LSTM for Financial Time Series

Gelsomina Pagliaro

Abstract

This study explores the application of deep learning in financial time series forecasting. Traditional methods, like Autoregressive Integrated Moving Average (ARIMA), face limitations in handling non-linear relationships and capturing long-term dependencies. Deep learning models, particularly LSTMs, offer a flexible and powerful approach to overcome these challenges. The work highlights the roots of deep neural networks to give a basic understanding of how they work and why they seem so promising for the forecasting field. Among them, particular attention is given to the long-term memory which preserves all of the advantages of a Recurrent Neural Network while fixing the vanishing-exploding gradient problem. The work aims to understand how to optimize the usage of this architecture for financial time series forecasting and its potential limitations. While these precautions are taken the LSTM outperforms other models, particularly the classical ones. Comparisons with classical forecasting models demonstrate its superiority in enhancing forecasting accuracy, it proves to be a valuable tool for informed investment decision-making in the dynamic landscape of financial markets. The study concludes by considering also other deep neural network architectures to provide a complete portrait of the kind.

Keywords

LSTM, ANN, Financial Time Series, ARIMA, financial market

1. Introduction

The financial industry places a strong emphasis on forecasting, particularly asset price forecasting, as it helps implement either hedging or speculative strategies: a correct prediction can generate a remunerative advantage for those who can accurately do it.

Financial forecasting refers to the process of making predictions about future financial outcomes based on historical data and knowledge of economic and market trends. Investors have always looked forward to a powerful and effective technique that allows them to be as accurate as possible. However, it remains a hard and challenging task due to factors such as market volatility, information asymmetry, regulatory fluctuations, and the intricate nature of financial instruments.

1.1. Classical approaches

Stated the pivotal role of forecasting in making informed investment decisions many solutions have been proposed over the years. One notable approach to address this challenge is credited to Box and Jenkins (1976), who introduced the integrated Autoregressive moving average (ARIMA) model. It is a generalized model of the Autoregressive moving average (ARMA). This model has a simple structure: the non-stationary operators and the stationary ones: AR and MA which determine short-term prediction. Its parameters are: *p, d, q* where:

- p: is the order of the autoregressive component.
- d: is the degree of differencing.
- q: is the order of the moving average component.

Its extended equation is reported below:

$$\hat{y}_t = \mu + \phi_1 y_{t-1} + \ldots + \phi_p y_{t-p} - \theta_1 e_{t-1} - \ldots - \theta_q e_{t-q}$$

This model works fairly well with stationary time series and for short-term forecasting, so few lags ahead, but their performance tends to get worse for longer periods. The limitations arise from the linear relationship that the model imposes on the data and the assumption that the standard deviation in errors remains constant, which represents a simplification that likely would be unrealistic for modeling real-life dynamics. Moreover, they manage to work only with stationary time series, the data can not be a direct input for the model but they need to be reshaped with some specific technique to be stationary. Finally, it struggles to handle missing or corrupt data.

This linear-based approach has been developed over the years and accordingly, many variations have been introduced, such as SARIMA (Seasonal ARIMA) or ARIMAX (ARIMA with explanatory variables). These models try to fix

some of the issues presented by the ARIMA model; for instance: the SARIMA model takes into account any seasonality patterns, and it brings seasonality as a parameter, which makes it significantly more powerful than ARIMA in forecasting complex data spaces containing cycles, while the SARIMAX model takes into account exogenous variables, or in other words, use external data for the forecast.

1.2. Deep Learning

To overcome these issues once and for all Deep Learning comes in handy. Machine learning and in particular the new architecture implemented by Deep Learning allows us to drop all the tight assumptions required so far and to work with the data directly. Deep learning techniques can support multivariate inputs, and capture intricate structures and patterns within data, including non-linearities and complexities, granting great results when applied to financial time series. Financial time series, essentially sequences of correlated data, exhibit autocorrelation in practice. This characteristic makes them particularly suitable for Recurrent Neural Networks (RNN), notably Long Short-Term Memory (LSTM) neural networks.

The strength of this neural network lies in its capability to retain information over long time horizons (long memory) and so long-term values while managing and preserving the significance of small values (short memory). They are indeed popular in many fields involving sequences such as Natural Language Processing: language translation, sentiment analysis, speech recognition, and financial time series analysis.

