Day Trading Strategy Based on Transformer Model, Technical Indicators and Multiresolution Analysis

Salahadin A. Mohammed

Information and Computer Science Department,
King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia

***Abstract*—Stock prices are very volatile because they are affected by infinite number of factors, such as economical, social, political, and human behavior. This makes finding consistently profitable day trading strategy extremely challenging and that is why an overwhelming majority of stock traders loose money over time. Professional day traders, who are very few in number, have a trading strategy that can exploit this price volatility to consistently earn profit from the market. This study proposes a consistently profitable day trading strategy based on price volatility, transformer model, time2vec, technical indicators, and multiresolution analysis. The proposed trading strategy has eight trading systems, each with a different profit-target based on the risk taken per trade. This study shows that the proposed trading strategy results in consistent profits when the profit-target is 1.5 to 3.5 times the risk taken per trade. If the profit-target is not in that range, then it may result in a loss. The proposed trading strategy was compared with the buy-and-hold strategy and it showed consistent profits with all the stocks whereas the buy-and-hold strategy was inconsistent and resulted in losses in half the stocks. Also three of the consistently profitable trading systems showed significantly higher average profits and expectancy than the buy-and-hold trading strategy.**

Keywords*—Artificial neural network; saudi stock exchange; machine learning; deep learning; transformer model; stock price prediction; time series analysis; technical analysis; multiresolution analysis

I. Introduction

In the context of this study, day trading is a business of buying a number of shares on a trading day and selling them all before the end of the same trading day for a profit or a loss. Day trading is a business of probability. When a consistently profitable day trader enters a trade, he is not sure whether the trade will be a winner or a loser, but he is sure that after he does many trades, he will end up profitable. For example, the 2023 US investment champion's win rate was less than 35% but he ended the year with more than 805% profit [1], [2]. This is because he has a trading system with a positive expectancy. Expectancy, Φ, is defined as show by Eq. (1).

$$\Phi = AW \times WR - AL \times (1 - WR) \tag{1}$$

where, AW is average win, AL is average loss, and WR is win rate. For example, if a trader did a total of 1000 trades and 400 of them were winners, his WR is 0.4. If his average win is 500 dollars and his average loss is 200 dollars, then his Φ is $80 = 0.4 \times 500 - 0.6 \times 200$. This means he expects to gain 80 dollars per trade.

To increase their winning rates many professional day traders use technical indicators. Recently, systematic trading using deep learning has emerged as a powerful tool for predicting future stock prices [3]–[5]. In this study, a day trading strategy with a positive expectancy is proposed. To increase the win rate, the proposed trading strategy uses not only technical indicators (TIs) but also transformer neural network (TNN) and multiresolution analysis (MRA).

MRA was included in the proposed solution because many researchers reported that they got better performance when they combined MRA with their predictive model. For example, MRA resulted in better model performance when combined with each of ARIMA [6], descriptive statistical modeling [7], ANN [8], [9], RNN [10], CNN [11], GRU [12], LSTM [13], and stacked autoencoders [14].

TIs were also found to improve model performance by many researchers [5]. The problem is, there are more than 100 technical indicators and each technical indicator (TI) may have a number of parameters. Choosing the wrong combination of TIs or assigning a TI a wrong parameter value can degrade performance. So in this study, a systematic way of choosing TIs and their parameter values is presented.

There are many possible deep learning architectures. Choosing the wrong architecture can result in poor performance. For the proposed day trading strategy, several deep learning architectures were compared and the one that outperformed all of them was selected. The proposed deep learning model takes as input a dataset which consists of nine features, such as prices, volume, indices, and TIs. Some of these features are decomposed using empirical wavelet transform (EWT) before they are fed to the model. The model predicts the highest and the lowest stock prices of a given trading day. The proposed day trading strategy is based on these two predicted prices and will be explained in Section V-G. The proposed day trading strategy consists of eight trading systems; and unlike many of the existing systems, they were tested in different market conditions using ten randomly picked stocks listed in the Saudi stock market. They were compared with the buy-and-hold trading strategy and the experimental results show that five of the proposed systems showed positive expectancy consistently with all the ten stocks whereas the buy-and-hold strategy was inconsistent and resulted in losses in half of the stocks.

The main contributions of this work are:

1) A day trading strategy which combines TNN, TI, EWT MRA, and time2vec [15]. To the best of our

knowledge, this is the first study to do so.
2) The first consistently profitable day trading strategy which uses TNN for Saudi stock market.
3) A trading strategy based on a systematic way of choosing and combining TIs and their parameter values.
4) A study which presents the impact of profit-target on profitability.

The remainder of this paper is organized as follows. Section II gives background information relevant to the proposed solution. Section III presents related work. The proposed framework is explained in Section IV. Section V discusses the results and analysis of the proposed methodology, and Section VI is the conclusion.

II. Background

This section gives brief background information on some topics used in this study. The covered topics include deep learning models and wavelet transform.

A. Deep Learning

Deep learning is a subfield of machine learning based on deep neural networks (DNN). A DNN is essentially an artificial neural network (ANN) with more than three layers. These multiple layers give a DNN massive computing power and enables it to train on huge amounts of data with little human intervention, which makes it different from the other classical machine learning algorithms. Each DNN layer extracts certain information from the data and redirects the learned information to the next layer to perform another type of information extraction. This hierarchy of information extraction enables DNNs to perform better forecasting than the other classical machine learning algorithms. There are several DNN models but the most popular DNN models for time series data are, Recurrent Neural Network, Long Short-Term Memory, Gated Recurrent Unit, and transformers.

1) Recurrent neural network: A recurrent neural network (RNN) is a special type of ANN. However, unlike the standard feed-forward ANN, RNN networks have feedback connections which enables output from a previous step to be fed as input to the current step [16], [17]. RNNs also have the concept of memory that enables them to store limited information extracted from previous inputs which are then used to generate subsequent output. Having memory and feedback loop makes RNNs very popular for sequence data, such as time series, where one data point depends on previous data points.

2) Long Short-Term Memory (LSTM): Long short-term memory (LSTM) is a variant of RNN. It was proposed by Hochreiter and Schmidhuber [18] to resolve the vanishing and exploding gradient problems observed in the simple RNN, enabling it to capture longer-term dependencies.

LSTM architecture consists of a sequence of neurons and memory blocks known as cells, Fig. 1. The sequence of neurons form three gates, namely input gate, forget gate, and output gate. It uses these gates to control the flow of information to and from its neurons and to select the information it needs to discard or keep in its memory cells. The equations

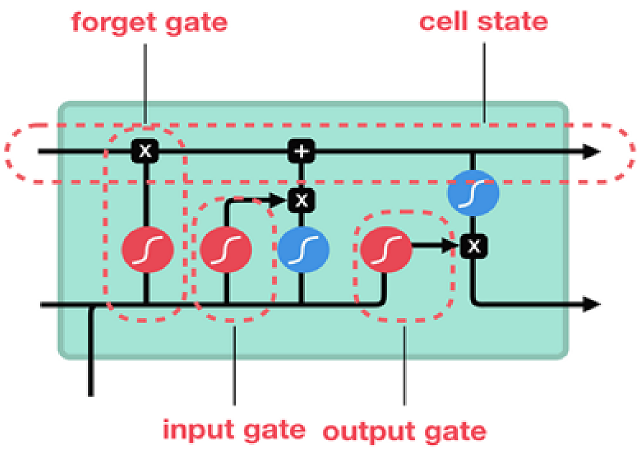

Fig. 1. LSTM architecture [19].

that are computed by the different neurons inside LSTM are as follows [20]:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + V_i c_{t-1}) \tag{2}$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + V_f c_{t-1}) \tag{3}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o c_t) \tag{4}$$
$$\tilde{c_t} = \tanh(W_c x_t + U_c h_{t-1}) \tag{5}$$
$$c_t = f_t^i \odot c_{t-1} + i_t \odot \tilde{c_t} \tag{6}$$
$$h_t = o_t \odot \tanh(c_t) \tag{7}$$

where, i_t, o_t, f_t and c_t denote the input-gate, the output-gate, the forget-gate, and the memory cells respectively. h_t represents a hidden state.

3) Gated Recurrent Unit (GRU): Gated recurrent unit (GRU) is also another variant of RNN [21]. Its structure is similar to LSTM but uses two gates, namely reset-gate and update-gate, to control the retention and flow of information (see Fig. 2). Its performance is comparable to that of LSTM but it is faster to train due to its fewer equations. The equations

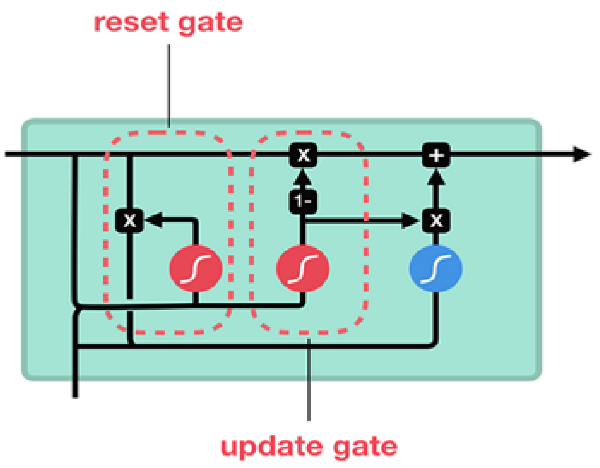

Fig. 2. GRU architecture [21].

computed by GRU are as follows [22]:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{8}$$
$$\tilde{h}_t = tanh(W_h x_t + U_h (r_t \odot h_{t-1}) + b_h) \tag{9}$$
$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{10}$$
$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{11}$$

where, r_t and z_t denote the reset-gate and update-gate, respectively. h_t and h_{t-1} represent the current and previous states, respectively.

