Applied Soft Computing Journal 152 (2024) 111213

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A portfolio trading system using a novel pixel graph network for stock selection and a mean-CDaR optimization for portfolio rebalancing

Milad Kamali Alamdari [a], Akbar Esfahanipour [a,*], Hossein Dastkhan [b]

[a] Department of Industrial Engineering and Management Systems, Amirkabir University of Technology, Tehran, Iran
[b] Department of Financial Engineering, Faculty of Financial Science, Kharazmi University, Tehran, Iran

HIGHLIGHTS

- A novel PGN model for image graph classification is proposed to predict BUY/SELL signals.
- Mean–Conditional Drawdown at Risk optimization model is developed for asset allocation.
- An ensemble feature selection algorithm with a continuous trend labeling has been utilized.
- Our model outperforms the benchmarks in both stock trend prediction and portfolio optimization.
- The results have been presented in terms of computational and financial performance metrics.

ARTICLE INFO

Keywords:
Financial forecasting
Graph neural network
Image classification
Conditional drawdown at risk
Portfolio optimization

ABSTRACT

Due to the complexity and dynamic nature of financial markets, portfolio management tasks require continuous adaptation with market intelligence. Furthermore, to make profitable trading decisions, it is essential to predict the future performance of the market. To that end, this study proposes a dynamic portfolio trading system in two stages: stock trend prediction and portfolio optimization. In the first stage, although many researchers proposed powerful classification models to predict BUY/SELL trading signals, some scholars only focused on converting time series' features into multichannel images without considering any spatial relationships between features. Considering the relational capacity of graphs, this study proposes a novel image-based classification model, called Pixel Graph Network (PGN), to analyze the constructed images with propagating information from all neighboring pixels, build up a representation of the images as special graphs, and get advantages of a graph-based learning algorithm. In the second stage, stocks with predicted BUY trading signals in the following days, are fed into the Mean–Conditional Drawdown at Risk (M-CDaR) optimization model to prevent any significant drawdown of the investment portfolio. Moreover, a hierarchical feature selection algorithm has been proposed, combined with a continuous trend labeling method to improve the efficiency of the training process. To demonstrate the superiority of our proposed PGN model in comparison with benchmarks, 18 stocks from the New York Stock Exchange (NYSE) are used for numerical experiments. The obtained results show the improvement and effectiveness of the proposed model in both stock trend prediction and portfolio optimization. In terms of financial performance, the proposed PGN+M-CDaR portfolio-based trading system achieves an average annual return of 112%, an average annual Sortino ratio of 10.78, and an average annual Sharpe ratio of 2.79 which are promising to achieve substantial profits while mitigating downside risk.

1. Introduction

The primary objective of this study is to propose a dynamic portfolio trading system in two stages: stock trend prediction and portfolio optimization. Therefore, we outline the essence of this study concerning its two main stages in the next two subsections, respectively.

* Correspondence to: Department of Industrial Engineering & Management Systems, Amirkabir University of Technology, No. 350, Hafez Ave, Valiasr Square, 1591634311 Tehran, Iran.
E-mail address: esfahaa@aut.ac.ir (A. Esfahanipour).

https://doi.org/10.1016/j.asoc.2023.111213
Received 5 September 2023; Received in revised form 21 December 2023; Accepted 22 December 2023
Available online 28 December 2023

1.1. Stock trend prediction

Forecasting the stock market is always a challenging task with numerous obstacles and intricacies. Since the price changes are highly chaotic [1], complex [2], volatile [3] and resultant of a dynamic combination of many factors such as economic, political, or even environmental factors such as the widespread COVID-19 pandemic which was leading to the bearish trend of markets in 2020. This challenge has captured the attention of many researchers, who have done their best to develop innovative approaches for predicting stock prices and returns. Some of these approaches have been regression methods [4–6], while others have focused on classification methods to predict future trends in stock prices [7,8]. Moreover, investors and for-profit practitioners are most interested in predicting the future trends of stock markets rather than a stock's return. In recent years, a spectrum of techniques has emerged to address the intricate challenge of stock market prediction. Initially, traditional statistical models were the go-to approach. However, with the advancement of technology, machine learning and deep learning techniques have emerged as promising solutions for handling the complex financial data involved in prediction tasks [9]. Neural networks, a common machine learning technique, have proven adept at capturing non-additive and non-linear characteristics of data, resulting in more accurate predictions [10]. While early neural networks had a shallow structure and struggled to model the data's complexity, the rise of deep learning in fields such as image processing and text analysis has piqued researchers' interest in applying these models to stock market prediction. Today, various deep learning models, including the restricted Boltzmann machine (RBM) [11], recurrent neural network (RNN) [12], long short-term memory (LSTM) [13,14], and convolutional neural network (CNN) [15,16], exhibit remarkable capabilities in forecasting stock trends. Of the previously mentioned models, CNN stands out as a multi-layer neural network architecture that simulates the operational principles of a biological visual system. It has shown remarkable effectiveness in capturing multi-scale localized spatial attributes, making it well-suited for image classification and recognition tasks.

1.2. Portfolio optimization

Markowitz mean-variance (MV) optimization technique is considered as the foundation of modern portfolio theory, aiming to maximize the expected return while minimizing the investment risk of a portfolio [17]. The model creates an efficient frontier that provides the optimal investment strategy at each level of expected return [18]. Meanwile, effective stock selection is critical for successful portfolio management [19], as individual investors in the stock market typically aim to estimate the future return of their stocks and then determine the optimal weight for each stock to construct their portfolio [20]. Many scholars have utilized the aforementioned machine learning and deep learning models to select stocks before portfolio formation, and have achieved satisfactory results [19,21–25]. This process is primarily based on modern portfolio theory [19,23,26], which offers various models for computing the optimal portfolio weights by optimizing one or more objective functions under different constraints.

2. Problem statement

In this section, the problem statement is explored in the first two subsections, covering the two stages of this study. Following that, the main contributions of this study are described in the third subsection.

2.1. Stock trend prediction

Recently, scholars have explored the transformation of stock market data into a visual representation and used CNNs to predict future trends. However, most existing image-based models do not consider the importance of spatial relationships between the pixels in images, leading to the inconsistency characteristics of the model's training process [27]. Accordingly, when using CNNs for image classification, it is essential to pay attention to the location of the pixels in the images to have a more robust and stable learning process. For the sake of clarity, most of the researchers in the stage of image formation [15,28–30] pointed out that the image's pixels (indicators) from the same intensity (similar color) should be in close proximity to have better convolutional operation for detecting the corresponding BUY/SELL patterns. Although most researchers have made much effort to respect the spatial arrangement of pixels in their images, there is still no guarantee that the corresponding arrangement is optimal and efficient enough to have a consistent learning process. Therefore, in the first stage, we develop a novel image-based model to predict a stock's trends which is called Pixel Graph Network (PGN) model, in which the graph construction of the pixels in the whole image can allow the network to capture non-local dependencies between pixels and lead to better performance on different tasks such as image segmentation and classification, where traditional CNNs may struggle to some extent.

2.2. Portfolio optimization

Despite the MV's usefulness, it has practical limitations, including imposed assumptions and computational complexities when dealing with larger asset scales. To address these issues, several alternative models have been proposed. For instance, Konno and Yamazaki [31] developed Markowitz's model using mean absolute deviation (MAD) risk measure. Similarly, Alexander and Baptista [32] proposed value at risk (VaR) risk measure instead of variance to estimate acceptable loss within a specific confidence level. However, because VaR can lead to numerous local optima and lacks the sub-additive property for risk management, the conditional value at risk (CVaR) model was put forth by Rockafellar and Uryasev [33]. Investors are often interested in determining the potential maximum loss and its probability, which makes the conditional value at risk measure more appropriate. Financial managers are mainly concerned with significant losses since it can lead to a loss of their assets and prevent further investments. The 2008 financial crisis and 2019 COVID-19 pandemic serve as examples where investment funds experienced a capital loss, taking some years to recover. As a result, investors are more cautious and try to avoid significant capital losses. The conditional drawdown at risk (CDaR) is a risk measure introduced by Chekhlov et al. [34] and further developed by Krokhmal et al. [35]. It calculates the fall in the value of a stock portfolio from its maximum value over a specific period. This measure accounts for the tail risks in the distribution of stock returns and addresses the limitations of Semi-variance and CVaR by considering the time frame of stock value reduction. The CDaR aims to evaluate the average of the worst $(1-\beta)\%$ losses and similarly follows the CVaR modeling except that the CDaR quantifies portfolio losses. CDaR risk measure is conservative in nature because it focuses on preventing any significant price reductions or drawdowns. It goes beyond other widely used risk measures by not only estimating the potential loss magnitude but also aiming to limit the likelihood of large drawdowns. Given the high volatility of the period of 2017–2023 due to events such as the COVID-19 pandemic and other substantial economic tensions, the CDaR risk measure can be suitable for portfolio optimization modeling. In this regard, we adopt the Mean-CDaR model for portfolio optimization and determine each asset's weight accordingly.

2.3. Research contributions

In summary, the main contributions of this study are as follows.

- Our proposed PGN model, designed for image classification based on the graph structure of pixels, integrates graph neural networks with time series images. This amalgamation enables the model to learn the dependencies between all corresponding pixels, facilitating the classification of images into BUY/SELL categories.
- To improve the efficiency of our asset allocation problem and mitigate the risk of substantial drawdowns, this paper integrates Mean-CDaR portfolio optimization with the PGN trend prediction results.
- Additionally, an ensemble feature selection and labeling algorithm are employed to enhance the performance of the proposed PGN model and benchmarks. This approach aims to augment knowledge discovery by strengthening the relationship between features and their corresponding labels.

The remainder of this paper is organized as follows: After the brief introduction and problem statement in Sections 1 and 2, In Section 3, we review some related works in the literature. In Section 4, the proposed methodologies are presented thoroughly. In Section 5, we outline performance evaluation to show the superiority of our proposed model to the common benchmarks. Finally, in Section 6, we discuss our key findings, theoretical and practical implications, limitations, and future directions of this study.

3. Literature review

Numerous studies have delved into stock trading investments using various computational models. However, this paper exclusively focuses on reviewing a selection of pertinent studies within the realms of both stock trend prediction and portfolio optimization, aligning with the specific scope of this study.

3.1. Machine learning models

Traditional machine learning models such as support vector machines (SVM), artificial neural networks (ANN), and hybrid techniques have been extensively utilized for forecasting stock trends. For instance, Parray et al. [36] used logistic regression (LS), SVM, and ANN to predict the following trend of stocks. Lee [37] proposed a model for predicting trends by combining SVM with a hybrid approach for selecting features, while Chiang et al. [38] employed particle swarm optimization (PSO) in conjunction with a neural network for predicting movements. Also, Picasso et al. [39] improved the balancing technique and applied ANN, SVM, and random forest (RF) for trend prediction. Zhang et al. [40] explored ensemble-based models such as genetic algorithm (GA), probabilistic SVM, and AdaBoost algorithm for predicting stock movements. Özorhan et al. [41] first utilized a modified Zigzag technical indicator to identify stock trend motifs and then employed SVM for motif classification. In addition, Lei [42] introduced the wavelet neural network method (WNN) for trend prediction, and Bisoi et al. [43] devised a model that incorporated extreme learning machine, variational mode decomposition, and differential evolution at the same time for stock movement prediction. Recently, with the increasing computational capacity of certain machine learning algorithms, new approaches for predicting stock trends have emerged. In the following subsection, we will discuss the most frequently utilized deep learning models found in the literature for trend prediction.

3.2. Deep learning models

Deep learning is a type of machine learning technique that employs multiple layers with distinct functionalities to model data at a high level of abstraction. This approach ultimately results in superior performance compared to its shallow counterparts [44]. In the realm of stock trend prediction, a range of advanced deep learning techniques have been employed. For example, Nelson et al. [13] utilized LSTM with historical price and technical indicators to forecast future stock trends. Liang et al. [11] combined RBM and other classifiers to forecast short-term trends. Meanwhile, Chen et al. [14] analyzed public mood and emotion in text data with LSTM to predict stock trends. Moews et al. [45] introduced deep feed-forward neural networks combined with step-wise linear regressions with exponential smoothing as a feature engineering process. Furthermore, Zhao et al. [12] applied an attention mechanism on RNN, LSTM, and GRU for stock trend prediction. Naik and Mohan [46] employed DNN to identify stock movement trends by considering technical indicators and candlestick data values. Moreover, Wu et al. [47] investigated the impact of various labeling techniques on prediction metrics and utilized different models, including LSTM and GRU.

The remarkable success of CNNs in the field of computer vision has attracted researchers' interest in applying image-based models to address stock trend prediction tasks. Several researchers have transformed stock data into images using either 2D or 3D CNNs. For instance, Sezer and Ozbayoglu [15] developed a novel algorithmic trading approach by transforming technical indicators into 2D images based on CNN, and again Sezer and Ozbayoglu [48] utilized 2D stock bar chart images without any additional features with CNN to forecast trends. Long et al. [49] employed CNN to extract deep features in conjunction with representing trading behavior patterns, generated by three matrices. Hao and Gao [50] investigated characteristics of price series with different time scales through various layers of CNN to predict the direction of movements. Lastly, Sinha et al. [29] developed an ensemble learning to incorporate the influence of one company on another in conjunction with 3D CNN to predict the trend of different stocks from different market sectors.

Most of these image-based models tend to overlook an important aspect, which is the process of creating each image and the locality of neighboring pixels in the specific region, where the particular patterns of BUY/SELL classes are detected by the CNN's convolution operation. As far as we know, the process of times series' image formation was behind the scenes of the researchers, leading to an awkward learning process to some extent. To ensure efficient and consistent learning, it is crucial to overcome the major challenge of identifying groups of pixels that form a region with similar intensity via a separate leaning-based algorithm. In order to address this issue, this paper introduces a novel approach that has not been previously explored. For the sake of clarity, we create each image on graphs, where each pixel of the image is considered as a node of the graph, and the edges as the connection between the corresponding pixels. This type of image conversion causes its pixels to have non-Euclidean structured characteristics, which are designed to capture more spatial relationships between every two pixels in the entire image rather than convolutional operation on the limited part of the image, denoted by filter size of CNNs. Accordingly, when dealing with non-Euclidean structured data of varying sizes and dimensions, we must consider learning algorithms based on graph-structured data. Graph neural network (GNN), which can effectively analyze these kinds of data, is gradually favored by many scholars, and studies in this area are expanding and becoming hot topics these days. Graph neural networks have shown excellent performance in processing graph-structured data and have been studied in financial applications such as stock market prediction [51], loan default risk prediction [52,53], recommender system of e-commerce [54], fraud detection [55], and event prediction [56].

3.3. Portfolio optimization

Portfolio optimization is a critical task in finance that aims to maximize returns while minimizing risks. Markowitz's mean-variance (MV) model is one of the most widely used models for portfolio optimization [57]. This model enables investors to balance risk and return trade-offs. In financial analysis, numerous researchers have proposed diverse models for stock selection and portfolio optimization. Nevertheless, the MV optimization model has its limitations, prompting the introduction of several extensions and improvements in the corresponding literature. For instance, some studies have focused on multi-period portfolio selection [58–61], alternative risk measurements [31–33], and real-world constraints [62,63]. Meanwhile, due to the abnormal chaotic and non-linear characteristics of assets in the stock market, an initial selection of high-quality assets is more crucial than applying complex portfolio optimization models. In the last few years, some studies have explored using artificial intelligence techniques for portfolio optimization in the stock market. To address this issue, some researchers have proposed methods that integrate asset selection with portfolio optimization models. For instance, Huang [21] used Support Vector Regression (SVR) and Genetic Algorithms (GAs) to forecast upcoming returns of stocks, select top-ranked, and build an equally weighted portfolio. Krauss et al. [22] analyzed the advantages of several models, including Gradient-Boosted Trees, Random Forest, and deep neural network, and found that a simple equally-weighted ensemble model could have better results. Fischer and Krauss [64] used LSTM neural networks to predict trends of S&P 500 stocks, selected to be in the optimal portfolio, and their results showed that LSTM outperform other memory-free models in portfolio optimization. However, these models exclusively employ simplistic methods, such as threshold-based and equally-weighted approaches in portfolio construction. These methods overlook the individual risk profiles of each stock, resulting in an imbalanced distribution of expected return and risk across the portfolio. In response to this limitation, several recent studies have addressed the need for more nuanced approaches. Paiva et al. [23] developed an algorithmic day trading investment model using support vector machine (SVM) in conjunction with the MV model, and concluded that the corresponding system outperforms other benchmark models in terms of returns and risks. Wang et al. [19] utilized a fusion approach of LSTM and MV for the same intention to show the outperformance in cumulative returns, Sharpe ratio, and average return-to-risk ratio when compared to other benchmark models. Ma et al. [65] developed a strategy that integrates return prediction during portfolio formation using machine learning models such as Random Forrest (RF) and SVR, along with three deep learning models of LSTM, DMLP, and CNN. Chen et al. [66] established a new model for constructing portfolios called IFA-XGBoost. This approach combines eXtreme Gradient Boosting (XGBoost) with an enhanced firefly algorithm (IFA) to forecast stock prices and utilizes the MV model for selecting portfolios. Chaweewanchon and Chaysiri [67] introduced a novel portfolio construction model that integrates CNN and bidirectional LSTM for stock price prediction and the MV model for portfolio optimization. Chen et al. [68] combined various machine learning models such as SVM, LSTM, RF, extreme learning machine (ELM), and back propagation neural network (BPNN) with the modified mean–variance (MMV) model to determine the asset allocation. Behera et al. [69] introduced a hybrid model that employs various machine learning models, including RF, AdaBoost, SVR, KNN, and ANN, to identify stocks with the highest predicted returns. Subsequently, the mean–VaR portfolio optimization model is utilized for portfolio selection. Ma et al. [70] utilized an autoencoder (AE) for feature extraction and the LSTM model for predicting stock returns. These predictions were subsequently integrated into a portfolio optimization model, alongside a worst-case omega model. The evident superiority of their model over equally weighted portfolios and other prediction-based portfolio optimization models underscores its effectiveness. Ashrafzadeh et al. [71] employed a hybrid approach combining

Table 1
The position of the current study among similar studies in the portfolio selection literature.

