**QUEEN'S
UNIVERSITY
BELFAST**

A probabilistic, recurrent, fuzzy neural network for processing noisy time-series data

Li, Y., Gault, R., & McGinnity, T. M. (2022). A probabilistic, recurrent, fuzzy neural network for processing noisy time-series data. *IEEE Transactions on Neural Networks and Learning Systems*, 33(9), 4851 - 4860.
<https://doi.org/10.1109/TNNLS.2021.3061432>

Published in:
IEEE Transactions on Neural Networks and Learning Systems

Document Version:
Peer reviewed version

Queen's University Belfast - Research Portal:
[Link to publication record in Queen's University Belfast Research Portal](#)

Publisher rights
Copyright 2021 IEEE.
This work is made available online in accordance with the publisher's policies. Please refer to any applicable terms of use of the publisher.
No embargo on AM.

General rights
Copyright for the publications made accessible via the Queen's University Belfast Research Portal is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The Research Portal is Queen's institutional repository that provides access to Queen's research output. Every effort has been made to ensure that content in the Research Portal does not infringe any person's rights, or applicable UK laws. If you discover content in the Research Portal that you believe breaches copyright or violates any law, please contact openaccess@qub.ac.uk.

A probabilistic, recurrent, fuzzy neural network for processing noisy time-series data

Yong Li, Richard Gault and T. Martin McGinnity, *Senior Member, IEEE*

Abstract—The rapidly increasing volumes of data and the need for big data analytics have emphasised the need for algorithms that can accommodate incomplete or noisy data. The concept of recurrency is an important aspect of signal processing, providing greater robustness and accuracy in many situations, such as biological signal processing. Probabilistic fuzzy neural networks (PFNN) have shown potential in dealing with uncertainties associated with both stochastic and non-stochastic noise simultaneously. Previous research work on this topic has addressed either the fuzzy-neural aspects or alternatively the probabilistic aspects, but there currently does not exist a probabilistic fuzzy neural algorithm with recurrent feedback.

In this paper, a probabilistic fuzzy neural network with a recurrent probabilistic generation module (designated PFNN-R) is proposed to enhance and extend the ability of the PFNN to accommodate noisy data. A back-propagation based mechanism, which is used to shape the distribution of the probabilistic density function of the fuzzy membership, is also developed. The motivation of the work was to develop an approach that provides an enhanced capability to accommodate various types of noisy data. We apply the algorithm to a number of benchmark problems and demonstrate via simulation results that the proposed technique incorporating recurrency advances the ability of probabilistic fuzzy neural networks to model time-series data with high intensity, random noise.

Index Terms — Computational neuroscience, neural network, probabilistic fuzzy system, recurrent.

I. INTRODUCTION

ARTIFICIAL neural networks are a powerful tool for modelling based on data. They have already been applied to problems of regression, classification, computational neuroscience, computer vision, data processing, and time series analysis. Recent developments in the field of artificial intelligence have led to a renewed interest in neural network research, but the problem of how to deal with uncertainty

remains an extremely important factor when modelling using neural networks. There are two categories of uncertainties. One is the non-stochastic uncertainties related to uncertainties or ambiguity in describing the fact. The other one is uncertainties about training data with random noise which is referred to as stochastic uncertainty. An effective method of tackling uncertainties is to use probabilistic techniques for stochastic uncertainties and fuzzy techniques for non-stochastic uncertainties. Effective methods for integrating probabilistic and fuzzy techniques into neural networks remain an active research topic.

Fuzzy systems and their learning variants, fuzzy neural systems (FNN), have proven to be quite effective in addressing non-stochastic uncertainties [1], [2], [3], [4], [5], [6]. In addition, a number of authors have extended and improved the FNN by the incorporation of a recurrent component into the algorithm, based on recurrent neural networks (RNN). RNN provide a very elegant way of dealing with (time) sequential data that embody correlations between data points that are close in a sequence [7]; which makes them applicable to handle stochastic noise with high efficiency. Separately, Li et al [8], [9] addressed stochastic uncertainties by incorporating a probabilistic component into the FNN to produce a probabilistic fuzzy neural network, but without recurrency. Despite the potential of recurrent networks, research to date has not addressed the challenge of integrating the three concepts of probability, fuzzy neural systems, and recurrency to significantly improve an algorithm's ability to deal with noisy data. Such an approach would be expected to optimize the algorithm's ability to deal effectively with both stochastic and non-stochastic uncertainties, so prevalent in many datasets.

The motivation of the work reported herein is to contribute to the development of effective solutions to the problems of processing stochastic and non-stochastic uncertainties in noisy time-series data by integrating fuzzy neural, recurrent and probabilistic approaches into a single algorithm. Building on the work of Li et al [8], [9], we incorporate a recurrent feedback module for enhanced stochastic and non-stochastic noise performance. Furthermore, we address one of the weaknesses in the original PFNN in terms of the generation of a more accurate probabilistic density function.

We evaluate the approach on three simulation challenges of increasing complexity, namely a numerical periodic function test; the Mackey-Glass time series with random noise; and twelve datasets from the M3 time-series prediction competition [10].

Manuscript received April 17, 2020; ; This work is partially supported by the National Natural Science Foundation of China (NSFC) under grant 61906125.

Yong Li is with School of Electrical Engineering Shenyang University of Technology, Shenyang, 110870, China, and is also with the Intelligent Systems Research Centre, Ulster University, Magee Campus, Londonderry BT48 7JL, U.K. (e-mail:liyong@sut.edu.cn).

T. M. McGinnity is with the Intelligent Systems Research Centre, Ulster University, Magee Campus, Londonderry BT48 7JL, U.K., and is also with the Department of Computer Science, Nottingham Trent University, NG11 8NS Nottingham, U.K. (e-mail: tm.mcginny@ulster.ac.uk).

Richard Gault is with the School of Electronics, Electrical Engineering and Computer Science, Queen's University, Belfast, 18 Malone Road, BT9 6RT, Belfast, U.K. (e-mail: richard.gault@qub.ac.uk).

The remainder of this paper is structured into four sections, as follows. Section II summarises the related work and the rationale for the contribution. Section III begins by laying out the framework of the modified neural network and gives details about the Recurrent Probabilistic Generation module and parameter learning. Section IV presents simulations, results and analysis and Section V presents the conclusions of the paper.

II. RELATED WORK

S. Horikawa proposed three types of fuzzy neural networks (FNNs) namely Type-I, Type-II and Type-III in their research work in the 1990s [11]–[13] and integrated the back propagation algorithm into them [14]. The FNNs can identify the fuzzy model of a nonlinear system automatically. Subsequently, many modifications were made to the basic FNN [15]–[18], with a substantial range of applications addressed. Recent papers include [19], where the FNN structure is applied to identify the unknown plant model of a constrained robot interacting with its environment. Simulations illustrated the effectiveness of the proposed scheme. A new method based on the particle swarm optimization algorithm and a Type-II FNN was proposed in [20]. The method combines the fuzzy system's expert knowledge and the neural network's learning capability for accurate wind power forecasting. Sun et al [21] developed a novel force observer using a Type-II Fuzzy Neural Network based Moving Horizon Estimation, to estimate external force/torque information and simultaneously filter out the system disturbances. Together, these studies indicate that FNNs demonstrate good performance when dealing with non-stochastic uncertainties.

