



UNIVERSITY OF NAIROBI

**MULTI FACTOR BASED MOBILE VOICE AND DATA TRAFFIC
FORECASTING USING ARTIFICIAL NEURAL NETWORKS**

BY

THOMAS W. LUTI

F56/82353/2015

**A dissertation submitted in partial fulfilment of the requirements for
the award of the degree of Master of Science in Electrical and
Electronic Engineering of the University of Nairobi**

2022

DECLARATION OF ORIGINALITY

I declare that this dissertation is my original work and has not been submitted elsewhere for examination, award of a degree or publication. Where other people's works, or my own work has been used, this has properly been acknowledged and referenced in accordance with the University of Nairobi requirements.

Sign.....

Date...10th November 2022.....

Thomas W. Luti

F56/82353/2015

Department of Electrical & Information Engineering

University of Nairobi

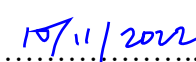
This dissertation is submitted with our approval as research supervisors.

Sign

Date

Prof. H. A. Ouma

.....

.....

Department of Electrical & Information Engineering

University of Nairobi

P.O. Box 30197-00100

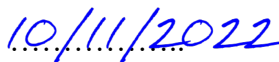
Nairobi, Kenya

Sign

Date

Prof. G. N. Kamucha

.....

.....

Department of Electrical & Information Engineering

University of Nairobi

P.O. Box 30197-00100

Nairobi, Kenya

ACKNOWLEDGEMENTS

Firstly, I would extend appreciation and thanks to my family, especially my Mother for their unwavering support.

My utmost thanks extended to all my academic supervisors, Prof. H.A. Ouma and Prof. G. N. Kamucha, for guiding me through the development of this work. Thank you.

ABSTRACT

As traffic levels in a mobile network increase, it is important for the Mobile Network Operators to adequately forecast future traffic demands. Strategic planning by the service provider, will result in timely rollout the required capacity for adequate future service provision which will alleviate network congestion and ultimate failure of the systems in place. This will ensure that acceptable Quality of Experience (QoE) and Quality of Service (QoS) is achieved by the deployment and rollout of systems and equipment and optimization of radio access resources in a timely manner.

In this research, artificial neural networks were used to forecast mobile traffic demands. The artificial neural network (ANN) was trained on input data consisting of a combination of quantified values of factors influencing mobile voice and data traffic together with the related target data variables. The taught artificial neural network was subsequently analysed on its ability to make forecasts based on the relationships obtained during training of the network. The ANN was used to predict the network busy hour for voice and data traffic per Long Term Evolution (LTE) relay nodes, Radio Network Controller (RNC) and Base Station Controller (BSC) and which could be used to determine the maximum demand for network resources that the network would be subjected to and how robust the system was to traffic growth and the capacity designs that need to be implemented to ensure continuity of service within QoS and QoE limits. The data was sourced from one local telecommunication service operator and the research considered various network nodes (BSC, RNC, LTE relay nodes) and cellular voice and data service technologies. Data from 21 BSC, 10 RNC and 1 LTE nodes was used.

The research involves the use of multiple factors as inputs to the neural networks that can be used to establish additional non-linear relationships to the associated targets and therefore improve the accuracy of forecasts. The results of the performance of the trained networks had average MSE values that ranged from 0.950694 to 0.982268 for cellular traffic forecasts (voice and data) and for prediction of the bouncing busy hour (BBH), the range spread from 0.049477 for LTE Data BBH and a maximum of 0.67827 for RNC voice (BBH) which demonstrate the accuracy and applicability of the use of ANN in the forecasting of cellular traffic and associated busy hours.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
LIST OF FIGURES.....	viii
LIST OF TABLES	ix
\LIST OF ABBREVIATIONS AND ACRONYMS.....	x
CHAPTER 1.....	1
INTRODUCTION.....	1
1.1 BACKGROUND	1
1.2 PROBLEM STATEMENT	4
1.3 OBJECTIVES.....	5
1.3.1 Overall Objective	5
1.3.2 Specific Objectives.....	5
1.4 JUSTIFICATION FOR THE STUDY	6
1.5 LIMITATIONS.....	6
1.6 THESIS LAYOUT	7
CHAPTER 2.....	8
LITERATURE REVIEW	8
2.1 MOBILE NETWORK PERFORMANCE	8
2.2 THE CELLULAR MOBILE SERVICES MARKET	9
2.3 MOBILE TRAFFIC FORECASTING	10
2.4 ARTIFICIAL NEURAL NETWORKS.....	11
2.4.1 Architecture of Neural Networks	15
2.4.2 The Learning Process (Training)	16
2.5 SURVEY OF EARLIER AND RELATED WORK.....	25

2.6 THE CASE FOR THE UTILIZATION OF ARTIFICIAL NEURAL NETWORKS IN MOBILE TRAFFIC FORECASTS	29
2.7 RESEARCH GAP	30
2.8 SUMMARY	31
CHAPTER 3.....	33
MATERIALS AND METHODS	33
3.1 DATA SOURCING.....	33
3.2 ARTIFICIAL NEURAL NETWORK DESIGN	33
3.4 PERFORMANCE METRICS	36
3.5 PERFORMANCE OF THE ARTIFICIAL NEURAL NETWORK.....	37
CHAPTER 4.....	38
DATA ORGANIZATION AND CHARACTERISTICS	38
4.1 NETWORK INPUT	38
4.2 CLUSTERING OF INPUT DATA	41
4.3 NETWORK TARGETS	43
4.3.1 Daily Traffic Trends.....	43
4.3.2 Hourly Traffic Trends.....	47
4.4 SUMMARY	55
CHAPTER 5.....	56
DESIGN OF THE ARTIFICIAL NEURAL NETWORK	56
5.1 DESIGN STEPS.....	56
5.2 ASSEMBLING OF THE TRAINING DATA	57
5.3 CREATION OF THE NETWORK OBJECT.....	58
5.4 SELECTION OF OPTIMAL ALGORITHM AND ANN ARCHITECTURE.....	59
CHAPTER 6.....	61
RESULTS AND DISCUSSION	61

6.1 CELLULAR TRAFFIC FORECASTING	61
6.1.1 Initial Training, Validation and Testing.....	61
6.1.2 Fine testing and final selection of neural network modules.....	66
6.1.3 Discussion of Results for Artificial Neural Network Performance on Cellular Traffic Data....	68
6.2 BOUNCING BUSY HOUR (BBH) FORECASTING.....	70
6.2.1 Training, Validation and Testing.....	70
6.2.2 Discussion of Results for Artificial Neural Network Performance on Bouncing Busy Hour (BBH) Data	75
6.3 SUMMARY OF RESULTS	75
CHAPTER 7.....	78
CONCLUSION AND RECOMMENDATIONS	78
7.1 CONCLUSION	78
7.2 RECOMMENDATION FOR FUTURE WORK.....	80
REFERENCES	82
BIBLIOGRAPHY	87
APPENDIX	88
A1 INPUT AND TARGET MATRICES	88
A2 SELF-ORGANIZING MAPS (SOM) PLOTS	91
A3 PERFORMANCE RESULTS FOR BEST SIMULATION RUNS.....	95
A4 PROGRAM LISTING	99

LIST OF FIGURES

Figure 2.1: Basic Structure of an Artificial Neural Network.....	13
Figure 2.2: Block Diagram of a Neural Network.....	14
Figure 2.3: A Feed Forward Neural Network.....	15
Figure 2.4: A Recurrent Neural Network.....	16
Figure 2.5: Connectivity Pattern of a Neural Network.....	21
Figure 4.1: Sample Hits Plot for Input Variables for BSC Voice Traffic.....	42
Figure 4.2: BSC Voice Traffic Input Variables Weight Planes.....	43
Figure 4.3: 2G Daily Voice Traffic.....	44
Figure 4.4: 3G Daily Voice Traffic.....	45
Figure 4.5: 2G and 3G Voice Traffic Trend.....	45
Figure 4.6: 3G Daily Data Traffic.....	46
Figure 4.7: 3G Data Traffic Trend.....	46
Figure 4.8: LTE Daily Data Traffic.....	47
Figure 4.9: 2G Hourly Voice Traffic Trend per Day.....	48
Figure 4.10: 3G Hourly Voice Traffic Trend per Day.....	48
Figure 4.11: 3G Hourly Data Traffic Trend per Day.....	49
Figure 4.12: LTE Hourly Data Traffic Trend per Day.....	49
Figure 4.13: 2G Hourly Voice Traffic Trend per Week.....	50
Figure 4.14: 3G Hourly Voice Traffic Trend per Week.....	51
Figure 4.15: 3G Hourly Data Traffic Trend per Week.....	51
Figure 4.16: LTE Hourly Data Traffic Trend per Week.....	52
Figure 4.17: 2G Voice Traffic BBH Distribution.....	53
Figure 4.18: 3G Voice Traffic BBH Distribution.....	53
Figure 4.19: 3G Data Traffic BBH Distribution.....	54
Figure 4.20: LTE Data Traffic BBH Distribution.....	54

LIST OF TABLES

Table 2.1: Pre- fired Hamming Distance Truth Table.....	14
Table 2.2: Post-fired Hamming Distance Truth Table.....	15
Table 4.1: Special Days	39
Table 4.2: Base Station Controllers.....	40
Table 4.3: Radio Network Controllers.....	40
Table 5.1: Input and Target Matrices for Cellular Traffic Prediction.....	58
Table 5.2: Input and Target Matrices for BBH Prediction.....	58
Table 6.1: BSC Voice Training Algorithm Performance.....	62
Table 6.2: RNC Voice Traffic Training Algorithm Performance.....	63
Table 6.3: RNC Data Training Algorithm Performance.....	64
Table 6.4: LTE Data Training Algorithm Performance.....	65
Table 6.5: Summary of Initial Testing Traffic Training Algorithm Performance.....	65
Table 6.6: BSC Voice Traffic ANN fine testing results.....	66
Table 6.7: RNC Voice Traffic ANN fine testing results.....	67
Table 6.8: RNC Data Traffic ANN fine testing results.....	67
Table 6.9: LTE Data Traffic ANN fine testing results.....	68
Table 6.10: Summary of Fine Testing Traffic Training Algorithm Performance.....	69
Table 6.11: BBH Distribution per Technology.....	70
Table 6.12: BSC Voice BBH Forecasting Tests.....	72
Table 6.13: RNC Voice BBH Forecasting Tests.....	73
Table 6.14: RNC Data BBH Forecasting Tests.....	73
Table 6.15: LTE Data BBH Forecasting Tests.....	74
Table 6.16: Summary of BBH Training Algorithm Performance.....	75
Table 7.1: Recommended network design per data category	79

ABBREVIATIONS AND ACRONYMS

2G, 3G, 4G, 5G	Second, Third, Fourth, Fifth Generation
ANN	Artificial Neural Networks
BER	Bit Error Rate
BR	Bayesian Regularization
BS	Base Station
BSC	Base Station Controller
BTS	Base Transceiver Station
CA	Communications Authority of Kenya
GSM	Global System for Mobile Communications
IPTV	Internet Protocol Television
ISP	Internet Service Provider
ITU	International Telecommunications Union
KPI	Key Performance Indicators
LTE	Long Term Evolution
LM	Levenberg-Marquardt
MNO	Mobile Network Operator
NN	Neural Network
OTT	Over The Top
QoE	Quality of Experience
QoS	Quality of Service
SCG	Scaled Conjugate Gradient
SDCCH	Stand-alone Dedicated Control Channel
TCP/IP	Transport Control Protocol/Internet Protocol

TCH	Traffic Channel
TRX	Transceiver
UE	User Equipment

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Mobile cellular networks have become one of the biggest infrastructures known to man with millions of components. Many individuals and businesses depend on them for daily communication activities. With the increase in uptake of these services, QoS and QoE have emerged with more pressure being put on Mobile Network Operators (MNOs) to expand and optimize their networks in order to meet customer requirements. The development of an appropriate hardware deployment and optimization plan is important to accommodate the steady increase of subscribers coupled with the increase of end user devices that are resource hungry. There is pressure on the service providers to rollout ICT infrastructure and services that will provide seamless QoS and QoE commensurate with the increased demand.

The planning and operation of a mobile network operating business needs a suitable model for mobile traffic forecasting. Mobile traffic forecasts are pivotal in helping a mobile services operator to make important decisions on infrastructure development, upgrading of existing modules and optimization of resources. In Kenya, market demand projections have been the main means of predicting cellular traffic demands. However, from a technical perspective, this approach has proved to be inaccurate with the result that the service providers pursue a reactive approach in terms of meeting QoS and QoE requirements as outlined in their operating licence conditions.

In ITU recommendation E.800, QoS is specified as “Totality of characteristics of a telecommunication service that bear on its ability to satisfy stated and implied needs of the user of the service.” [1] while ITU recommendation G.100, QoE is specified as “The overall acceptability of an application or service, as perceived subjectively by the end-user.” which may be influenced by context and user expectation.

QoS of mobile services is measured through Key Performance Indicators (KPIs) that benchmark network coverage, quality and performance of voice, SMS and mobile data. Compliance of each mobile network operator (MNOs) to the prescribed QoS standards, require that they meet the target KPIs.

A major challenge that has led to the MNOs continued non-compliance is based on three of the KPIs set out in their licence conditions. These KPIs are:

1. Call Set-up Success Probability. This KPI is influenced by a breakdown in the call setup process which could be due to a number of causes such as poor network coverage, interference and attenuation of the mobile signal due to fading and path loss. These causes can result in degraded signal levels and increased Bit Error Rate (BER) and therefore poorer signal quality. A low value of this KPI indicates inadequate network capacity.
2. Call Blocking Probability which is caused by traffic channels not being available even after a Stand-alone Dedicated Control Channel (SDCCH) has been allocated to a call. A high value of this KPI indicates inadequate network capacity.
3. Call Completion Probability. A call is termed to be complete when it undergoes the normal channel release mechanism once the call has been terminated by either party. If a condition occurs that results in call ending prematurely, then it is deemed to affect this KPI. A low value of this KPI indicates inadequate network capacity.

These particular KPIs are major indicators of inadequate capacity in the radio access networks in terms of telecommunications hardware and frequency resources. These inadequacies come in a number of forms such as failed handovers, blocked calls, dropped calls, poor network coverage, poor voice call and browsing quality. This means that the network has inadequate capacity to meet the threshold for call setup success probability and call completion probability. Therefore, in order to improve these particular KPIs:

- i. The number of cells must be increased to accommodate issues like reduced signal levels in cluttered and multipath environments.
- ii. Rollout and activation of more Transceivers (TRXs) can reduce the block rate.

- iii. Optimization of frequency reuse plans can greatly reduce interference levels from adjacent or co-channels.

Further as per the industry regulators in many countries, the MNO is required to maintain a certain threshold for QoS KPIs. The Kenyan regulator, the Communications Authority of Kenya (CA), has placed these KPIs in the conditions laid out in the licences awarded to the MNOs. Not meeting the set out thresholds will be deemed a contravention of these conditions and may result in severe penalties imposed on the MNOs. A network planner and optimizer therefore, can adequately plan for future network expansion by monitoring these KPIs and considering how radio resources and capacity can be optimally utilized to cater for future system capacity shortfalls.

Mobile networks will face an exponential increase in capacity requirements over the next few years due to increase in demand for traditional voice and data services as well as new and emerging technologies which have come about as a result of convergence of technologies such as Internet Protocol Television (IPTV) and Over the Top (OTT) services.

Engineers previously based their arguments for rolling out additional capacity by analysing traffic patterns vis a vis network capacity to develop a rollout strategy. In the past, network planners have solely utilized cell by cell database analysis techniques and spreadsheet analysis to do traffic forecasting. The approach of using spreadsheet is a lengthy process and requires a lot of preparation. Additionally, cell by cell analysis more often than not, does not consider surrounding areas or cells. Furthermore, the accuracy of the predictions is by large, reliant on the experience and skills of the network planner. A lot of step by step analysis was required which took up a lot of preparation [2]. This method required the Engineer to be skilled and experienced in network traffic dynamics to adequately give a prediction of the network capacity that would be required in future. Further, this approach was limited to small geographic areas of coverage which do not give a holistic picture of the total capacity shortfall of a mobile network.

Mobile traffic forecasting cannot be overstated in its importance on such a network, where it is important to achieve and maintain acceptable service quality and also weigh against the time involved and cost in installing additional transceivers or commissioning new base stations [2]. This generally has the result of optimizing the balance between an acceptable network-wide grade of service and capital expenditure on network hardware. For example, rollout plans can be put in place in advance to the time that a shortfall in network capacity is anticipated. Market analytics indicate significant increase in mobile traffic quantities and mobile network operators upgrading existing networks as well as the parallel rolling out of new communications infrastructure in a bid to meet the network capacity demands [3].

Along with predictions of growth of revenues and market share, traffic forecasts are of paramount importance to mobile network operators and telecommunications equipment vendors to include in their business planning and strategies. This is because a significant portion of their expenses will go into the rollout of networks and the capacity upgrades of existing infrastructure [3]. The importance of a methodology for forecasting the future traffic levels cannot be overstated as the turnaround time for upgrading a network is on average 2 to 4 months [4]. Therefore, it can be clearly observed that inadequate planning for traffic levels can lead to congestion which will severely affect QoS and QoE as well as user perception and by extension loss of business and revenue for the MNO.

Therefore, a network traffic forecasting tool is a powerful means to continually mitigate future shortfalls in poor QoS and QoE by mobile service subscribers.

1.2 PROBLEM STATEMENT

Mobile traffic determines the major source of revenue for mobile network operators and is therefore an important variable that needs to be determined with a degree of accuracy. Mobile traffic forecasting is a major and fundamental process in the mobile network operations and planning which requires the precise forecast of the amount and physical settings of mobile traffic during the various intervals (usually hours) of the planning horizon. Some knowledge of imminent traffic levels is therefore necessary. Improper dimensioning of resource capacity will

result in subscribers being subjected to degraded QoS or the operators overspending on unnecessary infrastructure. Mobile network operators therefore need a means or method that enables them to forecast mobile traffic in order to augment its planning formulation make its management more efficient. Precise forecasting of mobile traffic is problematic, as it is largely influenced by variables that are "uncertain" and which are not directly related to final traffic load.

The process of analysis of areas that require dimensioning of ICT resources is long and arduous. based mostly on reactive responses to reduced levels of QoS and also based on economic factors that would result in optimizing sales volumes rather than providing seamless levels of network quality across the country. Coupled with the lengthy budgeting and approval process for rolling out new infrastructure, the MNOs would require a tool to determine, with a fair degree of accuracy, projected cellular traffic volumes and required resources in ample time to allow proper planning and implementation to prevent degraded service levels in their networks. This will result in increased customer satisfaction levels as well as increased compliance to ascribed QoS conditions as set out in the operating licences.

1.3 OBJECTIVES

1.3.1 Overall Objective

The purpose in this work was investigation of the utilization of artificial neural networks in the forecasting of cellular traffic volumes and further, to develop software based artificial neural network module that would forecast mobile voice and data traffic.

1.3.2 Specific Objectives

The specific objectives were three-fold, namely;

- i. To determine the factors influencing cellular traffic volumes and incorporate them as inputs to improve the accuracy of forecasts.
- ii. To implement a neural network based forecasting model for cellular traffic.
- iii. To validate the accuracy of the method by comparing the predicted values with the actual values.

1.4 JUSTIFICATION FOR THE STUDY

Statistics from the CA Quality of Service reports show that the existing mobile networks are not up to par with the prevailing KPI requirements. The specific KPIs that are affected can be improved by properly dimensioning network resources according to the mobile traffic loading. This is enshrined in the operating licence conditions of the three MNOs that operate under the regulatory arm of the CA. Failure to meet the minimum required KPI thresholds encourages a penalty of 0.1% of annual gross turnover for not meeting the QoS conditions. With the review of the QoS framework to include SMS and data services in addition to the voice KPIs and ancillary QoE metrics, this figure is set to be revised to 0.2 % of their annual gross turnover.

A method of forecasting future network traffic is important for the purpose of technical and financial planning. The findings from the research and the creation and implementation of the ANN based forecasting tool will benefit the Kenyan mobile network operators by allowing them to develop schemes for network expansion planning. Further, the research findings will aid the ICT industry regulator, the CA, to develop network rollout targets and obligations to fill the communication access gaps being faced in the country by integrating the forecasts with the Universal Access objectives and plans. It is therefore justified to develop a methodology and tool for giving a large scale traffic forecast to enable the planning engineers as well as the financial and budgeting experts to plan for future network expansion.

1.5 LIMITATIONS

The greatest challenge was the availability of consistent and exhaustive data for training, validation and testing of the ANN. To enable an artificial neural network to make predictions with a high degree of accuracy, a large amount of historical mobile traffic data is required. Mobile network operators have different policies on storage and archiving of such data which is a pitfall in streamlining data for similar periods of time from the operators. Different levels of prediction accuracy are therefore expected based on this scenario. No simulated data is used to train or validate data in the ANN. However, due to the diversity of the equipment and systems deployed by the mobile network operators, there is a challenge in obtaining data that is synchronous across all networks. Additionally, deployment of technologies may be done at

different times and locations by different operators and with different equipment depending on their individual rollout plans. Further, other data like demographic indicators was unavailable for a variety of contexts or in a form that can be fed into the neural network system.

Some data inputs which were found to have been sampled at longer intervals have to be normalized over a long period which is unsuitable for making accurate short term mobile traffic forecasts. As a result, the data has to be graphed and data points obtained by interpolation. This in turn compromised the accuracy of the forecast.

Further, variables that are continuous in value must be converted to discrete values for input into the neural network.

1.6 THESIS LAYOUT

The rest of the thesis is arranged as follows:

In Chapter 2, a review on the literature of mobile traffic forecasting is conducted as well as an analysis of artificial neural networks. In Chapter 3, cellular voice and traffic data from a mobile network operator is analyzed and interpreted through graphing and self-organizing map (SOM) plots. In Chapter 4, the step by step process of designing the artificial neural network is documented which includes the training, validation and testing of the networks. In Chapter 5, the results are analyzed and discussed. Chapter 6 summarizes this thesis by giving a review of the study in the preceding chapters and the outcomes of the research. Finally, Chapter 7 identifies some problems and recommendations for work that can be done in the future in this area of research.

CHAPTER 2

LITERATURE REVIEW

2.1 MOBILE NETWORK PERFORMANCE

A typical mobile wireless network is composed of different modules and nodes, transmission equipment as well as radio access and spectrum. Typically, it is composed of Node-B (3G/4G, 5G) and Base Transceiver Stations (BTS) (2G) distributed across a service area and they act as the first point of connection between a mobile terminal user and the mobile network. Each BTS or Node-B has the capability to handle a specified number of calls simultaneously above a specified threshold of QoS before voice and data quality degenerates due to exhaustion of resources.

With the growth and expansion of a network, the cellular traffic grows and there is increased need for new network solutions to meet the expected quality of service and experience. The MNO therefore has to come up with an optimization strategy solution which may be a combination of: [2] [5]

- a) Underlaid-overlaid cells
- b) Micro-cells and femto cells for ‘hot-spots’
- c) Data offloading to WiFi Networks
- d) New radio channels added to a cell
- e) Addition of new spectrum resources and re-farming
- f) Dual-band operation (GSM 900/1800)
- g) Cell splitting (additional base-stations)

Every base station (BS) is commissioned with transceivers (TRXs), which dictate the maximum capacity of traffic that can be carried. As standard practice, when a new BS is rolled out, it would normally be commissioned with the minimum number of transceivers and more are added after the cell traffic is analysed and traffic congestion is noted.

Network congestion has been aptly defined as “state of network elements (e.g. switches, concentrators, cross-connects and transmission links) in which the network is not able to meet the

negotiated network performance objectives for the already established connections and/or for the new connection requests. It is a network state in which performance degrades due to the saturation of network resources, such as communication links, processor cycles, and memory buffers” [1].

Network capacity shortfall is characterized by network congestion which can be due to different causes. Such causes could be congestion of the Paging Channel (PCH), Stand-Alone Dedicated Control Channel (SDCCH), Random Access Channel (RACH), Access Grant Channel (AGCH) or Traffic Channel (TCH). However, the most likely causes of congestion are due to TCH and SDCCH congestion as the other channels are usually configured with large capacities hence are unlikely to undergo congestion.

The two main types of congestion that lead to degradation of QoS are therefore SDCCH and TCH congestion

i. SDCCH Congestion

This is defined as the failure to access an SDCCH during set up. The SDCCH is used to provide a reliable connection in GSM for signalling and Short Message Services and is used for location updating, mobile terminal authentication and assignment of TCH during idle periods [1]. Unavailability of the SDCCH results in unsuccessful attempts in call setup.

ii. TCH Congestion

This is defined as the failure to access a TCH. The TCH carries user speech or data at either full or half duplex depending on the configuration preferred. There are eight radio channels in a radio frequency carrier and most are configured to be traffic channels [1]. Unavailability of such channels in the call setup or handover phase results in TCH congestion.

2.2 THE CELLULAR MOBILE SERVICES MARKET

The exponential growth of subscriber numbers has led to overload of the mobile networks leading to degraded QoS and QoE. It is therefore of increasing importance for MNOs to improve their QoS and QoE and by extension their capacity to retain their customers and acquire

additional subscribers in order to secure and grow their revenues. Mobile traffic investigation and optimization is therefore of paramount importance for network expansion planning [6].

User perception is guided by the QoS and QoE. The KPIs are on the service mobility (handover success rate), accessibility (Call Setup Success rate, call blocking rate, call set-up time), retention (call drop rate, call completion rate), speech quality and availability. With the evolution of new technologies and services based on mobile broadband, there has been additional focus on the KPIs that are related to the performance of packet switched (PS) networks such as network jitter and latency.

Delivery of acceptable QoS has been a challenge for mobile operators as there has been a mismatch in the growth of the network capacity as compared to the exponential growth of subscriber bases which leads to traffic congestion and consequently poor KPIs, user experience, and customer dissatisfaction [6].

Forecasting traffic growth and how future demands will impact a network and by extension the KPIs is therefore a cornerstone in long term technical and financial plan formulation.

2.3 MOBILE TRAFFIC FORECASTING

A major indicator for measuring the utility rate and load of telecommunications equipment is the traffic. The accurate forecast of traffic is therefore important for network planning and optimization [6].

In the past, planning for future mobile traffic was often done by exponential or linear estimations centred on past data. These methods were not adequately accurate and gave significant deviations from the actual data [6]. Different factors have been identified to affect mobile traffic [7]. These include:

- 1) Day of the week /time of the day
- 2) Number of users
- 3) Change of tariff levels
- 4) Presence of service offers or promotions
- 5) Technology trends

- 6) Demography i.e. age, population distribution.
- 7) Economic factors e.g. GDP
- 8) Historical traffic volumes

Refined techniques of sampling and presenting such data in a format and frequency, that complements analysis of cellular mobile traffic, are unavailable. However, with the emergence of Big Data Analytics, the process of integrating social and economic indicators as an input in the analysis of short term fluctuations in mobile traffic trends will improve the accuracy of results and shorten the forecast horizon.

2.4 ARTIFICIAL NEURAL NETWORKS

A promising component of Artificial Intelligence (AI) research is centred on Artificial Neural Networks (ANN). It is described as a “promising new generation of information-processing systems that demonstrates the ability to learn, recall, and generalize from training patterns or data” [8]. Artificial neural networks are analogous to biological neural networks in their performance and characteristics. The artificial neuron was developed in a bid to reproduce the human neuron. Data is input into the neuron and is multiplied by its respective synaptic weights after which it is added and computed using an activation function which bounds or suppresses the output of the neuron [9]. A simple configuration of an ANN is demonstrated in Figure 2.1.

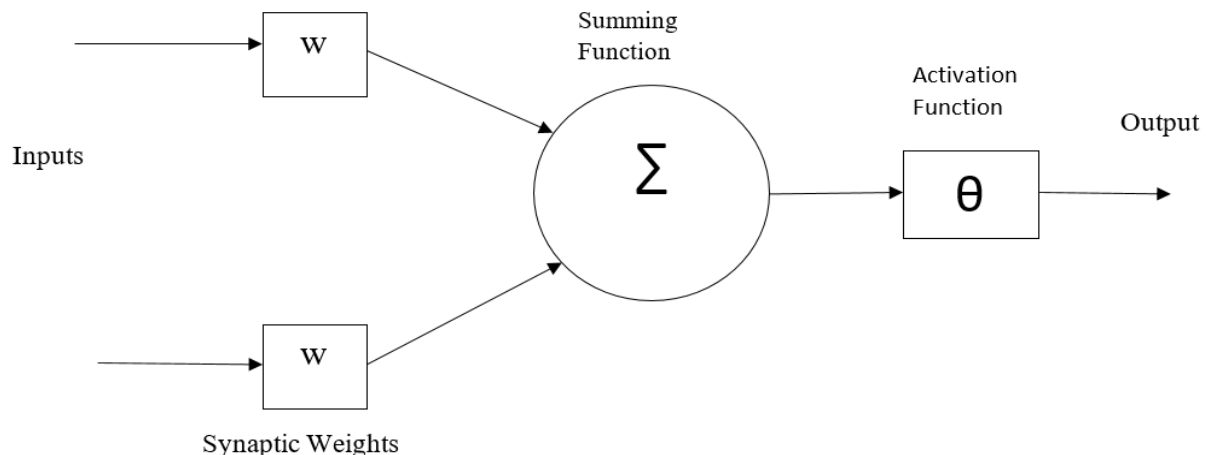


Figure 2.1: Basic Structure of an ANN

A large collection of a simple processing units, neurons, nodes or cells operating in parallel to each other is referred to as a neural network [10]. Every neuron has an activation, which is basically a function of the received inputs. Connection between neurons is through directed communication paths, each with associated weights which denote data that is being utilized to solve the problem by the neural network. The block diagram of the operation of a NN is demonstrated in Figure 2.2.

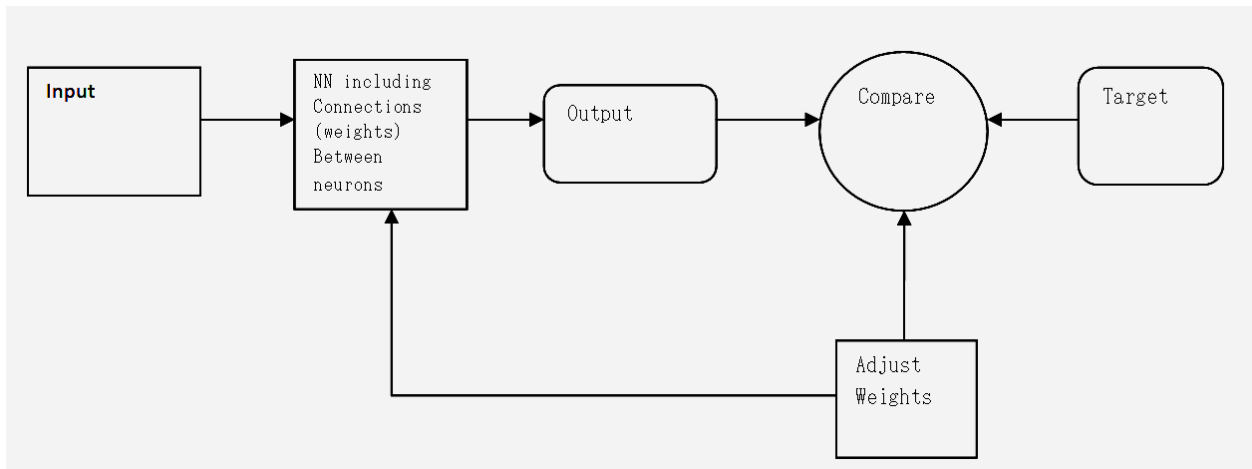


Figure 2.2: Block Diagram of a NN

An ANN is a system specifically designed for applications which, through a learning process, can conduct data classification and pattern recognition.

A variety of approaches are used by artificial neural networks to solve problems as compared to conventional computers which utilize algorithmic approaches [11]. The conventional computer has to be provided with the specific sequential steps to follow which consequently limits the capability to solving problems that are already understood and have been defined by human beings on how they are to be solved [12].

The Levenberg-Marquardt Algorithm is one of the many adaptations of learning algorithms in use and is especially used as a first choice in developing forecasting solutions. It is based on the backward propagation model which is proven to be one of the fastest converging networks for training feed-forward neural networks that are moderately sized [1].

Artificial neural networks cannot be programmed to perform specific tasks but learn through examples. The selection of these task examples must be carefully done in order to prevent the wastage of useful time or resulting in an incorrectly performing network. As the artificial neural network learns how to problem solve by itself, the way it operates may have a level of unpredictability.

The “firing rule” is a major conception used in NN and contributes to their heightened adaptability. The firing rule gives a decision if a neuron should be triggered for any input pattern. An association to all the input patterns is established, in addition to the input patterns that were used to train the node.

The Hamming distance technique can be used to form the simplest firing rule [10, 13]. The rule is described below:

Use a group of training patterns for a node, which include a zero-taught set, which prevent it from firing, and a one-taught set which cause it to fire. By extension, the set of training patterns that are not contained in the group will cause the node not to be fired if, when compared, they contain additional input elements alike to the ‘closest’ pattern in the one-taught set as compared to the ‘closest’ pattern in the zero-taught set [10, 13]. The pattern will remain in an undefined state when a tie is encountered.

A case in point is when a three-input neuron is trained to give an output of 0 when the input provided is 001 or 000 and to give an output of 1 when the input provided (X1, X2 and X3) is 101 or 111. Therefore, prior to the firing rule being applied, the truth table is as seen in Table 2.1

Table 2.1: Pre- fired Hamming Distance Truth Table

X1	0	0	0	0	1	1	1	1
X2	0	0	1	1	0	0	1	1
X3	0	1	0	1	0	1	0	1
OUTPUT	0	0	0 or 1	0 or 1	0 or 1	1	0 or 1	1

An instance of the application of the firing rule can be shown by taking the pattern 010 which varies by one element from 000, two elements from 001, three elements from 101, and two elements from 111. From this, the ‘closest’ pattern is 000 which resides in the zero-taught set. Therefore, when the provided input is 010, the neuron should not fire according to the firing rule. In contrast, 011 has an equal distance to two taught patterns which have dissimilar outputs which has the result of the output remaining undefined (0 or 1).

Table 2.2 shows the obtained truth table on application of the firing rule in each column,

Table 2.2: Post-fired Hamming Distance Truth Table

X1	0	0	0	0	1	1	1	1
X2	0	0	1	1	0	0	1	1
X3	0	1	0	1	0	1	0	1
OUTPUT	0	0	0	0 or 1	0 or 1	1	1	1

The generalization of the neuron is what is defined as the variation between the two truth tables [10, 9]. This means that the neuron is given a recognition of similarity by the firing rule and is enabled to ‘sensibly’ respond to patterns that it was not exposed to during training.

A NN is deemed to have good generalization when outputs are fairly precise for inputs that have not been incorporated in the training group. Overfitting is a common issue plaguing network models. This is also referred to as memorization and has the meaning that the network model has learnt input to output patterns of the training set and in addition, there is storage of undeliberate relations in the synaptic weights. This has the undesired effect of the network providing accurate outputs from the input patterns of the training set but unexpected responses for a slight variations of input patterns [9].

Three factors influence generalization. These are the material difficulty of the problem to be solved, the network architecture (model structure), efficiency and size of the training set [9]. The first factor is uncontrollable, therefore, avoidance of overfitting, is limited to the last two factors.

Overfitting is less likely to occur when training datasets are larger. Nonetheless, there should be a restriction of only inclusion of input to output patterns, that accurately reflect actual processes being modeled. Incorrect, irrelevant and invalid data should therefore be excluded from datasets.

2.4.1 Architecture of Neural Networks

2.4.1.1 Feed-forward

It describes the data movement to the output from the input. The computation on the data can stretch over numerous levels of neurons, without the presence of any feedback connections [9, 12]. Feedbacks are connections outstretching from neuron outputs to neuron inputs in the corresponding layer or preceding layers [13]. Feed-forward ANNs gravitate towards being elementary and simple networks that relate outputs with inputs. Feed-forward ANNs are widely used in pattern recognition [9]. A pictorial depiction of a feed forward neural network is captured in Figure 2.3.

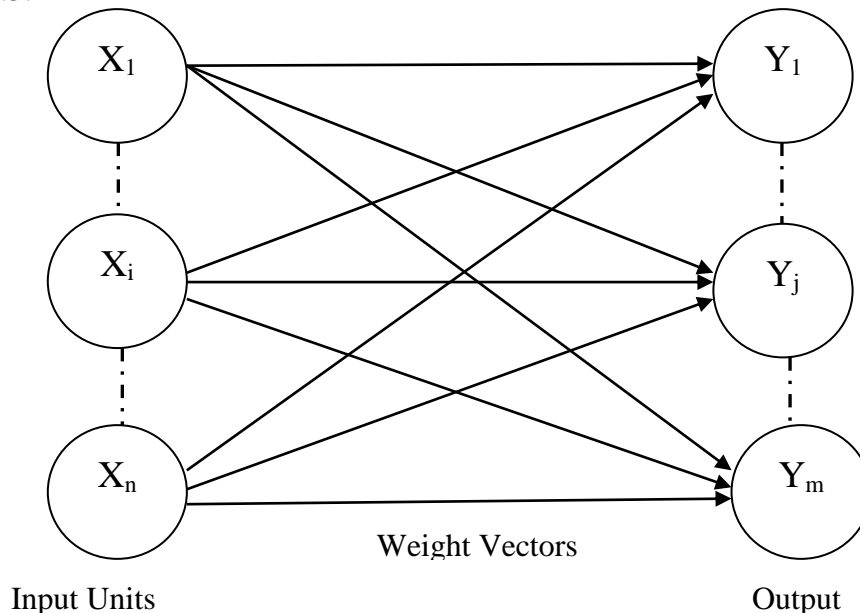


Figure 2.3 A Feed-Forward Neural Network

2.4.1.2 Feedback (Recurrent)

Signals are able to travel bilaterally due to introduction of loops within the network. Feedback networks can get very complicated and are extremely capable. They have an element of dynamism; where their 'state' is unceasingly changing until a balanced point is obtained and thereafter endured at the balanced point until there are changes to the input and a fresh balanced

point is to be obtained [9]. Feedback networks are also defined as interactive or recurrent, even though the latter term is frequently used to designate feedback relationships in single-layer setups [12].

A pictorial representation of a recurrent neural network is as shown in Figure 2.4

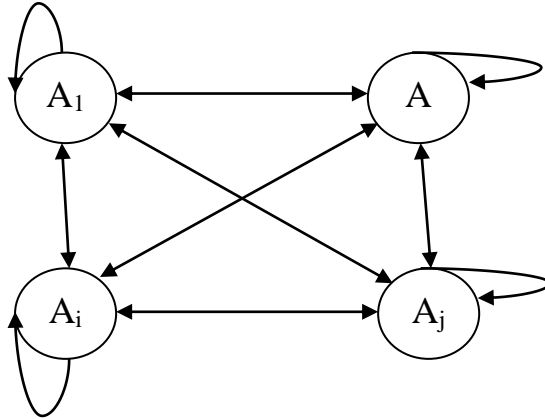


Figure 2.4 A Recurrent Neural Network

2.4.2 The Learning Process (Training)

Neural networks undergo a process of training which allows them to recognize non-linear relationships and patterns between inputs and targets by locating the most suitable weights of the connections between neurons.

