A Comparative Study of Bayesian Regularization Neural Network and Multilayer Perceptron Neural Network in the Hybridization of Self-Exciting Threshold Autoregressive Models

Jayson N. Payla[1*] and Bernadette F. Tubo[2]
[1]Department of Applied Mathematics
University of Science and Technology of Southern Philippines – Cagayan de Oro
Cagayan de Oro City, 9000, Philippines
**jaysonpayla7@gmail.com*

[2]Department of Mathematics and Statistics
Mindanao State University Iligan Institute of Technology
Iligan City, 9200, Philippines

Date received: August 28, 2025
Revision accepted: December 29, 2025

Abstract

Artificial Neural Networks (ANNs), particularly Bayesian Regularization Neural Networks (BRNNs), have demonstrated strong capabilities for modeling complex nonlinear patterns in time-series data. This study explores the hybridization of Self-Exciting Threshold Autoregressive (SETAR) models with BRNN and Multilayer Perceptron Neural Networks (MLPNNs) to improve forecasting accuracy of nonlinear cyclical data, using the Canadian Lynx dataset. The hybrid models aim to capture regime shifts and nonlinear dynamics more effectively. Results show that the SETAR-BRNN hybrid outperforms individual models and other hybrids, achieving a Mean Absolute Percentage Error (MAPE) of 2.261%, representing a 45.8% reduction compared to the standalone SETAR model (MAPE = 4.17%) and a 14.7% reduction compared to BRNN alone (MAPE = 2.65%). Additionally, the SETAR-BRNN model reduces the Root Mean Square Error (RMSE) by 65.3% relative to the MLPNN-SETAR hybrid and suggest a forecasting performance across multiple horizons. The findings indicate that integrating BRNN into the SETAR framework significantly enhances predictive accuracy and model robustness for nonlinear, regime-dependent time series. This highlights the effectiveness of the SETAR-BRNN hybrid approach in accurately modeling complex cyclical behaviors and regime shifts in real-world data.

***Keywords**:* bayesian regularization, hybrid models, multilayer perceptron, neural networks, SETAR

1. Introduction

1.1 Time-series Forecasting

Forecasting plays a vital role across fields such as economics, weather prediction, signal processing, and astronomy by enabling the estimation of future events from past observations. The central idea in time series forecasting is to analyze historical data organized in time sequence to predict future values (Alsuwaylimi, 2023). Traditional time series forecasting methods, such as the Autoregressive Integrated Moving Average (ARIMA) model, are widely used because they can model linear, stationary data (Melina *et al.*, 2024). However, ARIMA has limitations in modeling nonlinear patterns commonly found in real-world data (Rahayu *et al*., 2023).

To address the shortcomings of linear forecasting models, researchers have introduced nonlinear time-series approaches, such as the Self-Exciting Threshold Autoregressive (SETAR) model. The Self-Exciting Threshold Autoregressive (SETAR) model is a nonlinear time-series modeling technique that partitions the data into different regimes or states based on the value of a threshold variable. In each regime, the data are modeled using separate linear autoregressive (AR) processes, allowing the model to capture different behaviors or patterns within the series (Polestico *et al.*, 2024). Still, SETAR may fall short when dealing with highly nonlinear or nonstationary data that do not align cleanly with predefined threshold regions.

1.2 Artificial Neural Networks

Artificial Neural Networks (ANNs), specifically Bayesian Regularization Neural Networks (BRNNs), are an advanced form of artificial neural networks designed to improve performance and generalization during training. Unlike traditional back-propagation neural networks that solely rely on minimizing error, BRNN incorporates network weights into a training objective function with regularization parameters. These parameters balance the network's fit to the data against the model's complexity to prevent overfitting (Dalipi & Yildirim, 2015). However, BRNNs alone may struggle to accurately model structured linear or threshold-driven components of a time series, especially in scenarios with limited training data or multistep forecasting needs. The Multilayer Perceptron Neural Network (MLPNN) as a model that consists of an input layer, one or more hidden layers, and an output layer. It is a popular type of feed-forward neural network that utilizes the back-propagation

algorithm for training. The architecture of an MLPNN, including the number of layers and neurons, critically affects its ability to solve problems effectively, with the optimal architecture being crucial for achieving good performance without overfitting or underfitting (Ramchoun *et al.*, 2016).

1.3 Hybrid Models

In recent research, a hybrid forecasting approach that combines linear models with neural networks has shown promising results for handling complex data patterns. Atesongun and Gulsen (2024) emphasize that integrating ARIMA and ANN models in a sequential hybrid framework can significantly enhance forecasting performance, especially for complex datasets exhibiting multiple patterns. Faruk (2010) demonstrated that a hybrid ARIMA-ANN model is more effective in predicting water quality parameters, such as dissolved oxygen and chemical oxygen demand, than when using ARIMA or neural networks independently. Alsuwaylimi (2023) emphasized that hybrid models combining ARIMA and ANN techniques outperform individual models in forecasting accuracy for monthly gold prices. Melina *et al.* (2024) emphasized the importance of thorough model evaluation and the potential advantages of nonlinear approaches in financial time series forecasting. While Velasco *et al.* (2019) found hybrid models to be promising tools in time series forecasting, careful selection and validation are necessary to realize their full potential, and hybridization should be approached with consideration of the context and data characteristics.

This study developed the additive hybridization of Self-Exciting Threshold Autoregressive (SETAR) models with Bayesian Regularization Neural Networks (BRNN) and Multilayer Perceptron Neural Networks (MLPNN). The SETAR component is employed to model threshold-driven regime changes, while BRNN and MLPNN are utilized to model the nonlinear structure in the residuals left unexplained by the SETAR process. Despite ongoing advances, the use of SETAR-based hybrid models, particularly those incorporating BRNN and MLPNN, remains relatively unexplored.

The proposed SETAR-BRNN and SETAR-MLPNN hybrid models were evaluated using the Canadian Lynx dataset, which exhibits nonlinear cyclical and regime-switching behavior. Although the present study focuses exclusively on this benchmark dataset, the proposed methodology may have potential applications in other nonlinear time series domains, such as finance, energy demand, tourism, traffic flow, and environmental studies. Future

research should validate the proposed models using datasets from these application areas before broader generalization can be made. These domains commonly exhibit recurring structures and abrupt changes, making the SETAR–BRNN and SETAR-MLPNN hybrid a robust tool for improving predictive accuracy in real-world forecasting scenarios.

The specific objectives of this study are to: to present a comprehensive overview of the structure and modeling process for Bayesian Regularization Neural Network (1); to investigate the additive hybridization of SETAR with both BRNN and MLPNN frameworks (2); to assess the predictive performance of these hybrid models using Canadian Lynx time series (3); and to compare the forecasting effectiveness between SETAR-BRNN and SETAR-MLPNN hybrid configurations (4). The models will be applied to the well-known Canadian Lynx dataset (Kajintani *et al*., 2005), which is characterized by pronounced nonlinearity and cyclical trends.

2. Methodology

2.1 Time-series Model

The Autoregressive Integrated Moving Average (ARIMA) model, introduced by Box and Jenkins (1976), has become one of the most widely used techniques for time series forecasting. This model combines three components: autoregression (AR) of order p, moving average (MA) of order q, and differencing of order d to address nonstationary resulting in the general form ARIMA (p, d, q). In this framework, future values are modeled as linear combinations of previous observations and random disturbances. Selecting appropriate values for p, d, and q involves analyzing the sample autocorrelation function (ACF) and partial autocorrelation function (PACF) and aligning them with their theoretical counterparts (Khashei & Bijari, 2010). The given steps will guide in finding p, d, and q:

Step 1. Make a scatter plot for Z_t versus t to determine constant mean and variance;

Step 2. If not stationary, difference to remove trend, find d. Plot sample ACF and sample PACF to identify potential *AR* and *MA* model;

Step 3. Estimate the parameters of the model and determine if they are significant;

Step 4. Get the residuals of the fitted model;

Step 5. Plot sample ACF and sample PACF of the residuals and determine if it is a white noise;

Step 6. If not, go back to step 2 and find an appropriate model.

