Deep Learning-Based Stock Price Forecasting for the Saudi Telecommunication Sector: A Comparative Evaluation with Baseline Models

Hadi S. AlQahtani [1,*], Mohammed J. Alhaddad [1], and Mutasem Jarrah [2]

[1] Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia
[2] Department of Data Science and Artificial Intelligence, Faculty of Information Technology, Applied Science University, Amman, Jordan
Email: halqahtani0193@stu.kau.edu.sa (H.S.A.); malhaddad@kau.edu.sa (M.J.A.); m_jarrah@asu.edu.jo (M.J.)
*Corresponding author

***Abstract*—Accurately predicting stock prices is a long-standing challenge due to the volatility and nonlinear structure of financial markets. To address this, the present study conducts a comprehensive evaluation of Deep Learning (DL) architectures for univariate stock price forecasting within the Saudi telecommunications sector. Four DL models—Long Short-Term Memory (LSTM), Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), and a hybrid CNN-LSTM—are benchmarked against the Naïve, SMA-5, and Autoregressive Integrated Moving Average (ARIMA) models using five years of daily closing prices for Saudi Telecom Company (STC), Mobily, and Zain. All models performances were assessed using Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) on inverse-scaled predictions. The findings clearly show that DL models capture complex temporal dependencies more effectively than traditional baselines. LSTM consistently achieved the lowest test RMSE across all datasets (Mobily: 1.169705; STC: 0.708495; Zain: 0.271470), confirming its superior predictive capability. CNN and CNN-LSTM delivered competitive accuracy, while RNN exhibited the weakest performance. Overall, the results demonstrate the strong potential of deep learning—particularly LSTM—for improving short-term stock price prediction and advancing data-driven financial analytics in the Saudi stock market.**

***Keywords*—Deep Learning (DL), stock market, prediction, models, regression, time series**

I. Introduction

The stock market represents one of the most dynamic, intricate, and unpredictable domains within the global economy, yet its accurate forecasting remains of paramount importance to financial institutions, hedge funds, traders, and market speculators seeking to gain a competitive advantage [1]. Stock prices are driven by a complex interplay of factors, including macroeconomic indicators, geopolitical events, corporate earnings reports, and shifts in investor sentiment, all of which contribute to high levels of uncertainty and volatility [2]. As a result, achieving precise stock price prediction is inherently challenging. Nevertheless, even minor improvements in predictive accuracy can yield substantial benefits for portfolio optimization, risk management, and strategic investment decision-making [3].

Historically, stock market forecasting has evolved through several methodological paradigms. Early approaches were grounded in statistical and econometric techniques, most notably the Autoregressive Integrated Moving Average (ARIMA) model [2]. Such linear models are computationally efficient and perform reasonably well under stable and stationary conditions; however, they are limited in their ability to capture nonlinear relationships, abrupt structural changes, and long-term temporal dependencies that typify real-world financial time series [4]. To complement these methods, simpler baselines, such as the Naïve persistence model, which assumes the next price equals the last observed value, and the Simple Moving Average (SMA) model, which smooths short-term fluctuations, are often employed as reference benchmarks. While these baseline methods provide useful interpretability and serve as lower bounds for performance comparison, their inability to adapt to nonlinear and dynamic market behaviors restricts their predictive power in volatile environments.

The emergence of Machine Learning (ML) and, more recently, Deep Learning (DL) has revolutionized the field of financial forecasting. Deep learning models possess the unique ability to automatically learn intricate, nonlinear patterns directly from historical data without relying on manually defined mathematical formulations [5]. This capacity allows them to uncover hidden dependencies and temporal correlations that traditional linear techniques tend to overlook. Moreover, deep learning's adaptability to large-scale, high-dimensional, and noisy datasets makes it

Manuscript received July 28, 2025; revised August 18, 2025; accepted December 9, 2025; published April 16, 2026.

particularly effective in capturing the fast-paced and evolving nature of stock markets [6].

The central objective of this study is to identify which deep learning architecture—Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN), or a hybrid CNN-LSTM model—offers the highest predictive accuracy and robustness in univariate stock market forecasting. The research focuses on the Saudi Telecommunication sector, a vital yet relatively underexplored segment of the Saudi stock market, and evaluates each model's performance in predicting daily closing prices. The study hypothesizes that the LSTM model will outperform the other architectures due to its advanced memory mechanisms, which enable it to capture long-term dependencies and retain information over extended time horizons [7].

While the application of deep learning in financial modeling has been widely studied, there remains a notable gap in domain-specific research that targets emerging markets and industry-focused forecasting. Existing literature primarily concentrates on large international indices such as the S&P 500 or NASDAQ, often overlooking regional sectors with distinct market dynamics. This study contributes to bridging that gap by providing an empirical analysis of deep learning model performance within the Saudi Telecommunication industry. Through this investigation, the research not only enriches the existing body of financial forecasting literature but also offers practical insights for investors and policymakers into how advanced DL techniques can enhance predictive modeling, optimize trading strategies, and support informed financial decision-making in evolving market environments.

This research uses various approaches in the field of stock market prediction, and it provides additional evidence. How the research is described illustrates how it was conducted, making it simple for other researchers to conduct the research again and comprehend it. It also acts as a useful guideline for additional research in this area. The research indicates that, despite being complex, deep learning models are more capable of examining stock market trends and providing a more accurate means of predicting them [8].

The remainder of this paper is structured as follows: Section II provides a comprehensive review of the key literature on deep learning models applied to time series prediction. Section III outlines the methodology, including data collection and preprocessing procedures, as well as the construction and training of the proposed models. Section IV presents the experimental results of the model evaluations for each company. Finally, Section V concludes the paper by summarizing the main findings and offering directions for future research.

II. Literature Review

Deep learning models have emerged as highly effective tools for time series prediction due to their ability to automatically learn complex temporal dependencies and nonlinear relationships within data [9]. Unlike traditional machine learning approaches, they do not require extensive feature engineering, as they can inherently extract meaningful representations directly from raw input sequences. This capability makes deep learning particularly advantageous in financial forecasting, where stock market data often display chaotic, volatile, and nonlinear behaviors that are difficult to capture using conventional statistical methods [9].

A. Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are particularly well-suited for modeling sequential data, where the order and temporal relationship between observations are critical [10]. Unlike traditional feedforward neural networks, RNNs possess recurrent connections—loops that allow information from previous time steps to influence the network's current state. At each point in time, the output generated by a neuron is fed back into the model as input for the next step, enabling the network to maintain a form of memory known as the hidden state. This internal state preserves contextual information, allowing RNNs to learn temporal dependencies and capture patterns within sequential datasets. Consequently, they are widely applied to time series forecasting tasks, where recalling past information is essential for predicting future outcomes [11].

However, despite their advantages, RNNs face several limitations. One of the most significant challenges is the vanishing gradient problem—a phenomenon that occurs during backpropagation when gradients diminish exponentially over long sequences, impeding the network's ability to learn long-term dependencies effectively [12]. As a result, RNNs often struggle to retain crucial information from earlier time steps, leading to performance degradation in complex temporal modeling. Additionally, when applied to noisy and volatile datasets such as stock market prices, RNNs can be prone to overfitting, thereby reducing their generalization capability and predictive robustness.

The recurrent architecture of RNNs enables them to model sequential correlations that conventional feedforward networks are incapable of capturing [13]. This structural feature allows information from previous time steps to influence future predictions, making RNNs particularly suitable for sequence-based data such as financial time series. However, standard or vanilla RNNs struggle to process long sequences effectively due to the exploding and vanishing gradient problems, which arise during backpropagation and hinder the network's ability to learn long-term dependencies [14].

To overcome these challenges, RNNs with memory mechanisms, such as LSTM and Gated Recurrent Unit (GRU) networks, were developed. These architectures incorporate gated memory cells that regulate the flow of information, allowing the model to retain relevant knowledge over extended periods while mitigating gradient instability [15]. Through their recurrent connections, such networks can effectively capture temporal dependencies within sequential observations [16]. As highlighted in Ref. [17], deep learning models based on memory-enhanced RNNs have demonstrated significant potential in stock market forecasting, as they can identify

intricate temporal relationships and patterns that are often undetectable by traditional machine learning or statistical models.

B. Long Short-Term Memory (LSTM) Networks

The Long Short-Term Memory (LSTM) network was developed as an advancement over the conventional RNN to effectively address the vanishing gradient problem that limits RNNs in learning long-term dependencies. As highlighted by Fozap [18], LSTM architectures excel at capturing long-range temporal patterns within sequential data by introducing a distinctive internal structure composed of memory cells and three regulating mechanisms known as gates: the input gate, forget gate, and output gate. These gates dynamically control the flow of information into, within, and out of the cell state, enabling the network to selectively retain relevant information and discard irrelevant or outdated data over time. This gating mechanism grants LSTMs the ability to maintain contextual information across extended time horizons—something standard RNNs struggle to achieve [18].

