Journal of Statistics Applications & Probability
An International Journal

http://dx.doi.org/10.18576/jsap/140105

An end-to-end Combined Forecasting Architecture: Forecasting Stock Price Data

Katleho Makatjane[1,*], *Claris Shoko*[1] *and Caston Sigauke* [2]

[1]Department of Statistics, University of Botswana, Goborone, Botswana
[2]Department of Mathematical and Computational Sciences, University of Venda, Private Bag X5050, Thohoyandou, 0950, South Africa

Received: 3 Mar. 2024, Revised: 20 Oct. 2024, Accepted: 4 Nov. 2024
Published online: 1 Jan. 2025

Abstract: In this paper we evaluated five models within a Bayesian framework in quantifying aleatoric uncertainty of financial time series data and, in particular, stock prices. Statistical-based predictions with deep learning algorithms improve the performance of stock price forecasting models not by choosing the model structure expected to predict the best but by developing a model whose results are a combination of models with different shapes. Using the minimum combined score to conglomerate the TBATS and the α-RNN, we found that the combined model mimics the forecasting error compared to individual deep learning algorithms, with coverages of 98.7% and 88.76%, for five and ten-day steps ahead, respectively. By leveraging TBATS for capturing complex seasonality and α-RNN for modeling memory decay and long-term dependencies, the estimated model demonstrates robust performance in short and medium-term predictions. our findings further highlight the ability of the model to address volatility clustering, trend detection, and seasonality more effectively than traditional methods such as ARIMA, GARCH, and standalone RNN-based models methods. The model significantly lowers error metrics, improves forecast accuracy and better handles financial uncertainties.

Keywords: Bayesian methodology, Exponential Smoothed Recurrent Neural Network, Anglo-American Stock prices.

1 Introduction

The fundamental units of contemporary sequential learning are recurrent neural networks (RNNs). With minimal parameters, RNNs employ recurrent layers to capture nonlinear temporal dependencies [16]. Using feedforward and recurrent neural network layers enables the translation of input sequences into hidden state sequences and their corresponding outputs. This architecture is particularly adept at capturing temporal dynamics in sequential data. Researchers have extensively explored the application of recurrent neural networks in financial time series, such as historical limit order books and price histories. Notable empirical studies, including those by [7], [6], and [34], have investigated the effectiveness of these networks in modelling and understanding the temporal dependencies present in financial data. In the realm of financial time series analysis, different approaches have been explored by various researchers; for example, [4] combined wavelet transformations and stacked autoencoders with long-short-term memory (LSTM) networks, focusing on open-high-low-close (OHLC) bars and technical indicators. In contrast, [34] observed that the performance of intraday stock data, when integrated with technical indicators, improves with the stacking of networks. [7] further presented evidence suggesting that dilated convolutional networks surpass LSTM in performance in various metrics; while [12] contributed by demonstrating that recurrent neural networks exhibit superior performance when applied to limit order book data compared to feedforward networks with lag characteristics. Interestingly, there seems to be a divergence between the statistical modelling literature, represented by works like [8] and [17], and the machine learning literature, exemplified by the contributions of [20] and [5]. This disparity suggests varying perspectives or methodologies in addressing the complexities of financial time series data within different research paradigms.

The landscape of data analysis has seen a surge in the application of artificial intelligence (AI), with a particular emphasis on machine learning and deep learning techniques. This shift involves leveraging learning models constructed based on data rather than predefined models and optimising these models to suit specific application domains. Convolutional neural networks (CNNs) have proven highly effective in tasks like image recognition, while RNNs

* Corresponding author e-mail: makatjanek@ub.ac.bw

demonstrate prowess in modelling time series data. Various iterations of RNN-based models exist, with a key distinguishing factor being their capacity to retain incoming data, as highlighted by [3]. This evolution in AI-based data analysis signifies a move towards more adaptable and data-driven approaches to handling diverse analytical tasks. Generally, vanilla RNNs face challenges in retaining information from past data due to their feedforward learning process in deep learning. Long short-term memory networks address this limitation by facilitating associations between longer input and output data. Unlike traditional RNNs, feedback-based models, such as LSTMs, incorporate multiple gates in their network architecture. These gates enable the model to effectively learn from previous data, allowing the development of a comprehensive model that considers both historical and current information. Consequently, just one pass through the input data is needed; hence, literature, as indicated by [33], suggests that deep learning models, particularly those employing RNNs or LSTM architectures, outperform statistical models such as Autoregressive Integrated Moving Average (ARIMA) models in forecasting time series. This superiority is especially noticeable in addressing long-term prediction challenges, emphasising the effectiveness of deep learning approaches in handling complex temporal dependencies in data.

Even though LSTM has been demonstrated to perform better than ARIMA, it is intriguing to study whether adding more layers of training data to LSTM can enhance its performance even more. To investigate whether incorporating additional layers of training into the architecture of an RNN improves its prediction, this study explores the performance of exponentially smoothed recurrent neural networks (α-RNN). Then, it benchmarks with three RNN architectures: simple RNN, LSTM-RNN, and gated recurrent unit-RNN (GRU-RNN). A simple RNN is chosen because it is the most straightforward RNN architecture, making it relatively easy to understand and implement. The architecture has a simple structure, leading to faster training times than complex RNN architectures. The LSTM networks are designed to address the vanishing gradient problem in RNNs, enabling them to capture long-term dependencies in time series data. This is crucial for stock price forecasting, where past information may significantly impact future prices. Moreover, introducing a memory cell in LSTM helps to store and access information for longer periods, making it effective for modelling sequences with long-term dependencies. Finally, LSTMs offer more flexibility in learning and remembering patterns over time, making them suitable for complex time series data. The GRU is a simplified version of the LSTM, combining some of the benefits of LSTM while reducing complexity. It may be computationally more efficient than LSTM, leading to faster training times. In particular, we would like to perform a behavioural analysis comparing these four architectures when training their models. To do so, we report the results of an experiment in which the performance and behaviour of these RNN architectures are compared. In particular, we are interested in addressing the following research questions:

–Is the prediction improved when time series data are learned from high epochs and batch size?
–How differently do these architectures (α-RNNs, simple RNNs, LSTM-RNNs, and GRU-RNNs) treat input data?
–How fast do these four architectures reach equilibrium?

The main contribution of this study lies in demonstrating the application of recurrent neural networks, specifically a novel class known as exponentially smoothed RNN, within a financial time series modelling framework, as proposed by [29]. The approach involves utilising cross-validation along with statistical diagnostics to identify the optimal architecture within this framework. Statistical tests are employed to assess whether the data is suitable for longer-term forecasting and whether the model needs to account for nonstationarity by characterising stationarity and memory cutoff length. Additionally, we investigate whether the addition of more training layers enhances prediction capabilities in the context of financial time series. Furthermore, a behavioural analysis of the learning procedures employed in training GRU-RNN, LSTM-RNN, and simple-RNN models is conducted. This research contributes to advancing the understanding and application of RNNs, specifically α-RNNs, in financial time series modelling and explores factors influencing model performance and behaviour.

The proposed class of α-RNNs is developed for time series forecasting using numeric data, unlike state-of-the-art RNNs such as LSTMs and gated recurrent units [10], which were primarily designed for speech transcription. i) The α-RNNs solve the gradient problem by using fewer parameters, recurrent units, and samples to achieve similar prediction accuracy. [1] ii) supports both stationary and nonstationary time series. Simple RNNs model stationary time series, whereas GRUs and LSTMs model nonstationary time series, but no hybrid gives the researcher control over which one to deploy; and iii) be mathematically accessible and characterised in terms of well-known concepts in classical time series modelling, rather than appealing to logic and circuit diagrams. Consequently, we demonstrate via a straightforward examination of time series characteristics for α-RNNs how the smoothing parameter directly defines its dynamic behaviour and offers a model that is efficient and more understandable for time series modelling than GRUs and LSTMs. We contend that some more intricate elements, like reset gates and cell memory, present in GRUs and LSTMs but absent in α-RNNs—may be superfluous for our data regarding time series modelling issues in finance. We utilise these attributes in two ways. i) We

[1] Sample complexity bounds for RNNs have recently been derived by [2]. Theorem 3.1 of [2] shows that for a recurrent unit, inputs of length at most b, and a single real-valued output unit, the network requires only $O(\frac{\alpha^4 b}{\varepsilon^2})$ samples to attain a population prediction error of ε. Thus, the more recurrent units are required, the more training data is needed. See Appendix A for this theorem.

decide whether to use a static or dynamic α-RNN model by first applying a statistical test for stationarity; ii) we shorten the training period and the memory needed to store the model, and in general, we expect α-RNN to be more accurate for shorter time series because they require less training data and are less likely to overfit. The latter is a practical point because many finance applications are not inherently large data problems, and the limited amount of data makes a design with fewer parameters more advantageous to prevent overfitting.

1.1 Research Highlights and Key Findings

This study is one of the first to apply the combination of deep learning with statistical methods such as the TBATS; unlike [1] who used ARIMA and exponential smoothing with advanced deep learning techniques such as long short-term memory and convolutional neural networks. Therefore, the highlights and key findings of this study are summarised in Table 1.