2.4.2.1 Recognition Methods

Network responses and the associated pattern memorization can be categorized in two major ways; these are Associative mapping and Regulatory detection.

a) Associative mapping

The ANN comprehends how to make specific patterns on a set of input neurons when a separate specific pattern is used on the group of input neurons [14]. Associative mapping is further grouped in two general methods:

- 1) *Hetero-association*: which is related to two recall methods:
 - i. *Nearest-neighbor* recall, in which the input pattern stored, which is most similar to the presented pattern, corresponds to the created output pattern.

- ii. *Interpolative* recall, where the patterns presented correspond to the patterns stored which in turn have a similarity dependent interpolation to the output pattern.
- 2) *Auto-association*: input patterns are associated with themselves and input and output unit's states coincide which is used to provide completion of patterns. A pattern is produced when a distorted pattern or a portion of the pattern is presented.

b) Regularity detection

In regularity detection each unit's response has a unique 'meaning'. Specific characteristics of input patterns elicit responses in the neurons. The importance of this version of learning is knowledge representation and feature discovery [14].

Each ANN has connection weights whose values possess knowledge. The learning rule for modifying the weight values is used to modify information kept in the ANN as a function of experience. Data is kept in the neural network's weight matrix (\mathbf{w}). The learning of an ANN is done through the determination of the weights [14].

ANNs can therefore be grouped in two main ways depending on how the learning is performed. These are:

- 1) **Adaptive networks** that have the capability of altering their weights, that is $\frac{dw}{dt} \neq 0$
- 2) **Fixed networks** where the weights are fixed cannot be changed, that is $\frac{dw}{dt} = 0$.

These networks have their weights being specified as per the problem that is being solved.

2.4.2.2 Learning Methods

Artificial neural networks generally follow two methods of learning. These are Unsupervised and Supervised.

- i. **Unsupervised learning**: In this method, the artificial neural network does not utilize an extrinsic teacher and is only based on determinate localized environment state

information. Data is provided to the ANN and incipient collective properties are detected through self-organization. When an artificial neural network learns and operates concurrently, it is said to be learning on-line. [4, 11, 12, 14].

- ii. **Supervised learning:** In this method, the artificial neural network uses an extrinsic teacher. Every output neuron is instructed on what the desired reaction to inputs should be. Global environment state information could be needed, while training processes are ongoing. The main examples of supervised learning include *reinforcement learning* which is on punishing unwanted behaviors and/or rewarding desired behaviors, *stochastic learning* which is based on introduction of random variables and use of random based learning or optimization techniques and *error-correction learning* which involves comparison of system output to desired outputs. A major issue of concern in supervised learning is error convergence (which is the error minimization between computed and desired unit values). The goal of error convergence is the determination of a collection of weights that will minimize the error [4, 11, 12, 14]. A common method in many supervised learning examples is the least mean square (LMS) convergence.

Principally, unsupervised learning is conducted on-line whereas supervised learning is done off-line.

2.4.2.3 Transfer Functions

Transfer functions translate inputs to output signals. This transformation occurs between the output and input nodes of the neuron. ANN behavior has a dependence on both the weights, which have been specified for the neurons, and the transfer functions [4, 11, 14]. Transfer functions work with activation functions which trigger the neuron according to the results of the bias and weighted sum. Transfer functions are generally grouped into four categories:

- i. **Linear (ramp):** where the total weighted output is proportionate to the output activity.
- ii. **Sigmoid:** where the output constantly varies but more non-linearly than the variation of the input. The output's relationship to the input is a function of the weighted sum of inputs and is non-linear in nature.
- iii. **Threshold:** where output is fixed on either of 2 levels, which is dependent on if the total input is less than or greater than a threshold value.

- iv. **Gaussian:** where the output follows a normal distribution curve where there is an equal quantity of measurements below and above the mean.

Sigmoid functions have more equivalence to real neurons as compared to threshold, linear or Gaussian functions. However, all the transfer functions should be considered rough approximations.

2.4.2.4 Back Propagation (BP) Learning Process

The BP algorithm is used for weights setting and therefore in multi-layer perceptrons training [10]. For a neural network to be trained to perform a particular task, each unit's weight should be modified in a method that will reduce the error between the actual output and the desired output [15]. The BP algorithm is a determined method for deploying gradient descent technique in the weight space (parameter space between input and output layer), where the gradient of the sum of squared errors with respect to the weights is estimated by transmitting the gradient of the error function in reverse from the input layer in the network [9]. Gradient descent is an optimization technique used to minimize the cost function such that there is minimal deviation of the outputs. BP algorithm has a requirement that the ANN performs a calculation to determine the error derivative of the weights (E_w). That is to say, it should compute the changes in error as each weight is slightly decreased or increased. The BP algorithm is the commonest way for computing the E_w . The BP algorithm is most simply understood when all the network neurons are linear in nature. It works by first calculating the E_a , which is the speed with which the error switches as the level of activity of a unit is modified, after which it calculates each E_w . For output neurons, the E_a is basically the variation between the desired and the actual output. To calculate the E_a belonging to a hidden layer neuron, all weights between output neurons and connected hidden layer neurons must first be recognized and then multiplied by the E_a 's of these output neurons and sum the products. The addition is equivalent to the E_a for the selected hidden layer neuron. Post computing all the E_a 's within the hidden layer just prior to the output layer, the E_a 's for the additional layers can be computed in a similar manner, going from one layer to another in an opposite orientation to the direction actions move in the network. Hence the name back propagation. When the E_a has been calculated for a neuron, it is simple to calculate the E_w for

every arriving link of the neuron. The E_w is a result of the activity through the arriving link and E_a .

The BP algorithm includes an additional step for units that are non-linear, whereby prior to propagating backwards, the E_a is transformed into the E_i , which is the velocity at which the error switches as the total input experienced by a unit is switched.

2.4.3.1 Derivation

The BP algorithm necessarily begins with calculating the output layer, as this is where the wanted outputs are present. However, intermediate layers do not have the presence of outputs [10].

After randomly choosing the network weights, the necessary corrections are computed. The general steps in the BP algorithm can be broken down into the following:

Step 1: Feed - forward computation

Step 2: Back propagation to the output layer

Step 3: Back propagation to the hidden layer

Step 4: Updates of weights

When the error function achieves a suitably minimal value, the algorithm stops to operate. Neurons are joined to each other and the weight of the connection is a real number related with every link. The connection weights from unit u_k to unit u_i is denoted as ω_{ki} . It is consequently appropriate to constitute the connectivity pattern in the network using a weight matrix ω whose elements are the weights ω_{ki} .

Connection types are twofold and defined as:

- Inhibitory which inhibits the generation of a signal and
- excitatory which promotes the generation of a signal.

A negative weight typifies an inhibitory connection whereas positive weight typifies an excitatory connection [10]. This connectivity pattern distinguishes the structure of a network as demonstrated in Figure 2.5 [10].

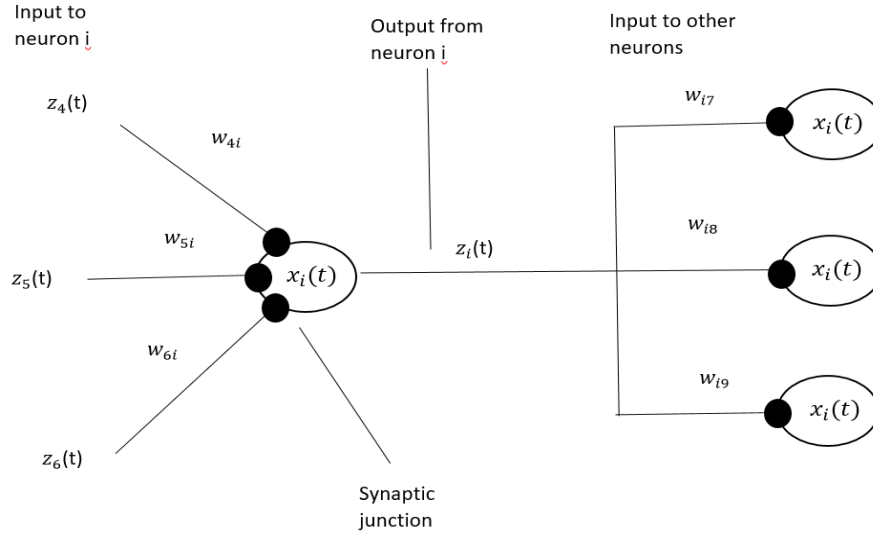


Figure 2.5 Connectivity Pattern of a NN

The error energy found at the output layer is shown as:

$$\varepsilon \triangleq \frac{1}{2} \sum_k (d_k - y_k)^2 = \frac{1}{2} \sum_k e_k^2 \quad (2.1)$$

Where $k = 1 \dots N$; and N is the sum total of neurons in the output layer, and d_k is the measure of activity of the k th unit in the upper layer whereas y_k is the wanted output of the k th unit. Consequently, a gradient of \mathcal{E} is considered, where:

$$\nabla \varepsilon_k = \frac{\partial \varepsilon}{\partial \omega_{kj}} \quad (2.2)$$

Using the gradient (steepest descent) procedure, then: -

$$\omega_{kj}(m+1) = \omega_{kj}(m) + \Delta \omega_{kj}(m) \quad (2.3)$$

The connection weights from unit u_k to unit u_i is denoted as ω_{ki} . At the k^{th} neuron in the output layer, j denotes the j^{th} input and η is the learning rate, where by the gradient procedure:

$$\Delta \omega_{kj} = -\eta \frac{\partial \varepsilon}{\partial \omega_{kj}} \quad (2.4)$$

The negative (-) symbol in Eq. (2.4) shows a downhill direction approaching a minimum.

A perceptron is defined as a mathematical model of a neuron where the k 's perceptron's node output z_k is shown by

$$z_k = \sum_j \omega_{kj} x_j \quad (2.5)$$

x_j which is the neuron's j^{th} input, while it can be noted that y_k is the perceptron's output:

$$y_k = F_N(z_k) \quad (2.6)$$

F_N being a non-linear activation function at the N th neuron in the output layer. It can now be substituted that

$$\frac{\partial \varepsilon}{\partial \omega_{kj}} = \frac{\partial \varepsilon}{\partial z_k} \frac{\partial z_k}{\partial \omega_{kj}} \quad (2.7)$$

further, shown through Eq. (2.5):

$$\frac{\partial z_k}{\partial \omega_{kj}} = x_j(p) = y_j(p-1) \quad (2.8)$$

where p is designating the output layer and y_j is the perceptrons's output due to x_j , so Eq. (2.7) transforms to:

$$\frac{\partial \varepsilon}{\partial \omega_{kj}} = \frac{\partial \varepsilon}{\partial z_k} x_j(p) = \frac{\partial \varepsilon}{\partial z_r} y_j(p-1) \quad (2.9)$$

The ϕ function transforms the weighted sum of inputs in accordance with Eq. 2.10

$$\phi_k(p) = -\frac{\partial \varepsilon}{\partial z_k(p)} \quad (2.10)$$

then Eq. (2.9) becomes:

$$\frac{\partial \varepsilon}{\partial \omega_{kj}} = -\phi_k(p) x_j(p) = -\phi_k(p) y_j(p-1) \quad (2.11)$$

and, by Eqs. (2.4) and (2.11):

$$\Delta\omega_{kj} = \eta\phi_k(p)x_j(p) = \eta\phi_k(p)y_j(p-1) \quad (2.12)$$

Neuron k of the output (p) layer has its j^{th} input denoted as j .

Further, using Eq. (2.10):

$$\phi_k = -\frac{\partial\varepsilon}{\partial z_k} = -\frac{\partial\varepsilon}{\partial y_k} \frac{\partial y_k}{\partial z_k} \quad (2.13)$$

Though, using Eq. (2.1):

$$\frac{\partial\varepsilon}{\partial y_k} = -(d_k - y_k) = y_k - d_k \quad (2.14)$$

where, for a sigmoid non-linearity:

$$y_k = F_N(z_k) = \frac{1}{1 + \exp(-z_k)} \quad (2.15)$$

it can be seen that:

$$\frac{\partial y_k}{\partial z_k} = y_k(1 - y_k) \quad (2.16)$$

In consequence; using Eqs. (2.13), (2.14) and (2.16):

$$\phi_k = y_k(1 - y_k)(d_k - y_k) \quad (2.17)$$

such that, in the output layer, using Eqs. (2.4), (2.7):

$$\Delta\omega_{kj} = -\eta \frac{\partial\varepsilon}{\partial\omega_{kj}} = -\eta \frac{\partial\varepsilon}{\partial z_k} \frac{\partial z_k}{\partial\omega_{kj}} \quad (2.18)$$

so that by Eqs. (2.12)

$$\Delta\omega_{kj} = \eta\phi_k(p)y_j(p-1) \quad (2.19)$$

ϕ_k as denoted by Eq. (2.17), to finish the deduction of the placing of output layer weights.

Propagating backwards to the r^{th} hidden layer, it can be seen that

$$\Delta\omega_{ji} = -\eta \frac{\partial \varepsilon}{\partial \omega_{ji}} \quad (2.20)$$

for the i^{th} branch into the j^{th} neuron of the r^{th} hidden layer. Subsequently, in equivalence to Eq. (2.7):

$$\Delta\omega_{ji} = -\eta \frac{\partial \varepsilon}{\partial z_j} \frac{\partial z_j}{\partial \omega_{ji}} \quad (2.21)$$

while considering Eq. (2.8) and the designation of ϕ in Eq. (2.13):

$$\Delta\omega_{ji} = -\eta \frac{\partial \varepsilon}{\partial z_j} y_i(r-1) = \eta \phi_j(r) y_i(r-1) \quad (2.22)$$

so that, using the right hand-side relation of Eq. (2.13)

$$\Delta\omega_{ji} = -\eta \left[\frac{\partial \varepsilon}{\partial y_j(r)} \frac{\partial y_j}{\partial z_j} \right] y_i(r-1) \quad (2.23)$$

Where $\frac{\partial \varepsilon}{\partial y_j}$ is not accessible (where, also $\phi_j(r)$ above is inaccessible).

However, ε can only be influenced by neurons that are upstream when back propagation is done the output. Any other information is unavailable at this juncture. Hence:

$$\frac{\partial \varepsilon}{\partial y_j(r)} = \sum_k \frac{\partial \varepsilon}{\partial z_k(r+1)} \left[\frac{\partial z_k(r+1)}{\partial y_j(r)} \right] = \sum_k \frac{\partial \varepsilon}{\partial z_k} \left[\frac{\partial}{\partial y_j(r)} \sum_m \omega_{km}(r+1) y_m(r) \right] \quad (2.24)$$

addition over k is done across the neurons of the subsequent $(r+1)$ layer which join to $y_j(r)$, while addition over m is across every input to every k^{th} neuron of the $(r+1)$ layer.

Therefore, whilst taking note of the designation of ϕ , Eq. (2.24) becomes:

$$\frac{\partial \varepsilon}{\partial y_j(r)} = \sum_k \frac{\partial \varepsilon}{\partial z_k(r+1)} \omega_{kj} = - \sum_k \phi_k(r+1) \omega_{kj}(r+1) \quad (2.25)$$

because only ω_{ki} . $(r+1)$ is linked to $y_j(r)$.

Subsequently, through Eqs. (2.13), (2.16), (2.25):

$$\phi_j(r) = \frac{\partial y_j}{\partial z_j} \sum_k \phi_k(r+1) \omega_{kj}(r+1) = y_j(r) [1 - y_j(r)] \sum_k \phi_k(r+1) \omega_{kj}(r+1) \quad (2.26)$$

and, through Eq. (2.19):

$$\Delta\omega_{ji}(r) = \eta\phi_j(r)y_i(r-1) \quad (2.27)$$

to obtain $\Delta w_{ji}(r)$ as a function the weights of the $(r+1)$ layer and ϕ , while taking note of Eq. (2.26). It is important to note that partial derivatives of ε in regard to the considered hidden layer cannot be taken. Therefore, the partial derivatives of ε with respect to the upstream variables in the output direction must be taken, as they are the only ones that influence ε . This consideration is the background for the Back-Propagation process, to enable curbing the absence of attainable error data in the hidden layers [10, 12].

To finalize its derivation, the BP algorithm reverse propagates all the way to $r=1$ (which is the primary layer).

2.5 SURVEY OF EARLIER AND RELATED WORK

Mobile traffic forecasting techniques can be classified into different categories according to the methods that have been employed in forecasting and analysis. The methods that have been used in load and traffic forecasting techniques include:

1. Quantitative mobile broadband traffic prediction illustration, which uses the Gompertz function as a mathematical basis [3]. This method is centred on mathematical modelling structure for the mobile broadband growth of traffic that is used in evolution studies. It is demonstrated that an “S-curve” model premised on a Gompertz function can be suitably parametrized and gives adequate workability in forecasting different traffic increase scenarios. A network evolution study is done as an example application of the formulated model. The study was done in an urban based European network in conjunction with postulations on the target network KPIs and user equipment (UE) receiver limitations. Many prediction examples are demonstrated with a disclosure of the potential bearing on the needed radio network capacity growth.
2. Driving factors for traffic growth were analysed for mobile data trends in the Western European Market in [16]. Long term prediction patterns for cellular broadband

penetration of various subscription categories are done using extrapolation and diffusion models and used for rollout of new technology, upgrade planning and network dimensioning.

3. Dynamic Harmonic Regression (DHR) and Autoregressive Moving Average [17] have been used to analyze different methods for long term forecasts of packet switched traffic from live 3G networks. The dataset consists of 400 values, each representing the peak load for a separate day. Four methods were applied to forecast the increase in traffic: linear and exponential regression, ARMA and DHR. Sophisticated methods such as ARMA and DHR performed better. Authors showed simulation results for long term, (more than 100 days) as well as short term forecasting (hourly or daily). ARMA and DHR are more sophisticated than linear regression or exponential regression. Authors used METAWIN statistical software Meta-Analysis with Resampling Tests to monitor and record the traffic in a mobile core network. It is noted that the authors remove any user payload to meet the privacy requirements. The authors also used NEXUS reporting suit provided by Nexus telecom to monitor GSM, GPRS and UMTS network components. NEXUS is designed to monitor usage trends and traffic volumes within a multi-vendor core network.
4. In Traffic Predicting for Mobile Networks with Multiplicative Seasonal ARMA Models [18] which is rooted on the study of real data accrued by China Mobile Communications Corporation (CMCC) Heilongjiang Co. Ltd, the authors proposed to use the multiplicative seasonal Autoregressive Integrated Moving Average (ARIMA) models for mobile communication traffic forecasting. Experiments and test results showed that [18] was feasible and effective to fulfil the requirements in traffic forecasting application for mobile networks.
5. Network Traffic Prediction and Result Analysis hinged on Seasonal ARIMA and Correlation Coefficient [6] where focus on traffic prediction for network planning and network optimization was used. Time series from a mobile network in Heilongjiang province in China was studied. Multiplicative seasonal Autoregressive Integrated Moving Average model (ARIMA) is employed to make traffic series prediction.

6. Accumulation Predicting Model (APM) is suitable for time series showing steady seasonal pattern and is easier to employ as compared to ARIMA. A time series of traffic from a certain mobile network of Heilongjiang province in China was studied. Average daily traffic per month for the province as well as its every sub-region was forecasted by using APM. The resulting precision was high and even comparable with ARIMA [6].
7. Multifractal Exploration based traffic prediction has been used in [19]. Box-Jenkins model has been proven to be appropriate for voice traffic (since the arrival calls follows a Poisson distribution). Internet traffic exhibits statistical self-similarity and has to be modelled using the Fractional Autoregressive Integrated Moving Average (FARIMA) process. Origins of self-similarity in the internet traffic have been mainly attributed to the heavy tailed probability distribution of the transfer sizes and inter-arrival times. Unlike other datasets, network traffic in the Internet core was effectively represented by a Poisson model within the sub-second time scales. After employing the heavy tailed – infinite variance “Levy” distribution, the simulation results indicate that the backbone traffic of the internet nearly follows a Poisson distribution and closely resembles a self-similar process at the same time. To forecast, the nature of the time series need to be determined through comparison to short-range dependence and long range dependence and then applying the most suitable forecasting model [19].
8. Least Squares Support Vector Machine (LS-SVM) has been employed to achieve analysis of characteristic and forecasting of the mobile communication traffic. Compared with Seasonal ARIMA models, experiments and test results show that the LS-SVM solution increased the implementation efficiency greatly and improved accuracy [20].
9. Soft Computing Paradigms have been used to perform contrastive investigation of six (6) soft computing modes (multi-layer perceptron networks) Elman recurrent neural network (RNN), radial basis function network, Hopfield model, fuzzy inference system and hybrid fuzzy neural network [21]. Simulation results indicated that hybrid fuzzy neural networks were the best candidates for the analysis and forecasting.
10. Real Time Prediction Using Ad-hoc Networks where a procedure to develop a multilayer feed forward neural network combined with a backpropagation algorithm for forecasting travel time and traffic congestion is described in [22]. The prediction of travel time and

traffic congestion was based on past and current traffic information and was found to be not straightforward due to among others, the high complexity and ill predictability of traffic process, incorrect observations and different data sources. However, it was found that neural networks could exhaustively be used to solve these problems. The Real Time Prediction Using Ad-hoc Network was designed on top of a mesh-based communication infrastructure for the mobile nodes to communicate. The mesh-based communication approach enabled easy deployment of the system in real world. Optimized Link State Routing Protocol (OLSR) routing protocol was used for establishing an ad-hoc network for peer-to-peer communication with a high level of accuracy of forecast achieved.

11. The use of an ensemble of neural networks to improve HSDPA traffic forecasting through machine learning algorithms including Support Vector Regression (SVR) and abductive networks to improve the forecasting accuracy is reported in [23]. The trained model was characterized with a good forecasting performance. A data set containing 44160 recordings of hourly high-speed data packet access (HSPDA) data traffic in UMTS based cellular network.
12. Use of deep spatio-temporal neural networks long-term mobile traffic prediction method is reported in [24]. A double Spatial transformer networks (STN) method was created which singularly combined STN forecasts and historical statistics, resulting in extended term projections of mobile traffic.
13. In [25], Seasonal Auto Regression Integrated Moving Average (SARIMA) has been explored as a complementary method of predicting UMTS based data traffic in the capital city of Ethiopia, Addis Ababa. Past UMTS data traffic load was used to predict future loads as a basis for infrastructure expansion.
14. 3D convolutional models have been used to forecast mobile traffic in [26] where spatiotemporal features from neighbouring stations to the target base station are utilized to forecast mobile traffic with the experimental outcomes showing better forecast results when compared to multivariate Auto Regressive Moving Average model (ARMA), and a model based on the n-to-n with 1-to-1 model for traffic forecasting.
15. A predictor model which employed ANN has been used to forecast mobile network traffic in [27] with attention given to its performance when compared to linear predictors. The

ANN based predictor model was found to outperform linear predictors in shorter observation window lengths which make them more useable in the situation where information on historical network traffic data is unavailable.

16. A study was done comparing time series analysis methods and supervised learning to forecast LTE busy hour traffic in [28] where it was found that supervised learning techniques performed better than time series methods as regards to forecast precision and data storage space requirements.

2.6 THE CASE FOR THE UTILIZATION OF ARTIFICIAL NEURAL NETWORKS IN MOBILE TRAFFIC FORECASTS

ANN can be used in pattern recognition and trend detection to obtain solutions for problems which have a complexity level that is too high for solving or detection by conventional computer methods or by human analysis. This is due to the ability of artificial neural networks to develop meaning from incomplete, imprecise or complex data. A trained neural network can be described as an "expert" in the information category it has been provided to analyze [12].

Additional benefits of artificial neural networks over other techniques are:

- 1) It utilizes simple calculation operations such as multiplication, summing and basic logic operations to resolve stochastic, complex, nonlinear problems or mathematically ill-defined problems [9, 11].
- 2) Self-Organization: Artificial neural networks are capable of creating its own representation or organizational structure of information obtained during its learning process [9, 12].
- 3) Adaptive learning: this is the artificial neural network's ability to adapt its learning depending on initial experience and data fed to it. [9, 12].
- 4) Fault Tolerance via Redundant Information Coding: When a neural network is partially destroyed, computing performance undergoes a level of degradation that corresponds to the level of damage. Artificial neural networks, however, possess a degree of fault tolerance and can retain some network capabilities despite major network damage [12].
- 5) Real Time Operation: Artificial neural networks can carry out computation processes in parallel [18].

Artificial neural networks have found use in various forecasting tools. They have been used to forecast electric loads in the short, medium and long term [10, 21]. Their flexibility in use has been demonstrated in the estimation and real time prediction of road traffic and travel time in intelligent traffic management systems [22]. Their ease of combination with other forecasting techniques has resulted in hybrid forecasting techniques such as the combination of artificial neural networks with Holt-Winters model to forecast TCP/IP based network traffic volumes for ISPs [29].

Artificial Neural networks have been used to forecast mobile GSM voice traffic in Nigeria [7] where future traffic was predicted based on historical traffic per BSC. Traffic from six BSCs was considered. The data used in [7] was obtained for each hour of the day spanning a period of five months.

However, this approach only considered historical voice traffic as training data without considering that future traffic could be influenced by other factors such as network node service area density, special events such as public holidays, and time of day, as well as consideration of these factors in the forecasting of mobile data traffic in addition to mobile voice traffic. Additionally, cellular traffic rides on various technologies (2G, 3G, 4G) which is carried through BSC, RNC and LTE nodes which was considered in the work done in this thesis which also used data spanning over a larger time horizon of a year so as to improve the accuracy of forecasts all year round. Robustness of the ANN to variation in service area density factors has been improved by selecting network nodes in various service areas. Traffic prediction based on various network nodes i.e. 21 BSC, 10 RNC and 1 LTE relay node which all serve different service areas. Therefore, it is possible to drill down and obtain granular traffic and BBH values of these different service areas that each of these nodes serve.

2.7 RESEARCH GAP

The approach used in Nigeria [7] only considered historical voice traffic as training data and further did not delve into the forecasting of cellular data traffic.

Additionally, consideration of QoE is not restricted by the technology in use. The case in [7] only considered 2G voice traffic from BSC nodes. Consumers are transparent to what technology is in play when using mobile services and it is inconsequential whether they are on 2G, 3G 4G, or 5G as long as the experience is seamless. Most research on cellular traffic volumes has been constrained to GSM (2G) as is the case in [7]. However, this approach does not provide the full picture as cellular service users do not lock their mobile terminals to only operate on a certain technology but may oscillate across 2G, 3G, 4G or 5G depending on which technology is optimally available at their location. To get a clearer picture of the QoE requirements, research and design of the ANN module in this work spanned across all technologies in a bid to determine the capacity requirements across all mobile service technologies through forecasts of the maximum voice and mobile data traffic expected at network nodes as well as the expected network node busy hours each day. Additionally, to enhance the precision of the forecasts, a larger data set was considered so as to allow to accurately predict cellular traffic all year round.

Further, as convergence of technology accelerates, there is increased use of packet switched (mobile data) services in the provision of services that were traditionally circuit switched in nature. Such services include voice, text, audio and video messaging as well as video calling. Addition of internet protocol television (IPTV) and over-the-top (OTT) services, such as Voice over Internet Protocol (VoIP), on the cellular platform has put the provision of quality mobile data services into perspective. Therefore, any capacity forecasts of circuit switched (voice traffic) network resources need to be coupled with forecasting of packet switched (data) capacity resource requirements. Previous research material has only focused on voice traffic and data traffic separately with more focus being placed on the traditional voice services delivered over the GSM platform. Finally, additional inputs to the training data on factors such as time, location, special events, network node types and service technologies were incorporated to give a more accurate result.

2.8 SUMMARY

The capability of learning by example makes artificial neural networks extremely versatile and powerful. Additionally, they make it unnecessary to design a task specific or data specific

algorithm for the performance of a specific task. It is therefore unnecessary to analyze or understand how that task will be deciphered. Artificial neural networks have parallel architecture and quick computational and response times that make them suitable for real time and live systems. Although the selection of the best combination of processing time, least error value and optimum neural network architecture in artificial neural networks is iterative there is less computational effort required and is easy to implement when compared to other forecasting methods [30].

The major advantage of the artificial neural network method is that an initial mobile traffic load model is not required. The main drawback of artificial neural networks is there is no assurance of convergence in the training process [21]. A further disadvantage is that there is no explicit law for selecting the size of hidden layer to circumvent under-fitting and over-fitting [21]. This is overcome by creating a number of network configurations and testing them against algorithms and subsequently performing cross validation of performance results.

CHAPTER 3

MATERIALS AND METHODS

The proposed neural network should make traffic forecasts for each hour in the day. It should also be capable of determining the busy hour for the sampled cellular network. To achieve this, relevant input variables were identified, formulated and gathered from the field.

Traffic in dense urban environment generally follow a static and linear trend and will give a more likely worst case setting for network expansion due to network congestion attributed to a high population density.

3.1 DATA SOURCING

The study benchmarked with similar works done in other jurisdictions and adapted to the context in Kenya. The research data was sourced from verified traffic volume data from a Kenyan MNO. Cellular voice traffic in Erlangs and cellular data in Gigabytes logged over a period of over one year on an hourly basis was used in the training, validation and testing of the designed ANNs. The busy hours for voice and data services were obtained by taking note of the hour of the day that maximum traffic is recorded.

3.2 ARTIFICIAL NEURAL NETWORK DESIGN

The notion backing the use of multi-layer artificial neural network models in mobile traffic forecasting is the assumption that future traffic is dependent on historical traffic volumes as well as external influencing factors (e.g. time of day/month, year, ICT penetration, population growth, service tariffs and special events), and the multi-layer artificial neural network is used to estimate this dependency.

Therefore, the building of a multi-layer artificial neural network model for mobile traffic forecasting can be deemed to be a nonlinear system identification task. Determination of the network model structure involves the choosing of network structure and the input variables. The estimation of parameters is achieved by training the ANN on historical mobile traffic data which require a choice on a suitable training data and an appropriate learning algorithm. Validation of

the model/ trained network is done by testing on new traffic data. The neural network model is therefore assumed to be founded on pattern recognition functions.

The learning process involves, the input layer receiving data or information from an external source and passing this data to the hidden layer which processes the data, simple mathematical processing which involves the values of the inputs as well as the associated link weights. The produced results from the hidden layer are imposed onto suitable threshold functions resulting in final outputs from that layer. These output values subsequently act as inputs to the elements in the next layer which could be the output layer or a second hidden layer. The computation processes recur in the layers till final output values are generated by the output layer. During this stage, the value of the output error is obtained by calculating the variance between the actual outputs and that of the multi-layer ANN. This training process undergoes several epochs and is repetitive, and terminates when the margin of error is small enough to be acceptable. When the learning process is finished, the artificial neural network should be capable of giving an output answer(s) to a provided input dataset that is founded on the developed generalized mapping [30].

The behavior of traffic characteristics undergoes rapid changes which make artificial neural networks suitable for modelling forecasting future traffic loads. This is an issue with statistical models, as they aren't always able to keep up with the abrupt changes in the dependencies of the traffic. Artificial neural networks can recognize changes in conditions without having to re-estimate parameters. Conditions matching to new states need not have been already utilized in training of the network and subsequent inputs must be in the same format and contain information that will enable the network to recognize the condition.

The forecasting problem in this research is of a times series nature, therefore, a supervised learning approach was utilized through a multi-layer feed-forward artificial neural network which made adjustments to the network weights on its internal nodes and inputs in an iterative fashion. The goal of iterative adjustments is in minimizing errors between actual target and predicted values through error back propagation, which is a stochastic gradient descent, over several

epochs. The final product is the achievement of optimized network weights and values with a local minimum of error response.

The proposed model was trained using a back-propagation algorithm using two layers namely an input layer (not counted as no processing is done here), a hidden layer and an output layer and trained using a supervised learning algorithm.

A network that is well trained using the back-propagation algorithm is inclined to give fairly accurate responses and predictions to new data provided as inputs [12]. The generalization property of trained artificial neural networks enables the training of the network on some but not all outcomes or possible solutions and still get good results with a fairly high degree of accuracy.

There were four steps that were implemented in the training process:

- i. Assembly of training data.
- ii. Creation of the neural network object.
- iii. Training of the neural network.
- iv. Testing and validation of the neural network response.

The traffic forecasting framework was based on 2G, 3G and 4G radio access technologies.

As the network equipment used may be different, a traffic forecasting module was created for each network node i.e. BSC, RNC and LTE. Therefore, the tool was able to forecast for both mobile voice and data traffic:

1. The hourly volumes for BSC voice traffic and RNC voice traffic
2. The hourly volumes for RNC data traffic and LTE data traffic
3. The Bouncing Busy Hour for BSC voice traffic and RNC voice traffic
4. The Bouncing Busy Hour for RNC data traffic and LTE data traffic

The implementation of this traffic forecasting module was done using the commercially available MATLAB Student R2018a software with Neural Network toolbox. The software has three inbuilt learning algorithms, which are: The Levenberg-Marquadt, Scaled Conjugate Gradient, and the Bayesian Regularization algorithm which are efficiently implemented in MATLAB as the

solution of the Hessian Matrix which involve second order derivatives that are used in machine learning [31].

The hardware system used was Intel core i7-8550U CPU 1.99GHz processor with 8.00 GB RAM, 1 TB hard drive and a 64-bit OS, x64 based processor system running Windows 10 Pro operating system.

Training data was identified from factors that influence mobile traffic patterns and captured in a workbook format which was imported into the neural network toolbox.

3.4 PERFORMANCE METRICS

A combination of the Levenberg-Marquadt, Bayesian Regularization, and the Scaled Conjugate Gradient learning algorithms and network architectures of ten, twenty, thirty, forty, and fifty neurons in the hidden layer were tested against the data and a comparison of the error margins tabulated for each outcome in the selection of the optimal design for a mobile traffic forecasting problem. Mean Squared Error (MSE) results was used to determine accuracy of forecasting capability of designed networks.

The MSE is the variance between the ANN's predictions and the ground truth, which is squared (to remove negative signs) and averaged out across the whole dataset. It is utilized in the determination of the closeness of predicted values to actual values where a lower value indicates a closer relationship. The range of MSE ranges from 0 to infinity.

The formula for MSE is given in Equation 3.1 as:

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 \quad (3.1)$$

Where: MSE = mean squared error
 Y_i = observed values

\hat{Y}_t = predicted values

n = number of data points

3.5 PERFORMANCE OF THE ARTIFICIAL NEURAL NETWORK

Training, testing and validation of the ANN was done with cellular data obtained at the BSC, RNC and LTE levels. This gave a bearing of the network expansion requirements looking inwards towards the core and outwards towards the radio access.

Generally, there are no standard limits for average MSE, however, a lower value of MSE indicates higher degree of accuracy of a predicted value to the actual value. An arbitrary value of average MSE of less than one (1) was targeted in this research especially considering that the ANN was trained on associated values of factors that influence mobile traffic fluctuations and consideration of temporal (time based) factors cascaded to hourly quantities spread over a twenty-four-hour horizon.

The ANN was expected to determine the bouncing busy hour for network voice and data traffic across all technologies i.e. 2G, 3G and 4G/LTE, as well as the hourly traffic. The hourly traffic could be totalled to obtain a general trend in traffic growth over a long period. This would aid in the design of the capacity requirements and the distribution of network resources and adoption of short term mechanisms for handling traffic overloading such as utilization of half rate coding and offloading of data traffic to Wi-Fi hotspots.

CHAPTER 4

DATA ORGANIZATION AND CHARACTERISTICS

4.1 NETWORK INPUT

The accuracy of a time series forecast is influenced to a great extent by the amount of historical information available to form sensible patterns. A dataset of at least one calendar year provided conditions that would capture the cellular network conditions that could be analysed by the ANN for each month, day and hour of the year.

The data set used in this research was for a period of one year and two months taken hourly for BSC, RNC and LTE network nodes spanning from September 2019 to November 2020. The data analysed was composed of all the network traffic data obtained at the major cellular network nodes.

To develop a suitable sampling matrix, the following data fields were selected as inputs to develop a unique training profile for the targets which are expected to form the network outputs.

- i. Month
- ii. Day of the Week
- iii. Special days such as public holidays
- iv. Hour of the Day
- v. BSC/ RNC/ LTE node identifiers

In the input variable matrix, the rows were configured as follows:

- Column 1 to 4: Month number in binary form i.e. January = 0001, December = 1100.
- Column 5 to 7: Day of the week i.e. Monday = 001, Sunday = 111.
- Column 8: Table 4.1 gives a breakdown of the special days which were represented by 1 and 0 otherwise. Public holidays that fall on Saturday were ignored and marked as 0. Those that fell on Sunday were marked 0 but the subsequent Monday was marked as 1 as the Monday is counted as a holiday and therefore affects calling patterns and population distribution especially in urban settings.

Table 4.1: Special Days

	Day	Date	Special Day Name
1	Thursday	10/10/2019	Huduma day
2	Sunday	20/10/2019	Mashujaa Day
3	Sunday	27/10/2019	Diwali
4	Thursday	12/12/2019	Jamhuri Day
5	Wednesday	25/12/2019	Christmas Day
6	Thursday	26/12/2019	Boxing Day
7	Wednesday	01/01/2019	New Year
8	Tuesday	11/02/2020	Funeral of Former President Moi
9	Friday	10/04/2019	Good Friday
10	Monday	13/04/2020	Easter Monday
11	Friday	01/05/2020	Labour Day
12	Monday	25/05/2020	Idd-ul-Fitr
13	Monday	01/06/2020	Madaraka Day
14	Friday	31/07/2020	Idd-ul-Azha
15	Saturday	10/10/2020	Huduma day
16	Tuesday	20/10/2020	Mashujaa Day
17	Saturday	14/11/2020	Diwali

- Column 9 to 13: Hour of the day i.e.00000 for hour 0 to 10111 for hour 23
- Column 14 to 17/18: Table 4.2 gives the breakdown of the BSC identifiers in binary code while Table 4.3 gives RNC identifiers in binary code.

Table 4.2: Base Station Controllers

	BSC_Name	BSC Binary code				
1	BGMBSC2	0	0	0	0	1
2	ELDBSC2	0	0	0	1	0
3	FTRBSC1	0	0	0	1	1
4	IKUBSC2	0	0	1	0	0
5	KIAMBA_BSC	0	0	1	0	1
5	KISBSC3	0	0	1	1	0
7	KRTBSC2	0	0	1	1	1
8	KSMBSC2	0	1	0	0	0
9	KSMBSC3	0	1	0	0	1
10	KTEBSC1	0	1	0	1	0
11	KTIBSC1	0	1	0	1	1
12	KTLBSC2	0	1	1	0	0
13	MDIBSC2	0	1	1	0	1
14	MSABSC4	0	1	1	1	0
15	NKUBSC3	0	1	1	1	1
16	NKUBSC4	1	0	0	0	0
17	PKSBSC5	1	0	0	0	1
18	PKSBSC6	1	0	0	1	0
19	RNGBSC2	1	0	0	1	1
21	VPKBSC2	1	0	1	0	1
20	RUKBSC2	1	0	1	0	0
21	VPKBSC2	1	0	1	0	1

Table 4.3: Radio Network Controllers

	RNC_Name	RNC Binary code			
1	ELDRNC1	0	0	0	1
2	NKURNC2	0	1	1	0
3	KSMRNC2	0	1	0	0
4	MSARNC2	0	1	0	1
5	PKSRNC2	1	0	0	0
5	VPKRNC2	1	0	1	0
7	KSMRNC2	0	1	0	0
8	VPKRNC2	1	0	1	0
9	PKSRNC1	0	1	1	1
10	PKSRNC2	1	0	0	0

In total, four input matrices were created with ancillary target matrices. These were BSC Voice, RNC Voice, RNC Data and LTE data.

