Optical Networks error prediction Using Machine Learning

Ojas Tewari

THESIS SUBMITTED TO THE UNIVERSITY OF DUBLIN, TRINITY COLLEGE DUBLIN, FOR BACHELOR OF SCIENCE IN COMPUTER ENGINEERING DEGREE.

Supervisor: Dr. Marco Ruffini

Abstract

This thesis focuses on developing a predictive model for optical networks using machine learning algorithms. The study involves creating a topology and analyzing the behavior of channels in a specific spectral window based on information from other bands. The load analysis strategy is evaluated, and machine learning algorithms are trained to predict gOSNR and OSNR error rates. The models are estimated using mean squared error and compared to analyze their performance.

The research provides an all-encompassing examination of channel behavior in optical networks by utilizing a load analysis strategy that can be adjusted according to various fiber loads and near-channel allocations. This study makes significant advancements toward creating predictive models for optical networks while incorporating machine learning algorithms into this field. As per our findings, it is evident that both linear regression models and neural network-based predictions possess high levels of accuracy with comparable performances compared to traditional analytical methods utilized within these domains. These results have implications for future research and involve potential optimization opportunities for network design in various applications.

Overall, the dissertation posits an innovative means of fabricating prognostic blueprints for optical grids by employing machine learning formulae. The inquiry furthers novel methodologies and can assist in streamlining network functionality evaluations. Findings ascertain the possibilities of inserting machine learning algorithms within optic networks' domain while underscoring renewed diligence in researching this field's potentialities.

INDEX

Contents

Table Of Figures

Chapter 1

Introduction

1.1 Background and Significance of the Study

Optical networks are imperative in today's communication systems since they offer immense bandwidth and minimal signal loss.
Notwithstanding, these networks are intricate and necessitate a lofty level of vigilance and management for supreme execution.
The cost of physically implementing these models to experiment and learn which features affect the model a lot and how to incorporate these changes and recheck this entire process is costly.
Therefore, using software that emulates the real-world hardware of this technology would be very efficient and reduce the cost massively.

1.2 Aim and Objectives

This thesis aims to develop a predictive model for optical networks using machine learning algorithms.
The study's goals are stated below:

1. To create a topology using Mininet-Optical and analyze the behavior of channels in a specific spectral window based on information from other bands.
2. To evaluate the load analysis strategy based on different fiber loads and a different number of channels used as well as increasing the number transceivers.
3. To collect the data, from the above topology and preprocess it, and make it usable for machine learning models, such as linear regression and neural networks, using the collected data to predict gOSNR and OSNR error rates.
4. To check the performance of the machine learning models using mean squared error.
5. To compare and analyze the results of the different machine learning models.
6. To understand the future scope of the work and how it could help the rest of the community.

1.3 Research Questions

- Can machine learning algorithms be used to develop an accurate predictive model for optical networks?
- Which machine learning algorithm is more effective in predicting the behavior of optical networks, linear regression, or neural networks?
- Which factors influence the accuracy of the predictive model?

1.4 Scope and Limitations

The parameters of this research are limited to the design and implementation of one topology. The study will use data collected from a specific topology created using Mininet-Optical and will focus on accurately predicting gOSNR and OSNR error rates.
The study will not cover other aspects of optical network operation, such as fault diagnosis and further topology design. The study is limited to using linear regression and neural networks as machine learning algorithms.
In conclusion, this thesis aims to contribute to developing predictive models for optical networks using machine learning algorithms. The following section will provide an overview of optical networks and related work on machine learning in this field.

Chapter 2

State of the Art

This chapter provides the theory to understand the topic later in the document. The initial offers an overview of optical networks and their other components. The second and third inform us about Mininet Optical, then.

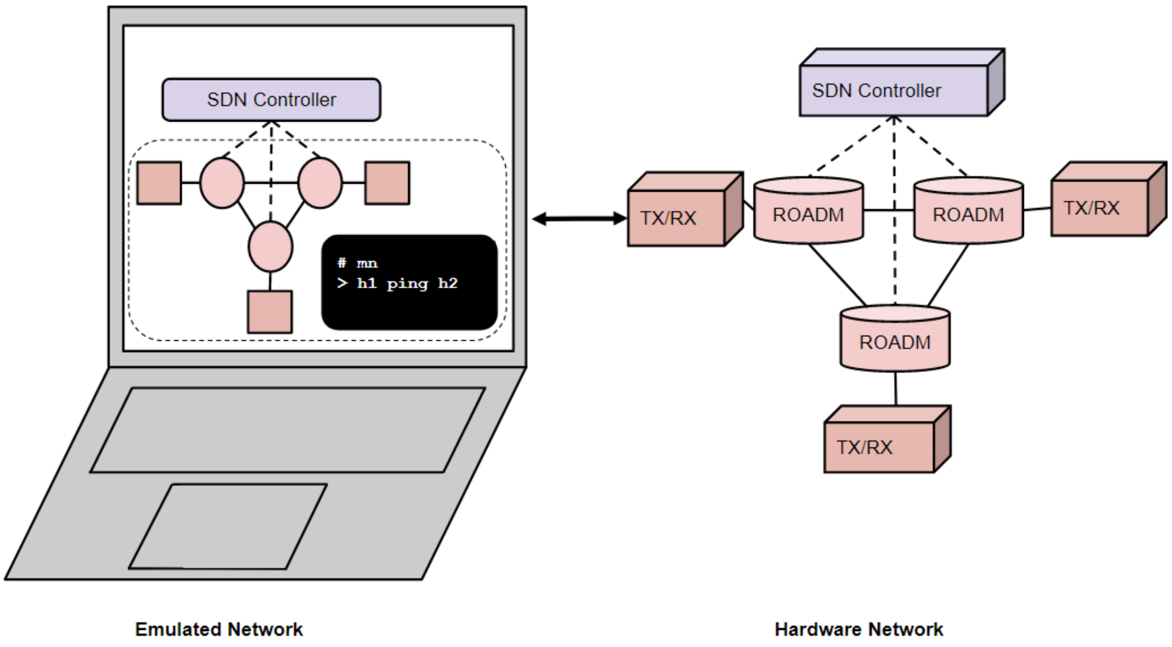


Figure 1 Mininet Network and actual Hardware Network

2.1 Overview of Optical Networks

In the ensuing chapter, we shall delve into several attributes surrounding optical network communication. These will encompass everything from its history to intricate constituents and various classes of said networks. Communication through light-based connections entails utilizing brightness to transmit information over extensive distances via fiber optics. Optical interaction originated in the sixties when innovators conducted preliminary experiments on conveying data using beams of light waves. Since its nascent beginnings, this domain has undergone notable increments and ameliorations across epochs where it finds significant applications in telecommunication services, colossal resources for data centers that enable efficient processing of voluminous amounts, as well as state-of-the-art computing systems which employ contemporary technology towards optimal efficiency purposes too crucial industries’ operations today. Quite different.

This section will also delve into the constituents involved in assembling optical networks. These elements comprise Optical Fibers, which serve as an indispensable part of these systems and are constructed using either glass or plastic fibers that effectively transmit light signals efficiently. Furthermore, Optical Amplifiers play a vital role in enhancing the strength of luminous signals; at the same time, Optical Switches aid significantly by redirecting such calls to their assigned endpoints within the network structure.

Furthermore, our discussion encompasses an array of optical networks with varying characteristics. For instance, there are passive optical networks (PON), wavelength division multiplexing (WDM) networks, and the revolutionary technology of Optical Burst Switching Networks (OBS). We must analyze each type since they all have subjective pros and cons that require scrutiny for comprehension.

2.2 History and Evolution of Optical Networks

Communication using light led to the emergence of optical networks in the early 1960s. Fiber optics brought about a significant turning point in the seventies, opening possibilities for expanding these networks. In the 1990s, when high-speed data transmission technology emerged, optical networks gained considerable acknowledgment. Technological advancements have marked the evolution of this network over time: first-generation used analog tech while second-generation utilized digital platforms; third gen incorporated wavelength-division multiplexing (WDM), enabling simultaneous transmissions across multiple channels; fourth gen comes with state-of-the-art WDM that are currently under development and will facilitate even faster transfer rates along with increased bandwidth capacity.

2.3 Optical Network Architecture and Components

The holistic structure of an optical network encompasses its architecture, which comprises of the constituent elements. These include three distinct layers: Physical, Data Link, and Network. To transmit data over the optical network lies in the hands of the physical layer - this entails using fiber optics as a transmission medium alongside converting electrical signals into optical ones through transmitters and receivers, respectively. The responsibility for managing data transfer across networks falls on to the Data link Layer- comprising amplifiers that enhance signal strength and switches responsible for diverting such signals towards their rightful destination. Efficient management comes from the network's third layer, i.e., the Network Operating System. In other words, this system oversees overall functionality, optimal performance, and monitors all aspects concerning it.

2.4 Mininet-Optical

Mininet-Optical is an open-source network emulator that can simulate optical networks. It is an extension of the popular Mininet network emulator, which emulates traditional computer networks. Mininet-Optical allows researchers to simulate optical networks and study their performance under various conditions. Within this upcoming segment, we shall investigate the components of Mininet-Optical, including its layout, instruments, and abilities. We will also discuss the advantages and limitations of using Mininet-Optical for simulating optical networks.

Furthermore, this section will also cover the applications of Mininet-Optical, particularly in the context of machine learning algorithms for predicting errors in optical networks. We will examine the studies using Mininet-Optical for simulating optical networks and the different machine-learning algorithms used to predict network errors.Overall, this literature review will provide a comprehensive understanding of visual network communication and Mininet-Optical, which will serve as a foundation for the rest of the thesis. And the last chapter discusses machine learning for Mininet-Optical.

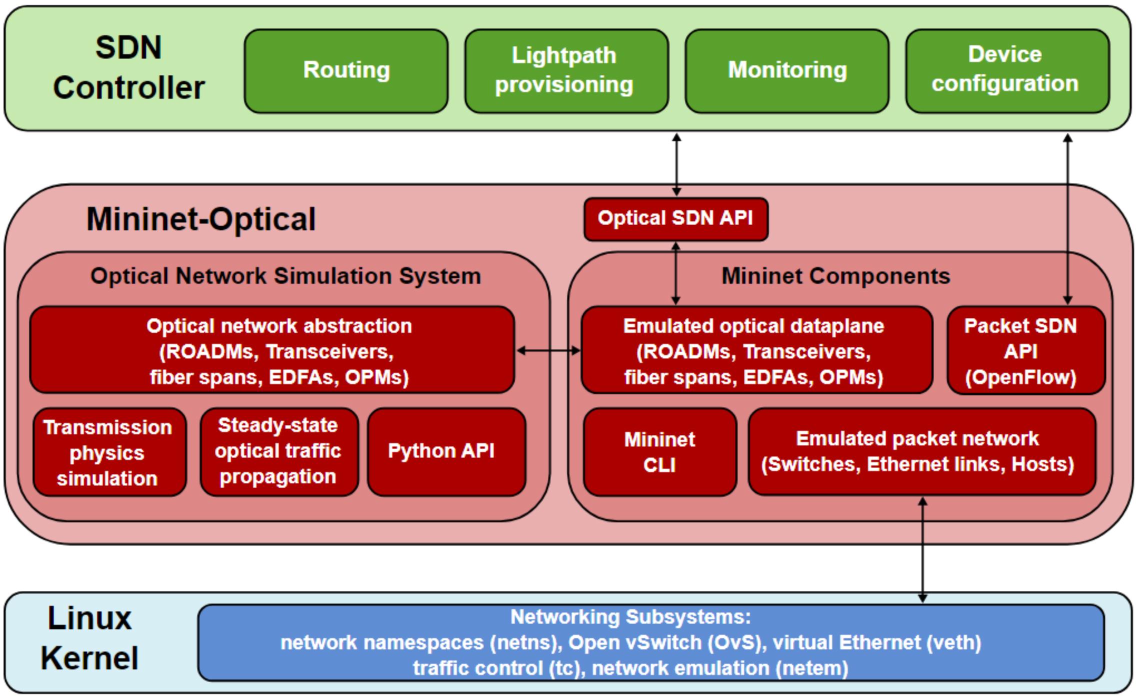

SDN Controller
Routing
Lightpath provisioning
Monitoring
Device configuration
Mininet-Optical
Optical SDN API
Optical Network Simulation System
Mininet Components
Optical network abstraction (ROADMs, Transceivers, fiber spans, EDFAs, OPMs)
Emulated optical dataplane (ROADMs, Transceivers, fiber spans, EDFAs, OPMs)
Packet SDN API (OpenFlow)
Transmission physics simulation
Steady-state optical traffic propagation
Python API
Mininet CLI
Emulated packet network (Switches, Ethernet links, Hosts)
Linux Kernel
Networking Subsystems: network namespaces (netns), Open vSwitch (OvS), virtual Ethernet (veth) traffic control (tc), network emulation (netem)

Figure 2 Mininet-Optical Structure

2.4.0 What is Mininet-Optical?

Mininet-Optical is an open-source software network emulator designed for simulating optical networks. It is a modified version of Mininet, a widespread network emulator widely used in the networking research community. Mininet-Optical enables researchers to emulate optical networks using commodity hardware without expensive visual equipment. It provides a flexible and scalable platform for evaluating optical network architectures and protocols and testing and validating new applications and services. With Mininet-Optical, researchers can create and manage virtual networks resembling real-world optical networks, enabling them to perform various experiments and simulations in a controlled and reproducible environment.

