CLAM: A Synergistic Deep Learning Model for Multi-Step Stock Trend Forecasting

Nguyen Quoc Anh[1](✉) and Phan Thi Quynh Nhu[2]

[1] Department of Economics and Finance, The Business School, RMIT University Vietnam, Ho Chi Minh City, Vietnam
s3926339@rmit.edu.vn

[2] Department of Finance and Banking, Ho Chi Minh City University of Foreign Languages - Information Technology, Ho Chi Minh City, Vietnam
21dh202557@st.huflit.edu.vn

Abstract. This paper introduces CLAM, a hybrid deep learning framework that integrates CNNs, LSTMs, and Attention Mechanism (AM) for straightforward multi-step stock trend forecasting. By leveraging CNNs for spatial feature extraction, LSTMs for capturing temporal dependencies, and AM for dynamically focusing on relevant data, CLAM significantly outperforms traditional models in predictive accuracy. Evaluated on diverse stock datasets from different industries, CLAM demonstrates an average reduction of over 80% in MAE and RMSE compared to standalone CNN, LSTM, and fused CNN-LSTM. The model's ability to capture both short-term and long-term trends is particularly advantageous for real-time financial trading, resulting in 75% trend prediction accuracy, with most cases witnessing consecutive accurate forecasts of flash crashes or uptrends. Code and data are available at: https://github.com/TheQuantScientist/CNN-LSTM-AM.

Keywords: Deep learning · Hybrid model · Stock Trend · Attention mechanism · Financial trading · Trend prediction

1 Introduction

Concerning the evolving industrial landscape, the intersection of technological innovation and financial strategy has become vital [1]. Artificial Intelligence (AI) is leading this transformation, with advanced neural network architectures such as Convolutional Neural Networks (CNNs) [2], Recurrent Neural Networks (RNNs) [3], Long Short-Term Memory Networks (LSTMs) [4], and Transformers [5] driving advancements in healthcare [6] and meteorology [7] is well-recognized. Nevertheless, its influence on finance is the most promising [8] as the digital economy amplified the role of Machine Learning (ML) in financial markets, where predictive modeling is essential for forecasting stock prices [9] and informing investment strategies [10]. Recovering from the pandemic, the need for precise financial predictions has grown strongly [11]. Investors increasingly rely on algorithmic modeling to navigate through market complexities, thereby capitalizing on opportunities from quantitative trading of stocks [12].

Traditional models such as ARIMA (Autoregressive Integrated Moving Average) [13], SARIMA (Seasonal ARIMA) [14], and Linear Regression [15] are predicated on the assumption that historical patterns and trends can effectively predict future stock prices. However, the non-linear nature of stock movements often introduces significant biases in these models [16], which tend to oversimplify complex financial dynamics by assuming fixed data variance and linearity [17]. This simplification overlooks critical factors influencing stock prices over time, such as macroeconomic indicators [18], market sentiment [19], and herd behavior [20]. The emergence of Machine Learning (ML) and Deep Learning (DL) models has addressed many of these limitations [4]. Techniques like Support Vector Machines (SVM) [21], Decision Trees [22], and Random Forest Regression (RFR) [23] have advanced the ability to capture non-linear relationships by integrating multiple features and learning from large, complex datasets [24]. DL, as a subset of ML, has further transformed forecasting with deep neural architectures like CNNs [2], RNNs [3], and LSTMs [4], employing hierarchical end-to-end learning that excels in handling unstructured data and scaling with increasingly large time series datasets [25]. Additionally, hybrid models that combine elements of traditional statistical methods with ML and DL techniques are reshaping the forecasting landscape [26]. These hybrid approaches, by integrating multiple modalities, enhance feature extraction and provide deeper insights into data irregularities [27] for a more robust financial system. Nonetheless, despite their improved accuracy, ML and DL models bring challenges such as overfitting [28], interpretability issues [29], and the necessity for large datasets.

Building on this concept, this paper introduces the CLAM model to forecast the weekly trend of financial assets. CLAM is a hybrid deep learning architecture that combines stacked layers of **C**NN, **L**STM, and an **A**ttention **M**echanism (AM) to address the challenges identified in previous models. Our CLAM leverages CNNs for effective feature extraction, capturing spatial patterns in stock price data. LSTMs are then employed to manage temporal dependencies, effectively handling the sequence of data points over time. The attention mechanism further enhances the model by dynamically focusing on relevant features, allowing the model to prioritize critical information and mitigate the impact of less significant data. This combination of techniques enhances the model's ability to capture both short-term and long-term dependencies thus strengthening its robustness in handling non-linear relationships within the data. Nonetheless, recognizing the impracticality of forecasting stock price value, CLAM focuses on being a computationally efficient model by only focusing on equity movements, which is well-suited for real-time swing trading within volatile financial markets.

This paper is organized as follows: Section 2 reviews financial forecasting literature, emphasizing the limitations of traditional models and advances in hybrid approaches. Section 3 outlines the data processing, metrics, and the proposed CLAM architecture. Section 4 describes the experimental setup, model comparisons, and performance results. Section 5 discusses the findings and potential improvements. Finally, Section 6 summarizes and suggests directions for real-time financial trading applications. CLAM is built on our previous research [30–32].

2 Literature Review

2.1 Related Work

The stock market represents a complex system influenced by a wide range of often unrelated factors, such as psychological behavior and economic conditions. Among various forecasting models, the Back Propagation Neural Network (BP-NN) has been shown to outperform others, such as ARIMA and Random Forest Regression (RFR), in predicting the one-year stock prices of Chinese vaccine manufacturers [33]. This outcome is particularly beneficial for investors within the pharmaceutical sector. Additionally, a combination of Seasonal ARIMA and Extreme Gradient Boosting (SARIMA-XGBoost) has demonstrated impressive accuracy in forecasting the Indian Stock Index [34], reflecting the potential of hybrid models in achieving high predictive performance. Similarly, the RFR model has been effective in predicting stock prices of companies listed on Indian exchanges, further indicating the utility of ML models in financial forecasting [35]. These traditional ML approaches have proven effective, but the advent of Deep Learning, a subset of ML, has introduced a new level of precision in model development by autonomously learning hierarchical data representations [36]. To advance stock market prediction, Wang et al. [27] introduced a model that combines the LightGBM algorithm with wavelet packet decomposition (WPD) to filter out data noise before forecasting the Shanghai Composite Index. This hybrid approach successfully predicted the market trend over a 10-day period, surpassing the performance of ARIMA and Support Vector Regression (SVR) [27]. Furthermore, the integration of CNNs to enhance LSTM networks has led to major improvements in short-term prediction accuracy, showing a 25% increase on the CSI300 index [37]. A more advanced hybrid model, combining CNN, BiLSTM, and Efficient Channel Attention (ECA), showed strengths in predicting long-term trends by leveraging spatial and temporal data processing [38].