These input matrices were used as inputs to the ANN to map to both hourly traffic volumes as well as network busy hours with prediction targets being:

1. The Bouncing Busy Hour
2. The hourly traffic

The weekly busy day and the maximum hourly traffic can then be obtained by the analysis and summations of the ANN hourly prediction values.

4.2 CLUSTERING OF INPUT DATA

To analyze the degree of similarity between the input variables, a low dimensional visualization of the ANN training samples and the relative associative weights was done through a process of clustering the input data. Data is grouped by similarity by using a Self Organizing Feature Map (S.O.F.M) or a Self Organizing Map (S.O.M.) composed of a two dimensional finite region composed of regularly arranged rectangular or hexagonal grid.

The ANN pattern recognition problem can be visualized as low dimensional representations through a process of competitive learning or vector quantization, while preserving the top level properties of the input space such as similarity relations amongst data elements [32]. The trained ANN outputs a feature map which does vector classification through identification of the node that has the minimal distance metric weight vector to the input space vector. A configuration of 100 neurons arranged in a 10 by 10 matrix was used to represent the input data.

A hit plot was plotted for the BSC voice traffic input data as shown in Figure 4.1. Data with attributes of a similar nature are grouped together and their closeness in similarity results in closer distances than attributes that are less similar resulting in a two dimensional visualization of multi-dimensional data points.

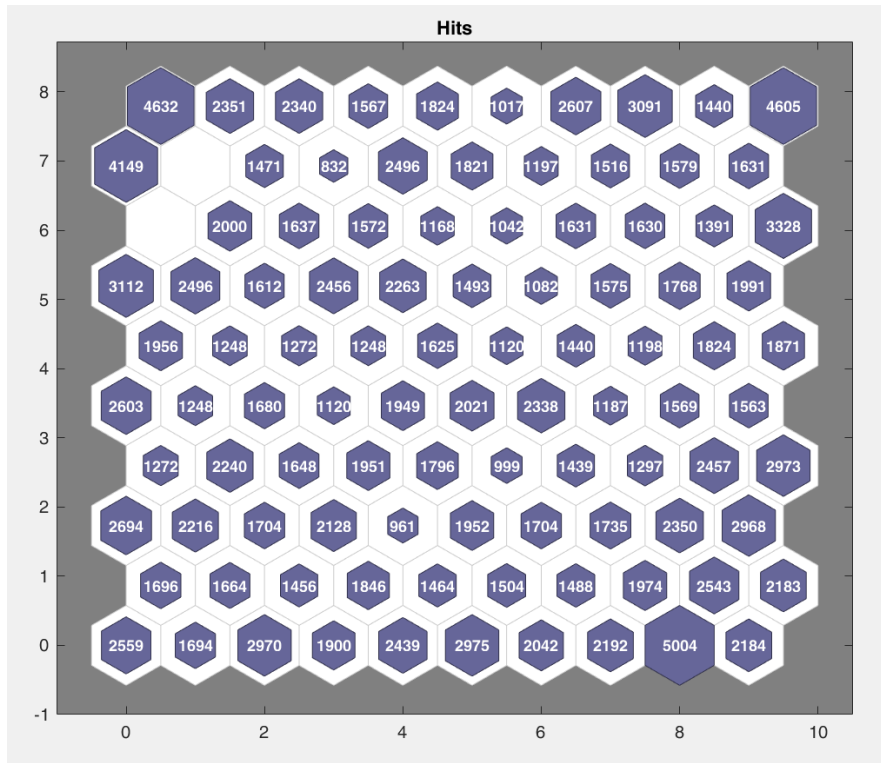


Figure 4.1: Sample Hits Plot for Input Variables for BSC Voice Traffic

Figure 4.1, demonstrates the number of BSC voice traffic training data which are linked with each of the cluster centers (neurons). The figures on the axes are a two dimensional scale of the number of neurons on the lattice which in this case is a 10 by 10 matrix configuration of a 100 neurons. The input data is composed of the columns 1 to 18 of the BSC voice traffic training data. The highest number of hits linked with any neuron is 5004. Therefore, 5004 input vectors are in that cluster which is the number of data points associated with that neuron. Zero hits indicate that no input vector in that cluster is associated with that neuron. This is true of the other neurons and their associated values.

Each element of the input vector has a weight plane as demonstrated in Figure 4.2. This is shown for columns 1 – 18 of the BSC voice traffic training input matrix. Each plot is a visualization of the weights that connect each of the neurons to each input variable where darker colors represent bigger weights. Similar connection patterns of any inputs provide an assumption that the inputs are highly associated. From a visual analysis and comparison, all the inputs have connections that are dissimilar. None of the input variables have similar weight patterns.

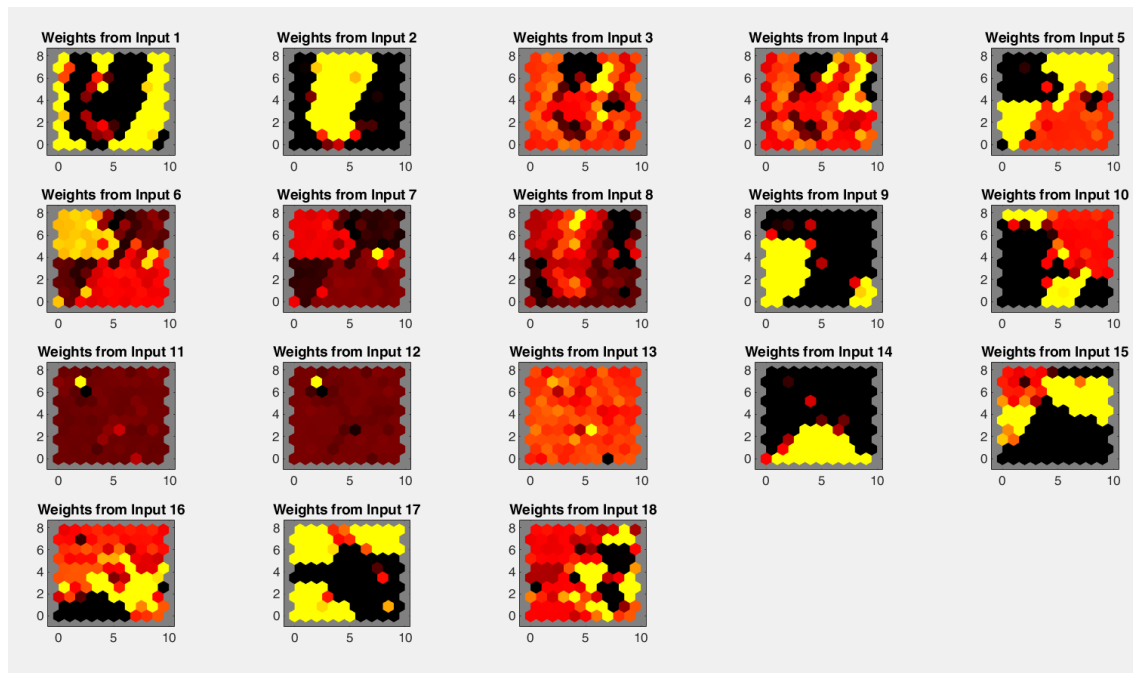


Figure 4.2: BSC Voice Traffic Input Variables Weight Planes

Elements that have similar weight patterns can be eliminated from the input matrix to speed up the training process if necessary. Through visual inspection, it was seen that elimination of any of the eighteen input elements will compromise the accuracy of the neural network as none of the elements has a degree of similarity in the patterns formed from the 2 dimensional visualizations of the input variables weight planes. The largest number of hits related with a single neuron was 5,004 out of a total of 191, 754 input variables. Similar tests were done for RNC Voice, RNC Data and LTE input data and visual comparisons of the input weight planes of each of the columns in the input matrix did not show any similarities in patterns. The SOM plots for BSC and RNC voice and RNC and LTE data input variables are as shown in Figure A 2.1.1 to A 2.4.2 in the appendix.

4.3 NETWORK TARGETS

4.3.1 Daily Traffic Trends

For the study period, the daily traffic load profiles indicate a constant volume for 2G voice traffic as evidenced by the BSC traffic profile in Figure 4.3, however, a gradual increase for 3G voice

and data traffic can be seen in Figure 4.4. Additionally, LTE data traffic also increased during the period. This could be attributed to the factors such as uptake of smart mobile devices that utilized the new packet switch technologies such as 3G and LTE. Mobile network operators have focused on rolling out newer technologies that are cheaper and that have higher traffic capacity [33]. This has resulted in a migration from legacy systems and an increase in deployments of technologies such as 3G and LTE.

Significant peaks and drops in network traffic observed in Figures 4.3, 4.4 and 4.6 which can be attributed to failures in network elements as well as temporary migration of network traffic to accommodate network elements that are undergoing maintenance or replacement.

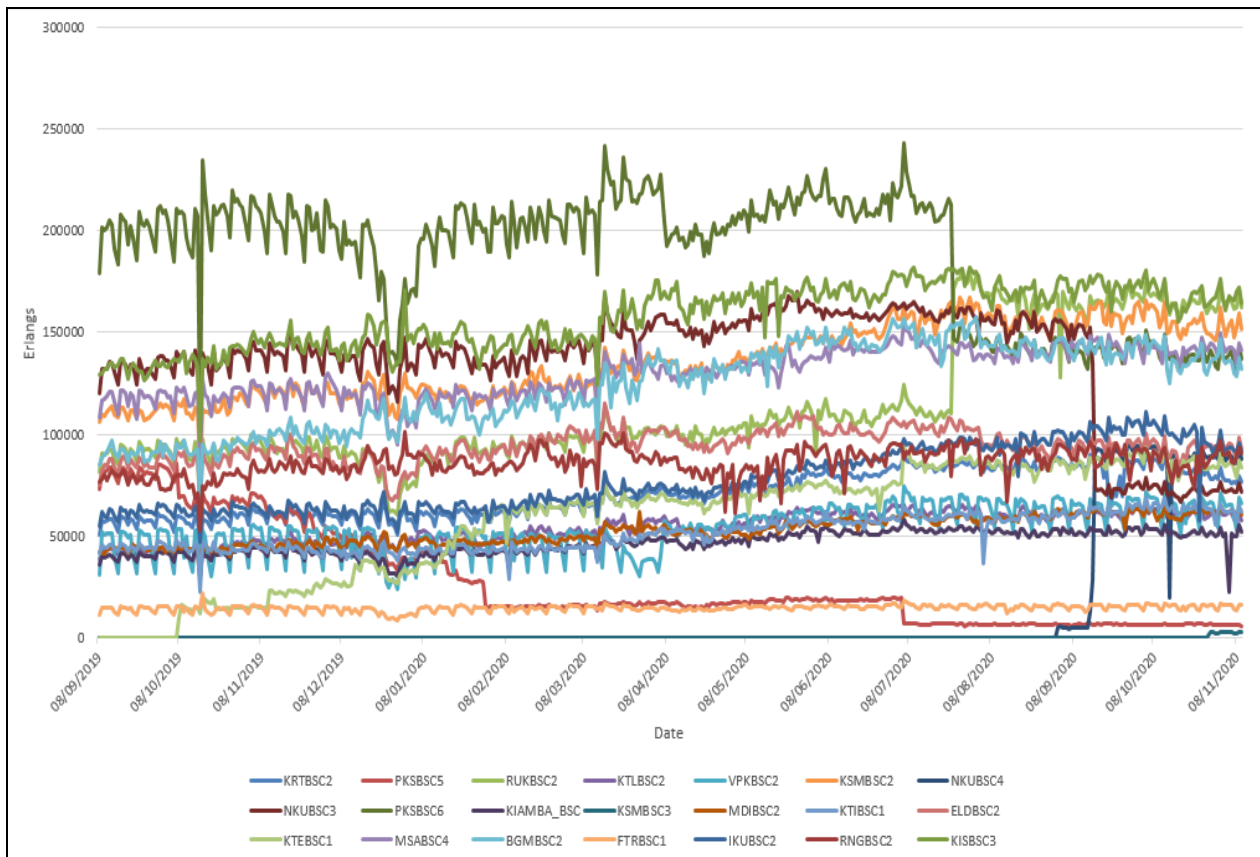


Figure 4.3: 2G (BSC) Daily Voice Traffic

With time, it is expected that 2G traffic will reduce as the traditional 2G access frequencies will be re-farmed to support 3G and later technologies as the mobile networks mature and as the market adopts an increased uptake of smarter data driven end user devices.

Figure 4.3 and 4.4 show that 3G traffic is constantly increasing during the study period while 2G traffic is constant and reducing in some of the BSCs analyzed.

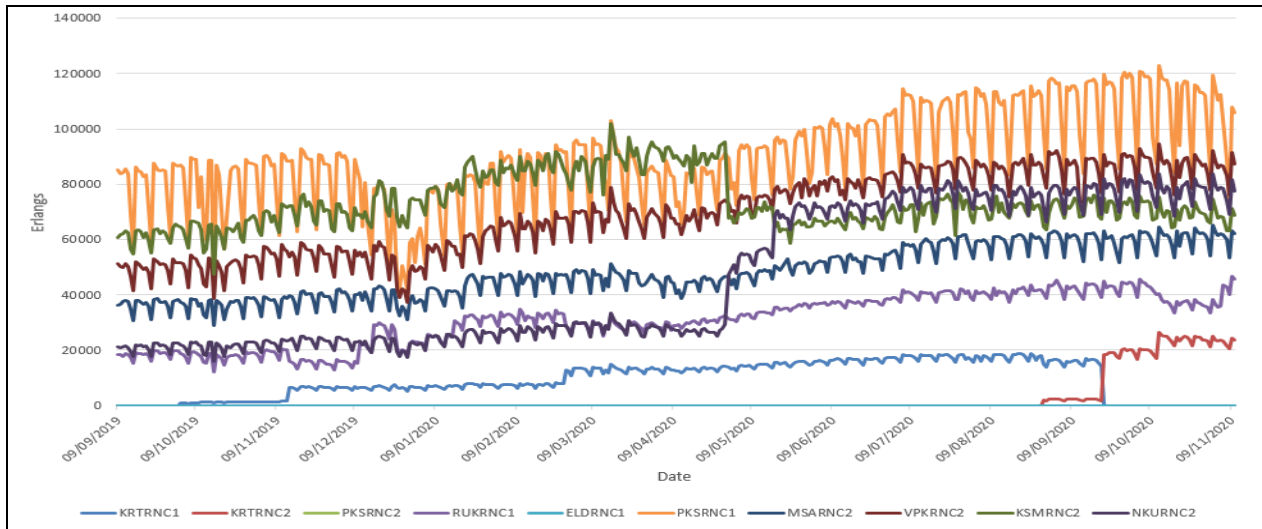


Figure 4.4: 3G (RNC) Daily Voice Traffic

It can be noted from Figure 4.5, which shows total aggregated network 2G and 3G voice traffic per day, 2G voice traffic patterns maintain a fairly constant and slower percentage growth where traffic increased from 1,338,229 to 1,803,268 Erlangs (35% increase) as compared with 3G volumes which had a significant percentage increase in total volumes from 213,212 to 470,442 Erlangs (120.6% increase). This is a comparison between traffic volumes on the first date and the last date of the data set.

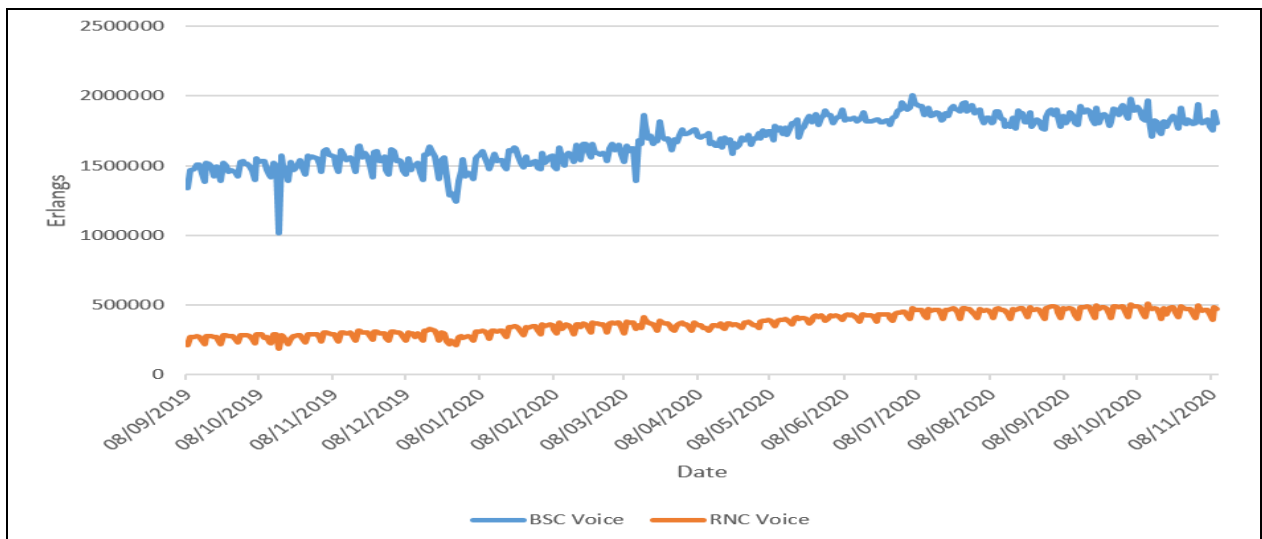


Figure 4.5: 2G (BSC) and 3G (RNC) Voice Traffic Trend

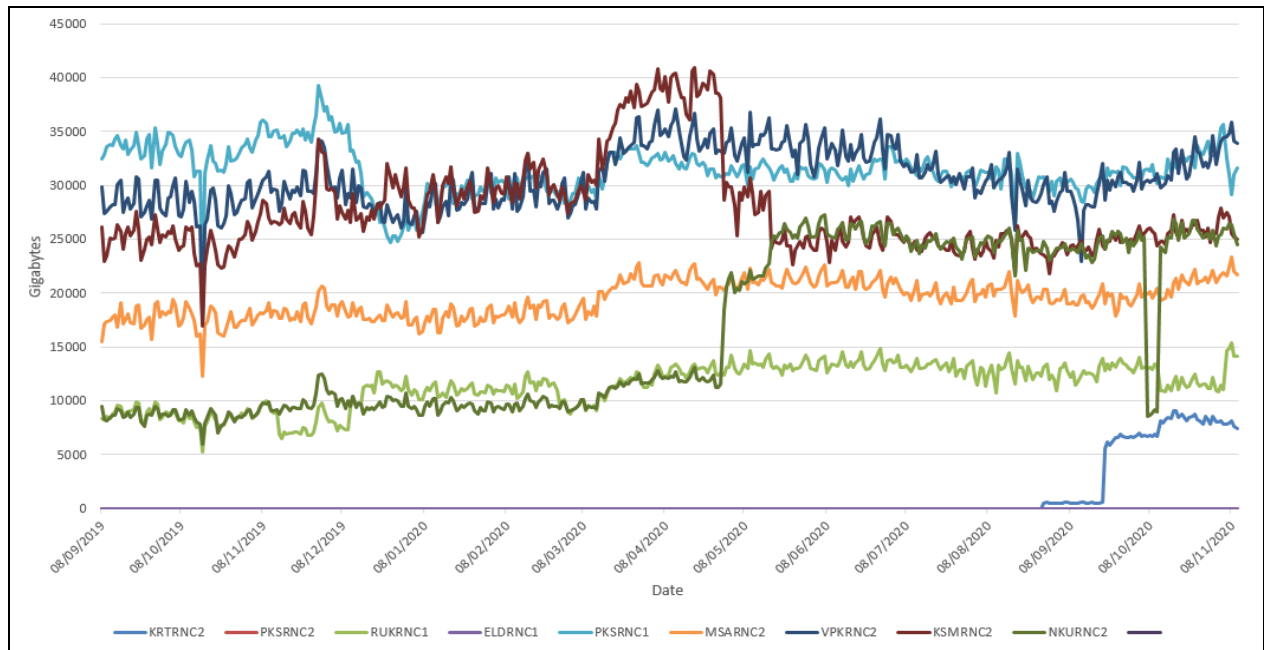


Figure 4.6: 3G (RNC) Daily Data Traffic

Fig 4.6 shows total aggregated network 3G data traffic per day and demonstrates a fairly constant and slight upward trend for the analyzed RNCs. As observed from the trend, KRTRNC2 came online at the beginning of September 2020 as part of network rollout.

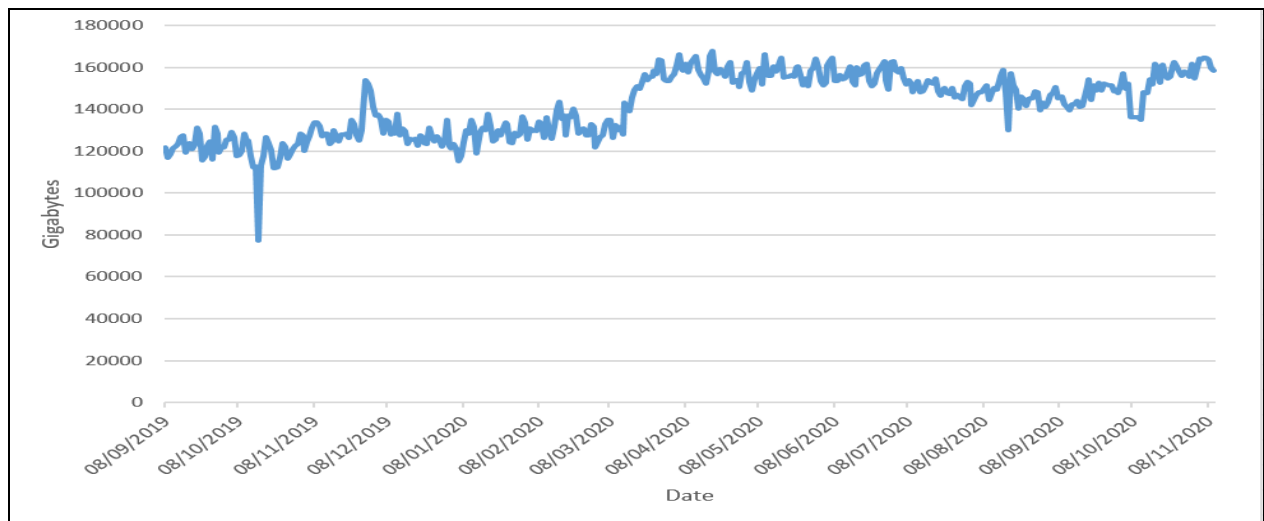


Figure 4.7: 3G (RNC) Data Traffic Trend

A summation of the total daily 3G traffic is shown in Figure 4.7 which depicts a slow increment pattern for 3G which changes from 121,705 GB to 158,205 GB (30% increase) from the first day to the last day of the series.

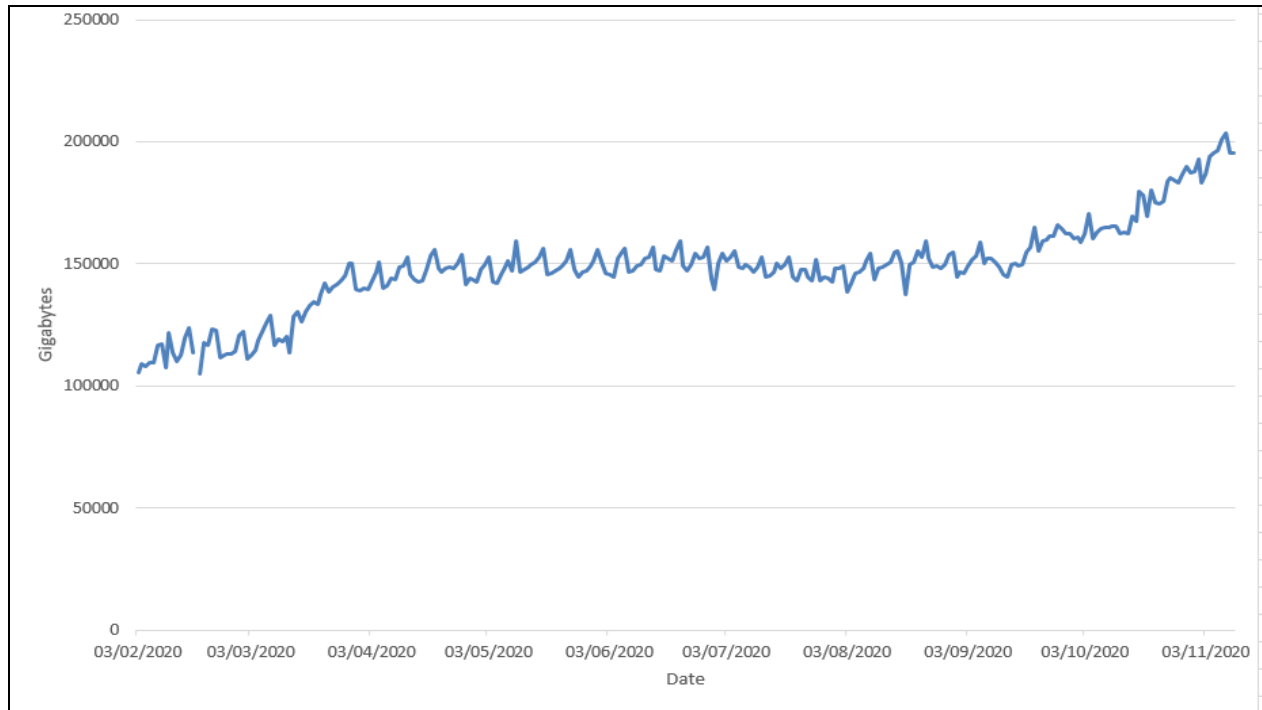


Figure 4.8: LTE Daily Data Traffic

However, as seen in Figure 4.8, LTE data volumes show a significant upward trend from approximately 105,681 GB to 195,578 GB per day which translates to an 85.1% increase over the whole period. As the mobile network operator was piloting the LTE technology on their network, traffic was recorded as a single node.

4.3.2 Hourly Traffic Trends

An analysis of the different traffic series data demonstrates that primarily two traffic trends were evident: the first for hourly (0000 hrs to 2300 hrs) and another giving a daily profile through the week. An arbitrary date of 1st September 2020 which was in the period under research was selected. Figure 4.9 shows that the 2G voice traffic is lowest between 0000 hrs to 0500 hrs.

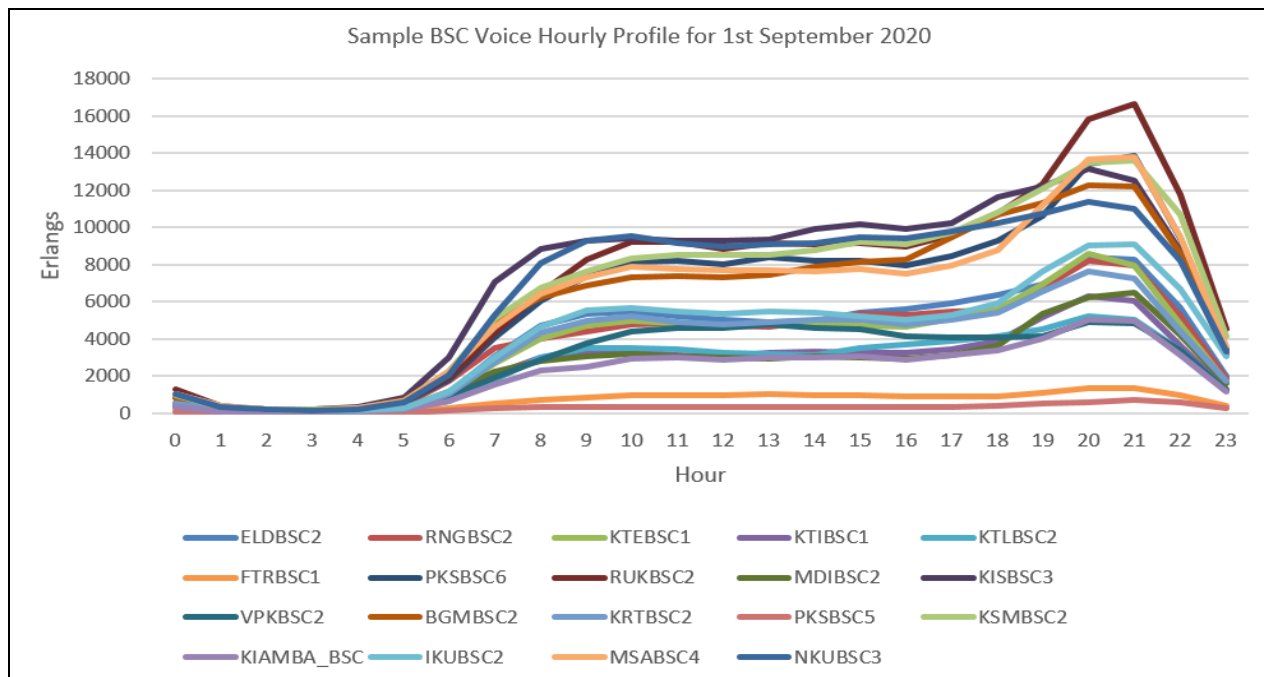


Figure 4.9: 2G (BSC) Hourly Voice Traffic Trend per Day

From Figure 4.9, it can be seen that 2G voice traffic peaks in the evenings, generally rising from 1800 hrs until 2100 hrs. It however has a plateau during daylight hours from 0800 hrs up to about 1700 hrs.

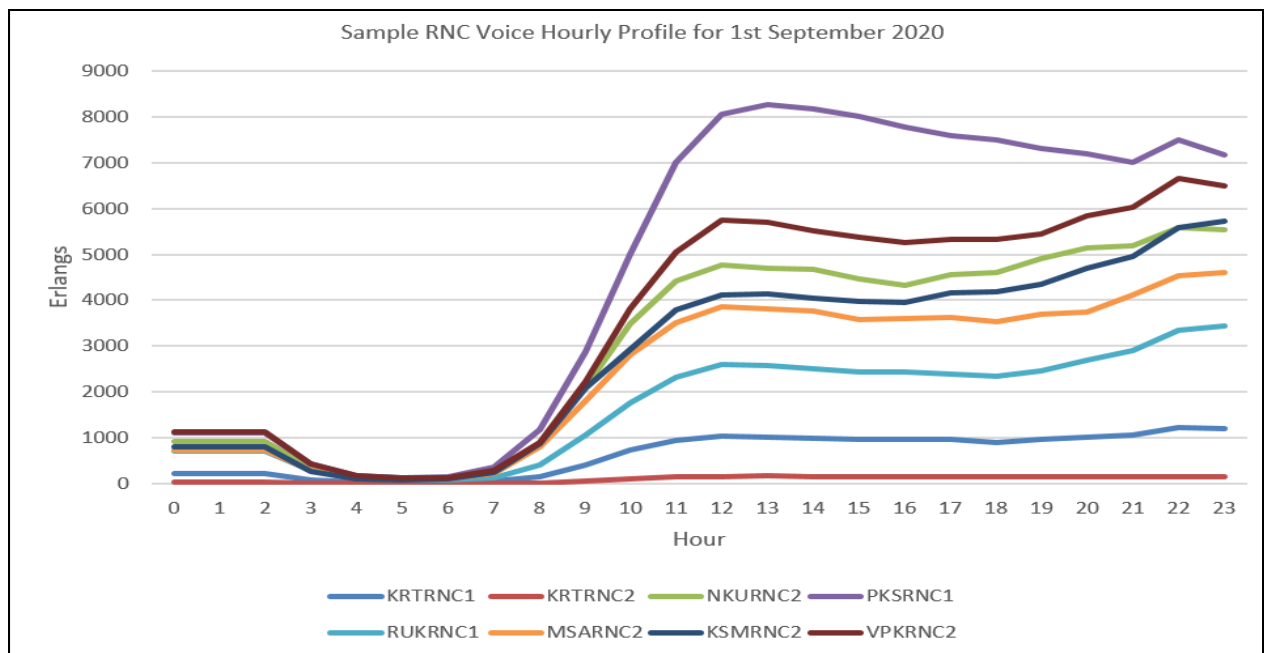


Figure 4.10: 3G (RNC) Hourly Voice Traffic Trend per Day

In Figure 4.10, the 3G hourly voice profile indicates a peak at 1200 hrs and maintains a fairly constant profile afterwards for the rest of the day and evening. It is generally accepted that broadband (3G/4G) voice traffic would peak during midday as there would be a large concentration of users in urban centres (towns /CBDs) as opposed to early in the morning or later in the evenings when users would have migrated back to sub-urban areas where they live, and where 2G services have a higher availability. This is because 3G/4G network nodes are more concentrated in urban settings to cater for the increased peak traffic during business hours

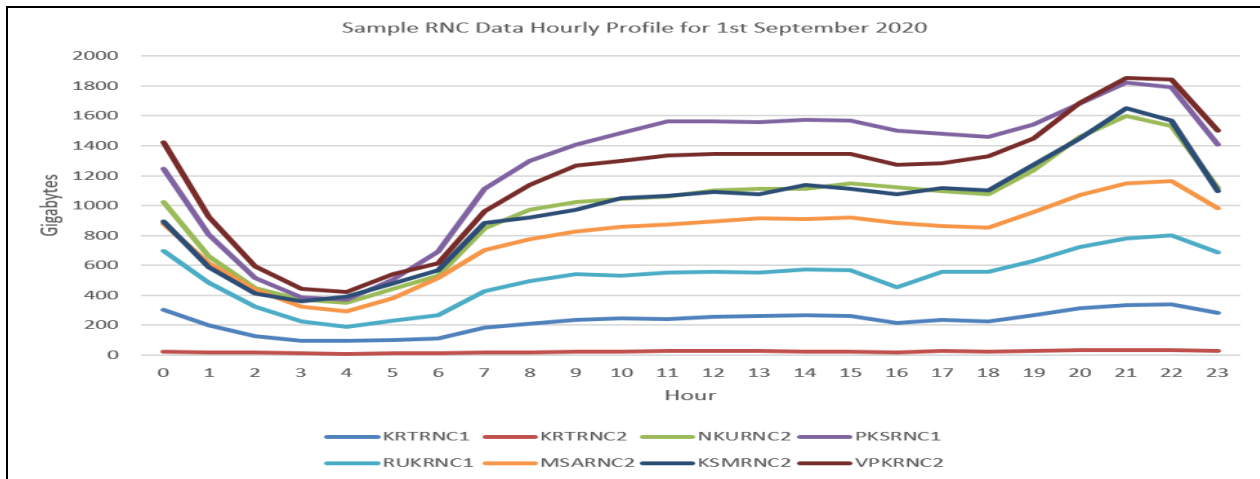


Figure 4.11: 3G (RNC) Hourly Data Traffic Trend per Day

Figures 4.11 and 4.12 show 3G and LTE data at an hourly profile. It can be seen that both 3G and LTE data volumes follow the same trend peaking at about 2200 hrs and lowest volumes witnessed at about 0500 hrs. However, the trend exhibits a fairly constant volume of about 7,000 GB per hour from 1200 hrs to 2000 hrs.

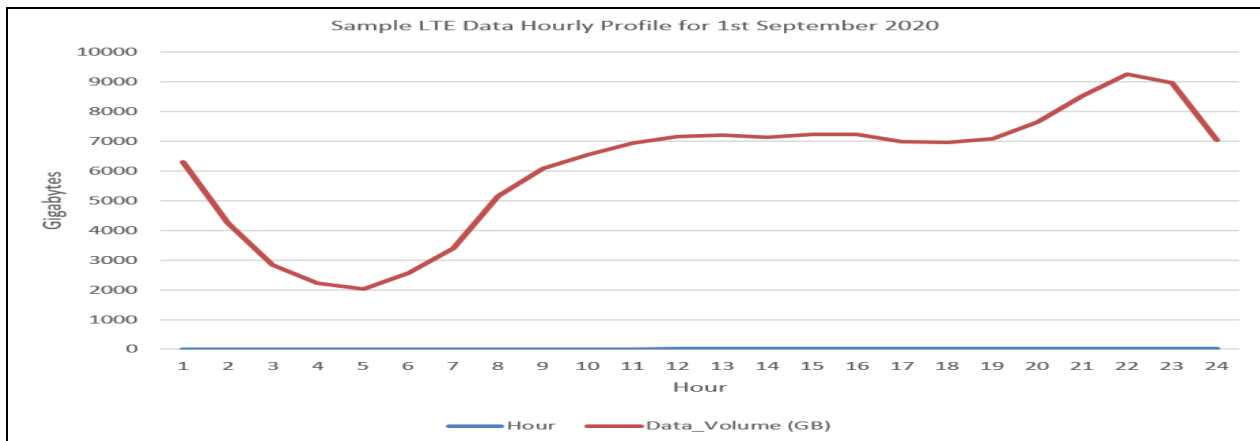


Figure 4.12: LTE Hourly Data Traffic Trend per Day

An arbitrary week of 14th to 20th September 2020 which was in the period under research was selected. Weekly traffic trends generally follow a varying trend across all technologies and services. Voice and data traffic volumes show significant variations in daily patterns as shown in the Figures 4.13, 4.14, 4.15, and 4.16. Demand and usage for most of the cellular traffic recorded at the network nodes is similar over the days analyzed as viewed from this perspective with no evidence of a particular day of the week exhibiting peak traffic loads except for the 3G voice traffic trend that shows lower volumes on the 19th and 20th of September 2020 which fell on a weekend.

However, some variations such as the sudden traffic dips indicated by the trend line for KSMBSC2, can be attributed to network outages as noted in Figure 4.13. This network node underwent scheduled maintenance on 17th and 18th September 2020 which affected its normal traffic trend. Similarly, NKUBSC4 had low traffic volumes on 14th and 15th September 2020 attributed to transmission failure.

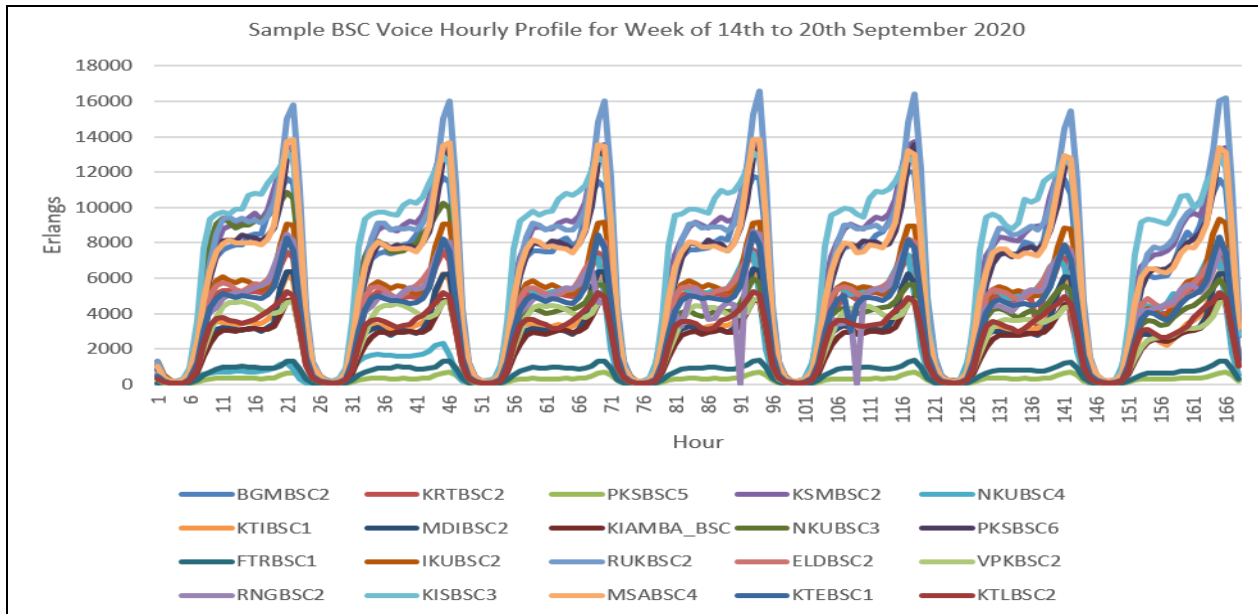


Figure 4.13: 2G (BSC) Hourly Voice Traffic Trend per Week

Network outages will affect the target values in the training of artificial neural networks as they give uncommon conditions that cannot be used as a baseline to perform a forecast.

