Analysis of Activation Functions for Particle Swarm Optimised Feedforward Neural Networks

Andrich B. van Wyk
Department of Computer Science
University of Pretoria
Pretoria, South Africa
avanwyk@cs.up.ac.za

Andries P. Engelbrecht
Department of Computer Science
University of Pretoria
Pretoria, South Africa
engel@cs.up.ac.za

***Abstract*—Previous studies of feedforward neural networks (FFNNs) have found that asymptotically bounded activation functions used by particle swarm optimised (PSO) FFNNs have a significant impact on the swarm behaviour and FFNN performance. A number of alternative activation functions have however been developed that offer potential advantages over popularly used functions. The purpose of this study is to compare the Elliot, rectified linear, leaky rectified linear and softplus functions with the sigmoid and hyperbolic tangent functions on classification and regression problems. It is shown that the rectified linear function has equal performance to the sigmoid and hyperbolic tangent without the disadvantages of the bounded activation functions. Adaptive versions of the functions are compared on unscaled data sets using the PSO *lambda-gamma* algorithm. It is shown that shallower gradients are beneficial to accuracy, but that the FFNNs have inferior generalisation capability when compared to networks trained on scaled data.**

I. Introduction

Previous work that investigated particle swarm optimisation (PSO) as a training algorithm for feedforward neural networks (FFNNs) has shown that, although the PSO is well suited to the task of FFNN training, the activation function used in the hidden and output layers of the FFNN has a significant impact on particle behaviour and ultimately the effectiveness of the optimisation algorithm [1]. Notably, large scale divergence of the swarm was shown to occur, especially in the absence of a correctly adjusted maximum velocity parameter [1], [2].

Specifically, when using a sigmoid or hyperbolic tangent function (tanh), it was hypothesised by van Wyk [1], and confirmed in [3], [4], that saturation of the activation functions occurs at the bounds of the functions where the function gradient is extremely shallow leading to an insignificant change in function output relative to the input [1], [2]. A similar effect was shown to occur if the activation function gradient is adjusted to be more shallow using the function's parameters [1]. The use of the linear activation function, with gradient one, in the hidden layer was proposed as it is unbounded and has a significantly steeper gradient than the sigmoid or tanh function. The linear function was shown to improve convergence at significant cost to the training and generalisation accuracy of the network [1] if using the same network architecture as the sigmoid FFNNs.

An alternative approach is to adapt the activation function as part of the optimisation process [2], [5], [6]. Using an adaptive sigmoid activation function it was shown that PSO training outperforms gradient based training algorithms of FFNNs using the same activation function and that the PSO was further capable of training FFNNs on completely unscaled data [7]. However, the adaptive sigmoid FFNNs performed significantly worse than static activation function networks on scaled data sets [1], [7].

A similar saturation problem has been shown to occur with the use of the sigmoid and tanh functions in deep neural networks (DNNs) [8]. Known as the *vanishing gradient* problem, it was found that extremely shallow gradients (close to 0) would occur in the lower layers of the DNN due to saturation of hyperbolic tangent units in the higher layers of the DNN. The vanishing gradient problem was shown to lead to slower convergence of the DNN and convergence to sub-optimal local minima [9].

Due to these and other findings surrounding the use of the sigmoid or hyperbolic tangent functions [10] a number of alternative activation functions have been developed. The objective of this study is to investigate the use of the Elliot, rectified linear, leaky rectified linear and softplus activation functions with FFNNs as trained by a PSO. The study includes an investigation using scaled data sets as well as adaptation of the functions and training on unscaled data. The empirical results show that the use of the rectified linear function results in FFNNs with accuracy statistically similar to those using a sigmoid or tanh activation functions while also leading to convergent swarm behaviour.

The remainder of this paper is structured as follows: Section II gives background on the algorithms used in this study with specific focus on the selected activation functions. This is followed by the experimental methodology in Section III. The results of the experimental work are given and discussed in Section IV. Finally, the study is concluded in Section V.

II. Background

This section provides background on the PSO algorithm in Section II-A and a brief overview of FFNNs given in Section II-B. Finally, Section II-C discusses each of the activation functions used in the experimental work.

A. Particle Swarm Optimisation

The PSO algorithm is a stochastic, population-based optimisation algorithm developed by Eberhart and Kennedy [11], [12]. Inspired by the complex movement and social interaction of a flock of birds, the algorithm uses a *swarm* of *particles* which are moved through the problem search space, guided by information shared between neighbourhoods of particles, in order to locate optima. The PSO algorithm described in this section is the modified PSO algorithm of Shi and Eberhart that includes the use of an *inertia* term in the velocity update equation [13].

Given an objective function, $f(\mathbf{x})$, each particle represents a candidate solution to the optimisation problem. For a particle i, let the particle's position in the search space at algorithm time step t be defined as $\mathbf{x}_i(t)$. Initialisation of the PSO algorithm typically entails initialising the particle positions randomly throughout the search space: $\mathbf{x}_i(0) \sim U(\mathbf{x}_{min}, \mathbf{x}_{max})^n$, where n is the dimensionality of the search space and $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$ define the minimum and maximum bounds of the search space respectively. The particles are then, at each time step, moved through the search space using the position update equation,

