Information Sciences 663 (2024) 120276

Contents lists available at ScienceDirect

Information Sciences

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

FE-RNN: A fuzzy embedded recurrent neural network for improving interpretability of underlying neural network

James Chee Min Tan [a], Qi Cao [b,*], Chai Quek [a]

[a] School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798, Singapore
[b] School of Computing Science, University of Glasgow, UK and Singapore

ARTICLE INFO

Keywords:
Fuzzy neural networks
Deep neural networks
Fuzzy-embedded recurrent neural network
Data driven implication
Financial assets trading

ABSTRACT

Deep learning enables effective predictions. But deep structures face some challenges on human interpretability compared to conventional techniques, e.g., fuzzy inference systems. It motivates more research works to alleviate the black box nature of deep structures with performance maintained. This paper proposes a fuzzy-embedded recurrent neural network (FE-RNN) to improve interpretability of the underlying neural networks. It is a parallel deep structure comprising an RNN and a Pseudo Outer-Product based Fuzzy Neural Network (POPFNN) that share a common set of input and output linguistic concepts. The inference processes undertaken are associated by RNN using fuzzy rules in the embedded POPFNN. Fuzzy *IF-THEN* rules provide better interpretability of the inference process of the hybrid networks. It allows an effective realisation of a data driven implication using RNN in the modelling of fuzzy entailment within a fuzzy neural networks (FNN) structure. FE-RNN obtains more consistent results than other FNN in the experiment using the Mackey-Glass dataset. FE-RNN achieves about 99% correlation for forecasting prices of market indexes. Its interpretability is also discussed. FE-RNN then acts as a prediction tool in a financial trading system using forecast-assisted technical indicators optimised with Genetic Algorithms. It outperforms the benchmark trading strategies in the trading experiments.

1. Introduction

Deep learning is a type of machine learning methods that utilises multiple layers of neural networks to extract useful information from data. The information is abstracted by performing various transformations to the data that can help support the modelling or prediction of data. However, these transformations are done through many levels of mathematical computations, that pose a great challenge in critical tasks required in learning to model a complex problem. Fortunately, recent technological breakthroughs bring greater computational power at relatively low costs, making the deep learning to be a highly viable research field. As deep learning techniques advance, many complex modelling tasks can be accurately and reliably conducted.

Deep learning-based approaches are introduced for prediction tasks on time series analysis [33,34]. As the deep learning models grow in complexity with stacking of more layers, it encounters some challenges on the human interpretability on the insights how deep learning structures derive the results [30,36,43]. It is not easy to interpret the reasoning behind the models, that are in a black box manner with hidden representations and calculations in the network [32,37]. Interpretability is an important requirement for many

* Corresponding author.
E-mail addresses: jame0019@e.ntu.edu.sg (J. Chee Min Tan), qi.cao@glasgow.ac.uk (Q. Cao), ashcquek@ntu.edu.sg (C. Quek).

https://doi.org/10.1016/j.ins.2024.120276
Received 8 May 2023; Received in revised form 19 December 2023; Accepted 31 January 2024
Available online 9 February 2024

critical domains and legal compliance systems, e.g., medical domain applications and financial systems [10]. But there is yet consensus on the definitions of interpretability in machine learning, as different meanings are presented for different domains and scenarios [30]. Interpretability could refer to the ability to comprehend and explain how a model performs the decision-making process [27], or could refer to providing a qualitative understanding of the relationship between input and output features [22,30]. The eXplainable Artificial Intelligence (XAI) attracts a lot of research interests recently to help the interpretability of deep learning [10,21]. A Long Short-Term Memory (LSTM) model is introduced to identify malicious behaviours and cyberattacks for Internet of Things (IoT) networks [21], that uses four methods to extract features from the dataset classes and generate explanations defined by human. The detected features for the classes of cyberattacks are plotted into figures, but it may have some learning curves for some users to understand how to interpret the plots. A Kronecker convolutional neural network architecture is employed for classification of kidney stones in medical image processing [29], where knowledge with good interpretability can be accumulated. An architecture is presented to improve the interpretability of neural networks by splitting the networks into levels, which constitute one or several layers, to provide insights into the layers [17]. There are hierarchical representations of the input windows to help users gain better visibility of multiple layers structure in neural networks. But there may be a smaller version of black box in the LSTM layer in each level when deriving the hidden state. The *meta*-predictor is introduced to extract interpretable *meta*-features from neural architectures and regression models [30], that uses slightly different definitions of the interpretability, for understandability by users equipped with basic knowledge of neural architectures.

Some systems make use of fuzzy logic or Bayesian logic to draw inferences and make decisions. Rules of fuzzy logic are usually specified by domain experts, providing interpretable insights of the knowledge in the form of fuzzy rules [35]. By observing the firing of the rules during the use of the fuzzy system, it achieves better interpretability [24,32]. Fuzzy logic is capable of dealing with inaccurate, uncertain, or ill-defined data similar to human experts [13,31,49]. However, manually curating fuzzy rules for a complex model is deemed to be tedious and challenging. In recent years, fuzzy neural networks (FNN) or neuro-fuzzy computing (NFC) have been introduced as an alternative to traditional fuzzy systems [8,26,39].

Neural networks and fuzzy logic are integrated in FNN as a hybrid paradigm, that can mimic some aspects of human reasoning [3]. The fuzzy rules and membership functions make the neural networks interpretable or translatable [47]. The semantic transparency and interpretability of fuzzy logic can be incorporated with the learning capability of neural networks [28]. It can make use of fuzzy logic for processing inputs or outputs that are coupled with deep learning models to allow accelerating the training of deep learning with fuzzy logic systems [37], when the data are noisy, heterogeneous, incomplete, or vague. As a useful approach, fuzzy neural networks are broadly adopted in fields to solve practical problems in various applications. A nonstationary FNN consisting of fuzzy logic and neural networks is introduced for clustering and regression problems [47]. A multilayer FNN is reported for the tasks of image clustering [42], while another work of FNN is described for medical images super resolution [41]. FNN is employed for robotic controls as a complex nonlinear application [48]. Das *et al.* [7] present their work on fuzzy-neuro model for data classification and feature reduction in data analytics. An adaptive FNN is introduced in the application of anomaly detection on measurement information of underwater acoustic ranging errors of autonomous underwater vehicles [44]. A self-organizing FNN is presented for the nonlinear system modeling [36].

FNN systems are also reported in prediction tasks according to past events or time series data. A multi-layer adaptive FNN is presented for student performance prediction in four online courses [40]. A prediction model using time-series recurrent neural network is reported for the stock price prediction [25]. Ferdaus *et al.* introduce a rule-based FNN learning system with two multi-objective evolutionary algorithms to forecast time-varying stock indexes [11]. A type-3 fuzzy aggregator is ensembled with the neural networks to be a prediction method on time series data of Humanitarian Data Exchange and Dow Jones [6]. A multi-functional recurrent FNN is introduced for the Chaotic time series prediction that utilises Takagi-Sugeno-Kang (TSK) fuzzy rules [28]. A residual deep fuzzy system is presented with several time-series datasets including subway passenger flow, traffic flow, and chaotic time series, etc. [24]. A type-2 FNN is tested with the Henon chaotic time series prediction [23]. The bankruptcy prediction and financial distress prediction are reported using the fuzzy convolutional neural networks [15].

There exists a large area of interest in the application of machine learning in prediction systems that involve constant changes in data, patterns, and trends, such as the prediction of financial markets [25,45] etc. Trend reversals of time series financial data can be predicted through analysing technical indicators [18,45,46]. Various machine learning (ML) or evolutionary algorithms are utilized to optimise the parameters to improve the accuracy of time series predictions. Genetic algorithms (GA) are employed as the optimisation algorithms in the predictions of financial trading systems [45]. Several ML regression algorithms optimised by GA are introduced for stock price forecasting [46]. An approach using recurrent neural networks (RNN) is optimised by GA to predict daily price movements of three market indexes [12]. The implementation of such predictions can reap financial benefits in the financial trading markets.

Even though FNN systems provide interpretability for human experts without prior knowledge in machine learning, it is not easy to understand how the systems draw conclusions in the output layer based on the input data. The mechanism and behaviour of FNN systems are only known by designers. It lacks transparency to users on how rules are generated and how data links are connected from multiple layers of FNN systems, which is similar to other black box machine learning models [4]. Black box nature of these models is difficult be directly explained [1]. As such, it is not easy for users to know how to tune the parameters and how to explore the maximum performance potentials of the FNN systems. FNN systems still fall behind deep learning models in terms of their predictive ability in complex modelling tasks when there are significant data shifts.

The main contributions of this paper are as follows.

A fuzzy-embedded recurrent neural network (FE-RNN) architecture is proposed to learn incrementally and inference on unseen time-series data using a developed pseudo-online incremental learning. It is able to incorporate newly acquired knowledge into classes of rules through the proposed learning process of FE-RNN, with the merged membership functions or derived new classes of

membership functions. FE-RNN performs the inferencing and data prediction using a deep RNN with back-propagation through time. Its predictive performance is compared against those of several other architectures reported in literature. The RNN within an embedded FNN is used to derive the data driven implication to map the input and output fuzzy spaces. The data driven implication mimics closely the entailment of data from the input to the output spaces. In this paper, we use the definition of interpretability as the qualitative understanding of the relationship between input and output features in the form of *IF-THEN* rules according to the membership functions derived.

To evaluate the performances of the proposed FE-RNN, several experiments are conducted. It is first assessed based on its forecasting ability on Chaotic Mackey-Glass time-series datasets. Two performance metrics are used in the benchmark assessment: root mean square error (RMSE); and Pearson's product-moment correlation coefficient (Pearson's R) between the predicted results and the actual results of the time series data. The experiment results according to the performance metrics of the FE-RNN will be analysed. Next, the FE-RNN is employed to predict the daily prices of three market indexes. The experiment results and interpretability of FE-RNN will be discussed in detail.

Next, the proposed FE-RNN is used as a stock price predictor alongside a GA optimised trading-decision strategy, named as the GA-optimised forecast-assisted Moving Average Convergence-Divergence Histogram (GA-fMACDH). In this paper, we illustrate the procedure to derive the GA-fMACDH strategy, and account for the whipsaw effects to reduce the unnecessary transactions. It is then benchmarked against the vanilla GA optimised MACDH (GA-MACDH) trading strategy, the conventional buy and hold strategy, and Tactical Buy and Hold (TBH) trading strategy reported in [45] using SeroFAM neuro-fuzzy network with GA-optimised fMACDH indicator in various high-volume exchange-traded funds (ETF). The result comparisons will be performed in terms of the improvements of investment returns and maximum drawdown.

The remaining parts of the paper are organised as follows. Section 2 introduces related works on RNN, online and offline learning of fuzzy systems. Section 3 presents the architecture and implementation of the proposed FE-RNN. Section 4 describes the benchmark experiments based on the chaotic time series data. Section 5 depicts the GA-fMACDH trading-decision system that utilises the predictions from the proposed FE-RNN to make judicious trading decisions. Section 6 concludes this paper.

2. Background knowledge

In this section, the background knowledge on the RNN and several types of fuzzy systems is introduced that are relevant to the design of FE-RNN.