Alternatively, combining LSTM and Gated Recurrent Unit (GRU) networks has yielded superior predictions of the S&P500 adjusted closing price, outperforming models that rely solely on GRU, LSTM, or Multilayer Perceptron (MLP) architectures [39]. The success of these hybrid models underscores the importance of carefully engineered combinations, which can significantly enhance the precision and reliability of stock price forecasts [40]. Ultimately, constructing an effective forecasting model goes beyond selecting the most advanced architectures; it also requires the integration of appropriate mathematical and physical principles to analyze complex, chaotic systems that exhibit unpredictable and non-repetitive patterns due to their sensitivity to initial market conditions [41]. As a result, the inclusion of the Attention Mechanism (AM) has revolutionized the time series field [42]. AM allows models to focus on the most relevant features within the data across multiple time intervals by selectively weighting the importance of different inputs. Hence, AM can improve the accuracy of predictions, especially in models like Transformers, where it serves as a core component [5]. Therefore, attention-based models and their variants [43, 44] have advanced financial time series forecasting with their large multi-head attention mechanism.

2.2 Theoretical Background

Convolutional Neural Networks

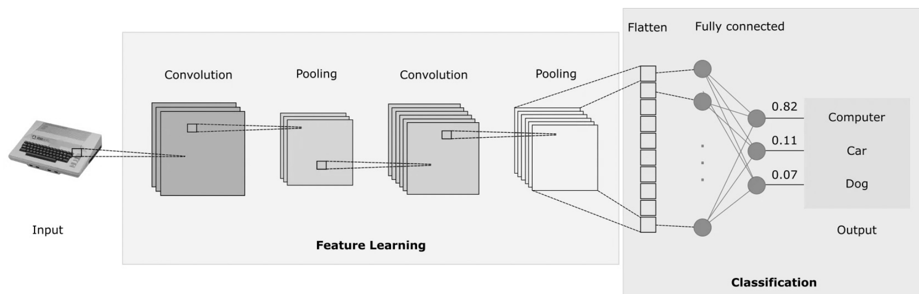

Fig. 1. CNN architecture [45]

Convolutional Neural Networks (CNNs) were introduced as a type of feedforward neural network that excels in tasks such as image processing and natural language processing (NLP) [45]. CNNs have also proven effective in time series prediction [46]. The ability of CNNs to utilize local connectivity and weight sharing reduces the number of parameters, leading to more efficient learning models. A typical CNN architecture consists of three primary components: convolutional layers, pooling layers, and fully connected layers [47]. Each convolutional layer comprises multiple convolutional kernels, with the operation of these layers described by Equation (1). The convolutional layers extract features from the input data, but this often results in high-dimensional feature maps. Therefore, to address this thus decreasing the computational cost, pooling layers are employed after the convolutional layers to reduce the dimensionality of the features [48].

$$l_t = \tanh(x_t * k_t + b_t) \tag{1}$$

In this equation, l_t represents the output after applying the convolution, tanh is the activation function, x_t is the input vector, k_t denotes the convolution kernel weights, and b_t is the bias associated with the convolution kernel.

Long Short-Term Memory

Long Short-Term Memory (LSTM), introduced by Schmidhuber et al. [49], was designed to address the challenges of gradient explosion and vanishing gradients in Recurrent Neural Networks (RNNs) [50]. Unlike the standard RNN, which consists of a single repeating tanh module, LSTM includes four interactive components, making it more effective in capturing long-term dependencies [49]. Accordingly, the core of LSTM is its memory cell, which is regulated by three

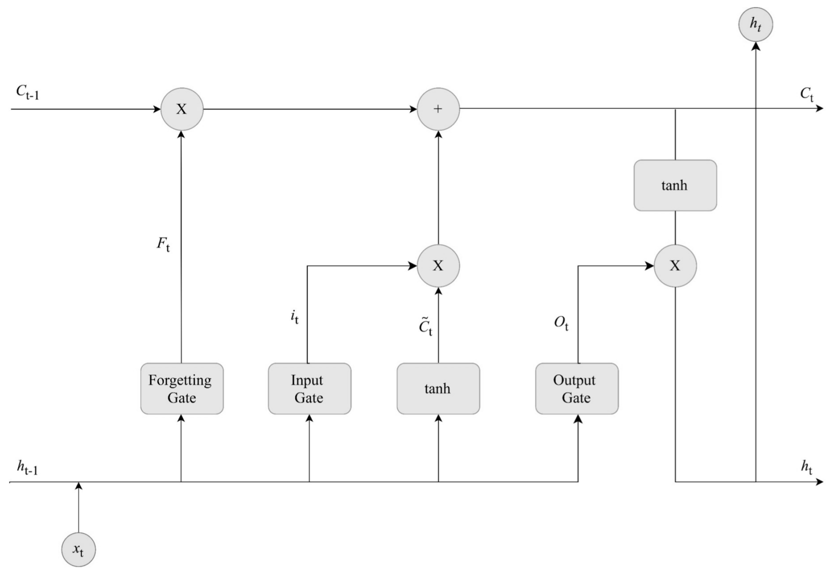

Fig. 2. LSTM architecture

gates: the forget gate, the input gate, and the output gate. These gates control the flow of information and update the cell state, which are outlined as follows:

1. The forget gate determines what portion of the previous cell state C_{t-1} should be retained, calculated as:
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2}$$
where σ is the activation function, W_f and b_f are the weights and bias of the forget gate, respectively.
2. The input gate updates the cell state with new information, determined by:
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3}$$
$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{4}$$
where i_t is the input gate output, and $\tilde{C}_t$ is the candidate cell state.
3. The cell state C_t is updated by combining the forget gate and input gate:
$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{5}$$
4. The output gate decides the next hidden state h_t by:
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{6}$$
5. Finally, the output of the LSTM is computed as:
$$h_t = o_t \cdot \tanh(C_t) \tag{7}$$

Here, x_t represents the current input, h_{t-1} is the previous hidden state, and b terms denote biases for the respective gates.