RNN, LSTM, and GRU are incredibly slow because their training is difficult to parallelize. This is because inputs must be sequentially fed and the next step relies on the analysis of the previous step.

4) Transformer Neural Network (TNN): Transformer neural network (TNN) is a type of DNN which doesn't rely on recurrent connections [23]. Instead, it uses a mechanism known as self-attention which enables it to handle long-range dependencies and process input sequences in parallel for more efficient computation [23]. A typical TNN consists of an encoder and a decoder (see Fig. 3).

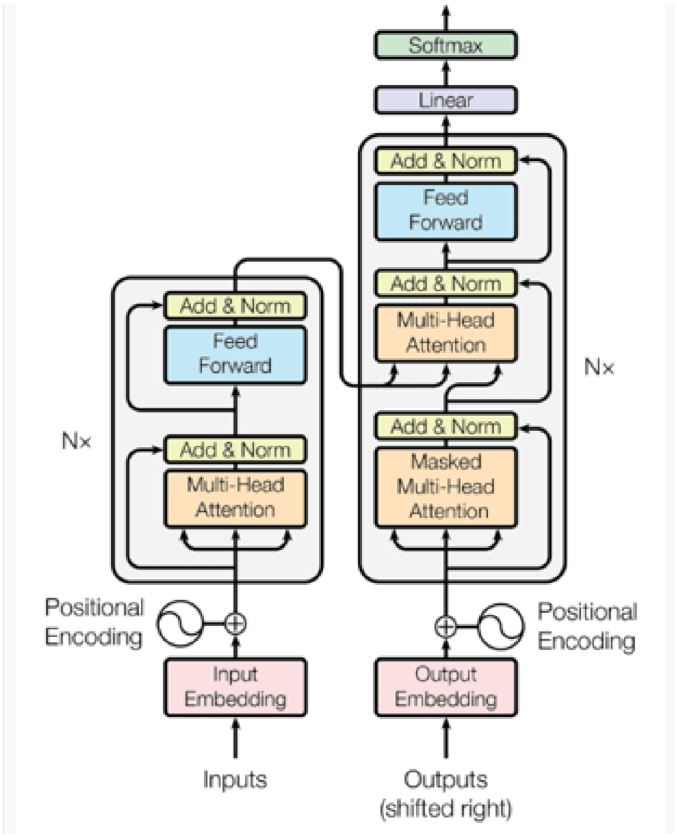

Fig. 3. TNN architecture [23].

The encoder is made up of multiple identical layers, and each layer consist of two sub-layers, namely a multi-head self-attention mechanism and a fully connected feedforward neural network (FFNN) [24]. To improve the performance and training stability of the encoder, each of the above mentioned sub-layers is followed by a residual connection and a normalization step. The input data is augmented with positional encoding [25] and processed through the stacked layers, sub-layers, and steps of the encoder.

A decoder is also made of multiple identical layers. The sub-layers of a decoder layer are similar to that of an encoder but has an additional sub-layer known as an encoder-decoder attention mechanism. This additional sub-layer enables the decoder to selectively focus on different parts of the encoded input sequence while generating the output. At each layer, the decoder processes each sequence with multi-head self-attention, position-wise FFNN, residual connections, and normalization.

The self-attention mechanism allows a TNN to prioritize the various input sequences according to their importance. The input sequence is linearly projected into multiple sets of queries, keys, and values, which are then used to compute attention scores. The scores are used to weigh the values, and the resulting weighted values are summed to produce the output of the self-attention layer. This process is repeated for each head, and the outputs are concatenated and linearly transformed to create the final output.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt(d_k}\right) \tag{12}$$

where, Q, K, and V are the query, key, and value matrices, respectively, and D_k is the dimension of K.

TNN uses multi-head attention layer to concatenate the attention weights of many single-head attention layers and then apply a non-linear transformation with a dense layer. Increasing the number of attention heads enables TNN to capture long-distance dependencies.

$$Multihead(Q, K, V) = Concat\left(h_1, h_2, \ldots, h_n\right) W^0 \tag{13}$$

where, $h_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right)$

TNN uses the position-wise FFNN to add non-linearity and to identify complex patterns in the input sequence. FFNN consists of two linear layers separated by a ReLU activation function. The independent application of this sub-layer to every part of the input sequence makes parallelism possible.

$$FFNN(x) = max\left(0, xW_1 + b_1\right) W_2 + b_2 \tag{14}$$

where, x is a sequence, and W_i and b_i are the weight matrix and the bias vector at layer i, respectively.

B. Wavelet Transform

Time-series data can be decomposed using Fourier or wavelet transforms. For analyzing non-stationary data, such as financial time series, wavelet transform has been found to outperform Fourier transform [7]. A wavelet transform is the representation of a function by wavelets [26], [27]. A wavelet is a waveform of a limited duration with an average value of zero, Fig. 4. It is a mathematical function with two basic parameters, namely scale (or dilation) and translation (location). Scale defines how squished or stretched a wavelet is and translation defines where the wavelet is located in time or space. A wavelet is mathematically defined as:

$$\psi_{s,u}(t) = \frac{1}{\sqrt{|s|}}\psi\left(\frac{t-u}{s}\right), \quad s,u \in \mathbb{R}, \quad s \neq 0 \quad (15)$$

where, s and u are the dilation and the translation parameters, respectively. These two basic parameters are related to frequency as defined for waves. When the value of parameter s is increased, a wavelet is squashed and it captures high-frequency information, and when it is decreased, the wavelet is stretched and it captures low-frequency information. The parameter u defines the translation of the wavelet. Increasing u will shift the wavelet to the right and decreasing it will shift the wavelet to the left.

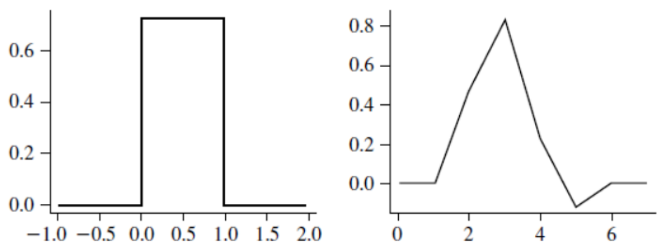

Fig. 4. Examples of wavelets.

The basic idea behind wavelet transform is to compute how much of a wavelet is in a signal for a particular scale and location. This is done by picking a wavelet of a particular scale, slide this wavelet across the entire signal, and at each time step, multiply the wavelet and the signal. The product of this multiplication gives us a coefficient for that wavelet scale at that time step. We then change the wavelet scale and repeat the process.

To best match a particular signal, there are a wide variety of prototype wavelets, called mother wavelets, to choose from. Popular examples of mother wavelets are Daubechies, Haar, Coiflets, Morlet, Symlets, Meyer, Mexican Hat and Biorthogonal [28]. A particular episode of wavelet transform uses one type of mother wavelet; the user decides which type and size to use depending on the characteristics of the signal to be analysed. Many time-series forecast applications use Daubechies [28]. After transformation of a signal using a particular mother wavelet, we end up with basis waveforms consisting of a series of daughter wavelets. The daughter wavelets are all compressed or expanded versions of their mother wavelet, and each daughter wavelet extends across a different part of the original signal. The important point is that each daughter wavelet is associated with a corresponding coefficient that specifies how much the daughter wavelet at that scale contributes to the raw signal at that location. It is these coefficients that contain the information relating to the original input signal.

The two major wavelet transforms in wavelet analysis are Continuous Wavelet Transform (CWT) and Discrete Wavelet Transform (DWT). CWTs operate over every possible scale and translation values; whereas DWTs use a finite set of wavelets defined at a particular set of scales and locations values.