2.1. Recurrent neural networks

The RNN concept was introduced as a network that was able to perform back-propagation through time, which requires a hidden state to capture a representation of the previous inputs. The computation of the gradients in RNNs involves long products of matrices. It may result in exploding gradients or vanishing gradients, which may obstruct the learning capability of the RNN. Hence, many techniques have been reported to counter these issues, such as Long short-term memory (LSTM) and Gated Recurrent Units (GRU) [5]. The GRU RNN has two gates: reset gate and update gate, having less parameters and higher training efficiency compared to those of LSTM. There is a gating mechanism of GRU for hidden states allowing the network to decide if the hidden states should be updated and reset. This mechanism allows for the network to selectively capture observations and reset the hidden state. For a GRU RNN, the information of the previous timestep is captured within the hidden state.

2.2. Online and offline learning of fuzzy systems

Computationally, the values in fuzzy systems are encoded as floating-point values between 0 and 1. These values are called degree of membership according to the membership functions (MF), where it quantifies the grade of the element to the corresponding fuzzy set. The antecedent is the cause of a fuzzy rule, and the consequent is the effect of the fuzzy rule.

Fuzzy systems are generally categorised into two different systems, Mamdani and TSK fuzzy systems. Mamdani fuzzy systems generally perform better in interpretability, while TSK systems generally perform better in precision. They differ in the fuzzy rules that are composed of, particularly the consequents of the fuzzy rules. The consequent of the Mamdani fuzzy rules is in linguistic terms, with being more interpretable. While the fuzzy rules in the TSK fuzzy systems use a linear piece-wise function of inputs as the consequent, with being harder to interpret. This paper uses the Mamdani fuzzy system that embeds the fuzzy rules within a RNN to associate the inference of RNN with the fuzzy counterpart.

In offline learning systems, the whole training data is available during the design phase. This means that the structure can be optimised for the current training data set and can optimally predict for unseen data. However, the structure is fixed after the design phase. Having a fixed structure may require to be redesigned and retrained when new data having a significantly different distribution, as seen in ANFIS [16]. For some time-critical applications, online fuzzy systems may be more suitable. Online learning fuzzy systems evolve their structure whenever new data arrives. These systems attempt to incorporate new data into their existing fuzzy clusters or create a new fuzzy cluster if needed. Online learning fuzzy systems often make use of one-pass clustering techniques, such as evolving fuzzy clustering method (EFCM) [19]. As the number of fuzzy clusters can change throughout the system operations, the underlying structure can also be evolved.

3. Proposed FE-RNN architecture

The proposed FE-RNN architecture is described in detail in this section. FE-RNN differs from a typical FNN structure by having an RNN layer as the inference layer. A typical architecture of a FNN has five layers: input layer, condition layer (also known as antecedent layer), rule-base layer (also known as inference layer), consequence layer and output layer, e.g., the Pseudo Outer-Product based Fuzzy Neural Network (POPFNN) [50]. Similarly, the deep FE-RNN architecture developed in this paper also has five layers. It aims to provide a data-driven entailment in the observed input and output relationships of the model, as well as the interpretability by the fuzzy system on operation of the RNN. This dual aspect of implication and explanation is achieved by embedding the fuzzy system with the RNN, allowing both networks to share the same input and output linguistic vocabulary.

As both the fuzzy system and the deep structure are incrementally tuned according to the same data, we can exploit the complementary relations of both systems by employing the fuzzy system to explain the data driven inference operation of the RNN. It combines the accuracy in the realisation of the high fidelity of the data driven implication capabilities of RNN and the interpretability of FNN. In the FE-RNN, a multi-input single-output (MISO) architecture is assumed. The high-level architecture of the developed FE-RNN with five layers is shown in Fig. 1.

Layer 1: Input Layer.

This layer is the input linguistic layer, receiving the crisp input values. Each neuron in this layer represents one feature of the input data into the system. The inputs are transmitted to the second layer, i.e., the condition layer.

Layer 2: Condition Layer.

The neurons in this layer are the input-label neurons. They are the antecedents of the fuzzy rules with their own individual parameters for their membership functions, generated by the clustering technique employed by the system. The values in this layer are fuzzified and then passed on to the third layer.

Layer 3: Inference/Rule-base Layer.

Neurons in this layer represent the fuzzy rules of the system. The linguistic terms – fuzzy memberships of the input and output linguistic variables form the vocabulary to the rule layer as well as the deep RNN. The fuzzy rules are generated using pseudo outer product (POP). The deep RNN is trained on the fuzzy membership values of the inputs and expected output. In the example of stock trading context, the inputs will be the trading volume, stock price, price changes, and momentum at time t. The expected output will be predicted price at the look-ahead time. Similarly, when using the deep RNN to forecast new data, the new data at time t are then fuzzified and passed to the deep RNN, where the output is then defuzzified to get the actual predicted output value.

Layer 4: Consequence Layer.

The neurons in this layer are called the output-label neurons. They are the consequents of the fuzzy rules and deep RNN inferencing of the previous layer.

Layer 5: Output Layer.

Each neuron in this layer represents one feature of the output data derived from the system. In this layer, defuzzification occurs to transform and derive the final inferencing results.

The overall system framework for the proposed FE-RNN is shown in Fig. 2. It shows the organisation of the different modules of FE-RNN and its respective data pathways during the training phase, inference phase and the interpretation phase.

In the training phase, the offline data is used to generate the parameters of the FE-RNN system. In the inference phase, the online data is passed through the system to generate the forecasted data. In the interpretation phase, the online data and forecasted data are used to provide interpretation of the system.

The FE-RNN is first trained through the learning modules on the top half of Fig. 2. The training data is used for the generation of the model. The learning algorithms employed by FE-RNN are split into two major parts: pseudo-online incremental learning used in the fuzzy structure, and back-propagation through time used in the deep RNN structure. These two parts will be presented individually in Sub-sections 3.1 and 3.2.

After the training and the generation of MFs, the input data is taken in the FE-RNN with inference process starting, shown in the bottom half of Fig. 2. The inference process of the FE-RNN is split into three steps: (1) fuzzification of inputs, (2) prediction using the

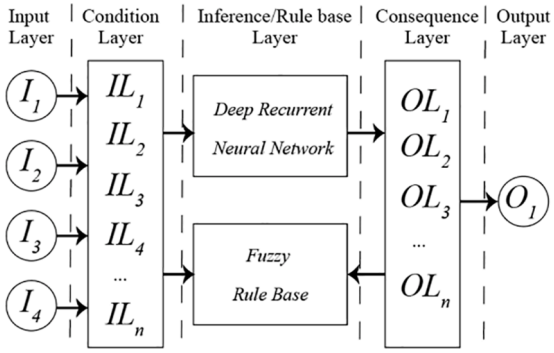

Fig. 1. Architecture of FE-RNN.

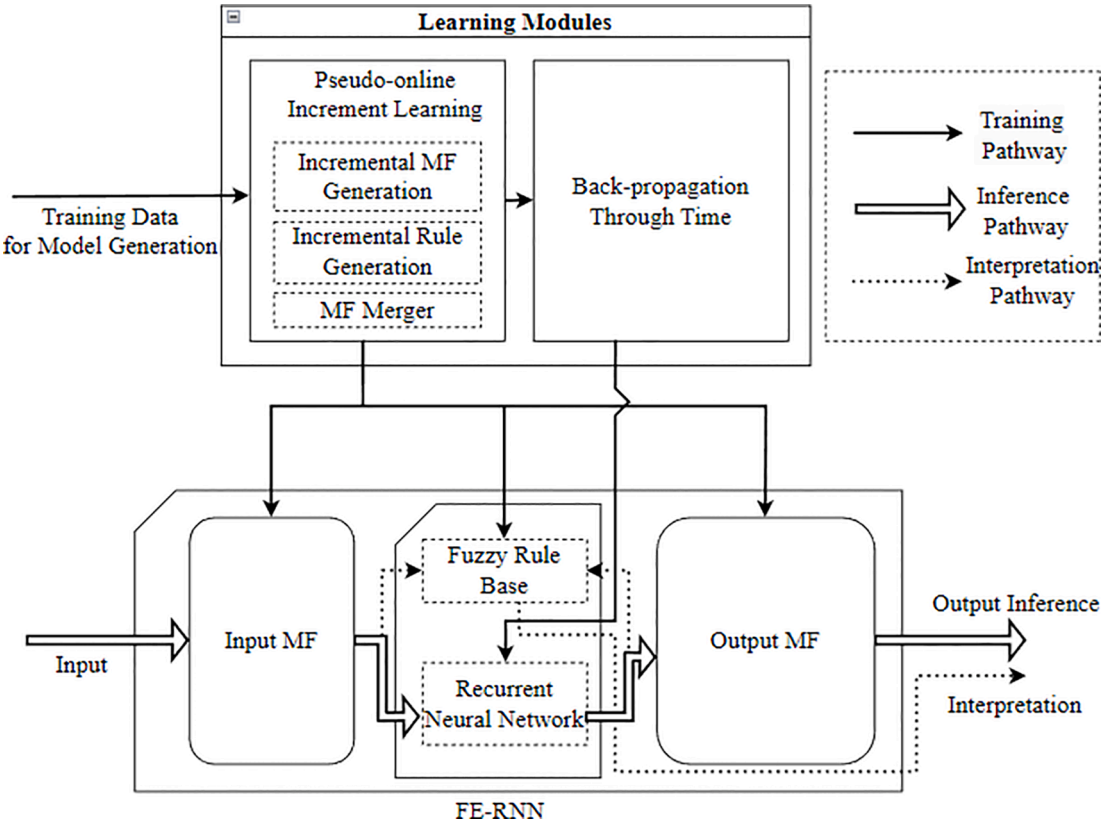

Fig. 2. Data Pathways inside FE-RNN.

data driven implementation of associative implication which is based on the deep GRU RNN within FE-RNN, and (3) defuzzification of the predicted outputs. The output is derived from FE-RNN next. The three steps of the inferencing process will be presented in the next sub-sections.

Lastly, the interpretation process of FE-RNN is done using the fuzzified inputs and raw output of the RNN to look up in the fuzzy rule base to understand the fired fuzzy rules. Thus, we are able to interpret the complex calculations within FE-RNN using simple fuzzy *IF-THEN* rules.

3.1. Pseudo-online incremental learning in FE-RNN

There are two steps for the pseudo-online incremental learning in FE-RNN as follows, that will be described in Sub-sections 3.1.1 and 3.1.2 next.

- Incremental membership function generation using a modified two-phase DIC algorithm adapted from [8]. The first phase is for the membership function generations, while the second phase is for the membership function merger.
- Incremental rule generation using POP and Hebbian weights adapted from [50] with another two phases: identifying the winning clusters in Phase 1 and updating the rule-base in Phase 2.

3.1.1. Step 1 - incremental MF generation

In the first step of the pseudo-online incremental learning in FE-RNN, the DIC algorithm is a bottom-up approach where the MF clusters are generated and grown to incorporate new training data [8]. In the FE-RNN learning process, a new singleton MF(centre value Γ_{new}, width σ_{new}) is created for the first training data. For the singleton MF, its centre value $\Gamma_{new} = \tau$ and its width $\sigma_{new} = 1e^{-10}$, where τ is the value of this data point.

Next, all relevant fuzzy rules are identified using the POP learning process [50]. Every training data is fed into the input layer and output layer simultaneously, where the membership values of each input layer node and output layer node are derived by their corresponding membership functions. It then produces the firing strength of each rule node in the inference/rule-base layer. The rule Hebbian weights of the links connecting the rule nodes and the output layer nodes are also updated accordingly.