Reference	Modeling approach		Prediction type		Types of inputs		Special characteristics of prediction		Special characteristics of portfolio optimization		Additional features		
	Prediction model	Portfolio optimization model	Regression	Classification	Lagged return observations	Technical indicators	Graph-bases learning process on inputs	Spatial Relations between inputs	Estimating the potential loss magnitude	Limiting the likelihood of large drawdowns	Transaction costs	Feature selection	Labeling algorithm
[23]	SVM	MV		✓	✓	✓					✓		
[19]	LSTM	MV	✓		✓	✓					✓	✓	
[66]	IFA-XGBoost	MV	✓		✓	✓					✓		
[65]	SVR, LSTM, CNN, DMLP, ARIMA	Omega	✓		✓						✓		
[67]	CNN-BiLSTM	MV	✓		✓						✓		
[68]	RF, SVR, LSTM, ELM, BPNN	modified MV	✓		✓	✓					✓		
[69]	RF, XGBoost, AdaBoost, SVR, KNN, ANN	M–VaR	✓		✓								
[70]	Autoencoder-LSTM	The worst-case Omega	✓		✓						✓		
[71]	PSO-CNN	MV	✓		✓	✓					✓	✓	
This study	**PGN**	**M-CDaR**		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

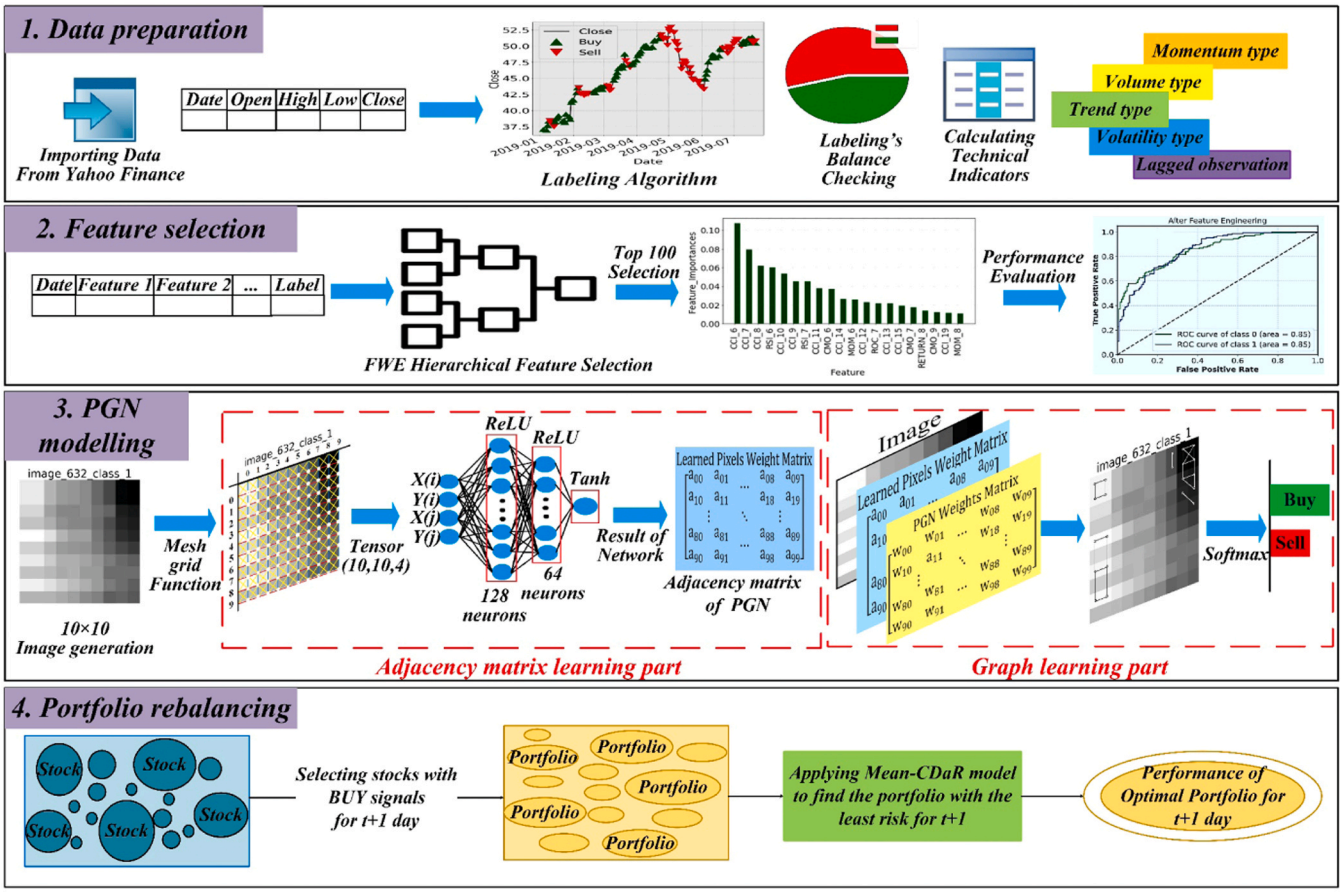

Fig. 1. The overall framework of our proposed portfolio trading system.

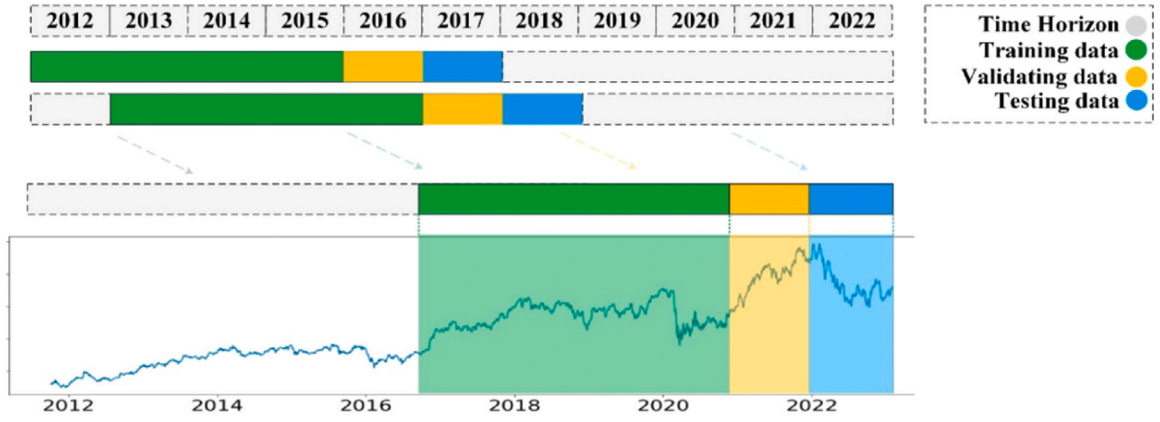

Fig. 2. Training, validating and testing approach.

Fig. 3. Labeling results.

a convolutional neural network with hyperparameters optimized through particle swarm optimization (PSO-CNN) for stock pre-selection. In this phase, the computational complexity is reduced by training the CNN on clustered stocks using the K-means method, as opposed to training on individual stocks. Subsequently, a mean–variance with forecasting (MVF) model is employed for portfolio optimization.

All the aforementioned studies indicate that combining artificial intelligence techniques with modern portfolio optimization yields more satisfactory performance than using each technique in isolation. However, traditional portfolio optimization models have been paired with commonly used predictive models may not be well-suited for short-term practical investments to make a reasonable profit and avoid substantial drawdowns. This underscores the necessity to explore more effective approaches that integrate novel predictive models, resulting in higher accuracy, with efficient portfolio optimization techniques, thereby leading to reduced drawdowns. As a conclusion to this section, Table 1 is provided to contextualize the position of this study in the related literature.

4. The proposed methodologies for the portfolio trading system

The overall framework of the proposed portfolio trading system is illustrated in Fig. 1. Building upon this framework, this section is organized into four main sub-sections, each dedicated to developing a specific contribution of this study. In the first subsection (4.1. Data preparation), the trading system receives OHLCV data. Using these variables, it computes various technical indicators and constructs additional features. It should be noted that these features are calculated for different periods ranging from 6 to 20 days, and used for daily trading. Longer ranges can be chosen for models aiming for long-term trades which are weekly or monthly. Additionally, to ensure that the classes are balanced and to simulate real-world trading conditions, a labeling algorithm is implemented that considers the balance rate and the total transaction cost for each BUY/SELL signal. In the second subsection (4.2. Feature selection), we aim to select the appropriate features by implementing a hierarchical approach. This approach enables us to not only retain the relevant statistical features but also learn additional features that are aligned with the corresponding labeling algorithm. In the third subsection (4.3. PGN modeling), our proposed image-based model is discussed as an effective classification model that outputs more accurate BUY/SELL trading signals. In the fourth subsection (4.4. Portfolio rebalancing), stocks with a BUY signal generated from our PGN model are suggested for portfolio formation to minimize drawdowns.

4.1. Data preparation

Numerous scholars have concurred that it is not practical for individual investors to manage portfolios with tens of thousands of different stocks [19,23,66,67]. In light of this, the present study selects eighteen stocks from the NYSE market as sample data, a sufficiently large set for preliminary asset selection before forming an optimal portfolio. The selected stocks include “Goldman Sachs Group” (GS), “Salesforce” (CRM), “Walt Disney” (DIS), “Merck & Co” (MRK), “McDonald” (MCD), “Cisco Systems” (CSCO), “Travelers” (TRV), “Caterpillar” (CAT), “Johnson & Johnson” (JNJ), “International Business Machines” (IBM), “UnitedHealth Group” (UNH), “Procter & Gamble” (PG), “3 M” (MMM), “Coca-Cola” (KO), “Chevron” (CVX), “Verizon Communications” (VZ), “JPMorgan” (JPM), “Visa” (V). We obtained OHLCV data for randomly selected stocks from Yahoo Finance,[1] covering the period between January 1, 2012, and January 1, 2023, for training and testing our proposed model and benchmarks. To facilitate this, we implemented a sliding window approach, utilizing a 5-year period for training and a subsequent 1-year period for testing.

For example, the initial training span is from 2012 to 2016, with the testing period from 2017 to 2018. Subsequently, we advanced both the training and testing periods by one year, retraining the model and testing with the following year, such as the training period from 2013 to 2017 and the testing period from 2018 to 2019. This process is iterated for every year between 2017 and 2022. Fig. 2 provides an illustration of our sliding training and testing approach. From the overall 5-year training dataset, we reserved the last year for validating data and optimizing models’ hyperparameters. This window-based approach used in this study, based on [15], involves removing older and less relevant data and incorporating more recent data to update the model, which is particularly useful in the dynamic and non-stationary nature of the financial time series analysis. Additionally, using the sliding window approach allows for the evaluation of the model in different time periods with different events, demonstrating the effectiveness and generalizability of the model across the corresponding periods.

Supervised learning models require balanced labeled data for training, which is mandatory for more accurate prediction without bias. Labeling financial time series data is challenging due to the non-linear nature of the data, especially in the short term. The labeling algorithms used in several articles [15,29,72] are inefficient for the learning process. These approaches have several flaws [47]. Firstly, the corresponding short-term labeling method leads to a complicated and unbalanced learning process. Secondly, this method does not consider the transaction costs, which can lead to closing positions at a loss even if the model has 100% accuracy. Therefore, in this study, we utilize a labeling method called "Continuous Trend Labeling," which takes into account transaction costs and balance rate, ensuring that all trades made within the corresponding window size result in a profit in conjunction with having balanced BUY/SELL labels to some extent. The pseudo-code of this approach is presented in Algorithm 1.

Algorithm 1. Continuous Trend Labeling Algorithm.

[1] https://finance.yahoo.com/

Input: $Return\ of\ closed\ prices\ \ R = \{r_1, r_2, \ldots, r_{dataset-1}\}$
Output: $Labelling\ data\ set\ L = \{l_1, l_2, \ldots, l_{dataset-windowSize}\}$

Requirements:
$windowSize = \mathbf{19}\ \boldsymbol{days}$
$LABEL\ [t = 0] = \boldsymbol{SELL}$
$Balance\ rate(br) = \mathbf{0.005}$
$Total\ transaction\ costs(ttc)$:
$Cost\ for\ buying(cb) = \mathbf{0.5}\ \ precentage\ of\ transaction\ volume$
$Cost\ for\ selling(cs) = \mathbf{1}\ \ precentage\ of\ transaction\ volume$

```
for t ϵ days do
    if LABEL [t − 1] == SELL then
        if return [t + 1] < br then
            LABEL [t] = SELL
        end
        if return [t + 1] ≥ ttc + br then
            LABEL [t] = BUY
        end
        else:
        for i ϵ range (2, windowSize) do
            if(And(return [t + i] ≥ i ∗ br)) then
                if (Or(return [t + i] ≥ i ∗ br + ttc )) then
                    LABEL [t] = BUY
                end
            else:
                LABEL [t] = SELL
        end
        else:
            LABEL [t] = SELL
    end
    if LABEL [t − 1] == BUY
        if return [t + 1] < br − ttc then
            LABEL [t] = SELL
        end
        if return [t + 1] ≥ br then
            LABEL [t] = BUY
        end
        else:
        for i ϵ range (2, windowSize) do
            if(And(return [t + i] < i ∗ br)) then
                if (Or(return [t + i] < i ∗ br + ttc )) then
                    LABEL [t] = SELL
                end
            else:
                LABEL [t] = BUY
        end
        else:
            LABEL [t] = BUY
        end
    end
```

For the sake of clarity, Algorithm 1 is designed to determine BUY or SELL labels for a given window size. This decision is contingent on the previous day's label and the expected return of the corresponding stock in the next 19 days. In this regard, to address the crucial aspects of "no early BUY the stock" and "no late SELL the stock," which are consistently overlooked in existing labeling approaches, a set of conditions is formulated. If the previous day's label for a stock is SELL, it indicates that the investor does not own the stock and has no intention to BUY it. Therefore, for the label to change from SELL to BUY, the expected return of the stock over the next 19 days must be attractive enough to justify the transaction cost with more additional returns. This additional return is considered as a balancing rate which is adjusted to be 0.005. Balancing rate consideration lets us be free from using under sampling and over sampling techniques, which are not well-suited for outstanding results for time series [28,29]. Conversely, if the previous day's label for a stock is BUY, it means that the investor already owns the stock and is seeking a suitable time to SELL. Thus, the conditions are designed to limit an early SELL if the stock's return in the next 19 days is unsatisfactory. Trade-offs between balance rate and time window size involve finding a balance that aligns with the goal of achieving balanced dataset in each 4-year training dataset while also reducing the frequency of successive position switches. This aims to minimize transaction costs and prevent the models from becoming confused during the learning process. A longer time window with a higher balance rate has the potential to delayed position switches, potentially missing out on opportunities in rapidly changing markets. On the other hand, a shorter time window with a lower balance rate is more responsive to short-term trends, albeit at the cost of paying transaction fees without gaining significant benefits. Acknowledging that these parameters are determined via trial and error highlights the iterative nature of developing and fine-tuning trading algorithms [15,29]. Fig. 3 demonstrates the labeling process with its balance checking for the Salesforce stock. The results of the corresponding labeling and balance checking for the remaining 17 stocks are also provided in the supplementary material file.

4.2. Feature selection and image generation

Predicting the movement of stock prices is more complex and complicated. In order to accurately predict stock price movements, it is important to extract and select features that contain useful information. This makes feature extraction and selection a critical component in predicting the direction of stock prices due to the presence of uncertainties in the prediction process. To construct additional features from the OHLCV data, we utilize technical indicators to examine historical information about stocks and estimate their future trends. Based on Peng et al. [73] study on the comprehensive technical indicators upon deep learning models, we select 35 informative technical indicators (TSI, UO, UI, ATR, MI, KAMA, FI, EOM, TMA, VWAP, MOM, DC, KC, BB, AROON, NVI, OBV, D/A, PSAR, DMI, CFMI, PPO, CCI, ROC, CMO, MACD, WILLIAMS%R, RSI, SO, HMA, WMA, TEMA, DEMA, EMA, SMA) with all 6 to 20 lags which are appropriate for daily trading. Also, based on [19,23,66,68], 15 lagged return observations are beneficial for portfolio formation with stock selection using prediction models. It is obvious that the optimal lags for technical indicators are important for better prediction. To generate 10×10 images, we have to reduce the number of features to 100 in an efficient way. To this end, we propose hierarchical feature selection in two steps to guarantee the quality of features used to build the main PGN model and benchmarks. Algorithm 2 demonstrates our two-step feature selection process. In the first step as a filtering part, all 35 technical indicators with all 6 to 20 lags and 15 lagged return observations are tested based on the following three statistical tests, and the features which are meet all three statistical tests requirements will be considered as the inputs for the second step. These three statistical tests used in the first step to filter the features are mentioned below according to [74]:

4.2.1. Variance analysis

This statistical test employs the F-value measure to assess the significant difference between the averages of each group when numerical features are grouped based on the target vector (labels). In the case of a binary target vector, such as BUY and SELL classes, paired with a quantitative feature like a technical indicator, the F-value helps determine whether the average value of that indicator for the BUY class is different from the average for the SELL class. If there is no significant difference, it implies that the indicator does not contribute to predicting BUY and SELL signals, and consequently, the corresponding indicator will be eliminated.

