IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Mathematical Modelling of Engineering Problems
Vol. 12, No. 3, March, 2025, pp. 754-762
Journal homepage: http://iieta.org/journals/mmep

Comparative Analysis of Activation Functions in Recurrent Neural Network: An Application to Indonesian Inflation Forecasting

Rahmadi Yotenka[1], Muhammad Muhajir[1*], Hermansah[2], Paulo Canas Rodrigues[3]

[1] Department of Statistics, Universitas Islam Indonesia, Yogyakarta 55584, Indonesia
[2] Department of Mathematics Education, Universitas Kepulauan Riau, Batam 29422, Indonesia
[3] Department of Statistics, Federal University of Bahia, Salvador 40170-110, Brazil

Corresponding Author Email: mmuhajir@uii.ac.id

https://doi.org/10.18280/mmep.120302

Received: 2 January 2025
Revised: 3 March 2025
Accepted: 10 March 2025
Available online: 31 March 2025

Keywords:
inflation forecasting, RNN, AF, model optimization, SGD, AdaGrad

ABSTRACT

Inflation is a complex and fluctuating economic phenomenon that requires accurate modeling approaches for effective forecasting. This study conducts a comparative analysis of activation function (AF) in Recurrent Neural Network (RNN) models for forecasting inflation in Indonesia. Three AFs: Logistic, Gompertz, and Hyperbolic Tangent (Tanh) are evaluated alongside two weight optimization methods, Stochastic Gradient Descent (SGD) and Adaptive Gradient (AdaGrad). The results show that the combination of the Logistic AF with the SGD optimizer achieves the highest accuracy, with an accuracy rate of 93.41% on training data and 93.57% on testing data. Compared to traditional statistical models such as Autoregressive Integrated Moving Average (ARIMA), Exponential Smoothing (ETS), Feedforward Neural Network (FFNN), and General Regression Neural Networks (GRNN), the RNN model demonstrated superior performance, achieving a SMAPE value of 14.81%, the lowest error among all models. These findings highlight the crucial role of AF selection in enhancing RNN model performance for economic time-series forecasting. This study provides valuable insights for researchers and policymakers to improve inflation forecasting accuracy using deep learning approaches.

1. INTRODUCTION

Inflation is an economic phenomenon characterized by a general increase in the prices of goods and services over a sustained period [1]. The government has a long-term goal of maintaining inflation stability at a low level to support economic growth and financial stability. Bank Indonesia (BI), as the monetary authority, is responsible for controlling inflation through various policies, particularly in dealing with exchange rate fluctuations and external economic shocks [2, 3].

Before the economic crisis, BI's monetary policy was more focused on rupiah stability and inflation control. However, the priority given to exchange rate policy resulted in inflation and monetary growth targets not being achieved [2, 3]. During the crisis, BI shifted to a floating exchange rate regime and raised interest rates by 70% in August 1998 to control hyperinflation caused by rupiah depreciation and uncontrolled base money growth. Although inflation was successfully reduced to a single-digit level until 2000, inflation rates rose again in subsequent years and have fluctuated ever since [4]. According to the Central Bureau of Statistics (BPS), the inflation rate in 2022 and 2023 was recorded at 5.51% and 2.61%, respectively, reflecting the continuously changing inflation dynamics [5]. Therefore, a more accurate nonlinear model is needed for inflation forecasting [6].

Inflation forecasting is a crucial aspect of monetary policy formulation and economic decision-making. Traditional statistical models such as ARIMA and ETS are often used for inflation forecasting [7]. While these models can capture linear trends and seasonal patterns, they tend to be less effective in handling fluctuating and nonlinear inflation patterns. In recent years, machine learning-based methods, particularly deep learning, have demonstrated superiority in modeling complex patterns in time series data [8, 9].

Among deep learning models, RNNs have proven effective in analyzing sequential data and capturing long-term dependencies in inflation trends [10, 11]. The main advantage of RNNs over traditional models is their ability to store information from previous observations, enabling more accurate modeling. RNNs have also been shown to outperform autoregressive models and backpropagation neural networks in inflation forecasting in various countries such as China [8], India [9], Canada [10], and the United Kingdom [11]. However, one crucial aspect of RNN optimization that remains underexplored is the selection of AFs, which play a significant role in gradient propagation and model stability, ultimately affecting prediction accuracy [12].

Previous research has shown that various AFs, such as Logistic [13], Gompertz [14], and Tanh [15], exhibit higher accuracy compared to ReLU. Additionally, optimization methods such as SGD [16] and AdaGrad [17] can enhance the

effectiveness of forecasting models.

In 2021, the research conducted by Szandała [18] evaluated various activation methods for artificial neural networks, including Sigmoid, Tanh, and ReLU, and provided guidelines for selecting the best activation algorithm for real-world applications. In 2022, Ali et al. [19] compared 26 AFs in Long Short-Term Memory (LSTM) models for classification and found that alternative functions such as Modified Elliott and Softsign achieved higher accuracy than Tanh. Meanwhile, Ranjan et al. [20] demonstrated that using log-Sigmoid activation in LSTM for time-series classification resulted in improved accuracy with various optimization algorithms. This research underscores the crucial role of AF selection in enhancing the performance of artificial neural networks.

However, to date, no studies have specifically examined the impact of AF selection and weight updates in RNN models for time-series applications, particularly in the context of inflation forecasting. Therefore, this study aims to fill this gap by conducting a comparative analysis of various AFs in RNN models. This research will consider three AFs Logistic, Gompertz, and Tanh along with two weight update methods SGD and AdaGrad to compare predictive model performance.