Specht proposed a probabilistic neural network (PNN) by replacing the sigmoid activation function often used in neural networks with an exponential function [22]. PNNs are widely applied to many fields such as earthquake magnitude prediction [23], estimation of battery state of health [24], brain tumour identification and classification [25], [26], bleeding detection in wireless capsule endoscopy [27], indoor sound source localization [28], and many others. It may be concluded that stochastic uncertainties are well solved via PNNs. However, when both types of uncertainty exist concurrently, as is often the case, a probabilistic fuzzy system (PFS) approach may be a more effective solution.

Essentially, the PFS is a methodology that is built on a fuzzy inference system (FIS), which has been modified to accommodate a probabilistic fuzzy rule base. Zadeh [29] proposed the original model and it has been applied in many areas, such as finance and capital markets. Den Berg [30] concentrated on the probabilistic Takagi–Sugeno fuzzy system and their design for financial markets data. Their results show that fuzzy and probabilistic uncertainties can be simultaneously addressed. In [31], a probabilistic fuzzy logic system was proposed for modelling control problems. It is applied to a function approximation problem as well as a robotic system and shows a better performance than an ordinary fuzzy logic system under stochastic circumstances. However, the fuzzy membership function and the probabilistic function in [30], [31] are both acquired from data based on an analytical method and

there is no integrated learning mechanism. Li proposed a probabilistic fuzzy neural network (PFNN) to handle complex stochastic uncertainties in [8] and extended this in [9] into a classification framework for both stochastic and fuzzy uncertainties. The learning mechanism is incorporated via the neural network framework. The vagueness was handled by the fuzzy system, randomness by the embedded probabilistic function, and time-dependent variations by neural learning, respectively. Crucially there was no utilization of recurrent feedback which would be expected to enhance the performance.

In [8] and [9] a fuzzy c-mean algorithm is used in the fuzzy membership function (MF) modelling. The fuzzy membership U_{MF} of the crisp value for input x_c is calculated based on the MF. Li et al were doing this initially from the generation of M items in the neighbourhood of x_c and then feeding the $M+1$ items to the MF. Notably, U_{MF} is not a scalar but a vector with $M+1$ dimensions. Different probabilities are attached to the $M+1$ dimensions according to the probabilistic density function (PDF). In the equation, the estimated variance is adjusted using the SPC-based variance monitoring of the input data. A successful simulation demonstrated the modelling and classification effectiveness.

The advantages of incorporating recurrent feedback into approaches for data analytics has been demonstrated in a number of papers and algorithms. These include the Nonlinear Autoregressive models with exogenous input Neural Network (NARX) [32]; the Simple Recurrent Network (SRN) [33]; Echo State Networks (ESN) [34]; and Long-Short Term Memory (LSTM) [35]. The Multi Recurrent Network (MRN) was discussed by Ulbricht [36] and more recently Tepper et al. [37] showed that the MRN dynamics improved learning and achieved better accuracy (compared to the SRN, NARX and ESN networks).

Crucially there is no integrated approach that incorporates the concepts of fuzzification, probability, neural network learning, and time-dependent recurrency into a single algorithmic approach. Based on the literature of recurrent fuzzy neural systems such an integrated algorithm would be expected to enhance the performance in dealing with stochastic and non-stochastic noisy and incomplete datasets. To address this, we propose a recurrent, probabilistic fuzzy-neural system in this paper by developing a recurrent probabilistic generation module and extending the work of Li et al [8]. Training data across a specific time span are selected as the input for the recurrent module. These are fed into a modified LSTM framework to generate the probability and a mechanism to guarantee the position of the largest probability is presented. Then the weights of the LSTM are updated by the output. We hypothesise that a more accurate PDF may be constructed if both the input and output data are considered.

Furthermore, we address the issue that in [8] and [9] there is no mechanism for generating the M items in the neighbourhood of x_c ; this remains a significant issue in the algorithm design. In our approach, we calculate the output errors of the distances between the M items and corresponding x_c using the back-propagation algorithm and use these to adjust the position of the M items during the training process.

We demonstrate that the approach is both feasible and effective, by first presenting simulation results from a function approximation problem and then a Mackey-Glass time series prediction task. Finally, we present results from applying the algorithm to twelve datasets from the M3 time series prediction competition and show overall significantly improved performance.

III. DESIGN OF THE PROBABILISTIC FUZZY NEURAL NETWORK WITH RECURRENT PROBABILISTIC GENERATION MODULE

A. Probabilistic Fuzzy Logic System

The standard fuzzy logic system (FLS) can be expressed as a group of fuzzy rules. For example, the j^{th} fuzzy rule can be expressed as: if x_1 is $I_{1,j}$ and... x_i is $I_{i,j}$... and x_n is $I_{n,j}$ then y is O_j , where $x_i (i \in 1 \sim n)$ is the i^{th} dimension of a certain input. $I_{i,j} (i \in 1 \sim n)$ and O_j correspond to the fuzzy sets in the antecedent and consequent part of the model respectively and y is the corresponding output. In contrast, the j^{th} rule of a PFS is as follows: if x_1 is $\bar{I}_{1,j}$ and... x_i is $\bar{I}_{i,j}$... and x_n is $\bar{I}_{n,j}$ then y is $\bar{O}_j$. In this case $\bar{I}_{i,j} (i \in 1 \sim n)$ and $\bar{O}_j$ are the probabilistic fuzzy sets of the antecedent and consequent parts respectively [31].

There are many similarities between the FLS and PFS. The main difference is that in a PFS, a crisp input is scattered across several MFs with different probabilities. For example, in the FLS case, when $x=1$, the MF could be $u=0.9$ (according to the particular fuzzy membership function) with the probability $p=1$. In comparison, for the PFS when $x=1$, the MF is not just a single number but a set of numbers generated around 0.9. Alternatively, we could have an MF $u_1=0.9$ with the probability p_1 , MF $u_2=0.85$ with the probability p_2 and MF $u_3=0.94$ with the probability p_3 . Obviously, the sum of probabilities for all the MF of x should equal to 1, $\sum_{k=1}^3 p_k = 1$.

B. Framework of The Modified PFNN To Incorporate Recurrency (PFNN-R)

The overall framework of the probabilistic fuzzy neural network with a recurrent probabilistic generation module (PFNN_R) is shown in Figure 1, with the novel component indicated by the dashed block labelled Recurrent Probabilistic Generator. The PFNN_R utilizes recurrency information via feedback from the system output error and the input data variation, to achieve an adaptive adjustment of the PDF, and other parameters. Specifically, the PDF in PFNN_R is generated based on the extracted information from the input data and the output of the PFNN_R via a long short term memory recurrent mechanism; an appropriate update algorithm and its implementation frequency for different parameters in PFNN_R is proposed, based on error back-propagation and output error surveillance.

The network has seven layers in total. The input data denoted by x_i represents the i^{th} input and $x_{i,i}$ is the i^{th} element of x_i .