Table 1: Financial Time Series Characteristics and Model Suitability

Characteristic	**Simple RNN**	**LSTM**	**GRU**	**α-RNN**	**TBATS-α-RNN**
Volatility Clustering	Limited	Excellent	Good	Good	Excellent
Trend Detection	Weak	Excellent	Good	Excellent	Excellent
Seasonality	Weak	Moderate	Moderate	Excellent	Excellent
Handling Noise	Poor	Strong	Strong	Strong	Strong
Event Shocks (Jumps)	Poor	Moderate to Strong	Moderate	Moderate	Strong
Long-Term Dependencies	Poor	Excellent	Good	Good	Excellent
Training Efficiency	Fast	Slow	Moderate	Moderate	Moderate to Fast

The overall takeaway from Table 1 is that, the TBATS-α-RNN combination stands out as a highly powerful model for forecasting financial time series, thanks to its ability to capture volatility clustering, trends, seasonality, and long-term dependencies, while also handling noise and event shocks effectively. While it may not be the fastest to train, its overall performance across key financial characteristics makes it an excellent choice for advanced forecasting tasks in volatile markets. The LSTM model is also a strong contender, especially for long-term dependencies, but its performance is less robust in capturing seasonality and event shocks compared to TBATS-α-RNN. Simple RNNs, while fast, offer limited performance across most characteristics, making them less suitable for complex financial time series analysis.

The rest of the paper is organised as follows: Section 2 presents the models; empirical results are presented and discussed in Section 3; and Section 4 concludes the study.

2 Models

This section of the study presents the methods and procedures followed. The data used for the experimental analysis is Anglo American Platinum Limited (AMS.JO) stock prices. Anglo American Platinum Limited (JSE: AMS) is the world's largest primary producer of platinum, accounting for about 38% of the world's annual supply. The index is accessed by the Yahoo finance package using the following Python command:

–$Anglo = 'AMS.JO'$
–$start.date = '2018-01-01'$
–$end.date = '2024-02-13'$
–$data = yf.download(Anglo, start = start.date, end = end.date)$.

As [27] has suggested, the index is kept in its original currency to avoid any fluctuations due to exchange rate fluctuations. For ease of exposition, it is assumed that the time series data are univariate and are integrated once, making the predictor endogenous. Without loss of generality, we set (1) $U_h = \omega_h = \phi \in \Re$, (2) $\omega_y = 1$, (3) $b_y = 0$, and (4) $b_y = \mu \in \Re$ for the RNN with one hidden unit $H = 1$. The RNN of the form $F_{\omega,b}(x_t)$ with sequence length ρ is a nonlinear Autoregressive model of order ρ using backward substitution. Because Autoregressive $AR(\rho)$ models are known in time series modelling,

we now show that simple RNNs follow a nonlinear $AR(\rho)$ model with drift in μ and coefficients $\{\phi_i\}_{i=1}^{\rho}$ and this is shown in Equation 1 as

$$\hat{h}_{t-\rho+1} = g(\phi y_{t-\rho+1}), \hat{h}_{t-\rho+2} = g(\phi \hat{h}_{t-\rho+1} + \phi y_{t-\rho+2}), \cdots = \cdots, \hat{h}_t = g(\phi \hat{h}_{t-1} + \phi y_t, \hat{y}_{t+m} = \hat{h}_t + \mu, \tag{1}$$

to yield a nonlinear Autoregressive model in Equation 2 as

$$\hat{y}_{t+m} = \mu + g\left(\phi\left(1 + g\left(L + g\left(\phi\left(L^2 + \cdots + g\left(\phi L^{\rho-1}\right)\right)\right)\right)\right)\cdots\right) y_t. \tag{2}$$

The next step is to recover an AR model by

$$\hat{y}_{t+m} = \mu + \sum_{i=0}^{\rho-1} \phi_{i+1} L^i y_t \tag{3}$$

when the activation has an identity function of $g =: ID$. In Equation 3, $\phi_i = \phi^i$, and thus these Autoregressive coefficients geometrically decay with an increasing lag when $|\phi| < 1$.

2.1 Proposed Exponential Smoothing Recurrent Neural Network

In this situation, the α-RNN(ρ) is almost the same as a simple RNN, except for an extra scalar smoothing parameter α, which provides the recurrent network "long memory," or Autoregressive memory that goes beyond the sequence length. To be clear, we are not suggesting that the α-RNN has an extracellular memory similar to that of LSTMs. To see this, we examine a one-step-ahead univariate α-RNN(ρ) with a fixed smoothing parameter. For every time step say $s = t - p + 2, \cdots, t$ then

$$\hat{y}_{t+m} = \omega_y \hat{h}_t + b_y, \hat{h}_s = g\left(U_h \widetilde{h}_{s-1} + \omega_h y_s + b_h\right), \widetilde{h}_s = \alpha \hat{h}_s + (1+\alpha)\widetilde{h}_{s-1}. \tag{4}$$

Equation 4 consists of the output function, the hidden state update, and the smoothing function. By replacing $\hat{h}_{s-1}$ in the hidden layer with an exponentially smoothed hidden state $\widetilde{h}_{s-1}$, this improves the simple RNN model. The smoothing produces the appearance of infinite memory when $\alpha = 1$. The simple RNN with a small memory of length $p < N$ is recovered for the exceptional situation where $\alpha = 1$. By considering inactivated situation and simplifying the circumstances, we may easily understand this informally. Therefore, Equation 4 is transformed to Equation 5 by setting $b_y = b_h, U_h = \phi \in \Re$, and $\omega_y = 1$, and this application yields

$$\hat{y}_{t+1} = \hat{h}_t = \phi\left(\widetilde{h}_{t-1} + y_t\right) = \phi\left(\alpha \hat{h}_{t-1} + (1-\alpha)\widetilde{h}_{t-2} + y_t\right) \tag{5}$$

with the starting condition in each sequence to be $\hat{h}_{t-1p+1} = \phi y_{t-p+1}$. As this happens, we consider $\rho = 2$ lags in the model so that, $\hat{h}_{t-1} = \phi y_{t-1}$ hence,

$$\hat{h}_{t-1} = \phi\left(\alpha\phi y_{t-1} + (1-\alpha)\widetilde{h}_{t-2} + y_t\right) \tag{6}$$

and the model can be written in the following simpler form

$$\hat{y}_{t+1} = \phi_1 y_t + \phi_2 y_{t-1} + \phi(1-\alpha)\widetilde{h}_{t-2} \tag{7}$$

with AR weights being $\phi_1 = \phi$ and $\phi_2 = \alpha\phi^2$. Upon closer inspection, we observe that Equation 7 has a third component on the right side that disappears when $\alpha = 1$. This gives the model indefinite memory because $\widetilde{h}_{t-1}$ depends on y_1 which is the first observation in the time series, not just the first observation in the sequence. To see this, we unroll the recursion relation in the exponential smoother; then,

$$\widetilde{h}_{t+1} = \alpha \sum_{s=0}^{t-1} (1-\alpha)^s \hat{h}_{t-s} + (1-\alpha)^t y_t \tag{8}$$

where we used the property that $\widetilde{h}_1 = y_1$. It is often convenient to characterise exponential smoothing (ES) by the half-life, the number of lags needed for the coefficient $(1-\alpha)^s$ to equal a half, which is $s = \frac{1}{\log_2(1-\alpha)}$. We then use partial Autocorrelations to characterise the model's memory to gain further insight.

2.2 Partial Autocorrelation Function for AR Model

We consider the AR model with the additive white noise of the form $y_t = \hat{y}_t + \varepsilon_t, \varepsilon \tilde{N}(0, \sigma_n^2)$, which, when combined with covariance stationary time series data, bears a signature that makes it possible to establish its order, i.e., ρ. This signature, provided by partial Autocorrelations, encodes the model's memory. Given a time series y_t, the partial Autocorrelation of lag k, denoted by $\phi_{k,k}$, is the Autocorrelation between y_t and y_{t+k} with linear dependence of y_t on y_{t+k} through y_{t+k-1} removed. Equivalently, the Autocorrelation between y_t and y_{t+k} is not accounted for by lags 1 through $k-1$ inclusive [40] and it can be shown that

$$\rho_k = \text{Cov}(y_t y_{t-1}) = \text{E}\,(y_t - \mu_y)(y_t - \mu_y). \tag{9}$$

Given the AR(ρ) process: $y_t = \Phi_1 y_{t-1} + \cdots + \Phi_\rho y_{t-\rho} + \varepsilon_t$, we now show that $\rho_k = \sum_{j=1}^{\rho} \Phi_j \rho_{k-k} + \text{E}[\varepsilon_t y_{t-k}], \forall k = 0,1,2,\cdots$. But, $\text{E}[\varepsilon_t y_{t-k}] = 0, \forall k = 1,2,3$. Therefore, $\rho_k = \sum_{k=1}^{\rho} \Phi_j, k = 1,2,3\cdots$. For a stationary time series, a theoretical partial Autocorrelation function (PACF) is given by

$$\rho_k = \begin{cases} 1 & \text{if } k = 0 \\ \sum_{j=1}^{\rho} \Phi_j \rho_{k-j}, & k = 2,3,\cdots,\rho. \end{cases} \tag{10}$$