DWT basis function at dyadic scaling m and time location n is given by, [29],

$$\psi_{m,n}(\mathrm{t}) = 2^{\frac{-m}{2}}\psi(2^{-m}\cdot t - n) \quad (16)$$

The inner product of a time-series data denoted by $x(t)$ and the basis function $\psi_{m,n}$, Eq. (17) returns the high frequency information, also known as the detailed coefficients, contained in the signal.

$$d_{m,n} = \sum_{t=0}^{N-1} x(t)\psi_{m,n}(t) \quad (17)$$

Decomposing a signal using mother wavelet can result in infinite number of basis functions to accurately represent the signal. In order to have a finite set of basis wavelet, an auxiliary function $\phi(t)$, known as a scaling function or father wavelet, is defined and associated with the mother wavelet to capture the rest of the signal. Eq. (18) is a definition of a scaling function at level m and translation n.

$$\phi_{m,n}(t) = 2^{\frac{-m}{2}}\phi(2^{-m}\cdot t - n) \quad (18)$$

The inner product of a signal $x(t)$ and $\phi_{m,n}$, as defined in Eq. (19), returns the low frequency information, also known as the approximation coefficients, contained in the signal.

$$a_{m,n} = \sum_{t=0}^{N-1} x(t)\phi_{m,n}(t) \quad (19)$$

An approximation of $x(t)$ at level m can be computed from $a_{m,n}$ as shown in Eq. (20).

$$x_m(t) = \sum_n a_{m,n}\phi_{m,n}(t) \quad (20)$$

Given $d_{m,n}$ at levels $1, 2, .., m_0$ and $a_{m_0,n}$, the final multiresolution representation of $x(t)$ can be computed as shown by Eq. (21).

$$x(t) = \textstyle\sum_n a_{m_0,n}\phi_{m_0,n}(t) + \sum_{m=1}^{m_0}\sum_n d_{m,n}\psi_{m,n}(t) \quad (21)$$

1) Empirical Wavelet Transform: Empirical wavelet transform (EWT) is a technique to decompose a signal using an adaptive subdivision scheme [30]. EWT starts by dividing the signal spectrum into N continuous segments where each segment can be defined as $\Lambda_n = [\omega_{n-1}, \omega_n]$, where each ω_n $(\omega_0 = 0 and \omega_N = \pi)$ denotes a boundary of a segment. The empirical wavelet is regarded as wavelet filters of each Λ_n

The empirical scaling function $\hat{\phi}_n(\omega)$ and the empirical wavelet function $\hat{\psi}_{\mathrm{n}}(\omega)$ are computed according to Eq. (22) and Eq. (23):

$$\hat{\phi}_n(\omega) = \begin{cases} 1 & \text{if } |\omega| \leq (1-\gamma)\omega_n \\ K & \text{if } (1-\gamma)\omega_n \leq |\omega| \leq (1+\gamma)\omega_n \\ 0 & \text{otherwise} \end{cases} \quad (22)$$

$$\hat{\psi}_n(\omega) = \begin{cases} 1 & \text{if } (1+\gamma)\omega_n \leq |\omega| \leq (1-\gamma)\omega_{n+1} \\ M & \text{if } (1-\gamma)\omega_{n+1} \leq |\omega| \leq (1+\gamma)\omega_{n+1} \\ N & \text{if } (1-\gamma)\omega_n \leq |\omega| \leq (1+\gamma)\omega_n \\ 0 & \text{otherwise} \end{cases} \quad (23)$$

where:
$K = \cos\left[\frac{\pi}{2}\beta\left(\frac{1}{2\gamma\omega_n}(|\omega| - (1-\gamma)\omega_n)\right)\right]$,
$M = \cos\left[\frac{\pi}{2}\beta\left(\frac{1}{2\gamma\omega_{n+1}}(|\omega| - (1-\gamma)\omega_{n+1})\right)\right]$,
$N = \sin\left[\frac{\pi}{2}\beta\left(\frac{1}{2\gamma\omega_n}(|\omega| - (1-\gamma)\omega_n)\right)\right]$,
$\beta(x) = x^4(35 - 84x + 70x^2 - 20x^3)$,
$\gamma \leq \gamma \leq \min_n\left[\frac{\omega_{n+1}-\omega_n}{\omega_{n+1}+\omega_n}\right]$, and
γ is restricted between 0 and 1 to insure $\hat{\phi}_n(\omega)$ and $\hat{\psi}_{\text{n}}(\omega)$ is a tight frame of $L^2(R)$.

III. Related Work

Publications related to stock price prediction using TNN started in the last few years, and as a result, existing trading systems based on TNN are very limited, and this section summarizes most, if not all, of them. This section also presents some existing work on TNN based stock price prediction techniques.

Malibari et al. [31] proposed a stock price prediction technique using TNN. Their proposed TNN architecture was influenced by a vision transformer (ViT) [32]. They used it to predict the next day closing values of four Saudi stock market indices, namely TASI, TBNI, TMTI, and TTSI. They used MAE, MSE, MAPE, and RMSE performances matrices and found out that their proposed model can predict closing prices with a probability higher than 90%. Stock price prediction using TNN and time2vec was proposed by Muhammad et al. [33]. Their model predicts the next day and the next week closing prices of eight stocks listed in Dhaka Stock Exchange. They used the closing prices of eight previous days to predict that of the next day and they used the closing prices of the previous eight weeks to predict the closing price of the next week. They used MAE and RMSE to measure the performance of their model. For the daily solution, the MAE was less than 0.083 and the RMSE was less than 0.11, and for the weekly, the MAE was less than 0.3 and the RMSE was less than 0.33.

A number of researchers compared the stock price prediction accuracy of TNN with other models. For example, Anass [34] compared the accuracy of LSTM and TNN in predicting the next day values of Nasdaq, S&P, and Dow. He used the value of trading day t to predict the value of trading day $t+1$. He compared them using accuracy, MAE, MSE, RMSE, and execution time. He found out that TNN consistently outperformed LSTM. Also the comparison of LSTM and TNN was done by Lin et al [35]. They compared the performance of TNN and LSTM to predict the next minute and then next day stock prices. They used the historical data of Shanghai Stock Index. The daily trading data spans from December 17, 2002 to December 17, 2022 and minute-level data spans from 9:30 on December 23, 2019 to 15:00 on December 23, 2022. They compared them using MAE and MSE and they reported that LSTM outperformed TNN consistently in all the measures. Similarly Saeed [36] proposed using the stock prices of the previous ten days to predict the next day price. He used the historical prices of Yahoo, Facebook, and JPMorgan from January 1, 2017, to September 2017. He compared the performance of TNN with ARIMA, LSTM, and Random Forest using MAE, RMSE, and MAPE. He reported that TNN consistently outperformed all of them. Performance comparison of TNN, ARIMA and LSTM using the average daily prices of eight stocks listed in the Brazilian Ibovespa was illustrated by Lorenzo et al. [37]. They used 2008 historical prices totaling 80 values per share. TNN outperformed both ARIMA and LSTM obtaining the lowest RMSE in 60% of the tests, followed by LSTM in 22% and, finally, ARIMA in 18%. Wang et al. [38] compared TNN with LSTM and Hidden Markov model (HMM) using historical prices of Shanghai and Shenzhen CSI new energy stock index from June 17, 2019 to June 17, 2022. They used MAE, RMSE, and MSE, R2 to compare them. They reported that TNN consistently outperformed both of them in all the above mentioned four performance measures. Stock price predicting model consisting of BiLSTM and MTRAN-TCN was proposed by Wang et al. [39]. BiLSTM is a bidirection LSTM, MTRAN is a modified TNN, and TCN is a time conventional network. They used BiLSTM to capture bidirectional information in sequences and TCN to identify sequence dependencies. They used five index stocks and 14 Shanghai and Shenzhen stocks and measured the performance of their proposed model using MAE, RMSE, and MSE, and R2. They reported that the combination of BiLSTM and MTRAN-TCN consistently outperformed any of its subset components with all the datasets and all the above mentioned four performance measures. TNN and LSTM were also compared using LOB (Limit Order Book) data of cryptocurrency by Bilokon and Qiu [40]. They compared them using three financial time series prediction tasks, namely LOB mid-price prediction, LOB mid-price difference prediction, and LOB mid-price movement prediction of cryptocurrency LOB data. They reported that TNN outperformed LSTM by a large margin in terms of the limited metrics for mid-price prediction; whereas LSTM outperformed TNN in the other two tasks. They concluded that LSTM-based models are generally better in financial time series prediction for electronic trading.

A limited number of researchers proposed one or more trading systems based on TNN. For example, Aman [41] compared two trading systems, one based on LSTM and the other on TNN. He used the data of four stocks listed in Nifty 50 of the Indian stock market. He collected data of 21 years ranging from January 1, 2000 to December 31, 2020. Using a sliding window of size four years, he divided the data into 17 overlapping windows. The data of each window was then split into two: the data of the first three years was used for training and that of the last year for testing. He took both long and short trades. The average daily returns of the trading system based on LSTM was 2.22%. For long trades, the TNN based system gave better returns than that of LSTM. For short trades, the TNN results were inconsistent. Four trading systems based on TNN, RNN, CNN, and LSTM were proposed by Wang et al. [42]. They used data of four global indices namely, the Shanghai and Shenzhen 300 Index (CSI 300) in China, the Standard & Poors 500 Index (S&P 500) in the US, the Nikkei 225 Index (N225) in Japan, and the Hang Seng Index (HSI) in Hong Kong. The collected data that spans 11 years, from January 1, 2010 to December 31, 2020. The trading strategy proposed is as follows: if the predicted value y_{t+1} is higher than the last observed value y_t, then a long position is entered; and if y_{t+1} is lower, then a short position is entered, other wise no position is taken. According to the performance measures MAE, MSE, and MAPE, TNN consistently outperformed the other three and gave better trading returns with all the above mentioned four global indices. A trading system based on TEANet, which is a model consisting of TNN and LSTM, was proposed by Zhang et al. [43]. Input to TEANet are tweet corpus and historical prices. The TNN component of TEANet takes as input tweets and feeds its output to LSTM. The LSTM also takes as input normalized historical prices. To evaluate the performance of TEANet, the researchers used a standard profit method and adopted a market simulation strategy proposed by Ding et al. [44]. The trading strategy is as follows: if TEANet predicts the price of a stock to rise, then a long position is entered and if it predicts the price to fall, then a short position is entered. The size of each position is $10,000. A long position is sold if a 2% profit is achieved, otherwise it is sold at the closing price at the end of the day. Similarly, a short position is sold if a 1% profit is achieved, otherwise it is sold at the closing price at the end of the day. The proposed trading system was tested using data of 44 trading days collected from 12 randomly selected stocks. The returns of TEANet were compared to the returns of a system known as CapTE [45]. TEANet outperformed CapTE and showed an average return of 22.31% compared to 20.18% of CapTE.