Extensive empirical studies have demonstrated that LSTM models consistently outperform traditional RNNs and other simpler architectures in forecasting financial time series, particularly stock prices. Researchers have widely acknowledged that LSTMs deliver higher predictive accuracy and produce fewer errors, especially in short-term forecasting scenarios [19]. In addition, Peivandizadeh *et al.* [10] confirmed that LSTM networks perform robustly on financial datasets characterized by nonlinearity and nonstationarity, refining the algorithm further to enhance its predictive precision. Similarly, Gajamannage *et al.* [12] proposed an advanced LSTM framework integrated with sentiment analysis, which significantly improved prediction accuracy in volatile market conditions, demonstrating the model's adaptability to noisy financial environments. Moreover, findings in Ref. [20] reaffirm that LSTMs consistently outperform RNN and CNN models in long-term stock price forecasting, establishing their superiority and suitability for modeling complex temporal dependencies in financial time series data.

C. Convolutional Neural Networks (CNNs)

Lu *et al.* [3] highlighted that CNNs can be highly effective in capturing short-term variations in stock prices, although they tend to struggle with long-term dependencies compared to LSTM models. A typical time series-based CNN architecture consists of convolutional layers that extract local temporal features from sequential data, followed by pooling layers—commonly max pooling—which reduce the dimensionality of feature maps while retaining the most significant information. This hierarchical reduction process enhances computational efficiency and mitigates overfitting by summarizing dominant trends within smaller feature sets. Subsequently, the extracted features are flattened into a one-dimensional vector and passed through one or more fully connected (dense) layers, which are responsible for performing the final prediction tasks, such as forecasting future stock prices [5].

A Further study [7] has demonstrated that a CNN-based model can achieve competitive performance in financial forecasting, particularly when integrated with other deep learning architectures such as an RNN or an LSTM. Although a CNN is traditionally recognized for its strength in processing image and spatial data, it has been successfully adapted for time-series analysis by applying one-dimensional convolutions along the temporal axis. This adaptation enables the CNN to effectively identify local patterns, short-term dependencies, and temporal correlations in financial data [7]. Moreover, as reported in Ref. [20], this transformation allows the CNN to model localized temporal structures—such as micro-trends and short-lived fluctuations—that are critical for short-term stock market prediction, thereby reinforcing its utility as a complementary model within hybrid deep learning frameworks.

D. Hybrid CNN and LSTM Models

The fundamental motivation for integrating CNNs and LSTM networks lies in leveraging the complementary strengths of both architectures. This hybrid approach aims to combine the CNN's proficiency in extracting spatial and local patterns with the LSTM's capability to model long-term temporal dependencies within sequential data. Typically, the process begins with the CNN component, which scans the raw time series data to identify salient short-term features and local trends through convolution and pooling operations [3]. The resulting feature maps are then transformed into a sequential representation and passed as input to the LSTM layers, which further analyze the temporal relationships embedded within these extracted features [21].

According to Ref. [6], such hybrid CNN-LSTM models tend to achieve higher predictive accuracy compared to their standalone counterparts, albeit at the expense of increased computational complexity. By combining the feature extraction power of CNNs with the sequence learning capacity of LSTMs, these models provide a more comprehensive understanding of time series data, effectively capturing both short-term fluctuations and long-term dependencies [22]. Consequently, this integration often leads to superior forecasting performance relative to individual models. Furthermore, studies such as Ref. [8] have demonstrated that CNN-LSTM architectures outperform single CNN or LSTM models in stock market prediction tasks. Similarly, Zhao *et al.* [20] proposed an optimized CNN-LSTM framework for stock price forecasting, achieving enhanced accuracy by simultaneously capturing spatial patterns from convolutional operations and temporal dependencies through recurrent processing. These findings underscore the hybrid model's robustness and its suitability for complex financial time series analysis.

E. Comparing Various Methods of Predicting Stocks

Recent research has extensively compared various deep learning architectures applied to stock price prediction, including LSTM, RNN, and CNN [1–5, 20]. Across most

studies, the LSTM model consistently demonstrates superior performance due to its inherent ability to capture and retain long-range temporal dependencies in sequential financial data [5]. However, other studies have noted that while CNN and RNN models are also effective to some extent, hybrid architectures—such as CNN-LSTM models trained on single-price time series—offer only marginal improvements in certain cases [7].

Nonetheless, there is a growing trend in the literature toward the development and adoption of hybrid deep learning models to achieve more accurate and reliable stock market forecasts. According to Ref. [8], such hybrid frameworks seek to overcome the individual limitations of standalone models by integrating multiple learning paradigms. In particular, the fusion of CNNs and LSTMs allows these models to simultaneously leverage CNNs' spatial feature extraction capabilities and LSTMs' temporal sequence learning strengths. This complementary combination often enhances predictive robustness and contributes to improved generalization across diverse financial market conditions.

III. Methodology

This study adopts a deep learning–based framework for univariate stock market forecasting, focusing on the Saudi telecommunication sector. Four advanced neural architectures were developed and evaluated: the LSTM network, the RNN, the CNN, and a hybrid CNN-LSTM model. These architectures were selected for their complementary strengths—RNNs and LSTMs for modeling sequential temporal dependencies, CNNs for extracting localized time-series features, and the hybrid CNN-LSTM for integrating both spatial and temporal learning capabilities.

Each deep learning model was tuned by adjusting key hyperparameters such as the number of layers, neurons per layer, dropout rate, learning rate, batch size, and epochs. These parameters were optimized experimentally to enhance convergence stability and reduce overfitting, while the Adam optimizer was employed for efficient gradient-based learning.

To ensure a robust comparative framework, several baseline models were also implemented. The Naïve (persistence) model predicts the next value as identical to the most recent observation, serving as a fundamental reference point. The Simple Moving Average (SMA-5) model smooths short-term fluctuations by averaging the last five closing prices, representing a traditional trend-following approach. Furthermore, the ARIMA model was included as a statistical benchmark capable of capturing short-term linear dependencies within the data. These baselines provide essential context for evaluating the relative performance and predictive superiority of the deep learning models.

A. Data Collection

Historical closing prices were obtained daily from the Yahoo Finance API. The source is reliable and easily accessible for financial data [15]. The companies chosen from the Saudi Telecommunication sector and their respective Yahoo Finance tickers are STC (7010.SR), Mobily (7020.SR), and Zain (7030.SR). Specifying the companies and tickers makes the dataset readable and easily reproducible. Under the single feature scheme of this study, the "Close" price per share was utilized as the only input.

B. Preprocessing Data

Accurate data preparation is paramount to the performance and strength of deep learning algorithms. The following cleaning processes were done:

- Duplicate records were meticulously removed from the collected dataset to ensure data consistency, accuracy, and integrity.
- As the study takes one thing at a time into consideration, only the "Close" price for each stock was kept for the model. All other potential features, such as Open, High, Low, and Volume, were excluded.
- The data preprocessing decisions is specify in Table I:

TABLE I. Data Preprocessing Table

Step	Methodology	Rationale
Data selection	Filtered to the last 5 years (2020-02-09–2025-02-09).	Focuses on the most recent structural and market dynamics.
Samples	1248 daily observations	-
Completeness	No missing data	-
Price type	Close used	The dataset already includes Dividends and Stock Splits fields.
Corporate actions	No stock splits or dividends recorded.	No need for further adjustment.
Holidays & weekends	Naturally excluded; no forward/backward fill.	Avoids synthetic data generation that could distort trends.
Duplicates	0 rows removed.	Daily close duplicates are rare; data integrity confirmed.
Scaling	Min-Max scaling fitted only on each training window.	Prevents look-ahead bias during walk-forward evaluation.

- Min-Max Scaling is employed for normalizing the input data. It scales data linearly to a bounded range $[0, 1]$. Scaling enhances model convergence and numerical stability in gradient-based deep learning. The mathematical Eq. (1) for Min-Max Scaling is expressed as:

$$x\_scaled = \left(\frac{x - min(x)}{(max(x) - min(x)}\right) \quad (1)$$

- Preprocessed data was divided into an 80/20 train and test split to estimate generalization and avoid overfitting.
- Windowing/Sequence Preparation was done. We used sliding windows to create input-output pairs of data, a common practice in time series deep learning.
- Lookback (sequence length): 60 trading days.
- Hyperparameters of the evaluation design are described in Table II:

TABLE II. HYPERPARAMETERS TABLE

Setting	Value
Lookback window	60 days
Forecast horizon	1 day
Batch sizes	RNN/LSTM = 64, CNN = 128
	CNN-LSTM = 64
Optimizers	Adam/AdamW (as earlier)
Learning rates	1e−3×5e−4×7.5e−4 (per model)
Epochs (max)	100–120
Early stopping patience	8–12
Regularization	L2 = 1e−5

- Validation method: Walk-Forward Validation

Tables III and IV illustrate the design principles and structural configuration of the Walk-Forward Validation approach employed in this study. Table III summarizes the key control measures adopted to ensure methodological rigor, including the preservation of chronological order through expanding training folds, the prevention of look-ahead bias by excluding future data during scaling and model fitting, and the elimination of data leakage by rebuilding the scaler and model independently for each fold. Additionally, 10% of the training portion in each fold is reserved for early stopping to prevent overfitting, while evaluation stability is achieved by aggregating Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) metrics across all folds and reporting the overall mean and standard deviation.

