Communications for Statistical Applications and Methods
2024, Vol. 31, No. 1, 105–127

https://doi.org/10.29220/CSAM.2024.31.1.105
Print ISSN 2287-7843 / Online ISSN 2383-4757

Forecasting realized volatility using data normalization and recurrent neural network

Yoonjoo Lee[a], Dong Wan Shin[b], Ji Eun Choi[1,c]

[a]School of Computing, KAIST, Korea;
[b]Department of Statistics, Ewha Womans University, Korea;
[c]Department of Statistics, Pukyong National University, Korea

Abstract

We propose recurrent neural network (RNN) methods for forecasting realized volatility (RV). The data are RVs of ten major stock price indices, four from the US, and six from the EU. Forecasts are made for relative ratio of adjacent RVs instead of the RV itself in order to avoid the out-of-scale issue. Forecasts of RV ratios distribution are first constructed from which those of RVs are computed which are shown to be better than forecasts constructed directly from RV. The apparent asymmetry of RV ratio is addressed by the Piecewise Min-max (PM) normalization. The serial dependence of the ratio data renders us to consider two architectures, long short-term memory (LSTM) and gated recurrent unit (GRU). The hyperparameters of LSTM and GRU are tuned by the nested cross validation. The RNN forecast with the PM normalization and ratio transformation is shown to outperform other forecasts by other RNN models and by benchmarking models of the AR model, the support vector machine (SVM), the deep neural network (DNN), and the convolutional neural network (CNN).

Keywords: asymmetry, realized volatility, normalization, ratio transformation, recurrent neural network

1. Introduction

Forecasting volatility for financial assets has recently garnered significant interest among investors and professional analysts because of its importance in risk management, derivative pricing, and portfolio allocation. In recent years, availability of high frequency intra-day asset price data sets has enabled us to realize the hidden volatility via realized volatility (RV). Since volatility is related to future asset price uncertainty, forecast of RV is one of the main issues. Diverse methodologies for forecasting RV have been developed during the last decade in various academic fields and applied in real markets. Major results on RV forecasting were reviewed by McAleer and Medeiros (2008a), Andersen and Terasvirta (2009), and Shin (2018).

These days, taking well advantage of hidden nonlinearity in time series data, deep learning has been widely considered in regression and classification problems, and has proved to perform very well compared with other methods. Furthermore, since neural network methods do not require normality assumptions for population distribution, financial analysts, economists and statisticians are increasingly using these methods for data analysis (Mantri *et al*., 2010). With this trend, there have been

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2023-00239009, 2022R1F1A1068578, RS-2023-00242528).

[1]Corresponding author: Department of Statistics, Pukyong National University, Busan 48513, Korea. E-mail: choije@pknu.ac.kr

Published 31 January 2024 / journal homepage: http://csam.or.kr

many research works which compare methods using deep neural networks with others for time series data. For example, there were comparisons between the performances of artificial neural network (ANN) and autoregressive integrated moving average models for forecasting commodity prices (Kohzadi *et al*., 1996); ANNs and support vector machines were used to predict prices on the Istanbul stock exchange national 100 index (Kara *et al*., 2011); Mantri *et al*. (2010) calculated the volatilities of Indian stock markets using the artificial networks and the statistical time series methods of GARCH, EGARCH, GJR-GARCH, IGARCH models.

The main characteristic of time series data is serial dependence. In particular, financial volatility data sets such as RV data sets and implied volatility data sets show very strong persistent serial dependence, called long-memory, see Section 2 and for example Engle and Patton (2007), Andersen and Terasvirta (2009), and Bollerslev *et al*. (2016). The serial dependence of RV data sets leads us to adopt recurrent neural networks (RNNs), particularly long short-term memory (LSTM), a variant of RNN, which has recently emerged as an effective model for a variety of applications involving sequential data (Hochreiter and Schmidhuber, 1997; Karpathy *et al*., 2015). In addition, gated recurrent unit (GRU) is widely used for analyzing time series data including language, video, and others, also can outperform LSTM units in terms of convergence in CPU time, parameter updates, and generalization (Chung *et al*., 2014; Cho *et al*., 2014). There is also bidirectional RNN, an extended version of RNN, which can be trained without limitations using input information just up to a preset future frame and offers better performance than RNN. The hyperparameters of the recurrent network with LSTM units and GRU such as the input length for serial dependence consideration, the number of layers, and others are chosen by the nested cross validation (CV).

In the recent literature, RNN models have been sucessfully adopted for forecasting volatilities of financial time series. Kim and Won (2018) considered a combination of the deep learning LSTM for volatility forecast of the KOSPI (Korean stock price index) and compared the proposed model with other standard models. Liu (2019) compared volatility forecasts of the US S&P 500 index and AAPL (Nasdaq Apple stock price) by the SVM, the LSTM and the statistical GARCH model and identified the situations in which the LSTM has better performance. Bucci (2020) studied forecast of monthly realized volatility of the US S&P stock price index and compared some RNN models and some econometric models. Lei *et al*. (2021) considered the LSTM with text mining as a sentiment analysis to improve forecast of the realized volatility of Stock Rambler in China stock market. They used the comments of investors in the stock bar as the basis for the sentiment analysis. We see some RNN methods for forecasting volatilities of commodity prices: The CNN-LSTM by Vidal and Kristjanpoller (2020) for gold price and the LSTM+GARCH model by Hu *et al*. (2020) for copper price. In the above RNN methods, they do not address the dominant data feature of long-memory and asymmetry of financial volatility data sets. The RNN models would produce better volatility forecasts if the long-memory and asymmetry features of volatility are well addressed. These features will be addressed by ratio transformation and piecewise min-max normalization. This consideration will be shown to produce considerably better RV forecasts for 10 major world stock price indices than other methods which ignore the dominant data features.

We forecast daily RV of major stock price indices of the US and the EU. Recall that the deep learning forecasts usually show out-of-scale issue that forecasted values are limited within the range of training data, and the issue is generally resolved by considering ratios rather than data themselves (Fisher and Karauss, 2018; Liu and Mehta, 2019; Oztekin *et al*., 2016). Moreover, compared to original RV's, the ratioed RV's has substantially less persistent serial dependence which render the RNN forecast model to perform better. We therefore consider relative ratio of adjacent RVs instead of the RV itself. Learning and forecasting are first made for the RV ratios, and forecasts of RV values

Table 1: Summary statistics of 100RV, 100 v_t, $t = 1, \ldots, T$; The total number observation is $T = 3344$ for each asset

Index	Mean	SD	Skewness	CoV	Min	Median	Max	$\hat{\rho}_1^v$	$\hat{\rho}_{10}^v$	$\hat{\rho}_{100}^v$
S&P	0.92	0.69	15.16	0.58	0.11	0.62	29.66	0.82	0.65	0.29
RUSSELL	1.01	0.64	14.76	0.39	0.20	0.73	27.65	0.81	0.63	0.30
DJIA	0.90	0.63	16.14	0.57	0.14	0.63	30.47	0.78	0.63	0.27
NASDAQ	0.86	0.69	16.41	0.48	0.18	0.65	24.81	0.82	0.63	0.29
FTSE	0.76	0.53	17.78	0.45	0.21	0.57	23.63	0.83	0.68	0.36
CAC	0.96	0.80	15.09	0.48	0.22	0.85	26.75	0.81	0.62	0.28
DAX	0.99	0.82	14.88	0.50	0.19	0.83	27.69	0.81	0.63	0.23
AEX	0.86	0.69	14.65	0.54	0.21	0.71	24.53	0.83	0.66	0.31
SSMI	0.76	0.62	19.58	0.42	0.24	0.62	25.45	0.82	0.63	0.31
IBEX	1.11	0.73	14.89	0.55	0.28	0.94	27.24	0.79	0.58	0.29

$\hat{\rho}_k$ is the sample autocorrelation of lag $k = 1, 10$.

are next computed from those of RV ratio values. The forecasts based on ratioed RVs will be shown to be better than those based on RV themselves.

In addition to the long-memory feature of RV data sets, another important one is asymmetry as reviewed by Corsi (2009), McAleer and Medeiros (2008a), and many others. The asymmetric feature of RV data sets still remains in RV ratio data sets: Heavier right tails than left tails, as shown in Sections 2 and 5. We address the asymmetry issue by piecewise min-max (PM) normalization to scale the data. The PM normalization is shown to faithfully address the asymmetry in the RV ratio data resulting in better forecasts than the other usual normalizations: Min-max (MM) normalization, Gaussian mixture (GM) normalization.

Superiority of the proposed method is illustrated in an out-of-sample forecast comparison for RV's of 10 major US and EU stock price indices: S&P 500, Russell 2000, DJIA, NASDAQ, FTSE 100, CAC 40, DAX, AEX, SSMI, IBEX 35. (i) The proposed RNN model with ratio transformation and PM normalization (RNN-R-PM) is the best among 6 RNN models (with or without ratio transformation and MM, GM, PM other normalization) and 4 benchmarking models of the AR, the SVM, the DNN, and the CNN. (ii) The RNN is shown to be better than the DNN and the CNN when applied to the original RV with the MM normalization as is the usual practice. (iii) Ratio transformation of RV gives the RNN better forecasts than original RV. (iv) PM normalization yields better forecast for the RNN than MM and GM normalizations.

The rest of the paper is organized as follows. Section 2 describes our research data. Section 3 presents three normalization methods, RNN-based methods, and the nested CV. Section 4 explains experimental design, evaluation measures, an implementation of the nested CV method. Section 5 provides an illustration of the modeling procedure, the CV results and forecast performance comparison results of the RNN-based methods and the benchmarking models. Finally, Section 6 concludes this work.

2. RV data and RV ratio data

A realized variance is simply the sum of squares of intra-day log-returns for which the unit of time is usually one day and it is indexed by $t = 1, 2, \ldots$. For a day t, if we let equally spaced intra-day log price observations be $p_{t-1+\gamma j}$ for $j = 0, 1, \ldots, M, \gamma = 1/M$, γ-spaced realized variance for day t is given as $\sum_{j=1}^{M} (r_{tj}^{(\gamma)})^2$ for $t = 1, 2, \ldots$, where $r_{tj}^{(\gamma)} = p_{t-1+\gamma j} - p_{t-1+\gamma(j-1)}$ is the log-return for the γ-spaced intraday interval $[t - 1 + \gamma(j - 1), t - 1 + \gamma j]$. Then, RV for a day t is given by $v_t = \sqrt{\sum_{j=1}^{M} (r_{tj}^{(\gamma)})^2}$.