Other parameters are defined as follows: n and $\bar{J}$ are the number of input dimensions and fuzzy rules respectively. The size of the training data is q . B_{delta} is the number of inputs for the recurrent probabilistic generator module. y_i is the output with respect to x_i and y_i^e is the corresponding output of the PFNN_R.

The basic idea is to take a specific quantity of the historical input data preceding the current trained input data to generate the probability for the third layer of the network based on the long-short term memory (LSTM) approach [38]. The rest of this section will present the details of the novel recurrent probability generator (RPG) module and the parameter learning through the training of PFNN_R. More details of the fuzzy and probabilistic inference aspects can be found in [8]. The framework for one learning iteration can be described using the pseudo-code presented in Algorithm 1.

Figure 1: The framework of PFNN_R (adapted from Li et al [8])

one learning iteration

```

for t=1:q
  feed  $x_t=[x_{t,1}, x_{t,2}, \dots, x_{t,i}, \dots, x_{t,n}]$  to the input layer of PFNN_R
  for j=1:  $\bar{J}$ 
    set  $M_j=[m_{j,1}, m_{j,2}, \dots, m_{j,i}, \dots, m_{j,n}]$ 
    randomly generate  $\sigma_j=[\sigma_{j,1}, \sigma_{j,2}, \dots, \sigma_{j,i}, \dots, \sigma_{j,n}]$  where  $\sigma_{j,i}=[\sigma_{j,i,1}, \sigma_{j,i,2}, \dots, \sigma_{j,i,k}, \dots, \sigma_{j,i,m_{ji}}]$ 
    compute  $\tilde{x}_{t,j}=[\tilde{x}_{t,j,1}, \tilde{x}_{t,j,2}, \dots, \tilde{x}_{t,j,i}, \dots, \tilde{x}_{t,j,n}]$ 
    where  $\tilde{x}_{t,j,i}=[x_{t,i} + \sigma_{j,i,1}, x_{t,i} + \sigma_{j,i,2}, \dots, x_{t,i} + \sigma_{j,i,k}, \dots, x_{t,i} + \sigma_{j,i,m_{ji}}]$ 
    calculate the MF  $U_{t,j}=[U_{t,j,1}, U_{t,j,2}, \dots, U_{t,j,i}, \dots, U_{t,j,n}]$ 
    where  $U_{t,j,i}=[u_{t,j,i,1}, u_{t,j,i,2}, \dots, u_{t,j,i,k}, \dots, u_{t,j,i,m_{ji}}]=F_{MF}(\tilde{x}_{t,j,i})$   $F_{MF}(\bullet)$  is the fuzzy membership function
    =====RPG=====
    generate the probability using recurrent probability generator module
    =====RPG=====
  end
  Do the fuzzy and probabilistic inference then calculate  $y_t^e$  using output weight  $W_o=[w_{o,1}, \dots, w_{o,j}, \dots, w_{o,J}]^T$ .
  Compute the error using  $y_t^e$  and  $y_t$ 
  end
  =====UPDATE=====
  Update the corresponded parameters based on error.
  =====UPDATE=====

```

Algorithm 1 Pseudo-code outlining the parameter learning algorithm**C. Recurrent Probability Generator Module**

The pseudo code for the recurrent probability generator module, namely the top dashed block in Figure 1, is given in Algorithm 2.

```

=====RPG=====
if the number of inputs  $< B_{\delta}$ 
  calculate the average value from  $U_{1,j}$  to  $U_{t,j}$ , dubbed as  $\bar{U}_{t,j}$ , Duplicate  $\bar{U}_{t,j}$  for  $B_{\delta}$  times and feed them to the recurrent probability generator module
otherwise
  Feed  $U_{t-B_{\delta}+1,j}$  to  $U_{t,j}$ , to the recurrent probability generator module
end
generate the probability using recurrent probability generator module
 $Prob_{t,j}=[Prob_{t,j,1}, Prob_{t,j,2}, \dots, Prob_{t,j,i}, \dots, Prob_{t,j,n}]$ 
where
 $Prob_{t,j,i}=[prob_{j,i,1}(t), prob_{j,i,2}(t), \dots, prob_{j,i,k}(t), \dots, prob_{j,i,m_{ji}}(t)]$ 
=====RPG=====

```

Algorithm 2 Overview of the RPG module computation

Notice the i^{th} component of $U_{t,j}$ is $U_{t,j,i}=[u_{t,j,i,1}, u_{t,j,i,2}, \dots, u_{t,j,i,k}, \dots, u_{t,j,i,m_{ji}}]$ and is also a vector. The details of how to compute $Prob_{t,j,i}$ from $U_{t,j,i}$ are illustrated in Figure 2 by taking $u_{t,j,i,k} \rightarrow prob_{j,i,k}(t)$ as an instance.

The RPG utilizes the B_{δ} input terms up to and including the current training iteration t and the intermediate probability $prob_temp_{j,i,k}(t-1)$ of iteration $t-1$ as input. These are appropriately weighted and summed. The weights are updated using the output error to calculate the four gates namely at , it , ot and ft (see Figure 2). So, both the input variation and the output of iteration $t-1$ contribute to generating the probability for the current iteration t . The details are presented as follows.

In Figure 2, $W_{label}^{j,i}$ are the $1 * B_{\delta}$ vectors and $u_{label}^{j,i}$ are scalars, where $label \in [at, it, ft, ot]$. Generation of the probability for the remainder of the dimension of $U_{t,j,i}$ is found in the same manner as for $u_{t,j,i,k}$. The weights for the same dimension of input and fuzzy rules are also the same, so there is no $W_{label}^{i,j,k}$ and $u_{label}^{i,j,k}$. The RPG for the i^{th} dimension of the input scattered in the j^{th} fuzzy rule is marked as $RPG_{j,i}$, $i \in 1 \sim n, j \in 1 \sim \bar{J}$. So the total number of RPG inside PFNN_R equals $n * \bar{J}$.

Figure 2: Recurrent probability generator module.

In the t^{th} training iteration, $RPG_{j,i}$ will generate probabilities $Prob_{t,j,i}$ with $m_{j,i}$ items using:

$$U_{input}^{t,j,i}(k) = [u_{t-B_{delta}+1,j,i,k}, \dots, u_{t,j,i,k}], \quad k = 1 \sim m_{j,i} \text{ or } \bar{U}_{t,j,i}.$$

To ensure the crisp value appears in $\tilde{x}_{t,j,i}$, the first dimension of $\sigma_{j,i}$ is set to zero. Namely $\sigma_{j,i} = [0, \sigma_{j,i,2}, \dots, \sigma_{j,i,k}, \dots, \sigma_{j,i,m_{j,i}}]$ for all the i and j . The $prob_temp_{j,i,1}(t)$ is set to a fixed value marked as $prob_{j,i}^{max}$. This ensures that the MF value for the crisp input will be linked to the largest probability, which is obviously reasonable. The rest of $prob_temp_{j,i,k}(t)$, $k = 2 \sim m_{j,i}$ are generated using the following equations.

