OVER-FITTING AMELIORATION IN BAYESIAN NEURAL NETWORK MODEL ESTIMATION USING HETEROGENEOUS ACTIVATION FUNCTIONS

BY

Tayo Peter OGUNDUNMADE

Matric. Number: 151194

A Thesis in the Department of Statistics

Submitted to the Faculty of Science

in partial fulfilment of the requirements for the Degree of

DOCTOR OF PHILOSOPHY

of the

UNIVERSITY OF IBADAN

16TH AUGUST, 2023

CERTIFICATION

I certify that this work was carried out by Tayo Peter OGUNDUNMADE, with matriculation number 151194 in the Department of Statistics, Faculty of Science, University of Ibadan under my supervision.

....

Supervisor

Abosede A. Adepoju

B.Sc.(Ilorin), M.Sc., Ph.D(Ibadan)

Professor, Department of Statistics,

University of Ibadan, Nigeria.

DEDICATION

I dedicate this work to the Almighty God, who Himself represents wisdom, knowledge and understanding; to my dear wife and children (Omololaeni, Eriifeoluwasimi and Anuoluwapo) whose love, support and endurance has enabled me be where I am today; I cherish you; and lastly to my late father; Pastor Samuel Ogundunmade.

ACKNOWLEDGEMENTS

I thank God for His unconditional mercy, guidance and blessing over me since the commencement of my life and particularly during the sleepless day and night researches on this write up, without you nothing is possible.

I sincerely acknowledge my supervisor, Professor Adedayo A. Adepoju for painstakingly read through the draft, thorough supervision, positive criticism, immeasurable contribution, motherly support, unquantifiable encouragement, uncommon kindness, lexical advice, guidance and patience to the success of this thesis. Ma, I shall be eternally grateful.

I specially appreciate the assistance of all the teaching and non-teaching staff of the Department of Statistics, University of Ibadan, the Head of Department, Prof. O.I Osowole and the entire staff of the department. All your ebullient efforts and contributions are immensely valued and cherished. I wish you all, abundant blessing in all your endeavours.

My hearty thanks goes to my Late Father, Pastor Samuel Ogundunmade. Also to my caring indefatigable Mother, Deaconess Felicia Ogundunmade, Brothers and Sisters whose names are; Mr Sunday Ogundunmade, Mrs Tolulope Ogundunmade, Mrs Abosede Ogundunmade and Mr Dapo Ogundunmade. and others. I appreciate you all.

My deepest appreciation goes to My wife, Mrs Omolola Ogundunmade and my children, Eriifeoluwasimi and Anuoluwapo Ogundunmade, for your care and support. I love you all.

Greatest ideological Uites, it is quite an opportunity and a privilege to be a graduate of this strong, politically conscious, economically vigilant, academically dynamic, indefatigable, invincible, ineffable, indomitable, virile, articulate, iconoclastic, **the first** and **the best** University in Nigeria. For many were called to this great citadel of learning but few were chosen yet the fittest survive.

Once again, I give a special thanks to God Almighty God for the perseverance, endurance and life of full health to achieve the Golden Feat. To God be the glory!!!
OGUNDUNMADE, Tayo Peter

ABSTRACT

Neural Network (NN) allows complex nonlinear relationships between the response variables and its predictors. The Deep NN have made notable contributions across computer vision, reinforcement learning, speech recognition and natural language processing. Previous studies have obtained the parameters of NN through the classical approach using Homogeneous Activation Functions (HOMAFs). However, a major setback of NN using the classical approach is its tendency to over-fit. Therefore, this study was aimed at developing a Bayesian NN (BNN) model to ameliorate over-fitting using Heterogeneous Activation Functions (HETAFs).

A BNN model was developed with Gaussian error distribution for the likelihood function; inverse gamma and inverse Wishart priors for the parameters, to obtain the BNN estimators. The HOMAFs (Rectified Linear Unit (ReLU), Sigmoid and Hyperbolic Tangent Sigmoid (TANSIG)) and HETAFs (Symmetric Saturated Linear Hyperbolic Tangent (SSLHT) and Symmetric Saturated Linear Hyperbolic Tangent Sigmoid (SSLHTS)) were used to activate the model parameters.The Bayesian approach was used to ameliorate the problem of over-fitting, while the Posterior Mean (PM), Posterior Standard Deviation (PSD) and Numerical Standard Error (NSE) were used to determine the estimators' sensitivity. The performance of the Bayesian estimators from each of the activation functions was evaluated in the Monte Carlo experiment using the Mean Square Error (MSE), Mean Absolute Error (MAE) and training error as metrics. The proximity of MSE and training error values were used to generalise on the problem of over-fitting.

The derived Bayesian estimators were $\beta \sim N(K_\beta, H_\beta)$ and $\gamma \sim \exp\left(-\frac{1}{2}\{F_\gamma + M_\gamma\right)$; where K_β is derived mean of β, H_β is derived standard deviation of β; F_γ and M_γ are the derived posteriors of γ. For ReLU, the PM, PSD and NSE values for β and γ were 0.4755, 0.0646, 0.0020; and 0.2370, 0.0642, 0.0020, respectively; for Sigmoid: 0.4476, 0.2734, 0.0087; and 1.0269, 0.2732, 0.0086, respectively; for TANSIG: 0.4718, 0.0826, 0.0026, and 1.0239, 0.0822, 0.0026, respectively. For SSLHT, the PM, PSD and NSE values for β and γ were 0.8344, 0.0567, 0.0018; and 1.0242, 0.0566, 0.0016, respectively; and for SSLHTS: 0.89825, 0.01278, 0.0004; and 1.0236, 0.0127, 0.0003, respectively. The MSE, MAE and training error values for the performance of the activation functions were ReLU: 0.1631, 0.2465, 0.1522; Sigmoid: 0.1834, 0.2074, 0.1862; TANSIG: 0.1943, 0.269, 0.1813; SSLHT: 0.0714, 0.0131, 0.0667; and SSLHTS: 0.0322, 0.0339, 0.0328, respectively. The HETAFs showed closer proximity between MSE and training error implying amelioration of overfitting and minimum error values compared to HOMAFS.

The derived Bayesian neural network estimators ameliorated the problem of overfitting with close values of Mean Square Error and training error, thus making them more appropriate in handling Neural Network models. They could be used in solving problems in machine learning.

Key words: Bayesian estimator, Estimators performance evaluation, Bayesian neural network, Monte Carlo experiment, Homogeneous activation function
Word count: 494

Table of Contents

List of Tables

List of Figures

Chapter 1
INTRODUCTION

1.1 BACKGROUND OF THE STUDY

The discipline of Artificial Neural Networks (ANN) arose from the thought of mimicking the functioning of the same human brain that was trying to solve a problem(Yang et al, 2020). ANN has contributed a noticeable benchmark in Artificial intelligence(AI) and this makes it one of the most commonly used models in deep learning.

The study of AI, often known as machine intelligence, tries to provide computer cognitive abilities so that we may teach them to learn and solve problems. Its goal is to imitate human intelligence in computer. Computer can only be designed to do particular functions of the human brain, hence artificial intelligence (AI) cannot fully mimic human intelligence. A subfield of artificial intelligence called "machine learning" enables computer to self-program using input data. AI can now solve problems based on data, thanks to machine learning. Machine learning algorithms include ANNs as an illustration.(Shen et al, 2020)

Deep learning, which develops very abstract concepts, is a complex network of neural networks with additional processing layers. They are frequently employed for difficult tasks including handwriting recognition, image categorization, and image recognition. In ANNs, the neuron is defined as a central processing unit that conducts a mathematical operation to produce one output from a group of inputs, much like the biological neuron structure(Wang et al, 2019). The weighted sum of the inputs plus the bias determines a neuron's output. If the overall signal received exceeds an activation threshold, each neuron performs a very basic function called activation. The computation of all the neurons' outputs, which is a completely predictable calculation, is the sole purpose of the entire neural network.

ANN is just a collection of approximations of mathematical functions. The terms Input layer, Hidden layer, Output layer, Weights, Bias, and Activation functions are used in relation to ANNs. There is a processor, a set of inputs, and a set of outputs. Neural networks also use this tiered strategy. The inputs make up the input layer, the processing is done by one or more hidden layers in the middle, and the results are displayed in the output layer.

The hidden layer has the magic to convert the input to the desired output. The understanding of the hidden layer requires knowledge of weights, bias, and activation functions(Qihen et al, 2023). Deep neural networks have been successfully applied to many domains, including very sensitive domains like health-care, security, fraudulent transactions and many more. These domains rely heavily on the predictions accuracy of the model and even one overconfident decision can result in a big problem. Also, these domains have very imbalanced datasets (one in a million fraudulent transactions, nearly five percent of all tests result in positive cancer results, less than one percent email is spam), and this leads to the model being over-fitted to the over-sampled class. Bayesian neural networks, on the other hand, are more robust to over-fitting and can easily learn from small datasets. The Bayesian approach further offers estimates via its parameters in the form of probability distributions(Magris and Iosifidis, 2023). At the same time, by using a prior probability distribution to integrate out the parameters, the average is computed across many models during training, which gives a regularization effect to the network, thus preventing over-fitting.

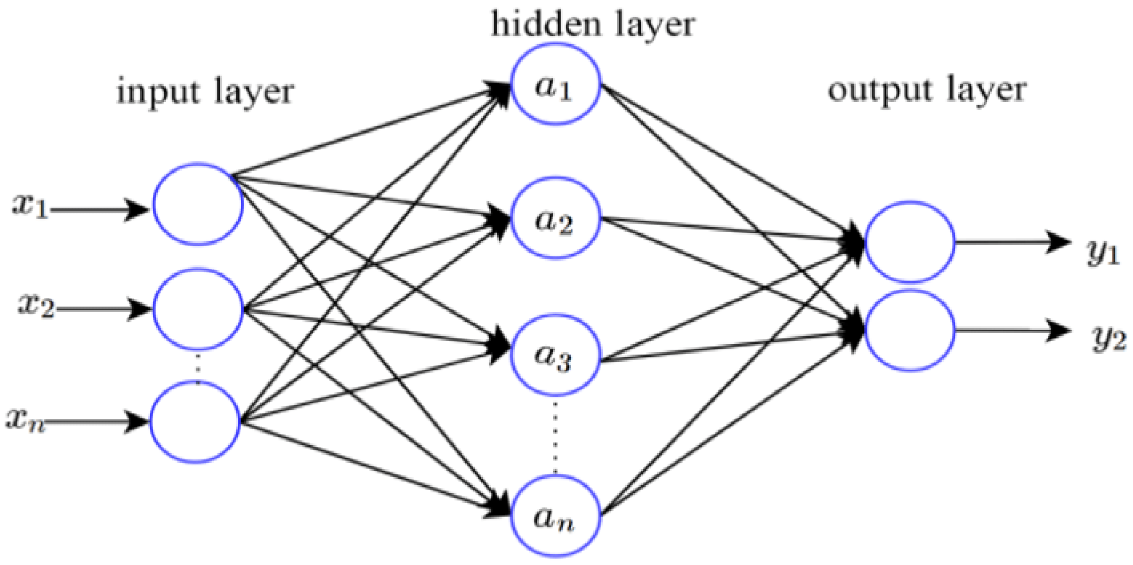

Figure 1.1: **Neural Network**
Source: Artificial Intelligence Wiki

1.2 STATEMENT OF THE STUDY

Deep learning models are designed to generalize well to any data from the problem area using training data. This is really important since we want our model to make predictions on previously unexplored datasets. When a model is over-fitted, it attempts to learn both the noise in the training data and too many details from it. The model's performance on unknown or test datasets suffers as a result. As a result, the training dataset's characteristics and patterns are not generalized by the network. Therefore, it is crucial to offer a better method of addressing over-fitting in neural network models in order to lessen the issue and make the model trustworthy for accurate prediction.

1.3 JUSTIFICATION OF THE STUDY

In supervised machine learning, over-fitting is a common problem that cannot entirely be avoided. Either the restrictions of algorithms which are too complex and require too many parameters, or the limitations of training data, which may be small or contain a lot of noise, cause it to happen. Numerous methods are created to lessen the impact of over-fitting in response to these issues. On the one hand, algorithms based on the ”early-stopping” strategy enable us stop training before learning noises to deal with noises in the training set; on the other hand, algorithms based on the ”reduce the size of network” strategy give us a method to minimize noises in the training set. On the other hand, the ”data-expansion” approach is suggested for complex models that need a lot of data to precisely adjust their hyper-parameters. Additionally, ”Regularization”-based algorithms assist us in differentiating between noise, meaningful, and meaningless information and giving each a distinct weight.

The majority of models are complex because, in most cases, the final output might be influenced by dozens or even hundreds of parameters when solving real-world situations. Instead of arbitrarily eliminating the aspects that are similar to being worthless, a well-generalized model will be more likely to take into account all possible features. To fine-tune the set of hyper-parameters, such as the weights, the increase in parameters necessitates a large amount of training data. Data thus becomes a crucial component of machine learning, particularly supervised machine learning. The majority of the time, the more training data we utilize, the more accurate the final model is. Based on the issues mentioned above, this study aims to reduce over-fitting in neural network models utilizing the Bayesian framework at various data training sets and sample sizes, both small and large.

1.4 MOTIVATION OF THE STUDY

The Multi-Layer Perceptron (MLP), which employs homogeneous activation functions (HOMAFs), has been demonstrated to be effective in terms of model precision, particularly with complicated or big data sets. In the MLP, the hyperbolic tangent, sigmoid, and symmetric saturated linear HOMAFs are frequently employed. These investigations revealed that HOMAFs have model precision constraints when used with complex systems, such as security data sets, which can result in significant mistakes and poor model performance. The body of research on reliable models for complex HOMAFs is scant. Therefore, it is necessary to create better models for these functions. The goal of this study is to develop models that are understandable, capable of modeling a wide variety of problems, comparable to the current MLP in terms of precision and generalization, as well as avoid large errors seen in previous models. To that end, heterogeneous activation functions (HETAFs) with various complexities will be combined within the Bayesian network.

1.5 AIM AND OBJECTIVES OF THE STUDY

The aim of this study is to develop a Bayesian Neural Network(BNN) model using heterogeneous activation functions to handle the problem of over-fitting in neural network model. The objectives are to:

(1) determine the weights of Bayesian Neural Network using both homogenous and heterogeneous activation functions.

(2) study the asymptotic behaviour of BNN using these activation functions.

(3) examine the performance of BNN model under different activation functions.

1.6 SIGNIFICANCE OF THE STUDY

Artificial neural networks (ANNs) have recently been successfully used in a wide variety of problem domains, including those in the fields of engineering, geology, finance, medicine, and other physical and biological sciences, as well as water resources and environmental research. The attempt to simulate the capabilities of the human brain through these networks is what has people so excited. Neural networks are intriguing from a statistical standpoint due to their potential application in prediction and classification issues.

In reality, it has been applied to a huge range of applications that typically involve statistical approaches. ANNs are used to solve issues that were typically resolved using traditional statistical techniques like regression, discriminant analysis, logistic regression, Bayes analysis, multiple regression, and ARIMA time-series models. All areas of statistics have seen the introduction of neural networks, including econometrics and econometric issues.

1.7 ORGANISATION OF THE DISSERTATION PRESENTED

The thesis consists of five chapters. Following this chapter is the chapter which reviews literature on Artificial Neural Network and past research work in this field. Chapter three contains theoretical framework: Homogeneous artificial neural network model, Heterogeneous artificial neural network model and Bayesian estimationof Neural Network parameters. Analysis of data and interpretations are discussed in chapter four while chapter five presents the summary of findings, conclusions and recommendations along with research contribution to knowledge and suggestions for further research.

Chapter 2
LITERATURE REVIEW

2.1 CHAPTER OVERVIEW

This chapter reviews literature on neural network weights, over-fitting and also on neural network and Bayesian neural network models. Few terminologies used in this work are also explained.

2.2 DEFINITION OF SOME TERMS

(1) **Overfitting:**A machine learning model overfits when it matches the training data too closely, losing the capacity to categorize and predict test data.

(2) **Activation function:** The activation function of a node in artificial neural networks determines the node's output given an input or group of inputs.

(3) **Weights:** Weights are the actual values that are associated with each input or characteristic, and they communicate the significance of that input or feature in predicting the outcome.

(4) **Bayesian Approach:** The first step in the Bayesian approach is to define a prior distribution for the parameters that need to be estimated. Without taking into account the dataset upon which the model is estimated, the prior reflects the knowledge held by the researcher. By considering previous data that is not in the sample, a prior can be created in a time series setting.

2.3 WEIGHTS AND BIASES

Weights in an ANN are the most important factor in converting an input to impact the output. This is similar to slope in linear regression, where a weight is multiplied to the input to add up to form the output. Weights are numerical parameters which determine how strongly each of the neurons affects another.

For a typical neuron, if the inputs are x_1, x_2, and x_3, then the synaptic weights to be applied to them are denoted as w_1, w_2, and w_3 and the output is

$$y = f(x) = \sum x_i w_i \tag{2.1}$$

where i is from 1 to the number of inputs. Simply, this is a matrix multiplication to arrive at the weighted sum. Bias is like the intercept added in a linear equation. It is an additional parameter which is used to adjust the output along with the weighted sum of the inputs to the neuron. The processing done by a neuron is thus denoted as:

$$output = sum(weights * inputs) + bias \tag{2.2}$$

A function is applied on this output and is called an activation function. The input of the next layer is the output of the neurons in the previous layer, as shown in the figure 2.1.

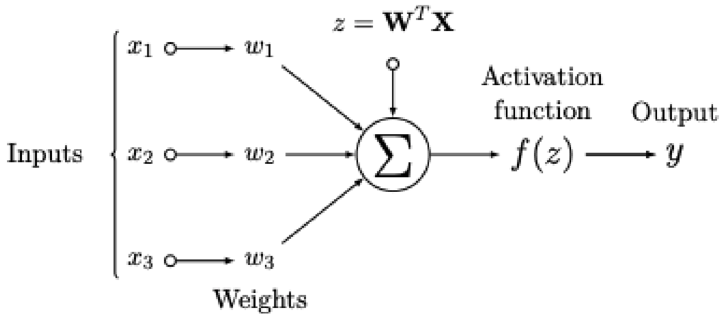

Figure 2.1: **Weights and Activation Function**
Source: Artificial Intelligence Wiki

2.4 ACTIVATION FUNCTIONS

The activation functions are primarily responsible for the abstraction of neural network computation. A mathematical function called an activation function adds the magic of neural network processing by converting an input to an output. Without activation functions, neural networks will operate similarly to linear functions. A polynomial of degree one is a linear function, i.e, a straight line. The majority of the issues that neural networks attempt to address are complicated and nonlinear in nature. The activation functions are utilized to achieve the nonlinearity. A nonlinear function's graph is curved, which increases complexity. As true universal function approximators, neural networks are given the nonlinearity property through activation functions.

2.5 OVERFITTING IN MACHINE LEARNING

When a statistical model does not correctly anticipate the outcomes of testing data, it is said to be overfitted. A model starts learning from the noise and erroneous data entries in our data set after receiving such a large amount of training data. Additionally, when testing using test data yields high variance. Because of the noise and excess details, the model fails to appropriately identify the data. Non-parametric and non-linear approaches are the root causes of overfitting since they provide machine learning algorithms more freedom to create the model depending on the dataset, which can lead to the creation of highly irrational models.

2.5.1 WHY DOES OVERFITTING OCCUR

Only if the machine learning model generalizes to all varieties of data within its domain then precise predictions is obtained. When a model can't generalize and instead fits too closely to the training dataset, this is known as overfitting. Several factors contribute to overfitting, including:

(1) Insufficient data samples and a lack of sufficient training data prevent an accurate representation of all possible input data values.

(2) Large volumes of unimportant information, or noisy data, are present in the

training data.

(3) The model spends too much time training with just one sample set of data.

(4) Due of the high model complexity, the training data's noise is learned.

2.5.2 HOW TO PREVENT OVERFITTING

By growing and diversifying training data set, as well as by employing other data science techniques, such as the ones listed below, overfitting can be avoided.

(1) Early Stopping: Early stopping puts the machine learning model's training phase on hold before it can pick up on the data noise. The timing must be perfected, or the model won't produce reliable results.

(2) Prunning: When modellling, a number of features or parameters that have an effect on the final prediction may be found and removing unimportant features from the training set and selecting the most crucial ones is done. This process is known as feature selection or pruning.

(3) Regularization: A group of training/optimization methods called regularization aim to decrease overfitting. By ranking variables according to relevance, these techniques attempt to exclude those aspects that have no impact on the results of the predictions.

(4) Ensembling: Predictions from various different machine learning algorithms are combined in assembling. Because they frequently produce erroneous results, some models are referred to as weak learners. For more accurate results, ensemble methods mix all the weak learners. To examine sample data and select the most precise results, they use a variety of models. Bagging and boosting are the two primary ensemble techniques. While bagging trains them concurrently, boosting trains various machine learning models one at a time until the desired outcome is achieved.

(5) Data Augumentation: When using data augmentation, a machine learning

technique, the sample data is slightly altered each time the model uses it. Small adjustments to the input data can be made to achieve this. Data augmentation, when used sparingly, makes the training sets seem special to the model and stops it from learning out their characteristics.

2.6 THEORETICAL REVIEW

During the past five years the Bayesian deep learning community has developed increasingly accurate and efficient approximate inference procedures that allow for Bayesian inference in deep neural networks. A study was carried out to cast doubt on the current understanding of Bayes posteriors in popular deep neural networks, demonstrated through careful MCMC sampling that the posterior predictive induced by the Bayes posterior yields systematically worse predictions compared to simpler methods including point estimates obtained from Stochastic Gradient Descent (SGD) (Wenzel, 2020). The findings showed that predictive performance is improved significantly through the use of a "cold posterior" that overcounts evidence. Such cold posteriors sharply deviate from the Bayesian paradigm but are commonly used as heuristic in Bayesian deep learning papers.

Udomboso (2013) revealed that for transfer functions to map the input layer of the statistical neural network model to the output layer perfectly, they must lie within bounds that characterize probability distributions. The heterogeneous transfer function is established as a Probability Distribution Function (p.d.f), and its mean and variance are shown.
Deep neural networks could be grounded and perform better, thanks to Bayesian inference. It promises to be resistant to overfitting, to streamline the training process and the hyperparameter space, and to offer a calibrated measure of uncertainty that can improve agent exploration, forecast fairness, and decision-making. Bayesian inference is made possible by Markov Chain Monte Carlo (MCMC) techniques, which produce samples from the posterior distribution of the model's parameters. For large-scale deep learning problems, sampling approaches have so far performed worse than optimization methods despite the theoretical benefits of Bayesian inference and the resemblance between MCMC and optimization meth-

ods.

Many specialists and academics have been researching overfitting issues in the regression process since the emergence of machine learning technology, and they have produced a number of research findings. For instance, Hinton et al. (2012) and Srivastava et al. (2014) originally suggested the dropout approach in machine learning to overcome the overfitting problem. Li et al. (2002) is also among those who proposed to solve the overfitting problem in the algorithm. To prevent the occurrence of the overfitting issue, Chen et al. (2014) modified algorithm parameters and iterations by Singular Value Decomposition (SVD), and other researchers such as Shen et al. (2020), Qin and Li (2006), and Yang et al. (2020) have also conducted relevant research. Many academics and engineering professionals have researched of this issue on overfitting issue with the current deep learning CNN model. A number of solutions, including data expansion enhancement, regularization, discarding, and early stop method, have been put forth by Yang et al. (2020), Cha et al. (2019), Ashiquzzaman et al. (2018), Gong et al. (2017), Cheng et al. (2019), Zhabg et al. Gong et al. (2017) proposed to improve the CNN based on the immune system in response to issues including lengthy training times and overfitting of the CNN model, although the accuracy of this upgraded network model on the test set is not great (only 81.6%). Yang et al. (2020) introduced an attribute reduction approach based on visual ranging to handle training data sets. This algorithm addressed the issue of network overfitting by replacing CNN fixed weights with a Bayesian weight factor distribution, however its scope of use is relatively constrained.

Zhang et al. (2018) studied the typical reinforcement learning agent in-depth during the model's training, then performed a comprehensive debate on reinforcement learning overfitting and generalized it from the viewpoint of inductive bias. Cha et al. (2019) use synthetic mammograms as training data enhancement, which is a common technique, to address the overfitting issue of the deep learning breast mass detection system for synthetic pictures.

The issue of deep learning overfitting has not been addressed particularly well by research on deep learning behavior. A model prediction averaging approach based on dropout double probability weighted pooling was proposed by Cheng et al. (2019). It effectively lowers the error rate and prevents overfitting, however the algorithm's pace of convergence slows down after double probability is added. Gonzalez et al. (2018) introduced the particle swarm optimization (PSO) algorithm to the CNN model design in order to decrease back propagation of error, avoid lag error and image over-combination, and improve convergence speed, but the PSO is not theoretically supported for the CNN weight update. While increasing the effectiveness of CNN's data processing, other techniques have more or less "side effects" (Qi et al., 2021).

By scaling SG-MCMC approaches to big models and datasets, the work aims to make Bayesian inference useful for deep learning (Heek and Kalchbrenner, 2019). The contributions listed in their work can be divided into three groups. They was suggested the use of the Adaptive Thermostat Monte Carlo (ATMC) sampler, which provides better convergence and stability. The amount of momentum and noise applied to each model parameter is dynamically changed by ATMC. Second, a second order numerical integration technique that is already in use and required for the ATMC sampler was enhanced. The ResNet++ network was created by taking the original ResNet architecture, removing BatchNorm, and adding SELUs, Fixup initialization, and weight normalization. This is because ATMC, like other SG-MCMC samplers is not directly compatible with stochastic regularization methods like batch normalization (BatchNorm), Dropout, and weight normalization. The Cifar10 benchmark and the large-scale ImageNet benchmark were tested using a ResNet architecture without batch normalization, and the results showed that ATMC outperformed a strong optimization baseline in terms of classification accuracy and test log-likelihood. On the basis of the training data, it was demonstrated that ATMC is inherently resistant to overfitting and that it offers a more accurate measure of uncertainty than the optimization baseline.

Popular samplers for approximate inference are stochastic gradient Markov Chain Monte Carlo (SGHMC) techniques, however they are typically biased. The unpleasant truth that SGHMC and other widely used samplers cannot be modified via rejection because their acceptance probability are zero was brought to light by Fortuin (2020) in his study. Gradient-Guided Monte Carlo, which generalizes

Stochastic Gradient Langevin Dynamics (SGLD) and Hamiltonian Monte Carlo, is one example of a sampler with realizable reverse trajectories (HMC). Then, a method for creating precise posterior samples solely utilizing stochastic gradients was developed. This method involved computing the acceptance probabilities of GGMC across a number of stages.

The posteriors over neural network weights are high dimensional and multimodal. Each mode typically characterizes a meaningfully different representation of the data. A research was conducted to replace the traditional decreasing stepsize schedule in SG-MCMC with a cyclical variant (Zhang, 2020). This method was referred to as Cyclical Stochastic Gradient MCMC (cSG-MCMC) to automatically explore complex multimodal distributions. This approach is particularly compelling for Bayesian deep learning, which involves rich multimodal parameter posteriors corresponding to meaningfully different representations. A cyclical stepsize schedule was proposed, where larger steps discover new modes, and smaller steps characterize each mode. Cyclical SG-MCMC methods provided more accurate uncertainty estimation by capturing more diversity in the hypothesis space corresponding to settings of model parameters. Furthermore, extensive experimental results were obtained, including ImageNet to demonstrate the effectiveness of cyclical SG-MCMC in learning complex multimodal distributions, especially for fully Bayesian inference with modern deep neural networks.

The key distinguishing property of a Bayesian approach is marginalization instead of optimization, where represent solutions given by all settings of parameters weighted by their posterior probabilities, rather than bet everything on a single setting of parameters. Neural networks are typically underspecified by the data, and can represent many different but high performing models corresponding to different settings of parameters, which is exactly when marginalization will make the biggest difference for accuracy and calibration (Yang et al., 2020).

Wilson and Izmailov (2020) showed that deep ensembles provide an effective mechanism for approximate Bayesian marginalization, and propose a related approach that further improves the predictive distribution by marginalizing within basins of attraction, without significant overhead. The prior over functions was also investigated implied by a vague distribution over neural network weights, explaining the generalization properties of such models from a probabilistic perspective. From this perspective, results that have been presented as mysterious and distinct to neural network generalization were explained, such as the ability to fit images with random labels, and show that these results can be reproduced with Gaussian processes. Also, this multimodal approach to Bayesian model averaging, MultiSWAG can entirely alleviate double descent to enable monotonic performance improvements with increases in model flexibility as well significant improvements in generalization accuracy and log-likelihood over SGD and single basin marginalization.

Recently, there has been much attention in the use of machine learning methods, particularly deep learning for stock price prediction. A major limitation of conventional deep learning is uncertainty quantification in predictions which affect investor confidence. Bayesian neural networks feature Bayesian inference for providing inference (training) of model parameters that provides a rigorous methodology for uncertainty quantification in predictions. Markov Chain Monte Carlo (MCMC) sampling methods have been prominent in implementing inference of Bayesian neural networks; however certain limitations existed due to a large number of parameters and the need for better computational resources. The COVID-19 pandemic had a drastic impact in the world economy and stock markets given different levels of lockdowns due to rise and fall of daily infections. It is important to investigate the performance of related forecasting models during the COVID-19 pandemic given the volatility in stock markets.(Shen et al., 2020)

Id and He (2021) used novel Bayesian neural networks for multi-step-ahead stock price forecasting before and during COVID-19. The pre-COVID-19 datasets were examined if they are useful for modelling stock price forecasting during COVID-19. The results indicated that due to high volatility in the stock-price during COVID-19, it is more challenging to provide forecasting. However, it was discovered that Bayesian neural networks could provide reasonable predictions with uncertainty quantification despite high market volatility during the first peak of the COVID-19 pandemic.

Cranmer et al. (2021) presented a neural network that, trained only on short time series of the orbits in compact planetary systems, not only improves on long-term predictions of previous models based on engineered features but also significantly reduced the model bias and improved generalization beyond the training set. The model was designed as a Bayesian neural network (BNN) which naturally incorporates confidence intervals into its instability time predictions, accounting for model uncertainty as well as the intrinsic uncertainty due to the chaotic dynamics. The model, trained directly from short N-body time series of raw orbital elements, is more than two orders of magnitude more accurate at predicting instability times than analytical estimators, while also reducing the bias of existing machine learning algorithms by nearly a factor of three. Despite being trained on compact resonant and near-resonant three-planet configurations, the model demonstrated robust generalization to both nonresonant and higher multiplicity configurations, in the latter case outperforming models fitted to that specific set of integrations. The model computed instability estimates up to 105 times faster than a numerical integrator, and unlike previous efforts provides confidence intervals on its predictions. Finally, unlike previous machine learning models based on decision trees this model is differentiable. That is, instability times can be extracted from the model estimates of the derivatives of the predicted with respect to the parameters defining the orbital configuration in question. Characterizing drug–protein interactions (DPIs) is crucial to the high-throughput screening for drug discovery. The deep learning-based approaches have attracted attention because they can predict DPIs without human trial and error. However, because data labeling requires significant resources, the available protein data size is relatively small, which consequently decreases model performance.(Cranmer et al. ,2021)

Kim et al. (2021) proposed two methods to construct a deep learning framework that exhibits superior performance with a small labeled dataset. At first, transfer learning was used in encoding protein sequences with a pretrained model, which trains general sequence representations in an unsupervised manner. Secondly, a Bayesian neural network was utilized to make a robust model by estimating the data uncertainty. The resulting model performs better than the previous baselines at predicting interactions between molecules and proteins. It was also revealed that the quantified uncertainty from the Bayesian inference is related to confidence and can be used for screening DPI data points.

Radev (2021) addressed the problem with a novel combination of epidemiological modeling with specialized neural networks. The approach entailed two computational phases: In an initial training phase, a mathematical model describing the epidemic was used as a coach for a neural network, which acquired global knowledge about the full range of possible disease dynamics. In the subsequent inference phase, the trained neural network processed the observed data of an actual outbreak and infered the parameters of the model in order to realistically reproduce the observed dynamics and reliably predict future progression. Moreover, since the method was fully Bayesian, it was designed to incorporate all available prior knowledge about plausible parameter values and returns complete joint posterior distributions over these parameters. Application of the method to the early Covid-19 outbreak phase in Germany demonstrated that reliable probabilistic estimates for important disease characteristics coul be obtained, such as generation time, fraction of undetected infections, likelihood of transmission before symptom onset, and reporting delays using a very moderate amount of real-world observations.

Nan (2021) applied a Bayesian neural network (BNN) algorithm to build a predictive model for occupant thermal preference using the ASHRAE Global Thermal Comfort Database II. The results showed that the BNN model outperformed conventional thermal comfort models such as Predicted Mean Vote (PMV) and adaptive comfort model. The BNN model tends to produce more confident "prefer cooler" predictions with high possibility and low uncertainty. In contrast, the BNN model produced less certain predictions for "prefer no change" and "prefer warmer" across all occupants. The findings suggested that linking occupants' subjective evaluation measures and window opening/closing behavior to thermal comfort modeling effectively improved predictive performance.
With a focus on solutions depending on variational inference and the use of natural gradients, Magris and Iosifidis (2023) provided both traditional and contemporary methodologies for Bayesian inference. They also talked about the use of manifold optimization, a cutting-edge method for Bayesian learning. They looked at the distinguishing characteristics of all the methods that were described and provided pseudo-codes for their application, giving close attention to the practical details like computing gradients.

Using mean field variational Bayesian Neural Networks (BNNs), Qihan et al. (2023) investigated the representational capacity of such BNNs by identifying the notions that are less likely to be encoded by the BNN. It has been observed and researched that, in the knowledge representation of an adequately trained neural network, a very small collection of interacting concepts typically develop, and such concepts can accurately describe the network output.Because of this, their research shown that BNNs are less likely to encode complicated concepts than conventional deep neural networks (DNNs).

Chapter 3
RESEARCH METHODOLOGY

3.1 CHAPTER OVERVIEW

This chapter explains the Bayesian Neural Network model with the use of activation function. Other areas discussed include data generation for the study, Evaluation Metrics for the simulation study and description of the real life data used for the study.

3.2 DIFFERENT ACTIVATION FUNCTIONS

There are many activation functions available for a neural network to use. These are called homogenous transfer functions(HOTFs). They are

(1) Sigmoid
(2) hyperbolic tangent(TANH)
(3) hyperbolic tangent sigmoid(TANSIG)
(4) symmetric saturating linear(SATLINS)
(5) Rectified Linear Unit(ReLU)

The choice of the activation functions used in this research is based on preliminary investigations of ranking of activation functions by the Error Variance(Udomboso, 2014). The authour observed that the best performances of activation functions came from:

(1) Hyperbolic Tangent activation function (TANH)

The Tanh function (also 'tanh' and 'TanH') is another name for the hyperbolic tangent activation function. It shares the same S-shape with the sigmoid activation function. Any real value may be used as an input, and the function returns values between -1 and 1.

$$g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.1}$$

It squashes a real-valued number to the range [-1, 1].

(2) Hyperbolic Tangent Sigmoid activation function (TANSIG)
Another common and mostly utilized activation function is the tanh function. This is a nonlinear function, characterized in the scale of values (-1, 1). One thing to make clear is that the gradient is better for tanh than sigmoid (the derivatives are steeper). Settling between sigmoid and tanh will be based on the gradient strength prerequisite. Like the sigmoid, tanh also has the missing slope constraint. The function is specified by the formula:

$$g(x) = \frac{2}{1 - e^{-2x}} - 1 \tag{3.2}$$

(3) Symmetric Saturating Linear activation function (SATLINS)
SATLINS function appears to be also capable of use as the activation function for nonlinear system modeling aside providing the smaller median of the final error function value over all tested numbers of neurons in topologies.

$$g(x) = \begin{cases} -1, & x < -1 \\ x, & -1 \leq x \leq 1 \\ 1, & x > 1 \end{cases} \tag{3.3}$$

respectively.
(4) Sigmoid activation function
A mathematical function called the sigmoid function produces an S-shaped curve known as a sigmoidal curve. The earliest and most popular activation function is this one. This makes the model logistic and compresses the input to any number

between 0 and 1. This function is known as a special case of logistic function defined by the following formula:

$$f(x) = \frac{1}{1+e^{-x}} \tag{3.4}$$

(5) Rectified Linear Unit (ReLU) activation function
One activation function that is frequently used is the rectified linear unit (ReLU). The function is defined by the following formula

$$g(x) = \begin{cases} 0, \quad when \quad x < 0 \\ \\ x, \quad when \quad x >= 0 \end{cases} \tag{3.5}$$

The scale of the result is between 0 and infinity. ReLU finds usage in computer vision and speech identification using deep neural networks.

3.3 HETEROGENEOUS ACTIVATION FUNCTIONS

Further investigation conducted on the choice of convolution revealed that the best convolutions arose from the best three individual transfer functions. On the whole, in the overall ranking, it was found out that best performance was obtained in the convolution of the Symmetric Saturating Linear transfer function and the Hyperbolic Tangent transfer function (SATLINS-TANH), followed by the convolution of the Symmetric Saturating Linear transfer function and the Hyperbolic Tangent Sigmoid activation function (SATLINS-TANSIG)(Udomboso, 2014). The summary of the derived function is given as:

(1) Symmetric Saturating Linear Hyperbolic Tangent (SSLHT)

$$g(x) = \begin{cases} log(\frac{e^x - e^{-x}}{e^x + e^{-x}}), \quad for \quad x < -1 \\ \\ (x+1)log(e^x + e^{-1})^{-1}, for - 1 \leq x \leq 1 \\ \\ log(\frac{e^x - e^{-x}}{e^x + e^{-1}}), \quad for \quad x > 1 \end{cases} \tag{3.6}$$

The function above is the derived HETTF for the Symmetric Saturated Linear transfer function and the Hyperbolic Tangent transfer function.

(2) Symmetric Saturating Linear Hyperbolic Tangent Sigmoid (SSLHTS)

$$g(x) = \begin{cases} \sum_{p=1}^{\infty} \frac{e^{-2px}}{p} - \sum_{p=1}^{\infty} \frac{e^{2px}}{p} - (x + r), \quad for\ x < -1 \\ (2x^2 + 3x + 1/2) - x(\sum_{p=1}^{\infty} \frac{e^{-2px}}{p} - \sum_{p=1}^{\infty} \frac{e^{-2px}}{p}), for - 1 \le x \le 1 \\ (x-1) - \sum_{p=1}^{\infty} \frac{e^{-2px}}{p} + \sum_{p=1}^{\infty} \frac{e^{-2px}}{p}, \quad for \quad x > 1 \end{cases} \tag{3.7}$$

The function above is the derived HETTF for the Symmetric Saturated Linear transfer function and the Hyperbolic Tangent Sigmoid transfer function. The derivation of the two heterogeneous activation functions are discussed below

(i) $satslins \otimes tanh = f1(n) \otimes f2(n)$

(i1) $satslins \otimes tansig = f1(n) \otimes f2(n)$

3.4 SYMMETRIC SATURATING LINEAR AND HYPERBOLIC TANGENT

(i) Let

$$f(n) = f_1(n) \otimes f_2(n) = \int_a^b f_1(n-m) f_2(m) dm \tag{3.8}$$

For $n < -1, f_1(n) = -1$, which implies also that $f_1(n-m) = -1$.

Also from (3.1.12),

$$f_2(m) = \frac{e^n - e^{-n}}{e^n + e^{-n}} - \frac{d(e^n - e^{-n})}{e^n + e^{-n}} \tag{3.9}$$

Therefore,

$$f_1(n) \otimes f_2(n) = \int_r^{-p} (-1) \left(\frac{d|e^n - e^{-n}|}{e^n + e^{-n}} \right) dm, r < n < -1 \tag{3.10}$$

$$= -\int_{-1}^{n} \frac{d(e^n - e^{-n})}{e^n + e^{-n}} dm = -log(e^n + e^-n)$$

$$= log(e^n + e^-n)^{-1}$$

$$= log\left(\frac{d(e^r - e^{-r})}{e^n + e^{-n}}\right) \tag{3.11}$$

(ii) Similarly, for $-1 \leq n \leq 1, f_1(n) = n$, which implies that $f_1(n-m) = n - m$, such that $-1 \leq m < n$.
Therefore,

$$f_1(n) \otimes f_2(n) = \int_{-1}^{n} f_1(n-m) f_2(m) dm \tag{3.12}$$

(iii) Also, for $n > 1, f_1(n) = a = 1$. This implies that $f_1(n-m) = 1$.

Therefore,

$$\begin{aligned} f_1(n) \otimes f_2(n) &= \int_1^n f_1(n-m) f_2(m) dm \\ &= \int_1^n 1 \left(\frac{e^n - e^{-n}}{e^n + e^{-n}}\right) dm \\ &= \int_1^n \frac{d(e^n - e^{-n})}{e^n + e^{-n}} dm \\ &= log(e^n + e^-n)|_1^n \end{aligned}$$

$$= log\left(\frac{e^n + e^-n}{e + e^{-1}}\right) \tag{3.13}$$

The summary of he derived function is given as

$$f(n) = \begin{cases} log\left(\frac{e^r + e^{-r}}{e + e^{-1}}\right), & for \quad n < -1 \\ (n+1) log(e + e^{-1})^{-1}, & for \quad -1 \leq n \leq 1 \\ log\left(\frac{e^n + e^{-n}}{e + e^{-1}}\right), & for \quad n > 1 \end{cases} \tag{3.14}$$

Equation (3.14) is the derived HETTF for the Symmetric Saturated Linear activation function and the Hyperbolic Tangent transfer function.

