Hydrological Modeling – An Approach with Advanced Neural Network Models and their Sensitivity to Initial Solution

Seema Narain[1], Ashu Jain[2]
[1]*Civil Academic Officer, Department of Civil Engineering, College of Military Engineering, Pune,411001, India*
[2]*Professor, Department of Civil Engineering, Indian Institute of Technology Kanpur, Kanpur-208 016, India*

Abstract- This study introduces a novel approach to hydrological modeling through the application of a Generalized Neuron (GN), an advanced artificial neuron architecture. Unlike the conventional McCulloch and Pitts' artificial neuron (MPAN) commonly utilized in artificial neural networks (ANNs), the GN is structured to handle complex nonlinearities in hydrological systems using a non-linear discriminant function. This compact model architecture eliminates the need for specifying hidden layers and neurons, offering a streamlined modeling process. Two neural system (NS) models were developed: (a) a standard multilayer perceptron (MLP) model trained using the back-propagation (BP) algorithm, and (b) the GN-based model. Both models were tested on rainfall and discharge data from the Kentucky River basin, using ten distinct initial weight configurations to assess sensitivity. Evaluation was carried out using multiple performance indicators. Results show that the GN model not only outperforms the traditional MLP model in terms of accuracy and training efficiency but also demonstrates robustness against initial weight sensitivity. This highlights the potential of GN as a powerful tool for modeling nonlinear hydrological processes.

Keywords- hydrological modeling, ANN, Advanced neural network, water resources engineering, sensitivity of models

I. INTRODUCTION

Hydrological modeling plays a crucial role in the planning, design, and management of water resources systems. Traditionally, hydrologists have relied on two primary modeling approaches: conceptual models that incorporate physical processes, and empirical models that are data-driven and independent of system physics. In recent decades, the use of artificial neural networks (ANNs) for modeling hydrological systems has gained significant traction. These models are capable of capturing nonlinear relationships in rainfall-runoff processes without requiring explicit representation of the watershed's internal mechanisms.

ANN-based modeling efforts in hydrology range from relatively simple applications (Minns and Hall, 1996; Shamseldin et al., 1997; Campolo et al., 1999; Jain and Indurthy, 2003) to more sophisticated integrations involving genetic algorithms (Jain and Srinivasulu, 2004), hybrid frameworks (Chen and Adams, 2006), and data decomposition techniques (Abrahart and See, 2000; Jain and Srinivasulu, 2006). These studies underscore the growing need for robust, flexible, and accurate models capable of delivering reliable flow forecasts. The majority of ANN applications in hydrology utilize the McCulloch and Pitts artificial neuron (MPAN), originally introduced in the 1940s. These conventional networks use a summation-based aggregation function followed by a nonlinear transformation. However, this design has several limitations: extended training time (Shamseldin, 1997), complexity in determining the optimal network architecture (Hsu et al., 1995), linear aggregation of inputs, and susceptibility of the back-propagation (BP) training algorithm to local minima and flow bias (Rumelhart et al., 1986).

To address these challenges, several modifications to the MPAN architecture have been proposed. Innovations include time-delay differential equations in neural models (Chunguang et al., 2004), dendritic

computation structures, and spiking neural networks (Cios et al., 2004; Prete et al., 2004). The present study builds on this trajectory by exploring the performance of a novel artificial neuron, the Generalized Neuron (GN), which incorporates nonlinear discriminant functions (Chaturvedi et al., 1999; Yadav et al., 2006).

Using rainfall and streamflow data from the Kentucky River Basin, this research compares a GN-based neural system with a traditional multi-layer perceptron (MLP) model trained via BP. Both models are evaluated across multiple error metrics, with a particular focus on sensitivity to initial weights—an important factor affecting model reliability. The findings demonstrate that the GN model offers significant improvements in performance and robustness over conventional ANN architectures.

