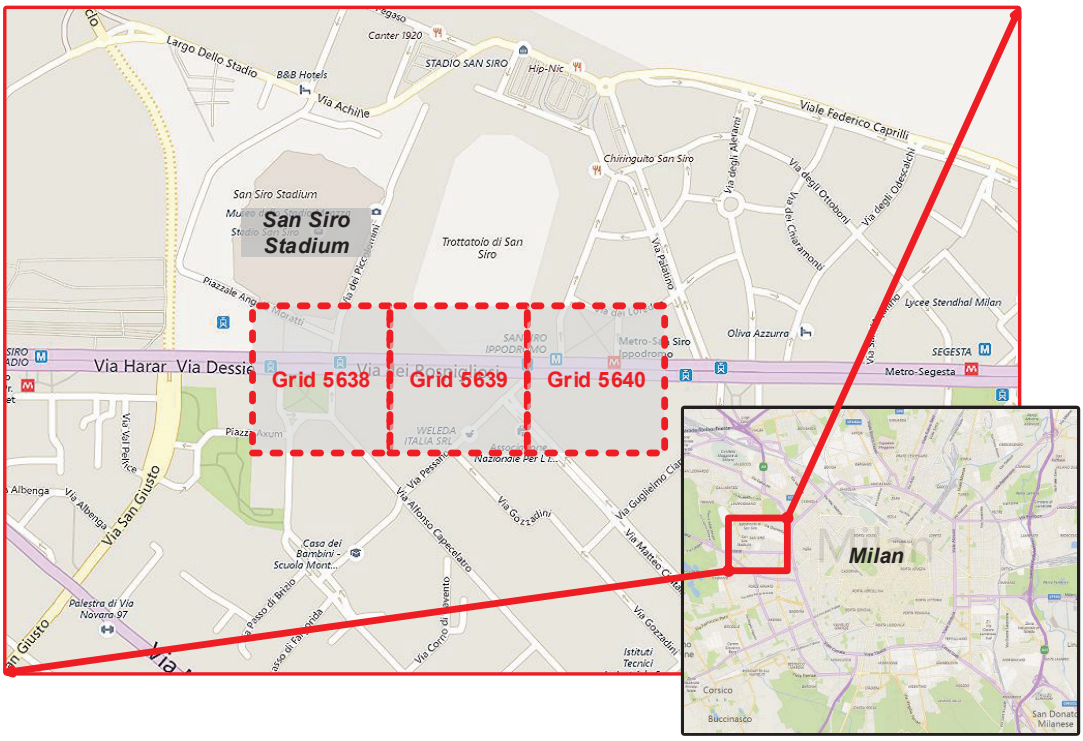

Figure 3.4: Location of Grids 5638-5640 in Milan, Italy.

which were detected using the data of the first week of December, 2013 (for our algorithm, this data is included in the training set) and upon investigation, it was found out that the unusual network behaviors were due to very high traffic flow which occurred because of an ongoing football match and also because of a busy hour. We used the coordinates to locate the stadium and the road where these anomalies occurred, illustrated in Figure 3.4. It can be seen in the figure that grid 5638 is near a stadium named "Stadio San Siro" and grids 5639 and 5640 are located on a nearby road. Figure 3.5(a)-(d) shows the results of our algorithm for these grids. It can be observed, that our algorithm has successfully detected these anomalies corresponding to high traffic activity. In addition, the algorithm has also falsely classified a few normal instances as anomalies which are interpreted as sleeping cells, due to their very low user activity values. Figure 3.5(c) depicts the detected anomalies for grid 5640 in a different perspective, showing a region having normal instances. Figure 3.5(d) shows some additional marked anomalies

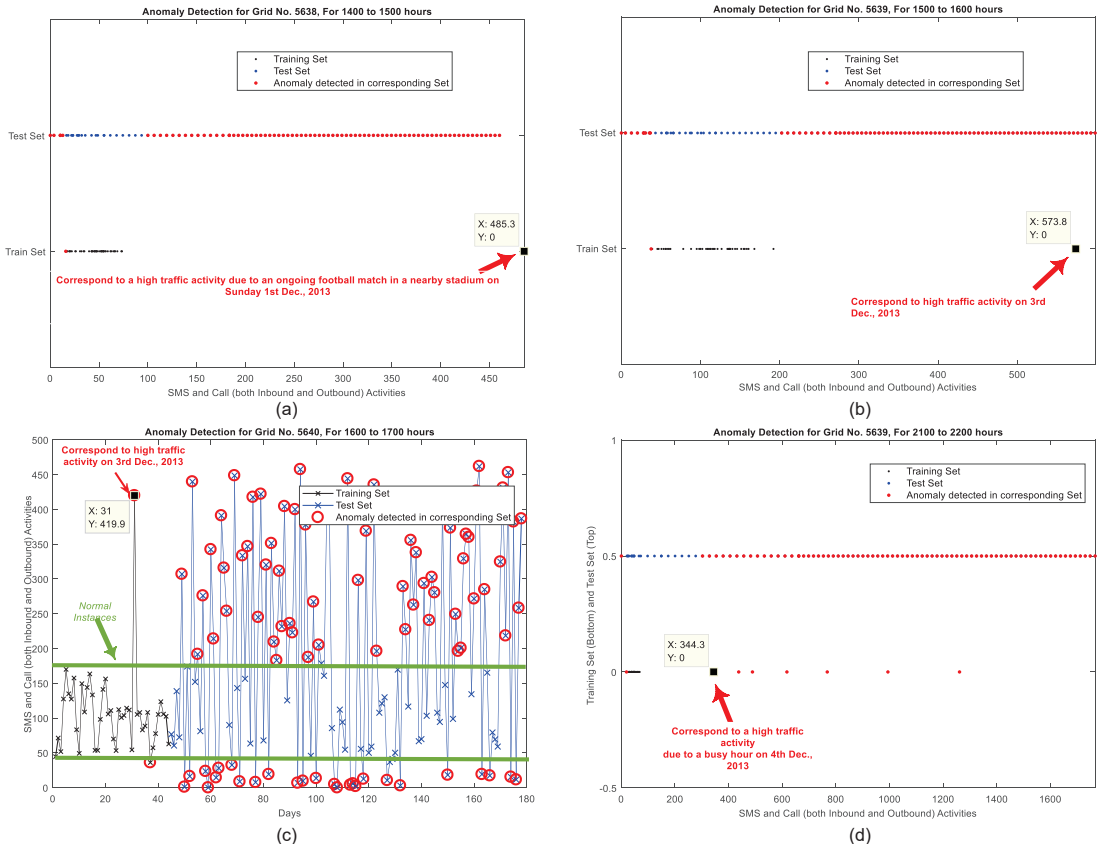

Figure 3.5: Anomaly detection for Grids 5638-5640 in Milan, Italy for different time instances.

which can be interpreted as the instances where there was a busy hour for several days.

3.5.1 Performance Evaluation and Analysis of Overall Results

We have selected 200 out of a total $10,000$ grids, and also selected three different hours for the performance evaluation of our algorithm. The selected hours are morning hour: $7-8$ am, afternoon hour: $12-1$ pm and night hour: 11 pm$-$12 am. Test dataset is utilized to determine various performance measures mentioned in the earlier section, the cumulative distribution function (CDF) plots of accuracy and FPR are illustrated in Figure 3.6.

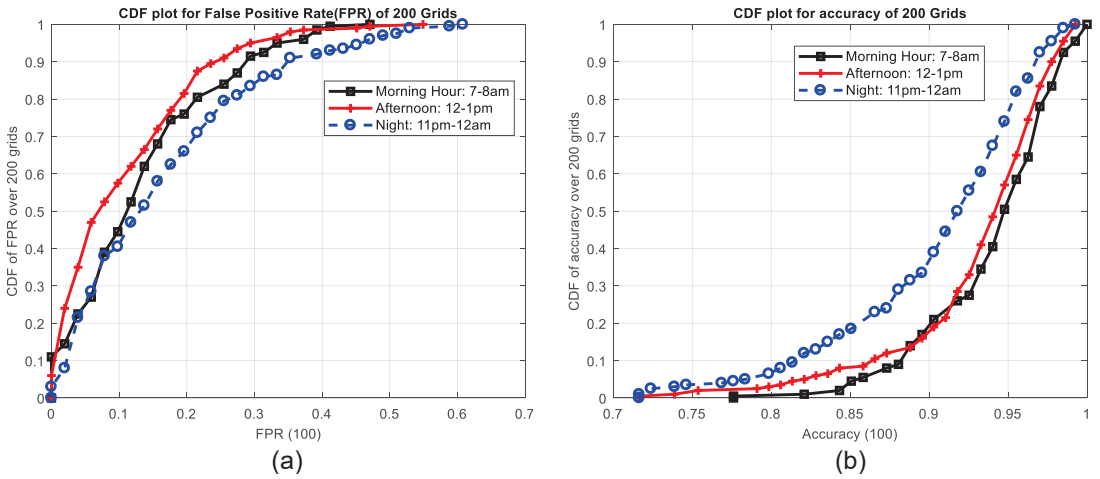

Figure 3.6: Different performance measures of our algorithm, calculated using user activity data for 200 grids at three different hours.

Table 3.1: Performance Statistics of our Anomaly Detection Algorithm

Measures	**Morning Hour**	**Afternoon Hour**	**Night Hour**	**Overall**
Accuracy	94.29%	93.36%	90.73%	92.79%
Error rate	5.71%	6.64%	9.27%	7.21%
F_1 Score	95.63%	94.61%	92.55%	94.26%
FPR	13.91%	11.41%	17.08%	14.13%
Precision	92.45%	93.68%	90.89%	92.34%
Recall	99.33%	96.29%	95.52%	97.05%

We report the performance statistics of our algorithm in Table 3.1 and its graphical representation in Figure 3.7. Our proposed method to detect anomalies (pertaining to outage and high surge in traffic activity), is able to achieve an overall detection accuracy of about 92% while retaining the overall error rate within approx. 7%. The achieved accuracy of our algorithm is about 2% higher than the reported accuracy of [7], which detected anomalies related to only high surge in user traffic activity. Overall precision of 92% is an evidence for our algorithm to be trustworthy. High recall of about 97% shows our algorithm's wide breadth. Lastly, F_1 score achieved by our algorithm is about 94%.

It can also be observed that accuracy, F_1 score and recall values are slightly

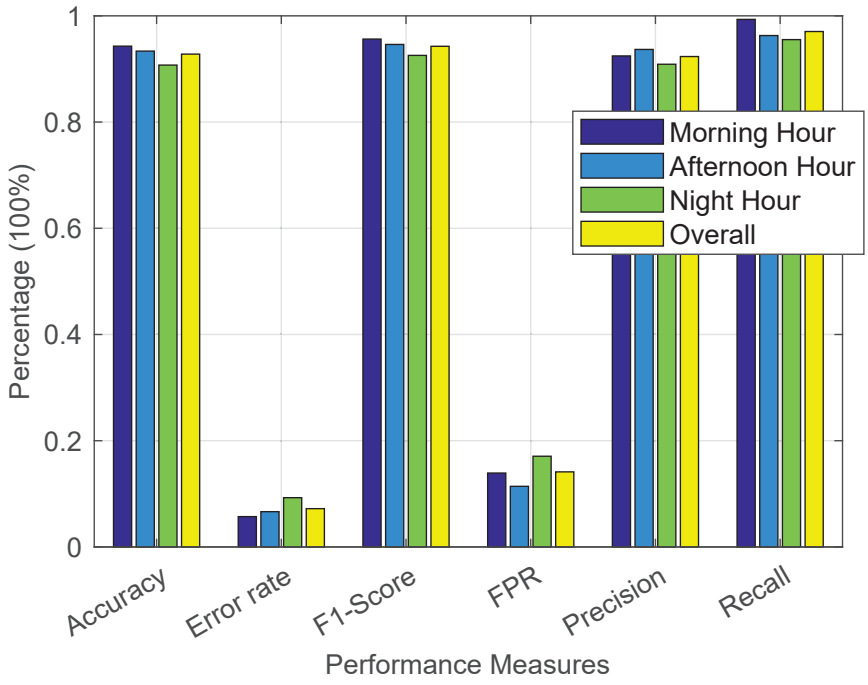

Figure 3.7: Performance of our proposed algorithm.

higher in morning hour as compared with afternoon and night hours, while precision value for afternoon hour is higher than the rest of hours. Holistically, the performance of the algorithm degrades at the night hour, due to very low user activity which results in normal values being assembled near the origin and marking anomaly if there is a slight increase in the user activity. This is also reflected in a relatively high FPR of 17% at night hour mentioned in Table 3.1 and depicted in Figure 3.6(a).

3.6 Summary

The purpose of this work is to utilize data science and machine learning to make inference from the large amount of dark data (big data), which in our case is the CDR dataset collected in 4G LTE-A network mainly for billing purpose by the customer services department but is never utilized or even accessed by the OAM department to derive insights about the network operations and to improve

the overall network's performance [78]. To extract actionable knowledge out of it, we utilized semi-supervised statistical-based anomaly detection algorithm which analyzed the spatiotemporal information on hourly basis and marked anomaly when it observed an unusual network behavior in a particular region.

The identified anomaly is further categorized into two classes. First, an outage cell (either crippled or catatonic, and caused by any of the discussed failure) having very low or no user activity values, for which the drive test team or technicians can be sent immediately to take appropriate actions. Second, a region having a very high user activity such as a stadium having an ongoing sports match, busy highway, etc. for which additional resources may require to be allocated for a smooth run of the network.

From the perspective of a cellular network operator, our results show that our method successfully leverages big data to identify ROIs in time i.e. in an hour, which otherwise would take several hours or even days [35]. Our method also provides business value in terms of reducing OPEX because of an automatic and prompt detection of outage cell; provides a lightweight solution for anomaly detection in a sense that it requires lesser network resources due to utilization of CDR data; prevents serious revenue loss as timely detection of anomalies contributes in providing enhanced user QoE and consequently reduces churn rate; and increases user satisfaction because additional resources, when the traffic demand is high, can be provided upon successful detection of anomalies.

The proposed method can also contribute towards self-healing capability of the SON and can trigger cell outage compensation (COC) function upon detection of sleeping cell to maintain as much normal services to the subscribers as possible [38, 79] for example, by serving the affected users by re-connecting them to neighboring cells until the fault is solved. In addition, the anomaly detection method can also be functional in the perspective of smart IoT community [80], however it needs further investigation.

Chapter 4

Feed-forward Deep Neural Network and Mobile Edge Computing-Based Cell Outage and Congestion Detection

4.1 Motivation

Deep learning (DL) surpassed many traditional ML models' performances and achieved breakthroughs in various fields: genomics, natural language processing, and computer vision [81]. In addition, mobile edge computing (MEC)—with distributed computation, storage, and network management, in contrast to centralized cloud computation design—has lately grabbed recognition for its possible service in 5G cellular networks to shift computation close to the borders (e.g. base stations and access points). The aim is to aid core network (CN) in running bulky assignments and empower computation-intensive and latency-critical services at resource-limited user devices by utilizing colossal resources accessible at the edges [82, 83]. Motivated by the utility of MEC and popularity of DL

technology, we speculate that DL combined with MEC might have a significant role to play in anomaly detection.

4.2 Overview and Contributions

We develop further our last work in Chapter 3 and propose an improved framework for the anomaly detection that achieves better detection accuracy with reduced false positive rate (FPR) values. Our framework is MEC-supported and run at MEC servers which are overseeing a group of base stations, as depicted in the system model in Figure 4.1 and described in Section 4.5. Each server executes our framework to ease off the core network's load.

This chapter makes the following prominent additions to the existing literature:

1. Employs a framework that takes advantage of various state-of-the-art DL techniques to achieve optimum performance. Framework is designed to be MEC-supported, which aids in easing off the CN from bulky computations by offloading them to MEC servers for efficient and robust anomaly detection.

2. Capitalizes on the past information (based on CDRs) to learn from the old traffic behavior and identifies anomaly in the recently-acquired 10-minute user activity (test instance). Previously, detection on 1-hour data was executed and hence this work presents faster anomaly detection.

3. Incorporates a surplus feature to achieve robustness by utilizing Internet activity, which was ignored in the past research.

4.3 Literature Survey

Identification of outages invoked by hardware misfunction is performed in [32, 35, 38, 74, 75] having catatonic cell in focus. While [33, 76] orient towards identifying RACH failure-based crippled cell. K-nearest neighbor (KNN) method was employed by Imran *et al.* [1] which yielded 94% accuracy. Masood *et al.* [59] utilized deep autoencoders for the anomaly detection. Their proposed method utilizes different KPIs (like reference signal received power (RSRP) and signal to interference plus noise ratio (SINR)) related to the serving and neighboring base stations. The KPI values are acquired through MDT functionality of the LTE network. The above-mentioned researches, however, only take into account the spatial information acquired for an individual time incident yielding immediate outage detection; consequently, this could lead towards a momentary result with a small imprint on QoS and may vanish quickly afterwards [6, Sec. IV C]. Besides outage, congestion detection is performed by Ramneek *et al.* [10] for the networks ensuring a guaranteed QoS to their clients.

In contrast, the following studies utilized CDRs instead of KPIs which makes their proposed solution lighter. Parwez *et al.* [7] leveraged CDR-based big data (Milan dataset) to detect network's abnormal conduct in terms of abrupt hike in traffic activity which can ignite congestion. K-means and Hierarchical (unsupervised) clustering algorithms were employed for this purpose and the proposed solution yielded 90% accuracy. However, time-efficiency of the proposed method is a major concern as it was utilizing past week data for the anomaly detection which might loose the practical utility. Ameliorating their work, our previous chapter's research published [62] also used CDR-based Milan dataset to detect traffic activity surges (apart from both crippled and catatonic-type cells) within an hour by employing semi-supervised statistical-based method. The proposed solution is more accurate (yielding 92% instead of 90% accuracy), lighter (since

Algorithm 2 Preprocessing of the Data

Inputs: ***CDRDataset***: comprising subscriber events (in raw form), captured every 10-minute interval and stored in (d =) 62 documents, each reflecting a single date.
CID: Chosen cell's identification.
TimeStampValues: Contains numeric values of the beginning of every 10-minute time interval (in Unix epoch) during the intended 3-hours range.
Output: $\boldsymbol{X}_{total}$
Methodology:
1: **for** d in ***CDRDataset***
2: Import and save the contents of document d in a matrix.
3: Substitute voids with 0.
4: Delete "Country code" column.
5: Delete all entries in the matrix except the ones linked with ***CID***.
6: Delete the ***CID*** column.
7: **for** timestamp t in ***TimeStampValues***
8: Add all received SMS values and save them as $SMSin$.
9: Add all sent SMS activity values and save them as $SMSout$.
10: Add all inbound call activity values and save them as $CALLin$.
11: Add all sent call activity values and save them as $CALLout$.
12: Add all Internet activity values and save them as $Internet$.
13: Save $Internet$, $CALLout$, $CALLin$, $SMSout$, and $SMSin$ as a single instance in x vector.
14: Save instance x as a column entry in matrix $\boldsymbol{X}_{total}$.
15: **end**
16: **end**
17: **return** $\boldsymbol{X}_{total}$.

it utilizes spatio-temporal data instead of KPIs), and faster (since it detects anomalies within an hour); however, it also yielded 14% false positive rate (FPR) indicating the wastage of precious network resources due to false alarms leading to heightened OPEX.

The proposed method in this chapter, in contrast to the above-discussed works, is based on data analytic procedure in which we incorporate past information (based on CDRs) having temporal characteristics for the detection of long—instead of instantaneous/short—term anomalies. It also applies latest DL techniques to achieve maximal accuracy and minimal FPR along-with detecting anomalies within a timestep duration (i.e. 10 minutes).

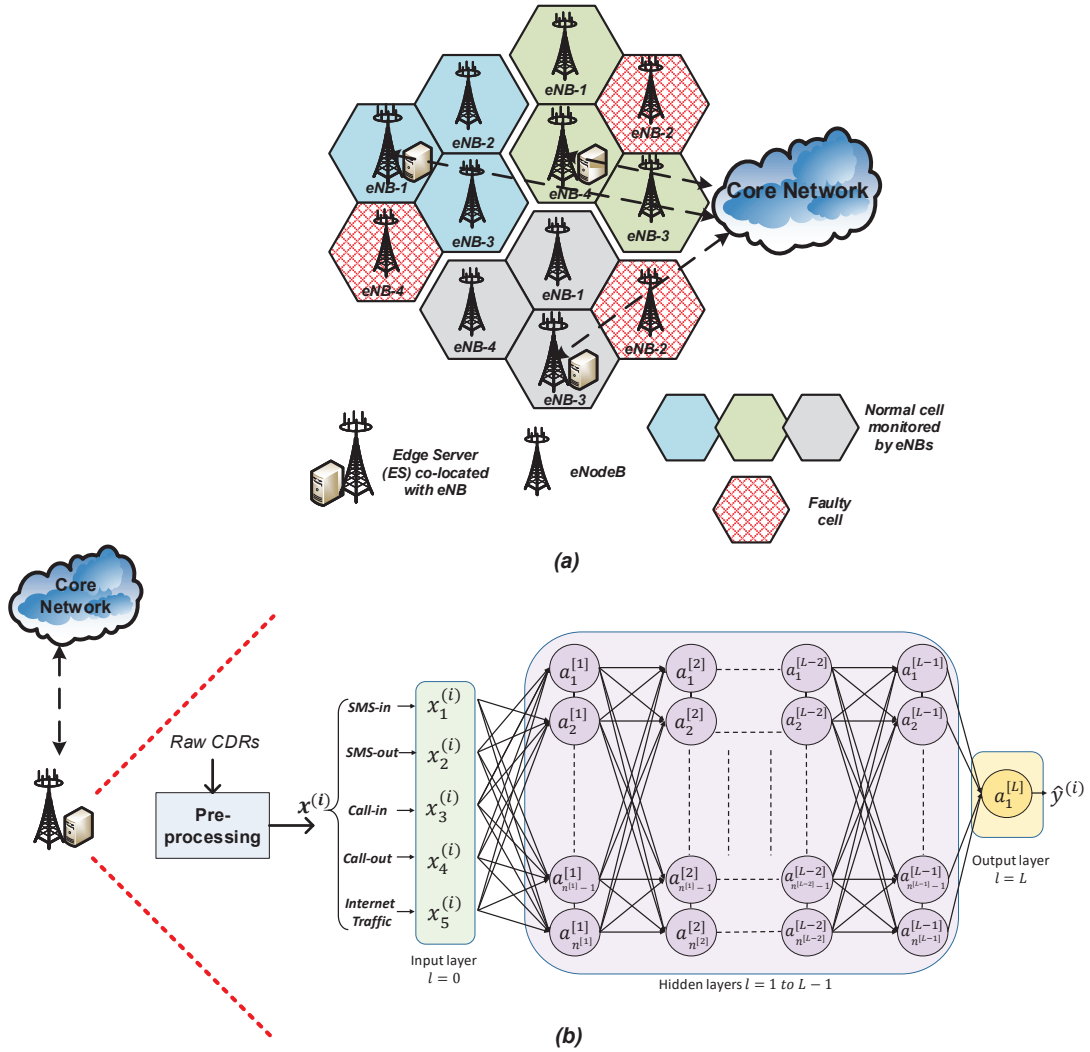

Figure 4.1: **(a)** System model for the proposed feed-forward DNN and MEC-based anomaly detection framework. **(b)** Functioning of the edge server.

4.4 Preliminaries

4.4.1 Data Preprocessing and Synthesis

In this research, we continue with Milan dataset. For each cell, day, and 10-minute timeslot in a 24-hour period; original CDRs are pre-processed to obtain attributes, which are then merged to form a vector $x^{(i)} \in \mathbb{R}^5$ (hereafter, denoted as an instance), with i as an index of the instance. The DL model demands a substantial number of instances to operate on (hundreds or even thousands),

which may equate to CDRs of over a year; but, we just have 62 instances (for each timeslot and corresponding to 62 days). For data augmentation and to get around this limitation, we treat all instances in a 3-hour period as the ones relating to a single 10-minute timeslot. By doing so, we have $1,116$ instances (62 days $\times$ 3 hours $\times$ 6 instances/hour) and we arrange them as a matrix $X_{total} \in \mathbb{R}^{5 \times 1,116}$.

Since human behaviors change throughout the day, choosing a single 3-hour range will limit the analysis of our findings to that time frame. As a result, we use 3 distinct ranges in our experimentations: morning, from 6 to 9 a.m.; midday, from 11 a.m. to 2 p.m.; and evening, from 5 to 8 p.m. We summarize the pre-processing procedure in Algorithm 2. For improving the algorithm's effectiveness and to get an identical distribution, we synchronously shuffle the instances X_{total} [84, Ch. 8]. They are then split into training and test sets with 781 (70% of the total) and 335 instances, respectively.

Since the deep neural network (DNN) used in this study is focused on supervised learning, labeled data is compulsory for training and testing. We synthesize the output label $y^{(i)} \in \mathbb{R}^1$ (for each instance in both sets) by utilizing Euclidean norm as it is absent in the dataset. We consider an instance $x^{(i)}$ in 5D Euclidean space and label its output $y^{(i)}$ as 1 (anomaly) if its norm $\|x^{(i)}\|_2$ departs beyond the norm of one standard deviation (SD) $\sigma_{SD} \in \mathbb{R}^5$ from the mean $\mu \in \mathbb{R}^5$: $\|\mu - \sigma_{SD}\|_2 > \|x^{(i)}\|_2 > \|\mu + \sigma_{SD}\|_2$; else 0 (cell functioning as per norm). Note that a larger SD implies that more points are included as normal and that there are less aberrant points; this may not function well for detecting performance divergences in a cell, so we select one SD. Using traditional statistics formulas, we can determine the components of mean and SD. For this reason, we use the train set. We create $Y_{train} \in \mathbb{R}^{1 \times 781}$ and $Y_{test} \in \mathbb{R}^{1 \times 335}$ matrices by arranging the labels from both sets.

4.4.2 Performance Metrics

We employ prediction *Accuracy, Error rate, FPR, F_1 score, Precision, and Recall* (elaborated in Section 2.5) for our model's performance analysis by exploiting expected test set $\hat{Y}_{test} \in \mathbb{R}^{1 \times 335}$ and actual test set Y_{test} labels.

4.4.3 Software

The preprocessing and results are generated by exploiting MATLAB and the complete DNN is actualized using Python (programming language). Experimentation is performed in a commercial PC (i7-7700T CPU, 16GB RAM, and Windows 10 64-bit operating system).

4.5 Implementation

In this section, we briefly discuss the implementation details of L-layer feedforward deep neural network (DNN), integrated in our anomaly detection framework and how it is trained for each individual cell—optimally tuned in terms of number of layers, number of units each hidden layer contains, weight initialization strategy, regularization, and optimization method to yield maximum performance. Once trained, the framework residing in the MEC server can utilize the DNN to detect anomalies in the testing phase: when CDRs arrive after every 10-min duration. The framework can occasionally re-train the network as the performance degrades over time.

4.5.1 Deep Learning Based Anomaly Detector

We apply L-layer feedforward DNN having an input layer $l = 0$, hidden layers from $l = 1$ to $L - 1$ and an output layer L, illustrated in Figure 4.1(b), where L represents number of (hidden and output) layers in the network. In contrast to

a shallow neural network (consists of one hidden layer), a DNN consists of two or more hidden layers. It also requires comparatively lesser (artificial) neurons or units for achieving the same amount of performance and effectively deals with more complex problems [85, 86]. Each layer has one or more units (represented by circles in the figure) that uses one of the following mathematically expressed non-linear activation functions to produce the output:

Sigmoid function:

$$g(z) = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{4.1}$$

Hyperbolic tangent (tanh) function:

$$g(z) = tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{4.2}$$

Rectified linear unit (ReLU) function:

$$g(z) = relu(z) = max(0, z) \tag{4.3}$$

Leaky ReLU (LReLU) function:

$$g(z) = max(0.01 \times z, z) \tag{4.4}$$

Swish function:

$$g(z) = z \times \sigma(z) \tag{4.5}$$

During gradient descent (GD) algorithm, the derivative of sigmoid and tanh functions becomes very small (≈ 0) for large positive and negative values of z, known as vanishing gradient problem. It causes slow optimization convergence [84]; thus, usage of sigmoid function is restricted to the output layer for single-label classification [87]. Tanh function typically works better as compared with sigmoid function for training a DNN, but ReLU function [88] is computa-

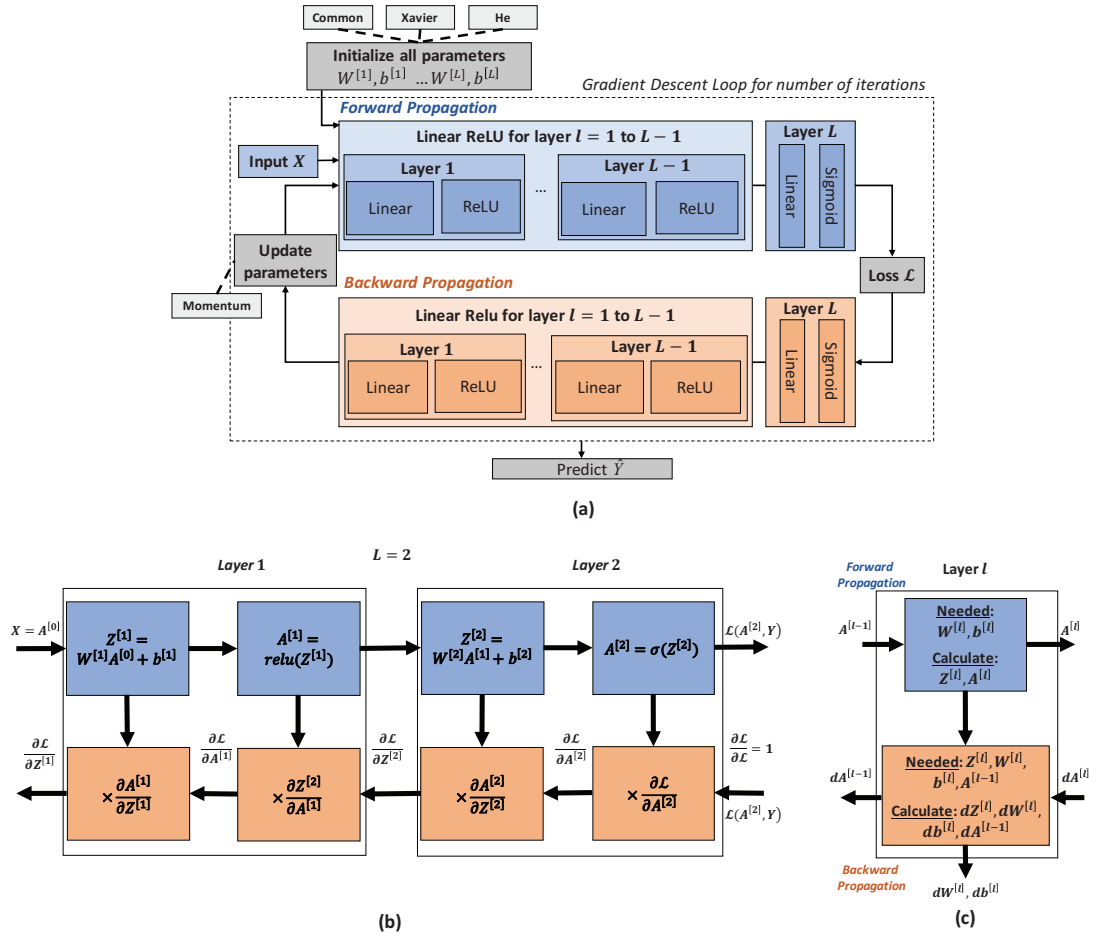

Figure 4.2: (a) Schema of the L-layer DNN model (b) Illustration of how gradient is computed using the chain rule of calculus in a simple 2-layer network (c) Single building block of a DNN.

tionally cheaper and yields equal or even better performance than tanh function. ReLU avoids vanishing gradient problem when it is activated above 0; however, the gradient becomes 0 when the unit is inactive which could lead to a situation when a unit never activates [89]. In contrast, Leaky ReLU [89] has a non-zero gradient over the entire domain which prevents it from the aforementioned problems. Swish—gated version of sigmoid activation function—is a new function, reported to yield better results as compared with ReLU [90]. Sigmoid function is utilized in the output layer and one of the aforementioned functions is applied in the hidden layers.

