A Back Propagation Based Spiking Neural Network Approach for Intelligent Link Decisions In Satellite Communication

by
Meenakshi Visweswaran

A thesis submitted to the Department of Engineering Technology,
University of Houston
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in Engineering Technology

Chair of Committee: Dr. Ricardo Lent

Committee Member: Dr. Deniz Gurkan

Committee Member: Dr. Ahmed Abdelhadi

University of Houston
April 2021

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitude to my advisor Dr. Ricardo Lent for his continuous guidance and immense knowledge. His support and motivation brought me to successfully complete this research study. I would like to appreciate the one-to-one meetings and knowledge sharing discussions we had throughout the journey of this research.

Besides my advisor, I would like to thank my thesis committee: Dr. Gurkan and Dr. Abdelhadi, for accepting to be part of the committee chair and reviewing my work and providing valuable suggestions. Thanks to professor Gandhimathi Velusamy and my other classmates for all the technical discussions, timely review meetings and ample support in learning necessary tools. Thanks to my academic advisor Ms. Dawn Wolf-Taylor for her constant support and motivation.

Finally, I would like to thank my parents, sister, in-laws, and friends for the immense love and understanding. Thanks to my dear friend Premvishnu for his boundless patience and valuable talks whenever needed. Thanks to my beloved husband Vignesh for having faith in me and being my strength.

ABSTRACT

A Spiking Neural Network (SNN) with neuromorphic architecture for optimal link decisions is put forward in this paper. SNNs can adapt to the various changes in the working environment quickly, for maintenance or advancement of the selected performance metrics. Such results can be appealing for satellite networks with orbital operations involving either stationary or manned aids, which would provide directions for autonomy in CN decisions. The satellite on-board processing capabilities, traditionally, have been a limiting factor for advanced satellite communication strategies. Additionally, with deep space explorations rising, the demand for bandwidth is increasing, which can be achieved by making communication systems more efficient. Manual updating procedures for satellite operations gives rise to chances of configuration errors. Since AI has been showing continuous improvements and glorious performances, when applying it to convert manual operations to intelligent ones, some errors can be avoided. In scenarios where the delay time of an operator responding is considerable, the spacecraft must be able to autonomously make decisions. Intelligent systems can help improve spacecraft reliability by being trained to react to unexpected situations and guide the spacecraft to safer operational states with autonomous decision-making. This serves as an apt area to apply an SNN model for a lighter space network on a first-hop level. This literature will be focused on enabling flexible routing for link selection with the help of Spiking Neural Networks. This path selection problem is approached by applying Spike Neural Network (SNN) to classify satellite downlinks based on link cost to improve learning, and later analyze the classification and link decision capabilities of the network with respect to a traditional neural network. The

spiking network has inbuilt Back Propagation (BP) implemented in the framework, Nengo. The system managed to achieve better accuracy even when activation was provided in hidden layer instead of output layers. Tweaking the firing rates, epochs and batch size of the data might yield better results. For the LEO scenario, a maximum accuracy of 86% was obtained for synthetic data using SNN and for the GEO scenario, a maximum of 98.5% was obtained.

TABLE OF CONTENTS

ACKNOWLEDGMENTS ..... **ii**
ABSTRACT ..... **iii**
TABLE OF CONTENTS ..... **v**
LIST OF TABLES ..... **vii**
LIST OF FIGURES ..... **viii**
1. INTRODUCTION **1**
1.1. Challenges in current space systems 1
1.2. Challenges in SNN Implementation 2
1.3. Motivation 3
1.4. Thesis Organization 4
2. SPIKING NEURAL NETWORKS **6**
2.1. Neural Networks: a brief insight 6
2.2. States of neuron 6
2.3. Membrane potential of a neuron 7
2.4. Neuron model and working of NN 7
2.5. Why SNN? 9
2.6. Assumptions 9
2.7. Use of Artificial neural networks in space 10
2.8. Keras 11
2.9. NengoDL 11
3. LITERATURE REVIEW **13**
3.1. Prior works 13
3.2. Inference from prior works 23
4. PRINCIPLES OF SATELLITE COMMUNICATION **26**
4.1. Satellite Link Budget 26
4.1.1. Free space losses 27
4.1.2. Equivalent Isotopically radiated power: EIRP 28
4.1.3. Additional Losses 29
4.1.4. Signal to Noise Ratio (SNR) 30
4.2. Satellite communication example 31
4.3. Link budget conventions 32

4.4. Bit Error Rate (BER) 32
4.5. Cost calculation 33
5. DATASET GENERATION 35
5.1. Simulation procedure 35
5.1.1. Transmitter and receiver properties 36
5.1.2. Space Scenario 38
5.2. Reports and graphs 40
5.2.1. GEO Scenario without explicit RF loss 41
5.2.2. GEO Scenario with explicit RF loss 42
5.2.3. GEO Access durations 43
5.2.4. LEO Scenario without explicit RF loss 44
5.2.5. LEO scenario with explicit RF losses 46
5.2.6. LEO Access durations 48
5.3. Link budget reports 49
6. DATA ANALYSIS 51
6.1. Behavior of measured and theoretical data 51
6.2. LEO scenario 53
6.2.1. Synthetic data 53
6.2.2. Simulated data 57
6.3. GEO Scenario 61
6.3.1. Synthetic data 61
6.3.2. Simulated data 66
7. IMPLEMENTATION OF SNN 71
7.1. Creating the Keras model 71
7.2. Converting the model to Nengo Model 72
8. RESULTS OF SNN 78
8.1. LEO Scenario Results 79
8.2. GEO scenario Results 83
9. CONCLUSION AND FUTURE WORK 88
REFERENCES 92

LIST OF TABLES

3.1 Background information of various works based on SNNs 15

3.2 Average energy per inference and power consumption chart. Source: [39] 24

5.1 List of recommended values for satellite communication by ITU 36

5.2 List of common features 37

6.1 Sample Measured and Theoretical BER values 51

6.2 Sample Eb/N0 and BER for LEO scenario Synthetic data 53

6.3 Sample Occupancy and Response time for LEO Scenario Synthetic data 53

6.4 Sample Simulated data LEO (Eb/N0 vs BER) 57

6.5 Sample Occupancy and Response time for LEO Scenario Simulated data 58

6.6 Sample Synthetic values for GEO Synthetic data (Eb/N0 vs BER) 61

6.7 Sample Cost and Occupancy for GEO synthetic data 62

6.8 Sample Simulated values Eb/N0 vs BER for GEO 66

6.9 Sample Simulated values cost and occupancy for GEO 67

8.1 Impact chart LEO synthetic data Keras 79

8.2 Impact chart Leo Synthetic data Nengo 79

8.3 LEO Simulated data in Keras 81

8.4 LEO Simulated data in Nengo 81

8.5 GEO synthetic data Keras 83

8.6 GEO Synthetic data Nengo 84

8.7 GEO simulated data Keras 85

8.8 GEO Simulated data in Nengo 85

LIST OF FIGURES

2.1 The perceptron; Source: [24] 6
2.2 Structure of a neuron. Source: [38] 7
4.1 Wireless Transmission 26
4.2 Quadratic law 27
4.3 EIRP scenario 29
5.1 Satellite scenario considered 35
5.2 STK Objects 36
5.3 Sample Transmitter properties of LEO orbit 37
5.4 LEO Orbit simulation screenshot before contact. 39
5.5 LEO Orbit simulation screenshot at common contact. 39
5.6 GEO Orbit simulation screenshot. 39
5.7 2D view of GEO scenario 39
5.8 Antenna pattern visualization from one of the ground stations. 40
5.9 Access page for transmitter 40
5.10 Explicit RF environment losses 41
5.11 Report and Graph manager page 41
5.12 GEO - BER and Distance Vs Time 42
5.13 GEO - BER and Free space loss vs Time 42
5.14 BER, Free space loss Vs Time 43
5.15 Rain Loss and CloudsFog Loss Vs Time 43
5.16 Access duration for GEO for Transmitter to Receiver 1 44
5.17 Access duration for GEO for Transmitter to Receiver 2 44
5.18 Access duration for GEO for Transmitter to Receiver 3 44
5.19 BER, Distance Vs Time (at first common contact) 45
5.20 BER, Freespace Loss Vs Time (All) 45
5.21 BER, Freespace Loss Vs Time (At first common access) 46
5.22 BER, Propagation Distance Vs Time (at first common contact) 46
5.23 BER, Free space loss Vs Time (all) 47
5.24 BER, Free space loss Vs Time (at first common contact) 47
5.25 Rain and fog loss (All) 47
5.26 Rain and Fog Loss Vs Time (At first common contact) 47

5.27 Access duration for Receiver 1 48
5.28 Access duration for Receiver 2 48
5.29 Access duration for Receiver 3 49
5.30 Link budget report 49
6.1 Measured BER 52
6.2 Theoretical BER 52
6.3 Link 1 Eb/N0 Vs BER 55
6.4 Link 2 Eb/N0 Vs BER 55
6.5 Link 3 Eb/N0 Vs BER 55
6.6 Link 1 Occupancy and Cost for each observation 56
6.7 Link 2 Occupancy and Cost for each observation 56
6.8 Link 3 Occupancy and Cost for each observation 56
6.9 Cost for each observation 57
6.10 Link 1 Eb/N0 Vs BER 59
6.11 Link 1 Eb/N0 Vs BER 59
6.12 Link 1 Eb/N0 Vs BER 59
6.13 Link 1 Occupancy and Cost for each observation 60
6.14 Link 2 Occupancy and Cost for each observation 60
6.15 Link 3 Occupancy and Cost for each observation 60
6.16 Cost for each observation 61
6.17 Synthetic data BER Vs Eb/N0 link 1 63
6.18 Synthetic data BER Vs Eb/N0 link 2 64
6.19 Synthetic data BER Vs Eb/N0 link 3 64
6.20 Link 1 Occupancy and Cost for each observation 64
6.21 Link 2 Occupancy and Cost for each observation 65
6.22 Link 3 Occupancy and Cost for each observation 65
6.23 Cost at each observation 66
6.24 Link 1 Eb/N0 Vs BER 68
6.25 Link 2 Eb/N0 Vs BER 68
6.26 Link 3 Eb/N0 Vs BER 68
6.27 Link 1 Occupancy and Cost for each observation 69
6.28 Link 2 Occupancy and Cost for each observation 69
6.29 Link 3 Occupancy and Cost for each observation 69

6.30 Cost at each observation..... 70
7.1 Code snippet from keras..... 72
7.2 Code snippet from Nengo 73
7.3 Sample model summary snippet 73
7.4 Sample SNN code snippet..... 74
7.5 Sample run_network() code snippet..... 74
7.6 Prediction for imparting spiking activites..... 75
7.7 Prediction scaling firing rates snippet..... 77
7.8 Prediction performing synaptic smoothing..... 77
8.1 Generic loss plot..... 78

1. INTRODUCTION

1.1. Challenges in current space systems

With the rise in new horizons of discoveries in space, organizations, such as NASA, have been looking to expand their communication infrastructure beyond the current conventions. Outer space involves huge budgets for satellite, and this comes with several challenges. The challenges in satellite communication may include errors in data transmission, low signal to noise ratio (SNR) of the link, poor antenna directivity, high noise spectral density, high bit error rate (BER), atmospheric losses, polarization mismatch, satellite drag, error during configuration updates at the satellite, etc. Errors are every common in satellite systems due to the numerous uncertainties such as equipment error, shared band, error planning, intentional errors, etc. For example, due to certain environmental effects (signal corruption, satellite drag, etc.), assuming an error to have occurred while updating a certain operational attribute at the satellite. In such cases, an algorithmic approach would involve a good amount of time in performing troubleshooting and resuming operations as the speed of light is $3x10^8$ **m/s**. For instance, if the satellite is 6 light minutes way, it would take a precious 12 light minutes, source-destination, and destination-source, to troubleshot and bring the system back to normalcy. But, the involvement of an artificial agent, such as a neural network, could help us avoiding the error by learning the associated features and making opportune decisions and predictions before such an unfortunate event may occur, thereby avoiding the time taken to troubleshoot and resume operations altogether.

The above stated issues lead to the need of autonomous satellite behavior implementations to avoid manual updating and less exchange of signals from ground to satellite without human supervision or interference, and make adaptive decisions considering the uncertainties. Further, recent advances in this area have provided directions for enhanced on-board processing which can enable

futuristic communication technologies, such as flexible routing/channelization, beamforming, free-space optics and also signal regeneration [1]. On a practical basis, once deployed to the orbit, performing physical modifications would be costly in space systems. Among the various renditions to exploit use of neural networks for numerous applications, the area of SNN has not been completely exploited specially in space explorations. Given a set of attributes, the SNN can choose an optimal choice or perform certain modifications according to the learned features. For example, assuming a link selection problem based on associated costs of available links, the SNN can be used to make an optimal link decision to successful data transmission to the destination. For simplicity, this literature will be focusing on predicting the link on a first-hop basis.

1.2. Challenges in SNN Implementation

Literature [8]-[23] show different approaches to implement SNN on software and hardware platforms. Some of the major challenges are inadequate data, bad input feature selection, inadequate learning rate, overfitting/underfitting, etc. During the early stages of the thesis, one of the major concerns was to create the SNN. Although several SNN simulation platforms are available as described in [23], they required several prerequisites and knowledge for immediate usage and deployment. The commonly known frameworks such as Tensorflow worked only for creating convolutional neural networks (CNN) and deep neural networks (DNN) without spiking activities.

Secondly, finding relevant satellite data was difficult due to restricted access. Previous works which provided graphic results involved grants from governmental organizations. Several data sources had restricted access, which lead to the use of simulation software for obtaining necessary data. The simulation software used in this literature is STK communication tool (More details later in this thesis). Thirdly, Nengo provides an affable UI for visualizing the simulation of SNN, but

the specific method that is considered in this literature does not enable visualizing the firing of neurons of SNN graphically. Finally, the satellite scenario considered is a simple network with three links for selection. Increasing the number of inputs would mean creating more links within the simulation and data extraction. Although this is not a cumbersome task, the post processing would require ample time involving budget calculations, data processing, training, and predictions. Typically, a LEO scenario involves more than 400 links and simulating such a scenario is arduous.

1.3. Motivation

The recent Mars-Rover incident clearly indicates the rise in the number of Space missions today compared to those in the past. As mentioned earlier, deeper space explorations put forward the necessity of better communication systems, which is a challenge. This pulled at the thread of designing artificial neural network for communication systems so that errors can be avoided in satellite data transmission. This sparked a motivation to gather more information in this arena.

Previous literatures to implement SNN using several methods such as the one using MATLAB, Verilog and CMOS by Di Hu et al [5] and the hardware and software implementation methods in works [5] - [21], shows clearly the level of complexity involved in creating SNNs. Thanks to the reliable technological developments and improvements of today that this has been made easier for today's generation of researchers. Among the several literatures to apply neural networks in improving satellite systems, such as image classification, sensory improvements and activations, navigation systems, intelligent propulsion systems, prediction of satellite drag, optimized routing, etc., this literature stands out in its simplicity for creating a fully functional SNN for link selection using satellite downlink data. The literature involving cognitive network controller by R.Lent et al. in [32] provided a wider perspective to use SNNs for this application..

This literature explains in detail the method to use the NengoDL framework to create a spiking neural network, perform testing and preform optimization and discusses the performance of the NN in terms of metrics such as training time, accuracy, an impact of certain features on these metrics. The main contributions in this thesis:

1. Firstly, creating a spiking neural network model for intelligent decision making in predicting the best downlink for a given set of links in a satellite scenario.
2. Secondly, proving that the spiking nature of the neural network predicts with better accuracy when compared to traditional models.

1.4. Thesis Organization

1. Chapter 2 discusses in detail, the working of neural networks, major frameworks involved in this literature and how the neuromorphic architecture is built, and the parameters associated with it.
2. Chapter 3 summarizes the previous works in the implementation of SNNs via hardware and software implementations.
3. Chapter 4 describes in brief, the case under study the important terms involved for link budget such as SNR, buffer occupancy, Service time, response time, with necessary formulas and expressions.
4. Chapter 5 Describes the STK simulation tool and its operations for satellite communication and data extraction.
5. Chapter 6 explains the method to create SNN step by step using clear snippets of code and the experiments conducted.
6. Chapter 7 Shows the results of each experiment and detailed analysis of the impact of data on the time, accuracy, loss, and other metrics.

7. Chapter 8 concludes this thesis and describes suggestions for potential future works in this area.

2. SPIKING NEURAL NETWORKS

2.1. Neural Networks: a brief insight

Artificially intelligent systems use neural network to make decisions and perform predictions. A neural network is formed by connecting several neurons together in form of layers. The output of a neuron is acts as a potential that activates the next neuron. Neural Networks (NN) are computing systems that correspond to the biological neural networks in a human brain. Communication between neurons happen through connection between the neurons called synapse. The neurons that send signals are called presynaptic neurons, and the ones that receive signals are called postsynaptic neurons. The signals can be processed by the postsynaptic neurons and signal the downstream neurons connected to them [24]. Unlike conventional artificial neural networks used in ML, a spiking neural network (SNN) stands out. SNNs work using spikes, the discrete events that occur at different time instances.

2.2. States of neuron

Neurons may have states which typically represented by real numbers between 0 and 1. Each synapse has a weight, a number that controls the signal between presynaptic and postsynaptic neurons [24]. During simulation of SNN, a stateless behavior is set to the neurons for making the study simpler. The processor is the activation function of the neuron to fire.

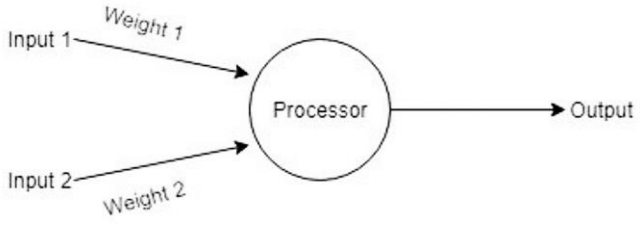

Figure 2.1 The perceptron; Source: [24]

2.3. Membrane potential of a neuron

A spike's occurrence is explained by differential equations that represent various biological processes. Membrane potential is the most significant aspect of neural activity. Basically, a neuron spikes once it reaches a certain potential, and then the potential of that neuron is reset.

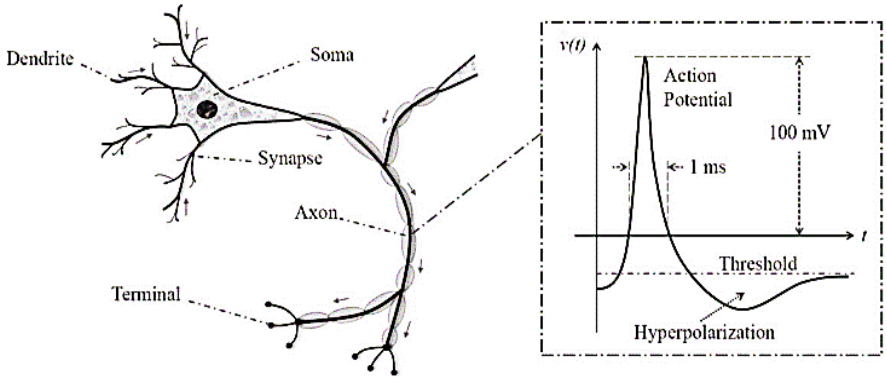

Figure 2.2 Structure of a neuron. Source: [38]

To model a spike event, the post synaptic potential is obtained as a weighted synaptic efficacy of the terminal, which is the pre-synaptic terminal [26]. The pre-synaptic neuron input where i ϵ I and neuron j ϵ J is given as the sum of all synaptic contributions given in Equation (1):

$$y_i(t) = \sum_{k}^{m} y_i^k(t) \qquad (1)$$

Similarly, the post synaptic input can be represented as given in Equation (2) below:

$$x_j(t) = \sum_{i} \sum_{k} w_{ij}^k y_i^k(t) \qquad (2)$$

As per neurotic conventions, the synaptic terminal k is associated with weight w_{ij}^k. When the above value crosses the threshold ϑ, the firing time t_j of neuron j [26].

2.4. Neuron model and working of NN

Each spiking neuron keeps up a value representing its current state. The neurons accept the input spikes and integrate the corresponding weights to update their states. The neuron is activated when

the state reaches a threshold value, and outputs a spike to the next neuron and after a pre-configured propagation delay, the successors receive the spike and update their states accordingly[6]. Though these operating principles are common, depending on the neuron model, the neuron behavior changes. The most common neuron model is the Leaky integrate-and-fire (LIF) model (have a look at Table 3.1) for its proven improvements and realistic brain-like behavior. LIF neurons are inspired by the membrane voltage leaking in biological neurons, with their states decreasing over time if no input spikes present [6].

[4] In 2003, E. Izhikevich proposed a simple neuron model having two equations and one nonlinear term. Starting from here the works of various authors from different parts of the world has been analyzed according to its relevance in this section. The least mean squared error function for the target algorithm learn the desired firing times $\{t_j^d\}$ at the output neurons. If the corresponding input patterns are denoted by $\{P[t_1, t_2, \cdots t_i]\}$, the following represents the error equation, the desired and actual firing times are $\{t_j^d\}$ and $\{t_j\}$ [26].

$$E = \frac{1}{2}\sum_j \left(t_j - t_j^d\right)^2 \tag{3}$$

The following equations represent the calculations necessary for error propagation [26][27][29], or in other words known as back propagation to update the weights, thus the change of weight is as follows:

$$\Delta\omega_{ij}^k = -\eta \frac{\partial E}{\partial w_{ij}^k} \tag{4}$$

Where η is the learning rate and ω_{ij}^k is the weight connecting neuron i to neuron j with delay d_k[26][27]. The above computations are inbuilt in NengoDL for immediate usage in functional ways and classes. Now that the basic firing of a neuron in an SNN is understood, the building of neural network comes into picture. In a neural network, input layer, hidden layer, and output layer

are associated. A neural network starts with input layer, followed by few hidden layers, and finally output layer. The number of layers at each layer is decided using several trials and errors for the data extracted from each scenario under different circumstances.

2.5. Why SNN?

The SNN doesn't necessarily have to know each piece of information associated with the data. Since their computational speed is very high and their capacity for working with uncertain data is high, spiking neural networks is a great choice for solving the problem [3]. Further, SNNs are capable of learning temporal data on realistic terms, i.e., the SNN works well with unsupervised data also, but in this thesis, partial supervised learning is done for proper evaluation of the model. SNNs are event based thus enabling it to be a good choice to be applied on satellite systems, where uncertainty is the most, and analyze the performance of the SNN and the system itself. This can be installed to any other such environment and the SNN would learn to make decisions based on its knowledge associated to its environment. Applying SNNs to satellite systems would enable new prospects of autonomous intelligent space systems.

2.6. Assumptions

Assuming that a bundle is a file or set of packets and the uplink transmission is complete, multiple routes are available to dispatch a given data bundle and they are accessible by the system, a best route or link can be selected to dispatch the data based on a few features describing the link. This happens regularly during routing, except that in this thesis, this decision is made by a neural network instead of a rigid algorithm incapable of adapting to uncertainties. Secondly, assuming to begin with an ad-hoc network of 1 input layer with 16 neurons, 1 output layer with 4 neurons and 1 hidden layer with 32 neurons, further experiments are conducted to find a good model. Finally, the BER is considered as a representation of the quality of the link and buffer occupancy (the

number of bundles waiting to be transmitted) is considered as the traffic in that link as the features to determine the inputs to the system. These two attributes are used to obtain a relevant cost to each link to obtain the best link and route them. This also provides us an option of implementing a distributed service instead of centralized behavior, but that is a whole other region to explore for the future.

2.7. Use of Artificial neural networks in space

A work by Hou et al, proposes link planning for LEO satellites to reduce the number of link switching, by using relay satellites and Inter satellite links (ISLs)[30]. A lot of satellite attributes such as orbital height, orbital inclination, location of GS, were considered and a main assumption was that, the transmission was already complete[30]. Routing is a crucial process that has a significant impact on the network's performance in modern communication networks. Ideally, routing algorithms include finding the best path(s) between source and destination router, steer clear of packet losses and allowing high-speed data transmission. In space, communication traffic is subject to high variability, nonlinearity, and unpredictability, thus making the routing policy a very cumbersome task [3]. Under certain assumptions, the optimal routing may be considered as, the first hop selection of the available shortest path (SP) computations. This thesis provides evidence that use of SNN aids accomplishing more human like behavior provided availability of enough data and based on the scenario at hand. For instance, in [33] Zhenyu Na et al. propose a distributed routing strategy focusing on LEO satellite to enhance the speed of training targeting the traffic prediction accuracy. They achieved this through extreme machine learning. [41] focuses on applications such as anomaly detection in telemetry data, flexible payload optimization, interference detection and classification, and other IOT services involving large amounts of data using AI or Machine learning. In such applications, the use of SNNs can produce attractive results

considering its spiking behavior. Another challenge that can be addressed is the satellite downlink replanning problem as mentioned in [45] for effective communication between satellites, which is a more sophisticated approach. Dynamic power allocation architectures for HTP satellites is another such area where AI was applied to set reward functions for minimizing the unmet system demands and power consumptions [46]. Thus, from above examples we can wee the wide areas for AI application in space.

2.8. Keras

For implementation of the SNN, Keras is the first step for the specific method used in this thesis. It is an open-source software library with python interface for building artificial neural networks. It is an interface for the TensorFlow (TF) library which is also open source for machine learning tasks. TF can be used as an interface for deep neural networks having lucid ecosystem of tools and resources for several data based and statistical applications.

