International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Enhancing Financial Market Prediction: A Comparative Analysis of RNN and LSTM Models for Cryptocurrency Price Forecasting

Sadanand Sundaray

Email: ssray2025@gmail.com

Abstract

Advances in deep learning models such as Recur-rent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks have shown promise in forecasting financial markets, yet their application to cryptocurrency remains un-derexplored. A detailed evaluation of both RNN and LSTM models is conducted, with a focus on Ethereum price prediction using ten years of historical data. The study compares model performance in capturing the intricate, non-linear dynamics of highly volatile cryptocurrency markets. Results indicate that LSTMs outperform RNNs in learning long-term dependencies, but both models face limitations in noisy, non-stationary market conditions. A hybrid approach integrating traditional technical indicators—Moving Averages, Relative Strength Index (RSI), and Bollinger Bands—demonstrates improved prediction accuracy. The research highlights the potential for real-time learning and dynamic feature selection in enhancing model adaptability to sudden market changes, offering new insights into the capabilities and challenges of applying deep learning to cryptocurrency trading.

Index Terms: Recurrent Neural Networks, Long SHort-Term MEmory, Cryptocurrency, Crypto mArket

1. Introduction

It has been observed that virtual currencies are recognized as a groundbreaking phenomenon in the realm of global finance, with traditional notions of money and the exchange of value being reshaped by them. Among the myriad of cryptocurrencies that have proliferated over the past decade, Ethereum (ETH) is distinguished as a pioneering platform that is regarded to extend beyond the realm of mere digital currency [1], [2]. The inception of Ethereum in 2015 was marked as a significant milestone in the world of blockchain technology, with the revolutionary concept of smart contracts being introduced and a new era of decentralized applications (dApps) being enabled. In order for the evolution of Ethereum and its profound impact on the financial landscape to be comprehended, a comprehensive history of virtual currencies must be examined.

The journey of virtual currencies was initiated with the launch of Bitcoin in 2009 by the enigmatic pseudonymous figure known as Satoshi Nakamoto. Bitcoin, often hailed as digital gold, was credited with the introduction of the concept of a decentralized, peer-to-peer digital currency operating on a blockchain—a distributed ledger technology. The primary function of Bitcoin was defined to be that of a

medium of exchange and a store of value, with transactions being enabled without reliance on intermediaries such as banks or payment processors. The monumental success of Bitcoin led to the creation of numerous alternative cryptocurrencies, collectively known as altcoins, with efforts being made to refine Bitcoin's design or to offer distinctive features.

While the foundation for the cryptocurrency revolution was laid by Bitcoin, the realm of possibilities was expanded by Ethereum through the introduction of smart contracts. Launched in July 2015 by the visionary young programmer Vitalik Buterin, Ethereum was conceived as a blockchain platform capable of facilitating not only digital currency trans-actions but also the execution of self-executing, programmable contracts. Predefined actions are automatically executed by smart contracts when specific conditions are met, thereby obviating the necessity for intermediaries in a multitude of contractual agreements. By this groundbreaking innovation, the doors were flung open to a wide array of decentralized applications capable of autonomous and transparent operation, and the concepts of decentralized finance (DeFi), non-fungible tokens (NFTs), and more were given rise to.

Conventional machine learning techniques were used as the foundation for traditional quantitative trading strategies [3]. In these methods, the utilization of algorithms was involved in the scrutiny of historical market data, the recognition of patterns, and the formulation of predictions. Trading decisions were commonly informed by features such as moving averages, rel-ative strength indices, and simple regression models. Although some level of success was achieved by these approaches, limitations were encountered in the capturing of the intricate and dynamic relationships inherent in financial time series data.

A transformative shift in the field of quantitative trading was marked by the advent of deep learning. The potential to model intricate patterns and dependencies within financial data was heralded by deep learning techniques, particularly neural networks. Layers of interconnected neurons, in which information is processed and transmitted to the subsequent layer, are what neural networks are comprised of. Nonlinear relationships were effectively captured by this architectural design, which was regarded as a critical advantage in financial markets where intricacies and nonlinearity are often exhibited by patterns [3].

Within the domain of deep learning, recurrent neural networks (RNNs) and their variant, long short-term memory (LSTM) networks, have come to be recognized as indispens-able tools for time series analysis [4]. Data sequences are specifically handled by RNNs, rendering them particularly well-suited for the analysis of time series data. The unique ability of LSTMs to capture long-range dependencies within sequences sets them apart, as a challenge to be grappled with by traditional RNNs due to the vanishing gradient problem. A memory cell is employed by LSTMs to retain and update information over extended time intervals, thereby empowering the recall and leveraging of past information when predictions about future data points are made. In the context of quantitative trading, the learning of complex temporal patterns is rendered invaluable by LSTMs, thereby enhancing their utility in the prediction of price movements, the identification of trading signals, and the management of risk.

