Comparison between Traditional Neural Networks and Radial Basis Function Networks

Tiantian Xie, Hao Yu and Bogdan Wilamowski
Electrical and Computer Engineering, Auburn University
tzx0004@auburn.edu, hzy0004@auburn.edu, wilam@ieee.org

***Abstract*- The paper presents the properties of two types of neural networks: traditional neural networks and radial basis function (RBF) networks, both of which are considered as universal approximators. In this paper, the advantages and disadvantages of the two types of neural network architectures are analyzed and compared based on four different examples. The comparison results indicate approaches to be taken relative to the network model selection for practical applications.**

***Keywords*- neural networks, radial basis function networks**

I. INTRODUCTION

As a very important member of computational intelligence, artificial neural networks (ANNs) which include backpropagation (BP) networks [1], radial basis function (RBF) networks [2], counterpropagation networks [3], Kohonen networks [4] and so on, show their strong expertise of data classification, pattern recognition and function approximation. Neural networks are applied for solving various problems in industrial applications, such as nonlinear control [5], image/audio signal processing [6-7], system diagnosis [8] and faults detection [9].

With the principle of "learning to behave", traditional neural networks need to be well trained before applied for applications. Training data can be directly applied as network inputs, and the networks parameters, called "weights", are adjusted iteratively according with the differences between desired network behaviors and actual network behaviors. traditional neural networks have various architectures and the most popular one is multiplayer perceptron (MLP) networks (Fig. 1a). Practically, MLP networks are very inefficient for solving problems. traditional neural networks with connections across layers, such as fully connected cascade (FCC) networks (Fig. 1b) and bridged multilayer perceptron (BMLP) networks (Fig. 1c), are much more powerful, and also require more challenging computations [10]. Fig. 1 shows the minimum traditional neural network architectures for solving parity-7 problem.

RBF networks have fixed three-layer architecture (Fig. 4) which consists of input layer, hidden layer and output layer. The input layer provides network inputs; the hidden layer remaps the input data in order to make them linearly separable; the output layer does linear separation. The special architecture determines that the design of RBF networks is normally organized in three steps: (1) find proper network size; (2) find proper initial parameters (centers and widths); (3) train the networks.

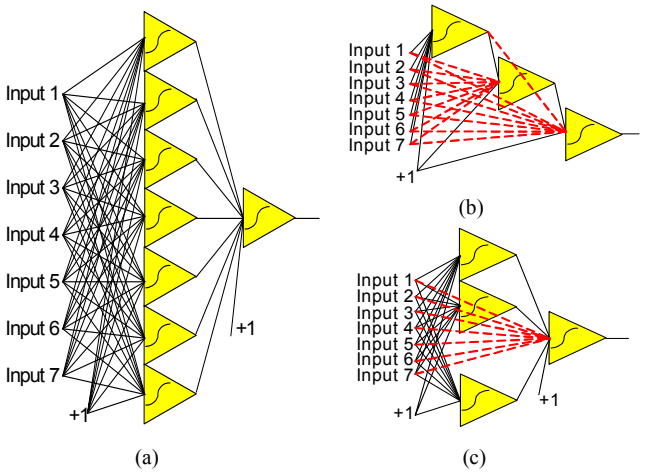


Fig. 1 Minimum BP networks to solve parity-7 problem: (a) standard MLP network; (b) FCC networks; (c) BMLP networks; Red dash lines are connections across layers

Because of the similar layer-by-layer topology, it is often considered that RBF networks belong to MLP networks. It was proven that RBF networks can be implemented by MLP networks with increased input dimensions [11]. Except the similarities of topologies, RBF networks and MLP networks behaves very differently. First of all, RBF networks are simpler than MLP networks which may have more than three layers architectures, so the training process is generally faster than that of MLP networks. Secondly, RBF networks act as local approximation networks, because the network outputs are determined by specified hidden units in certain local receptive fields; while MLP networks work globally, since the network outputs are decides by all the neurons. Thirdly, it is essential to set correct initial states for RBF networks; while MLP networks use randomly generated parameters initially. Last and most importantly, the mechanisms of classification for RBF networks and MLP networks are different: RBF clusters are separated by hyper spheres; while in neural networks, arbitrarily shaped hyper surfaces are used for separation. In the simple two-dimension case as shown in Fig. 2, the RBF network in Fig. 2a separates the four clusters by circles or ellipses (Fig. 2b); while the neural network in Fig. 2c does the separation by lines (Fig. 2d).

