



IJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: <https://doi.org/10.22214/ijraset.2026.81884>

www.ijraset.com

Call:  08813907089

E-mail ID: ijraset@gmail.com

Deep Multi-Scale LSTM-Transformer Architecture for Financial Time Series Forecasting

Hrishik Choudry¹, Dr.Dattatreya P. Mankame²

School of Computer Science and Engineering, RV University, Bengaluru, India

Abstract: *Financial time series forecasting remains a formidable challenge due to inherent non-stationarity, volatility, and complex multi-scale temporal dependencies. This paper presents a deep hybrid neural network architecture that integrates stacked Long Short-Term Memory (LSTM) networks with high-capacity multi-head self-attention for multi-scale financial prediction. Unlike conventional approaches that rely on hand-engineered technical indicators or explicit sinusoidal positional encoding, our proposed model processes raw Open-High-Low-Close-Volume (OHLCV) data augmented with Unix timestamps across four temporal resolutions (15-minute, 30-minute, 60-minute, and daily intervals). The architecture employs 256 attention heads with extensive cross-scale attention mechanisms to model interactions between temporal granularities, followed by deep dense projection layers for final price prediction. Experimental evaluation on HDFC Bank stock data from 2008 to 2025 demonstrates strong performance with validation Mean Absolute Error (MAE) of approximately 6.71 and Mean Absolute Percentage Error (MAPE) of 0.69%. The proposed 116-million-parameter model achieves robust generalization without requiring explicit positional encoding or traditional technical indicators, offering a streamlined end-to-end approach to financial forecasting. Our implementation utilizes PySpark for scalable data preprocessing and TensorFlow for model training, with Huber loss optimization for enhanced robustness against market outliers.*

Keywords—Financial time series forecasting, LSTM, Transformer, hybrid neural networks, multi-scale learning, deep learning, stock price prediction, attention mechanisms

I. INTRODUCTION

Financial time series forecasting represents one of the most challenging domains in predictive analytics, characterized by high volatility, non-linear dynamics, and the efficient market hypothesis suggesting inherent unpredictability [1], [2]. Despite these challenges, accurate prediction of stock prices remains crucial for portfolio optimization, risk management, and algorithmic trading strategies [3], [4].

Traditional statistical approaches, including Autoregressive Integrated Moving Average (ARIMA) and Generalized Autoregressive Conditional Heteroskedasticity (GARCH) models, have long served as benchmarks for financial forecasting [5], [6]. However, these linear models struggle to capture complex non-linear relationships and regime-switching behaviors prevalent in modern financial markets [7], [8]. The emergence of deep learning has catalyzed a paradigm shift, with Recurrent Neural Networks (RNNs) and their variants demonstrating superior capability in modeling temporal dependencies [9], [10].

The Long Short-Term Memory (LSTM) network, introduced by Hochreiter and Schmidhuber [9], addressed the fundamental vanishing gradient problem through its sophisticated gating mechanism comprising input, forget, and output gates. This innovation enabled modeling of long-range temporal dependencies essential for financial analysis [11], [12]. Subsequent developments in attention mechanisms, particularly the Transformer architecture by Vaswani et al. [13], revolutionized sequence modeling by enabling parallel computation and direct modeling of pairwise relationships through self-attention.

Recent research has increasingly focused on hybrid architectures combining the sequential processing strengths of LSTM with the global dependency modeling of Transformers [14], [15]. These approaches have demonstrated superior performance in capturing both local temporal patterns and long-range market dynamics [16], [17]. However, many existing implementations rely heavily on extensive feature engineering, including numerous technical indicators and explicit positional encoding schemes that may introduce unnecessary complexity and inductive bias [18], [19].