By comparing RNN based forecasting results with traditional statistical models such as ETS, ARIMA, FFNN, and GRNN, this study will provide empirical insights into the optimal AF for inflation forecasting [12]. The findings of this research are expected to contribute to the literature on economic forecasting and deep learning while providing practical insights for policymakers and researchers to improve the accuracy of financial time-series prediction models.

2. METHODS

2.1 RNN

RNN is a method that is used repeatedly to analyze the input (generally the sequential type of data) as part of deep learning. The RNN shown in Figure 1 utilizes looping structures to simulate human decision-making, by saving and retrieving old data to be utilized at any time [21]. This type of neural network generates its output based on present input, prior outputs, inputs, and the hidden state of the system [22]. RNN is a hidden state network distributed over time, making it possible to keep huge quantities of data about the past efficaciously. The principle of its design is that its recurrent connectivity allows for the memories of inputs from the past to be stored in the internal structure of the network [23] thus influencing the output of the network [24].

The notation $x^{(t)}$ represents the input during every time step t. st denotes the hidden state at every time step t, and $o^{(t)}$ indicates the output for every time step that is t. U, V, and W are a matrix of parameters used in an RNN.

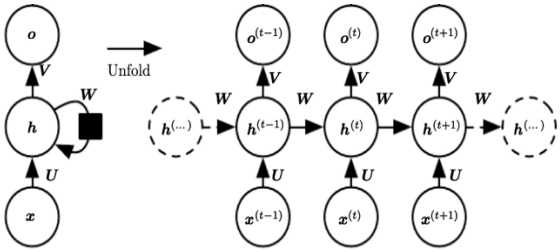

Figure 1. Architecture of RNN

2.2 AF

AF can be described as functions of the RNN that are involved in the calculation of biases and weights. These functions are designed to generate results from the neural network. They are an integral part of the process of training and optimization of a neural network they are accountable for analyzing patterns in the dataset. AFs play an essential part as they facilitate the non-linear mapping process between both output and input data of deep neural networks [25]. Many activation methods have been designed and implemented in various neural network models. The functions employed include Logistic Gompertz, Logistic, and Tanh [26, 27].

The function of logistic activation also referred to as the sigmoid function, is still a common option in a variety of neural network models because of its high computational efficiency as well as an easy definition of the model [27, 28]. The formula for the logarithmic AF can be explained as follows [29-31]:

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

This is known as the Tanh function which is commonly called Tanh. In addition to sigmoid or logistic, Tanh is also frequently utilized as an activation factor within neural networks [27]. Tanh is usually preferred over logistics due to its symmetry in the direction of origin. In addition, Tanh has gradients that are flexible in their direction change and have a zero-centered [32]. It is also known that the Tanh function can be described as continuous, varies, and has values that range between -1 and 1. Tanh function is defined by Tanh function is described as [29, 30]:

$$f(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2}$$

The Gompertz AF was an algorithm invented by Benjamin Gompertz to analyze his model of demographics, which is an enhancement of the Malthus model [33]. In the realm of biology, it is in the field of biology, Gompertz function, or the curve, is often utilized to track growth patterns, in which the growing phase is shorter than the period of decreasing growth [34]. It is the Gompertz function, used to analyze time series is a specific variant of the generalized logistics function. The Gompertz function is typically represented as follows [14]:

$$f(x)=a*exp(-\exp(b-c*x)) \tag{3}$$

In this case, $f(x)$ is the anticipated value as a function of time x. a is the higher asymptote, b controls the change along the x-axis (moving the graph either direction) while c represents the coefficient of growth (which influences the slope) The Euler's number (e equals 2.71828) as well $exp(x)$ is the ex.

2.3 Optimizer

Neural networks have attracted the attention of academics and industry due to their exceptional performance in various pattern recognition and machine learning tasks. Other machine learning models, such as support vector machines (SVMs), decision trees, and RNNs, have unique structures and have been thoroughly researched theoretically. Therefore, there are specialized optimization algorithms designed for these

models. However, for neural networks, theoretical results that can be used to design specialized optimization algorithms are still very limited. As an alternative, gradient descent, a general algorithm, is often the first choice [35].

Optimizer is an algorithm used to update weights and biases during the learning process of artificial neural networks, aiming to reduce the error or difference between the network output and the desired target. The most commonly employed foundational algorithm is the gradient descent. However, this algorithm comes with one drawback, because it is extremely slow to accomplish a low error rate. The most commonly employed foundational algorithm is the gradient descent. However, this algorithm comes with one drawback, because it is extremely slow to accomplish a low error rate [36]. To overcome this problem different algorithms are being developed, such as SGD and AdaGrad.

SGD is commonly utilized as a substitute for gradient descent in dealing with large-scale issues. While SGD typically requires more iterations than gradient descent, this method significantly decreases the requirement for storage and computation since it doesn't have to store or compute gradients based on all instances [37]. SGD changes parameters according to each trial, typically which makes it more efficient. The process can be carried out with many variances that can result in significant variations in the goal function. This can lead to SGD frequently exceeding its targets and making it difficult to reach a precise minimum value. However, with the gradual decrease in the learning rate, SGD eventually exhibits convergence behavior similar to batch gradient descent and is almost certain to converge to a local or global minimum. The SGD optimization formula can be written as follows [38]:

$$\theta = \theta - \eta \nabla \left(J(\theta x_i y_i) \right) \tag{4}$$

where, η is a learning rate, x_i, y_i is the i-th data sample, where x_i is the input y_i is the corresponding label or target, and θ is the parameter vector of the model being optimized. At each iteration, the value of θ will be updated to minimize the loss function J.

