mathematics

MDPI

Article

Deep Learning Models for Predicting Monthly TAIEX to Support Making Decisions in Index Futures Trading

Duy-An Ha [1], Chia-Hung Liao [2], Kai-Shien Tan [3] and Shyan-Ming Yuan [2,*]

[1] EECS International Graduate Program, National Yang Ming Chiao Tung University, Hsinchu 300, Taiwan; hdan.c@nycu.edu.tw
[2] Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu 300, Taiwan; aiallen.cs07g@nctu.edu.tw
[3] Institute of Computer Science and Engineering, National Yang Ming Chiao Tung University, Hsinchu 300, Taiwan; tkaishien.cs08g@nctu.edu.tw
* Correspondence: smyuan@nycu.edu.tw

Abstract: Futures markets offer investors many attractive advantages, including high leverage, high liquidity, fair, and fast returns. Highly leveraged positions and big contract sizes, on the other hand, expose investors to the risk of massive losses from even minor market changes. Among the numerous stock market forecasting tools, deep learning has recently emerged as a favorite tool in the research community. This study presents an approach for applying deep learning models to predict the monthly average of the Taiwan Capitalization Weighted Stock Index (TAIEX) to support decision-making in trading Mini-TAIEX futures (MTX). We inspected many global financial and economic factors to find the most valuable predictor variables for the TAIEX, and we examined three different deep learning architectures for building prediction models. A simulation on trading MTX was then performed with a simple trading strategy and two different stop-loss strategies to show the effectiveness of the models. We found that the Temporal Convolutional Network (TCN) performed better than other models, including the two baselines, i.e., linear regression and extreme gradient boosting. Moreover, stop-loss strategies are necessary, and a simple one could be sufficient to reduce a severe loss effectively.

Keywords: stock market prediction; deep learning; machine learning; Taiwan Capitalization Weighted Stock Index; Mini-TAIEX futures; simulated trading

Citation: Ha, D.-A.; Liao, C.-H.; Tan, K.-S.; Yuan, S.-M. Deep Learning Models for Predicting Monthly TAIEX to Support Making Decisions in Index Futures Trading. *Mathematics* **2021**, *9*, 3268. https://doi.org/10.3390/math9243268

Academic Editors: Vladimir A. Plotnikov and Massimiliano Ferrara

Received: 16 November 2021
Accepted: 13 December 2021
Published: 16 December 2021

1. Introduction

Predicting the stock market is a challenging task. The difficulties arise mainly from its non-stationary nature, noisy data, and high volatility [1]. Since stock prices are influenced by numerous factors, such as the characteristics of industries, the economic condition of the company, political events, news, the movement of other stock markets, and even the psychology of investors [2,3], the changes in stock prices resemble a random walk process [4]. Hence, it is hard to predict future stock prices based on their past movement or trend.

Nevertheless, numerous empirical research studies [1–3,5,6] have shown that the stock market can be predicted to some extent, and currently, machine learning is one of the preferred techniques for stock price prediction. Some studies have shown that we can achieve quite high accuracy in predicting stock prices using machine learning techniques. For example, Weng et al. [7] adopted three conventional machine learning models and reported an accuracy up to 85% in predicting AAPL stock movement one day ahead. Among various machine learning techniques, deep learning has emerged as a promising tool that can bring breakthroughs in this research area, as it has outperformed traditional machine learning methods in many disciplines, such as Natural Language Processing (NLP), computer vision, medicine, biology, playing games, and robotics [5].

The Taiwan Capitalization Weighted Stock Index (TAIEX) is a stock market index for measuring the performance of all stocks listed on the Taiwan Stock Exchange (TAIEX) and is the most prominent benchmark of Taiwan's securities market. TAIEX futures (TX) and Mini-TAIEX futures (MTX) are stock market index futures that use the TAIEX as the underlying asset. Considering annual trading volumes, TX and MTX are among the top three equity index futures in Taiwan (for more information on the highlights of annual trading volume, please see https://www.taifex.com.tw/enl/eng7/annualTrading, accessed on 30 September 2021). The rules for trading TX and MTX are almost identical except that the contract size of MTX is smaller than that of TX, i.e., NT$50 per index point compared to NT$200 per index point. In regular trading sessions, investors can trade many MTX contracts with different delivery months: the spot month, the next two calendar months, and the next three quarterly months. Weekly futures can also be listed in a given trading week (for more information on MTX trading rules, please see https://www.taifex.com.tw/enl/eng2/mTX, accessed on 30 September 2021). Our objective is to predict the Monthly Average of TAIEX (MAT) to support decision-making in trading monthly MTX contracts. Thus, the target users of our approach are individual investors who like to invest in the futures market for the relatively long term.

This study presents an approach to apply deep learning network architectures to predict MAT with different forecast distances. In our approach, we adopted three deep learning architectures for time series modeling, namely Long Short-Term Memory (LSTM), the hybrid architecture of Convolutional Neural Network (CNN) and LSTM called CNN-LSTM, and Temporal Convolutional Network (TCN). Since the TAIEX comes from an export-oriented economy that is very sensitive to changes in international markets, we assume that global financial and economic factors, such as world indices and commodity prices, could contribute to the prediction of the TAIEX. Therefore, we have inspected many global factors and selected the most valuable features by evaluating their correlation with TAIEX. Furthermore, to denote the effectiveness and provide guidance on applying the predicted models in practice, we conducted a simulation on trading MTX with a simple trading strategy. In summary, our main contributions include:

- We have proposed an approach to apply time series modeling based on deep learning for predicting MAT.
- We performed extensive experiments to evaluate three different deep learning architectures on the task of supporting decision-making in trading MTX contracts.
- We proposed and compared the effectiveness of two simple stop-loss strategies that can help avoid losses in trading index futures using predictive models.

The rest of this article is structured as follows. Section 2 is a literature review. Section 3 introduces three deep learning architectures used in this study. Section 4 and Section 5 present the details of our proposed approach and the experimental results, respectively. Lastly, Section 6 concludes this article and gives some directions for future work.

2. Literature Review

Table 1 provides a summary of recent studies that apply deep learning techniques to stock market prediction. The reader can refer to [1–3,5,6] to have more comprehensive and detailed reviews.

Table 1. A summary of academic articles on stock market prediction based on deep learning.

Article	Target Data	Input Variables	Target Output	Main Method	Simulated Trading	Year
Wang et al. [8]	SSE, TAIEX, KOSPI, and NIKKEI	Stock prices	Price; Daily	RNN	N	2016
Persio and Honchar [9]	S&P 500	Stock prices	Direction; Daily	MLP, RNN, CNN	N	2016
Chong et al. [10]	KOSPI 38 stock returns	Intraday stock returns	Return; 5 min ahead	DNN	N	2017
Gunduz et al. [11]	Borsa Istabul BIST 100 stocks	Technical indicators, stock prices, temporal variable	Direction; hourly	CNN	N	2017
Li and Tam [12]	HIS, SSE, SZSE, TAIEX, NIKKEI, KOSPI	Technical indicators, stock prices	Price; Daily	LSTM	N	2017
Fischer and Krauss [13]	S&P 500	Stock prices	Direction; Daily	LSTM	N	2018
Baek and Kim [14]	S&P 500, KOSPI	Stock prices	Price; Daily	LSTM	Y	2018
Chung and Shin [15]	KOSPI	Technical indicators, stock prices	Price; Daily	LSTM	N	2018
Chen et al. [16]	CSI 300 futures contract	Transaction data	Price; 1 min ahead	DNN	N	2018
Zhou et al. [17]	Stocks in CSI 300	Technical indicators	Price; 1 min ahead	GAN based on LSTM and CNN	N	2018
Zhong and Enke [18]	SPY	Technical indicators, financial and economic factors	Direction; Daily	DNN, ANN	Y	2019
Hoseinzade and Haratizadeh [19]	S&P 500, NASDAQ, DJI, NYSE, and RUSSELL	Technical indicators, stock prices, financial and economic factors	Direction; Daily	CNN	N	2019
Sim et al. [20]	S&P 500	Technical indicators	Direction; 1 min ahead	CNN	N	2019
Wen et al. [21]	S&P 500 and its individual stocks	Stock prices	Price; Daily	CNN	N	2019
Lee et al. [22]	US stocks	Stock chart images of daily closing price and volume data	Return; Daily	CNN	N	2019
Long et al. [23]	CSI 300	Stock prices	Direction; 5–30 min ahead	MFNN based on CNN and LSTM	Y	2019
Pang et al. [24]	Shanghai A-share composite index , and 3 stocks	Technical indicators, stock prices	Price; Daily	LSTM	N	2020
Kelotra and Fiaz [25]	6 stocks	Technical indicators	Price; Daily	ConvLSTM	N	2020
Chung and Shin [26]	KOSPI	Technical indicators	Direction; Daily	CNN	N	2020
Nabipour et al. [27]	4 stock market groups	Technical indicators	Price; 1–30 days ahead	ANN, RNN, LSTM	N	2020
Long et al. [28]	CITIC Securities, GF Securities, China Pingan	Transaction records data and public market information	Direction; 1 day ahead for stock price movement; 5- and 7-days trend	DSPNN based on CNN and LSTM	N	2020
Nabipour et al. [29]	4 stock market groups	Technical indicators	Direction; Daily	RNN, LSTM, DT, RF, Adaboost, XGBoost, SVC, NB, KNN, LR, ANN	N	2020
Lei et al. [30]	SSE	Technical indicators, stock prices, investor attention factor synthesized by Baidu search index	Volatility; Daily	TCN	N	2021

DSPNN: Deep Stock-trend Prediction Neural Network, ANN: artificial neural network, MLP: Multi-layer Perceptron, GB: Gradient Boosting, DT: Decision Tree, RF: Random Forest, SVC: Support Vector Classifier, NP: Naïve Bayes, KNN: K-Nearest Neighbors, LR: Logistic Regression, SVR: Support Vector Regression, AR: Autoregression, Y: Yes, N: No. Column "Target output" shows the type of the output variable and how far in the future that the model can predict, e.g, "Price; Daily" means that the output of the model is a stock price (real number) and this is a predicted value for one day ahead.

Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) are deep learning architectures that have dominated the field of stock market forecasting. RNN is a kind of neural network, but with hidden states that allow it to "remember" previous events in time series data, hence improving its ability to predict the subsequent events. LSTM is a well-known improved version of RNN with the ability to learn long-term dependencies and is famous for tasks in NLP. Almost all the works using RNN or LSTM

model the problem as a regression problem [8,12,14,15,24,27], only a few studies try to build classification models [9,13,29]. CNN is well known in the computer vision domain, and its ability to extract efficient features has been proven in various disciplines. In contrast with RNN and LSTM, CNN was not developed to work with time series data. Hence, studies on the application of CNN in stock market prediction have only become popular recently. However, Persio and Honchar [9] have empirically demonstrated that CNN can outperform LSTM in predicting the direction of stock market movement. Moreover, we also note that most studies using CNN have attempted to build classifiers for predicting the direction of stock market movement [9,11,19,20,26] instead of regression models [21,22]. In order to exploit the capabilities of CNN in feature extraction and LSTM in handling time series data, some studies have tried to apply the hybrid deep learning architecture with the combination of CNN and LSTM to predict stock market prices [23,25] and directions [28].

Deep Neural Network (DNN), an early deep learning architecture, has also been used in many studies. DNNs are typical feedforward neural networks with multiple layers between the input and output layers. Zhong and Enke [18] employed DNN with PCA, while Chong et al. [10] studied the performance of DNN with PCA as well as Restricted Boltzmann Machine (RBM) and autoencoder for unsupervised feature extraction. Chen et al. [16] also used DNN with RBM and autoencoder, but they were used for initializing the weights and pre-training the network.

In addition to the above architectures, many other articles have proposed other types of deep learning models for stock market prediction. Zhou et al. [17] proposed a Generative Adversarial Network (GAN) employing LSTM for the generative model and CNN for the discriminative model to predict high-frequency stock prices. Their extensive experiments showed that the proposed model could efficiently enhance the accuracy of stock price prediction and reduce errors. Lei et al. [30] applied a new deep learning architecture, namely Temporal Convolutional Network (TCN), to predict the volatility of Shanghai Securities Composite Index (SSE). They found that the TCN model with an investor attention factor worked better than many other models, including the TCN model without investor attention, the LSTM model with investor attention, and three traditional econometric models. Long et al. [23] proposed an innovative end-to-end architecture called Multi-Filters Neural Network (MFNN). It is particularly suitable for price prediction and feature extraction from financial time series data. MFNN was constructed based on the integration of convolutional filters and recurrent filters for feature learning. Their experiments have shown that the proposed model achieves better results in terms of accuracy, stability, and profitability than traditional machine learning methods, statistical models, and single-structure networks (CNN, RNN, and LSTM).

Besides the main method, Table 1 also provides other information, including the input variables, the target output, and whether simulated trade exists in the studies. The two most common input variables used for stock market forecasting are technical indicators and stock prices. By applying mathematical calculations, technical indicators are derived from stock prices, such as opening price, closing price, low price, high price, and volume. In terms of target output, most studies have attempted to predict the direction or price of stocks one day ahead, and it is rare to find a study that tried to predict further into the future. Finally, we note that most studies of stock market prediction do not show how their models are applied in practice. Of the 23 articles in Table 1, we found only three articles [14,18,23] that suggested how to leverage their models to develop trading strategies by providing a trading simulation in their experiments.

TAIEX is one of the most popular stock market indices used in the literature to evaluate the proposed stock market forecasting methods [1]. However, not too many studies currently use deep learning to forecast the TAIEX. Wang et al. [8] proposed ST-ERNN model, which integrates an Elman recurrent neural network (ERNN) with a stochastic time effective function to forecast the values of SSE, TAIEX, KOSPI, and NIKKEI. They reported that ST-ERNN outperformed the backpropagation neural network (BPNN), the ERNN, and the stochastic time-effective neural network (STNN). Li and Tam [12] presented a

hybrid model combining the real-time wavelet denoising and the LSTM to predict the values of six East Asian stock indices, including the TAIEX. The experimental results indicated that the hybrid model has remarkable performance improvement compared with the original LSTM model. Hsieh et al. [31] introduced an integrated system using wavelet transforms to eliminate noise from stock price time series and RNN to build a model to predict the values of DJIA, FTSE, NIKKEI, and TAIEX. In this paper, the RNN weights and biases were optimized by the artificial bee colony algorithm under a parameter space design. The stepwise regression-correlation selection was adopted to select the input variables that affect the target variable the most. The experiments showed that the proposed model was superior to many conventional models, namely BPNN, ANN optimized by the ABC, and fuzzy time series model. All three studies [8,12,31] attempted to predict the values of stock market indices for the next day, and the input variables were stock prices or stock prices combined with technical indicators.

For predicting the monthly TAIEX average, Sun et al. [32] and Ha et al. [33] adopted linear regression while Tan et al. [34] tried both linear regression and extreme gradient boosting. All three studies used correlation analysis and statistical hypothesis testing for feature selection and worked with monthly averages for both the independent and dependent variables. The forecast distance is one and one to five months ahead in [32] and [33,34], respectively. They also used a similar strategy to our study but without a stop-loss strategy.

3. Background

3.1. LSTM

LSTM networks are a special kind of RNN networks that can learn long-term dependencies and overcome the problem of vanishing or exploding gradients in standard RNN networks. Hochreiter and Schmidhuber proposed LSTM in 1997 [35], and it has subsequently been improved by many researchers. LSTM works remarkably well on a variety of problems and is widely used today.

A typical LSTM network usually consists of one or more LSTM layers, followed by a Fully-Connected layer (FC). The network's input is a sequence of input features, while the output can be a sequence of data or a value, depending on the specific application. Figure 1 illustrates an unrolled representation of an LSTM network with one LSTM layer. In broad terms, LSTM networks look similar to RNN networks, except for the LSTM memory block.

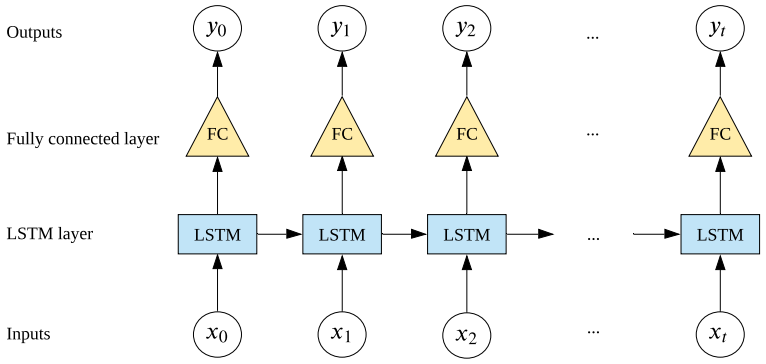


Figure 1. An unrolled LSTM network with one LSTM layer. x_t is the input features at time step t.

Figure 2 demonstrates the inner structure of the LSTM memory block. Instead of having only one hidden state as in RNN, the LSTM has another hidden state called the cell state. The cell state stores long short-term memory, while the hidden state focuses on the next input to predict the output. In this figure, h_t, c_t, and x_t represent the hidden state, the cell state, and the input features at time step t, respectively. The rectangles, called

gates, represent four linear layers. The activation functions of the gates labeled F, I, and O are sigmoid, while the activation function of the gate labeled T is tanh. The circles denote element-wise operations, including addition (+), multiplication (×), and tanh function (f_{th}). At each time step t, the inputs of the four gates are a concatenation of x_t and the previous hidden state h_{t-1}. These gates allow the LSTM block to regulate the flow of information as follows:

- The forget gate F defines which things should be forgotten in cell state c_{t-1}.
- The input gate I cooperates with gate T to update the cell state with new input x_t and the previous hidden state h_{t-1}.
- The output gate O specifies what information from the new cell state c_t is used to generate the new hidden state h_t. The new hidden state is also used as the output of the LSTM block.