$$at_{j,i}^k(t) = \frac{1}{p_{at} + e^{-(W_{at}^{j,i} * U_{input}^{t,j,i}(k) + u_{at}^{j,i} * prob_temp_{j,i,k}(t-1) + B_{at})}} \quad (1)$$

$$it_{j,i}^k(t) = \frac{1}{1 + e^{-(W_{it}^{j,i} * U_{input}^{t,j,i}(k) + u_{it}^{j,i} * prob_temp_{j,i,k}(t-1) + B_{it})}} \quad (2)$$

$$ft_{j,i}^k(t) = \frac{1}{1 + e^{-(W_{ft}^{j,i} * U_{input}^{t,j,i}(k) + u_{ft}^{j,i} * prob_temp_{j,i,k}(t-1) + B_{ft})}} \quad (3)$$

$$ot_{j,i}^k(t) = \frac{1}{1 + e^{-(W_{ot}^{j,i} * U_{input}^{t,j,i}(k) + u_{ot}^{j,i} * prob_temp_{j,i,k}(t-1) + B_{ot})}} \quad (4)$$

$$state_{j,i}^k(t) = at_{j,i}^k(t) * it_{j,i}^k(t) + ft_{j,i}^k(t) * state_{j,i}^k(t-1) \quad (5)$$

$$prob_temp_{j,i,k}(t) = \frac{1}{p_{out} + e^{-state_{j,i}^k(t)}} * ot_{j,i}^k(t) \quad (6)$$

where B_{at} , B_{it} , B_{ft} , B_{ot} , p_{at} , p_{out} , are all preset basis parameters. Finally, the output of the $RPG_{j,i}$ $Prob_{t,j,i} =$

$[prob_{j,i,1}(t), \dots, prob_{j,i,k}(t), \dots, prob_{j,i,m_{j,i}}(t)]$ is available, where:

$$prob_{j,i,k}(t) = \frac{prob_temp_{j,i,k}(t)}{\sum_{k=1}^{m_{j,i}} prob_temp_{j,i,k}(t)} \quad (7)$$

D. Parameter Learning

The design methodology for M_j , the primary MF and the related parameter learning algorithm through training are the same as in [8] and accordingly no further discussion on these topics is provided here. In summary, we define the performance criteria $L_t^c = (1/2)e_t^2$ with the modelling error e_t as the difference between y_t and y_t^c . The learning rules are derived from minimizing the performance criteria L_t^c . The details of the remaining aspects of the parameter learning, based on the gradient descent algorithm, are presented in [8].

1) Adaptation for $\sigma_{j,i}$ and $prob_{j,i}^{max}$

Both $\sigma_{j,i} = [0, \sigma_{j,i,2}, \dots, \sigma_{j,i,k}, \dots, \sigma_{j,i,m_{j,i}}]$ and $prob_{j,i}^{max}$ are very important during the generation of the probabilities. That is because, first of all, the distribution of input for $RPG_{j,i}$ is determined by $\sigma_{j,i}$. Secondly, the biggest dimension of $Prob_{t,j,i}$ is decided by $prob_{j,i}^{max}$, so they should be updated during training. The learning rules for $\sigma_{j,i}$ and $prob_{j,i}^{max}$ are developed as

$$\sigma_{j,i}^{t+1} = \sigma_{j,i}^t - \eta_l \frac{1}{q} \sum_{t=1}^q \frac{\partial L_t^c}{\partial \sigma_{j,i}^t} \quad (8)$$

$$prob_{j,i}^{max,t+1} = prob_{j,i}^{max,t} - \eta_l \frac{1}{q} \sum_{t=1}^q \frac{\partial L_t^c}{\partial prob_{j,i}^{max,t}} \quad (9)$$

where $\eta_l > 0$ is the learning rate, $\frac{\partial L_t^c}{\partial \sigma_{j,i}}$, and $\frac{\partial L_t^c}{\partial prob_{j,i}^{max}}$ are

calculated as follows:

$$\frac{\partial L_t^c}{\partial \sigma_{j,i}} = \frac{\partial L_t^c}{\partial y_t^c} \times \frac{\partial y_t^c}{\partial \phi_j^t} \times \frac{\partial \phi_j^t}{\partial \sigma_{j,i}} \times \frac{\partial \phi_j}{\partial \sigma_{j,i}} \quad (10)$$

$$L_t^c = \frac{1}{2} (y_t^c - y_t)^2 \quad (11)$$

$$\text{so } \frac{\partial L_t^c}{\partial y_t^c} = \frac{1}{2} \times 2(y_t^c - y_t) \times (-1) = (y_t^c - y_t) \quad (12)$$

$$y_t^c = \sum_{j=1}^{\bar{J}} w_{o,j} \phi_j^t \quad \text{so } \frac{\partial y_t^c}{\partial \phi_j^t} = w_{o,j} \quad (13)$$

$$\phi_j^t = \frac{\phi_j}{\sum_{j=1}^{\bar{J}} \phi_j} \quad \text{so } \frac{\partial \phi_j^t}{\partial \phi_j} = \left(\sum_{j=1}^{\bar{J}} \phi_j - \phi_j \right) / \left(\sum_{j=1}^{\bar{J}} \phi_j \right)^2 \quad (14)$$

Setting

$$\begin{aligned} bp_j^{com} &= \frac{\partial L_t^c}{\partial y_t^c} \times \frac{\partial y_t^c}{\partial \phi_j^t} \times \frac{\partial \phi_j^t}{\partial \phi_j} \\ &= (y_t^c - y_t) w_{o,j} \left(\sum_{j=1}^{\bar{J}} \phi_j - \phi_j \right) / \left(\sum_{j=1}^{\bar{J}} \phi_j \right)^2 \end{aligned} \quad (15)$$

then

$$\frac{\partial L_t^c}{\partial \sigma_{j,i}} = bp_j^{com} \frac{\partial \phi_j}{\partial \sigma_{j,i}} \quad \text{and} \quad \frac{\partial L_t^c}{\partial prob_{j,i}^{\max}} = bp_j^{com} \frac{\partial \phi_j}{\partial prob_{j,i}^{\max}} \quad (16)$$

2) Adaptation for $W_{label}^{j,i}$ and $u_{label}^{j,i}$ in $RPG_{j,i}$

$W_{label}^{j,i}$ and $u_{label}^{j,i}$ are weights inside the $RPG_{j,i}$ (recalling that $label \in [at, it, ft, ot]$). The root mean square error (RMSE) is utilised as the performance criteria. To minimize this metric, we develop the following learning rule:

$$W_{label}^{j,i,t+1} = W_{label}^{j,i,t} - \eta_2 \frac{1}{q} \sum_{t=1}^q \frac{\partial L_t^c}{\partial W_{label}^{j,i}} \quad (17)$$

$$u_{label}^{j,i,t+1} = u_{label}^{j,i,t} - \eta_3 \frac{1}{q} \sum_{t=1}^q \frac{\partial L_t^c}{\partial u_{label}^{j,i}} \quad (18)$$

$$\frac{\partial L_t^c}{\partial W_{label}^{j,i}} = bp_j^{com} \frac{\partial \phi_j}{\partial W_{label}^{j,i}} \quad \text{and} \quad \frac{\partial L_t^c}{\partial u_{label}^{j,i}} = bp_j^{com} \frac{\partial \phi_j}{\partial u_{label}^{j,i}} \quad (19)$$

where η_2 and η_3 are the learning rate for $W_{label}^{j,i}$ and $u_{label}^{j,i}$ respectively. $W_{label}^{j,i}$. The learning rates are not necessarily the same because the influence of input and preceding output probability are generally different. The parameters $\sigma_{j,i}$, $prob_{j,i}^{\max}$, $W_{label}^{j,i}$ and $u_{label}^{j,i}$ are impacted by the probability generated by $RPG_{j,i}$. As discussed in the next section $\sigma_{j,i}$, $W_{label}^{j,i}$, $u_{label}^{j,i}$ cannot be updated simultaneously during training, as this could lead to instabilities in RMSE. Hence the

update of $\sigma_{j,i}$ and the adjustment of $W_{label}^{j,i}$ and $u_{label}^{j,i}$ are carried out iteratively during training.