3.5 SYMMETRIC SATURATING LINEAR AND HYPERBOLIC TANGENT-SIGMOID

(i) For $n < -1$, $f_1 = a = -1$. This implies that $f_1(n-m) = -1$.

$$f_1(m) = \frac{2}{1-e^{-2m}} - 1 - \frac{1+e^{-2m}}{1-e^{-2m}}$$

Let

$$f(n) = f_1(n) \otimes f_2(n) = \int_{-r}^{n} f_1(n-m) f_2(m)\, dm$$

$$f_1(n) \otimes f_2(n) = \int_{-r}^{n} (-1)\left(\frac{2}{1-e^{-2m}} - 1\right) dm$$

$$= \int_{-r}^{n} \left(1 - \frac{2}{1-e^{-2m}}\right) dm - 2\int_{-r}^{n} \frac{1}{1-e^{-2m}}\, dm$$

$$= m|_{-p}^{n} - 2\int_{-r}^{n} (1+e^{-2m})^{-1}\, dm = (n+r) - 2\int_{-r}^{n} (1+e^{-2m}+e^{-4m}+\dots)\, dm$$

$$= (n+r) - 2\left[m - \frac{e^{-2m}}{2} - \frac{e^{-4m}}{4} - \dots\right]_{-r}^{n}$$

$$= (n+r) - 2\left((n - \frac{e^{-2n}}{2} - \frac{e^{-4n}}{4} - \frac{e^{-6n}}{6} - \dots)\right) - \left[-2\left((-r - \frac{e^{-2r}}{2} - \frac{e^{-4r}}{4} - \frac{e^{-6r}}{6} - \dots)\right)\right]$$

$$= \left(-n + e^{-2n} + \frac{e^{-4n}}{4} + \frac{e^{-6n}}{6} + \frac{e^{-8n}}{8} + \dots\right) + \left(-r + e^{-2r} - \frac{e^{-4r}}{4} - \frac{e^{-6r}}{6} - \frac{e^{-8r}}{8} - \dots\right)$$

$$= (2n - n - 2 - 1) - \sum_{p=1}^{\infty} \frac{e^{-2pm}}{p} + \sum_{p=1}^{\infty} \frac{e^{-2p}}{p}$$

$$= (n-1) - \sum_{p=1}^{\infty} \frac{e^{-2pm}}{p} + \sum_{p=1}^{\infty} \frac{e^{-2p}}{p} \tag{3.15}$$

The summary of the derived function is given as

$$f(n) = \begin{cases} \sum_{p=1}^{\infty} \frac{e^{-2pn}}{p} + \sum_{p=1}^{\infty} \frac{e^{-2pr}}{p} - (n+r), & n < -1 \\ \\ (2n^2 + 3n + \frac{1}{2}) - n \left(\sum_{p=1}^{\infty} \frac{e^{-2pm}}{p} + \sum_{p=1}^{\infty} \frac{e^{-2p}}{p} \right), & -1 \leq n \leq 1 \\ \\ (n-1) \sum_{p=1}^{\infty} \frac{e^{-2pm}}{p} + \sum_{p=1}^{\infty} \frac{e^{-2p}}{p}, & n > 1 \end{cases} \tag{3.16}$$

Equation (3.16) is the derived HETTF for the Symmetric Saturated Linear transfer function and the Hyperbolic Tangent Sigmoid transfer function.

3.6 BAYESIAN NEURAL NETWORK MODEL

The statistical neural network model proposed by Anders (1996) is given as

$$y = f(X, w) + e \tag{3.17}$$

where y is the dependent variable, X = $(x_0 = 1, x_1, ..., x_n\ )$ is a vector of independent variables,
w = (α, β, γ) is the network weight,
α is the weight of the input unit,
β is the weight of the hidden unit, and
γ is the weight of the output unit, and
e_i is the stochastic term that is normally distributed (that is, $e_i \sim N(0, \sigma^2 I_n)$).
The assumptions of the statistical neural network are the same as the usual regression models. Basically, f(X,w) (Anders, 1996) is the artificial neural network function, expressed as

$$f(X, w) = \alpha X + \sum_{h=1}^{H} \beta_h g(\sum_{i=0}^{l} \gamma_{hi} x_i) \tag{3.18}$$

where g(.) is the homogenous transfer function(HOMTF),
X is a set of input data and
y is a set of output data.
h = 1, 2,...,H is the number of hidden units.

i = 0, 1,...,I is the number of input units.
Putting eqn(3.2) in eqn(3.1) gives the homogenous SNN (HOMSNN) model.
Given a convoluted form of the artificial neural network function given by Anders (1996) using product convolution:

$$f(X,w) = \alpha X + \sum_{h=1}^{H} \beta_h [g_1(\sum_{i=0}^{l} \gamma_{hi} x_i) g_2(\sum_{i=0}^{l} \gamma_{hi} x_i)] \tag{3.19}$$

where $g_1(.)$ and $g_2(.)$ are transfer functions, which is generally known as heterogenous transfer function(HETTF).
Equation (3.3) above is called the heterogenous SNN (HETSNN) model.
Consider random variables $y_1, \cdots, y_N$ such that :

$$y_i = \sum_{j=1}^{M} \beta_j \Psi(X_i^T \gamma_j) + e_i = \beta^T \eta_i(\gamma) + e_i, \tag{3.20}$$

where;

$$\begin{aligned} \beta &= (\beta_1, \cdots, \beta_M)^T; \\ X_i &= (x_{i1}, \cdots, x_{ip})^T; \text{ and} \\ \eta_i(\gamma) &= \Psi(X_i^T \gamma_1), \cdots, \Psi(X_i^T \gamma_M))^T, \ i = 1, \cdots, N \end{aligned}$$

Also, the errors e_i are assumed to be iid $N(0, \sigma^2 I)$. The regression part of the model can be identified with the feedforward neural network. The X_i are the inputs, M is the number of hiddden nodes (here assumed to be known), and β_j are the weighs attached to the inputs for node $j (j = 1, \cdots, M)$, and Ψ is the activation function. We write $\theta = (y_1, \cdots, y_N)^T$.
At the first stage of the hierarchical prior, $\beta, \gamma_1, \cdots, \gamma_M$ are mutually independent with $\beta \sim N(\mu_\beta 1_M, \sigma_\beta^2 I_M)$ and $\gamma_1, \cdots, \gamma_M \overset{iid}{\sim} N(\mu_\gamma, S_\gamma)$.
At the second stage, the prior parameters $\mu_\beta, \mu_\gamma, \sigma^2, \sigma_\beta^2$, and S_γ are mutually independent with $\mu_\beta \sim N(a_\beta, A_\beta), \mu_\gamma \sim N_p(a_\gamma, A_\gamma), \sigma^2 \sim IG(c_\sigma/2, c_\sigma C_\sigma/2), \sigma_\beta^2 \sim IG(c_\beta/2, c_\beta C_\beta/2)$, and $S_\gamma \sim IW(c_\gamma, c_\gamma^{-1} C_\gamma^{-1})$. Here, IG and IW denote the inverse gamma and inverse Wishart distributions.

Specifically, σ_β^2 has pdf

$$f(\sigma_\beta^2) \propto \exp\left(-\frac{c_\beta C_\beta}{2\sigma_\beta^2}\right) (\sigma_\beta^2)^{-c_\beta/2-1} \tag{3.21}$$

and S_γ has pdf

$$f(S_\gamma) \propto |S_\gamma|^{-(c_\gamma+p+1)/2} \exp\left[-\frac{1}{2}tr(S_\gamma^{-1}c_\gamma C_\gamma)\right] \tag{3.22}$$

The selection of priors is usually problem-specific. Ideally, one would like to elicit these priors from past history. For this study, this was done for $\mu_\beta, \mu_\gamma, \sigma^2, \sigma_\beta^2$, and S_γ.

The joint posterior of $\theta, \gamma, \beta, \mu_\beta, \mu_\gamma, \sigma^2, \sigma_\beta^2$, and S_γ, given y, is:

$$f(\theta, \gamma, \beta, \mu_\beta, \mu_\gamma, \sigma^2, \sigma_\beta^2, S_\gamma|y) \tag{3.23}$$

$$\propto \prod_{i=1}^{N} \left[\exp\left\{\phi_i^{-1}(\theta_i y_i - \kappa(\theta_i))\right\}\right] (\sigma^2)^{-N/2} \tag{3.24}$$

$$\times \exp\left[-(2\sigma^2)^{-1}\sum_{i=1}^{N}(\theta_i - \eta_i^T(\gamma)\beta)^2\right]$$

$$\begin{aligned}
\times &\quad (\sigma_\beta^2)^{-M/2}\exp[-\|\beta - \mu_\beta 1_M\|^2/(2\sigma_\beta^2)] \\
\times &\quad |S_\gamma|^{M/2}\exp\left[-\frac{1}{2}\sum_{j=1}^{M}(\gamma_j - \mu_\gamma)^T S_\gamma^{-1}(\gamma_j - \mu_\gamma)\right] \\
\times &\quad \exp\left[-(2A_\beta)^{-1}(\mu_\beta - a_\beta)^2 - \frac{1}{2}(\mu_\gamma - a_\gamma)^T A_\gamma^{-1}(\mu_\gamma - a_\gamma)\right] \\
\times &\quad \exp[-(c_\sigma C_\sigma)/(2\sigma^2)](\sigma^2)^{-c_\sigma/2-1} \\
\times &\quad \exp[-(c_\beta C_\beta)/(2\beta^2)](\sigma_\beta^2)^{-c_\beta/2-1} \\
\times &\quad |S_\gamma|^{-(c_\gamma+p+1)/2}\exp\left[-\frac{1}{2}tr(S_\gamma^{-1}c_\gamma C_\gamma)\right].
\end{aligned}$$

One of the objectives is to find the posterior distribution of θ given y, and use this for finding the predictive distribution of the future unobserved y_u given the observed y. This is obtained by the formula:

$$f(y_u|y) = \int f(y_u|\theta)f(\theta|y)d\theta \tag{3.25}$$

However, the posterior $f(\theta|y)$ is analytically intractable, because its evaluation requires high-dimensional numerical integration. Thus, the Gibbs sampling is used to generate samples from this posterior. Some of the full conditionals needed in this proce dure are available only up to unknown normalizing constants, and the study used either a regular Metropolis-Hastings algorithm or the adaptive rejection sampling procedure of Gilks and Wild (1992) to sample from these conditionals. These full conditionals are as follows:

(a) $\beta|\theta, \gamma, \mu_\beta, \mu_\gamma, \sigma^2, \sigma_\beta^2, S_\gamma, y \sim N(K_\beta, H_\beta)$
where

$$K_\beta = (\sigma^{-2}\sum_{i=1}^{N}\eta_i(\gamma)\eta_i^T(\gamma) + \sigma_\beta^{-2}I_M)^{-1} \times (\sigma^{-2}\sum_{i=1}^{N}\theta_i\eta_i(\gamma) + \sigma_\beta^{-2}\mu_\beta 1_M)$$

$$H_\beta = \sigma^{-2}\sum_{i=1}^{N}\eta_i(\gamma)\eta_i^T(\gamma) + \sigma_\beta^{-2}I_M)^{-1}$$

(b)

$$\begin{aligned}&\mu_\beta|\theta, \beta, \gamma, \mu_\gamma, \sigma^2, \sigma_\beta^2, S_\gamma, y\\ &\qquad \sim N\left[(M\sigma_\beta^{-2} + A_\beta^{-1})^{-1}\left(\sigma_\beta^{-2}\sum_{j=1}^{M}\beta_j + A_\beta^{-1}a_\beta\right),\right.\\ &\qquad\qquad \left.(M\sigma_\beta^{-2} + A_\beta^{-1})^{-1}\right];\end{aligned}$$

(c)

$$\begin{aligned}&\mu_\gamma|\theta, \beta, \gamma, \mu_\beta, \sigma^2, \sigma_\beta^2, S_\gamma, y\\ &\qquad \sim N_p\left[(MS_\gamma^{-1} + A_\gamma^{-1})^{-1}\left(S_\gamma^{-1}\sum_{j=1}^{M}\gamma_j + A_\gamma^{-1}a_\gamma\right),\right.\\ &\qquad\qquad \left.(MS_\gamma^{-1} + A_\gamma^{-1})^{-1}\right];\end{aligned}$$

(d)

$$\begin{aligned}&\sigma^2|\theta, \beta, \gamma, \mu_\beta, \mu_\gamma, \sigma_\beta^2, S_\gamma, y\\ &\qquad \sim IG\left(\frac{N + c_\sigma}{2}, \frac{\sum_{i=1}^{N}(\theta_i - \eta_i^T(\gamma)\beta)^2 + c_\sigma C_\sigma}{2}\right);\end{aligned}$$

(e)

$$\begin{aligned}&\sigma_\beta^2|\theta, \beta, \gamma, \mu_\beta, \mu_\gamma, \sigma^2, S_\gamma, y\\ &\qquad \sim IG\left(\frac{M + c_\beta}{2}, \frac{\|\beta - \mu_\beta 1_M\|^2 + c_\beta C_\beta}{2}\right);\end{aligned}$$

(f)

$$S_\gamma|\theta, \beta, \gamma, \mu_\beta, \mu_\gamma, \sigma^2, \sigma^2_\beta, y$$
$$\sim IW\left(M + c_\gamma, \left(c_\gamma C_\gamma + \sum_{j=1}^{M}(\gamma_j - \mu_j)(\gamma_j - \mu_\gamma)^T\right)^{-1}\right);$$

(g)

$$\pi(\gamma|\theta, \beta, \mu_\beta, \mu_\gamma, \sigma^2, \sigma^2_\beta, S_\gamma, y) \propto \exp[-\frac{1}{2}\{F_\gamma + M_\gamma] \tag{3.26}$$

where

$$F_\gamma = -\frac{1}{2}\sum_{i=1}^{N}(\theta_i - \eta_i^T(\gamma)\beta)^2 \tag{3.27}$$

$$M_\gamma = \sum_{j=1}^{M}(\gamma_j - \mu_\gamma)^T S_\gamma^{-1}(\gamma_j - \mu_\gamma). \tag{3.28}$$

The pdf's given in (a)-(f) are standard, and it is easy to generate samples from them. But this is not so for the pdf's given in (g). Generating samples from (g) is even more complicated, and requires Metropolis-Hastings updates at every step.

3.7 DATA GENERATION

The data will be generated using:

$$y = \beta x + 0.3sin(2\pi(\gamma x + \epsilon)) + \epsilon \tag{3.29}$$

where $\epsilon \sim$ N(0, 0.02), x $\sim$ N(0,1), $\beta = 1.0$ and $\gamma = 1.0$

(1) The results are based on the prediction and model selection criterion given the level of hidden neuron at different sample sizes. (2) The hidden neuron used is 2 while the sample sizes include 50, 100, 200, 500, and 1000, 2000, 5000, 10000, 20000 and 50000. That is, taking each sample size, statistics will be conducted at different levels of the training sets(70%, 80% and 90%).

(3) Real life data is also used in this study.

(4) R package was used throughout the study for analysis.

The parameters of the model are obtained using Bayesian approach and the sensitivity of the estimators are assessed with the following techniques,

(1) Posterior Mean Weight (2) Posterior Standard Deviation (3) Numerical Standard Error (NSE).

3.8 EVALUATION METRICS FOR SIMULATION DATA

(1) **Training error**: Training error is calculated by using the same data for training the model and calculating its error rate. Calculating the error rate for a predictive model is called model validation. It is important to validate our model before they go into production in order to decide if the expected model performance will be good enough for production. The predictive performance of the model is obtained by comparing the predicted values with the true values for Y. When the model is applied to the data it was trained on, the training error is being calculated. It is calculated as:

$$TrainError = \frac{1}{n}\sum_{i=1}^{n}|Y_T - \hat{Y_T}| \tag{3.30}$$

where
Y_T is the trained data
$\hat{Y_T}$ is the predicted value for the trained data.
(2)**Mean Square Error**: This measures the average of the squares of the errors, that is, the average squared difference between the estimated value and the actual value. If a vector of n predictions is generated from a sample of n data points on all variables, and Y is the vector of observed values of the variable being predicted, with $\hat{Y}$ being the predicted values, then the MSE is computed as

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_t - \hat{Y_t})^2 \tag{3.31}$$

where
Y_t is the test data
$\hat{Y_t}$ is the predicted value for the test data.
This MSE is computed on test data that are not used in estimating the model, because they were held back for this purpose. The MSE here is often called the test MSE.

(3) **Mean Absolute Error**: is the average over the sample of the absolute values of the differences between model predicted values and the test data. It is computed as

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |Y_t - \hat{Y}_t| \tag{3.32}$$

where

Y_t is the test data

$\hat{Y}_t$ is the predicted value for the test data.

3.9 APPLICATION TO TERRORISM DATA

The dataset used is a primary data collected via an electronic medium, designed in a Google form and distributed randomly to different platforms in the six geopolitical zones (North east, North west, North Central, South East, South West and South South) in Nigeria. The form (which is accessible through the link: www.https://forms.gle/gQ3idgnUT5kk8gx47) was designed in such a way that each respondent's personal information was not traced to their responses. This was done to keep the confidentiality of the respondents and for obtaining exactness of correct information provided during the longitudinal survey. For this study, 508 responses are received was received for the analysis for this study.

Different variables are gathered in the data collection. The nature of the terrorist attack is categorized into different levels of attack, namely: armed assault, assassination, bombing/explosion, facility/infrastructure attack, hijacking, unarmed assault and unknown type. Obviously, terrorist attack cannot be achievable without the application of some forms of weapons. The dataset also shows the targets of the terrorists (educational institution, government, media, military, NGO, police, private business organization, private citizen and property, religious institutions, and political parties). The perpetrators' information which include name, number of perpetrators, and so on, are also included in the data. And finally, other information also gathered include the number of people injured or died, number of perpetrators injured or killed, the success of the attack, time of the attack, date, vicinity and city of the attack, frequency of the attack and the number of kidnapped victims.

3.10 MODEL VARIABLES FOR THE TERRORISM DATA

The target or dependent variables are:

(1) The nature of the terrorist attack: armed assault, assassination, bombing/explosion, facility/infrastructure attack, hijacking, unarmed assault and Kidnapping.

(2) suicide attack(do people die): Yes or NO.

The variables used as feature variables in the study are:

(1) Weapon Type: Fake Weapons, Firearms, Melee like knife, Nuclear/Explosives, Sabotage Equipment and so on. (2) Number of perpetrators (3) The targets of the terrorists : educational institution, government, media, military, NGO, police, private business organization, private citizen and property, religious institutions, and political parties). (4) Attack type: armed assault, assassination, bombing/explosion, facility/infrastructure attack, hijacking, unarmed assault and Kidnapping. (5) State of the incidence (6) Other information gathered in the study are number of people injured or died, number of perpetrators injured or killed, the success of the attack, time of the attack, date, vicinity and city of the attack, frequent of the attack, number of kidnapped victims and so on.

3.11 MODEL PERFORMANCE FOR TERRORISM DATA

The performance of the models are on these four criteria: Accuracy, Precision, Recall and F1-score. The mathematical expressions for the criteria are discussed below:

- Accuracy = (TN + TP)/ (TN + TP + FP + FN)
- Precision = TP / (TP + FP)
- Recall = TP / (TP + FN)
- F1-score = 2 x (Precision x Recall)/ (Precision + Recall)

where TN, TP, FN and FP are True Negative, True Positive, False Negative and False Positive respectively. These are obtained in the confusion matrix and then used to compute the performance criteria. The confusion matrix is a performance measurement in machine learning classification problems.

Chapter 4
RESULTS AND DISCUSSION

4.1 CHAPTER OVERVIEW

This chapter discusses the analyses and results from the simulations and the real life data on terrorism attack. The purpose of using the data is to verify the precision of the heterogeneous models of the Bayesian neural network (BNN). The results are based on the prediction and model selection criterion at different data trainings and sample sizes. The parameter estimates, asymptotic performance of the derived bayesian estimators for the homogeneous(HOMAFs) and heterogeneous activation function (HETAFs) are presented in this section. The result for the real life data is also presented.

The results presented are categorized under different activation functions, data trainings percentages and sample sizes. The hidden neuron of 2 is used while the sample sizes are 50, 100, 200, 500, 1000, 5000, 10000, 20000 and 50000. That is, taking each sample size, statistics are computed at the level of the choice of hidden neuron. The intention is to see the behavior of the network at different variables. Three primary transfer functions (HOTTFs), as well as two derived activation functions (HETTFs) arising from the convolution of the HOTTFs, are used, namely;
(i) ReLU activation function (ReLU)
(ii) Sigmoid activation function (Sigmoid)
(iii) Hyperbolic Tangent sigmoid activation function (TANSIG)
(iv) Symmetric Saturated Linear and Hyperbolic Tangent activation function (SSLHT)
(v) Symmetric Saturated Linear and Hyperbolic Tangent Sigmoid activation function (SSLHTS)
Training sets of 70, 80 and 90 are used for the simulation study. The results show the posterior weights for the parameters, asymptotic performance and the

performance between the activation functions are judged using the mean square error(mse), mean absolute error(mae) and train error.

4.2 BAYESIAN RESULTS FOR ReLU ACTIVATION FUNCTION

The results for the parameters of the model using Bayesian approach are displayed in this sesction. These results are tabled under the headings of Posterior Mean Weight, Posterior Standard Deviation and the Numerical Standard Error (NSE). The results are obtained for the ReLU activation functions at 70%, 80% and 90% training sets and at sample size of 50, 100, 200, 500, 1000, 5000, 10000, 20000 and 50000.

Table 4.1: **Summary Table of Posterior Weight, Standard Deviation and Numerical Standard Error for ReLU activation function at N=50.**

Training set	Parameters	results_pweight	results_pstd	results_nse
	β	0.4275	1.0581	0.0335
	γ	0.2165	1.2154	0.0384
	μ_β	0.2187	0.5067	0.016
70	μ_γ	0.4158	0.6875	0.0217
	σ	0.3526	0.0664	0.0021
	σ_β	1.5102	0.7627	0.0241
	S_γ	0.0049	0.0024	1.00E-04
	π	0.2322	0.2373	0.0075
	β	0.3159	1.0041	0.0318
	γ	0.1797	1.1374	0.036
80	μ_β	0.286	0.495	0.0157
	μ_γ	0.4158	0.6875	0.0217
	σ	0.2845	0.0536	0.0017
	σ_β	1.5102	0.7627	0.0241
	S_γ	0.0049	0.0024	1.00E-04
	π	0.4492	0.4592	0.0145
	β	0.2812	0.9831	0.0311
	γ	0.17	1.1059	0.035
90	μ_β	0.3302	0.4874	0.0154
	μ_γ	0.4158	0.6875	0.0217
	σ	0.2615	0.0493	0.0016
	σ_β	1.5102	0.7627	0.0241
	S_γ	0.0049	0.0024	1.00E-04
	π	0.656	0.6706	0.0212

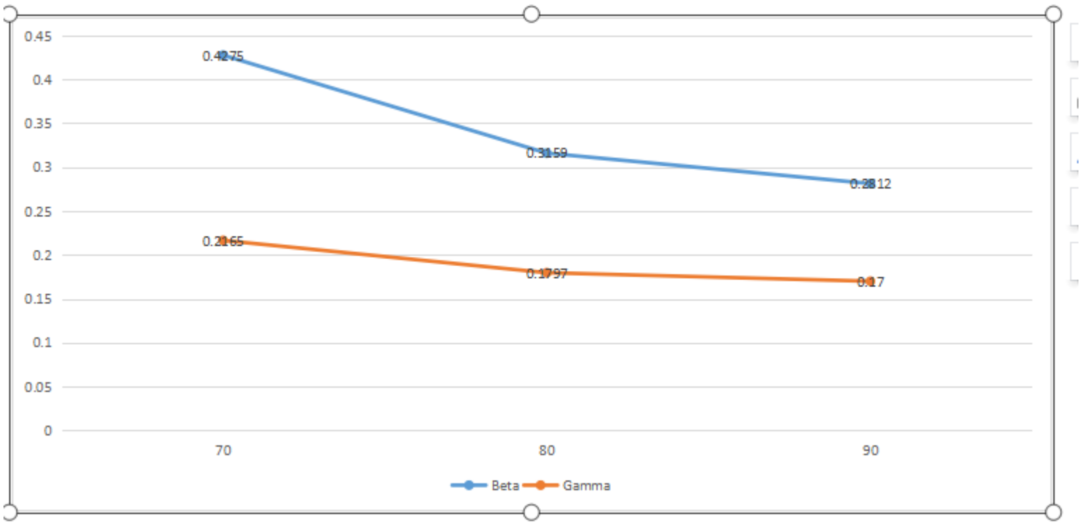

Figure 4.1: **Posterior weight estimates of beta and gamma at N=50 using ReLU Activation function**

Table 4.1 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using ReLU activation function at the sample size of 50. The table shows that β has a posterior weight of 0.4275, 0.3159 and 0.2812 and γ shows 0.2165, 0.1797 and 0.17 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for both β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 1.0581, 1.0041 and 0.9831 and γ also has a posterior standard deviation of 1.2154, 1.1374 and 1.1059 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0335, 0.0318 and 0.0311 and γ also shows NSE values of 0.0384, 0.036 and 0.035 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.1 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 50.

Table 4.2: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error using ReLU activation function at N=100.**

Training set	Parameters	results_pweight	results_pstd	results_nse
	β	0.2302	0.9577	0.0303
	γ	0.3208	1.0584	0.0335
70	μ_β	0.4421	0.4703	0.0149
	μ_γ	0.4167	0.6872	0.0217
	σ	1.4905	0.2024	0.0064
	σ_β	1.5168	0.7589	0.024
	S_γ	0.0049	0.0024	1.00E-04
	π	0.0129	0.0133	0.0004
	β	0.2811	0.9385	0.0297
	γ	0.346	1.0484	0.0332
80	μ_β	0.4709	0.4658	0.0147
	μ_γ	0.4167	0.6872	0.0217
	σ	1.7579	0.2387	0.0075
	σ_β	1.5168	0.7589	0.024
	S_γ	0.0049	0.0024	1.00E-04
	π	0.0108	0.0111	0.0004
	β	0.242	0.8368	0.0265
	γ	0.4446	0.9412	0.0298
90	μ_β	0.5182	0.4575	0.0145
	μ_γ	0.4167	0.6872	0.0217
	σ	2.3166	0.3146	0.0099
	σ_β	1.5168	0.7589	0.024
	S_γ	0.0049	0.0024	1.00E-04
	π	0.008	0.0083	0.0003

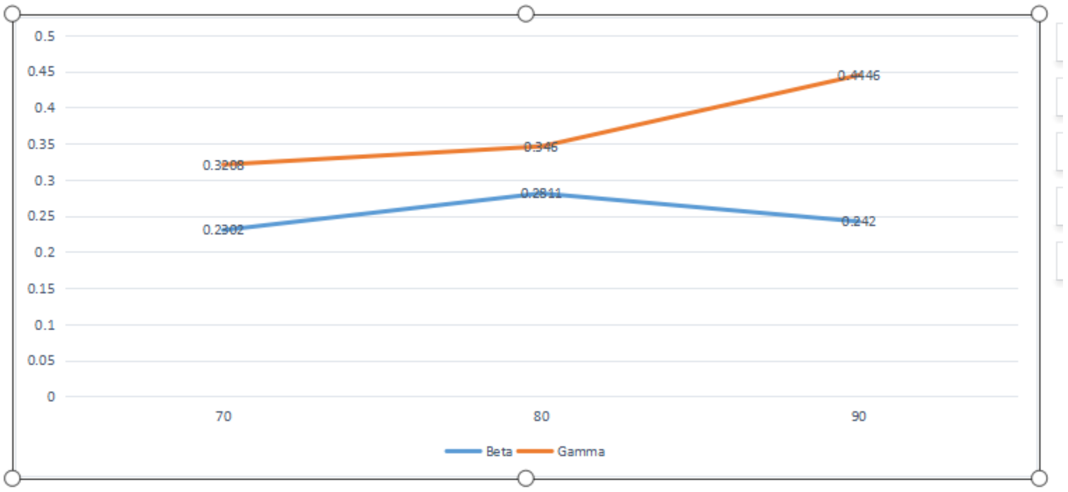

Figure 4.2: **Posterior Mean weights of beta and gamma at N=100 using ReLU Activation function**

Table 4.2 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using ReLU activation function at the sample size of 100. The table shows that β has a posterior weight of 0.2302, 0.2811 and 0.242 and γ shows 0.3208, 0.346 and 0.4446 at training sets of 70%, 80% and 90% respectively. The result shows an consistent posterior weights for both β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.9577, 0.9385 and 0.8368 and γ also has a posterior standard deviation of 1.0584, 1.0484 and 0.9412 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0303, 0.0297 and 0.0265 and γ also shows NSE values of 0.0335, 0.0332 and 0.0298 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.2 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 100.

Table 4.3: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error using ReLU activation function at N=200.**

Training set	Parameters	results_weight	results_pstd	results_nse
70	β	0.4305	0.9123	0.0288
	γ	0.1835	0.8303	0.0263
	μ_β	0.5696	0.4439	0.014
	μ_γ	0.4191	0.6868	0.0217
	σ	6.6677	0.6462	0.0204
	σ_β	1.5153	0.7588	0.024
	S_γ	0.0049	0.0024	1.00E-04
	π	0.0014	0.0014	0.00E+00
80	β	0.3613	0.8604	0.0272
	γ	0.1568	0.7871	0.0249
	μ_β	0.6034	0.438	0.0139
	μ_γ	0.4191	0.6868	0.0217
	σ	8.8793	0.8606	0.0272
	σ_β	1.5153	0.7588	0.024
	S_γ	0.0049	0.0024	1.00E-04
	π	0.001	0.0011	0.00E+00
90	β	0.3341	0.8051	0.0255
	γ	0.1533	0.7308	0.0231
	μ_β	0.6222	0.4348	0.0137
	μ_γ	0.4191	0.6868	0.0217
	σ	11.7606	1.1399	0.036
	σ_β	1.5153	0.7588	0.024
	S_γ	0.0049	0.0024	1.00E-04
	π	0.0008	0.0008	0.00E+00

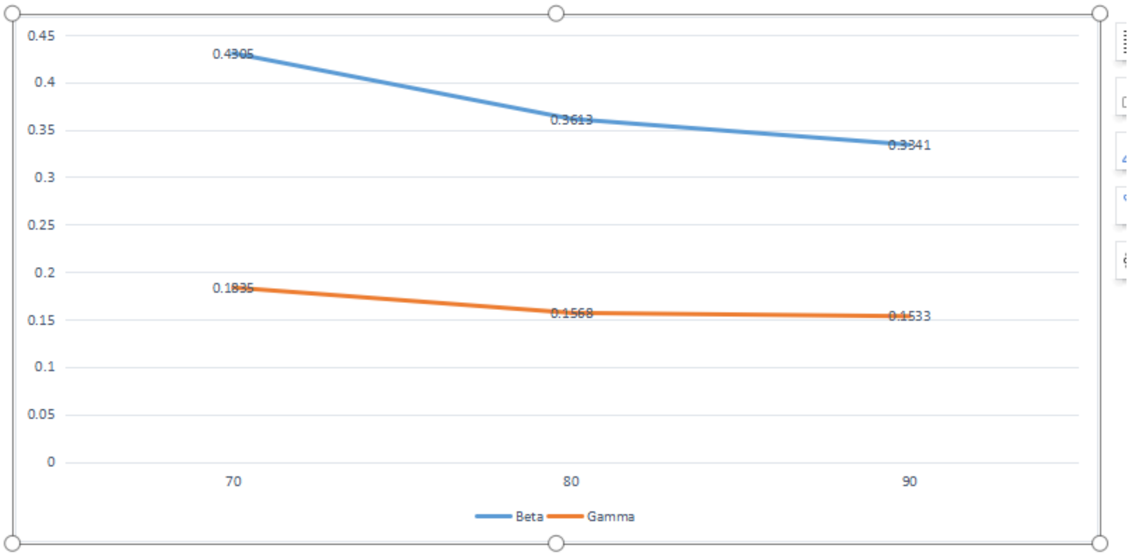

Figure 4.3: **Posterior Mean weights of beta and gamma at N=200 using ReLU Activation function**

Table 4.3 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using ReLU activation function at the sample size of 200. The table shows that β has a posterior weight of 0.4305, 0.3613 and 0.3341 and γ shows 0.1835, 0.1568 and 0.1533 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for both β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.9123, 0.8604 and 0.8051 and γ also has a posterior standard deviation of 0.8303, 0.7871 and 0.7308 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0288, 0.0272 and 0.0255 and γ also shows NSE values of 0.0263, 0.0249 and 0.0231 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.3 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 200.

Table 4.4: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error using ReLU activation function at N=500.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.2359	0.5571	0.0176
	γ	0.1666	0.4587	0.0145
	μ_β	0.7181	0.4193	0.0133
	μ_γ	0.4283	0.6905	0.0218
	σ	38.8611	2.4032	0.076
	σ_β	1.5233	0.7584	0.024
	S_γ	0.0049	0.0024	1.00E-04
	π	1.00E-04	0.0001	0.00E+00
80	β	0.2078	0.5228	0.0165
	γ	0.1533	0.428	0.0135
	μ_β	0.7299	0.4172	0.0132
	μ_γ	0.4283	0.6905	0.0218
	σ	50.7201	3.1366	0.0992
	σ_β	1.5233	0.7584	0.024
	S_γ	0.0049	0.0024	1.00E-04
	π	1.00E-04	1.00E-04	0.00E+00
90	β	0.1834	0.4922	0.0156
	γ	0.1403	0.4012	0.0127
	μ_β	0.739	0.4155	0.0131
	μ_γ	0.4283	0.6905	0.0218
	σ	63.405	3.9211	0.124
	σ_β	1.5233	0.7584	0.024
	S_γ	0.0049	0.0024	1.00E-04
	π	1.00E-04	1.00E-04	0.00E+00

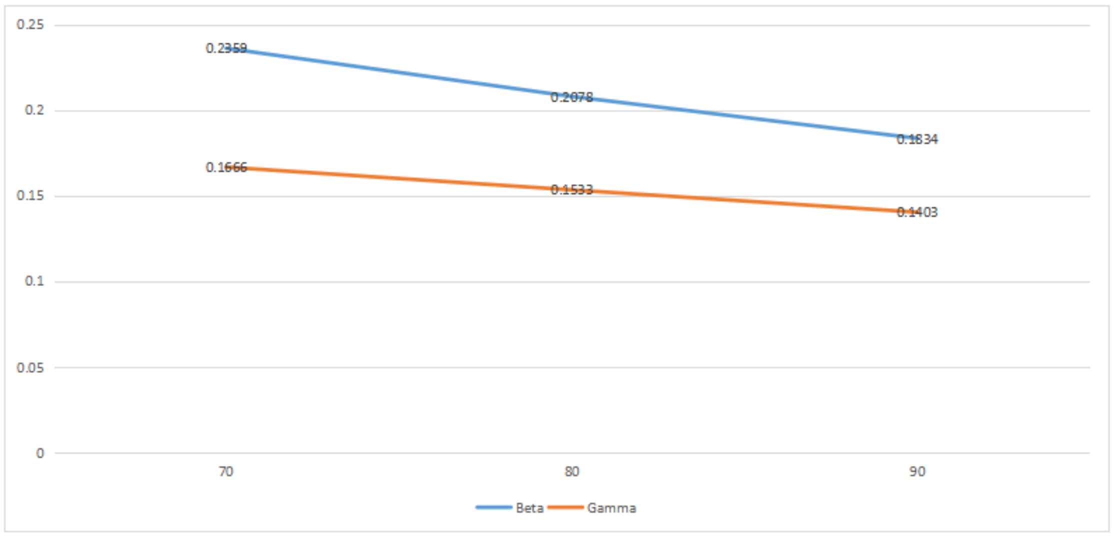

Figure 4.4: **Posterior Mean weights of beta and gamma at N=500 using ReLU Activation function**

Table 4.4 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using ReLU activation function at the sample size of 500. The table shows that β has a posterior weight of 0.2359, 0.2078 and 0.1834 and γ shows 0.1666, 0.1533 and 0.1403 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for both β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.5571, 0.5228 and 0.4922 and γ also has a posterior standard deviation of 0.4587, 0.428 and 0.4012 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0176, 0.0165 and 0.0156 and γ also shows NSE values of 0.0145, 0.0135 and 0.0127 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.4 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 500.

Table 4.5: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error using ReLU activation function at N=1000.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.4872	0.3919	0.0124
	γ	0.2	0.3196	0.0101
	μ_β	0.7722	0.4084	0.0129
	μ_γ	0.4281	0.692	0.0219
	σ	62.3068	2.7322	0.0864
	σ_β	1.5248	0.7568	0.0239
	S_γ	0.0048	0.0023	1.00E-04
	π	0.00E+00	0.00E+00	0.00E+00
80	β	0.4011	0.3649	0.0115
	γ	0.1901	0.2972	0.0094
	μ_β	0.7791	0.4073	0.0129
	μ_γ	0.4281	0.692	0.0219
	σ	81.2728	3.5639	0.1127
	σ_β	1.5248	0.7568	0.0239
	S_γ	0.0048	0.0023	1.00E-04
	π	0.00E+00	0.00E+00	0.00E+00
90	β	0.4752	0.3468	0.011
	γ	0.1838	0.2826	0.0089
	μ_β	0.7842	0.4064	0.0129
	μ_γ	0.4281	0.692	0.0219
	σ	102.5323	4.4961	0.1422
	σ_β	1.5248	0.7568	0.0239
	S_γ	0.0048	0.0023	1.00E-04
	π	0.00E+00	0.00E+00	0.00E+00

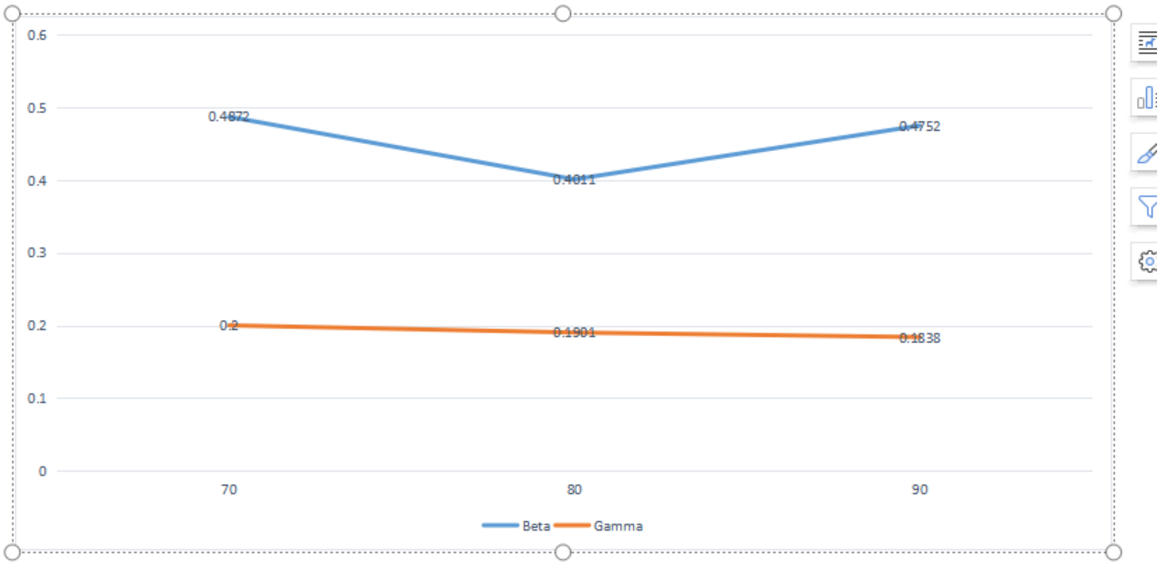

Figure 4.5: **Posterior Mean weights of beta and gamma at N=1000 using ReLU Activation function**

Table 4.5 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using ReLU activation function at the sample size of 1000. The table shows that β has a posterior weight of 0.4872, 0.4011 and 0.4752 and γ shows 0.2, 0.1901 and 0.1838 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for both β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.3919, 0.3649 and 0.3468 and γ also has a posterior standard deviation of 0.3196, 0.2972 and 0.2826 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0124, 0.0115 and 0.011 and γ also shows NSE values of 0.0101, 0.0094 and 0.0089 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.5 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 1000.

Table 4.6: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error using ReLU activation function at N=5000.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.3412	0.1837	0.0058
	γ	0.271	0.1743	0.0055
	μ_β	0.8196	0.4032	0.0127
	μ_γ	0.4286	0.6976	0.0221
	σ	85.0329	5.6284	0.178
	σ_β	1.523	0.7498	0.0237
	S_γ	0.0048	0.0023	1.00E-04
	π	0.00E+00	0.00E+00	0.00E+00
80	β	0.3642	0.1713	0.0054
	γ	0.266	0.1627	0.0051
	μ_β	0.8207	0.4031	0.0127
	μ_γ	0.4286	0.6976	0.0221
	σ	70.9248	7.3244	0.2316
	σ_β	1.523	0.7498	0.0237
	S_γ	0.0048	0.0023	1.00E-04
	π	0.00E+00	0.00E+00	0.00E+00
90	β	0.3735	0.162	0.0051
	γ	0.264	0.154	0.0049
	μ_β	0.8216	0.403	0.0127
	μ_γ	0.4286	0.6976	0.0221
	σ	69.7376	9.2756	0.2933
	σ_β	1.523	0.7498	0.0237
	S_γ	0.0048	0.0023	1.00E-04
	π	0.00E+00	0.00E+00	0.00E+00

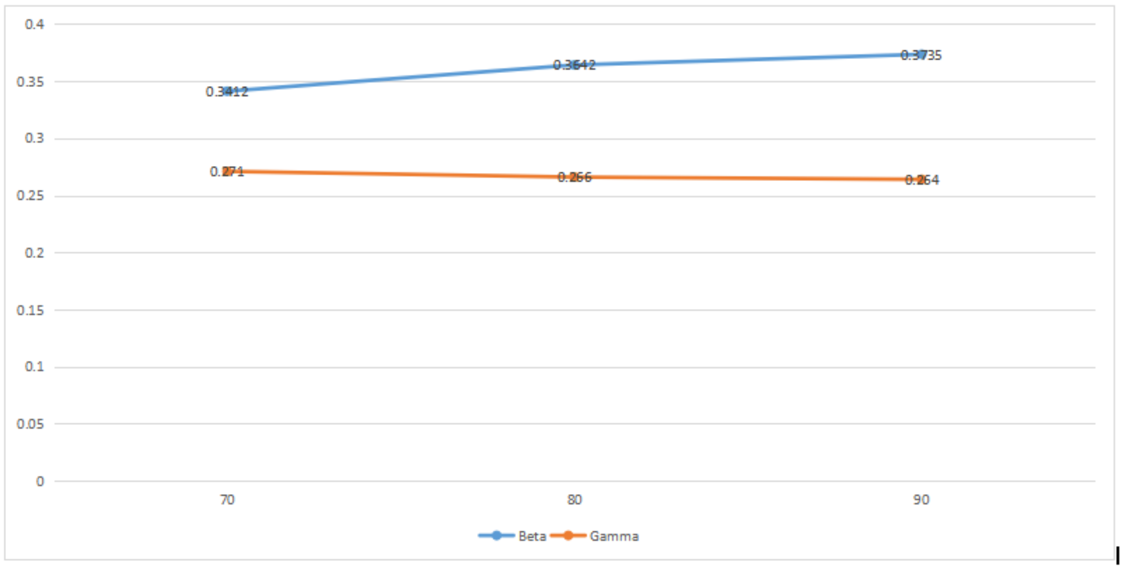

Figure 4.6: **Posterior Mean weights of beta and gamma at N=5000 using ReLU Activation function**

Table 4.6 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using ReLU activation function at the sample size of 5000. The table shows that β has a posterior weight of 0.3412, 0.3642 and 0.3735 and γ shows 0.271, 0.266 and 0.264 at training sets of 70%, 80% and 90% respectively. The result shows an increasing posterior weights for β and a decreasing posterior weights for γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.1837, 0.1713 and 0.162 and γ also has a posterior standard deviation of 0.1743, 0.1627 and 0.154 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0058, 0.0054 and 0.0051 and γ also shows NSE values of 0.0055, 0.0051 and 0.0049 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.6 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 5000.