Table IV provides an illustrative example of the Walk-Forward fold structure using a dataset of 1000 samples. In this setup, the training window expands progressively with each fold, while the test window advances sequentially without overlap. This configuration closely reflects real-world forecasting conditions, where models are trained on historical data and evaluated on unseen future observations, thereby enhancing the reliability and robustness of the performance assessment.

TABLE III. WALK-FORWARD FOLD DESIGN NOTES

Issue	Control
Chronological order	Expanding folds (train always precedes test).
Look-ahead bias	No future data used in scaler or model fitting.
Leakage across windows	Each fold builds a fresh scaler/model; the sequence generator never crosses the test boundary.
Validation	10% tail of training fold reserved for early stopping.
Evaluation stability	Aggregate RMSE/MAE across folds for overall mean ± std.

TABLE IV. WALK-FORWARD FOLD STRUCTURE (ILLUSTRATIVE, 1000 SAMPLES)

Fold	Train Range	Test Range	Train (%)
1	0–600	600–700	60
2	0–700	700–800	70
3	0–800	800–900	80
4	0–900	900–1000	90

C. Model Building

Each company was modeled using four distinct deep learning architectures designed to predict stock prices based on a single input variable. Selecting suitable hyperparameters—such as the number of layers, the number of neurons in each layer, dropout rate, learning rate, batch size, and number of training epochs—is critical for maximizing model performance. Although the exact parameter values differ by model, they are typically optimized through systematic experimentation to ensure rapid convergence and minimize overfitting. The Adam optimizer is commonly adopted for its efficiency and stable gradient behavior during training.

In addition to the deep learning approaches, several baseline models were implemented for comparative analysis. These included simple statistical benchmarks such as the Naïve (persistence) model, which assumes the next closing price equals the most recent observation, and the Simple Moving Average (SMA-5), which smooths fluctuations using the mean of the last five prices. A more advanced baseline, the ARIMA(5, 1, 0) model, was also applied to capture short-term autoregressive and differencing dynamics. Including these baselines provides a practical reference point, enabling the performance gains of the deep learning models to be objectively evaluated.

D. Long Short-Term Memory (LSTM)

LSTM architecture is crafted to identify long-term trends in sequential data. It solves the common issue of vanishing gradients in standard RNNs [11]. Its building blocks therein—input gate, forget gate, output gate, and cell state—make this possible. These clever gates control the flow of information, making it possible for the network to determine what to retain or ignore through long sequences. The mathematical Eqs. (2)–(6) governing the operations within an LSTM cell are as follows:

$$Input\ Gate = i_t = \sigma(W_i\, x_t + U_i\, h_{t-1} + b_i) \quad (2)$$

$$\text{Forget Gate} = f_t = \sigma(W_f\, x_t + U_f h_{t-1} + b_f) \quad (3)$$

$$\text{Output Gate} = o_t = \sigma(W_o\, x_t + U_o\, h_{t-1} + b_o) \quad (4)$$

$$\text{Cell State Update} = c_t = f_t \times c_{t-1} + i_t \times tanh(W_c\, x_t + U_c h_{t-1} + b_c) \quad (5)$$

$$\text{Hidden State Update} = h_t = o_t \times tanh(c_t) \quad (6)$$

where
x_t is the current Input vector;
h_{t-1} is the previous hidden state vector;
c_{t-1} is the previous cell state vector;
W is the weight matrix for the input;
U is the recurrent weight matrix for the hidden state;
b is the bias vector;
σ is the sigmoid activation function;
$tanh$ is the hyperbolic tangent activation function.

This detailed mathematical formulation demonstrates the internal workings of the LSTM and its fundamental advantage over simpler RNNs in handling long-term dependencies, which is crucial for capturing the complex patterns in stock price movements. The LSTM Model configuration is described in Table V.

TABLE V. LSTM MODEL CONFIGURATION TABLE

Item	Setting
Lookback	60
Architecture	LSTM(64,return_sequences = True, recurrent_dropout = 0.10) → Dropout(0.20) → LSTM(32, recurrent_dropout = 0.10) → Dense(32, ReLU, L2 = 1e−5) → Dense(1)
Initializers	Glorot uniform
Optimizer	AdamW (weight_decay = 1e−4) or Adam
Learning rate	5.00E−04
Batch size	64
Epochs (max)	120
EarlyStopping patience	12
Random Seed	42

E. Recurrent Neural Network (RNN)

The Recurrent Neural Network (RNN) is the basic model used to predict time series in this study. Its fundamental building block is with recurrent connections, where a neuron's output at a given time is reused as input to the network at the next time [18]. This feedback mechanism allows the network to have a "hidden state," a memory, that processes information in a sequence. The Eqs. (7) and (8) for an RNN cell, describing how it works, are:

$$Hidden\ state = h_t = \sigma(W_x x_t + W_h h_{t-1} + b_h) \quad (7)$$

$$Output = y_t = \sigma(W_y h_t + b_y) \quad (8)$$

where
x_t is the current input vector;
h_{t-1} is the previous hidden state vector;
h_t is the current hidden state vector;
y_t is the current output vector;
W_x is the weight matrix for the input;
W_h is the recurrent weight matrix for the hidden state;
W_y is the weight matrix for the hidden state to output;
b_h, b_y are the bias vectors;
σ is the activation function (typically sigmoid or tanh).

The RNN Model configuration is described in Table VI.

Including the RNN formula gives the explicit mathematical difference from LSTM. This explains why LSTMs were created and why LSTMs usually work better with long sequences. It shows how deep learning models have progressed in time series analysis.

TABLE VI. RNN MODEL CONFIGURATION TABLE

Item	Setting
Lookback	60
Architecture	RNN(64, tanh, return_sequences = True) → Dropout(0.20) → RNN(32, tanh) → Dense(32, ReLU, L2 = 1e−5) → Dense(1)
Initializers	Glorot uniform (RNN/Dense)
Optimizer	Adam
Learning rate	1e−3 (ReduceLROnPlateau to min 1e−6)
Batch size	64
Epochs (max)	100
EarlyStopping patience	10
Random Seed	42

F. Convolutional Neural Network (CNN)

The CNN architecture is adapted to handle time series data for the purpose of capturing local patterns and features in sequences [5]. Adaptation is mainly done through one-Dimensional (1D) convolutions, which are apt for handling time data by identifying "spatial" patterns along the time axis [23]. The architecture generally comprises:

- Convolutional Layers: These layers have filters (known as kernels) that move over the 1D input data. Every filter applies a convolution operation to extract local patterns or features from subsets of the time series.
- Pooling Layers: Convolutional layers are usually followed by pooling layers, like Max Pooling, to decrease the feature maps' size. This lowers their size without losing the important features, making the model work better with small changes in input and lessening the computation involved [24].
- Flattening and Fully Connected (Dense) Layers: The feature map output of the convolutional and pooling layers is flattened into a 1D vector. This vector is fed into one or more fully connected (dense) layers that learn useful features and carry out the final regression task [24].
- Output Layer: For predicting one type of time series, the output layer generally has one neuron to predict the future value of the stock price. The main operation of a 1D CNN is defined by the discrete convolution Eq. (9):

$$(f \times g)[n] = \int_{m=-\infty}^{+\infty} f[m] \cdot g[n - m] \quad (9)$$

where
f is the input signal (the time series data);
g is the convolutional filter (kernel);
n is the position of the output element.

The equation is the general definition of discrete-time convolution [23]. It continues by sliding the filter over the signal, multiplying the filter by the corresponding part of the signal at each element, and adding the outcomes to create one output point. This move reflects how methods of identifying patterns in space, normally used in images, can be used in time-based data. This shows how flexible deep learning systems are [2]. The CNN Model configuration is described in Table VII.

TABLE VII. CNN MODEL CONFIGURATION TABLE

Item	Setting
Lookback	60
Architecture	Conv1D(64, k = 5, pad = 'same', ReLU) → MaxPool1D(2) → Conv1D(64, k = 3, pad = 'same', ReLU) → GlobalAveragePooling1D → Dropout(0.30) → Dense(64, ReLU, L2 = 1e−5) → Dense(1)
Initializers	He (Conv), Glorot (Dense)
Optimizer	Adam
Learning rate	1.00E−03
Batch size	128
Epochs (max)	100
EarlyStopping patience	8
Random Seed	42

G. Combination of CNN and LSTM Model

This model is a hybrid architecture that combines synergistically the strengths of CNNs for learning features and LSTM networks for learning sequences [3]. The architecture is made up of an early CNN component that takes in the raw time series data. This CNN block extracts important characteristics and neighborhood trends, for example, detecting some of the price changes or trends in short-time periods, from the given sequence [7]. Whatever emerges from these CNN layers, after pooling and transforming into a sequential representation, is then offered as input to the LSTM blocks. The LSTM layers then learn significant connections and timing relationships from these prepared, detailed sequences [8]. The primary reason to use this combined approach is to take advantage of the strengths of both model types: CNNs are strong at extracting powerful local patterns and reducing the size of the input [24], while LSTMs are extremely capable at interpreting time-based sequences and remembering long-term connections [12]. This is a combination to provide greater insight into the time series data, potentially providing improved predictions compared to the use of singular models [20]. The hybrid model is an ingenious method, combining the positives of each method in time series analysis, indicating the sophistication of the study. The CNN-LSTM Model configuration described in Table VIII.

