Chapter 4

Experiments and Simulation
Abstract:

Here in this chapter, we have two phases of experiments. In first phase, we have pattern storage neural network architecture of Hopfield type with 900 processing units to reflect the associative memory feature for storage and recalling of static images. In this phase, there were three methods of feature extraction used namely Edge Dilation (ED), Fast Fourier Transformation (FFT) and Self Organizing Map (SOM) to analyzing the performance of pattern storage network for storage and recalling. We have analyzed the storage capacity and recall efficiency of the pattern storage network for original images as well as distorted or noisy images. In second phase we have optimize the Hopfield Neural Network using sub optimal genetic algorithms. We have used crossover and mutation operator to enhance recalling of Hopfield Neural Network.
4.1 Introduction:

The pattern storage networks with their energy surfaces behave similar to the associative memory architectures of the brain where the complete information can be recalled with given only partial knowledge of the information content. The dynamical behavior of the neuron states strongly depends on the synaptic strength between neurons and the used state-updating scheme. Association is a salient feature of human memory. Associative memory models, known as content-addressable memories, are a class of the most extensively analyzed neural networks. A pattern can be stored in memory through a learning process. For an imperfect input pattern, associative memory has the capability to recall the stored pattern correctly by performing a collective relaxation search. Associative memories can be either hetero-associative or auto-associative. For hetero-association, the input and output vectors range over different vector spaces, while for auto-association, both the input and output vectors range over the same vector space.

Recalling of Patterns will better if the memorized patterns are considered effectively and if the information of pattern is encoded sufficiently and efficiently. Therefore to accomplish the efficient pattern information the features from the input stimuli should be extracted uniformly. There are lots of methods in literature which have been proposed for the feature extraction to consider the task of associative memory in Hopfield neural network architecture [157]. In this context the methods like Edge, Dilation, Fast Fourier Transform (FFT), Self Organizing Maps and Discrete Wavelet Transform have been identified. These methods are good ways for forming the pattern vectors to encode them into the pattern storage network as
associative memory. Frequency domain filtering i.e. FFT and Space domain filtering i.e. ED have proved to explore different aspects of the pattern spaces [158-160]. These pattern storage networks with their energy surfaces behave similar to the associative memory architectures of the brain where the complete information can be recalled given only partial knowledge of the information content [161].

The specification of the synaptic weights is conventionally referred to as learning. Initially Hopfield used Hebbian learning to determine the network weights and since then a number of learning rules have been suggested to improve its performance [162, 163]. The Hebbian rule has associated with it the advantages of being local and incremental. This means that the update of a particular connection depends on the information available on either side of the connection and also patterns can be incrementally added to the network. This learning rule exhibits the following limitations:

1. The maximum capacity is limited to just \(0.15N\) with binary input and \(\frac{N}{2 \log_2 N}\) with bipolar patterns, where \(N\) is the number of neurons in the network [164].
2. The recall efficiency of the network deteriorates as the number of patterns stored in the network increases [165, 166].
3. The network’s ability to correct noisy patterns is also extremely limited and deteriorates with packing density of the network.
4. New patterns could hardly be associated to the stored patterns.

Since Hebbian rule had the above-mentioned limitations, hence other learning rules were considered to improve upon the learning behavior of the Hopfield
Network. Pseudo inverse Rule is one of them. The standard pseudo inverse rule is known to be better than the Hebbian rule in terms of the capacity (N), recall efficiency and pattern corrections [167, 168]. But pseudo inverse rule is neither local nor incremental as compared to the Hebbian rule. These problems can be solved by modifying the rule in such a way that some characteristics of Hebbian learning are also incorporated such that locality and incrementality is ensured. Therefore the weight matrix is first calculated using Hebbian rule and then the pseudo inverse of the weight matrix is calculated. This method may overcome the locality and incrementality problems associated with the pseudo inverse rule or projection learning.

As we have used standard Hopfield model which uses a very simple weight matrix formed with one-shot Hebbian learning that produces a network with relatively poor capacity and performance. In the second phase of experiment, we have employed genetic algorithm on Hopfield Neural Network’s to obtain such optimal weight matrices so that recalling of memorized patterns corresponding to the presented noisy prototype input patterns could be improve. The objective of this study is to determine the optimal weight matrix for correct recalling of static images.

