Applied Soft Computing Journal 152 (2024) 111233

Contents lists available at ScienceDirect

Applied Soft Computing

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

Learning and processing framework using Fuzzy Deep Neural Network for trading and portfolio rebalancing

Nicole Hui Lin Kan [a], Qi Cao [b,*], Chai Quek [a]

[a] School of Computer Science and Engineering, Nanyang Technological University, Singapore
[b] School of Computing Science, University of Glasgow, UK

ARTICLE INFO

Keywords:
Moving Average Convergence Divergence
Fuzzy Deep Neural Network
Portfolio rebalancing
Reinforcement Learning
Genetic Algorithm

ABSTRACT

Trading strategies are an interesting topic of financial research. Moving Average Convergence Divergence (MACD) indicator is susceptible to performing worse than expected in unstable financial markets. This paper first presents a data-driven Interpretable Fuzzy Deep Neural Network (IFDNN) that provides insight into neural network inferences using fuzzy logic. Fuzzy rules are induced from the inference process of Neural Networks. Next, a learning and processing framework is proposed using IFDNN to detect trend reversals by forecasting look-ahead prices. IFDNN not only learns the drifts and shifts in market patterns, but also provides traders an option to dive into the reasoning behind why Neural Networks predict certain values. Genetic Algorithms are used to optimise trading parameters of the proposed framework. The proposed framework can perform portfolio rebalancing. The effectiveness of the framework is evaluated on three financial market indexes. The whipsaw effects cause frequent entrances and exits from the market. In this paper, a custom percentage oscillator is implemented to avoid this issue. The performances of the proposed framework using f-MACD are compared with those of the vanilla MACD. Two types of Reinforcement Learning models, Advantage Actor Critic and Deep Deterministic Policy Gradient are incorporated into the proposed framework with results compared.

1. Introduction

Financial stock prices are crucial indicators relevant to the health of the economy. To be able to understand the risks or trends in the market, analytical methods are required to provide not only accurate predictions, but also robust strategies with good performance in different market conditions. Conventionally, financial market price predictions are performed by statistical forecasting methods, such as time series analysis [1], autoregressive models [2] and machine learning methods [3,4]. Such models produce results based on mathematical and computational theory, which is not easily understood by investors. Furthermore, in the case of neural networks, although the accuracy advancements of deep neural network and training models are obtained significantly, the complexity of deep architecture keeps increasing by stacking more layers [5]. Challenges that are caused by the black box nature of deep structures surface out on the human interpretability for the insights how deep neural networks are connected to derive the results [6–9]. The reasoning and decision-making process of the networks are not straightforwardly interpretable by human users [10,11]. On the other hand, fuzzy systems are able to provide semantic meanings by having fuzzy sets and corresponding linguistic labels, but they lack the learning ability that neural networks have.

Fuzzy neural networks have been applied in many applications [12–15], which aim to address the challenges of black box to understand the inner workings of a deep neural network. By integrating fuzzy systems, the inputs and outputs act as antecedents and consequents, with fuzzy rules endowing the reasoning performed by the deep structure. Fuzzy rules aim to provide insights to deep learning in semantic forms that are human interpretable.

In this paper, we first present the proposed Interpretable Fuzzy Deep Neural Network (IFDNN) adopts the data-driven approach to discover patterns automatically, exploiting the advantage of superior learning abilities in neural networks. It does not require prior assumptions regarding input data and does not require domain expertise for defining the fuzzy partition space. This motivates the idea of an IFDNN, which alleviates the aforementioned issues. By including an intelligent system such as fuzzy logic [16], it expands the capability of a vanilla neural network by including reasoning. Thus, fuzzy neural networks combine the interpretability of a fuzzy inference system with the high accuracy performance of a neural network [17–19]. The IFDNN is integrated with a fuzzy system, namely the Pseudo Outer-Product Based Fuzzy Neural Network (POPFNN) [20] with a deep Long Short-Term Memory (LSTM) [21] neural network to predict financial market index prices.

* Corresponding author.
E-mail addresses: nkan001@e.ntu.edu.sg (N.H.L. Kan), qi.cao@glasgow.ac.uk (Q. Cao), ashcquek@ntu.edu.sg (C. Quek).

https://doi.org/10.1016/j.asoc.2024.111233
Received 7 June 2023; Received in revised form 9 October 2023; Accepted 30 December 2023
Available online 4 January 2024

The proposed IFDNN model is a five-layer hybrid fuzzy neural network that forecasts closing prices using the deep structure and extracts meaningful information from the input data and output predictions using fuzzy rules. Furthermore, to quantify the relative importance of fuzzy rules, Hebbian Learning is employed to obtain the pseudo-weights, allowing the rules to be ranked with respect to one another.

Algorithmic finance research combines computer science and finance, where decisions can be made dynamically based on certain market conditions. Algorithmic trading or automated trading is designed to follow certain guidelines and decide when to perform a trade, at which trading price and quantity [22]. This can be done at accuracies and speeds that human traders cannot achieve [23,24]. There have been many works focused on using a variety of indicators to better tune trading strategies for obtaining better results [25–27]. Machine learning and neural networks are able to handle concept drift in time series data [28,29], including financial market data [30,31].

Algorithmic trading needs to fine tune parameters to improve trading performance. The process of finding the best combination of parameters for different algorithms is usually time-consuming and computationally intensive. This is because the model would need to be trained repeatedly on each permutation of parameters. This process could be done manually or automatically. However, the manual method is tedious and impractical when there are too many combinations of parameters to test. On the other hand, automated hyperparameter tuning is where ideal parameters are found using an algorithm that automates and optimises the process.

There have been many reported methods for automated hyperparameter tuning, including Grid Search, Random Search [32], Bayesian [33], Gradient-based [34], Genetic Algorithms (GA) [35–38], and other Evolutionary Algorithms (EA) [39]. EA is able to handle large search spaces and provide good results for reasonable computational time, which is advantageous over exhaustive or manual search techniques [40]. As a subset of EA, GA is often employed to optimise hyperparameters of machine learning models [9,35]. GA is reported to perform well for parameters optimisation in time series algorithms [36]. GA is utilised for hyperparameters tuning of Support Vector Regression for predicting the prices of five market indexes [35]. GA is employed to search optimal or sub-optimal trading rules to derive decisions in a stock trading system [41]. Two technical analysis indicators are optimised by GA in the rule-based trading algorithms for financial portfolio rebalancing [37]. GA acts as the optimiser on the feature selections for a machine learning method, Extreme gradient boosting consisting of three-stage feature engineering process, to predict the stock price trend [9]. GA is applied to help obtain the optimal portfolio for a multi-period mean-VaR model [38].

A portfolio is a collection of financial investments including stocks, bonds, commodities, cash and exchange traded funds (ETFs). As an important concept in portfolio management, diversification helps with reducing risks by allocating investments among a variety of financial categories. The aim is to maximise returns by investing in different areas that react differently towards the same event. Dynamic portfolio rebalancing is the process of changing the weights of a portfolio of assets over a time span. It involves periodically buying or selling assets in a portfolio, according to changes of dynamic market conditions. If a certain market is performing better than others, the investor would increase his portfolio weights in that asset while reducing the others. Many machine learning methods have been deployed for optimal portfolio allocation, including evolutionary computation [42], GA [43], K-Means [44], Markov-Driven strategies [45], and Reinforcement Learning (RL) [4,46].

In the portfolio rebalancing, the buy and sell decisions will be derived according to the price trend reversals. As such, the detection accuracy of trend reversals is very important. Technical analysis using indicators plays an important role in stock price predictions. Time series analysis is a main method for forecasting future share prices that deals with a series of price data gathered during certain time frames [47]. Future price behaviours can be extracted according to the previously observed patterns. Artificial neural networks (ANN) including Recurrent Neural Networks (RNN) are one of the most popular approaches for the predictions [47,48]. Time series stock price data is inherently noisy which may affect the performances of RNN. Features extraction steps using Ensemble Empirical Mode Decomposition are added as a preprocessor to decompose the raw data first, before feeding into RNN [48]. A hybrid ANN is introduced with GA utilised for feature selection of input variables for stock price predictions that performs better than another time series model, autoregressive integrated moving average (ARIMA) [47]. LSTM as one of the RNN models is the most preferred implementation for the predictions of financial stock prices, where many prior works are reported in literature [49–51]. According to the survey, LSTM is the most dominant architecture in time-series financial price forecasting [49,51]. But overfitting is a potential issue for LSTM caused by the increment of the number of variables [49,52]. The dimension reduction is one of the solutions to address overfitting. However, the dimension reduction may cause information loss and reduce the accuracy or interpretability [49]. A prior work is reported to combine LSTM and random forest, as random forest is robust to utilise many variables without overfitting [49]. AdaBoost-LSTM ensemble model is introduced to reduce overfitting for cryptocurrency trading [52]. Two neural networks are combined with a Gated Recurrent Unit (GRU) at the first layer and a LSTM at the second layer to increase the prediction performance for future price predictions of foreign currencies [53]. While ANN, LSTM, or deep networks approaches achieve good accuracies for time series price forecasting, they still encounter the challenges on the interpretability of black box deep structure.

GA is reported to optimise technical analysis indicators in stock trading strategies. Trading strategy of foreign currencies using a directional changes (DC) indicator for trend reverses optimised by GA is introduced [54]. Typically, traders use different technical indicators to determine the strength or weakness of stock prices to predict future price movements [37,55,56]. Several popular trading technical indicators are employed for financial markets forecasting, including simple moving average (SMA), Moving Average Convergence and Divergence (MACD), Relative Strength Index (RSI), Price Percentage Oscillator (PPO) and Bollinger Bands [49,57]. Among these technical indicators, MACD is one of the most popular and widely used indicators by many investors and traders worldwide [50,57–59]. MACD is an indicator that can derive the information on both trend and momentum of stock prices [60]. MACD can be used to determine the short-term trend reversal of the market [61]. The MACD indicator is employed to predict the trend of 30 stock prices in DOW index [51]. Buy and sell signals are generated using the MACD indicator in experiments with three major indexes: Ibex35, DAX and Dow Jones Industrial [62]. Traditional MACD model parameters may not be optimal. Parameters of MACD can be selected by GA or other methods [59], where trading strategies based on improved MACD are reported for Japanese Nikkei 225 Futures Market. It is shown that the selections of MACD parameters are relevant to the validity and sensitivity of MACD impacting to the investment returns, where higher returns are derived from the GA optimised MACD indicator [57,58]. A combination of two technical indicators: MACD histogram (MACDH) and RSI whose parameters are optimised by GA is reported for portfolio rebalancing [37]. It shows the good potential of the combination of GA optimisation with multiple technical indicators for trend reversal forecasting in financial markets.

This paper will be using the MACD indicator, which is designed to reveal changes in the strength, momentum and direction of a trend in price. The MACD indicator has been shown to be effective in identifying shifts in trend reversals [63–65]. The downside of the MACD indicator is that it is a lagging indicator derived from historical price data. The lagging MACD indicator could be enhanced if we use future predicted prices to replace a part of the historical price data in the computations.

A problem would occur in algorithmic trading if there are overly frequent entrances and exits within the market caused by whipsaw effects [66]. This would not be optimal if the net profits do not exceed commission costs. A learning and processing framework is proposed in this paper, which implements a custom percentage oscillator to set a threshold before a buy or sell action can occur. In order to decide and optimise the different window sizes for MACD and thresholds for the custom percentage oscillator, an evolutionary optimisation technique, GA, is utilised in the proposed framework.

There are many ways to diversify a portfolio. The proposed learning and processing framework handles portfolio rebalancing and focuses on managing financial assets consisting of indexes from different international markets. Thus, market trends would depend on the economy of different countries.

RL is able to solve dynamic optimisation problems, such as portfolio allocation, in a model-free way, alleviating the assumptions often required for other approaches [67]. For example, requiring the assumption that the asset returns are normally distributed or that the utility function is quadratic. This will not usually be the case in a realistic financial market. RL will be used for the portfolio rebalancing in the proposed framework.

In the proposed learning and processing framework, the IFDNN architecture is employed to detect trends reversals by predicting look-ahead prices. The IFDNN is able to provide interpretability over black-box machine learning algorithms, such as vanilla deep neural networks. The interpretability is important for fields like financial trading, as traders and investors need to know why the deep neural network arrives at a prediction, which can be endowed by the fuzzy logic integrated within IFDNN. This paper will use 15 forecasted look-ahead values derived from the IFDNN to detect trend reversals in closing price. It will be integrated into the traditional MACD indicator, to develop an enhanced MACD indicator named forecasted MACD (f-MACD) indicator. The f-MACD indicator aims to reduce the time lag and push it closer to hindsight MACD for a more accurate strategy.

The main contributions of this paper are as follows:

1. An interpretable deep structure, IFDNN is created, using fuzzy logic for concept generation and Hebbian Learning to tag quantifiable pseudo-weights to rule nodes, in parallel with a deep neural network. IFDNN is robust in learning data drifts and shifts.
2. A learning and processing framework is proposed. It uses the IFDNN to predict the look-ahead prices up to 15 days, which is utilised by the forecasted MACD indicator to improve the performance.
3. In the proposed learning and processing framework, the improved trading strategy using f-MACD is presented with parameters being optimised by the GA and custom price percentage oscillator.
4. Two types of RL models in the proposed framework are employed to conduct the portfolio rebalancing for financial index assets, with the results of the f-MACD and Vanilla MACD compared and analysed.

The remaining parts of the paper are organised as follows: Section 2 introduces the background of relevant theoretical concepts. Section 3 describes the proposed IFDNN architecture, followed by the proposed learning and processing framework using the IFDNN for predictions. Experiments and results analysis are discussed in Section 4, where Section 4.2 presents the comparison results of three types of MACD indicators. Section 4.3 depicts experimental results of the proposed framework in trading of individual finance index assets. Section 4.4 presents experimental results of the proposed framework in portfolio rebalancing of three finance index assets, using two types of RL models. Section 5 concludes the paper and discusses the directions on our future works.

2. Background

This section provides an overview of concepts relevant to the proposed learning and processing framework, including MACD, GA and RL.

2.1. Moving average convergence divergence

MACD is calculated from moving averages of historical closing prices. It is the difference between the fast exponential moving average (EMA) and the slow EMA, which creates a first-order momentum oscillator. EMA can be computed using Eq. (1):

$$\begin{aligned} k &= \frac{2}{N+1} \\ EMA(t) &= price(t) \times k + EMA(t-1) \times (1-k) \end{aligned} \tag{1}$$

where k is the weight; t is the current time; and N is the number of days.

An MACD histogram helps account for the rate of change of the MACD. It is calculated by subtracting the EMA of MACD (signal line), from the MACD. There is now a second-ordered oscillator that measures the rate of change of momentum. Whenever there is a crossover, meaning a change from positive to negative values in the histogram or vice versa, it indicates a trend reversal. This can be used to create a buy or sell signal.

For example, if the MACD is higher than the signal line, then the histogram has a positive value, indicating a bullish buy signal. Conversely, when the MACD is below the signal line, it is a bearish sell signal as the histogram has negative values. It signifies that the momentum of the price action has reached a turning point, and thus a start of trend reversal. The typical values used for the EMA windows are 12, 26 and 9 for the fast, slow and signal respectively.

The proposed learning and processing framework uses IFDNN to firstly predict look-ahead closing prices for multiple timesteps. Subsequently, f-MACD is created, incorporating the forecasted prices to reduce the time lag present in the original MACD. This forecasted MACD will be used in trading to test its effectiveness.

2.2. Genetic algorithm

GA is an evolutionary algorithm that can be used for general optimisation [68]. It uses a global optimisation method to vary the different combination of hyperparameters in the neural network. It applies the principles of evolution found in natural selection to the problem of finding optimal solutions. GA is commonly used to generate high-quality solutions to both optimisation and search problems by using biologically inspired concepts, such as mutation, crossover and selection [69].

The proposed learning and processing framework will be employing GA for searching and determining optimised parameters for trading experiments. The population would consist of models with different combinations of parameters, the chromosome would be the list of selected values for each parameter. The fitness function would be the evaluation metrics for the model. The optimal parameters can be determined from the chromosomes of the individual with the best fitness.

2.3. Reinforcement learning

RL is an area of machine learning focusing on how intelligent agents take actions in an environment to maximise the cumulative rewards. RL tasks are defined by three main components: states, actions and rewards. States represent the environment of the task. Actions represent the allowed steps that the RL agent can take at every state, which affects the next state the RL agent would end up in. Rewards are incentives that the agent receives for performing good actions. Following this, a policy is a strategy that the RL agent uses in pursuit of its goals.