The iteration is kept ongoing for all training data in the first phase of MF generations. If the next training data can fit in any current

MF, then all relevant fuzzy rules will be identified for it and the rule Hebbian weights will be updated as well. However, if it does not fit in any current MF, then another new singleton MF will be generated accordingly with the centre value Γ_{new} = value of this data point.

The second phase of the incremental membership function generation is the MF merger that merges two relatively close membership functions according to the merging condition, as shown in Eq. (1).

$$abs\left(\frac{\Gamma^{p}_{left}+\Gamma^{p}_{right}}{2}-\frac{\Gamma^{p+1}_{left}+\Gamma^{p+1}_{right}}{2}\right)<\frac{\frac{\Gamma^{1}_{left}+\Gamma^{1}_{right}}{2}+\frac{\Gamma^{m}_{left}+\Gamma^{m}_{right}}{2}}{2\times(m-1)} \tag{1}$$

where the p^{th} MF and $(p+1)^{th}$ MF are two neighbouring MFs; Γ^{p}_{left} and Γ^{p}_{right} are the left centre and right centre of the p^{th} MF respectively; Γ^{p+1}_{left} and Γ^{p+1}_{right} are the left centre and right centre of the $(p+1)^{th}$ MF respectively; m is the total number of membership functions belonging to the feature; Γ^{m}_{left} and Γ^{m}_{right} are the left centre and right centre of the m^{th} MF respectively.

The second phase starts calculating the distance between the centres of the first MF and the last MF of the m number of MF. For each MF, it iterates through every neighbouring MF to check if their distance fulfils the merging condition.

If a neighbouring MF pair fulfils the merging condition, these two MF will be merged into one MF, that is adapted from [8]. The parameters for the merged MF are given in Eq. (2).

$$\begin{aligned}\Gamma^{new}_{left} &= \Gamma^{p}_{left}\\ \Gamma^{new}_{right} &= \Gamma^{p+1}_{right}\\ \sigma^{new}_{left} &= \sigma^{p}_{left}\\ \sigma^{new}_{right} &= \sigma^{p+1}_{right}\end{aligned} \tag{2}$$

where Γ^{new}_{left} is the left centre of the merged MF;Γ^{new}_{right} is right centre of the merged MF; σ^{new}_{left} is the left width of the merged MF;σ^{new}_{right} is the right width of the merged MF; σ^{p}_{left} is the left width of the p^{th} MF; σ^{p+1}_{right} is the right width of the $(p+1)^{th}$ MF.

Algorithm 1 Pseudocode of the FE-RNN training.

```
Algorithm 1: FE-RNN Training Dataflow
Result: Trained FE-RNN model
/* generate membership functions and rulebase */
for i in trainingData do
    if i fits in current membership functions then
        expand current membership function;
    else
        generate new membership function;
    end
    if winning clusters have existing rule then
        update rule with firing strength;
    else
        create new rule;
    end
    for rule in ruleBase do
        if rule is current firing rule then
            continue
        else
            apply forgetting factor to rule;
        end
    end
end
/* performing merging of nearby clusters*/
calculate distance(mf_0, mf_n);
for mf_a in membershipFunctions do
    for mf_b in membershipFunctions do
        calculate distance(mf_a, mf_b);
        if mf_a and mf_b are close enough then
            merge mf_a and mf_b;
            update rulebase;
            recalculate distance(mf_0, mf_n);
        else
            continue
        end
    end
end
/*prepare RNN training data*/
```

(continued on next page)

(*continued*)

```
for i in trainingData do
                                    fuzzify i using final membership functions;
end
/*perform RNN training*/
for i = 0; i < maxEpochs; i = i + 1 do
                                    perform backpropagation using fuzzified data;
end
```

Next, the rule base is updated to cater for the changes in the MFs due to the MF merger. Those rules with the same antecedent and consequent have their weights added together.

The iteration is repeated for every MF, till all MF pairs that fulfil the merging condition are merged and their rule bases are updated. It marks the completion of the incremental MF generation process for FE-RNN to derive the final MF. All the training data are fuzzified using the final MF. The fuzzified data will be used to train the RNN networks in the FE-RNN training process.

The modified discrete incremental clustering algorithm is adapted from [8] and highlighted in Algorithm 1. The first *for loop* is to generate the MF and rule base iteratively. For every training data, if the membership value of the input or output layer node is less than its threshold (i.e., 0.5), it is deemed unfit for the current clusters of MF. Hence, a new singleton MF is created.

Next, all MFs under the same input/output layer nodes are updated with the new width σ_{new} as shown in Eq. (3).

$$\sigma_{new} = \frac{\tau - 0.5 \times \left(\max\left(d\text{ata}_p\right) - \min\left(d\text{ata}_p\right)\right)}{\sqrt{2\ln(100)}} \tag{3}$$

where $data_p$ are the existing data points of the same input/output layer nodes.

On the contrary, when the membership value of the training data is within [0.5, 1), it is deemed fit for the current MF. The current MF will be expanded. The expanded MF is a two-sided Gaussian function with the left centre Γ'_{left} and right centre Γ'_{left}, respectively shown in Eq. (4).

$$\Gamma'_{left} = \Gamma_{left} - \varphi \times (\Gamma_{left} - \tau)$$

$$\Gamma'_{right} = \Gamma_{right} - \varphi \times \left(\tau - \Gamma_{right}\right) \tag{4}$$

where Γ_{left} and Γ_{right} are the left centre and right centre of the original MF before expansion, respectively; φ is the plasticity parameter of the original MF; τ is the value of the training data.

After the expansion of MF, the value of φ is reduced by 1/3 to decrease the amount of the expansion shown in Eq. (5).

$$\varphi = \varphi \times \frac{2}{3} \tag{5}$$

3.1.2. Step 2 - incremental rule generation

As the second step of the pseudo-online incremental learning in FE-RNN, the incremental rule generation algorithm is also a bottom-up approach using POP and Hebbian Learning adapted from [50], that consists of two phases as follows.

The first phase is to identify the winning clusters. At each iteration for the data in this phase, the algorithm identifies the winning MF clusters for each input and output layer node based on the computations in Eq. (6).

$$u_{c_i}(x_i) = \max \mu_{i,j}(x_i)$$

$$u_D(y) = \max \mu_l(y) \tag{6}$$

where $u_{i,j}(x_i)$ is the membership value of the j^{th} MF of the i^{th} input layer node;$u_l(y)$ is the membership value of the output layer node;c_i is the winning cluster for the i^{th} input; and D is the winning cluster for the target value.

The winning clusters for the input layer nodes are selected and denoted with C consisting of a vector of c_i. The winning cluster for the output layer node is then selected and denoted with D.

The second phase is to update the rule base. In this phase, the rule base will be updated according to the identified winning clusters. Each rule is defined as an ordered pair *{C, D}*, where C is the antecedent of the rule and D is the consequent of the rule. Each rule is also tagged with a pseudo-weight $w_{\{C,D\}}$, which represents its importance.

The firing strength of the rule, f_{rule} is computed using POP shown in Eq. (7).

$$f_{fw} = \min\left(\mu_{C_i}(x_i)\right),\ \mu = u_D(y)$$

$$f_{rule} = f_{fw} \times \mu \tag{7}$$

where f_{fw} is the forward rule firing strength as the minimum value among all membership values in the rule antecedent C; μ is the backward rule firing strength, as the membership value of the corresponding output layer node.

In the iteration of each data, if there is a rule with the same antecedent and consequent as those of the winning clusters C and D, the pseudo-weight of the rule will be increased to be $w_{\{C,D\}} = w_{\{C,D\}} + f_{rule}$ according to the Hebbian Learning Rule, where f_{rule} is the firing strength of this data at the current iteration. Otherwise, a new rule is created. Its weight is the firing strength of this rule, as $w_{\{C,D\}} = f_{rule}$.

For other rules that are not fired, the pseudo-weights are decreased by an amount dictated by the dynamic forgetting factor λ as: $w_{\{C,D\}} = w_{\{C,D\}} \times \lambda$. The computation of the λ value is shown in Eq. (8).

$$\lambda = e^{-\frac{\left|\frac{p-l^e}{5}\right|+1}{\sqrt{ni_e \times n_r}}} \tag{8}$$

where p is the current iteration;l^e is the last iteration when the rule e was fired; ni_e is the number of times the rule e was fired; and n_r is the total number of rules in the rule-base.

The forgetting factor is adapted from [8]. In order to ensure that the weightage of rules is not drastically reduced, the value of λ is set in the range of 0.9 and 0.99, shown in Eq. (9).

$$\lambda = \begin{cases} 0.9, & if\ \lambda \leq 0.9 \\ \lambda, & if\ 0.9 < \lambda < 0.99 \\ 0.99, & if\ \lambda \geq 0.99 \end{cases} \tag{9}$$

Lastly, the pseudo-weights $w_{\{C,D\}}$ are normalised, as inspired by the concept of lateral inhibition.

3.2. Back-propagation through time in FE-RNN

Besides the pseudo-online incremental learning for the fuzzy structure of FE-RNN, the back-propagation through time is another block in the learning module of FE-RNN, that is used for the deep RNN structure. The parallel deep embedded structure used in this paper is a five-layer structure: one input layer, three fully connected 100-neuron GRU layers, and an output layer. The input layer of the deep structure shares the same vocabulary as the condition layer of FE-RNN. As such, the fuzzified membership values become the input of the deep structure. Similarly, the output layer of the deep structure shares the same vocabulary as the consequent layer of the FE-RNN. That is the output of the deep structure is the input to the fuzzified membership values of the consequent layer. The learning starts after the pseudo-online incremental learning of the membership functions. The model is trained with the fuzzified membership values of the input data and target data. It is trained with a lookback window as 25 timesteps using the adaptive moment estimation (Adam) optimizer. The decay rate of gradient moving average for the Adam solver is set 0.9 (default value), with the squared gradient decay rate chosen 0.999. The total training time is set as 30 epochs. The initial learning rate is set as 0.005, which is reduced by half every 3 epochs. The value of the L_2 regularisation factor is chosen as the default value 0.001 for the GRU training in MATLAB. Table 1 shows the training parameters used for the model. The input and target data for the GRU RNN prediction will be discussed in the next sub-section.

3.3. Inference process of FE-RNN

The inferencing process of FE-RNN includes three steps as follows.

3.3.1. Fuzzification of inputs

Firstly, the input is fed from the input layer of FE-RNN to the condition layer. The equations for the input layer are described in Eq. (10).

$$f_i^I = x_i$$
$$o_i^I = f_i^I \tag{10}$$

Table 1
Parameters used in Deep Structure Training.

Lookback Window (timesteps)	25
Optimiser	Adam
Gradient Threshold	50
Initial Learning Rate	0.005
Learning Rate Reduction Factor	2
Learning Rate Reduction Period (epochs)	3
L_2 Regularisation Factor	0.0001
Adam Gradient Decay Factor	0.9
Adam Squared Gradient Decay Factor	0.999
Training Time (epochs)	30

where f_i^I and o_i^I are the i^{th} input and output of the input layer; x_i is the i^{th} feature of the data input.