Attention Mechanism

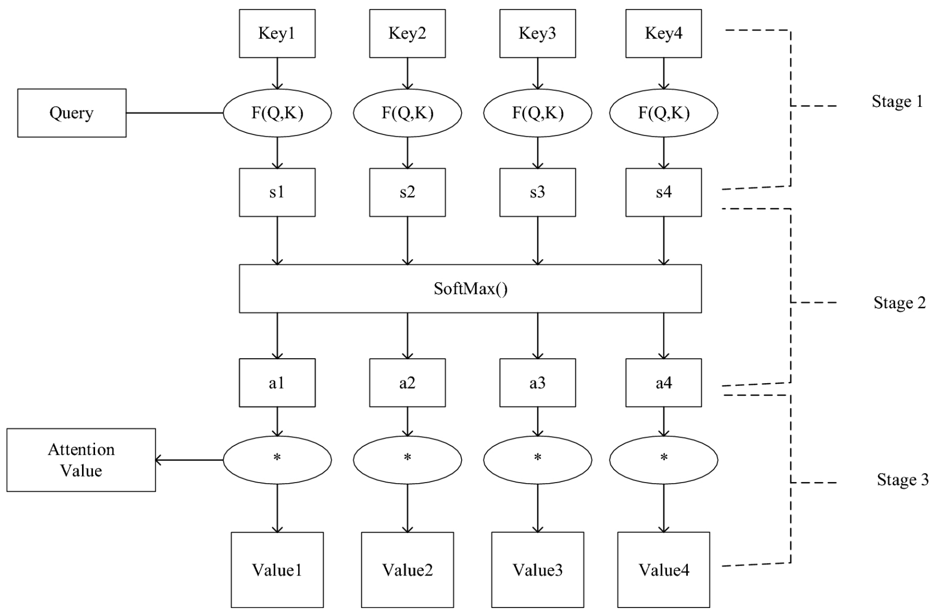

Fig. 3. AM architecture [51]

The Attention Mechanism (AM), introduced by Treisman et al. [52], optimizes models by focusing on the most relevant information within large datasets. By calculating the probability distribution of attention, AM highlights key inputs, effectively enhancing the traditional model based on human visual attention principles. This mechanism prioritizes important information while disregarding less relevant details, thus efficiently allocating attention. The AM calculation process, as illustrated in Fig. 3, can be divided into three main stages:

1. The similarity between the Query (output feature) and Key (input feature) is calculated using:
$$s_t = \tanh(W_h h_t + b_h) \tag{8}$$
where W_h and b_h are the shared weights and biases of AM, and h_t is the input vector.
2. The similarity score is then normalized via the softmax function:
$$a_t = \frac{\exp(s_t^\top v)}{\sum_t \exp(s_t^\top v)} \tag{9}$$
where v represents the attention value.

3. The attention value is calculated by performing a weighted summation:

$$s = \sum_t a_t h_t \tag{10}$$

3 Methodology

3.1 Data Collection and Evaluation Metrics

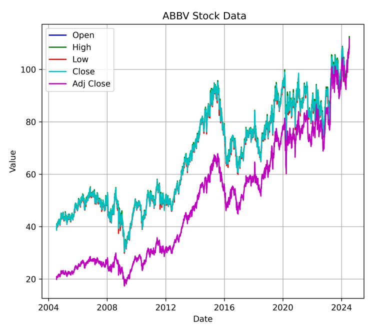

Fig. 4. ABBV raw data

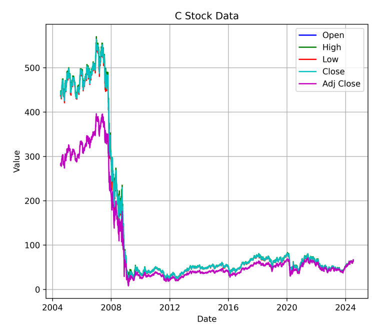

Fig. 5. C raw data

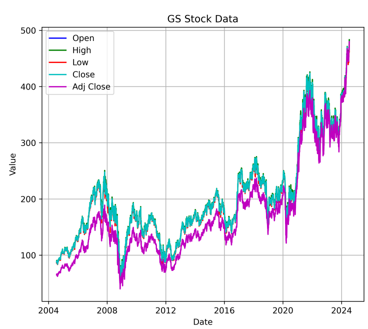

Fig. 6. GS raw data

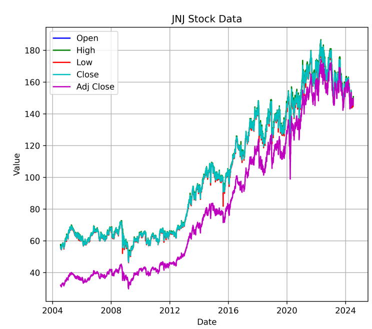

Fig. 7. JNJ raw data

The OHLCV stocks data, sourced from Yahoo Finance's S&P 500 Index, are divided into two distinct groups: Pharmaceuticals (AbbVie Inc., ABBV; Johnson & Johnson, JNJ) and Financials (Goldman Sachs Group Inc., GS; Citigroup Inc., C). This division is made because the inherent nature of pharmaceutical companies differs significantly from that of financial institutions, presenting a challenge for forecasting models. The data timeline is divided into a training set spanning from July 19, 2004, to July 12, 2024 (5,031 observations), with the official forecast starting on July 13, 2024, predicting the following week. The test set covers July 15, 2024, to July 19, 2024 (5 observations). The training set is further split 90:10 for training and validation purposes, which is done to ensure the model's robustness by validating its performance on a small subset. The experiment is compiled at the end of July 20 to guarantee that data remains unseen. The Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) were chosen as evaluation metrics for their effectiveness in identifying predictive errors in univariate analysis. MAE (11) represents the average magnitude of errors in forecasts, offering a direct measure of overall prediction accuracy, which is crucial in the volatile field of stock market values. RMSE (12), known for signifying larger errors, underscores discrepancies that can have serious financial implications. Therefore, the higher the errors, the larger the price deviation:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |Y_i - \hat{Y}_i| \tag{11}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2} \tag{12}$$

