Predicting Stock Market time-series data using CNN-LSTM Neural Network model: A Review

Claudio Saponaro

May 2025

Abstract

The stock market has been identified as a primary indicator of stakes in business ownership. In the absence of adequate equity, firms encounter significant challenges in sustaining their financial operations. It is widely recognized that forecasting stock performance is a notoriously challenging endeavor, given the continuous fluctuations in market prices. Despite the plethora of machine learning approaches that have been proposed, many have been found to be deficient, often due to inadequate library support or suboptimal accuracy when applied to live data. The experimental results obtained by the authors, which were derived from real-time stock market datasets, demonstrate that the CNN–LSTM model outperforms competing methods, achieving a significantly higher level of accuracy. The objective of this review the aim is to methodically analyze and reproduce the experiment conducted by the authors. This experiment involved investigating a proposed architecture, a hybrid CNN-LSTM model, to address such task.

1 Introduction

In recent years, Machine Learning (ML) techniques have been introduced to forecast stock performance and provide investment suggestions to shareholders, with the aim of supporting financial growth. However, these models continue to demonstrate deficiencies in terms of precision.
This paper reviews existing approaches to stock market prediction and proposes a novel method that combines Convolutional Neural Networks (CNN) with Long Short-Term Memory (LSTM) networks for analyzing time series data. The objective of this review activity is twofold: first, to examine the key points of the paper, and second, to reproduce the proposed experiments using different models and compare their performance.

1.1 Previous Works

Prior to the development of the proposed model based on a CNN-LSTM architecture, the authors conducted a comprehensive literature survey to elucidate

the merits and limitations of previous models.
In summary, this list shows the papers analyzed by the authors before implementing their current solution.
The following section offers a more precise summary of the results found by the authors who analyzed all of the aforementioned papers.

- **Subhadra and Kalyana**
    1. Compared various ML methods.
    2. Found that Random Forest outperformed Linear Regression.
    3. Accuracy degraded without smoothed preprocessing.
- **Pang, Zhou et al.**
    1. Compared RNN vs. AELSTM (LSTM + Auto-Encoder).
    2. Accuracy was low due to outdated libraries and real-time data issues.
    3. Emphasized challenges in real-time training.
- **Uma and Kotrappa**
    1. Proposed LSTM with a Log Bilinear layer.
    2. High accuracy on COVID-19 data.
    3. Not tested on real-time or post-COVID data.
- **Kimoto, Yoda, Takeoka**
    1. Built a modular neural network with technical/economic indices.
    2. Outperformed multiple regression using correlation metrics.
- **Guresen, Kayakutlu, Daim**
    1. Evaluated DAN2, MLP, hybrid models.
    2. MLP was most reliable; hybrid methods failed to improve results.
- **Nelson, Pereira, Oliveira**
    1. Used LSTM with 15-minute time steps and indicators.
    2. Achieved 55.9% trend prediction accuracy.
- **Selvin, Menon, Soman et al.**
    1. Compared CNN, RNN, LSTM with sliding windows.
    2. CNN outperformed due to use of current window data.
- **Hiransha, Gopalakrishnan, Soman**
    1. Compared ANN, MLP, RNN, LSTM, CNN.

2. CNN performed best over time, despite accuracy drops.

- **Bansal, Hasija et al.**
  1. Proposed intelligent decentralized stock model using DAG-crypto and ML.
  2. Used LSTM with open/close/low/high features.
  3. Achieved 99.71% accuracy (100 epochs, batch size 50).

2 Objectives

The goal of this review is to :

- Analyze the effectiveness and the architecture of the hybrid CNN-LSTM model for stock market prediction
- Reproduce and validate the experimental results presented in the original paper
- Compare the performance of the CNN-LSTM model with the other model LSTM-GRU.

3 Relevance and Impact for stock market analysis

3.1 Relevance in the stock market analysis

This paper is highly relevant to the field of stock market prediction since is focused on achieved a low error using a real dataset instead of a sample one; to do that the authors specifically designed their model to parse and predict using real-time stock market data from multiple markets (NYSE, NIFTY, NASDAQ).

3.2 Impact on Stock Market Analysis

In this subsection, I will focus the attention on analyzing the impact of this paper on stock market analysis.

1. **Methodological Innovation:**
   - The CNN-LSTM architecture represents an innovative approach based on the combination of the CNN's feature extraction capabilities with LSTM's pattern recognition for time-series data.
   - The paper demonstrates how to transform 1D financial data into tensor formats suitable for CNN processing, which represents a technical contribution.

2. **Performance metrics:** According to the paper the model achieved impressive result in with MSE values between MSE values between 0.001 and 0.035 using different datasets[1].

3. **Focus on practical implementation:**

    - Unlike many academic papers, this research take in consideration also the deployment aspects, including Docker containerization and Kubernetes support.

