Journal Homepage: -www.journalijar.com
INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)
ISSN NO. 2320-5407
Article DOI:10.21474/IJAR01/5597
DOI URL: http://dx.doi.org/10.21474/IJAR01/5597

RESEARCH ARTICLE

COMPARATIVE ANALYSIS OF RBF (RADIAL BASIS FUNCTION) NETWORK AND GAUSSIAN FUNCTION IN MULTI-LAYER FEED-FORWARD NEURAL NETWORK (MLFFNN) FOR THE CASE OF FACE RECOGNITION.

Arvind Kumar.
Qtr No-512, Sector-4, CPWD qtrs, Near Balak Rram Hospital, Timarpur, Delhi India-110054.

Manuscript Info

Manuscript History

Received: 11 August 2017
Final Accepted: 13 September 2017
Published: October 2017

Key words:-
Neural Network, Face Recognition, RBF-Radial Basis Function, RBFNN-Radial Basis Function Neural Network, MLFNN--Multi-layer Feed-forward Neural Network. Caltech 101- Face database of California Institute of Technology.

Abstract

We know that there are generally two ways in MATLAB software by which we apply radial basis function.
1. Directly use radbas (radial Basis transfer) function in hidden layer of MLFFNN (Multi-layer Feedforward Neural Network).
2. We make Radial Basis network with the help of newrb or newrbe function.

If we apply both the function in the case of face recognition then I see that both conditions are not show equal performance, second condition is fast with respect to first condition. Because second condition is used for local approximation and first condition is used for universal approximations.
For recognition of face I take Caltech 101 database.
Caltech 101 database is created in California Institute of Technology in September 2003. This database contains color-full digital images. This database is compiled by Fei-Fei Li, Marco Andreetto, Marc 'Aurelio Ranzato and PietroPerona. This is used for facilitate computer vision research and technique. This is mainly used for image recognition, classification and categorization. Caltech 101 contains 9146 images. All the images are spilt into 101 distinct object categories such as faces, watches, ants etc.

Introduction:-

There are mainly two learning technologies have been used in recognition system:-

1. Supervised learning method.
2. Un-supervised learning method.

In supervised learning method, we have given target and I achieve that target. In this process first of all I found out (output of the network) and then I calculate its errors, by the following manner:-
Algorithm:-

```
Error=given_target - output
If (Error=0)
    Network becomes steady
Else
```

Corresponding Author:-Arvind Kumar.
Address:-Qtr No-512, Sector-4, CPWD qtrs, Near Balak Ram Hospital, Timarpur, Delhi India-110054.

```
        Again Calculate error
    End
    (do until error==0)
```

Example:- Multi-Layer Feedforward Neural Network, RBF Network, ADALINE etc.

In Un-supervised learning method, we have **not** given target. So we make a group that is called Cluster, so this method is also called Clustering Method.

Example:- SOM (Self Organization Map) Network.