2.2 Self-exciting Threshold Autoregressive Model

The Self-Exciting Threshold Autoregressive (SETAR) model is characterized by linear autoregressive behavior within each regime but becomes nonlinear when there are at least two distinct regimes, each governed by a different linear equation. This model is capable of capturing complex dynamics such as limit cycles, amplitude-dependent frequencies, and abrupt change features that linear time series models cannot adequately describe (Asikgil, 2018).

Gonzalo and Pitarakis (2002) examined parameter estimation in both single- and multiple-threshold models using joint sequential methods. The number of regimes was initially assumed to be known in advance; a model-selection-based framework was later introduced, allowing simultaneous estimation of both the model parameters and the number of regimes. By establishing the asymptotic properties of the sequential estimation method, the authors demonstrated that models with multiple thresholds could be estimated. The study further showed that estimating threshold parameters sequentially could yield consistent estimators of the true values (Gonzalo & Pitarakis, 2002).

Following the procedure outlined by Gonzalo and Pitarakis (2002), the model determines the lag order (p) and the delay parameter (d), and identifies the threshold value. This is achieved through an exhaustive search and model selection based on minimizing the Akaike Information Criterion (AIC). The process for constructing a conditional nonlinear mean model for the dependent variable Y_t involves the following steps shown in Figure 1:

Steps for building a conditional nonlinear mean model for the dependent variable Y_t are as follows:

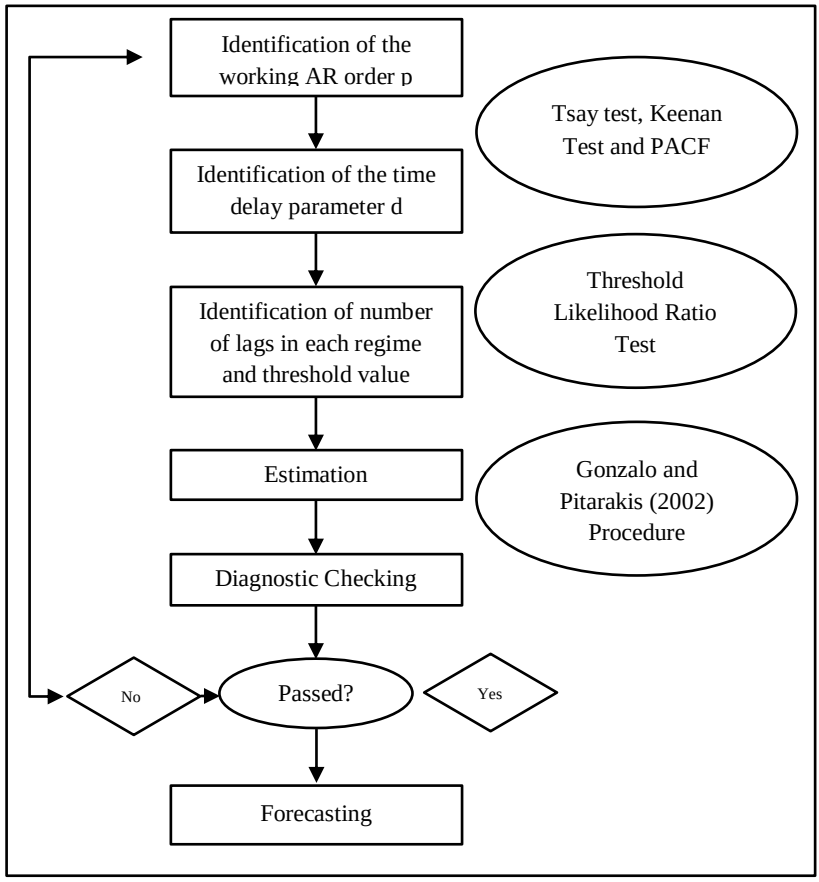

Figure 1. Iterative SETAR modeling procedure

Step 1. Tentatively choose the autoregressive (*AR*) order. This involves selecting a maximum lag length L to be considered in the two-piecewise linear AR model. This initial choice can be guided by the Partial Autocorrelation Function (PACF), Akaike Information Criterion (AIC), or results from the Tsay test for detecting nonlinearity.

Step 2. Determine the number of regimes. The suitable number of regimes can be identified using Hansen's test for nonlinearity.

Step 3. Identify the delay parameter d and the corresponding threshold variable X_{t-d}. Let $D = \{1, 2, \ldots, L\}$ represent the set of potential delay lags. For each $d \in D$, fit autoregressive models and conduct a likelihood ratio test to

detect threshold nonlinearity. The delay that produces the most statistically significant result is a good candidate to begin modeling.

Step 4. Refine the model. After identifying a preliminary structure, further adjustments may be made to improve model performance and accuracy.

2.3 Multilayer Perceptron Neural Network Model

A Multilayer Perceptron Neural Networks (MLPNN) consists of multiple layers of interconnected nodes, with each node performing a specific mathematical function known as a perceptron. The structure begins with an input layer that receives external data, followed by one or more hidden layers that process this information, and ends with an output layer that produces the final result. These hidden layers serve as intermediaries between the input and output layers, enabling the network to learn complex patterns (Miller *et al*., 2018).

The architecture of an Artificial Neural Network (ANN) often relies on a single hidden layer MLPNN, which operates under a supervised learning framework. This configuration features three main components: an input layer, a hidden layer, and an output layer. These layers consist of basic processing units connected through directed, non-cyclic pathways (Ramchoun *et al*., 2016). The relationship between the predicted output (Z_t) and input sequence (Z_{t-1}, Z_{t-2}, …, Z_{t-d}) can be expressed using the following mathematical formulation (Equation 1):

$$Z_t = f\left[\varphi_o + \sum_{j=1}^{c} \varphi_j f(\omega_{oj} + \sum_{i=1}^{d} \omega_{ij} Z_{t-i})\right] + \epsilon_t \quad , \qquad (1)$$

where φ_j, $j = 1,2,3,\ldots,c$ and $\omega_{ij} = 1,2,3,\ldots,d$; $j = 1,2,\ldots,c$ represent the connection weights, including bias weights denoted by φ_o and $\omega_{oj}'s$, d refers to the number of input neurons, while c corresponds to the number of hidden neurons. The function f serves as the activation function used in the hidden layer.

A key element of the MLPNN is the bias node, which functions similarly to the hidden neurons. These bias nodes enable the network to model data patterns more effectively and should not be confused with the statistical concept of bias. The sigmoid function, defined as $f(x) = \frac{1}{1+e^{-x}}$ is commonly used as the activation function in hidden layers due to its smooth and bounded output properties (Alpaslan *et al*, 2012).

Once the architecture of the MLPNN model is established, the network must undergo a training process. This involves adjusting the weights and biases to minimize the error between predicted and actual outputs for each training sample, ultimately leading to improved prediction performance. The process of adjusting weights will continue until error function is less than the desired limit. However, in this study the researcher employs different ways of data training. Instead of setting an error value where the training will stop if the error function is less than the desired said error value, the researcher set a fixed 1000 iteration where one iteration with the smallest network error after 1000 iterations will be chosen as the neural network model to be used in testing set. The goal of training is to find a minimum point in the error surface which corresponds to the weights that result to the least forecasting error given that the hyperparameters are set. In every run of the algorithm up to 1000 epochs and submitting all the training patterns into the network in each epoch, the algorithm takes a route or path in the error surface to reach the minimum (Miller *et al*., 2018).

Backpropagation relies on two key parameters alongside gradient descent. The first is the learning rate, which determines the extent to which the gradient influences the weight updates. Essentially, the gradient is scaled by the learning rate and applied to adjust the weight matrix incrementally, guiding the model toward reducing the overall error. Among the candidate models, the one that achieves the lowest Mean Squared Error (MSE) during training will be selected as the optimal neural network model. This chosen model will then be validated using a hold-out sample to assess its forecasting capability (Ramchoun *et al*., 2016). Forecasting will be performed using a one-step-ahead approach, where actual lagged input values $Z_{t-1}, Z_{t-2}, \dots, Z_{t-p}$ are used to predict the next value Z_t. The predictive performance of the model will be evaluated using two common metrics: Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE).

