Financial Time Series Analysis using Deep Learning Methods

Harishankar M Kartik Agrawal Komaragiri Sai Pranav Hiya Mehta Dheeraj M

Abstract

Financial time series data, characterized by its non-linear and dynamic nature, poses significant challenges for accurate forecasting. Traditional forecasting methods, such as ARIMA and GARCH, often struggle to capture the complexities inherent in financial markets, however the recent advancements in deep learning, particularly through the use of RNNs, LSTMs, and Transformer models, have shown promise in addressing these challenges. In our work, we implement LSTMs, GRUs, Transformer and Hybrid models for recursive as well as non-recursive predictions. We observed that Hybrid model performs better than the individual models, while transformer model proved to be efficient for long term, recursive prediction. However, the continuous generation and shifts in market regimes present significant challenges for predictive modeling, leading to issues of data and concept drift that can degrade model performance over time. To address these challenges, continual learning strategies are proposed, enabling models to incrementally learn from new data while retaining knowledge from previous tasks.[1].

1. Introduction

The finance industry has long been invested in accurately forecasting financial time series data. Numerous studies have demonstrated that machine learning (ML) models often outperform traditional time series forecasting methods. Concurrently, the pervasive adoption of automated electronic trading systems, along with growing demands for enhanced returns, continues to drive researchers and practitioners to refine and develop more advanced predictive models. In recent years, deep learning has emerged as the leading predictor class within the field of machine learning, demonstrating top performance across various application areas.

The major problem of interest of these models is to predict the next movement of the underlying asset. Even though there are several subtopics of this general problem, including stock price forecasting, index prediction, forex price prediction, commodity (oil, gold, etc.) price prediction, and cryptocurrency price forecasting, the underlying dynamics are the same in all of these applications. Studies can also be clustered into two main groups based on their expected outputs: price prediction and price movement (trend) prediction. In the modelling framework, price prediction is a regression problem, and trend prediction is a classification problem in the fame.

Traditional methods for forecasting stock prices can be broadly classified into linear and non-linear algorithms. Linear algorithms include AR (AutoRegressive), MA (Moving Average), and ARIMA (AutoRegressive Integrated Moving Average), which assume that future prices are linearly related to past prices. Non-linear algorithms include ARCH (AutoRegressive Conditional Heteroskedasticity), GARCH (Generalized ARCH), and Neural Networks, which can handle more complex, non-linear relationships in the data. These methods typically focus on predicting a company's stock price using historical daily closing prices.

However, these techniques(AR, MA, & ARIMA) are based on the assumption that the underlying system is linear and the data is stationary. Even the use of non-linear techniques are not efficient in extracting the underlying features or learning proper representations of the data. Adding to this, traditional models are generally static once trained, which makes them unable to learn or adapt in real time as new data arrives. This is particularly problematic in financial markets where new information continuously affects prices.

Deep learning (DL)-based methodologies provide an effective solution to overcome the challenges associated with traditional methods. Due to their strong ability to process big data and learn nonlinear relationships between input features and predicted targets, DL models perform better in prediction tasks than linear and ML models in the financial field. They can also incorporate multiple features beyond historical prices, like trading volume, news sentiment, social media trends, and macroeconomic indicators. Through this, they gain a more holistic understanding of stock price movements and adapt quickly to changing market conditions.

[1]Our implementation is provided here

Although existing DL frameworks are highly effective in learning representations and analyzing or forecasting financial time series, they still face notable challenges due to the dynamic nature of financial markets. The models are typically trained and tested on historical market data, which can be a significant drawback. Below are the two key challenges associated with this draw back:

- **Continuous generation of market data**:Financial market data is generated continuously, necessitating that models adapt and update regularly to remain effective. However, maintaining historical knowledge while incorporating new data is computationally challenging. Retraining models with both old and new data for every update is not a feasible solution.
- **Shift in market regimes**: Market regimes represent distinct periods in financial markets, each characterized by unique economic conditions, trends, or behaviors. Changes in market regimes can significantly alter the statistical properties of the data. If models do not account for these shifts, their effectiveness and accuracy may deteriorate.

The above two key challenges present themselves as issues of data and concept drift due to which the model's performance degrades over time. These challenges can be overcome by inculcating continual learning strategies into the existing learning frameworks. Through continual learning strategies, the model is trained incrementally on a sequence of tasks while retaining the knowledge gained from previous tasks. Unlike traditional learning approaches that assume access to a static dataset, continual learning allows models to learn and adapt over time, mitigating the problem of catastrophic forgetting. Fig.1 gives an overall outlook of the continual learning framework.

2. Literature Review

The research on various financial forecasting methodologies is briefly summarised in this section. Keeping in mind our study, more importance was given to the deep learning frameworks rather than the traditional frameworks.

2.1. Traditional Techniques

Traditional statistical methods, including auto regressive models [43], hidden Markov models [9] and tree models [1] have been extensively studied over the years for their interpretability of the data. However, they fail in the presence of significant noise, uncertain and dynamic factors in the market, and the need for longer prediction horizons in stock price forecasting [21].