II. STUDY AREA AND DATA

The models developed in this research were trained and validated using historical data from the Kentucky River Basin, a significant watershed located in the state of Kentucky, USA. This basin spans over 17,820 square kilometers (approximately 4.4 million acres) and includes all or parts of 40 counties. The Kentucky River serves as the primary source of drinking water for numerous municipalities within the state.

Model development focused on the section of the river upstream of Lock and Dam 10 (LD10), near Winchester, Kentucky, which drains an area of around 10,240 km². The watershed has an estimated time of concentration of two days, reflecting the time required for water to travel from the furthest point in the basin to the outlet.

The dataset used for this study comprises daily average streamflow measurements (in m³/s) from two gauging stations: LD10 and LD11 (near Heidelberg), as well as daily rainfall data (in mm) from five strategically placed rain gauges: Manchester, Hyden, Jackson, Heidelberg, and Lexington Airport. The available dataset spans a period of 26 years (1960–1989), although some gaps are present in the record.

To enable robust model training and validation, the data were divided into two subsets. The training dataset includes 13 years of daily rainfall and flow observations from 1960 to 1972. The testing dataset comprises data from 1977 to 1989. This temporal separation helps in objectively evaluating model generalizability and predictive capability under different hydrological conditions.

III. MODEL PERFORMANCE STATISTICS

To objectively assess the predictive accuracy of the developed neural models, four widely used statistical performance indicators were employed:

1. Nash–Sutcliffe Efficiency (E)
2. Pearson's Correlation Coefficient (R)
3. Average Absolute Relative Error (AARE)
4. Threshold Statistics (TS)

The equations to compute these statistics are provided below.

$$E = 1 - \frac{\sum (XE - XO)^2}{\sum (XO - \overline{XO})^2} \quad (1)$$

$$R = \frac{\sum (XO - \overline{XO})(XE - \overline{XE})}{\sqrt{\sum (XO - \overline{XO})^2 \sum (XE - \overline{XE})^2}} \quad (2)$$

$$AARE = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{XO(t) - XE(t)}{XO(t)} \right| * 100\% \quad (3)$$

$$TS_x = \frac{n_x}{N} \times 100\% \quad (4)$$

Where XO is the observed value of the variable, XE is the estimated value of the variable from a model, $\overline{XO}$ is the average observed value of the variable, $\overline{XE}$ is the average estimated value of the variable, n_x is the number of data points estimated for which the absolute relative error (ARE) is less than x%, N is the total number of data points estimated, and all the summations run from 1 to N. The value of x of 1%, 25%, 50%, and, 100% were considered in this study to compute threshold statistics. Values of AARE close to 0.0 represent good model performance. The distribution of errors is very well represented by the threshold statistics. The TS_x may be defined as the percentage of data points forecasted for which the

absolute relative error is less than x%, therefore higher value of threshold statistics indicates the better performance of the model. The TS can range between 0% and 100% with higher values representing good model performance. Coefficient of correlation can range between -1.0 and +1.0 with magnitudes close to 1.0 meaning good linear dependence between observed and modeled outputs. The Values of Nash efficiency can range between $-\infty$ and +1.0 with values close to 1.0 being very good. The values of E equal to 0.0 means the model is as good as the mean values. Together, these statistics provide a comprehensive evaluation of model fidelity across multiple dimensions — overall fit, linear correlation, absolute error, and error distribution.

IV. MODEL DEVELOPMENT

Two types of neural system (NS) models were constructed for this study: a traditional feed-forward multilayer perceptron (MLP) model and a model based on the Generalized Neuron (GN) architecture.

MLP Model

The MLP model was developed using the conventional McCulloch and Pitts Artificial Neuron (MPAN) as the fundamental processing unit. The architecture consists of an input layer, a hidden layer, and an output layer (see Figure 2). Training was carried out using the back-propagation (BP) algorithm, enhanced with a momentum factor to improve convergence stability.

To identify the optimal number of hidden neurons, various MLP configurations (ranging from 1 to 20 neurons in the hidden layer) were tested using a trial-and-error approach. Performance was evaluated based on multiple error metrics. The optimal MLP architecture was found to be 5-4-1, indicating five input nodes, four hidden neurons, and one output node.