Table 2: Summary statistics of ratio of adjacent RV values, $u_t = v_t/v_{t-1}, t = 2, \ldots, T$

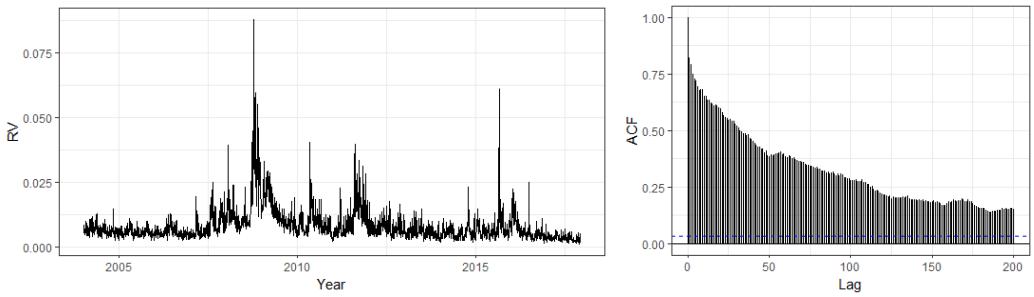
Index	Mean	SD	Skewness	CoV	Min	Median	Max	$\hat{\rho}^u_1$	$\hat{\rho}^u_{10}$	$\hat{\rho}^u_{100}$
S&P	1.07	0.41	3.71	0.36	0.27	0.99	8.957	−0.34	0.01	−0.01
RUSSELL	1.06	0.47	2.79	0.33	0.23	0.99	6.58	−0.35	0.02	0.01
DJIA	1.08	0.46	3.30	0.37	0.26	1.01	8.17	−0.37	−0.01	−0.02
NASDAQ	1.04	0.35	3.26	0.36	0.22	0.99	6.58	−0.30	0.01	−0.01
FTSE	1.04	0.34	6.23	0.28	0.18	1.00	8.82	−0.31	0.06	0.00
CAC	1.04	0.32	4.77	0.30	0.27	1.04	7.84	−0.32	0.06	−0.01
DAX	1.04	0.34	3.29	0.33	0.31	0.99	6.21	−0.33	0.04	−0.01
AEX	1.03	0.33	4.98	0.31	0.24	1.00	8.14	−0.31	0.06	−0.02
SSMI	0.99	0.31	7.07	0.25	0.25	0.99	8.19	−0.30	0.02	−0.03
IBEX	1.04	0.35	5.56	0.32	0.27	1.00	7.34	−0.30	0.04	−0.02

We consider RV data sets of 10 major stock price indices, 4 in the US and 6 in the EU for the period from Jan. 5, 2004 to Nov. 30, 2017 which are obtained from the oxford-man institute's realized library (https://realized.oxford-man.ox.ac.uk/). sampling interval for the RVs is $\gamma = 5/(24 \times 60)$ day, i.e., 5 minute. The total number of RV values for each asset is $T = 3344$.

Summary statistics of the RV data sets are shown in Table 1. The means of RV are all roughly 1% and skewness values are all much larger than 0 indicating very right-skewed data distributions, which can also be confirmed by much larger values of (max - median) than (median - min) values, for example $29.66 - 0.62 > 0.62 - 0.11$ for S&P. The CoV denotes the coefficient of variation, SD/Mean. As the randomness limits the accuracy of forecasts in general, forecasting with less dispersed data is presumably advantageous, and the CoV can be utilized as a rough indicator for this. The CoV of RV has a value around 0.5 for every index.

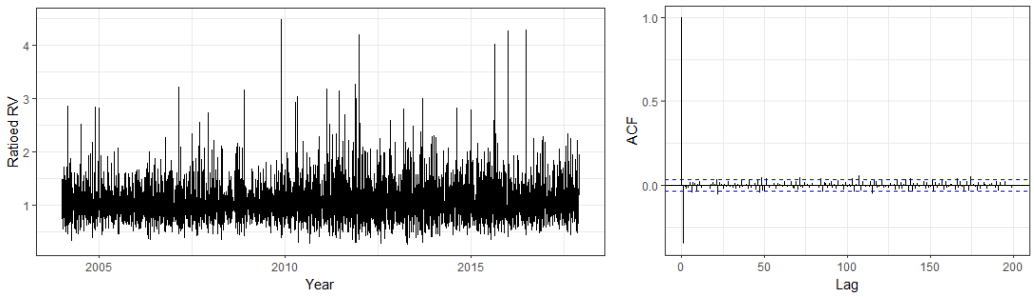
From Table 1, it can be inferred that the RV data is somewhat irregular due to high skewness and CoV. Also, it is generally reported that the RNN-based forecasts usually suffer from the out-of-sacle issue that the forecast value tends to be limited within the range seen from the training data. These concerns can be relieved by using the relative ratio of RV instead of the original value to forecast (Liu and Mehta, 2019). The relative ratio of RV is denoted $u_t = v_t/v_{t-1}, t = 1, \ldots, T-1$. Since RV values are positive, the relative ratio of adjacent values of RV is defined well. When we compute a forecast $\hat{v}_{t+1}$ of v_{t+1} at time t, we first conducted forecast $\hat{u}_{t+1}$ of u_{t+1} using a trained model for $u_1, \ldots, u_t$ and next construct forecast $\hat{v}_{t+1} = \hat{u}_{t+1} v_t$. Summary statistics of the ratio data are reported in Table 2. We can check from the relatively small values for the skewness and CoV than those of RV that the relative ratio of RV is less dispersed and skewed than RV. These advantageous stochastic features support taking relative ratios instead of RV values for forecasting.

In addition to avoiding the out of scale issue and having smaller skewness and CoV than the RV data, the ratioed RV data has another important advantage of resolving the long memory of the original RV data, which considerably improve deep learning forecasts. For this discussion, we choose the S&P 500. The conclusion of this paragraph holds good for all the other assets. Figure 1 shows time series plot of the daily RV series v_t of the S&P 500 and its autocorrelation function (ACF) $\hat{\rho}^v_k = \widehat{\text{corr}}(v_t, v_{t-k}), k = 1, \ldots, 200$. The figure reveals that the RV series v_t has persistent strong autocorrelation even for large k, for example, $\hat{\rho}^v_{10} = 0.65$, $\hat{\rho}^v_{100} = 0.29$ which is called long memory. A long memory of v_t means that two observations (v_t, v_{t-k}) have non-negligible correlation even though they are far apart, i.e. $\text{corr}(v_t, v_{t-k})$ is not close to 0 even for large k. This means that v_t's have high similarity for a wide range of t. The long-memory of v_t is a consequence of the repeated "∧"-like peaks as shown in Figure 1(a), which span for long periods of more than several weeks or several months. When a batch of consecutive v_t's are input to an RNN, information in the batch is not well



(a) Timeseries (b) ACF

Figure 1: *Timeseries plot of the RV of the S&P 500 and its autocorrelation function (ACF) plot.*



(a) Timeseries (b) ACF

Figure 2: *Timeseries plot of the ratioed RV of the S&P 500 and its ACF plot.*

utilized by the RNN because of high similarity of neighboring observations in the batch, producing poor forecasts. The high similarity of neighboring values arising from long memory is considerably resolved in the ratio data $u_t = v_t/v_{t-1}$ as shown in its Figure 2(a) and autocorrelation function $\hat{\rho}_k^u = \widehat{\text{Corr}}(u_t, u_{t-k})$ in Figure 2(b). In Figure 2(b), $\hat{\rho}_k^u$, $k = 2, \ldots, 200$, all lie within or slightly out of the dotted ± 0.035 band, which is the $\pm 2 \times$ standard error $(\hat{\rho}_k^u) = \pm 2/\sqrt{T} = \pm 0.035$. Note however$\hat{\rho}_1^u = -0.34$ is significantly non-zero, telling that u_t has a short-memory. Now, a batch of consecutive u_t in the ratioed RV consists of less autocorrelated values than v_t of the original RV and would be more well utilized in RNN forecast model to produce better forecasts as will be revealed in Section 5.3.

The remaining non-negligible autocorrelation for one lag $k = 1$ (significantly nonzero $\hat{\rho}_1^u$) as well as significant asymmetry (skewness > 0) as shown in Table 2 lead us to consider forecasting by the RNN, which is more suitable for serially dependent data than other deep learning models such as the DNN and the CNN. In our RNN methods, we will address the asymmetry by PMM normalization and the serial correlation by the LSTM and the GRU.

3. Data normalization and RNN methods

3.1. Normalization

The values passed between layers in the RNN models need to be scaled to $[0, 1]$ or $[-1, 1]$. Normalization techniques can be used to scale the data and can speed up training time by starting the training

process with the scaled data (Liu and Mehta, 2019). Since we focus on forecasting the relative ratio of RV, normalization is adopted to it. The parameters required for the normalization process are usually obtained from a set of values, which is mostly a train set. Here, let $\mathcal{U}$ be the set on which normalization is based. In this work, three kinds of normalization techniques are employed as described in the following.

The min-max normalization (MMN) is the simplest method to adjust range of data to lie in $[0, 1]$ (Liu and Mehta, 2019). For a value u, the MMN is applied as follows

$$f_m(u; \mathcal{U}) = \frac{u - \min(\mathcal{U})}{\max(\mathcal{U}) - \min(\mathcal{U})}. \tag{3.1}$$

It preserves the stochastic properties of the data except that only the range is scaled to $[0, 1]$. Also, the MMN has the advantage of low complexity of denormalization.

The Gaussian mixture normalization (GMN) exploits a density estimation method using linear superposition of Gaussian distributions. The distribution of values in $\mathcal{U}$ can be approximated through Gaussian mixture model. We use a well-known fact that $F_X(X)$ follows a uniform distribution on $(0, 1)$ for any random variable X whose distribution function F_X is continuous. The GMN transforms a value u to be in $(0, 1)$ as

$$f_g(u; \mathcal{U}) = \sum_{m=1}^{M} \pi_m \Phi\left(u \mid \mu_m, \sigma_m^2\right), \quad \sum_{m=1}^{M} \pi_m = 1, \ \pi_m \geq 0,$$

where $\Phi(u|\mu_m, \sigma_m^2)$'s are the distribution functions of the Gaussian distributions $\mathcal{N}(\mu_m, \sigma_m^2)$. The parameters π_m, μ_m and σ_m are usually estimated using the expectation maximization algorithm (Bishop, 2006). This technique enables learning independent of stochastic characteristics such as the skewness of data. However, the GMN has the disadvantage of complexity of denormalization because the inverse of Gaussian mixture requires numerical computations. In this work, $M = 3$ is used.