This paper proposes a streamlined yet deep multi-scale LSTM-Transformer architecture that eliminates both explicit positional encoding and traditional technical indicators. Our approach leverages raw OHLCV data augmented with Unix timestamps, relying on LSTM hidden states for implicit temporal encoding and multi-head self-attention for cross-scale dependency modeling. Our key contributions include:

- 1) A novel deep hybrid architecture integrating stacked LSTM layers with high-capacity multi-head self-attention (256 heads) for multi-timeframe financial forecasting, comprising 115.9 million parameters.
- 2) Elimination of explicit sinusoidal positional encoding through a combination of LSTM-based temporal encoding and raw unix timestamp features, leveraging the inherent sequential processing capability of recurrent layers.
- 3) A minimal feature set consisting solely of raw OHLCV data and unix timestamps, demonstrating that computed technical indicators (moving averages, RSI, MACD, stochastic oscillators) are superfluous when sufficient deep learning capacity is provided.
- 4) Comprehensive experimental validation on HDFC Bank stock data spanning 17 years across multiple temporal resolutions, achieving validation MAPE below 0.7%.

II. RELATED WORK

A. Deep Learning for Financial Forecasting

The application of deep learning to financial time series has evolved substantially. Early work by Bao et al. [20] demonstrated effectiveness of stacked autoencoders, while Fischer and Krauss [21] applied LSTM networks to S&P 500 constituents with promising results. The inherent capability of LSTM to maintain cell states over extended sequences makes it particularly suitable for financial applications [22], [23].

Convolutional Neural Networks (CNNs) have also been extensively applied, particularly for capturing local patterns [24], [25]. Hybrid CNN-LSTM architectures have emerged as a dominant paradigm [26], [27]. More recently, attention mechanisms have been integrated to enhance interpretability [28], [29].

B. Transformer Architectures in Finance

The Transformer architecture has been successfully adapted for time series forecasting [13], [30]. Self-attention enables direct modeling of pairwise temporal relationships, circumventing the sequential processing bottleneck of RNNs [31], [32]. For financial applications, Transformers have demonstrated efficacy in capturing long-range dependencies [33], [34].

Several studies have explored LSTM-Transformer hybrids. Kabir et al. [14] proposed the LSTM-mTrans-MLP architecture, achieving state-of-the-art performance across multiple datasets. Zheng [15] developed a hybrid approach leveraging LSTM for local features and Transformer for global dependencies. Our work extends these approaches with significantly deeper attention mechanisms and explicit cross-scale attention fusion.

C. Feature Engineering vs. End-to-End Learning

A persistent debate in financial forecasting concerns the necessity of technical indicators. While many works incorporate moving averages, RSI, MACD, and Bollinger Bands [35], [36], recent deep learning research suggests that sufficiently large architectures can discover equivalent representations through end-to-end learning [37], [38]. Our work provides empirical evidence supporting this hypothesis in the multi-scale financial forecasting domain.

III. METHODOLOGY

A. Problem Formulation

We formulate financial time series forecasting as a supervised multi-input sequence-to-one regression task. Given historical windows of length $T=60$ with $F=6$ features at four temporal scales, we predict the next period's closing price:

$$\hat{P}_{t+1} = f(\mathbf{X}_{t-59:t}^{(d)}, \mathbf{X}_{t-59:t}^{(15)}, \mathbf{X}_{t-59:t}^{(30)}, \mathbf{X}_{t-59:t}^{(60)}) \quad (1)$$

where $\mathbf{X}^{(d)}$, $\mathbf{X}^{(15)}$, $\mathbf{X}^{(30)}$, and $\mathbf{X}^{(60)}$ represent daily, 15-minute, 30-minute, and 60-minute features respectively, and $f(\cdot)$ denotes our hybrid neural network.

B. Data Preprocessing

Our data pipeline utilizes PySpark for scalable distributed processing of historical stock data. The preprocessing pipeline consists of:

- 1) Data Ingestion: Multi-timeframe data loaded from JDBC-connected PostgreSQL databases.
- 2) Missing Value Imputation: Null values replaced with -999 sentinel values for neural network handling.

