# Time Series Forecasting using Neural Networks: Are Recurrent Connections Necessary?

**Salihu A Abdulkarim · Andries P Engelbrecht**

**Abstract** Artificial neural networks (NNs) are widely used in modeling and forecasting time series. Since most practical time series are non-stationary, NN forecasters are often implemented using recurrent/delayed connections to handle the temporal component of the time varying sequence. These recurrent/delayed connections increase the number of weights required to be optimized during training of the NN. Particle swarm optimization (PSO) is now an established method for training NNs, and was shown in several studies to outperform the classical backpropagation training algorithm. The original PSO was, however, designed for static environments. In dealing with non-stationary data, modified versions of PSOs for optimization in dynamic environments are used. These dynamic PSOs have been successfully used to train NNs on classification problems under non-stationary environments. This paper formulates training of a NN forecaster as dynamic optimization problem to investigate if recurrent/delayed connections are necessary in a NN time series forecaster when a dynamic PSO is used as the training algorithm. Experiments were carried out on eight forecasting problems. For each problem, a feedforward NN (FNN) is trained with a dynamic PSO algorithm and the performance is compared to that obtained from four different types of recurrent NNs (RNN) each trained using gradient descent, a standard PSO for static environments and the dynamic PSO algorithm. The RNNs employed were an Elman NN, a Jordan NN, a multirecurrent NN (MRNN) and a time delay NN (TDNN). The

Salihu A. Abdulkarim
Computer Science Department,
University of Pretoria,
Pretoria 002, South Africa
E-mail: u12211746@tuks.co.za

Andries P. Engelbrecht
Department of Industrial Engineering,
Department of Computer Science,
Stellenbosch University,
Stellenbosch, 7600, South Africa

performance of these forecasting models were evaluated under three different dynamic environmental scenarios. The results show that the FNNs trained with the dynamic PSO significantly outperformed all the RNNs trained using any of the other algorithms considered. These findings highlight that recurrent/delayed connections are not necessary in NNs used for time series forecasting (for the time series considered in this study) as long as a dynamic PSO algorithm is used as the training method.

**Keywords** Time series forecasting · Neural networks · Recurrent neural networks · Resilient propagation · Particle swarm optimization · Cooperative quantum particle swarm optimization

## 1 Introduction

Time series forecasting is a very important research area because of its practical application in many fields. Interest in using neural networks (NNs) to model and forecast time series has been growing. While feedforward NNs (FNNs) are the type dominantly used, recent trends indicate that recurrent NNs (RNNs) are now the widely used architecture because of their ability to model dynamic systems implicitly. The recurent/delayed connections in RNNs, however, increase the number of weights that are required to be optimized during training of the NN.

To build a NN forecaster using RNNs, one has to choose among the different special cases of RNNs (such as Elman NN, Jordan NN, Multirecurrent NN or Time delay NN). It is, however, very difficult to single out a particular special case of the RNNs that is better in modeling different types of time series [1]. Thus, selecting which RNN to use becomes a problem. Another issue that needs to be resolved is how to decide on the number of context layers if an Elman NN is used, or the number of state layers if a Jordan NN is used, or how to decide on the number of previous values to remember if a time delay NN is used.

Particle swarm optimization (PSO), which is a population based search algorithm, has been used with success to training NNs for a number of problems [9], [19], [21]. PSO has also been applied in training NNs for time series prediction [30], [33], [34]. Even though several studies have applied PSO to the task of training NN forecasters, producing favorable results, these studies have assumed static environments, making them unsuitable for many real-world time series which are generated by varying processes. For example, the performance of PSO as a training algorithm for FNNs in forecasting non-stationary time series was evaluated in [3], where the authors considered the problem as a static optimization problem. The results showed that PSO performed better than the backpropagation algorithm (BP), but concluded that the result of PSO could significantly be improved by using a suitable NN structure (that can handle dynamic data). Jar *et al* [23] compared the performance of PSO and BP in training FNNs for time series forecasting. The results indicate su-

periority of the PSO over BP, but Jar *et al* concluded that the PSO trained FNNs were unable to track non-stationary data.

A number of PSO variants have been developed to solve optimization problems in dynamic environments [13]. Some of these algorithms have been applied to NN training on classification problems, in order to cope with concept drift [36], [37]. The studies showed that dynamic PSO is indeed applicable to NN training in dynamic environments, and has produced significantly better or similar performance compared to the classic NN training algorithm.

This study tests a hypothesis that recurrent/delayed connections are not necessary in a NN trained for non-stationary time series forecasting if a dynamic PSO is used as the training algorithm. If this hypothesis holds true, then the problems of using a RNN to model non-stationary time series are resolved. To test the hypothesis, a set of experiments using eight forecasting problems were carried out. For each problem, a pairwise performance comparison between a FNN trained with a dynamic PSO algorithm is carried out with four different RNNs each trained differently with a classical gradient descent algorithm, a standard PSO algorithm and a dynamic PSO algorithm. The RNNs employed were Elman NNs, Jordan NNs, MRNNs, and TDNNs. All performances were evaluated under three different dynamic environmental scenarios.

In the reminder of this paper, section 2 presents background on the algorithms used in this study. Section 3 describes the experimental procedure employed. Results are presented and discussed in section 4. Section 5 concludes the paper.

## 2 Background

This section provides the background information necessary for this study. Section 2.1 presents a short overview on NNs, and Section 2.2 discusses NN training algorithms.

### 2.1 Neural Networks

A brief overview on the five different NN structures used in this study is provided in this section.

#### *2.1.1 Feedforward Neural Network*

A FNN has a network of artificial neurons usually organized in three layers (input, hidden, and output) as shown in Figure 1a. In FNNs, information flows in only one direction, from the input layer, through the hidden layer to the output layer.

For any given input pattern $\mathbf{x}$, the output from a FNN is calculated as,

$$o_k = f_{o_k}\left(\sum_{j=1}^{J+1} w_{kj} f_{y_j}\left(\sum_{i=1}^{I+1} v_{ji} x_i\right)\right) \tag{1}$$

where $x_i, i = 1 \ldots I$, and $y_j, j = 1, \ldots J$ are the nodes in the input and hidden layers respectively; the $(I+1)^{th}$ input and $(J+1)^{th}$ hidden units are bias units for nodes in the next level; $w_{kj}$ is the weight between output unit $o_k$ and hidden unit $y_j$; $v_{j,i}$ is the weight between hidden unit $y_j$ and input unit $x_i$; $f_{o_k}$ and $f_{y_i}$ are the activation functions of output unit $o_k$ and hidden unit $y_j$ respectively.

### *2.1.2 Elman Neural Network*

The Elman recurrent NN [16] structure is similar to a standard three layer FNN structure with feedback connections from the hidden layer to a context layer, as shown in Figure 1b. The context layer serves as an extension to the input layer. Thus, the input vector becomes

$$\mathbf{x} = \underbrace{x_1, \ldots, x_{I+1}}_{\text{input}}, \underbrace{x_{I+2}, \ldots, x_{I+1+J}}_{\text{context}} \tag{2}$$

and the output of each node in the output layer is calculated as

$$o_k = f_{o_k}\left(\sum_{j=1}^{J+1} w_{kj} f_{y_j}\left(\sum_{i=1}^{I+1+J} v_{ji} x_i\right)\right) \tag{3}$$

where $(x_{1+2}, \ldots, x_{I+1+J}) = (y_1(t-1), \ldots, y_J(t-1))$.

Since a single context layer is limited in representing past information to only one step (i.e. one window), more context layers are required to deal with a long history in the data. This introduced the issue of how to decide the number of context layers to used.

### *2.1.3 Jordan Neural Network*

The Jordan recurrent NN [24] structure is very similar to the Elman recurrent NN structure, except that the state layer stores a copy of the output layer instead of the hidden layer, as shown in Figure 1c. The state layer of a Jordan recurrent NN extends the input layer to

$$\mathbf{x} = \underbrace{x_1, \ldots, x_{I+1}}_{\text{input}}, \underbrace{x_{I+2}, \ldots, x_{I+1+K}}_{\text{state}} \tag{4}$$

and the output of each output node is calculated as

$$o_k = f_{o_k}\left(\sum_{j=1}^{J+1} w_{kj} f_{y_j}\left(\sum_{i=1}^{I+1+K} v_{ji} x_i\right)\right) \tag{5}$$

where $(x_{1+2}, \ldots, x_{I+1+J}) = (o_1(t-1), \ldots, o_K(t-1))$.

To capture long term dependency, as for the Elman recurrent NN, one has to deal with the problem of deciding on the number of state layers to use.

### *2.1.4 Multirecurrent Neural Network*

A Multirecurrent NN (MRNN) [12] is obtained by combining an Elman NN and a Jordan NN. It has feedback connections from both the hidden and the output layers connecting to the context layer and the state layer as shown in Figure 1d. The output layer is extended to

$$\mathbf{x} = \underbrace{x_1, \ldots, x_{I+1}}_{\text{input}}, \underbrace{x_{I+2}, \ldots, x_{I+1+J}}_{\text{context}}, \underbrace{x_{I+2}, \ldots, x_{I+1+K}}_{\text{state}} \tag{6}$$

and the output from the network is calculated as

$$o_k = f_{o_k}\left(\sum_{j=1}^{J+1} w_{kj} f_{y_j}\left(\sum_{i=1}^{I+1+J+K} v_{ji} x_i\right)\right) \tag{7}$$

where $(x_1, \ldots, x_{I+1}, x_{I+2}, \ldots, x_{I+1+J}, x_{I+2}, \ldots, x_{I+1+K}) = (y_1(t-1), \ldots, y_j(t-1), o_1(t-1), \ldots, o_J(t-1))$.

An MRNN has a larger number of weights compared to the Elman and Jordan RNNs. Also, the context/state layer is only one time step (i.e. a time window of 1). To remember more time steps, more context layers are needed.

### *2.1.5 Time Delay Neural Networks*

TDNNs [46] are a special case of FNNs designed with extra time delay connections (which is a form of memory mechanism) for effective handling of temporal data. Input patterns to the network are delayed to arrive at hidden units at different points in time. A special type of neurons with $n_t$ delayed patterns, illustrated in Figure 1e are used in the input layer of a TDNN.

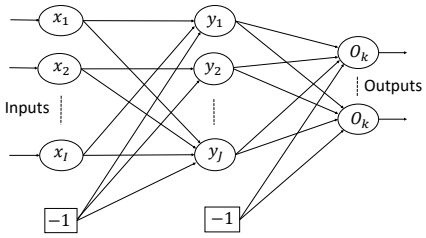
The output of a TDNN is calculated as

$$o_k = f_{o_k}\left(\sum_{j=1}^{J+1} w_{k,j} f_{y_j}\left(\sum_{i=1}^{I}\sum_{t=0}^{n_t} v_{j,i} x_i(t) + x_{I+1} v_{I+1}\right)\right) \tag{8}$$

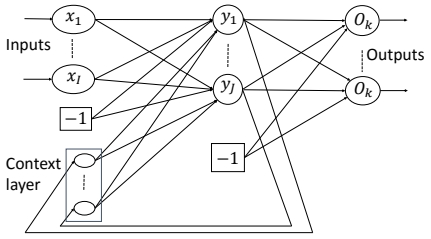
The TDNN is identical to a time window and can also be viewed as an autoregressive model. The major disadvantage of the TDNN is that the time period processed is strictly limited by the number and arrangement of the time delays.