The short comings of conventional Hopfield feedback neural network architecture were minimizing with the use of evolutionary algorithms mainly simulated annealing and genetic algorithms. It has been analyzed [169] that the genetic algorithm with Hopfield Neural Network architecture is exhibiting more efficient and optimal solution in the form of weight matrices. Thus genetic algorithms with Hopfield Neural Network that is hybrid evolutionary algorithms is most suitable
approach for obtaining the global optimum solution of weight matrices for efficient pattern recalling process. In this hybrid evolutionary approach the genetic algorithms whereas a neural network is based on models of human cognition. One common application of the genetic algorithm is as a function optimizer and another common application of the genetic algorithm is for evolving organisms that perform well in a given environment. In either application, the genetic algorithm is based on the survival-of-the-fittest (natural selection) tenet of Darwinian evolution.

Genetic algorithms and neural networks can be integrated into a single application to take advantage of the best features of these technologies [170]. Much work has been done on the evolution of neural networks with GA [171-175]. There have been a lot of researches which apply evolutionary techniques to layered neural networks. However, their applications to fully connected neural networks remain few so far. The first attempt to conjugate evolutionary algorithms with Hopfield neural networks dealt with training of connection weights. Evolution has been introduced in neural networks at three levels: architectures, connection weights and learning rules [176]. The evolution of connection weights proceeds at the lowest level on the fastest time scale in an environment determined by architecture, a learning rule, and learning tasks. The evolution of connection weights introduces an adaptive and global approach to training, especially in the reinforcement learning and recurrent network learning paradigm. Various researches were reported that the pattern recalling efficiency can improve by amalgam of genetic algorithms with Hopfield Neural Network. The combined approach i.e. hybrid evolutionary algorithms proves the superiority over the conventional feedback type neural network [177]. This research work is effort for improving the recalling efficiency of both the noiseless and noisy
patterns by introducing Self Organizing Map (SOM) as the feature extraction mechanism and present it to the Hopfield Neural Network with Hebbian learning, pseudo inverse learning and hybrid learning rule.

In this chapter, we have two phases of experiments. In the first phase of experiment, we are working with three methods of feature extraction, pattern storage, storing of pattern in Hopfield neural network via Edge dilation method (ED), Fast Fourier Transformation (FFT) and Self Organizing Map (SOM). Here we are passing all the images features in original form and their corresponding distorted form up to 10%, 20%, 30%, 40% and 50% error for the input image to analyze the performance of the network. In this analysis if regression value R is greater than 60% then we consider that the performance is acceptable. The simulated results of the implementation are suggests that up to 40% of error the performance of Hopfield neural network for pattern recall is better if feature are extracted from FFT method.

In this second phase of experiment, we have applied genetic algorithm on Hopfield's neural network to obtain weight matrices for efficient recalling of input prototype patterns. For this purpose we consider the scanned static images of English alphabets as input stimuli which we have preprocessed and filtered using Edge dilation method & FFT method. These processed inputs which we have received from FFT method are presented as pattern information to Self Organizing Map (SOM). Here the processed image patterns are input to SOM of fixed dimension 10×10. The code words generated from the SOM are input into the Hopfield Neural Network to test the performance.
This process of pattern encoding creates a sub optimal parent weight matrix. Therefore to improve the efficiency of Hopfield Neural Network the genetic algorithm is applied on this sub optimal weight matrix to obtain global optimum solution. Thus in this case the genetic algorithm starts from sub optimal solution instead of initial random population. In order to apply genetic algorithm, we explore the population generation technique, the crossover operator and the fitness evaluation function in order to generate the optimal weight matrix. The experiments consist of neural network training with multiple numbers of patterns using the Hebbian learning rule. In most cases, the recalling of patterns using a genetic algorithm seems to give better results than the conventional recalling with the Hebbian rule. The results suggest that the genetic algorithm is the better searching technique for recalling noisy prototype input patterns.

It is observed from the simulation result that the capability of the Hopfield Neural Network can be sufficiently enhanced by making modification of the input patterns. Edge Dilation and SOM method of feature extraction with genetic algorithms together can be used to significantly improve the recall efficiency by exploring optimal weight matrix as a global stable state in basin of attraction for the presented prototype input pattern.

4.2 Feature Extraction:

The feature extraction techniques are used to extract unique feature from the images to construct the pattern information. The pattern set used for the current study and analysis are scanned images of hand written character of English alphabets. These scanned RGB images scaled down to dimension 30 x 30 which are not of perfect
quality so there is a requirement for enhancement via various methods to reveal the fine details of the images which may remain uncovered due to light impact of pen or writing flow of individual. So these images are preprocessed before converting them to suitable patterns information for further processing. The images are first scanned and then converted to Gray scale to retain the fine details in the images. After that the images are enhanced and made sharper using the operation as edging, dilation, resizing, reshaping etc. The image is finally converted into a binary image by thresholding.

