Continuous Emotion Recognition using a Particle Swarm Optimized NARX Neural Network

Ntombikayise Banda
Computer Laboratory
University of Cambridge
Email: nb395@cam.ac.uk

Andries Engelbrecht
Department of Computer Science
University of Pretoria
Email: engel@cs.up.ac.za

Peter Robinson
Computer Laboratory
University of Cambridge
Email: pr10@cam.ac.uk

***Abstract*—The recognition of continuous dimensional emotion remains a challenging task due to large variations in the expression of emotion, and the difficulty of modeling emotion as temporal processes. This work proposes the use of a Nonlinear AutoRegressive with eXogenous inputs recurrent neural network (NARX-RNN) to learn emotional patterns in a given a dataset. The application of particle swarm optimisation in training the NARX-RNN is considered and compared to a gradient descent algorithm. We show that the NARX-RNN outperforms other methods in its emotion recognition ability, and can be easily trained with both gradient-free and gradient-based optimization methods.**

I. Introduction

The emergence of affective computing and subsequent incorporation of emotion recognition systems in computer interfaces has led to powerful systems such as tele-home health care systems that monitor patient health and emotional states and provide empathetic feedback in response to a detected negative emotional state [1].

The emotion recognition problem can be approached as a classification or regression task depending on which of the dominant emotion theories - categorical or dimensional - is followed. The categorical theory views emotions as discrete classes (for example, anger, frustration and happiness) while the dimensional approach conceptualizes emotion as occurring along two primary dimensions, valence and arousal which measure the level of pleasantness and emotional intensity respectively [2]. This work addresses the emotion recognition problem as a regression task based on the valence-arousal dimensions.

It is generally accepted that emotions are processes that unfold over time [3]. This view therefore highlights the need for models to capture the unfolding temporal dynamics of emotion for accurate recognition. Recurrent neural networks have been widely applied in the task of emotion recognition as they have shown great success in representing context through their recurrent connections [4]. The recurrent connections enable the model to retain information about past inputs and to discover temporal correlations between events that are possibly far away from each other in the data [5]. However, basic recurrent neural networks often struggle with the task of learning long term dependencies as they suffer from a problem of *vanishing gradients* [5]. This term refers to a large decrease in the norm of the gradients during training as a result of long-term contributions exponentially decreasing to gradient norm zero when errors are backpropagated, making it impossible for the model to learn correlations between temporally distant events [5]. As a result, numerous variations of the recurrent neural network have been developed to overcome the vanishing gradient problem. These include the Bidirectional Long Short-Term Memory (BLSTM) recurrent neural networks [6] and Echo State Networks [7].

This paper focuses on another class of recurrent neural networks which is based on a discrete nonlinear system called Nonlinear AutoRegressive with eXogenous inputs (NARX) model, and is thus termed the NARX recurrent neural network (NARX-RNN). In their work, Lin *et al* [8] showed that although the NARX-RNN does not circumvent the vanishing gradient problem, it is easier for the NARX-RNN to discover long-term dependencies with gradient-descent. Furthermore, when benchmarked against nine other recurrent neural network architectures, the NARX network was found to converge much faster and generalize better than the other nine networks for grammatical inference and nonlinear system identification problems [9]. The NARX-RNN network has been applied in facial expression analysis to distinguish between seven basic emotions [10] but never applied in the more challenging task of recognizing continuous dimensional emotions.

This paper further looks at a gradient-free optimization technique called Particle Swarm Optimization (PSO) as a way of overcoming the possible vanishing gradient challenges observed in classical recurrent neural networks, and to determine whether gradient descent is truly effective in optimizing the NARX-RNN model. PSO is a population based stochastic optimization technique first introduced by Kennedy and Eberhart [11] based on the movement and intelligence of swarms (e.g. bird flocking). The PSO, in either its original form or variants of it, has been used across a wide range of applications such as neural network training [12], and applied to various affective computing problems as a feature selection mechanism in the recognition of basic emotions from physiological signals [13]–[15], and as a pattern search mechanism that maps facial action units to three of the six basic emotions [16].

The paper therefore provides the following contributions:

- the use of the NARX recurrent neural network for recognition of continuous dimensional emotion (section II),

- optimization of the NARX-RNN using particle swarm theory, which according to the authors' best knowledge has not been investigated yet (section III-B), and
- a comparative analysis of the gradient-optimized and particle swarm optimized NARX-RNN (section IV-E).