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \tag{1}$$

where $\mathbf{v}_i(t+1)$ is the velocity of particle i at time step $t+1$. The velocity is therefore the driving force behind the optimisation process and is calculated using the velocity update equation,

$$\begin{aligned}\mathbf{v}_i(t+1) = \omega\mathbf{v}_i(t) + c_1\mathbf{r}_1(t)[\mathbf{y}_i(t) - \mathbf{x}_i(t)] \\ + c_2\mathbf{r}_2(t)[\hat{\mathbf{y}}_i(t) - \mathbf{x}(t)]\end{aligned} \tag{2}$$

where ω represents the *inertia* weight. The coefficients c_1 and c_2 respectively represent the cognitive and social accelerations which control the influence of the cognitive, $[\mathbf{y}_i(t) - \mathbf{x}_i(t)]$, and social, $[\hat{\mathbf{y}}_i(t) - \mathbf{x}(t)]$, difference vectors. These terms calculate stochastically weighted difference vectors between the particle's current position and the particle's previous best position, $\mathbf{y}_i(t)$, and the neighbourhood best position, $\hat{\mathbf{y}}_i(t)$, respectively. As determined using the objective function, $f(\mathbf{x}_i(t))$, the neighbourhood best position is the best personal best position of all particles that belong to the same neighbourhood (as discussed below). The vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ are each random numbers sampled from $U(0,1)^n$ which provide the stochastic component to the velocity equation. Effectively, the velocity equation partly maintains the velocity of the previous iteration while drawing the particle to a promising position found personally and that found by other particles in the swarm.

It is possible for velocities to become extremely large, causing particles to leave the feasible search space [14], [15]. Velocity clamping was introduced into the PSO algorithm to prevent such extremely large velocities from occurring [15]. Clamping is performed after the velocity has been calculated as follows:

$$v_{ij}(t+1) = \begin{cases} v'_{ij}(t+1) & \text{if } |v'_{ij}(t+1)| < V_{max,j} \\ -V_{max,j} & \text{if } v'_{ij}(t+1) \leq -V_{max,j} \\ V_{max,j} & \text{if } v'_{ij}(t+1) \geq V_{max,j} \end{cases} \tag{3}$$

where $\mathbf{V}_{max}$ is a problem dependent algorithm control parameter that defines the maximum velocity step sizes.

Particles are organised into social structures referred to as neighbourhood topologies. The neighbourhood topologies dictate the flow of information throughout the swarm by determining the calculation of the neighbourhood best position, $\hat{\mathbf{y}}_i(t)$ [16]. Various such neighbourhood structures exist [11], [17] and have been used to successfully train FFNNs [18], [19], [20], [21], [22], [2], [7]. Notably, the *lbest* topology connects particles in a one-dimensional ring lattice where a particle's neighbourhood is defined as the n_s closest particles (based on index) to the particle on the lattice.

B. Feedforward Neural Networks

Feedforward neural networks are artificial neural networks consisting of multiple, connected layers of neurons. The first such layer is referred to as the input layer with the last layer being the output layer. All other layers are referred to as hidden layers. Each neuron encapsulates an *activation function* which serves as an abstraction for the action potential found in biological neurons. Activations are typically not calculated in the input layer, which simply passes the values from the input vector through to the hidden layer. A typical FFNN consists of three layers (i.e. a single hidden layer) that are fully connected. The hidden and input layers also contain a single *bias* unit each which serves the purpose of adjusting the threshold activation values of units in the subsequent layer [23].

Considering a FFNN with a single hidden layer of size J and an output layer of size K, and given an input vector $\mathbf{z}_p$ of dimensionality I, the FFNN's output is calculated by performing a feedforward pass, as follows:

$$y_{j,p} = f_{y_j}(\sum_{i=1}^{I+1} w_{ji}z_{i,p}), \ \ \forall\, j \in \ \{1, ..., J\} \tag{4}$$

$$o_{k,p} = f_{o_k}(\sum_{j=1}^{J+1} w_{kj}y_{j,p}), \ \ \forall\, k \in \ \{1, ..., K\} \tag{5}$$

where y_j is the j^{th} hidden unit, w_{ji} is the weight from input value z_i to hidden unit y_j; o_k is the k^{th} output unit and w_{kj} is the weight from hidden unit y_j to output unit o_k. Functions f_{o_k} and f_{y_j} represent the activation functions for units o_k and y_j respectively.

C. Activation Functions

Central to the working of the FFNN is the activation function which calculates an activation from the net input signal to a neuron. The net input signal is defined as the weighted sum that serves as input to the activation function as shown in Equations 4 and 5. The activation functions used in this study are discussed below. The parameterized form of each function is given.

Sigmoid: The sigmoid activation function is widely used with FFNNs [23]. The parameterized form is given by:

$$f_{sig}(net, \lambda, \gamma) = \frac{\gamma}{1 + e^{-\lambda(net)}} \tag{6}$$

where λ controls the function's steepness and γ the output range of the function. The function is differentiable, monotonically increasing and yields an output in $(0, \gamma)$. The function has asymptotic bounds at 0 and γ.

Hyperbolic Tangent: The hyperbolic tangent function is given by:

$$f_{tanh}(net, \lambda, \gamma) = \frac{\gamma(e^{\lambda(net)} - e^{-\lambda(net)})}{e^{\lambda(net)} + e^{-\lambda(net)}} \tag{7}$$

where λ controls the steepness and γ the range of the function. Similar to the sigmoid the hyperbolic tangent function is also differentiable and monotonically increasing. The hyperbolic tangent function has a larger output range when compared with the sigmoid function, with the asymptotic bounds limiting the output range to $(-\gamma, \gamma)$.