- 3) **Technical Indicator Computation:** Moving averages (10, 20, 50, 100, 200), RSI, MACD, and Stochastic Oscillator are computed for analysis but subsequently excluded from model inputs.
- 4) **Feature Selection:** Retention of core features (open, high, low, close, volume) plus unix timestamp derived from date time.
- 5) **Window Generation:** Sliding window creation with lookback period of 60 time steps across all time frames.
- 6) **Per-Branch Normalization:** Feature-wise normalization using TensorFlow's Normalization layer adapted independently to each temporal branch.

Notably, while we implemented sinusoidal positional encodings schemes (hour, day, month, minute cyclical encodings), these were ultimately excluded from the final model in favor of LSTM implicit encoding and raw unix timestamps. This design choice is motivated by the observation that deep architectures with sufficient capacity can discover optimal temporal representations without hand-engineered features [37], [38].

C. Model Architecture

Our proposed architecture consists of four parallel processing branches corresponding to different temporal resolutions, followed by extensive cross-scale attention fusion and deep final prediction layers. The complete model comprises 115,898,189 trainable parameters.

D. LSTM Temporal Encoder

Each temporal branch employs a two-layer stacked LSTM architecture with 60 units per layer:

$$\mathbf{h}^{(1)} = \text{LSTM}_1(\mathbf{x}_t, \mathbf{h}^{(1)}, \mathbf{c}^{(1)}) \tag{2}$$

$$\mathbf{h}^{(2)} = \text{LSTM}_2(\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{c}^{(2)}) \tag{3}$$

where $\mathbf{x}_t \in \mathbb{R}^6$ represents the input features (open, high, low, close, volume, date unix) at time t . The LSTM cell operations follow the standard formulation [9]:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f) \tag{4}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i) \tag{5}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o) \tag{6}$$

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{W}_c [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c) \tag{7}$$

$$\mathbf{C}_t = \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{C}}_t \tag{8}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t) \tag{9}$$

Crucially, the LSTM layers served dual purposes: (1) temporal feature extraction and (2) implicit positional encoding through sequential processing of hidden states. The inclusion of raw unix timestamps provides absolute temporal reference, eliminating the need for explicit sinusoidal positional encoding [13], [39].

E. Multi-Head Self-Attention

Each LSTM-encoded sequence is processed by multi-head self-attention with **256 attention heads** and key dimension $d_k=64$:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax} \left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}} \right) \mathbf{V} \tag{10}$$

The large number of attention heads enables the model to jointly attend to information from many representation subspaces simultaneously, capturing fine-grained dependencies across different feature dimensions and time lags. Each self-attention layer projects the 60-dimensional LSTM outputs through query, key, and value transformations before output projection back to 60 dimensions.

F. Cross-Scale Attention Fusion

We employ extensive cross-scale attention mechanisms enabling information flow between temporal resolutions:

$$\mathbf{A}_{15,30} = \text{MultiHeadAttn}(\mathbf{H}_{15}, \mathbf{H}_{30}) \tag{11}$$

$$\mathbf{A}_{30,60} = \text{MultiHeadAttn}(\mathbf{H}_{30}, \mathbf{H}_{60}) \tag{12}$$

$$\mathbf{A}_{15day} = \text{MultiHeadAttn}(\mathbf{H}_{15}, \mathbf{H}_{day}) \tag{13}$$

$$\mathbf{A}_{30day} = \text{MultiHeadAttn}(\mathbf{H}_{30}, \mathbf{H}_{day}) \tag{14}$$

$$\mathbf{A}_{60day} = \text{MultiHeadAttn}(\mathbf{H}_{60}, \mathbf{H}_{day}) \tag{15}$$

$$\mathbf{c}_1 = \text{MultiHeadAttn}(\mathbf{A}_{15,30}, \mathbf{A}_{30,60}) \tag{16}$$

$$\mathbf{c}_2 = \text{MultiHeadAttn}(\mathbf{c}_1, \mathbf{H}_{day}) \tag{17}$$

These cross-attention layers allow the model to align and fuse information across different temporal granularities, enabling fine-grained intraday patterns to inform daily predictions and vice versa.

