
Faculty of Engineering Science
Celestijnenlaan B-3001 Heverlee

# Artificial Neural Networks (B-KUL-H00G8a)



| | |
|---|---|
| **Student Name** | Sonai Pandiyan Subramanian |
| **Student Number** | R0648684 |
| **Programme** | Masters in Mechanical Engineering |

# 1 Backpropagation in Feedforward Multi-layer networks

The aim of the exercise is to compare different optimization algorithm in approximating a non linear function

$$y = sin(x^2) \tag{1}$$

using a neural network with 1 hidden layer. Each algorithm is checked for speed and performance criteria.
For checking the performance of the algorithm RMSE ( Root mean square error) metric is used

$$RMSE = sqrt(y - \hat{y}) \tag{2}$$

where y is the actual value of the function & $\hat{y}$ refers to the value predicted by the network

## 1.1 Function without noise

when no noise is added to the underlying function the output performance (RMSE) of the network for different algorithms is shown in fig(1(b)) also fig(1(a)) shows how the performance of network varies with increase of epoch.

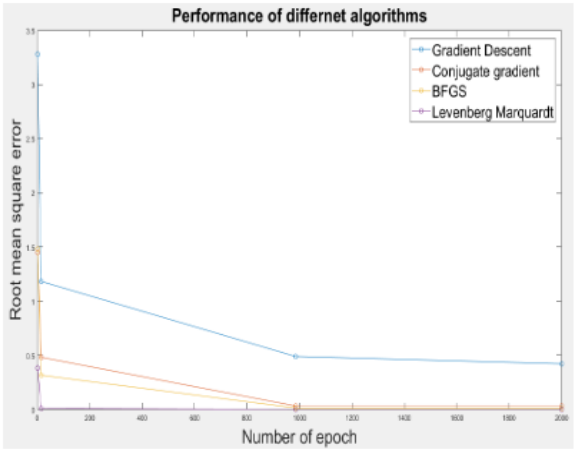



Figure 1: # epoch vs performance of different algorithm

(a)

| Algorithm | RMSE at end of 2000 epoch |
|---|---|
| Gradient Descent | 0.4215 |
| Conjugate Gradient | 0.0321 |
| BFGS | 0.0082 |
| Levenberg Marquardt | 4.3924*10-5 |

Table 1: Performance of different algorithm

(b)

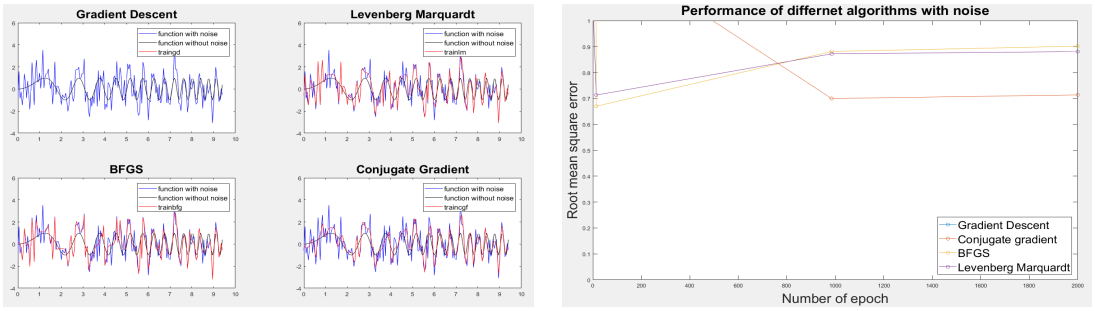
Figure 1: Performance of different algorithm

The speed of the network in descending order is as follows, Gradient Descent, Conjugate Gradient, Levenberg Marquardt, BFGS

## 1.2 With noise in the underlying function

With noise in the function the gradient descent algorithm performance keep on increasing which crossed the limit so output of network is 'NAN'.

To find the generalization of the networks RMSE is calculated between the output of the network and the actual underlying function instead of the function with noise and the output of this error with respect to number of epoch is shown in figure(2(b)) it is clearly seen that without any validation methods such as early stopping or n fold cross validation techniques or regularization the network starts to overfits and loses generality in case of Levenberg Marquardt




(a) (b)

Figure 2: (a)Compares the output of network with different algorithms; (b)Compares the performance of different algorithm when the function has noise;

and BFGS whereas in conjugate Gradient method the network generalizes the function better than with other methods.

| **Algorithm** | **RMSE (function with noise)** | **RMSE (actual underlying function)** |
|---|---|---|
| Gradient descent | NAN | NAN |
| Conjugate Gradient | 0.7206 | 0.7135 |
| BFGS | 0.4946 | 0.9015 |
| Levenberg Marquardt | 0.5290 | 0.8807 |

Table 1: Performance of different algorithm on function with noise

*Trainbr* because of availability of regularization the function is generalized better(see fig(3)), Table(2) shows the RMSE with the noisy function and actual underlying function. It is found that the convergence of Bayesian Regularization algorithm is very fast. Speed of different algorithm in descending order is *traincgf,trainbr,trainlm,trainbfgs.*

| **Algorithm** | **RMSE (function with noise)** | **RMSE (actual underlying function)** |
|---|---|---|
| Bayesian | 0.5844 | 0.7657 |
| Levenberg Marquardt | 0.5416 | 0.8671 |
| BFGS | 0.4521 | 0.8672 |
| Conjugate Gradient | 0.6141 | 0.7401 |

Table 2: comparision of Bayesian Regularization with other algorithms

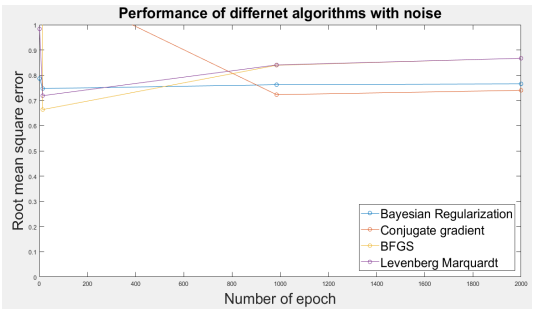



Figure 3: Comparison of different algorithms with Bayesian Regularization

## 2 Recurrent Neural Network

The aim of this exercise is to find the capability of Hopfield network and application of Elman network to the Hammerstein system.

### 2.1 Hopfield network

The given data set contain hand written digits in binary image with 240 pixels. To this Gaussian noise is added by specifying the standard deviation of the Gaussian noise. Hopfield network is used to retrieve the original image from this noisy image. And the Metrics used for testing the correctness of the output is

- When the output has reached its attractor states the number of elements(out of 10 i.e. 0-9) which are predicted incorrectly by the network.