The following steps for the Multilayer Perceptron Neural Network:

Step 1. Choose random initial values for the weights. Say $w = (w_1, w_2, \dots, w_N)^T$;

Step 2. All n training input-output patterns are submitted to the network one by one;

Step 3. Calculate E and the gradient of E, ∇E, with respect to each weight;

Step 4. Update the weights such that $\Delta w(t+1) = -\eta \nabla E + \mu \Delta w(t)$, and $w(t+1) = w(t) + \Delta w(t+1)$, where η is called the learning rate and μ is the momentum;

Step 5. Repeat steps 2, 3, and 4 until 1000 epoch. Let $\nabla E(w)$ be the gradient of function E is the vector of the first partial derivatives with respect to each weight in the neural network. $\nabla E(w) = \left(\frac{\delta E}{\delta w_1}, \frac{\delta E}{\delta w_2}, \ldots, \frac{\delta E}{\delta w_N}\right)^T$, where N is the total number of weights.

2.4 Bayesian Regularization Neural Network Model

Bayesian regularization as an effective approach to mitigating overfitting by incorporating prior distributions on the network weights, which naturally penalizes overly complex models and promotes smoother, more generalizable solutions. Unlike dropout, which randomly deactivates neurons during training to prevent co-adaptation, or weight decay, which applies a fixed penalty to large weights, Bayesian regularization employs a probabilistic framework that adaptively determines the optimal regularization parameters based on the data itself. This leads to a more nuanced control over model complexity and tends to yield more robust and consistent performance across multiple training runs, as it effectively balances bias and variance without requiring extensive hyperparameter tuning. Consequently, Bayesian regularization enhances the model's generalization ability and stability, making it particularly suitable for complex or noisy datasets where traditional regularization techniques may fall short (Dalipi & Yildirim, 2015).

Using the Bayesian inference framework, this involves optimizing the weight distribution to accurately map inputs to outputs, thereby preventing overfitting. Typically, training a neural network focuses on minimizing the sum of squared errors, denoted as $F = E_D$. However, when regularization is introduced, the objective function includes an additional term and takes the form (Equation 2):

$$F = \beta E_D + \alpha E_W \tag{2}$$

where E_W represents the sum of the squared network weights, and α and β are parameters that control the balance between minimizing errors and reducing model complexity. The relative values of and α and β determine the training behavior: when $\alpha \ll \beta$, the training focuses more on minimizing error; when $\gg \beta$, the model prioritizes reducing the weight magnitudes, resulting in

smoother outputs at the potential cost of higher error (Dalipi & Yildirim, 2015).

A major challenge in applying regularization is choosing appropriate values for these parameters. The most effective approach involves calculating the Hessian matrix, which can be computationally intensive. To address this, Dalipi and Yildirim, (2015) suggested using a Gauss-Newton approximation of the Hessian, which can be efficiently computed when employing the Levenberg–Marquardt algorithm during training.

In a Bayesian context, neural network weights are treated as random variables. Once data is observed, the posterior distribution of these weights can be updated using Bayes' theorem (Equation 3):

$$P(w|D,\alpha,\beta,M) = \frac{P(D \mid w,\beta,M)\,P(w|\alpha,M)}{P(D\,|\alpha,\beta,M)} \tag{3}$$

where D represent the observed dataset, M denoted the selected neural network model, and w be the vector of model weights. The prior distribution $P(w\,|\alpha,M)$ reflects the belief about weights before any data is observed. The likelihood function $P(D\,|w,\beta\,,M)$ represents the probability of observing data D given a specific set of weights w. The term. $P(\,D\,|\alpha,\beta,M)$ serves as the normalizing constant, ensuring the posterior distribution is properly scaled. Assuming Gaussian noise in the data and a Gaussian prior for the weights, the probability distributions are expressed as Equations 4, 5, and 6:

$$P(D \mid w\,,\beta\;\; M) = \frac{1}{Z_D(\beta)} exp(-\alpha E_D) \tag{4}$$

$$P(w|\alpha\,,M) = \frac{1}{Z_W(\alpha)} exp(-\alpha E_W) \tag{5}$$

$$P(D\,|\alpha,\beta,M)\; = \frac{Z_F(\alpha,\beta)}{Z_D(\beta)Z_W(\alpha)} \tag{6}$$

where the partition functions are given by Equation 7:

$$Z_D(\beta) = \left(\frac{\pi}{\beta}\right)^{\frac{n}{2}} \quad and \quad Z_{w(\alpha)} = \left(\frac{\pi}{\alpha}\right)^{\frac{N}{2}}. \tag{7}$$

To estimate $Z_F(\alpha, \beta)$, apply a Taylor series expansion around the most probable weights w^{MP}, under the assumption that the objective function is approximately quadratic near the minimum. This gives Equation 8:

$$Z_F = (2\pi)^{N/2} \, (det((\boldsymbol{H}^{MP})^{-1})^{\frac{1}{2}} exp(-F(w^{MP}))) \, , \tag{8}$$

where $H = \beta\nabla^2 E_D + \alpha\nabla^2 E_W$ is the Hessian of the objective function defined earlier in Equation 2. Substituting this into Equation 6 allows for determining the optimal regularization parameters by taking the logarithmic derivative and setting it to zero, yielding Equations 9 and 10:

$$\alpha^{MP} = \frac{\gamma}{2E_W(w^{MP})} \tag{9}$$

$$\beta^{MP} = \frac{n-\gamma}{2E_D(w^{MP})} \tag{10}$$

The term $\gamma = N - 2\alpha^{MP} \, tr(H^{MP})^{-1}$ is known as the effective number of parameters, representing how many parameters in the network significantly contribute to error reduction. This value can range from 0 to N, the total number of weights.

To avoid the computational cost of calculating the full Hessian matrix, the Gauss-Newton approximation is used. This approximation is readily available in training methods such as the Levenberg–Marquardt algorithm (Dalipi & Yildirim, 2015).

The following steps for Bayesian Regularization Procedure using Gauss-Newton Approximation as shown in Figure 2:

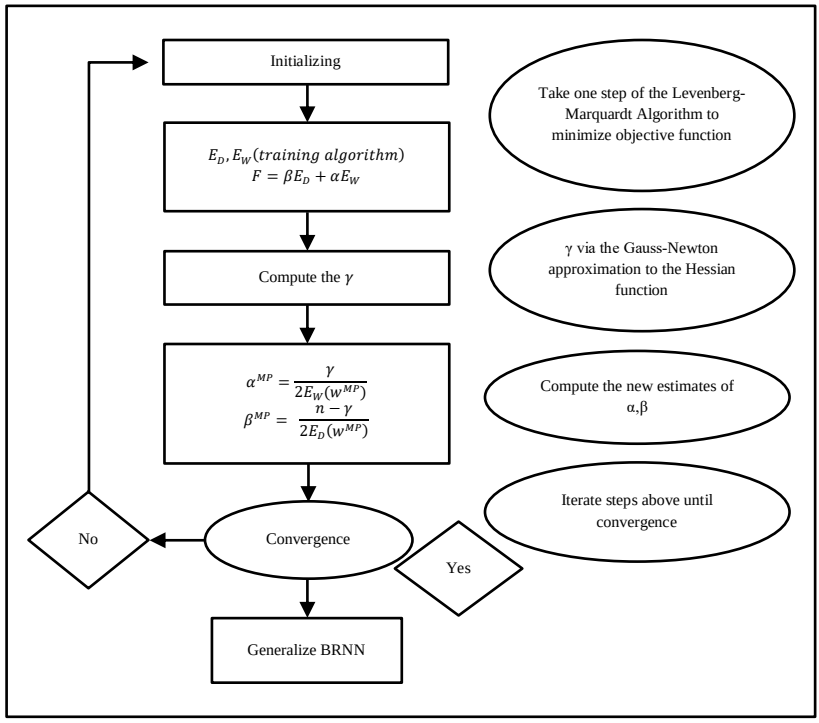

Figure 2. Flow chart for Bayesian optimization regularization parameters α and β in the neural networks

Step 1. Initiate the parameters α , β and the weight vector w. After the initial training step, the model will update these values from their initial states.