For every input in the condition layer, it goes through fuzzification by obtaining the membership values from each MF. In FE-RNN, each MF is a two-sided Gaussian function. The output of the condition layer is expressed by Eq. (11).

$$f_{i,j}^{II} = o_i^I$$

$$o_{i,j}^{II} = \begin{cases} exp\left(\dfrac{-\left(f_{i,j}^{II} - \left(\Gamma_{left}^{i,j}\right)\right)^2}{2 \times \left(\sigma_{left}^{i,j}\right)^2}\right), \; for \; f_{i,j}^{II} < \Gamma_{left}^{i,j} \\ 1, \; for \; \Gamma_{left}^{i,j} \le f_{i,j}^{II} \le \Gamma_{right}^{i,j} \\ exp\left(\dfrac{-\left(f_{i,j}^{II} - \left(\Gamma_{right}^{i,j}\right)\right)^2}{2 \times \left(\sigma_{right}^{i,j}\right)^2}\right), \; for \; f_{i,j}^{II} > \Gamma_{right}^{i,j} \end{cases} \tag{11}$$

where $\Gamma_{left}^{i,j}$ and $\Gamma_{right}^{i,j}$ are the left and right centres of the j^{th} MF for the i^{th} feature respectively; $\sigma_{left}^{i,j}$ and $\sigma_{right}^{i,j}$ are the left and right widths of the j^{th} MF for the i^{th} feature respectively; $f_{i,j}^{II}$ and $o_{i,j}^{II}$ are the input and output for the j^{th} MF for the i^{th} feature of the condition layer respectively.

3.3.2. Prediction using deep GRU RNN

The output from the condition layer is concatenated together, with the first element being the first MF from the first input feature, and the last element being the last MF from the last input feature. Then, the input is fed into the GRU RNN model which has been trained using the parameters shown in Table 1. The model takes the input f^{III} and the hidden states that contain the extracted information of the previous 24 timesteps to compute the output. The output o^{III} is the prediction derived from the GRU RNN model. The input and output of the inference layer are given in Eq. (12).

$$f^{III} = concat\left(\mathrm{o}_{1,1}^{\mathrm{II}}, \mathrm{o}_{1,2}^{\mathrm{II}}, \cdots \mathrm{o}_{i,j}^{\mathrm{II}}\right) \tag{12}$$

where $\mathrm{o}_{i,j}^{\mathrm{II}}$ is the output for the j^{th} MF of the i^{th} feature of condition layer; f^{III} and o^{III} are the input and output of the inference layer in the GRU RNN model respectively.

3.3.3. Defuzzification of predicted outputs

Finally, the output of the inference layer is assigned as the input to the consequence layer of FE-RNN. The output of the consequence layer is derived subsequently. The process of the consequence layer is shown in Eq. (13).

$$f_{\mathrm{j}}^{\mathrm{IV}} = \mathrm{o}_{\mathrm{j}}^{\mathrm{III}}$$

$$\mathrm{o}_{\mathrm{j}}^{\mathrm{IV}} = f_{\mathrm{j}}^{\mathrm{IV}} \tag{13}$$

where $\mathrm{o}_{\mathrm{j}}^{\mathrm{III}}$ is the corresponding output for the j^{th} MF of the output; $f_{\mathrm{j}}^{\mathrm{IV}}$ and $o_{\mathrm{j}}^{\mathrm{IV}}$ are the input and output of the j^{th} output MF in the consequence layer respectively.

Then, the output of the consequence layer is passed on next as the input of the output layer. Defuzzification occurs in the output layer. The defuzzification is performed on the aggregated areas based on the corresponding inferred membership values using the centre-of-area method. The equation for the defuzzification is described in Eq. (14).

$$f_{\mathrm{j}}^{\mathrm{V}} = \mathrm{o}_{i,\mathrm{j}}^{\mathrm{IV}}$$

$$o^v = \frac{\sum_{j=1}^{n_{out}} f_{\mathrm{j}}^{\mathrm{V}} \times \dfrac{\left(\Gamma_{left}^{j} + \Gamma_{right}^{j}\right)}{2} \times \dfrac{\left(\sigma_{left}^{j} + \sigma_{right}^{j}\right)}{2}}{\sum_{j=1}^{n_{out}} f_{\mathrm{j}}^{\mathrm{V}} \times \dfrac{\left(\sigma_{left}^{j} + \sigma_{right}^{j}\right)}{2}} \tag{14}$$

where $f_{\mathrm{j}}^{\mathrm{V}}$ is the membership value of the j^{th} output MF; o^v is the final crisp value of the prediction; Γ_{left}^{j} and Γ_{right}^{j} are the left and right centres of the j^{th} output MF respectively; σ_{left}^{j} and σ_{right}^{j} are the left and right widths of the j^{th} output MF respectively.

As the output MF is a two-sided Gaussian function, the centre and widths are approximated by obtaining the mean of both left and right values for each centre and width.

4. Benchmark Experiments

A series of experiments are conducted to evaluate the performance of the proposed deep FE-RNN architecture. The experiments include forecasting data from the chaotic Mackey-Glass time series dataset and daily financial ETF prices. The proposed FE-RNN is an architecture that deals with regression problems. Hence, the performance metrics of RMSE and Pearson's R are used for the quantitative evaluations in the experiments.

RMSE is an evaluation metric used to measure the differences between values predicted by a model and the actual values observed, shown in Eq. (15). The lower the RMSE, the higher the accuracy of the prediction model.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(o_i - y_i)^2} \tag{15}$$

where n is the length of the sequence to be predicted; o_i is the predicted output at the i^{th} timestep; and y_i is the actual observed value at the i^{th} timestep.

The Pearson's R is another evaluation metric used in regression analysis to measure the strength of the relationship between the relative movements of two variables. The value of the Pearson's R ranges between -1 to 1. Its score of 1 signifies that the variable is perfectly and positively correlated to the other variable, meaning that an upward movement in the first variable results in an upwards movement in the second variable. Conversely, A Pearson's R score of -1 signifies that the data is perfectly and negatively correlated to the other data, meaning that an upward movement in the first data is matched by a downwards movement in the second data. The Pearson's R can be calculated in Eq. (16).

$$R = \frac{\sum_{i=1}^{n}(o_i - \bar{o})^2 (y_i - \bar{y})^2}{\sqrt{\left(\sum_{i=1}^{n}(o_i - \bar{o})^2\right)\left(\sum_{i=1}^{n}(y_i - \bar{y})^2\right)}} \tag{16}$$

where n is the length of the sequence to be predicted;o_i is the predicted output at the i^{th} timestep;$\bar{o}$ is the mean of the predicted outputs;y_i is the actual observed value at the i^{th} timestep; and $\bar{y}$ is the mean of the actual observed values.

For the experiments on the chaotic Mackey-Glass time series dataset, FE-RNN is compared against other Mamdani and TSK fuzzy systems, and a deep 3-layered 100-neuron GRU RNN.

4.1. Chaotic Mackey-Glass time series

For the experiment on the chaotic Mackey-Glass time series, the hypothesis is that due to the good online learning ability and interpretability of the proposed FE-RNN, it should be able to achieve stable and consistent prediction accuracy under different look-ahead time windows.

To validate the experiment hypothesis, the chaotic Mackey-Glass time series is generated using the Mackey-Glass equation [2], which is a nonlinear time-delay differential equation, shown in Eq. (17):

$$\frac{dx_t}{dt} = \beta\frac{x_{t-\tau}}{1 + x_{t-\tau}^{n}} - \gamma x_t, \quad \gamma, \beta, n > 0 \tag{17}$$

where the parameters γ, β, n are constant values; and $x_{t-\tau}$ is the value of the variable x_t at time $(t - \tau)$; τ is the delayed timesteps from

Table 2
Results of Mackey-Glass Experiments.

Architecture	$t+1$			$t+2$			$t+4$		
	RMSE ($\times 10^{-2}$)	R ($\times 10^{-2}$)	No. of Rules	RMSE ($\times 10^{-2}$)	R ($\times 10^{-2}$)	No. of Rules	RMSE ($\times 10^{-2}$)	R ($\times 10^{-2}$)	No. of Rules
Proposed FE-RNN	5.52	99.50	423	6.05	99.38	439	6.54	99.10	460
Mamdani Systems									
PIE-RSPOP [14]	2.29	99.49	533	6.68	96.25	224	9.08	92.61	241
ieRSPOP [8]	2.37	99.45	701	3.59	98.75	701	6.23	96.24	701
EFuNN [19]	1.10	98.70	13	2.95	98.40	18	4.78	97.50	32
SAFIN [39]	6.70	99.40	85	23.00	97.90	85	67.00	93.20	85
TSK Systems									
ANFIS [16]	0.05	100.00	13	0.16	100.00	15	58.10	99.90	17
DENFIS [20]	0.03	100.00	13	0.13	100.00	13	83.10	99.90	13
Deep Structure									
Vanilla GRU RNN	0.03	99.20	–	4.52	97.99	–	7.27	94.72	–

the time t. In the experiment of this paper, we use $\gamma = 0.1, \beta = 0.2, n = 10, \tau = 17$ as the constant values. Depending on the value settings of these parameters, the Mackey-Glass equation illustrates the appearance in a range of complex chaotic dynamics with periodic oscillations [2].

The chaotic Mackey-Glass time series can be formulated using $\{x_{t-m+1}, x_{t-m+2}, x_{t-m+3}, \cdots, x_t\}$ as the inputs to predict the future value x_{t+k}, where m is the look-back time window; k is the look-ahead time window; and t is the index of the time series. We set the following parameters: $m = 6$, $x_0 = 1.2$, and $k = 1, 2, 4$ for the three different experiments. The three different experiments benchmark the FE-RNN forecasting ability for look-ahead time windows of $t + 1$, $t + 2$, and $t + 4$, respectively.

For each experiment, a total number of 1000 data points is used. The first 500 data points are used as the training set, and the remaining 500 data points are the testing set. The 1000 data points are plotted alongside their corresponding predicted values by FE-RNN; But only the data points of the testing set are used in the computation of the values of RMSE and Pearson's R. The training parameters for FE-RNN can be found in Table 1.

The Mackey-Glass experiment results of FE-RNN and other benchmarking methods adapted from [14] are shown in Table 2. The experiments are conducted 30 times using different random initial weights, with the mean values utilized in the comparisons. It is seen that the TSK fuzzy systems outperform the Mamdani fuzzy systems. TSK systems have a lower RMSE score and a higher Pearson's R score, except for the $t + 4$ experiment, where the TSK systems have much higher RMSE. It shows that the two TSK systems, ANFIS and DENFIS, are good at correlating and having predicted values following the movements of the actual values. But the accuracy of the predicted values gets notable drops with a larger look-ahead time at $t + 4$. Furthermore, TSK systems are not as interpretable as Mamdani systems as they do not make use of the complete fuzzy to fuzzy *if-then* rule structure. When compared against other Mamdani systems, FE-RNN has a higher Pearson's R score consistently in all experiments, that indicates its capability to perform consistently in the prediction tasks for future values. EFuNN [19] has the overall lowest RMSE score and generates the fewest rules. Compared to the vanilla GRU RNN deep structure, FE-RNN manages to perform better on the consistency of Pearson's R score while being interpretable. At $t + 4$ forecasting, FE-RNN obtains higher R values than those of four Mamdani systems and vanilla GRU RNN by up to 7.0%. The number of rules of FE-RNN shown in Table 2 are derived from the learning mechanism to cover the regions of data distribution. Only relevant rules will be fired at a time, that are activated in the right region of data distributions for the data driven applications. FE-RNN has similar Pearson's R as TSK fuzzy systems, while having similar number of fuzzy rules to Mamdani systems, overall achieving a good level of balance on interpretability for deep networks with little reduction in performance.