2.9. NengoDL

A special python package called the Nengo Brain Maker is used for building, testing, and deploying neural networks. NengoDL is a simulator specific to simulating Nengo models. In other words, it accepts as input a Nengo network, and enables the user to simulate the network with the help of certain underlying computational framework (in this case, Keras: TensorFlow). The classes available to use in NengoDL is designed in such a way that it is similar to the usage of libraries in TensorFlow. One can Build a Nengo Model with basic understanding of TensorFlow model. NengoDL enables a user to convert a regular Keras model to a Nengo model and apply spiking activities so that the people experienced with TensorFlow are benefitted with easy adaptation to spiking networks. It makes easier to apply and explore spiking activities using special activation

functions and neuron types, setting firing rates and applying synaptic smoothing for obtaining better plot curves.

In the next chapter we will see how other authors all over the world have implemented SNNs in detail, and which of the methods would work for this application.

3. LITERATURE REVIEW

3.1. Prior works

Understanding back propagation is important while studying neural networks. While applying BP is difficult for a NN containing spiking activities, Nego provides this functionality built within so that learning is smoother. The approximated chain rule approach by the authors of [26] provides detail exegesis of back propagation implementations with spiking neural networks. Considering from Equations (1) - (4), the approximated chain rule is implemented as follows:

$$\frac{\partial E}{\partial \omega_i^k j} = \frac{\partial E}{\partial t_j}\frac{\partial t_j}{\partial w_{ij}^k} = \frac{\partial E}{\partial t_j}\frac{\partial t_j}{\partial x_j(t)}\bigg|_{t=t_j}\frac{\partial x_j(t)}{\partial w_{ij}^k}\bigg|_{t=t_j} \tag{5}$$

From the equations of post synaptic potential and error equations, the approximated threshold function for firing of a neuron is given as $\delta_{t_j} = -\delta x_j(t_j) / \alpha$, where α equals the local derivative of x_j with respect to $\frac{\partial x_j(t)}{\partial t}\big|_{t=t_j}$. Thus, the second term in the above equation (5) from [25] is expressed as follows:

$$\frac{\partial t_j}{\partial x_j(t)}\bigg|_{t=t_j} = -\frac{1}{\alpha} = \frac{-1}{\frac{\partial x_j(t)}{\partial t}\big|_{t=t_j}} = \frac{-1}{\Sigma_{i,j} w_{ij}^l \frac{\partial y_i^l(t)}{\partial t}\big|_{t=t_j}} \tag{6}$$

$$\Delta w_{ij}^k(t_j) = -\eta \frac{y_i^k(t_j)\cdot(t_j^d - t_j)}{\Sigma_{i,j} w_{ij}^l \frac{\partial y_i^l(t_j)}{\partial t_j}} \tag{7}$$

$$\delta_j \equiv \frac{\partial t_j}{\partial x_j(t_j)}\frac{\partial E}{\partial t_j} = \frac{\left(t_j^d - t_j\right)}{\Sigma_{i,l} w_{ij}^l \frac{\partial y_i^l(t_j)}{\partial t_j}} \tag{8}$$

For error propagation apart from the output layer (J), the generalized delta error in layer I after applying the approximated chain rule.

$$\delta_i = \frac{\partial t_i}{\partial x_i(t_i)} \sum \delta_j \frac{\partial x_j(t_j)}{\partial t_i} = \frac{\Sigma_j \delta_j \left\{ \Sigma_k w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_i} \right\}}{\Sigma_{h,l} w_{hi}^l \frac{\partial y_h^l(t_i)}{\partial t_i}} \quad (9)$$

Thus, for hidden layer weight adaptation value reads as follows. An approximation of post-synaptic potential $x_j(t)$ is made

$$\Delta w_{hi}^k = -\eta \frac{y_h^k(t_i) \Sigma_j \left\{ \delta_j \Sigma_k w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_i} \right\}}{\Sigma_{h,l} w_h^l i \frac{\partial y_n^l(t_i)}{\partial t_1}} \quad (10)$$

The authors have represented the spike times in milliseconds where the third neuron always fired at t = 0 for reference starting time. [26] During training, the result showed that the network learned the XOR pattern with learning rate equal to 0.001 within 500 epochs, while the same rate required just about 10 to 60 epochs to reach more than 80% prediction accuracy in our model. The number of epochs for the current thesis is less depending on the amount of data and features selected. For the second part of the testing, they used an extrapolated XOR function with 6 hidden neurons, 3 input and 1 output neuron, which is slightly different in the current thesis. The study of [26] provided wide perspectives and abundant knowledge to understand how back propagation could be applied to a spiking neural network for improving the learning. Have a look at Table 3.1 for a detailed analysis of various works on creating SNN for different applications.

Table 3.1 Background information of various works based on SNNs

Paper	Description		Implementation Technique
[1]	Technology	FPGA	Using "Reject output" to separate the training data set into two parts for classifying. The term "Reject output" is used for separating all indiscernible patterns from the training data set. The CNN has 10 outputs.
	Application	Image recognition	
	Model	LIF	
[5]	Technology	MATLAB, Verilog, and CMOS.	To find a random target from a map, an indirect training algorithm, implemented in MATLAB, Verilog and CMOS are used to train SNN. Virtual insect is based on a re-scalable neural network having one input, output and hidden layer each.
	Application	To train a virtual insect to navigate through a terrain with obstacles from the environment	
	Model	spike timing-dependent plasticity (STDP)	
[6]	Technology	Xilinx ZC706 evaluation board with a XC7Z045 SoC to build the test platform	Use of hybrid of conventional time-stepped updating algorithm and event-driven updating algorithm. Signed 16-bit fixed-point numbers are

	Application	Classification task on the MNIST dataset.	used to represent the weights and neuron states. Maximum of 1024 neurons for a layer, with fully connected synapses.
	Model	LIF	
[8]	Technology	Xilinx Virtex-5 (xc5vlx85ff676-3) FPGA device.	Reduced precision method is analyzed for training and inference of SNN. SNN is first trained and used with the conventional 32-bit single-precision format. Statistical analysis of network weights, activations, and gradients is performed. By analyzing the data distribution, the possible minimal numerical format is determined. Then SNN accuracy is tested and computations are done.
	Application	classify the handwritten digits in MNIST database.	
	Model	LIF	
[9]	Technology	None mentioned	Algorithm 1. ML training using online SGD.

	Application	A review of models for probabilistic SNNs within a probabilistic signal processing framework	Algorithm 2. ML training by online doubly SGD.
	Model	Probabilistic model: generalized linear models for SNNs, known as the spike response model with escape noise.	
[10]	Technology	None mentioned	The synaptic transistors and neuron circuits are fabricated in different wafers, and the hardware SNN simulation is studied after modeling the behavior of each building-block. To recover clean images from noisy images, a NN is trained using noisy train images and mean squared error (MSE) loss function. The trained NN is converted to the hardware SNN.
	Application	Pattern recognition using synaptic transistors and neuron circuits	
	Model	LIF model	

[11]	Technology	Software based learning followed by hardware-based recognition implemented.	Created neuro-synaptic network, used LIF neuron model, analyzed synapse's response to a pre-neuron's spike (that will be proportional to the synapse's strength) and obtained a weight update rule.
	Application	Fisher Iris flower classifier	
	Model	LIF	
[13]	Technology	Xilinx XC2V1000 FPGA	Merging NoCs (network on chips) with programmable spiking neurons. Use of a 2-D array of interconnected neural tiles surrounded by I/O blocks. Neural tiles are connected forming a nearest neighbor connect routine. An SNN was analyzed by programming the tile functionality and connectivity.
	Application	Spike packet routing	
	Model	Not mentioned	
[14]	Technology	Intel core 2 Quad Q6600 CPU (2.4 GHz) and GeForce 8800 GTX graphics hardware.	Used GPU language CUDA (compute unified device architecture) recently released from NVIDIA, to create feature extraction module for the NNs using OpenMP (Open Multi-Processing)
	Application	neural networks-based text detection system	

	Model	Fork-join model OpenMP	
[15]	Technology	Intel Xeon CPU	The whole flow consists of two types of programs, the first type is only run once, to calculate gray level, or doing normalization, and calculating value at the field of reception, image intensity is integrated. The second type of program was executed at every time step to simulate the dynamic behaviors of spiking neurons.
	Application	SNN model for segmentation of color image (blood smeared images)	
	Model	Conductance-based integrate and-fire neuron model used to simulate neurons in the network	
[16]	Technology	Not mentioned	First, both models are trained with the learning rule for 800 epochs with a constant rate $\eta = 0.05$, depending on 20 trials with varying random seeds. Then the Bayesian learning rule is

	Application	Classifying a handwritten number and detecting possible rotation of the image of the handwritten digit	considered and the impact of the prior prediction values is studied on overfitting.
	Model	probabilistic model is introduced for a generalized set-up	
[17]	Technology	GPU device Radeon HD 7970 using OpenCL framework	2 major computation segments: Update and propagation. Update feeds synaptic events into SNN, propagate solves neuron model equations for each neuron and generates spike events;
	Application	Performance analysis	
	Model	Izhikevich	
[18]	Technology	Intel Xeon E5-2680 CPU and NVIDIA Tesla K40C GPU	The inputs to the algorithm are the trained network and the spike trains representing the input. The algorithm evaluates the SNN and produces an output class label. SNN is iteratively
	Application	SNN evaluation	

	Model	LIF	evaluated for each timestep. Finally, the class label corresponding to the output neuron having the maximum spike rate is chosen.
[19]	Technology	CARLsim 24 CPU cores and 8 GPUs	A CARLsim simulation occupying CONFIG, SETUP, and RUN states at execution is done. An 80-20 random spiking network implementation is done using 2 CPUs and 2 GPUs.
	Application	SNN simulations	
	Model	Izhikevich and LIF spiking neuron models	
[20]	Technology	Spartan6 FPGA	4-layer ConvNet trained for recognition of poker card symbols has been implemented in a Spartan6 FPGA, including 22 convolutional nodes and a decision block. Different strategies for the decision block are analyzed.
	Application	Observing a deck of 40 poker cards and determine the kind of card (hearts, spade, clubs, or diamond)	
	Model	LIF	

[21]	Technology	Verilog-HDL, NANGATE 45nm technology library	Using the datasets of Inverted Pendulum and Wisconsin dataset, for latency comparison. Seven input/output port modules for each direction in addition to the Switch-Allocator was used, and the Crossbar module which handles the transfer of spikes to the next spiking neural processing core was used.
	Application	Evaluating hardware based SNNs	
	Model	Not mentioned	
[22]	Technology	SpiNNaker 102 board	Comparing scaling potential of SpiNNaker to that of a PC. The experiment helps users of SpiNNaker dividing graphs for mapping maximum number of cores and utilize hardware capabilities to the complete extent.
	Application	graph processing applications for transfer of data packets	
	Model	LIF	
[23]	Technology	NEURON, GENESIS, NEST simulation tool, NCS, CSIM, XPPAUT, SPLIT, Mvaspike	Evaluating various kinds of simulation techniques and algorithms presently implemented and the precision of those in which plasticity

	Application	review different aspects of the simulation of spiking neural networks	is based on the exact timing of the spikes.
	Model	Hodgkin-Huxley type, IF models, using event-driven integration strategies	

3.2. Inference from prior works

From the above table it is clear that most of the applications makes use of MNIST data for classification in [6], [8], [11], [14], [16] and [21], and most of them involve complex implementation methods and structures created on simulation platforms [23] and on chips like FPGAs [1], [6], [8], [13], [20] and 94[22] Other software based implementations include MATLAB, Verilog HDL like in [5] and [21], and [11]. Although access to some tools is available, coding in Verilog HDL happened to create more difficulty. Use of CPUs and GPUs for implementing SNNs significantly exude more complex structures and [14], [15], [17], [18] and [19] appear cumbersome for immediate application or deployment. Further, such implementations require ample prior knowledge and time to complete implementation successfully. Evidently, from Table 3.1 we can see that the number of input and the output neurons depends on the features of the data considered for inputs and the response required. Far from classifications involving SNNs to understand types of flowers and digits [11], works like [31] seem to us as eye openers to the ways and several applications in which neural networks can be applied, which can be used but there isn't sufficient details in these papers to implement this idea. All the prior works studied

above provide ample information regarding creating SNNs and using them to make intelligent systems. These works appear to shed very little light on applying SNNs to space operations. Unlike [43], instead of identifying satellite downlink based on elements such as memory filter and high power amplifier, in this thesis, I will be using BER for link quality and link occupancy to deduce the response time (cost). This thesis covers in detail the challenges in space communications and the need of SNN based AI in the same. Many path selection algorithms are available such as in [44], which provided a wide outlook on such methods involving neural networks, but the neuronal activities in most works lack spiking activities. Secondly, this thesis proposes the use of NengoDL for immediate deployment and integration as neuromorphic chips consuming comparatively less power than CPUs and GPUs. SNN created through this framework can be integrated into Intel's Loihi neuromorphic chip which is proven to consume very less power and energy as shown in Table 3.2. This is the reason for emphasis on use of SNNs.

Table 3.2 Average energy per inference and power consumption chart. Source: [39]

Hardware	Running (Watts)	Idle (Watts)	Dynamic (Watts)	Joules per inference	inference per sec
CPU	28.48	17.01	11.47	0.0063	1813.63
GPU	37.83	14.97	22.86	0.0298	770.39
Intel's Loihi	0.110	0.029	0.081	0.00027	296

From above, we can see that SNNs are event driven and can be implemented with less power. First one is Von Neumann architecture. The neurons are evaluated in the central processing unit while the synapse weight information and neuron outputs are stored in the RAM. The transfer between CPU and Ram for the data is the challenge in this case, which limits the speed at which the entire

network can be evaluated. Secondly, GPUs or graphic processing units, due to their capability to perform large scale matric multiplication operations, have been able to improve the speed of the neural networks. The issue here is the higher power consumption. This could make the CPU and RAM operations look simpler, but their hardware doesn't resemble that of a neural net. To resolve this, development of neuromorphic hardware to accelerate the speed of the networks and power consumption has become essential for next gen systems. Hence, seeing from the performance of SNNs so far, we can say that designing a SNN for a space application would be a better choice when compared to CPUs, GPUs and traditional ANNs known by the ML community today.

In the net chapter, we will look at the principles of satellite communications necessary to understand that falls within the scope of this thesis.

4. PRINCIPLES OF SATELLITE COMMUNICATION

4.1. Satellite Link Budget

This section contains a brief description of the considered physical layer attributes for the satellite link. Satellite communication system consists of two types of links: Uplink and Downlink. The satellite link budget involves relatively simple calculations. Certain non-trivial concepts could also be taken into account. On highly professional grounds, all the second order effects are to be considered that would affect the link budget, of which most are skipped in this case. The most important few attributes are considered in this case for computing the link budget. Satellite communication is wireless communication through Radio Frequency link, which includes free space losses and additional losses[49].

Figure 4.1 Wireless Transmission

Hence, at Transmission there may be several things such as data, modulators, encoder, power amplifier, etc. Here considering it a black box, what comes out of this is a power P_t (Transmitted power) which is fed to an antenna with a gain G_t. Similarly at the receiver, there will be a series of things like filters, amplifiers, down Converter mixers, etc., which when seen as a black box, the input has a Received power P_r, coming from the reception antenna with a gain of G_r. Between the two antennas, the wireless communication takes place with the Free space lowering the intensity of the signal.

4.1.1. Free space losses

[49] The increasing intensity of the signal due to the distance between the two stations, goes with Quadratic law. The intensity of the signal received depends on square of the distance between transmitting and receiving antenna. According to quadratic law, if the distance is twice as far, the power flux reduces four times. This issue persists for all wireless communication and not particular only to satellite communication.

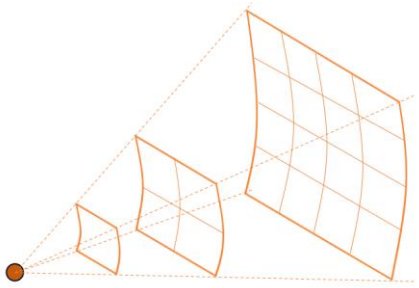

Figure 4.2 Quadratic law

Transmission in satellite communication, means that the transmitter performs sends section of the payload onboard of the satellite with the corresponding transmitting antenna pointed towards the earth. There are several types on antennas but one thing to remember is that they do not create energy. They only transmit and receive by focusing the collected power on one focus point. The directive gain of the antenna is the ratio of power density of antenna in one direction to that radiated by an ideal isotropic antenna (emits uniformly in all directions) radiating the same total power. This depends on the direction the antenna is pointed. For simulation, we enabled the constant pointing by the use of targeted option on the sensor properties. The highest value of the directive gain is given by D, which is the directivity of the antenna[49]. Evidently, two angles need to be defined that would render the position w.r.t an ideal sphere that is a certain distance away from the antenna. Thus, the direction at which the antenna is focused renders energy at best. The directivity is rendered in dBi as we are comparing with isotropic antenna [50].

$$D(dB_i) = 10\log_{10}\left(\frac{\text{Radiated power density in one direction}}{\text{Isotripically radiated power density}}\right)$$

$$D = \frac{4\pi A_e}{\lambda^2}$$

Where A_e is the equivalent area or effective surface of the antenna and λ is the carrier electromagnetic wavelength, depicting that higher the wavelength, lower will be the frequency of radiation and smaller will be the directivity. The effective surface may not be the physical surface are of the antenna, and this carries the properties of the antenna. If we have an aperture This attribute in handled within the simulation for obtaining the link budget. The gain is not same as directivity but can be calculated as the efficiency times directivity. As available for dipole antenna the way the radiation pattern recedes with distance thus resulting in total power flux dropping, we are unaware of such representations for general patterns.

$$G = \eta_{rad} D$$

4.1.2. Equivalent Isotopically radiated power: EIRP

[49] It is assumed that we have an ideal isotropic radiator which is equivalent to the antenna under consideration in the given direction. This is defined as the power with which an isotropic antenna would have to emit to have the same power flux given as follows:

$$EIRP = GP_t$$

Considering two antennas, one for emission and one to receive, with the knowledge of distance between the antenna and the receiver the power can be obtained.

Considering emitted total power P_t, The EIRP is known from above equation, so, a distance x away from it, the power would have dropped by $4\pi x^2$. Therefore, the power received is obtained as follows:

$$P_r = \frac{EIRP\ .\ A_e}{4\pi x^2} = \frac{P_t G_t}{4\pi x^2} A_e$$

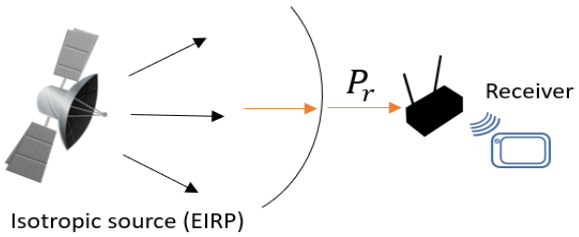

Figure 4.3 EIRP scenario

This can be represented in terms of gain as a function of wavelength:

$$P_r = \frac{P_t G_t G_r \lambda^2}{4\pi x^2}$$

Hence, converting to dB as it is more preferential, we obtain the total received power in dBW as the FRIIS question for propagation.

$$P_r(\text{dBW}) = P_t(\text{dBW}) + G_r(\text{dBi}) + G_t(\text{dBi}) - 10\log_{10}\left(\frac{4\pi x}{\lambda}\right)^2$$

With a link receiving from a Tx, the received power depends on the power transmitted, gains of the transmitter and the receiver and is counter proportional to its losses [50]. The last term denotes the free space path loss.

4.1.3. Additional Losses

From the above FRIIS equation, all the losses can be subtracted easily as shown above. Conventional practices follow adding all the gains (improving the signal) and subtracting the losses (those deteriorating the signal).

a) Free space propagation loss

b) Attenuation loss

c) Polarization mismatch

d) Interference, etc.

It is not possible to describe the quality of signal just using the absolute power value. The noises are to be included while understanding the signal quality as mentioned in the propagation equation.

4.1.4. Signal to Noise Ratio (SNR)

In previous section, the several types of noises were introduced. These noises can be quantified into an expression in terms of an effective temperature, which is an equivalent temperature taking all these noises and recreate their power in terms of thermal noise. Sure, thermal noise by itself stands out as a type of noise, but even the other sources of noise could be obtained by mathematically substituted as Nyquist noise formula with an equivalent temperature.

$$N = k_B T_{sys}$$

Where N is the noise spectral density (NSD) describing the power of the noise per unit Bandwidth (BW), k_B is the Boltzmann's constant and T_{sys} is the effective temperature of the system. This formula is called as the Nyquist noise formula. T_{sys} includes all the noises and is a parameter of the system as a whole. To obtain the full power, the BW is multiplied with this value. Hence, the total received power from FRIIs equation, divided by the NSD times the BW, renders the carrier to noise ratio (C/N)

$$\frac{C}{N} = \frac{P_t G_t G_r \lambda^2}{(4\pi x)^2 L_a k_B T_s B}$$

Energy per bit per unit spectral density for the noise is Eb/N0 is a quantity that is directly connected to the error rate. The Eb is obtained by dividing the previous expression by transmission rate and to obtain N0, the previous expression is multiplied by BW.

$$\frac{E_b}{N_0} = \frac{\text{energy per 1 bit}}{\text{Noise Spectral Density}}$$

$$\frac{E_b}{N_0} = \frac{P_t G_t G_r \lambda^2}{(4\pi x)^2 L_a k_B T_s R}$$

$$\frac{E_b}{N_0}(\text{dB}) = G_t + P_t + \frac{G_r}{T_s} - L_s - L_a + 228.5 - 10\log_{10} R$$

The term G_r / T_s represents the gain per system temperature of the receiver in dBK and 228.5 stands for the Boltzmann's constant in dB.

4.2. Satellite communication example

Consider the following: A GEO satellite link with transmission rate of 100Kbits/sec with receiver gain 35dB, transmitter gain 20dBi. If the satellite is assumed to be exactly at the equator and transmission power of 700Watts at 10GHz frequency. (Source: [50])

Ts = 300K | Pt = 700W | f = 10GHz | $\lambda = 1/f$

Thus, $P_t(dBW)$ = 10 $\log_{10}$ (700) = 28.5dBW

$$P_r(dBW) = P_t(dBW) + G_r(dBi) + G_t(dBi) - 10\,log_{10}\left(\frac{4\pi x}{\lambda}\right)^2$$

$$= 28.5dBW + 20dBi + 35dB - 10\,log_{10}\left(\frac{4\pi * 36000}{1/10GHz}\right)^2$$

$$= -229.61\ dBW$$

$\frac{G_r}{T_s} = 35 - 10\log_{10}(300) = 10.2$dBK

$$\frac{E_b}{N_0}(dB) = 20 + 28.5 + 10.2 - 203.5\ (Ls) - 5\ (La) - 228.5 - 50$$

$$= 28.7dB$$

$$\simeq \mathbf{29dB}$$

From the above value is it clear that the quality of the signal is very good. Using this calculation, we can deduce the associated BER suiting the type of modulation used. The BER is a probability of error in the transmitted data.

4.3. Link budget conventions

1. Compute or evaluate the transmission power – Pt and obtain the gain Gt in the direction considered.
2. Calculate the free space path loss for the frequency under consideration and thus estimate Ls. It is essential to note that the distance varies according to which orbit the satellite currently is.
3. Additional losses such as polarization mismatch, loss due to satellite drag, atmospheric losses, etc. need to be estimated. These losses are subject to change according to location or seasons.
4. Compute the system temperature Ts and the Rx gain Gr. Obtain the Gr/.Ts ratio in dBk.
5. Obtain the BER based on modulation and compare the calculated Eb/N0 with the one required for the desired BER by calculating the link margin.
6. To obtain a positive link margin, modify the parameters such as EIRP, Pt, etc.

4.4. Bit Error Rate (BER)

The BER value is a measured as the number of error bits received per unit time. If X bits were transmitted, and had x error bits in them, the ratio of x/X is the BER, which is a unitless measure. The probability p_e is the expectation value of the BER. The bundle error ratio can be calculated similarly as follows[34]:

$$p_b = 1 - (1 - p_e)^L$$

where L is the Data bundle length of L bits. The BER values were obtained through simulation for time series data covering 24hrs of time. The BER describes the link quality inclusive of all the losses. From [35] evidently, the probability of error can be obtained in terms of Q function expressed as using Energy per symbol to noise spectral density as:

$$P(\text{Error}) = Q\left(\sqrt{\frac{2E_s}{N_0}}\right) = 0.5\,\text{erfc}\left(\sqrt{\frac{E_s}{N_0}}\right)$$

For synthetic data, Obtaining BER was the most challenging task. For obtaining the BER, the following procedure was adopted. Firstly, the simulation of the satellite scenario was done for both LEO and GEO orbits. Post this, the range of energy per bit to noise power spectral density was obtained through the simulator. This range was used to randomly obtain the Eb/N0 values and corresponding Es/N0 values using the following formula [36]:

$$\frac{E_s}{N_0} = \frac{E_b}{N_0}\log_2(M)$$

Where M = 4 for QPSK channel. This is expressed under the "Reg" column after each Eb/N0 value in the data sheet. These values are randomly generated for obtaining 10,000 synthetic observations behaving like simulated data.