Step 2. Apply one iteration of the Levenberg–Marquardt algorithm to minimize the regularized objective function $F = \beta E_D + \alpha E_W$;

Step 3. Compute the effective number of parameters $\gamma = n - 2\alpha\, tr(\boldsymbol{H})^{-1}$, utilizing the Gauss-Newton approximation to the Hessian matrix.

Step 4. Update the values of the regularization parameters using Equations 11 and 12:

$$\alpha = \frac{\gamma}{2E_W(w^{MP})} \tag{11}$$

$$\beta = \frac{n-\gamma}{2E_D(w^{MP})} \tag{12}$$

Step 5. Repeat Steps 2 through 4 iteratively until the algorithm converges.

2.5 Proposed Additive Hybrid Model

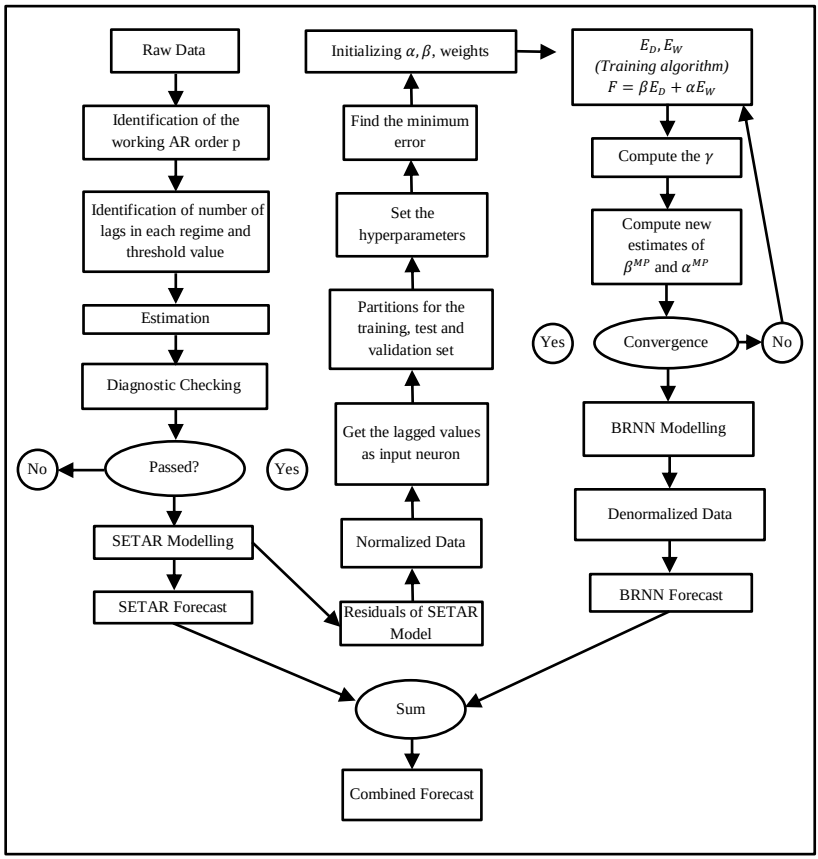

Figure 3. Hybrid SETAR-BRNN Model

A hybrid modeling approach that integrates both linear and nonlinear techniques is proposed to better capture the intricate patterns and complex autocorrelation structures present in the data. This method enhances forecasting accuracy by addressing the limitations of purely linear or nonlinear models (Atesongun & Gulsen, 2024). For illustration, the Canadian Lynx dataset known to exhibit both linear dependencies and nonlinear dynamics is used (Kajintani *et al*., 2005). The underlying assumption is that the time series can be decomposed as follows (Equation 13):

$$Y_t = L_t + N_t + \epsilon_t \tag{13}$$

where L_t represents linear component, N_t captures the nonlinear portion, and ϵ_t denotes the residuals at time t. In this framework, the linear structure is first modeled using the Self-Exciting Threshold Autoregressive (SETAR) model. The remaining residuals those unexplained by the SETAR model are then used as inputs to the Bayesian Regularization Neural Network (BRNN), which identifies the nonlinear patterns.

The final hybrid forecast is obtained by summing the predictions from the SETAR model and the BRNN. A visual summary of this hybrid SETAR-BRNN modeling process is provided in Figure 3.

2.6 Experimental Setup and Data Partitioning

The Canadian Lynx Time Series data spanning from 1821 to 1934 comprises 114 observations. To effectively develop and evaluate the forecasting models, the dataset was partitioned into distinct subsets as follows:

a. Training Set: Observations from index 3 to 91, corresponding to the years 1823-1913, and totaling 89 data points. This subset was used for initial model fitting and parameter estimation.
b. Testing Set: Observations from index 92 to 104, corresponding to years 1914-1927, and totaling 13 data points. This set was employed during model development for model comparison and hyperparameter selection.
c. Validation Set: Observations from index 105 to 114, covering the years 1928-1934, consisting of 10 data points. Representing the most recent observations, was held out entirely until the final stage of the study and was used exclusively to evaluate the forecasting performance of the selected model.

2.7 Statistical tool: Mann-Whitney U Test

This study employed the Mann-Whitney U test to statistically compare the forecasting accuracy, measure by the Mean Absolute Percentage Error (MAPE), across different models. The Mann-Whitney test was selected because it is a non-parametric method suitable for evaluating differences between independent samples when the data distribution cannot be assumed to be normal, which is often the case with forecast error metrics such as MAPE. The test statistic for the Mann-Whitney U test is (Equations 14 and 15):

$$U_1 = n_1 n_2 + \frac{n_1(n_1+1)}{2} - R_1 \quad (14)$$

$$U_2 = n_1 n_2 + \frac{n_2(n_2+1)}{2} - R_2 \quad (15)$$

Where n_1 = sample size of group 1, n_2 = sample size of group 2, R_1 = sum of ranks for group 1, and R_2 = sum of ranks for group 2.

For each model, the forecast errors (MAPE values) were obtained over multiple independent runs or cross-validation splits, resulting in sample sizes of $n_1 = n_2 = 30$ for the models being compared. These sample sizes were chosen based on the number of forecast periods, ensuring sufficient power to detect meaningful differences (Alpaslan *et al*., 2012).

In addition to the Mann-Whitney U test, effect size was calculated to quantify the magnitude of the differences between model pairs. The effect sized were interpreted using Cohen's guidelines, where values of approximately 0.2, 0.5, and 0.8 indicate small, medium and large effects, respectively.

2.8 Implementation Details

For all neural network models and time series analysis methods employed in this study, R programming software was utilized, along with specific packages such as 'RSNNS' for Multilayer Perceptron Neural Networks (MLPNN) and 'brnn' for Bayesian Regularization Neural Networks (BRNN) (Foresee & Hagan, 1997). The hardware setup included 12th Gen Intel® Core ™ i5-1235U and 8 GB RAM system, and all computations were performed on this machine to ensure consistency. The training time for the models varied depending on the complexity and parameters selected, with typical training durations approximately ranging from 10-30 minutes per model.

3. Results and Discussion

3.1 Nonlinear Time Series (SETAR)

Figure 4 illustrates the Canadian Lynx time series, which exhibits nonlinear and cyclical behavior. The residuals derived from this model will be employed in the hybridization process. Comparing the results of Polestico *et al*. (2024)

work from their Canadian Lynx data through hybridization of MLPNN model. Thus, automatically getting the time series models that they used, and these are SETAR (2,2,2), MLPNN (10-7-1) and SETAR-MLPNN.