The AdaGrad optimizer is a gradient-based optimization algorithm and a modified version of SGD that uses a per-parameter learning rate, meaning it adjusts the learning rate according to each parameter. AdaGrad can enhance the performance of SGD by adjusting the learning rate [27]. The update formulas used are as follows [39]:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot g_t \tag{5}$$

where, η is a learning rate, θ_t is the parameter at iteration t, G_t is the cumulative sum of the gradients, ϵ is a small value to prevent division by zero (usually ϵ is around 10^{-8}), and g_t is the gradient for parameter θ at iteration t.

2.4 Methodology

This section outlines the methodology used for time series data prediction, specifically in the context of forecasting inflation rates. The case study focuses on inflation in Indonesia, using monthly Indonesian inflation data from January 2005 to December 2023. This data was sourced from BI and is accessible at https://www.bi.go.id/id/statistik/indikator/data-inflasi.aspx.

The Indonesian inflation data was analyzed using a machine learning approach with RNNs. A comparative study was conducted on various AFs, namely Logistic, Gompertz, and Tanh, as well as weight updating or optimization functions in the RNN model using the SGD and AdaGrad optimizers. The objective is to identify the best-performing prediction model.

This study aims to identify the most effective RNN model for forecasting inflation trends in Indonesia. The analysis follows a structured approach, adopting the research algorithm proposed by Hermansah et al. [40] to evaluate and optimize RNN performance.

1. Determining neurons in the input layer (autoregressive lag) by analyzing data frequency x, assuming it represents time series data for forecasting. If the data frequency is m, autoregressive lags from 1 to m are used. For instance, for monthly data, the lag is set to 1:12.

2. Data preprocessing, including scaling or normalization, using the equation:

$$y = \frac{x - min(x)}{max(x) - min(x)} \tag{6}$$

where, y is the normalized value; x is the data to be predicted; min(x) and max(x) are the minimum and maximum predicted data values.

3. Splitting the dataset into training data (January 2005 - December 2022) and testing data (January 2023 - December 2023).

4. Building the RNN model by configuring key parameters, including the number of neurons, learning rate, epochs, optimization method, AF, and hyperparameter tuning.

5. Performing data predictions.

6. Denormalizing the predicted results using the equation:

$$x^* = y * [max(x) - min(x)] + min(x) \tag{7}$$

to restore values to their original scale, where x^* is the value of data denormalization.

7. Predicted and actual data using Root Mean Squared Error (RMSE) and Symmetric Mean Absolute Percentage Error (SMAPE) values. RMSE is a measure used to assess how well a predictive model is at predicting value. RMSE measures the difference between the value predicted by the model and the actual value, giving more weight to larger errors. A lower RMSE value indicates a model that is better at making predictions, because it shows that the average prediction error is smaller. A higher RMSE value indicates that the model is less accurate in its predictions, with larger errors. RMSE also has the same units as the measured data, making interpretation easier. However, keep in mind that RMSE is sensitive to outliers, so it should be considered along with other metrics for a more comprehensive model evaluation. RMSE is calculated using the following formula:

RMSE, which measures prediction errors, calculated as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(A_i - F_i)^2} \tag{8}$$

A lower RMSE indicates better model accuracy, while a higher RMSE suggests greater prediction errors. RMSE is sensitive to outliers and should be considered alongside other evaluation metrics.

SMAPE, which assesses forecasting accuracy by comparing predictions with actual values, calculated as:

$$SMAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|A_i - F_i|}{\frac{|A_i| + |F_i|}{2}} \times 100\% \quad (9)$$

where, A_i is the actual value in the period i, F_i is the predicted value in period i, and n is the number of data points.

8. Forecasting inflation in Indonesia for the period January 2023 - December 2023.

Figure 2 shows the main steps of the RNN model process to predict inflation. This flowchart summarizes phases of research that begin at data processing and models design through evaluation and forecasting. Every phase plays an important part in the precision as well as the reliability of forecasts.

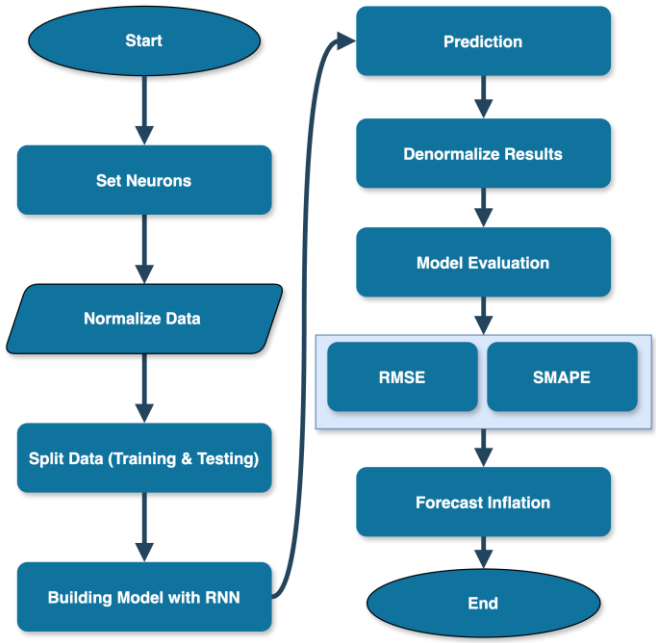

Figure 2. RNN modeling flowchart for inflation prediction

3. RESULTS AND DISCUSSION

In this research, a case study is conducted using data on the inflation rate in Indonesia. The data observed is monthly data starting from January 2005 to December 2023. Training data is collected on the first 216 data (from January 2005 to December 2022) and the last 12 data are used as testing data (from January 2023 to December 2023). This data can be found and accessed on the BI website. Furthermore, testing was conducted consisting of testing the optimization model by comparing two models, testing the AF by comparing three functions, testing the learning rate, and testing the number of epoch iterations.