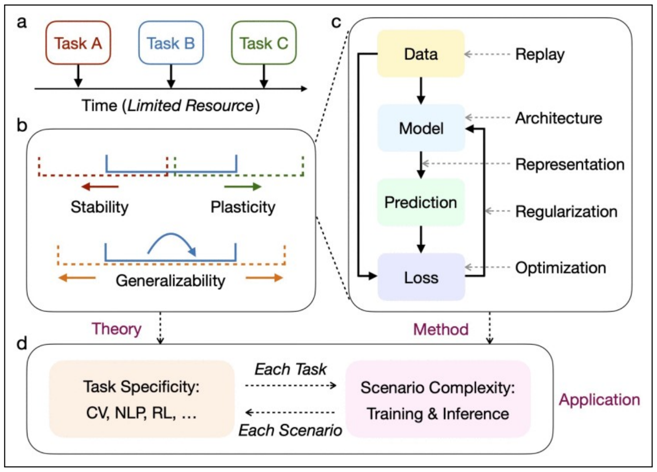

Figure 1. Conceptual framework of continual learning [41]. (a), Continual learning requires adapting to incremental tasks with dynamic data distributions. (b), A desirable solution should ensure an appropriate trade-off between stability (red arrow) and plasticity (green arrow), (c), and to achieve the objective of continual learning, representative methods have targeted various aspects of machine learning. Continual learning is adapted to practical applications.

2.2. Deep Learning Based Techniques

2.2.1 Convolutional Neural Networks (CNNs)

CNNs, traditionally used in image processing, have been effectively applied to stock market prediction for their ability to uncover hidden patterns in time series data. In [13], CNN's effectiveness on Tata Motors stock data from the NSE was demonstrated, showing that CNN outperformed ARIMA, MLP, RNN, and LSTM models in terms of prediction accuracy. Another study [34] used CNNs to analyze minute-wise stock data from 1,721 companies, achieving superior results by focusing on hidden patterns rather than historical data trends. Additionally, CNNs have been applied to event-driven prediction by integrating news events into the model, capturing both short and long-term effects on stock prices [5].

2.2.2 RNNs and LSTMs

Recurrent Neural Networks (RNNs) are widely used for sequential data, outperforming traditional feedforward networks in detecting patterns in such data by incorporating feedback mechanisms that simulate human long-term and short-term memory [31]. Unlike feedforward networks where information flows in one direction, RNNs use loops to pass information to themselves, allowing them to account for previous inputs. This makes RNNs one of the most common DL models for stock price prediction [35]. Additionally, RNNs share the same weight parameters across layers, unlike feedforward networks, where each layer has unique weights.

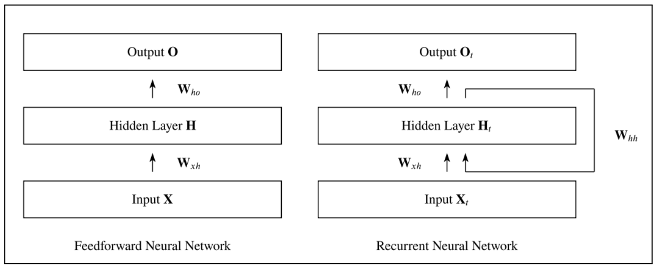

Figure 2. Difference between feedforward networks and RNNs[33]

Despite their merits, RNNs have a few drawbacks. They require significant amount of memory and compute resources, making implementation challenging. Additionally, activation functions like $tanh$ can cause vanishing gradients during backpropagation, leading to negligible updates for earlier layers. On the other end, repeated multiplication due to long sequences can lead to exploding gradients, causing the model to wildly oscillate.[36]

As a consequence, Long Short-Term Memory (LSTMs) and Gated Recurrent Units (GRU) were developed. These architectures incorporate mechanisms (called gates) to regulate the flow of information and gradients, mitigating both problems mentioned above. Unlike standard RNNs with a single repeating module, LSTMs consist of three gates—forget, input, and output—that interact to manage memory and learning more effectively.

Sonkavde et al. [38] performed an extensive review and comparison between various existing RNN based models for time series analysis and forecasting such as: LSTM, BiLSTM, GRU, BiGRU models and the ARIMAX. They conclude that bidirectional RNN and ARIMAX models present lowest forecasting errors and overall perform better than LSTMs and GRUs.

2.2.3 Hybrid Models

Hybrid models combine different neural network architectures, such as LSTM, CNN, and GRU, to leverage the strengths of each. These models offer a more comprehensive analysis by capturing both short-term patterns and long-term dependencies in time series data. Hybrid approaches often outperform individual models due to their ability to integrate diverse learning techniques, which enhances prediction accuracy and robustness in complex datasets.

In [14], Maximum Entropy (ME), Support Vector Regression (SVR), and Artificial Neural Network (ANN) based trend models were used for forecasting financial time series. A hybrid model combining LSTM and GRU network outperformed individual models and standard neural networks in predicting long-term stock price changes [3]. Similarly, a hybrid model combining multivariate LSTM with ARIMA has shown superior performance compared to other methods [10]. Hybrid models combining CNN and LSTM have also shown promising results [46]. In another study, the authors proposed a hybrid model combining Temporal Convolutional Network (TCN), CNN-LSTM, and GRU, which outperformed individual models in predicting opening prices using CSI300 Index data [40].

2.2.4 Transformers

Transformer-based models which are already being widely used for natural language processing(NLP) and computer vision(CV) applications, is picking up pace for stock price predictions [30] tasks as well. Transformers have been proved to possess strong modelling abilities to capture long-time dependencies in sequential data [42], and thus been studied for stock predictions [44]. They have already been used to forecast S&P volatility [30] as well as used on natural language data from social media and news for sentiment analysis based stock price predictions [19][39].