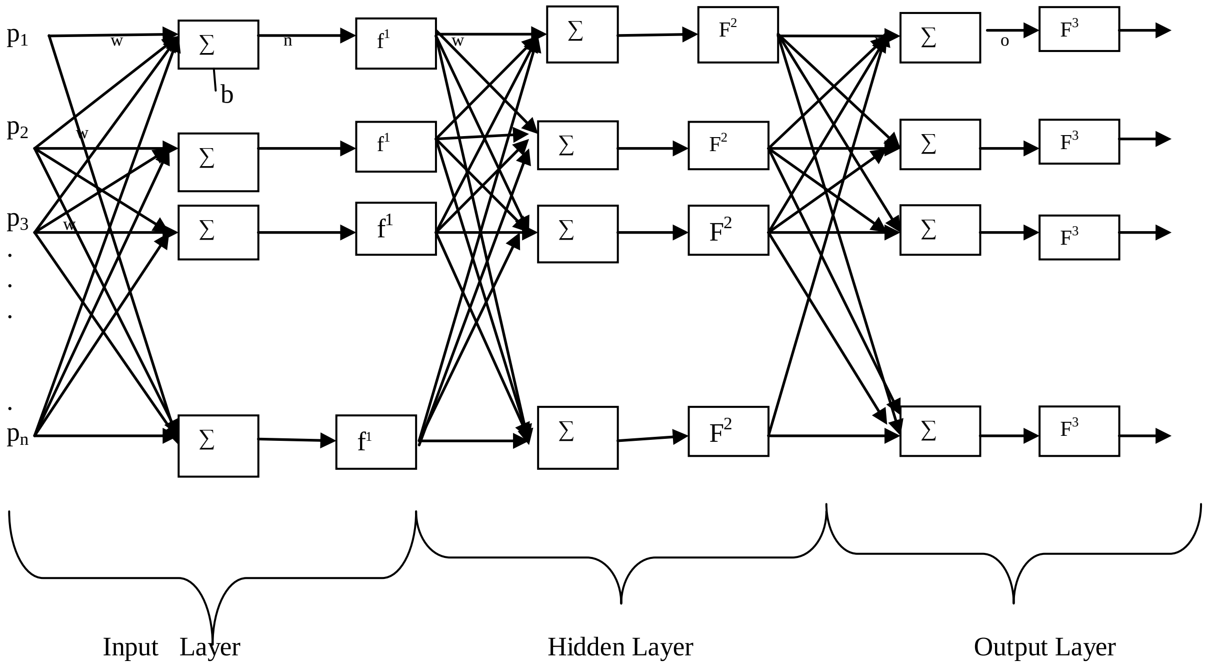


Fig. 1:- Multi-layer feedforward Neural Network

Introduction of Multilayer Feed-Forward Neural Network (MLFNN):-

This network comes under supervised learning process.

Related work:-

First of all Single-layer neural network was given by Rosenblatt. But this network is limited to classification of linearly patterns only.

Then Widrow and Hoff's had given LMS algorithm. But this algorithm is based on a Single Linear Neuron with adjustable weights, which limits the computing power of the algorithm. To remove the limitations of the perceptron and the LMS algorithm another network comes that is called Multilayer Perceptron, which is given by Minsky and Papert (1996).

Multiple layers of Neurons [21]:-

This network has several layers. Each layer has its own weight matrix **w**, its own bias vector **b**, a net input vector **n** and an output vector **a**.

There are mainly three layers of neural are very famous, which is shown in figure 1.

In figure 1:-

First Layer-	Input Layers
Second Layer-	Hidden Layer
Third Layer-	Output Layer

Of course multi-layer networks are more powerful than Single-layer networks.

Training in multilayer perceptron is known as the back-propagation algorithm, which also includes special case of LMS(Provided by Widrow and Hoff).

In this network there are two phases of training [22]:-

1. **Forward phase:-** In this phase layer-by-layer, the synaptic weights of the network are calculated and the inputs signal is propagated through the network until we reaches the outputs.
2. **Backward phase**:- We calculate

error=(target)-(induced output)

This error is again propagated through the network [layer by layer] by the backward direction. Again we calculate the synaptic weight and output in straightforward manner.

Function of the Hidden Layer:-
Hidden neurons work as like a feature detectors. So, by applying the hidden neurons, we find out features of the system.

Method of perform in Multi_layer perceptron:-
Multi-layer perceptron is actually performed in two different methods:-

1. **Batch learning:-** Epoch-by-epoch basis manner, we adjust the synaptic weights of multi-layer perceptron.
2. **On-line learning:-** This is also called example-by-example basis method.

Suppose example are in the order {x(1), d(1)}, {x(2), d(2)},…… ……… {x(n), d(n)}.
The first examples pair {x(1), d(1)}. Now weight adjusts using method of gradient descent. Then I take another example {x(2), d(2)} and again adjust the weight and bias of the network……..This procedure is continue to the last method.

So, by this procedure, this method is also used for solving pattern-classification problems.