II. NARX Recurrent Neural Network

The NARX recurrent neural network (depicted in Figure 1) is a dynamical neural architecture commonly used for input-output modeling of nonlinear dynamical systems. The network takes as input a window of past input and output values and computes the current output.

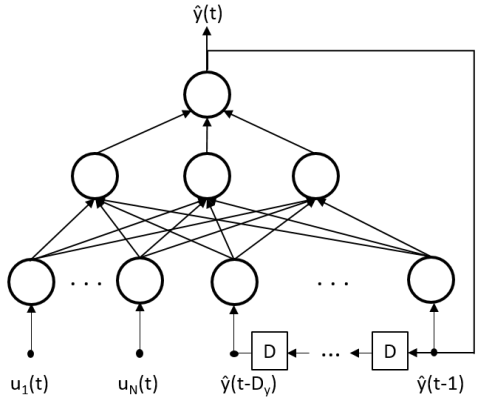

Fig. 1. Architecture of the NARX recurrent neural network

The defining equation for the NARX model is

$$y(t) = f(\mathbf{u}(t-D_u), \cdots, \mathbf{u}(t-1), \mathbf{u}(t), \\ y(t-D_y), \cdots, y(t-1)) \tag{1}$$

where $\mathbf{u}(t) \in \mathbb{R}^N$ and $y(t) \in \mathbb{R}^1$ represent the inputs and output of the network at time t respectively. D_u and D_y are the input and output time lags, and the function f is a nonlinear function approximated by a Multilayer Perceptron (MLP). In this paper we shall consider a NARX network with zero input order ($D_u = 0$) resulting in the network exploring long term dependencies of the output signal only. The MLP consists of three layers, namely, the input, hidden and output layers with recurrent connections from the output to the hidden layer. The architecture of the NARX-RNN can be flattened allowing for the regressed outputs to be concatenated to the exogenous input features and be processed simultaneously. The MLP input at time t is therefore

$$x(t) = [\mathbf{u}(t), y(t-D_y), \cdots, y(t-1)]. \tag{2}$$

The hidden layer consists of neurons which compute a nonlinear or linear transformation of the input depending on the activation function used. The hidden states $h(t)$ and network predictions $\hat{y}(t)$ at time t are computed as follows:

$$h(t) = \Psi(W_{hx} \cdot x(t) + b_h) \tag{3}$$

$$\hat{y}(t) = \Gamma(W_{yh} \cdot h(t) + b_y) \tag{4}$$

where W_{hx} and W_{yh} are the input-to-hidden and hidden-to-output weight matrices respectively, b_h and b_y are the biases, and Ψ and Γ are the activation functions which are both set to hyperbolic tangent functions in this work. The hyperbolic tangent was selected for the output activation function to scale the output between -1 and 1 (similar to the external inputs) prior to re-feeding it into the model as delayed output.

III. NARX-RNN Training

During training, the NARX recurrent neural network aims to learn optimal weight values $\theta = [W_{hx}\ W_{yh}\ b_h\ b_y]$ by minimizing a loss (objective) function defined as $J(\theta) = L(\hat{y}, y)$, where L is a distance function that measures the deviation of the predictions, $\hat{y}$, from the target outputs, y, for each training case c. A typical loss function is the mean squared error (MSE) defined as

$$J(\theta) = \frac{1}{2m} \sum_c \sum_t (y(t) - \hat{y}(t))^2 \tag{5}$$

where m is the total number of time instances over all training cases. A regularization term is added to the loss function to prevent the neural network from overfitting the training data. This is done by scaling the squared weights (excluding the biases) with a user-supplied parameter λ as shown in equation (6).

$$J(\theta) = J(\theta) + \frac{\lambda}{2m} \sum_j \sum_k (w_{jk})^2 \tag{6}$$

Learning of the NARX-RNN parameters can be achieved through backpropagation in conjunction with gradient descent (described in the following sub-section), or by gradient-free methods such as particle swarm optimization described in section III-B.

A. Backpropagation

Gradient descent traverses the error surface (defined by a loss function) whose global minimum translates to optimum weights. At each iteration, the algorithm attempts to move towards this minimum through the following update equation:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \tag{7}$$

where α is the gradient descent learning rate. The weight gradients provide the direction of the search and the learning rate controls the step size of the search.