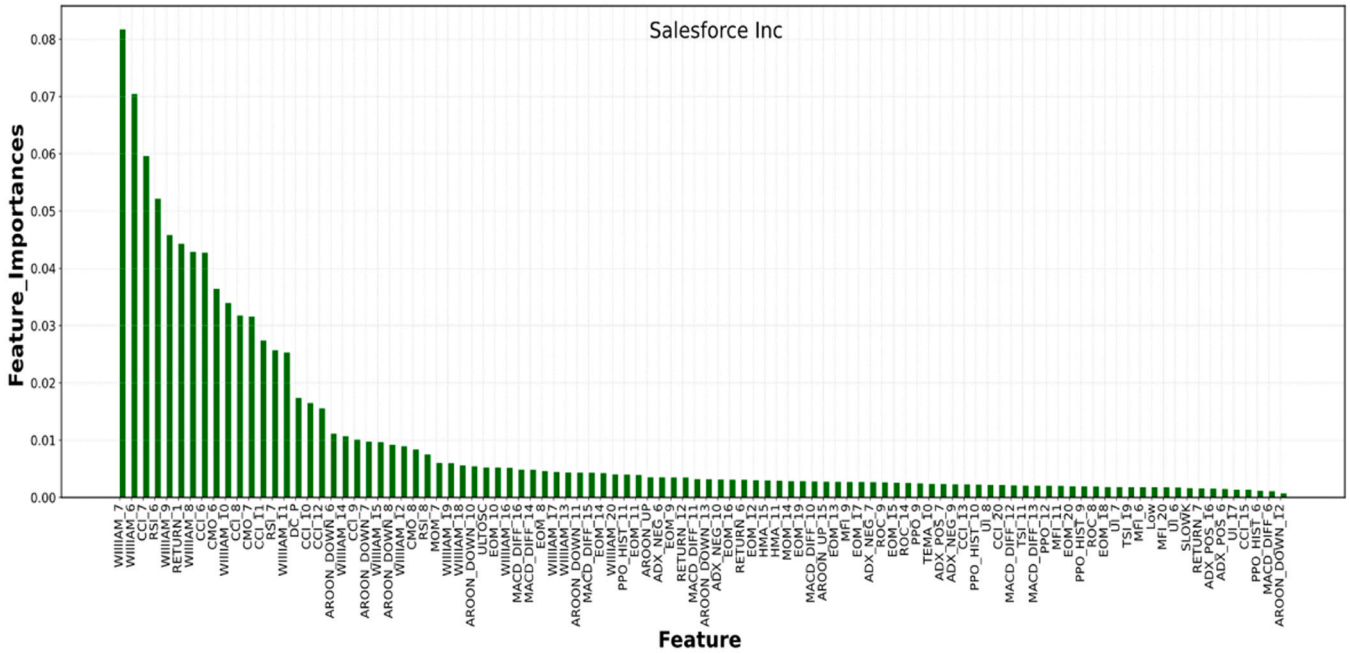

Fig. 4. Feature selection results for Salseforce stock.

4.2.2. Chi-square

By calculating the Chi-square statistics between a feature and the target vector, we can measure the independence between them. If the target is independent of a particular feature, it becomes irrelevant to our analysis because it lacks significant information for classification. Conversely, if these two features show high correlation, they are considered essential for training the model.

4.2.3. Mutual information

Mutual information is a crucial concept in information theory. It measures the information shared between the technical indicator and the corresponding label. It is intimately linked to the entropy of individual features, which quantifies the uncertainty or unpredictability of their outcomes. Mutual information helps us understand how much knowledge of one feature reduces the uncertainty of labels, leading to a deeper understanding of the relationship between them. This statistical test is essential, and it can determine how suitable a feature is for target estimation.

In the second step, we pass the selected features from the first stage through the Wrapper and Embedded methods. At this step, we make the features removed by random forest-based recursive feature elimination (RF-RFE) until we obtain 100 features. For the sake of simplicity, the detailed and operational steps performed in the entire feature selection algorithm are mentioned in the form of pseudo code in Algorithm 2. This Algorithm is executed for each stock within the portfolio, resulting in selecting top 100 features for each stock. Fig. 4 shows the results of our hierarchical feature selection for Salesforce stock. This histogram shows the interpretability analysis in our feature selection algorithm, which could help us find which feature has the greatest impact on the accuracy of prediction results [19]. The results of the corresponding feature selection for the remaining 17 stocks are also provided in the supplementary material file.

In order to ensure consistent and meaningful image representations, the location of indicators is significantly considered. Based on [15,29], By clustering the selected indicators based on their behavior (such as oscillators or trends) and arranging these indicators in close proximity, the resulting images provide a cohesive visual representation. Fig. 5 showcases some 10 × 10 BUY/SELL images generated during the image creation phase.

Algorithm 2. FWE Hierarchical Feature Selection Algorithm.

Input: $Original\ indicator\ set\ M = \{l_1, l_2, \dots, l_m\}$
Output: $Optimal\ indicator\ subset\ N = \{l_1, l_2, \dots, l_n\}$

Input of layer 1: $Original\ indicator\ set\ \{l_1, l_2, \dots, l_m\}\ AND\ target\ values\ for\ training\ section.$

for $1 \leftarrow index\ to\ m$ ***do***

STEP 1

$\{l_i\}_{index=1}^{i} \leftarrow List\ of\ \boldsymbol{i}\ indicators\ with\ high\ \boldsymbol{F-value}\ in\ accordance\ with\ target\ values.$

$\{l_i\}_{index=1}^{j} \leftarrow List\ of\ \boldsymbol{j}\ indicators\ with\ high\ \boldsymbol{Mutual-information}\ for\ any\ kinds\ of\ statistical\ dependencies\ with\ target\ values.$

$\{l_i\}_{i=1}^{k} \leftarrow List\ of\ \boldsymbol{k}\ indicators\ with\ high\ \boldsymbol{Chi-square}\ p-values\ relative\ to\ the\ target\ values.$

end

Output of layer 1: $S = \{l_i\}_{index=1}^{p} = \bigcap_{index=1}^{i,j,k} \{l_i\} \leftarrow Select\ S\ Bests$

Input of layer 2: $training\ set\ of\ features\ in\ \{l_i\}_{index=1}^{p}$

$P = \{1, 2, \dots, p\} \leftarrow Number\ of\ feature\ to\ select$

for $1 \leftarrow index\ to\ p\ in\ S$ ***do***

STEP 2

$Perform\ \boldsymbol{RFE}\ and\ select\ p\ features \rightarrow F^*$

$Train\ with\ Random\ Forest\ using\ F^*$

$Compute\ the\ classification\ accuracy\ of\ model \rightarrow r_{N_p}^2$

if $r_{N_p}^2 > r_{N_{p-1}}^2$ ***then***

$F^{**} \leftarrow F^*$

end

$S = F^*$

end

Output of layer 2: $Optimal\ indicator\ subset\ F^{**} = \{l_1, l_2, \dots, l_n\}\ with\ the\ highest\ accuracy$

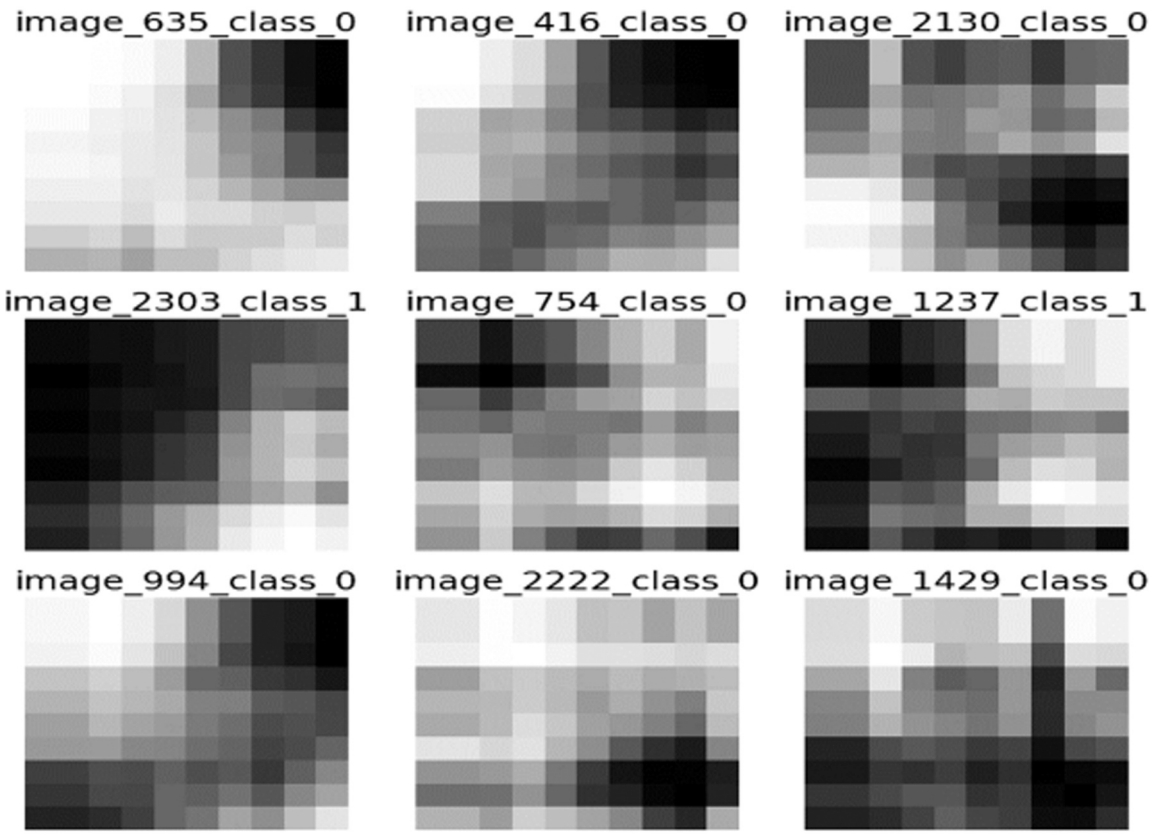

Fig. 5. 10 × 10 labeled images after image creation phase.

4.3. PGN modeling

Convolutional neural networks (CNNs) have performed excellently in various image classification tasks. These networks learn to classify images based on a pixel-grid representation. However, they have two limitations based on [27]: 1) The filters in the first CNN layer treat all pixels equally and process them in a predetermined order from top-left to bottom-right; 2) CNNs require images to be of the same size, leading to resizing and uniform down sampling. Unfortunately, this uniform down sampling may not be optimal, considering real-world data exhibits natural variations in spatial and multiscale features. When training a CNN on images, we implicitly depict images on a graph. This grid remains consistent for all training and test images, maintaining uniformity with each pixel connected to its neighbors in the same manner across all images (having the same number of neighbors, edge length, etc.). Consequently, the regular grid graph lacks distinguishing information to differentiate one image from another. Although CNNs still dominate in image classification, several promising studies have explored using GNNs for image classification [75]. In this regard, we develop a novel Pixel Graph Network (PGN) based on two main steps: Pixels' adjacency matrix leaning part and the graph neural network part for the learned adjacency matrix. At first, adjacency matrix (often denoted as A) is learned based on the pixels' coordinates and their intensities in the image. To incorporate neighbor aggregation in graph neural networks, the graph structure is often represented by an adjacency matrix, which describes the connectivity between nodes in the graph. This adjacency matrix is created with the inductive bias that connects pixels with similar colors with higher weights, while pixels with significant color differences have no or thin connections. This bias is motivated by the observation that adjacent pixels in natural images often correspond to the same object or interact closely. Besides having only 100 features of X in our 10 × 10 image, we now have the adjacency matrix A with values ranging from 0 to 1. It's crucial to emphasize that when we identify our input as a graph, we presume the absence of a canonical order of nodes that remains consistent across all other graphs in the dataset. In the context of images, this implies that pixels are assumed to be randomly shuffled and determining a canonical order of nodes is unsolvable in practical terms. To address this, we can utilize the adjacency matrix A, incorporating it into the GNN as follows:

$$X^{(l+1)} = AX^{(l)}W^{l} \tag{1}$$

Where $X^{(l)}$ is the image, and W^{l} is a learnable parameter of the layer l. To ensure that each row in the adjacency matrix A matches the properties of the corresponding node in X, we use a normalized version of A, denoted as $\bar{A}$. Normalizing A is vital because it allows us to capture the average characteristics of neighboring nodes. Typically, A is normalized using the equation $\bar{A} = A/\Sigma_i A_i$. Based on Eq. (1) the corresponding operation is similar to a convolutional neural network with a fixed Gaussian filter that remains unchanged during the training stage. This filter essentially smooths the image, which does not provide any specific advantages in solving our problem of classifying BUY/SELL images. To enhance the performance of GNNs on regular graphs like images, instead of using a predefined Gaussian filter, we employ a learning process to predict edges between each pair of pixels using a Meshgrid function as shown in Fig. 6.

Meshgrid function is used to create a rectangular grid from two one-dimensional arrays representing cartesian indexing or matrix indexing. It returns two-dimensional arrays representing the X and Y coordinates of each point on the grid, which can be helpful for various purposes, such as evaluating functions on a grid. For our 10 × 10 input image, the Meshgrid function generates two arrays: one representing the column coordinates and the other representing the row coordinates of each pixel. These arrays are then stacked along a new third axis, resulting in a new array with dimensions $(100,2)$, where each row represents the (x, y) coordinates of a pixel. Specifically, the first tensor is created for both

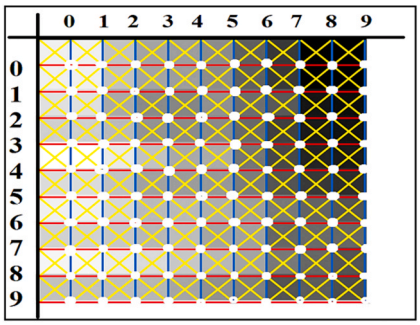

Fig. 6. Meshgrid function on image creation phase.

rows and columns, forming a tensor with dimensions $(10, 10, 2)$. The second tensor is similar to the first but with dimensions rearranged as (y, x). Both tensors are then normalized, resulting in tensors with dimensions $(100, 2)$. Finally, the two tensors are concatenated along the last dimension, yielding a tensor with dimensions $(100, 100, 4)$. This tensor represents the proximity matrix and relationships between pixels in the image. By learning to predict edges between pixels based on their coordinates, the model can effectively capture the spatial and non-local relationships between pixels, leading to improved performance in tasks such as image classification. The edge prediction for each pair of pixels in the adjacency matrix is learned by the following network:

$$h^{(0)} = [x_i, y_i, x_j, y_j] \in R^4, \forall i, j \in \{0, 9\}, i \neq j \tag{2}$$

$$h^{(1)} = ReLU(W^1 h^{(0)} + b^1) \in R^{128} \tag{3}$$

$$h^{(2)} = ReLU(W^2 h^{(1)} + b^2) \in R^{64} \tag{4}$$

$$h^{(3)} = ReLU(W^3 h^{(2)} + b^3) \in R \tag{5}$$

$$y = Tanh(h^{(3)}) \in [-1, 1] \tag{6}$$

Where x_i, y_i, x_j, y_j are the normalized coordinates of two i, j pixels, and W^l, b^l are learnable parameters and the bias of each layer l, and R represents the number of neurons in each layer, respectively. The adjacency matrix is learned via a neural network by training it to predict the edge weights between any two nodes (i.e., pixels in the image). For the sake of clarity, the adjacency matrix tensor is fed into a fully connected neural network with three layers with *ReLU* activation functions. The output of the last layer is passed through a *Tanh* function to constrain it to the range $[-1, 1]$, which produces the edge weights between the pixel pairs. During the training phase, the parameters of the neural network (i.e., the weights and biases of the layers) are updated using *Adam* optimizer with the learning rate of 0.001. This means that the neural network learns to predict the adjacency matrix by adjusting its weights and biases to better match the patterns in the input data. Once the adjacency matrix has been learned, it can be used in the graph neural network to compute the node embeddings (i.e., the feature vectors that represent each pixel in the image) and to propagate information across the graph during the forward pass to model the relationships between the pixels in the image, which can lead to improved performance in image classification tasks. This process is iterated for a fixed number of epochs, allowing the network to learn the relationships between pixels encoded in the adjacency matrix. Finally, the characteristics of the last node are passed through another layer, enabling the prediction of BUY and SELL signals based on Eqs. (7) and (8).

$$Z = \sigma\left(A_{learned}\, X^{(l)} W^l + b\right) \tag{7}$$

$$class = Softmax(Z) \tag{8}$$

In this study, the adjacency matrix, called the proximity matrix, is learned from our 10×10 images. Each pixel (feature) in the image is considered a node in the graph, and the goal is to learn a function that predicts edges between pixels of similar colors. By considering the existence of an edge between two nodes, it implies a connection between the corresponding pixels and their importance in classifying BUY/SELL images. Moreover, the network design follows the concept of updating node features by aggregating neighboring node features and applying a learned transformation. This aggregation is performed by weighting the features of neighboring nodes, where the weights are learned during the training phase.

The corresponding prediction method is designed with a limited number of hyperparameters, offering PGN a distinct advantage over existing deep learning methods. However, similar to other deep learning approaches, our PGN method encounters computational complexities to some extent. The overall computational complexity is the sum of the complexities of its individual components. Firstly, the creation of the proximity matrix using the Meshgrid function involves generating coordinate grids for each pixel. Assuming there are N pixels, the complexity of creating the Meshgrid is $O(N^2)$. Secondly, in the adjacency matrix learning part, predicting edges between pairs of pixels involves using a fully connected neural network. Assuming there are N pixels in each row or column and E edges to be predicted, the complexity of predicting edges for all pairs is $O(N^2 \times E)$. Thirdly, in the GNN's information propagation, the forward pass involves matrix multiplication and non-linear activations, which can be done in $O(N^2 \times F)$ dimension, where F is the number of features.

4.4. Portfolio rebalancing

While the MV model is widely used, it has limitations, particularly concerning its risk criteria. To address these shortcomings, a relatively new risk measure, known as conditional drawdown at risk (CDaR), has emerged. This concept was initially proposed by Chekhlov et al. [33] and subsequently refined by Krokhmal et al. [34], which is particularly suitable for conservative investors who prioritize reducing significant capital losses during critical periods. CDaR is a metric that calculates the potential loss in the value of a stock's portfolio from its maximum value over a specific period. Considering the tail risks in the distribution of stock returns, this measure offers a more comprehensive view of portfolio risk compared to other measures like Semi-variance and CVaR. Additionally, CDaR addresses the limitations of these measures by considering the time frame of stock value reduction, thus providing a more accurate picture of potential portfolio losses. The corresponding model can be mathematically described by the following steps and formulas.

