POLITECNICO DI TORINO

Quality of Transmission Estimation For An Open Line System

USING MACHINE LEARNING

Author:
Bilal Muhammad

Supervisor:
Vittorio Curri

A Master's thesis is submitted in fulfillment of the
requirements for the degree of CCNE
in the

OPTCOM GROUP - DET

March 30, 2019

Abstract

The adoption of Machine Learning techniques in optical communication is motivated by extraordinary growth of network traffic. Various machine learning models are used to predict the quality of transmission (QoT-E) of an unestablished light path and their performance is evaluated. Estimation of (QoT) is vital for diminishing provisioned margins and for optimizing design of optical network. It could be during design phase of network or in an already deployed network.

In this thesis, 11-span OLS is established to simulate 35 channels at 100 GHz with largest spectral hole burning effects. Machine Learning based simulation using LR and DNN regression by using TensorFlow® library as well as interpolated simulation is performed on synthetic data taken from line emulator. It is also applied on dataset collected from real line system. It is demonstrated that, mean and standard deviation are very high when OSNR is predicted by using interpolation as compare to ML-based OSNR estimation. ML regression is investigated that predicts, whether the OSNR of unestablished light paths fulfills system requirements or not. It is trained and tested on synthetic data and as well as on real data. Its performance is assessed by using different algorithms and also by using different combinations of features.

Dedication

To my parents,
Without whom non of my success would be possible.

Acknowledgements

I would like to thank Prof. Vittorio Curri, my supervisor for his guidance and thoughtful advices.

Contents

List of Figures

5.1 LaTeX Error: There's no line here to endSee the LaTeX manual or LaTeX Companion for explanation.Your command was ignored.Type I ¡command¿ ¡return¿ to replace it with another command,or ¡return¿ to continue without it.
81

List of Tables

Chapter 1

Introduction, Motivations and Goals

1.1 Introduction

Artificial intelligence will shape the future more effectively than any other innovation. Machine learning (ML) is a branch of artificial intelligence that precipitate the idea that, by giving access to right data and by using right algorithm, machine can learn by themselves how to solve a peculiar problem. ML is a technique that is being hailed as a new direction of innovation to address many emerging challenges in department of optics. Over the years, its demand has unquestionably been on rise.

1.2 Motivation

Application based on internet, which fully rely on mesh of optical network to fulfill connectivity requirements. To achieve this kind of support a lot of innovations have been seen like amplifier, laser and fibers etc. Traditionally, industry was using hardware equipment like photonic integration and space division multiplexing etc. then moved toward SDN. It also captured interest of service provider, now a day's network configuration is under control of SDN in large data centers, due to operational challenges in service provider, faster provisioning and to achieve high scalability it requires automation. A lot of challenges in optical network have been resolved fully or partially using machine learning paradigm.

A wider range of degrees of freedom (parameters) is available to system engineers like path, spectrum, modulation format, baud rate, single or multi carrier transmission, non-linearity mitigation solution and combinations of these light path parameters grow dramatically. Possibly, for all of these combinations QoT should be calculated. Marginated formulas like Power Budget and Gaussian model etc. works faster and more scalable but they are inaccurate, high margination, underutilization of network resources (up to extra 2 dB for design margins [3]). Machine Learning exploits knowledge extracted from field data like QoT of established light-paths for example using monitors (OPM's) at the receiver to predict the QoT of un-established light-paths. by using ML its not required to use complex analytical models, its fast and scale-able but it requires training phase with historical data.

Optical network is built to last and that is achieved by fail-safe approach. Which is feasible for design of traditional network but its in-feasible specially in terms of OLS. An intelligent network is essential for planning and controlling operations and parallel computing. ML algorithms have ability to learn from past and can predict future responses based on trained model. Therefore, application of ML in optical network to predict an OSNR of unestablished light path in OLS is vital during planning phase of network architecture. Using machine learning techniques to predict the Quality of transmission not a new strategy. A few researches have been conducted on machine learning in bettering the estimation. (references)

1.3 Research Objective

Main objective of thesis is to predict OSNR of an unestablished light. Machine learning technique depends upon data set having different system parameters. One approach is to setup hardware implementation to get system parameters. Although, it requires a lot of time for implementation. While alternative approach is to design emulator to get dummy data-set. Which is comparatively easy and require less time.

An Emulator is designed which consists of two main blocks EDFA (Erbium-doped fiber amplifiers) and VOA (variable optical attenuator). It's used to calculate gain, generate noise, amplify signal and noise of input spectrum. VOA are placed after each span's EDFA is used to emulate a 10-dB transmission loss per span. It attenuates signal as well as noise. A Supervised ML technique is used for prediction. ML basically estimates the output of an unknown function.

- features: the input of function
- Labels: the values to be estimated.

During the training phase ML learns from a dataset having features and labels generated by the line emulator which provides telemetry. Once the model is trained, set of new inputs containing those features is fed. In prediction phase labels are predicted by ML. For QoT-E possible scenarios are following.

- Improving QoT-E of already deployed light path, in which light path depends on card flex-rate. In this case ML prediction uses both spectral load and telemetry on ILAs. ML-assisted QoT-E improvement enables larger rate.
- Improving QoT-E prediction during planning phase where light path is not deployed yet. ML prediction can use only on spectral load.
- reduced margin LP deployment feasibility.

Wider the exploration of machine learning potentialities, leaded by a deeper awareness of its mechanisms, which is helpful for getting reliable predictions with less computational cost. It also useful in achieving low latency. So, OSNR is predicted for an unestablished light path by using interpolation on telemetry data has inaccuracy of 1.5dB on the other hand by implying ML techniques on 3000 cases, significant difference reduced to $\pm$2dB.

1.4 Organization of Thesis

The thesis is structured as follows:

- **Chapter 1** Introduces the thesis topic and relevant concepts.
- **Chapter 2** Provides the background and literature survey.
- **Chapter 3** Methodology
- **Chapter 4** Simulation and results
- **Chapter 5** Conclusion
- **Chapter 6** Future Work
- **References**

Chapter 2

Background & State of the Art

2.1 Background &Theoretical Concepts

2.1.1 IP over WDM network:

With the devolvement of internet technology, Packet based data transfer be- came more popular in the mature technologies such as an Internet Protocol. Different kinds of network technologies have been developed to provide supportive bandwidth and transmission rate to an Internet Protocol. WDM is the most promising technology that enables IP for long haul network. Optical layer uses the Asynchronous Transfer Mode (ATM) and Synchronous Digital Hierarchy (SDH) for the IP over WDM integration. This implementation of IP over WDM by using of intermediate layers not only reduced the bandwidth due to the addition large header information but also restrict the transmission rate as each intermediate layer support some specific transmission rate. In order to remove the limitation of both IP over ATM and IP over SDH, now IP packets are passed directly to WDM layer.

Fig. below illustrates the IP over WDM network architecture. The IP over the WDM network comprises of two basic folds, the IP fold and the WDM fold.

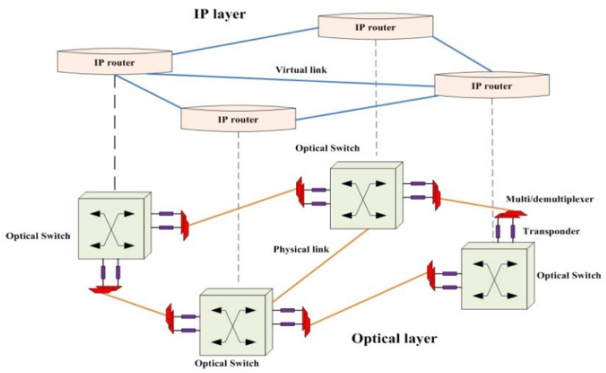


Figure 2.1: An IP over WDM Architecture.

Each node is equipped with an IP router at the IP layer while optical switches, OADM, EDFA and transponders are present at WDM layer. Basic function of the IP router is the traffic aggregation and its routing according to the requests. Similarly, the optical/WDM layer can supply huge bandwidth capacity to support high traffic

exchange among the IP routers. OXC's are inter-connected by means of optical fibers. OADM are used on each fiber to add/drop wavelengths. The transponders perform OEO conversion process. Finally, the EDFA's are deployed to magnify the optical signals on each link for a long-haul communication.

2.1.2 Erbium-Doped Fiber Amplifier (EDFA)

EDFA is an optical repeater device which is used to increase strength of signal. As fiber is doped with erbium and according to glass property light can be absorbed at a frequency and can be emitted at another frequency. To excite atom of erbium, a semiconductor laser couples light with fiber at infrared wavelength. A signal enters the fiber and stimulate the excited erbium to emit photons at the same wavelength as the incoming signal. By doing so, weak signal is amplified.

Usually, 10-span of EDFA's are used. EDFA's are placed after 80 km span so in total 800km can be covered by using chain of EDFA's. After 800km a repeater is required to reshape the incoming signal. First optical amplifier is EDFA's and its deployment was growing in 1990's.

The study presented in [10] author used chain of EDFA's to improve wavelength dependent power excursions in gain-controlled erbium doped EDFA. In another study [11] author presented his experiment in which evaluation of optical circuit switching is performed. It is said that by policing wavelength assignment gain offset can be reduced upto 0.7dB after chain of 5 EDFA's.

2.1.3 Variable Optical Alternator's (VOA)

Variable Optical Alternator's are used for controlling optical signal level. VOA's can be controlled electronically. In optical communication insertion loss and dynamic attenuation range are crucial. Typically insertion loss is less than 1 and dynamic attenuation range is in between 15-25. Attenuation in VOA's is manually tune-able. Figure 2.2 illustrates working principle of VOA.

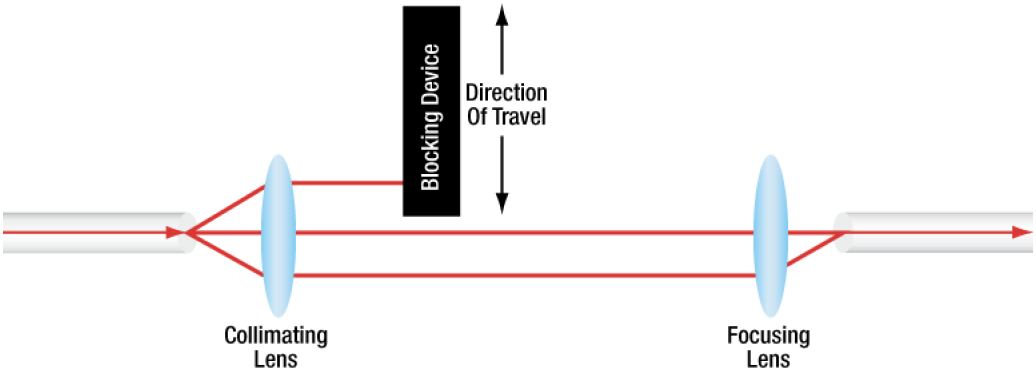


Figure 2.2: Operation of Variable Optical Attenuator

In figure 2.2 to collimate the light from input a lens is used an another lens is used at output of VOA to couple light. There is a blocking device in between lenses

it adjustment is manual. to obtain required attenuation coupling efficiency is tuned. Main applications of VOA's are following:

- Optical power equalization
- Channel transmission equalization in WDM networks
- Automatic gain configuration for optical amplifiers
- Protection of the optical receivers

2.1.4 Software Defined Network (SDN)

Traditional networks are vertically integrate in which control and data planes are within each device and control plane distributed across the switches/routers. its complex and hard to manage because control plane is distributed.Its difficult to understand the state of the network and its history every vendor has its own specific commands so is not easy to manage switches/routers from different vendors.

SDN is an Emerging networking paradigm in which control and data plane are separated. Routers and switches just acts as forwarding elements. Instead of destination-based forwarding SDN implements flow-based forwarding. A software platform running on commodity servers which is logically centralized.

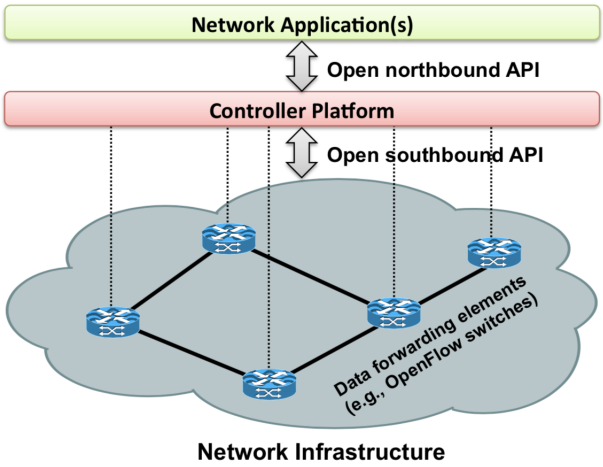


Figure 2.3: Simplified View of SDN Architecture

Network architectures of SDN is flexible, which is useful for complex and dynamic scenarios. In application of SDN it can be seen that, Google is using SDN in WAN to interconnect data centers and also within the data center. 5G networks are based on SDN. Moreover, ML is being used to make configuration of network more optimized in terms of feasibility and resource management. It can be seen that in study [13] author combines SDN with ML techniques and showed the benefits of Machine learning. Another study presented in [1] author introduced dynamic routing in SDN by using ML.showed that model performed much better on huge data-set and results are compared with historical methods.

2.1.5 Machine Learning (ML):

ML is branch of Artificial intelligence, it gives ability to learn without being explicitly programmed. In real world we have human and computers. Being a human, we use our body parts as an input to extract data from surroundings, then this data is used for meaningful purposes and in other words we learn, and we use this experience in future which we learned from past. Similarly, computers can learn as human do but we have to give learning ability to computer. So, by using that it can predict future events. That's what ML does. There are different ML techniques and algorithms are used for prediction. Main distinction of ML approaches are following

- Supervised learning

  In supervised learning, "labeled" data (i.e., "ground truth" input/output relationship) is given. For instance given traffic exchanges to or from a Data Center in past months or years and its task is to Predict traffic for the next period (regression problem) or Predict if available resources will be sufficient (classification problem) another example is where SNR is observed, Predict if quality of transmission will be degraded. some other examples are speech/image recognition, spam classifier and house prices prediction/estimation.

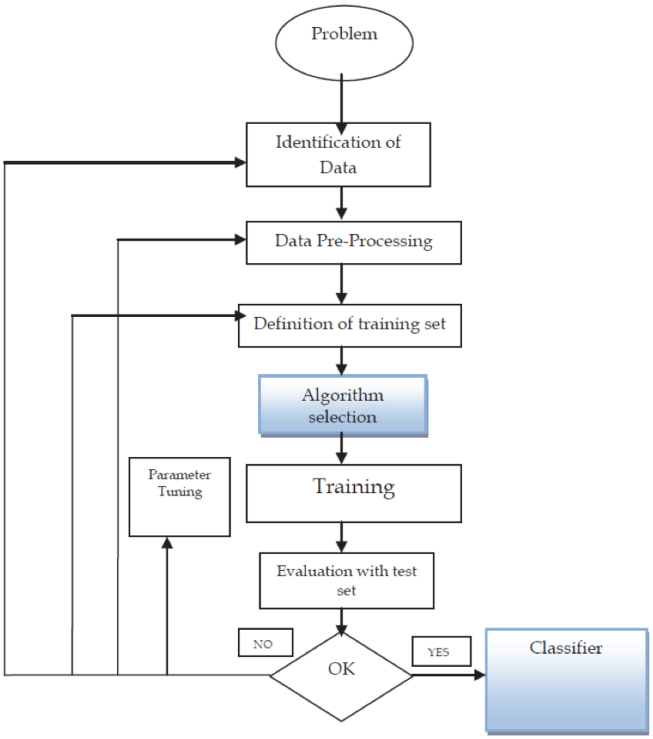


Figure 2.4: Supervised learning

- Unsupervised learning

  In unsupervised learning, available data is not "labeled". Main objective of unsupervised learning is to derive structures (patterns) from the available data, finding "groups" in data according to a similarity measure and anomaly detection. For example, traffic traces in different cell sites are given, its task is

to understand if some cells provide similar patterns or not. They might cover same type of urban areas (theatre, cinema, stadium etc). This information can be used to make network resources planning. some other examples are, Group people according to their interests to improve advertisement and group together different genes if they provide similar information

- Semi-supervised learning

  Semi-supervised learning approaches are mixture of the previous two (supervised and un-supervised) presented in above section, this approach address toward problems where majority samples of the training are unlabeled, even though only limited data points with label are available. Advantage of this is that in various areas a huge amount of unlabeled data points is willingly available. Applications where semi-supervised learning is used are nearly same as supervised learning. This type of learning is mostly beneficial when the labeled data points we have are not too common or so exclusive to get then using of that unlabeled available data points can rise the performance.

- Reinforcement learning

  Reinforcement Learning (RL) is one of the machine learning technique used in general, in applications such as robotics, finance, and record management, the ultimate objective here is to learn a policy. For example, a mapping between states of the environment into actions to be performed, while directly interacting with the environment. The RL allows the agents to learn by experiencing the available actions and decontaminating their conduct by only using an evaluative response, referred to as the return. The ultimate objective of the agents are to increases its performance for long time. Therefore, the agents does not only consider the instant reward, but even it assess the penalties of its actions on the future. Late reward and trial and error establish two of the most significant features of RL. The context of performance of RL is typically Markov decision processes (MDP).

Widely used ML algorithms are following,In general these are applicable on any kind of data.

- Linear Regression

  Main goal is to predict a line with smallest prediction error that fits the data. LR is used to find linear relationship between target and predictors. LR is sub divided into two categories, Simple LR and Multiple LR. In simple LR a single variable is used for the prediction. It is used when relationship between two continuous variables is required like predictor and response. While multiple LR is used when two or more than variables are used to predict a variable.

- Support Vector Machine(SVM)

  This approach is presented by Luis Gonz in 2005 as SVM implements classification by building N different Dimensional hyper plane which efficiently

splits data into two different classes. Support Virtual Machine models are thoroughly correlated to neural networks. It is obvious, that the use of a SVM model with a sigmoid kernel function is comparable to a two layer, also known as perceptron neural network. SVM models are very close to classical multi-layer perceptron neural networks. By using a kernel function, SVM's are an alternate training technique for , multi-layer perceptron classifiers and polynomial where the heaviness of the network are establish by resolving a quadratic programming problem with linear constraints, comparatively by solving a non-convex, unconstrained minimization problem in a standard neural network training. In phraseology of SVM literature, an interpreter variable is known as an attribute, while a transformed attribute which is used to define the hyper plane is known as a feature. The job of selecting the utmost fit demonstration is called feature selection. A group of features that defines one case is known as vector. In nutshell the objective of SVM modelling is to find the best hyper plane which splits bunches of vector in a manner that fall in one group of the targeted variables in one cases and the other group are on the other side of the plane. The vectors which fall nearer to the hyper plane are the support vectors.

- Random Forest

  Mechanism of Random forest is to built multiple decision trees and merge them as illustrated in 2.5. Merged decision tree is used to predict, its prediction is more accurate and stable than simple decision trees.it can solve classification problem as well as regression problems.Random forest algorithm is also used to check importance of features like which feature is contributing more to make prediction accurate. Feature importance is vital for ML because if more features are used for prediction the more chances are that model will suffer from over fitting. so features that are not contributing toward the prediction can be removed to get more accurate prediction. Random forest is much better in prediction as compare to decision tree because in random forest prevents model to over-fit by creating sub-set of features and these sub-sets are used as trees. but there is no surety of this to happen every time.

- Logistic Regression
- Naive Bayes
- K-Means
- Dimensionality Reduction Algorithms
- Gradient Boosting algorithms

In [18] author performed anomaly detection and compared different ML algorithms like (SVM), (RF) and neural networks.

2.2 State of Art - ML in Optical Network

Use of ML in optical networks is an area under research to improve performance. Main purpose of this section is to overview applied techniques in optics. Firstly,

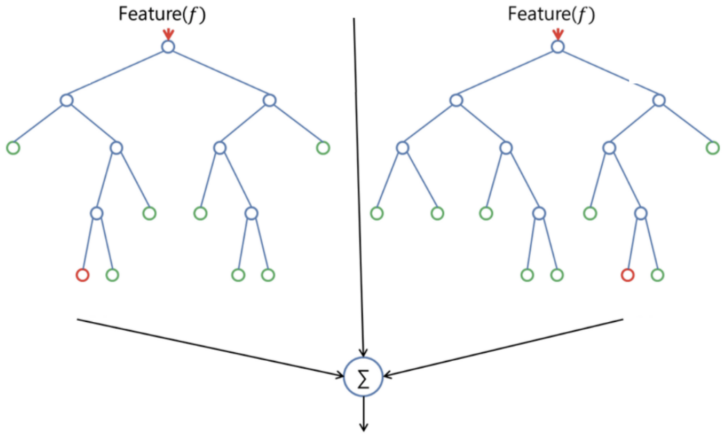


Figure 2.5: Random Forest

wide range of ML algorithms are applied on EDFA. Basic responsibility of EDFA is to regenerate incoming signal by using DWDM (dense wavelength division multiplexing). EDFA plays an important role in optical transport network but it also poses some challenges to optimize performance of link. ML proposes an efficient solution to these challenges.

A regression problem is described in [9] by using ML techniques and multi span EDFA's to model the dependence of channel on power excursion. This model based on historical data. To reduce power disparity in between channels obtained from AGC (automatic gain controller), it's recommendations to add or drop channel are accurate. But it's not suitable for live network in which configurations are changing dynamically - Is it Suitable for an EDFA network having different design – to cope with problem of power excursion in live network an extended study [10] is proposed. This research emphasizes on improving spectral efficiency in which dynamic defragmentation plays a key role on its improvement. Ridge regression and logistic regressions are used for magnitude of sub-channel and to compute weather its addition will increase or decrease power discrepancy. Then ML algorithms is applied to determine power adjustments. Post EDFA power discrepancy among channels are showed by 62%.

Finally, an automatic approach is presented in [2] by using a MLP (multilayer perceptron) neural network to adjust operating point of amplifiers in cascade of EDFA's. performance of link is optimized by reducing noise figure and ripple of frequency response of transmission system while it ensures predefined power levels of input and output. As compare to previous studies [17], The main endowment of ML is integration of ensuring predefined power levels of input and output to adjustment process. Indeed, A link is returned having noise figure 30.06 dB and frequency response 5.2 dB by defining the gain of 6 amplifiers to maintain input and output power levels around 3dB. Presented error is less than 0.1dB in this study.

Another broadly used trend is to review compatibility of ML applications techniques in monitoring and mitigating the effects which degrades the performance of links like (PMD) polarization mode dispersion, (CD) chromatic dispersion and

(OSNR) optical signal to noise ratio. due to use of higher transmission rates and advanced modulation formats, the transmission nature inside the optical fiber is becoming nonlinear, which is causing dynamic and heterogenous transmission systems.

An approach presented in [6] to cope with tendency described above is to use (OPM) optical performance monitoring to estimate physical parameters of optical signal. Which ensures efficiency and robustness of network operation. It diagnoses network to detect malfunctions, damages and their repairment. It also reroutes traffic where link is non-optimal in network.

In order to monitor OSNR, another study presented in [19] in which DNN is trained on previous raw data. Although this study has limited scalability but OSNR is predicted in range of 7.5 to 31 dB correctly. In order to improve accuracy of this model DNN should be trained around 400,000 samples, and it must be configured with 5-layers at least. For this kind of training it requires a lot of time to train the model. More advanced OSNR estimator presented in [20] which is useful for the systems having modulation (up to 64-QAM). Author applied neural network and SVM. It can be seen that, predicted OSNR from modulation format classifier achieves 94 % accuracy on the other hand estimator reaches at 0.7dB error. which is total mean estimation error, in worst case it is 3.5dB. in this approach only, white Gaussian noise is considered while nonlinear and linear optical fiber impairments are ignored.

In another study same trend has been Followed but it emphasizes more on the mitigation of nonlinearities in optical fibers transmissions, last year extensive research has been done on NLPN (nonlinear phase noise). Which plays an important role. To solve this issue an electronic method is used which rely on deterministic information of fixed fiber link. It can be found in [12] and [16] as maximum likelihood estimation and digital back propagation respectively. Now a days, ML techniques are being applied on digital signal processing to mitigate nonlinearities in a more efficient way which allows more accurate symbol detection.

ML techniques are also being used to improve the performance of optical network in Cognitive dynamic optical networks [14] which describes the current situation of network. To make it an adaptive network it modifies the configuration of network. It makes use of past history to make decision. If situation is different than past, it can act differently. As this approach is using past experience in making decision so to cope with heterogeneity, it a perfect candidate for improving quality of transmission and services.

Usually, cognition is developed in CHRON (Cognitive Heterogeneous Reconfigurable Optical Network) to extract state of art its subdivided into two categories.

