Artificial Intelligence Review (2024) 57:209
https://doi.org/10.1007/s10462-024-10851-x

Versatile time-window sliding machine learning techniques for stock market forecasting

Zeqiye Zhan[1] · Song-Kyoo Kim[1]

Accepted: 3 July 2024 / Published online: 16 July 2024

Abstract
The stock market which is a critical instrument in the modern financial system consistently attracts a significant number of individuals and financial institutions. The amalgamation of machine learning with stock forecasting has seen increased interest due to the rising prominence of machine learning. Among numerous machine learning models, the long short-term memory (LSTM) is favored by many researchers for its superior performance in long time series. This paper presents an innovative approach that integrates LSTM with versatile sliding window techniques to enhance prediction results and training performance. Moreover, the attached experiments incorporate convolutional filters and combined bivariate performance measures, which are invaluable methodologies for enhancing stock forecasting systems. These significant contributions are expected to be beneficial for researchers operating within this domain.

Keywords Compact data learning · Stock market prediction · Machine learning · Deep learning · Convolutional neural network · Forecasting · Computational finance

1 Introduction

By investing in stocks is the most attractive way of investing in the financial market due to its excellent returns, but it also includes a huge risk factor of potentially losing a large amount of capital. As an example, the historical data of the famous NASDAQ-100 technology index since its inception in 1986 shows that consecutive losing years are very rare, and while it fell 33% in 2022, but rebounded with a 46% return by 2023. On average, several financial institutions predict that its stock growth rate could reach 24.1% by 2024 (Cochintu 2024; Lehtonen 2024; Pizio 2024). This makes predicting stock prices a desirable and important task for stock market investors, but it is also the most challenging task due to the frequently changing nature of stock prices. Thus, accurately predicting stock prices enables investors to improve returns and minimize losses (Jahan et al. 2019). Moreover, many methods have been used to predict the stock market in the past decades, such

✉ Song-Kyoo Kim
amang@mpu.edu.mo

1 Faculty of Applied Sciences, Macao Polytechnic University, Rua de Luis Gonzaga Gomes, Macau, SAR, China

as machine learning models, collection of recent operations of stock companies and news sentiment based systems (Pai and Liu 2018). The main step in most traditional machine learning models is to train them with input data to get a predictive model and later use the model to make predictions.

There are different types of models mainly used for the application of deep learning models in time series prediction problems. RNN (Recurrent Neural Network) which is a Deep State Space model has been proposed by Rangapuram et al. (2018). The model is based on the state space idea that the observations at the current moment are only related to the current state, and the current state is only related to the state at the previous moment. Therefore, the whole prediction model is divided into two parts: modeling the relationship between two consecutive hidden states, and the relationship from the hidden state at the current moment to the prediction result at the current moment. Compared with DeepAR (Salinas et al. 2017), the model does not need to input the real or predicted value of the previous moment at each moment in the training or prediction phase, and establishes the connection between two consecutive moments completely from the hidden state, which solves the problem of inconsistency between training and prediction in DeepAR. The original RNN also has problems, RNN takes a linear sequence structure to continuously collect input information from front to back, but this linear sequence structure has optimization difficulties when back propagation, because the back propagation path is too long, which can easily lead to serious gradient vanishing or gradient explosion problems. In order to solve this problem, researchers developed the LSTM (Long Short-Term Memory) (Hochreiter and Schmidhuber 1997; Gers et al. 2000; Gers and Schmidhuber 2001) and GRU (Gated Recurrent Unit) models (Chung et al. 2014), by increasing the intermediate state information directly backward propagation, to alleviate the problem of gradient disappearance, and obtained very good results. Convolutional Neural Networks (CNNs) are widely used in the fields of CV and NLP, and in the field of time series prediction (Bai et al. 2018). It uses a detailed convolutional combined with a causal convolutional network structure, which is also known as the underlying structure of CNN-based time series prediction models. Causal convolution represents the output at moment *t*, which is obtained by convoluting the inputs at and before moment *t*. In this way, the output at moment *t* avoids the data leakage problem caused by the dependence on information after moment *t*. Dilated convolution solves the problem that the original CNN can only see data within a linear-sized window of history. When the history sequence is long, ordinary convolution needs to increase the convolution size in order to see longer history information, resulting in lower training efficiency. This realizes the introduction of longer history information with the same time complexity and improves the prediction effect. Transformer model firstly achieved significant results in the field of NLP (Natural language processing), and since both NLP and time series are sequential data, Transformer has been gradually applied to time series prediction tasks as well. The structure of Transformer, which is similar to GPT, is targeted for the time series prediction task and achieved better results (Wu et al. 2020). Although Transformer realizes the long period feature alignment through Attention mechanism, the position information can only rely on position embedding. To solve this problem, the combination of CNN and transformer has been proposed which can better uncover sequence fragments with similar patterns and an upgraded version of transformer for long-period prediction model have been studied. Additionally, the ProbSparse self-attention has been proposed to improve the operation efficiency of Transformer in long-period prediction (Li et al. 2019; Zhou et al. 2021). The Nbeats model which has no RNN, CNN or Attention in its internal structure, and the network is all composed of full connections, achieving better results on some open source datasets (Oreshkin et al. 2019). The core idea of Nbeats is that multiple Blocks are connected in series, and each Block learns a part of the information of the sequence. The input of the next Block removes the information that has been learned by the previous Block and only fits the information that has not been learned by the previous Block, similar to the idea of GBDT (Friedman 2001). While these models may correctly predict stock prices, uncertain variables such as government intervention in the business sector may lead to price fluctuations. For example, if we consider a well-run company, the overall trend of the traditional machine learning (ML) model training data is gradually rising, and the stock is assumed to be profitable under normal circumstances at the time of forecasting. However, the company's stock price changes significantly when the government suddenly makes some policy changes in the company's industry. Unfortunately, the traditional ML model could not reflect this sudden change and the traditional ML model will continue to consider the stock as profitable based on its previous cumulative training.