The model is shown schematically in Figure 4.2(a). In the training process, an

Table 4.1: Values of Hyperparameters for various optimization techniques.

Hyperparameters	**GD**	**Mini-batch GD**	**Mini-batch GD with momentum**	**Mini-batch GD with ADAM**
No. of Iterations/Epochs	1000	1000	1000	75-400[a]
Learning rate α	0.0075	0.0075	0.0075	0.0075
Initialization	He	He	He	He
Mini-batch size	781[b]	64	32	64
Momentum β	-	-	0.9	-
β_1[c]	-	-	-	0.9[d]
β_2[c]	-	-	-	0.999[d]

[a] Some cells converged in lesser number of epochs as compared with others.
[b] Full batch size of the training set.
[c] Controls the exponentially weighted averages in ADAM.
[d] Suggested default values [84, Ch. 8].

input matrix $X \in \mathbb{R}^{n_x \times m}$ containing m (= 781) training examples, each having n_x (= 5) features, is fed into the network along with all the parameters. The parameters are weight matrix $W^{[l]} \in \mathbb{R}^{n_h^{[l]} \times n_h^{[l-1]}}$ containing weights, and bias vector $b^{[l]} \in \mathbb{R}^{n_h^{[l]}}$ containing biases, where $n_h^{[l]}$ is the number of hidden units of layer l. The contained weights and biases relate to each neuron of each layer l. Weights are initialized to small random values, to break symmetry between the hidden units, and biases to zero [84, Ch. 8]. The information flows forward through the network, starting from the input and moving through the hidden layers until generating the loss $\mathcal{L}$: known as forward propagation (shown using blue blocks in Figure 4.2(a)) [84, Ch. 6]. Each layer l computes the following equations—vectorized over all the examples to avoid explicit for-loops in the code [91, Sec. "Vectorization"]:

$$Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]} \tag{4.6}$$

$$A^{[l]} = g^{[l]}(Z^{[l]}) \tag{4.7}$$

where, $Z^{[l]} \in \mathbb{R}^{n_h^{[l]} \times m}$ is a linear function (denoted by blue "Linear" blocks in Figure 4.2(a)), $A^{[l]} \in \mathbb{R}^{n_h^{[l]} \times m}$ is the activation function (denoted by blue "ReLU" blocks in Figure 4.2(a)), and $A^{[l-1]}$ is the previous layer's output starting with $A^{[0]} = X$. The above two equations are repeated $L-1$ times for layers $l = 1$ to $L-1$ that utilizes ReLU activation, followed by a last repetition for layer L in which the activation $g^{[L]}$ is a sigmoid function, depicted as blue "Sigmoid" block in Figure 4.2(a).

Cross-entropy cost J, which is an average of all the losses $\mathcal{L}$ (or errors) of

individual examples, is expressed as:

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) \\ = -\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}log(a^{[L](i)}) + (1-y^{(i)})log(1-a^{[L](i)})) \quad (4.8)$$

where, $\hat{y}^{(i)}, y^{(i)} \in \mathbb{R}$ are predicted and labeled outputs, respectively, for an example $x^{(i)}$.

After propagating forward and computing the loss, the information then flows through all the hidden layers backward starting from layer L to compute gradient of the loss function with respect to the parameters, known as backward propagation (shown using orange blocks in Figure 4.2(a)). The partial derivatives are derived using the chain rule of calculus, explicated in Figure 4.2(b) and are mathematically expressed below [84, Ch. 6], [91].

At layer $l = L$:

$$dA^{[L]} = \frac{\partial\mathcal{L}}{\partial A^{[L]}} = -\frac{Y}{A^{[L]}} + \frac{1-Y}{1-A^{[L]}} \quad (4.9)$$

$$dZ^{[L]} = \frac{\partial\mathcal{L}}{\partial Z^{[L]}} = A^{[L]} - Y \quad (4.10)$$

where, $Y \in \mathbb{R}^{1\times m}$ is labeled output vector and $A^{[L]} = \hat{Y} \in \mathbb{R}^{1\times m}$ is the predicted output vector. The post-activation gradient $dA^{[L]}$ and $dZ^{[L]}$ (computed by orange "Sigmoid" and "Linear" blocks, respectively, on the right side of Figure 4.2(a)) is then used to further propagate backwards. The remaining orange "Linear" and "ReLU" blocks compute the following equations for each corresponding layer [92]:
For l :

$$dZ^{[l]} = \frac{\partial\mathcal{L}}{\partial Z^{[l]}} = dA^{[l]} * g'^{[l]}\left(Z^{[l]}\right) \quad (4.11)$$

$$dW^{[l]} = \frac{\partial \mathcal{L}}{\partial W^{[l]}} = \frac{1}{m} dZ^{[l]} A^{[l-1]^T} \tag{4.12}$$

$$db^{[l]} = \frac{\partial \mathcal{L}}{\partial b^{[l]}} = \frac{1}{m} \sum_{i=1}^{m} dZ^{[l](i)} \tag{4.13}$$

$$dA^{[l-1]} = \frac{\partial \mathcal{L}}{\partial A^{[l-1]}} = W^{[l]^T} dZ^{[l]} \tag{4.14}$$

L-layer DNN can be perceived in terms of a sequence of blocks, each block representing a layer l comprising a forward and a backward propagation function. A single building block is illustrated in Figure 4.2(c), note that forward propagation function shares some variables with back propagation function to calculate the required gradient. GD algorithm then utilizes the gradient to perform learning by finding the optimal solution corresponding to a minimum cross-entropy cost J; achieved by iteratively updating the parameters as per following equations:

$$W^{[l]} = W^{[l]} - \alpha \; dW^{[l]} \tag{4.15}$$

$$b^{[l]} = b^{[l]} - \alpha \; db^{[l]} \tag{4.16}$$

where, α is the learning rate.

Once the parameters (weights and biases) are fine-tuned, the trained DNN uses forward propagation to predict the output $\hat{Y}_{test}$ by utilizing the test set. We empirically evaluated the impact of L and $n_h^{[l]}$ on the test accuracy of our DNN; L was varied from 2 to 20 while $n_h^{[l]}$ was varied from 1 to 50. GD was applied for this purpose with values of hyperparameters given in Table 4.1. We also experimented with different activations in hidden layers: sigmoid, tanh, ReLU, leaky ReLU, and Swish; and observed their effect on the DNN's performance in terms of error rate. For this purpose, L and $n_h^{[l]}$ were set to 17 and 25, respectively, with remaining

parameters same as before. The experiment was performed on the data of several cells and different hours.

4.5.2 Improving Performance of DNN

We leveraged different modern DL techniques in our framework, described below, to improve and render optimal performance.

Weight Initialization Methods

A problem in training a DNN occurs when the gradient explodes or vanishes, due to poor weight initialization, making learning difficult. We can remedy the situation by heedfully choosing an initialization strategy (mentioned in Figure 4.2(a)), so that the values of weights are neither too small nor too large [87, Ch. 6]. Weight initialization can strongly affect the performance of a DNN. A commonly used heuristic [93] is to set all the weights to normally distributed random numbers, centered at 0 and having a variance:

$$Var_{Common}(W^{[l]}) = \frac{1}{n^{[l-1]}} \tag{4.17}$$

where $n^{[l-1]}$ is the number of input units of layer l. Xavier initialization proposed in [93] yields better results for a DNN having tanh activation functions as compared with the previous strategy. Their derived variance of weights is:

$$Var_{Xavier}(W^{[l]}) = \frac{2}{n^{[l-1]} + n^{[l]}} \tag{4.18}$$

where $n^{[l]}$ is the number of output units of layer l. For ReLU activation function, He *et al.* [94] proposed the following variance:

$$Var_{He}(W^{[l]}) = \frac{2}{n^{[l-1]}} \tag{4.19}$$

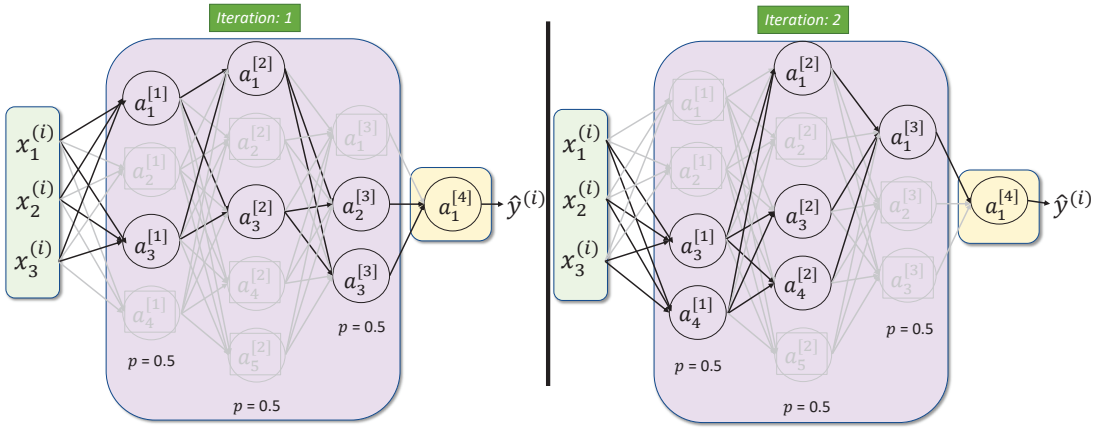

Figure 4.3: Demonstration of the Dropout technique using a 4-layer DNN.

We continued with our previous model configuration and utilized the above weight initialization strategies for several cells and hours.

Regularization

A fundamental challenge to DNN is of overfitting, in which the model performs well on training set but fails to generalize to new examples. Regularization, which refers to modification of the learning algorithm, is used to control overfitting and reduce the test error [84]. L^2 regularization, also known as weight decay, is the most common type of regularization. It penalizes the square values of the weights in the cost function in order to drive all the weights to smaller values. Smaller values lead to simpler hypotheses, which are most generalizable [87].

Dropout [95] is another regularization technique in which neurons (along with their connections) are randomly shut down during training of a DNN; and hence at each iteration, a different model is trained that uses only a subset of the total neutrons. The dropped neurons do not contribute to the training in both forward and backward propagations. A better generalization to an unseen data can be achieved as this technique prevents the network to have dependency on any particular neuron by making its presence unreliable [96]. Figure 4.3 demonstrates dropout mechanism using a 4-layer network (for simplicity), in which p is the

retention probability.

Our experiments embed the above-discussed regularization techniques in the DNN model.

Optimization Methods

Besides GD—also known as batch GD—we utilized mini-batch GD, which converges relatively faster and yields superior results [97]. It divides the training set (X, Y) into mini-batches $(X^{\{1\}}, Y^{\{1\}}), ..., (X^{\{t\}}, Y^{\{t\}})$, where {t} represents the index of a mini-batch. It is computationally efficient because it employs a single mini-batch at a time to compute gradient before performing an update step; as compared with GD which has to read an entire batch of training set [87]. Due to the inherent nature of mini-batch GD, the direction of the update has some variance and hence, the path taken by it oscillates towards convergence. To further accelerate learning and reduce oscillations, an update technique known as momentum [84, Ch. 8] (mentioned in Figure 4.2(a)) is used with mini-batch GD.

Momentum accumulates exponentially weighted moving averages of the previous gradients and continues to move in their direction. The method can be utilized by using Eq. 4.20 and 4.21 during the update rule instead of using Eq. 4.15 and 4.16.

$$W^{[l]} = W^{[l]} - \alpha \; v_{dW^{[l]}} \tag{4.20}$$

$$b^{[l]} = b^{[l]} - \alpha \; v_{db^{[l]}} \tag{4.21}$$

where, $v_{dW^{[l]}}$ and $v_{db^{[l]}}$ are used to store the past gradients' values and are given as follows:

$$v_{dW^{[l]}} = \beta v_{dW^{[l]}} + (1 - \beta) dW^{[l]} \tag{4.22}$$

$$v_{db^{[l]}} = \beta v_{db^{[l]}} + (1 - \beta) db^{[l]} \tag{4.23}$$

where, $\beta \in [0, 1)$ is the momentum which determines the number of past gradients that should be taken into account.

ADAM [98] is one of the most effective adaptive learning rate optimization algorithm for training a DNN that combines ideas from momentum (described in detail in [99]) and RMSProp (another optimization method for the details of which, readers can refer to [100]). ADAM uses the following update rule for weight $W^{[l]}$:

$$W^{[l]} = W^{[l]} - \alpha \frac{v^{corrected}_{dW^{[l]}}}{\sqrt{s^{corrected}_{dW^{[l]}} + \epsilon}} \tag{4.24}$$

where, $v^{corrected}_{dW^{[l]}}$ and $s^{corrected}_{dW^{[l]}}$ (given below) are bias corrections, of first moment and second raw moment estimates, respectively, to account for their zero initialization [84, Ch. 8], [98]; and ϵ is a small number added for numerical stability.

$$v^{corrected}_{dW^{[l]}} = \frac{v_{dW^{[l]}}}{1 - (\beta_1)^t} \tag{4.25}$$

$$s^{corrected}_{dW^{[l]}} = \frac{s_{dW^{[l]}}}{1 - (\beta_2)^t} \tag{4.26}$$

where, $v_{dW^{[l]}}$ and $s_{dW^{[l]}}$ (given below) are exponentially weighted moving averages of historical gradient and the squared gradient, respectively; t counts the steps carried by ADAM update; and $\beta_1, \beta_2 \in [0, 1)$ are hyperparameters that control the two averages.

$$v_{dW^{[l]}} = \beta_1 \, v_{dW^{[l]}} + (1 - \beta_1) \, dW^{[l]} \tag{4.27}$$

$$s_{dW^{[l]}} = \beta_2 \, s_{dW^{[l]}} + (1 - \beta_2) \, (dW^{[l]})^2 \tag{4.28}$$

The update rule for bias parameter $b^{[l]}$ is similar to the above rule. We imple-

Table 4.2: Comparison of mini-batch gradient descent (GD) with ADAM and Momentum.

Metric	**Average over** $1,000$ **cell IDs**		
	Momentum	**ADAM**	**Improvement**
Accuracy	90.44%	**98.8%**	8.36%
Error rate	9.55%	**1.19%**	8.36%
Precision	86.11%	**99.07%**	12.96%
Recall	84.54%	**97.27%**	12.73%
FPR	6.66%	**0.44%**	6.22%
F_1	85.32%	**98.16%**	12.84%

ment ADAM in our DNN model and compare its test performance (in terms of various metrics mentioned in Sec. 2.5) with gradient descent (GD), mini-batch GD, and momentum. For this purpose, the hyperparameter values mentioned in Table 4.1 are used, along with $\epsilon = 1 \times e^{-8}$ (suggested default value [84, Ch. 8]). Additionally, we investigate their training time.

4.6 Experimental Results and Performance Evaluation

We present various experimental results in this section. Although the CDR dataset contains records pertaining to $10,000$ cells, our DNN model performs anomaly detection for a single cell at a time. To demonstrate robustness and transferability of our model, we present results based (averaged) on randomly chosen $1,000$ cell IDs out of the total $10,000$ cell IDs (available in Milan dataset). In addition, we also present results processed by using a small subset (up to ten cell IDs) for a detailed analysis and comparison. Note that mentioning of morning, afternoon or evening followed by a cell ID indicates that the model is trained and tested on a corresponding 3-hours range data (discussed in Sec. 4.4.1).

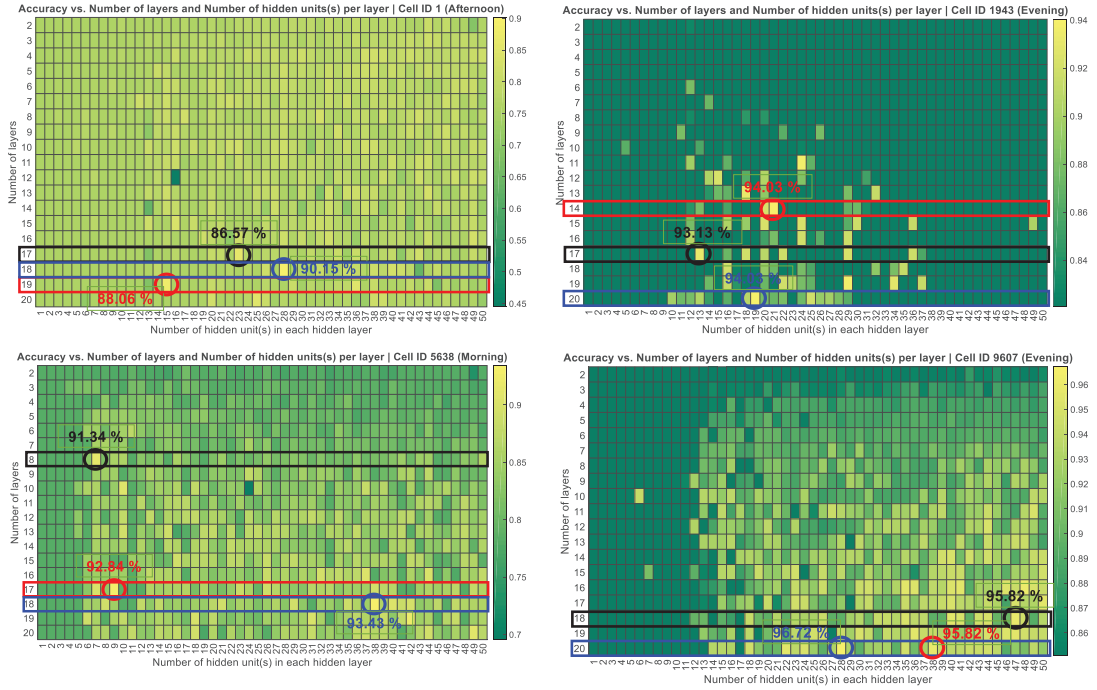

Figure 4.4: Effect of different configuration of number of layers and number of hidden units(s) per layer on test accuracies.

4.6.1 Number of Layers and Hidden Units

The performance of a DNN can vary across the spectrum of L and $n_h^{[l]}$. In practice, framework would search for their optimum values that yield maximum accuracy for each cell by empirically evaluating their impact on the test accuracy of our DNN. To demonstrate this, we vary L from 2 to 20 and $n_h^{[l]}$ from 1 to 50 using data from cell IDs 1 (Afternoon hours), 1943 (Evening hours), 5638 (Morning hours), and 9607 (Evening hours)—due to the inadequate space, we only show outcomes of these four randomly chosen cell IDs.

Our empirical results in the form of heatmaps, illustrated in Figure 4.4, elucidates the impact of various settings of $n_h^{[l]}$ and L on the test accuracy. We also highlighted three particular examples signifying maximum accuracies. It can be seen that deeper layer having moderate number of hidden units yield the highest accuracy. Dual maximum accuracies imply that one might be computationally efficient to attain than other. For simplicity, we set L and $n_h^{[l]}$ to 17 and 25,

respectively, for our further experiments (for all cell IDs).

4.6.2 Activation Functions

We run our model with mini-batch GD having hyperparameter values listed in Table 4.1, to find an activation function that yields maximum performance. Figure 4.5 (top) and (bottom) illustrates the effect of utilizing various activation functions in terms of error rate by using a subset of total cell IDs and $1,000$ cell IDs, respectively. We can clearly observe that sigmoid achieved the feeblest performance with highest error rate for most of the cell IDs in Figure 4.5 (top) while Swish also yielded overall poor performance that is evident in Figure 4.5 (bottom). Interestingly, for cell ID 2321, all the activations performed uniformly. Overall, ReLU surpassed other activation functions as evident in both of the figures and hence we choose ReLU for further experiments.

4.6.3 Weight Initializations

We continue with our previous model configuration and the randomly chosen cell IDs, and initialize weights according to Common, Xavier, and He initialization methods. We also set ReLU activation in hidden layers for this purpose, as discussed previously. Figure 4.6 exemplifies the impact of selecting various weight initialization schemes on DNN's test accuracy. We can observe that He surpassed other initialization strategies and yielded highest average accuracy.

4.6.4 Optimization Techniques

The superiority of mini-batch GD with momentum and ADAM over ordinary batch GD is clear in Figure 4.7. Although, in cell ID 4671, momentum has slightly better performance than ADAM but overall mini-batch GD with ADAM surpassed all other optimization techniques. It accomplished highest accuracy,

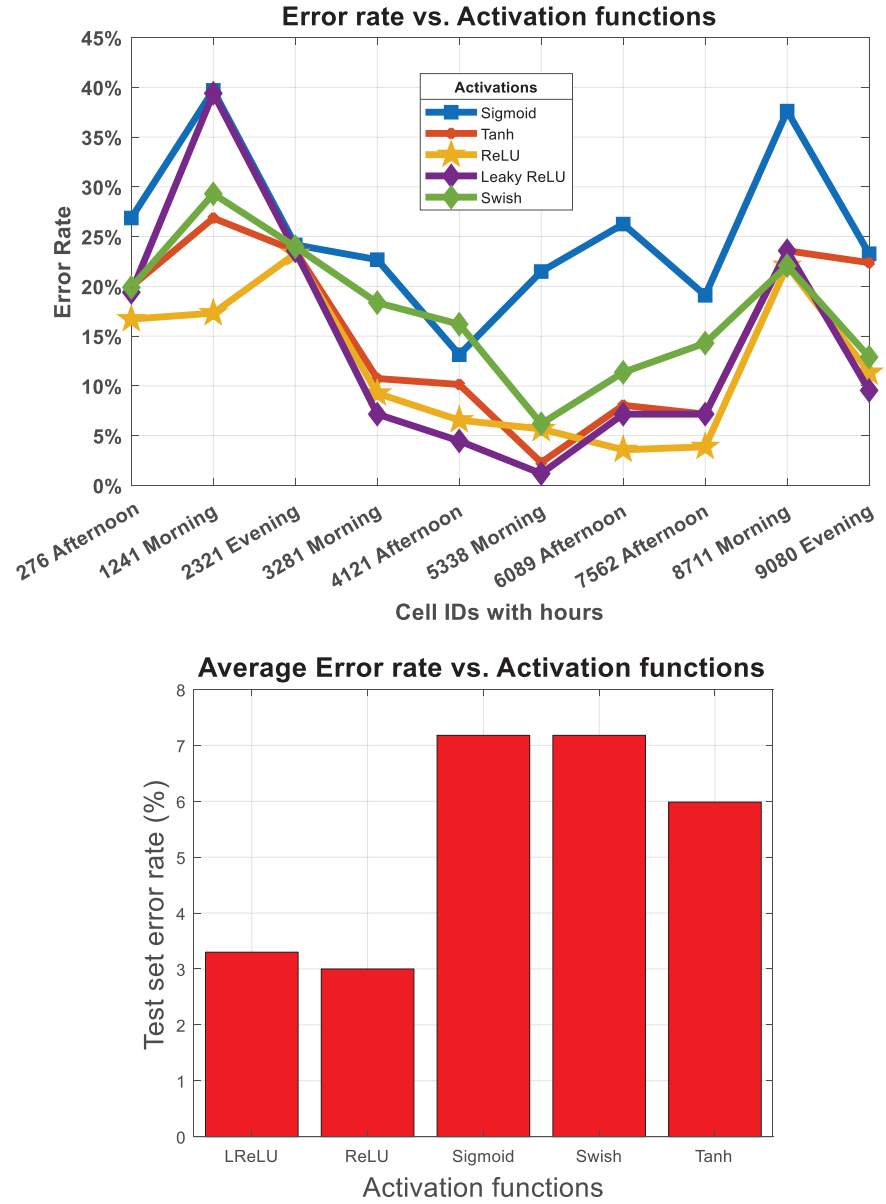

Figure 4.5: Effect of using different activations on performance.

recall, and F_1; and also, lowest error rate and FPR in most of the cells. Note, for cell ID 7816, ADAM achieved a perfect performance. In Table 4.2, we report various performance measures of our anomaly detector, averaged over the results from randomly selected $1,000$ cell IDs, along with the improvement we got by utilizing ADAM as compared with the momentum.

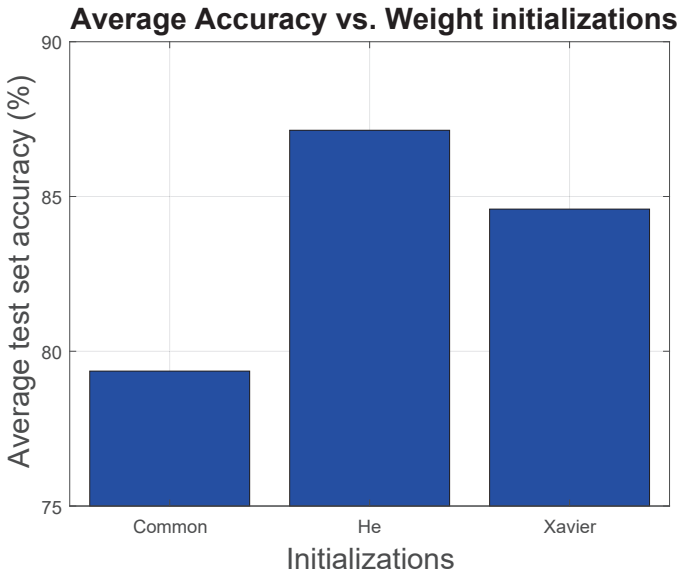

Figure 4.6: Effects of different weight initialization techniques on the performance.

4.6.5 Training Time

Another advantage of utilizing ADAM is faster training time that is evident in Figure 4.8 in which we compare the average training time of our model utilizing all the discussed optimization methods. Mini-batch GD with momentum consumes maximum training time, while ADAM deplete the lowest, and is the most suitable optimization method.

4.7 Conclusions and Insights for Future Work

Performance-wise, our MEC-based DL framework eclipsed the previous anomaly detection methods [1, 7, 62]. It can potentially improve network's QoS and user's QoE; and truncate OPEX for the network operators. Our proposed framework accomplished 0.44% FPR (Table 4.2), a significantly reduced value as compared with the reported 14% in [62]; and 98.8% accuracy, a great improvement as compared with the reported 94% accuracy in [1].

Our study endorses the concept of harnessing the largely untapped CDRs

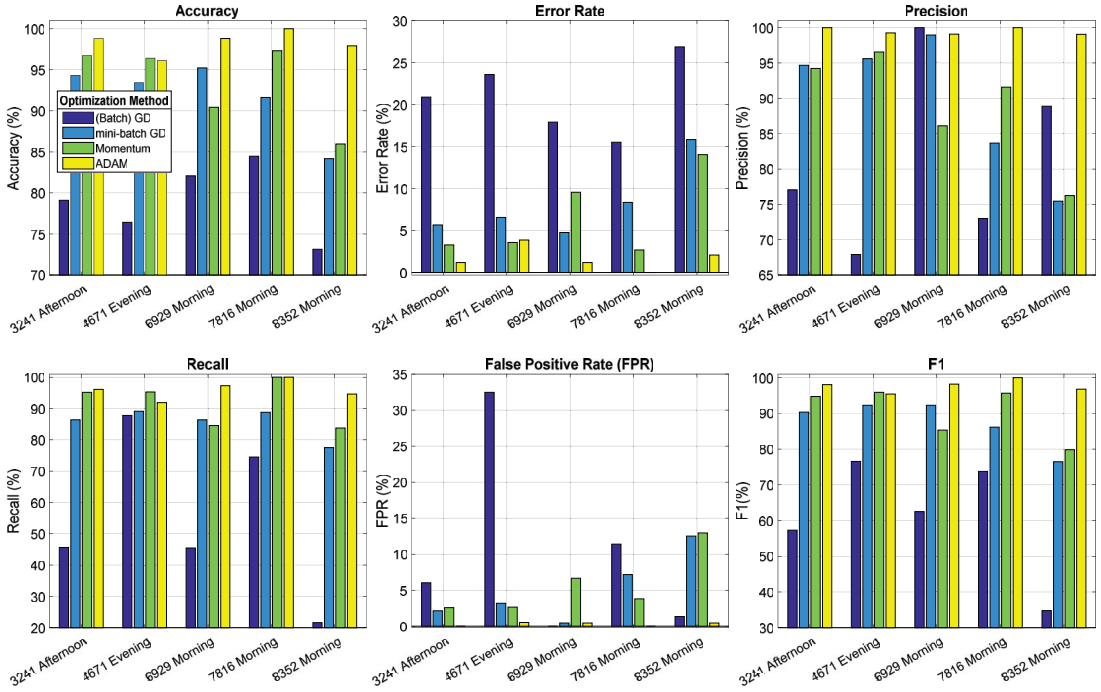

Figure 4.7: Effect of various optimization techniques on different performance measures.

