Optimal Parameter Estimation for acheiving Low Latency and Reliable Communication using Deep Neural Nets

Battula Navya

Prasad V. Potluri Siddhartha Institute of Technology, Vijayawada, Andhra Pradesh, 520007

Dr. P Vijay Kumar

Department of Electrical Communication Engineering, Indian Institute of Science, Bengaluru, 560012

Table of Contents

Optimal Parameter Estimation for acheiving Low Latency and Reliable Communication using Deep Neural Nets

Battula Navya

Prasad V. Potluri Siddhartha Institute of Technology, Vijayawada, Andhra Pradesh, 520007

Dr. P Vijay Kumar

Department of Electrical Communication Engineering, Indian Institute of Science, Bengaluru, 560012

Abstract

The possibility of estimating channel parameters of Gilbert Elliot channel from the packet drop statistics is essential especially in situations when only packet drop statistics were present to analyze the channel during which it would be difficult to track the channel parameters in absence of any procedures dedicated to estimating them. In our work, we focused on using certain deep learning procedures to estimate the approximate p, r, h and k (the channel parameters) values from the given packet drop statistics. In this work, we have also discussed various attempts dedicated to training the model for desired results. We explained the results obtained in different models used and provided an analysis of these models in terms of accuracy, performance and tolerance.

Keywords or phrases: Gilbert Elliot Channel, Artificial Neural Network, Adam Optimizer.

Abbreviations

Abbreviations

MSD	Maximum Distance Seperable
MSE	Mean Squared Error
ReLU	Rectified Linear Unit

1 INTRODUCTION

1.1 Background

The need for reliable low latency communication which can be best termed as Ultra Reliable, Low Latency Communication has garnered a great deal of research in the field of 5G technology. The technologies including live gaming, virtual reality, high-quality video conferencing, Internet of Things (IOT), etc were developed by using reliable and low latency transmission as the key component. However, there are certain challenges and tradeoffs in attaining reliable, low latency communication over the channels. There is some research in this area aimed at finding out optimal conditions for packet transmission and channel modelling. Although various research topics focused on various problems that networks often faced, the necessity for modelling a channel with parameters that give desired rate, latency and performance is the primary motivation for this research. In this work we mainly focus on using packet drop statistics to obtain the channel parameters using an efficient Deep Neural Network.

1.2 Problem Statement

The erasure probabilities and the transition probabilities in a Gilbert Elliot channel are crucial in studying other parameters like the burst erasure length b or the number of random erasures a that occur in a channel. Therefore these parameters are crucial in studying channel properties. However in cases where only packet drop statistics are available, there should be an effective method that could estimate the approximate parameter values from the given packet drop statistics. Using the binary packet drop values as inputs, a model should be able to identify the underlying patterns within those packet sequences to output the approximate parameters.

1.3 Objectives of the Research

The basic objectives of the following research include - collection of random values and generating the packet drop statistics to construct a dataset, label the parameters and fit the

model, re-evaluate the model by adjusting the hyper parameters, table the observations and evaluate the results.

1.4 Scope

The values used in these experiments are generated through experiments and are not real values. The dataset uses the generated values and not those values that are taken from the actual physical channel.

2 LITERATURE REVIEW

2.1 Information

The literature survey mentioned includes various areas where there is a potential research to optimize these tradeoffs to give maximum performance and efficiency. Dealing with burst erasures and random erasures is no smaller challenge. In John Apostolopoulos, 2019, the authors propose the optimal streaming codes that are suitable for handling multiple arbitrary erasures or a single burst erasure in a given window of a fixed size over the Gilbert Elliot channel and Fritchman channel. In P. Vijay Kumar, 2020, the authors propose a novel method staggered diagonal approach which gives simpler, low complexity construction of rate optimal streaming codes having linear field size. They address the efficacy of the staggered diagonal approach over the diagonal approach by illustrating examples and by providing streaming code construction that use staggered diagonal approach. The maximum rate achieved under this approach is presented using MDS base codes. The problem of variable length streaming packets is addressed in K. V. Rashmi, 2018 where the author presents the streaming codes for variable size packets that come with tradeoffs like rate of the code and minimum decoding delay. The approach in Gerhard Hasslinger, 2008 discusses the use of packet loss patterns based on second order statistics for fitting the Gilbert Elliot model. Although various research topics focused on various problems that networks often faced, the necessity for modelling a channel with parameters that give desired rate, latency and performance is a crucial step towards this research.