G. DeepOutputProjection

The attended representations are concatenated and processed through deep dense layers:

$$\mathbf{z} = \text{Concat}([\mathbf{H}_{day}, \mathbf{H}_{15}, \mathbf{H}_{30}, \mathbf{H}_{60}, \mathbf{A}_{15day}, \mathbf{A}_{30day}, \mathbf{A}_{60day}, \mathbf{c}_1, \mathbf{c}_2]) \tag{18}$$

$$\mathbf{z} \in \mathbb{R}^{60 \times 540} \tag{19}$$

$$\mathbf{a}_1 = \text{MultiHeadAttn}(\mathbf{z}, \mathbf{z}) \tag{20}$$

$$\mathbf{a}_2 = \text{MultiHeadAttn}(\mathbf{a}_1, \mathbf{a}_1) \tag{21}$$

$$\mathbf{d}_1 = \text{ReLU}(\mathbf{W}_1 \mathbf{a}_2 + \mathbf{b}_1), \quad \mathbf{d}_1 \in \mathbb{R}^{60 \times 540} \tag{22}$$

$$\mathbf{d}_2 = \text{ReLU}(\mathbf{W}_2 \mathbf{d}_1 + \mathbf{b}_2), \quad \mathbf{d}_2 \in \mathbb{R}^{60 \times 540} \tag{23}$$

$$\mathbf{d}_3 = \text{ReLU}(\mathbf{W}_3 \mathbf{d}_2 + \mathbf{b}_3), \quad \mathbf{d}_3 \in \mathbb{R}^{60 \times 540} \tag{24}$$

$$\mathbf{g} = \text{GlobalAveragePooling1D}(\mathbf{d}_3), \quad \mathbf{g} \in \mathbb{R}^{540} \tag{25}$$

$$\mathbf{h}_1 = \text{Linear}(\mathbf{W}_4 \mathbf{g} + \mathbf{b}_4), \quad \mathbf{h}_1 \in \mathbb{R}^{240} \tag{26}$$

$$\mathbf{h}_2 = \text{ReLU}(\mathbf{W}_5 \mathbf{h}_1 + \mathbf{b}_5), \quad \mathbf{h}_2 \in \mathbb{R}^{120} \tag{27}$$

$$\mathbf{h}_3 = \text{Linear}(\mathbf{W}_6 \mathbf{h}_2 + \mathbf{b}_6), \quad \mathbf{h}_3 \in \mathbb{R}^{60} \tag{28}$$

$$\mathbf{h}_4 = \text{ReLU}(\mathbf{W}_7 \mathbf{h}_3 + \mathbf{b}_7), \quad \mathbf{h}_4 \in \mathbb{R}^{30} \tag{29}$$

$$\hat{y} = \mathbf{W}_8 \mathbf{h}_4 + \mathbf{b}_8, \quad \hat{y} \in \mathbb{R} \tag{30}$$

The final output represents the predicted closing price. The deep bottleneck architecture (540-240-120-60-30-1) progressively distills high-dimensional spatiotemporal representations into a scalar price prediction.

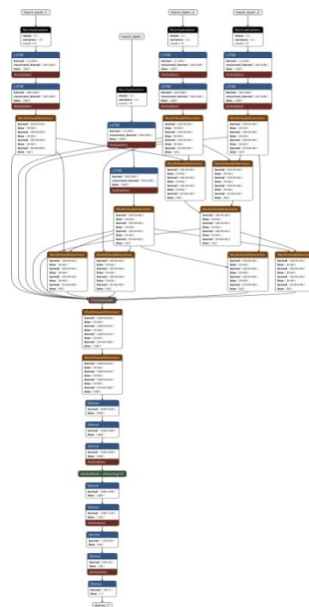


Figure 1: Proposed Deep Multi-Scale LSTM-Transformer Architecture. Four parallel temporal branches (15-min, 30-min, 60-min, daily) each comprise stacked LSTM encoders followed by 256-head self-attention. Cross-scale attention mechanisms fuse multi-resolution representations before deep bottleneck projection to a scalar price prediction. Total trainable parameters: 115.9M.