The Piecewise min-max normalization (PMN) is a simple combination of the two techniques mentioned above in that the distributional feature of data is addressed by different min-max scaling to the left and right half of data. For a value u, the PMN is applied as follows

$$\begin{aligned} f_p(u; \mathcal{U}) = & \frac{1}{2} \frac{u - \min(\mathcal{U})}{\operatorname{median}(\mathcal{U}) - \min(\mathcal{U})} I\{u < \operatorname{median}(\mathcal{U})\} \\ & + \left\{ \frac{1}{2} + \frac{1}{2} \frac{u - \operatorname{median}(\mathcal{U})}{\max(\mathcal{U}) - \operatorname{median}(\mathcal{U})} \right\} I\{u \geq \operatorname{median}(\mathcal{U})\}, \end{aligned} \tag{3.2}$$

where $I\{\cdot\}$ is an indicator function. The asymmetric features of an RV data set are more faithfully addressed by the PMN than the MMN and the GMN because, in the PMN, different normalizations are applied to the left half and the right half of the RV data set. This normalization transforms the median of data to $1/2$ and we can retrieve the median of the data from the forecast value of $1/2$ through denormalization. As in the case of the MMN, it has also the advantage of low complexity of denormalization.

3.2. RNN methods

We use RNNs which are widely used for sequential data (Karpathy *et al*., 2015). Computations in most RNN's are composed of a sequence of 3 blocks having parameters to be updated and associated transformations: Block 1, from the input to the hidden state; block 2, from the previous hidden state

to the next hidden state; and block 3, from the last hidden state to the output. To construct a recurrent network of this structure, hyperparameters should be determined: The input length, the number of time steps of an input sequence; the hidden size; the number of units of a hidden layer; the number of layers, and others. In practical applications, gated RNNs are used as highly effective sequence models, which handle the issue of vanishing or exploding derivatives by creating paths through time. For a cell of a gated RNN, we consider two structures LSTM and GRU in this work. In addition, a bidirectional structure that depends on the whole input sequence by using not only the forward causal structure but also the backward direction can also be considered (Schuster and Paliwal, 1997). As the structure of the network varies according to the cell structure and the RNN direction, these are also considered as hyperparameters (Goodfellow *et al*., 2016). A method for hyperparameter determination is discussed in Section 3.3

Here we briefly explain the RNN model we use. Let the input length be given as q. For an input sequence $(x_1, \ldots, x_q)$ of length q, which are normalized ratios through one of the MMN, the GMN, or the PMN mentioned in Section 3.1. The x_t at t is also represented by $\mathbf{h}_t^0$ for later use. Although a boldface is used here, it is actually a scalar in this work. We denote the hidden state vectors as $\mathbf{h}_t^l \in \mathbb{R}^n$, where t is a time step, $l = 1, \ldots, L$ is the index of layer, and n is the hidden size. The model consists of a total of $L+1$ layers, with L hidden layers and 1 output layer. Each element of the time series data is injected into the network one at a time sequentially. Although the detailed structure for calculating the hidden states $\mathbf{h}_t^l$s of the hidden layers is different for each cell structure, it is common in that $\mathbf{h}_{t-1}^l$, from the previous time of the same layer, and $\mathbf{h}_t^{l-1}$, from the previous layer at present time, are used as

$$\mathbf{h}_t^l = g\left(W^l \begin{pmatrix} \mathbf{h}_t^{l-1} \\ \mathbf{h}_{t-1}^l \end{pmatrix}\right),$$

where W^l is the parameter matrix of the l^{th} hidden layer. A detailed description of the transformation $g(\cdot)$ in each cell structure and bidirectional recurrent network are given in Appendix A. For the last hidden state $\mathbf{h}_q^L$, we apply an additional layer to this n-dimensional vector to get an estimate $\hat{x}_{q+1}$ of the normalized ratio of RV at time $q+1$. The operation of the output layer can be expressed as $\hat{x}_{q+1} = \text{sigm}(W_O \mathbf{h}_q^L)$, where $W_O \in \mathbb{R}^{1\times n}$ is the parameter matrix. Sigmoid activation 'sigm' guarantees that the estimated value is in between 0 and 1. Then, the parameter matrices are updated in the way of minimizing the mean squared error (MSE) loss between $\hat{x}_{q+1}$ and x_{q+1} over all the train data.

3.3. Nested CV for hyperparameter determination

The RNN-based methods discussed in the previous subsection have various hyperparameters, such as the RNN direction, the input length, the number of layers, and others that need to be specified. It is important to choose good hyperparameter combination for training an RNN-based model for the good forecast. In order to compare the forecast accuracies of two or more methods with different hyperparameters, researchers tend to use CV (Oztekin *et al*., 2016). Typically, the data for training an RNN-based model and evaluating forecast performance consist of a train set and a test set. A train set is further divided into a train subset and a validation subset. Since there are temporal dependencies in the data, in order to accurately simulate the real world forecast environment in which we stand in the present and forecast the future (Tashman, 2000), we need to set the validation set comes after the train subset, and the test set after the train set. Also, if we ignore the time sequence and choose a test set arbitrarily, then our test set error is a poor estimate of error on an independent test set. For these reasons, traditional k-fold CV is inappropriate when dealing with time series data. Instead, we use the 'Nested CV' procedure which provides an almost unbiased estimate of the true error (Varma and

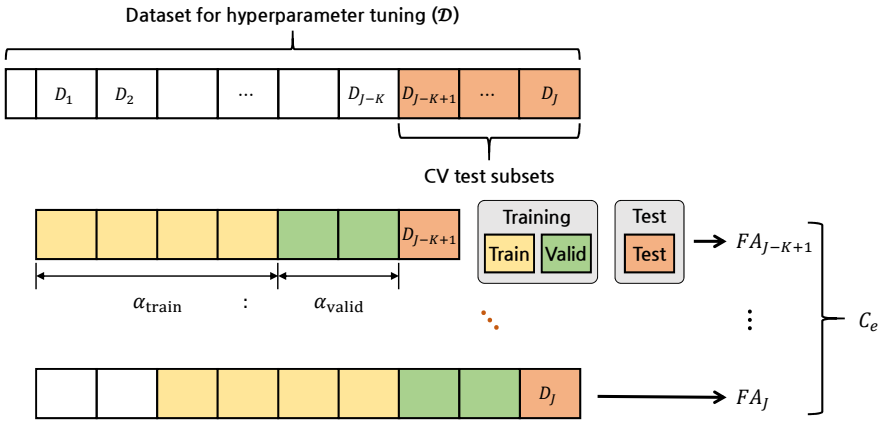


Figure 3: *Nested CV method.*

Simon, 2006): We create multiple train/test splits and average the errors over all the splits to produce a better estimate of model forecast error. Day forward-chaining technique which is based on method called forward-chaining or rolling-origin evaluation (Tashman, 2000) is used in this study. A detailed implementation of the CV is described in Section 4.4.

4. Experimental design

We elaborate how the methods described in Section 3 are applied to our data. For the purpose of forecasting future RV values, we perform CV in exactly the same way it would be done in the actual forecast process. That is, the models are trained on past values and their performances are tested on future values. This type of CV can be used not only to determine hyperparameters but also to examine the effects of normalization techniques through the stochastic characteristics inherent in the data.

Let $\mathcal{D}$ denote the dataset that will be used for tuning hyperparameters. First, the dataset ($\mathcal{D}$) is divided into subsets consisting of d elements. If we assume that J subsets denoted by $D_1, D_2, \ldots, D_J$ are used, the time index of the data is adjusted so that the elements of D_j become $\{v_{(j-1)d+1}, \ldots, v_{jd}\}$. The RV values before the first element v_1 of D_1 are naturally indexed with values below zero. We assume that one subset serves as a test set in the CV process. When we want to evaluate forecast performance with a subset as the test set, we configure a train subset and a validation subset preceding the test set so that the sizes of train subset and validation subset have a ratio of $\alpha_{\text{train}} : \alpha_{\text{valid}}$ where α_{train} and α_{valid} are positive integers. The validation subset is used for adopting early stopping technique (Goodfellow *et al.*, 2016). In the following we discuss in more detail how the RNN model training is performed and the CV process is applied. The methods of this section as well as those in Section 3 are illustrated in Section 5.1 in a forecasting setup for RV data.

4.1. Data for training

We describe how to construct input-output pairs for training our RNN-based models discussed in Section 3.2. Let a subset D_k for a test set be given, where k is larger than $\alpha_{\text{train}} + \alpha_{\text{valid}}$. Then, $\alpha_{\text{train}} + \alpha_{\text{valid}}$ consecutive subsets preceding D_k are utilized for training. When the length q of the input data sequence for an RNN-based model is given, we use a set of $q + 1 + (\alpha_{\text{train}} + \alpha_{\text{valid}})d$ consecutive

RV values prior to $v_{(k-1)d+1}$, which is the first value of D_k, as

$$\{v_s : (k-1-\alpha_{\text{train}}-\alpha_{\text{valid}})\,d-q \le s \le (k-1)d\} \tag{4.1}$$

to construct a set $\mathcal{U}_{k,q}$ of $q+(\alpha_{\text{train}}+\alpha_{\text{valid}})d$ consecutive RV ratio values as

$$\mathcal{U}_{k,q} = \{u_s : (k-1-\alpha_{\text{train}}-\alpha_{\text{valid}})\,d-q+1 \le s \le (k-1)d\}\,. \tag{4.2}$$

Now, the normalization is based on $\mathcal{U}_{k,q}$. We apply one of the normalization techniques given in Section 3.1 to the values in $\mathcal{U}_{k,q}$, and make a set of normalized ratio values as

$$\left\{x_s = f_{\cdot}\left(u_s;\mathcal{U}_{k,q}\right) : (k-1-\alpha_{\text{train}}-\alpha_{\text{valid}})\,d-q+1 \le s \le (k-1)d\right\}.$$

Finally, we construct input-output pairs for RNN learning by sliding the window of length $q+1$ by a time unit over the sequence of normalized relative ratios. Then, we get $(\alpha_{\text{train}}+\alpha_{\text{valid}})d$ input-output pairs as

$$\left\{\left((x_{s-q},\ldots,x_{s-1}),x_s\right) : (k-1-\alpha_{\text{train}}-\alpha_{\text{valid}})\,d+1 \le s \le (k-1)d\right\}.$$

Among them, the first $\alpha_{\text{train}}d$ input-output pairs are used as a train subset for actual update of the parameter matrices of RNN model and the last $\alpha_{\text{valid}}d$ pairs are used as a validation subset for adopting early stopping technique.