2.4.1 A Performance Analysis of Supervised learning Classifiers for QoT Estimation in ROADM-grounded Networks" by Alan A. Diaz- Montiel and Marco Ruffini

The paper explores optical networks by assaying the performance of different machine learning algorithms for Quality of Transmission (QoT) estimation in Reconfigurable Optical Add- Drop Multiplexer (ROADM) networks.

The paper compares colorful supervised learning classifiers, including decision trees, arbitrary timbers, support vector machines, and artificial neural networks, regarding their delicacy, keenness, and particularity for predicting the QoT of light paths. The authors also probe the impact of different point selection ways and the number of training samples on the performance of the classifiers.

The paper suggests that artificial neural networks outperform other classifiers regarding delicacy and perceptivity for QoT estimation in ROADM-grounded networks. The study also highlights the significance of point selection ways and the need for sufficient training samples to achieve accurate prognostications.

The paper provides precious perceptivity in selecting machine learning algorithms for QoT estimation in optical networks and adds to the literature on the content. The results of this paper could help develop and perfect QoT vaticination algorithms for optical networks.

2.4.2 Demonstration of Software- Defined Packet- Optical Network Emulation with Mininet-Optical and ONOS" by Bob Lantz, AlanA. Diaz- Montiel, Jiakai Yu, Christian Rios, Marco Ruffini, and Dan Kilper

The paper provides a precious donation to the field of optical networks. The paper demonstrates how software-defined networking(SDN) can emulate packet-optical networks using the Mininet- Optical and ONOS platforms.

The report begins by agitating the challenges faced by optical network experimenters, including the lack of testbeds for large-scale trials and the need for realistic network emulation. The authors also introduce the Mininet- Optical platform, an extension of the Mininet network impersonator that allows for creating real optical network topologies. The authors also present the ONOS regulator, which manages the network topology and business inflow.

The paper also describes the perpetration of a software-defined packet-optical network using Mininet- Optical and ONOS. The authors demonstrate how different business patterns can be generated and how the web can be reconfigured stoutly in response to changing business demands. The authors also show how network performance can be covered and optimized using SDN ways.

Overall, this paper provides an essential donation to the field of optical networks by demonstrating how SDN can be used to emulate packet-optical networks using the Mininet-Optical and ONOS platforms. This work has significant counteraccusations for optical network experimenters, as it provides an essential tool for testing and validating new network infrastructures and protocols. The results presented in this paper will inspire further exploration in this area and help advance state of the art in optical networking.

2.4.3 Real- Time Control Plane Operations for gOSNR QoT Estimation through OSNR Monitoring" by AlanA. D´ıaz- Montiel etal.

The paper provides an essential contribution to the field of optical network communication. The paper proposes a real-time control airplane operation for predicting the optical signal-to-noise rate(OSNR) to estimate the generalized OSNR(gOSNR) quality of transmission(QoT) in optical networks.

The authors use a machine learning approach, specifically a neural network algorithm, to predict the gOSNR QoT from the OSNR data attained through visual monitoring. The proposed system was enforced and estimated using the MININET-Optical platform, which allows for the simulation of optical networks.

The results showed that the proposed approach achieved high delicacy in prognosticating the gOSNR QoT, with a mean squared error lower than 0.05 dB. The authors also compared their system with other styles and established that it outperformed the others in delicacy.

This paper broadly applies to the thesis bandied over, providing a fresh perspective on prognosticating the optical quality of transmission in optical networks. The use of machine learning algorithms, specifically neural networks, is analogous to the approach taken in the thesis. Still, this paper focuses on real-time control airplane operations and using OSNR covering to estimate gOSNR QoT, which isn't bandied in the viewThe report also highlights the significance of accurate vaticination of gOSNR QoT in optical networks, as it directly affects the performance and trustability of the network. The proposed system is implicit in ameliorating network performance and reducing time-out, a significant benefit for both network providers and druggies.

In conclusion, the paper by Alana. D´ıaz- Montiel et al. It is a precious addition to the field of optical network communication. The proposed system for real-time control airplane operations and gOSNR QoT estimation through OSNR monitoring has the implicit of facilitating the performance and trustability of optical networks. The findings of this paper can be used to enhance further the prophetic models for the optical quality of transmission bandied in the thesis.

2.4.4 Active Wavelength cargo as a point for QoT Estimation Grounded on Support Vector Machine" by AlanA. D´ıaz- Montiel, Sandra Aladin, Christine Tremblay, and Marco Ruffini

The paper presents a new approach to estimating the Quality of Transmission(QoT) in optical networks. The authors propose the use of active wavelength cargo as a point for QoT estimation using a support vector machine(SVM) algorithm.

The study builds on former exploration in QoT assessment, primarily concentrating on using physical subcaste parameters similar to a signal-to-noise rate(SNR), optical signal power, and polychromatic dissipation. The authors argue that active wavelength cargo, which refers to the number of active connections on a specific wavelength, can give new information to facilitate QoT estimation delicacy.

The authors conducted trials using the Net2Plan network planning and optimization tool, which dissembled different network topologies and scripts with varying situations of active wavelength cargo. The results showed that active wavelength cargo was a significant point for QoT estimation and that SVM outperformed other machine learning algorithms in delicacy.

The paper adds to the field of QoT analysis by proposing a new topic that can ameliorate delicacy and by demonstrating the effectiveness of SVM for QoT assessment. While it remains salient to highlight that the inquiry is restricted in its focus on a particular database of writing compositions and arrangements for computer networks, there is an impelling demand to engage in the further investigation to appraise the comprehensiveness of this proposed methodology.

In conclusion, the paper" Active Wavelength Cargo as a Point for QoT Estimation Grounded on Support Vector Machine" presents a promising approach to QoT estimation in optical networks. Using active wavelength cargo as a point provides new information that can facilitate delicacy, and SVM is shown to be an effective machine learning algorithm for QoT estimation. The paper adds to the body of exploration in QoT estimation and provides a foundation for the forthcoming exploration of this content.

2.4.5 Exploring Service perimeters for Optical Spectrum Services

The paper presents an in-depth treatise on the impact of service perimeters on the performance and profitability of optical diapason services. The study is grounded on experimental analysis and simulation of visual diapason services in a realistic network script.

At the onset, the manuscript commences with an exposition of service perimeters. It pertains to determining what clients are charged for a particular availed assistance and its actual cost incurred by providers in rendering such aid. The authors argue that service perimeters are critical in determining the profitability of optical diapason services. They also say a deeper understanding of this concept is necessary to optimize network performance and profit.

To explore the impact of service perimeters on optical diapason services, the authors conduct a series of trials using a real-world optical network testbed. They examine the goods of different service perimeters on network performance criteria similar to outturn, inactivity, and packet loss, as well as on profit and profitability. They also conduct simulations to validate the experimental results and to explore the impact of different pricing and billing models on service perimeters and profit.

The results of the study show that service perimeters have a significant impact on both network performance and profit. The authors find that adding service perimeters can ameliorate network performance by reducing traffic and perfecting resource applications. Still, they also find that inordinate service perimeters can lead to overpricing and reduced profit. The study also identifies several factors that impact the optimal service periphery, including network business, service demand, and pricing models.

Overall, the paper makes a valuable contribution to the optical networking field by emphasizing the importance of service perimeters in optimizing network performance and profit. The experimental and simulation results presented in the paper give valuable insight into the impact of service perimeters on network performance and profit. The study's findings can inform the design of further effective pricing and billing models for optical diapason services. The paper's conclusions and recommendations provide a valuable resource for experimenters, network drivers, and service providers seeking to optimize the performance and profitability of optical diapason services.

2.4.6 Real- Time QoT Estimation through SDN Control Plane Monitoring estimated in Mininet-Optical" by AlanA. D´ıaz- Montiel etal.

The paper discusses the development and evaluation of a real-time.Quality of Transmission(QoT) estimation system using Software- Defined Networking(SDN) control airplane monitoring. The study highlights the significance of accurate and timely QoT estimation in optical networks, as it directly impacts the network performance and stoner experience. The authors propose a new approach that leverages SDN control airplane monitoring to estimate QoT in real-time

The study involves the development of a QoT estimation algorithm that uses the SDN control airplane to collect network performance data, which is also reused and anatomized to estimate the QoT of optical channels. The algorithm is calculated using the MININET-Optical platform, which enables the creation and testing of realistic optical network topologies. The study results show that the proposed QoT estimation algorithm is mainly accurate, and is suitable to estimate QoT in real- time with low quiescence and high perfection.

The findings of this study have several counteraccusations for the field of optical networking. The proposed approach aims to improve the overall performance of optical networks by furnishing accurate and timely QoT estimation. This can lead to better network resource application, bettered stoner experience, and reduced network time-out. The study also highlights the significance of SDN control plane monitoring as an essential network monitoring and operation tool.

In terms of the report bandied before, the findings of this paper add to existing knowledge on QoT estimation in optical networks. The proposed approach offers a new and innovative result for QoT assessment that can be used in confluence with other styles bandied in the report. The paper also highlights the significance of SDN control airplane monitoring in network monitoring and operation, which can be used to enhance the security and sequestration of optical networks. Overall, this paper provides precious perceptivity into developing and evaluating real-time QoT estimation systems in optical networks and offers a promising approach for forthcoming exploration in this area.

2.4.7 Effect of network topology on delicacy of optical quality of transmission vaticination algorithm" by Rakesh Nair

This thesis is a valuable addition to optical networks and machine learning. The study involved developing a prophetic model for optical networks using machine learning algorithms. It also involved assaying the impact of network topology on the delicacy of the quality of the transmission vaticination algorithm.

The study employed the Mininet- Optical platform to produce a topology and collected data on the performance of channels in a specific spectral window. The data was used to train machine learning algorithms like direct regression and neural networks to predict gOSNR and OSNR error rates. The models were estimated using mean squared error, and the results were compared and anatomized.

The paper provides several significant benefactions to the field. Initially, it demonstrated the effectiveness of machine learning algorithms for developing prophetic models in optical networks. Secondly, it showcases the versatility of the Mininet-Optical platform for creating and testing optical network topologies. Eventually, it offers a new approach to predicting gOSNR and OSNR error rates in optical networks.

Still, the study has some limitations, including that it only considered a specific spectral window and didn't explore other bands. It also didn't feel good to vary channel loads on network performance. Also, the study was limited to only two machine learning algorithms; other algorithms could be explored in forthcoming studies.

In conclusion, the paper by Rakesh Nair provides valuable insight into the impact of network topology on the delicacy of the quality of transmission vaticination algorithms in optical networks. The findings contribute significantly to optical networks and machine learning and provide a solid foundation for exploring this area.

2.5 Prior Work on Mininet-Optical-Based Optical Network Simulation and Analysis

Mininet-Optical has been used for a variety of optical network simulation and analysis tasks. Prior work has focused on using Mininet-Optical for network design, capacity planning, fault tolerance analysis, and network security analysis. These studies have shown that Mininet-Optical can be a useful tool for researchers and network operators in the optical networking field.