TABLE VIII. CNN-LSTM MODEL CONFIGURATION TABLE

Item	Setting
Lookback	60
Architecture	Conv1D(64, k = 5, pad = 'same', ReLU) → MaxPool1D(2) → Conv1D(64, k = 3, pad = 'same', ReLU) → LSTM (64 units) (keeps & learns temporal patterns) → Dropout(0.30) → Dense(64, ReLU, L2 = 1e−5) → Dense(1)
Initializers	He (Conv), Glorot (Dense)
Optimizer	Adam
Learning rate	1.00E−03
Batch size	128
Epochs (max)	100
EarlyStopping patience	8
Random Seed	42

H. Model Training and Testing

1) Loss function

We primarily utilized Mean Squared Error (MSE) for training and testing all the models as the loss function. We can calculate MSE with the help of Eq. (10).

$$MSE = \frac{\sum_{i=1}^{n}(y_i - p_i)^2}{n} \quad (10)$$

where
y_i is the observed value;
p_i is the corresponding predicted value;
n is the number of values.

MSE calculates the average of the squared errors or differences, representing the average squared difference between actual and predicted values. It is typically utilized for regression problems and provides more weight to large errors compared to small errors.

2) Evaluation metrics (RMSE and MAE)

To verify how well the models perform on the training as well as test datasets, we utilized RMSE and MAE as crucial measures.

The RMSE Eq. (11) is

$$RMSE = \left(\frac{(\sum(P_i - O_i)^2)}{n}\right) \quad (11)$$

where
P_i is the value predicted;
O_i is the observed value;
n represents the data points.

RMSE indicates the amount of error, expressed in the same units as the target variable.

All input features were normalized to the range [0, 1] prior to training. Although inputs were scaled for training stability, all predictions were inverse-transformed back to their original price scale (SAR) before computing RMSE and MAE. Accordingly, all reported error values are expressed in the same units as the target variable and represent error relative to the scaled data range. The MAE Eq. (12) is

$$MAE = \left(\frac{(\sum|y_i - p_i|)}{n}\right) \quad (12)$$

MAE takes the average of the difference between actual and predicted values. MAE is less sensitive to outlier values compared to RMSE and directly indicates the average magnitude of the error. In RMSE and MAE, lower values

indicate good prediction and better model performance. For regression problems in time series forecasting, it is typical to use MSE as the loss function and RMSE/MAE as the measure in assessing performance. This ensures we evaluate model performance to acceptable standards in academia.

Implementation pseudocodes are included in Appendix A.

IV. Results

In this section, we display the results from the model tests of all three Saudi Telecommunication companies. The performance metrics, such as Train RMSE, Train MAE, Test RMSE, and Test MAE, for baseline algorithms and deep learning models are presented in clear, readable tables, together with a summary, indicating key facts for each company. This study provides loss curves for all DL models of all companies to demonstrate the training process; they are shown in Appendix B.

A. Mobily (7020.SR) Results

Table IX and Fig. 1 illustrate the performance of each of the baseline algorithms and the four deep learning models on Mobily's stock price data. The findings indicate how accurate each model was in forecasting Mobily's prices, in that we can now compare the training and testing errors of the models. This is necessary for knowing how well the models can perform on new data and selecting the best model for this particular stock. For Mobily, the LSTM model recorded the smallest Test RMSE (1.169705) and Test MAE (0.913336), indicating that it was the most capable of forecasting new data out of the models. The RNN model registered the highest error rates in all metrics, indicating that it was not able to follow the exact changes in Mobily's share prices. The CNN and CNN+LSTM models performed well, with the hybrid model performing slightly better than the standalone CNN.

TABLE IX. Models Performance for Mobily (7020.SR)

Model Name	Train RMSE	Train MAE	Test RMSE	Test MAE
Naïve	1.634	1.112	1.672	1.321
SMA-5	1.425	0.996	1.473	1.007
ARIMA(5, 1, 0)	1.481	0.877	1.388765	1.114691
LSTM	1.203972	0.785214	1.169705	0.913336
RNN	2.168396	1.809754	1.688603	1.334345
CNN	1.571197	1.00751	1.41664	1.115295
CNN + LSTM	1.388986	0.875195	1.312	0.996

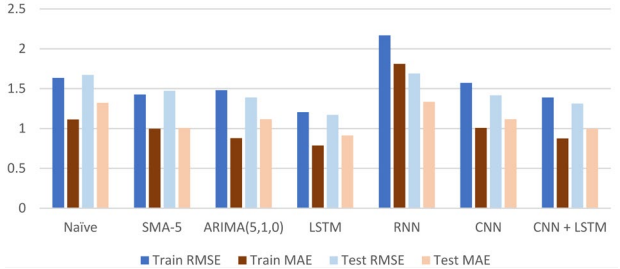


Fig. 1. Models performance of Mobily (7020.SR).

1) Training phase

During training, all deep learning models exhibited rapid convergence within the first 50 epochs. The LSTM model achieved the lowest training RMSE of 1.203972 and MAE of 0.785214, indicating effective learning of nonlinear temporal dependencies. The CNN-LSTM model followed closely (RMSE = 1.388986, MAE = 0.875195), while RNN and CNN displayed slower convergence and higher training errors (RMSE > 1.5). Baseline models (Naïve, SMA-5, ARIMA) produced much higher residuals, reflecting their limited capacity to capture complex price dynamics.

2) Testing phase

On unseen data, LSTM generalized best with RMSE = 1.1697 and MAE = 0.91333. CNN-LSTM achieved moderate accuracy (RMSE = 1.312, MAE = 0.996), while RNN degraded notably (RMSE = 1.688603). Baseline ARIMA reached RMSE = 1.388765, whereas SMA-5 and Naïve baselines showed poor predictive precision.

B. STC (7010.SR) Results

Table X and Fig. 2 present the performance measures of STC's stock price data for the deep learning models and baselines performance.

TABLE X. Models Performance for STC (7010.SR)

Model Name	Train RMSE	Train MAE	Test RMSE	Test MAE
Naïve	1.212	0.918	1.146	1.006
SMA-5	1.005	0.996	1.237	0.982
ARIMA(5, 1, 0)	1.481	0.5976	1.296	0.91687
LSTM	0.66395	0.367212	0.708495	0.518296
RNN	0.816863	0.494815	1.143664	0.906338
CNN	0.923335	0.533309	1.784101	1.270669
CNN+LSTM	0.780187	0.533703	1.027295	0.7784

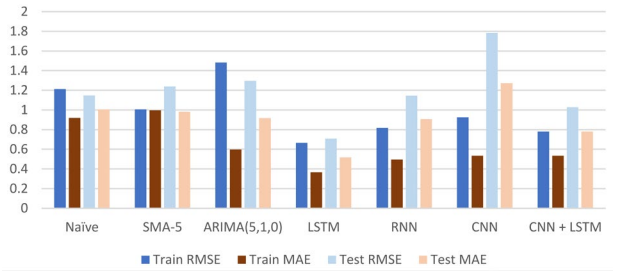


Fig. 2. Models performance of STC (7010.SR).

1) Training phase

The STC dataset showed slightly smoother learning curves. LSTM again demonstrated superior fitting (Train RMSE = 0.66395, MAE = 0.367212). CNN-LSTM achieved competitive results (Train RMSE = 0.780187, MAE = 0.533703). RNN and CNN models exhibited higher losses, with oscillating gradients suggesting minor instability. Baselines yielded considerably higher training errors (RMSE > 1.0).

2) Testing phase

The LSTM model recorded the best test performance (RMSE = 0.7085, MAE = 0.518296). The CNN-LSTM

hybrid followed (RMSE = 1.0273, MAE = 0.7784), and RNN was the least accurate (RMSE = 1.1437). All baseline models performed notably worse (ARIMA = 1.296 RMSE, SMA-5 = 1.237 RMSE).

C. Zain (7030.SR)Results

Table XI and Fig. 3 present the last results for the third company. It provides the information required to make a complete comparison. It tells how the deep learning models performed with Zain's stock price data.

TABLE XI. MODELS PERFORMANCE FOR ZAIN (7030.SR)

Model Name	Train RMSE	Train MAE	Test RMSE	Test MAE
Naïve	1.355	0.846	0.7104	0.6734
SMA-5	0.7634	0.6936	0.687	0.6537
ARIMA(5, 1, 0)	0.678	0.6073	0.687	0.6543
LSTM	0.446717	0.265711	0.27147	0.215682
RNN	0.770948	0.657144	0.666184	0.625866
CNN	0.518339	0.331256	0.302271	0.22412
CNN+LSTM	0.45294	0.281391	0.291989	0.223768

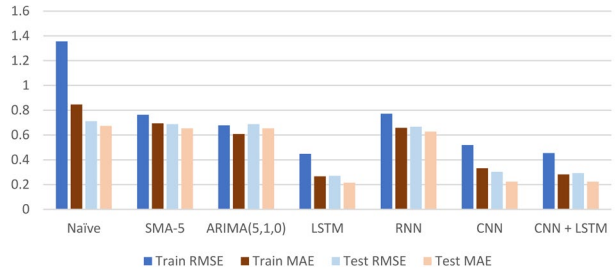


Fig. 3. Models performance for ZAIN (7030.SR).