The proposed model is a two-objective linear programming model that aims to maximize the return of the stocks' portfolio as the first objective function while minimizing the risk level of the stock portfolio using the conditional drawdown at risk measure as the second objective function. The main objective of the model is to minimize the risk of the stocks' portfolio while ensuring that the amount of capital loss in each period remains within certain limits, the expected return of the investor is met, and a positive investment weight in each asset is guaranteed. The amount of capital loss (drawdown of the portfolio) in each period is calculated by Eq. (9).

$$D(w, r, t) = \mathrm{Max}_{1 \le k \le S} \left\{ \sum_{i=1}^{M} \left(1 + \sum_{t=1}^{k} r_{ti}\right) w_i \right\} - \left\{ \sum_{i=1}^{M} \left(1 + \sum_{t=1}^{S} r_{ti}\right) w_i \right\} \tag{9}$$

where w_i is the vector of stocks' weights in the corresponding portfolio; M is the total number of assets in that portfolio; r_{ti} is the return of asset i in period t. The first summation of Eq. (9) represents the maximum value of the stocks' portfolio from the beginning of the investment period until the current period S. The second summation represents the current value

of the stock portfolio. Additionally, CDaR is calculated as the expected value of the worst $(1-\beta)\%$ of losses in different periods. In this case, CDaR can be calculated using Eq. (10).

$$\mathrm{CDaR}_{\alpha}(w,\beta) = \frac{1}{(1-\beta)}\int_{D(w,r,t)\geq\alpha(w)} D(w,r,t)p(r(t))dr(t) \tag{10}$$

where α is the portfolio's drawdown with confidence level β, and $p(r(t))$ is the probability distribution of assets' cumulative daily returns. It is challenging to optimize Eq. (10) due to the integration over VaR values. However, Rockafellar and Uryasev [33] proposed a key insight that allows us to optimize a convex function instead of nasty Eq. (10) in the form of Eq. (11):

$$\mathrm{CDaR}_{\beta}(w,\alpha) = \alpha + \frac{1}{(1-\beta)}\int [D(w,r,t)-\alpha]^{+}p(r(t))dr(t) \tag{11}$$

where $[x]^{+} = \max(x,0)$. The objective of minimizing $CDaR_{\beta}(w,\alpha)$ by the variables w and α is to minimize the risk measure of capital loss. The integral in the expression can be transformed into a summation, resulting in the conversion of the CDaR optimization problem into a linear problem.

Therefore, our proposed Mean-CDaR portfolio optimization model is shown in Eqs. (12) to (17):

$$\underset{w_i,\alpha}{\text{minimize}} z = \mathrm{CDaR}_{\beta} = \alpha + \frac{1}{(1-\beta)S}\sum_{s=1}^{S} u_s \tag{12}$$

subject to :

$$u_s \geq \left\{\sum_{i=1}^{M}\left(1+\sum_{t=1}^{k} r_{ti}\right)w_i\right\} - \left\{\sum_{i=1}^{M}\left(1+\sum_{t=1}^{S} r_{ti}\right)w_i\right\} - \alpha, \tag{13}$$

$$\sum_{i=1}^{M} w_i E(R_i) \geq E(r_p), \tag{14}$$

$$\sum_{i=1}^{M} w_i = 1, \tag{15}$$

$$u_s \geq 0, \tag{16}$$

$$w_i \geq 0. \tag{17}$$

Eq. (12) defines the objective function of the optimization problem, which aims to minimize the capital loss under conditional drawdown at risk with respect to w_i and α. This risk measure is calculated as the average of $(1-\beta)S$, where α is the threshold exceeded by $(1-\beta)S$ of capital losses. Eq. (13) is the linearized version of the non-linear Eq. (9) and ensures the maximum capital loss in each period is calculated. Eq. (14) ensures that the value of the stocks' portfolio consisting M assets is greater than the expected minimum return of the investors. Eq. (15) ensures that the total percentage of all assets' weights in the portfolio equals one. Eq. (16) guarantees that the amount of capital loss in each period is positive, and Eq. (17) implies that short selling is not allowed.

The formation of the stocks' portfolio trading system begins with the implementation of the PGN predictive model, which forecasts stocks likely to experience upward movement in each $t+1$ period. Stocks identified with BUY signals are then forwarded to the proposed Mean-CDaR optimization model, responsible for determining the optimal weights for each stock on the $t+1$ day. This sequential process is iterated over a time window of 1512 days, with the portfolio's value is continuously adjusted, taking into account the optimal weights, transaction costs, and our initial investment. By adhering to this approach, the investor can continually adjust the portfolio value in response to market conditions throughout the critical testing period (2017–2023).

5. Performance evaluation

In this section, we focus on the results of all implementations in terms of the main sub-sections presented in Section 4. First, we present the results of feature selection in conjunction with our labeling algorithm in subsection 5.1. The applied evaluation metrics are described in subsection 5.2. Finally, the comparative results of our proposed and benchmark models are presented in subsection 5.3. for both computational and financial performance.

5.1. Feature selection in conjunction with labeling results

AUC-ROC (Area Under the Curve - Receiver Operating Characteristics) curve is particularly useful when examining or visualizing the performance of a binary classification problem. The AUC-ROC curve captures the trade-off between the true positive rate (TPR) and the false positive rate (FPR). The precision-recall curve illustrates the balance between precision and recall at various thresholds. A larger area under the curve indicates both high precision and high recall. When both scores are high, it indicates that the classifier produces more accurate results.

For Salesforce stock, the effectiveness of the labeling and feature selection algorithms, which are created our overall feature engineering process, is evident from the curves which are presented in Fig. 7. In this regard, it is worth mentioning that classes 0 and 1 are SELL and BUY labels respectively. The ability to accurately distinguish between BUY and SELL classes is significantly improved, as indicated by the increased area under the ROC curve from 0.54 to 0.86 for the SELL class, and precision vs. recall curve from 0.547 to 0.918 for the BUY class, respectively.

This highlights the importance of employing appropriate feature selection and labeling techniques in stock trend forecasting and generally in supervised learning. By addressing this often-neglected issue, our models give us higher prediction metrics, significant profit, and reduced losses in real portfolio trading systems. The results of our feature engineering process for the remaining 17 stocks are also provided in the supplementary material file.

5.2. Evaluation metrics

The performance of the PGN model for stock trend prediction is evaluated in two ways: computational and financial performance evaluation.

5.2.1. Evaluation metrics for computational performance

To evaluate computational performance, the PGN model is compared to several widely used stock trend prediction benchmarks. All implementations of the methods use Pytorch,[2] TensorFlow,[3] and Sklearn[4] libraries in Python. To speed up the models' training process, we utilize the Google Colab[5] environment and take advantage of GPU and CUDA.

To evaluate the performance of the proposed PGN model and other benchmarks for stock trend prediction, several evaluation metrics in binary classification problems are needed. Some well-known metrics are Accuracy, Recall, Precision, F-measure (F1), and Matthews Correlation Coefficient, which are formulated as follows.

$$\text{Accuracy} = \frac{TB+TS}{TB+FS+FB+TS}, \tag{18}$$

$$\text{Precision}_{BUY} = \frac{TB}{TB+FB}, \tag{19}$$

[2] https://pytorch.org/
[3] https://www.tensorflow.org/
[4] https://scikit-learn.org/
[5] https://colab.research.google.com/

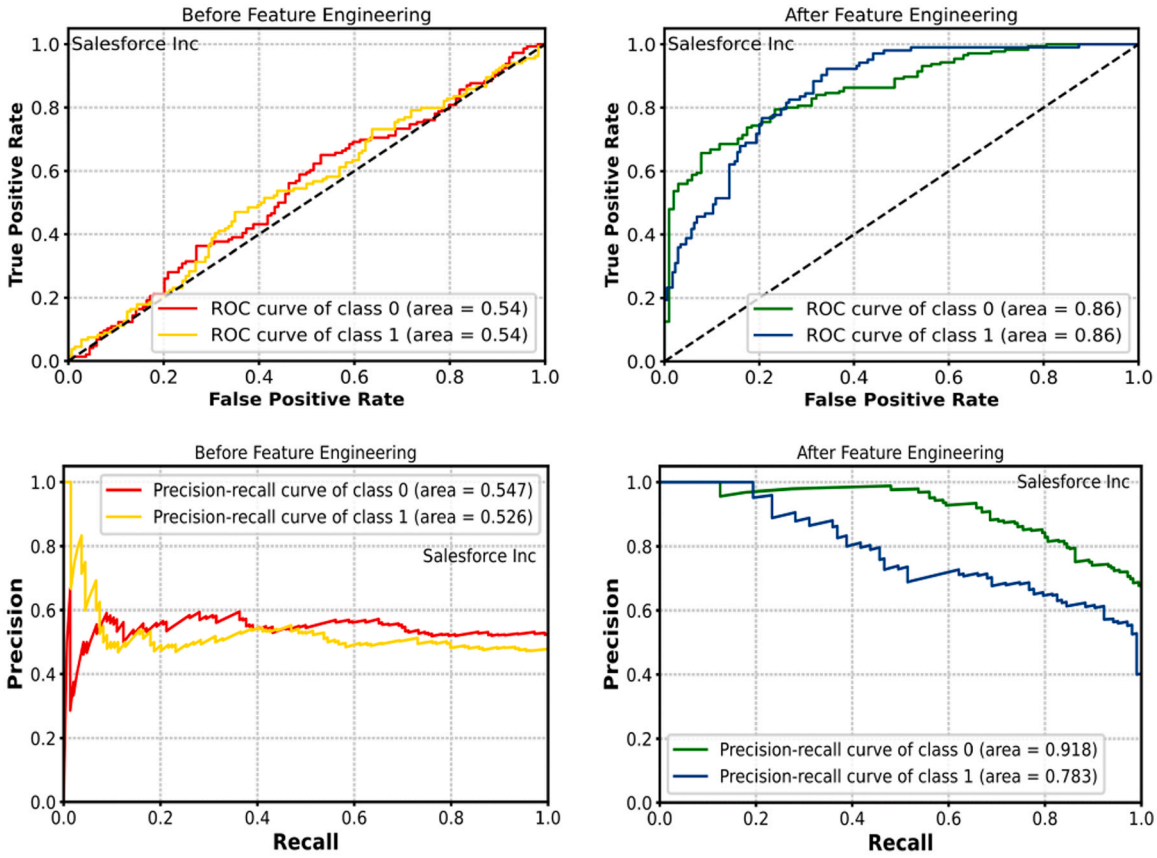

Fig. 7. AUC-ROC and precision vs. recall curves for our feature engineering process.

$$\text{Precision}_{\text{SELL}} = \frac{\text{TS}}{\text{TS} + \text{FS}}, \tag{20}$$

$$\text{Recall}_{\text{BUY}} = \frac{\text{TB}}{\text{TB} + \text{FS}}, \tag{21}$$

$$\text{Recall}_{\text{SELL}} = \frac{\text{TS}}{\text{TS} + \text{FB}}, \tag{22}$$

$$\text{F1}_{\text{Score}} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}, \tag{23}$$

$$\text{MCC} = \frac{(\text{TB} \times \text{TS}) - (\text{FB} \times \text{FS})}{\sqrt{(\text{TB} + \text{FB})(\text{TB} + \text{FS})(\text{TS} + \text{FB})(\text{TS} + \text{FS})}}. \tag{24}$$

where TB is the number of true BUY signals, TS is the number of true SELL signals, FB is the number of false BUY signals, and FS is the number of false SELL signals. Note that each of these evaluation metric takes values between 0 and 1, in which higher values indicate better performance.

5.2.2. Evaluation metrics for financial performance

In the financial evaluation, the focus is on simulated portfolio trading systems, with an emphasis on portfolios' returns and risks. To evaluate the performance, we use four metrics: Total portfolio value (TPValue), average annual return (AAR), annual maximum drawdown (AMDD), average annual Sharpe ratio (AASHR), and average annual Sortino ratio (AASOR). The annual risk-free rate is set to 2%, according to the US treasuring bill rate in recent ten years.

$$\text{AAR} = \left(\frac{\text{TPValue}}{\text{Start } \textit{Money}} \right)^{\frac{1}{\text{Number of } \textit{Years}}} - 1 \Bigg) \times 100, \tag{25}$$

$$\text{AMDD} = \left(\frac{\text{The Lowest Value of } \textit{Portfolio} - \text{The highest Value of } \textit{Portfolio}}{\text{The highest Value of } \textit{Portfolio}} \right) \times 100, \tag{26}$$

$$\text{AASHR} = \frac{\text{AAR} - \text{Risk Free } \textit{Rate}}{\text{Standard Deviation of Annual } \textit{Return}}, \tag{27}$$

$$\text{AASOR} = \frac{\text{AAR} - \text{Annual Risk Free } \textit{Rate}}{\text{Negetive Standard Deviation of Annual } \textit{Return}}. \tag{28}$$

5.3. Comparative models

We compare the performance of the proposed PGN model with other benchmark models, such as 2D CNN-Bi-LSTM, 2D CNN, Bi-LSTM, MLP, XGB, SVM, and DT, to demonstrate its superior computational and financial results. Each model is executed 20 times to reduce randomness. It is important to highlight that the parameters for benchmark models are established through a two-step process. Initially, a trial-and-error approach is employed to identify the approximate optimal range for the parameters. Subsequently, a grid search is conducted to pinpoint the best-suited parameters. The structure of all these models, along with their respective parameters, is outlined in Appendix B.

5.3.1. Computational performance of different models

Table 2 illustrates the computation performance of different models during the testing period for 3 stocks. The rest of table for the other 15 stocks are also provided in Appendix A. We mark the best results in bold style. The results of our feature engineering process from AUC-ROC and precision vs. recall curves in subsection 5.1 unequivocally demonstrate a substantial knowledge discovery level between technical indicators and BUY/SELL signals, as highlighted in Table 2. Notably, all models

Table 2
Comparison results for computational performance evaluation forthree stocks.

Stock	Model	Accuracy	Buy signals			Sell signals			MCC	AR[a]
			Recall	Precision	F1	Recall	Precision	F1		
GS	SVM	76.59% (2)	83.12% (2)	71.64% (3)	76.95% (2)	70.79% (6)	82.53% (2)	76.21% (3)	0.5403(3)	2.87
	DT	74.01% (7)	74.96% (8)	71.26% (5)	73.06% (8)	73.16% (2)	76.70% (8)	74.89% (6)	0.4804(7)	6.37
	XGB	76.46% (3)	**84.67% (1)**	70.91% (6)	**77.18% (1)**	69.16% (8)	**83.56% (1)**	75.68% (4)	**0.5415(1)**	3.12
	MLP	75.40% (5)	79.89% (4)	71.27% (4)	75.34% (5)	71.41% (5)	80.00% (5)	75.46% (5)	0.5130(5)	4.75
	Bi-LSTM	75.00% (6)	81.15% (3)	70.28% (8)	75.33% (6)	69.54% (7)	80.61% (3)	74.66% (7)	0.5079 (6)	5.75
	2D CNN	73.68% (8)	76.09% (7)	70.35% (7)	73.11% (7)	71.54% (4)	77.12% (7)	74.22% (8)	0.4755(8)	7.00
	2D CNN-Bi-LSTM	76.06% (4)	79.75% (5)	72.23% (2)	75.80% (4)	72.78% (3)	80.19% (4)	76.31% (2)	0.5247(4)	3.5
	Proposed PGN	**76.98% (1)**	78.48% (6)	**74.10% (1**	76.23% (3)	**75.66% (1)**	79.84% (6)	**77.69% (1)**	0.5404(2)	2.62
CRM	SVM	76.65% (6)	78.33% (4)	78.81% (5)	78.57% (5)	74.63% (6)	74.10% (5)	74.36% (6)	0.5293(6)	5.37
	DT	77.71% (2)	**84.26% (1)**	77.08% (8)	**80.51% (1)**	69.83% (8)	**78.65% (1)**	73.98% (7)	0.5490(3)	3.87
	XGB	**77.78% (1)**	81.72% (2)	78.49% (6)	80.07% (2)	73.03% (7)	76.84% (2)	74.89% (3)	**0.5504(1)**	3.00
	MLP	76.52% (7)	77.36% (6)	79.18% (3)	78.26% (7)	75.51% (3)	73.48% (6)	74.48% (4)	0.5276(7)	5.37
	Bi-LSTM	76.71% (5)	76.76% (7)	79.85% (2)	78.27% (6)	76.68% (2)	73.26% (7)	74.93% (2)	0.5327(4)	4.37
	2D CNN	76.72% (4)	78.45% (3)	78.83% (4)	78.64% (4)	74.64% (5)	74.20% (4)	74.42% (5)	0.5306(5)	4.25
	2D CNN-Bi-LSTM	74.96% (8)	74.94% (8)	78.26% (7)	76.56% (8)	74.93% (4)	71.29% (8)	73.06% (8)	0.4971(8)	7.37
	Proposed PGN	77.58% (3)	77.72% (5)	**80.55% (1)**	79.11% (3)	**77.41% (1)**	74.27% (3)	**75.80% (1)**	0.5497(2)	**2.4**
DIS	SVM	74.21% (2)	76.49% (4)	64.05% (2)	**69.72% (1)**	72.76% (2)	82.98% (2)	77.53% (2)	**0.4812(1)**	**2.00**
	DT	72.95% (3)	74.45% (6)	62.79% (3)	68.12% (5)	72.00% (3)	81.62% (7)	76.51% (3)	0.4541(5)	4.37
	XGB	72.16% (6)	76.32% (5)	61.37% (7)	68.03% (6)	69.51% (6)	82.23% (4)	75.34% (7)	0.4470(6)	5.87
	MLP	72.48% (5)	77.00% (3)	61.66% (5)	68.48% (4)	69.62% (5)	82.67% (3)	75.59% (5)	0.4546(4)	4.25
	Bi-LSTM	72.09% (7)	72.57% (8)	62.01% (4)	66.88% (8)	71.78% (4)	80.48% (8)	75.89% (4)	0.4341(7)	6.25
	2D CNN	70.70% (8)	77.34% (2)	59.42% (8)	67.21% (7)	66.49% (8)	82.22% (5)	73.52% (8)	0.4272(8)	6.75
	2D CNN-Bi-LSTM	72.49% (4)	**78.36% (1)**	61.42% (6)	68.86% (3)	68.76% (7)	**83.36% (1)**	75.36% (6)	0.4593(3)	3.87
	Proposed PGN	**74.34% (1)**	73.94% (7)	**64.08% (1)**	69.11% (2)	**74.59% (1)**	81.85% (6)	**78.05% (1)**	0.4762(2)	2.62

[a] Average Rank

exhibited commendable performance across all eight comparative criteria, surpassing relevant trend prediction studies. This intensifies the competition faced by the PGN model in comparison to its counterparts.

Upon scrutinizing the prediction results for all stocks, it becomes evident that our proposed PGN model outperforms the other models according to the average rank of all evaluation metrics. This signifies that by considering a separate learning part for image's components (pixels), the prediction performance can be enhanced. While the suggested PGN model exhibits superiority over the comparative models, it may not consistently outperform in all evaluation metrics for both BUY and SELL classes. It is evident that in predicting future stock trends, no single model consistently achieves the highest values for Accuracy, Recall, Precision, F1, and MCC simultaneously. Forecasting performance can vary considerably based on the characteristics of the model and the specific stock under consideration.