For a SETAR (2,2,2), the model can be written as Equation 16:

$$Y_t = \begin{cases} \phi_{10} + \phi_{11} Y_{t-1} + \phi_{12} Y_t - 2 + \varepsilon_t, & if\ Y_{t-2} \leq r, \\ \phi_{20} + \phi_{21} Y_t - 1 + \phi_{22} Y_t - 2 + \varepsilon_t, & if\ Y_{t-2} > r, \end{cases} \quad (16)$$

where:

Y_t= observation at time t
r= threshold value
$d = 2$= delay parameter, so the threshold variable is Y_{t-2}
$\phi_{10}, \phi_{11}, \phi_{12}$= coefficients for Regime 1
$\phi_{20}, \phi_{21}, \phi_{22}$= coefficients for Regime 2
ε_t= white-noise error term

The selected nonlinear model was a SETAR (2,2,2), consisting of two regimes separated by a threshold value r. Each regime follows an autoregressive process of order two, while the regime-switching mechanism is determined by the delayed threshold variable Y_{t-2}. Thus, observations are governed by different autoregressive equations depending on whether the threshold variable falls below or exceeds the threshold value.

The notation MLPNN (10-7-1) means 10 input neurons, 7 hidden neurons, 1 output neuron. The mathematical form is (Equation 17):

$$\widehat{Y_t} = \beta_0 + \sum_{j=1}^{7} \beta_j f(w_{j0} + \sum_{i=1}^{10} w_{ji} X_i) \quad (17)$$

where:

$X_1, \ldots, X_{10}$ are the 10 lagged inputs,
$w_{ji}$are the input-to-hidden weights,
$\beta_j$are the hidden-to-output weights,
$f(\cdot)$is the activation function (sigmoid),
$\hat{Y}_t$is the predicted output

The SETAR-MLPNN Hybrid Model, the additive hybrid model assumes (Equation 18):

$$Y_t = L_t + N_t + \epsilon_t \quad (18)$$

where

L_t= linear/nonlinear threshold component modeled by SETAR,
N_t= nonlinear residual component modeled by MLPNN,
ε_t= random error

The SETAR Model (Equation 19):

$$\widehat{L_t} = \begin{cases} \phi_{10} + \phi_{11}Y_{t-1} + \phi_{12}Y_t - 2, & if\ Y_{t-2} \leq r, \\ \phi_{20} + \phi_{21}Y_t - 1 + \phi_{22}Y_t - 2, & if\ Y_{t-2} > r, \end{cases} \quad (19)$$

Residual Modeling by MLPNN (Equation 20)

$$\epsilon_t = Y_t - \widehat{L_t} \quad (20)$$

MLPNN Forecast (Equation 21)

$$\widehat{N_t} = \beta_0 + \sum_{j=1}^{7} \beta_j f(w_{j0} + \sum_{i=1}^{10} w_{ji} e_{t-i}) \quad (21)$$

Hybrid Forecast (Equation 22)

$$\widehat{Y_t} = \widehat{L_t} + \widehat{N_t} \quad (22)$$

The SETAR-MLPNN hybrid model combines the threshold autoregressive structure captured by the SETAR (2,2,2) model with the nonlinear residual dynamics modeled by an MLPNN (10-7-1) network. Residuals obtained from the fitted SETAR model are used as inputs to the MLPNN, and the final forecast is obtained by summing the forecasts from the SETAR and MLPNN components.

Table 1. Data partitioning of Canadian Lynx data

Partition	Number of Observation	Time Index
Training Set	89	3-91
Testing Set	13	92-104
Validation Set	10	105-114

This series is the annual number of Canadian Lynx for 1821-1934, giving the total of 114 observations. Observed that there are 2 lagged values from AR component that utilized as neuron input, since there are no available inputs for the first 2 observations, in this analysis, 102 observations were used for

modelling, and the last 10 observations were used for comparing model performance shown in Table 1.

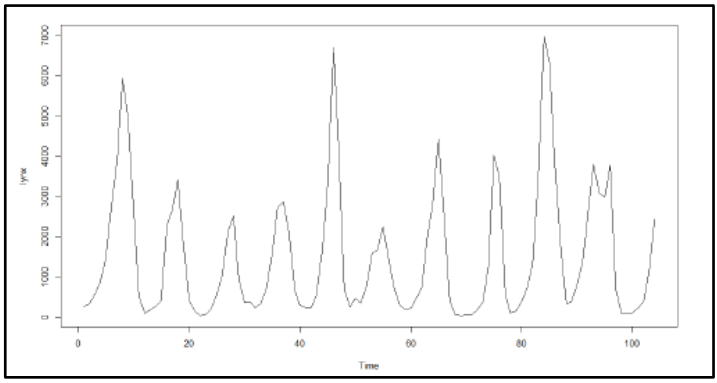

Figure 4. Time series plot of the Canadian Lynx data

In Figure 4 shows the time series plot of Canadian lynx data from year 1821-1934. It can observe that the time series data is stationary with respect to the variance since the fluctuation of the data is steady and stationary with respect to mean. Moreover, using Augmented Dickey-Fuller test where the p-value of 0.01 is less than the level of significance $\alpha=0.05$. Thus, the time series data is stationary with respect to both mean and variance.

3.2 Hybrid Time Series Model via BRNN and MLPNN

The residuals obtained from the SETAR (2, 2, 2) model are used as inputs for the hybrid modeling process with the Bayesian Regularization Neural Network (BRNN). To enhance computational efficiency and improve the accuracy of the residual forecasts, these residuals are first standardized to have a mean of 0 and a variance of 1. Prior to training the BRNN, key hyper parameters specifically, the learning rate and the number of hidden neurons must be defined. Various configurations of these hyper parameters were explored, as presented in Table 2. In total, 45 different combinations of learning rates and hidden neuron counts were evaluated.

Table 2. Setting the number of hidden neurons and learning rates in BRNN

Number of Hidden Neurons	2,3,5,9,15
Learning rate	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9

Once the hyperparameters were configured using the grid search method, the neural network was trained using the training dataset to optimize the network weights. Simultaneously, the testing dataset was used to track performance and compute the minimum error during each iteration until the algorithm reached convergence. The results indicated a sum of squared errors ED of 12.21826 and a sum of squared weights EW of 10.39893 using the Equations 2 and 6. The lowest error recorded was 1.007113, which was achieved using 8 network parameters, 2 hidden neurons, and a learning rate of 0.1 (Table 3 and Figure 5). This configuration corresponds to objective function parameters α=2.204 and β=14.0965.

To update the estimates of α and β, Bayesian regularization requires the Hessian matrix of the objective function. Therefore, the Gauss-Newton approximation is applied to reduce computational complexity during this step.

Table 3. Canadian Lynx results for BRNN hyper parameters across varying numbers of parameters, where $i \in \{8, 12, 20, 36, 60\}$

8 number of parameters and 2 hidden neurons				12 number of parameters and 3 hidden neurons			
Learning rate	α	β	error	Learning rate	α	β	error
0.1	2.204	14.1	1.0071	0.1	1.414	19.792	1.182
0.2	1.788	15.33	1.097	0.2	1.412	19.805	1.182
0.3	2.204	14.1	1.008	0.3	1.411	19.814	1.181
0.4	1.788	15.33	1.0964	0.4	1.410	19.818	1.181
0.5	1.788	15.33	1.097	0.5	1.408	19.834	1.181
0.6	1.788	15.33	1.097	0.6	1.411	19.813	1.181
0.7	1.788	15.33	1.097	0.7	1.409	19.824	1.181
0.8	1.788	15.33	1.097	0.8	1.411	19.814	1.181
0.9	1.788	15.33	1.097	0.9	1.407	19.836	1.181
20 number of parameters and 5 hidden neurons				36 number of parameters and 9 hidden neurons			
Learning rate	α	β	error	Learning rate	α	β	error
0.1	1.397	23.69	1.1084	0.1	1.244	28.1172	1.115
0.2	1.397	23.69	1.1084	0.2	1.243	28.1445	1.115
0.3	1.395	23.72	1.1083	0.3	1.242	28.1689	1.115
0.4	1.391	23.77	1.1083	0.4	1.241	28.1848	1.117
0.5	1.391	23.76	1.1081	0.5	1.244	28.145	1.115
0.6	1.392	23.74	1.108	0.6	1.243	28.1615	1.115
0.7	1.39	23.73	1.108	0.7	1.243	28.1626	1.114
0.8	1.39	23.79	1.108	0.8	1.243	28.1365	1.114
0.9	1.39	23.78	1.1078	0.9	1.242	28.186	1.118
60 number of parameters and 15 hidden neurons							
Learning rate		α		β		error	
0.1		1.0055		45.5317		1.196016	