Significant differences are exhibited by the virtual currency market when compared to traditional stock markets. Operation around the clock is exhibited by the virtual currency mar-ket, without restrictions by geographical locations or market exchanges, thereby leading to greater volatility and price fluctuations. Additionally, the relative youth of the virtual currency market and the influence of different factors, such as social media, news events, and regulatory changes, serve to set it apart from traditional stock markets. Substantial fluctuations in liquidity are experienced in the virtual currency market, with susceptibility to manipulation being exhibited by some smaller-cap cryptocurrencies. Additional challenges in the execution of trading strategies and the management of risks are presented by these characteristics. In comparison to tradi-tional stock markets, greater non-stationarity in terms of price and trading volume is exhibited by virtual currency markets, implying that more trends, seasonality, and periodicity may be displayed by time-series data, and additional preprocessing steps are necessitated to address these non-stationarities. A high degree of price volatility in the virtual currency market is observed, often accompanied by noise, and misleading signals in the short term may be produced by models, thus necessitating the use of effective filtering and noise handling strategies.

The capabilities of LSTMs to analyze historical market data and to make well-informed trading decisions can be harnessed by their integration into the quantitative trading framework. A competitive edge in the dynamic and rapidly evolving landscape of virtual currency trading can be conferred by their aptitude for modeling intricate time series patterns. In this project, the efficacy of LSTMs in the prediction of Ethereum's price is to be explored, with Ethereum being recognized as a leading cryptocurrency that has witnessed a meteoric rise in popularity and adoption over the past few years. Series of RNN and LSTM models will be trained on historical Ethereum price data, and their performance in predicting future price movements will be evaluated. The impact of hyperparameter tuning on the models' performance will also be explored, and the efficacy of the models in generating trading signals will be examined.

The intricacies of the application of advanced time-series models to the domain of financial market prediction are delved into in this paper. Following the introduction, exploration is initiated in Section ??, where the methods employed are outlined, starting with Feature Engineering. A detailed exam-ination of various technical indicators, such as the Relative Strength Index (RSI), Bollinger Bands, Moving Averages, Moving Average Convergence Divergence (MACD), Aver-age True Range (ATR), and On-Balance Volume (OBV), is provided in this section. Subsequently, the core Time Series Models used in the study, specifically focusing on Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) Networks, are discussed. The Experimen-tation process is dedicated to in Section IV, beginning with Data Preprocessing, which includes Data Acquisition, Data Augmentation, and Price Normalization techniques. Further elaboration on the Experimentation Environment is provided, with the computational setup being detailed, and a thorough discourse on Experimental Parameters and Model Hyperpa-rameter Selection being presented. Results are showcased in Section V, where the challenges posed by extended time-series in training are highlighted and the limitations of RNN and LSTM models in this context are explored. Finally, a comprehensive Summary and Conclusion of the findings is provided in Section VI. In this section, insights gained from the experiments and discussions are synthesized, a critical perspective on the efficacy of deep learning models in financial time series forecasting is offered, and potential avenues for future research are suggested.

2. Method

A. Feature Engineering

In the context of time series forecasting for quantitative trading—particularly within the volatile cryptocurrency mar-kets—it has been determined that relying solely on raw price data is insufficient. The inherent noise and rapid fluctuations in price render such data prone to overfitting and limit the generalization capability of predictive models. Therefore, a refined approach to feature engineering is adopted, whereby raw prices are transformed using classical technical indicators. In doing so, a more robust and informative dataset is created for advanced AI-based trading strategies. Specifically, traditional financial analysis tools such as the Relative Strength Index (RSI), Bollinger Bands, Moving Averages, Moving Average Convergence Divergence (MACD), Average True Range (ATR), and On-Balance Volume (OBV) are integrated to capture trends, momentum, and volatility in the market.

1) Relative Strength Index (RSI): The Relative Strength Index (RSI) is a momentum oscillator that quantifies the speed and magnitude of recent price movements. Originally proposed by J. Welles Wilder, the RSI is used to detect overbought or oversold conditions in a market by oscillating between 0 and 100 [5], [6]. It is computed using the average gains and losses over a specified period (commonly 14 periods), according to the formula:

$$RSI = 100 - \frac{100}{1 + RS},$$

where RS is the ratio of the average upward price changes to the average downward price changes. This transformation enables a more nuanced interpretation of market momentum compared to raw price data.