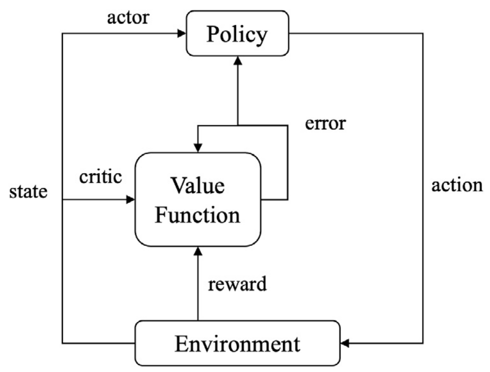

Fig. 1. Actor-Critic Network.

RL agents make decisions based on Q-values, calculated using Eq. (2):

$$Q_\pi(s,a) = E[R_1 + \gamma R_2 + \gamma R_3 + \cdots | S_0 = s, A_0 = a, \pi] \tag{2}$$

where $\gamma \in [0,1]$ is the discount factor between 0 and 1; π is the policy; and a is the value of action taken for state s.

Maximising the Q-value, $Q_\pi(s,a)$, will provide the optimal policy π. RL algorithms fall into 3 main categories: value-based, policy-based and actor-critic-based [70].

- *Value-based Algorithms.* They approximate the state action function that aims to lead the policy. It iteratively updates Q values by updating at every step. It works for discrete state and action environments.
- *Policy-based Algorithms.* They update the parameters of the policy instead of approximating the function. It uses a policy gradient for each update step, utilising a neural network to model the policy directly.
- *Actor-critic Algorithms.* They are a mix of the value and policy-based algorithms by updating two neural networks. One network is the actor network, updating the probabilities from the policy. The other is the critic network, approximating the state–action function.

RL models chosen in the proposed learning and processing framework will be from the actor-critic-based algorithm as it combines the benefits from value and policy based algorithms. In particular, the proposed framework will use Advantage Actor Critic (A2C) [71] and Deep Deterministic Policy Gradient (DDPG) [72] models. Their general network architecture can be found in Fig. 1:

2.3.1. Advantage actor critic (A2C) model

The actor network chooses an action at each timestep; and the critic network evaluates the Q-value of the input state. The actor network outputs a probability distribution corresponding to each action. The agent samples actions from this probability distribution according to each action's probability. The critic network learns which states are better than others. The actor uses this knowledge to teach the agent which good states to enter and which bad states to avoid. It outputs a single value denoting the target state.

The A2C model maintains a policy, $\pi(a_t|s_t;\theta)$, and an estimate of the value function, $V(s_t;\theta_v)$. These values are updated every t_{max} action or if there is a terminal state. The update step uses $\nabla_{\theta'} log\pi(a_t|s_t;\theta')A(s_t, a_t;\theta,\theta_v)$, where $A(s_t,a_t;\theta,\theta_v)$ is the estimated value of the advantage function of $\sum_{i=0}^{k-1}\gamma^i r_{t+i} + \gamma^k V(s_{t+k};\theta_v) - V(s_t;\theta_v)$. This function computes the agent's prediction error.

2.3.2. Deep deterministic policy gradient (DDPG) model

The network of the DDPG model is similar to the A2C RL model. But in DDPG RL model, the actor directly maps states to actions instead of outputting probability distributions across discrete action spaces. It also uses two additional target networks, Q' and μ', to add stability to training, as the target networks are tracking the predicted networks. It uses a replay buffer, R, to learn by sampling from all accumulated experiences, each sample being (s_i, a_i, r_i, s_{i+1}). For better explorations by the actor network, noisy sampled from a correlated normal distribution is added.

These two types of RL models will be used for dynamic portfolio rebalancing in the proposed framework, with the environment being different market conditions, and actions being buy, sell or hold. It aims to maximise cumulative returns.

3. Proposed learning and processing framework

This section first presents the architecture of the proposed IFDNN. It then elaborates on the proposed learning and processing framework, including the preprocessing steps taken, as well as the creation of enhanced indicators to be used during trading and portfolio rebalancing.

3.1. Interpretable Fuzzy deep neural network

IFDNN is data-driven interpretable deep neural network, where it can learn drifts and shifts by exploiting the advantage of superior learning abilities in neural networks. Since it is data-driven, it does not require prior assumptions of any statistical distributions and does not require domain expertise for defining the fuzzy partition space. It is an integration of a POPFNN [73] with a deep LSTM [74] neural network to predict financial market index prices.

The proposed IFDNN is a five-layer hybrid fuzzy neural network: Layer 1 - Input Layer, Layer 2 - Fuzzification Layer, Layer 3 - Rule Base/ Inference Layer, Layer 4 - Consequent Layer and Layer 5 - Output Layer. The architecture of the data-driven IFDNN is visualised in Fig. 2. It is similar to the architecture of the POPFNN [73], where they both have five layers, each with identical semantic meanings. However, the difference in the proposed IFDNN is the presence of a deep neural network parallel to the rule base layer. The neural network acts as the supervised learning algorithm to model the input data, while the fuzzy system tags the rules that are inferred from the neural network.

3.1.1. Five layers of IFDNN

The input data will be denoted as $X = [x_1, x_2, \ldots, x_n]$ and the outputs as $Y = [y_1, y_2, \ldots, y_m]$, where x_1 and y_1 are vectors. For this research, the type of neural networks are predominantly LSTMs, explicitly, $NN_1 = LSTM$, $NN_2 = LSTM$, $NN_3 = LSTM$, $NN_4 = MLP$. MLP is a Multi-Layer Perceptron, the simplest from of artificial neural networks. The capital letters I, C, R, N, Y, and O in the structure denote different types of nodes.

Layer 1: Input Layer. Each node I_i indicates the corresponding non-fuzzy input feature from the dataset. The input and outputs of each node of Layer 1 are shown in Eq. (3) respectively.

$$\begin{aligned} f_i^I &= x_i \\ o_i^I &= f_i^I \end{aligned} \tag{3}$$

where x_i is a vector of data points from feature I_i; and f_i is the input data at node I_i.

Layer 2: Fuzzification Layer. The nodes in this layer, $C_{i,j}$, constitute the antecedents of the fuzzy rules, R_k. To obtain the nodes, each input feature, I_i, is localised clustered using Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [75], resulting in a different

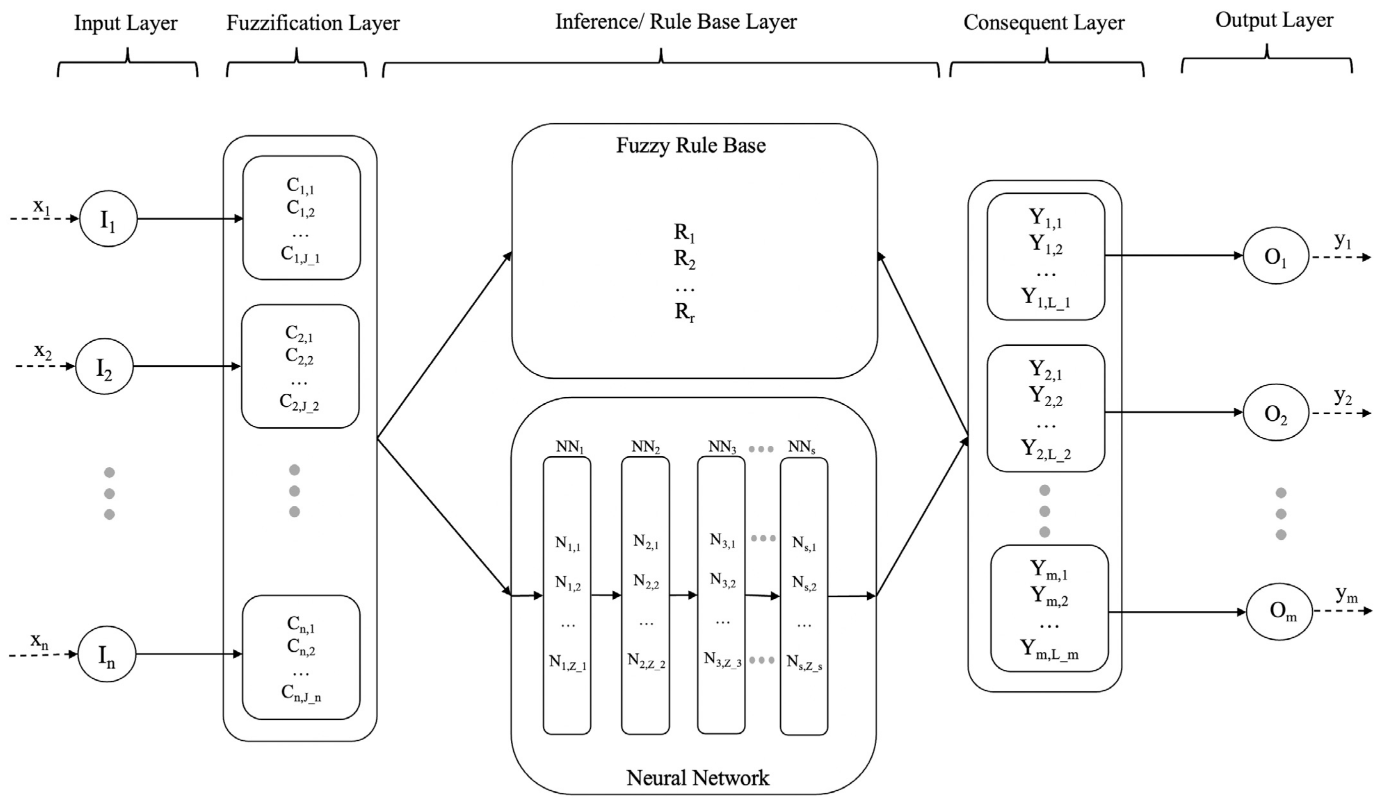

Fig. 2. IFDNN Architecture.

number of clusters for every feature. Eq. (4) will be used to generate the Gaussian memberships for each cluster during fuzzification.

$$f_{i,j}^{II} = o_i^I \qquad o_{i,j}^{II} = e^{-\frac{(f_{i,j}^{II}-\mu_{i,j}^{II})^2}{2\sigma_{i,j}^2}} \tag{4}$$

where i is the ith input feature; j is the jth cluster; $\mu_{i,j}^{II}$ is the sample mean from data in $C_{i,j}$; and $\sigma_{i,j}$ is the sample standard deviation from data in $C_{i,j}$.

The membership values across all features are concatenated together to form a single array for each data sample. This gives the overall output of Layer 2, which is $o^{II} = [o_{1,1}^{II}, o_{1,2}^{II}, \dots \dots, o_{n,J\_n}^{II}]$, where o_{n,J_n}^{II} is the last membership value obtained from the last cluster C_{n,J_n} of the nth input feature. The same logic in Eq. (4) can also be used to fuzzify the outputs, that will act as the target output features for the neural network to predict on. The output of Layer 2 will be the input to both aspects of Layer 3, the Fuzzy Rule Base and the Neural Network.

Layer 3: Rule Base/ Inference Layer. There are 2 parallel paths in this layer, the Fuzzy Rule Base and the Neural Network.

We will first discuss the Fuzzy Rule Base path. It uses a set of IF-THEN Mamdani fuzzy rules induced from the antecedent (Layer 2) and consequent (Layer 4) layers, given in Eq. (5).

$$\begin{aligned}&\textbf{IF} x_1 \text{ is } C_{1,a}^{(k)} \text{ AND } x_2 \text{ is } C_{2,b}^{(k)} \text{ AND } \dots \text{ AND } x_n \text{ is } C_{n,z}^{(k)}\\ &\textbf{THEN} y_1 \text{ is } Y_{1,p}^{(k)} \text{ AND } \dots \text{ AND } y_m \text{ is } Y_{m,q}^{(k)}\end{aligned} \tag{5}$$

where:

k is the kth fuzzy rule;

a, b, z are the winning clusters for input features 1, 2 and n respectively;

p and q are the winning clusters for output features 1 and m respectively;

$C_{i,j}^{(k)}$ is the antecedent, $C_{i,j}$, connected to Rule R_k;

$Y_{i,j}^{(k)}$ is the consequent, $Y_{i,j}$, connected to Rule R_k.

Individual rules generated by the data are combined to create the fuzzy rule base of IFDNN. The input of Layer 3 is shown in Eq. (6).

$$f^{III} = [o_{1,1}^{II}, o_{1,2}^{II}, \dots \dots, o_{n,J\_n}^{II}] \tag{6}$$

where J_n is the number of fuzzy clusters from I_n.

There is no output produced from the Fuzzy path of Layer 3, as the inference is done by the Neural Network path. However, there is Hebbian weight computation involved in the Fuzzy Rule Base after the consequents have been obtained, which will be elaborated in Section 3.1.3. The fuzzified clusters are used as inputs to the Neural Network path in IFDNN, where the predictions are mainly conducted by the Neural Network path. In general, the fuzzy rules are mainly to help explain and interpret how the weights are derived. The rules do not limit the workings of the Neural Network path, where the Neural Network performs normally with the inputs derived from the cluster membership functions, instead of absolute values of the delta changes.

Next, we will discuss the Neural Network path. f^{III} is passed into the first layer of the deep neural network, which in this case is an LSTM. This is followed by more LSTM layers and finally an MLP layer. The number of neurons in the input LSTM layer is the number of input fuzzy clusters, $Z\_1 = J\_1 + J\_2 + \cdots + J\_n$. The number of neurons in the output MLP layer is the number of output fuzzy clusters, $Z\_s = L\_1 + L\_2 + \cdots + L\_n$. This is because the neural network aims to act as the implication for the fuzzy membership values of the consequents, thus they share the same input and output vocabulary. The net input and outputs for the Neural Network path are given by Eq. (7).

$$f^{NN_1} = f^{III} = [o_{1,1}^{II}, o_{1,2}^{II}, \dots \dots, o_{n,J\_n}^{II}] \qquad o^{NN_s} = [\hat{y}_{s,1}, \hat{y}_{s,2}, \dots \dots, \hat{y}_{s,Z\_s}] \tag{7}$$

where $Z\_s = L\_1 + L\_2 + \cdots + L\_m$; and $\hat{y}_{s,i}$ is the predicted membership values from the neural network at $N_{s,i}$. o^{NN_s} makes up the memberships of consequents that the fuzzy rules are created from. This connects the Fuzzy Rule Base and the implications performed by the data-driven deep neural network.

To be able to perform tagging later, we will be using the activation values from each NN_i along with Hebbian Learning applied to the fuzzy rules (elaborated in Section 3.1.2).

Layer 4: Consequent Layer. It retrieves the inferred consequents from Layer 3 Neural Network path. The nodes $Y_{i,j}$ constitute the consequents of the fuzzy rules, R_k. The inputs and outputs of nodes in Layer 4 are in Eq. (8).

$$\begin{aligned} f_{i,j}^{IV} &= o_{s,k}^{NN_s} \\ o_{i,j}^{IV} &= f_{i,j}^{IV} \end{aligned} \tag{8}$$

where $o_{s,k}^{NN_s}$ is the output value of the kth node from NN_s. The overall output of Layer 4 is $o^{IV} = [o_{1,1}^{IV}, o_{1,2}^{IV}, \dots \dots, o_{m,L\_m}^{IV}]$, where o_{m,L_m}^{IV} is the last membership value obtained from the last cluster, Y_{m,L_m}, from the mth output feature.

Layer 5: Output Layer. This layer performs defuzzification on the aggregated areas formed from Layer 4 to get a crisp output. It is computed using centre-of-area defuzzification, following Eq. (9).

$$O_i = \frac{\sum_{j=1}^{m} o_{i,j}^{IV} * \mu_{i,j} * \sigma_{i,j}}{\sum_{j=1}^{m} o_{i,j}^{IV} * \sigma_{i,j}} \tag{9}$$

where O_i is the final crisp value at the ith output feature; $o_{i,j}^{IV}$ is the membership value at $Y_{i,j}$; $\mu_{i,j}$ is the centroid at $Y_{i,j}$; and $\sigma_{i,j}$ is the standard deviation at $Y_{i,j}$. The inputs and outputs of nodes in Layer 5 are in Eq. (10).

$$\begin{aligned} f_i^{V} &= o_i^{IV} \\ o_i^{V} &= O_i \end{aligned} \tag{10}$$

3.1.2. Tagging mechanism

We can now tag the corresponding nodes between the Fuzzy Rule Base and the Neural Network. The purpose is to understand the significance of the structure and how different inputs and outputs contribute to the inference process.