0.2	1.0078	45.2552	1.189153
0.3	1.2435	28.1503	1.115712
0.4	1.2392	28.266	1.120525
0.5	1.2445	28.128	1.114017
0.6	1.2449	28.1129	1.115143
0.7	1.2436	28.1498	1.114124
0.8	1.244	28.1312	1.112328
0.9	1.2428	28.1637	1.112876

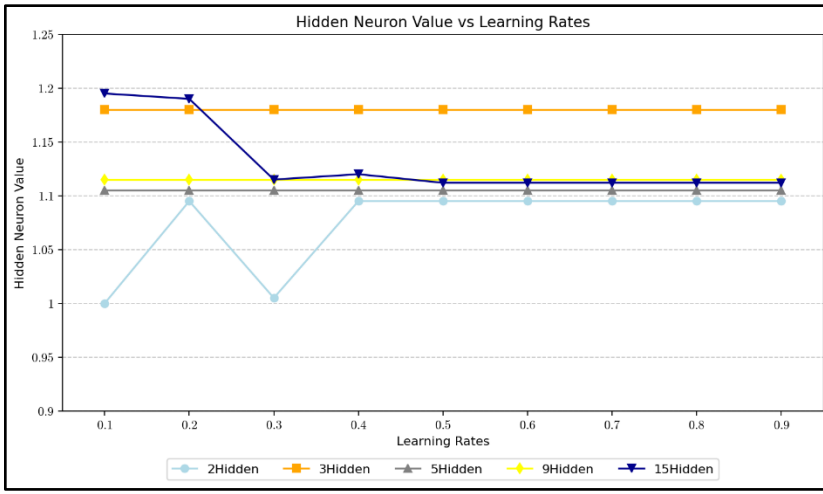

Figure 5. Line graph result of learning rate with minimum error at 0.1

The findings indicate that the effective number of parameters denoted as $\gamma = 7$, implying that 7 parameters within the neural network significantly contributed to minimizing the error when training ceased at the 25th epoch (see Table 4). The number of neurons in the hidden layer was determined through empirical testing by incrementally adjusting the neuron count until γ stabilized at a consistent value. As a result, the updated estimates for the objective function parameters are $\alpha^{MP} = 0.3392$ and $\beta^{MP} = 3.3534$ which reflect an improvement from the initial values, satisfying the condition $\alpha > \alpha^{MP}$ and $\beta > \beta^{MP}$.

Table 4. Iteration stopped at 25, showing the corresponding values of $E_D, E_W, \gamma, \alpha^{MP}$, and β^{MP}

Epoch	E_D	E_W	γ	α^{MP}	β^{MP}
1	124.9727	1.466292	2.3114	0.7882	0.3468
2	119.0666	1.188088	3.2482	1.367	0.3601
3	89.3701	1.056724	2.2142	1.0477	0.4855
4	69.21623	2.242689	2.8582	0.6372	0.6223
5	32.09258	2.875263	2.7546	0.479	1.3437
6	16.60779	4.43988	4.7354	0.5333	2.5369
7	13.9317	4.828582	5.5633	0.5761	2.9945
8	13.60215	6.362224	6.3438	0.4986	3.0384
9	13.52437	6.339007	6.1312	0.4836	3.0637
10	13.08341	7.448731	6.732	0.4519	3.144
11	12.89058	7.741926	6.699	0.4308	3.1934
12	12.71454	8.096759	6.9266	0.4277	3.2275
13	12.52703	8.465707	6.8442	0.4042	3.2791
14	12.42026	8.905159	6.9946	0.3927	3.3013
15	12.36409	9.203018	6.973	0.3788	3.3171
16	12.31128	9.585834	7.0247	0.3664	3.3293
17	12.26897	9.942115	7.0273	0.3534	3.3407
18	12.25464	10.06744	7.0445	0.3499	3.3439
19	12.23868	10.2193	7.0429	0.3446	3.3483
20	12.23648	10.22548	7.0495	0.3447	3.3486
21	12.22591	10.32496	7.0499	0.3414	3.3515
22	12.22243	10.35882	7.0535	0.3405	3.3523
23	12.22045	10.37824	7.054	0.3398	3.3528
24	12.21894	10.39236	7.0541	0.3394	3.3532
25	12.21826	10.39893	7.0549	0.3392	3.3534

The finalized BRNN model uses 2 lagged input values, 2 hidden neurons (as identified through optimal hyper parameter tuning), and 1 output neuron, resulting in the model architecture BRNN (2-2-1). Once the optimal structure was determined, the model was applied to produce 10-step ahead forecasts, and the predictions were subsequently denormalized to evaluate forecasting accuracy.

For model comparison, the last 10 data points from the validation set (observations 105 to 114) were used. Table 5 presents the forecasting performance of three models SETAR (2, 2, 2), BRNN, and the hybrid SETAR-BRNN for 1-step, 5-step, and 10-step ahead predictions.

Table 5. Forecasting accuracy for $i - step$ ahead predictions using SETAR (2, 2, 2), BRNN and hybrid SETAR-BRNN where $i \in \{1,5,10\}$

Models	1-step ahead		5-step ahead		10-step ahead	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
SETAR	0.268	3.28	0.442	5.28	0.38	4.17
BRNN	0.2135	2.6096	0.2963	3.3646	0.237	2.65
SETAR-BRNN	0.0274	0.335	0.296	2.852	0.233	2.261

The results demonstrate that the SETAR-BRNN hybrid model outperforms the individual models in forecasting accuracy, as it consistently achieves the lowest values of Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) across 1-step to 10-step ahead forecasts. This consistency highlights the robustness of the hybrid approach for both short- and long-term forecasting. Observed that, the MAPE of the SETAR-BRNN model is reduced by approximately 45.8% compared to the standalone SETAR model, underscoring the effectiveness of integrating BRNN in capturing the nonlinear components of the time series. These findings confirm that the hybrid SETAR-BRNN model delivers superior forecasting performance relative to both the SETAR and BRNN models.

For additional comparison, the forecasting results of the Multilayer Perceptron (MLPNN) and the SETAR- MLPNN hybrid for 1-step, 5-step, and 10-step ahead forecasts are referenced from the study conducted by Polestico *et al*. (2024) as shown in Table 6.

3.3 Comparison of BRNN and MLPNN

As presented in Table 6, the SETAR-BRNN model yields the lowest RMSE and MAPE values for both short-term and long-term forecasting horizons, indicating its superior predictive performance and model fit compared to both the SETAR and SETAR- MLPNN models. Among the three, SETAR also performs better than SETAR- MLPNN, highlighting its capability in capturing the underlying dynamics of the data. Furthermore, to statistically assess the differences in forecasting accuracy (in terms of MAPE), the Mann-Whitney test was employed, providing additional evidence of the significance of the observed performance differences.