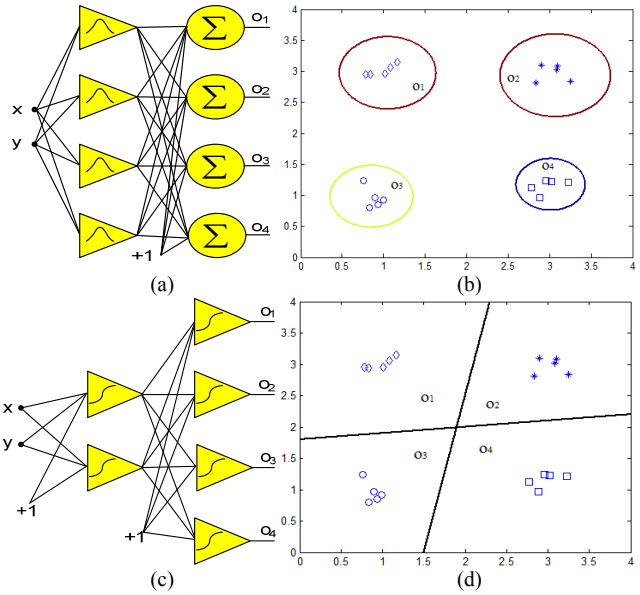


Fig. 2 Different classification mechanisms for pattern classification in two-dimension space: (a) RBF network; (b) Separation result of RBF network; (b) MLP network; (c) Separation result of MLP network

In this paper, traditional neural networks and RBF networks are studied and compared based on four examples. With the comparison results, several clues are provided on network model selection for solving practical problems.

The paper is organized as follows. In the section II, the basic concepts of traditional neural networks are introduced briefly. Section III presents the computational fundamentals of RBF networks. In section IV, traditional neural networks and RBF networks are tested and compared based on four examples. Section V is the conclusion.

II. TRADITIONAL NEURAL NETWORKS

Fig. 3 shows the basic unit of traditional neural networks, with N inputs and M outputs.

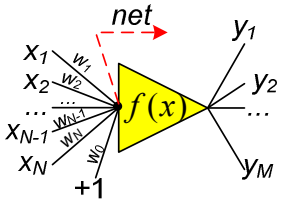


Fig. 3 Single neuron with N inputs and M outputs

Computations related with the single neuron include:

i) Net computation

$$net = \sum_{n=1}^{N} x_n w_n + w_0 \tag{1}$$

Where: n is the index of inputs and weights, from 1 to N; w_n is the weight on input x_n; w_0 is the bias weight.

ii) Output computation

$$y_m = f(net) = \tanh(net) \tag{2}$$

Where: y_m is the output of the neuron; $f(\bullet)$ is the activation function and normally chosen as sigmoidal shape.

For more neurons interconnected together, the two basic computations (1) and (2) for each neuron remain the same; while the only difference is that the inputs of a neuron could be provided by either the outputs of neurons from previous layers or network inputs.

Weight values are the only type of parameters and can be updated by learning algorithms. Based on error backpropagation procedure, various gradient algorithms are developed for traditional neural network learning. First order gradient methods [1] are stable, but very time consuming, and usually fail to converge to very small errors. Training speed and accuracy are significantly improved by applying second order gradient methods, such as Levenberg Marquardt algorithm [12] and neuron-by-neuron algorithm [13]. The recently developed algorithm in [14] inversed the traditional backpropagation procedure and improved the training efficiency for traditional neural networks with multiple outputs.

III. RADIAL BASIS FUNCTION NETWORKS

Fig. 4 shows the general form of RBF networks, with N inputs, L hidden units and M outputs.

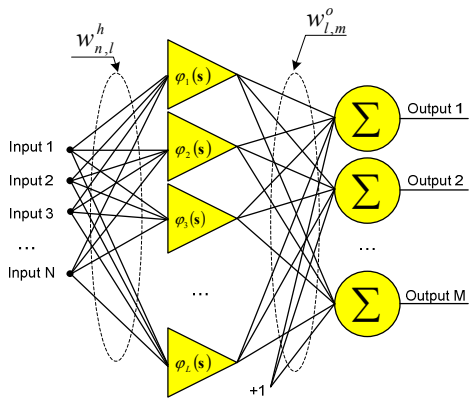


Fig. 4 RBF network with N inputs, L hidden units and M outputs.