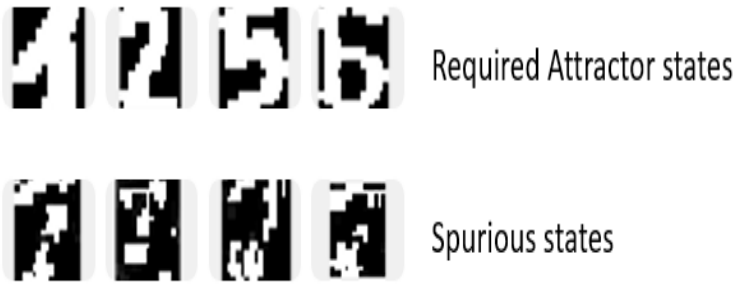
To check whether final attractor state is reached the error between predicted and required output is found and it is compared with the error in the previous iteration if it is same then it is considered to have reached the final attractor state.

It is also checked whether the output of the network is one of the attractor states or some spurious states.

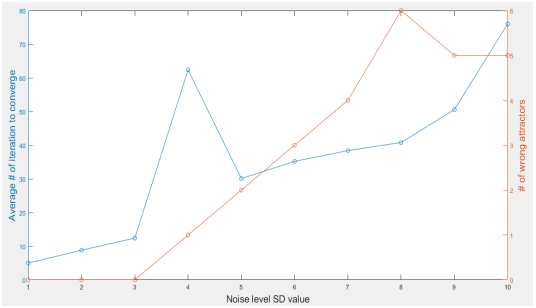
The neurons in the Hopfield network uses *satlins* as transfer function so they are saturated at +1 and -1.

And the reason why we have spurious states in Hopfield network is how these network select its attractor states The learning rule projects the given input into the attractor space and finds the distance of the projected (input) vector to each attractor vectors and assigns the output to the attractor state which is closest to the projected input vector. To have a spurious state the projected vector should be exactly in between two attractor states also the attractor states can be non orthogonal to each other so the distance between them is not maximum.

Figure(4(a)) shows some of the spurious state attracted by the network and the actual required attractor state.




(a)




(b)

Figure 4: (a): Shows the spurious states and the corresponding correct attractor states (b):Performance of hopfield network on noisy image

It is noticed that when the noise level is increased the number of iteration required by the network to converge to attractor state increases and also the number of items with wrong attractor states also increases. Figure(4(b)) shows the average number of iterations taken (average of 10 networks) and the average wrong attractor states for noise level from 1 sigma to 10 sigma.

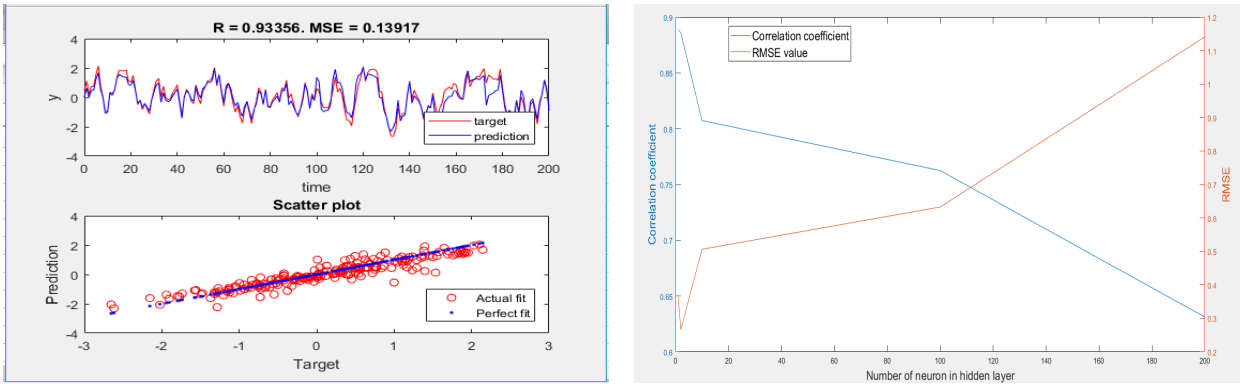
## 2.2 Elman Network

An Elman Network is created to predict the *Hammerstein system* and the parameters that can be varied in the network are

- Number of training data's
- Number of epoch
- Number of neurons in the hidden layer and
- The type of activation function used.

The metrics used to see the quality of the network is the R value for the training data and RMSE value for the testing data

The figure (5(a)) shows the output of the network trained with 1 neuron in the hidden layer with default activation fuction(*tansig*). It is also seen that increasing the number of neurons not necessarily improve the performance of the network figure(5(b)) shows the average performance of network's(average of 10 networks) for different number of neurons in the hidden layer. In order to get better performance with higher number of neurons in the hidden layer more training data and more number of epoch is required than with less number of neurons in the hidden layer.




(a) (b)

Figure 5: (a): Shows the Output of Elman Network (b): Number of neurons vs performance of network

| # of neurons in hidden layer | Corr Coeff | RMSE |
|---|---|---|
| 1 | 0.888923 | 0.363317 |
| 2 | 0.885979 | 0.266134 |
| 10 | 0.807286 | 0.506388 |
| 100 | 0.762426 | 0.63264 |
| 200 | 0.631076 | 1.141286 |

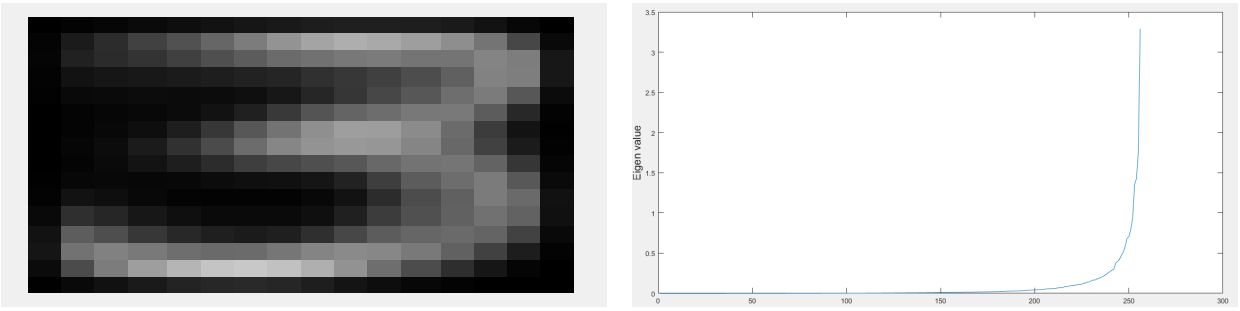
Table 3: Performance of Elman Network for different # of neurons