IV. Framework of the Proposed Trading Solution

The framework of the proposed trading solution can be summarized as follows:

1) Ten out of 223 stocks listed in the Saudi stock market are randomly selected. Then ten datasets, one from each selected stock, are collected. The selected stocks and their selection criteria are discussed in Section V-B.
2) From each dataset nine features are selected. These features are four daily stock prices, namely open (O), low (L), high (H), and close (C), the daily volume traded (V), the Saudi market index TASI (T), sector index (S), and two TIs, namely Bollinger Bands (BB), and average true range (ATR).
3) Six performance measure are selected and used in this study and they are presented in Section V-C.
4) Out of more than 100 TIs, 10 are selected based on literature survey. Then for each selected TI, its best parameter values are identified. These selected TIs are further filtered and combined based on their performance. The details of TIs selection, best parameter value identification, filtration, and combination are discussed in Section V-D.
5) The 10 datasets are preprocessed. One of the features is converted into log returns and decomposed using EWT, and the remaining eight features are scaled. Then all of them are reshaped and split into training and testing. The data preprocessing of the datasets is detailed in Section V-E and EWT is explained in Section II-B1.
6) Seven deep learning models are compared and the one with the best performance is selected for the proposed solution, Section V-F. The architecture of each of the seven compared models is shown in Fig. 5

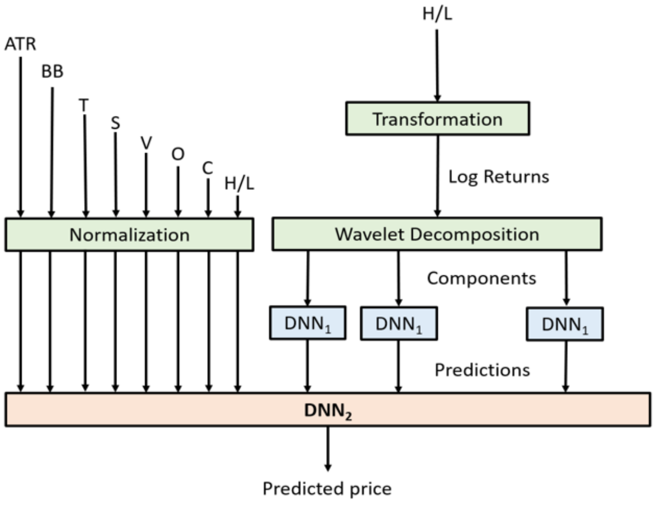

Fig. 5. Architecture of the proposed model.

7) The selected model is further optimized by choosing the best possible hyper-parameters. The list of the selected hyper-parameters and optimizers are presented in Section V-F.
8) A day trading strategy which consists of eight trading systems is proposed. The trading strategy is based on the open price and two predicted daily stock prices, namely the daily high and the daily low, Fig. 6. The details of the trading strategy is presented in Section V-G.

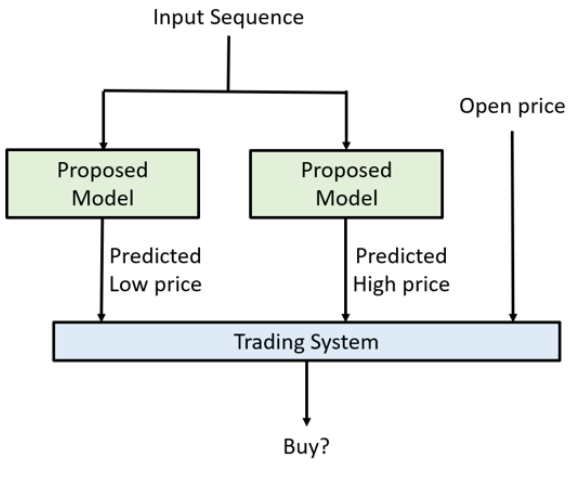

Fig. 6. Trading methodology.

9) At last, the proposed solution is evaluated, analyzed and compared, Section V-H.

V. Results and Analysis

To study the performance of the proposed trading strategy, six sets of experiments were conducted. The first set of experiments was done to choose the best possible deep learning model. The second set was done to identify the best possible hyperparameter values for the selected model. The third set was done to select the most relevant TIs. The fourth set was done to identify the best possible parameter values for each selected TI. The fifth set was done to choose the best possible combination of TIs. The last set of experiments was done to study the performance of the proposed trading strategy. But before the results and analysis of the above mentioned experiments are presented, let us discuss the experimental environment, the datasets, and the performance measures used in this study.

A. Experimental Machines and Tools

All the experiments were conducted on a Windows 10 machine with Intel(R) Core(TM) i7-7700HQ CPU @ 2.80 and 2.81 GHz, and a 16 GB RAM.

The datasets were downloaded using TickerChart [46] and preprocessed using Amibroker Formula Language (AFL) [47] , Python, and Matlab. The deep learning models were implemented in Python using Keras open-source package with TensorFlow back-end [48]. EWT was implemented using EWT MATLAB toolbox.

B. Dataset Selection, Collection, and Features

The experimental datasets were collected from 10 out of the 223 stocks listed in the Saudi Stock exchange. These stocks are Bahri, Al-Rajhi, STC, Maaden, Tawuniya, Jarir, Jabel Omer, Budget, Sharqiya, and Mouwasat. These datasets were chosen because they exhibit different characteristics such as, they belong to different sectors, they have different number of free shares, they have been in the market since 2009 so they have enough data for training and testing, and for bull, bear, and side way markets, they are liquid enough to minimize errors due to slippage when entering and exiting positions, and so on. The datasets were downloaded using TickerChart . Table I shows some characteristics of the datasets.

TABLE I. THE DATASETS

Dataset	Stock	Symbol	Sector	Free Shares (in Millions)	Range	Standard Deviation
D1	Alrajhi	1120	Banks	3908.1	18.17 - 117.40	19.1
D2	Maaden	1211	Materials	807.7	4.87 - 57.76	9
D3	Mouwasat	4002	Healthcare	65.0	6.88 - 129.7	28.4
D4	Bahri	4030	Energy	380.4	5.71 - 30.28	5.15
D5	Jarir	4190	Retail	110.7	4.3 - 22.5	4.69
D6	Jabel Omar	4250	Real estate	929.2	11.0 - 88.75	18.85
D7	Budget	4260	Transportation	71.1	9.18 - 54.5	11.44
D8	Sharqiya	6060	Food & Beverages	7.5	8.49 - 52.59	8.49
D9	STC	7010	Telecommunication	1800.0	13.2 - 55.92	11
D10	Tawuniya	8010	Insurance	92.7	22.48 - 106.0	22.86

Each dataset contains nine features. These features are based on the daily time frame and they are: the daily open (O), high (H), low(L), and close (C) prices of a stock, the daily Volume (V) of shares traded, the sector index value (S), the TASI (T) which is the Saudi stock market index value, and two technical indicators, namely the Bollinger Bands (BB) and the average true range (ATR) which will be discussed in Section V-D.

Each dataset contains daily data from January 1, 2010 to December 31, 2022, which consists of 3244 observations.

TABLE II. THE FEATURES

Symbol	Name	Description
O	Open	Daily open price
L	Low	Daily Lowest price
H	High	Daily highiest price
C	Close	Daily close price
V	Volume	Daily traded shares
T	TASI	Saudi market Index
S	stock specific	Sector index
BB	Bollinger Bands	TI
ATR	Average True Range	TI

Table II contains the description of the above mentioned nine features.

According the proposed trading strategy, an open trade is closed when its stock price reaches a profit-target or a stop-loss, otherwise it is closed at the end of the trading day. A trade is a winner, if the intraday price of its stock reaches the profit-target before it hits the stop-loss; and it is a loser if the intraday price reaches first the stop-loss. The data in the above ten datasets is based on daily time frames or price-bars, meaning, the values of O, L, H, and C are daily prices. They don't include prices of lower time frames, such as hourly, 15-minute or 5-minute. From daily price-bars, it is impossible to know if a trade entered on day t is a winner or a loser if the price-bar covers both the predicted high and the predicted low. To avoid mistakes that can be created by such price-bars, supplementary datasets were collected from each of the above mentioned stocks. These datasets contain the highest and the lowest stock prices of 5-minute time frames from January 1, 2020 to December 31, 2022. These datasets are only used during trading and not used for training or testing the proposed model. If a 5-minute price-bar covers both the profit-target and the stop-loss of a trade, then the trade is ignored. Fortunately, the number of such 5-minute price-bars is so insignificant that there is no need to use price-bars of a lesser time frame.