2) Bollinger Bands: Bollinger Bands, introduced by John Bollinger, serve as a dynamic envelope to assess price volatil-ity and potential reversal points in financial markets [5], [7]. They consist of three distinct components:

- Middle Band (MB): Typically calculated as the Simple Moving Average (SMA) over a predetermined period (commonly 20 periods).
- Upper Band (UB): Defined as the Middle Band plus twice the standard deviation (SD) of prices over the same period.

- Lower Band (LB): Defined as the Middle Band minus twice the standard deviation.

These bands are mathematically expressed as:

$$MB = SMA(price, n),$$

$$UB = MB + 2 \times SD(price, n),$$

$$LB = MB - 2 \times SD(price, n),$$

where n denotes the number of periods over which the calculations are performed. This indicator not only highlights volatility but also provides context for potential market ex-tremes.

3) Moving Averages: Moving Averages are essential for smoothing out short-term fluctuations and revealing the un-derlying trend in price data [5]. Two widely used forms are:

1) Simple Moving Average (SMA): This is computed by averaging the prices over a specified number of periods, assigning equal weight to each observation:

$$SMA = \frac{1}{n} \sum_{i=1}^{n} Price_i.$$

2) Exponential Moving Average (EMA): By contrast, the EMA assigns greater weight to more recent observa-tions, thereby making it more responsive to new data. It is calculated as:

$$EMA_t = \alpha \cdot Price_t + (1 - \alpha) \cdot EMA_{t-1},$$

where α represents the smoothing factor.

These averages are fundamental in identifying trends and possible reversal points within the market.

4) Moving Average Convergence Divergence (MACD): The Moving Average Convergence Divergence (MACD) is a momentum indicator used to detect changes in trend and market momentum [8]. It is composed of three elements:

1) MACD Line: The difference between the 12-period EMA and the 26-period EMA.

2) Signal Line: A 9-period EMA of the MACD Line.

3) MACD Histogram: The difference between the MACD Line and the Signal Line.

The indicator is computed as follows:

- MACD Line:

$$MACD = EMA_{12}(Price) - EMA_{26}(Price),$$

- Signal Line:

$$Signal = EMA_9(MACD),$$

- MACD Histogram:

$$Histogram = MACD - Signal.$$

By comparing these components, traders are able to glean insights into shifts in momentum and the potential onset of new trends.

5) Average True Range (ATR): The Average True Range (ATR) is a volatility indicator developed by J. Welles Wilder that measures the degree of price variability by considering the complete trading range over a specified period [9]. The ATR is defined as:

$$\mathrm{ATR} = \frac{1}{n}\sum_{i=1}^{n} \max\left(H_i - L_i, |H_i - C_{i-1}|, |L_i - C_{i-1}|\right),$$

where H_i and L_i are the high and low prices at time i, and C_{i-1} is the previous closing price. This measure aids in the determination of appropriate risk management levels, such as stop-loss orders, by quantifying market volatility.

6) Recurrent Neural Network (RNN) Pseudocode: The fol-lowing pseudocode outlines the operation of a basic Recurrent Neural Network (RNN), which is employed to process sequen-tial data:

7) On-Balance Volume (OBV): The On-Balance Volume (OBV) indicator, initially proposed by Joseph Granville, fo-cuses on the relationship between trading volume and price changes to evaluate market sentiment. OBV is defined by the formula:

$$\mathrm{OBV}_t = \mathrm{OBV}_{t-1} + \mathrm{Volume}_t \cdot \mathrm{Sign}(\mathrm{Close}_t - \mathrm{Close}_{t-1}),$$

where OBV_t is the current On-Balance Volume, Volume_t represents the trading volume at time t, and the function Sign(x) returns 1 if $x > 0$, -1 if $x < 0$, and 0 when $x = 0$. This indicator assists in determining whether the market is experiencing an accumulation or distribution phase, thereby providing valuable insights into the prevailing market trend.

B. Time Series Models

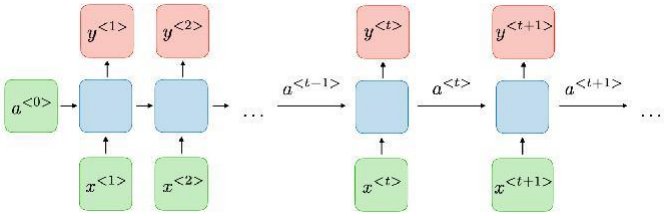

Fig. 1. Plot of Architecture of a traditional RNNRecurrent neural networks, also known as RNNs, a class of neural networks that allow previous outputs to be used as inputs while having hidden states.