Muhammad et al.[24] utilized *time2vec* encoding mechanism in their transformer model achieving low error rates within a short execution time and predicted future prices over both short and long-time horizons. MASTER[17], a MArket-Guided Stock TransformER, models the momentary and cross-time stock correlation and leverages market information for automatic feature selection. Specifically, MASTER achieves an average improvement of 13% on ranking metrics and a remarkable 47% on portfolio-based metrics, when compared with other baseline models.

2.3. Continual Learning Strategies

Wang et al.[41] have given a comprehensive survey of various theory methodologies and applications associated with continual learning strategies. Continual learning addresses the stability-plasticity trade-off and the challenge of catastrophic forgetting, where a model forgets previously learned knowledge when acquiring new information. A well-designed continual learning system must retain essential past knowledge (stability) while adapting efficiently to new data or tasks (plasticity), ensuring that old and new information coexist effectively. This can be achieved in several ways grouped based on their approach into categories ([41]) :

- **Regularization based approach**([16],[18]):This direction is characterized by adding explicit regularization terms to balance the old and new tasks, which usually requires storing a frozen copy of the old model for reference.
- **Replay Based Methods**([28],[20]): These methods are

characterised by incorporating past data or synthetic representations of past experiences into the learning process. These approaches involve revisiting a subset of previously seen data (or approximations) while learning new information, allowing the model to retain old knowledge and adapt to new tasks or data.

- **Optimization-Based Approach**([6],[32]):This direction is characterized by explicit design and manipulation of the optimisation programs, such as gradient projection with reference to the gradient space or input space of the old tasks , meta-learning of sequentially arrived tasks within the inner loop etc.
- **Representation Based Approach**([25],[23]): This direction is characterised by creating and leveraging the strengths of representations for continual learning, such as by using self-supervised learning and pre-training.
- **Architecture-Based Approach**([15],[22]):This direction is characterized by constructing task-specific parameters with a properly designed architecture, such as assigning dedicated parameters to each task (parameter allocation), constructing task-adaptive sub-modules or sub-networks (modular network), and decomposing the model into task-sharing and task-specific components (model decomposition).

Continual learning strategies have found numerous use-case scenarios improving the existing deep learning frameworks ([12], [11], [26], [7], [4], [45], [8], [37]). To the relevance of financial time series forecasting Ragot et al.[2] have provided a comprehensive survey of existing trends of using continual learning strategies for time series analysis. Philips [27] has used continual learning, in general, to obtain investment decisions better. Ramjattan et al.[29] have compared continuous learning strategies in financial time series. They have reported a significant difference between traditional learning approaches and those using continual learning in the return profits (higher risk-to-return ratios). While some of the above works offer insights into the potential of continual learning strategies for time series, including financial data, their full potential remains untapped and warrants further research and in-depth study.

3. Datasets

The financial market data was collected from Yahoo Finance and loaded using the yfinance library in Python.

3.1. For Comparative Study:

For training and testing of the models, we used the daily closing price of the stock data of Tata Motors from 01-01-2010 to 20-10-2023 (Fig.3). The closing price is the final price at which a stock is traded during a regular trading session, typically reflecting the most recent market value at the end of the day. The data is divided into train and test data in the ratio of 70:30. The train data is further split in the ratio of 80:20 as the actual training set and validation set, respectively. The dataset was normalized using min-max normalization.

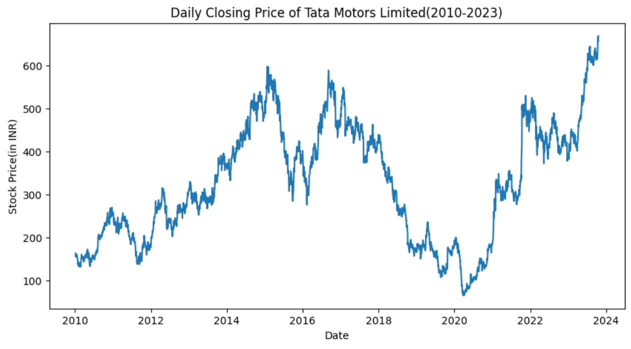

Figure 3. Stock data: Tata Motors Limited (2010-2023)

3.2. For Continual Learning Models:

For continual learning models, we utilised the Bitcoin price dataset, sampled every 5 minutes over a period of 60 days, representing short-term and high-frequency market fluctuations(Fig.4). The cryptocurrency dataset was chosen for its high liquidity and volatility, and to enable intra-day trading, we look at the asset's value every 5 minutes and try to learn the evolving trends within a few days. The dataset was prepared using a look back window of 60-time steps to capture temporal dependencies in the price trends. The data was divided into train and test sets in a 70:30 ratio, with the training set further split into training and validation sets in a 90:10 ratio. Min-max normalization was applied to scale the data to a uniform range, ensuring efficient model training and performance stability.