- Reconfiguration of virtual topologies.
- Quality of transmission estimation.

Reconfiguration of virtual topologies is related to LP which optimize the performance of network by tuning the different parameters like by reducing energy

consumption, congestion controlling, delay, QoT and blocking probability etc. Instead of configuring network statically it is reconfigured dynamically according to traffic pattern. To design virtual topologies genetic algorithm is used in [10]

to control congestion and energy consumption. Additionally, by using genetic algorithm up to 20-25% savings can be made in operational cast and also in capital have been presented in [7]. To reconfigure virtual network, monitored data is used to estimate that is useful to cope with dynamic changings in traffic proactively. Another approach for reconfiguration the virtual network presented in [15] is based on volume of traffic and a direction is predicted based on destination by neural network. Time to time data collection is performed to train model for prediction in near future.

Finally, quality of transmission estimation is very important in network design and operation. A quality estimator "QTool" is presented in [3]. It evaluates quality factor of LP's in a topology. Accuracy of this estimator is admirable, but it requires a lot of time for computation. To overcome this, researcher have proposed cognitive approach.

Chapter 3

Methodology

In this study, main goal is to estimate transmission multivendor optical open line system using machine learning techniques. ML technique needs a data set of different system parameters. In order to achieve this, an emulator is designed to get synthetic dataset of system parameters. Firstly, interpolation is applied on synthetic data to predict OSNR as well as ML algorithms like LR and DNN regression are applied for estimation. Results are compared, it is observed that ML performs much better than interpolation. Secondly, after preparing experimental setup in lab telemetry data is collected and again interpolation is applied on telemetry data to predict OSNR as well as ML algorithms are applied for estimation. During the comparison, results showed that QoS Estimation by ML performs much better on telemetry data as compare to interpolation.

3.1 Emulator Design

Emulator consists of two main blocks which are following.

- EDFA
- VOA

As described in study presented in [26], Numerous EDFA's are cascaded in testbed having different brands and models. A brief review of system model is shown in figure 1.1 each EDFA exhibits different gain tilt and shows different behavior. To attain 20dB attenuation per span, VOA are placed after each span's EDFA. A regression model is used to predict OSNR by using channel OFF and ON states and also the power levels. The SNR of light path over several OLS is following.

$$\frac{1}{SNR} = \sum i \frac{1}{SNR_i}$$

Here SNR_i is the contribution of SNR of every crossed OLS. In general SNR is following.

$$\frac{1}{SNR} = \frac{1}{SNR_{NL}} = \frac{1}{OSNR}$$

Where $\frac{1}{SNR_{NL}}$ is calculated using Non-linear model but in this case, it is 0. While $\frac{1}{SNR_{NL}}$ is calculated using Machine Learning.

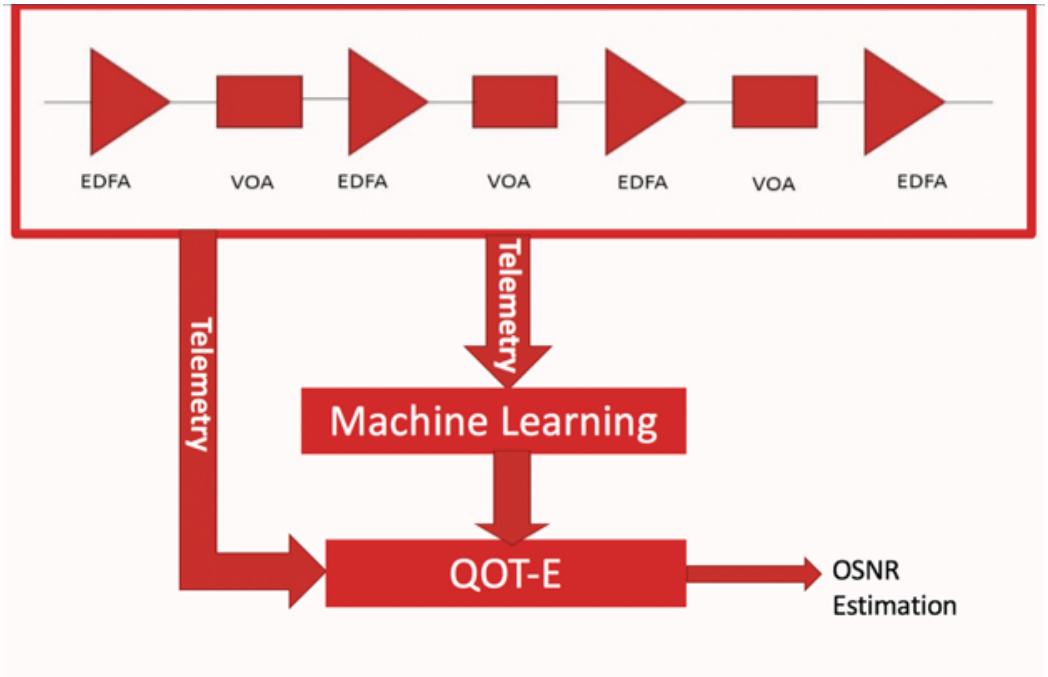


Figure 3.1: Chain of EDFA's

3.1.1 EDFA Design:

Cascaded EDFA's are being used between spans of transmission fiber. Advantage of using EDFA's in fiber is that it compensates transmission loss without letting the power level low, which can cause fiber non-linearities and destroy SNR ratio. To implement EDFA two parameters are considered.

- Gain (G)
- Noise Figure (NF)

Two modes of emulator are considered

- Mode 1 = Fixed Gain
- Mode 2 = $FixedPout_Target$

Polynomial expansion is used to calculate noise figure considering frequency dependent Gain.

$$NF(f) = nf_0 + G(f)nf_1 + G(f)^2nf_2 + G(f)^3nf_3 + \dots$$

Where G(f) is gain of EDFA, dependent on frequency and nf_i is its noise figure coefficients. On the other hand, flat (NF) noise figure have fixed value. Gain ripples G_rpp is generated using circular shift, to generate it for each EDFA three modifications are following.

- Change the range between the maximum and the minimum of the gain ripple for every EDFA.
- Shift the gain ripple by a certain amount of wavelength for every EDFA.

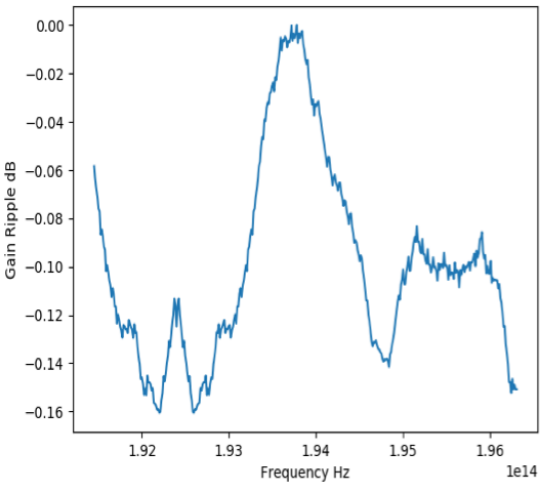


Figure 3.2: Ripples Gain without tilt

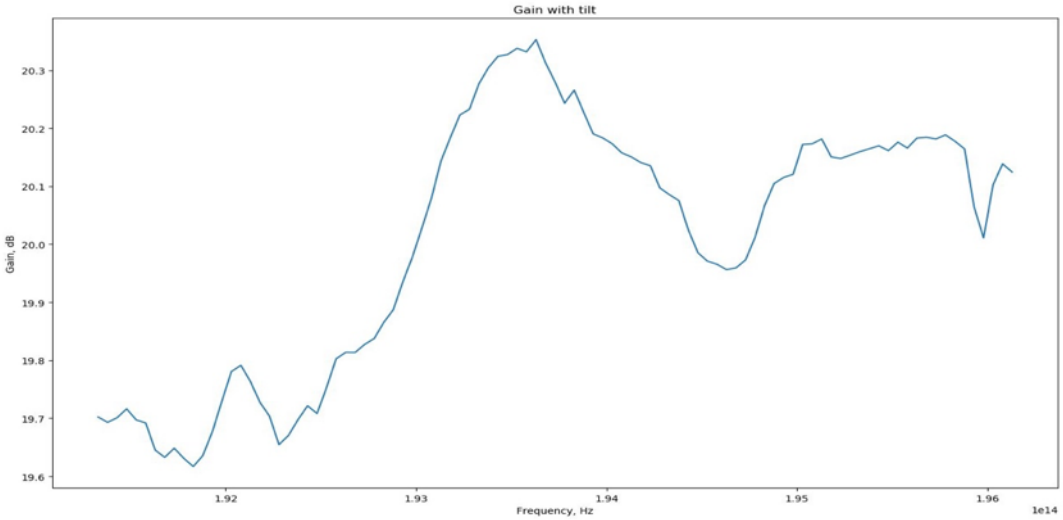


Figure 3.3: Ripples Gain with tilt.

- Introduce a certain tilt in the Gain of each EDFA to compensate the tilt introduced in VOA.

Ripples Controlling Parameter are following

$$G_{rpp\_k} = kG_{rpp}$$

Where

$$k = \frac{\Delta G_m}{\Delta G}$$

$G_{SHB\Delta}(f)$ is depended on input power so it is not the characteristic of EDFA on the other hand $G_{rpp\_k}(f)$ is independent of input power so it is considered as characteristic parameter of EDFA. Equations are following

$$g(f) = G_{SHB\Delta}(f).G_{rpp\_k}(f)$$

$$K = \frac{G_{Target}}{g(f)_{avg}}$$

$$G(f) = K.g(f)$$

Spectral hole Burning (SHB) having parameters

- $G_{SHB} = 1$ 1528.1nm -1568nm
- $G_{SHB} = f(sin)$ for particular wavelength window (centering at 1531nm)

Slice size parameter is $G_{SHB\_\Delta} = \Delta G.G_{SHB}$

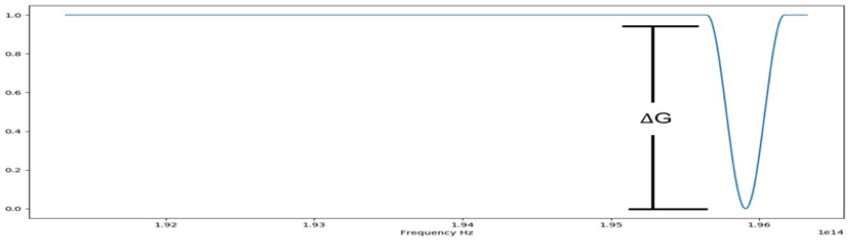


Figure 3.4: Spectral hole Burning

In EDFA design ASE Noise is

$$P_{N(f)} = hf_c(G(f) - 1)NF(f)B_{ref}$$

Where, "h" is Plank's Constant, "f_c " is Central Frequency of C band, "Bref" is Reference bandwidth And OSNR is

$$OSNR(f) = \frac{P_s}{P_N(f)}$$

3.1.2 VOA Design:

Variable optical attenuators are used to reduce power level of optical signal. Output of VOA are signal and noise. Signal can be described as

$$P_{out\_Signal} = \frac{P_Signal}{Attenuation}$$

$$P_{out\_Noise} = \frac{P_Noise}{Attenuation}$$

And Attenuation is as following,

$$Attenuation(f) = Loss + tilt(\Delta f)$$

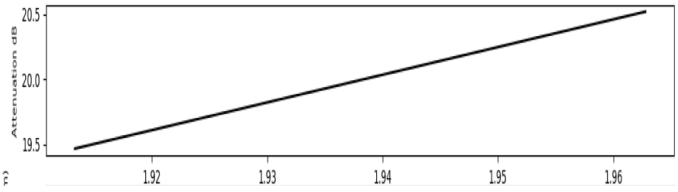


Figure 3.5: Attenuation

3.1.3 Emulator Schematic:

During first phase gain ripple is generated.

Figure 3.6: Emulator Schematic Phase-I

During secound phase

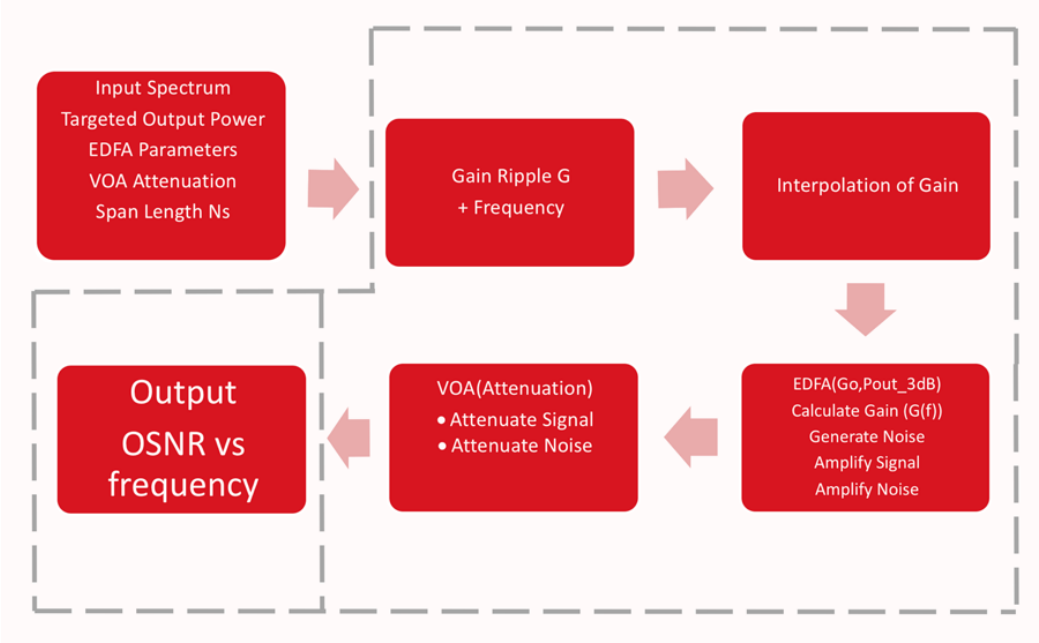


Figure 3.7: Emulator Schematic Phase-II

Finally, the developed line emulator is providing synthetic telemetry data. OSNR is calculated at the end of emulator chain.

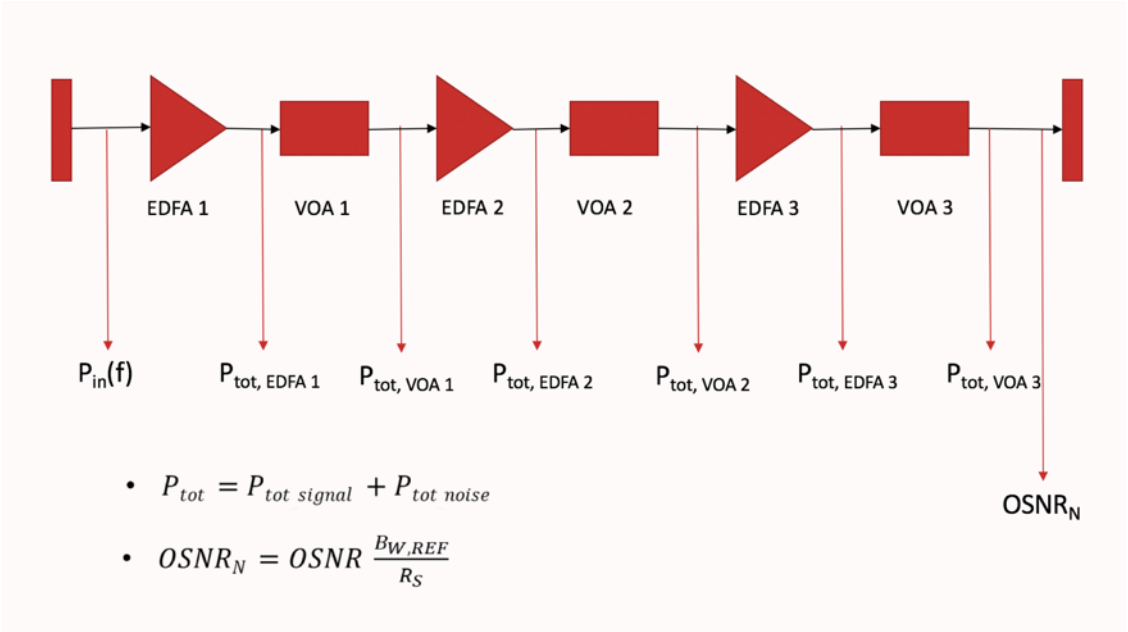


Figure 3.8: EDFA's Chain

PARAMETERS OF EMULATOR	
Number of Amplifiers	**10**
Number of VOAs	10
Number of Channels	96
Attenuation	20dB
Input power per channel	-16dBm
Number of NF coefficient	8
Targeted output power	10dBm
Frequency distribution	191.327 THZ- 196.322 THZ
P_max	2mW
ΔG_{max}	0.8 dB
NF Coefficients	[-2.0286, -0.00096452, -4.5079e-08, -1.3439e-13, 1.2203e-10, 8.7561e-06, 0.0605, 32.8541]

Figure 3.9: Emulator Parameters

3.1.4 Results From Emulator

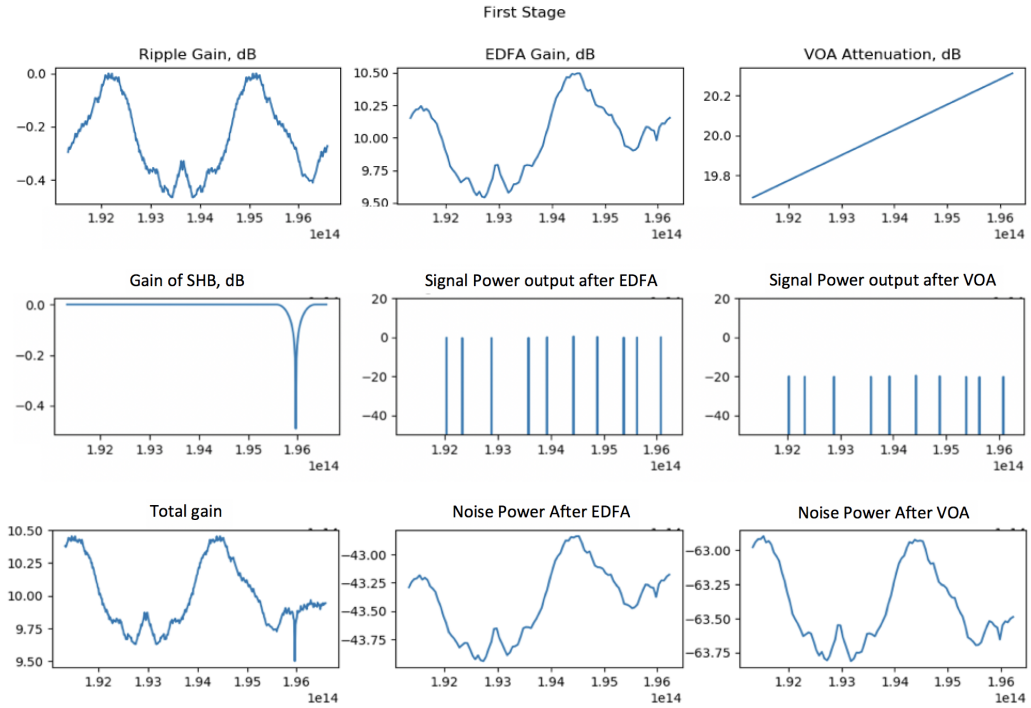


Figure 3.10: Output after the 1st EDFA

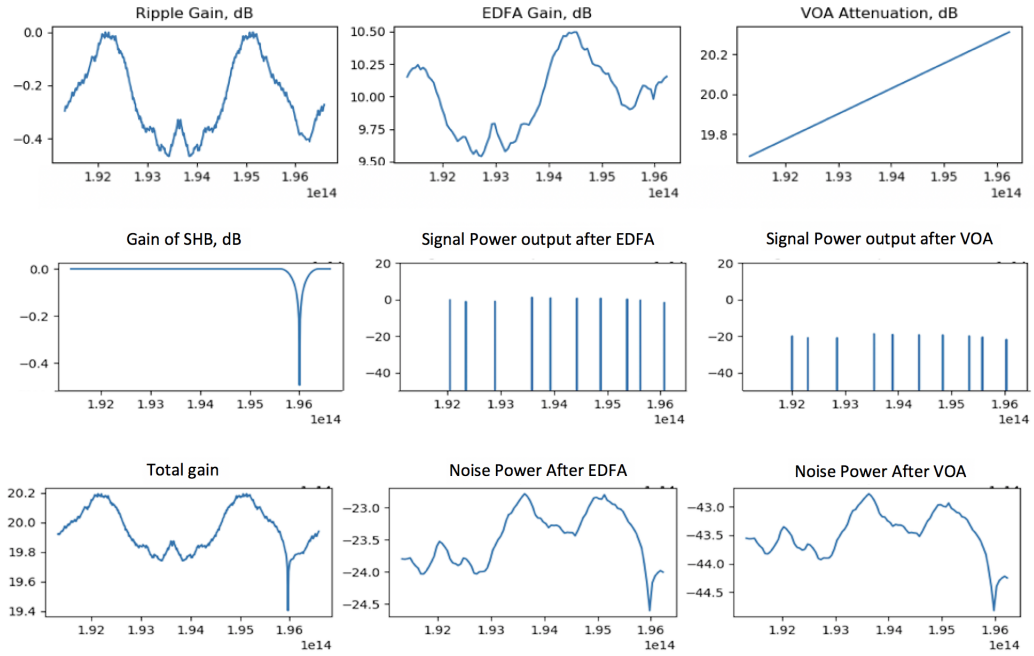


Figure 3.11: Output after the 10th EDFA

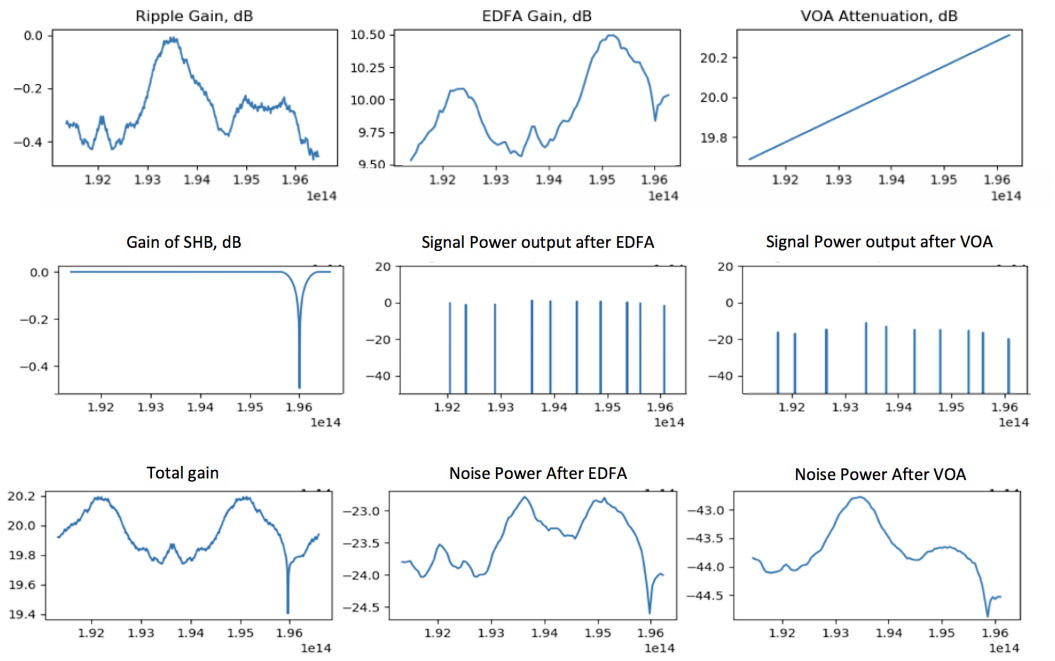


Figure 3.12: Output at the end of EDFA chain

Test channel is central channel and OSNR values are depicted in table 1.1. Figure below shows OSNR comparison when only 10 channels are on and with full spectral load.