Algorithm 1 Pseudocode for a Recurrent Neural Network (RNN)

```
procedure RNN(Input Sequence)
Initialize the hidden state h
Set weight matrices W_hx, W_hh, W_hy and bias vectors b_h, b_y
for each time step t with input x_t do
    h_t ← tanh(W_hx · x_t + W_hh · h_{t−1} + b_h)        ▷ Update hidden state
    y_t ← softmax(W_hy · h_t + b_y)                      ▷ Generate output
 Append y_t to the output sequence
 end for
 return the output sequence
 end procedure
```

1) Recurrent Neural Networks (RNNs): Recurrent Neural Networks (RNNs) are a class of neural networks specifically designed to handle sequential data. At the core of RNN architecture lies the unique feature of maintaining a form of memory that captures information about previous elements in the sequence. Unlike traditional neural networks, which assume independence between inputs, RNNs can process input data in a sequential manner, retaining state information from one step of the sequence to the next. This is achieved through loops within the network architecture, allowing information to persist.

One of the key strengths of RNNs is their ability to manage varying input lengths. In time series forecasting, such as cryptocurrency trading, where the input sequences can be of different durations, RNNs adaptively process this variable-length data, making them highly suitable for this domain [10]. Furthermore, RNNs can process not only single data points (such as a day's closing price) but also sequences of data (such as prices over the past week), which is essential in understanding the temporal dynamics of financial markets.

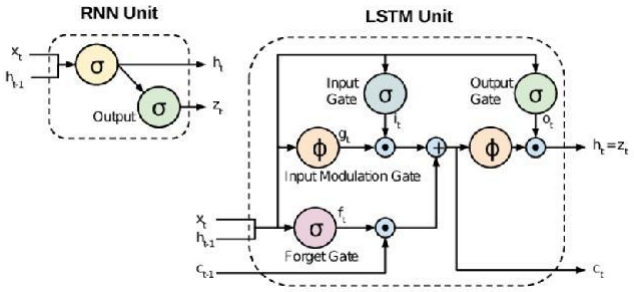

Fig. 2. Plot of how the information in time step t − 1 influence time step t in a traditional RNN model.

However, traditional RNNs face challenges such as the vanishing gradient problem, which hinders the learning of long-range dependencies within the sequence data. This is particularly problematic in financial time series, where long-term trends and cycles are critical. To address this, variants of RNNs, like Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs), have been developed [4]. These architectures incorporate mechanisms to control and maintain a balance between

short-term input and long-term historical information, thus effectively capturing dependencies across different time scales.

2) Long Short-Term Memory (LSTM) Networks: Long Short-Term Memory (LSTM) networks, a significant advance-ment in the field of recurrent neural networks (RNNs), were introduced by Sepp Hochreiter and Jurgen¨ Schmidhuber in 1997 [4]. The primary motivation behind the development of LSTMs was to address the limitations of traditional RNNs, particularly their inability to learn long-term dependencies in sequence data. This inability, known as the vanishing gradient problem, arises because the gradients, which are used to update the network's weights, tend to either vanish or explode as they are propagated back through time in a deep network.

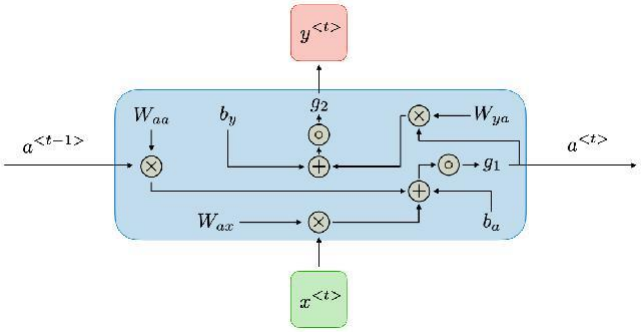

Fig. 3. Plot of a demo of the differences between RNN and lstm in handling input.

3. Methodology

A. LSTM Architecture and Operation

Long Short-Term Memory Networks (LSTMs) have been devised to resolve the vanishing gradient challenge by inte-grating a sophisticated system of gates. In this framework, three distinct gates are implemented:

- Input Gate: Regulates the amount of new information to be incorporated into the cell state.
- Forget Gate: Determines which information should be discarded from the cell state.
- Output Gate: Controls which parts of the cell state are emitted as the output at each time step.

