01 Jan 2006

Online Training of a Generalized Neuron with Particle Swarm Optimization

R. Kiran

Ganesh K. Venayagamoorthy
Missouri University of Science and Technology

Sandhya R. Jetti

Recommended Citation

R. Kiran et al., "Online Training of a Generalized Neuron with Particle Swarm Optimization," *Proceedings of the IEEE International Joint Conference on Neural Network, 2006*, Institute of Electrical and Electronics Engineers (IEEE), Jan 2006.
The definitive version is available at https://doi.org/10.1109/IJCNN.2006.247237

2006 International Joint Conference on Neural Networks
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Online Training of a Generalized Neuron with Particle Swarm Optimization

Raveesh Kiran, *Student Member, IEEE*, Sandhya R. Jetti, *Student Member, IEEE* and Ganesh K. Venayagamoorthy, *Senior Member, IEEE*

Abstract— **Neural networks are used in a wide number of fields including signal and image processing, modeling and control and pattern recognition. Some of the most common type of neural networks is the multilayer perceptrons and the recurrent neural networks. Most of these networks consist of large number of neurons and hidden layers, which results in a longer training time. A Generalized Neuron (GN) has a compact structure and overcomes the problem of long training time. Due to its simple structure and lesser memory requirements, the GN is attractive for hardware implementations. This paper presents the online training of a GN with the Particle Swarm Optimization (PSO) algorithm. A comparative study of the GN and the MLP online trained with PSO is presented for function approximations. The GN based identification of the Static VAR Compensator (SVC) dynamics in a 12 bus FACTS benchmark power system trained online with the PSO is also presented.**

I. Introduction

THE role of the neural networks in the present world applications is gradually increasing and as a result faster networks and faster training algorithms are being developed. The conventional neural network that is used generally consists of three layers - input, hidden and output layers. The basic problem faced during the application of neural networks to highly complex problems is large training time due to the number of neurons and the number of layers in the network.

A Generalized Neuron (GN) trained with the Backpropagation (BP) learning algorithm has been shown to overcome some of these drawbacks [1]. Some of the advantages of the GN are that it requires few neurons, i.e. fewer weights. The training time for the network is reduced. This structure requires lesser memory and hardware requirements, thus making it attractive for practical cheap and fast applications.

Particle Swarm Optimization (PSO) technique, which is based on the behavior of a flock of birds or school of fish, is a type of evolutionary computing technique [2], [3]. It has been shown that the PSO training algorithm takes fewer computations and is faster than the BP algorithm for neural networks to achieve the same performance [4].

Since PSO is a population based algorithm, it requires a number of input/output pairs to determine the weights of a neural network. This paper presents the application of the PSO algorithm for online training of the GN. The use of a sliding window with a minimum number of input/output pairs is proposed for online training. The GN is applied for nonlinear static and time varying function approximations, and nonlinear system identification in a power system.

The paper is organized as follows: In section II, the architecture of the GN considered in this paper is explained. In section III, a brief overview of the particle swarm optimization technique is given. Section IV deals with the application of the PSO algorithm for online training of the GN. Simulation results obtained for various nonlinear functions are compared with the results obtained with multilayer perceptron (MLP), both trained with the PSO, in this section. The online identification of the dynamics of a Static Var Compensator (SVC) in a power system with a GN is presented in section V. Finally, the conclusion is given in section VI.

II. Generalized Neuron

The general structure of the common neuron model is an aggregation function and a thresholding function. The general neural network model consists of three distinct layers namely the input layer, the hidden layer and the output layer. Each of these layers consists of a number of simple neurons that are interconnected. There may be more than one hidden layer in cases involving more complex problems. Also the number of neurons in each layer depends on the type of application it is being used for. Thus, it can be seen that as the complexity of the problem increases, the number of neurons and the number of weights to be found also tends to increase. Although the aggregation operators are generally crisp, they overlook the fact that most of the processing in the neural networks is done with incomplete information at hand. The GN model uses partly sum and partly product to take into account the vagueness involved, thus over coming such drawbacks.

The use of a sigmoidal thresholding function and an ordinary product or summation aggregation in the simple neuron model does not always give satisfactory results. This is because real life problems generally involve some amount of nonlinearity. Hence the GN, which has both sigmoidal and the Gaussian functions with weight sharing, can be used to overcome such problems. Due to this the GN has more flexibility and the ability to cope better with the nonlinearity involved in any application. Unlike the thresholding functions like the sigmoidal and the Gaussian functions used here, other functions like sine, cosine, hyperbolic tangent,

The support from the National Science Foundation under the grants: SENSORS ECS # 0529292 and CARRER ECS # 0348221 are gratefully acknowledged for this work.

R. Kiran, S. R. Jetti and G. K. Venayagamoorthy are with the Real-Time Power and Intelligent Systems (RTPIS) Laboratory, University of Missouri, Rolla, MO 65401 USA (e-mail: rkktb@umr.edu, sj7v3@umr.edu, and gkumar@ieee.org).

linear functions, etc can also be used [1].

Unlike the common neuron model which has either $\prod$ (product) or $\sum$ (summation) aggregation function, the GN model has both $\sum$ and $\prod$ aggregation functions. But other fuzzy operators such as the max, min and the compensatory operators can also be used.