## 2.2 Training Algorithms

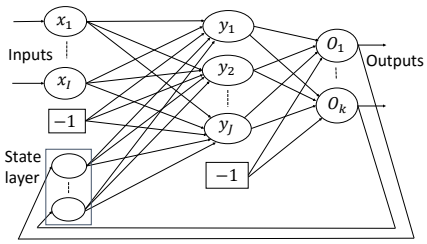
This section briefly discusses the resilient propagation algorithm (Rprop), the original PSO and the cooperative quantum PSO (CQSO) used.



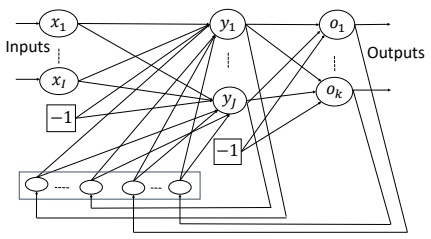

(a) Feedforward Neural Network



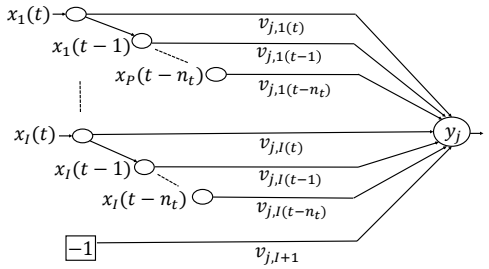

(b) Elman Neural Network




(c) Jordan Neural Network




(d) Multi-Recurrent Neural Network




(e) Time Delay Neural Network illustrated for one hidden neuron

Fig. 1: Neural Network Structures

### *2.2.1 Resilient propagation*

Rprop is an efficient supervised batch learning scheme that performs a direct adaption of weights based on local gradient information. During each iteration, if the derivative $^{\partial E}/_{\partial w_{ij}}$ changes sign, weight update value, $\delta_{ij}(t)$, is decreased by the factor, $\eta^-$ (to reduce the effect of previous large update) while on the contrary, if the derivative retains its sign, the update value is increased by $\eta^+$ (to promote convergence) before weight adjustment. Riedmiller [40] suggested that $\eta^+ = 1.2$ and $\eta^- = 0.5$. For most problems, Rprop does not require optimizing parameters to obtain optimal or at least near optimal convergence times, which is one of its key strengths [41].

### *2.2.2 Particle Swarm Optimizers*

PSO is a meta-heuristic search algorithm inspired by the social behaviour of birds in a flock [14]. A PSO algorithm manipulates a swarm of particles, where each particle is a potential solution to an optimization problem. The swarm traverses a search space searching for an optimum solution. As the particles move around, each particle is attracted to both the best position it has found so far, as well as the overall best position found within its neighborhood. Due to these dynamics, a swarm that was uniformly distributed in the search space converges on a small area around the optimum position [45], [10]. Refer to [14], [17] for a detailed description of PSO.

To train a NN using PSO, each particle in the swarm represents a weight vector of the NN. The dimension of each weight vector is set to be equal to the total number of weights (and biases) of the NN [31]. The quality of each particle is computed using the sum square error (SSE) over the training set. The objective of the PSO is to minimize the SSE.

### *2.2.3 Particle Swarm Optimizers for Dynamic Environments*

PSO was designed for solving optimization problems in a static environment. When PSO is applied in a dynamic environment, the algorithm faces two main challenges:

- Outdated memory: Once there is a change in the environment, the personal and neighborhood best positions of the PSO are no longer representative of the new environment [17]. Thus, the personal and neighborhood best positions should not continue to be used as attractors.
- Loss of swarm diversity: As the swarm starts to converge, all particles are concentrated around one point in the search space. Any environmental change that causes the optimum to move outside of this region can not be tracked due to lack in swarm diversity.

Several modifications of PSO have been developed to track optima in dynamic environments. These dynamic PSO algorithms include charged PSO [6], quantum PSO (QSO) [7], cooperative charged PSO (CCPSO) [35], and cooperative quantum PSO (CQSO) [44]. Refer to [13], [39] for an in-depth review of dynamic PSO algorithms. CQSO is used in this study because it was shown to have superiority over a number of other dynamic PSO algorithms over a wide range of different dynamic environment types [44].

### *2.2.4 Cooperative Quantum Particle Swarm Optimizers*

The CQSO partitions the search space dimension-wise into $k$ disjoint groups where each group is assigned to an individual subswarm to optimize. The algorithm maintains a complete solution (or context) vector that individual subswarms uses to evaluate the quality of its particles. Quality is evaluated for each particle in a subswarm by substituting values of the dimensions the

subswarm is accountable for into the context vector, while keeping constant the remaining values in the vector (which are the best values from other subswarms). The particle that, when substituted into the context vector, had the best quality is returned as the best solution in the subswarm. When all subswarms evaluate their particles' quality, the context vector contains the optimal solution discovered so far.

The subswarms in CQSO uses the QSO algorithm. QSO is inspired by the quantum model of an atom. In QSO, a portion of the particles implement the standard PSO velocity and position updates. The rest of the particles, referred to as quantum particles, have their positions sampled from a probability distribution centered on the global best position. Positions of the quantum particles are calculated as

$$\mathbf{x}_i \sim d(\hat{\mathbf{y}}, r_{cloud}) \tag{9}$$

where $d$ is the probability distribution and $r_{cloud}$ is the quantum radius. The control parameter, $r_{cloud}$ is problem and environment dependent [13], [8].

Recently, Harrison *et al* [20] proposed using a parent centric crossover (PCX) operator [11] to generate the positions of quantum particles instead of using the radius and probability distribution parameters used in the standard QSO. In an empirical study using single-peak dynamic environment types, the new variant (i.e. QSO-PCX) was shown to be superior to QSO on progressive and chaotic environments, while no superior approach on quasi-static and abrupt environments was found [20]. The performance of the QSO-PCX algorithm was, however, not evaluated on real-world and multi-peak environments.

## 3 Experimental Procedure

This study investigates the necessity of recurrent/delayed connections in NN forecasters when a dynamic PSO algorithm is used as the training method. For this purpose, a set of experiments under three different dynamic environmental scenarios, given in Table 4, were carried out on the eight forecasting problems described in Section 3.1. Each experiment involves training a FNN using a dynamic PSO algorithm and the results are compared to the results obtained from four different types of RNNs (i.e. Elman NN, Jordan NN, Multi-Recurrent NN and Time Delay NN) trained separately using Rprop, standard PSO and a dynamic PSO algorithm. For sound performance evaluation, 30 independent runs for each experiment was carried out and the average over these 30 runs was computed. All algorithms used were implemented in the Computational Intelligence library (CIlib) version 0.9. CIlib is available at https://github.com/cirg-up/cilib.

The remainder of this section discusses the control parameters of all the algorithms, the problems used in the experiments carried out, and also describes the performance metrics used in evaluating the models investigated.

### 3.1 Datasets

Eight time series with different complexities were used in the experiments. Six are real world time series obtained on-line from the Time Series Data Library [22]. The other two, Mackey glass and the Logistic map were artificially generated and can be obtained at https://tinyurl.com/yybxk3rj. Table 1 summarizes the statistics of the datasets used.

All datasets were scaled to the range [-1, 1] and standardized such that the mean of each input variable over the training set is close to zero as suggested in [27].

Each dataset was serially divided into two independent subsets. The first 80% of the dataset was used for training and the remaining 20% for testing. For control parameter optimization purposes only, the training subset was split further in 70:30 ratio for training and validation respectively.

### 3.2 Neural Networks Setup

For each problem, the architectures for all the NN models were selected as follows: 10 and 12 input nodes were chosen for problems with annually collected data (each representing a year in the decade) and monthly collected data (each representing a month of the year) respectively. For problems with data collected quarterly, four input nodes were used (each representing a quarter of the year), while 30 and 24 input nodes were used for problems with data collected daily (each representing a day of the month) and hourly (each representing an hour in the day) respectively. This intuitive method was used by a number of researchers such as [26], [18], [43] and has been effective in constructing optimal NN architectures. For the Mackey glass and Logistic map (which are synthetic problems), the number of input nodes were as used in [2].

A single hidden layer was used for all the NN models, and the optimal number of hidden nodes were selected iteratively for each training set from the discrete range [2, 50]. For every value within the range, 30 independent runs were conducted and the number of hidden units that yielded the minimum average training and validation errors was chosen as optimal. The number of time steps (or delayed patterns) in TDNNs were optimized in a similar way.

Table 1: Statistic of the Time Series Data under Study

| Dataset | Observations | Minimum | Mean | Maximum | St. Deviation | Variance | Skew |
|---|---|---|---|---|---|---|---|
| International Airline Passengers (AIP) | 144 | 104.00 | 280.30 | 622.00 | 119.97 | 14391.92 | 0.58 |
| Australian Wine Sales (AWS) | 187 | 1954.00 | 3262.61 | 5725.00 | 728.36 | 530504.45 | 0.73 |
| US Accidental Death (USD) | 72 | 6892.00 | 8787.74 | 11317.00 | 958.34 | 918411.75 | 0.35 |
| Sunspot Annual Measure (SAM) | 289 | 0.00 | 48.90 | 190.20 | 39.58 | 1566.47 | 1.00 |
| Hourly Internet Traffic (HIT) | 1657 | 13321.26 | 46397.15 | 125058.79 | 22136.72 | 490034325.27 | 0.84 |
| Daily Minimum Temperature (DMT) | 3650 | 0.00 | 11.19 | 26.30 | 4.06 | 16.50 | 0.18 |
| Mackey Glass (MG) | 480 | 0.00 | 0.57 | 1.00 | 0.25 | 0.06 | -0.40 |
| Logistic Map (LM) | 150 | 0.00 | 0.64 | 1.33 | 0.48 | 0.23 | 0.12 |

Table 2: Neural network parameters for each dataset

| Problem | Input nodes | Delays for TDNN | Hidden nodes | | | | | Output nodes |
|---|---|---|---|---|---|---|---|---|
| | | | FNN | Elman NN | Jordan NN | MRNN | TDNN | |
| MG | 4 | 1 | 11 | 3 | 6 | 2 | 4 | 1 |
| LM | 3 | 4 | 6 | 13 | 6 | 10 | 4 | 1 |
| HIT | 12 | 2 | 3 | 3 | 8 | 3 | 2 | 1 |
| AWS | 12 | 2 | 13 | 4 | 6 | 3 | 2 | 1 |
| USD | 12 | 1 | 2 | 2 | 7 | 2 | 2 | 1 |
| SAM | 10 | 1 | 4 | 3 | 11 | 5 | 2 | 1 |
| HIT | 24 | 1 | 2 | 3 | 3 | 2 | 2 | 1 |
| DMT | 30 | 6 | 3 | 3 | 3 | 3 | 3 | 1 |

The same approach was used in optimizing all parameters for all algorithms used in the study.

A single output node was used for all the NN models (one step ahead forecasting was considered). For each problem, parameters selected for the NNs are listed in Table 2.