The Back-propagation Algorithm [22]

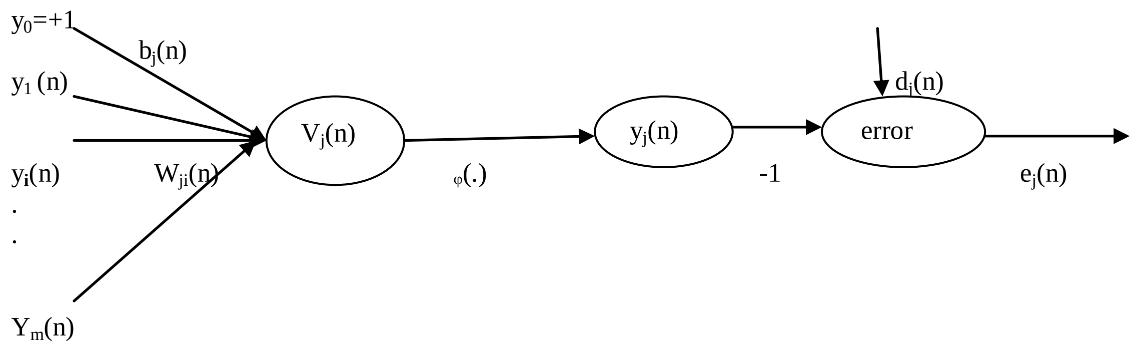


Figure 2:-Signal-Flow graph highlighting the details of output neuron j.

In the figure 2,

$v_j(n)$=induced local field (Summation of multiplication of inputs and weights are called induced local field.)

i.e.

$v_j(n)=\sum_{i=0}^{m} wy$……………………………..(i)

where, $w=w_{ji}(n)$= Weight or the first layer
and $y=y_i(n)$=input of the first layer
m=total number of inputs (including bias).

If activation function is $\varphi(v_J(n))$, then

$y_j(n)=\varphi_j(v_J(n))$……………………….……(ii)

Suppose, Instantaneous error energy of the neuron j is defined by:-

$\varepsilon_j=\frac{1}{2}(e_j^2(n))$……………………………..…….(iii)

Where $e_j=d_j(n)-y_j(n)$……………………….………...(iv)

So, by chain rule,

$\frac{d(\varepsilon j(n))}{d(wji(n))}=\frac{\partial\varepsilon j(n)}{\partial ej(n)}$ x $\frac{\partial ej(n)}{\partial yj(n)}$x $\frac{\partial yj(n)}{\partial vj(n)}$x $\frac{\partial vj(n)}{\partial wji(n)}$…..(v)

Now, for finding $\frac{\partial\varepsilon j(n)}{\partial ej(n)}$, we use equation (iii),

$\frac{\partial\varepsilon j(n)}{\partial ej(n)}=\frac{1}{2}.2.\ e_j(n)=e_j(n)$…………………….(vi)

Now, for finding $\frac{\partial ej(n)}{\partial yj(n)}$, we use equation (iv),

$\frac{\partial ej(n)}{\partial yj(n)}=-1$…………………….………….(vi)

Now, for finding $\frac{\partial yj(n)}{\partial vj(n)}$, we use equation (ii),

$\frac{\partial yj(n)}{\partial vj(n)}=\varphi'_j(v_J(n))$………………………..……(vii)

Now, for finding $\frac{\partial vj(n)}{\partial wji(n)}$, we use equation (i),

$\frac{\partial vj(n)}{\partial wji(n)}=y_i(n)$………………..…………….(viii)

So, putting all of above value in equation (v)

$\frac{d(\epsilon j(n))}{d(w(n))}=-e_j(n).\ \varphi'_j(v_J(n)).\ y_i(n)$….……….(ix)

By, the rule of delta

$\Delta w_{ji}(n)=-\eta.\frac{d(\epsilon j(n))}{d(wji(n))}$……………………….(x)

where, η=learning rate parameter.

When we use gradient in weight space, then

$\Delta w_{ji}(n)=-\eta.\ \delta_j(n).y_i(n)$…………………….(xi)

Where $\delta_j(n)$=local gradient and

$\delta_j(n)=\frac{d(\epsilon j(n))}{d(vj(n))}=e_j(n).\ \varphi'_j(v_J(n))$…………(xii)

Here two cases are arise:-

Case(i):- Neuron j is an output node:-

Here, we do straight forward to compute the local gradient $\delta_j(n)$.