3.2 CLAM: CNN-LSTM-AM

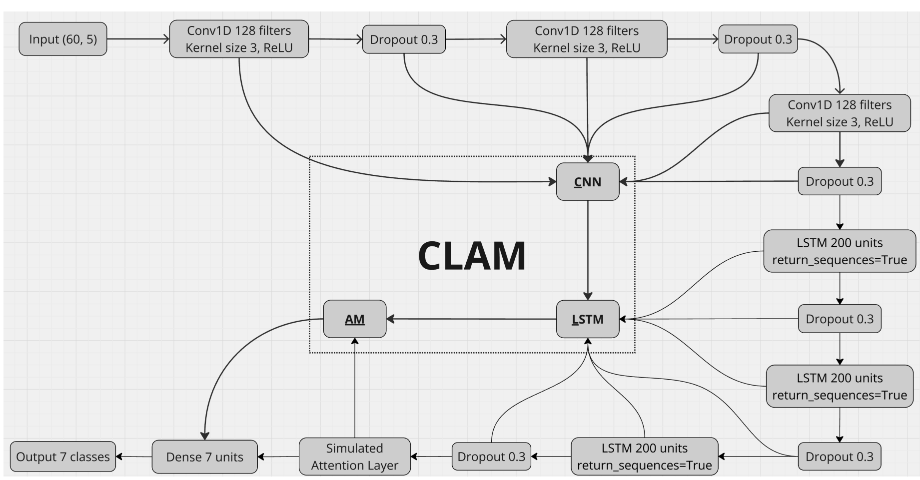

Fig. 8. The proposed CLAM hybrid architecture

Table 1. Model architecture summary

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 60, 5)	0
conv1d (Conv1D)	(None, 60, 128)	2,048
dropout (Dropout)	(None, 60, 128)	0
conv1d_1 (Conv1D)	(None, 60, 128)	49,280
dropout_1 (Dropout)	(None, 60, 128)	0
conv1d_2 (Conv1D)	(None, 60, 128)	49,280
dropout_2 (Dropout)	(None, 60, 128)	0
lstm (LSTM)	(None, 60, 200)	263,200
dropout_3 (Dropout)	(None, 60, 200)	0
lstm_1 (LSTM)	(None, 60, 200)	320,800
dropout_4 (Dropout)	(None, 60, 200)	0
lstm_2 (LSTM)	(None, 60, 200)	320,800
dropout_5 (Dropout)	(None, 60, 200)	0
dense (Dense)	(None, 60, 1)	201
dense_1 (Dense)	(None, 60, 7)	14
Total params	1,005,623	
Trainable params	1,005,623	
Non-trainable params	0	

The input data first passes through a series of convolutional layers, each employing 128 filters with a kernel size of 3, aimed at extracting intricate spatial features from the input sequence. The convolutional layers are activated using the ReLU function, which aids in introducing non-linearity to the model while retaining computational efficiency. Dropout, applied at a rate of 0.3 after each convolutional layer, serves to prevent overfitting by randomly omitting a fraction of the neurons during training, thus enhancing the generalization capability of the model. Following the convolutional layers, the extracted features are fed into a sequence of LSTM layers. Each LSTM layer, consisting of 200 units, is designed to capture the temporal dynamics within the data by maintaining a memory of the previous time steps. The inclusion of multiple LSTM layers allows the model to build a hierarchical understanding of the temporal relationships. To further mitigate overfitting and improve model robustness, Dropout is consistently applied after each LSTM layer. A simulated Attention Mechanism is introduced after the LSTM layers, acting as a dense layer to approximate the focus on critical features across different time steps. This mechanism allows the model to prioritize the most relevant information for the forecasting task, enhancing its ability to capture dependencies that are crucial for accurate predictions.

Finally, the processed features are passed through a dense layer that outputs the forecasted values for the next 7 days. This dense layer represents the final synthesis of the model's learned features, mapping the encoded information to the predicted outcomes. The CLAM model's architecture, depicted in Fig. 8, showcases its capability to combine spatial, temporal, and attention mechanisms effectively. By employing a combination of CNNs, LSTMs, and a simulated AM, CLAM is well-equipped to handle the complex patterns inherent in sequential financial data, providing a robust framework for accurate time series forecasting. Dropout regularization is applied throughout the model, and an early stopping criterion ensures that the model remains both efficient and stable during the training process. These features contribute to the model's ability to generalize well to unseen data, thereby enhancing its utility in real-world applications.

4 Experiments and Results

4.1 Experimental Process

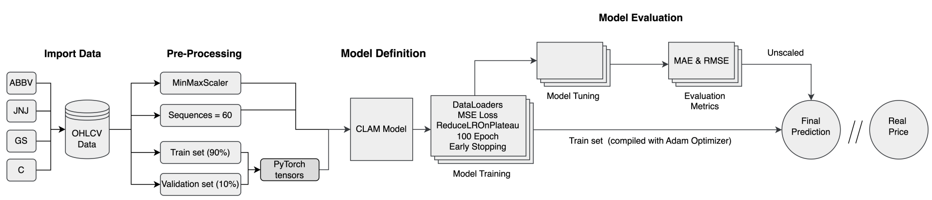

Fig. 9. Full experimental process

Our experimental procedure began with the initialization of random number generators in both NumPy and TensorFlow libraries, ensuring consistency and reproducibility by setting a seed value of 42. The stock datasets, loaded from `csv` files, contained OHLCV data, which were first normalized to mitigate the effects of varying magnitudes within the features. Two separate `MinMaxScaler` instances were employed: one for the feature columns ("Open", "High", "Low", "Adj Close", "Volume") and another for the target column ("Close"). This normalization ensured that the data was scaled uniformly, addressing potential biases and enhancing the model's performance by keeping values within the [-1, 1] range. Following normalization, sequences were constructed from the dataset, each with a length of 60 days to predict the subsequent 7-day closing prices. This sequence creation process, which produced numerous input-output pairs, was conducted exclusively on the normalized dataset, thereby preventing any leakage of future information. The sequences were subsequently split into training and validation sets, maintaining a 90:10 ratio to preserve the integrity of the experimental setup. Hence, this separation was crucial to evaluate the model on unseen data and prevent overfitting, preventing outsample forecast disruption.