Model parameters — the learning rate (η) and momentum correction factor (α) were also tuned using iterative experimentation.

GN Model

In contrast to the MLP, the GN model utilizes a more compact structure by integrating both linear and nonlinear discriminant functions within a single processing unit. It includes five components: linear and nonlinear weighted input aggregators, their respective activation functions (sigmoid in this study), and an assimilation function that combines the outputs. The GN model eliminates the need to define hidden layers or multiple neurons, simplifying the model design.

Both the MLP and GN models used identical input variables, selected based on cross-correlation and partial correlation analyses. These inputs include:

- Rainfall values at current and previous time steps (P_t, P_{t-1}, P_{t-2})
- Historical streamflow at LD10 ($Q10_{t-1}$, $Q10_{t-2}$)

The output variable modeled by both systems is the streamflow at LD10 on day t ($Q10_t$).

The statistical characteristics of the input data sets used for model training and testing are summarized in Table 1.

Model training for the GN was also based on the gradient descent method, with the key difference being that the GN model requires significantly fewer weights ($2n + 3$ for n inputs), which reduces both the computational load and the required training data volume.

A. Generalized Neuron Model

Traditional artificial neurons typically aggregate weighted inputs through summation and then apply a nonlinear activation function. While effective in many applications, this design becomes computationally intensive and structurally complex when modeling intricate, nonlinear systems—particularly due to the need for multiple hidden layers and extensive training data (Chunguang et al., 2004; Cios et al., 2004; Prete et al., 2004).

To overcome these limitations, this study employs a Generalized Neuron (GN) model, which offers a simplified yet highly expressive alternative to conventional neural structures. The GN is constructed with five key components:

1. Linear Discriminant Function ($\sum$) – aggregates weighted inputs similar to traditional neurons.
2. Nonlinear Discriminant Function (Π) – multiplies inputs, enabling nonlinear transformations.
3. Two Independent Activation Functions – process outputs of the discriminant functions; in this study, sigmoid functions were used for both.

4. Assimilation Function – combines outputs from the linear and nonlinear pathways to generate the final response.
5. Bias Terms – added to both discriminant components, as in standard ANN designs.

The weighted inputs to the GN are represented as follows:

$$NetL_j = \sum WL_{ij} X_i + s\_bias \quad (5)$$

$$NetNL_j = \prod WNL_{ij} X_i \times pi\_bias \quad (6)$$

Where, X_i's are the inputs. For the (Σ_1 and $\int$) part of the GN model, $NetL_{ij}$ is the net input to the GN, WL_{ij}'s are the weights and *s_bias* is the bias weight. $NetNL_{ij}$ is the net input to the GN, WNL_{ij}'s are the weights and *pi_bias* is the bias weight for the part (Π_1 and $\int$) of the GN model. The output from the discriminant portions (Σ_1 and Π_1) is calculated using the respective activation functions ($\int$), which can be either a sigmoid, a Gaussian, a Spline, or any other mathematical function satisfying the conditions of being an activation function in the traditional ANNs employing MPANs. In this study, sigmoid activation functions were used to calculate the outputs from the components of the GN model.

$$y_1 = \frac{1}{(1+e^{-NetL_j})} \quad (7)$$

$$y_2 = \frac{1}{(1+e^{-NetNL_j})} \quad (8)$$

The assimilation function Σ was a linear combination of the two outputs. The equation to calculate the overall output from the GN model can be represented as follows:

$$O_{pk} = w\, y_1 + (1-w)\, y_2 \quad (9)$$

Where O_{pk} is the overall output from the GN model; w is the weight corresponding to the linear discriminant function's output y_1; and y_2 is the non- linear discriminant function's output from the GN model. The training of the GN model is carried out in a similar fashion as a traditional ANN using gradient descent method. The optimized value of parameters, learning rate (η) and momentum correction factor (α) were found by trial and error method. The weight parameter w is also optimized during training so that the GN model consists of a total of $2n+3$ weight parameters for n inputs. The overall structure of the GN model described above provides a very compact ANN model as compared to the traditional ANN model having many times more weights due to the number of hidden neurons involved in them. The details of training of a GN model are not included here and can be found in (Chaturvedi et al., 1999 and Chaturvedi et al., 2004). The generalized neural network has characteristics of both simple and high order neurons. The non-linearity present in the system is incorporated with suitable discriminant and activation functions in GN model. The proposed model has both linear and non-linear discriminant functions associated with sigmoid activation function with weight sharing. The number of weights in the case of a GN model is equal to twice the number of inputs plus two, which is very low in comparison to a multi-layer feed-forward ANN. The weights are determined through training. Hence, by reducing the number of unknown weights, training time as well as the minimum number of patterns required for training can be reduced. Also, there is no need to select the number of hidden layers and the number of hidden neurons. This reduces the complexity and dimensionality of the overall ANN model. A schematic of the GN model is presented in Figure 3. The GN model receives inputs from external source and gives output to the external source like in an ANN model built using MPANs. The GN model differs in its structure that is actually responsible for capturing the complex input-output relationships.

V. RESULTS AND DISCUSSIONS

The comparative performance of the MLP and GN models was evaluated using standard error metrics, with detailed results presented in Tables 2A and 2B. Both models demonstrated strong predictive capabilities, with Nash-Sutcliffe efficiency (E) and Pearson correlation coefficient (R) values exceeding 0.90 during both training and testing phases, indicating excellent agreement between observed and predicted flows.

During training, the GN model achieved a lower Average Absolute Relative Error (AARE) than the MLP model, reflecting improved accuracy. Testing

results showed comparable AARE values for both models, though GN maintained a slight edge in overall predictive consistency. In terms of the Threshold Statistics (TS), the GN model outperformed the MLP across all thresholds, particularly at higher thresholds (e.g., TS50 and TS100), where a larger percentage of predictions fell within acceptable error margins.

Training time also significantly favored the GN model. As shown in Table 2A, the GN architecture required substantially less computational time to converge compared to the MLP, which underscores its efficiency and suitability for practical applications. Despite having only a single neuron, the GN model delivered performance comparable to or better than a multi-layer network, highlighting its structural compactness and efficiency.

Visual comparisons of model outputs against observed streamflows are presented in Figure 4. The scatter plots clearly show that the MLP struggled to predict higher-magnitude flow events accurately, resulting in a broader scatter around the 1:1 line. In contrast, the GN model demonstrated a tighter clustering of predicted values along the ideal line, especially in the higher flow range—further supporting its superior modeling capabilities.

In summary, the GN model delivered more accurate and consistent results, particularly in terms of training efficiency and its ability to capture nonlinear hydrological behavior. These findings suggest that GN offers a valuable alternative to traditional ANN architectures, especially when simplicity, speed, and reliability are critical.

VI. SENSITIVITY ANALYSIS

Multilayer perceptron (MLP) models, though widely used, are often criticized for their sensitivity to initial weight settings. This sensitivity stems from the complex error surfaces created by the multiple hidden layers and neurons, which result in numerous local optima. The back-propagation (BP) algorithm, typically used for training, can become trapped in these local minima, leading to inconsistent model performance.

To assess this issue, a sensitivity analysis was conducted for both the MLP and Generalized Neuron (GN) models. Each model was retrained ten times using different randomly initialized weight vectors. For each run, key performance metrics—Nash-Sutcliffe efficiency (E), correlation coefficient (R), and average absolute relative error (AARE)—were computed. The results are summarized in Tables 3A and 3B.

The GN model consistently produced better average performance across all metrics compared to the MLP model, during both training and testing phases. Notably, the standard deviations of the performance indicators for the GN model were significantly lower. This indicates that the GN model's outcomes were more stable and less affected by the initial weight configuration.