4 Proposed Methodology

The present section is focused on the analysis of the methodology employed by the authors during the model's creation process is as follows: the authors have focused on the essential requirements for developing a machine learning model that is capable of effectively predicting stock market trends.
In light of the aforementioned insights and their subsequent research, a CNN-LSTM hybrid model was developed.
This model utilizes CNN for the features extraction and LSTM for temporal pattern recognition. The model was specifically tailored for time-series regression tasks.
In order to enhance performance, the model's architecture was fine-tuned and the Mean Squared Error was used as the primary evaluation metric.

4.1 Convolutional Neural Networks (CNNs)

In the subsequent section, a concise exposition of the mathematical models of CNN and LSTM will be provided, as these models constitute the foundational components of the examined model.
A Convolutional Neural Network (CNN) is a deep learning model frequently employed for the extraction of spatial features from data, including images and time series. The model utilizes convolutional layers, which implement filters (kernels) over input data to discern local patterns such as edges, trends, or movements.
For CNN layers:

$$z_{i,j,k} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \sum_{c=0}^{C-1} x_{i+m,j+n,c} \cdot w_{m,n,c,k} + b_k \tag{1}$$

The provided equation describes the convolutional operation in a CNN layer

- $z_{i,j,k}$: represent the output value at position (i, j) in the k-th output channel (feature map).

- $x_{i+m,j+n,c}$: represent the input value at the offset position of m, n in the input channel c.

[1] These results will be discussed in details in the *Results section*

- $w_{m,n,c,k}$: Weight (kernel) value at *(m, n)* for the input channel and output channel k.
- b_k : bias term for the output channel k

4.2 Long Short-Term Memory Networks (LSTMs)

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) designed to capture long-term dependencies in sequential data by using gated mechanisms to control information flow. The following mathematical formulation of LSTM is hereby presented:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3}$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{4}$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{5}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{6}$$
$$h_t = o_t \odot \tanh(C_t) \tag{7}$$

4.3 CNN-LSTM Architecture

This part of the section is devoted to the analysis of the architecture proposed by the authors.

1) CNN:

The authors adopted a custom architecture for the CNN component of their model, opting against a conventional ascending order in layer sizes. Instead, they employed three convolutional layers with neuron sizes of 64, 128, and 64, respectively, each using a kernel size of 3.

MaxPooling layers were inserted between each convolutional layer to reduce dimensionality.

A Flatten layer was included at the end of the CNN section to convert the output tensors into a one-dimensional array.

To accommodate the time-series nature of the data, all CNN layers were wrapped with the TimeDistributed function, enabling the model to process each temporal slice of the input individually.

The output from this CNN section was then passed to the LSTM component.

2) LSTM:

The LSTM module consisted of two Bidirectional LSTM (Bi-LSTM) layers, each with 100 units, designed to capture temporal features in both forward and backward directions.

To enhance model stability and prevent overfitting, dropout layers with a rate of 0.5 were interleaved between the Bi-LSTM layers.

The final prediction layer was a dense layer with a linear activation function.

For training the model, the authors selected the Adam optimizer, although the SGD optimizer was also tested and found viable, Adam was ultimately chosen due to superior accuracy observed during experimentation. Mean Squared Error (MSE) was used as the loss function, with both MSE and Mean Absolute Error (MAE) employed as evaluation metrics.
Here provided the full architecture employed by the authors:

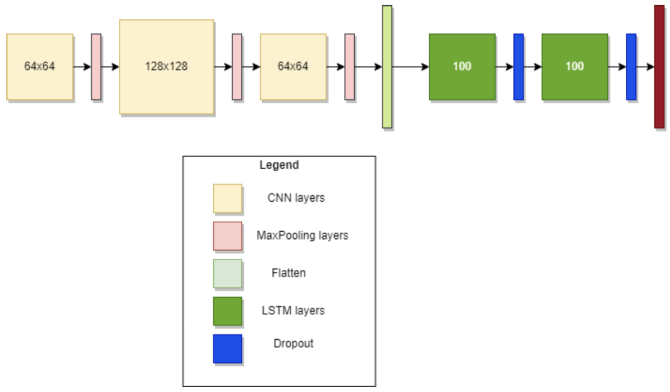

Figure 3: CNN-LSTM model's architecture [1]

5 Data Preprocessing

Before going into the details about the data preprocessing step and model training here is the pipeline realized by the paper's authors to accomplish the task starting from the data acquisition and finishing with the results collection.

5.1 Dataset Description

In this section, i will describe the dataset used to run the experiment, its structure, and the preprocessing operation before training the model. The dataset used for training the model was obtained using the Alpha Vintage API, which has the main features of providing data in csv format ready for preprocessing. The dateset obtained using the aformentioned API present this charachteristics:

- The features are the following:
    1. Unammed colums: represent an identifier column
    2. Open: The opening price of the financial instrument for the trading period.