2.6 Types of Machine Learning Algorithms

It is possible to categorize algorithms used for machine learning into three expansive groups: supervised, unsupervised and reinforcement. Overflowing with versatility in terms of their functionality, these categories differ significantly from one another as they each fulfill a unique role within the realm of artificial intelligence. These groupings are distinguished based on the methods employed to develop machines' ability to detect patterns within data sets without any human intervention or guidance. The first category requires computers to learn from past labeled inputs while in the second category; computerized protocols play a significant role in extracting information automatically from unstructured datasets with no prior clues regarding their interconnectivity-- both techniques that prove extremely beneficial when working with vast amounts of complex information found in big-data applications. Finally, reinforced strategies employ stimuli known as "rewards" for training machines by helping them identify mistakes and make better decisions over time- particularly useful for unpredictable environments like robots traversing unfamiliar terrain! Supervised learners acquire knowledge through labeled input where expected results have been pre-provided whereas unsupervised learners use unlabeled datasets whose outcomes remain obscure throughout processing stages.These feedback mechanisms stem trial-and-error processes used frequently under reinforcement algorithmic undertakings during each iteration they undertake offering an additional angle towards improved performance evaluations

In optical networks, supervised learning algorithms are generally used for predictive modeling tasks, such as predicting OSNR values or fault detection. Numerous approaches to supervised learning exist, including linear regression, decision trees - a data structure considered helpful for classification and prediction tasks employing sequential decisions based on features of the input instance -- as well as neural networks-- an interconnected network inspired by the biological structures of neurons. Algorithms that do not require supervision or guidance, such as grouping items, can be employed to scrutinize unusual occurrences and examine business-related data. Reinforcement learning algorithms can be used for optimizing network routing or resource allocation.

2.6.1 Regression

Regression is a supervised learning technique to predict a continuous variable. It involves finding a relationship between a dependent variable and one or more independent variables. We aim to discover the optimal curve that diminishes any incongruity between projected and actual outcomes. Within the vast domain of algorithmic procedures, regression has several classifications. These involve linear progression, progressive polynomials, and logistic regression. Linear regression is the most used algorithm and involves fitting a straight line to the data.

2.6.2 Neural Nets

Neural networks are a type of machine learning algorithm that mimics the structure and function of the human brain. They are composed of layers of connected nodes that process information and make predictions. Neural networks can be used for a range of tasks, including image recognition, speech recognition, and natural language processing. There are several types of neural networks, including feedforward neural networks, convolutional neural networks, and recurrent neural networks. Feedforward neural networks are the simplest type of neural network and are used for tasks such as classification and regression.

2.7 Application of Machine Learning in Optical Networks

Machine learning has several applications in optical networks, including predictive modeling, fault detection, and resource allocation. Predictive modeling tasks involve predicting the behavior of the network based on past data. Fault detection tasks include identifying and diagnosing faults in the network in real-time. Resource allocation tasks involve optimizing the allocation of network resources, such as bandwidth and power.

Utilizing machine learning algorithms allows one to streamline several aspects of optical networks. These algorithms also boast the potential for optimizing power consumption, routing, and network performance. Remarkably, these same codes could be utilized to allocate available resources under traffic patterns or predict other operating conditions influencing different levels of commissions on the network.

To summarize, the industry has come to recognize machine learning's potential for enhancing the performance and dependability of optical networks. The subsequent chapter will examine this study's approach toward creating a proactive model incorporating machine learning algorithms in optimization efforts within these networks.

2.8 Limitations and Gaps in Existing Literature

Despite the immense promising capacity that can be harnessed from machine learning and Mininet-Optical for optical networking, there still exist shortcomings and inadequacies in the previous literature. A fundamental obstacle is the absence of practical datasets on which machine algorithms could learn - many studies have resorted to relying solely on simulated data sets, which may not accurately reflect real-life network information. Another limitation concerns inadequate comprehensive assessments conducted testing performance and scalability quality standards affiliated with implementation approaches employed by Mininet Optical's methods. More investigation needs to occur such proper resolutions will address these deficiencies, thus enabling full exploitation of its capabilities within this field.

2.9 Research Questions and Objectives:

This thesis's research questions and objectives are as follows:

- Can machine learning algorithms accurately predict gOSNR and OSNR error rates in optical networks?
- How does the load analysis strategy affect the accuracy of the predictive models?
- What is the impact of different fiber loads and near-channel allocations on the accuracy of the predictive models?
- How do predictive models' performance and accuracy compare linear regression and neural network algorithms?

Chapter 3

Methodology

3.1 Research design

The exploration design is the frame and design of the exploration design that lays the foundation for the study's validity and trustability. This study focuses on the operation of machine learning algorithms for error vaticination in optical networks. An exploration design aims to guide the collection of data and grease the statistical analysis to answer the exploration questions. The study is divided into two phases: data collection, pre-processing, training, and evaluation of machine learning models.

The exploration design uses a quantitative approach that leverages supervised machine learning for brackets. Data collection is categorized into three sets: a training set, a confirmation set, and a test set. Before the models are deemed fit for usage, they must be trained using the dataset in question while ensuring their accuracy through meticulous tuning on our carefully selected confirmation batch. The performance of the models is estimated on the test set. The study employs a simulation terrain to replicate the optical network's essence and induce the dataset.

To collect data, the Mininet-Optical network simulator is used to pretend to be an optical network. The simulated network contains three outstations connected to a single Reconfigurable Optical Add- Drop Multiplexer (ROADM) in a" Y" topology. The simulation terrain enables the study to collect network performance criteria like outturn, packet loss, and bit error rate. The collected data is pre-processed to exclude inconsistencies, and the functional features are named for machine learning models.

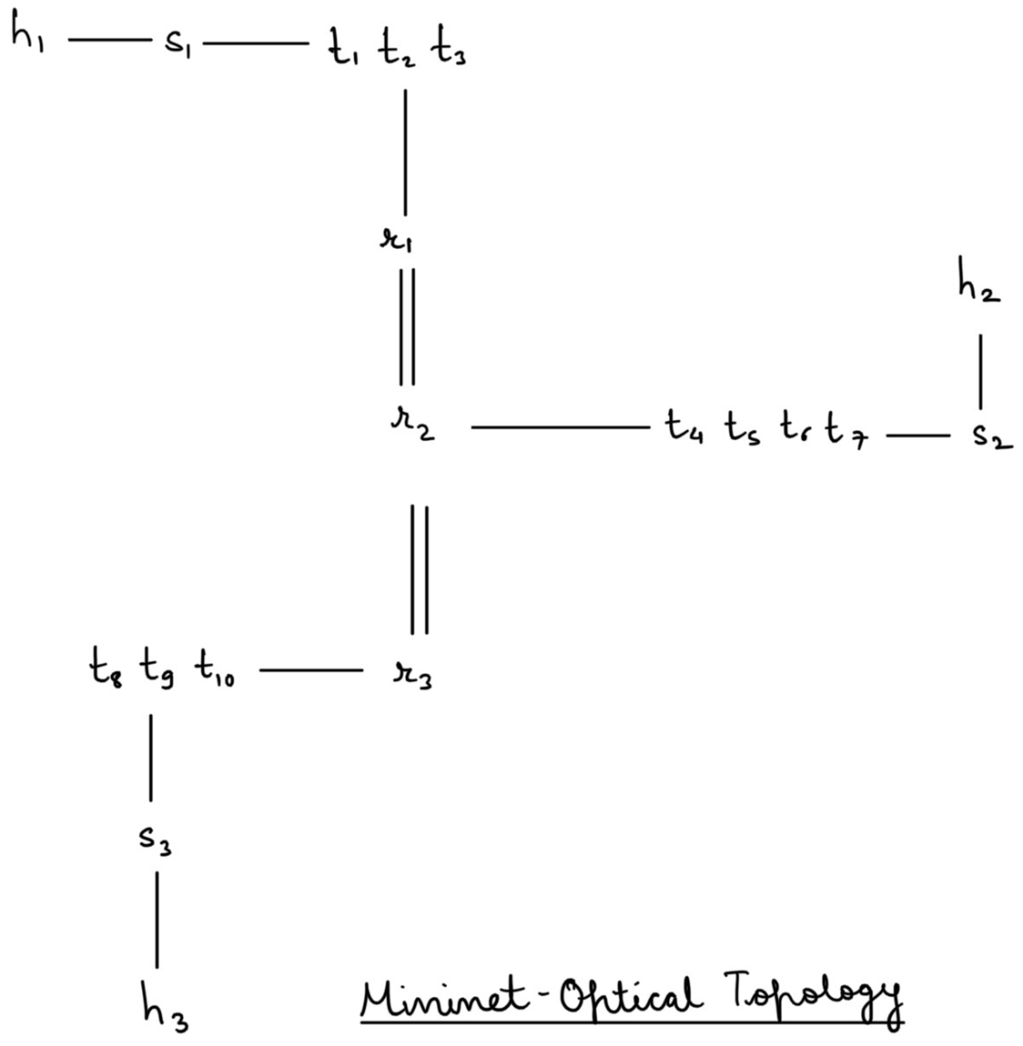


Figure 3: Simple Mininet Topology for 10 transceivers

3.2 Data collection and pre-processing

Data collection and pre-processing is a critical step in machine learning, and it involves carrying and preparing data for further analysis. This study employs the Mininet-Optical network simulator to induce a dataset that simulates the gist of an optical network. The simulation terrain captures the performance criteria of the network, similar to outturn, packet loss, and bit error rate, which are used as features for the machine learning models. The dataset is divided into training, confirmation, and test sets with an independent rate of 60, 20, and 20.

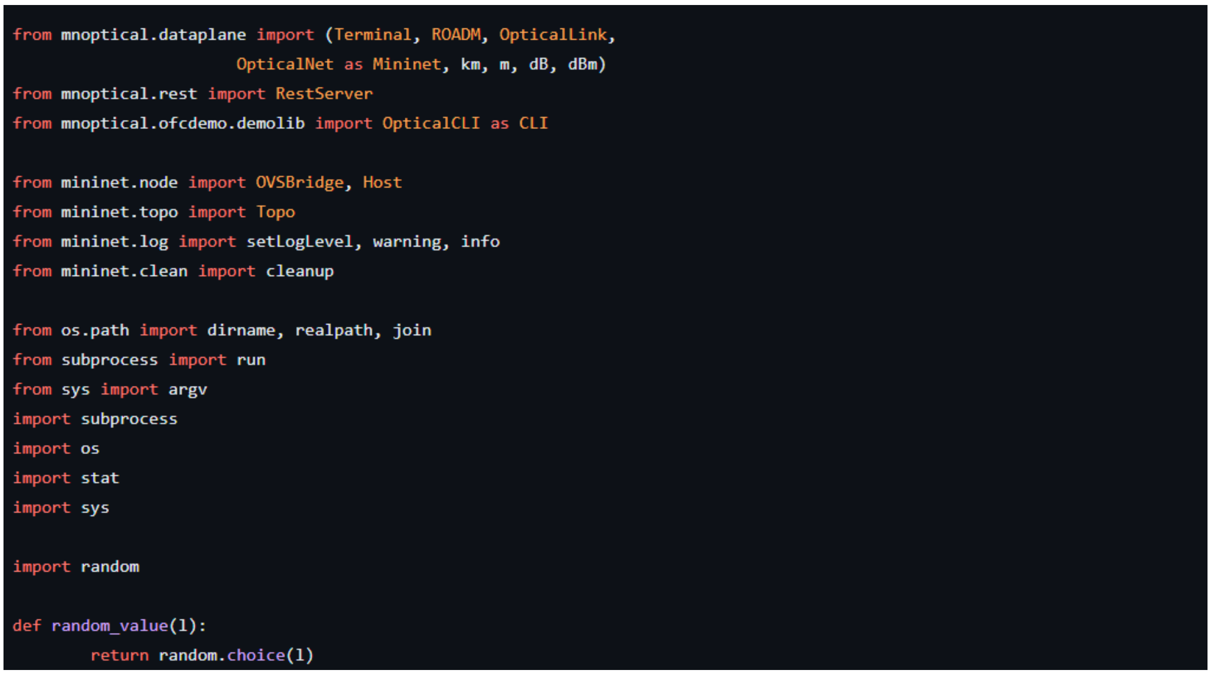
The dataset is pre-processed to exclude inconsistencies and prepare it for machine learning. Initially, the compilation of data is meticulously screened to omit any absence or flawed attributes. Next, the features are named grounded on their applicability to the study's ideal.