In the $t + 1$ experiment, the GRU RNN model obtains the accuracy close to the ANFIS [16] and DENFIS [20] under the category of TSK fuzzy systems. However, the GRU RNN model suffers in larger performance loss when the look-ahead time is increased as compared to the FE-RNN, seen in the lower R score in the $t + 2$ and $t + 4$ experiments. The overall performance of FE-RNN is more consistent when the look-ahead time is increased.

Although the FE-RNN architecture does not perform as good as other fuzzy systems benchmarked on the RMSE score, the FE-RNN performs well in terms of the Pearson's R score, meaning FE-RNN excels in predicting trends as compared to other fuzzy systems. This is likely caused by the ability of the deep RNN inference system to capture time-dependencies in data. The FE-RNN also manages to perform similarly, if not better than the vanilla GRU RNN model, while increasing the interpretability of the structure compared to the vanilla GRU RNN.

The predicted results for the Mackey-Glass time series by the developed FE-RNN can be seen in Fig. 3 - Fig. 5 for $t + 1$, $t + 2$ and $t + 4$ predictions, respectively. The red plots are the mean prediction values after the 30 times of experiment with random initial weights. The blue plots are the mean values added with the standard deviations. The orange plots are the mean values minus the standard deviations. The purple plots are for the target data. It is observed from Figs. 3 - 5 that the FE-RNN is able to achieve good prediction accuracy close to the actual data.

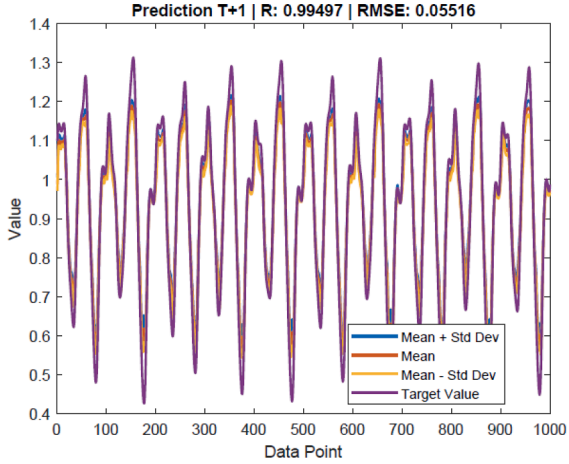

Fig. 3. Results of FE-RNN for $t + 1$ Mackey Glass Benchmark.

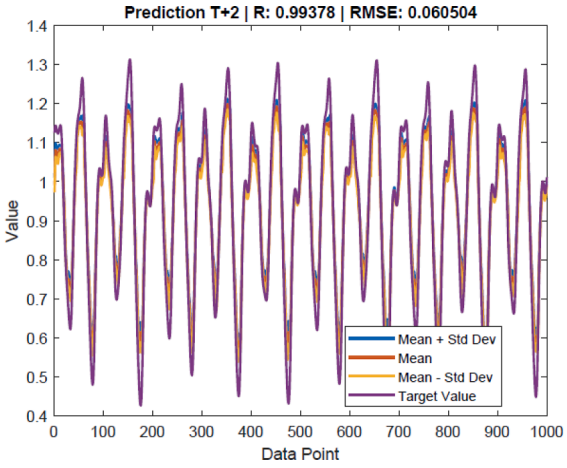

Fig. 4. FE-RNN Results for $t + 2$ Mackey Glass Benchmark.

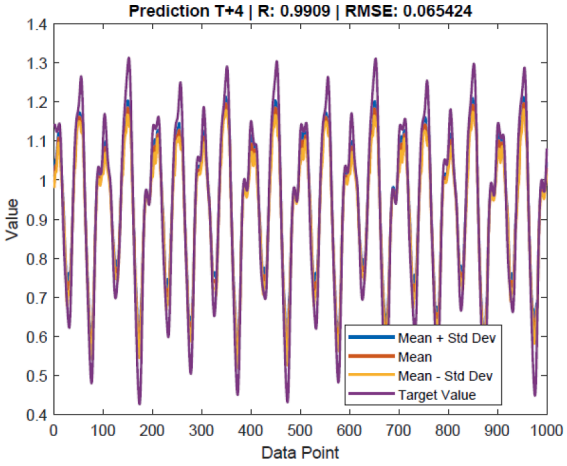

Fig. 5. FE-RNN Results for $t + 4$ Mackey Glass Benchmark.

4.2. *Daily stock price forecasting*

In the next experiment, the daily price stock forecasting is used to illustrate the ability of FE-RNN to model time series data. For this experiment, we benchmark FE-RNN for predicting multiple high-volume indexes in the stock market. The experiment evaluates the look-ahead periods from 1 day to 7 days. The indexes that we are attempting to forecast in this benchmark experiment include the Standard & Poor's S&P 500 index (^GSPC), DJIA index (^DJI), and the Hang Seng Index (^HSI). The daily pricing data is sourced from Yahoo! Finance and split into training and testing datasets. The training dataset is the data from 24th March 2005 – 19th January 2011 for daily stock closing prices. The testing dataset is the data from 20th January 2011 – 5th January 2021 for daily stock closing prices, as shown in Table 3. The training parameters are shown in Table 1.

4.2.1. *Data pre-processing*

For the data pre-processing of this experiment, the input and output used for FE-RNN are formulated in Eq. (18).

$$x_1 = V$$

Table 3
Stock Price Forecasting Experiment.

Stock	Training data size	Testing data size
S&P 500	1,467	2,507
DJIA	1,467	2,507
HSI	1,443	2,456

$$x_2 = \dot{p}_t = \frac{p_t - p_{ref,t}}{p_{ref,t}}$$

$$x_3 = \ddot{p}_t = \dot{p}_t - \dot{p}_{t-1} = \frac{p_t - p_{ref,t}}{p_{ref,t}} - \frac{p_{t-1} - p_{ref,t-1}}{p_{ref,t-1}} \tag{18}$$

$$x_4 = p_t^{\cdots} = \ddot{p}_t - \ddot{p}_{t-1} = \frac{p_t - p_{ref,t}}{p_{ref,t}} + \frac{p_{t-2} - p_{ref,t-2}}{p_{ref,t-2}}$$

$$y = \dot{P}_{t+1} = \frac{p_{t+1} - p_{ref,t+1}}{p_{ref,t+1}}$$

where V is the volume of units of the stock bought and sold;$\dot{p}_t$ is the normalised price of the stock at time t;$\ddot{p}_t$ is the velocity of the price or the rate of change of $\dot{p}_t$ at time t;$p^{\cdots}{}_t$ is the momentum of the price, or the rate of change of $\ddot{p}_t$ at time t;p_t is the closing price of the stock; and $p_{ref,t}$ is the reference price of the stock at time t.

Generating the input and output values requires a reference price point, as shown in Fig. 6. The reference price for time t is a closing price of time somewhere between t-21 to t-8, depending on the sequencing of the data. Having a reference point for the prediction allows FE-RNN to perform predictions based on the delta changes of values, instead of the absolute values. After the prediction of the value is done, the reference price is then used again to derive the actual crisp prediction value. Finally, the predictions are benchmarked using the Pearson's R score and RMSE of a 5-day moving average between the prediction and actual data values.

4.2.2. *Multi-day look-ahead forecasting analysis*

We generate one model for each look-ahead forecast target. In total, 21 models are generated, with seven models for each of the three indexes. The results of the experiment are tabulated in Table 4.

As expected, the prediction of the FE-RNN is the best when predicting $t + 1$ values for all three indexes observed in Table 4. The Pearson's R score goes lower as the look-ahead period increases. Similarly, the RMSE score also increases as the look-ahead period increases. Interestingly, the correlation during 1 day to 7 days look-ahead predictions for the US market manages to be above 99%, while the Hong Kong market drops to 96.4% when attempting to predict prices 7 days into the future. Although the correlation scores are good, the predicted value still lags behind the actual value by 2–5 days, as can be seen in the $t + 7$ lagged prediction results of FE-RNN for S&P 500 index as shown in Fig. 7.

Deng *et al.* [9] present their neutral network model named Multivariate Empirical Mode Decomposition LSTM (MEMD-LSTM) that use three market indexes in the experiments: S&P 500, HSI, and Shanghai Stock Exchange (SSE), using the metrics of mean absolute percentage error (MAPE), RMSE, and directional symmetry (DS). The experiment results of other three neutral network models are also introduced, including backpropagation neural networks (BPNN), LSTM, and Empirical Mode Decomposition LSTM (EMD-LSTM) to predict values on $t + 1$, $t + 3$, $t + 5$, $t + 10$, etc. [9]. Their experiment datasets and timeframes are different from those of FE-RNN. The experiment datasets for S&P 500 in [9] are: 1845 training data from 4th January 2010 to 2nd May 2017; 462 validation data from 3rd May 2017 to 5th March 2019 to tune the network parameters; and 462 testing data from 6th March 2019 to 31st December 2020. Their experiment datasets for HSI are: 1504 training data from 4th January 2010 to 17th February 2016; 376 validation data from 18th February 2016 to 28th August 2017; and 376 testing data from 29th August 2017 to 8th March 2019.

In this paper, the four neural network models, BPNN, LSTM, EMD-LSTM, and MEMD-LSTM are employed to benchmark with FE-RNN and illustrate the predictions with multiple look-ahead time. It is observed in Table 4 that FE-RNN outperforms these four neural network models in terms of RMSE when predicting the $t + 1$ values. FE-RNN obtains lower RMSE values by 21.6% − 74.6% for S&P 500 index, and 8.1% − 60.0% for HSI index. The $t + 2$ prediction value on S&P 500 index by FE-RNN is even better than the $t + 1$ predictions of the other four models. While MEMD-LSTM model has lower RMSE values in $t + 3$ and $t + 5$ predictions, there is no Pearson's R value reported in [9]. It is not so straightforward to compare the correlations in the predicted values with FE-RNN in the multiple look-ahead time windows. These four benchmark neural network models, BPNN, LSTM, EMD-LSTM, and MEMD-LSTM focus more on accuracy, without analysis on the interpretability of their networks. It is different from FE-RNN, that puts architecture design considerations on both accuracy and interpretability.

Observed results of this experiment, FE-RNN manages to predict the daily prices with a high degree of correlation (> 96%), even when attempting to predict multiple timesteps ahead. FE-RNN can infer price movements and predict future pricing directions in the

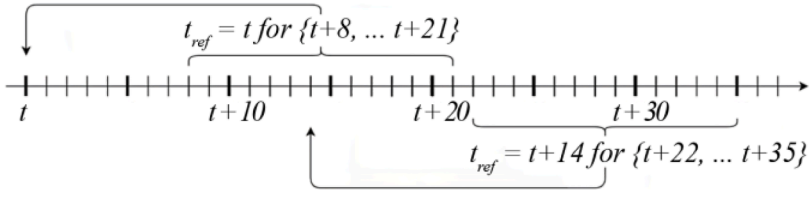

Fig. 6. Reference Pricing Method used in Stock Prediction.

Table 4
Results of FE-RNN in Multi-day Look-ahead Forecasting.