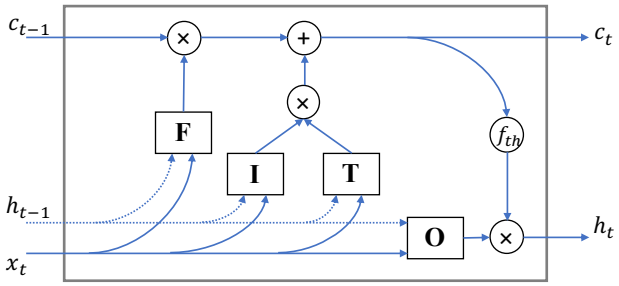


Figure 2. Inner structure of LSTM memory block following Olah [36] and Chevalier [37].

3.2. CNN-LSTM

CNN-LSTM is a hybrid architecture that can take advantage of both CNN and LSTM. Specifically, CNN layers are used for feature extraction from the input data, while LSTMs support sequence prediction. The combination of CNN and LSTM is suitable for applications with visual time series data, such as generating textual descriptions from videos. However, with a slight adaptation, it can also work with other types of time series data. In practice, CNN and LSTM can be integrated in various ways to form CNN-LSTM networks [38–41]. In this study, we use a CNN-LSTM architecture as in Figure 3. This architecture is similar to those used by Donahue et al. [40] and Livieris et al. [41]. It includes a CNN network that receives the input sequence, and the output is then fed into a subsequent LSTM network. In the CNN network, 1D convolution is used because the time series data in our problem has a 1D structure.

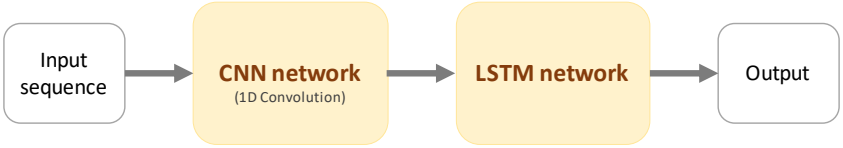


Figure 3. CNN-LSTM architecture using in our study.

3.3. TCN

Bai et al. [42] presented a temporal convolutional network architecture in 2018; it is a modified version of CNN for sequence modeling problems. This study showed that TCNs could outperform conventional recurrent networks, i.e., LSTM, GRU, and standard RNN, on various tasks and datasets. TCNs can overcome the inherent drawbacks of recurrent networks, such as the vanishing gradient problem or the lacking memory retention. In addition, TCNs can improve the computational efficiency of the model compared to recurrent networks by allowing parallel computation of outputs.

TCN was developed based on two principles that differ from previous convolutional architectures for sequential data: (1) the convolutions used in the networks should be causal to ensure that the leakage of information from the future to the past cannot happen, and (2)

as a standard RNN network, TCN can take inputs of arbitrary sequence length and map them to an output sequence of the same length. TCN employs causal convolutions to satisfy the first requirement. In causal convolutions, the output at time steps t depends only on the inputs up to time step t in the preceding layer. For the second design goal, the authors adopted the 1D fully-convolutional network architecture [43] with zero padding to the left of the input sequence.

Besides the two design goals, another desirable property of the TCN model is that the output at time step t depends on all elements from time step 0 to t in the original input; this is referred to as "full history coverage." This requirement can lead to an extremely deep network or a large kernel size for a large sequence length. Therefore, dilated convolution was proposed to avoid making the network too complex. Unlike conventional convolution, in dilated convolution, the distance d between the elements of the input sequence used to compute one entry of the output sequence can be greater than 1, and d is called the dilation factor of dilated convolution. This mechanism allows the fixed-size kernel to process a wider area by skipping part of the input to obtain more information. Figure 4 depicts a dilated casual convolution with kernel size $k = 3$ and $d = 2^i$ at i-th network layer. In this example, the network with three dilated convolutional layers can have a receptive field (the receptive field is the set of elements of the original input on which a given output element depends) up to 15.

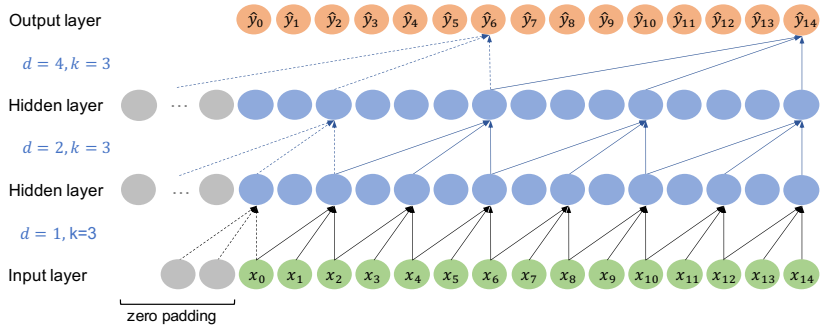


Figure 4. An illustration of dilated causal convolution following Bai et al. [42]. The circles depict the input/output of the dilated causal convolutional layers.

To make the deep and large TCNs stable, a residual block [44] is used to replace a single dilated causal convolutional layer. Figure 5 shows the structure of a residual block with kernel size k and dilation factor d. A residual block consists of two dilated causal convolutional layers with a nonlinear activation unit, i.e., the rectified linear units (ReLU), for each layer. Weight normalization [45] and dropout [46] are also added to each convolutional layer for normalization and regularization. The output of the second convolutional layer is added to the input of the residual block through a connection called residual connection before it is passed to the ReLU activation unit and becomes the output of the residual block. With the residual connection, the stacked nonlinear layers fit a transformation function $F(x) = H(x) - x$ instead of the entire transformation $H(x)$. The optimization of $F(x)$ is assumed to be simpler than that of the original function $H(x)$. In reality, the residual connection has proven useful in very deep networks to avoid the problem of degradation [44]. Lastly, since the output of the convolutional layers and the input of the residual block could have different channel widths, the optional 1×1 convolution is used to ensure that they have the same shape to perform the element-wise addition.

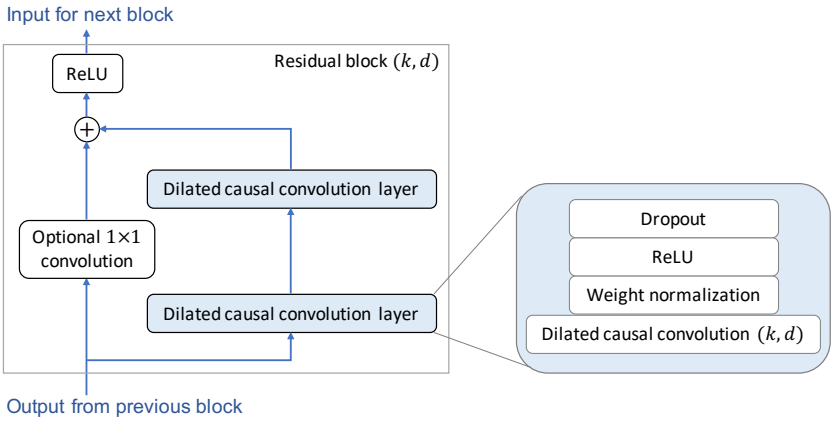


Figure 5. Structure of a residual block (k, d) following Bai et al. [42].

4. Method

4.1. The Framework for Predicting Monthly Average of TAIEX

As mentioned in the introduction section, we aim to predict the Monthly Average of TAIEX (MAT) to support decision-making when trading MTX contracts in the futures market. Because the delivery months of MTX contracts traded on the Taiwan Futures Exchange can vary from 1 to 12 months, we need models that can predict the MATs of different months in the future. We have limited our framework to support the forecast distance s of up to five months in the future ($s \in [1 \ldots 5]$) for two reasons: The further in the future, the harder it is to predict the value of a variable in general, and the low trading volume (the daily trading volume of the contracts can be found at the following link: https://www.taifex.com.tw/enl/eng3/futDailyMarketView, accessed on 30 September 2021) of contracts that have long delivery months.

In this study, the main structure of the framework for building models to predict MAT includes three stages: (1) identifying potential factors that have a relationship with the development of TAIEX, (2) data preprocessing, and (3) training models. For the first and second stages, we used an approach similar to that used in our previous study [33]. The details of these stages are described later in this section.

4.1.1. Identifying Potential Factors

Because of economic globalization, stock markets around the world are always in contact with and affect one another to some extent [19]. This connection is even stronger for export-oriented economies such as Taiwan's economy. Hence, we mainly considered global financial and economic factors as potential features for predicting the TAIEX. Thus, we examined a total of 46 potential factors, most of them fall into three categories: World indices, commodity prices (including fuel prices and precious metal prices), and New Taiwan Dollar (TWD) exchange rates. Readers can refer to Table A1 in Appendix A for more details about these factors. In this study, we considered much more potential factors compared to previous studies [32–34] (References [32–34] inspected 16 and 22 factors, respectively), and the closing prices of these independent variables were collected daily. It is worth noting that the target variable TAIEX was also collected daily, but the opening price was used instead of the closing price (we assume that the opening and closing prices can be used interchangeably).