Case(ii):- Neuron j is a Hidden node:-

In this case, we redefine the local gradient $\delta_j(n)$ for hidden neuron j as

$\delta_j(n)=\frac{d(\epsilon j(n))}{d(vj(n))}.\frac{\partial yj(n)}{\partial vj(n)}=\frac{d(\epsilon j(n))}{d(vj(n))}.\varphi'j(vJ(n)).$………(xiii)

Neuron j is hidden.

Now take another layer, say k-layer.

So,

$\varepsilon(n)=\frac{1}{2}(e_k^2(n))$……………………..……(xiv)

By the above method, we calculate

$\frac{d(\epsilon(n))}{d(yj(n))}=\Sigma e_k\frac{\partial ej(n)}{\partial yj(n)}$…………………….….(xv)

In summarize manner, we say that in the back-propagation algorithm, we do:-

weight correction $(\Delta w_{ji}(n))=\eta.\ \delta_j(n).y_i(n)$………..(xvi)

Here η=learning rate parameter

$\delta_j(n)$=local gradient

$y_i(n)$=input signal of neuron j.

$w_{new}=w_{old}+\Delta w_{ji}(n)$
$b_{new}=b_{old}+\eta$ (error)

Activation Function:-
There are following activation functions, we use in the neural network:- (let a=output and n=neuron)

(a) Hard limit (hardlim) $a=0;$ if $n<0;$
$a=1;$ if $n>=0$

(b) Symmetrical hard limit (hardlims)
$a=-1$ if $n<0;$
$a=+1$ if $n>=0;$

(c) Linear (Purelin)
$a=n;$

(d) Saturating Linear (Satlin)
$a=0$ if $n<0;$
$a=n$ if $0<=n<=1;$
$a=+1$ if $n>1;$

(e) Symmetric Saturating Linear (Satlins)
$a=-1$ if $n<-1;$
$a=n$ if $-1<=n<=1;$
$a=+1$ if $n>1;$

(f) log sigmoid (logsig)
$a=1/(1+e^{-n})$

(g) Hyperbolic Tangent Sigmoid (tansig)
$a=(e^{n}-e^{-n})/(e^{n}+e^{-n})$

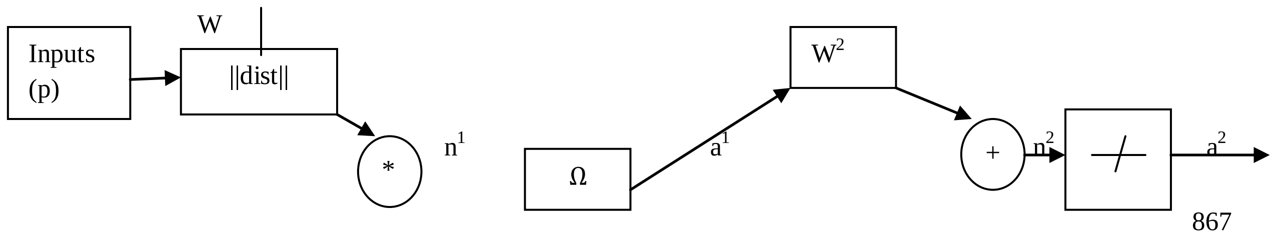
(h) Positive Linear (poslin)
$a=0$ if $n<0;$
$a=n$ if $n>=0;$

(i) Competitive (compet)
$a=1;$ neuron with max n
$a=0;$ all other neurons

(A) Introduction of RBF(Radial Basis Function) Network [21]:-
RBF was performed by Powell and others during the 1980.
It is used for exact interpolation in a multi dimensional space.

Broomhead&Lowe(Brlo88) were the first to develop the RBF network neural model.
Figure 3 is a RBF Network. This Radial Basis Function Network (RBF) network is a two-layer network.