Many works proved the superiority of the Deep Neural Network in the field, just to mention some: Sima Siami-Namini et al.[9] provided a direct comparison on prediction performance between the classical models and the LSTM on a data set of 120,132 observations of financial indexes. The results showed an impressive performance for the LSTM which reduced the RMSE by 88.07%. Sreelekshmy Selvin et al.[8] worked on a different data set and still, comparing the ARIMA with the LSTM, captured the better results of the latter when coming to forecasting.

1.3. Artificial Neural Network

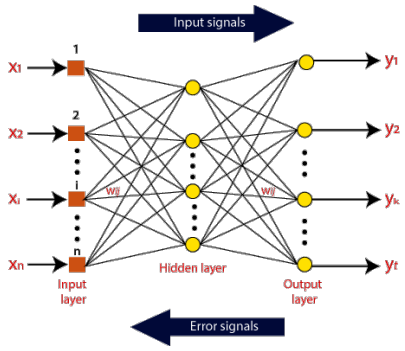
Deep learning is an artificial neural network consisting of multiple processing layers that facilitate the creation of high-level abstractions, allowing the model to represent and analyze complex datasets effectively.

Yann et al. [5] defined deep learning as *"representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input)into a representation at a higher, slightly more abstract level"*. By combining a sufficient number of these transformations, very complex functions can be learned.

The key aspect of deep learning is that these layers of features are not manually crafted instead, they are acquired from data through a general-purpose learning process.[5]

The basis of deep learning are ANNs: they are inspired by biological learning systems which consist of a complex web of interconnected neurons. The historical roots of ANNs can be traced back to the perceptron, innovated by Frank Rosenblatt in 1957, by then their architecture has become extremely sophisticated and effective.

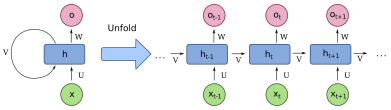
Nowadays their structure consists of three different layers: input, hidden, and output layer all connected by nodes that simulate the role of neurons in a human brain. According to Schmidhuber [7] *"A standard neural network consists of many simple, connected processors called neurons, each producing a sequence of real-valued activations."*. Input neurons take signals from the environment X_i, and output neurons y_j present signals to the environment. Neurons that are not directly connected to the environment, but which are connected to other neurons, are called hidden neurons. The simplest form of neural network is the *feed-forward neural network* where each neuron provides an input to each neuron in the following layer, and none of the weights give an input to a neuron in a previous layer. The connection between the neurons of each layer has weights and biases that transform the row data given as input, consequently, they are transformed by an activation function, typically nonlinear, which may vary according to the objective of the neural network. The most common neural network learning technique is the error Backpropagation algorithm. It uses gradient descent to learn the weights in multilayer networks. It starts backward from the output layer towards the input layer, indeed the activation function of the neuron must be differentiable. The weights and biases are initialized randomly and adjusted through the training, each epoch computes better parameters than the last one based on the computed gradients.



Deep ANNs are most commonly referred to as deep learning (DL). DNNs are essentially Artificial Neural Networks (ANNs) with multiple hidden layers between the input and output layers. Over recent years, various deep learning models have emerged, with some of the most typical ones being Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Stacked Auto-Encoders (SAE), and Deep Belief Networks (DBN). The one I will dive into for this work is the Recurrent Neural Network.

RNNs are suitable for multivariate time series data and capable of capturing temporary dependencies over variable periods. This is due to the main property of the RNN: it has an internal state which is fed back to the input. It uses the current information on the input and the prediction of the last input for the next prediction. This allows the model to build a *memory*.

The Backpropagation algorithm can be adapted to train a recurrent network by unfolding the network through time and restricting some of the connections to keep the same weights at all times. These feedback connections enable RNNs to propagate data from earlier events to current processing steps.



The most common and well-documented learning algorithms for training RNNs for supervised learning tasks are Backpropagation through time (BPTT) and real-time recurrent learning (RTRL).

This feature makes this neural network extensively suitable for time series prediction since it has all the benefits of an ANN, i.e. to adapt to any non-linear data set without any static assumption and prior knowledge of the time series, and it can exploit its memory of past values for future prediction.