The cell state itself functions as a persistent memory channel that carries essential information through the entire sequence. In effect, the input gate modulates the introduction of novel data, the forget gate eliminates redundant or irrelevant details, and the output gate selects the pertinent information required to produce the network's output.

This architecture stands in marked contrast to conventional Recurrent Neural Networks (RNNs) and even to Gated Recur-rent Units (GRUs). Although GRUs also address the vanishing gradient issue, they achieve this by merging the input and forget gates into a unified update gate and combining the cell state with the hidden state. LSTMs, however, deliberately maintain separate cell and hidden states. This

separation enables a more granular control over the information flow, allowing the model to capture longer and more complex temporal dependencies—albeit with a higher computational cost.

4. Experimentation

A. Data Preprocessing

1) Data Acquisition: Automated trading in the cryptocur-rency domain is facilitated through robust interfaces such as the Binance API. This API enables programmatic access to real-time and historical market data, execution of trades, and account management. In the implementation, the Binance Spot API is utilized to retrieve K-line (candlestick) data for cryptocurrencies like Ethereum (ETH). Each candlestick encapsulates essential market information over a specified interval by providing the open, high, low, and close prices, as well as trading volume.

Functions (e.g., get_klines_data) are employed to systematically extract this data across various time scales (e.g., 1 minute, 1 hour), thereby supplying a flexible foundation for feature engineering, pattern analysis, and subsequent predic-tive modeling. The comprehensive dataset thus obtained is cru-cial for enabling data-driven decision making in quantitative trading.

2) Data Augmentation and Price Normalization: In quan-titative trading, data augmentation is employed to enhance the raw dataset by incorporating additional derived features, thereby capturing a broader spectrum of market dynamics. This process, synonymous with feature engineering, is exe-cuted through several steps:

1) Calculation of Technical Indicators:

- SMA and EMA: These moving averages smooth price fluctuations over designated periods to reveal underlying trends.
- RSI: This momentum oscillator quantifies the mag-nitude of recent price changes, assisting in the identification of overbought or oversold conditions.
- Bollinger Bands: By combining a moving average with a volatility band (standard deviation), these bands offer insights into price extremes.

- ATR: The Average True Range dissects the full range of price movement, providing a measure of market volatility.

- OBV: On-Balance Volume leverages trading volume data to anticipate price shifts.

2) Normalization of Price Data: Price data (e.g., Open, High, Low, Close) is normalized using percentage change, transforming absolute prices into rates of change. This approach reduces sensitivity to absolute price scales and emphasizes relative movements.

3) Standardization of Additional Features: Non-price features are standardized via z-score normalization (sub-tracting the mean and dividing by the standard deviation) to ensure each feature contributes proportionally during model training.

4) Treatment of Missing Values: Any missing values are initially forward-filled—propagating the last valid observation—and any remaining NaNs are subsequently removed to maintain data continuity and integrity.

Collectively, these procedures convert raw market data into a feature-rich and normalized dataset, thereby enhancing the performance and convergence of machine learning models used in quantitative trading.

B. Computational Environment

Experiments are conducted on a high-performance comput-ing system configured with the following specifications:

- RAM: 256 GB
- GPU: NVIDIA GeForce RTX 3090
- CPU: Intel Xeon CPU E5-2696 v3
- Operating System: Ubuntu 22.04 LTS (kernel 6.2.0-37-generic)
- Programming Language: Python 3.11.5
- Deep Learning Framework: PyTorch 2.1.0

This robust setup delivers the requisite computational power and efficiency for processing extensive datasets and running complex deep learning models. The stability and versatility of Ubuntu, combined with the ample 256 GB of RAM, ensure that large-scale data analysis and iterative model training are executed seamlessly. The synergy of Python 3.11.5 and PyTorch 2.1.0 further facilitates the rapid prototyping and fine-tuning of sophisticated financial models.

C. Performance Evaluation

The following table summarizes the performance metrics obtained from models of varying complexities over a pro-longed time series dataset (from January 1, 2017, to December 31, 2022, at a daily resolution). The results indicate that both RNN and LSTM models encountered difficulties in learning meaningful patterns from such an extended dataset.

D. Experimental Parameters and Model Hyperparameter Se-lection

A comprehensive experimental framework has been devised to encompass a wide range of parameters and hyperparameters that are deemed essential for evaluating the effectiveness of various trading strategies under diverse conditions. At the core of the investigation, a diverse array of model architectures has been employed, each tailored to satisfy the unique require-ments of financial time-series data.