First Layer: - Here network do following operations:-



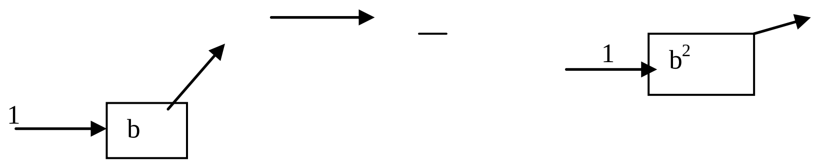


Figure 3:-Radial Basis Function Neural Network

(a) First of all find out distance between the input vector and the rows of the weight matrix.

(b) After then the result is multiple by the bias of the first layer.

i.e. $n_i^1 = ||P-W||.b_i^1$………..(xvii)

(c) Then apply Gaussian function:-

$a = e^{-n^2}$…………..………..(xviii)

and it is plotted as:-

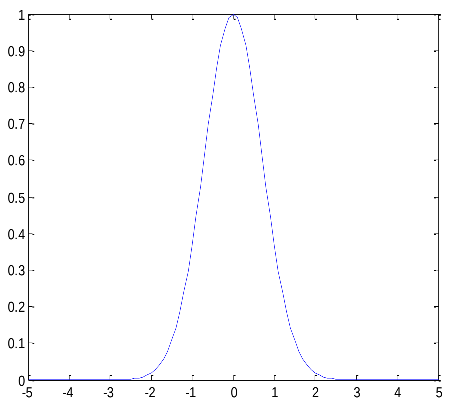


Figure 4:-plot of a Gaussian function

Second Layer:- This layer of the network is a standard linear layer:-

$a^2 = w^2 a^1 + b^2$………………(xix)

Experimentation:-
For proving that RBF Neural Network (by using formula newrb or newrbe) is faster than the directly use radbas function in MLFFNN, I have taken area of face recognition, because it is most popular area of research in Biometrics System, Computer vision, access control system etc.

There are generally two process, I use in this research. First of all I construct the network and train it. Second I recognize new image from this trained network.

(M) There are following methods by which we construct the network:-
Algorithm:-
(i) Step 1:- Take all the images of database. (n-sample of one image, similarly n-sample of second image……………..………upto m-images. i.e. total no of images=mn)

(ii) Step 2:-Do some pre-processing on it. There are following steps of pre-processing:-
Step (a):- Resize it into 15*15, by using formula (imresize) (note-small size taken due to fast processing).

Step (b):- Then we convert RGB to Grey by using formula (rgb2gray). (note- for identification of a person there is no need of colors, so we remove it, by which system becomes very fast.)

Step(c):- For looking good contrast and removing some illumination effect from image, I do histogram equalization by using formula (histeq).

Step (d):- do double operation on it.

(iii) Step 3:- Then I create new database, which is called pre_processed_Image database.
In my database consists m.n –images and size of all mn images is=15x15.

If (Network=multi-layer feedforward Neural Network):-

(iv) Step 4:-Then I create Neural Network. There are following methods by which, We create neural network:-

Step (a):- For the inputs layers, I take all thepre_processed_image database.

Step (b):- Convert it into column_wise_ matix.

Step(c):- I take some initial weight and bias of Hidden layer.

Step(d):- I take activation function of hidden layer radbas(Gaussian Function).

Step(e):- For output layer, I take some initial weight & bias of this layer.

Step(f):- Then I take purelin activation function.

```
net2=newff(inputs, targets, 20)
            net2.layer{1}.transferFcn='radbas';
            net2.layer{2}.transferFcn='purelin';
```

(v) Step 5:-Now, for performing a network, I take following value:-

```
            net2.trainParam.min_grade=1e-10;
            net2.trainParam.goal=1e-10;
```

(vi) Step 6:-Now, take some iteration (epochs) for proper assigning o weights & bias.

```
            net2.trainParam.epochs=1500;
```

(vii) Step 7:- Then, I train this network.

```
            [net2, tr]=train(net2, inputs, targets)
```

By the above process I get exact weight and bias of the network of both layers.