For the MLP model, larger standard deviations were observed across all metrics, reflecting high variability in performance depending on initial weights. This variability confirms the model's susceptibility to getting stuck in suboptimal solutions due to its complex architecture.

The GN model, by contrast, showed minimal variation across different runs, especially in AARE and threshold statistics (TS), where standard deviations were close to zero in many cases. This stability suggests that the GN model is more robust and reliable for practical applications, as it reduces the uncertainty associated with weight initialization and the risk of suboptimal convergence.

In essence, the sensitivity analysis underscores a key advantage of the GN architecture: its reduced dependence on initial conditions, which enhances model reliability and simplifies the training process. This makes it particularly suitable for modeling complex physical systems where data availability and computational efficiency are critical concerns.

VII. SUMMARY AND CONCLUSIONS

This study introduced and evaluated the Generalized Neuron (GN) model as an alternative to conventional multilayer perceptron (MLP) architectures for simulating the complex and nonlinear rainfall-runoff processes inherent in hydrological systems. Using daily

rainfall and streamflow data from the Kentucky River Basin, both GN and MLP models were developed and rigorously tested. The GN model demonstrated several advantages over the MLP framework. It produced higher or comparable accuracy across standard performance metrics, required significantly less training time, and achieved greater robustness to initial weight conditions. Unlike the MLP, which involves selecting the number of hidden layers and neurons—a process typically guided by trial and error—the GN model's compact structure simplifies architecture design and reduces computational effort. Furthermore, the GN model showed remarkable insensitivity to the choice of initial weights. Sensitivity analysis confirmed that the GN model's performance remained stable across different initialization scenarios, a critical strength when modeling nonlinear systems where traditional ANNs may yield widely varying outcomes due to local minima in the error surface.

These findings underscore the GN model's potential as a reliable and efficient tool for hydrological modeling. Its compactness, reduced training requirements, and consistent performance make it a strong candidate for broader application in water resource studies. The results also suggest that continued exploration of novel neuron architectures, such as the GN, could lead to substantial improvements in the modeling of complex physical systems.

Future work should aim to validate the GN model across additional watersheds and under varying climatic and hydrological conditions. Additionally, integrating the GN model with advanced training algorithms beyond standard back-propagation may further enhance its performance and applicability in real-world hydrological forecasting and management scenarios.

REFERENCE

[1] Abrahart, R.J. and See, L. 2000. Comparing Neural Network and Autoregressive Moving Average Techniques for the Provision Of Continuous River Flow Forecasts in Two Contrasting Watersheds, *Hydrol. Processes*, 14, 2157-2172.

[2] Campolo, M., Andreussi, P., and Soldati, A. 1999. River Flood Forecasting with Neural Network Model. *Water Resour. Res.,* 35(4), 1191-1197.

[3] Chaturvedi, D.K., Malik, O.P., and Kalra, P.K. 2004, Generalised Neuron-Based Adaptive Power System Stabilizer, *IEEE Proc-Genetation, Transmission and Distribution ,* 151(2), 213-218.

[4] Chaturvedi, D.K., Satsangi, P.S., and Kalra, P.K. 1999. New Neuron Models for Simulating Rotating Electrical Machines and Load Forecasting Problems, Elec. *Power Sys. Res*., 52, 123-131.

[5] Chen, J. and Adams, B.J. 2006. Integration Of Artificial Neural Networks with Conceptual Models in Rainfall-Runoff Modeling, *J. Hydrol*., 318, 232-249.

[6] Chunguang, L.,a, Guangrong, C., Xiaofeng L., and Juebang Y. 2004, Hopf Bifurcation and Chaos in a Single Inertial Neuron Model with Time Delay, *The European Physical Journal*, 41, 337–343.

[7] Cios, K.J., Swiercz, W, Jackson, W. 2004, Networks of Spiking Neurons In Modeling and Problem Solving, *Neurocomputing* 61, 99 – 119.

[8] Hsu, K., Gupta., H.V., and Sorooshian, S. 1995, Artificial Neural Network Modeling of the Rainfall-Runoff Process, *Water Resources Research*, 31(10), 2517-2530.