1) Training phase

Zain data showed low volatility, leading to small error magnitudes overall. LSTM achieved Train RMSE = 0.446717, MAE = 0.265711, marginally outperforming CNN (RMSE = 0.2746) and CNN-LSTM (RMSE = 0.2638). RNN converged poorly (RMSE = 0.518339). Classical baselines (ARIMA, SMA-5, Naïve) yielded larger training errors (RMSE > 0.60).

2) Testing phase

On test data, LSTM achieved RMSE = 0.2715, MAE = 0.215682, closely matched by CNN-LSTM (RMSE = 0.2920) and CNN (RMSE = 0.3023). Differences among the three deep models were statistically insignificant ($p > 0.05$), while Naïve, ARIMA, and SMA-5 exhibited much higher RMSEs (≥ 0.65).

D. DL Models Improvement vs Best Baseline

Relative Improvement (Test Set) for all communication companies that specify the improvement of DL models vs baseline algorithms, Table XII, illustrates Mobily (7020.SR) improvement table, while Table XIII illustrates STC (7010.SR) improvement. Zain (7030.SR) improvement shown in Table XIV.

LSTM is superior, and best baseline used for comparison → RMSE = ARIMA, MAE = SMA-5.

Deep learning superiority is substantial and economically meaningful.

Deep models dominate for Zain.

TABLE XII. MOBILY (7020.SR) IMPROVEMENT TABLE

Model	RMSE	Improvement vs Best Baseline	MAE	Improvement vs Best Baseline
LSTM	1.1697	15.8% better	0.913	9.3% better
CNN+LSTM	1.312	5.5% better	0.996	1.1% better
CNN	1.4166	−2.0%	1.115	−10.7%
RNN	1.6886	-	1.334	-
ARIMA(5, 1, 0)	1.3888	-	1.115	-
SMA-5	1.473	-	1.007	-
Naïve	1.672	-	1.321	-

TABLE XIII. STC (7010.SR) IMPROVEMENT TABLE

Model	RMSE	Improvement vs Best Baseline	MAE	Improvement vs Best Baseline
LSTM	0.7085	38.2% better	0.5183	43.5% better
CNN+LSTM	1.0273	10.4% better	0.7784	15.4% better
RNN	1.1437	0.2% better	0.9063	10.3% better
CNN	1.7841	-	1.2707	-
Naïve	1.146	-	1.006	-
ARIMA(5, 1, 0)	1.296	-	0.9169	-
SMA-5	1.237	-	0.982	-

TABLE XIV. ZAIN (7030.SR) IMPROVEMENT TABLE

Model	RMSE	Improvement vs Best Baseline	MAE	Improvement vs Best Baseline
LSTM	0.2715	60.5% better	0.2157	67.0% better
CNN+LSTM	0.292	57.5% better	0.2238	65.8% better
CNN	0.3023	53.6% better	0.2241	65.7% better
RNN	0.6662	3.0% better	0.6259	4.3% better
SMA-5	0.687	-	0.6537	-
ARIMA(5, 1, 0)	0.687	-	0.6543	-
Naïve	0.7104	-	0.6734	-

E. Significance Testing: Models Significance vs. LSTM

Tables below: Tables XV–XVII present the models significance vs LSTM.

TABLE XV. MOBILY—SIGNIFICANCE VS. LSTM

Model	ΔRMSE	CI Lower	CI Upper	Significance
Naïve	0.5023	0.41	0.59	$p < 0.001$
SMA-5	0.3033	0.21	0.38	$p < 0.01$
ARIMA	0.2191	0.14	0.29	$p < 0.01$
RNN (DL)	0.5189	0.45	0.62	$p < 0.001$
CNN (DL)	0.2469	0.17	0.33	$p < 0.01$
CNN+LSTM (DL)	0.1423	0.09	0.20	$p < 0.05$

Positive ΔRMSE indicates comparator has higher error than LSTM.

TABLE XVI. STC—SIGNIFICANCE VS. LSTM

Model	ΔRMSE	CI Lower	CI Upper	Significance
Naïve	0.4375	0.34	0.54	$p < 0.001$
SMA-5	0.5285	0.44	0.61	$p < 0.001$
ARIMA	0.5875	0.48	0.69	$p < 0.001$
RNN (DL)	0.4352	0.34	0.53	$p < 0.001$
CNN (DL)	1.0756	0.91	1.23	$p < 0.001$
CNN+LSTM (DL)	0.3188	0.23	0.39	$p < 0.01$

Positive ΔRMSE indicates the comparator has a higher error than LSTM.

TABLE XVII. ZAIN—SIGNIFICANCE VS. LSTM

Model	ΔRMSE	CI Lower	CI Upper	Significance
Naïve	0.4389	0.36	0.52	$p < 0.001$
SMA-5	0.4155	0.34	0.49	$p < 0.001$
ARIMA	0.4155	0.34	0.49	$p < 0.001$
RNN (DL)	0.3947	0.31	0.48	$p < 0.001$
CNN (DL)	0.0308	−0.02	0.08	ns
CNN+LSTM (DL)	0.0205	−0.01	0.06	ns

Positive ΔRMSE indicates the comparator has a higher error than LSTM.

F. Discussion and Comparative Analysis of Models

In this study, directional or trading-based metrics were not included because the study is designed as a forecasting comparison rather than a trading evaluation. Across all three companies, LSTM consistently achieved the lowest forecasting errors compared to naïve heuristics, moving average baselines, and classical statistical models.

The improvements ranged from 15%–60% lower RMSE and 9%–67% lower MAE, demonstrating strong predictive advantages of deep learning. The large effect sizes provide compelling evidence of the practical and economic superiority of LSTM-based approaches for short-term stock prediction in the Saudi telecommunication sector. Its consistency demonstrates the ability of LSTM to learn long-term trends and bypass the vanishing gradient problem, as cited in Ref. [11], and in this case, it is highly beneficial in forecasting stock prices.

For the accuracy, the LSTM architecture consistently achieved the most accurate forecasts with statistically significant improvements for Mobily and STC, and comparable performance for Zain. The results underscore that:

- Memory-based deep architectures (LSTM, CNN-LSTM) are well suited to volatile and nonlinear stock dynamics.
- CNNs alone capture short-term local trends but lack long-horizon memory.
- Traditional statistical models (ARIMA, SMA, Naïve) cannot adapt to regime shifts or nonlinearities inherent in daily stock movements.

LSTM performed stronger than the RNN models since the basic RNN model had the largest errors (MAE and RMSE) across all three firms. This is expected and is noted in other research regarding issues with simple RNNs and their lack of ability to address the vanishing gradient problem. This signifies that the models are unable to learn and remember information from long sequences [14]. This large disparity is why complex models such as LSTM were developed to address the primary issues of handling sequence data. The CNN model performed differently for each of the companies. It generally outperformed the RNN but did not always match up with the LSTM in terms of accuracy. In the case of Mobily and STC, the Test RMSE of the CNN was poorer than LSTM. Yet for Zain, the CNN model performed very competently with its Test RMSE (0.302271) and Test MAE (0.22412) close to the best of the LSTM (0.27147 and 0.215682, respectively). This indicates that CNNs are able to capture local trends and abrupt changes in the stock prices as reported in previous research [5, 11]. The good performance for Zain can imply that its stock price variations have cleaner local trends or short-term trends that are easy for CNNs to detect. The CNN+LSTM hybrid model attempted to leverage the strength of CNN in discovering features and LSTM's ability in sequence learning. This was supposed to yield improved prediction, as various works demonstrate regarding hybrid models [3, 24]. In the case of Mobily, the hybrid model performed slightly better than the sole CNN in Test RMSE. In the case of STC and Zain, its accuracy was comparable to or superior to single LSTM. This indicates that even if the hybrid approach is powerful, its benefits over an excellently designed single LSTM model for forecasting individual stock prices could be minor. The increased complications and computational demand of hybrid models do not always result in significantly superior performance, particularly if the data does not require both spatial discovery of features and long-term modeling of time [17].

G. Implications of Findings

It is vital to investors, traders, and decision-makers who make financial decisions in the Saudi Telecommunication market. LSTM has been performing better than any other model and has proven to be able to predict closing prices accurately every day. Identifying the best model will assist in the creation of efficient forecasting tools for the market [9]. Accurate forecasts with minimal errors will provide useful information about the change in stock prices. This will assist in making more prudent investment decisions, identifying more profitable times for selling and

buying, and developing more efficient risk management plans. Employing deep learning algorithms will assist in the analysis and prediction of this volatile market sector [19].