Standard RNN cannot bridge more than 5–10 time steps. This is due to that back-propagated error signals tend to either grow or shrink with every time step. This is the reason why it is not considered state-of-the-art for this purpose and it is called *vanishing-exploding* gradient problem, which means that the gradient gets smaller and smaller approaching zero or it gets larger and larger. In the first case, the gradient may not converge, in the latter, it diverges causing high oscillations of the weights. Both cases do not allow the weights and biases to be updated correctly making the RNN impossible to train on the given data set. The problem was explored in depth by Hochreiter (1991) and Bengio, et al. (1994) [3], who found some pretty fundamental reasons why it might be difficult.

To address this issue, Long short-term memory proves to be a valuable solution.

2. Related Work

2.1. Long short-term memory architecture

Long Short-Term Memory Networks (LSTM) are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter and Schmidhuber (1997) [4]. The main difference with the previous models is the presence of a *cell state* i.e. the long-term memory functioning as a sluice gate and controller of storing past states. An LSTM cell consists of:

Cell State Runs through the entire network and can add or remove information with the help of gates.

$$c_t = \tanh(W_c h_{t-1} x_t + b_c)$$

Forget gate layer also called 'sigmoid layer'. It decides the fraction of the information to be allowed.

$$f_t = (W_f h_{t-1} x_t + b_f)$$

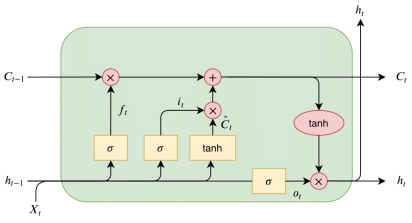
Input gate layer: consists of the input.

$$i_t = (W_i h_{t-1} x_t + b_i)$$

Output gate layer: output generated by the LSTM.

$$o_t = (W_o h_{t-1} x_t + b_o)$$

$$h_t = o_t \tanh(c_t)$$



In the recurrence of the LSTM, the activation function is the identity function with a derivative of 1.0. So, the backpropagated gradient neither vanishes nor explodes when passing through, but remains constant.

The linear activation function, also known as "no activation," or "identity function", is where the activation is proportional to the input. The function doesn't do anything to the weighted sum of the input, it simply spits out the value it was given.

The effective weight of the recurrency is equal to the forget gate activation. So, if the forget gate is on (activation close to 1.0), then the gradient does not vanish. Since the forget gate activation is never > 1.0, the gradient can't explode either.

2.2. LSTM for Time Series

LSTM may appear as the perfect and the most suitable model for time series forecasting due to its properties. However, to develop and utilize the model for this purpose some considerations might be essential. I will present some basic concepts that are essential when dealing with LSTM for time series forecasting and they are going to be deeply discussed with some examples.

The first step regards the data, i.e. the input that is going to feed the model for its training. One essential step is to normalize the data: all features need to have a similar scale, in a range of [0,1].

$$\text{Normalized Value} = \frac{x - \mu}{\sigma}$$

Many works include this step in their prediction process but Kai Chen et al. [1] show how this process is crucial. They focus on China's stock return prediction using LSTM, the features chosen for the model were: closing price and trade volume; the model was first fed with the unnormalized features and then with the normalized ones, the accuracy improved from 15.6% to 19.2%. Higher results were obtained adding more features to train the model. This is reasonable, Neural networks, including LSTMs, are sensitive to the scale of input features, and this step prevents certain features from dominating the learning process due to their larger magnitudes, in this way, the network focuses on the patterns in the data rather than the absolute values.

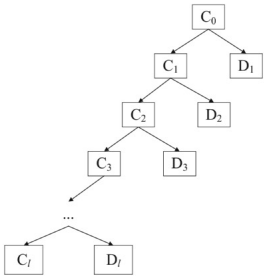
An alternative approach that can be exploited to smooth the data and remove any possible noise is to use the moving average filter as a type of low-pass filter as reported by the work of Alaa Sagheer & Mostafa Kotb [6]. This filter calculates a weighted average of previous data points within a five-point time in the time series, producing a smoothed estimation.