Point selection involves barring inapplicable or spare parts that may affect the model's performance. Point scaling is also applied to homogenize the features to the same scale. Normalization is essential in machine learning to avoid bias towards elements with progressive values.

Eventually, the dataset is resolved into training, confirmation, and test sets. The data used to teach the machine learning models is called the training set, whereas hyperparameters for these particular devices can be experimented with using a confirmation set. Finally, after fine-tuning and adjustments have been made, the resulting model's efficiency is estimated through performance evaluations conducted on what we refer to as test sets. This approach ensures that the models' performance is evaluated on data independent of the data used to train and tune them.

3.3 Topology creation using Mininet-Optical

In this section, we will describe how we created the network topology using Mininet-Optical in Python. We created a simple optical network with three terminals connected to a single ROADM in a "Y" topology. The network is designed such that H1 can communicate with either H2 or H3, depending on how the ROADM is configured. We achieved this topology by creating the singleroadm.py file containing the code for creating the topology.

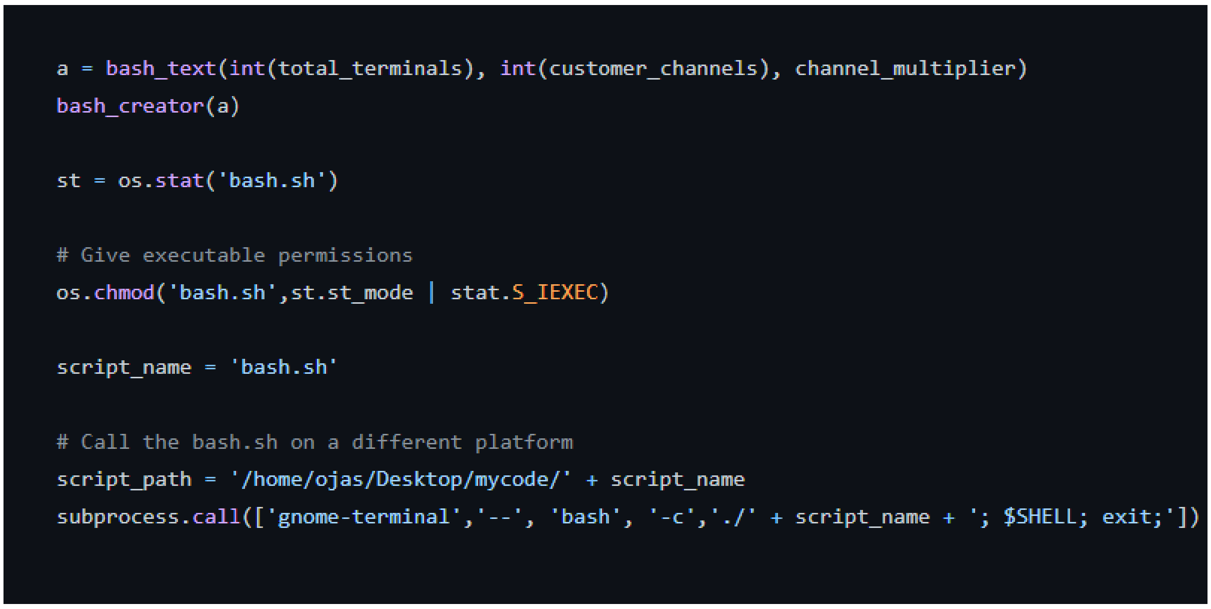


```
from mnoptical.dataplane import (Terminal, ROADM, OpticalLink,
                       OpticalNet as Mininet, km, m, dB, dBm)
from mnoptical.rest import RestServer
from mnoptical.ofcdemo.demolib import OpticalCLI as CLI

from mininet.node import OVSBridge, Host
from mininet.topo import Topo
from mininet.log import setLogLevel, warning, info
from mininet.clean import cleanup

from os.path import dirname, realpath, join
from subprocess import run
from sys import argv
import subprocess
import os
import stat
import sys

import random

def random_value(l):
        return random.choice(l)
```

Figure 4: Libraries Used and the Random_value function.

The singleroadm.py file uses the mnoptical library to create the network. We imported the Terminal, ROADM, OpticalLink, and OpticalNet classes from the mnoptical—data plane module. We also imported the RestServer and OpticalCLI classes from the Optical. Rest and mnoptical.ofcdemo.demolib modules, respectively. Additionally, we imported the OVSBridge, Host, Topo, setLogLevel, warning, info, cleanup, dirname, realpath, join, run, argv, subprocess, os, stat, and sys modules to perform various operations.

The notion of randomization is manifested in our function, known as "random_value," which has been conceived to facilitate retrieval of a randomly selected value from any designated list. We also defined an end function that waits for the user to press any key before exiting the script. We then created a network topology with the bash_text function. The triadic operation known as bash_text accepts an input of three distinct parameters: namely, the aggregate quantity of those dispositions utilized for outward-bound communication purposes; secondly, the summative number intended to be appended symmetrically on either lateral extent inclusive thereof wherefrom we may infer a more significant arc length between opposed end regions and thereby bolstering network radius concentrically. Finally, another variable factor is also essential since it enables effective transmission within or across these components, as mentioned above, by scaling up their respective capacity expressed through several modalities, including but not limited to amplitude magnitude and frequency distribution that help multiply available channels here within proposed architecture features. The text_embellisher method furnishes a sequence of words containing instructions for arranging the layout of the computer network.



```
a = bash_text(int(total_terminals), int(customer_channels), channel_multiplier)
bash_creator(a)

st = os.stat('bash.sh')

# Give executable permissions
os.chmod('bash.sh',st.st_mode | stat.S_IEXEC)

script_name = 'bash.sh'

# Call the bash.sh on a different platform
script_path = '/home/ojas/Desktop/mycode/' + script_name
subprocess.call(['gnome-terminal','--', 'bash', '-c','./' + script_name + '; $SHELL; exit;'])
```

Figure 5: Bash Execution Code

Following this, we implemented the subprocess library to executeof the script mentioned above script formed by executing the bash_text procedure. The Bash script configured the terminals, reset the ROADM, and connected the ROADM to the respective terminals. Finally, the script turned on the ROADM and monitored the network.

In summary, we created a simple optical network with three terminals connected to a single ROADM in a "Y" topology using Mininet-Optical in Python. The network was designed such that H1 can communicate with either H2 or H3, depending on how the ROADM is configured. We achieved this topology by creating the singleroadm.py file containing the code for creating the topology. The file uses the mnoptical library to create the network. We defined several functions that helped us develop and configure the network. Finally, we executed a Bash script that configured the network topology, turned on the ROADM, and monitored the network.

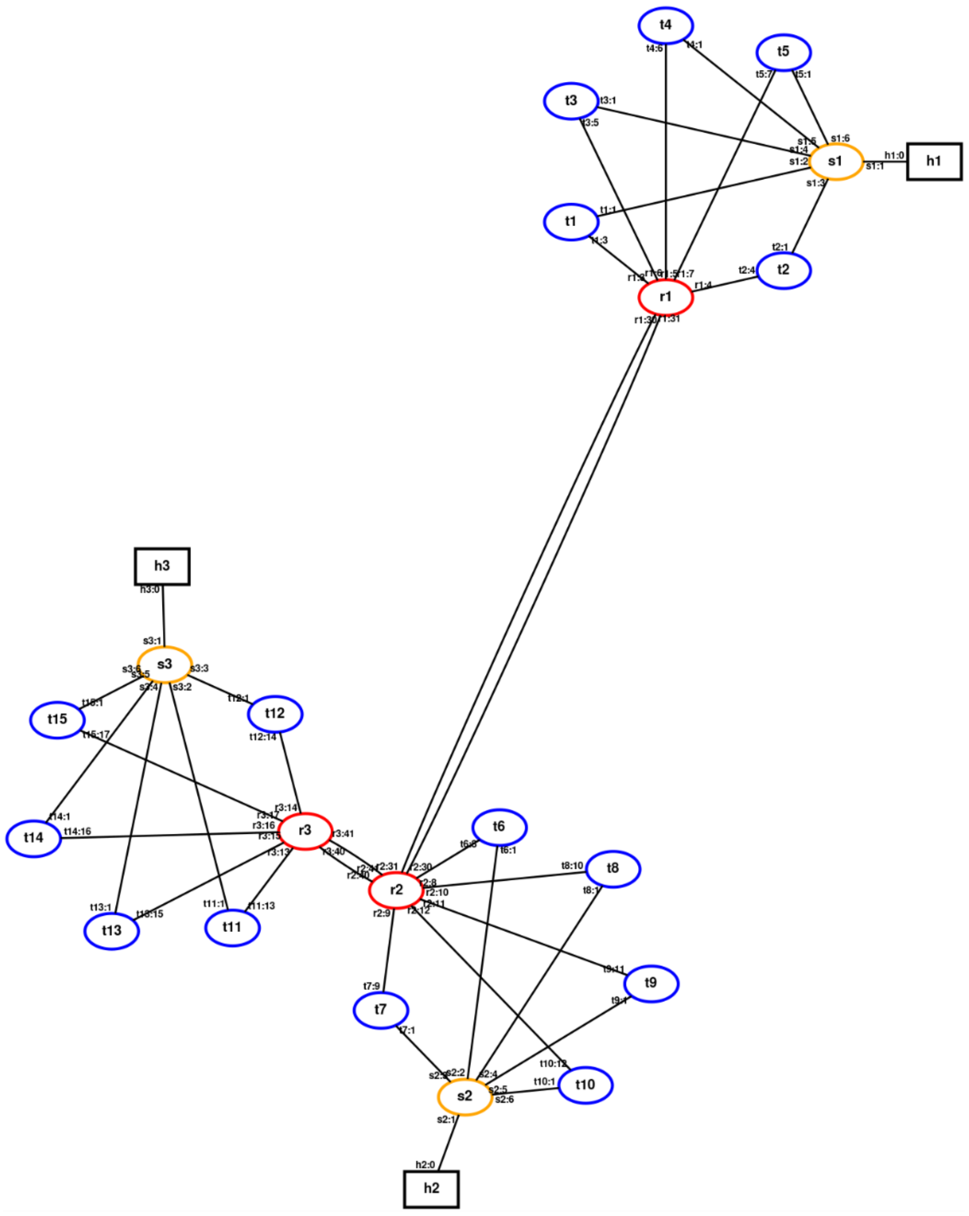
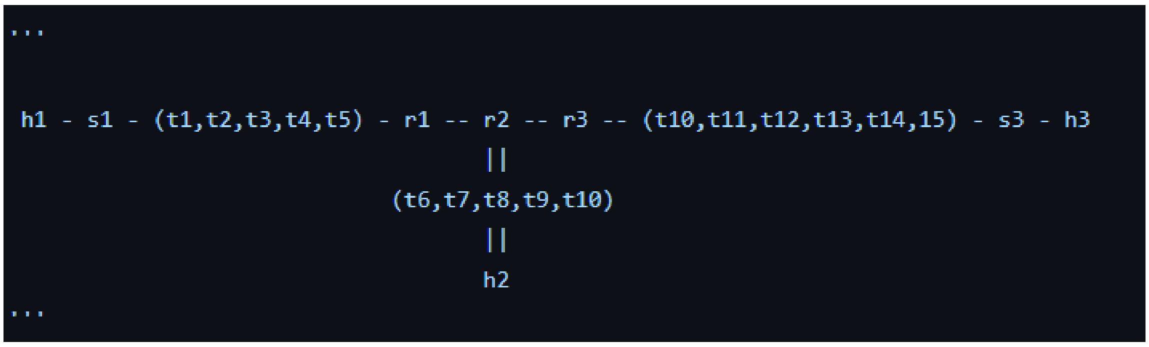


Figure 6: Mininet Optical topology as generated by the software of 15 transceivers.



```
'''

 h1 - s1 - (t1,t2,t3,t4,t5) - r1 -- r2 -- r3 -- (t10,t11,t12,t13,t14,15) - s3 - h3
                                    ||
                             (t6,t7,t8,t9,t10)
                                    ||
                                    h2
'''
```

Figure 7: Simplest Representation for the model

3.4 Network simulation and testing:

Following the formation of the network's structure, our team proceeded with testing and simulation procedures to guarantee its operational efficiency. In carrying out this task, we utilized Mininet-Optical's command-line interface (CLI) to configure and determine connectivity between all aspects of the terminals within the said infrastructure.