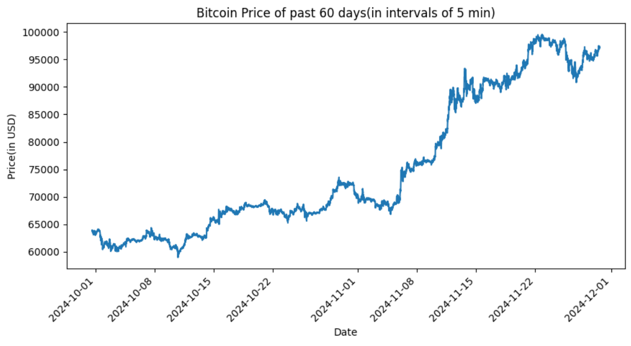

Figure 4. Bitcoin Price data for the past 60 days

4. Models

4.1. LSTMs

The architecture includes two LSTM layers with 10 and 20 units, using Tanh and ReLU activations, respectively, and dropout rates of 0.2 and 0.3 to prevent overfitting. A dense output layer handles prediction, with the model comprising 2,981 parameters. This architecture captures both short-term and long-term dependencies in the data, with dropout layers introduced to enhance generalization.

4.2. GRU

The architecture consists of three stacked GRU layers with 16 units each, using Tanh activations, followed by a dropout layer(0.2) and a dense output layer for predictions. The model consists of 4,193 total parameters.

4.3. Transformer

The Transformer architecture includes a single encoder layer with 4 attention heads (head size: 256) and feed-forward layers with a hidden dimension of 4 and ReLU activation. Residual connections and dropout (rate: 0.1) enhance stability and regularization. Global average pooling, followed by dense layers with 20 and 1 units, performs feature extraction and prediction. The model, comprising 9,027 parameters, is optimized using Adam with MSE loss for effective sequence modeling.

4.4. Hybrid (LSTMs + GRU)

The model is a sequential neural network that combines LSTM and GRU layers. The architecture consists of two LSTM layers with 10 and 20 units, and a GRU layer with 10 units, with all the layers using Tanh activation. All layers have dropout regularization and the output layer has a dense layer for prediction. The hybrid model consists of 3,931 total parameters.

5. Continual Learning Framework

We have utilised the methods of Elastic Weight Consolidation (EWC), Learn Without Forgetting (LWF) (both regularization-based approaches) and Gdumb(replay-based approach) to enable incremental learning for our deep learning framework. Apart from these methods we also implemented methods which don't incorporate any particular methodology to mitigate catastrophic forgetting like Joint training, Naive training and Re incremental training.

5.1. Elastic Weight Consolidation (EWF)

Elastic Weight Consolidation ([16]) addresses catastrophic forgetting by adding a regularisation term to the loss function, which penalises deviations of important parameters from their previously optimised values:

$$\mathcal{L}_{EWC} = \mathcal{L}_{current} + \frac{\lambda}{2}\sum_i F_i(\theta_i - \theta_i^*)^2,$$

where $\mathcal{L}_{current}$ is the loss for the current task, θ_i are the model parameters, θ_i^* are the optimal parameters from the previous task, F_i (the Fisher Information Matrix) quantifies the importance of each parameter, and λ controls the regularization strength.

The Fisher Information Matrix is computed as:

$$F_i = E\left[\left(\frac{\partial \log p(y|\mathbf{x},\theta)}{\partial \theta_i}\right)^2\right],$$

where $p(y|\mathbf{x},\theta)$ is the likelihood of the model output $\mathbf{y}$ given input $\mathbf{x}$ and parameters θ. This ensures that changes to highly important parameters (large F_i values) are penalised more heavily, allowing the model to retain critical knowledge from previous tasks while learning new ones.

5.2. Learn Without Forgetting (LWF)

Learn Without Forgetting ([18]) addresses catastrophic forgetting by using Knowledge Distillation loss in regularization. This can be implemented by minimising the Mean Squared Error (MSE) between the outputs of a model prior to each epoch (teacher model) and a student model, which gets updated during the epoch (training in batches). The student learns to approximate these predictions (not losing previous information) while fitting the new data.

The Knowledge Distillation loss for regression is defined as the MSE between the teacher's and student's outputs:

$$\mathcal{L}_{KD} = \frac{1}{n}\sum_{i=1}^{n}(y_t(i) - y_s(i))^2, \tag{1}$$

where: - $y_t(i)$ is the teacher's output for sample i, - $y_s(i)$ is the student's output for the same sample.

Additionally, the standard regression loss (MSE) between the student's prediction and the true data labels is:

$$\mathcal{L}_{\text{ground}} = \frac{1}{n}\sum_{i=1}^{n}(y_{\text{true}}(i) - y_s(i))^2, \tag{2}$$

where $y_{\text{true}}(i)$ is the ground truth value for sample i.

The total loss function is a weighted sum of the ground truth loss and the distillation loss:

$$\mathcal{L} = \mathcal{L}_{\text{ground}} + \alpha\mathcal{L}_{KD}, \tag{3}$$

where α is a hyperparameter that controls the balance between the regression loss and the distillation loss.

The student model is trained to minimise this total loss, which allows it to stay close to the teacher's predictions (not forgetting older trends) while also fitting the true labels.

5.3. Gdumb

GDumb ([28]) is a continual learning framework designed to address catastrophic forgetting with a simple and generalisable strategy. It employs a greedy balancing sampler to maintain a memory of samples from a data stream, ensuring class balance within a fixed memory budget. At inference, the model is retrained from scratch using only the stored samples, a strategy that avoids overfitting to earlier tasks and accommodates the evolving label space. Unlike traditional methods, GDumb imposes minimal assumptions about task boundaries, the ordering of samples, or the nature of the label space, enabling its application across diverse continual learning formulations.

Naive Training

In the naive training approach, the model is trained sequentially on new data, overwriting any knowledge acquired from previous tasks. This simplicity comes at the cost of catastrophic forgetting, as the model does not retain or utilize information from past data during subsequent training.