Although stock predictions are not easy due to the nonlinear nature of stock market behavior and new methods are still being tried. It is very popular in the computer field to use machine learning to build a stock prediction model, in this paper we improve the traditional machine learning model algorithm by using a technique called time window sliding (Huynh et al. 2021; Zhou et al. 2018; Li et al. 2018; Thu et al. 2021). In the most cases, a single machine learning predictive model will not perform well in the long period, especially if the unexpected circumstances described above occur. Therefore, traditional machine learning predictive models need to be able to consider long-term data while making real-time model adjustments based on short-term occurrences, rather than fixing parameters after a certain amount of training. The main difference between the tumbling window (TW) technique and the traditional technique is that instead of using all the data at once, the sliding technique divides the data into small segments and trains, validates and tests each segment. Then sliding to the next segment, the previously trained model will also be used to retrain, validate and test on that segment. Each segment will be tested to predict only a small number of stock prices. After several iterations, the prediction of the entire stock data segment is completed. The problem of dramatic short-term changes is well addressed using the TW technique, which allows the model to be retrained and updated each time the next piece of data is predicted. In addition, the model will always be used, where the influence of the initial parameters is also retained, but as more time passes, the initial data has less influence on the prediction. This paper is targeted to improve the stock prediction efficiency by combining LSTM with the adaptive sliding window method and introducing convolutional filters, and meet the demand for more dynamic and accurate prediction models. This research takes the critical role of stock markets in the financial ecosystem and the growing interest in machine learning for stock predictions, acknowledging its ability to process complex financial data and generate actionable insights. The ability of machine learning to process complex financial data and generate actionable insights is recognized in this paper.

This paper is structured as follows: Sect. 2 introduces the theoretical background for machine learning algorithms, sliding window techniques and performance measures. All methodologies in this section are applied into our newly proposed prediction systems. The setup of our experiments, the description of the experiment process are provided in Sect. 3. Additionally, selecting the best system parameters including the selections of the ML algorithm, the sliding window technique and the convolutional filter are also included in this section. The various results of our experiments are described on Sect. 4. Lastly, the conclusion of our research is represented on Sect. 5.

2 Theoretical background

In the early days, the main models used for time series prediction were single series linear models such as ARIMA (Autoregressive Integrated Moving Average Model) (Ariyo et al. 2014), which fitted each series separately. On the basis of ARIMA, optimization methods such as non-linearity and external features have been proposed. However, those models similar to ARIMA have lower efficiency in processing large-scale time series, and due to the independent fitting of each sequence, they cannot share the similar patterns that exist in different sequences. After successful applications in fields such as NLP and CV, deep learning models have gradually been used to solve time series prediction problems. By using a deep learning model with different sequences, the model can learn knowledge from multiple sequences and improve solving efficiency on large-scale time series data.

2.1 Candidate machine learning algorithm

After the description and introduction in the previous section, combined with the favor of many researchers, the recurrent neural network (RNN) structured LSTM algorithm becomes the best candidate model because RNN prone to serious gradient vanishing or exploding problems. Although the performance of LSTM in processing long time series data is not unjustified, it has excellent performance precisely due to its unique structural design. Compared to traditional RNNs, the network structured LSTM significantly improves its learning ability for long-term dependencies, demonstrating stronger modeling and prediction accuracy in numerous tasks involving long sequence data. Although the performance of LSTM also depends on various factors (e.g., specific network architecture, parameter tuning, data quality, and training strategies), the network structured LSTM has been specifically built to address the challenges of long time series data processing and has been widely applied and positively evaluated in various cases. Gradient explosion indicates that gradient updates will increase exponentially. The gradient value becomes larger, we continuously iterate the steps for finding the proper gradient to capture long-distance information. Additionally, the most important concept in LSTM is the cell state and *gate*

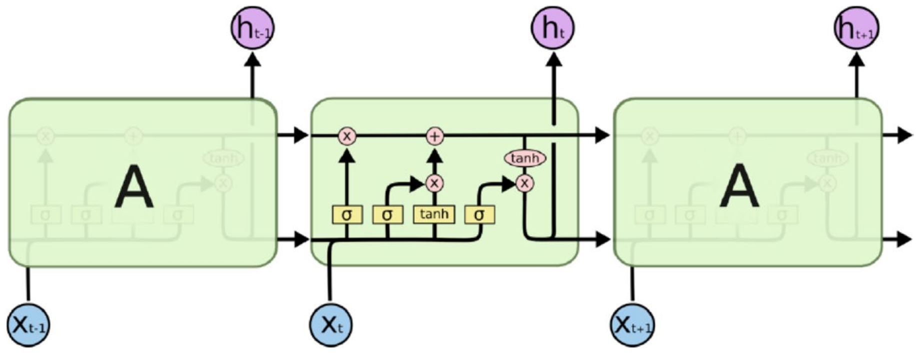

Fig. 1 The structure of the LSTM model (Protopapas and Glickman 2019)

structure, which is divided into forget gate, input gate and output gate. The difference between the standard LSTM and GRU is not very big, in general the construction of GRU has one less gate than LSTM, so there are fewer matrix multiplications. h is the hidden state and represents short-term memory; X represents input (see Fig. 1).

The cell state work C_t as a long term memory and the updates depends on the relation between the hidden state in $t-1$ and the input.

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t, \tag{1}$$

where

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_{t-1}] + b_f\right), \tag{2}$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right), \tag{3}$$

$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right), \tag{4}$$

$$\tilde{C}_t = \tanh\left(W_c \cdot \left[h_{t-1}, x_t\right] + b_f\right), \tag{5}$$

$$h_t = o_t \odot \tanh\left(C_t\right), \tag{6}$$

where the operator $\odot$ represents element wise product. The dimensions and cell state variables of all cells are consistent. Gates can only perform few matrix transformations, and activating the *sigmoid* and *tanh* functions can magically solve all RNN problems. In this experiment, we take the LSTM as the model for traditional machine learning algorithms. The forget gate f_t determines how much of the cell state from the previous moment is retained to the current moment and the input gate i_t determines how much of the network's input at the current moment is saved to the cell state. The output gate o_t determines what will be treated as the output at the current moment. W_f is the weight matrix of the forget gate, $[h_{t-1}, x_t]$ represents connecting two vectors into a longer vector, b_f is the bias term of the forget gate, σ is a *sigmoid* function. $\tilde{C}_t$ is a candidate memory state, C_t is a memory state, and h_t is a hidden state. From (1), take the partial derivative of C_t with C_{t-1} then we have:

$$\frac{\partial c_t}{\partial c_{t-1}} = f_t + c_{t-1}\frac{\partial f}{\partial c_{t-1}} + \tilde{c}_t\frac{\partial i_t}{\partial c_{t-1}} + i_t\frac{\partial \tilde{c}_t}{\partial c_{t-1}}, \tag{7}$$

which means that LSTM shall not experience gradient explosion or vanishing because f_t which is the forget gate has only an effect on the derivative of c_t from (7). Although There are some disadvantages of LSTM which occurs higher computational complexity and longer training time compared to others. This paper focuses more on selecting a model that performs better in small-scale training situations, and the issue of computational complexity and training time between models usually occurs in large-scale model deployment. LSTM is also sensitive to the selection of hyper-parameters (such as learning rate, hidden layer size, gating parameters, etc.) due to its inclusion of multiple trainable gating units and memory cells. Optimizing hyper-parameters is a crucial part but finding optimal hyper-parameters could be resolved through rigorously designed experiments and adjustment.