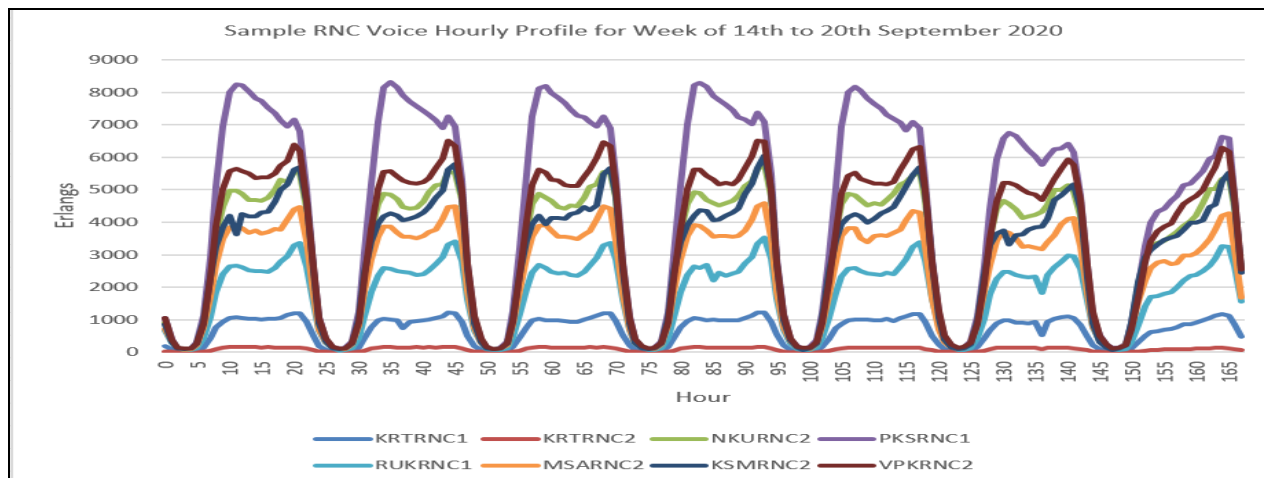


Figure 4.14: 3G (RNC) Hourly Voice Traffic Trend per Week

Variations in traffic trends experienced over different days as seen in Figure 4.14 which shows 3G voice traffic volumes can be attributed to the general mobility of users and service usage trends which vary depending on the day of the week and location of the users which results in having different network nodes providing connectivity services over different periods. In Figure 4.14, PKSRNC1 can be seen to follow a different traffic trend as compared to the other RNCs. This is due to the fact that it serves the central business district in Nairobi, which is the main business hub in Kenya, which has a high number of users during business hours on weekdays. Additionally, the trend shows a variation on the last day of the week, where the traffic is seen to reduce. This can be attributed to the understanding that Kenya, being a predominantly Christian religious society, will have a significant number of users not contributing to network traffic especially on Sunday morning.

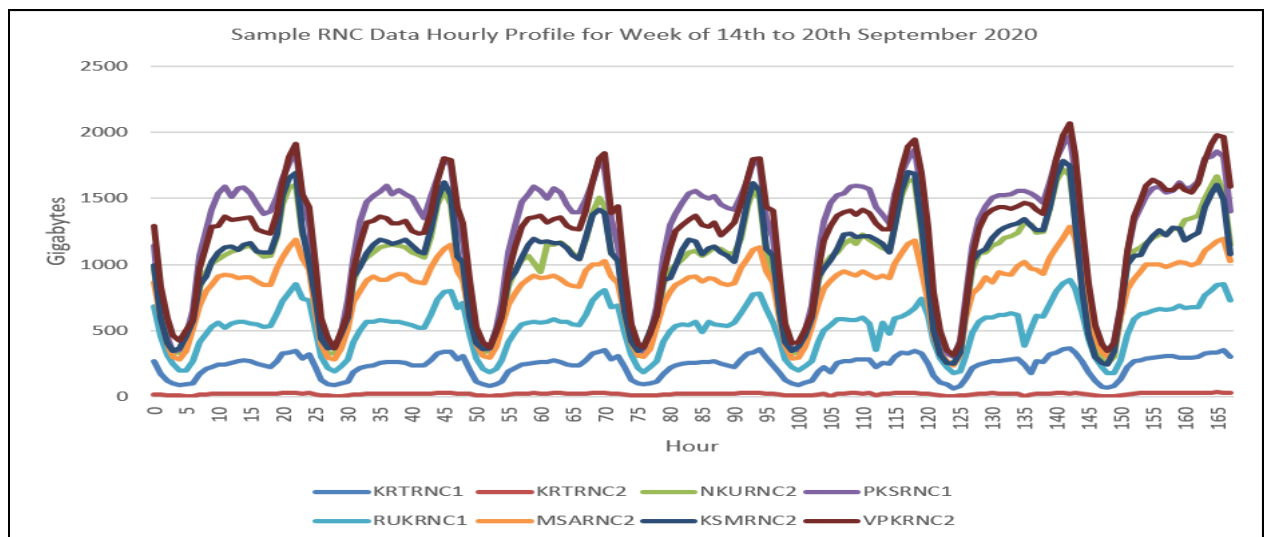


Figure 4.15: 3G (RNC) Hourly Data Traffic Trend per Week

LTE data hourly traffic, as seen in Figure 4.16 was recorded as a single node for the whole network as at the time of study, the mobile network operator was conducting a pilot deployment of the LTE service technology on their network. The hourly trend for the sampled week indicates a similar traffic profile for each day as shown in Figure 4.16.

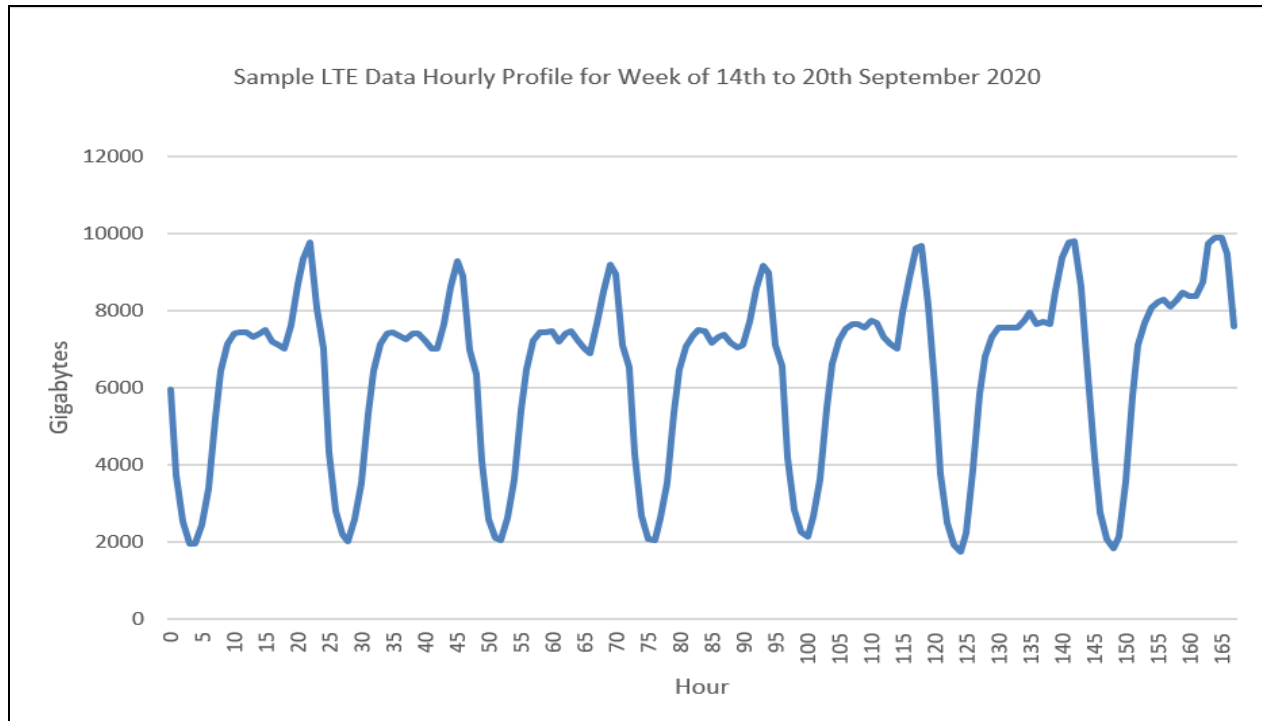


Figure 4.16: LTE Hourly Data Traffic Trend per Week

Small traffic volumes variations can generally be attributed to temporary network faults which are impossible to predict. Additionally, network maintenance and equipment upgrades may lead to disruption of services and traffic. Due to the sensitivity and the financial implications of the cellular services market, such faults are fixed as soon as they occur to prevent customer churn/attrition and loss of revenue.

The different traffic patterns underscore the importance of forecasting traffic to allow for the dynamic allocation of available network resources as well as monitoring of the long term growth of traffic. In addition, the dimension of the evolution of traffic transfer from traditional circuit switched to packet switched could be forecast and analyzed to allow for the re-farming of spectrum resources and financial outlay planning.

It is also of note that different access technologies have different distribution frequency of their bouncing busy hours (BBH). This is shown in Figures 4.17, 4.18, 4.19, and 4.20. Majority of the daily busy hour at the BSC and RNC nodes occur during the evenings.

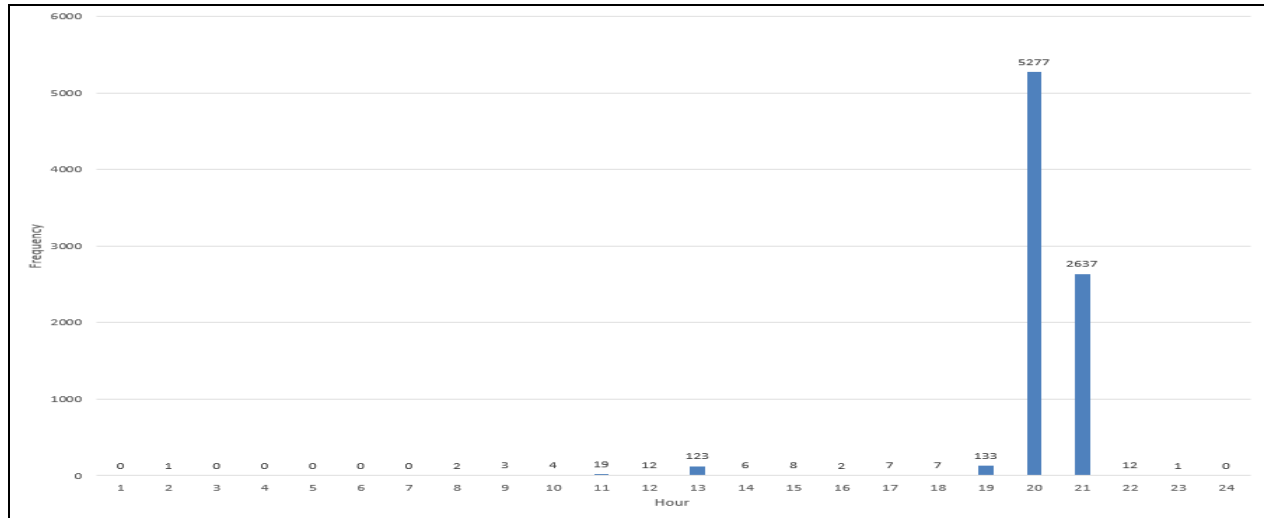


Figure 4.17: 2G (BSC) Voice Traffic BBH Distribution

The Figure 4.17 for 2G voice traffic shows that the BBH mostly occurs from 1900 hrs with the highest concentration of busy hours occurring at 2000 hrs and 2100 hrs when most voice calls are made presumably from homes. However, a small sample of BBH occur at 1300 hrs and can be attributed to BSC nodes serving town centers and business centres because at that time, such localities have a high concentration of users.

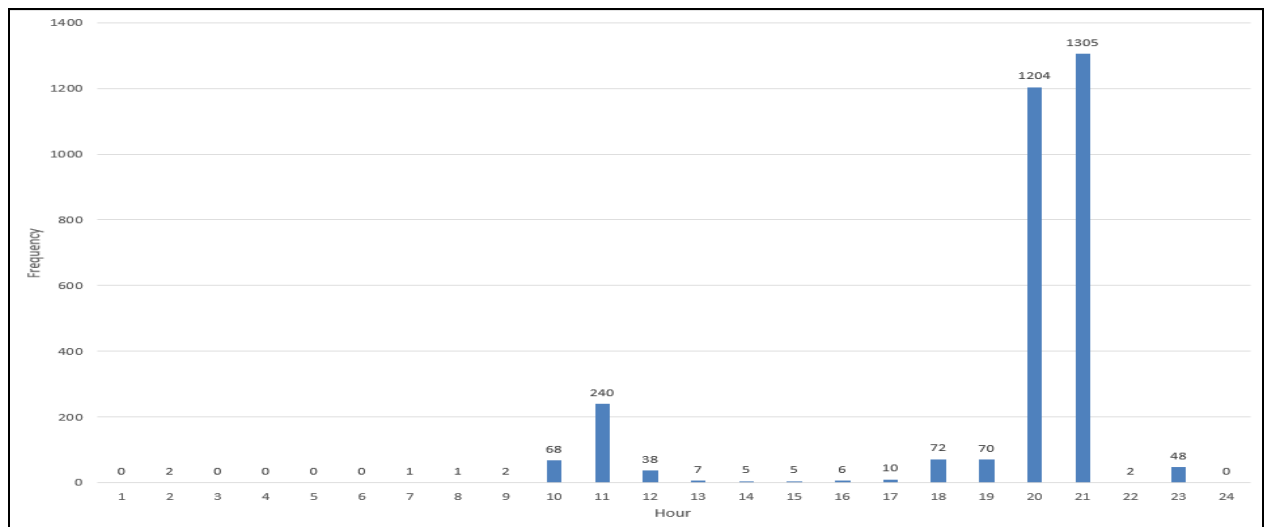


Figure 4.18: 3G (RNC) Voice Traffic BBH Distribution

As can be seen from Figure 4.18, 3G voice has a more distributed occurrence of BBH, however, the period between 2000 hrs and 2100 hrs has the bulk of network traffic. As 3G technology is more prevalent in urban settings, it can be noted that there are more occurrences of BBH in 3G voice as compared to 2G voice traffic around midday. Most RNC nodes serve urban and business districts due to heavier user presence and proliferation of smart devices.

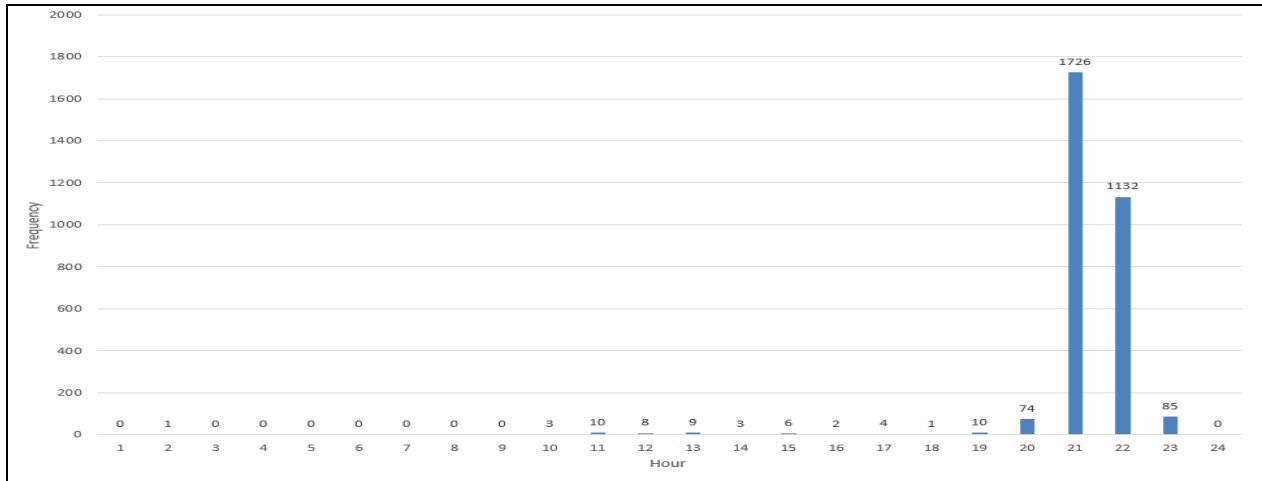


Figure 4.19: 3G (RNC) Data Traffic BBH Distribution

The 3G and LTE data traffic BBH have almost identical distribution with a large portion of traffic occurring at 2100 hrs and to a lesser extent at 2200 hrs. As can be seen from Figures 4.19 and 4.20, a large amount of browsing on mobile devices is done using mobile data at these times as most of the users of these smart devices tend to use Wi-Fi as an internet connectivity mechanism during the day. Many users will resort to the use of mobile data in their homes where there may be lack of access to Wi-Fi and outside official working hours.

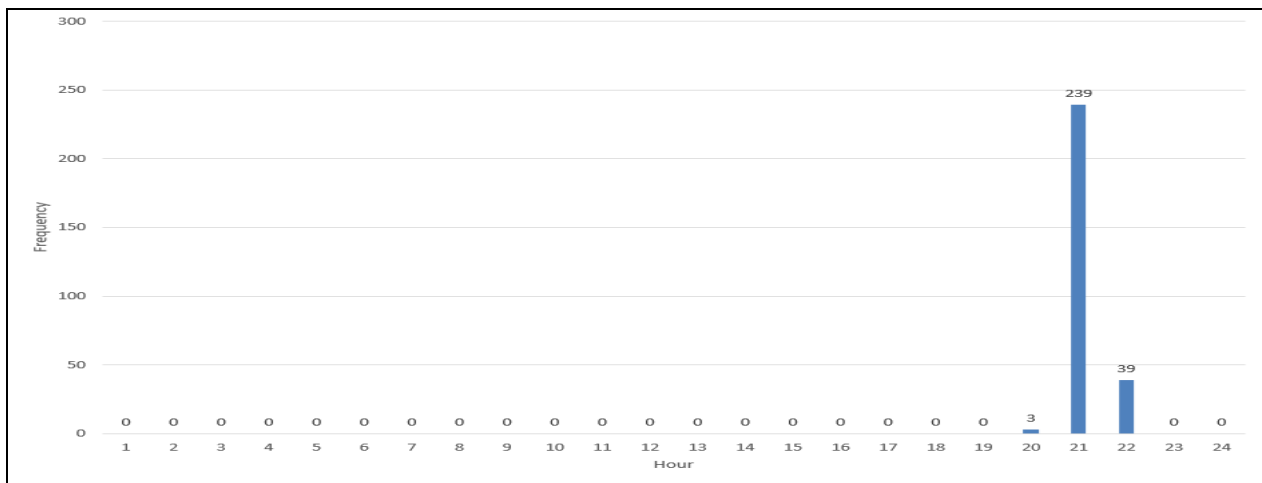


Figure 4.20: LTE Data Traffic BBH Distribution

4.4 SUMMARY

The analysis of the input and target data in this work confirm the application and suitability of neural networks in mobile traffic prediction. There exists a non-linear relationship between the proposed network inputs and their target values. A neural network is therefore a powerful tool that can be used to establish the non-linear mapping relationships which can be used to forecast various elements of the network across all the technologies such as voice and data traffic values in addition to the likelihood of occurrence of busy hours at each network node that is of interest.

CHAPTER 5

DESIGN OF THE ARTIFICIAL NEURAL NETWORK

5.1 DESIGN STEPS

The structure of an ANN determines the level of freedom available to model the data. If the ANN is too simplistic, then the network will not have the ability to learn the function associating the input to output. Resolution of more intricate problems require networks with additional neurons and therefore require more computation. In the instance that the hidden layer is too large, the network must optimize more parameters than there are data vectors in an effort of constraining these parameters. The computational problem could be under-characterized which results in unnecessary computation. An over-complex network will learn the noise (irrelevant or meaningless data) in the data and will not have the ability to generalize. Additional layers need additional computation, however, complex problems will be more efficiently solved.

Learning processes involve input processing elements obtaining input data from an external source and passing this data to hidden processing elements, which conduct uncomplicated, yet, practical mathematical computations with the involvement of input values and weight of the connections. Outputs coming from the hidden processing elements are mapped onto the appropriate threshold function of every processing element and the final outputs are generated. These outputs subsequently form inputs to all processing elements in the succeeding layer, and the computation operations are replicated all through the layers until eventually, outputs are created at the output processing elements. Once output values are created, an output error value is computed through calculating variance between the neural network outputs and target outputs. The whole training operation is iterative, and ceases when a suitably minimal error is obtained. On finalization of a learning operation, the ANN should have the capability of giving output solution(s) when provided with an input dataset hinged on the developed generalized mapping capability. The proposed model in this research had a network architecture composed of an input, an output, and one hidden layer and was trained using supervised learning process with the back-propagation algorithm. In the multilayer neural network, the output is a non-linear functions of the weights which connect the input and hidden units to the output unit. Accordingly, non-linear

equations can be resolved for each output weight. During every epoch (iteration) across the training data, the output weight optimization training method utilizes regular back propagation to refine hidden unit weights and subsequently determine the output weights' non-linear equations. Target vectors and their corresponding input vectors are used to train a NN up to the point it can estimate a function relating input vectors to a particular target value.

Regular back propagation can be looked at as a steep descent algorithm, where progression of the network weights is done in the direction of the negative of the performance function gradient. Appropriately trained back propagation networks tend to provide outputs that are near to expected values when provided with new inputs that have not been encountered before by the network. In general, fresh inputs lead to outputs that resemble the true output for the input vectors that were utilized during the network training that are alike to the new provided inputs. The generalization ability creates a possibility of training networks using indicative sets of target and input pairs and obtain proper solutions and outputs without having to train networks on all conceivable input/output pairings.

Four general stages constitute the training process:

- 1) Assembling of the training data
- 2) Creation of the network object
- 3) Selection of the optimal algorithm and ANN architecture
- 4) Fine testing and finally selecting neural network modules.

5.2 ASSEMBLING OF THE TRAINING DATA

The input and target data was assembled on a spreadsheet forming a matrix composed of input and target vectors. The matrix composition was $(n \times p)$ for input values and $(1 \times p)$ for target values. 'n' is the number of rows with input elements while 'p' represents the number of columns which have unique samples of the inputs. A data clean-up was done to eliminate input values associated with target values that were deemed to be outliers to the general data trends. These were low values approaching zero which would serve to eliminate values recorded at the network nodes during network downtime and outages. Figures below fifty (50) Erlangs for voice traffic and one (1) Gigabyte for mobile data were chosen for elimination from the target data. However,

high values were maintained in the data as these represented recorded peak traffic values. This process is necessary to improve the accuracy of forecasts by eliminating improbable values in the input and target matrices. The resultant matrices are as in Table 5.1 and 5.2.

Table 5.1: Input and Target Matrices for Cellular Traffic Prediction

	Input Matrix (n x p)	Target Matrix (1 x p)
BSC Voice Traffic	18 x 191,754	1 x 191,754
RNC Voice Traffic	17 x 69,099	1 x 69,099
RNC Data Traffic	17 x 72,186	1 x 72,186
LTE Data Traffic	13 x 6,744	1 x 6,744

Table 5.2: Input and Target Matrices for BBH Prediction

	Input Matrix (n x p)	Target Matrix (1 x p)
BSC Voice BBH	13 x 8,254	1 x 8,254
RNC Voice BBH	12 x 3,086	1 x 3,086
RNC Data BBH	12 x 3,074	1 x 3,074
LTE Data BBH	8 x 281	1 x 281

Table 5.1 and 5.2 show the sizes of input and target matrices that resulted from the data cleanup. It is noted that the value of ‘n’ in Table 5.2 is less by 5 as compared to Table 5.1 since the hour variable is a target element in the determination of BBH whereas it is an input element in the determination of the cellular traffic which results in a reduction of 5 columns (hour 1 to hour 24 takes up 5 columns in binary code). Additionally, the value of ‘p’ is significantly less in Table 5.2 as compared to Table 5.1 as BBH is determined once for each day whereas cellular traffic is determined for each hour of the day. Samples of the matrices in Table 5.1 and 5.2 are in the appendix A1.

5.3 CREATION OF THE NETWORK OBJECT

A two-layer feed-forward networks is feasibly able of representing all input-output relationships with a sparse amount of discontinuities, assuming that there are an adequate number of neurons in the hidden layer. A feed-forward NN having a linear transfer function in the output layer and a tan-sigmoid transfer function in the hidden layer was created. The linear output layer allows the NN to process values outside the range -1 to $+1$ since forecasts given are expected to be continuous / non-discrete values of traffic and busy hours. The tan-sigmoid function is most

suitable to be used in MATLAB neural network design because the default ‘mapminmax’ processing function transforms outputs to the range (-1,1) for which the tan-sigmoid transfer function is most adequate. The feedforward network is useful for approximation of functions of any order (or regression) and therefore was deemed suitable for creation of a neural network for purposes of cellular traffic forecasting which is basically a function approximation problem.

The optimal number of neurons in the hidden layer was tested through various trials and selected through adjusting the design of the network and going through the training process and comparing the performance of the Levenberg- Marquardt (LM), Scaled Conjugate Gradient (SCG) and Bayesian Regularization (BR) algorithms on various network architectures. The work in this thesis investigates the variation of the number of neurons in the hidden layer, with the number ranging from ten to fifty in steps of ten neurons. The range and step size were arbitrarily chosen as there is no set rule in the selection of the quantity of neurons in the hidden layer. Cross-validation was used to compare accuracy of forecasts by comparing the average MSE of the trained networks. The networks each had a single output neuron, based on the fact that there is only a single target value (cellular traffic or busy hour) associated with each input vector.

5.4 SELECTION OF OPTIMAL ALGORITHM AND ANN ARCHITECTURE

The different network architectures were trained using a selection of three training algorithms to identify best performance. These algorithms were:

1. **Levenberg-Marquardt:** This algorithm generally needs more memory but less computing time. It is a repetitive method which locates the minimum of a function that is exhibited as the sum of squares of non-linear functions. Training automatically ceases when generalization stops improving, which is evidenced by an increment of the mean square error (MSE) of validation samples [34].
2. **Bayesian Regularization:** This algorithm provides good generalization for noisy, difficult, or small datasets but generally needs more computing time. Training ceases as per adaptive weight minimization (shrinkage/ regularization). The minimization methods

involve applying certain initial distributions on the parameters of the model and penalizes bigger weights in expectation of attaining smoother mapping [35].

3. **Scaled Conjugate Gradient:** This algorithm needs reduced memory. Training automatically ceases when generalization ceases getting better, as demonstrated by a growth in the mean square error (MSE) of the validation samples [36].

These algorithms operate in such a way the weights are advanced in the negative gradient direction. There is no fixed rule on deciding the division ratios for training, validation and testing. However, as a general rule, smaller datasets require a larger ratio of training data as they are prone to problems of overfitting and loss of generalization. Typically, researchers would use a ratio of 80: 10: 10 for smaller datasets and 60: 20: 20 for larger datasets. The default MATLAB ratio is 70: 15: 15 which formed a good compromise for the size of datasets in this research. The MATLAB application performs a random division of target vectors and input vectors into three sets using the 'dividerand' function as below: 70% is used for training, 15% is used to perform validation on the network generalizing and also ceasing of training prior to over-fitting and the final 15% is utilized as an entirely independent test of network generalization.

Training on the input and target vectors proceeds as long as the training results in a reduction of the neural network's overall error on the validation vectors. However, to prevent the training going on indefinitely when convergence is not being achieved, the process is programmed to terminated once 1,000 epochs is achieved. This process termination criterion could be set higher or lower depending on the problem complexity and the size of the dataset. Training will also stop when the error increases over 6 cycles as it is deemed that the network has converged to the point where its output will have the least error when optimized at that point. Any further training results in divergence as opposed to convergence as the network weights are not being optimized in the error reduction gradient. In the wake of the network memorizing the training set (which will be at the cost of poorer generalizing), the training is ceased. This occurs when generalization error starts increasing and it is determined that the performance of the model on the holdout validation dataset begins to degrade. This method spontaneously circumvents the issue of over-fitting, which afflicts many optimization and learning algorithms.

CHAPTER 6

RESULTS AND DISCUSSION

Training, validation and testing were conducted on the cellular traffic data and the performance of the training algorithms summarized. The neural network training stopping criterion was set at a threshold of 1,000 epochs to prevent overfitting and overtraining and further to allow the tested networks to have a degree of generalization. As earlier discussed, this number of epochs was selected as a compromise between the problem complexity and the size of the dataset. However, this figure can be adjusted upwards if general convergence is not achieved during network run simulations. This further prevents a scenario where the training process goes on indefinitely and does not approach convergence (when the error does not fall below a threshold value). The training will cease when the error value does not improve over 6 epochs as the network cannot learn additional input and output relationships past that point. The results and discussions are categorized into:

1. Cellular traffic forecasting, and
2. Bouncing busy hour forecasting

6.1 CELLULAR TRAFFIC FORECASTING

6.1.1 Initial Training, Validation and Testing

The results of performance of the training the different architectures through the Levenberg-Marquardt (LM) algorithm. Bayesian Regularization (BR) and Scaled Conjugate Gradient (SCG) algorithms is as shown in Tables 6.1 to 6.4. Values of individual MSE for Training, Validation and Testing are used to calculate the average MSE.

6.1.1.1 BSC Voice Traffic

Comparison of performance is quantified by consideration of the lowest average MSE across the combination of training algorithms and the hidden layer architecture. As can be seen from the Table 6.1, the training algorithm exhibiting best performance as regards to convergence for BSC Voice is the Levenberg- Marquardt (LM) algorithm. Convergence of the training process depends on the initialization of weights, which is random. This results in different outputs and in extension the performance will vary each time the neural network is trained using the same

algorithm, inputs and target values. The initial round of training, validation and testing has the purpose of selecting and eliminating network and training models that take too long to converge and those with high MSE values through the process of cross-validation.

Table 6.1: BSC Voice Training Algorithm Performance

BSC Voice	Algorithm	Quantity of Neurons in Hidden Layer	Epochs	Average MSE (Training, Validation, Testing)	Time (hrs:min:secs)
	LM	10	701	0.94995	00:06:10
	LM	20	138	0.97092	00:03:01
	LM	30	586	0.97389	00:56:35
	LM	40	854	0.97682	02:08:03
	LM	50	654	0.97752	02:54:37
	BR	10	764	0.96234	00:13:39
	BR	20	1000	0.97293	00:43:21
	BR	30	1000	0.97503	01:37:40
	BR	40	739	0.97737	01:53:11
	BR	50	514	0.97835	01:50:44
	SCG	10	518	0.95163	00:02:47
	SCG	20	1000	0.96236	00:06:34
	SCG	30	1000	0.96387	00:03:26
	SCG	40	1000	0.96711	00:03:06
	SCG	50	1000	0.96892	00:03:25

The BR and SCG algorithms exhibit challenges in achieving convergence to a minimum error value and fail in some instances especially when the network size increases past 10 neurons in the hidden layer. This makes the LM algorithm most suitable to use for the BSC voice traffic dataset.

The lowest average MSE, which is the average of the squared differences between actual values and estimated values, was noted at 0.94995 and a training time of 6 minutes and 10 seconds using the LM algorithm with a network architecture consisting of 10 hidden layer neurons. This was the lowest value of MSE obtained in the results.

6.1.1.2 RNC Voice Traffic

RNC Voice traffic was used as inputs in five different network architectures trained using the LM, BS and SCG algorithms providing the results captured in Table 6.2.

Table 6.2: RNC Voice Traffic Training Algorithm Performance

RNC Voice	Algorithm	Quantity of Neurons in Hidden Layer	Epochs	Average MSE (Training, Validation, Testing)	Time (hrs:min:secs)
	LM	10	238	0.95665	00:00:45
	LM	20	161	0.96538	00:01:01
	LM	30	133	0.96732	00:01:44
	LM	40	111	0.96885	00:03:10
	LM	50	113	0.96819	00:04:24
	BR	10	1000	0.95747	00:03:27
	BR	20	364	0.96604	00:04:58
	BR	30	598	0.96827	00:09:32
	BR	40	921	0.96917	00:36:35
	BR	50	609	0.9696	00:43:31
	SCG	10	265	0.94517	00:00:07
	SCG	20	611	0.95963	00:00:23
	SCG	30	429	0.95301	00:00:20
	SCG	40	848	0.96423	00:00:50
	SCG	50	361	0.95744	00:00:53

The BR algorithm performed unpredictably with changes in the number of neurons in the hidden layer with no convergence being achieved in one instance. The LM algorithm’s 10 neuron architecture had the best average MSE of the LM trained networks with a value of 0.95665, while the BR trained 10 neuron architecture performed with an average MSE value of 0.95747. The best performing test was for the 10 neuron SCG trained network which had an average MSE of 0.94517 and a training time of 7 seconds. It should be noted that cross-validation is the best method to compare performance of trained neural networks due to the variations of performance that appear monotonic because neural networks are stochastic in nature and therefore utilize randomness in initializing weights.

6.1.1.3 RNC Data Traffic

The performance on training of the networks using RNC data traffic as captured in Table 6.3 and shows the performance across network architectures and training algorithms.

Table 6.3: RNC Data Training Algorithm Performance

RNC Data	Algorithm	Quantity of Neurons in Hidden Layer	Epochs	Average MSE (Training, Validation, Testing)	Time (hrs:min:secs)
	LM	10	515	0.95777	00:01:49
	LM	20	59	0.97403	00:00:47
	LM	30	76	0.97597	00:01:18
	LM	40	64	0.97700	00:02:55
	LM	50	94	0.97742	00:07:08
	BR	10	179	0.96689	00:00:41
	BR	20	697	0.97572	00:07:20
	BR	30	410	0.97659	00:09:43
	BR	40	755	0.97743	00:28:42
	BR	50	710	0.97795	00:44:46
	SCG	10	555	0.96402	00:00:16
	SCG	20	567	0.96924	00:00:21
	SCG	30	794	0.97200	00:00:41
	SCG	40	519	0.97147	00:00:30
	SCG	50	748	0.97212	00:00:57

There was convergence in all scenarios tested, however the LM algorithm trained 10 neuron architecture exhibited the best average MSE results at 0.95777 and a training time of 1 minute and 49 seconds.

6.1.1.4 LTE Data Traffic

Table 6.4 shows the results for the test scenarios for LTE data traffic. It can be noted that across all training algorithm tests, convergence times were achieved fairly fast, because the dataset was smaller than those used to train for BSC and RNC traffic. All tested network architectures exhibited nearly similar average MSE results of approximately 0.98. Only one result exhibited lower average MSE across the tests at 0.97896, and this was a 40 neuron architecture trained using the SCG training algorithm Which makes it the best performing network to forecast LTE data traffic according to the dataset provided.

Table 6.4: LTE Data Training Algorithm Performance

LTE Data	Algorithm	Quantity of Neurons in Hidden Layer	Epochs	Average MSE (Training, Validation, Testing)	Time (hrs:min:secs)
	LM	10	104	0.9866	00:00:01
	LM	20	33	0.9877	00:00:01
	LM	30	25	0.98851	00:00:01
	LM	40	18	0.98828	00:00:03
	LM	50	15	0.98769	00:00:03
	BR	10	122	0.98764	00:00:03
	BR	20	228	0.98881	00:00:10
	BR	30	429	0.98921	00:00:31
	BR	40	816	0.98931	00:01:39
	BR	50	806	0.98932	00:03:34
	SCG	10	177	0.98112	00:00:01
	SCG	20	271	0.98209	00:00:01
	SCG	30	183	0.98165	00:00:01
	SCG	40	167	0.97896	00:00:01
	SCG	50	225	0.98395	00:00:01

In conclusion, the best performance, based on average MSE for the traffic samples is as in Table 6.5:

Table 6.5: Summary of Initial Training, Validation and Testing Performance

	No. of Neurons in Hidden Layer	Training Algorithm
BSC Voice Traffic	10	Levenberg-Marquardt
RNC Voice Traffic	10	Scaled Conjugate Gradient
RNC Data Traffic	10	Levenberg-Marquardt
LTE Data Traffic	40	Scaled Conjugate Gradient

The table 6.5 is a general indication of performance, however, during the training of artificial neural networks, random weights and values of gains are assigned at the first iteration which results in differing starting points for each training simulation. Due to the diversity of the data and the numerous non-linear relationships between input and target values, different algorithms and network configurations will have varying error values in their forecasts. Additionally, as

neural networks operate on a “black-box” model, it is difficult to identify what data points resulted in the adjustments of the weights in the hidden layer of the neurons.

Therefore, to fine-tune the cellular traffic forecasting networks, a further five simulation runs were conducted on each with the network architecture used per data set as captured in Table 6.5. They were subsequently trained with the algorithm that exhibited the best performance per data category.

6.1.2 Fine testing and final selection of neural network modules

Five simulation runs were conducted per dataset were tested according to the results obtained in Table 6.5.

6.1.2.1 BSC Voice Traffic

In Table 6.6, the 5 networks tested in this data category showed variations in training, validation and testing time with varying degrees of average MSE.

Table 6.6: BSC Voice Traffic ANN Fine Testing Results

BSC Voice	Training Algorithm	Number of Neurons in Hidden Layer	Simulation Identifier	Epochs	Average MSE (Training, Validation, Test)	Time (seconds)
	LM	10	A	303	0.9577	308
	LM	10	B	563	0.9561	394
	LM	10	C	128	0.96194	75
	LM	10	D	397	0.95989	234
	LM	10	E	263	0.96087	171
	Average			330.8	0.9593	236.4

Comparing the five simulation runs, Simulation B was the best performing simulation with an MSE of 0.9561. However, it was noted that the run that was conducted in the initial training, validation and testing using the LM algorithm 10 neuron network had a better performance with an average MSE of 0.94995 as seen in Table 6.1. As explained before, different runs on the same network using the same training algorithm will post varying results due to the stochastic nature of neural networks. An average of 330.8 epochs, MSE value of 0.9593 and processing time of 236.4 seconds was obtained from five networks A to E. A detailed breakdown of the results can be found in the appendix A3.

6.1.2.2 RNC Voice Traffic

RNC voice traffic prediction results of performance are shown in Table 6.7. The fine tuning network selection indicated Simulation G as the best performing network.