The basic computations in the RBF network above include:

i) Input layer computation

At the input of hidden unit l, the input vector $\boldsymbol{x}$ is weighted by input weights $\boldsymbol{w}^h$:

$$\mathbf{s}_l = \left[x_1 w^h_{1,l}, x_2 w^h_{2,l} \cdots x_n w^h_{n,l} \cdots x_N w^h_{N,l}\right] \tag{3}$$

Where: n is the index of input; l is the index of hidden units; x_n is the n-th input; $w^h_{n,l}$ is the input weight between input n and hidden unit l.

ii) Hidden layer computation

The output of hidden unit l is calculated by:

$$\varphi_l(\mathbf{s}_l) = \exp\left(-\frac{\|\mathbf{s}_l - \mathbf{c}_l\|^2}{\sigma_l}\right) \tag{4}$$

Where: the activation function $\varphi_l(\bullet)$ for hidden unit l is normally chosen as Gaussian function; $\boldsymbol{c}_l$ is the center of hidden unit l and σ_l is the width of hidden unit l.

iii) Output layer computation

The network output m is calculated by:

$$o_m = \sum_{l=1}^{L} \varphi_l(\mathbf{s}_l) w^o_{l,m} + w^o_{0,m} \quad (5)$$

Where: m is the index of output; $w^o_{l,m}$ is the output weight between hidden unit l and output unit m; $w^o_{0,m}$ is the bias weight of output unit m.

From the basic computations (3), (4) and (5), one may notice that there are four types of parameters, input weight matrix $\boldsymbol{w}^h$, output weight matrix $\boldsymbol{w}^o$, center matrix $\boldsymbol{c}$ and width vector $\boldsymbol{\sigma}$. Normally, the input weights are all set as "1". The simple linear least squares (LS) method can only adjust the output weights and it performs for nonlinear cases. Iteratively LS method [15] improves the nonlinear performance of output layer. First order gradient methods [16] can adjust output weights, widths and centers during the training process, but they often take long time for convergence and have limited search ability. Kalman filter training algorithm [17] provides similar performance with first order gradient methods, but it improves the training speed significantly. Genetic algorithm [18] performs robust training and does not suffer from local minima problem, but it is very time and computation expensive, especially when parameter dimension is huge. The recently developed improved second order (ISO) method [19] performs efficient and powerful training, and can solve problems with very compact RBF networks.

IV. COMPARISONS

In this section, four problems are applied to test and compare the performance of traditional neural networks and RBF networks, from the points of architecture complexity, generalization ability and noise-tolerant ability. For traditional neural networks, the recently developed neuron-by-neuron (NBN) algorithm [20-21] is applied for training; while for RBF networks, the improved second order (ISO) method [19] is used for parameter updating.

The training/testing results are evaluated by the averaged sum square error calculated by:

$$E = \frac{1}{P}\frac{1}{M}\sum_{p=1}^{P}\sum_{m=1}^{M} e^2_{p,m} \quad (6)$$

Where: p is the index of patterns, from 1 to P, where P is the number of patterns; m is the index of outputs, from 1 to M, where M is the number of outputs. $e_{p,m}$ is the error at output m when training pattern p, calculated as the difference between desired output and associated actual output.

The testing environment consists of: Windows 7 Professional 32-bit operating system; AMD Athlon (tm) ×2 Dual-Core QL-65 2.10GHz processor; 3.00GB (2.75GB usable) RAM; MATLAB 2007b platform.

A. Forward-Kinematics

Forward kinematics [22] is one of practical examples which are well-solved by neural networks. The two-link planar manipulator is modeled to determine the position and orientation of robot's end effectors when the joint angles change (Fig. 5).

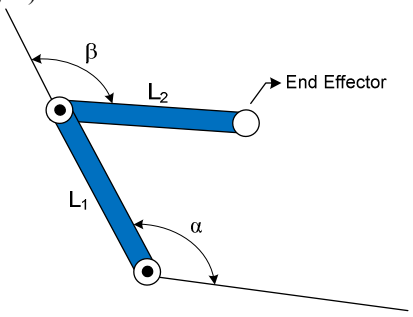


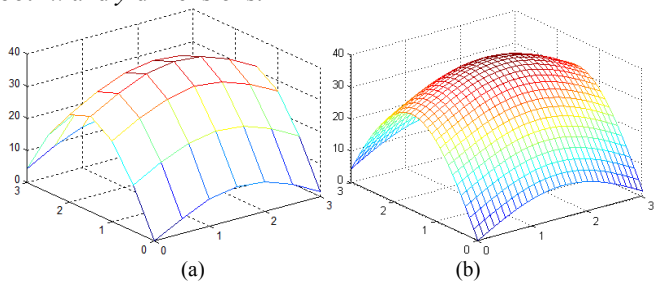
Fig. 5 Two-link planar manipulator

As shown in Fig. 5, in the two-dimension space, the end effector coordinates of the manipulator is calculated by