4.1.2. Data Preprocessing

In this stage, we performed several steps to prepare the data for training the forecasting models. These steps include handling missing values, identifying the most valuable features for predicting the TAIEX, and feature transformation.

1. Handling missing values: When collecting daily financial and economic factors representing global stock markets, some missing values occurred in the dataset due

to the different holidays in each country or region. Since there are only a few missing values in the dataset, we adopted a simple solution to this problem: to fill the missing value with the value of the preceding trading day.

2. Identifying the most valuable features: To select the most relevant independent variables from all candidates, we evaluated the pairwise correlation between each variable and the target variable TAIEX, and then picked out only those with a high correlation value. Correlation reflects the degree to which two variables are linearly associated, and the value is usually in the range of $[-1, 1]$. The positive or negative sign indicates whether two variables are positively or negatively related, and -1 or 1 means that two variables have a perfect negative or positive correlation. We used Spearman's rank correlation coefficient to measure the correlation. Compared to another well-known correlation coefficient, Pearson's coefficient, which only works with a linear relationship between two variables, Spearman's coefficient can also work with monotonic relationships. In addition, the Spearman correlation is less sensitive to outliers than the Pearson correlation. The optimized subset of features may differ for each forecast distance s; therefore, we calculated the pairwise correlation for each case of s. In this way, the time interval between any pair of values of the independent variable and the target variable in the calculation of correlation was defined by $s * 22$, where 22 is the average number of regular trading days in a month for a stock market. In addition to the correlation coefficient, we also performed a hypothesis test of the "significance of the correlation coefficient" to ensure that the linear relationship in the sample data was strong enough to reflect the actual relationship in the whole population. Any feature having the significance of the correlation coefficient (p-value) less than the significance level 0.05 would not be selected.
3. Feature transformation: We applied two well-known feature transformation techniques to enhance the framework's performance. The first technique is standardization, a feature scaling method that allows normalizing the range of values of independent variables. Through standardization, the values of each variable have a zero mean and unit variance. The second technique is Principal Component Analysis (PCA), a popular method for dimensionality reduction. Many studies have shown that using PCA can improve the performance of stock prediction models in terms of accuracy and stability [10,18,47,48]. PCA helps preserve only the majority of the variance (information); hence the complexity of the models and the training time can be significantly reduced. Moreover, by reducing the complexity, the overfitting problem can also be prevented to a certain extent. Our framework tried to select the number of components n so that the amount of variance they can explain is at least 95% of the total variance.

4.1.3. Training Models

The framework trains models based on three deep learning architectures, LSTM, CNN-LSTM, and TCN, and each model works for a particular forecast distance s. Both LSTM and TCN networks require input as sequences of feature values for training. By using 1D convolution, the CNN-LSTM networks can also work with the input sequences. Specifically, the training set for all models is a set of pairs (X_t, Y_t), where $X_t = (x_{t-sl}, \ldots, x_{t-1}, x_t)$ and $Y_t = (y_{b+1}, \ldots, y_{b+21}, y_{b+22})$ with $b = t + (s-1) \times 22$ are the input sequence and the corresponding target output for the networks at time step $t \in \mathbb{N}$, respectively. Y_t contains 22 values corresponding to the values of the TAIEX on 22 trading days of a month. x_t and y_t are the feature vector with size n and the value of the TAIEX at time step t, respectively. Finally, sl is the length of the input sequences, called the sequence length.

Since TAIEX has large values, which can cause the learning process to be unstable, we scaled them down by simply dividing by a constant that is 10,000 in our experiment. Moreover, given an input sequence X_t, the output $\hat{Y}_t$ of the models in our framework is the prediction of TAIEX for 22 days in the target month, which does not match our

target, namely MAT. Therefore, we took the average of the 22 values in $\hat{Y}_t$ to represent the predicted value of MAT.

To build models used to support decision making in trading MTX in a year Y, we adopted a two-stage training approach:

1. Hyperparameter tuning: We split the available data into a training set D_{train} and a validation set D_{val}. The data from year $(Y-2)$ backward are used for D_{train}, while the data from year $(Y-1)$ are used for D_{val} to tune two hyperparameters: the batch size bs used in the Stochastic Gradient Descent and the sequence length sl.
2. Training final model: The framework retrains the models using all available data $(D_{train} \cup D_{val})$ and the optimal hyperparameters found in the previous phase. These models are the final models that will be used to predict MATs in year Y.

4.2. Simulated Trading

To evaluate the performance of the models in practice, we conducted simulations on MTX trading, where the selection of MTX contracts for investment was based on the predictions of the models. Concerning trading strategy, we adopted a similar approach as in [33], which is suitable for individual investors who do not want to spend too much time tracking the volatility of the stock market. The trading fee was ignored in the simulations for simplicity but without loss of generality. The trading period for one simulation was one year, and the principal amount was NT $200,000. On the first trading day of each month, only one MTX contract that was expected to rise was considered for purchase, i.e., we only hold long positions during trading time. If more than one contract was predicted to rise, the highest was selected. After we purchased a contract, we held it until it met one of the following conditions: (1) the expected value was reached, (2) the contract's delivery date was reached, or (3) the contract's value dropped to the stop-loss threshold V_{st}.

Since the futures market is highly volatile and somewhat unpredictable, we need a mechanism to mitigate the loss when the market trend suddenly changes in an unexpected direction. Therefore, we have proposed two simple stop-loss strategies that can be used with the above trading strategy. These strategies differ only in the way the stop-loss threshold is defined:

1. Manual Determination (MD strategy): Individual investors can define the threshold based on their bearable loss, i.e., if the investor can afford a loss τ, he can hold their long position in an MTX contract until its value falls to the stop-loss threshold $V_{st} = V_o - \tau$, where V_o is the value of the contract at the time the investor owns the contract.
2. Inferring from the predicted Standard Deviation (STD strategy): Since the contract's value can fluctuate for a while before reaching our predicted value, we do not want to abandon a contract too early due to a temporary drop in a "normal" range. We have proposed to take the predicted standard deviation $\hat{\sigma}$ of the daily predicted contract's values to determine the normal range for the target month. In this way, we allow the value to fall up to $(\hat{V} - \hat{\sigma})$, where $\hat{V}$ is the predicted MAT value and $\hat{\sigma}$ is calculated from the models' 22 output values for the target month. In conjunction with the affordable loss concept mentioned earlier, we suggested to compute the stop-loss threshold as $V_{st} = \min(\hat{V} - \hat{\sigma}, V_o - \tau)$.

5. Empirical Results

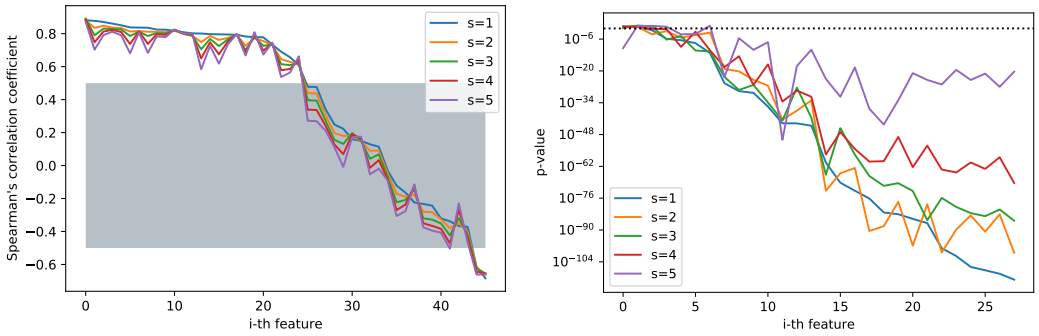
This section presents our experimental results on the performance of different deep learning models built with the framework. The experiments were implemented in Python using the PyTorch framework.

5.1. Data Specification

The datasets used for the experiments include the daily opening price of TAIEX and the daily closing price of 46 factors as mentioned in Section 4.1.1. The data were collected

from January 2008 to December 2020 and were mainly obtained from Yahoo Finance. (https://finance.yahoo.com/, accessed on 30 September 2021).

At the step of identifying valuable features, we considered features with pairwise correlation in $[-1,-0.5) \cup (0.5,1]$ and p-value < 0.05 as the most useful features. As showing in Table 2, the number of selected features was 24 and 25 for $s \in \{3,5\}$ and $s \in \{1,2,4\}$, respectively. Most of the selected features were the world indices; only a few features were from commodity prices and TWD exchange rates. Moreover, the selected features were almost the same for all forecast distances s, as we can see in Figure 6. In Figure 6a, the potential features were sorted by their correlation coefficient with $s = 1$, and the most correlated features are those whose correlation coefficient is outside the gray area. Figure 6b represents the p-value of the most highly correlated features, and the black dotted line denotes the threshold p-value $= 0.05$. There are just a few features that fail the significance test.