2.2 Sliding window techniques

In order to improve the prediction performance, we proposed to combine the original LSTM model with the technique of the time sliding window. A simple schematic diagram is shown in Fig. 2: first based on the length of the test data and the distance of each slide, there will be split into different time t. In t_1 time iteration, we are given a fixed window size n, then the input training data will be P_1 to P_n, after the model has completed training, predict the next P_{n+1} data and sliding to the next time iteration t_2. we re-based on the window size n, select the new input training data, actually discard the first training data of t_1 time iteration and add the actual data of the latest predicted data just now, finally get P_2 to P_{n+1}. After that, we retrain and predict P_{n+2} and keep sliding. At this point, we have actually completed two Sliding Window operations, and then the subsequent operation is to continue sliding until the set length of the test data.

And there is another type of Time-Window Sliding technique, which is called Tumbling Window, it is similar to the Sliding Window. For Tumbling Window (see Fig. 3), we did not slide nor re-train the model every time. In t_1 time iteration, we still input the train data from P_1 to P_n, train the model and predict the next P_{n+1} data. But we will continue to use the model to predict P_{n+2} data without any retraining, until t_n time iteration or say we have the test length of n, we will stop the testing and *Tumble*. For the next tumble operation, we will repeat the previous step, use data from P_{n+1} to P_{2n} as training data, and predict data from P_{2n+1} to P_{3n}.

2.3 Convolutional filter techniques

A convolution filter is a discrete convolution with the original signal and a small window used for enhancing the signals (Vaillancourt 2013). It is also known as a kernel that originally designed from image processing and applied for blurring, sharpening, embossing, edge detection, and more. This is accomplished by doing a convolution between the kernel and an original signal. The discrete convolution for one dimension signal is defined as follows:

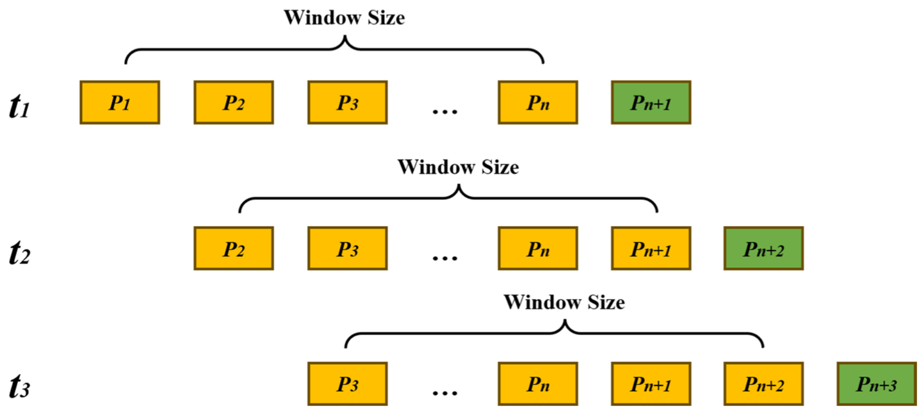

Fig. 2 Sliding Windows (SW)

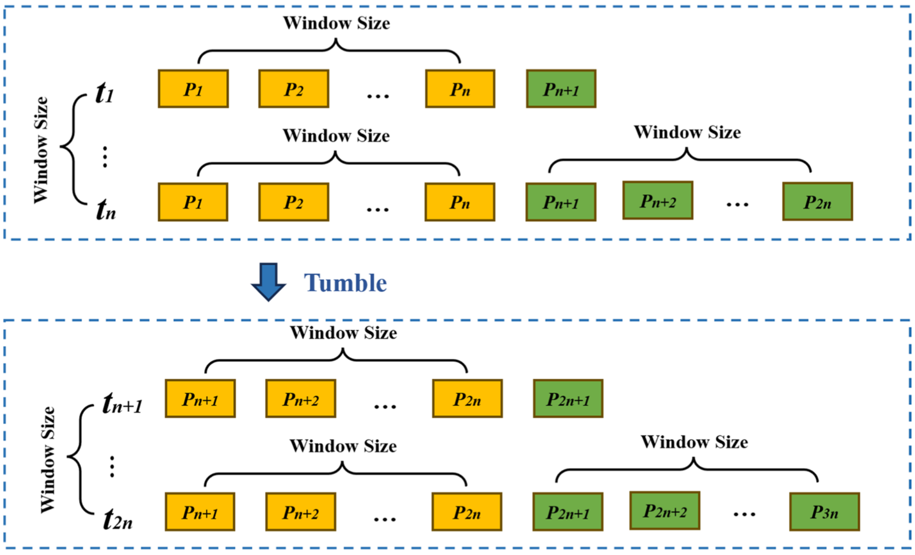

Fig. 3 Tumbling Windows (TW)

$$F(n) = (w * f)(n) = \sum_{i=0}^{n} w(t_i)f(t_{n-i}). \tag{8}$$

Convolutional filters can effectively process data by using a data-driven convolutional filter design method to remove Gaussian noise from images (Fanello et al. 2014). The shortcomings of discrete convolutional filters in our models can naturally capture the combination information and long-term dependency information (Yang 2018). For the data processing required for stock forecasting, Convolutional filter can help in extracting specific features from input data, especially considering that the input training data should not all have the same weight, recent stock data fluctuations should have more reference value for model training, while long-term data should only have a slight impact. In this case, we can design different filters according to the requirements and optimize the data to varying degrees. The related results by adapting the convolution filter shall be provided on the experiment result section. The critical role of convolutional filter techniques in extracting features from noisy financial data, prioritizing the impact of the latest data, and providing a balanced assessment of predictive effectiveness is acknowledged.