4.5. Cost calculation

Considering a bundle of data is transmitted, this bundle can be viewed as a file of size 100MBytes. Size of one bundle = 100Mbytes | therefore, Size of One bundle = 800 Mbits

$$\text{Probability of 1 bit error in a bundle} = P_e = \text{BER}$$

$$\text{Probability of 1 bundle loss} = p_b = 1 - (1 - p_e)^L$$

Where L is the length of the bundle. If we consider 1 bundle to be file, then L is the size.

$$\text{Tranmsission time} = \frac{\text{Size of Bundle}}{\text{Data Rate}}$$

Propagation time is the time taken to cover d distance in $3*10^8$ m/s (Speed of light). If we consider LTP type of transmission, The Round-Trip time (RTT) is twice the propagation time. Thus, Time Required to transmit the bundle[51]:

$$\text{Propagation time} = \frac{d}{3*10^8 m/s}$$

Bundle time = (Transmission time) + 2*(Propagation time)

$$\text{Link Tx time } = \frac{\text{size of bundle}}{\text{rate}} \times \frac{1}{(1 - p_b)}$$

According to queuing theory, the service time is the time taken to serve a customer. In this case, the time taken to serve one bundle. Occupancy is the number of bundles waiting to be transmitted. The system would have to transmit n such bundles. This time is considered as the cost of that link.

$$\text{Service time} = s = \frac{\text{Bundle time}}{\text{Probability of successful Transmission}} = \frac{\text{Bundle time}}{(1 - p_b)}$$

$$\text{Response time } = (\text{Occupancy} * \text{Average Bundle time}) + \text{ service time}$$

5. DATASET GENERATION

The STK cloud simulation tool is used to simulate satellite communication scenarios to obtain the BER values as no real data is available in case of satellite downlinks. One is GEO and the other is LEO. The link budget obtained from these scenarios are used to obtain a range of values to create another set of larger dataset as synthetic values for experiments. This synthetic data is used to see how the SNN would respond to a large data set of random events. To simulate using STK tool, one must undergo certain necessary trainings to correctly use the STK tool specific to creating communication scenarios. On a generic level, the scenario can be visualized as the following image, refer to Figure 5.1. The satellite being a LEO or Geo satellite makes a selection of the best link for transmission, giving the idea that the reception need not be monitored and controlled manually.

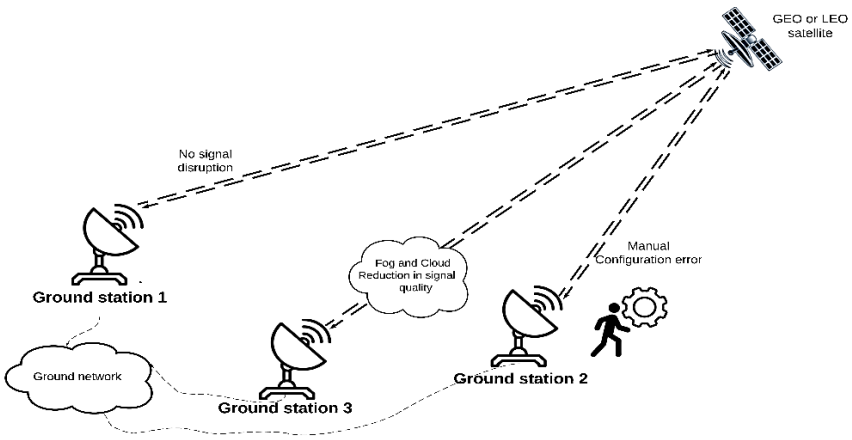

Figure 5.1 Satellite scenario considered

5.1. Simulation procedure

The objects that are used for the scenario created are satellite, transmitter, receiver and sensors, places (locations/facility). All the necessary objects are available to be inserted from the object browser. The properties can be defined by using one of the several methods listed on the right side

and the properties are set. The properties are set such a way that it mimics real satellites transmitters and receivers. Further, the environmental conditions can also be set, such as rain and fog.

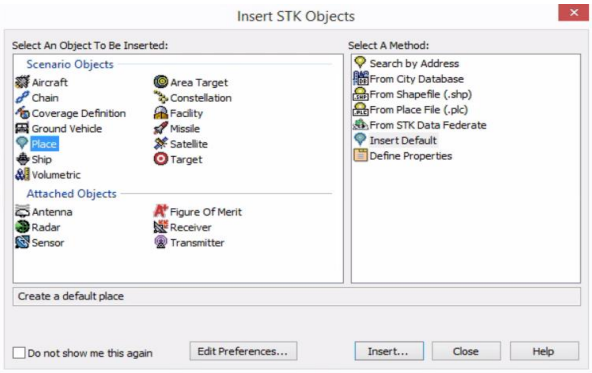

Figure 5.2 STK Objects

5.1.1. Transmitter and receiver properties

The downlink properties were set for the transmitter and the uplink properties were set to the transmitter and the receiver, refer *Figure 5.3*. This scenario assumes the link is established such that the uplink transmission was already done. The type of transmitter is simple transmitter model and that of the receiver is complex receiver model given by STK. The values such as frequency, EIRP, data rate and polarization are set according to the recommended values chart provided by the International Telecommunication Union (ITU)[37]. The following table shows the list of recommended values necessary of simulation from ITU as well as particular to conventional satellite values.

Table 5.1 List of recommended values for satellite communication by ITU

Service	GEO	LEO
Frequency (space-to-Earth) (GHz)	19.92	19.2
EIRP (dBW)	61.8	58

Polarization	Left hand circular	Right Hand Circular
Bandwidth (MHz)	41.84	4.1
Data Rate (Mbps)	51.84	8900
Distance rage (km)	36000	2000
Propagator	J2Perturbation	SGP4

Several other factors can be monitored and set according the ITU recommendations. Once all these parameters are set, each element is provided access to each other element with the help of the "Access" feature for the objects. The transmitter can now access the details of receivers. The "Link-Budget" is one such attribute that can be accessed through navigation to the access page. Similarly, the "Report and Graph Manager" feature of STK cloud enables crating new styles of graphs and reports for the simulated period, for the factors that one is interested in analyzing.

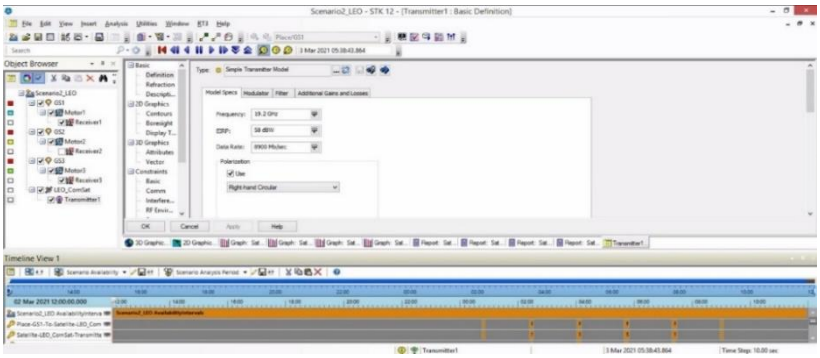

Figure 5.3 Sample Transmitter properties of LEO orbit

Table 5.2 List of common features

Modulation	QPSK
Pointing	targeted
Antenna diameter	0.5 m
Sensor Type	Simple conic (5 deg)

Antenna model type	Parabolic
Transmitter	Simple Transmitter model
Receiver	Complex receiver model

Explicitly, other environmental Losses can be set by selecting respective databases from the STK cloud database. This enables one to add additional noise to the signals transmitted is the location of the places change. The databases contain rain and cloud records from several places and days. A separate link budget was obtained for BER analysis with explicit Rain and fog loss to analyze how the system responds in such cases. From the graphs obtained through simulation, it was observed that the behavior of the signals is the same except there is a change in the range of values.

5.1.2. Space Scenario

LEO satellites are closer to earth at around 2000kms and at the same time provide access to receivers from various parts of the same country. GEO satellites are at about 36000kms away from earth and this mode along with the earth, and hence the name GEO stationary. From Figure 5.8 it is we can see that antenna pattern. It the same for all the receivers provided the features are set with the appropriate values. Yellow lines in the orbit path on 2D graphics show that the satellite mildly busy and the red lines shows that the satellite is accessing all 3 receivers and very busy. The blue line indicates change in satellite is not accessing any of the receivers.

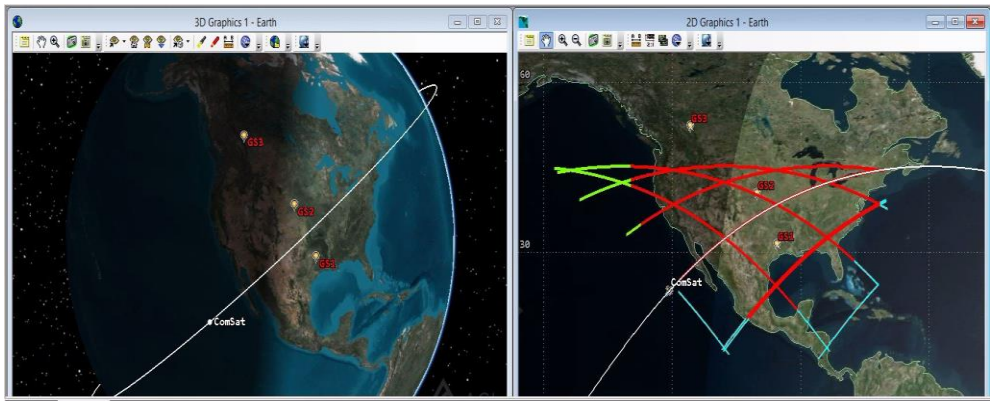

Figure 5.4 LEO Orbit simulation screenshot before contact.

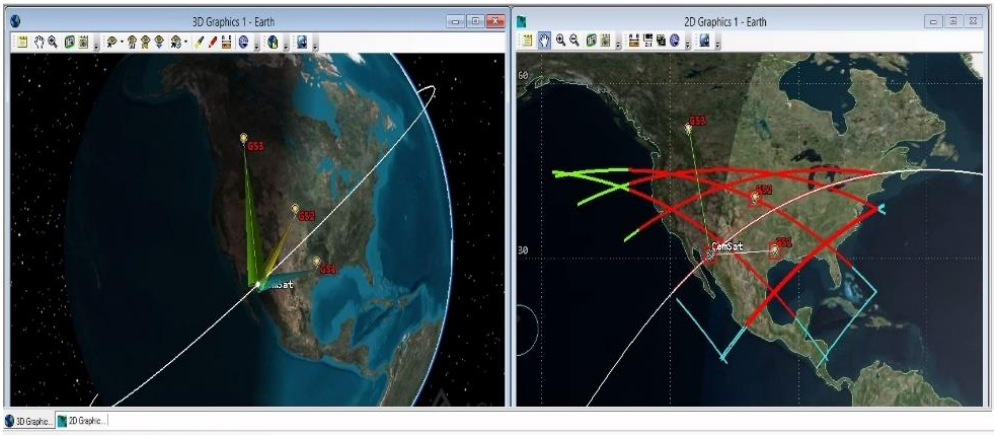

Figure 5.5 LEO Orbit simulation screenshot at common contact.

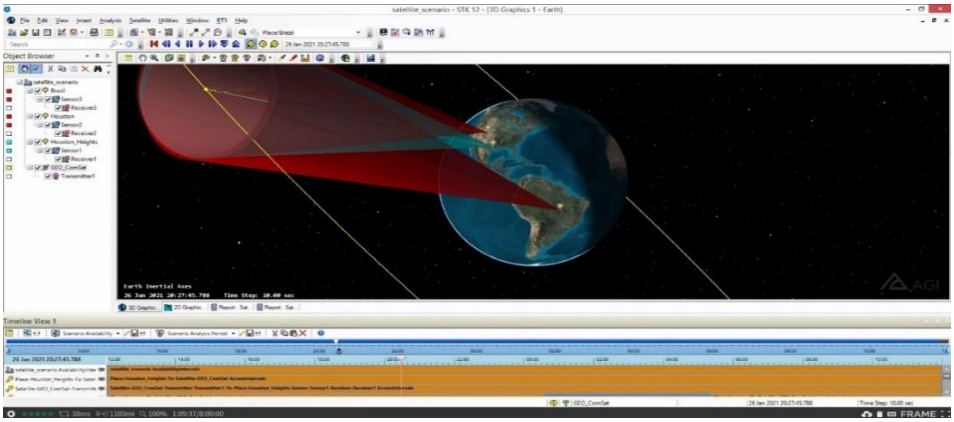

Figure 5.6 GEO Orbit simulation screenshot.

Figure 5.7 2D view of GEO scenario

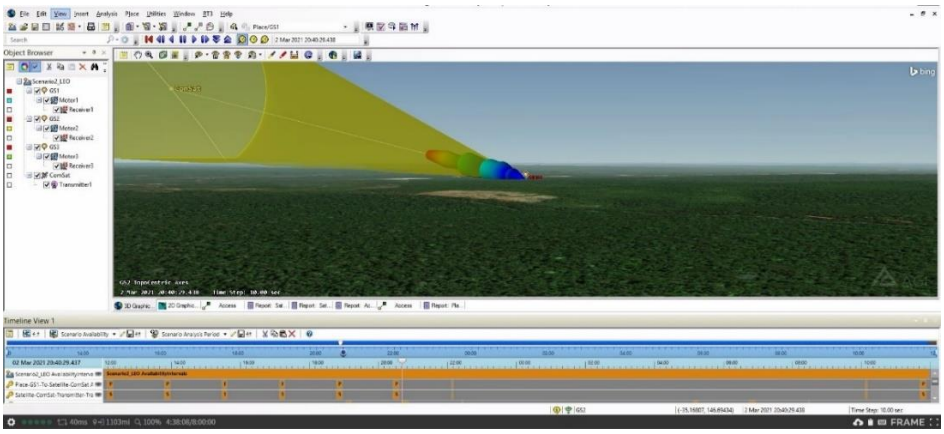

Figure 5.8 Antenna pattern visualization from one of the ground stations.

5.2. Reports and graphs

The “Report and Graph Manager” can be accessed to obtain the desired graphs for analyzing the scenarios, refer Figure *5.9*. The following are the graphs obtained from different scenarios under different circumstances. All the graphs will be generated with time on the x axis and customizes y axes options in STK tool. Explicit RF environment losses can be enabled or disabled by checking or unchecking the model use option mentioned in Figure 5.10.

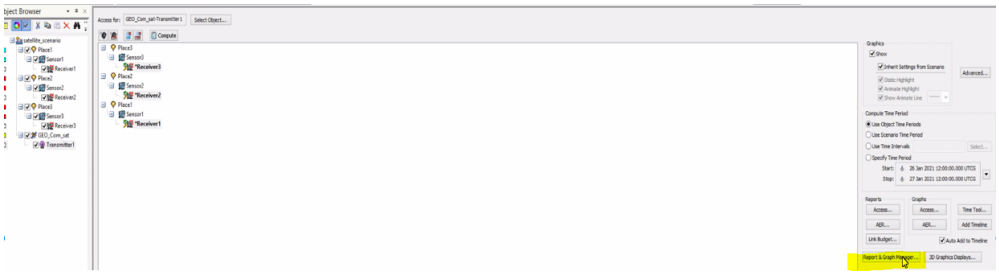

Figure 5.9 Access page for transmitter

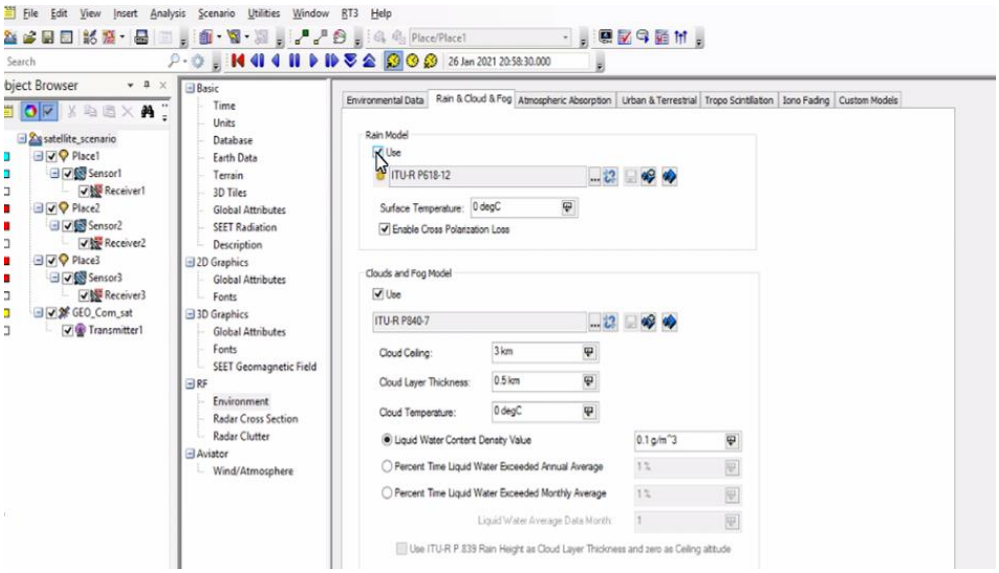

Figure 5.10 Explicit RF environment losses

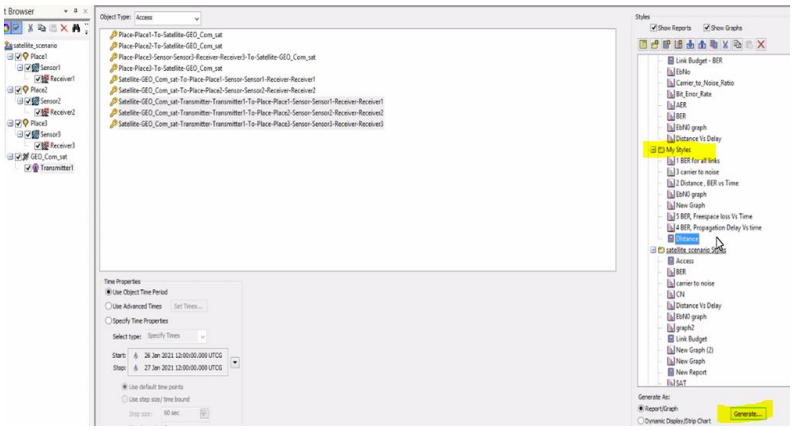

Figure 5.11 Report and Graph manager page

5.2.1. GEO Scenario without explicit RF loss

This thesis analyzes three kinds of data for each scenario. One is synthetic data; one is simulated without explicitly setting the RF environmental losses and finally a dataset with explicit RF environmental losses. Figure 5.12, and Figure 5.13 show the graphs representing how the distance(left Y-axis) and the free space losses(right y axis) vary with time(X-axis) along with BER(right Y-axis and left y axes respectively in each image). Since we have not provided explicit rain and fog loss, it can be observed that the BER is lower and so are the free space losses.

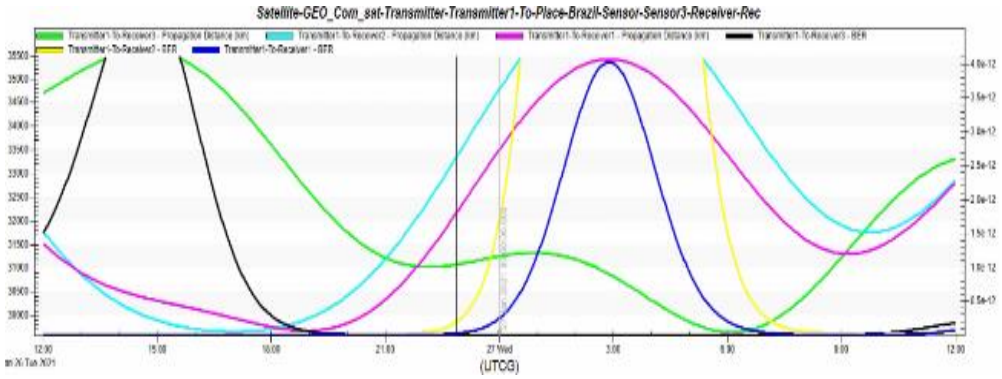

Figure 5.12 GEO - BER and Distance Vs Time

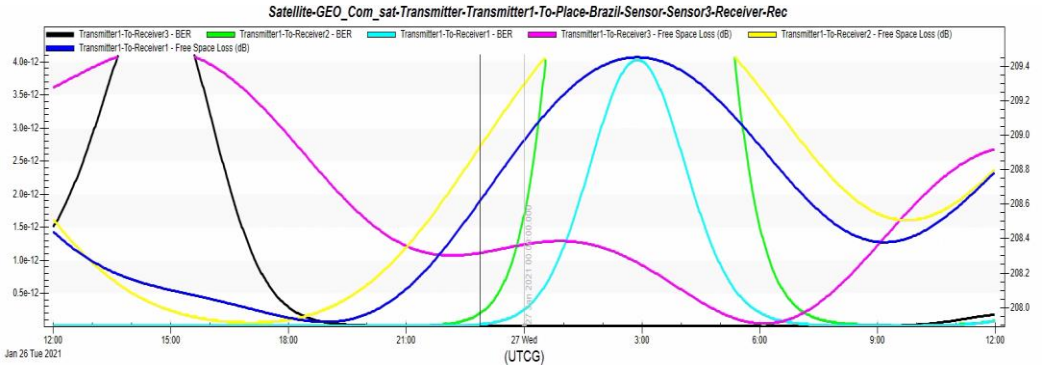

Figure 5.13 GEO - BER and Free space loss vs Time

5.2.2. GEO Scenario with explicit RF loss

The second part is to create the same scenario, but this time we give explicit rain and fog loss to make the links to have poor RF environmental conditions. We can see from the following graphs that BER curves are wider and higher, showing the lower quality of the link and The free space losses in Figure 5.14 show very high levels measured in dBs implicating that the quality of environment is poor due to explicit RF losses. The explicit rain and fog loss recorded in dB can be seen in Figure 5.15. X-axis shows time, left and right y axes show BER and free space loss respectively.

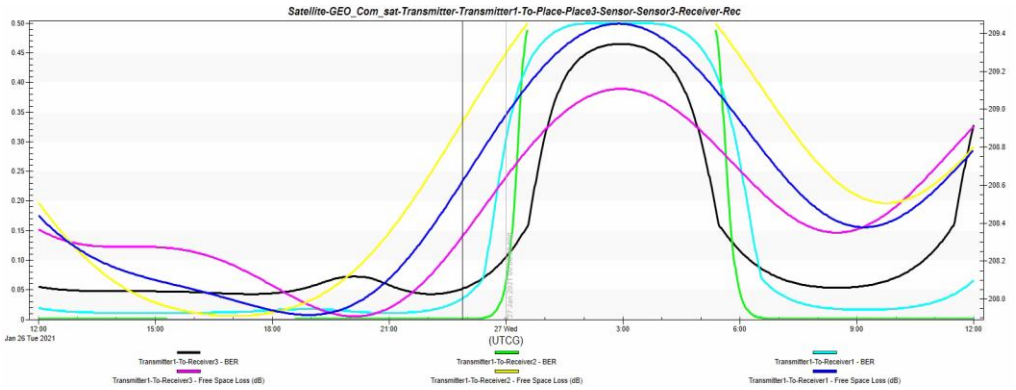

Figure 5.14 BER, Free space loss Vs Time

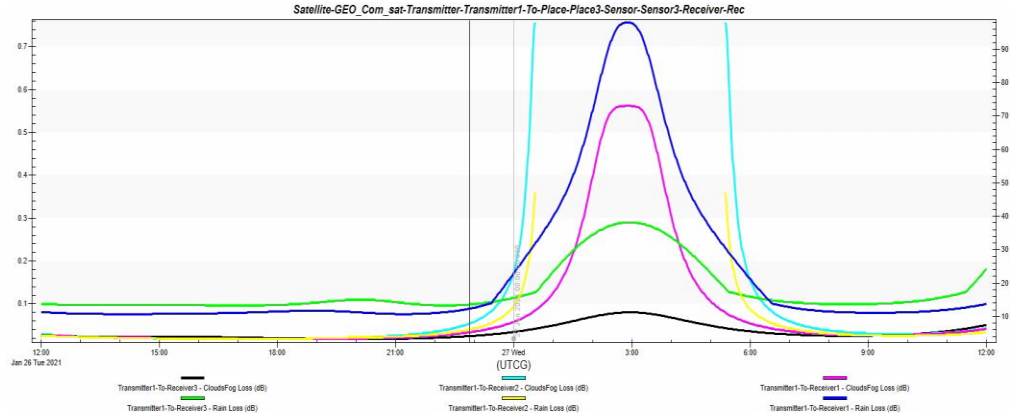

Figure 5.15 Rain Loss and CloudsFog Loss Vs Time

5.2.3. GEO Access durations

The following pie charts represents the access durations for the GEO satellite among the three receivers. We can see that the duration of access in long enough for all three receivers showing the stationary behavior of satellite. This shows that the satellite is constantly available for a given area for a long time.