Model		S&P 500		DJIA		HSI	
	Forecast Target	RMSE	R ($\times 10^{-2}$)	RMSE	R ($\times 10^{-2}$)	RMSE	R ($\times 10^{-2}$)
Our FE-RNN	$t+1$	**28.221**	99.92	**293.695**	99.85	**298.061**	99.56
	$t+2$	**34.900**	99.85	399.906	99.72	504.897	98.73
	$t+3$	46.875	99.73	529.295	99.51	609.849	98.33
	$t+4$	52.170	99.67	578.594	99.42	680.135	97.85
	$t+5$	58.754	99.58	612.125	99.34	772.444	97.26
	$t+6$	63.323	99.50	637.920	99.28	816.179	96.88
	$t+7$	66.569	99.46	683.641	99.18	850.554	96.38
BPNN [9]	$t+1$	111.162	–	–	–	620.924	–
	$t+3$	153.525	–	–	–	636.540	–
	$t+5$	165.574	–	–	–	650.775	–
Single LSTM [9]	$t+1$	67.724	–	–	–	425.822	–
	$t+3$	69.953	–	–	–	436.977	–
	$t+5$	74.684	–	–	–	442.816	–
EMD-LSTM [9]	$t+1$	56.453	–	–	–	351.424	–
	$t+3$	58.106	–	–	–	365.785	–
	$t+5$	60.859	–	–	–	379.278	–
MEMD-LSTM [9]	$t+1$	36.019	–	–	–	324.292	–
	$t+3$	**38.568**	–	–	–	**331.833**	–
	$t+5$	**39.975**	–	–	–	**337.892**	–

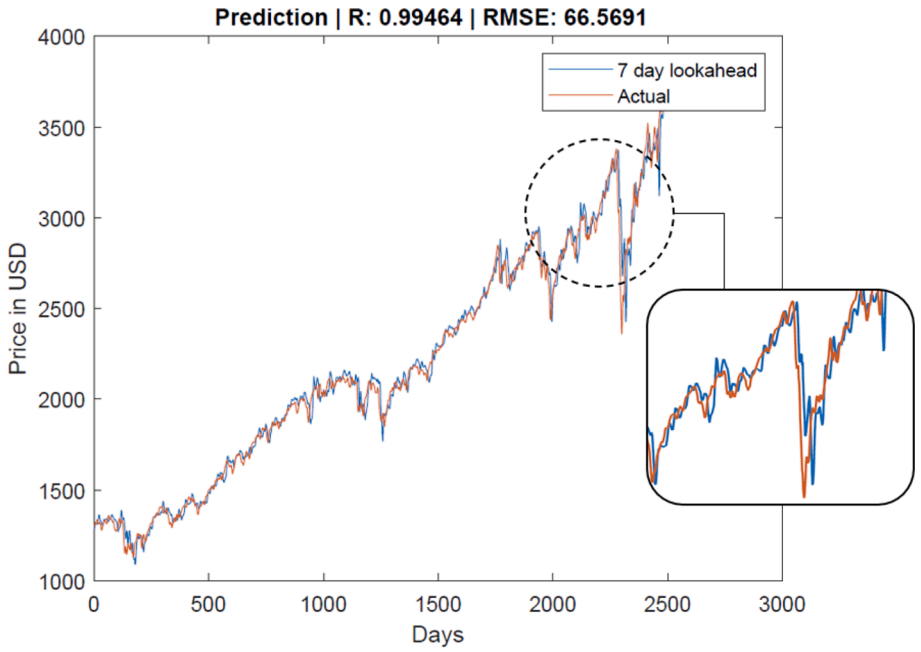

Fig. 7. Lagged Prediction on $t+7$ of S&P 500 index.

stock market well.

4.3. *Interpretability of FE-RNN inference process*

We attempted to interpret the inference process of FE-RNN, using an example of the daily stock prices of an ETF trust, SPDR S&P 500 ETF Trust (SPY). The rules fired with the highest firing strength are recorded and analysed. The main rules fired in the $t+1$ prediction for the SPY from 30/10/2019 to 9/6/2020 are shown in Fig. 8. Understandably, it is observed that the input price is largely correlated with the predicted price. We can also observe the impact of the other input features. For example, when the volume of the ETF is low, we can infer that the price change for the next day will be close to 0.

We can use the fired rules to explain the predicted price for the SPY ETF on 20/03/2020 shown in Fig. 9. It is observed that FE-RNN predicts the peak of the pricing for the SPY. The rule can be interpreted as: "IF the volume is *very high* AND the price is very *low* AND the

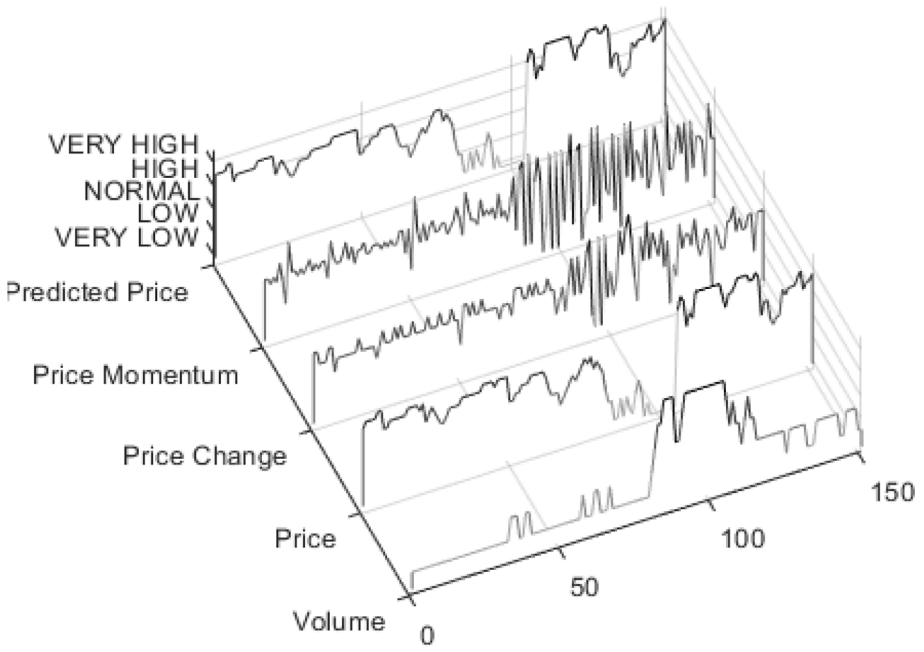

Fig. 8. Semantic Values of Rules Fired in FE-RNN for SPY prices.

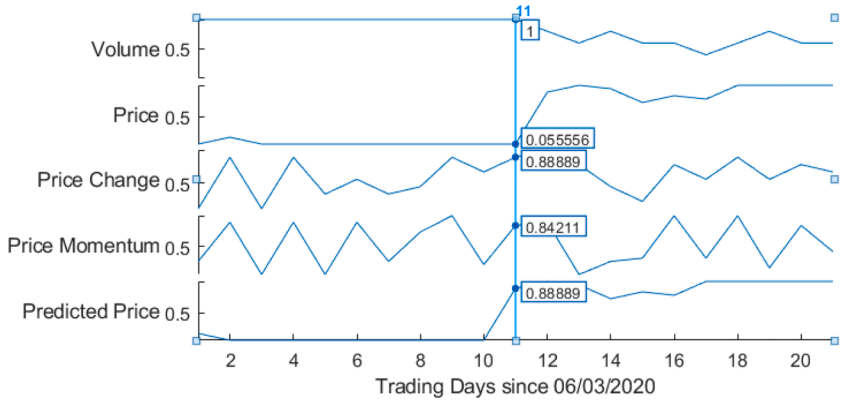

Fig. 9. Semantic Values of Rules Fired in FE-RNN on 20/03/2020.

price change is *very high* AND the price momentum is *very high*, THEN the predicted price is *very high*". Intuitively, this also means that we can interpret the deep RNN network signaling that there is a strong bullish trend, so the price will increase the next day. Hence, by observing the fired rules with the fuzzified input and output values, we can interpret the inference process of FE-RNN in the form of human explainable *IF-THEN* rules. The same interpretability principles are applicable to all the scenarios in the inference process explained using the fired rules. As there is little consensus at the moment according to the literature review [30], in this paper, we use the definition of interpretability as the qualitative understanding of the relationship between input and output features in the form of *IF-THEN* rules according to the membership functions derived. Our analyses illustrate the interpretability in qualitative terms achieved by FE-RNN, that increases the interpretability and tackles the black box nature of RNN, where it may not be so straightforward to achieving with normal RNN architecture. But the drawback of our current analysis on the interpretability of FE-RNN is that the quantitative measurements are missing. It is one of the limitations of this work to be addressed in our future work.

5. Trend trading using GA-fMACDH

In order to further evaluate the performance of the developed FE-RNN, we incorporate FE-RNN in a financial stock trading system. The prices of the assets in the portfolio are predicted by FE-RNN within a Moving Average Convergence-Divergence Histogram (MACDH) trading strategy adapted from [38], that employs GA to optimise the parameters of the trading strategy. The trading strategy consisting of FE-RNN is denoted as the GA-fMACDH strategy. We use GA-fMACDH to perform trading on six financial assets, namely:

- SPY, tracking the investment performance of the S&P 500 Index since January 1993,
- Vanguard European Stock Index Fund ETF (VGK), tracking the stock investment performance to companies in the developed economies of Europe since August 2002,
- Vanguard Emerging Markets Stock Index Fund ETF (VWO) tracking the performance of an index composed of companies in emerging market countries since June 2006,
- iShares Core US Aggregate Bond ETF (AGG) tracking the index performance to measure investment returns of the total U.S. investment-grade bond market since September 2003,
- Philadelphia Gold and Silver Index (XAU) composing of 30 companies involved in the gold or silver mining industry since January 1979,
- iShares US Energy ETF (IYE) tracking the investment performance of component companies in the energy segment including oil companies, services, and pipelines, etc. since June 2000.

These six financial assets cover and track the performances of different market segments. The data size and time frames for these six financial assets for experiments are shown in Table 5.

In the experiments, we assume that there is no slippage; this means that we can fill the order to purchase the assets at the closing price of the next day. The returns of the GA-fMACDH are compared against the buy and hold strategy and GA-MACDH strategy without using forecasted prices. The maximum drawdown across the testing period is also used to evaluate the performance of the trading strategy. The maximum drawdown is defined as the maximum percentage drop in the total returns from the start to the end of a period. It can be used to represent the risk profile of the portfolio, where a large maximum drawdown may mean the current strategy not be suitable for risk-averse investors.

5.1. MACD and MACDH

The Moving Average Convergence-Divergence (MACD) is a technical indicator used by many technical analysts for observing the market trends. The MACD is one of the simple and reliable technical indicators available in practice. However, the MACD is a lagging indicator. Hence, the MACDH has been developed to reduce the time lag of the MACD. The MACDH describes the rate of change of the MACD by subtracting the exponential moving average (EMA) from the MACD, as shown in Eq. (19):

$$MACD = MACD(fast, slow, signal) = EMA_{fast} - EMA_{slow}$$

$$MACDH = MACD - EMA_{signal}(MACD) \tag{19}$$

where $MACD(fast, slow, signal)$ is the MACD value calculated using fast and slow as the window size for EMA_{fast} and EMA_{slow} at time t respectively; $EMA_{signal}(MACD)$ is the EMA of MACD values using a window of size signal; and MACDH at time t is the EMA of window size signal subtracted from the MACD value.

The MACDH now represents a second-ordered oscillator that measures the rate of change of momentum. As such, when the MACDH crosses over the zero axis, assuming the absence of time lag, it represents that the momentum of the price action has peaked and it is the start of a reversal.