The partial derivatives of the loss function with respect to the weights can be calculated through backpropagation. For ease of notation and clarity purposes, the inputs to activation functions in equations (3) and (4) are annotated as a_1 and a_2 respectively:

$$a_1(t) = W_{hx} \cdot x(t) + b_h \tag{8}$$
$$a_2(t) = W_{yh} \cdot h(t) + b_y \tag{9}$$

The partial derivative of the loss function with respect to each layer of the MLP is next computed using the chain rule:

$$\frac{\partial J(\theta)}{\partial y} = y(t) - \hat{y}(t) \tag{10}$$
$$\begin{aligned}\frac{\partial J(\theta)}{\partial h} &= \frac{\partial J(\theta)}{\partial y} \cdot \frac{\partial y}{\partial h} \\ &= \frac{\partial J(\theta)}{\partial y} \cdot \text{sech}^2(a_2(t)) \cdot W_{yh}\end{aligned} \tag{11}$$

From this, the partial derivative of the loss function with respect to each weight matrix is obtained as follows:

$$\frac{\partial J(\theta)}{\partial W_{hx}} = \frac{\partial J(\theta)}{\partial h} \cdot \text{sech}^2(a_1(t)) \cdot x(t) \tag{12}$$
$$\frac{\partial J(\theta)}{\partial b_h} = \frac{\partial J(\theta)}{\partial h} \cdot \text{sech}^2(a_1(t)) \tag{13}$$
$$\frac{\partial J(\theta)}{\partial W_{yh}} = \frac{\partial J(\theta)}{\partial y} \cdot \text{sech}^2(a_2(t)) \cdot h(t) \tag{14}$$
$$\frac{\partial J(\theta)}{\partial b_y} = \frac{\partial J(\theta)}{\partial y} \cdot \text{sech}^2(a_2(t)) \tag{15}$$

The weight gradients are then summed over all training instances and converted into a single vector for use in the learning rule update (equation (7)) of the gradient descent algorithm, that is

$$\frac{\partial}{\partial \theta_j} J(\theta) = \Big[\frac{\partial J(\theta)}{\partial W_{hx}} \quad \frac{\partial J(\theta)}{\partial W_{yh}} \quad \frac{\partial J(\theta)}{\partial b_h} \quad \frac{\partial J(\theta)}{\partial b_y}\Big] \tag{16}$$

B. Particle Swarm Optimisation

PSO is used to explore a multidimensional search space of a given problem to find the parameters required to minimize a particular objective function. It does this by maintaining a swarm of particles where each particle represents a potential solution. At the beginning of the search, the system is initialized with a swarm of random solutions (particles). A particle moves through the search space toward an optimum solution by adjusting its position and velocity according to its own previous best performance, and the best performance of its neighbours.

Given a swarm of n_p particles, each particle i consists of a position vector, $\mathbf{x}_i \in \mathbb{R}^{n_x}$, representing parameters of dimension n_x that need to be optimized (e.g. neural network weights) and a velocity vector, $\mathbf{v}_i \in \mathbb{R}^{n_x}$ which governs the direction of travel and step size. The swarm is initialized by setting the positions to $x_{ij} \sim U[-x_{max,j}, x_{max,j}]$, where U is a uniform distribution, and the velocities, v_{ij}, to zero for $j = 1, ..., n_x$.

At each iteration, a particle moves towards the optimum solution using the following velocity and position update equations:

$$\begin{aligned}v_{ij}(t+1) = w v_{ij}(t) &+ c_1 r_{1j}[p_{ij}(t) - x_{ij}(t)] \\ &+ c_2 r_{2j}[g_j(t) - x_{ij}(t)]\end{aligned} \tag{17}$$
$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \tag{18}$$

where

w	is the inertia weight,
c_1, c_2	are positive acceleration constants,
r_{1j}, r_{2j}	are random values sampled from a uniform distribution in the range $(0, 1)$,
p_{ij}	is the personal best position (that is, the best position that particle i in dimension j has visited since the first iteration
g_j	is the global best position (that is, the best position found by the swarm so far) in dimension j.

In its search for the best solution, the swarm varies between exploratory behaviour, that is, searching a broad region of space, and exploitative behaviour, that is, a local search that promotes convergence of the particles towards the best detected solution. The performance of an algorithm is characterized by its exploration-exploitation balance which is achieved by the careful selection of the parameters w, c_1 and c_2.