Joint Training

Joint training combines all past and new data for retraining the model, ensuring that knowledge from all tasks is preserved. While this method is effective at maintaining task performance, it is computationally expensive and demands substantial storage resources, making it impractical for large-scale continual learning.

Re-incremental Training

Re-incremental training focuses on training a new model solely on the incremental data, reducing the training complexity due to the smaller dataset size. However, this approach may fail to capture historical trends and patterns, as it disregards past data during model updates.

6. Training and Evaluation

6.1. For comparative analysis across models

The model aims to predict the stock's closing price on the following day if the stock's closing price for the last *d* days is given. This number of days considered while training and prediction is called the look-back window. The dataset is sampled into batches with the size of d days, with the label being the closing price of the next day corresponding to the last day in the window. Two different kinds of testing are done: Recursive and Non-Recursive. In Non-Recursive testing, we take *d* days of actual test data and predict the next day's price. This process of taking the actual data is done by shifting the window being considered and then taking the actual prices. On the other hand, in Recursive testing, we try to predict a general trend over a longer period (60 days was considered). This is done by taking the last *d* known prices in the training dataset and predicting all the next days by recursively using the predicted output instead of actual test data. We use the following evaluation metrics to assess the model's efficiency in predicting the closing price of the stock:

Mean Absolute Error (MAE): Indicates the average magnitude of the prediction errors, giving a sense of how far off predictions are from the true values on average.

R squared score(R^2): Measures how well the model explains the variance in the target variable, with values closer to 1 indicating better fit and negative values indicating poor performance.

$$\mathbf{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

$$\mathbf{R^2} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where:
- n is the total number of data points.
- y_i is the true value for the i-th data point.
- $\hat{y}_i$ is the predicted value for the i-th data point.
- $\bar{y}$ is the mean of the true values, i.e., $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$.

6.2. For Continual Learning Methodologies

The different models, each encompassing a strategy discussed in the previous sections, are initially trained on a 30-day window (Baseline,29/09/24 - 29/10/24). Further, we incrementally train them on three time windows(Increment 1: 29/10/24 - 08/11/24, Increment 2: 08/11/24 - 18/11/24, Increment 3: 18/11/24 - 28/11/24). The data points were sampled in intervals of five minutes, and a lookback window of 60 timesteps was used while training. To understand the financial returns aspect through the models, we have implemented a simple investment strategy. The strategy involves predicting the next timestep's stock price using the model and taking buy, sell, or hold decisions based on the predicted price trend. Specifically, the stock is bought if the model predicts a price increase, sold if a decrease is predicted, and retained if no significant change is expected. The profit across the testing dataset is calculated, and the profit percentage is determined relative to the initial investment.

7. Results and Observations

7.1. Performance Analysis of Models

We analyse the performance of LSTM, GRU, Transformer and Hybrid models upon training. Fig.5 shows the visual comparisons of Tata Motors actual and predicted stock closing prices for Transformer and hybrid models. Table 5 provides the evaluation metrics of the four models alongside

Table 1. Baseline (29/09/24 - 29/10/24) Performance

Model Type	R^2 Score	RMSE	Profit (%)
EWC	0.814	386.177	-1.296
LWF	0.493	609.309	-9.115
Gdumb	0.810	353.173	-5.339
Naive Training	-0.102	842.609	0.548
Joint Training	0.902	280.363	3.542
Reincrement Training	-13.991	3574.31	-6.487

the number of trainable parameters for Tata Motors stock prices. Among the models, the hybrid model demonstrates superior performance in terms of evaluation metrics when compared with the standalone models. We also experimented with increasing the number of layers to improve accuracy, but this made it more challenging to prevent overfitting. Further, we analyzed the effect of varying the lookback window size and observed that LSTM performs better than other models for a lower lookback window. For larger lookback window frame, the performance of all the model improves.

We also analyse the plots for recursive prediction using lookahead (Fig.6), and observe that the Transformer model performs better than the other models in terms of following the actual value. Hence, for prediction of long term trends, transformer would be the optimal choice while Hybrid would be the choice for short term prediction.

7.2. Performance Analysis of Continual Learning methodologies

We analysed the performance of various strategies to enable data incremental learning. Along with it, we trained models without using any mitigation strategies to prevent catastrophic forgetting, such as naive training and re-incremental training. Table 1, Table2, Table3 Table4 summarizes the results obtained. It was observed that all the strategies used to enable continual learning, the continual learning strategies (EWC, LWF and Gdumb) have given comparable or better results when compared to joint training, naive training and re-incremental training. Among all the methods, EWC has shown superior performance in all incremental training instances. In terms of returns obtained via investment (profit per cent), the continual learning strategies have given comparable performance to others. The poor results in re-incremental training are an indication that the time series have shown historical data trends and can't be just retrained on an incremental data instance. The superiority of continual learning strategies over joint training has proven that simply training on the entire dataset maynot give us goofd performance as the data may might have more dependency over the recent data .