Elliot: The Elliot activation function is a computationally efficient approximation of the hyperbolic tangent function [10], given by:

$$f_{ell}(net, \lambda, \gamma) = \frac{\gamma\lambda net}{1 + |\lambda(net)|} \tag{8}$$

where λ controls the steepness of the function and γ the range. The function has similar properties and advantages to the hyperbolic tangent function with an output range of $(-\gamma, \gamma)$.

Rectified Linear: The rectified linear function is extremely simple and is given by [24]:

$$f_{ReL}(net, \lambda) = max\{0, \lambda(net)\} \tag{9}$$

where λ controls the steepness, and $net > 0$. Unlike the sigmoid and hyperbolic tangent functions the rectified linear function only has a lower bound and may therefore be used to represent data in the range $[0, \infty)$. Rectified linear function units have been successfully applied in the area of DNNs [25], [9], [26], [27].

Leaky Rectified Linear: The rectified linear function has the potential disadvantage of having a 0 gradient when $net < 0$ making the function unsuited to certain gradient based algorithms [9]. The leaky rectified linear function addresses this issue by allowing for a small non-zero gradient when $net < 0$ [9]:

$$f_{LReL}(net, \lambda) = max(0.01\lambda(net), \lambda(net)) \tag{10}$$

where λ controls the steepness of the function. Although the output range of the leaky rectified linear function is $(-\infty, \infty)$, due to the extremely shallow gradient for $net < 0$ the effective range remains similar to that of the rectified linear function.

Softplus: The softplus activation function is a smooth approximation of the rectified linear function, lacking the discontinuity at $net = 0$ [24]:

$$f_{soft}(net, \lambda) = log(1 + e^{\lambda(net)}) \tag{11}$$

where λ controls the steepness of the function. The softplus function is asymptotically bounded at 0 leading to an output range of $(0, \infty)$.

III. Methodology

This section discusses the methodology used in the experimental work of this study. Two experiments are performed. The first compares PSO training of FFNNs using each of the activation functions discussed in Section II-C on scaled data sets. The second experiment investigates PSO training of FFNNs using adaptive versions of the functions on unscaled data.

The data sets used in the experimental work are given in Section III-A. The method used for training FFNNs using PSO is given in Section III-B with the PSO configuration given in Section III-C. The algorithm measurements and experimental procedure is given in Sections III-D and III-E respectively.

A. Data sets

Five classification and five time series regression data sets were selected for use in the study. The data sets differ in size, number of features, scale of features and complexity. The data sets and selected neural network architectures are given in Table I. The architectures correspond to those used in previous work and as such the results are directly comparable [7].

Table I: Data sets used; number of patterns (P); input (I), hidden (J) and output layer (K) sizes; search space dimensionality (n)

Data Set		P	I	J	K	n
Classification problems						
Glass	UCI MLR [28]	214	9	12	6	133
Iris	UCI MLR [28]	150	4	8	3	49
Diabetes	UCI MLR [28]	768	8	20	2	201
Breast Cancer	UCI MLR [28]	569	30	25	2	801
Wine	UCI MLR [28]	178	13	10	3	151
Regression problems						
Henon Map	Equation (12) [18]	100	2	10	1	41
Logistic Map	Equation (13)	981	4	10	1	61
Mackey-Glass	Equation (14) [29]	981	4	20	1	121
Sunspots(Annual)	NGDC [30]	305	4	10	1	61
TS5	Equation (15)	291	10	5	1	61

The equations for the time series used as regression problems are given below:

- Henon Map:

$$z_t = 1 + 0.3z_{t-2} - 1.4z_{t-1}^2 i,\ z_0 \text{and} z_1 \sim U(-1, 1) \tag{12}$$

- Logistic Map:

$$z_t = rz_{t-1}(1 - z_{t-1});\ z_0 = 0.01,\ r = 4 \tag{13}$$

- Mackey-Glass Equation:

$$z_t = (1 - b)z_{t-1} + a\frac{z_{t-\tau}}{1 + z_{t-\tau}^{10}},\ b = 0.1,\ a = 0.2,\ \tau = 30 \tag{14}$$

- TS5:

$$z_t = 0.3z_{t-6} - 0.6z_{t-4} + 0.5z_{t-1} + 0.3z_{t-6}^2 - 0.2z_{t-4}^2 + n_t;\ n_t \sim N(0, 0.05) \tag{15}$$

For the Henon Map, Logistic Map, Mackey-Glass and TS5 time series the data set patterns were constructed by sampling 1000 points from each series. A training pattern consists of a target value of $t+1$, $t > I$ and the I preceding points in the series as input parameters; I is the size of the input later as given in Table I. The only exception to this procedure was the Mackey-Glass equation where instead the input parameters consist of the t, $t-6$, $t-12$ and $t-18^{th}$ preceding points in the series as in [29].

Where scaling was required for any of the data sets, the following procedure was used: input values were min-max scaled to the range $[-2, 2]$, a rational approximation of the active domain, $[-\sqrt{3}, \sqrt{3}]$, of the sigmoid function [5]. Pattern target values were scaled to the output range of each function: $(0, 1)$ for the sigmoid, rectified linear, leaky rectified linear and softplus functions and $(-1, 1)$ for the hyperbolic tangent and Elliot functions.

In order to determine network generalisation accuracy the data sets were divided in a 2:1 ratio into a training set D_T and a generalisation set D_G. For the classification problems, patterns were assigned to each set at random and the sets shuffled per iteration of the algorithm to prevent inadvertent memorisation of the order of the patterns. Due to the nature of the time series data sets, the patterns were assigned to the training and generalisation sets in order.