$$x = L_1 \cos\alpha + L_2 \cos(\alpha + \beta) \quad (7)$$

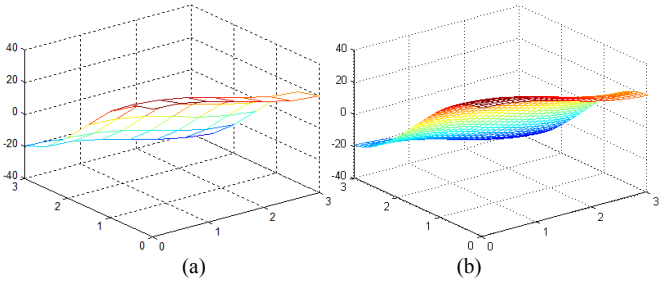
$$y = L_1 \sin\alpha + L_2 \sin(\alpha + \beta) \quad (8)$$

The data set of the two-dimensional forward-kinematics consist of 49 training patterns and 961 testing patterns which are generated from equations (7) and (8), with parameters α and β uniformly distributed in range [0, 3], and L_1=30, L_2=10. Figs. 6 and 7 below visualizes the training/testing points in both x and y dimensions.



(a) (b)

Fig. 6 Data set in x-dimension: (a) 7×7=49 training patterns; (b) 31×31= 961 testing patterns



(a) (b)

Fig. 7 Data set in y-dimension: (a) 7×7=49 training patterns; (b) 31×31= 961 testing patterns

For traditional neural networks, all neurons are connected in FCC architectures, with randomly generated initial weights between [-1, 1]. For RBF network, randomly selected patterns are used as initial centers, and the weights and widths are randomly generated between (0, 1]. For each architecture, the testing is repeated for 100 times and the averaged trajectories of training/testing errors are presented in Figs. 8 and 9 below.

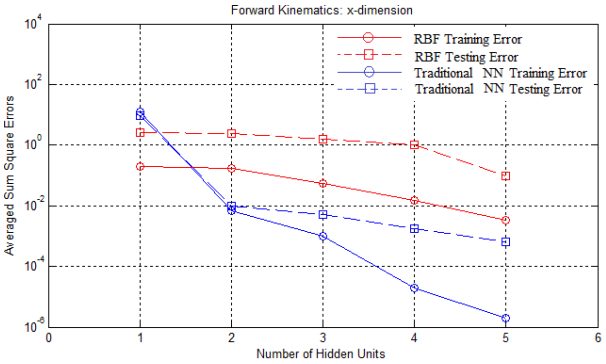


Fig. 8 X-dimension of forward kinematics: training/testing errors vs. the number of hidden units

As shown in Fig. 6, in x-dimension of kinematics, there is a big but not regular peak. Comparison results in Fig 8 show that RBF networks obtained smaller training/testing error than traditional neural networks at first. While as the number of hidden units increase, traditional neural networks perform much better.

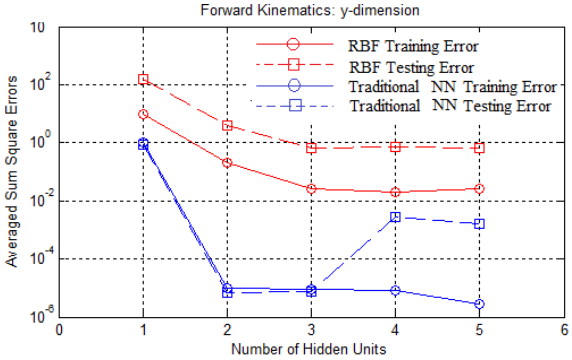


Fig. 9 Y-dimension of forward kinematics: training/testing errors vs. the number of hidden units

In y-dimension kinematics, there are no regular peaks and valleys (Fig. 7). As the comparison results shown in Fig. 9, traditional neural networks perform much better than RBF networks. As the number of hidden units increase, all errors decrease at first; however, the testing errors of trained traditional neural networks increase due to the over-fitting problem [23].