3 Proposed work

3.1 Gilbert Elliot channel

The Gilbert Elliot channel introduced by Edgar Gilbert and E.O. Elliot is a simple channel model based on a Markov chain with two states – good state and bad state. E. N. Gilbert, 1960 E. O. Elliott, 1963, E. O. Elliott, 1963. The Gilbert Elliot channel is widely used for describing of errors. The good state represents a state where the data transmission is fruitful and the bad state represents the vice versa. The parameters p and r represent the transition probabilities of the channel model. The 1-h and 1-k values represent the good state and bad state erasure probabilities. Look at figure 1 to understand the transition probabilities and state dependant error rates. The GE model is defined by the transition matrix M between states s at time t:

$$M = \left(\begin{bmatrix} 1-p & p \\ r & 1-r \end{bmatrix} \right)$$

where p = P(s = B|s1 = G), and r = P(s = G|s1 = B)

Here the state s and state s1 are two states that describing the previous state and the current state in the transition. The transition probability p is defined when state s is in bad state and state s1 is in good state and r as vice versa. The values 1-k and 1-h represent the state dependent error rate in good state and bad state respectively.

The sim2net package consists of methods like GilbertElliot and packet_loss within the packet loss module. The packet loss module consists of various methods that can be used to understand the packet statistics and the parameters of the Gilbert Elliot model. The full documentation and implementations of GilbertElliot method and packet_loss are available in

github. The packet sequences represent the number of erasures that take place based on which, the p, r, h and k values are estimated.

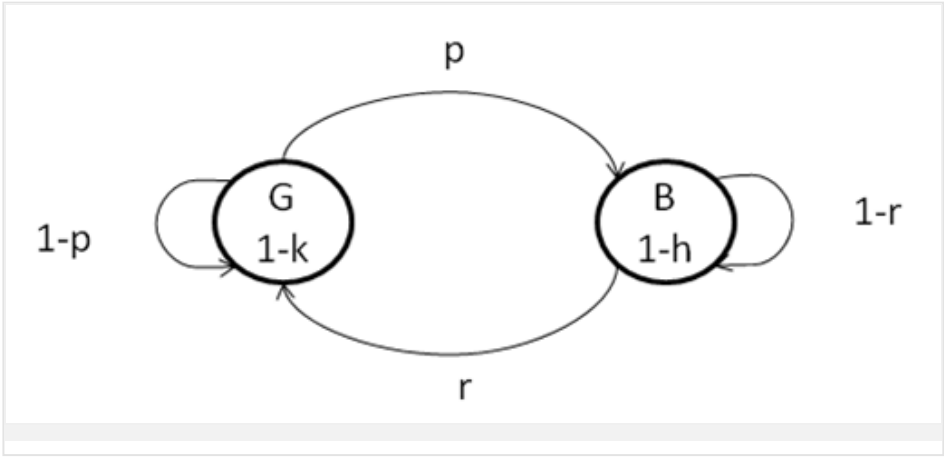


Figure 1:Gilbert Elliot model diagram.

Our research focuses on estimating optimal values of p, r, h and k by training the deep neural network model to provide approximately accurate values for the given packet drop statistics. The packet sequences were given as inputs for the model and based on the length of the sequence, the no of erasures and the accuracies, the model predicts the closest p, r, h and k values. The packet drop sequences are the binary values that indicate whether the given packet has an erasure or not. The typical packet sequence of length 10 looks like this: 0010011101. Note that the 0's represent no erasures and the 1's represents occurrences of erasures. These packet sequences are to be obtained using the random p, r, h and k values where the p, r, h and values are given to the Gilbert Elliot method to generate packet sequences for desired number of packets. On collecting the values of this packet drop sequences, we can use them to train the model to learn the correlation between the parameters and the packet sequences. Once the model starts to converge, additional packet sequence values out of the training set can also be provided to obtain the parameters.