Algorithm 2 Pseudocode for the LSTM Module

procedure LSTM(Input Sequence)
Initialize cell state c and hidden state h
Initialize weight matrices: W_{xi}, W_{xf} , W_{xo}, W_{xc}, W_{hi}, W_{hf} , W_{ho}, W_{hc}
Initialize bias vectors: b_i, b_f , b_o, b_c
for each element x_t in the Input Sequence do
Compute input gate: $i_t \leftarrow \sigma(W_{xi} \cdot x_t + W_{hi} \cdot h_{t-1} + b_i)$
Compute forget gate: $f_t \leftarrow \sigma(W_{xf} \cdot x_t + W_{hf} \cdot h_{t-1} + b_f)$
Compute output gate: $o_t \leftarrow \sigma(W_{xo} \cdot x_t + W_{ho} \cdot h_{t-1} + b_o)$
Compute candidate cell state: $\tilde{c}_t \leftarrow \tanh(W_{xc} \cdot x_t + W_{hc} \cdot h_{t-1} + b_c)$
Update cell state: $c_t \leftarrow f_t * c_{t-1} + i_t * \tilde{c}_t$
Update hidden state: $h_t \leftarrow o_t * \tanh(c_t)$
Compute output: $y_t \leftarrow \text{softmax}(W_{hy} \cdot h_t + b_y)$
Append y_t to the output sequence
end for
return the output sequence
end procedure

Multiple configurations of convolutional and recurrent neu-ral network layers have been explored in order to assess their impact on predictive capabilities. Configurations have ranged from simpler setups, such as two-layer convolutional networks, to more elaborate architectures comprising up to five layers. In parallel, various recurrent neural network configurations have been examined by adjusting both the number of layers and the sizes of hidden units. This diversity in model architecture has been found to facilitate a balanced trade-off between computational efficiency and the depth of learning necessary for accurate market predictions.

The training process has been enhanced through the in-corporation of advanced data handling and optimization tech-niques. Both a standard Trainer class and a more sophisticated GroupedTrainer have been implemented; the latter divides the dataset into manageable groups to allow for more efficient training on the large datasets typically encountered in financial applications. Loss functions have been selected according to the nature of the problem—binary cross-entropy for two-class issues, cross-entropy for multi-class challenges, and mean squared error for regression tasks—to ensure alignment with the specific objectives of each model. The Adam optimizer has been utilized because of its adaptability and effectiveness, par-ticularly with sparse gradients, while a step-based learning rate

scheduler has been adopted to periodically reduce the learning rate, thus fine-tuning the models as training progresses. This methodical approach has been recognized as instrumental in balancing rapid convergence with the avoidance of local minima.

Performance evaluation has been conducted using a com-bination of traditional machine learning metrics and financial performance indicators. Metrics such as the Sharpe ratio and annualized returns have been included to provide a clear pic-ture of the risk-adjusted performance of the trading strategies. The training regimen was extended over 7200 epochs, with evaluations conducted at regular intervals to ensure that the learning capacity of each model configuration was thoroughly explored.

Flexibility has been built into the framework by providing functions capable of processing both numpy arrays and pandas DataFrames. This flexibility has been regarded as crucial for the seamless integration of varied data sources common in financial modeling. Additionally, models have been deployed on the most appropriate hardware available, whether CPU or GPU, in order to optimize resource utilization and to ensure efficient model training and evaluation.

5. Results

A. Challenges Posed by Extended Time-Series in Training

The experimental results obtained from comprehensive models designed to predict financial market trends using daily data from 2012 to 2022 have revealed significant challenges in performance. Negative Sharpe ratios and suboptimal an-nualized returns have been observed, prompting a critical examination of the issues associated with training on extended time-series data. It has been recognized that one of the primary challenges arises from the inherent complexity and variability of financial markets over long periods. Financial markets are influenced by a multitude of factors—including economic changes, political events, and evolving market sen-timent—which have been observed to change over time. Data spanning a decade has been found to encapsulate multiple market cycles, such as bull and bear phases, recessions, and recoveries. Consequently, the model has been required to generalize across highly diverse market conditions, a task that has proven challenging even for sophisticated models like RNNs and LSTMs.