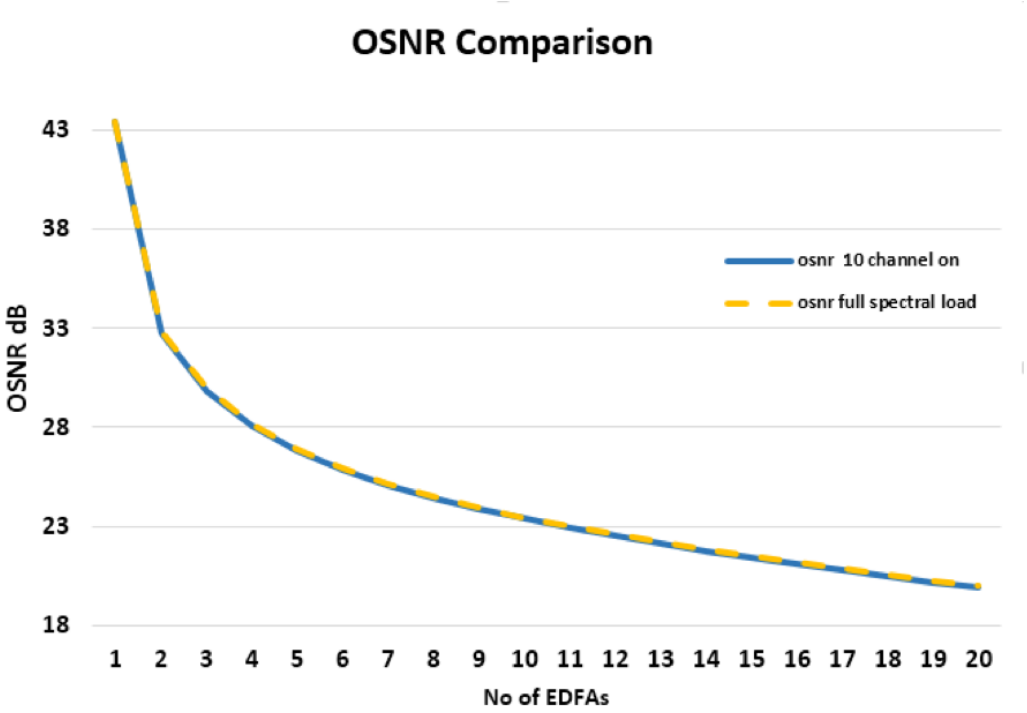


Figure 3.13: OSNR vs Number of EDFA's

At start 0 is booster, after that number of EDFA's are added. Curves shows that with the increase in number of EDFA's OSNR decreases. Usually span between two adjacent EDFA's is 80 km for single mode fiber. As OSNR is decreasing, at some

point it requires to regenerate the signal. When test channel is in Spectral hole

After EDFA	Case 1 (10 channels)	Case 2 (full load)
1	43.458496	43.4434
2	32.72673	32.8459
3	29.848955	29.9409
4	28.073635	28.1551
5	26.778596	26.8566
6	25.881328	25.9552
7	25.094825	25.1662
8	24.457289	24.5278
9	23.885648	23.9545
10	23.369646	23.439
11	22.92556	22.9955
12	22.517102	22.5882
13	22.139528	22.2117
14	21.78349	21.8567
15	21.440825	21.5147
16	21.103212	21.1756
17	20.777925	20.8497
18	20.476569	20.5471
19	20.189682	20.2581
20	19.906613	19.9713

Table 3.1: OSNR vs Number of EDFA's

burning (SHB) Region then OSNR comparison is shown in figure 3.13. while values are depicted in table 3.1

3.2 Experimental Line System:

To simplify the problem under investigation a line system is deployed with variable optical attenuator in place of the fiber, in order to avoid non-linear effects. This experimental setup is based on commercial EDFAs [commercial EDFAs]. The optical line is composed by 11 spans, in each span VOA is followed by an EDFA. VOA is placed for span attenuation of 10 dB.

Optical Spectrum Analyzer (OSA) is an instrument which measures and displays the distribution of power of an optical source over a specified wavelength span. It has bandwidth of 3.5[TH]. In graph 3.16, it can be seen that where channel is ON we have power of channel (Showed as peaks) but where channel is OFF, peak is not observed and there is ASE noise as power received.

3.2.1 Optical Power Meter (OPM)

Wavelength Selective Switch (WSS) has a single optical port as an input and N opposing multi-wavelength as output ports. where input can be switched to any

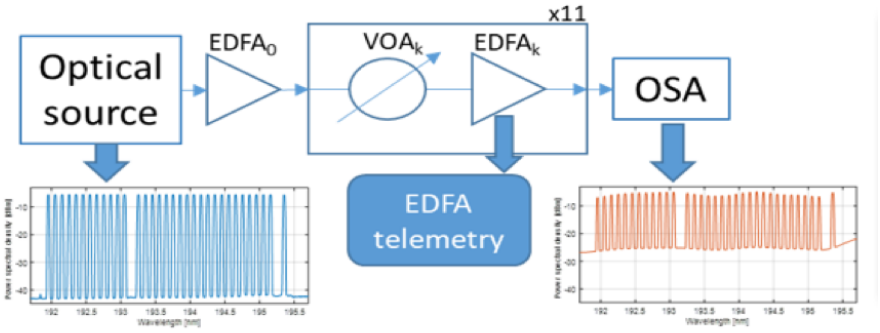


Figure 3.14: Experimental setup Diagram

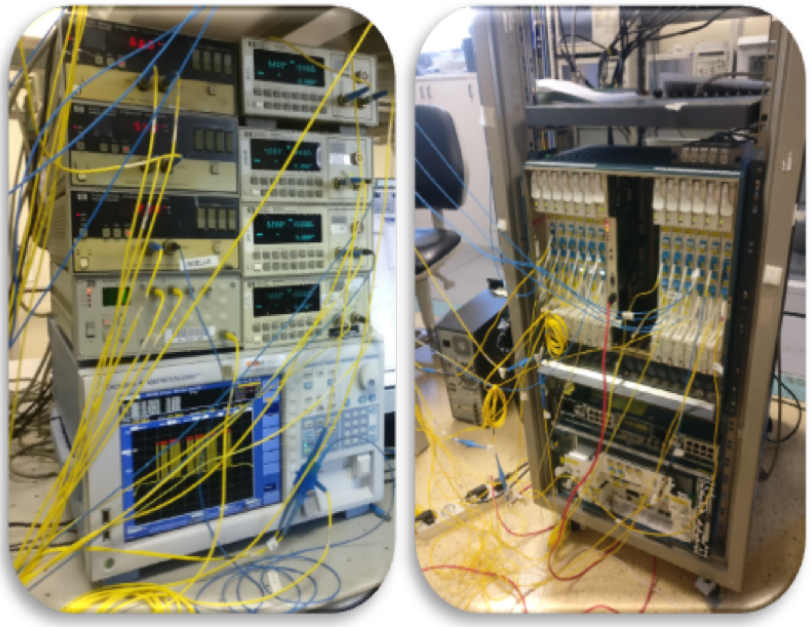


Figure 3.15: Lab Experimental setup

port independently from other routed channels. Filters are added to 100GHz power of each channel. So, in total it produces 35 channels.

3.2.2 ML-Based QoT-Estimation

Training ML: After developing line emulator next step is to train the ML with a subset of cases by providing spectral load, telemetry and corresponding OSNR values. Basically, ML estimates the output of an unknown function so here "Feature" is input of our function and "Label" is the value that we want to estimate. During training phase ML learns from dataset of Features and Labels, which are generated by line emulator. While on the other hand during prediction phase set of features are feeded in ML to estimates their corresponding label. As shown in schematic figure 3.17 During the training phase 99 channels are used in wavelength of 1528-1568nm. Fifteen EDFA's are used having working mode with "Fixed Power" which is 0 dBm. Frequency at center of spectral hole burning is most affected by EDFA during amplification and that frequency is 1531. So, CuT is fixed for central chan-

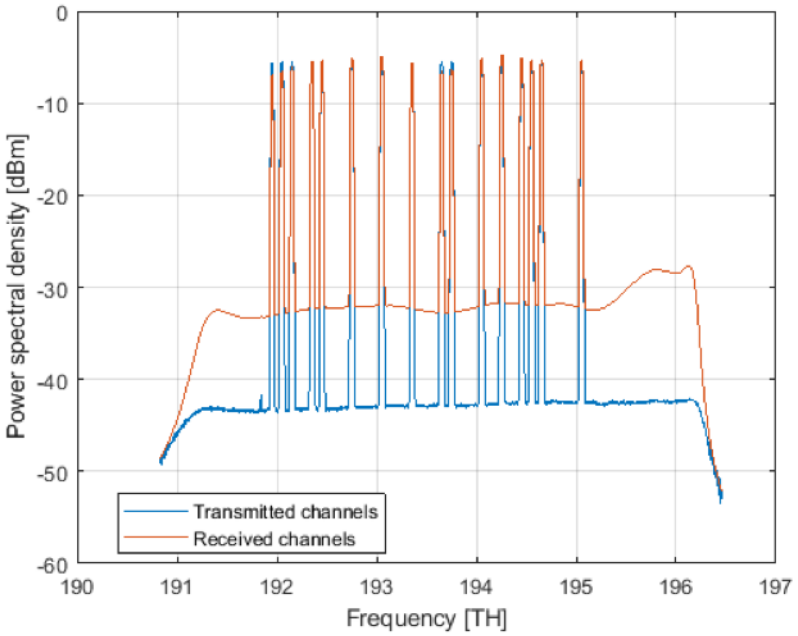


Figure 3.16: Optical Spectrum Analyzer(OSA)

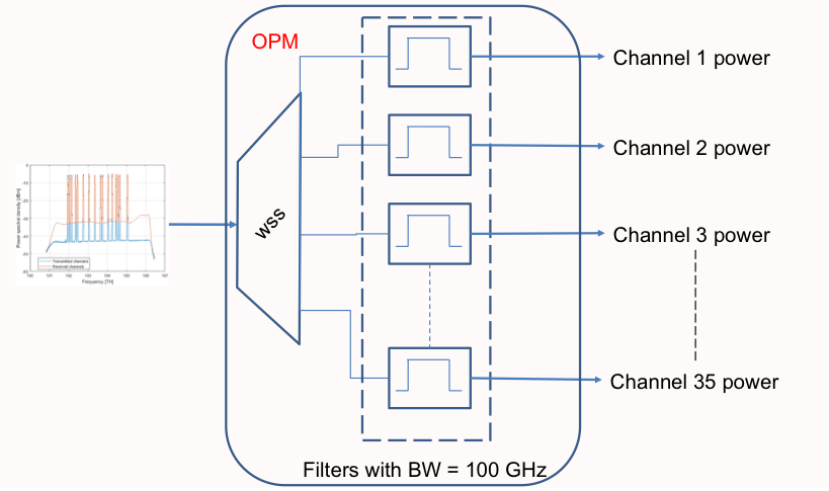


Figure 3.17: Schematic Diagram of OPM

nels, the channels with spectral hole burning. By keeping CuT ON, progressively spectrum is loaded with interfering channels N_{int} . N_{int} starts from 0 to $N_{ch} - 1$, where $N_{ch} = 99$. Interfering channels frequencies are generated randomly. And to grow the dataset with different cases, each N_{int} is repeated for N_{try} times which is 10. Line Emulator provides Telemetry and OSNR of CuT for this configuration, used to train the ML.

After training the ML, the OSNR of CuT is predicted in a general case. Then OSNR of CuT provided by the Line Emulator is compared to the ML prediction. To predict OSNR only telemetry and spectral load of the test case are given to ML and ML estimation is compared with Line Emulator OSNR. Synthetic Data generated from emulator is subdivided into training data and prediction data. Its main purpose of this division is to make data rich enough and flexible for machine learning estimator. This data is in numerical form.

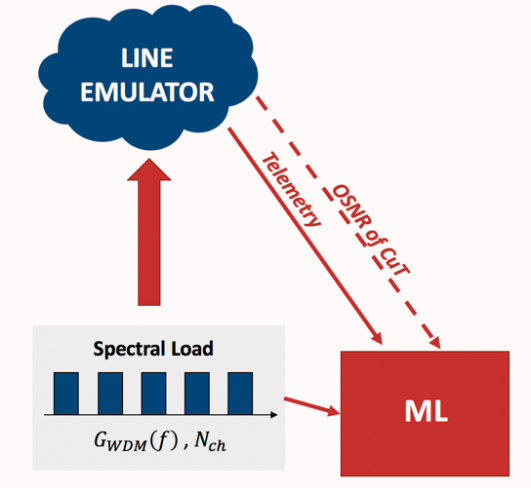


Figure 3.18: Overall Model

features	Description
Spectral Load	Channel ON along the spectrum
$P_{tot,in}$	Total power at the input of each EDFA
$P_{tot,out}$	Total power at the output of each EDFA
label	**Description**
$OSNR_{CuT}$	OSNR of the Channel under Test

Figure 3.19: Feature and Label Parameters.png

Figure 3.20: OSNR of CuT provided by the Line Emulator

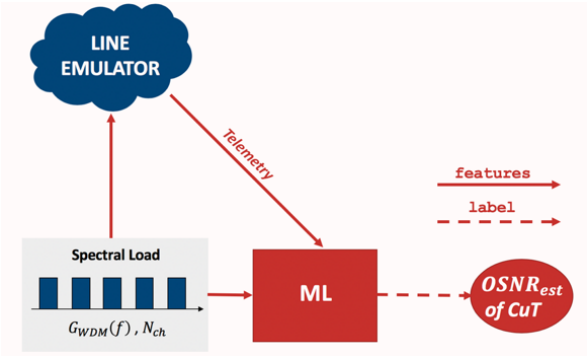


Figure 3.21: Diagram for OSNR prediction

Selection of hyperparameters is vital in machine learning estimation. Some ML algorithms explicitly defines hyperparameters which controls the execution of algorithm. In order to achieve best results in prediction practitioners tunes a lot hyperparameters. In this study shuffle, batch and step sizing is used and tuned as hyperparameters. Overall ML model is depicted in Figure 3.23

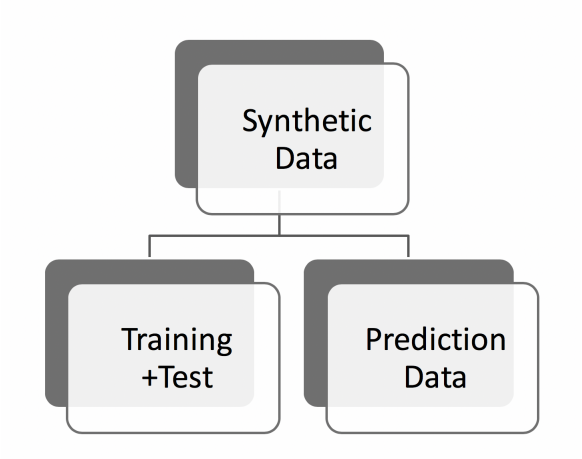


Figure 3.22: Dataset Preparation

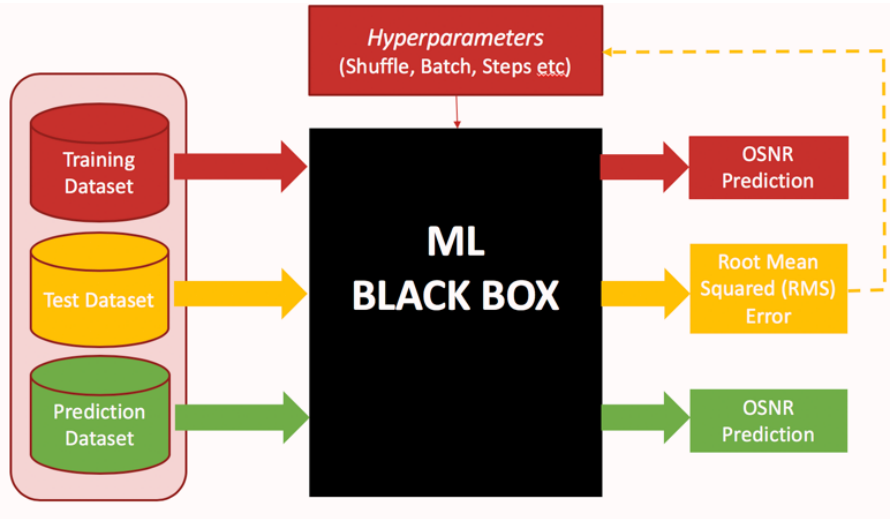


Figure 3.23: ML Model

An open source ML library "TensorFlow" is used in the following DNN and Linear Regression algorithms.

Algorithm 1 Linear Regression Algorithm

1: *InputParameters:*
2: $features(f_r), TrainingRatio(T_r), TrainingData(T_d), Shuffle, Batch$
3: $(PredictionData(P_d), Delta(), RootMeanSquare(rms), Steps(S_p)$

4: *Build the training:*
5: $(Train, Test) = TrainTestSeperator(T_r, T_d)$
6: $InputTrain = ShuflingAndBatching(Train, Shuffle, Batch)$
7: $InputTest = ShuflingAndBatching(Test, Shuffle, Batch)$

8: *Build the Estimator:*
9: $model = tf.estimator.LinearRegresson(f_r)$

10: *Train the model:*
11: $model.train(InputTrain, steps)$

12: *Evaluation of model:*
13: $evaluate\_result = model.evaluate(InputTest)$
14: $rms = CalculateRMS(res)$
15: $Predict\_results = model.predict(P_d)$

The algorithm 1 Builds, trains, and evaluates the model. Procedure InputTrain Builds the training input function. While InputTest Builds the validation. At line 7 and 12 data is shuffled with a buffer larger than the data set ensures, that the examples are well mixed. Line 8 is to repeat forever. During Training of the model, the Estimators log output every 100 steps by default. But this hyperparameter is tuned with different number of steps. At the end root mean square error of model is calculated, then prediction is performed.

Algorithm 2 DNN Regression Algorithm

InputParameters:
$features(f_r), TrainingRatio(T_r), TrainingData(T_d), Shuffle, Batch$
$(PredictionData(P_d), Delta(), RootMeanSquare(rms), Steps(S_p)$

Build the training:
    $(Train, Test) = TrainTestSeperator(T_r, T_d)$
    $InputTrain = ShufflingAndBatching(Train, Shuffle, Batch)$
    $InputTest = ShufflingAndBatching(Test, Shuffle, Batch)$

Build the Estimator:
    $model = tf.estimator.DNNRegressor($
        $f_r, HiddenUnits = [32, 64, 128]$
        optimizer=lambda: tf.train.AdamOptimizer(
            learningRate=tf.train.exponentialDecay(
            learningRate=0.001,
            globalStep=tf.train.getGlobalStep(),
            decaySteps=10000,
            decayRate=0.096))
)

Train the model:
    $model.train(InputTrain, steps)$

Evaluation of model:
    $evaluate\_result = model.evaluate(InputTest)$
    $rms = CalculateRMS(res)$
    $Predict\_results = model.predict(P_d)$

Mini batch training technique is used, in which data is shuffled and batch are made to converge fast training. To avoids model from learning the order of the training. It also helps to avoid biasness during training phase.

- Shuffle: The size of data taken from Training data set to be shuffled in a single step.
- Batch size : Batch size of dataset from the shuffled Training Data for a single training step.
- Steps: Tshe total number of training iterations done. One step calculates the loss from one batch and uses that value to modify the model's weights once.

  Total number of trained samples = Batch size x Steps

- Learning Rate: The amount that the weights are updated during training is referred to as the step size or the "learning rate", it's a tunable hyperparameter used in the training of neural networks and it has small positive value. In general, learning rate should be which is low enough that the network converges to something useful, but high enough that don't take too much time to train the model. When learning rate increases training time increase respectively.

- Hidden Layers: Inspired by biological neural network, in terms of computer science its represented as a set of layers. These layers are categorized into three layers input, output and hidden layer, in between input and output. Every network has single input and output layer but different number of hidden layers. It's a big challenge to identify number of hidden layers. In DNN algorithm defined above default number of hidden layers are used.

- Root Mean Square Error: RMS value is calculated for the test data set on the basis of difference of actual value and predicted one.

3.3 Dataset Preparation

35 channels are randomely selected from the specterum of 97 channaels. Approximately 1152 samples are used to train the model. As shown in figure 3.24 in-circled channel under test is label and rest are features.

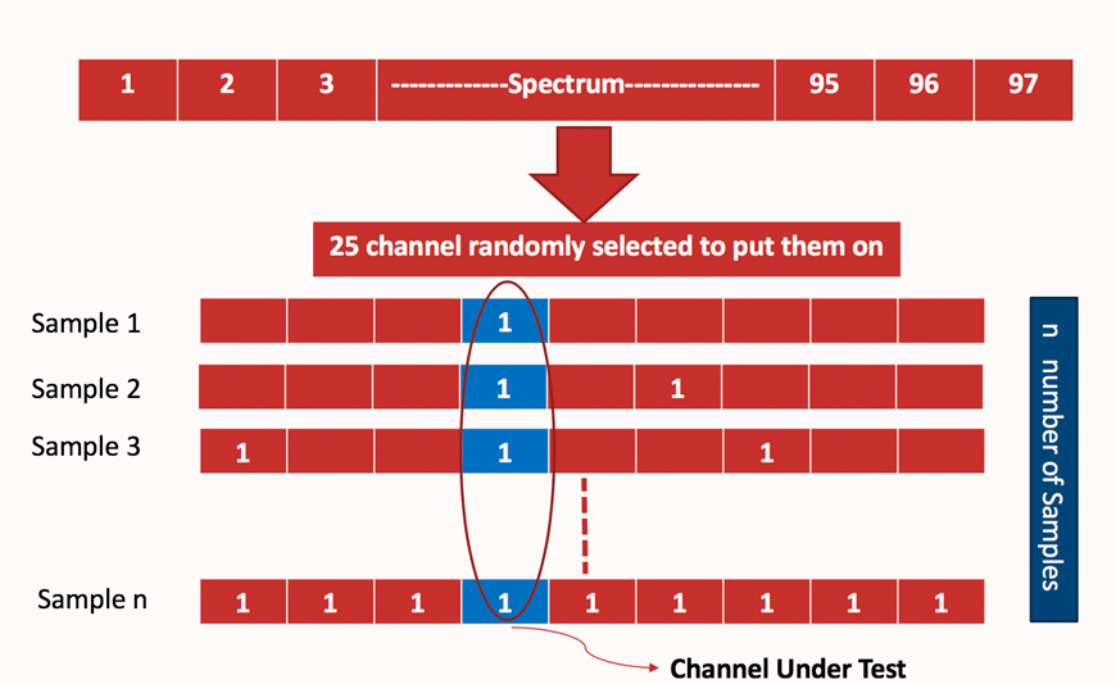


Figure 3.24: Dataset for simulation

Total number channels are 97, Max number of channels (ON)= 25 and allowable channels on fixed slots are 25. Here "n" represents number of emulator iterations for each number of active interfering channels.

Samples = Dataset size

Samples (Size of Dataset) = 15x25 = 360 (n = 15 for case 1)

- Training + Test samples = 360

- Training Samples = 70 % of 360

- Test Samples = 30% of 360

Main goal is to Predict OSNR of Test Channel, which is at the center of Spectral Hole Burning range. Then to deviation is computed by following formula.

Deviation = Predicted OSNR (from ML) – Telemetry OSNR (from Emulator)

From given samples ML shuffle and makes batchs of 1152 samples. To improve ML quality and it also improves predictive performance of model.

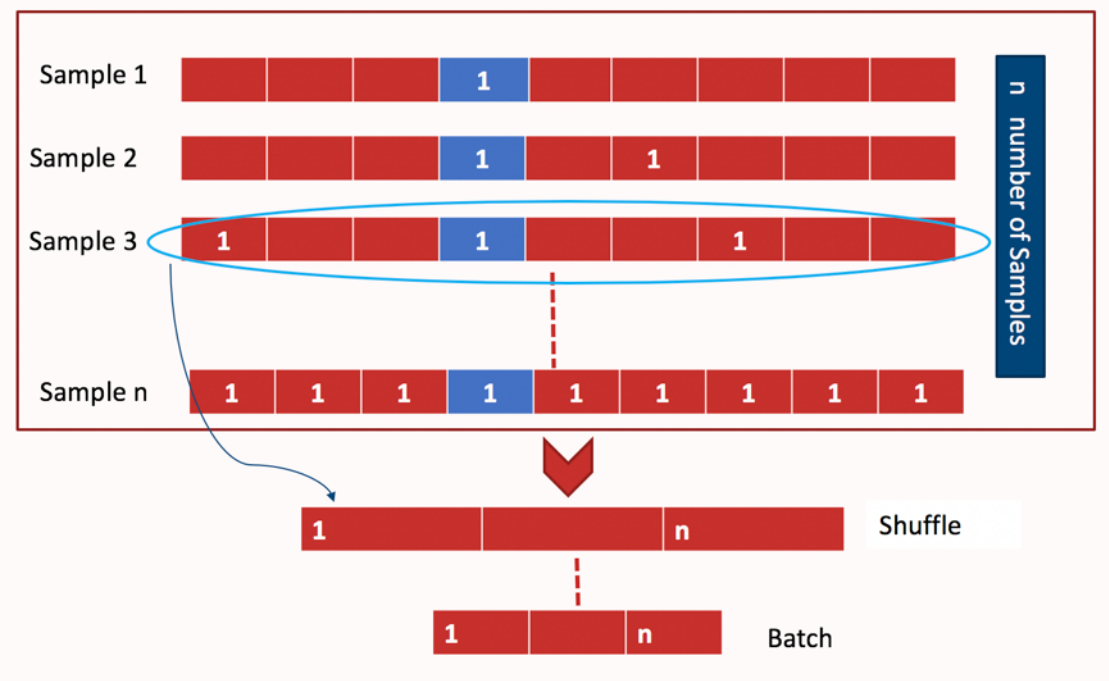


Figure 3.25: Dataset for simulation with hyperparameters

Chapter 4

Simulation and Results

To evaluate the performance of the different machine learning algorithms proposed, simulations have been carried out with different parameters and cases. Firstly, Machine Learning based simulation as well as interpolated simulation is performed on synthetic data taken from line emulator. Secondly, it is applied on data collected from real line system. LR and DNN Regression are used for estimation. Then both type of Predictions are compared using mean and standard deviation. In given dataset some channels are ON and some of them are OFF, OSNR of OFF channel is interpolated for ML based predication by using linear interpolation cubic interpolation. Then this interpolated OSNR is used as feature in ML based estimation. ML algorithms are applied and compared. Normalization is applied on dataset and Removal of Reliability factor from data set. Then the correlation between feature is found using Heat-map. ML techniques are applied on different cases with reliability and without reliability. Results shows that by adding reliability factor prediction is less bias while on the other hand reliability factor does not affect standard deviation.