3.2 Intuition on Deep Neural Networks

Before actually going into the deep neural network used by us, we want to provide a brief introduction to neural networks and deep neural networks. Neural Network is an algorithm that identifies the underlying relationships between the data just like other algorithms like Support Vector machines, Random Forests and Regression models. They are composed of basic units called neurons which are the primitive function unit in the network. Every neuron whether it is biological or artificial possesses a threshold for firing. In artificial neural networks this threshold is known as activation function. Each neuron functions as an individual perceptron and process inputs to recognize any patterns and correlations between the data. While connecting with other neurons, the synapses contain weights which act as

inputs to the next layers. Typically neural networks can have one or several hidden layers. The network layers stacked on one another not only feed forwards the data but also back propagates them. In most cases there can also be biases added to the neurons for maintaining proper results. Learning rate is a hyper parameter that indicates the rate at which the machine learns. Learning rate can be tricky because a higher learning rate could make the machine converge quickly but the final accuracy is not desirable as the model didn't converge properly. A lower learning rate on the other hand delays the process enormously which is not desirable. Therefore in most cases, the learning rate is adaptive according to the application and data size. A neural network typically learns using supervised learning by using results and the estimated values and the resultant errors or losses as the key to learn the relationships by using probabilistic association of weights. When the error is significant enough and the required data value varies the actual value by a larger value the model back propagates and adjusts its weights to accompany proper predictions. The model trains continuously until a convergence point is reached from which the progress is negligible.

Along with the hyper parameters and the data set, the network architecture is also crucial in developing a neural network. Here is a sample diagram of a neural network.

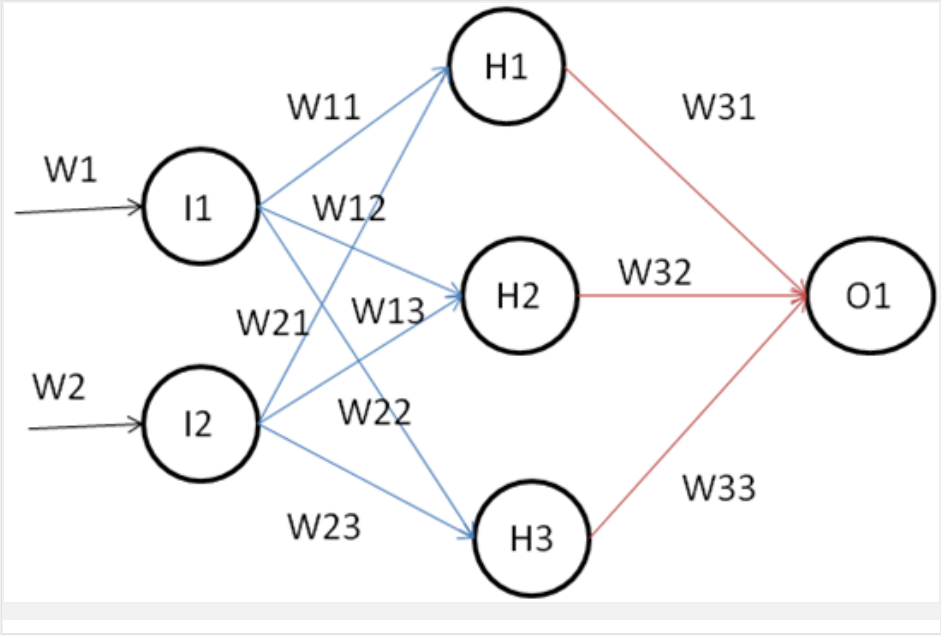


Figure 2: A sample neural network with 3 layers

The above network consists of one input layers, one hidden layer and one output layer. Note that the neurons in input layer are labeled as I1 and I2 and those of hidden layer are labeled as H1, H2 and H3. Note that the weights represent the inputs to the neurons which help them fire. The output layer consists of a single neuron O1 which outputs the desired output value. In developing architecture it is essential to note that the number of neurons, the number of hyper parameters and the number of inputs to each layer can have a crucial effect on the

training procedure. Therefore finding a proper architecture for a model is always adaptive. In our model we have experimented on different types of structures which gave different results each time. There were some failed attempts, partially successful attempts and attempts that met the requirements.