From the Fuzzy aspect, to determine which antecedent nodes to tag, the winning clusters from the input features can be determined from getting the clusters, $C_{i,j}$, with maximum membership. On the other hand, for the consequents nodes, the winning output clusters, $Y_{i,j}$, can be determined from the outputs of the neural network.

From the Neural Network aspect, to determine which neuron to tag in the deep structure, we will be looking at the activation values, represented by Eq. (11).

$$a^{(NN_i)} = [a_{i,1}^{(NN_i)}, a_{i,2}^{(NN_i)}, \dots, a_{i,Z\_i}^{(NN_i)}] \tag{11}$$

where NN_i is the ith layer in the Neural Network; and $a_{i,j}^{(NN_i)}$ is the activation value from Layer NN_i at Node $N_{i,j}$. A neuron, $N_{i,j}$, will be considered "activated" following Eq. (12):

$$activated = \begin{cases} 1, & \text{if } \left(\frac{\sum_{d=1}^{D} a_d^{(NN_i)}}{D} \right)_j \geq threshold \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

where *threshold* is the specified activation threshold; D is the number of data samples that correspond to selected Rule R_k; and $a_d^{(NN_i)}$ is the vector of activation values at Layer NN_i for the dth data sample.

If $D < 3$, then the top 3 neurons with highest activation values are activated. This tagging process is repeated for each layer of the neural network. Altogether, we now have the activated neurons from the neural network and the rules from the fuzzy rule base, allowing us to create mappings from input to outputs for both paths.

3.1.3. Pseudo-weights of Fuzzy rules

This subsection explains how the pseudo-weight computation of rules is derived using Hebbian Learning. This is necessary because although we have the antecedent, consequents and fuzzy rules, we do not know their strengths.

The algorithm is adapted from POPFNN [20]. Firstly, for each data sample, the winning cluster for both input and output is determined. The winning cluster is simply the cluster that has the highest value of membership for each feature. The winning clusters for an input sample data are denoted as $C = \{C_{1,a}, C_{2,b}, \dots, C_{n,z}\}$, where $C_{i,j}$ is the winning cluster j for input feature i. The winning cluster for the output is denoted by D, since we only have 1 output feature for this project.

Each rule is denoted by $\{C \to D\}$, where set C is the antecedent for the rule; and D is the consequent. The firing strength of the rule, f_{rule}, or pseudo-weight, can be calculated using Eq. (13).

$$\begin{aligned} f_{fw} &= min_i(\mu_{C_i}(x_i)), \\ f_{bw} &= \mu_D(y), \\ f_{rule} &= f_{fw} * f_{bw} \end{aligned} \tag{13}$$

where $\mu_{C_i}(x_i)$ is the membership values in set C, for a particular data sample x_i; f_{fw} is the forward rule firing strength, which is the minimum of all membership values in set C; and f_{bw} is the backward rule firing strength, which is the membership value of the target data.

After each rule has been processed, it is added to the knowledge base. If there is a rule that is already in the knowledge base, i.e., has the same antecedent and consequent as $\{C \to D\}$, then the rule's pseudo-weight is increased according to Hebbian Learning. Conversely, if there is no existing rule $\{C \to D\}$, the rule is created with its corresponding weight, which is its firing strength, f_{rule}. This can be illustrated in Eq. (14).

$$\begin{aligned} &matching\ rule\ found : w_{\{C\to D\}} = w_{\{C\to D\}} + f_{rule} \\ &NO\ matching\ rule : createRule(\{C \to D\}) \Rightarrow w_{\{C\to D\}} = f_{rule} \end{aligned} \tag{14}$$

where $w_{\{C\to D\}}$ is the original pseudo-weight of the rule $\{C \to D\}$; C is the set of winning clusters of the antecedent; D is the winning cluster for the consequent; and f_{rule} is the firing strength of the data sample.

For the other rules that are not fired in the current iteration, their pseudo-weights are decreased by multiplying the weight with the dynamic forgetting factor, λ. This incorporates the memory decay to account for any cases of indefinitely increasing weights. The forgetting factor can be calculated using Eq. (15).

$$\lambda = e^{-\frac{|\frac{p-l}{5}|+1}{\sqrt{n_i * n_r}}} \tag{15}$$

where p is the index of the current iteration; l is the index of the last iteration that the matching rule was fired; n_i is the number of times that the rule was fired; and n_r is the total number of rules in the rule-base as of the current iteration.

The forgetting factor ranges between 0.9 to 0.99, ensuring that the weights are not reduced by too much after many iterations, using Eq. (16).

$$\lambda = \begin{cases} 0.9, & \text{for } \lambda \leq 0.9 \\ \lambda, & \text{for } 0.9 < \lambda < 0.99 \\ 0.99, & \text{for } \lambda \geq 0.99 \end{cases} \tag{16}$$

The significance of this architecture is that the Fuzzy Rule Base of IFDNN endows the reasoning performed by inference from the data-driven deep neural network. Additionally, Hebbian Learning is applied to quantify the pseudo-weights such that the rules can be ranked relative to one another.

3.2. Data-driven concept generation of IFDNN

This sub-section describes the steps the data goes through before being passed into IFDNN.

3.2.1. *Generation of delta changes as inputs*

The purpose of this step is to account for the data drift and shift across different time periods. Since each market index has a different range of absolute values for closing prices, it would not be appropriate to use these values directly. This is because the weights of the model would only be tuned to those specific range of values, when it should be more flexible towards the changes in prices instead. Thus, we will conduct feature engineering, by calculating the delta price changes as the inputs. Similarly, the output that we aim to predict is the delta change for closing price.

After calculating the delta price changes, these values will be locally normalised within a sliding window. This is preferable compared to global normalisation as localised captures changes with respect to recent day prices instead of across multiple years, which is less relevant.

We have empirically assigned the size of the sliding window size to 20, with a stride of 1. To construct the input vector, the difference in price between consecutive days can be calculated using Eq. (17).

$$\delta y(t) = y(t) - y(t-1) \tag{17}$$

where $y(t)$ is the closing price at time t. These values are then normalised, so that the neural network can learn the changes in trends regardless of the actual values of price.

3.2.2. *Monte Carlo evaluative selection (MCES)*

MCES is a feature selection method [76]. It is data-driven and instance based, independent on application domain, removing irrelevant and correlated features. It uses Monte Carlo, which are simulation based methods that learn from data without requiring complete knowledge of the environment, based on randomly generated simulations. It uses a weighting method to rank the features according to the mask of selected features applied. Thus, it is suitable for our data-driven approach in modelling.

Although having 20 initial features as input does not seem like a lot, in the subsequent steps when performing fuzzification, the total number of feature inputs would be equivalent to the sum of clusters from all 20 features, which could easily exceed 100, if each feature forms at least 5 clusters. It would be beneficial to use MCES to prune out the less meaningful features first. The non-positively weighted features are removed, as they represent features that are irrelevant. The rest of the features are ranked in decreasing order of relative importance.

3.2.3. *Cluster memberships*

We will be using clustering for concept generation of the fuzzy sets. DBSCAN Clustering [75] is most suitable for our problem. This is because we do not know the initial number of clusters to specify for clustering, since we are letting the data decide how the resulting fuzzy sets are formed. Furthermore, we aim to handle concept drift, thus we cannot assume that the data follows any distribution. Lastly, we want to be able to identify anomalies in price changes and have as few arbitrary parameters as possible to obtain a robust clustering result. This makes DBSCAN desirable as it requires minimal domain knowledge to determine the input parameters. DBSCAN is able to discover clusters with arbitrary shape, identify outliers or anomalies with good performance efficiency.

Localised clustering is performed, meaning that each of the features will independently put through DBSCAN, where each feature forms different number of clusters and distribution of memberships. This was chosen in preference to globalised clustering. In globalised clustering, all features are be clustered together in a n-dimensional space, providing the same number of clusters for all features. If the memberships of one feature have excessive overlaps, the fuzzy rules may become obscure or poorly defined [77]. Thus, a repair has to be done along that feature axis. Due to such scenarios, it would be better off to perform local clustering instead, clustering along each axis independently such that every feature can have a different number of clusters depending on their distributions.

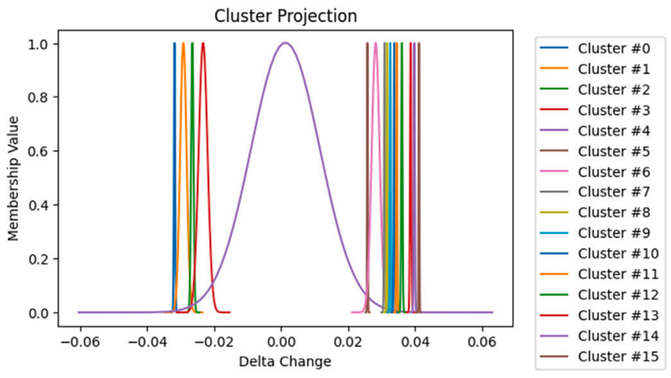

Fig. 3. Membership Projections on One Feature.

Projection of Initial Cluster Memberships. Fig. 3 shows the generated membership functions from one feature.

There are way too many clusters formed. In fact, the only distinctive Gaussian curve that can be seen is the middle one, while the rest are too narrow to be visualised. This will be resolved using a custom merge and repair function.

Merging and Repair of Cluster Memberships. This aims to reduce the number of clusters formed from DBSCAN by merging the membership functions with low variance that are relatively close to each other. This can be done by calculating the distance between the centres between every pair of memberships. Firstly, the distance between the first and last membership is calculated. Then we iterate through every neighbouring pair of membership functions to check if they are sufficiently close enough to merge, using the condition in Eq. (18).

$$\left(\frac{\tau_{right} - \tau_{left}}{2}\right) < \left(\frac{\frac{\tau_{last}}{2} - \frac{\tau_{first}}{2}}{2(m-1)}\right) \tag{18}$$

where m is the total number of membership functions belonging to the feature; τ_{right} is the centroid of the membership function on the right; τ_{left} is the centroid of the membership function on the left; τ_{last} is the centroid of the last membership function; and τ_{first} is the centroid of the first membership function.

Fig. 4 illustrates the result from the merging and repair process. The clusters are reduced to a more concise number for fuzzy sets. Furthermore, it retains the minimal overlapping property, which generates better separation of membership degrees between clusters. The threshold value for maximum clusters is set to below 12 for the experiments, which means that each feature will not have more than 12 clusters. The rationale to set this maximum value is the empirical domain knowledge dependent, to find a suitable number of clusters to balance the trade-off between complexity and variability. Excessive number of membership clusters may result in complex rules that are not ideal for creating fuzzy sets. The maximum value chosen can ensure enough variability for data values to not all be centred within too few clusters.

3.2.4. *Rule tagging and interpretability*

Currently there is no consensus on the definition of interpretability due to the black box challenges for machine learning and deep network structures [78]. In some works, the ability to comprehend and explain how a decision-making process is conducted is defined as interpretability [79]. Other works introduce the interpretability as describing a qualitative relationship between input features and output features [78,80]. The tagged network of IFDNN consists of nodes from the input to output layers. To facilitate easier understanding of the process, an example of a network graph is created for the S&P500 index dataset in Fig. 5, that shows Rule 1 from S&P500 index. It is the rule with the highest pseudo-weight, and that majority of data points align with. The visualisation corresponds to the proposed IFDNN architecture, that shows the connections of nodes from input to output

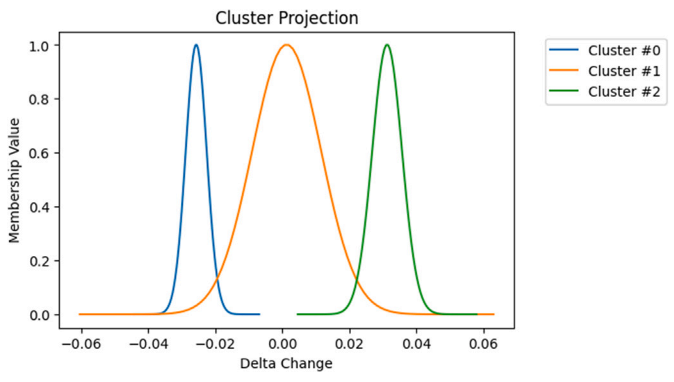

Fig. 4. Merged Memberships of One Feature.

layers, where we can use fuzzy rules to explain how the connections come about.

There are three hidden layers of the Neural Network path, where three columns of nodes in the bottom half are visualised. The first hidden layer has 128 hidden units, and the next two layers have 64 hidden units each. These respective values may differ between each model and datasets used. It illustrates why the dynamic visualisation is required to take in a different number of input, output and hidden nodes for every model. Shown in Fig. 5, the grey lines represent the connections to respective nodes among the layers. The red lines represent the antecedent and consequent of the rules, connected to their rule nodes. The nodes in the middle section are split into two halves. The top half is the fuzzy rule nodes, ordered in sequence of decreasing importance. The first rule node is the most important and commonly used rule, and the last rule node is the least important one, with lowest pseudo-weight. The bottom half nodes are the hidden nodes from the Neural Network path of IFDNN.

To understand the network graph, all the antecedents and consequent (red lines) are tagged to the first rule node. We can also easily determine which winning clusters the rules originated from. In this case, the winning clusters are from clusters C6, C6, C4, C6, C6, C4 from the six input features, respectively. A further step to understanding the data can be conducted by checking the distribution and membership values of each cluster accordingly. On the Neural Network path, four nodes from the first hidden layer are activated, followed by three activated nodes in the second and third hidden layers each. The winning output feature is the cluster C6, as shown in Fig. 5.

The same logic and analysis are applicable to the other rules, where we can see the dynamic changes in connections for every rule and node. As such, it provides good interpretability of the IFDNN network to users.

3.3. High-level pipeline of the framework

The IFDNN forecasts closing prices using the deep structure and extracts meaningful information from the input data and output predictions using fuzzy rules. Furthermore, it quantifies the relative importance of fuzzy rules by using pseudo-weights, allowing the rules to be ranked with respect to one another.

IFDNN is chosen as the learning model because of its performance as well as ability to provide an option to interpret the inference of the neural network using fuzzy rules. The IFDNN is robust in learning data drifts and shifts. The IFDNN is able to forecast closing share prices up to 15 days ahead of time, with accurate results. This could be helpful to traders or investors that need to understand the data before allowing the algorithm to make a trade. The effectiveness of the IFDNN can benefit the price forecast and portfolio rebalancing.

The time series data analysis by IFDNN is to forecast the prices. We utilise MACD as a trading indicator in our framework, as MACD helps position the time points that will be best for trading. We use the forecasted time series as the additional layer to provide more information that helps reduce the lag of MACD.

The overall proposed learning and processing framework involves data preprocessing, which includes feature engineering of delta changes of closing price, feature selection of relevant timesteps and clustering for fuzzification. Subsequently, the processed data into the IFDNN for inference, to predict look-ahead price trends which is employed to improve the performance of the traditional MACD. The improved forecasted-MACD will be used for two sets of experiments, trading of individual assets and portfolio rebalancing of a set of financial assets. GA is used to tune and optimise the trading parameters. RL is used during dynamic portfolio rebalancing. The high-level pipeline of the proposed learning and processing framework can be visualised in Fig. 6.

The delta changes of inputs are generated for the IFDNN during feature engineering, that is the second module in the proposed learning and processing framework shown in Fig. 6. As discussed in Sub- Section 3.2.1, after calculating the delta price changes, these values will be locally normalised within a sliding window. When the normalised values are passed into the IFDNN, the neural network can learn the changes in trends regardless of the actual values of price.

After the IFDNN model training and inference, a crisp value is obtained. However, this crisp value corresponds to the predicted delta changes in closing price, and not the actual absolute value of price itself. In order to obtain the predicted absolute closing price, denormalisation is performed. It is the reversal of calculations from the normalisation step, using the baseline price from the start of the sliding window and predicted delta price.

3.4. Calculation of MACDs

Since MACD is based on moving averages, it is naturally a lagging indicator. Thus, having the predicted prices at look-ahead of time aims to reduce the lag to get more accurate trend reversal detections. We will be comparing three types of MACD: the forecasted-MACD (f-MACD), vanilla-MACD (v-MACD) and perfect-MACD (p-MACD), whose differences are described as follows.

- Generated by the proposed learning and processing framework, the f-MACD incorporates the predicted look-ahead values derived from the IFDNN into the MACD.
- The v-MACD is the traditional MACD formula without incorporating IFDNN to predict the look-ahead or hindsight values.
- The p-MACD assumes that hindsight values are available. It would be the most accurate version of MACD possible under ideal cases, i.e., the “ground-truth” reference, assuming that all other model parameters are kept constant.