5.2. Forecast-assisted MACD and MACDH

As the MACDH indicator provides a good signal for trend reversals, we incorporate the forecasted prices from FE-RNN to reduce the time lag of the indicator. The fMACDH indicator uses the forecast-assisted EMA (fEMA). The fEMA uses exponentially weighted historical prices and reversed exponentially weighted forecasted prices to reduce the time lag, as shown in Fig. 10. Half of the prices come from the historical prices from t-13; while the other half ending at $t + 7$ is sourced from the forecasted prices, with a maximum forecast period of 7 days look-ahead using the proposed FE-RNN. Take note that the value of weights in the example shown in Fig. 10 serves as only an illustration to better understand the distribution between forecasted and historical prices without quantitative values.

The forecasted prices are put through further pre-processing before being used in the fEMA. The forecasted prices are extracted and converted into delta change values, which are then applied to the last seen closing price. The pre-processing to the forecasted price can be formulated in Eq. (20).

Table 5
Assets Data Used in GA-fMACDH Trading.

Stock	Training Dates	Training Data Size	Testing Dates	Testing Data Size
SPY	24/03/2005 – 19/01/2011	1,462	20/01/2011 – 05/01/2021	2,507
VGK	24/03/2005 – 19/01/2011	1,462	20/01/2011 – 05/01/2021	2,507
VWO	24/03/2005 – 19/01/2011	1,462	20/01/2011 – 05/01/2021	2,507
AGG	24/03/2005 – 19/01/2011	1,462	20/01/2011 – 05/01/2021	2,507
XAU	24/03/2005 – 19/01/2011	1,462	20/01/2011 – 05/01/2021	2,507
IYE	24/03/2005 – 19/01/2011	1,462	20/01/2011 – 05/01/2021	2,507

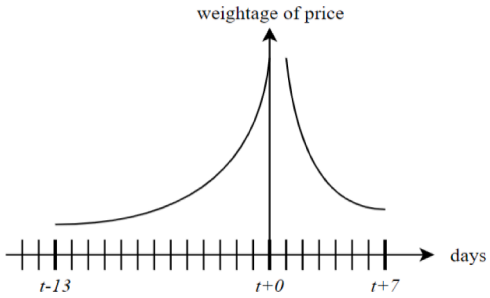

Fig. 10. Distribution of weights in a 21-day fEMA.

$$\widehat{y}(t+k) = y(t) + \left(y_{predict}(t+k) - y_{predict}(t+k-1)\right) \tag{20}$$

where $\widehat{y}(t+k)$ is the processed forecasted price at time $t+k$;$y(t)$ is the actual historical price at time t; and $y_{predict}(t+k)$ is the latest forecasted price for time $t+k$.

The equations for computing the fEMA, fMACD and fMACDH are described in Eq. (21).

$$\begin{aligned} fEMA(window) &= \frac{EMA_{d_{historical}}(y) + EMA_{d_{forecast}}(reverse(\widehat{y}))}{2} \\ d_{forecast} &= \max\left(\frac{window}{2}, 7\right) \\ d_{historical} &= window - d_{forecast} \\ fMACD(fast,\ slow) &= fEMA(fast) - fEMA(slow) \\ fMACDH(fast,\ slow,\ signal) &= fMACD(fast,\ slow) - EMA_{signal}(fMACD) \end{aligned} \tag{21}$$

where *fMACD(fast, slow, signal)* is the forecast-assisted MACD value calculated using fast and slow as the window size for $fEMA_{fast}$ and $fEMA_{slow}$ at time t respectively; $d_{forecast}$ is the number of forecasted daily price used in fEMA; $d_{historical}$ is the number of historical daily price in fEMA; and fMACDH is the forecast-assisted MACDH at time t that is the EMA of window size signal subtracted from the fMACD value.

5.3. Accounting for whipsaw effects

As the vanilla MACDH trading strategy is solely dependent on the MACDH value, it is susceptible to the whipsaw effects. This is where the MACDH frequently fluctuates at the zero axis, causing the system to buy and sell stocks very frequently without a significant amount of return. This makes the trading system less profitable as a larger portion of the profits would be lost to the commission leaks when there are unnecessarily frequent changes in the trading position. To account for this, we introduce a price percentage oscillator $fMACDH_{\%}$, that represents the price movement relative to the price in terms of percentage, shown in Eq. (22).

$$fMACDH_{\%}(fast,\ slow,\ signal) = \left| \frac{fMACDH(fast,\ slow,\ signal)}{0.5 \times (fEMA(fast) + fEMA(slow))} \right| \tag{22}$$

As such, the trading strategy is modified such that the trade signal is only valid when $fMACDH_{\%}$ is larger than the oscillation threshold, a. The modified trading strategy is formulated in Eq. (23).

$$P(t) = \begin{cases} 1, & if\ fMACDH_{\%} > a\ \cap\ fMACDH > 0 \\ 0, & if\ fMACDH_{\%} > a\ \cap\ fMACDH \leq 0 \\ & P(t-1),\ otherwise. \end{cases} \tag{23}$$

where $P(t)$ is the position held at time t;$P(t) = 1$ is a buy decision; $P(t) = 0$ is a sell decision; and $P(t) = P(t-1)$ means a hold decision without buying and selling stocks. The modified trading strategy only allows the buying and selling of financial assets and does not consider a short position as an option in this experiment.

5.4. Trading parameters optimisation using GA

GA is used to optimise the parameters *long*, *short*, *signal*, and a in the modified trading strategy. In the GA fitness function, the fitness score (the value that we are attempting to maximise) is the final portfolio value. The final portfolio value includes the deduction of 0.08% for each transaction to account for the transaction commission fees. The algorithm stops when there is no improvement in the

best performing population for 50 generations. The first 500 days of the dataset are used as the training data for the GA to benchmark the score of the population. The parameters are derived using GA shown in Table 6. It can be observed that the derived parameters generally follow the trend of *signal* < *fast* < *slow*.

5.5. Results & analysis

The results on the investment returns and max drawdown of the modified trading strategy are shown in Table 7. The results are benchmarked with GA-MACDH, as well as the buy and hold strategy. The improvements on the investment returns made by GA-fMACDH over GA-MACDH, and GA-fMACDH over buy and hold strategy are shown in Table 8. It is observed that GA-fMACDH outperforms GA-MACDH in all financial assets by 1.30% to 229.47% higher returns in the experiment. The FE-RNN based GA-fMACDH also outperforms the buy and hold strategy in five financial assets: SPY, VGK, VGO, XAU, and IYE by 48.54% to 137.90% higher returns; while the buy and hold strategy outperforms slightly the GA-fMACDH in the AGG ETF by 2.92% in return. However, the GA-fMACDH has a lower maximum drawdown on average when compared against the buy and hold strategy with −3.30% to −34.90%. This means that the GA-fMACDH is more defensive and exits a position when the price drops a lot. The FE-RNN based GA-fMACDH strategy can perform well when the underlying equity is volatile, as it produces more opportunity for the trading system.

For the illustration purposes of the trading outcomes, the improvements on the investment returns made by FE-RNN-based GA-fMACDH are compared to those of the TBH trading strategy using SeroFAM neuro-fuzzy network with GA-optimised fMACDH indicator reported in [45] on the same four financial assets as shown in Table 8. It is observed that FE-RNN based GA-fMACDH strategy makes much larger returns over its counterparts than those of the TBH over the Buy and hold strategy. But there are some differences on the timeframes and transaction commission fees of TBH strategy, that are 1st March 2017 – 31st March 2021 with 0.1% commission per transaction [45]. While the timeframe of the experiments in this paper is from 24th March 2005 to 5th Jan 2021 as shown in Table 5; and the transaction commission fees used in this paper is 0.08%. As such, strictly speaking, it is not a fair companion. It only helps us sense to certain extends and demonstrate qualitatively how well the FE-RNN based GA-fMACDH can improve the trading returns.

6. Conclusion

In this paper, the FE-RNN is proposed by embedding the features of a fuzzy system together with a deep RNN. The dual-view of this embedding allows the fuzzy association in FE-RNN to provide the better interpretation of RNN encoding and decoding. It employs the deep RNN to learn and compute the data driven implication for the fuzzy association from the input to the output fuzzy spaces. Such an approach permits a more accurate realisation of entailment in the fuzzy inference process.

Several experiments are conducted to evaluate the functions and performance of FE-RNN. In the first experiment for the prediction tasks of chaotic Mackey-Glass time series, FE-RNN is compared with seven benchmark models. FE-RNN obtains higher Pearson's R scores up to 7% consistently in all experiments than those of the Mamdani systems and Vanilla GRU RNN. FE-RNN achieves more consistent RMSE accuracy when predicting multiple days of look-ahead time windows, compared to the notable drops on RMSE of the TSK systems. It shows that FE-RNN gets more balanced performance and interpretability consistently.

In the second experiment for multiple days look-ahead forecasting of FE-RNN, three market indexes, S&P 500, DJI, and HSI are utilized for $t+1$ to $t+7$ days predictions. FE-RNN obtains the best results when predicting $t+1$ values. It achieves high R values consistently at different look-ahead days forecasting. FE-RNN also outperforms four benchmarking neutral network models in RMSE for $t+1$ predictions, with lower RMSE values by 21.6% to 74.6% for S&P 500 index, and 8.1% to 60.0% for HSI index.

In the last experiment, FE-RNN is used as a price predictor in the stock trading system using the GA-fMACDH trading strategy. The design of the stock trading system incorporates FE-RNN, GA optimization algorithms, GA-fMACDH optimized by GA, and countermeasure of the whipsaw effects in trading. With transaction commission fees included, it is employed to trade six financial assets: SPY, VGK, VGO, AGG, XAU, and IYE. The experiment results of FE-RNN based GA-fMACDH trading system are compared to GA-MACDH, as well as buy and hold strategies. It is observed that the GA-fMACDH trading system achieves 1.30% to 229.47% higher returns than those of GA-MACDH trading model for all financial assets. It also obtains 48.54% to 137.90% higher returns than those of the buy and hold strategy for five assets, except for AGG. The GA-fMACDH trading system has lower maximum drawdown compared to the buy and hold strategy for all six assets. The improvements on the returns of FE-RNN based GA-fMACDH trading system are observed higher than those of the TBH trading strategy using SeroFAM neuro-fuzzy network [45].

Table 6
Parameters Obtained by GA Optimisation.

Assets	GA-fMACDH				GA-MACDH			
	fast	*slow*	*signal*	α	*fast*	*slow*	*signal*	α
SPY	15	19	7	0.0018	11	49	9	0.0119
VGK	16	37	9	0.0075	22	32	17	0.0014
VWO	12	24	7	0.0040	26	43	17	0.0033
AGG	13	20	7	0.0010	9	29	7	0.0020
XAU	12	35	6	0.0062	15	27	5	0.0020
IYE	24	46	17	0.0088	7	42	4	0.0111

Table 7
Experiment Results Comparisons on Returns and Max Drawdown.

Asset	Return			Max Drawdown		
	GA-fMACDH (%)	GA-MACDH (%)	Buy and Hold (%)	GA-fMACDH (%)	GA-MACDH (%)	Buy and Hold (%)
SPY	**238.46**	8.99	189.92	23.61	**18.08**	34.10
VGK	**153.02**	15.55	23.71	**17.60**	32.59	41.57
VWO	**146.05**	72.35	8.15	28.66	**24.94**	43.7
AGG	9.09	7.79	**12.01**	**6.28**	6.99	9.58
XAU	**118.17**	60.68	23.82	**48.14**	56.25	83.04
IYE	**8.70**	−26.48	−47.18	58.70	**55.43**	78.21

Table 8
Improvements on Investment Returns.