C. Performance Measures

To study the performance of the proposed solution, six performance measures, namely, Mean Square Error (MSE), Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), coefficient of determination (R^2), and Expectancy (Φ) were used. These measures were chosen because they are frequently used in related works. A good model must have RMSE, MAE, and MAPE close to 0, R^2 close to one, and a positive Expectancy. Expectancy is defined in Eq. (1), but the other five measures are mathematically defined as follows:

$$MSE = \frac{1}{n}\sum_{t=0}^{n-1}(y_t - \hat{y}_t)^2 \tag{24}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=0}^{n-1}(y_t - \hat{y}_t)^2} \tag{25}$$

$$MAE = \frac{1}{n}\sum_{t=0}^{n-1}|y_t - \hat{y}_t| \tag{26}$$

$$MAPE = \frac{\sum_{t=0}^{n-1} |\frac{y_t - \hat{y}_t}{y_t}|}{n} \times 100\% \tag{27}$$

$$R^2 = 1 - \frac{\sum_{t=0}^{n-1} (y_t - \hat{y}_t)^2}{\sum_{t=0}^{n-1} (y_t - \bar{y})^2} \tag{28}$$

where, y_t and $\hat{y}_t$ represent the actual and forecast values at step t for $0 \leq t < n$, respectively, and $\bar{y} = \sum_{t=0}^{n-1} y_t/n$.

D. Selecting TIs, TI Parameters, and TI Combination

There are more than 100 TIs and each TI has a number of parameters. Some TI parameters have infinite number of possible values. Hence, selecting the best TIs with their best parameter values is very challenging. In this study, a four step filtering process was adopted to choose a set of TIs and their parameter values. In Step 1, based on literature survey, the top 10 most frequently used TIs were identified (see Table III).

TABLE III. MOST FREQUENTLY USED TECHNICAL INDICATORS IN THE SURVEYED LITERATURE

TI	Frequency
MA	35
RSI	34
Williams R%	28
Stochastic %K	25
MACD	22
ROC	19
Momentum	16
Bollinger Bands	15
ATR	14
CCI	14

In Step 2, the most frequently used TI parameter values are identified. Then each TI and each of its most frequently used parameters value were used to predict the next day high and low prices using the proposed model which will be defined shortly. If a TI and its best parameter doesn't improve the prediction performance of the proposed architecture, then it is dropped. This step is done after normalizing the TI values using min-max normalization. In Step 3, The least relevant TIs are identified and eliminated. This is done using recursive feature elimination [49]. Table III shows the TIs and the parameter values that were selected after the above three steps.

TABLE IV. SELECTED TIS AND THEIR SELECTED PARAMETER VALUES

TI	Name	Parameters
STO	Stochastic	$15, 3, 3$
MACD	Moving Average Convergence Divergence	$12, 26, 9$
RSI	Relative Stength Index	14
BB	Bollinger Bands	$20, 2$
ATR	Average True Range	14

In Step 4, the selected TIs were combined in all the possible ways and the best combination was selected (see Table IV). Since there are five selected TIs, the maximum number of combinations is 32, including the one with no TIs used. Table V shows the performance of each set of TI combination. Each measurement is based on the average of the predicted high and low prices of each stock.

TABLE V. PERFORMANCE MEASURES OF TI COMBINATIONS

ATR	MACD	BB	STO	RSI	E
0	0	0	0	0	2.31
0	0	0	0	1	1.38
0	0	0	1	0	1.74
0	0	0	1	1	1.66
0	0	1	0	0	1.59
0	0	1	0	1	2.31
0	0	1	1	0	2.64
0	0	1	1	1	1.24
0	1	0	0	0	1.51
0	1	0	0	1	1.55
0	1	0	1	0	2.68
0	1	0	1	1	1.39
0	1	1	0	0	3.98
0	1	1	0	1	1.42
0	1	1	1	0	2.09
0	1	1	1	1	1.16
1	0	0	0	0	2.12
1	0	0	0	1	1.12
1	0	0	1	0	1.68
1	0	0	1	1	1.47
1	0	1	0	0	1.09
1	0	1	0	1	1.31
1	0	1	1	0	2.21
1	0	1	1	1	1.20
1	1	0	0	0	1.94
1	1	0	0	1	2.28
1	1	0	1	0	4.98
1	1	0	1	1	1.88
1	1	1	0	0	2.17
1	1	1	0	1	1.99
1	1	1	1	0	2.11
1	1	1	1	1	2.18

Each row of the table represents a unique combination of TIs. In each of the first five columns, a 0 value represents the absence of the corresponding TI from the combination, and a 1 represents its presence. For example, in the row before the last, all TIs except RSI were in the combination. The last column, E, was computed using Eq. (29).

$$E_i = \widehat{1 - r_i^2} + \widehat{rmse}_i + \widehat{mae}_i + \widehat{mape}_i \tag{29}$$

where, i represents i^{th} row of Table V, r_i^2, $rmse_i$, mae_i, and $mape_i$ are the r^2, RMSE, MAE, and MAPE values of row i, respectively, and $\widehat{x}$ represents the normalized value of x using min-max, Equation 30. The best combination is the one with the lowest E. According the results in Table V, the lowest E is 1.09 and corresponds to the combination of BB and ATR; and is the combination that was selected for the proposed model.

Nearly all the normalization operations in this study were done using min-max. It is one of the most popular techniques and is frequently used by similar studies. If $X = x_1, x_2, \ldots x_n$, then scaling x_i using min-max is mathematically defined as:

$$\acute{x_i} = \frac{x_i - min(X)}{max(X) - min(X)} \tag{30}$$

where, x_i is the i^{th} value in X, $\acute{x_i}$ is the normalized value of x_i, and $min(x)$ and $max(x)$ are the lowest and the highest values in X, respectively.

E. Data Preprocessing

Data processing is done in five phases. In Phase 1, L or H is converted to log returns. If the proposed model is going to predict the next day highest price, then H is converted to log returns and if it is going to predict the next day lowest price, then L is converted to log returns. If $X = x_1, x_2, \dots X_n$ is a time series data, then the log return of x_t is mathematically defined as:

$$\acute{x_t} = log(x_t/x_{t-1}) \tag{31}$$

where, x_t is the value at time t, x_{t-1} is the value at time $t-1$, and $\acute{x_t}$ is the log return of x_t. Converting stock prices into log returns increases the stationarity of the dataset.

In Phase 2, the log returns generated in Phase 1 are decomposed using the EWT algorithm proposed by Jérôme Gilles [30] as explained in Section II-B1. Decomposition results in building better forecasting models, because it enables the identification and the removal of noisy and irrelevant parts of a time series data. EWT was chosen because it is an adaptive wavelet subdivision scheme which performs wavelet decomposition without prior information about the data, produces a small number of coefficients to pack the signal information, and results in a higher resolution of time-frequency which simplifies the analysis of time-series data.

In Phase 3, all the remaining eight features are scaled between 0 and 1 using the min-max formula. This phase may not be important because TNNs can handle unscaled data.

In Phase 4, each one-dimensional time series data of size N observations is reshaped into a two dimensional array of size $N - k$ by $k + 1$ using a sliding window of size $K + 1$. This is done to predict the stock price on day t, p_t, using $p_{t-1}, p_{t-2}, \dots, p_{t-k}$. Finding the best possible value of k is explained in Section V-F.

In Phase 5, the data is split into training and testing. The data from January 1, 2010 to December 31, 2019 was used for training and the data from January 1, 2020 until December 31, 2022 was used for testing.

TABLE VI. The Performances of the Seven Deep Learning Models

Model	RMSE	MAE	MAPE	r^2	E
VLSTM	77.13597	46.23983	0.07594	0.97639	3.506
SLSTM	14.06473	10.01123	0.04511	0.99002	0.087
BLSTM	14.38421	19.35125	0.05318	0.97971	1.099
GRU	82.98782	51.12156	0.07823	0.97214	4.000
SGRU	14.17115	10.16732	0.04569	0.99013	0.104
BGRU	15.66173	18.72891	0.05408	0.98895	0.643
TNN	13.86754	9.88712	0.04436	0.99115	0.000

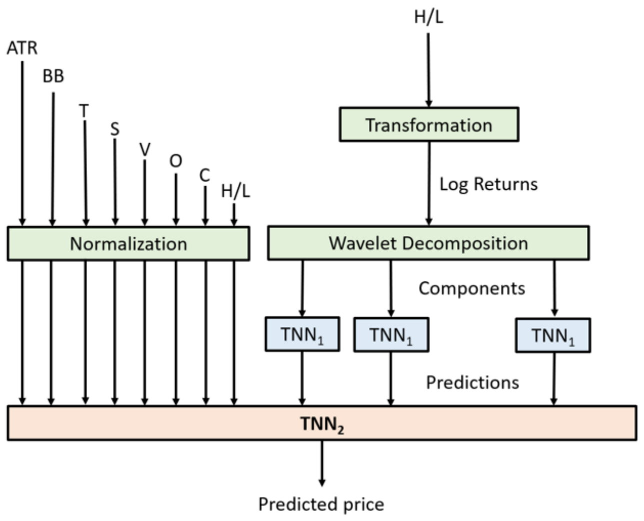

Fig. 7. The architecture of the proposed deep learning model.