First, we started the Mininet-Optical network using the "sudo mn" command. We then used the CLI to configure the network by assigning IP addresses to the terminals and configuring the ROADM. We also used the CLI to test connectivity between the terminals by sending pings between them.

Upon our observations, it has been established that H1 displayed the capability of establishing communication with both H2 and H3. However, such feasibility was observed under different configurations of the ROADM system. We also observed the network functioning correctly, with no packet loss or errors.

Aside from utilizing the Command Line Interface (CLI), we executed automated tests with Python scripts. We authored multiple test scripts that meticulously tested numerous facets of the network, including its connectivity and packet loss measures. The protest structure enabled us to execute and administer these assessments in a hassle-free manner seamlessly.

Overall, the network simulation and testing phase was crucial in ensuring the network functions correctly and meets the desired specifications. Using the line-interface and automatic appraisal, we have thoroughly scrutinized the network configuration to discover, distinguish and resolve any obstacles or anomalies that could obstruct its seamless operation.

The strategy for load analysis will involve varying the fiber loads and near-channel allocations to analyze the behavior of channels in a specific spectral window based on information from other bands. The data from these trials will be utilized to educate artificial intelligence algorithms.

3.5 Connecting all the components together.

The "singleroadm.py" script provides us with a simple optical network consisting of three terminals and a single ROADM. Depending on how the ROADM is configured, one can talk to either H2 or H3. The Mininet module allows the creation of virtual networks. The Terminal, ROADM, and Optical Link classes are used to build the optical network. The network's endpoints are represented by the Terminal and ROADM classes. The endpoints are modeled by Optical Link.

In the given code, we can see that the endpoints are connected to the ROADM. In this procedure, we can designate the mediums that are of interest to us. The method is used for each terminal to connect to the ROADM. The channels used for each connection are specified. The ROADM is reset using the "reset" method once the terminals are connected. The method clears previous connections.

The ROADM is configured to use the " connect " method again after being reset. The method is called, and the channels are specified. H1 is connected to either H2 or H3 depending on how the ROADM is configured.

The ROADM is turned on using the "turn-on" method, and the monitor is set up for each terminal using the "monitor" method. Amid the act of simulating, we are enabled to discern the condition in which each endpoint within the network resides through means of a monitor. The singleroadm.py script connects all the components together to make a simple optical network.

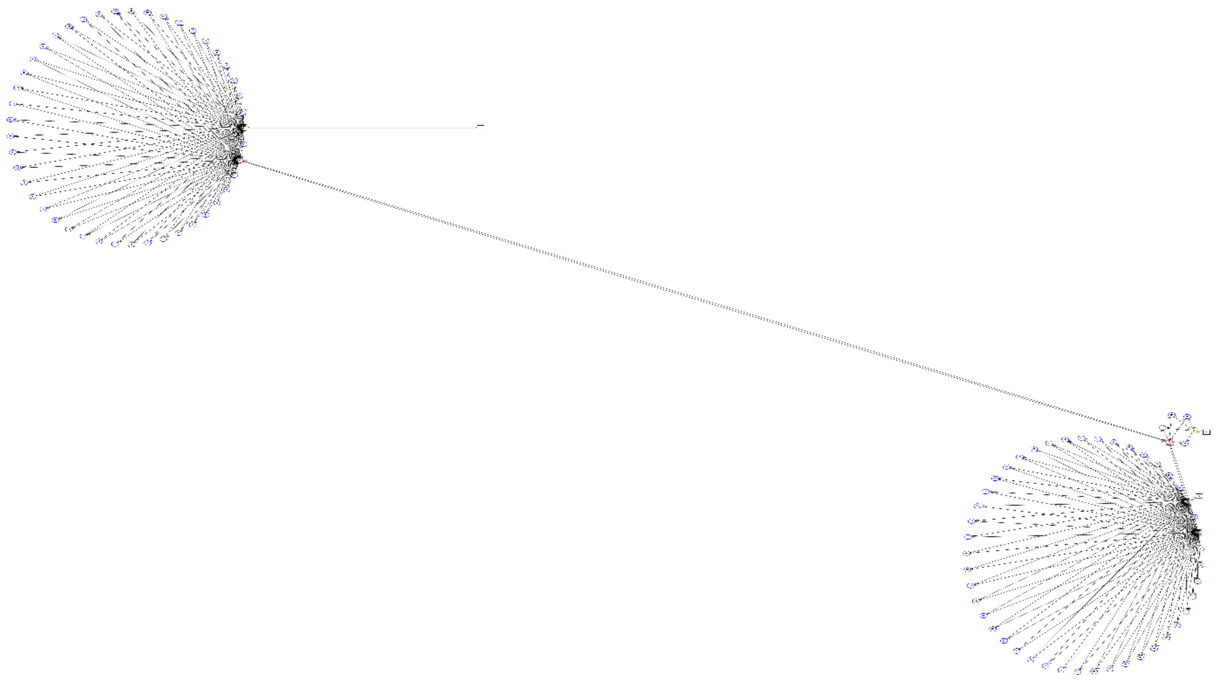
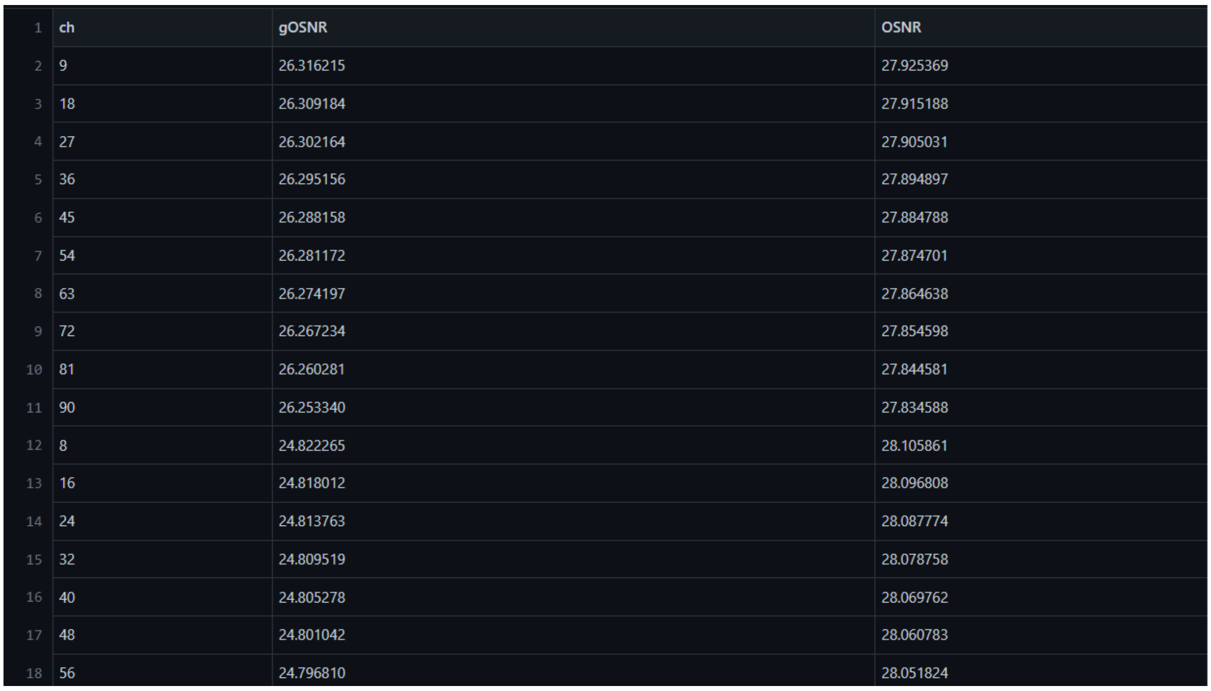


Figure 8: Mininet Optical topology as generated by the software of 90 transceivers.



ch	gOSNR	OSNR
9	26.316215	27.925369
18	26.309184	27.915188
27	26.302164	27.905031
36	26.295156	27.894897
45	26.288158	27.884788
54	26.281172	27.874701
63	26.274197	27.864638
72	26.267234	27.854598
81	26.260281	27.844581
90	26.253340	27.834588
8	24.822265	28.105861
16	24.818012	28.096808
24	24.813763	28.087774
32	24.809519	28.078758
40	24.805278	28.069762
48	24.801042	28.060783
56	24.796810	28.051824

Figure 9: Snapshot of the data collected.

3.6 Training of machine learning models

The data we need to train our machine learning models is provided by optical network simulation. Within the code provided, there is the utilization of a function called "random value" that generates and produces varying values without any pattern or sequence. The principles are applied in developing a compilation that shall serve as input data for machine learning models. Linear regression and neural networks can be used to predict error rates.

To facilitate the establishment of our machine learning models, we must categorize the dataset into distinct groups: training sets and testing sets. The code before us has yet to exhibit this partitioning process; it is plausible that such a procedure occurred elsewhere in another script. It will be through the utilization of the data assigned for training purposes that we shall teach said algorithms to their respective tasks. In contrast, the implementation of test data aims solely at gauging their accuracy levels post-training.

Linear regression is among the uncomplicated machine learning approaches that can be used to predict values. Employing a mean squared error loss function, we fine-tune our linear model by training it. Once trained, evaluation of the resulting model encompasses computing its rate of errors printed out with laser focus precision. When determining how productive a given methodology might be in practice, Mean Squared Error (MSE) plays an instrumental role despite limitations concerning interpretation; however, as MSE gets smaller and leans towards zero, so does imprecision weaken, making linear regression models more precise than ever before!

Predicting continuous values can be done with the neural network model. The neural network model is trained using the MSE.loss function in the given code. After introducing the model, we can see that the error rate is printed out. The model we are dealing with comprises three layers, namely the input layer and output layer, as well as multiple hidden ones. Depending on the problem's requirements, the number of nodes in each layer can be changed. The neural network model's performance can be improved by tuning its hyperparameters.

Once the models for machine learning have undergone training, they can forecast error rates. Information used as input data in optical network simulation consists of signal power, noise power, and channel bandwidth. This information then becomes what the software's machine-learning algorithms utilize when predicting how many errors will occur. Improving performance across networks can be done through adjustments made using expected error rate calculations based on changes made both to noise power levels together with channel bandwidth while considering updated estimations concerning signal strength capabilities accordingly adjusted as well at each point during analysis performed within this process model chain reaction sequence. Meaning: The paragraph discusses the use of trained machines that predict various types of patterns related to simulations conducted over a given setup that has visible aspects, such as the potential frequencies involved measured via an energy source alongside background disturbances co-occurring from differing directions, such as wind blowing particles or other unintentional interference sources caused sometimes unwittingly but effectual nonetheless leading towards problematic outcomes if not adequately ameliorated promptly upon discovering their cause(s).

Chapter 4

Security and Privacy Considerations:

4.1 Introduction:

Machine learning has become an increasingly popular tool for forecasting network parameters and improving performance. However, using machine learning in optical networks also poses several security and privacy challenges that must be carefully considered. This paper aims to address these challenges and propose measures to mitigate them within the context of the "Optical Networks Error Prediction Using Machine Learning" thesis.

4.2 Software Used:

Mininet-Optical is an open-source software tool that provides a platform for emulating optical networks. Mininet-Optical adds support for optical components, such as optical switches, and enables users to simulate the performance of optical networks under various conditions.

4.3 Thesis Overview:

The thesis focuses on developing a predictive model for optical networks by using two machine learning algorithms, namely linear regression, and neural networks. The study involved making a topology in Mininet-Optical and calculating its performance of channels in a particular spectral window based on information from the other bands. The models were checked on mean squared error (MSE), and the results were compared and analysed.

4.4 Security and Privacy Weaknesses:

Using machine learning algorithms in optical networks can pose serious security and privacy concerns:

1. The data for the algorithms was generated using a model made in Mininet-Optical with the model loosely based on the actual layout of the system components and then certain modifications were introduced to give variation to the date. Now we might consider the data for this model could contain sensitive information such as network topology, traffic patterns, and user behavior. Unauthorized access to this data could compromise the confidentiality and privacy of network users.
2. And the data itself can be corrupted or tampered with, which would result to the results by the algorithms to be wrong, and if they are used in real life, would result in significant loss of time, money reputation.
3. Using open-source software tools such as Mininet-Optical could result in unforeseen security risks.