Table 2. Increment 1 (29/10/24 - 08/11/24): Data Performance

Model Type	R^2 Score	RMSE	Profit (%)
EWC	0.864	1002.053	-6.206
LWF	-6.875	2291.936	-1.073
Gdumb	-0.991	1160.059	-1.415
Naive Training	0.810	1186.7305	-6.737
Joint Training	0.789	1252.121	-4.817
Reincrement Training	-0.196	2980.473	-4.481

Table 3. Increment 2 (08/11/24 - 18/11/24): Data Performance

Model Type	R^2 Score	RMSE	Profit (%)
EWC	0.964	181.57	3.100
LWF	0.577	350.195	1.458
Gdumb	0.529	222.440	0.213
Naive Training	0.384	754.77	-1.389
Joint Training	-49.609	6840.400	-2.615
Reincrement Training	0.1445	889.36	0.028

Table 4. Increment 3 (18/11/24 - 28/11/24): Performance

Model Type	R^2 Score	RMSE	Profit (%)
EWC	0.991	200.405	6.118
LWF	0.961	274.636	3.122
Gdumb	0.936	353.083	0.908
Naive Training	0.963	400.199	-2.726
Joint Training	0.869	752.470	8.042
Reincrement Training	0.409	1592.585	2.402

8. Summary and Future Work

Deep learning has revolutionized financial time series forecasting, with extensive research currently focusing on enhancing models by incorporating real-time factors like sentiment analysis and external market dynamics. However, stock price prediction remains inherently complex, as prices are influenced by a multitude of factors, including global markets, political events, and social media trends.

In this work, we concentrated on predicting stock prices using only historical closing price data, achieving high prediction accuracy. Through our literature survey, we found that most existing implementations rely on LSTMs, GRUs, and Transformers. We extended this by experimenting with lookahead-based predictions and exploring the use of continual learning. Several continual learning strategies were implemented, demonstrating notable improvements in prediction accuracy and adaptability to evolving financial trends.

For future work, integrating sentiment analysis using natural language processing (NLP) and exploring intra-stock correlations through graph neural networks (GNNs) could provide more realistic and robust predictions. Ad-

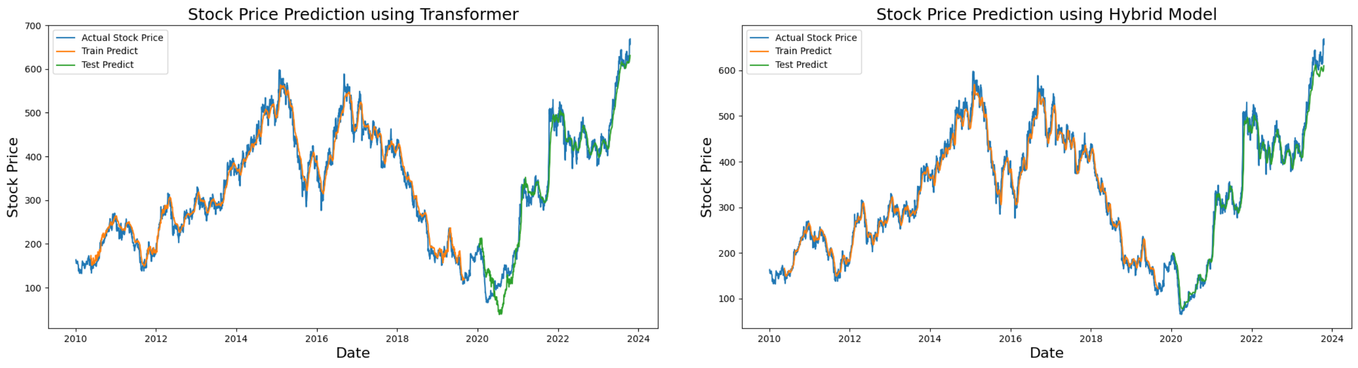

Figure 5. Comparative study for non-recursive predictions (lookback = 90)