Table 6.7: RNC Voice Traffic ANN fine testing results

RNC Voice	Training Algorithm	Number of Neurons in Hidden Layer	Simulation Identifier	Epochs	Average MSE (Training, Validation, Test)	Time (seconds)
	SCG	10	F	769	0.95327	45
	SCG	10	G	405	0.93557	20
	SCG	10	H	445	0.95625	12
	SCG	10	I	427	0.95263	21
	SCG	10	J	678	0.95575	34
Average				544.8	0.950694	26.4

It is of importance in noting that the input vector size has an impact on processing time and larger data sets require more computation and processing time. As the RNC voice has 69,099 samples as compared to the BSC voice dataset which has 191,754 samples, the processing times in Table 6.7 are lower than those seen in Table 6.6. Simulation G had an average MSE of 0.93577 which was much better than the other networks tested for this data set. An average of 544.8 epochs, MSE value of 0.950694 and processing time of 26.4 seconds was obtained from five networks F to J. A detailed breakdown of the results can be found in the appendix A3.

6.1.2.3 RNC Data Traffic

Simulation K exhibited the best average MSE results as indicated in Table 6.8 for the simulation runs conducted on the RNC data dataset.

Table 6.8: RNC Data Traffic ANN fine testing results

RNC Data	Training Algorithm	Number of Neurons in Hidden Layer	Simulation Identifier	Epochs	Average MSE (Training, Validation, Test)	Time (seconds)
	LM	10	K	515	0.95777	89
	LM	10	L	196	0.96444	43
	LM	10	M	147	0.96272	21
	LM	10	N	245	0.96075	52
	LM	10	O	197	0.96542	43
Average				260	0.96222	49.6

The MSE result for simulation K was 0.95777 although it had the longest training time of 1 minute 29 seconds due to the number of epochs it took to reach convergence which was

significantly more than what networks L, M, N, and O took. However, the MSE value of simulation K was better by 0.3% to that of simulation N which was closest in terms of performance. An average of 260 epochs, MSE value of 0.96222 and processing time of 49.6 seconds was obtained from five simulations K to O. A detailed breakdown of the results can be found in the appendix A3.

6.1.2.4 LTE Data Traffic

The LTE data traffic dataset was used to perform five training runs using the SCG algorithm in a 40 neuron network as a fine-tuning exercise to the initial network selection performed in Table 6.5. The results are as captured in Table 6.9.

Table 6.9: LTE Data Traffic ANN fine testing results

LTE Data	Training Algorithm	Number of Neurons in Hidden Layer	Simulation Identifier	Epochs	Average MSE (Training, Validation, Test)	Time (seconds)
	SCG	40	P	318	0.98743	2
	SCG	40	Q	213	0.97998	1
	SCG	40	R	262	0.98379	1
	SCG	40	S	146	0.97684	1
	SCG	40	T	219	0.9833	2
	Average			231.6	0.982268	1.4

Simulation S had the lowest average MSE of 0.97684 as compared to simulations P, Q, R, and T tested on this dataset. It can be noted that the time required to reach convergence for all the runs was 2 seconds or less and this could be ascribed to the small size of the LTE data traffic dataset which was composed of 6,744 samples. An average of 231.6 epochs, MSE value of 0.982268 and processing time of 1.4 seconds was obtained from five networks P to T. A detailed breakdown of the results can be found in the appendix A3.

6.1.3 Discussion of Results for Artificial Neural Network Performance on Cellular Traffic Data

In general, results on algorithm performance and network architecture exhibited a variety of outcomes as regards to achieving convergence and MSE. It was apparent that larger data sets (BSC voice, RNC voice and RNC data, in that order) had better performance results when a

smaller number of neurons were utilized in the design of the hidden layer of the network. It could be said that with more data points (input and target vectors) to develop a baseline for forecasting, the neural network requires fewer hidden layer neurons to discover patterns in the data due to the larger quantity of input to target mapping examples to learn from. Table 6.10 shows the test results of the networks selected for every set of the traffic data with the average MSE expected in the forecasting of network traffic at the hourly basis. This can be seen in the summary of the fine testing of the selected networks and training algorithms. BSC voice, RNC Voice and RNC Data all had best forecasting performance exhibited by networks that had ten (10) hidden layer neurons while LTE data traffic exhibited best average MSE performance in networks that had forty (40) neurons.

Table 6.10: Summary of Fine Testing Traffic Training Algorithm Performance

	No. of Neurons in Hidden Layer	Algorithm	Average Epoch Count	Average of Average MSE Achieved	Average Processing Time (Seconds)
BSC Voice Traffic	10	LM	330.8	0.9593	236.4
RNC Voice Traffic	10	SCG	544.8	0.950694	26.4
RNC Data Traffic	10	LM	260	0.96222	49.6
LTE Data Traffic	40	SCG	231.6	0.982268	1.4

Training algorithm performance demonstrated clear preference for Levenberg-Marquardt (LM) and the Scaled Conjugate Gradient (SCG) algorithms across all networks tested. Convergence was achieved in all tested scenarios before the maximum epoch count of 1,000 was reached. The Bayesian Regularization algorithm did not achieve convergence in some instances with the 1,000 epoch count reached before optimal performance of training and testing was achieved. Additionally, the average MSE was higher than the other tested algorithms.

Table 5.1 showed the sizes of the data matrices tested and a relationship can be seen where larger sample data sets (BSC voice, RNC voice and RNC data) require a larger number of hidden layer neurons to perform best results for network fitting and pattern recognition as compared to smaller data sets such as the data for LTE data traffic.

Availability of clean data is a major influence on the size of the datasets that are used in the training of artificial neural networks and as was seen in Table 5.1, the input and target vectors

vary from 191, 754 for BSC voice to 6,744 for LTE data traffic which contributed to the variation of the time performance for the networks and algorithms tested. Therefore, comparison can only be done for runs within the same dataset and conclusions drawn from each as captured in Table 6.10.

6.2 BOUNCING BUSY HOUR (BBH) FORECASTING

Prediction of BBH follows a similar procedure as the one used for cellular traffic.

6.2.1 Training, Validation and Testing

The performance of the Levenberg- Marquardt (LM), Bayesian Regularization (BR) and Scaled Conjugate Gradient (SCG) algorithms were employed across different network architectures and on the input and target vectors, as shown in Table 5.2. The input and target data are used to train, validate and test varying number of neurons in the hidden layer with the number varying from ten to fifty in steps of ten neurons. As neural networks are stochastic models, weights are initialized randomly which results in varying degrees of MSE and outputs. Cross-validation was then used to compare performance by comparing the average MSE of the trained networks.

However, the number of elements and samples per element reduce in number as a BBH is calculated for each day unlike network traffic that is predicted in an hourly manner. It follows that the processing time is much lower. Due to small size of input and target matrices for the BBH as seen in Table 5.2, it is assumed that fine testing will not add much value because overfitting and over validation can occur when using small data sets which leads to a loss of generalization. Therefore, one round of training, validation and testing was conducted per BBH data category. Table 6.11 is a summary of the BBH distribution seen in Figures 4.17, 4.18, 4.19, and 4.20.

The results of performance of the training the different architectures for BBH prediction through the different algorithms is given in Tables 6.14 to 6.18.

Table 6.11: BBH Distribution per Technology

Hour	BSC BBH Distribution	RNC Voice Traffic BBH Distribution	RNC Data Traffic BBH Distribution	LTE BBH Distribution
0	0	0	0	0
1	0	0	0	0
2	1	2	1	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	1	0	0
8	2	1	0	0
9	3	2	0	0
10	4	68	3	0
11	19	240	10	0
12	12	38	8	0
13	123	7	9	0
14	6	5	3	0
15	8	5	6	0
16	2	6	2	0
17	7	10	4	0
18	7	72	1	0
19	133	70	10	0
20	5277	1204	74	3
21	2637	1305	1726	239
22	12	2	1132	39
23	1	48	85	0
Total Samples	8254	3086	3074	281

6.2.1.1 BSC Voice BBH

Table 6.12 shows that a network architecture of 20 neurons in hidden layer trained using the Scaled Conjugate Gradient (SCG) algorithm gives the best performance for BSC voice BBH forecast. The average MSE is 0.17546 with an epoch value of 56 that converges in less than 1 second.

Table 6.12: BSC Voice BBH Forecasting Tests

BSC Voice	Algorithm	Quantity of Neurons in Hidden Layer	Epochs	Average MSE (Training, Validation, Testing)	Time (hrs:min:secs)
	LM	10	21	0.34264	00:00:01
	LM	20	10	0.28210	00:00:01
	LM	30	15	0.35219	00:00:02
	LM	40	12	0.34616	00:00:02
	LM	50	19	0.97752	00:00:06
	BR	10	220	0.36531	00:00:01
	BR	20	573	0.37053	00:00:41
	BR	30	547	0.36128	00:05:26
	BR	40	838	0.37292	00:02:22
	BR	50	459	0.97835	00:03:19
	SCG	10	111	0.23130	< 00:00:01
	SCG	20	56	0.17546	< 00:00:01
	SCG	30	70	0.19219	<00: 00:01
	SCG	40	89	0.20144	< 00:00:01
	SCG	50	100	0.19804	< 00:00:01

From Table 6.12, an observation can be made that the SCG algorithm outperforms the LM and BR algorithms in terms of processing time and the average MSE achieved for all the network architectures that were investigated.

The convergence time of the training were all below 1 second for the SCG algorithm, however the MATLAB software rounds these times to the nearest second. The performance metrics for the network configuration for BSC voice BBH which was the SCG trained 20 neuron network is found in the appendix A3.

6.2.1.2 RNC Voice BBH

Network training, validation and testing of RNC voice BBH gave the results captured in Table 6.13. Overall processing times for all algorithms and architectures considered were quite fast. The Scaled Conjugate Gradient algorithm with a 30 neuron architecture posted best results with an average MSE of 0.67827 which converged at epoch 25.

Table 6.13: RNC Voice BBH Forecasting Tests

RNC Voice	Algorithm	Quantity of Neurons in Hidden Layer	Epochs	Average MSE (Training, Validation, Testing)	Time (hrs:min:secs)
	LM	10	26	0.88874	< 00:00:01
	LM	20	17	0.89213	<00: 00:01
	LM	30	51	0.89521	< 00:00:01
	LM	40	13	0.89295	< 00:00:01
	LM	50	17	0.89743	< 00:00:01
	BR	10	313	0.8911	00:00:02
	BR	20	322	0.90183	00:00:05
	BR	30	448	0.90306	00:00:14
	BR	40	420	0.90359	00:00:25
	BR	50	932	0.90552	00:01:31
	SCG	10	72	0.84981	<00: 00:01
	SCG	20	47	0.81424	< 00:00:01
	SCG	30	25	0.67827	< 00:00:01
	SCG	40	39	0.76285	< 00:00:01
	SCG	50	76	0.8306	< 00:00:01

The performance metrics for the best performing network configuration for RNC voice BBH is found in the appendix A3. Time for convergence was less than 1 seconds for the SCG trained 30 neuron network.

6.2.1.3 RNC Data BBH

Table 6.14: RNC Data BBH Forecasting Tests

RNC Data	Algorithm	Quantity of Neurons in Hidden Layer	Epochs	Average MSE (Training, Validation, Testing)	Time (hrs:min:secs)
	LM	10	10	0.094561	<00: 00:01
	LM	20	10	0.23165	< 00:00:01
	LM	30	10	0.22297	< 00:00:01
	LM	40	10	0.16633	< 00:00:01
	LM	50	13	0.16585	<00: 00:01
	BR	10	198	0.33025	00:00:01
	BR	20	467	0.34021	00:00:07
	BR	30	435	0.32464	00:00:17
	BR	40	850	0.36427	00:00:58
	BR	50	1000	0.28815	00:01:46
	SCG	10	19	0.083091	< 00:00:01
	SCG	20	41	0.087673	<00: 00:01
	SCG	30	43	0.11532	<00: 00:01
	SCG	40	57	0.19465	< 00:00:01
	SCG	50	89	0.18783	< 00:00:01

Table 6.14 shows that a network architecture of 10 hidden layer neurons trained with the SCG algorithm giving the best performance for RNC Data BBH forecast. The performance metrics for the best performing network configuration for RNC Data BBH has an average MSE of 0.083091 with an epoch value of 19 that converges within 1 second. The details of the performance of this network is found in the appendix A3.

6.2.1.4 LTE Data BBH

Forecasting of LTE Data BBH gave the results captured in Table 6.15. Overall processing times for all algorithms and architectures considered were fairly fast overall. The Scaled Conjugate Gradient algorithm with a 10 neuron architecture posted best results with an average MSE of 0.049477 which converged at epoch 10.

Table 6.15: LTE Data BBH Forecasting Tests

LTE Data	Algorithm	Quantity of Neurons in Hidden Layer	Epochs	Average MSE (Training, Validation, Testing)	Time (hrs:min:secs)
	LM	10	10	0.37486	<00: 00:01
	LM	20	9	0.34035	< 00:00:01
	LM	30	6	0.41204	< 00:00:01
	LM	40	5	0.49976	< 00:00:01
	LM	50	5	0.47871	< 00:00:01
	BR	10	1000	0.25769	00:00:08
	BR	20	1000	0.25556	00:00:07
	BR	30	1000	0.25063	00:00:12
	BR	40	1000	0.23793	00:00:21
	BR	50	1000	0.25104	00:00:33
	SCG	10	10	0.049477	< 00:00:01
	SCG	20	26	0.44341	<00: 00:01
	SCG	30	27	0.38111	<00: 00:01
	SCG	40	65	0.38801	< 00:00:01
	SCG	50	39	0.42951	< 00:00:01

The performance metrics for the best performing network configuration for LTE Data BBH are found in the appendix A3.

6.2.2 Discussion of Results for Artificial Neural Network Performance on Bouncing Busy Hour (BBH) Data

In all tested scenarios, the Scaled Conjugate Gradient (SCG) algorithm achieved the best performance for all BBH datasets. The average MSE of networks trained by the SCG algorithm, was lower than the one achieved using the Levenberg-Marquardt (LM) algorithm and that of the Bayesian Regularization (BR) algorithm as captured in Table 6.12 to 6.15. The LM and BR trained networks reported higher average MSE when compared to the SCG trained networks. The BR algorithm did not achieve convergence before the maximum epoch count of 1,000 in some instances when tested against RNC data and LTE Data BBH samples.

Table 5.2 shows the sizes of the data matrices used in the training of the BBH forecasting neural networks. No definitive relationship can be identified which shows that dataset sample sizes exhibit better performance based on the number of neurons in the hidden layer of the designed networks. The best performance of the networks was exhibited between 10 to 30 neurons as shown in Table 6.16 which shows the performance of the networks selected for each of the BBH data sets with the average MSE that is expected in the forecasting of BBH.

Table 6.16: Summary of BBH Training Algorithm Performance

	Quantity of Neurons in Hidden Layer	Algorithm	Epoch Count	Average MSE Achieved	Processing Time (hr:min:secs)
BSC Voice BBH	20	Scaled Conjugate Gradient	56	0.17546	<00:00:01
RNC Voice BBH	30	Scaled Conjugate Gradient	25	0.67827	< 00:00:01
RNC Data BBH	10	Scaled Conjugate Gradient	19	0.083091	< 00:00:01
LTE Data BBH	10	Scaled Conjugate Gradient	10	0.049477	< 00:00:01

6.3 SUMMARY OF RESULTS

The results confirm the applicability of neural network techniques to the forecast of cellular network traffic and bouncing busy hour (BBH) prediction. Neural networks are capable of identifying the non-linear relationships between the influencing factors, which form the input vectors, and cellular traffic and busy hours, which form the target vectors. The low MSE values obtained during the runs and simulations indicates that the performance of the designed networks will achieve fairly accurate forecasts with a low deviation between actual and predicted values. The lower the MSE values, the closer the forecast value is to the actual value. Performance

details of the training, validation and testing of the best performing networks are found from Figures A 3.1 to A 3.8 in the appendix.

The MATLAB designed networks is capable of giving outputs based on inputs at the command line prompts where the input variables relevant to a future date, time, network node identifier, and special day identifier in the similar format that the neural network had been trained with. The output would be a value that depicts the cellular traffic or the BBH depending on the type of forecast required. It is assumed that due to the dynamic nature of telecommunications sector, the neural networks will require periodic fine tuning and further training on any additional quantifiable factors which could influence traffic patterns that would become apparent during the passage of time and change in technology.

The conditions vary from one test case to another, such as the size of datasets, the input elements (factors), the traffic type (voice/ data), the type of network node, time, the technology platform of the dataset (2G, 3G, LTE). Therefore, average MSE evaluation has been done for the training algorithms and tested architecture of the neural networks that were considered in this research.

In the creation and simulation of the NNs, the networks exhibited different optimal results with different training algorithms and various network architectures. Due to the multi factor approach used in this work (period of the day, day of the week, special days, type of network node, type of cellular traffic, technology in use), and the various cellular network elements targeted (Base station controllers, radio network controllers, LTE relay nodes), data set sizes across the different technologies had an influence on the specific algorithm and network configuration that could be used to give best prediction values. Therefore, due to the diversity of the sample and element types, a diversified approach for each dataset was used to identify the most suitable algorithms and network configurations to be used for each cellular technology under study.

After the fine testing stage/process that was used to test the error performance that is to be expected in forecasting cellular traffic and busy hours, the selected neural networks chosen as the most suitable for making cellular traffic forecasts had average MSE values that ranged from

0.950694 to 0.982268 as recorded in Table 6.10 which shows the suitability of neural networks in forecasting cellular traffic.

The implication is that the neural networks have a high degree of accuracy and at the same time it has a degree of generality, much similar to human reasoning process. This generalizing capability allows the network to process a set of input vectors that it has not encountered before and be able to come up with forecasts for cellular traffic across all the target technologies (2G, 3G, LTE). To make a forecast, the trained network will be provided a set of input vectors which is in the same format as the dataset that was used to train it. For example, in this research, a prediction can be made of a particular hour on a particular day/date and a specific cellular network node such as BSC voice traffic by giving an input of: month, day of the week, special days (public holidays), hour of the day, and BSC/ RNC/ LTE node identifier (which is of interest) in the 18 row binary format described in Chapter 4 of this thesis.

BBH prediction had average MSE values ranging from a minimum of 0.049477 for LTE Data and a maximum of 0.67827 for RNC voice as seen from Table 6.16. Due to the fewer neural network input vectors needed (as busy hour is predicted once a day as opposed to traffic which is predicted for every hour in a day), and the limited variance of chances of occurrence of the BBH outside Hour 20 or 21 in a day, the neural network can quickly learn the pattern of occurrence of a busy hour in a particular day. In summary, larger input and target training datasets provide a higher degree of generalization due to exposure to a wide range of possible test conditions (as evidenced by higher MSE value) while smaller input and target data sets give near perfect models that give high accuracy (low MSE value) but low generalization capability. However, the neural network has to have a degree of accuracy and generalization to perform forecasting tasks.

Due to the stochastic properties of an ANN, and the opaqueness of the hidden layer that follows a “black-box” model where any function can be approximated but any study of its structure will not provide any insight on its inner workings. Therefore, the design and selection of an ideal neural network to make an accurate yet generalized forecast is an iterative trial and error process, that requires cross-validation, as the neural networks are data specific.

CHAPTER 7

CONCLUSION AND RECOMMENDATIONS

7.1 CONCLUSION

Results achieved in this research confirm the relevance and application of neural networks in cellular voice and data traffic and BBH forecasting. The ANN was capable of determining the non-linear association which was existent connecting the historical traffic and hourly data it was supplied in the training phase as indicated by the low average MSE observed that give a bearing on the accuracy of predictions of what the traffic and likelihood of the occurrence of the busy hour in a particular network node (BSC, RNC, LTE relay node) in a particular day would be when given suitable input and target variables. The design and testing of the artificial neural network modules adopted a multi-factor approach that incorporated input vectors that were composed of the network node identity (which identifies the location and area served), radio access network (RAN) technology (2G, 3G, LTE), traffic type (voice, data). time factors (hour, day of the week), special days (public holidays) as well as maximum hourly traffic per day (to determine the busy hour).

As outlined in the methodology, the created networks have been designed and trained to have a accuracy and generalization in the prediction of:

1. The hourly volumes for BSC and RNC voice traffic
2. The hourly volumes for RNC and LTE data traffic
3. The Bouncing Busy Hour for BSC and RNC voice traffic
4. The Bouncing Busy Hour for RNC and LTE data traffic

To make predictions, an input matrix is created composed of the variables such as the time, date (of interest) as well as the target network node identifier. As highlighted in previous chapters, this input matrix would have to be in the same format (order) that was used in training the neural network. The trained neural network will give output values that are predictions of traffic and BBH. With accurate predictions of hourly traffic, a trend of the expected traffic growth can be obtained by summing predicted volumes over a time period of interest i.e. to get the total traffic

in a day/ week, the predicted traffic values in each hour are totaled over the time horizon of interest. Armed with this information, a network planner can dimension a network and design for expansion and optimization to seamlessly accommodate future traffic demands so as to maintain QoS and QoE requirements.

From this research, a number of factors were identified to influence traffic and BBH. These were time/ day/ month factors, occurrence of special days, type of network node and the radio access technology in use. These factors were used in tandem with traffic and BBH values spanning one year to train networks that had the capability to predict hourly traffic and daily BBH. From the results of this research, Table 7.1 summarizes the recommendations for design of neural networks for cellular traffic and BBH.

Table 7.1: Recommended network design per data category

	No. of Neurons in Hidden Layer	Algorithm
BSC Voice Traffic	10	Levenberg- Marquardt
RNC Voice Traffic	10	Scaled Conjugate Gradient
RNC Data Traffic	10	Levenberg- Marquardt
LTE Data Traffic	40	Scaled Conjugate Gradient
BSC Voice BBH	20	Scaled Conjugate Gradient
RNC Voice BBH	30	Scaled Conjugate Gradient
RNC Data BBH	10	Scaled Conjugate Gradient
LTE Data BBH	10	Scaled Conjugate Gradient

Artificial neural networks, therefore can be powerful tools for decision making and planning in mobile service provision companies in as far as deployment of network elements and radio resources. ANN’s ability to learn non-linear relationships between input and target vectors speaks to their flexibility and power. Additionally, it is unnecessary to conceive an algorithm to conduct a particular task; i.e. it is unnecessary to know or make sense of the internal workings of that problem.

The specific objectives of the work carried out in this report, as highlighted below, have been addressed and accomplished.

- i. To determine the factors influencing cellular traffic volumes and incorporate them as inputs to improve the accuracy of forecasts. The factors identified and used in this research were: month of the year, day of the week, special days such as public holidays, hour of the day and BSC/ RNC/ LTE node identifiers.
- ii. To implement a neural network based forecasting model for cellular traffic. Through design and cross validation through performance comparison between various designed and tested networks, a model was developed for cellular voice and cellular data based on data from a Kenyan MNO spanning one year.
- iii. To validate the accuracy of the method by analysis the predicted values to actual values using the mean square error (MSE). Validation and testing with the results of this analysis was conducted in Chapter 6 of this research with a breakdown and comparison of the average MSE achieved through simulation runs.

In conclusion, the overall objective of this research was to explore the viability of the utilization of ANNs in the forecasting of cellular traffic volumes and further, to design artificial neural networks using MATLAB software with an aim to ascertain the combination of algorithm and neural network architecture that would give high accuracy in the prediction of cellular traffic and BBH. This was achieved with the ANN modules developed using MATLAB software and the performance of the modules analyzed by cross-validation. The resultant MSE values as enumerated in the discussion of results in Chapter 6, show the usability of the created ANN modules in predicting cellular voice and data traffic as well as the bouncing busy hours.

7.2 RECOMMENDATION FOR FUTURE WORK

Future research on this field can assimilate information pertaining to the economic factors such as consumer expenditure on services, mobile service penetration levels and subscriptions, special service offerings and availability of feature/smart devices in a particular area so as to provide a bigger set of input vectors for the neural network to learn from. This will improve the accuracy of forecasts by incorporating factors that influence cellular traffic.

In this research, the created neural networks were based on the service technology and traffic type. However, the aspect of network specialization, which is the utilization of a NN for peak traffic periods of the day and a different NN for other hours of the day, could be incorporated to provide better accuracy of forecasts especially at peak traffic times.

The accuracy of the forecasts of a neural network is dependent on the quality of data that it is trained on. Missing data and outliers negatively impact the performance of the neural network. Data preprocessing techniques could be employed to develop good quality datasets through methods such as stochastic neighbour embedding which reduces the high dimensional map of features into a lower dimensional, probability density distribution.

REFERENCES

- [1] M. A. Raheem and O. U. Okereke "A Neural Network Approach to GSM Traffic Congestion Prediction" American Journal of Engineering Research (AJER) e-ISSN , 2014: 2320-0847 p-ISSN : 2320-0936 Volume-03, Issue-11, pp-131-138
- [2] P. Darwood, I. Oppermann, S. Jakas and W. Linton, "Mobile network traffic forecasting," Vehicular Technology Conference Fall 2000. IEEE VTS Fall VTC2000. 52nd Vehicular Technology Conference (Cat. No.00CH37152), pp. 2932-2936 vol.6, doi: 10.1109/VETEFCF.2000.886853.
- [3] I. Z. Kovacs, P. Mogensen, B. Christensen and R. Jarvela, "Mobile Broadband Traffic Forecast Modeling for Network Evolution Studies," IEEE Vehicular Technology Conference (VTC Fall), 2011, pp. 1-5, doi: 10.1109/VETEFCF.2011.6092960.
- [4] R. Afkhami and F. M. Yazdi, "Application of neural networks for short-term load forecasting," IEEE Power India Conference, 2006, pp. 5 pp.-, doi: 10.1109/POWERI.2006.1632536.
- [5] Frédéric PUJOL "Mobile traffic forecasts 2010-2020 & offloading solutions" Idate consulting and research at <https://documents.pub/reader/full/mobile-traffic-forecasts-2010-2020-offloading-solutions-pdf> [accessed 15th February
- [6] Yanhua Yu, Meina Song, Zhijun Ren and Junde Song, "Network traffic analysis and prediction based on APM," 2011 6th International Conference on Pervasive Computing and Applications, 2011, pp. 275-280, doi: 10.1109/ICPCA.2011.6106517.
- [7] E. O. Oladeji, E. N. Onwuka and M. A. Aibinu, "Determination of voice traffic busy hour and traffic forecasting in Global System of Mobile Communication (GSM) in Nigeria," 2013 IEEE 11th Malaysia International Conference on Communications (MICC), pp. 184-189, doi: 10.1109/MICC.2013.6805822.
- [8] Zhang, DongLing & Jia, Zhenhong & Qin, XiZhong & Li, DianJun & Du, ChaoBen & Chen, Li & Sheng, Lei & Li, Hong. "Busy Hour Traffic of Wireless Mobile Communication Forecasting Based on Hidden Markov Model" Advances in Intelligent and Soft Computing 2012, pp 169. 607-612. 10.1007/978-3-642-30223-7_96.

- [9] Pauli Murto, "Neural Network Models For Short-Term Load Forecasting" PhD thesis Helsinki University Of Technology, Department of Engineering Physics and Mathematics
- [10] Graupe Daniel, "Principles of Artificial Neural Networks 2nd Edition, Advanced Series on Circuits and Systems" – Vol. 6, World Scientific Publishing Co. Pte. Ltd 2007
- [11] Ummuhan Basaran Filik, Mehmet Kurban. "A New Approach for the Short-Term Load Forecasting with Autoregressive and Artificial Neural Network Models". International Journal of Computational Intelligence Research. 2007. ISSN 0973-1873 Vol.3, No.1, pp. 66–71 Research India Publications
- [12] Fausett, Laurene V. Fundamentals of Neural Networks: Architectures, Algorithms, and Applications. Englewood Cliffs, NJ: Prentice-Hall, 1994. Print.
- [13]. Ben Krose, Patrick van der Smagt, "An Introduction to Neural Networks" The University of Amsterdam, 8th Edition 1996.
- [14] Myint Myint Yi, Khin Sandar Linn, and Marlar Kyaw "Implementation of Neural Network Based Electricity Load Forecasting" World Academy of Science, Engineering and Technology 42, 2008
- [15] Kutsurelis, Jason. "Forecasting Financial Markets Using Neural Networks: An Analysis Of Methods And Accuracy" Calhoun: The NPS Institutional Archive DSpace Repository 2001.
- [16] K. Stordahl and N. K. Elnegaard, "Long-term mobile data traffic forecasts for the Western European market," 15th International Telecommunications Network Strategy and Planning Symposium, 2012, pp. 1-6, doi: 10.1109/2012.6381714
- [17] P. Svoboda, M. Buerger and M. Rupp, "Forecasting of traffic load in a live 3G packet switched core network," 6th International Symposium on Communication Systems, Networks and Digital Signal Processing, 2008, pp. 433-437, doi: 10.1109/CSNDSP.2008.4610775.
- [18] Y. Yu, J. Wang, M. Song and J. Song, "Network Traffic Prediction and Result Analysis Based on Seasonal ARIMA and Correlation Coefficient," International Conference on Intelligent System Design and Engineering Application, 2010, pp. 980-983, doi: 10.1109/ISDEA.2010.335

- [19] Y. Yu, M. Song, Y. Fu and J. Song, "Traffic prediction in 3G mobile networks based on multifractal exploration," in *Tsinghua Science and Technology*, 2013 vol. 18, no. 4, pp. 398-405, doi: 10.1109/TST.2013.6574678.
- [20] J. Guo, Y. Peng, X. Peng, Q. Chen, J. Yu and Y. Dai, "Traffic forecasting for mobile networks with multiplicative seasonal ARIMA models," 9th International Conference on Electronic Measurement & Instruments, 2009, pp. 3-377-3-380, doi: 10.1109/ICEMI.2009.5274287.
- [21] S. Wang, J. Guo, Q. Liu and X. Peng, "On-Line Traffic Forecasting of Mobile Communication System," First International Conference on Pervasive Computing, Signal Processing and Applications, 2010, pp. 97-100, doi: 10.1109/PCSPA.2010.32.
- [22] Khan, Muhammad & Abraham, Ajith. "Short Term Load Forecasting Models in Czech Republic Using Soft Computing Paradigms" 2004 at <https://arxiv.org/abs/cs/0405051>
- [23] I. A. Lawal, S. A. Abdulkarim, M. K. Hassan and J. M. Sadiq, "Improving HSDPA Traffic Forecasting Using Ensemble of Neural Networks," 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 308-313, doi: 10.1109/ICMLA.2016.0057.
- [24] Chaoyun Zhang and Paul Patras.. "Long-Term Mobile Traffic Forecasting Using Deep Spatio-Temporal Neural Networks". In *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc '18) Association for Computing Machinery*, New York, NY, USA, 231–240.
- [25] S. Medhn, B. Seifu, A. Salem and D. Hailemariam, "Mobile data traffic forecasting in UMTS networks based on SARIMA model: The case of Addis Ababa, Ethiopia," *IEEE AFRICON*, 2017, pp. 285-290, doi: 10.1109/AFRCON.2017.8095496.
- [26] Mejia, Jose & Ochoa-Zezzati, Alberto & Cruz-Mejia, Oliverio. "Traffic Forecasting on Mobile Networks Using 3D Convolutional Layers" *Mobile Networks and Applications* 2020. 25. 10.1007/s11036-020-01554-y.
- [27] A. Kirmaz, D. S. Michalopoulos, I. Balan and W. Gerstaecker, "Mobile Network Traffic Forecasting Using Artificial Neural Networks," 2020 28th International Symposium on

Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 1-7, doi: 10.1109/MASCOTS50786.2020.9285949.

- [28] C. Gijón, M. Toril, S. Luna-Ramírez, M. L. Marí-Altozano, and J. M. Ruiz-Avilés, "Long-Term Data Traffic Forecasting for Network Dimensioning in LTE with Short Time Series," *Electronics*, vol. 10, no. 10, p. 1151, [Online]. Available: <http://dx.doi.org/10.3390/electronics10101151>
- [29] F. Batool and S. A. Khan, "Traffic estimation and real time prediction using adhoc networks," *Proceedings of the IEEE Symposium on Emerging Technologies*, 2005, pp. 264-269, doi: 10.1109/ICET.2005.1558892.
- [30] Paulo Cortez, Miguel Rio, Miguel Rocha, Pedro Sousa "Multi-scale Internet traffic forecasting using neural networks and time series methods" *Blackwell Publishing Ltd Expert Systems*, 2012, Vol. 29, No. 2
- [31] Junita Mohamad-Saleh, Brian S. Hoyle. "Improved Neural Network Performance Using Principal Component Analysis on Matlab" *International Journal of The Computer, the Internet and Management*, 2008, Vol.16. N.o.2. pp 1-8
- [32] Dubravko Miljković "Brief Review of Self-Organizing Maps" 2017 *Croatian Society for Information and Communication Technology, Electronics and Microelectronics MIPRO /CTS* pp. 1252-1257
- [33] Vieira, Robson & Paiva, Rafael & Hulkkonen, Jari & Jarvela, Rauli & Iida, Renato & Säily, Mikko & Tavares, Fernando & Niemelä, Kari. "GSM evolution importance in re-farming 900 MHz band." 2014, 38th. *Vehicular Technology Conference, IEEE* pp 1-5. 10.1109/VETECF.2010.5594534.
- [34] Lourakis, Manolis I. A. "A Brief Description of the Levenberg-Marquardt Algorithm Implemented by levmar." *Foundation for Research and Technology - Hellas (FORTH)*, 2005.
- [35] Okut, Hayrettin. "Bayesian regularized neural networks for small n big p data. *Artificial neural networks-models and applications*", 2016: pp 21-23.

- [36] Møller, Martin, Moller, M.F. “A Scaled Conjugate Gradient Algorithm For Fast Supervised Learning. Neural Networks” Neural Networks. 6. 1993 6, pp 525-533. doi: 10.1016/S0893-6080(05)80056-5.