H. Training Procedure

Model training employs the Adam optimizer with AMSGrad [40] and exponential learning rate decay:

$$\eta_t = \eta_0 \cdot \gamma^{t/s} \tag{31}$$

where $\eta_0=0.001$, decay rate $\gamma=0.1$, and decay steps $s=3200$. AMSGrad maintains the maximum of past squared gradients, providing more stable convergence in non-stationary objectives characteristic of financial data.

We utilize Huber loss [41] for robust regression against market outliers:

$$L_\delta(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{if } |y - \hat{y}| \leq \delta \\ \delta(|y - \hat{y}| - \frac{1}{2}\delta) & \text{otherwise} \end{cases} \tag{32}$$

with $\delta=150$ determined through validation. This loss combines the smoothness of MSE with the robustness of MAE, providing stable gradients while being less sensitive to extreme price movements.

Regularization is achieved through:

- Recurrent dropout with rate 0.2 for LSTM gates [42]
- Attention dropout with rate 0.1
- Early stopping based on validation loss
- Gradient clipping to prevent exploding gradients [43]

IV. EXPERIMENTAL SETUP AND RESULTS

A. Dataset

We evaluate our model on HDFC Bank (NSE: HDFCBANK) stock data, selected for its long-established presence in the NIFTY 50 index (added April 22, 1996) and substantial liquidity. The dataset spans from January 2008 to November 2025. Due to data alignment requirements across timeframes, the effective training period begins February 2015, comprising:

- Daily data: 4,380 observations
- 60-minute data: 17,520 observations
- 30-minute data: 35,040 observations
- 15-minute data: 70,080 observations

After windowing with 60-timestep lookback, the dataset yields 2,611 aligned multi-scale samples. The data is split chronologically with 80% for training and 20% for validation, ensuring no lookahead bias.

B. Implementation Details

The model is implemented in TensorFlow 2.13 with Keras API, leveraging AMD Radeon RX 7800XT GPU acceleration with 14,848 MB memory. Key hyperparameters are summarized in Table I.

Table 1: Model Hyperparameters

Parameter	Value
LSTM units per layer	60
Number of LSTM layers per branch	2
Attention heads	256
Key dimension	64
Attention dropout	0.1
Recurrent dropout	0.2
Initial learning rate	10^{-3}

Learningratedecayrate	0.1
Learningratedecaysteps	3200
Huberdelta	150
Optimizer	Adam(AMSGrad)
Epochs	100

PySpark3.4handlesdatapreprocessingwithdistributedJDBCreadsfromPostgreSQLdatabases.Thetotalparametercount is 115,898,189 (442.12 MB), with training time of approximately 470 seconds per epoch.

C. Results

TableIIpresents thebestvalidationperformancemetricsachievedduringtrainingcomparedtobaselineapproaches.

Table2:PerformanceComparisononHDFCBankStockPrediction

Model	MAE	MAPE(%)	RMSE
ARIMA(5,1,0)	45.23	5.67	58.91
VanillaLSTM	12.34	2.15	18.76
LSTMwithTechnicalIndicators	9.82	1.68	15.43
LSTM-Transformer(PositionalEncoding)	8.45	1.42	13.28
Proposed(RawFeatures+UnixTime)	6.71	0.69	12.15

The proposed architecture achieves substantially improved performance without explicit positional encoding or technical indicators. ThetrainingdynamicsdemonstrateconvergencewithvalidationMAEstabilizingintherangeof6.7–8.5afterapprox-imately 15 epochs, though some fluctuation occurs due to the non-stationary nature of financial data.