The PSO algorithm determines the personal best position ($\mathbf{p}_i$) of each particle and global best position of the swarm by evaluating the objective function using the candidate solutions at iteration t. The objective function, f, measures how close a candidate solution is from the optimum solution (as in the case of the mean square error function defined in equation (5)). The result is called the fitness value. Therefore, the personal best position at time $t + 1$, assuming minimisation, is

$$\mathbf{p}_i(t+1) = \begin{cases}\mathbf{p}_i(t) & \text{if} f(\mathbf{x}_i(t+1) \geq f(\mathbf{p}_i(t)) \\ \mathbf{x}_i(t+1) & \text{if} f(\mathbf{x}_i(t+1) < f(\mathbf{p}_i(t))\end{cases} \tag{19}$$

The global best position is obtained from the particle that yields the lowest fitness value amongst all particles, for all previous time steps as shown in equations (20) and (21).

$$\mathbf{g}'_i(t+1) = \underset{\mathbf{p}(t+1)}{\arg\min}\, f(\mathbf{p}(t+1)) \tag{20}$$

where $\mathbf{p}(t+1) \in \{\mathbf{p}_0(t+1), \cdots, \mathbf{p}_i(t+1), \cdots, \mathbf{p}_{n_x}(t+1)\}$.

$$\mathbf{g}_i(t+1) = \begin{cases}\mathbf{g}_i(t) & \text{if} f(\mathbf{g}'_i(t+1) \geq f(\mathbf{g}_i(t)) \\ \mathbf{g}'_i(t+1) & \text{if} f(\mathbf{g}'_i(t+1) < f(\mathbf{g}_i(t))\end{cases} \tag{21}$$

Since the particle velocities are stochastic variables, it is possible for the velocities to grow unbounded, causing the particles to move too far beyond the search space [17]. Velocity clamping was introduced to curb this behaviour by limiting the velocity of each particle in each dimension to $\pm v_{max,j}$, where $v_{max,j} = \mu \times x_{max,j}$ and μ is a scaling

Algorithm 1 : Global Best PSO

```
Initialize swarm size to n_p
Initialize swarm positions to U[-x_{max,j}, x_{max,j}]
Initialize swarm velocities to zero
Initialize personal best positions to swarm positions
repeat
  for each particle i = 1, ..., n_p do
    //Set personal best position
    if f(x_i) < f(p_i) then
      p_i = x_i
    end if

    //Set global best position
    if f(p_i) < f(g) then
      g = p_i
    end if
  end for

  for each particle i = 1, ..., n_p do
    update particle velocity, v_i, using equation (17)
    apply velocity clamping
    update particle position, x_i, using equation (18)
    apply position boundary constraints
  end for
until maximum iterations exceeded or minimum error criteria is met
```

factor which lies in the range $0.1 \leq \mu \leq 1.0$. In the same light, a position constraint rule can be applied to ensure the particle positions stay within the search space. The random initialization is applied in this work which re-initializes the position for each dimension j which violates the boundary condition to $U[-x_{max,j},\, x_{max,j}]$.

The implementation of PSO is summarized in Algorithm 1.

The basic PSO has two general types of neighbourhoods: the star and the ring neighbourhood, resulting respectively in the global best and local best PSO. The particles in the star neighbourhood are interconnected such that each particle has the entire swarm as its neighbourhood and therefore considers information from the entire swarm when making decisions on its search trajectory. The swarm behaviour described in the paragraphs above assumes this topology.

For the ring neighbourhood, the trajectory of each particle is influenced by a small group of particles (a sub-swarm) with different neighbourhoods overlapping. This structure allows for parallel search for an optimum solution, with the sub-swarms slowly converging towards the sub-swarm with the lowest fitness value while thoroughly searching the regions they are currently in. This behaviour reduces the chances of the swarm being trapped into a local minimum [18]. A common configuration of the ring topology is that where each particle is connected to its two adjacent neighbours.