(a) **(b)**

Figure 6. The pairwise correlation coefficient and the p-value of the potential features. (**a**) Correlation coefficient; (**b**) p-value.

Table 2. Types of feature sets and the number of selected features for each forecast distance s after applying correlation analysis and hypothesis test.

Type of Feature Set	# of Features	# of Selected Features for Each Forecast Distance	
		{1, 2, 4}	{3, 5}
World Indices	25	20	19
US Treasury bonds rates	4	0	0
Commodity	8	1	1
TWD exchange rates	7	2	2
Companies	2	2	2
Total	46	25	24

Lastly, by applying PCA for dimensionality reduction, the number of selected features of the training dataset is reduced to 4 and 5 for $s \in \{1,5\}$ and $s \in \{2,3,4\}$, respectively.

5.2. Models Setting

Since all deep learning models have some hyperparameters, we had to select values for these hyperparameters before training the models. Roughly speaking, there are two types of hyperparameters: hyperparameters related to a specific network structure and general training hyperparameters. In our experiments, most of the hyperparameters were fixed, except for the batch size bs and the sequence length sl, for which the system tries to find the optimal values from a set of values through the training process.

1. Hyperparameters related to specific network structure
   (a) LSTM:
      - Number of LSTM layers: 2

- Number of neurons in hidden layer: 200
- Dropout probability: 0.3 (using dropout between LSTM layers and before passing the output of the final LSTM layer to the FC layer to reduce overfitting. A good value for dropout probability is between 0.3 and 0.9 [46]. We selected 0.3, since the number of neurons in the hidden layer is not much.)

(b) CNN-LSTM:

- For CNN network:
  - Number of 1D convolution layers: 1
  - Out channels of 1D convolution layer: $2n$
  - Kernel size 5 and stride 5
- For LSTM network: same as the above LSTM network.

(c) TCN:

- Number of residual blocks (TCN networks require a minimum number of residual blocks, which depends on the sequence length, for full history coverage): $\left\lceil \frac{sl-1}{4} + 1 \right\rceil$
- Kernel size 3 and stride 1
- Dilation factor: 2^i for i-th residual block
- Out channels of the dilated causal convolutions: 22
- Dropout probability: 0.3

2. General training hyperparameters
   - Input sequence length sl: $\{132, 264, 396, 528, 660, 792\}$ (these values correspond to the number of trading days in 6, 12, 18, 24, 30, and 36 months, respectively).
   - Batch size bs: $\{16, 32, 64, 128\}$
   - Optimizer: Adam
     - Learning rate: 5×10^{-4}
     - $\beta_1 = 0.9, \beta_2 = 0.999$
     - $eps = 10^{-8}$
     - Weight decay: 10^{-4}
   - Loss function: Mean squared error
   - Max Epochs: 100
   - Early training-stopped condition: No further improvement on the loss of validation data over 10 consecutive epochs.

Most of the general training hyperparameters were the default and common values. For example, the values for $\beta_1, \beta_2,$ and eps are the default values set by the PyTorch framework for the Adam optimizer. Our principle in choosing the hyperparameters for the network structures was to form a typical network model for each type that is neither too complicated nor too simple.

5.3. Evaluation

In this study, two conventional machine learning models, Linear Regression (LR) and eXtreme Gradient Boosting (XGBoost), were selected as baselines for comparison with the performance of the three deep learning models. We took the same approach as Ha et al. [33] and Tan et al. [34] to build models using LR and XGBoost, but used our dataset. We assessed the performance of the models using two metrics: Root Mean Squared Percentage Error (RMSPE) for various forecast distances and the profit in terms of the Rate of Return (RoR) and index points gained in simulated trading in the three years 2018, 2019, and 2020.

Given a set of target MATs and their predicted values $\{v_k, \hat{v}_k\}_{k=1}^{K}$, the RMSPE is computed as in Equation (1).

$$\text{RMSPE} = \sqrt{\frac{100}{K}\sum_{k=1}^{K}\left|\frac{v_k - \hat{v}_k}{v_k}\right|^2} \quad (1)$$

5.3.1. RMSPE

Figure 7 compares the RMSPE of the five models. Overall, the RMSPE of all models is lower in the first two years, 2018 and 2019, than in 2020. This can be explained in part by the fact that 2020 was an unpredictable year because of the COVID -19 pandemic; therefore, MAT has very high volatility in 2020, which can be observed in Figure 8. Indeed, as shown in Figure 9, the standard deviation of MAT in 2020 is more than twice as high as in the previous two years. Furthermore, as expected, the difficulty of the prediction increased as the forecast distance enlarged, and this trend is even more clear in 2020.

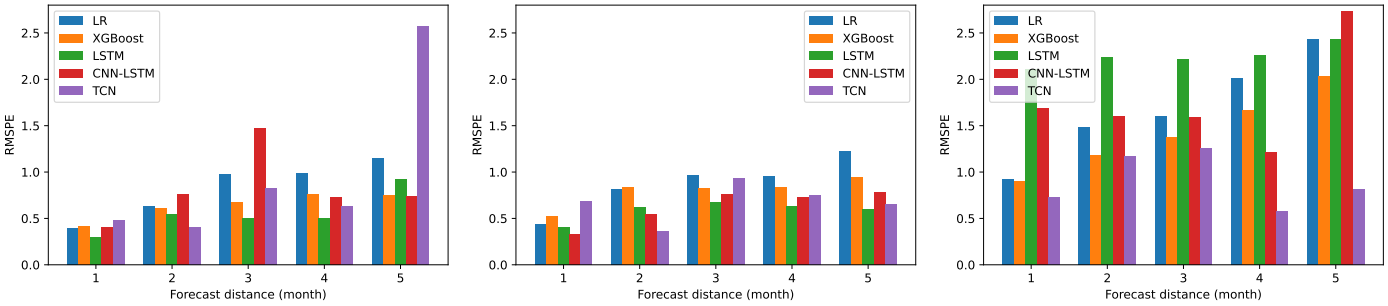


Figure 7. RMSPE of models with different forecast distances in three years. (**a**) 2018. (**b**) 2019. (**c**) 2020.

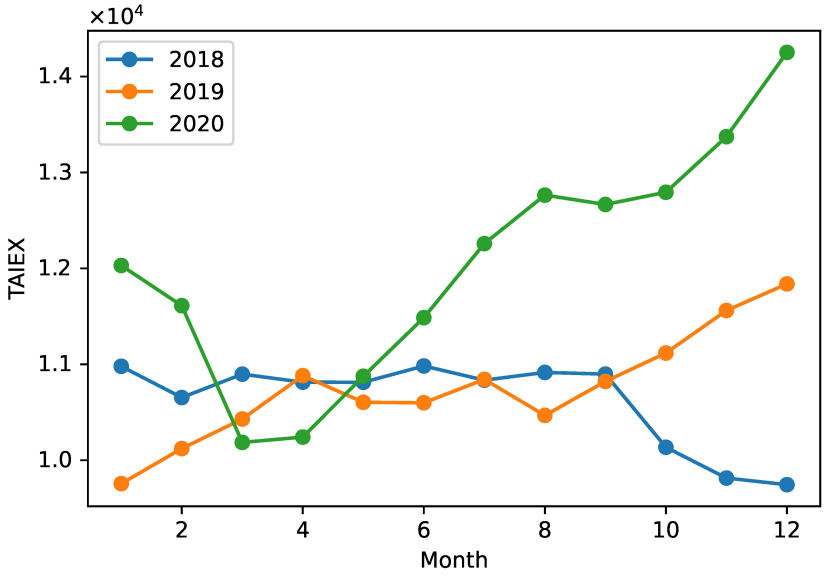


Figure 8. Monthly average values of TAIEX in 3 years.

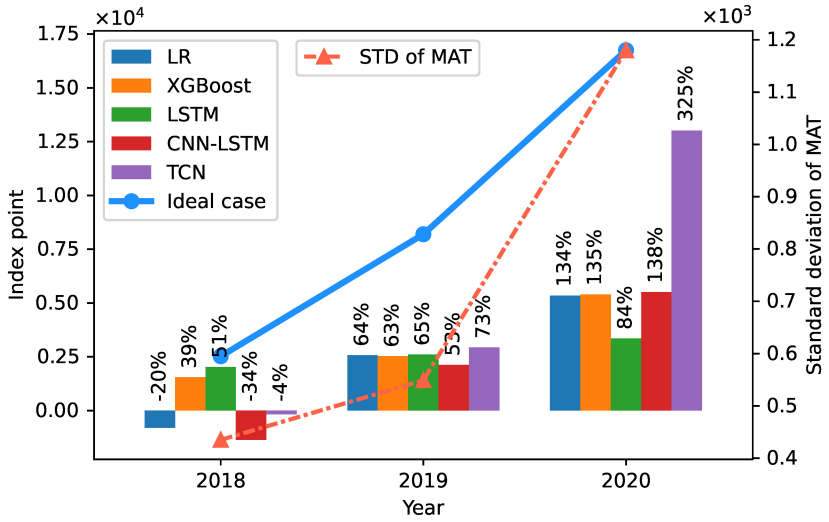


Figure 9. The profit gained from the simulated trading without using any stop-loss strategy.