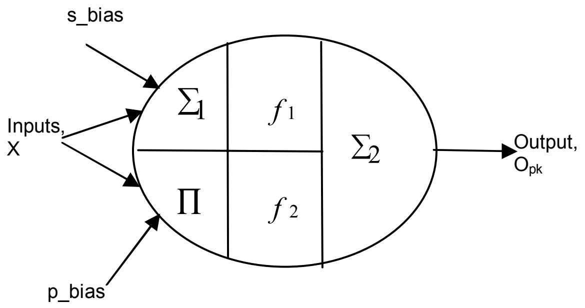

Fig. 1. Generalized neuron model.

The sigmoidal characteristic function (f_1) is used with the $\sum_1$ summation aggregation function while the Gaussian characteristic function (f_2) is used with the $\prod$ product aggregation function. The output of the $\sum_1$ part with the sigmoidal activation function for f_1 of the GN is as shown below [1].

$$O_\Sigma = f_1(s\_net) = \frac{1}{1+e^{(-\lambda_s \times s\_net)}} \tag{1}$$

where,

$$s\_net = \sum W_{\Sigma i} X_i + X_{o\Sigma} \tag{2}$$

and λ_s and $X_{o\Sigma}$ are the gain and bias of $\sum_1$ part respectively.

The output of the $\prod$ part with the Gaussian activation function for f_2 of the GN is as shown below:

$$O_\pi = f_2(pi\_net) = e^{(-\lambda_p \times pi\_net^2)} \tag{3}$$

where,

$$pi\_net = \prod W_{\pi i} X_i \times X_{o\pi} \tag{4}$$

and λ_p and $X_{o\pi}$ are the gain and bias of $\prod$ part respectively.

The final output O_{pk} of the neuron is a function of the two outputs O_Σ and O_π with the weights W and $(1-W)$, respectively, and can be written in the mathematical form [1] as,

$$O_{pk} = O_\pi \times (1-W) + O_\Sigma \times W \tag{5}$$

For multiple output problems, multiple GN models in parallel are required. The number of weights in case of the GN is equal to twice the number of inputs plus one. This is very much lower when compared to the number of weights in a multilayer feedforward neural network. By reducing the number of unknown weights, the training time can be reduced.

III. Particle Swarm Optimization

Particle swarm optimization is a type of evolutionary computing technique. The PSO algorithm is a population-based search algorithm, based on the simulation of the social behavior of birds within a flock. A swarm consists of a set of particles, where each particle represents a potential solution. The changes to the position of a particle and its operation in a swarm are influenced by the experience and the knowledge of its neighbors.

Initially a set of random solutions or a set of particles are considered. A random velocity is given to each particle and they are flown through the problem space. Each particle has memory which is used to keep track of the previous best position and corresponding fitness. The best value of the position of each individual is stored as 'p_{id}'. In other words, 'p_{id}' is the best position acquired by an individual particle during the course of its movement within the swarm. It has another value called the 'p_{gd}', which is the best value of all the particles 'p_{id}' in the swarm. The basic concept of the PSO technique lies in accelerating each particle towards its 'p_{id}' and 'p_{gd}' locations at each time step.

- Fig. 2 briefly illustrates the concept of PSO where,
- x_{id} *(k)* is the current position of i^{th} particle with d dimensions at instant *k.*
- x_{id} *(k+1)* is the position of i^{th} particle with d dimensions at instant *k+1.*
- v_{id} *(k)* is the initial velocity of i^{th} particle with d dimensions at instant *k.*
- v_{id} *(k+1)* is the velocity of i^{th} particle with d dimensions at instant *k+1.*
- w - Inertia weight.
- V_{max} is the maximum velocity for the particles.
- c_1 is the cognition acceleration constant.
- c_2 is the social acceleration constant.

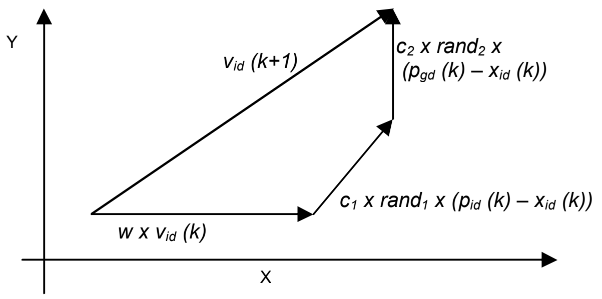
Y
$v_{id}\ (k+1)$
c_2 x $rand_2$ x (p_{gd} (k) – x_{id} (k))
c_1 x $rand_1$ x (p_{id} (k) – x_{id} (k))
w x v_{id} (k)
X

Fig. 2. PSO particle update process illustrated for a two dimensional. case.

i. Initialize a population of particles with random positions and velocities in the problem space.
ii. For each particle, evaluate the desired optimization fitness function.
iii. Compare the particles fitness evaluation with the particles pid If current value is better than the pid

then set p_{id} value equal to the current location.

iv. Compare the best fitness evaluation with the population's overall previous best. If the current value is better than the p_{gd}, then set pgd to the particle's array and index value.

v. Update the particle's velocity and position according to the equations shown below:

The velocity of the i^{th} particle of d dimension is given by:

$$v_{id}(k+1) = w \times v_{id}(k) + c_1 \times rand_1 \times (p_{id}(k) - x_{id}(k)) + c_2 \times rand_2 \times (p_{gd}(k) - x_{id}(k)) \quad (6)$$

The position vector of the i^{th} particle of d dimension is updated as follows:

$$x_{id}(k+1) = x_{id}(k) + v_{id}(k+1) \quad (7)$$

vi. Repeat the step (ii) until a criterion is met, usually a sufficiently good fitness or a maximum number of iterations or epochs.