For a given sample y_t, let there $\bar{y}$ be a sample mean. Then, at lag one, the sample Autocorrelation of y_t is given by

$$\hat{\rho}_1 = \frac{\sum_{t=2}^{T}(y_1 - \bar{y}_t)(y_{t-1} - \bar{y}_t)}{\sum_{t-1}^{T}(y_t - \bar{y}_t)^2}. \tag{11}$$

Under some general conditions, Equation 11 is a consistent estimator for ρ_1. However, if $y_t \sim i.i.d$ (herein referenced independently and identically distributed) sequence, and $E(y_T^2) < \infty$, then [37] indicated that ρ_1 is asymptotically normal with mean zero and variance $\frac{1}{T}$. This result, in practice, tests the hypothesis $H_0 : \rho_1 = 0$ versus $H_1 : \rho_1 \neq 0$. A test statistic is the usual t ratio, which is $\sqrt{T\hat{\rho}_1}$, and it is asymptotically a standard normal distribution. In general, the lag-l sample Autocorrelation of y_t is now given by

$$\hat{\rho}_l = \frac{\sum_{t=l+1}^{T}(y_t - \bar{y}_t)(y_{t-1} - \bar{y}_t)}{\sum_{t=1}^{T}(y_t - \bar{y}_t)^2}. \tag{12}$$

In finite samples, $\hat{\rho}_l$ is a biased estimator of ρ_l, and $\frac{1}{T}$ is the one that causes this bias when the sample size is small, that is, $T < 30$, and to overcome this problem in financial applications, T must be large, that is, $T > 30$. In this case, ρ_l can now be plotted against time. [22] revealed that if the spikes of plotted PACF fall beyond the control bands of the plot, this indicates highly correlated time series at different lags, and [37] exhibited that there is no correlation if $\rho_l = 0$, for $l > 0$. Nonetheless, financial applications often necessitate joint testing if Autocorrelations of the return series are zero, i.e., $\rho_k = 0$. Finally, [25] proposed the following portmanteau statistic

$$Q^*(m) = T\sum_{l}^{m} \hat{\rho}_l^2, \tag{13}$$

as a test statistic for the following hypothesis $H_0 : \rho_1, \cdots, \rho_m = 0$ versus $H_1 : \rho_1, \cdots, \rho_m \neq 0$ under the assumption that a time series is an i.i.d. sequence with certain moment conditions. Henceforth, [9] showed that Equation 13 is asymptotically a chi-squared random statistic with m degrees of freedom that increases the power of a test. Because of finite samples, [25] modified Equation 13 to

$$Q(m) = T(T+2)\sum_{l}^{m} \frac{\hat{\rho}_l^2}{T-l}. \tag{14}$$

Reject the null hypothesis if the calculated probability value is less than the observed probability value and conclude that the series is highly correlated. The PACF of the RNN(ρ) can be used to determine the lower bound on the sequence length in an α-RNN(ρ). To see this, we first show that the partial Autocorrelation of the α-RNN(ρ) is time-independent and has a sharp cutoff after ρ lags if $\alpha = 1$; and [13] showed that this exhibits $\widetilde{\tau}_s = 0, s > \rho$ if $\alpha = 1$. If $\alpha \in (0,1)$, the α-RNN(ρ) has nonzero partial Autocorrelations at lags beyond the sequence length. As in the Autoregressive model, the partial Autocorrelation is used to identify the order of the α-RNN model from the estimated PACF and hence determine the sequence length in the α-RNN, which is guaranteed to have at least the same order for $\alpha \in (0,1]$. This is evident from an extra term in Equation 7 that contains α. Figure 1, which displays the fitted correlogram from data produced by an α-RNN(3) with additive white noise, provides more information. When $\alpha \in (0,1)$, the memory is always longer than

the sequence length 3. The memory of the model grows as α gets closer to zero but has no memory when $\alpha = 0$. The sequence length ρ can be ascertained from the fitted PACF of every covariate. This is according to Theorem 2.2 of [11] and the characteristics of α-RNN(ρ). Furthermore, it is recommended that the prediction horizon should not be larger than the maximum order of statistically significant partial Autocorrelation.

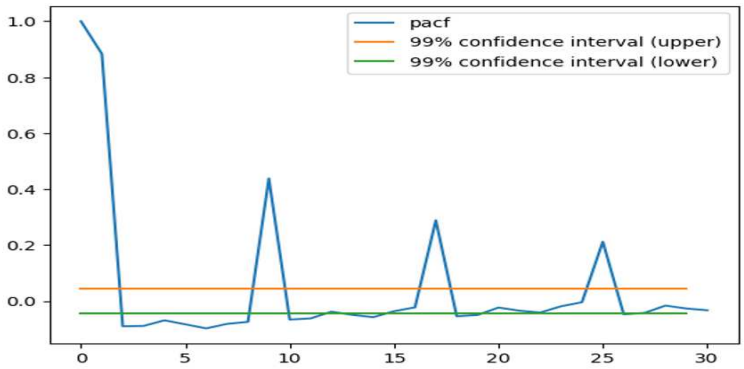


Figure 1: Fitted PACF for α- RNN(3). Source: Authors own Computations

2.3 Testing Stability and Training of RNN Models

Time series modelling also emphasises model "stability," the feature that causes prior random disturbances to dissipate in the model, and the effect of lagged data becomes less significant to the model output as the lag increases. Therefore, Theorem 2 of [13] demonstrates that the stability of α-RNN model is completely dictated by the hidden state's activation function, with $|g(\cdot)| < 1$. Suppose that the form has an invertible nonlinear function of the lag operator $\Phi(\beta)$, then,

$$y_t = \Phi^{-1}(B)\varepsilon_t = \left(1 - g\left(\phi g\left(\phi g\left(\phi g\left(\cdots\right)\right) + \cdots + \phi\beta^2\right) + \phi\beta\right)^{-1}\varepsilon_t\right), \tag{15}$$

where, without loss of generality, we have again conveniently set $\omega_x = U_h = \phi$, $\omega_y = 1$, and $b_h = b_y = 1$. The RNN now becomes stable if the condition $|g(x) < 1, \forall x|$ is met. We now employ a stochastic gradient descent with α as an extra parameter to train the α-RNN model. In backpropagation, resolving the vanishing gradient problem is one of the benefits of smoothing the hidden state. A stationary time series is the only scenario in which it is possible to predict the independence of time. Although restrictive, it suggests that the technique's efficacy can be predicted by a straightforward statistical analysis of a time series. Moreover, if the data exhibits covariance-stable behaviour, the α-RNN will maintain the stationarity of the partial Autocorrelation structure. This eliminates the need for intricate architectures like GRUs and LSTMs, which are frequently motivated exclusively by the vanishing gradient problem. Using a dynamic version of exponential smoothing, we extend the model to nonstationary time series, which exhibit dynamic partial Autocorrelation. It is noteworthy that dynamical time series models, which are perfect for nonstationary time series data utilising dynamic exponential smoothing, may be created with RNNs. Depending on the input and hidden state, the α_t-RNN models can update α; however, a recurrent layer is a helpful alternative. We may simulate the smoothing parameter $\hat{\alpha}_t \in [0,1]$ by assuming a hidden state vector $\hat{h}_t \in \Re^H$ and provide a sorted time series as output:

$$\widetilde{h}_t = \hat{a}_t \cdot \hat{h}_t + (1 - \hat{\alpha}_t) \cdot \widetilde{h}_{t-1}, \tag{16}$$

where $\cdot$ denotes the Hadamard product between vectors, and this smoothing is a vectorised form of the above classical setting. Only here, we note that the i^{th} component of the hidden variable does not change when $(\hat{\alpha}_t)_i = 1$. Conversely, the filtered variable that was previously filtered is overlooked. As this happens, the i^{th} component of the hidden variable becomes outdated when $(\hat{\alpha}_t)_i = 1$, returning the filtered hidden variable to its initial value. Equation 16 takes the smoothing into account by employing a convex combination of the previously smoothed hidden variable and the present hidden covariate to update long-term memory. The hidden variable is obtained via the semi-affine transformation as shown in Equation 17; where

$$\hat{h}_t = g\left(U_h\widetilde{h}_{t-1} + \omega_h x_t + b_h\right) \tag{17}$$

which in turn depends on the previous smoothed hidden variable. Substituting Equation 17 into Equation 16, this gives a function of $\widetilde{h}_{t-1}$ and x_t as

$$\widetilde{h}_t = g\left(\widetilde{h}_{t-1}, x_t; \alpha\right) = \hat{\alpha}_t \cdot g\left(U_t \widetilde{h}_{t-1} + \omega_h x_t + b_h\right) + (1 - \hat{\alpha}_t) \cdot \widetilde{h}_{t-1}. \tag{18}$$

Updates from the input x_t are not received by the smoothed hidden variable $\widetilde{h}_t$ when $\hat{\alpha}_t = 0$. The smoothing parameter can be understood as the sensitivity of a smoothed hidden state to the input x_t. This is because the hidden variable acts as a nonlinear Autoregressive series when $\widetilde{\alpha}_t = 1$. The tricky part is figuring out how to calculate the dynamically required amount of error correction. With the recurrent layer defined by the following weights $(\omega_\alpha, U - \alpha, b_\alpha)$ and biases, we now solve this problem by learning $\hat{\alpha} = F_{\omega_\alpha, U_\alpha, b_\alpha} x_t$ from the input variables, just like in GRUs and LSTMs. This is to ensure that

$$\hat{h}_t = \sigma_s\left(U_\alpha, \widetilde{h}_{t-1} + \omega_\alpha x_t + b_\alpha\right) \tag{19}$$

where, $\sigma_s(x) := \frac{1}{1+\exp^{-1}}$. Again, the one-step-ahead forecast of the smoothed hidden state $\widetilde{h}_t$ is the filtered output of another simple RNN with weights and biases as (ω_h, U_h, b_h).