3.1 Comparing two optimization model

This research uses SGD and AdaGrad optimization methods. SGD is an iterative learning algorithm that uses training data to update the model. The algorithm is iterative which means that each step will endeavor to slightly improve the model parameters. Each iteration involves using the model with the current parameters to make predictions on some training data, comparing the predictions with the expected results, calculating the error, and using the error to update the model parameters [41]. AdaGrad is a derivative algorithm of SGD that adapts to the learning rate with smaller parameters and model updates. These two methods are often used for model optimization.

Furthermore, the neurons in the input layer in this study are proposed based on lag autoregressive with frequency approach, where time series data has time and frequency attributes. The time attribute states the time unit of each observation point, while the frequency attribute states the quantity of data in a certain period, usually defined per year, such as monthly data (frequency =12), quarterly data (frequency =4), quarterly data (frequency =3), semi-annual data (frequency =2), and annual data (frequency =1). In addition, both optimization models use a learning rate =0.05 and 500 epochs to get more detailed training from both optimization methods. The AF used is the Gompertz AF and the determination of other hyperparameter values as reported by Keskar and Socher [42]. The test results with both optimization methods can be seen in Table 1.

Table 1. Optimization model test result

Optimization	Training Data		Testing Data	
	SMAPE	Accuracy	SMAPE	Accuracy
SGD	0.19957	0.93289	0.15071	0.92589
AdaGrad	0.67879	0.38981	0.80961	0.44716

Based on the test results in Table 1, the SGD and AdaGrad optimization models produce significantly different levels of accuracy, whereas the SGD optimization model provides much better accuracy. The SGD optimization method is superior to the AdaGrad optimization method because it has a much lower error value (SMAPE value) and a much higher accuracy value. The SGD optimization method shows to have an advantage in the optimization model by getting a SMAPE value of 19.96% for training data and 15.07% for testing data. At the same time, it shows that it has an accuracy value of 93.29% for training data and 92.59% for testing data. Meanwhile, the AdaGrad optimization model shows that it has an error value (SMAPE value) of 67.88% for training data and 80.96% for testing data. At the same time, it shows that it has an accuracy value of 38.98% for training data and 44.72% for testing data.

3.2 Comparing between three AFs

This research uses logistic, Gompertz, and Tanh AFs. These three AFs are often used for the learning process of forecasting models. In addition, the three AFs use a learning rate =0.05 and 500 epochs to get more detailed training from the SGD optimization method. Meanwhile, the autoregressive lag and hyperparameter values are determined in the same way as testing the optimization model. The test results with the three AFs can be seen in Table 2.

Table 2. AF test results

AF	Training Data		Testing Data	
	SMAPE	Accuracy	SMAPE	Accuracy
Logistic	0.20498	0.93948	0.15329	0.93081
Gompertz	0.19957	0.93289	0.15071	0.92589
Tanh	1.39794	0.01113	1.46814	0.04255

Based on the results presented in Table 2, the logistic and Gompertz AFs exhibit similar accuracy levels, with only minor differences. In contrast, the Tanh AF produces the lowest accuracy. Both the logistic and Gompertz AFs have their respective strengths and weaknesses. However, this study indicates that the logistic AF outperforms the Gompertz AF due to its higher accuracy. Specifically, the logistic AF achieves an accuracy of 93.95% for training data and 93.08% for testing data, while the Gompertz AF attains 93.29% for training data and 92.59% for testing data. Despite its slightly lower accuracy, the Gompertz AF demonstrates a lower error value, as reflected in its SMAPE score.

The logistic AF proves to be the most effective among the three tested functions (Logistic, Gompertz, and Tanh). One key factor contributing to its superiority is its stability during training. Being a monotonic function, logistic activation prevents drastic fluctuations in gradients, ensuring stable model learning—especially when using the SGD optimization method applied in this study. Additionally, logistic activation has the advantage of output scalability, producing values in the range (0,1), which makes it well-suited for prediction tasks with limited-scale data. Although the Gompertz AF shares similar properties, its slightly different growth curve results in marginally lower accuracy. The logistic AF also mitigates the vanishing gradient problem more effectively than tanh. While tanh is widely used in neural networks due to its broader output range (-1,1), it is more susceptible to vanishing gradients, leading to slower training and convergence issues, ultimately resulting in significantly lower accuracy. The superior performance of logistic activation is further evident in its lower SMAPE score compared to Gompertz, indicating its ability to capture relationships within the data more effectively. However, the Gompertz AF has a slight advantage in terms of lower error, likely due to its ability to model gradual changes more efficiently.

On the other hand, Indonesia's inflation data has distinct characteristics that affect the effectiveness of the AFs used in the model. One of the primary characteristics is its long-term fluctuations and trends. Inflation tends to follow cyclical patterns with short-term fluctuations and long-term trends influenced by monetary policy, commodity prices, and global factors. Therefore, AFs that can capture these changing patterns without being overly sensitive to noise are preferred. Logistic and Gompertz functions are capable of handling gradual growth patterns, whereas tanh is more sensitive to drastic changes, which may not align with the more stable inflation data trends in the long term.

The distribution of inflation data also plays a crucial role in selecting the appropriate AF. Generally, inflation data does not exhibit extreme changes in short periods. The logistic function, with its moderate growth curve, is more suitable for handling data with smooth variations. Conversely, Tanh is more prone to significant gradient changes, which can make the model overly sensitive to minor fluctuations that should not significantly impact inflation predictions. Additionally, inflation can be influenced by seasonal factors, such as price increases before major holidays, and government policies that cause gradual changes. In this case, the Gompertz function, which is often used in economic growth models, has an advantage in handling gradual changes. However, in this study, Gompertz still showed slightly lower performance than logistic activation. Overall, the choice of AF highly depends on the characteristics of the data used. For Indonesia's inflation data, the logistic function proved to be more stable and provided higher accuracy than Gompertz and Tanh, although Gompertz had a slight advantage in terms of lower error. Meanwhile, tanh exhibited significantly poor performance, likely due to its incompatibility with the stable inflation data patterns, making the learning process more challenging.