The feature extraction techniques extract unique features from the images [178]. The efficiency of the adopted feature extraction method decides to a great extent the quality of the image for further processing as well as for efficiency of pattern storage network. Therefore, the following feature extraction algorithms have been considered for extracting the useful features from the images to construct the pattern information for their storage and recalling.

The efficiency of the adopted feature extraction method decides to a great extent the quality of the image for further processing. Therefore we are employing Edge Dilation, FFT and SOM techniques for feature extraction to construct the pattern information.

4.2.1 Edge Dilation:

Edge detection is a type of image segmentation techniques, which determines the presence of an edge or line in an image and outlines them in an appropriate way [179]. The main purpose of edge detection is to simplify the image data in order to
minimize the amount of data to be processed [180]. Generally, an edge is defined as the boundary pixels that connect two separate regions with changing image amplitude and attributes such as different constant luminance and stimulus values in an image [179-181]. The detection operation begins with the examination of the local discontinuity at each pixel element in an image. Amplitude, orientation, and location of a particular subarea in the image that is of interest are essentially important characteristics of possible edges [179]. Based on these characteristics, the detector has to decide whether each of the examined pixels is an edge or not. Frei and Chen [179] suggest that edge detection is best carried out by simple edge detector, followed by a morphological thinning and linking process to optimize the boundaries.

There are different ways for edge detection methods out of which first and second order derivative edge detections, edge fitting detection model are used to extract the features [183]. In these methods we are evaluating the gradients generated along two orthogonal directions. An edge is judged present if the gradient of the image exceeds our defined threshold value, \( t = T \). The gradient can be computed as the derivatives along both orthogonal axes as

\[
G(x, y) = \frac{\partial F(x, y)}{\partial x} \cos \theta + \frac{\partial F(x, y)}{\partial y} \sin \theta
\]  

(4.1)

The gradient is estimated in a direction normal to the edge gradient. The spatial average gradient can be written as

\[
G(j, k) = \sqrt{[G_R(j, k)]^2 + [G_C(j, k)]^2}
\]  

(4.2)
A simplest discrete row and column gradient is given by

\[ G_R(j, k) = F(j, k) - F(j, k-1) \]  \hspace{1cm} (4.3)

\[ G_C(j, k) = F(j, k) - F(j+1, k) \]  \hspace{1cm} (4.4)

The figure 4.1 depicts the different steps for preprocessing task in the Edge dilation method are given as:

![Figure 4.1: a. Colored Image, b. Gray Image, c. Gray thresh image, d. Edge of image, e. Dilation of image.](image)

4.2.2 Fast Fourier Transform:

Images are mathematically represented as a function of spatial variable \( f(x, y) \). The values of variables \( x \) and \( y \) at a particular location represent the intensity of the image at that point. This is the called the Spatial Domain representation of the image. An alternative representation of the same image can be through the representation of its frequency, phase or other complex exponentials. This is referred to as the Frequency Domain representation.

Transforms are the mathematical representation of the Frequency Domain of the images. Transforms may be used for image enhancement, feature extraction,
Experiments & Simulation

compression etc. Fast Fourier Transform (FFT) is a form of Fourier Transform whose input and output are discrete samples. The FFT is usually defined for a discrete function \( f(x, y) \) that is nonzero only over the finite region \( 0 \leq x \leq X-1 \) and \( 0 \leq y \leq Y-1 \). The two-dimensional \( X \)-by-\( Y \) FFT and inverse \( X \)-by-\( Y \) FFT relationships are respectively given as [184]

\[
f(p, q) = \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} f(x, y) e^{-j2\pi px/X} e^{-j2\pi qy/Y} \quad p=0,1,...,X-1
\]

\[
q=0,1,...,Y-1 \quad 4.5
\]

and

\[
f(x, y) = \frac{1}{XY} \sum_{p=0}^{X-1} \sum_{q=0}^{Y-1} F(p, q) e^{j2\pi px/X} e^{j2\pi qy/Y} \quad x=0,1,...,X-1
\]

\[
y=0,1...,Y-1 \quad 4.6
\]

The values \( F(p, q) \) are the FFT coefficients of \( f(x, y) \). We apply FFT on our binarized images producing the FFT transform. The filtered transform is then subjected to Inverse transform to produce the refined images. Consequent to the filtering the transformation produced the refined images. Figure 4.2 represents refined and filtered images are obtained by Inverse of FFT. The modified image after the FFT filtering of the binarized crisper image obtained after FFT Filtering is depicted in the images.
The image obtained after FFT or ED filtering is converted to bipolar pattern with each pixel having value +1 or -1. This bipolar pattern information is now to pass in the Hopfield neural network for the pattern storage. The bipolar pattern is represented as a vector of dimension $900 \times 1$. Thus, the pattern vector $x_i$ can be represented in general form as $X_i = [x_{i1}, x_{i2}, x_{i3}, \ldots, x_{in}]^T$ where $n = 1$ to $900$.