In case the velocity of the particle exceeds V_{max} then it is reduced to V_{max}. Thus, the resolution and fitness of search depends on the V_{max}. If V_{max} is too high, then particles will move in larger steps and so the solution reached may not be the as good as expected. If V_{max} is too low, then particles will take a long time to reach the desired solution.

IV. NONLINEAR FUNCTION APPROXIMATIONS

Selection of the PSO parameters plays an important role in the optimization of any problem. The selection of the parameters of the GN also plays an important role is getting satisfactory results. The PSO parameters and the parameters of the GN are determined by trail and error. The value of the maximum velocity and the search space limitation depends on the type of problem it is being applied to. The number of particles is also varied. But it is observed that a higher number of particles lead to an increase in the computational time. Hence, a compromise between the computational time and the performance needs to be taken.

The following sets of parameters are used for the nonlinear function approximations [4].

• Maximum velocity, V_{max}	2
• Maximum search space range	(-100,100)
• Inertia weight, w	0.8
• Acceleration constants, c_1 c_2	2, 2
• Size of swarm	25

The parameters of the GN (λ_s, λ_p) are also varied over a wide range and it is found that the values of the parameters are problem dependent. Thus, they are determined by trial and error for each problem.

A. *Nonlinear Static Function*

The GN is online trained with PSO to approximate a nonlinear quadratic function, $y = 2x^2+1$ for values of x in the range of (-1, 1) in increments of 0.01. The values of λ_s and λ_p are taken to be 10 and 1 respectively. Both the biases are taken as 1. Fig. 3 shows the GN output when online trained with a window size of 50 input/output pairs. The window slides in steps of 1. A Mean Square Error (MSE) of 0.006 is obtained. Fig. 4 shows the GN output when online trained with a window size of 50 input/output pairs but the window slides in steps of 50. A MSE of 0.045 is obtained.

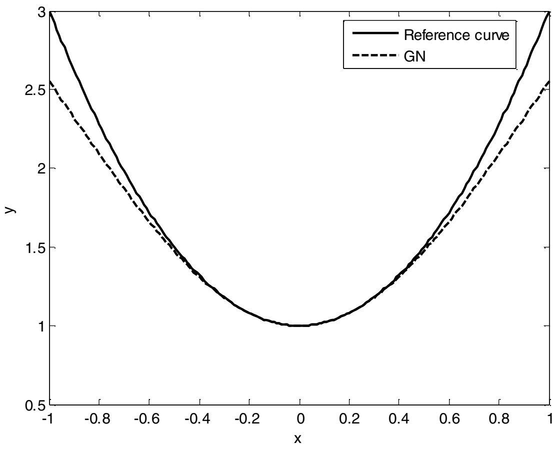

Fig. 3. Output for $y = 2x^2+1$ with GN online trained with PSO with a window sliding in steps of 1.

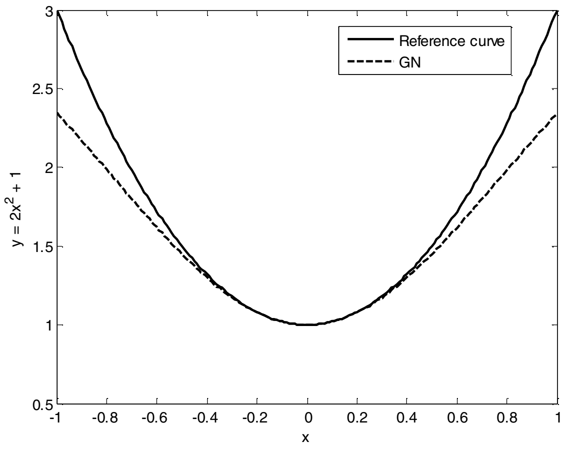

Fig. 4. Output for $y = 2x^2+1$ with GN online trained with PSO with a window sliding in steps of 50.

The following results are obtained with MLP network of size 2 × 5 × 1 with bias 1 online trained with PSO. Fig. 5 shows the MLP output when online trained with a window size of 50 input/output pairs. The window slides in steps of 1. A MSE of 0.067 is obtained. Fig. 6 shows the MLP output when online trained with a window size of 50 input/output pairs but the window slides in steps of 50. A MSE of 1.001 is obtained here. The comparison of the performances of the GN and the MLP both online trained with PSO is shown in Table I.

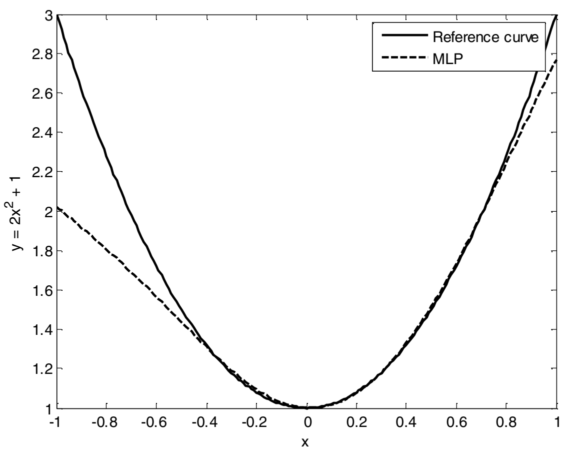

Fig. 5. Output for $y = 2x^2+1$ with MLP online trained with PSO with a window sliding in steps of 1.