3) Adaptation for W_o

W_o is the output weight and is initialized by the method in [8]. During training, while the RMSE is not decreasing, W_o is updated using the following equation.

$$W_o = [w_{o,1}, \dots, w_{o,j}, \dots, w_{o,J}]^T = (\phi^T \phi)^{-1} \phi^T Y^T \quad (20)$$

where $\phi = [\phi^1, \dots, \phi^t, \dots, \phi^q]^T$, $\phi^t = [\phi_1^t, \dots, \phi_j^t, \dots, \phi_J^t]$ and $Y = [y_1, \dots, y_t, \dots, y_q]$.

IV. SIMULATION AND RESULT ANALYSIS

To assess the performance of the proposed algorithm, we have applied it in three extensive simulations, namely: a numerical periodic function test; the Mackey-Glass time series with random noise; and the M3 time series prediction competition. The numerical periodic function test has been chosen as it was used in the original paper by Li [8] and it is useful to perform a comparison. The Mackey-Glass time series is a standard benchmark test for noisy data. The M3 time series prediction data have previously been utilised to assess algorithmic performance in time-series analysis and, as results are readily available for a range of methods, represent an opportunity to quantify the performance of the approach against other well-known algorithms.

A. Numerical Periodical Function Testing

We first assess the performance of the PFNN_R using the same non-linear model as in the PFNN paper by Li et al [8]. The nonlinear model is expressed as

$$y(t) = \frac{y(t-1)y(t-2)[y(t-1)+2.5]}{1+y(t-1)^2+y(t-2)^2} + u(t-1) \quad (21)$$

where $y(t)$ is the output of the nonlinear system and $u(t)$ is a sinewave signal with random noise ε . The random noise $\varepsilon(t)$ is described by a PDF $N(\mu_\varepsilon, \sigma_\varepsilon^2(t))$ with $\sigma_\varepsilon(t)$ assumed to be unknown and time-varying. The input of the PFNN_R is defined as $x(t) = [y(t-1), y(t-2), u(t)]^T$. Parameters are initially set as: $\mu_\varepsilon=0.1$ and $\sigma_\varepsilon^2(t) \in [0.01, 0.02]$ ($\sigma_\varepsilon(t) \in [0.1, 0.141]$), later increased to $\sigma_\varepsilon^2(t) \in [0.1, 0.11]$ ($\sigma_\varepsilon(t) \in [0.316, 0.332]$). 500 pairs of data were fed into the PFNN_R for training over 5000 iterations. The mutual parameters of PFNN and PFNN_R are identical, with variation of learning parameters (η_1, η_2, η_3) in the recurrent probability generator module for PFNN_R.

Figure 3 shows the RMSE between y and y^c , plotted for both the original PFNN and PFNN-R for 500 data pairs and 5000 training iterations. The upper group of curves in Figure 3 are the result for the larger $\sigma_\varepsilon(t)$ and the lower group for the smaller $\sigma_\varepsilon(t)$.

Figure 3: RMSE comparison between the PFNN [8] and the PFNN-R developed in this work for the numerical periodic function approximation used in [5] for 500 data pairs and 5000 training iterations. Solid line PFNN; dot-dash PFNN-R with lower learning rate; large-dashed line PFNN-R with higher learning rate. Upper group of curves higher noise level than that used for the lower group.

The solid line in Figure 3 shows the PFNN performance, the dot-dash the PFNN-R performance with lower learning rate and the large-dashed line PFNN-R with higher learning rate. The upper group of lines is for larger $\sigma_{\epsilon}(t)$. It may be observed that, irrespective of the learning rate, the PFNN-R achieves a lower RMSE with increasing number of iterations. Comparing the RMSE during training, Figure 3 shows that the PFNN_R network outperforms the PFNN and that with increasing randomness of the system, the gap between PFNN_R and PFNN grows bigger. Furthermore, appropriately tuning of η_2 and η_3 leads to better performance.

B. Mackey-Glass Time Series Prediction Integrated with Random Noise

The Mackey-Glass time series is a standard benchmark test. This time-series function is defined as

$$y(t+1) = (1-a)y(t) + \frac{by(t-\tau)}{1+y^{10}(t-\tau)} + \epsilon(t) \quad (22)$$

$$a=0.1, b=0.2, \tau=17, y(0)=1.2.$$

where the random noise $\epsilon(t)$ is as described in Section IV.A. Here, 500 input-target data between $t=126$ and 625 are selected as the training data, and the subsequent 50 data points between $t=626$ to 675 are used to test the performance of the PFNN_R. The input of the PFNN_R is defined as:

$$x(t) = [y(t-18), y(t-12), y(t-6)]^T \quad (23)$$

Parameters are initially set as follows: $\mu_{\epsilon}=0.1$ and $\sigma_{\epsilon}^2(t) \in [0.001, 0.011]$, ($\sigma_{\epsilon}(t) \in [0.0316, 0.1048]$). The RMSE between y and y^c for training the PFNN and PFNN_R over 5000 iterations are shown in Table 1. After training with the 500 values of input-target data, the PFNN_R is used to predict the next data

point (one step prediction) and the next 50 data points i.e. predicting the data points between $t=626$ to $t=675$. Table 1 also presents these results.

Table 1: Prediction of Mackey-Glass data points between $t=626$ and $t=675$ for both one step ahead and all 50 steps.

Range 626-675	PFNN RMSE	PFNN-R RMSE	Noise
One Step ahead prediction	0.1680	0.1565	$\sigma_{\epsilon}^2(t) \in [0.001, 0.011]$ ($\sigma_{\epsilon}(t) \in [0.0316, 0.1048]$)
50 steps ahead prediction	0.1872	0.1760	$\sigma_{\epsilon}^2(t) \in [0.001, 0.011]$ ($\sigma_{\epsilon}(t) \in [0.0316, 0.1048]$)

Finally, we compare the PFNN-R results with other methods in the literature as reported by [1] using the same method of noise insertion (described in [2]). The results are presented in Table 2 and it may be seen that for all three noise levels, the PFNN-R outperforms all other approaches considered.

Table 2: Comparison of PFNN-R with other methods for Mackey-Glass Time Series (PFNN-R results appended to those of reference [1]).