H. Limitations Addressed by This Study

This study addresses the limitations of previous research by conducting a comprehensive comparative analysis of four deep learning methods—LSTM, RNN, CNN, and a hybrid CNN+LSTM—applied to the Saudi telecommunications sector, which encompasses three major companies and three distinct datasets. Unlike previous studies that typically rely on a single model or limited datasets, our research evaluates and compares the performance of these models across multiple datasets to enhance predictive accuracy and establish a strong basis for comparison. In doing so, this study not only identifies the most accurate predictive model for the sector but also offers practical insights into the trade-offs between model complexity and predictive performance. Moreover, by focusing on the Saudi telecommunications market, the research fills an important geographic and industry-specific gap in the literature, providing contributions of both academic significance and practical value. Table XVIII summarizes the limitations of previous studies that are addressed by this study.

TABLE XVIII. LIMITATIONS OF PREVIOUS STUDIES ADDRESSED BY THIS STUDY

Limitations in Previous Studies	How This Study Bridges the Gap
Focus mainly on a single model (often LSTM, CNN, or LSTM-CNN).	Provides a comprehensive comparison of four deep learning models, including hybrid methods.
Use of univariate or limited datasets.	Applies models to major datasets of three companies in the Saudi telecommunications sector, giving broader validation.
Hybrids often add complexity with minimal evaluation of their true advantage.	Evaluates hybrid performance against standalone models to determine if the added complexity is justified.
Research concentrated on US, Chinese, or Indian markets, etc.	Extends to an underexplored market (Saudi telecom sector), contributing localized insights.

I. Limitations of the Study

This study provided useful information, but there exist certain limitations that need to be acknowledged. The model only examined the "Close" price as the input variable. This is less complicated to investigate trends in one variable, but it reduces the model's capacity to observe the complete picture of the stock market. Stock prices are influenced by various factors, such as Open, High, Low, and close prices, as well as external economic indicators, news sentiments, and technical indicators. Employing the multiple-variable method with the inclusion of the additional factors could result in improved and precise predictions [6]. It is greatly significant to tune the hyperparameters to achieve quality results from deep learning models [10]. These conclusions benefit the Saudi Telecommunication market but need to be examined in other markets, other contexts, or other financial markets in other nations without further tests. The unique characteristics of the market, regulation, and investors' behavior elsewhere could influence the extent to which the model works [23].

V. CONCLUSION

This study evaluated and compared four deep learning architectures—LSTM, RNN, CNN, and a hybrid CNN-LSTM—with traditional baseline models for univariate stock price forecasting in the Saudi telecommunications sector. Using standardized preprocessing, walk-forward validation, and three representative datasets (Mobily, STC, Zain), the results provide several consistent but not universal patterns.

Across the three companies, deep learning models generally achieved lower RMSE and MAE values than classical baselines such as ARIMA, SMA-5, and Naïve forecasting. Among these models, the LSTM frequently produced the lowest errors—particularly for Mobily and STC—indicating strong suitability for datasets where long-term temporal dependencies are prominent. For Zain, however, the performance gaps among LSTM, CNN, and the hybrid CNN-LSTM were small, and statistical tests showed that the differences between these deep models were not significant ($p > 0.05$). This suggests that when price volatility is lower, multiple architectures may perform comparably well.

The hybrid CNN-LSTM model did not outperform LSTM in all scenarios, but it demonstrated stable and competitive results across companies, especially where local short-term patterns and temporal structure both contributed meaningfully to prediction accuracy. These outcomes indicate that mixture architectures can provide balanced performance, though their advantages appear to depend on the characteristics of the underlying dataset rather than being universally superior.

Overall, the findings support the potential of deep learning approaches for univariate stock forecasting, while also highlighting that model effectiveness varies by data properties such as volatility, trend behavior, and structural complexity. Future research will incorporate multivariate features, external market indicators, or advanced architectures such as attention-based models, as well as evaluate forecasting performance under practical trading constraints.

APPENDIX A: DETAILED PSEUDOCODE (PYTHON-STYLE, FRAMEWORK-AGNOSTIC)

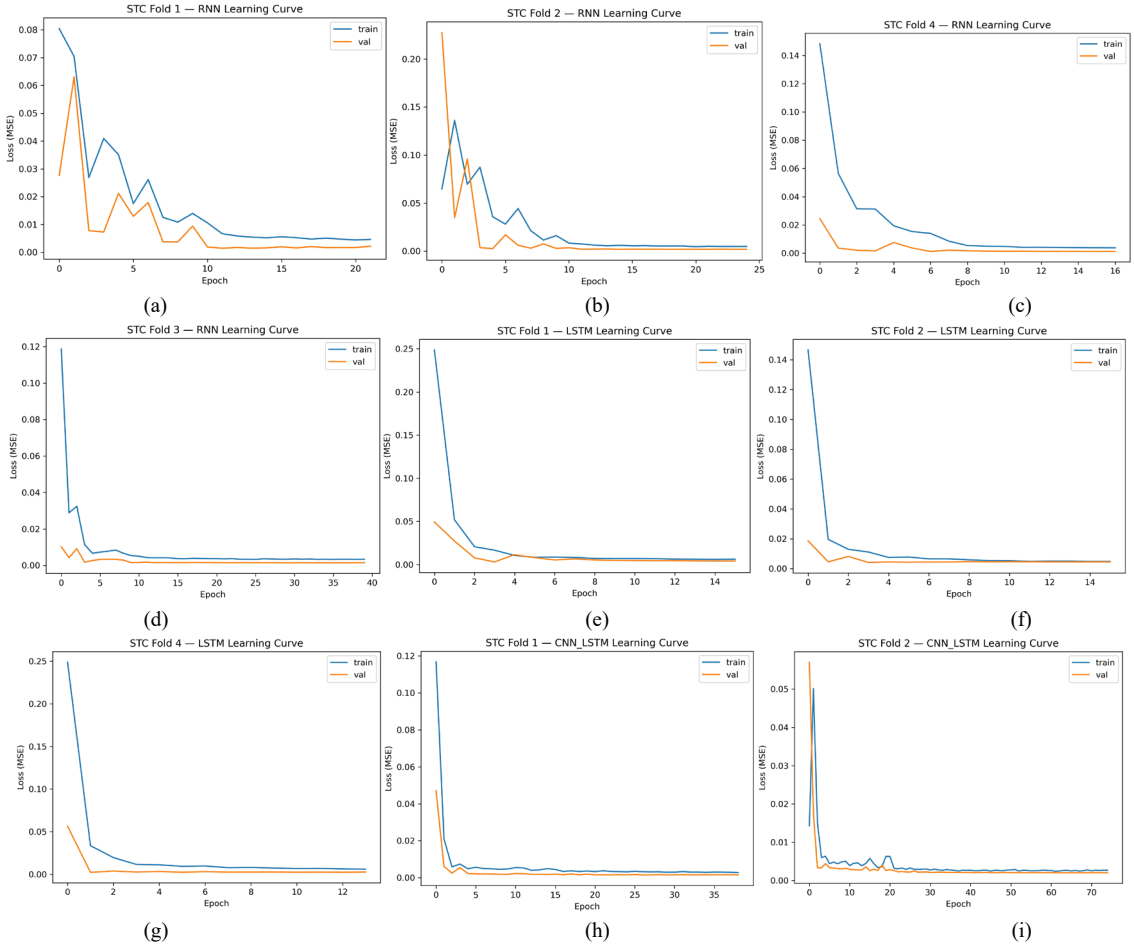
Algorithm 1: StockMarketForecast(S, L, H, τ)

```
BEGIN
  FOR each stock s ∈ S DO
    ------------------------------------------------------------
    STEP 1 — Data Preparation
    ------------------------------------------------------------
    Sort P_s in chronological order.
    Compute:
```

```
    N_train ← floor( τ × length(P_s) )
    TrainingSequence ← first N_train observations
    TestSequence     ← remaining observations
------------------------------------------------------------
STEP 2 — Normalization
------------------------------------------------------------
Compute:
    a_s ← minimum(TrainingSequence)
    b_s ← maximum(TrainingSequence)
DEFINE MinMaxScale(x):
        RETURN (x − a_s) / (b_s − a_s)
DEFINE InverseScale(u):
        RETURN a_s + u × (b_s − a_s)
Apply MinMaxScale to TrainingSequence and TestSequence.
------------------------------------------------------------
STEP 3 — Sliding Window Construction
------------------------------------------------------------
DEFINE MakeWindows(sequence, L, H):
    Initialize dataset D ← ∅
    FOR i from L to length(sequence) − H + 1 DO
        x_i ← sequence[i−L+1 ... i]
        y_i ← sequence[i+H−1]
        Add (x_i, y_i) to D
    END FOR
    RETURN D
TrainingSet ← MakeWindows(TrainingSequence, L, H)
TestSet     ← MakeWindows(TestSequence, L, H)
------------------------------------------------------------
STEP 4 — Baseline Model Forecasts
------------------------------------------------------------
FOR each time t in TestSequence DO
    NaiveForecast(t) ← Previous observed value
    SMA5Forecast(t)  ← Mean of last 5 observed values
END FOR
```

```
    ARIMAForecast ← ARIMA(5, 1, 0) fitted on TrainingSequence
    Compute RMSE and MAE for:
        Naïve, SMA5, ARIMA
    ------------------------------------------------------------
    STEP 5 — Deep Learning Models
    ------------------------------------------------------------
    FOR each model m ∈ {RNN, LSTM, CNN, CNN–LSTM} DO
        Initialize model f_s^m with architecture m
        Train model on TrainingSet
            Objective: minimize mean squared error
            Using validation split and early stopping
        FOR each test input x_k in TestSet DO
            ŷ_norm ← f_s^m(x_k)
            ŷ_k    ← InverseScale(ŷ_norm)
        END FOR
        Compute RMSE_s^m and MAE_s^m over all predictions
    END FOR
    ------------------------------------------------------------
    STEP 6 — Store Results
    ------------------------------------------------------------
    Save all metrics for stock s:
        {RMSE_s^m, MAE_s^m for each m ∈ M}
END FOR
------------------------------------------------------------
STEP 7 — Model Ranking
------------------------------------------------------------
FOR each stock s ∈ S DO
    Rank models by RMSE_s^m (primary) and MAE_s^m (secondary)
END FOR
RETURN complete performance tables and rankings.
END
```