End

Else if (Network=RBF network)

(iv) Step 4:- After doing above step from 4(a) to 4(b), I use directly newrbe function to making RBFNN, by the following manner:-

```
            net2=newrbe (inputs, targets)
```

(v) Step 5:- Suppose I do not do any training parameter on it and do following things:-

```
             % Simulate the network
            sim_test=sim (net2, inputs);
            % Showing indices where images are found.
            output=vec2ind(sim_test);
```

end

(N) There are following methods by which we recognize the network:-

Step 1:- I take new images.
Step 2:- Then I do all the (step II) of above algorithm.
Step 3:- Now, I convert it into column_wise matrix.
Step 4:- Now, I simulate it with the network.
Output 1=sim (net2, new_inputs)

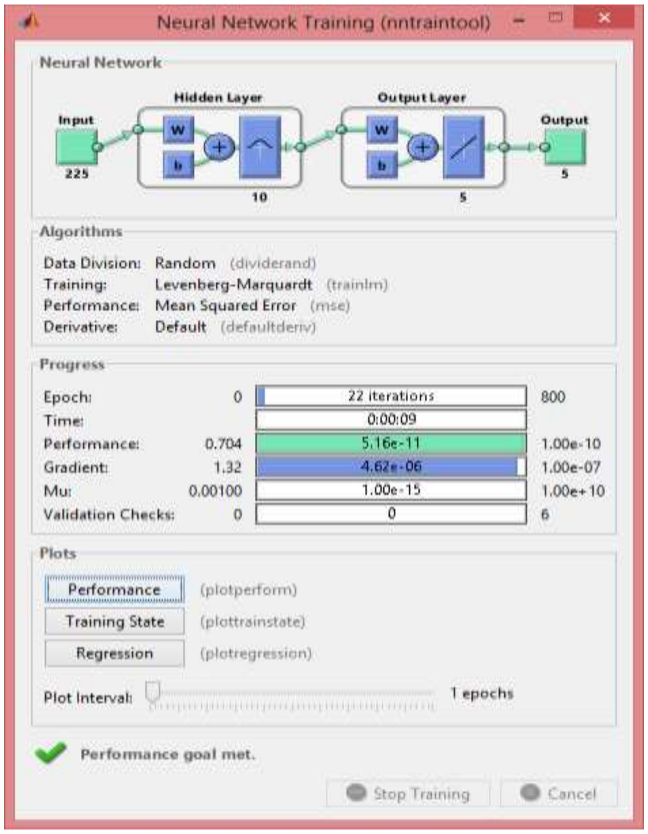
Step 5:-By taking some threshold value, I recognize the new_image with existing database image.

Result and Neural Network Diagram:-

Step 1 of above experiment:-

Figure 5:-25 images of database for training purpose.

Step 4 of above algorithm



Algorithms

Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Derivative: Default (defaultderiv)

Progress

```
Epoch:                0        22 iterations     800
Time:                          0:00:09
Performance:          0.704    5.16e-11          1.00e-10
Gradient:             1.32     4.62e-06          1.00e-07
Mu:                   0.00100  1.00e-15          1.00e+10
Validation Checks:    0        0                 6
```

Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)

Plot Interval: 1 epochs

Performance goal met.

Figure 6:- Multi-layer perceptron using Gaussian function in First layer

>>recognition
Recognize................................!!!!
He is 99.9999 percent A1 to A5