Type	Algorithm	RMSE (Trained in presence of noise of S.D.)		
		0.1	0.2	0.3
Type 1	SONFIN	0.139	0.178	0.236
	DENFIS	0.147	0.161	0.175
	EFuNN	0.143	0.195	0.257
	ANFIS	0.13	0.249	0.33
Type 2	SEIT2FNN	0.119	0.141	0.216
	IT2FNN-SVR	0.109	0.125	0.151
	eT2FIS	0.127	0.154	0.177
	McIT2FIS-UM	0.104	0.117	0.126
	McIT2FIS-US	0.104	0.11	0.128
	eRIT2IFNN	0.068	0.0818	0.0876
	eRIT2IFNN-A	0.098	0.1044	0.1148
	eRIT2FNN-B	0.0855	0.0958	0.1043
This Work	PFNN-R	0.0465	0.0485	0.0556

Both sets of results (outlined in Tables 1 and 2) show that the PFNN_R achieved better results when dealing with random noise integrated into the Mackey-Glass time series prediction as compared to the PFNN and other approaches in the literature.

C. M3 Competition Time Series Prediction

The M3-time series prediction competition is an internationally recognised competition to compare the performance of various time series analysis methods. In this work the monthly data series, which consists of 1428 different datasets, has been used [39][40]. The PFNN_R was evaluated using the data in twelve monthly subsets selected from the M3-Competition. Twelve datasets were randomly selected from each category, with a minimum of one from each category and a greater emphasis on the industry category. The selected sub-set labels were N2516, N2521, N1807, N1980, N2012, N2159, N2150, N1918, N2905, N2213, N2773 and N2596. Each dataset has 144 entries, except N1807 which has 126 entries, N2213 which has 134 entries and N2773 which has 138 entries. The last 18 items of each dataset are used for a prediction test while the remaining items are used for training as outlined in the M3 competition guidelines. Since the data is collected monthly, the input is defined as

$x(t) = [y(t-12), y(t-8), y(t-4)]^T$. Figure 4 shows two examples of the datasets where the modelling challenge is visually apparent.

Figure 4: Two examples of the M3 monthly dataset

To ensure clarity on the performance assessment, we use both the **symmetric mean absolute error (SMAE)** and **RMSE** metrics, which are calculated as follows:

$$SMAE = \sum \frac{|y - y^c|}{0.5(y + y^c)} \quad (24)$$

$$RMSE = \sqrt{\frac{1}{n} \sum (y - y^c)^2} \quad (25)$$

As this is a public competition with common datasets and challenge, we can compare the PFNN_R results with a range of methods. The results are shown in Table 3 in terms of SMAE and Table 4 for RMSE. From Tables 3 and 4, it may be seen that the PFNN-R has both the lowest average SMAE and RMSE for the 12 datasets under consideration, and the lowest actual SMAE and RMSE in four of the 12 datasets. No other

technique achieves a top ranking in more than three of the twelve datasets. The best results are in N2516, N2521 and N2596 according to both metrics. It is interesting to analyse potential reasons for the various performances relative to a particular dataset. From Figure 5, which plots the Fast Fourier Transform (FFT) for two datasets, it may be seen that for N1980, where the prediction of PFNN_R is not as good as the other methods, the data is more periodic and with less noise. Conversely, N2516 has significantly less periodicity and it is for this dataset that the PFNN_R performs best among the other techniques. The variation in performance may be related to the specific characteristics of the dataset but a detailed understanding of this remains to be determined with further research. However, it is evident that, in general, the PFNN_R improves the capability to address significant uncertainty.

Figure 5: Fast Fourier Transform of the datasets N2516 and N1980

Table 3: SMAE for each method and average value for 12 data sets

Methods\Data set No.	N2516	N2521	N1807	N1980	N2012	N2159	N2150	N1918	N2095	N2213	N2773	N2596	Average SMAE
PFNN_R	0.679	4.824	0.294011	2.887	0.957	2.353	0.572	0.489	0.922	0.406	0.045	3.487	1.4929
aam2	2.143	5.454	0.342408	3.179	0.614	2.056	0.606	0.569	1.4	0.275	0.071	3.748	1.7048
theta	2.5	5.458	0.30992	3.177	1.016	1.703	0.987	0.432	0.88	1.112	0.034	3.828	1.7864
SINGLE	2.52	5.455	0.383981	3.138	1.341	1.637	1.13	0.428	0.75	1.265	0.123	4.739	1.9092
DAMPEN	2.541	5.409	0.38599	3.196	1.367	1.582	1.266	0.433	0.976	1.035	0.064	4.737	1.9160
ararma	1.944	5.481	0.257081	2.9	0.773	2.159	1.505	0.507	0.909	1.374	0.085	6.175	2.0058
HOLT	3.734	5.384	0.359566	3.191	0.775	2.721	0.839	0.456	1.182	0.935	0.07	5.606	2.1044
autobox1	2.21	5.053	0.196122	3.095	0.688	1.79	1.291	0.638	1.514	1.002	0.104	8.047	2.1357
NAIVE2	2.52	8.823	0.298137	3.142	1.356	1.564	1.13	0.428	0.731	1.302	0.123	4.943	2.1967
WINTER	3.734	5.384	0.359566	4.027	0.945	2.76	1.18	0.478	0.807	1.172	0.07	5.606	2.2102
rbf	1.333	9.082	0.259433	3.134	0.68	4.259	1.161	0.655	1.299	0.954	0.049	4.619	2.2904
auto-ann	2.734	5.383	0.310727	2.582	0.918	6.862	6.884	0.497	1.52	0.458	0.038	3.983	2.6808
robust-trend	2.385	14.291	0.333161	3.534	0.806	1.688	0.815	0.467	2.058	0.883	0.061	6.759	2.8400

Table 4: RMSE for each method and average value for 12 data sets

Methods\Data set No.	N2516	N2521	N1807	N1980	N2012	N2159	N2150	N1918	N2095	N2213	N2773	N2596	Average RMSE
PFNN_R	252.84	1832.69	201.79	755.7	318.4	690.65	93.01	166.8	397.62	211.64	18.96	2028.38	580.71
theta	882.9	2028.49	250.55	808.82	297.57	478.53	171.7	143.28	360.04	417.33	16.21	2133.75	665.76
aam2	760.42	2038.54	387.2	738.63	201.74	753.03	82.45	164.58	485.01	186.51	29.76	2296.68	677.05
SINGLE	912.64	2026.72	452.88	728.18	376.74	452.64	168.2	137.79	302.49	453.84	53.77	2475.54	711.79
DAMPEN	919.59	2011.66	456.16	815.61	382.95	437.78	183.47	137.59	392.37	406.22	27.28	2475.03	720.48
ararma	693.75	2054.82	404.6	740.02	240.81	609.03	207.03	164.32	335.21	509.8	35.86	3482.71	789.83
NAIVE2	912.64	3088.26	318.04	693.27	380.29	431.32	168.2	137.79	291.24	464.11	53.57	2569.01	792.31
HOLT	1408	2006.53	325.49	811.84	241.02	726.31	128.03	151.81	439.27	390.81	29.66	3008.24	805.58
rbf	444.29	3146.3	233.4	707.76	238.16	1052.58	160.02	209.42	463.87	392.89	21.58	2634.34	808.72
WINTER	1408	2006.53	325.49	1111.12	280.57	904.71	174.75	158.32	331.36	429.32	29.66	3008.24	847.34
auto-ann	980.65	2021.41	257.34	637.65	274.18	2749.77	1129.83	155.17	533.49	187.82	16.06	2158.74	925.18
autobox1	793.13	2003.2	504.47	668.29	222.87	546.49	192.21	180.18	519.85	398.33	43.91	5161.31	936.19
robust-trend	860.16	4094.89	287.46	867.96	248.66	476.69	123.14	137.57	748.08	334.44	25.79	3898.88	1008.64

D. Computational Complexity

To assess the computational complexity of the PFNN-R compared with that of the PFNN algorithm [8], the Mackey-Glass problem is revisited. Figure 6 shows the RMSE for training over 5000 iterations for the PFNN-R and for 10,000 iterations for the PFNN.