2.4 Performance measures

Many evaluation metrics can be used to estimate the accuracy of predictive models. MSE (Mean squared error) is one of them, which is a widely used performance metric for accuracy measurement. Here, the original and predicted values are denoted by y_i, $\hat{y}_i$ respectively, and n denotes the total amount of data. The MSE and another error metric called MAPE (Mean Absolute Percentage Error) are defined as follows:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{9}$$

and

$$\mathrm{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{\hat{y}_i}\right| \tag{10}$$

of which is used as a metric that simply shows the percentage of wrong prediction. In this research, MSE and MAPE have been utilized as performance metrics to assess the outcomes. MSE is determined by squaring the difference between the predicted and actual values for each sample point, followed by averaging these squared differences across all sample points. Meanwhile, MAPE quantifies the average percentage of absolute errors between the predicted and true values. This is achieved by calculating the absolute difference between the predicted and actual values for each sample point, dividing this difference by the actual value to convert it into a percentage form, and then averaging these percentage errors across all sample points. Additionally, a newly proposed innovative performance measure has been applied in this research. The Combined Bivariate Performance Measure (CBPM) is a novel performance metric for evaluating a system or process that encompasses two conflicting performance measures (Kim 2024). The similar phenomena of quality tolerance models (Kolarik 1995) could be applied into the performance measures instead of the quality measures and the trend of performance measures could described as follows (Kim 2024):

- *bigger-is-better* (BIB) means that a bigger or larger value of the performance measure is considered as the better performance;
- *smaller-is-better* (SIB) means that a smaller value or less value of the performance measure (or metrics) is considered as better performance.

The performance measures in this research are the MSE (or MAPE) and the ML training times. The trends of both measures are SIB (*smaller-is-better*). The set represents the independent variable which effects on both the performance functions $f(x)$ and $g(x)$. Then the CBPM function is defined as follows:

$$\xi(x) = \varphi(x) \cdot \varrho(x), \tag{11}$$

where

$$\varphi(x) = \frac{f(x) - f_0}{\Delta f} + \epsilon, \tag{12}$$

$$\rho(x) = \frac{g(x) - g_0}{\Delta g} + \epsilon, \tag{13}$$

where $\epsilon \sim 0.01$. Unlike the continuous case, the functions may not be monotone. It is assumed that $f(x)$ is a primary performance function which is considered more important than the other function. Let us assume this set X as follows:

$$X = \{x_0, x_1, \ldots, x_b\}, \tag{14}$$

then the performance functions are defined as follows:

$$F = \{f_0, f_1, \dots, f_b\}, G = \{g_0, g_1, \dots, g_b\}, \tag{15}$$

and

$$f_k = f(x_k), k = 1, \dots, b-1, \tag{16}$$

$$f_0 = \min(F), f_b = \max(F), \tag{17}$$

from (15) to (17),

$$x_0 = f^{-1}(f_0), x_b = f^{-1}(f_1), \tag{18}$$

then we have

$$g_k = g(x_k), k = 0, 1, \dots, b. \tag{19}$$

From (12)-(18), the the set of the discrete CBPM could be defined as follows:

$$\Xi = \{\xi_0, \xi_1, \dots, \xi_b\}, \tag{20}$$

where

$$\xi_k = \xi(x_k), k = 0, 1, \dots, b. \tag{21}$$

Since the trends of both performance measures in this research are SIB, the optimum x^* shall be determined as follows from (12, 13–21):

$$x^* = \underset{x_k}{\operatorname{argmin}}\left\{x : \xi(x_k), x_k \in \{x_0, \dots, x_b\}\right\}. \tag{22}$$

where the trend of both the performance ratio functions $\varphi(x)$ and $\rho(x)$ are the same as the SIB trend and the best value of the combined single performance measure $\xi(x^*)$ shall a minimum. The CBPM is recognized as being easily adaptable for the selection of the highest-performing systems from among the candidates. This performance measure is considered the most advanced integrated performance measure available, providing a single value that facilitates easy comparison across various machine learning systems (Kim 2024).

3 Advanced stock market forecasting

In this section, we will go through each step of the experiment in detail and give results in many aspects, then we try to compare Sliding Window and Tumbling Window technique, find the advantages and disadvantages of them, test the best option of Window Size by the way.

3.1 Experiment setups

Due to the TW technique in our research work, the program needs to constantly adjust the parameters and data to be inputted for the model will be different at different stages, so overall after the data has been collected, the program runs in a loop. First step is *Model*

Initialization which is mainly gathering the stock market datasets. We used Yfinance (Saiktishna et al. 2022) in Python to get stock data for stocks on NASDAQ including AAPL(Apple inc.), AMD(Advanced micro devices inc.), MSFT(Microsoft corporation) and so on. Since different Window Sizes require different lengths of training data, thus the starting points of training time are depend on different window sizes. Furthermore, we set all the start of their tests points to January 2, 2019, and the testing data were all set to 500 stock days, ending on December 3, 2020. Therefore, we have the definition of the LSTM model in Table 1: All experiments were performed at Kaggle and equipped with two T4 GPUs. We choose to use Adam as optimizer by combining the advantages of two optimization algorithms, AdaGrad and RMSProp (Kingma and Ba 2014). The first moment estimation and the second moment estimation are considered together to calculate the update step size. Adam is simple to implement, efficient in computation, and has low memory requirement, which is very suitable for large-scale data and parameter scenarios. Comprehensive Adam counts as an optimizer that works better by default in many cases (Ruder 2016).

Data processing and parameters setting is the next step which process the data according to certain parameter settings in advance. For example, according to the Window Size to know what is the length of the input training data, read the corresponding training data. In the loop, we set certain variables to record the history of the model training, and record the specific parameters in each update. This will help the program to get where the updated training data is in the new round of the loop.

Data normalization provides an alternative data normalization. For traditional normalization of input data (Jenipher and Radhika 2021; Wang et al. 2023), people usually find the difference between the maximum and the minimum values and calculate the ratio which aligned with the max-min distance. Contrarily, we do not directly normalize the whole time series data, we record the position in the overall input data for the current training and test data of each slide. Let us consider the following dataset $D = \{d_1, d_2, \dots, d_m\}$ and the normalized dataset $\Psi = \{\psi_1, \psi_2, \dots, \psi_m\}$. This newly proposed normalization could be defined as follows:

$$\psi_k = \frac{d_k}{\widetilde{d}}, k = 1, \dots, m, \tag{23}$$

where $\widetilde{d} = \text{Max}(D)$. As the demonstration purposes, let us assign the stock market price dataset for four sliding windows is $D = \{102, 100.5, 103, 106\}$ (see Fig. 4). Then the normalized dataset Ψ could be found as follows from (23):

$$\Psi = \{1.00, 0.99, 1.01, 1.04\}. \tag{24}$$

Table 1 Parameters setting of LSTM model

Parameter	Value
Loss function	Mean squared error
Network architecture	2 LSTM layers with 50 hidden dimension each
Optimizer	Adam
Batch Size	Align with Window Size
Epoch	1000
Learning Rate	0.01