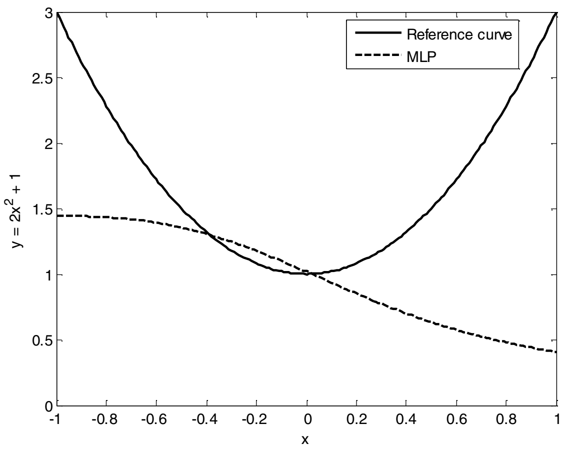

Fig. 6. Output for $y = 2x^2+1$ with MLP online trained with PSO with a window sliding in steps of 50.

TABLE I
COMPARISION OF A MLP NETWORK AND A GN FOR A NONLINEAR STATIC FUNCTION APPROXIMATION

Parameters	MLP		GN	
Number of weights	15		4	
Sliding step size	1	50	1	50
MSE	0.067	1.001	0.006	0.045
*Training time (in sec)	10.359	0.297	0.109	0.016

* performed on the same PC.

Table II shows the variation in the MSE and the training time for y = $2x^2 + 1$, for a window size of 50 and a step size of 1, while Table III shows the variation in the MSE and the training time for a window size of 50 and a step size of 50. The Fig. 7 and Fig. 8. show the variation in the MSE for the GN with the variation in the swarm size. Fig 9. shows the variation in the training time of the GN with the variation of the swarm size.

TABLE II
COMPARISION OF THE MSE AND TRAINING TIME OF THE GN FOR A NONLINEAR STATIC FUNCTION WITH A STEP SIZE OF 1

Particles size	5	10	15	20	25	30
MSE	0.0921	0.0498	0.029	0.024	0.0063	0.0068
*Training time (in sec)	0.031	0.046	0.062	0.094	0.109	0.125

* performed on the same PC.

TABLE III
COMPARISION OF THE MSE AND TRAINING TIME OF THE GN FOR A NONLINEAR STATIC FUNCTION WITH A STEP SIZE OF 50

Particles size	5	10	15	20	25	30
MSE	0.1172	0.1152	0.0591	0.0246	0.0452	0.0497
*Training time (in sec)	0.016	0.016	0.016	0.016	0.016	0.016

* performed on the same PC.

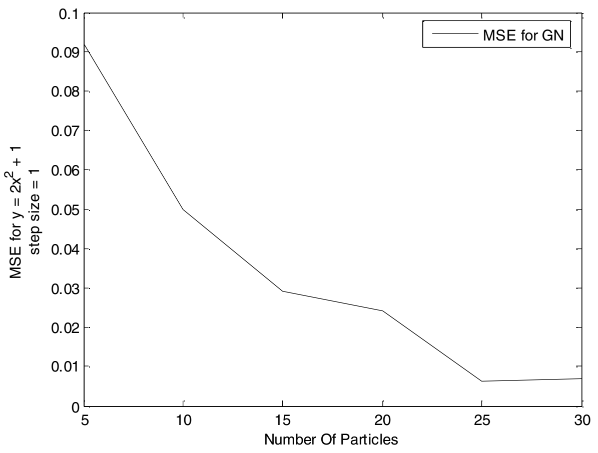

Fig. 7. Number of Particles Vs MSE, for $y = 2x^2+1$ with GN online trained with PSO with a window sliding in steps of 1.

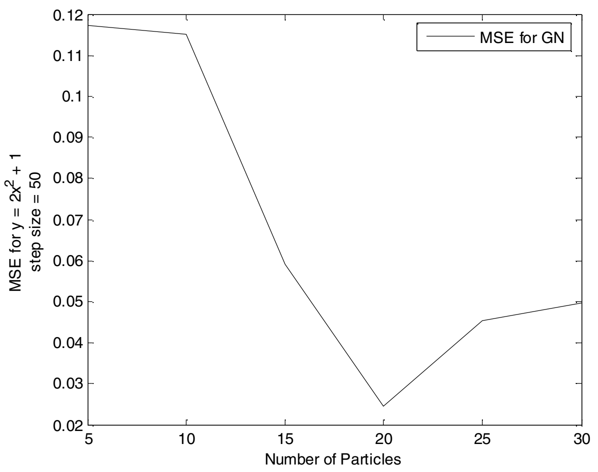

Fig. 8. Number of Particles Vs MSE, for $y = 2x^2+1$ with GN online trained with PSO with a window sliding in steps of 50.