3.3 Learning rate testing

The optimisation model used is SGD and the epoch is 500. The AF used is logistic. While the autoregressive lag and other hyperparameter values are determined in the same way as in the optimization model testing. Furthermore, experiments were conducted with learning rates of 0.001, 0.002, 0.050, 0.100, 0.200, 0.500, and 0.900. This is done to determine the effect of learning rate size on the learning process and also data testing. The results obtained from the learning rate experiment can be seen in Table 3.

In Table 3, we can see the accuracy results based on various learning rates. The highest accuracy value is 95.65% for training data with a learning rate of 0.100, while the highest accuracy value is 91.02% for testing data with a learning rate of 0.050. At a learning rate of 0.050, the difference in the accuracy of training data and testing data is not too significant. While the learning rate 0.100 and learning rate 0.200 were obtained the results of the difference in the accuracy level of training data and testing data are very significant. This is because when the learning of a machine is too large, then the gradient descent value inadvertently increases the error rather than reducing the error during training. Therefore, the larger the learning rate value will cause errors in updating the weights which will affect the training accuracy results. If the learning rate is too small, too many iterations are required to reach the desired target. Meanwhile, if it is too large, the optimization model becomes unstable, thus preventing the error from reaching the desired target.

3.4 Epoch iteration testing

Experiments were conducted with the SGD optimization model, logistic AF, learning rate 0.05, and other hyperparameter values as in reference [42]. While the autoregressive lag is determined in the same way as testing the optimization model. This is done to determine the effect of iteration size on epoch on the learning process and also data testing. The results obtained from the epoch value experiment can be seen in Table 4.

Table 4 shows the difference in accuracy results obtained based on the epoch value. Epoch with the best accuracy value is obtained at an epoch value of 350 with results on training data of 93.41% and on testing data of 93.57%. The greater the epoch value, the better the accuracy, but if the epoch value is too large, the optimization model becomes less stable, thus preventing the error from reaching the desired target.

Furthermore, the best RNN model obtained is compared with several models available in the literature, namely the ETS model described in reference [43], the ARIMA model described in reference [44], the FFNN model described in reference [45], and the GRNN model described in reference [46]. The empirical study comparison results can be seen in Table 5 for the best RNN model with several models available in the literature. The empirical study results show that the RNN model provides the best results. This is indicated by the lowest SMAPE value on the testing data.

Table 5 shows that the ETS, ARIMA, FFNN, and GRNN models were chosen as baselines because they represent commonly used approaches in time series forecasting. The ETS model [43] was selected for its ability to capture trends and seasonality using ETS techniques. The ARIMA model [44] is widely used in time series analysis due to its capability to handle both stationary and non-stationary data through a combination of autoregression, differencing, and moving average components. The FFNN model [45] is a popular artificial neural network model for nonlinear modeling, capable of capturing complex patterns in data. The GRNN model [46] is a kernel-based neural network that employs probabilistic density estimation, making it a flexible choice for forecasting and classification tasks without requiring extensive parameter tuning. By combining classical statistical methods (ETS, ARIMA) with neural network-based approaches (FFNN, GRNN), these baseline models provide a strong comparative foundation to evaluate the advantages of RNN models in this study.

The ETS model was implemented using the *forecast* package in R or *statsmodels* in Python, with automatic selection of error (E), trend (T), and seasonality (S) components based on AIC criteria to ensure the best model. The ARIMA model was developed using the Box-Jenkins method, where the *auto.arima()* function in R's *forecast* package or *pmdarima* in Python was used to determine optimal parameters (p, d, q) to capture time series patterns. The neural network-based models were developed using TensorFlow/Keras, adopting a multilayer perceptron (MLP) architecture with one or two hidden layers and ReLU activation. Optimization was performed using the Adam algorithm, and Mean Squared Error (MSE) was used as the loss function to minimize prediction errors. The GRNN model was implemented using *scikit-learn* or *neupy*, leveraging Gaussian kernels to capture complex relationships in data. The smoothing parameter in GRNN was determined through cross-validation to enhance model generalization. Each baseline model provided a comprehensive perspective for comparing the performance of RNN models tested in this study.

Table 3. AF test results

Learning Rate	Training Data		Testing Data	
	SMAPE	Accuracy	SMAPE	Accuracy
0.001	0.40664	0.21545	0.31161	0.18597
0.002	0.40754	0.90522	0.31371	0.85066
0.050	0.19578	0.94075	0.21612	0.91017
0.100	0.18083	0.95649	0.24133	0.73286
0.200	0.19591	0.95016	0.50612	0.27017
0.500	0.22223	0.86962	0.16621	0.87568
0.900	0.38180	0.56023	0.44838	0.27857