Due to the saturation problem caused by bounded activation functions when PSO is used to train FNNs [48], [38], linear activation functions were used in the hidden units of the FNNs. However, modified hyperbolic tangent functions [27] were used in the hidden layer nodes of the RNNs and in the output nodes of all the NN models. The modified hyperbolic tangent function is defined as,

$$f(net) = 1.7159tanh(\frac{2}{3}net) \tag{10}$$

Bounded activation functions were used in the RNNs in order to avoid passing blown-up activations (i.e. large outputs) from the unbounded functions to the context/state layer, since large context/layer inputs may dominate the real NN inputs (which their mean is close to zero).

All NN weights were initialized randomly in the range $[-\frac{1}{\sqrt{fanin}}, \frac{1}{\sqrt{fanin}}]$, where $fanin$ is the number incoming connections to a node. This range was shown to be good for weight initialization [47].

### 3.3 Resilient Propagation Algorithm

Default Rprop parameters were used in this study, since Rprop does not require optimizing parameters to obtain optimal convergence times on most problems [41].

### 3.4 PSO Setup

For all the experiments, a linearly decreasing inertia weight was used, with an initial value of 0.9 and a final value of 0.5. Acceleration coefficients values were fixed at $c_1 = c_2 = 1.49$, based on [15], where it was shown that such

parameter settings give convergent behaviour. Velocity was not constrained and the Von Neumann topology was used since it facilitates diversity [28], [25], which is good for dynamic problems. For each experiment, the swarm size was determined as equal to the total number of particles used in the CQSO in order to facilitate fair comparison.

### 3.5 Cooperative Quantum Particle Swarm Optimzation

In addition to the PSO parameter setup given in Section 3.4, the radius of the quantum cloud and the percentage of quantum particles per swarm were iteratively selected from the ranges given in Table 3 as suggested in [7].

Table 3: PSO parameter ranges considered

| Parameter | Range |
|---|---|
| Quantum radius, $r$ | [0.2, 0.5, 0.8, 1, 2] |
| % of quantum particles | [10, 20, 30, 40, 50] |
| Number of dimensions per group | [4, 6, 8, 10,12] |

For each dataset, the number of subswarms, $k$, in the CQSO was determined as the ratio $[N_w/d]$, where $N_w$ is the total number of weights and biases in the NN and $d$ is the number of weights grouped together. The value of parameter $d$ was iteratively optimized from the range of values given in Table 3. The size of each subswarm was set to 10 particles based on [4], [5].

### 3.6 Simulating Dynamic Environment

For each dataset, performance was investigated under three different dynamic environmental scenarios. The scenarios were simulated using a sliding time window technique. This involves choosing a window of size $w$ and a step value $s$ for sliding the window over the dataset. The window is used to train the NNs for $f$ number of iterations (i.e change frequency) before the window slides over the dataset. The sliding process involves throwing away the $s$ values at the beginning of the window and adding the next $s$ values in the data series to the end of the window. The training and sliding process is repeated until all of the datasets are used. An example to illustrate this process with $w = 6$ and $s = 4$ is given in Figure 2. When the window slides, four values, $\{x_1, x_2, x_3, x_4\}$, are discarded and new values, $\{x_7, x_8, x_9, x_{10}\}$, are added, while $x_5$ and $x_6$ remain in the window.

The step size determines the spatial severity of the change. A small value for $s$ implies a slight change, while a large value implies a drastic change. An algorithm runs on a window for $f$ iterations before the window slides, controlling the temporal severity. Table 4 presents the parameter setup used

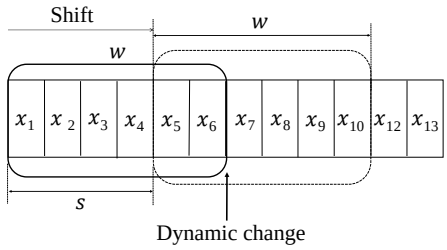



Fig. 2: Sliding Time Window

Table 4: Dynamic scenarios settings for each dataset

| Dataset | Scenario I | | | Scenario II | | | Scenario III | | |
|---|---|---|---|---|---|---|---|---|---|
| | $w$ | $s$ | $f$ | $w$ | $s$ | $f$ | $w$ | $s$ | $f$ |
| MG | 84 | 30 | 50 | 84 | 60 | 100 | 84 | 84 | 200 |
| LM | 31 | 10 | 50 | 31 | 25 | 100 | 31 | 31 | 150 |
| IAP | 32 | 10 | 50 | 32 | 25 | 100 | 32 | 32 | 150 |
| AWS | 42 | 20 | 50 | 42 | 35 | 100 | 42 | 42 | 150 |
| USD | 20 | 8 | 50 | 20 | 16 | 100 | 20 | 20 | 150 |
| SAM | 60 | 20 | 50 | 60 | 40 | 100 | 60 | 60 | 150 |
| HIT | 584 | 250 | 50 | 584 | 500 | 100 | 584 | 584 | 150 |
| DMT | 510 | 200 | 10 | 510 | 400 | 50 | 510 | 510 | 100 |

to simulate the nine different dynamic scenarios for all the problems. As shown in the table, the combination of $s$ and $f$ differs under different scenarios, and therefore requires different numbers of iterations to traverse the entire dataset, calculated as

$$T = f * \frac{N - w}{s} + f \tag{11}$$

### 3.7 Performance Measurement

The collective mean fitness (CMF) [32] was employed as the performance measure for all the experiments. The CMF is given as

$$F_{mean}(t) = \frac{\sum_{t=1}^{T} F(t)}{T} \tag{12}$$

where $F(t)$ is a measure of the quality of the NN and $T$ is the total number of iterations. CMF is commonly used when dynamic environments are considered, because it reflects algorithm performance across the entire range of the landscape dynamics. The mean square error (MSE) calculated over the data set during each iteration was adopted as the quality of the NN.

The generalization factor, $\rho$, was used to quantify the overfitting behavior of the forecasting models. The generalization factor was proposed in [42] and

is defined as $G_E/T_E$, where $T_E$ and $G_E$ are the training and generalization errors respectively. Overfitting is a phenomenon where a NN performs well on training data but poorly on generalization data. A $\rho < 1$ is an indication of good generalization performance while $\rho > 1$ is an indication of overfitting. Since dynamic problems are considered, $\rho$ was calculated using Equation (12), but with $F(t)$ replaced with $\rho(t)$. Thus, all reported values of $\rho$ reflect the generalization factor across the entire algorithm run.

A two tailed Mann Whitney U test [29] was used to check whether the difference in performance of two models was statistically significant. The null hypothesis, $H_0 : \mu_1 = \mu_2$, where $\mu_1$ and $\mu_2$ are the means of the two samples being compared, was evaluated at a significance level of 0.95. The alternative hypothesis is defined as $H_1 : \mu_1 \neq \mu_2$. Thus, any p-value less than 0.05 corresponds to rejection of the null hypothesis that there is no statistically significant difference between the sample means. For the sake of convenience, all p-values were bounded below by 0.0001.

## 4 Experimental Results and Discussion

Tables 5 to 12 summarize the CMF $T_E$ and $G_E$ values with their corresponding confidence intervals (in parenthesis) obtained from the experiments. Minimum error values are printed in bold. Also reported in the tables is the generalization factor, $\rho$. To visualize the results obtained, graphs which are considered as representative or interesting are shown. For convenience, the naming convention Y-X was employed, where Y refers to either Elman, Jordan, MRNN, or TDNN, and X refers to either RPROP, PSO, or CQSO.

As shown in Tables 5 and 6, the results obtained for the MG and HIT problems indicate that the FNN-CQSO model yielded the lowest CMF $T_E$ and $G_E$ values compared to the remaining models for all three dynamic scenarios. For both the MG and HIT problems, all p-values for the pairwise comparison between the FNN-CQSO and the other models were below the 0.05 threshold. This indicates that the difference in performance between the FNN-CQSO and each of the other RNN models were significant for all three scenarios. All the NN models indicate no or slight overfitting.

Figure 3 illustrates the performance progression over time for the FNN-CQSO and the three top performing models, one from each of the RPROP, the PSO and the CQSO trained models for scenario I in predicting the MG time series. Figure 3 shows that all the models had a stable performance progression throughout the experiments, with a slight drop in training errors and a minor increase in the peak of the generalization errors after environmental changes. The figure also shows that the FNN-CQSO model outperformed the other models right from the first epoch to the end.

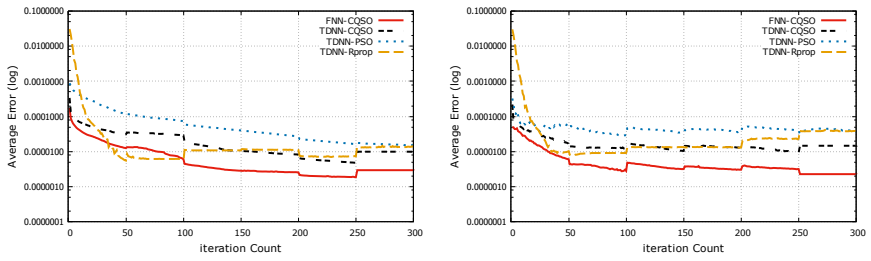
Tables 7 to 10 show that the FNN-CQSO model produced the lowest training errors in predicting the DMT, SAM, LM and AWS time series for all three dynamic scenarios. The FNN-CQSO also yielded the lowest generalization errors in predicting these problems in at least two out of the three scenarios.