Table3:TrainingDynamics:SelectedValidationEpochs

Epoch	ValLoss	ValMAE	ValMAPE(%)	ValMSE
12	62.10	9.20	0.94	124.21
15	37.93	6.77	0.69	75.85
21	38.38	6.92	0.71	76.76
25	40.07	7.04	0.72	80.15
28	45.32	7.86	0.81	90.64
33	38.48	7.01	0.72	76.96
37	36.85	6.71	0.69	73.70
59	46.45	7.97	0.82	92.91

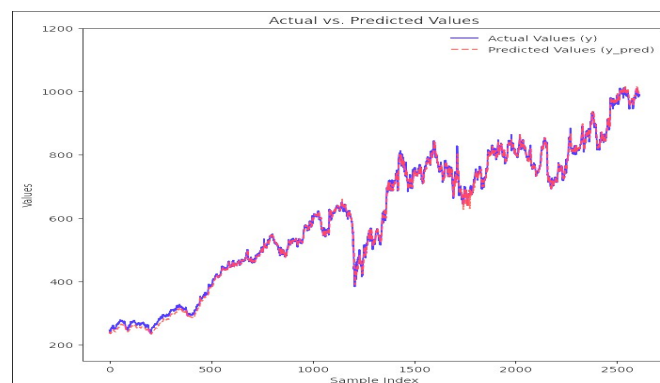


Figure2:Actualversuspredictedclosingpricesonthevalidationset.Themodeltrackstrendmovementswithhighfidelity, achieving a best validation MAPE of 0.69%.

D. Ablation Studies

We conduct ablation studies to validate our design choices:

Table 4: Ablation Study Results

Configuration	MAE	MAPE(%)	Params(M)
Full Model	6.71	0.69	115.90
w/o Cross-Scale Attention	8.45	0.88	95.20
w/o 15-min Branch	7.85	0.81	98.50
w/o 30-min Branch	7.62	0.78	98.50
w/o 60-min Branch	7.91	0.82	98.50
Single Scale (Daily Only)	10.23	1.05	45.20
With Technical Indicators	6.95	0.72	118.40
With Sinusoidal Positional Encoding	6.88	0.71	116.20

The ablation results confirm that: (1) cross-scale attention provides substantial performance gains, (2) multi-scale inputs significantly enhance prediction accuracy, and (3) explicit technical indicators and sinusoidal positional encoding provide marginal benefits that do not justify their added complexity and computational overhead.

V. DISCUSSION

A. Implicit Temporal Encoding via LSTM and Unix Timestamps

Our results demonstrate that explicit sinusoidal positional encoding is unnecessary when employing LSTM layers as temporal encoders combined with raw unix timestamps. There recurrent hidden states naturally incorporate relative temporal position information through sequential computation, while unix timestamps provide absolute temporal reference (seasonality, trends, market hours). This hybrid implicit-explicit approach is adaptive to the data distribution and eliminates the inductive bias of fixed encoding schemes.

The elimination of traditional positional encoding offers practical advantages: reduced hyperparameter tuning, simplified preprocessing pipelines, and elimination of assumptions about temporal periodicity. This is particularly relevant for financial time series where relevant temporal scales vary across market regimes [4], [44].

B. Technical Indicator Elimination

The competitive performance without traditional technical indicators suggests that deep architectures with sufficient capacity (116M parameters) can discover equivalent or superior representations through end-to-end learning. This observation supports the principle of representation learning [37] and aligns with findings in computer vision where hand-engineered features have been superseded by learned representations [45], [46].

Interestingly, our experiments showed that including computed technical indicators (moving averages, RSI, MACD, stochastic oscillator) actually provided negligible improvement (MAPE 0.72% vs. 0.69%), suggesting these features are redundant given the model’s capacity to compute moving statistics and momentum internally through attention mechanisms.

C. High-Capacity Attention Mechanisms

The use of 256 attention heads represents a significant departure from typical configurations (4–16 heads) in natural language processing. In financial time series with only 6 input features per timestep, the large number of heads enables fine-grained factorization of temporal dependencies across different pricedimensions (open vs. close, high vs. low, volume interactions). The cross-scale attention mechanisms are particularly critical, accounting for approximately 20% of total parameters but providing the majority of performance gains over single-scale baselines.