Table 3 shows the performance of network for different number of neurons in the hidden layer. The performance of network with 200 neuron got improved when the number of epoch is increased form 500 to 1000 by the following figure R(Corr coefficient) became 0.7986 and RMSE became (0.6414)

# 3 Unsupervised Learning PCA and SOM

## 3.1 PCA

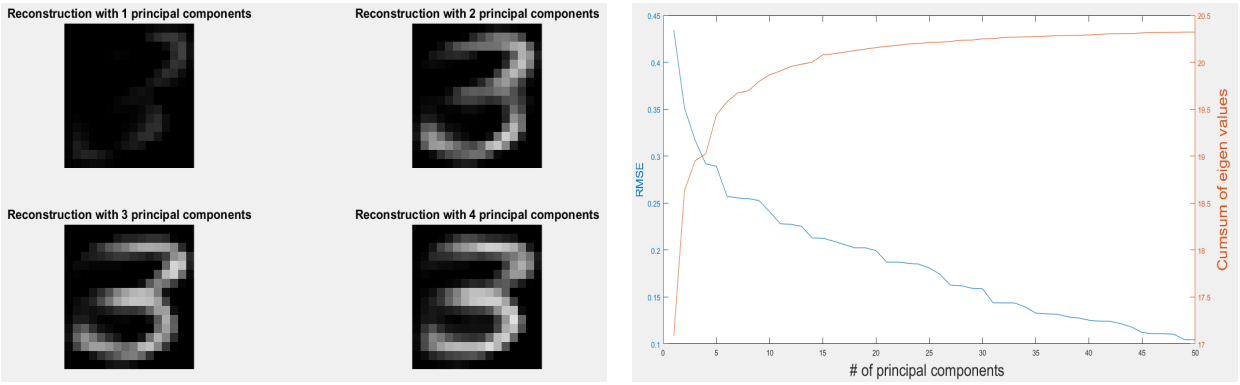
The aim of this exercise is to use PCA on hand written digit of number 3. Data set provided has 500 images of hand written number 3. The mean of these samples is shown in fig (6(a)).




(a) (b)

Figure 6: (a): Shows the mean of the samples of hand written digit 3 (b): Eigen values of the covariance matrix of handwritten number 3

Covariance of the total dataset is taken and Eigen values of the covariance matrix is found the values are plotted and shown in fig(6(b)).




(a) (b)

Figure 7: (a): Shows the reconstruction of image with 1 2 3 4 principal components. (b): Shows the Error in reconstruction for first 50 principal components.

First 1 to 4 principal components are used to reconstruct the image and it is shown in figure(7(a)). To calculate the error RMSE (Root Mean Square Error)metrics is used, Figure(7(b)) shows the error vs considering 1 to 50 principal components. It is seen that error decreases exponentially and beyond certain number of principal components the Regeneration error is very less.

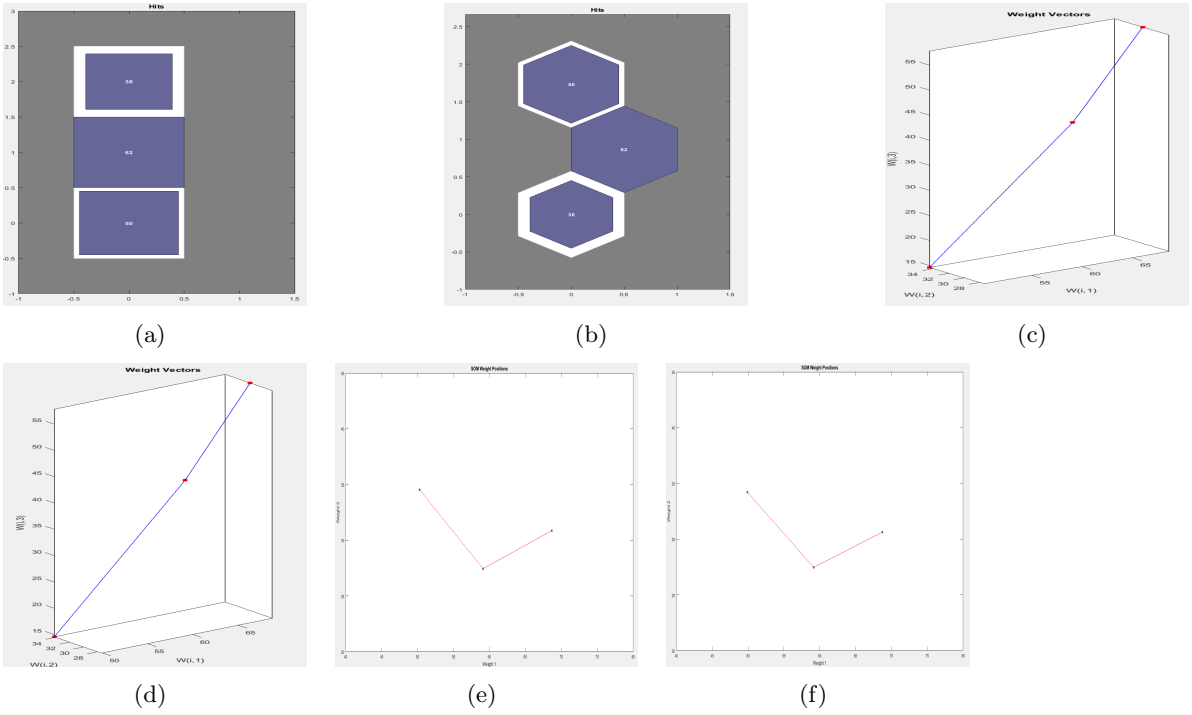
It is found that even when all the principal components are included there is a very small amount of error $5.7*10^{-6}$.

Figure(7(b)) also shows the sum of all Eigen values eliminating one of the component. It is seen that the later principal components does not add up much to the sum.

### 3.2 SOM

The aim is to use Self organizing map on data set *Iris* which has 150 flower samples in which 4 different features are measured and it has 3 different types of flowers. SOM is used to group the data into 3 different cluster and the metrics used to compare the effectiveness of the network is ARI (Adjusted Random Index).

Possible parameters that can be changed to change the performance of network are Topology function (*hextop, gridtop,randtop*) and the Distance function (textitlinkdist,dist,mandist).