Table 5: Results of predicting the MG time series

| Model | Scenario I | | | Scenario II | | | Scenario III | | |
|---|---|---|---|---|---|---|---|---|---|
| | $T_E$ | $G_E$ | $\rho$ | $T_E$ | $G_E$ | $\rho$ | $T_E$ | $G_E$ | $\rho$ |
| FF-CQSO | **5.69E-06** | **3.50E-06** | 0.62 | **5.29E-06** | **3.84E-06** | 0.73 | **4.34E-06** | **4.42E-06** | 1.05 |
| | **(4.29E-07)** | **(2.33E-07)** | (0.02) | **(5.17E-07)** | **(3.51E-07)** | (0.02) | **(3.51E-07)** | **(1.77E-07)** | (0.06) |
| Elman-Rprop | 5.82E-04 | 6.86E-04 | 1.15 | 7.78E-04 | 8.62E-04 | 1.03 | 1.52E-04 | 1.48E-04 | 0.93 |
| | (4.42E-04) | (5.14E-04) | (0.10) | (4.55E-04) | (5.19E-04) | (0.04) | (7.57E-05) | (7.88E-05) | (0.04) |
| Elman-PSO | 7.71E-05 | 4.66E-05 | 0.61 | 6.95E-05 | 4.33E-05 | 0.62 | 5.26E-05 | 2.66E-05 | 0.53 |
| | (6.92E-06) | (6.07E-06) | (0.10) | (6.95E-06) | (5.86E-06) | (0.04) | (9.79E-06) | (3.71E-06) | (0.04) |
| Elman-CQSO | 3.69E-05 | 3.55E-05 | 0.97 | 4.05E-05 | 2.72E-05 | 0.84 | 1.96E-05 | 1.48E-05 | 0.76 |
| | (5.28E-06) | (6.37E-06) | (0.11 ) | (1.80E-05) | (3.34E-06) | (0.07) | (2.24E-06) | (1.59E-06) | (0.04) |
| Jordan-Rprop | 1.99E-01 | 2.02E-01 | 1.06 | 9.91E-02 | 9.35E-02 | 1.04 | 2.71E-01 | 2.77E-01 | 1.03 |
| | (1.86E-01) | (1.89E-01) | 0.06 | (1.32E-01) | (1.25E-01) | (0.04) | (2.17E-01) | (2.20E-01) | (0.05) |
| Jordan-PSO | 7.59E-05 | 4.21E-05 | 0.55 | 5.74E-05 | 3.11E-05 | 0.55 | 4.30E-05 | 2.03E-05 | 0.50 |
| | (5.75E-06) | (4.03E-06) | (0.03) | (4.36E-06) | (3.92E-06) | (0.06) | (7.24E-06) | (2.50E-06) | (0.05) |
| Jordan-CQSO | 1.36E-05 | 1.31E-05 | 0.94 | 1.37E-05 | 1.19E-05 | 0.85 | 8.83E-06 | 6.85E-06 | 0.80 |
| | (1.41E-06) | (2.22E-06) | (0.11) | (1.88E-06) | (2.05E-06) | (0.06) | (8.92E-07) | (4.77E-07) | (0.03) |
| MRNN-Rprop | 9.64E-03 | 9.90E-03 | 1.03 | 1.53E-02 | 1.61E-02 | 1.07 | 9.58E-03 | 1.01E-02 | 1.13 |
| | (2.84E-03) | (2.86E-03) | (0.02) | (6.85E-03) | (7.02E-03) | (0.04) | (5.22E-03) | (5.40E-03) | (0.21) |
| MRNN-PSO | 7.60E-05 | 4.81E-05 | 0.63 | 6.60E-05 | 3.84E-05 | 0.62 | 4.41E-05 | 2.18E-05 | 0.51 |
| | (5.99E-06) | (5.11E-06) | (0.04) | (1.21E-05) | (4.51E-06) | (0.06) | (6.33E-06) | (3.05E-06) | (0.05) |
| MRNN-CQSO | 9.15E-05 | 5.45E-05 | 0.61 | 6.89E-05 | 4.93E-05 | 0.73 | 4.91E-05 | 3.23E-05 | 0.69 |
| | (1.13E-05) | (6.06E-06) | (0.05) | (6.26E-06) | (5.22E-06) | (0.06) | (5.14E-06) | (2.62E-06) | (0.06) |
| TDNN-Rprop | 1.33E-04 | 1.70E-04 | 1.29 | 1.22E-04 | 1.28E-04 | 1.05 | 9.37E-05 | 8.84E-05 | 0.93 |
| | (2.50E-05) | (3.12E-05) | (0.07) | (2.40E-05) | (2.54E-05) | (0.02) | (1.99E-05) | (2.00E-05) | (0.02) |
| TDNN-PSO | 6.83E-05 | 4.46E-05 | 0.64 | 5.65E-05 | 3.05E-05 | 0.54 | 3.42E-05 | 1.79E-05 | 0.53 |
| | (6.71E-06) | (6.34E-06) | (0.05) | (4.29E-06) | (3.07E-06) | (0.03) | (2.76E-06) | (1.37E-06) | (0.02) |
| TDNN-CQSO | 1.81E-05 | 1.61E-05 | 0.91 | 1.43E-05 | 1.21E-05 | 0.86 | 1.00E-05 | 8.42E-06 | 0.85 |
| | (2.97E-06) | (2.48E-06) | (0.04) | (1.26E-06) | (9.63E-07) | (0.04) | (7.41E-07) | (5.85E-07) | (0.04) |

All the NNs models trained using CQSO obtained lower errors compared to the NNs trained using either PSO and Rprop. For these four problems, all p-values for the pairwise comparison between the FNN-CQSO and the other models were below the threshold of 0.05 (which indicates that the difference in performance was significant statistically) except for a few cases.

For the DMT problem, the exceptions are the FNN-CQSO vs Elman-CQSO comparisons in terms of $G_E$ for scenarios I and III, FNN-CQSO vs Jordan-CQSO comparisons for scenarios I and II, and FNN-CQSO vs TDNN-CQSO comparisons for scenario I. For the SAM problem, the exceptions are the FNN-




(a) $T_E$ (b) $G_E$

Fig. 3: Training and generalization error results for MG time series, scenario I

Table 6: Results of predicting the HIT time series

| Model | Scenario I | | | Scenario II | | | Scenario III | | |
|---|---|---|---|---|---|---|---|---|---|
| | $T_E$ | $G_E$ | $\rho$ | $T_E$ | $G_E$ | $\rho$ | $T_E$ | $G_E$ | $\rho$ |
| FF-CQSO | **5.55E-06** | **5.17E-06** | 0.93 | **6.14E-06** | **6.36E-06** | 1.04 | **6.18E-06** | **5.65E-06** | 0.92 |
| | **(3.69E-07)** | **(3.21E-07)** | (0.01) | **(3.87E-07)** | **(3.83E-07)** | (0.01) | **(3.93E-07)** | **(3.40E-07)** | (0.01) |
| Elman-Rprop | 4.73E-04 | 4.77E-04 | 1.01 | 3.68E-04 | 3.66E-04 | 1.00 | 3.87E-04 | 3.88E-04 | 1.00 |
| | (1.09E-04) | (1.10E-04) | (0.01) | (9.45E-05) | (9.37E-05) | (0.01) | (1.04E-04) | (1.05E-04) | (0.01) |
| Elman-PSO | 8.26E-05 | 8.37E-05 | 1.01 | 7.42E-05 | 6.25E-05 | 0.84 | 7.02E-05 | 6.55E-05 | 0.92 |
| | (1.04E-05) | (1.14E-05) | (0.01) | (1.02E-05) | (9.03E-06) | (0.01) | (1.02E-05) | (1.20E-05) | (0.01) |
| Elman-CQSO | 1.21E-05 | 1.09E-05 | 0.91 | 1.19E-05 | 1.16E-05 | 0.98 | 1.25E-05 | 1.15E-05 | 0.92 |
| | (8.75E-07) | (7.35E-07) | (0.01) | (7.57E-07) | (6.27E-07) | (0.02) | (1.02E-06) | (8.95E-07) | (0.01) |
| Jordan-Rprop | 9.57E-03 | 9.85E-03 | 0.99 | 5.62E-03 | 5.59E-03 | 1.00 | 1.07E-02 | 1.06E-02 | 0.99 |
| | (5.05E-03) | (5.59E-03) | (0.01) | (2.36E-03) | (2.35E-03) | (0.01) | (4.15E-03) | (4.06E-03) | (0.00) |
| Jordan-PSO | 8.23E-05 | 8.07E-05 | 0.97 | 7.20E-05 | 6.41E-05 | 0.87 | 8.50E-05 | 8.02E-05 | 0.92 |
| | (1.57E-05) | (1.63E-05) | (0.03) | (1.01E-05) | (1.09E-05) | (0.03) | (2.53E-05) | (2.66E-05) | (0.03) |
| Jordan-CQSO | 8.28E-06 | 7.77E-06 | 0.94 | 8.27E-06 | 8.50E-06 | 1.03 | 8.21E-06 | 7.85E-06 | 0.96 |
| | (4.84E-07) | (4.26E-07) | (0.01) | (4.39E-07) | (4.41E-07) | (0.01) | (4.50E-07) | (4.09E-07) | (0.01) |
| MRNN-Rprop | 1.09E-02 | 1.08E-02 | 0.99 | 8.72E-03 | 8.64E-03 | 1.00 | 5.17E-03 | 5.15E-03 | 1.00 |
| | (4.14E-03) | (4.07E-03) | (0.00) | (3.26E-03) | (3.22E-03) | (0.01) | (1.98E-03) | (1.97E-03) | (0.01) |
| MRNN-PSO | 5.46E-05 | 5.09E-05 | 0.93 | 5.23E-05 | 4.66E-05 | 0.87 | 5.07E-05 | 4.46E-05 | 0.87 |
| | (7.59E-06) | (7.06E-06) | (0.02) | (9.93E-06) | (1.06E-05) | (0.03) | (1.27E-05) | (1.18E-05) | (0.02) |
| MRNN-CQSO | 1.01E-05 | 9.35E-06 | 0.93 | 5.23E-05 | 4.66E-05 | 0.87 | 1.12E-05 | 1.07E-05 | 0.96 |
| | (1.04E-06) | (8.81E-07) | (0.01) | (9.93E-06) | (1.06E-05) | (0.03) | (2.39E-06) | (2.10E-06) | (0.01) |
| TDNN-Rprop | 5.57E-04 | 5.54E-04 | 1.00 | 6.56E-04 | 6.46E-04 | 0.99 | 4.50E-04 | 4.33E-04 | 0.98 |
| | (1.50E-04) | (1.48E-04) | (0.01) | (1.99E-04) | (1.95E-04) | (0.01) | (1.37E-04) | (1.32E-04) | (0.02) |
| TDNN-PSO | 1.20E-04 | 1.16E-04 | 0.99 | 1.33E-04 | 1.16E-04 | 0.86 | 9.87E-05 | 9.65E-05 | 0.93 |
| | (3.23E-05) | (2.90E-05) | (0.04) | (4.93E-05) | (4.47E-05) | (0.03) | (2.53E-05) | (3.00E-05) | (0.04) |
| TDNN-CQSO | 3.10E-05 | 2.99E-05 | 0.95 | 1.41E-05 | 1.37E-05 | 1.03 | 1.59E-05 | 1.49E-05 | 0.95 |
| | (3.76E-05) | (3.70E-05) | (0.02) | (5.62E-06) | (3.95E-06) | (0.02) | (7.42E-06) | (6.77E-06) | (0.01) |

CQSO vs Jordan-CQSO comparisons for all three scenarios. For the LM problem, the exceptions are the FNN-CQSO vs Jordan-CQSO comparisons in terms of $G_E$ for the three scenarios, and for scenario III in terms of $T_E$. The other exceptions are when FNN-CQSO is compared to Elman-CQSO, Elman-PSO, Jordan-PSO, MRNN-PSO, TDNN-Rprop, and TDNN-CQSO in terms of $G_E$ for scenario I. For the AWS problem, the exceptions are when FNN-CQSO is compared to Elman-Rprop, Elman-CQSO, Elman-PSO, Jordan-PSO, MRNN-PSO, MRNN-CQSO, TDNN-PSO and TDNN-CQSO in terms of $G_E$ for scenarios II and III. In terms of $T_E$ comparisons, the exceptions are FNN-CQSO vs Elman-Rprop, FNN-CQSO vs Elman-CQSO, and FNN-CQSO vs TDNN-CQSO for scenario III.