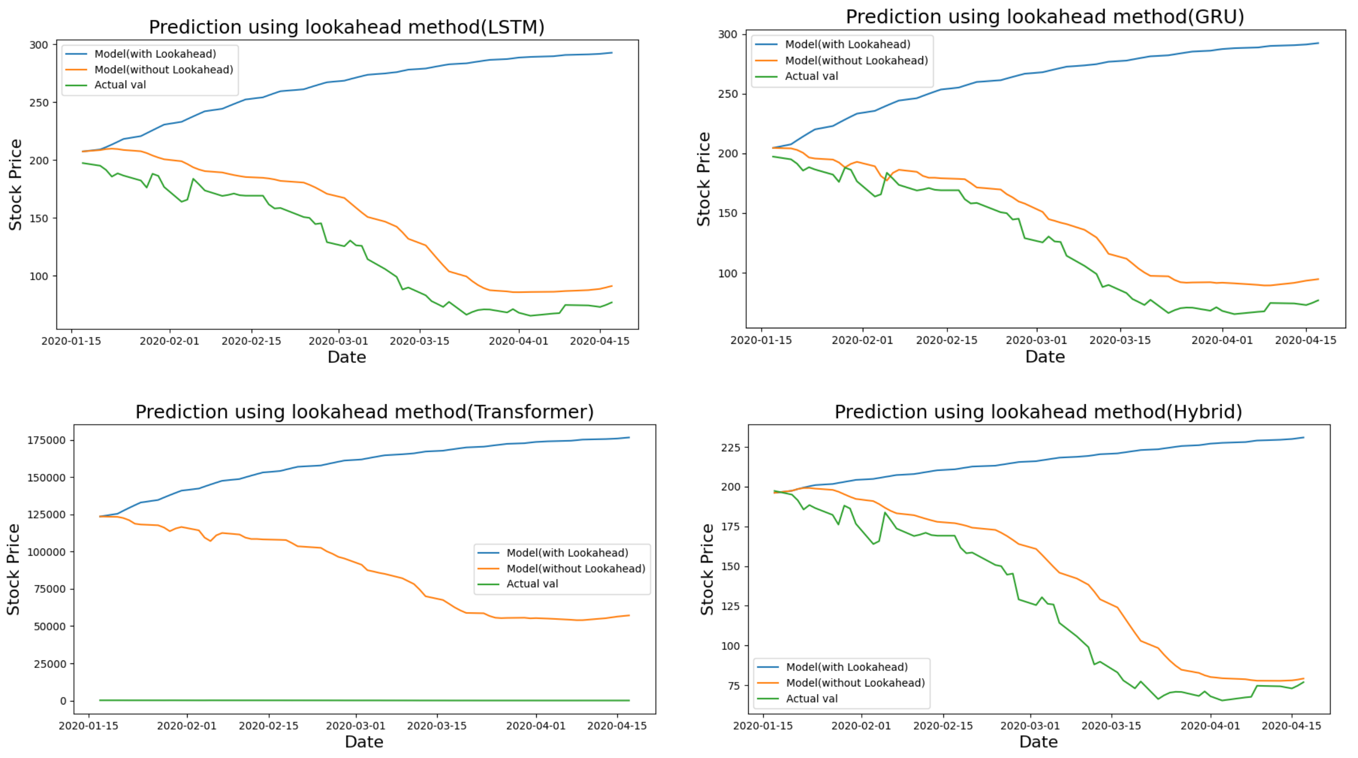

Figure 6. Prediction: Recursive and Non Recursive for 60 days (lookback = 90)

ditionally, developing investment strategies based on aggregated trend predictions from multiple models can further enhance decision-making processes in financial markets. Integration of multiple continual learning strategies such as EWC and replay-based methods are also under consideration.

References

[1] Ernest Kwame Ampomah, Zhiguang Qin, and Gabriel Nyame. Evaluation of tree-based ensemble machine learning models in predicting stock price direction of movement. *Information*, 11(6):332, 2020. 2

[2] Quentin Besnard and Nicolas Ragot. Continual learning for time series forecasting: A first survey. *Engineering Proceedings*, 68(1), 2024. 4

[3] A. Bhavani, A. Venkata Ramana, and A. S. N. Chakravarthy. Comparative analysis between lstm and gru in stock price

Parameters	**LSTM (Params=4,193)**			**GRU (Params=2,981)**			**Transformer (Params=7,827)**			**Hybrid (Params=3,931)**		
Lookback Window (days)	30	60	90	30	60	90	30	60	90	30	60	90
RMSE	18.938	23.986	23.782	20.368	24.079	24.885	31.824	31.054	30.30	24.509	21.765	21.765
R2 Score	0.986	0.978	0.977	0.984	0.977	0.975	0.960	0.962	0.964	0.977	0.981	0.981
Time per epoch(s)	1.65	3.43	4.27	2.41	7.41	9.64	0.96	9.81	19.34	2.18	9.53	10.12
Epochs	50	50	50	50	50	50	50	50	50	50	50	50

Table 5. Performance comparison of LSTM, GRU, Transformer, and Hybrid Models.

prediction. In *2022 International Conference on Edge Computing and Applications (ICECAA)*, pages 532–537, 2022. 3

[4] Fabio Cermelli, Massimiliano Mancini, Samuel Rota Bulo, Elisa Ricci, and Barbara Caputo. Modeling the background for incremental learning in semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9233–9242, 2020. 4

[5] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Deep learning for event-driven stock prediction. In *Twenty-fourth international joint conference on artificial intelligence*, 2015. 2

[6] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR, 2020. 4

[7] Mohammad Navid Fekri, Harsh Patel, Katarina Grolinger, and Vinay Sharma. Deep learning for load forecasting with smart meter data: Online adaptive recurrent neural network. *Applied Energy*, 282:116177, 2021. 4

[8] Tao Feng, Mang Wang, and Hangjie Yuan. Overcoming catastrophic forgetting in incremental object detection via elastic response distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9427–9436, 2022. 4

[9] Md Rafiul Hassan and Baikunth Nath. Stock market forecasting using hidden markov model: a new approach. In *5th international conference on intelligent systems design and applications (ISDA'05)*, pages 192–196. IEEE, 2005. 2

[10] Chenxi He. A hybrid model based on multi-lstm and arima for time series forcasting. In *2023 8th International Conference on Intelligent Computing and Signal Processing (ICSP)*, pages 612–616. IEEE, 2023. 3

[11] Yujiang He and Bernhard Sick. Clear: An adaptive continual learning framework for regression tasks. *AI Perspectives*, 3 (1):2, 2021. 4

[12] Yujiang He, Janosch Henze, and Bernhard Sick. Continuous learning of deep neural networks to improve forecasts for regional energy markets. *IFAC-PapersOnLine*, 53(2):12175–12182, 2020. 21st IFAC World Congress. 4

[13] MEAG Hiransha, E Ab Gopalakrishnan, Vijay Krishna Menon, and KP Soman. Nse stock market prediction using deep-learning models. *Procedia computer science*, 132: 1351–1362, 2018. 2

[14] Dong Huang, Xiaolong Wang, Jia Fang, Shiwen Liu, and Ronggang Dou. A hybrid model based on neural networks for financial time series. In *2013 12th Mexican International Conference on Artificial Intelligence*, pages 97–102, 2013. 3