For the f-MACD, we will be using the Simple Moving Average (SMA) instead of the usual EMA. This is because the IFDNN is capable of predicting for up to 15 timestep values of the look-ahead prices. The SMA would have better results than EMA. f-MACD uses a modified version of SMA which is the forecasted-SMA shown in Eq. (19). It measures the simple moving average of historical prices in conjunction with forecasted prices.

$$\begin{aligned} &fSMA(window) = \frac{SMA_{d\_historical}(y) + SMA_{d\_forecasts}(\hat{y})}{2} \\ &d\_forecasts = max(\frac{window}{2}, 15) \\ &fMACD(fast, slow) = fSMA(fast) - fSMA(slow) \\ &fMACDH = fMACD(fast, slow) - EMA(fMACD) \end{aligned} \tag{19}$$

where y is the actual historical price at timestep t; $\hat{y}$ is the forecasted price at a specific look-ahead timestep; *fMACD* is the forecasted-MACD value, calculated with the *fast* and *slow* window sizes using *fSMA*; *d_forecasts* is the number of forecasted daily closing prices; *d_historical* is the number of historical daily closing prices; *fMACDH* is the forecasted-MACD histogram at timestep t.

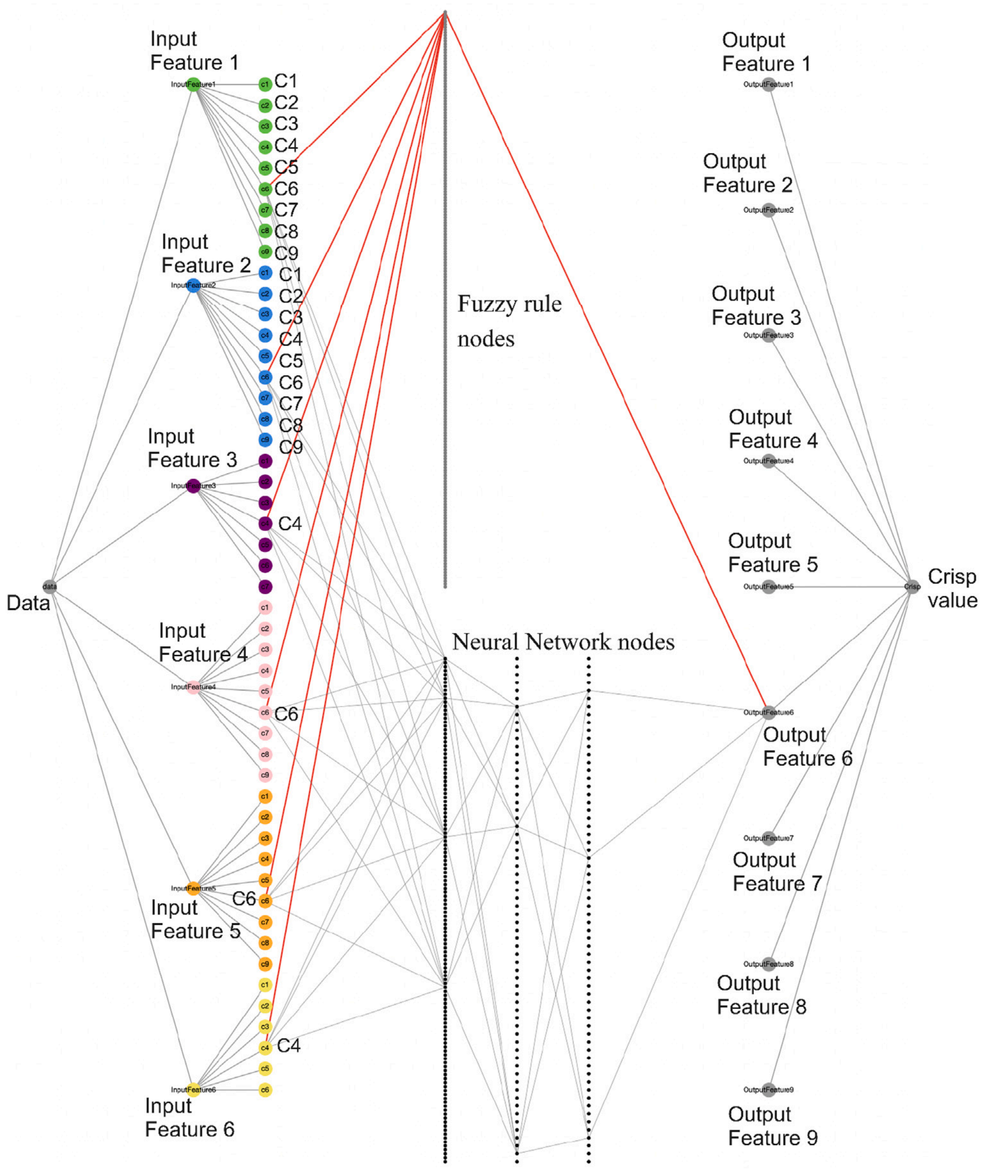

Fig. 5. Interpretability of Rule 1 of S&P500 dataset.

3.5. Accounting for whipsaw effects

The MACD strategy indicates buy and sell signals whenever there is a crossover. However, this makes it susceptible to generating very frequent signals when the MACD histogram values fluctuate frequently at 0, causing each buy and sell to be close together and gaining only marginal returns. This would not be desirable in practice because it has to take into account commission costs, which may exceed returns with too many entrances and exits in the trade.

To resolve this, a price percentage oscillator is introduced, representing the relationship of the price movement relative to the closing price, calculated using Eq. (20).

$$MACDH_{\%}(fast, slow, signal) = \left| \frac{MACDH(fast, slow, signal)}{\frac{1}{2}(fast\_SMA + slow\_SMA)} \right|$$
$$P(t) = \begin{cases} 1, & \text{if } MACDH_{\%} > \alpha \text{ and } MACDH > 0 \\ 0, & \text{if } MACDH_{\%} > \alpha \text{ and } MACDH \le 0 \\ P(t-1), & \text{otherwise} \end{cases} \tag{20}$$

where $P(t)$ is the position held at time t; $P(t) = 1$ indicates a buy position; and $P(t) = 0$ indicates a sell position. Now, the strategy is modified such that the trade signal only occurs whenever $MACDH_{\%}$ exceeds the oscillation threshold value α.

4. Experiments and results analysis

Experiments are conducted to evaluate the look-ahead prediction performance of the proposed IFDNN in Sub-Section 4.1. Next, a series of experiments are performed to evaluate and compare the performance in various configurations of the proposed learning and processing framework using the IFDNN. The result comparisons among three MACD indicators: f-MACD, v-MACD and p-MACD for each individual financial index will be presented in Sub-Section 4.2. Utilising GA for parameters tuning and optimisation, the trading performance for individual financial indexes will be introduced in Sub-Section 4.3. The experiments for portfolio rebalancing by a deep reinforcement learning framework, FinRL [81], are described in Sub-Section 4.4.

The correlations of the financial assets are relevant to the portfolio diversification and risk levels [82,83]. According to the guidelines in [83–86], the magnitudes of correlation coefficients within 0.7 to 1.0 are considered as strong correlation; within 0.5 to 0.7 considered as

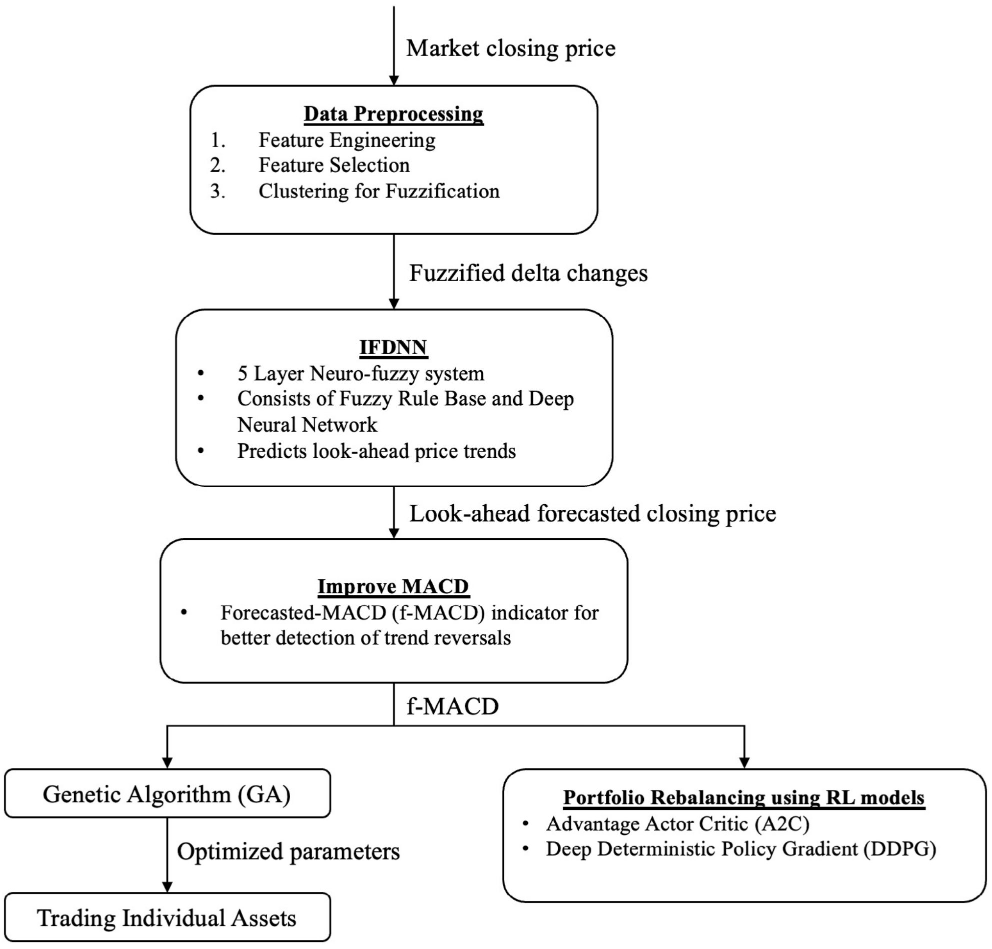
Market closing price
Data Preprocessing
1. Feature Engineering
2. Feature Selection
3. Clustering for Fuzzification
Fuzzified delta changes
IFDNN
• 5 Layer Neuro-fuzzy system
• Consists of Fuzzy Rule Base and Deep Neural Network
• Predicts look-ahead price trends
Look-ahead forecasted closing price
Improve MACD
• Forecasted-MACD (f-MACD) indicator for better detection of trend reversals
f-MACD
Genetic Algorithm (GA)
Optimized parameters
Trading Individual Assets
Portfolio Rebalancing using RL models
• Advantage Actor Critic (A2C)
• Deep Deterministic Policy Gradient (DDPG)

Fig. 6. High-level pipeline of the proposed learning and processing framework.

moderate correlation, etc. We chose the portfolio index assets covering different major markets with moderate correlations whose correlation magnitudes are commonly seen, as it is not easy to find zero correlation or exact antiphase indexes.

The Standard and Poor's (S&P500) index is a market capitalisation weighted index of 500 publicly leading trading companies in the U.S. It is regarded as one of the best gauges of American equities and stock market performance due to its wide diversity. The Financial Times Stock Exchange (FTSE100) index is a share index of the 100 blue chip companies listed on the London Stock Exchange with the highest market capitalisation. It is the most widely used UK stock market indicator. Cotation Assistée en Continu (CAC40) is the French stock market index that tracks the 40 largest French stocks based on the Euronext Paris market capitalisation, trading activity, and liquidity, etc. It is the benchmark equity index for funds investing in the French stock market.

These three indexes are chosen for experiments and evaluations with the aim of keeping in mind an international diversified portfolio, with moderate correlation coefficients shown in Table 1. This gives investors opportunities to be exposed to emerging and developed markets, thus providing diversification. For these three market indexes, the training period consist of data from 1998 to 2015, and the testing data from 2015 to 2021. An overview of the testing data is shown in Fig. 7.

4.1. Look-ahead results of IFDNN

The prediction performance of the proposed IFDNN is benchmarked against a vanilla LSTM neural network using room mean squared error (RMSE) and R^2. The equations of RMSE and R^2 are given in Eqs. (21) and (22) respectively.

Table 1
Correlation between the three indexes in the portfolio.

	S&P500	FTSE100	CAC40
S&P500	1	0.684	0.554
FTSE100		1	0.700
CAC40			1

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{21}$$

where y_i is the actual value from the dataset; $\hat{y}_i$ is the predicted value from the model; and n is the total number of data instances.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{22}$$

where $\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$.

In this work, GA is utilised for model hyperparameters tuning for both IFDNN and vanilla LSTM in the search space shown in Table 2. The preliminary study on the hyperparameters selections is measured under three particular scenarios, for multiple days look-ahead forecasting with t+5, t+10, t+15. The candidate number of hidden layers range between 2 to 4; number of nodes between 32 to 256; the activation function among ReLu, Tanh, and Sigmoid; the optimiser between Adam and RMSprop; the batch size between 64 to 256. The fitness functions are the R^2 and RMSE evaluated on the moving average window for the

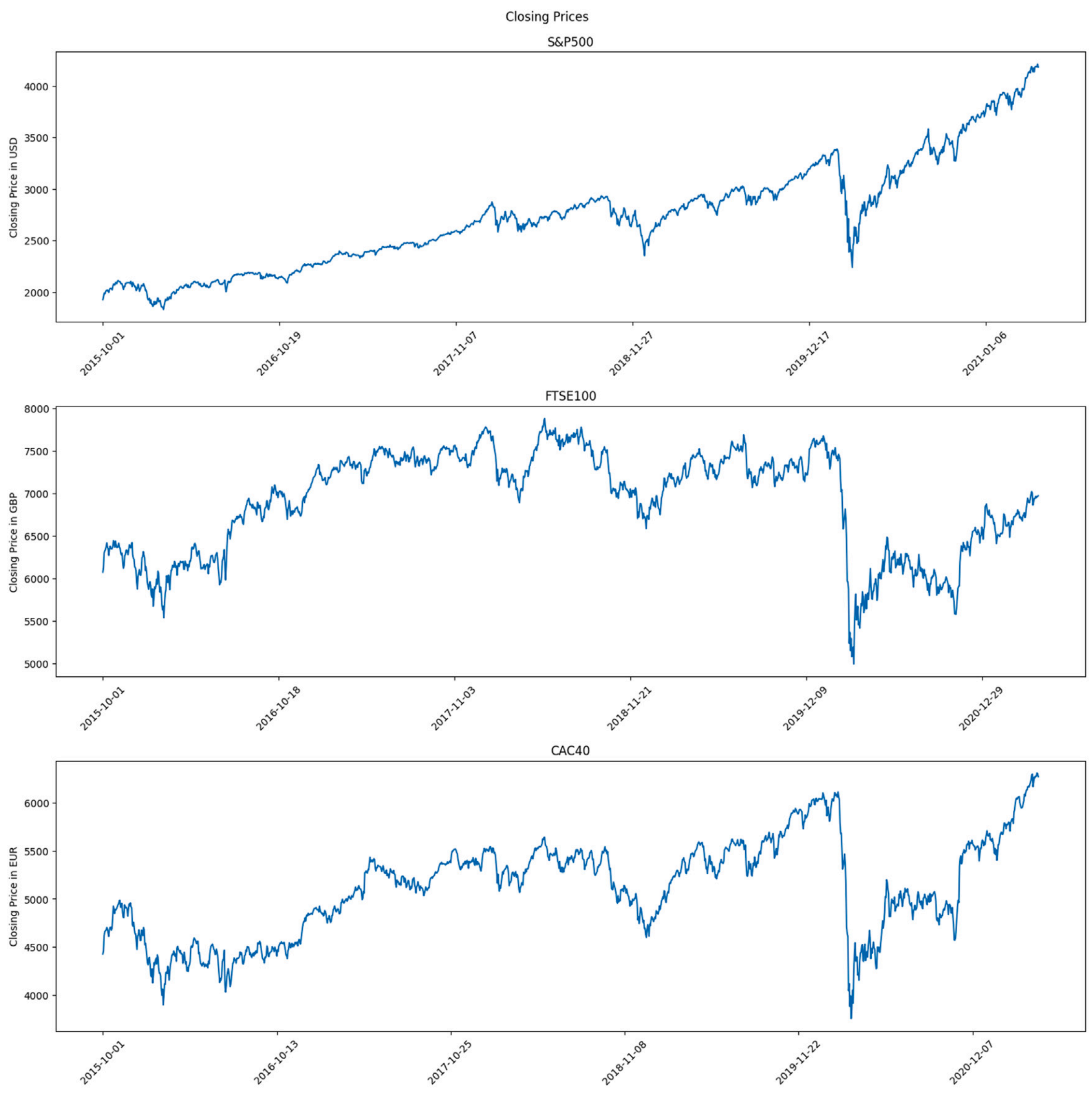

Fig. 7. Data for three financial indexes.

three experimenting market indexes: S&P500, FTSE100, and CAC40. There are 10 generations configured for the GA algorithms. It takes a significant amount of time for the GA algorithms to perform computations on the searching. Each model takes around 8 h for computing a full 10 generations. Through the evaluations and comparisons aiming to achieving the most balanced results for three scenarios and three market indexes, the hyperparameters are selected as follows: 4 hidden layers neural network, 128 nodes, the activation function as ReLU, the optimiser as Adam, and the batch size as 256. In the experiments, the epochs are set as 200. The same sets of chosen hyperparameters for both IFDNN and vanilla LSTM are shown in Table 2.