F. Deep Learning Model Selection

To find the best deep learning model for the proposed solution, seven models namely, simple or vanilla LSTM (VLSTM), Stacked LSTM (SLSTM), Bidirectional LSTM (BLSTM) [50], GRU, Stacked GRU (SGRU), and Bidirectional GRU (BGRU) were compared. Fig. 5 shows the architecture of the proposed model that was selected after trying many other architectures. A number of experiments were conducted to choose the deep learning model that can best predict the next day high and low stock prices. The data was reshaped so that the previous 16 days are used to predict the price of the next day. Each model was experimented using different hyperparameter values and its performance was measured using RMSE, MAE, MAPE, and R^2. Table VI shows the average performance measures of all the experiments.

To choose the best performing model for the proposed solution, a normalized sum, E, was used, Eq. (29). The model with the lowest E has the best performance and thus selected for the proposed solution. As shown in Table VI, TNN has the lowest E, hence selected as the model of the proposed solution. Fig. 7 shows the architecture of the proposed model. As can be seen from the figure, the architecture contains two levels of TNNs. The TNNs in the first level are labeled as TNN_1 and the one in the second level is labeled as TNN_2. Since each TNN in the proposed model is used for the prediction of a continuous value, it doesn't need a decoder. It only uses encoders similar to the architecture of the popular BERT [51]. It also includes some dense layer, dropout layers, a global average pooling layer, and an output layer which consists of a single neuron that emits the predicted continuous value.

To train the model for the prediction of the next day high price, the following steps were taken. H is first converted into log returns and then the log returns are decomposed into multiple levels using EWT. Each level is then reshaped into input sequences of size 16 each, encoded with time2vec, and fed into a Level 1 TNN. Similarly, each of the input features O, L, C, V, S, T, BB, and ATR are scaled between 0 and 1 using using the min-max. Then these scaled values and the output of all Level 1 TNNs are reshaped into input sequences of size 16, encoded with time2vec, and fed to the Level 2 TNN. The output of the Level 2 TNN is the predicted value. The same steps were taken to train the model for the prediction of the next day low price, except L and H exchange places in the model. A TNN requires a notion of time when processing stock prices. Without time encoding, a TNN will be oblivious to the temporal order of stock prices. In order to overcome this, the proposed model uses Time2Vec [15], a time encoding layer. Many experiments were conducted to arrive at the proposed model and Table VII summarizes the selected parameters and hyperparameters.

TABLE VII. THE PROPOSED MODEL HYPERPARAMETERS AND OTHER PARAMETERS

Parameter	TNN_1	TNN_2
time embedding	time2vec	time2vec
sequence size	16	16
number of heads	3	4
ff dimension	32	64
encoder blocks	3	4
dropout rate	0.2	0.2
loss function	MSE	MSE
optimizer	Adam	Adam
learning rate	0.0001	0.0001
epochs	150	200
batch size	64	64

G. Proposed Trading Strategy

As was mentioned before, day trading is a business of probability. When a day trader enters a position, he is not sure if it will be a winner or a loser. Most consistently profitable traders have a win rate between 40% and 60%. They are consistently profitable because their average win is much higher than their average loss. They do that by cutting their losers short and letting their winners run. They decide the amount they will risk per trade, denoted as R, before they enter a position. Most of them limit R to be less than 2% of their trading capital. This way, they can survive a number of consecutive losers.

The proposed trading strategy uses stop-loss and is based on two predicted values, namely, $\hat{h_t}$ and $\hat{l_t}$, where, $\hat{h_t}$ and $\hat{l_t}$ are the predicted highest and lowest prices of a stock on a trading day t, respectively. These predictions are based on the proposed deep learning model, shown in Fig. 7, and are computed on the morning of the trading day and before the market opens.

The proposed trading strategy consist of four buying and three selling rules. The four buying rules are:

1) $B == o_t + \psi$, where, B is the buying price per share, o_t is the open price on day t, and ψ is the minimum allowed bid-ask spread value.
2) The risk (R) per trade must be no more than 1% of the total capital. $R = B - \hat{l_t} - \psi * N$, where N is the number of shares bought. For example, for a trading capital of 10000 dollars, R must be no more than 100 dollars.
3) $\frac{\hat{h_t} - B}{\times} N \geq n \times R$ for n in $(1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5)$.
4) $n \times R > F$, where F is amount of fees and commissions paid to enter and exit a position.

The first rule is a conditional buy and is needed to make sure that a position is entered only when the stock price is trending towards the profit-target $\hat{h_t}$. The second rule limits the maximum risk taken on a trade. The third rule is there to make sure that the potential reward (profit) of a trade is significantly higher than the potential risk taken. The fourth rule is needed to ensure that the potential profit of a trade exceeds the fees paid to enter and exit the trade. When the above four rules are meet, a 10,000 Saudi riyals position is entered.

An open position is closed when any of the following three conditions is true.

1) If the intraday loss on a trade reaches R. This is done by using a predefined stop-loss price.
2) If the intraday profit on a trade reaches a prespecified profit-target, $n \times R$, for n in $(1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5)$.
3) If none of the above conditions is met, then the position is sold at the closing price of the day.

To study the performance of the proposed solution, eight trading systems were created. These systems satisfy the above mentioned buying and selling conditions but they differ in their profit-target. Table VIII shows these trading systems and their corresponding profit-targets.

TABLE VIII. THE EIGHT TRADING SYSTEMS OF THE PROPOSED TRADING STRATEGY

Trading System	TS1	TS2	TS3	TS4	TS5	TS6	TS7	TS8
Profit-target	1.5R	2R	2.5R	3R	3.5R	4R	4.5R	5R

H. Trading Results and Analysis

A number of experiments were conducted to study the profitability of each of the eight trading systems. The trades included in this study cover three year period, from January 1, 2020 to December 31, 2022, a total of 749 trading days. Each trading system was used with each dataset and the stats, such as the number of trades executed, the number of winning trades, the average profit of a wining trade, the average loss of a losing trade, etc was collected. Table IX shows some of the stats obtained by trading Alrajhi shares. The table shows that 138 positions were entered using TS1. As was mentioned before, the size of each position is 10,000 riyals. 90 of the 138 positions were winners and the rest were losers. A breakeven position is considered to be a loser because of the fees paid and the resources wasted to execute the position. The average profit per a winning trade is 140.23 riyals and the average loss is 63.89 riyals. The gross and net profits of all the positions are 12620 and 4999 riyals. The amount of fees and commissions paid to execute the trades was 4554 riyals. The expectancy, Φ, of trading Alrajhi stocks using TS1 is 36.23 riyals. The table also shows the stats of the other seven trading systems. TS2 is more profitable and TS3 has the best expectancy. TS2 was more profitable than TS3 because more trades were executed using TS2. The results also show that five of the eight trading systems were winners and the rest were losers. TS8 was the worst loser with 36.24% loss. The table also shows that the fees paid to execute trades are significant, between 30% and 61% of the gross profits.

TABLE IX. TRADING STATS OF ALRAJHI SHARES

Trading System	TS1	TS2	TS3	TS4	TS5	TS6	TS7	TS8
positions entered	138	130	117	106	105	99	91	82
Winners	90	78	69	58	46	40	35	29
Average win	140.23	171.19	183.32	191.88	208.83	201.15	178.02	151
Average loss	63.89	70.71	74.66	77.84	80.23	82.34	83.93	87.64
Gross profit	12620	13352	12649	11129	9606	8046	6230	4379
Loss	3066	3676	3583	3736	4733	4858	4700	4644
Fees	4554	4290	3861	3498	3465	3267	3003	2706
Net profit	4999	5385	5204	3894	1407	-79	-1472	-2971
Φ	36.23	41.43	44.48	36.74	13.41	-0.80	-16.18	-36.24

The percentage profits and losses obtained by trading Alrajhi shares are shown in Fig. 8. The figure shows that TS6, TS7 and TS8 resulted in losses. This is mainly due to the following two reasons. First, there are fewer trades with $4R$ or more profit-targets; and second, the probability of hitting the stop-loss before the profit-target is higher when the profit-target is greater than or equal to $4R$.

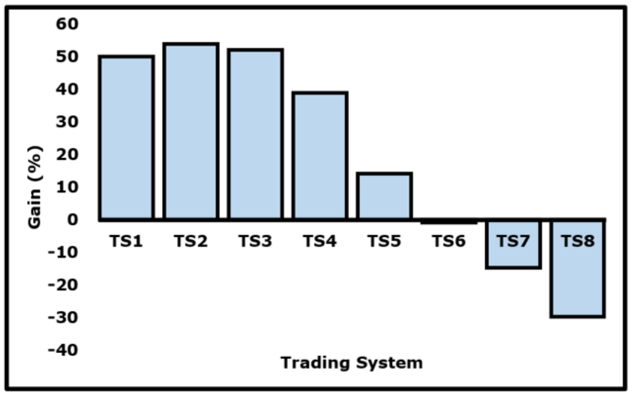

Fig. 8. Percentage profits and losses obtained by trading Alrajhi stock.