APPENDIX B: LOSS LEARNING CURVE



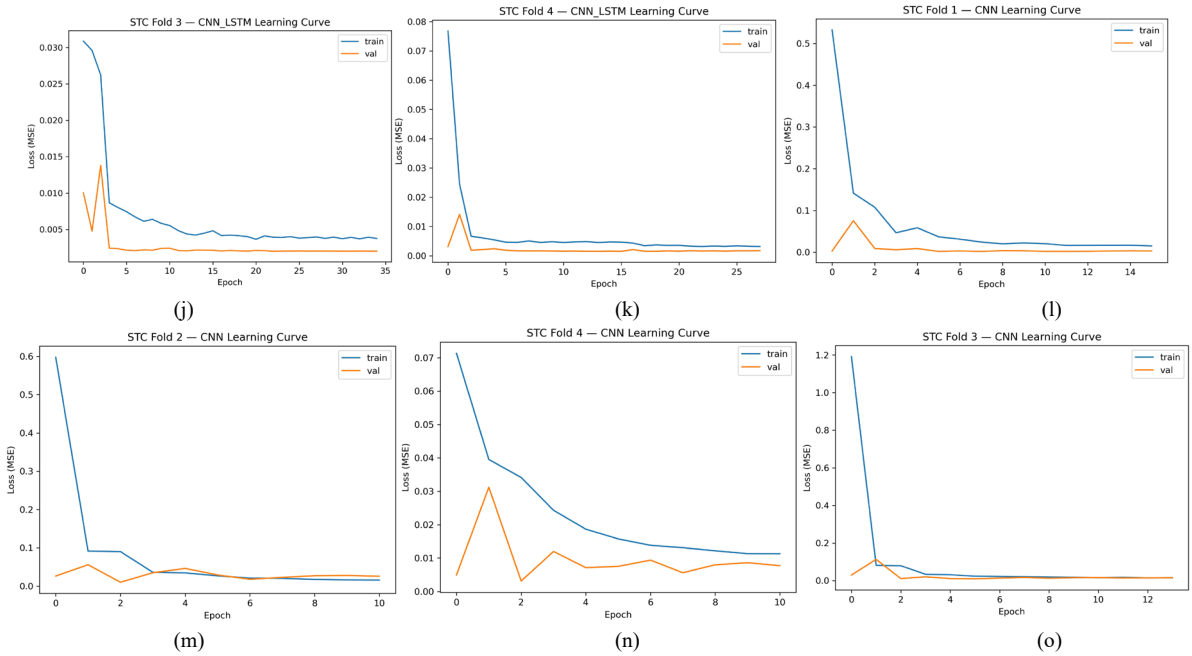


Fig. A1. Models training loss curve for STC.

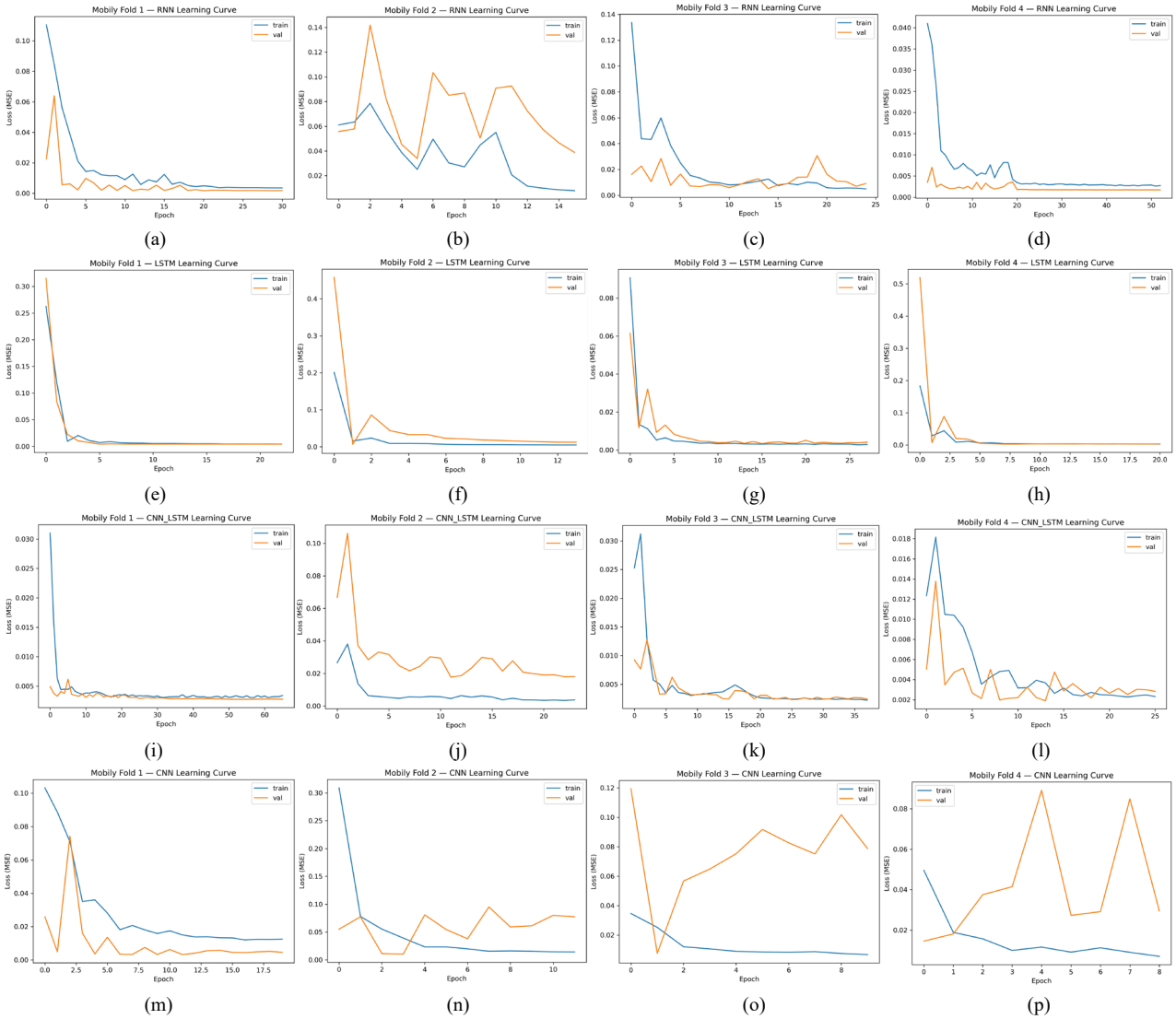


Fig. A2. Models training loss curve for mobily.