The difference between the IFDNN and vanilla LSTM is that the vanilla LSTM only contains the deep neural networks without including any components of fuzzy logic and fuzzy inference. It means the black box nature of deep architecture is applicable to the vanilla LSTM with less interpretability compared to IFDNN. The vanilla LSTM takes in all original features, without going through MCES feature selection or fuzzification through clustering. The purpose of the experiments for 15-day look-ahead forecasting and comparisons between the IFDNN and vanilla LSTM is to explore the performance differences of the networks with and without fuzzy logic inference.

The vanilla LSTM has all 20 input feature timesteps, which provides more historical timestep information. The IFDNN only contains a maximum of 6 of these timestep features, in order to limit the amount of fuzzy clusters formed subsequently. Secondly, the vanilla LSTM has data that does not go through clustering. As such, the input to the LSTM model is the direct delta changes for the timesteps, not subjected to variability of clusters and parameters of DBSCAN. Lastly, the target output is directly the delta changes instead of the membership values, implying it does not require defuzzification. Thus, the combination of these elements would cause the vanilla LSTM to outperform the IFDNN in terms of R^2 and RMSE. Nevertheless, we are trying to ensure that IFDNN does not lose too much on capabilities of price changes tracking for 15-day look-ahead forecasting, in the aim to achieving better interpretability brought in by the human understandable rules and connections than that of vanilla LSTM.

The experimental results for predicting 15 look-ahead values are shown in Table 3. It is observed that the vanilla LSTM obtains better results in terms of the R^2 and RMSE. It generates moving averages that are closer to hindsight values. Despite good performance achieved by the vanilla LSTM, it still encounters the challenges of the black-box nature. We do not easily understand what each timestep feature

Table 2
Neural network hyperparameters.

Hyperparameters	Candidate values for GA search	Selected hyperparameters
Number of hidden layers	2, 3, 4	4
Number of nodes	16, 32, 64, 128, 256	128
Activation function	ReLu, Tanh, Sigmoid	ReLu
Optimiser	Adam, RMSprop	Adam
Batch size	64, 128, 256, 512	256

Table 3
Look-ahead results.

Model	S&P500				FTSE100				CAC40			
	R^2		RMSE		R^2		RMSE		R^2		RMSE	
	IFDNN	Vanilla	IFDNN	Vanilla	IFDNN	Vanilla	IFDNN	Vanilla	IFDNN	Vanilla	IFDNN	Vanilla
t+1	0.9909	0.9988	68.33	23.86	0.9202	0.9908	159.64	53.99	0.9511	0.9944	149.69	50.67
t+2	0.9895	0.9981	73.28	30.97	0.9147	0.9818	165.14	76.14	0.9365	0.9894	170.70	69.50
t+3	0.9897	0.9970	72.43	38.73	0.9058	0.9749	173.54	89.59	0.9407	0.9908	165.00	82.73
t+4	0.9896	0.9966	73.08	41.75	0.8960	0.9700	182.40	97.88	0.9474	0.9807	155.41	94.06
t+5	0.9876	0.9959	79.64	45.85	0.9016	0.9647	177.50	106.19	0.9374	0.9765	169.54	103.67
t+6	0.9870	0.9948	81.49	51.43	0.8880	0.9573	189.38	116.86	0.9409	0.9744	164.65	108.24
t+7	0.9867	0.9945	82.35	53.07	0.8902	0.9549	187.59	120.22	0.9307	0.9704	178.34	116.52
t+8	0.9854	0.9938	86.48	56.21	0.8658	0.9490	207.42	127.78	0.9301	0.9667	179.16	123.53
t+9	0.9858	0.9932	85.17	58.93	0.8768	0.9445	198.80	133.43	0.9287	0.9624	180.82	131.368
t+10	0.9845	0.9925	89.16	61.95	0.8674	0.9373	206.31	141.82	0.9230	0.9590	188.01	137.12
t+11	0.9832	0.9914	92.81	66.08	0.8546	0.9295	216.11	150.40	0.9175	0.9552	194.63	143.32
t+12	0.9834	0.9907	92.14	68.95	0.8535	0.9236	216.97	156.59	0.9233	0.9519	187.57	148.49
t+13	0.9837	0.9894	91.30	73.38	0.8415	0.9174	225.68	162.96	0.9122	0.9505	200.70	150.67
t+14	0.9826	0.9889	94.30	75.16	0.8407	0.9132	226.33	167.08	0.9149	0.9432	197.58	161.36
t+15	0.9832	0.9891	92.76	74.50	0.8369	0.9099	229.03	170.17	0.9027	0.9434	211.36	161.19

represents, which is more significant, and what the model does with it once the data enters the neural network. While IFDNN has a better balance of interpretability and accuracy due to the integration of the fuzzy logic and neural networks.

Observed from the IFDNN results, the R^2 values range from 0.98 to 0.99 for S&P500 (∼0.01), 0.83 to 0.92 for FTSE100 (∼0.09); and 0.9 to 0.95 (∼0.05) for CAC40. For the vanilla LSTM results, the R^2 values range from 0.98 to 0.99 for S&P500 (∼0.01), 0.90 to 0.99 for FTSE100 (∼0.09), and 0.94 to 0.99 (∼0.05) for CAC40. Both the networks have the same range of values for R^2. For IFDNN, despite having less features initially and feature engineered fuzzified inputs and outputs, the results are not degrading at an increasing rate. In fact, they stay consistent with the vanilla network, just that it is less accurate by 4%–7%.

4.2. Experiment results comparing with three MACD indicators

The purpose of these comparisons is to investigate whether the f-MACD, which utilises predictions from IFDNN in the proposed learning and processing framework, is closer to predicting a perfect look-ahead MACD compared to the v-MACD. Thus, both the f-MACD and v-MACD are plotted against the p-MACD. The fast, slow and signal window size are kept to the default values of 12, 26 and 9.

Observed in Fig. 8 for trading the S&P500 index, the f-MACD does perform better than the v-MACD when benchmarking with the p-MACD. This is because it is found to have a higher R^2 and lower RMSE values. The plots and result comparisons for the FTSE100 and CAC40 indexes can be found in Figs. 9 and 10. The observations and findings are consistent with that of the S&P500 index, that the f-MACD outperforms v-MACD for three indexes.

To summarise the values of R^2 and RMSE, the experiment results for all three datasets benchmarking f-MACD and v-MACD are shown in Table 4. The forecasted look-ahead results from IFDNN resulted in significant improvements reflected in the f-MACD indicator. It is observed from Table 4 that the R^2 values derived by the f-MACD indicator are improved by 17.00% to 23.81%; while the RMSE values derived by the f-MACD indicator are reduced by 34.73% to 38.09%. This is attributed to the predictions from IFDNN in the proposed learning and processing framework. Its accuracy and ranges of predicted look-aheads allow trend reversal detection to be identified more accurately. It enables us to use the f-MACD for trading.

Table 4
Performance comparisons of v-MACD and f-MACD for individual index.

	RMSE			R^2		
	v-MACD	f-MACD	Improvement	v-MACD	f-MACD	Improvement
S&P500	29.446	18.23	38.09%	0.5564	0.6510	17.00%
FTSE100	67.38	43.98	34.73%	0.5286	0.6501	22.99%
CAC40	61.94	39.07	36.92%	0.5337	0.6608	23.81%

Table 5
GA forecasted-MACD search space.

Parameter	Values
fast	2 <fast <30
slow	10 <slow <50
signal	2 <signal <30
α	α <0.05

Table 6
Parameters optimised by GA for the f-MACD indicator.

	fast	slow	signal	α
S&P500	18	24	7	0.0024
FTSE100	24	41	5	0.0035
CAC40	25	33	13	0.0050

4.3. GA for trading parameters and results optimisation

The value of the oscillator threshold, α, cannot be determined empirically as it is data dependent. Thus, we will be using GA to decide its value. This can be coupled with other tunable parameters such as fast, slow and signal window sizes. The maximum number of stall generations used will be 10. GA will stop running when there is no further improvement to the results in the best performing population. 500 data points from the data will be used to train the GA and the rest for testing. The parameters for the trading GA search space are described in Table 5.

The results derived from GA optimisation are in Table 6. The GA generated parameters will be utilised for the experiments to trade three individual indexes. The commission fees per transaction will be 1.25%.

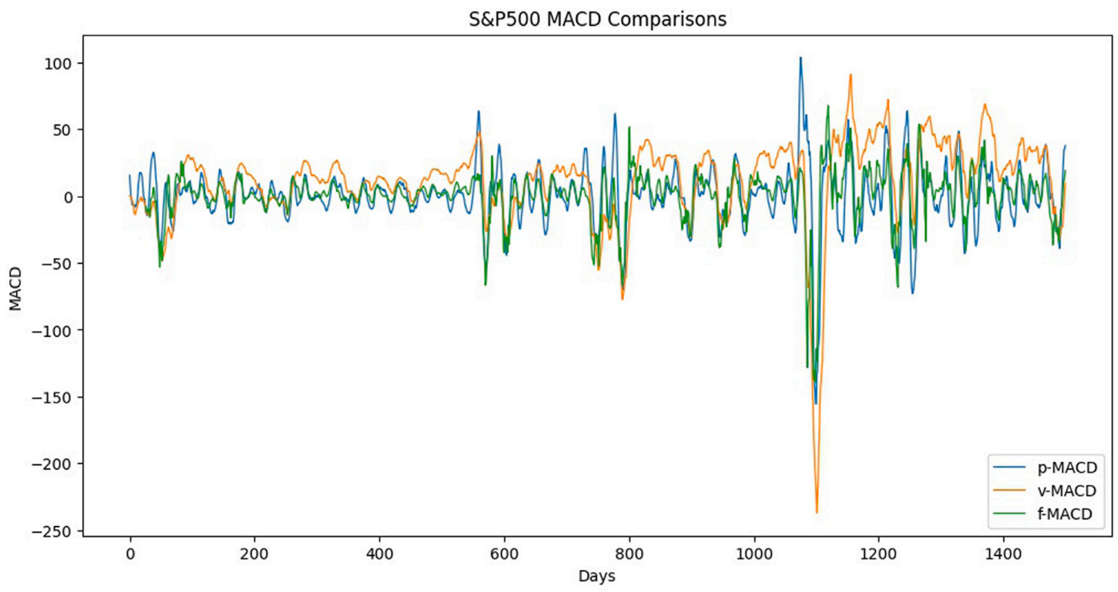

Fig. 8. Results Comparisons of 3 MACD Indicators for the S&P500 Index.

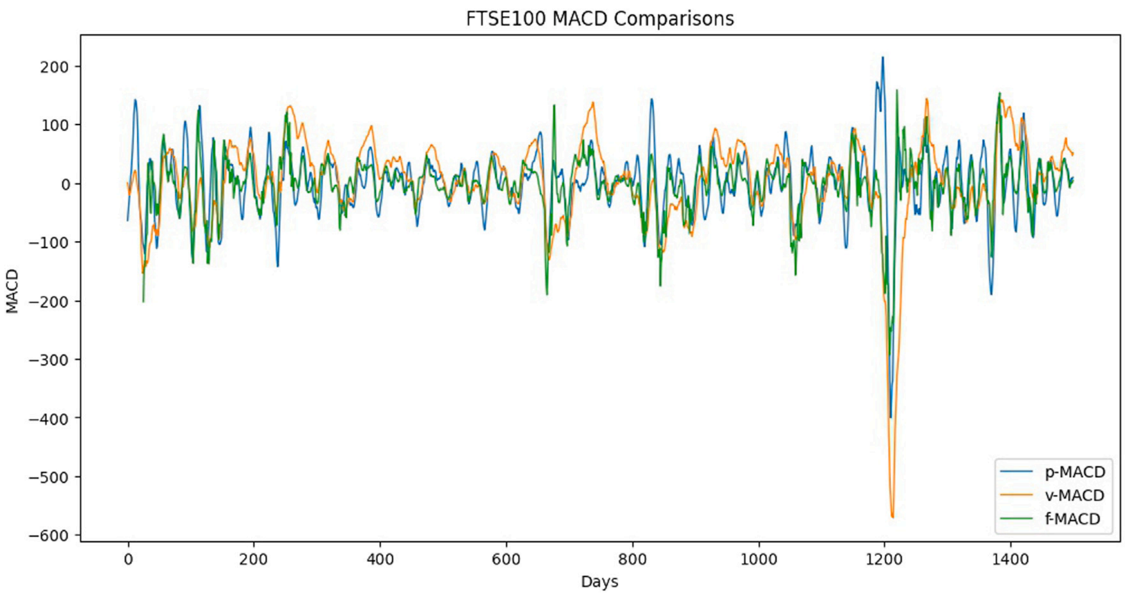

Fig. 9. Results Comparisons of 3 MACD Indicators for the FTSE100 Index.

This will be integrated within the f-MACD strategy in the proposed learning and processing framework. Fig. 11 shows the trading for the S&P500 index using the IFDNN based f-MACD with the respective GA optimised parameters.

The green upward arrows denote buy signals and the red downward arrows denote sell signals. The gain or loss is denoted at the bottom of the graph above the x-axis, where a green bar means that the trade is profitable, and the red bar means a loss trade. The intensity value of the bar is linear to the amount of profit/loss from the trade.

From Fig. 11, the buy/sell signals are not occurring too frequently, attributed to the price percentage oscillator. Other than that, most of the potential buy signals in the bullish periods have been captured. For example, at the first two buy and sell signals, they are both bullish trends and the proposed learning and processing framework manages to enter the market when the prices are still low, and exits while the price is relatively high. It is also observed in Fig. 11 that the losses seem to occur when the buy and sell signals are close to each other.

There is a steep drop in closing prices in early 2020, due to COVID-19. The proposed learning and processing framework does not manage to exit the market early enough. The selling price is lower than it could have been if it exits earlier. However, there is still a profit gained because the sell signal happens as soon as possible after detecting a steep drop in prices. Furthermore, a buy signal appears shortly after the trend starts to become bullish again.

Fig. 12 shows the trading patterns for the FTSE100 index using f-MACD with the respective GA optimised parameters. The proposed learning and processing framework manages to capture some larger swings in price changes. However, there are much fewer trade signals in this scenario. This could be attributed to the fact that the slow SMA has a large window size (see Table 6). Thus, some of the buy signals are not captured. Regardless, it still manages to make more profits than losses by capturing a few strong swings.