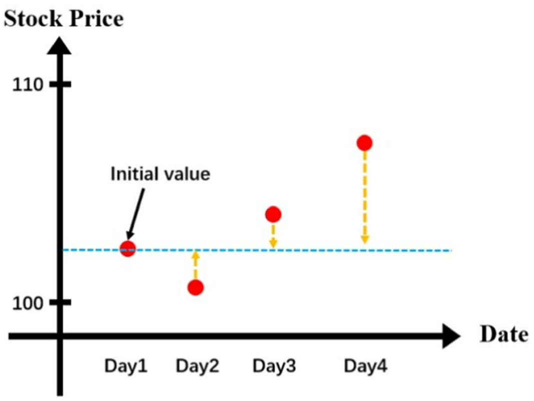

Fig. 4 Samples for the data normalization

Data will have a more uniform data distribution after it is normalized. It helps to achieve faster convergence during training, eliminate the dimensional influence between data, and often improve the accuracy of the final result (Baretto et al. 2021).

After the data processing, we also consider to use the Convolutional Filter technique which introduced in previous 2.3 section: we self-design three different convolutional filter, they can help us to filter the data with different levels of data filtering and try to get better performance results as much as possible. After the above steps are processed, the regular model training is performed and tested to get the predicted results. In particular, since we have previously normalized the data, the output of the model training also needs to be reverse normalized once to get the normal results (i.e., the stock price).

The final step is the Evaluation of Predicted Results. In this step, we need to save it and output the MSE and MAPE mentioned in the previous section to evaluate the result after we finally finish all the tests when we get the output prediction result each time. Generally speaking, the longer a prediction model is trained and the more parameters it has, the better the prediction performance is, so we additionally record the time spent on each training, so that to have a comparison of the performance of the predictions, we can also find out the cases that should spend less time and get better prediction performance. At this step, we have completed a loop, adjust the input training data ranges, record program related variables and other parameters to move to the next batch based on the Window Size and start a new round of the looping procedure. The conditional judgment logic will help us to

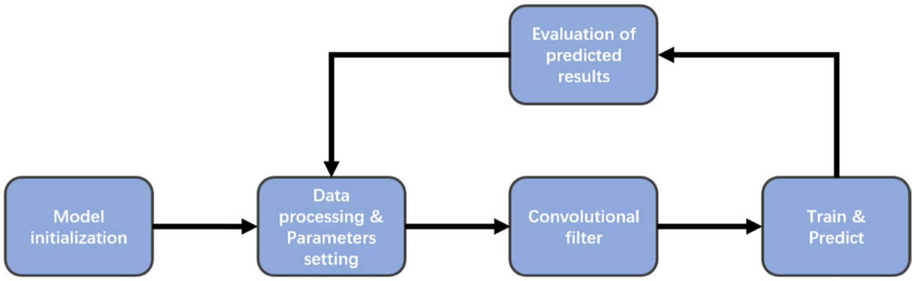
Evaluation of predicted results
Model initialization
Data processing & Parameters setting
Convolutional filter
Train & Predict

Fig. 5 Experiment steps flow diagram

Table 2 Two performance measures of the tumbling window technique based on the window size (reference stock sample: SP500)

Window size	2	2^2	2^3	2^4	2^5	2^6	2^7
Error rate (MAPE) [%]	1.50	2.30	2.17	3.80	4.47	6.24	9.30
Training time [sec]	393.12	205.78	115.88	71.96	60.91	82.52	263.15

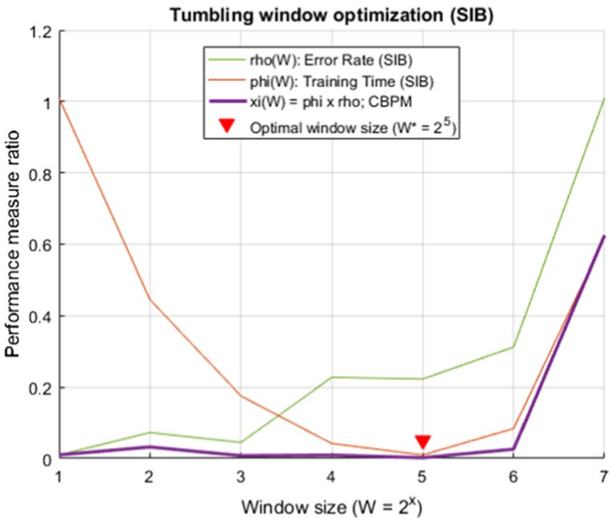

Fig. 6 CBPM Optimization of the window size in the TW technique

implement the corresponding requirements of the Sliding Window and Tumbling Window techniques (see Fig. 5).

3.2 Window size optimization for the tumbling window technique

In order to compare sliding window (SW) and tumbling window (TW), also to find out which window size is the best choice, is the most optimized option, we tested the difference for SW and TW techniques: They all have same parameters setting, generally SW has much better MAPE (or MSE) due to the model is retrained after sliding only a single block data every time in this experiment. However, the SW takes more training time than TW because SW slides every time to retrain the model which definitely spend much more time than TW (see Appendix A). There are two conflicted performance measures which are the error rate (MAPE) and the training time. It is noted that the trends of both performance measures are SIB (i.e., *smaller-is-better*). The TW presents a different pattern, where both small and large window size are inferior to the medium window size. The training time for SW takes up to 100 times slower than the time for TW (see Table 4) although the performance of error rate (MAPE) of SW is slight better than the performance of the TW (see Table 5). Based on this interesting phenomenon, the TW technique has been selected for implementing our prediction system. The performance dataset for the MAPE and the training time of the tumbling window adapted ML system are provided in Table 2.

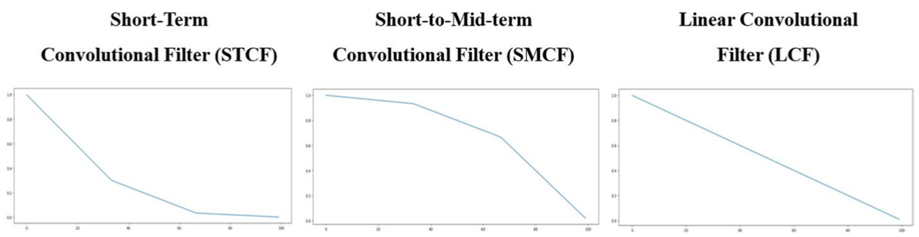

Fig. 7 Convolutional filter samples

Table 3 Convolutional filter comparison (stock sample: SP 500, TW size = 32)

Filter type	Error (MAPE) [%]	Training time [sec]
No filter	4.47	60.91
STCF	3.50	60.91
SMCF	4.92	59.74
LCF	4.41	57.03

From (11) and (22), the CBPM could be calculated for each window size and the optimal window size for the tumbling window technique is 32($= 2^5$) which gives the best performance by combining two conflicted performance measures (see Fig. 6).