In terms of computational evaluation, only in the case of CAT (Caterpillar) and DIS (Walt Disney) stocks, the PGN model ranks second in the average ranking. If we focus on the average ranking results for the mentioned two stocks, it becomes apparent that PGN ranks second, with a noticeably smaller difference compared to the SVM model. However, the PGN model achieving the first rank in most of the criteria. For instance, for the DIS stock, the PGN reaches the first rank in 4 criteria, such as Accuracy, Precision $_{Buy}$, Recall $_{Sell}$, and $F1_{Sell}$, which are 74.34% (1), 64.08% (1), 74.59% (1), and 78.05% (1), respectively. Likewise, for the CAT stock, the PGN reaches the first rank in five criteria, such as Accuracy, Precision $_{Buy}$, Recall $_{Sell}$, ad $F1_{sell}$, and MCC, which are 76.32% (1), 79.83% (1), 81.29% (1), 77.14% (1), and 0.5301(1), respectively.

The performance stability of the proposed PGN model is evident and surpasses that of other comparative models. For example, although comparative models might demonstrate high recall in predicting BUY/SELL signals, conversely, their precision values tend to decrease. In contrast, the stability of the PGN model is advantageous, as it maintains relatively high recall and precision values with insignificant discrepancies. This characteristic of the PGN is valuable in maximizing profits and enabling better identification of profitable BUY and SELL signals.

It is crucial to highlight that, in financial performance evaluation, high computational performance is not the primary factor for a successful portfolio trading system. Furthermore, the model's profitability should be evaluated based on the accuracy of its predicted signals. A model may demonstrate high prediction metrics in identifying BUY and SELL signals, but it might overlook the potential for significant profits while capturing smaller gains. Therefore, the next step involves real-world engagement in the market for a portfolio-based trading system.

5.3.2. *Financial performance of different models*

One valuable application of a novel forecasting model is the formation of an optimal portfolio, which can effectively mitigate risks associated with risky assets like company stocks. The main aim of this study is to investigate that whether the outcome obtained from the proposed PGN model in conjunction with the formation of an optimal portfolio outperforms other benchmark models or not. As previously mentioned, our research encompasses two main stages. In the first stage, the PGN model is utilized to predict the direction of stock prices in the next period. The predicted results are then classified into two categories: price increases (BUY signal) and price decreases (SELL signal). Stocks that receive a BUY signal for the next period proceed to the next stage. In the second stage, the M-CDaR model is employed to determine the optimal weight for each stock included in the portfolio. Subsequently, based on transaction costs (0.5% for the BUY and 1% for SELL), optimal weights, and an initial capital of $100,000 which is allocated to our investment, the portfolio optimization process is carried out over 1512 trading days (equivalent to 6 years) to rebalance our investment portfolio dynamically. Ultimately, we can plot the value of optimal portfolios during the testing period (2017–2023) as shown in Fig. 8.

Fig. 8. demonstrates that our proposed PGN+M-CDaR model has outperformed other models in managing optimal portfolio formation. The Bi-LSTM+M-CDaR model ranks the second, followed by the SVM+M-CDaR model in the third place. However, to provide a more detailed assessment of whether our proposed model consistently delivered the highest returns each year, we can refer to the heat plot in Fig. 9. This plot allows us to evaluate the profitability of both our proposed PGN+M-CDaR system and the corresponding benchmark systems for each year.

The PGN+M-CDaR model of this study beats all other portfolio formations in terms of return in all time frames. However, the Bi-LSTM+M-CDaR in 2021–2022 outperforms other models. Relying solely on profit for decision-making is not advisable; the level of risk should also be considered to arrive at a comprehensive conclusion. During a specific

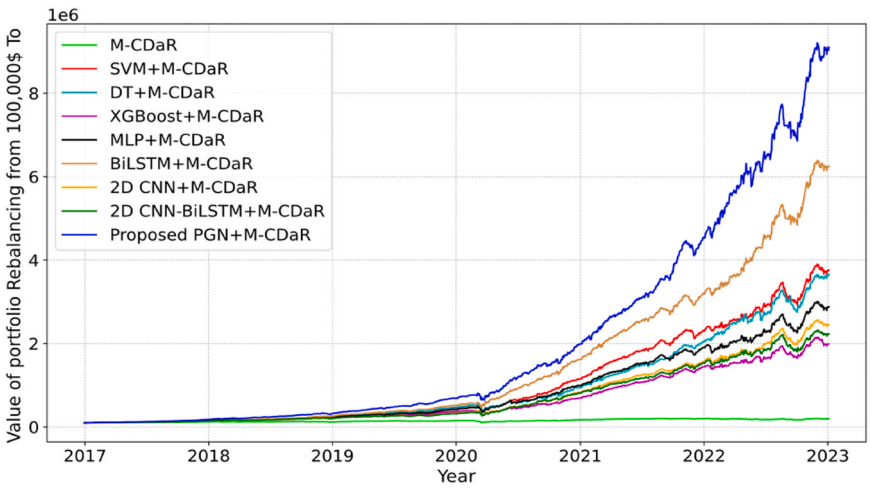

Fig. 8. Value of portfolio rebalancing including transaction cost (1.5%).

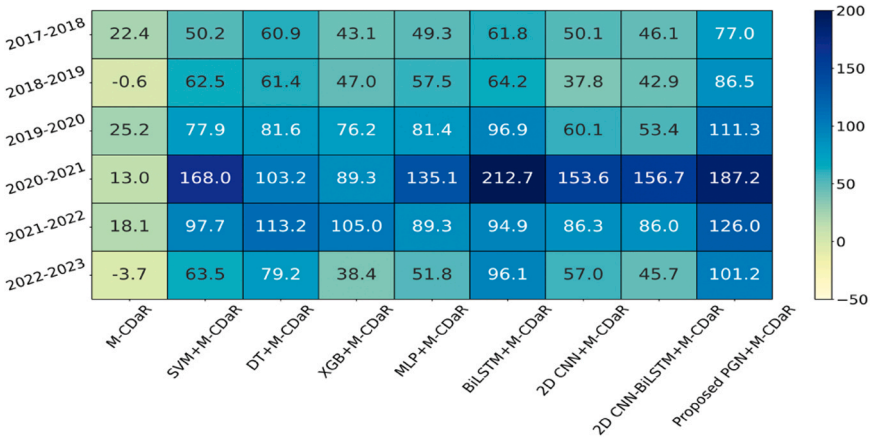

Fig. 9. Average returns of our proposed PGN+M-CDaR model and benchmarks with transaction costs of 1.5%.

time period, the portfolio trading system may have encountered low levels of risk, leading to increased profitability. In essence, the criteria of risk and profitability should be examined simultaneously. For this reason, Figs. 10 and 11 summarize the performance of each model with respect to the Sharpe ratio and the Sortino ratio each year.

As shown in Fig. 10, we use the Sharpe ratio performance of each model every year. We can observe that, of the six years, four of them show that the Sharpe ratio of the PGN+M-CDaR model has a better result than other models during the corresponding periods. Fig. 11 depicts the Sortino ratio per year with transaction costs in which our PGN+M-CDaR model, with transaction costs of 1.5%, has better performance. Specifically, among the six surveyed years, four of them have the highest Sortino ratios. Lastly, Table 3 summarizes the performance of each model for different portfolio trading systems, in which Panel A, B, and C illustrate daily returns' descriptive statistics, annualized risk-return metrics, and statistical tests, respectively.

According to Panel A in Table 3, it is clear that the proposed PGN+M-CDaR achieves the highest daily mean return of 0.0031, the Bi-LSTM+M-CDaR follows, with a return of 0.0028, and then the 2D CNN+M-CDaR, with 0.0025. Also, standard deviation and downside deviation are used as risk measures. We can see that the proposed PGN+M-CDaR has the lowest risk, with a standard deviation of 0.012 and a downside deviation of 0.009, and the Bi-LSTM+M-CDaR ranks the second with corresponding risk measures.

From Panel B in Table 3, the PGN+M-CDaR model reaches the highest annualized return of 1.1228. However, the average annual Sharpe ratio of DT+M-CDaR has the highest level as 3.7533, and the PGN+M-CDaR comes second with 2.7937. As for the Sortino ratio, the PGN+M-CDaR has better performance, followed by Bi-LSTM+M-CDaR. As for the annual maximum drawdown, again the PGN+M-CDaR ranks the first with the lowest maximum drawdown of 0.1981.

In Panel C of Table 3, to put an approval seal on the superiority of the proposed portfolio trading system, one-tailed T-test (for mean evaluation) and ANOVA-test (for variance analysis) are carried out statistically, and the M-CDaR optimization model is considered as the baseline model in this analysis, with no incorporation of any prediction method. By observing the probability of p-value, it can be stated that all portfolio-based trading systems are remarkably profitable compared to the base system of this research (M-CDaR), with a very high confidence level of 99%. In other words, portfolio optimization model in conjunction with prediction models has a meaningful difference, and it is strongly recommended to combine them. Finally, by looking at both value of T-test and ANOVA-test to analyze statistical significance of each system, the proposed PGN+M-CDaR model is recognized as the most effective stock portfolio trading system in this study, achieving higher profits and minimizing drawdowns.

6. Discussion and conclusion

6.1. Discussion for key findings

In this paper, our primary objective was to develop a portfolio-based algorithmic trading system based on the novel Pixel Graph Network (PGN) model in conjunction with Mean–Conditional Drawdown at Risk

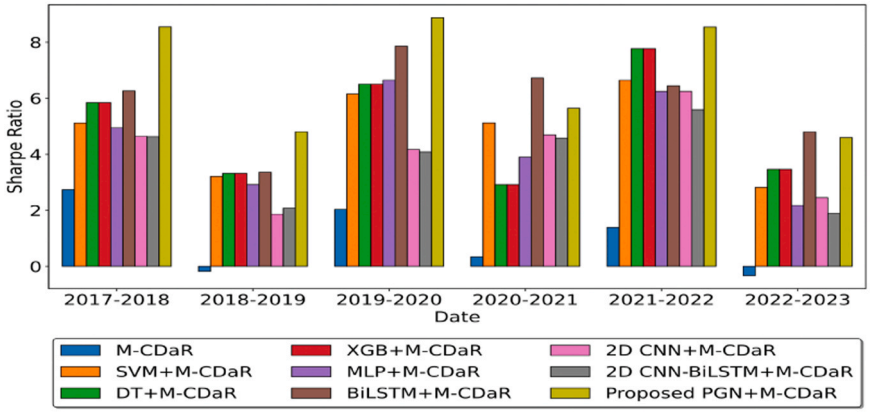

Fig. 10. Sharpe ratio of each portfolio trading system including transaction costs of 1.5%.

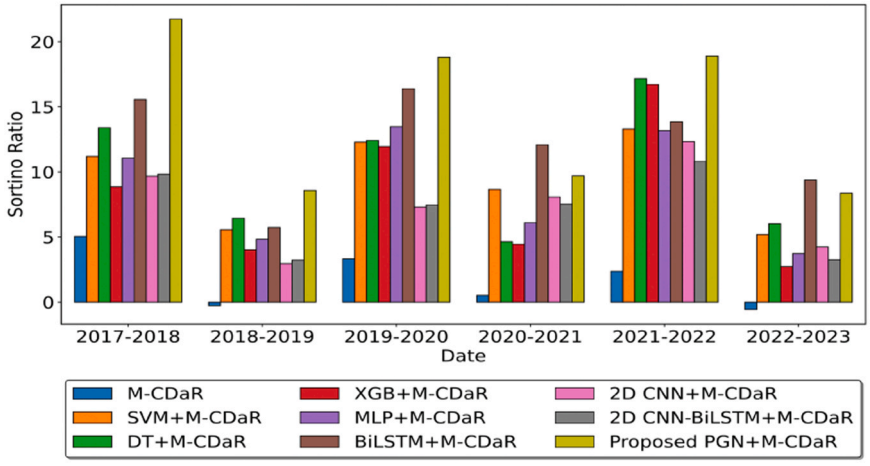

Fig. 11. Sortino ratio of each portfolio trading system including transaction costs of 1.5%.

Table 3
Performances of portfolio trading systems with transaction cost of 1.5% during 2017–2023.

Models	M-CDaR	SVM+ M-CDaR	DT+ M-CDaR	XGB+ M-CDaR	MLP+ M-CDaR	Bi-LSTM+ M-CDaR	2D CNN+ M-CDaR	2D CNN-Bi-LSTM+ M-CDaR	Proposed PGN+ M-CDaR
Panel A: Daily return descriptive statistics									
Mean	0.0005	0.0024	0.0024	0.0020	0.0023	0.0028	0.0025	0.0021	**0.0031**
Standard dev.	0.0122	0.0124	0.0128	0.0122	0.0128	0.0123	0.0126	0.0130	**0.0120**
Downside dev.	0.0101	0.0097	0.0106	0.0099	0.0108	0.0099	0.0099	0.0106	**0.0090**
Panel B: Annualized risk-return metrics (initial capital: $100,000)									
TPValue	195707.71	3762261.19	3659539.99	1990086.12	2883458.81	6251851.21	2457962.46	2228578.18	**9105848.68**
AAR	0.1193	0.8320	0.8236	0.6475	0.7526	0.9938	0.7065	0.6789	**1.1228**
Standard dev.	0.1205	0.4305	**0.2141**	0.2761	0.3263	0.5546	0.4205	0.4454	0.3948
Downside dev.	0.1256	0.1118	0.1193	0.1155	0.1230	0.1089	0.1163	0.1227	**0.1023**
AMDD	0.3277	0.2735	0.2834	0.3000	0.2948	0.2079	0.2349	0.2543	**0.1981**
AASHR	0.8241	1.8862	**3.7533**	2.2727	2.2451	1.7559	1.6326	1.4793	2.7937
AASOR	0.7906	7.2629	6.7360	5.4329	5.9561	8.9421	5.9028	5.3700	**10.7808**
Panel C: Statistical tests for daily returns									
T-test	N.A.	4.3720	4.2794	3.4617	3.9292	5.2029	3.7229	3.5357	**5.6982**
ANOVA-test	N.A.	19.114	18.3131	11.9833	15.4390	27.0699	13.8598	12.1514	**32.4690**
p-value	N.A.	6.36e-06	9.66e-06	2.72e-04	4.35e-05	1.04e-7	1.00e-04	2.06e-04	**6.64e-09**

(M-CDaR) portfolio optimization model. To achieve this, firstly, we select some stocks from various industries listed on the New York Stock Exchange as our financial time series data. Secondly, in order to train the network effectively, we incorporate transaction costs and balance rate into our Continuous Trend Labeling process. This ensures that the network is learned to identify only those BUY and SELL points that would result in profitable trades. Thirdly, we employ a two-step feature selection approach, enabling us to select technical indicators that are most suitable for the labeling process. Fourthly, we transform the selected indicators into clustered two-dimensional images with graph structure. The goal is to predict the entry and exit points of these time series by generating BUY and SELL signals for profitable trades. Lastly, stocks with BUY prediction for the next period are fed into the M-CDaR model to determine the optimal weights of stocks to rebalance the investment portfolio value over the testing period. Accordingly, we have several findings discussed below.

First of all, the impact of the labeling and feature selection algorithms utilized in this study as a feature engineering process is

outstanding. It is noteworthy that AUC-ROC and precision vs. recall curves enhanced by more than 30% for each BUY/SELL class after applying the corresponding feature engineering process.

Having good performance of our PGN model in computational performance compared to the benchmarks, we evaluate its financial performance by developing a stock portfolio trading system with the aim of minimizing a risk measure called CDaR. It can be concluded that the graph-based representation on the image's components with exploring spatial relationships between the selected features, boosting a neural network's capability of capturing high-level information and complex patterns from time series' sparse images. This leads to an improvement of PGN's binary classification metrics and a high expected rate of return in conjunction with the least possible risk for portfolio rebalancing. Our PGN's higher Sortino ratio compared to the Sharpe ratio indicates that the model is efficient in generating returns while minimizing the risk associated with negative returns. It implies that the model is particularly successful in managing and reducing downside risk, which can be desirable for investors who prioritize the avoidance of losses. In other words, the results have put an approval seal on the assertion that our proposed PGN, considering graph network on the component of the images (i.e., pixels), increases the model's performance in both single stock prediction and portfolio optimization to provide investors with the most profitable investment decisions.

6.2. Theoretical and practical implications

This study contributes to both theoretical and practical perspectives on vision-based deep learning and financial investing. The discussion revolves around graph-structured data and their learning method, namely the graph neural network, which is introduced as a progressive and promising approach. This type of graph-based modeling is currently gaining traction in fields such as biology, chemistry, and neuroscience, and we have innovatively incorporated this emerging concept into the discourse on portfolio-based algorithmic trading. In the current investment landscape, investors tend to be more conservative, aiming to avoid potential capital losses in their portfolios. Consequently, the corresponding asset allocation problem is formulated with the drawdown risk measure, in conjunction with the novel PGN predictive model, for the first time. The goal is to minimize the likelihood of significant drawdowns. A key advantage of the predictive PGN method is its adaptive learning of the graph structure based on input data, eliminating the need for a predefined graph structure or fixed-size input, as seen in CNNs. This characteristic enhances the flexibility and compatibility of the model across various applications, allowing it to handle diverse types of data and assist financial practitioners in making well-informed investment decisions. To sum up, our PGN's dynamic learning capability enables the model to capture intricate relationships between various nodes (pixels) in the graph and holds potential for various tasks, including image segmentation, social network analysis, and recommendation systems, where understanding complex relationships is crucial.

6.3. Limitations and future work

While this research provides valuable insights, it is essential to point out its limitations, which open up opportunities for future research. Firstly, one notable limitation is using only technical indicators for predicting future movements. It would be beneficial to explore the inclusion of additional external macroeconomic factors, such as governmental policies, interest rates, public events, and other market influencers, as input indicators for the models. Secondly, another important limitation of this study is the exclusive use of asset data from the US, which restricts the generalizability of the proposed model to other countries' stock markets. The political environment, economic conditions, and market dynamics can vary significantly across different countries, potentially impacting the effectiveness of the proposed PGN model in those contexts. To address this limitation, future research should consider incorporating asset data from multiple developed and underdeveloped countries in experiments. Thirdly, like 3D CNNs, our PGN model can be developed with 3D images to capture more information across different 2D channels. Fourthly, in terms of all underlying assumptions and choices for our proposed portfolio trading system, such as transaction cost, number of training/testing features, etc., sensitivity analysis can be carried to boost the generalizability of the findings in much more results.