BIBLIOGRAPHY

1. U. Hentschel, A. Schmidt and A. Polze, "Predictable Communication for Mobile Systems," 2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, 2011, pp. 24-28, doi: 10.1109/ISORC.2011.13.
2. Xiaodong Wang and Ximing Hu "Mobile Traffic Forecasting based on the Relative Position of Multi-scale Average Lines" 2012 IEEE International Conference on Information Science and Technology Wuhan, Hubei, China; March 23-25, 2012
3. Xiuting Yu, Xizhong Qin, Zhenhong Jia, Chuanling Cao and Chun Chang "Telephone Traffic Forecasting Based on Grey Neural Network Optimized by Improved Particle Swarm Optimization Algorithm" International Journal of Hybrid Information Technology Vol.8, No.1 (2015), pp.1-10
4. M. Papadopouli, E. Raftopoulos and H. Shen, "Evaluation of short-term traffic forecasting algorithms in wireless networks," 2006 2nd Conference on Next Generation Internet Design and Engineering, 2006. pp. 8-109, doi: 10.1109/NGI.2006.1678229.
5. D. Zhang, S. Wang, D. Meng and X. Xu, "A Ten-year traffic forecast study for China," 2014 XXXIth URSI General Assembly and Scientific Symposium (URSI GASS), 2014, pp. 1-4, doi: 10.1109/URSIGASS.2014.6929276.
6. Liu, X., Fang, X., Qin, Z. et al. "A Short-Term Forecasting Algorithm for Network Traffic Based on Chaos Theory and SVM". Network System Management 2011, pp 427–447

APPENDIX

A1 INPUT AND TARGET MATRICES

A1.1 Sample BSC Voice Input (Blue) and Target (Yellow) Matrix for Hour 0 (0000 to 0100 hrs) and Hour 1 (0100 to 0200 hrs) for 8th September 2019

Month Binary Code				Day of the Week in BC			Special Day in BC	Hour	Hour Binary Code				BSC_Name	BSC Binary code				Voice traffic (Erl)	
1	0	0	1	1	1	1	0	0	0	0	0	0	RNGBSC2	1	0	0	1	1	439.26
1	0	0	1	1	1	1	0	0	0	0	0	0	NKUBSC3	0	1	1	1	1	1054.81
1	0	0	1	1	1	1	0	0	0	0	0	0	BGMBSC2	0	0	0	0	1	601.89
1	0	0	1	1	1	1	0	0	0	0	0	0	KTIBSC1	0	1	0	1	1	339.32
1	0	0	1	1	1	1	0	0	0	0	0	0	ELDBSC2	0	0	0	1	0	583.58
1	0	0	1	1	1	1	0	0	0	0	0	0	KISBSC3	0	0	1	1	0	800.33
1	0	0	1	1	1	1	0	0	0	0	0	0	KSMBSC2	0	1	0	0	0	920.42
1	0	0	1	1	1	1	0	0	0	0	0	0	PKSBSC5	1	0	0	0	1	628.92
1	0	0	1	1	1	1	0	0	0	0	0	0	PKSBSC6	1	0	0	1	0	2009.27
1	0	0	1	1	1	1	0	0	0	0	0	0	RUKBSC2	1	0	1	0	0	1012.29
1	0	0	1	1	1	1	0	0	0	0	0	0	IKUBSC2	0	0	1	0	0	746.97
1	0	0	1	1	1	1	0	0	0	0	0	0	KIAMBA_BSC	0	0	1	0	1	340.84
1	0	0	1	1	1	1	0	0	0	0	0	0	FTRBSC1	0	0	0	1	1	130.6
1	0	0	1	1	1	1	0	0	0	0	0	0	VPKBSC2	1	0	1	0	1	321.19
1	0	0	1	1	1	1	0	0	0	0	0	0	MDIBSC2	0	1	1	0	1	419.13
1	0	0	1	1	1	1	0	0	0	0	0	0	MSABSC4	0	1	1	1	0	1205.06
1	0	0	1	1	1	1	0	0	0	0	0	0	KRTBSC2	0	0	1	1	1	529.68
1	0	0	1	1	1	1	0	0	0	0	0	0	KTLBSC2	0	1	1	0	0	249.05
1	0	0	1	1	1	1	0	1	0	0	0	0	NKUBSC3	0	1	1	1	1	398.94
1	0	0	1	1	1	1	0	1	0	0	0	0	RUKBSC2	1	0	1	0	0	334.19
1	0	0	1	1	1	1	0	1	0	0	0	0	KIAMBA_BSC	0	0	1	0	1	117.06
1	0	0	1	1	1	1	0	1	0	0	0	0	VPKBSC2	1	0	1	0	1	110.32
1	0	0	1	1	1	1	0	1	0	0	0	0	KISBSC3	0	0	1	1	0	275.78
1	0	0	1	1	1	1	0	1	0	0	0	0	KTLBSC2	0	1	1	0	0	84.03

A1.2 Sample RNC Voice and Data Input (Blue) and Targets (Yellow) Matrix for Hour 0 (0000 to 0100 hrs) and Hour 1 (0100 to 0200 hrs) for 8th September 2019

Month Number in BC				Day of the Week in BC				Special Day in BC	HOUR ID	Hour number in BC				rnc_name	RNC Binary code				Voice traffic (Erl)	Data Volume (GB)
1	0	0	1	1	1	1	1	0	0	0	0	0	0	NKURNC2	0	1	1	0	304.52	348.54
1	0	0	1	1	1	1	1	0	0	0	0	0	0	KRTRNC1	0	0	1	0	0	0
1	0	0	1	1	1	1	1	0	0	0	0	0	0	KSMRNC2	0	1	0	0	835.67	791.06
1	0	0	1	1	1	1	1	0	0	0	0	0	0	PKSRNC1	0	1	1	1	1080.79	1337.69
1	0	0	1	1	1	1	1	0	0	0	0	0	0	MSARNC2	0	1	0	1	564.76	764.13
1	0	0	1	1	1	1	1	0	0	0	0	0	0	RUKRNC1	1	0	0	1	328.11	363.05
1	0	0	1	1	1	1	1	0	0	0	0	0	0	VPKRNC2	1	0	1	0	870.64	1193.88
1	0	0	1	1	1	1	1	0	1	0	0	0	1	NKURNC2	0	1	1	0	120.88	216.22
1	0	0	1	1	1	1	1	0	1	0	0	0	1	RUKRNC1	1	0	0	1	138.14	224.16
1	0	0	1	1	1	1	1	0	1	0	0	0	1	KRTRNC1	0	0	1	0	0	0
1	0	0	1	1	1	1	1	0	1	0	0	0	1	KSMRNC2	0	1	0	0	306.36	450.73
1	0	0	1	1	1	1	1	0	1	0	0	0	1	PKSRNC1	0	1	1	1	481.85	882.86
1	0	0	1	1	1	1	1	0	1	0	0	0	1	MSARNC2	0	1	0	1	291.38	530.24
1	0	0	1	1	1	1	1	0	1	0	0	0	1	VPKRNC2	1	0	1	0	363.76	770.67

A1.3 Sample LTE Data Input (Blue) and Target (Yellow) Matrix for Hour 0 (0000 to 0100 hrs) to Hour 23 (2300 to 0000 hrs) for 3rd February 2020

Month Number in BC				Day of the Week in BC			Special Day in BC	Hour	Hour number in BC					Data_Volume (GB)
0	0	1	0	0	0	1	0	0	0	0	0	0	0	3268.9024
0	0	1	0	0	0	1	0	1	0	0	0	0	1	1830.727473
0	0	1	0	0	0	1	0	2	0	0	0	1	0	1118.509809
0	0	1	0	0	0	1	0	3	0	0	0	1	1	875.1222832
0	0	1	0	0	0	1	0	4	0	0	1	0	0	868.4543173
0	0	1	0	0	0	1	0	5	0	0	1	0	1	1384.766502
0	0	1	0	0	0	1	0	6	0	0	1	1	0	2696.950598
0	0	1	0	0	0	1	0	7	0	0	1	1	1	4072.005837
0	0	1	0	0	0	1	0	8	0	1	0	0	0	4692.341674
0	0	1	0	0	0	1	0	9	0	1	0	0	1	4908.244876
0	0	1	0	0	0	1	0	10	0	1	0	1	0	5112.9291
0	0	1	0	0	0	1	0	11	0	1	0	1	1	5190.640786
0	0	1	0	0	0	1	0	12	0	1	1	0	0	5235.479092
0	0	1	0	0	0	1	0	13	0	1	1	0	1	5297.043561
0	0	1	0	0	0	1	0	14	0	1	1	1	0	5271.396329
0	0	1	0	0	0	1	0	15	0	1	1	1	1	5224.80168
0	0	1	0	0	0	1	0	16	1	0	0	0	0	5325.564619
0	0	1	0	0	0	1	0	17	1	0	0	0	1	5449.872474
0	0	1	0	0	0	1	0	18	1	0	0	1	0	5533.517317
0	0	1	0	0	0	1	0	19	1	0	0	1	1	5870.521343
0	0	1	0	0	0	1	0	20	1	0	1	0	0	6594.488233
0	0	1	0	0	0	1	0	21	1	0	1	0	1	7269.939226
0	0	1	0	0	0	1	0	22	1	0	1	1	0	7082.638393
0	0	1	0	0	0	1	0	23	1	0	1	1	1	5506.66749

A2 SELF-ORGANIZING MAPS (SOM) PLOTS

A2.1 BSC Voice

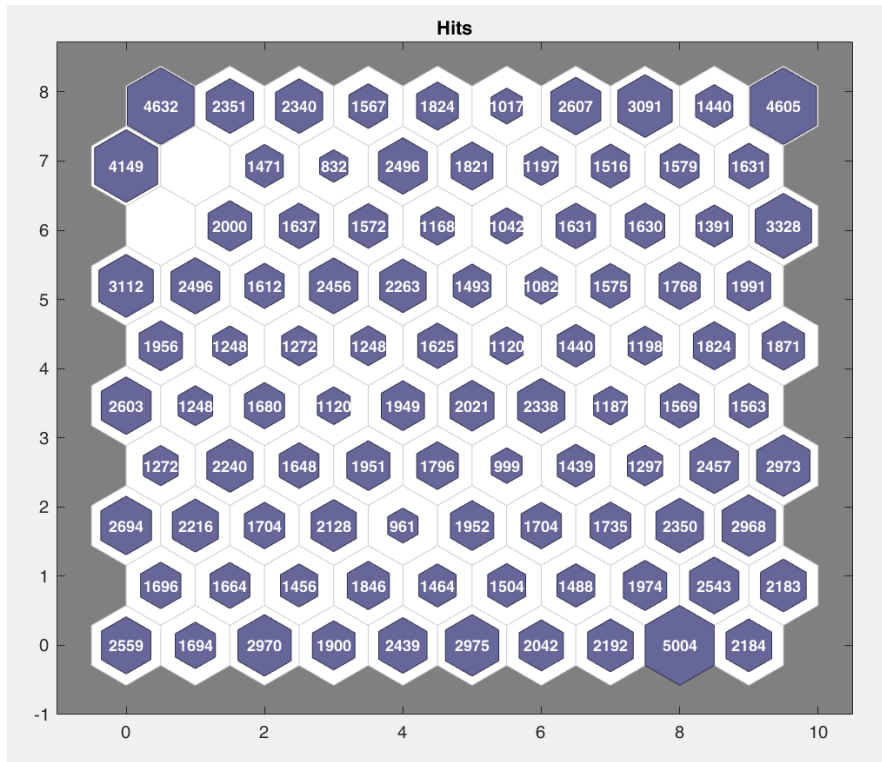


Figure A2.1.1 Sample Hits Plot for Input Variables for BSC Voice Traffic

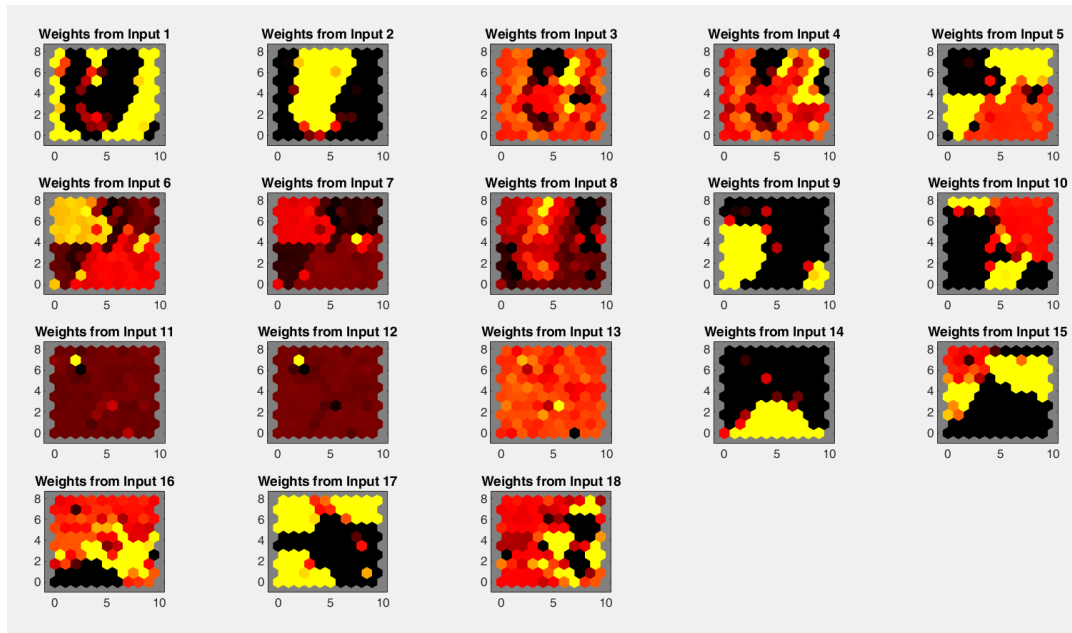


Figure A2.1.2: BSC Voice Traffic Input Variables Weight Planes

A2.2 RNC Voice

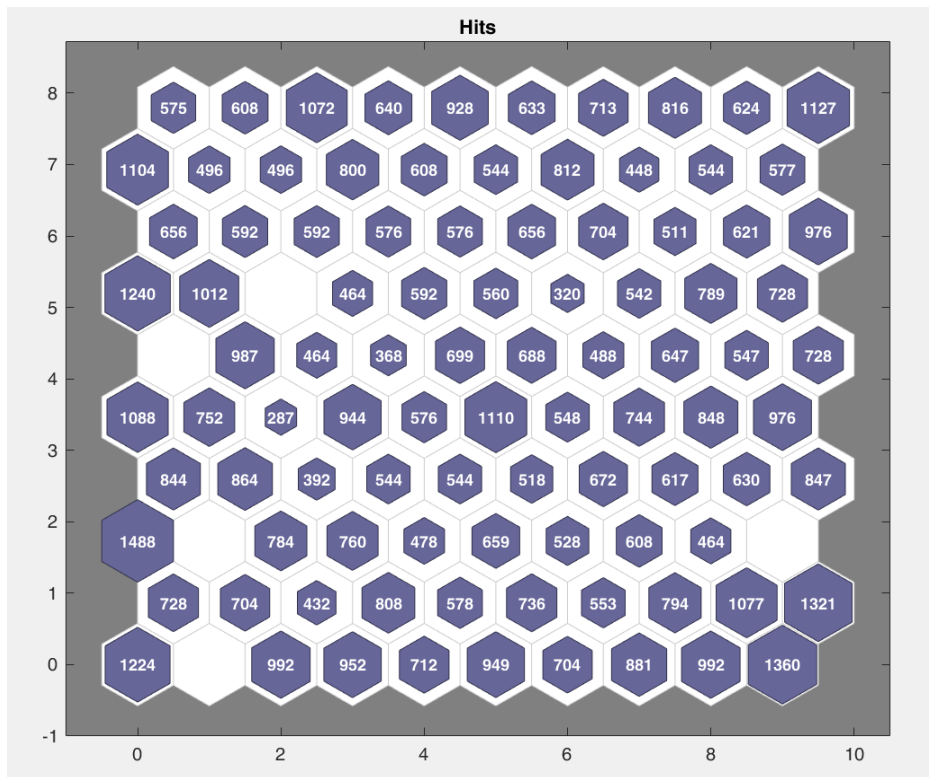


Figure A2.2.1: Sample Hits Plot for Input Variables for RNC Voice Traffic

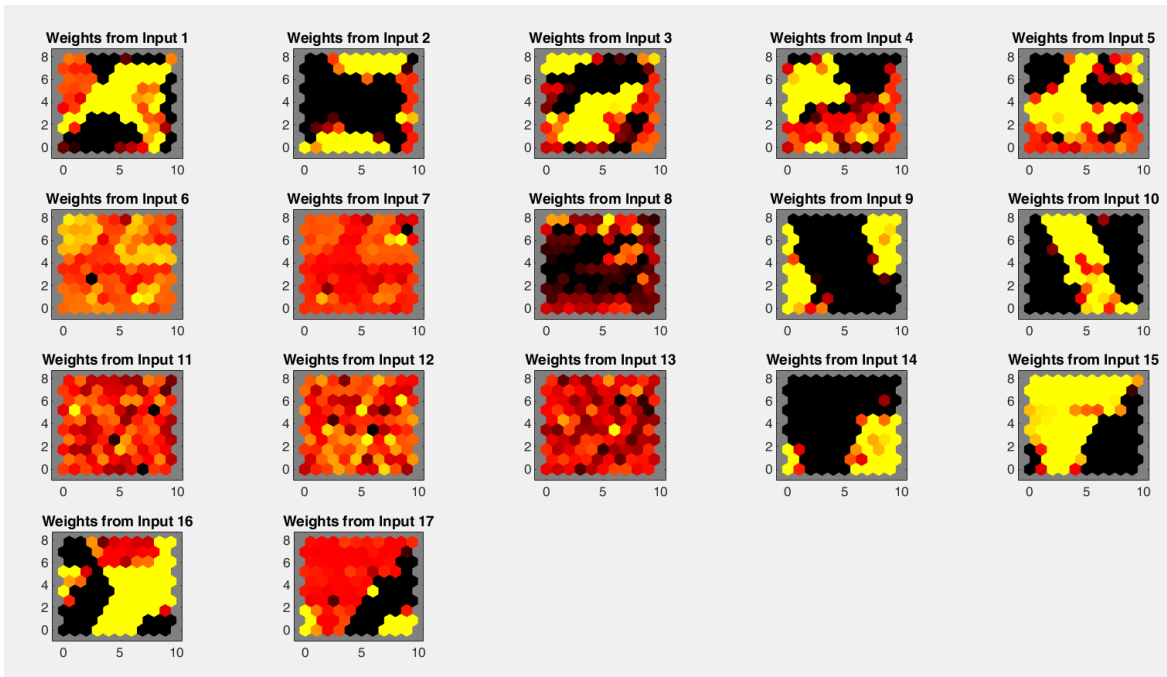


Figure A2.2.2: RNC Voice Traffic Input Variables Weight Planes

A2.3 RNC Data

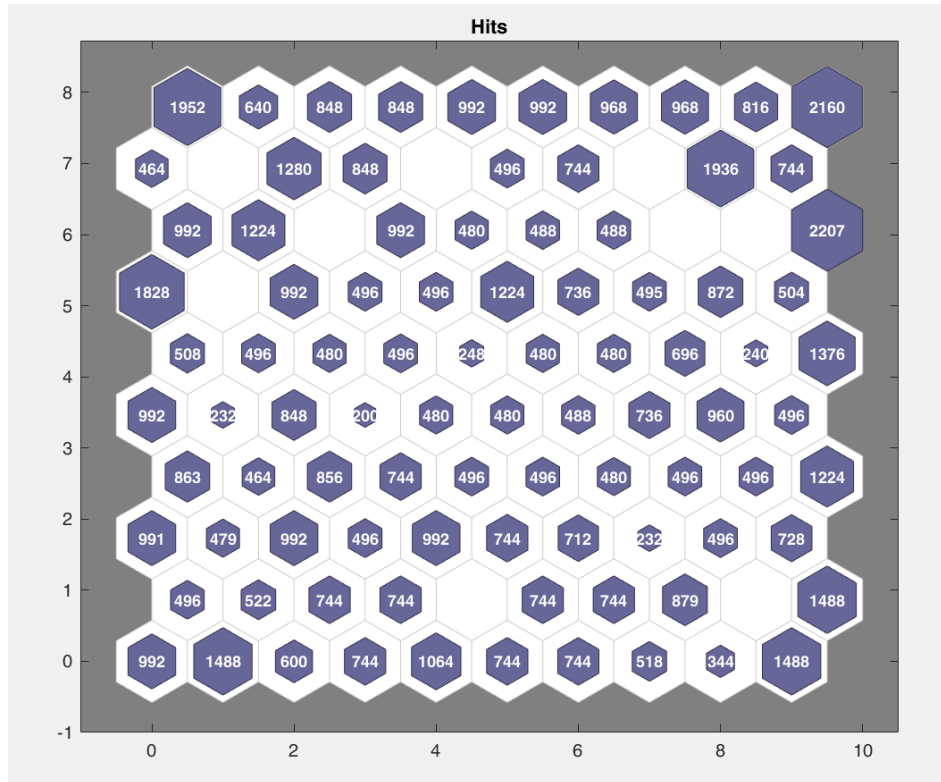


Figure A2.3.1: Sample Hits Plot for Input Variables for RNC Data Traffic

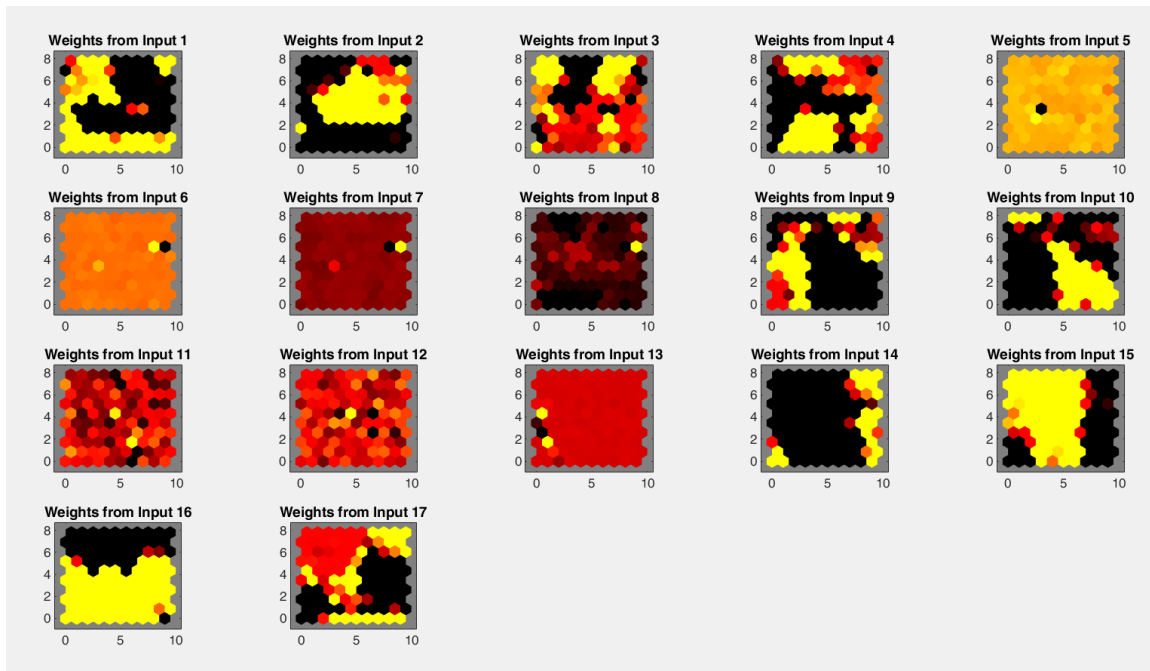


Figure A2.3.2: RNC Data Traffic Input Variables Weight Planes

A2.4 LTE Data

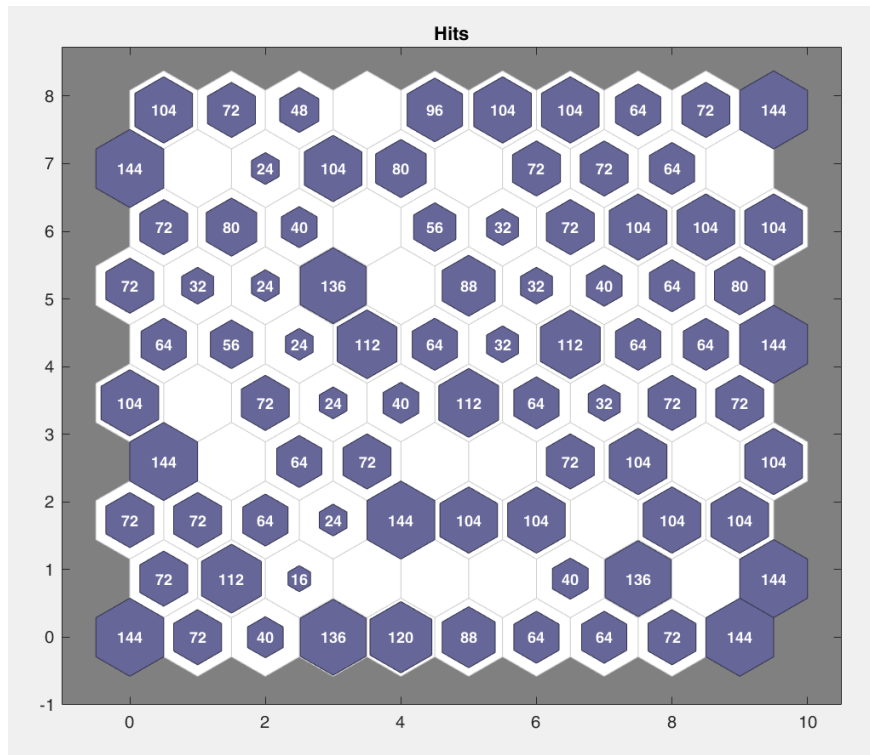


Figure A2.4.1: Sample Hits Plot for Input Variables for LTE Data Traffic

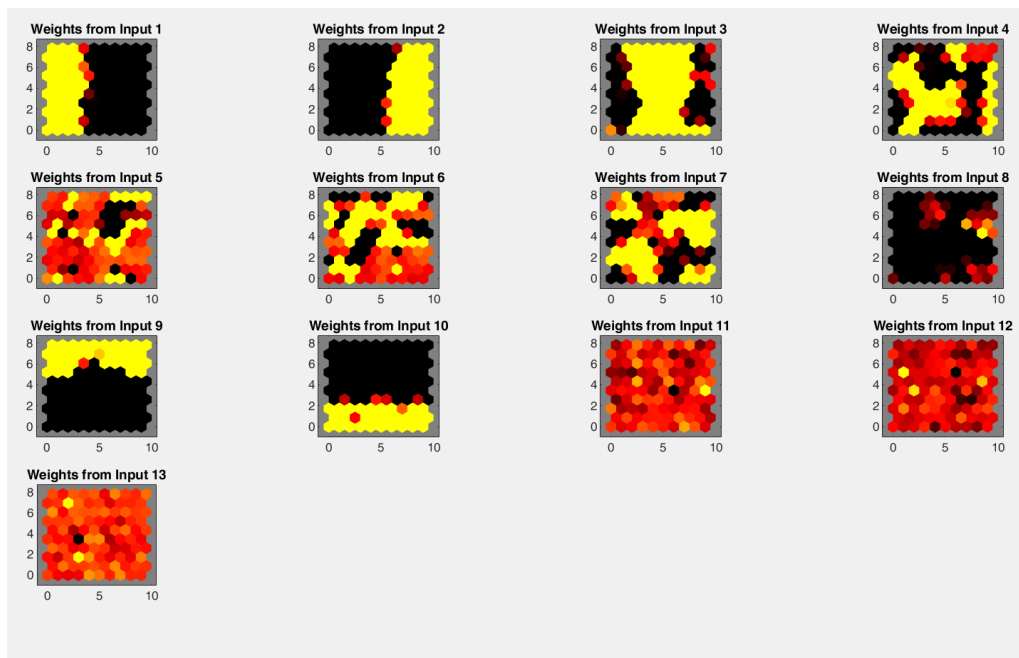


Figure A2.4.2: LTE Data Traffic Input Variables Weight Planes

A3 PERFORMANCE RESULTS FOR BEST SIMULATION RUNS

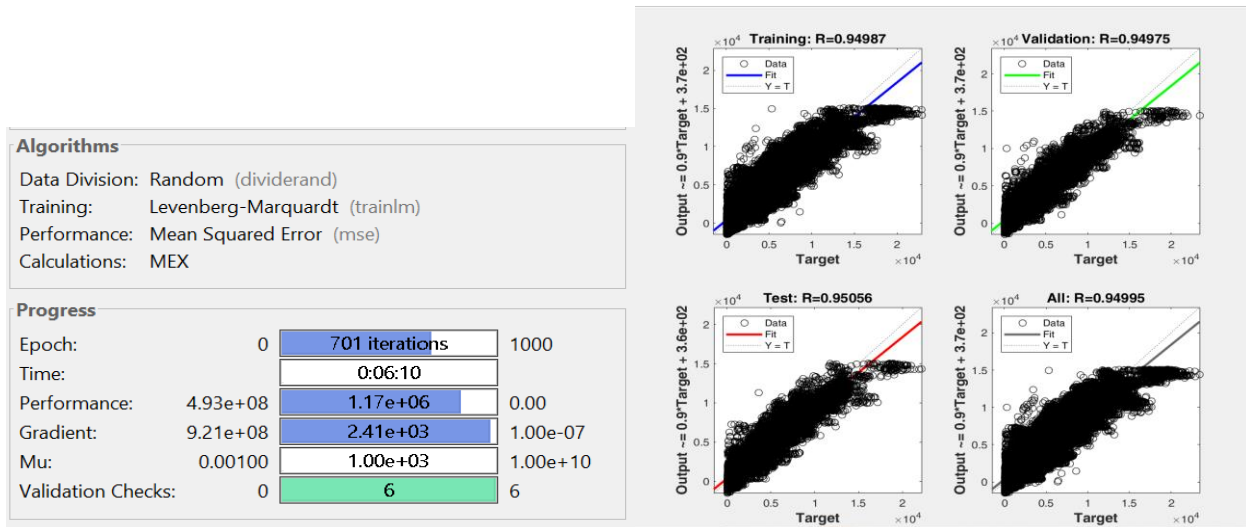


Figure A3.1: BSC voice traffic training performance results

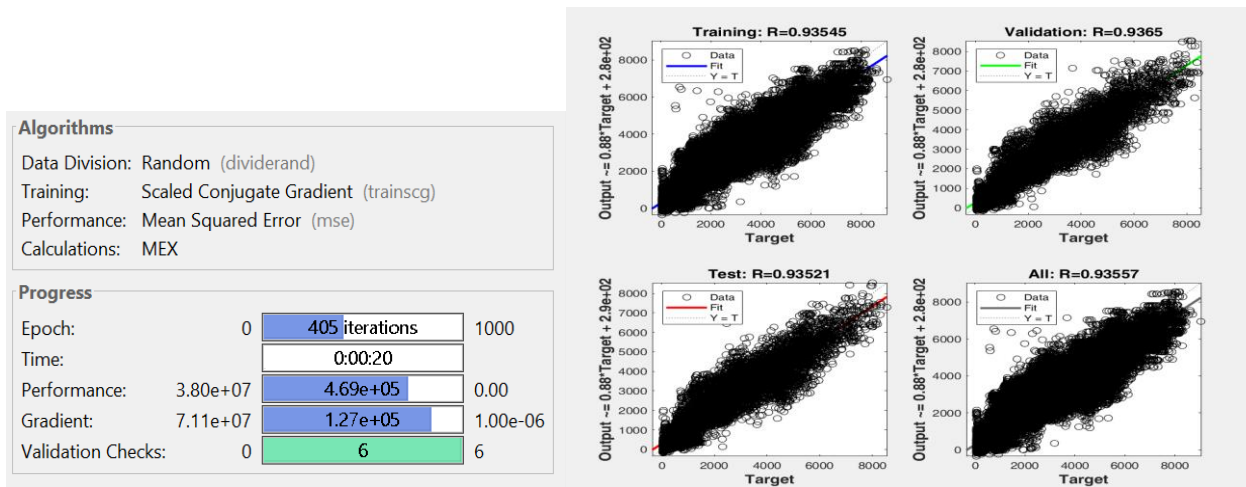


Figure A3.2: RNC voice traffic Network G performance results

Figure A3.1 to A3.4 show a snapshot of the optimal performance results as obtained from the MATLAB simulations. A summary of the same is captured in Table 6.1 to 6.4.

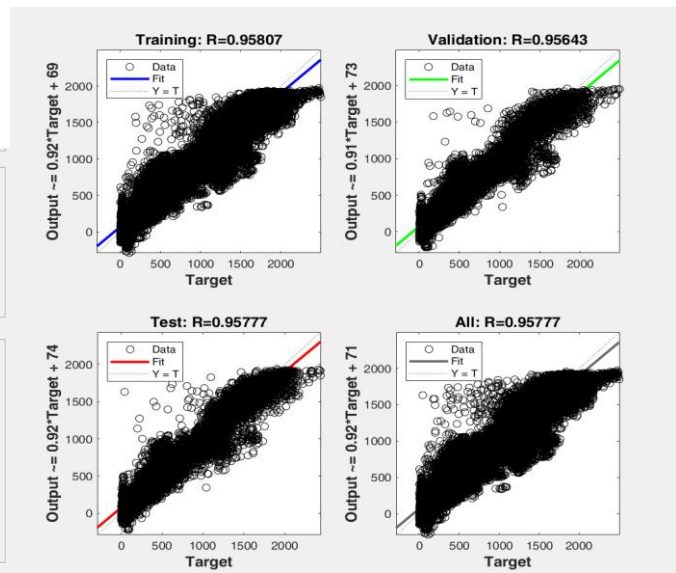
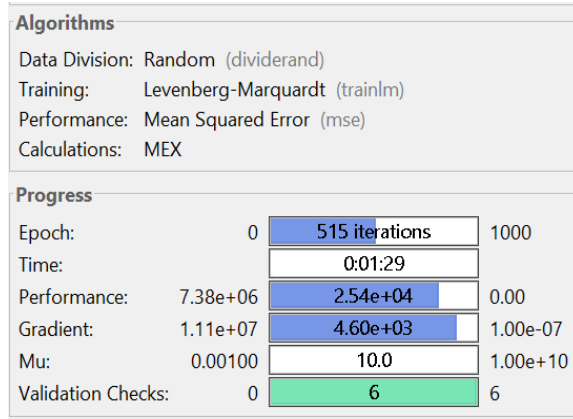


Figure A3.3: RNC data traffic Network K performance results

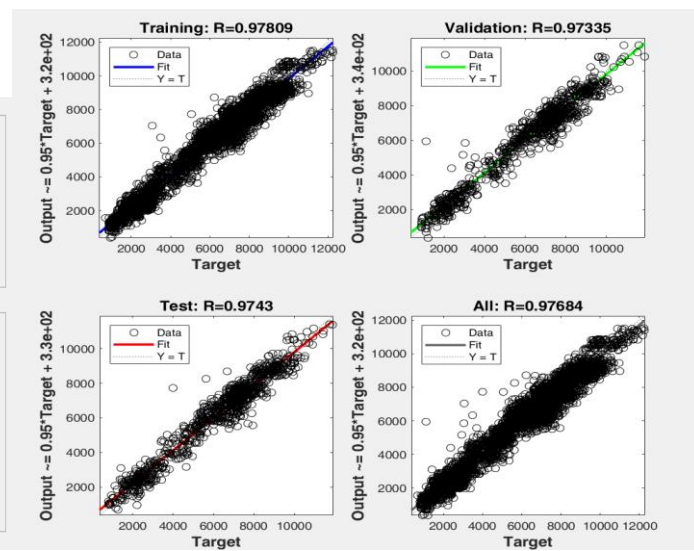
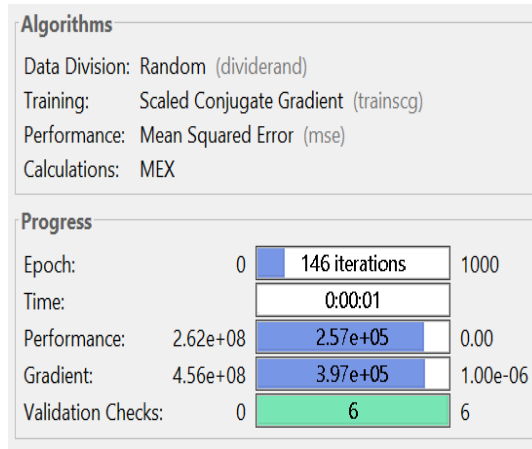


Figure A3.4: LTE data traffic Network S performance results

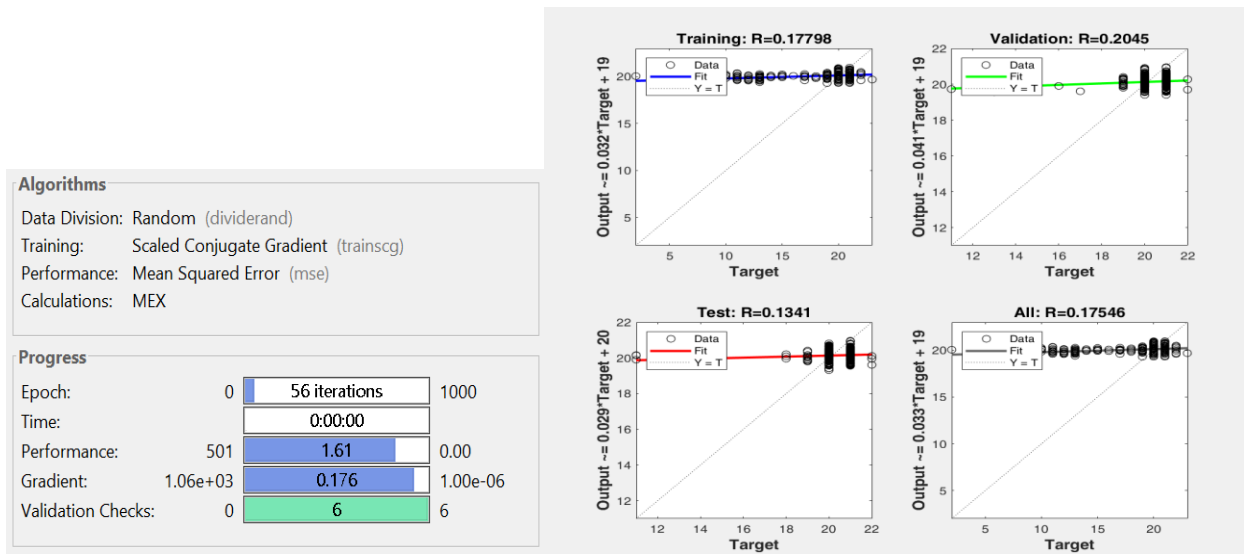


Figure A3.5: BSC Voice BBH SCG trained (20 Neurons) performance results

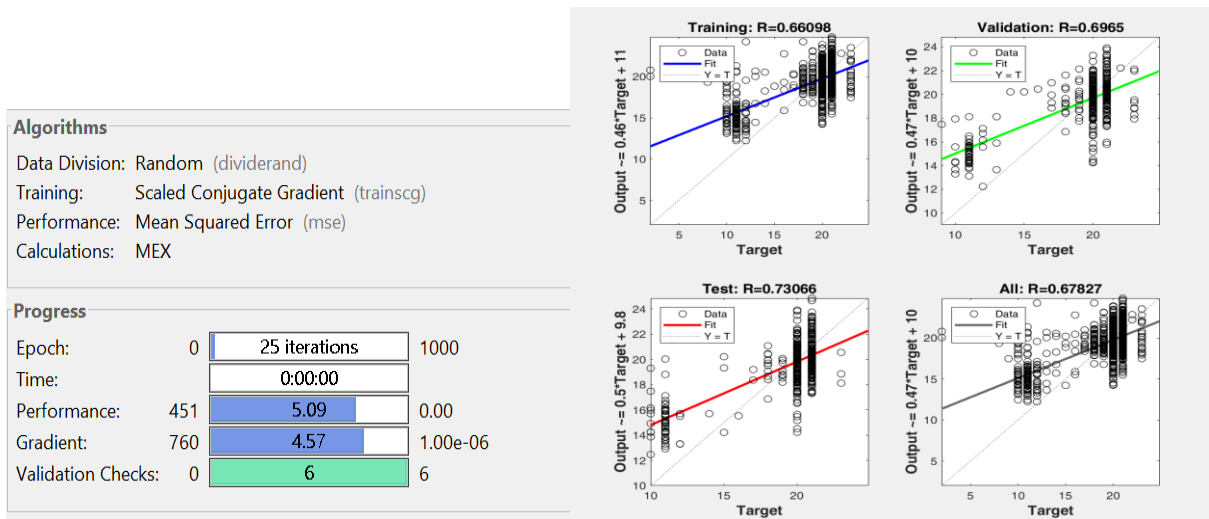


Figure A3.6: RNC Voice BBH SCG trained (30 Neurons) performance results

Figure A3.5 to A3.8 show a snapshot of the optimal performance results as obtained from the MATLAB simulations. A summary of the same is captured in Table 6.12 to 6.15.