3.3 Convolutional filter optimization

We have applied convolutional filter to process the data with different weights and designed three different filters. They are based on the TW are considered for improving our system (see Fig. 7).

The weight design of the short-term Convolutional Filter (STCF) for data has the feature of having the largest proportion of recent data among the three filters, and the smallest proportion of past data. The Short-to-Mid-term Convolutional Filter (SMCF) is just the opposite, it has the largest weighting of medium-term and long-term data among the three filters, and theoretically the long-term data is not very informative for stock forecasting, so the results of SMCF are supposed to be the worst among the three filters. As the name of Linear Convolutional Filter (LCF), the weight design of the recent, medium-term, and long-term data in the LCF follows a linear decrease, in this way the effect of LCF on the data is between STCF and SMCF.

According to our experiments, applying a SMCF provides the better improvement in terms of accuracy compare to apply other filters even though it requires more training time for applying other filters (see Table 3). Several experiments by using the different stock samples have been completed and indicate that the SMCF provides the better results than without using filters (see Appendix B). But it is not guaranteed that applying a filter is always giving the better performance that without applying it because it has the data dependency which relatively weak and applying SMCF usually gives the better performance than without using it (see Table 6).

4 Experiment results

According to the previous section, the TW is decided to be adapted our forecasting system. To show the effectiveness of the TW on ML algorithms, the LSTM without the TW technique and the LSTM with the TW technique have been compared. Five stock samples including Apple, AMD, Micro Soft, Google and SP500 are selected for performance comparisons. All stock samples are collected from the National Association of Securities Dealers Automated Quotations (NASDAQ). Additionally, the performance results by adapting the convolutional filters are also included in this experiment.

4.1 Tumbling window performance

In order to evaluate the performance difference between the LSTM models with and without TW, the length of test dataset is fixed to 500 days which mentioned in the previous section. The training dataset for each stock sample is 1,166 days which are from May 15, 2014 to January 1, 2019 and the testing dataset for each stock sample is 200 days afterward. From Sect. 3.2, the optimal number of tumbling window size has been determined as 32 to be adapted on the original LSTM model and the rest parameters are the same as the set up of the LSTM model without adapting the TW technique.

According to Fig. 8, we can find the comparison between original RNN, LSTM, GRU and LSTM based on TW. Generally, LSTM based on TW is much better than other models at MAPE, which means TW uses entirely less input training data (only 32 inputs at a time, compared to 1,166 inputs for the original LSTM), and also spends only about half the training time of the original LSTM, but gets better performance results in all aspects. Additionally, we can find the result from original RNN is worse than original LSTM and original GRU is actually at same level due the difference in their construction is not very significant.

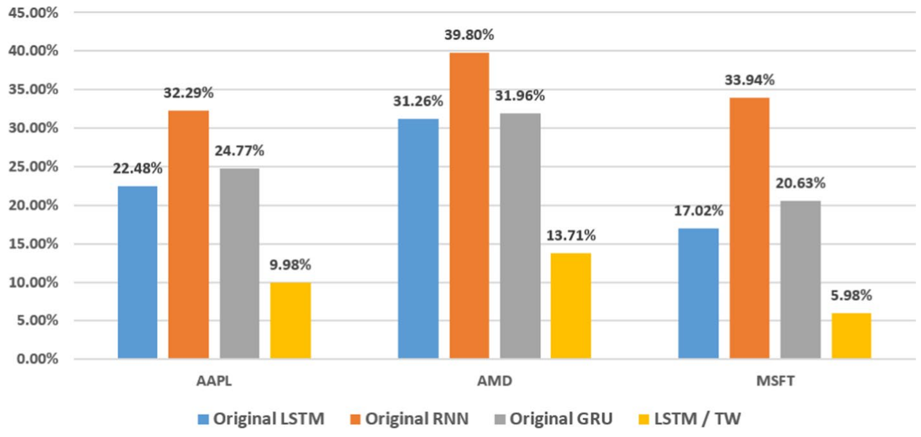

Fig. 8 Error Rate (MAPE) results for various stock samples

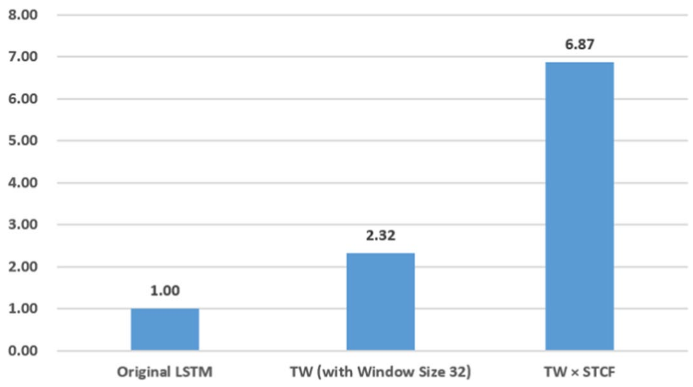

Fig. 9 Accuracy performance result

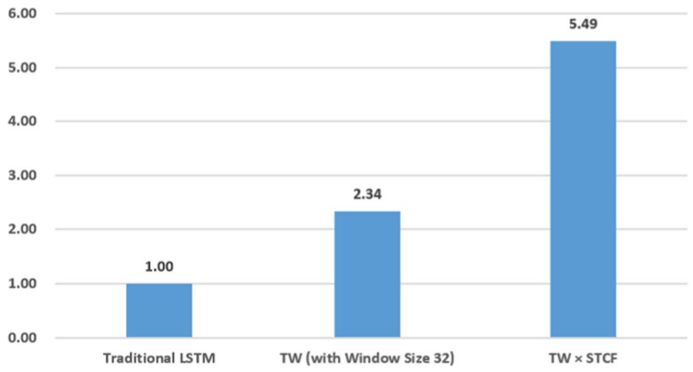

Fig. 10 Training time performance result

4.2 Convolutional filter performance

We consider the performance result of the accuracy for original LSTM as 1.00, then we have separate comparison results for TW and TW x STCF (see Fig. 9). Obviously, TW with STCF has a higher performance index, which prove the Convolutional filter technique can efficiently help to improve stock forecasting result.