4.1 Synthetic Data Results

4.1.1 Interpolation based OSNR Estimation

In figure 4.1 red points show the OSNR of ON-Channels. While OSNR of channel under test is not known so that value is interpolated from OSNR of adjacent nearest channel.

Delta = Interpolated value – Real value

In figure 4.2 values of OSNR is predicted by using interpolation but mean and S.D are very high as compare to ML-based OSNR estimation.

Simulation scenario as shown in table 4.7 where parameters are tuned, and cases are made with both linear and DNN regression models.

4.1.2 ML Models

4.1.2.1 Model-I : Linear Regression

- RMS error for the Model: 0.2903
- Max Delta = = 0.691

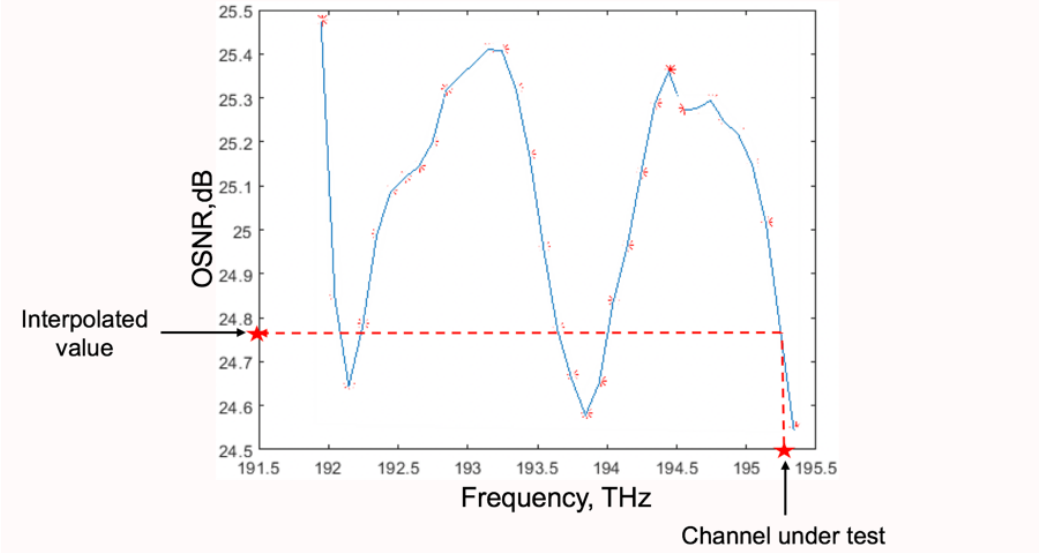


Figure 4.1: OSNR Prediction Based on Interpolation

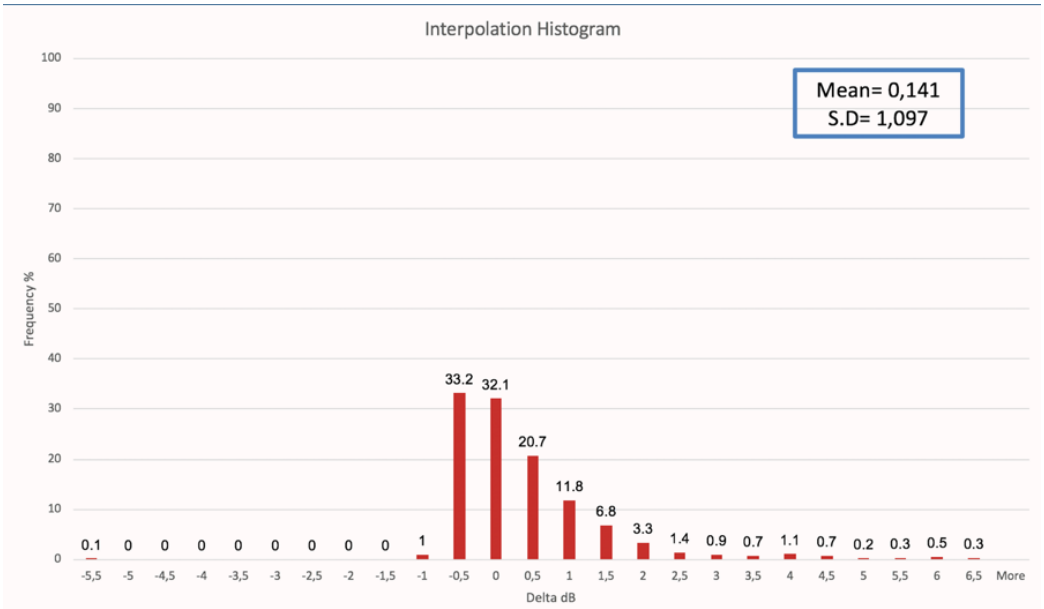


Figure 4.2: OSNR Interpolation Histogram

Hyperparameter	Model-I
Data Size	2000
Steps	5000
Shuffle	1100
Batch Size	100

Table 4.1: Hyperparameter of Model-I

- Min Delta = -0.012

The table 4.7 shows RMS comparison between different Training & Test Ratio by applying Linear Regression on 2000 samples.

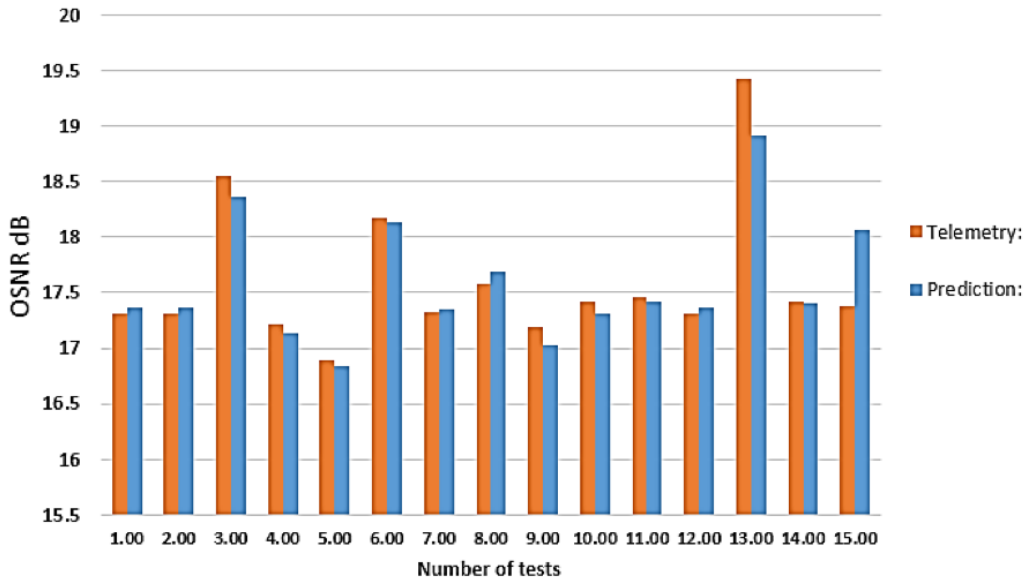


Figure 4.3: Model-I: OSNR Comparison

Steps	Samples	Training Data	Test Data	RMS
5000	2000	50%	50%	0.33
5000	2000	70%	30%	0.17
5000	2000	80%	20%	0.21
5000	2000	90%	10%	0.33

Table 4.2: RMS comparison Linear Regression

4.1.2.2 Model-II : DNN Regression

Hyperparameter	Model-I
Data Size	2000
Steps	5000
Shuffle	1100
Batch Size	100

Table 4.3: Hyperparameter of Model-II

- RMS error for the Model: 0.06
- Max Delta = 0.090
- Min Delta = 0.001

The table 4.4 shows RMS comparison between different Training & Test Ratio by applying DNN regression on 2000 samples.

Results shows that lowest RMS is found when 70% Training Ratio used along with 30% Test Ratio. So, this ratio is selected for further simulations.

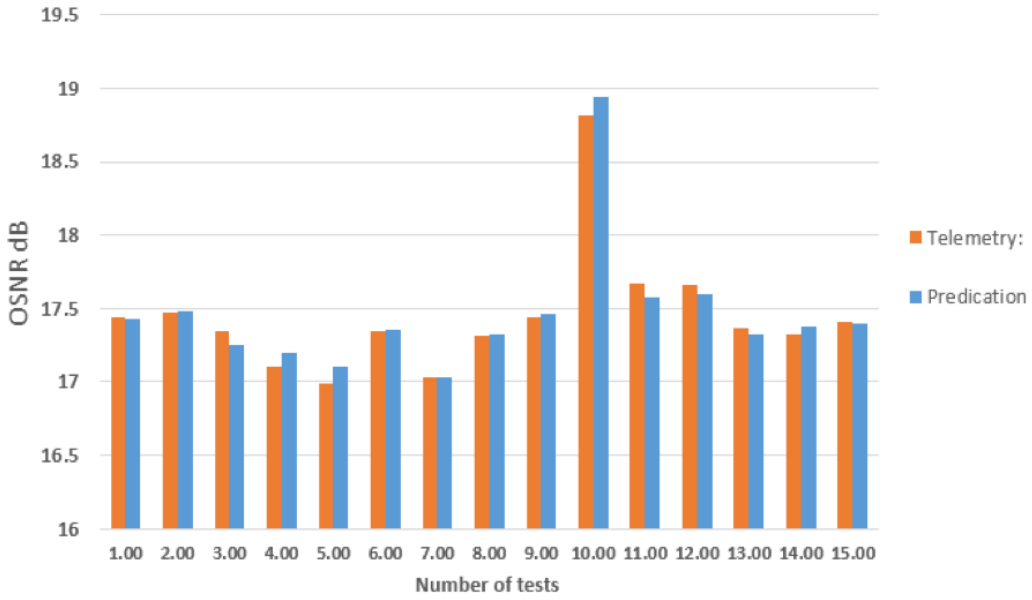


Figure 4.4: Model-II: OSNR Comparison

Steps	Samples	Training Data	Test Data	RMS
5000	2000	50%	50%	0.07
5000	2000	70%	30%	0.06
5000	2000	80%	20%	0.08
5000	2000	90%	10%	0.07

Table 4.4: RMS comparison DNN Regression

4.1.3 ML based OSNR Estimation:

To predict OSNR using machine learning, sufficient large dataset is required from which (training+test) data and prediction data chunks are extracted. Hyperparameters, Input parameters and out parameters are following for OSNR estimation.

- Input

  Training Set = 70% of the Dataset
  Test Set = 30% of the Dataset

- Hyper parameters
  Model Training Steps = 500,1000,5000,10000
  Shuffle = 1100
  Batch = 100

- Output
  OSNR Prediction

LR and DNN ML algorithms are used for estimation. And results are depicted in the form of histograms with different steps.

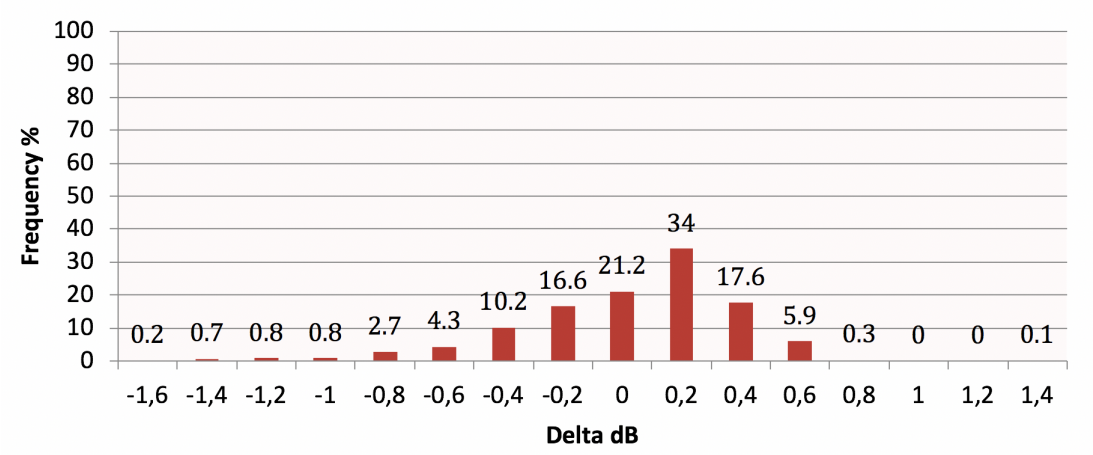


Figure 4.5: Case 1 : Linear Regression with 500 steps

4.1.3.1 Linear Regression Algorithm:

4.1.3.1.1 Case 1: 70% training 500 steps :

Prediction RMS (Delta) Histogram along with Mean = -0,077 S.D = 0,3677 which shows frequency percentage of delta in bin. For instance, 34.0% of all delta's falls between (-0.1 to 0.1).

4.1.3.1.2 Case 2: 70% training 1,000 steps :

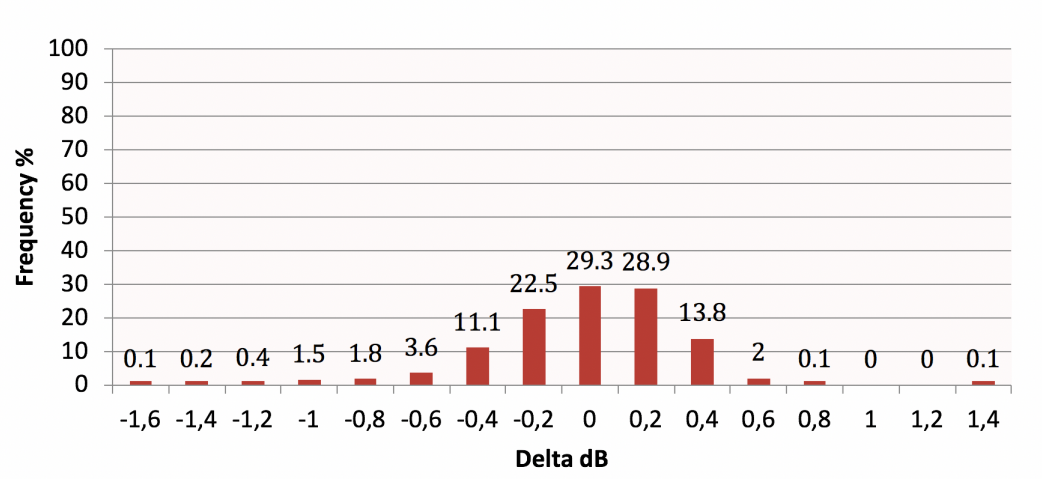


Figure 4.6: Case 2 : Linear Regression with 1000 steps

Prediction RMS (Delta) Histogram along with Mean = -0,118 S.D= 0,319 which shows frequency percentage of delta in bin. For instance, 29.3% of all delta's falls between (-0.1 to 0.1). As it can be seen that case 2 is comparatively better than case 1.

4.1.3.1.3 Case 3: 70% training 5,000 steps: :

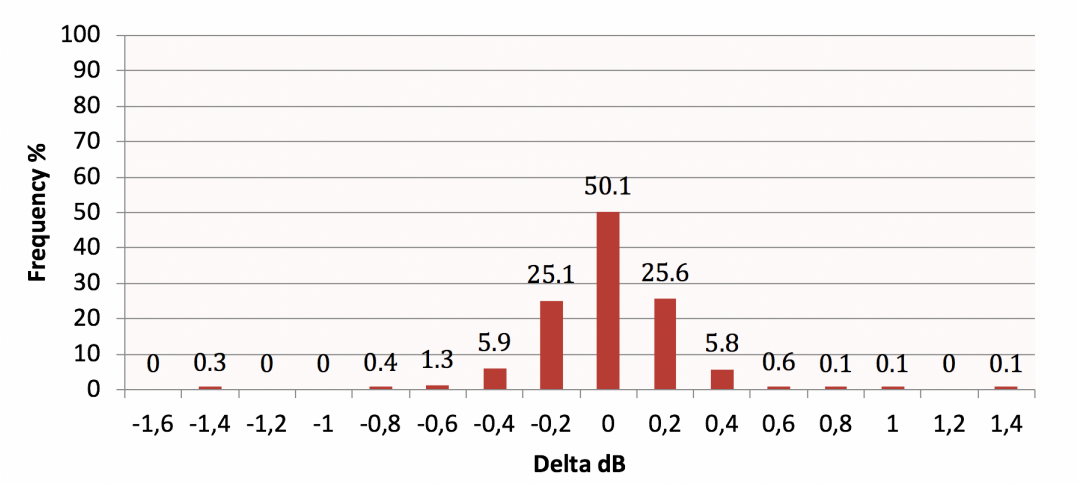


Figure 4.7: Case 3 : Linear Regression with 5,000 steps

Prediction RMS (Delta) Histogram along with Mean = -0,109 S.D= 0,214 which shows frequency percentage of delta in bin. For instance, 50.1 % of all delta's falls between (-0.1 to 0.1). which is showing that case 3 is better than case 2 and case 1.

4.1.3.1.4 Case 4: 70% training 10,000 steps :

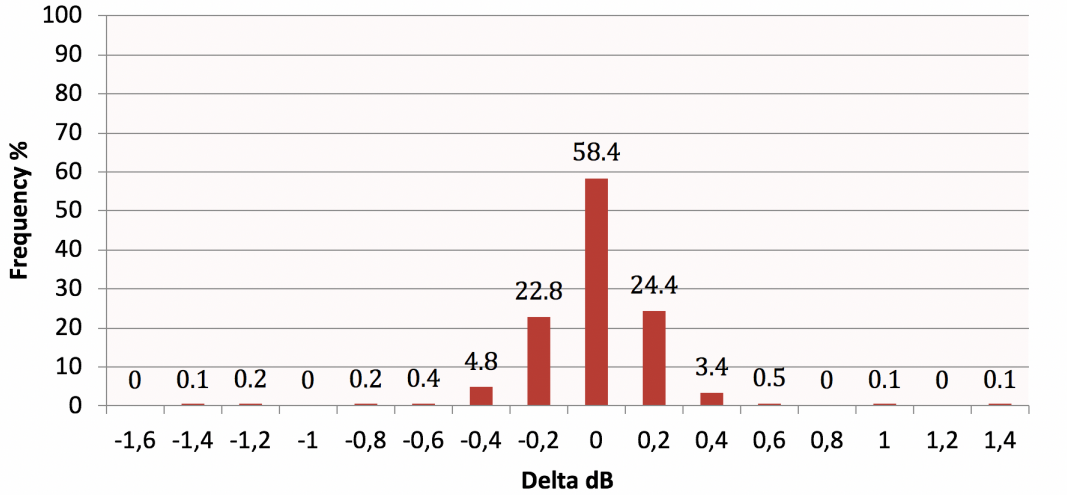


Figure 4.8: Case 4 : Linear Regression with 10,00 steps

Prediction RMS (Delta) Histogram along with Mean -0,109 S.D= 0,188 which shows frequency percentage of delta in bin. For instance, 58.4 % of all delta's falls between (-0.1 to 0.1). here it can be seen that 10000 steps are optimal for dataset when LR algorithm is used, above this threshold model starts to over trained.

4.1.3.2 DNN Regression Algorithm

4.1.3.2.1 Case 1: 70% training 500 steps :

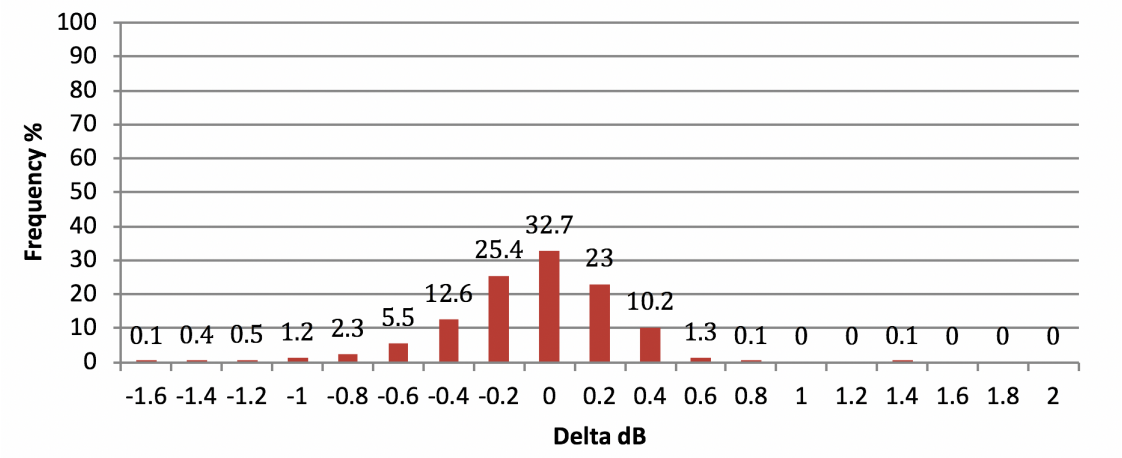


Figure 4.9: Case 1 : DNN Regression with 500 steps

Prediction RMS (Delta) Histogram along with Mean = -0,164 S.D = 0,323 which shows frequency percentage of delta in bin. For instance, 32.7% of all delta's falls between (-0.1 to 0.1).

4.1.3.2.2 Case 2: 70% training 1,000 steps :

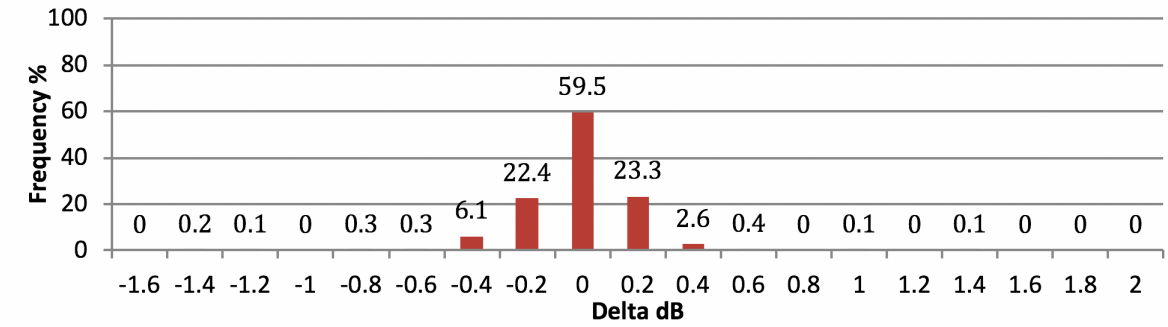


Figure 4.10: Case 2 : DNN Regression with 1,000 steps

Prediction RMS (Delta) Histogram along with Mean = -0,121 S.D = 0,189 which shows frequency percentage of delta in bin. For instance, 59.5% of all delta's falls between (-0.1 to 0.1). S.D and Mean is reduced, which is much better than the case having 500 steps.