3 Empirical Results and Discussion

This section describes an empirical analysis of the study using time series data to test different RNN models. Unless specified otherwise, all models are implemented in TensorFlow version 2.15 developed by [30]. Time series cross-validation is carried out on independent training, validation, and test sets. Each set reflects a contiguous sample period to avoid look-ahead bias, with the test set including the most recent observations.

3.1 Exploratory Data Analysis

Figure 2 shows plots of the closing stock prices. While 2(c) and 2(d) demonstrate that the data is non-normal, the kernel density plot in Figure 2(b) indicates that the distribution of closing stock prices has a small rightward bias. Furthermore, it is noteworthy that in Figure 2(a), seasonality is paired with certain positive and negative patterns. These moments of volatility clustering, according to [21], are the result of events like the COVID-19 pandemic and the market panic caused by the European debt crisis in 2009–2010. As Figure 2(a) shows, this financial market likewise has the most concentrated stock return losses. Moreover, this figure represents a potential benefit when conditional heteroscedasticity is considered. Two important factors are highlighted here: the reason for weight loss; and the erratic nature of weight loss. The latter contends that irregular shocks in the actual business sector have a greater influence on future volatility, whereas the former contends that downturn volatility follows these shocks rather than significant losses or gains. However, [44] stated that the South African Reserve Bank's (SARB) implementation of a contractionary monetary policy is what produced these significant losses.

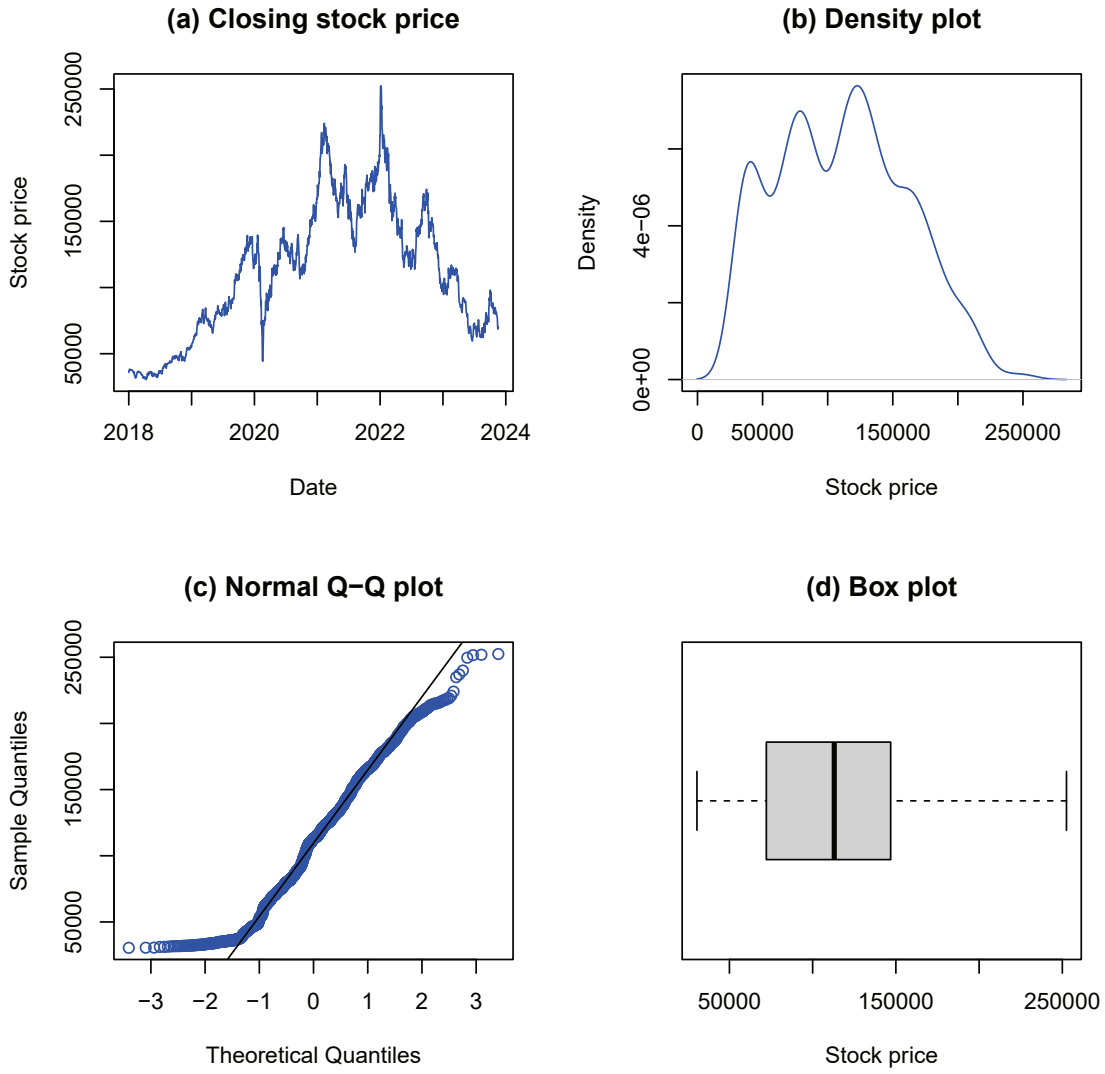


Figure 2: Plots of the closing stock prices.

3.2 Short-term Forecast of Closing Stock Prices

We use "rolling" forecasts to set up the training, validation, and testing sets for the m-step ahead prediction. For each input sequence, the former takes lags from $t-p+1,\cdots,t$ and sets the target variables (response) to the observation, which y_{t+m} is increased repeatedly till the conclusion of each test set. This is because the direct forecast approach is not only faster, but it is also better at identifying longer-term trends and seasonal variations. Every element in the input sequence is either a vector or a scalar, and the target variable in our empirical analysis is a scalar. We create a fully connected RNN by using a Keras function as in the work of [23], and proceed to set the number of units $H \in \{5,10,20\}$ and L_1 regularisation via a cross-validation process. Then, $\lambda \in \{0,10^{-3},10^{-2}\}$ is employed using the hidden layer that is activated by the tanh function. Lastly, the recurrence weights are initialised using an orthogonal matrix for stability, ensuring that the initial bound of the eigenvalues' absolute value is unity; see for instance, [19]. The non-recurrent weight matrices are initialised using the Glorot and Bengio uniform approach of [15]. The gated recurrent unit is estimated using the Keras 2.15.0 version by first applying the reset gate to the hidden state and then performing matrix multiplication. The recurrence layer employs tanh activation functions, whereas the remaining gates use sigmoid activation function. To create our proposed $alpha$-RNN model, Keras's exponential smoothing layer combines an elementary RNN with the exponential smoothing layer. More complex versions of the basic RNN were investigated by [35], which stacked many recurrent layers and used

dual attention and dilation to capture multi-scale effects and a method suitable for high-dimensional datasets, respectively. These differences are significant.

Each architecture is trained using a mini-batch size 32 and the stochastic gradient descent algorithm with 0.01 as the learning rate and 0.9 as momentum. Finally, "patience" is set to range from 25 to 75 and a minimum absolute difference between 10^{-8} and 10^{-6} to enable early stopping [42] with 1200 epochs. The performance of the four recurrent neural networks is measured in a Bayesian framework, thereby quantifying aleatoric uncertainty. Our dataset consists of daily closing prices for Anglo-Americans with a total sample of 1529. The closing stock price time series is used because it displays both long-term trends and short-term cyclical patterns. To eliminate look-ahead bias in the test data, the predictor in both the training and test sets is normalised using only the moments of the training data. When testing for stationarity, the augmented Dickey-Fuller (ADF) test failed to reject the null hypothesis and conclude that the data is nonstationary. These results are presented in Table 2.

Table 2: Stationarity Test

ADF tests 1%	5%	10%	p-value	
-1.7134	-3.4358	-2.8639	-2.568	0.4242

[13]'s empirical analysis also shows the same results of nonstationary stock price data. This author used daily adjusted close prices of IBM for the period of January 3, 2006, to December 29, 2017. At the same time, the current study uses AMS.JO closing stock prices from January 02, 2018, to February 13, 2024. Based on the PACF in Figure 3, we select a sequence length of $\rho = 1$ and then execute a rolling of five steps and a ten step ahead forecast. This corresponds to a one/two-week projection, as a week consists of five business days. The Bayesian RNNs of [14] are now implemented using the Blitz PyTorch module. Cross-validation is performed over $\{5, 10, 15\}$ hidden neurons and λ_1 is varied over the set of $10^{-3}, 10^{-2}, 10^{-1}, 0\}$ just as in the work of [13]. For each model, the optimal λ_1 is zero. Table 3 compares the performance of the Bayesian recurrent networks by five-day-ahead and ten-day-ahead forecasts. The next day, 99% confidence intervals are back-tested over the test set and found to be comparable across different networks. They consistently underestimate the empirical confidence intervals and more closely correspond to the 90% empirical confidence intervals. The α-RNN exhibits the most accurate confidence intervals—only 5% of the observations fell outside the 99% confidence intervals.