Table 4.7: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error using ReLU activation function at N=10000.**

Training set	Parameters	results_pweight	results_pstd	results_nse
	β	0.4094	0.1379	0.0044
	γ	0.255	0.1319	0.0042
	μ_β	0.8238	0.4029	0.0127
70	μ_γ	0.4286	0.6976	0.0221
	σ	95.8872	6.9248	0.219
	σ_β	1.523	0.7498	0.0237
	S_γ	0.0048	0.0023	1.00E-04
	π	0.00E+00	0.00E+00	0.00E+00
	β	0.4202	0.1284	0.0041
	γ	0.252	0.1231	0.0039
	μ_β	0.8244	0.4029	0.0127
80	μ_γ	0.4286	0.6976	0.0221
	σ	89.8349	9.0746	0.287
	σ_β	1.523	0.7498	0.0237
	S_γ	0.0048	0.0023	1.00E-04
	π	0.00E+00	0.00E+00	0.00E+00
	β	0.428	0.1209	0.0038
	γ	0.249	0.1161	0.0037
	μ_β	0.8249	0.4029	0.0127
90	μ_γ	0.4286	0.6976	0.0221
	σ	82.0568	11.4936	0.3635
	σ_β	1.523	0.7498	0.0237
	S_γ	0.0048	0.0023	1.00E-04
	π	0.00E+00	0.00E+00	0.00E+00

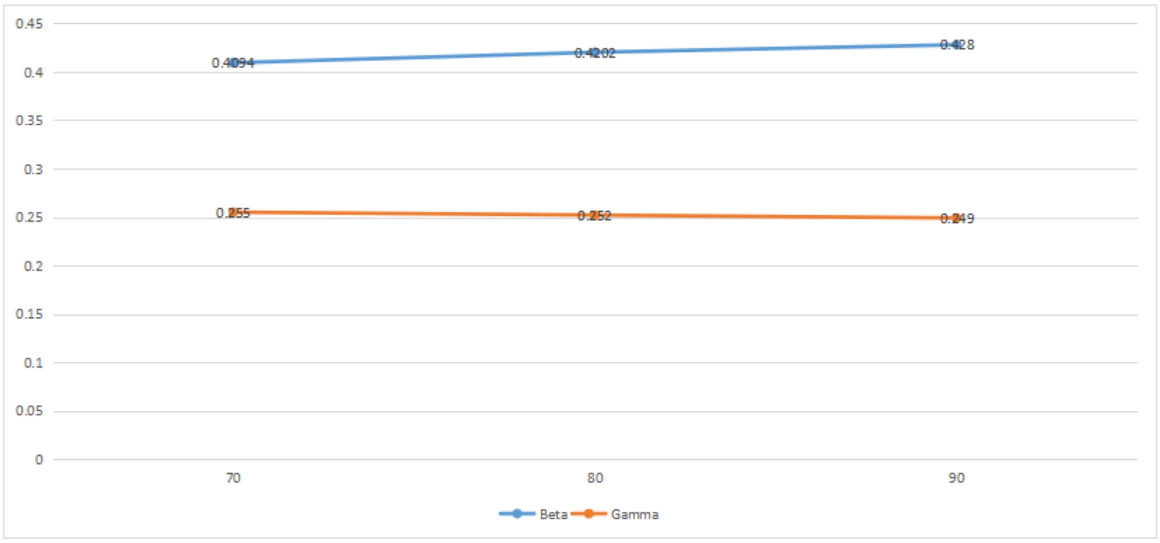

Figure 4.7: **Posterior Mean weights of beta and gamma at N=10000 using ReLU Activation function**

Table 4.7 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using ReLU activation function at the sample size of 10000. The table shows that β has a posterior weight of 0.4094, 0.4202 and 0.428 and γ shows 0.255, 0.252 and 0.249 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.1379, 0.1284 and 0.1209 and γ also has a posterior standard deviation of 0.1319, 0.1231 and 0.1161 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0044, 0.0041 and 0.0038 and γ also shows NSE values of 0.0042, 0.0039 and 0.0037 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.7 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 10000.

Table 4.8: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error using ReLU activation function at N=20000.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.4535	0.1076	0.0034
	γ	0.242	0.1056	0.0033
	μ_β	0.8263	0.4029	0.0127
	μ_γ	0.4286	0.6976	0.0221
	σ	1430.3183	14.1241	0.4466
	σ_β	1.523	0.7498	0.0237
	S_γ	0.0048	0.0023	1.00E-04
	π	0.00E+00	0.00E+00	0.00E+00
80	β	0.4574	0.1006	0.0032
	γ	0.243	0.0987	0.0031
	μ_β	0.8266	0.4029	0.0127
	μ_γ	0.4286	0.6976	0.0221
	σ	1870.4128	18.47	0.5841
	σ_β	1.523	0.7498	0.0237
	S_γ	0.0048	0.0023	1.00E-04
	π	0.00E+00	0.00E+00	0.00E+00
90	β	0.4601	0.0946	0.003
	γ	0.245	0.0928	0.0029
	μ_β	0.8268	0.4029	0.0127
	μ_γ	0.4286	0.6976	0.0221
	σ	2366.5608	23.3693	0.739
	σ_β	1.523	0.7498	0.0237
	S_γ	0.0048	0.0023	1.00E-04
	π	0.00E+00	0.00E+00	0.00E+00

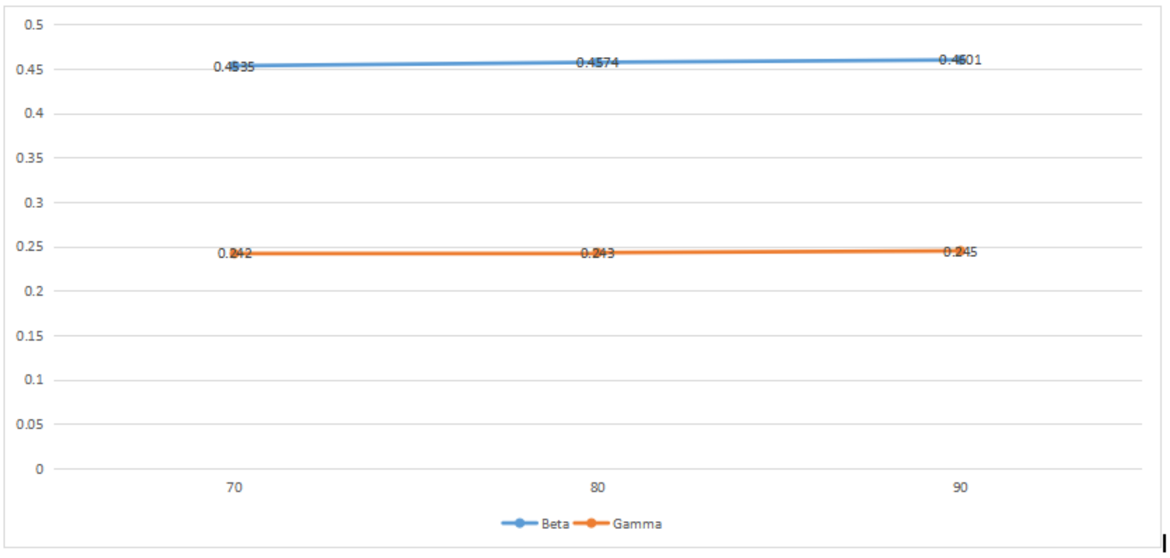

Figure 4.8: **Posterior Mean weights of beta and gamma at N=20000 using ReLU Activation function**

Table 4.8 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using ReLU activation function at the sample size of 20000. The table shows that β has a posterior weight of 0.4535, 0.4574 and 0.4601 and γ shows 0.242, 0.243 and 0.245 at training sets of 70%, 80% and 90% respectively. The result shows an increasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.1076, 0.1006 and 0.0946 and γ also has a posterior standard deviation of 0.1056, 0.0987 and 0.0928 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0034, 0.0032 and 0.003 and γ also shows NSE values of 0.0033, 0.0031 and 0.0029 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.8 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 20000.

Table 4.9: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error using ReLU activation function at N=50000.**

Training set	Parameters	results_pweight	results_pstd	results_nse
	β	0.4755	0.0646	0.002
	γ	0.2371	0.0642	0.002
	μ_β	0.8277	0.403	0.0127
70	μ_γ	0.4286	0.6976	0.0221
	σ	3038.9478	18.9794	0.6002
	σ_β	1.523	0.7498	0.0237
	S_γ	0.0048	0.0023	1.00E-04
	π	0.00E+00	0.00E+00	0.00E+00
	β	0.477	0.0603	0.0019
	γ	0.2370	0.0599	0.0019
	μ_β	0.8278	0.403	0.0127
80	μ_γ	0.4286	0.6976	0.0221
	σ	3971.4881	24.8034	0.7844
	σ_β	1.523	0.7498	0.0237
	S_γ	0.0048	0.0023	1.00E-04
	π	0.00E+00	0.00E+00	0.00E+00
	β	0.4781	0.0568	0.0018
	γ	0.2364	0.0565	0.0018
	μ_β	0.8279	0.403	0.0127
90	μ_γ	0.4286	0.6976	0.0221
	σ	5005.6174	31.262	0.9886
	σ_β	1.523	0.7498	0.0237
	S_γ	0.0048	0.0023	1.00E-04
	π	0.00E+00	0.00E+00	0.00E+00

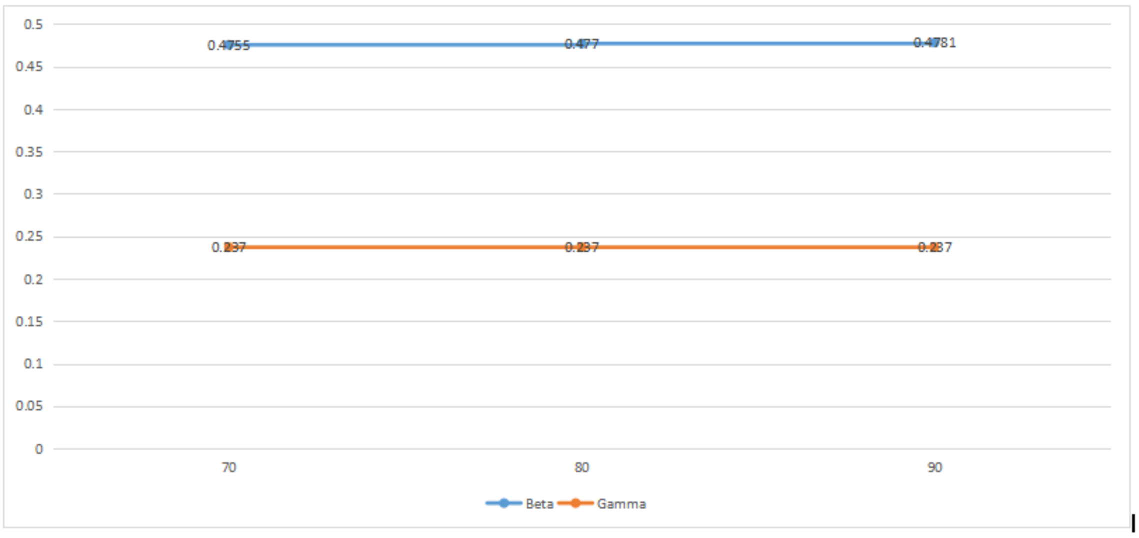

Figure 4.9: **Posterior Mean weights of beta and gamma at N=50000 using ReLU Activation function**

Table 4.9 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using ReLU activation function at the sample size of 50000. The table shows that β has a posterior weight of 0.4755, 0.477 and 0.4781 and γ shows 0.2371, 0.2370 and 0.2364 at training sets of 70%, 80% and 90% respectively. The result shows an increasing posterior weights for β and a decreasing posterior weights for γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.0646, 0.0603 and 0.0568 and γ also has a posterior standard deviation of 0.0642, 0.599 and 0.0565 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.002, 0.0019 and 0.0018 and γ also shows NSE values of 0.002, 0.0019 and 0.0018 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.9 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 50000.

4.3 BAYESIAN RESULTS FOR SIGMOID ACTIVATION FUNCTION

The results for the parameters of the model using Bayesian approach with sigmoid activation function are displayed in this sesction. These results are tabled under the headings of Posterior Mean Weight, Posterior Standard Deviation and the Numerical Standard Error (NSE). The results are obtained for the Sigmoid activation functions at 70%, 80% and 90% training sets and at sample size of 50, 100, 200, 500, 1000, 5000, 10000, 20000 and 50000.

Table 4.10: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error using Sigmoid activation function at N=50.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.345	0.4756	0.1504
	γ	0.1976	0.4888	0.1546
	μ_β	0.3478	0.4846	0.0153
	μ_γ	0.4158	0.6875	0.0217
	σ	0.5284	0.0995	0.0031
	σ_β	1.5102	0.7627	0.0241
	S_γ	0.0049	0.0024	1.00E-04
	π	0.1033	0.1056	0.0033
80	β	0.3159	1.0041	0.0318
	γ	0.1797	1.1374	0.036
	μ_β	0.286	0.495	0.0157
	μ_γ	0.4158	0.6875	0.0217
	σ	0.2845	0.0536	0.0017
	σ_β	1.5102	0.7627	0.0241
	S_γ	0.0049	0.0024	1.00E-04
	π	0.4492	0.4592	0.0145
90	β	0.2812	0.9831	0.0311
	γ	0.17	1.1059	0.035
	μ_β	0.3302	0.4874	0.0154
	μ_γ	0.4158	0.6875	0.0217
	σ	0.2615	0.0493	0.0016
	σ_β	1.5102	0.7627	0.0241
	S_γ	0.0049	0.0024	1.00E-04
	π	0.656	0.6706	0.0212

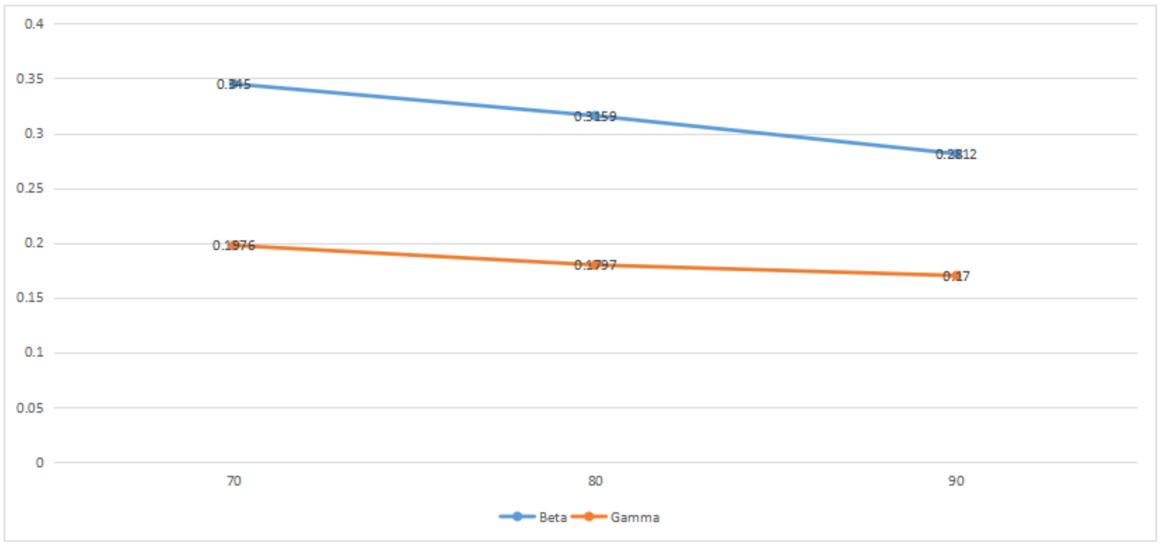

Figure 4.10: **Posterior Mean weights of beta and gamma at N=50 using Sigmoid Activation function**

Table 4.10 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using Sigmoid activation function at the sample size of 50. The table shows that β has a posterior weight of 0.345, 0.3159 and 0.2812 and γ shows 0.1976, 0.1797 and 0.17 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.4756, 1.0041 and 0.9831 and γ also has a posterior standard deviation of 0.4888, 1.1374 and 1.1059 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.1504, 0.0313 and 0.0311 and γ also shows NSE values of 0.1546, 0.036 and 0.035 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.10 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 50.

Table 4.11: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error using Sigmoid activation function at N=100.**

Training set	Parameters	results_pweight	results_pstd	results_nse
	β	0.4821	4.2331	0.1339
	γ	0.1577	4.3237	0.1367
	μ_β	0.5481	0.4517	0.0143
70	μ_γ	0.4167	0.6872	0.0217
	σ	5.7997	0.7877	0.0249
	σ_β	1.5168	0.7589	0.024
	S_γ	0.0049	0.0024	1.00E-04
	π	0.0031	0.0032	0.0001
	β	0.4395	4.1741	0.132
	γ	0.1582	4.2718	0.1351
	μ_β	0.5699	0.4482	0.0142
80	μ_γ	0.4167	0.6872	0.0217
	σ	7.2214	0.9808	0.031
	σ_β	1.5168	0.7589	0.024
	S_γ	0.0049	0.0024	1.00E-04
	π	0.0025	0.0026	1.00E-04
	β	0.3509	3.7577	0.1188
	γ	0.1601	3.8493	0.1217
	μ_β	0.5971	0.4433	0.014
90	μ_γ	0.4167	0.6872	0.0217
	σ	8.637	1.173	0.0371
	σ_β	1.5168	0.7589	0.024
	S_γ	0.0049	0.0024	1.00E-04
	π	0.0021	0.0022	1.00E-04

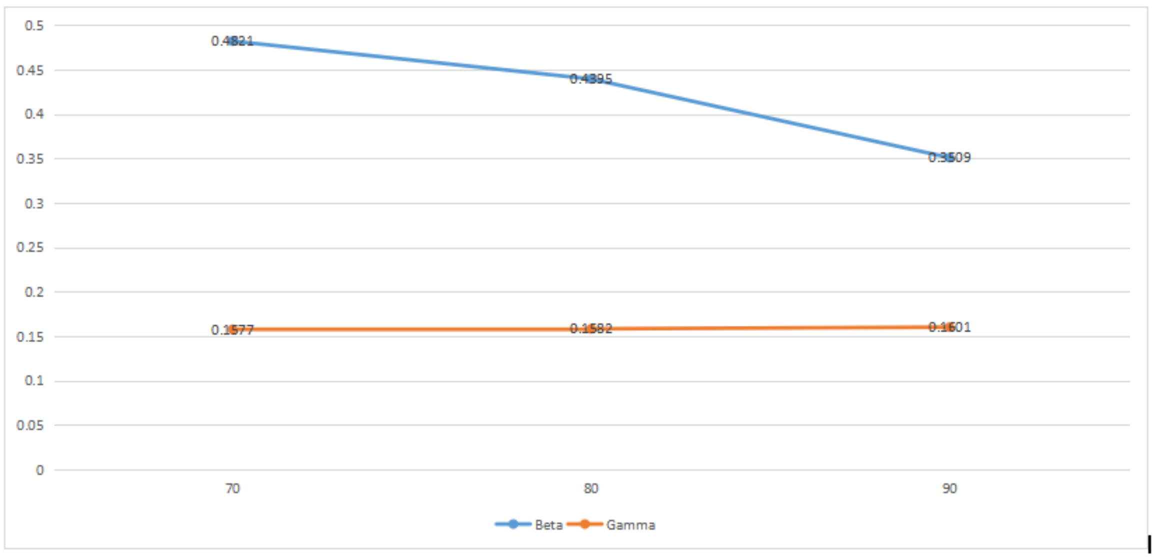

Figure 4.11: **Posterior Mean weights of beta and gamma at N=100 using Sigmoid Activation function**

Table 4.11 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using Sigmoid activation function at the sample size of 100. The table shows that β has a posterior weight of 0.4821, 0.4395 and 0.3509 and γ shows 0.1577, 0.1582 and 0.1601 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 4.2331, 4.1741 and 3.7577 and γ also has a posterior standard deviation of 4.3237, 4.2718 and 3.8493 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.1339, 0.132 and 0.1183 and γ also shows NSE values of 0.1367, 0.1351 and 0.1217 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.11 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 100.

Table 4.12: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error using Sigmoid activation function at N=200.**

Training set	Parameters	results_pweight	results_pstd	results_nse
	β	0.23073	3.7703	0.11923
	γ	0.19497	3.68798	0.11662
	μ_β	0.65804	0.42854	0.01355
70	μ_γ	0.41915	0.68682	0.02172
	σ	16.73863	1.62234	0.0513
	σ_β	1.51529	0.75878	0.02399
	S_γ	0.00492	0.00242	8.00E-05
	π	0.00055	0.00057	2.00E-05
	β	0.22223	3.56154	0.11263
	γ	0.11241	3.4871	0.11027
	μ_β	0.67867	0.42495	0.01344
80	μ_γ	0.41915	0.68682	0.02172
	σ	21.8621	2.11892	0.06701
	σ_β	1.51529	0.75878	0.02399
	S_γ	0.00492	0.00242	8.00E-05
	π	0.00042	0.00043	1.00E-05
	β	0.20027	3.31203	0.10474
	γ	0.10778	3.2397	0.10245
	μ_β	0.69245	0.42258	0.01336
90	μ_γ	0.41915	0.68682	0.02172
	σ	28.26497	2.7395	0.08663
	σ_β	1.51529	0.75878	0.02399
	S_γ	0.00492	0.00242	8.00E-05
	π	0.00032	0.00034	1.00E-05

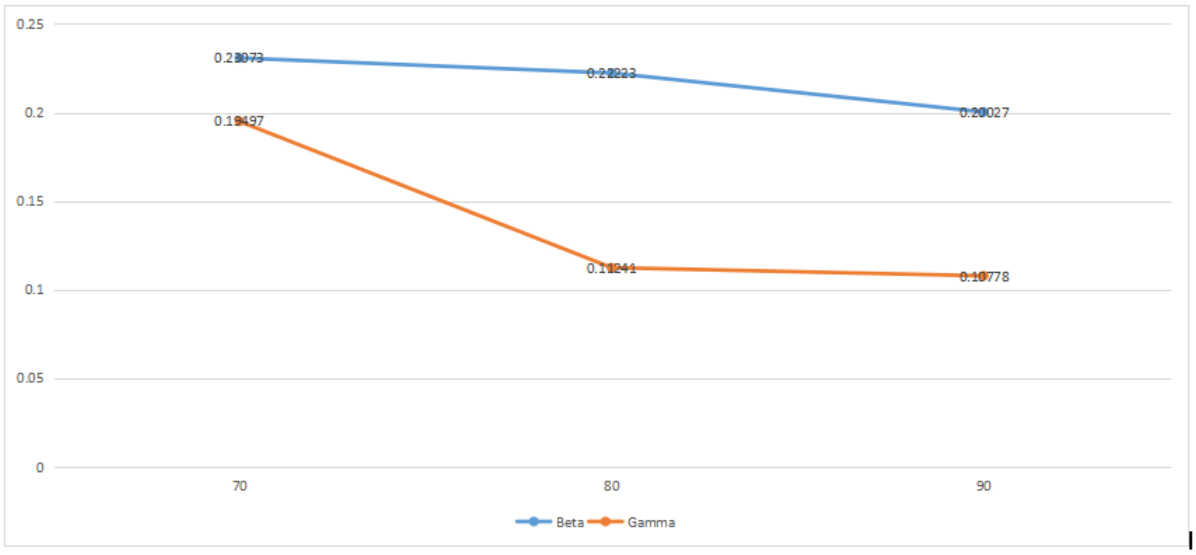

Figure 4.12: **Posterior Mean weights of beta and gamma at N=200 using Sigmoid Activation function**

Table 4.12 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using Sigmoid activation function at the sample size of 200. The table shows that β has a posterior weight of 0.2307, 0.2222 and 0.2002 and γ shows 0.1949, 0.1124 and 0.1077 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 3.7703, 3.5615 and 3.3120 and γ also has a posterior standard deviation of 3.6879, 3.4871 and 3.2397 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.1192, 0.1126 and 0.1047 and γ also shows NSE values of 0.1166, 0.1102 and 0.1024 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.12 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 200.

Table 4.13: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error using Sigmoid activation function at N=500.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.78159	2.20249	0.06965
	γ	0.75345	2.11861	0.067
	μ_β	0.75086	0.41334	0.01307
	μ_γ	0.42833	0.6905	0.02184
	σ	57.95334	3.58395	0.11333
	σ_β	1.52328	0.75835	0.02398
	S_γ	0.00485	0.00238	8.00E-05
	π	6.57E-05	6.80E-05	2.15E-06
80	β	0.69851	2.06025	0.06515
	γ	0.68108	1.97961	0.0626
	μ_β	0.75938	0.41185	0.01302
	μ_γ	0.42833	0.6905	0.02184
	σ	75.28146	4.65556	0.14722
	σ_β	1.52328	0.75835	0.02398
	S_γ	0.00485	0.00238	7.52E-05
	π	5.06E-05	5.24E-05	1.66E-06
90	β	0.62407	1.93499	0.06119
	γ	0.61545	1.85779	0.05875
	μ_β	0.76639	0.41062	0.01298
	μ_γ	0.42833	0.6905	0.02184
	σ	94.99439	5.87465	0.18577
	σ_β	1.52328	0.75835	0.02398
	S_γ	0.00485	0.00238	8.00E-05
	π	4.01E-05	4.15E-05	1.31E-06

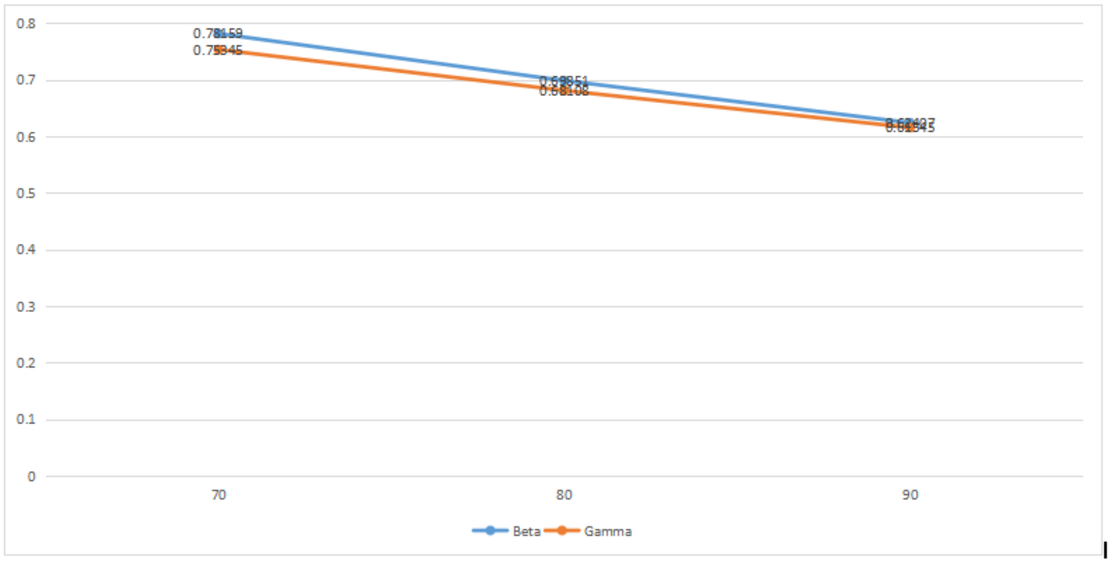

Figure 4.13: **Posterior Mean weights of beta and gamma at N=500 using Sigmoid Activation function**

Table 4.13 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using Sigmoid activation function at the sample size of 500. The table shows that β has a posterior weight of 0.7815, 0.6985 and 0.6240 and γ shows 0.7534, 0.6810 and 0.6154 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 2.2024, 2.0602 and 1.9349 and γ also has a posterior standard deviation of 2.1186, 1.9796 and 1.8577 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0696, 0.0651 and 0.0611 and γ also shows NSE values of 0.067, 0.0626 and 0.0587 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.13 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 500.

Table 4.14: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error using Sigmoid activation function at N=1000.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.41623	1.56511	0.04949
	γ	1.43441	1.49998	0.04743
	μ_β	0.78569	0.40621	0.01285
	μ_γ	0.42807	0.69204	0.02188
	σ	89.79078	3.93742	0.12451
	σ_β	1.52485	0.75682	0.02393
	S_γ	0.00483	0.00234	0.00007
	π	2.14E-05	2.21E-05	6.98E-07
80	β	0.35677	1.45543	0.04602
	γ	1.38161	1.39454	0.0441
	μ_β	0.79098	0.40536	0.01282
	μ_γ	0.42807	0.69204	0.02188
	σ	116.98538	5.12993	0.16222
	σ_β	1.52485	0.75682	0.02393
	S_γ	0.00483	0.00234	0.00007
	π	1.64E-05	1.70E-05	5.36E-07
90	β	0.31354	1.37997	0.04364
	γ	1.34362	1.32276	0.04183
	μ_β	0.79515	0.4047	0.0128
	μ_γ	0.42807	0.69204	0.02188
	σ	149.08055	6.53734	0.20673
	σ_β	1.52485	0.75682	0.02393
	S_γ	0.00483	0.00234	0.00007
	π	1.29E-05	1.33E-05	4.21E-07

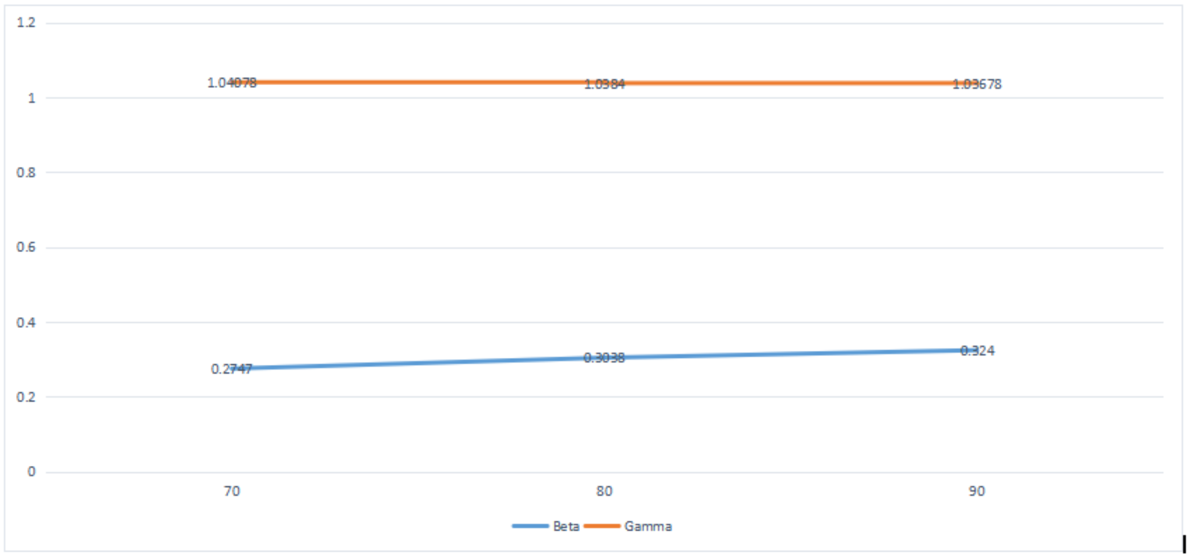

Figure 4.14: **Posterior Mean weights of beta and gamma at N=1000 using Sigmoid Activation function**

Table 4.14 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using Sigmoid activation function at the sample size of 1000. The table shows that β has a posterior weight of 0.4162, 0.3567 and 0.3135 and γ shows 1.4344, 1.3816 and 1.3436 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 1.5651, 1.4554 and 1.3799 and γ also has a posterior standard deviation of 1.4999, 1.3945 and 1.3227 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0494, 0.0460 and 0.0436 and γ also shows NSE values of 0.0474, 0.0441 and0.0418 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.14 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 1000.

Table 4.15: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error using Sigmoid activation function at N=5000.**

Training set	Parameters	results_pweight	results_pstd	results_nse
	β	0.1144	0.76396	0.02416
	γ	1.05284	0.7566	0.02393
	μ_β	0.8221	0.40298	0.01274
70	μ_γ	0.42856	0.69764	0.02206
	σ	492.35155	9.72218	0.30744
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	0.00007
	π	7.92E-07	8.13E-07	2.57E-08
	β	0.1637	0.7122	0.02252
	γ	1.04892	0.70557	0.02231
	μ_β	0.82289	0.40293	0.01274
80	μ_γ	0.42856	0.69764	0.02206
	σ	642.73478	12.69172	0.40135
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	0.00007
	π	6.06E-07	6.23E-07	1.97E-08
	β	0.1923	0.67398	0.02131
	γ	1.04683	0.66768	0.02111
	μ_β	0.82351	0.4029	0.01274
90	μ_γ	0.42856	0.69764	0.02206
	σ	810.85186	16.01143	0.50633
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	0.00007
	π	4.81E-07	4.94E-07	1.56E-08

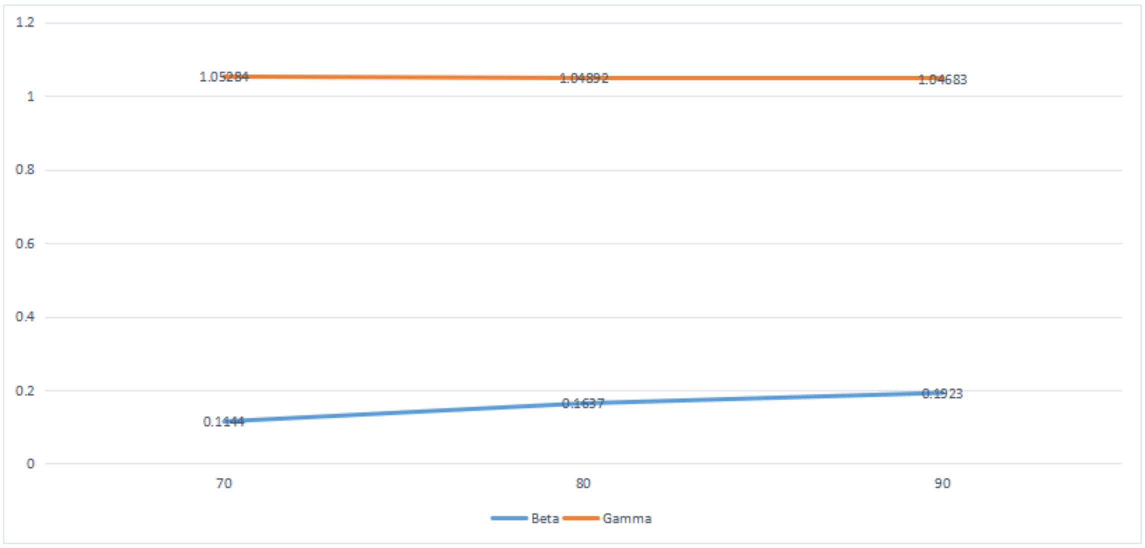

Figure 4.15: **Posterior Mean weights of beta and gamma at N=5000 using Sigmoid Activation function**

Table 4.15 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using Sigmoid activation function at the sample size of 5000. The table shows that β has a posterior weight of 0.1144, 0.1637 and 0.1923 and γ shows 1.0528, 1.0489 and 1.0468 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.7639, 0.7122 and 0.6739 and γ also has a posterior standard deviation of 0.7566, 0.7055 and 0.6676 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0241, 0.0225 and 0.0213 and γ also shows NSE values of 0.0239, 0.0223 and 0.0211 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.15 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 5000.

Table 4.16: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error using Sigmoid activation function at N=10000.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.2747	0.5742	0.01816
	γ	1.04078	0.56926	0.018
	μ_β	0.82525	0.40286	0.01274
	μ_γ	0.42856	0.69764	0.02206
	σ	936.39412	13.07632	0.41351
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	0.00007
	π	2.08E-07	2.14E-07	6.77E-09
80	β	0.3038	0.53497	0.01692
	γ	1.0384	0.53063	0.01678
	μ_β	0.82567	0.40286	0.01274
	μ_γ	0.42856	0.69764	0.02206
	σ	1228.65012	17.15754	0.54257
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	0.00007
	π	1.59E-07	1.63E-07	5.16E-09
90	β	0.324	0.50434	0.01595
	γ	1.03678	0.50041	0.01582
	μ_β	0.82599	0.40287	0.01274
	μ_γ	0.42856	0.69764	0.02206
	σ	1559.79981	21.78189	0.6888
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	0.00007
	π	1.25E-07	1.28E-07	4.06E-09

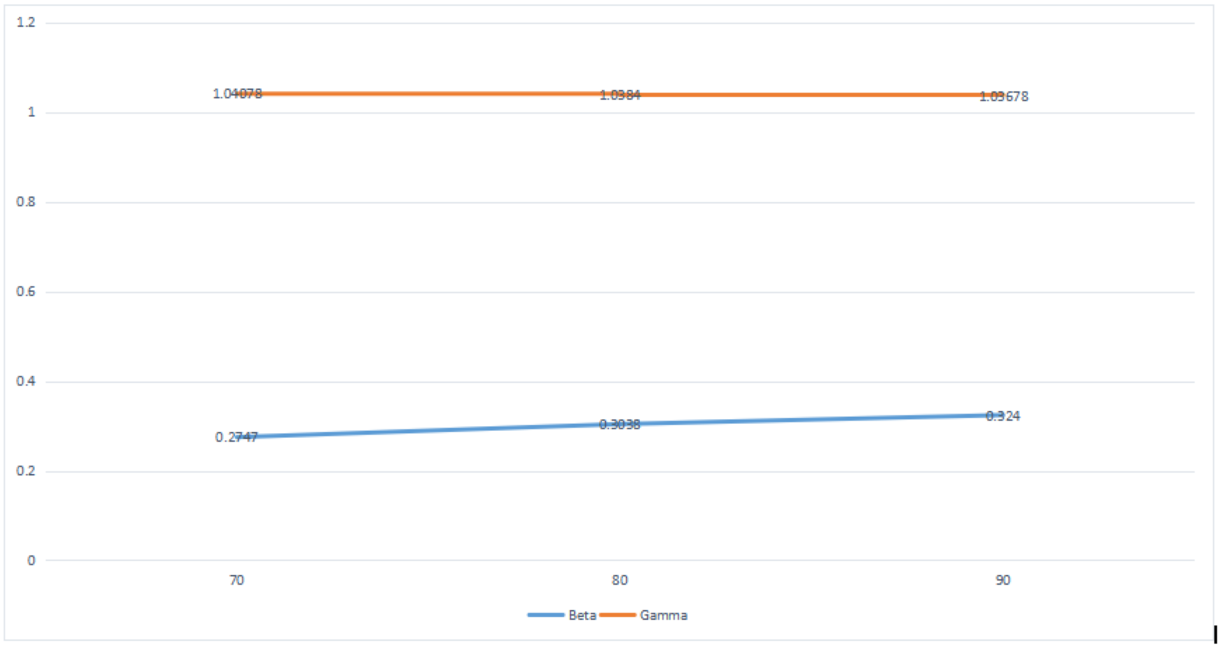

Figure 4.16: **Posterior Mean weights of beta and gamma at N=10000 using Sigmoid Activation function**

Table 4.16 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using Sigmoid activation function at the sample size of 10000. The table shows that β has a posterior weight of 0.2747, 0.3038 and 0.324 and γ shows 1.0407, 1.0384 and 1.0367 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.5742, 0.5349 and 0.5043 and γ also has a posterior standard deviation of 0.5692, 0.5306 and 0.5004 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0181, 0.0169 and 0.0159 and γ also shows NSE values of 0.018, 0.0167 and 0.0158 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.16 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 10000.