B. *Peaks Function Approximation*

In the peaks function approximation problem, 20×20=400 points (Fig. 10a) are applied as training set, in order to predict the values of 100×100=10,000 points (Fig. 10b) in the same range. The surface is generated by MATLAB function *peaks* and all training/testing points are uniformly distributed.

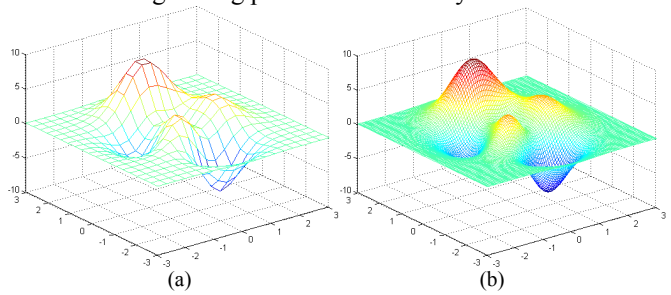


Fig. 10 Peaks function approximation problem: (a) training data, 20×20=400 points; (b) testing data, 100×100=10,000 points.

Using RBF networks for approximating the peaks surface, since there are three peaks and two valleys, at least 5 hidden units are required.

For traditional neural networks, FCC architectures are applied for training. Table I presents the experimental results. One may notice that, with the same 5 hidden units, feedforward neural network got more than 2 times larger training errors and more than two orders of magnitude larger testing errors than radial basis function network.

TABLE I COMPARISON RESULTS OF RADIAL RBF NETWORKS AND TRADITIONAL NEURAL NETWORKS ON PEAKS SURFACE APPROXIMATION PROBLEM

Architectures	***Training Errors***	***Testing Errors***
RBF with 5 hidden units	0.0111	0.0120
FCC with 4 hidden units	0.1361	1.3706
FCC with 5 hidden units	0.0294	1.1834
FCC with 6 hidden units	0.0040	1.1689
FCC with 7 hidden units	0.0018	1.1713
FCC with 8 hidden units	0.0010	1.1728

Fig. 11 below shows the generalization results of two types of neural networks, both of which have 5 hidden units.

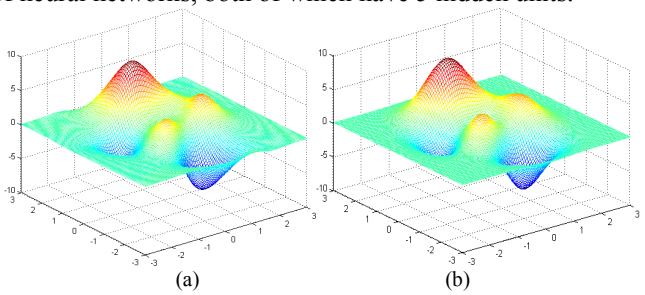


Fig. 11 Generalization results of neural networks with 5 hidden units: (a) traditional neural networks, testing error=1.1834; (b) RBF networks, testing error=0.0120

C. *Two-Spiral Problem*

Two-spiral problem [24] is considered as a very complex benchmark to evaluate the power and efficiency of training algorithms and network architectures. As shown in Fig. 12, the purpose of the two-spiral problem is to separate the 94 twisted two-dimension points into two groups, marked as +1 (blue circles) and -1 (red stars).

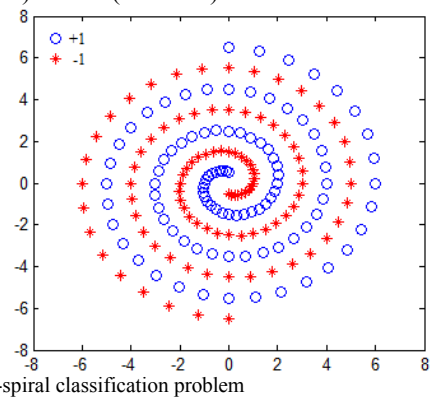


Fig. 12 Two-spiral classification problem

Using traditional neural networks, the two-spiral problem can be solved very efficiently and the minimum number of required hidden units depends on network architectures. For example, using standard MLP architecture with one hidden layer, at least 33 hidden units are required for successful training. For MLP architecture with two hidden layers (assume they have the same number of neurons), at least 14 hidden units are required for convergence [14]. The most efficient architecture, FCC networks, can solve two-spiral problem with only 7 hidden units [14]. Fig. 13 shows the generalization results of 13 hidden units in FCC networks.