4.1.3.2.3 Case 3: 70% training 5,000 steps :

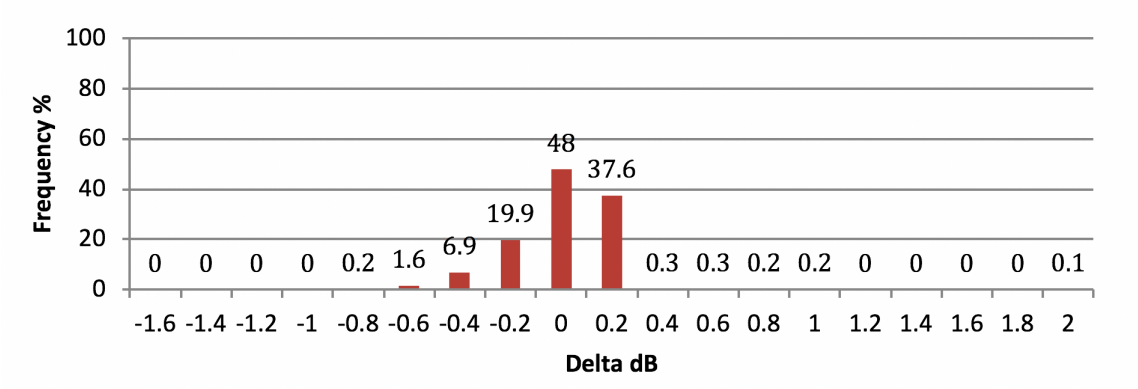


Figure 4.11: Case 3 : DNN Regression with 5,00 steps

Prediction RMS (Delta) Histogram along with Mean = = -0,103 S.D = 0,201 which shows frequency percentage of delta in bin. For instance, 48.0% of all delta's falls between (-0.1 to 0.1). S.D and Mean is reduced, which is much better than the case having 500 & 1000 steps.

4.1.3.2.4 Case 4: 70% training 10,000 steps :

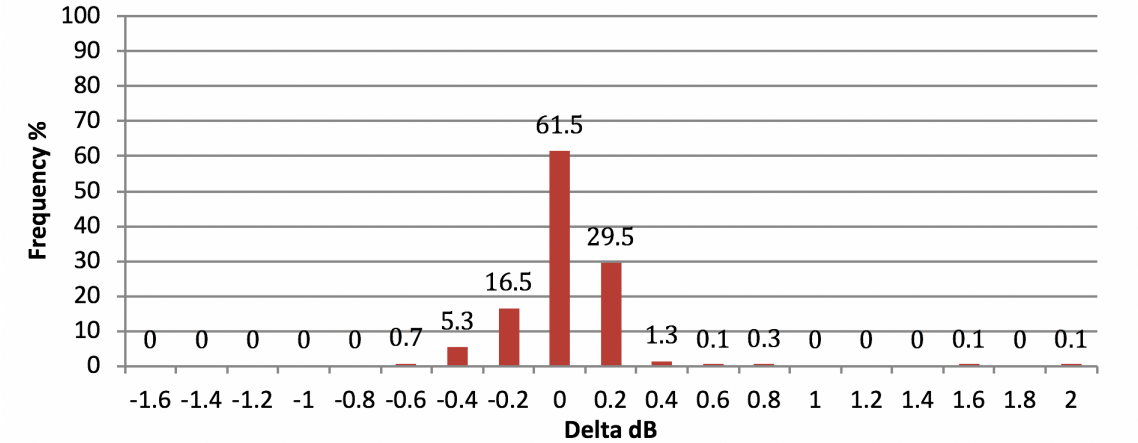


Figure 4.12: Case 4 : DNN Regression with 5,000 steps

Prediction RMS (Delta) Histogram along with Mean = -0,095 S.D = 0,178 which shows frequency percentage of delta in bin. For instance, 61.5% of all delta's falls between (-0.1 to 0.1). S.D and Mean is reduced, which is much better than the all previous cases. So like Linear Regression 70% training ratio with 10,000 steps is optimal when DNN regression is used.

4.2 Real Line System data based:

After using synthetic data for OSNR Estimation, Interpolation and ML-based estimation is applied on data extracted from experimental line system described in section 3.2. Instead of getting "0" as output power where channel is OFF, Linear and cubic interpolation is used to overcome this issue.

4.2.1 Interpolating OFF channel OSNR:

Two types of Interpolations are used to interpolate OSNR of channels whose are turned OFF.

- Linear Interpolation
- Lagrange Interpolation (Cubic)

After applying interpolation, interpolated OSNR is used as a feature in ML-Based Prediction for both Linear Regression and DNN Regression. Results of Linear Regression are depicted in figures 4.13 and 4.14. While by DNN Regression Mean and SD is showed in figures 4.15 and 4.16.

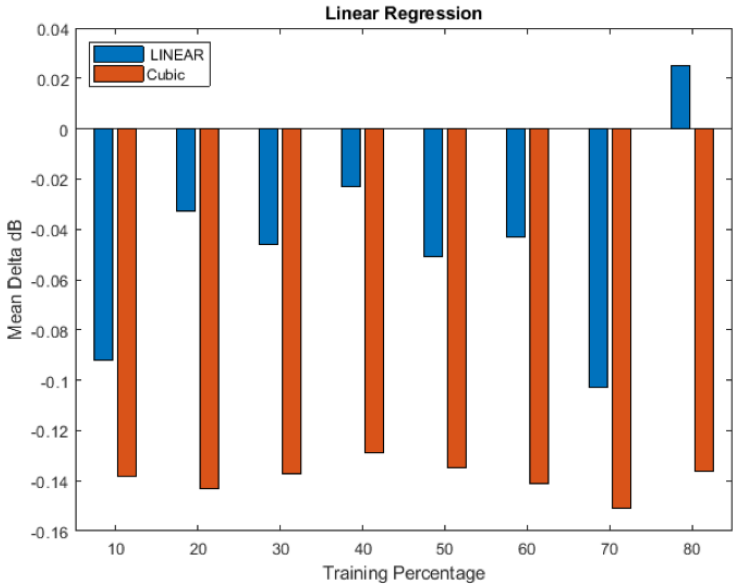


Figure 4.13: Mean by using Linear Regression

After the interpolation DNN Regression is applied and Mean and standard deviation is observed. As DNN is more intelligent than linear regression so it shows significant variation in standard deviation. Results shows that Mean calculated by Cubic Interpolation is more negative as compare to linear interpolation. While standard deviation calculated by using Cubic Interpolation is small as compare to linear interpolation. Linear Regression is the simplest type of regression algorithm therefore it shows small variation in standard deviation.

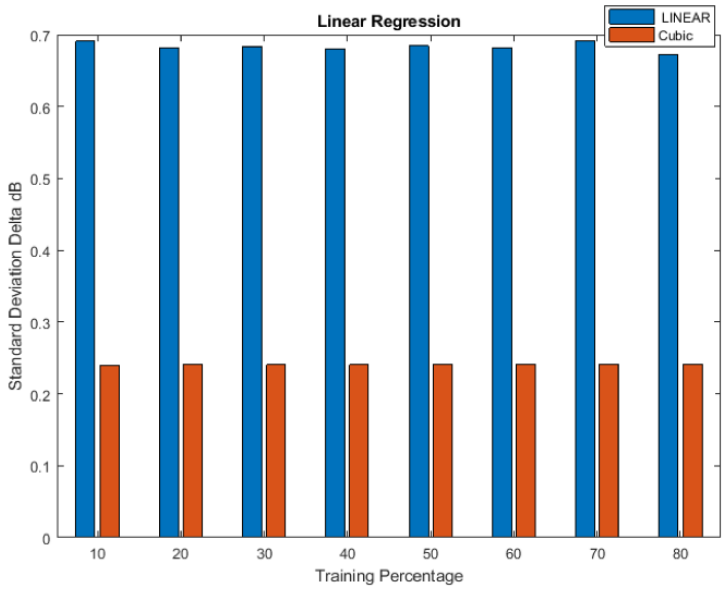


Figure 4.14: Standard Deviation by using Linear Regression

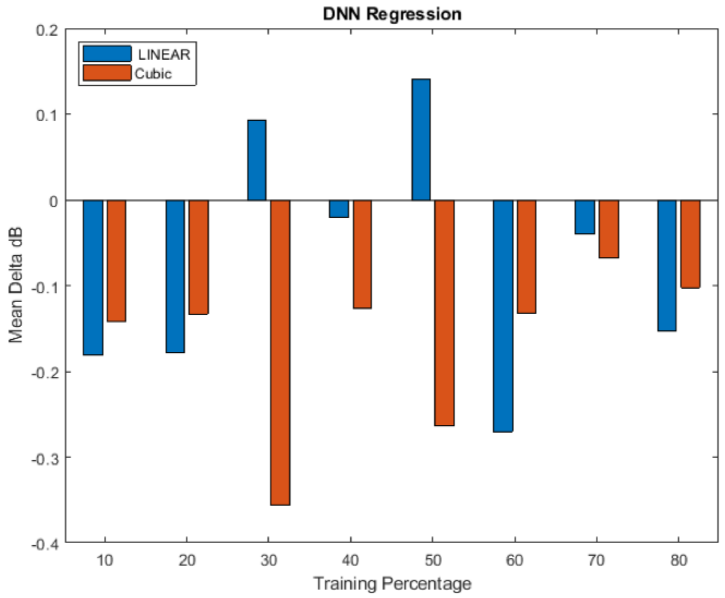


Figure 4.15: Mean by using DNN Regression

4.2.2 Impact of Normalization

Different normalization factors are applied to achieve optimal Mean and Standard deviation. Like here three different Normalization factors used are $1000, 3000, 6000$.

After normalization, Linear Regression is applied and Mean and SD is depicted in figures 4.17 & 4.18

Results shows that, higher the Reliability Normalization factor lower will be the bias. In dataset total channels are 35 and each of them has band width of 100 giga. So, in total band width is 3500 so normalization factors used here is 3500.

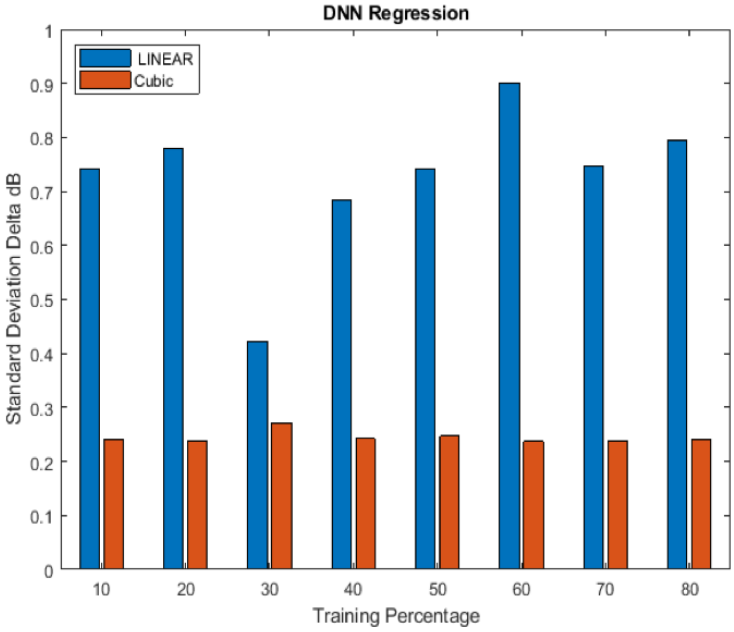


Figure 4.16: Standard Deviation by using DNN Regression

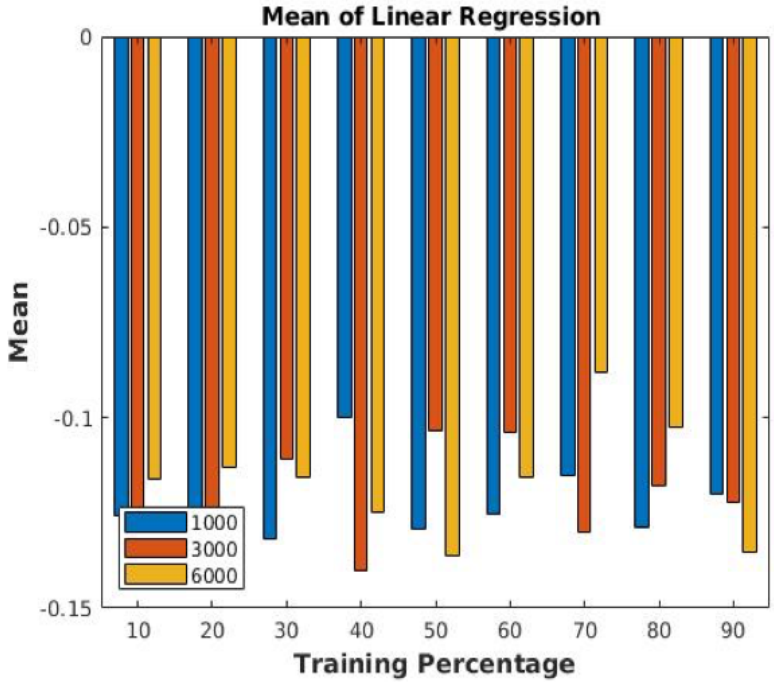


Figure 4.17: Mean after Normalization and Linear Regression

4.2.3 Impact of Reliability

4.2.3.0.1 Calculation of Reliability Factor : When channel is ON reliability is 1 but when channel is OFF reliability is calculated by following formula.

$$1 - \frac{change\ in\ frequency\ of\ adjacent\ channel}{Normalization\ factor}$$

4.2.3.0.2 Comparison based on Reliability factor :

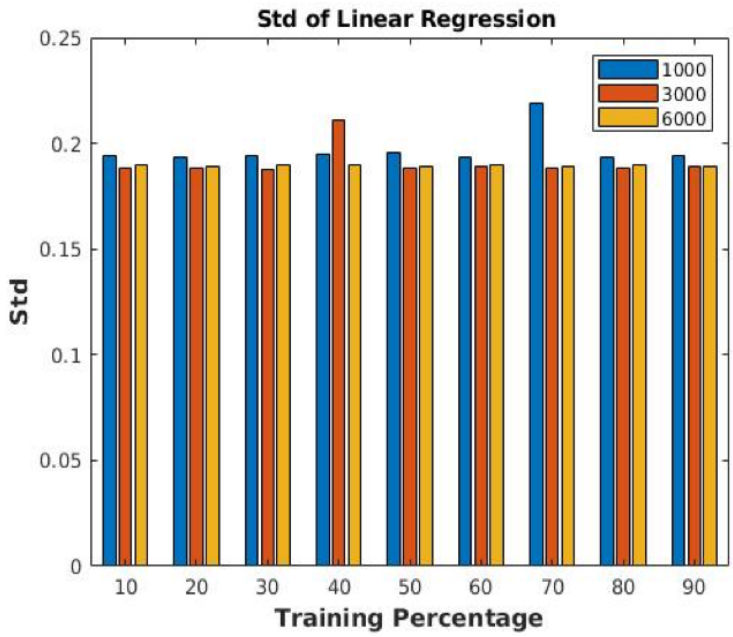


Figure 4.18: Standard Deviation after Normalization and Linear Regression

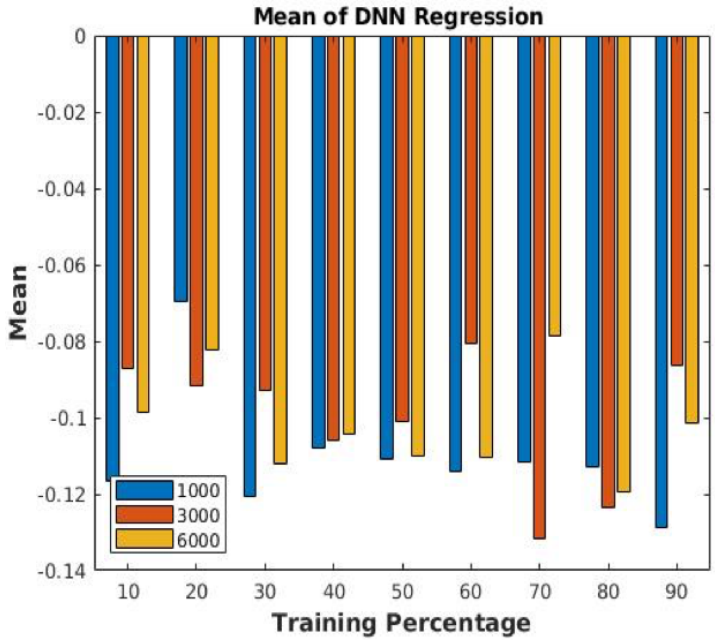


Figure 4.19: Mean after Normalization and DNN Regression

- Case 1:
  Prediction of OSNR by using ML techniques where reliability factor is considered.

- Case 2:
  Prediction of OSNR by using ML techniques where reliability factor is not considered.

In figure 4.21 & 4.22 it can be seen that, by adding reliability factor decrease the bias in the QoT-Estimation.

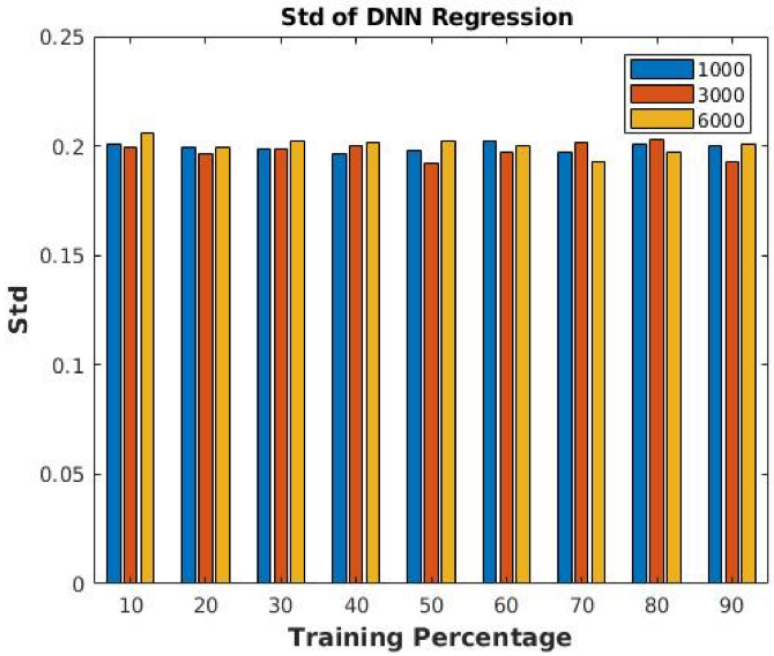


Figure 4.20: Standard Deviation after Normalization and DNN Regression

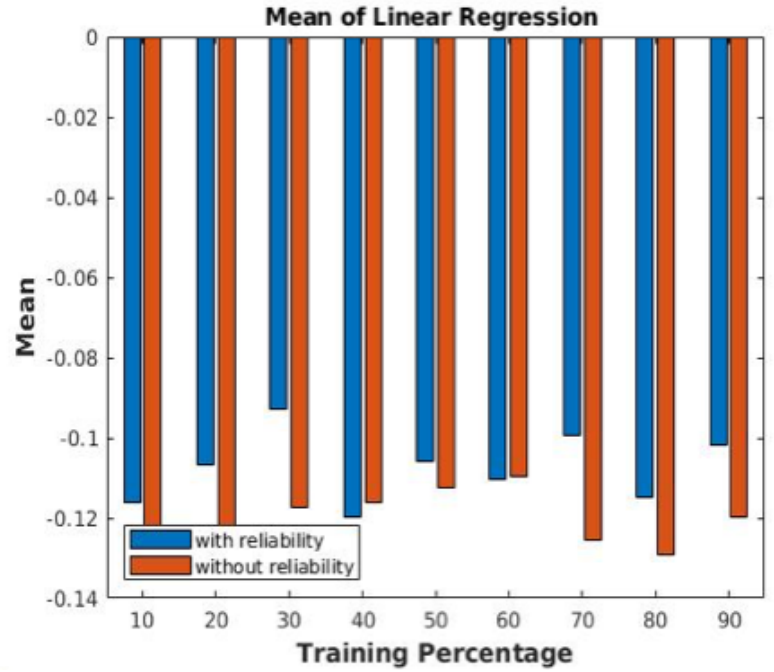


Figure 4.21: Mean after Linear Regression With & without Reliability factor

While on the other hand, In figure 4.23 & 4.24 it can be seen that by adding reliability factor has no significant effects on the standard deviation of the prediction.

4.2.4 Correlation Matrix with heat-map:

Correlation states how the features are related to each other or the target variable. There are two types of Correlation.

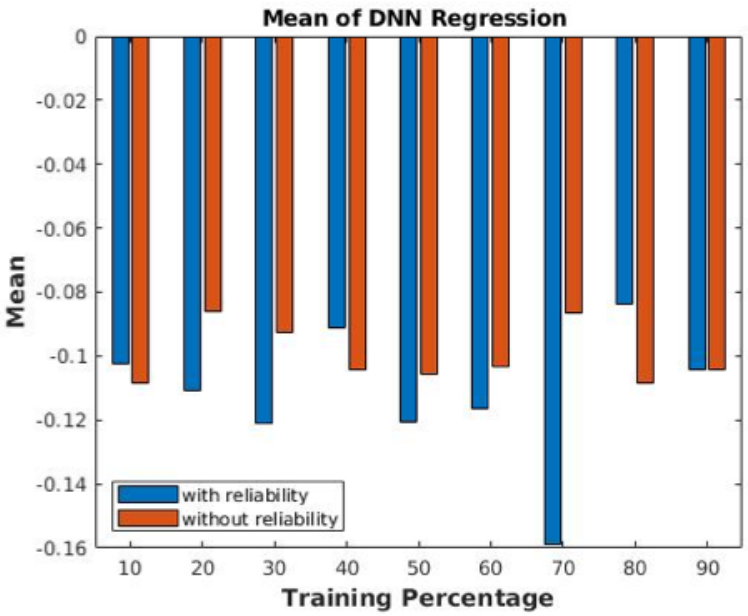


Figure 4.22: Mean after DNN Regression With & without Reliability factor

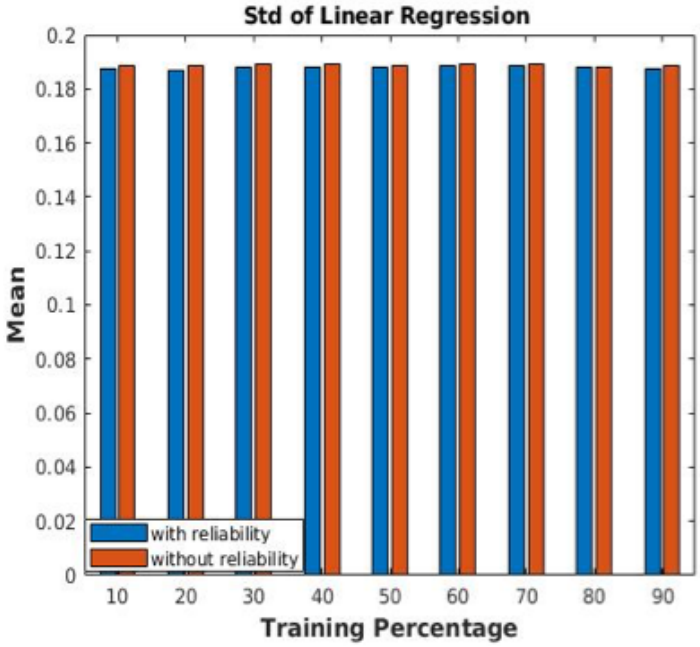


Figure 4.23: Standard Deviation after Linear Regression With & without Reliability factor

- Positive
- Negative

Positive correlation states that if there is an increase in one value of feature, it will increase the value of the target variable. While negative states that if there is an increase in one value of feature as consequences there will be decrease in the value of target variable. Figure 4.25 shows the correlation between features by using heat-map, which makes it easy to identify which features are most related to the

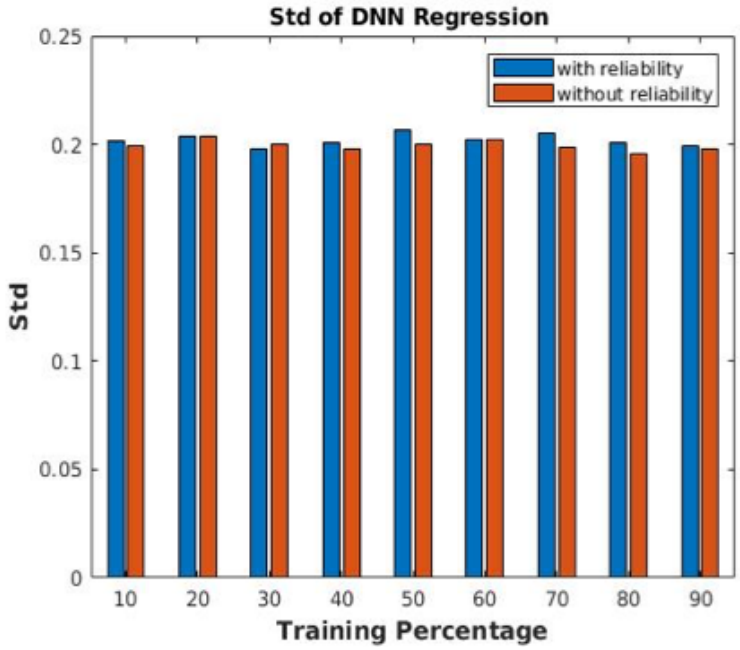


Figure 4.24: Standard Deviation after DNN Regression With & without Reliability factor

target variable.