Table 4.17: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error using Sigmoid activation function at N=20000.**

Training set	Parameters	results_pweight	results_pstd	results_nse
	β	0.3982	0.4536	0.01434
	γ	1.03059	0.45243	0.01431
	μ_β	0.82697	0.40292	0.01274
70	μ_γ	0.42856	0.69764	0.02206
	σ	2452.05682	24.21357	0.7657
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	0.00007
	π	3.98E-08	4.09E-08	1.29E-09
	β	0.4064	0.42387	0.0134
	γ	1.03003	0.4228	0.01337
	μ_β	0.82717	0.40294	0.01274
80	μ_γ	0.42856	0.69764	0.02206
	σ	3205.55708	31.65424	1.00099
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	0.00007
	π	3.04E-08	3.13E-08	9.89E-10
	β	0.4119	0.39873	0.01261
	γ	1.02967	0.39767	0.01258
	μ_β	0.82733	0.40296	0.01274
90	μ_γ	0.42856	0.69764	0.02206
	σ	4054.9759	40.04208	1.26624
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	0.00007
	π	2.40E-08	2.47E-08	7.82E-10

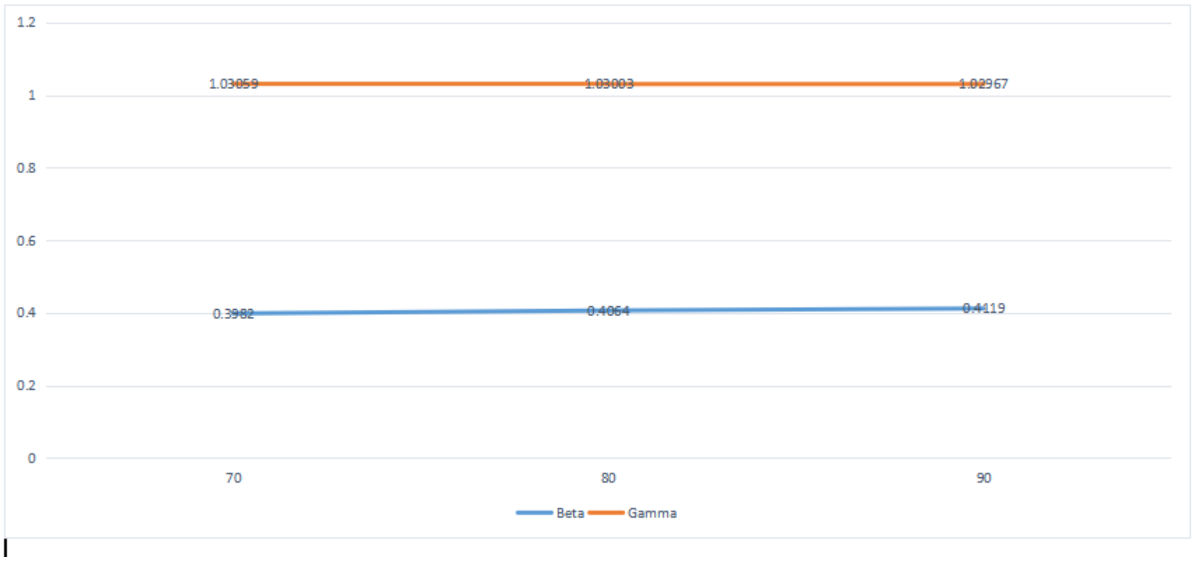

Figure 4.17: **Posterior Mean weights of beta and gamma at N=20000 using Sigmoid Activation function**

Table 4.17 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using Sigmoid activation function at the sample size of 20000. The table shows that β has a posterior weight of 0.3982, 0.4064 and 0.4119 and γ shows 1.0305, 1.0300 and 1.0296 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.4536, 0.4238 and 0.3987 and γ also has a posterior standard deviation of 0.4524, 0.4228 and 0.3976 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0143, 0.0134 and 0.0126 and γ also shows NSE values of 0.0143, 0.0133 and 0.0125 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.17 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 20000.

Table 4.18: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error using Sigmoid activation function at N=50000.**

Training set	Parameters	results_pweight	results_pstd	results_nse
	β	0.4476	0.27344	0.00865
	γ	1.02691	0.27322	0.00864
	μ_β	0.82797	0.40306	0.01275
70	μ_γ	0.42856	0.69764	0.02206
	σ	5548.14404	34.65024	1.09574
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	0.00007
	π	7.03E-09	7.23E-09	2.29E-10
	β	0.4508	0.25517	0.00807
	γ	1.0267	0.25496	0.00806
	μ_β	0.82806	0.40308	0.01275
80	μ_γ	0.42856	0.69764	0.02206
	σ	7246.12536	45.25477	1.43108
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	0.00007
	π	5.38E-09	5.53E-09	1.75E-10
	β	0.4534	0.2407	0.00761
	γ	1.02652	0.2405	0.00761
	μ_β	0.82813	0.4031	0.01275
90	μ_γ	0.42856	0.69764	0.02206
	σ	9158.65766	57.19925	1.8088
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	0.00007
	π	4.26E-09	4.38E-09	1.38E-10

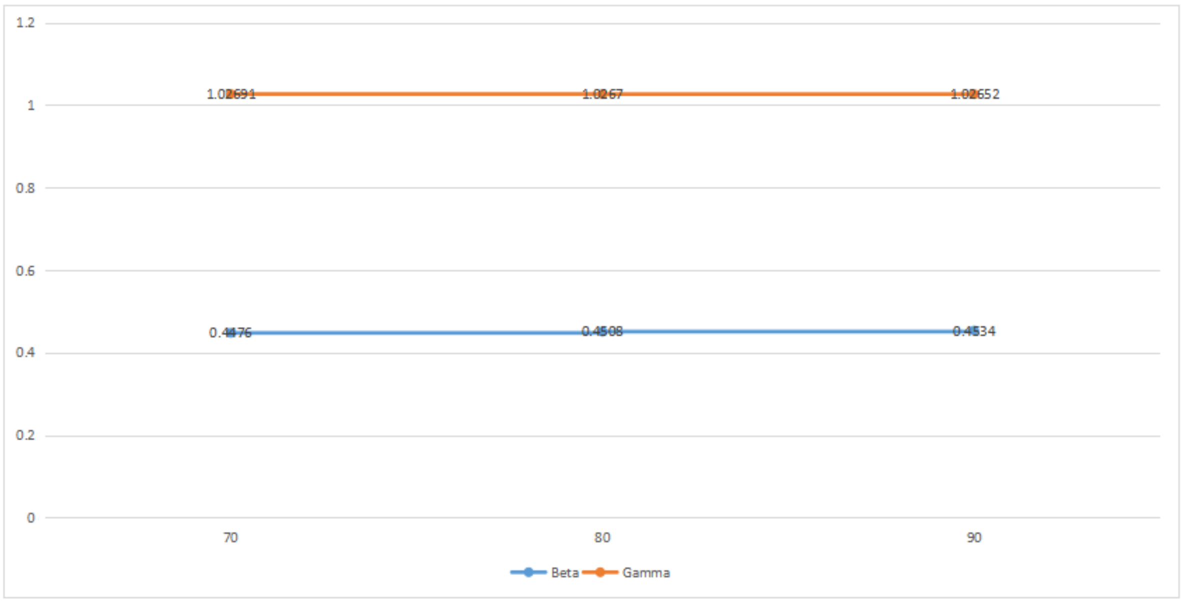

Figure 4.18: **Posterior Mean weights of beta and gamma at N=50000 using Sigmoid Activation function**

Table 4.18 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using Sigmoid activation function at the sample size of 50000. The table shows that β has a posterior weight of 0.4476, 0.4508 and 0.4534 and γ shows 1.0269, 1.0267 and 1.0265 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.2734, 0.2551 and 0.2407 and γ also has a posterior standard deviation of 0.2732, 0.2549 and 0.2405 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0086, 0.0080 and 0.0076 and γ also shows NSE values of 0.0086, 0.0080 and 0.0076 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.18 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 50000.

4.4 BAYESIAN RESULTS FOR TANGENT-SIGMOID ACTIVATION FUNCTION

The results for the parameters of the model using Bayesian approach with Tangent-Sigmoid activation function are displayed in this sesction. These results are tabled under the headings of Posterior Mean Weight, Posterior Standard Deviation and the Numerical Standard Error (NSE). The results are obtained for the Tangent-Sigmoid activation functions at 70%, 80% and 90% training sets and at sample size of 50, 100, 200, 500, 1000, 5000, 10000, 20000 and 50000.

Table 4.19: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error using Tangent-Sigmoid activation function at N=50.**

Training set	Parameters	results_pweight	results_pstd	results_nse
	β	0.54336	1.40993	0.04459
	γ	0.32396	1.55575	0.0492
	μ_β	0.14589	0.51756	0.01637
70	μ_γ	0.4158	0.6875	0.02174
	σ	0.50761	0.09562	0.00302
	σ_β	1.51024	0.76269	0.02412
	S_γ	0.00492	0.00243	0.00008
	π	0.11057	0.11303	0.00357
	β	0.38149	1.32814	0.042
	γ	0.75955	1.44937	0.04583
	μ_β	0.21439	0.5059	0.016
80	μ_γ	0.4158	0.6875	0.02174
	σ	0.4106	0.07734	0.00245
	σ_β	1.51024	0.76269	0.02412
	S_γ	0.00492	0.00243	0.00008
	π	0.16447	0.16813	0.00532
	β	0.34107	1.29615	0.04099
	γ	0.7237	1.40873	0.04455
	μ_β	0.26044	0.49824	0.01576
90	μ_γ	0.4158	0.6875	0.02174
	σ	0.35723	0.06729	0.00213
	σ_β	1.51024	0.76269	0.02412
	S_γ	0.00492	0.00243	0.00008
	π	0.22474	0.22974	0.00726

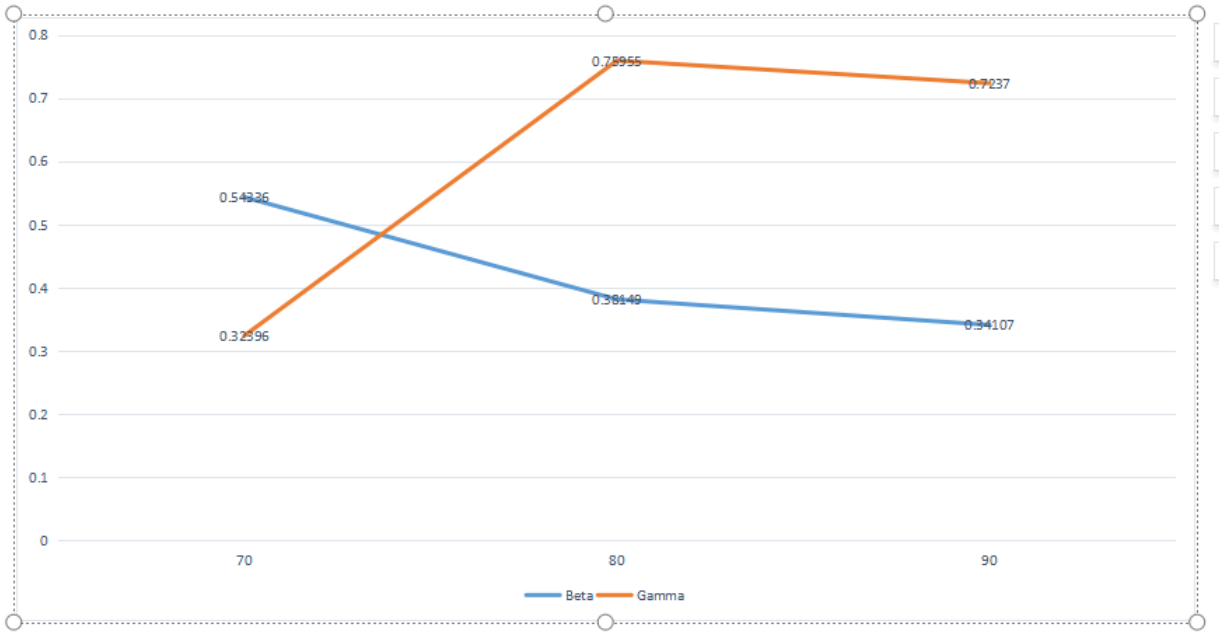

Figure 4.19: **Posterior Mean weights of beta and gamma at N=50 using Tangent-Sigmoid Activation function**

Table 4.19 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using Tangent-Sigmoid activation function at the sample size of 50. The table shows that β has a posterior weight of 0.5433, 0.3814 and 0.3410 and γ shows 0.3239, 0.7595 and 0.7237 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 1.4099, 1.3281 and 1.2961 and γ also has a posterior standard deviation of 1.5557, 1.4493 and 1.4087 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0445, 0.042 and 0.0409 and γ also shows NSE values of 0.0492, 0.0458 and 0.0445 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.19 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 50.

Table 4.20: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error for Tangent-Sigmoid activation function at N=100.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.3785	0.48118	0.01522
	γ	0.41674	0.68721	0.02173
	μ_β	0.93885	0.12751	0.00403
	μ_γ	1.51679	0.7589	0.024
	σ	0.00489	0.00239	0.00008
	σ_β	0.02149	0.02221	0.0007
	S_γ	0.39865	1.19626	0.03783
	π	0.15354	1.31271	0.04151
80	β	0.41123	0.47605	0.01505
	γ	0.41674	0.68721	0.02173
	μ_β	1.07891	0.14653	0.00463
	μ_γ	1.51679	0.7589	0.024
	σ	0.00489	0.00239	0.00008
	σ_β	0.01837	0.01899	0.0006
	S_γ	0.35377	1.107	0.03501
	π	0.26148	1.21713	0.03849
90	β	0.46138	0.46737	0.01478
	γ	0.41674	0.68721	0.02173
	μ_β	1.35325	0.18379	0.00581
	μ_γ	1.51679	0.7589	0.024
	σ	0.00489	0.00239	0.00008
	σ_β	0.01431	0.01479	0.00047
	S_γ	0.00489	0.00239	7.57E-05
	π	0.01430	0.01478	0.00047

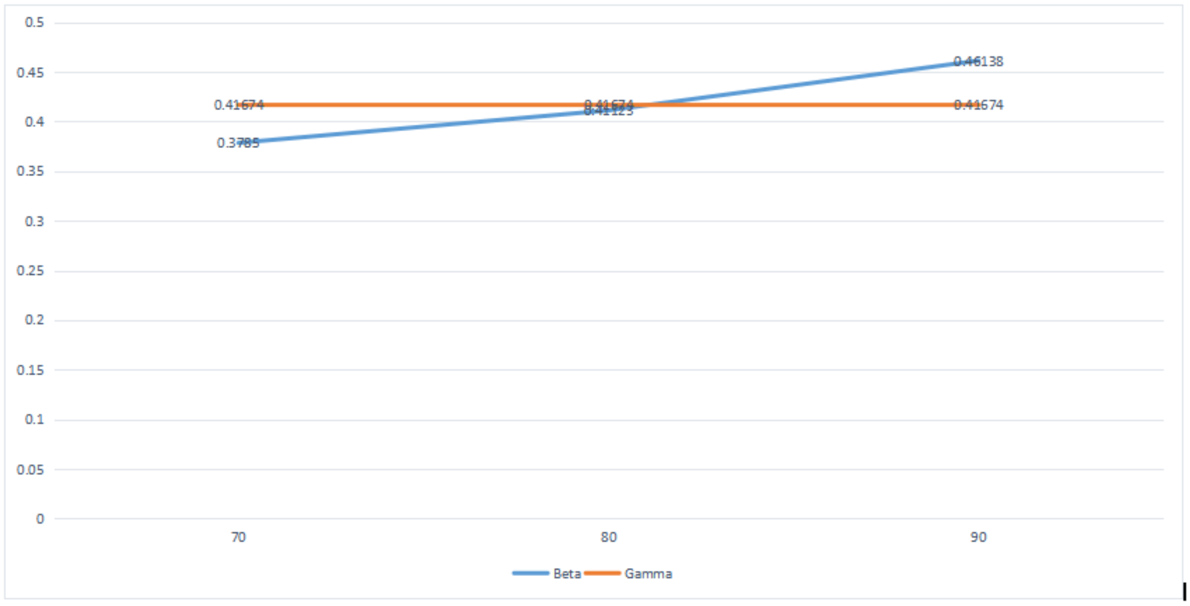

Figure 4.20: **Posterior Mean weights of beta and gamma at N=100 using Tangent-Sigmoid Activation function**

Table 4.20 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using Tangent-Sigmoid activation function at the sample size of 100. The table shows that β has a posterior weight of 0.3785, 0.4112 and 0.4613 and γ shows 0.4167, 0.4167 and 0.4167 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.4811, 0.4760 and 0.4673 and γ also has a posterior standard deviation of 0.6872, 0.6872 and 0.6872 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0152, 0.0150 and 0.0147 and γ also shows NSE values of 0.0217, 0.0217 and 0.0217 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.20 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 100.

Table 4.21: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error for Tangent-Sigmoid activation function at N=200.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.53087	0.45065	0.01425
	γ	0.41915	0.68682	0.02172
	μ_β	4.79492	0.46473	0.0147
	μ_γ	1.51529	0.75878	0.02399
	σ	0.00492	0.00242	0.00008
	σ_β	0.00193	0.002	0.00006
	S_γ	0.47254	1.03049	0.03259
	π	1.22088	0.9495	0.03003
80	β	0.56586	0.44458	0.01406
	γ	0.41915	0.68682	0.02172
	μ_β	6.32828	0.61335	0.0194
	μ_γ	1.51529	0.75878	0.02399
	σ	0.00492	0.00242	0.00008
	σ_β	0.00146	0.00151	0.00005
	S_γ	0.43477	0.9632	0.03046
	π	1.21092	0.88202	0.02789
90	β	0.58797	0.44073	0.01394
	γ	0.41915	0.68682	0.02172
	μ_β	8.41592	0.81569	0.02579
	μ_γ	1.51529	0.75878	0.02399
	σ	0.00492	0.00242	0.00008
	σ_β	0.0011	0.00113	0.00004
	S_γ	0.00492	0.00242	7.65E-05
	π	0.00109	0.00113	3.58E-05

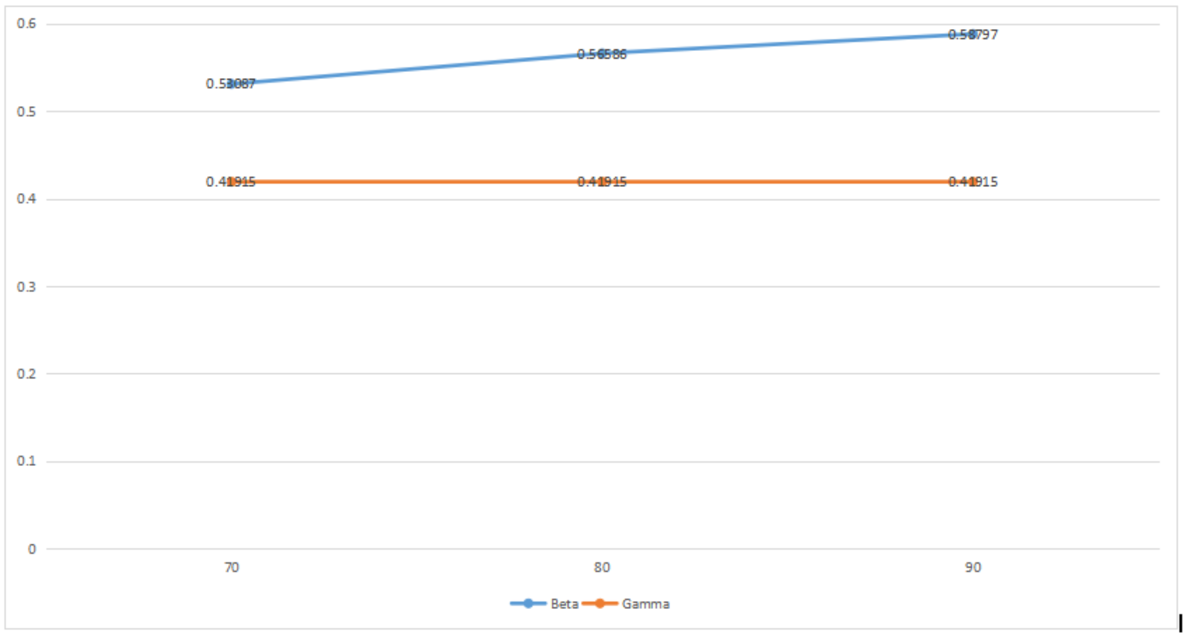

Figure 4.21: **Posterior Mean weights of beta and gamma at N=200 using Tangent-Sigmoid Activation function**

Table 4.21 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using Tangent-Sigmoid activation function at the sample size of 200. The table shows that β has a posterior weight of 0.5308, 0.5658 and 0.5879 and γ shows 0.4191, 0.4191 and 0.4191 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.4506, 0.4445 and 0.4407 and γ also has a posterior standard deviation of 0.6868, 0.6868 and 0.6868 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0142, 0.0140 and 0.0139 and γ also shows NSE values of 0.0217, 0.0217 and 0.0217 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.21 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 200.

Table 4.22: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error for Tangent-Sigmoid activation function at N=500.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.31947	0.69746	0.02206
	γ	1.22421	0.5943	0.01879
	μ_β	0.69748	0.423	0.01338
	μ_γ	0.42833	0.6905	0.02184
	σ	27.60052	1.70687	0.05398
	σ_β	1.52328	0.75835	0.02398
	S_γ	0.00485	0.00238	0.00008
	π	0.00014	0.00014	4.52E-06
80	β	0.28237	0.65263	0.02064
	γ	1.20467	0.55326	0.0175
	μ_β	0.71145	0.42047	0.0133
	μ_γ	0.42833	0.6905	0.02184
	σ	35.93071	2.22203	0.07027
	σ_β	1.52328	0.75835	0.02398
	S_γ	0.00485	0.00238	0.00008
	π	0.00011	0.00011	3.47E-06
90	β	0.24862	0.61076	0.01931
	γ	1.18479	0.51562	0.01631
	μ_β	0.72248	0.41846	0.01323
	μ_γ	0.42833	0.6905	0.02184
	σ	44.98916	2.78222	0.08798
	σ_β	1.52328	0.75835	0.02398
	S_γ	0.00485	0.00238	0.00008
	π	8.46E-05	8.76E-05	2.77E-06

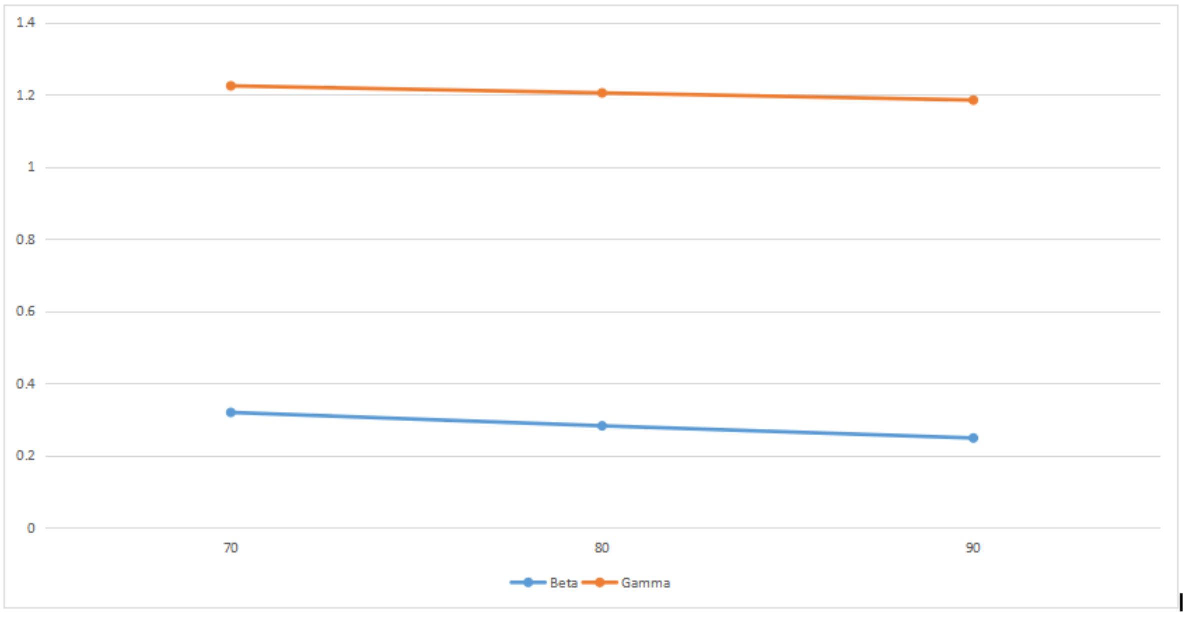

Figure 4.22: **Posterior Mean weights of beta and gamma at N=500 using Tangent-Sigmoid Activation function**

Table 4.22 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using Tangent-Sigmoid activation function at the sample size of 500. The table shows that β has a posterior weight of 0.3194, 0.2823 and 0.2486 and γ shows 1.2242, 1.2046 and 1.1847 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.6974, 0.6526 and 0.6107 and γ also has a posterior standard deviation of 0.5943, 0.5532 and 0.5156 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0220, 0.0206 and 0.0193 and γ also shows NSE values of 0.0187, 0.0175 and 0.0163 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.22 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 500.

Table 4.23: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error for Tangent-Sigmoid activation function at N=1000.**

Training set	Parameters	results_pweight	results_pstd	results_nse
	β	0.14197	0.48951	0.01548
	γ	1.12918	0.41395	0.01309
	μ_β	0.76106	0.4103	0.01297
70	μ_γ	0.42807	0.69204	0.02188
	σ	40.29244	1.76687	0.05587
	σ_β	1.52485	0.75682	0.02393
	S_γ	0.00483	0.00234	7.40E-05
	π	4.76E-05	4.92E-05	1.56E-06
	β	0.11823	0.45588	0.01442
	γ	1.11572	0.38506	0.01218
	μ_β	0.76923	0.4089	0.01293
80	μ_γ	0.42807	0.69204	0.02188
	σ	52.435	2.29933	0.07271
	σ_β	1.52485	0.75682	0.02393
	S_γ	0.00483	0.00234	7.40E-05
	π	3.66E-05	3.78E-05	1.20E-06
	β	0.10143	0.43244	0.01367
	γ	1.10661	0.36538	0.01155
	μ_β	0.77534	0.40787	0.0129
90	μ_γ	0.42807	0.69204	0.02188
	σ	66.41551	2.91239	0.0921
	σ_β	1.52485	0.75682	0.02393
	S_γ	0.00483	0.00234	7.40E-05
	π	2.89E-05	2.99E-05	9.44E-07

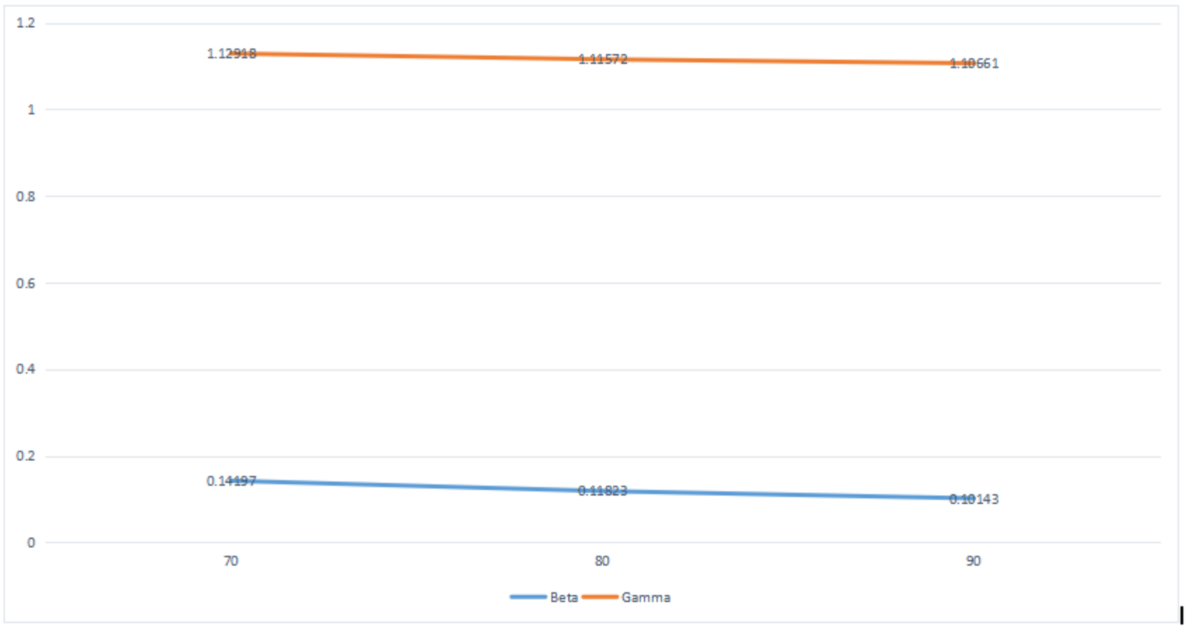

Figure 4.23: **Posterior Mean weights of beta and gamma at N=1000 using Tangent-Sigmoid Activation function**

Table 4.23 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using Tangent-Sigmoid activation function at the sample size of 1000. The table shows that β has a posterior weight of 0.1419, 0.1182 and 0.1014 and γ shows 1.1291, 1.1157 and 1.1066 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.4895, 0.4558 and 0.4324 and γ also has a posterior standard deviation of 0.4139, 0.3850 and 0.3653 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0154, 0.0144 and 0.0136 and γ also shows NSE values of 0.0130, 0.0121 and 0.0115 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.23 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 1000.

Table 4.24: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error for Tangent-Sigmoid activation function at N=5000.**

Training set	Parameter	results_pweight	results_pstd	results_nse
70	β	0.2998	0.23607	0.00747
	γ	1.0287	0.22653	0.00716
	μ_β	0.81764	0.40334	0.01275
	μ_γ	0.42856	0.69764	0.02206
	σ	199.04132	3.93035	0.12429
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	0.00007
	π	1.96E-06	2.01E-06	6.36E-08
80	β	0.325	0.21976	0.00695
	γ	1.02787	0.21116	0.00668
	μ_β	0.81898	0.40322	0.01275
	μ_γ	0.42856	0.69764	0.02206
	σ	259.12274	5.11675	0.16181
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	1.50E-06	1.55E-06	4.89E-08
90	β	0.341	0.20806	0.00658
	γ	1.02759	0.19991	0.00632
	μ_β	0.82003	0.40313	0.01275
	μ_γ	0.42856	0.69764	0.02206
	σ	327.51623	6.46728	0.20451
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	1.19E-06	1.22E-06	3.87E-08

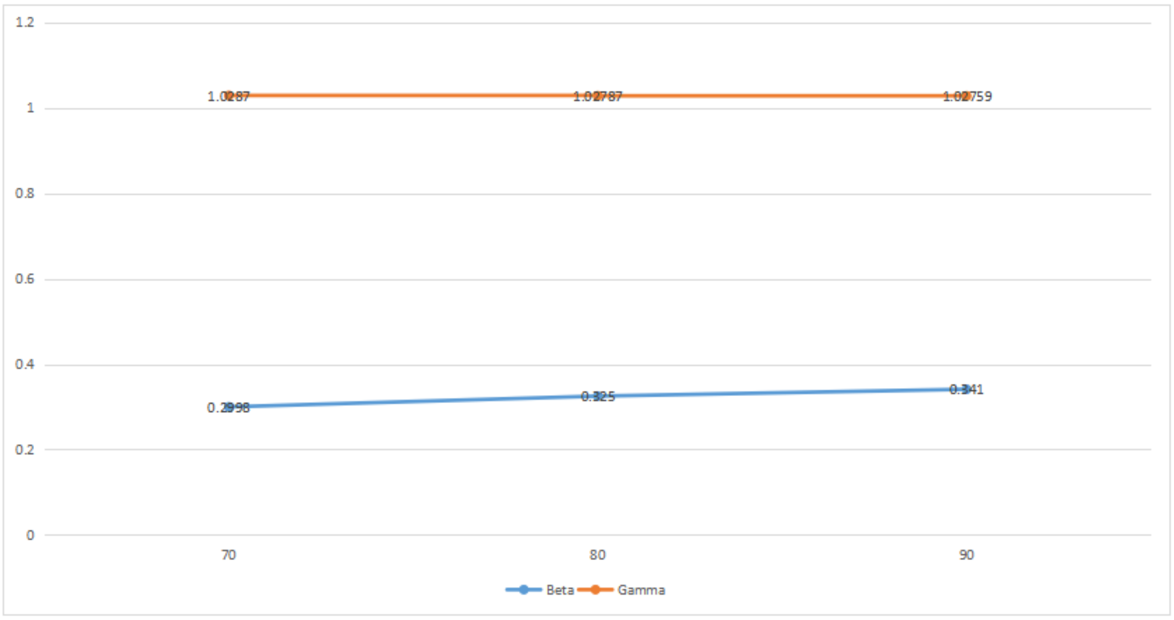

Figure 4.24: **Posterior Mean weights of beta and gamma at N=5000 using Tangent-Sigmoid Activation function**

Table 4.24 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using Tangent-Sigmoid activation function at the sample size of 5000. The table shows that β has a posterior weight of 0.2998, 0.325 and 0.341 and γ shows 1.0287, 1.0278 and 1.0275 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.2360, 0.2197 and 0.2080 and γ also has a posterior standard deviation of 0.2265, 0.2111 and 0.1999 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0074, 0.0069 and 0.0065 and γ also shows NSE values of 0.0071, 0.0066 and 0.0063 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.24 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 5000.

Table 4.25: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error for Tangent-Sigmoid activation function at N=10000.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.38699	0.17482	0.00553
	γ	1.02645	0.16857	0.00533
	μ_β	0.82279	0.40294	0.01274
	μ_γ	0.42856	0.69764	0.02206
	σ	348.79995	4.87083	0.15403
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	5.59E-07	5.74E-07	1.82E-08
80	β	0.40106	0.16273	0.00515
	γ	1.02595	0.15722	0.00497
	μ_β	0.8235	0.40291	0.01274
	μ_γ	0.42856	0.69764	0.02206
	σ	457.94558	6.395	0.20223
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	4.26E-07	4.38E-07	1.38E-08
90	β	0.41095	0.1535	0.00485
	γ	1.02565	0.14849	0.0047
	μ_β	0.82406	0.40289	0.01274
	μ_γ	0.42856	0.69764	0.02206
	σ	580.79302	8.11051	0.25648
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	3.36E-07	3.45E-07	1.09E-08

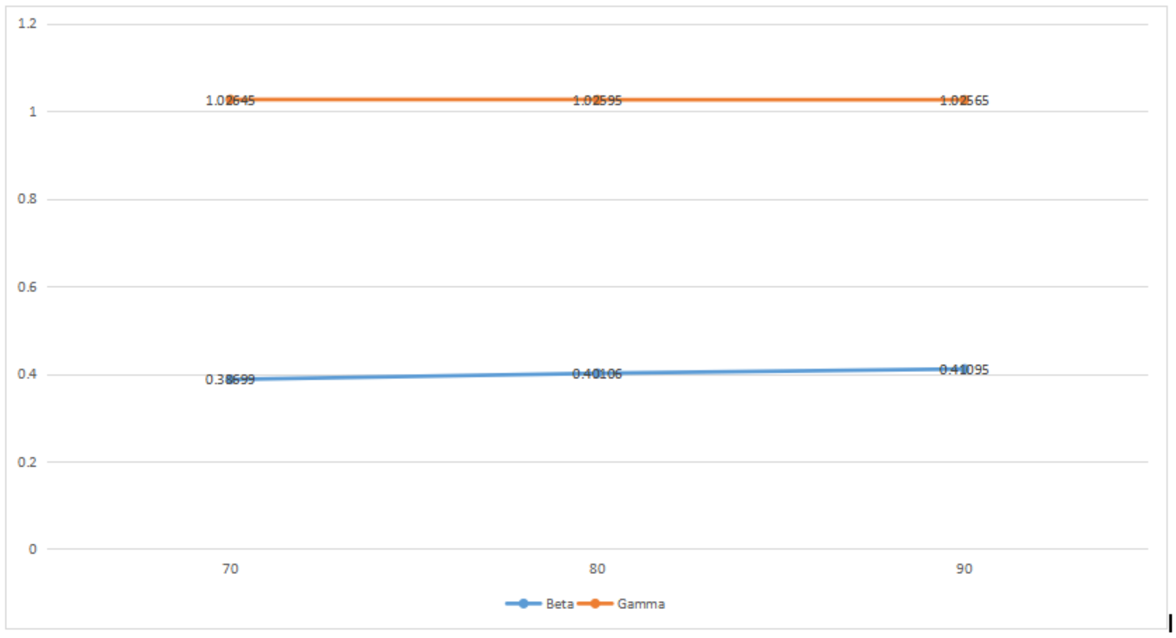

Figure 4.25: **Posterior Mean weights of beta and gamma at N=10000 using Tangent-Sigmoid Activation function**

Table 4.25 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using Tangent-Sigmoid activation function at the sample size of 10000. The table shows that β has a posterior weight of 0.3869, 0.4010 and 0.4109 and γ shows 1.0264, 1.0259 and 1.0256 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.1748, 0.1627 and 0.1535 and γ also has a posterior standard deviation of 0.1685, 0.1572 and 0.1484 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0055, 0.0051 and 0.0048 and γ also shows NSE values of 0.0053, 0.0049 and 0.0047 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.25 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 10000.

Table 4.26: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error for Tangent-Sigmoid activation function at N=20000.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.44547	0.13787	0.00436
	γ	1.02436	0.13614	0.0043
	μ_β	0.82577	0.40287	0.01274
	μ_γ	0.42856	0.69764	0.02206
	σ	1077.65099	10.64159	0.33652
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	9.05E-08	9.30E-08	2.94E-09
80	β	0.45015	0.1288	0.00407
	γ	1.02433	0.12721	0.00402
	μ_β	0.82612	0.40288	0.01274
	μ_γ	0.42856	0.69764	0.02206
	σ	1409.21619	13.91573	0.44005
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	6.92E-08	7.11E-08	2.25E-09
90	β	0.45342	0.12121	0.00383
	γ	1.02434	0.11965	0.00378
	μ_β	0.82639	0.40289	0.01274
	μ_γ	0.42856	0.69764	0.02206
	σ	1782.53574	17.60219	0.55663
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	5.47E-08	5.62E-08	1.78E-09

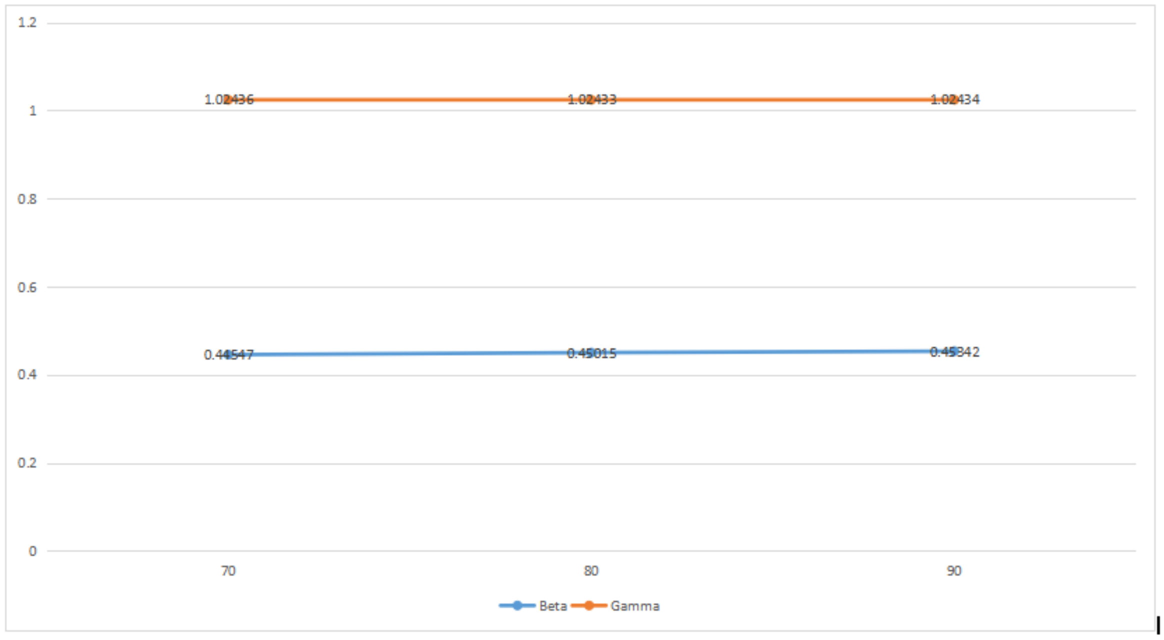

Figure 4.26: **Posterior Mean weights of beta and gamma at N=20000 using Tangent-Sigmoid Activation function**

Table 4.26 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using Tangent-Sigmoid activation function at the sample size of 20000. The table shows that β has a posterior weight of 0.4454, 0.4501 and 0.4534 and γ shows 1.02436, 1.02433 and 1.02434 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.1378, 0.1288 and 0.1212 and γ also has a posterior standard deviation of 0.1361, 0.1272 and 0.1196 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.00436, 0.0040 and 0.0038 and γ also shows NSE values of 0.0043, 0.00402 and 0.0037 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.26 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 20000.

Table 4.27: **Summary Table of the Posterior Mean, Standard Deviation and Numerical Standard Error for Tangent-Sigmoid activation function at N=50000.**

Training set	Parameters	results_pmean	results_pstd	results_nse
70	β	0.47175	0.08256	0.00261
	γ	1.02385	0.08221	0.0026
	μ_β	0.82746	0.40298	0.01274
	μ_γ	0.42856	0.69764	0.02206
	σ	2251.55746	14.06182	0.44467
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	1.73E-08	1.78E-08	5.63E-10
80	β	0.47348	0.07706	0.00244
	γ	1.02386	0.07672	0.00243
	μ_β	0.8276	0.403	0.01274
	μ_γ	0.42856	0.69764	0.02206
	σ	2941.96528	18.37368	0.58103
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	1.33E-08	1.36E-08	4.31E-10
90	β	0.4749	0.07266	0.0023
	γ	1.02386	0.07235	0.00229
	μ_β	0.82772	0.40302	0.01274
	μ_γ	0.42856	0.69764	0.02206
	σ	3708.71454	23.16231	0.73246
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	1.05E-08	1.08E-08	3.42E-10

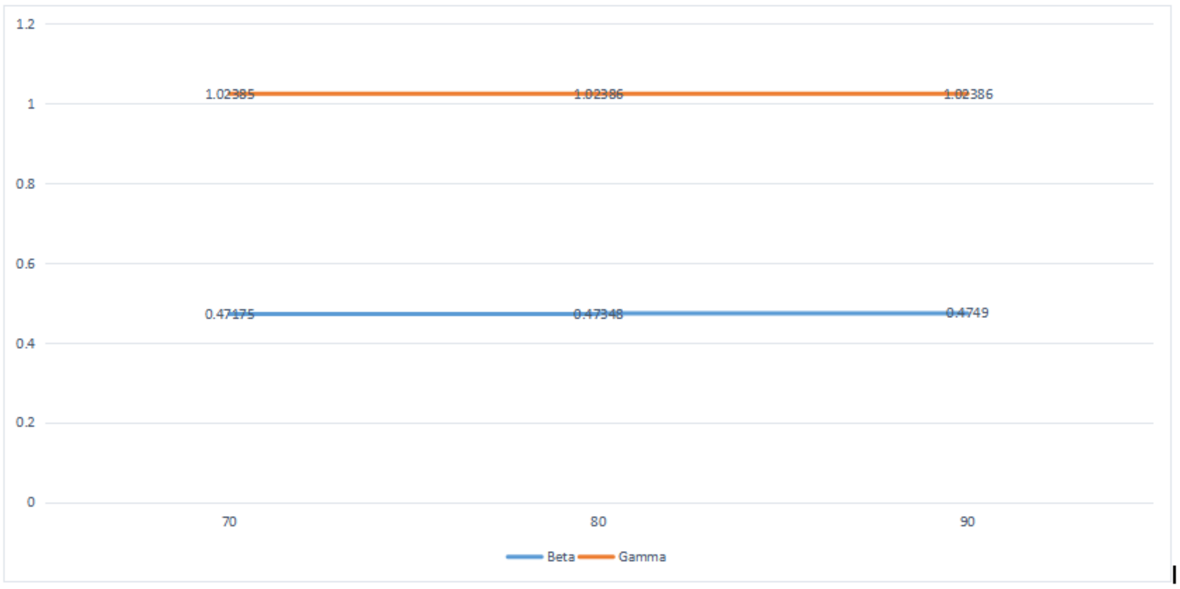

Figure 4.27: **Posterior Mean weights of beta and gamma at N=50000 using Tangent-Sigmoid Activation function**

Table 4.27 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using Tangent-Sigmoid activation function at the sample size of 50000. The table shows that β has a posterior weight of 0.4717, 0.4734 and 0.4749 and γ shows 1.02385, 1.02386 and 1.02386 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.0825, 0.0770 and 0.0726 and γ also has a posterior standard deviation of 0.0822, 0.0767 and 0.0723 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0026, 0.0024 and 0.0023 and γ also shows NSE values of 0.0026, 0.0024 and 0.0022 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.27 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 50000.