The model architecture, a hybrid of convolutional and LSTM layers, was then defined to capture both spatial and temporal patterns inherent in the financial time series data. A batch size of 64 was selected for training, allowing for efficient processing of the data. Notably, an Attention layer was incorporated to dynamically weigh the input features, enabling the model to focus on more critical aspects of the data. The model was compiled using the Mean Squared Error (MSE) loss function and the Adam optimizer, supplemented by a learning rate scheduler (`ReduceLROnPlateau`), which reduced the learning rate by a factor of 0.2 if the validation loss did not improve for 5 consecutive epochs, with a minimum learning rate set at 0.001. An `EarlyStopping` callback was integrated, halting the training if the validation loss did not improve after 10 epochs, thus preventing overfitting and ensuring that the model generalized well. The architecture summary of the model was generated to provide an overview of the layers and parameters involved. Throughout training, the model iterated over the dataset in batches, updating the weights via backpropagation after each batch. The early stopping criterion was monitored using validation loss, and the model's state with the lowest validation loss was saved as the best model. Upon completing the training, the model was evaluated on the validation set, where it achieved an MAE and RMSE that provided insights into its predictive accuracy. Finally, the trained model was utilized to make out-sample forecasts, and the predictions were unscaled to facilitate a direct comparison with the test data.

4.2 Performance Comparison

Table 2. Model performance on training and validation sets

Model	ABBV		C		GS		JNJ	
	Train	Val	Train	Val	Train	Val	Train	Val
MAE								
CNN [53]	0.112	0.115	0.138	0.135	0.129	0.131	0.102	0.108
LSTM [54]	0.124	0.128	0.148	0.151	0.153	0.156	0.125	0.131
CNN-LSTM [55]	0.096	0.102	0.105	0.110	0.098	0.106	0.093	0.100
LSTM-AM [56]	0.080	0.085	0.092	0.096	0.089	0.093	0.081	0.088
CLAM	0.018	0.007	0.011	0.018	0.016	0.022	0.031	0.017
RMSE								
CNN [53]	0.137	0.142	0.185	0.184	0.174	0.177	0.129	0.137
LSTM [54]	0.158	0.161	0.190	0.195	0.199	0.202	0.160	0.167
CNN-LSTM [55]	0.121	0.128	0.139	0.143	0.131	0.140	0.120	0.128
LSTM-AM [56]	0.106	0.113	0.125	0.129	0.118	0.125	0.110	0.116
CLAM	0.025	0.012	0.019	0.025	0.022	0.033	0.025	0.022

The performance evaluation in Table 2 clearly identifies the CLAM model as the superior performer, with the LSTM-AM model securing the second position. CLAM's validation MAE of 0.007 on the ABBV dataset is a remarkable 91.8% improvement over the 0.085 recorded by LSTM-AM, which itself is 16.7% better than CNN-LSTM's 0.102. This substantial reduction in error demonstrates

CLAM's advanced capability in capturing complex temporal dependencies more effectively than the other models. In contrast, while LSTM-AM's attention mechanism allows it to surpass CNN-LSTM and other models, reducing the MAE by 34.4% compared to CNN's 0.115, it still lags significantly behind CLAM, underscoring the latter's superior architecture and precision. Further emphasizing CLAM's dominance, the validation RMSE figures show a similar pattern. CLAM achieves a validation RMSE of just 0.012 on ABBV, representing an 89.4% reduction compared to LSTM-AM's 0.113, and an even more striking 91.5% decrease compared to CNN's 0.142. LSTM-AM, while effective, reduces the RMSE by 11.7% compared to CNN-LSTM's 0.128, but this improvement pales in comparison to the nearly perfect generalization displayed by CLAM. The consistency of CLAM's performance across both MAE and RMSE, with such substantial percentage reductions, clearly positions it as the best model, with LSTM-AM as a commendable yet significantly less effective second choice.

CLAM's performance can be attributed to its advanced architectural design, which likely incorporates sophisticated techniques such as enhanced attention mechanisms, deeper layers, and more effective regularization methods. These elements allow CLAM to capture intricate temporal patterns and dependencies in the data more effectively than its predecessors. The model's architecture is designed to minimize overfitting by balancing complexity with generalization, which explains its low error rates on validation sets. While LSTM-AM also benefits from attention mechanisms, its relatively simpler structure compared to CLAM limits its ability to achieve the same level of accuracy, leading to higher error rates. The stepwise improvements seen from CNN to LSTM-AM and finally to CLAM highlight the increasing sophistication and effectiveness of the models, with CLAM representing the current pinnacle in this evolutionary process.

4.3 CLAM Forecasting

Hyperparameter Tuning

Table 3. Hyperparameter tuning summary

Hyperparameter	Values Tested	Best Value
LSTM Units	100, 150, 200	200
Conv1D Filters	64, 128, 256	128
Kernel Size	3, 5, 7	3
Dropout Rate	0.2, 0.3, 0.4	0.3
Batch Size	32, 64, 128	64
Learning Rate	0.01, 0.005, 0.001	0.001
Optimizer	Adam, RMSprop, SGD	Adam

Hyperparameter tuning was not merely a process of optimization but also a critical exercise in contrasting the model's performance under various configurations. As illustrated in Table 3, a wide spectrum of values was tested across essential hyperparameters, including LSTM units, Conv1D filters, kernel size, dropout rate, batch size, learning rate, and optimizer selection. This exhaustive tuning process was essential to strike the delicate balance between model complexity and generalization, with a specific focus on avoiding overfitting, a common pitfall in deep learning models. The selection of 200 LSTM units, for instance, was the result of a careful trade-off. While a smaller number of units (such as 100 or 150) led to faster convergence, these configurations consistently underperformed in capturing the complexity of the sequential data, as evidenced by higher validation errors. Conversely, larger configurations, such as 256 Conv1D filters, introduced unnecessary computational overhead without a commensurate improvement in performance, highlighting the diminishing returns of increasing model depth and width. In addition, the kernel size of 3 was particularly effective in capturing localized temporal patterns, contrasting with larger kernels that diluted the model's ability to focus on fine-grained features.

Moreover, the dropout rate of 0.3 was determined to be optimal after observing that lower rates (e.g., 0.2) led to overfitting, while higher rates (e.g., 0.4) excessively hindered learning, reducing the model's capacity to generalize. The choice of a batch size of 64 was similarly contrasted against smaller and larger batch sizes, where smaller batches resulted in noisier gradient estimates and larger batches slowed down the training without significant gains in accuracy. The learning rate of 0.001, coupled with the Adam optimizer, was found to offer the best balance between learning speed and stability, particularly when compared to other optimizers like RMSprop and SGD, which either converged slower or required more meticulous tuning. Overall, these critical decisions in hyperparameter selection provides a clear contrast to alternative configurations that either overfitted or underperformed, promoting efficient model training scheme.