Figure 6: RMSE during 5000 training iterations (PFNN-R) and 10000 iterations (PFNN).

The dashed line is the result of PFNN whilst the solid line is PFNN_R with $\eta_1 = 0.2$ $\eta_2 = 1$ $\eta_3 = 0.5$. The PFNN_R contains the recurrent module for probability generation and so the computational cost is higher than that of PFNN. However, considering Figure 6, it may be seen that the efficiency of the PFNN-R is higher than that of PFNN, in that it reaches a lower RMSE after only 5000 iterations, whereas the RMSE of the PFNN is still higher than this figure after 10,000 iterations. Obviously, the run time of PFNN_R for 5000 iterations (12698 seconds) is significantly shorter than that for PFNN with 10,000 iterations (19038 seconds).

E. Effect of B_{delta}

The parameter B_{delta} , controls the degree of recurrency look-back and its impact was also investigated. Two example datasets are shown in Figure 7, in both cases for two different values of B_{delta} (10 and 20). The two datasets chosen represent examples of those datasets in which the PFNN-R algorithm performs very well (N2596) and where it performs quite badly (N2159) in terms of SMAE and RMSE (Tables 3 and 4) relative to the existing methods. It is noteworthy that for a dataset in which PFNN-R performs well (for example N2596), there is only a small improvement in RMSE achieved by increasing B_{delta} from 10 to 20. Conversely, for the N2159 dataset, there is a consistent dis-improvement in performance as B_{delta} increases

from 10 to 20. This might suggest that the PFNN_R is most useful when applied to modelling time-series data with greater uncertainty where the recurrent loop-back is particularly relevant.

Figure 7: Effect of varying B_{delta} during 1000 training iterations for two example datasets N2159 and N2596.

V. CONCLUSION

A probabilistic fuzzy neural network with an integrated recurrent probabilistic generator module has been proposed, for modelling time series data in the presence of noise. The results, taken together, provide important insights into the fact that the PFNN_R network has enhanced performance across a range of challenges, including a numeric periodic function approximation, Mackey-Glass chaotic time-series, and the M3 monthly prediction challenge. We conclude that PFNN_R has a greater capability for modelling time-series data with greater uncertainty than other methods. Furthermore, an appropriate PDF can be generated based on the recurrent probabilistic generation module and its parameter neural network learning. Appropriate parameter selection is an ongoing limitation in the PFNN-R and equivalent approaches. Future work will be targeted at the development of a self-organising PFNN-R algorithm, potentially extending our previous work on the SOFNN [41], as there needs to be a self-organization mechanism for the number of fuzzy rules and M_j (dimensions of the probabilistic set for the j th fuzzy rule). In addition, there are plans to apply the algorithm to biomedical datasets, particularly retinal responses and other interesting applications in engineering, computational neuroscience, data analytics and robotics, which are known to involve noisy time-series data.