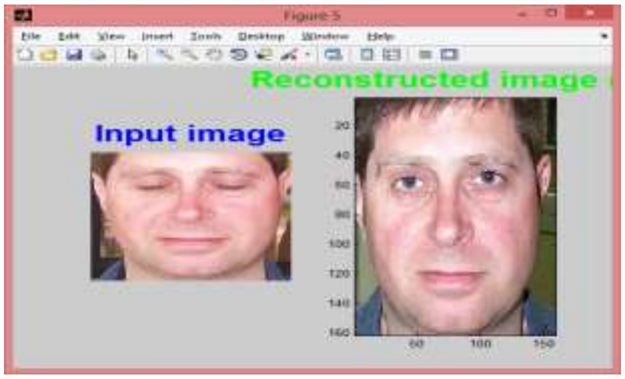
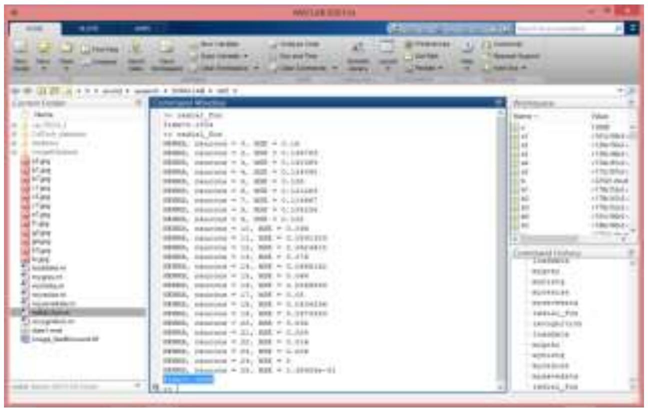


Figure 7:-Recognition using Multi-layer perceptron



```
[illegible]
```

Figure 8:- use newrb in MATLAB

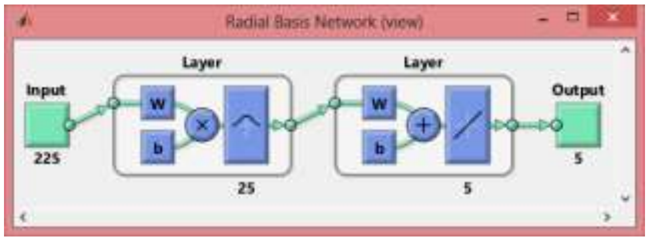


Figure 9:-Radial Basis Function in first layer

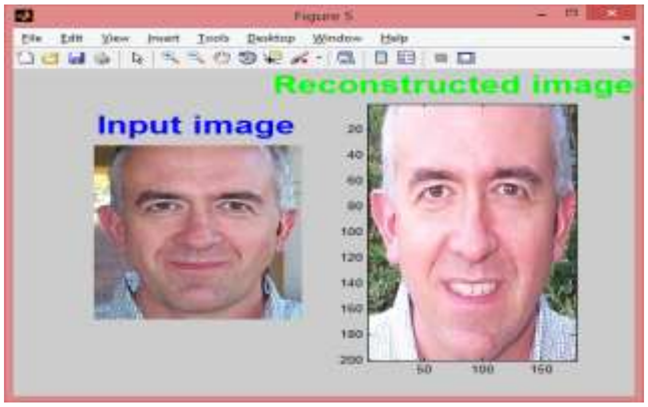


Figure 10: Recognition using Radial Basis Function

Databases:-
For experiment in this project, I use Caltech 101 database.

CALTECH 101
Caltech 101 database is created in California Institute of Technology in September 2003. This database contains color-full digital images. This database is compiled by Fei-Fei Li, Marco Andreetto, Marc 'Aurelio Ranzato and PietroPerona. This is used for facilitate computer vision research and technique. This is mainly used for image recognition, classification and categorization. Caltech 101 contains 9146 images. All the images are spilt into 101 distinct object categories such as faces, watches, ants etc.

This is also used by Paul Viola and Michael J. Jones for training on 4916 hand-labeled face in real-time face detection method.

Conclusion:-

IMAGES(MN)	GAUSSIAN FUNCTION IN MLFNN	RBFNN
	TIME (SECONDS)	TIME (SECONDS)
15(3*5)	4.8036	0.080042
25(5*5)	10.3691	1.9

Table 1:- Construction and Training Time of MLFNN and RBFNN

By the above result, we say that RBFNN is faster than we use Gaussian Function in MLFNN.