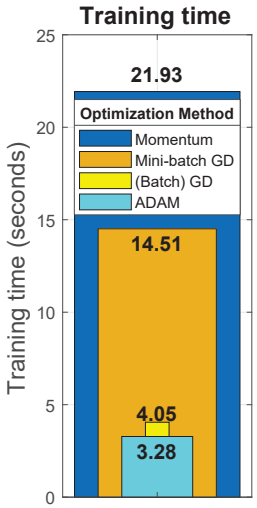

Figure 4.8: Comparison of training time of various optimization techniques.

(using big data analytics) instead of utilizing traditional measurements and analytical approaches for the network analysis [62, 101]. Our research's main innovation is the incorporation of the Internet activity feature (disregarded in previous works [7,62]) that makes our research more robust as our DL framework can detect anomalies pertaining to a situation when Internet activity swiftly rises/declines

but the call and SMS activities are normal. An example of such situation could be an abruptly increased Internet activity during a music festival inferring a necessity of additional network resource allotment. In addition, MEC-based approach reliefs core network from heavy computation tasks, offloaded to various MEC servers spread across the network.

A deterrent in practical implementation of our deep learning approach is the requirement of deluge of examples to extract a meaningful pattern in the CDR data; however, utilizing larger dataset—the acquisition of which is another issue due to privacy concerns—can surmount the difficulty. We can then preprocess the dataset using more sophisticated software: Apache Hadoop or Spark [62]. Another restraint on fully employing our approach is the possession of labeled data due to the supervised nature of our algorithm; affixing fault data, generated at the core network and containing historical alarms' logs [1], with CDRs and then labeling them accordingly can overcome this restraint.

The timestamp interval of 10 minutes is crucial for the results and hence more variation could be tested in the future studies to determine the impact of increasing the time duration granularity to perform more coarse-grained analysis, i.e. take three 10-min intervals instead of just one; or the granularity can also be decreased to perform more fine-grained analysis, i.e. by considering even smaller than a single 10-minute interval (the practical LTE network can be set to generate CDR dataset in such settings). Hence it will be an interesting future direction that could be explored. In this connection, our previous work in Chapter 3 considered a 1-hour interval instead of 10 minutes—we combined six 10-minute timestamp activities—and detected anomalies in the 1-hour user activity data by using semi-supervised machine learning method. In the current research work, we however chose to decrease the interval so that the anomaly detection could be performed quickly and hence the remedial or diagnostic actions could be taken sooner.

Because of the potential of upcoming cellular networks to have an AI-empowerment, the implemented algorithms need to be quicker, increasingly proficient and less perplexing: future works can explore meliorative methods. We can also extend our study for anomaly detection in Internet of things (IoT) [80]; however, due to the limited resources (such as power consumption) the IoT devices might have entirely different activity pattern that will need more examination. With rising fame of DL technology, which has an enormous potential for utility in 5G networks, our work applies DL to accomplish substantial performance betterments for abnormality detection. This indicates reduction in OPEX for cellular operators along with an improvement in the network's QoS and user's QoE.

Chapter 5

Deep Convolutional Neural Network and Mobile Edge Computing-Based Cell Outage and Congestion Detection

5.1 Motivation

In the previous chapter, we introduced MEC paradigm for anomaly detection and presented a feed-forward DNN-based framework which was executed at an MEC server collocated with a based station that oversaw a number of base stations. Although the aim to introduce MEC, in which computations are offloaded from the core network (CN) to the edge servers, was achieved; however, each server has to execute the framework separately for each connected base station under the proposed setting.

40–50 is the range of base stations per square kilometers estimated for 5G network [102]. For instance, city of Milan could demand $7,270 - 9,088$ base stations to fully cover its region, spread over 181.76 km^2. If previously proposed solu-

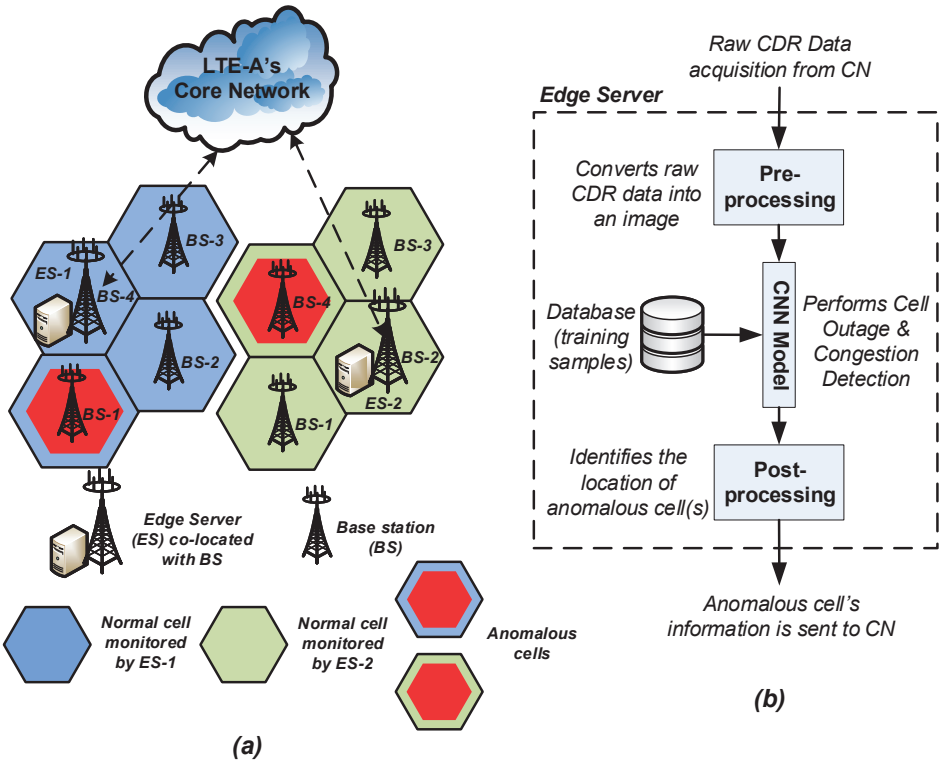

Figure 5.1: **(a)** System model for the proposed deep CNN and MEC-based anomaly detection framework. **(b)** Functioning of the edge server.

tion is applied for such a gigantic amount of base stations for anomaly detection, the cellular system could be computationally choked. In addition, the biggest drawback of executing a feed-forward DNN is the necessity for rich resources in terms of computation, storage and power—since every block in a given layer of a DNN is fundamentally linked to all of the previous layer's blocks, which require processing and storing copious parameters.

This chapter harnesses deep convolutional neural networks (CNN) to overcome the above limitations and enable anomaly detection for all the base stations connected to the MEC server, at once. The decision of choosing deep CNN is further elaborated in Section 5.5.2.

5.2 Overview and Contributions

This chapter takes inspiration from the milestones achieved by the CNNs in computer vision domain [81] and MEC, and proposes an innovative anomaly detection framework which alleviates the CN from heavy computations while consuming lesser resources in contrast to the feed-forward DNN (applied in the previous chapter). Rather than centralized processing where user activities belonging to all the base stations are processed at the CN, this research takes advantage of MEC paradigm in which heavy computational works are split among edge servers (ESs) spread across the network. The servers are co-located with the base stations and oversee a small group of base stations, as shown in Figure 5.1. They are AI-empowered and execute CDR (Trentino grid)-based data analytics: contrary to the previous work in Chapter 4 which employed classic feed-forward DNN, this work employs CNNs which are far more efficacious (further discussions in Section 5.5.1). Identification of abnormal cell(s) is then sent from the servers to the core network for further intervention. In case of a cell outage, self-healing functions (like diagnosis and compensation) [6] are invoked and in case of spiked activity, congestion-avoidance procedures are put into action. Similar to the MEC paradigm having cloud server as a centralized entity for processing and computations, and MEC server promoting a decentralized design for connectivity, storage, and computation; we can overall associate our framework with the paradigm in which the core network equates to the cloud server and the ES equates to the MEC server [82].

This chapter makes the following prominent additions to the existing literature:

1. Presents an innovative framework fully compatible with the MEC paradigm because of utilization of deep CNN models.

2. Utilizes state-of-the-art deep residual network (a type of CNN) for enhanced

performance in contrast to an elementary CNN model. The chapter also presents comparative analysis of these models with training time and performance in focus.

3. Presents additional experimentations and insights about the expansion of the CNN models from detecting anomalies in 100 cells at a time to 225.

5.3 Relevant Work

In this segment, we explore cell outage detection (COD) as well as congestion detection research that focuses on the utility of deep learning (DL) technology. For an extensive literature review on COD, the readers can consult [6] which is split into complete and partial CODs, each focusing on works that utilize: Heuristic (solutions based on pre-defined rules dictated by experts) and learning-based (solutions based on ML) methods. Additionally, Kline *et al.* [73] covers COD-related works that use machine learning approaches. However, neither [73] nor [6] contain any researches that use DL technologies for COD.

Our work in Chapter 4 (published in [3, 99]) suggested a method for detecting abnormalities in a single cell of a cellular network using feed-forward DNN. It pre-processes activity data to generate a 5D vector (each dimension of which corresponds to a single user activity of the designated base station), accepted as the input. The binary output 0 indicates a normal condition while 1 indicates an abnormality. However, because of the reasons described in the previous section, the solution is computationally costly when extended to the entire network. Masood *et al.* [59] proposed a sleeping cell detector based on a deep autoencoder (a form of feed-forward DNN) that uses simulator-generated user device data based on minimize drive test (MDT) measurements. The data consists of adjacent and serving BSs' reference signal received power (RSRP) and signal to interference

plus noise ratio (SINR) values. The model was trained using data from a typical operation using seven macro cells, and it was tested using the data from an outage situation. The main flaw in their method, as stated in [6, Sec. IV C], is that they only took into account spatial data gathered for a single case, which results in instantaneous identification of sleeping cells. As a result, the observed anomaly may be transient, with no effect on QoS, and may disappear until it is compensated.

Both, our previous work and Masood *et al.* [59] reported that their DL-based anomaly detection methods outperformed traditional ML approaches: semi-supervised statistical-based detection [62] and one-class support vector machine-based detection, respectively. Our preference for deep learning models over standard machine learning models stems from this rationale.

As part of their paper, Ramneek *et al.* [10] proposed an empirical solution for congestion detection in QoS-enabled networks. The key concept is to assess the congestion level by monitoring network load using information derived from the QoS-based scheduler. Parwez *et al.* [7] suggested a method for identifying region of interests (ROIs) as anomalies of extraordinarily high consumer traffic behavior using big-data (CDR) analytics and ML algorithms. Their method is inefficient for applications that need immediate identification since they examined CDRs for one week. To overcome this constraint and based on the fact that such ROIs will have congestion if suitable steps are delayed, our work in Chapter 3 (published in [62]) suggested a semi-supervised ML algorithm to detect surged user traffic in a cell's recent one hour CDR data by evaluating its past subscriber activity behavior. We also suggested a DL method for the identification of such ROIs in our subsequent work in Chapter 4 (published in [99]), which reduced the detection time from one hour to ten minutes while also improving the efficiency.

In comparison to all of the previous works, our methodology is unique in that it uses deep CNNs rather than feed-forward DNNs and a MEC-based structure

to distribute the computational load of the CN through many ESs, resulting in a lighter solution for anomaly detection. Our strategy is agile because we use already-available (CDR) data rather than requiring new KPI-based data [62]. It senses irregularities (such as outages and spiked traffic activity that could cause congestion) in several cells at once. Our method takes into account both spatial and temporal aspects, allowing us to track long-term outages rather than the instant ones.

5.4 Preliminaries

5.4.1 System Model

The system model is shown in Figure 5.1(a). For this work, we consider Trentino grid, described in Sec. 2.4.3. The main idea is to divide a network into regions called sub-grids, each consisting 100 cells and an edge server (ES) co-located with one of the BSs. The ES is equipped with our proposed anomaly detection framework that mainly handles preprocessing and comprises a deep CNN model. For every subsequent 10-min duration, the following process (illustrated in Figure 5.1(b)) executes:

1. ES acquires raw CDR data of each cell in its sub-grid from the CN;
2. Framework pre-processes the data to construct a grid-image that is acceptable as an input by the deep CNN model;
3. Model trains on a dataset (available in the attached database) containing past user behavior of the cells and detects anomalous cell(s) in the current example;
4. ES passes information of the faulty cell(s) to the CN that further takes curative actions.

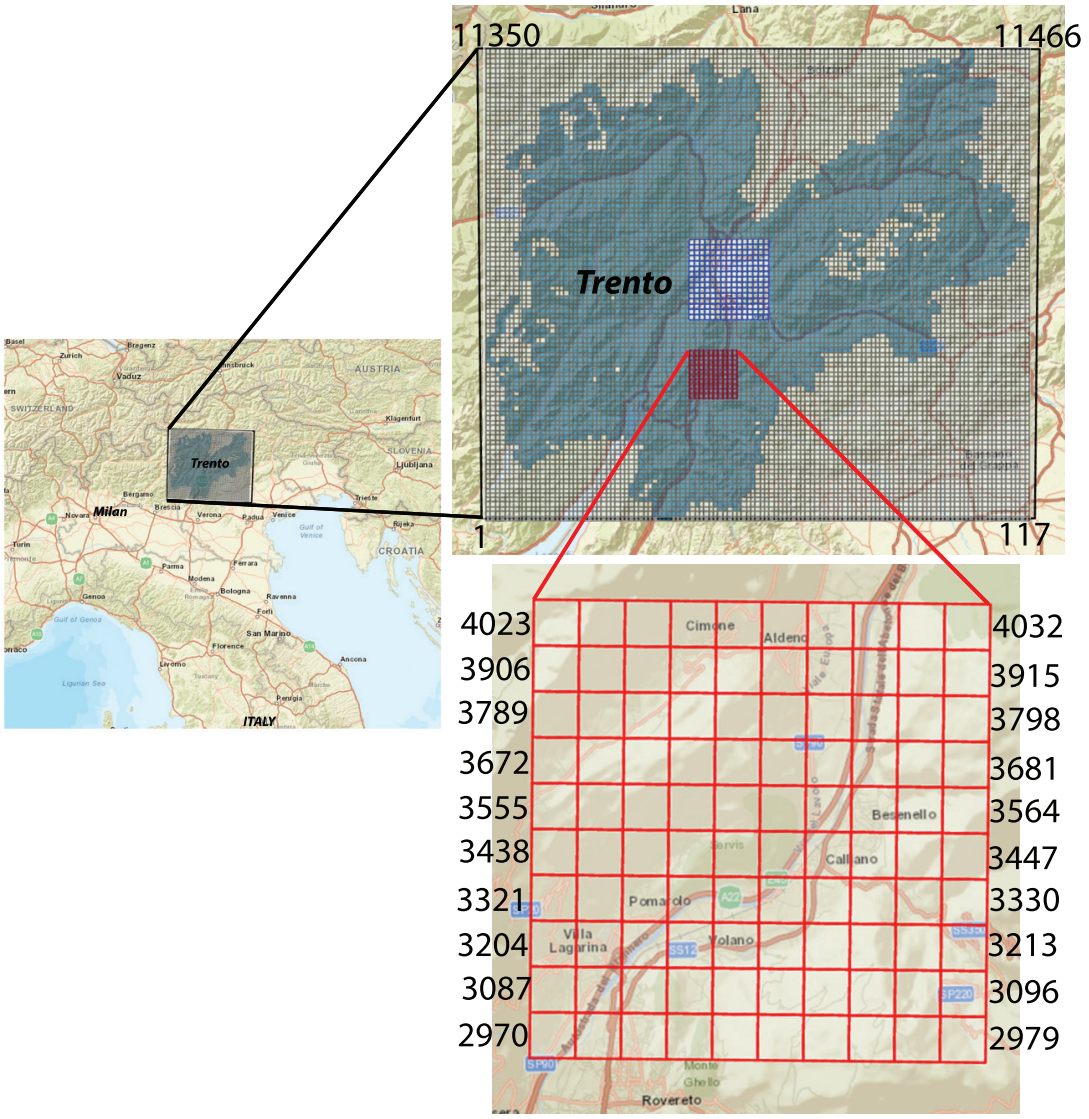

Figure 5.2: Trentino dataset's spatial description. 10×10 subgrid (red) is chosen for our experiments while 15×15 sub-grid (blue) is chosen to demonstrate the scalability of our proposed method.

5.4.2 Data Preprocessing and Synthesis

CNN processes grid-like data such as a time-series or an image [84, Ch. 9]. In preprocessing stage, we convert raw CDRs into a 10×10×5 3D matrix $x^{(i)} \in \mathbb{R}^{n_H^{[0]} \times n_W^{[0]} \times n_C^{[0]}}$ henceforth referred as "grid-image", where i is the index, $n_H^{[0]}$ is the height, $n_W^{[0]}$ is the width, and $n_C^{[0]}$ is the number of channels of the grid-image. The height and width make up 100 entries representing cells chosen from the bottom

portion of the Trentino grid, illustrated as red squares in Figure 5.2. The channels comprise 5 feature (subscriber activity) values of the selected cells: Call incoming, SMS incoming, call outgoing, SMS outgoing, and Internet usage. Hence, each pixel of the grid-image contains the above activity values of a corresponding cell, recorded during a 10-min duration. In order to excavate meaningful pattern in the dataset, an avalanche of examples each representing past instances are required; however, only 62 instances are available in the current dataset for each time-resolution. To remedy this, we combine timestamps for a 3-hour duration and generate 1,116 grid-images (6 timestamps per hour $\times$ 3 hours $\times$ 62 days), represented as a 4D matrix $X_{total} \in \mathbb{R}^{m \times n_H^{[0]} \times n_W^{[0]} \times n_C^{[0]}}$, where m is the total number of grid-images.

Since we are dealing with supervised learning and have unlabeled data, we generate labels $Y_{total} \in \mathbb{R}^{m \times 100}$ on the basis of euclidean distance, where 100 represents the total number of output classes (each denoting a cell). An output class indicating 1 means an anomaly and the corresponding cell is faulty, and 0 means the corresponding cell's operation is normal. For each output class, we mark 1 if $\|\mu - \sigma\|_2 > \|a\|_2 > \|\mu + \sigma\|_2$, where $a \in \mathbb{R}^5$ represents the corresponding cell's activity. The elements of mean $\mu \in \mathbb{R}^5$ and standard deviation $\sigma \in \mathbb{R}^5$ can be calculated using standard textbook equations (Eq. 3.1, 3.2).

5.4.3 Shuffling and Splitting the Data

The order of X_{total} and Y_{total} is synchronously shuffled to make the algorithm more effective since it is using mini-batches (a subset of the entire dataset). The mini-batches enable the optimization algorithm (mini-batch gradient descent) to rapidly compute approximate gradient estimates instead of computing exact gradient, making the algorithm converge faster [84, Ch. 8]. The shuffled dataset is then split into training and test sets according to a ratio of 7:3, each comprising

781 and 335 grid-images with labels, respectively.

5.4.4 Performance Metrics

For the performance evaluation of our framework, we utilized the following common metrics of machine learning literature: precision, recall, accuracy, error rate, false positive rate (FPR), and F_1. They a re elaborated in Section 2.5.

5.4.5 Software

MATLAB was exploited for preprocessing, GPS mapping, and results generation. Keras [103] was also utilized to actualize the CNN models. Experimentation was performed in a commercial PC (i7-7700T CPU, 16GB RAM, and Windows 10 64-bit operating system) with an in-built GPU (NVIDIA GeForce 930MX).

5.5 Implementation of Anomaly Detector

In this section, we describe generic architecture of the CNN followed by a discussion on how it fits in with our research, the architecture's utility in building a relatively simple deep CNN model and lastly, we describe the ResNet-50 model.

5.5.1 CNN's Generic Architecture

CNN [84, Ch. 9] has the following three fundamental layers, as can also be found in Figure 5.3(a):

Convolution layer

It accepts an input volume (or activations of previous layer) $A^{[l-1]} \in \mathbb{R}^{m \times n_H^{[l-1]} \times n_W^{[l-1]} \times n_C^{[l-1]}}$, where l represents number of the current layer; and filters $F^{[l]} \in \mathbb{R}^{f^{[l]} \times f^{[l]} \times n_C^{[l-1]} \times n_C^{[l]}}$, where $f^{[l]}$ is the filter size, $f^{[l]} \times f^{[l]} \times n_C^{[l-1]}$ is the dimension of a single filter and $n_C^{[l]}$

Figure 5.3: Architecture of **(a)** Simple model with the red box highlighting pooling function deliniated in Figure 5.4 and **(b)** residual network model with 50 layers (ResNet-50). **(c)** Conv and ID modules of ResNet-50 model.

is the total number of filters. The convolution layer performs parallel convolution operations between input volume and each filter, adds bias, applies a rectified linear unit (ReLU) [84, Sec. 6.3] function and lastly, stack up each result to form an output $A^{[l]} \in \mathbb{R}^{m \times n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}}$. The height $n_H^{[l]}$ can be calculated as:

$$n_H^{[l]} = \lfloor \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \rfloor \tag{5.1}$$

where, $p^{[l]}$ is the number of padding and $s^{[l]}$ is the stride. Padding is a technique to add zeros around the border of the input image to prevent the height and width from shrinking, as output dimension reduces due to convolution operation. Stride is the distance between successive utilization of filter on the input volume. Formula for width $n_W^{[l]}$ can be written by replacing $n_H^{[l-1]}$ with $n_W^{[l-1]}$ in Eq. 5.1.

Pooling layer

It improves computational efficiency, reduces requirement for storing parameters and adds robustness to some of the detected features [84, Sec. 9.3]. Max function is commonly utilized in pooling layers that pools maximum numbers from regions of input volume (and from each channel, independently) depending on the filter size f, to generate the output volume. If the dimension of input volume is $n_H \times n_W \times n_C$, the dimension of output volume can be derived using Eq. 5.1 with $p = 0$ as $\lfloor \frac{n_H - f}{s} + 1 \rfloor \times \lfloor \frac{n_W - f}{s} + 1 \rfloor \times n_C$.

As an example, we consider $MaxPool1$ layer, illustrated in Figure 5.4(a). The pooling layer accepts an input volume having $13 \times 13 \times 8$ dimension and results a volume of $6 \times 6 \times 8$ dimension—the height and width is calculated by using Eq. 5.1. The layer utilizes following hyperparameters: filter size $f = 2$ and stride $s = 2$. This combination of hyperparameter values is common and it shrinks the input's size by a factor of 2. For simplicity, we demonstrate the max pooling operation in a single channel, illustrated in Figure 5.4(b). The layer slides a (f, f)

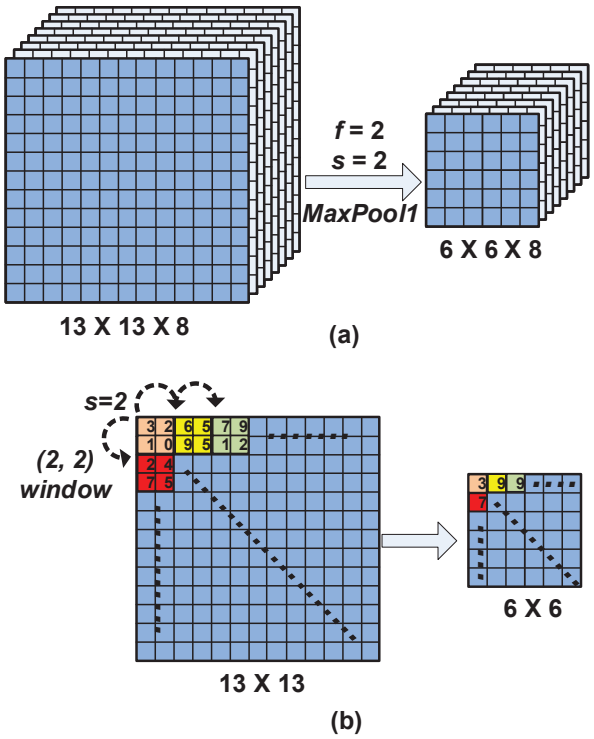

Figure 5.4: Max-Pooling Layer's functioning.

window over input and stores the maximum value of the window in the output. It performs the same operation for each channel and finally stacks the results to form the output volume.

Fully connected layer

It functions like the hidden layer of a feed-forward neural network (described thoroughly in the previous chapter), in which each hidden unit is connected to all hidden units of the previous layer.

5.5.2 Why Choose CNN?

Parameter sharing and sparse interactions [84, Sec. 9.2] are the main reasons for CNN's popularity and dramatic increase in computational efficiency as compared with feed-forward neural networks; because these result in lesser pa-

rameters to compute and store. For example, consider a convolution layer $Conv1$ in Figure 5.3(a) having an input volume of dimensions $14 \times 14 \times 5$, a filter size $f = 2$, and 8 number of filters. Using Eq. 5.1 with $p = 0$ and aforementioned values, we can calculate the dimension of output volume: $13 \times 13 \times 8$. The total number of parameters utilized in this (single convolution layer) operation is 40: $2 * 2$(for one filter) $+1$(for bias) $= 5$ parameters per filter and 40 parameters for 8 filters. However, if this was a feed-forward neural network, the input would be 980 units (flatten version of the input volume: $14 * 14 * 5$), the output would be 1352 units ($13 * 13 * 8$), and the total number of required parameters would be 1.32 million ($980 * 1352$). CNN is hence faster and require lesser resources (computation and storage). Due to the mentioned benefits and the fact that we are dealing with grid-like data (of 100 cells), CNN is our natural choice.

5.5.3 Simple CNN Model

Many models available today have put together the building blocks in different settings (in terms of number of layers and the approach of connecting them together) to form a CNN. LeNet-5 [92], AlexNet [104] and VGG [105] are some of the classical CNN models; while ResNet [106] and Inception-v4 [107] represent some modern ones (readers can refer to [103, Sec. Applications] for an exhaustive list of modern CNN models).

Our first approach, illustrated in Figure 5.3(a), is inspired from works of the aforementioned classical models, in which we utilize the building blocks in addition to batch normalization (thoroughly explained in the next paragraph) to detect anomalies. Our model accepts a $10 \times 10 \times 5$ grid-image as an input. It then pads zero along the edges (zero-padding) with $p = 2$ and passes the volume to a series of convolution and pooling layers ($Conv1$, $MaxPool1$, $Conv2$, $MaxPool2$, $Conv3$, $MaxPool3$). The dimension of output volume of each layer

can be computed by utilizing Eq. 5.1. The resultant volume is finally flattened and passed through two fully connected layers ($FC1$ and $FC2$). Finally, we utilize binary cross entropy loss function for a multi-labeled output as each class is not mutually exclusive.

Batch normalization (BN) [108] is a powerful technique of adaptive re-parametrization, used to accelerate training process and make DNN more robust. Training a DNN leads to a problem of covariance shift: distribution of earlier layers' parameters shifts, that affects the later layer's capability to adopt accordingly and results in a slow training process. Instead of just normalizing the input features values of the network, the technique normalizes the activations of each hidden layer. It makes the deeper layers' parameters more robust to changes, to earlier layers' parameters; hence, enhancing the network's stability [84, Sec. 8.7], [108]. Readers can refer to [109] for more detailed analysis on BN. We apply BN after the convolution operation and before utilizing the activation function. Therefore in Figure 5.3(a), each convolution layer incorporates BN in addition to convolution operation and ReLU activation.

5.5.4 Residual Network Model

To enhance the performance of our framework, we utilized residual network comprising 50 layers (ResNet-50), as shown in Figure 5.3(b). Depth of a neural network plays a crucial role in accurately representing more complex functions and in raising the overall network's performance [105]. However, deeper networks are harder to train as they suffer from gradient vanishing and exploding problems [84] that hinder with the convergence of the network, making it unbearably slow. Deeper networks also suffer from a degradation problem: as we add more layers the accuracy saturates and then quickly reduces, leading to an elevated training error [106].

Residual network [106] effectively deals with these problems by stacking residual modules on top of one another, shown as $Phase\ 2-5$ in Figure 5.3(b). We first elaborate functioning of a residual module used in the residual networks by using ID module of Figure 5.3(c). In the figure, the information flows from input $a^{[l]}$ to the output activation $a^{[l+3]}$ through two unique paths. The downward path, called main path, has three parts. The information first goes via initial part consisting three blocks having a convolution layer, BN, and a non-linear activation function, respectively; governed by the following standard equations:

$$z^{[l+1]} = W^{[l+1]}a^{[l]} + b^{[l+1]} \tag{5.2}$$

$$a^{[l+1]} = g(z^{[l+1]}) \tag{5.3}$$

where, $W^{[l+1]}$ is the weight matrix, $b^{[l]}$ is the bias vector, $g(.)$ is the non-linear activation function, $a^{[l]}$ is the input, and $a^{[l+1]}$ is the output of the initial part. The BN is utilized throughout the model to boost up the training.

Similarly, the blocks in the third part are governed by the following equations (ignoring the other path and a summation operation):

$$z^{[l+3]} = W^{[l+3]}a^{[l+2]} + b^{[l+3]} \tag{5.4}$$

$$a^{[l+3]} = g(z^{[l+3]}) \tag{5.5}$$

In residual networks, $a^{[l]}$ is fast-forwarded to a deeper hidden layer in the neural network where it is summed up with the output of that layer before applying a non-linear activation function. This is known as a skip connection, as shown in the figure. Hence, Eq. 5.5 will be altered as follows:

$$a^{[l+3]} = g(z^{[l+3]} + a^{[l]}) \tag{5.6}$$

The addition of $a^{[l]}$ makes it a residual module and this enables the activations of one layer to skip some layers and be directly fed to a deeper layer. This also allows a gradient (during back-propagation) to be directly back-propagated to an earlier layer. Here, we are assuming that the dimensions of both, input $a^{[l]}$ and $z^{[l+3]}$ (and therefore output $a^{[l+3]}$) are same in order to perform the summation. This kind of residual module is known as identity (ID) module.