A deep neural network is a large network composed of many hidden layers with many neurons embedded in each layer. Generally the basic difference between neural networks and the deep learning neural networks comes from their data handling capacities. Larger neural networks like deep neural networks handle larger data effectively compared to the smaller networks. Therefore deep neural networks efficiently deal non numerical data like images, video, audio, digital signals, etc. Of course there are different deep neural networks that deal separately with images, audio, etc. Convolutional networks are an example of deep neural networks that deal with images. Similarly audio, text and other kind of sequence data are handled by recurrent neural networks. Depending on the application, the structure of the neural network changes and for our application, we have employed a simple model that accepts numerical inputs. We will discuss its architecture briefly in the next section.

3.2.1 Dataset

The dataset that we used in training the model was developed by us. Initially we extracted a set of random values of p, r, h and k parameters between 0 and 1. The numpy method np.random can be used to generate the random values for the parameters. These random values can be used to obtain packet drop statistics for n number of packets. The sim2net package consists of different modules like packet_loss, gilbert_elliot, etc. The GilbertElliot() method takes the list of p, r, h and k values as input and the process them. The packet drop statistics can be obtained from the packet_loss() method whose return type is a Boolean value indicating the chance of erasure. The packet drop values are collected for the desired number of packets which becomes the sequence length of the packet sequence. Initially we developed a dataset of 10,000 sequences consisting 300 packet drop statistics in each sequence. The dataset was tested with 2 different procedures which gave surprisingly different results. The dataset was later improved to contain 12,000 packet sequences with 10,000 packet drop statistics in each sequence. Obviously the larger dataset gave better results after training procedure. The p, r, h and k values are floating point values between 0 and 1 with precision 6.

3.2.2 Various attempts

In the process of developing a model, we made different attempts with different algorithms and neural network architectures. Initially using the smaller dataset, we experimented with the simple linear regression model. The machine yielded poor results due to the size of the data. Then we experimented with a rudimentary neural network to test the viability of the procedure. There was a huge difference in the losses and the accuracy values and the closeness between the actual value and the predicted value. The rudimentary neural network was further developed to incorporate larger data and train efficiently taking less time to converge.

Failed attempt

Initially the smaller dataset comprising of 300 length sequences is trained using some machine learning techniques like support vector machine, simple linear regression and Random Forest classifier. These methods failed to give desired results and the resultant accuracies were too low. The percentage of data with the difference less than 0.1 between actual value and the predicted value is as low as 27%. The percentage of data with the difference in range 0.1 to 0.2 between actual and predicted value is about 24%. The percentage of data with the difference in range 0.2 to 0.6 between the actual and predicted value is about 40%. The rest of the data showed more than 0.6 difference than the actual data. The normalized mean squared errors are in the range -0.818151 to 0.012671. With very low accurate predictions and higher error rate the attempt of making predictions with the linear regression model was considered to be a failure. The similar results were observed with the Random Forest classifier and the support vector machine.

Rudimentary Neural Network

After repeated reports of lower accuracies in the machine learning models, we have decided that the amount of data is one of the main causes for these models to fail. Usually deep neural networks are said to be more effective for bulk data compared to machine learning methods. In that process of exploration, we have tried constructing a basic neural network with 8 layers. Seven layers had 16 neurons each with ReLU activation function. The last layer consisted of 4

neurons with sigmoid activation. The input layer accepted 300 inputs and the remaining layers accept 16 inputs each. The last layer however accepts 4 input values.