The local best position ($\mathbf{l}_i$) is computed as the best solution found in the neighbourhood N_i of particle i:

$$\mathbf{l}_i(t+1) \in \{\mathcal{N}_i | f(\mathbf{l}(t+1)) = \min\{f(\mathbf{x})\}, \quad \forall \mathbf{x} \in \mathcal{N}_i\} \tag{22}$$

and the resulting velocity update equation is:

$$\begin{aligned} v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}[p_{ij}(t) - x_{ij}(t)] \\ + c_2 r_{2j}[l_{ij}(t) - x_{ij}(t)] \end{aligned} \tag{23}$$

Another neighbourhood configuration that is considered in this study is the toroidal topology where particles are distributed on a grid lattice with each particle connected to its north, east, south and west neighbouring particles as shown in Figure 2(c). The neighbourhood topologies are determined using particle indices and are not based on any spatial information.

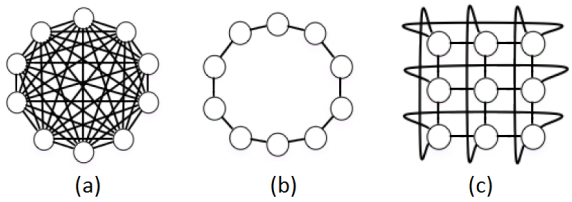

Fig. 2. PSO Neighbourhood Topologies: (a) star topology, (b) ring topology and (c) toroidal topology [19]

IV. Empirical Analysis

The main purpose of the study was to analyse the performance of the gradient descent algorithm on the NARX recurrent neural network, and to compare it to the gradient-free particle swarm optimization algorithm. This section describes the dataset and emotion features used in the empirical analysis, and also details the setup and chosen parameters of the NARX-RNN and of the various optimization algorithms. The results are presented and discussed in subsection IV-E.

A. Emotion Database

The database used in this study was obtained from the AVEC 2012 emotion challenge competition [20]. It contains audiovisual recordings of naturalistic dialogues between two human participants, with one of the participants simulating an artificial listener agent. The emotional content of these videos was annotated by at least two raters along the dimensions arousal, valence, power and expectancy. This study only presents results for the arousal and valence dimensions. It should be noted that this study partitioned the training and test datasets differently to the AVEC 2012 challenge, but in the same manner as the work presented in [21] for comparison.

B. Emotion Features

The emotion recognition task is accomplished by analysing the facial and vocal expressions of the recorded participants.

1) Visual Features: The visual features are obtained by firstly locating and extracting frontal faces in each video frame using the facial feature point from the GAVAM-CLM tracker [22]. The extracted faces are normalized and passed onto a temporal local binary pattern algorithm for feature extraction.

The temporal local binary algorithm used in this work is an extension of the original local binary pattern (LBP) operator [23] which captures the motion and appearance of an image sequence and produces a 1593-dimensional feature descriptor that describes the dynamic textures. The reader is referred to [21] for detailed information on the extraction of the visual features.

Subsequent to the visual feature extraction, principal component analysis (PCA) was applied for dimensionality reduction. Through cross-validation, it was found that reducing the 1593-dimensional feature vector to 20 principal components (which account for approximately 55% of the variance in the data) yielded the best results.

2) Audio Features: Emotion can be detected from speech by analysing the characteristics of speech utterance waveforms. Voice cues such as the pitch and loudness (perceived energy) indicate the nature of the emotion behind an utterance. Additional features adopted in this work are the articulation rate which identifies the number of syllables per second, peak slope which identifies the quality of voice (breathy to tense), and spectral stationarity which captures fluctuations and changes in the voice signal. The extraction details of these features can be found in [24].

C. NARX-RNN Setup

The NARX-RNN model described in section II contains free parameters which were obtained through a grid-search with Pearson's correlation as the evaluation metric (via a 5-fold cross validation exercise). Gradient descent was used as the optimization algorithm.

The parameters and their chosen values are listed in Table I.

TABLE I
SELECTED PARAMETERS FOR NARX-RNN MODELS

PARAMETER	DESCRIPTION	AROUSAL	VALENCE
D_x	Input time lag	0	0
D_y	Output time lag	10	5
n_h	Hidden layer size	50	80
λ	Regularization parameter	16	16

These parameters were used to optimize the weights (θ) of the NARX model.

D. Optimization Algorithms

1) Gradient-based methods: The batch gradient descent algorithm used in the back-propagation training of the NARX-RNN requires the learning rate to be set with care as a very high learning rate could make the algorithm oscillate and become unstable while a small learning rate could lead to slow convergence. The use of an adaptive learning rate has been reported to yield better performance when compared to a constant rate [25]. The learning rate was therefore annealed (gradually lowered) from 0.4 to 0.1 during training.