If the dimensions of input ($a^{[l]}$) and output activations ($a^{[l+3]}$) mismatch then a convolution layer in the skip connection is introduced to adjust the input $a^{[l]}$ to a different dimension, so that the dimensions match up in the final summation. This type of residual module is called Convolutional (Conv) module, illustrated in the Figure 5.3(c)(left).

Moreover, we can now analyze the residual network architecture with 50 layers depicted in Figure 5.3(b). As an example, we can concentrate on the parts starting from the input to $Phase2$ of the architecture. In the following, we will discuss in term of dimensions so that the purpose of ID and Conv modules can be explained subsequently; and Eq. 5.1 is extensively utilized in computing the output dimensions of various layers. The input grid-image having dimension $10 \times 10 \times 5$ is zero-padded with padding $p = 5$ to have an output volume with dimension $20 \times 20 \times 5$. It is then passed to $Phase1$ comprising a convolution layer with filter size $f = 7$, total number of filters $n_C = 64$, and stride $s = 2$; that transforms the dimension to $7 \times 7 \times 64$. Lastly, Max Pool having $f = 3$ and $s = 2$ generates the output volume with dimension $3 \times 3 \times 64$.

For $Phase2$, let's focus on Figure 5.3 (c)(left) having Conv module that will have an input dimension of $3 \times 3 \times 64$ from the earlier layer. The main path contains 3 parts. The initial part has convolution layer having $f = 1$, $n_C = F_1 = 64$ (see Table 5.1), and $s = 1$. It yields volume with identical dimensions as of the input's. The convolution layer in the second part also results output with same dimension as of the input's because it is utilizing "same" convolution

Table 5.1: Utilized ResNet-50 model's hyperparameters

Phase	**Number of filters used in the layers [F_1, F_2, F_3] of each module**	**Stride s**
2	[64, 64, 256]	1
3	[128, 128, 512]	2
4	[256, 256, 1024]	2
5	[512, 512, 2048]	2

(in which padding is set so that the output's dimension remains same as of the input's). The third part having a convolution layer with $f = 1$, $n_C = F_3 = 256$ (see Table 5.1), and $s = 1$ will convert the input's dimension from $3 \times 3 \times 64$ to $3 \times 3 \times 256$. Finally, convolution layer in the skip connection, that has input volume of dimension $3 \times 3 \times 64$, scales up the input's dimension to $3 \times 3 \times 256$ by utilizing the parameter values: $f = 1$, $n_C = F_3 = 256$ (see Table 5.1), and $s = 1$. The outputs from both convolution layers (one in the skip connection and the other in third part of the main path) can be added as they are now compatible: have the same dimensions.

The ID modules of *Phase*2 have similar function as of the aforementioned Conv module, with the exception of the skip connection's design that does not has any layer in it. This is because the input of the ID modules has same dimension as of the output of convolution layer in it's third part: $3 \times 3 \times 256$; hence, convolution layer is not needed in the skip connection.

The hyperparameter values used in our model can be found in Figure 5.3(b) and (c) (in red annotations), and Table 5.1.

5.6 Experimental Results and Performance Evaluation

We demonstrate performances of our simple CNN and ResNet-50 models in Figure 5.5 using the test set. The figure shows 10×10 heatmaps: blue ones

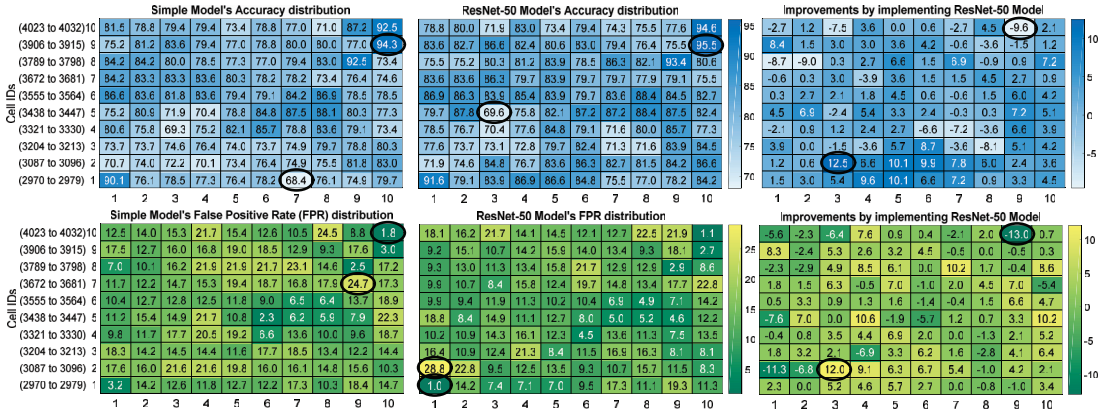

Figure 5.5: Dispersion of accuracy (blue) and false positive rate (FPR) (green) for the simple and ResNet-50 models along with the improvements achieved by implementing the latter model.

representing accuracy distributions and the green ones representing false positive rate (FPR) distributions; with the left, middle and right ones pertaining to the simple model, ResNet-50 model and improvements we achieved by implementing ResNet-50 over simple model, respectively. Each position in a heatmap relates to a corresponding cell of the sub-grid in Figure 5.2(bottom). The best and worst performance values in the left and middle heatmaps are marked in black annotations, while the annotations in right heatmaps represent maximum improvements and degradations.

As we can observe in the figure that the performance results pertaining to different cells vary; this is because fundamentally each cell has it's own unique distribution of user activity values in terms of call incoming, SMS incoming, call outgoing, SMS outgoing, and Internet usage, from which our framework creates grid-images. The model learns different underlying distributions and hence the performance result for each cell is different.

The accuracy of cell 2976 (row 1, column 7)—the worst performing cell—using the simple model is significantly improved from 68.4% to 75.5% by using ResNet-50 model. Cell 3915 (9, 10) yielded maximum accuracy 94.3% using simple CNN, and is slightly further improved to 95.5% using ResNet-50 model. Additionally,

the maximum and minimum FPRs using the simple model are 24.7% and 1.8% for cell 3680 (7, 9) and 4032 (10, 10), respectively; they are further reduced to 17.7% and 1.1%, respectively, when ResNet-50 is utilized. The minimum FPR in ResNet-50's distribution is 1% for cell 2970 (1, 1), a 3× reduction from 3.2% when simple model was utilized.

However, performance also degrades for some cells, as evident in the right-hand heatmaps (indicated with negative values). For example, observe accuracy of cell 3440 (5, 3) that worsened from 71.9% using simple model to 69.6% using ResNet-50 model. Similarly, ResNet-50 model resulted in higher FPR of 28.8% for cell 3087 (2, 1), a significant increase as compared with 17.5% when simple model is used.

Based on the above observations, the individual cell's performance can either be ameliorated or degraded by using ResNet-50 model; however, the overall performance of ResNet-50 model improves as compared with its counterpart, as evident in Table 5.2. Also note the training time for ResNet-50 model is about 7× higher than of the simple model.

To proof scalability of our proposed method, we scaled-up the size of our grid-image from $10 \times 10 \times 5$ to $15 \times 15 \times 5$, to include a total number of 225 grids. For this purpose, we selected cell IDs starting from 5076 to 6728, depicted as inner light-blue square grid in Figure 5.2 (top-right), and kept rest of the parameters of each model same as before. Table 5.3 conveys the overall test performance and training time of both models, and the improvements achieved by leveraging ResNet-50 model over simple CNN model. Figure 5.6 demonstrates performance (accuracy and FPR) distributions of both models for each of the chosen cell IDs in the form of top and middle 15×15 heatmaps. The bottom heatmaps represent the improvements.

Similar to our observations of Figure 5.5, we can also observe in Figure 5.6 that the performance of some cells has improved and for some cells, it has de-

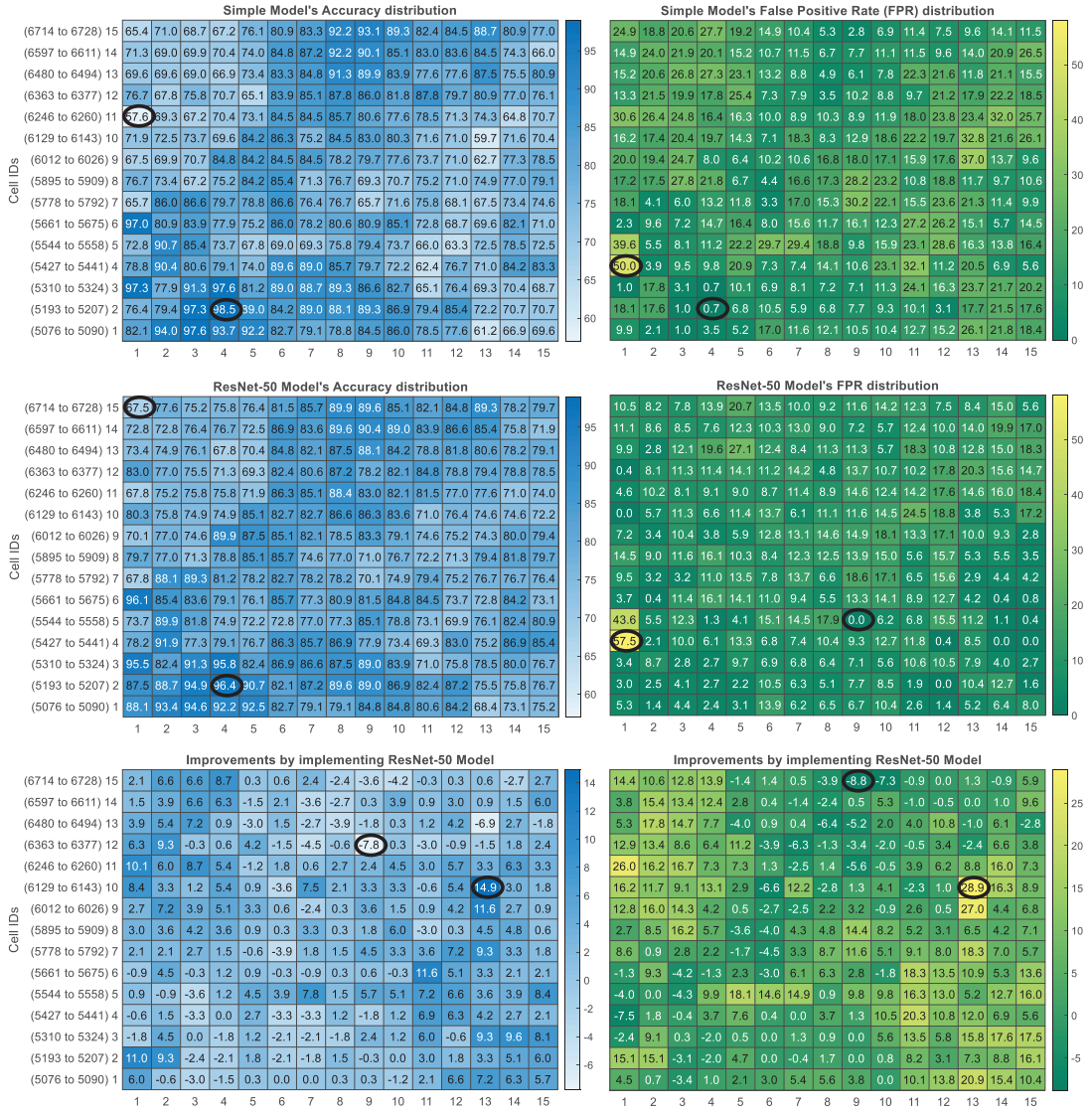

Figure 5.6: Performance dispersions achieved through considering $15 \times 15 \times 5$ input grid-image.

teriorated by applying ResNet-50 model. We can also observe that the overall accuracy and error rate values in Table 5.3 resemble their counterparts in Table 5.2. Additionally, similar to the trend we previously observed in Table 5.2, ResNet-50 model in our current experiments has also achieved better performance results as compared with the simple CNN model except for the recall. Hence, our proposed method is scalable.

If we compare Table 5.3 with Table 5.2, it is interesting to observe that overall

Table 5.2: Performance statistics of Simple CNN and ResNet-50 models.

Metric	**Simple CNN Model**	**ResNet-50 Model**	**Improvement**
Accuracy	78.99%	**81.06%**	2.07%
Error Rate	21%	**18.94%**	2.06%
Precision	69.99%	**73.59%**	3.6%
Recall	64.59%	**67.21%**	2.62%
FPR	13.81%	**12.03%**	1.78%
F1	67.18%	**70.26%**	3.08%
Training Time	**3.52 min**	25.58 min	-

Table 5.3: Performance statistics of Simple CNN and ResNet-50 models (when $15 \times 15 \times 5$ grid-image is used).

Metric	**Simple CNN Model**	**ResNet-50 Model**	**Improvement**
Accuracy	78.21%	**80.4%**	2.19%
Error Rate	21.78%	**19.59%**	2.19%
Precision	64.42%	**72.5%**	8.08%
Recall	**61.9%**	56.34%	-
FPR	14.74%	**9.21%**	5.53%
F1	63.13%	**63.41%**	0.28%
Training Time	**3.88 min**	26.34 min	-

training time do not proportionally increase as we increase the resolution of input image. Hence, the resolution can be enhanced to accommodate anomaly detection for a larger number of cells with the expense of slightly higher computation time. This is because of the two properties of CNN discussed in Section 5.5.2—which enable the number of parameters in a layer of CNN to remain constant even if the input's resolution is varied.

Finally, we compare our model's performance with the performance of feed-forward DNN proposed in Hussain et al. [3]. Hence for comparison, we adopt their feed-forward DNN model with the same hyper-parameter values and implement it on the 100 cells depicted in Figure 5.2 (red grid). Due to the space constraint, we only show the test accuracy distribution in Figure 5.7, which can be compared with our simple CNN model's accuracy distribution in Figure 5.5. In addition, comparison of overall test accuracy and training time of our simple CNN and

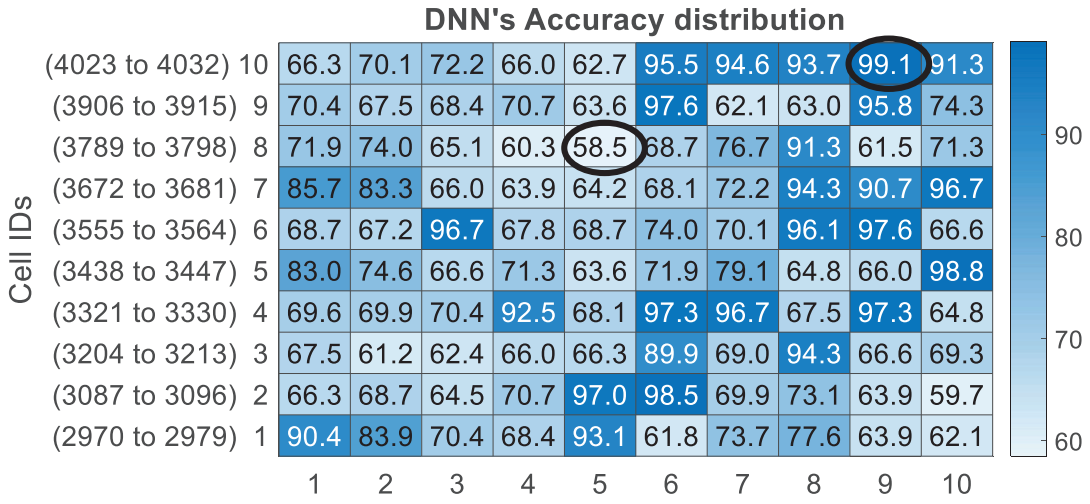

Figure 5.7: Dispersion of the accuracy using Feed-forward DNN model.

ResNet-50 models with feed forward DNN model is shown in Figure 5.8. Although we can find some instances of cells having feed forward DNN outperformed other models in Figure 5.7, but overall the DNN model performed poorly. As evident in Figure 5.8, DNN yielded worst overall test accuracy as well as training time as compared with both of our models.

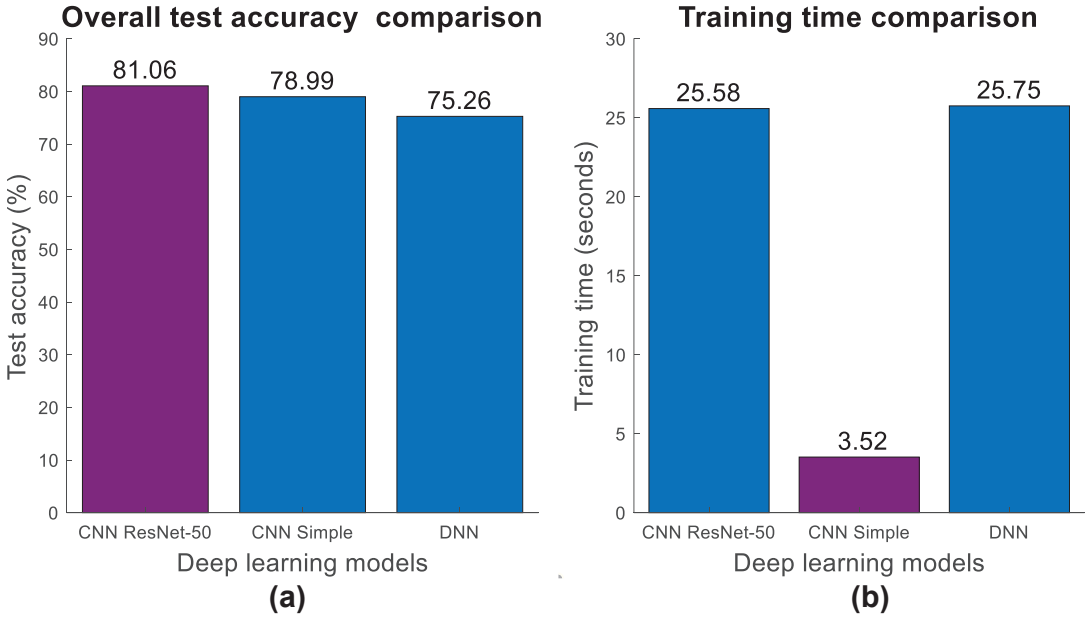

Figure 5.8: Comparison of performance of both models (simple and ResNet-50) by utilizing the DNN proposed in [3]. Best performance is highlighted as purple.

5.7 Conclusion and Insights for Future Work

We discovered that our AI-powered mobile edge computing (MEC)-based anomaly detection system (deployed in an edge server (ES) co-located with a base station) can effectively detect anomalous cell(s) in a 100-cell area with accuracy of $70 - 96\%$, contingent upon the characteristics of each individual cell. Our method is computationally lightweight in contrast to the state-of-the-art solution [99]: it reduces computational burden on the core network (CN) by using MEC approach and convolutional neural network (CNN)—which we analyzed to be more effective in terms of using fewer parameters than feed-forward deep neural network (DNN), as discussed and exhibited in Section 5.5.1. We further examined two CNN models: simple model (conceived from the conventional CNN models) and ResNet-50 model (adopted from a recent paper on residual learning [106]). We discovered that while the latter produced better overall results than the formal, it required considerably more training time, resulting in a trade-off between training time and performance.

Since our scheme is devised to identify abnormalities within minutes—conventional approaches include subscriber complaints and drive tests that take hours and sometimes days to identify the anomaly (cell outage) [35]—this possibly boosts QoS and reduces OPEX as prompt abnormal cell detection equates to a faster issue settlement. The detection of soared traffic movement in a particular area can also serve as an early warning system for possible network congestion. This improves the user experience by preventing user frustration by early detection of such situations. Our framework is robust as a result of inclusion of the Internet activity feature that was unavailable in most of the previous studies [7, 62], as it can identify events such as a sports match with marginally elevated SMS/call behaviors that are regarded to be common but have intensified Internet activity (as social media usage is typically high during such occasions).

In the existing (parametric and hardware) configuration, the simple CNN model appears more suitable for online learning environment as it can capture abnormalities within the arrival of next timestamp (10-min) unless we employ more sophisticated hardware for timely detection using ResNet-50 model. Perhaps, with a more efficient quantum processing hardware [110] in near-future, the emerging and future cellular networks will be able to train even deeper and sophisticated neural network models (ResNet-152 [106], Inception-v4 [107], etc.), quicker and in less time, resulting in improved results. A further impediment to our work's practical applicability is the need for ground-truth labels that can be subdued by generating labels based on the fault data having chronicles of alarms' logs [1]. Selection of optimal values of hyperparameters can also improve performance. Hyperparameter calibration is fundamentally an optimization cycle that reruns the ML model with different hyperparameter compositions in a search space (having ranges for all the hyperparameters) to achieve minimal error. We can do calibration manually which involve domain experts' intervention or implement an automatic but computationally costly method called grid search which involve a discrete hyperparameter search space. Alternatively, random search method [111] can be used which is relatively effective and can be considered for our future research.

For the practical settings, since we can categorize the cellular network with our proposed MEC-based approach as an MEC system with heterogeneous servers, the decision to choose the number of cells monitored by an edge server can depend on multifaceted reasons which mainly concerns resource management [82, Sec. III. C.]. For example, determining whether to offload computation to an ES or if the core network has sufficient computation power at a given instance to perform all the calculations (server selection problem [82, Sec. III. C.], [112]); for the case where the computations are offloaded to an ES, determining how much calculations an ES can handle and then performing pre-processing and

subsequently allocating number of cells accordingly; etc.

We speculate our framework can also conform to the cloud radio access network (C-RAN) architecture, where there are massive number (hundreds or even thousands) of remote radio heads (RRHs) controlled by a centralized, collaborative and cloud-based baseband unit (BBU) pool [113]. In our research context, a BBU pool can act as an ES monitoring user activities pertaining to several RRHs; however, this direction needs further investigation. In industrial Internet of things, our work can also be extended to address anomaly (fault due to device malfunction, connectivity failures, delayed communication, etc.) detection in which a middleware (fog) connected with various entities (actuators, robots, machines, sensors, etc.) monitors their data to report anomalies [114–116]. Fog computing is utilized in the industry for local computing to address delay and security concerns, and a fog node can perform tasks similar to the ones performed by the ES in our research.

In conclusion, this chapter presented a robust, scalable, and novel framework based on MEC, powered by deep CNN (computationally efficient than feed-forward DNN utilized in the latest research) and fueled by real CDR (spatio-temporal) dataset to detect anomalies (pertaining to cell outage and performance degradations, and surged cellular traffic activity leading to a potential congestion) in a 100-cell sub-grid; relieving CN from tremendous computational load of doing data analytics for each cell in the network.

Chapter 6

A Prescriptive Analytics-Based Modular Framework for Proactive Cell Outage and Congestion Detection in Cellular Networks

6.1 Motivation

We have been using CDRs for anomaly detection in the previous works. Although CDRs have advantages over MDT reports, but CDR usage also has a challenging aspect. By the time network collects logs from all the base stations (BSs), processes them to construct CDRs through various logical charging functions [61, Sec. III], and sends them to the relevant computing server (mobile edge computing (MEC) server in case of edge deployment [8]) for anomaly detection, the anomalies and their damage would have already occurred. Therefore, a de-

mand for proactive anomaly detection arises. Moreover, past studies [77, 117], attempting proactive cell outage detection, focus on the whole network rather than forecasting at the base station-level that compromises the model's efficacy in a practical setting.

6.2 Overview and Contributions

Motivated by the above and inspired by a broader idea to enable proactive self-healing in future networks [6, Fig. 7], this is the first study that proposes a prescriptive analytic-based modular framework and investigates the application of traffic forecasting for anomaly detection in cellular networks using deep learning-based techniques. We utilize CDRs as time-series data to perform traffic prediction by deploying a deep convolutional long short-term memory (ConvLSTM) model—adopted because of its promising and high-precision prediction performance in dealing with a similar problem of precipitation nowcasting in the weather forecasting domain [118]. We then feed the output to a second feed-forward deep neural network (ffDNN)-based model for the identification of the anomalies in a BS, as shown in Figure 6.1 and described in Sec. 6.4.2. The framework enables the network to prognosticate and detect anomalies up to 3 hours (10-min resolution) in advance with an average accuracy of over 92% depending on the overall anomalies in the dataset. The prominent contributions of our work are as follows, it:

1. Proposes a framework based on multi-variate multi-step ConvLSTM and feed-forward DNN models to predict cell's traffic 3 hours in advance that enables proactive anomaly detection at a BS.

2. Adopts a novel modular approach that could enable the network to reuse the module outputs for other tasks. For example, the network can switch

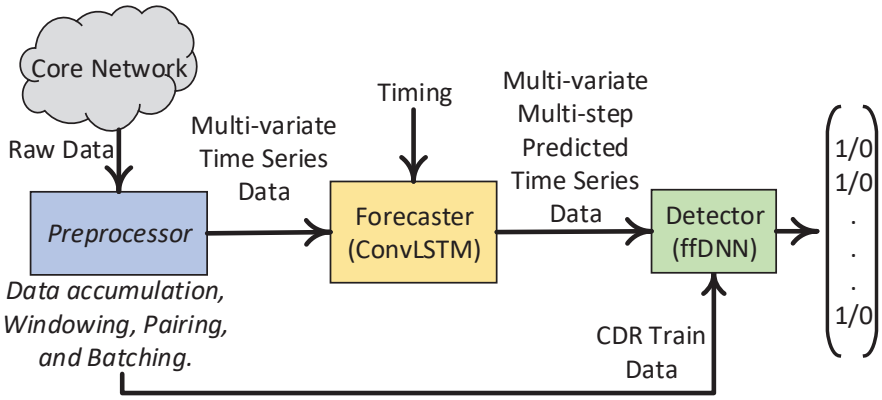

Figure 6.1: System Model.

BSs to sleep mode during low-traffic hours for green communications [119] according to the Forecaster's output.

3. Investigates and illustrates the relationship between forecasting and anomaly detection accuracies, and the factors affecting the latter.

4. Focuses not only on the degenerative performance and outages including the sleeping cell but also circumstances leading towards congestion as anomalies.

6.3 State of the Art

AI technologies have recently gained momentum in achieving higher efficiency as compared with the traditional solutions for anomaly detection; however, they are ignored in 5G standardization and rather anticipated to be adopted in the later phases or in 6G networks, which also infers their potential [20]. In the literature, there are several studies related to either time-series forecasting [118, 120] or anomaly detection [3, 7, 8, 62, 121] using the deep learning techniques in cellular networks. However, only a few studies exist that combine both ideas [77], [117].

Kumar *et al.* [77] utilized one-month time-series data containing fault-related information in a network to predict the next failure's timing. They utilized fault occurrence (index of the fault) and inter-arrival time (time between the adjacent faults, in hours) as variables for the learning models. They considered various machine learning techniques including deep neural networks with autoencoders in their study, which yielded the least error. With a bare 64.29% model accuracy, another major limitation of their work is that the model considered the whole network for the fault prediction as the data is unsegregated for each BS.

Kogeda *et al.* [117] proposed mobile intelligent agents (MIAs) and Bayesian belief network-based fault prediction in wireless networks. MIAs are software programs that can independently roam around in different nodes to monitor them and report if any fault is detected. They set up a miniature network having three wired and wireless devices connected to a router and induced artificial faults like switching off power to a randomly chosen node, etc. Their proposed model registered around 86% prediction accuracy. However, their work's applicability on a large-scale cellular network is questionable since the experiments are done in limited settings. For a BS where hundreds of nodes (users) are present, MIAs could cost the network heftily in terms of communication, storage, and computation resources. They will also affect privacy since they need to be installed on user devices.

Our work addresses the above issues by forecasting the anomalies at BS-level so that the anomalous cell can be pinpointed as CDRs are popular for their accurate user mobility information [61]. Additionally, they are generated at and extracted from the core network, without intervening the user devices to retain privacy.

6.4 Preliminaries

6.4.1 Description of the Dataset

CDRs utilized in this study are based on spatio-temporal data collected from the core network (CN) [61] of Telecom Italia, a network operator in Italy, released to the public as part of the Big Data Challenge 2015 [66]. The data contain user logs for $10,000$ cell IDs of Milan city (illustrated in Figure 6.4, top-left) for the duration of 2 months (62 days) at 10 min granularity. For each slot (10-min duration), there exist multiple records containing values of the following subscriber activities: incoming SMSs (SMS in), outgoing SMSs (SMS out), incoming calls (Call in), outgoing calls (Call out), and Internet.

6.4.2 System Model

The system model mainly consists of the following modules (depicted in Figure 6.1):

1. Preprocessor: it collects raw CDRs via CN and converts them into compatible forms to be accepted by the Forecaster and Detector modules.

2. Forecaster: it accepts multivariate time series data, trains a ConvLSTM [118] model, and outputs selected multivariate multi-step predicted time series data (test dataset for Detector) that represent future activity values in the cell. The selection is based on the "Timing" input.

3. Detector: it accepts CDR dataset from the Preprocessor for training a feed-forward deep neural network (ffDNN) model. It also accepts the test dataset from the Forecaster to test on, and outputs whether the cell at each future timestep will be normal (0) or abnormal (1).