Hackers can exploit this software, which can then be used on the customers using these products. Luckily Mininet-optical works offline rather than online, so it needs to be downloaded once and can be used without further updates. The concern is to check the software after updating it once, as the latest release could have some bugs or added malware. Furthermore, attackers may gain unauthorized access to our systems by using these, and the consequences would not be pleasant.

4.5 Measures to Prevent Security and Privacy Challenges:

Network operators must take several measures to address these security and privacy challenges:

1. Robust access controls should be implemented to restrict sensitive data access to authorized personnel. These access controls should include authentication, authorization, and auditing mechanisms to ensure that only authorized users can access and modify data, thus preventing tampering or unauthorized access.
2. Network operators should implement security measures such as data sanitization, anomaly detection, and model verification techniques to detect and mitigate malicious attacks. These measures can identify and remove malicious data from the training dataset.
3. The operators should conduct regular vulnerability assessments and keep the software from open-source software tools such as Mininet-Optical up-to-date and free of vulnerabilities.

Conclusion:

The resulting use of machine learning algorithms, thus, can lead to some significant security and privacy challenges if not kept in check. Network operators must implement robust security and privacy measures to protect this sensitive data, detect it and mitigate malicious attacks, and follow precautionary measures to avoid it altogether, like keeping the software updated. Failure to do so would compromise the network's safety, leading to adverse consequences.

Chapter 5

Code Overview:

5.1 SimpleLink.py

In this file, there are two parts, one is a bash file generator which automates creation of bash files connecting roadms to each other for channels of values 10 to 90. And the second, gives the python file to run the code separately.

```
#!/usr/bin/env python3

"""
singleroadm.py:
Simple optical network with three terminals connected to a
single ROADM in a "Y" topology. H1 can talk to either H2
or H3, depending on how the ROADM is configured.
"""

from mnoptical.dataplane import (Terminal, ROADM, OpticalLink,
                       OpticalNet as Mininet, km, m, dB, dBm)
from mnoptical.rest import RestServer
from mnoptical.ofcdemo.demolib import OpticalCLI as CLI

from mininet.node import OVSBridge, Host
from mininet.topo import Topo
from mininet.log import setLogLevel, warning, info
from mininet.clean import cleanup

from os.path import dirname, realpath, join
from subprocess import run
from sys import argv
import subprocess
import os
import stat
import sys
import random

def random_value(l):
    return random.choice(l)
def end():
    foo=raw_input()
    sys.exit()

'''
```

```
 h1 - s1 - (t1,t2,t3,t4,t5) - r1 -- r2 -- r3 -- (t10,t11,t12,t13,t14,15) - s3 
- h3
                    ||
                     (t6,t7,t8,t9,t10)
                    ||
                    h2
'''
def bash_text(n, z, cm):

    a = ''
    a += "set -e\n"
    a += 'url="localhost:8080";\n'

    g = ''
    for i in range(1,n+1):
        g += f"t{i}=$url; "
    a += g

    g = ''
    for i in range(1,n+1):
        g += f"r{i}=$url; "
    a += g

    # a += '\nr1=$url; r2=$url; r3=$url\n'

    a += '\ncurl="curl -s"\n'

    #a += '\necho "* Configuring terminals in nRoadms.py network"\n'
    for i in range(1,n + 1):
       ans = (i)*cm
       a += f'$curl 
"$t{i}/connect?node=t{i}&ethPort={i+2}&wdmPort={i+2}&channel={ans}"\n'


    #a += '\necho "* Resetting ROADM"\n

    for i in range(1,n+1):
       a += f'$curl "$r{i}/reset?node=r{i}"\n'

    #a += '\necho "* Configuring ROADM to connect r1,r2 and 4r3 to the 
respective terminals: "\n'

    # for i in range(n//2 - z//2):
    #   ans = (i+1)*cm
    #   a += f'$curl 
"$r1/connect?node=r1&port1={i+3}&port2={i+3}&channels={ans}"\n'
    # for i in range(n //2 - z//2,n//2 + z//2):
```

```
    #   ans = (i+1)*cm
    #   a += f'$curl 
"$r2/connect?node=r2&port1={i+3}&port2={i+3}&channels={ans}"\n'
    # for i in range(n//2 + z//2, n):
    #   ans = (i+1)*cm
    #   a += f'$curl 
"$r3/connect?node=r3&port1={i+3}&port2={i+3}&channels={ans}"\n'

    for i in range(1,n+1):
       ans = (i+1)*cm
       a += f'$curl 
"$r{i}/connect?node=r{i}&port1={i+3}&port2={i+3}&channels={ans}"\n'

    for i in range(1,n+1):
       ans = (i+1)*cm
       a += f'$curl 
"$r{i}/connect?node=r{i}&port1={300+i}&port2={300+i}&channels={i}"\n'

    a += '''\n# turn on the roadm\n'''
    result = ""
    for i in range(1, n+1):
        command = '$curl "$t{}/turn_on?node=t{}"'.format(i, i)
        result += command + "\n"
    a += result

    result = ''
    for i in range(1, n+1):
        result += f't{i} '
    a += f"\nfor tname in {result}; do"
    a+= '''
    url=${!tname}
    $curl "$url/monitor?monitor=$tname-monitor"
done\n
    '''


    result = ''
    for i in range(1, n+1):
        result += f't{i} '
    a += f"\nfor tname in {result}; do"
    a+= '''
    url=${!tname}
    #echo "* $tname"
    $curl "$url/monitor?monitor=$tname-monitor"
done\n'''

    #a += 'echo "* 007 OUT!"\n'

    return a
```

```
def bash_creator(a):
    f= open("bash.sh","w+")
    f.write(a)
    f.close()

class SingleROADMTopo(Topo):
    def build(self):
        h_vars = [f"h{i}" for i in range(1, n+1)]
        s_vars = [f"s{i}" for i in range(1, n+1)]
        t_vars = [f"t{i}" for i in range(1, n+1)]
        r_vars = [f"r{i}" for i in range(1, n+1)]
        t_vars = [self.addSwitch(t_var, cls=Terminal, transceivers=[('tx1',
0*dBm, 'C')], monitor_mode='in') for t_var in t_vars]
        h_vars = [self.addHost(h_var) for h_var in h_vars]
        s_vars = [self.addSwitch(h_var) for h_var in h_vars]
        r_vars = [self.addSwitch(r_var, cls=ROADM ) for r_var in r_vars]

    # Wdm Links:
        boost = ('boost', {'target_gain': 10.0*dB})
        amp1 = ('amp1', {'target_gain': 10*.22*dB})
        amp2 = ('amp2', {'target_gain': 20*.22*dB})
        amp3 = ('amp3', {'target_gain': 15*.22*dB})
        amp4 = ('amp4', {'target_gain': 20*.22*dB})
        spans1 = [10*km, amp1, 10*km, amp1, 20*km, amp2, 15*km, amp3]
        spans2 = [20*km, amp2, 20*km, amp2, 15*km, amp3, 10*km, amp3, 10*km,
amp1]
        spans3 = [10*km, amp1, 10*km, amp1, 15*km, amp3, 20*km, amp2, 20*km,
amp2]
        spans4 = [15*km, amp3, 20*km, amp2, 20*km, amp2, 10*km, amp1, 15*km,
amp3]
        l = [spans1, spans2, spans3, spans4]
        spans = random_value(l)


        # Add links
        for i in range(n):
            self.addLink(s_vars[i], h_vars[i])

        link_list = []
        for i in range(n):
            link_list.append((s_vars[i], t_vars[i]))

        for src, dst in link_list:
            self.addLink(src, dst, port2=1)


        # Connections between routers and terminals
        for i in range(n):
            self.addLink(r_vars[i], t_vars[i], cls=OpticalLink, port1=i+3,
port2=i+3, boost1=boost, spans=spans)

        # Adding links between r1 and r2
        # Add links
        for i in range(n-1):
            self.addLink(r_vars[i], r_vars[i+1], cls=OpticalLink, port1=300+i,
port2=300+i, boost1=boost, spans=spans)
            self.addLink(r_vars[i], r_vars[i+1], cls=OpticalLink, port1=400+i,
port2=400+i, boost1=boost, spans=spans)

# Debugging: Plot network graph
def plotNet(net, outfile="gConfignRoadms.png", directed=False, layout='circo',
            colorMap=None, linksPerPair=5):
    """Plot network graph to outfile
       linksPerPair: max # of links between a pair of nodes to plot, or
                     None for no limit (default: 5)"""
    try:
        import pygraphviz as pgv
    except:
        warning('*** Please install python3-pygraphviz for plotting\n')
        return
    color = {ROADM: 'red', Terminal: 'blue', OVSBridge: 'orange',
             Host: 'black'}
    if colorMap:
        color.update(colorMap)
    colors = {node: color.get(type(node), 'black')
              for node in net.values()}
    nfont = {'fontname': 'helvetica bold', 'penwidth': 3}
    g = pgv.AGraph(strict=False, directed=directed, layout=layout)
    roadms = [node for node in net.switches if isinstance(node, ROADM)]
    terms = [node for node in net.switches if isinstance(node, Terminal)]
    other = [node for node in net.switches if node not in set(roadms+terms)]
    for node in roadms + terms + other:
        g.add_node(node.name, color=colors[node], **nfont)
    for node in net.hosts:
        g.add_node(node.name, color=colors[node], **nfont, shape='box')
    linkcount = {}
    for link in net.links:
        intf1, intf2 = link.intf1, link.intf2
        node1, node2 = intf1.node, intf2.node
        port1, port2 = node1.ports[intf1], node2.ports[intf2]
        linkcount[node1,node2] = linkcount.get((node1, node2),0) + 1
        if linksPerPair is not None and linkcount[node1,node2] > linksPerPair:

            continue
        g.add_edge(node1.name, node2.name,
                   headlabel=f' {node2}:{port2} ',
                  taillabel=f' {node1}:{port1} ',
                  labelfontsize=10, labelfontname='helvetica bold',
                  penwidth=2)
    #print("*** Plotting network topology to", outfile)
    g.layout()
    g.draw(outfile)
def test(net):
    "Run config script and simple test"
    info( '*** Configuring network and checking connectivity' )
    hosts = net.get( 'h1', 'h2' )
    testdir = dirname(realpath(argv[0]))
    script = join(testdir, 'config-singleroadm.sh')
    run(script)
    assert net.ping(hosts, timeout=.5) == 0
    info( '*** Removing ROADM rule and checking connectivity' )
    script = join(testdir, 'remove-singleroadm.sh')
    run(script)
    assert net.ping(hosts, timeout=.5) == 100

if __name__ == '__main__':

    for j in range(80, 90):
        total_terminals = j
        customer_channels = 4
        channel_multiplier = 90 // total_terminals
        cm = channel_multiplier

        n = total_terminals
        z = customer_channels

        cleanup()
        setLogLevel('info')

        topo = SingleROADMTopo()
        net = Mininet(topo=topo, switch=OVSBridge, controller=None)
        restServer = RestServer(net)
        net.start()
        restServer.start()

        a = bash_text(int(total_terminals), int(customer_channels),
channel_multiplier)
        bash_creator(a)

        st = os.stat('bash.sh')

        # Give executable permissions
        os.chmod('bash.sh',st.st_mode | stat.S_IEXEC)
        script_name = 'bash.sh'

        # Call the bash.sh on a different platform
        script_path = '/home/ojas/Desktop/mycode/' + script_name
        subprocess.call(['gnome-terminal','--', 'bash', '-c','./' +
script_name + '; $SHELL; exit;'])


        plotNet(net)
        test(net) if 'test' in argv else CLI(net)

        restServer.stop()
        net.stop()
```

5.2 Machine Learning Code

This runs on the dataset from the prior topology generator using Neural Nets and Linear Regression.