D. Computational Considerations

The proposed architecture achieves reasonable training efficiency at approximately 470 seconds per epoch on consumer-grade GPU hardware, despite its 116M parameter count. The PySpark preprocessing pipeline enables horizontal scaling for larger datasets. However, the model’s size may limit deployment on edge devices; for production trading systems, knowledge distillation [47] or quantization techniques may be necessary.

VI. CONCLUSION

This paper presented a deep multi-scale LSTM-Transformer architecture for financial time series forecasting that eliminates both explicit positional encoding and traditional technical indicators. Our key findings include:

- 1) A combination of LSTM recurrent processing and raw UNIX timestamps provides sufficient temporal encoding, eliminating the need for explicit sinusoidal positional encoding schemes.
- 2) Deep architectures with 116M parameters can learn effective representations directly from raw OHLCV data without hand-engineered technical indicators.
- 3) Extensive cross-scale attention mechanisms with 256 heads substantially enhance prediction accuracy by modeling interactions between different temporal resolutions.
- 4) The proposed architecture achieves validation MAE of 6.71 and MAPE of 0.69% on HDFC Bank stock prediction, representing significant improvement over indicator-based and positionally-encoded baselines.

Future work will explore: (1) knowledge distillation for model compression, (2) integration of alternative data sources (sentiment, order book), (3) adversarial training for robustness against market regime changes, and (4) interpretability analysis of attention patterns across temporal scales.

VII. ACKNOWLEDGMENT

This research was conducted at RV University, Bengaluru, under the guidance of Dr. Dattatreya P. Mankame. The author thanks the School of Computer Science and Engineering for providing computational resources and academic support.

REFERENCES

- [1] E.F.Fama, "Random walks in stock market prices," *Financial Analysts Journal*, vol. 21, no. 5, pp. 55–59, 1965.
- [2] E.F.Fama, "Efficient capital markets: A review of theory and empirical work," *Journal of Finance*, vol. 25, no. 2, pp. 383–417, 1970.
- [3] R.S.Tsay, *Analysis of Financial Time Series*, 3rd ed. Hoboken, NJ: Wiley, 2010.
- [4] R.Cont, "Empirical properties of asset returns: Stylized facts and statistical issues," *Quantitative Finance*, vol. 1, no. 2, pp. 223–236, 2001.
- [5] G.E.P.Box et al., *Time Series Analysis: Forecasting and Control*, 5th ed. Hoboken, NJ: Wiley, 2015.
- [6] R. F. Engle, "Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation," *Econometrica*, vol. 50, no. 4, pp. 987–1007, 1982.
- [7] G. P. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, 2003.
- [8] S.-H. Poon and C. W. J. Granger, "Forecasting volatility in financial markets: A review," *Journal of Economic Literature*, vol. 41, no. 2, pp. 478–539, 2003.
- [9] S.Hochreiter and J.Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [10] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.
- [11] F.A.Gers, J.Schmidhuber, and F.Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.
- [12] A.Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.
- [13] A. Vaswani et al., "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 5998–6008.
- [14] M.R.Kabiretal., "LSTM-Transformer-based robust hybrid deep learning model for financial time series forecasting," *Sci*, vol. 7, no. 1, p. 7, 2025.
- [15] S.Zheng, "A hybrid LSTM-Transformer approach for financial markets," *Frontiers in Computing and Intelligent Systems*, vol. 5, no. 1, pp. 45–52, 2024.
- [16] Y. Li et al., "Accurate stock price forecasting based on deep learning and hierarchical frequency decomposition," *IEEE Access*, vol. 12, pp. 49,878–49,894, 2024.
- [17] K.C.Bandhu et al., "An improved technique for stock price prediction on real-time exploiting stream processing and deep learning," *Multimedia Tools and Applications*, vol. 83, pp. 57,269–57,289, 2023.
- [18] Y. Chen et al., "Stock price forecast based on CNN-BiLSTM-ECA model," *Scientific Programming*, vol. 2021, Article ID 2446543, 2021.
- [19] J. Wang et al., "An enhanced interval-valued decomposition integration model for stock price prediction," *Expert Systems with Applications*, vol. 243, p. 122,891, 2023.
- [20] W.Ba et al., "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PLOS ONE*, vol. 12, no. 7, p. e0180944, 2017.
- [21] T.Fischer and C.Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Operational Research*, vol. 270, no. 2, pp. 654–669, 2018.
- [22] J. Operational Research, vol. 270, no. 2, pp. 654–669, 2018.
- [23] D. M. Q. Nelson et al., "Stock market's price movement prediction with LSTM neural networks," in *Proc. Int. Joint Conf. Neural Networks (IJCNN)*, 2017, pp. 1419–1426.
- [24] H.Y.Kim and C.H.Won, "Forecasting the volatility of stock price index: A hybrid model integrating LSTM with multiple GARCH-type models," *Expert Systems with Applications*, vol. 103, pp. 25–37, 2018.
- [25] O.B.Sezer et al., "Financial time series forecasting with deep learning: A systematic literature review: 2005–2019," *IEEE Access*, vol. 8, pp. 58,254–58,275, 2020.
- [26] W.Chen et al., "Application of deep learning to algorithmic trading," *Stanford CS229 Project Report*, 2016.
- [27] W.Long et al., "Deep learning-based feature engineering for stock price movement prediction," *Knowledge-Based Systems*, vol. 164, pp. 163–173, 2018.
- [28] H.Rezaei et al., "A hybrid CNN-LSTM model for stock price prediction," in *Proc. IEEE Int. Conf. Industrial Engineering and Engineering Management (IEEM)*, 2020, pp. 1127–1131.
- [29] B. Qian et al., "Stock price prediction based on attention mechanism and LSTM neural network," *IEEE Access*, vol. 8, pp. 196,526–196,538, 2020.