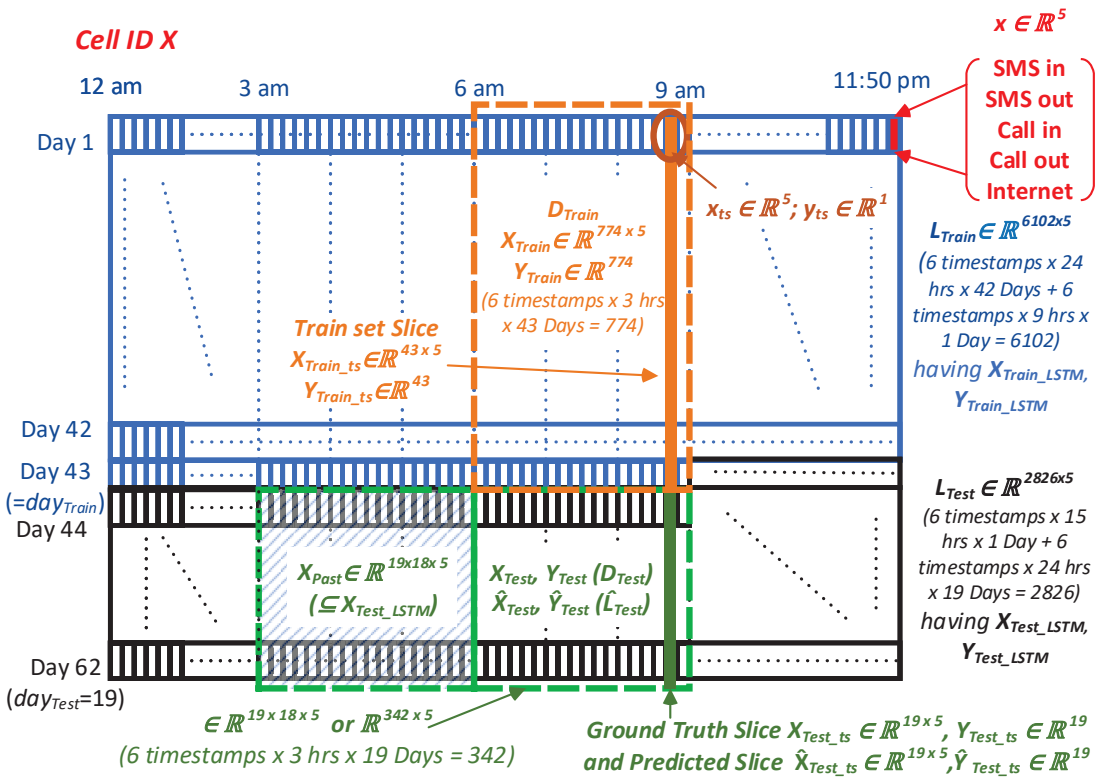

Figure 6.2: Time series dataset $L \in \mathbb{R}^{8,928\times5}$.

6.4.3 Data Preprocessing

We consider three 3-hour duration timings for diversity in our experimentation and to conform our anomaly detection model with the one in [3] for performance comparison in Sec. 6.6; hence preprocessing is executed accordingly. The timings include Morning from 6-9 am, Afternoon from 11 am-2 pm, and Evening from 5-8 pm. For simplicity, we consider Morning timings to describe this section. We preprocess raw CDRs according to the following steps, we:

1. Sum the corresponding activity values, for a given cell ID and a timestamp, from different logs to form a vector $x \in \mathbb{R}^5$ (highlighted in Figure 6.2) representing the total amount of individual activities recorded during the 10-min duration.

2. Concatenate all the activity vectors to form a time series dataset $L \in \mathbb{R}^{8,928\times5}$ (illustrated in the figure) containing a total $8,928 (62 \text{ days} \times 24 \text{ hours/day}$

$\times$ 6 timestamps/hour) 5D vectors.

For the Forecaster, we:

3. Split L into a train set $L_{Train} \in \mathbb{R}^{6,102\times5}$ (blue units in the figure) and a test set $L_{Test} \in \mathbb{R}^{2,826\times5}$ (black units) in approx. 70 : 30 ratio, about 43 (day_{Train}) days data for training and 19 (day_{Test}) days for testing. The splitting is done at the end of 43rd day's Morning timing: timestamp $6,102$ (42 days$\times$24 hours/day $\times$6 timestamps/hour + 1 day $\times$9 hours/day $\times$6 timestamps/hour).

4. Perform normalization on both sets using the mean and standard deviation $\mu, \sigma \in \mathbb{R}^5$ of L_{Train}.

5. Apply a single-step sliding window function with both history (T_x) and future target (T_y) sizes of 18 timesteps (3 hours) to transform the normalized sets into windowed train set having $X_{Train\_LSTM} \in \mathbb{R}^{6084\times T_x\times5}$ and $Y_{Train\_LSTM} \in \mathbb{R}^{6084\times T_y}$, and windowed test set having $X_{Test\_LSTM} \in \mathbb{R}^{2808\times T_x\times5}$ and $Y_{Test\_LSTM} \in \mathbb{R}^{2808\times T_y}$; where 6084 and 2808 are the total number of windows in train and test sets, respectively, and 5 is the number of activities.

6. Convert windowed train and test sets into batches before passing on to the ConvLSTM model. The batch size is set to 256 for training and a unit-sized batch is considered for the testing. Figure 6.3 (bottom) delineates the train batches.

For the Detector, we:

7. Extract a data-block (highlighted in orange and subsequent green dotted portion in Figure 6.2) pertaining to the Morning timings from L. We divide it into a train set D_{Train} (orange) with $X_{Train} \in \mathbb{R}^{774\times5}$ examples and

$Y_{Train} \in \mathbb{R}^{774}$ labels, and test set D_{Test} (green) with $X_{Test} \in \mathbb{R}^{342\times5}$ examples and $Y_{Test} \in \mathbb{R}^{342}$ labels; where 774 and 342 represent the number of activity vectors formed during 43 and 19 days, respectively. The labels are synthetically created according to [3], as they are unavailable in the original dataset.

6.4.4 Performance Metrics

We utilize the following standard metrics to evaluate Forecaster's performance: mean absolute error (MAE), mean squared error (MSE), root MSE (RMSE), and R-squared (R^2) [120].

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{6.1}$$

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{6.2}$$

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{6.3}$$

$$\text{R}^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y}_i)^2} \tag{6.4}$$

where, y_i is the ground truth, $\hat{y}_i$ is the prediction, and $\bar{y}_i$ is the mean of y_i.

For the Detector, we utilize accuracy as a metric, elaborated in Section 2.5.

6.5 Implementation

6.5.1 Forecaster

As depicted in Figure 6.3, we deploy a multi-variate multi-step ConvLSTM model [118] by passing the train batches that essentially have $[x^{<1>}, ..., x^{<T_x>}]$ and $[y^{<1>}, ..., y^{<T_y>}]$ as inputs for the training. Here, we consider $y \in \mathbb{R}^1$ as the ground

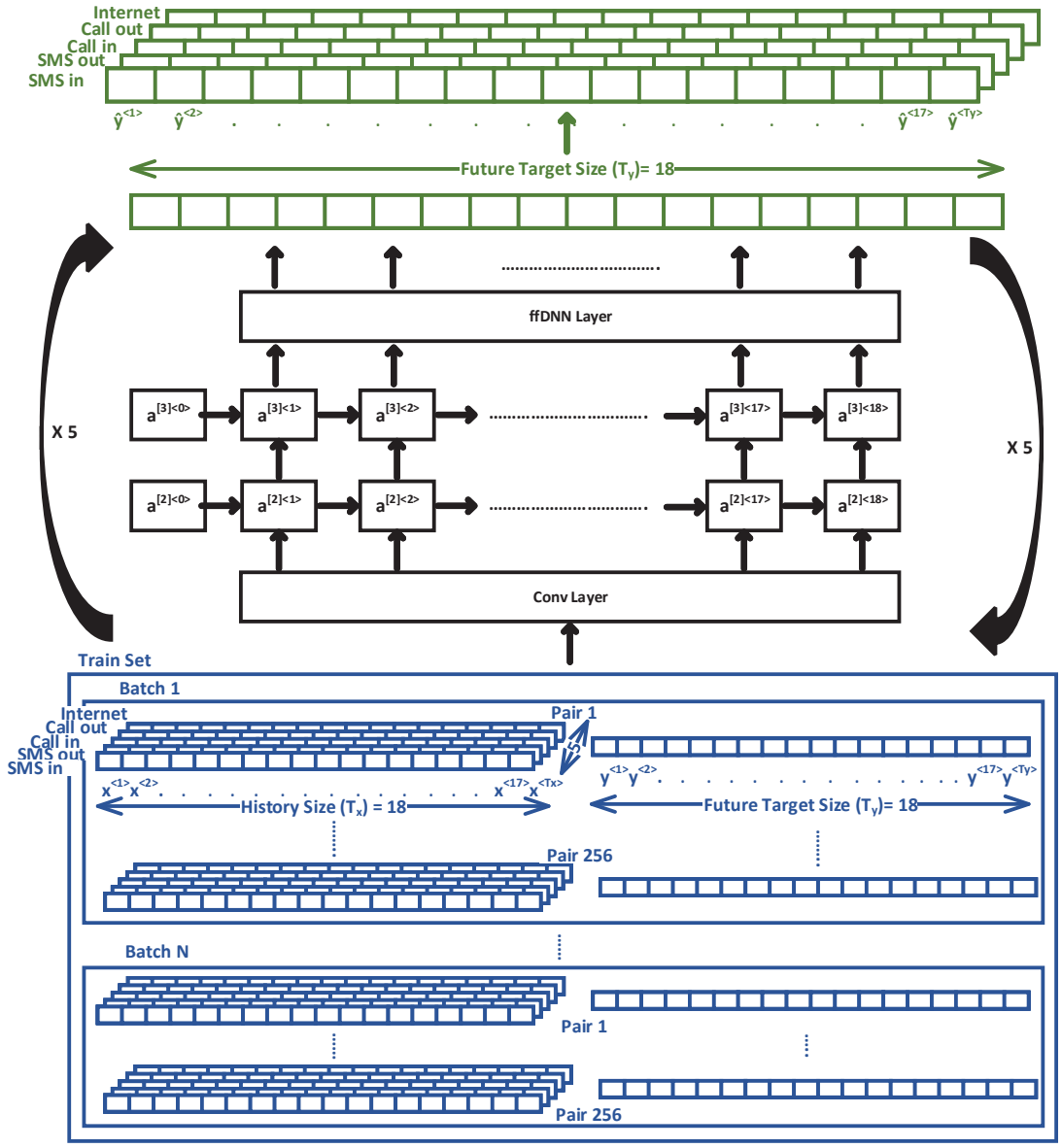

Figure 6.3: Operation of a ConvLSTM model.

truth (future) activity. We then select windows according to the (Morning, Afternoon, or Evening) timings from the windowed test set $\{X_{Test\_LSTM}, Y_{Test\_LSTM}\}$ for the testing. For example, sub-dataset $X_{Past} \in \mathbb{R}^{day_{Test} \times T_x \times 5} (\subseteq X_{Test\_LSTM})$, as shown in Figure 6.2, is chosen for the Morning timings containing day_{Test} 5D (multivariate) windows, each associated with a single day and of size T_x.

Although the model accepts multi-variate input, it generates a univariate output of size T_y. Hence, it is executed five times to generate five univariate

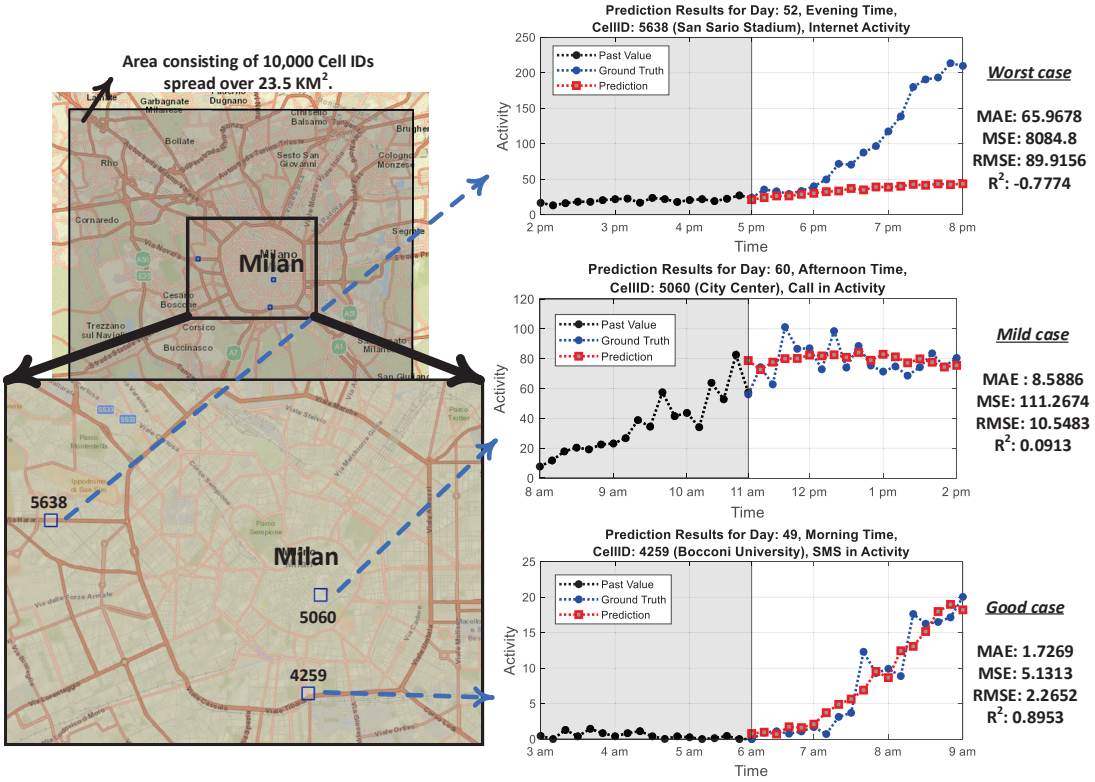

Figure 6.4: (Left) Milan's map overlayed with GPS coordinates of the total area associated with the 10,000 cell IDs (outer square black box) with a zone zoomed in the bottom highlighting three cells (blue boxes). (Right) Prediction results of the selected cell IDs alongside the performance metrics.

predicted windows each pertaining to a different activity during testing. These windows are then concatenated to form a 5D (multi-variate) predicted window $[\hat{y}^{<1>}, ..., \hat{y}^{<T_y>}]$, illustrated in Figure 6.3 (Top). Here, $\hat{y} \in \mathbb{R}^5$ is the predicted 5D activity vector. Due to the page limitation, the remaining details of the ConvLSTM model can be found in [118]. Finally, all the generated windows for day_{Test} days are merged to form $\hat{X}_{Test} \in \mathbb{R}^{day_{Test} \times T_y \times 5}$, as shown in Figure 6.2, containing the predicted traffic activities.

6.5.2 Detector

We implement an ffDNN based on [3] that trains on D_{Train}. We utilize the Forecaster's result $\hat{X}_{Test}$ with labels $\hat{Y}_{Test}$, synthetically generated according to [3],

as the test set $\hat{L}_{Test}$ to perform anomaly detection while D_{Test} act as the ground truth data.

6.6 Experimental Results and Performance Analysis

6.6.1 Forecaster's Preliminary Results

We select three cell sites for our preliminary experiments: San Sario stadium (cell ID 5638), City center (cell ID 5060), and Bocconi university (cell ID 4259). We utilize their GPS coordinates to overlay them as blue boxes with Milan's map in Figure 6.4 (left). For diversity, we select three different timings (evening, afternoon, and morning), activities (Internet, call in, and SMS in), and days to demonstrate the prediction results in the figure (right). We also utilize Eq. 6.1 - 6.4 with the ground truth (X_{Test}) and predicted ($\hat{X}_{Test}$) values to calculate the model's performance metrics displayed alongside the graphs in the figure.

Performance-wise, the plot for cell ID 5638 on the 52^{nd} day (22nd Dec. 2013) represents a worst-case where the gap between predicted and ground-truth values drastically increases from 6:20 pm (8th future timestep) onwards due to an ongoing soccer match at the coverage area [3]. The ground truth values (blue) indicate anomalies pertaining to the surge in Internet traffic demand that may need urgent additional resources otherwise this may cause congestion. Since predicted data ($\hat{L}_{Test}$) from our forecasting model are later utilized by our anomaly detection model, the anomaly detector might mis-classify them as normal instances resulting in false negatives. In contrast, the plot for cell ID 4259 represents a good case where the gap at every time step remains small. Hence, the anomaly detector will correctly classify the instances as true positives or true negatives, accordingly. Finally, cell ID 5060 represents a mild case where we have a small

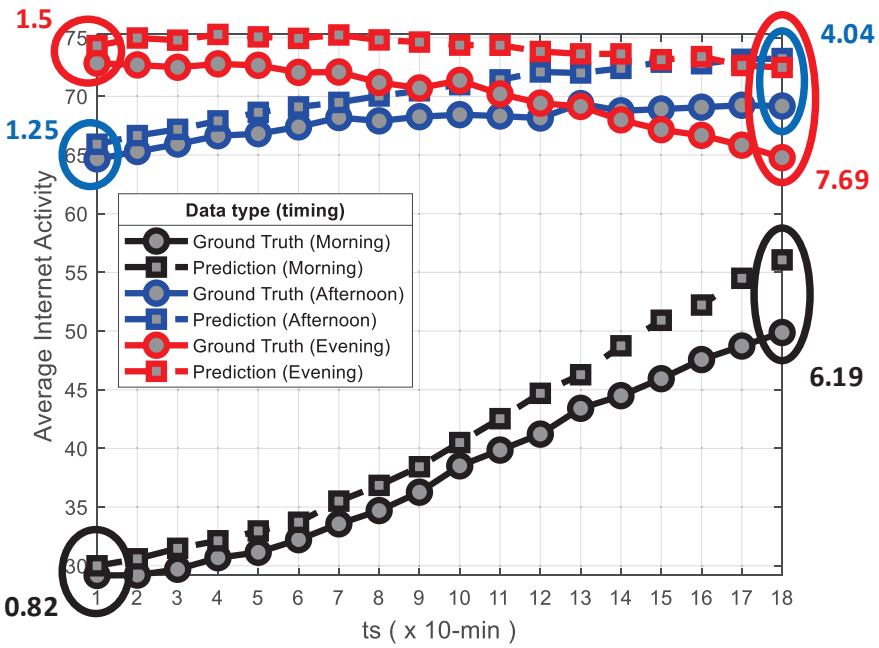

Figure 6.5: Average ground truth and predicted Internet activity values over a 3-hr duration consisting of 18 10-min timesteps (ts). The annotated values show the difference between the two values.

gap at almost every time step. For each case, the displayed performance metrics reflect the above-mentioned descriptions.

6.6.2 Forecaster's Bird's-eye Results

We scale up our experiments to observe Forecaster's overall test performance by randomly selecting 100 cell IDs out of the total 10,000. For each chosen cell ID, timing, and timestep (ts), we have co-located test set slices (as indicated in Figure 6.2): Ground truth slice $X_{Test\_ts} \in \mathbb{R}^{day_{Test} \times 5} (\subseteq X_{Test})$ and predicted slice $\hat{X}_{Test\_ts} \in \mathbb{R}^{day_{Test} \times 5} (\subseteq \hat{X}_{Test})$.

Due to the page-limitation and since Internet activities are dominant as compared with other activities in the original dataset, we choose this activity for the depiction of the overall performance in Figure 6.5 and 6.6. For this purpose, we average (a total of $day_{Test} = 19$) values in each slice to have a corresponding pair of ground truth and predicted Internet values. Since we have 18 such slices (for

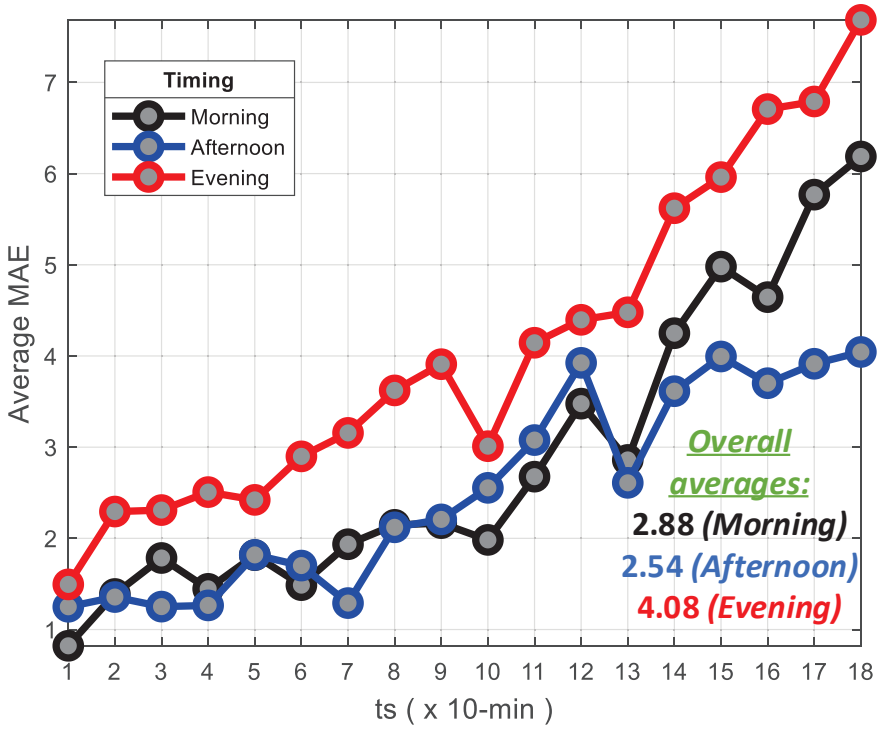

Figure 6.6: Average mean absolute errors (MAEs) over a 3-hr duration consisting of 18 10-min timesteps (ts).

the 3-hr duration), we have an equal number of such pairs. We finally take a grand average involving all the cell IDs; constituting the three pairs at each ts in Figure 6.5, each pair corresponding to a separate timing.

As observed in the figure, the gap between ground truth and predicted values from the first ts to the last jumps about 7.5, 3, and 5 folds for the morning, afternoon, and evening hours, respectively. The further our model forecasts the higher the gap is, which infers the prediction accuracy of the Forecaster is inversely proportional to the number of ts. Alternatively, this is also reflected in Figure 6.6 where the MAEs (equivalent to the respective gaps in Figure 6.5) mostly increase with the ts.

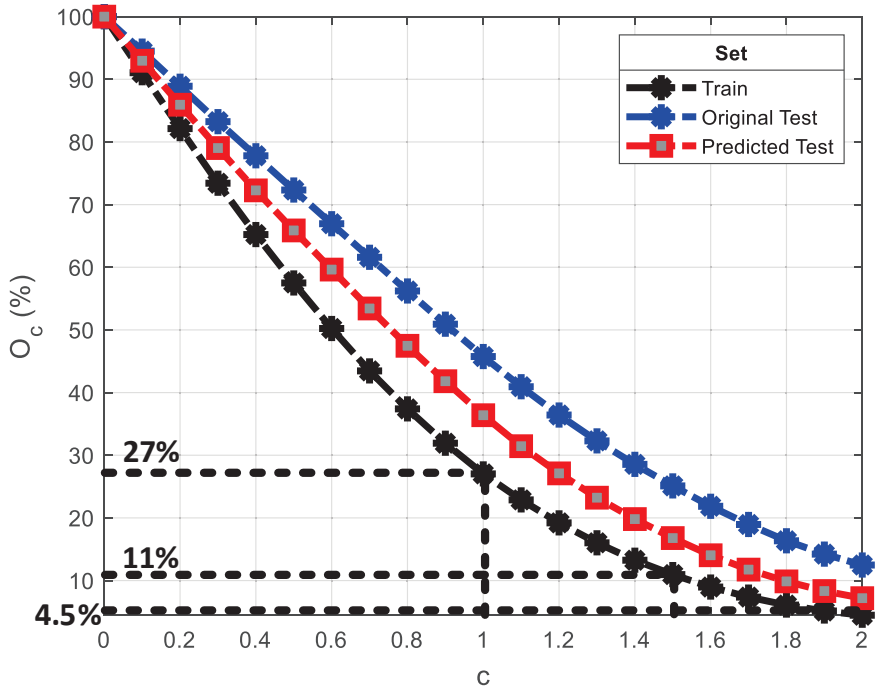

Figure 6.7: Effect of varying coefficient (c) in Eq. 6.5 on the average percentage of anomalies (O_c) in different sets.

6.6.3 Rule-Based Detector's Results and the Factors Affecting its Accuracy

The increasing gap in Figure 6.5 leading towards the debased prediction accuracy also translates into the tendency of an anomaly detection accuracy to degrade over the ts because of the potentially high classification error, as demonstrated in Figure 6.9 and explained later in this subsection. Another factor contributing towards the falling detection accuracy is the coefficient (c) in the following equation (adopted from [3] and utilized for artificially labeling the examples), which can influence the percentage of anomalies (ones) or the accepted range for normal instances (zeros) in Y_{Train}:

$$y_{ts} = \begin{cases} 0, & \text{if } \|\mu_{ts} - c \times \sigma_{ts}\|_2 \leq \|x_{ts}\|_2 \leq \|\mu_{ts} + c \times \sigma_{ts}\|_2 \\ 1, & \text{if otherwise} \end{cases} \tag{6.5}$$

where $x_{ts} \in \mathbb{R}^5$ and $y_{ts} \in \mathbb{R}^1$ (indicated in Figure 6.2) are an example and its label in the train (or test) set, respectively, pertaining to ts; and μ_{ts} and $\sigma_{ts} \in \mathbb{R}^5$ are mean and standard deviation, respectively, of the sub-train set $X_{Train\_ts} \in \mathbb{R}^{day_{Train} \times 5} (\subseteq X_{Train})$ which is a slice containing ($day_{Train} =$) 43 training set examples associated with ts. c controls the number of instances to be labeled as normal (or abnormal): with an increasing c, the anomalies in each set decreases because the normal range widens. Figure 6.7 demonstrates the relationship between c varying from 0 to 2 and the average percentage of ones (anomalies) O_c at each value of c in the train, original test, and predicted test sets. The figure is generated by utilizing the following general equation which computes O_c by averaging the count of ones o_{ts_c} in all the slice label vectors (each pertaining to ts and composed from the components build using Eq. 6.5); considering slices from all timings (t), cell IDs ($cell$), and ts:

$$O_c = \sum_{t=1}^{T} \left(\sum_{cell=1}^{CELLS} \left(\sum_{ts=1}^{TS} \left(\frac{o_{ts_c}}{DAY} \times 100 \right) \frac{1}{TS} \right) \frac{1}{CELLS} \right) \frac{1}{T} \tag{6.6}$$

where, $T(= 3)$ refers to the total number of timings; $CELLS(= 100)$ is the total number of cells; $TS(= 18)$ is the total number of timesteps; and DAY is the total number of days in the set (day_{Train} for train set or day_{Test} for test set) or the length of the slice.

Moreover, Figure 6.8 depicts the effect of varying c from 0 to 2 on the average detection test accuracy $A_{c,t} \in \mathbb{R}^1$ given c and t. The plot is generated by utilizing the following equation which calculates the accuracy by averaging the sub-test sets'/slices' accuracies $a_{ts_{c,t}} \in \mathbb{R}^1$ from all cell IDs and ts:

$$A_{c,t} = \sum_{cell=1}^{CELLS} \left(\sum_{ts=1}^{TS} \left(a_{ts_{c,t}} \right) \frac{1}{TS} \right) \frac{1}{CELLS} \tag{6.7}$$

where, $a_{ts_{c,t}}$ is computed using Eq. 2.4 by employing original Y_{Test_ts} and predicted

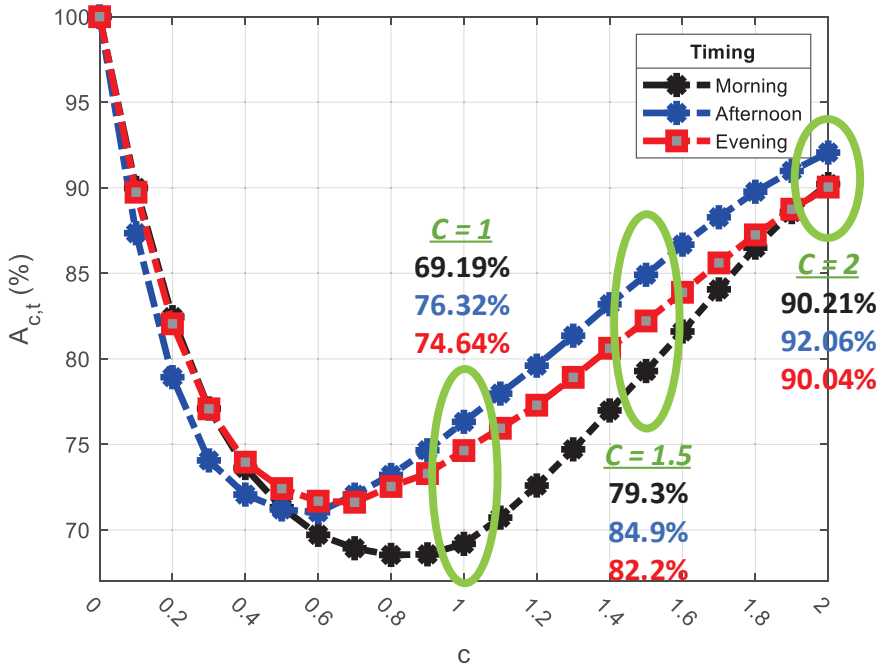

Figure 6.8: Effect of varying coefficient (c) in Eq. 6.5 on the average detection test accuracy ($A_{c,t}$) at different timings. Overall accuracies for each timing at $c = 1, 1.5$, and 2 are highlighted separately.

$\hat{Y}_{Test\_ts} \in \mathbb{R}^{day_{Test}}$ sub-test set (slice) labels (calculated using Eq. 6.5) pertaining to ts and for a particular c and t. $A_{c,t}$ has a bell-shaped curve and increases from the lowest point if c increases. Note, if $c = 0$ then only instances equal to μ_{ts} are marked normal (Eq. 6.5) as (almost) all the instances lie outside the single-point normal region and are marked anomalies. Although this condition is yielding 100% accuracy as the anomaly detection model is simply classifying every instance as an anomaly, however, it is impractical.