Besides the classical approaches, other precautions can be taken to help the convergence and to improve the generalization ability of the model. One of them may be the wavelet decomposition of the original time series. A meaningful study on this matter is provided by Hongju & Hongbing [11]: "Financial time series are susceptible to many factors ... which usually contains a lot of noise and show significant non-stationarity characteristics". To address this potential issue they propose a "wavelet denoising" technique to reduce the short-term disturbance. Wavelet analysis is a mathematical technique typically used for transforming signals or time-series data and it is based on the use of wavelets. The technique

involves multi-resolution analysis which breaks down each layer of the input signal into low-frequency and high-frequency components, with only the low-frequency part undergoing further decomposition. The mathematical formulation is:

$$C_0 = C_1 + D_l + D_{l-1} + \ldots + D_2 + D_1$$

Which is equivalent to the following decomposition of the time series:



C_0 represents the original time series while the decomposition components represent the low and high frequencies signals.

In the wavelet decomposition of financial time series data, the low-frequency component represents the overall trend, while the high-frequency component reflects short-term random disturbances. Removing the high-frequency part by setting it to zero helps eliminate noise, resulting in a smoother signal that approximates the original data. This approach prevents neural networks from overfitting to short-term disturbances. They used the sym4 wavelet basis to decompose the time series into four layers and then reconstructed it. This choice, combined with others, provided very good results. It is proven that the wavelet reconstruction can effectively smoothen the raw data and retain its approximation signals.

After pre-processing the data to optimize its suitability for LSTM prediction, other concerns can arise. Fine-tuning hyperparameters is crucial as any adjustments to them can affect the whole outcome.

In particular, it needs to be optimized the number of nodes, hidden layers, epochs, the presence of a dropout layer, the choice of the most suitable activation function for the pursued objective, the batch size, as much as the choice of the optimization algorithm. The whole setup has a very strong impact on the accuracy of the prediction so it has to be completed accurately.

For this particular concern, many potential solutions have been proposed. One of them is the employ of the Particle Swarm Optimization (PSO) algorithm to set the number of neurons and of previous time steps. The study, provided by Xuanyi Song et al. [10], proposes an LSTM to infer the production of fractured horizontal wells in a volcanic reservoir. Even though this method is applied to a non-financial time series it shows great potential, so it is a reasonable method that can be engaged also for a financial problem. The PSO is a heuristic search algorithm inspired by the biological and sociological behavior of bird flocks in searching for food (Kennedy and Eberhart, 1995).

The algorithm can be summarized as follows: birds are considered as particles and each has its position and velocity. Each particle adjusts them based on its own (local search) and the swarm's historical (global search) best solutions. The algorithm iteratively refines these positions to find the optimal solution to an optimization problem, balancing exploration and exploitation, alternatively, the algorithm stops when the maximum number of iterations is reached. The work performs the pre-processing steps on the raw data, then divides the obtained data set into training and testing data sets, and finally, before employing the LSTM for the prediction, the optimal number of neurons and lags are found with the PSO algorithm.

In the initial phase, the study is performed on simulator-generated data, where of course the noise component is reduced allowing the LSTM to obtain impressive results. In the simulated environment, optimal hyperparameters are identified as four for both the number of neurons and the window size. In the real-world scenario, the ideal hyperparameters are determined to be fifteen for the number of neurons and six for the window size. Even in this case, the LSTM outperformed other forecasting techniques, affirming that augmenting its complexity ensures accurate predictions.

However, it is important to not overfit the training set, the PSO should take care of this scenario. If it does not there are some valid solutions to address this grief. One of them is to add a Dropout layer to the LSTM architecture. It is a regularization layer that, during training, randomly "drops out" or deactivates a proportion of neurons in the network. This means that, for each training iteration, a subset of neurons is temporarily excluded from the forward and backward passes. This randomness allows the model to gain a more robust representation of data since it becomes less dependent on specific neurons.

Moreover, it needs to be specified the acti-

vation functions, and the optimization methods. Those are the other two important choices that can affect the final result. Typically, the preferred optimization method is *ADAM* while for the activation function, the most frequent are Hyperbolic Tangent (*tanh*) Activation and the Sigmoid Activation (σ) and, for the output layer Softmax Activation function.

Once the model is finalized and the possible precautions are taken, it is proved that the LSTM predict accurately the future out-sample values for a given time series, much better than the classical models.

2.3. Deep learning for financial time series

LSTM architecture is powerful and effective, it shows great results but it is not the only deep learning technique exploited for forecasting purposes. Indeed we need to make an essential distinction: the time length of the prediction.