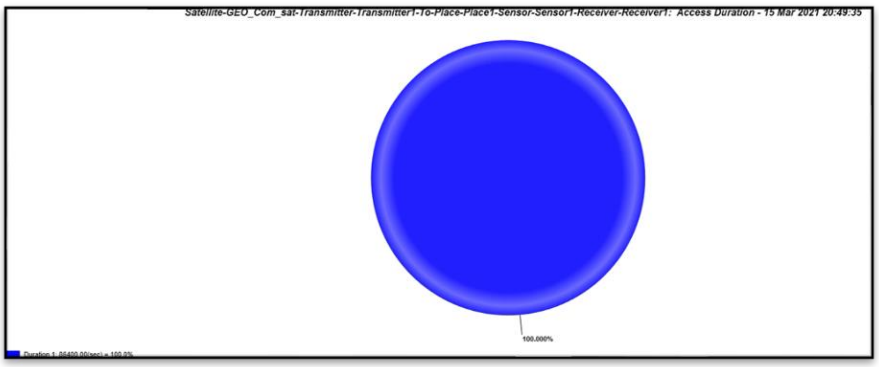

Figure 5.16 Access duration for GEO for Transmitter to Receiver 1

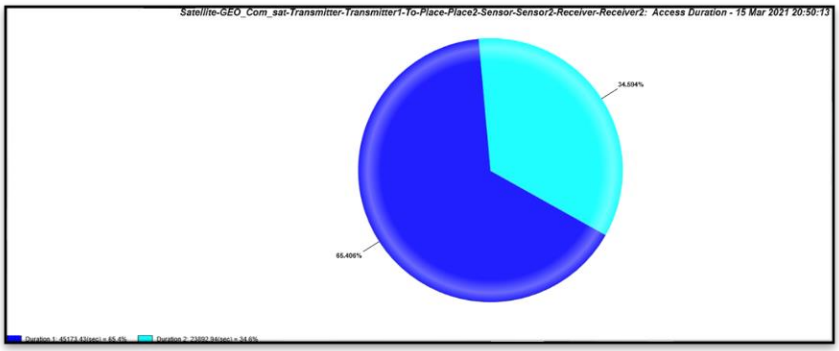

Figure 5.17 Access duration for GEO for Transmitter to Receiver 2

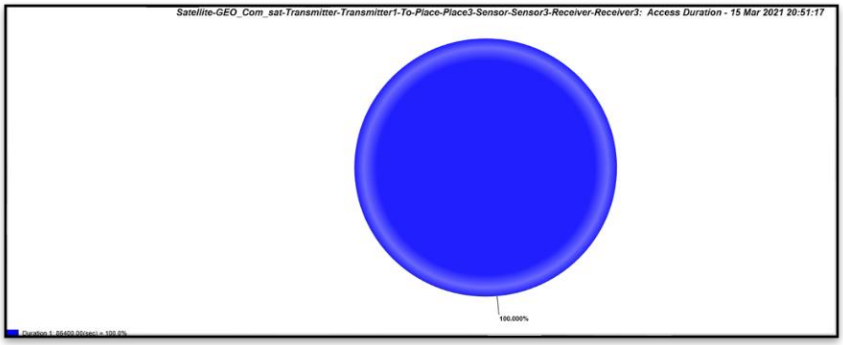

Figure 5.18 Access duration for GEO for Transmitter to Receiver 3

5.2.4. LEO Scenario without explicit RF loss

The following graphs show the graphs of BER for the LEO scenario. The closer the satellite, lower the BER and higher the link quality. Similarly, the graphs show the variation of BER and losses

with time during the contact period. This contact period is nothing but the access duration. BER varies with time starting smaller and gradually reaching higher values with increase in time for each access duration. A better view can be seen in the next graph as BER(right y axis) varies with distance(left y axis) at the first common contact in Figure 5.19. It is clear how behavior of BER in LEO and GEO scenarios are different. The free space losses(right y axis) at each contact duration is shown in Figure 5.20 and Figure 5.21 shows the BER is lower during the peak duration of access and the increases as the satellite moves away from the transmitter's region of capture. Keeping in mind that for this scenario the RF environmental losses were not set explicitly, the pattern appears in such a way that the BER is very low during the beginning phase of communication. The graphs also show a clearer view for the first common contact to show the pattern of the curves more understandably. For all, x-axis represents time of 24hr duration.

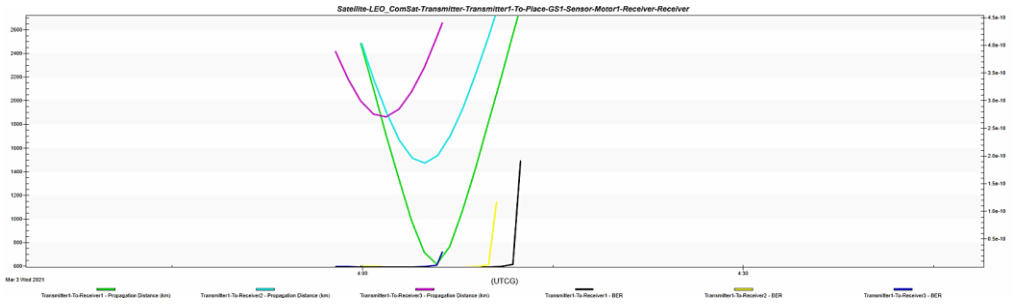

Figure 5.19 BER, Distance Vs Time (at first common contact)

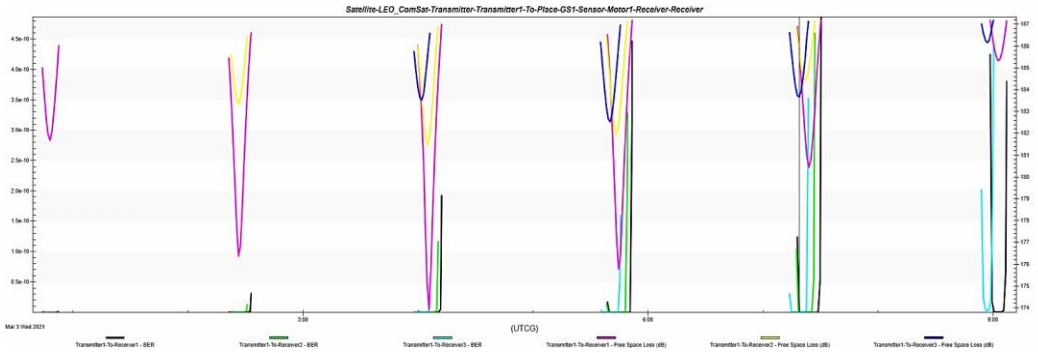

Figure 5.20 BER, Freespace Loss Vs Time (All)

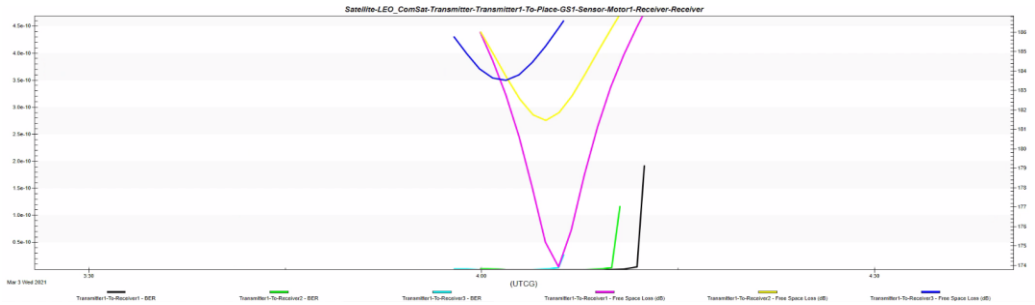

Figure 5.21 BER, Freespace Loss Vs Time (At first common access)

5.2.5. LEO scenario with explicit RF losses

The next part is to visualize the variations when explicit RF losses are provided to the scenario. When explicit RF loses are provided, the BER gradually changes from high values and peaks down at low values with decrease in distance at each receiver. The Figure 5.22 represent the BER(right y axis) variation with distance(left y axis). Figure 5.23, Figure 5.24, Figure 5.25, Figure 5.26 show the losses for this scenario and it can be seen that it behaves same as the one without explicitly specifying RF Losses(right y axis). Since the behavior of these attributes can all be represented through BER, the BER values can be used as feature for predicting the best link in the scenario.

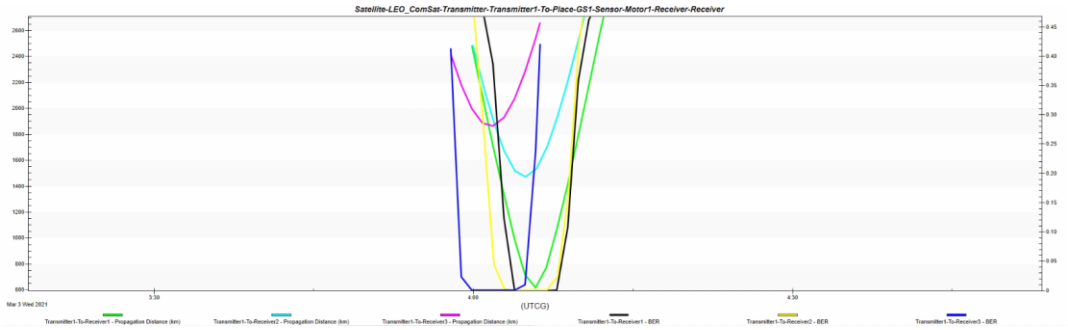

Figure 5.22 BER, Propagation Distance Vs Time (at first common contact)

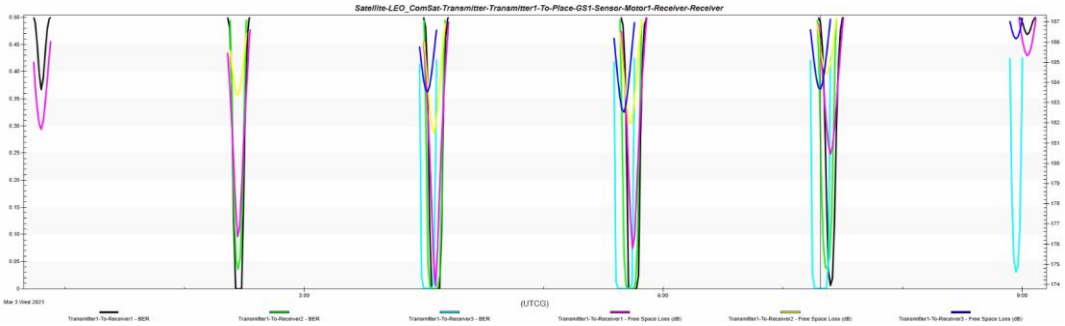

Figure 5.23 BER, Free space loss Vs Time (all)

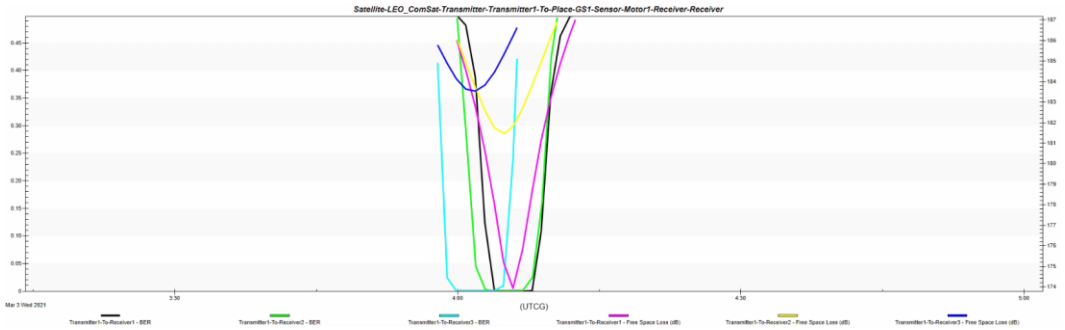

Figure 5.24 BER, Free space loss Vs Time (at first common contact)

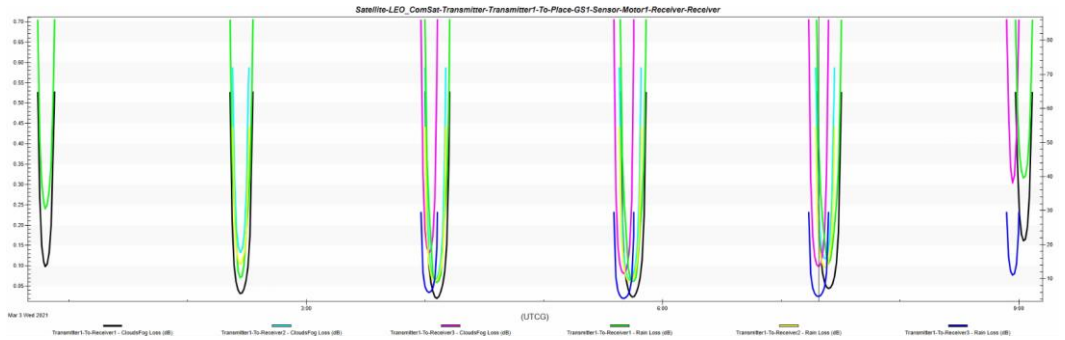

Figure 5.25 Rain and fog loss (All)

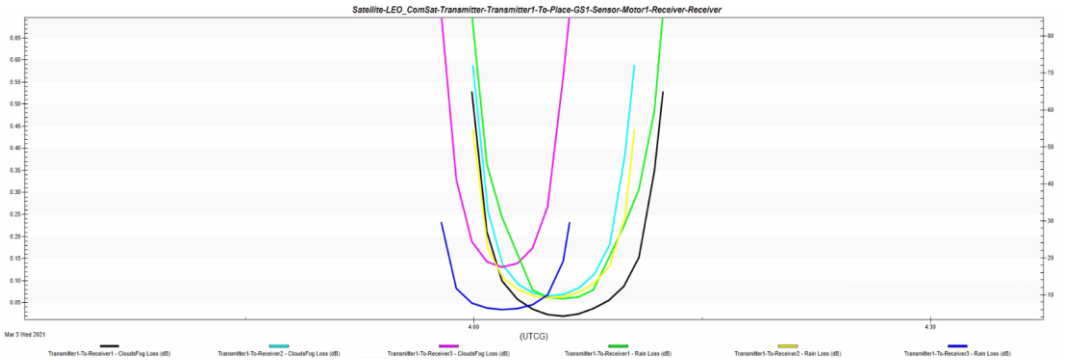

Figure 5.26 Rain and Fog Loss Vs Time (At first common contact)

5.2.6. LEO Access durations

The following pie charts in Figure 5.27, Figure 5.28, and Figure 5.29 represent the access durations for the LEO satellite at the receivers. We can see that the duration of access is shorter and distributed throughout the 24 hours at 6 to 7 different time periods for each receiver showing LEO satellite behavior clearly. Although this has nothing to do with the prediction at the NN end except the BER, it helps understand the satellite communication better.

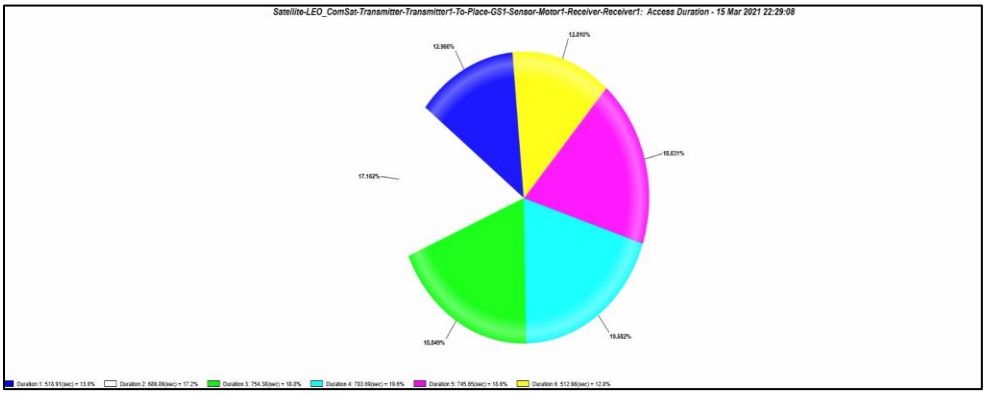

Figure 5.27 Access duration for Receiver 1

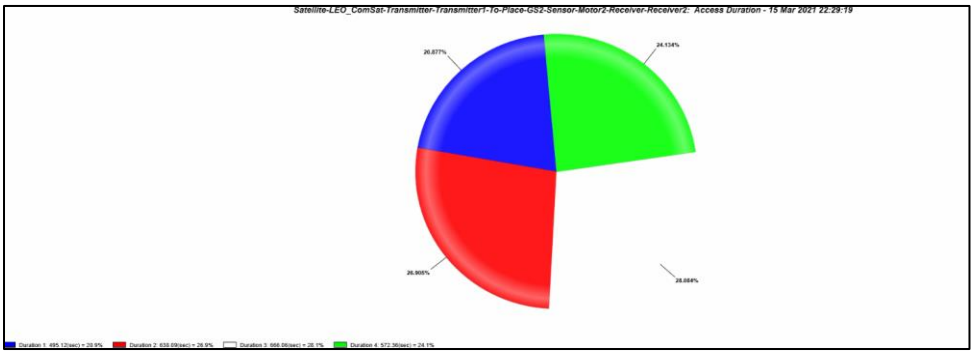

Figure 5.28 Access duration for Receiver 2

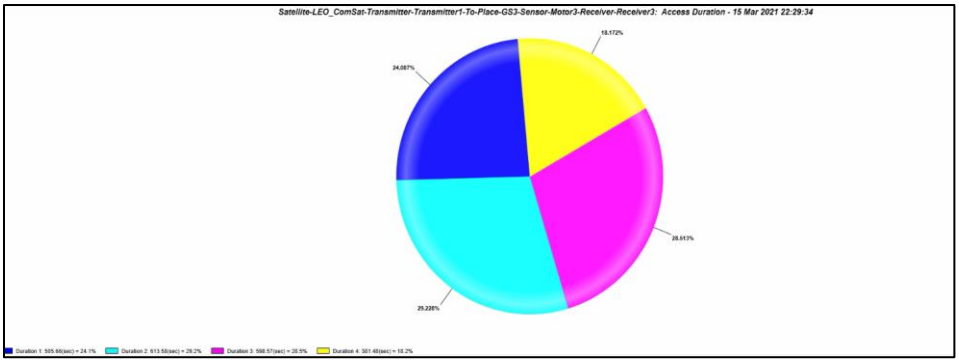

Figure 5.29 Access duration for Receiver 3

5.3. Link budget reports

The link budget report can be accessed the same way as the graphs. The Access page contains a "Link Budget" button which when clicked generates a short form of link budget for any link that is selected from the transmitter. Refer Figure 5.30, which shows the snippet from STK tool for receiver 1 as an example. The link budgets for the rest of the links are obtained in similar fashion.

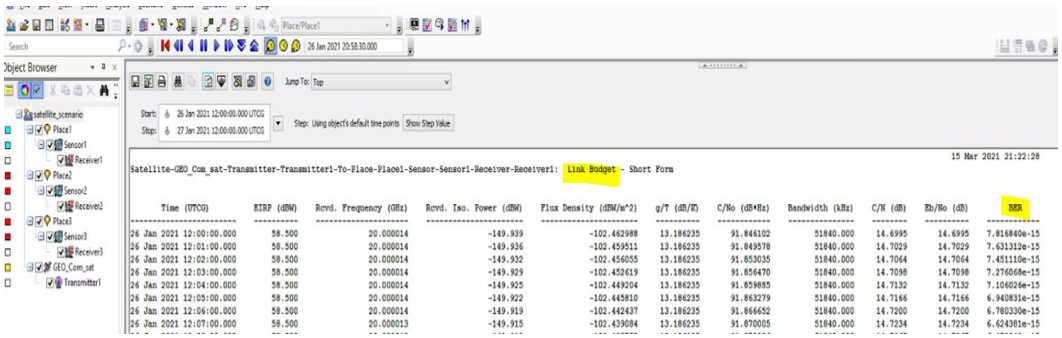

Figure 5.30 Link budget report

These results are stored as comma separated values (CSV) and used for applying further cost calculations to the models. From this chapter one can infer that instead of using every interdependent factor for link quality, we can use BER collectively to represent it and understand how other factors are related to it.

In the next chapter, we will analyze the collected data after applying the cost calculations to observe the behavior.

6. DATA ANALYSIS

6.1. Behavior of measured and theoretical data

The following table shows a sample of measures and theoretical BER values from the Eb/N0 using Q function. From Table 6.1 Sample Measured and Theoretical BER values, we observe that BER changes drastically, but the behavior of the curves is the same.

Table 6.1 Sample Measured and Theoretical BER values

Eb/No (dB)	Measured BER3	Theoretical BER
16.2024	3.32E-20	6.25884E-09
16.28	1.55E-20	5.77845E-09
15.9787	2.76E-19	7.87954E-09
15.3705	5.24E-17	1.47434E-08
14.5701	1.88E-14	3.36627E-08
13.6814	4.17E-12	8.43222E-08
12.7879	3.53E-10	2.1267E-07
12.9104	2.02E-10	1.87315E-07
13.3624	2.26E-11	1.17296E-07
13.6647	4.57E-12	8.57913E-08
13.774	2.50E-12	7.66219E-08
13.6757	4.30E-12	8.48208E-08
13.3839	2.02E-11	1.14714E-07
12.9359	1.80E-10	1.8243E-07
12.7401	4.37E-10	2.23473E-07

The theoretical BER values are higher than the measured values for all the links. Figure 6.1 and Figure 6.2 show the plot of BER vs time for simulated and synthetic data for satellite a LEO scenario. The observations are same for the GEO scenario also. Practically speaking, a model trained using this data set would fail miserably in making correct predictions. But since the behavior of the data is the same, theoretically, the model would give similar results as the data, behaves similar fashion.

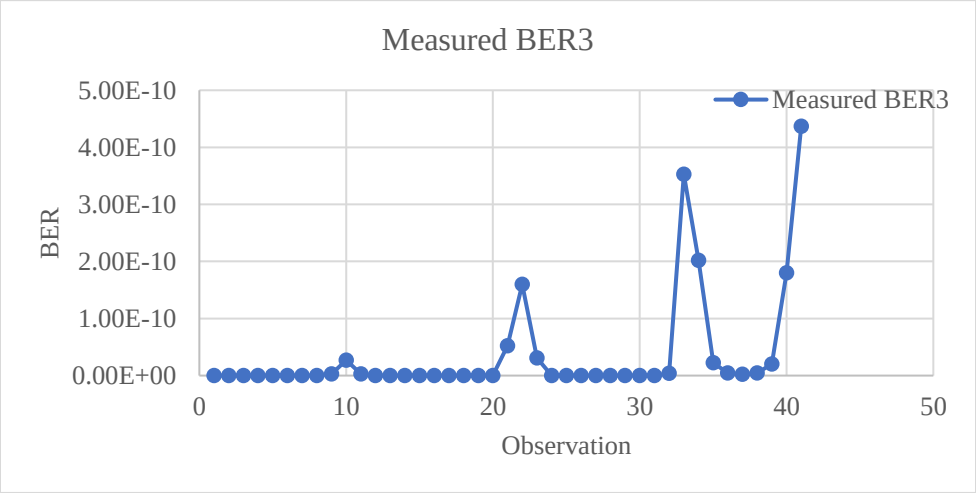

Figure 6.1 Measured BER

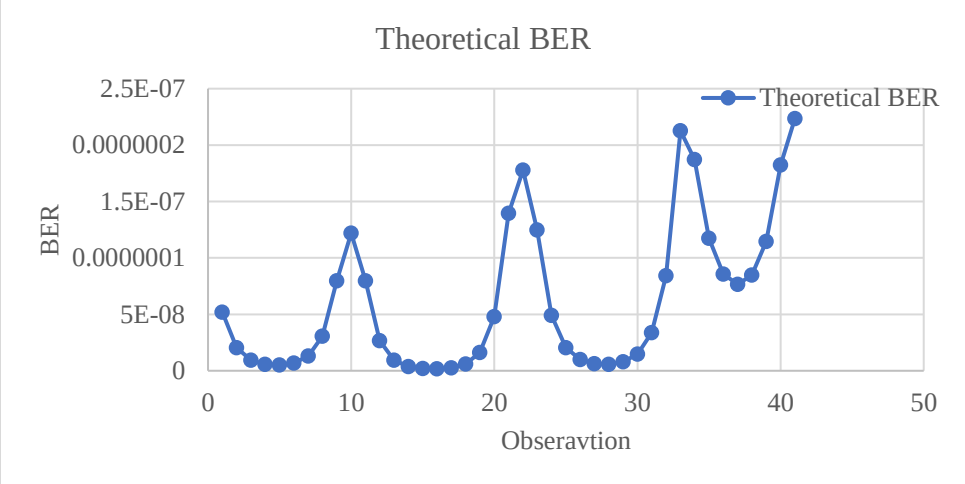

Figure 6.2 Theoretical BER

6.2. LEO scenario

BER vs EB/N0 graphs for each link can been seen in the following graphs. The following tables show the sample of synthetic data created followed by the graphs plotted for each link and simulated data. As mentioned earlier, the occupancy values are random generated using MS Excel. EbN0_1, Pe_BER1, Occupancy1 and Response_Cost1 are associated with link 1. Similarly, the naming conventions are for links 2 and 3 respectively for synthetic data.

6.2.1. Synthetic data

The tables, Table 6.2 and Table 6.3 show a sample of synthetic data for the LEO scenario. The values are obtained based on cost calculations (334.5).