B. *PSO FFNN Training*

The particle representation is the serialised weight vector of a FFNN; each particle therefore represents a complete candidate FFNN. The objective function used to evaluate particle positions is the mean squared error over the re-scaled training data set:

$$MSE_R = \frac{\sum_{p=1}^{P} \sum_{k=1}^{K} (t_{rk,p} - o_{rk,p})^2}{PK} \tag{16}$$

where P is the size of the data set, K is the number of output neurons, $t_{rk,p}$ represents the re-scaled (to the original range if scaling was applied) target value for output k and $o_{rk,p}$ the re-scaled output value of output unit k.

For the experimental work with adaptive functions the particle positions (weight vector) were augmented with the λ and γ values for each of the activation functions present in the network [1], [7]:

$$\mathbf{x} = (w_0, ..., w_n, \lambda_0, \lambda_1, ..., \lambda_{J+K}, \gamma_0, \gamma_1, ..., \gamma_{J+K}) \tag{17}$$

where n is the number of network weights and biases, J is the number of hidden units and K is the number of output units. The appropriate function parameters are then used for each activation function when calculating a feedforward pass through the network.

For the rectified linear, leaky rectified linear and softplus activation functions, no γ value is used in the parameterized version of the function. For these functions the position vector only included the weight and λ values: $\mathbf{x} = (\mathbf{w}, \lambda)$.

The PSO algorithm with *lambda* or *lambda-gamma* augmented particle positions are referred to as the PSO-LG algorithm for the remainder of this study.

C. *PSO Configuration*

The PSO algorithm used for the experimental work was an *lbest* PSO with a neighbourhood size of five and a swarm size of 25. These values are consistent with previous works [2], [7] and have been shown to provide a good balance between search space exploration and exploitation when training FFNNs [1]. Particle positions were initialised randomly in the range $[\frac{-1}{\sqrt{fanin}}, \frac{1}{\sqrt{fanin}}]$, where $fanin$ is the number of incoming connections to the corresponding neuron. Weights initialised in this way have been shown to aid in avoiding local minima [31].

For the adaptive activation function work, the λ value sub-vectors were initialised randomly in the range $(0, 0, 2.0]$ allowing exploration of functions with more gradual gradients less than one and steeper gradients larger than one [1]. Considering that the γ value controls the range of the activation functions, the γ sub-vectors were randomly initialised in $(0.0, max]$ where max is the maximum target output value across all patterns in the data set [1].

Particles were initialised with a velocity $\mathbf{v} = \mathbf{0}$ which has been shown to improve optimisation speed compared to other strategies [32].

Table II shows the ω, c_1 and c_2 parameters values used in the experimental work. These values were optimised on a per data set basis for a PSO trained FFNN using a static sigmoid function, but have been successfully used with adaptive activation function FFNNs in previous work [1]. A $\mathbf{V}_{max} = \mathbf{1}$ were used for all data sets which has been shown to allow for exploration of the search space while preventing severe swarm divergence from occurring when training FFNNs [1].

Table II: Optimal values for the ω, c_1 and c_2 parameters per data set.

Data set	ω	c_1	c_2
Diabetes	0.70342	0.22053	0.34541
Glass	0.70342	0.22053	0.34541
Iris	0.85952	1.90634	0.78247
WDBC	0.25075	1.50050	1.50050
Wine	0.59416	1.93756	1.93756
Henon Map	0.71903	0.68881	1.18831
Logistic Map	0.73464	1.15709	1.53172
Mackey-Glass	0.62538	0.25175	1.75025
Sunspot	0.46928	0.18931	1.68781
TS5	0.67220	0.78247	0.65759

Finally, a maximum number of 50000 objective function evaluations was used as the stopping criterion for the PSO algorithms, resulting in 2000 iterations.

D. *Algorithm Measurements*

The following algorithm measurements were calculated during each run of the algorithm:

E_T: The re-scaled mean squared error, as per Equation 16, over the training data set.

E_G: The re-scaled mean squared error, as per Equation 16, over the generalisation data set.

$\mathcal{D}_{avg}$: The swarm diversity as calculated using the *average distance around the swarm centre*:

$$\mathcal{D}_{avg} = \frac{1}{|S|} \sum_{i=1}^{|S|} \sqrt{\sum_{k=1}^{n} (x_{ik} - \bar{x}_k)^2} \tag{18}$$

where $|S|$ is the swarm size. This method for calculating swarm diversity has been found to be robust to outliers while accurately measuring the dispersion of the particles in the search space [33].

E. Experimental Procedure

The experimental work has two primary objectives: first, to compare, in terms of FFNN accuracy and PSO behaviour, each of the activation functions given in Section II-C on *scaled* data sets without any adaptation of the functions. That is, $\lambda = 1$ and $\gamma = 1$ for all activation functions. The second objective is to compare the activation functions using the PSO-LG algorithm on *unscaled* data.

For both experiments, the PSO algorithm and FFNNs were configured as per Section III-C and executed on each of the data sets. Each execution of the algorithm was repeated for 30 samples, where each sample used a distinct and unique seed for a Mersenne Twister PRNG.

Statistical testing was performed at a significance level of $\alpha = 0.05$ to determine if significant differences between any of the activation functions exist using a Friedman test and the following hypothesis:

H_0: There is no significant difference between any of the activation functions.

H_1: There is a one-tailed significant difference, lower values in the case of E_T and E_G and higher values in the case of $\mathcal{D}_{avg}$, between at least one pair of activation functions.