Software:-
MATLAB

References:-

Reference Research Papers:-

1. S. Thakur, J. K. Sing and others, "Face Recognition using Principal Component Analysis and RBF Neural Networks", First International Conference on Emerging Trends in Engineering and Technology, Page(s):695 – 700, 2008 IEEE.
2. KhairulAzha A. Aziz and other, "Face Detection using Radial Basis Function Neural Networks with variance Spread Value" , 2009 International Conference on Soft Computing and Pattern Recognition, Page(s):399 – 403, 2009 IEEE.
3. Yong Zhang and ZhiMaoXue, " RBF Neural Network Application to Face Recognition", 2010 International Conference on Challenges in Environment Science and Computer Engineering, Page(s):381 – 384, 2010 IEEE.
4. Mathew Turk and Alex Pentland, "Eigenfaces for Recognition", Journal of Cogitive Neurosciences, Volume-3.
5. M. Nandini, P.Bhargavi, G. Raja Sekhar, "Face Recognition using Neural Networks", International Journal of Scientific and Research Publications, Volume 3, Issue 3, March 2013 ISSN-2250-3153.
6. Ajeet Singh, Bhupesh Bhatia, VijayRajShokeen and Dr. B.K.Singh, "Comparison of PCA,LDA, ICA, SVM & HGPP", International Conference on Advances in Electronics, Electrical and Computer Science Engineering-EEC 2012.
7. MengJooEr. Member, IEEE, Shiqian Wu and Jumwi Lu, "Face Recognition using RBF Neural Networks", International Conference on Decision & Control Phoenix, USA, December 1999, Page(s): 2162 – 2167, 1999 IEEE.
8. Sue Inn Ch'ng, KahPhoolSeng, Li-MinnAng "Modular Dynamic RBF Neural Network for Face Recognition", 2008 IEEE.
9. VandanaAgarwal and SurekhaBhanot. "Evolutionary Design of Multiquadric Radial Basis Functions Neural Network for Face Recognition", Birla Institute of Technology and Science, Pilani India, IEEE.
10. Virgindia Espinosa-Duro, Electronic and Automatic Department, "Biometric Identification System using a Radial Basis Network" Page(s): 47 – 51, 2000 IEEE.
11. Kalpana C. Jondhale and Dr. L.M. Waghmare, "Improvement in PCA performance using FLD and RBF Neural Networks for Face Recognition", Third International Conference on Emerging Trends in Engineering and Technology, Page(s):500 – 505, 2010 IEEE

12. Gabor Feature Selection and Improvement Radial Basis Function Networks for Facial Expression Recognition” © 2010 IEEE.
13. Marcos Faundez-Zanuy and Enric Monte-Moreno, “Face Recognition using a Radial Basis Function Classifier”, © 2006 IEEE.
14. Sue Inn Ch’ngKahPhoolSeng and Li-MinnAng, “Modular Dynamic RBF Neural Network for Face Recognition’, © IEEE.
15. Thai Hoang Le, “Research Article on Applying Artificial Neural Networks for Face Recognition”, © IEEE.
16. Nilind Sharma and Asst. Prof. Shiv Kumar Dubey, “Face Recognition Analysis Using PCA, ICA and Neural Network”, International Journal of Digital Application & Contemporary research, volume 2, Issue 9, April 2014.
17. Digital Color Image Processing- Andreas Koschan&MongiAbidi, Publication-Wiley Interscience, A John Wiley & Sons, Inc. Publication.

Reference Books:-

18. Digital Color Image Processing- Andreas Koschan&MongiAbidi, Publication-Wiley Interscience, A John Wiley & Sons, Inc. Publication.
19. Image Processing with MATLAB- Omer Demirkaya, Musa HakanAsyali&PrasannaK.sahoo , Publication- CRC Press London New York.
20. MATLAB for Neuroscientists-Pascal wallisch& Michael lusignan et al publication-Elsevier
21. Neural Networks, A Comprehensive Foundation- Simon Haykin
22. Neural Network Design- Martin T. Hagan
23. Artificial Neural Networks- B. Yegnanarayana
24. Handbook of Face Recognition, 2nd Edition, Springer publication

Reference Internet sites:-

[1] www.face-rec.com
[2] www.ieee.com
[3] www.sciencedirectory.com