Stats, similar to that of Alrajhi stock trades, were also collected from the trades of the other nine stocks. Since these stats are too large in number and it is unnecessary to list them all, only their summaries are presented and discussed. Tables X shows percentage of profits and losses obtained by the proposed trading systems. As can be seen from the table, trading systems TS1, TS2, TS3, TS4, and TS5 were consistently profitable. TS7 and TS8 were consistent losers and TS6 showed inconsistent results. The results of TS6, TS7, and TS8 are inconsistent and poor because there are fewer trades with $4R$ or more profit-targets, and the probability of hitting the stop-loss before hitting the profit-target is higher.

TABLE X. PERCENTAGE OF PROFITS AND LOSSES OBTAINED BY THE PROPOSED TRADING SYSTEMS

Trading System	TS1	TS2	TS3	TS4	TS5	TS6	TS7	TS8
D2	50.11	48.72	46.57	37.97	22.84	10.11	-8.19	-23.92
D3	53.37	51.10	48.38	28.71	22.10	13.58	-1.0	-16.58
D4	48.48	54.24	44.92	23.18	22.95	20.74	-3.80	-23.83
D5	42.30	41.14	42.24	40.94	22.81	13.09	-1.20	-17.32
D6	28.11	30.10	30.79	27.45	17.20	2.19	-16.30	-45.25
D7	42.01	43.76	39.77	28.94	15.20	-12.46	-16.96	-38.32
D8	54.37	56.60	57.10	46.08	37.54	19.90	-13.46	-41.59
D9	48.52	52.81	46.65	29.66	18.96	15.91	-7.38	-16.36
D10	59.44	61.11	63.86	51.00	26.51	5.23	-12.05	-42.89

Fig. 9 shows expectancies obtained by trading Alrajhi shares. Again the trading systems with profit-targets between 1.5R and 3.5R showed positive expectancy and the rest were negative. TS3 showed the best expectancy of 44.48 riyals and TS8 was the worst with -36.24. The expectancies obtained by the other nine stocks are listed in Table XI. As can be seen from the table, the expectancies of all the trading systems with profit-targets of less than 4R were positive. TS6 results were inconsistent and TS7 and TS8 consistently resulted in negative expectancies. The negative results of the trading systems with higher profit-targets is due to the two reasons that were discussed before. The table also shows that TS2 has the highest expectancy when trading the stocks of D4, D7, and D9; TS3 has the highest expectancy when trading D1, D2, D3, D5, D6, D8, and D10. On the average, TS3 resulted in the best expectancy with 31.35 riyals.

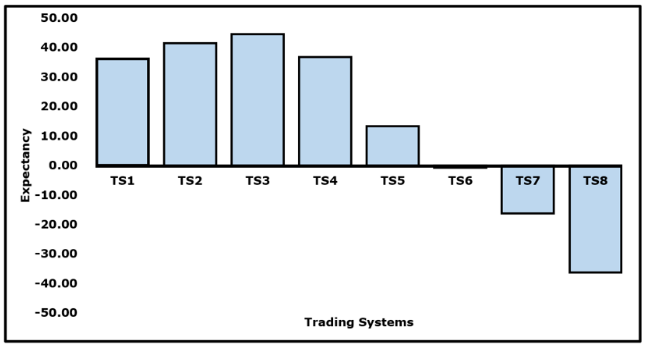

Fig. 9. Expectancy obtained by trading Alrajhi shares.

TABLE XI. EXPECTANCY OF THE PROPOSED TRADING SYSTEMS

Stock	TS1	TS2	TS3	TS4	TS5	TS6	TS7	TS8
D2	16.59	17.91	19.09	17.18	10.93	5.40	-4.76	-23.92
D3	18.53	18.93	20.07	12.54	10.28	6.56	-0.5	-8.87
D4	35.39	42.71	36.82	20.33	21.65	20.53	-3.92	-26.19
D5	38.11	38.45	44.46	46.00	26.53	15.59	-1.51	-22.50
D6	13.85	16.01	18.00	16.95	11.95	1.62	-12.44	-37.09
D7	19.82	22.44	22.22	17.43	9.81	-8.96	-13.90	-34.22
D8	18.62	21.77	25.15	22.26	19.97	10.76	-7.92	-25.83
D9	44.11	52.29	48.10	35.31	22.84	21.50	-9.84	-24.78
D10	27.52	31.34	35.09	29.83	15.97	3.51	-8.86	-33.25
Average	26.88	30.33	31.35	25.46	16.33	7.57	-7.57	-27.29

Each of the eight trading systems was compared with the buy-and-hold trading strategy. Tables XII shows profits and losses obtained by the buy-and-hold strategy. The best winning trade gave a profit of 192.81%, whereas the worst losing trade had a loss of -39.71% (see Fig. 10). Consistency in trading is very important for a trader. The buy-and-hold strategy has inconsistent results. Half of the positions taken were losers. Unlike the buy-and-hold strategy, TS1, TS2, TS3, TS4, and TS5 were consistently profitable. Also TS1, TS2, and TS3 showed significantly better average profits than the buy-and-hold strategy, (see Fig. 11). In summary, TS1, TS2, and TS3 showed consistency, better expectancy, and significantly better average profits than the buy-and-hold strategy.

TABLE XII. PROFITS AND LOSSES OBTAINED BY THE BUY-AND-HOLD STRATEGY

Stock	Open price	Last price	PnL	PnL (%)
D1	40.82	75.2	8389.34	83.89
D2	14.72	43.15	19280.86	192.81
D3	44	104.5	13717.00	137.17
D4	21.34	19.67	-815.57	-8.16
D5	16.54	15	-964.08	-9.64
D6	27.25	16.52	-3970.61	-39.71
D7	36.35	45.65	2525.46	25.25
D8	19.7	18.9	-439.09	-4.39
D9	40.64	36.6	-1027.09	-10.27
D10	76.4	80.5	503.65	5.04

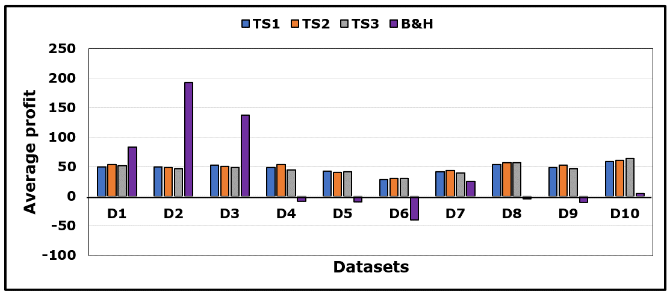

Fig. 10. Inconsistent profits of the buy-and-hold strategy.

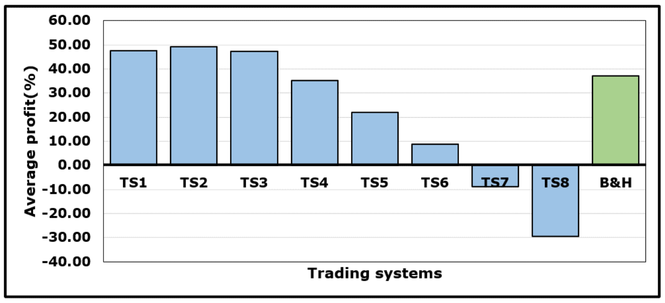

Fig. 11. Average profits of the proposed trading systems and the buy-and-hold trading strategy.

VI. Conclusion

In this study, a trading strategy based on two predicted stock prices using TNN, TIs, MRA, and time2vec was investigated. The proposed trading strategy consists of eight day trading systems, each with a different profit-target $n \times R$, where R is the risk taken per trade and for n in $(1.5, 2, 2.5 \ldots 5)$. To enter a position, the strategy requires four conditions to be simultaneously true, and to close a position, any of three conditions must be true. To choose the best deep learning model, seven architectures were investigated. TNN gave the best performance and hence it was selected. To improve the accuracy of the proposed model, some of the raw stock prices were converted to log returns and decomposed using MRA. To further improve its accuracy, a number of TIs were carefully analyzed and their best possible parameter values identified. Then the selected TIs were combined and the combination of TIs with the best performance was used. A number of experiments were also conducted to select the best TNN hyper-parameters such as, number of encoders, heads, epochs, learning rate, etc. Deep learning models are fast to overfit and to prevent that dropout layers were used.

The proposed trading strategy was tested using the data of ten randomly selected stocks listed in the Saudi Stock Exchange. The experimental results showed that trading systems with profit-target between $1.5R$ and $3.5R$ showed consistent profits. Those with profit-targets of $4R$ or more can result in losses. This is mainly due to the following two reasons: first, there are fewer trades with $4R$ or more profit-targets; and second, the probability of hitting the stop-loss before the profit-target is higher when the profit-target is greater than or equal to $4R$. The eight trading strategies were also compared with the buy-and-hold strategy. On the average, trading systems TS1, TS2, and TS3 outperformed the buy-and-hold strategy. Another weakness of the buy-and-hold strategy is its inconsistency. For one stock it gave a profit of 192.81%, and for another stock it resulted in a loss of -39.71%. Traders prefer trading systems with consistent results, such as TS1, TS2, and TS3, hence recommended.