4.5 BAYESIAN RESULTS FOR SSLHT ACTIVATION FUNCTION

The results for the parameters of the model using Bayesian approach with Symmetric Saturating Linear Hyperbolic Tangent (SSLHT) activation function are displayed in this sesction. These results are tabled under the headings of Posterior Mean Weight, Posterior Standard Deviation and the Numerical Standard Error (NSE). The results are obtained for the SSLHT activation functions at 70%, 80% and 90% training sets and at sample size of 50, 100, 200, 500, 1000, 5000, 10000, 20000 and 50000.

Table 4.28: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error for SSLHT activation function at N=50.**

Training set	Parameters	results_pweight	results_pstd	results_nse
	β	0.82261	0.87856	0.02778
	γ	1.13206	0.84362	0.02668
	μ_β	0.77423	0.40767	0.01289
70	μ_γ	0.4158	0.6875	0.02174
	σ	75.44157	14.21069	0.44938
	σ_β	1.51024	0.76269	0.02412
	S_γ	0.00492	0.00243	8.00E-05
	π	0.00044	0.00044	1.00E-05
	β	0.7481	0.81938	0.02591
	γ	1.09355	0.7945	0.02512
	μ_β	0.78162	0.40657	0.01286
80	μ_γ	0.4158	0.6875	0.02174
	σ	99.46493	18.73589	0.59248
	σ_β	1.51024	0.76269	0.02412
	S_γ	0.00492	0.00243	8.00E-05
	π	0.00033	0.00034	1.00E-05
	β	0.6068	0.80012	0.0253
	γ	1.08513	0.77801	0.0246
	μ_β	0.78615	0.4059	0.01284
90	μ_γ	0.4158	0.6875	0.02174
	σ	126.63107	23.85309	0.7543
	σ_β	1.51024	0.76269	0.02412
	S_γ	0.00492	0.00243	8.00E-05
	π	0.00026	0.00026	1.00E-05

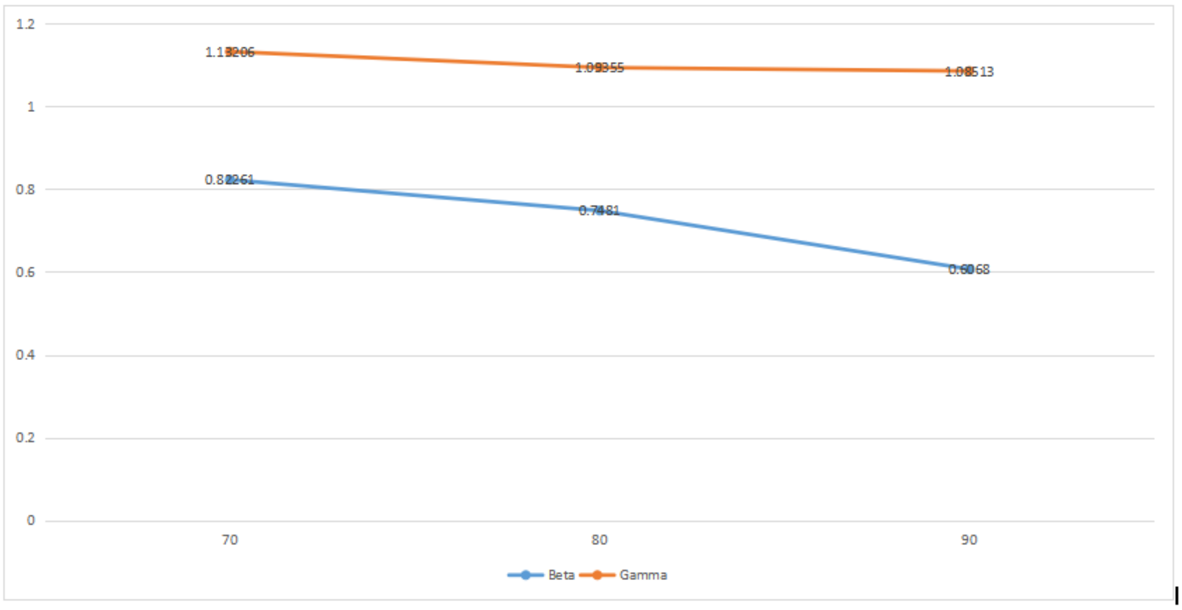

Figure 4.28: **Posterior Mean weights of beta and gamma at N=50 using SSLHT Activation function**

Table 4.28 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using SSLHT activation function at the sample size of 50. The table shows that β has a posterior weight of 0.8226, 0.7481 and 0.6068 and γ shows 1.1320, 1.0935 and 1.0851 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.8785, 0.8193 and 0.8001 and γ also has a posterior standard deviation of 0.8436, 0.7945 and 0.7780 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0277, 0.0259 and 0.0253 and γ also shows NSE values of 0.0266, 0.0251 and 0.0246 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.28 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 50.

Table 4.29: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error for SSLHT activation function at N=100.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.85297	0.78851	0.02493
	γ	1.09291	0.76564	0.02421
	μ_β	0.79915	0.4061	0.01284
	μ_γ	0.41674	0.68721	0.02173
	σ	203.3859	27.6222	0.87349
	σ_β	1.51679	0.7589	0.024
	S_γ	0.00489	0.00239	7.57E-05
	π	8.72E-05	9.02E-05	2.85E-06
80	β	0.85686	0.7827	0.02475
	γ	1.09991	0.75795	0.02397
	μ_β	0.80162	0.40576	0.01283
	μ_γ	0.41674	0.68721	0.02173
	σ	263.37708	35.76971	1.13114
	σ_β	1.51679	0.7589	0.024
	S_γ	0.00489	0.00239	7.57E-05
	π	6.74E-05	6.96E-05	2.20E-06
90	β	0.84803	0.74648	0.02361
	γ	1.09371	0.72278	0.02286
	μ_β	0.80371	0.40549	0.01282
	μ_γ	0.41674	0.68721	0.02173
	σ	322.00976	43.73272	1.38295
	σ_β	1.51679	0.7589	0.024
	S_γ	0.00489	0.00239	7.57E-05
	π	5.51E-05	5.69E-05	1.80E-06

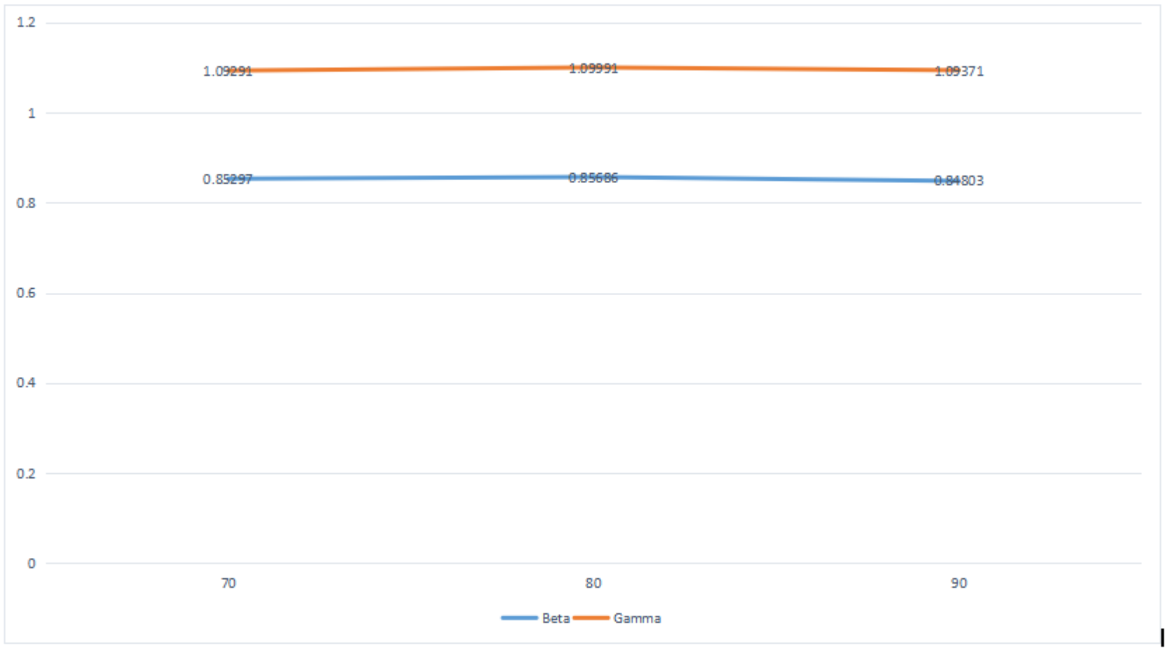

Figure 4.29: **Posterior Mean weights of beta and gamma at N=100 using SSLHT Activation function**

Table 4.29 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using SSLHT activation function at the sample size of 100. The table shows that β has a posterior weight of 0.8529, 0.8568 and 0.8480 and γ shows 1.0929, 1.0999 and 1.0937 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.7885, 0.7827 and 0.7464 and γ also has a posterior standard deviation of 0.7656, 0.7579 and 0.7227 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0249, 0.0247 and 0.0236 and γ also shows NSE values of 0.0242, 0.0239 and 0.0228 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.29 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 100.

Table 4.30: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error for SSLHT activation function at N=200.**

Training set	Parameters	results_pweight	results_pstd	results_nse
	β	0.80984	0.6171	0.01951
	γ	0.94416	0.64598	0.02043
	μ_β	0.80896	0.40553	0.01282
70	μ_γ	0.41915	0.68682	0.02172
	σ	422.09017	40.90982	1.29368
	σ_β	1.51529	0.75878	0.02399
	S_γ	0.00492	0.00242	7.65E-05
	π	2.17E-05	2.24E-05	7.10E-07
	β	0.99622	0.58828	0.0186
	γ	0.95684	0.61403	0.01942
	μ_β	0.81091	0.4052	0.01281
80	μ_γ	0.41915	0.68682	0.02172
	σ	549.5403	53.26255	1.68431
	σ_β	1.51529	0.75878	0.02399
	S_γ	0.00492	0.00242	0.00008
	π	1.67E-05	1.72E-05	5.45E-07
	β	0.92696	0.55245	0.01747
	γ	0.96552	0.57644	0.01823
	μ_β	0.81231	0.405	0.01281
90	μ_γ	0.41915	0.68682	0.02172
	σ	696.41129	67.49758	2.13446
	σ_β	1.51529	0.75878	0.02399
	S_γ	0.00492	0.00242	7.65E-05
	π	1.32E-05	1.36E-05	4.30E-07

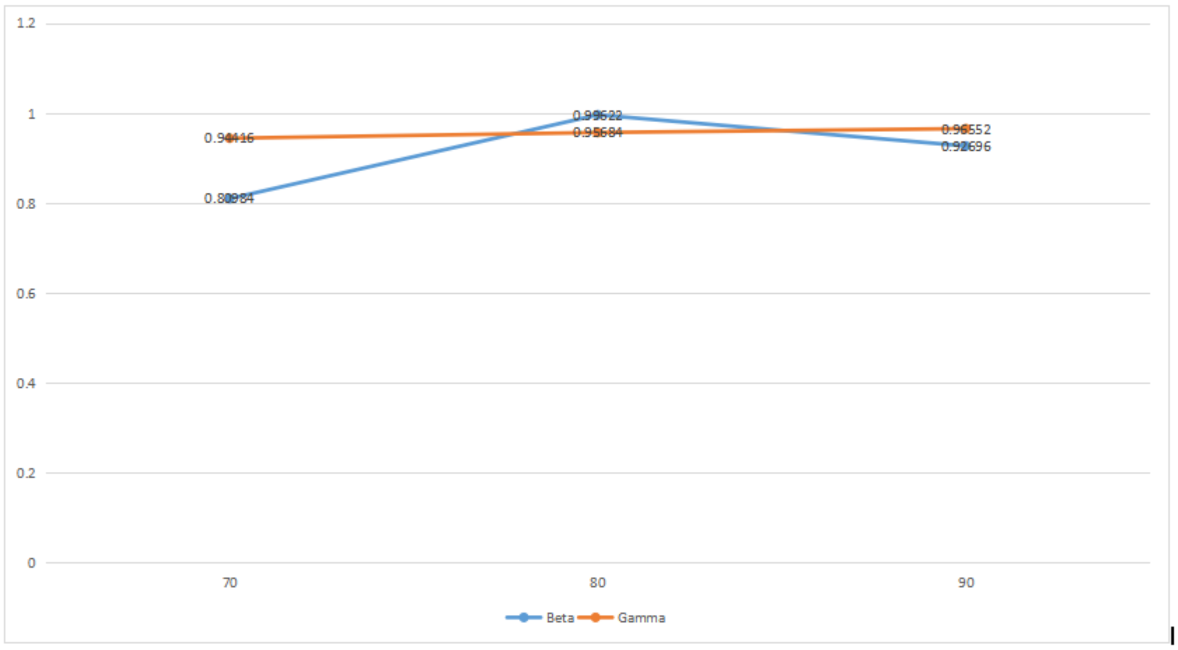

Figure 4.30: **Posterior Mean weights of beta and gamma at N=200 using SSLHT Activation function**

Table 4.30 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using SSLHT activation function at the sample size of 200. The table shows that β has a posterior weight of 0.8098, 0.9962 and 0.9269 and γ shows 0.9441, 0.9568 and 0.9655 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.6171, 0.5882 and 0.5524 and γ also has a posterior standard deviation of 0.6459, 0.6140 and 0.5764 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0195, 0.0186 and 0.0174 and γ also shows NSE values of 0.0204, 0.0194 and 0.0182 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.30 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 200.

Table 4.31: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error for SSLHT activation function at N=500.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.84006	0.42836	0.01355
	γ	0.98029	0.45088	0.01426
	μ_β	0.82438	0.40265	0.01273
	μ_γ	0.42833	0.6905	0.02184
	σ	1039.09264	64.25961	2.03207
	σ_β	1.52328	0.75835	0.02398
	S_γ	0.00485	0.00238	7.52E-05
	π	3.66E-06	3.79E-06	1.20E-07
80	β	0.80013	0.3992	0.01262
	γ	0.98525	0.42071	0.0133
	μ_β	0.82515	0.40255	0.01273
	μ_γ	0.42833	0.6905	0.02184
	σ	1352.98554	83.67139	2.64592
	σ_β	1.52328	0.75835	0.02398
	S_γ	0.00485	0.00238	7.52E-05
	π	2.81E-06	2.91E-06	9.21E-08
90	β	0.76064	0.369	0.01167
	γ	0.99	0.38946	0.01232
	μ_β	0.82579	0.40246	0.01273
	μ_γ	0.42833	0.6905	0.02184
	σ	1715.37513	106.0823	3.35462
	σ_β	1.52328	0.75835	0.02398
	S_γ	0.00485	0.00238	7.52E-05
	π	2.22E-06	2.30E-06	7.26E-08

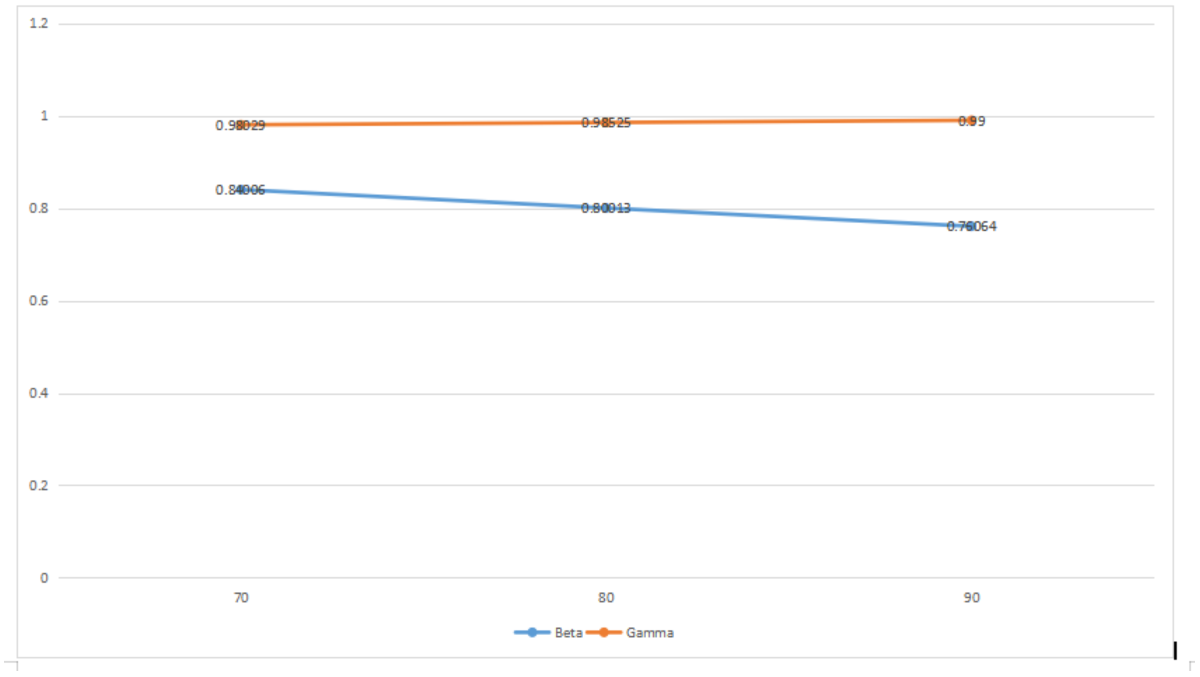

Figure 4.31: **Posterior Mean weights of beta and gamma at N=500 using SSLHT Activation function**

Table 4.31 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using SSLHT activation function at the sample size of 500. The table shows that β has a posterior weight of 0.8400, 0.8001 and 0.7606 and γ shows 0.9802, 0.9852 and 0.99 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.4283, 0.3992 and 0.369 and γ also has a posterior standard deviation of 0.4508, 0.4207 and 0.3894 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0135, 0.0126 and 0.0116 and γ also shows NSE values of 0.0142, 0.0133 and 0.0123 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.31 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 500.

Table 4.32: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error for SSLHT activation function at N=1000.**

Training set	Parameters	results_pweight	results_pstd	results_nse
	β	0.6672	0.2927	0.00926
	γ	1.00148	0.3106	0.00982
	μ_β	0.82547	0.40181	0.01271
70	μ_γ	0.42807	0.69204	0.02188
	σ	1927.52674	84.52405	2.67289
	σ_β	1.52485	0.75682	0.02393
	S_γ	0.00483	0.00234	7.40E-05
	π	9.95E-07	1.03E-06	3.25E-08
	β	0.64073	0.27239	0.00861
	γ	1.0047	0.28901	0.00914
	μ_β	0.82593	0.40176	0.0127
80	μ_γ	0.42807	0.69204	0.02188
	σ	2516.77188	110.36307	3.48999
	σ_β	1.52485	0.75682	0.02393
	S_γ	0.00483	0.00234	7.40E-05
	π	7.62E-07	7.88E-07	2.49E-08
	β	0.61905	0.25691	0.00812
	γ	1.00735	0.27228	0.00861
	μ_β	0.8263	0.4017	0.0127
90	μ_γ	0.42807	0.69204	0.02188
	σ	3201.31385	140.38094	4.43924
	σ_β	1.52485	0.75682	0.02393
	S_γ	0.00483	0.00234	7.40E-05
	π	5.99E-07	6.19E-07	1.96E-08

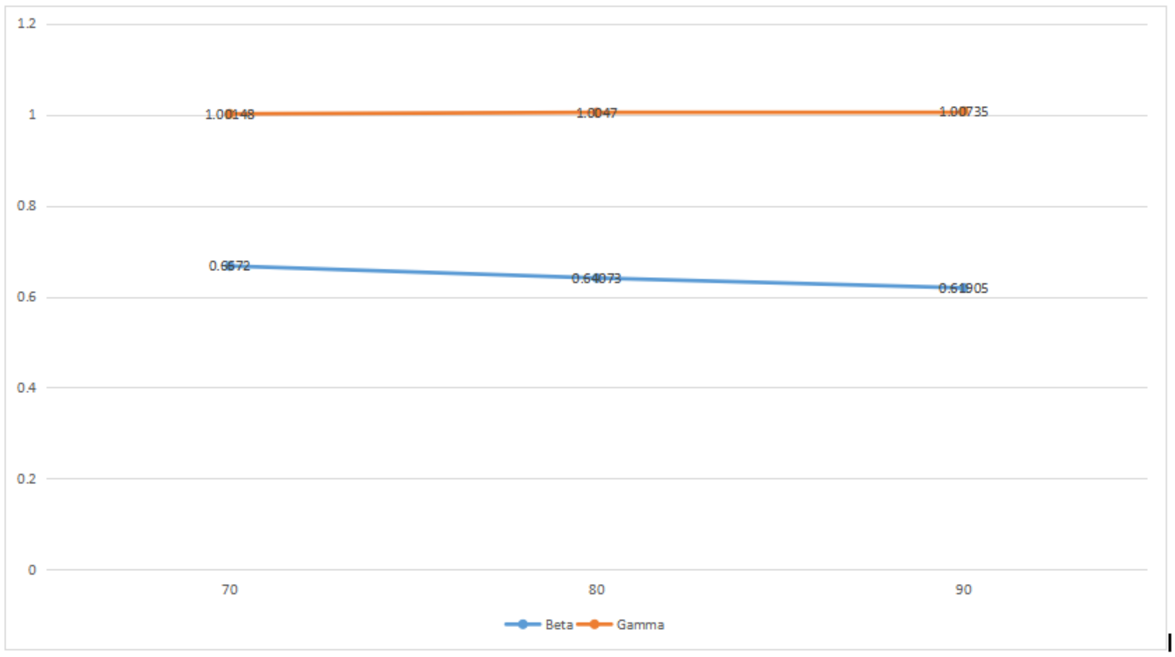

Figure 4.32: **Posterior Mean weights of beta and gamma at N=1000 using SSLHT Activation function**

Table 4.32 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using SSLHT activation function at the sample size of 1000. The table shows that β has a posterior weight of 0.6672, 0.6407 and 0.6190 and γ shows 1.0014, 1.0047 and 1.0073 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.2927, 0.2723 and 0.2569 and γ also has a posterior standard deviation of 0.3106, 0.2890 and 0.2722 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0092, 0.0086 and 0.0081 and γ also shows NSE values of 0.0098, 0.0091 and 0.0086 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.32 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 1000.

Table 4.33: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error for SSLHT activation function at N=5000.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.76896	0.15635	0.00494
	γ	1.02413	0.15787	0.00499
	μ_β	0.82809	0.40311	0.01275
	μ_γ	0.42856	0.69764	0.02206
	σ	10392.37071	205.21219	6.48938
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	3.75E-08	3.85E-08	1.22E-09
80	β	0.76967	0.14526	0.00459
	γ	1.02421	0.1466	0.00464
	μ_β	0.82816	0.40312	0.01275
	μ_γ	0.42856	0.69764	0.02206
	σ	13583.91039	268.23369	8.48229
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	2.87E-08	2.95E-08	9.32E-10
90	β	0.77524	0.13777	0.00436
	γ	1.02423	0.13904	0.0044
	μ_β	0.82822	0.40314	0.01275
	μ_γ	0.42856	0.69764	0.02206
	σ	17158.84149	338.82581	10.71461
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	2.27E-08	2.33E-08	7.38E-10

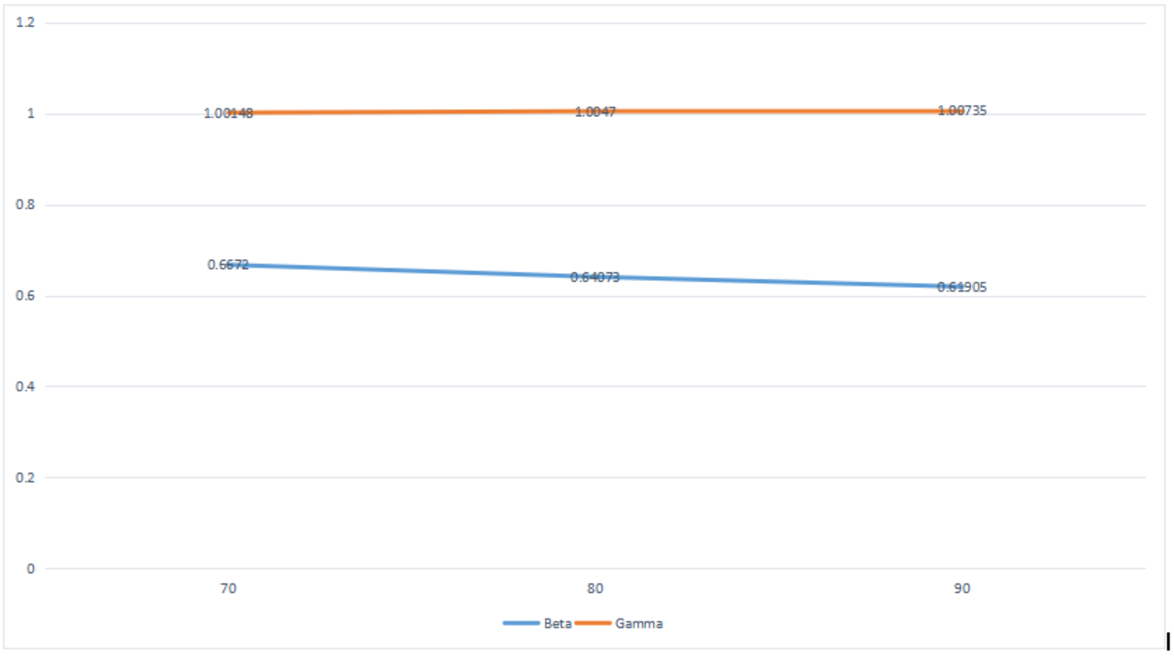

Figure 4.33: **Posterior Mean weights of beta and gamma at N=5000 using SSLHT Activation function**

Table 4.33 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using SSLHT activation function at the sample size of 5000. The table shows that β has a posterior weight of 0.7689, 0.7696 and 0.7752 and γ shows 1.0241, 1.0242 and 1.0242at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.1563, 0.1452 and 0.1377 and γ also has a posterior standard deviation of 0.1578, 0.1466 and 0.1390 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0049, 0.0045 and 0.0043 and γ also shows NSE values of 0.0049, 0.0046 and 0.0044 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.33 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 5000.

Table 4.34: **Summary Table of the Posterior Weight, standard deviation and NSE for SSLHT activation function at N=10000.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.79964	0.11509	0.00364
	γ	1.02421	0.11629	0.00368
	μ_β	0.8284	0.4032	0.01275
	μ_γ	0.42856	0.69764	0.02206
	σ	20897.41076	291.8228	9.22825
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	9.33E-09	9.59E-09	3.03E-10
80	β	0.80129	0.1072	0.00339
	γ	1.02424	0.10825	0.00342
	μ_β	0.82843	0.40322	0.01275
	μ_γ	0.42856	0.69764	0.02206
	σ	27340.46009	381.79704	12.07348
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	7.13E-09	7.33E-09	2.32E-10
90	β	0.80381	0.10133	0.0032
	γ	1.02425	0.10227	0.00323
	μ_β	0.82847	0.40323	0.01275
	μ_γ	0.42856	0.69764	0.02206
	σ	34656.733	483.96545	15.30433
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	5.63E-09	5.78E-09	1.83E-10

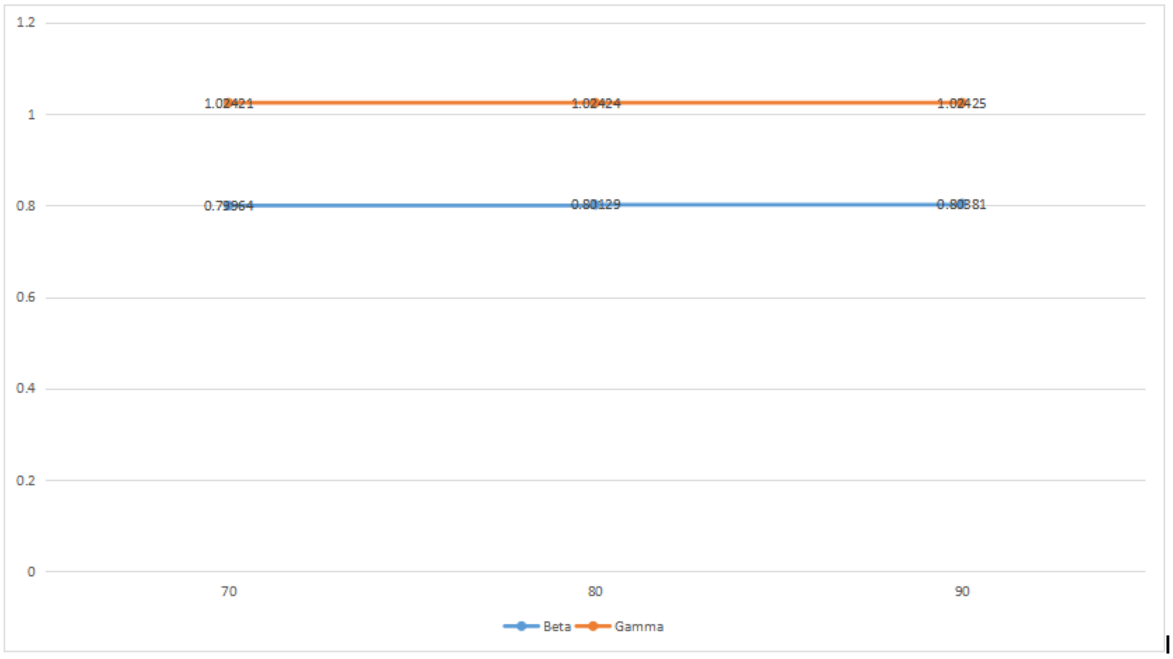

Figure 4.34: **Posterior Mean weights of beta and gamma at N=10000 using SSLHT Activation function**

Table 4.34 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using SSLHT activation function at the sample size of 10000. The table shows that β has a posterior weight of 0.7996, 0.8012 and 0.8038 and γ shows 1.02421, 1.02424 and 1.02425 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.1150, 0.1072 and 0.1013 and γ also has a posterior standard deviation of 0.1162, 0.1082 and 0.1022 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0036, 0.0033 and 0.0032 and γ also shows NSE values of 0.0036, 0.0034 and 0.0032 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.34 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 10000.

Table 4.35: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error for SSLHT activation function at N=20000.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.77629	0.09498	0.003
	γ	1.02463	0.09497	0.003
	μ_β	0.82855	0.40327	0.01275
	μ_γ	0.42856	0.69764	0.02206
	σ	44176.73736	436.2365	13.79501
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	2.21E-09	2.27E-09	7.17E-11
80	β	0.78602	0.08874	0.00281
	γ	1.02456	0.08873	0.00281
	μ_β	0.82857	0.40328	0.01275
	μ_γ	0.42856	0.69764	0.02206
	σ	57714.92439	569.9234	18.02256
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	1.69E-09	1.74E-09	5.49E-11
90	β	0.79482	0.08347	0.00264
	γ	1.0245	0.08347	0.00264
	μ_β	0.82859	0.40329	0.01275
	μ_γ	0.42856	0.69764	0.02206
	σ	73033.20745	721.1884	22.80598
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	1.34E-09	1.37E-09	4.34E-11

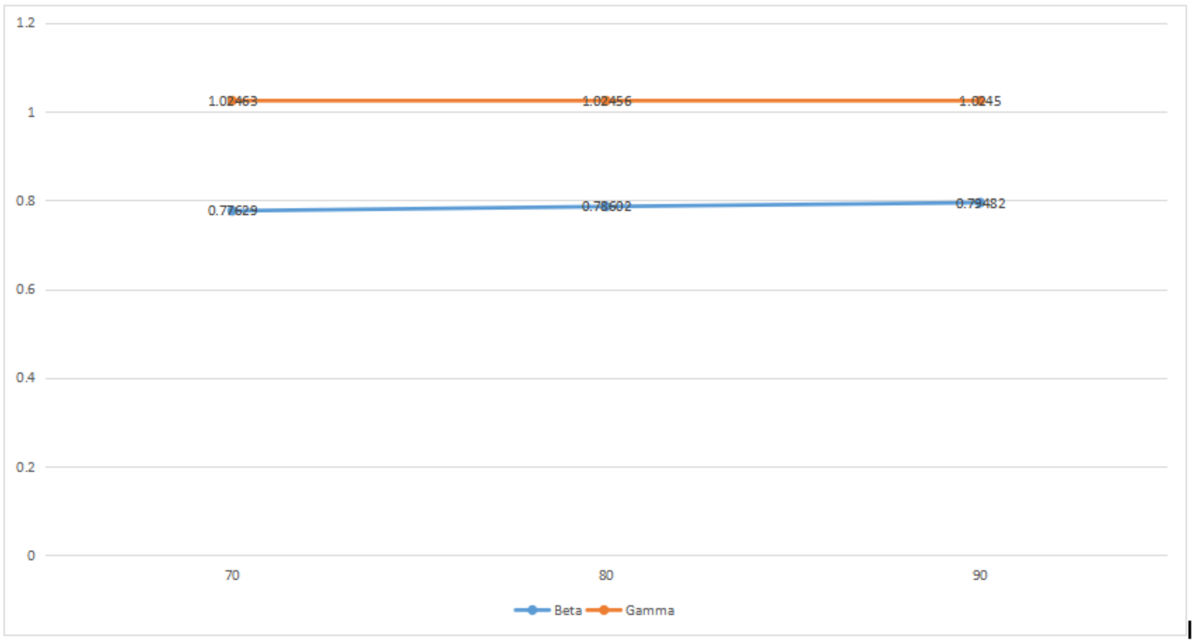

Figure 4.35: **Posterior Mean weights of beta and gamma at N=20000 using SSLHT Activation function**

Table 4.35 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using SSLHT activation function at the sample size of 20000. The table shows that β has a posterior weight of 0.7762, 0.7860 and 0.7948 and γ shows 1.0246, 1.02456 and 1.0245 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.0949, 0.0887 and 0.0834 and γ also has a posterior standard deviation of 0.0949, 0.0887 and 0.0834 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.003, 0.0028 and 0.0026 and γ also shows NSE values of 0.003, 0.0028 and 0.0026 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.35 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 20000.

Table 4.36: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error for SSLHT activation function at N=50000.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.83436	0.05667	0.00179
	γ	1.02419	0.0566	0.00179
	μ_β	0.82866	0.40334	0.01275
	μ_γ	0.42856	0.69764	0.02206
	σ	108991.9585	680.69567	21.52549
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	3.58E-10	3.68E-10	1.16E-11
80	β	0.84051	0.05285	0.00167
	γ	1.02414	0.0528	0.00167
	μ_β	0.82867	0.40335	0.01276
	μ_γ	0.42856	0.69764	0.02206
	σ	142332.7604	888.92148	28.11017
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	2.74E-10	2.82E-10	8.91E-12
90	β	0.84532	0.0498	0.00157
	γ	1.02411	0.04975	0.00157
	μ_β	0.82868	0.40336	0.01276
	μ_γ	0.42856	0.69764	0.02206
	σ	180208.1124	1125.46727	35.5904
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	2.16E-10	2.22E-10	7.04E-12

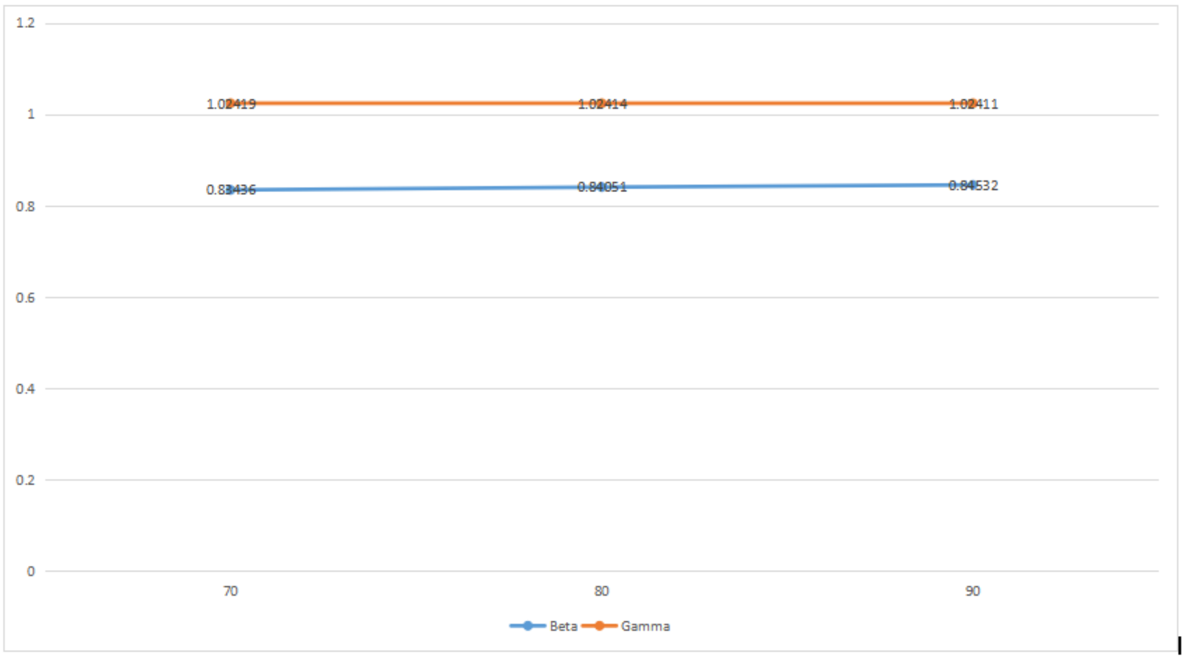

Figure 4.36: **Posterior Mean weights of beta and gamma at N=50000 using SSLHT Activation function**

Table 4.36 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using SSLHT activation function at the sample size of 50000. The table shows that β has a posterior weight of 0.8343, 0.8405 and 0.8453 and γ shows 1.02419, 1.02414 and 1.02411 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.0566, 0.0528 and 0.0498 and γ also has a posterior standard deviation of 0.0566, 0.0528 and 0.0497 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0017, 0.0016 and 0.0015 and γ also shows NSE values of 0.0017, 0.0016 and 0.0015 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.36 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 50000.

4.6 BAYESIAN RESULTS FOR SSLHTS ACTIVATION FUNCTION

The results for the parameters of the model using Bayesian approach with Symmetric Saturating Linear Hyperbolic Tangent Sigmoid(SSLHTS) activation function are displayed in this sesction. These results are tabled under the headings of Posterior Mean Weight, Posterior Standard Deviation and the Numerical Standard Error (NSE). The results are obtained for the SSLHTS activation functions at 70%, 80% and 90% training sets and at sample size of 50, 100, 200, 500, 1000, 5000, 10000, 20000 and 50000.

Table 4.37: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error for SSLHTS activation function at N=50.**

Training set	Parameters	results_pweight	results_pstd	results_nse
	β	0.7253	0.20619	0.00652
	γ	0.94861	0.25933	0.0082
	μ_β	0.78353	0.40852	0.01292
70	μ_γ	0.4158	0.6875	0.02174
	σ	77.33606	14.56755	0.46067
	σ_β	1.51024	0.76269	0.02412
	S_γ	0.00492	0.00243	7.67E-05
	π	0.00042	0.00043	1.37E-05
	β	0.6428	0.19528	0.00618
	γ	0.96585	0.23829	0.00754
	μ_β	0.79068	0.40699	0.01287
80	μ_γ	0.4158	0.6875	0.02174
	σ	108.94284	20.52122	0.64894
	σ_β	1.51024	0.76269	0.02412
	S_γ	0.00492	0.00243	7.67E-05
	π	0.0003	0.00031	9.73E-06
	β	0.61262	0.19123	0.00605
	γ	0.97413	0.22919	0.00725
	μ_β	0.79486	0.40617	0.01284
90	μ_γ	0.4158	0.6875	0.02174
	σ	143.09586	26.95451	0.85238
	σ_β	1.51024	0.76269	0.02412
	S_γ	0.00492	0.00243	7.67E-05
	π	0.00023	0.00023	7.40E-06

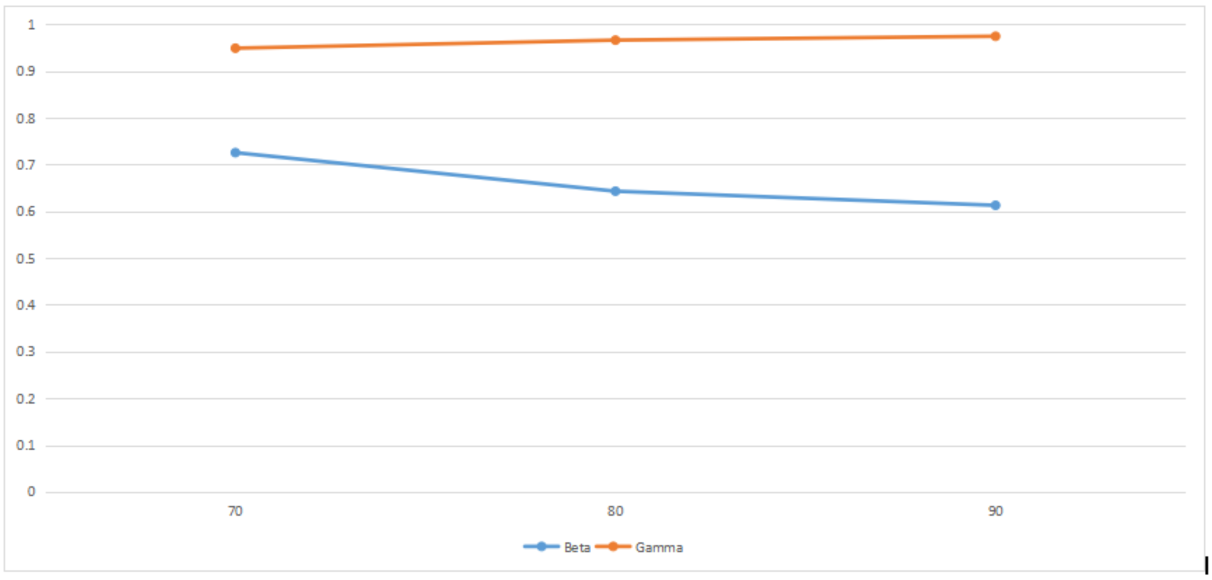

Figure 4.37: **Posterior Mean weights of beta and gamma at N=50 using SSLHTS Activation function**

Table 4.37 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using SSLHTS activation function at the sample size of 50. The table shows that β has a posterior weight of 0.7253, 0.6428 and 0.6126 and γ shows 0.9486, 0.9658 and 0.9741 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.2061, 0.1952 and 0.1912 and γ also has a posterior standard deviation of 0.2593, 0.2382 and 0.2291 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0065, 0.0061 and 0.0060 and γ also shows NSE values of 0.0082, 0.0075 and 0.0072 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.37 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 50.