(a) (b) (c)

(d) (e) (f)

Figure 8: Shows sample hits, weight vector and weight position for SOM with grid top and hextop topology

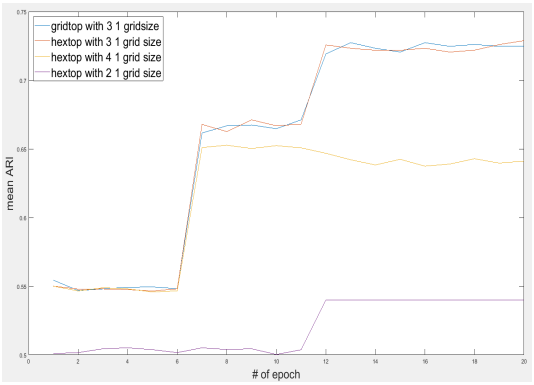


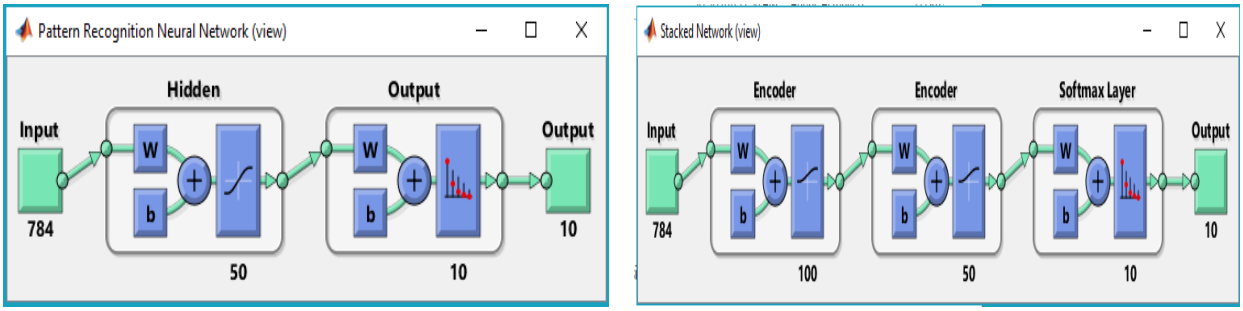

Figure 9: Comparison of different algorithms with Bayesian Regularization

It is seen that since the original dataset has 3 different classes if the grid size of the network if either increased or decreased to other different cluster then the ARI index value is reduced this can be seen in figure(9). It is also noticed that max possible ARI is around 0.74.

# 4 Deep Learning

## 4.1 Stacked Auto Encoders

The aim of the exercise is to train a deep neural network to classify the hand written numbers. Each of the hand written numbers is a image of 28 by 28 pixels which is 784 pixels in total. First a stacked auto encoder is used to reduced the input dimension from 784 to 50 in 2 steps 784 to 100 and 100 to 50 this 50 dimensional input is used for classification.




(a) (b)

Figure 10: (a): Shows the architecture used to achieve best performance on feed forward net (b): Shows the deep network architecture used

It is seen that the performance of deep neural network is lesser than the performance of shallow feed forward net (see table(4)) but when the weights of the deep networks are updated with finetuning method the perfoemance of deep network is improved and goes beyond the conventional shallow feed forward network.

| Network | Performance 3 layers | Performance 4 layers |
|---|---|---|
| FeedForward net | 85,78 | 75,22 |
| Deep Network | 96,68 | 95,66 |
| Deep Network with fine tuning | 99,76 | 99,66 |

Table 4: Comparison of deep and shallow feed forward net

Figure(11) shows the confusion matrix of different networks.

Fine tuning plays an important role in improving the performance of the deep network.The reason for this is as follows, When number of layers is increased the network can learn more non linearity in the underlying function but in a shallow feed forward network the because of vanishing gradient the weights of the first few layers are not adjusted clearly because of this performance of the network will be poor. But in in a deep network the dimension of the input is reduced in step by step fashion and when fine tuning is introduced the weights are already adjusted close to the local minimum slight fine tuning will improve the performance of the network far better than the shallow feed forward network.

(a) (b) (c)

Figure 11: confusion matrix (a): Shows the confusion matrix of ordinary feed forward net (b): Shows the confusion matrix of deepnet
(c):Shows the confusion matrix of deepnet after finetuning

## 4.2 Convolutional Neural Network

In this exercise image database from caltech is used and Pre-trained network *'Alex net'* is used. the input image size is 227x227x3

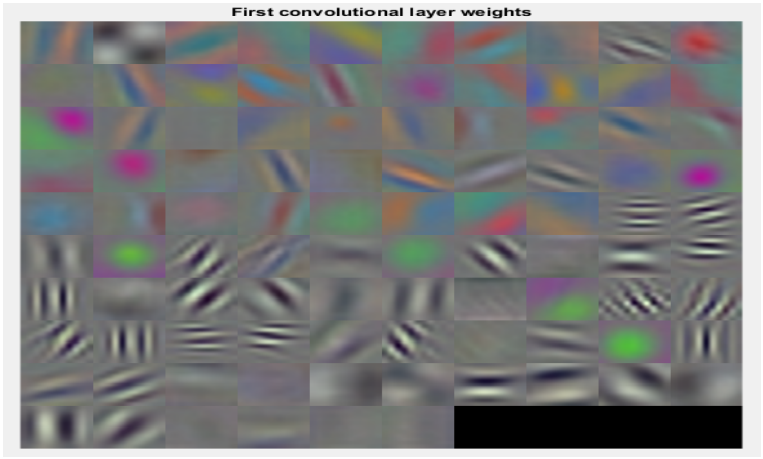



Figure 12: Weight of first convolutional layer

Figure(12) shows the weight of the convolutional layer 2. It has 96 weight which represents 96 filters this represents the first layer abstraction the network will look for such patterns in the image.It is also seen that some of the weights look for colour pattern while others dont.

Convolutinal layer is specified with filter, size stride and padding.The amount by which the filter slides is called stride. And padding is adding extra pixels in the edges of the images with value zero.

Since the filter size of convolution layer is 11x11 with stride of 4 and no zeropadding is added the output of this convolution layer in layer 2 is ((227-11+0)/4)+1 = 55. Maxpooling layer has the filter size of 3x3 with stride 2 therefore the output of this layer 5 has the output size((55-3)/2)+1 which is 27. Therefore the size of input of layer 6 conv2 layer is 27*27 Number of neurons used for the final classification task has 1000 neurons.Comparing this with the first convolutional layer which has (11*11*3+1)*96=34944 neurons.