If the alternative hypothesis was accepted based on the results of a Friedman test, pairwise Wilcoxon rank sum tests were performed to further compare the activation functions. The Holm-Bonferroni corrections were used.

IV. Results and Discussion

Results for the scaled data sets are given in Section IV-A and for the unscaled data sets in Section IV-B.

A. Scaled Data Sets

Table III shows the training error results of the FFNNs for each of the activation functions on scaled data sets. A Friedman test yielded a p-value of 0.03179 indicating a significant difference between at least two of the activation functions. The pairwise Wilcoxon rank sum tests for the training error results are shown in Table IV.

Similarly, the generalisation error results for the scaled data sets are given in Table V. A Friedman test yielded a significant result of 6.239×10^{-5} for the generalisation error results. The subsequent pairwise Wilcoxon rank sum tests for the generalisation error results are shown in Table VI.

It can be seen that the Elliot function performed similar to the other activation functions on the training data. However, in terms of generalisation, the Elliot function FFNNs performed significantly worse than both of the two bounded activation functions, sigmoid and tanh. Although the Elliot function might be a computationally efficient approximation of the tanh function, the increased efficiency comes at the cost of decreased FFNN accuracy when trained using a PSO.

The rectified and leaky rectified functions had similar training accuracy as compared to the other activation functions, including each other. No significant difference was found for the training error results. On the generalisation data sets the leaky rectified FFNNs had a significantly worse generalisation accuracy than the softplus and sigmoid function FFNNs.

The rectified FFNNs did not perform significantly worse than any of the other activation functions. Notably, the rectified function FFNNs obtained the lowest generalisation error on the Sunspot data set. Although previous work has shown that the standard linear activation function performed significantly

Table III: E_T results for FFNNs trained on scaled data using PSO. Values indicate mean over 30 samples.

Data Set	Elliot	Leaky	Rectified	Softplus	Sigmoid	TanH
Diabetes	0.144602	0.148344	0.147440	0.148606	0.143351	0.142542
Glass	0.538544	0.649032	0.628334	0.641327	0.554106	0.536826
Iris	0.020940	0.036394	0.036818	0.034035	0.016064	0.020971
WDBC	0.011014	0.034536	0.034187	0.028303	0.016744	0.013397
Wine	0.006946	0.018777	0.019497	0.015995	0.001077	0.001814
Henon Map	0.003407	0.001481	0.001601	0.002430	0.000674	0.001385
Logistic Map	0.001394	0.000601	0.000579	0.000880	0.000748	0.000847
Mackey-Glass	0.001084	0.000539	0.000535	0.000781	0.000598	0.000654
Sunspot	150.665	146.019	149.983	159.754	142.298	139.930
TS5	0.002671	0.002202	0.002199	0.002297	0.002339	0.002307
Friedman p-value: 0.03179						

Table IV: Wilcoxon rank sum tests for E_T results for FFNNs trained on scaled data using PSO.

Data Set	Elliot	Leaky	Rectified	Softplus	Sigmoid
Leaky	1.000	-	-	-	-
Rectified	1.000	1.000	-	-	-
Softplus	1.000	1.000	1.000	-	-
Sigmoid	1.000	0.178	0.178	0.029	-
TanH	0.322	0.322	0.204	0.029	1.000

Table V: E_G results for FFNNs trained on scaled data using PSO. Values indicate mean over 30 samples.

Data Set	Elliot	Leaky	Rectified	Softplus	Sigmoid	TanH
Diabetes	0.164601	0.162724	0.162327	0.161786	0.158753	0.159432
Glass	0.940607	1.001827	0.991323	0.985026	0.857341	0.912302
Iris	0.035235	0.046796	0.048235	0.043692	0.028697	0.033645
WDBC	0.030315	0.040526	0.040566	0.036144	0.027774	0.031882
Wine	0.048241	0.080810	0.081198	0.058950	0.032995	0.042137
Henon Map	0.031035	0.014845	0.008842	0.008698	0.006027	0.023773
Logistic Map	0.002031	0.002006	0.002238	0.001418	0.001327	0.001479
Mackey-Glass	0.001837	0.001561	0.001653	0.001742	0.001312	0.001396
Sunspot	263.809	249.618	215.188	224.298	225.509	234.414
TS5	0.004959	0.005020	0.005769	0.004612	0.004128	0.003888
Friedman p-value: 6.239×10^{-5}						

Table VI: Wilcoxon rank sum tests for E_G results for FFNNs trained on scaled data using PSO.

Data Set	Elliot	Leaky	Rectified	Softplus	Sigmoid
Leaky	1.000	-	-	-	-
Rectified	1.000	1.000	-	-	-
Softplus	1.000	0.025	0.422	-	-
Sigmoid	0.015	0.015	0.420	0.420	-
TanH	0.082	0.150	1.000	1.000	1.000

worse than the sigmoid and tanh functions [1], the same cannot be said of the rectified linear function. When using a PSO as training algorithm, the rectified linear function is therefore a viable unbounded alternative activation function to the bounded sigmoid and tanh function with no cost to accuracy.

The softplus function obtained significantly worse training error results when compared to either the sigmoid or tanh functions. On the generalisation data sets the softplus FFNNs achieved significantly better generalisation errors compared to the leaky rectified FFNNs, but was statistically similar to the other activation functions. Although the softplus FFNNs performed similar to the rectified FFNNs in terms of generalisation, the superior performance of the rectified FFNNs in terms of training error makes it more suitable for use as an activation function for PSO trained FFNNs.