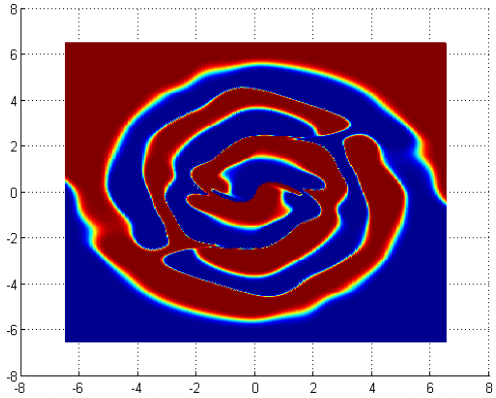


Fig. 13 Generalization result of FCC architecture with 13 hidden units

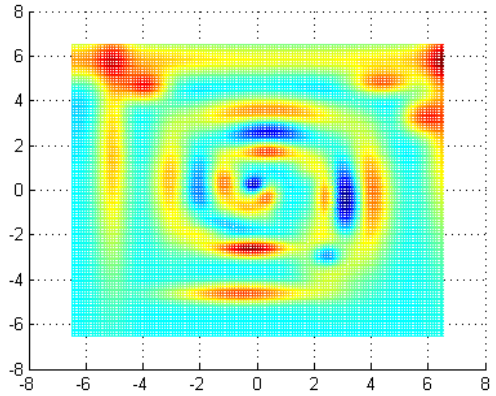


Fig. 14 Generalization result of RBF network with 40 hidden units

Using RBF networks, in order to reach the similar training error with the FCC architecture with 7 hidden units, at least 40 hidden units are required. The generalization result is shown in Fig. 14.

For the two-spiral classification problem, one may notice that, even using much less number of hidden units, traditional neural networks (Fig. 13) can get better classification results than RBF networks (Fig. 14).

D. Character Image Recognition

As shown in Fig. 15 below, for each column, there are 10 character images from "A" to "J", each of which consists of 8×7=56 pixels with normalized Jet degree between -1 and 1 (-1 for blue and 1 for red). The first column is the original image data without noise and used as training patterns; while the rest 7 columns, from the 2nd column to the 8th column, are noised and used as testing patterns. The strength of noise is calculated by:

$$NP_i = P_0 + i \times \delta \tag{9}$$

Where: P_0 is the original image in 1st column; NP_i is the image data with i-th level noise; i is the noise level from 1 to 7; δ is the randomly generated noise between [-0.5, 0.5].

The aim is to build neural networks based on the training patterns (1st column) and then test the networks with noised input data (from 2nd column to 8th column). For each noise level, the testing will be repeated for 100 times with randomly generated noise.

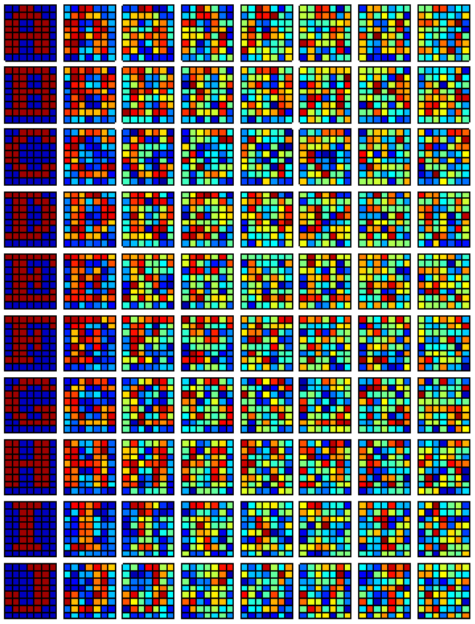


Fig. 15 Character images with different noise levels from 0 to 7 in left-to-right order (one data set in 100 groups)

Using traditional neural networks, the MLP architecture 56-10 is applied for training. The testing results on the trained network are presented in Table II below. One may notice that recognition errors appear when patterns with 2nd level noises are applied.