4.2. Forecasting process

We describe how forecasts are made using a trained RNN-based model. Let the subset D_k to be forecasted and the length q of the input are given. After the RNN-based model learning of Section 3.2 with the input-output pairs constructed as described in the previous subsection 4.1, for each element v_{t+1} in D_k, we compute a forecast $\hat{v}_{t+1}$ by using the trained model. We begin with the previous $q+1$ RV values $(v_{t-q},\ldots,v_t)$. Then, the sequence $(x_{t-q+1},\ldots,x_t)$ of the normalized ratios is generated by applying the same normalization technique as in the training process to the sequence of relative ratios. Specifically, $x_s = f_{\cdot}(u_s;\mathcal{U}_{k,q})$ for $t-q+1 \le s \le t$ where $\mathcal{U}_{k,q}$ is the set of RV ratios used for training defined in the previous subsection 4.1. If an input $(x_{t-q+1},\ldots,x_t)$ is applied to the trained model, an output $\hat{x}_{t+1}$ is generated. To make a forecast for v_{t+1}, we first denormalize this value by $\hat{u}_{t+1} = f_{\cdot}^{-1}(\hat{x}_{t+1};\mathcal{U}_{k,q})$. Then, our forecast value for v_{t+1} is given by $\hat{v}_{t+1} = v_t\hat{u}_{t+1}$.

4.3. Evaluation measures

We have 1-step forecasts $\{\hat{v}_{(k-1)d+1},\ldots,\hat{v}_{kd}\}$ for $\{v_{(k-1)d+1},\ldots,v_{kd}\}$ in D_k computed by the method of subsections 4.1 and 4.2. Then, forecast accuracies are usually compared by computing the measures given in the following:

$$\text{Root mean squared error; RMSE} = \sqrt{\text{MSE}},\quad \text{MSE} = \frac{1}{d}\sum_{t=(k-1)d+1}^{kd}(v_t-\hat{v}_t)^2. \tag{4.3}$$

$$\text{Mean absolute error; MAE} = \frac{1}{d}\sum_{t=(k-1)d+1}^{kd}|v_t-\hat{v}_t|\,. \tag{4.4}$$

$$\text{Mean absolute percent error; MAPE} = \frac{100\%}{d} \sum_{t=(k-1)d+1}^{kd} \left| \frac{v_t - \hat{v}_t}{v_t} \right|. \tag{4.5}$$

4.4. Application of nested CV

We elaborate the CV process adopted in this work. To evaluate the forecast performance of the RNN-based method for a combination of hyperparameters, the forecast accuracy is measured on the last K subsets $D_{J-K+1}, \ldots, D_J$. In other words, we put each of the last K subsets as a test set and evaluate the forecast performance through K different learning and forecast processes using the methods mentioned in subsections 4.1 and 4.2. This process is applied to actual forecasts in the same way, so that the CV simulates exactly what we want to do. The CV estimate of the overall performance criteria is calculated by simply the average of the forecast performances on the K consecutive subsets $\{D_{J-K+1}, \ldots, D_J\}$ as $C_e = (1/K) \sum_{j=J-K+1}^{J} \text{FA}_j$, where FA_j is a forecast accuracy such as MSE or those in (4.3)–(4.5) and others. The hyperparameters are determined to minimize C_e. This procedure is illustrated in Figure 3.

For each of possible hyperparameter combinations, we perform the above-mentioned process to get the CV estimate. Then, based on the results from this exhaustive CV process, the combination of hyperparameters to be used for the actual forecast is determined.

5. Results

Recall that the total number of RV values for each asset is $T = 3344$. Hyperparameter tuning and forecasting are made blockwise with block length $d = 150$. Let an asset be given. Let D_0 be the block of the first 344 observations. For notational convenience, we use the time index so that v_{-343} be the first observation and $D_0 = \{v_{-343}, \ldots, v_0\}$. Let $D_1, \ldots, D_{20}$ be the blocks of the size d of the remaining 3000 observations so that $D_1 = \{v_1, \ldots, v_d\}, \ldots, D_{20} = \{v_{19d+1}, \ldots, v_{20d}\}$. We wish to compare 1-step out-of-sample forecasts of the latest 450 RV values in D_{18}, D_{19}, D_{20} by the proposed RNN methods and those by the benchmarking models of the classical AR model and the SVR (support vector regression), the DNN, and the CNN. We use $J = 17$ blocks $D_1, \ldots, D_{17}$ for hyperparameter tuning. Since the forecast target has 450 elements, the test set consists of three latest blocks D_{18}, D_{19}, and D_{20}. To determine the hyperparameters for forecasting by the nested CV method in Section 4.4, we take $K = 5$ and utilize subsets $D_{13}, D_{14}, \ldots, D_{17}$ as test data sets for the nested-CV. For convenience, we denote $\cup_{j=13}^{17} D_j$ as D_{CV}. The sizes of train and validation subsets are set to $\alpha_{\text{train}} d = 10d$ and $\alpha_{\text{valid}} d = 2d$, respectively. The CV process gives three hyperparamter combinations for each normalization technique with which the RNN-model is developed for forecast. For each hyperparameter combination and a normalization technique, the weight W is first estimated by the deep learning models of LSTM or GRU described in Appendix A using data in $D_1, \ldots, D_{17}$. This weight is used for computing 1-step out-of-sample forecast of elements in D_{18}. For forecasts of elements in D_{19} and D_{20}, the weight W is newly estimated blockwise: For $j = 1, 2$, for forecasting elements in D_{18+j}, we use the weights newly estimated using data $D_{6+j}, \ldots, D_{17+j}$. We now have all forecast values of the last 450 RV values in blocks D_{18}, D_{19}, and D_{20}, which are compared with the corresponding observed values by the evaluation measures RMSE, MAE, MAPE in Section 4.3. More detailed modeling procedure is illustrated in Section 5.1. The CV results and forecasted results are given in Section 5.2 and in Section 5.3, respectively.

Figure 4: *Histograms of relative ratio of RV values for each cluster. Each row indicates the results for the clusters of a stock market index. In all cases, the histograms are plotted for the same range from* 0 *to* 3.

5.1. An illustration of the modeling procedure

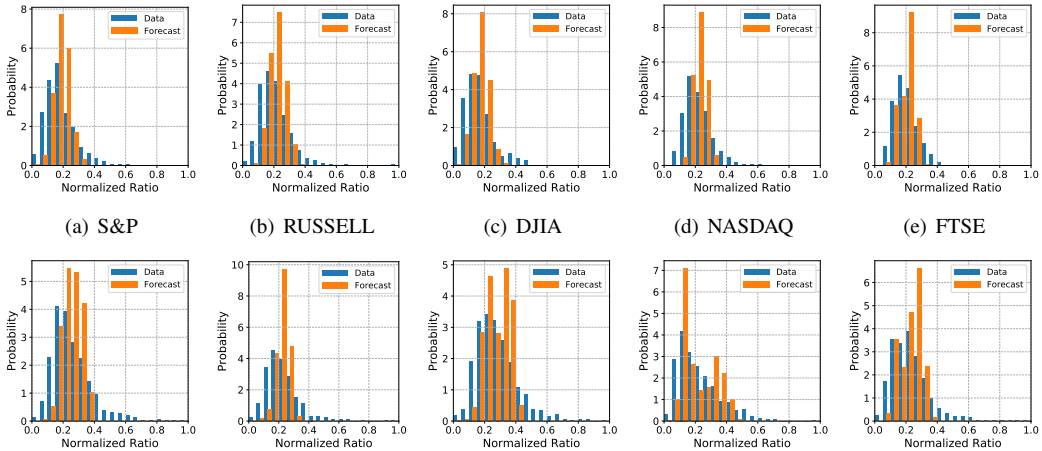
This subsection illustrates the methods in Sections 3, 4 in the setup of the first paragraph of Section 5. Recall that data lengths are all $T = 3344$. We first split the original RV data $\{v_{-343}, \ldots, v_{3000}\}$ into a sequence $D_0, D_1, D_2, \ldots, D_{20}$ such that $D_0 = \{v_{-343}, \ldots, v_0\}$, $D_1 = \{v_1, \ldots, v_d\}, \ldots, D_{20} = \{v_{19d+1}, \ldots, v_{20d}\}$. Our aim is constructing 1-step forecasts of elements in D_{18}, D_{19}, D_{20}.

5.1.1. Hyperparameter tuning and forecasting D_{18}, D_{19}, D_{20}

Hyperparameters are first tuned as in (i) and forecastings are next made as in (ii)–(iv).

(i) Hyperparameter tuning - nested CV method : Hyperparameters are determined by the nested CV method of Section 3.3. We use $J = 17$ blocks $D_1 - D_{17}$ for hyperparameter tuning as in Section 4.4 with the $\alpha_{train} = 10$ blocks for training, the $\alpha_{valid} = 2$ blocks for validation for early stopping and the last $K = 5$ blocks $D_{13} - D_{17}$ for testing. First, forecast of D_{13} and forecast accuracy FA_{13} are computed as follows. We train the RNN forecast model with train data $D_1 - D_{10}$, validation data $D_{11} - D_{12}$ and test data D_{13} as in Section 5.1.2 below. For each v_t in D_{13}, the trained RNN produces forecast of $\hat{v}_t$. We evaluate the forecast accuracy FA_{13} by comparing the forecasted values $\hat{v}_t$ and the actual values v_t in D_{13}. Next, with train data $D_2 - D_{11}$, validation data $D_{12} - D_{13}$ and test data D_{14}, FA_{14} is computed. This procedure is repeated to compute FA_{15}, FA_{16} and FA_{17}. The overall performance measure is $C_e = (1/5)(\mathrm{FA}_{13} + \cdots + \mathrm{FA}_{17})$. We choose the hyperparameter θ which minimizes C_e. For FA, we use the MSE of Section 4.3.

(ii) Weight estimation using data $D_6 - D_{17}$ and forecasting D_{18}: With θ obtained in (i), the RNN estimates weight W using train data $D_6 - D_{15}$ of α_{train}= 10 blocks and validation data $D_{16} - D_{17}$



(a) S&P (b) RUSSELL (c) DJIA (d) NASDAQ (e) FTSE

(f) CAC (g) DAX (h) AEX (i) SSMI (j) IBEX

Figure 5: *Histograms of normalized relative ratios for data subsets used in CV. MM normalization technique is adopted. Forecasted results are obtained from the RNN-based models trained with the best hyperparameter combination given in Table B.1 in Appendix B.*

of $\alpha_{valid} = 2$ blocks for early stopping by a method similar to Section 5.1.2 (i)–(iii) below. With this weight W, for each v_t in D_{18}, the trained RNN produces 1-step forecast $\hat{v}_t$ of v_t following a procedure similar to that in Section 5.1.2 (iv)–(vi) below.