Finally, the diversity results are given in Table VII. A Friedman test yielded a significant result of 0.01794 and the Wilcoxon rank sum tests are given in Table VIII.

Table VII: $\mathcal{D}_{avg}$ results for FFNNs trained on scaled data using PSO. Values indicate mean over 30 samples.

Data Set	Elliot	Leaky	Rectified	Softplus	Sigmoid	TanH
Diabetes	0.194250	0.340981	0.145037	0.319467	0.241396	0.328836
Glass	0.373641	0.227422	0.123325	0.154576	0.097689	0.232896
Iris	8.386465	3.573293	3.632451	4.218275	8.572313	4.757898
WDBC	0.208824	1.390591	1.559138	0.607671	0.443428	1.068307
Wine	10.2817	4.966799	4.759946	6.984028	11.29094	7.589250
Henon Map	3.282252	2.894216	2.356636	3.521299	3.205431	3.470861
Logistic Map	6.655724	2.909669	3.391428	4.978621	7.400112	4.081832
Mackey-Glass	1.970282	0.752058	0.637796	1.449996	1.220130	1.443160
Sunspot	0.104522	0.025510	0.031806	0.115163	0.243351	0.089068
TS5	1.362486	0.707241	0.551576	0.630896	1.506089	0.970597
Friedman p-value: 0.01794						

Table VIII: Wilcoxon rank sum tests for $\mathcal{D}_{avg}$ results for FFNNs trained on scaled data using PSO.

Data Set	Elliot	Leaky	Rectified	Softplus	Sigmoid
Leaky	1.00	-	-	-	-
Rectified	1.00	1.00	-	-	-
Softplus	1.00	0.97	0.26	-	-
Sigmoid	1.00	0.46	0.29	1.00	-
TanH	1.00	0.26	0.15	1.00	1.00

Although the Friedman test result was significant, none of the Wilcoxon rank sum tests yielded a significant result. H_0 is therefore not rejected and no significant difference in $\mathcal{D}_{avg}$ could be found when comparing the activation functions across all data sets. This is likely due to the choice of $\mathbf{V}_{max} = \mathbf{1}$ parameter which has been shown to delay the swarm divergence that has been seen to occur when training FFNNs [1]. Though too large for significance, smaller p-values were obtained when comparing the rectified and leaky rectified functions with other functions indicating marginally improved swarm convergence. Particularly, the rectified linear function obtained the absolute lowest diversity results on the Diabetes, Wine, Henon Map, Mackey-Glass equation and TS5 data sets. This is consistent with findings that have shown that the use of standard linear activation functions have superior convergence behaviour compared to sigmoid and tanh functions [1].

B. Unscaled Data Sets

The training and generalisation error results for the PSO-LG algorithm training FFNNs on unscaled data are given in Tables IX and X respectively. A Friedman test of the training error results yielded a p-value of 0.1301 indicating no significant difference in training performance between the activation functions. The Friedman test for the generalisation error yielded a significant result of 0.04162 with the subsequent Wilcoxon test results given in Table XI.

Table IX: E_T results for FFNNs trained on unscaled data using PSO-LG. Values indicate mean over 30 samples.

Data Set	Elliot	Leaky	Rectified	Softplus	Sigmoid	TanH
Diabetes	0.199879	0.172922	0.173749	0.172385	0.194117	0.196594
Glass	1.264158	0.888775	0.861122	1.071293	0.811909	1.530620
Iris	0.031744	0.036776	0.036477	0.034569	0.025839	0.033725
WDBC	0.047656	0.050185	0.049836	0.048131	0.039920	0.043564
Wine	0.103580	0.062476	0.055148	0.050097	0.029510	0.074949
Henon Map	0.005987	0.071129	0.131878	0.132298	0.018703	0.041112
Logistic Map	0.001057	0.000654	0.000645	0.000904	0.000888	0.000978
Mackey-Glass	0.000811	0.000564	0.000537	0.000755	0.000696	0.000755
Sunspot	644.171	140.152	142.616	137.039	202.375	936.294
TS5	0.002340	0.005050	0.009694	0.009568	0.002228	0.002304
Friedman p-value: 0.1301						

Table X: E_G results for FFNNs trained on unscaled data using PSO-LG. Values indicate mean over 30 samples.

Data Set	Elliot	Leaky	Rectified	Softplus	Sigmoid	TanH
Diabetes	0.21059	0.18314	0.18337	0.18245	0.20830	0.22367
Glass	1.42373	1.02333	1.03642	1.21307	1.02902	1.78148
Iris	0.03722	0.04114	0.04080	0.03805	0.03242	0.03864
WDBC	0.05410	0.05530	0.05400	0.06057	0.04979	0.09761
Wine	0.16804	0.08201	0.08143	0.09446	0.07557	0.14406
Henon Map	0.03607	0.09764	0.14641	0.13770	0.03755	0.08258
Logistic Map	0.00196	0.00155	0.00900	0.00182	0.00151	0.00166
Mackey-Glass	0.00193	0.00159	0.00444	0.00172	0.00153	0.00215
Sunspot	913.715	231.031	255.159	262.111	383.265	1702.62
TS5	0.00421	0.01232	0.01248	0.01692	0.00438	0.00412
Friedman p-value: 0.04162						

None of the activation functions performed exceptionally different from each other, with the exception of the tanh function, which obtained significantly worse generalisation

Table XI: Wilcoxon rank sum tests for E_G results for FFNNs trained on unscaled data using PSO-LG.