Fig. 13 shows the trading patterns for the CAC40 index using forecasted-MACD with the respective GA optimised parameters. The

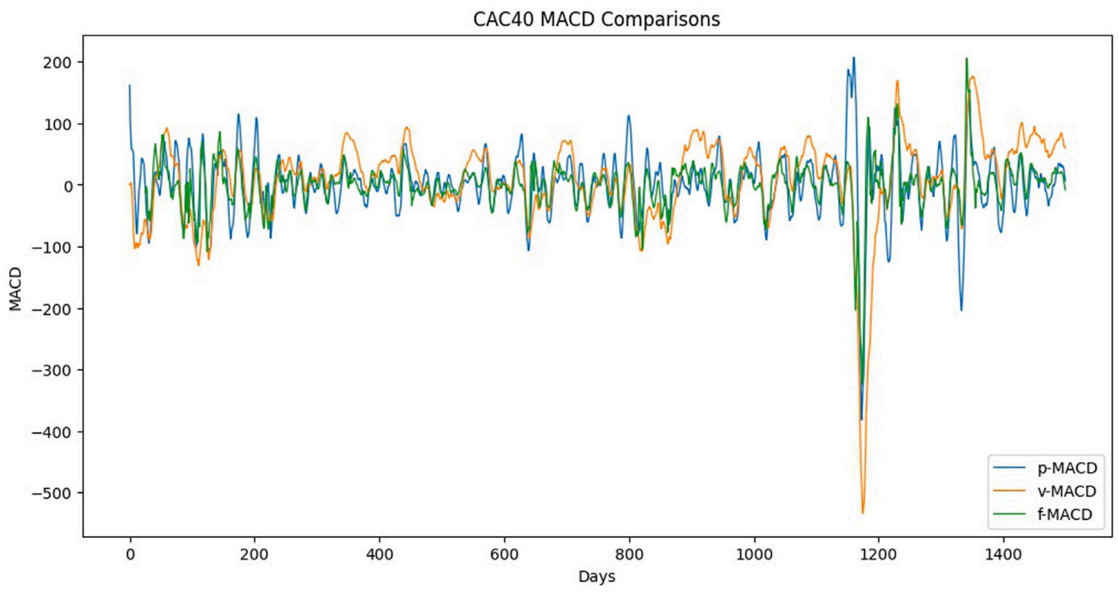

Fig. 10. Results Comparisons of 3 MACD Indicators for the CAC40 Index.

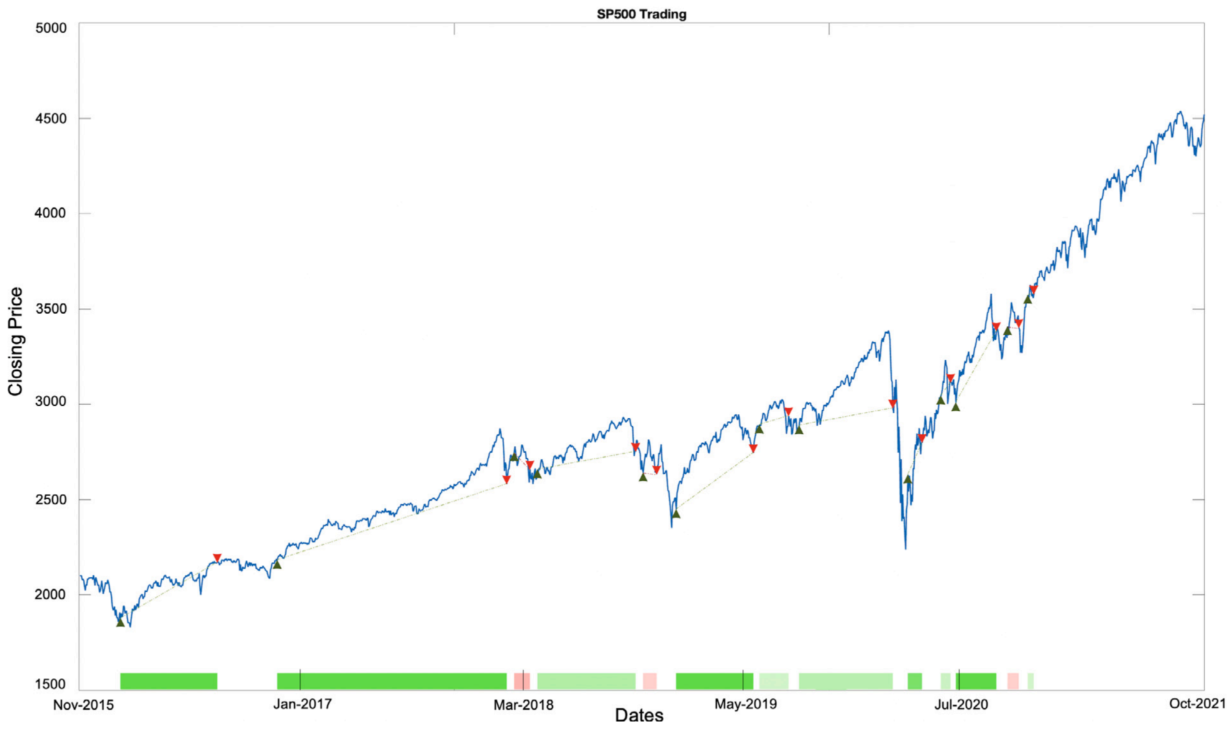

Fig. 11. Tuned Trading Results of the S&P500 Index by the Proposed Framework.

proposed learning and processing framework here performs similarly to that of FTSE100. It manages to capture even larger swings present, thus being able to generate higher profits. At the COVID-19 period, with a large drop in price, it does not capture the highest point of closing price to sell. This is because the drop is very steep and sudden, such that forecasting look-aheads would not be able to predict it due to requiring other external event-driven information. However, a profit is still gained for that period as the sell signal happens almost immediately after the detection of the drop in price. Furthermore, the next buy signal happens shortly after the prices start to climb back up.

Table 7 displays the experiment results of investment returns and maximum drawdowns of the three approaches: f-MACD, v-MACD and Raw for these three individual index assets. The f-MACD and v-MACD approaches are compared between the parameters optimised by GA and the standard MACD parameters (i.e., 26, 12, 9) shown in the same table.

The results in "Raw" denotes the trading strategy with v-MACD using the GA optimised parameters, but without incorporating the price percentage oscillator. It makes huge losses and has high maximum drawdowns for all datasets, compared to other benchmark configurations in the same table. It represents the maximum observed loss from a peak to a trough, before a new peak is attained. Maximum drawdown is an indicator of downside risk over a specified time period.

Thus, f-MACD and v-MACD include the price percentage oscillator, resulting in better trading performance. This also ensures us to make fair comparisons between f-MACD and v-MACD under two configurations: one with parameters optimised by GA, and the other with

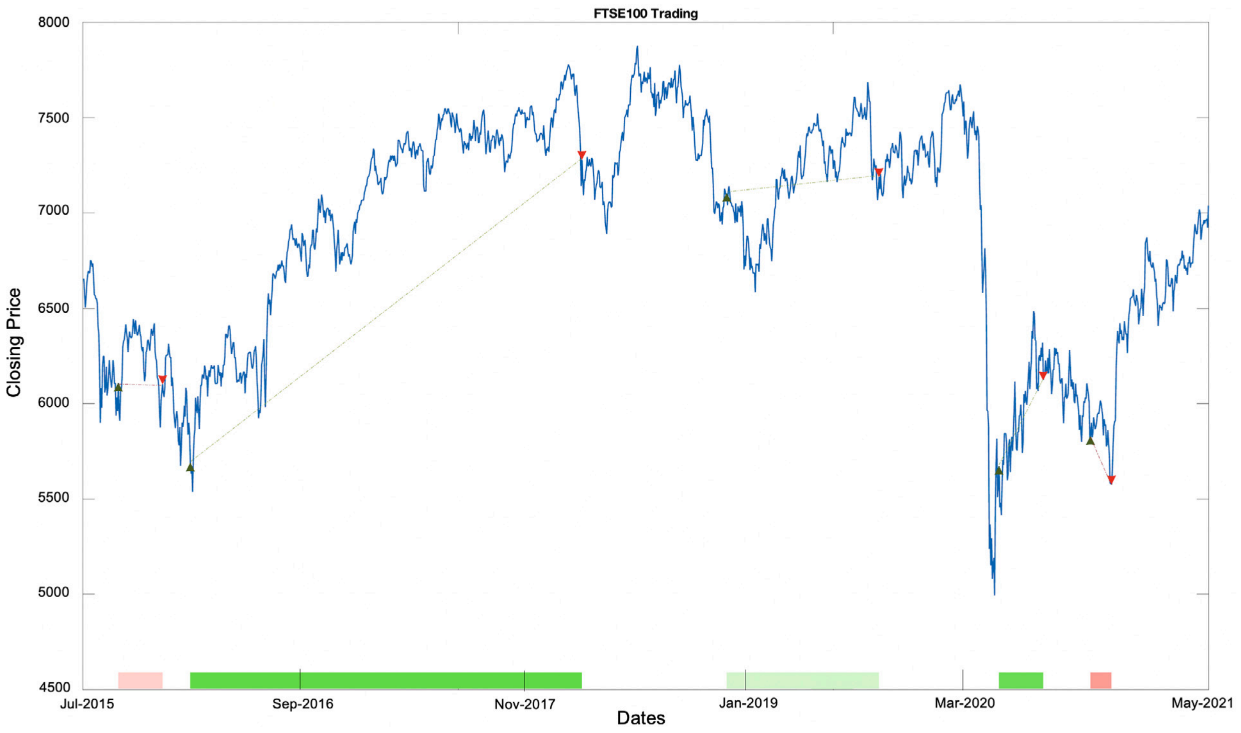

Fig. 12. Tuned Trading Results of the FTSE100 Index by the Proposed Framework.

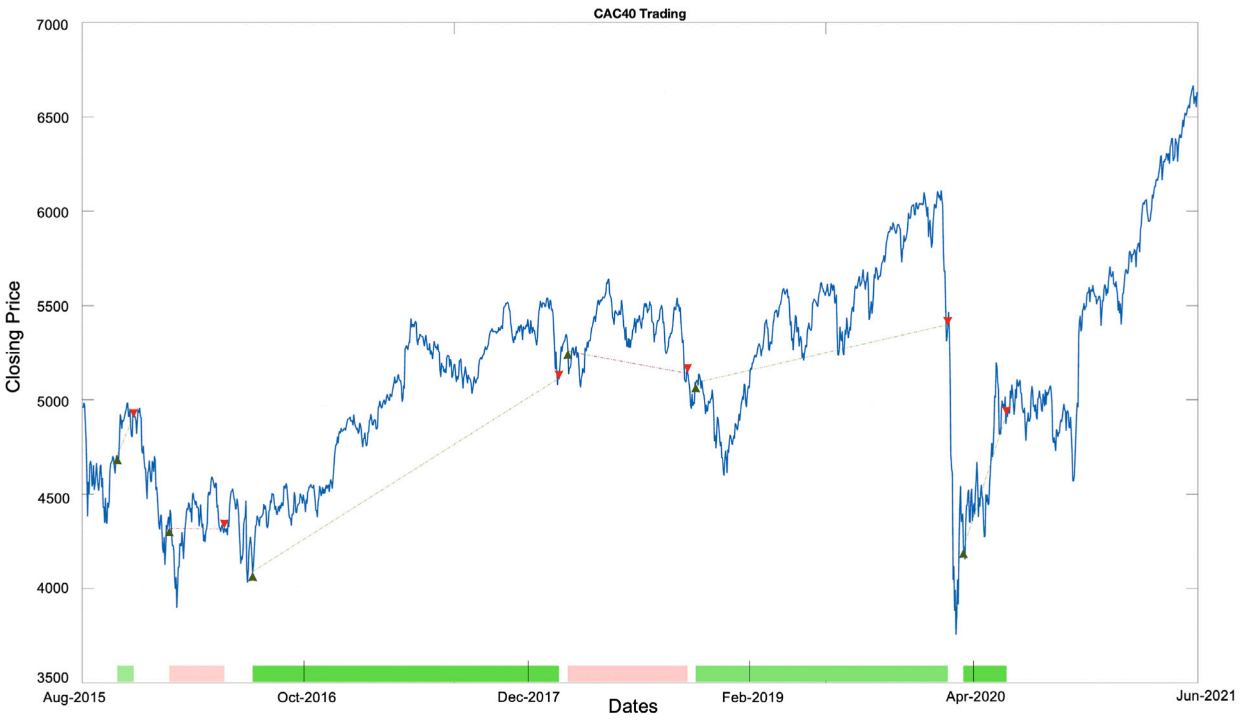

Fig. 13. Tuned Trading Results of the CAC40 Index by the Proposed Framework.

standard parameters. It is observed that the higher returns are achieved by f-MACD with the GA optimised parameters for all three indexes. The returns derived from f-MACD with the GA optimised parameters are higher than those of v-MACD with the GA optimised parameters by 1.10% to 26.90%. The f-MACD with the GA optimised parameters also outperforms the f-MACD and v-MACD with the standard parameters. The maximum drawdowns derived from v-MACD with GA optimised parameters are lower than those of f-MACD with GA optimised parameters. This could be because the v-MACD is more defencive and gets out of position more quickly when there is a price decrease.

It can also be seen in Table 7 for the effectivness of the GA optimised parameters over the standard parameters. The returns derived from f-MACD with the GA optimised parameters are higher than those of f-MACD with the standard parameters by 22.17% to 111.30% for all

Table 7
Trading results comparison.

	Returns					Max Drawdown				
	GA optimised parameters			Standard parameters		GA optimised parameters			Standard parameters	
	f-MACD	v-MACD	Raw	f-MACD	v-MACD	f-MACD	v-MACD	Raw	f-MACD	v-MACD
S&P500	**50.2%**	23.3%	−86.8%	−0.33%	−31.70%	0.195	**0.151**	0.873	0.399	0.441
FTSE100	**21.6%**	20.5%	−90.1%	−0.57%	−30.90%	0.154	**0.140**	0.902	0.617	0.408
CAC40	**67.3%**	52.1%	−88.2%	−44.00%	−11.20%	0.225	**0.121**	0.888	0.504	0.336

three indexes. The returns derived from v-MACD with the GA optimised parameters are higher than those of v-MACD with the standard parameters by 51.40% to 63.30%. The maximum drawdowns derived from f-MACD with the GA optimised parameters also are lower than those of f-MACD with the standard parameters by −0.204 to −0.463. The maximum drawdowns derived from v-MACD with the GA optimised parameters are lower than those of v-MACD with the standard parameters by −0.215 to −0.268.

With the GA optimised parameters, for the trading of the S&P500 index, the returns are more than double when using f-MACD of the proposed framework compared to v-MACD. This could be because the market was mostly bullish for majority of the time, thus the proposed learning and processing framework takes advantage of all the price increases with the forecasted values. On the other hand, v-MACD does not have the forecasting capability. Therefore, it would not have been able to fully benefit from the market as f-MACD of the proposed framework does. For the trading of the FTSE100 index, the trends are mostly fluctuating and f-MACD does not create that many buy and sell signals, thus it performs only slightly better than v-MACD because not every fluctuation is entered for trade. Furthermore, it is limited by the price percentage oscillator to not create too many signals for every single trend reversal detected. Lastly, the trading of the CAC40 index has more of an increasing trend, resulting in more profits from the large swings periods captured across the years. Having the forecasting capabilities also helps to better position buy and sell signals because a more accurate detection of trend reversal is being made.

To summarise, these results demonstrate the effectiveness of the f-MACD in the proposed learning and processing framework with the GA optimised parameters and the price percentage oscillator . The algorithm positions buy and sell signals such that the profits exceed the losses, without being too frequent such that a loss is made from commissions. It also holds up well during market crashes when prices plunge, and tries to sell the index as soon as a drop is detected, resulting in a net profit for those periods. f-MACD with the GA optimised parameters outperforms v-MACD in returns, verifying the effectiveness of the proposed learning and processing framework by generating much larger profits in certain cases. This varies according to market trends, but in general, f-MACD from the proposed learning and processing framework is robust, attributed to the accurate forecasting from IFDNN.

4.4. Portfolio rebalancing results

Portfolio rebalancing and managements are different from trading of individual index assets, as it re-balances the weights from each financial asset at every time step. This would be useful in the case where investors would like to invest multiple diversified assets concurrently for long periods of holding.

Table 8
RL attributes.

Components	Attributes
State	Open, High, Low, Close, Volume MACD Indicator
Action	Sell, Hold, Buy Portfolio Weights
Rewards	Portfolio Returns Sharpe Ratio
Environment	S&P500, FTSE100, CAC40 Foreign currency and exchange rate

We will be using a deep reinforcement learning framework, FinRL [81], to conduct the portfolio rebalancing in the proposed learning and processing framework. The three MACD indicators, f-MACD, v-MACD and p-MACD will be compared in the portfolio rebalancing tasks.

The state space for the RL agent would include financial market data, along with MACD values. The action space would be $\{-1, 0, 1\}$ to denote sell, hold and buy actions respectively. The reward function would be from the standpoint of portfolio returns and Sharpe ratios. A summary of the components for the RL attributes is shown in Table 8.

Creating the environment is crucial for the RL agent to learn well. This is because it takes in different market information provided and transfers those parameters to a Markov Decision Process problem. During the training phase, the RL agent observes the changes in price, takes an action and obtains a reward. It updates iteratively for each trading day, and the outcome would be a strategy that maximises the expected returns. The pseudocode of the DRL adapted from [81] is in Fig. 14.