It can be a short-term future prediction or a long-term one.

Short-term prediction emphasis is on capturing fine-grained patterns and rapid changes in recent data. Shallow network architectures are preferred to avoid overfitting and noise capture. Training on shorter data sequences, concentrating on recent patterns, is generally sufficient for effective short-term predictions.

Long-term prediction The focus is on capturing broader trends and structural shifts, incorporating macroeconomic indicators, fundamental factors, and historical data over longer periods. In this case, deeper network architectures can be more indicated since they can capture hierarchical representations and long-term dependencies. To enhance even more the model's ability it may be helpful to introduce attention mechanisms or memory augmentation.

Sreelekshmy Selvin et al.[8] employed an LSTM for a short-term future prediction. Their data set was composed of minute-wise stock prices for 1721 NSE-listed companies. Both RNN and LSTM utilize information from past time lags to forecast future instances. Given the highly dynamic nature of the stock market, the patterns and dynamics within the system are not consistently uniform. This variability poses challenges to the learning capabilities of LSTM and RNN architectures, leading to difficulties in accurately capturing dynamic changes. As a result, these models may struggle to adapt effectively to the evolving nature of the system. As an alternative, the work shows also the performance of a CNN for the same data set.

Convolutional Neural Networks (CNNs) are a specialized type of neural network designed for processing data with a known, grid-like topology. This encompasses various data types, including time-series data, conceptualized as a 1D structure. The network utilizes a mathematical operation known as convolution which is a specialized linear operation that involves sliding a small filter (kernel) over the input data to extract features, capturing spatial hierarchies.

The work shows that the CNN yields more accurate results compared to the other two models. This can be attributed to the fact that CNN does not rely on prior information for predictions. It solely utilizes the current window for forecasting, allowing the model to discern dynamic changes and patterns within the present context. The DNNs are compared based on the error percentage computed as:

$$\text{ep} = \frac{\text{abs}\left[X^i_{real} - X^i_{predicted}\right]}{X^i_{real}} \times 100$$

The table below shows the error percentage considering three different companies: 'Infosys', 'TCS', and 'Cipla'. The is computed as:

COMPANY	RNN	LSTM	CNN
Infosys	3.90	4.18	2.36
TCS	7.65	7.82	8.96
Cipla	3.83	3.94	3.63

The work aimed to *"identify wherever there is any long-term dependency existing in the given data"* [8]; by the performance of the three models it seems that there was not a long-term dependency indeed the best result in terms of error percentage was obtained by the CNN across all the three companies, with an average of 4.98%.

This case is not isolated, in the work of Hiransha M. et al. [2] where the data set consisted of day-wise closing price from two different stock markets: National Stock Exchange (NSE) of India and New York Stock Exchange (NYSE), the CNN outperformed other models. The network was able to predict for NYSE even though it was trained with NSE data. This was possible because both the stock markets share some common inner dynamics. From the NSE the corresponding stocks considered were: Maruti, Axis Bank, and Hcltech. To evaluate which model performed

better it used the Mean Absolute percentage error, computed as:

$$\text{MAPE}(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N-1} \frac{y_i - \hat{y}_i}{y_i}.$$

The table below shows the results to stress the outperformance of the CNN in the case of short-term prediction, where no clear patterns can be followed because of the dynamic and chaotic environment.

COMPANY	RNN	LSTM	CNN	MLP
MARUTI	7.86	6.37	5.36	6.29
HCL	8.53	6.97	6.42	7.38
AXIS BANK	9.27	8.13	7.94	8.10

However, this issue may be overcome with more sophisticated architecture. The study of Zhichao Zou and Zihao Qu [12] related to the use of LSTM in stock prediction and quantitative trading demonstrates that the predictive ability of the LSTM can be improved by involving the Attention mechanism. Machine learning algorithms are inspired by biological phenomena and human perception. We, as humans, do not treat all information with equal importance, instead, we focus on the important parts first for the newly received information. This phenomenon is analogous to the financial market as well, as the prices of securities assign different levels of importance to the market information, which prompted the authors to introduce an Attention mechanism to the original LSTM model.