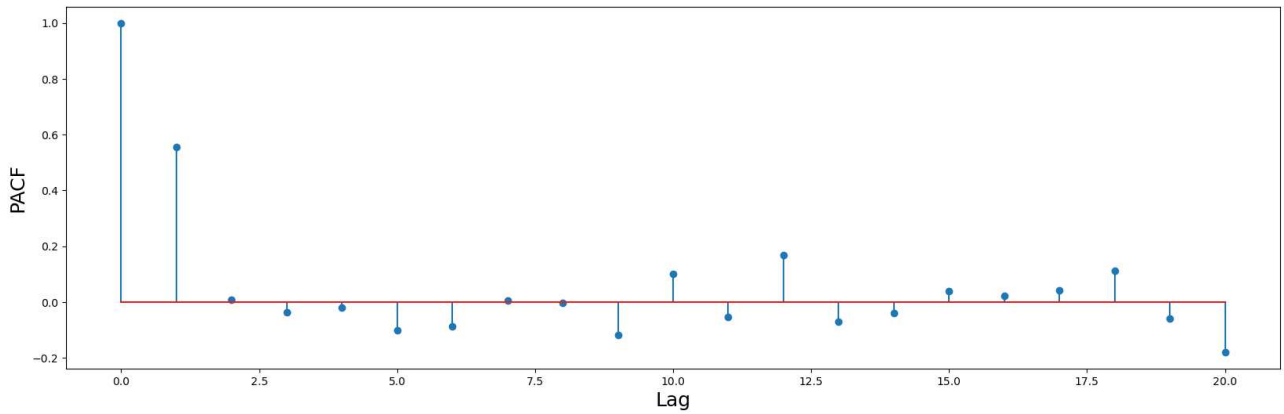


Figure 3: PACF for Selection of Alpha-RNN Order

Table 3: Five-step and ten-step ahead rolling forecasts are compared for various Bayesian recurrent networks.

Architecture	H	RMSE	MSE	MAE	Coverage	Parameters	Prediction time
Five-day ahead							
RNN	10	0.050396	0.00254	0.039958	0.879	17801	76.55.5
α-RNN	5	0.030063	0.000904	0.030063	0.949	30651	20.30.0
GRU	15	0.055588	0.00309	0.044819	0.859	53901	25.32.8
LSTM	10	0.031557	0.000996	0.024724	0.901	31901	40.31.2
Ten-day ahead							
RNN	10	0.237472	0.056393	0.508188	0.687	85	92.242
α-RNN	5	0.087185	0.0082475	0.506706	0.809	116	80.441
GRU	15	0.176856	0.031278	0.519653	0.779	315	100.142
LSTM	10	0.47125	0.02221	0.70556	0.679	240	80.276

For both five and ten days ahead, the α-RNN has the smallest MSE, RMSE, and MAE with coverage of 94.9% five days ahead and 80.9% ten days ahead. The prediction time of the RNN is less than that of the GRU but despite exhibiting fewer trainable parameters, it is slower than the LSTM. The LSTM takes longer to train but is faster to predict due to cellular memory-catching implementation details beyond the scope of this article. Figure 4 further confirms the reported results that our proposed $\alpha-$RNN has the lowest forecasting errors, making it superior to the other three RNN architectures. For each model performance, the reader is referred to Appendix B.

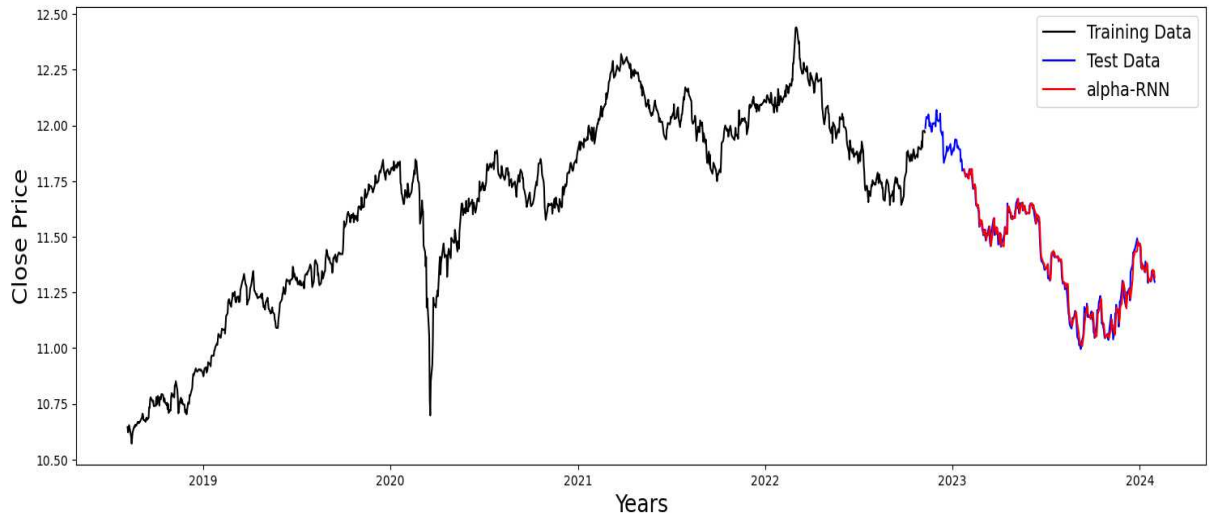


Figure 4: Fitted Conditional Mean for Closing Stock Prices.

3.3 Combined Forecasting Model

Anglo American's closing stock price is split into the training, validation, and test data sets. We therefore proceed and fit a TBATS model, and the best TBATS model is automatically selected by using the TBATS Python package for closing prices. The parameters of a fitted model are $\hat{\lambda} = 0.2603, \hat{\alpha} = 1.80744,\ \hat{\beta} = -0.374124$ and the damping parameter is $\phi = 0.7120$. Using the minimum score combining proposed by [39], we now combine α-RNN with the TBATS. Based on the estimates of our combined model, an out-of-sample analysis evaluates the ability of deep learning methods coupled with Box-Cox transformations, Fourier representations with time-varying coefficients, and ARMA error correction.

Table 4: Five-step and ten-step ahead rolling forecasts for TBATS-α-RNN.

Architecture	H	RMSE	MSE	MAE	Coverage	Parameters	Prediction time
Five-day ahead							
TBATS-α-RNN	5	0.00132	0.0035	0.00363	0.987	26812	18.37.1
Ten-day ahead							
TBATS-α-RNN	5	0.00171	0.00521	0.0606	0.8876	100	77.421

The results of the rolling forecasts for the TBATS-α-RNN model indicate that it performs well for short-term predictions (five-step ahead). For this horizon, the model achieves low errors (RMSE, MSE, MAE), high coverage (98.7%), and reasonable prediction time (18.37 seconds). However, as the forecast horizon extends to ten steps, the accuracy of the model decreases, with higher errors (RMSE, MSE, MAE) and lower coverage (88.76%). The prediction time also increases significantly to 77.42 seconds for the ten-step forecast. Overall, the model is highly effective for short-term forecasting but shows a decrease in performance as the forecast horizon lengthens. The possible reason of the decline in performance as the horizon increases is because the estimated ρ for our exponential smoothing recurrent neural network is 1 and [31] has recommended that the prediction horizon should not be larger than the maximum order of statistically significant partial Autocorrelation. The TBATS-α-RNN model proves effective for short-term predictions (five-step ahead), which aligns with the fast-moving and volatile nature of the commodities market, particularly in the context of platinum prices and production. This allows Anglo American Platinum (AAP) to make highly accurate predictions for immediate operational decisions, such as inventory management, production planning, and market positioning. As the prediction horizon extends (ten-step ahead), the performance of the model deteriorates, with increased errors and reduced coverage. This suggests that for longer-term planning, AAP should exercise caution when relying on such models and perhaps combine them with other strategies like expert judgment, market insights, and macroeconomic forecasting models to address uncertainty over longer time frames. The recommendation not to exceed the maximum order of statistically significant PACF is particularly relevant in this case, as it highlights the importance of staying within the model's optimal performance range.

Figure 5 presents the forecasting performance of the combined model. This model's key performance indicators are significantly lower than the others. We employ three error metrics in this stock price case study. Each model's performance is assessed using these metrics, and the results are summarised in Table 4. This table leads to a few speculative inferences. The findings demonstrate that combining point forecasts from various models enhances the predictive capability of the model. The best model is given in Table 4.