Table 4.38: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error for SSLHTS activation function at N=100.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.63023	0.19199	0.00607
	γ	0.99029	0.21792	0.00689
	μ_β	0.80444	0.40632	0.01285
	μ_γ	0.41674	0.68721	0.02173
	σ	206.92079	28.10228	0.88867
	σ_β	1.51679	0.7589	0.024
	S_γ	0.00489	0.00239	7.57E-05
	π	8.57E-05	8.86E-05	2.80E-06
80	β	0.68868	0.18838	0.00596
	γ	0.99022	0.21614	0.00683
	μ_β	0.80643	0.40615	0.01284
	μ_γ	0.41674	0.68721	0.02173
	σ	267.993	36.39661	1.15096
	σ_β	1.51679	0.7589	0.024
	S_γ	0.00489	0.00239	7.57E-05
	π	6.62E-05	6.84E-05	2.16E-06
90	β	0.69539	0.16206	0.00512
	γ	0.99677	0.18759	0.00593
	μ_β	0.80956	0.4057	0.01283
	μ_γ	0.41674	0.68721	0.02173
	σ	364.38366	49.48759	1.56494
	σ_β	1.51679	0.7589	0.024
	S_γ	0.00489	0.00239	7.57E-05
	π	4.87E-05	5.03E-05	1.59E-06

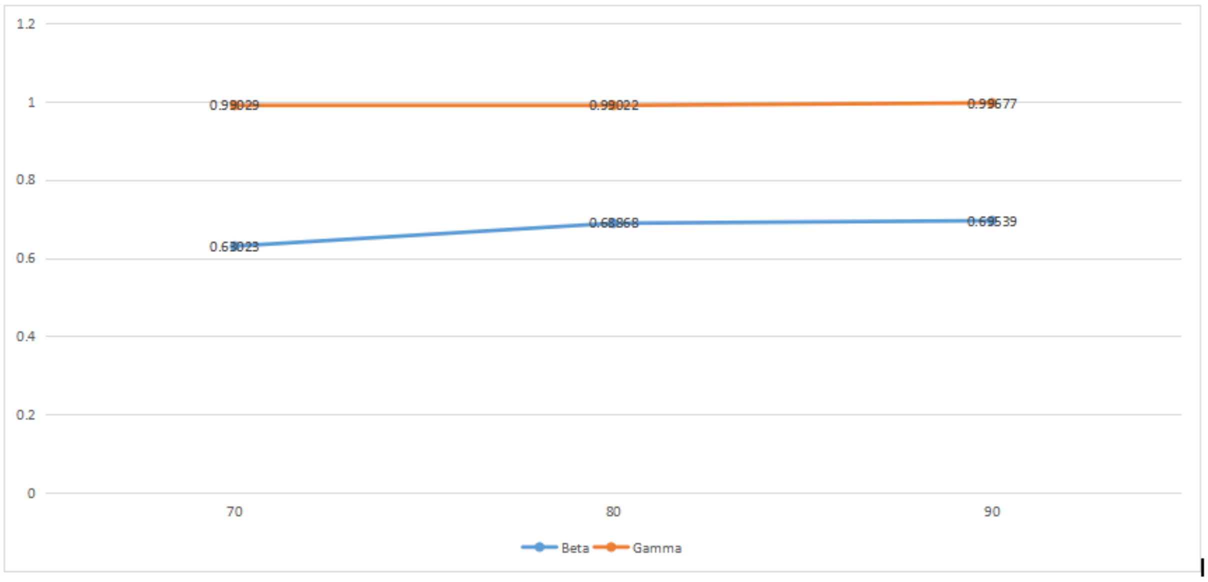

Figure 4.38: **Posterior Mean weights of beta and gamma at N=100 using SSLHTS Activation function**

Table 4.38 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using SSLHTS activation function at the sample size of 100. The table shows that β has a posterior weight of 0.6302, 0.6886 and 0.6953 and γ shows 0.99029, 0.99022 and 0.99677 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.1919, 0.1883 and γ also has a posterior standard deviation of 0.2179, 0.2161 and 0.1875 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0060, 0.0059 and 0.0051 and γ also shows NSE values of 0.00689, 0.00683 and 0.00593 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.38 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 100.

Table 4.39: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error for SSLHTS activation function at N=200.**

Training set	Parameters	results_pweight	results_pstd	results_nse
	β	0.72245	0.19678	0.00622
	γ	1.03354	0.17704	0.0056
	μ_β	0.81076	0.40417	0.01278
70	μ_γ	0.41915	0.68682	0.02172
	σ	507.27183	49.1658	1.55476
	σ_β	1.51529	0.75878	0.02399
	S_γ	0.00492	0.00242	7.65E-05
	π	1.81E-05	1.87E-05	5.90E-07
	β	0.73109	0.1837	0.00581
	γ	1.03155	0.16681	0.00528
	μ_β	0.81283	0.40399	0.01278
80	μ_γ	0.41915	0.68682	0.02172
	σ	674.08718	65.33388	2.06604
	σ_β	1.51529	0.75878	0.02399
	S_γ	0.00492	0.00242	7.65E-05
	π	1.36E-05	1.41E-05	4.44E-07
	β	0.89114	0.17192	0.00544
	γ	1.03123	0.15494	0.0049
	μ_β	0.81384	0.40391	0.01277
90	μ_γ	0.41915	0.68682	0.02172
	σ	864.33188	83.77278	2.64913
	σ_β	1.51529	0.75878	0.02399
	S_γ	0.00492	0.00242	7.65E-05
	π	1.06E-05	1.10E-05	3.47E-07

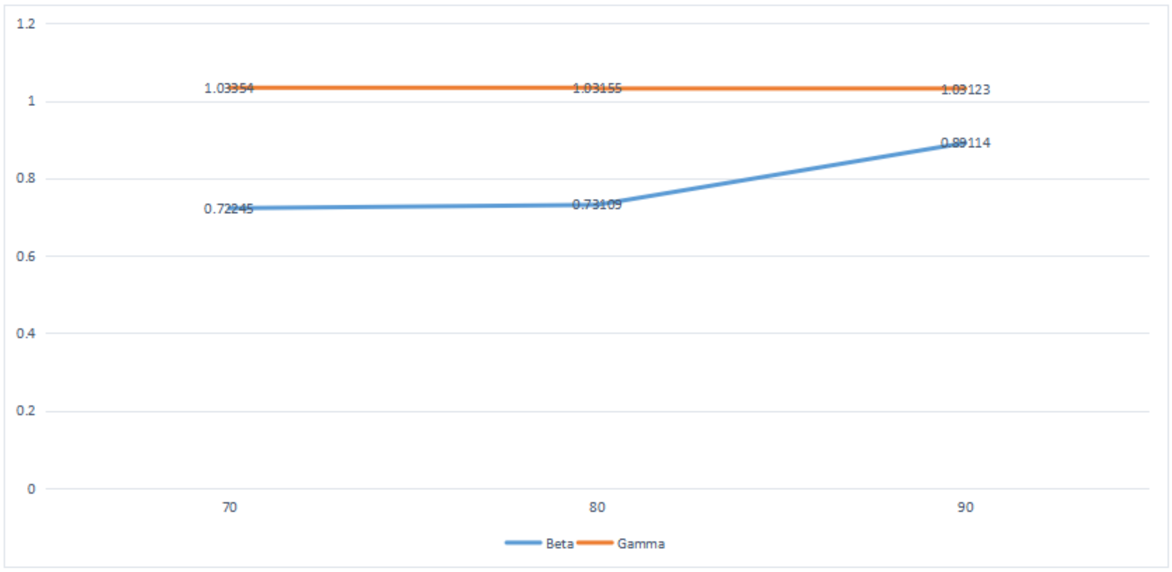

Figure 4.39: **Posterior Mean weights of beta and gamma at N=200 using SSLHTS Activation function**

Table 4.39 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using SSLHTS activation function at the sample size of 200. The table shows that β has a posterior weight of 0.7224, 0.7310 and 0.8911 and γ shows 1.0335, 1.0315 and 1.0312 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.1967, 0.1837 and 0.1719 and γ also has a posterior standard deviation of 0.1770, 0.1668 and 0.1549 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0062, 0.0058 and 0.0054 and γ also shows NSE values of 0.0056, 0.0052 and 0.0049 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.39 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 200.

Table 4.40: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error for SSLHTS activation function at N=500.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.78351	0.11283	0.00357
	γ	1.03045	0.09137	0.00289
	μ_β	0.82592	0.40163	0.0127
	μ_γ	0.42833	0.6905	0.02184
	σ	1925.05482	119.04932	3.76467
	σ_β	1.52328	0.75835	0.02398
	S_γ	0.00485	0.00238	7.52E-05
	π	1.98E-06	2.05E-06	6.47E-08
80	β	0.71154	0.10601	0.00335
	γ	1.02981	0.08545	0.0027
	μ_β	0.82647	0.4016	0.0127
	μ_γ	0.42833	0.6905	0.02184
	σ	2525.89515	156.20651	4.93968
	σ_β	1.52328	0.75835	0.02398
	S_γ	0.00485	0.00238	7.52E-05
	π	1.51E-06	1.56E-06	4.93E-08
90	β	0.82167	0.10024	0.00317
	γ	1.02923	0.08053	0.00255
	μ_β	0.82688	0.40158	0.0127
	μ_γ	0.42833	0.6905	0.02184
	σ	3168.82357	195.96652	6.19701
	σ_β	1.52328	0.75835	0.02398
	S_γ	0.00485	0.00238	7.52E-05
	π	1.20E-06	1.24E-06	3.93E-08

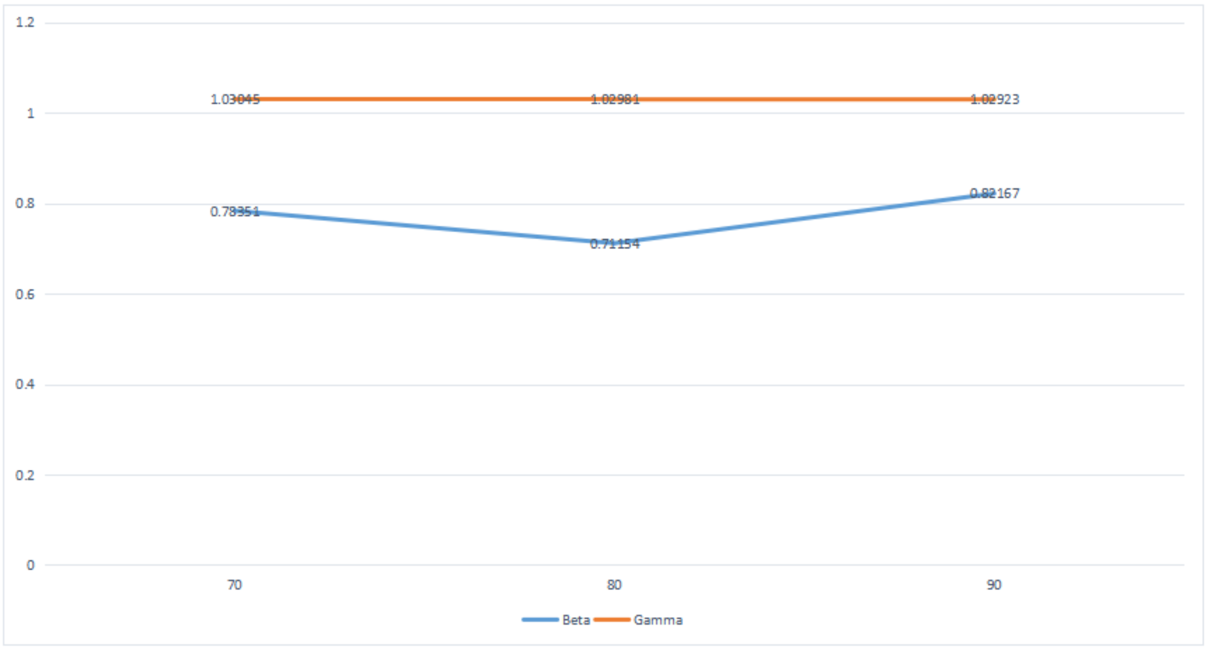

Figure 4.40: **Posterior Mean weights of beta and gamma at N=500 using SSLHTS Activation function**

Table 4.40 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using SSLHTS activation function at the sample size of 500. The table shows that β has a posterior weight of 0.7835, 0.7115 and 0.8216 and γ shows 1.0304, 1.0298 and 1.0292 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.1128, 0.1060 and 0.1002 and γ also has a posterior standard deviation of 0.0913, 0.0854 and 0.0805 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0035, 0.0033 and 0.0031 and γ also shows NSE values of 0.0028, 0.0027 and 0.0025 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.40 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 500.

Table 4.41: **Summary Table of the Posterior Weight, standard deviation and NSE for SSLHTS activation function at N=1000.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.82071	0.07901	0.0025
	γ	1.02735	0.06353	0.00201
	μ_β	0.82644	0.40108	0.01268
	μ_γ	0.42807	0.69204	0.02188
	σ	4038.70461	177.1014	5.60044
	σ_β	1.52485	0.75682	0.02393
	S_γ	0.00483	0.00234	7.40E-05
	π	4.75E-07	4.91E-07	1.55E-08
80	β	0.82952	0.07344	0.00232
	γ	1.02686	0.05901	0.00187
	μ_β	0.82675	0.40109	0.01268
	μ_γ	0.42807	0.69204	0.02188
	σ	5284.04736	231.71097	7.32734
	σ_β	1.52485	0.75682	0.02393
	S_γ	0.00483	0.00234	7.40E-05
	π	3.63E-07	3.75E-07	1.19E-08
90	β	0.83532	0.06986	0.00221
	γ	1.02657	0.05621	0.00178
	μ_β	0.82698	0.4011	0.01268
	μ_γ	0.42807	0.69204	0.02188
	σ	6633.33346	290.87857	9.19839
	σ_β	1.52485	0.75682	0.02393
	S_γ	0.00483	0.00234	7.40E-05
	π	2.89E-07	2.99E-07	9.45E-09

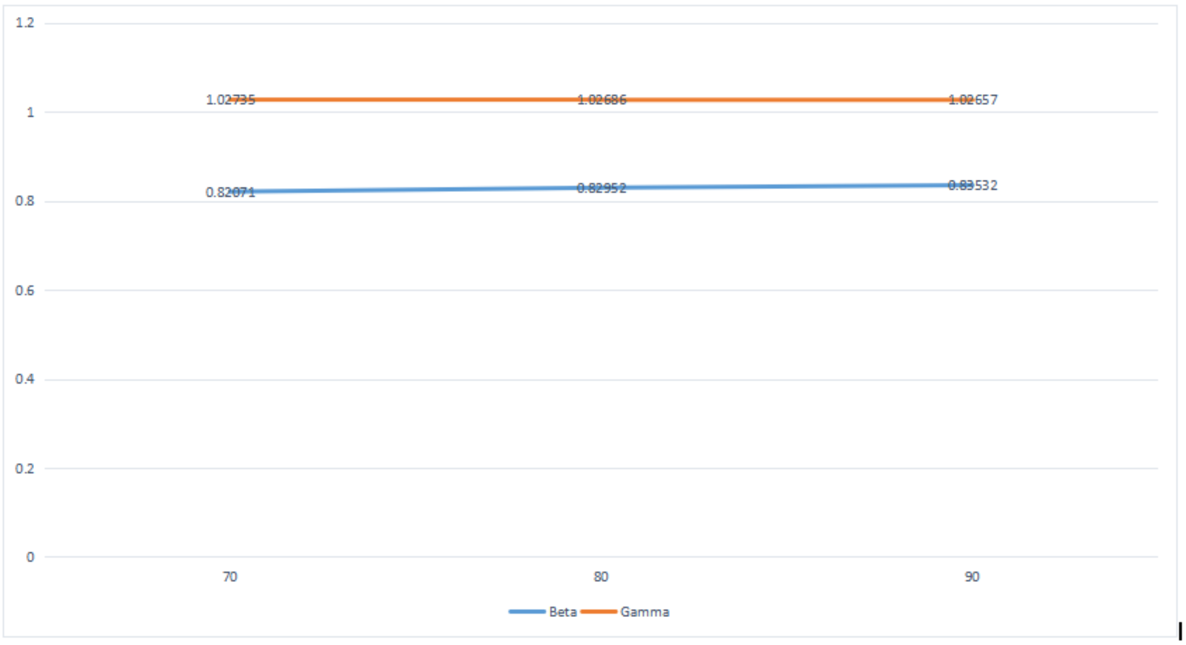

Figure 4.41: **Posterior Mean weights of beta and gamma at N=1000 using SSLHTS Activation function**

Table 4.41 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using SSLHTS activation function at the sample size of 1000. The table shows that β has a posterior weight of 0.8207, 0.8295 and 0.8353 and γ shows 1.0304, 1.0298 and 1.0292 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.0790, 0.0734 and 0.0698 and γ also has a posterior standard deviation of 0.0635, 0.0590 and 0.0562 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0025, 0.0023 and 0.0022 and γ also shows NSE values of 0.0020, 0.0018 and 0.0017 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.41 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 1000.

Table 4.42: **Posterior mean, standard deviation and NSE for SSLHTS activation function at N=5000.**

Training set	Parameters	results_pmean	results_pstd	results_nse
70	β	0.81289	0.03608	0.00114
	γ	1.02392	0.03411	0.00108
	μ_β	0.82827	0.40314	0.01275
	μ_γ	0.42856	0.69764	0.02206
	σ	17296.52571	341.54458	10.80059
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	2.25E-08	2.32E-08	7.32E-10
80	β	0.82366	0.03368	0.00107
	γ	1.02387	0.03189	0.00101
	μ_β	0.82832	0.40316	0.01275
	μ_γ	0.42856	0.69764	0.02206
	σ	22534.58014	444.97744	14.07142
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	1.73E-08	1.78E-08	5.62E-10
90	β	0.83139	0.03184	0.00101
	γ	1.02385	0.03015	0.00095
	μ_β	0.82836	0.40318	0.01275
	μ_γ	0.42856	0.69764	0.02206
	σ	28621.71281	565.17656	17.87245
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	1.36E-08	1.40E-08	4.43E-10

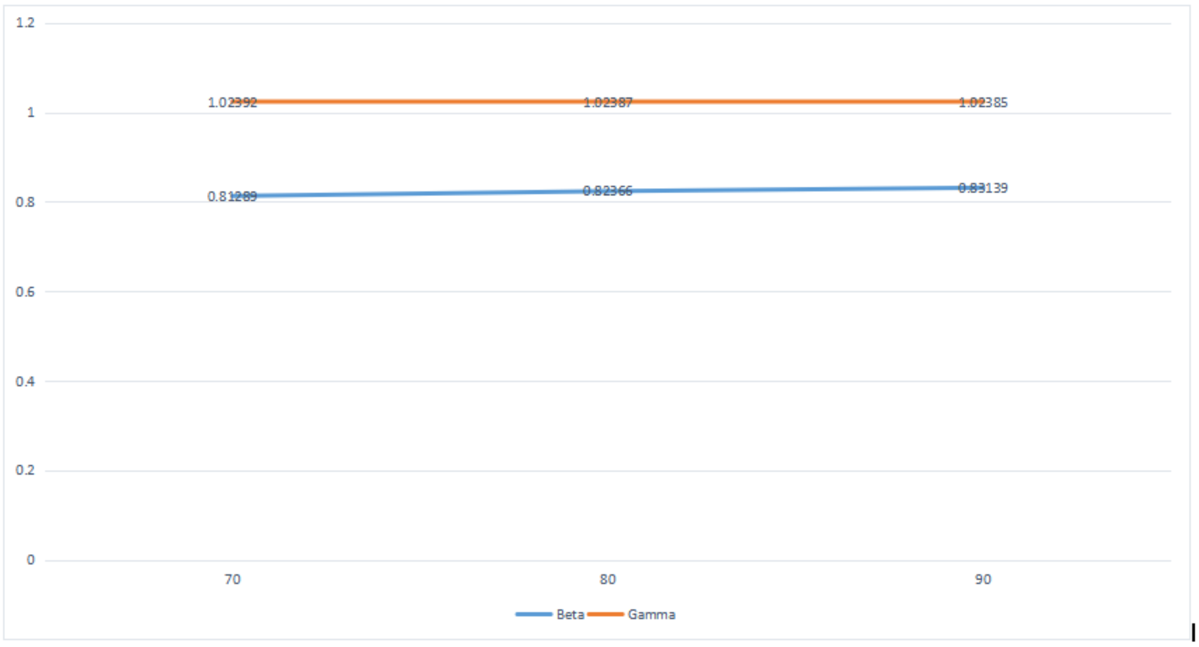

Figure 4.42: **Posterior Mean weights of beta and gamma at N=5000 using SSLHTS Activation function**

Table 4.42 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using SSLHTS activation function at the sample size of 5000. The table shows that β has a posterior weight of 0.8128, 0.8236 and 0.8313 and γ shows 1.0239, 1.02387 and 1.02385 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.0360, 0.0336 and 0.0318 and γ also has a posterior standard deviation of 0.0341, 0.0318 and 0.0301 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0011, 0.0010 and 0.0010 and γ also shows NSE values of 0.00108, 0.00101 and 0.00095 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.42 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 5000.

Table 4.43: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error for SSLHTS activation function at N=10000.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.8522	0.02742	0.00087
	γ	1.02378	0.02619	0.00083
	μ_β	0.82847	0.40323	0.01275
	μ_γ	0.42856	0.69764	0.02206
	σ	32254.4304	450.41839	14.24348
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	6.05E-09	6.21E-09	1.96E-10
80	β	0.85901	0.02555	0.00081
	γ	1.02375	0.02446	0.00077
	μ_β	0.8285	0.40324	0.01275
	μ_γ	0.42856	0.69764	0.02206
	σ	42058.62214	587.32945	18.57299
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	4.64E-09	4.76E-09	1.51E-10
90	β	0.86418	0.02404	0.00076
	γ	1.02373	0.02305	0.00073
	μ_β	0.82853	0.40325	0.01275
	μ_γ	0.42856	0.69764	0.02206
	σ	53113.5836	741.70694	23.45483
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	3.67E-09	3.77E-09	1.19E-10

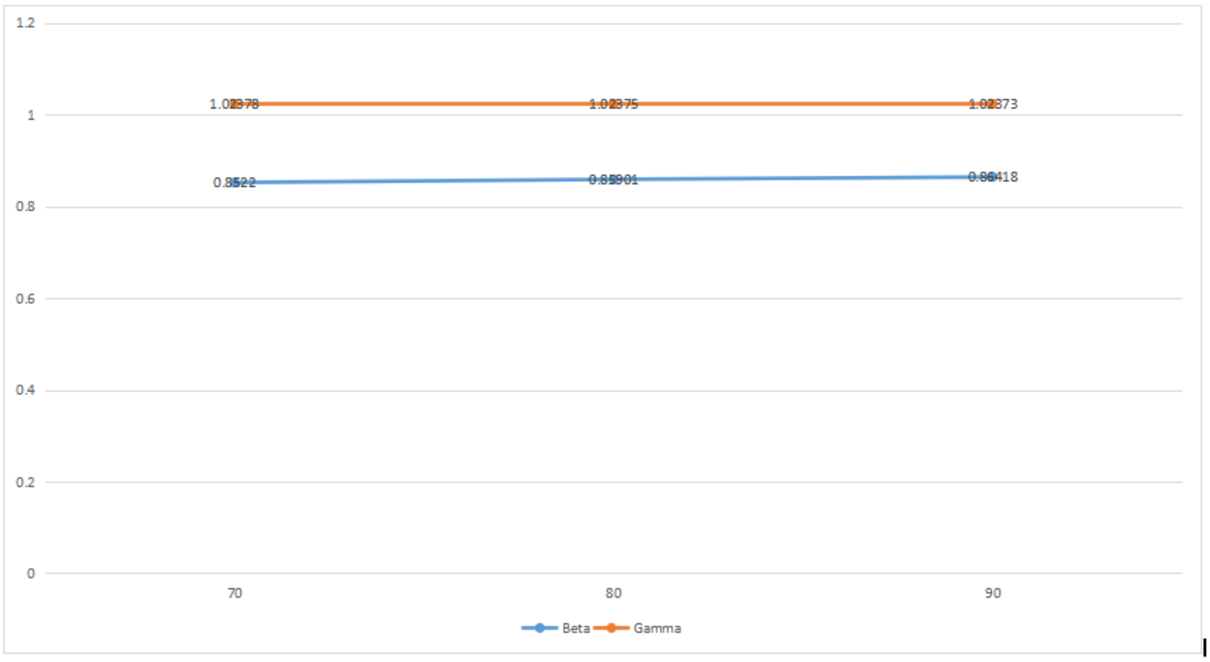

Figure 4.43: **Posterior Mean weights of beta and gamma at N=10000 using SSLHTS Activation function**

Table 4.43 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using SSLHTS activation function at the sample size of 10000. The table shows that β has a posterior weight of 0.8522, 0.8590 and 0.8641 and γ shows 1.02378, 1.02375 and 1.02373 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.0274, 0.0255 and 0.0240 and γ also has a posterior standard deviation of 0.0261, 0.0244 and 0.0230 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.00087, 0.00081 and 0.00076 and γ also shows NSE values of 0.00083, 0.00077 and 0.00073 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.43 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 10000.

Table 4.44: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error for SSLHTS activation function at N=20000.**

Training set	Parameters	results_pweight	results_pstd	results_nse
70	β	0.87819	0.02126	0.00067
	γ	1.02369	0.02081	0.00066
	μ_β	0.8286	0.4033	0.01275
	μ_γ	0.42856	0.69764	0.02206
	σ	6697.9276	668.5036	21.13994
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	1.44E-09	1.48E-09	4.68E-11
80	β	0.88159	0.01987	0.00063
	γ	1.02367	0.01945	0.00062
	μ_β	0.82862	0.40331	0.01275
	μ_γ	0.42856	0.69764	0.02206
	σ	8448.8386	873.4147	27.6198
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	1.10E-09	1.13E-09	3.58E-11
90	β	0.88426	0.01869	0.00059
	γ	1.02366	0.01829	0.00058
	μ_β	0.82863	0.40332	0.01275
	μ_γ	0.42856	0.69764	0.02206
	σ	11979.0864	1105.77122	34.96756
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	8.71E-10	8.95E-10	2.83E-11

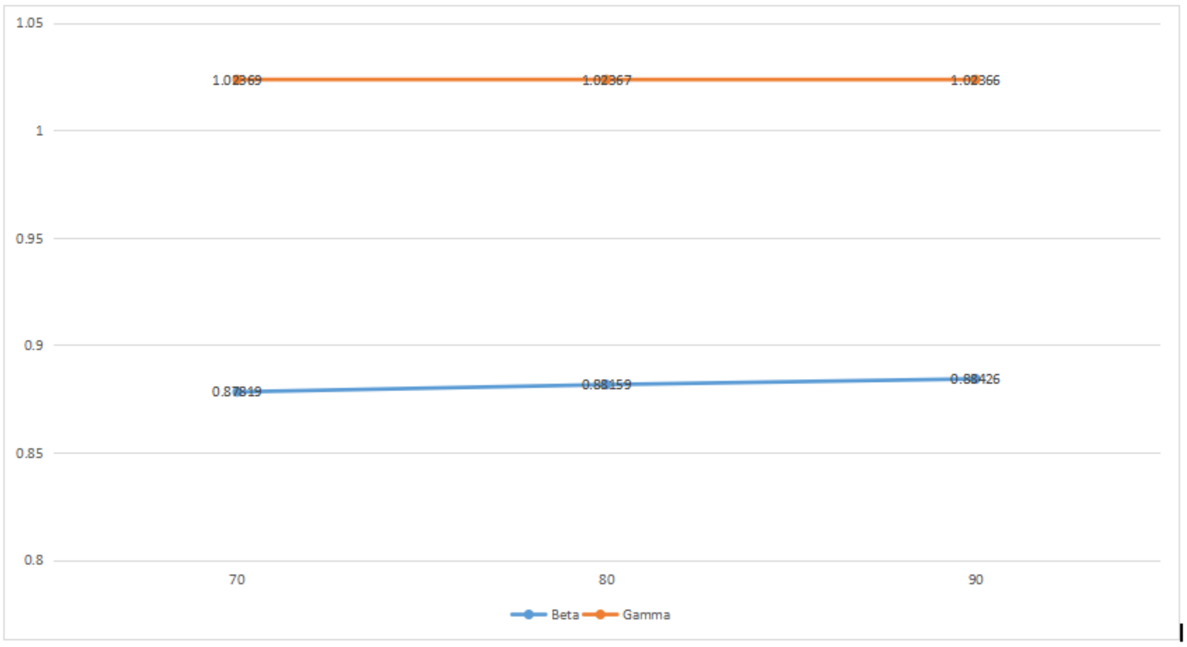

Figure 4.44: **Posterior Mean weights of beta and gamma at N=20000 using SSLHTS Activation function**

Table 4.44 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using SSLHTS activation function at the sample size of 20000. The table shows that β has a posterior weight of 0.8781, 0.8815 and 0.8842 and γ shows 1.02369, 1.02367 and 1.02366 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.0212, 0.0198 and 0.0186 and γ has a posterior standard deviation of 0.0281, 0.0194 and 0.0182 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.00067, 0.00063 and 0.00596 and γ also shows NSE values of 0.00066, 0.00062 and 0.00058 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.44 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 20000.

Table 4.45: **Summary Table of the Posterior Weight, Standard Deviation and Numerical Standard Error for SSLHTS activation function at N=50000**

Training set	Parameters	results_pweight	results_pstd	results_nse
	β	0.89825	0.01278	0.0004
	γ	1.02359	0.01269	0.0004
	μ_β	0.82869	0.40337	0.01276
70	μ_γ	0.42856	0.69764	0.02206
	σ	16920.4436	1011.25391	31.97866
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	2.41E-10	2.48E-10	7.83E-12
	β	0.89994	0.01193	0.00038
	γ	1.02358	0.01184	0.00037
	μ_β	0.82869	0.40338	0.01276
80	μ_γ	0.42856	0.69764	0.02206
	σ	21609.8463	1321.58287	41.79212
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	1.84E-10	1.89E-10	5.99E-12
	β	0.90129	0.01126	0.00036
	γ	1.02358	0.01117	0.00035
	μ_β	0.8287	0.40338	0.01276
90	μ_γ	0.42856	0.69764	0.02206
	σ	26206.8167	1668.80681	52.7723
	σ_β	1.52303	0.7498	0.02371
	S_γ	0.00481	0.0023	7.27E-05
	π	1.46E-10	1.50E-10	4.74E-12

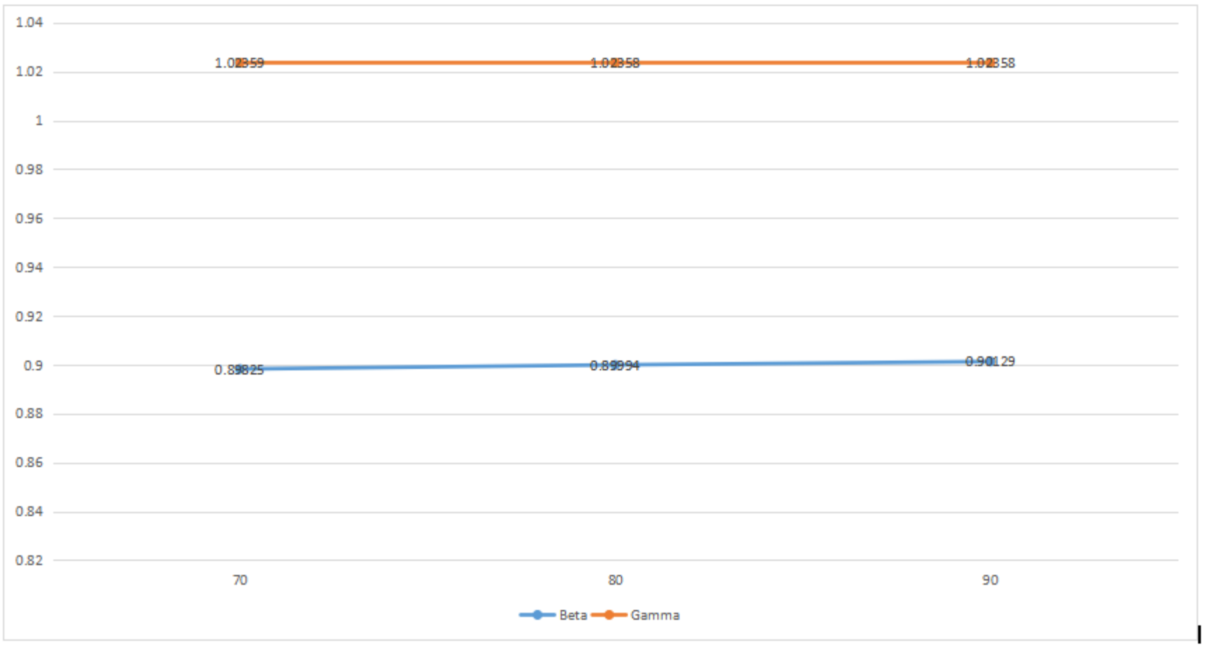

Figure 4.45: **Posterior Mean weights of beta and gamma at N=50000 using SSLHTS Activation function**

Table 4.45 shows the results for the posterior weight, standard deviation and numerical standard error of the bayesian neural network model using SSLHTS activation function at the sample size of 20000. The table shows that β has a posterior weight of 0.8982, 0.8999 and 0.9012 and γ shows 1.02359, 1.02358 and 1.02358 at training sets of 70%, 80% and 90% respectively. The result shows a decreasing posterior weights for β and γ as the training set increases.

The table also shows that β has a posterior standard deviation of 0.0127, 0.0119 and 0.0112 and γ also has a posterior standard deviation of 0.0126, 0.0118 and 0.0117 at training sets of 70%, 80% and 90% respectively. This implies that both β and γ produced decreasing posterior standard deviation as the training sets increases.

The result of the Numerical Standard Error(NSE) for both the β and γ are also displayed in the table. The table shows that β has nse results as 0.0004, 0.00038 and 0.00036 and γ also shows NSE values of 0.0004, 0.00037 and 0.00035 at training sets of 70%, 80% and 90% respectively. This implies a decreasing NSE values for both β and γ as the training sets increases.

Figure 4.45 shows the line plot for the posterior weights of β and γ at the varying values of training sets at the sample size of 50000.

4.7 ASYMPTOTIC PERFORMANCE FOR ReLU ACTIVATION FUNCTION AT VARIOUS SAMPLE SIZES AND ACTIVATION FUNCTIONS

The asymptotic performance of the ReLU activation function is displayed in this section. The results at training set of 70, 80 and 90 percents using the sample sizes of 50, 100, 200, 500, 1000, 5000, 10000, 20,000 and 50,000 of the Mean Square Error (MSE), Mean Absolute Error(MAE) and the Training set error of the SSLHT function are displayed.

Table 4.46: **Asymptotic Performance for ReLU activation function at different training set**

Training set	Sample size	MSE	MAE	Train.err
70	50	0.1631	0.2465	0.1522
	100	0.2412	0.2534	0.2498
	200	0.2508	0.3104	0.26
	500	0.283	0.3297	0.2818
	1000	0.3163	0.4807	0.3153
	5000	0.4696	0.6896	0.466
	10000	0.5044	0.6634	0.5043
	20000	0.5065	0.7452	0.5064
	50000	0.5726	0.7188	0.5725
80	50	0.1195	0.3091	0.1076
	100	0.363	0.5439	0.3449
	200	0.4463	0.55	0.4515
	500	0.5062	0.5541	0.5111
	1000	0.5323	0.5765	0.5306
	5000	0.553	0.5955	0.5524
	10000	0.5616	0.6603	0.5713
	20000	0.5795	0.7423	0.5793
	50000	0.5898	0.716	0.5892
90	50	0.1468	0.3183	0.1174
	100	0.2258	0.5499	0.3032
	200	0.2578	0.5519	0.3496
	500	0.262	0.5055	0.3568
	1000	0.3664	0.5473	0.3627
	5000	0.5566	0.6006	0.5555
	10000	0.5649	0.6543	0.5644
	20000	0.5657	0.6701	0.5754
	50000	0.5711	0.7173	0.581

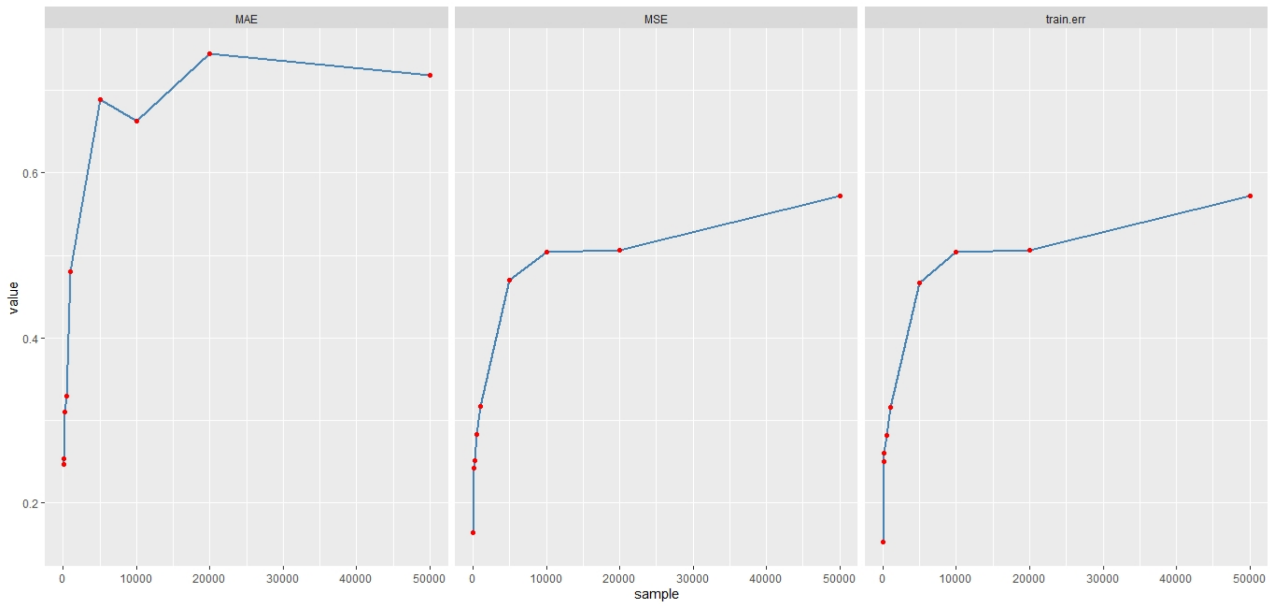

Figure 4.46: **Asymptotic Performance for ReLU activation function at Training set of 70**

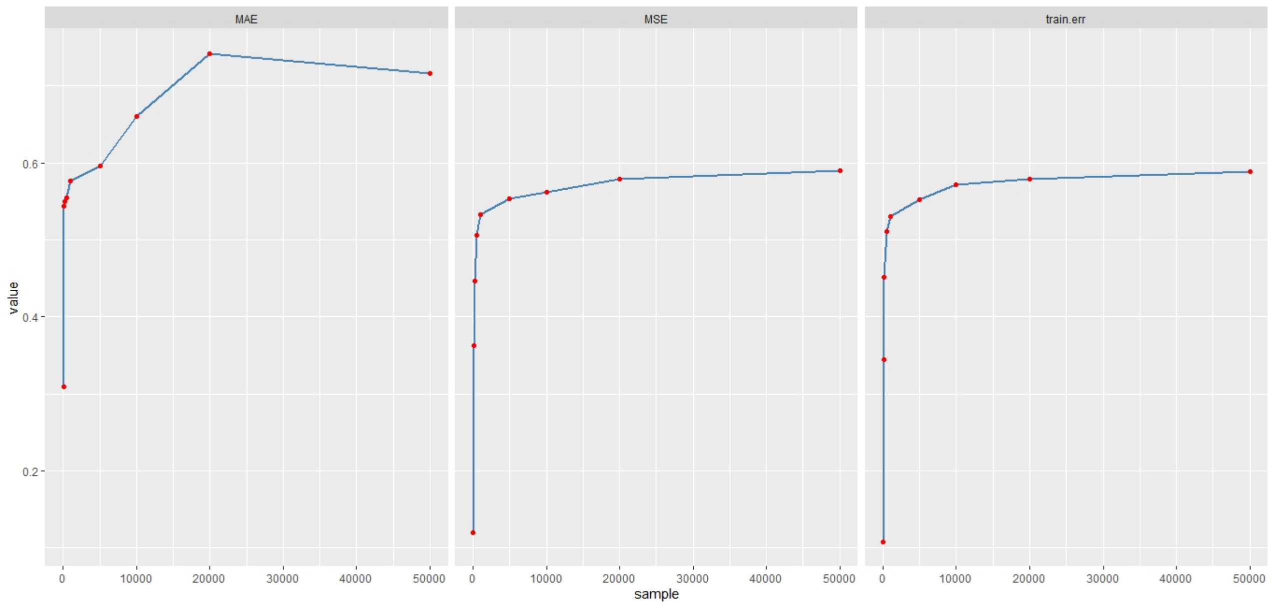

Figure 4.47: **Asymptotic Performance for ReLU activation function at Training set of 80**

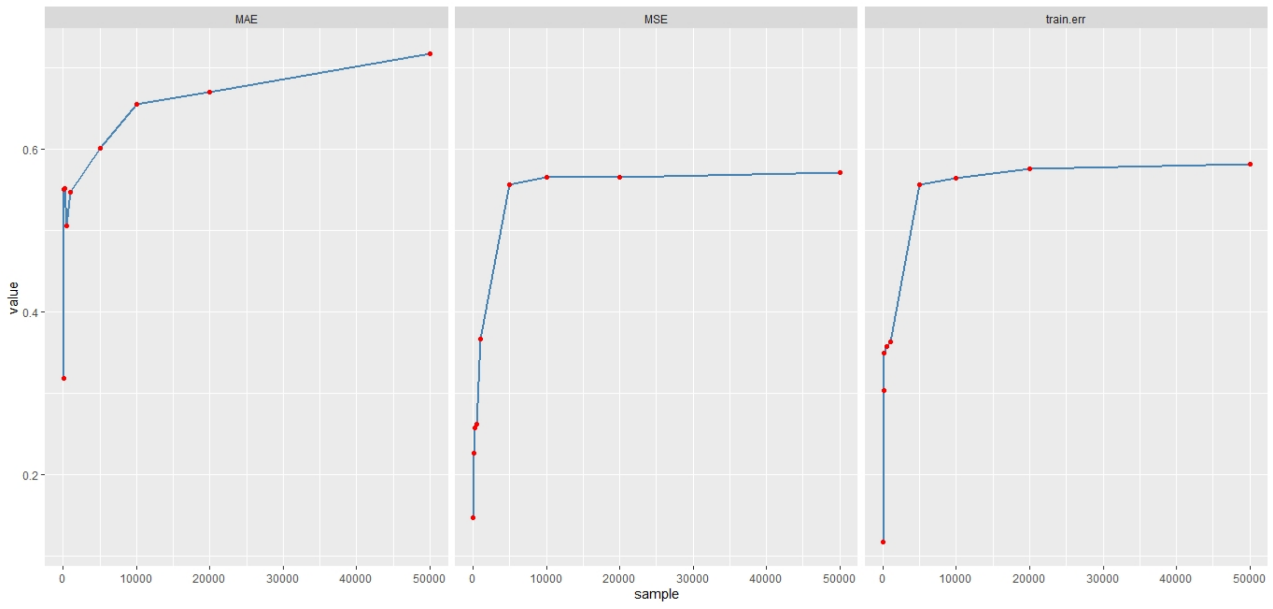

Figure 4.48: **Asymptotic Performance for ReLU activation function at Training set of 90**

Table 4.46 shows the asymptotic Performance result for ReLU activation function at different training set(70, 80, 90) and different sample sizes of 50,100, 200, 500, 1000, 5000, 10000, 20000 and 50000. The performance measured by the mean square error values(MSE), mean absolute error(MAE) and the training erro shows an increasing movement for ReLU activation function as the MSE, MAE and training error values increases as the sample size increases. Also, considering the performance using the training sets, the results shows no consistent pattern as the training set increases.