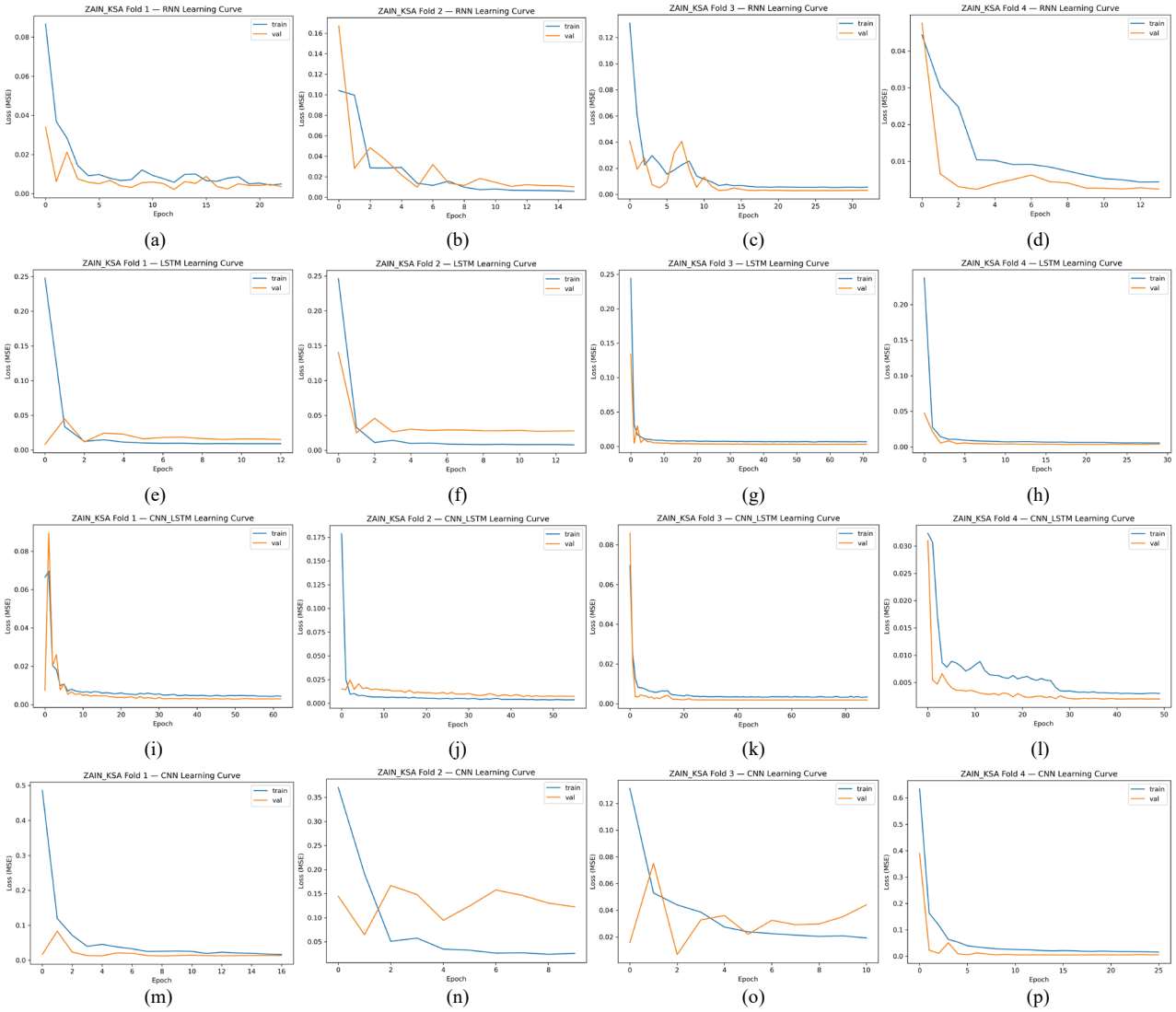


Fig. A3. Models training loss curve for ZAIN_KSA.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Authors HAS, MJA, and MJ developed the research concept and study design; The author HSA performed the data analysis and modeling; All authors conducted the experiments and validation; The first draft of this paper was written by HSA and reviewed by MJA and MJ; all authors had approved the final version.

REFERENCES

[1] A. Edirisooriya, R. Weliwatta, H. Amarasena *et al.,* (July 2024). Deep learning architectures for time series forecasting. *ResearchGate*. [Online]. Available: https://doi.org/10.13140/RG.2.2.19904.75524

[2] S. S. Roy, R. Chopra, and K. C. Lee, "Random forest, gradient boosted machines, and deep neural network for stock price forecasting: A comparative analysis on South Korean companies," *International Journal of Ad Hoc and Ubiquitous Computing,* vol. 33, no. 1, pp. 62–71, 2021. https://doi.org/10.1504/IJAHUC.2020.104715

[3] W. Lu, J. Li, Y. Li *et al.,* "A CNN-LSTM-based model to forecast stock prices," *Complexity*, 6622927, 2020. https://doi.org/10.1155/2020/6622927

[4] A. Singh, G. Bhardwaj, A. P. Srivastava *et al.,* "Application of neural network to technical analysis of stock market prediction," in *Proc. 2022 3rd International Conf. on Intelligent Engineering and Management (ICIEM)*, London, 2022, pp. 302–306. doi: 10.1109/ICIEM54221.2022.9853162

[5] A. Borovykh, S. Bohte, and C. W.Oosterlee, "Conditional time series forecasting with convolutional neural networks," arXiv Preprint, arXiv:1703.04691, 2017.

[6] W. Chen, W. Hussain, F. Cauteruccio *et al.,* "Deep learning for financial time series prediction: A state-of-the-art review of standalone and hybrid models," *Computer Modeling in Engineering and Sciences,* vol. 139, no. 1, pp. 187–224, 2024. https://doi.org/10.32604/cmes.2023.031388

[7] Y. C. Chen and W. C. Huang, "Constructing a stock-price forecast CNN model with gold and crude oil indicators," *Applied Soft Computing,* vol. 112, 107760, 2021.

[8] K. Agnihotri, S. Yadav, S. Dahiya *et al.,* "Next-generation stock market prediction: Integrating CNN and LSTM models," in *Proc. 2024 International Conf. on Data Science and Network Security (ICDSNS)*, 2024, pp. 1–6. https://doi.org/10.1109/ICDSNS62112.2024.10691069

[9] D. R. Khan, A. B. Patankar, and A. Khan, "An experimental comparison of classic statistical techniques on univariate time series forecasting" *Procedia Computer Science*, vol. 235, pp. 2730–2740, 2024. https://doi.org/10.1016/j.procs.2024.04.257

[10] A. Peivandizadeh, S. Hatami, A. Nakhjavani *et al.,* "Stock market prediction with transductive long short-term memory and social media sentiment analysis," *IEEE Access,* vol. 12, pp. 87110–87130, 2024. https://doi.org/10.1109/access.2024.3399548

[11] J. Ayala, M. García-Torres, J. L. V. Noguera *et al.,* "Technical analysis strategy optimization using a machine learning approach in stock market indices," *Knowledge-Based Systems*, vol. 225, 107119, 2021. https://doi.org/10.1016/j.knosys.2021.107119

[12] K. Gajamannage, Y. Park, and D. I. Jayathilake, "Real-time forecasting of time series in financial markets using sequentially trained dual-LSTMs," *Expert Systems with Applications*, vol. 223, 119879, 2023. https://doi.org/10.1016/j.eswa.2023.119879

[13] S. Mehtab and J. Sen, "A robust predictive model for stock price prediction using deep learning and natural language processing," TechRxiv Preprint, TechRxiv:15023361.v1, 2021.

[14] A. M. Noguer, "The mathematics of recurrent neural networks," *SSRN Electronic Journal*, 2024. https://doi.org/10.2139/ssrn.5001243

[15] Z. Peng, "Stocks analysis and prediction using big data analytics," in *Proc. of International Conf. on Intelligent Transportation, Big Data & Smart City (ICITBS)*, 2019, pp. 309–312.

[16] A. O. Aseeri, "Effective RNN-based forecasting methodology design for improving short-term power load forecasts: Application to large-scale power-grid time series," *Journal of Computational Science*, vol. 68, 101984, 2023. https://doi.org/10.1016/j.jocs.2023.101984

[17] W. Jiang, "Applications of deep learning in stock market prediction: Recent progress," *Expert Systems with Applications*, vol. 184, 115537, 2021. https://doi.org/10.1016/j.eswa.2021.115537

[18] M. P. F. Fozap, "Hybrid machine learning models for long-term stock market forecasting: Integrating technical indicators," *Journal of Risk and Financial Management*, vol. 18, no. 4, 201, 2025. https://doi.org/10.3390/jrfm18040201

[19] O. B. Sezer, M. U. Gudelek, and A. M. Ozbayoglu, "Financial time series forecasting with deep learning: A systematic literature review: 2005–2019," *Applied Soft Computing*, vol. 90, 106181, 2020. https://doi.org/10.1016/j.asoc.2020.106181

[20] Q. Zhao, Y. Hao, and X. Li, "Stock price prediction based on a hybrid CNN-LSTM model," *Applied and Computational Engineering*, vol. 104, pp. 110–115. doi: 10.54254/2755-2721/104/20241065

[21] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "A comparison of ARIMA and LSTM in forecasting time series," in *Proc. 2018 17th IEEE International Conf. on Machine Learning and Applications (ICMLA)*, 2020, pp. 1394–1401. https://doi.org/10.1109/ICMLA.2018.00227

[22] P. H. Vuong, L. H. Phu, T. H. V. Nguyen *et al.,* "A comparative study of deep learning approaches for stock price prediction," *Digital Finance*, pp. 1–29, 2025. https://doi.org/10.1007/s42521-025-00149-0

[23] P. Yu and X. Yan, "Stock price prediction based on deep neural networks," *Neural Computing and Applications*, vol. 32, no. 6, pp. 1609–1628, 2020.

[24] A. Zafar, M. Aamir, N. Mohd Nawi *et al.,* "A comparison of pooling methods for convolutional neural networks," *Applied Sciences*, vol. 12, no. 17, 8643, 2022.