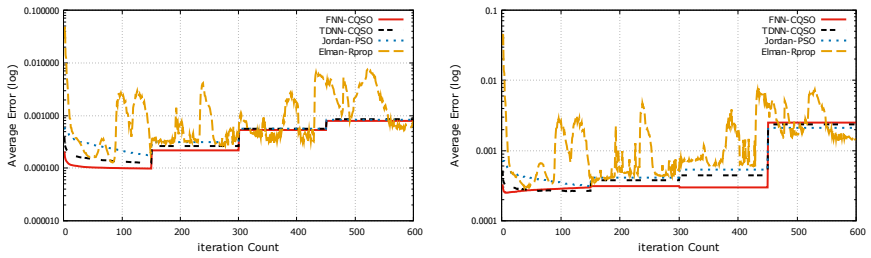
The $\rho$ values for the DMT problem show that all the NN models had slight or no sign of overfitting. For the SAM problem, the NN models trained using CQSO did not overfit for all scenarios, while the models trained using either PSO or Rprop showed slight signs of overfitting. For the LM problem, all the models overfitted for the slightly changing scenario I. The models, however, showed only slight overfitting behaviour for scenario II. For scenario III, where changes are abrupt, the models trained using CQSO and PSO showed no sign of overfitting, while the models trained using Rprop slightly overfitted. For the AWS problem, all the models overfitted.

Figure 4 shows the performance progression over time for the four models that achieved the best results in predicting the AWS time series for scenario III. As visualized in the figure, while the Elman-RPROP model fluctuated a lot, the

Table 7: Results of predicting the DMT time series

| Model | Scenario I | | | Scenario II | | | Scenario III | | |
|---|---|---|---|---|---|---|---|---|---|
| | $T_E$ | $G_E$ | $\rho$ | $T_E$ | $G_E$ | $\rho$ | $T_E$ | $G_E$ | $\rho$ |
| FF-CQSO | **1.19E-05** | **1.24E-05** | 1.04 | **1.09E-05** | **1.30E-05** | 1.20 | **1.08E-05** | 1.17E-05 | 1.09 |
| | **(2.43E-07)** | **(2.75E-07)** | (0.00) | **(1.30E-07)** | **(1.59E-07)** | (0.00) | **(2.97E-07)** | (2.90E-07) | (0.00) |
| Elman-Rprop | 7.29E-04 | 7.29E-04 | 1.00 | 3.42E-04 | 3.44E-04 | 1.01 | 1.88E-04 | 1.88E-04 | 1.00 |
| | (2.35E-04) | (2.36E-04) | (0.01) | (1.04E-04) | (1.03E-04) | (0.01) | (4.75E-05) | (4.72E-05) | (0.01) |
| Elman-PSO | 4.42E-05 | 4.58E-05 | 1.04 | 2.54E-05 | 2.81E-05 | 1.14 | 1.91E-05 | 1.97E-05 | 1.04 |
| | (1.31E-05) | (1.38E-05) | (0.01) | (6.05E-06) | (5.87E-06) | (0.01) | (2.78E-06) | (2.81E-06) | (0.01) |
| Elman-CQSO | 1.35E-05 | 1.38E-05 | 1.02 | 1.13E-05 | 1.34E-05 | 1.19 | 1.09E-05 | **1.17E-05** | 1.07 |
| | (2.05E-06) | (1.94E-06) | (0.01) | (1.39E-07) | (1.74E-07) | (0.01) | (1.33E-07) | **(1.81E-07)** | (0.01) |
| Jordan-Rprop | 1.47E-02 | 1.48E-02 | 1.00 | 5.11E-03 | 4.96E-03 | 0.99 | 3.79E-03 | 3.78E-03 | 1.00 |
| | (5.10E-03) | (5.16E-03) | (0.00) | (1.91E-03) | (1.81E-03) | (0.02) | (9.82E-04) | (9.83E-04) | (0.01) |
| Jordan-PSO | 2.91E-05 | 3.02E-05 | 1.05 | 2.10E-05 | 2.44E-05 | 1.18 | 1.73E-05 | 1.78E-05 | 1.03 |
| | (5.71E-06) | (5.80E-06) | (0.03) | (3.21E-06) | (3.44E-06) | (0.03) | (2.05E-06) | (2.03E-06) | (0.02) |
| Jordan-CQSO | 1.23E-05 | 1.27E-05 | 1.03 | 1.10E-05 | 1.31E-05 | 1.18 | 1.10E-05 | 1.19E-05 | 1.08 |
| | (4.29E-07) | (3.91E-07) | (0.01) | (1.56E-07) | (1.87E-07) | (0.01) | (3.71E-07) | (4.09E-07) | (0.01) |
| MRNN-Rprop | 1.62E-02 | 1.63E-02 | 1.01 | 4.47E-03 | 4.44E-03 | 0.99 | 3.13E-03 | 3.13E-03 | 1.00 |
| | (4.71E-03) | (4.72E-03) | (0.00) | (1.14E-03) | (1.14E-03) | (0.01) | (1.20E-03) | (1.21E-03) | (0.01) |
| MRNN-PSO | 3.18E-05 | 3.19E-05 | 1.02 | 2.27E-05 | 2.60E-05 | 1.17 | 1.54E-05 | 1.62E-05 | 1.05 |
| | (7.96E-06) | (7.33E-06) | (0.03) | (5.45E-06) | (5.10E-06) | (0.03) | (8.06E-07) | (8.61E-07) | (0.02) |
| MRNN-CQSO | 1.28E-05 | 1.31E-05 | 1.03 | 1.20E-05 | 1.40E-05 | 1.18 | 1.11E-05 | 1.19E-05 | 1.08 |
| | (5.76E-07) | (5.28E-07) | (0.01) | (8.11E-07) | (7.08E-07) | (0.01) | (1.10E-07) | (1.65E-07) | (0.01) |
| TDNN-Rprop | 7.99E-04 | 7.81E-04 | 0.98 | 4.85E-04 | 4.48E-04 | 0.97 | 2.97E-04 | 2.90E-04 | 0.99 |
| | (2.40E-04) | (2.33E-04) | (0.04) | (2.14E-04) | (1.87E-04) | (0.02) | (9.35E-05) | (9.15E-05) | (0.01) |
| TDNN-PSO | 4.90E-04 | 4.91E-04 | 1.04 | 4.32E-04 | 4.46E-04 | 1.06 | 3.41E-04 | 3.39E-04 | 0.98 |
| | (2.44E-04) | (2.42E-04) | (0.02) | (2.65E-04) | (2.77E-04) | (0.05) | (1.45E-04) | (1.45E-04) | (0.01) |
| TDNN-CQSO | 1.19E-05 | 1.25E-05 | 1.05 | 1.30E-05 | 1.51E-05 | 1.18 | 1.17E-05 | 1.27E-05 | 1.09 |
| | (2.59E-07) | (2.71E-07) | (0.01) | (2.31E-06) | (2.31E-06) | (0.01) | (8.94E-07) | (8.28E-07) | (0.01) |

other three models had a more stable error progression. Figure 4a shows that the FNN-CQSO had the best $T_E$ progression throughout the search. The $G_E$ progression of the models shown in Figure 4b indicates that the FNN-CQSO had similar or better performance compared to the other models, adapting well to the changes until the last environmental change, where the FNN-CQSO produced slightly the worst error.




(a) $T_E$ (b) $G_E$

Fig. 4: Training and generalization error results for AWS time series, scenario III

Table 8: Results of predicting the SAM time series

| Model | Scenario I | | | Scenario II | | | Scenario III | | |
|---|---|---|---|---|---|---|---|---|---|
| | $T_E$ | $G_E$ | $\rho$ | $T_E$ | $G_E$ | $\rho$ | $T_E$ | $G_E$ | $\rho$ |
| FF-CQSO | **1.45E-04** | **1.27E-04** | 0.88 | **1.49E-04** | 8.13E-05 | 0.55 | **1.24E-04** | **8.93E-05** | 0.72 |
| | **(2.95E-06)** | **(2.36E-06)** | (0.01) | **(2.99E-06)** | (1.03E-06) | (0.01) | **(1.90E-06)** | **(1.57E-06)** | (0.00) |
| Elman-Rprop | 3.77E-04 | 4.18E-04 | 1.14 | 3.14E-04 | 3.62E-04 | 1.17 | 5.02E-04 | 5.13E-04 | 1.04 |
| | (7.47E-05) | (7.37E-05) | (0.04) | (4.94E-05) | (5.31E-05) | (0.06) | (2.02E-04) | (2.01E-04) | (0.02) |
| Elman-PSO | 2.96E-04 | 3.81E-04 | 1.29 | 3.52E-04 | 1.88E-04 | 0.53 | 2.71E-04 | 2.16E-04 | 0.80 |
| | (1.30E-05) | (2.78E-05) | (0.04) | (1.83E-05) | (1.69E-05) | (0.06) | (2.31E-05) | (1.75E-05) | (0.02) |
| Elman-CQSO | 2.41E-04 | 2.49E-04 | 1.02 | 2.63E-04 | 1.37E-04 | 0.53 | 2.13E-04 | 2.04E-04 | 0.96 |
| | (1.27E-05) | (3.17E-05) | (0.10) | (1.29E-05) | (1.14E-05) | (0.06) | (9.74E-06) | (9.73E-06) | (0.03) |
| Jordan-Rprop | 1.89E-02 | 1.88E-02 | 1.12 | 4.32E-03 | 4.52E-03 | 1.09 | 2.89E-03 | 3.08E-03 | 1.09 |
| | (3.41E-02) | (3.36E-02) | (0.05) | (3.94E-03) | (3.99E-03) | (0.03) | (1.45E-03) | (1.55E-03) | (0.05) |
| Jordan-PSO | 4.90E-04 | 6.40E-04 | 1.31 | 6.59E-04 | 4.78E-04 | 0.71 | 6.53E-04 | 6.75E-04 | 1.06 |
| | (3.79E-05) | (8.48E-05) | (0.11) | (8.29E-05) | (7.48E-05) | (0.05) | (1.47E-04) | (1.51E-04) | (0.05) |
| Jordan-CQSO | 1.45E-04 | 1.31E-04 | 0.91 | 1.54E-04 | **7.91E-05** | 0.51 | 1.25E-04 | 9.01E-05 | 0.72 |
| | (5.40E-06) | (5.19E-06) | (0.04) | (3.87E-06) | **(2.59E-06)** | (0.01) | (3.56E-06) | (2.89E-06) | (0.00) |
| MRNN-Rprop | 5.36E-02 | 5.60E-02 | 1.11 | 7.45E-03 | 7.61E-03 | 1.02 | 4.65E-03 | 5.30E-03 | 1.17 |
| | (8.63E-02) | (8.94E-02) | (0.08) | (4.11E-03) | (4.42E-03) | (0.03) | (2.10E-03) | (2.42E-03) | (0.09) |
| MRNN-PSO | 3.91E-04 | 4.97E-04 | 1.28 | 5.03E-04 | 3.15E-04 | 0.64 | 4.00E-04 | 3.61E-04 | 0.91 |
| | (2.32E-05) | (4.12E-05) | (0.09) | (3.84E-05) | (1.99E-05) | (0.04) | (2.69E-05) | (2.21E-05) | (0.05) |
| MRNN-CQSO | 2.20E-04 | 1.89E-04 | 0.86 | 2.38E-04 | 1.10E-04 | 0.46 | 1.85E-04 | 1.52E-04 | 0.83 |
| | (1.00E-05) | (1.88E-05) | (0.06) | (1.28E-05) | (8.67E-06) | (0.02) | (1.06E-05) | (8.78E-06) | (0.04) |
| TDNN-Rprop | 4.03E-04 | 4.36E-04 | 1.11 | 4.07E-04 | 4.67E-04 | 1.21 | 3.65E-04 | 3.72E-04 | 1.05 |
| | (7.96E-05) | (7.95E-05) | (0.03) | (9.01E-05) | (8.75E-05) | (0.05) | (8.38E-05) | (7.81E-05) | (0.03) |
| TDNN-PSO | 3.16E-04 | 3.89E-04 | 1.24 | 3.38E-04 | 1.76E-04 | 0.52 | 2.55E-04 | 2.07E-04 | 0.81 |
| | (1.54E-05) | (2.47E-05) | (0.07) | (1.63E-05) | (1.62E-05) | (0.03) | (9.63E-06) | (1.08E-05) | (0.03) |
| TDNN-CQSO | 2.55E-04 | 2.31E-04 | 0.90 | 2.59E-04 | 1.25E-04 | 0.48 | 2.14E-04 | 1.70E-04 | 0.80 |
| | (1.19E-05) | (2.08E-05) | (0.05) | (9.16E-06) | (7.63E-06) | (0.02) | (5.85E-06) | (9.03E-06) | (0.04) |

Table 11 show that the FNN-CQSO model yielded the lowest CMF $T_E$ and $G_E$ values for scenarios II and III in forecasting the IAP problem. For scenario I, the Elman-Rprop model had the lowest $T_E$, while the Jordan-CQSO model produced the lowest $G_E$. All the p-values for pairwise comparison between the FNN-CQSO and the other models were below the 0.05 threshold, except for the FNN-CQSO vs Jordan-CQSO comparisons, where the two models produced statistically similar performance for all scenarios. All the NN models studied showed some sign of overfitting.