[15] Sangwon Jung, Hongjoon Ahn, Sungmin Cha, and Taesup Moon. Continual learning with node-importance based adaptive group sparse regularization. *Advances in neural information processing systems*, 33:3647–3658, 2020. 4

[16] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 3, 5

[17] Tong Li, Zhaoyang Liu, Yanyan Shen, Xue Wang, Haokun Chen, and Sen Huang. Master: Market-guided stock transformer for stock price forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(1):162–170, 2024. 3

[18] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017. 3, 5

[19] Jintao Liu, Hongfei Lin, Xikai Liu, Bo Xu, Yuqi Ren, Yufeng Diao, and Liang Yang. Transformer-based capsule network for stock movement prediction. In *Proceedings of the first workshop on financial technology and natural language processing*, pages 66–73, 2019. 3

[20] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017. 3

[21] Wenjie Lu, Jiazheng Li, Jingyang Wang, and Lele Qin. A cnn-bilstm-am method for stock price prediction. *Neural Computing and Applications*, 33(10):4741–4753, 2021. 2

[22] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European conference on computer vision (ECCV)*, pages 67–82, 2018. 4

[23] Sanket Vaibhav Mehta, Darshan Patil, Sarath Chandar, and Emma Strubell. An empirical investigation of the role of pre-training in lifelong learning. *Journal of Machine Learning Research*, 24(214):1–50, 2023. 4

[24] Tashreef Muhammad, Anika Bintee Aftab, Muhammad Ibrahim, Md. Mainul Ahsan, Maishameem Meherin Muhu, Shahidul Islam Khan, and Mohammad Shafiul Alam. Transformer-based deep learning model for stock price prediction: A case study on bangladesh stock market. *Interna-*

tional Journal of Computational Intelligence and Applications, 22(03):2350013, 2023. 3

[25] Quang Pham, Chenghao Liu, and Steven Hoi. Dualnet: Continual learning, fast and slow. *Advances in Neural Information Processing Systems*, 34:16131–16144, 2021. 4

[26] Quang Pham, Chenghao Liu, Doyen Sahoo, and Steven CH Hoi. Learning fast and slow for online time series forecasting. *arXiv preprint arXiv:2202.11672*, 2022. 4

[27] Daniel Philps. *A temporal continual learning framework for investment decisions*. PhD thesis, City, University of London, 2020. 4

[28] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 524–540. Springer, 2020. 3, 6

[29] Reshawn Ramjattan, Daniele Atzeni, and Daniele Mazzei. Comparative evaluation of continual learning methods in financial and industrial time-series data. In *2024 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2024. 4

[30] Eduardo Ramos-Pérez, Pablo J. Alonso-González, and José Javier Núñez-Velázquez. Multi-transformer: A new neural network-based architecture for forecasting sp volatility. *Mathematics*, 9(15), 2021. 3

[31] Nesma M. Rezk, Madhura Purnaprajna, Tomas Nordström, and Zain Ul-Abdin. Recurrent neural networks: An embedded computing perspective. *IEEE Access*, 8:57967–57996, 2020. 2

[32] Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. *arXiv preprint arXiv:2103.09762*, 2021. 4

[33] Robin Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview, 2019. 3

[34] Sreelekshmy Selvin, R Vinayakumar, E. A Gopalakrishnan, Vijay Krishna Menon, and K. P. Soman. Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1643–1647, 2017. 2

[35] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. *Applied Soft Computing*, 90:106181, 2020. 2

[36] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020. 3

[37] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. Incremental learning of object detectors without catastrophic forgetting. In *Proceedings of the IEEE international conference on computer vision*, pages 3400–3409, 2017. 4

[38] Gaurang Sonkavde, Deepak Sudhakar Dharrao, Anupkumar M. Bongale, Sarika T. Deokate, Deepak Doreswamy, and Subraya Krishna Bhat. Forecasting stock market prices using machine learning and deep learning models: A systematic review, performance analysis and discussion of implications. *International Journal of Financial Studies*, 11(3), 2023. 3

[39] Priyank Sonkiya, Vikas Bajpai, and Anukriti Bansal. Stock price prediction using bert and gan. *arXiv preprint arXiv:2107.09055*, 2021. 3

[40] Weijie Wan, Qingzhen Xu, Huiyan Chen, and Qiang Chen. Using deep learning neural networks and stacking ensemble learning to predict csi 300 index. In *2022 9th International Conference on Digital Home (ICDH)*, pages 81–86. IEEE, 2022. 3

[41] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 2, 3

[42] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey, 2023. 3

[43] Chun Shan Wong and Wai Keung Li. On a mixture autoregressive model. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 62(1):95–115, 2000. 2

[44] Linyi Yang, Tin Lok James Ng, Barry Smyth, and Riuhai Dong. Html: Hierarchical transformer-based multi-task learning for volatility prediction. In *Proceedings of The Web Conference 2020*, page 441–451, New York, NY, USA, 2020. Association for Computing Machinery. 3

[45] Chang-Bin Zhang, Jia-Wen Xiao, Xialei Liu, Ying-Cong Chen, and Ming-Ming Cheng. Representation compensation networks for continual semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7053–7064, 2022. 4

[46] Xinrong Zhou. Stock price prediction using combined lstm-cnn model. In *2021 3rd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, pages 67–71. IEEE, 2021. 3