Table 4. Epoch value testing results

Epoch	Training Data		Testing Data	
	SMAPE	Accuracy	SMAPE	Accuracy
50	0.40598	0.53419	0.30901	0.21330
100	0.37000	0.88571	0.28728	0.83467
150	0.21061	0.91785	0.20282	0.88478
200	0.23483	0.91553	0.16603	0.93558
250	0.26588	0.89410	0.33482	0.88655
300	0.27352	0.89956	0.34080	0.86936
350	0.21973	0.93413	0.16371	0.93570
400	0.23986	0.92865	0.29878	0.82488
450	0.20334	0.94695	0.15968	0.86839
500	0.19091	0.95574	0.19721	0.80388
550	0.19624	0.94549	0.13970	0.86205
600	0.17740	0.95152	0.16219	0.88050
650	0.17415	0.95987	0.15904	0.81246
700	0.17450	0.95655	0.17696	0.83959
750	0.16369	0.96125	0.18270	0.80479
800	0.16086	0.96425	0.18728	0.72012
850	0.17017	0.95816	0.18631	0.84378
900	0.16365	0.96613	0.17047	0.76152
950	0.16068	0.96550	0.17743	0.77399
1000	0.16661	0.96109	0.16752	0.81155

Table 5. Comparison results between the RNN model and several models

Month	Actual Data	RNN	ETS	ARIMA	FFNN	GRNN
Jan. 2023	5.28	8.06	2.61	2.54	2.53	3.41
Feb. 2023	5.47	6.06	2.61	2.54	2.39	3.25
Mar. 2023	4.97	5.16	2.61	2.54	2.41	3.07
Apr. 2023	4.33	4.97	2.61	2.54	2.55	2.91
May 2023	4.00	4.40	2.61	2.54	2.42	2.79
Jun. 2023	3.52	3.90	2.61	2.54	2.49	2.69
Jul. 2023	3.08	3.42	2.61	2.54	2.54	2.62
Aug. 2023	3.27	2.85	2.61	2.54	2.31	2.59
Sep. 2023	2.28	2.66	2.61	2.54	2.48	2.53
Oct. 2023	2.56	2.33	2.61	2.54	2.42	2.55
Nov. 2023	2.86	2.24	2.61	2.54	2.23	2.55
Dec. 2023	2.61	2.24	2.61	2.54	2.22	2.53
SMAPE		0.14812	0.32133	0.33857	0.38357	0.25463

In this study, the comparison between SGD and AdaGrad optimization methods showed that SGD outperformed AdaGrad. This was demonstrated by lower error values and significantly higher accuracy in SGD compared to AdaGrad. SGD updates model parameters iteratively based on small data subsets, allowing the model to quickly adapt to patterns without significant performance degradation. Conversely, AdaGrad, which adjusts the learning rate based on the magnitude of past gradients, suffered from diminishing effectiveness, as indicated by its lower accuracy in both training and testing data. Hence, SGD is recommended as the preferred optimization method for the models used in this study.

Additionally, this research compared three AFs: Logistic, Gompertz, and Tanh. Results indicated that the logistic AF achieved the best performance in terms of accuracy and error reduction. This function maps input values to a (0,1) range, making it well-suited for nonlinear forecasting tasks like inflation prediction. The Gompertz function also performed reasonably well but ranked below logistic activation in

accuracy. Meanwhile, the Tanh function exhibited significantly poorer performance compared to the other two, with the lowest accuracy and highest error. This is attributed to Tanh's broader output range (-1 to 1), making it more susceptible to vanishing gradient issues during training.

The study also examined the impact of learning rate on model performance by testing values from 0.001 to 0.900. Results showed that a learning rate of 0.100 achieved the highest training accuracy (95.65%) but led to overfitting in testing data. In contrast, a learning rate of 0.050 provided the best balance between training and testing accuracy (91.02%), making it the optimal choice. Extremely low learning rates, such as 0.001, required excessive iterations to converge, whereas very high rates, like 0.900, resulted in instability and high error rates. Thus, selecting an appropriate learning rate is critical for improving model performance.

Similarly, the effect of the number of epochs on model accuracy was analyzed. Experimental results indicated that 350 epochs yielded the best accuracy for both training (93.41%) and testing (93.57%). If the epoch count was too low (e.g., 50 or 100), the model underperformed due to insufficient training. However, excessive epochs (e.g., over 500) led to instability and overfitting risks. Hence, selecting an optimal number of epochs is essential to balance accuracy and generalization.

Finally, the developed RNN model was compared with other methods, including ETS, ARIMA, FFNN, and GRNN. The comparison revealed that the RNN model achieved the best performance, as indicated by the lowest SMAPE value (0.14812) compared to other models. This confirms that RNN-based models are superior in capturing nonlinear patterns in inflation data compared to traditional statistical methods like ETS and ARIMA, as well as other neural network models like FFNN and GRNN. Therefore, the use of RNN for inflation forecasting is validated as a more accurate and effective approach than the alternative methods tested in this study.

4. CONCLUSIONS

This study analyzes the impact of AF selection in RNNs for inflation forecasting in Indonesia. Among the Logistic, Gompertz, and Tanh AFs, combined with SGD and AdaGrad optimizers, the Logistic-SGD combination achieved the highest accuracy (93.41% on training data, 93.57% on testing data, and a SMAPE of 14.81%). This model outperformed traditional forecasting methods such as ARIMA, ETS, FFNN, and GRNN, confirming the advantage of deep learning in time-series prediction.

The results emphasize the critical role of AFs in optimizing RNN performance. Proper selection enhances prediction accuracy, model stability, and generalization capabilities, making RNN-based approaches highly effective for economic forecasting. However, this study is limited to standard RNNs and inflation data from 2005 to 2023. Future research should incorporate additional macroeconomic indicators to improve model precision.

To further advance forecasting capabilities, future studies should explore more advanced architectures such as LSTM, GRU, and hybrid models, along with ensemble learning techniques. These improvements could enhance model reliability and practical applications, benefiting policymakers, financial analysts, and researchers seeking robust economic forecasting solutions.

ACKNOWLEDGMENT

The authors would first like to thank the Department of Statistics, Universitas Islam Indonesia for providing funding for this research.