Figures 4.46,4.47 and 4.48 shows the asymptotic line plot for the ReLU activation function for MSE, MAE and training error at 70%, 80% and 90% training sets respctively.

4.8 ASYMPTOTIC PERFORMANCE FOR SIGMOID ACTIVATION FUNCTION AT VARIOUS SAMPLE SIZES AND ACTIVATION FUNCTIONS

The asymptotic performance of the Sigmoid activation function is displayed in this section. The results at training set of 70, 80 and 90 percents using the sample sizes of 50, 100, 200, 500, 1000, 5000, 10000, 20,000 and 50,000 of the Mean Square Error (MSE), Mean Absolute Error(MAE) and the Training set error of the SSLHT function are displayed.

Table 4.47: **Asymptotic Performance for Sigmoid activation function at different training set**

Training set	Sample size	MSE	MAE	Train.err
	50	0.1834	0.2074	0.1862
	100	0.2034	0.2156	0.2091
	200	0.2174	0.2417	0.2185
	500	0.2685	0.2828	0.283
70	1000	0.3748	0.3483	0.3742
	5000	0.429	0.4752	0.4286
	10000	0.4937	0.5546	0.4935
	20000	0.5585	0.6417	0.5584
	50000	0.5701	0.6569	0.5701
	50	0.1864	0.2797	0.1883
	100	0.203	0.3127	0.2089
	200	0.2142	0.3157	0.2165
	500	0.2491	0.3922	0.2794
80	1000	0.3661	0.4038	0.3687
	5000	0.4073	0.483	0.4168
	10000	0.4923	0.5526	0.4921
	20000	0.5549	0.6091	0.5557
	50000	0.5667	0.6143	0.5666
	50	0.1314	0.2631	0.1198
	100	0.1407	0.1752	0.1379
	200	0.2218	0.3704	0.2196
	500	0.342	0.4891	0.3409
90	1000	0.3218	0.3704	0.32196
	5000	0.4042	0.4891	0.409
	10000	0.4866	0.5474	0.4862
	20000	0.5513	0.637	0.5527
	50000	0.5689	0.6457	0.5688

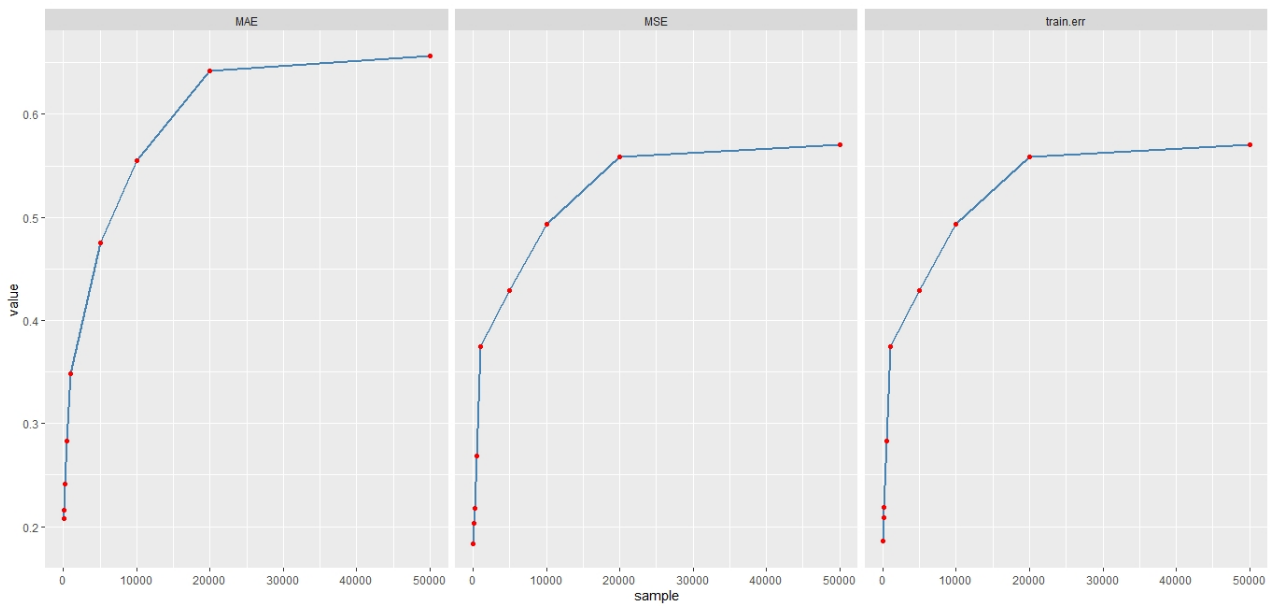

Figure 4.49: **Asymptotic Performance for Sigmoid activation function at Training set of 70**

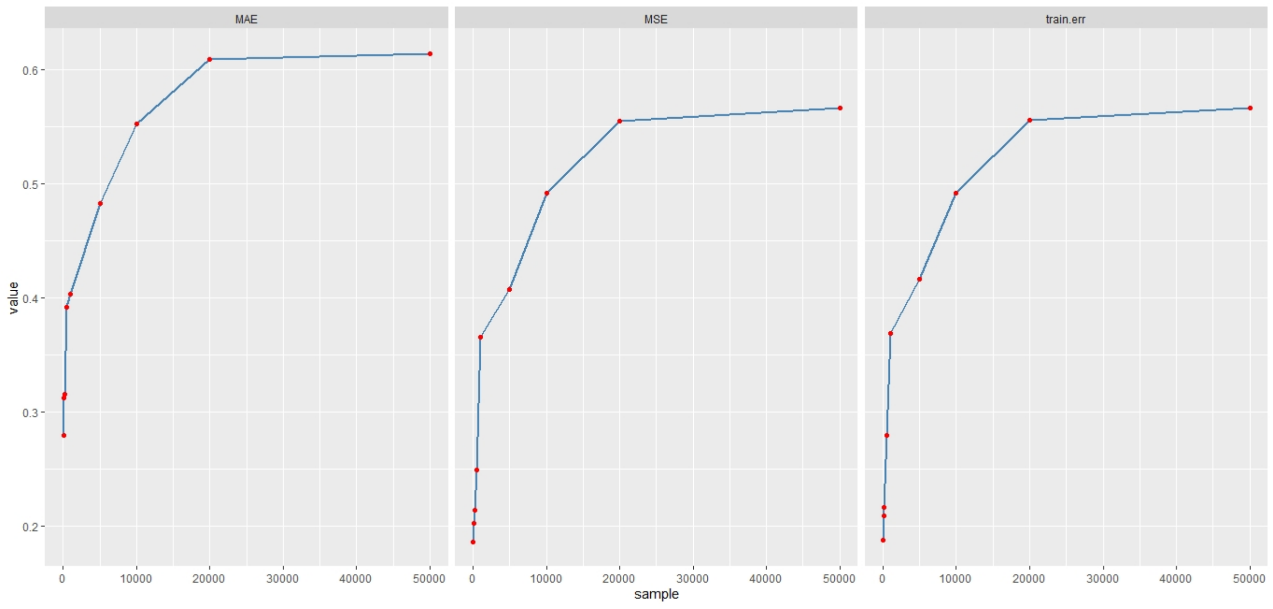

Figure 4.50: **Asymptotic Performance for Sigmoid activation function at Training set of 80**

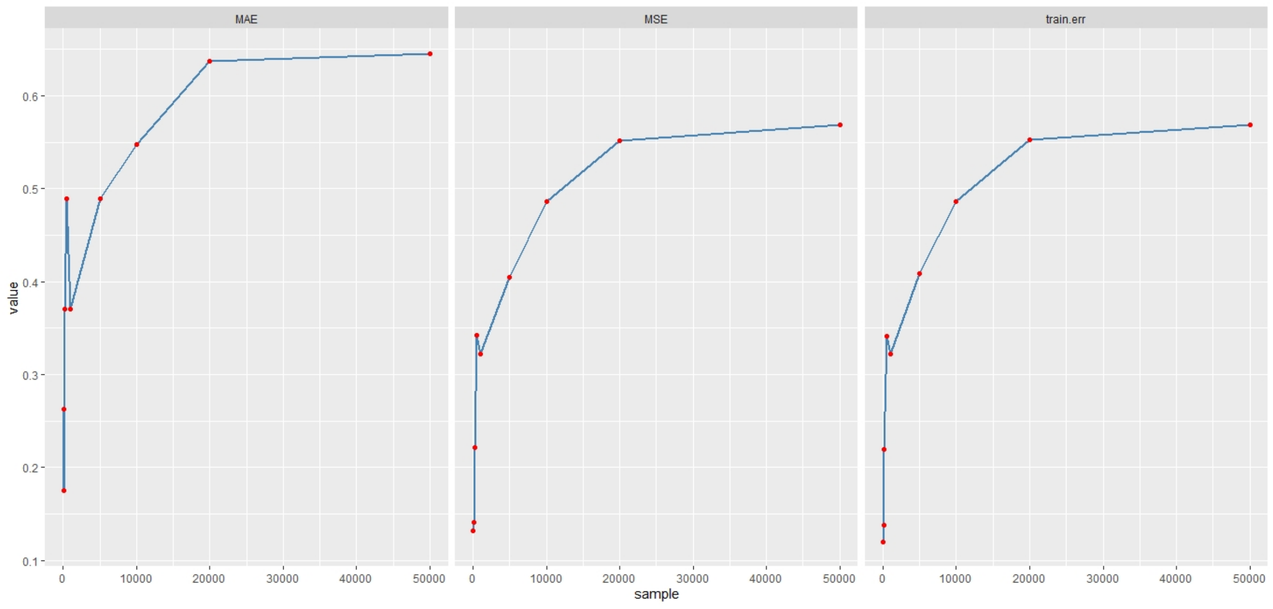

Figure 4.51: **Asymptotic Performance for Sigmoid activation function at Training set of 90**

Table 4.47 shows the asymptotic performance result for Sigmoid activation function at different training set(70, 80, 90) and different sample sizes of 50,100, 200, 500, 1000, 5000, 10000, 20000 and 50000. The performance measured by the mean square error values(MSE), mean absolute error(MAE) and the training erro shows an increasing movement for Sigmoid activation function as the MSE, MAE and training error values increases as the sample size increases. Also, considering the performance using the training sets, the results shows no consistent pattern as the training set increases.

Figures 4.49,4.50 and 4.51 shows the asymptotic line plot for the ReLU activation function for MSE, MAE and training error at 70%, 80% and 90% training sets respctively.

4.9 ASYMPTOTIC PERFORMANCE FOR TANGENT SIGMOID ACTIVATION FUNCTION AT VARIOUS SAMPLE SIZES AND ACTIVATION FUNCTIONS

The asymptotic performance of the Tangent-Sigmoid activation function is displayed in this section. The results at training set of 70, 80 and 90 percents using the sample sizes of 50, 100, 200, 500, 1000, 5000, 10000, 20,000 and 50,000 of the Mean Square Error (MSE), Mean Absolute Error(MAE) and the Training set error of the SSLHT function are displayed.

Table 4.48: **Asymptotic Performance for Tangent-Sigmoid activation function at different training set.**

Training set	Sample size	MSE	MAE	Train.err
70	50	0.19426	0.269	0.18131
	100	0.20105	0.34911	0.20235
	200	0.26467	0.30636	0.2636
	500	0.33945	0.35073	0.33852
	1000	0.48079	0.44036	0.47986
	5000	0.54091	0.58516	0.54054
	10000	0.58131	0.60105	0.58114
	20000	0.58206	0.64424	0.58196
	50000	0.59193	0.65183	0.59189
80	50	0.11295	0.19619	0.10166
	100	0.25287	0.24029	0.23522
	200	0.34993	0.45301	0.3868
	500	0.36301	0.3835	0.36138
	1000	0.49868	0.57088	0.49719
	5000	0.54769	0.59166	0.54715
	10000	0.5939	0.60824	0.59465
	20000	0.5982	0.64148	0.59805
	50000	0.59842	0.65156	0.59836
90	50	0.13562	0.30214	0.1085
	100	0.21569	0.35656	0.17412
	200	0.25766	0.48208	0.25478
	500	0.31729	0.49386	0.31294
	1000	0.33377	0.51578	0.33043
	5000	0.55179	0.69716	0.55069
	10000	0.56248	0.65248	0.56198
	20000	0.58452	0.67934	0.58423
	50000	0.59558	0.68699	0.59444

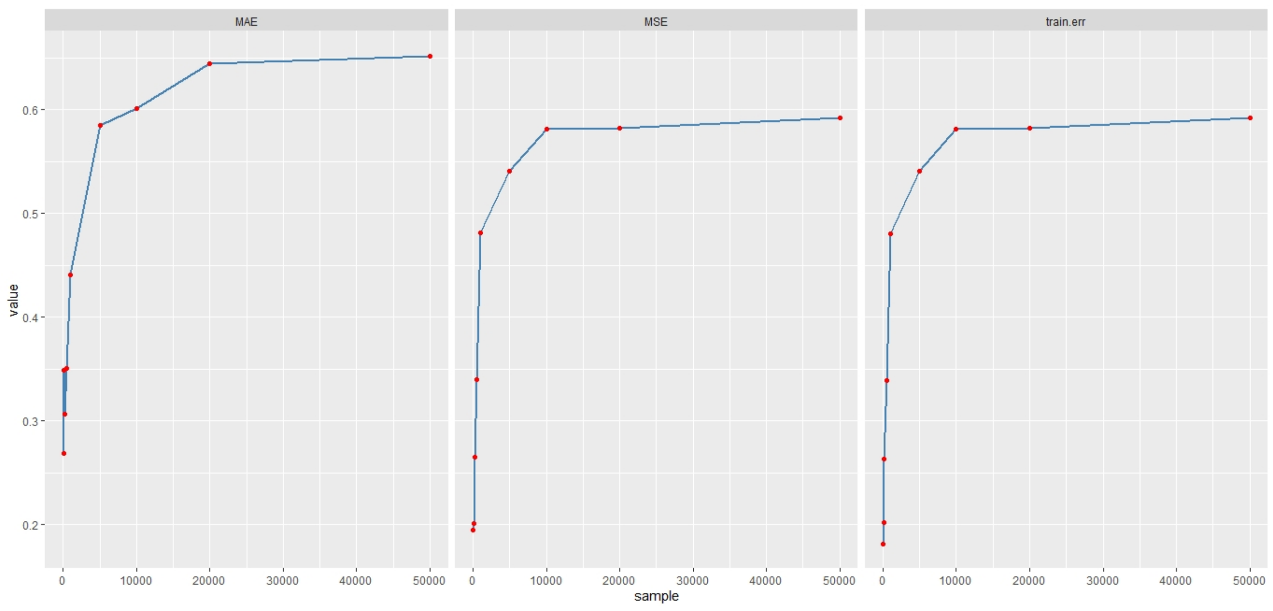

Figure 4.52: **Asymptotic Performance for Tangent-Sigmoid activation function at Training set of 70**

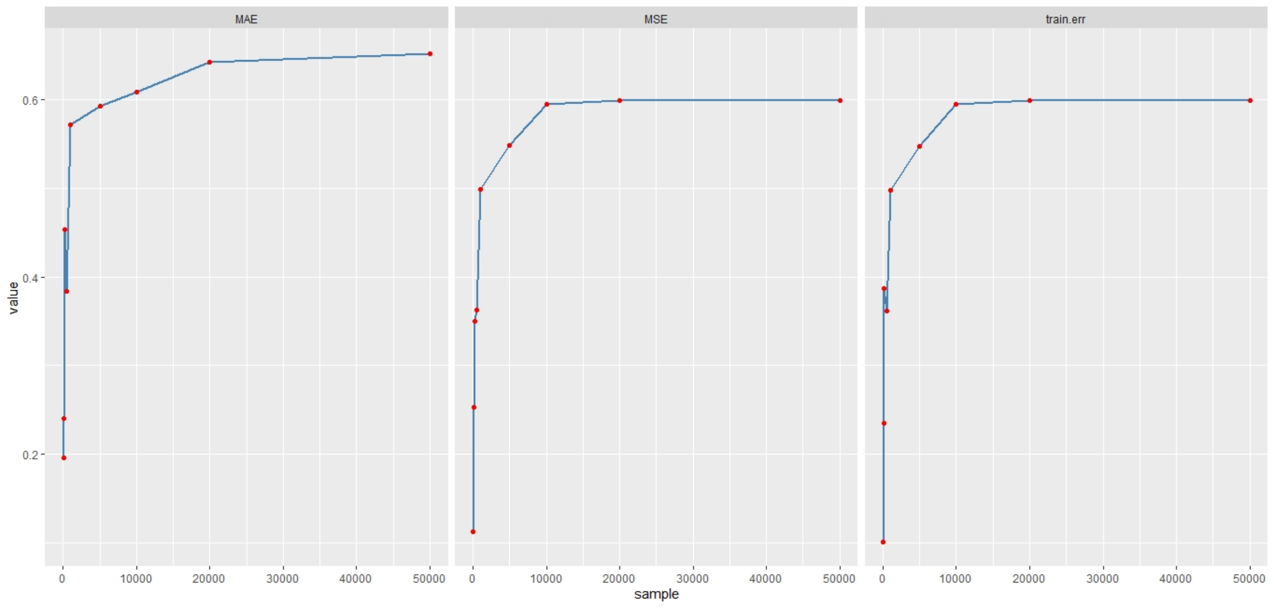

Figure 4.53: **Asymptotic Performance for Tangent-Sigmoid activation function at Training set of 80**

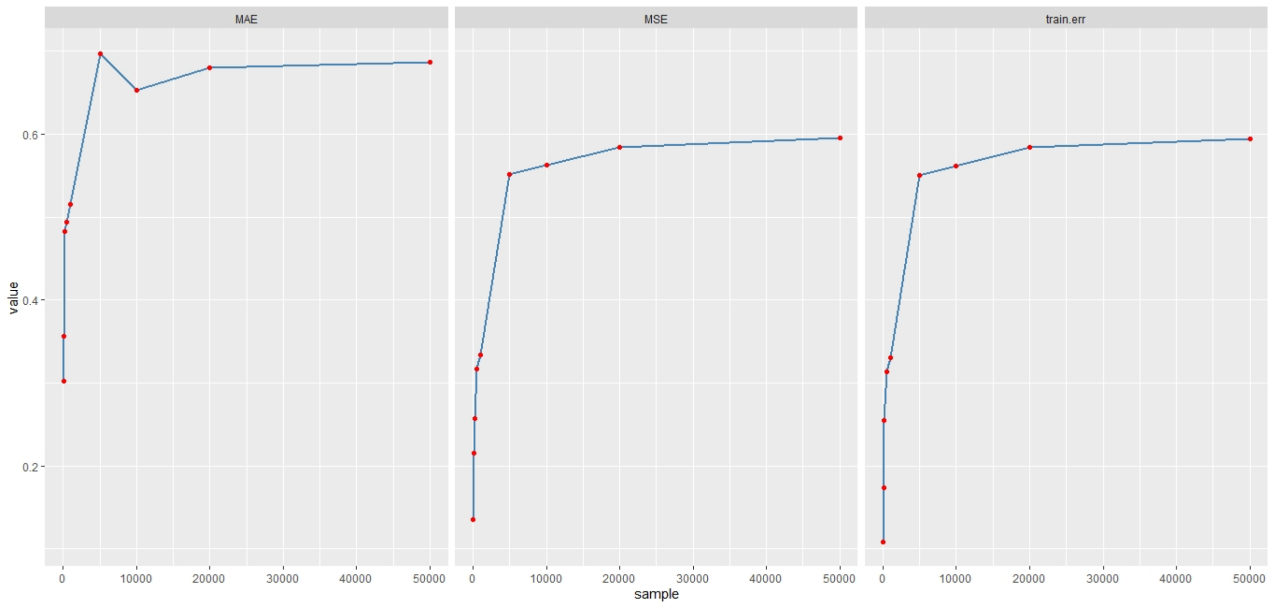

Figure 4.54: **Asymptotic Performance for Tangent-Sigmoid activation function at Training set of 90**

Table 4.48 shows the Asymptotic Performance result for Tangent-Sigmoid activation function at different training set(70, 80, 90) and different sample sizes of 50,100, 200, 500, 1000, 5000, 10000, 20000 and 50000. The performance measured by the mean square error values(MSE), mean absolute error(MAE) and the training erro shows an increasing movement for Sigmoid activation function as the MSE, MAE and training error values increases as the sample size increases. Also, considering the performance using the training sets, the results shows no consistent pattern as the training set increases.

Figures 4.52,4.53 and 4.54 shows the asymptotic line plot for the ReLU activation function for MSE, MAE and training error at 70%, 80% and 90% training sets respctively.

4.10 ASYMPTOTIC PERFORMANCE FOR SSLHT ACTIVATION FUNCTION AT VARIOUS SAMPLE SIZES AND ACTIVATION FUNCTIONS

The asymptotic performance of the SSLHT activation function is displayed in this section. The results at training set of 70, 80 and 90 percents using the sample sizes of 50, 100, 200, 500, 1000, 5000, 10000, 20,000 and 50,000 of the Mean Square Error (MSE), Mean Absolute Error(MAE) and the Training set error of the SSLHT function are displayed.

Table 4.49: **Asymptotic Performance for SSLHT activation function at different training set**

Training set	Sample size	MSE	MAE	Train.err
	50	0.07141	0.13081	0.06665
	100	0.12602	0.21535	0.12948
	200	0.13225	0.19587	0.13005
	500	0.16324	0.20593	0.16215
70	1000	0.16418	0.21388	0.16363
	5000	0.18982	0.23522	0.1897
	10000	0.19195	0.22397	0.19189
	20000	0.20289	0.23954	0.20286
	50000	0.22149	0.24002	0.22148
	50	0.08014	0.10238	0.07213
	100	0.10807	0.10885	0.10666
	200	0.11957	0.18545	0.11658
	500	0.1627	0.21346	0.16108
80	1000	0.16382	0.21575	0.16301
	5000	0.19087	0.23647	0.19068
	10000	0.20045	0.24253	0.20037
	20000	0.20134	0.24845	0.20129
	50000	0.21023	0.25905	0.21021
	50	0.09505	0.14822	0.07604
	100	0.13103	0.15168	0.14027
	200	0.14166	0.17162	0.15075
	500	0.16377	0.22092	0.16219
90	1000	0.1726	0.22441	0.17087
	5000	0.19143	0.23769	0.19105
	10000	0.1978	0.24025	0.19763
	20000	0.19993	0.24762	0.19983
	50000	0.20087	0.25947	0.20083

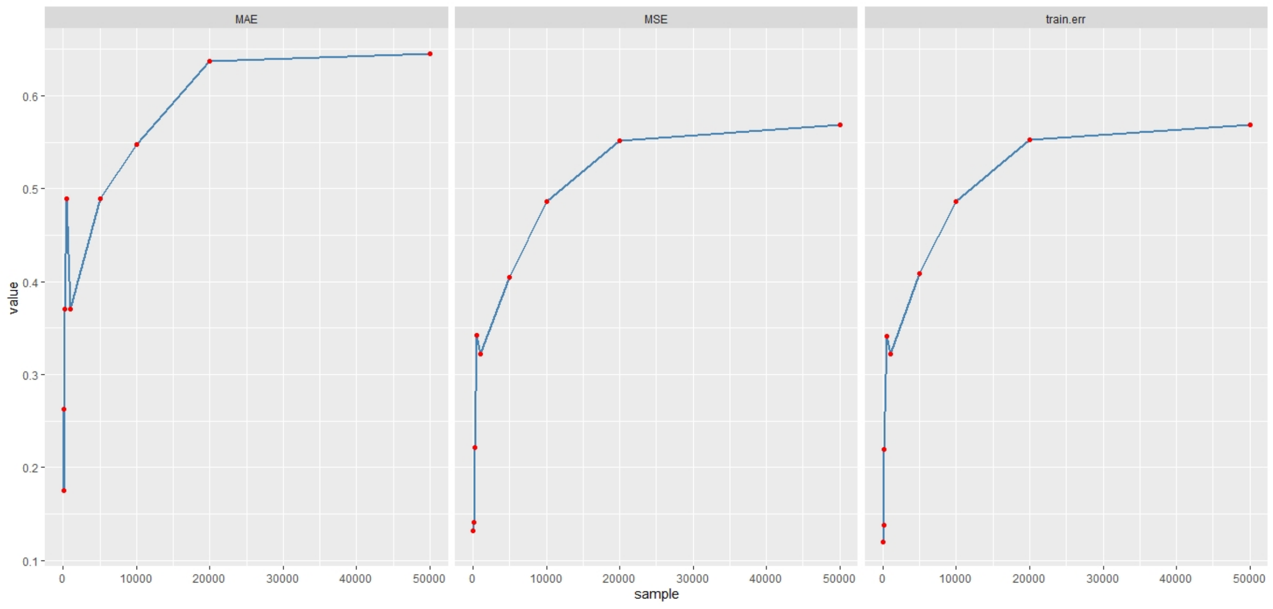

Figure 4.55: **Asymptotic Performance for SSLHT activation function at Training set of 70**

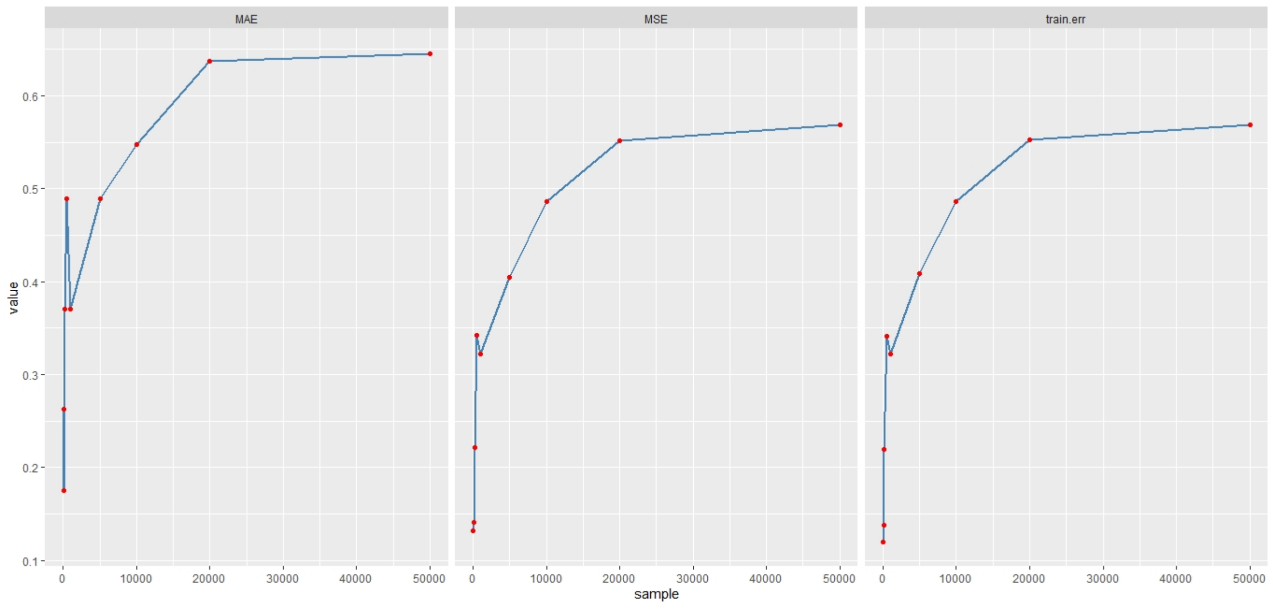

Figure 4.56: **Asymptotic Performance for SSLHT activation function at Training set of 80**

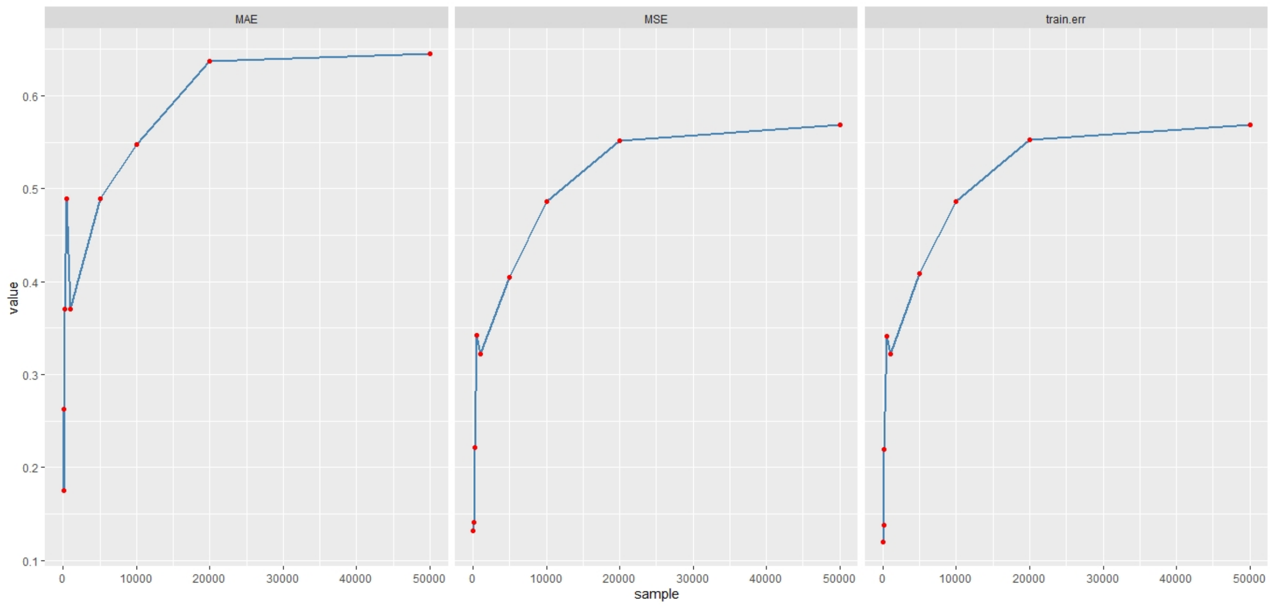

Figure 4.57: **Asymptotic Performance for SSLHT activation function at Training set of 90**

Table 4.49 shows the asymptotic performance result for SSLHT activation function at different training set(70, 80, 90) and different sample sizes of 50,100, 200, 500, 1000, 5000, 10000, 20000 and 50000. The performance measured by the mean square error values(MSE), mean absolute error(MAE) and the training erro shows an increasing movement for Sigmoid activation function as the MSE, MAE and training error values increases as the sample size increases. Also, considering the performance using the training sets, the results shows no consistent pattern as the training set increases.

Figures 4.55,4.56 and 4.57 shows the asymptotic line plot for the ReLU activation function for MSE, MAE and training error at 70%, 80% and 90% training sets respctively.

4.11 ASYMPTOTIC PERFORMANCE FOR SSLHTS ACTIVATION FUNCTION AT VARIOUS SAMPLE SIZES AND ACTIVATION FUNCTIONS

The asymptotic performance of the SSLHTS activation function is displayed in this section. The results at training set of 70, 80 and 90 percents using the sample sizes of 50, 100, 200, 500, 1000, 5000, 10000, 20,000 and 50,000 of the Mean Square Error (MSE), Mean Absolute Error(MAE) and the Training set error of the SSLHT function are displayed.

Table 4.50: **Asymptotic Performance for SSLHTS activation function at different training set**

Training set	Sample size	MSE	MAE	Train.err
	50	0.03216	0.03395	0.03282
	100	0.0331	0.03554	0.03487
	200	0.04241	0.04285	0.04187
	500	0.04454	0.05105	0.04425
70	1000	0.04715	0.05235	0.04699
	5000	0.05701	0.06061	0.05697
	10000	0.05764	0.06224	0.05762
	20000	0.05898	0.06494	0.05897
	50000	0.05951	0.06505	0.05951
	50	0.03499	0.0234	0.03149
	100	0.03744	0.03844	0.03357
	200	0.04875	0.04972	0.03901
	500	0.04987	0.0607	0.04443
80	1000	0.0533	0.06317	0.04706
	5000	0.05733	0.06399	0.05727
	10000	0.0582	0.06681	0.05817
	20000	0.06051	0.06691	0.06049
	50000	0.06131	0.06776	0.06125
	50	0.03977	0.04389	0.03182
	100	0.03801	0.04322	0.03321
	200	0.04824	0.04612	0.04682
	500	0.05005	0.05136	0.04905
90	1000	0.05011	0.05299	0.04961
	5000	0.0575	0.06336	0.05739
	10000	0.0584	0.06512	0.05835
	20000	0.06008	0.06635	0.06005
	50000	0.06732	0.07188	0.06731

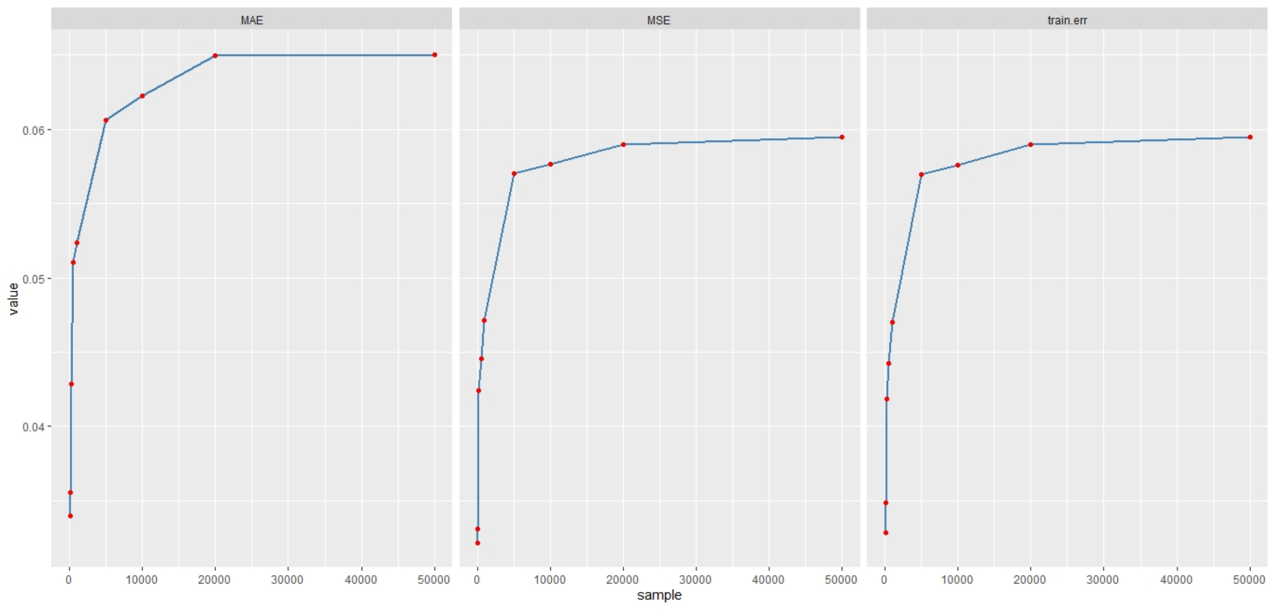

Figure 4.58: **Asymptotic Performance for SSLHTS activation function at Training set of 70**

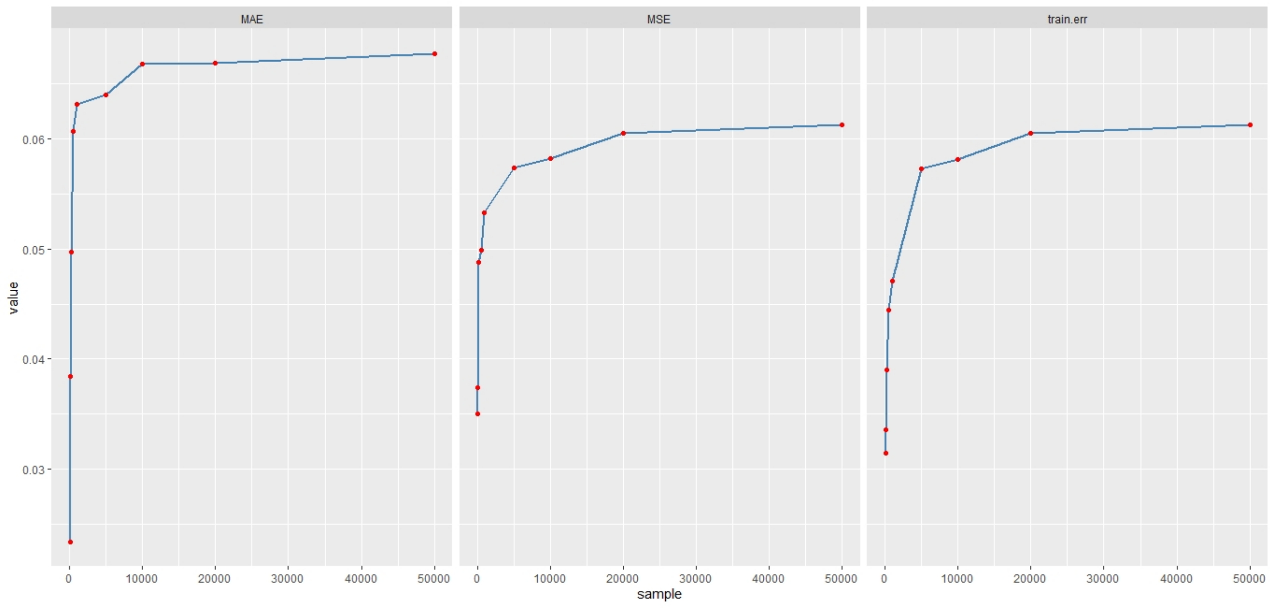

Figure 4.59: **Asymptotic Performance for SSLHTS activation function at Training set of 80**

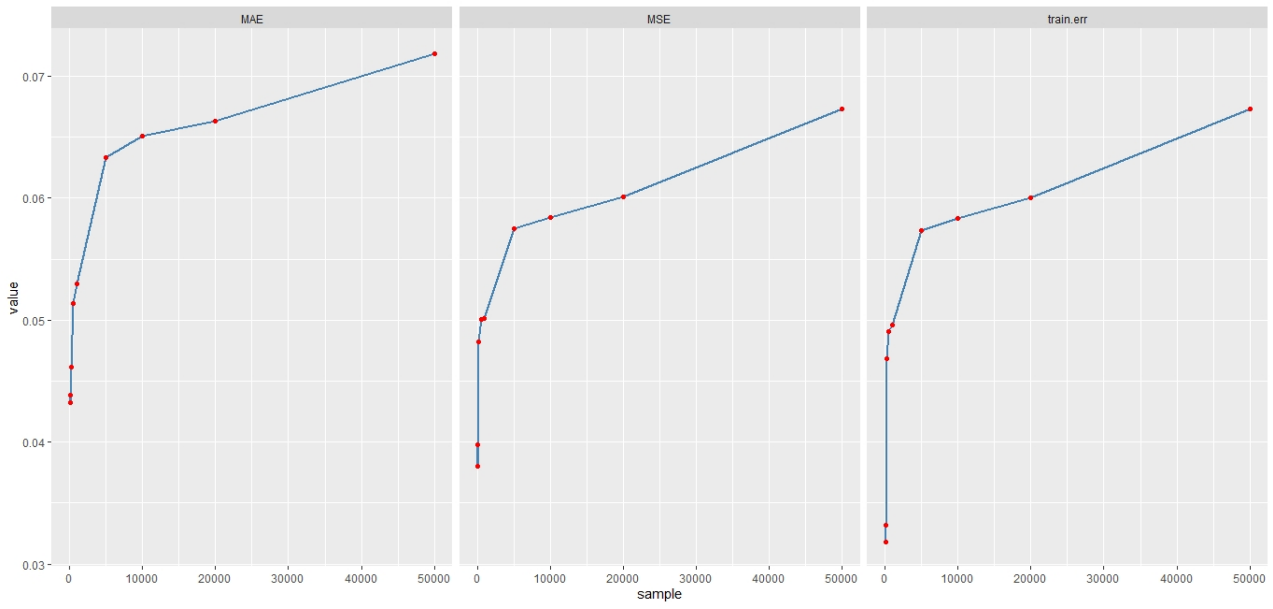

Figure 4.60: **Asymptotic Performance for SSLHTS activation function at Training set of 90**

Table 4.50 shows the asymtotic performance result for SSLHTS activation function at different training set(70, 80, 90) and different sample sizes of 50,100, 200, 500, 1000, 5000, 10000, 20000 and 50000. The performance measured by the mean square error values(MSE), mean absolute error(MAE) and the training erro shows an increasing movement for Sigmoid activation function as the MSE, MAE and training error values increases as the sample size increases. Also, considering the performance using the training sets, the results shows no consistent pattern as the training set increases.