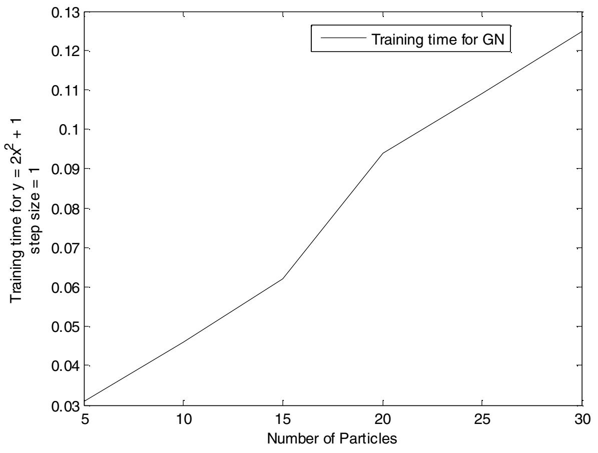

Fig. 9. Number of Particles Vs Training Time, for $y = 2x^2+1$ with GN online trained with PSO with a window sliding in steps of 1.

It is seen from Fig. 7 and Fig. 9 that, as the swarm size tends to increase, the training time needed for the GN also increases and also the MSE decreases. The relationship between the MSE and the training time seems to be linear to the number of particles in the swarm.

B. Time Varying Function

The GN is online trained to approximate a sinusoidal function, $y = 2x^2 + 1$, where $x = sin (2\pi ft)$ and 't' is varied in the range of (-2, 2) with an increment in steps of 0.02. The values of λs and λp are taken to be 1 and 10 respectively. Both the biases are taken as 1. The result in Fig. 10 is obtained from a GN for a window size of 50, sliding in steps of 1. The MSE is found to be 2.559×10^{-4}. Fig. 11 shows the results of a GN with a window of size 50, sliding in steps of 50. The MSE is found to be 0.041.

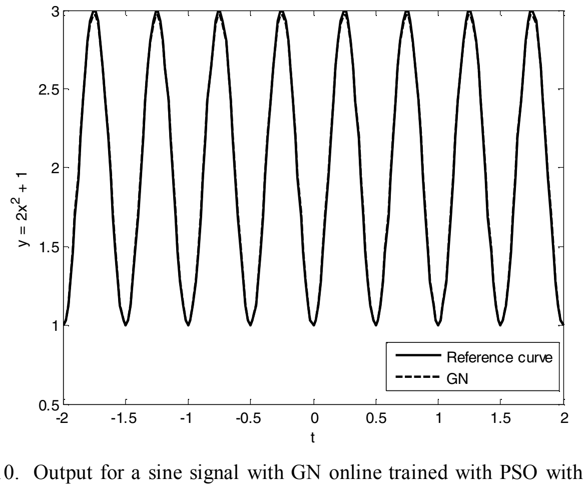

Fig. 10. Output for a sine signal with GN online trained with PSO with a window sliding in steps of 1.

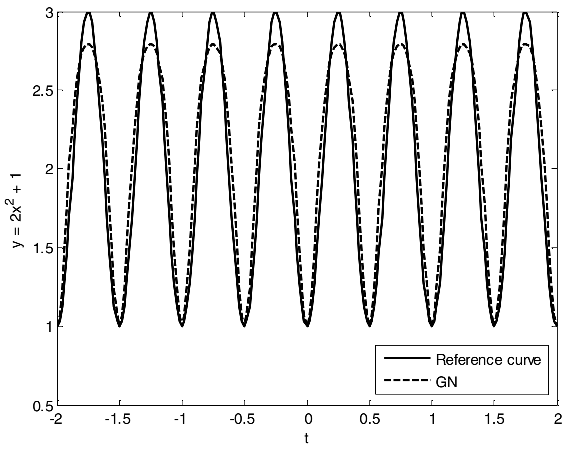

Fig. 11. Output for a sine signal with GN online trained with PSO with a window sliding in steps of 50.

The following results are obtained with the MLP network of size 2 × 5 × 1 with a bias of 1 online trained with PSO. The result in Fig. 12 is obtained from a MLP for a window size of 50, sliding in steps of 1. The MSE is found to be 3.312×10^{-4}. Fig. 13 shows the results of a MLP with a window of size 50, sliding in steps of 50. The MSE is found to be 0.210. The comparison of the performances of the GN and the MLP both online trained with PSO is shown in Table II.

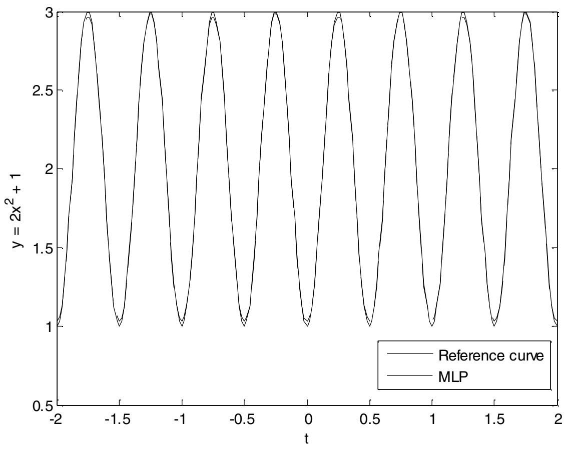

Fig. 12. Output for a sine signal with MLP online trained with PSO with a window sliding in steps of 1.