[9] Jain, A. and Srinivasulu, S. 2004. Development of Effective And Efficient Rainfall-Runoff Models Using Integration of Deterministic, Real-Coded Genetic Algorithms, and Artificial Neural Network Techniques, *Water Resour. Res*., 40(4), W04302, doi:10.1029/2003WR002355.

[10] Jain, A. and Srinivasulu, S. 2006. Integrated Approach to Modelling Decomposed Flow Hydrograph using Artificial Neural Network and Conceptual Techniques, *J. Hydrol*., 317(3-4), 291-306.

[11] Jain, A., and Indurthy, S.K.V.P. 2003. Comparative Analysis of Event Based Rainfall-Runoff Modeling Techniques-Deterministic, Statistical, And Artificial Neural Networks, *J. Hydrol. Engg*., ASCE, 8(2), 93-98.

[12] Minns, A.W. and Hall, M.J. 1996. Artificial Neural Networks as Rainfall Runoff Models, *Hydrol. Sci. J*., 41(3), 399-417.

[13] Prete, V.D., and Coolen, A.C.C. 2004, Non-Equilibrium Statistical Mechanics of Recurrent Networks with Realistic Neurons, *Neurocomputing*, 58-60, 239-244.

[14] Rumelhart, D.E., Hinton, G.E. and Williams, R. J. 1986. Learning Representations by Back-Propagating Errors, *Nature*, 323, 533-536.
[15] Shamseldin, A. Y., 1997. Application of a Neural Network Technique to Rainfall-Runoff Modeling, *J. Hydrol*., 199, 272-294.
[16] Yadav, R.N., Kalra, P.K., and John, J. 2006. Neural Network Learning with Generalized Mean Based Neuron Model, *Soft Comput*., 10, 257-263.

TABLE 1: Kentucky River Data Statistics

Model Inputs	Rainfall (mm)	Flow (m^3/s)
Average		
Training	3.24	149.94
Testing	3.16	144.78
Maximum		
Training	77.40	2528.69
Testing	81.99	2806.19
Minimum		
Training	0.00	3.60
Testing	0.00	3.28
Std. Deviation		
Training	6.20	243.30
Testing	6.18	232.81
Skewness		
Training	3.49	3.67
Testing	3.94	4.00

Table 2A: Error Statistics of NS Models

Model	E	R	AARE	Execution Time
During Training				
MLP	0.904	0.954	32.8	00:16:55.21
GN	0.920	0.959	29.7	00:01:52.68
During Testing				
MLP	0.823	0.948	27.7	
GN	0.916	0.957	29.8	

Table 2B: Threshold Error Statistics of NS Models

Model	TS1	TS25	TS50	TS100
During Training				
MLP	1.9	38.4	52.4	64.7
GN	1.8	46.1	67.6	82.5
During Testing				
MLP	1.2	36.4	63.9	74.7
GNC	1.9	46.9	67.5	81.2

Table 3A: Sensitivity Analyses Results with respect to Initial Solutions

Model	E	R	AARE
Average During Training			
MLP	0.905	0.952	28.5
GN	0.921	0.960	29.8
Average During Testing			
MLP	0.895	0.946	28.3
GN	0.915	0.957	29.8
Standard Deviation During Training			
MLP	0.025	0.013	6.016
GN	0.001	0.001	0.295
Standard Deviation During Testing			
MLP	0.022	0.012	6.548
GN	0.002	0.001	0.345

Table 3B: Sensitivity Analyses Results with respect to Initial Solutions

Model	TS1	TS25	TS50	TS100
Average During Training				
MLP	2.0	43.5	64.0	75.4
GN	1.8	48.2	70.9	84.2
Average During Testing				
MLP	2.0	45.8	64.6	75.3
GN	1.9	48.7	70.3	83.5
Standard Deviation During Training				
MLP	0.98	10.40	9.55	10.87
GN	0.08	1.49	2.23	0.74
Standard Deviation During Testing				
MLP	0.73	9.07	7.79	9.45
GN	0.10	1.61	1.90	1.37