Table 6.2 Sample Eb/N0 and BER for LEO scenario Synthetic data

EbN0_1	Pe_BER1	EbN0_2	Pe_BER2	EbNo_3	Pe_BER3
13	1.33293E-10	21	5.3E-57	14	6.8E-13
16	2.2674E-19	15	9.1E-16	25	7E-140
18	1.39601E-29	16	2.3E-19	18	1.4E-29
20	1.04424E-45	14	6.8E-13	25	7E-140
17	6.75897E-24	24	1E-111	24	1E-111
22	3.29609E-71	19	1E-36	20	1E-45
12	9.00601E-09	26	2E-175	24	1E-111
20	1.04424E-45	22	3.3E-71	19	1E-36
21	5.2997E-57	20	1E-45	21	5.3E-57
18	1.39601E-29	13	1.3E-10	20	1E-45

Table 6.3 Sample Occupancy and Response time for LEO Scenario Synthetic data

Occupancy 1	Response Cost1	Occupancy 2	Response Cost2	Occupancy 3	Response Cost3
7	0.83738	7	0.82577	2	0.30972
5	0.61933	5	0.61933	6	0.72255
6	0.72255	6	0.72255	6	0.72255
1	0.20644	1	0.2065	6	0.72255
2	0.30966	2	0.30966	6	0.72255
10	1.13543	10	1.13543	6	0.72255
1	139.027	1	0.20644	5	0.61933
4	0.5161	4	0.5161	1	0.20644
9	1.03221	9	1.03221	2	0.30966
9	1.03221	9	1.04382	1	0.20644

Eb/N0 values are random generated based on the range obtained from link budget simulation. Figure 6.3, Figure 6.4, and Figure 6.5 show Eb/N0 vs BER for links 1, 2 and 3 respectively of synthetic data generated all showing similar behavior. This makes it a good dataset for training the SNN to learn selection between closer values. Concurrently, Figure 6.6, Figure 6.7, Figure 6.8 show the cost and occupancy of each link and how it varies with respect to each other. The response time or cost is obtained through mathematical calculations involving the occupancy, thus showing some level of proportionality with its rise and fall. Figure 6.9 shows the cost at each observation that the SNN would see at a given random instance of time. 3 bars for each observation representing cost of links 1, 2, and 3 respectively.

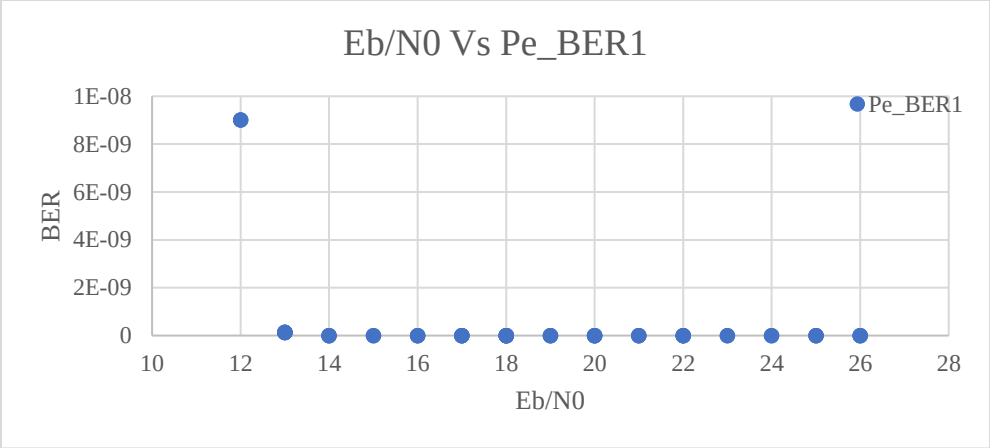

Figure 6.3 Link 1 Eb/N0 Vs BER

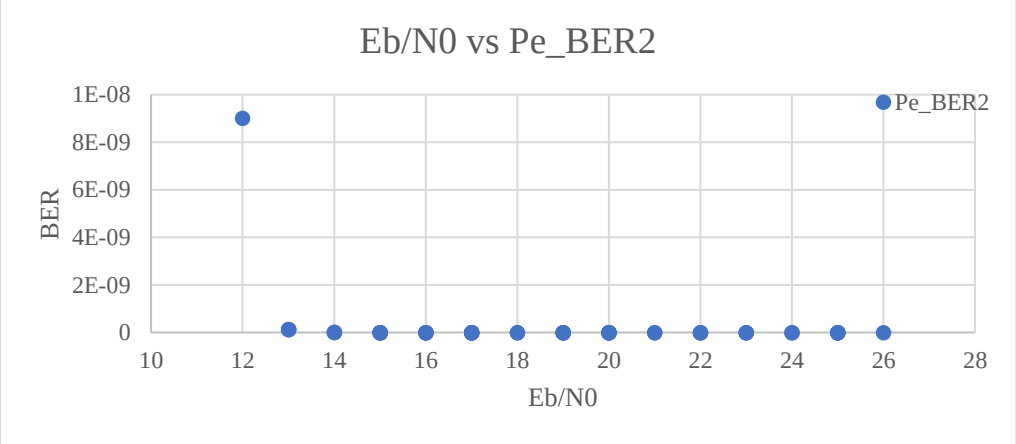

Figure 6.4 Link 2 Eb/N0 Vs BER

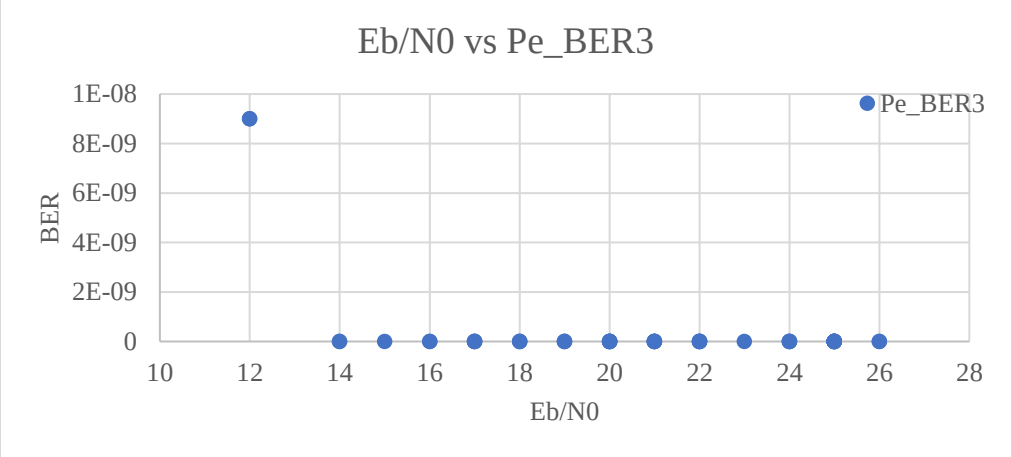

Figure 6.5 Link 3 Eb/N0 Vs BER

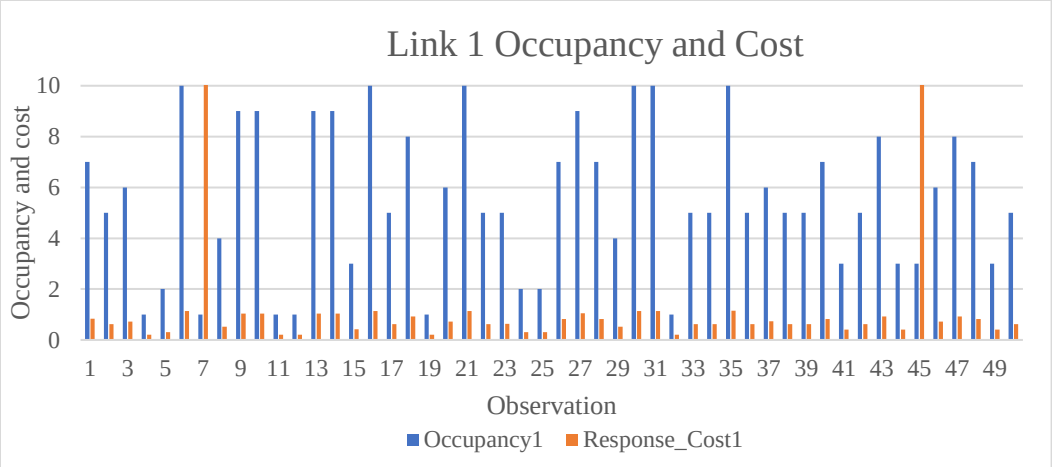

Figure 6.6 Link 1 Occupancy and Cost for each observation

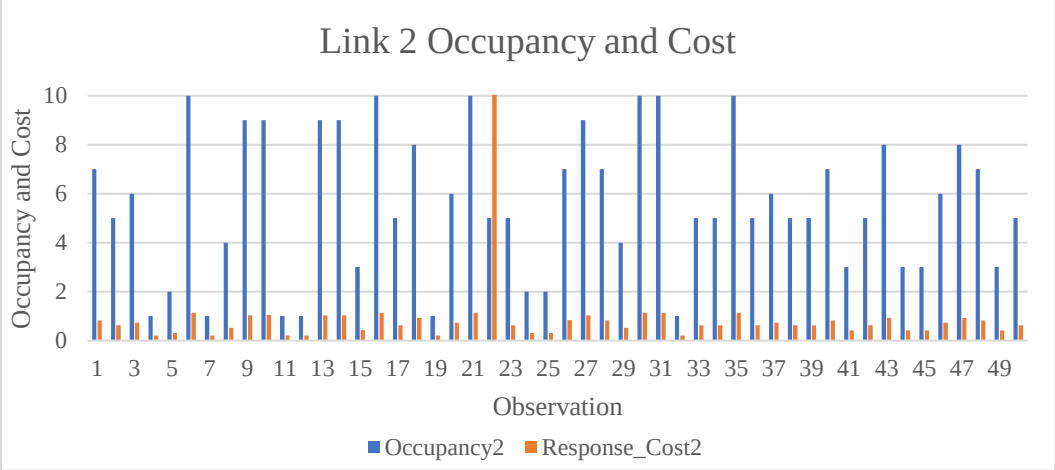

Figure 6.7 Link 2 Occupancy and Cost for each observation

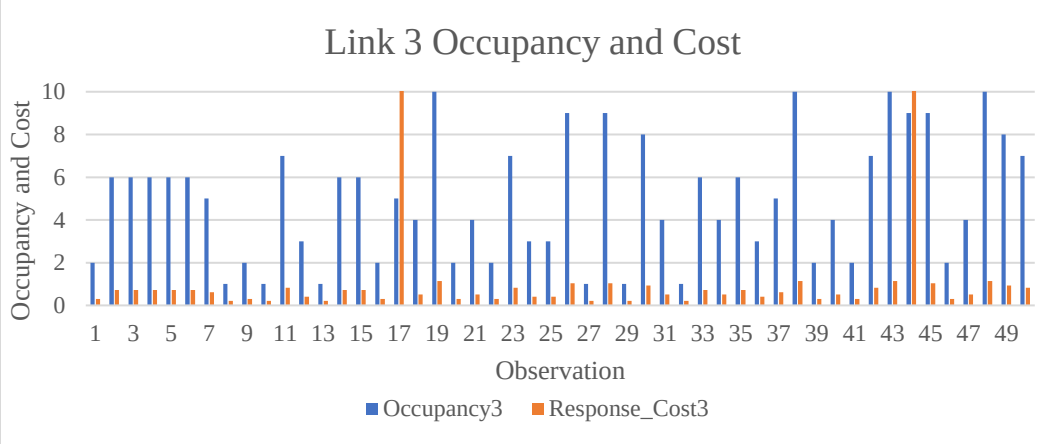

Figure 6.8 Link 3 Occupancy and Cost for each observation

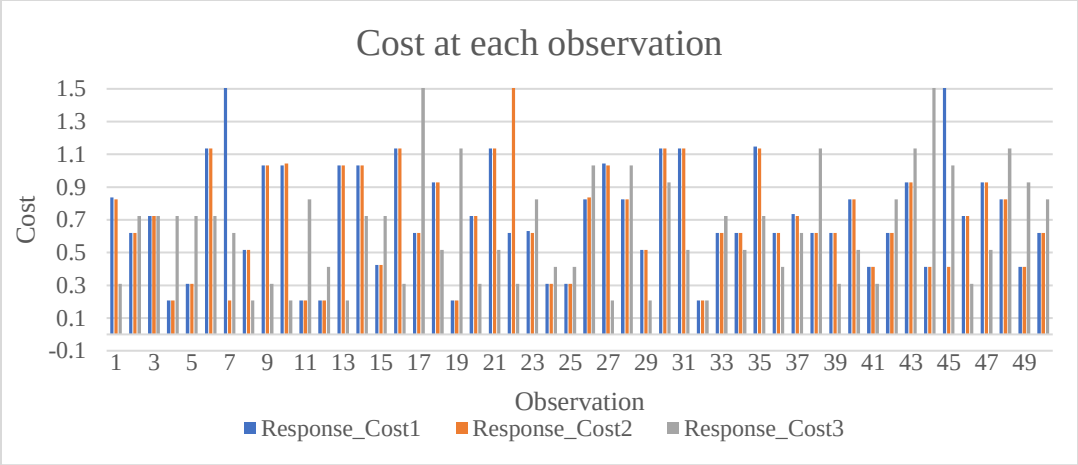

Figure 6.9 Cost for each observation

6.2.2. Simulated data

The following table shows the simulated data sample Eb/N0 and BER. Refer Table 6.4 for the values and Table 6.5 for the cost and occupancy for each link. STK tool was helpful in creating the scenario and extracting the BER.

Table 6.4 Sample Simulated data LEO (Eb/N0 vs BER)

Eb/No (dB)	BER1	Eb/No (dB)	BER2	Eb/No (dB)	BER3
14.9166	1.68E-15	14.327	9.21E-14	14.15	2.72E-13
16.1028	8.64E-20	15.239	1.48E-16	15.05	6.34E-16
17.2058	5.80E-25	16.01	2.06E-19	15.81	1.26E-18
18.0049	1.30E-29	16.493	1.80E-21	16.3	1.23E-20
18.2564	1.00E-30	16.57	7.98E-22	16.42	3.96E-21
17.8647	9.82E-29	16.224	2.70E-20	16.12	7.34E-20
16.98	8.52E-24	15.545	1.25E-17	15.49	1.95E-17

15.8507	8.85E-19	14.671	9.54E-15	14.66	1.03E-14
14.6631	1.00E-14	13.718	3.41E-12	13.74	3.08E-12
13.8986	1.23E-12	13.463	1.34E-11	13.33	2.72E-11

Table 6.5 Sample Occupancy and Response time for LEO Scenario Simulated data

Occupancy 1	Response Cost1	Occupancy 2	Response Cost2	Occupancy 3	Response Cost3
7	0.83738	7	0.82577	2	0.30972
5	0.61933	5	0.61933	6	0.72255
6	0.72255	6	0.72255	6	0.72255
1	0.20644	1	0.2065	6	0.72255
2	0.30966	2	0.30966	6	0.72255
10	1.13543	10	1.13543	6	0.72255
1	139.027	1	0.20644	5	0.61933
4	0.5161	4	0.5161	1	0.20644
9	1.03221	9	1.03221	2	0.30966
9	1.03221	9	1.04382	1	0.20644

The following graphs show Eb/N0 vs BER for all three links for this scenario. The curves behave the same way as the synthetic values in QPSK modulation. The range varies from 12 to 26 dB for Eb/N0. Refer Figure 6.10, Figure 6.11, and Figure 6.12 for the graphs. Figure 6.13, Figure 6.14, and Figure 6.15 show the link occupancy and cost for the three links directly proportional to each other. The range of BER is low as the satellite is a lot closer to the earth.

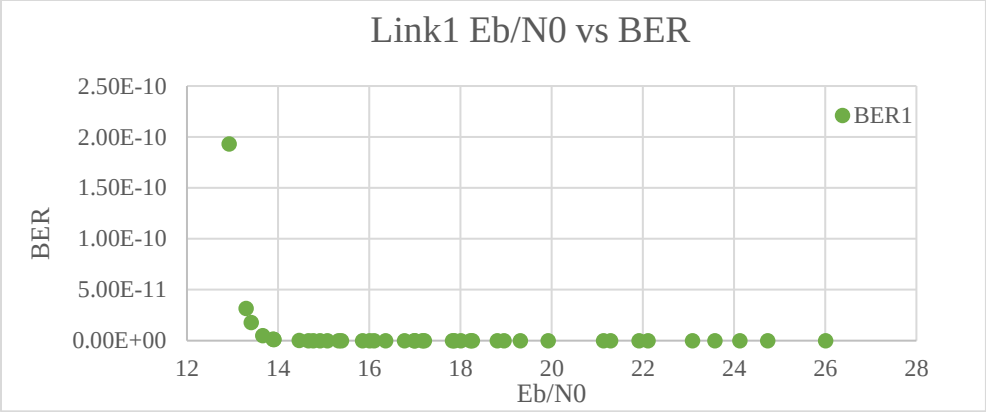

Figure 6.10 Link 1 Eb/N0 Vs BER

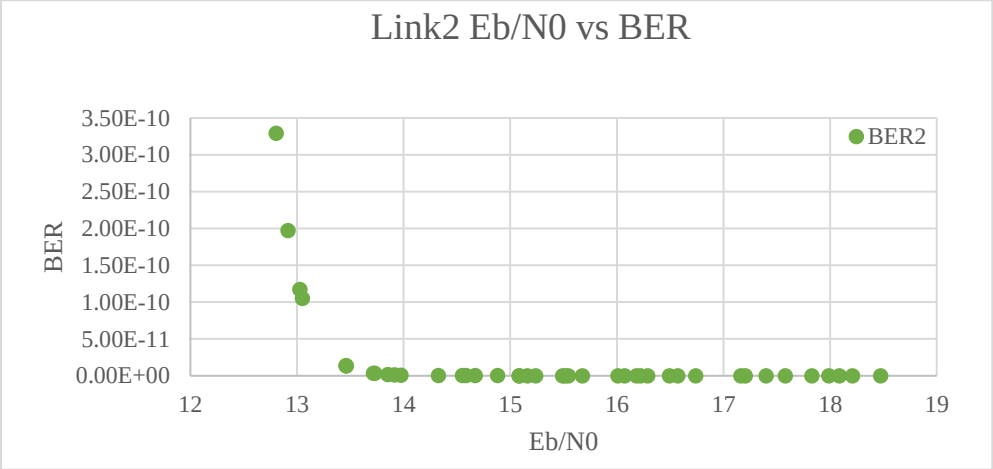

Figure 6.11 Link 1 Eb/N0 Vs BER

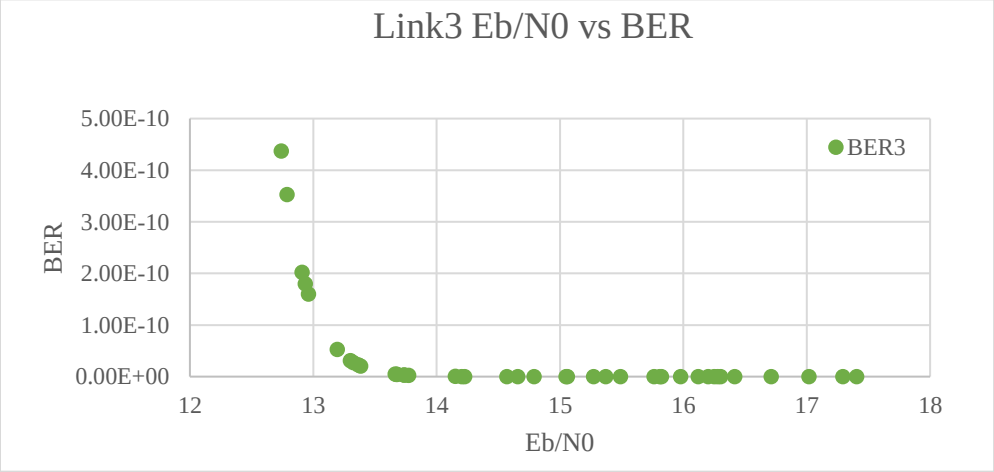

Figure 6.12 Link 1 Eb/N0 Vs BER

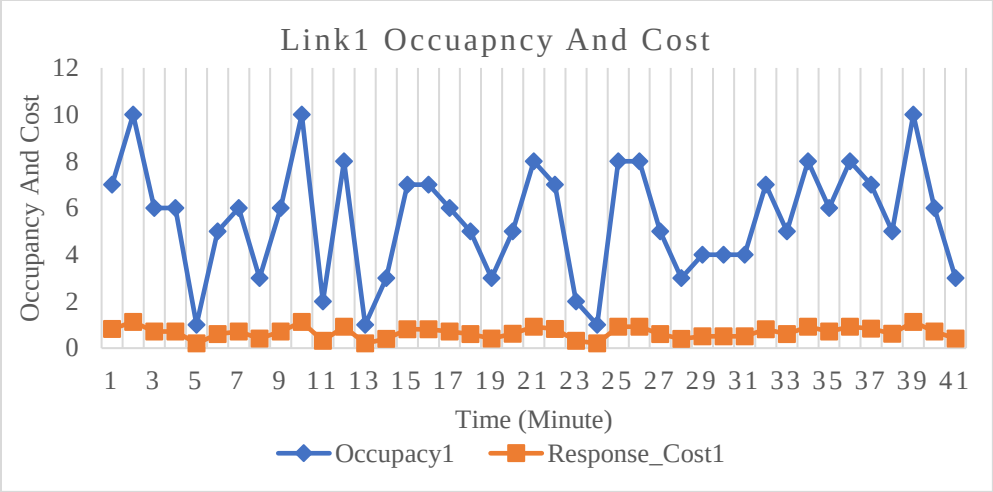

Figure 6.13 Link 1 Occupancy and Cost for each observation

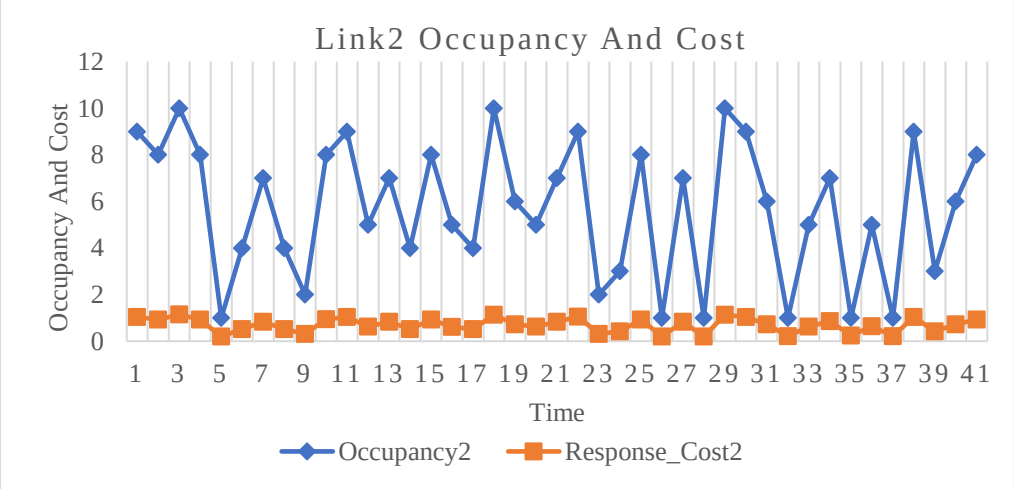

Figure 6.14 Link 2 Occupancy and Cost for each observation

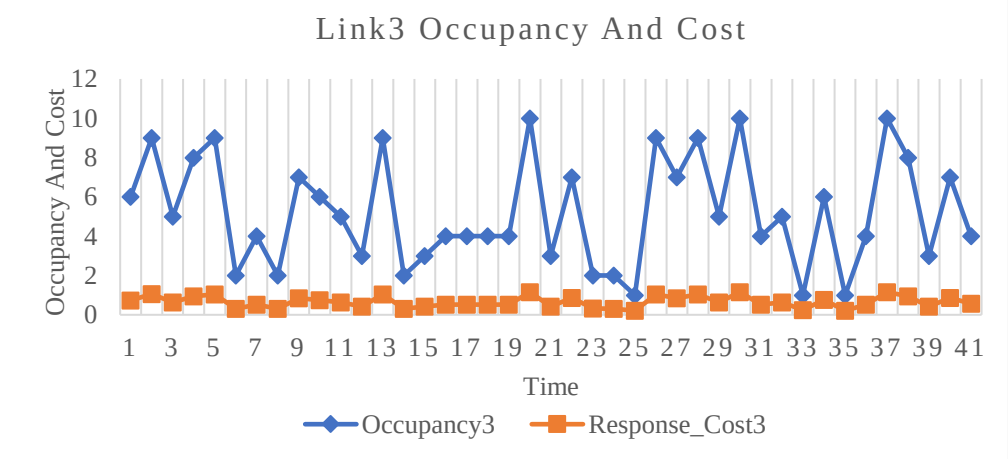

Figure 6.15 Link 3 Occupancy and Cost for each observation

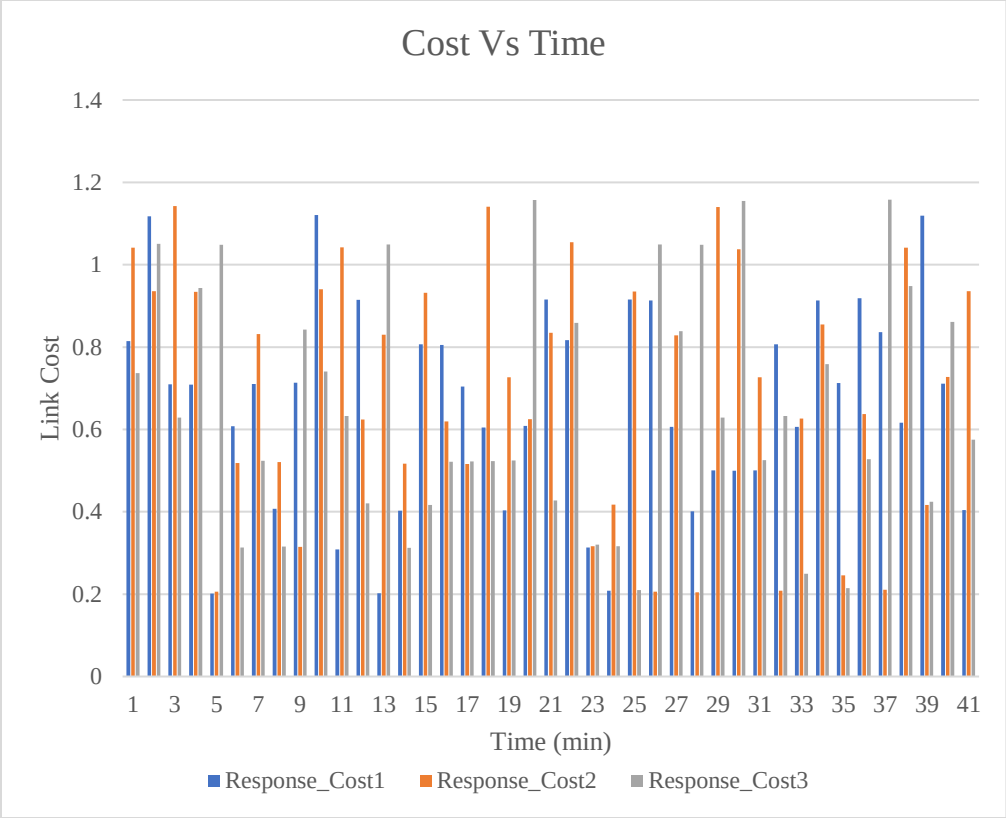

Figure 6.16 Cost for each observation

6.3. GEO Scenario

The following tables are figures are associated with synthetic and simulated data of GEO satellite scenario. The observations show higher error rates since the distance is very large as compared to that of a LEO satellite.