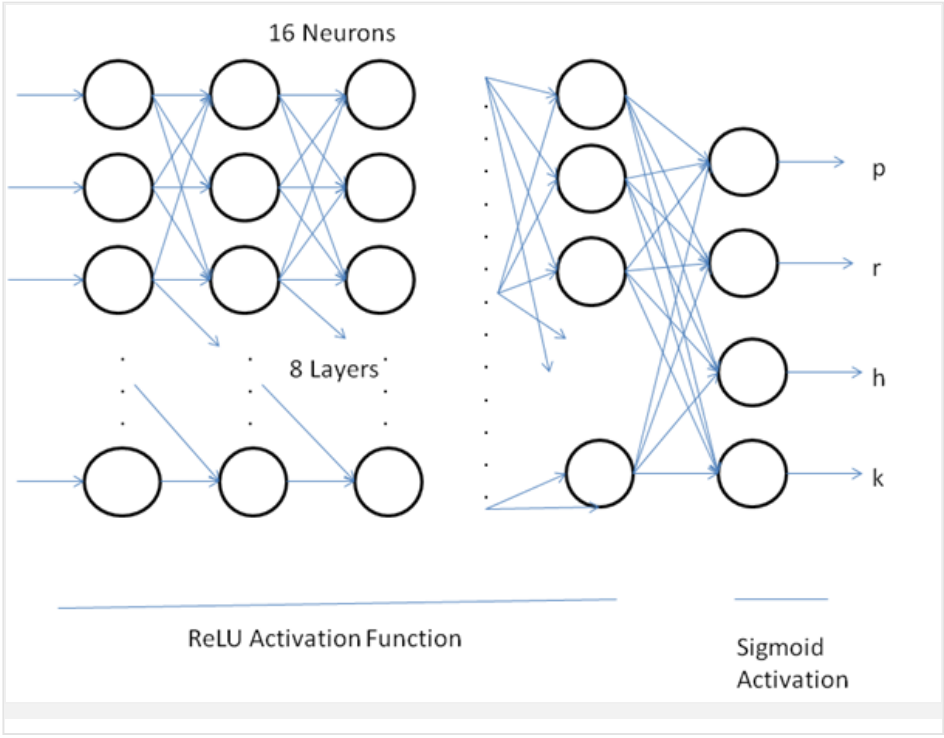


Figure 3: Rough diagrammatic representation ofthe rudimentary neural network.

The model gave an accuracy of 55% on running 1000 epochs with a loss of around 3.7%. By accuracy we mean that the accuracy achieved is on a single precision rather than a 6 precision. The values of parameters are rounded off to a single precision rounding the closest values to nearest single digit after the decimal by the model. The formula for the accuracy is given by:

accuracy = (num of values that correspond to original values / total num of values) * 100

and the formula for mean squared error is given by the formula:

$$MSE = (\sum (y1 - y2)^2 / x1) * 100,$$

where the values y1 and y2 correspond to the actual value and the predicted value and x1 corresponds to the total number of values.

The model is implemented using the keras sequential model which uses mean squared error as the loss function and Adam optimizer Jui-Yu Wu, 2012 as a replacement for stochastic gradient descent approach used in machine learning models. The Adam Optimizer is a combination of

RMSprop, Stochastic gradient with momentum. Unlike stochastic gradient, it computes individual learning rates for various parameters and hence an adaptive learning method. By using adaptive learning methods like Adam Optimizer the model can slowly unravel the loose relationships between the data values hence giving higher accuracies.

The model converged slowly and showed gradual increments in the accuracy. Although the closeness of the actual values and the predicted values is high, the model did suffer from some performance issues.

Final Model

To address the performance issues in the rudimentary neural network and to incorporate larger dataset, the model was revised with more layers with more neurons in each layer and with a prominent structure to train efficiently and address the performance issues. The final model comprised of 16 layers with decremental fashion of arrangement of the number of neurons. The larger dataset was used to train this model which consisted of 10,000 packet drop statistics in a single sequence and 12,000 packet sequences. The model initially started with lower accuracies and higher loss values in the first few epochs. But unlike the rudimentary neural network discussed previously, the final model started converging sooner with faster increments in accuracies. When it comes to accuracies and losses, the similar methods were used as in the rudimentary network. The precision of the output values was reduced to 1 instead of 6 by the model. The initial accuracy the model demonstrated is 33% with a 7% loss. As the training progressed the model yielded a final accuracy between 65% to 75% and final loss of approximately 2.4% to 1.6%.