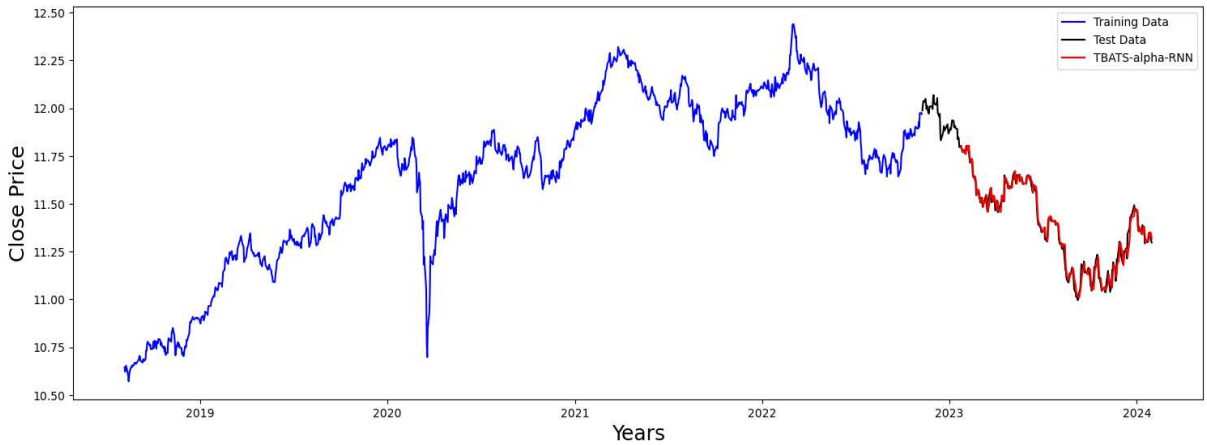


Figure 5: Fitted Conditional Mean for Combined TBATS-α RNN

3.4 Discussion of Results

Forecast combinations offer a practical method for synthesising the information acquired from data by separate models. Its success is ascribed to its capacity to decrease individual model misspecification, its robustness against structural

breakdowns, and the diversification improvements that result from combining forecasts derived under multiple assumptions [26]. In this study, we set a two-stage procedure where in the first stage we train four models, namely RNN, α-RNN, GRU-RNN, and finally the LSTM-RNN, and perform the comparative analysis. This was to find the best-fitting model to the data before combining it with the TBATS model. For the five-day-ahead forecast, the RNN shows moderate RMSE (0.050396) and MAE (0.039958) with a coverage of 87.9%. Its prediction time is 76.55 seconds with 17,801 parameters. α-RNN performs better in RMSE (0.030063) and MSE (0.000904) with high coverage (94.9%) and a faster prediction time (20.30 seconds), though it has the highest number of parameters (30,651). The GRU-RNN has the highest RMSE (0.055588) and lowest coverage (85.9%) among the five-day-ahead models. The prediction time is relatively fast at 25.32 seconds. Moreover, LSTM performs well with a low RMSE (0.031557), decent MAE (0.024724), and coverage of 90.1%, but with a slower prediction time of 40.31 seconds. With the ten-day ahead forecast, the RNN has significantly higher RMSE (0.237472) and MAE (0.508188) compared to the five-day ahead forecast, with low coverage (68.7%) and prediction time of 92.242 seconds. The α-RNN outperforms the others in this set, with the RMSE of 0.087185, the MSE of 0.0082475, high coverage (80.9%), and the lowest prediction time (80.441 seconds). The GRU-RNN has an RMSE of 0.176856, slightly better coverage (77.9%) than RNN, and a prediction time of 100.142 seconds. The LSTM has a high RMSE (0.47125) and the lowest coverage (67.9%) in the ten-day ahead forecast. The prediction time is similar to α-RNN (80.276 seconds) but has significantly more parameters. In summary, the α-RNN appears to consistently perform well across five-day and ten-day forecasts with lower RMSE, higher coverage, and relatively quick prediction times.

The RNN struggles with higher RMSE and lower coverage, particularly for ten-day forecasts, and the LSTM shows good performance in the five-day forecast but underperforms in the ten-day forecast. Finally, the GRU-RNN performs decently across both horizons but does not outperform α-RNN. The performance differences between models in five and ten-day forecasts have direct the following economic and stock market implications:

- **Short-term vs. Long-term Forecasting:** Investors relying on RNN or LSTM models for longer-term (ten-day) forecasts face increased risk due to higher RMSE and lower coverage. This may result in sub-optimal trading decisions, particularly in volatile markets.
- **Model Selection for Trading Strategies:** α-RNN consistently outperforms others, making it more reliable for both short- and long-term predictions. Traders using this model can better manage risk and optimize their portfolio management strategies.
- **Computational Trade-offs:** While GRU performs decently, it still lags behind α-RNN, suggesting that traders seeking a balance between prediction accuracy and computational cost may prefer α-RNN for efficient decision-making.

Moreover, [32] showed that popular deep learning architectures for stock market forecasting include the gated recurrent unit and long short-term memory models. According to several studies, forecasting that takes into account the emotion of financial news might outperform stock attributes alone. Therefore the study of these authors objectively evaluated the importance of using financial news feelings in stock market forecasting by comparing the normalised performances of LSTM and GRU for stock market forecasting under identical settings. However, our study combined the TBATS with α-RNN.

Our approach is different from that of [18] where these authors combined the ARIMA model with the LSTM. The advantage of our approach is that TBATS handles complex seasonal patterns, especially in time series with multiple seasonalities like the closing stock price of Anglo-American used in this study. While the α-RNN captures nonlinear relationships and long-term dependencies. This fusion of models improves the overall forecasting accuracy by addressing both seasonal and nonlinear elements in stock prices. [41] on the other side performed the comparative analysis to see if the gated unit architecture has a positive impact and whether LSTM is still better than RNN in flood forecasting work. These authors used the Bayesian optimisation algorithm (BOA) to optimise the hyperparameters while in this study we only use Bayesian architecture for the proposed model and enable early stopping to avoid overfitting of the parameters. This is because training for more epochs allows the model to learn complex patterns more deeply, potentially improving accuracy. However, overfitting can occur if trained too long, where the model performs well on training data but poorly on new data. Larger batches can stabilise the training process and lead to faster convergence but may miss finer details in the data patterns, which smaller batches could capture more effectively. [43] in their study proposed an effective ES strategy that consistently detects near-peak performance in various computational imaging tasks and deep image prior (DIP) variants. Simply based on the running variance of DIP intermediate reconstructions, their early stopping method not only outpaces the existing ones—which only work in very narrow regimes but also remains effective when combined with methods that try to mitigate overfitting.

To answer the second question of this study, we find that the architectures used treat the closing stock price of Anglo-Americans differently. Simple-RNN processes inputs sequentially, but struggles with long-term dependencies due to vanishing gradients. The LSTM-RNN uses gates (input, forget, output) to manage long-term dependencies, making it effective for sequential tasks. While the GRU-RNN is similar to LSTM but with fewer gates (update, reset), making it

computationally lighter. Finally the α-RNN: Enhances memory retention by incorporating exponential smoothing, aiding in capturing both short- and long-term patterns. Finally, we found that the speed at which these four architectures reach equilibrium, meaning the point where they stabilise in learning, varies based on their internal mechanisms. With simple-RNN, this architecture reaches equilibrium relatively quickly but struggles with long-term patterns due to vanishing gradients. Then, the LSTM-RNN is found to be slower to reach equilibrium because of complex gating mechanisms that handle long-term dependencies effectively, but, the GRU-RNN is faster than LSTM, as it has fewer gates but still captures long-term dependencies. While we found that α-RNN is faster due to its exponential smoothing nature, but depends on the specific application of the smoothing parameter. For more readings the reader is directed to the work of [28].

Nonetheless, our results show that combining the TBATS with α-RNN is a robust framework that potentially outperforms simple time series models, providing more accurate stock price predictions over various forecast horizons. This could be particularly be useful for investors seeking precise short and long-term forecasts in highly volatile markets. Investors can mitigate risks associated with volatile markets by leveraging the precise predictions from this combined model, which improves capital allocation and reduces unexpected losses. This combination enhances the efficiency of algorithmic trading strategies by accurately predicting price movements, reducing transaction costs, and enhancing profit margins. In addition to improved prediction accuracy and risk management, combining TBATS with α-RNN offers several other benefits: (1) handling complex seasonality. The TBATS is effective at modelling multiple seasonal cycles, which is crucial for stock prices influenced by recurring market conditions. (2) Flexibility in Modeling. Our combined approach is versatile for various financial data, accommodating both linear and nonlinear patterns. (3) The method can be applied to different forecast horizons, making it useful for both short and long-term financial planning. (4) By combining these models, the system can generalise better to unseen data. Our findings are similar to that of [36] who explored time series forecasting using Prophet (which has similarities with TBATS) combined with machine learning models to improve accuracy. While [35] used an advanced hybrid of exponential smoothing models (similar to TBATS) with neural networks to win the M4 forecasting competition.

4 Limitations, Recommendations and Conclusion

In this section of the study, we present the limitations, recommendations and conclusion.