In 2018 and 2019, LSTM appeared to perform better than the other models. In six out of ten cases, LSTM was the best performing model. In 2020, however, TCN outperformed the other models at all forecast distances, while LSTM was the worst model. Besides Figure 7, Table 3 can help to compare the performance of models easier. This table shows the count of cases (there are a total of 15 cases, i.e., five forecast distances multiplied by three years) where a model was among the top one or two models ranked by the RMSPE metric. From this perspective, the deep learning models were clearly superior to the baselines, especially TCN and LSTM.

Table 3. The number of times a model is in the top one or two models ranked by the RMSPE metric.

	LR	XGBoost	CNN-LSTM	LSTM	TCN
In top one model	0	0	2	6	7
In top two models	1	6	6	8	9

5.3.2. Profit

Figure 9 displays index points gained by simulated trading without using a stop-loss strategy ($V_{st} = -\infty$) and the corresponding RoR for each model in three years. The RoR is shown above the head of the bars in percentage and is calculated as follows: RoR $= G * U_v / I$, where G, U_v, and I are the gained index points, the price per index point, and the initial capital, respectively. In our experiments, we had $U_v = 50$ and $I = 200,000$. To illustrate the risk of investing in MAT, the standard deviation (STD) (STD is a common measure of risk used in the financial industry) of MAT is also shown. Finally, the blue line in the figure denotes the index points gained in the ideal case where we could predict MAT exactly.

In 2018, LSTM had the best performance with 51% RoR, almost reaching the ideal point. However, in the other two years, TCN was the best model, especially in 2020, with 325% RoR, TCN was far above the other models and was not far from the ideal point. LSTM also had a good performance in 2019 with 65% RoR, which was above both baselines. However, its RoR was the lowest in 2020, although it was much higher relative to itself in the previous two years. CNN-LSTM was probably the worst model among the deep learning models. It had a good performance in 2020 with 138% RoR, above both baselines, but its RoR was the lowest in the first two years and even deeply negative in 2018. These results are consistent with the analysis of the RMSE in Section 5.3.1.

Figure 10 presents the profit achieved in simulated trading for the models using different stop-loss strategies: None ($V_{st} = -\infty$), MD50 (the MD strategy with $\tau = 50$),

and STD50 (the STD strategy with $\tau = 50$). Since the output of the LR and XGBoost models are values of MAT, the STD strategy is not applicable to these two models.

Overall, both MD50 and STD50 can help prevent losses effectively. Figure 10a shows that after applying stop-loss strategies, LR, CNN-LSTM, and TCN had very good RoR instead of significant loss as in 2018.

Nevertheless, stop-loss strategies can also reduce the potential profit by abandoning a contract that eventually rises to the predicted value after a temporary deep fall. Indeed, we can observe this trend in Figure 10, and it was true for all models in the simulated years. In practice, this problem can be mitigated by using more sophisticated trading strategies, such as investing in another promising contract to immediately replace the one taken out, rather than waiting until next month. Regarding the effectiveness of stop-loss strategies, Figure 10 demonstrates that all models achieved almost the same RoR for both MD50 and STD50, with the exception of TCN in 2018, where the trading with STD50 had an RoR higher 2.5% than that with MD50. This result can be explained by the fact that V_{st}^{STD50} was almost always equal to V_{st}^{MD50} for most of the time in the simulations due to $(V_0 - 50) \leq (\hat{V} - \hat{\sigma})$. STD50 hardly helped investors have more profit than MD50 in this experiment. However, in the ideal case where MAT and its associated standard deviation could be predicted precisely, profit with STD50 can increase up to 7.3% and 25.9% in 2018 and 2019, respectively, compared to MD50. So we can expect that STD50 can perform better if the accuracy of the models is improved. For more details on the profit in the ideal case and the values of $(V_0 - 50)$ and $(\hat{V} - \hat{\sigma})$ in the simulations, see Tables A2 and A3 in the Appendix A.

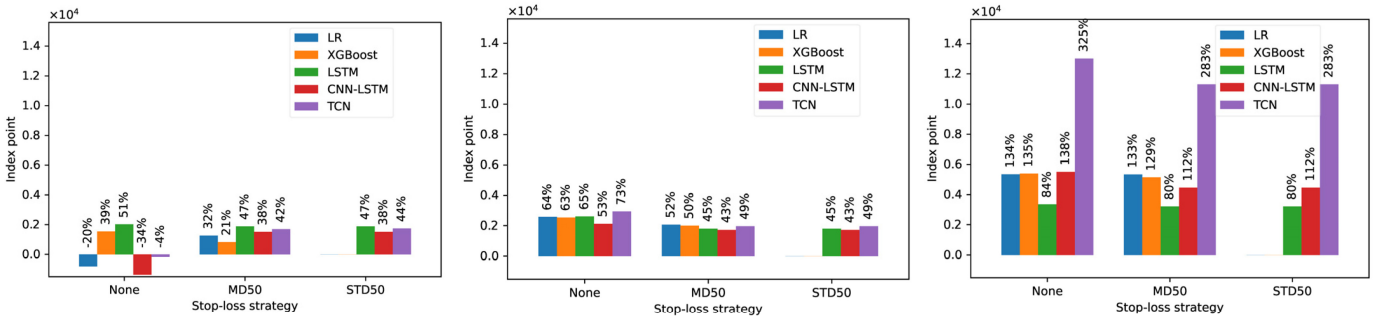


Figure 10. Compare the profit getting by using different stop-loss strategies in three years. (**a**) 2018. (**b**) 2019. (**c**) 2020.

6. Conclusions

The empirical results show that the proposed approach can be used in practice to support trading TX or MTX effectively, even with a simple trading strategy. External factors, i.e., global financial and economic factors outside Taiwan stock market, are useful predictor variables for TAIEX. Among the five models in this study, LSTM and TCN are the better choices. Especially, LSTM is probably more suitable for the low volatility period, while TCN can work effectively in a high volatility period. Nevertheless, stop-loss strategies are always necessary, especially for individual investors who can usually only afford a certain amount of loss and cannot continue to hold their positions in futures contracts if the price changes too much in an unfavorable direction. The STD strategy has the potential to improve profitability compared to the MD strategy. However, we also note that the STD strategy carries a higher risk for investors than the MD strategy because the stop-loss threshold of STD50 is always less than or equal to that of MD50, i.e., $V_{st}^{STD50} \leq V_{st}^{MD50}$, which leads to the risk that we can lose much more money if V_{st}^{STD50} is not a final limit for a deep drop before it recovers and rises to the expected value. In reality, additional strategies are needed to reduce this risk, e.g., if V_{st}^{STD50} is too low, V_{st}^{MD50} is used instead.

This study contains some limitations that can be improved in future work. First, the trading strategy is simple. We focused on trading long positions and bought only one contract on the first trading day of each month. Consequently, there could be a lot of time during the trading period when we did not hold a contract because there was no suitable contract. The waste of time can be even more severe with stop-loss strategies when the contracts sometimes have to be sold too early. Hence, a more sophisticated strategy can help improve the rate of return significantly. Second, deep learning models give us a convenient way to derive the standard deviation of MAT in the future without having to build separate models, and they can be used to develop more sophisticated stop-loss strategies. However, getting high accuracy in predicting standard deviation in this way is not straightforward. We suppose that further efforts to improve this accuracy could significantly improve trading performance in practice. Thus, it is worth paying attention to this aspect in future work. Third, feature engineering is also one of the aspects that can be further improved. We only tried to identify a group of practical external factors that are easy to collect, so other important external factors might be overlooked. In addition, just a few factors from the Taiwan market, i.e., TSM stocks and TWD exchange rates, were considered in the experiment. Indeed, more domestic elements and other types of data, such as technical indicators, keyword trends in search engines, or public market information, may help improve the prediction accuracy. Additionally, other unsupervised data representation methods such as the restricted Boltzmann machine or autoencoders can potentially enhance the performance of the predictive models. Fourth, LR and XGboost are very popular and generally have good performance; therefore, they can suffice to be used as simple baselines. However, these two algorithms cannot represent all conventional machine learning models. A comparison with other advanced methods, e.g., support vector regression, hidden Markov model, or fuzzy-based techniques, could be conducted in a future study to provide a more comprehensive view of the performance of the three deep learning models on this problem. Fifth, although common settings might be sufficient to essentially show the power of each deep network architecture, more intensive tuning of hyperparameters can help to evaluate the performance of each type of deep network more accurately. Finally, deep learning for time series and sequences is evolving very rapidly. Many techniques have been proposed and achieved the best results in recent benchmark studies, e.g., Inceptiontime [49], and ROCKET [50]. These techniques can be explored in future work to improve the current results further.