(iii) Weight updating usng data $D_7 - D_{18}$ and forecasting D_{19}.

(iv) Weight updating using data $D_8 - D_{19}$ and forecasting D_{20}.

5.1.2. Forecasting elements of D_{13} using data $D_1 - D_{12}$

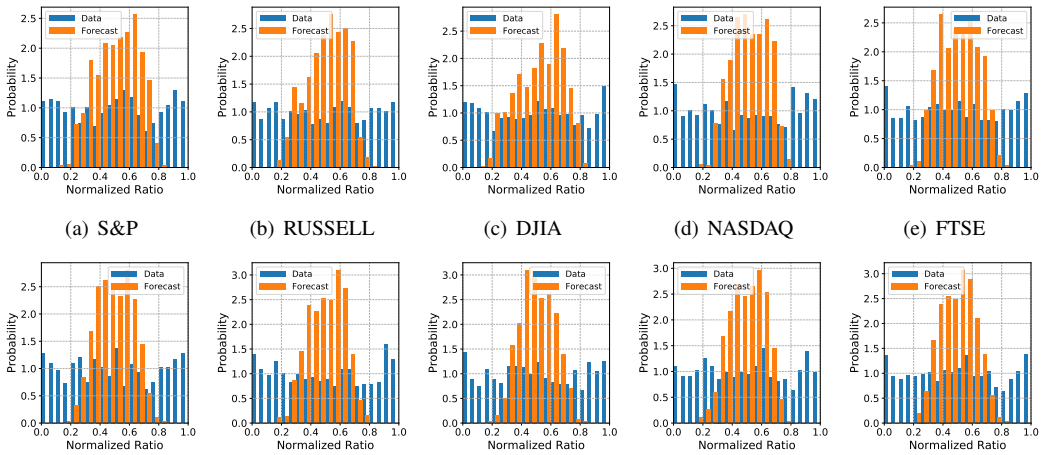
One-step forecasts $\{\hat{v}_{12d+s}, s = 1, \ldots, d\}$ of D_{13} using $D_1 - D_{12}$ is constructed as discussed in Section 4.2 and is illustrated in (i)–(vi) below. The illustration is given for PM normalization. We first make ratio transformation to the whole data $u_t = v_t/v_{t-1}, t = -343, \ldots, 3000$. Let an input length q and other hyperparameters be given. Since the forecast target is D_{13}, we have $k = 13$. Now $\mathcal{U}_{k,q}$ in (4.2) corresponding to the set of $(\alpha_{train} + \alpha_{valid}) = 12$ train-validation blocks for PMN of Section 3.1 and for the RNN input-output pairs of Section 4.1 is $\mathcal{U}_{k,q} = \{u_t : -q + 1 \le t \le 12d\}$.

(i) Bring ratio transformed data: $u_t = v_t/v_{t-1}, t = -q + 1, \ldots, 12d$.

(ii) PMN for normalization: In the piecewise min-max normalization (PMN) (3.2) of Section 3.1, we use the minimum m, the median M_e and the maximum M of $\mathcal{U}_{k,q} = \{u_{-q+1}, u_{-q+2}, \ldots, u_{12d}\}$. We apply PMN in (3.2) to $\{u_{-q+1}, \ldots, u_{12d}\}$ and obtain normalized data

$$x_t = \frac{1}{2}\frac{u_t - m}{M_e - m} \quad \text{if } u_t < M_e \quad \text{and} \quad x_t = \frac{1}{2} + \frac{1}{2}\frac{u_t - M_e}{M - M_e} \quad \text{if } u_t \ge M_e, \ t = -q + 1, \ldots, 12d.$$

(iii) RNN training: Update the weight W of the RNN in Section 3.2 using (input, output) pairs $((x_{s-q}, \ldots, x_{s-1}), x_s), s = 1, \ldots, 12d$. The first $\alpha_{train}d = 10d$ pairs are used for training and the remaining $\alpha_{valid}d = 2d$ pairs are used for validation for early stopping.



(a) S&P (b) RUSSELL (c) DJIA (d) NASDAQ (e) FTSE

(f) CAC (g) DAX (h) AEX (i) SSMI (j) IBEX

Figure 6: *Histograms of normalized relative ratios for data subsets used in CV. GM normalization technique is adopted. Forecasted results are obtained from the RNN-based models trained with the best hyperparameter combination given in Table B.2 in Appendix B.*

Table 3: Hyperparameter settings for CV

Parameter	Settings
RNN direction	Uni, Bi
RNN cell	LSTM, GRU
Input length (q)	3, 4, 5, 6, 7, 8, 9, 10
Number of layers (L)	2, 4, 8, 16
Number of hidden units (n)	4, 8, 16, 32

(iv) Forecasting normalized data of D_{13}: Inputting $(x_{12d-q+s}, \ldots, x_{12d-1+s})$ to the RNN trained in (iii), compute the output $\hat{x}_{12d+s}$, the 1-step forecast of x_{12d+s}, $s = 1, 2, \ldots, d$.

(v) Inverse PMN for denormalization: We apply the inverse transform of PMN in (ii) to compute forecast $\hat{u}_{12d+1}, \ldots, \hat{u}_{13d}$

$$\hat{u}_{12d+s} = m + 2\hat{x}_{12d+s}(M_e - m), \quad \text{if } \hat{x}_{12d+s} < \frac{1}{2} \quad \text{and} \quad \hat{u}_{12d+s} = M_e + 2\left(\hat{x}_{12d+s} - \frac{1}{2}\right)(M - M_e),$$
$$\text{if } \hat{x}_{12d+s} \geq \frac{1}{2}, \quad s = 1, 2, \ldots, d.$$

(vi) Inverse ratio transformation for forecasts of D_{13} : $\hat{v}_{12d+s} = v_{12d+s-1}\hat{u}_{12d+s}$, $s = 1, \ldots, d$.

5.2. CV results

5.2.1. Descriptive analysis

Before applying CV, we first perform descriptive analysis on the relative ratio of RV values. For the relative ratio of RV values corresponding to RV values in $\mathcal{D} = \cup_{j=1}^{17} D_j$, we perform K-means clustering, which is a widely used unsupervised learning algorithm to cluster the data. Taking $q = 10$,

K-means clustering is applied to the set of 10-dimensional vectors $\{(u_{t-10}, \ldots, u_{t-1}) : v_t \in \mathcal{D}\}$, and, for each cluster c, we see the probability distribution of $\{u_t : (u_{t-10}, \ldots, u_{t-1})$ is in cluster $c\}$. Here, we set the number of clusters as 10 by applying the elbow method (Kodinariya and Makwana, 2013). The results are shown in Figure 4. As can be seen in the figure, although the 10 consecutive relative ratios are similar, i.e., contained in the same cluster, the next value is still randomly distributed. Therefore, forecasts by the RNN-based model trained to minimize the average loss are more likely to show some averaged results than the correct ones. Another dominant feature observed from Figure 6 is asymmetry: All the distributions of relative ratios in Figure 6 are significantly right skewed. This point will be discussed in terms of normalization in Section 5.2.3 through the CV results.

5.2.2. Chosen hyperparameters for forecast

As discussed in Section 4.4, the main purpose of the CV is to select the optimal hyperparameters for the RNN methods. For the performance measure FA in Section 4.4, we use MSE. The hyperparameters RNN direction, RNN cell, input length (q), number of layers (L), hidden size (n) are chosen from the candidates in Table 3 by using the nested CV method. The candidate values are typically considered in the literature (Goodfellow *et al*., 2016). We applied early stopping technique and the other parameters are set to the common values widely used in the literature: MSE loss function; Adam optimizer; epochs = 1000; batch size = 40; learning rate = 0.001.

We perform the CV process for each normalization technique described in Section 3.1. According to the CV errors C_e, we take the best three hyperprameter combinations of RNN direction, RNN cell, q, L, and n. The chosen parameter combinations are given in Tables B.1– B.3 in Appendix B. Forecasts are later made in Section 5.3 by averaging three forecasts by the RNN methods with the three chosen hyperparameter combinations.

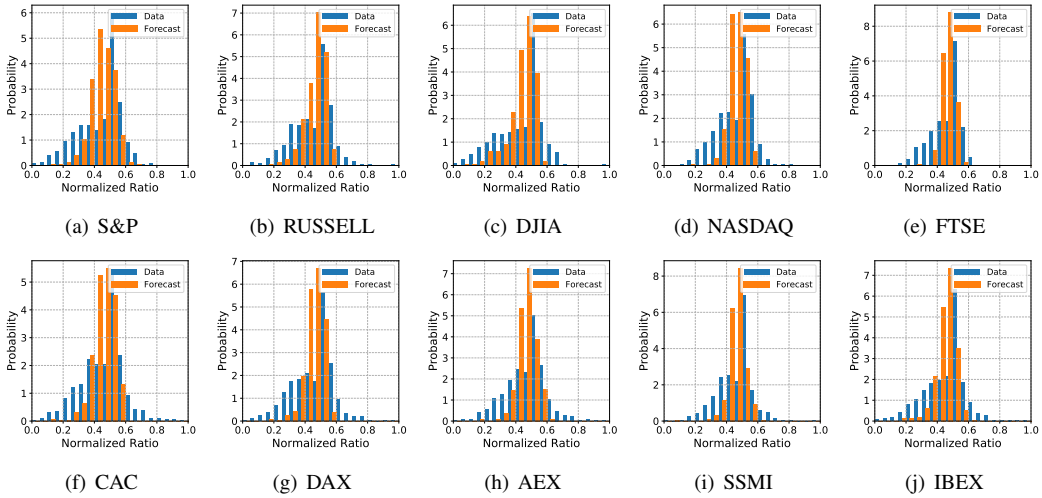
5.2.3. Analysis of CV results

Here, we analyze the stochastic characteristics of the data that can be exploited in the actual forecast. The analysis is performed with the results obtained in the CV process. Since the RNN-based model forecasts the relative ratio of RV values first, we focus on the analysis of this. Specifically, we examine how the normalization techniques affect the forecast of the relative ratios.