REFERENCES

[1] Farichah, S.A. (2022). Analisis inflasi di Indonesia: Pendekatan Autoregressive Distributed Lag (ARDL). Jurnal Cakrawala Ilmiah, 1(10): 2467-2484. https://doi.org/10.53625/jcijurnalcakrawalailmiah.v1i10.2577

[2] Endri, E. (2008). Analysis of factors influencing inflation in Indonesia. Economic Journal of Emerging Markets, 13(1): 1-13. https://doi.org/10.20885/ejem.v13i1.356

[3] Salsabila, D. (2024). Bank Indonesia' s role in mitigating issues of monetary economic sovereignty and human rights. Journal Central Banking Law & Institutions, 3(3): 449-470. https://doi.org/10.21098/jcli.v3i3.251

[4] Nugroho, P.W., Basuki, M.U. (2012). Analisis faktor-faktor yang mempengaruhi inflasi di Indonesia Periode 2000.1–2011.4. Doctoral dissertation, Fakultas Ekonomika dan Bisnis, 1(1): 69.

[5] Indrawati, S.M., Satriawan, E., Abdurohman. (2024). Indonesia's fiscal policy in the aftermath of the pandemic. Bulletin of Indonesian Economic Studies, 60(1): 1-33. https://doi.org/10.1080/00074918.2024.2335967

[6] Wahyudin, S. (2019). Prediksi inflasi Indonesia memakai model ARIMA dan artificial neural network. Jurnal Tata Kelola dan Kerangka Kerja Teknologi Informasi, 5(2): 57-63. https://doi.org/10.34010/jtk3ti.v5i1.2297

[7] Almosova, A., Andresen, N. (2023). Nonlinear inflation forecasting with recurrent neural networks. Journal of Forecasting, 42(2): 240-259. https://doi.org/10.1002/for.2901

[8] Yang, C., Guo, S. (2021). Inflation prediction method based on deep learning. Computational Intelligence and Neuroscience, 2021(1): 1071145. https://doi.org/10.1155/2021/1071145

[9] Rani, S.J., Haragopal, V.V., Reddy, M.K. (2017). Forecasting inflation rate of India using neural networks. International Journal of Computer Applications, 158(5): 45-48. https://doi.org/10.5120/ijca2017912866

[10] Moshiri, S., Cameron, N.E., Scuse, D. (1999). Static, dynamic, and hybrid neural networks in forecasting inflation. Computational Economics, 14: 219-235. https://doi.org/10.1023/A:1008752024721

[11] Binner, J.M., Elger, T., Nilsson, B., Tepper, J.A. (2004). Tools for non-linear time series forecasting in economics–an empirical comparison of regime switching vector autoregressive models and recurrent neural networks. In Applications of Artificial Intelligence in Finance and Economics, pp. 71-91. https://doi.org/10.1016/S0731-9053(04)19003-8

[12] Cihan, P. (2023). Effect of parameter selection on heart attack risk prediction in an RNN model. In 5th International Conference on Applied Engineering and Natural Sciences ICAENS, Konya, Turkey, pp. 56-60. https://doi.org/10.59287/icaens.964

[13] Xu, B., Huang, R., Li, M. (2016). Revise saturated activation functions. arXiv preprint, arXiv:1602.05980.