For the USD problem, Table 12 shows that FNN-CQSO produced the lowest $G_E$ for all three scenarios and the lowest $T_E$ for scenario II. The Jordan-CQSO, however, obtained the lowest $T_E$ for scenarios I and II. The $\rho$ values in Table 12 show that all the models exhibited overfitting behaviour. All the p-values for the pairwise comparisons between the FNN-CQSO and the other models were below the 0.05 threshold (which indicate that the difference in performance was statistically significant ), except when compared to:

- Jordan-CQSO (in terms of $T_E$) and Elman-Rprop for scenario I
- Elman-CQSO (in terms of $T_E$) and Jordan-CQSO for scenario II
- Elman-Rprop, TDNN-Rprop and TDNN-CQSO in terms of $T_E$ for scenario III,

Table 9: Results ofpredicting the LM time series

| Model | Scenario I | | | Scenario II | | | Scenario III | | |
|---|---|---|---|---|---|---|---|---|---|
| | $T_E$ | $G_E$ | $\rho$ | $T_E$ | $G_E$ | $\rho$ | $T_E$ | $G_E$ | $\rho$ |
| FF-CQSO | **2.29E-03** | 3.67E-03 | 1.61 | **2.38E-03** | **2.55E-03** | 1.07 | **2.42E-03** | **2.26E-03** | 0.93 |
| | **(7.76E-05)** | (7.04E-05) | (0.06) | **(7.10E-07)** | **(3.54E-06)** | (0.00) | **(7.85E-06)** | **(1.67E-05)** | (0.01) |
| Elman-Rprop | 5.10E-03 | 6.54E-03 | 1.47 | 5.32E-03 | 5.73E-03 | 1.12 | 8.76E-03 | 9.10E-03 | 1.01 |
| | (1.29E-03) | (1.19E-03) | (0.15) | (1.36E-03) | (1.32E-03) | (0.05) | (2.87E-03) | (3.25E-03) | (0.04) |
| Elman-PSO | 3.27E-03 | 3.73E-03 | 1.17 | 3.78E-03 | 2.88E-03 | 0.78 | 3.51E-03 | 2.45E-03 | 0.72 |
| | (2.21E-04) | (2.57E-04) | (0.15) | (2.50E-04) | (1.44E-04) | (0.05) | (2.41E-04) | (1.14E-04) | (0.04) |
| Elman-CQSO | 2.48E-03 | 3.78E-03 | 1.55 | 2.67E-03 | 2.69E-03 | 1.01 | 2.86E-03 | 2.37E-03 | 0.84 |
| | (1.26E-04) | (8.31E-05) | (0.07) | (1.01E-04) | (7.21E-05) | (0.02) | (1.31E-04) | (5.75E-05) | (0.03) |
| Jordan-Rprop | 4.96E-03 | 6.38E-03 | 1.40 | 7.65E-02 | 8.15E-02 | 1.09 | 4.32E-03 | 4.39E-03 | 1.02 |
| | (1.62E-03) | (1.62E-03) | (0.08) | (1.35E-01) | (1.44E-01) | (0.02) | (6.13E-04) | (5.91E-04) | (0.02) |
| Jordan-PSO | 2.62E-03 | 3.63E-03 | 1.41 | 2.54E-03 | 2.59E-03 | 1.02 | 2.47E-03 | 2.30E-03 | 0.93 |
| | (1.28E-04) | (8.90E-05) | (0.07) | (6.58E-05) | (5.00E-05) | (0.02) | (5.24E-05) | (3.56E-05) | (0.02) |
| Jordan-CQSO | 2.54E-03 | 3.76E-03 | 1.51 | 2.44E-03 | 2.58E-03 | 1.06 | 2.42E-03 | 2.28E-03 | 0.94 |
| | (1.51E-04) | (9.30E-05) | (0.08) | (2.45E-05) | (3.52E-05) | (0.01) | (1.81E-05) | (3.50E-05) | (0.01) |
| MRNN-Rprop | 6.52E-03 | 7.65E-03 | 1.22 | 5.99E-03 | 6.38E-03 | 1.07 | 4.94E-03 | 5.04E-03 | 1.02 |
| | (1.33E-03) | (1.29E-03) | (0.05) | (9.01E-04) | (9.13E-04) | (0.02) | (7.75E-04) | (7.92E-04) | (0.02) |
| MRNN-PSO | 3.55E-03 | 4.06E-03 | 1.20 | 3.67E-03 | 2.80E-03 | 0.78 | 3.99E-03 | 2.64E-03 | 0.70 |
| | (3.85E-04) | (4.17E-04) | (0.10) | (2.80E-04) | (1.49E-04) | (0.05) | (4.39E-04) | (2.50E-04) | (0.07) |
| MRNN-CQSO | 2.38E-03 | 3.77E-03 | 1.59 | 2.64E-03 | 2.71E-03 | 1.03 | 2.67E-03 | 2.32E-03 | 0.87 |
| | (6.72E-05) | (6.46E-05) | (0.05) | (5.24E-05) | (4.89E-05) | (0.01) | (8.17E-05) | (5.60E-05) | (0.02) |
| TDNN-Rprop | 2.41E-03 | 3.91E-03 | 1.63 | 2.68E-03 | 3.48E-03 | 1.30 | 3.16E-03 | 3.40E-03 | 1.06 |
| | (2.47E-04) | (3.61E-04) | (0.03) | (2.41E-04) | (3.38E-04) | (0.04) | (5.18E-04) | (6.99E-04) | (0.04) |
| TDNN-PSO | 2.57E-03 | **3.55E-03** | 1.40 | 2.95E-03 | 3.09E-03 | 1.06 | 2.74E-03 | 2.61E-03 | 0.95 |
| | (1.64E-04) | **(1.98E-04)** | (0.07) | (2.63E-04) | (2.14E-04) | (0.04) | (1.53E-04) | (1.76E-04) | (0.03) |
| TDNN-CQSO | 2.77E-03 | 3.80E-03 | 1.38 | 2.60E-03 | 2.83E-03 | 1.09 | 2.69E-03 | 2.28E-03 | 0.85 |
| | (1.08E-04) | (9.71E-05) | (0.06) | (4.85E-05) | (4.90E-05) | (0.03) | (2.91E-05) | (2.63E-05) | (0.01) |

Table 10: Results of predicting the AWS time series

| Model | Scenario I | | | Scenario II | | | Scenario III | | |
|---|---|---|---|---|---|---|---|---|---|
| | $T_E$ | $G_E$ | $\rho$ | $T_E$ | $G_E$ | $\rho$ | $T_E$ | $G_E$ | $\rho$ |
| FF-CQSO | **3.92E-04** | **6.71E-04** | 1.70 | **3.59E-04** | 6.06E-04 | 1.68 | **4.11E-04** | **8.51E-04** | 2.07 |
| | **(2.29E-05)** | **(5.27E-05)** | (0.04) | **(1.45E-05)** | (3.42E-05) | (0.04) | **(1.43E-05)** | **3.34E-05)** | (0.02) |
| Elman-Rprop | 5.71E-04 | 7.27E-04 | 1.33 | 9.53E-04 | 1.32E-03 | 1.7 | 1.53E-03 | 1.80E-03 | 1.71 |
| | (1.10E-04) | (1.07E-04) | (0.07) | (6.51E-04) | (6.27E-04) | (0.13) | (9.33E-04) | (7.71E-04) | (0.19) |
| Elman-PSO | 5.98E-04 | 9.71E-04 | 1.63 | 5.97E-04 | 6.42E-04 | 1.08 | 5.54E-04 | 9.37E-04 | 1.68 |
| | (2.78E-05) | (5.66E-05) | (0.07) | (2.66E-05) | (2.92E-05) | (0.13) | (3.46E-05) | (8.28E-05) | (0.19) |
| Elman-CQSO | 4.31E-04 | 7.38E-04 | 1.70 | 4.52E-04 | 6.28E-04 | 1.38 | 4.69E-04 | 9.29E-04 | 1.96 |
| | (2.37E-05) | (6.53E-05) | (0.08) | (2.76E-05) | (6.59E-05) | (0.09) | (3.77E-05) | (9.39E-05) | (0.05) |
| Jordan-Rprop | 5.61E-03 | 6.08E-03 | 1.15 | 6.22E-02 | 5.72E-02 | 1.42 | 5.47E-02 | 5.74E-02 | 1.27 |
| | (2.29E-03) | (2.49E-03) | (0.05) | (1.16E-01) | (1.05E-01) | (0.17) | (9.59E-02) | (1.00E-01) | (0.12) |
| Jordan-PSO | 5.40E-04 | 9.03E-04 | 1.67 | 5.83E-04 | 6.23E-04 | 1.07 | 4.92E-04 | 8.65E-04 | 1.75 |
| | (1.78E-05) | (5.65E-05) | (0.07) | (2.82E-05) | (3.48E-05) | (0.04) | (2.06E-05) | (5.52E-05) | (0.06) |
| Jordan-CQSO | 4.42E-04 | 7.71E-04 | 1.72 | 4.36E-04 | 6.49E-04 | 1.48 | 4.93E-04 | 9.96E-04 | 1.99 |
| | (2.74E-05) | (7.33E-05) | (0.07) | (2.29E-05) | (5.57E-05) | (0.07) | (3.64E-05) | (1.01E-04) | (0.06) |
| MRNN-Rprop | 1.01E-02 | 1.04E-02 | 1.12 | 1.32E-02 | 1.40E-02 | 1.24 | 7.19E-03 | 8.38E-03 | 1.28 |
| | (6.75E-03) | (6.59E-03) | (0.05) | (5.08E-03) | (5.27E-03) | (0.13) | (3.48E-03) | (3.70E-03) | (0.16) |
| MRNN-PSO | 5.70E-04 | 9.58E-04 | 1.69 | 5.81E-04 | 6.34E-04 | 1.09 | 5.04E-04 | 8.66E-04 | 1.71 |
| | (2.99E-05) | (5.09E-05) | (0.06) | (2.81E-05) | (4.07E-05) | (0.05) | (2.12E-05) | (4.99E-05) | (0.05) |
| MRNN-CQSO | 4.87E-04 | 8.70E-04 | 1.78 | 4.64E-04 | **5.91E-04** | 1.27 | 5.10E-04 | 9.95E-04 | 1.92 |
| | (2.82E-05) | (7.97E-05) | (0.10) | (1.91E-05) | **(4.38E-05)** | (0.07) | (4.12E-05) | (1.17E-04) | (0.08) |
| TDNN-Rprop | 4.11E-03 | 5.91E-03 | 1.43 | 4.83E-03 | 5.71E-03 | 1.16 | 2.12E-03 | 2.68E-03 | 1.48 |
| | (1.32E-03) | (1.78E-03) | (0.12) | (2.90E-03) | (3.22E-03) | (0.07) | (6.95E-04) | (7.22E-04) | (0.10) |
| TDNN-PSO | 5.15E-04 | 9.09E-04 | 1.75 | 5.82E-04 | 6.34E-04 | 1.09 | 4.81E-04 | 8.87E-04 | 1.83 |
| | (2.43E-05) | (6.87E-05) | (0.07) | (2.17E-05) | (3.52E-05) | (0.04) | (2.67E-05) | (7.45E-05) | (0.06) |
| TDNN-CQSO | 4.96E-04 | 9.13E-04 | 1.80 | 5.28E-04 | 6.51E-04 | 1.19 | 4.56E-04 | 8.68E-04 | 1.85 |
| | (3.58E-05) | (1.09E-04) | (0.12) | (4.50E-05) | (1.05E-04) | (0.09) | (4.23E-05) | (1.23E-04) | (0.10) |