The mean squared error for individual parameters was also calculated and the model yielded overall mean squared error less than 0.1 which is a significant improvement compared to machine learning techniques. The mean squared error approximately obtained for the parameters p, r, h, k values respectively are: p= 0.07-0.03, r= 0.05-0.02, h= 0.03-0.015, k=0.03-0.01. The normalised mean squared error for p, r, h and k are respectively in ranges: p=0.011343-0.811819, r=0.001753-0.781100, h=0.000113-0.711222, k=0.003423-0.918111. The model performance is also dependent on hardware performance and it is possible that results

are also dependent on the hardware condition. However the range of accuracies, losses and mean squared error for individual parameters remain same.

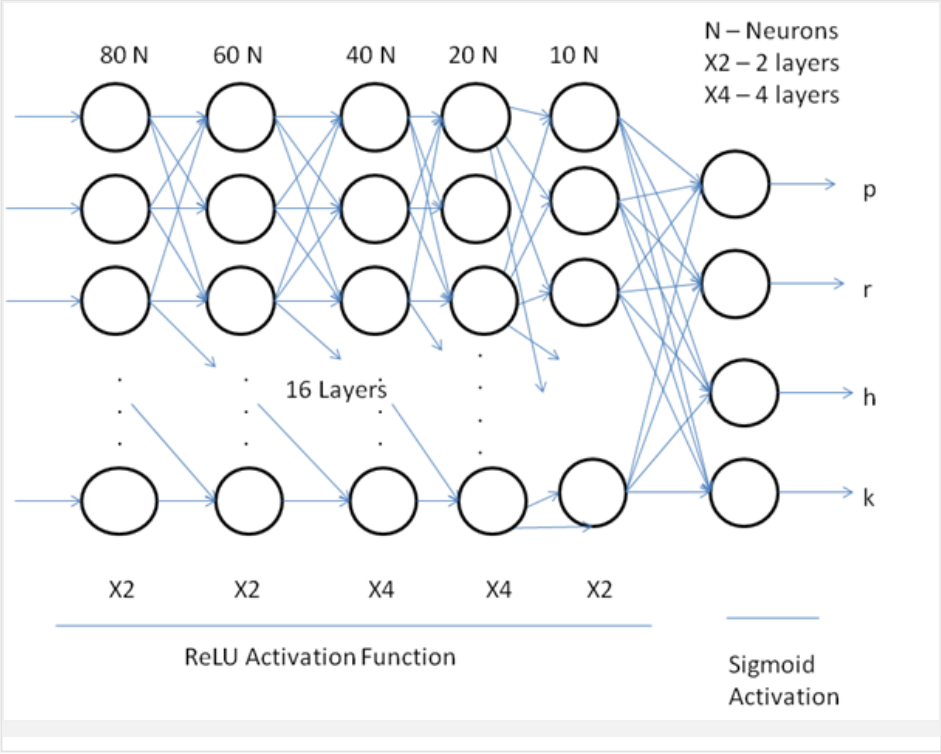


Figure 5: Rough diagrammaticrepresentation of the final neural network.

4 RESULTS AND DISCUSSION

In terms of accuracies and the closeness of the actual value and the predicted value, there is a gradual increase between different attempts. The simple linear regression model gave lower performance and accuracy after training and the accuracies reported in these methods were consistently quite low say around 15 – 16%. The r2 score and the mean squared error obtained on training the data is shown in table 1. The accuracies of the model were between 15-16% for 300 bit packet sequence dataset and between 14-15% in the 10,000 bit packet sequence.

Table 1 The r2Scores and accuracies for datasets in simple linear regression.

Dataset	r2 Score	Mean Squared Error
300 bit packet sequence	15.69	26.35

10,000 bit packet sequence	14.91	27.93

The accuracies of the rudimentary neural network are around 52 to 56%. Note that the reported accuracies in both rudimentary and final model are taken under consideration on rounding off to a single point precision. Initially the model demonstrated 15% accuracy and 27% loss in the first epoch. In the final epoch there was a tremendous reduce in loss and improvement in accuracy. The final accuracy was around 54.54% and final loss was around 3.7%. The accuracies of the model and its losses in different epochs were demonstrated in table 2.

Table 2 The accuracies and losses at different epochs in rudimentary neural network.