REFERENCES

- [1] C. Luo, C. Tan, X. Wang, and Y. Zheng, "An evolving recurrent interval type-2 intuitionistic fuzzy neural network for online learning and time series prediction," *Appl. Soft Comput. J.*, vol. 78, pp. 150–163, May 2019.
- [2] C. F. Juang, R. B. Huang, and W. Y. Cheng, "An interval type-2 Fuzzy-neural network with support-vector regression for noisy regression problems," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 4, pp. 686–699, Aug. 2010.
- [3] J. Soto, P. Melin, and O. Castillo, "Time series prediction using ensembles of ANFIS models with genetic optimization of interval type-2 and type-1 fuzzy integrators," *Int. J. Hybrid Intell. Syst.*, vol. 11, no. 3, pp. 211–226, Apr. 2016.
- [4] P. Melin, J. Soto, O. Castillo, and J. Soria, "A new approach for time series prediction using ensembles of ANFIS models," *Expert Syst. Appl.*, vol. 39, no. 3, pp. 3494–3506, Feb. 2012.
- [5] C. L. P. Chen, Y. J. Liu, and G. X. Wen, "Fuzzy neural network-based adaptive control for a class of uncertain nonlinear stochastic systems," *IEEE Trans. Cybern.*, vol. 44, no. 5, pp. 583–593, 2014.
- [6] Y. Li and Q. Zhu, "Stability Analysis for Discrete-Time Stochastic Fuzzy Neural Networks with Mixed Delays," *Math. Probl. Eng.*, vol. 2019, p. Article ID 8529053.
- [7] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, 1997.
- [8] H. Li, S. Member, and Z. Liu, "A Probabilistic Neural-Fuzzy Learning System for Stochastic Modeling," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 4, pp. 898–908, 2008.
- [9] H. X. Li, Y. Wang, and G. Zhang, "Probabilistic Fuzzy Classification for Stochastic Data," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 6, pp. 1391–1402, 2017.
- [10] T. M3-Competition, "https://forecasters.org/resources/time-series-data/m3-competition/", last accessed 5-11-20."
- [11] S. Horikawa, "A fuzzy controller using a neural network and its capability to learn expert's control rules," in *Proceedings of international conference on Fuzzy logic and neural networks*, 1990, pp. 103–106.
- [12] S. Horikawa, T. Furuhashi, S. Okuma, and Y. Uchikawa, "Composition methods of fuzzy neural networks," in *IECON '90: 16th Annual Conference of IEEE Industrial Electronics Society*, 1990, pp. 1253–1258.
- [13] S. Horikawa, T. Furuhashi, and Y. Uchikawa, "Composition methods of fuzzy neural networks (III)," in *7th Fuzzy System Symp.*, 1991, pp. 493–496.
- [14] S. Horikawa, T. Furuhashi, and U. Yoshiki, "On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm," *IEEE Trans. Neural Networks*, vol. 3, no. 5, pp. 801–806, 1992.
- [15] Shiqian Wu, Meng Joo Er, and Yang Gao, "A fast approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks," *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 4, pp. 578–594, 2001.
- [16] W. Yu and X. Li, "Fuzzy Identification Using Fuzzy Neural Networks With Stable Learning Algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 3, pp. 411–420, Jun. 2004.
- [17] G. Leng, T. M. McGinnity, and G. Prasad, "An approach for on-line extraction of fuzzy rules using a self-organising fuzzy neural network," *Fuzzy Sets Syst.*, vol. 150, no. 2, pp. 211–243, Mar. 2005.
- [18] Shiqian Wu and Meng Joo Er, "Dynamic fuzzy neural networks-a novel approach to function approximation," *IEEE Trans. Syst. Man Cybern. Part B*, vol. 30, no. 2, pp. 358–364, Apr. 2000.
- [19] W. He and Y. Dong, "Adaptive Fuzzy Neural Network Control for a Constrained Robot Using Impedance Learning," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 29, no. 4, pp. 1174–1186, Apr. 2018.
- [20] A. Sharifian, M. J. Ghadi, S. Ghavidel, L. Li, and J. Zhang, "A new method based on Type-2 fuzzy neural network for accurate wind power forecasting under uncertain data," *Renew. Energy*, vol. 120, pp. 220–230, May 2018.
- [21] D. Sun, Q. Liao, T. Stoyanov, A. Kiselev, and A. Loutfi, "Bilateral telerobotic system using Type-2 fuzzy neural network based moving horizon estimation force observer for enhancement of environmental force compliance and human perception," *Automatica*, vol. 106, pp. 358–373, Aug. 2019.
- [22] D. F. Specht, "Probabilistic neural networks," *Neural Networks*, vol. 3, no. 1, pp. 109–118, Jan. 1990.
- [23] H. Adeli and A. Panakktat, "A probabilistic neural network for earthquake magnitude prediction," *Neural Networks*, vol. 22, no. 7, pp. 1018–1024, Sep. 2009.
- [24] H.-T. Lin, T.-J. Liang, and S.-M. Chen, "Estimation of Battery State of Health Using Probabilistic Neural Network," *IEEE Trans. Ind. Informatics*, vol. 9, no. 2, pp. 679–685, May 2013.
- [25] M. F. Othman and M. A. M. Basri, "Probabilistic Neural Network for Brain Tumor Classification," in *2011 Second International Conference on Intelligent Systems, Modelling and Simulation*, 2011, pp. 136–138.
- [26] N. Varuna Shree and T. N. R. Kumar, "Identification and classification of brain tumor MRI images with feature extraction using DWT and probabilistic neural network," *Brain Informatics*, vol. 5, no. 1, pp. 23–30, Mar. 2018.
- [27] G. Pan, G. Yan, X. Qiu, and J. Cui, "Bleeding Detection in Wireless Capsule Endoscopy Based on Probabilistic Neural Network," *J. Med. Syst.*, vol. 35, no. 6, pp. 1477–1484, Dec. 2011.
- [28] Y. Sun, J. Chen, C. Yuen, and S. Rahardja, "Indoor Sound Source Localization With Probabilistic Neural Network," *IEEE Trans. Ind. Electron.*, vol. 65, no. 8, pp. 6403–6413, Aug. 2018.
- [29] L. A. Zadeh, "Probability of fuzzy events," *J. Math. Anal. Appl.*, vol. 23, no. 2, pp. 421–427, 1968.
- [30] J. van den Berg, U. Kaymak, and W.-M. van den Bergh, "Financial markets analysis by using a probabilistic fuzzy modelling approach," *Int. J. Approx. Reason.*, vol. 35, no. 3, pp. 291–305, Mar. 2004.
- [31] Z. Liu and H.-X. Li, "A Probabilistic Fuzzy Logic System for Modeling and Control," *IEEE Trans. FUZZY Syst.*, vol. 13, no. 6, pp. 848–859, 2005.
- [32] T. D. Chaudhuri, T. D. Chaudhuri, and I. Ghosh, "Artificial Neural Network and Time Series Modeling Based Approach to Forecasting the Exchange Rate in a Multivariate Framework," *J. Insur. Financ. Manag.*, vol. 1, no. 5, pp. 92–123, Jul. 2016.
- [33] J. L. Elman, "Finding Structure in Time," *Cogn. Sci.*, vol. 14, no. 2, pp. 179–211, Mar. 1990.
- [34] H. Jaeger, "The 'Echo State' Approach to Analysing and Training Recurrent Neural Networks," *GMD-Report 148, Ger. Natl. Res. Inst. Comput. Sci.*, Jan. 2001.
- [35] C. Tallec and Y. Ollivier, "Can recurrent neural networks warp time?," *arXiv: 1804.11188*, 2018.
- [36] C. Ulbricht, "Multi-Recurrent Networks for Traffic Forecasting," in *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 2)*, 1994, pp. 883–888.
- [37] J. A. Tepper, M. S. Shertil, and H. M. Powell, "On the importance of sluggish state memory for learning long term dependency," *Knowledge-Based Syst.*, vol. 96, pp. 104–114, Mar. 2016.
- [38] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A Critical Review of Recurrent Neural Networks for Sequence Learning," *arXiv 1506.00019*, no. June, 2015.
- [39] S. Makridakis and M. Hibon, "The M3-competition: Results, conclusions and implications," *Int. J. Forecast.*, vol. 16, no. 4, pp. 451–476, 2000.
- [40] W. L. Gorr and M. J. Schneider, "Large-change forecast accuracy: Reanalysis of M3-Competition data using receiver operating characteristic analysis," *Int. J. Forecast.*, vol. 29, no. 2, pp. 274–281, Apr. 2013.
- [41] G. Leng, G. Prasad, and T. M. McGinnity, "An on-line algorithm for creating self-organizing fuzzy neural networks," *Neural Networks*, vol. 17, no. 10, pp. 1477–1493, Dec. 2004.

Yong Li received the BS degree in Automation and the MS and Ph.D. degree in Control Theory and Control Engineering from Northeastern University, Shenyang, China, in 2003, 2006, and 2010, respectively.

After graduation, he was involved in teaching and researching at the Shenyang University of Technology, Shenyang, China, where he is currently an Associate Professor. He is now a research assistant in Intelligent Systems Research Centre, Ulster University, Magee Campus.

His research interests mainly include data analysis, neural network and robotic system modelling and multi-objective optimization.

T. Martin McGinnity (SMIEEE, FIET) received a First Class (Hons.) degree in Physics in 1975, and a Ph.D degree in 1979.

He currently holds a part-time Professorship in both Nottingham Trent University (NTU), UK and Ulster University. He is the author or co-author of 350+ research papers and leads the Computational Neuroscience and Cognitive Robotics research group at NTU.

His current research is focused on the development of biologically-compatible computational models of human sensory systems, including auditory signal processing; human tactile emulation; human visual processing; sensory processing modalities in cognitive robotics; and implementation of neuromorphic systems on electronic hardware.

Richard Gault (MIEEE, FHEA) received a First Class (Hons.) degree in Mathematics and Computer Science in 2013 from Queen's University, Belfast, and a PhD degree in Computer Science from Ulster University in 2017.

After graduating, he became a Research Fellow at Nottingham Trent University, NTU, and in 2018 became a Lecturer (Education) at Queen's University, Belfast. He is currently involved in research and teaching at Queen's University, Belfast, as a Doctoral Fellow and is Vice-Chair of IEEE UK & Ireland Chapter of the Engineering in Medicine & Biology Society.

His current research is focused on the development of machine learning approaches to address challenges in medical applications.