There are four main ideas that can enhance the profitability of the proposed work and are planned as future works. First, to find a technique that can predict whether an intraday price will first hit the profit-target or the stop-loss. This can significantly enhance the profitability of the proposed strategy because most of the losing trades were due to the intraday price reaching the stop-loss before the profit-target. Second, to investigate different buying points instead of always buying using the open price. Third, day trading incurs significant amount of fees and commissions. As can be seen from the posted results, between 30% and 61% of the gross profits were paid as fees. To reduce the amount of fees paid, swing and positional trading strategies can be investigated. They require fewer trades and thus less fees. Fourth, to study the impact of price volatility, market and sector trends, news, and sentiments on the profitability of the proposed trading strategy.

Acknowledgment

The author would like to thank King Fahd University of Petroleum and Minerals (KFUPM), Saudi Arabia, and KFUPM Interdisciplinary Research Center for Intelligent Secure Systems for the support during this work.

References

[1] N. Zadeh, “Financial competitions,” https://financial-competitions.com/, 2024, last accessed 02 February, 2024.

[2] R. Moglen and G. Gajjala, “+805% trading champion of 2023 reveals his powerful day trading setups,” https://www.youtube.com/watch?v=10pHBNVi4Jc, 2024, last accessed 02 February, 2024.

[3] A. Thakkar and K. Chaudhari, “A comprehensive survey on deep neural networks for stock market: The need, challenges, and future directions,” *Expert Systems with Applications*, vol. 177, p. 114800, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417421002414

[4] Z. Hu, Y. Zhao, and M. Khushi, “A survey of forex and stock price prediction using deep learning,” *Applied System Innovation*, vol. 4, no. 1, 2021. [Online]. Available: https://www.mdpi.com/2571-5577/4/1/9

[5] K. A. Althelaya, E.-S. M. El-Alfy, and S. Mohammed, “Evaluation of bidirectional lstm for short-and long-term stock market prediction,” in *2018 9th International Conference on Information and Communication Systems (ICICS)*, 2018, pp. 151–156.

[6] M. T. Ismail and A. Dghais, “Multiresolution analysis of bursa malaysia klci time series,” in *AIP Conference Proceedings*, vol. 1847, 05 2017, p. 020020.

[7] D. K. Kılıc and O. Ugur, “Multiresolution analysis of s & p500 time series,” *Annals of Operations Research*, vol. 260, no. 1-2, pp. 197–216, 2018.

[8] S. Bekiros and M. Marcellino, “The multiscale causal dynamics of foreign exchange markets,” *Journal of International Money and Finance*, vol. 33, pp. 282–305, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0261560612002069

[9] B.-L. Zhang, R. Coggins, M. Jabri, D. Dersch, and B. Flower, “Multiresolution forecasting for futures trading using wavelet decompositions,” *IEEE Transactions on Neural Networks*, vol. 12, no. 4, pp. 765–775, 2001.

[10] A. Aussem and F. Murtagh, “Combining neural network forecasts on wavelet-transformed time series,” *Connection Science*, vol. 9, 03 1997.

[11] L. Di Persio and O. Honchar, "Artificial neural networks architectures for stock price prediction: Comparisons and applications," *International Journal of Circuits, Systems and Signal Processing*, vol. 10, pp. 403–413, 01 2016.

[12] D. Zhang, G. Lindholm, and H. Ratnaweera, "Use long short-term memory to enhance internet of things for combined sewer overflow monitoring," *Journal of Hydrology*, vol. 556, 11 2017.

[13] H. Yan and H. Ouyang, "Financial time series prediction based on deep learning," *Wireless Personal Communications*, vol. 102, no. 2, pp. 683–700, 2018.

[14] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PLoS ONE*, vol. 12, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:37606221

[15] S. M. Kazemi, R. Goel, S. Eghbali, J. Ramanan, J. Sahota, S. Thakur, S. Wu, C. Smyth, P. Poupart, and M. Brubaker, "Time2vec: Learning a vector representation of time," 2019.

[16] K.-L. Du and M. Swamy, *Recurrent Neural Networks*, 12 2014, pp. 337–353.

[17] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735—-1780, nov 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[19] C. Olah, "Understanding lstm networks," http://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015, last accessed 13 February, 2024.

[20] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[21] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014.

[22] R. Zhao, D. Wang, R. Yan, K. Mao, F. Shen, and J. Wang, "Machine health monitoring using local feature-based gated recurrent unit networks," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 2, pp. 1539–1548, 2018.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 6000–6010.

[24] M. Sazli, "A brief review of feed-forward neural networks," *Communications Faculty Of Science University of Ankara*, vol. 50, pp. 11–17, 01 2006.

[25] J. Zheng, S. Ramasinghe, and S. Lucey, "Rethinking positional encoding," *ArXiv*, vol. abs/2107.02561, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:235742682

[26] A. Grossmann and J. Morlet, "Decomposition of hardy functions into square integrable wavelets of constant shape," *SIAM Journal on Mathematical Analysis*, vol. 15, pp. 723–736, 07 1984.

[27] S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[28] S. Mallat and S. Mallat, *A Wavelet Tour of Signal Processing*. Academic Press, 01 1999.

[29] P.-H. Chiang, S. P. V. Chiluvuri, S. Dey, and T. Q. Nguyen, "Forecasting of solar photovoltaic system power generation using wavelet decomposition and bias-compensated random forest," in *Proceedings of the IEEE* 9^{th} *Annual Green Technologies Conference (GreenTech)*, 2017, pp. 260–266.

[30] J. Gilles, "Empirical wavelet transform," *IEEE Transactions on Signal Processing*, vol. 61, no. 16, pp. 3999–4010, 2013.

[31] N. Malibari, I. Katib, and R. Mehmood, "Predicting stock closing prices in emerging markets with transformer neural networks: The saudi stock exchange case," *International Journal of Advanced Computer Science and Applications*, vol. 12, 01 2021.

[32] G. Bertasius, H. Wang, and L. Torresani, "Is space-time attention all you need for video understanding?" 2021.

[33] T. Muhammad, A. B. Aftab, M. Ibrahim, M. M. Ahsan, M. M. Muhu, S. I. Khan, and M. S. Alam, "Transformer-based deep learning model for stock price prediction: A case study on bangladesh stock market," *International Journal of Computational Intelligence and Applications*, vol. 22, no. 03, Apr. 2023. [Online]. Available: http://dx.doi.org/10.1142/S146902682350013X

[34] A. Meddah, "American stock index forecasting using transformers model," Al Akhawayn University, Tech. Rep., 2023.

[35] Z. Lin, "Comparative study of lstm and transformer for a-share stock price prediction," in *Proceedings of the 2023 2nd International Conference on Artificial Intelligence, Internet and Digital Economy (ICAID 2023)*. Atlantis Press, 2023, pp. 72–82. [Online]. Available: https://doi.org/10.2991/978-94-6463-222-4_7

[36] T. S. Mian, "Evaluation of stock closing prices using transformer learning," *Engineering, Technology and Applied Science Research*, vol. 13, no. 5, p. 11635–11642, Oct. 2023. [Online]. Available: https://etasr.com/index.php/ETASR/article/view/6017

[37] L. Costa and A. Machado, "Prediction of stock price time series using transformers," in *Anais do II Brazilian Workshop on Artificial Intelligence in Finance*. Porto Alegre, RS, Brasil: SBC, 2023, pp. 85–95. [Online]. Available: https://sol.sbc.org.br/index.php/bwaif/article/view/24955

[38] Q. Wang and Y. Yuan, *Stock Price Forecast: Comparison of LSTM, HMM, and Transformer*. Atlantis Press, 07 2023, pp. 126–136.

[39] S. Wang, "A stock price prediction method based on bilstm and improved transformer," *IEEE Access*, vol. 11, pp. 104 211–104 223, 2023.

[40] P. Bilokon and Y. Qiu, "Transformers versus lstms for electronic trading," 2023.

[41] S. Aman, "Forecasting stock price movements for intra-day trading using transformers and lstm," *International Journal of Computing and Artificial Intelligence*, vol. 2, no. 1, pp. 45–52, 2021.

[42] C. Wang, Y. Chen, S. Zhang, and Q. Zhang, "Stock market index prediction using deep transformer model," *Expert Systems with Applications*, vol. 208, p. 118128, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422013100

[43] Q. Zhang, C. Qin, Y. Zhang, F. Bao, C. Zhang, and P. Liu, "Transformer-based attention network for stock movement prediction," *Expert Systems with Applications*, vol. 202, p. 117239, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422006170

[44] Q. Ding, S. Wu, H. Sun, J. Guo, and J. Guo, "Hierarchical multi-scale gaussian transformer for stock movement prediction," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, C. Bessiere, Ed. International Joint Conferences on Artificial Intelligence Organization, 7 2020, pp. 4640–4646, special Track on AI in FinTech. [Online]. Available: https://doi.org/10.24963/ijcai.2020/640

[45] J. Liu, H. Lin, X. Liu, B. Xu, Y. Ren, Y. Diao, and L. Yang, "Transformer-based capsule network for stock movement prediction," 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:201626326

[46] Tickerchart, "Tickerchart," https://www.tickerchart.com/en/, 2023, last accessed 22 March, 2023.

[47] Amibroker, "Amibroker formula language," https://www.amibroker.com/index.html, 2024.

[48] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[49] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, no. 1, pp. 389–422, 2002.

[50] Y. Fan, Y. Qian, F.-L. Xie, and F. K. Soong, "TTS synthesis with bidirectional LSTM based recurrent neural networks," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[51] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2019.