6.3.1. Synthetic data

Table 6.6 presents a sample of 10 observations for 3 satellite links in the GEO orbit generated in MS Excel. Eb/N0_1, Eb/N0_2, and Eb/N0_3 are the corresponding values for inks 1, 2 and 3. The bit error rates are denoted with Pe_BER1, Pe_BER2 and Pe_BER3 for each link. The BER range is very high due to the considered losses at different instances.

Table 6.6 Sample Synthetic values for GEO Synthetic data (Eb/N0 vs BER)

Eb/N0_1	**Pe_BER1**	**Eb/N0_2**	**Pe_BER2**	**Eb/N0_3**	**Pe_BER3**
10	1.56846E+16	11	1.56846E+16	8	1.56846E+16
14	62.6969357	10	1.56846E+16	19	47.01629631
6	1.56846E+16	10	1.56846E+16	9	1.56846E+16
19	156.7209877	14	31.35273816	12	21139.88535
5	1.56846E+16	9	1.56846E+16	13	158.484531
19	94.03259262	6	1.56846E+16	9	1.56846E+16
11	1.56846E+16	8	1.56846E+16	19	141.0488889
8	1.56846E+16	13	64.45193839	12	21218.24584
8	1.56846E+16	11	1.56846E+16	17	156.7209877
9	1.56846E+16	7	1.56846E+16	14	47.02483693

Table 6.7 Sample Cost and Occupancy for GEO synthetic data

Occupancy1	Response Cost1	Occupancy2	Response Cost2	Occupancy3	Response Cost3
7	2E+16	6	1.56846E+16	9	1.56846E+16
3	62.7	10	1.56846E+16	2	47.01629631
10	2E+16	7	1.56846E+16	9	1.56846E+16
9	156.7	1	31.35273816	3	21139.88535
2	2E+16	6	1.56846E+16	9	158.484531
5	94.03	2	1.56846E+16	4	1.56846E+16
3	2E+16	7	1.56846E+16	8	141.0488889
2	2E+16	3	64.45193839	8	21218.24584

7	2E+16	1	1.56846E+16	9	156.7209877
6	2E+16	6	1.56846E+16	2	47.02483693

The graphs showing the BER vs Eb/N0 in dB can be seen in the following images, refer Figure 6.17, Figure 6.18, and Figure 6.19. This is followed by the graphs showing Occupancy and cost for sample observations in GEO satellite transmitter. Evidently, the data closely resembles the QPSK modulation behavior as seen in real data. Refer Figure 6.20, Figure 6.21, and Figure 6.22 for the observations at 50 random time instances for each link. Finally, the bar chart representing cost of the three links at each instance is shown for a few sample observations, which is not clearly discernable due to the huge amount of data in Figure 6.23.

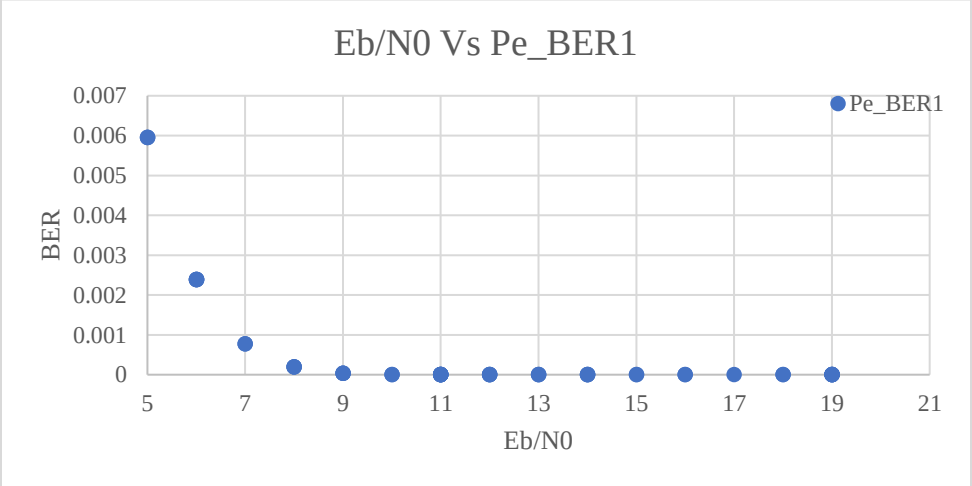

Figure 6.17 Synthetic data BER Vs Eb/N0 link 1

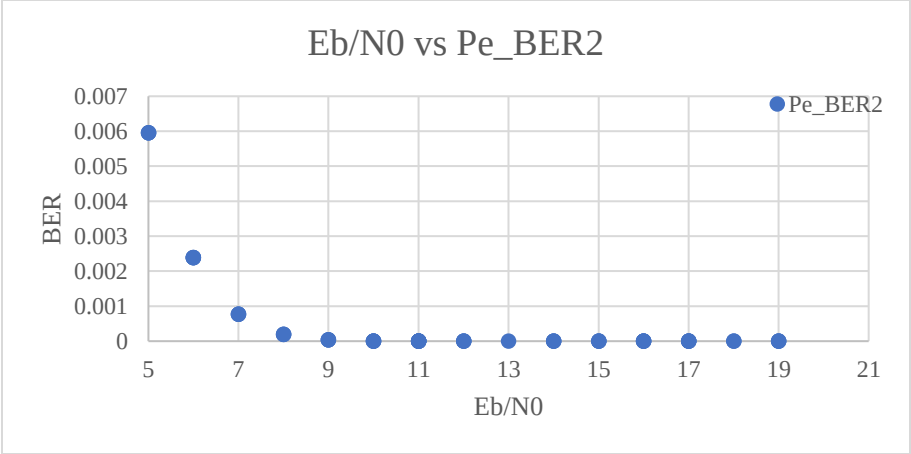

Figure 6.18 Synthetic data BER Vs Eb/N0 link 2

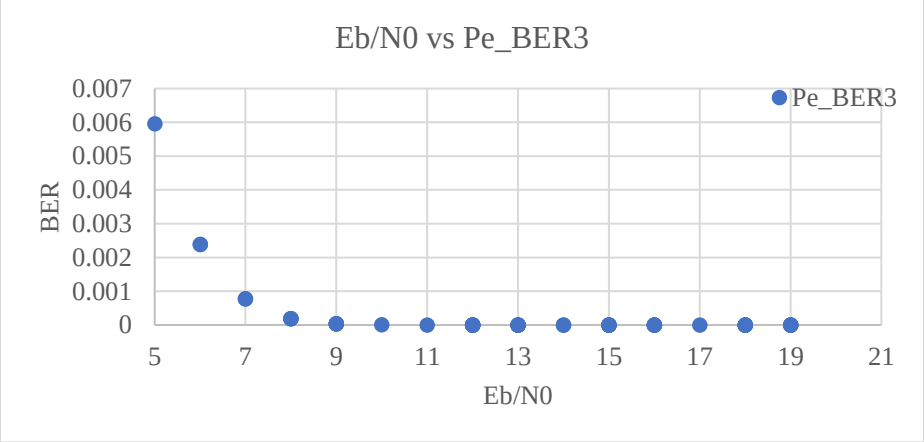

Figure 6.19 Synthetic data BER Vs Eb/N0 link 3

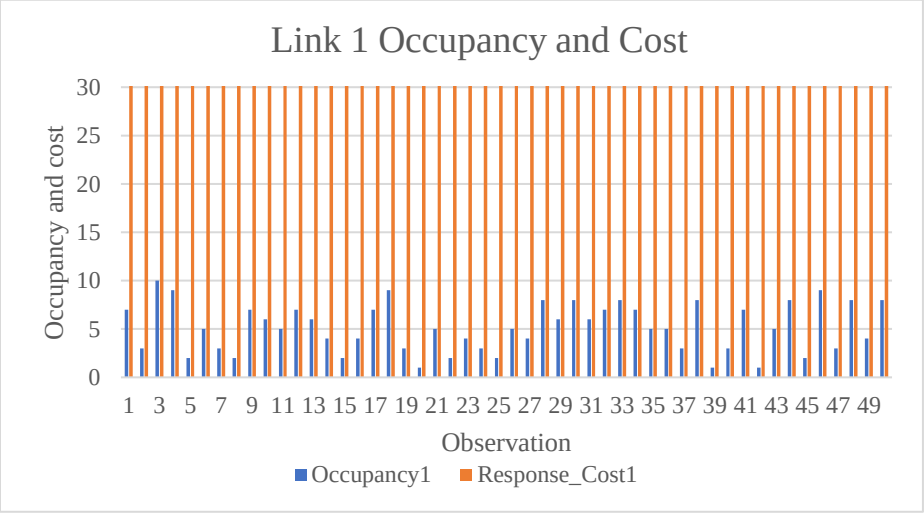

Figure 6.20 Link 1 Occupancy and Cost for each observation

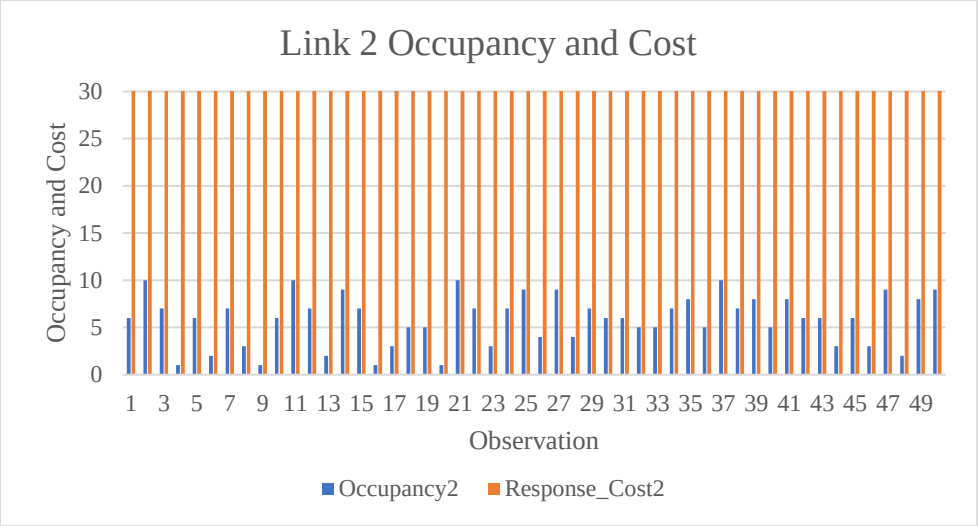

Figure 6.21 Link 2 Occupancy and Cost for each observation

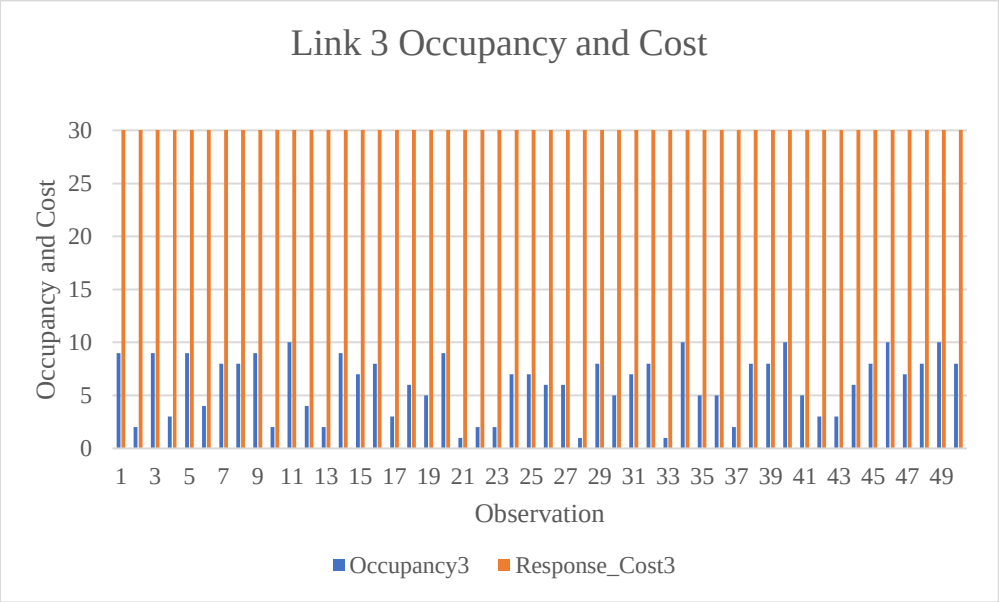

Figure 6.22 Link 3 Occupancy and Cost for each observation

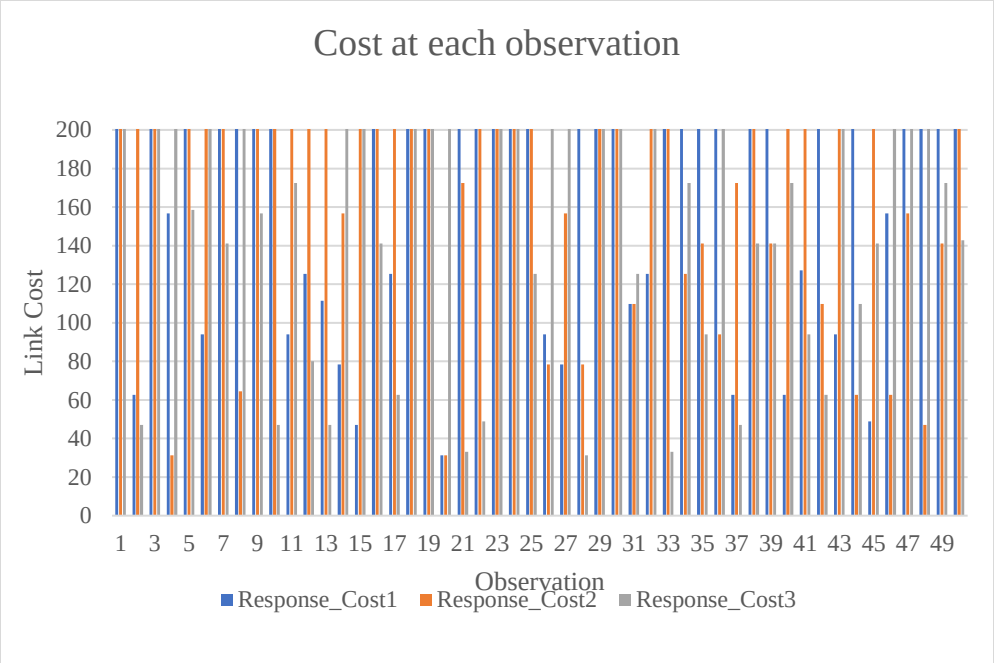

Figure 6.23 Cost at each observation

6.3.2. Simulated data

Same as the synthetic data, the following tables (Table 6.8 and Table 6.9) show the Eb/N0, BER, occupancies and cost of each link simulated for GEO scenario using STK tool. The range of BER is very low as rain and fog loss is not enabled in this data. This means that the sky is clear, and data is subject to less error and loss. Refer Figure 6.24, Figure 6.25, and Figure 6.26 for Eb/N0 vs BER for GEO simulated scenario. Figure 6.27, Figure 6.28, and Figure 6.29 show the cost and occupancy variation with respect to rise and fall in each other. Finally, Figure 6.30 shows the cost at each observation.

Table 6.8 Sample Simulated values Eb/N0 vs BER for GEO

Eb/No (dB)	BER1	Eb/No (dB)	BER2	Eb/No (dB)	BER3
17.9995	1.41E-29	17.9302	3.85E-29	18.0712	4.88E-30

18.0029	1.34E-29	17.9349	3.59E-29	18.0731	4.75E-30
18.0064	1.27E-29	17.9396	3.36E-29	18.075	4.61E-30
18.0098	1.21E-29	17.9443	3.14E-29	18.0768	4.49E-30
18.0132	1.15E-29	17.949	2.93E-29	18.0787	4.37E-30
18.0166	1.09E-29	17.9537	2.74E-29	18.0805	4.25E-30
18.02	1.04E-29	17.9583	2.56E-29	18.0823	4.14E-30
18.0234	9.91E-30	17.9629	2.40E-29	18.084	4.03E-30
18.0267	9.43E-30	17.9675	2.24E-29	18.0858	3.93E-30
18.03	8.98E-30	17.9721	2.10E-29	18.0875	3.83E-30

Table 6.9 Sample Simulated values cost and occupancy for GEO

Occupacy1	Response Cost1	Occupancy2	Response Cost2	Occupancy3	Response Cost3
10	172.0983	5	93.86021661	5	93.7891133
9	156.4526	4	78.21683678	10	171.9376962
1	31.28765	5	93.85998667	8	140.6781994
5	93.86998	5	93.85987242	10	171.9376062
4	78.2243	10	172.0760817	3	62.52948171
10	172.0978	9	156.4327038	8	140.6780669
3	62.57853	6	109.5027972	8	140.678024
5	93.86966	9	156.4324787	4	78.15907852
6	109.5152	10	172.0756315	5	93.78876245
4	78.22389	3	62.57266782	10	171.93735