Funding

Not Applicable.

CRediT authorship contribution statement

Milad Kamali Alamdari: Conceptualization, Data curation, Methodology, Software, Validation, Writing – original draft. **Akbar Esfahanipour:** Conceptualization, Methodology, Supervision, Validation, Writing – review & editing. **Hossein Dastkhan:** Conceptualization, Methodology, Validation, Writing – review & editing. .

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data Availability

Data will be made available on request.

Appendix A. Computational performance evaluation for other 15 stocks

Table A.1 The rest of the Table 2 for computational performance evaluation for other 15 stocks

Stock	Model	Accuracy	Buy signals			Sell signals			MCC	AR
			Recall	Precision	F1	Recall	Precision	F1		
MRK	SVM	73.28% (5)	67.43% (6)	75.75% (2)	71.35% (5)	78.98% (2)	71.34% (5)	74.97% (3)	0.4675(5)	4.12
	DT	71.56% (8)	68.90% (5)	72.19% (8)	70.51% (6)	74.15% (7)	71.00% (6)	72.54% (8)	0.4312(8)	7.00
	XGB	73.40% (4)	69.44% (4)	74.86% (4)	72.04% (4)	77.28% (4)	72.20% (4)	74.65% (4)	0.4689(3)	3.87
	MLP	72.28% (7)	66.62% (7)	74.51% (5)	70.35% (7)	77.81% (3)	70.53% (7)	73.99% (6)	0.4474(7)	6.12
	Bi-LSTM	74.14% (2)	**76.14% (1)**	72.73% (7)	**74.39% (1)**	72.19% (8)	**75.65% (1)**	73.88% (7)	0.4835(2)	3.62
	2D CNN	73.41% (3)	70.11% (3)	74.50% (6)	72.24% (3)	76.63% (6)	72.47% (3)	74.49% (5)	0.4685(4)	4.12
	2D CNN-Bi-LSTM	72.29% (6)	62.33% (8)	**77.11% (1)**	68.94% (8)	**81.98% (1)**	69.09% (8)	74.99% (2)	0.4525(6)	5.00
	Proposed PGN	**75.00% (1)**	73.32% (2)	75.34% (3)	74.32% (2)	76.64% (5)	74.68% (2)	**75.64% (1)**	**0.4999(1)**	**2.12**

(*continued on next page*)

(*continued*)

Stock	Model	Accuracy	Buy signals			Sell signals			MCC	AR
			Recall	Precision	F1	Recall	Precision	F1		
MCD	SVM	69.67% (7)	69.19% (3)	71.58% (8)	70.37% (4)	70.80% (7)	68.38% (4)	69.57% (7)	0.3998(6)	5.75
	DT	69.31% (8)	67.01% (5)	71.60% (6)	69.23% (8)	71.76% (5)	67.18% (8)	69.37% (8)	0.3878(8)	7.00
	XGB	70.90% (3)	72.14% (2)	71.59% (7)	71.87% (2)	69.58% (8)	70.15% (2)	69.86% (6)	0.4173(3)	4.12
	MLP	70.17% (5)	69.06% (4)	71.93% (5)	70.46% (3)	71.35% (6)	68.46% (3)	69.87% (5)	0.4040(5)	4.50
	Bi-LSTM	69.78% (6)	66.75% (6)	72.42% (4)	69.47% (7)	72.99% (4)	67.38% (7)	70.07% (4)	0.3977(7)	5.62
	2D CNN	71.03% (2)	65.73% (8)	**74.96% (1)**	70.04% (5)	**76.67% (1)**	67.79% (5)	71.96% (2)	0.4258(2)	3.25
	2D CNN-Bi-LSTM	70.57% (4)	66.11% (7)	73.99% (3)	69.83% (6)	75.31% (2)	67.65% (6)	71.27% (3)	0.4153(4)	4.37
	Proposed PGN	**72.95% (1)**	**72.79% (1)**	74.21% (2)	**73.49% (1)**	73.12% (3)	**71.66% (1)**	**72.38% (1)**	**0.4589(1)**	**1.37**
CSCO	SVM	75.99% (2)	77.87% (2)	74.36% (2)	76.07% (2)	74.19% (4)	77.72% (2)	75.91% (2)	0.5207(2)	2.25
	DT	73.41% (6)	71.93% (7)	73.31% (5)	72.62% (6)	74.84% (3)	73.50% (6)	74.16% (6)	0.4680(6)	5.62
	XGB	74.54% (5)	73.68% (5)	72.94% (7)	74.62% (5)	72.76% (7)	76.22% (3)	74.45% (5)	0.4915(5)	5.25
	MLP	75.26% (3)	75.03% (4)	**74.63% (1)**	74.83% (3)	75.49% (2)	75.88% (5)	75.68% (3)	0.5052(3)	3.00
	Bi-LSTM	74.60% (4)	76.25% (3)	73.09% (6)	74.64% (4)	73.02% (6)	76.18% (4)	74.57% (4)	0.4927(4)	4.37
	2D CNN	72.88% (7)	73.14% (6)	71.98% (8)	72.56% (7)	72.63% (8)	73.78% (7)	73.20% (8)	0.4577(7)	7.25
	2D CNN-Bi-LSTM	71.49% (8)	65.32% (8)	73.56% (4)	69.19% (8)	**77.43% (1)**	69.91% (8)	73.48% (7)	0.4310(8)	6.50
	Proposed PGN	**76.39% (1)**	**79.22% (1)**	74.30% (3)	**76.68% (1)**	73.67% (5)	**78.67% (1)**	**76.09% (1)**	**0.5293(1)**	**1.75**
TRV	SVM	74.67% (3)	72.86% (4)	67.72% (2)	70.19% (4)	75.92% (2)	80.14% (4)	77.98% (2)	0.4832(3)	3.00
	DT	71.63% (7)	70.44% (5)	63.93% (8)	67.03% (5)	72.45% (8)	77.95% (5)	75.10% (8)	0.4238(6)	6.50
	XGB	74.07% (4)	**75.61% (1)**	66.01% (4)	70.48% (3)	73.01% (7)	**81.20% (1)**	76.89% (4)	0.4791(4)	3.50
	MLP	74.93% (2)	74.96% (2)	67.44% (3)	71.00% (2)	74.92% (4)	81.19% (2)	77.93% (3)	0.4925(2)	2.50
	Bi-LSTM	71.56% (8)	68.66% (7)	64.30% (7)	66.41% (7)	73.58% (6)	77.20% (7)	75.34% (7)	0.4186(8)	7.12
	2D CNN	72.09% (5)	66.72% (8)	65.66% (5)	66.19% (8)	75.81% (3)	76.67% (8)	76.24% (5)	0.4243(5)	5.87
	2D CNN-Bi-LSTM	71.76% (6)	69.14% (6)	64.46% (6)	66.72% (6)	73.57% (5)	77.48% (6)	75.47% (6)	0.4232(7)	6.00
	Proposed PGN	**75.66% (1)**	73.34% (3)	**69.10% (1)**	**71.16% (1)**	**77.27% (1)**	80.70% (3)	**78.95% (1)**	**0.5021(1)**	**1.50**
CAT	SVM	76.26% (2)	73.99% (2)	78.16% (3)	**76.02% (1)**	78.60% (3)	**74.49% (1)**	76.49% (2)	0.5261(2)	**2.00**
	DT	72.49% (7)	70.09% (7)	74.34% (7)	72.16% (7)	74.97% (7)	70.78% (7)	72.81% (7)	0.4509(7)	7.00
	XGB	75.20% (4)	**74.38% (1)**	76.27% (4)	75.31% (3)	76.04% (5)	74.15% (2)	75.08% (4)	0.5042(4)	3.37
	MLP	74.87% (5)	73.99% (3)	75.97% (5)	74.97% (5)	75.77% (6)	73.79% (3)	74.77% (5)	0.4976(5)	4.62
	Bi-LSTM	72.02% (8)	69.96% (8)	73.70% (8)	71.78% (8)	74.16% (8)	70.46% (8)	72.26% (8)	0.4414(8)	8.00
	2D CNN	74.21% (6)	72.30% (4)	75.85% (6)	74.03% (6)	76.18% (4)	72.66% (6)	74.38% (6)	0.4849(6)	5.50
	2D CNN-Bi-LSTM	75.66% (3)	72.17% (5)	78.28% (2)	75.10% (4)	79.27% (2)	73.35% (5)	76.20% (3)	0.5154(3)	3.37
	Proposed PGN	**76.32% (1)**	71.52% (6)	**79.83% (1)**	75.45% (2)	**81.29% (1)**	73.39% (4)	**77.14% (1)**	**0.5301(1)**	2.12
JNJ	SVM	73.02% (3)	74.29% (3)	69.52% (3)	71.82% (4)	71.92% (4)	76.44% (3)	74.11% (3)	0.4608(3)	3.25
	DT	71.63% (6)	72.29% (5)	68.29% (6)	70.23% (6)	71.06% (6)	74.84% (6)	72.90% (6)	0.4323(5)	5.75
	XGB	72.62% (4)	73.71% (4)	69.17% (5)	71.37% (5)	71.67% (5)	75.98% (5)	73.76% (4)	0.4527(4)	4.50
	MLP	73.61% (2)	71.14% (7)	**70.82% (1)**	71.96% (2)	74.01% (2)	76.17% (4)	**75.08% (1)**	0.4707(2)	2.62
	Bi-LSTM	72.29% (5)	**76.57% (1)**	67.76% (7)	71.90% (3)	68.60% (8)	77.25% (2)	72.67% (7)	0.4509(6)	4.87
	2D CNN	71.36% (7)	68.14% (8)	69.43% (4)	68.78% (8)	**74.14% (1)**	72.97% (8)	73.55% (5)	0.4234(7)	5.37
	2D CNN-Bi-LSTM	70.37% (8)	71.71% (6)	66.76% (8)	69.15% (7)	69.21% (7)	73.95% (7)	71.50% (6)	0.4081(8)	6.62
	Proposed PGN	**74.14% (1)**	76.14% (2)	70.41% (2)	**73.16% (1)**	72.41% (3)	**77.88% (1)**	75.05% (2)	**0.4842(1)**	**1.62**
IBM	SVM	72.88% (4)	68.22% (4)	69.48% (4)	68.84% (4)	76.53% (4)	75.47% (4)	76.00% (5)	0.4485(4)	4.12
	DT	72.75% (6)	67.62% (5)	69.50% (3)	68.55% (5)	76.77% (3)	75.17% (5)	75.96% (6)	0.4453(5)	4.75
	XGB	73.88% (2)	74.40% (2)	68.71% (7)	71.44% (2)	73.47% (8)	78.56% (2)	75.93% (7)	0.4757(2)	4.00
	MLP	73.68% (3)	72.89% (3)	68.95% (6)	70.86% (3)	74.29% (6)	77.78% (3)	76.03% (3)	0.4695(3)	3.75
	Bi-LSTM	70.97% (8)	64.46% (8)	67.83% (8)	66.10% (8)	76.06% (5)	73.21% (8)	74.61% (8)	0.4078(8)	7.62
	2D CNN	72.82% (5)	64.91% (7)	**70.77% (1)**	67.71% (6)	**79.01% (1)**	74.20% (6)	**76.53% (1)**	0.4444(6)	4.12
	2D CNN-Bi-LSTM	72.35% (7)	65.06% (6)	69.90% (2)	67.39% (7)	78.07% (2)	74.05% (7)	76.01% (4)	0.4354(7)	5.25
	Proposed PGN	**74.27% (1)**	**74.70% (1)**	69.18% (5)	**71.83% (1)**	73.94% (7)	**78.87% (1)**	76.32% (2)	**0.4834(1)**	**2.37**
UNH	SVM	74.07% (3)	**77.40% (1)**	69.81% (4)	73.41% (3)	71.22% (4)	78.56% (2)	74.71% (4)	0.4849(3)	3.00
	DT	70.44% (6)	70.24% (8)	67.26% (6)	68.72% (7)	70.60% (5)	73.40% (6)	71.97% (6)	0.4075(7)	6.37
	XGB	73.94% (4)	75.54% (4)	70.31% (3)	72.83% (4)	72.57% (3)	77.53% (4)	74.97% (3)	0.4797(4)	3.62
	MLP	74.80% (2)	76.25% (3)	**71.26% (1)**	73.67% (2)	**73.55% (1)**	78.27% (3)	75.83% (2)	0.4966(2)	2.00
	Bi-LSTM	71.69% (5)	73.96% (5)	67.76% (5)	70.73% (5)	69.74% (6)	75.70% (5)	72.60% (5)	0.4358(5)	5.12
	2D CNN	69.58% (8)	70.39% (7)	66.04% (8)	68.14% (8)	68.88% (7)	73.01% (8)	70.89% (8)	0.3916(8)	7.75
	2D CNN-Bi-LSTM	70.37% (7)	73.53% (6)	66.15% (7)	69.65% (6)	67.65% (8)	74.83% (7)	71.06% (7)	0.4108(6)	6.75
	Proposed PGN	**74.93% (1)**	76.97% (2)	71.16% (2)	**73.95% (1)**	73.19% (2)	**78.70% (1)**	**75.84% (1)**	**0.5000(1)**	**1.37**
PG	SVM	74.80% (2)	68.24% (6)	**75.00% (1)**	71.46% (4)	**80.44% (1)**	74.66% (5)	**77.44% (1)**	0.4917(2)	2.75
	DT	72.35% (6)	64.66% (8)	72.55% (3)	68.38% (7)	78.97% (2)	72.22% (8)	75.44% (5)	0.4420(7)	5.75
	XGB	74.27% (4)	71.96% (3)	72.27% (4)	72.11% (3)	76.26% (6)	75.98% (3)	76.12% (4)	0.4824(4)	3.87
	MLP	74.40% (3)	69.24% (5)	73.78% (2)	71.44% (5)	78.84% (3)	74.88% (4)	76.81% (2)	0.4837(3)	3.37
	Bi-LSTM	72.82% (5)	**76.11% (1)**	68.56% (7)	72.14% (2)	69.99% (7)	77.31% (2)	73.47% (7)	0.4598(5)	4.5
	2D CNN	69.38% (8)	69.38% (4)	66.08% (8)	67.69% (8)	69.67% (8)	72.49% (6)	70.90% (8)	0.3866(8)	7.25
	2D CNN-Bi-LSTM	71.83% (7)	66.09% (7)	70.97% (6)	68.44% (6)	76.75% (4)	72.47% (7)	74.55% (6)	0.4314(6)	6.12
	Proposed PGN	**75.00% (1)**	75.39% (2)	71.90% (5)	**73.60% (1)**	76.66% (5)	**77.92% (1)**	76.26% (3)	**0.4994(1)**	**2.37**
MMM	SVM	75.33% (2)	75.62% (3)	69.04% (2)	72.18% (3)	75.11% (5)	80.76% (3)	77.84% (2)	0.5027(3)	2.87
	DT	72.69% (6)	71.09% (5)	66.62% (8)	68.78% (6)	73.85% (7)	77.68% (5)	75.72% (6)	0.4462(6)	6.12
	XGB	75.26% (3)	**77.19% (1)**	68.42% (3)	72.54% (2)	73.86% (6)	**81.52% (1)**	77.50% (3)	0.5048(2)	2.62
	MLP	74.40% (5)	75.00% (4)	67.89% (5)	71.27% (4)	73.97% (8)	80.12% (4)	76.92% (4)	0.4849(4)	4.75
	Bi-LSTM	73.41% (4)	70.31% (6)	67.98% (4)	69.12% (5)	75.69% (4)	77.65% (6)	76.66% (5)	0.4581(5)	4.87

(*continued on next page*)

(*continued*)