All the image pattern vectors $x_i$ is presented in the sample patterns of the training set of the $N \times L$ as

$$P_{fft} = \begin{bmatrix}
  x_{11} & x_{21} & x_{23} & \ldots & x_{L1} \\
  x_{21} & x_{22} & x_{23} & \ldots & x_{L2} \\
  \ldots & \ldots & \ldots & \ldots & \ldots \\
  x_{N1} & x_{N2} & \ldots & \ldots & x_{LN}
\end{bmatrix}_{N \times L}$$

where $l = 1$ to $L$ is the number of patterns to be stored in the network during learning.
4.2.3 Self-Organizing Maps (SOM):

Self-organizing feature map is an unsupervised neural network works by detecting regularities and patterns in the input data and creates an organized description of data. They convert input patterns of arbitrary dimensions into one or two dimensional array of neurons. Each input is connected to all output neurons. Attached to every neuron there is a weight vector with the same dimensionality as the input vectors [184].

In the SOM variety of competitive type learning occurs through adaptation of the weights connecting the input layer to the array of neurons [185]. SOMs differ from pure competitive learning in the way that neighboring neurons in the self-organizing map learn to recognize neighboring sections of the input space. Thus, self-organizing maps learn both the distribution and the topology of the input vectors. The neurons are connected to adjacent neurons by a neighborhood relation, which dictates the topology and structure of the map. The weight change for adoption in SOM [186] can be given as:

\[ c = \arg \max_{m_i \in \mathbb{N}} m \quad i \quad n \quad || x(t) - m_i(t) ||, \quad i = 1, 2, 3, \ldots N \quad 4.8 \]

\[ m_i(t + 1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)] \]

\[ where \ h_{ci}(t) = \alpha(t)e^{-||} \quad 4.9 \]
The importance of the neighborhood lies in the fact that weight adjustment is done only for the neurons that lie in the neighborhood of the winning neuron. Further the size of the neighborhood shrinks as training progresses, thus localizing the area of maximum activity. This continues until the weights connecting the input data to the neurons have stabilized. As a result of this collective and co-operative learning, the network is tuned to create localized responses to input vectors and thus reflects the topological ordering of the input vectors. This ordering reflects the feature space for the pattern set P.

Here the SOM is applied with the training set of the sample images of size 900×26. The SOM network determines the feature vector for the sample images of given training set and represent these in the form of codebook. The codebook for the sample image A can represent in table 4.1 as below:

<table>
<thead>
<tr>
<th>1</th>
<th>0.310500</th>
<th>21</th>
<th>0.001500</th>
<th>41</th>
<th>0.619172</th>
<th>61</th>
<th>0.976030</th>
<th>81</th>
<th>0.941527</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.100340</td>
<td>22</td>
<td>0.075677</td>
<td>42</td>
<td>0.552665</td>
<td>62</td>
<td>0.942141</td>
<td>82</td>
<td>0.870619</td>
</tr>
<tr>
<td>3</td>
<td>0.040699</td>
<td>23</td>
<td>0.177372</td>
<td>43</td>
<td>0.055859</td>
<td>63</td>
<td>0.749206</td>
<td>83</td>
<td>0.774945</td>
</tr>
<tr>
<td>4</td>
<td>0.230813</td>
<td>24</td>
<td>0.178270</td>
<td>44</td>
<td>0.120249</td>
<td>64</td>
<td>0.538967</td>
<td>84</td>
<td>0.674052</td>
</tr>
<tr>
<td>5</td>
<td>0.394127</td>
<td>25</td>
<td>0.262666</td>
<td>45</td>
<td>0.139737</td>
<td>65</td>
<td>0.306960</td>
<td>85</td>
<td>0.516828</td>
</tr>
<tr>
<td>6</td>
<td>0.342574</td>
<td>26</td>
<td>0.114236</td>
<td>46</td>
<td>0.338534</td>
<td>66</td>
<td>0.209280</td>
<td>86</td>
<td>0.622222</td>
</tr>
<tr>
<td>7</td>
<td>0.064621</td>
<td>27</td>
<td>0.046062</td>
<td>47</td>
<td>0.403131</td>
<td>67</td>
<td>0.254530</td>
<td>87</td>
<td>0.471788</td>
</tr>
<tr>
<td>8</td>
<td>0.298691</td>
<td>28</td>
<td>0.257849</td>
<td>48</td>
<td>0.418302</td>
<td>68</td>
<td>0.249240</td>
<td>88</td>
<td>0.675438</td>
</tr>
<tr>
<td>9</td>
<td>0.591981</td>
<td>29</td>
<td>0.469893</td>
<td>49</td>
<td>0.359507</td>
<td>69</td>
<td>0.265360</td>
<td>89</td>
<td>0.616722</td>
</tr>
<tr>
<td>10</td>
<td>0.765191</td>
<td>30</td>
<td>0.512730</td>
<td>50</td>
<td>0.230079</td>
<td>70</td>
<td>0.328904</td>
<td>90</td>
<td>0.794702</td>
</tr>
<tr>
<td>11</td>
<td>0.078979</td>
<td>31</td>
<td>0.234987</td>
<td>51</td>
<td>0.899097</td>
<td>71</td>
<td>0.975727</td>
<td>91</td>
<td>0.902841</td>
</tr>
</tbody>
</table>
The SOM grid for the training set obtained with FFT and ED method can show in figure 4.3 as:

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>0.103620</td>
<td>32</td>
<td>0.160365</td>
<td>52</td>
<td>0.860759</td>
<td>72</td>
<td>0.912833</td>
<td>92</td>
<td>0.842613</td>
</tr>
<tr>
<td>13</td>
<td>0.096119</td>
<td>33</td>
<td>0.135065</td>
<td>53</td>
<td>0.451427</td>
<td>73</td>
<td>0.799224</td>
<td>93</td>
<td>0.785367</td>
</tr>
<tr>
<td>14</td>
<td>0.227656</td>
<td>34</td>
<td>0.073921</td>
<td>54</td>
<td>0.394281</td>
<td>74</td>
<td>0.584037</td>
<td>94</td>
<td>0.740258</td>
</tr>
<tr>
<td>15</td>
<td>0.374373</td>
<td>35</td>
<td>0.054285</td>
<td>55</td>
<td>0.253382</td>
<td>75</td>
<td>0.345836</td>
<td>95</td>
<td>0.701112</td>
</tr>
<tr>
<td>16</td>
<td>0.305124</td>
<td>36</td>
<td>0.193047</td>
<td>56</td>
<td>0.294198</td>
<td>76</td>
<td>0.326570</td>
<td>96</td>
<td>0.783982</td>
</tr>
<tr>
<td>17</td>
<td>0.070651</td>
<td>37</td>
<td>0.184768</td>
<td>57</td>
<td>0.401922</td>
<td>77</td>
<td>0.212025</td>
<td>97</td>
<td>0.759133</td>
</tr>
<tr>
<td>18</td>
<td>0.254669</td>
<td>38</td>
<td>0.384770</td>
<td>58</td>
<td>0.333223</td>
<td>78</td>
<td>0.365051</td>
<td>98</td>
<td>0.853855</td>
</tr>
<tr>
<td>19</td>
<td>0.586054</td>
<td>39</td>
<td>0.398697</td>
<td>59</td>
<td>0.277607</td>
<td>79</td>
<td>0.371870</td>
<td>99</td>
<td>0.832107</td>
</tr>
<tr>
<td>20</td>
<td>0.681811</td>
<td>40</td>
<td>0.373976</td>
<td>60</td>
<td>0.180579</td>
<td>80</td>
<td>0.577359</td>
<td>100</td>
<td>0.893763</td>
</tr>
</tbody>
</table>

Table 4.1: Codebook of sample image A.

Figure 4.3: SOM grid for FFT and ED method features.
4.3 Hopfield Neural Network:

The proposed Hopfield Model to store the images of hand written 26 alphabets of English language as input patterns each of which is of order 90×1 consists of 900 processing units and 900×900 connection strengths. The state of the processing unit is considered bipolar with symmetric connection strength between the processing units. Each neuron can be in one of the two stable states i.e. ±1. The Hybrid learning rule is used to encode the pattern information from the training set.

In this proposed method we are using a hybrid learning rule which is the correlation of Hebbian rule and projection learning rule. These rules can be discussed in the following sub sections.