We consider three values on the x-axis as $c = 1, 1.5,$ and 2; and show their accuracy distributions over ts in Figure 6.9. As can be observed, increasing c positively impacts both overall (Figure 6.8) and individual (Figure 6.9) accuracies and the trend for the latter also leans flat. For a visual demonstration, we plot straight (green) lines in Figure 6.9 that best fit afternoon timing accuracies at different c using the least-squares regression method and calculate their slopes

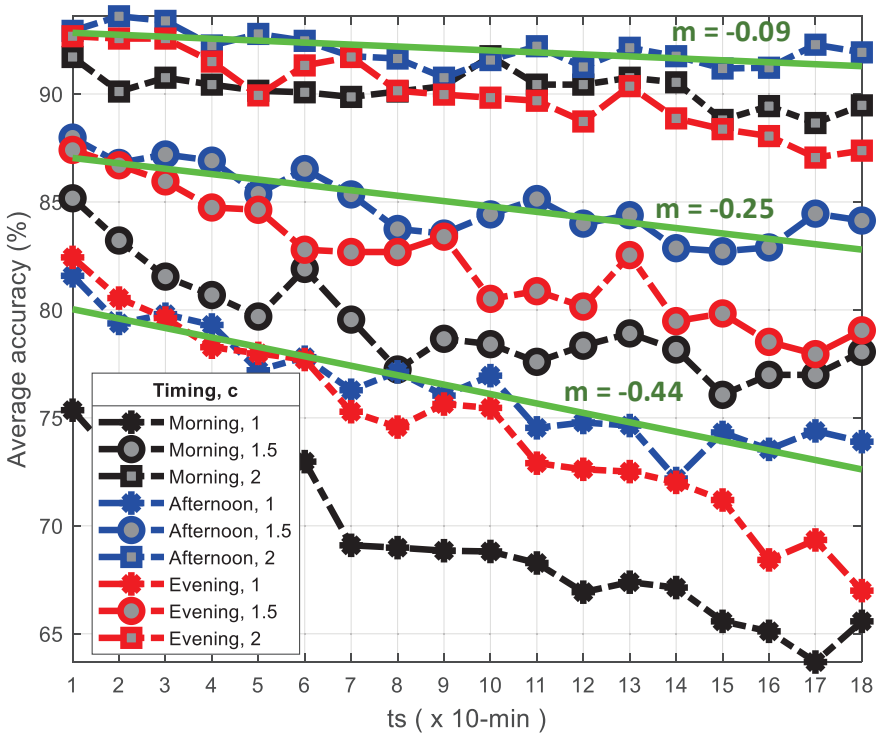

Figure 6.9: Average accuracy distributions over 3-hr duration consisting of 18 10-min timesteps (ts) for the rule-based anomaly detector at different coefficients (c) and timings.

(m) using the following standard equation:

$$m = \frac{TS \times \sum_{ts=1}^{TS}(ts \times a_{ts}) - \sum_{ts=1}^{TS} ts \times \sum_{ts=1}^{TS} a_{ts}}{TS \times \sum_{ts=1}^{TS}(ts^2) - \left(\sum_{ts=1}^{TS} ts\right)^2} \tag{6.8}$$

where, a_{ts} is the accuracy at ts. As we increase c, m moves closer to 0 inferring we can have high classification accuracy even for the farther ts if c is large. For example, when $c = 2$ (around 5% anomalies), we can achieve over 90% average accuracy for each timing (Figure 6.8). In practical settings, c is usually high because the anomaly is by nature rare. This is also evident in Figure 6.6 where the overall averages (mean of 18 MAE values) are lesser than the MAE value of the mild case discussed in Figure 6.4, which indicates the worst case is a seldom event.

Table 6.1: Comparison of Test Accuracies of Different Detectors

c	**Rule-Based Detector**	**ffDNN-Based Detector**		
	Predicted Data	**Original Data**	**Predicted Data**	**Difference**
1	73.38%	85.51%	71.38%	14.14%
1.5	82.13 %	92.31%	85%	7.31%
2	**90.77**	**94.4%**	**92.72%**	**1.68%**

6.6.4 ffDNN-based Detector for Comparative Analysis and Improvements

We show the test accuracies (averaged over all timings and chosen cell IDs) for both models (rule-based and ffDNN-based) in Table 6.1. Note, labels ($\hat{Y}_{Test}$) are generated according to the values of c mentioned in the table. As can be observed, the difference between the accuracies calculated by utilizing original and predicted data through the ffDNN-based model keeps on minimizing with the increase in c.

Moreover, if we compare Figure 6.10, where we show the accuracies (averaged over all cell IDs) of the ffDNN-based model at each timestep, with Figure 6.9; the ffDNN-based model yielded a slightly improved test accuracy as compared with the rule-based model with the predicted test set. This is because of the superiority of deep learning-based models over traditional models as also concluded in [3, 8].

6.7 Discussion and Conclusion

We discovered that for higher coefficients (c)—equating to around 5% or fewer anomalies in train set—our framework achieved over 92% average anomaly detection test accuracy (Table 6.1) by utilizing the forecasted traffic data with feed-forward deep neural network (ffDNN)-based detector; having only a difference of

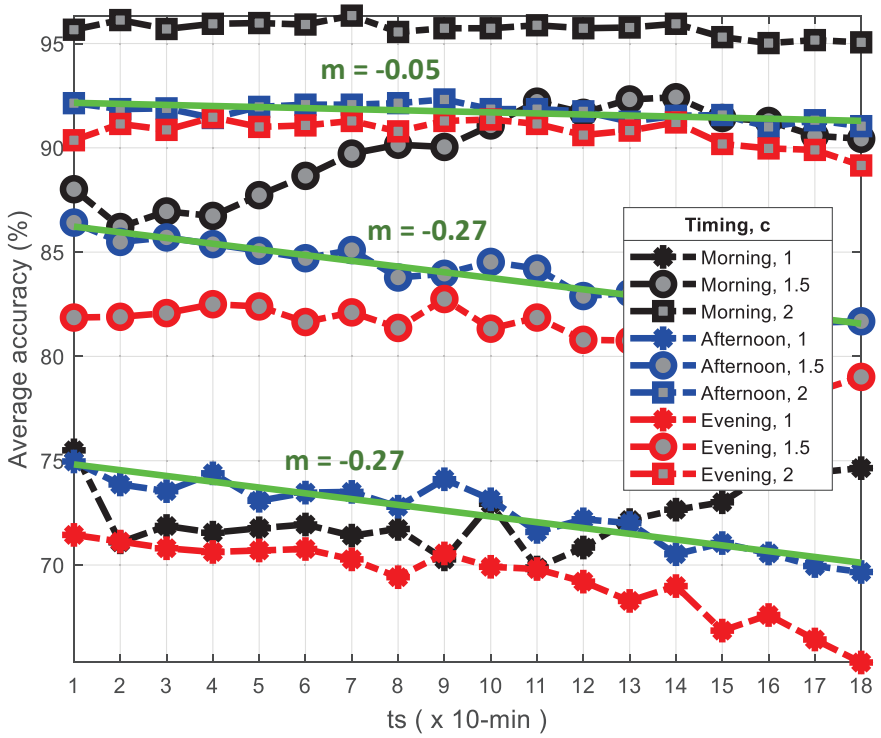

Figure 6.10: Average accuracy distributions over 3-hr duration consisting of 18 10-min timesteps (ts) for the ffDNN-based anomaly detector at different coefficients (c) and timings.

1.68% lesser than the case where the original data is utilized. This highlights the efficacy of our proposed framework and the potential of effectively enabling the cellular network to rely on the forecasted traffic to perform proactive anomaly detection for up to 3 hours in advance. We also observed that the further we forecast traffic, the higher the gap between the ground truth and predicted values (Figure 6.5) or MAEs (Figure 6.6); which in turn translates into lower anomaly detection accuracies (Figure 6.9 and Figure 6.10) due to the larger misclassification error. However, this deterrent is diluted with higher values of c suggesting we can further stretch the 3-hour forecasting window—perhaps up to the 6-hour design limit of the ConvLSTM model [118].

Our work concentrated on anomalies pertaining to the individual base station (BS) rather than the whole network, overcoming the principal limitation in [77]; while also preserving user privacy since the processing occurs at the core network (CN) instead of involving the user devices, which was one of the constraints

in [117]. Our work can be further extended (because of the modular approach) to add extra features: 1) integrating the diagnosis and compensation modules to support the proactive self-healing paradigm [6, Fig. 7]; 2) adding multi-source data from social media to enable self-awareness; 3) updating the existing modules with other advanced ones for higher performance, such as replacing ffDNN in the Detector with CNNs [8] to efficiently introduce edge computing for relieving CN from heavy computation; and 4) recycling the outputs of any module for an additional problem-solving task in the cellular network, for instance, utilizing Forecaster's output to switch BSs to sleep mode during low-traffic hours for green communications [119]. Moreover, congestion detection is another strong characteristic of our work that was also missing in the earlier studies. This will empower the network to prepare for a situation in advance, e.g. if congestion is expected after 30 minutes, perhaps due to an ongoing soccer match in the vicinity, the network can pre-allocate the resources, such as deploying unmanned aerial vehicles [122] to support additional users in the ROI.

On the other hand, the lack of fault-related information in CDRs limits our work as we synthetically labeled the anomalies based on Eq. 6.5. This can be compensated by bonding the fault data with the timestamps in the CDRs. Moreover, the deep ConvLSTM model is computationally expensive which may be difficult for the CN to execute for all the BSs. This limitation can be overcome by promoting MEC-based infrastructure [8].

Given the financial incentive for the operators to have efficient solutions for anomaly detection, our proactive approach can be expanded to various networks and infrastructures, and can significantly aid in reducing their OPEX. For example, our framework can play a vital role in smart-city IoT [121] and industrial IoT infrastructure [26] where an exorbitant number of devices communicate with each other and with the cellular network without human intervention making it more difficult to identify the faults because of the lack of complaints about

the degraded services from the customers. For cyber–physical systems (CPSs) connected via cellular networks, our framework can proactively identify anomalous events such as distributed denial of service attacks to avoid (both, physical and monetary) damages [63]. However, the conformity of our model to the industrial and smart-city IoT environments and CPS infrastructure needs further investigation.

This work demonstrated prescriptive analytics in which the proposed modular framework forecasted user activities 3 hours beforehand and employed them to identify whether a base station will undergo an outage or congestion. Inspired from the precipitation nowcasting problem in the weather forecasting domain, it adopted the ConvLSTM model for the Forecaster module while for the Detector it adopted an ffDNN-based model from the literature. Our framework manifested the efficaciousness to achieve over 92% detection accuracy which could be even further improved in the future. The proposed framework has the potential to be integrated into later versions of 5G or future 6G networks.

Chapter 7

Deep Convolutional Neural Network-Based Distributed Denial of Service-Attack Identification for Cyber-Physical Systems over 5G Networks

7.1 Motivation

AI/DL algorithms are gaining attention among the researchers from industry/nation state [123] and academia [124] alike to implement them for the cybersecurity of CPSs utilized in critical infrastructures like financial networks, smart grids, etc. Besides DL technology's burgeoning success in image recognition domain [81], it is gaining popularity across a wider domain of applications which is also apparent from the fact that the granted AI patent applications have recently surged worldwide [125, Fig. 1].

This motivated the research carried out in this chapter to apply powerful DL-based image recognition algorithms for the security of cellular networks in the context of CPSs.

7.2 Overview and Contributions

This chapter extends the Chapter 5's knowledge and applies DL (specifically, convolutional neural networks (CNNs)) for detecting distributed denial-of-service (DDoS) attack (involving SMS flooding, silent call, signaling, and their composite attacks described in Section 2.2) in cellular networks under the backdrop of cyber-physical system (CPS) security. The void in existing literature that this chapter attempts to fill is that most of the existing works implement heavy-computational solutions that analyze the actual contents of the user activities to execute attacks detection; which also compromises user privacy [64, 126]. The proposed method is lightweight and preserves privacy because: 1) actual contents of individual activities i.e. SMS, call, or Internet, are not involved; and 2) it employs call detail record (CDR) data which is already present in the system, rather than relying on data demanding extra resources (communication, observation time, computation, etc.) for their collection. Since record of users and network interactions are contained in CDRs inferring behavior of a normal base station, they are a great source to detect behavior of an under-attack base station [64, 65].

This chapter makes the below-mentioned salient additions to the existing literature:

1. Offers an original and joint framework to detect SMS spamming, signaling, silent call, and their composite attacks that trigger DDoS attack in the network infrastructure.

2. Proposes an expandable and scalable resolution to the availability-attack iden-

tification by employing CNNs. The input image resolution can be stretched so that it includes more cells without any model modification.

3. Employs latest models which are very deep i.e. residual network having 50 layers (subsequently referred as ResNet-50). Additionally, this chapter deploys a comparatively simpler model named as deep rudimentary convolutional neural network (DRC) model with six layers. It results in improved identification accuracy for a majority of the attacks.

7.3 Relevant Work

A unified framework that offers detection of signaling, SMS flooding, and silent call attacks' detection is missing in the literature as most of the works focus on the individual detection of these availability attacks—which is the reason of studying them separately in this section. Past several works have employed content-based methods for the detection having analyzed the original contents of the subscriber activities (SMS messages, IP packets, etc.) [64, 126]. But, such approaches have excessive computational burden and may be infeasible in practice.

Tu *et al.* in [2] and in their elongated paper [47] thoroughly discussed the severity of the 4G network's **silent call attack**. In [126], Ruan *et al.* monitored crests and difference in the traffic data volume by employing game theory for the attack's detection. As compared to the past works which offered content-based and computationally-rigorous solutions, they asserted their solution to be lightweight.

Regarding the **signaling attacks**, authors in [127] presented a hidden semi-Markov-based detection design which utilize the bearer wakeup packet formation rate in wireless sensor and actuator network (WSAN). The problem with the design is its craving for the training instances composed from the past network data which might require up to several days of observations to obtain. A criteria

was set in [48] in which bearer requests/user/minute were utilized to identify the attack i.e. if number of the requests are beyond a threshold, then the attack is declared. But, the study failed to highlight a firm procedure to ascertain the threshold which will heavily influence the performance of the detector. A method using support vector machine (SVM) algorithm by analyzing a group of IP packets of a user device was proposed by Gupta *et al.* [44] to detect the attacks. But, the presented approach is computationally-rigorous because of using the actual contents of the devices.

Authors in [50] presented an exhaustive discussion on **SMS flooding / spamming attack** and Papadopoulos *et al.* [64, 65] studied the attack's detection. Using a simulation to create SMS flooding and signaling attack scenarios, the authors in [64] utilized the synthetically-generated CDRs (containing reflections of the attacks on user activities) for the attacks' detection. By utilizing graphs, a descriptor is proposed for the detection of abnormal cellular devices within hourly data belonging to a cell. By utilizing clustering techniques the authors in their subsequent article [65] clustered malicious users causing SMS flooding attack into groups according to their distinctive traffic behaviors. In a similar way and under the context of machine-to-machine (M2M) communications, Murynets *et al.* [45] proposed a clustering approach using graphs and SMS activities (via CDRs) to discover DDoS attack triggered by SMS flooding attacks.

This chapter, in contrast to the above works, provides: (1) a lighter alternative to the content-based methods by utilizing CDRs to identify the attacks; (2) a unified framework by detecting all three attacks (SMS flooding, signaling, and silent call); (3) a faster solution that detects attacks within 10 minutes; and (4) a broad-scale identification mechanism as it can detect numerous cells concurrently as it utilizes a deep CNN. Our study detects lasting attacks rather than the instantaneous ones as it integrates historical user activities related to a cell into the learning mechanism; since real CDRs contain the temporal features. To

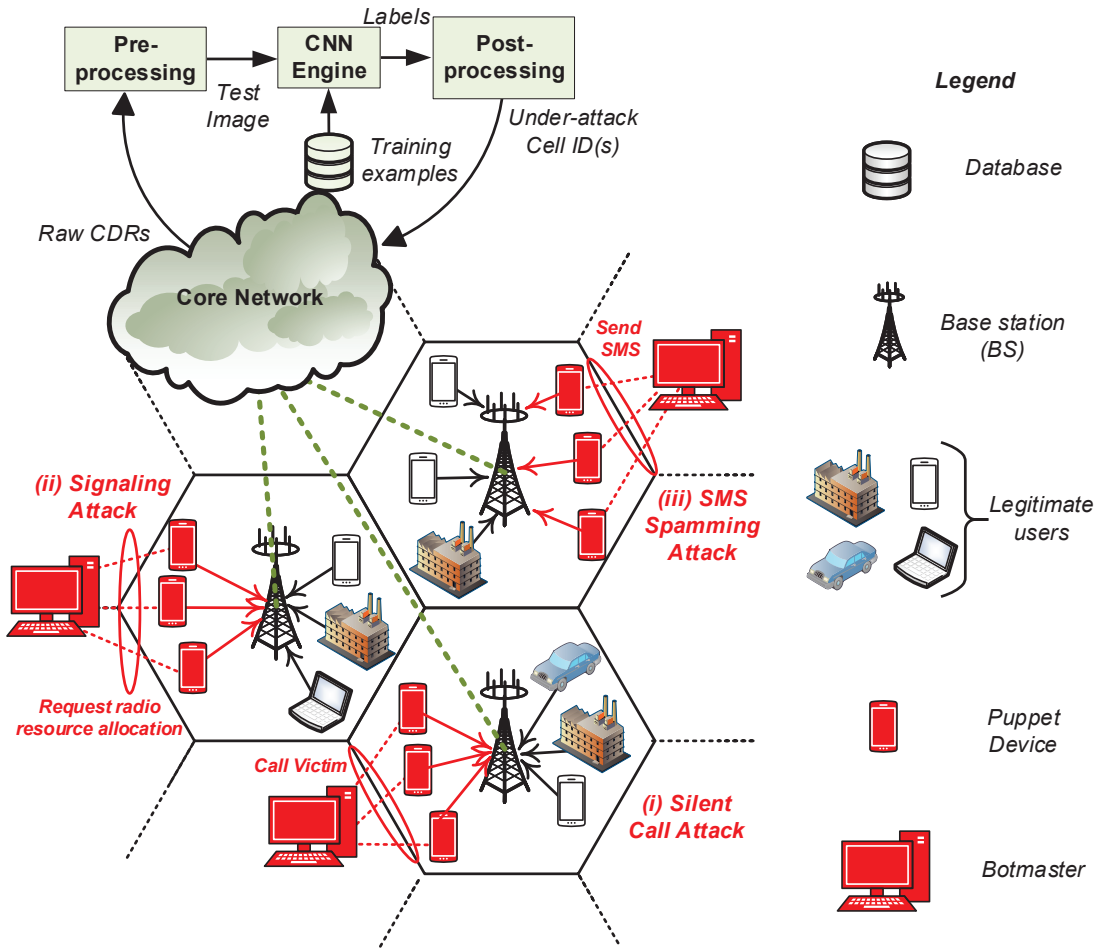

Figure 7.1: System model for the proposed deep CNN-based DDoS Attack Detector.

elaborate this further, because of an instant hype in traffic activity which can be caused due to several reasons (such as, when a base station is covering an airport with frequent newly connected devices and executing traffic activities in bursts), the detector could mistakenly identify such situations as an attack if previous traffic trend is not taken into account.

7.4 Emulating Each Attack's Effect

In order to preserve privacy, Telecom Italia has concealed the information about the subscribers, including the number of user devices in each cell; hence,

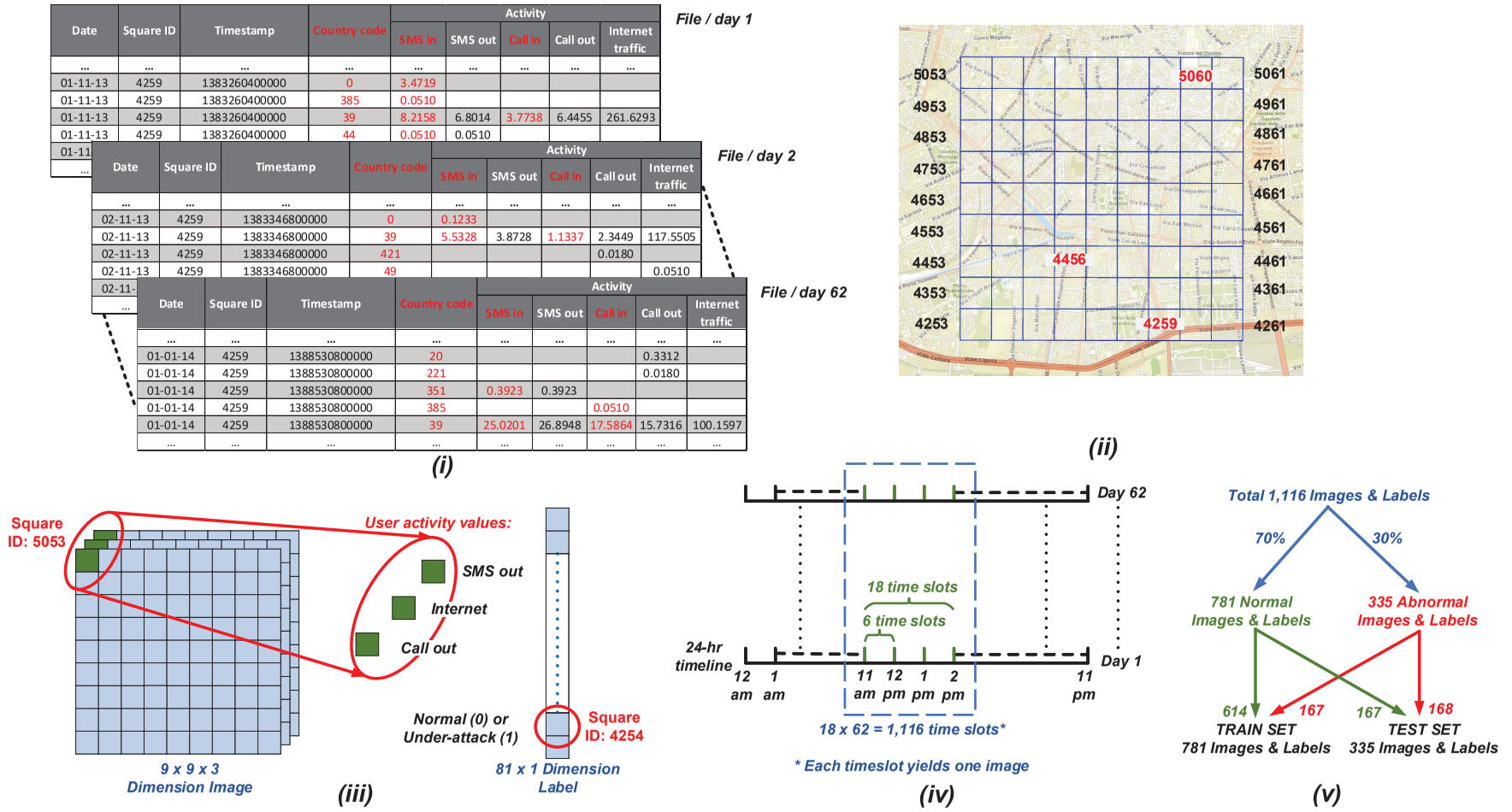

Figure 7.2: Preludes: **(i)** CDR Samples from Milan dataset. The red features are ignored because of their irrelevance to this research. **(ii)** Selected 9×9 sub-grid. **(iii)** Formation of the input image based on the activity values from 81 cells. Each pixel value of the image corresponds to the three user activity values of the corresponding cell ID. **(iv)** Depiction of how data is aggregated in terms of time slots to generate $1,116$ images. **(v)** Settings to generate training and testing sets.

we approximate the number in an indirect way: 0.235 kilometer square (sub-grid's area [66]) $\times$ 7,157 residents per kilometer square (population density of Milan city in the year 2014 when the dataset was accumulated [128]) $\times$ 34% (Telecom Italia's market share [66]) $\times$ 1.509 (cellular subscribers per capita of Italy in 2014 [129]) $\approx$ 863 cellular handsets/devices per sub-grip/cell of Milano grid. According to [46], 6% botnet-controlled devices i.e. around 52 devices in our case are adequate to initiate a 4G network DDoS attack. In the subsequent section, we compute the amount of user activity values relevant to each attack that we have to modify in order to emulate the corresponding attack's effect.

7.4.1 Silent Call Attack

761.5 minutes per 30 days is the duration that a usual phone subscriber talks [46], which makes 0.302 minutes per 10 minute period (we assume most calls execute in 14 hours a day i.e. 8am - 10pm). 1 : 33 is the proportion of concurring active with average users in a usual base station [46], leading towards $\approx$ 26 active users in the cell. Under normal circumstances, there are $10/0.302 \times 26 \approx 861$ CDRs or outgoing calls generated per 10 minute period. If 26 puppet devices are assumed to be executing the attack, only 26 CDR logs will be rendered by them while the legitimate users will be refused the services. As a result, this makes around $861/26 \approx 33.1$ times reduced user activity/CDRs records greatened in the base station.

7.4.2 Signaling Attack

A maximum of 8 devoted bearers can be initiated by a single malicious device. For each bearer, three activation and three deactivation messages (signaling messages) are created within two minutes [48]. This aggregates into each device generating a total of 240 messages during 10 minutes. As the CDR pertaining

to the Internet activity gets recorded every instance when a device initiates or discontinues an Internet session [66], we can assume that every single message out of the total 240 yields a CDR. Therefore, $12,480$ CDRs will be generated by the 52 botnet-controlled devices. A plot representing number of user devices versus HTTP requests in [130, Fig. 1(left)] can be analyzed to calculate approx. number of CDRs produced by a usual (normally functioning) device. From the linear trend in the plot, we can process the middle point as 10^5 (requests) $/10^3$ (devices) = 100 requests/device over 7 days $\approx$ 0.3404 CDRs per device in 10 minutes (considering 14 hours per day, as mentioned previously, and also each HTTP request can generate two CDRs, each upon device connection and disconnection). Hence, a total of $12,756$ CDRs ($12,480$ CDRs by 52 malicious devices + 276 CDRs by remaining 811 normal devices) will be generated under the attack scenario; while only 294 CDRs (0.3404×863 users) will be generated in a cell under normal condition. Overall, about $12,756/294 \approx 43.3\times$ more Internet CDRs/user activity will be logged.

7.4.3 SMS Spamming Attack

At most 30 SMSs in half an hour are allowed in an android-based phone to be sent; however, *HackFacebook* app grants over 1002 SMS [50]. Let's say a typical user sends 10 SMS/day; a total of 103 SMS/10-min (assuming 14-hours a day, as mentioned previously) can be send by all devices under normal conditions and a total of $17,465$ SMS/10-min can be send under the attack scenario ($17,368$ SMS by 52 malicious devices + 97 SMS by the remaining 811 devices). In total, about $17,465/103 \approx 169.5\times$ more SMS CDRs or user activity will be registered.

7.4.4 Blended Attack

We consider each compromised device carries out all the three attacks in this intense case. As a result, we adjust the user activity values accordingly.

7.5 Preliminaries

7.5.1 System Model, Description of the Dataset, and Data Preprocessing

We show the system model in Figure 7.1. Each cell is assumed to have legitimate and also illegitimate (botmaster-controlled) devices (such as IoT devices, mobile phones, devices serving a CPS, etc.)—the exact number of devices is determined in Section 7.4. The jeopardized devices trigger a mutual DDoS attack, robbing genuine devices from using the resources, by separately executing the signaling, silent call, or SMS flooding attack.

We use Milan dataset (elaborated in Section 2.4.3) and choose a 9×9 sub-grid composed of 81 cells, illustrated in Figure 7.2(ii), for our experimentations, since CNNs fundamentally accept data having a grid-like topology [84, Ch. 9]. We have chosen a smaller size of the sub-grid to smoothly illustrate and discuss the results in later sections; however, a larger sub-grid size can also be chosen. The highlighted cell IDs 5060, 4456, and 4259 in the figure are covering a few renowned places in Milan: city center, nightlife places, and Bocconi university, respectively. We can observe their phone usage plots illustrating the behavioral trends in [66, Fig. 7].

Raw CDRs linked to the chosen 81 cell IDs are pre-processed by the framework for each timestamp so that a $9 \times 9 \times 3$ dimension image (shown in Figure 7.2(iii)) can be created. We denote the image as $i^{(j)} \in \mathbb{R}^{n_H^{[0]} \times n_W^{[0]} \times n_C^{[0]}}$ in which j represents the index, and $n_H^{[0]}$, $n_W^{[0]}$, and $n_C^{[0]}$ are the height, width, and depth/channels of

the image, respectively. The height and width denotes the selected cell IDs while the channels contain the activity values. The framework then sends the image to a CNN-based engine, illustrated in Figure 7.1. The engine contains a database of past images (training examples) associated with all the 10-min slots in a 24-hour duration. Given a test example i.e. current image associated with a certain timeslot, the CNN model is trained using all the previous images belonging to the exact timeslot. It then identifies under-attacked and normal cell ID(s) in the test example. Lastly, the information is then transferred to the CN for further actions.

7.5.2 Data Synthesis and Splitting

Typical DL models are data-hungry requiring thousands of training examples; however, our dataset contain just 62 images per 10-min timeslot (each image associated with a single day). Hence, as done in Chapter 4, we consider all the images formed during a 3-hour period as linked to a single timeslot and we consider the morning hours ranging from 11 am - 2 pm. This data augmentation yields a total of $1,116$ images (62 days $\times$ 3 hours $\times$ 6 images per hour), with activity data from 81 cells in each. These images reflect a normal demeanor and therefore, we label each cell as 0 in the corresponding labeled output $o^{(j)} \in \mathbb{R}^{81\times 1}$ (illustrated in Figure 7.2(iii)) of the image.

For an image exhibiting behavior influenced by an attack, we would ideally attack an operational 4G network and notice the changes in the recorded CDRs—potentially resulting in, an economically and legally unfeasible, network breakdown. We hence reserve a set of randomly chosen 335 images (30% of the total) and for each attack scenario, we utilize the set to modify the relevant user activity (call out, Internet or SMS out) values according to Section 7.4 to mimic the effect of the attack (silent call, signaling or SMS spamming) on CDRs. This step is inspired from [64] in which the authors utilize simulation software

to mimic the effect of different attacks and generate a synthetic CDR dataset for their experiments. Practical networks deal with normal scenarios more often as compared with the abnormal ones (anomalies), this is reflected in our model as the normal instances are chosen to be in larger quantity (70%) than the abnormal ones (30%). For the purpose of modification, we randomly choose about 50% cell IDs in each image and also change their labels to 1 (under-attack).

As shown in Figure 7.2(v), our train set contains 781 images (70% of the total) $I_{train} \in \mathbb{R}^{781 \times n_H^{[0]} \times n_W^{[0]} \times n_C^{[0]}}$, and their corresponding labels $O_{train} \in \mathbb{R}^{781 \times 81}$, and the test set contains the remaining ones: $I_{test} \in \mathbb{R}^{335 \times n_H^{[0]} \times n_W^{[0]} \times n_C^{[0]}}$ and $O_{test} \in \mathbb{R}^{335 \times 81}$. Out of the 781 labeled images in train set, 614 are normal and the remaining 167 are the modified images. Similarly, out of the 335 labeled images in the test set, 167 are normal and the remaining 168 are the modified images. Note, for each attack scenario, we have a separate train and test sets because of the modifications discussed previously.