Stock	Model	Accuracy	Buy signals			Sell signals			MCC	AR
			Recall	Precision	F1	Recall	Precision	F1		
	2D CNN	71.83% (7)	66.25% (7)	66.88% (7)	66.56% (7)	75.92% (2)	75.40% (7)	75.66% (7)	0.4222(7)	6.37
	2D CNN-Bi-LSTM	70.97% (8)	62.03% (8)	66.95% (6)	64.40% (8)	**77.52% (1)**	73.56% (8)	75.49% (8)	0.4003(8)	6.87
	Proposed PGN	**75.93% (1)**	76.09% (2)	**69.77% (1)**	**72.80% (1)**	75.80% (3)	81.20% (2)	**78.41% (1)**	**0.5143(1)**	**1.50**
KO	SVM	72.29% (2)	69.44% (3)	**72.30% (1)**	70.84% (3)	74.97% (4)	72.28% (3)	73.60% (2)	0.4449(2)	2.50
	DT	69.51% (6)	63.44% (6)	70.67% (8)	66.86% (6)	75.22% (3)	68.62% (6)	71.77% (6)	0.3897(6)	5.87
	XGB	72.02% (3)	70.67% (2)	71.35% (5)	71.01% (2)	73.30% (7)	72.65% (2)	72.97% (4)	0.4398(3)	3.50
	MLP	71.36% (5)	68.35% (4)	71.37% (4)	69.83% (4)	74.20% (5)	71.36% (4)	72.75% (5)	0.4264(5)	4.50
	Bi-LSTM	71.63% (4)	67.67% (5)	72.09% (2)	69.81% (5)	73.35% (6)	71.24% (5)	73.24% (3)	0.4318(4)	4.25
	2D CNN	68.52% (7)	59.89% (7)	70.69% (7)	64.84% (7)	76.64% (2)	67.00% (7)	71.50% (7)	0.3710(7)	6.37
	2D CNN-Bi-LSTM	68.19% (8)	58.39% (8)	70.86% (6)	64.02% (8)	**77.41% (1)**	66.41% (8)	71.49% (8)	0.3653(8)	6.87
	Proposed PGN	**73.94% (1)**	**76.67% (1)**	71.59% (3)	**74.04% (1)**	71.37% (8)	**76.48% (1)**	**73.84% (1)**	**0.4806(1)**	**2.12**
CVX	SVM	74.27% (4)	72.28% (6)	71.55% (3)	71.91% (5)	75.94% (3)	76.59% (6)	76.27% (3)	0.4819(4)	4.25
	DT	72.95% (6)	76.63% (2)	68.04% (8)	72.08% (4)	69.87% (8)	78.12% (4)	73.77% (8)	0.4633(6)	5.75
	XGB	75.33% (2)	75.62% (3)	71.76% (2)	73.64% (3)	75.09% (4)	78.63% (2)	76.82% (2)	0.5055(2)	2.50
	MLP	73.81% (5)	73.00% (5)	70.55% (4)	71.75% (6)	74.48% (5)	76.72% (5)	75.59% (4)	0.4738(5)	4.87
	Bi-LSTM	74.34% (3)	**79.10% (1)**	69.07% (6)	**73.75% (1)**	70.35% (7)	**80.08% (1)**	74.90% (6)	0.4930(3)	3.50
	2D CNN	72.02% (7)	66.33% (8)	70.52% (5)	68.36% (8)	**76.79% (1)**	73.15% (8)	74.93% (5)	0.4340(7)	6.12
	2D CNN-Bi-LSTM	71.89% (8)	69.52% (7)	69.02% (7)	69.27% (7)	73.88% (6)	74.33% (7)	74.10% (7)	0.4337(8)	7.12
	Proposed PGN	**75.66% (1)**	74.75% (4)	**72.64% (1)**	73.68% (2)	76.43% (2)	78.33% (3)	**77.37% (1)**	**0.5107(1)**	**1.87**
VZ	SVM	75.20% (4)	66.79% (8)	**63.52% (1)**	65.12% (6)	**79.66% (1)**	81.89% (8)	**80.76% (1)**	0.4593(5)	4.25
	DT	74.74% (5)	71.17% (6)	61.75% (5)	66.13% (4)	76.62% (3)	83.37% (5)	79.85% (5)	0.4644(4)	4.62
	XGB	74.34% (6)	71.18% (5)	61.15% (6)	65.78% (5)	76.01% (5)	83.26% (6)	79.47% (6)	0.4578(6)	5.62
	MLP	75.52% (2)	71.19% (4)	63.01% (2)	66.85% (3)	77.83% (2)	83.59% (3)	80.61% (2)	0.4779(3)	2.62
	Bi-LSTM	75.33% (3)	75.56% (2)	61.78% (4)	67.98% (2)	75.20% (7)	85.30% (2)	79.94% (4)	0.4889(2)	3.25
	2D CNN	72.75% (8)	72.71% (3)	58.62% (8)	64.91% (7)	72.77% (8)	83.41% (4)	77.73% (8)	0.4372(8)	6.75
	2D CNN-Bi-LSTM	73.88% (7)	68.89% (7)	60.88% (7)	64.64% (8)	76.52% (4)	82.26% (7)	79.29% (7)	0.4426(7)	6.75
	Proposed PGN	**75.53% (1)**	**75.57% (1)**	62.07% (3)	**68.16% (1)**	75.51% (6)	**85.35% (1)**	80.13% (3)	**0.4922(1)**	**2.12**
JPM	SVM	76.12% (2)	**80.03% (1)**	70.04% (3)	74.70% (2)	73.05% (4)	**82.29% (1)**	77.40% (2)	0.5270(2)	2.12
	DT	73.94% (6)	78.23% (5)	67.66% (7)	72.56% (6)	70.57% (8)	80.46% (5)	75.19% (7)	0.4846(6)	6.25
	XGB	74.87% (5)	79.13% (2)	68.62% (6)	73.50% (3)	71.51% (6)	81.32% (3)	76.10% (6)	0.5029(4)	4.37
	MLP	74.93% (3)	78.68% (3)	68.86% (5)	73.44% (4)	71.99% (5)	81.09% (4)	76.27% (5)	0.5031(3)	4.00
	Bi-LSTM	74.92% (4)	75.83% (6)	69.85% (4)	72.71% (5)	74.23% (3)	79.59% (6)	76.82% (3)	0.4975(5)	4.50
	2D CNN	71.83% (8)	72.37% (7)	66.57% (8)	69.35% (7)	71.39% (7)	76.65% (7)	73.93% (8)	0.4349(8)	7.50
	2D CNN-Bi-LSTM	73.21% (7)	65.77% (8)	71.22% (2)	68.38% (8)	**79.08% (1)**	74.58% (8)	76.76% (4)	0.4532(7)	5.62
	Proposed PGN	**77.45% (1)**	78.53% (4)	**72.45% (1)**	**75.41% (1)**	76.60% (2)	81.92% (2)	**79.17% (1)**	**0.5479(1)**	**1.62**
V	SVM	71.76% (3)	73.75% (4)	75.96% (4)	74.84% (3)	69.12% (4)	66.57% (3)	67.82% (4)	0.4270(3)	3.5
	DT	70.97% (5)	73.64% (5)	74.94% (5)	74.28% (4)	67.43% (5)	65.92% (4)	66.67% (5)	0.4096(5)	4.75
	XGB	72.16% (2)	74.33% (2)	76.19% (3)	75.25% (2)	69.28% (3)	67.11% (2)	68.18% (2)	0.4346(2)	2.25
	MLP	71.23% (4)	71.43% (6)	76.49% (2)	73.87% (6)	70.96% (2)	65.25% (6)	67.99% (3)	0.4207(4)	4.12
	Bi-LSTM	70.24% (6)	**74.45% (1)**	73.59% (7)	74.02% (5)	64.67% (8)	65.68% (5)	65.17% (6)	0.3920(6)	5.50
	2D CNN	68.85% (7)	70.62% (8)	73.61% (6)	72.08% (8)	66.51% (6)	63.12% (8)	64.77% (7)	0.3693(7)	7.12
	2D CNN-Bi-LSTM	68.84% (8)	70.96% (7)	73.44% (8)	72.18% (7)	66.05% (7)	63.24% (7)	64.61% (8)	0.3684(8)	7.5
	Proposed PGN	**72.69% (1)**	73.98% (3)	**77.12% (1)**	**75.52% (1)**	**70.97% (1)**	**67.35% (1)**	**69.11% (1)**	**0.4471(1)**	**1.25**

Appendix B. The comparative models

B.1 2D CNN-Bi-LSTM Model

In the study by Khodaee et al. [28], a novel framework structure was proposed to enhance the accuracy of turning point detection in financial time series. This framework combines the power of the convolutional neural network (CNN) and long short-term memory (LSTM) neural network. The CNN component extracts valuable information through its convolutional layers, while the LSTM component aims to capture dependencies within the time series. However, in the comparative model of this research, the LSTM component is further enhanced. Drawing inspiration from research [76], a Bi-directional LSTM network (Bi-LSTM) is employed. This architecture incorporates both forward and backward LSTM networks for each training series. The structure utilized for 2D CNN-Bi-LSTM network for our BUY/SELL prediction is provided in Fig. 12.

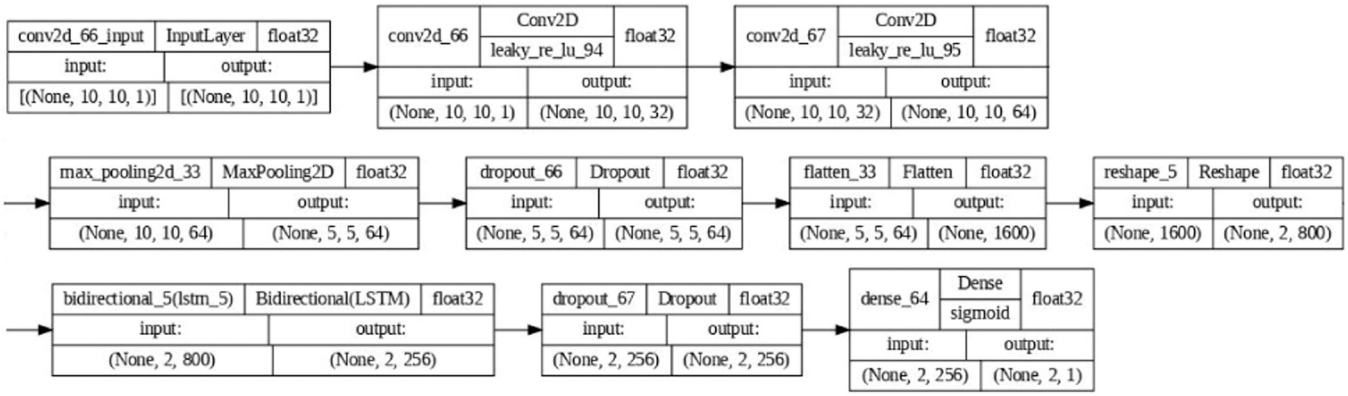

Fig. 12. 2D CNN-Bi-LSTM's architecture as a comparative model.

.

B.2 2D CNN Model

Based on [15], a novel approach was introduced, which involves transforming one-dimensional financial time series into a two-dimensional visual representation. This transformation enables the utilization of the powerful deep convolutional neural network (CNN) for an algorithmic trading system. In the proposed neural network, images served as the two-dimensional representation of the financial time series data derived from technical analysis. These images were fed as input to a deep CNN model for the classification task, enabling the development of a new financial trading system. The structure utilized for this network is provided in Fig. 13.

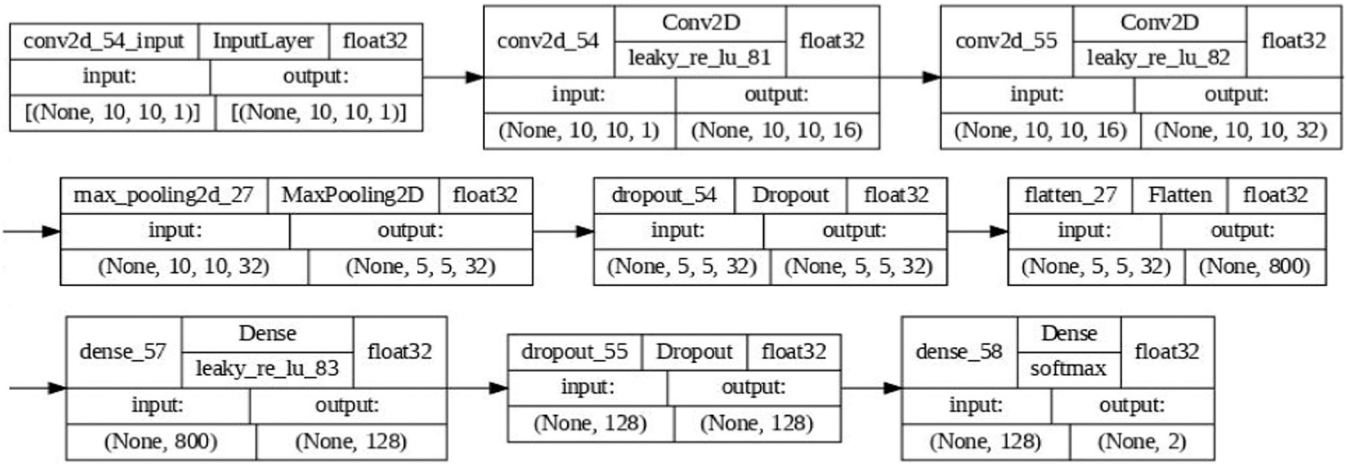

Fig. 13. 2D CNN's architecture as a comparative model.

.

B.3 Bi-LSTM Model

In the aforementioned study [76], in the context of time series processing, LSTM models have a limitation of disregarding future information. To overcome this limitation, Bi-LSTM (Bidirectional Long Short-Term Memory) model is introduced. It employs two separate hidden layers that process the series data in both forward and backward directions, building upon the LSTM architecture. These two hidden layers are then connected to the same output layer, allowing the model to capture and retain both past and future information as the current time basis for analyzing time series data. As a result, Bi-LSTM exhibits improved theoretical prediction performance compared to unidirectional LSTM models. The structure utilized for this model is provided in Fig. 14.

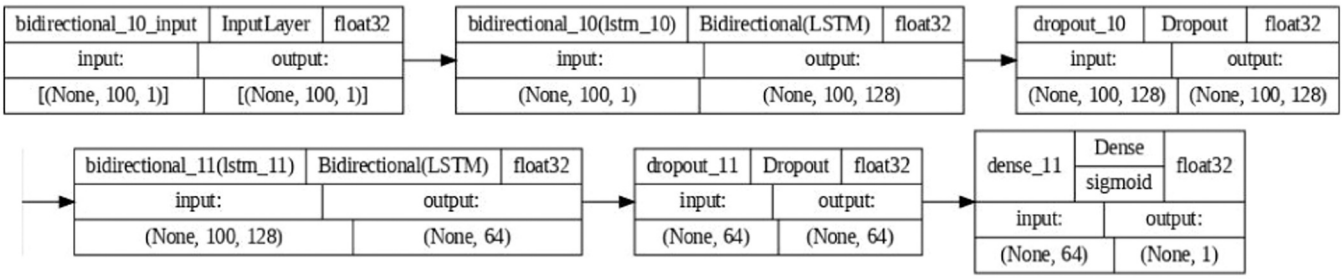

Fig. 14. Bi-LSTM's architecture as a comparative model.

.

B.4 MLP Model

In the research by Namdari and Durrani [77], neural network-based methods, specifically the multilayer perceptron (MLP), have gained popularity in stock forecasting due to its ability to learn and adapt through learning algorithms. MLP neural networks are considered suitable for classification and are often deemed superior to statistical techniques. However, determining the appropriate MLP architecture can be challenging, as it involves selecting the learning algorithms, determining the number of layers, specifying the number of neurons in each layer, and choosing suitable activation functions between layers. Consequently, it is crucial to identify the optimal parameters for the MLP network. The parameters used here are outlined in Table 4.

Table 4
MLP's parameters as a comparative model.

Methods / **Parameters**	Doing trail & error to find the approximate range of parameter	Doing grid search to find the best parameter
learning_rate	[*"constant"*, *"adaptive"*]	***"adaptive"***
solver	[*"sgd"*, *"adam"*]	***"sgd"***
activation	[*"softmax"*, *"relu"*]	***"relu"***
alpha	$[0.1, 0.01, 0.001, 0.0001]$	***0.1***
max_*iter*	$[50, 100, 150, 200, 250, 300]$	***300***
hidden_layer_sizes	$[(5,5),(6,6),(7,7),(8,8),(9,9)]$	$(7,7)$

B.5 XGB Model

As per the findings of the study [78], there exists a technique known as boosting, which aims to construct a powerful learner by combining multiple weak learners. This algorithm involves a collection of predictors that collectively attempt to predict the same target variable. The fundamental principle of boosting is to iteratively match a set of weak learners with weighted versions of the training data. After each iteration, greater emphasis is placed on the training samples that were misclassified in the previous rounds. Eventually, all successive models are assigned weights based on their performance, and their outputs are combined using majority voting on different issues to create the final model. The parameters associated with this model are also presented in Table 5.

Table 5
XGB's parameters as a comparative model.

Methods / **parameters**	Doing trail & error to find the approximate range of parameter	Doing grid search to find the best parameter
learning_rate	$[0.1, 0.01, 0.001, 0.0001]$	***0.01***
max_*depth*	*range*$(2, 10, 1)$	***2***
n_estimators	*range*$(60, 220, 40)$	***140***

B.6 SVM Model

As per the findings of the study [37], the support vector machine (SVM) model is a powerful machine learning technique that has gained popularity due to its attractive features and excellent performance across a wide range of problems. Unlike conventional neural networks, the SVM model captures the geometric features of the variable space without extracting network weights from the training data. This allows it to find optimal solutions even with small training set sizes. Additionally, SVM incorporates the principle of structural risk minimization (SRM), which has been shown to outperform the empirical risk minimization (ERM) principle used by traditional neural networks. SRM aims to minimize an upper bound on the generalization error, whereas ERM minimizes the error in the training data. Consequently, SVM solutions have the potential to be the best solution (global optimum), whereas other neural network models may only reach local optima. Despite SVM's impressive performance, its classification ability and classifier generalization can be affected by the dimensionality or number of feature variables. The parameters associated with this model are

shown in Table 6.

Table 6
SVM's parameters as a comparative model.

Methods / Parameters	Doing trail & error to find the approximate range of parameter	Doing grid search to find the best parameter
kernel	[*"linear"*, *"rbf"*]	***"rbf"***
gamma	[1, 0.1, 0.01, 0.001, 0.0001]	***0.0001***
C	[1, 10, 100, 1000]	***100***

B.7 Decision Tree

Based on the findings of research [79], the decision tree classification algorithm can be viewed as a process of learning from if/then/else problems layer by layer. It determines the feature criterion for each layer based on the amount of information gained. However, the performance of decision tree classification heavily relies on the training dataset, often leading to overfitting. This means that the importance of each feature in the decision outcome may vary, and the decision tree produced by the algorithm changes when the training set is altered. The specific parameters of this model are shown in Table 7.

Table 7
Decision Tree's parameters as a comparative model.

Methods / Parameters	Doing trail & error to find the approximate range of parameter	Doing grid search to find the best parameter
criterion	[*"gini"*, *"entropy"*]	***"gini"***
max_*depth*	*range*(1, 10, 1)	***4***
max_*features*	*range*(5, 50, 5)	***10***
min_*samples_leaf*	*range*(1, 20, 2)	***3***

Appendix C. Supporting information

Supplementary data associated with this article can be found in the online version at doi:10.1016/j.asoc.2023.111213.

References

[1] A. Abraham N.S. Philip P. Saratchandran Modeling Chaotic Behavior of Stock Indices Using Intelligent Paradigms 2004. https://doi.org/10.48550/arXiv.cs/0405018.

[2] D. Berger, K. Pukthuanthong, J. Jimmy Yang, International diversification with frontier markets, J. Financ. Econ. 101 (2011) 227–242, https://doi.org/10.1016/j.jfineco.2011.02.009.

[3] R. Singh, S. Srivastava, Stock prediction using deep learning, Multimed. Tools Appl. 76 (2017) 18569–18584, https://doi.org/10.1007/s11042-016-4159-7.

[4] M.Y. Chen, B.T. Chen, A hybrid fuzzy time series model based on granular computing for stock price forecasting, Inf. Sci. (Ny.). 294 (2015) 227–241, https://doi.org/10.1016/j.ins.2014.09.038.

[5] M. Jiang, L. Jia, Z. Chen, W. Chen, The two-stage machine learning ensemble models for stock price prediction by combining mode decomposition, extreme learning machine and improved harmony search algorithm, Ann. Oper. Res. 309 (2022) 553–585, https://doi.org/10.1007/s10479-020-03690-w.

[6] J. Zhang, L. Li, W. Chen, Predicting stock price using two-stage machine learning techniques, Comput. Econ. 57 (2021) 1237–1261, https://doi.org/10.1007/s10614-020-10013-5.

[7] L. Cagliero, P. Garza, G. Attanasio, E. Baralis, Training ensembles of faceted classification models for quantitative stock trading, Computing 102 (2020) 1213–1225, https://doi.org/10.1007/s00607-019-00776-7.