Training and Validation

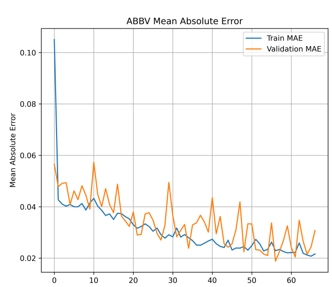

Fig. 10. MAE of ABBV

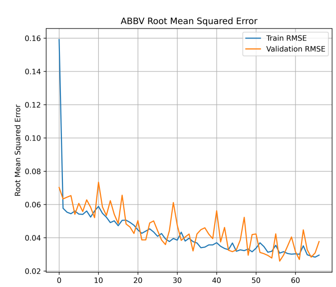

Fig. 11. RMSE of ABBV

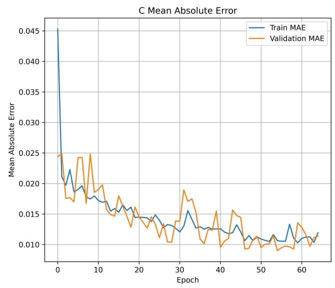

Fig. 12. MAE of C

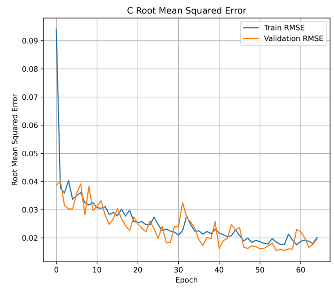

Fig. 13. RMSE of C

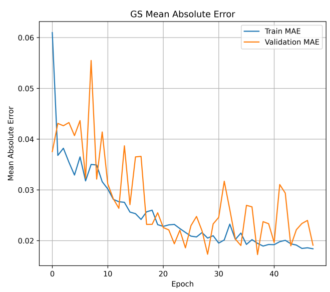

Fig. 14. MAE of GS

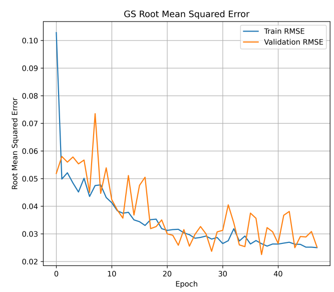

Fig. 15. RMSE of GS

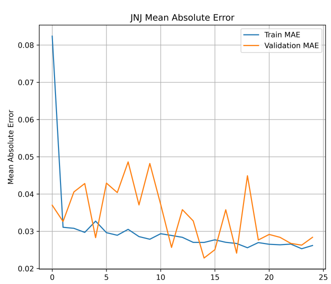

Fig. 16. MAE of JNJ

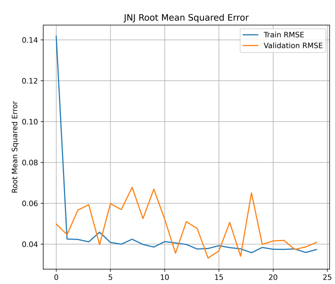

Fig. 17. RMSE of JNJ

The training process for ABBV shows effective learning, with MAE dropping from 0.10 to 0.025 by epoch 20 and validation MAE converging around 0.025 by epoch 65, despite some fluctuations. RMSE follows a similar trend, decreasing from 0.16 to below 0.04, with both metrics stabilizing in the final epochs, indicating good generalization. Regarding stock C, the training MAE declines rapidly from 0.045 to 0.020 by epoch 10, with validation MAE showing fluctuations but stabilizing around 0.015 by epoch 30. In general, both MAE and RMSE converge close to 0.010 and 0.02 respectively by epoch 60, reflecting strong learning and generalization, though some validation variability remains during the evaluation.

For GS, training MAE decreases from 0.06 to 0.03 by epoch 10, with validation MAE peaking around 0.05 but converging by epoch 20. Both metrics stabilize around 0.02 by epoch 30, and RMSE follows suit, dropping from 0.10 to 0.05 by epoch 10 and stabilizing below 0.03. This suggests effective pattern capture, though validation fluctuations hint at possible fine-tuning needs. Finally, for JNJ, training MAE drops quickly from 0.08 to 0.03 by epoch 5, while validation MAE fluctuates, peaking near 0.05 before stabilizing around 0.03 by epoch 10. RMSE shows a rapid decrease from 0.14 to below 0.06 by epoch 5, with stabilization around 0.04 from epoch 10 onward, indicating effective learning but with potential for further model tuning to address validation variability.

Trend Forecasting

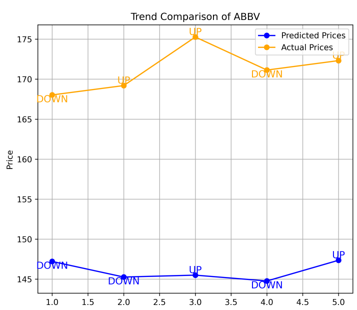

Fig. 18. ABBV forecasting results

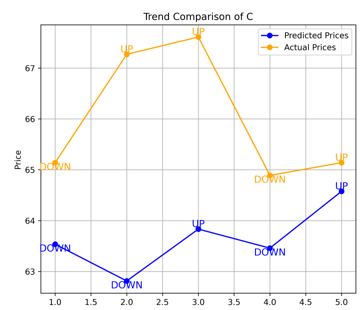

Fig. 19. C forecasting results

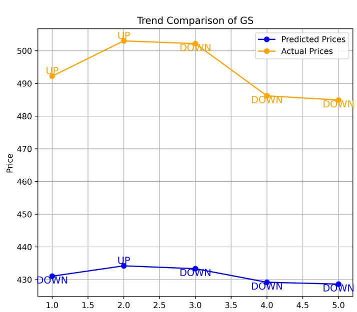

Fig. 20. GS forecasting results

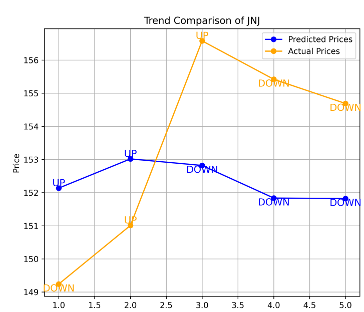

Fig. 21. JNJ forecasting results

Instead of focusing on precise price forecasting, which could be disrupted by market volatility and external factors, CLAM prioritizes predicting the direction and strength of trends. By leveraging historical price patterns, it aims to capture shifts in market momentum, labeling trends as “UP” if the next day’s price is expected to rise and “DOWN” if it is expected to fall. This approach reduces the influence of short-term noise, providing more insights for trading.