In terms of training time in Fig. 10, we can see it is pretty similar with accuracy figure, so the convolutional filter technique also can be applied in terms of improving training time.

The integration of LSTM with TWS and convolutional filters is demonstrated to collectively mitigate the effects of volatility, adapt to market dynamics, and enhance overall forecast robustness, as revealed by the analysis.

5 Conclusion

This research has presented a novel approach to stock forecasting utilizing machine learning techniques, with a particular emphasis on the tumbling window sliding technique. Versatile sliding window techniques which incorporating both simple sliding window (SW) and tumbling window (TW) are applied on the original Long Short-Term Memory (LSTM) machine learning model. The results from the experiment provide strong evidence that the TW technique could significantly enhance prediction performance while drastically reducing training time, compared to original LSTM prediction models. Moreover, the implementation of the TW technique with convolutional filters was explored, resulting in the optimization of both. The convolutional filter technique allows for easier modification of the weights of the customized training data at different periods to achieve the desired prediction results. The combination of time-series targeted machine learning algorithms with TWS and filter techniques offers versatile applicability across diverse domains, including but not limited to gas, oil, gold, and currency exchanges. This innovative approach even holds potential to be extended to predict the moments of cyberattacks in security defense systems. Subsequent efforts will concentrate on enhancing the proposed technique and broadening its utilization across these domains.

Appendix A: performance comparison between sliding and tumbling windows

The comparison of the training time between the sliding window (SW) and the tumbling window (TW) is shown on Table 4. It indicates how the training time is changed within the different window size. Usually, the TW has the shorter training time on most cases. In the

Table 4 Training time comparison between sliding window (SW) and tumbling window (TW)

Window size	SW [sec]	TW [sec]	S/T ratio
2 (= 2^1)	712.03	393.12	1.81
4 (= 2^2)	768.77	205.78	3.74
8 (= 2^3)	839.32	115.88	7.24
16 (= 2^4)	1086.53	71.96	15.10
32 (= 2^5)	2589.80	**60.91**	42.52
64 (= 2^6)	8581.17	82.52	**104.00**
128 (= 2^7)	14690.38	263.15	55.86

Table 5 Error rate (MAPE) comparison between SW and TW

Window size	SW [%]	TW [%]	T/S ratio
2	1.50	**1.50**	1.00
4	1.96	2.30	1.17
8	1.54	2.17	1.41
16	2.02	3.80	1.88
32	2.13	4.47	2.10
64	2.30	6.24	2.71
128	2.53	9.30	**3.67**

view point of the training time, the TW with 32 windows provides the best performance and 64 windows brings make the TW to have the most gap with the SW.

The comparison with another performance measure which is the error rate (MAPE) is provided on Table 5. This comparison indicates that the smaller window size brings the best performance for the error rate regardless which sliding window techniques are applied. The TW performs similarly to the the SW results for the smaller window size, it predicts worse when the window size became larger because the model is less trained and by the opposite T/S ratios, we see that they differ most significantly with 128 windows.

As it has been mentioned, the performance of machine learning based system could be evaluated with multiple performance measures. The optimal window size has been determined based on the combined measure which is called the CBPM (see Sect. 3.2).

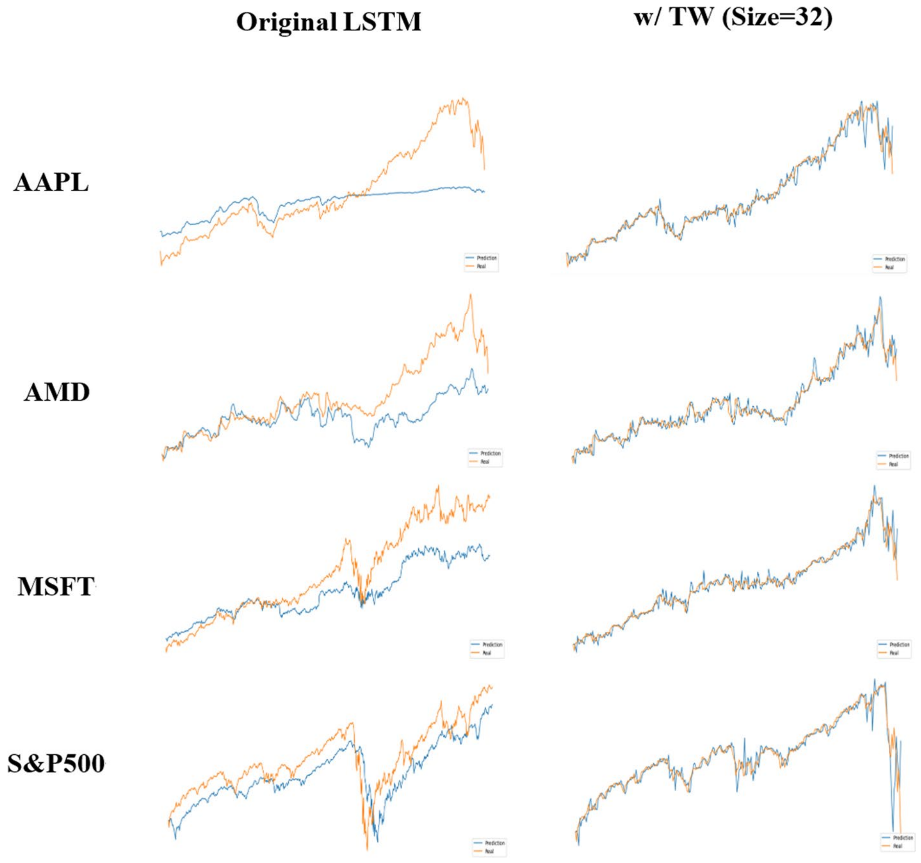

Fig. 11 Prediction comparisons for selected stock samples

Table 6 Performance comparison (MAPE) between the original LSTM and the LSTM with the tumbling window and with the STC filter (window size = 32)

LSTM	w/ TW only		w/ TW & STCF	
MAPE [%]	MAPE [%]	Ratio	MAPE [%]	Ratio
Apple				
22.48	9.98	2.25	8.59	2.62
AMD				
31.26	13.71	2.28	13.44	2.33
Micro Soft				
17.02	5.98	2.85	5.61	3.03
Google				
8.09	4.82	1.69	5.80	1.39
SP500				
10.37	4.47	2.32	3.50	2.96

Appendix B: performance results for various stock samples

For the comparison of original LSTM and TW technique, we can clearly see the graphical comparison of their predicted results through Fig. 11, especially the TW technique has a better predicted result in some regions of the original LSTM where the prediction results are poor.