Figures 5–7 show the distributions of normalized relative ratios x_s for data subsets used in CV with the MM, GM, and PM normalization techniques given in Section 3.1, respectively. For each normalization technique, the histogram labeled 'Data' is for the set $\cup_{k=13}^{17}\{f.(u_s; \mathcal{U}_{k,q}) : v_s \in D_k\}$ for each index, where q is the input length of the best hyperparameter combination. On the other hand, the ones labeled 'Forecast' are for the forecasted results from the five RNN methods: One for each data subset, trained with the best hyperparameter combination given in Table B.1 in Appendix B. These forecasted results can be expressed by $\{\hat{x}_s : v_s \in D_{\mathrm{CV}}\}$. Consider the first Figure 5 for MMN. Forecasted results account for only part of the middle of the data. Also, we can see the discrepancy between the modes of the data and the forecasted results. The data are usually right-skewed but the forecasted results show relatively symmetric distributions. From these observations, it can be checked that the RNN-based method neglects data asymmetry and gives some averaged results instead of exact ones, as we have expected in Section 5.2.1 via clustering to data.

As we expected for GMN, the data show uniformly distributed histograms as in Figure 6. Although the forecasted results explain only the middle part of the data as above, it can be seen that the median is generally matched well and asymmetry is neglected.

As described in Section 3.1, PMN technique applies two different affine transformations based on



(a) S&P (b) RUSSELL (c) DJIA (d) NASDAQ (e) FTSE

(f) CAC (g) DAX (h) AEX (i) SSMI (j) IBEX

Figure 7: *Histograms of normalized relative ratios for data subsets used in CV from various indices. PM normalization technique is adopted. Forecasted results are obtained from the RNN-based models trained with the best hyperparameter combination given in Table B.3 in Appendix B.*

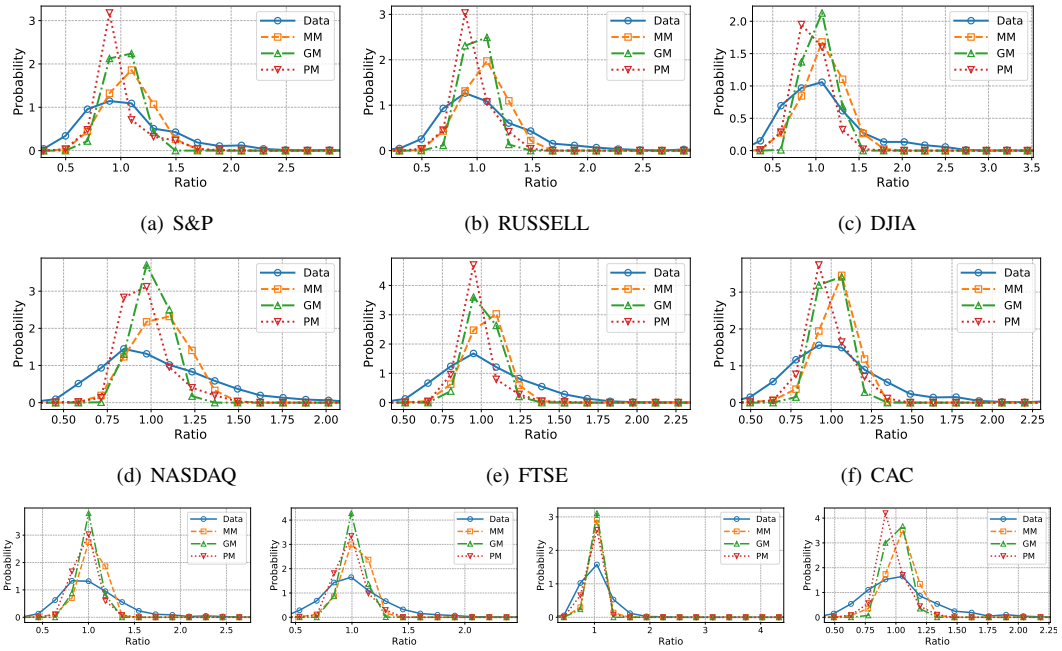
the median. From Figure 7, it can be seen that the right-skewness of the data is partially alleviated through the application of this normalization technique. In the case of the forecasted result, unlike MMN and GMN, it can be checked that the mode of the data is well-matched.

In Figure 8, distributions of forecasted ratios $\hat{u}_t$ and the observed ratios u_t are illustrated. The curve labeled 'Data' indicates the distribution of the set $\{u_s : v_s \in D_{\text{CV}}\}$ for each index. Each of other curves is obtained by denormalizaing the forecasted results shown in Figures 5–7 according to the normalization technique corresponding to its label. For example, a curve with label 'GM' indicates the distribution of $\cup_{k=13}^{17}\{f_g^{-1}(\hat{x}_s; \mathcal{U}_{k,q}) : v_s \in D_k\}$ with q of 'Rank 1' for each index from Table B.2 in Appendix B. As can be seen in the figures, forecasted results via GMN account for a relatively small range of data compared to other techniques based on affine transformations because the inverse of the Gaussian mixture squashes the range of the values in our case. In many cases, the results via the MMN show different modes when the distribution of ratios of the data is skewed, and the PMN compensates for this. Thus, we may expect that the PMN technique shows better performance than the other techniques in that it preserves the proportions of the range that accounts for the data and matches the mode of the data well.

5.3. Forecast results

We perform forecasts for data subsets D_{18}, D_{19}, and D_{20}. The forecast results by the RNN based models and other benchmarking models are compared through various performance measures discussed in Section 4.3. As a benchmarking classical model, we consider the AR model of order p, AR(p), which is a statistical counterpart of the current RNN-based model:

$$v_t = \varphi_0 + \sum_{i=1}^{p} \varphi_i v_{t-i} + \varepsilon_t, \tag{5.1}$$



(a) S&P (b) RUSSELL (c) DJIA

(d) NASDAQ (e) FTSE (f) CAC

(g) DAX (h) AEX (i) SSMI (j) IBEX

Figure 8: *Distributions of relative ratios for data subsets used in CV. Forecasted results are obtained by denormalizing the results labeled 'Forecast' shown in Figures 5–7.*

Table 4: The MAPE(%) of the forecast models

Index	Benchmarking models				RNN-O			RNN-R		
	AR	SVM	DNN	CNN	MM	GM	PM	MM	GM	PM
S&P 500	28.96	53.84	70.16	48.93	32.72	37.95	30.22	26.65	24.01	22.97
RUSSELL 2000	24.96	31.33	35.43	27.34	24.27	25.00	22.06	24.17	23.25	19.80
DJIA	28.36	49.88	66.92	48.34	42.23	41.37	44.02	26.34	23.97	22.59
NASDAQ	25.68	43.24	44.42	34.50	28.87	29.54	22.88	24.35	22.72	21.54
FTSE 100	19.46	24.38	30.74	21.97	19.70	20.49	18.14	18.96	18.71	17.88
CAC 40	21.25	28.00	34.39	25.59	25.52	26.73	20.79	20.24	19.84	18.91
DAX	22.51	29.25	38.46	28.09	42.78	29.24	33.14	22.64	22.09	21.12
AEX	21.38	27.87	32.58	25.03	24.37	25.05	21.84	20.52	20.18	19.18
SSMI	16.17	18.93	28.39	17.89	16.79	17.39	15.66	16.10	15.38	15.18
IBEX 35	20.42	25.17	35.29	27.37	25.05	27.95	21.95	20.61	20.07	18.96

where $\varphi = (\varphi_0, \ldots, \varphi_p)'$ are the parameters of the model, and $\{\varepsilon_t\}_{t\geq 0}$ is a sequence of regression error. The autoregression order p is determined by the minimum Bayesian information criterion (BIC). Other important models of the support vector machine (SVM), the deep neural network (DNN), and the convolutional neural network (CNN) are also considered as benchmarking models. While the CNN was originally applied to image analysis, in many recent studies, the CNN has been applied to forecasting of time series such as stock price, volatility, and many others, see Vidal and Kristjanpoller (2020), Elalem *et al*. (2023), Liang *et al*. (2022), Choi and Shin (2022). This is because CNN captures local features of a time series and has a lower computational cost than the RNN. The DNN and the

Table 5: The MAE × 1000 of the forecast models

Index	Benchmarking models				RNN-O			RNN-R		
	AR	SVM	DNN	CNN	MM	GM	PM	MM	GM	PM
S&P 500	1.18	1.85	2.27	1.63	1.25	1.34	1.18	1.16	1.09	1.09
RUSSELL 2000	1.37	1.71	1.72	1.44	1.32	1.33	1.28	1.37	1.35	1.34
DJIA	1.17	1.76	2.24	1.65	1.49	1.45	1.53	1.14	1.09	1.08
NASDAQ	1.25	1.83	1.81	1.52	1.31	1.32	1.16	1.23	1.19	1.16
FTSE 100	1.00	1.20	1.36	1.08	0.99	1.01	0.96	0.98	1.01	0.98
CAC 40	1.52	1.76	2.04	1.66	1.63	1.66	1.49	1.48	1.52	1.46
DAX	1.57	1.86	2.11	1.71	2.27	1.72	1.91	1.56	1.59	1.55
AEX	1.40	1.68	1.77	1.49	1.46	1.44	1.39	1.35	1.41	1.36
SSMI	0.97	1.13	1.44	1.03	0.99	1.00	0.96	0.99	0.98	0.98
IBEX 35	1.88	2.25	2.69	2.20	2.09	2.24	1.99	1.90	1.96	1.84

Table 6: The RMSE × 1000 of the forecast models

Index	Benchmarking models				RNN-O			RNN-R		
	AR	SVM	DNN	CNN	MM	GM	PM	MM	GM	PM
S&P 500	1.76	2.38	2.64	2.10	1.77	1.84	1.76	1.76	1.74	1.77
RUSSELL 2000	1.75	2.21	2.07	1.83	1.71	1.72	1.71	1.78	1.78	1.78
DJIA	1.83	2.33	2.65	2.15	2.02	1.97	2.12	1.82	1.82	1.81
NASDAQ	1.72	2.33	2.17	1.93	1.75	1.76	1.70	1.75	1.74	1.73
FTSE 100	1.54	1.78	1.79	1.62	1.52	1.54	1.52	1.51	1.57	1.53
CAC 40	2.41	2.63	2.75	2.50	2.45	2.48	2.43	2.35	2.50	2.38
DAX	2.34	2.65	2.72	2.43	2.94	2.42	2.65	2.32	2.44	2.38
AEX	2.43	2.63	2.67	2.45	2.42	2.41	2.46	2.35	2.54	2.41
SSMI	1.53	1.77	1.89	1.57	1.54	1.56	1.54	1.54	1.59	1.60
IBEX 35	3.77	4.17	4.27	3.94	3.83	4.03	3.98	3.71	3.97	3.75

CNN are applied to the original RV data with the min-max normalization as is the common practice. The AR and the SVM are applied to the original RV as is the usual practice. For the SVM, we consider the radial basis function (RBF) as the kernel function. We also consider the kernel parameters C and γ for the SVM which are optimized over $\{0.01, 0.1, 10, 100\}$ and $\{0.25, 1, 4\}$, respectively. The hyperparameters (q, L, n) for the DNN are chosen by the same method for RNN as in Section 5.2.2. For the CNN, the 1D-CNN with a size 3 kernel is considered, the number of filter is optimized over $\{8, 16, 32, 64\}$, and the number of layer is optimized over $\{2, 3\}$. The RNN models with original RV are also compared. The AR model and the SVM update the coefficients every time when an additional observation is available, while the RNN, the DNN, and the CNN update weight matrix W every 150 times when all the additional observations in the block of length 150 are available.