TABLE II SUCCESS RATES OF THE TRAINED TRADITIONAL NEURAL NETWORK FOR CHARACTER IMAGE RECOGNITION

Data / Char	*Noise level 1*	*Noise level 2*	*Noise level 3*	*Noise level 4*	*Noise level 5*	*Noise level 6*	*Noise level 7*
"A"	100%	99%	90%	70%	59%	44%	39%
"B"	100%	100%	100%	95%	94%	84%	81%
"C"	100%	98%	81%	52%	51%	45%	41%
"D"	100%	97%	84%	64%	56%	34%	33%
"E"	100%	100%	92%	70%	52%	48%	42%
"F"	100%	95%	83%	71%	49%	36%	35%
"G"	100%	96%	72%	58%	53%	34%	32%
"H"	100%	94%	60%	50%	32%	32%	23%
"I"	100%	100%	100%	95%	83%	76%	71%
"J"	100%	100%	96%	81%	70%	54%	46%

For RBF networks, 10 hidden units are chosen and their centers are corresponding to 10 characters, respectively. Applying the testing patterns, the performance of the trained RBF network is shown in Table III below. One may notice that recognition errors appear until 3rd level noised patterns applied.

TABLE III SUCCESS RATE OF THE TRAINED RBF NETWORK FOR CHARACTER IMAGE RECOGNITION

Data / Char	*Noise level 1*	*Noise level 2*	*Noise level 3*	*Noise level 4*	*Noise level 5*	*Noise level 6*	*Noise level 7*
"A"	100%	100%	100%	100%	100%	97%	97%
"B"	100%	100%	100%	99%	97%	96%	87%
"C"	100%	100%	99%	98%	90%	88%	80%
"D"	100%	100%	100%	98%	98%	95%	88%
"E"	100%	100%	100%	95%	94%	76%	76%
"F"	100%	100%	100%	97%	92%	83%	79%
"G"	100%	100%	99%	96%	88%	81%	77%
"H"	100%	100%	100%	100%	100%	98%	91%
"I"	100%	100%	100%	100%	100%	100%	97%
"J"	100%	100%	100%	100%	100%	99%	95%

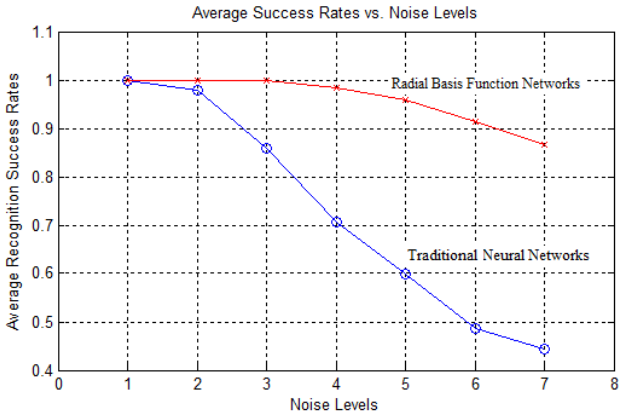


Fig. 16 Average recognition success rates of traditional neural networks and RBF networks under different levels of noised inputs

Fig. 16 shows the average success rate of two types of neural network architectures. It can be seen that, RBF networks (red line) is more robust, and have better tolerant ability to input noises than traditional neural networks (blue line).

V. CONCLUSION

The paper presents the comparison of traditional neural networks and RBF networks based on four examples. According with the comparing results and properties of two types of neural networks, the conclusions below can be made to provide suggestions for network model selection:

- For function approximation problems, RBF networks are specially recommended for surface with regular peaks and valleys, since efficient and accurate design can be obtained. While, for surfaces without regular peaks and valleys, traditional neural networks are preferred as a general model.
- For classification problems, traditional neural networks can get better classification results with much more efficient networks than RBF networks.
- For trained networks, RBF networks perform more robustly and tolerantly than traditional neural networks, when dealing with noised input data set.

REFERENCES

[1] D. E. Rumelhart, G. E Hinton and R. J. Williams, "Learning Representations by Back-Propagating Errors," *Nature*, vol. 323, pp. 533-536, 1986.

[2] J. Moody and C. J. Darken, "Fast Learning in Networks of Locally-Tuned Processing Units," *Neural Computation*, vol. 1, no. 2, pp. 281-294, 1989.

[3] R. Hecht-Nielsen, "Counterpropagation Networks," *Appl. Opt.* 26(23):4979-4984, 1987.

[4] A. G. Ivakhnenko and J. A. Mueller, "Self-Organizing of Nets of Active Neurons," *System Analysis Modeling Simulation*, vol. 20, pp. 93-106, 1995.

[5] K. Derr and M. Manic, "Wireless based object tracking based on neural networks", ICIEA 2008, *3rd IEEE Conference on Industrial Electronics and Applications*, Singapore, June 3-5, pp.308-313, 2008.