http://arxiv.org/abs/1602.05980
[14] Vilca-Huayta, O.A., Tito, U.Y. (2022). Efficient function integration and a case study with Gompertz functions for Covid-19 waves. International Journal of Advanced Computer Science and Applications, 13(8): 545-551. https://doi.org/10.14569/IJACSA.2022.0130863
[15] De Ryck, T., Lanthaler, S., Mishra, S. (2021). On the approximation of functions by tanh neural networks. Neural Networks, 143: 732-750. https://doi.org/10.1016/j.neunet.2021.08.015
[16] Banerjee, K., Georganas, E., Kalamkar, D.D., Ziv, B., Segal, E., Anderson, C., Heinecke, A. (2019). Optimizing deep learning RNN topologies on intel architecture. Supercomputing Frontiers and Innovations, 6(3): 64-85. https://doi.org/10.14529/jsfi190304
[17] Xia, M., Shao, H., Ma, X., De Silva, C.W. (2021). A stacked GRU-RNN-based approach for predicting renewable energy and electricity load for smart grid operation. IEEE Transactions on Industrial Informatics, 17(10): 7050-7059. https://doi.org/10.1109/TII.2021.3056867
[18] Szandała, T. (2021). Review and comparison of commonly used activation functions for deep neural networks. Bio-Inspired Neurocomputing, 203-224. https://doi.org/10.1007/978-981-15-5495-7_11
[19] Ali, M.H.E., Abdel-Raman, A.B., Badry, E.A. (2022). Developing novel activation functions based deep learning LSTM for classification. IEEE Access, 10: 97259-97275. https://doi.org/10.1109/ACCESS.2022.3205774
[20] Ranjan, P., Khan, P., Kumar, S., Das, S.K. (2023). Log-sigmoid activation-based long short-term memory for time-series data classification. IEEE Transactions on Artificial Intelligence, 5(2): 672-683. https://doi.org/10.1109/TAI.2023.3265641
[21] Kaur, M., Mohta, A. (2019). A review of deep learning with recurrent neural network. In 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, pp. 460-465. https://doi.org/10.1109/ICSSIT46314.2019.8987837
[22] Sehovac, L., Grolinger, K. (2020). Deep learning for load forecasting: Sequence to sequence recurrent neural networks with attention. IEEE Access, 8: 36411-36426. https://doi.org/10.1109/ACCESS.2020.2975738
[23] Sherstinsky, A. (2020). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D: Nonlinear Phenomena, 404: 132306. https://doi.org/10.1016/j.physd.2019.132306
[24] Lateko, A.A., Yang, H.T., Huang, C.M., Aprillia, H., Hsu, C.Y., Zhong, J.L., Phương, N.H. (2021). Stacking ensemble method with the RNN meta-learner for short-term PV power forecasting. Energies, 14(16): 4733. https://doi.org/10.3390/en14164733
[25] Hayou, S., Doucet, A., Rousseau, J. (2019). On the impact of the activation function on deep neural networks training. In International Conference on Machine Learning, pp. 2672-2680.
[26] Ratnawati, D.E., Marjono, M., Widodo, W., Anam, S. (2020). Comparison of activation function on extreme learning machine (ELM) performance for classifying the active compound. AIP Conference Proceedings, 2264(1): 140001. https://doi.org/10.1063/5.0023872
[27] Shen, S.L., Zhang, N., Zhou, A., Yin, Z.Y. (2022). Enhancement of neural networks with an alternative activation function tanhLU. Expert Systems with Applications, 199: 117181. https://doi.org/10.1016/j.eswa.2022.117181
[28] Lin, Y.S., Fang, S.L., Kang, L., Chen, C.C., Yao, M.H., Kuo, B.J. (2024). Combining recurrent neural network and sigmoid growth models for short-term temperature forecasting and tomato growth prediction in a plastic greenhouse. Horticulturae, 10(3): 230. https://doi.org/10.3390/horticulturae10030230
[29] Nwankpa, C., Ijomah, W., Gachagan, A., Marshall, S. (2018). Activation functions: Comparison of trends in practice and research for deep learning. arXiv preprint arXiv:1811.03378. http://arxiv.org/abs/1811.03378
[30] Dewa, C.K. (2018). Suitable CNN weight initialization and activation function for Javanese vowels classification. Procedia Computer Science, 144: 124-132. https://doi.org/10.1016/j.procs.2018.10.512
[31] Zaki, P.W., Hashem, A.M., Fahim, E.A., Mansour, M.A., ElGenk, S.M., Mashaly, M., Ismail, S.M. (2019). A novel sigmoid function approximation suitable for neural networks on FPGA. In 2019 15th International Computer Engineering Conference (ICENCO), Cairo, Egypt, pp. 95-99. https://doi.org/10.1109/ICENCO48310.2019.9027479
[32] Sharma, S., Sharma, S., Athaiya, A. (2017). Activation functions in neural networks. International Journal of Engineering Applied Sciences and Technology, 4(12): 310-316. https://doi.org/10.33564/ijeast.2020.v04i12.054
[33] Iliev, A.I., Kyurkchiev, N., Markov, S. (2017). A note on the new activation function of Gompertz type. Biomath Communications, 4(2). https://doi.org/10.11145/bmc.2017.10.201
[34] Gupta, R., Pandey, G., Pal, S.K. (2021). Comparative analysis of epidemiological models for COVID-19 pandemic predictions. Biostatistics & Epidemiology, 5(1): 69-91. https://doi.org/10.1080/24709360.2021.1913709
[35] Zhou, B.C., Han, C.Y., Guo, T.D. (2021). Convergence of stochastic gradient descent in deep neural network. Acta Mathematicae Applicatae Sinica, English Series, 37(1): 126-136. https://doi.org/10.1007/s10255-021-0991-2
[36] Nasuha, A., Sardjono, T.A., Purnomo, M.H. (2018). Pengenalan viseme dinamis bahasa Indonesia menggunakan convolutional neural network. Jurnal Nasional Teknik Elektro dan Teknologi Informasi, 7(3): 258-267. https://doi.org/10.22146/jnteti.v7i3.433
[37] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010: 19th International Conference on Computational Statistics, Paris France, pp. 177-186. https://doi.org/10.1007/978-3-7908-2604-3
[38] Alagözlü, M. (2022). Stochastic gradient descent variants and applications. Università della Svizzera Italiana. https://doi.org/10.13140/RG.2.2.12528.53767
[39] Kingma, D.P., Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. https://doi.org/10.48550/arXiv.1412.6980
[40] Hermansah, Muhajir, M., Canas Rodrigues, P. (2024). Indonesian inflation forecasting with recurrent neural network long short-term memory (RNN-LSTM). Enthusiastic: International Journal of Applied Statistics and Data Science, 4(2): 132-142.

https://doi.org/10.20885/enthusiastic.vol4.iss2.art5
[41] Tyagi, A.K., Abraham, A. (2022). Recurrent Neural Networks: Concepts and Applications. CRC Press.
[42] Keskar, N.S., Socher, R. (2017). Improving generalization performance by switching from Adam to SGD. arXiv preprint arXiv:1712.07628. https://doi.org/10.48550/arXiv.1712.07628
[43] Hyndman, R.J., Koehler, A.B., Snyder, R.D., Grose, S. (2002). A state space framework for automatic forecasting using exponential smoothing methods. International Journal of Forecasting, 18(3): 439-454. https://doi.org/10.1177/008124631004000402
[44] Hyndman, R.J., Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for R. Journal of Statistical Software, 27: 1-22. https://doi.org/10.18637/jss.v027.i03
[45] Hermansah, H., Rosadi, D., Abdurakhman, A., Utami, H. (2020). Selection of input variables of nonlinear autoregressive neural network model for time series data forecasting. Media Statistika, 13(2): 116-124. https://doi.org/10.14710/medstat.13.2.116-124
[46] Martínez, F., Charte, F., Frías, M.P., Martínez-Rodríguez, A.M. (2022). Strategies for time series forecasting with generalized regression neural networks. Neurocomputing, 491: 509-521. https://doi.org/10.1016/j.neucom.2021.12.028