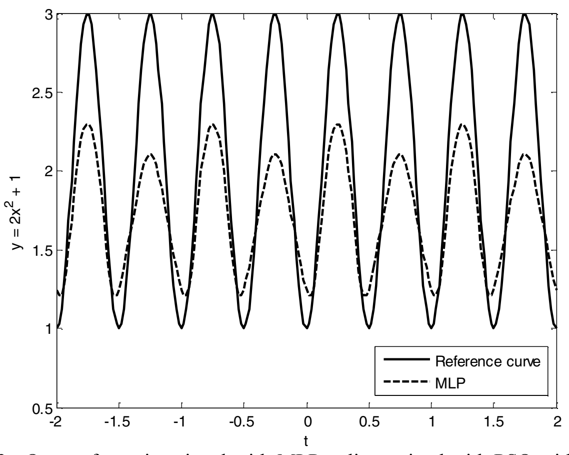

Fig. 13. Output for a sine signal with MLP online trained with PSO with a window sliding in steps of 50.

TABLE IV
COMPARISION OF MLP NETWORK AND GN FOR TIME VARYING FUNCTION APPROXIMATION

Parameters	MLP		GN	
Number of weights	15		4	
Sliding step size	1	50	1	50
MSE	3.312×10^{-4}	0.210	2.559×10^{-4}	0.041
*Training time (in seconds)	11.078	0.328	0.188	0.016

*performed on the same PC.

It can be seen from the Table IV that the MSE in case of the PSO trained GN is lesser than the MSE in case of the PSO trained MLP. It is also observed that the training time of the MLP is much higher than the training time of the GN. When the step size equals the entire length of the data points, it is nothing but offline training. Thus the GN has better performance than the MLP.

Table V shows the variation in the MSE and the training time for $y = 2x^2 + 1$, where $x = \sin(2\pi ft)$ and 't' is varied in the range of (-2, 2), for a window size of 50 and a step size of 1, while Table VI shows the variation in the MSE and the training time for a window size of 50 and a step size of 50. The Fig. 14 and Fig. 15. show the variation in the MSE for the GN with the variation in the swarm size. Fig 16. shows the variation in the training time.

TABLE V
COMPARISION OF THE MSE AND TRAINING TIME OF THE GN FOR A TIME VARYING FUNCTION WITH A STEP SIZE OF 1

Particles size	5	10	15	20	25	30
MSE	0.0034	0.0018	0.0014	6.076×10^{-4}	2.559×10^{-4}	3.736×10^{-4}
*Training time (in sec)	0.047	0.094	0.125	0.156	0.188	0.235

* performed on the same PC.

TABLE VI
COMPARISION OF THE MSE AND TRAINING TIME OF THE GN FOR A TIME VARYING FUNCTION WITH A STEP SIZE OF 50

Particles size	5	10	15	20	25	30
MSE	0.0859	0.0775	0.0637	0.028	0.0347	0.0495
*Training time (in sec)	0.015	0.016	0.016	0.016	0.016	0.016

* performed on the same PC.

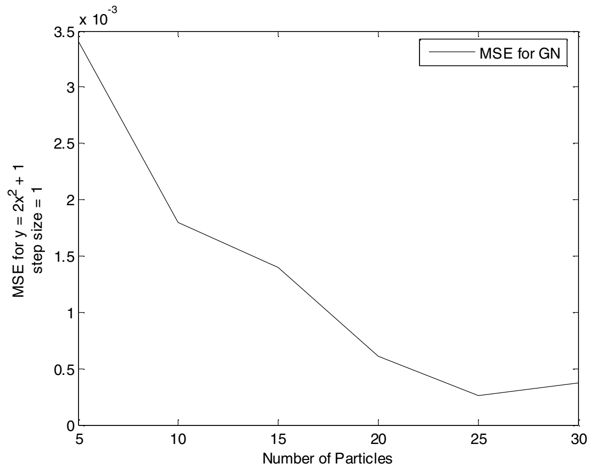

Fig. 14. Number of Particles Vs MSE, for sine signal with GN online trained with PSO with a window sliding in steps of 1.

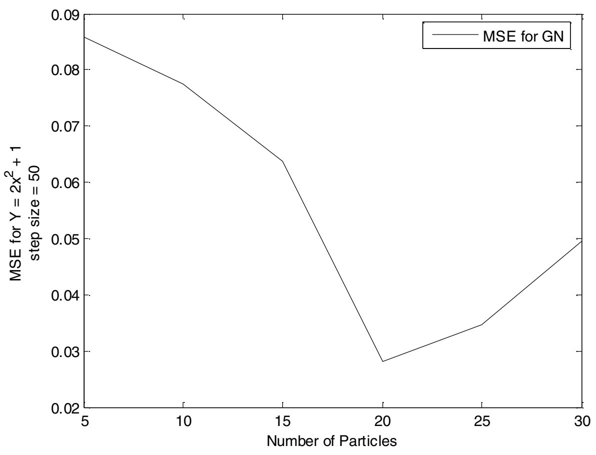

Fig. 15. Number of Particles Vs MSE, for sine signal with GN online trained with PSO with a window sliding in steps of 50.