4.3.1: Hebbian Rule:

In general the Hebbian Rule is used to store L patterns is given by the summation of correlation matrices for each pattern as:

\[
W_{ij} = \frac{1}{N} \sum_{l=1}^{L} \sum_{i,j} x_{li} * x_{lj} \quad \text{for } i \neq j
\]

\[
= 0 \quad \text{for } i=j, \ 1 \leq i \leq N \quad \text{4.10}
\]

where, \( N \) is the number of units/neurons in the network, \( x_{li} \) for \( l = 1 \) to \( L \) are the patterns / images to be stored, where each component of \( x \) is bipolar. For storing \( L \) patterns there should be one stable state corresponding to each stored pattern. Thus the following activation dynamics equation must be satisfied to accomplish the
storage. The units receive input from every other unit except for itself. The net input of a unit \( i \) at any time \( t \) is computed as:

\[
s_i(t) = \sum_{j \neq i} w_{ij} s_j(t)
\]

where \( w_{ij} \) is the weight of the connection between units \( i \) and \( j \) and \( s_j \) is the state of unit \( j \) at time \( t \).

Consider the initial weights \( w_{ij} \approx 0 \) prior to learning between processing units \( i \) and \( j \) where \( i, j = 1 \) to \( N \). The change in weight to store the 1st pattern can be considered as:

\[
w_{ij}^{\text{new}} = w_{ij}^{\text{old}} + \sum_{i, j} x_{ii} x_{ij}
\]

And

\[
w_{ij}^{\text{new}} = w_{ij}^{\text{old}}
\]

Similarly for the \( L \)-th pattern

\[
w_{ij}^{L} = w_{ij}^{L-1} + \sum_{i, j} x_{ii} x_{ij}
\]

This can be generalized as

\[
w_{ij}^{L} = \sum_{l=1}^{L} \sum_{i, j} x_{ii} x_{ij}
\]

The weight matrix thus obtained is normalized over all \( N \). Hence the normalized weight matrix is given by

\[
w_{ij}^{L} = \frac{1}{N} \sum_{l=1}^{L} x_{ii} x_{ij}
\]
The same weight matrix can be computed from pattern vectors as below

\[ w_{ij}^l = \frac{1}{N} \sum_{i=1}^{l} x_i \cdot (x_i)' \]  

4.17

Hence, in order to store 26 letters of English alphabet (all capitals) in a 900 unit bipolar Hopfield neural network, there should be one stable state corresponding to each stored pattern. Thus, the following activation dynamics equation must be satisfied to accomplish the storage [187]:

\[ f(\sum_{j \neq i}^{N} w_{ij} a_j) = a_i; \quad \text{where } (i, j=1,2,\ldots,N). \]

Let the pattern set be

\[
p = \begin{bmatrix}
    a_1^1 & a_2^1 & a_3^1 & \ldots & a_N^1 \\
    a_1^2 & a_2^2 & a_3^2 & \ldots & a_N^2 \\
    \vdots & \vdots & \vdots & \ddots & \vdots \\
    a_1^L & a_2^L & a_3^L & \ldots & a_N^L
\end{bmatrix}_{L \times N}
\]

4.18

Where

\[
a^1 = (a_1^1, a_2^1, \ldots, a_N^1)^T,
\]

\[
a^2 = (a_1^2, a_2^2, \ldots, a_N^2)^T,
\]

\[ \ldots \]

\[ \ldots \]

\[
a^L = (a_1^L, a_2^L, \ldots, a_N^L)^T,
\]

4.19

Where N=900 and L = 26].

-(101)-
From the synaptic dynamics as vectors we have the following equations for encoding the patterns information with initialization of weight vector by zero:

\[ W = P^T P \]  \hspace{1cm} 4.20

Thus, after the learning for all the patterns, the final parent weight matrix can be represented as [187]:

\[
W = \begin{bmatrix}
0 & \frac{1}{N} \sum_{l=1}^{L} a_1^l a_2^l & \frac{1}{N} \sum_{l=1}^{L} a_1^l a_3^l & \ldots & \frac{1}{N} \sum_{l=1}^{L} a_1^l a_N^l \\
\frac{1}{N} \sum_{l=1}^{L} a_2^l a_1^l & 0 & \frac{1}{N} \sum_{l=1}^{L} a_2^l a_3^l & \ldots & \frac{1}{N} \sum_{l=1}^{L} a_2^l a_N^l \\
\frac{1}{N} \sum_{l=1}^{L} a_3^l a_1^l & \frac{1}{N} \sum_{l=1}^{L} a_3^l a_2^l & 0 & \ldots & \frac{1}{N} \sum_{l=1}^{L} a_3^l a_N^l \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\frac{1}{N} \sum_{l=1}^{L} a_N^l a_1^l & \frac{1}{N} \sum_{l=1}^{L} a_N^l a_2^l & \frac{1}{N} \sum_{l=1}^{L} a_N^l a_3^l & \ldots & 0
\end{bmatrix}_{N \times N}  \hspace{1cm} 4.21