Table 11: Results of predicting the AIP time series

| Model | Scenario I | | | Scenario II | | | Scenario III | | |
|---|---|---|---|---|---|---|---|---|---|
| | $T_E$ | $G_E$ | $\rho$ | $T_E$ | $G_E$ | $\rho$ | $T_E$ | $G_E$ | $\rho$ |
| FF-CQSO | 3.10E-04 | 3.60E-04 | 1.21 | **1.30E-04** | **1.70E-04** | 1.54 | **1.00E-04** | **1.28E-04** | 1.33 |
| | (2.00E-04) | (2.40E-04) | (0.06) | **(4.91E-05)** | **(5.01E-05)** | (0.13) | **(2.43E-05)** | **(3.02E-05)** | (0.16) |
| Elman-Rprop | **2.30E-04** | 3.10E-04 | 1.46 | 2.07E-04 | 3.18E-04 | 1.71 | 2.41E-04 | 4.04E-04 | 1.97 |
| | **(4.00E-05)** | (5.00E-05) | (0.11) | (4.75E-05) | (5.71E-05) | (0.15) | (6.30E-05) | (8.08E-05) | (0.29) |
| Elman-PSO | 2.15E-03 | 2.32E-03 | 1.07 | 2.03E-03 | 2.97E-03 | 1.47 | 1.90E-03 | 2.98E-03 | 1.56 |
| | (2.30E-04) | (2.60E-04) | (0.11) | (2.34E-04) | (3.41E-04) | (0.15) | (1.74E-04) | (2.95E-04) | (0.29) |
| Elman-CQSO | 1.75E-03 | 2.15E-03 | 1.23 | 2.28E-03 | 1.94E-03 | 0.83 | 1.81E-03 | 2.90E-03 | 1.60 |
| | (2.30E-04) | (2.80E-04) | (0.03) | (3.37E-04) | (3.32E-04) | (0.04) | (1.88E-04) | (3.10E-04) | (0.01) |
| Jordan-Rprop | 1.45E-01 | 1.47E-01 | 1.05 | 1.09E-02 | 1.09E-02 | 1.23 | 2.07E-02 | 1.98E-02 | 1.09 |
| | (1.24E-01) | (1.25E-01) | (0.02) | (1.00E-02) | (1.00E-02) | (0.09) | (2.72E-02) | (2.49E-02) | (0.06) |
| Jordan-PSO | 2.60E-03 | 2.82E-03 | 1.08 | 2.72E-03 | 3.97E-03 | 1.51 | 1.89E-03 | 3.70E-03 | 2.03 |
| | (5.90E-04) | (6.40E-04) | (0.03) | (5.75E-04) | (7.77E-04) | (0.08) | (3.40E-04) | (5.87E-04) | (0.06) |
| Jordan-CQSO | 2.40E-04 | **2.70E-04** | 1.10 | 1.65E-04 | 2.16E-04 | 1.60 | 1.51E-04 | 2.56E-04 | 1.57 |
| | (7.00E-05) | **(8.00E-05)** | (0.05) | (6.76E-05) | (6.55E-05) | (0.16) | (6.37E-05) | (1.32E-04) | (0.21) |
| MRNN-Rprop | 1.47E-02 | 1.58E-02 | 1.13 | 1.17E-02 | 1.42E-02 | 1.26 | 1.56E-02 | 1.72E-02 | 1.36 |
| | (1.11E-02) | (1.14E-02) | (0.06) | (5.62E-03) | (7.57E-03) | (0.13) | (1.63E-02) | (1.78E-02) | (0.19) |
| MRNN-PSO | 2.09E-03 | 2.27E-03 | 1.08 | 1.83E-03 | 2.70E-03 | 1.49 | 1.55E-03 | 3.04E-03 | 1.99 |
| | (2.20E-04) | (2.50E-04) | (0.01) | (1.93E-04) | (2.72E-04) | (0.05) | (2.34E-04) | (4.23E-04) | (0.03) |
| MRNN-CQSO | 1.36E-03 | 1.49E-03 | 1.09 | 8.74E-04 | 1.17E-03 | 1.38 | 6.99E-04 | 1.42E-03 | 2.11 |
| | (3.50E-04) | (3.90E-04) | (0.02) | (2.64E-04) | (3.58E-04) | (0.10) | (2.61E-04) | (4.83E-04) | (0.04) |
| TDNN-Rprop | 4.30E-04 | 8.60E-04 | 1.93 | 6.53E-04 | 1.28E-03 | 1.88 | 4.50E-04 | 8.71E-04 | 2.00 |
| | (1.10E-04) | (3.10E-04) | (0.20) | (2.57E-04) | (6.92E-04) | (0.16) | (9.44E-05) | (2.42E-04) | (0.26) |
| TDNN-PSO | 2.37E-03 | 2.58E-03 | 1.09 | 2.16E-03 | 3.09E-03 | 1.45 | 1.69E-03 | 3.27E-03 | 1.98 |
| | (3.80E-04) | (4.10E-04) | (0.01) | (2.29E-04) | (3.01E-04) | (0.04) | (2.87E-04) | (5.12E-04) | (0.04) |
| TDNN-CQSO | 3.58E-03 | 4.01E-03 | 1.21 | 2.33E-03 | 2.84E-03 | 1.57 | 2.38E-03 | 4.21E-03 | 1.77 |
| | (1.65E-03) | (1.83E-03) | (0.07) | (1.30E-03) | (1.48E-03) | (0.19) | (1.45E-03) | (2.50E-03) | (0.13) |

Table 12: Results of predicting the USD time series

| Model | Scenario I | | | Scenario II | | | Scenario III | | |
|---|---|---|---|---|---|---|---|---|---|
| | $T_E$ | $G_E$ | $\rho$ | $T_E$ | $G_E$ | $\rho$ | $T_E$ | $G_E$ | $\rho$ |
| FF-CQSO | 4.90E-04 | **1.01E-03** | 2.11 | **3.96E-04** | **6.28E-04** | 1.59 | 4.53E-04 | **1.02E-03** | 2.28 |
| | (2.37E-05) | **(5.77E-05)** | (0.16) | **(1.77E-05)** | **(2.46E-05)** | (0.04) | (2.22E-05) | **(2.39E-05)** | (0.08) |
| Elman-Rprop | 1.02E-03 | 2.06E-03 | 2.91 | 1.11E-03 | 1.55E-03 | 1.69 | 1.65E-03 | 2.18E-03 | 2.28 |
| | (5.95E-04) | (6.74E-04) | (0.42) | (5.78E-04) | (6.40E-04) | (0.15) | (1.18E-03) | (1.16E-03) | (0.33) |
| Elman-PSO | 6.74E-04 | 1.52E-03 | 2.29 | 5.95E-04 | 9.75E-04 | 1.65 | 5.91E-04 | 1.33E-03 | 2.27 |
| | (3.94E-05) | (1.05E-04) | (0.42) | (3.06E-05) | (5.28E-05) | (0.15) | (2.95E-05) | (6.18E-05) | (0.33) |
| Elman-CQSO | 5.44E-04 | 1.30E-03 | 2.39 | 4.10E-04 | 7.15E-04 | 1.75 | 4.91E-04 | 1.20E-03 | 2.46 |
| | (2.98E-05) | (1.25E-04) | (0.17) | (1.94E-05) | (3.28E-05) | (0.07) | (2.23E-05) | (4.97E-05) | (0.12) |
| Jordan-Rprop | 2.75E-03 | 3.64E-03 | 2.03 | 1.74E-03 | 2.30E-03 | 1.81 | 3.59E-03 | 4.29E-03 | 1.45 |
| | (1.39E-03) | (1.59E-03) | (0.31) | (1.03E-03) | (1.12E-03) | (0.16) | (1.48E-03) | (1.62E-03) | (0.21) |
| Jordan-PSO | 7.48E-04 | 1.73E-03 | 2.33 | 6.99E-04 | 1.16E-03 | 1.67 | 6.80E-04 | 1.31E-03 | 1.94 |
| | (3.78E-05) | (1.07E-04) | (0.12) | (3.52E-05) | (6.90E-05) | (0.07) | (2.67E-05) | (5.98E-05) | (0.08) |
| Jordan-CQSO | **4.45E-04** | 1.04E-03 | 2.34 | 4.38E-04 | 6.96E-04 | 1.61 | **3.76E-04** | 1.28E-03 | 3.41 |
| | **(1.81E-05)** | (9.22E-05) | (0.18) | (2.04E-05) | (2.29E-05) | (0.08) | **(1.02E-05)** | (3.65E-05) | (0.10) |
| MRNN-Rprop | 8.52E-03 | 9.43E-03 | 1.77 | 8.76E-03 | 9.49E-03 | 1.43 | 9.84E-03 | 1.03E-02 | 1.15 |
| | (3.17E-03) | (3.22E-03) | (0.39) | (3.92E-03) | (3.84E-03) | (0.18) | (4.17E-03) | (4.08E-03) | (0.11) |
| MRNN-PSO | 6.97E-04 | 1.75E-03 | 2.54 | 6.11E-04 | 1.01E-03 | 1.67 | 5.94E-04 | 1.47E-03 | 2.48 |
| | (4.93E-05) | (1.45E-04) | (0.17) | (4.36E-05) | (5.55E-05) | (0.07) | (2.33E-05) | (7.34E-05) | (0.12) |
| MRNN-CQSO | 6.67E-04 | 1.72E-03 | 2.59 | 4.68E-04 | 7.95E-04 | 1.71 | 5.97E-04 | 1.25E-03 | 2.12 |
| | (4.54E-05) | (1.29E-04) | (0.13) | (1.62E-05) | (3.13E-05) | (0.07) | (3.00E-05) | (6.23E-05) | (0.11) |
| TDNN-Rprop | 8.46E-04 | 2.54E-03 | 3.34 | 7.39E-04 | 1.22E-03 | 1.88 | 5.27E-04 | 1.47E-03 | 3.12 |
| | (1.42E-04) | (2.08E-04) | (0.33) | (1.47E-04) | (1.10E-04) | (0.19) | (8.36E-05) | (1.57E-04) | (0.38) |
| TDNN-PSO | 7.46E-04 | 2.36E-03 | 3.20 | 6.20E-04 | 1.11E-03 | 1.80 | 5.86E-04 | 1.42E-03 | 2.46 |
| | (3.18E-05) | (1.24E-04) | (0.20) | (3.04E-05) | (4.11E-05) | (0.07) | (2.55E-05) | (6.24E-05) | (0.15) |
| TDNN-CQSO | 6.76E-04 | 1.73E-03 | 2.54 | 4.91E-04 | 7.65E-04 | 1.58 | 4.22E-04 | 1.34E-03 | 3.18 |
| | (3.25E-05) | (1.53E-04) | (0.16) | (3.31E-05) | (3.18E-05) | (0.06) | (1.18E-05) | (4.54E-05) | (0.10) |