Table 6. Forecasting performance comparison for short-term horizons (i ∈ {1,5}) and long-term horizon (j=10) using SETAR, BRNN, MLPNN, SETAR-BRNN and SETAR- MLPNN models

Models	1-step ahead		5-step ahead		10-step ahead	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
SETAR	0.268	3.28	0.442	5.28	0.38	4.17
BRNN	0.2135	2.6096	0.2963	3.3646	0.237	2.65
MLPNN	0.24	2.98	0.284	3.44	0.6674	7.43
SETAR-BRNN	0.0274	0.335	0.296	2.852	0.233	2.261
SETAR-MLPNN	0.573	6.99	0.5858	6.39	0.671	6.645

Table 7. Displays the results of statistical significance testing conducted between different model pairings: SETAR vs. SETAR-BRNN, SETAR-BRNN vs. SETAR-MLPNN, Actual vs. SETAR-BRNN, and SETAR vs. SETAR-MLPNN

Group Comparison	1-step ahead	5-step ahead	10-step ahead	10-step Effect Size
Actual vs. SETAR-BRNN	1.0	0.6858	0.0469*	0.709
SETAR vs SETAR-BRNN	1.0	0.0796	0.0206*	0.564
SETAR-BRNN vs SETAR-MLPNN	1.0	0.3452	0.0469*	0.6
SETAR vs. SETAR-MLPNN	1.0	0.0431*	0.0051**	0.236

(*) represents a p-value less than 0.05, indicating a statistically significant difference, (**) corresponds to a p-value less than 0.01, denoting a highly significant difference

As shown in Table 7, the comparison between Actual and SETAR-BRNN reveals no significant difference in forecasting performance for the 1-step and 5-step ahead horizons. However, for the 10-step ahead forecast, a p-value of 0.0469* suggests statistical significance at the 5% level, leading to the rejection of the null hypothesis. This result implies that the SETAR-BRNN hybrid model closely aligns with the actual values and effectively captures the underlying patterns of the Canadian Lynx time series. Similarly, the comparisons between SETAR and SETAR-BRNN as well as SETAR-BRNN and SETAR-MLPNN show no significant differences for short-term forecasts (1-step and 5-step ahead). In contrast, for the 10-step ahead forecast, the results indicate a significant difference ($p < 0.05$), demonstrating the superior long-term forecasting capability of the SETAR-BRNN model, as evidenced by its lower RMSE and MAPE values.

The comparison between SETAR and SETAR-MLPNN reveals a significant difference in the 5-step and 10-step ahead forecasts, with the SETAR model outperforming SETAR-MLPNN in both short- and long-term horizons. In addition to the statistical significance tests, effect sized were calculated to evaluate the practical magnitude of the differences between the forecasting models. The comparison between the actual values and the SETAR-BRNN model yielded the largest effect size (0.709), indicating a moderate-to-large practical differences. The comparisons between SETAR and SETAR-BRNN (0.564) and between SETAR-BRNN and SETR-MLPNN (0.6) exhibited moderate effect sizes, suggesting meaningful differences in forecasting performance. Conversely, the comparison between SETAR and SETAR-MLPNN produced a small effect size (0.236), indicating that although statistically significant differences were observed, the practical magnitude of the difference between these two models were relatively limited.

Overall, when evaluating the forecasting performance of SETAR, MLPNN, and BRNN, it becomes evident that the neural network models compete well with SETAR, particularly for longer-term predictions. Among them, BRNN shows better long-term forecasting accuracy than both MLPNN and SETAR, while SETAR performs better than MLPNN over both short- and long-term horizons as shown in Figure 6.

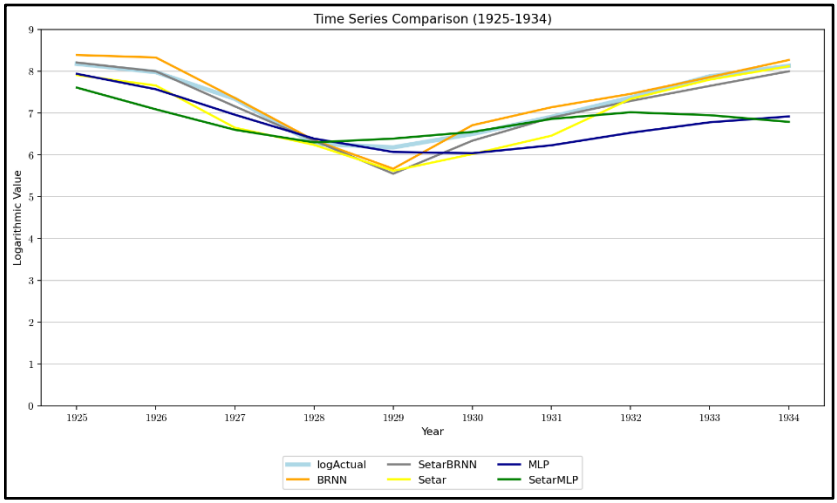

Figure 6. Time series comparison (1925–1934) showing the forecasting performance of the selected models relative to the actual values (log-transformed lynx data)

3.4 Canadian Lynx in ARIMA Model with Hybridization

To examine a time series that was initially modeled using an ARIMA approach and later combined with BRNN and MLPNN techniques for hybrid forecasting. The Canadian Lynx data intentionally executed an ARMA model as a benchmark linear model for comparison purposes. Although ARMA can be fitted to the stationary Canadian Lynx series, it does not imply that the underlying data-generating process is linear. Rather, the ARMA model serves as a baseline to evaluate whether the proposed nonlinear hybrid models provide meaningful improvements over conventional linear approaches.

The Canadian Lynx data series appeared stationary, the identified model was ARMA (2, 0, 1). Following the same methodology, the residuals from the ARMA (2, 0, 1) model were used as inputs for the hybridization process involving BRNN and MLPNN. This led to the development of two hybrid models: ARMA-BRNN (2-5-1) and ARMA-MLPNN (2-15-1).

As shown in Table 8, the ARMA-MLPNN model recorded the lowest RMSE and MAPE for 1-step ahead forecasting, while the ARMA model alone performed best for the 5-step ahead horizon. On the other hand, the SETAR-BRNN model outperformed all others in long-term forecasting, achieving the lowest MAPE values of 2.852 and 2.261 for the 5-step and 10-step ahead forecasts, respectively.

Table 8. Presents the performance results of ARMA, SETAR, SETAR-MLPNN, SETAR-BRNN, ARMA-MLPNN and ARMA-BRNN modes for $i-step$ ahead predictions where $i \in \{1, 5, 10\}$

Models	1-step ahead		5-step ahead		10-step ahead	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
ARMA	0.175	2.14	0.2148	2.9696	0.7286	8.184
SETAR	0.268	3.28	0.442	5.28	0.38	4.17
SETAR-MLPNN	0.573	6.99	0.5858	6.39	0.671	6.645
SETAR-BRNN	0.0274	0.335	0.296	2.852	0.233	2.261
ARMA-MLPNN	0.017	0.2088	0.5613	4.823	0.7485	8.163
ARMA-BRNN	0.209	2.557	0.3259	3.0063	0.7115	7.666

These findings highlight the necessity of aligning the model structure with the inherent properties of the dataset. Since the Canadian Lynx series exhibits nonlinear and cyclical characteristics, linear models such as ARMA, as well as its hybrids (ARMA-MLPNN and ARMA-BRNN), are less effective for long-term prediction. A visual comparison of the forecasting performance between ARMA-based and SETAR-based models is presented in Figure 7.