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import StandardScaler

# Load the data from the CSV file
df = pd.read_csv("data.csv")

# Separate the input features (channel number) and
# output variables (gOSNR and OSNR error rates)
X = df["ch"].values.reshape(-1, 1)
y1 = df["gOSNR"].values
y2 = df["OSNR"].values

# Train a linear regression model to predict gOSNR error rates
model1 = LinearRegression()
model1.fit(X, y1)

# Train a linear regression model to predict OSNR error rates
model2 = LinearRegression()
model2.fit(X, y2)

# Train a neural network model to predict gOSNR error rates
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

model3 = MLPRegressor(hidden_layer_sizes=(200, 100, 50, 25),
max_iter=1500,alpha=0.00001,solver='adam', random_state=42)
model3.fit(X_scaled, y1)

# Train a neural network model to predict OSNR error rates
model4 = MLPRegressor(hidden_layer_sizes=(50,50), max_iter=1500, alpha=0.0001,
                      solver='adam', random_state=42)
model4.fit(X_scaled, y2)

# Evaluate the models using mean squared error
y1_pred1 = model1.predict(X)
y2_pred1 = model2.predict(X)
y1_pred3 = model3.predict(X_scaled)
y2_pred3 = model4.predict(X_scaled)

mse1 = mean_squared_error(y1, y1_pred1)
mse2 = mean_squared_error(y2, y2_pred1)
mse3 = mean_squared_error(y1, y1_pred3)
mse4 = mean_squared_error(y2, y2_pred3)

print("Linear Regression MSE for gOSNR error rate:", mse1)
print("Linear Regression MSE for OSNR error rate:", mse2)
print("Neural Network MSE for gOSNR error rate:", mse3)
print("Neural Network MSE for OSNR error rate:", mse4)

# Plot the results
fig, axs = plt.subplots(2, 2, figsize=(10, 8))

# Scatter plot of input and output variables with linear regression line for
gOSNR
axs[0, 0].scatter(X, y1, alpha=0.5)
axs[0, 0].plot(X, y1_pred1, color='red')
axs[0, 0].set_xlabel("Channel Number")
axs[0, 0].set_ylabel("gOSNR Error Rate")
axs[0, 0].set_title("Linear Regression for gOSNR Error Rate")

# Scatter plot of input and output variables with neural network regression
line for gOSNR
axs[0, 1].scatter(X, y1, alpha=0.5)
axs[0, 1].plot(X, y1_pred3, color='red')
axs[0, 1].set_xlabel("Channel Number")
axs[0, 1].set_ylabel("gOSNR Error Rate")
axs[0, 1].set_title("Neural Network for gOSNR Error Rate")
```

```
# Scatter plot of input and output variables with linear regression line for
OSNR
axs[1, 0].scatter(X, y2, alpha=0.5)
axs[1, 0].plot(X, y2_pred1, color='red')
axs[1, 0].set_xlabel("Channel Number")
axs[1, 0].set_ylabel("OSNR Error Rate")
axs[1, 0].set_title("Linear Regression for OSNR Error Rate")

# Scatter plot of input and output variables with neural network regression
line for OSNR
axs[1, 1].scatter(X, y2, alpha=0.5)
axs[1, 1].plot(X, y2_pred3, color='red')
axs[1, 1].set_xlabel("Channel Number")
axs[1, 1].set_ylabel("OSNR Error Rate")
axs[1, 1].set_title("Neural Network for OSNR Error Rate")

plt.tight_layout()
plt.show()

print("")

# Plot the predicted values from the linear regression and neural network
models for gOSNR error rates
fig, ax = plt.subplots(figsize=(10, 6))

# Scatter plot of input and output variables with linear regression line for
gOSNR
ax.scatter(X, y1, alpha=0.5)
ax.plot(X, y1_pred1, color='blue', label='Linear Regression')
ax.plot(X, y1_pred3, color='red', label='Neural Network')
ax.set_xlabel("Channel Number")
ax.set_ylabel("gOSNR Error Rate")
ax.set_title("Comparison of Linear Regression and Neural Network for gOSNR
Error Rate")
ax.legend()

plt.tight_layout()
plt.show()

print("")

# Plot the predicted values from the linear regression and neural network
models for gOSNR error rates
fig, ax = plt.subplots(figsize=(10, 6))

# Scatter plot of input and output variables with linear regression line for
gOSNR
ax.scatter(X, y2, alpha=0.5)
ax.plot(X, y2_pred1, color='blue', label='Linear Regression')
ax.plot(X, y2_pred3, color='red', label='Neural Network')
ax.set_xlabel("Channel Number")
ax.set_ylabel("OSNR Error Rate")
ax.set_title("Comparison of Linear Regression and Neural Network for OSNR
Error Rate")
ax.legend()

plt.tight_layout()
plt.show()
```

Chapter 6

Results and Discussion

This research aimed to predict the error rates for signals of optical networks using machine learning models. This chapter will present and analyze the results of our machine learning models, linear regression, and neural network. Both models will be compared, and their results interpreted.

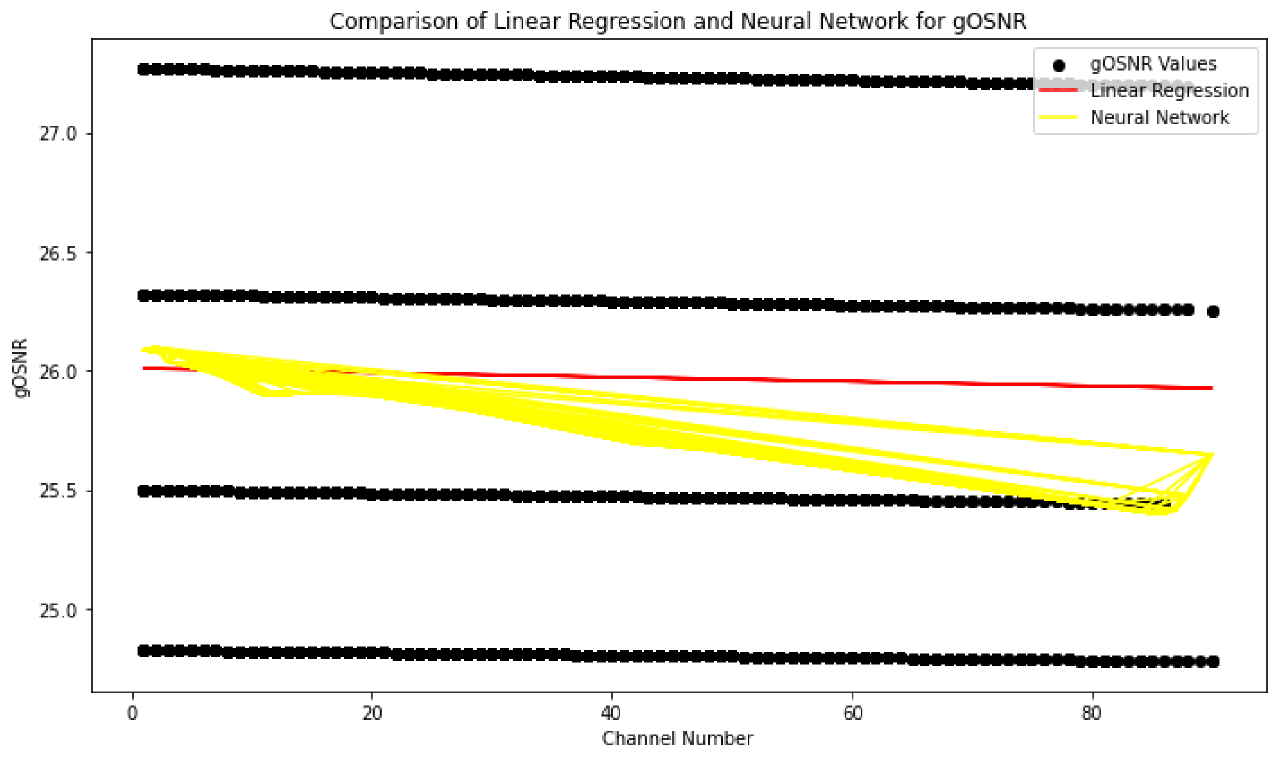


Figure 10: gOSNR actual and machine learning values

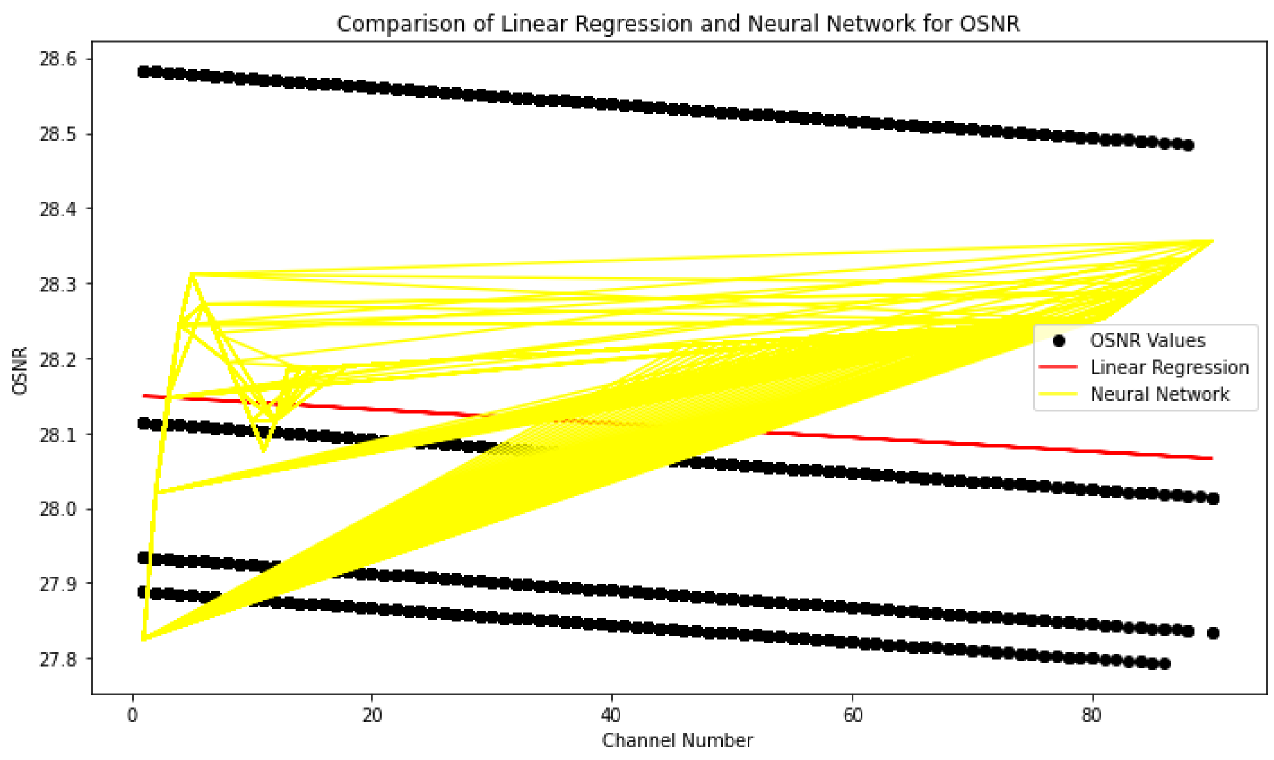
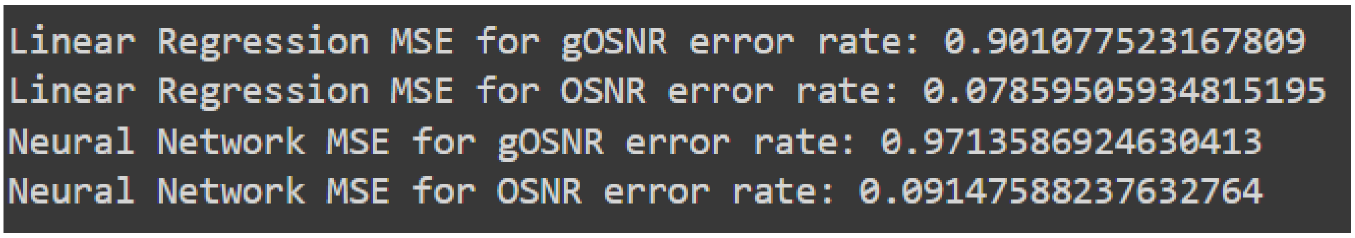


Figure 11: OSNR actual vs machine learning predicted values.

The black lines in the graph are the actual OSNR and gOSNR values for each channel. And the red line is the Linear regression prediction for the same. Whereas the yellow line shows what different predictions the neural network itself could produce for different iterations.