They implemented soft attention, updating the model's input by assigning weights to information based on learning outcomes. This adjustment results in a more logical order of information. The researchers adopt the framework established by Qianqian for the Attention-LSTM, customizing it to suit financial models.

They proved, on a dataset of daily prices and volumes of the top 10 stocks in SP 500 based on market capitalization from 2004 to 2014, that the LSTM combined with the Attention mechanism can reach better results than without. To measure the model performance they engaged the Mean Squared Error computed as follows:

$$\text{MSE}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

In the table below the results are shown:

Mean Squared Error (*10^{-3})			
Stock Ticker	**LSTM**	**Stacked-LSTM**	**Attention-LSTM**
XOM	25	25	11
GE	18	2	3
WMT	128	163	76
MSFT	40	27	16
IBM	28	7	6
AAPL	54	61	44
GOOG	35	37	26
GS	12	16	18
PFE	32	64	49
JNJ	57	123	41

Finally, this comparison helps to understand that, when it comes to financial time series, even though the LSTM can be the most suitable deep neural network, the objective needs always to play a crucial role, since in some cases other architecture or some adjustments can highly improve the forecasting results.

3. Conclusion and future work

This essay aims to dive into the benefits provided by the DNNs, especially long short-term memory. The study emphasizes the importance of understanding the specific requirements and challenges posed by financial data, which often exhibits nonlinear patterns, long-term dependencies, and dynamic market conditions.

It discusses techniques, like data normalization moving filters and wavelet decomposition to handle noise and non-stationarity. The importance of tuning hyperparameters, such as the number of nodes, hidden layers and activation functions is emphasized. Additionally, it examines the application of optimization algorithms like Particle Swarm Optimization (PSO).

The article presents an analysis between term and long-term prediction scenarios. It highlights that the choice of architecture and preprocessing steps depends on the forecasting objective. Some studies demonstrate the effectiveness of LSTMs in short-term prediction while also mentioning cases where Convolutional Neural Networks (CNNs) outperform LSTMs due to their ability to capture changes within short-term contexts.

Furthermore, advancements in LSTM architectures are discussed, including the integration of Attention mechanisms. These mechanisms have been shown to enhance the capabilities of LSTMs. Lastly, the review concludes by emphasizing that deep learning models offer versatility for financial time series forecasting. While LSTMs are a tool, in this domain it is crucial to tailor the choice of architecture and adjustments according to the characteristics and objectives of the financial data being analyzed.

The continuous investigation, into structures and methods of attention brings attention to the changing nature of research in this field. As financial markets keep evolving the use of learning techniques combined with a comprehension of the data at hand shows tremendous potential for making precise and well-informed predictions, in the finance industry.

References

[1] Kai Chen, Yi Zhou, and Fangyan Dai. A lstm-based method for stock returns prediction: A case study of china stock market. In *2015 IEEE international conference on big data (big data)*, pages 2823–2824. IEEE, 2015.

[2] Ma Hiransha, E Ab Gopalakrishnan, Vijay Krishna Menon, and KP Soman. Nse stock market prediction using deep-learning models. *Procedia computer science*, 132:1351–1362, 2018.

[3] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[5] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[6] Alaa Sagheer and Mostafa Kotb. Time series forecasting of petroleum production using deep lstm recurrent networks. *Neurocomputing*, 323:203–213, 2019.

[7] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[8] Sreelekshmy Selvin, R Vinayakumar, EA Gopalakrishnan, Vijay Krishna Menon, and KP Soman. Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)*, pages 1643–1647. IEEE, 2017.

[9] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. A comparative analysis of forecasting financial time series using arima, lstm, and bilstm. *arXiv preprint arXiv:1911.09512*, 2019.

[10] Xuanyi Song, Yuetian Liu, Liang Xue, Jun Wang, Jingzhe Zhang, Junqiang Wang, Long Jiang, and Ziyan Cheng. Time-series well performance prediction based on long short-term memory (lstm) neural network model. *Journal of Petroleum Science and Engineering*, 186:106682, 2020.

[11] Hongju Yan and Hongbing Ouyang. Financial time series prediction based on deep

learning. *Wireless Personal Communications*, 102:683–700, 2018.

[12] Zhichao Zou and Zihao Qu. Using lstm in stock prediction and quantitative trading. *CS230: Deep Learning, Winter*, pages 1–6, 2020.