Data Set	Elliot	Leaky	Rectified	Softplus	Sigmoid
Leaky	1.00	-	-	-	-
Rectified	1.00	0.785	-	-	-
Softplus	1.00	0.26	1.00	-	-
Sigmoid	1.00	1.00	1.00	1.00	-
TanH	0.546	0.785	1.00	0.801	0.044

accuracy compared to the sigmoid function. Surprisingly, the Elliot function, which is an approximation of the tanh function, did not achieve significantly worse results when compared to the sigmoid function. The Elliot function has a much shallower gradient compared to the tanh function, similar to sigmoid, which is clearly a benefit when used as an adaptive activation function with the PSO-LG algorithm.

Although insignificantly so, performance did vary between the other functions with some functions obtaining much lower errors on specific data sets as shown in Tables IX and X. For example, the Rectified, Leaky Rectified and Softplus functions obtained much lower generalisation errors on the Sunspot data set compared to the bounded activation functions.

Of the three unbounded functions the leaky rectified function did manage to achieve the lowest generalisation errors on most of the data sets, especially the regression data sets. However, as reflected in the significance tests, the margins between the functions were very small.

Considering the comparable performance of the unbounded activation functions and the advantage of a significant reduction in search space dimensionality due to the lack of γ parameters, they provide viable alternatives to the use of adaptive sigmoid functions for learning from unscaled data.

Finally, a comparison was made to the scaled data set results. Statistical testing of all unscaled results with all scaled results is difficult since it would entail more comparisons (six for the unscaled data sets and another six for the scaled) than there are data sets to compare them on. Therefore the rectified and sigmoid functions, having shown the most promise on either scaled and unscaled data, were chosen for comparison. The training and generalisation error results are given in Tables XII and XIV respectively. For both sets of results the Friedman tests yielded significant p-values and as such the Wilcoxon test results are given in Tables XIII and XV.

Considering the training error results in Tables XII and XIII, it was found that both PSO-LG trained FFNNs were outperformed by the PSO trained rectified FFNNs. However, neither of the PSO-LG algorithms obtained significantly worse results than the PSO trained sigmoid FFNNs. Therefore, on training data, the PSO-LG trained FFNNs, especially the rectified FFNNs, trained on unscaled data had very similar performance to sigmoid FFNNs trained on scaled data.

The PSO-LG algorithms were however not as successful when comparing generalisation accuracy. The results shown in Tables XIV and XV show that FFNNs using the sigmoid function and trained on scaled data generalised significantly

Table XII: E_T results for FFNNs trained on both scaled and unscaled data. Values indicate mean over 30 samples.

	PSO-LG *unscaled*		**PSO** *scaled*	
Data Set	**Rectified**	**Sigmoid**	**Rectified**	**Sigmoid**
Diabetes	0.173749	0.194117	0.147440	0.143351
Glass	0.861122	0.811909	0.628334	0.554106
Iris	0.036477	0.025839	0.036818	0.016064
WDBC	0.049836	0.039920	0.034187	0.016744
Wine	0.055148	0.029510	0.019497	0.001077
Henon Map	0.131878	0.018703	0.001601	0.000674
Logistic Map	0.000645	0.000888	0.000579	0.000748
Mackey-Glass	0.000537	0.000696	0.000535	0.000598
Sunspot	142.616	202.375	149.983	142.298
TS5	0.009694	0.002228	0.002199	0.002339
Friedman p-value: 0.006605				

Table XIII: Wilcoxon rank sum tests for E_T results for FFNNs trained on both scaled and unscaled data.

	PSO *scaled*		**PSO-LG** *unscaled*
Data Set	**Rectified**	**Sigmoid**	**Rectified**
PSO Sigmoid	0.055	-	-
PSO-LG Rectified	0.024	0.160	-
PSO-LG Sigmoid	0.018	0.055	0.0812

Table XIV: E_G results for FFNNs trained on both scaled and unscaled data. Values indicate mean over 30 samples.

	PSO-LG *unscaled*		**PSO** *scaled*	
Data Set	**Leaky Rectified**	**Sigmoid**	**Leaky Rectified**	**Sigmoid**
Diabetes	0.183367	0.208305	0.162327	0.158753
Glass	1.036425	1.029021	0.991323	0.857341
Iris	0.040798	0.032419	0.048235	0.028697
WDBC	0.054001	0.049789	0.040566	0.027774
Wine	0.081431	0.075566	0.081198	0.032995
Henon Map	0.146409	0.037549	0.008842	0.006027
Logistic Map	0.008999	0.001509	0.002238	0.001327
Mackey-Glass	0.004437	0.001528	0.001653	0.001312
Sunspot	255.16	383.265	215.188	225.509
TS5	0.012484	0.004382	0.005769	0.004128
Friedman p-value: 0.000125				

Table XV: Wilcoxon rank sum tests for E_G results for FFNNs trained on both scaled and unscaled data.

	PSO *scaled*		**PSO-LG** *unscaled*
Data Set	**Leaky Rectified**	**Sigmoid**	**Leaky Rectified**
PSO Sigmoid	1.00	-	-
PSO-LG Leaky Rectified	0.0391	0.0059	-
PSO-LG Sigmoid	0.4131	0.0059	1.00

better than the FFNNs trained on unscaled data. When comparing only the rectified results on scaled and unscaled data, generalisation was also significantly better on scaled data sets.