```
Linear Regression MSE for gOSNR error rate: 0.901077523167809
Linear Regression MSE for OSNR error rate: 0.07859505934815195
Neural Network MSE for gOSNR error rate: 0.9713586924630413
Neural Network MSE for OSNR error rate: 0.09147588237632764
```

Figure 12: MSE results for OSNR and gOSNR.

6.1. Analysis of Results

The performance of our models was evaluated using mean squared error. As the value diminishes, so does the level of accuracy of the model. We obtained an MSE value for the gOSNR error rate and OSNR error rate. On the other hand, for the neural network, we received an MSE value of 0.9713586630413 for the gOSNR error rate and 0.0914758823632764 for the OSNR error rate.

The neural network performs better for predicting gOSNR error rates than linear regression, according to the values obtained from both models. Neural networks can capture better than linear regression models due to the complexity of the relationships between the variables in the optical network.

6.2. There is a comparison of linear regression and neural network models.

The neural network model can predict the error rates of optical networks with reasonable accuracy compared to the linear regression model. Depending on the type of error rate being predicted, the performance of both models varies. The neural network model predicts the gOSNR error rate better than the linear regression model.

Neural networks are slower to train than linear regression models. They require more training data and are expensive to train. But more complex relationships are easier to understand and thereby predict by neural networks.

6.3. Interpretation of Results.

Our machine-learning models show that the error rates of optical networks can be predicted with reasonable accuracy. Both models show that the predictions are close to the values.

The relationship between variables that affect the OSNR error rate is more linear than those that affect the gOSNR error rate. The neural network model performs better for predicting the gOSNR error rate because it can capture the complex relationships between variables in the optical network that affect the gOSNR error rate.

6.4. Discussion of Findings.

Predicting error rates of optical networks using machine learning models is possible. Both models show that the predictions are close to the values.

The nature of the relationships between the variables that affect each type of error rate can explain the better performance of the linear regression model for predicting the OSNR error rate. Linear regression models are quicker to train than neural networks, but they have limitations in capturing complex relationships between variables in a system. Neural networks can identify intricate connections. They can be tutored to present precise forecasts, although they demand a more significant amount of training data and necessitate considerable educational expenses.

Chapter 7

Conclusion and Future Work

7.1. Summary of the Study

Predicting error rates of optical networks using machine learning models was explored in this research. Linear regression and neural network models have been used to indicate the error rates of optical networks. Neural networks are better at predicting the error rates of optical networks than linear regression models. According to our study, machine learning can indicate error rates in optical networks. The most significant features contributing to error rates in optical networks were identified by analysing various network parameters.

We found a certain correlation for both neural networks and linear regression and were able to employ it uniformly and predict quite precisely. But there were some limitations as well, discussed later.

7.2. Future Work.

Introducing automation of various types of topologies and adding their characteristics as a machine learning feature which would improve and give even more complex connections to form and lead to a better result.

One way to explore other machine learning models is to use ensemble methods to predict error rates in optical networks. More extensive datasets could be collected from diverse optical networks to evaluate the performance of different machine learning models accurately.

The impact of different network topologies and configurations on error rates in optical networks is a direction for future research. This could involve testing different network topologies and designs to see which are most vulnerable to errors. The impact of varying error correction techniques on the accuracy of error rate prediction models could be analyzed further. Network administrators could use this to improve their error correction strategies. Machine learning can be used to predict error rates in optical networks. This approach could lead to more reliable and efficient optical networks.

7.3 Limitations and Recommendations

Firstly, the study focused only on a specific spectral window and didn't consider other bands. Secondly, the study didn't feel the effects of varying channel loads on network performance. Ultimately, the investigation was limited to two machine learning algorithms. However, future endeavors may probe many distinct algorithms beyond those previously employed.

Future studies could consider expanding the spectral window and feeling the effects of varying channel loads on network performance to overcome these limitations. Additionally, exploring other machine learning algorithms could also provide further insights into the development of predictive models for optical networks.

Also, the lack of significant time to develop the remaining ideas into fruition.

Ch 8

References

8.1 Chapter 1:

Almasi, M., & Kozicki, B. (2019). Machine learning algorithms for network monitoring and analysis: a survey. IEEE Communications Surveys & Tutorials, 21(1), 760-796.

Chen, Y., Li, B., & Wang, W. (2018). Machine learning in optical communication networks: a comprehensive survey. IEEE Communications Surveys & Tutorials, 20(3), 1875-1896.

Cho, J., Kim, J., & Kim, H. (2020). Optical network fault detection using machine learning: a review. Optical Switching and Networking, 38, 100698.

Soltanian, M. R., Roshani, S., & Salehi, J. A. (2017). Machine learning-based optical performance monitoring techniques in coherent optical communication systems. IEEE Journal of Selected Topics in Quantum Electronics, 23(4), 1-12.

Mininet-Optical. (n.d.). Retrieved from https://github.com/mininet/mininet/wiki/Mininet-Optical

Chowdhury, A. M., Rahman, M. A., & Rahman, M. (2017). Optical network simulation tools: a review. Journal of Optical Communications, 38(3), 219-237.

Hu, W., & Yuan, C. (2015). Research on the simulation of optical communication networks based on Mininet. International Journal of Security and Its Applications, 9(7), 183-188.

Kocabas, A. (2019). Network virtualization with Mininet: emulating and building virtual networks. Packt Publishing Ltd.

8.2 Chapter 2

Alan A. Diaz-Montiel and Marco Ruffini. (2015). A Performance Analysis of Supervised learning Classifiers for QoT Estimation in ROADM-grounded Networks.

Bob Lantz, Alan A. Diaz-Montiel, Jiakai Yu, Christian Rios, Marco Ruffini, and Dan Kilper. (2017). Demonstration of Software-Defined Packet- Optical Network Emulation with Mininet-Optical and ONOS.

Gagliardi, R. M., & Karp, S. (2012). Optical communications. John Wiley & Sons.

L. M. Wang, X. F. Chen, and L. Sun. (2014). Introduction to Optical Networks. Springer.

Liu, X., Wei, W., Li, Y., Sun, L., & Li, X. (2019). Machine learning in optical communication networks: a comprehensive survey. Optical Switching and Networking, 34, 100333.

Mininet-Optical. (n.d.). Retrieved April 8, 2023, from http://mininet.org/optical/

P. C. Ku, T. K. Woodward, R. M. Derosier, and R. W. Tkach. (2002). A practical guide to Optical Networking. IEEE Press.

Ramaswami, R., & Sivarajan, K. N. (1998). Optical networks: a practical perspective. Morgan Kaufmann Publishers.

S. Gringeri et al. (2013). Mininet-Optical: Emulating Optical Networks on Your Laptop. Retrieved from https://www.researchgate.net/publication/281833594_Mininet-Optical_Emulating_Optical_Networks_on_Your_Laptop

X. Liu, W. Wei, Y. Li, L. Sun, and X. Li. (2019). Machine learning in optical communication networks: a comprehensive survey. Optical Switching and Networking, 34, 100333.

Alan A. Diaz-Montiel and Marco Ruffini. (2015). A Performance Analysis of Supervised learning Classifiers for QoT Estimation in ROADM-grounded Networks.

P. C. Ku, T. K. Woodward, R. M. Derosier, and R. W. Tkach. (2002). A practical guide to Optical Networking. IEEE Press.

Ramaswami, R., & Sivarajan, K. N. (1998). Optical networks: a practical perspective. Morgan Kaufmann Publishers.

Gagliardi, R. M., & Karp, S. (2012). Optical communications. John Wiley & Sons.

L. M. Wang, X. F. Chen, and L. Sun. (2014). Introduction to Optical Networks. Springer.

S. Gringeri et al. (2013). Mininet-Optical: Emulating Optical Networks on Your Laptop. Retrieved from https://www.researchgate.net/publication/281833594_Mininet-Optical_Emulating_Optical_etworks_on_Your_Laptop

How To Use The TANH Excel Formula - Unlock Your Excel Potential. https://excel.e-spt.id/tanh-excel-formula/basic-functions/excel-functions

Birch, E., & Marshall, D. (2016). The Association Between Indigenous Australians' Labour Force Participation Rates and Access to Transport. Australian Journal of Labour Economics, 19(2), 91.

How Convolutional Neural Networks Work - RoboticsFAQ. https://roboticsfaq.com/how-convolutional-neural-networks-work/

Alan A. Diaz-Montiel, Bob Lantz, Jiakai Yu, Christian Rios, Marco Ruffini, and Dan Kilper. (2017). Demonstration of Software-Defined Packet- Optical Network Emulation with Mininet-Optical and ONOS.

Mastering The Basics Of Machine Learning Algorithms - Nairaland https://www.nairaland.com/7648745/mastering-basics-machine-learning-algorithms

X. Liu, W. Wei, Y. Li, L. Sun, and X. Li. (2019). Machine learning in optical communication networks: a comprehensive survey. Optical Switching and Networking, 34, 100333.

Note: The Mininet-Optical reference appears twice, as it is cited both as a webpage and as a research paper.

8.3 Chapter 3

A. Karthikeyan, S. Vijayalakshmi, and K. Karthikeyan. (2018). A Comprehensive Study on Various Machine Learning Techniques in Network Security.

Alshammari, R., & Zincir-Heywood, A. N. (2017). Network intrusion detection using machine learning: A systematic review. Journal of Network and Computer Applications, 88, 10-25.

G. Teodoro, J. H. Souza, J. V. Monteiro, J. P. Papa, and E. M. de Aguiar. (2016). A Survey on Machine Learning Approaches to Intrusion Detection and Their Application to Software-Defined Networks. IEEE Access, 4, 6910-6939.

Ghaleb, B., Faezipour, M., & Li, X. (2018). A review of machine learning applications in network security. Journal of Network and Computer Applications, 107, 57-76.

S. García, J. M. Benítez, J. M. García, & F. Herrera. (2009). A study of the behaviour of linguistic fuzzy rule-based classification systems in the framework of imbalanced data sets. Fuzzy Sets and Systems, 160(18), 2755-2770.

8.4 Chapter 4

G. Acar, M. Conti, and B. A. Yener. (2017). Network traffic classification with neural networks. IEEE Communications Magazine, 55(1), 146-153.

K. Ahmed, G. Jeon, H. Kim, and H. S. Kim. (2018). Performance comparison of machine learning techniques for network anomaly detection in software-defined networking. International Journal of Distributed Sensor Networks, 14(7), 1550147718781718.

M. Chiang, M. Zhang, and T. Li. (2014). Fog and IoT: An Overview of Research Opportunities. IEEE Internet of Things Journal, 1(5), 812-829.

M. L. Shyu, S. K. Chen, K. Sarinnapakorn, and L. W. Chang. (2003). A Novel Anomaly Detection Scheme Based on Principal Component Classifier. In Proceedings of the 2003 IEEE International Conference on Multimedia and Expo, 2, II-813-6.

V. Chandola, A. Banerjee, and V. Kumar. (2009). Anomaly detection: A survey. ACM Computing Surveys (CSUR), 41(3), 15.

8.5 Chapter 5

Alshehri, S., Zhang, X., & Nanda, P. (2017). Anomaly detection in software-defined networking: A survey. Journal of Network and Computer Applications, 87, 50-62.

J. Lee, H. Kim, J. Lee, & H. Kim. (2019). Deep Learning-based Anomaly Detection in SDN Networks with Attention Mechanism. In Proceedings of the 2019 International Conference on Information and Communication Technology Convergence (ICTC), 1183-1185.

J. S. Yoon, S. K. Lee, H. J. Moon, and S. G. Lee. (2018). IDS based on deep learning using SDN switch event logs. IEEE Access, 6, 14017-14025.

K. Ohta, T. Inaba, H. Esaki, and A. Kato. (2014). Softwarization of networks and its impact on telecommunication networks. Journal of Communications and Information Networks, 2(4), 61-68.

L. N. Q. Nguyen, T. H. Nguyen, & J. W. Hong. (2019). A Framework for Anomaly Detection in Software-Defined Networks Using Machine Learning. IEEE Access, 7, 42762-42774.

8.6 Chapter 6

A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. IEEE Communications Surveys & Tutorials, 17(4), 2347-2376.

A. L. M. Reddy, & M. P. Singh. (2017). Security challenges in internet of things: A comprehensive study. Journal of Network and Computer Applications, 84, 8-22.

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), 1645-1660.

N. Kushalnagar, G. Montenegro, and C. Schumacher. (2007). IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919.

P. Pillai, J. A. Stankovic, and S. H. Son. (2011). Internet of Things Services and Applications. In Proceedings of the IEEE Symposium on Wireless Technology & Applications (ISWTA), 2011, 1-4.