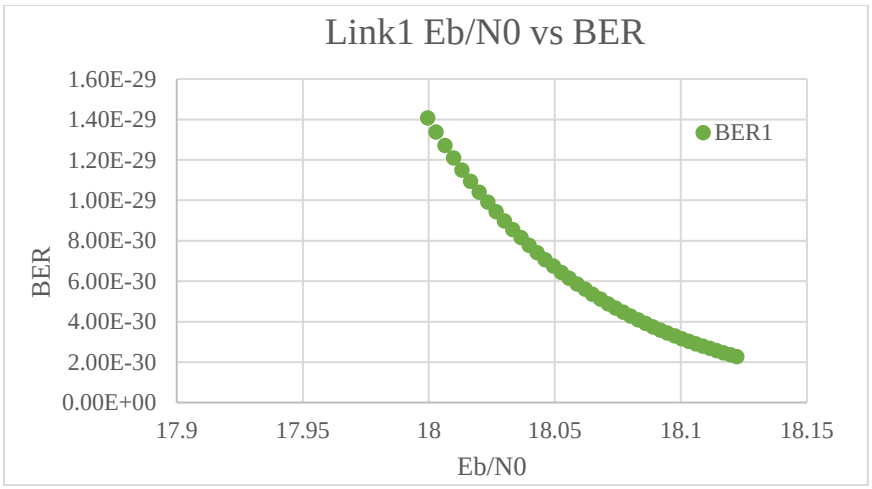

Figure 6.24 Link 1 Eb/N0 Vs BER

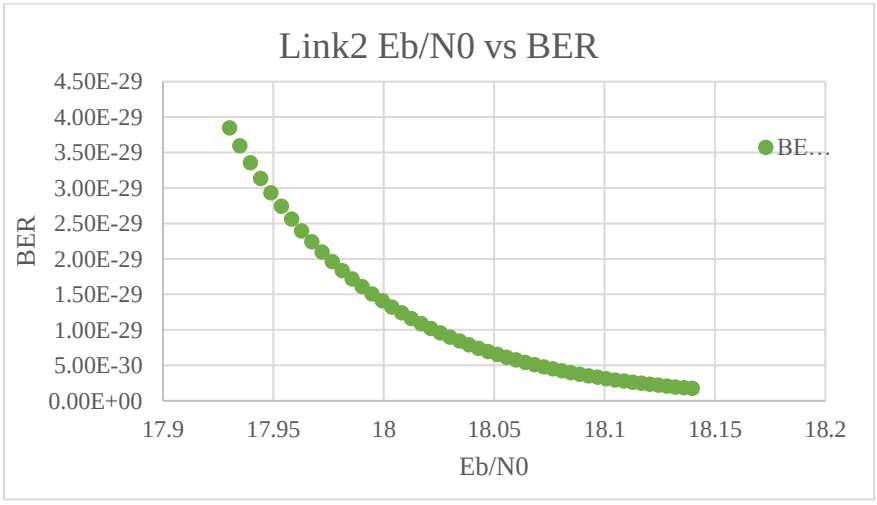

Figure 6.25 Link 2 Eb/N0 Vs BER

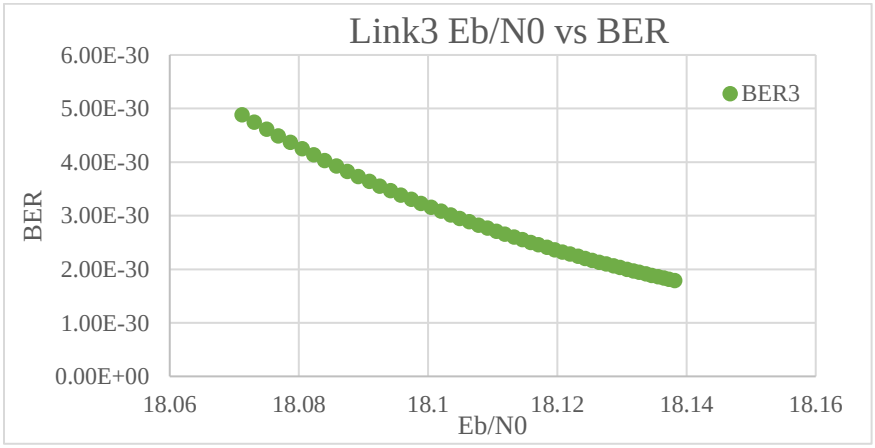

Figure 6.26 Link 3 Eb/N0 Vs BER

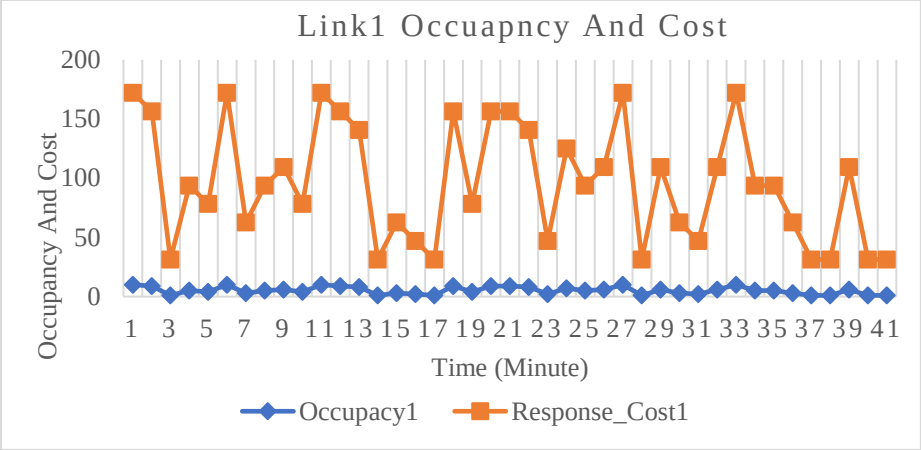

Figure 6.27 Link 1 Occupancy and Cost for each observation

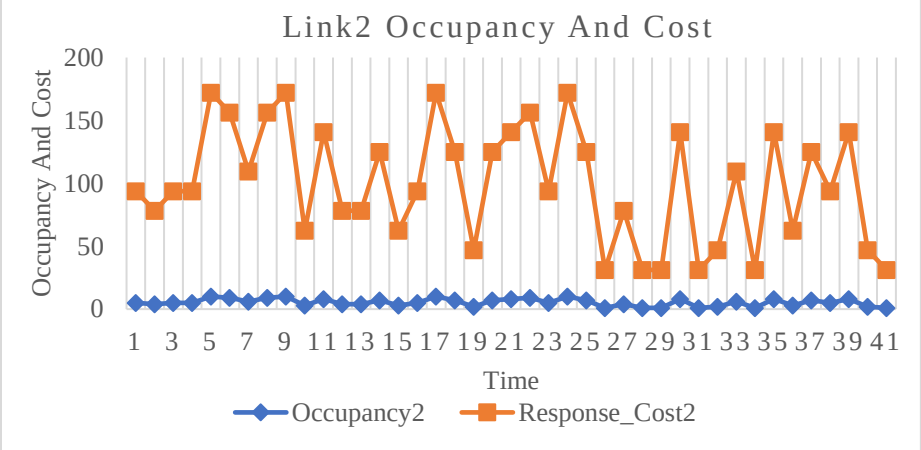

Figure 6.28 Link 2 Occupancy and Cost for each observation

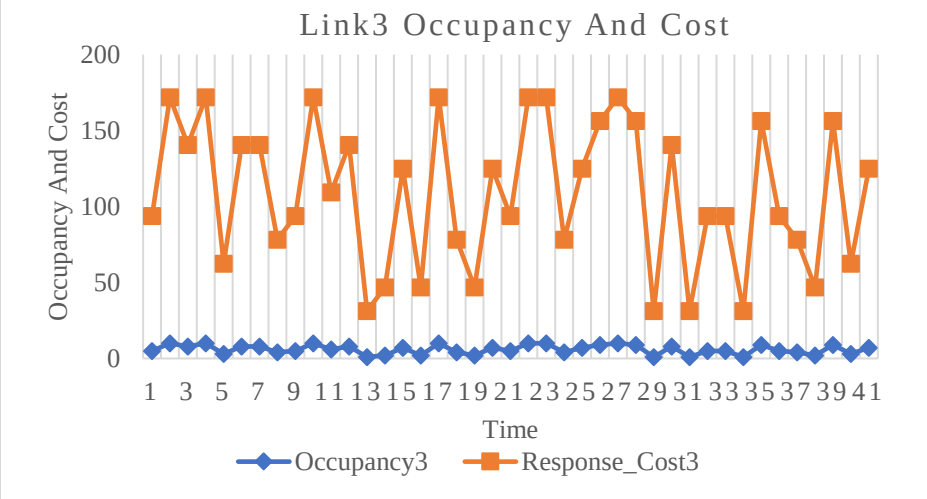

Figure 6.29 Link 3 Occupancy and Cost for each observation

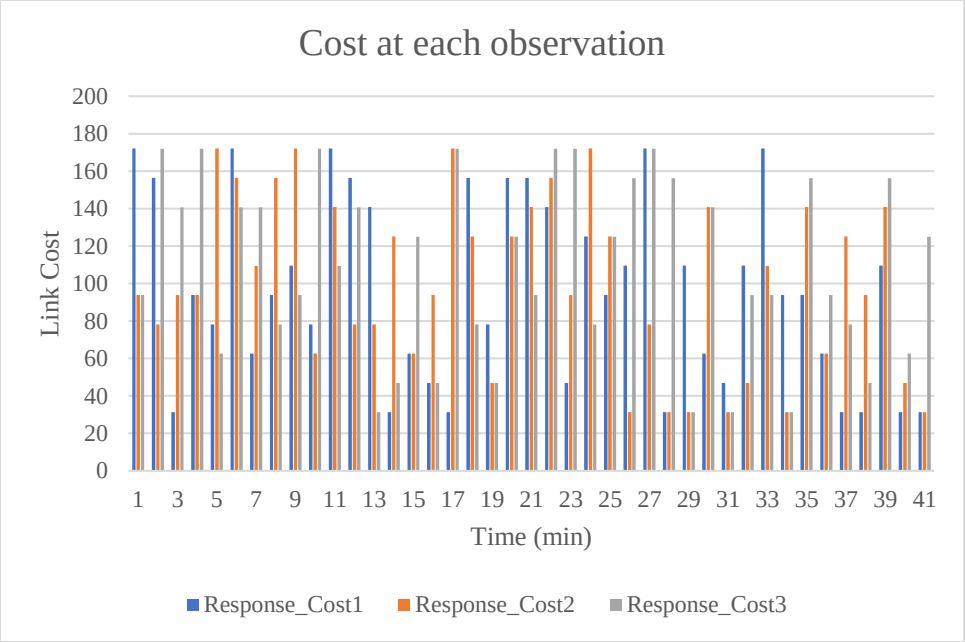

Figure 6.30 Cost at each observation

These graphs verify that the proportionality matches both in synthetic and simulated data making it apt for use to train and evaluate the SNN. The data analysis shows that the BER behaves as it should for both the scenarios under the specific conditions given. The range of values are low with nearer distances and when the sky is clear, whereas it is the opposite otherwise. There is a big difference in the range, but the plotted curves show the same behavior. Further, the cost at each observation from the bar graphs show how convenient it would be for the SNN to make prediction at each observation.

In the next chapter, we will discuss how to implement a spiking neural network starting from traditional neural network and moving to Nengo framework.

7. IMPLEMENTATION OF SNN

The data collected from the simulation is used to find the cost of each link and fed to the SNN for predicting the best link. The workflow used to complete the SNN model is done using the steps mentioned in [42], such as data collection, analysis, model construction, model validation, and inference.

7.1. Creating the Keras model

A sequential model is created with a few dense layers, having 1 input layer, 1 output layer and 1 hidden layer starting from 3 input neurons, 16 hidden neurons, and 4 output neurons. This architecture is a quicker and easier but works on the same principles while other methods require sightly varying architecture[47]. The dataset is split into train, test, and validation. The appropriate amounts of batch size and epochs depending on each dataset and the model are compiled and fit. The model is compiled using sparse categorical cross-entropy loss function and Adam optimizer with 0.001 learning rate and evaluated for the accuracy metrics. A good model is chosen according to the results obtained after testing in Keras, is selected for conversion. Figure 7.1 Code snippet from keras shows a glimpse from DNN implementation.[48] provides a brief insight on how to build a simple deep neural network for classification.

As the model parameters are not known and NNs are mostly trial and error based, several experiments were conducted by changing the number of layers, activation functions, optimizers, and size of layers. The results if impact is recorded in a table in the next chapter. What started with just a few layers picked pace and quicky rise to different number of layers and different size for each layer. Finally, a couple of chosen models were chosen to see if SNN shows improvement in performance.

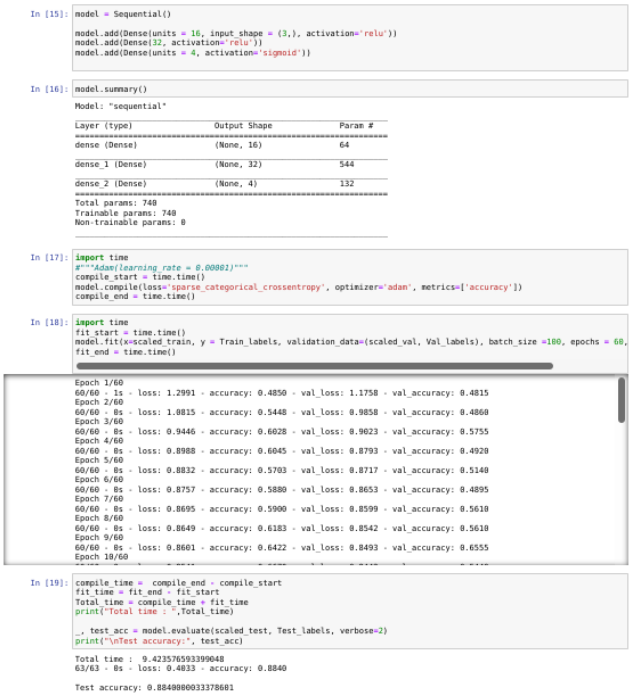

Figure 7.1 Code snippet from keras

7.2. Converting the model to Nengo Model

The experiments which started with an ad-hoc network has now evolved with 3 neurons in the input layer, 16, 32 and 64 neurons in each hidden layer and 4 neurons in the output layer. Totally 2980 parameters seen from the model summary, refer Figure 7.3. This may change depending on which model was selected and there is possibility it may not work for certain other applications.

After the Keras model is created, it is passed into the NengoDL Converter, which is a tool designed to automate the translation from Keras to Nengo. After training the weights, the do_training condition is set to true and the nengo_dl.Simulator() class is used to load the weights from the saved file and used to evaluate the model. The model is compiled and fit before this step. An example from the NengoDL tutorials was referred to plot the predictions and perform training in SNN[47]. The impact of various factors such as activation functions, neuron types, synaptic smoothing, and firing rates on the accuracy, time taken, and loss was conducted.

```
# input
_input = tf.keras.Input((3,))

# Dense layers
dense = tf.keras.layers.Dense(units=16, activation = 'relu')(_input)
dense1 = tf.keras.layers.Dense(units=32,activation='relu')(dense)
dense2 = tf.keras.layers.Dense(units=64,activation='sigmoid')(dense1)
#dense3 = tf.keras.layers.Dense(units=128,activation='sigmoid')(dense2)
dense4 = tf.keras.layers.Dense(units=4)(dense2)

model = tf.keras.Model(inputs=_input, outputs=dense4)
```

```
converter = nengo_dl.Converter(model)
```

Figure 7.2 Code snippet from Nengo

```
model.summary()
```

```
Model: "model_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_9 (InputLayer)         [(None, 3)]               0
_________________________________________________________________
dense_34 (Dense)             (None, 16)                64
_________________________________________________________________
dense_35 (Dense)             (None, 32)                544
_________________________________________________________________
dense_36 (Dense)             (None, 64)                2112
_________________________________________________________________
dense_37 (Dense)             (None, 4)                 260
=================================================================
Total params: 2,980
Trainable params: 2,980
Non-trainable params: 0
```

Figure 7.3 Sample model summary snippet

For conversion to spiking neurons, a helper function is defined that will build the network, load weights from the parameters file, and make it easy to meddle with few other features of the network [47]. The run_network() function contains lines of codes to speed up the simulation at the network level, predicts the best link for the test data and evaluates the model using validation data. Initially, the output is constant for each observation as the data fed is not temporal. After swapping the activations with Spiking Rectified Linear function, the spikes would start to appear.

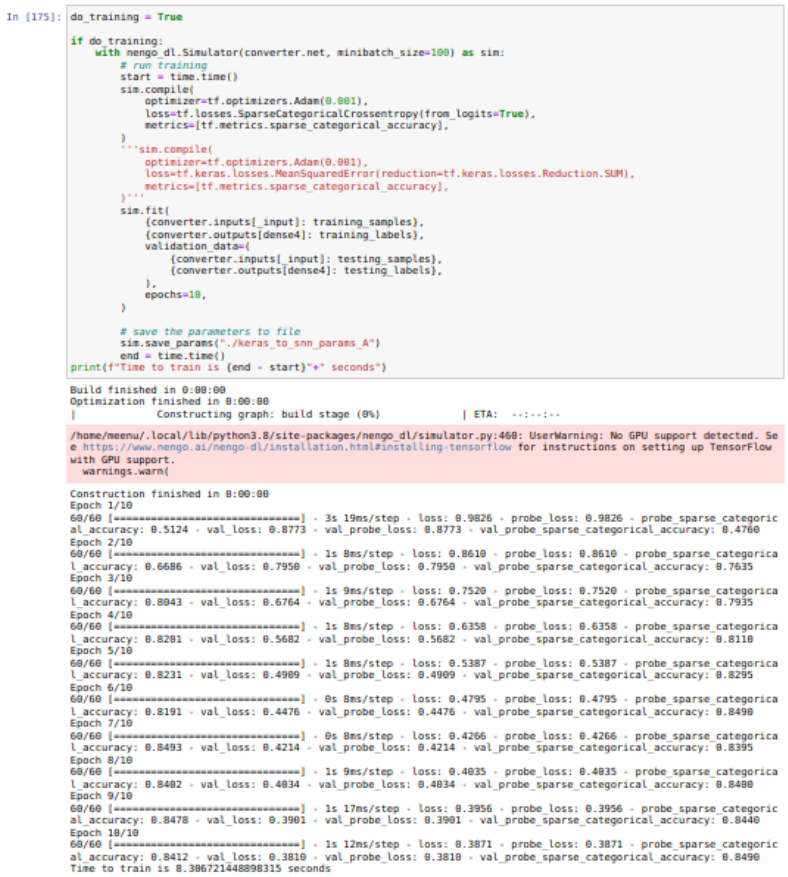

Figure 7.4 Sample SNN code snippet

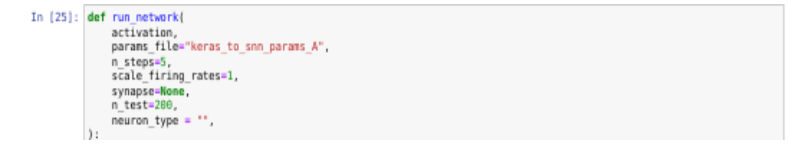

Figure 7.5 Sample run_network() code snippet

The function contains arguments to specify activation, number of time steps to predict (n_step), scale firing rates, synapse value, number of testing data, and the neuron type. The shape of the data is different compared to that in Keras. There is an additional time parameter that adds to the third dimension of the dataset.

Experiments for each dataset were conducted step by step to analyze the performance of the NN. Each section contains sub-sections to obtain the results for the NN based on the change in parameters such as activation functions, neuron types, number of layers and size of layers, epochs, batch size, and optimizer. Activation functions tested include sigmoid, SoftMax, Tanh and Relu functions tested in this framework as these are the most used activations for classification.

Neuron types are (a) LIF neuron nengo_dl.SoftLIFRate() with smoothing around the firing threshold (b) Spiking version of LeakyReLU, nengo_dl.SpikingLeakyReLU() [47]. Applying these neuron types along with spiking activation produces good results. The optimizers most suited for this type of multi-class classification are Adam or RMSprop. Further, synaptic smoothing and changing the firing rates of the neurons are also done based on values ranging from 0.00001 to 0.5 and 0.1 to 1000 exponentially respectively.

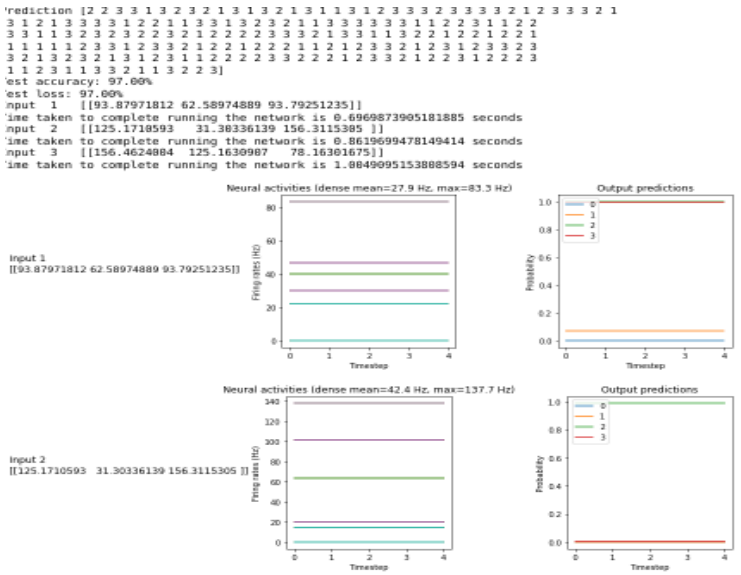

Figure 7.6 Prediction for imparting spiking activites

Let's first understand synaptic smoothing. The reasons for performing synaptic smoothing is to obtain smoother curves and reduce rapid fluctuations during spiking. This is achieved through

synaptic filters. The Synapse parameter in this run_network() function does this activity, which basically acts as a low pass filter time constant. This creates a low-pass filter with the given time constant at the output of all the spiking neurons[47]. Hence this helps compute a running average of each neuron's activity for a time period instead of observing the final time step alone. So, if there are any rapid changes in the inputs, the network output will be less responsive making it take more time for the output to settle, refer Figure 7.8. The accuracy, slightly improves at times but deteriorates if too high of a value is given, meaning that the spiking is restricted too much.

Next attribute is firing rate, which can also be sued to improve the model performance. We have seen that the output of a neuron is a spike in SNN. If this rate is increased, the output will also be updated correspondingly. If the firing happens more often, the model will begin resembling the original non-spiking model, where the output is the actual fire rate. This can be done without training the model again by giving a linear scale to all the input neurons, which agrees well with ReLU activations functions[47]. Refer Figure 7.7 to see how the neurons spike when firing rates vary from 0.1 to 8.

Very high firing rates would almost give a constant output without any spiking activity, which is not the aim of this literature. For regularizing by optimizing the firing rates during training is another option to increase the firing rates, but it is not done in this thesis to save time. It is important to be aware that there are tradeoffs depending on the specific application for these factors.

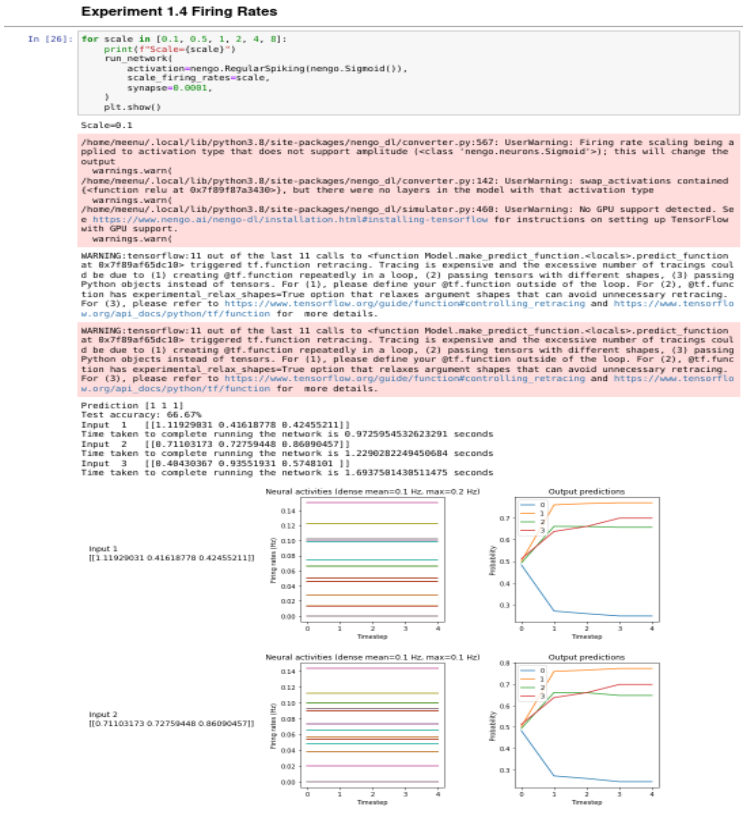

Figure 7.7 Prediction scaling firing rates snippet

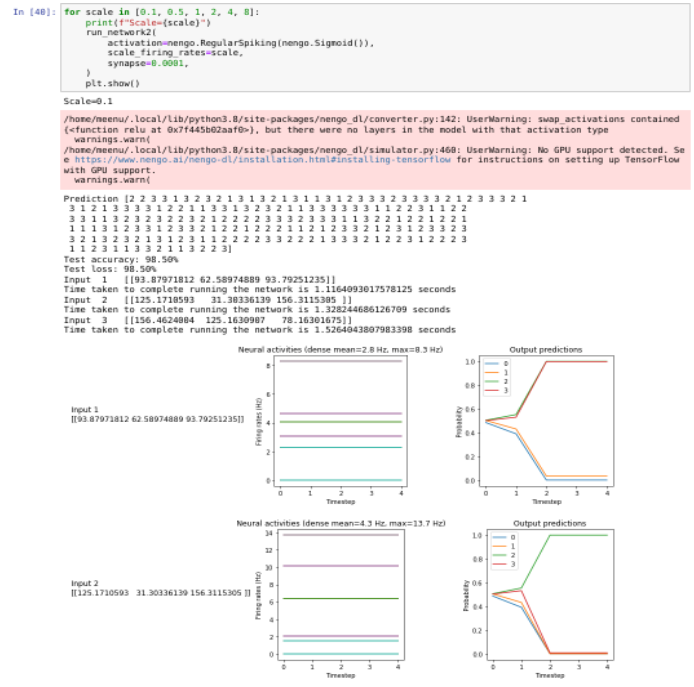

Figure 7.8 Prediction performing synaptic smoothing

8. RESULTS OF SNN

After several trials and errors experimenting with the number of neurons, layers and the sizes, for the LEO scenario, the best model consists of 3 input neurons 3 hidden layers of 16, 32 and 64 neurons each and output layer of 4 neurons for the Keras model. Unless mentioned specifically, the output layer activation with 4 neurons, is given as ‘sigmoid’. When converted to Nengo, the model shows better accuracy at certain conditions. The following graph shows the generic loss plot that shows the loss reduces with each epoch for both testing and training datasets. The model might produce the same results after conversion to the new framework but gives a possibility of increasing the accuracy when a few parameters are tweaked. In most of the cases increasing the synaptic smoothing decreased the prediction accuracy and increase in firing rates up to a certain level produced increase in prediction accuracy. Very high firing rates resulted in a drop in the accuracy.

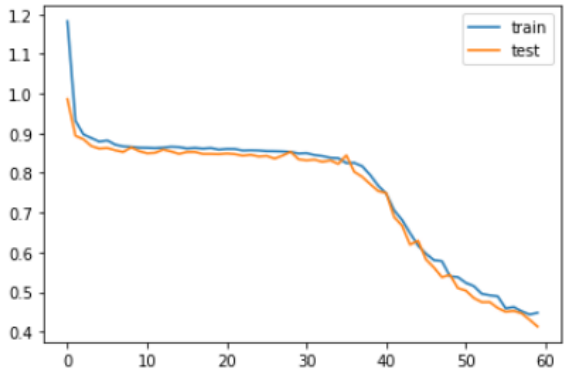

Figure 8.1 Generic loss plot

The tables below show the results of the two models for the 3 different datasets obtained for LEO and GEO scenario. Typically, the loss plots for all the models look like above Figure 8.1 but results vary according to the data at hand.

8.1. LEO Scenario Results

The results of impact on accuracy and time for the synthetic data on the Keras neural network and SNN according to each parameter is listed below. Refer Table 8.1 and Table 8.2 for results of LEO scenario synthetic data. Refer Table 8.3 and Table 8.4 for results of simulated data.

Table 8.1 Impact chart LEO synthetic data Keras

Model	Parameter	Layers	Loss	Time	Optimizer	Accuracy	Batch	Epoch
1	Optimizer	3,16,32,4	0.394	9.69	Adam	0.89	100	60
2			0.393	8.35	Rms	0.89	100	60
3	Layers	3,16,32(X2),4	0.3	8.63	Adam	0.894	100	60
4		3,16,32(X4),4	0.32	10.14	Adam	0.86	100	60
5		3,16,32(X8),4	0.3	11.41	Adam	0.87	100	60
6		3,16,32(X4),4	0.33	5.23	Adam	0.87	100	30
7	Size	3,16,32,64,4	0.33	9.85	Adam	0.88	100	60
8		3,16,64,64,4	0.33	10.03	Adam	0.88	100	60
9	activation	3,16,32, 64(sigmoid),4	1.3863	9.201	Adam	0.48	100	60

Table 8.2 Impact chart Leo Synthetic data Nengo

Data	Parameter	Model 7			Model 9		
		Loss	Time	Accuracy	Loss	Time	Accuracy
Neuron	Softlifrate	0.31	1.5	0.84	0.97	1.53	0.81
	Spikingleakyrelu	0.31	1.6	0.84	0.97	1.63	0.81
Ac	Spikingrelu	0.31	1.34	0.84	0.97	1.37	0.81

	Regularspiking(Tanh)	0.31	1.38	0.845	0.97	1.41	0.81
	Regularspiking(Sigmoid)	0.31	1.37	0.845	0.97	1.40	0.81
	Regularspiking(Relu)	0.31	1.6	0.845	0.97	1.9	0.81
Synaptic Smoothing	0.00001	0.31	1.56	0.845	0.97	1.55	0.67
	0.0001	0.31	1.3	0.845	0.97	1.56	0.667
	0.001	0.31	1.45	0.845	0.97	1.95	0.665
	0.005	0.31	1.91	0.84	0.97	1.77	0.665
	0.01	0.31	1.91	0.8	0.97	1.78	0.660
	0.05	0.31	1.68	0.58	0.97	1.75	0.475
	0.5	0.31	1.55	0.39	0.97	1.88	0.3950
Firing Rate	0.1	0.31	1.52	0.64	0.97	2.01	0.475
	0.5	0.31	1.29	0.84	0.97	1.88	0.5350
	1	0.31	1.48	0.845	0.97	1.51	0.67
	2	0.31	1.47	0.855	0.97	1.91	0.7350
	4	0.31	1.3	0.86	0.97	1.86	0.81
	8	0.31	1.52	0.86	0.97	1.63	0.81
	10	0.31	1.67	0.86	0.97	1.75	0.81
	1000	0.31	1.65	0.86	0.97	1.85	0.81

After conducting several experiments for the same set of layers, Keras showed 88% accuracy in prediction of best link. But Nengo showed only 81%. But when the activation was removed from Keras and added to the layer having maximum neurons, the performance of Keras model deteriorated. The prediction accuracy of Keras model was 41% showing poor performance,

whereas Nengo was able to achieve 84%. When the firing rates were tweaked, the model's accuracy raised to 86%.