3. High: The highest price reached by the financial instrument during the trading period.
4. Close: The final price of the financial instrument at the end of the trading period.
5. Adjusted close: The closing price that has been adjusted to reflect corporate actions like dividends, stock splits. This provides a more accurate measure for long-term analysis.
6. Volume: The number of shares or contracts traded during the period, indicating the level of trading activity.
7. Divided amount: The amount of dividend paid per share during this period, if any. Zero values indicate that no dividend was paid.
8. Split cf (Split coefficient factor): A value of 1 indicates no split, while other values indicate stock splits or reverse splits. For example, a value of 2 could indicate a 2-for-1 split.
9. Date: represents the date using the standard format YYYY-MM-DD; data were collected starting from 1999-11-01 till 2021-11-26.

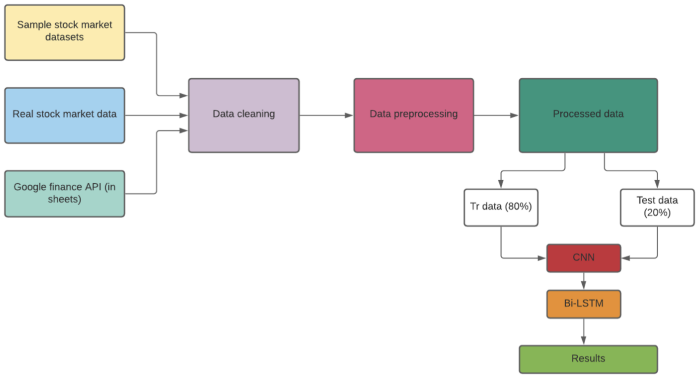

Figure 4: Pipeline of the project [1]

5.2 Feature Engineering

This subsection is devoted to the analysis of the preprocessing steps, which were done by examining the source code. The initial step in this process entails populating the *Nan* elements of the data set with the mean calculated for the designated feature. A rapid modification was implemented to the source code to make it work.

```
data.fillna(data.mean(), inplace=True)
```

I have used:

```
data.fillna(data.select_dtypes(include='number').mean(), inplace=True)
```

which compute the averaging operation only on numeric columns.
Without delving into all the granular details of the preprocessing, in the source code, what the authors have done is to enhance the initial data set by calculating the moving averages of the closing price over 10, 50, and 100 days in order to smooth out the price data and identify trends/pattern in the data.

5.3 Data Transformation

Following the principle of the ML preprocessing, the authors have done two fundamental steps before model training:

- Creating the sequence for the LSTM input
- Data splitting using the sklearn library (train 80% and test 20%)

6 Experimental Analysis

This section is devoted to the reproducibility of the results of the original paper In order to reproduce the result of the original i have cloned the repository from github and used pycharm as IDE, within pycharm i used jupyter notebook to run the code as the authors suggested.

6.1 Models Tested

The models which as been tested for this reproducibility experiment are:

- The proposed hybrid CNN-LSTM
- LSTM-GRU

6.2 Evaluation Metrics

For evaluating the models the basic regression metrics have been used:

- Mean Squared Error (MSE)
- R^2 Score
- Mean Absolute Error (MAE)

7 Reproduction of Neural Network Calculation & Code

The results have been reproduced using the notebook provides by the authors in the github repository.

Model	MSE	R^2 SCORE	MAE
CNN-LSTM	0.0013	94.2633	0.0272
LSTM-GRU	2.0843	96.5065	8.5636

Table 1: Performance comparison between the two different models

Here a brief table of the results, in the full jupyter notebooks provide other metrics which have been employed.

The main difference is that I only did training and testing using both models, while the authors also re-trained the main architecture (CNN-LSTM) using a different dataset (data2.csv), but only for this model and tested on it. In my opinion, to make the comparison fair, it was necessary to use only one training phase using the main dataset (data.csv). Otherwise, it is possible to see the clear superiority of the CNN-LSTM model over the LSTM-GRU model.

Model	MSE
CNN-LSTM (paper result)[2]	0.0035
CNN-LSTM (my result)	0.0013

Table 2: Performance comparison between the proposed model

8 Conclusion

In conclusion, after thoroughly analyzing the performance of both models on the stock prediction task, CNN-LSTM demonstrates significantly superior results compared to LSTM-GRU. The empirical evidence presented in this study strongly supports this finding across multiple evaluation metrics.
The substantial performance gap between these architectures can be attributed to several key factors:

- CNN-LSTM's convolutional layers excel at capturing spatial patterns and local features in the time series data, allowing it to better identify relevant short-term market signals that are critical for stock prediction.
- While LSTM-GRU offers theoretical advantages in processing sequential data, its performance appears constrained when handling the multi-dimensional aspects of stock market behavior, particularly in volatile market conditions.

These findings have important implications for financial forecasting applications.

[2]On average because according to the source code they have documented the results with two datasets (data.csv, data2.csv)

References

[1] Aadhitya A, Rajapriya R, Vineetha R S, Anurag M Bagde; Predicting Stock Market time-series data using CNN-LSTM Neural Network model.