Now, to represent \( W \) in the convenient form, let us assume the notation \( s_i \) to express the state of \( i^{th} \) unit at stability. So that, states of the units at fixed point stability is expressed as:

\[
s_1 s_2 = \sum_{l=1}^{L} a_1^l a_2^l, s_1 s_3 = \sum_{l=1}^{L} a_1^l a_3^l, \ldots s_1 s_N = \sum_{l=1}^{L} a_1^l a_N^l \\
\]

\[
s_2 s_1 = \sum_{l=1}^{L} a_2^l a_1^l, s_2 s_3 = \sum_{l=1}^{L} a_2^l a_3^l, \ldots s_2 s_N = \sum_{l=1}^{L} a_2^l a_N^l \\
\]

\[
\vdots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \\
\]

\[
s_N s_1 = \sum_{l=1}^{L} a_N^l a_1^l, s_N s_3 = \sum_{l=1}^{L} a_N^l a_3^l, \ldots s_N s_N = \sum_{l=1}^{L} a_N^l a_N^l
\]  \hspace{1cm} 4.22
Therefore, from Equations (4.21) and (4.22), we get [2]:

\[
W = \frac{1}{N} \left[ \begin{array}{cccc}
0 & s_1 s_2 & s_1 s_3 & \cdots & s_1 s_N \\
s_2 s_1 & 0 & s_2 s_3 & \cdots & s_2 s_N \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
s_N s_1 & s_N s_2 & s_N s_3 & \cdots & 0 \\
\end{array} \right]_{L \times N}
\]

4.3.2 Pseudo Inverse Rule:

The pseudo inverse rule or projection rule to store L patterns is given by

\[
w_{ij}^+ = \frac{1}{N} \sum_{l=1}^{L} \sum_{i,j=1}^{n} x_i^l (w_i^{-1}) x_j^l
\]

As in vector form

\[
w^I = \frac{1}{N} \sum_{l=1}^{L} P_L \ast w_l^+
\]

Where \( w_l^+ = \frac{1}{N} (P_L + P_L^T)^{-1} + P_L^T \)

\[
W_L = \left[ \begin{array}{cccc}
x_{11} & x_{12} & x_{13} & \cdots & x_{1l} \\
x_{21} & x_{22} & x_{23} & \cdots & x_{2l} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
x_{N1} & x_{N2} & \cdots & \cdots & x_{NL} \\
\end{array} \right]
\]

-(103)-
4.3.3 Hybrid Learning Rule:

The proposed Hybrid learning rule to encode the pattern information from the given training set, consist with corelation of hebbian and projection learning rule. The Hebbian learning as specific in equation 4.16 is local and incremental but has a low absolute storage capacity of $\frac{n}{2\log n}$. This capacity decreases significantaly if pattern are correlated. Its performance is poor in term of storage capacity, attraction and spurious memories [188] where as the projection learning rule or pseudo inverse rule as specify in equation 4.24 exhibits more accuracy and storage capacity from hebbian correlation rule but it is not local or increemental, because it involves the calculation of an inverse. The proposed hybrid correlation learning rule overcomes from the disadvantages as loss the learning rule.

The hybrid correlation learning rule can specifies for storing the L patterns as

$$w_{ij}^{hl} = \frac{1}{N} \sum_{l=1}^{L} [w_{ij}^l + w_{ij}^{tl}]$$

4.27

Where $w_{ij}^l = \sum_{ij} x_{li} * x_{lj}$ and

$$w_{ij}^{tl} = [\sum_i \sum_j ((x_{li} * x_{lj})^T (x_{li} * x_{lj})^{-1}) * (x_{li} * x_{lj})^T]$$

4.28

Thus we have

$$w_{ij}^{hl} = \frac{1}{N} \sum_{l=1}^{L} [\sum_{i,j} x_{li} * x_{lj} + \sum_{i,j} ((x_{li} * x_{lj})^T (x_{li} * x_{lj})^{-1}) (x_{li} * x_{lj})^T]$$