[8] M. Ananthi, K. Vijayakumar, Stock market analysis using candlestick regression and market trend prediction (CKRM), J. Ambient Intell. Humaniz. Comput. (2020), https://doi.org/10.1007/s12652-020-01892-5.

[9] H. Tang, P. Dong, Y. Shi, A new approach of integrating piecewise linear representation and weighted support vector machine for forecasting stock turning points, Appl. Soft Comput. J. 78 (2019) 685–696, https://doi.org/10.1016/j.asoc.2019.02.039.

[10] L. de Haan, C. Mercadier, C. Zhou, Adapting extreme value statistics to financial time series: dealing with bias and serial dependence, Financ. Stoch. 20 (2016) 321–354, https://doi.org/10.1007/s00780-015-0287-6.

[11] Q. Liang, W. Rong, J. Zhang, J. Liu, Z. Xiong, Restricted Boltzmann machine based stock market trend prediction, Proc. Int. Jt. Conf. Neural Netw. 2017-May (2017) 1380–1387, https://doi.org/10.1109/IJCNN.2017.7966014.

[12] J. Zhao, D. Zeng, S. Liang, H. Kang, Q. Liu, Prediction model for stock price trend based on recurrent neural network, J. Ambient Intell. Humaniz. Comput. 12 (2021) 745–753, https://doi.org/10.1007/s12652-020-02057-0.

[13] D.M.Q. Nelson, A.C.M. Pereira, R.A. De Oliveira, Stock Market's Price Movement Prediction With LSTM Neural Networks, IEEE Trans. Neural Networks, 2017, pp. 1419–1426, https://doi.org/10.1109/IJCNN.2017.7966019.

[14] M.Y. Chen, C.H. Liao, R.P. Hsieh, Modeling public mood and emotion: stock market trend prediction with anticipatory computing approach, Comput. Hum. Behav. 101 (2019) 402–408, https://doi.org/10.1016/j.chb.2019.03.021.

[15] O.B. Sezer, A.M. Ozbayoglu, Algorithmic financial trading with deep convolutional neural networks: time series to image conversion approach, Appl. Soft Comput. J. 70 (2018) 525–538, https://doi.org/10.1016/j.asoc.2018.04.024.

[16] S. Barra, S.M. Carta, A. Corriga, A.S. Podda, D.R. Recupero, Deep learning and time series-To-image encoding for financial forecasting, IEEE/CAA J. Autom. Sin. 7 (2020) 683–692, https://doi.org/10.1109/JAS.2020.1003132.

[17] F.J. Fabozzi, F. Gupta, H.M. Markowitz, The legacy of modern portfolio theory, J. Invest. 11 (2002) 7–22, https://doi.org/10.3905/joi.2002.319510.

[18] G.F. Deng, W.T. Lin, C.C. Lo, Markowitz-based portfolio selection with cardinality constraints using improved particle swarm optimization, Expert Syst. Appl. 39 (2012) 4558–4566, https://doi.org/10.1016/j.eswa.2011.09.129.

[19] W. Wang, W. Li, N. Zhang, K. Liu, Portfolio formation with preselection using deep learning from long-term financial data, Expert Syst. Appl. 143 (2020) 113042, https://doi.org/10.1016/j.eswa.2019.113042.

[20] Y. Zhang, X. Li, S. Guo, Portfolio selection problems with Markowitz ' s mean – variance framework: a review of literature, Fuzzy Optim. Decis. Mak. 17 (2018) 125–158, https://doi.org/10.1007/s10700-017-9266-z.

[21] C.F. Huang, A hybrid stock selection model using genetic algorithms and support vector regression, Appl. Soft Comput. J. 12 (2012) 807–818, https://doi.org/10.1016/j.asoc.2011.10.009.

[22] C. Krauss, X.A. Do, N. Huck, Deep neural networks, gradient-boosted trees, random forests: statistical arbitrage on the S&P 500, Eur. J. Oper. Res. 259 (2017) 689–702, https://doi.org/10.1016/j.ejor.2016.10.031.

[23] F.D. Paiva, R. Tom, N. Cardoso, Decision-making for financial trading: a fusion approach of machine learning and portfolio selection, Expert Syst. Appl. 115 (2018) 635–655, https://doi.org/10.1016/j.eswa.2018.08.003.

[24] V.D. Ta, C.M. Liu, D.A. Tadesse, Portfolio optimization-based stock prediction using long-short term memory network in quantitative trading, Appl. Sci. 10 (2) (2020) 437, https://doi.org/10.3390/app10020437.

[25] V.D. Ta, C.M. Liu, D. Addis, Prediction and portfolio optimization in quantitative trading using machine learning techniques, ACM Int. Conf. Proc. Ser. (2018) 98–105, https://doi.org/10.1145/3287921.3287963.

[26] B. Chen, J. Zhong, Y. Chen, A hybrid approach for portfolio selection with higher-order moments: empirical evidence from Shanghai Stock Exchange, Expert Syst. Appl. 145 (2020) 113104, https://doi.org/10.1016/j.eswa.2019.113104.

[27] V. Vasudevan, M. Bassenne, M.T. Islam, L. Xing, Image classification using graph neural network and multiscale wavelet superpixels, Pattern Recognit. Lett. 166 (2023) 89–96, https://doi.org/10.1016/j.patrec.2023.01.003.

[28] P. Khodaee, A. Esfahanipour, H. Mehtari Taheri, Forecasting turning points in stock price by applying a novel hybrid CNN-LSTM-ResNet model fed by 2D segmented images, Eng. Appl. Artif. Intell. 116 (2022) 105464, https://doi.org/10.1016/j.engappai.2022.105464.

[29] S. Sinha, S. Mishra, V. Mishra, T. Ahmed, Sector influence aware stock trend prediction using 3D convolutional neural network, J. King Saud. Univ. - Comput. Inf. Sci. 34 (2022) 1511–1522, https://doi.org/10.1016/j.jksuci.2022.02.008.

[30] Y.C. Chen, W.C. Huang, Constructing a stock-price forecast CNN model with gold and crude oil indicators[Formula presented], Appl. Soft Comput. 112 (2021) 107760, https://doi.org/10.1016/j.asoc.2021.107760.

[31] H. Konno, H. Yamazaki, Mean-absolute deviation portfolio optimization model and its applications to tokyo stock market, Manag. Sci. 37 (1991) 519–531, https://doi.org/10.1287/mnsc.37.5.519.

[32] G.J. Alexander, A.M. Baptista, Economic implications of using a mean-VaR model for portfolio selection: a comparison with mean-variance analysis, J. Econ. Dyn. Control. 26 (2002) 1159–1193, https://doi.org/10.1016/S0165-1889(01)00041-0.

[33] R.T. Rockafellar, S. Uryasev, Optimization of conditional value-at-risk, J. Risk. 2 (2000) 21–41, https://doi.org/10.21314/jor.2000.038.

[34] A. Chekhlov, S. Uryasev, M. Zabarankin, Portfolio optimization with drawdown constraints, Supply Chain Financ (2004) 209–228, https://doi.org/10.1142/9789812562586_0013.

[35] P. Krokhmal, S. Uryasev, G. Zrazhevsky, 29. numerical comparison of conditional value-at-risk and conditional drawdown-at-risk approaches: application to hedge funds, Appl. Stoch. Program. 32611 (2005) 609–631, https://doi.org/10.1137/1.9780898718799.ch29.

[36] I.R. Parray, S.S. Khurana, M. Kumar, A.A. Altalbe, Time series data analysis of stock price movement using machine learning techniques, Soft Comput. 24 (2020) 16509–16517, https://doi.org/10.1007/s00500-020-04957-x.

[37] M.C. Lee, Using support vector machine with a hybrid feature selection method to the stock trend prediction, Expert Syst. Appl. 36 (2009) 10896–10904, https://doi.org/10.1016/j.eswa.2009.02.038.

[38] W.C. Chiang, D. Enke, T. Wu, R. Wang, An adaptive stock index trading decision support system, Expert Syst. Appl. 59 (2016) 195–207, https://doi.org/10.1016/j.eswa.2016.04.025.

[39] A. Picasso, S. Merello, Y. Ma, L. Oneto, E. Cambria, Technical analysis and sentiment embeddings for market trend prediction, Expert Syst. Appl. 135 (2019) 60–70, https://doi.org/10.1016/j.eswa.2019.06.014.

[40] X. dan Zhang, A. Li, R. Pan, Stock trend prediction based on a new status box method and AdaBoost probabilistic support vector machine, Appl. Soft Comput. J. 49 (2016) 385–398, https://doi.org/10.1016/j.asoc.2016.08.026.

[41] M.O. Özorhan, İ.H. Toroslu, O.T. Şehitoğlu, Short-term trend prediction in financial time series data, 2019. https://doi.org/10.1007/s10115–018-1303-x.

[42] L. Lei, Wavelet neural network prediction method of stock price trend based on rough set attribute reduction, Appl. Soft Comput. J. 62 (2018) 923–932, https://doi.org/10.1016/j.asoc.2017.09.029.

[43] R. Bisoi, P.K. Dash, A.K. Parida, Hybrid variational mode decomposition and evolutionary robust kernel extreme learning machine for stock price and movement prediction on daily basis, Appl. Soft Comput. J. 74 (2019) 652–678, https://doi.org/10.1016/j.asoc.2018.11.008.

[44] A.M. Ozbayoglu, M.U. Gudelek, O.B. Sezer, Deep learning for financial applications: a survey, Appl. Soft Comput. J. 93 (2020) 106384, https://doi.org/10.1016/j.asoc.2020.106384.

[45] B. Moews, J.M. Herrmann, G. Ibikunle, Lagged correlation-based deep learning for directional trend change prediction in financial time series, Expert Syst. Appl. 120 (2019) 197–206, https://doi.org/10.1016/j.eswa.2018.11.027.

[46] N. Naik, B.R. Mohan, Intraday stock prediction based on deep neural network, Natl. Acad. Sci. Lett. 43 (2020) 241–246, https://doi.org/10.1007/s40009-019-00859-1.

[47] D. Wu, X. Wang, J. Su, B. Tang, S. Wu, A labeling method for financial time series prediction based on trends, Entropy 22 (2020) 1–25, https://doi.org/10.3390/e22101162.

[48] O.B. Sezer, A.M. Ozbayoglu, Financial trading model with stock bar chart image time series with deep convolutional neural networks, Intell. Autom. Soft Comput. 26 (2020) 323–334, https://doi.org/10.48550/arXiv.1903.04610.

[49] J. Long, Z. Chen, W. He, T. Wu, J. Ren, 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. J. 91 (2020) 106205, https://doi.org/10.1016/j.asoc.2020.106205.

[50] Y. Hao, Q. Gao, Predicting the trend of stock market index using the hybrid neural network based on multiple time scale feature learning, Appl. Sci. 10 (2020), https://doi.org/10.3390/app10113961.

[51] F. Feng, X. He, X. Wang, C. Luo, Y. Liu, T.S. Chua, Temporal relational ranking for stock prediction, ACM Trans. Inf. Syst. 37 (2019), https://doi.org/10.1145/3309547.

[52] B. Xu, H. Shen, B. Sun, R. An, Q. Cao, X. Cheng, Towards consumer loan fraud detection: graph neural networks with role-constrained conditional random field, 35th AAAI Conf. Artif. Intell. AAAI 2021. 5B (2021) 4537–4545, https://doi.org/10.1609/aaai.v35i5.16582.

[53] T. Liang, G. Zeng, Q. Zhong, J. Chi, J. Feng, X. Ao, J. Tang, Credit risk and limits forecasting in e-commerce consumer lending service via multi-view-aware mixture-of-experts nets, WSDM 2021 - Proc. 14th ACM Int. Conf. Web Search Data Min. (2021) 229–237, https://doi.org/10.1145/3437963.3441743.

[54] W. Kudo, M. Nishiguchi, F. Toriumi, GCNEXT: graph convolutional network with expanded balance theory for fraudulent user detection, Soc. Netw. Anal. Min. 10 (2020), https://doi.org/10.1007/s13278-020-00697-w.

[55] S.X. Rao, S. Zhang, Z. Han, Z. Zhang, W. Min, M. Cheng, Y. Shan, Y. Zhao, C. Zhang, Suspicious Massive Registration Detection via Dynamic Heterogeneous Graph Neural Networks, (2020). ⟨http://arxiv.org/abs/2012.10831⟩.

[56] M. Harl, S. Weinzierl, M. Stierle, M. Matzner, Explainable predictive business process monitoring using gated graph neural networks, J. Decis. Syst. 29 (2020) 312–327, https://doi.org/10.1080/12460125.2020.1780780.

[57] H. Markowitz, Portfolio Selection, J. Financ. 7 (1952) 15, https://doi.org/10.2307/2345307.

[58] S.S. Meghwani, M. Thakur, Multi-objective heuristic algorithms for practical portfolio optimization and rebalancing with transaction cost, Appl. Soft Comput. J. 67 (2018) 865–894, https://doi.org/10.1016/j.asoc.2017.09.025.

[59] X. Li, A.S. Uysal, J.M. Mulvey, Multi-period portfolio optimization using model predictive control with mean-variance and risk parity frameworks, Eur. J. Oper. Res. 299 (2022) 1158–1176, https://doi.org/10.1016/j.ejor.2021.10.002.

[60] Y. Dai, Z. Qin, Multi-period uncertain portfolio optimization model with minimum transaction lots and dynamic risk preference, Appl. Soft Comput. 109 (2021) 107519, https://doi.org/10.1016/j.asoc.2021.107519.

[61] X. Gong, L. Min, C. Yu, Multi-period portfolio selection under the coherent fuzzy environment with dynamic risk-tolerance and expected-return levels, Appl. Soft Comput. 114 (2022), https://doi.org/10.1016/j.asoc.2021.108104.

[62] H. Babazadeh, A. Esfahanipour, A novel multi period mean-VaR portfolio optimization model considering practical constraints and transaction cost, J. Comput. Appl. Math. 361 (2019) 313–342, https://doi.org/10.1016/j.cam.2018.10.039.

[63] M.A. Akbay, C.B. Kalayci, O. Polat, A parallel variable neighborhood search algorithm with quadratic programming for cardinality constrained portfolio optimization, Knowl. -Based Syst. 198 (2020) 105944, https://doi.org/10.1016/j.knosys.2020.105944.

[64] T. Fischer, C. Krauss, Deep learning with long short-term memory networks for financial market predictions, Eur. J. Oper. Res. 270 (2018) 654–669, https://doi.org/10.1016/j.ejor.2017.11.054.

[65] Y. Ma, R. Han, W. Wang, Portfolio optimization with return prediction using deep learning and machine learning, Expert Syst. Appl. 165 (2021) 113973, https://doi.org/10.1016/j.eswa.2020.113973.

[66] W. Chen, H. Zhang, M.K. Mehlawat, L. Jia, Mean–variance portfolio optimization using machine learning-based stock price prediction, Appl. Soft Comput. 100 (2021) 106943, https://doi.org/10.1016/j.asoc.2020.106943.

[67] A. Chaweewanchon, R. Chaysiri, Markowitz mean-variance portfolio optimization with predictive stock selection using machine learning, Int. J. Financ. Stud. 10 (2022), https://doi.org/10.3390/ijfs10030064.

[68] W. Chen, H. Zhang, L. Jia, A novel two-stage method for well-diversified portfolio construction based on stock return prediction using machine learning, North Am. J. Econ. Financ. 63 (2022), https://doi.org/10.1016/j.najef.2022.101818.

[69] J. Behera, A.K. Pasayat, H. Behera, P. Kumar, Prediction based mean-value-at-risk portfolio optimization using machine learning regression algorithms for multi-national stock markets, Eng. Appl. Artif. Intell. 120 (2023), https://doi.org/10.1016/j.engappai.2023.105843.

[70] Y. Ma, W. Wang, Q. Ma, A novel prediction based portfolio optimization model using deep learning, Comput. Ind. Eng. 177 (2023), https://doi.org/10.1016/j.cie.2023.109023.

[71] M. Ashrafzadeh, H. Mehtari, M. Gharehgozlou, S. Hashemkhani, Journal of King Saud University – computer and Clustering-based return prediction model for stock pre-selection in portfolio optimization using PSO-CNN + MVF, J. King Saud. Univ. - Comput. Inf. Sci. 35 (2023) 101737, https://doi.org/10.1016/j.jksuci.2023.101737.

[72] W. Chen, M. Jiang, W.G. Zhang, Z. Chen, A novel graph convolutional feature based convolutional neural network for stock trend prediction, Inf. Sci. (Ny.). 556 (2021) 67–94, https://doi.org/10.1016/j.ins.2020.12.068.

[73] Y. Peng, P.H.M. Albuquerque, H. Kimura, C.A.P.B. Saavedra, Feature selection and deep neural networks for stock price direction forecasting using technical analysis indicators, Mach. Learn. Appl. 5 (2021) 100060, https://doi.org/10.1016/j.mlwa.2021.100060.

[74] M. Rashidpoor Toochaei, F. Moeini, Evaluating the performance of ensemble classifiers in stock returns prediction using effective features, Expert Syst. Appl. 213 (2023) 0–2, https://doi.org/10.1016/j.eswa.2022.119186.

[75] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, M.M. Bronstein, Geometric deep learning on graphs and manifolds using mixture model CNNs, Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognit., CVPR 2017. 2017-Janua (2017) 5425–5434, https://doi.org/10.1109/CVPR.2017.576.

[76] M. Yang, J. Wang, Adaptability of financial time series prediction based on BiLSTM, Procedia Comput. Sci. 199 (2021) 18–25, https://doi.org/10.1016/j.procs.2022.01.003.

[77] A. Namdari, T.S. Durrani, A multilayer feedforward perceptron model in neural networks for predicting stock market short-term trends, Oper. Res. Forum 2 (2021) 1–30, https://doi.org/10.1007/s43069-021-00071-2.
[78] J. Nobre, R.F. Neves, Combining principal component analysis, discrete wavelet transform and XGBoost to trade in the financial markets, Expert Syst. Appl. 125 (2019) 181–194, https://doi.org/10.1016/j.eswa.2019.01.083.
[79] J. Xianya, H. Mo, L. Haifeng, Stock classification prediction based on spark, Procedia Comput. Sci. 162 (2019) 243–250, https://doi.org/10.1016/j.procs.2019.11.281.