Value	Epoch 1	Epoch 100	Epoch 500	Epoch 750	Epoch 1000
Accuracy	15.23%	28.15%	37.87%	46.32%	54.54%
Loss	27.65%	19.6%	9.6%	4.51%	3.7%

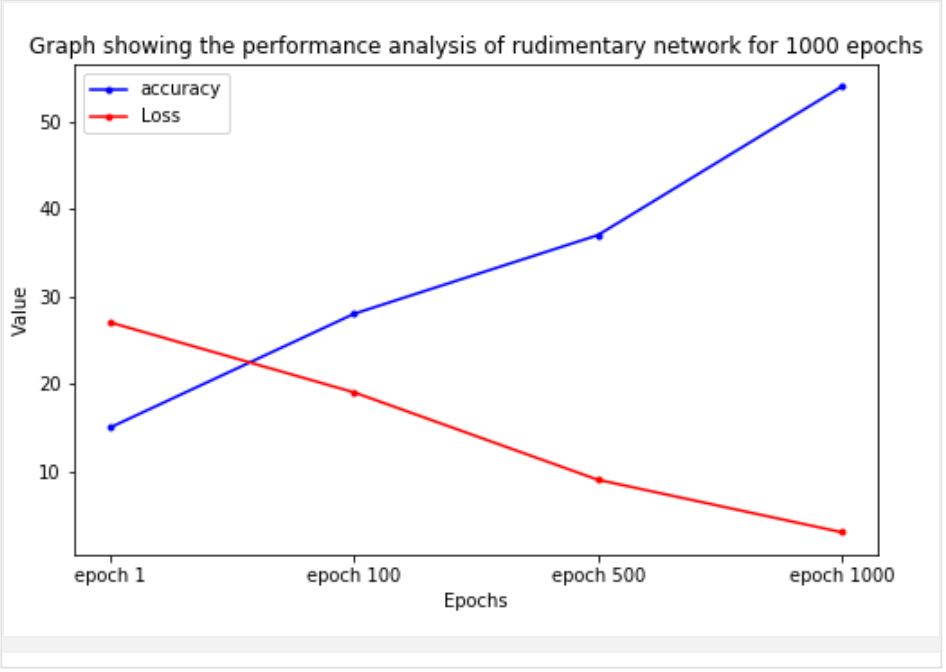
The accuracies on the final model were more desirable than that of the rudimentary neural network. The accuracies obtained for the final model in different epochs were demonstrated in table 3.

Table 3 The accuracies andlosses at different epochs in final model.

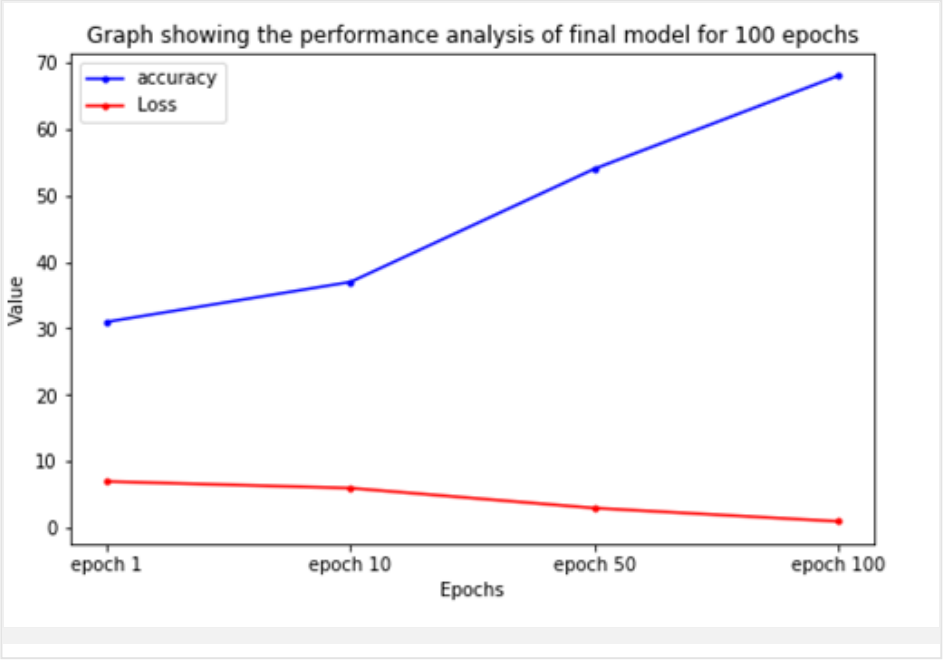
Value	Epoch 1	Epoch 10	Epoch 50	Epoch 750	Epoch 100
Accuracy	31.46%	37.21%	54.62%	62.35%	68.67%
Loss	7.37%	6.32%	3.7%	2.66%	1.96%