Author Contributions: Conceptualization, C.-H.L. and S.-M.Y.; methodology, C.-H.L., K.-S.T. and D.-A.H.; software, K.-S.T.; validation, K.-S.T. and D.-A.H.; formal analysis, D.-A.H. and K.-S.T.; investigation, K.-S.T., C.-H.L. and D.-A.H.; resources, S.-M.Y.; data curation, K.-S.T.; writing—original draft preparation, D.-A.H.; writing—review and editing, C.-H.L. and S.-M.Y.; visualization, D.-A.H.; supervision, S.-M.Y.; project administration, S.-M.Y. and C.-H.L.; funding acquisition, S.-M.Y.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the author.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. List of potential independent variables.

#	Variable	Symbol	Description	Types
1	AORD	^AORD	ALL ORDINARIES	World Indices
2	AXJO	^AXJO	S&P/ASX 200	World Indices
3	N225	^N225	Nikkei 225	World Indices
4	KOSPI	^KS11	KOSPI Composite Index	World Indices
5	SSE	000001.SS	SSE Composite Index	World Indices
6	SSEA	000002.SS	SSE A Share Index	World Indices
7	SZSE	399001.SZ	Shenzhen Component	World Indices
8	HSI	^HSI	HANG SENG INDEX	World Indices
9	KLSE	^KLSE	FTSE Bursa Malaysia KLCI	World Indices
10	STI	^STI	STI Index	World Indices
11	PSEI	PSEI.PS	PSEi INDEX	World Indices
12	JKSE	^JKSE	Jakarta Composite Index	World Indices
13	BSESN	^BSESN	S&P BSE SENSEX	World Indices
14	FTSE	^FTSE	FTSE 100	World Indices
15	GDAXI	^GDAXI	DAX PERFORMANCE-INDEX	World Indices
16	FCHI	^FCHI	CAC40	World Indices
17	SSMI	^SSMI	SMI PR	World Indices
18	DJI	^DJI	Dow Jones Industrial Average	World Indices
19	GSPC	^GSPC	S&P 500	World Indices
20	IXIC	^IXIC	NASDAQ Composite	World Indices
21	SOX	^SOX	PHLX Semiconductor	World Indices
22	GSPTSE	^GSPTSE	S&P/TSX Composite index	World Indices
23	MXX	^MXX	IPC MEXICO	World Indices
24	BVSP	^BVSP	IBOVESPA	World Indices
25	VIX	^VIX	CBOE Volatility Index	World Indices
26	IRX	^IRX	13 Week Treasury Bill	US Treasury bonds rates
27	TYX	^TYX	Treasury Yield 30 Years	US Treasury bonds rates
28	FVX	^FVX	Treasury Yield 5 Years	US Treasury bonds rates
29	TNX	^TNX	Treasury Yield 10 Years	US Treasury bonds rates
30	TSM	TSM	Taiwan Semiconductor Manufacturing Company Limited	Taiwan Company
31	MSCI	MSCI	MSCI Inc.	U.S. Company
32	CrudeOil	CL=F	Crude Oil May 21	Commodity
33	HeatingOil	HO=F	Heating Oil May 21	Commodity
34	NaturalGas	NG=F	Natural Gas May 21	Commodity
35	Gold	GC=F	Gold Jun 21	Commodity
36	Platinum	PL=F	Platinum Jul 21	Commodity
37	Silver	SI=F	Silver May 21	Commodity
38	Copper	HG=F	Copper May 21	Commodity
39	Palladium	PA=F	Palladium Jun 21	Commodity
40	TWDUSD	TWDUSD=X	TWD/USD exchange rates	TWD exchange rates
41	TWDCNY	TWDCNY=X	TWD/CNY exchange rates	TWD exchange rates
42	TWDJPY	TWDJPY=X	TWD/JPY exchange rates	TWD exchange rates
43	TWDHKD	TWDHKD=X	TWD/HKD exchange rates	TWD exchange rates
44	TWDKRW	TWDKRW=X	TWD/KRW exchange rates	TWD exchange rates
45	TWDEUR	TWDEUR=X	TWD/EUR exchange rates	TWD exchange rates
46	TWDCAD	TWDCAD=X	TWD/CAD exchange rates	TWD exchange rates

Table A2. Index points gained in the ideal case where MAT and $\hat{\sigma}$ could be predicted exactly.

Year	None	MD50	STD50
2018	2525	2352	2525
2019	8196	5157	6496
2020	16,746	12,084	12,084

Table A3. This table shows the values of the two terms ($V_1 = (\hat{V} - \hat{\sigma})$ and $V_2 = (V_0 - 50)$) that used to calculate the stop-loss threshold V_{st}^{STD50} based on the STD strategy for the three deep learning models. Each row corresponds with an MTX contract that was selected for buying in the simulation. The table also shows the values of $d_1 = (V_0 - V_{st}^{STD50})$ and $d_2 = (V_0 - V_{lowest})$, where V_{lowest} is the lowest value of the contract during trading time. For d_2, only the values with $V_{lowest} < 0$ are showed, because we are only interested in cases where there was a drop in value of the buying contract. From the columns d_1 and d_2, we can know which case the stop-loss threshold become functional.

Year	#	LSTM				CNN-LSTM				TCN			
		V_1	V_2	d_1	d_2	V_1	V_2	d_1	d_2	V_1	V_2	d_1	d_2
2018	1	10,840	10,556	50	-	11,417	10,535	50	-	10,612	10,535	50	-
	2	10,914	10,557	50	-	10,813	10,557	50	-	12,001	11,012	50	873
	3	10,792	10,849	107	17	11,758	10,809	50	380	10,028	10,551	573	-
	4	11,100	10,118	50	-	10,833	10,577	50	126	10,811	10,809	50	380
	5	10,795	10,545	50	-	10,883	10,373	50	-	12,818	10,118	50	-
	6	10,668	10,495	50	-	11,878	10,450	50	-	9850	10,545	745	-
	7	11,212	10,885	50	329	11,872	10,920	50	1229	10,605	10,450	50	-
	8	10,301	9692	50	126	11,959	10,935	50	1584	12,291	10,885	50	1534
	9	10,170	9951	50	375	11,967	10,889	50	1538	10,930	10,889	50	1538
	10					10,762	9692	50	126	9412	9699	337	133
	11					10,171	9951	50	375				
2019	1	9932	9657	50	388	9866	9657	50	388	9676	9657	50	388
	2	10,573	9895	50	-	10,003	9895	50	-	10,389	9895	50	-
	3	10,617	10,336	50	182	10,906	10,336	50	182	10,029	10,145	166	-
	4	10,576	10,613	87	36	11,043	10,627	50	50	10,447	10,519	122	-
	5	10,530	9978	50	-	11,069	10,895	50	603	10,606	10,551	50	421
	6	10,627	10,475	50	-	10,559	10,145	50	-	10,318	10,482	214	-
	7	10,806	10,500	50	-	10,852	10,655	50	-	10,530	10,778	298	-
	8	10,253	10,482	279	-	11,085	10,583	50	453				
	9					10,737	10,516	50	9				
	10					10,939	10,800	50	41				
2020	1	11,343	11,236	50	148	11,465	11,236	50	148	12,882	11,164	50	2690
	2	10,924	10,950	76	-	11,987	10,908	50	2434	11,944	10,960	50	2486
	3	8672	9401	779	-	10,413	9401	50	-	13,002	9401	50	-
	4	9641	10,058	467	-	10,327	10,276	50	-	13,725	10,276	50	-
	5	10,209	10,564	405	-	12,168	10,564	50	-	13,954	10,635	50	-
	6									13,985	11,206	50	-
	7									14,786	12,358	50	263
	8									15,114	12,444	50	344
	9									15,451	12,421	50	-
	10									13,967	12,357	50	-

References

1. Gandhmal, D.P.; Kumar, K. Systematic analysis and review of stock market prediction techniques. *Comput. Sci. Rev.* **2019**, *34*, 1–13. [CrossRef]
2. Henrique, B.M.; Sobreiro, V.A.; Kimura, H. Literature review: Machine learning techniques applied to financial market prediction. *Expert Syst. Appl.* **2019**, *124*, 226–251. [CrossRef]
3. Kumar, G.; Jain, S.; Singh, U.P. Stock Market Forecasting Using Computational Intelligence: A Survey. *Arch. Comput. Methods Eng.* **2021**, *28*, 1069–1101. [CrossRef]
4. Fama, E.F. Random Walks in Stock Market Prices. *Financ. Anal. J.* **1995**, *51*, 75–80. [CrossRef]
5. Sezer, O.B.; Gudelek, M.U.; Ozbayoglu, A.M. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Appl. Soft Comput.* **2020**, *90*, 106181. [CrossRef]
6. Bustos, O.; Pomares-Quimbaya, A. Stock market movement forecast: A Systematic review. *Expert Syst. Appl.* **2020**, *156*, 113464. [CrossRef]
7. Weng, B.; Ahmed, M.A.; Megahed, F.M. Stock market one-day ahead movement prediction using disparate data sources. *Expert Syst. Appl.* **2017**, *79*, 153–163. [CrossRef]
8. Wang, J.; Wang, J.; Fang, W.; Niu, H. Financial Time Series Prediction Using Elman Recurrent Random Neural Networks. *Comput. Intell. Neurosci.* **2016**, *2016*, 1–14. [CrossRef]
9. Persio, L.D.; Honchar, O. Artificial Neural Networks architectures for stock price prediction: Comparisons and applications. *Int. J. Circuits Syst. Signal Process.* **2016**, *10*, 403–413.
10. Chong, E.; Han, C.; Park, F.C. Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Syst. Appl.* **2017**, *83*, 187–205. [CrossRef]
11. Gunduz, H.; Yaslan, Y.; Cataltepe, Z. Intraday prediction of Borsa Istanbul using convolutional neural networks and feature correlations. *Knowl.-Based Syst.* **2017**, *137*, 138–148. [CrossRef]
12. Li, Z.; Tam, V. Combining the real-time wavelet denoising and long-short-term-memory neural network for predicting stock indexes. In Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence (SSCI), Honolulu, HI, USA, 27 November–1 December 2017; pp. 1–8.