2) PSO Variants: Three PSO algorithms were implemented using the neighbourhood topologies mentioned in section III-B. These are the star PSO (*starPSO*), the ring topology PSO (*ringPSO*), and the toroidal topology PSO (*torPSO*).

Each PSO was initialized with a swarm of 30 particles and their position boundaries ($[-\mathbf{x}_{max}, \mathbf{x}_{max}]$) were placed at -1 and 1. The inertia weight w in the velocity update equation (17) was set to 0.72 while the acceleration constants (c_1, c_2) were both set to 1.49. These velocity control parameters were shown to lead to convergence of the algorithm [26]. The velocity clamping rule was applied to limit the swarm velocities to 0.2 and 0.3 for the arousal and valence PSO implementations respectively. The velocity limits were determined empirically on the basis of their ability to allow for sufficient exploration of the search space while curbing the unbounded growth of the velocities.

E. Results

The performance of the optimization algorithms is measured using Pearson's correlation coefficient following the evaluation strategy used in the AVEC 2012 challenge. This measurement is obtained by computing the correlation between the emotion predictions and ground truth for each video in the dataset, and then averaging over all videos in a specific dimension. Thirty independent runs were performed for each algorithm, and each run consisted of 1000 iterations. The following correlation results and plots are an average of the thirty independent runs.

1) PSO Convergence Analysis: Figure 3 illustrates the convergence profiles of the different neighbourhood topologies for the valence and arousal dimensions. The ring topology is known to have a slower convergence rate (as observed in the figures), as the best solution found has to propagate through several neighborhoods before affecting all particles in the swarm [27]. The toroidal topology is expected to converge faster than the ring topology due to an increased neighbourhood size but slower than the star topology. However, for the valence dimension, the toroidal topology exhibits a profile similar to the star PSO. As noted in section III-B, the advantage of slower convergence is to avoid being trapped in a local minimum.

2) PSO Diversity Profiles: Additionally, a diversity analysis of the swarms was conducted. Swarm diversity is defined as the degree of dispersion of the particles in the swarm, which is computed as the average distance of the particles from the spatial center of the swarm [28]. This measure quantifies the exploration or exploitation of the swarm. Intuitively, a large diversity value should imply that a large area of the search space is being explored, while a small swarm diversity implies that the particles are exploiting a small area of the search space [28].

In Figure 4, the PSOs begin their searches with high swarm diversities with the star PSO rapidly transitioning to exploitation mode, as opposed to the toroidal and ring PSOs which

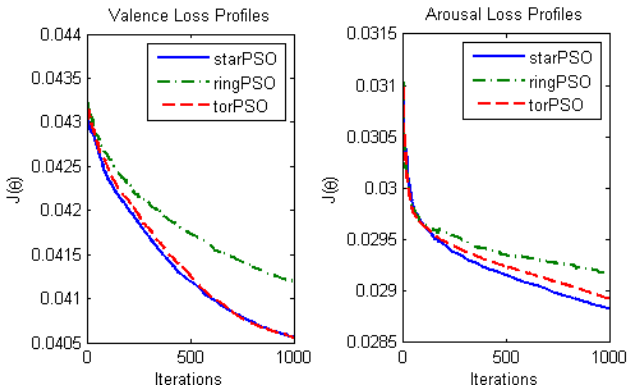

Fig. 3. Loss profiles of the star, ring and toroidal PSOs

maintain a high swarm diversity for longer. The high, relatively unchanging diversity of the ring PSO indicates stagnation which is likely caused by the swarm being trapped in a local minimum. In contrast, the continued downward trend of the star PSO swarm diversity shows that it could still yield better results if the algorithm is run for more than 1000 iterations.