Table 8.3 LEO Simulated data in Keras

Model	Parameter	Layers	Loss	Time	Accuracy	Batch	Epoch
5	Layers	3,16,32,4	0.89	1.77	0.389	5	7
6		3,16,32(X2),4	0.4	1.2	0.60	4	8
3	Size	3,16,32,64,4	0.75	0.68	0.66	4	8
7		3,16,32,64,4	0.03	1.5	0.60	3	11
4		3,8,16,32,4	1.15	0.89	0.66	4	8
8	Activation	3,16,32, 64(Sigmoid),4	1.38	1.51	0.66	3	11

Table 8.4 LEO Simulated data in Nengo

Data	Parameter	Model 7			Model 8			Epoch
		Loss	Time	Accuracy	Loss	Time	Accuracy	
Batch size 3	Epochs	0.92	3.1	0.33	1.16	3.1	0.33	8
		0.92	3.6	0.33	0.92	3.6	0.33	11
		0.84	4.06	0.6667	0.92	4.06	0.33	20
Neuron	SoftLIFRate	0.84	1.17	0.6667	0.92	1.31	0.33	20
	SpikingLeakyReLU	0.84	1.400	0.6667	0.92	1.400	0.33	20
Acti	Spikingrelu	0.84	1.22	0.6667	0.92	1.35	0.33	20

	Regularspiking(Tanh)	0.84	1.16	0.6667	0.92	1.34	0.33	20
	Regularspiking (Sigmoid)	0.84	1.05	0.6667	0.92	1.38	0.33	20
	Regularspiking(Relu)	0.84	1.03	0.6667	0.92	1.60	0.33	20
Synaptic Smoothing	0.00001	0.84	1.46	0.6667	0.92	1.44	0.33	20
	0.0001	0.84	1.55	0.6667	0.92	1.43	0.33	20
	0.001	0.84	1.25	0.6667	0.92	1.28	0.33	20
	0.005	0.84	1.77	0.6667	0.92	1.34	0.6667	20
	0.01	0.84	1.44	0.33	0.92	1.10	0.6667	20
Firing Rate	0.1	0.84	1.51	0.00	0.92	1.69	0.6667	20
	0.5	0.84	1.18	0.33	0.92	1.24	0.6667	20
	1	0.84	1.60	0.6667	0.92	1.43	0.333	20
	2	0.84	1.33	0.6667	0.92	1.75	0	20
	4	0.84	1.469	0.6667	0.92	1.57	0	20
	8	0.84	1.39	0.6667	0.92	1.58	0	20
	10	0.84	1.19	0.6667	0.92	1.42	0	20
	1000	0.84	1.45	0.333	0.92	1.27	0	20

For the simulated data, the results obtained are not what was anticipated. Firstly, the amount of data obtained in the simulation for LEO satellite is very less and this might lead to the risk of overfitting in the model. Due to the geographical locations of the receiver, fewer access durations were obtained. A total of only 42 minutes of common contact was achieved where each event represents a reading recorded for 1 minute.

After shuffling the data for training and fitting multiple times with different sets of training data, the Keras model was able to give a prediction accuracy of 60%. But Nengo gave 66.7% of accuracy. After removing the activation from output layer and giving it to the layer with maximum number of neurons, surprisingly, Keras predicted with 66.66% of accuracy but Nengo gave only 33% accuracy. But after tweaking the firing rates to 0.5, SNN was able to achieve 66.67% of accuracy. The model is still not fairly good, due to the low accuracy rate. But it is still better than Keras. As shown from synthetic data, if larger dataset is available, the model might give better results.

8.2. GEO scenario Results

The results for GEO satellite is given in Table 8.1 and Table 8.2 for synthetic data. Refer Table 8.7 and Table 8.8 for results of simulated fata for the selected models. Though initially the loss is high for Model 10, after spiking activities are imparted, the loss reduces drastically.

Table 8.5 GEO synthetic data Keras

Model	Parameter	Layers	Loss	Time	Optimizer	Accuracy	Batch	Epoch
1	Optimizer	3,16,32,4	0.45	7.18	Adam	0.701	100	60
2			0.45	6.88	Rms	0.70	100	60
3	Layers	3,16,32(X2),4	0.44	7.8	Rms	0.70	100	60
4		3,16,32(X4),4	0.453	9.1	Adam	0.70	100	60
5		3,16,32(X8),4	0.453	10.42	Adam	0.68	100	20 (ES)
7		3,16,32(X4),4	0.44	10.42	Adam	0.68	15	10
8	Size	3,16,32,64,4	0.44	8.44	Adam	0.69	100	60
9		16,32(X2),64,4	0.448	12.11	Adam	0.69	100	60

10	Activation	16,32(X2), 64(Sigmoid),4	1.38	1.55	Adam	0.354	100	10

Table 8.6 GEO Synthetic data Nengo

Data	Parameter	Model 9			Model 10		
		Loss	Time	Accuracy	Loss	Time	Accuracy
Neuron	Softlifrate	1.3	2.16	0.29	0.54	1.86	0.805
	Spikingleakyrelu	1.3	2.0	0.29	0.54	1.49	0.805
Activation	Spikingrelu	1.3	1.2	0.29	0.54	1.37	0.805
	Regularspiking(Tanh)	1.3	1.3	0.29	0.54	1.7	0.805
	Regularspiking(Sigmoid)	1.3	1.84	0.29	0.54	1.94	0.805
	Regularspiking(Relu)	1.3	1.5	0.29	0.54	1.48	0.805
Synaptic Smoothing	0.00001	1.3	1.7	0.29	0.54	1.54	0.805
	0.0001	1.3	1.6	0.29	0.54	1.49	0.805
	0.001	1.3	1.56	0.29	0.54	1.669	0.805
	0.005	1.3	1.85	0.285	0.54	1.56	0.80
	0.01	1.3	1.75	0.285	0.54	1.67	0.785
	0.05	1.3	1.75	0.28	0.54	1.67	0.705
	0.5	1.3	1.8	0.15	0.54	1.58	0.445
Firing Rate	0.1	1.3	3.23	0.3333	0.54	2.4	0.78
	0.5	1.3	2.8	0.330	0.54	1.98	0.80
	1	1.3	2.33	0.30	0.54	1.97	0.805

	2	1.3	1.74	0.29	0.54	1.89	0.805
	4	1.3	1.76	0.285	0.54	1.83	0.805
	8	1.3	1.76	0.285	0.54	1.51	0.805
	1000	1.3	1.5	0.285	0.54	1.44	0.805

Table 8.7 GEO simulated data Keras

Model	Parameter	Layers	Loss	Time	Optimizer	Accuracy	Batch	Epoch
1	Optimizer	3,16,32,4	0.031	8.7	Adam	0.99	18	64
2			0.0092	8.69	Rms	1	18	54
3	Layers and size	16,32,64,4	0.069	3.68	Adam	1	18	10
4		3,8,16,32,4	0.041	7.99	Adam	1	18	64
5		3,8,16,32,4	0.096	1.398	Adam	1	100	20 (ES)
6	Activation	3,8,16,32 (Sigmoid),4	1.386	1.07	Adam	0.6	18	10

Table 8.8 GEO Simulated data in Nengo

Data	Parameter	Model 5			Model 6		
		Loss	Time	Accuracy	Loss	Time	Accuracy
Neuron	Softlifrate	0.9702	1.32	0.645	0.1167	1.86	0.97
	Spikingleakyrelu	0.9702	1.59	0.645	0.1167	1.49	0.97
Activation	Spikingrelu	0.9702	1.32	0.645	0.1167	1.37	0.97
	Regularspiking(Tanh)	0.9702	1.73	0.645	0.1167	1.7	0.97
	Regularspiking(Sigmoid)	0.9702	1.37	0.645	0.1167	1.94	0.97

	Regularspiking(Relu)	0.9702	1.48	0.645	0.1167	1.48	0.97
Synaptic Smoothing	0.00001	0.9702	1.70	0.645	0.1167	1.54	0.97
	0.0001	0.9702	1.49	0.645	0.1167	1.49	0.97
	0.001	0.9702	1.81	0.645	0.1167	1.669	0.97
	0.005	0.9702	2.04	0.645	0.1167	1.56	0.98
	0.01	0.9702	1.67	0.645	0.1167	1.67	0.85
	0.05	0.9702	1.67	0.645	0.1167	1.67	0.705
	0.5	0.9702	1.63	0.645	0.1167	1.58	0.445
Firing Rate	0.1	0.9702	1.68	0.645	0.1167	2.4	0.985
	0.5	0.9702	1.35	0.645	0.1167	1.98	0.97
	1	0.9702	1.481	0.645	0.1167	1.97	0.97
	2	0.9702	1.411	0.645	0.1167	1.89	0.97
	4	0.9702	1.39	0.645	0.1167	1.83	0.97
	8	0.9702	1.88	0.645	0.1167	1.51	0.97
	1000	0.9702	1.588	0.645	0.1167	1.44	0.97

The layers are as follows: one input layer with 3 neurons, three hidden layers with 16,32,and 64 neurons and last output layer with 4 neurons for classification. The last layer is a sigmoid layer for the first set of experiments in Keras. Later this activation is moved to the hidden layer having the highest number of neurons. This is the procedure adopted for both the scenarios.

For the GEO satellite, the synthetic data was only able to achieve an average accuracy of 35.4% in Keras. The SNN predicted with 80.5% accuracy (range 78% to 70.5%). When the activation was changed to the hidden layer, the same model predicts with 69.5% accuracy in Keras but SNN predicts with 87% of accuracy. Though this range is low for a model to be called as a good model, the model predicts better than Keras when spiking activities are involved. Coming to simulated dataset, without changing the activations, Keras produced good accuracy of 99% for just 10 epochs than SNN at 52% to 64%. But, when activation was changed, Keras performs very poorly at 56%

but Nengo reaches up to 97% of accuracy in predicting the best link. Further, tweaking the firing rates achieves 98.5% of prediction accuracy performing better than Keras model.

9. CONCLUSION AND FUTURE WORK

Thus, a spiking neural network for optimal decision-making using classification of downlink data to predict the best link was designed with the help of NengoDL framework. The results obtained were very surprising for certain datasets. In most cases, without changing the output activation function, the spiking activities did not produce any improvement in the accuracy unless there was a change in the firing rates in case of LEO scenario synthetic data. But in GEO, there is an increase in prediction accuracy in rage of 2% to 10% when firing rate is increased. Extreme firing rates produce a -2% to -10% range of reduction in prediction accuracy.

Since training a neural network depends on several factors like features of data, parameters like batch size and number of epochs, tweaking these to balance the network, produced better results in Keras. Nengo, although it effectively translated the Keras NN to a Nengo framework, the model did not produce better results for lesser data like that obtained in LEO scenario. It was finally observed that the models produced better results when large dataset was available. Smaller datasets yielded an average accuracy of 66.67% both in Keras and Nengo. But larger dataset produced on an average 84% accuracy in Keras and 87% accuracy in Nengo. Similarly, for the GEO scenario, the model in Keras produced 68%-96% accuracy and a 0.5% increase in SNN. But after changing the output activation to the hidden layer having the most neurons, the results change drastically. It was observed that Keras models perform very poorly when compared to SNN predictions by at least 30% difference. For LEO, the maximum accuracy in SNN is 86% when Keras was just at 41%. For GEO scenario, the prediction of Keras was just 56% while SNN manages to achieve 98.5% of accuracy.

To conclude, from the various experiments conducted using the SNN created using Nengo, it can be observed that the model with spiking activities does produce better predictions compared to

that in Keras depending on the datasets used and the activation functions. Smaller datasets produced good accuracy but high loss. Increase in amount of data produced better accuracy and lower loss. Using many layers of networks produced better results to an extent but increases the training time and reduces the accuracy after a level. SNNs are thus good candidate for intelligence in space applications.

This work reveals several research opportunities where certain aspects require more study. This work can thus be a link for future works focusing on SNNs. Future works in this thesis may focus on performing regression for predicting the next best link for a certain amount of time in the near future. This would require large amount of real satellite downlink data to obtain a good prediction. Secondly, experimenting with link cost reduction methods is a good area to work on. Running evaluations for SNN with more than 3 input links is an interesting experiment. Further, other regularization and optimization techniques can be evaluated to analyze the performance of SNN. There are several such renditions that can be focused on exploiting SNN for intelligent systems in future.

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
NN	Neural network
SNN	Spiking Neural network
STK	Simulation Tool Kit
RF	Radio Frequency
LTP	Licklider Transmission Protocol
GEO	Geosynchronous equatorial orbit
LEO	Low Earth orbit
SNR	Signal to noise ratio
BER	Bit Error Rate
EIRP	Equivalent Isotopically radiated power
CNN	Convolutional neural network
DNN	Deep neural network
LIF	Leaky integrate and fire
GS	Ground station
SP	Shortest path
IoT	Internet of Things
HTP	High Throughput

TF	Tensorflow
BP	Back propagation
SGD	Stochastic Gradient Descent
QPSK	Quadrature Phase Shift Keying
MSE	Mean Squared Error
GPU	Graphic Processing unit
CPU	Central Processing Unit
FPGA	Field programable Gate Array
MNIST	Modified National Institute of Standards and Technology
ML	Machine Learning
BW	Bandwidth
NSD	Noise Spectral Density
RTT	Round Trip Time
ITU	International Telecommunication Unit

REFERENCES

[1] O. Kodheli et al. (2020). Satellite Communications in the New Space Era: A Survey and Future Challenges, in IEEE Communications Surveys & Tutorials, doi: 10.1109/COMST.2020.3028247.

[2] https://towardsdatascience.com/spiking-neural-networks-558dc4479903 [Last accessed: February 23, 2020].

[3] Koji, Nenad & Reljin, Irini & Reljin, Branimir. (2006). Neural Network for Optimization of Routing in Communication Networks. Facta universitatis - series: Electronics and Energetics. doi: 19. 10.2298/FUEE0602317K.

[4] Le Dung, & Mizukawa, M. (2007). A Pattern Recognition Neural Network Using Many Sets of Weights and Biases. International Symposium on Computational Intelligence in Robotics and Automation. doi:10.1109/cira.2007.382856

[5] Di Hu, Xu Zhang, Ziye Xu, Ferrari, S., & Mazumder, P. (2014). Digital implementation of a spiking neural network (SNN) capable of spike-timing-dependent plasticity (STDP) learning. 14th IEEE International Conference on Nanotechnology. doi:10.1109/nano.2014.6968000

[6] Jianhui Han, Zhaolin Li, Weimin Zheng, and Youhui Zhang. (2020). Hardware Implementation of Spiking Neural Networks on FPGA. Tsinghua Science And Technology ISSNll1007-0214 04/10 pp479–486 doi:10.26599/TST.2019.9010019

[7] Izhikevich, E. M. (2003). Simple model of spiking neurons. IEEE Transactions on Neural Networks, 14(6), 1569–1572. doi:10.1109/tnn.2003.820440

[8] Wang, Y., Shahbazi, K., Zhang, H., Oh, K.-I., Lee, J.-J., & Ko, S.-B. (2019). Efficient spiking neural network training and inference with reduced precision memory and computing. IET Computers & Digital Techniques. doi:10.1049/iet-cdt.2019.0115

[9] Jang, H., Simeone, O., Gardner, B., & Gruning, A. (2019). An Introduction to Probabilistic Spiking Neural Networks: Probabilistic Models, Learning Rules, and Applications. IEEE Signal Processing Magazine, 36(6), 64–77. doi:10.1109/msp.2019.2935234

[10] Kim, H., Hwang, S., Park, J., Yun, S., Lee, J.-H., & Park, B.-G. (2018). Spiking Neural Network Using Synaptic Transistors and Neuron Circuits for Pattern Recognition With Noisy Images. IEEE Electron Device Letters, 39(4), 630–633. doi:10.1109/led.2018.2809661

[11] Shukla, A., Kumar, V., & Ganguly, U. (2017). A software-equivalent SNN hardware using RRAM-array for asynchronous real-time learning. 2017 International Joint Conference on Neural Networks (IJCNN). doi:10.1109/ijcnn.2017.7966447

[12] Maxence Bouvier, Alexandre Valentian, Thomas Mesquida, Francois Rummens, Marina Reyboz, Elisa Vianello, And Edith Beigne, Cea-Leti, (2019). Spiking Neural Networks Hardware Implementations and Challenges: A Survey, Minatec Campus J. Emerg. Technol. Comput. Syst., Vol. 15, No. 2, Article 22. doi: https://doi.org/10.1145/3304103

[13] Harkin, J., McDaid, L., Hall, S., Dowrick, T., & Morgan, F. (2008). Programmable architectures for large-scale implementations of spiking neural networks. IET Irish Signals and Systems Conference (ISSC 2008). doi:10.1049/cp:20080691

[14] Jang, H., Park, A., & Jung, K. (2008). Neural Network Implementation Using CUDA and OpenMP. Digital Image Computing: Techniques and Applications. doi:10.1109/dicta.2008.82

[15] Xie, E., McGinnity, M., Wu, Q., Cai, J., & Cai, R. (2011). GPU implementation of spiking neural networks for color image segmentation. 4th International Congress on Image and Signal Processing. doi:10.1109/cisp.2011.6100451

[16] Jang, H., & Simeone, O. (2019). Training Dynamic Exponential Family Models with Causal and Lateral Dependencies for Generalized Neuromorphic Computing. ICASSP 2019 - 2019

IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). doi:10.1109/icassp.2019.8682960

[17] Yudanov, D., & Reznik, L. (2012). Scalable multi-precision simulation of spiking neural networks on GPU with OpenCL. The 2012 International Joint Conference on Neural Networks (IJCNN). doi:10.1109/ijcnn.2012.6252440

[18] Roy, A., Venkataramani, S., Gala, N., Sen, S., Veezhinathan, K., & Raghunathan, A. (2017). A Programmable Event-driven Architecture for Evaluating Spiking Neural Networks. IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED). doi:10.1109/islped.2017.8009176

[19] Chou, T.-S., Kashyap, H. J., Xing, J., Listopad, S., Rounds, E. L., Beyeler, M., … Krichmar, J. L. (2018). CARLsim 4: An Open Source Library for Large Scale, Biologically Detailed Spiking Neural Network Simulation using Heterogeneous Clusters. 2018 International Joint Conference on Neural Networks (IJCNN). doi:10.1109/ijcnn.2018.8489326

[20] Camunas-Mesa, L. A., Linares-Barranco, B., & Serrano-Gotarredona, T. (2019). Low-power hardware implementation of SNN with decision block for recognition tasks. 2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS). doi:10.1109/icecs46596.2019.8964964

[21] Vu, T. H., & Abdallah, A. B. (2019). Low-Latency K-Means Based Multicast Routing Algorithm and Architecture for Three Dimensional Spiking Neuromorphic Chips. 2019 IEEE International Conference on Big Data and Smart Computing (BigComp). doi:10.1109/bigcomp.2019.8679363

[22] L. Blin, A. J. Awan and T. Heinis (2018). Using Neuromorphic Hardware for the Scalable Execution of Massively Parallel, Communication-Intensive Algorithms. 2018 IEEE/ACM

International Conference on Utility and Cloud Computing Companion (UCC Companion), Zurich, Switzerland, pp. 89-94, doi: 10.1109/UCC-Companion.2018.00040.

[23] Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J. M., Diesmann, M., Morrison, A., Goodman, P. H., Harris, F. C., Jr, Zirpe, M., Natschläger, T., Pecevski, D., Ermentrout, B., Djurfeldt, M., Lansner, A., Rochel, O., Vieville, T., Muller, E., Davison, A. P., … Destexhe, A. (2007). Simulation of networks of spiking neurons: a review of tools and strategies. Journal of computational neuroscience, 23(3), 349–398. https://doi.org/10.1007/s10827-007-0038-6.

[24] Jirayut Poksawat,(2017). Comparison of GPU-Enhanced Spiking Neural Network Simulators, The Department of Engineering Technology University of Houston.

[25] Shihui Yin and Shreyas K. Venkataramanaiah and Gregory K. Chen and Ram Krishnamurthy and Yu Cao and Chaitali Chakrabarti and Jae-sun Seo (2017). Algorithm and Hardware Design of Discrete-Time Spiking Neural Networks Based on Back Propagation with Binary Activations. IEEE Biomedical Circuits and Systems (BioCAS).

[26] Bohte, Sander & Kok, Joost & Poutré, Johannes. (2000). SpikeProp: backpropagation for networks of spiking neurons. ESANN. 48. 419-424.

[27] Matteo Cartiglia, Germain Haessig and Giacomo Indiveri, (2020). An error-propagation spiking neural network compatible with neuromorphic processors. IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS), doi: 10.1109/AICAS48895.2020.9073856.

[28] S. Tamura et al.,(2016). Multiplex communication by BP learning in neural network, 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Datong, pp. 825-828, doi: 10.1109/CISP-BMEI.2016.7852824.

[29] Wu, Yujie, et al. (2018). Spatio-Temporal Backpropagation for Training High-Performance Spiking Neural Networks. Frontiers in Neuroscience, Gale In Context: Science. https://link.gale.com/apps/doc/A540007545/SCIC?u=txshracd2588&sid=SCIC&xid=b4dbd7b7.

[30] Hou, Z., Yi, X., Zhang, Y., Kuang, Y., & Zhao, Y. (2019). Satellite-Ground Link Planning for LEO Satellite Navigation Augmentation Networks. IEEE Access, 7, 98715–98724. doi:10.1109/access.2019.2930626

[31] Veronica Toro-Betancur, Augusto Carmona Valencia, and José Ignacio Marulanda Bernal. (2020). Signal Detection and Modulation Classification for Satellite Communications. In Proceedings of the 2020 3rd International Conference on Signal Processing and Machine Learning (SPML 2020). Association for Computing Machinery, New York, NY, USA, 114–118. DOI:https://doi.org/10.1145/3432291.3432297

[32] Lent, R., Brooks, D. E., & Clark, G. (2020). Validating the Cognitive Network Controller on NASA's SCaN Testbed. ICC 2020 - 2020 IEEE International Conference on Communications (ICC). doi:10.1109/icc40277.2020.9149160

[33] Zhenyu Na, Zheng Pan, Xin Liu, Zhian Deng, Zihe Gao, Qing Guo. (2018). Distributed Routing Strategy Based on Machine Learning for LEO Satellite Network, Wireless Communications and Mobile Computing, vol. 2018, Article ID 3026405, 10 pages, 2018. https://doi.org/10.1155/2018/3026405

[34] Lent R (2019) Analysis of the Block Delivery Time of the Licklider Transmission Protocol, IEEE Trans Commun; doi:10.1109/TCOMM.2018.2875717

[35] https://www.gaussianwaves.com/2012/07/intuitive-derivation-of-performance-of-an-optimum-bpsk-receiver-in-awgn-channel/ [Last accessed: July 25, 2012]

[36] https://en.wikipedia.org/wiki/Eb/N0 [Last Accessed: January 22, 2021]

[37] https://www.itu.int/dms_pubrec/itu-r/rec/s/R-REC-S.1328-3-200102-S!!PDF-E.pdf [Last Accessed: February 20, 2001]

[38] Haddad, Wassim M.; Hui, Qing; Bailey, James M. (2014). Human Brain Networks: Spiking Neuron Models, Multistability, Synchronization, Thermodynamics, Maximum Entropy Production, and Anesthetic Cascade Mechanisms, Entropy 16, no. 7: 3939-4003. https://doi.org/10.3390/e16073939

[39] Blouw, Peter & Choo, Xuan & Hunsberger, Eric & Eliasmith, Chris. (2019). Benchmarking Keyword Spotting Efficiency on Neuromorphic Hardware. 1-8. 10.1145/3320288.3320304.

[40] Gennadi Bersuker, Maribeth Mason, and Karen L. Jones (2018). Game Changer - Neuromorphic Computing: The Potential for High-Performance Processing In Space, Center For Space Policy And Strategy.

[41] Miguel Ángel Vázquez et al. (2019). On the Use of AI for Satellite Communciaitons, Centre Tecnològic de Telecomunicacions de Catalunya, Castelldefels (Spain).

[42] Mowei Wang, Yong Cui, and Shihan Xiao are with Tsinghua University (2017) Yong Cui is the corresponding author. Xin Wang is with Stony Brook University. Junchen Jiang is with Carnegie Mellon University. doi: 10.1109/MNET.2017.1700200

[43] Langlet, F & Abdulkader, Hasan & Roviras, Daniel & Lapierre, L & Castanie, Francis. (2001). Comparison of neural network natural and ordinary gradient algorithms for satellite down link identification. Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on. 2. 10.1109/ICASSP.2001.941164.

[44] Koji, Nenad & Reljin, Irini & Reljin, Branimir. (2006). Neural Network for Optimization of Routing in Communication Networks. Facta universitatis - series: Electronics and Energetics. 19. 10.2298/FUEE0602317K.

[45] Yan-Jie Song, Bing-Yu Song, Zhong-Shan Zhang , And Ying-Wu Chen, (2018). The Satellite Downlink Replanning Poblem: A BP Neural Network and Hybrid algorithm approach for IoT Connection, IEEE Access, Special Section On Collaboration For Internet Of Things. doi: 10.1109/ACCESS.2018.2855800

[46] Juan Jose Garau Luis et al. (2019). Deep Reinforcement Learning Architecture for Continuous Power Allocation in High Throughput Satellites, Reinforcement Learning for Real Life (RL4RealLife) Workshop in the 36 th International Conference on Machine Learning, Long Beach, California, USA.

[47] https://www.nengo.ai/nengo-dl/examples/keras-to-snn.html [Last Accessed: November 26, 2020]

[48] https://towardsdatascience.com/building-our-first-neural-network-in-keras-bdc8abbc17f5 [Last Accessed: June 26, 2019]

[49] https://www.youtube.com/watch?v=FTHt-c8hWKw&t=916s&ab_channel=SpaceChallenges [Last Accessed: Sep 25, 2018]

[50] http://innospacecomm.com/project/link-budget/ [Last Accessed: Feb 7, 2019]

[51] https://en.wikipedia.org/wiki/Transmission_time [Last Accessed: March 15, 2021]