For the RNN methods with ratio transformation, forecasts are made for RV elements in D_{18}, D_{19}, and D_{20}. For each asset, all of the best 3 hyperparameter combinations given in Tables B.1–B.3 are used for forecast. For each hyperparameter combination, we perform forecast five times. Then, as an ensemble method, the final forecast results are made by averaging the total of 15 forecast results, five forecasts from each of the three hyperparameter combinations. Forecasts by other models are constructed similarly. Tables 4–6 show forecast results for the forecasting performance criteria of MAPE, MAE, and RMSE. In the tables and discussions below, for simplicity, we write "O" and "R" for orignal RV and ratioed RV, respectively; "MM", "GM", "PM" for min-max, Gausssian mixture, and piecewise min-max, respectively, normalization. Therefore, for example, RNN-R-PM denote the RNN for ratioed RV with PM. Note that RNN-R-PM adopts our main strategy of resolving long-memory of and addressing asymmetry of RV data sets by R and PM, respectively. We discuss general

findings from the results of Tables 4–6.

(i) RNN-R-PM produces generally the best results. The superiority of the RNN-R-PM model to the other models holds strongly in the MAPE performance being best in all the 10 assets (Table 4) and somewhat strongly in the MAE performance being best in 6 assets out of 10 assets (Table 5). In the RMSE performance, even though RNN-R-MM tends to produce the best results, RNN-R-PM has also good RMSE performance not much worse than that of the best model as shown in Table 6. For all the 10 assets, the RNN-R-PM is considerably better than the benchmarking the AR, the SVM, the DNN, and the CNN models in all of MAPE, MAE, and RMSE with an exception of the AR for asset SSMI in RMSE.

(ii) RNN-O produces much better forecasts than the DNN and somewhat better than the CNN except for 1 asset of the DAX out of 10 assets. Recall that RNN-O, the DNN, and the CNN all apply to the original RV data with the same MM normalization. Recall also that original RV data sets have long memory of persistent strong autocorrelation as discussed in Section 2. The RNN-O takes well advantage of the long memory of the RV data sets and produce better forecasts than the DNN and the CNN which do not explicitly address the serial dependence.

(iii) Ratio transformation generally yields forecast improvement. In forecast performances of RNN-O and RNN-R, R is generally better than the corresponding O except for RMSE of RNN-R-PM model. For example, RNN-R-MM is better than RNN-O-MM. The better forecast performance of R than O is a consequence of resolved long memory in ratioed RV as discussed in Section 2.

(iv) PM is better than MM and GM for the RNN. Superiority of PM over MM and GM holds both in RNN-O and in RNN-R in all of MAPE, MAE and RMSE with an excepton that, in RMSE performance for RNN-R, MM tends to be best but PM is not far away from the best. This better performance of PM than MM and GM is a result of better exploiting of the asymmetry in the RV or in the ratioed RV as in (2).

6. Conclusion

We considered recurrent neural network (RNN) forecast of realized volatility (RV) of 10 stock price indices, four from the US, and six from the Europe. We took three different normalization methods into the relative ratio of RV to adjust the range of data to fit the RNN based model. The piecewise min-max (PM) normalization is proven to faithfully address the asymmetric data distribution feature resulting in better forecasts than min-max or Gaussian mixture normalizations. Also two RNN-variant models LSTM, GRU and bidirectional of them are utilized. We tested the combinations to obtain the optimal models by applying the nested cross-validation (CV). The RNN-based model with PM normalization and ratio transformation (RNN-R-PM) outperformed other RNN models and the bench marking models of AR (autoregressive regression), the SVM (support vector machine), the DNN (deep neural network), and the CNN (convolutional neural network) model in most of the data sets. In particular, the RNN-R-PM performed substantially better than other models in all 10 the data sets in terms of MAPE. Forecasts via ratioed RV data were shown to be better than those based on RV themselves.

Appendix A: Structure of recurrent neural network

Here, we briefly introduce the RNN structures based on the framework provided in Karpathy *et al.* (2015). We denote the hidden state vectors as $\mathbf{h}_t^l \in \mathbb{R}^n$, where $t = 1, \ldots, q$ is regarded as time,

$l = 1, \ldots, L$ is the depth and n is the hidden size. The model consists of L hidden layers and 1 output layer with a total of $L + 1$ layers.

Cell structures for gated RNNs

Although detailed structure for calculating the hidden states $\mathbf{h}_t^l$s of the hidden layers is different for each model, it is common in that $\mathbf{h}_{t-1}^l$, which is from the previous time of the same layer, and $\mathbf{h}_t^{l-1}$, which is from the previous layer at present, are used. In the following, parameter matrices are denoted by W with the corresponding subscript and superscript.

LSTM

LSTM, one of variants of RNN, is designed to prevent gradient vanishing and exploding problems in RNN. This model has cell states and uses input, forget, and output gates for each time step to control what information to throw away from and store in the cell state and update the old cell state. The detailed expression of the steps in LSTM is as follows.

$$\begin{aligned}
\begin{pmatrix} \mathbf{i}_t^l \\ \mathbf{f}_t^l \\ \mathbf{o}_t^l \\ \mathbf{g}_t^l \end{pmatrix} &= \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} \mathbf{h}_t^{l-1} \\ \mathbf{h}_{t-1}^l \end{pmatrix}, \\
\mathbf{c}_t^l &= \mathbf{f}_t^l \odot \mathbf{c}_{t-1}^l + \mathbf{i}_t^l \odot \mathbf{g}_t^l, \\
\mathbf{h}_t^l &= \mathbf{o}_t^l \odot \tanh(\mathbf{c}_t^l),
\end{aligned}$$

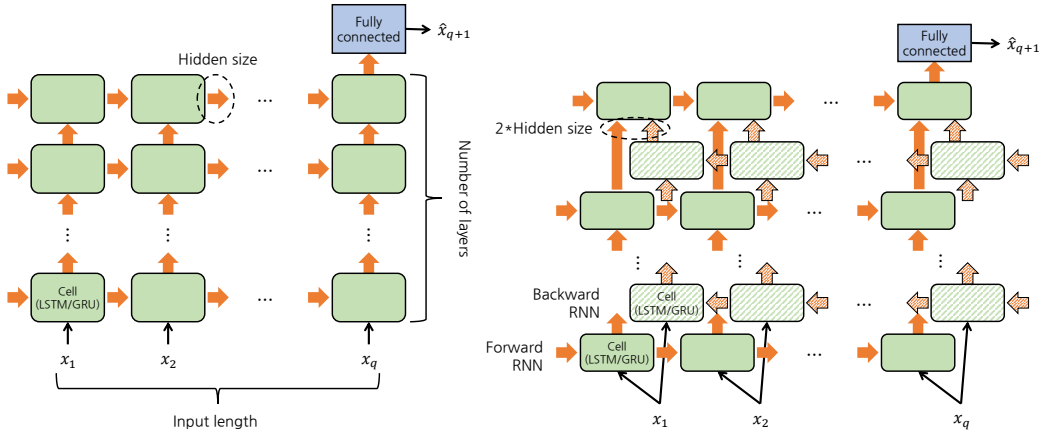
where $\mathbf{i}_t^l, \mathbf{f}_t^l, \mathbf{o}_t^l, \mathbf{g}_t^l \in \mathbb{R}^n$, $W^l \in \mathbb{R}^{4n \times 2n}$, and $\odot$ means element-wise product. In this work, since $\mathbf{h}_t^0$ is a scalar, $W^1 \in \mathbb{R}^{4n \times (n+1)}$. Also, the activation functions, sigmoid function 'sigm' and 'tanh', are applied element-wise. Note that 'sigm' and 'tanh' have ranges of [0, 1] and [−1, 1], respectively. The three vectors $\mathbf{i}_t^l$, $\mathbf{f}_t^l$ and $\mathbf{o}_t^l$ act as input, forget, and output gates which are used to control the cell state. Sigmoid activation derives values from 0 to 1 to keep the model differentiable. The $\mathbf{g}_t^l$ vector through tanh acts as a new candidate, with a value between −1 and 1, and reflected to cell state as vector $\mathbf{i}_t^l$ adjusts. The structure of the LSTM solves the long-term dependency problem in that the gradient of the cell state uninterrupted during backpropagation even if the length of the input sequence is long.

GRU

GRU introduced by Cho *et al*. (2014) is a simpler model than the standard LSTM model. It combines the forget and input gates into a single 'update' gate, also it merges the cell state and hidden state. The detailed expression of the steps in GRU is as follows.

$$\begin{aligned}
\begin{pmatrix} \mathbf{r}_t^l \\ \mathbf{z}_t^l \end{pmatrix} &= \begin{pmatrix} \text{sigm} \\ \text{sigm} \end{pmatrix} W_r^l \begin{pmatrix} \mathbf{h}_t^{l-1} \\ \mathbf{h}_{t-1}^l \end{pmatrix}, \\
\tilde{\mathbf{h}}_t^l &= \tanh\left(W_x^l \mathbf{h}_t^{l-1} + W_g^l \left(\mathbf{r}_t^l \odot \mathbf{h}_{t-1}^l \right) \right), \\
\mathbf{h}_t^l &= \left(1 - \mathbf{z}_t^l \right) \odot \mathbf{h}_{t-1}^l + \mathbf{z}_t^l \odot \tilde{\mathbf{h}}_t^l,
\end{aligned}$$

where $\mathbf{r}_t^l, \mathbf{z}_t^l \in \mathbb{R}^n$, $W_r^l \in \mathbb{R}^{2n \times 2n}$, $W_g^l, W_x^l \in \mathbb{R}^{n \times n}$. For the same reason as in the case of LSTM, $W_r^1 \in \mathbb{R}^{2n \times (n+1)}$ and $W_x^1 \in \mathbb{R}^{n \times 1}$.