In figure 4.26 correlation between reliability and OSNR is depicted. Where strongly correlated vales are goes towards (+0.8) on $y - axis$. And those values are represented in green.

For instance, figures 4.26 shows that, reliability_01 is strongly correlated with osnr_01 because then value is +0.36 while on the other hand reliability_01 is not correlated with osnr_02 because its value is (-0.93) represented in red color.

4.3 QoT-Estimation

At this point real data is collected from experimental line system and pre-processed by normalization with optimal normalization factor. In order to fill empty spaces in dataset, OSNR of OFF channels is interpolated. As it is known that OPM provides the output power of 100 GHz per channel only. So, OSNR can not be measured directly. Therefore, dataset is modified to estimate QoT.

In modified dataset OSNR is replaced with out-power of every channel. If input channel is OFF **Pin** is zero and **Pout** is equal to the Power of ASE. Similarly, if input channel is ON, then **Pin** is channel input power and **Pout** is received output power of channel.

4.3.1 Prediction of Pout

Three types of predications are made with dataset depicted in 4.28

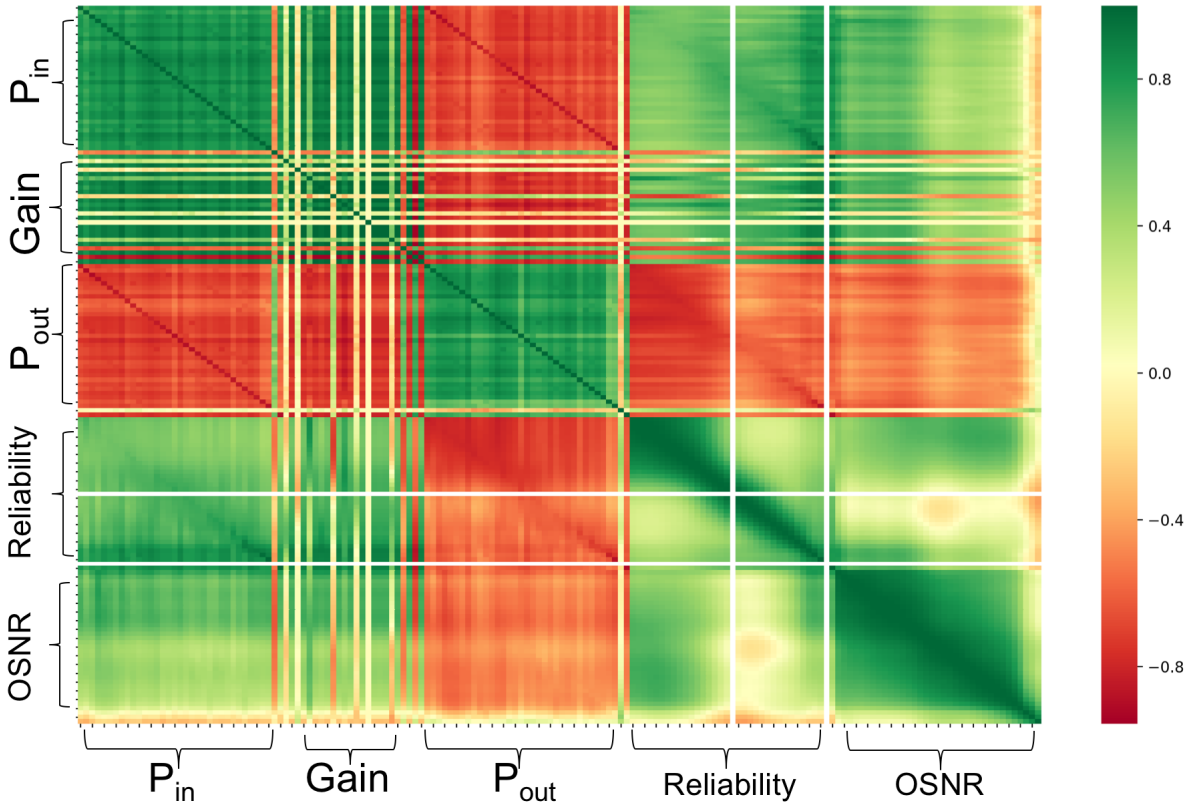


Figure 4.25: Correlation between all features using Heat-Map

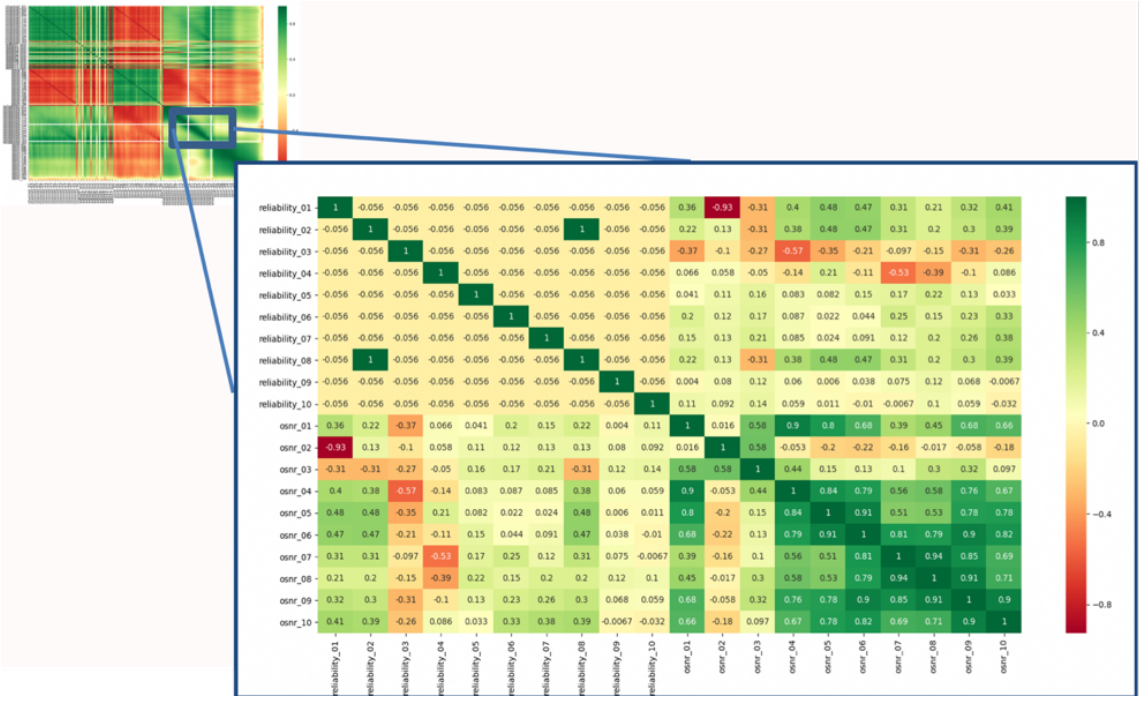


Figure 4.26: Correlation between Reliability and OSNR using Heat-Map

4.3.1.1 Fixing single label

In this section ML model is set to predict only single label. In the figure 4.29 Pout34 is predicted by using Linear and DNN Regression.

Figure 4.27: Previous Dataset

Figure 4.28: New Dataset

4.3.1.2 Fixing 35 labels

In this section ML model is predicting set of labels and prediction results of (Pout34) are depicted in 4.30 .

Pout of all the 35 channels is predicted while in the figure 4.30 Pout34 is depicted only in order to compare the differences by predicting only only one label or set of labels. Is can be seen that in both cases ML algorithms performed very well. There is no significant difference by using Linear and DNN Regression. But here difference is that, By using Linear Regression the difference between SD is (0.588-0.510)=0.078 so its doesn't make significant difference while predicting single label and predicting set of labels.

While on the other hand, DNN Regression shows significant difference between SD (1.572-0.707)=0.865. As it is known that DNN works better when complexity increases.

4.3.2 Approach-I

4.3.2.1 Dataset Preparation

Dataset illustrated in 4.28 can predicted Pout only so in order to estimate OSNR dataset is modified and illustrated in 4.31 where Transfer Function and P_{ASE} are added to previous dataset. The transfer function is basically the ratio between

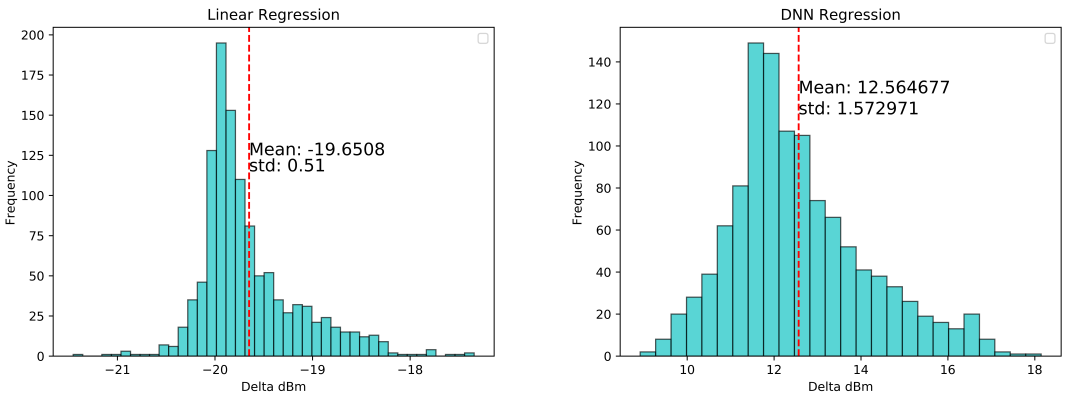


Figure 4.29: Fixing single label (pout34)

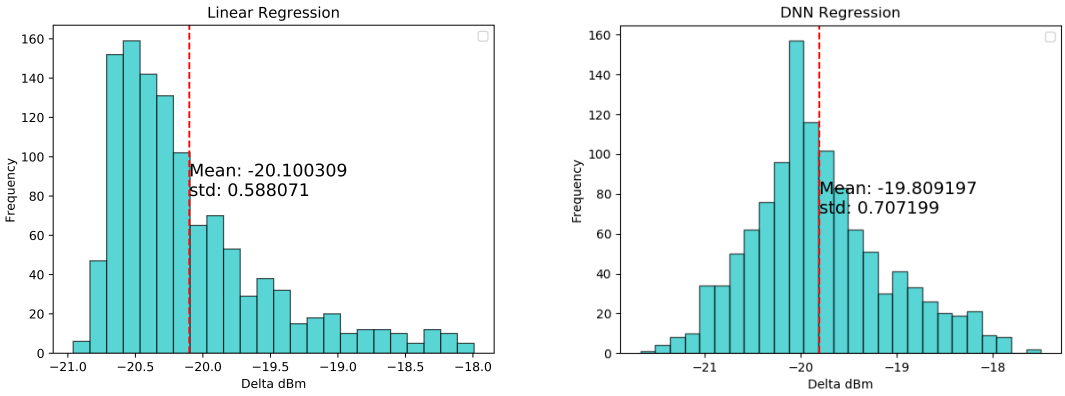


Figure 4.30: Fixing 35 labels

output power and ASE.

$$Transferfunction = P_{out}/P_{in} \tag{4.1}$$

To prepare this dataset two approaches are used as following

Figure 4.31: Modified Dataset

- Normalization
- Cubic Interpolation

4.3.2.1.1 Normalization

- Create an array of 35 elements having tf and ASE. Put tf for on channels and ASE for off channels
- Separate the array into two arrays one is tf array leave off channel empty in this array and other is ASE array leave the on channel empty.
- After getting Transfer function array and ASE noise array we normalize tf array by the mean of transfer function and ASE noise array mean of ASE
- Merge the two arrays togethers to get combined normalized array
- Tf array is obtained by multiplying the normalized combined array with the mean of transfer function
- Tf array is obtained by multiplying the normalized combined array with the mean of transfer function
- ASE array is obtained by multiplying the normalized combined array with the mean of ASE

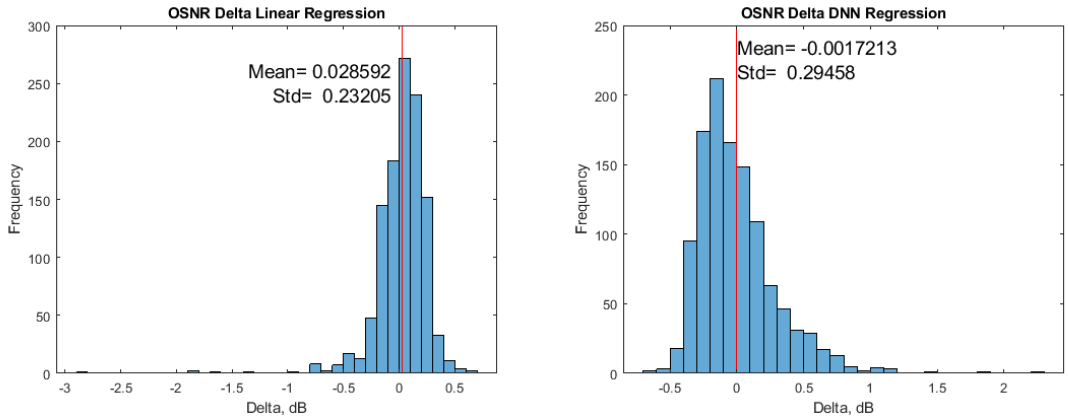


Figure 4.32: OSNR by using Normalization

4.3.2.1.2 Cubic Interpolation

- Create an array of transfer functions, for on channels we get tf from the ratio of pout and pin. For the off channels we get it by using cubic interpolation.
- Create an array of ASE, for off channels we get ASE power from OPM. For the on channels we get it by using cubic interpolation.

4.3.2.2 Prediction Types

With the above data-set, two types of predication are made.

4.3.2.2.1 Training with OFF-Channel 34 During training of model label (channel 34) is OFF and in prediction channel 34 is ON.

4.3.2.2.1.1 Method-I Normalization

- Linear
- DNN

Normalization	Linear Regression		DNN Regression	
	Mean	**S.D**	**Mean**	**S.D**
Pout	0.01977	0.5591	0.00432	0.5105
ASE	-0.00882	0.45156	0.00603	0.73084
OSNR	0.02859	0.23205	-0.001721	0.29458

Table 4.5: Approach 1 (Normalization)

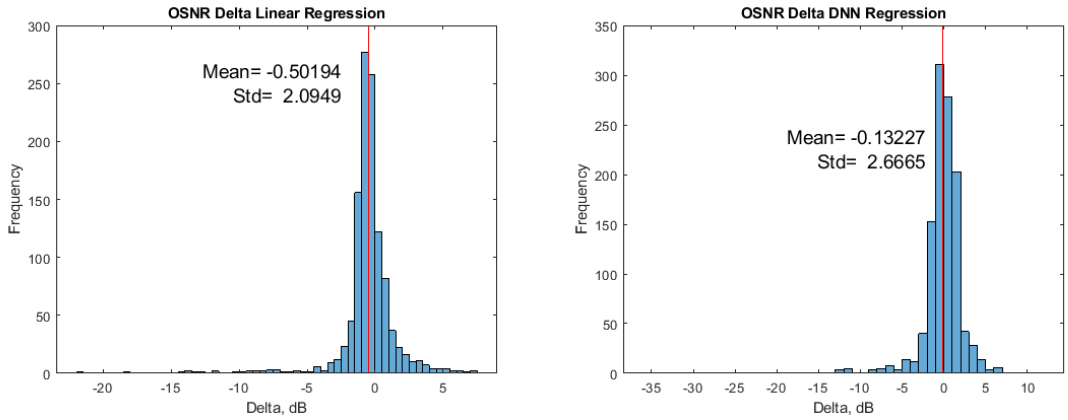


Figure 4.33: OSNR by using Interpolation

Interpolation	Linear Regression		DNN Regression	
	Mean	**S.D**	**Mean**	**S.D**
Pout	-0.00238	0.35	-0.028136	0.65668
ASE	- 0.49955	2.0116	0.10413	2.3811
OSNR	-0.50194	2.0949	-0.13227	2.6665

Table 4.6: Approach 2 (Cubic Interpolation)

4.3.2.2.1.2 Method-I Cubic Interpolation

- Linear
- DNN

Form results of figure 4.7 it is observed that, Normalization is much better than Interpolation. Because in every case standard deviation & Mean values by using 1st approach is less than 2nd approach. For instance in figure 4.33 S.D of predicted OSNR is much higher than S.D of OSNR illustrated in 4.32

4.3.2.2.2 Training with mixed ON & OFF-Channel 34 Model is trained with mixed data where 34 is ON & OFF. Half of the data is randomly chosen when the channel under-test is ON and mixed with data when the channel under test is OFF. then Normalization is applied as it performed much better than Interpolation.

4.3.2.2.2.1 Method-I Normalization

- Linear
- DNN

Changes are made in previous dataset to incorporate the calculation of OSNR. As its already described that the OPM only provides the output power and is not dealing with any kind of OSNR measurements. To calculate the OSNR it is necessary to have noise power along with signal power. Therefore, new dataset is introduced

Normalization	Linear Regression		DNN Regression	
	Mean	**S.D**	**Mean**	**S.D**
Pout	-0.015713	0.35667	0.0039115	0.35829
ASE	- 0.010758	0.30987	0.13081	0.63013
OSNR	-0.0049582	0.20734	-0.1269	0.41408

Table 4.7: OSNR Mixed ON & OFF Data

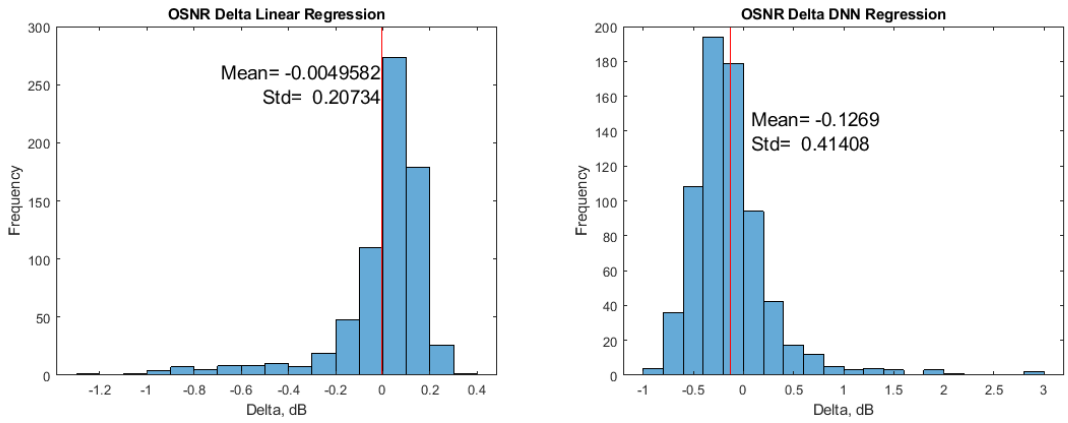


Figure 4.34: OSNR Mixed ON & OFF Data

having ASE and Transfer Function (T_f). The Transfer Function is basically the ratio between output power and ASE. Along with this, An approach is selected which selects randomly half of the data when the channel under-test is on, mixed with data when the channel under test is OFF and starts training the machine learning with dataset depicted in figure 4.31.

4.3.3 Approach-II

4.3.3.1 Dataset Preparation

To prepare the dataset, two approaches are used.

4.3.3.1.1 Separate Normalized Tf and ASE Normalization and separate array of T_f and ASE is created by following,

- Create an array of 35 elements having tf and ASE. Put tf for on channels and ASE for off channels
- Separate the array into two arrays one is tf array leave off channel empty in this array and other is ASE array leave the on-channel empty.
- After getting Transfer function array and ASE noise array we normalize tf array by the mean of transfer function and ASE noise array mean of ASE
- Merge the two arrays togethers to get combined normalized array
- Tf array is obtained by multiplying the normalized combined array with the mean of transfer function

- Tf array is obtained by multiplying the normalized combined array with the mean of transfer function
- ASE array is obtained by multiplying the normalized combined array with the mean of ASE
- Here both tf and ASE are used as a separate array to train machine learning

4.3.3.1.2 Single Normalized Array Normalization to get single normalized array by following,

- Create an array of 35 elements having (T_f) and ASE. Put (T_f) for on channels and ASE for off channels
- Separate the array into two arrays one is (T_f) array leave off channel empty in this array and other is ASE array leave the on-channel empty.
- After getting Transfer function array and ASE noise array we normalize (T_f) array by the mean of transfer function and ASE noise array mean of ASE
- Merge the two arrays togethers to get combined normalized array
- A single normalized array in used this case not (T_f) and ASE to train machine learning

4.3.3.2 Prediction Types

With the above dataset two types of predication are made.

4.3.3.2.1 Training with mixed ON & OFFChannel 34 Half of the data has been chosen randomly when the channel under test is ON and rest half is chosen when the channel under test is OFF, mixed and TF and ASE are predicted.

4.3.3.2.1.1 Predict TF and ASE

- Linear
- DNN

4.3.3.2.2 Training with mixed ON & OFFChannel 34 Half of the data has been chosen when the channel under test is ON and rest half is chosen when the channel under test is OFF, mixed and T_f and ASE are predicted. Instead of choosing randomly, half of the data from every case when the channel under test is ON is selected and mixed with data when the channel under test is OFF T_f and ASE are predicted.

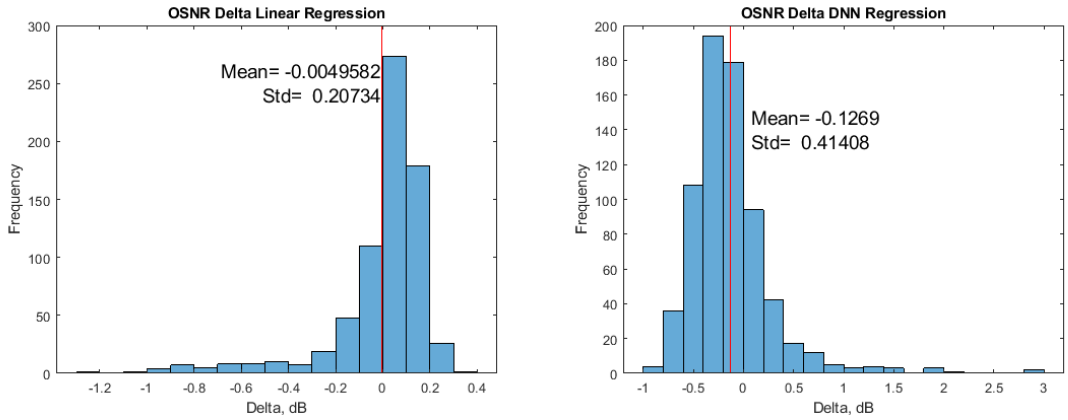


Figure 4.35: Approach-II OSNR Mixed ON & OFF Data

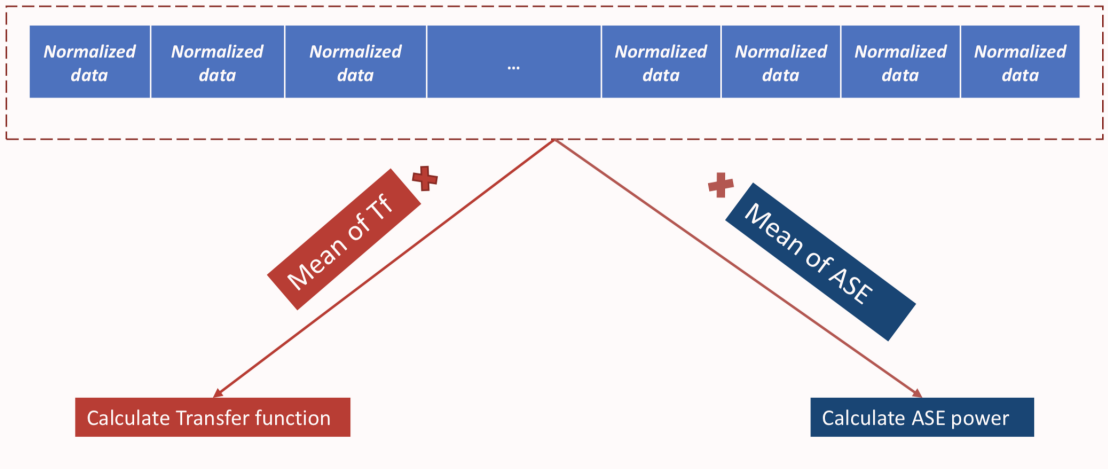


Figure 4.36: Normalized Array

4.3.3.2.2.1 Predict TF and ASE

- Linear
- DNN
- Predict single normalized array

Single normalized array is predicted depicted in figure 4.36 using machine learning. After prediction, got the T_f array by multiplying the normalized predicted array with the mean of transfer function. Similarly, ASE array is obtained by multiplying the normalized predicted array with the mean of ASE.

$$OSNR = \frac{Channel P_{in} \ X \ Transfer \ Function \ (T_f)}{ASE}$$

$$delta = OSNR_{predicted} - OSNR_{telemetry}$$

It is observed that, In figure 4.37 SD. is 0.28336 for DNN Regression while in figure 4.35 SD. is 0.41408 when data is mixed randomly. It can be clearly seen that

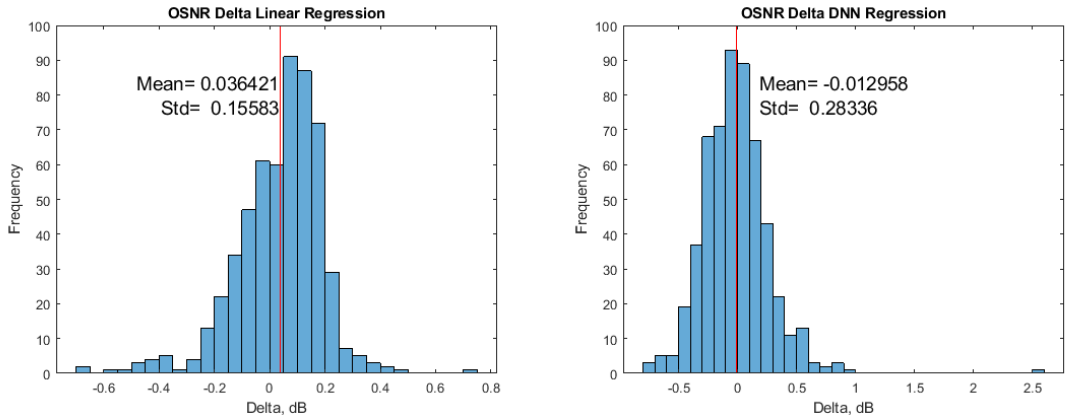


Figure 4.37: OSNR - Mixed Every Combinations

prediction of OSNR is much better in figure 4.37 when half of the data from every case when the channel under test is ON is selected. so this approach is selected for further simulation.