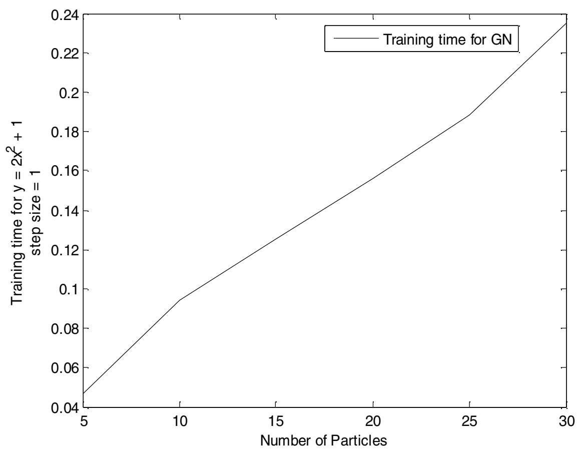

Fig. 16. Number of Particles Vs Training Time, for sine signal with GN online trained with PSO with a window sliding in steps of 1.

It can be seen from Fig. 14 and Fig. 16 that, as the swarm size tends to increase, the training time needed for the GN also increases and the MSE decreases. The relationship between the MSE and the training time is found to be linear to the number of particles in the swarm.

V. Identification of SVC Dynamics in a Power System

A Static VAR Compensator (SVC) is a shunt Flexible AC Transmission Systems (FACTS) employed for regulation of system voltage and for improving the power system stability [5]. The correct identification of SVC dynamics and the power network is necessary for the design of an adaptive controller. Fig. 17 shows a SVC connected at bus 4 in the 12 bus FACTS benchmark system [6].

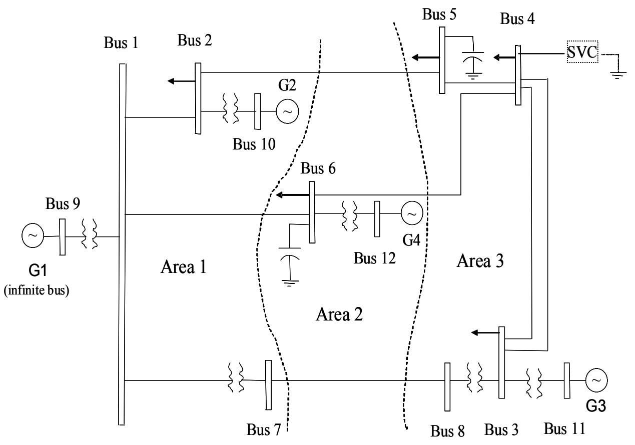

Fig. 17. 12 Bus FACTS benchmark power system with a SVC at bus 4.

The remote signals from the power network and local signals from the SVC are fed to a generalized neuron in order to predict the voltage deviations at bus 4 at time instant k+1. The remote signals are the speed deviations ($\Delta\omega_3$ and $\Delta\omega_4$) of the generators G3 and G4 at time instants k, k-1 and k-2. The advantage of using remote/wide area signals to improve the power system stability is shown in [7]. The local signals are the voltage deviations (ΔV_4) at bus 4 and control signal, susceptance value (ΔB), to the SVC at time instants k, k-1 and k-2. The inputs and output of the GN are shown in Fig. 18.

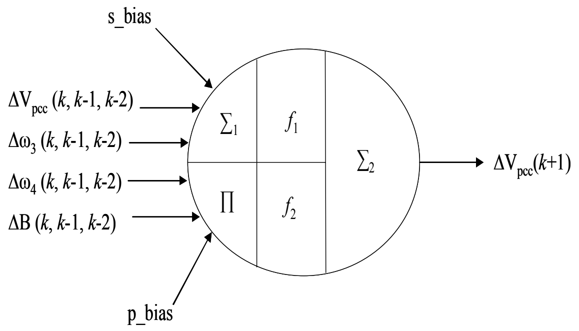

Fig. 18. Inputs and output of the GN.

In order to train the GN, Pseudorandom Binary Signals (PRBS) are applied to the SVC control signal and excitations of generators G3 and G4. These PRBS signals excite the full range of the dynamics of the power network [8]. PRBS signals applied to the generators are of frequencies 5, 3 and 2 Hz and those applied to the SVC are of frequencies 2, 1.5 and 0.5 Hz. The values of the parameters λs and λp in (1) and (3) for this problem are taken to be 0.025 and 10 respectively. Number of particles of PSO to train the GN are 25. The GN is online trained for 10 seconds using a window size of 2.5 seconds, sliding every 5 ms. Since PSO is a population based algorithm, it requires a set of data points in order to update the weights of the neural network. Hence, here a window size of 2.5 seconds which is equivalent to 500 samples of data is used for updating the weights of the GN.

Fig. 19 shows the PRBS signal applied to the SVC, while Fig. 20 shows the estimated and actual bus 4 voltage deviations by the GN and the power network respectively. The MSE for t = 2.5 seconds to 10 seconds is 0.109%.

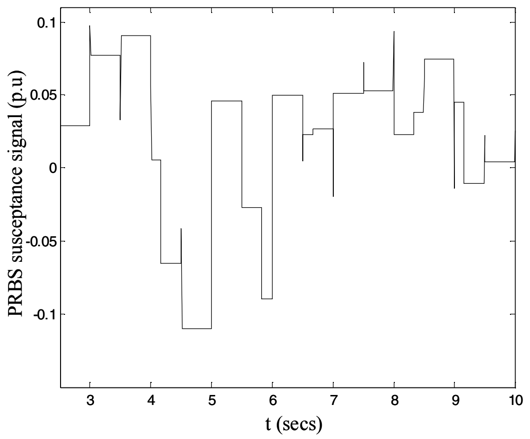

Fig. 19. PRBS signal applied to SVC during training.