The buy and hold trading strategy is chosen to act as the benchmark which is the baseline to compare with the RL strategy in the proposed learning and processing framework. This means investors only buy the index asset at the start, and sell it at the end of the trading period. This would only incur twice the commission fees. The 1/N portfolio strategy (i.e., equally weighted portfolio strategy) is another popular and effective strategy to rebalance total portfolio capital equally at each rebalancing date [87–89]. The 1/N portfolio strategies with both monthly equally rebalancing every 22 trading days and quarterly equally rebalancing every 66 trading days as the benchmarks, with the same rate of commission fees applicable in the experiments.

The metrics to evaluate trading performance include cumulative returns and Sharpe ratio. Cumulative returns are the difference between the initial value and final value, normalised by the initial value. The Sharpe ratio is the average return earned in excess of the risk-free rate per unit of volatility.

After training the RL models incorporating f-MACD data in the proposed framework, the portfolio results for the A2C RL model and DDPG RL model are shown in Fig. 15, together with other benchmarking strategies. The returns and max drawdown for all these strategies are shown in Table 9. The differences in percentage for the A2C RL strategy in our proposed framework compared over other benchmarking strategies are shown in the column of "A2C RL over other strategies". While the differences in percentage for the DDPG RL strategy in our proposed

Protocol 1 Portfolio allocation using deep reinforcement learning (DRL)

```
1: Input: $s$, state space includes covariance matrix for stocks and technical indicators
2: Output: Final portfolio value
3: Initialize $P_0 = \$1,000,000$, $\boldsymbol{w_0} = (\frac{1}{m}, ..., \frac{1}{m})$, $P_0$ is the initial portfolio value, $w_0$ is the initial
   portfolio weights, m is the number of stocks in the portfolio;
4: for $t = 1, ..., n$ do
5:     Portfolio manager DRL observes a state $s$ and outputs a portfolio weights vector $\boldsymbol{w_t}$;
6:     Normalize the weights $\boldsymbol{w_t}$ to sum to 1;
7:     Calculate stock returns vector $\boldsymbol{r_t} = (\frac{v_{1,t}-v_{1,t-1}}{v_{1,t-1}}, ..., \frac{v_{m,t}-v_{m,t-1}}{v_{m,t-1}})$, $v$ is the closing price;
8:     Portfolio incurs period return $\boldsymbol{w_t}^T \boldsymbol{r_t}$;
9:     Update portfolio value $P_t = P_{t-1} \times (1 + \boldsymbol{w_t}^T \boldsymbol{r_t})$;
10: end
```

Fig. 14. Pseudocode for Reallocation.
Source: Adapted from [81].

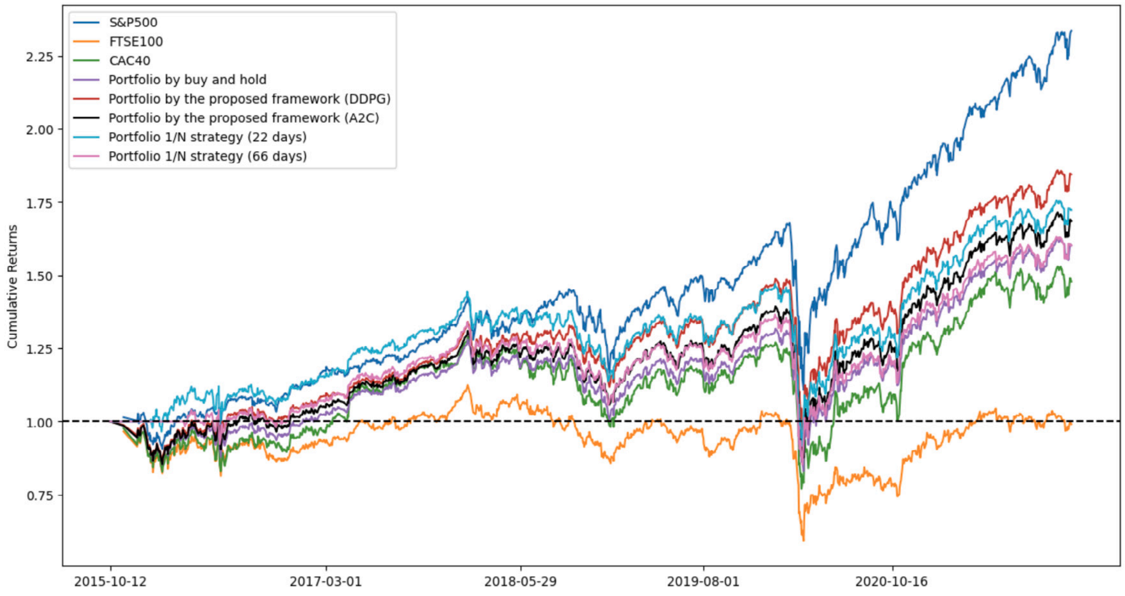

Fig. 15. Returns of benchmarking strategies with A2C RL and DDPG RL strategies of the proposed framework.

framework compared over other benchmarking strategies are shown in the column of "DDPG RL over other strategies" in the same table.

It is seen in Fig. 15 that the DDPG RL strategy in our proposed framework outperforms all other strategies, except for the strong bullish S&P500 index in the experiment time frame. This could be due to the strong bullish trend from the S&P500 index. However, because the FTSE100 and CAC40 indexes are more volatile, the RL strategy in the proposed framework manages to take advantage of their fluctuations to optimally allocate assets based on trend reversals detected from the forecasted-MACD. The plots of the A2C RL model with f-MACD indicator show lower performance than those of the S&P500 index, DDPG RL model, and 1/N strategy with monthly rebalancing.

It is also observed in Table 9 that the returns of the created portfolio managed by the A2C RL model with f-MACD indicator of the proposed framework outperforms those of both the 1/N strategy with quarterly rebalancing and the buy & hold strategies by 8.0% in returns. However, the 1/N strategy with monthly rebalancing outperforms the A2C RL model by 4.0% in returns. It probably because that the 1/N strategy takes rides on the strong bullish trend of the S&P500 index with the monthly rebalancing.

As seen in Table 9, the returns of the portfolio managed by the DDPG RL model with f-MACD indicator are higher than all other strategies except for the strong bullish S&P500 index. The similar trend on the lower max drawdown for the DDPG RL model comparing to other strategies can be observed, except for the S&P500 index. The DDPG RL model outperforms the buy and hold strategy by 24.0% in returns, outperforms the 1/N strategy with quarterly rebalancing by 24.0% in returns, and outperforms the 1/N strategy with monthly rebalancing by 12.0% in returns.

Furthermore, the DDPG RL model has higher cumulative returns of 16.0% than those of the A2C RL model. This could be due to the nature of the DDPG algorithm, where it is off-policy and the objective function does not directly depend on a policy's probability distribution. Instead, it only requires a single timestep for every update of states, actions and rewards. This differs from the A2C RL model, which has to be trained based on policies. After an update, trajectories generated from older policies would not be applicable. Thus, the DDPG RL model is able to learn better without relying on conditional probability distributions induced by the policy.

Fig. 16 displays the portfolio weights generated by the DDPG RL model. These changes in the weight allocation are caused by the fluctuations in closing price and trend reversals in each index, with the RL rebalancing strategy allocating changes in weights depending on the action that gives maximal returns. It is prominent that the weights for the FTSE100 index are generally lower, and the CAC40 index is generally higher. The S&P500 index has stayed mostly consistent, but when it does fluctuate, it would have an increased weight. This could be partly attributed to the MACD values, determining the position in time where the buy and sell signals are created. However, it is also attributed to the RL algorithm for optimisation based on changing states and actions. From a pure observatory point of view, it seems that the FTSE100 index has fewer high return opportunities compared to CAC40 and S&P500 indexes, which causes its weight to be lower, indicating that it is being sold while other indexes are being bought instead.

Table 9
Comparisons of returns and max drawdown of various portfolio strategies.

Portfolio strategy	Results		A2C RL over other strategies		DDPG RL over other strategies	
	Returns	Max Drawdown	Returns	Max Drawdown	Returns	Max Drawdown
S&P500 alone	**234.0%**	**0.339**	−66.0%	3.1%	−50.0%	2.7%
FTSE100 alone	99.1%	0.473	68.9%	10.3%	84.9%	−10.7%
CAC40 alone	148.0%	0.403	20.0%	−3.3%	36.0%	−3.7%
Buy and Hold Strategy	160.0%	0.369	8.0%	0.1%	24.0%	−0.3%
1/N Strategy (22 days)	172.0%	0.376	−4.0%	−0.6%	12.0%	−1.0%
1/N Strategy (66 days)	160.0%	0.376	8.0%	−0.6%	24.0%	−1.0%
A2C RL Model (ours)	168.0%	0.370	–	–	16.0%	−0.4%
DDPG RL Model (ours)	184.0%	0.366	−16.0%	0.4%	–	–

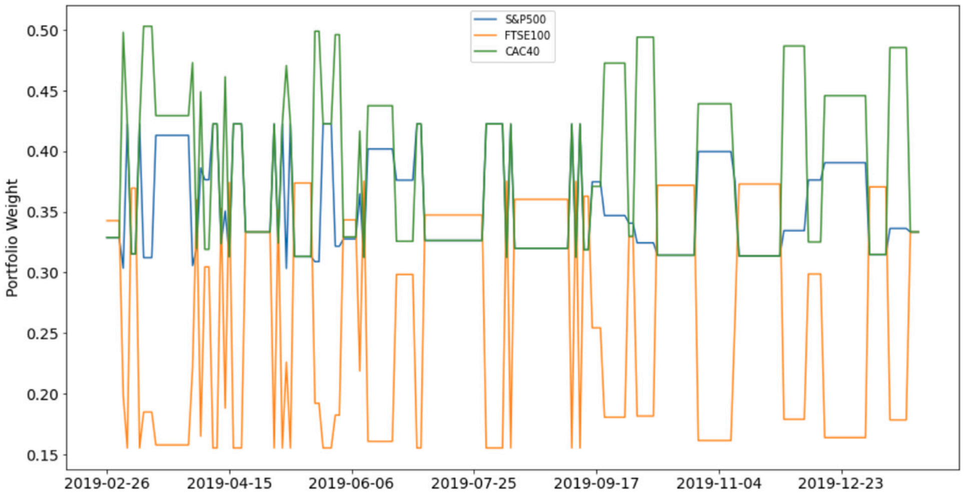

Fig. 16. Changes in portfolio weights generated by the DDPG RL model.

Table 10
Comparisons of returns and Sharpe ratio of three MACD using A2C and DDPG RL models.

	Returns		Sharpe ratio	
	A2C RL model	DDPG RL model	A2C RL model	DDPG RL model
vanilla-MACD	**70.0%**	63.3%	**0.642**	0.612
forecasted-MACD	68.5%	**84.5%**	0.634	**0.745**
perfect-MACD	65.4%	70.1%	0.611	0.655

In the next experiment, the f-MACD indicator which is utilised by the A2C and DDPG RL models in the proposed framework is replaced by the v-MACD and p-MACD. The purpose of the experiment is to compare the effectiveness and impact of the three types of the MACD indicators in the framework for the tasks of portfolio rebalancing. The returns and Sharpe rations of the portfolio rebalancing by the three types of MACD, i.e., f-MACD, v-MACD, and p-MACD employed in the A2C and DDPG RL models are shown in Table 10. It is observed in the A2C RL model that the returns of f-MACD are higher by 3.1% than those of p-MACD, but lower by 1.5% than those of v-MACD. Similarly, the Sharpe Ratio of f-MACD is higher by 3.6% than that of p-MACD and lower by 1.3% than that of v-MACD. In the DDPG RL model, f-MACD outperforms v-MACD and p-MACD on both the returns and Sharpe Ratio by 12.1% to 21.2%.

We would have expected the returns from perfect-MACD to perform the best compared to those of the f-MACD and v-MACD. However, it is observed in Table 10 that the f-MACD and v-MACD outperform the p-MACD instead. Similar trends are observed on the results of the Sharpe ratios. v-MACD performs the best when it is incorporated into the A2C RL model in terms of returns and Sharpe ratios. While f-MACD performs the best in the case of DDPG RL model for portfolio rebalancing on both returns and Sharpe ratios. This makes the comparisons between MACDs inconclusive. .

We can explain these results by aligning the MACD values to actual peaks and troughs of closing price, to see if it is exact in determining buy and sell signals. Figs. 11–13 could be used to exemplify this point. The buy and sell signals produced do not perfectly lie on the most optimal points of buy and sell actions, as it is a lagging indicator. For example, the sell signal would have been more optimal if it is at the highest point before the sharp drop in end 2019. However, this will never be possible because hindsight values are simply not available.

This implies that for the case of portfolio rebalancing, having a single technical indicator, MACD, is not sufficient as a standalone feature to train the RL model and learn the optimal actions to take. The input data should include other technical indicators to better capture market information without relying too heavily on a single one. For example, the RSI and PPO indicators could have been additions on top of MACD. This is crucial because we would need more features to help take into account the changes in all markets concurrently.

Regardless, the f-MACDs with predictions by the IFDNN in the proposed learning and processing framework have proven to produce high returns in a portfolio rebalancing environment, despite the RL model only having the MACD indicator. It manages to outperform the benchmark MACD values for the DDPG RL model, by more than 10% in returns.

5. Conclusion

In this paper, the IFDNN architecture is developed with the capability of multiple days look-ahead forecasting. The f-MACD indicator is introduced to detect trend reversals of stock prices. IFDNN is compared with the performance metrics of RMSE and R^2 to the vanilla LSTM that is with less interpretability, when they are employed to conduct 15-day look-ahead forecasting values. A learning and processing framework incorporates the IFDNN as an inferencing module to enhance the detections by the f-MACD indicator, where GA is employed for parameters optimisation to determine the best combination of parameters among multiple variables. In the experiments of trading of individual financial index assets, the f-MACD indicator achieves positive returns for all datasets, outperforming the v-MACD by 34.7%–38.1% improvement on the RMSE results and 17.0%–23.8% improvement on the R^2 results. The f-MACD indicator with IFDNN also obtains better investment returns by 1.1%–27.0% than those of v-MACD.

The improved f-MACD indicator by IFDNN and GA is incorporated into A2C and DDPG RL models for the portfolio re-balancing tasks. The portfolio consists of three indexes representing different major international markets with moderate correlations. The f-MACD based DDPG RL model achieves notably better results of investment returns and Sharpe ratio, compared to all other benchmarking portfolio strategies, including buy and hold, 1/N strategy with monthly rebalancing and 1/N strategy with quarterly rebalancing strategy.

Lastly, the f-MACD indicator is replaced by the v-MACD and p-MACD indicators in the A2C and DDPG RL models in the proposed framework, to compare the effectiveness and impact of the three types of the MACD indicators in the portfolio rebalancing tasks. The f-MACD based DDPG RL model performs better than those of v-MACD and perfect-MACD. But the f-MACD based A2C RL mode loses out to v-MACD based model marginally by −1.5% on investment returns and −1.2% on Sharpe ratio. As such, the results are less conclusive in determining the effectiveness of the proposed f-MACD in the proposed learning and processing framework. This is attributed to the lack of sufficient factors, as MACD might not be the sole best indicator for reallocation using RL. Thus, the RL agent may not have the most holistic representation of state action pairs for optimisation, leading to inconclusive results..

As one limitation of this research, currently only the MACD technical indicator is utilised for detecting the trend reversal of the stock prices. Besides MACD, there are various indicators for technical analysis on financial stock trading that are also reported to be effective in literature, such as RSI, MACDH, or Bollinger Bands, etc. Some prior works are also introduced to combine two or more technical indicators in such tasks. Our research will be further conducted to explore other types of technical indicators and include additional customised technical indicators. They will be employed for portfolio rebalancing that incorporates the forecasted predictions from the IFDNN in the proposed learning and processing framework.

For another limitation of this research, the maximum value for the number of membership clusters to derive fuzzy rules is chosen according to the empirical domain knowledge. However, in some of the time series applications, there may be highly dynamic scenarios where the empirically chosen maximum value becomes unfit. It will need a more automated method to look for the suitable maximum value. This will be one of our future works.

Other future works could include further exploring and improving the performance of the proposed learning and processing framework. Other than that, GA itself also has parameters that require tuning, including the number of generations, types of crossovers and mutation rates.