For ABBV, CLAM successfully identified the immediate downtrend on Day 1 (Monday), a day typically marked by high volatility and investor reactions post-weekend. Although it initially missed the recovery on Day 2, it quickly adjusted and accurately predicted the upward trend on Day 3. CLAM then continued to forecast the correct trend for the final two days, resulting in three consecutive accurate predictions and capturing the overall uptrend of ABBV, achieving an accuracy of 80%. In the case of C, CLAM once again identified the initial downtrend on Monday but failed to predict the following day’s trend. However, it redeemed itself by accurately forecasting the trend for the last three days consecutively, with minimal price deviation, leading to an overall accuracy of 80%. GS demonstrated the model’s most promising trend capture performance. Despite missing the initial rise on Day 1, CLAM consistently predicted the correct trend for the remaining four consecutive days, successfully capturing the overall downtrend for the week and once again achieving 80% accuracy. Finally, for JNJ, while CLAM effectively captured the overall downward trend, it struggled with consistency, failing to predict the trends on Days 1 and 3. Nevertheless, its ability to accurately capture the sharp decline in the final two days provided significant forecasting value, resulting in a 60% accuracy for JNJ.

Overall, CLAM achieved an average trend forecasting accuracy of 75% across pharmaceutical and finance stocks. The model demonstrated strong performance in most cases, particularly in forecasting consecutive trends, identifying immediate downtrends at the week’s start, and detecting potential flash crashes.

5 Discussion

5.1 Summary of Findings and Contributions

Our research demonstrates that the integration of stacked deep learning layers in the CLAM model—comprising Convolutional Neural Networks (CNN), Long Short-Term Memory networks (LSTM), and Attention Mechanisms (AM). CLAM significantly outperforms traditional standalone models. This superiority is attributed to CLAM’s advanced feature extraction and attention mechanisms, which enhance the LSTM architecture’s inherent ability to capture long-term dependencies in time series forecasting. As a result, CLAM achieves an average reduction of over 80% in MAE and RMSE compared to conventional models such as CNN, LSTM, CNN-LSTM, and LSTM-AM. Additionally, our experiments indicate that in 3 out of 4 cases, CLAM successfully forecasts the correct trends for 3-4 consecutive days, effectively capturing market downturns on the opening day, which tends to be unpredictable due to abnormal traders’ behavior.

Finally, CLAM achieves a 75% accuracy rate in stock trend forecasting. This highlights the potential of hybrid models in time series forecasting, demonstrating that attention-based deep learning hybrids like CLAM are particularly well-suited as live-trading indicators for investors focused on weekly trends. Hence, CLAM enhances decision-making processes in financial trading, providing a robust and lightweight tool for forecasting short-term financial market movements.

5.2 Limitations and Recommendations

Our study is grounded in raw OHLCV data of stock prices, which offers ease of access but also introduces potential market biases. This limitation arises from the model's inability to account for external economic or financial factors, such as extreme events, technical indicators, or government policies, all of which are crucial drivers of stock prices. Additionally, since our research primarily focuses on trend detection, the CLAM architecture is not fully optimized for value trading, where precise price gaps are critical. As a result, CLAM's potential may extend further and be more robust than demonstrated in its current form. We encourage researchers to expand upon our work by developing deeper hybrid models or employing explainable models to study stock trends effectively. In addition, practitioners are advised to enhance data collection processes by incorporating data from stock indexes and integrating relevant technical trading indicators, thereby uncovering more intricate stock patterns within the raw target features.

6 Conclusion

This study introduced the CLAM model, a novel hybrid combination of Convolutional Neural Networks, Long Short-Term Memory networks, and Attention Mechanisms designed for multi-step stock price trend forecasting. The model demonstrated significant improvements in predictive accuracy, with an average reduction of over 80% in MAE and RMSE compared to traditional models. CLAM's ability to effectively capture complex temporal dependencies and its robust performance across various stock datasets highlight its potential as a valuable tool in financial forecasting. The findings of this research suggest that CLAM is particularly well-suited for short-term trend forecasting, providing accurate and timely predictions that can assist investors and traders in making informed decisions. The model's design, which integrates feature extraction and attention mechanisms, offers a significant advancement over existing approaches. Moreover, there are several promising directions for further research. Future studies could explore enhancing CLAM's adaptability to different market conditions, such as varying levels of market volatility or the impact of external economic factors. Additionally, integrating CLAM with other financial analysis techniques, such as sentiment analysis or reinforcement learning, could further improve its predictive capabilities. Thus, real-time deployment of CLAM in live trading environments of daily stocks or hourly cryptocurrency prices is another area that warrants investigation, allowing for real-time performance assessment.

References

1. Michael E Porter. *Technology and Competitive Advantage*. Harvard Business Review, 1985.
2. Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
3. Tomas Mikolov, Martin Karafiát, Lukáš Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. *Interspeech*, 2010.
4. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
5. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
6. Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg S Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. *Nature medicine*, 25(1):24–29, 2019.
7. Yoo-Geun Ham, Jeong-Hwan Kim, and Jing-Jia Luo. Deep learning for multi-year enso forecasts. *Nature*, 573(7775):568–572, 2019.
8. Ngoc Nguyen, Kenji Shirai, and Julien Velcin. Neural networks for financial time series forecasting: a review. *Neural Computing and Applications*, 25(2):305–315, 2015.
9. Shihao Gu, Bryan Kelly, and Dacheng Xiu. Empirical asset pricing via machine learning. *arXiv preprint arXiv:1809.01078*, 2018.
10. Bruno M Henrique, Vinicius A Sobreiro, and Hiroshi Kimura. Literature review: Machine learning techniques applied to financial market prediction. *Expert Systems with Applications*, 124:226–251, 2019.
11. Arshian Sharif, Chaker Aloui, and Larisa Yarovaya. Covid-19 pandemic: Impact of restriction policies on the economy. *Environment, Development and Sustainability*, 23(6):9389–9412, 2021.
12. Michael Hagenau, Mario Liebmann, and Dirk Neumann. Automated news reading: Stock price prediction based on financial news using context-capturing features. *Decision Support Systems*, 55(3):685–697, 2013.
13. George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
14. Jan G De Gooijer and Rob J Hyndman. Forecasting using a hybrid sarima and support vector machines model: An application to uk electricity demand. *International Journal of Forecasting*, 27(3):735–758, 2011.
15. Douglas C Montgomery, Elizabeth A Peck, and G Geoffrey Vining. *Introduction to Linear Regression Analysis*. John Wiley & Sons, 2012.
16. Ruey S Tsay. *Analysis of financial time series*, volume 543. John Wiley & Sons, 2005.
17. Rama Cont. Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative finance*, 1(2):223–236, 2001.
18. Eugene F Fama. Stock returns, real activity, inflation, and money. *The American economic review*, 71(4):545–565, 1981.
19. Malcolm Baker and Jeffrey Wurgler. Investor sentiment and the cross-section of stock returns. *The Journal of Finance*, 61(4):1645–1680, 2006.