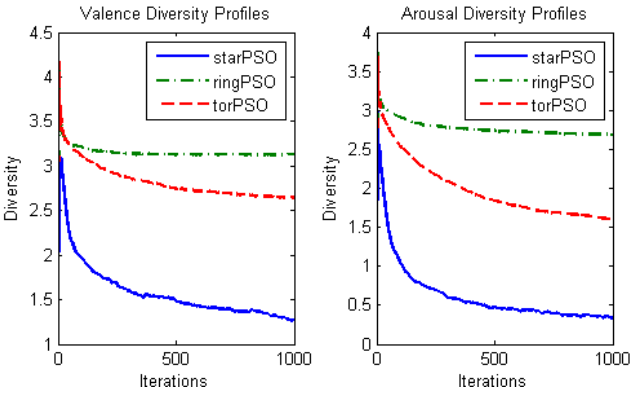

Fig. 4. Diversity profiles of the star, ring and toroidal PSOs

3) Algorithm Comparison: The evaluation results of the three PSOs and the gradient descent algorithm are summarized in Table II. A Mann-Whitney U test ($p < 0.05$) was conducted and indicates that there is no significant difference between the star PSO and gradient descent algorithms for the arousal emotion, and that the star PSO, toroidal PSO and gradient descent significantly outperform the ring PSO. The valence results portray a different picture with the gradient descent significantly outperforming all three PSO algorithms. The star PSO was found to be significantly better than the ring PSO while no significant difference was found between the toroidal and ring PSOs. Engelbrecht [29] argues that the performance of a PSO neighbourhood topology is problem specific, and recommends that a neighbourhood topology be included as one of the tunable parameters.

The correlation results confirm that the gradient descent algorithm is able to successfully train the NARX recurrent neural network without suffering from the explosion or attenuation of weight gradients during training.

TABLE II
CORRELATION RESULTS FOR THE AROUSAL AND VALENCE NARX-RNN MODELS

OPTIMIZER	AROUSAL	VALENCE	MEAN
starPSO	0.418 ± 0.020	0.420 ± 0.020	0.419 ± 0.020
ringPSO	0.379 ± 0.021	0.389 ± 0.046	0.384 ± 0.036
torPSO	0.407 ± 0.023	0.415 ± 0.025	0.411 ± 0.024
Gradient Descent	$\mathbf{0.420 \pm 0.011}$	$\mathbf{0.435 \pm 0.014}$	$\mathbf{0.428 \pm 0.014}$

TABLE III
MODEL COMPARISON BASED ON CORRELATION RESULTS

	AROUSAL	VALENCE	MEAN
SVR	0.241	0.170	0.206
CCRF	0.341	0.326	0.334
CA-CCRF	0.333	0.343	0.338
NARX-RNN	**0.420**	**0.435**	**0.428**

4) Model Comparison: The performance of the NARX-RNN is compared to that of the support vector regression (SVR) model, which serves as the benchmark for the analysis of this emotion dataset, the Continuous Conditional Random Field (CCRF) and Correlation Aware CCRF (CA-CCRF) proposed in [21]. The correlation results are provided in Table III.

The NARX-RNN model yielded a significant increase in the emotion recognition performance when compared to the CCRF and SVR models. The nonlinear mapping of the features to the emotion indicator and the ability of the model to capture temporal dynamics makes the NARX-RNN model a strong predictor of continuous dimensional emotion.

V. CONCLUSION

This paper investigated the use of a NARX-RNN to accurately recognize continuous emotion along the arousal and valence dimensions. Furthermore, the paper set out to test the ability of the gradient descent optimization algorithm to train the NARX recurrent neural network amidst challenges of vanishing gradients that prevent the model from learning long range dependencies. The gradient descent algorithm was compared to the gradient-free particle swarm optimization algorithms of various topologies, namely star, ring and toroidal topologies. The star topology PSO was found to be comparable to the gradient descent algorithm for the arousal dimension, while the ring topology PSO lagged in performance in both emotion dimensions.

The NARX-RNN model overall showed significant improvement in its performance when compared to the CCRF and CA-CCRF models presented in [21], thus demonstrating its strong capability in the capturing and modeling of temporal data.

Future work includes exploring advanced PSO algorithms and differential evolution which can possibly perform better than gradient-based algorithms.

REFERENCES

[1] H. Prendinger and M. Ishizuka, "What affective computing and life-like character technology can do for tele-home health care," in *Proc. Workshop HCI and Homecare*. Citeseer, 2004.

[2] J. A. Russell and L. F. Barrett, "Core affect, prototypical emotional episodes, and other things called emotion: dissecting the elephant." *Journal of personality and social psychology*, vol. 76, no. 5, p. 805, 1999.

[3] J. Gratch, S. Marsella, and P. Petta, "Modeling the cognitive antecedents and consequences of emotion," *Cognitive Systems Research*, vol. 10, no. 1, pp. 1–5, 2009.

[4] Y. Bengio, "Artificial neural networks and their application to sequence recognition," 1991.

[5] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," *arXiv preprint arXiv:1211.5063*, 2012.