(a) Unidirectional RNN (b) Bidirection RNN

Figure A.1: *Considered structures for RNN model.*

Bidirectional RNN

A bidirectional RNN connects two hidden layers of opposite directions to the same output. With this structure, it is possible to get information from backward and forward states which do not have any interactions each other simultaneously (Schuster and Paliwal, 1997). The structure of a bidirectional model is illustrated in Figure A.1(b). We call a typical RNN model described above a unidirectional model, which is illustrated in Figure A.1(a), to distinguish it from a bidirectional model. Specifically, in the bidirectional model, bidirectional structure is applied to the bottom $L-1$ layers except the last layer.

Appendix B: Hyperparameters tuned by the nested CV

According to the CV errors, we take the best three hyperprameter combinations for RNN-R, which are applied to ratioed RV, and these are given in Tables B.1–B.3.

Table B.1: The best three hyperparameter combinations for MM normalization

Index	Rank	Direction	Cell	q	L	n	Index	Rank	Direction	Cell	q	L	n
	1	Uni	GRU	10	2	8		1	Uni	GRU	10	2	8
S&P	2	Uni	GRU	10	2	4	RUSSELL	2	Uni	GRU	8	2	4
	3	Uni	GRU	9	2	16		3	Uni	GRU	9	2	4
	1	Uni	LSTM	10	2	8		1	Uni	LSTM	8	2	4
DJIA	2	Uni	GRU	8	2	16	NASDAQ	2	Uni	LSTM	10	2	8
	3	Bi	GRU	10	2	16		3	Uni	LSTM	10	2	16
	1	Uni	LSTM	9	2	8		1	Uni	GRU	7	2	16
FTSE	2	Uni	LSTM	9	2	16	CAC	2	Uni	GRU	6	2	16
	3	Uni	GRU	9	2	8		3	Uni	LSTM	9	2	16
	1	Uni	GRU	6	2	16		1	Uni	GRU	10	2	16
DAX	2	Uni	GRU	8	2	16	AEX	2	Uni	LSTM	6	2	16
	3	Uni	GRU	9	2	16		3	Uni	GRU	7	2	16
	1	Bi	GRU	7	2	16		1	Uni	LSTM	9	2	16
SSMI	2	Bi	GRU	8	2	16	IBEX	2	Uni	GRU	7	2	8
	3	Bi	GRU	10	2	16		3	Uni	GRU	7	2	16

Table B.2: The best three hyperparameter combinations for GM normalization

Index	Rank	Direction	Cell	q	L	n	Index	Rank	Direction	Cell	q	L	n
	1	Bi	GRU	8	2	16		1	Bi	LSTM	10	4	16
S&P	2	Bi	GRU	10	2	8	RUSSELL	2	Bi	GRU	5	8	16
	3	Uni	GRU	10	2	16		3	Bi	LSTM	5	2	16
	1	Bi	GRU	6	2	8		1	Uni	GRU	6	4	16
DJIA	2	Uni	GRU	10	2	8	NASDAQ	2	Uni	GRU	9	4	16
	3	Bi	GRU	10	4	8		3	Uni	LSTM	6	2	16
	1	Bi	GRU	7	8	16		1	Bi	GRU	6	4	16
FTSE	2	Bi	GRU	10	8	16	CAC	2	Bi	GRU	5	4	16
	3	Bi	GRU	8	4	16		3	Bi	GRU	9	2	16
	1	Uni	GRU	6	4	16		1	Bi	GRU	6	2	16
DAX	2	Bi	GRU	7	4	16	AEX	2	Bi	GRU	6	4	16
	3	Bi	GRU	8	4	16		3	Uni	GRU	6	2	16
	1	Uni	GRU	6	2	16		1	Uni	GRU	9	2	8
SSMI	2	Bi	GRU	7	8	16	IBEX	2	Uni	GRU	6	4	16
	3	Uni	GRU	10	4	16		3	Bi	GRU	7	2	16

Table B.3: The best three hyperparameter combinations for PM normalization

Index	Rank	Direction	Cell	q	L	n	Index	Rank	Direction	Cell	q	L	n
	1	Uni	GRU	8	2	16		1	Bi	LSTM	10	4	16
S&P	2	Bi	GRU	10	2	4	RUSSELL	2	Bi	LSTM	6	4	16
	3	Uni	LSTM	10	2	4		3	Bi	LSTM	5	2	16
	1	Bi	GRU	6	8	16		1	Bi	GRU	10	4	16
DJIA	2	Bi	GRU	7	8	16	NASDAQ	2	Bi	GRU	8	8	16
	3	Bi	LSTM	9	4	16		3	Bi	LSTM	7	4	16
	1	Bi	GRU	7	4	8		1	Bi	GRU	10	8	16
FTSE	2	Bi	GRU	6	4	16	CAC	2	Bi	GRU	5	4	16
	3	Uni	GRU	9	4	4		3	Bi	GRU	6	4	4
	1	Uni	LSTM	6	4	8		1	Bi	LSTM	7	4	16
DAX	2	Uni	LSTM	5	4	16	AEX	2	Uni	LSTM	10	2	8
	3	Bi	GRU	6	4	16		3	Bi	LSTM	5	4	16
	1	Bi	LSTM	10	2	16		1	Bi	GRU	10	8	16
SSMI	2	Uni	LSTM	6	2	8	IBEX	2	Uni	GRU	6	4	8
	3	Uni	LSTM	5	2	4		3	Bi	GRU	9	8	16

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2023-00239009, 2022R1F1A1068578, RS-2023-00242528).

References

Andersen TG and Teräsvirta T (2009). *Realized Volatility, Handbook of Financial Time Series*, Springer, Berlin, Heidelberg.

Bishop CM (2006). *Pattern Recognition and Machine Learning*, Springer-Verlag, New York.

Bollerslev T, Pattona A, and Quaedvlieg R (2016). Exploiting the errors: A simple approach for improved volatility forecasting, *Journal of Financial Econometrics*, **192**, 1–18.

Bucci A (2020). Realized volatility forecasting with neural networks, *Journal of Financial Econometrics*, **18**, 502–531.

Cho K, Merriënboer B-V, Bahdanau D, and Bengio Y (2014). On the properties of neural machine translation: Encoder–decoder approaches, *Proceedings of SSST 2014 - 8th Workshop on Syntax,*

Semantics and Structure in Statistical Translation, 103–111.

Choi JE and Shin DW (2022). Parallel architecture of CNN-bidirectional LSTMs for implied volatility forecast, *Journal of Forecasting*, **41**, 1087–1098.

Chung J, Gulcehre C, Cho K, and Bengio Y (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, Available from: *arXiv preprint arXiv:1412.3555*

Corsi F (2009). A simple approximate long-memory model of realized volatility, *Journal of Financial Econometrics*, **7**, 174–196.

Elalem YK, Maier S, and Seifert RW (2023). A machine learning-based framework for forecasting sales of new products with short life cycles using deep neural networks, *International Journal of Forecasting*, **39**, 1874–1894.

Engle RF and Patton AJ (2007). *What Good is a Volatility Model?, Quantitative Finance, Forecasting Volatility in the Financial Markets* (3rd ed), Butterworth-Heinemann, Oxford, UK.

Fisher T and Krauss C (2018). Deep learning with long-short term memory networks for financial market predictions, *European Journal of Operational Research*, **270**, 654–669.

Goodfellow I, Bengio Y, and Courville A (2016). *Deep Learning*, MIT Press, Cambridge, MA.

Hochreiter S and Schmidhuber J (1997). Long short-term memory, *Neural Computation*, **9**, 1735–1780.

Hu Y, Ni J, and Wen L (2020). A hybrid deep learning approach by integrating LSTM-ANN networks with GARCH model for copper price volatility prediction, *Physica A: Statistical Mechanics and its Applications*, **557**, 124907.

Kara Y, Boyacioglu MA, and Baykan OK (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange, *Expert Systems with Applications*, **38**, 5311–5319.

Karpathy A, Johnson J, and Li F-F (2015). Visualizing and Understanding Recurrent Networks, Available from: *arXiv preprint, arXiv:1506.02078v2*

Kim HY and Won CH (2018). Forecasting the volatility of stock price index: A hybrid model integrating LSTM with multiple GARCH-type models, *Expert Systems with Applications*, **103**, 25–37.

Kodinariya TM and Makwana PR (2013). Review on determining number of cluster in K-means clustering, *International Journal of Advance Research in Computer Science and Management Studies*, **1**, 90–95.

Kohzadi N, Boyd MS, Kermanshahi B, and Kaastra I (1996). A comparison of artificial neural network and time series models for forecasting commodity prices, *Neurocomputing*, **10**, 169–181.

Lei B, Liu Z, and Song Y (2021). On stock volatility forecasting based on text mining and deep learning under high-frequency data, *Journal of Forecasting*, **40**, 1596–1610.

Liang Y, Lin Y, and Lu Q (2022). Forecasting gold price using a novel hybrid model with ICEEMDAN and LSTM-CNN-CBAM, *Expert Systems with Applications*, **206**, 117847.

Liu Y (2019). Novel volatility forecasting using deep learning–Long Short Term Memory Recurrent Neural Networks, *Expert Systems with Applications*, **132**, 99–109.

Liu Y and Mehta S (2019). *Hands-On Deep Learning Architectures with Python: Create deep Neural Networks to Solve Computational Problems Using TensorFlow and Keras*, Packt Publishing Ltd, Birmingham, UK.

McAleer M and Medeiros MC (2008a). Realized volatility: A review, *Econometric Reviews*, **27**, 10–45.

Mantri JK, Gahan P, and Nayak BB (2010). Artificial neural networks-an application to stock market volatility, *International Journal of Engineering Science and Technology*, **2**, 1451–1460.

Oztekin A, Kizilaslan R, Freund S, and Iseri A (2016). A data analytic approach to forecasting daily stock returns in an emerging market, *European Journal of Operational Research*, **253**, 697–710.

Schuster M and Paliwal KK (1997). Bidirectional recurrent neural networks, *IEEE Transactions of Signal Processing*, **45**, 2673–2681.

Shin D-W (2018). Forecasting realized volatility: A review, *Journal of the Korean Statistical Society*, **47**, 395–404.

Tashman L-J (2000). Out-of-sample tests of forecasting accuracy: An analysis and review, *International Journal of Forecasting*, **16**, 437–450.

Varma S and Simon R (2006). Bias in error estimation when using cross-validation for model selection, *BMC Bioinformatics*, **7**, 91.

Vidal A and Kristjanpoller W (2020). Gold volatility prediction using a CNN-LSTM approach, *Expert Systems with Applications*, **157**, 113481.

Received July 8, 2023; Revised October 23, 2023; Accepted November 14, 2023