# 5 Nonlinear Regression With MLP

The aim of this project is to find the underlying non linear function using Feed Forward Neural network. The function to be found out is a weighted linear sum of 5 nonlinear function weighted by 8, 8, 6, 6, 4 and divided by sum of the weights . Five non linear function is sampled at 13600 points and its corresponding output are given in the dataset. Figure(13) shows the function that need to be approximated

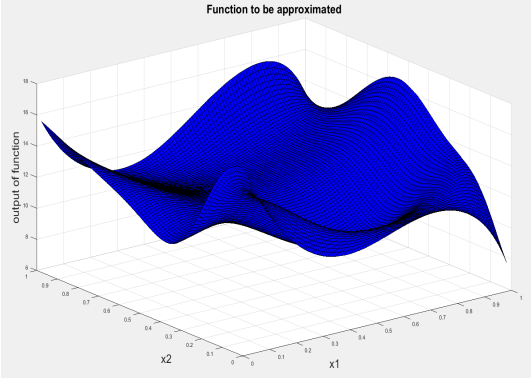



Figure 13: Nonlinear function that needs to be approximated

Metrics used to compare different network and network configuration is the best performance of the test set in each training case where the performance function while training is the mean square error (MSE)

Initially using 2 hidden layer with 16 neurons in each layer 4 different algorithms are tested *trainlm,trainBFGS,traincfg,trainbr* and the best performing algorithm is chosen for further improving the approximation.Figure(14) shows the output of function approximated by each algorithms.From table 5 it is clearly seen that *trainbr* outperforms all other algorithms.

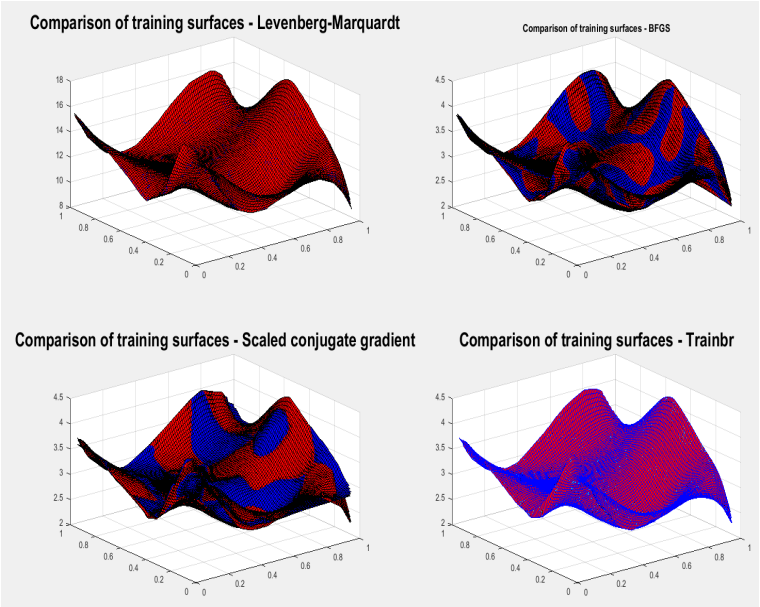



Figure 14: Performance of different algorithms

| Trainlm | TrainBFG | TrainCGF | Trainbr |
|---|---|---|---|
| 8,9146E-09 | 1,3619E-03 | 2,7744E-03 | 1,9857E-11 |
| 5,5777E-09 | 5,0943E-04 | 1,4911E-03 | 3,2042E-11 |
| 4,1391E-09 | 6,8445E-04 | 5,1995E-03 | 1,0405E-10 |
| 3,3782E-09 | 3,3819E-04 | 4,2209E-03 | 5,9924E-11 |
| 9,0607E-10 | 2,2773E-04 | 3,0540E-03 | 4,0487E-11 |

Table 5: Performance of different algorithm

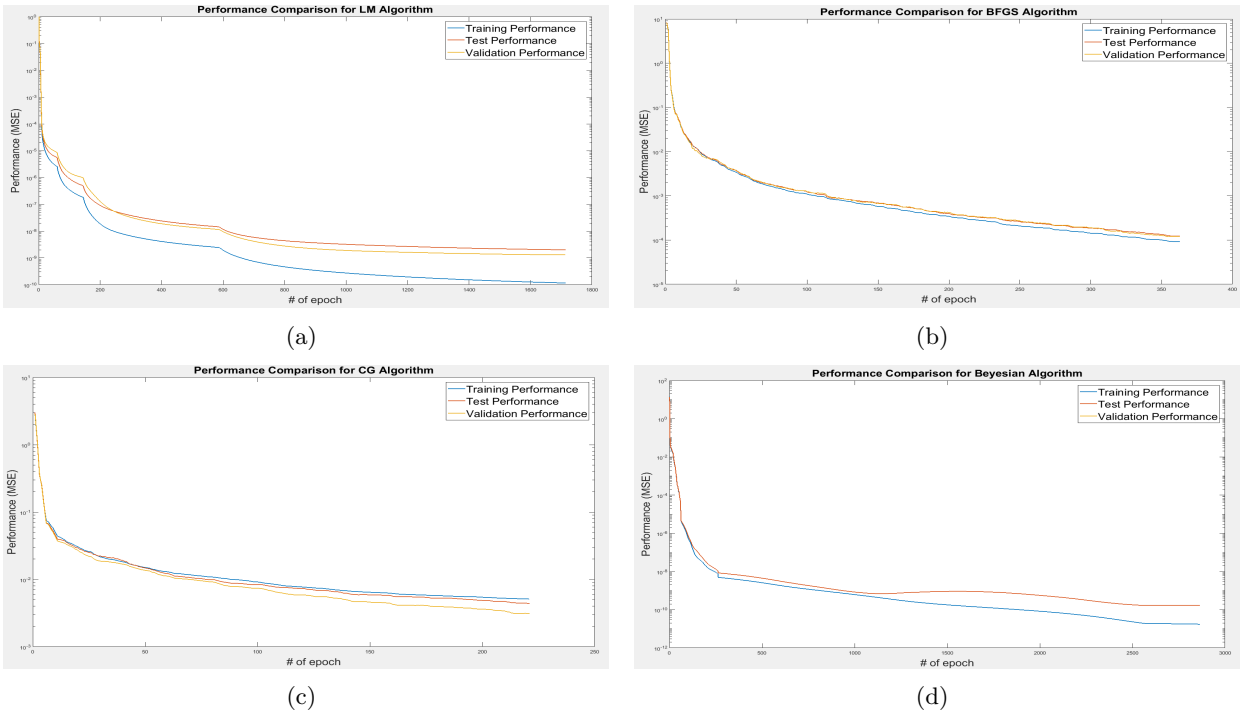



Figure 15: (a): Performance of LM Algorithm. (b): Performance of BFG Algorithm. (c):Performance of conjugate Gradient Algorithm. (d): Performance of Bayesian Regularization Algorithm.

| 2 layers | 3 layers | 4 layers |
|---|---|---|
| 8,5801E-04 | 1,4554E-04 | 1,7184E-05 |
| 8,2808E-04 | 1,0089E-04 | 1,1681E-05 |
| 9,9245E-04 | 1,2448E-04 | 2,7988E-05 |
| 3,9860E-04 | 5,9267E-05 | 2,4318E-05 |
| 4,9064E-03 | 1,2649E-04 | 4,2351E-05 |

Table 6: Performance of network with 4 neurons per layer and different number of layers

It is also seen that with the increase in number of layers increase the performance of the network. To get the best network the following parameters are used

- Optimization algorithm - trainbr
- Number of layers - 4
- Number of neurons per layer - 16

figure(16) shows the output of the network with best performance the performance value is 1.25*e-10.