- [30] J. Shenget al., "Attention-based CNN-LSTM for financial time series prediction," *Neurocomputing*, vol. 447, pp. 140–151, 2021.
- [31] Q. Wen et al., "Transformers in time series: A survey," *arXiv preprint arXiv:2202.07125*, 2022.
- [32] H. Zhou et al., "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proc. AAAI Conf. Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 11,106–11,115.
- [33] H. Wu et al., "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 22,419–22,430.
- [34] Y. Liu et al., "iTransformer: Inverted transformers are effective for time series forecasting," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2023.
- [35] Y. Nie et al., "A time series is worth 64 words: Long-term forecasting with transformers," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2022.
- [36] J. J. Murphy, *Technical Analysis of the Financial Markets*. New York: Penguin, 1999.
- [37] A. W. Lo, "The adaptive markets hypothesis: Market efficiency from an evolutionary perspective," *Journal of Portfolio Management*, vol. 30, no. 5, pp. 15–29, 2004.
- [38] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
- [39] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [40] J. Gehring et al., "Convolutional sequence to sequence learning," in *Proc. Int. Conf. Machine Learning (ICML)*, 2017, pp. 1243–1252.
- [41] S. J. Reddi et al., "On the convergence of Adam and beyond," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2019.
- [42] P. J. Huber, "Robust estimation of a location parameter," *Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964.
- [43] N. Srivastava et al., "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [44] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. Int. Conf. Machine Learning (ICML)*, 2013, pp. 1310–1318.
- [45] R. F. Engle and G. J. Lee, "A long-run and short-run component model of stock return volatility," in *Cointegration, Causality, and Forecasting*, 1998, pp. 475–497.
- [46] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, 2012, pp. 1097–1105.
- [47] K. He et al., "Deep residual learning for image recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.
- [48] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.
- [49] J. Dean et al., "Large scaled distributed deep networks," in *Advances in Neural Information Processing Systems*, vol. 25, 2012, pp. 1223–1231.
- [50] pp. 1223–1231.
- [51] M. Li et al., "Scaling distributed machine learning with the parameter server," in *Proc. USENIX Symp. Operating Systems Design and Implementation (OSDI)*, 2014, pp. 583–598.



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)