## 5 Conclusion

The aim of the study was to investigate if using recurrent connections or time delays are unnecessary in neural networks (NNs) used in time series forecasting when a particle swarm optimization (PSO) algorithm designed for dynamic environments is used as the training algorithm. A set of experiments using eight forecasting problems were carried out to test this hypothesis. Feedforward NNs (FNNs) were trained using cooperative quantum particle swarm optimization (CQSO) to forecast each of the problems under three different dynamic scenarios, and the results were compared to those obtained from four different recurrent NNs (RNNs) (i.e. Elman NN, Jordan NN, Multirecurrent NN and Time delay NN), each trained differently with resilient propagation (Rprop), PSO and dynamic PSO algorithms. Mann Whitney U tests were used to check the statistical significance of the difference in performance between the results obtained from the FNN trained with CQSO and each of the remaining NN models.

Analysis of the results showed that the FNN trained with CQSO produced significantly better results compared to each of the RNN models for the eight problems. It was observed that, in general, training the RNNs with CQSO improved performance over training with Rprop or PSO.

The results supported the hypothesis that FNNs trained with a dynamic PSO algorithm is sufficient for time series forecasting in non-stationary environments and the model is able to handle temporal relationships without necessarily introducing recurrent connections.

Further studies will include a detailed study of more variants of the dynamic PSO applied to different forms of RNNs, such as Hopfield and echo state networks. Other properties of training algorithms not explored in this paper, such as recovery speed after a change, time complexity and CPU time will be looked into.

## References

1. Abdulkarim SA (2016) Time Series Prediction with Simple Recurrent Neural Networks. Bayero Journal of Pure and Applied Sciences 9(1):19–24
2. Abdulkarim SA (2018) Time Series Forecasting using Dynamic Particle Swarm Optimizer Trained Neural Networks. Phd thesis, University of Pretoria
3. Adhikari R, Agrawal R (2011) Effectiveness of PSO Based Neural Network for Seasonal Time Series Forecasting. In: Proceedings of the Fifth Indian International Conference on Artificial Intelligence, pp 231–244
4. Van den Bergh F, Engelbrecht A (2001) Effects of Swarm Size on Cooperative Particle Swarm Optimisers. In: Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, pp 892–899

5. Van den Bergh F, Engelbrecht A (2004) A Cooperative Approach to Particle Swarm Optimization. IEEE Transactions on Evolutionary Computation 8(3):225–239
6. Blackwell T, Bentley P (2002) Dynamic Search With Charged Swarms. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp 9–16
7. Blackwell T, Branke J (2006) Multiswarms, Exclusion, and Anti-Convergence in Dynamic Environments. IEEE Transactions on Evolutionary Computation 10(4):459–472
8. Blackwell T, Branke J, Li X (2008) Particle Swarms for Dynamic Optimization Problems. Swarm Intelligence Springer Berlin Heidelberg pp 193–217
9. Chatterjee S, Hore S, Dey N, Chakraborty S, Ashour AS (2017) Dengue Fever Classification using Gene Expression Data: A PSO Based Artificial Neural Network Approach. In: Proceedings of the 5th international conference on frontiers in intelligent computing: theory and applications, Springer, pp 331–341
10. Clerc M, Kennedy J (2002) The Particle Swarm-Explosion , Stability , and Convergence in a Multidimensional Complex Space. IEEE Transaction on Evolutionary Computation 6(1):58–73
11. Deb K, Joshi D, Anand A (2002) Real-coded Evolutionary Algorithms with Parent-centric Recombination. In: Proceedings of Congress on Evolutionary Computation, IEEE, vol 1, pp 61–66
12. Dorffner G (1996) Neural Networks for Time Series Processing. Neural Network World 6:447–468
13. Duhain J (2011) Particle Swarm Optimization in Dynamically Changing Environment An Empirical Study. MSc Thesis, University of Pretoria
14. Eberhart R, Kennedy J (1995) A New Optimizer using Particle Swarm Theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, vol 1, pp 39–43
15. Eberhart R, Shi Y (2000) Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. In: Proceedings of IEEE Congress on Evolutionary Computation, vol 1, pp 84–88
16. Elman J (1990) Finding Structure in Time. Cognitive Science 14:179–211
17. Engelbrecht A (2007) Computational Intelligence: An Introduction. John Wiley & Sons
18. Hamzacebi C (2008) Improving Artificial Neural Networks Performance in Seasonal Time Series Forecasting. Information Sciences 178(23):4550–4559
19. Han HG, Lu W, Hou Y, Qiao JF (2018) An Adaptive-PSO-based Self-organizing RBF Neural Network. IEEE Transactions on Neural Networks and Learning Systems 29(1):104–117
20. Harrison K, Ombuki-berman B, Engelbrecht A (2016) A Radius-Free Quantum Particle Swarm Optimization Technique for Dynamic Optimization Problems. In: Proceedings of IEEE Congress on Evolutionary Computation, pp 578–585

21. Hore S, Chatterjee S, Santhi V, Dey N, Ashour AS, Balas VE, Shi F (2017) Indian Sign Language Recognition Using Optimized Neural Networks. In: Information Technology and Intelligent Transportation Systems, Springer, pp 553–563
22. Hyndman R (2013) Time Series Data Library. Available from Internet: http://robjhyndman com/TSDL (Accessed: 2015-12-05)
23. Jha G, Thulasiraman P, Thulasiram R (2009) PSO Based Neural Network for Time Series Forecasting. In: Proceedings of IEEE International Joint Conference on Neural Networks, pp 1422–1427
24. Jordan M (1986) Attractor Dynamics and Parallellism in a Connectionist Sequential Machine pp 531–546
25. Kennedy J, Mendes R (2002) Population Structure and Particle Swarm Performance. In: Proceedings of IEEE Congress on Evolutionary Computation, vol 2, pp 1671–1676
26. Lawal I, Abdulkarim S, Hassan M, Sadiq J (2016) Improving HSDPA Traffic Forecasting Using Ensemble of Neural Networks. In: Proceedings of 15th IEEE International Conference Machine Learning and Applications, IEEE, pp 308–313
27. LeCun Y, Bottou L, Orr G, Müller K (2012) Efficient Backprop. Neural Networks: Tricks of the Trade Springer Berlin Heidelberg pp 9–48
28. Li X, Dam K (2003) Comparing Particle Swarms for Tracking Extrema in Dynamic Environments. In: IEEE Proceedings on Evolutionary Computation, vol 3, pp 1772–1779
29. Mann H, Whitney D (1947) On a Test of Whether One of Two Random Variables is Stochastically Larger than the Other. Annals of Mathematical Statistics 18(1):50–60
30. Mao C, Lin R, Xu C, He Q (2017) Towards a Trust Prediction Framework for Cloud Services based on PSO-driven Neural Network. IEEE Access 5:2187–2199
31. Mendes R, Cortez P, Rocha M, Neves J (2002) Particle Swarms for Feedforward Neural Network Training. In: Proceedings of IEEE International Joint Conference on Neural Networks, pp 1895–1899
32. Morrison R (2003) Performance Measurement in Dynamic Environments. In: Proceedings of the GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, 5–8
33. Muralitharan K, Sakthivel R, Vishnuvarthan R (2018) Neural Network Based Optimization Approach for Energy Demand Prediction in Smart Grid. Neurocomputing 273:199–208
34. Parsaie A, Azamathulla HM, Haghiabi AH (2018) Prediction of Discharge Coefficient of Cylindrical Weir–gate Using GMDH-PSO. ISH Journal of Hydraulic Engineering 24(2):116–123
35. Rakitianskaia A, Engelbrecht A (2008) Cooperative Charged Particle Swarm Optimiser. In: IEEE Congress on Evolutionary Computation, pp 933–939
36. Rakitianskaia A, Engelbrecht A (2009) Training Neural Networks with PSO in Dynamic Environments. In: Proceedings of IEEE Congress on

Evolutionary Computation, pp 667–673
37. Rakitianskaia A, Engelbrecht A (2012) Training Feedforward Neural Networks with Dynamic Particle Swarm Optimisation. Swarm Intelligence 6(3):233–270
38. Rakitianskaia A, Engelbrecht A (2015) Saturation in PSO Neural Network Training : Good or Evil ? In: IEEE Congress on Evolutionary Computation, vol 2, pp 125–132
39. Rezaee J (2014) Particle Swarm Optimisation for Dynamic Optimisation Problems: A Review. Neural Computing and Applications 25:1507–1516
40. Riedmiller M (1994) Rprop - Description and Implementation Details. Tech. Rep. January
41. Riedmiller M, Braun H (1993) A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In: IEEE International Conference on Neural Networks, pp 586–591
42. Röbel A (1994) The Dynamic Pattern Selection Algorithm : Effective Training and Controlled Generalization of Backpropagation Neural Networks. In: Technical Report, Tech- nische Universität Berlin
43. Tang Z, Fishwick P (1993) Feedforward Neural Nets as Models for Time Series Forecasting. ORSA Journal on Computing 5(4):374—-385
44. Unger N, Ombuki-Berman B, Engelbrecht A (2013) Cooperative Particle Swarm Optimization in Dynamic Environments. In: Proceedings of IEEE Symposium on Swarm Intelligence, pp 172–179
45. Van Den Bergh F (2001) An Analysis of Particle Swarm Optimizers. Phd Thesis, University of Pretoria
46. Waibel A, Hanazawa T, Hinton G, Shikano K, Lang KJ (1990) Phoneme Recognition using Time-delay Neural Networks. In: Readings in Speech Recognition, Elsevier, pp 393–404
47. Wessels L, Barnard E (1992) Avoiding False Local Minima by Proper Initialization of Connections. IEEE Transactions on Neural Networks 3(6):899–905
48. Wyk A, Engelbrecht A (2010) Overfitting by PSO Trained Feedforward Neural Networks. In: IEEE Congress on Evolutionary Computation, pp 1–8