Asset	FE-RNN based trading		TBH trading strategy [45]
	GA-fMACDH over GA-MACDH (%)	GA-fMACDH over Buy and Hold (%)	TBH over Buy and Hold (%)
SPY	**229.47**	48.54	7.59
VGK	**137.47**	129.31	49.13
VWO	73.70	**137.90**	31.46
AGG	1.30	−2.92	**45.41**
XAU	57.49	**94.35**	–
IYE	35.18	**55.88**	**–**

The fuzzy rules embedded within the FE-RNN system provide a method of interpreting the underlying deep RNN, since both embedded structures share the same vocabulary in the input and output fuzzy spaces. It allows the interpretation of the operations in the deep network structure during the encoding and decoding phases with the help of the embedded fuzzy structure. Hence, providing the much-needed transparency for the deep learning structure. The duality of the embedding also allows the deep structure to learn and recall the empirical data driven implication for the embedded fuzzy system. The FE-RNN assisted GA-fMACDH trading system illustrates promising results and would be a useful tool for analysts and investors.

Future works will be done for the deep structure of FE-RNN to improve the prediction capability of the system, such as using a convolutional neural network or performing pruning on the deep RNN. Other latest works in literature are reported on deep neural networks with improved interpretability. As a part of the future works, we will explore to learn and ensemble such deep neural networks into the architecture. Other research directions will be also explored for the development of an evolving neural network that enables FE-RNN as an online system. It could explore the tagging between fuzzy rules and the GRU nodes within the deep structure to gain a deeper and causal understanding of the encoding and decoding of the deep structure. Currently the interpretability of FE-RNN is only explained and measured in qualitative terms. It will be an important future work to evaluate and benchmark the interpretability in quantitative terms. It could also explore the capabilities of FE-RNN within other forecast assisted trading strategies using other technical indicators optimised by evolutionary algorithms, such as the Relative Strength Index, Price Percentage Oscillator, and Bollinger Bands. The system will be assessed by a wider range of financial market assets including ETF, stocks, or market indexes.

CRediT authorship contribution statement

James Chee Min Tan: Conceptualization, Investigation, Methodology, Writing – review & editing. **Qi Cao:** Conceptualization, Investigation, Methodology, 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] A. Adadi, M. Berrada, Peeking inside the black-box: a survey on explainable artificial intelligence (XAI), IEEE Access (2018) 52138–52160.
[2] M. Akhmet, K. Başkan, C. Yeşil, Revealing chaos synchronization below the threshold in coupled Mackey-Glass systems, Mathematics 11 (2023) 3197.
[3] M. Alateeq, W. Pedrycz, Development of two-phase logic-oriented fuzzy AND/OR network, Neurocomputing 482 (2022) 129–138.

[4] A.B. Arrieta, N. Díaz-Rodríguez, J.D. Ser, et al., Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI, Information Fusion (2020) 82–115.
[5] N. Bacanin, L. Jovanovic, M. Zivkovic, et al., Multivariate energy forecasting via metaheuristic tuned long-short term memory and gated recurrent unit neural networks, Information Sciences 642 (2023).
[6] O. Castillo, J.R. Castro, P. Melin, Interval type-3 fuzzy aggregation of neural networks for multiple time series prediction: The case of financial forecasting, Axioms 11 (2022) 251.
[7] H. Das, B. Naik, H.S. Behera, A hybrid neuro-fuzzy and feature reduction model for classification, Advances in Fuzzy Systems (2020).
[8] R.T. Das, K.K. Ang, C. Quek, ieRSPOP: A novel incremental rough set-based pseudo outer-product with ensemble learning, Applied Soft Computing (2016) 170–186.
[9] C. Deng, Y. Huang, N. Hasan, Y. Bao, Multi-step-ahead stock price index forecasting using long short-term memory model with multivariate empirical mode decomposition, Information Sciences 607 (2022) 297–321.
[10] W. Ding, M. Abdel-Basset, H. Hawash, A.M. Ali, Explainability of artificial intelligence methods, applications and challenges: A comprehensive survey, Information Sciences 615 (2022) 238–292.
[11] M.M. Ferdaus, R.K. Chakrabortty, M.J. Ryan, Multiobjective automated type-2 parsimonious learning machine to forecast time-varying stock indices online, IEEE Transactions on Systems, Man, and Cybernetics: Systems (2022) 2874–2887.
[12] R. Gao, S. Cui, H. Xiao, W. Fan, H. Zhang, Y. Wang, Integrating the sentiments of multiple news providers for stock market index movement prediction: A deep learning approach based on evidential reasoning rule, Information Sciences 615 (2022) 529–556.
[13] S. Hašková, P. Šuler, R. Kuchár, A fuzzy multi-criteria evaluation system for share price prediction: A tesla case study, Mathematics 11 (2023).
[14] A.R. Iyer, D.K. Prasad, C. Quek, PIE-RSPOP: A brain-inspired pseudo-incremental ensemble rough set pseudo-outer product fuzzy neural network, Expert Systems with Applications (2018) 172–189.
[15] S.B. Jabeur, V. Serret, Bankruptcy prediction using fuzzy convolutional neural networks, Research in International Business and Finance 64 (2023).
[16] J. Jang, ANFIS: adaptive-network-based fuzzy inference system, IEEE transactions on systems, man, and cybernetics, (1993) 665-685.
[17] M.J. Jiménez-Navarro, M. Martínez-Ballesteros, F. Martínez-Álvarez, G. Asencio-Cortés, PHILNet: A novel efficient approach for time series forecasting using deep learning, Information Sciences 632 (2023) 815–832.
[18] A.F. Kamara, E. Chen, Z. Pan, An ensemble of a boosted hybrid of deep learning models and technical analysis for forecasting stock prices, Information Sciences 594 (2022) 1–19.
[19] N. Kasabov, Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning, IEEE Transactions on Systems, Man, and Cybernetics, Part B (cybernetics) (2001) 902–918.
[20] N.K. Kasabov, Q. Song, DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction, IEEE Transactions on Fuzzy Systems (2002) 144–154.
[21] M. Keshk, N. Koroniotis, N. Pham, N. Moustafa, B. Turnbull, A.Y. Zomaya, An explainable deep learning-enabled intrusion detection framework in IoT networks, Information Sciences 639 (2023).
[22] 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.
[23] J. Liu, T. Zhao, J. Cao, P. Li, Interval type-2 fuzzy neural networks with asymmetric MFs based on the twice optimization algorithm for nonlinear system identification, Information Sciences 629 (2023) 123–143.
[24] Y. Liu, X. Lu, W. Peng, C. Li, H. Wang, Compression and regularized optimization of modules stacked residual deep fuzzy system with application to time series prediction, Information Sciences 608 (2022) 551–577.
[25] M. Lu, X. Xu, TRNN: An efficient time-series recurrent neural network for stock price prediction, Information Sciences 657 (2024).
[26] X. Meng, Y. Zhang, L. Quan, J. Qiao, A self-organizing fuzzy neural network with hybrid learning algorithm for nonlinear system modeling, Information Sciences 642 (2023).
[27] T. Molnar, Interpretable machine learning – A guide for making black box models explainable, Independently Published ISBN-13 (2022) 979–8411463330.
[28] H. Nasiri, M.M. Ebadzadeh, MFRFNN: Multi-functional recurrent fuzzy neural network for chaotic time series prediction, Neurocomputing 507 (2022) 292–310.
[29] K.K. Patro, J.P. Allam, B.C. Neelapu, R. Tadeusiewicz, U.R. Acharya, M. Hammad, O. Yildirim, P. Pławiak, Application of Kronecker convolutions in deep learning technique for automated detection of kidney stones with coronal CT images, Information Sciences 640 (2023).
[30] 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, Information Sciences 649 (2023).
[31] H. Rafiei, M.-R. Akbarzadeh-T, Reliable Fuzzy Neural Networks for Systems Identification and Control, IEEE Transactions on Fuzzy Systems 31 (2023) 2251–2263.
[32] J.N. Reimann, A. Schwung, S.X. Ding, Neural logic rule layers, Information Sciences 596 (2022) 185–201.
[33] X. Shen, Q. Dai, W. Ullah, An active learning-based incremental deep-broad learning algorithm for unbalanced time series prediction, Information Sciences 642 (2023).
[34] M. Song, Y. Li, W. Pedrycz, Time series prediction with granular neural networks, Neurocomputing 546 (2023).
[35] P.V.d.C. Souza, E. Lughofer, H.R. Batista, An explainable evolving fuzzy neural network to predict the k barriers for intrusion detection using a wireless sensor network, Sensors 22 (2022).
[36] T. Szandała, Unlocking the black box of CNNs: Visualising the decision-making process with PRISM, Information Sciences 642 (2023).
[37] 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, Artificial Intelligence Review 56 (2023) 865–913.
[38] J. Tan, W.J. Zhou, C. Quek, Trading model: Self reorganizing Fuzzy Associative Machine-forecasted MACD-Histogram (SeroFAM-fMACDH). In: International Joint Conference on Neural Networks, 2015.
[39] S.W. Tung, C. Quek, C. Guan, SaFIN: A self-adaptive fuzzy inference network, IEEE Transactions on Neural Networks (2011) 1928–1940.
[40] R.L. Ulloa-Cazarez, N. García-Díaz, L. Soriano-Equigua, Multi-layer adaptive fuzzy inference system for predicting student performance in online higher education, IEEE Latin America Transactions (2021) 98–106.
[41] C. Wang, X. Lv, M. Shao, Y. Qian, Y. Zhang, A novel fuzzy hierarchical fusion attention convolution neural network for medical image super-resolution reconstruction, Information Sciences 622 (2023) 424–436.
[42] Y. Wang, H. Ishibuchi, M.J. Er, J. Zhu, Unsupervised multilayer fuzzy neural networks for image clustering, Information Sciences 622 (2023) 682–709.
[43] D. Wu, A. Lisser, CCGnet: A deep learning approach to predict Nash equilibrium of chance-constrained games, Information Sciences 627 (2023) 20–33.
[44] B. Xu, S. Li, A.A. Razzaqi, Y. Guo, L. Wang, A novel measurement information anomaly detection method for cooperative localization, IEEE Transactions on Instrumentation and Measurement (2021) 1–18.
[45] L.L.X. Yeo, Q. Cao, C. Quek, Dynamic portfolio rebalancing with lag-optimised trading indicators using SeroFAM and genetic algorithms, Expert Systems with Applications 216 (2023) 1–18.
[46] K.K. Yun, S.W. Yoon, D. Won, Interpretable stock price forecasting model using genetic algorithm-machine learning regressions and best feature subset selection, Expert Systems with Applications 213 (2023).
[47] B. Zhang, X. Gong, J. Wang, F. Tang, K. Zhang, W. Wu, Nonstationary fuzzy neural network based on FCMnet clustering and a modified CG method with Armijo-type rule, Information Sciences 608 (2022) 313–338.
[48] K. Zheng, Q. Zhang, L. Peng, S. Zeng, Adaptive memetic differential evolution-back propagation-fuzzy neural network algorithm for robot control, Information Sciences 637 (2023).
[49] Y. Zheng, Z. Xu, X. Wang, The fusion of deep learning and fuzzy systems: A state-of-the-art survey, IEEE Transactions on Fuzzy Systems 30 (2022) 2783–2799.
[50] R.W. Zhou, C. Quek, POPFNN: A pseudo outer-product based fuzzy neural network, Neural Networks (1996) 1569–1581.