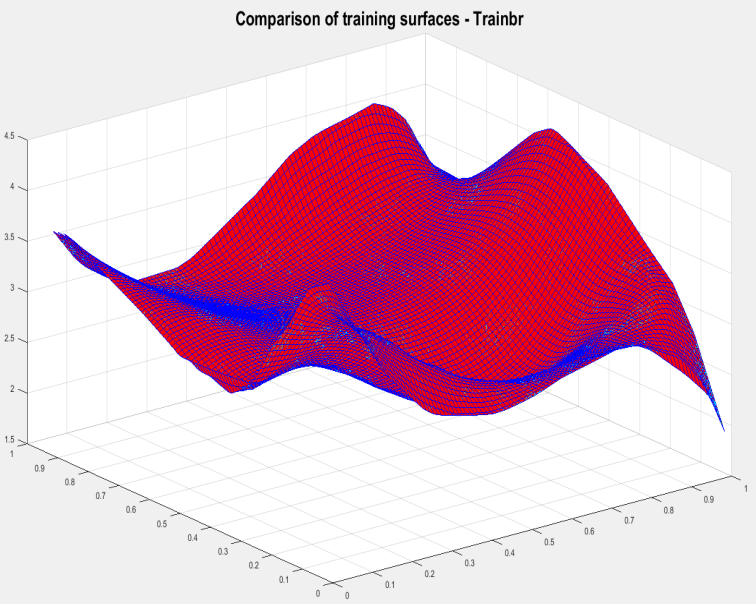



Figure 16: Best approximation of the function

**How to further improve the network:**

If the network has some noise available, instead of training multiple network and selecting the best network we can use combined network which is linear combination of networks made of either different networks trained with *trainbr* or combination of networks trained with different algorithms such as *trainbr,trainlm.*

# 6 Classification

Aim of this project is to develop a model to classify wine as per its quality. The given data set contain certain number of samples of red and white wine's on which 11 different features and measured and to denote the quality, number 3-8 is given. Here model is created to classify white wine that belong to class 4 from the wine that belong to class 5 and 6. Metrics that is used to check the performance of the network is CCR(Correct Classification Ratio).

## 6.1 FeedForward network

First all 11 different features are fed as an input for the network and the quality is fed as desired output while training the network. Some of the parameters that affect the performance of the network are

- number of layers
- number of neurons per layers
- activation function in the hidden and output layer
- optimization algorithm used

Other parameters that influences the is number of epoch. Since the possible combinations are large following constraints are included based on previous exercise session.

- Number of epoch is set to 15000.
- Number of hidden layer is not increased beyond 4
- Number of neurons in all hidden layer is maintained as same
- Activation function of hidden layers are maintained *tansig* and the output layer is maintained as *softmax*
- optimization algorithm is set to dafault *trainscg*
- Default performance function *cross entropy* is used.

Here one at a time approach is used to find the hyperparameters

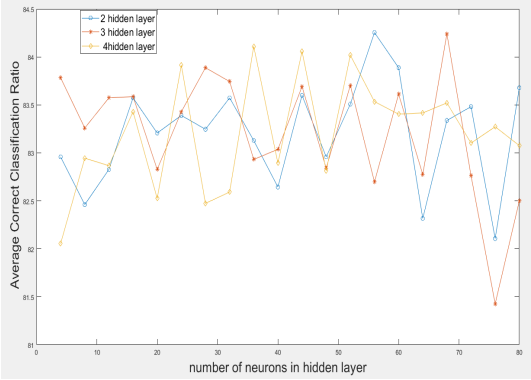



Figure 17: Average Network performance with different number of layers and different number of neurons

The figure(17) shows the average performance(average of 10 networks) with increase in number of neurons per layer and number of layers. From the above graph following things can be deduced increasing number of neurons not always increase the performance.

Based on the above results the following Hyper parameters are chosen

- Number of layers 2
- Number of neurons in hidden layer 56

To find the best hyper parameter setting following methods can be used

- Grid search
- Random search
- Bayesian optimization

since this activity is time consuming it is not done in this case.

Covariance matrix for the data set is calculated and its eigen value and eigen vectors are obtained. Figure(18) shows the eigen value of the correlation matrix for the training Data set

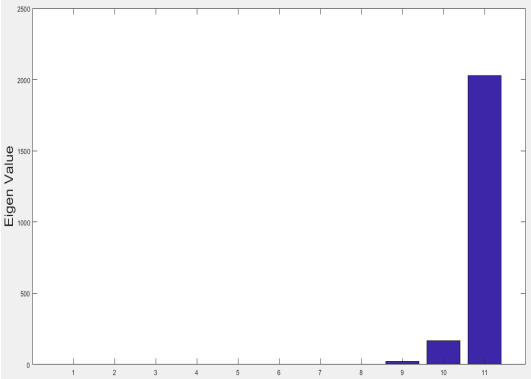



Figure 18: Eigen value of correlation matrix of the Wine data

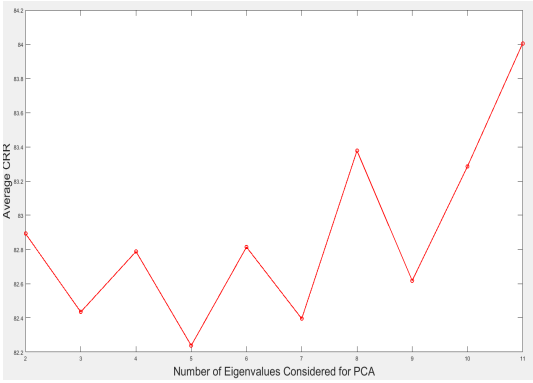



Figure 19: Principal components used vs classification accuracy

from fig(19) it is seen that when considering all principle components the average performance of the network is high.

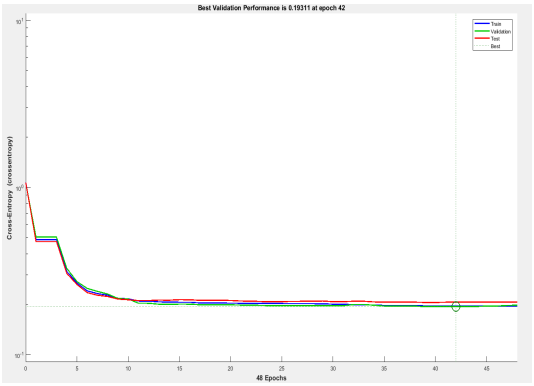



Figure 20: Principal components used vs classification accuracy

Figure(20) shows the performance of the network while training with all 11 principle components. max CCR of 86.2 is acheived while taking 11 principle components into account.which is higher than the max CCR achieved.