4.3.4 Approach-III

4.3.4.1 Dataset Preparation

To prepare the data-set two approaches are used.

4.3.4.1.1 Separate Normalized T_f and ASE Normalization and separate array of T_f and ASE is created by following,

- Create an array of 35 elements having T_f and ASE. Put T_f for on channels and ASE for off channels
- Separate the array into two arrays one is T_f array leave off channel empty in this array and other is ASE array leave the on-channel empty.
- After getting Transfer function array and ASE noise array we normalize T_f array by the mean of transfer function and ASE noise array mean of ASE
- Merge the two arrays togethers to get combined normalized array
- T_f array is obtained by multiplying the normalized combined array with the mean of transfer function
- ASE array is obtained by multiplying the normalized combined array with the mean of ASE
- Both T_f and ASE are used as a separate array to train machine learning.

4.3.4.1.2 Single Normalized Array To get single normalized array, procedure is following.

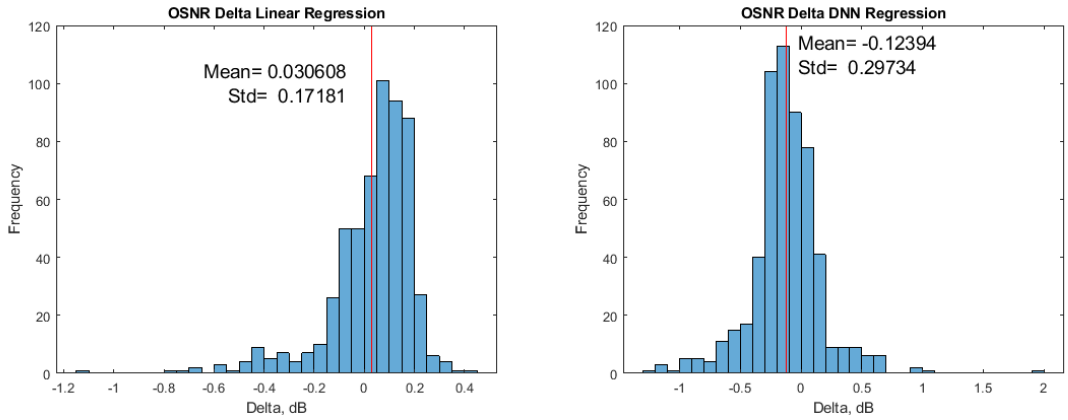


Figure 4.38: OSNR - Prediction of T_f & ASE

- Create an array of 35 elements having T_f and ASE. Put T_f for on channels and ASE for off channels
- Separate the array into two arrays one is T_f array leave off channel empty in this array and other is ASE array leave the on-channel empty.
- After getting Transfer function array and ASE noise array we normalize T_f array by the mean of transfer function and ASE noise array mean of ASE
- Merge the two arrays togethers to get combined normalized array
- T_f array is obtained by multiplying the normalized combined array with the mean of transfer function
- T_f array is used along with the normalized array instead of ASE array to train machine learning.

4.3.4.2 Prediction Types

With the above data-set, two types of predictions are made.

4.3.4.2.1 Training with mixed ON & OFF-Channel 34 During the training of model, channel 34 mixed ON and OFF while in prediction channel 34 ON. Half of the data has been chosen when the channel under test is ON and rest half is chosen when the channel under test is OFF, mixed and Tf and ASE are predicted. Instead of choosing randomly, half of the data from every case when the channel under test is ON is selected and mixed with data when the channel under test is OFF. Then Tf and ASE are predicted.

4.3.4.2.1.1 Predict TF and ASE

4.3.4.2.1.2 Predict Tf and single normalized array

$$OSNR = \frac{Channel\ P_{in}\ X\ Transfer\ Function\ (T_f)}{Mean\ of\ ASE\ X\ Normalized\ value}$$

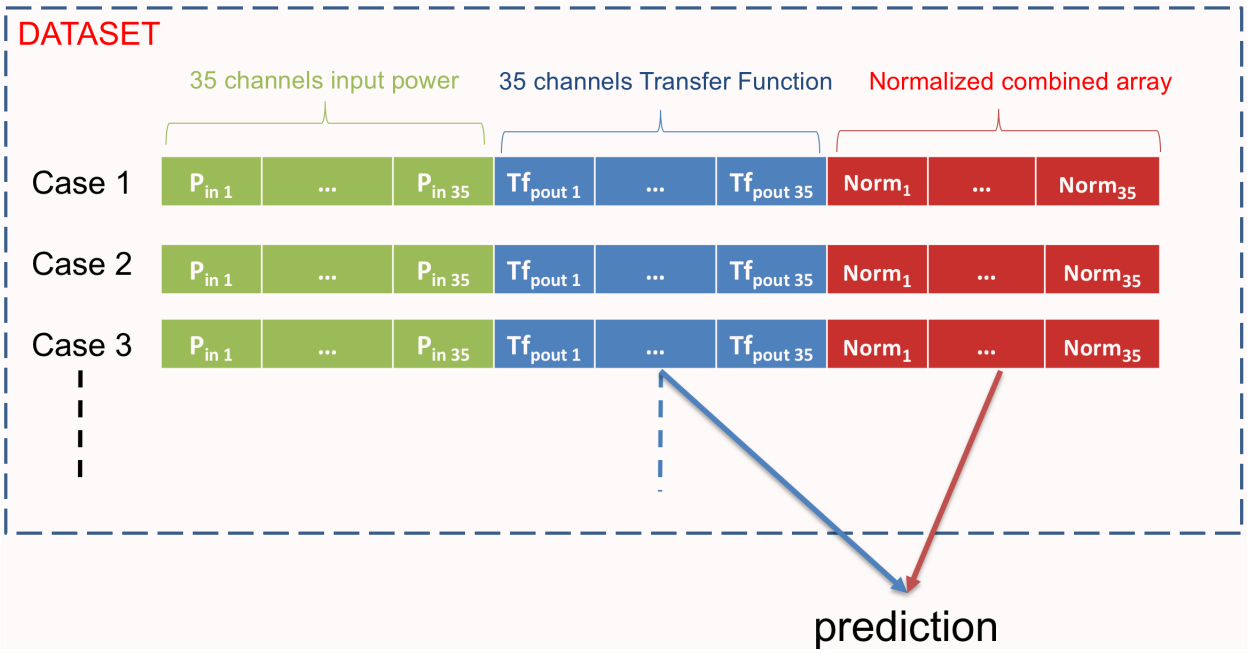


Figure 4.39: Prediction of Normalized Array, T_f & ASE

$$delta = OSNR_{predicted} - OSNR_{telemetry}$$

On base of standard deviation it is observed that prediction of Normalized Array & T_f is the best prediction type and will be used for further experiments.

Form now on selected approach is to **Predict Tf and single normalized array using DNN** and will be used for further experiments.

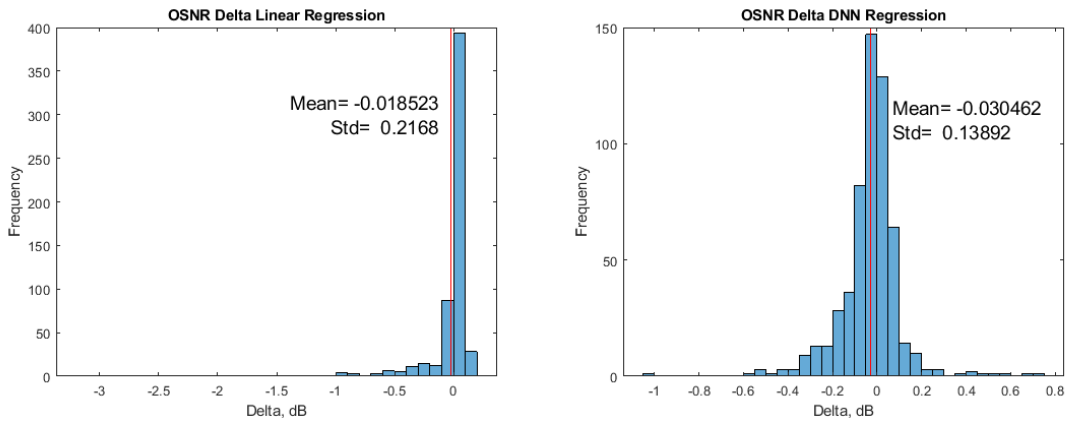
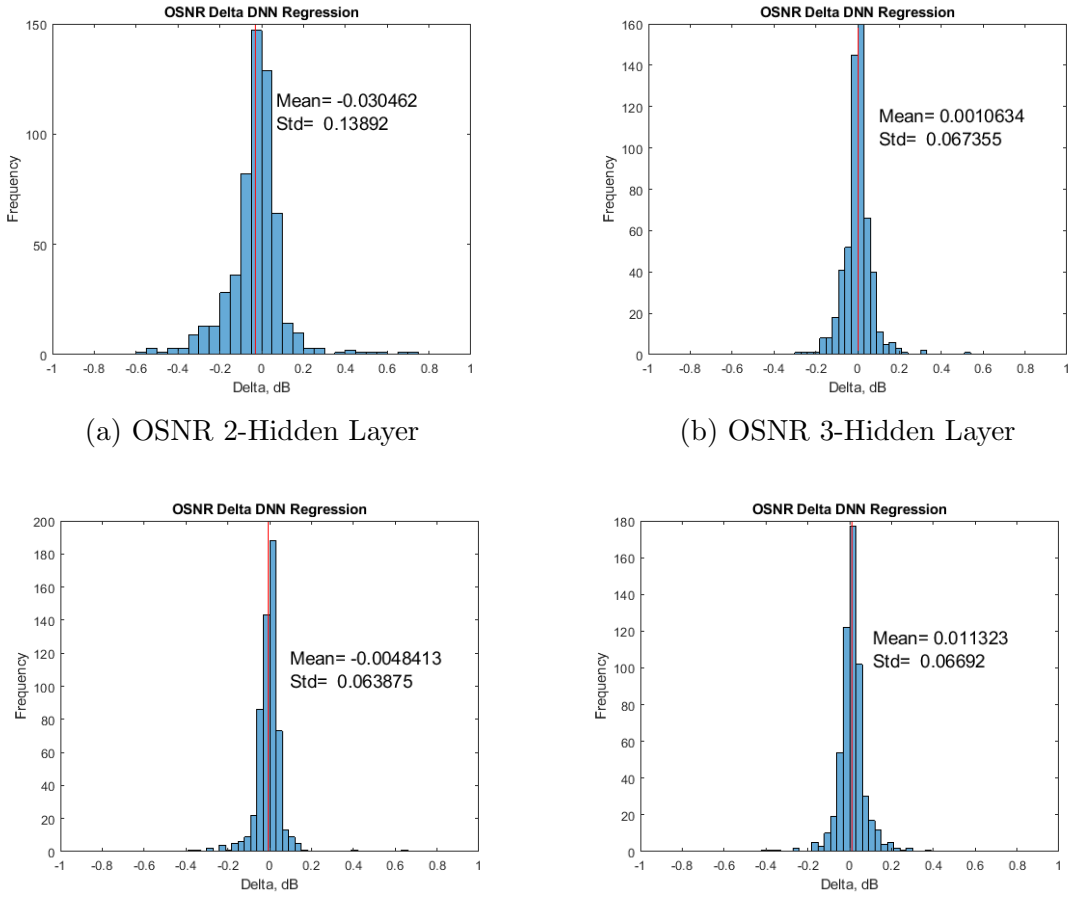


Figure 4.40: OSNR - Predict T_f and Single Normalized Array

4.3.5 Hidden Layers

After adding hidden layers significant improvements in prediction of OSNR is observed figure 4.42 shows that standard deviation decreases with increase of hidden layer. in figure 4.41 it can be seen that with two hidden layers SD is "0.13892" and with 5 hidden layer SD is "0.06692"



(a) OSNR 2-Hidden Layer

(b) OSNR 3-Hidden Layer

(c) OSNR 4-Hidden Layer

(d) OSNR 5-Hidden Layer

Figure 4.41: Comparison between different Hidden-Layers

As shown in figure 4.42 standard deviation decrease with the increase of number of hidden layer but after the specific number of layer the difference is negligible. It can b observed that in figure 4.42 after 3rd hidden layer difference between standard deviation of 3rd and 4th layer is 0.003. Similarly for 4th and 5th layer. So its better to use three hidden layers with DNN Regression.

4.3.6 Data Analysis

It is observed that, DNN Regression works much better with three hidden layers. The distribution of OSNR is given in figure 4.43. On the basis of previous distribution of OSNR depicted in figure 4.43. Difference between predicted and telemetry $OSNR = (Std = 0.067355)$ is selected and illustrated histogram in figure 4.44.

On the basis of the distribution depicted in figure 4.44 there are maximum 2/3 of the the cases where delta is maximum and only 1/2 cases where delta is minimum

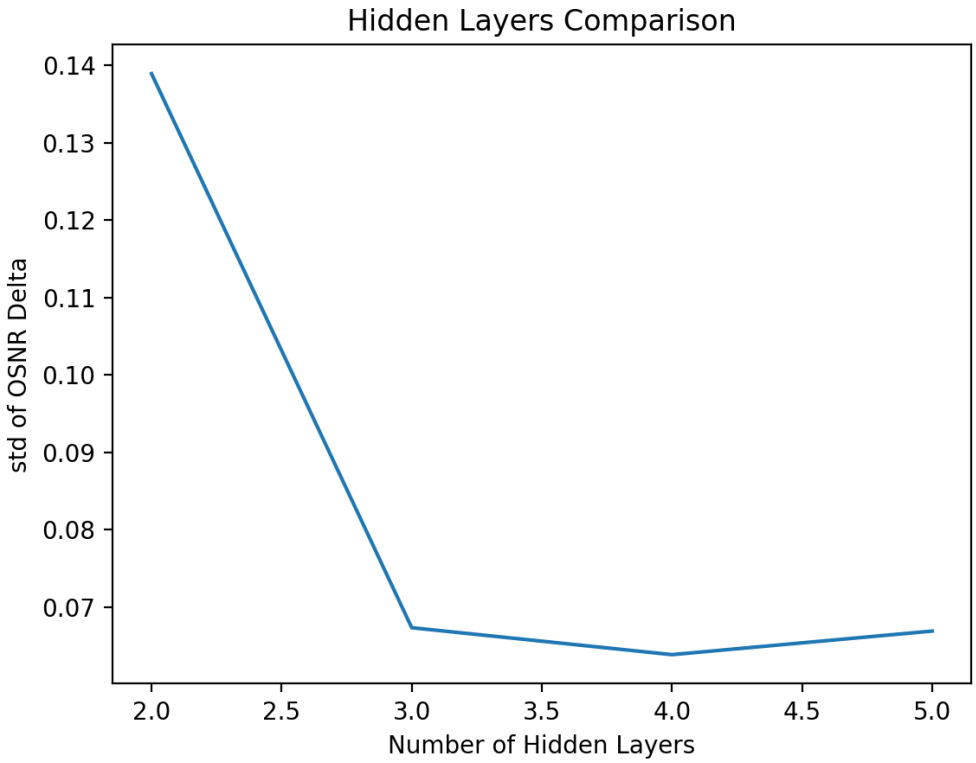


Figure 4.42: Effect of Hidden Layers

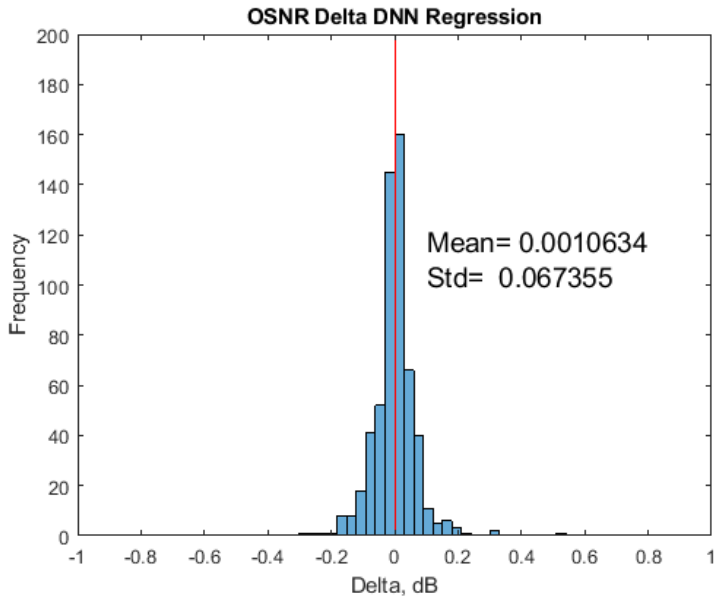


Figure 4.43: OSNR Delta DNN Histogram with 3-Hidden Layers

(0.067355) figure 4.45. Previously, half of data has been chosen when channel under test was ON. Instead of fixing data selection, flexibility in data selection is introduced on base of figure 4.45. For instance, to predict OSNR of case number (30 to 35) 50% of data is chosen (from every case where channel under test is ON) for training and rest for prediction and similarly 66% for case number (3 & 5).

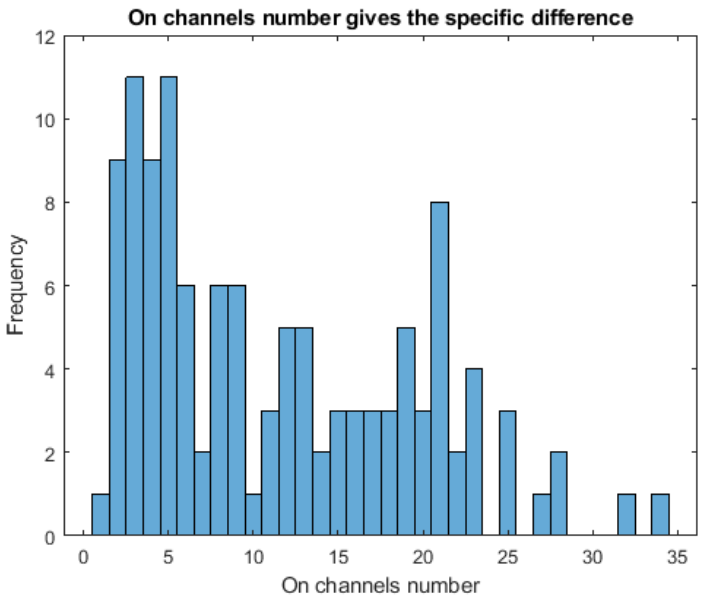


Figure 4.44: Number of ON-Channel

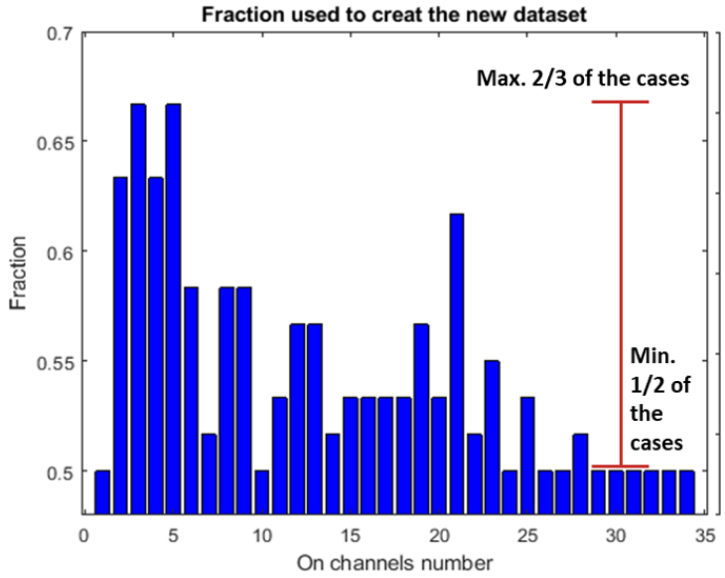


Figure 4.45: Data Fraction

On the basis of selected data, another distribution shown in figure 4.46.On the basis of that distribution, number of ON-Channels are shown in 4.47.On the basis of new distribution of OSNR 4.46 Selected difference between the predicted and telemetry is $OSNR = (Std = 0.070271)$. Number of on channels are showed in figure 4.48. With the new data and difference between predicted and telemetry shown in figure 4.43 $OSNR = (Std = 0.067355)$ number of ON channels are illustrated in 4.47. It can be seen that both histograms are same even threshold is different.

By creating new data-set with different way of data selection, there is no significant improvement in standard deviation is observed as shown in figure 4.43 and 4.46. but number of cases are reduced with this new method of creating data-set. it

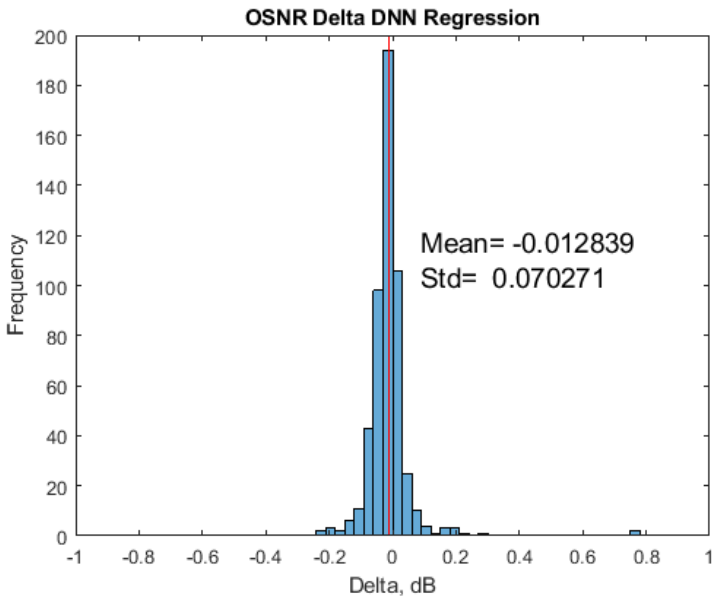


Figure 4.46: OSNR Delta DNN Histogram

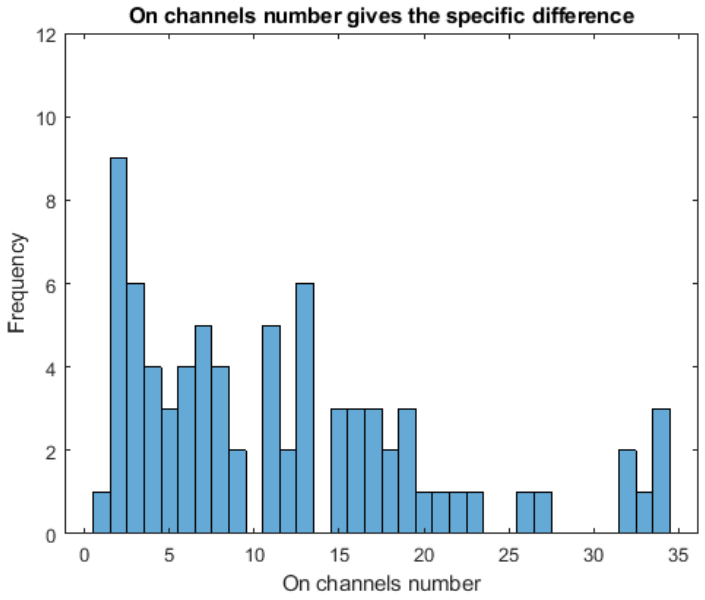


Figure 4.47: Number of ON-Channels

can be seen in figure 4.47 where case number 4 and 5 have 4 channels ON but with the new data-set its reduce to 3 figure 4.48

4.3.6.1 Data Analysis of Approach-II

In order to find out which combination contributes a lot in prediction of OSNR and which contributes to make prediction worst. To do so, all combinations (of telemetry) are sorted in increasing order with respect to OSNR while predicted OSNR combinations are sorted with respect to telemetry OSNR. Each combination has a unique ID as can be seen on xaxis in figure 4.49 while on yaxis represents OSNR.