7.5.3 Performance Metrics and Software Utilized

We utilize the following common metrics (described in Section 2.5) for the performance evaluation: accuracy, error rate, precision, recall, false positive rate (FPR), and F_1. We use MATLAB and Keras (Python's DL library) for the preprocessing, GPS mapping, and building the CNN models. We perform experimentation using a commercial PC (i7-7700T CPU, Windows 10 64-bit operating system, and 16GB RAM) with an in-built GPU (NVIDIA GeForce 930MX).

7.6 Realization of CNN Models

Although the methodology utilized in this chapter resembles with the one in previous chapter; however, there are a few modifications. Hence, for the sake of completeness, we describe the methodology in this section.

7.6.1 Generic Architecture

CNN has the following three fundamental building blocks:

Convolution layer

processes images or previous layer's activations $A^{[l-1]} \in \mathbb{R}^{m \times n_H^{[l-1]} \times n_W^{[l-1]} \times n_C^{[l-1]}}$, having m as the total number of images in the (train or test) dataset and l as an index of the present layer; and kernels $K^{[l]} \in \mathbb{R}^{k^{[l]} \times k^{[l]} \times n_C^{[l-1]} \times n_C^{[l]}}$, having $k^{[l]} \times k^{[l]} \times n_C^{[l-1]}$ as the single kernel's dimension with $k^{[l]}$ as the kernel size, and $n_C^{[l]}$ as the total number of kernels. To demonstrate a convolution layer's functionality, we focus on an example highlighted in the red box of Figure 7.5. Here, the index of present (output) layer l is 2, while the previous (input) layer's index will be $l - 1 = 1$. We also consider a single image as an example; hence, $m = 1$. The input activations will then be represented as $A^{[1]} \in \mathbb{R}^{1 \times n_H^{[1]} \times n_W^{[1]} \times n_C^{[1]}}$, having $n_H^{[1]} = n_W^{[1]} = 13$ and $n_C^{[1]} = 3$. The dimension of input activations is $13 \times 13 \times 3$, as shown in the figure. Additionally, the kernels are represented as $K^{[2]} \in \mathbb{R}^{k^{[2]} \times k^{[2]} \times n_C^{[1]} \times n_C^{[2]}}$, having a kernel size $k^{[2]} = 2$ and total number of kernels $n_C^{[2]} = 8$. A single kernel's dimension is $2 \times 2 \times 3$.

The convolution layer applies convolution operation between the input activations and each kernel separately, as shown in the figure. A general convolution operation between input and a single kernel is demonstrated in [84, Fig. 9.1]. The output from each operation is then added with bias (a real number) and a non-linear activation function called Swish is also utilized.

Swish is a gated version of sigmoid function which has some desirable properties that even the widely-used and most successful activation function like rectified linear unit (ReLU) lacks: non-monotonicity and smoothness [90]. The inventors of Swish function claim that it yields matching or outperforming results as com-

pared with ReLU for deeper neural networks. Mathematically, it is defined as:

$$g(z) = z \times \sigma(z) \tag{7.1}$$

where, $\sigma(z) = (1 + e^{-z})^{-1}$ is the sigmoid function.

Finally, the layer piles up each result on top of one another to create an output $A^{[l]} \in \mathbb{R}^{m \times n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}}$ which is represented as $A^{[2]} \in \mathbb{R}^{1 \times n_H^{[2]} \times n_W^{[2]} \times n_C^{[2]}}$. The height $n_H^{[2]}$ or width $n_W^{[2]}$ are computed by using:

$$n_{H/W}^{[l]} = \lfloor \frac{n_{H/W}^{[l-1]} + 2p^{[l]} - k^{[l]}}{s^{[l]}} + 1 \rfloor \tag{7.2}$$

having, $p^{[l]}$ as number of zero-padding (a technique used to insert zeros around the input image's edge to prevent shrinking of output dimension during the convolution operation [84, Sec. 9.5]) and $s^{[l]}$ as stride (distance between consecutive application of kernel on the input). For this example, the zero-padding is already previously performed (see Figure 7.5 (bottom)), hence $p^{[2]} = 0$ and $s^{[2]}$ is given as 1. By utilizing Eq. 7.2, we can calculate $n_H^{[2]} = n_W^{[2]} = 12$. Hence, the output's dimension will be $12 \times 12 \times 8$, which can also be observed in the figure.

In addition, batch normalization (BN) [108] technique is utilized to boost training speed and make the model robust. It is applied between convolution operation and the activation function.

Pooling layer

utilizes a max or avg function to pool maximum or average numbers, respectively, from groups of its input (and from each channel, independently) depending on the kernel size k, to generate the output volume. This reduces requirement for storing parameters and improves model's computational efficiency [84, Sec. 9.3]. If the input has $n_H \times n_W \times n_C$ dimension, the output's dimension can be derived

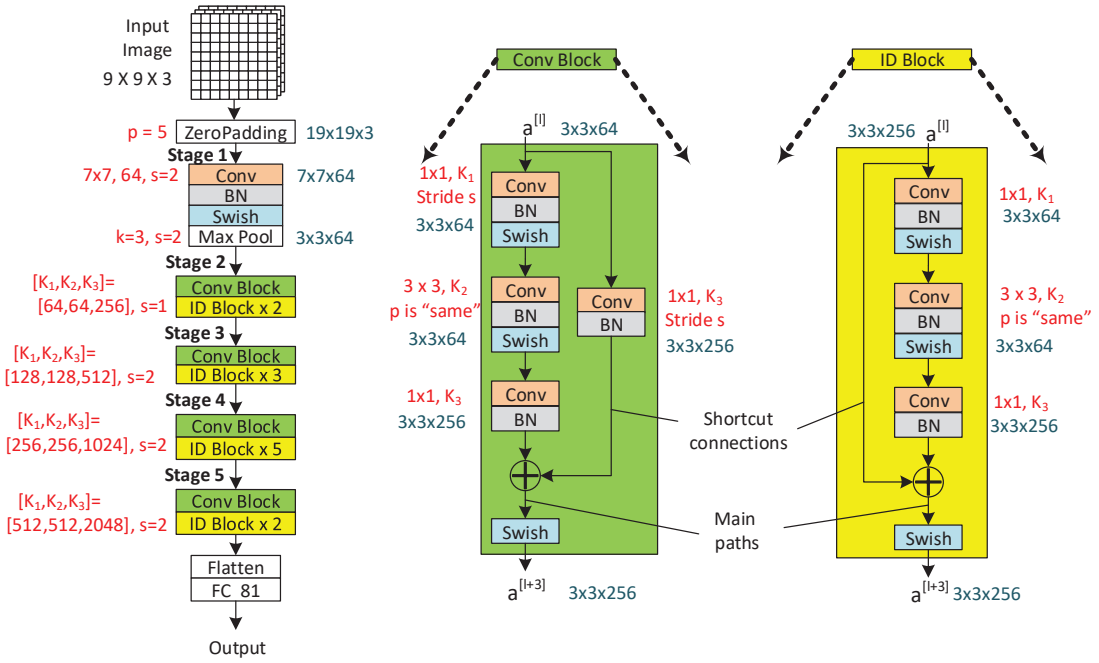

Figure 7.3: Architecture of residual network with 50 layers (ResNet-50). Red notations indicate the hyperparameters values utilized in this work and the blue ones show the layers' output dimensions (pertaining to stage 2).

using Eq. 7.2 with $p = 0$: $\lfloor \frac{n_H - k}{s} + 1 \rfloor \times \lfloor \frac{n_W - k}{s} + 1 \rfloor \times n_C$.

Fully connected layer

has the same purpose as of a feed-forward neural network's hidden layer [99], having each neuron connected with all other previous layer's neurons.

7.6.2 Residual Network Model

The fundamental building blocks can be utilized in multiple settings (with different number of layers and the way they are linked together) to create various CNN models like residual network comprising 50 layers (ResNet-50) [106], illustrated in Figure 7.3. It is one of the most advanced CNN models that we utilize in this study. Residual networks are effective in dealing with the problems encountered by a typical (very) deep neural network—gradient exploding or vanishing [84] and degradation [106] problems—by adopting residual learning in

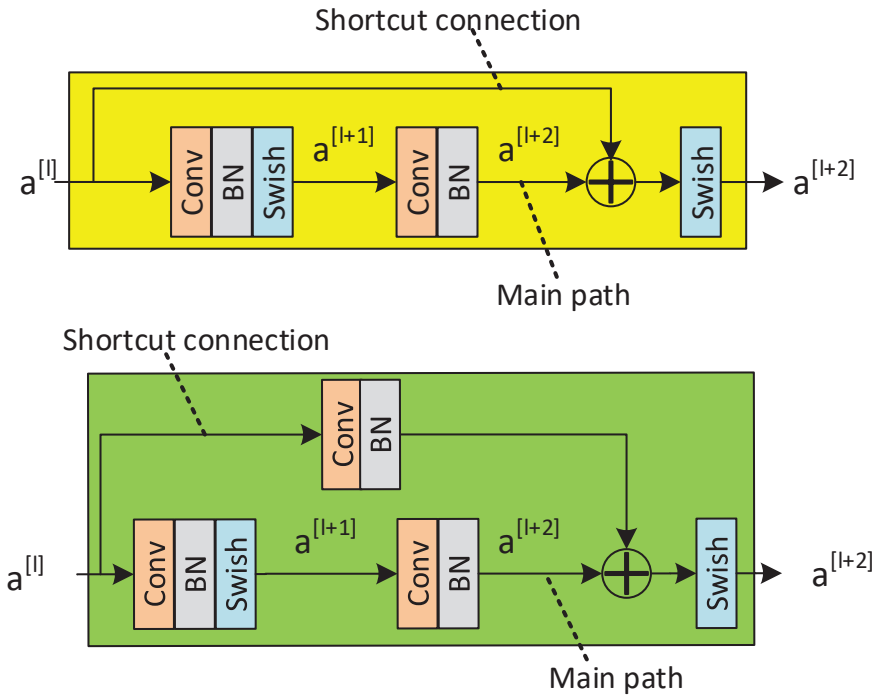

Figure 7.4: Structure of (Top) Identity (ID) and (Bottom) Convolutional (Conv) residual blocks.

which residual blocks are extensively used.

We first elaborate functioning of a residual block using Fig. 7.4 (top). In the figure, the information flows from input $a^{[l]}$ to the output activation $a^{[l+2]}$ via two different paths. In the downward path, known as main path, there are two parts. The information first goes through the initial part having three modules consisting of a convolution layer, batch normalization, and a non-linear activation function, respectively; governed by the following standard equations:

$$z^{[l+1]} = W^{[l+1]}a^{[l]} + b^{[l+1]} \tag{7.3}$$

$$a^{[l+1]} = g(z^{[l+1]}) \tag{7.4}$$

where, $W^{[l+1]}$ is the weight matrix, $b^{[l]}$ is the bias vector, $g(.)$ is the non-linear activation function, $a^{[l]}$ is the input, and $a^{[l+1]}$ is the output of the first part. The batch normalization module is added to accelerate the training.

Similarly, the modules in the second part are governed by the following equations (ignoring the other path and an addition operation):

$$z^{[l+2]} = W^{[l+2]}a^{[l+1]} + b^{[l+2]} \tag{7.5}$$

$$a^{[l+2]} = g(z^{[l+2]}) \tag{7.6}$$

In residual networks, $a^{[l]}$ is fast-forwarded to a deeper hidden layer in the neural network where it is added with the output of that layer before applying a non-linear activation function. This is known as a short-cut connection, as shown in the figure. Hence, Eq. 7.6 will be modified as follows:

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]}) \tag{7.7}$$

The addition of $a^{[l]}$ makes it a residual block. Here, we are assuming that the dimensions of both, input $a^{[l]}$ and $z^{[l+2]}$ (and therefore output $a^{[l+2]}$) are same in order to perform the addition. This kind of residual block is known as identity (ID) block. If the dimensions of input ($a^{[l]}$) and output activations ($a^{[l+2]}$) do not match then a convolution layer in the shortcut connection is inserted to resize the input $a^{[l]}$ to a different dimension, so that the dimensions match up in the final addition. This type of residual block is known as Convolutional (Conv) block, as shown in the Fig. 7.4 (bottom). Note, we utilize residual blocks that skips 3 hidden layers in our paper instead of skipping 2 hidden layers as delineated in the figure.

In ResNet-50 model, residual blocks are piled up on top of one another (see *Stage* $2-5$ in Figure 7.3 (left)) to grant activations of one layer to skip some layers and be directly fed to the deeper layers. During back-propagation, shortcut connections also allow a gradient to be directly back-propagated to the previous layers. As can be seen in the figure that the input image having dimension $9\times9\times3$

is zero-padded with padding $p = 5$ to have an output volume with dimension $19 \times 19 \times 3$ (Eq. 7.2 can be utilized in calculating the output dimension of various layers). The resultant volume is then passed to *Stage* 1 having a convolution layer with kernel size $k = 7$, total number of kernels $n_C = 64$, and stride $s = 2$; that converts the dimension to $7 \times 7 \times 64$. Finally, pooling layer (Max Pool) having $k = 3$ and $s = 2$ yields the output volume with dimension $3 \times 3 \times 64$.

For *Stage* 2, middle part of Fig. 7.3 having Conv block will have an input dimension of $3 \times 3 \times 64$ from the previous layer. The main path contains 3 parts. The first part has convolution layer having $k = 1$, $n_C = K_1 = 64$, and $s = 1$. It outputs volume with same dimensions as of the input's. The convolution layer in the second part also results output with same dimension as of the input's, because it is utilizing "same" convolution (in which padding is set so that the output's dimension remains same as of the input's). The third part having a convolution layer with $k = 1$, $n_C = K_3 = 256$, and $s = 1$ will transform the input's dimension from $3 \times 3 \times 64$ to $3 \times 3 \times 256$. Finally, convolution layer in the shortcut connection, that has input volume of dimension $3 \times 3 \times 64$, scales up the input's dimension to $3 \times 3 \times 256$ by utilizing the following parameter values $k = 1$, $n_C = K_3 = 256$, and $s = 1$. The outputs from both convolution layers (one in the shortcut connection and the other in third part of the main path) can be added as they are now compatible: have same dimensions.

The ID blocks of *Stage* 2 have similar function as of the above-mentioned Conv block, with the exception of the shortcut connection's design that does not have any layer in it. This is because the input of the ID blocks has same dimension as of the output of convolution layer in it's third part: $3 \times 3 \times 256$; hence, convolution layer is not needed in the shortcut connection.

The rest of the stages (*Stage* $3 - 5$) follow a similar pattern as above and ultimately yield a resultant volume of dimension $1 \times 1 \times 32$. It is then flattened in the form of an array and passed on to a final fully-connected layer (50^{th} layer)

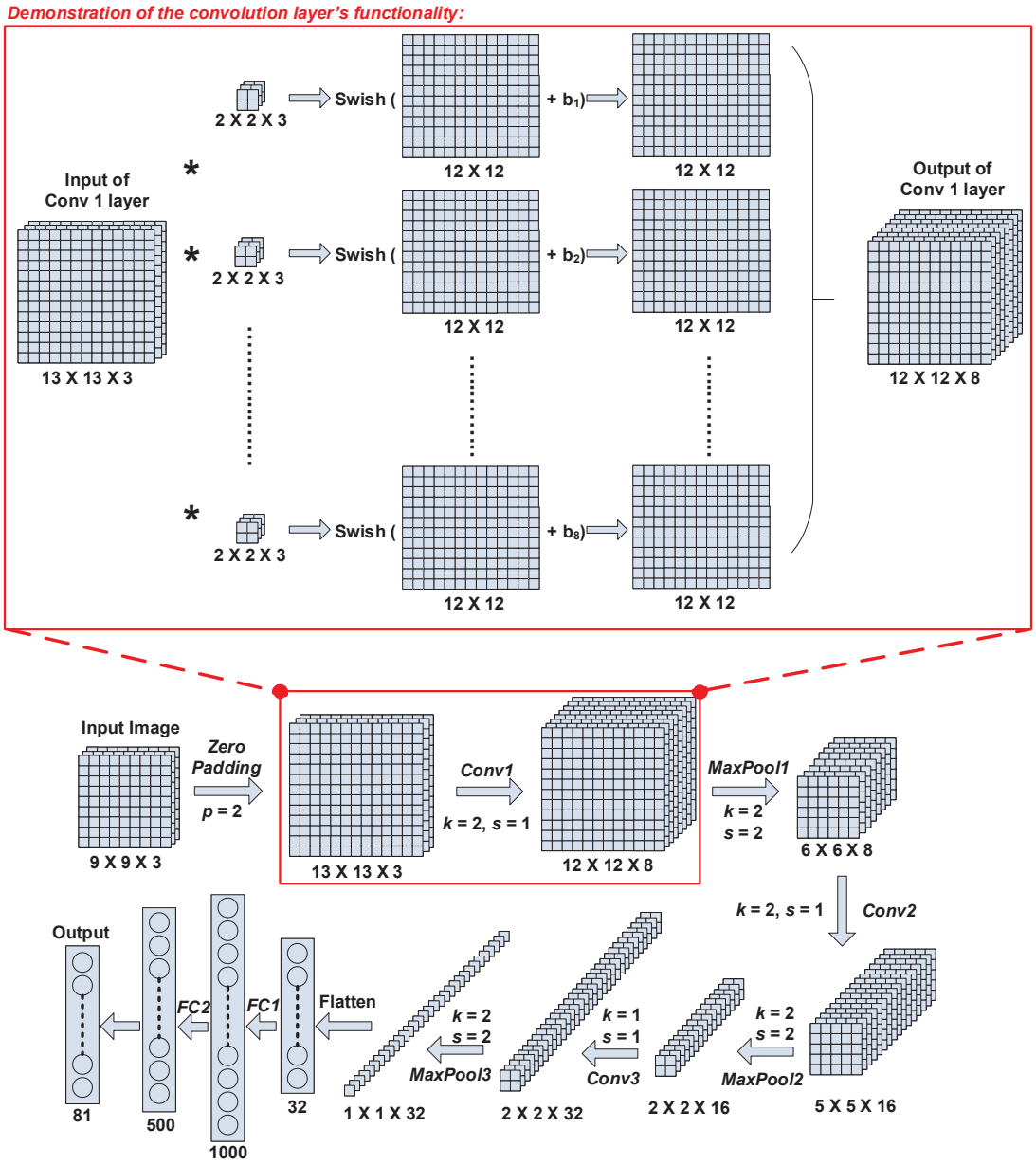

Figure 7.5: **(Bottom)** Architecture of the deep rudimentary CNN (DRC) model with **(Top)** the manifestation of convolution layer's working.

to be processed as a 81×1 dimension output vector carrying normal and under-attack cell IDs.

The hyperparameters used in our model and the above-described dimensions of various layers from input layer to the layers utilized in *Stage* 2 can be found in Figure 7.3 in the form of red and blue annotations, respectively. A softmax function [84, Sec. 4.1] is typically utilized in the output layer for a multi-class

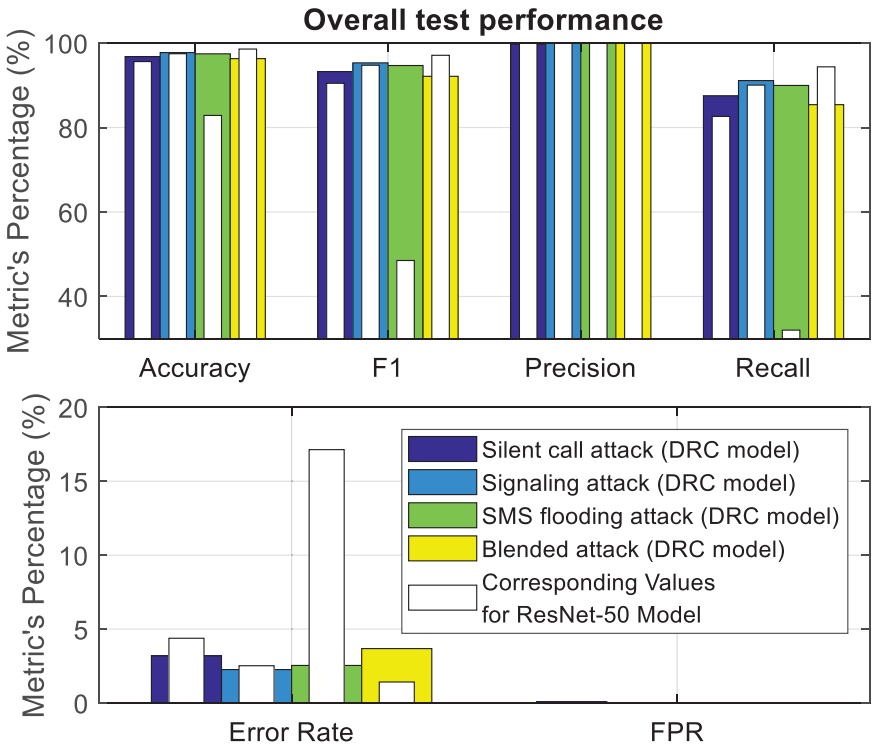

Figure 7.6: Overall test performance.

classification problem; however, since we are dealing with multi-label classification problem, we use binary cross entropy loss function.

7.6.3 Deep Rudimentary CNN Model

Keeping in view the relatively lesser input image dimensions ($9 \times 9 \times 3$ i.e. 81 pixels) that we are dealing with, we design a relatively simple, 6-layer model named as deep rudimentary CNN (DRC) model. It is built from the fundamental (convolution, pooling, and fully connected) layers extensively described in Sec. 7.6.1; and is inspired from the designs of classical models like VGG [105], AlexNet [104], and LeNet-5 [92]. It is illustrated in Figure 7.5 (Bottom).

The model takes an input image having dimension $9 \times 9 \times 3$ and expands it by padding zeros with $p = 2$ to yield a volume with dimension $13 \times 13 \times 3$. Then, the model passes the volume through a convolution layer to transform its dimension to $12 \times 12 \times 8$ (see Sec. 7.6.1 for complete details of this step).

Figure 7.7: Accuracy dispersions of our models under various attack scenarios. (a)-(d) Results of our DRC model (e) ResNet-50 model's performance for blended attack scenario, in which it outperformed our model. (f) Improvements we achieved with ResNet-50 model under the blended attack scenario.

Next, a max-pooling layer is utilized with $k = s = 2$. We can apply Eq. 7.2 to get the output volume's dimension as $6 \times 6 \times 8$. In a similar manner, the DRC model then passes the resultant volume through a series of convolution and pooling layers ($Conv2$, $MaxPool2$, $Conv3$, and $MaxPool3$), delineated in the figure. The resultant volume is finally flattened and passed through the two fully-connected layers ($FC1$ and $FC2$) to give a 81×1 dimension output vector (by utilizing binary cross entropy loss function), having identification of normal and under-attack cells.

7.7 Experimental Results and Performance Evaluation

We utilize test set $\{I_{test}, O_{test}\}$ (containing 335 images and their corresponding labels) for performance evaluation of our models under various attack scenarios, and report the results in Figure 7.6 and 7.7. Overall, for all the attack scenarios except the blended attack, our deep rudimentary CNN (DRC) model surpassed ResNet-50 model in terms of all the performance metrics, as evident in Figure 7.6. Additionally, the maximum difference in the performance between the two models is observed during the SMS flooding attack for which ResNet-50 model performed poorly; while for signaling attack both models performed in a similar fashion.

Due to the limited space, we only illustrate accuracies yielded from the various attack scenarios as heatmaps in Figure 7.7. Each 9×9 heatmap from part (a)-(e) of the figure contains accuracy values of the corresponding cells in Figure 7.2(ii). We can observe that the results are varied across the spectrum of cells depending on the individual cellular activity levels and the consequent learning of the model. Lowest accuracies for each attack scenario are marked by black ovals in the figure. We can observe that our DRC model yielded more than 91% accuracy for every

cell in the sub-grid under every attack scenario (green sub-grids).

Since, ResNet-50 model has superior performance under blended attack scenario, we also demonstrate accuracy heatmap (blue sub-grid) for our ResNet-50 model and the improvement (red sub-grid) it achieved as compared with DRC model in Figure 7.7(e) and (f), respectively. The higher performance can also be judged from the clear difference in minimum accuracy values yielded by both models (annotated with black ovals) in Figure 7.7(d) and (e): 91% for DRC model and 97% for ResNet-50 model. Additionally, it can also be observed in Figure 7.7(d)-(f) that the worst performing cell 4855 (row 7, column 3) with 91% accuracy under the blended attack scenario improves to have 97.6% by applying ResNet-50 model.

Interestingly, it can also be noted that the cells covering the Bocconi university (cell ID 4259: row 1, column 7) and the city center (cell ID 5060: row 9, column 8) have steady and high accuracy values (highlighted in black rectangles in the figure) throughout all the attack scenarios and for both models. This might be because of the relative high user activities in these popular areas during the selected (lunch) timings (from 11 am to 2 pm), for which both models were able to easily distinguish the hidden pattern and hence detected normal and under-attacked cell(s) with high accuracies; in contrast to the relatively low accuracy values for the cell covering nightlife places (ID 4456: row 3, column 4) that has relatively low user activity values during the selected timings.

7.8 Conclusion and Insights for future work

Our framework achieved higher than 91% normal and under-attack cell detection accuracy by utilizing deep rudimentary CNN (DRC) model for silent call, signaling, and SMS flooding attacks that target a cellular network to cause DDoS to the cellular connectivity-dependent legitimate devices, including the ones uti-

lized in the CPSs. The framework also attained higher than 97% accuracy for a more sophisticated blended attack, in which each puppet device performs all the three attacks, by using ResNet-50 model. Our results suggest that for an individual attack, where its effect is limited to a single user activity value modification in the CDRs, our framework employing DRC model can more effectively detect the cell ID(s) under attack as compared with utilizing a ResNet-50 model. While for the blended attack ResNet-50 model can yield better accuracy due to its very deep neural network design that can effectively learn the intricate structure in the dataset.

Upon detection, the information can then be sent from our coarse-grained analysis framework to the CPSs to trigger defensive/mitigative measures and can also be utilized to further perform fine-grained analysis [41, Sec. VI. C.]. For example, by acquiring more denser and richer under-attack cell's data including every user equipment's data, and feeding them to a feed forward deep neural network. This would heavily aid in identifying the bots/adversary devices within a short time, such as in minutes—it usually takes a month for most organizations to identify and clear the puppet devices [42, Fig. 18]. Our work can naturally fit to support mobile edge computing (MEC) paradigm [82] in cellular networks having MEC servers geographically located across the network and each server, co-located with a base station, monitoring cellular activity of a sub-grid and running our proposed framework. The benefit of such setting resides in dividing computation-intensive tasks across the network (among MEC servers), easing computation and storage for the core network. By leveraging voice CDRs, our work can be extended to detect overcharging attacks that can potentially be engineered to launch DDoS attacks [47].

Our robust framework can perform simultaneous analysis on multiple cells, depending on the size of sub-grid, due to the inherent utilization of CNN architecture. It can be scaled-up to consider a larger sub-grid; however, the computation

requirements need to be investigated keeping in view the on-line and off-line settings. As we had a limited dataset, we combined 3 hours data of 62 days and considered it as past data belonging to a 10-min slot (explained thoroughly in Sec. 7.5); in practice, historical CDR dataset is maintained for record-keeping within the cellular network and may easily be acquired. They might also yield improved results as the model would learn from the data containing same temporal characteristics (one 10-min slot instead of 18 slots).

Since many devices, including the ones utilized in CPSs, depend on cellular infrastructure and its services for connectivity—for example, IoT devices use voice services [47], wireless sensor and actuator network devices rely on Internet services [127], and machine-to-machine (M2M) communication network devices utilize SMS services [45]—our research is compatible as our framework leverages each service's usage data, and has solid applications in their security and earlier detection of DDoS attacks against them.

In conclusion, this is a pioneering study that investigated the application of CNNs for the cellular network's security in a coarse-grained manner to detect various attacks that lead to a DDoS (voice, Internet, and SMS) and achieved more than 91% accuracy—contributing to resolve an open issue of DDoS attack mitigation in cellular networks [40]. Besides the primary subscriber devices, our study has solid implications in securing cellular-dependent CPS devices (utilized in vertical industries and critical infrastructures) against cellular DDoS attacks that could serve as a beachhead or smoke screen to attack the CPS infrastructure and disrupt its services.

Chapter 8

Conclusions and Future Insights

8.1 Summary

The work presented in this thesis explored various ways to achieve an efficient, scalable, and timely detection of cell outage and situation leading towards congestion—Objective 1, defined in Section 1.4.

First, we applied a semi-supervised statistical-based algorithm in Chapter 3 for improved detection accuracy as compared with state-of-the-art and proposed a consolidated method for detecting both (outages and a situation leading towards congestion) anomalies [7]. A major limitation in our work was heightened false positive rate (FPR) which can drastically increase operational expenditures (OPEX). To overcome this and to have an efficient solution, we introduced powerful deep neural network (DNN)-based model in Chapter 4 which significantly reduced FPR and also further improved detection accuracy. We also introduced mobile edge computing (MEC) paradigm to offload computations from the core network (CN) to the edge servers (ESs) as training DNN models require heavy computational resources. Although MEC can provide relief to the CN, however, our solution suffers from scalability problem. An ES has to run a separate DNN model for each base station which could work fine if the ES oversees a few base

stations; however, if we scale the system to tens or hundreds of base stations per ES (a fair possibility since network density is increasing day by day), then the anomaly detection system will probably collapse.

MEC for anomaly detection was fully realized in Chapter 5, in which we proposed a novel framework executing deep convolutional neural networks (CNNs). We discovered that CNNs can naturally align with MEC paradigm and offer scalable and most efficient solution for the anomaly detection—surpassing the deficiency of the previous chapter. With this work, heavy computations from CN to the ESs can not just be divided but the ESs can execute the models and process data in a much faster manner. We also demonstrated scalability in this work. Finally, for a timely solution, we introduced proactive anomaly detection in Chapter 6 which forecasted cellular traffic 3 hours in advance by utilizing convolutional long short-term memory (ConvLSTM) model and detected anomalies by utilizing a subsequent feed-forward DNN model and the forecasted traffic. This yields proactiveness which could be leveraged by allocating required or extra resources in advance to the region of interest (ROI).