# 7 Character recognition with Hopfield Network

The aim is to investigate the retrieval capabilities of Hopfield Network. The given dataset contain images of capital letter of English alphabets with size 7x5 in black and white to this 8 more character in lower case s,o,n,a,i,p,d,y are added.

The figure(7) shows the first 10 alphabets in the database.

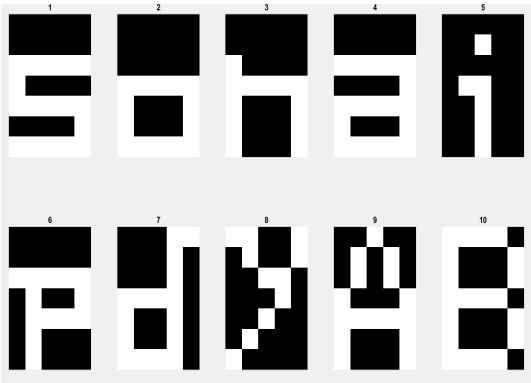
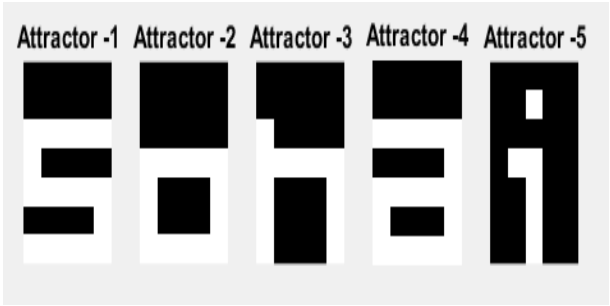
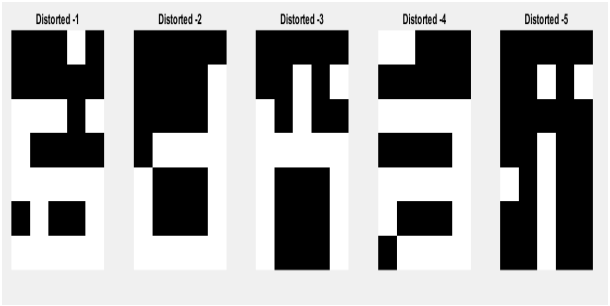



Figure 21: Shows the first 10 alphabets of the dataset

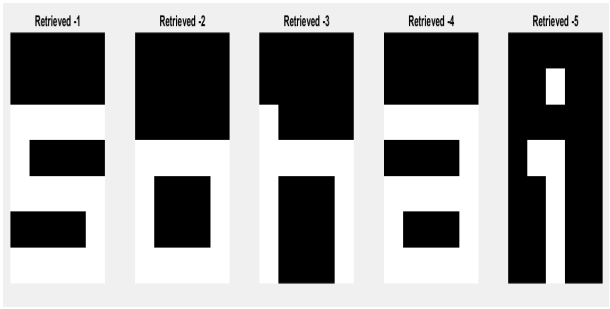
Hopfield network is created with first 5 letters as their attractor states. Three of the randomly selected pixel is inverted to the input image and then it is fed into the input for the network and the output is checked.



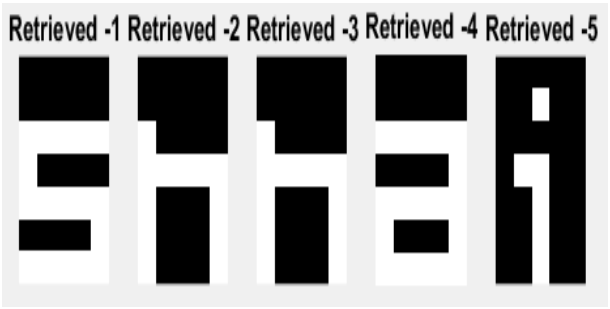

(a)




(b)




(c)




(d)

Figure 22: (a): Shows the First 5 attractor states of the network. (b): Shows the input image distorted. (c): Shows the output of the network. (d): Shows the output of the network with wrong attractor states.

It is seen that even though the network output goes to wrong attractor states as shown in

fig(22(d)), there was no spurious states found. But when the number of patterns to be stored is increased some of the spurious patterns were found.Also the number of spurious pattern is found to increase with the number of patterns to be stored. Figure(23) show some of the spurious states arrived by the network.

Number of patterns to be stored is increased form 1 to 34 and the error (number of wrong pixel) is recorded figure(24) number of patterns to be stored vs error ( number of error in pixel)