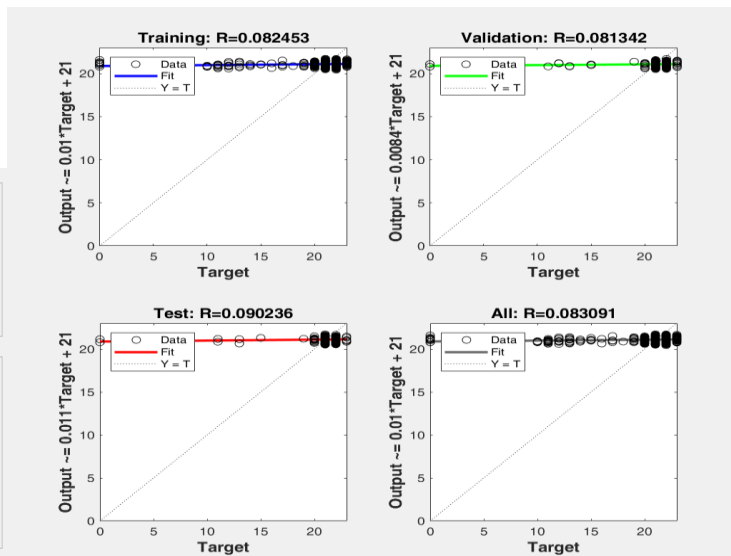
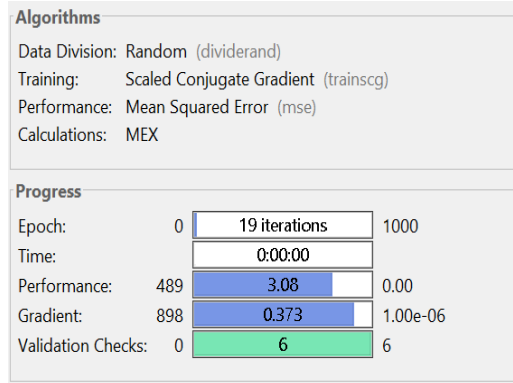


Figure A3.7: RNC Data BBH SCG trained (10 Neurons) performance results

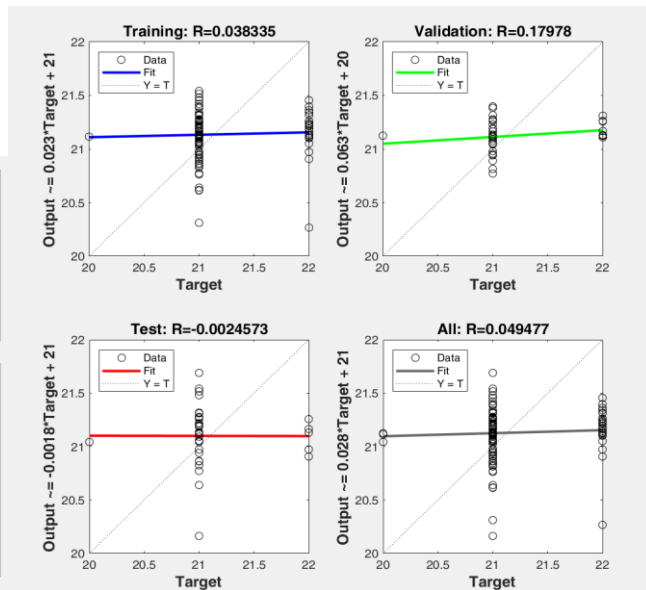
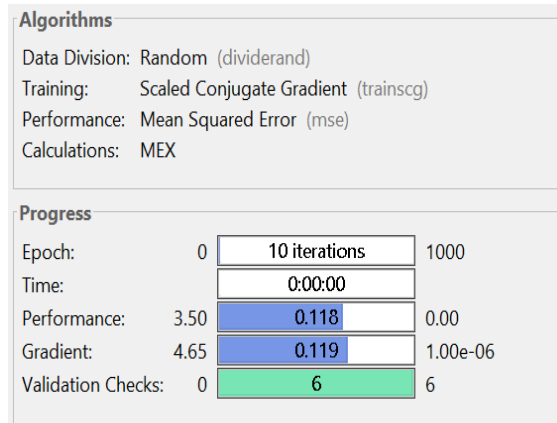


Figure A3.8: LTE Data BBH SCG trained (10 Neurons) performance results

A4 PROGRAM LISTING

A4.1 Cellular Traffic Forecasting

A4.1.1 BSC Voice (LM Algorithm, 10 Hidden Layer Neurons) Script

```
function [Y,Xf,Af] = myNeuralNetworkFunction(X,~,~)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Generated by Neural Network Toolbox function genFunction, 04-May-2021 12:37:54.
%
% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:
%
% X = 1xTS cell, 1 inputs over TS timesteps
% Each X{1,ts} = Qx18 matrix, input #1 at timestep ts.
%
% and returns:
% Y = 1xTS cell of 1 outputs over TS timesteps.
% Each Y{1,ts} = Qx1 matrix, output #1 at timestep ts.
%
% where Q is number of samples (or series) and TS is the number of timesteps.

%#ok<*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
xl_step1.xoffset = [0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0];
xl_step1.gain = [2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2];
xl_step1.ymin = -1;

% Layer 1
b1 = [-1.2764382934686320237;-17.774302175477174615;-
2.0580067857312340784;2.9422321457463764816;2.2149674015187970078;6.4334212812265842985
;4.2057480502980144621;-
5.3902234236662183164;5.3184528175760030777;5.7552192487958651057];
IW1_1 = [0.0017014917743573331008 0.0021538349452580327417 7.4642027329476051917e-05 -
0.00044908877009884324953 -0.0034369803778680463845 -0.0036383911720904997707 -
0.00066271058828519188461 -0.0037380812196891512883 -1.9875110585067641278 -
1.9953795564322682754 0.015427883162162061431 0.012984587726920559028
0.006821515320304892202 0.086834075078332503828 0.042604002674872479273
0.082557285211570727301 0.078141448984457573257 -
0.0013757851568895072738;0.53683029586364017227 -0.014631532381605969959
0.10406992147723077846 0.17150278252204406715 -0.048929235120565926775 -
0.021322273496277886068 -0.046161376737821155358 0.19496790667490568327
12.46671324838867001 3.8748157837795478287 10.905825485317684809 -8.1980556168762692693
0.32311221318691424331 6.241611508432382216 -5.348151949688420892 10.298482803530232488
-18.586611465124171616 -13.07519611195802689;0.00036539147780787211046
0.0079694278415069885052 0.0010850531773203373354 0.0012109981458369690425 -
0.0034559805479776090249 0.00087855603538973341647 0.0044644197688713102221 -
0.013342962842604914539 1.0744257880743823819 0.30529338465920707879
0.81534622271439294305 0.29518489711996392932 0.12702740821358879386
0.04351086184439742488 -0.0016522539206111196836 0.021189426623919462023
0.093536994072166451675 -0.11584331714663047286;0.035204568336705217846
0.033424145221601017641 0.033463426638006062153 -0.0045302199027989079397 -
0.0087407138185738638553 -0.0081046145049968928292 0.00066546500050339654504 -
0.0011230979797206414039 2.197002638314172529 2.2842633597242651966 -
0.078665185656198649555 -0.0012398193013336668636 -0.011019005238961876447 -
1.6241181007914573531 -1.0095341717028338824 -1.4288406269675153482
4.6221838142726037191 -0.26802642469475923548;-0.0026782602805556347293 -
```

```

0.008886803579665685654 -0.0006160907882273558003 -0.0014950010925680841996
0.0037722144699116593597 -0.00015091534446889503249 -0.0035274575591210037377
0.010377449462774534367 2.77996873110718834 3.4848933471733474221 -
0.86242919025923625576 -0.30380062049226713716 -0.12924079180227396324 -
0.035635103040003196506 0.0013456884139533623057 -0.01887669125490890748 -
0.087922922460178187798 0.097613001412776442822;0.18910068823929443438 -
0.11424416874963459756 0.029704894268509293709 -0.018386880111536461901
0.053532082602252395453 0.047984901729935490466 0.036109304944796700998
0.021836001031347251899 1.4871474631231498265 1.0295307668878199703
0.34597649518225709331 0.15718851794726540483 0.075079884645399386223
3.2889065570805384731 -5.0185444297024472959 1.4290442876764601898
7.8956390597864691827 -7.5486426926711231999;0.018527118903511632236 -
0.0027906099356373552378 -0.044561910364454629385 -0.017939642417732250268
0.014390483042103136002 0.014263706295527046539 0.036049765940181964508
0.014348419385174544446 2.2009843057761253604 0.8762760485220887352
0.77905966846496710332 0.77337412445134656735 0.43478654246353448665
5.0166272140305103733 10.781976905707759684 5.3138330009333269999 -
3.3512402055896317599 10.719130338866847296;-10.204947723113289015 -
5.246950277955008346 1.6023668616255191122 -7.003461398541179328 -8.5790878530150322945
-8.4869055311451031542 0.014258456254262367319 0.0748512238979706368 -
8.3308797051933130007 -16.926554252149923485 0.06359963440281597713
0.022330626935513984976 8.5922529154059912315 6.3120038378417104852
8.9731658607543920425 1.2138081241536113897 -9.7817133195512528943
8.9849655053318802089;-0.24086832380883582694 0.14363538670760331306
0.1056989946338638825 0.079567513580289739306 0.0022808418864926783415 -
0.00025215681961784138648 -0.014565587851691801785 0.020795659013317788905
1.4274521799412567269 1.8351072881008219095 -0.21277053150425689898 -
0.20634356878258025558 -0.15110963516078199098 -0.49952681676725307902 -
1.4592053134952205351 -0.93019180317378757916 -8.3296363856318862418 -
1.6583471806167158213;-0.071838020845830963257 0.10280350661553171177 -
0.014639304054397809726 -0.009686400549237056401 -0.028530297964709121217 -
0.029267880351509168585 0.015164430272550539006 -0.012110089888442875788
3.8410634656682409727 0.42765823144188419569 1.6675794539182833276
1.6450375039335756178 1.2549657177552071552 4.1998694583694309301 -
0.065807264275128105058 4.4145964365493304271 -4.2213786967738959888 -
0.080818339453036749931];

```

```
% Layer 2
```

```

b2 = 5.7588465608022110231;
LW2_1 = [9.6630668896769780218 0.03047779594867823863 7.2192518930103828367
0.3374703021852111795 9.1107574233020720555 -0.16725294249319669349 -
0.1514379059265757721 -0.013111822468326239308 0.21475681788895942259
0.11321295389560444289];

```

```
% Output 1
```

```

y1_step1.ymin = -1;
y1_step1.gain = 8.55622660139407e-05;
y1_step1.xoffset = 50;

```

```
% ===== SIMULATION =====
```

```
% Format Input Arguments
```

```

isCellX = iscell(X);
if ~isCellX
    X = {X};
end

```

```
% Dimensions
```

```

TS = size(X,2); % timesteps
if ~isempty(X)
    Q = size(X{1},1); % samples/series

```

```

else
    Q = 0;
end

% Allocate Outputs
Y = cell(1,TS);

% Time loop
for ts=1:TS

    % Input 1
    X{1,ts} = X{1,ts}';
    Xp1 = mapminmax_apply(X{1,ts},x1_step1);

    % Layer 1
    a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*Xp1);

    % Layer 2
    a2 = repmat(b2,1,Q) + LW2_1*a1;

    % Output 1
    Y{1,ts} = mapminmax_reverse(a2,y1_step1);
    Y{1,ts} = Y{1,ts}';
end

% Final Delay States
Xf = cell(1,0);
Af = cell(2,0);

% Format Output Arguments
if ~isCellX
    Y = cell2mat(Y);
end
end

% ===== MODULE FUNCTIONS =====

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);
y = bsxfun(@plus,y,settings.ymin);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings)
x = bsxfun(@minus,y,settings.ymin);
x = bsxfun(@rdivide,x,settings.gain);
x = bsxfun(@plus,x,settings.xoffset);
end

```

A4.1.2 RNC Voice (SCG Algorithm, 10 Hidden Layer Neurons) Script

```
function [Y,Xf,Af] = myNeuralNetworkFunction(X,~,~)
```



```

%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Generated by Neural Network Toolbox function genFunction, 04-May-2021 14:07:29.
%
% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:
%
% X = 1xTS cell, 1 inputs over TS timesteps
% Each X{1,ts} = Qx17 matrix, input #1 at timestep ts.
%
% and returns:
% Y = 1xTS cell of 1 outputs over TS timesteps.
% Each Y{1,ts} = Qx1 matrix, output #1 at timestep ts.
%
% where Q is number of samples (or series) and TS is the number of timesteps.

%#ok<*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
xl_step1.xoffset = [0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0];
xl_step1.gain = [2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2];
xl_step1.ymin = -1;

% Layer 1
b1 = [1.5150118181666285544;-
3.5192719929615243757;2.087642133831082436;1.6100078859849642487;-
0.11926541570825524574;0.2585315844836101884;0.36450836217800963723;-
4.3559994860700443908;-0.78524005841525035265;-1.1333392966965423199];
IW1_1 = [-0.25544943288653437774 -0.23441461121138329471 -0.00038512434248238515151 -
0.19163755984994573778 -0.18025861053014280433 0.29896124443346511201
0.4225983771541256262 0.30915631861349146536 1.1024778854380761217 -
0.1749345402760091317 -0.6878185502315060651 0.16534597416938479064 -
0.30978815091309602758 -0.3097219643285043511 0.62232828831591813223 -
0.31132451357724394425 -0.60936799776836647613;1.3488123601656640815 -
1.1953025903860621248 -0.23547835965465477304 -0.157823746118212066 -
0.032746352306673040533 -0.048414489057190292221 -0.022921265039214728382 -
0.22056555852931003514 1.0349200505819426965 1.005806908602132177 -
0.049919379777817712851 -0.067122247813543506334 -0.024597880228451232676
0.53262182866557938876 0.71318683778499547543 0.071980329064474907197 -
0.0022099495250095858336;0.56604988680988443939 0.20151158441862054849
0.055156360327085748874 0.10446038350524497473 0.049378187813921836646
0.069293739308049898118 0.046264637855059145832 -0.047053010249260611431 -
1.6035290943612285908 -1.5127976084220513542 0.14158791613222895811
0.22062694085360104457 0.10019432129095857953 -1.3274012552576848201
0.33634893276107741267 0.79490096636986073975
2.1257288271794698709;0.020127800364637737918 0.22480582746305430297
0.0052785267051370559147 -0.0098348182227753055268 -0.31829718771265885779 -
0.337664257058866209839 -0.25417260756366855112 -1.2342134731741287723
3.1373243232589516083 2.5691279652196015171 1.7675137940297611649 1.8277131281211294045
1.0805927632395675442 3.2823941379189798262 3.2798363216833061884 -
0.076414778046131309752 0.16009171122737519966;0.53512765794277472597 -
1.0707115762679055848 -0.25584136043115240922 -0.45344428171170975128
0.12027716591436468652 0.043148361585510344729 -0.087939867919356179415
0.19672144762861928924 -1.0218810711808119507 -0.59759349013675966411 -
0.58871400893870451387 0.49679547944403712467 -0.034296834669847911514
0.48889572393909214165 -0.58636990785725762176 -0.31439875581336573784 -
0.46856626536098522351;0.045670643094314095078 -0.10204733904535706612
0.092010826015892555052 -0.028692075636958743606 0.26756060912249313155
0.2953685145444583382 0.21763287104647738301 0.18433009444373615882 -
0.42810314854660130868 -0.0016699981439251823309 -0.58569678581568307418 -

```

```

0.72403100865755820603 -0.17611311969443199654 0.028689091959147164368
0.32832822422826385411 1.0834022591453640327 -1.078028692672237554;-
0.087318926307459285296 0.016172436014927314696 -0.052274862692227745675 -
0.023044005795686120508 -0.30788959040273689016 -0.3096635338398605608 -
0.14239725832444105613 -0.73301774171767608568 2.0114936702644929944
0.7468195007975301225 1.2885210639972417557 1.4498155554995595384
0.14674735460896418116 0.17687460143072517016 0.057186726285563427297 -
3.4057799690563714812 3.7984525765556806931;-0.16675479618901026257
0.04244945807844799357 0.094373060169401196129 0.02091791505561819281 -
0.20632698860334489077 -0.18735018599144662166 -0.092150760347160451391 -
0.17487910181468033222 1.0014760410036593896 1.4911927010392316273 -
0.28838861704378526696 -0.27028541837070668707 -0.1177820371993459625
0.67256388984075232695 0.997239267669645435 2.0490974969227084834
2.3877662089540963031;-0.51788066221336581663 0.91177885894468480021
0.451192929526887887 0.34006132430870600336 -0.038046741554673975394 -
0.014237262488254038423 -0.0046833599797337626541 0.12636069021870544038
2.5880156728884662343 2.5004259057545188405 -0.17114120766060164636 -
0.21483525386176860472 -0.11503608994165769042 0.6524089481190383788
1.3548233601111581148 -0.83764393224398325799 -0.4808060140771346358;-
0.18750364535436811475 0.034754352747803383961 -0.25348487364524530108
0.18849899342742118535 0.40568274367887990906 0.33446131760628994289
0.25501223667289130148 -0.61405236811384844575 -0.31269292536486359424
0.58555501276006427513 -0.2243513696254704759 -0.028929264856925152039 -
0.24983403915516272376 1.0660729993283599715 -0.28273424956598069979 -
1.3794930045034710098 1.9397726463253492923];

% Layer 2
b2 = 0.79623167579099718427;
LW2_1 = [0.0019216448948164461131 0.6708480486067948112 -0.31051067688016981139
0.14046174076988776624 -0.0036367988245196527543 0.13776254386076852709
0.17073246656964649604 0.41882522336712613154 0.23607264746161690794 -
0.080796736876508748471];

% Output 1
y1_step1.ymin = -1;
y1_step1.gain = 0.000222560534004334;
y1_step1.xoffset = 50.013041171;

% ===== SIMULATION =====

% Format Input Arguments
isCellX = iscell(X);
if ~isCellX
    X = {X};
end

% Dimensions
TS = size(X,2); % timesteps
if ~isempty(X)
    Q = size(X{1},1); % samples/series
else
    Q = 0;
end

% Allocate Outputs
Y = cell(1,TS);

% Time loop
for ts=1:TS

    % Input 1

```

```

X{1,ts} = X{1,ts}';
Xp1 = mapminmax_apply(X{1,ts},x1_step1);

% Layer 1
a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*Xp1);

% Layer 2
a2 = repmat(b2,1,Q) + LW2_1*a1;

% Output 1
Y{1,ts} = mapminmax_reverse(a2,y1_step1);
Y{1,ts} = Y{1,ts}';
end

% Final Delay States
Xf = cell(1,0);
Af = cell(2,0);

% Format Output Arguments
if ~isCellX
    Y = cell2mat(Y);
end
end

% ===== MODULE FUNCTIONS =====

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);
y = bsxfun(@plus,y,settings.ymin);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings)
x = bsxfun(@minus,y,settings.ymin);
x = bsxfun(@rdivide,x,settings.gain);
x = bsxfun(@plus,x,settings.xoffset);
end

```

A4.1.3 RNC Data (LM Algorithm, 10 Hidden Layer Neurons) Script

```

function [Y,Xf,Af] = myNeuralNetworkFunction(X,~,~)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Generated by Neural Network Toolbox function genFunction, 04-May-2021 15:38:07.
%
% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:
%
% X = 1xTS cell, 1 inputs over TS timesteps
% Each X{1,ts} = Qx17 matrix, input #1 at timestep ts.
%
% and returns:
% Y = 1xTS cell of 1 outputs over TS timesteps.

```

```

% Each Y{l,ts) = Qx1 matrix, output #1 at timestep ts.
%
% where Q is number of samples (or series) and TS is the number of timesteps.

%#ok<*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
xl_step1.xoffset = [0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0];
xl_step1.gain = [2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2];
xl_step1.ymin = -1;

% Layer 1
b1 =
[0.0056935764533825160977;25.421713210193892962;5.4997375585570411616;75.04606759563475
2934;-8.6217675523331500642;-1.9357265499626143956;-7.6192407465626770247;-
0.9127433809717876434;-0.0056095949189850554534;4.3153335594914183915];
IW1_1 = [0.0011318476714934854431 0.0052330186427958034351 -0.00029607637478871773632
0.00022136548944004381099 0.00015183716882228629982 0.00029187117845678968095
5.1896815546951490232e-05 -0.00023840808231682182779 -0.010204635253298437875 -
0.0078837913272263217962 0.11643606143358781191 -0.012346457417190493874 -
0.0030117513222154795698 0.076531953065621299848 0.073573131765689123163
0.091220165486575674474 0.0034043447451994799345;-2.8719844278494894851
6.4525614378514903891 7.735508772757178399 2.6284413341603296566
0.094629268932834034733 -0.082092657895000475943 0.271152650252955707
0.027055857753592121862 4.6295015278490909694 9.8596038356411419556 -
0.05645151617048337761 -0.12677804159355041125 0.19469833259312038787
10.01528487874609219 -4.3672189449177096776 13.465838038765340912
4.9903838996101912429;0.075584440863537993849 -0.062589937178449600585
0.1062932583978902501 -0.010794982261374980528 0.046835018023735588977
0.022515742544358358784 0.041794877702713228995 0.12158883021005671699
10.093930632421711735 10.137504656235684308 5.1666278283510589375 5.1712045131161623601
5.2001695944439187969 7.1200888383839773255 12.242187963884838098 1.647103238466476105
-3.6757071457856271834;0.13085897658683390277 0.88148097810537007213
1.3067962782237600994 1.0873105135258338549 0.018795941988058170041 -
0.021809900244183777485 -0.065051258687271781844 17.768163546614960779
48.57752829804240946 32.016127937909544698 15.000145538475210927 14.433299530724209347
19.390830361857009478 66.758450563910074038 61.578616300386009641 -
8.5777596572129066743 -49.077734664601109671;0.22760629203702595347 -
14.08705300517187986 -7.1486594659068698476 -6.9052270379031321923 -
0.13798700415924591844 0.0047248041932220520567 -0.023193011241507580783 -
0.13910199537167347583 -2.7737383261094996634 -1.5735060737650057305 -
0.53156285869563291158 0.16089836360808945637 0.21586102865408304807 -
21.998355689584418116 -10.873536069170729235 10.54615690094462721
3.6639479204065552587;0.056626364136468432253 0.026316815779299326805
0.025349720260114078779 0.023200729671636170087 -0.0067649954387801721281 -
0.0034824389202587424999 -0.00042633426493973573292 -0.014191921497111369133 -
2.1234869314445163901 -2.1010022330747553099 -0.011261281664594292515
0.0027053880365397343227 0.012282237225291077129 -4.5559850092705254099 -
4.7135433359238687601 -3.4683940776833086517 0.19326575365105772009;-
13.442773770623810847 -0.051473518952233399126 0.25112529292912566525
0.023898942218216534528 0.050247356699869495655 13.649018792653313525
0.011959868063081904518 0.0780322229199455486 4.7590774216310736122 -
1.6757692161364712291 -5.3204221281168724289 3.1185882092272905197 -
11.504054730657058414 15.68103742907777645 2.1354803309276131351 -
15.095182858343397569 -1.055518755420385535;-4.0715864844407576584 -
5.1034885367999809347 0.45308664119811731252 0.2820448823044878206 -
0.0024523936574709446351 0.00018187444254580143735 -0.0032266559249459099423
0.088178240313671815698 0.20584784886010862315 0.20920733919446968252
0.019299896002979915338 0.020612404260062066469 0.024412016756439482157 -

```

```

0.23272402667622113515 -0.35608822759789798829 -1.3335675839562672884 -
0.86933691688909664474;-0.00033620494918532348822 0.0056241010156397337055 -
0.00096874135232637170485 0.00082952458920745441445 0.0058364867168828301128
0.003260138966925264347 0.0015223677465235089731 0.0027487135943651602385 -
0.11692853058691107804 -0.098443976243061143139 1.0287744684171862275 -
0.19898464265858931133 -0.047658782525303429456 0.061088831745246124916
0.065055042118730393996 0.039813680006409148115
0.0034182523954572607838;0.74157165510574285516 -2.617306909761622169 -
0.16130907790306811544 -0.16052181459360925775 -0.0073665581999295560306 -
0.0089164023379311573708 0.0026233881412554044439 0.029530312696045016296 -
0.038774272609115624511 0.01750854148593923576 0.0041458129520467266321
0.0019041066460859078918 0.0037315474091305640518 -0.07860222456044940742
4.0426630064241066975 -8.5136173428515320438 -3.9896908210615036339];

% Layer 2
b2 = 0.072062505845864238685;
LW2_1 = [-32.665353725739812774 -0.029220649931067120553 0.10384385932160006272 -
0.049736521568582735264 -0.065743429821652141221 -2.7767963825702790537
0.027637022281575336635 -0.18389256209477022885 4.9834740002870319131 -
0.44341981565920896013];

% Output 1
y1_step1.ymin = -1;
y1_step1.gain = 0.000801191198990983;
y1_step1.xoffset = 1.39657649025321;

% ===== SIMULATION =====

% Format Input Arguments
isCellX = iscell(X);
if ~isCellX
    X = {X};
end

% Dimensions
TS = size(X,2); % timesteps
if ~isempty(X)
    Q = size(X{1},1); % samples/series
else
    Q = 0;
end

% Allocate Outputs
Y = cell(1,TS);

% Time loop
for ts=1:TS

    % Input 1
    X{1,ts} = X{1,ts}';
    Xp1 = mapminmax_apply(X{1,ts},x1_step1);

    % Layer 1
    a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*Xp1);

    % Layer 2
    a2 = repmat(b2,1,Q) + LW2_1*a1;

    % Output 1
    Y{1,ts} = mapminmax_reverse(a2,y1_step1);
    Y{1,ts} = Y{1,ts}';

```

```

end

% Final Delay States
Xf = cell(1,0);
Af = cell(2,0);

% Format Output Arguments
if ~isCellX
    Y = cell2mat(Y);
end
end

% ===== MODULE FUNCTIONS =====

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);
y = bsxfun(@plus,y,settings.ymin);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings)
x = bsxfun(@minus,y,settings.ymin);
x = bsxfun(@rdivide,x,settings.gain);
x = bsxfun(@plus,x,settings.xoffset);
end

```

A4.1.4 LTE Data (SCG Algorithm, 40 Hidden Layer Neurons) Script

```

function [Y,Xf,Af] = myNeuralNetworkFunction(X,~,~)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Generated by Neural Network Toolbox function genFunction, 04-May-2021 19:25:39.
%
% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:
%
% X = 1xTS cell, 1 inputs over TS timesteps
% Each X{1,ts} = Qx13 matrix, input #1 at timestep ts.
%
% and returns:
% Y = 1xTS cell of 1 outputs over TS timesteps.
% Each Y{1,ts} = Qx1 matrix, output #1 at timestep ts.
%
% where Q is number of samples (or series) and TS is the number of timesteps.

%#ok<*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
x1_step1.xoffset = [0;0;0;0;0;0;0;0;0;0;0;0;0];
x1_step1.gain = [2;2;2;2;2;2;2;2;2;2;2;2;2];

```

```
x1_step1.ymin = -1;
```

```
% Layer 1
```

```
b1 = [1.7769345068624589068;2.0300068678461906657;-  
1.7937796495891151771;1.6757982200682079998;-  
1.4591618409584594662;1.3702944643444856521;0.95440127530179574844;-  
1.3092414741421756208;0.67378551682963672143;-1.0585968230327853501;-  
0.77089268838680424789;-0.6278478290695499453;-0.70894920095099678292;-  
0.75125003151791303058;-  
0.23242009736930860364;0.64304332929943797126;0.43155643050241232483;0.6700217968445525  
4183;0.023934268656497621752;0.17674801870432657602;0.32922336653220585223;0.0579159466  
58811785774;0.34743487956385116222;0.32874205115729243643;0.43336143002123023171;0.6016  
4307963371188581;-0.62030155599794711829;-  
0.71926585289748379282;0.75141148314516625906;-  
1.0597965148128301038;1.0921179446604141194;1.0764781119783624064;-  
1.2160969547136943891;1.2724007513552653137;1.5198783349626956429;-  
1.3956665034699784478;-1.5420726166452229933;-  
1.7911330257746849171;1.777117991111398343;1.8144882003773594725];  
Iw1_1 = [-1.0187176352551758907 -0.22908757112125857636 -0.33653613152579503476  
0.038142225819071770565 -0.33030682376603598405 -0.11581723516496715554 -  
0.50956759008925134324 0.58037536556707292057 -0.99893159317603841263 -  
0.46912371048848400257 -0.011045866053684062075 -0.15150718812811600489 -  
0.38284867933999028322;-0.052575531656011524617 -0.18653745352622691511 -  
0.63998227412904273859 0.55839059802536794574 0.63610189812343531646  
0.39145638100915658431 0.060716868116689144685 -0.48386650610107356796 -  
0.48408371844413722318 -0.7229722374320975975 0.40574517352651218172  
0.0010353126407589615526 0.23151139150202815209;0.3226814040194386779 -  
0.34379218632751157303 -0.50472496416740264547 -0.0067466834267952430909  
0.089605906817161490308 0.16421191917422708895 0.56338528991844472049  
0.29065287555933705388 1.1159205521752824097 -0.47645580877665894004  
0.29968933255621293954 -0.27418904765834817283 -0.54210321895640600776;-  
0.045956429772212321894 -0.66285982627652151322 -0.66010486213185104454  
0.19607862068304898751 0.6200943983074161947 -0.55888224410180098456  
0.38844390521984173414 0.17624305912001245189 -0.58682296245435017212  
0.20990508905943078277 -0.26593110457172081107 0.4980098208676024818 -  
0.50061489960582883363;-0.15745097395357346426 -0.46211517553560010896  
0.47485797214411834544 -0.61618644892870677587 0.05795710168953020075  
0.82844390910819332152 0.26269881344416934343 -0.78182733977418705784 -  
0.3860089454901844408 0.13445548215700556249 -0.52710068314460256644 -  
0.78309802875354317386 -0.3168543145551593887;-0.16277049368648699446 -  
0.097317423978861783418 -0.64224791771981526445 0.32288813583444425648  
0.60562138736208270817 -0.40696944886388664031 0.022707784684200902486  
0.12689396159249788365 -0.46630511502614196973 0.84842547242747590186  
0.56123627616768412274 0.1477786486767551366 -0.97055142721460019306;-  
0.054941525797094749317 0.81655612238569541539 0.29307013972928164236  
0.32332616647934436793 -0.56841977022465894898 -0.061344535224081113423 -  
0.54881396069582999431 0.38025528600098112975 0.63483148619081275488  
1.4033356518694262149 -0.17908762604273170682 0.09983557482556150442 -  
0.07120723555650457226;0.93173538921111831446 0.05960208218231295707  
0.44785942084996632762 0.15862795289760839301 -0.50237420777354468981 -  
0.14332303687029063211 -0.37379119229213764175 0.91631703113634055402  
0.25102980812936098731 -0.10871272625706036863 -0.41830111782240497753 -  
0.14632499585878980253 -0.3602495437723371996;-0.60096842023970897806 -  
0.57170821332047205843 -0.32360120405476555661 0.33369570265337988868  
0.3220230307974265016 0.044032538182417951811 0.032207509126013077194 -  
0.30719104526754975604 -0.78012031500899148284 -1.0352608344862834766 -  
0.54813918508369163796 -0.27324220477019550701 -0.17465738890958759644;-  
0.12121960409976791084 -0.11360430434069722361 0.055111855358720922282 -  
0.050743321866396573494 -0.090550791057735835476 -0.0075226043897679535991 -  
0.00070993230661525047465 0.082140776243466898099 -1.5183065626939078463 -  
0.93567263503128417224 0.99605458127873536256 0.79802226634074435729
```

0.59051451351459249839;0.37484103183680927973 -0.15214307485147401944 -
0.48474006848563344629 -0.12551632246466656873 -0.047898974611028807824 -
0.27915055572789237992 -0.080340065188366838189 -0.8953132856192008715 -
0.11459602162559161254 1.6215191810814657547 0.025103519290762640881 -
0.099987809363856733347 0.17600257563005602957;-0.077427834637959755404 -
0.069714978505972247103 -0.065089488530173697511 -0.046896733110483930507 -
0.098407646999807835142 -0.006323518825538646318 0.031115442035441955027 -
0.52880735524353761345 1.1403483352236725157 0.23705170029994745984 -
1.8126324156223858619 1.17976625451570305
0.53149035575573033707;0.031289152305627475348 -0.3247274256340700882 -
0.18774198689532675677 -0.73871910795519601933 -0.25099603070554132866
0.47292487653271375603 -0.12956014178897054534 -0.97003392810298005866 -
0.50942789682081157654 -0.73703883256818258918 -0.69756295735258988522 -
0.044579138661712737479 -0.24778907578824341251;0.6459891918105628017
0.34989129775299376846 0.33970088541966336759 0.10506146748870009988
0.63981406451104105848 0.11445367384409539657 -0.038836640970360387926
0.68269034744722401076 1.1770851824605910174 0.073189725550425785427 -
0.41587308433097069926 -0.41181804299336094433 -
0.32803125875715766213;0.5246024622969174267 0.32263434463410817221 -
0.41643131494693186312 -0.021367117711959811993 -0.085677597647302622397 -
0.38305077980470958465 -0.36636995758883422747 -0.61991916303956318135
0.66720446085691353932 0.81608680923501131232 -0.43673354888216980862 -
0.68608682690497002632 -0.68735394681026951158;-0.012752131799495839207 -
0.52239268627525337063 0.59067682407699417801 -0.6319552522118063731 -
0.37287026259555688945 0.59626380084462704545 -0.23203307687036575646
0.25870648367680781909 0.88446255034896648084 -0.34740491159787689757 -
0.3961631155838918894 0.036864902232288658535 0.40462135856963687974;-
0.64463470893011598495 0.50618337282119585385 -0.065568630802482291697 -
0.050888110019011487117 -0.2645907804787390738 0.73852432165473225201 -
0.47661950794891222438 0.066296512013745645397 0.19714977394234103891 -
0.97813720131261561175 0.22957332588821718899 0.38560349173527258548
0.74742986198896033123;-0.76778621685071335445 0.87220362173286170027
0.87082844133096992678 0.32684049549038596272 -0.010850377830622030817 -
0.14084939049132791378 -0.13731343311202137691 -0.96956926050649028248
0.94236226808385314957 0.66506596389258954005 0.036614193606473187381
0.27580963064428759557 0.087444939370833918435;-0.24950640873448695634
0.20454564698253660526 -0.15113231264172480306 0.098245254024894007605
0.59745574149066227321 -1.0647878813932369901 0.23244762989319317503
1.0673245644844058244 0.092728720170505846143 -0.32242106737546522011
0.23033131734348677444 -0.43149633042282370399
0.51815468027370348292;0.71450605508663256327 -0.017476887323763536791
0.41472147379525148292 0.134013472999158334 -0.5342641988582178536
0.021016995745662660211 -0.63249769878277961599 -0.51526685676498551469
0.81129952562813656414 0.27721265872786493567 -0.5636885223960924618 -
0.18918600315595124339 -0.52267188591928226415;-0.24784290582698098593
0.10853302561589367459 -0.12958949302660366376 -0.10890366995431562114
0.10965878725184864673 0.16766020524696037541 0.040303850809405621092 -
0.17922193994408955331 -0.21718329269832631345 -1.7207395494373032285 -
1.1849831720890295905 -1.1434306832594895997 -
0.69003416899254754746;0.36116848426678321848 0.47350513316090087068
0.13513285614953929326 0.72818751874139542313 0.67473585866170104897
0.58000777476773190955 0.61294706675057208933 0.66768369637498425906
0.50028086502854329787 0.34659673192027440347 0.12918155423132163251 -
0.61690129321166542464 0.022077683053164130073;0.21354203596758489225
0.15905406810761354985 -1.0564461476217816838 0.41897865434188369393 -
0.86512563177481593968 0.17901856928865325647 -0.67870630447801016949 -
0.16765732021079199354 0.037086468227292460065 -0.25974421185291085568
0.43629065123476667987 0.33277331116424235802 0.1496035762735569985;-
0.24898746269467492143 -0.80461841760414964941 -0.0080890082593530476807 -
0.097074706745615760295 0.1621275893886423014 -0.35714203277504524836
0.8000331426104222432 0.96913411232336554857 -0.59530676739779941897

0.014133362184587293375 0.49102684410630564882 -0.025810224175010023501
0.042026460293037172511;0.36523911244200646387 -0.036032300270747892601 -
0.4934420750221880092 0.097710299278526355216 0.036418118269106833362 -
0.98852482700793031789 0.39022374486157940243 -0.62846747691572779626
0.96228575027204787951 -0.48240343650376277251 -0.081629368549380451614 -
0.26768518416592590459 -0.45722442366286397686;0.5051556691866911919 -
0.13384402142381121581 -0.19704440322991378931 -0.096274596308986229443
0.44608908249181000727 0.42785102652224343833 -0.46882349925903843424
0.34057528663847685957 1.071424501096088111 0.7312164775972194164 -
0.29288431394325165069 -0.26590813958985853871 0.10670484973750024582;-
0.53697711413482718257 0.61616406839285031705 0.055622046693455731714 -
0.73599010496757921729 -0.55383283875394095919 0.020233387744441576339
0.55697870680630923168 0.51218738101062932788 -0.65653063206344131064 -
0.20591027051304477968 0.413343068224896526 0.7396107929476555265 -
0.19106118343018874128;-0.68232507548570464628 -0.17182175907218491528 -
0.35744604434032412055 0.053349602299798744298 -0.67110975705105802014 -
0.2758505280204727267 -0.51270647209361019936 0.47189002547293124046 -
0.94072313189422696045 0.12996615871779831908 0.24586575079632588814 -
0.29720349500086729044 -0.47261452018339733527;0.34600638179869558364 -
0.43472825299499240348 0.57332818989426803657 -0.22315192207122525048
0.79460188749973403866 0.048926614242495149176 0.3127213830351920909 -
0.14621507095049474101 -0.79068697544682908784 -0.1687441251361746497
0.6646615671669253711 0.51239100234050227733 0.77153943390398815882;-
0.30245046274242154416 -0.30353081671520459883 -0.53357140342378717168 -
0.39573799646451468526 0.50294627617294163091 0.71828848296861025347
0.40868169309781038345 0.94547782779754963034 -0.037203486462533817691
0.48805164156522545804 -0.1782249121995647112 0.49452850577388290931
0.19937814146034296692;0.27675474440125980014 0.82107475435143784104 -
0.35954373198034522696 0.86766795085953973832 -0.68481141980841586658 -
0.10329184008367033043 0.32541284966014555335 0.0072604028857458993707
0.20735250836339189084 0.21466595460115500682 -0.42973328636938318281
0.69254436436144295453 -0.16165312026754508068;0.50869720728525913422 -
0.38905572116441250463 -0.11626129666020081077 0.19854582998480879708
0.60055123601302340397 0.45830108329126284517 0.79755535780389574807
0.035254554615281478991 -1.1263406340375132775 0.17621461568799717812
0.18823060493073437427 0.13269724467967319859 -0.35076035543393235816;-
0.57260321586819362683 0.47059143854644486238 0.10280729203396199445
0.054343708757518684183 -0.82671980180057047516 0.68005610704941765743 -
0.75980665888586207402 -0.4074674607368053425 -0.42747359783779409481 -
0.48025851847399991801 -0.30879934760891508683 0.012926069231589055325 -
0.38299781646701724913;0.58318812127211405727 -0.49853971288046000643 -
0.61573361319217068122 0.27545246676823215148 -0.16753129966743565693 -
0.589923297748286668 0.16107867965144168076 0.39628126937705632216 -
0.58745890384196719491 0.25727129167859802417 0.8453181955630496569 -
0.374690393328592386 -0.72953450594687596631;0.58328855237045429405 -
0.62603162271786083615 -0.63702705083991784107 -0.41428605854033107025
0.10665621035237043968 -0.25312191974861419919 0.06602616046683743023 -
0.66618806195699464823 0.35718901038596473363 0.42162879461450203378 -
0.26188687676095900869 0.54769720427104773286 -0.057716067531464104734;-
0.2928592354642772233 0.089785855664017186206 -0.56654844743979770882
0.96038636425561119125 -0.52972186614997396337 0.052440298777892052629 -
0.1588307444604281915 -0.72150290521954718592 -0.65463695733874971427
0.20176685030377725094 -0.45323348104325911212 0.49822411362867580209
0.13824516094164018831;-0.23979589763690309367 0.24713257893427917344 -
0.00042761797802725492242 -0.5683293358487815361 0.60989698125865132727
0.38696874075433551621 -0.48897439147127247372 0.65555788376572565745 -
0.56862394202210220229 -0.3184869129614033012 -0.64148357079701812111 -
0.7141208171998000287 0.43117891565668392539;-0.5630473276294490903 -
0.35371525885517490773 -0.2770635547470743365 -0.13888742766756159708
0.64902709140149728029 -0.033831087337707392571 0.6126027147844947951 -
0.61077648608264945018 0.14298953783117299143 0.18961013997714504242

```

0.34390399255648224086 -0.70976172983094321101 -
0.32073976149918020928;0.95169584531003137418 0.47385738394265319862 -
0.090234853830733943725 0.10400362920755142071 -0.022625362106657279787 -
0.030938235417945117445 0.0030493731935744968972 0.45232302348656427959
1.0456061871942792241 -0.019720210552630405149 -0.46271053027249792189
0.5652659073155443803 0.23848000132361679704;0.74236347815230308012
0.54718656490651707003 0.15558712159472731784 -0.62309702441507930359 -
0.55869255056357447486 0.46508095479304856212 0.60794205262185818661 -
0.070834707382438438783 0.01899872922803730671 0.088469184563731789739
0.86515631835440065522 -0.15076191174712946785 0.076086788079311240551];

% Layer 2
b2 = 0.037014674657301502303;
LW2_1 = [-0.13547833666447653411 0.038056942244190872238 0.022278255650280045369 -
0.005202006241318883617 -0.023818599886230470281 -0.0024861988576421593412
0.038312104456639045624 0.075799915993217087395 -0.12728294196732825583 -
0.36631043768266896565 0.0081202050788539321957 -0.20428477579040793533 -
0.026158202273497164342 0.059215574073305139724 -0.020618136433103492755
0.027046070188968727999 0.0021874403514608592447 0.11310608981770206571 -
0.024697049186870989601 -0.014366810423842994379 -0.21494101390456898937 -
0.00092673282474285700611 0.017436785562950793405 -0.036279009949787305311
0.0074571876227458215938 0.0097672201606948793468 -0.0036931891748628519964 -
0.092301845219597486447 0.0027443539534420836878 0.013249989333218732332
0.019656875851653446419 0.0044909097200043854997 -0.015933714168190102334
0.0072682468755176019015 0.087592483561746453669 0.0039088575731136493766
0.013020899058623433206 -0.031205317556497681047 0.13466911850543952789
0.032239189096652333399];

% Output 1
y1_step1.ymin = -1;
y1_step1.gain = 0.000172941358797959;
y1_step1.xoffset = 684.922698687762;

% ===== SIMULATION =====

% Format Input Arguments
isCellX = iscell(X);
if ~isCellX
    X = {X};
end

% Dimensions
TS = size(X,2); % timesteps
if ~isempty(X)
    Q = size(X{1},1); % samples/series
else
    Q = 0;
end

% Allocate Outputs
Y = cell(1,TS);

% Time loop
for ts=1:TS

    % Input 1
    X{1,ts} = X{1,ts}';
    Xp1 = mapminmax_apply(X{1,ts},x1_step1);

    % Layer 1
    a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*Xp1);