The work in Chapter 7 attempted to achieve secondary objective of our thesis, defined in Section 1.4. It applied the methodology of Chapter 5 involving CNNs in the context of cybersecurity and detected attacks targeting the availability of the network resources. As compared with many past studies which utilize content-based methods for the attack detection, our solution preserves user privacy and is lightweight since it utilized CDRs. It is also large-scale and expandable as it utilized CNNs that can handle anomaly detection in numerous base stations at a time.

Table 8.1 summarizes salient features of each chapter of the thesis.

Chapter #	Focused area / Objective	Explored ML/DL Models	Salient features
3	Cellular Network Management / 1	Semi Supervised statistical-based ML Model	- Introduced CDRs instead of KPIs - Consolidated method for the detection of outages and situation escalating towards congestion - Anomaly detection within an hour instead of 1 week - Detection accuracy improved from 90% to 92%
4	Cellular Network Management / 1	feed-forward deep neural network (DNN) Model	- Introduced MEC - Incorporated Internet activity feature (ignored in the past work) - Anomaly detection within 10-min duration instead of 1 hour - Detection accuracy increased to 97% and FPR reduced from previously reported 14% to 0.44%
5	Cellular Network Management / 1	Deep convolutional neural network (CNN) Models	- Framework fully compatible with MEC - Scalable, lightweight, and most efficient solution - Detects anomalies in 100 base stations at a time as compared to only one in the previous work - Offered lesser training time and higher overall test accuracy than the previous DNN-based solution
6	Cellular Network Management / 1	Convolutional long short-term memory (ConvLSTM) and DNN Models	- Traffic forecasting (3 hours in advance) - Proactive anomaly detection on predicted CDRs - Introduction of a modular framework
7	Cybersecurity / 2	Deep CNN Models	- Consolidated framework for various attack detections - Lightweight, largescale, expandable, and privacy-preserving solution

Table 8.1: Thesis progression

8.2 Research Contributions / Major Findings

Overall, our work in the thesis added the following contributions to the existing literature:

1. Explored and experimented with various supervised and semi-supervised machine learning techniques for the anomaly (both, pertaining to cellular network management and cybersecurity domains) detection in cellular networks by keeping efficiency, scalability, and timely detecting in focus.

2. Promoted call detail record (CDR) dataset utilization instead of KPIs which has multi-fold advantages: (a) we proposed consolidated frameworks for the detection of both, cellular outages and a situation leading towards congestion which were previously done separately because of different KPIs utilized for their detection; (b) since MDT dataset requires additional resources for collection, we can avoid unnecessary resource utilization by utilizing CDRs which are already present in the networks. After the coarse-grained analysis which was largely done in our work, KPIs can then be requested for a fine-grained analysis if required. (c) Unlike many studies which employ subscriber content-based algorithms and compromise user privacy, CDRs contain the activity values instead of the actual contents and the models that utilize CDRs (employed in our works) preserve subscriber privacy. (d) Better location accuracy is achieved as compared with MDT measurements, which helps in pinpointing the anomalous region.

3. Demonstrated the powerful utility and a natural combination of integrating deep convolutional neural networks (CNNs) with mobile edge computing (MEC). This enables the anomaly detection framework to be applied in a large-scale and more efficient manner.

4. Presented a consolidated framework for the detection of various availability-

related attacks towards the cellular networks.

8.3 Potential Future Research Directions

8.3.1 Transfer Learning to Tackle Data-Shortage Challenge

Throughout our work, we highlighted a limitation of having inadequate data to train on. Deep learning (DL) models require a large number of examples and their acquisition may take long observation time. An inherent assumption in the machine learning algorithms applied in this research is that the training and future instances belong to the same feature space having the same distribution. This limits the scope of the application of deep learning (DL) models to the targeted scenarios only and hinders in their generalization.

If we can elevate from the above-mentioned limitation and have two datasets from different feature spaces having different distributions: *Dataset A* containing copious examples pertaining to a base station (relevant when applying a feed-forward DNN model as done in Chapter 4) or a cluster of base stations (relevant when applying a CNN model as done in Chapter 5), and *Dataset B* containing small amount of examples pertaining to another base station or cluster of base stations. We train a DL model for anomaly detection using *Dataset A* (known as pre-training phase) and then the knowledge (learned parameters) of the trained model is transferred to another model which is utilizing *Dataset B* (known as fine-tuning phase). This will enable the latter model to train on small number of examples (*Dataset B*) but yield results as if it was trained using a dataset having copious example (*Dataset A*).

This method is known as Transfer Learning [131]. It is a promising direction to explore because learning from low-level features at the initial layers of a DL

model trained on *Dataset A* might be helpful and come handy when learning from the *Dataset B*.

In practice, an extensive data collection drive can be initiated to acquire data (denoted as *Dataset A*) from selected cell sites. The selection can be done on the basis of unique location that these sites cover so that the acquired cellular traffic data represent the user behaviors in those locations. Such cell sites may include the ones covering parks, sports venues, restaurants, hospitals, schools and universities, subway stations, places of worship, etc. A secondary short-duration drive can be launched to capture another set of base station or cluster of base stations (compiling *Dataset B*).

The benefit of this method is the significant reduction in data observation time of a base station or cluster of base stations during which the data is collected. This implies a quicker anomaly detection setup in a cellular network.

8.3.2 MEC-based Fully Proactive Anomaly Detection System

The Forecaster component in Chapter 6 accepted a single base station's activity data and yielded traffic predictions which were later utilized by a feed-forward DNN model for anomaly detection. We can integrate mobile edge computing (MEC) paradigm to scale-up the proactive anomaly detection operation by modifying the Forecaster to rather accept and output images. Then, a CNN model can be applied (like in Chapter 5) to detect anomalies in numerous base stations at a time.

8.3.3 Applicability of Anomaly Detection Frameworks for IIoT Networks

As mentioned in Section 1.3, cellular networks and industrial Internet of Things (IIoT) networks share a common goal: reduction in operational expenditures (OPEX). Efforts can be made to modify our methods, especially the ones involving MEC, for anomaly detection in the context of IIoT networks. However, this need further investigation and thorough analysis.

8.3.4 A Consolidated Alarm System for Outage, Congestion, and Cyber-attacks in Cellular Networks

We can integrate both objectives in this thesis related to cellular network management and cybersecurity to propose a single framework for the detection of outages, situation leading towards congestion, and various cyber-attacks since they can are detected by utilizing call detail record (CDR) dataset.

Bibliography

[1] A. Imran, A. Zoha, and A. Abu-Dayya, "Challenges in 5G: how to empower SON with big data for enabling 5G," *IEEE Netw.*, vol. 28, no. 6, pp. 27–33, 2014.

[2] G. Tu, C. Li, C. Peng, and S. Lu, "How voice call technology poses security threats in 4G LTE networks," in *2015 IEEE Conference on Communications and Network Security (CNS)*, 2015, pp. 442–450.

[3] B. Hussain, Q. Du, S. Zhang, A. Imran, and M. A. Imran, "Mobile edge computing-based data-driven deep learning framework for anomaly detection," *IEEE Access*, vol. 7, pp. 137 656–137 667, 2019.

[4] W. Saad, M. Bennis, and M. Chen, "A Vision of 6G Wireless Systems: Applications, Trends, Technologies, and Open Research Problems," *IEEE Network*, vol. 34, no. 3, pp. 134–142, 2020.

[5] Cisco, "Cisco Annual Internet Report (2018–2023)," *White Paper*, vol. 2020, no. 3, p. 4, March 2020.

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

[7] M. S. Parwez, D. B. Rawat, and M. Garuba, "Big Data Analytics for User-Activity Analysis and User-Anomaly Detection in Mobile Wireless Net-

work," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 2058–2065, 2017.

[8] B. Hussain, Q. Du, A. Imran, and M. A. Imran, "Artificial Intelligence-Powered Mobile Edge Computing-Based Anomaly Detection in Cellular Networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 4986–4996, 2020.

[9] Y. Li, B. Shen, J. Zhang, X. Gan, J. Wang, and X. Wang, "Offloading in HCNs: Congestion-Aware Network Selection and User Incentive Design," *IEEE Transactions on Wireless Communications*, vol. 16, no. 10, pp. 6479–6492, 2017.

[10] Ramneek, P. Hosein, W. Choi, and W. Seok, "Congestion detection for QoS-enabled wireless networks and its potential applications," *Journal of Communications and Networks*, vol. 18, no. 3, pp. 513–522, 2016.

[11] A. Humayed, J. Lin, F. Li, and B. Luo, "Cyber-Physical Systems Security - A Survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1802–1831, 2017.

[12] H. Song, D. B. Rawat, S. Jeschke, and C. Brecher, *Cyber-Physical Systems: Foundations, Principles and Applications*, 2016.

[13] 5G-PPP, "5G and the Factories of the Future," 2015, White Paper. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-White-Paper-on-Factories-of-the-Future-Vertical-Sector.pdf

[14] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Comput. Surv.*, vol. 41, no. 3, Jul. 2009.

[15] Y. Wang, Z. Wu, Q. Li, and Y. Zhu, "A Model of Telecommunication Network Performance Anomaly Detection Based on Service Features Clustering," *IEEE Access*, vol. 5, pp. 17 589–17 596, 2017.

[16] I. A. Karatepe and E. Zeydan, "Anomaly Detection In Cellular Network Data Using Big Data Analytics," in *European Wireless 2014; 20th European Wireless Conference*, 2014, pp. 1–5.

[17] W. Sun, X. Qin, S. Tang, and G. Wei, "A QoE anomaly detection and diagnosis framework for cellular network operators," in *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2015, pp. 450–455.

[18] R. Shafin, L. Liu, V. Chandrasekhar, H. Chen, J. Reed, and J. C. Zhang, "Artificial Intelligence-Enabled Cellular Networks: A Critical Path to Beyond-5G and 6G," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 212–217, 2020.

[19] I. Ahmad, S. Shahabuddin, H. Malik, E. Harjula, T. Leppänen, L. Lovén, A. Anttonen, A. H. Sodhro, M. Mahtab Alam, M. Juntti, A. Ylä-Jääski, T. Sauter, A. Gurtov, M. Ylianttila, and J. Riekki, "Machine Learning Meets Communication Networks : Current Trends and Future Challenges," *IEEE Access*, vol. 8, pp. 223 418–223 460, 2020.

[20] I. F. Akyildiz, A. Kak, and S. Nie, "6G and Beyond: The Future of Wireless Communications Systems," *IEEE Access*, vol. 8, pp. 133 995–134 030, 2020.

[21] S. Han, T. Xie, C. L. I, L. Chai, Z. Liu, Y. Yuan, and C. Cui, "Artificial-Intelligence-Enabled Air Interface for 6G: Solutions, Challenges, and Standardization Impacts," *IEEE Communications Magazine*, vol. 58, no. 10, pp. 73–79, 2020.

[22] A. L'Heureux, K. Grolinger, H. F. Elyamany, and M. A. M. Capretz, "Machine Learning With Big Data: Challenges and Approaches," *IEEE Access*, vol. 5, pp. 7776–7797, 2017.

[23] K. David and H. Berndt, "6G Vision and Requirements: Is There Any Need for Beyond 5G?" *IEEE Veh. Technol. Mag.*, vol. 13, no. 3, pp. 72–80, 2018.

[24] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y. A. Zhang, "The Roadmap to 6G: AI Empowered Wireless Networks," *IEEE Communications Magazine*, vol. 57, no. 8, pp. 84–90, 2019.

[25] B. Hussain, J. Zhang, and Q. Du, "A Prescriptive Analytics-Based Modular Framework for Proactive Cell Outage and Congestion Detection in Mobile Networks (Under-Review)," *IEEE Transactions on Network and Service Management*, vol. X, no. X, pp. XXXX–XXXX, 2021.

[26] Cisco and Jasper, "The Hidden Costs of Delivering IIoT Services: Industrial Monitoring & Heavy Equipment," April 2016, White Paper. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.pdf

[27] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0," *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 17–27, 2017.

[28] C. Tranoris, S. Denazis, L. Guardalben, J. Pereira, and S. Sargento, "Enabling cyber-physical systems for 5G networking: A case study on the automotive vertical domain," *Proc. IEEE/ACM 4th Int. Workshop Softw. Eng. Smart Cyber-Phys. Syst.*, pp. 37–40, 2018.

[29] 5G-PPP, White Papers. [Online]. Available: https://5g-ppp.eu/white-papers/

[30] S. Jeschke, C. Brecher, H. Song, and D. B. Rawat, *Industrial Internet of Things: Cybermanufacturing Systems*. Cham, Switzerland: Springer, 2017.

[31] M. Cosovic, A. Tsitsimelis, D. Vukobratovic, J. Matamoros, and C. Anton-Haro, "5G Mobile Cellular Networks: Enabling Distributed State Estimation for Smart Grids," *IEEE Commun. Mag.*, vol. 55, no. 10, pp. 62–69, 2017.

[32] F. Chernogorov, J. Turkka, T. Ristaniemi, and A. Averbuch, "Detection of Sleeping Cells in LTE Networks Using Diffusion Maps," in *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*, 2011, pp. 1–5.

[33] F. Chernogorov, S. Chernov, K. Brigatti, and T. Ristaniemi, "Sequence-based detection of sleeping cell failures in mobile networks," *Wireless Networks*, vol. 22, no. 6, pp. 2029–2048, Aug. 2016.

[34] R. Barco, P. Lazaro, and P. Munoz, "A unified framework for self-healing in wireless networks," *IEEE Communications Magazine*, vol. 50, no. 12, pp. 134–142, 2012.

[35] 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)*, 2015, pp. 203–210.

[36] K. Dangerfield, E. Bensadoun, and S. Boynton, "Rogers wireless services restored for 'vast majority' of customers after mass outage," *Global News*, April 19, 2021.

[37] "ASSH: Designing Agile and Scalable Self-Healing Functionalities for Ultra Dense Future Cellular Networks," http://www.bsonlab.com/ASSH, note = Accessed: 2021-03-25.

[38] A. Zoha, A. Saeed, A. Imran, M. A. Imran, and A. Abu-Dayya, "A SON solution for sleeping cell detection using low-dimensional embedding of MDT measurements," in *2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, 2014, pp. 1626–1630.

[39] Sandvine, "Network Congestion Management: Considerations and Techniques," 2015, White Paper. [Online]. Available: https://www.sandvine.com/hubfs/downloads/archive/whitepaper-network-congestion-management.pdf

[40] S. Mavoungou, G. Kaddoum, M. Taha, and G. Matar, "Survey on Threats and Attacks on Mobile Networks," *IEEE Access*, vol. 4, pp. 4543–4572, 2016.

[41] L. He, Z. Yan, and M. Atiquzzaman, "LTE/LTE-A Network Security Data Collection and Analysis for Security Measurement: A Survey," *IEEE Access*, vol. 6, pp. 4220–4242, 2018.

[42] Verizon, "2018 Data Breach Investigations Report, 11th ed." 2018, Research report. [Online]. Available: https://enterprise.verizon.com/resources/reports/DBIR_2018_Report.pdf

[43] I. Kolochenko, "DDoS attacks: a perfect smoke screen for APTs and silent data breaches," Sept. 28, 2015.

[44] A. Gupta, T. Verma, S. Bali, and S. Kaul, "Detecting MS initiated signaling DDoS attacks in 3G/4G wireless networks," in *2013 Fifth Inter-*

national Conference on Communication Systems and Networks (COMSNETS), 2013, pp. 1–60.

[45] I. Murynets and R. P. Jover, "Anomaly detection in cellular Machine-to-Machine communications," in *2013 IEEE International Conference on Communications (ICC)*, 2013, pp. 2138–2143.

[46] M. Khosroshahy, D. Qiu, and M. K. Mehmet Ali, "Botnets in 4G cellular networks: Platforms to launch DDoS attacks against the air interface," in *2013 International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT)*, 2013, pp. 30–35.

[47] T. Xie, C. Li, J. Tang, and G. Tu, "How Voice Service Threatens Cellular-Connected IoT Devices in the Operational 4G LTE Networks," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[48] R. Bassil, A. Chehab, I. Elhajj, and A. Kayssi, "Signaling Oriented Denial of Service on LTE Networks," in *Proceedings of the 10th ACM International Symposium on Mobility Management and Wireless Access*, ser. MobiWac '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 153–158. [Online]. Available: https://doi.org/10.1145/2386995.2387024

[49] G. Gorbil, O. H. Abdelrahman, M. Pavloski, and E. Gelenbe, "Modeling and Analysis of RRC-Based Signalling Storms in 3G Networks," *IEEE Transactions on Emerging Topics in Computing*, vol. 4, no. 1, pp. 113–127, 2016.

[50] G.-H. Tu, C.-Y. Li, C. Peng, Y. Li, and S. Lu, "New Security Threats Caused by IMS-Based SMS Service in 4G LTE Networks," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA:

Association for Computing Machinery, 2016, p. 1118–1130. [Online]. Available: https://doi.org/10.1145/2976749.2978393

[51] D. Cielen, A. D. B. Meysman, and M. Ali, *Introducing Data Science: Big Data, Machine Learning, and more, using Python tools*. Shelter Island, NY: Manning Publications, 2016. [Online]. Available: http://bedford-computing.co.uk/learning/wp-content/uploads/2016/09/introducing-data-science-machine-learning-python.pdf

[52] J. Debattista, C. Lange, S. Scerri, and S. Auer, "Linked 'Big' Data: Towards a Manifold Increase in Big Data Value and Veracity," in *2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC)*, 2015, pp. 92–98.

[53] M. Kulin, C. Fortuna, E. De Poorter, D. Deschrijver, and I. Moerman, "Data-Driven Design of Intelligent Wireless Networks: An Overview and Tutorial," *Sensors*, vol. 16, no. 6, 2016.

[54] D. Barber, *Bayesian Reasoning and Machine Learning*. Cambridge, UK: Cambridge University Press, 2016.

[55] E. Alpaydın, *Introduction to Machine Learning*. The MIT Press.

[56] S. Gollapundi, *Practical Machine Learning*. Packt Publishing Ltd.

[57] 3GPP, "Universal Mobile Telecommunications System (UMTS); LTE; Universal Terrestrial Radio Access (UTRA) and Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Measurement Collection for Minimization of Drive Tests (MDT); Overall Description; Stage 2 (3GPP TS 37.320 version 12.2.0 Release 12)," Sept., 2014, Technical Specification Report. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/137300_137399/137320/12.02.00_60/ts_137320v120200p.pdf

[58] W. A. Hapsari, A. Umesh, M. Iwamura, M. Tomala, B. Gyula, and B. Sebire, "Minimization of drive tests solution in 3GPP," vol. 50, no. 6, 2012, pp. 28–36.

[59] 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)*, 2018, pp. 206–212.

[60] O. Onireti, A. Zoha, J. Moysen, A. Imran, L. Giupponi, M. Ali 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, 2016.

[61] D. Naboulsi, M. Fiore, S. Ribot, and R. Stanica, "Large-Scale Mobile Traffic Analysis: A Survey," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 124–161, 2016.

[62] 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.

[63] B. Hussain, Q. Du, B. Sun, and Z. Han, "Deep Learning-Based DDoS-Attack Detection for Cyber-Physical System over 5G Network," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 860–870, 2021.

[64] S. Papadopoulos, A. Drosou, and D. Tzovaras, "A Novel Graph-Based Descriptor for the Detection of Billing-Related Anomalies in Cellular Mobile Networks," *IEEE Transactions on Mobile Computing*, vol. 15, no. 11, pp. 2655–2668, 2016.

[65] S. Papadopoulos, A. Drosou, I. Kalamaras, and D. Tzovaras, "Behavioural Network Traffic Analytics for Securing 5G Networks," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2018, pp. 1–6.

[66] G. Barlacchi, M. De Nadai, R. Larcher, A. Casella, C. Chitic, G. Torrisi, F. Antonelli, A. Vespignani, A. Pentland, and B. Lepri, "A multi-source dataset of urban life in the city of Milan and the Province of Trentino," *Scientific Data*, vol. 2, 2015.

[67] C. Quadri, S. Gaito, and G. P. Rossi, "Proximity-aware offloading of person-to-person communications in LTE networks," in *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2016, pp. 608–613.

[68] R. Bali, D. Sarkar, B. Lantz, and C. Lesmeister, *R: Unleash Machine Learning Techniques*. Birmingham, UK: Packt Publishing, Oct. 24, 2016.

[69] C. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag .

[70] Y. Zhang, N. Meratnia, and P. Havinga, "Outlier Detection Techniques for Wireless Sensor Networks: A Survey," *IEEE Communications Surveys Tutorials*, vol. 12, no. 2, pp. 159–170, 2010.

[71] Q. Plessis, M. Suzuki, and T. Kitahara, "Unsupervised multi scale anomaly detection in streams of events," in *2016 10th International Conference on Signal Processing and Communication Systems (ICSPCS)*, 2016, pp. 1–9.

[72] Q. Liao and S. Stanczak, "Network State Awareness and Proactive Anomaly Detection in Self-Organizing Networks," in *2015 IEEE Globecom Workshops (GC Wkshps)*, 2015, pp. 1–6.

[73] P. V. Klaine, M. A. Imran, O. Onireti, and R. D. Souza, "A Survey of Machine Learning Techniques Applied to Self-Organizing Cellular Networks," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2392–2431, 2017.

[74] C. M. Mueller, M. Kaschub, C. Blankenhorn, and S. Wanke, "A Cell Outage Detection Algorithm Using Neighbor Cell List Reports," in *Proc. International Workshop on Self-Organizing Systems (IWSOS)*, 2008, pp. 218–229.

[75] J. Turkka, F. Chernogorov, K. Brigatti, T. Ristaniemi, and J. Lempiäinen, "An Approach for Network Outage Detection from Drive-Testing Databases," *Journal of Computer Networks and Communications*, vol. 2012, pp. 1–13, 2012.

[76] F. Chernogorov, S. Chernov, and K. Brigatti, "Data Mining Approach to Detection of Random Access Sleeping Cell Failures in Cellular Mobile Networks," 2015.

[77] Y. Kumar, H. Farooq, and A. Imran, "Fault prediction and reliability analysis in a real cellular network," in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2017, pp. 1090–1095.

[78] N. Baldo, L. Giupponi, and J. Mangues-Bafalluy, "Big Data Empowered Self Organized Networks," in *European Wireless 2014; 20th European Wireless Conference*, 2014, pp. 1–8.

[79] S. Hämäläinen, H. Sanneck, and C. Sartori, *LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency.* Wiley, Jan. 2012.

[80] Q. Du, H. Song, and X. Zhu, "Social-Feature Enabled Communications Among Devices Toward the Smart IoT Community," *IEEE Communications Magazine*, vol. 57, no. 1, pp. 130–137, 2019.

[81] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436–444, May 2015.

[82] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[83] E. Ahmed, A. Ahmed, I. Yaqoob, J. Shuja, A. Gani, M. Imran, and M. Shoaib, "Bringing Computation Closer toward the User Network: Is Edge Computing the Solution?" *IEEE Communications Magazine*, vol. 55, no. 11, pp. 138–144, 2017.

[84] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[85] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Machine Learning for Wireless Networks with Artificial Intelligence: A Tutorial on Neural Networks," *ArXiv*, vol. abs/1710.02913, 2017.

[86] Y. Bengio, "Learning Deep Architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, p. 1–127, Jan. 2009. [Online]. Available: https://doi.org/10.1561/2200000006

[87] J. Patterson and A. Gibson. Sebastopol, CA, USA: O'Reilly Media, Inc., Aug. 2017.

[88] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks," in *Proceedings of the Fourteenth International Conference on*

Artificial Intelligence and Statistics, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudík, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, 11–13 Apr 2011, pp. 315–323. [Online]. Available: http://proceedings.mlr.press/v15/glorot11a.html

[89] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier Nonlinearities Improve Neural Network Acoustic Models," in *Proc. International Conference on Machine Learning (ICML)*, vol. 30, no. 1, 2013, p. 3.

[90] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for Activation Functions." [Online]. Available: http://arxiv.org/abs/1710.05941v2

[91] A. Ng, J. Ngiam, C. Y. Foo, Y. Mai, C. Suen, A. Coates, A. Maas, A. Hannun, B. Huval, T. Wang, and S. Tandon, "Deep Learning Tutorial," Stanford University. [Online]. Available: http://deeplearning.stanford.edu/tutorial/

[92] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[93] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256. [Online]. Available: http://proceedings.mlr.press/v9/glorot10a.html

[94] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," 2015.

[95] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[96] N. Buduma, *Fundamentals of Deep Learning* . Sebastopol, CA, USA: O'Reilly Media, Inc.

[97] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," 2012.

[98] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[99] B. Hussain, Q. Du, and P. Ren, "Deep Learning-Based Big Data-Assisted Anomaly Detection in Cellular Networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[100] T. Tieleman and G. Hinton, "Neural Networks for Machine Learning: Lecture 6.5 RMSProp," 2012. [Online]. Available: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

[101] A. Zoha, A. Saeed, H. Farooq, A. Rizwan, A. Imran, and M. A. Imran, "Leveraging Intelligence from Network CDR Data for Interference Aware Energy Consumption Minimization," *IEEE Transactions on Mobile Computing*, vol. 17, no. 7, pp. 1569–1582, 2018.

[102] X. Ge, S. Tu, G. Mao, C. Wang, and T. Han, "5G Ultra-Dense Cellular Networks," *IEEE Wireless Communications*, vol. 23, no. 1, pp. 72–79, 2016.

[103] "Keras," https://keras.io/, accessed: 2021-04-12.

[104] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Neural Information Processing Systems*, vol. 25, 01 2012.

[105] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1409.1556

[106] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2015.

[107] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, ser. AAAI'17. AAAI Press, 2017, p. 4278–4284.

[108] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," 2015.

[109] Y. Cai, Q. Li, and Z. Shen, "A Quantitative Analysis of the Effect of Batch Normalization on Gradient Descent," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 882–890. [Online]. Available: http://proceedings.mlr.press/v97/cai19a.html

[110] F. Tacchino, C. Macchiavello, D. Gerace, and D. Bajoni, "An artificial neuron implemented on an actual quantum processor," *Nature partner journals (npj) Quantum Information*, vol. 5, no. 26, pp. 1–8, Mar. 2019.

[111] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *J. Mach. Learn. Res.*, vol. 13, no. null, p. 281–305, Feb. Jan. 2012.

[112] Z. Liao, J. Wang, S. Zhang, J. Cao, and G. Min, "Minimizing Movement for Target Coverage and Network Connectivity in Mobile Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 7, pp. 1971–1983, 2015.

[113] M. Peng, Y. Sun, X. Li, Z. Mao, and C. Wang, "Recent Advances in Cloud Radio Access Networks: System Architectures, Key Techniques, and Open Issues," *IEEE Communications Surveys Tutorials*, vol. 18, no. 3, pp. 2282–2308, 2016.

[114] M. Aazam, S. Zeadally, and K. A. Harras, "Deploying Fog Computing in Industrial Internet of Things and Industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4674–4682, 2018.

[115] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[116] A. Gharaibeh, M. A. Salahuddin, S. J. Hussini, A. Khreishah, I. Khalil, M. Guizani, and A. Al-Fuqaha, "Smart Cities: A Survey on Data Management, Security, and Enabling Technologies," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2456–2501, 2017.

[117] O. P. Kogeda and J. I. Agbinya, "Proactive Cellular Network Faults Prediction Through Mobile Intelligent Agent Technology," in *The 2nd International Conference on Wireless Broadband and Ultra Wideband Communications (AusWireless 2007)*, 2007, pp. 55–55.

[118] 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," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 802–810.

[119] J. Wu, Y. Zhang, M. Zukerman, and E. K. Yung, "Energy-Efficient Base-Stations Sleep-Mode Techniques in Green Cellular Networks: A Survey," *IEEE Communications Surveys Tutorials*, vol. 17, no. 2, pp. 803–826, 2015.

[120] C. Zhang, H. Zhang, J. Qiao, D. Yuan, and M. Zhang, "Deep Transfer Learning for Intelligent Cellular Traffic Prediction Based on Cross-Domain Big Data," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1389–1401, 2019.

[121] O. G. Manzanilla-Salazar, F. Malandra, H. Mellah, C. Wetté, and B. Sansò, "A Machine Learning Framework for Sleeping Cell Detection in a Smart-City IoT Telecommunications Infrastructure," *IEEE Access*, vol. 8, pp. 61 213–61 225, 2020.

[122] M. Mozaffari, W. Saad, M. Bennis, Y. H. Nam, and M. Debbah, "A Tutorial on UAVs for Wireless Networks: Applications, Challenges, and Open Problems," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2334–2360, 2019.

[123] Department of Defense, USA, "Summary of the 2018 DoD artificial intelligence strategy," Feb. 2019, Report. [Online].

Available: https://media.defense.gov/2019/Feb/12/2002088963/-1/-1/1/SUMMARY-OF-DOD-AI-STRATEGY.PDF

[124] C. S. Wickramasinghe, D. L. Marino, K. Amarasinghe, and M. Manic, "Generalization of Deep Learning for Cyber-Physical System Security: A Survey," in *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, 2018, pp. 745–751.

[125] E. Ernst, R. Merola, and D. Samaan, "Economics of Artificial Intelligence: Implications for the Future of Work," *International Labour Organization (ILO) Research Paper Series*, 2018.

[126] N. Ruan, Q. Hu, L. Gao, H. Zhu, Q. Xue, W. Jia, and J. Cui, "A Traffic Based Lightweight Attack Detection Scheme for VoLTE," in *2016 IEEE Global Communications Conference (GLOBECOM)*, 2016, pp. 1–6.

[127] J. ho Bang, Y.-J. Cho, and K. Kang, "Anomaly detection of network-initiated LTE signaling traffic in wireless sensor and actuator networks based on a Hidden semi-Markov Model," *Computers & Security*, vol. 65, pp. 108–120, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404816301614

[128] B. Thomas, "CityPolulation.de," https://www.citypopulation.de/php/italy-lombardia.php?cityid=015146, accessed: 2021-04-13.

[129] The World Bank Group, "Mobile cellular subscriptions (per 100 people)," https://data.worldbank.org/indicator/IT.CEL.SETS.P2?view=map&year=2014, accessed: 2021-04-13.

[130] X. An and G. Kunzmann, "Understanding mobile Internet usage behavior," in *2014 IFIP Networking Conference*, 2014, pp. 1–9.

[131] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.