An additional challenge has been attributed to the nature of traditional time-series models. These models, which are typi-cally adept at capturing short- to medium-term dependencies, have been observed to struggle with long-term historical data. The inability to effectively leverage the full depth of informa-tion in a decade-long dataset has been noted, particularly as fi-nancial time-series data often exhibit non-stationarity—where

TABLE I

SUMMARY OF PERFORMANCE METRICS FOR VARIOUS MODEL ARCHITECTURES. ALL EXPERIMENTS WERE CONDUCTED ON A LONG-TERM DAILY DATASET SPANNING FROM JANUARY 1, 2017, TO DECEMBER 31, 2022. THE SUBOPTIMAL RESULTS INDICATE THAT NEITHER RNNS NOR LSTMS WERE CAPABLE OF EFFECTIVELY LEARNING PATTERNS OVER SUCH AN EXTENDED PERIOD.

c

Hyperparameter	Losses	Train Sharpe Ratios	Train Annualized Returns	Test Sharpe Ratios	Test Annualized Returns
conv 2 rnn 1 hidden 50	0.0025	-0.891	0.046	-0.982	-0.127
conv 2 rnn 2 hidden 50	0.0026	-1.147	0.040	-1.244	-0.103
conv 2 rnn 2 hidden 100	0.0025	-0.927	0.044	-1.017	-0.121
conv 2 rnn 2 hidden 200	0.0026	-0.920	0.045	-1.013	-0.124
conv 2 rnn 2 hidden 400	0.0027	-1.464	0.032	-1.559	-0.077
conv 2 rnn 4 hidden 50	0.0025	-0.891	0.046	-0.982	-0.127
conv 2 rnn 8 hidden 50	0.0025	-0.891	0.046	-0.983	-0.127
conv 3 rnn 1 hidden 50	0.0025	-0.890	0.046	-0.982	-0.127
conv 3 rnn 2 hidden 50	0.0025	-0.891	0.046	-0.982	-0.127
conv 3 rnn 2 hidden 100	0.0025	-0.895	0.046	-0.985	-0.127
conv 3 rnn 2 hidden 200	0.0025	-0.899	0.045	-0.990	-0.125
conv 3 rnn 2 hidden 400	0.0025	-0.903	0.047	-0.999	-0.134
conv 3 rnn 4 hidden 50	0.0025	-0.891	0.046	-0.982	-0.127
conv 3 rnn 8 hidden 50	0.0025	-0.891	0.046	-0.983	-0.127
conv 4 rnn 1 hidden 50	0.0025	-0.891	0.046	-0.982	-0.127
conv 4 rnn 2 hidden 50	0.0025	-0.891	0.046	-0.982	-0.127
conv 4 rnn 2 hidden 100	0.0025	-0.891	0.046	-0.982	-0.127
conv 4 rnn 2 hidden 200	0.0025	-0.893	0.046	-0.985	-0.127
conv 4 rnn 2 hidden 400	0.0025	-0.901	0.045	-0.991	-0.123
conv 4 rnn 4 hidden 50	0.0025	-0.891	0.046	-0.982	-0.127
conv 4 rnn 8 hidden 50	0.0025	-0.891	0.046	-0.982	-0.127
conv 5 lstm 2 hidden 200	0.0025	-0.891	0.046	-0.982	-0.127
conv 5 rnn 1 hidden 50	0.0025	-0.891	0.046	-0.982	-0.127
conv 5 rnn 2 hidden 50	0.0025	-0.891	0.046	-0.982	-0.127
conv 5 rnn 2 hidden 100	0.0025	-0.891	0.046	-0.982	-0.127
conv 5 rnn 2 hidden 200	0.0025	-0.891	0.046	-0.982	-0.127
conv 5 rnn 2 hidden 400	0.0025	-0.895	0.045	-0.986	-0.125
conv 5 rnn 4 hidden 50	0.0025	-0.891	0.046	-0.982	-0.127
conv 5 rnn 8 hidden 50	0.0025	-0.891	0.046	-0.982	-0.127

statistical properties like mean and variance change over time. This non-stationarity has made it difficult for the models to learn stable and generalizable patterns. Furthermore, the extensive volume of data associated with long-term time-series has introduced significant computational challenges, as training deep learning models on such datasets demands considerable resources and time. The risk of overfitting has also been acknowledged, wherein models may learn patterns or noise that do not generalize well to unseen market conditions.

B. Limitations of RNN and LSTM in Handling Extended Time-Series

A modified approach was adopted by transforming the prediction task into a classification problem. It was revealed by the performance metrics—especially when comparing various dataset lengths—that a critical limitation exists in Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks when handling extended time-series data.