The performance measures indicate that the TW technique exhibits comprehensive and enhanced performance, as tested on various stocks. The performance measure for each sample stock is presented in Table 6. It should be noted that, for certain stocks (e.g., Google), better results may be achieved without the application of a convolutional filter (e.g. STCF). However, typically, the use of STCF yields better performance results than those achieved without adapting the filter in terms of the accuracy.

Acknowledgements This paper has been revised by using AI/ML-assisted tools.

Author contributions Writing–draft: Z. Zhan; Data collection: Z. Zhan; Conceptualization: S.-K. Kim; Software: Z. Zhan; Visualization: Z. Zhan; Writing–revision: S.-K. Kim; Experiments: Z. Zhan; Review: S.-K. Kim.

Funding This work was supported in part by the Macao Polytechnic University (MPU), under grant RP/FCA-04/2023.

Data availibility The datasets for this paper is available on Kaggle repository (https://www.kaggle.com/datasets/zeqiyezhan/2010-1-1-2019-12-31, No DOI available).

Declarations

Conflict of interest No Conflict of interest.

Ethical approval Not applicable for ethical approval.

References

Ariyo AA, Adewumi AO, Ayo CK (2014) Stock price prediction using the arima model. In: 2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, pp. 106–112

Bai S, Kolter JZ, Koltun V (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271

Baretto AM, Salvi RS, Labhsetwar SR, Kolte PA, Venkatesh VS (2021) Analysis of dimensional influence of convolutional neural networks for histopathological cancer classification. In: 2021 4th Biennial International Conference on Nascent Technologies in Engineering (ICNTE), pp. 1–6

Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555

Cochintu C. NASDAQ 100 forecast and price predictions 2024 and beyond: AI drives best annual performance since 1999. https://capex.com/eu/overview/nasdaq-100-price-prediction

Fanello SR, Keskin C, Kohli P, Izadi S, Shotton J, Criminisi A, Pattacini U, Paek T (2014) Filter forests for learning data-dependent convolutional kernels. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1709–1716. https://doi.org/10.1109/CVPR.2014.221

Friedman JH (2001) Greedy function approximation: a gradient boosting machine. Ann Stat 29:1189–1232

Gers FA, Schmidhuber J, Cummins F (2000) Learning to forget: cuntinual prediction with lSTM. Neural computation 12(10):2451–2471

Gers FA, Schmidhuber E (2001) LSTM recurrent networks learn simple context-free and context-sensitive languages. IEEE Trans Neural Netw 12(6):1333–1340

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural computation 9(8):1735–1780

Huynh D, Audet G, Alabi N, Tian Y (2021) Stock price prediction leveraging reddit: The role of trust filter and sliding window. In: 2021 IEEE International Conference on Big Data (Big Data), pp. 1054–1060

Jahan I, Sajal SZ, Nygard KE (2019) Prediction model using recurrent neural networks. In: 2019 IEEE International Conference on Electro Information Technology (EIT), pp. 1–6

Jenipher VN, Radhika S (2021) Svm kernel methods with data normalization for lung cancer survivability prediction application. In: 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), pp. 1294–1299

Kim S-K (2024) Combined bivariate performance measure. Submitted

Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980

Kolarik WJ (1995) Creating quality: concepts, systems, strategies, and tools. McGraw-Hill, London

Lehtonen S. Stock market forecast 2024: a soft landing may not bring the gains you expect. https://www.investors.com/news/

Li B, Yin J, Zhang A, Zhang Z (2018) A precise tidal level prediction method using improved extreme learning machine with sliding data window. In: 2018 37th Chinese Control Conference (CCC), pp. 1787–1792

Li S, Jin X, Xuan Y, Zhou X, Chen W, Wang Y-X, Yan X (2019) Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. Advances in neural information processing systems **32**

Oreshkin BN, Carpov, D, Chapados N, Bengio Y (2019) N-beats: Neural basis expansion analysis for interpretable time series forecasting. arXiv preprint arXiv:1905.10437

Pai P-F, Liu C-H (2018) Predicting vehicle sales by sentiment analysis of twitter data and stock market values. IEEE Access 6:57655–57662

Pizio AD. History says the Nasdaq Could Soar in 2024 – 5 stocks you'll wish you'd bought if it does. https://www.nasdaq.com/articles/

Protopapas P, Glickman M (2019) Lecture 10: Recurrent neural networks. URL: CS109B/lectures/lecture10/presentation/cs109b_lecture10_RNN.pdf

Rangapuram SS, Seeger MW, Gasthaus J, Stella L, Wang Y, Januschowski T (2018) Deep state space models for time series forecasting. Advances in neural information processing systems **31**

Ruder S (2016) An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747

Saiktishna C, Sumanth NSV, Rao MMS, J T (2022) Historical analysis and time series forecasting of stock market using fb prophet. In: 2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 1846–1851

Salinas D, Flunkert V, Gasthaus J (2017) Deepar: pobabilistic forecasting with autoregressive recurrent networks. Int J Forecast 36:1181–1191

Thu HGT, Thanh TN, Quy TL (2021) A neighborhood deep neural network model using sliding window for stock price prediction. In: 2021 IEEE International Conference on Big Data and Smart Computing (BigComp), pp. 69–74

Vaillancourt MBRAR (2013) Convolution theorems for quaternion fourier transform: Properties and applications. Abstract and Applied Analysis, 1–10

Wang N, Cai X, Xu Z, Guo Y, Zhang T (2023) Short-term wind power probabilistic forecasting based on attention mechanism and bidirectional lstm. In: 2023 International Conference on Smart Electrical Grid and Renewable Energy (SEGRE), pp. 188–192

Wu N, Green B, Ben X, O'Banion S (2020) Deep transformer models for time series forecasting: The influenza prevalence case. arXiv preprint arXiv:2001.08317

Yang Y (2018) Convolutional neural networks with recurrent neural filters. arXiv preprint arXiv:1808.09315

Zhou W, Wang D, Li H, Song W (2018) Long-term forecasting of time series based on sliding window information granules and fuzzy inference system. In: 2018 5th International Conference on Information, Cybernetics, and Computational Social Systems (ICCSS), pp. 375–380

Zhou H, Zhang S, Peng J, Zhang S, Li J, Xiong H, Zhang W (2021) Informer: Beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 11106–11115