In terms of performance, the final neural network demonstrated better performance than the rudimentary neural network. The performance analysis of the rudimentary network is shown in graph 1. We can see that at the 100^{th} epoch, the rudimentary network didn't yet converge

properly. But when we observe the 100^{th} epoch in the final model in graph 2, it is clear that the model has converged properly and started giving desired results.



Graph 1:The graph that shows the performance ofrudimentary neural network.



Graph 2: Thegraph that shows performance analysis of the final neural network.

5 CONCLUSION AND FURTHER WORK

The use of deep learning techniques to estimate optimal parameters by using packet drop statistics provides us an insight on what deep neural networks can accomplish. If we were able to gather data properly, the deep neural networks can easily train on even the complex data and output favorable results. The application can be expanded various other problems in communication channels and models. The use of a better neural network structure can also prompt us to appropriate results. The intuition of deep neural networks can also be better realized with the dynamics of results according to data, hardware conditions, speed of convergence and the accuracies obtained in the end. With such intuitions, the model capacities can be enhanced by improvising the model performance.

Moreover, the p, r, h and k values play a crucial role in developing rate efficient streaming codes. The parameters like which represent the number of random erasures in a channel and b which represent the size of the burst erasure in a channel can be estimated by using these p, r, h and k values.

ACKNOWLEDGEMENTS

I want to thank the Indian Academy of Sciences for giving me this wonderfull opportunity. I am grateful to my guide Dr. K. Vijay Kumar for his guidance in the work and his support in completing the project. I am thankful for his patience in guiding me and analyzing my work. I want to extend my special thanks to Mr. Vinayak Ramakumar, Ph.D scholar in Indian Institute of Science for his help in clarifying doubts and injecting the concepts at anytime. An extra thanks for his support in evaluating my work and suggesting me any required changes. I am also thankful to the team that includes Myna Vajha and Nikhil Dande for analyzing my work and suggesting me anything required. I am also thankful to my family for their cosistent support in the process.

References

1. Silas L. Fong, Ashish Khisti, Baochun Li, Wai-Tian Tan, Xiaoqing Zhu, John Apostolopoulos, 2019, Optimal Streaming Codes for Channels With Burst and Arbitrary Erasures, IEEE Transactions on Information Theory, vol. 65, no. 7, pp. 4274-4292

2. M. Nikhil Krishnan, Vinayak Ramkumar, Myna Vajha, P. Vijay Kumar, 2020, Simple Streaming Codes for Reliable, Low-Latency Communication, IEEE Communications Letters, vol. 24, no. 2, pp. 249-253
3. Michael Rudow, K. V. Rashmi, 2018, Streaming Codes For Variable-Size Arrivals, 2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)
4. Oliver Hohlfeld, Rudiger Geib, Gerhard Hasslinger, 2008, Packet Loss in Real-Time Services: Markovian Models Generating QoE Impairments, 2008 16th Interntional Workshop on Quality of Service
5. E. N. Gilbert, 1960, Capacity of a Burst-Noise Channel, Bell System Technical Journal, vol. 39, no. 5, pp. 1253-1265
6. E. O. Elliott, 1963, Estimates of Error Rates for Codes on Burst-Noise Channels, Bell System Technical Journal, vol. 42, no. 5, pp. 1977-1997
7. Jui-Yu Wu, 2012, Stochastic Global Optimization Method for Solving Constrained Engineering Design Optimization Problems, 2012 Sixth International Conference on Genetic and Evolutionary Computing