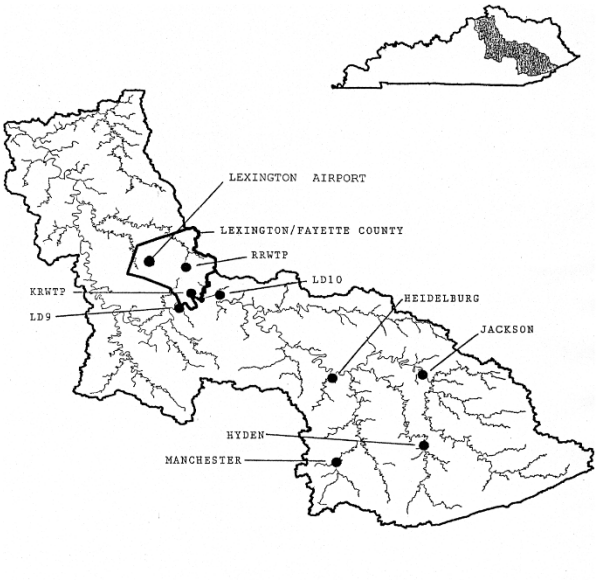


Figure 1. Kentucky River Basin

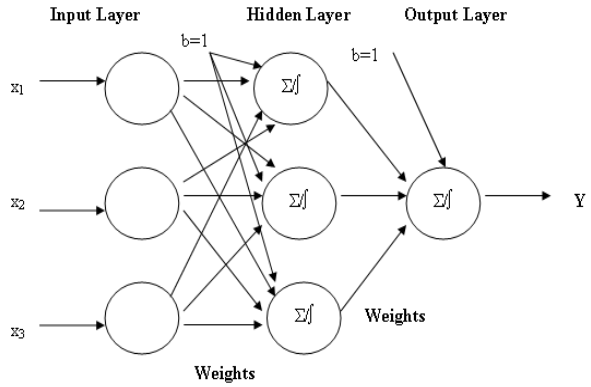


Figure 2: MLP Model

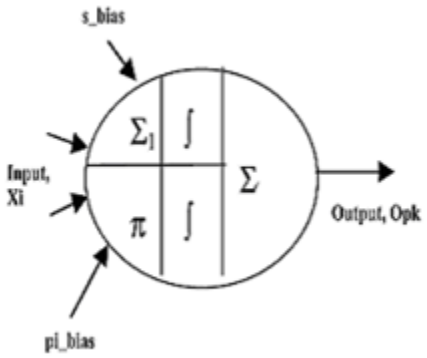
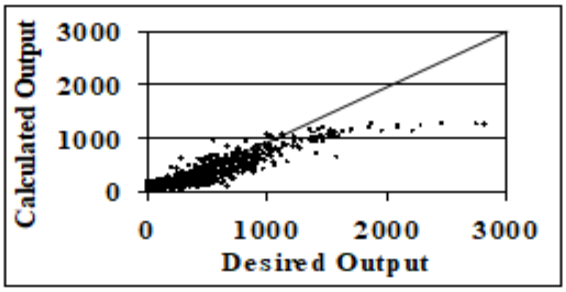
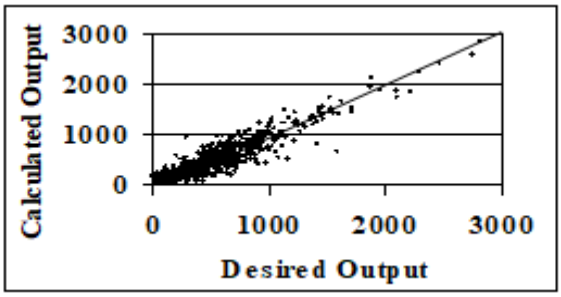


Figure 3: Generalized Neuron Model



(a) MLP Model



(b) GN Model

Figure 4: Scatter Plot of Neural System Models during Testing