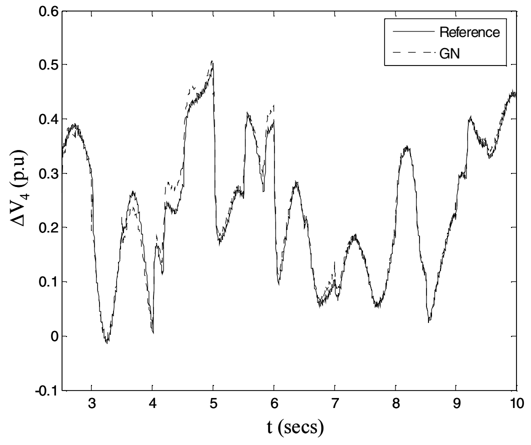

Fig. 20. Estimated and actual bus 4 voltage deviations during training.

After training, the weights of the GN are fixed and its predictions are tested for the next 10 seconds with the PRBS signals is still applied. Figs. 21 and 22 show the PRBS signal applied to the SVC and corresponding responses respectively. The MSE for t = 10 seconds to 20 seconds is 0.3991%.

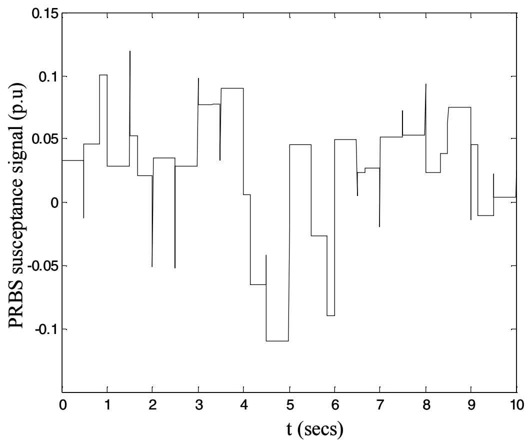

Fig. 21. PRBS signal applied to SVC during testing

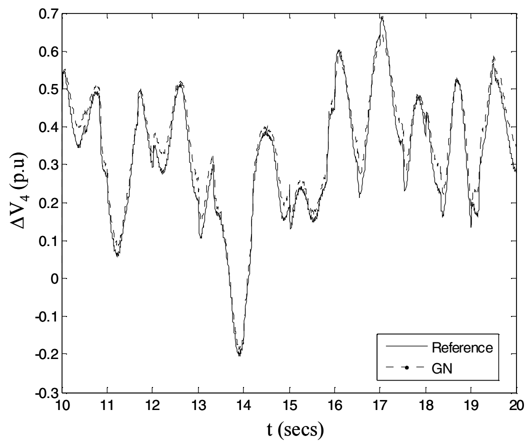

Fig. 22. Estimated and actual bus 4 voltage deviations during testing.

VI. CONCLUSIONS

This paper presents the successful online training of the generalized neuron and MLP with particle swarm optimization. The GN has been shown to approximate non-linear static and time varying functions accurately with fast convergence. Further, the GN online trained with PSO has been demonstrated to identify the nonlinear dynamics of a static VAR compensator in a 12 bus FACTS benchmark power system accurately. The training time taken for the GN to learn the nonlinear functions and the power system dynamics is much lesser and with fewer weights. The GN structure is simple making it attractive for hardware implementations with less computational and memory requirements.

REFERENCES

[1] D. K. Chaturvedi, O. P. Malik, and P. K. Kalra, “Generalized neuron based adaptive power system stabilizer,” IEE Proc.-Generation, Transmission and Distribution, vol. 151, no. 2, pp. 213-218, March 2004.

[2] J. Kennedy and R. Eberhart, “Particle Swarm Optimization,” Proceedings, IEEE conference on neural networks, Perth, Australia, vol. 4, pp. 1942-1948, December 1995.

[3] Y. Shi and R. Eberhart, “A modified particle swarm optimizer,” Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence, pp. 69-73, May 1998.

[4] V. G. Gudise, G. K. Venayagamoorthy, “Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks” Swarm Intelligence Symposium (SIS’03), Proceedings of 2003 IEEE, pp. 110-117, April 2003.

[5] N. G. Hingorani and L. Gyugi, Understanding FACTS, concepts and Technology of Flexible AC Transmission Systems, New Jersey, IEEE press.

[6] S. Jiang, U. D. Annakkage and A. M. Gole, “A Platform for Validation of FACTS Models,” IEEE Transaction on Power Delivery, vol. PP, no. 99, pp. 1 – 8, 2005.

[7] G. K. Venayagamoorthy and S. Ray, “A neural network based optimal wide area control scheme for a power system.” Industry Application Conference, Fortieth IAS Annual Meeting, vol. 1, pp. 700-706, October 2005.

[8] G. K. Venayagamoorthy and R. G. Harley, “A Continually Online Trained Neurocontroller for Excitation and Turbine Control of a Turbogenerator,” IEEE Transaction on Energy Conversion, vol. 16, no. 3, pp. 261-269, June 2001.