13. Fischer, T.; Krauss, C. Deep learning with long short-term memory networks for financial market predictions. *Eur. J. Oper. Res.* **2018**, *270*, 654–669. [CrossRef]
14. Baek, Y.; Kim, H.Y. ModAugNet: A new forecasting framework for stock market index value with an overfitting prevention LSTM module and a prediction LSTM module. *Expert Syst. Appl.* **2018**, *113*, 457–480. [CrossRef]
15. Chung, H.; Shin, K.s. Genetic Algorithm-Optimized Long Short-Term Memory Network for Stock Market Prediction. *Sustainability* **2018**, *10*, 3765. [CrossRef]
16. Chen, L.; Qiao, Z.; Wang, M.; Wang, C.; Du, R.; Stanley, H.E. Which Artificial Intelligence Algorithm Better Predicts the Chinese Stock Market? *IEEE Access* **2018**, *6*, 48625–48633. [CrossRef]
17. Zhou, X.; Pan, Z.; Hu, G.; Tang, S.; Zhao, C. Stock Market Prediction on High-Frequency Data Using Generative Adversarial Nets. *Math. Probl. Eng.* **2018**, *2018*, 4907423. [CrossRef]
18. Zhong, X.; Enke, D. Predicting the daily return direction of the stock market using hybrid machine learning algorithms. *Financ. Innov.* **2019**, *5*, 1–20. [CrossRef]
19. Hoseinzade, E.; Haratizadeh, S. CNNpred: CNN-based stock market prediction using a diverse set of variables. *Expert Syst. Appl.* **2019**, *129*, 273–285. [CrossRef]
20. Sim, H.S.; Kim, H.I.; Ahn, J.J. Is Deep Learning for Image Recognition Applicable to Stock Market Prediction? *Complexity* **2019**, *2019*, 4324878. [CrossRef]
21. Wen, M.; Li, P.; Zhang, L.; Chen, Y. Stock Market Trend Prediction Using High-Order Information of Time Series. *IEEE Access* **2019**, *7*, 28299–28308. [CrossRef]
22. Lee, J.; Kim, R.; Koh, Y.; Kang, J. Global Stock Market Prediction Based on Stock Chart Images Using Deep Q-Network. *IEEE Access* **2019**, *7*, 167260–167277. [CrossRef]
23. Long, W.; Lu, Z.; Cui, L. Deep learning-based feature engineering for stock price movement prediction. *Knowl.-Based Syst.* **2019**, *164*, 163–173. [CrossRef]
24. Pang, X.; Zhou, Y.; Wang, P.; Lin, W.; Chang, V. An innovative neural network approach for stock market prediction. *J. Supercomput.* **2020**, *76*, 2098–2118. [CrossRef]
25. Kelotra, A.; Pandey, P. Stock Market Prediction Using Optimized Deep-ConvLSTM Model. *Big Data* **2020**, *8*, 5–24. [CrossRef] [PubMed]
26. Chung, H.; Shin, K.s. Genetic algorithm-optimized multi-channel convolutional neural network for stock market prediction. *Neural Comput. Appl.* **2020**, *32*, 7897–7914. [CrossRef]
27. Nabipour, M.; Nayyeri, P.; Jabani, H.; Mosavi, A.; Salwana, E.; Shahab, S. Deep learning for stock market prediction. *Entropy* **2020**, *22*, 840. [CrossRef]
28. Long, J.; Chen, Z.; He, W.; Wu, T.; Ren, J. An integrated framework of deep learning and knowledge graph for prediction of stock price trend: An application in Chinese stock exchange market. *Appl. Soft Comput.* **2020**, *91*, 106205. [CrossRef]
29. Nabipour, M.; Nayyeri, P.; Jabani, H.; Shahab, S.; Mosavi, A. Predicting Stock Market Trends Using Machine Learning and Deep Learning Algorithms Via Continuous and Binary Data; a Comparative Analysis. *IEEE Access* **2020**, *8*, 150199–150212. [CrossRef]
30. Lei, B.; Zhang, B.; Song, Y. Volatility Forecasting for High-Frequency Financial Data Based on Web Search Index and Deep Learning Model. *Mathematics* **2021**, *9*, 320. [CrossRef]
31. Hsieh, T.J.; Hsiao, H.F.; Yeh, W.C. Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. *Appl. Soft Comput.* **2011**, *11*, 2510–2525. [CrossRef]
32. Sun, W.T.; Chiao, H.T.; Chang, Y.S.; Yuan, S.M. Forecasting Monthly Average of Taiwan Stock Exchange Index. In *New Trends in Computer Technologies and Applications*; Springer: Singapore, 2019; pp. 302–309.
33. Ha, D.A.; Lu, J.D.; Yuan, S.M. Forecasting Taiwan Stocks Weighted Index Monthly Average Based on Linear Regression—Applied to Taiwan Stock Index Futures. In *Education and Awareness of Sustainability*; World Scientific: Singapore, 2020; pp. 431–435.
34. Tan, K.S.; Lio, C.H.; Yuan, S.M. Futures Trading Strategy based on Monthly Average Prediction of TAIEX by using Linear Regression and XGBoost. In Proceedings of the 7th IEEE International Conference on Applied System Innovation 2021, Alishan, Taiwan, 24–25 September 2021.
35. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
36. Olah, C. Understanding LSTM Networks. Available online: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (accessed on 30 September 2021).
37. Chevalier, G. LSTM Cell. Available online: https://commons.wikimedia.org/wiki/File:LSTM_Cell.svg (accessed on 30 September 2021).
38. Shi, X.; Chen, Z.; Wang, H.; Yeung, D.Y.; Wong, W.K.; Woo, W.C. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. *arXiv* **2015**, arXiv:cs.CV/1506.04214.
39. Sainath, T.N.; Vinyals, O.; Senior, A.; Sak, H. Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks. In Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), South Brisbane, Australia, 19–24 April 2015; pp. 4580–4584.
40. Donahue, J.; Hendricks, L.A.; Rohrbach, M.; Venugopalan, S.; Guadarrama, S.; Saenko, K.; Darrell, T. Long-term Recurrent Convolutional Networks for Visual Recognition and Description. *arXiv* **2016**, arXiv:cs.CV/1411.4389.
41. Livieris, I.E.; Pintelas, E.; Pintelas, P. A CNN-LSTM model for gold price time-series forecasting. *Neural Comput. Appl.* **2020**, *32*, 17351–17360. [CrossRef]

42. Bai, S.; Kolter, J.Z.; Koltun, V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv* **2018**, arXiv:cs.LG/1803.01271.
43. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.
44. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
45. Salimans, T.; Kingma, D.P. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. In Proceedings of the 30th International Conference on Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 901–909.
46. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
47. Zhong, X.; Enke, D. Forecasting daily stock market return using dimensionality reduction. *Expert Syst. Appl.* **2017**, *67*, 126–139. [CrossRef]
48. Weng, B.; Lu, L.; Wang, X.; Megahed, F.M.; Martinez, W. Predicting short-term stock prices using ensemble methods and online data sources. *Expert Syst. Appl.* **2018**, *112*, 258–273. [CrossRef]
49. Ismail Fawaz, H.; Lucas, B.; Forestier, G.; Pelletier, C.; Schmidt, D.F.; Weber, J.; Webb, G.I.; Idoumghar, L.; Muller, P.A.; Petitjean, F. InceptionTime: Finding AlexNet for time series classification. *Data Min. Knowl. Discov.* **2020**, *34*, 1936–1962. [CrossRef]
50. Dempster, A.; Petitjean, F.; Webb, G.I. ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. *Data Min. Knowl. Discov.* **2020**, *34*, 1454–1495. [CrossRef]