Although the PSO-LG trained rectified FFNNs come exceptionally close to achieving similar results on both scaled and unscaled data sets, the increase hardness of training on unscaled data along with the significantly increased dimensionality of the search space data proves a challenge to FFNN generalisation.

V. Conclusion

This paper presented a comparison of a number of alternative activation functions for use when training FFNNs using PSO on scaled and unscaled data sets. It was found that the Elliot function, an approximation of the hyperbolic tangent, had significantly worse performance compared to the tanh function on scaled data sets.

Further it was shown that, on scaled data sets, the rectified linear function has performance equal to that of the sigmoid or tanh functions but is unbounded, avoiding problems with function saturation and gradient.

On unscaled data using adaptive activation functions it was shown that the shallower gradients of the sigmoid and Elliot function outperformed the tanh function FFNNs. In a comparison to scaled results it was found that on unscaled data the rectified FFNNs has comparable training accuracy to scaled data FFNNs, but had significantly worse generalisation accuracy.

Future work should conduct a parameter study using the rectified linear functions which, due to their unbounded nature, could potentially allow larger $\mathbf{V}_{max}$ parameter choices for faster training of FFNNs. A study using larger and more complex data sets could also provide additional insight into the differences between each of the activation functions.

References

[1] A. B. van Wyk, "An Analysis of Overfitting in Particle Swarm Optimized Neural Networks," (Unpublished master's thesis), University of Pretoria, 2015.

[2] A. B. van Wyk and A. P. Engelbrecht, "Overfitting by PSO trained feedforward neural networks," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*, 2010, pp. 1–8.

[3] A. Rakitianskaia and A. Engelbrecht, "Saturation in PSO Neural Network Training: Good or Evil?" 2015.

[4] C. Scheepers and A. Engelbrecht, "Analysis of Stagnation Behaviour of Competitive Coevolutionary Trained Neuro-controllers," 2014.

[5] A. P. Engelbrecht, I. Cloete, J. Geldenhuys, and J. M. Zurada, "Automatic scaling using gamma learning for feedforward neural networks," in *Proceedings of the International Workshop on Artificial Neural Networks*. Springer-Verlag, 1995, pp. 374–381.

[6] J. M. Zurada, "Lambda learning rule for feedforward neural networks," in *IEEE International Conference on Neural Networks*, 1993, pp. 1808–1811.

[7] A. B. van Wyk and A. P. Engelbrecht, "Lambda-gamma learning with feedforward neural networks using particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium*, 2011, pp. 1–8.

[8] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Neural Networks, IEEE Transactions on*, vol. 5, no. 2, pp. 157–166, 1994.

[9] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *ICML Workshop on Deep Learning for Audio, Speech, and Language Processing*, 2013.

[10] D. L. Elliott, "A better activation function for artificial neural networks," 1993.

[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*, 1995, pp. 39–43.

[12] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, 1995.

[13] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, May 1998, pp. 69–73.

[14] M. Clerc and J. Kennedy, "The Particle Swarm — Explosion , Stability , and Convergence in a Multidimensional Complex Space," *Convergence*, vol. 6, no. 1, pp. 58–73, 2002.

[15] R. C. Eberhart, P. K. Simpson, and R. W. Dobbins, *Computational Intelligence PC Tools*. Academic Press Professional, 1996.

[16] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proceedings of the Congress on Evolutionary Computation*, vol. 2, 2002, pp. 1671–1676.

[17] R. Mendes, P. Cortez, M. Rocha, and J. Neves, "Particle swarms for feedforward neural network training," *In Proceedings of the International Joint Conference on Neural Networks*, pp. 1895–1899, 2002.

[18] F. van den Bergh, "An Analysis of Particle Swarm Optimizers," Ph.D. dissertation, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.

[19] F. van den Bergh and A. P. Engelbrecht, "Cooperative Learning in Neural Networks using Particle Swarm Optimizers," *South African Computer Journal*, vol. 26, pp. 84–90, 2000.

[20] ——, "Training product unit networks using cooperative particle swarm optimisers," in *International Joint Conference on Neural Networks*, vol. 1, 2001, pp. 126–131.

[21] A. Ismail and A. P. Engelbrecht, "Global optimization algorithms for training product unit neural networks," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 1, 2000, pp. 132–137 vol.1.

[22] ——, "Pruning product unit neural networks," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 1, 2002, pp. 257–262.

[23] C. M. Bishop, *Neural Networks for Pattern Recognition*. New York, NY, USA: Oxford University Press, Inc., 1995.

[24] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia, "Incorporating second-order functional knowledge for better option pricing," *Advances in Neural Information Processing Systems*, pp. 472–478, 2001.

[25] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8609–8613.

[26] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807–814.

[27] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and Others, "On rectified linear units for speech processing," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3517–3521.

[28] K. Bache and M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[29] J. Platt, "A Resource-Allocating Network for Function Interpolation," *Neural Computation*, vol. 3, no. 2, pp. 213–225, Jun. 1991.

[30] National Geophysical Data Center, "NGDC/WDC STP, Boulder-Sunspot Number Data via FTP from NGDC," November 2009, accessed 4/11/2009 18:06 SAST. [Online]. Available: http://www.ngdc.noaa.gov/stp/ SOLAR/ftpsunspotnumber.html

[31] L. F. A. Wessels and E. Barnard, "Avoiding false local minima by proper initialization of connections," *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 899–905, 1992.

[32] A. P. Engelbrecht, "Particle swarm optimization: Velocity initialization," in *Proceedings of the Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–8.

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