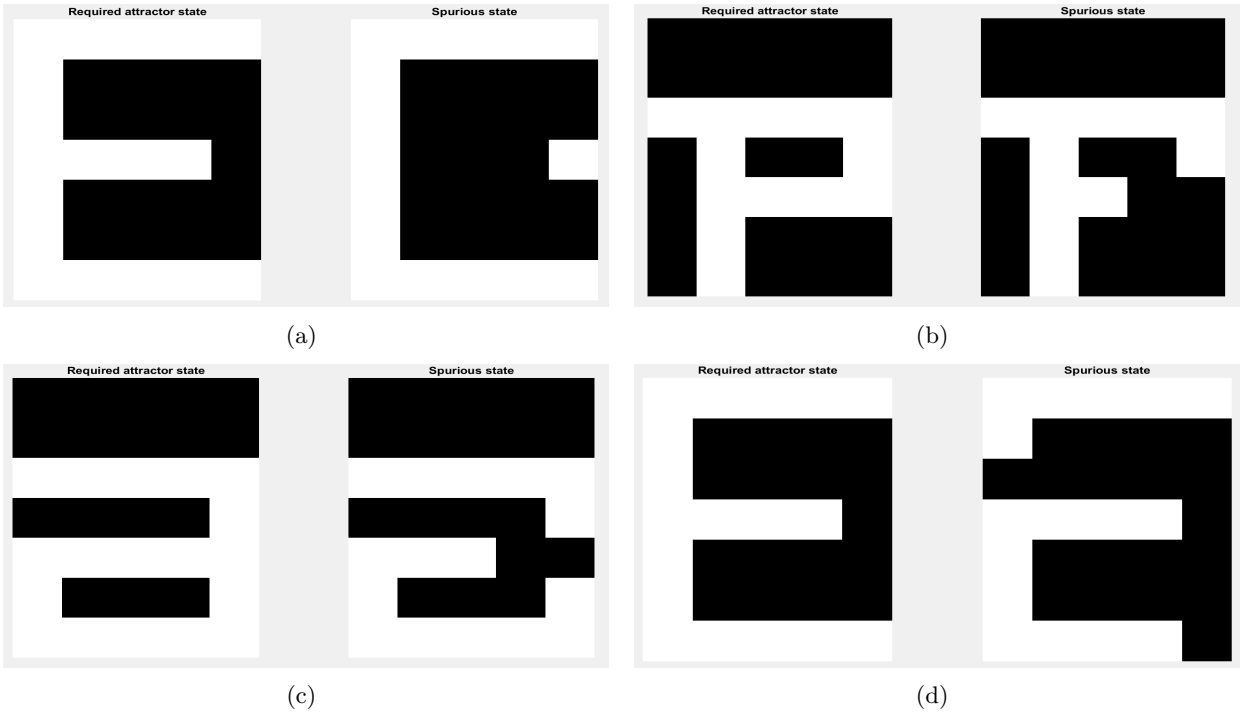



Figure 23: The figure shows various spurious state (on the right of every subplot) and the desired attractor state (on the left of every subplot)

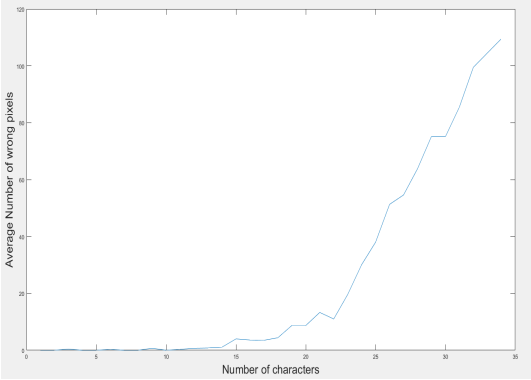



Figure 24: Shows the number of pattern to be stored vs average error

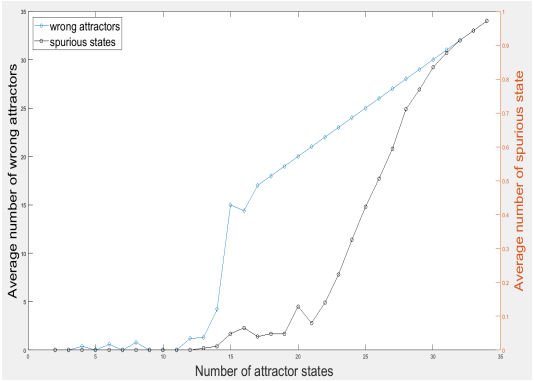



Figure 25: Shows the number of attractor vs number of wrong attractor and number of spurious states

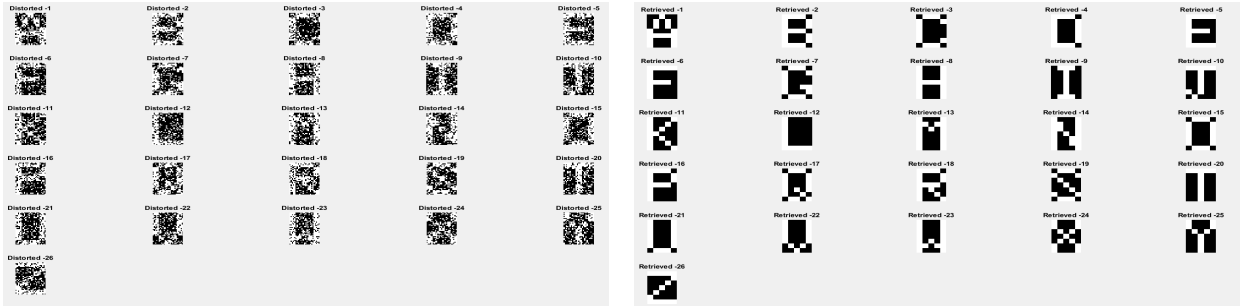
form figure(25) it is seen that if pattern to be stored is above 16 then all the output are converging to wrong attractor state.

The critical capacity of the network form the above trail is 4 wrong attractors are starting to appear while storing more than 4 patterns. According to Hebb rule the number of pattern that can be stored is given by the formula

$$capacity = n/sqrt(2 * ln(n)) \tag{3}$$

for input neuron n=35 (number of pixels in image) we get the number of patterns that can be stored is 13.12 ≈13. It is seen from the figure(25) from 13 the number of wrong attractors are steeply raising.

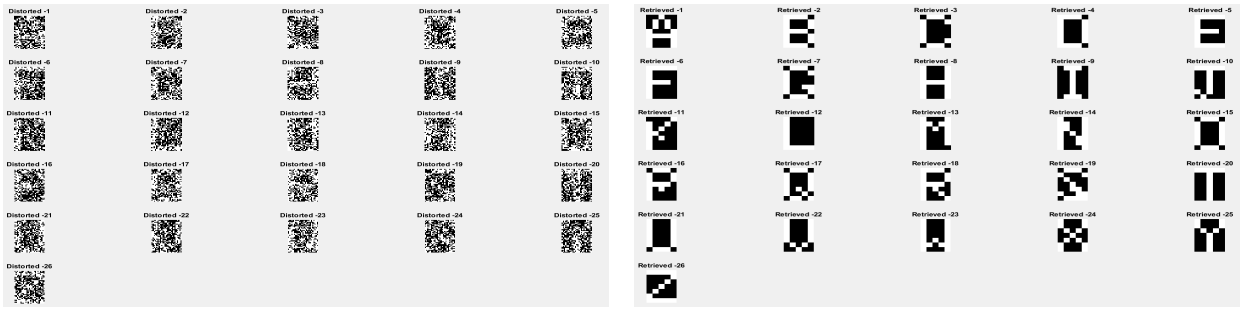
Inorder to store more characters either the learning rule should be changed from Hebbian learning rule or the input size has to be increased form 35 that is increasing the number of pixel this can be done by duplicating the pixels. Each pixel is again subdivided into 25 pixels in this way the size of image changes from 5x7 to 25x35 so the input size of the network becomes 875 this way the number of attractor states that can be stored according to Hebb rule becomes 234. Figure(26(a)) shows the distorted image and figure(26(b)) shows the retrieved attractor states.



(a) (b)

Figure 26: (a): 200 random pixels changed in the input (b): Output of the network

When Noise level is increased more we can find some spurious states for example in the following figure(27(b)) alphabet p and alphabet k has spurious state.




(a) (b)

Figure 27: (a): alphabets distorted by changing 300 pixels (b): Existence of spurious state