Figures 4.58,4.59 and 4.60 shows the asymptotic line plot for the SSLHTS activation function for MSE, MAE and training error at 70%, 80% and 90% training sets respctively.

4.12 PERFORMANCE EVALUATION FOR THE ACTIVATION FUNCTIONS

In this section, the performance of the activation function are examined. The activation functions are compared in performance using the mean square error, mean absolute error and the training error. The results are compared at training sets of 70, 80 and 90 percents using the sample sizes of 50, 100, 200, 500, 1000, 5000, 10000, 20,000 and 50,000 fpr each of the activation functions.

Table 4.51: **Performance evaluation for the activation functions using Mean Square Error**

Training size	SAMPLE SIZE	RELU	SIGMOID	TANGENT-SIG	SSLHT	SSLHTS
70	50	0.1631	0.1834	0.19426	0.07141	0.03216
	100	0.2412	0.2034	0.20105	0.12602	0.0331
	200	0.2508	0.2174	0.26467	0.13225	0.04241
	500	0.283	0.2685	0.33945	0.16324	0.04454
	1000	0.3163	0.3748	0.48079	0.16418	0.04715
	5000	0.4696	0.429	0.54091	0.18982	0.05701
	10000	0.5044	0.4937	0.58131	0.19195	0.05764
	20000	0.5065	0.5585	0.58206	0.20289	0.05898
	50000	0.5726	0.5701	0.59193	0.22149	0.05951
80	50	0.1195	0.1864	0.11295	0.08014	0.03499
	100	0.363	0.203	0.25287	0.10807	0.03744
	200	0.4463	0.2142	0.34993	0.11957	0.04875
	500	0.5062	0.2491	0.36301	0.1627	0.04987
	1000	0.5323	0.3661	0.49868	0.16382	0.0533
	5000	0.553	0.4073	0.54769	0.19087	0.05733
	10000	0.5616	0.4923	0.5939	0.20045	0.0582
	20000	0.5795	0.5549	0.5982	0.20134	0.06051
	50000	0.5898	0.5667	0.59842	0.21023	0.06131
90	50	0.1468	0.1314	0.13562	0.09505	0.03977
	100	0.2258	0.1407	0.21569	0.13103	0.03801
	200	0.2578	0.2218	0.25766	0.14166	0.04824
	500	0.262	0.342	0.31729	0.16377	0.05005
	1000	0.3664	0.3218	0.33377	0.1726	0.05011
	5000	0.5566	0.4042	0.55179	0.19143	0.0575
	10000	0.5649	0.4866	0.56248	0.1978	0.0584
	20000	0.5657	0.5513	0.58452	0.19993	0.06008
	50000	0.5711	0.5689	0.59558	0.20087	0.06732

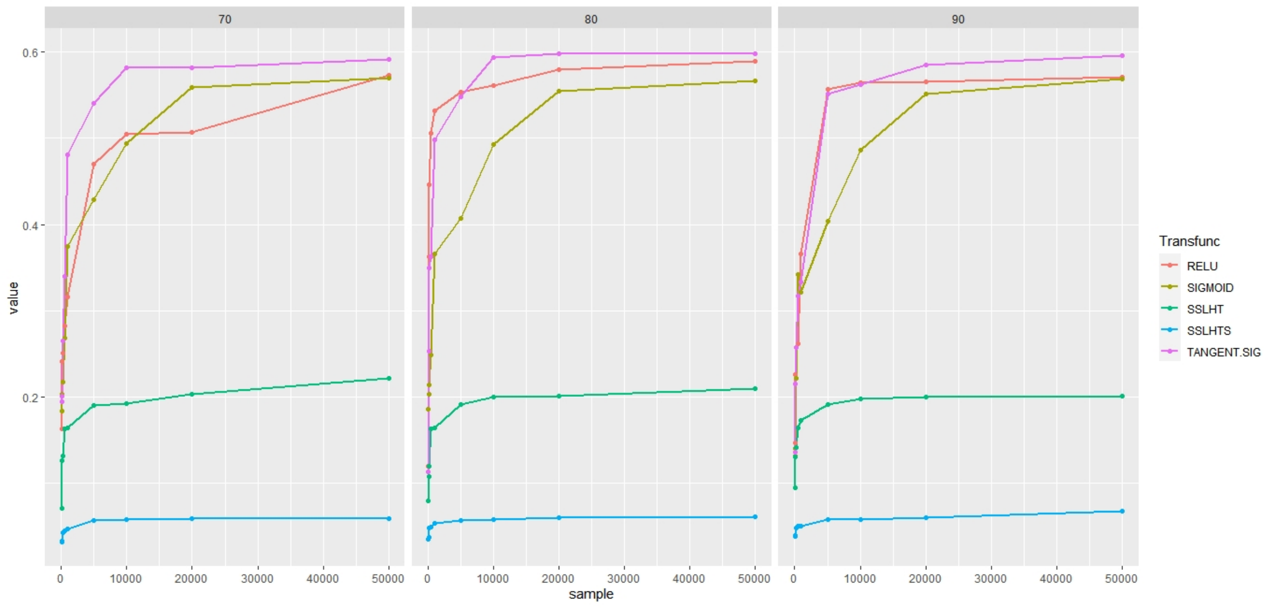

Figure 4.61: **Performance evaluation for the activation functions using Mean Square Error**

Table 4.51 shows the performance of the activation functions using the Mean Square Error(MSE) as the performance measure. They are compared at different sample sizes and at different training sets. The results shows that at the training set of 70, the ReLU activation function is considered best among the HOMAFs, while SSLHTS is better among the HETAFs and they produced minimum MSE values. But in all at training set of 70, HETAFs activation functions performed better compared to the HOMAFs activation functions with minimum MSE values. At training set of 80, ReLU activation function still perform best among HOMAFs while SSLHTS still perform better among the HETAFs. At training set of 90, Sigmoid activation function perform best among the HOMAFs as it produced minimum MSE values. But in summary, the HETAFs activation functions has minimum MSE values compared to HOMAFs and this make them better in performance.

Figure 4.61 above shows the performance evaluation plot for the activation functions using the Mean Square Error values. It shows that SSLHTS activation function produced lowest Mean Square Error values at all the considered sample sizes and at all the training sets considered, followed by SSLHT activation function. Other activation functions i.e ReLU, Sigmoid and Tangent Sigmoid produced higher Mean Square Error values compared to the first two mentioned.

Table 4.52: **Performance evaluation for the activation functions using Mean Absolute Error**

Training size	SAMPLE SIZE	ReLU	SIGMOID	TANGENT-SIG	SSLHT	SSLHTS
70	50	0.2465	0.2074	0.269	0.13081	0.03395
	100	0.2534	0.2156	0.34911	0.21535	0.03554
	200	0.3104	0.2417	0.30636	0.19587	0.04285
	500	0.3297	0.2828	0.35073	0.20593	0.05105
	1000	0.4807	0.3483	0.44036	0.21388	0.05235
	5000	0.6896	0.4752	0.58516	0.23522	0.06061
	10000	0.6634	0.5546	0.60105	0.22397	0.06224
	20000	0.7452	0.6417	0.64424	0.23954	0.06494
	50000	0.7188	0.6569	0.65183	0.24002	0.06505
80	50	0.3091	0.2797	0.19619	0.10238	0.0234
	100	0.5439	0.3127	0.24029	0.10885	0.03844
	200	0.55	0.3157	0.45301	0.18545	0.04972
	500	0.5541	0.3922	0.3835	0.21346	0.0607
	1000	0.5765	0.4038	0.57088	0.21575	0.06317
	5000	0.5955	0.483	0.59166	0.23647	0.06399
	10000	0.6603	0.5526	0.60824	0.24253	0.06681
	20000	0.7423	0.6091	0.64148	0.24845	0.06691
	50000	0.716	0.6143	0.65156	0.25905	0.06776
90	50	0.3183	0.2631	0.30214	0.14822	0.04389
	100	0.5499	0.1752	0.35656	0.15168	0.04322
	200	0.5519	0.3704	0.48208	0.17162	0.04612
	500	0.5055	0.4891	0.49386	0.22092	0.05136
	1000	0.5473	0.3704	0.51578	0.22441	0.05299
	5000	0.6006	0.4891	0.69716	0.23769	0.06336
	10000	0.6543	0.5474	0.65248	0.24025	0.06512
	20000	0.6701	0.637	0.67934	0.24762	0.06635
	50000	0.7173	0.6457	0.68699	0.25947	0.07188

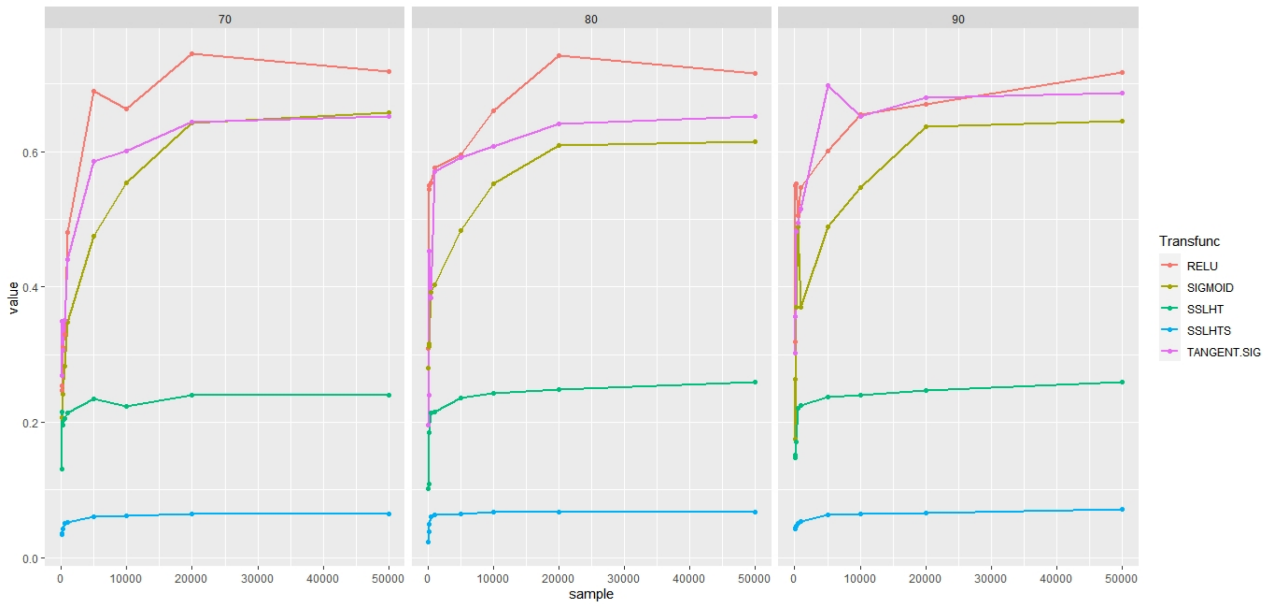

Figure 4.62: **Performance evaluation for the activation functions using Mean Absolute Error**

Table 4.52 shows the performance of the activation functions using the Mean Absolute Error(MAE) as the performance measure. They are compared at different sample sizes and at different training sets. The results shows that at the training set of 70, the Sigmoid activation function is considered best among the HOMAFs, while SSLHTS is better among the HETAFs and they produced minimum MAE values. But in all at training set of 70, HETAFs activation functions performed better compared to the HOMAFs activation functions with minimum MSE values. At training set of 80, Tangent-Sigmoid activation function still perform best among HOMAFs while SSLHTS perform better among the HETAFs. At training set of 90, Sigmoid activation function perform best among the HOMAFs as it produced minimum MAE values. But in summary, the HETAFs activation functions has minimum MAE values compared to HOMAFs and this make them better in performance.

Figure 4.62 above shows the performance evaluation plot for the activation functions using the Mean Absolute Error values. It shows that SSLHTS activation function produced lowest Mean Absolute Error values at all the considered sample sizes and at all the training sets considered, followed by SSLHT activation function. Other activation functions i.e ReLU, Sigmoid and Tangent Sigmoid produced higher Mean Square Error values compared to the first two mentioned.

Table 4.53: **Performance evaluation for the activation functions using Training Error**

Training size	SAMPLE SIZE	ReLU	SIGMOID	TANGENT-SIG	SSLHT	SSLHTS
70	50	0.1522	0.1862	0.18131	0.06665	0.03282
	100	0.2498	0.2091	0.20235	0.12948	0.03487
	200	0.26	0.2185	0.2636	0.13005	0.04187
	500	0.2818	0.283	0.33852	0.16215	0.04425
	1000	0.3153	0.3742	0.47986	0.16363	0.04699
	5000	0.466	0.4286	0.54054	0.1897	0.05697
	10000	0.5043	0.4935	0.58114	0.19189	0.05762
	20000	0.5064	0.5584	0.58196	0.20286	0.05897
	50000	0.5725	0.5701	0.59189	0.22148	0.05951
80	50	0.1076	0.1883	0.10166	0.07213	0.03149
	100	0.3449	0.2089	0.23522	0.10666	0.03357
	200	0.4515	0.2165	0.3868	0.11658	0.03901
	500	0.5111	0.2794	0.36138	0.16108	0.04443
	1000	0.5306	0.3687	0.49719	0.16301	0.04706
	5000	0.5524	0.4168	0.54715	0.19068	0.05727
	10000	0.5713	0.4921	0.59465	0.20037	0.05817
	20000	0.5793	0.5557	0.59805	0.20129	0.06049
	50000	0.5892	0.5666	0.59836	0.21021	0.06125
90	50	0.1174	0.1198	0.1085	0.07604	0.03182
	100	0.3032	0.1379	0.17412	0.14027	0.03321
	200	0.3496	0.2196	0.25478	0.15075	0.04682
	500	0.3568	0.3409	0.31294	0.16219	0.04905
	1000	0.3627	0.32196	0.33043	0.17087	0.04961
	5000	0.5555	0.409	0.55069	0.19105	0.05739
	10000	0.5644	0.4862	0.56198	0.19763	0.05835
	20000	0.5754	0.5527	0.58423	0.19983	0.06005
	50000	0.581	0.5688	0.59444	0.20083	0.06731

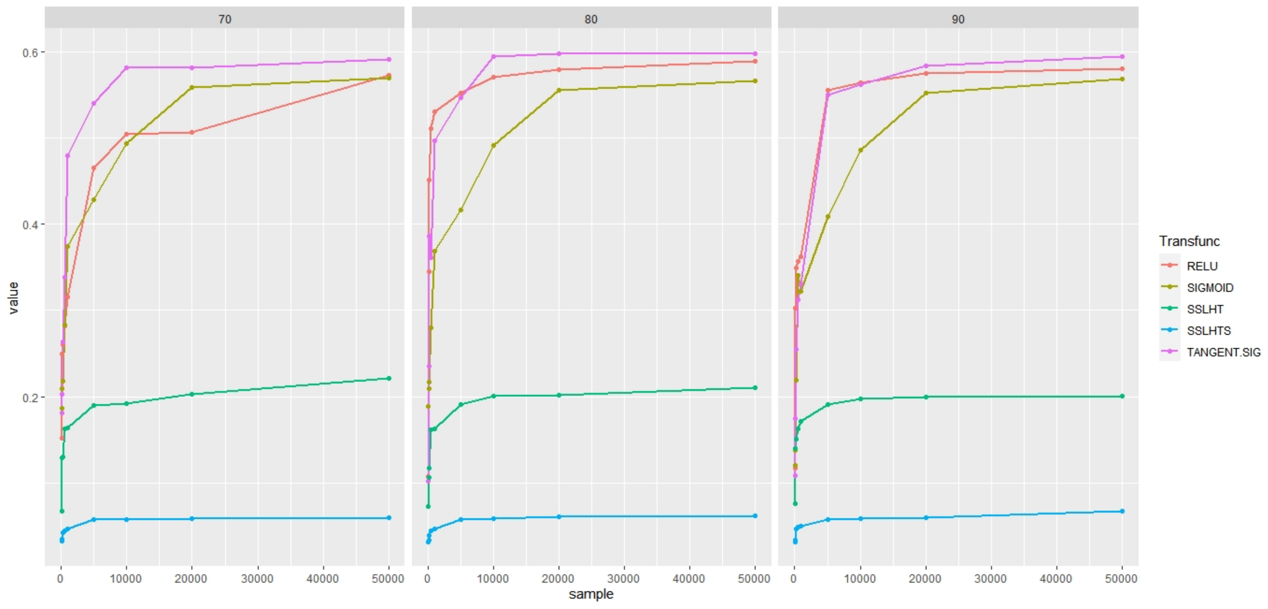

Figure 4.63: **Performance evaluation for the activation functions using Training Error**

Table 4.53 shows the performance of the activation functions using the training error as the performance measure. They are compared at different sample sizes and at different training sets. The results shows that at the training set of 70, the ReLU activation function is considered best among the HOMAFs, while SSLHTS is better among the HETAFs and they produced minimum training error values. But in all at training set of 70, HETAFs activation functions performed better compared to the HOMAFs activation functions with minimum training error values. At training set of 80, Sigmoid activation function still perform best among HOMAFs while SSLHTS perform better among the HETAFs. At training set of 90, Sigmoid activation function perform best among the HOMAFs as it produced minimum MAE values. But in summary, the HETAFs activation functions has minimum MAE values compared to HOMAFs and this implies better in performance.

Figure 4.63 above shows the performance evaluation plot for the activation functions using the training error values. It shows that SSLHTS activation function produced lowest training error values at all the considered sample sizes and at all the training sets considered, followed by SSLHT activation function. Other activation functions i.e ReLU, Sigmoid and Tangent Sigmoid produced higher training error values compared to the first two mentioned.

4.13 RESULT OF ANALYSIS FOR THE TERRORISM DATA

In this section, the results for each activation function considered and their performance measures are displayed. Table 4.54 shows information about the data used and Table 4.55 and Table 4.56 show the model performance for the activation functions used. For this study, 508 responses were received and Table 4.54 shows the frequency in cross tabulation between the regions and the suicide attack response. Figures 4.64 shows composite bar chart for the attack type by zones in Nigeria, while Figure 4.65 shows the bar chart for the weapon type by zones in Nigeria.

Table 4.54: **Cross tabulation table between Region and Suicide Attack response**

Region	suicide attack			
Zone	No	Yes	Total	Percent
North Central	7	5	12	2.4%
North East	40	32	72	14.2%
North West	75	87	162	31.9%
South east	32	44	76	15.0%
South South	6	4	10	2.0%
South West	86	90	176	34.6%

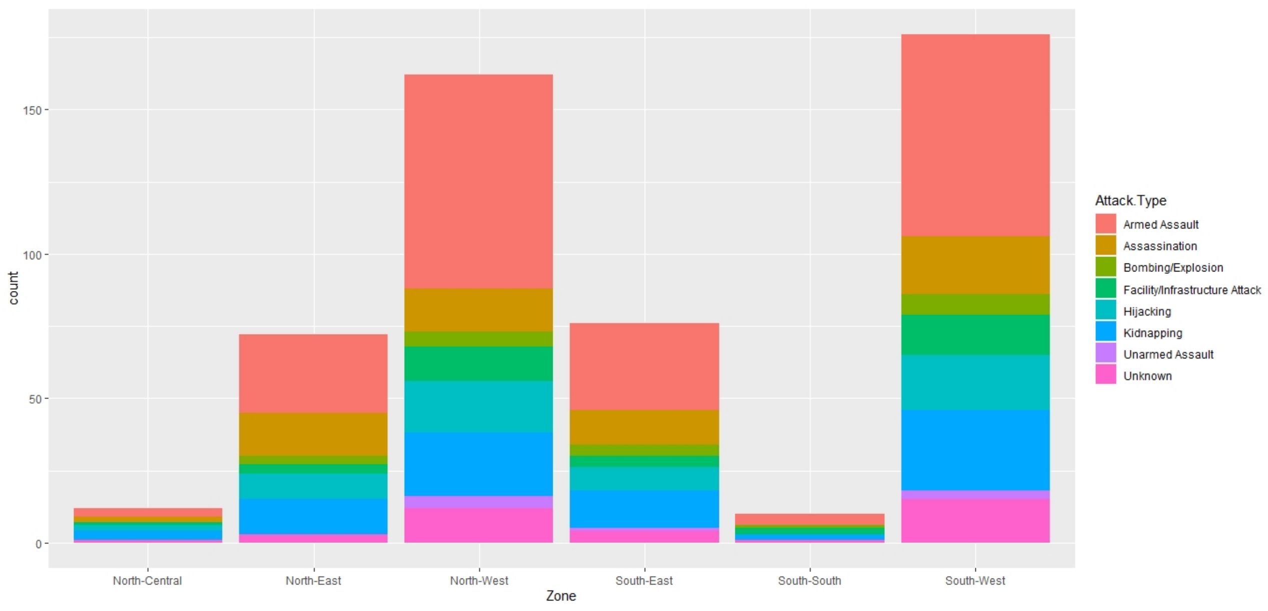

Figure 4.64: **Attack type by Zones in Nigeria**

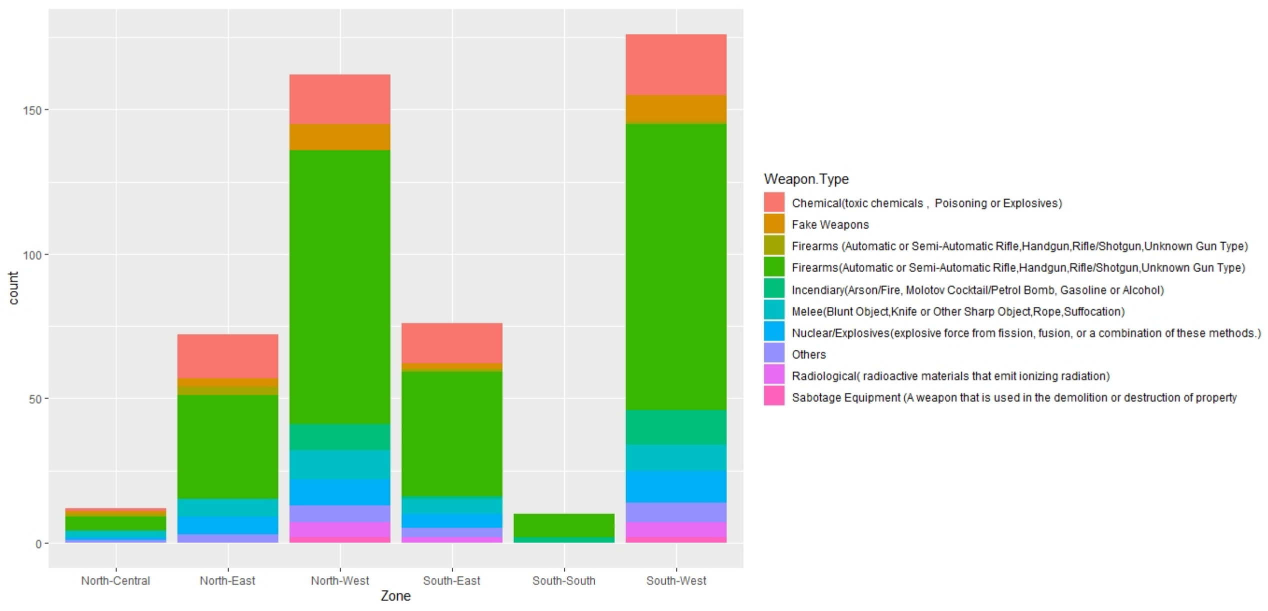

Figure 4.65: **Weapon type by zones in Nigeria**

Table 4.55: **Model Performance for the Suicide Attack**

Model	Accuracy	Precision	Recall	F1-Score
RELU	0.878	0.861	0.878	0.896
Sigmoid	0.854	0.876	0.854	0.865
Tangent-sigmoid	0.902	0.913	0.902	0.907
SSHLT	0.913	0.926	0.917	0.921
SSHLTS	0.982	0.975	0.988	0.981

Table 4.55 above shows the performance measures for the five activation functions for the Suicide Attack. The RELU, Sigmoid, Tangent-Sigmoid, SSHLT and SSHLTS activation functions produced 87.8%, 85.4%, 90.2%, 91.3% and 98.2% accuracy respectively. For precision, RELU, Sigmoid, Tangent-Sigmoid, SSHLT and SSHLTS activation functions produced, 86.1%, 87.6%, 90.2%, 92.6% and 97.5% respectively. For Recall performance, the models produced, 87.8%, 85.4%, 91.3%, 91.7%, and 98.8% respectively. Their F1-score results show 89.6%, 86.5%, 90.7%, 92.1%, and 98.1% respectively.

Table 4.56: **Relative Importance for the Suicide Attack**

Features	Relative Importance
Number of Perpetrators	62.320
Attack Type	17.852
Weapon Type	11.613
Target/Victim Type	8.215
State of the incident	0.000

Table 4.56 shows that, the Number of Perpetrators takes 62.320% determination on whether the terrorist attack will be suicidal or not. Attack Type takes 17.852%, Weapon Type determines 11.613%, Target/Victim Type determines 8.215% and State of the incident determines zero percentage.

Table 4.57: **Model Performance for the Attack Type**

Activation functions	Accuracy	Precision	Recall	F1-score
RELU	0.902	0.913	0.902	0.886
Sigmoid	0.854	0.876	0.854	0.805
Tangent sigmoid	0.913	0.926	0.917	0.951
SSHLT	0.975	0.975	0.975	0.974
SSHLTS	0.976	0.976	0.976	0.975

Table 4.57 shows the performance measures for the five activation functions considered. The RELU, Sigmoid, Tangent-Sigmoid, SSHLT and SSHLTS activation functions produced 90.2%, 85.4%, 91.3%, 97.5% and 97.6% accuracy respectively. For precision, RELU, Sigmoid, Tangent-Sigmoid, SSHLT and SSHLTS activation functions produced 91.3%, 87.6%, 92.6%, 97.5% and 97.6% respectively. For Recall performance, the models produced, 90.2%, 85.4%, 91.7%, 97.5%, and 97.6% respectively. Their F1-score results show 88.6%, 80.5%, 95.1%, 97.4%, and 97.5% respectively.

Table 4.58: **Relative Importance for Attack Type**

Features	Relative Importance
Number of Perpetrators	68.271
Weapon Type	16.964
Target/Victim Type	10.469
State of the incident	4.3

Table 4.58 shows the relative importance table for the attack type. The table shows that, the Number of Perpetrators takes 68.271% in determining the type of the Attack the terrorist are to carry out. Weapon Type determines 16.964%, Target/Victim Type determines 10.469% and State of the incident determines 4.3 percent.

4.14 DISCUSSION OF RESULTS

This section discusses the findings from the analysis carried out in this study. The aim of this study is to reduce overfitting in Neural network modelling using Bayesian approach with heterogeneous activation function. This is truely achieved in this study by looking closely to the results obtained from the mean square error and the train error. The closer these values, the lower the problem of overfitting. Looking at the reults obtained, the performance evaluation for the activation functions using the mean square error and Training Error deduced a close value between them.

Three homogeneous transfer functions and two heterogeneous transfer functions were taken into consideration in this study at varied training set percentages of 70%, 80%, and 90%. ReLU produced the majority of the lowest mean square errors (MSE), mean absolute errors (MAE), and training errors among homogeneous activation functions. In contrast, the two heterogeneous transfer functions produced the lowest mean square errors, mean absolute errors, and training errors when compared to the homogeneous transfer functions considered, which makes them more accurate predictors. The findings also demonstrate that the mean square error value reduces with increasing sample size.

Mean Square Error (MSE) was utilized in this study to evaluate each ANN model's performance. Finally, across all sample sizes, ReLU delivered the majority of the lowest mean square errors (MSEs). Additionally, the mean square error typically increases as the training proportion does too. In terms of prediction metrics, the performance of the investigated heterogeneous transfer functions was good because they almost always produced the lowest mean square error for training sets and at various levels of sample sizes. This result supports the work by Ogundunmade and Adepoju (2021) on the classical performance of artificial neural network model with heterogeneous transfer functions.

Diverse studies on the application of homogeneous and heterogeneous activation functions in neural network models have been published in the literature. The traditional approach was used by Udomboso, C. G. (2013) to compare the homogeneous and heterogeneous activation functions. In order to address the issue of over-fitting, we created a Bayesian framework for the neural network model in this study. Both the homogeneous and the heterogeneous activation functions were taken into account in this work. Therefore, this research found that a neural network model with a Bayesian framework and heterogeneous activation functions reduced over-fitting.

The results discussion revealed that all of the study's aim and objectives had been achieved, which revealed gaps in the research and filled gaps in the body of existing knowledge.

Chapter 5
SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 SUMMARY

In this thesis, Bayesian Neural Network (BNN) with heterogenous activation function has been established to perform better in prediction compare to the the use of homogeneous activation function. The summary of the work is described below.

The Posterior mean, standard deviation and NSE for ReLU, Sigmoid, Tangent-Sigmoid, SSLHT and SSLHTS activation functions at different sample sizes and training sets showed that the parameter estimate values for β and γ increase tends closed to the assume values as the sample size increase. The standard deviation values decrease as the sample size increase for both β and γ.

The Asymtotic Performance for the activation functions using the mse, mae and train error values and at different sample sizes and training sets showed that MSE, MAE and train error values increase as the sample size increases at considered level of training sets. The close values between the MSE and train error also shows reduction in overfitting as stated in the aim of the study. But the SSLHTS showed better closer values between the MSE and train error.

The performance evaluation for the activation functions using the mse, mae and train error values was also examined. It shows that SSLHTS activation function produced lowest MSE, MAE and train error values at all the considered sample sizes and at all the training sets considered, followed by SSLHT activation function. Other activation functions i.e ReLU, Sigmoid and Tangent Sigmoid produced higher train error values compared to the first two mentioned. Among the

HOMAFs, ReLU activation function is considered to have performed better compare to Sigmoid and Tangent-Sigmoid activation functions.

5.2 CONCLUSION

In this study, it can be deduced that:
The weights of the BNN at different samples sizes and training sets for both homogeneous and heterogeneous activation functions were obtained.
The performance of heterogeneous activation functions (SSHLT and SSHLTS) compared to the homogeneous activation functions were better.
The asymptotic performance in terms of mse, mae and train error of the functions show that as the sample size increases, these mentioned performance estimators also increases.
For the real life data, the accuracy and F1-score of the BNN model using the heterogeneous activation function is higher than that of the homogeneous activation functions and this implies better classification performance.

5.3 LIMITATIONS OF THE STUDY

The first limitation encountered in this study was the compuational rigour around estimation of this model. It took highly intensive computer hardware to be able to obtain the results in this study. Also, another limitation of this study is the problem encountered obtaining real life data used to be used. Primary data was finally resulted into for this study.

5.4 RECOMMENDATIONS

This study therefore recommends the use of Bayesian Neural Network modelling with heterogeneous activation function in various machine learning and data science areas, especially in deep learning. It can be applied in different areas of machine learning and artificial intelligence as models in other to reduce overfitting especially in Neural Network models. Overcoming amelioration in neural network modelling can be reduced using the heterogeneous activation functions. Also, we recommend the model in areas like security detection, image and language detection.

5.5 CONTRIBUTIONS TO KNOWLEDGE

This research work have identified gaps and addressed the methodology of Bayesian Neural Network using heterogenous activation functions. Also the Bayesian Neural Network (BNN) model has been shown analytically and empirically, to be more efficient in terms of reducing overfitting in Bayesian Neural Network modelling. It is also seen in this study that when Bayesian neural network model performs better using heterogenous activation functions than using homogeneous activation using Bayesian approach of estimation . The asymptotic behavior of the activation functions also showed that as the samples increases, the mean square error, mean absolute error and the training error increase.

One of the so-called important models used in data science for pattern and image recognition, computer vision, and other applications is the neural network model. Since the neural network is the fundamental functional component of the model, it cannot be used without the usage of activation functions. Most of these locations indicated above have been predicted in earlier studies using homogeneous transfer functions. It is advised that neural network models for heterogeneous transfer functions be taken into consideration in light of the study's findings. The forecast performance of the heterogeneous transfer functions in this study has demonstrated that improved outcomes will be reached when applied in neural network models for the aforementioned fields of research and many more.

5.6 SUGGESTIONS FOR FURTHER RESEARCH

For future research, one might find it important to

1. Consider other newly developed areas of neural network models like Deep Neural Network(DNN) and Convolutionary Neural Network (CNN).
2. Make use of heterogeneous activation functions in these new areas of Neural Networks
3. explore further into other activation functions that may give better optimal

performances.

4. Determine analytically the number of hidden neurons needed to obtain an optimal performance.

REFERENCES

Anders U. (1996). Statistical Model Building for Neural Networks. AFIR Colloqium. Nunberg, Germany.

Aitchison, L. (2020) A statistical theory of cold posteriors in deep neural networks. *arXiv preprint arXi0*v:2008.05912. https://doi.org/10.48550/arXiv.2008.05912

Aitchison L., Yang, A. X., and Ober, S. W.(2020). Deep kernel processes. arXiv preprint arXiv:2010.01590. https://doi.org/10.48550/arXiv.2010.01590.

Alonso A.G, Fortuin, V. (2021). Exact Langevin Dynamics with Stochastic Gradients.*Proceedings of the 28 th International Conference on Machine Learning*, Bellevue, WA, USA, 2011. *https://arxiv.org/abs/2102.01691.*

Bayes, T. An essay towards solving a problem in the doctrine of chances. *Philosophical transactions of the Royal Society of London*, 53:370-418, 1763. By the late Rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFRS.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. arXiv preprint arXiv:1505.05424. https://doi.org/10.48550/arXiv.1505.05424

Chandra R, He Y (2021) Bayesian neural networks for stock price forecasting before and during COVID-19 pandemic. *PLoS ONE* 16(7): e0253217. https://doi.org/10.1371/journal.pone.0253217.

Christopher Godwin Udomboso(2013). On Some Properties of a Heterogeneous Transfer Function Involving Symmetric Saturated Linear (SATLINS) with Hyperbolic Tangent (TANH)Transfer Functions.*Journal of Modern Applied Statistical Methods.***12**(2), Article 26.

Christopher Godwin Udomboso(2014). On the level of precision of an heterogenous statistical neural network model. PhD thesis, Department of Statistics, University of Ibadan, Nigeria.

Cranmer M, Tamayo D, Rein H, Battaglia P, Hadden S, Armitage PJ, (2021). A Bayesian neural network predicts the dissolution of compact planetary systems. *PNAS*,October 1, 2021, 118 (40) e2026053118. https://doi.org/10.1073/pnas.2026053118.

Fortuin, V. (2020). Exact Langevin Dynamics with Stochastic Gradients. 1–13.

Id, R. C. and He, Y. (2021) Bayesian neural networks for stock price forecasting before and during COVID-19 pandemic. doi: 10.1371/journal.pone.0253217.

Kim, Qh., Ko, J.-H., Kim, S., Park, N., & Jhe, W. (2021). Sequence analysis Bayesian neural network with pretrained protein embedding enhances prediction accuracy of drug-protein interaction, *Bioinformatics*, Volume 37, Issue 20, October 2021, Pages 3428–3435. https://doi.org/10.1093/bioinformatics/btab346.

Heek, J., & Kalchbrenner, N. (2019). Bayesian Inference for Large Scale Image Classification. *http://arxiv.org/abs/1908.03491*

Ma, Nan, Chen, Liang, Hu, Jian, Perdikaris, Paris, & Braham, William W. Adaptive behavior and different thermal experiences of real people: A Bayesian neural network approach to thermal preference prediction and classification. United Kingdom. https://doi.org/10.1016/j.buildenv.2021.107875.

Radev, S. T., Graw, F., Chen, S., Mutters, N. T., Eichel, V. M., Bärnighausen, T., & Köthe, U. (2021). Outbreak Flow: Model-based Bayesian inference of disease outbreak dynamics with invertible neural networks and its application to the COVID-19 pandemics in Germany, *PLoS Comput Biol.* 2021 Oct; 17(10): e1009472, pp. 1–26. https://doi.org/10.1371/journal.pcbi.1009472.

Wenzel, F., Roth, K., Veeling, B. S., ´Swi atkowski, J., Tran, L., Mandt, S., Snoek, J., Salimans, T., Jenatton, R., and Nowozin, S(2020a). How good is the Bayes posterior in deep neural networks really? *In International Conference on Machine Learning.*

Wilson, A. G. and Izmailov, P.(2020) Bayesian deep learning and a probabilistic perspective of generalization. arXiv preprint, *arXiv*:2002.08791. https://doi.org/10.48550/arXiv.2002.08791

W. R. Gilks and P. Wild(1992).Adaptive Rejection Sampling for Gibbs Sampling.Journal of the Royal Statistical Society. Series C (Applied Statistics), Vol. 41, No. 2 (1992), pp. 337-348.

R. Zhang , Li, C., Zhang, J., Chen, C., and Wilson, A. G(2019). Cyclical stochastic gradient MCMC for Bayesian deep learning. arXiv preprint arXiv: 1902.03932. https://doi.org/10.48550/arXiv.1902.03932.

Ogundunmade TP, Adepoju AA (2021). The performance of artificial neural network using heterogeneous transfer functions. *Int J Data Sci Anal*, 2021; 2: 92-103. DOI: 10.18517/ijods.2.2.92-103.

L. Qi, C. Hu, X. Zhang et al., "Privacy-aware data fusion and prediction with spatial-temporal context for smart city industrial environment," IEEE Transactions on Industrial Informatics, vol. 17, no. 6, pp. 4159–4167, 2021.

X. Yang, X. Li, Y. Guan, J. Song, and R. Wang, "Overfitting reduction of pose estimation for deep learning visual odometry," China Communications, vol. 17, no. 6, pp. 196–210, 2020.

K. H. Cha, N. Petrick, A. Pezeshk, C. G. Graff, and B. Sahiner, "Reducing overfitting of a deep learning breast mass detection algorithm in mammography using synthetic images," in Progress in Biomedical Optics and Imaging-Proceedings of SPIE, pp. 188–194, San Diego, CA, USA, March 2019.

A. Ashiquzzaman, A. K. Tushar, M. R. Islam et al., "Reduction of overfitting in diabetes prediction using deep learning neural network," in IT Convergence and Security 2017. Lecture Notes in Electrical Engineering, vol 449, pp. 35–43, Springer, Singapore, 2018.

D. Chen, Z. Yan, and H. Liu, "Overfitting phenomenon of SVD series algorithms in scoring prediction," Journal of Shandong University (Engineering Science Edition), vol. 44, no. 3, pp. 15–21, 2014.

G. González, S. Y. Ash, R. San José Estépar, and G. R. Washko, "Reply to Mummadiet al.: overfitting and use of mismatched cohorts in deep learning models: preventable design limitations," American Journal of Respiratory and Critical Care Medicine, vol. 198, no. 4, pp. 545–555, 2018.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detector," Computer Science, vol. 3, no. 4, pp. 212–223, 2012.

J. Cheng, G. Zeng, D. Lu, and B. Huang, "Dropout-based improved convolutional neural network model averaging method," Journal of Computer Applications, vol. 39, no. 6, pp. 1601–1606, 2019.

J. Li, G. Qin, X. Wen, and F. Hu, “Over-fitting in neural network learning algorithms and its solving strategies,” Journal of Vibration, Measurement and Diagnosis, vol. 22, no. 4, pp. 260–264, 2002.

J. Wang, Z. Wang, and H. Wang, “Improved CNN algorithm based on PSO algorithm and dropout,” Journal of Changchun University of Technology, vol. 40, no. 1, pp. 26–30, 2019.

M. Magris, A. Iosifidis, Bayesian learning for neural networks: an algorithmic survey. Artif Intell Rev (2023). https://doi.org/10.1007/s10462-023-10443-1.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” Journal of Machine Learning Research, vol. 15, no. 1, pp. 1929–1958, 2014.

T. Gong, T. Fan, J. Guo, and Z. Cai, “GPU-based parallel optimization of immune convolutional neural network and embedded system,” Engineering Applications of Artificial Intelligence, vol. 62, no. 25, pp. 384–395, 2017.

Qihan Ren, Huiqi Deng, Yunuo Chen, Siyu Lou, Quanshi Zhang (2023). Bayesian Neural Networks Tend to Ignore Complex and Sensitive Concepts. https://doi.org/10.48550/arXiv.2302.13095.

Qin and Z. Li, “Over-fitting of BP NN research and its application problem,” Engineering Journal of Wuhan University, vol. 39, no. 6, pp. 55-58, 2006.

W. Shen, Y. Li, Z. Yang, X. Wang, and X. Ye, “Attribute reduction to prevent overfitting,” Application Research of Computers, vol. 37, no. 9, pp. 2665–2668, 2020.

Z. Yang, F. Gao, S. Fu, M. Tang, and D. Liu, “Overfitting effect of artificial neural network based nonlinear equalizer: from mathematical origin to transmission evolution,” Science China Information Sciences, vol. 63, no. 6, pp. 82–97, 2020.

Zhang, R., Li, C., Zhang, J., Chen, C., and Wilson, A. G. Cyclical stochastic gradient MCMC for Bayesian deep learning. arXiv preprint arXiv:1902.03932, 2019.