[6] Y. J. Lee, J. Yoon, "Nonlinear Image Upsampling Method Based on Radial Basis Function Interpolation," *IEEE Trans. on Image Processing*, vol. 19, issue 10, pp. 2682-2692, 2010.

[7] F. Moreno, J. Alarcón, *et al.*, "Reconfigurable Hardware Architecture of a Shape Recognition System Based on Specialized Tiny Neural Networks With Online Training," *IEEE Trans. on Industrial Electronics*, vol. 56, no. 8, pp. 3253-3263, 2009.

[8] O. Linda and M. Manic, "Online Spatio-Temporal Risk Assessment for Intelligent Transportation Systems." *IEEE Trans. On Intelligent Transportation Systems*, vol. 12, no. 1, pp. 194-200, 2011.

[9] S. Huang and K. K. Tan, "Fault Detection and Diagnosis Based on Modeling and Estimation Methods," *IEEE Trans. on Neural Networks*, vol. 20, issue 5, pp. 872-881, Apr. 2009.

[10] B. M. Wilamowski, N. J. Cotton, O. Kaynak, G. Dundar, "Computing Gradient Vector and Jacobian Matrix in Arbitrarily Connected Neural Networks," *IEEE Trans. on Industrial Electronics*, vol. 55, no. 10, pp. 3784-3790, Oct. 2008.

[11] B. M. Wilamowski, R. C. Jaeger, "Implementation of RBF Type Networks by MLP Networks," *IEEE International Conference on Neural Networks*, Washington DC, June 3-6, 1996, pp. 1670-1675.

[12] M. T. Hagan, M. B. Menhaj, "Training Feedforward Networks with the Marquardt Algorithm," *IEEE Trans. on Neural Networks*, vol. 5, no. 6, pp. 989-993, Nov. 1994.

[13] B. M. Wilamowski and H. Yu, "Improved Computation for Levenberg Marquardt Training," *IEEE Trans. on Neural Networks*, vol. 21, no. 6, pp. 930-937, June 2010.

[14] B. M. Wilamowski and H. Yu, "Neural Network Learning Without Backpropagation," *IEEE Trans. on Neural Networks*, vol. 21, no.11, pp. 1793-1803, Nov. 2010.

[15] B. M. Wilamowski, "Modified EBP Algorithm with Instant Training of the Hidden Layer," *Proceedings of Industrial Electronic Conference (IECON'97)*, New Orleans, November 9-14, 1997, pp. 1097-1101.

[16] N. B. Karayiannis, "Reformulated Radial Basis Neural Networks Trained by Gradient Descent," *IEEE Trans. Neural Networks*, vol. 10, issue 3, pp. 657-671, Aug. 2002.

[17] D. Simon, "Training Radial Basis Neural Networks with the Extended Kalman Filter," *Neurocomputing*, vol. 48, pp. 455-475, 2002.

[18] B. A. Whitehead and T. D. Choate, "Cooperative-Competitive Genetic Evolution of Radial Basis Function Centers and Widths for Time Series Prediction," *IEEE Trans. on Neural Networks*, vol. 7, no. 4, pp. 869-880, July, 1996.

[19] H. Yu, T. T. Xie, B. M. Wilamowski and J. Hewlett, "Fast and Efficient Second Order Method for Training Radial Basis Function Networks," *IEEE Trans. on Neural Networks*. (submitted)

[20] H. Yu and B. M. Wilamowski, "Fast and efficient and training of neural networks," in *Proc. 3nd IEEE Human System Interaction Conf. HSI 2010*, Rzeszow, Poland, May 13-15, 2010, pp. 175-181.

[21] H. Yu and B. M. Wilamowski, "Efficient and Reliable Training of Neural Networks," in *Proc. 2nd IEEE Human System Interaction Conf. HSI 2009*, Catania, Italy, May 21-23, 2009, pp. 109-115.

[22] A. Malinowski and H. Yu, "Comparison of Embedded System Design for Industrial Applications," *IEEE Trans. on Industrial Informactics*. (accepted for publication)

[23] B. M. Wilamowski, "Neural Network Architectures and Learning Algorithms: *How Not to Be Frustrated with Neural Networks*," *IEEE Industrial Electronics Magazine,* vol. 3, no. 4, pp. 56-63, Dec. 2009.

[24] H. Yu and B. M. Wilamowski, "C++ Implementation of Neural Networks Trainer," *13-th International Conference on Intelligent Engineering Systems*, *INES09*, Barbados, April 16-18, 2009.