4.1 Limitations

While our study introduces an innovative combination of TBATS and α-RNN models for forecasting financial time series, several potential limitations exist. (1) Complexity and interpretability. The combination of TBATS and α-RNN introduces significant complexity, affecting a model's interpretability. Understanding how each component (seasonality, memory decay, and noise) influences the predictions is challenging, especially when trying to explain the results to non-technical stakeholders. (2) Data sensitivity. The performance of our model is highly sensitive to the quality and quantity of the data. If the financial time series used in the study has noise, missing data, or anomalies, it could skew the predictions and reduce the model's robustness. (3) Computational resources. The use of deep learning models alongside statistical methods like TBATS is computationally expensive, especially with large datasets. The training and forecasting process requires significant computational power, which limits the scalability of the model. (4) Limited consideration of external factors. While our study combines TBATS and α-RNN, it is still limited by not incorporating other external variables such as macroeconomic indicators, geopolitical factors, or sentiment from financial news in real-time. These factors significantly influence financial markets and might improve forecasting accuracy when included. (5) Forecast horizon limitations. As observed in our analysis, the model performs well in short-term forecasts but the performance deteriorates as the forecast horizon extends. This limitation may hinder the model's applicability for longer-term strategic planning or investment decisions. (6) Model validation. Given the novelty of combining TBATS with α-RNN, extensive validation across different financial markets and datasets is required to establish the generalisability of the findings. If the model works well in one market (e.g., platinum) but not in others, its broader applicability might be limited.

4.2 Recommendations

To address the limitations of our study, several recommendations are proposed. First, enhancing interpretability can be achieved by leveraging explainable AI techniques, such as SHAP or LIME, to visualise and quantify the contribution of each model component. Improving data quality and robustness involves implementing pre-processing techniques like

outlier detection, imputation for missing data, and expanding datasets with diverse historical financial data. To optimise computational resources, cloud-based platforms and model optimisation techniques, such as pruning or quantisation, should be utilised. Incorporating external factors, including macroeconomic indicators, geopolitical data, and sentiment from real-time financial news, will significantly improve forecasting accuracy. To address forecast horizon limitations, the model should focus on short to medium-term projections and explore hybrid models for long-term forecasting. Expanding model validation is essential, requiring tests across diverse datasets, asset classes, and market conditions, alongside stress testing to evaluate robustness. Finally, continuous model updates and real-time monitoring are necessary to adapt to changing market conditions, with periodic retraining to maintain relevance and accuracy. These strategies aim to enhance the utility, reliability, and generalisability of the proposed TBATS-α-RNN model in financial forecasting.

4.3 Conclusion

The study focused on evaluating five models within a Bayesian framework in quantifying aleatoric uncertainty of financial time series data and, in particular, stock prices. Despite the computational challenges, the study of daily closing prices for Anglo-Americans has shown patterns indicative of both long-term trends and short-term cyclical behaviours common in stock market dynamics. Through various methodologies, including lagged adjusted closing price observations and normalization techniques, the study has addressed biases and paved the way for accurate forecasting. The confirmation of nonstationarity in our data aligns with previous empirical analyses, emphasizing the complexity inherent in financial datasets. The α-RNN model showed precise confidence interval estimations, highlighting its effectiveness in capturing the uncertainties within the data. Additionally, integrating traditional time series methods, such as TBATS, with deep learning approaches, has provided a comprehensive framework for enhanced forecasting performance. Through out-of-sample analyses and the application of various error metrics, we have demonstrated the effectiveness of combining multiple models, leading to improved predictive capabilities. This study demonstrates the value of using Bayesian methodologies in addressing aleatoric uncertainty within financial time series data, offering valuable insights for practitioners and researchers alike in quantitative finance.

Acknowledgments

The authors thank numerous people for their helpful comments on this paper.

Competing Interests

The authors declare that they have no competing interests

Declaration of generative AI

The authors hereby disclose that no generative AI tools were used to compile this research work.

References

[1] Ade, M. (2023). Hybrid Approaches in Time Series Forecasting: Combining Statistical Models and Deep Learning for Financial Predictions. *Unpublished work*. Available at https://www.researchgate.net/profile/Martins-Ade/publication/384665223_Hybrid_Approaches_in_Time_Series_Forecasting_Combining_Statistical_Models_and_Deep_Learning_for_Financial_Predictions/links/67041c279e6e82486f0a478c/Hybrid-Approaches-in-Time-Series-Forecasting-Combining-Statistical-Models-and-Deep-Learning-for-Financial-Predictions.pdf. (Accessed 2025 January, 14)

[2] Akpinar, N.-J., Kratzwald, B., and Feuerriegel, S. (2019). Sample Complexity Bounds for Recurrent Neural Networks with Application to Combinatorial Graph Problems. Available at https://doi.org/10.48550/arXiv.1901.10289. (Accessed on 2024 February 26).

[3] Alizadegan, H., Rashidi Malki, B., Radmehr, A., Karimi, H., and Ilani, M. A. (2024). Comparative Study of Long Short-term Memory (LSTM), Bidirectional LSTM, and Traditional Machine Learning Approaches for Energy Consumption Prediction. *Energy Exploration and Exploitation*, 01445987241269496. https://doi.org/10.1177/0144598724126949

[4] Bao, W., Yue, J., and Rao, Y. (2017). A Deep Learning Framework for Financial Time Series using Stacked Autoencoders and Long-Short-Term Memory. *PloS one*, 12(7): e0180944. https://doi.org/10.1371/journal.pone.0180944
[5] Bayer, J. S. (2015). Learning Sequence Representations (Doctoral Dissertation), *Technische Universität München*. Available at https://mediatum.ub.tum.de/1256381. (Accessed on 2024 February, 23).
[6] Borovkova, S., and Tsiamas, L. (2019). An Ensemble of LSTM Neural Networks for High-frequency Stock Market Classification. *Journal of Forecasting*, 38(6): 600-619. https://doi.org/10.1002/for.2585
[7] Borovykh, A., Bohte, S., and Oosterlee, C. W. (2017). Conditional Time Series Forecasting with Convolutional Neural Networks. Available at https://doi.org/10.48550/arXiv.1703.04691 (Accessed on 2024 February 26)
[8] Box, G. (2013). Box and Jenkins: Time Series Analysis, Forecasting, and Control. In*A Very British Affair: Six Britons and the Development of Time Series Analysis During the* 20^{th} *Century*, 161-215: *Springer.* https://doi.org/10.1057/9781137291264_6
[9] Chen, M.Y., (2013). Financial Time Series and their Characteristics. *National Chung Hsing University*. Available at http://web.nchu.edu.tw/~finmyc/timserp.pdf. (Accessed on 2024 February, 26)
[10] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modelling. In *NIPS 2014 Workshop on Deep Learning*, December 2014. Available at https://courses.physics.illinois.edu/cs546/sp2018/Slides/Feb20_Chung.pdf. (Accessed on 2024 February, 26)
[11] Ding, X., Zhou, Z., (2024). On the Partial Autocorrelation Function for Locally Stationary Time Series: Characterisation, Estimation and Inference. arXiv preprint arXiv:2401.15778. Available at https://doi.org/10.48550/arXiv.2401.15778. (Accessed on 2024 February, 27)
[12] Dixon, M. (2018). Sequence Classification of the Limit Order Book using Recurrent Neural Networks. *Journal of computational science*, 24:277-286. https://doi.org/10.1016/j.jocs.2017.08.018
[13] Dixon, M. (2022). Industrial Forecasting with Exponentially Smoothed Recurrent Neural Networks. *Technometrics*, 64(1):114-124. https://doi.org/10.1080/00401706.2021.1921035
[14] Ferianc, M., Que, Z., Fan, H., Luk, W., and Rodrigues, M. (2021). Optimising Bayesian Recurrent Neural Networks on an FPGA-based Accelerator. In *International Conference on Field-Programmable Technology (ICFPT)*. https://doi.org.10.1109/ICFPT52863.2021.9609847
[15] Glorot, X., and Bengio, Y. (2010). Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 249-256. https://proceedings.mlr.press/v9/glorot10a.html
[16] Graves, A. (2013). Generating Sequences with Recurrent Neural Networks. arXiv preprint arXiv
[17] Hamilton, J. D. (2020). Time Series Analysis: *Princeton University Press*.
[18] Hamiane, S., Ghanou, Y., Khalifi, H., and Telmem, M. (2024). Comparative Analysis of LSTM, ARIMA, and Hybrid Models for Forecasting Future GDP. *Ingénieriedes Systèmes d'Information*. 29(3):853-861. http://iieta.org/journals/isi
[19] Henaff, M., Szlam, A., and LeCun, Y. (2016). Recurrent Orthogonal Networks and Long-memory Tasks. In *International Conference on Machine Learning*, 2034-2042.
[20] Hochreiter, S., and Schmidhuber, J. J. N. c. (1997). Long Short-term Memory. *Neural Computation MIT-Press*, 9(8):1735-1780.
[21] Jacobo, A.D., Marengo, A., (2020). Are the Business Cycles of Argentina and Brazil Different? New Features and Stylised Facts. *Paradigma ecońomico. Revista de economía regional y sectorial* 12(2):5–38. https://doi.org/10.36677/paradigmaeconomico.v12i2.14028
[22] Jonathan, D.C., Kung-Sik, C., 2008. Time series analysis with applications in R. 2nd ed.; Springer Texts in Statistics; Springer: Berlin/Heidelberg, Germany.
[23] Kumari, A., and Sood, M. (2021). Implementation of Simple RNN and LSTMs-based Prediction Model for Coronavirus Disease (Covid-19). In *IOP Conference Series: Materials Science and Engineering.* https://doi.org//10.1088/1757-899X/1022/1/012015
[24] Li, D., and Zhu, K. (2020). Inference for Asymmetric Exponentially Weighted Moving Average Models. *Journal of Time Series Analysis*, 41(1): 154-162. https://doi.org/10.1111/jtsa.12464
[25] Ljung, G. M., and Box, G. E. (1978). On a Measure of Lack of Fit in Time Series Models. *Biometrika*, 65(2):297-303. https://doi.org/10.1093/biomet/65.2.297
[26] Makatjane, K., and Mmelesi, K. (2024). An Improved Model Accuracy for Forecasting Risk Measures: Application of Ensemble Methods. *Journal of Applied Economics*, 27(1): 2395775. https://doi.org/10.1080/15140326.2024.2395775
[27] Makatjane, K., and Moroke, N. (2022). Examining Stylised Facts and Trends of FTSE/JSE TOP40: A Parametric and Non-parametric Approach. *Data Science in Finance and Economics*, 2(3):294-320. https://doi.org/10.3934/DSFE.2022015
[28] Or, B. (2020). The Exploding and Vanishing Gradients Problem in Time Series. Medium. Towards Data Science. Available at https://medium.com/metaor-artificial-intelligence/the-exploding-and-vanishing-gradients-problem-in-time-series-6 (Accessed on 2024 February, 23).