CRediT authorship contribution statement

Nicole Hui Lin Kan: Software, Implementation, Validation, Formal analysis, Data curation, Writing – original draft. **Qi Cao:** Conceptualization, Methodology, Investigation, Writing – review & editing. **Chai Quek:** Conceptualization, Methodology, Supervision, 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.

References

[1] B. Yan, M. Aasma, M. Aasma, A novel deep learning framework: Prediction and analysis of financial time series using CEEMD and LSTM, Expert Syst. Appl. 159 (2020) 113609.

[2] E. Chong, C. Han, F. Park, Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies, Expert Syst. Appl. 83 (2017) 187–205.

[3] M. Nikou, G. Mansourfar, J. Bagherzadeh, Stock price prediction using DEEP learning algorithm and its comparison with machine learning algorithms, Intell. Syst. Account., Finance Manag. 26 (2019) 164–174.

[4] Q.Y.E. Lim, Q. Cao, C. Quek, Dynamic portfolio rebalancing through reinforcement learning, Neural Comput. Appl. (2021) 1–15.

[5] X. Shen, Q. Dai, W. Ullah, An active learning-based incremental deep-broad learning algorithm for unbalanced time series prediction, Inform. Sci. (2023) 119103.

[6] G.T. Pereira, I.B.A. Santos, L.P.F. Garcia, T. Urruty, M. Visani, A.C.P.L.F. de Carvalho, Neural architecture search with interpretable meta-features and fast predictors, Inform. Sci. 649 (2023) 119642.

[7] D. Wu, A. Lisser, CCGnet: A deep learning approach to predict Nash equilibrium of chance-constrained games, Inform. Sci. 627 (2023) 20–23.

[8] T.B. Celik, O. Ican, E. Bulut, Extending machine learning prediction capabilities by explainable AI in financial time series prediction, Appl. Soft Comput. 132 (2023) 109876.

[9] K.K. Yun, S.W. Yoon, D. Won, Prediction of stock price direction using a hybrid GA-XGBoost algorithm with a three-stage feature engineering process, Expert Syst. Appl. 186 (115716) (2021).

[10] J.N. Reimann, A. Schwung, S.X. Ding, Neural logic rule layers, Inform. Sci. 596 (2022) 185–201.

[11] N. Talpur, S.J. Abdulkadir, H. Alhussian, M.H. Hasan, N. Aziz, A. Bamhdi, Deep neuro-fuzzy system application trends, challenges, and future perspectives: A systematic survey, Artif. Intell. Rev. 56 (2023) 865–913.

[12] P. Campos Souza, Fuzzy neural networks and neuro-fuzzy networks: A review the main techniques and applications used in the literature, Appl. Soft Comput. 92 (2020) 106275.

[13] M. Ferdaus, M. Pratama, S. Anavatti, M. Garratt, E. Lughofer, PAC: A novel self-adaptive neuro-fuzzy controller for micro aerial vehicles, Inform. Sci. 512 (2020) 481–505.

[14] J. Stojanović, D. Petkovic, I. Alarifi, Y. Cao, N. Denic, J. Ilic, H. Assilzadeh, S. Resic, B. Petkovic, K. A., M. MMilickovici, Application of distance learning in mathematics through adaptive neuro-fuzzy learning method, Comput. Electr. Eng. 93 (2021) 107270.

[15] M. Wang, C. Tai, Q. Zhang, Z. Yang, J. Li, K. Shen, Application of improved and optimized fuzzy neural network in classification evaluation of top coal cavability, Sci. Rep. 11 (2021) 1–12.

[16] A. Takahashi, S. Takahashi, A new interval type-2 fuzzy logic system under dynamic environment: Application to financial investment, Eng. Appl. Artif. Intell. 100 (2021) 104154.

[17] F. Fan, J. Xiong, M. Li, G. Wang, On interpretability of artificial neural networks: A survey, IEEE Trans. Radiat. Plasma Med. Sci. 5 (2021) 741–760.

[18] A. Salimi-Badr, M. Ebadzadeh, A novel self-organizing fuzzy neural network to learn and mimic habitual sequential tasks, IEEE Trans. Cybern. 52 (2022) 323–332.

[19] J. Fei, Y. Chen, L. L., Y. Fang, Fuzzy multiple hidden layer recurrent neural control of nonlinear system using terminal sliding-mode controller, IEEE Trans. Cybern. 52 (2022) 9519–9534.

[20] A. Iyer, D. Prasad, C. Quek, PIE-RSPOP: A brain-inspired pseudo-incremental ensemble rough set pseudo-outer product fuzzy neural network, Expert Syst. Appl. 95 (2018) 172–189.

[21] H. Han, X. Zhu, Y. Li, Generalizing long short-term memory network for deep learning from generic data, ACM Trans. Knowl. Discov. Data 14 (2020) 1–28.

[22] R.L. Kissell, Algorithmic Trading Methods: Applications using Advanced Statistics, Optimization, and Machine Learning Techniques, Academic Press, 2020.

[23] C. Borch, High-frequency trading, algorithmic finance and the flash crash: reflections on eventalization, Econ. Soc. 45 (3–4) (2016) 350–378.

[24] T.S. Grindsted, Algorithmic finance: Algorithmic trading across speculative time-spaces, Ann. Am. Assoc. Geogr. (2021) 1–13.

[25] A. Nan, A. Perumal, O.R. Zaiane, Sentiment and knowledge based algorithmic trading with deep reinforcement learning, 2020, arXiv preprint arXiv:2001.09403.

[26] A.C. Briza, P.C. Naval, Jr., Stock trading system based on the multi-objective particle swarm optimization of technical indicators on end-of-day market data, Appl. Soft Comput. 11 (1) (2011) 1191–1201.

[27] D. Moldovan, M. Moca, S. Nitchi, A stock trading algorithm model proposal, based on technical indicators signals, Inform. Econ. 15 (1) (2011) 183.

[28] J. Zenisek, F. Holzinger, M. Affenzeller, Machine learning based concept drift detection for predictive maintenance, Comput. Ind. Eng. 137 (2019) 106031.

[29] R.C. Cavalcante, L.L. Minku, A.L. Oliveira, Fedd: Feature extraction for explicit concept drift detection in time series, in: 2016 International Joint Conference on Neural Networks, (IJCNN), IEEE, 2016, pp. 740–747.

[30] R.C. Cavalcante, A.L. Oliveira, An approach to handle concept drift in financial time series based on extreme learning machines and explicit drift detection, in: 2015 International Joint Conference on Neural Networks, (IJCNN), IEEE, 2015, pp. 1–8.

[31] B. Silva, N. Marques, G. Panosso, Applying neural networks for concept drift detection in financial markets, in: Workshop on Ubiquitous Data Mining, 2012, p. 43.

[32] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (2) (2012).

[33] H. Cho, Y. Kim, E. Lee, D. Choi, Y. Lee, W. Rhee, Basic enhancement strategies when using bayesian optimization for hyperparameter tuning of deep neural networks, IEEE Access 8 (2020) 52588–52608.

[34] D. Maclaurin, D. Duvenaud, R. Adams, Gradient-based hyperparameter optimization through reversible learning, in: International Conference on Machine Learning, PMLR, 2015, pp. 2113–2122.

[35] M. Beniwal, A. Singh, N. Kumar, Forecasting long-term stock prices of global indices: A forward-validating genetic algorithm optimization approach for support vector regression, Appl. Soft Comput. 145 (110566) (2023).

[36] K. Syuhada, V. Tjahjono, A. Hakim, Improving value-at-risk forecast using GA-ARMA-GARCH and AI-KDE models, Appl. Soft Comput. 148 (110885) (2023).

[37] L.L.X. Yeo, Q. Cao, C. Quek, Dynamic portfolio rebalancing with lag-optimised trading indicators using SeroFAM and genetic algorithms, Expert Syst. Appl. 216 (2022) 119440.

[38] Y. Dai, Z. Qin, Multi-period uncertain portfolio optimization model with minimum transaction lots and dynamic risk preference, Appl. Soft Comput. 109 (107519) (2021).

[39] F.D. Krüger, M. Nabeel, Hyperparameter tuning using genetic algorithms: A study of genetic algorithms impact and performance for optimization of ML algorithms, 2021.

[40] P. Liashchynskyi, P. Liashchynskyi, Grid search, random search, genetic algorithm: A big comparison for NAS, 2019, arXiv preprint arXiv:1912.06059.

[41] Y. Kim, W. Ahn, K. Oh, D. Enke, An intelligent hybrid trading system for discovering trading rules for the futures market using rough sets and genetic algorithms, Appl. Soft Comput. 55 (2017) 127–140.

[42] K. Matsumura, H. Kakinoki, Portfolio strategy optimizing model for risk management utilizing evolutionary computation, Electron. Commun. Japan 97 (8) (2014) 45–62.

[43] P.K. Aithal, M. Geetha, D.U.B. Savitha, P. Menon, Real-time portfolio management system utilizing machine learning techniques, IEEE Access 11 (2023) 32595–32608.

[44] D. Wu, X. Wang, S. Wu, Construction of stock portfolios based on k-means clustering of continuous trend features, Knowl.-Based Syst. 252 (109358) (2022).

[45] Q. Yang, W.-K. Ching, T.-K. Siu, Z. Zhang, A Markov-driven portfolio execution strategy with market impact, Numer. Math.: Theory Methods Appl. (2018).

[46] A.M. Aboussalah, C. Lee, Continuous control with stacked deep dynamic recurrent reinforcement learning for portfolio optimization, Expert Syst. Appl. 140 (112891) (2020).

[47] M.S. Farahani, S.H.R. Hajiagha, Forecasting stock price using integrated artificial neural network and metaheuristic algorithms compared to time series models, Soft Comput. 25 (2021) 8483–8513.

[48] B. Alhnaity, M. Abbod, A new hybrid financial time series prediction model, Eng. Appl. Artif. Intell. 95 (103873) (2020).

[49] H.J. Park, Y. Kim, H.Y. Kim, Stock market forecasting using a multi-task approach integrating long short-term memory and the random forest framework, Appl. Soft Comput. 114 (108106) (2022).

[50] M.M. Mamoudan, A. Ostadi, N. Pourkhodabakhsh, A. Fathollahi-Fard, F. Soleimani, Hybrid neural network-based metaheuristics for prediction of financial markets: A case study on global gold market, J. Comput. Des. Eng. 10 (3) (2023) 1110–1125.

[51] A.M. Ozbayoglu, M.U. Gudelek, O.B. Sezer, Deep learning for financial applications: A survey, Appl. Soft Comput. 93 (106384) (2020).

[52] S. Park, J. Yang, Intelligent cryptocurrency trading system using integrated AdaBoost-LSTM with market turbulence knowledge, Appl. Soft Comput. 145 (110568) (2023).

[53] M.S. Islam, E. Hossain, Foreign exchange currency rate prediction using a GRU-LSTM hybrid network, Soft Comput. Lett. 3 (100009) (2021).

[54] A. Adegboye, M. Kampouridis, F. Otero, Algorithmic trading with directional changes, Artif. Intell. Rev. 56 (2023) 5619–5644.

[55] C.D. Kirkpatrick, II, J.A. Dahlquist, Technical Analysis: The Complete Resource for Financial Market Technicians, FT Press, 2010.

[56] S.B. Achelis, Technical Analysis from A to Z, McGraw Hill, New York, 2001.

[57] A.A. Aguirre, N.D. Mendez, R.A. Medina, Artificial intelligence applied to investment in variable income through the MACD (moving average convergence/divergence) indicator, J. Econom. Financ. Administr. Sci. 26 (52) (2021).

[58] J. Wang, J. Kim, Predicting stock price trend using MACD optimized by historical volatility, Math. Probl. Eng. (9280590) (2018).

[59] B. Kang, Improving MACD technical analysis by optimizing parameters and modifying trading rules: Evidence from the Japanese nikkei 225 futures market, J. Risk Financial Manag. 14 (37) (2021).

[60] K. Saetia, J. Yokrattanasak, Stock movement prediction using machine learning based on technical indicators and google trend searches in thailand, Int. J. Financial Stud. 11 (5) (2023).

[61] S. Ahmed, S. Hassan, N.R. Aljohani, R. Nawaz, FLF-LSTM: A novel prediction system using forex loss function, Appl. Soft Comput. 97 (106780) (2020).

[62] J. Ayala, M. Garcia-Torres, J.L.V. Noguera, F. Gomez-Vela, F. Divina, Technical analysis strategy optimization using a machine learning approach in stock market indices, Knowl.-Based Syst. 225 (107119) (2021).

[63] D. Vezeris, T. Kyrgos, C. Schinas, Take profit and stop loss trading strategies comparison in combination with an MACD trading system, J. Risk Financial Manag. 11 (3) (2018) 56.

[64] Y. Lei, Q. Peng, Y. Shen, Deep learning for algorithmic trading: Enhancing MACD strategy, in: Proceedings of the 2020 6th International Conference on Computing and Artificial Intelligence, 2020, pp. 51–57.

[65] R. Vaidya, Moving average convergence-divergence (MACD) trading rule: An application in nepalese stock market "NEPSE", Quant. Econ. Manag. Stud. 1 (6) (2020) 366–374.

[66] J.J. Murphy, Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications, Penguin, 1999.

[67] P.N. Kolm, G. Ritter, Modern perspectives on reinforcement learning in finance, Mod. Perspect. Reinforcement Learn. Finance (September 6, 2019). J. Mach. Learn. Finance 1 (1) (2020).

[68] A.E. Eiben, J.E. Smith, et al., Introduction to Evolutionary Computing, Vol. 53, Springer, 2003.

[69] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, Cambridge, MA, 1996.

[70] S. Bhagat, H. Banerjee, H. Ren, Deep reinforcement learning for soft robotic applications: Brief overview with impending challenges, Robotics (2018).

[71] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: International Conference on Machine Learning, PMLR, 2016, pp. 1928–1937.

[72] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, 2015, arXiv preprint arXiv:1509.02971.

[73] R.W. Zhou, C. Quek, POPFNN: A pseudo outer-product based fuzzy neural network, Neural Netw. 9 (9) (1996) 1569–1581.

[74] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[75] M. Ester, H. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, 1996, pp. 226–231.

[76] K. Quah, C. Quek, MCES: A novel Monte Carlo evaluative selection approach for objective feature selections, IEEE Trans. Neural Netw. 18 (2007) 431–448.

[77] W.L. Tung, C. Quek, GenSoFNN: A generic self-organizing fuzzy neural network, IEEE Trans. Neural Netw. 13 (5) (2002) 1075–1086.

[78] Z.C. Lipton, The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery, Queue 16 (2018) 31–57.

[79] T. Molnar, Interpretable Machine Learning – A Guide for Making Black Box Models Explainable, Independently published, 2022.

[80] G. Pereire, I. Santos, L. Garcia, T. Urruty, M. Visani, A. de Carvalho, Neural architecture search with interpretable meta-features and fast predictors, Inform. Sci. 649 (119642) (2023).

[81] X.-Y. Liu, H. Yang, J. Gao, C.D. Wang, FinRL: Deep reinforcement learning framework to automate trading in quantitative finance, 2021, arXiv preprint arXiv:2111.09395.

[82] Y. Berouaga, C. El Msiyah, J. Madkour, Portfolio optimization using minimum spanning tree model in the moroccan stock exchange market, Int. J. Financial Stud. 11 (2023) 53.

[83] J. Lau, Portfolio diversification: Correlation risk management, 2010, URL https://www.prudentinvestors.com/blog/portfolio-diversification-correlation-risk-management/.

[84] P. Schober, C. Boer, L. Schwarte, Correlation coefficients: Appropriate use and interpretation, Anesth. Analg. 126 (2018) 1763–1768.

[85] B. Ratner, The correlation coefficient: Its values range between positive 1 and negative 1, J. Target., Meas. Anal. Market. 17 (2009) 139–142.

[86] M.M. Mukaka, A guide to appropriate use of correlation coefficient in medical research, Malawi Med. J. 24 (2012) 69–71.

[87] V. DeMiguel, L. Garlappi, R. Uppal, Optimal versus naive diversification: How inefficient is the 1/N portfolio strategy? Rev. Financ. Stud. 22 (5) (2009) 1915–1953.

[88] R.E. Bernoussi, M. Rockinger, Rebalancing with transaction costs: Theory, simulations, and actual data, Financial Mark. Portfolio Manag. 37 (2023) 121–160.

[89] W. Bessler, G. Taushanov, D. Wolff, Factor investing and asset allocation strategies: A comparison of factor versus sector optimization, J. Asset Manag. 22 (2021) 488–506.