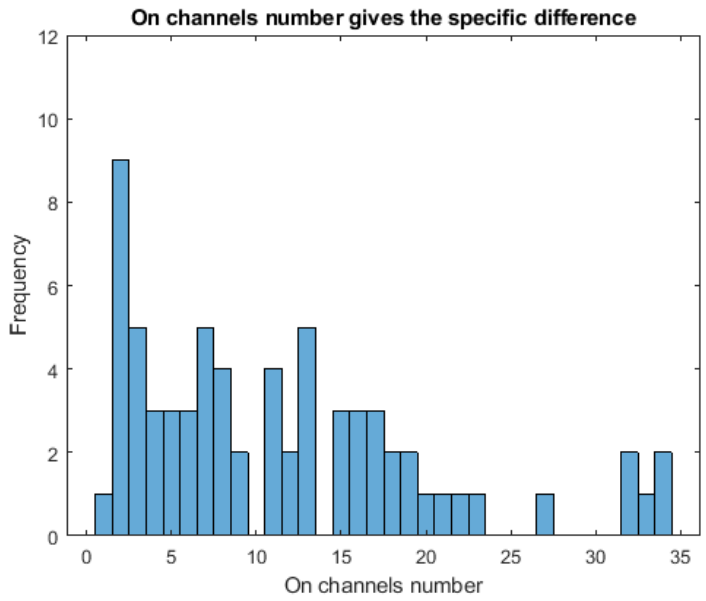


Figure 4.48: Number of ON-Channels

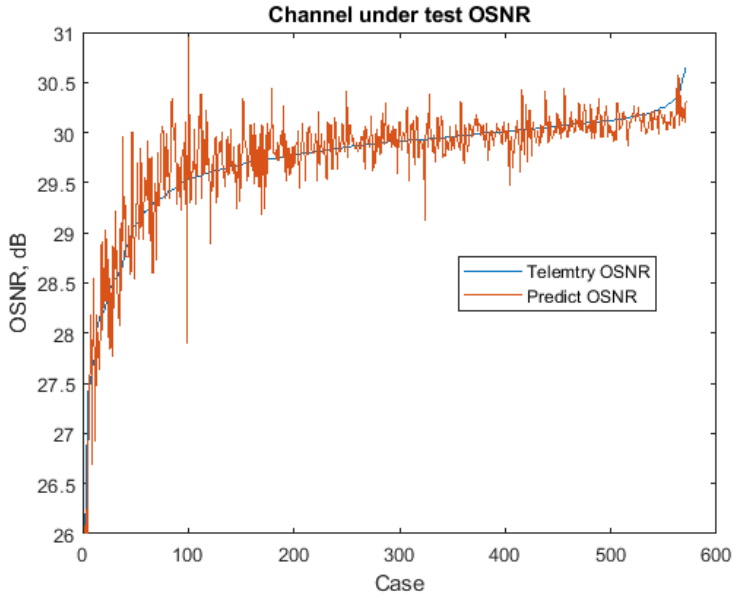


Figure 4.49: DATA Analysis osnr channel34 Linear curve

After applying DNN Regression, line curve shows telemetry data while on that curve fluctuations represents prediction. A threshold is set to filter cases where difference between the predicted and telemetry $OSNR = 0.5$ to observe behavior of combination towards prediction. Only 35 cases are found depicted in figure 4.50

Here half of the data has been chosen when the channel under test is ON and rest half is chosen when the channel under test is OFF. In figure 4.49 it can be seen that combination having ID=99 is a worst prediction.

Figure 4.50 represents number of ON channels in in each combination, for instance there were only 4 channels were ON in case 34. It is observed that when number of ON channels decreases, prediction is more close to original value and Δ

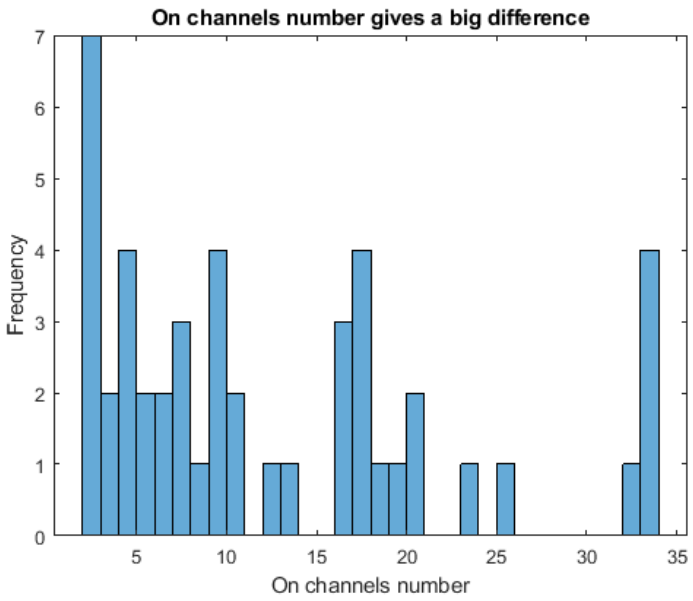


Figure 4.50: DATA Analysis osnr channel34 DNN histogram

is very small but can not be not conversely.

4.3.6.2 Data Analysis of Approach-III

In section approach III, this type of prediction is already described in which randomly the half of the data from every case is chosen where the channel under test is ON, mixed with data when the channel under test is OFF and data is analyzed with different number of layers.

4.3.6.2.1 Two Hidden Layers :

In figure 4.51 DNN Regression is used to predict OSNR. Predicted values and telemetry values of OSNR are compared, It can be seen that fluctuation are very less as compare to figure 4.49. Its just because in figure 4.51 used dataset is different and number of ON-Channels are are very few as compare to previous case so prediction is more close to telemetry.

4.3.6.2.2 Three Hidden Layers :

In figure 4.52 same dataset is used but DNN Regression with three hidden layers. it can be seen that fluctuations are reduced are compare to case where 2 hidden layers were used. Another factor of reducing fluctuations is very few number of channels are ON. There is only one case in which number of ON channel is one.

4.3.6.2.3 Four Hidden Layers :

In figure 4.53 same experiment is performed but with four hidden layers. it can be seen that fluctuations are reduced but the difference is not significant as compare to 3 layers. Also here only one case found which have OSNR greater than 0.5 and in that case there is only one channel is ON.

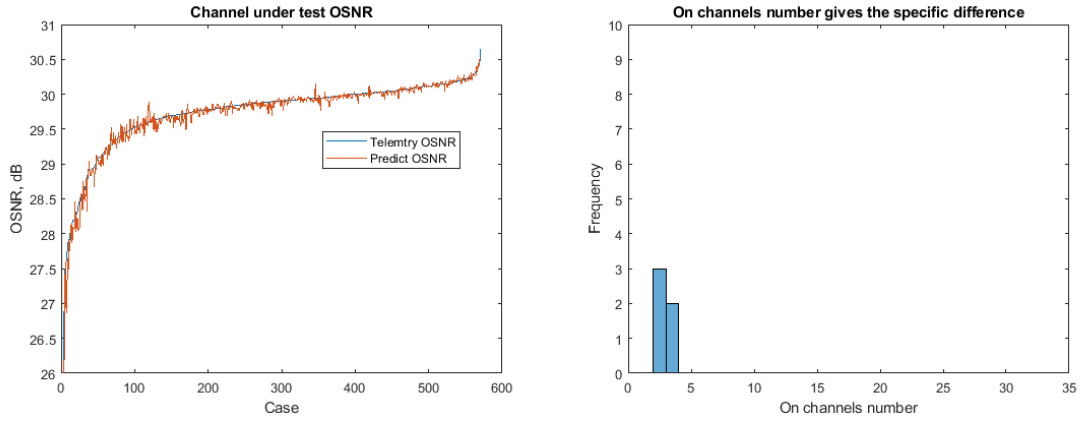


Figure 4.51: DATA Analysis 2 hidden layers

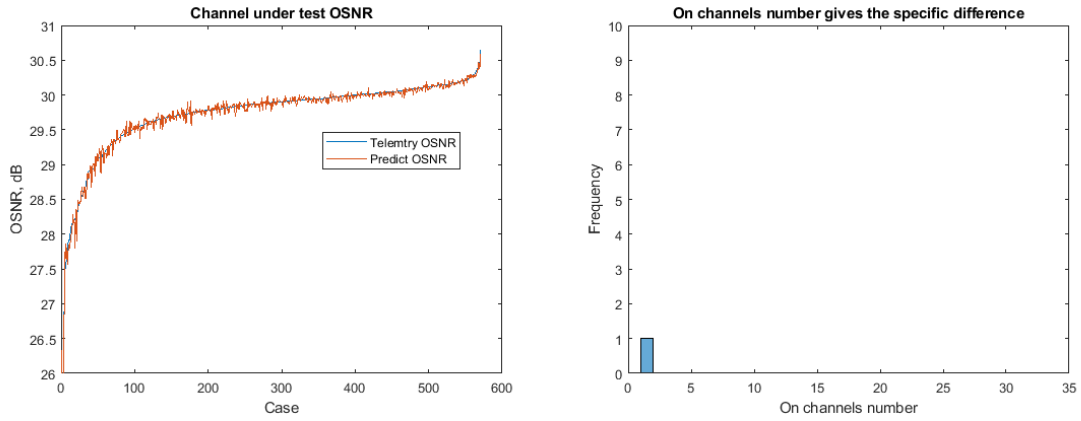


Figure 4.52: DATA Analysis 3 hidden layers

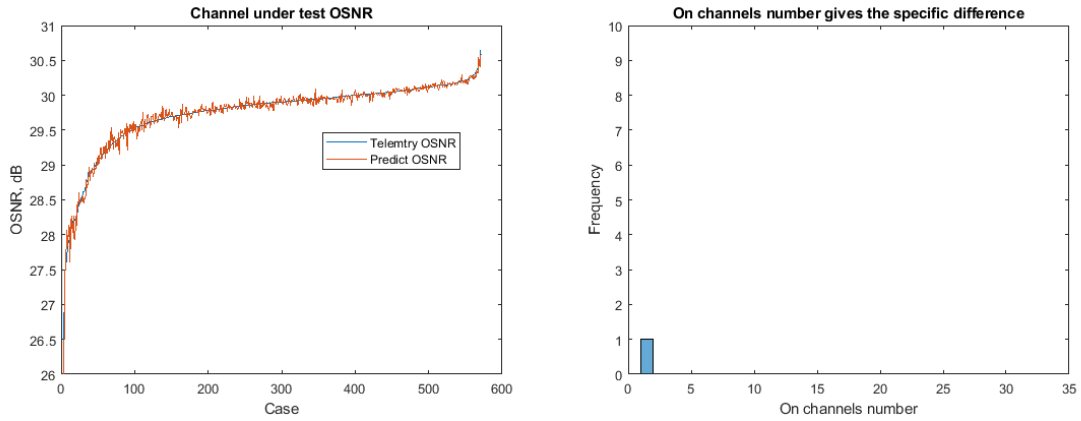


Figure 4.53: DATA Analysis 4 hidden layers

Chapter 5

Conclusion

Abrupt increase in network traffic demands open and flexible network on basis of optical network.In order to cope with heavy data traffic, intention of operators is to develop an optical network with lowest possible cost by using open line system (OLS). In state of the art network its is explained that QoT figure is SNR which is computing BER in OLS [8]. Therefore, it is important to predict QoT-Estimation a light path before its deployment. In study [5] author presented an approach for network planning, signaling and recovery. In this thesis, 11-span OLS is established to simulate 35 channels at 100 GHz with largest spectral hole burning effects [4].

Firstly, Machine Learning based simulation using LR and DNN regression by using TensorFlow® library as well as interpolated simulation is performed on synthetic data taken from line emulator. Secondly, it is applied on dataset collected from real line system. It is demonstrated that, mean and S.D are very high when OSNR is predicted by using interpolation as compare to ML-based OSNR estimation. Three different approaches are proposed to prepare dataset and to make prediction on real data extracted from experimental setup. Data is pre-processed by normalization with optimal normalization factor. In order to fill empty spaces in dataset, OSNR of OFF channels is interpolated. Basic data set depicted in 4.27 have OSNR but as it is known that OPM provides the output power only so, OSNR can not be measured directly. Therefore, dataset is modified to estimate QoT 4.28. In modified dataset OSNR is replaced with out-power of every channel. If input channel is OFF Pin is zero and Pout is equal to the Power of ASE. Similarly, if input channel is ON, then Pin is channel input power and Pout is received output power of channel. By using ML, Pout of all the 35 channels is predicted while in the figure 4.30 Pout34 is depicted only in order to compare the differences by predicting only one label or set of labels. Is can be seen that in both cases ML algorithms performed vey well. There is no significant difference observed in S.D when LR is used for predicting single label or predicting set of labels. But when DNN Regression is used it showed significant difference in SD because DNN works better when complexity increases.

In first approach, dataset illustrated in 4.28 can predicted Pout only so in order to estimate OSNR dataset is modified and illustrated in 4.31 where Transfer Function and P_{ASE} are added in previous dataset. The transfer function is basically the ratio between output power and ASE. To prepare this dataset two approaches are used as following

- Normalization

- Cubic Interpolation

Model is trained with two different ways and applied on both normalization and interpolation. In first method, during the training of model label Channel-34 is OFF and in prediction Channel-34 is ON. Form 4.7 results its observed that normalization is much better than interpolation. Because in every case standard deviation & Mean values by using normalization are less than interpolation. In second method, model is trained with mixed data where Channel-34 is ON & OFF where half of the data is randomly chosen when the channel under-test is ON, and mixed with data when the channel under test is OFF then Normalization is applied as it performed much better than Interpolation. In second approach, same dataset is prepared but with two different methods.

- Separate arrays for Tf and ASE

- A single normalized array

Tf and ASE is predicted by using two methods. Firstly, prediction is performed by choosing randomly half of the data when the channel under test is ON and rest when the channel under test is OFF. After mixing and Tf and ASE are predicted. Secondly, Instead of choosing randomly, half of the data is chosen, from each subset (on basis of how many channels are ON) of dataset where channel under test is ON and mixed with data when the channel under test is OFF then Tf and ASE are predicted. To increase performance of ML model, ASE is normalized before prediction in order to be in same range with Tf in single normalized array depicted in figure **??**. After prediction, denormalization is performed by multiplying the normalized predicted array with the mean of transfer function to get original values of Tf array . Similarly, ASE array is denormalized by multiplying the normalized predicted array with the mean of ASE. OSNR is calculated by following.

$$OSNR = \frac{P_{OUT}(=P_{in}x(Tf)}{ASE}$$

It is observed that, In figure 4.37 SD. is 0.28336 for DNN Regression while in figure 4.35 SD. is 0.41408 when data is mixed randomly. It can be clearly seen that prediction of OSNR is much better in figure 4.37 when half of the data from every case when the channel under test is ON is selected.

In third approach, dataset preparation is modified to train ML in such a way so that is it can predict with more accuracy. It's prepared by using Tf array along with the normalized array instead of ASE array to avoid normalization before prediction and denormalization after prediction. As its described above during the training half of the data is chosen, from each subset (on basis of how many channels are ON) of dataset where channel under test is ON and mixed with data when the channel under test is OFF. With this dataset two types of predication are made in order to compare their performances.

- Predict TF and ASE

- Predict Tf and single normalized array

On base of standard deviation it is observed that prediction of Normalized Array & Tf is the best prediction type with DNN among all others.

After implementing different techniques for dataset preparation and prediction types number of hidden layers in DNN is tuned. From experiments it is observed that with the increasing of number of hidden layers can cause significant decrease in standard deviation. Figure 4.41 shows the comparison between different hidden layer with DNN. Remarkable improvements can be seen when number of hidden layer stepped from 2 to 3 standard deviation reduces from 0.13892 to 0.06692. But from hidden layer 3 to onward till 5, there is no such notable difference is standard deviation. So its better to use three hidden layers with DNN Regression for QoT-Estimation. The distribution of OSNR is given in figure 5.1. Data analysis is also investigated by tuning number of hidden layers. Consequences are same as described above. In figure 4.52 DNN with three hidden layers is performed and it can be seen that fluctuations are reduced are compare to case where 2 hidden layers were used.

In order to find out which combination gives more Δ (difference between predicted value and original value) data analysis is performed in which a filter is applied to examine the cases where difference between predicted and telemetry OSNR in more than 0.5. Only 35 cases are found and depicted in figure 4.50. The distribution depicted in figure 4.44 is used to reproduce new dataset by giving flexibility between 1/2 of the cases over 2/3 cases. Previously, half of data has been chosen when channel under test was ON. Instead of fixing data selection, flexibility in data selection is introduced on base of figure 4.45. For minimum number of cases 1/2 is chosen and for maximum number of cases 2/3. Selection will vary between this range of minimum and maximum.

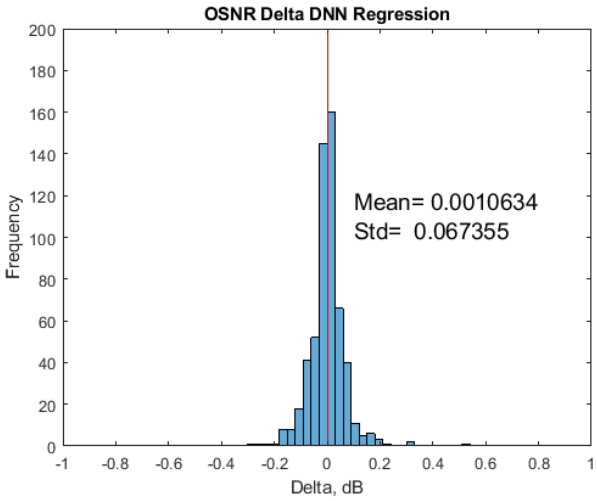


Figure 5.1:
OSNR Delta DNN Histogram with
3-Hidden Layers

On the basis of new distribution of OSNR 4.46 Selected difference between the predicted and telemetry is $OSNR = (Std = 0.070271)$. Number of on channels are showed in figure 4.48. With the new data and difference between predicted and telemetry shown in figure 5.1 $OSNR = (Std = 0.067355)$ number of ON channels are illustrated in **??**. It can be seen that both histograms are same even threshold is different. By creating new dataset with different way of data selection, there is no significant improvement in standard deviation is observed as shown in figure 5.1 and 4.46. But number of cases are reduced with this new method of creating dataset. it can be seen in figure **??** where case of four ON-Channels and five ON-Channels but with the new dataset its reduce to 3 figure 4.48

Finally, OSNR of un-established light path in open line system is predicted with prediction error 1.5(dB) by using telemetry data but by using ML techniques there is

significant decrease in prediction error and it is reduced to 0.06(dB). The distribution of OSNR is given in figure 5.1.

Chapter 6

Future Work

Further work will be to automate the measurement process of operational parameters from network equipment, especially optical amplifiers, to streamline the overall QoT estimation process.

Bibliography

[1] Abdelhadi Azzouni, Raouf Boutaba, and Guy Pujolle. "NeuRoute: Predictive dynamic routing for software-defined networks". In: *2017 13th International Conference on Network and Service Management (CNSM)*. IEEE. 2017, pp. 1–6.

[2] Erick de A Barboza et al. "Self-adaptive erbium-doped fiber amplifiers using machine learning". In: *2013 SBMO/IEEE MTT-S International Microwave & Optoelectronics Conference (IMOC)*. IEEE. 2013, pp. 1–5.

[3] Luca Barletta et al. "QoT estimation for unestablished lighpaths using machine learning". In: *Optical Fiber Communication Conference*. Optical Society of America. 2017, Th1J–1.

[4] Maxim Bolshtyansky. "Spectral hole burning in erbium-doped fiber amplifiers". In: *Journal of lightwave technology* 21.4 (2003), pp. 1032–1038.

[5] Vittorio Curri, Mattia Cantono, and Roberto Gaudino. "Elastic all-optical networks: A new paradigm enabled by the physical layer. How to optimize network performances?" In: *Journal of Lightwave Technology* 35.6 (2017), pp. 1211–1221.

[6] Zhenhua Dong et al. "Optical performance monitoring: A review of current and future technologies". In: *Journal of Lightwave Technology* 34.2 (2016), pp. 525–543.

[7] Natalia Fernández et al. "Techno-economic advantages of cognitive virtual topology design". In: *39th European Conference and Exhibition on Optical Communication (ECOC 2013)*. IET. 2013, pp. 1–3.

[8] Mark Filer et al. "Multi-vendor experimental validation of an open source QoT estimator for optical networks". In: *Journal of Lightwave Technology* 36.15 (2018), pp. 3073–3082.

[9] Yishen Huang et al. "A machine learning approach for dynamic optical channel add/drop strategies that minimize EDFA power excursions". In: *ECOC 2016; 42nd European Conference on Optical Communication*. VDE. 2016, pp. 1–3.

[10] Yishen Huang et al. "Dynamic power pre-adjustments with machine learning that mitigate EDFA excursions during defragmentation". In: *2017 Optical Fiber Communications Conference and Exhibition (OFC)*. IEEE. 2017, pp. 1–3.

[11] Kiyo Ishii, Junya Kurumida, and Shu Namiki. "Wavelength assignment dependency of AGC EDFA gain offset under dynamic optical circuit switching". In: *Optical Fiber Communication Conference*. Optical Society of America. 2014, W3E–4.

[12] Alan Pak Tao Lau and Joseph M Kahn. "Signal design and detection in presence of nonlinear phase noise". In: *Journal of Lightwave Technology* 25.10 (2007), pp. 3008–3016.

[13] Álvaro López-Raventós et al. "Machine Learning and Software Defined Networks for High-Density WLANs". In: *arXiv preprint arXiv:1804.05534* (2018).

[14] Ignacio de Miguel et al. "Cognitive dynamic optical networks". In: *Journal of Optical Communications and Networking* 5.10 (2013), A107–A118.

[15] Fernando Morales et al. "Virtual network topology adaptability based on data analytics for traffic prediction". In: *IEEE/OSA Journal of Optical Communications and Networking* 9.1 (2017), A35–A45.

[16] Antonio Napoli et al. "Reduced complexity digital back-propagation methods for optical communication systems". In: *Journal of lightwave technology* 32.7 (2014), pp. 1351–1362.

[17] Juliano R Oliveira et al. "Demonstration of EDFA cognitive gain control via GMPLS for mixed modulation formats in heterogeneous optical networks". In: *Optical Fiber Communication Conference*. Optical Society of America. 2013, OW1H–2.

[18] Shahin Shahkarami et al. "Machine-learning-based soft-failure detection and identification in optical networks". In: *2018 Optical Fiber Communications Conference and Exposition (OFC)*. IEEE. 2018, pp. 1–3.

[19] Takahito Tanimura et al. "OSNR monitoring by deep neural networks trained with asynchronously sampled data". In: *2016 21st OptoElectronics and Communications Conference (OECC) held jointly with 2016 International Conference on Photonics in Switching (PS)*. IEEE. 2016, pp. 1–3.

[20] Jakob Thrane et al. "Machine learning techniques for optical performance monitoring from directly detected PDM-QAM signals". In: *Journal of Lightwave Technology* 35.4 (2017), pp. 868–875.