[6] M. Wöllmer, A. Metallinou, F. Eyben, B. Schuller, and S. S. Narayanan, "Context-sensitive multimodal emotion recognition from speech and facial expression using bidirectional lstm modeling." in *INTERSPEECH*, 2010, pp. 2362–2365.

[7] S. Scherer, M. Oubbati, F. Schwenker, and G. Palm, "Real-time emotion recognition from speech using echo state networks," in *Artificial neural networks in pattern recognition*. Springer, 2008, pp. 205–216.

[8] T. Lin, B. G. Horne, P. Tino, and C. L. Giles, "Learning long-term dependencies in narx recurrent neural networks," *Neural Networks, IEEE Transactions on*, vol. 7, no. 6, pp. 1329–1338, 1996.

[9] B. G. Horne and C. L. Giles, "An experimental comparison of recurrent neural networks," *Advances in neural information processing systems*, pp. 697–704, 1995.

[10] R. Alazrai and C. G. Lee, "An narx-based approach for human emotion identification," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 4571–4576.

[11] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the sixth international symposium on micro machine and human science*, vol. 1. New York, NY, 1995, pp. 39–43.

[12] R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, vol. 1. IEEE, 2001, pp. 81–86.

[13] G. Wu, G. Liu, and M. Hao, "The analysis of emotion recognition from gsr based on pso," in *Intelligence Information Processing and Trusted Computing (IPTC), 2010 International Symposium on*. IEEE, 2010, pp. 360–363.

[14] C. Defu, L. Guangyuan, and Q. Yuhui, "Applications of particle swarm optimization and k-nearest neighbors to emotion recognition from physiological signals," in *Computational Intelligence and Security, 2008. CIS'08. International Conference on*, vol. 2. IEEE, 2008, pp. 52–56.

[15] Y. Xu, G. Liu, M. Hao, W. Wen, and X. Huang, "Analysis of affective ecg signals toward emotion recognition," *Journal of Electronics (China)*, vol. 27, no. 1, pp. 8–14, 2010.

[16] B. M. Ghandi, R. Nagarajan, and H. Desa, "Particle swarm optimization algorithm for facial emotion detection," in *Industrial Electronics & Applications, 2009. ISIEA 2009. IEEE Symposium on*, vol. 2. IEEE, 2009, pp. 595–599.

[17] A. P. Engelbrecht, *Fundamentals of computational swarm intelligence*. Wiley Chichester, 2005, vol. 1.

[18] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[19] A. J. R. Medina, G. T. Pulido, and J. G. Ramírez-Torres, "A comparative study of neighborhood topologies for particle swarm optimizers." in *IJCCI*, 2009, pp. 152–159.

[20] B. Schuller, M. Valster, F. Eyben, R. Cowie, and M. Pantic, "Avec 2012: the continuous audio/visual emotion challenge," in *Proceedings of the 14th ACM international conference on Multimodal interaction*. ACM, 2012, pp. 449–456.

[21] T. Baltrusaitis, N. Banda, and P. Robinson, "Dimensional affect recognition using continuous conditional random fields," in *Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on*. IEEE, 2013, pp. 1–8.

[22] T. Baltrusaitis, P. Robinson, and L. Morency, "3d constrained local model for rigid and non-rigid facial tracking," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2610–2617.

[23] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[24] D. Ozkan, S. Scherer, and L.-P. Morency, "Step-wise emotion recognition using concatenated-hmm," in *Proceedings of the 14th ACM international conference on Multimodal interaction*. ACM, 2012, pp. 477–484.

[25] M. Moreira and E. Fiesler, "Neural networks with adaptive learning rate and momentum terms," *Technique Report 95*, vol. 4, 1995.

[26] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 1. IEEE, 2000, pp. 84–88.

[27] S. Hamdan, "Hybrid particle swarm optimiser using multi-neighborhood topologies," *INFOCOMP Journal of Computer Science*, vol. 7, no. 1, pp. 36–44, 2008.

[28] O. Olorunda and A. P. Engelbrecht, "Measuring exploration/exploitation in particle swarms using swarm diversity," in *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*. IEEE, 2008, pp. 1128–1134.

[29] A. Engelbrecht, "Particle swarm optimization: Global best or local best?" in *Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI & CBIC), 2013 BRICS Congress on*. IEEE, 2013, pp. 124–135.