[29] Pirani, M., Thakkar, P., Jivrani, P., Bohara, M. H., and Garg, D. (2022). A Comparative Analysis of ARIMA, GRU, LSTM, and BiLSTM on Financial Time Series Forecasting. In *2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE), Ballari, India*, 1-6. https://doi.org/10.1109/ICDCECE53908.2022.9793213.

[30] Ramchandani, M., Khandare, H., Singh, P., Rajak, P., Suryawanshi, N., Jangde, A. S., and Sahu, M. (2022). Survey: Tensorflow in Machine Learning. In *Journal of Physics: Conference Series*, 2273(1): 012008. IOP Publishing. https://doi.org/10.1088/1742-6596/2273/1/012008

[31] Rastogi, V. R., and Dhar, J. (2012). Effect of Increasing the Forecast Horizon on Correlation between Forecasted Returns and Actual returns: An Empirical Analysis. *International Journal of Accounting and Finance*, 3(3): 193-206. https://doi.org/10.1504/IJAF.2012.048498

[32] Shahi, Tej Bahadur, Ashish Shrestha, Arjun Neupane, and William Guo. (2020). Stock Price Forecasting with Deep Learning: A Comparative Study Mathematics 8(9): 1441. https://doi.org/10.3390/math8091441

[33] Siami-Namini, S., Tavakoli, N., and Namin, A. S. (2018). A Comparison of ARIMA and LSTM in Forecasting Time Series. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 1394-1401. IEEE. https://doi.org/10.1109/ICMLA.2018.00227

[34] Sirignano, J., and Cont, R. (2021). Universal Features of Price Formation in Financial Markets: Perspectives from Deep Learning. In *Machine Learning and AI in Finance*, 5-15: Routledge. https://doi.org/10.1080/14697688.2019.1622295

[35] Smyl, S. (2020). A Hybrid Method of Exponential Smoothing and Recurrent Neural Networks for Time Series Forecasting. *International journal of forecasting*, 36(1):75-85. https://doi.org/10.1016/j.ijforecast.2019.03.017.

[36] Taylor, S. J., and Letham, B. (2018). Forecasting at Scale. *The American Statistician*, 72(1):37-45. https://doi.org/10.1080/00031305.2017.1380080

[37] Tsay, R. S. (2014). An Introduction to the Analysis of Financial Data with R. Hoboken, NJ: John Wiley and Sons.

[38] Tsay, R. S. (2010). Financial Time Series and Their Characteristics. *Analysis of Financial Time Series*, Third Edition, 1-27. Hoboken, NJ: Wiley.

[39] Trućıos, C., Taylor, J.W., (2020). Forecasting Value-at-Risk and Expected Shortfall of Cryptocurrencies using Combinations Based on Jump-robust and Regime-switching Models. Available at http://dx.doi.org/10.2139/ssrn.3751435. (Accessed on 2024 February, 24).

[40] Ullrich, T. (2021). On the Autoregressive Time Series Model using Real and Complex Analysis. *Forecasting*, 3(4):716-728. https://doi.org/10.3390/forecast3040044

[41] Wang, Y., Wang, W., Zang, H., and Xu, D. (2023). Is the LSTM Model Better than RNN for Flood Forecasting Tasks? A Case Study of HuaYuankou Station and LouDe Station in the Lower Yellow River Basin. *Water* 15(22):3928. https://doi.org/10.3390/w15223928

[42] Wang, H., Li, T., Zhuang, Z., Chen, T., Liang, H., and Sun, J. (2023). Early Stopping for Deep Image Prior. *Transactions on Machine Learning Research*. Available at https://openreview.net/forum?id=231ZzrLC8X

[43] Wang, H., Li, T., Zhuang, Z., Chen, T., Liang, H., and Sun, J. (2021). Early Stopping for Deep Image Prior.

[44] Wang, L., Ma, F., Niu, T., He, C., (2020). Crude Oil and BRICS Stock Markets under Extreme Shocks: New evidence. *Economic Modelling*, 86:54–68. https://doi.org/10.1016/j.econmod.2019.06.002

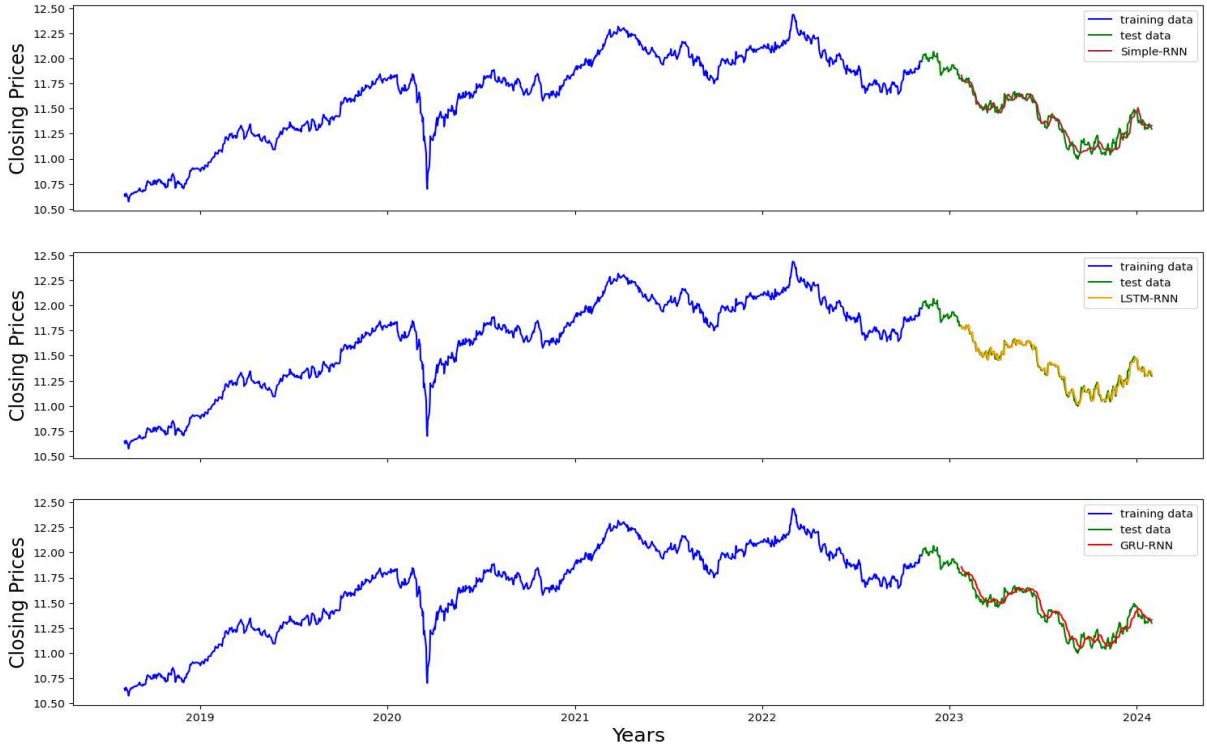
Appendix A

Theorem 1.*The previous size-adaptive RNN with ReLU activation functions can be trained with sample complexity limited to*

$$M_L(\varepsilon,\delta) \leq \frac{128}{\varepsilon^2}\left[\ln\left(\frac{16}{\delta}\right)+\ln\left(\frac{34}{\varepsilon}\right)4\left(n^2+4n+3\right)\left(4n^4+8n^3+4n+10+log_2(8e)\right)\right]$$

and thus $M_L(\varepsilon,\delta) \in \tilde{O}\left(\frac{n^6}{\varepsilon^2}\right)$.

Appendix B



Appendix C