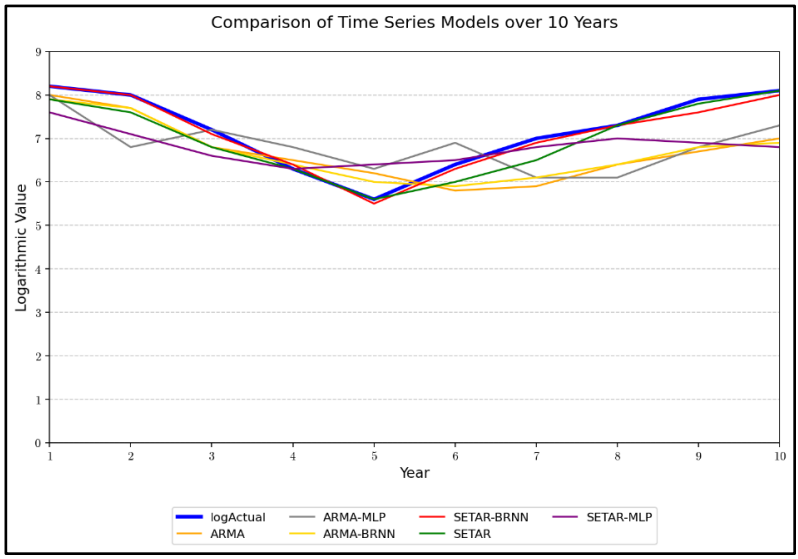

Figure 7. Time series comparison over a 10-year period showing the forecasting performance of ARMA and SETAR models and their hybrid variants (ARMA-BRNN, ARMA-MLPNN, SETAR-BRNN, and SETAR-MLPNN) relative to the actual log-transformed values.

4. Conclusion and Recommendation

4.1 Comparative Analysis of Forecasting Performance

This study explored the application of Bayesian Regularization Neural Networks (BRNN) and Multilayer Perceptron Neural Networks (MLPNN), along with their hybridization with time series models. Using the Canadian Lynx dataset spanning from 1821 to 1934, the hybrid models suggested improved forecasting accuracy relative to the benchmark linear (ARMA) and nonlinear models evaluated in this study. Observed that, the results of the

SETAR model from Polestico *et al*. (2024) provided valuable insights into the nonlinear and cyclical patterns in the data, underpinning the effectiveness of hybrid approaches in capturing complex dynamics.

Among all models tested, the SETAR-BRNN hybrid consistently produced the lowest forecast errors from 1-step to 10-step ahead, indicating strong predictive performance. Furthermore, the Mann-Whitney test showed that the difference between actual values and SETAR-BRNN predictions was statistically significant ($p = 0.0469$), suggesting that the proposed hybrid model captures important nonlinear characteristics of the Canadian Lynx series. The moderate effect sized observed for most significant comparisons indicate that the improvements achieved by the hybrid model are not only statistically significant but also practically meaningful for long-horizon forecasting.

Overall, the findings suggest that integrating Bayesian Regularization Neural Networks with threshold autoregressive models constitutes a promising approach for improving nonlinear time-series forecasting, particularly for datasets exhibiting threshold behavior and complex nonlinear dynamics.

4.2 Limitations and Future Directions

Canadian Lynx data by a linear time series and having a result of ARMA (2,0,1). The result shows that ARMA, ARMA-MLPNN and ARMA-BRNN cannot capture the actual data since the Canadian Lynx is a nonlinear cyclical feature. Therefore, consider the associated assumption of the data prior to modeling since it had different features of nonlinear models.

However, the study's reliance on a single dataset limits the generalizability of the findings across different types of time series. Future research should consider applying these hybrid models to diverse datasets in various domains such as economics, environmental studies, and traffic analysis to validate their robustness. Additionally, exploring other nonlinear and hybrid modeling frameworks, as well as incorporating alternative linear models like ARFIMA, ARIMAX, or ARMAX, could further enhance forecasting performance. These directions will contribute to advancing the applicability and accuracy of hybrid models in real-world forecasting scenarios.

5. Acknowledgement

This research was made possible through the generous support of the Department of Science and Technology – Science and Technology Regional Alliance of Universities for National Development (DOST-STRAND) and the Premiere Research Institute of Science and Mathematics (PRISM) of Mindanao State University – Iligan Institute of Technology (MSU-IIT). Their financial assistance and technical support were instrumental in the successful completion and presentation of this study.

6. References

Alpaslan, F., Egrioglu, E., Aladag, C., & Tiring, E. (2012). A statistical research on feed forward neural networks for forecasting time series. *American Journal of Intelligent Systems, 2*(3), 21–25. https://doi.org/10.5923/j.ajis.20120203.02

Alsuwaylimi, A. (2023). Comparison of ARIMA, ANN and Hybrid ARIMA-ANN models for time series forecasting. *Information Sciences Letters, 12*(2). http://dx.doi.org/10.18576/isl/120238

Asikgil, B. (2018). An adapted approach for self-exciting threshold autoregressive disturbances in multiple linear regression. *Gazi University Journal of Science, 31*(4). https://izlik.org/JA56WB54WN

Atesongun, A., & Gulsen, M. (2024). A hybrid forecasting structure based on arima and artificial neural network models. *Applied Sciences, 14*, 7122. https://doi.org/10.3390/app14167122

Box, G.E.P., & Jenkins, G.M. (1976). Time series analysis: Forecasting and control, 2nd ed. Holden-Day, San Francisco.

Dalipi, F., & Yildirim, S. (2015). The impact of environmental factors to skiing injuries: Bayesian regularization neural network model for predicting skiing injuries. In *6th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Dallas-Fortworth, TX, USA.* https://doi.org/10.1109/ICCCNT.2015.7395218

Faruk, D.O. (2010). A Hybrid Neural Network and ARIMA model for water quality time series prediction. *Engineering Applications of Artificial Intelligence. 23(4), 586-594.* https://doi.org/10.1016/j.engappai.2009.09.015.

Foresee, F.D., & Hagan, M.T. (1997). Gauss-Newton approximation to Bayesian Learning. *Proceedings of International Conference on Neural Networks (ICNN'97), Houston, TX, USA*. https://doi.org/10.1109/ICNN.1997.614194

Gonzalo, J., & Pitarakis, J. (2002). Estimation and model selection based inference in single and multiple threshold models. *Journal of Econometrics, 110*(2), 319-352. https://doi.org/10.1016/S0304-4076(02)00098-2

Kajintani, Y., Hipel, K., & McLeod, A.I. (2005). Forecasting nonlinear time series with feed-forward neural networks: A case study of Canadian lynx data. *Journal of Forecasting, 24*(2), 105–117. https://doi.org/10.1002/for.940

Khashei, M., & Bijari, M. (2010). An artificial neural network (p,d,q) model for time series forecasting. *Expert Systems with Applications, 37*(1), 479–489. https://doi.org/10.1016/j.eswa.2009.05.044

Melina, M., Sukono, F., Napitupulu, H., Mohamed, N., Chrisnanto, Y.H., Hadiana, A., Kusumaningtyas, V.A., & Nabilla, U. (2024). Comparative analysis of time series forecasting models using ARIMA and neural network autoregression methods. *Journal of Mathematics and Its Applications, 18*(4), 2563–2576. https://doi.org/10.30598/barekengvol18iss4pp2563-2576

Miller, S., Curran, K., & Lunney, T. (2018). Multilayer perceptron neural network for detection of encrypted VPN network traffic. https://doi.org/10.1109/CyberSA.2018.8551395

Polestico, D.L., Bangcale, A.L., & Velasco, L.C. (2024). Forecasting implementation of hybrid time series and artificial neural network models. *Procedia Computer Science, 232*, 4268–4277. https://doi.org/10.1016/j.procs.2024.03.010

Rahayu, W., Santi, V.M., Siregar, D., & Djati, D.K. (2023). Hybrid ARIMA–ANN model for solving nonlinearity in time series data. *Advances in Computer Science Research, 109*, 238–244. https://doi.org/10.2991/978-94-6463-332-0_9

Ramchoun, H., Idrissi, M.A., & Ettaouil, Y.G. (2016). Multilayer perceptron: Architecture optimization and training. *International Journal of Interactive Multimedia and Artificial Intelligence, 4*. https://doi.org/10.9781/ijimai.2016.415.

Tsay, R.S. (1986). Nonlinearity tests for time series. *Biometrika, 73*(2), 461–466. https://doi.org/10.1093/biomet/73.2.461

Velasco, L.C., Polestico, D.L., Macasieb, G.P., Reyes, M.B., & Vasquez, F. (2019). A hybrid ARIMA–ANN model for load forecasting. *International Journal of Advanced Computer Science and Applications, 10*(11), 221–226. https://doi.org/ 10.14569/IJACSA.2019.0101103