```

```

    % Layer 2
    a2 = repmat(b2,1,Q) + LW2_1*a1;

    % Output 1
    Y{1,ts} = mapminmax_reverse(a2,y1_step1);
    Y{1,ts} = Y{1,ts}';
end

% Final Delay States
Xf = cell(1,0);
Af = cell(2,0);

% Format Output Arguments
if ~isCellX
    Y = cell2mat(Y);
end
end

% ===== MODULE FUNCTIONS =====

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);
y = bsxfun(@plus,y,settings.ymin);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings)
x = bsxfun(@minus,y,settings.ymin);
x = bsxfun(@rdivide,x,settings.gain);
x = bsxfun(@plus,x,settings.xoffset);
end

```

A4.2 Bouncing Busy Hour (BBH) Forecasting

A4.2.1 BSC Voice BBH (SCG Algorithm, 20 Hidden Layer Neurons) Script

```
function [Y,Xf,Af] = myNeuralNetworkFunction(X,~,~)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Generated by Neural Network Toolbox function genFunction, 25-Jul-2021 18:32:45.
%
% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:
%
% X = 1xTS cell, 1 inputs over TS timesteps
% Each X{l,ts} = Qx13 matrix, input #1 at timestep ts.
%
% and returns:
% Y = 1xTS cell of 1 outputs over TS timesteps.
% Each Y{l,ts} = Qx1 matrix, output #1 at timestep ts.
%
% where Q is number of samples (or series) and TS is the number of timesteps.

%#ok<*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
xl_step1.xoffset = [0;0;0;0;0;0;0;0;0;0;0;0;0];
xl_step1.gain = [2;2;2;2;2;2;2;2;2;2;2;2;2];
xl_step1.ymin = -1;

% Layer 1
b1 = [-2.2628900194920338329;-1.6765233392630791709;1.4942111440970526903;-
1.1689313517185673241;-
1.1272576945451879826;0.85926928789802803355;0.67173066369418410471;-
0.37396598174943312953;-0.2903080424877578225;0.037292306194654498142;-
0.22389975277918860996;0.31945949265229695024;0.46196219047401321234;-
0.57620510272626590531;0.88948124729737076954;-0.98023408042284776442;-
1.199812774713133301;1.3971240435256206958;1.5813683736985604256;1.7871461330380749022]
;
IW1_1 = [0.24598394115919161584 0.31759262479426092485 0.088153301871922662425
0.3484073059330728972 -0.40815526460042772516 -0.30775487094839737434 -
0.42653286840139231773 0.61456767271465362334 0.64687843869983352452
0.31289902486987453134 0.1609540995812643871 -0.21577776596809664089 -
0.35786671372127171198;0.59027192519241200586 -0.74134222093874646564 -
0.40630051306965442315 -0.17950589033712796572 0.2106608473979156293
0.4030720695717833757 0.73196925087179676161 0.38850866068731693126
0.096210922762482817272 -0.10637982503813142587 -0.30379811714282262125
0.59137230326748335685 -0.41038463308678102548;-0.55533731189173562992
0.48394556343494782302 0.2176572840842769252 0.14517105182675896224 -
0.13671319371824869338 -0.32899100769859074278 -0.7898287086959874248 -
0.25913906518222501463 -0.18418348149673124947 0.39207594302461584723
0.19920457404766708787 -0.44329548282057640307
0.53206189627371069939;0.84663653277080053172 0.7668654545584375759
0.40901930912242612237 0.39689101398098153295 0.05303150680359387803
0.027265619255687892353 0.47503202308199926618 0.018701853261289039937 -
0.11099673655053397814 -0.49011814441137285803 0.031187571594403776937 -
0.55588397930489541476 -0.81675559519002161046;0.066763240236826462692
0.27715511887544663772 -0.20316010702586345671 0.43449885656516512578
0.62424567491708204958 -0.50241433665276225007 0.56317858852776603307 -
0.1097827952816966407 -0.58864590328616606119 0.15950709693368933517 -
```

0.30663637413393712849 0.49298577141856664285 0.6761950829614004066;-
0.55712331164147188556 0.3264600476522952488 0.032629121775689585994
0.57071754754675152466 -0.26952259802175532188 0.17541515587328254666
0.48512966704943449869 -0.63120408660470550544 -0.61412890364286665257 -
0.48631408157817468396 0.58340814074172475667 0.65023920114536959414
0.36451400437749365002;-0.42205614374758559926 0.42303128825210750685 -
0.01864672637806437383 0.071559206314768822321 0.52774674074511118693 -
0.47167887259982438364 -0.82636029738713623338 -0.53210761772755954535
0.78718661888694818618 -0.32491123223513462825 0.69316802204828986689 -
0.16285954489009360491 -0.017269575558550194494;0.1013994747236659083 -
0.26147707972017425471 0.24579182494113588353 -0.67103671537026643446
0.55294490826455855448 0.33617337741414665686 -0.21216887747630008842 -
0.82618049506151614025 0.84840082524390836305 0.18791464990327139128
0.13333270737174551801 -0.75319091885514588558
0.15538409544717815547;0.73806376714468280298 0.19045292034247990332
0.43934052012519159147 -0.56873796099741258381 -0.20383966734508968122
0.26825104386178783011 -0.32822806238279500501 -0.63915238827291354085 -
0.018462630795984206339 0.26170699785138012983 0.18688823395496947932 -
0.6322698039643870338 -0.4594351401505066268;0.7587412468123664322 -
0.56682262175917152991 0.3628194894493546041 -0.37411058187081835946
0.15588545116456770101 0.40133136926947876599 0.45838198278029596011 -
0.5516500594864165441 -0.81090012139494205279 -0.7207138632409320067
0.1548035180867460836 -0.1199149881654885258 -0.032867688601041791796;-
0.52625710576678030872 0.34605595710852105062 -0.25343540741067560607
0.39745918597505042191 -0.81053639927196230452 0.043266882437432638286 -
0.16019423326163684251 -0.021908703048187698481 -0.44690172257415361479
0.62183672922344801481 -0.57062276725654259568 0.0014455602732028496969
0.70206409408865988464;0.68400222008602695745 -0.69928564271515525341
0.22213258382973358196 -0.33260141281722044315 -0.62522907622342205336 -
0.5787425510618919855 0.54952404375201568953 -0.69119731538086470746 -
0.2170361810011166015 0.4196275426643720774 -0.26795721152247831132 -
0.30977092561571833995 0.095593032322263959966;0.88942606998907713933 -
0.43389472090383013336 0.30495229178978827678 0.59022584737628358553
0.056017822813614823185 -0.51951969843115586301 -0.61615719624049236636
0.77511123674646587034 0.62001974406941251505 -0.026399268255536172123 -
0.054385747153324746905 0.19629518328017550588 0.043094285863110093349;-
0.099019171361694469091 -0.64329910963107728161 -0.60621875529031110652 -
0.27698136414875995648 0.43325730364445341758 0.61197092233797745209 -
0.44862704177665047034 0.63078909938746330344 -0.74450336268673555917 -
0.027625959221192205456 -0.35028420049425573746 0.39494830652059614851 -
0.37673385467276815364;0.54256813518626312298 -0.54797698467603850325 -
0.5784203813972638164 0.54476348100995042323 0.54923928107031383394 -
0.38141776275555777653 -0.57305937254157235383 -0.05769208056254472583 -
0.62636145720211477972 0.0027371068060371215973 0.55816663479915329393 -
0.21276386286354348476 -0.091237206078077698335;-0.56866133620373182644
0.48972198369380842697 -0.20351995867796471296 -0.043549664361111899147 -
0.75739330784926961471 0.49691888566376390957 -0.56779270033559259279 -
0.045505749453079273603 -0.48502797208369219373 -0.26949462500121951214 -
0.54077263225084670761 -0.60547327339986400929 0.36149764568169912016;-
0.20014138934853409335 0.49771291749158796947 0.81613438485800793298 -
0.51359671364607928368 0.026775057204163000418 0.083525477515430657238
0.88789876200075934509 -0.35548203589714760797 0.39307319263350087857
0.064114674501944496354 -0.57291656623899367418 -0.32672148510845189229
0.38866072225358955228;0.70898268348916038128 -0.37422825547898441734 -
0.23496521153015720307 -0.45580018536345923152 -0.046718976363884757996
0.87599705967868857215 0.23919871418101093519 0.72718421321964588078
0.4061302045431210983 -0.054070646250796702836 -0.60732045196948170052 -
0.32117170391695759601 -0.27826040727486189841;0.3477062183628588099
0.84752175784086625221 0.21405962833123198807 0.22262565517289867323 -
0.73584000065772647847 -0.72416377566769551954 0.09401482780268437256 -
0.17418459586133830386 0.32526032694345391461 0.62833430510564625138 -

```

0.41496747579871645639 -0.12294774579632489209 -
0.116589527896012915;0.71479790240606411178 -0.83113625989747674794 -
0.41765412653787581387 -0.026921363048695353543 -0.25995059648600948865 -
0.11671990689217767723 -0.49256466676086196887 -0.68104477351109904415 -
0.11805103063249232365 0.37346216114039859635 -0.10303079362364203631 -
0.044451008280126318162 0.77466829986740970426];

% Layer 2
b2 = -0.35673437713084821254;
LW2_1 = [-1.1208210139154004548 0.13250785859692376656 0.12893891627286435675
0.0030697964519078792429 -0.0024088970685932019865 0.01236432052072429684 -
0.00030491776739102716912 0.0075224065866479530168 0.0066500579050709704432 -
0.0093332857792688857235 0.01148793763605894698 -0.00027779740321880061699 -
0.0053048850572361885142 -0.00063892646327660029739 -0.0050913016418679462802 -
0.0099082630952155373305 0.0034891473381264458478 0.004429094711002639316
0.0063967294959621345124 -0.02609195557333231219];

% Output 1
y1_step1.ymin = -1;
y1_step1.gain = 0.0952380952380952;
y1_step1.xoffset = 2;

% ===== SIMULATION =====

% Format Input Arguments
isCellX = iscell(X);
if ~isCellX
    X = {X};
end

% Dimensions
TS = size(X,2); % timesteps
if ~isempty(X)
    Q = size(X{1},1); % samples/series
else
    Q = 0;
end

% Allocate Outputs
Y = cell(1,TS);

% Time loop
for ts=1:TS

    % Input 1
    X{1,ts} = X{1,ts}';
    Xp1 = mapminmax_apply(X{1,ts},x1_step1);

    % Layer 1
    a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*Xp1);

    % Layer 2
    a2 = repmat(b2,1,Q) + LW2_1*a1;

    % Output 1
    Y{1,ts} = mapminmax_reverse(a2,y1_step1);
    Y{1,ts} = Y{1,ts}';
end

% Final Delay States
Xf = cell(1,0);

```

```

Af = cell(2,0);

% Format Output Arguments
if ~isCellX
    Y = cell2mat(Y);
end
end

% ===== MODULE FUNCTIONS =====

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);
y = bsxfun(@plus,y,settings.ymin);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings)
x = bsxfun(@minus,y,settings.ymin);
x = bsxfun(@rdivide,x,settings.gain);
x = bsxfun(@plus,x,settings.xoffset);
end

```

A4.2.2 RNC Voice BBH (SCG Algorithm, 30 Hidden Layer Neurons) Script

```

function [Y,Xf,Af] = myNeuralNetworkFunction(X,~,~)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Generated by Neural Network Toolbox function genFunction, 25-Jul-2021 18:37:18.
%
% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:
%
% X = 1xTS cell, 1 inputs over TS timesteps
% Each X{1,ts} = Qx12 matrix, input #1 at timestep ts.
%
% and returns:
% Y = 1xTS cell of 1 outputs over TS timesteps.
% Each Y{1,ts} = Qx1 matrix, output #1 at timestep ts.
%
% where Q is number of samples (or series) and TS is the number of timesteps.

%#ok<*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
x1_step1.xoffset = [0;0;0;0;0;0;0;0;0;0;0;0];
x1_step1.gain = [2;2;2;2;2;2;2;2;2;2;2;2];
x1_step1.ymin = -1;

% Layer 1
b1 = [-1.8196243288390214765;1.8281478639651662643;-1.5142712312920731943;-
1.4715558950111331527;-
1.3624021914870725336;1.4455566623456594932;1.0506311741181026953;-

```

0.71602465672382276107;0.73778005929502643578;-
0.8237678095161304892;0.81731042830170286262;0.53736971294840696345;-
0.30647122389664704922;0.093869101695313442169;-0.060153047489819311267;-
0.047463068244932751782;-0.089877236870790244705;0.22184151299416773884;-
0.38826415519936352538;-
1.0962773284709703692;0.71623082621774314394;0.66579043133279158795;0.81075182529958833
566;1.4874080201010990709;-
1.1772778713246803139;1.0642709024857461841;1.5769145353252749242;-
1.9728713704719458288;-1.7506419636612009505;-1.8140610477798475575];
IW1_1 = [0.72027348756666442764 -0.58458836103995026612 -0.59900692621686046913 -
0.65931519654585490908 -0.59720921581029995373 -0.18002975963592426911
0.40711201459111268086 -0.32497573864709144997 -0.5917160916935171322 -
0.64358376745190015722 -0.66637696581870220491 -0.29256112827858876679;-
0.66790315866447791393 0.36546260945291958455 0.070822880609050065104 -
0.64929774567027642718 0.17026628339193272677 -0.33814127428963297595 -
0.49928898965307033642 -0.53122458584766329981 0.53191597325376460681 -
0.35781909767922487342 0.63014446539576218775 -
0.2450586945257239857;1.0873186113667769881 -0.38102626664872740747 -
0.29705625951000269591 0.49999912868715318259 -0.2728636026814001414 -
0.42235111432709521173 -0.47326676559650521225 -0.68657075047352189578 -
0.41901214732358371684 -0.038197062567067702199 -0.47855741789369715677
0.82899463748332924418;0.34315195115743796217 0.53622348275055398137 -
0.60231989784197492721 -0.78553853858587330272 -0.5873238439768884156 -
0.27433918116625732875 0.3603527077496063713 -0.81092688597300810471
0.18973957113121128759 0.5688297209851024272 -0.31101009213047658974 -
0.60091884452652577497;0.32116032821741991032 -0.84775160712196862001
0.15532060927353663615 0.4048481877109914584 0.33141329258717644457 -
0.14538075540151385034 0.12997031474125658734 0.85789876918978780207 -
0.68344492564191539064 0.81099936523479110306 -0.026983878881201451422
0.43342284794734026176;0.026978615894649452672 -0.14372648352872063882
0.22601006041269680602 0.21122025340659369985 -0.12005906202851371334 -
0.12323853441256228669 0.18121156487801634571 0.44123178724995748556 -
0.5632169501785435628 -0.049220610364420075422 1.5283564495992716381 -
1.4182376953959205945;-0.62551542244094593404 -0.44893614450221880352
0.82902857114406525163 -0.66321994326807764963 -0.65205480626376233211
0.19384411266922813022 0.6724077598919713239 0.40285164326835454229
0.56316102980734161765 0.18241655862464603577 0.10790506418655113063
0.20587096106580191068;0.02904634936333719622 0.20557545775676935862 -
0.50574730603554862718 -0.29225861987254764252 0.41482164310702540622
0.80865404182940792133 0.097198412047415749804 0.48724901839294726402 -
0.64169447761549969211 -0.77409028937718182828 -0.75409502451072119023 -
0.9912172886016158424;-0.72645413060025465501 0.049676377354677975218
0.16694753021181468755 -0.051036804065111060491 -0.51712664758784998575 -
0.6063898144980395255 0.33930561747469151213 0.40011489823551898981
0.33569583588549090081 0.93742519820793102081 -0.91945372344456810509
0.78567875205111481485;0.3606238396683955072 0.45789185904127338578
0.15621374411980062935 -0.42938371408377373761 0.50034937436094972796
0.30434433665738630514 -0.095137344748087590562 -0.24505096400310052984 -
0.66830611968517239507 1.0175693261774516607 -0.77770349652190595702 -
0.73719837315043468262;-0.30908581325210277013 -0.56409285273000620009
0.45652847460362233623 0.2038177502168925237 -0.85073261147982937036
0.89103530561964594536 0.60044322473574929244 -0.1728967333651509064
0.58874065121608698536 -0.23237673156719579337 -0.13659311054554226783
0.47453265747666867336;-0.36797534599770820307 0.64410271651541717297
0.085571956003517898881 -0.027297151196818171537 -0.42871839865521982338
0.74732351792136464397 0.36447386988234653504 -0.84325951115752995069
0.19442012619468337919 -0.39312525029384820519 -0.17262985473270547887 -
0.94179876615655810923;0.77800707113955602789 -0.27361723406992816132 -
0.26695739989752348409 -0.83267473310726025648 0.6225008179902997707 -
0.32039127145244877815 0.31024751203944544997 -0.8930140788400416918 -
0.38175054656799167763 -0.34848953169607305158 -0.21482645330085783919 -

0.48709767697130562647;-0.083249340958979428651 -0.55227426878829055656
0.34400731099133441404 0.16389090045429535736 -0.40242304434918907408 -
0.68026624878724351042 -0.53182046723577880432 -1.0637893787096353027
0.33506531292458263671 -0.73820878256083299895 0.058465976282797857411
0.47430305575110254246;0.56517566128314666862 0.68800607651597334158
0.75633989118440192456 0.40299596139393939476 0.0079502855147389125956
0.35560558819369531003 0.089663785351352706665 -0.71075770362361245791 -
0.88921565302456739843 0.3726700182562540653 0.35528402389197660804 -
0.51539666280359264228;-0.77718595247299682161 0.32499821955072388802
0.74619168622817200198 -0.60910996214498902557 -0.18209845327407492288 -
0.68900430304528714753 -0.10128009596987173901 -0.66006057750354263369
0.46541265843610457598 -0.32857371919917338632 -0.081010066009468861825 -
0.62972527227932062477;-0.53012216106068232868 -0.1497547376379832762
0.36513306217555591138 0.48705922360083842015 0.45343715770742926141 -
0.18108800964130486455 0.17876120262318029308 0.035326325505243838521
0.78090052202393633873 -0.54429158060136306752 -0.96087176981572253176 -
0.5212853875970301365;0.69847203536537871837 0.25513004151770690697
0.29988389874817511505 -0.42178587801426725701 0.056220875312539178581 -
0.17097756816177434036 0.02037954431930057847 -0.024922289383269412982
0.38479486823715403077 -0.73032688416622915106 -1.1231669338162102445
0.84630372120033603434;-0.48635154779742306719 -0.26750445552078028211
0.73948178984179391282 -0.90046141871352369801 0.74281022125086504371
0.053707039683138307895 -0.42154146865229102925 0.059900862575632561613
0.3925205519336043003 0.74552445288532520618 -0.036812447551721472205 -
0.20549399652203509392;-0.0086320787294669430734 -0.14286299585018058522 -
0.27880356175363246729 -0.066718166631595868243 -0.62535814968144831649 -
0.57181680204993368122 -0.14110213476235730634 -0.47663895465059874512 -
0.27330872816518525514 0.73970982546502217492 1.0610609161357407171 -
0.84878255311914863057;0.18847712530064245073 -0.98218840714149102666 -
0.22841659282348641957 0.94206996348132843178 -0.64969681394728795443
0.23670831783686871463 0.74282079574332759986 0.24433642325304341658
0.56678631203979057407 0.01582325478989604281 0.080969432603504740142
0.2235929987636738614;-0.12286284669025167304 -0.047513924752870924373 -
0.47124998220483027556 -0.10890432391040935634 -0.20905637540216195913 -
0.1908246456479631703 -0.070332778626158179436 0.88474700976864173452 -
0.73177467834474141384 -0.71457827801816808933 0.45502907497533023884 -
1.5226100195820149175;0.21107067702534676457 -0.0066462693329144449314
0.2222525962492110374 0.069070114781707395157 -0.61070504198251807093 -
0.49614388038568613126 -0.4823151005052589424 -0.2118029998064443753
0.10385158989642295779 -0.3929984720498937989 -1.0384115539591685096 -
1.2801578293272379838;-0.21122066539966674603 -0.29704885403288394663
0.097866168125161995262 -0.046844959420279629059 0.76665593869385917358
0.67113722578735701685 0.36217199117877141301 -0.77145044829276421261 -
0.38587347885164458017 -0.88001909622014762302 -1.5304494495768730111 -
1.7175476416628678855;-0.64296744932908267423 -0.3156435235031317954
0.24735044523673427319 0.0075144693933962281029 -0.28289525006510168881 -
0.23621077857719338655 -0.42509763999966515557 -0.61551485525784355168 -
0.0061497914022639122225 0.59203425561099620733 0.16108459157999652223 -
1.3009244053814807973;0.10964018957141417887 0.12920470346426413122 -
0.087745125339159277256 -0.1305845542861092945 0.70356868881019407613
0.81337728678026022244 0.28058305826341256584 -0.86079778916362081898
0.96998091994145030359 -0.65836815790653557823 0.71127394161162005304
0.38316831760565811527;0.61296364154260174484 -0.41260672024115102596 -
0.86718306210908491227 0.6807181149878317461 -0.36594548107807384207 -
0.26195247381812120091 -0.078237443088304950423 -0.21382789959964410009
0.46314406535582236524 0.84402118347775600959 -0.23460644191053689478
0.36108894694088861099;0.090467276876092239668 -0.11184180726489549584
0.16864999191628071706 -0.066345944860988478875 0.79162453334179128106
0.75720461570018193775 0.35884855481790428611 1.0896662391267024628 -
0.1848509441612447568 0.45463217604586164677 0.46920645976820579692
0.33803356514509658171;-0.97115915781081041391 0.64256905506615391577

```

0.10076819220105466546 0.62035404793413861935 0.5864200816423531526 -
0.50184528950667084413 0.22979275852142308501 -0.47034290308389048452
0.34527197745586862832 0.70724696029833045152 -0.18743348739471535347
0.18416457538960501195;-0.60526899744961060179 -0.5218689699932806958 -
0.62408745509011154251 0.58574376029807007082 0.61471571770201272056 -
0.24108903622990132676 -0.708847253181280057 -0.57843554008078257311 -
0.64589953484370388193 0.3848357682620111575 -0.054138333688360629203
0.2823774767565208621];

% Layer 2
b2 = 0.74920461183928832583;
LW2_1 = [0.041111756643350066964 -0.022676019660268397632 -0.076331701338816740354 -
0.01736086592275532664 -0.045435768206633406208 -0.1006862752748510087 -
0.036156384897980183013 -0.10009207348844603724 -0.051972775616041258506
0.022305301509040767471 0.03548027529810229147 -0.05589794132208676658
0.0077006450103245952313 -0.05477749366819599508 -0.025873367309896463939 -
0.01798315149780534139 0.00044805949365996858985 0.0050200621468747923248 -
0.041823605440821538803 -0.10102917929777895101 -0.019228250852969881041
0.076179677541501955584 0.071950353139461534213 0.25265665867713099635
0.060148149268784451005 -0.10278378675952862253 -0.036421246581646832263
0.23159380678834581846 -0.01298919186788975548 -0.014835972021690266992];

% Output 1
y1_step1.ymin = -1;
y1_step1.gain = 0.0952380952380952;
y1_step1.xoffset = 2;

% ===== SIMULATION =====

% Format Input Arguments
isCellX = iscell(X);
if ~isCellX
    X = {X};
end

% Dimensions
TS = size(X,2); % timesteps
if ~isempty(X)
    Q = size(X{1},1); % samples/series
else
    Q = 0;
end

% Allocate Outputs
Y = cell(1,TS);

% Time loop
for ts=1:TS

    % Input 1
    X{1,ts} = X{1,ts}';
    Xp1 = mapminmax_apply(X{1,ts},x1_step1);

    % Layer 1
    a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*Xp1);

    % Layer 2
    a2 = repmat(b2,1,Q) + LW2_1*a1;

    % Output 1
    Y{1,ts} = mapminmax_reverse(a2,y1_step1);

```

```

        Y{1,ts} = Y{1,ts}';
end

% Final Delay States
Xf = cell(1,0);
Af = cell(2,0);

% Format Output Arguments
if ~isCellX
    Y = cell2mat(Y);
end
end

% ===== MODULE FUNCTIONS =====

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);
y = bsxfun(@plus,y,settings.ymin);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings)
x = bsxfun(@minus,y,settings.ymin);
x = bsxfun(@rdivide,x,settings.gain);
x = bsxfun(@plus,x,settings.xoffset);
end

```

A4.2.3 RNC Data BBH (SCG Algorithm, 20 Hidden Layer Neurons) Script

```

function [Y,Xf,Af] = myNeuralNetworkFunction(X,~,~)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Generated by Neural Network Toolbox function genFunction, 25-Jul-2021 18:39:14.
%
% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:
%
%   X = 1xTS cell, 1 inputs over TS timesteps
%   Each X{1,ts} = Qx12 matrix, input #1 at timestep ts.
%
% and returns:
%   Y = 1xTS cell of 1 outputs over TS timesteps.
%   Each Y{1,ts} = Qx1 matrix, output #1 at timestep ts.
%
% where Q is number of samples (or series) and TS is the number of timesteps.

%#ok<RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
x1_step1.xoffset = [0;0;0;0;0;0;0;0;0;0;0;0];
x1_step1.gain = [2;2;2;2;2;2;2;2;2;2;2;2];

```

```

xl_step1.ymin = -1;

% Layer 1
b1 = [-
1.8669275804009648478;1.6012413868722519261;1.4417615313075069494;1.2446523011480228327
;1.0710650908133168002;-
0.78230150892941185958;0.52187928101465685948;0.33891814555103788598;0.2542353846602962
153;0.13615030619906887122;0.026091674455060335364;-0.3162224359154305073;-
0.49270356214896071645;-0.598430287938447103;-
0.85986542678613142598;1.0558204761155309814;-
1.2605961983447788732;1.4109282958585014356;1.5161999412204405502;1.9077821799504799394
];
IW1_1 = [0.068004928701636435662 -0.7528177274168810218 0.5797795216085287473 -
0.045552282794384703346 -0.31139854349077306095 -0.73783029027255442323 -
0.75828025946962596127 0.68094478631344479425 0.17912593161526957353 -
0.45387816533991387491 0.053732742884089725044 0.26812717243596490002;-
0.22394759324572305337 0.0077423032325045493263 0.40057620919633585288 -
0.78778374791178684688 -0.71015097467251153684 0.36927098909183975195 -
0.36379841131958523537 -0.13994880380913993223 0.0083109379055960955207
0.65165112222165955114 -0.87244347677547839925 -0.60891037795818114819;-
0.54192901445684948936 0.36735836837653368381 0.61757828349997123674
0.026656131621971758394 0.54161360966067251255 0.11106672615923515735
0.10627527467641863235 -0.65222181552907332325 0.21707440989729484548 -
0.32840986604237243363 0.71245380483249587389 -0.91149546253671998031;-
0.40046331372377708124 0.31037661358164869707 -0.79399219114729746583 -
0.69806914479046433541 0.45330376199630950662 0.025299184727700284314
0.55591857486606599537 -0.3779551625498508427 -0.50129136908082561064
0.39782328682047224389 0.63999059363675192547 -0.57169159201076602539;-
0.10796026744743723869 0.57289647180032199358 -0.64944477251349730373 -
0.61931056892790881108 0.18334664743539003107 -0.36490979760206310178
0.26883769342600072338 -0.75880264444605116658 0.29560664736290415133 -
0.5451163163812261514 -0.75583853343523554891 -
0.45594710723815190523;0.68469389141441960867 0.34297405541334857748
0.074234925046904778889 0.40467689585243188377 -0.6158675722983290024
0.51013550473693591858 0.52148002982410412987 -0.74026193453368449582 -
0.7395012682109579627 0.11513939402019490732 0.22069318326775097527
0.41528772736359781392;-0.2821454384254469816 0.78676432186111511236
0.59098840329137292127 -0.22280980217034418445 -0.42573145109399718189
0.38032891137740698628 0.29432554881139494363 -0.60261369835867073519 -
0.35127793781353289226 -0.81050809556663860889 0.65863082013715268381
0.22501444280745516724;-0.31826952042686823541 -0.60686169640445364326 -
0.40097166458472788264 -0.2647981366994505148 0.28522055935140400162 -
0.92724575170896861298 -0.53667075677435294168 0.19518235809954875837
0.71717470038198383175 0.11222787071824048832 -0.072948753062074184195 -
0.80595262792815347552;-0.35157025599887786749 0.58863666395889047145
0.56730479626635921253 0.55704563875512602689 -0.1561206722239520148
0.046422699017709040392 0.78121875785558303473 0.15817297662027815508
0.70875211952347461075 -0.84294410490962723426 0.28381612680130896553 -
0.13166882727019385557;0.02452932644532236639 -0.81636747675667376001
0.26630832620701866409 0.90299381835703251831 0.4991180806318771257
0.687951376268732262 -0.21165719706090535546 -0.36546819296290905532
0.00041694818385905059021 0.80386088261011234835 -0.040146452648433673949
0.24049696033161138686;0.58510411116864669001 -0.052739619580238782104
0.45742737425858537303 0.90179922269587875494 -0.66457474128439142369
0.10012290879589867687 0.19753883626836868381 0.5576732402332869043 -
0.81472146821804092287 0.3299862647249039993 -0.21888904447235491002 -
0.18063641892478465856;-0.37172158273964134301 0.42693942580152488553
0.7629097139010145856 0.25276688214289044776 0.13743401789225981324
0.47743883598551473124 0.49859433060695135254 0.33609830507788013287 -
0.71464709291601202779 -0.47112436783919781069 -0.75500506736248584883
0.11382701709314951843;-0.76427288249755842564 0.39772084354498848935

```

```

0.41076117292896530131 -0.43053523685611100325 -0.44488369296639607153
0.092195190133874033167 0.5578317079079299079 0.70034835483945367951 -
0.75125746560434758514 -0.10404650569890254441 0.55555257267088098416 -
0.43264151904883357824;-0.27196425330719664615 -0.38836558507743035307 -
0.58173672012057286995 -0.32830232703397005389 -0.31038141839293487134 -
0.18263399738953300933 -0.31166621374165015546 -0.077558465083139307983
0.77926012140311418097 0.75382663294160834333 0.84535429026408748232 -
0.61173588060840977665;-0.42091810677167257326 0.22757304049848847693 -
0.94355120467626540925 0.35209600061726742393 -0.31839443653810028945 -
0.990691840118814393 -0.11613802748009086074 0.047350944333044570067
0.68779226150231143055 0.022026692330503205158 -0.051630106183390575558 -
0.61154136807851589008;0.5626042526454951842 0.64984753827941366122 -
0.64604473635704806611 0.093318787729909258299 0.7910821453433981576
0.045487947415723040123 -0.5704026643221999926 0.51026543946576752298 -
0.081004939599092662394 -0.77259423136414140387 0.13343553746978198404
0.41703538962860869921;-0.56035105963239173477 0.37380192085513824773
0.38090396876143334248 -0.36309167637872979961 0.32752848853248150984
0.51541677903621507806 0.067819181685181487151 0.16111833682282450386
0.78881668293125450919 0.19519950341869002353 0.91476005289408324206 -
0.45802300219348512922;0.59398472175733307843 0.55619456499015973439 -
0.67644638849209770548 0.88094124078243540854 0.21350156307034484637
0.71264618263901335915 -0.075794598377262975553 0.49011428554397346025
0.53748861077124199248 -0.12244688375479963194 -0.17625279414177000281 -
0.43049356993675386152;0.64111059115349122717 0.11204820956111347097 -
0.49078388267527200961 -0.19909413099242256884 0.52946793042137019203 -
0.41899804096278059351 -0.68900935725718948088 0.87714379394701291748
0.037377872103131225268 -0.83312221462284208329 -0.42818248503078704115 -
0.085137088930583249002;-0.0045489622272585177043 -0.20548259213718098048 -
0.072422435778179097832 -0.52730273190944443318 -0.39568126012237736511 -
0.62444677639630408184 0.19776929217005731476 -0.65949322684440414566 -
0.61771596686920038355 0.44100597306704680811 0.65965275774661735753 -
0.46181764886968873496];

% Layer 2
b2 = 0.8150358630787665204;
LW2_1 = [0.0012709486552648615112 0.0043352645110647783019 -0.00068137221015313140932 -
0.013979377983237275748 0.024847529642296882491 -0.024727361909873190465 -
0.0016365322500814000369 -0.01747071965043949654 0.013054082331007844237
0.00032134730246722708474 0.024310709444513640964 0.0003045751410993807317 -
0.031772930936930003643 0.0027109382619553562543 0.0075252054040103287524
0.0076755220497079432174 0.010010383851404527911 -0.01203051523904382844 -
0.0087713866224117288672 0.037323417384380490336];

% Output 1
y1_step1.ymin = -1;
y1_step1.gain = 0.0869565217391304;
y1_step1.xoffset = 0;

% ===== SIMULATION =====

% Format Input Arguments
isCellX = iscell(X);
if ~isCellX
    X = {X};
end

% Dimensions
TS = size(X,2); % timesteps
if ~isempty(X)
    Q = size(X{1},1); % samples/series
else

```

```

    Q = 0;
end

% Allocate Outputs
Y = cell(1,TS);

% Time loop
for ts=1:TS

    % Input 1
    X{1,ts} = X{1,ts}';
    Xp1 = mapminmax_apply(X{1,ts},x1_step1);

    % Layer 1
    a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*Xp1);

    % Layer 2
    a2 = repmat(b2,1,Q) + LW2_1*a1;

    % Output 1
    Y{1,ts} = mapminmax_reverse(a2,y1_step1);
    Y{1,ts} = Y{1,ts}';
end

% Final Delay States
Xf = cell(1,0);
Af = cell(2,0);

% Format Output Arguments
if ~isCellX
    Y = cell2mat(Y);
end
end

% ===== MODULE FUNCTIONS =====

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);
y = bsxfun(@plus,y,settings.ymin);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings)
x = bsxfun(@minus,y,settings.ymin);
x = bsxfun(@rdivide,x,settings.gain);
x = bsxfun(@plus,x,settings.xoffset);
end

```

A4.2.4 LTE Data BBH (SCG Algorithm, 10 Hidden Layer Neurons) Script

```

function [Y,Xf,Af] = myNeuralNetworkFunction(X,~,~)
%MYNEURALNETWORKFUNCTION neural network simulation function.

```

```

%
% Generated by Neural Network Toolbox function genFunction, 25-Jul-2021 18:43:30.
%
% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:
%
% X = 1xTS cell, 1 inputs over TS timesteps
% Each X{1,ts} = Qx8 matrix, input #1 at timestep ts.
%
% and returns:
% Y = 1xTS cell of 1 outputs over TS timesteps.
% Each Y{1,ts} = Qx1 matrix, output #1 at timestep ts.
%
% where Q is number of samples (or series) and TS is the number of timesteps.

%#ok<*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
x1_step1.xoffset = [0;0;0;0;0;0;0;0];
x1_step1.gain = [2;2;2;2;2;2;2;2];
x1_step1.ymin = -1;

% Layer 1
b1 = [-1.841070826128158977;-1.5300060628821479103;-
1.2068045755102507766;0.67699506495029870834;0.29384465623219291341;0.24257582342308797
729;-0.63818312700885204691;1.0403380537983835108;-1.5254825877322828465;-
2.1637219370156115517];
IW1_1 = [1.1848246747757413821 -0.45176574968034466373 -0.0079326367088919345905 -
0.11037434403661426641 -0.4177754807965828876 1.1919463020989931046
0.55542837260158228752 -0.094247319256036909962;0.91678358986345565018
0.77543267345607003183 0.76113539332838908713 0.82477562480772070863
0.16097095898432997596 -0.36503007676620591315 0.4807398199095806457 -
0.11673866105527358483;0.3375344734107957323 -0.28707761356803424002
0.54088397920001496377 1.0633603992587905562 -0.10968140291163813727 -
0.24005277237964930959 0.078766834419021852787 1.2263041146557207561;-
1.0384361668465769135 0.029754714625310325904 1.3787688305692764601
0.27469143126774825392 -0.34298459192258706763 0.23629096786113601425
0.16457522641622518433 -0.31719101919934600264;-0.65482046071117328001 -
0.54802462046520927785 0.8989652867841417816 0.34420792339460232201 -
0.438115783518322921 -0.71887016667184977514 -0.27518919335117270064
0.986061642211970770417;0.062561238848485398178 0.17135043479127601285
0.043814108967061865529 -0.78238253293513693176 1.11485240190217616 -
1.1763697830716435266 0.12730552589635241167 -0.27413344489377983182;-
0.27489475470955027081 -0.79079482689325231171 -0.55595841637943910207
0.039528788665624939314 0.82216691206869152442 0.69625964600435441554
0.73713185896515187601 -0.81392275201919039862;0.56836532614282009312 -
0.39979059755114731978 -0.78576781143465190826 1.0339222333441291379 -
0.70972972209287121803 -0.091300598455414869181 -0.41874755154144238656
0.74132283946542210096;-0.27673507759283832996 0.12968682149501098433 -
0.41879171514601248871 -0.57637593991666646875 -1.1245653103740946843 -
0.81434124839406984986 0.78537942990075138017 0.042269637313261131106;-
0.34905575419456691399 0.18573267694522843652 0.17674992582645973616 -
0.22693865614746575043 -0.82074882539417670646 -0.51098068122174511441 -
1.1334320288955670453 0.13345871115632723658];

% Layer 2
b2 = 0.49491381820111651768;
LW2_1 = [0.08434097792928171855 0.16677466569730414747 -0.079119040641708604511 -
0.1452833294933391084 0.055230619436454464477 0.018311286893213013505 -

```

```

0.0020445785352372451704 -0.05106404669849021194 -0.00012396529544786427657
0.23639605737666016805];

% Output 1
y1_step1.ymin = -1;
y1_step1.gain = 1;
y1_step1.xoffset = 20;

% ===== SIMULATION =====

% Format Input Arguments
isCellX = iscell(X);
if ~isCellX
    X = {X};
end

% Dimensions
TS = size(X,2); % timesteps
if ~isempty(X)
    Q = size(X{1},1); % samples/series
else
    Q = 0;
end

% Allocate Outputs
Y = cell(1,TS);

% Time loop
for ts=1:TS

    % Input 1
    X{1,ts} = X{1,ts}';
    Xp1 = mapminmax_apply(X{1,ts},x1_step1);

    % Layer 1
    a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*Xp1);

    % Layer 2
    a2 = repmat(b2,1,Q) + LW2_1*a1;

    % Output 1
    Y{1,ts} = mapminmax_reverse(a2,y1_step1);
    Y{1,ts} = Y{1,ts}';
end

% Final Delay States
Xf = cell(1,0);
Af = cell(2,0);

% Format Output Arguments
if ~isCellX
    Y = cell2mat(Y);
end
end

% ===== MODULE FUNCTIONS =====

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);

```



```
y = bsxfun(@plus,y,settings.ymin);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings)
x = bsxfun(@minus,y,settings.ymin);
x = bsxfun(@rdivide,x,settings.gain);
x = bsxfun(@plus,x,settings.xoffset);
end
```