20. Sushil Bikhchandani, David Hirshleifer, and Ivo Welch. A theory of fads, fashion, custom, and cultural change as informational cascades. *Journal of Political Economy*, 100(5):992–1026, 1992.
21. Vladimir Vapnik. *Support-vector networks*. Springer, 1995.
22. Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.
23. Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.
24. Edwin KS Chong, Chong Han, and Frank CM Park. Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, 83:187–205, 2017.
25. Hanzhi Zhu, Zhenghui Hua, and Zaiyong Huang. Deep learning for predicting stock market returns using newspaper headlines. *Journal of Finance and Data Science*, 4(1):39–57, 2018.
26. Yuqing Zhang, Shaohua Wang, Yihua Zhang, and Jia Xu. A hybrid model based on ceemdan and gwo for stock price forecasting. *Applied Soft Computing*, 57:211–225, 2017.
27. Xue Wang, Edward Lin, Jin Zhang, and Shashi Manos. Combining statistical and machine learning methods to predict financial time series. *Journal of Computational Finance*, 23(2):1–20, 2019.
28. Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009.
29. Zachary C Lipton. The mythos of model interpretability. *Communications of the ACM*, 61(10):36–43, 2018.
30. Nguyen Quoc Anh, Son Ha, and Hieu Thai. Phase space reconstructed neural ordinary differential equations model for stock price forecasting. In *Pacific Asia Conference on Information Systems*, 2024.
31. Nguyen Quoc Anh and Son Ha. Transforming stock price forecasting: Deep learning architectures and strategic feature engineering. In *Modeling Decisions for Artificial Intelligence*, 2024.
32. Nguyen Quoc Anh and Son Ha. A lightweight multi-head attention transformer for stock price forecasting. *SSRN Electronic Journal*, 2024.
33. Zongwu Chen, Man-Hon Leung, and Hazem Daouk. Forecasting the chinese stock market with a hybrid machine learning model. *Journal of Empirical Finance*, 29:1–13, 2015.
34. Manik Mahajan and Mohit Sinha. A hybrid sarima-xgboost model and its application to forecasting indian stock index. *Journal of Statistics and Management Systems*, 23(1):31–48, 2020.
35. Dheeraj Bhuriya, Gunjan Kaushik, Ashish Sharma, Utkarsh Singh, et al. Stock market prediction using a hybrid approach. *2017 International Conference on Computing, Communication and Automation (ICCCA)*, pages 305–310, 2017.
36. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
37. Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guanjie Jiang, and Garrison W Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. *In Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2627–2633, 2017.
38. Yang Zhang and Li Xu. Hybrid models combining convolutional neural networks and bidirectional lstm with eca for stock trend prediction. *Applied Soft Computing*, 89:106113, 2020.

39. Xiang Li, Pengfei Wu, Wei Wang, and Guang Shen. Deep learning-based stock index prediction: A comparative study. *Applied Intelligence*, 50(6):3403–3416, 2020.
40. Xingjie Pang, Haiyan Zhou, and Pingping Wang. Deep learning models for stock price prediction: A review. *Journal of Forecasting*, 39(4):623–659, 2020.
41. Rosario N Mantegna and H Eugene Stanley. *Introduction to econophysics: correlations and complexity in finance*. Cambridge University Press, 1999.
42. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
43. Bryan Lim, Boris N Oreshkin, Ondrej Bohdal, Fabio Ramos, and Slawek Smyl. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021.
44. Haoyi Zhou, Shanghang Zhang, Jianxin Peng, Shuai Zhang, Jianxin Li, Huitong Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):11106–11115, 2021.
45. Nafiz Shahriar. What is convolutional neural network (cnn)? deep learning, 2021. Accessed: 2024-08-18.
46. Aji Prasetya Wibawa, Agung Bella Putra Utama, Hakkun Elmunsyah, Utomo Pujianto, Felix Andika Dwiyanto, and Leonel Hernandez. Time-series analysis with smoothed convolutional neural network. *Journal of Big Data*, 9, 2022.
47. Andrew Mann and Surya R. Kalidindi. Development of a robust cnn model for capturing microstructure-property linkages and building property closures supporting material design. In *Frontiers in Materials*, 2022.
48. Xia Zhao, Limin Wang, Yufei Zhang, Xuming Han, Muhammet Deveci, and Milan Parmar. A review of convolutional neural networks in computer vision. *Artif. Intell. Rev.*, 57:99, 2024.
49. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.
50. Nicolas Zucchet and Antonio Orvieto. Recurrent neural networks: vanishing and exploding gradients are not the end of the story. *ArXiv*, abs/2405.21064, 2024.
51. Wenjie Lu, Jiazheng Li, Jingyang Wang, and Lele Qin. A cnn-bilstm-am method for stock price prediction. *Neural Computing and Applications*, 33:4741 – 4753, 2020.
52. Anne Treisman and Garry A. Gelade. A feature-integration theory of attention. *Cognitive Psychology*, 12:97–136, 1980.
53. Zýnnet Duygu Akehýr and Erdal Kılıç. How to handle data imbalance and feature selection problems in cnn-based stock price forecasting. *IEEE Access*, 10:31297–31305, 2022.
54. Khaled Abdul-Aziz Al-Utaibi, Shehneel Siddiq, and Sadiq M. Sait. Stock price forecasting with lstm: A brief analysis of mathematics behind lstm. *Biophysical Reviews and Letters*, 2023.
55. Liu Jialin et al. Cnn-lstm model stock forecasting based on an integrated attention mechanism. *2022 3rd International Conference on Pattern Recognition and Machine Learning (PRML)*, pages 403–408, 2022.
56. Yufan Chen, Fujian Wu, Guanzhi Li, and Choujun Zhan. Feature-ranked attention-driven lstm: A novel hybrid model for short-term stock price predictions. *2023 IEEE International Symposium on Product Compliance Engineering - Asia (ISPCE-ASIA)*, pages 1–6, 2023.