For shorter sequences (e.g., 100 or 200 data points), rela-tively high accuracy was achieved. However, as the length of the time-series increased to 400, 800, and particularly 1600 data points, both training and testing accuracies experienced a noticeable decline. This pattern has suggested that, despite be-ing

designed for sequential data, RNNs and LSTMs encounter difficulties in capturing and utilizing information from lengthy time-series. The vanishing gradient problem, where gradients diminish or explode as they are propagated through long sequences, has been frequently attributed as the underlying cause of these difficulties.

Moreover, financial time-series data have been observed to encompass complex and evolving market dynamics, which are challenging to encapsulate in a model trained on lengthy historical sequences. As the sequence length increases, the ability of the models to generalize and adapt to changes diminishes. Practical considerations, such as the increased computational demands and extended training times for longer sequences, have also been recognized. In many cases, the additional data length has not translated into a proportional increase in informative content, leading to inefficiencies and a heightened risk of overfitting, whereby noise or irrelevant patterns are learned from the extended sequences.

6. Summary and Conclusion

Extensive experimentation with RNNs and LSTMs for pre-dicting financial market trends using a decade's worth of daily data has yielded several critical insights and highlighted sig-nificant challenges. It was observed that although the models were designed to handle sequential data, they exhibited marked limitations when applied to extended time-series, as evidenced by the declining accuracy with increasing dataset lengths. A fundamental issue was identified in the models' ability to process and learn from long-term dependencies—an essential requirement in financial time-series analysis. The vanishing gradient problem was found to become more pronounced with longer sequences, thereby hindering the learning process. Additionally, the complexity and non-stationary nature of financial markets have compounded these challenges, as the models have struggled to generalize across the various market conditions represented in lengthy datasets.

The importance of data representation and feature selection has been underscored by the findings. It has been observed that more informative features can be derived by using closing prices and by computing returns as the ratio of the subsequent day's price to the previous day's price (minus one) instead of relying on raw prices. This approach, which is more aligned with underlying financial theory, provides a standardized and relative measure of market movements. Furthermore, the incor-poration of traditional financial analysis techniques as inputs to the neural networks has been suggested as a means to enhance model performance. Such integration has been posited as a more effective strategy than the exclusive use of complex models.

It has also been recognized that increasing model com-plexity does not necessarily result in improved performance, especially in markets characterized by high levels of noise. A delicate balance between model complexity and the risk of overfitting has been observed. While neural networks possess considerable power, they may also amplify the noise inherent in the data, rendering long-term predictions less reliable.

Looking ahead, the adoption of online learning approaches has been proposed as a potentially more effective strategy, given the dynamic and real-time nature of financial markets. In online learning, models are continuously updated with new data, thereby enhancing their ability to adapt swiftly to

evolving trends and market changes. This approach could help address some of the challenges posed by the non-stationary nature of financial data and improve overall model responsiveness.

In conclusion, a nuanced approach is underscored as nec-essary for the successful application of deep learning models to financial time-series prediction. The challenges presented by extended time-series data necessitate careful considera-tion of model architecture, feature engineering, and learning strategies. The integration of traditional financial analysis with advanced machine learning techniques, the careful balancing of model complexity, and the exploration of online learning methodologies are identified as promising directions for future research.

References

1. R. C. G. R. Wolfgang Karl Hardle,¨ Campbell R Harvey, “Understanding cryptocurrencies,” The Journal of Financial
2. Econometrics, 2020. Accessed via Oxford Academic [URL: https://doi.org/10.1093/jjfinec/nbz033].
3. C. Rueckert, “Cryptocurrencies and fundamental rights,” The Journal of Cybersecurity, 2019.
   e. Christian Szegedy, Wei Liu, “Going deeper with convolutions,” Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–9, 2015.
4. A. not specified, “Fuzzy inference-based lstm for long-term time series prediction,” Scientific Reports, Date not specified. Accessed via Nature.
5. S.-H. Poon and C. W. Granger, “Volatility forecasting i: Garch models,” Journal of Economic Literature, 1997. Accessed via JSTOR.
6. I. Psaradellis et al., “The profitability of a combined signal approach: Sma and ema trading signals,” International Review of Financial Anal-ysis, 2015. Accessed via ScienceDirect.
7. F. Jareno˜ and A. B. Garc´ıa, “Technical analysis and stock market efficiency,” Managerial Finance, 2001. Accessed via Emerald Insight.
8. P. T. Chio, “A comparative study of the macd-base trading strategies: evidence from the us stock market,” arXiv preprint arXiv:2206.12282, 2022. Accessed via arXiv.
9. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning represen-tations by back-propagating errors,” Nature, vol. 323, no. 6088, pp. 533– 536, 1986.
10. S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.