4.29
Or in the vector form we can write

\[ w_{ij}^{ll} = \frac{1}{N} \sum_{l=1}^{L} \left[ P_l \cdot P_l^T + (P_l \cdot P_l^T)^{-1} \cdot P_l^T \right] \]

Once the training pattern set \( P \) has been stored in the Hybrid learning neural network using hybrid learning rule i.e. Hebbian and pseudo inverse Hebbian learning rule, it is required that the performance of the network is analyzed for the stored patterns, their noisy variants and also for incomplete pattern information. Therefore the process of recalling is considered, whereby a prototype test pattern, which can be any already stored pattern or its noisy form, inputs into the network and the network is allowed to evolve through its activation dynamics. The output state of the network is then tested for resemblance with one of the expected stable states.

Let a memorized pattern \( X \) and its noisy or distorted form is \( X + \epsilon \) where \( \epsilon \) is the included error in terms of the percentage (number) of bits distorted i.e. 10\% (90) bits, 20\% (180) bits, 30\% (270) bits, 40\% (360) bits and 50\% (450) bits from the size of a single input pattern.

Let the state of the network corresponding to the stored \( l^{th} \) pattern is:

\[ N(s_l) = \{ s_{1l}, s_{2l}, \ldots, s_{Nl} \} \]

Hence to recall the stored image, the prototype pattern \( X \) and its noisy form \( X + \epsilon \) are presented to the network. The activation dynamics of the network produces the output state for \( X \) and \( X + \epsilon \) respectively as:

\[ -(105)- \]
The activation dynamics in equation 4.32 for the memorized pattern X and in equation 4.33 for the distorted form of X is executed for testing. The recall efficiency with Hybrid learning using the weight matrix obtained in (4.30) is

\[(s_i^l) = \sum_{j=1}^{N} w_{ij}^k(s_j^l)(t + 1)\]  \hspace{1cm} 4.32

\[(s_i^l + \epsilon) = \sum_{j=1}^{N} w_{ij}^k(s_j^l + \epsilon)(t + 1)\]  \hspace{1cm} 4.33

4.4 Simulation Design:

In this simulation design and implementation, three experiments are designed for the proposed Hopfield neural network consists with 900 processing units. These pattern storage networks are trained to encode the pattern information with proposed hybrid learning rule for three different methods of feature extraction i.e. Edge &

In these experiments, we are analyzing the performance of Hopfield neural network for pattern storage and recalling. Each experiment is using the same type of neural network structure used same learning rule but inputs the different pattern information. Thus each experiment is using different features information for the training due to different feature extraction methods. Therefore the performance of each Hopfield neural network varies. The network which provides more efficient feature information, the performance of neural network will more effective. The simulation designs for these experiments are considered for analysis this performance evaluation. The simulation results are obtained after the several trials and the best result were considered for analysis.

4.4.1 Implementation of ED, FFT & SOM:

The simulation design is started with scanning of the hand written English alphabets as shown in figure 4.4.

![Original 26 images of hand written English alphabets.](image)

Figure 4.4: Original 26 images of hand written English alphabets.
Now we applied three methods of feature extraction i.e. ED, FFT, SOM on these images to obtain the pattern information. After applying these methods the images are presented as pattern vector for the pattern storage network. The preprocessed images have been shown in figure 4.1 & 4.2 for Edge Dilation and FFT method respectively.

After getting the extracted data from these static images, some noisy data was also created by adding 10%, 20%, 30%, 40% and 50% random error in the existing data set. There are 26 static images of hand written English alphabets. Apart from this, 5 sets of 26 images with extracted features were also prepared after inserting the noise in the existing images. There were total 156 images available for recalling process from all the three Hopfield networks.

Images for Edge Dilation Method from original to distorted from 10% to 50%.
Table 4.2A: 26 English Alphabets A to Z from original to distorted with noise from 10% to 50% for ED method.
<table>
<thead>
<tr>
<th>Original Image</th>
<th>10% Error Image</th>
<th>20% Error Image</th>
<th>30% Error Image</th>
<th>40% Error Image</th>
<th>50% Error Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>J</td>
<td>J</td>
<td>J</td>
<td>J</td>
<td>J</td>
<td>J</td>
</tr>
<tr>
<td>K</td>
<td>K</td>
<td>K</td>
<td>K</td>
<td>K</td>
<td>K</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>
Table 4.2B: 26 English Alphabets A to Z from original to distorted with noise from 10% to 50% for FFT method.

The next experiment is conducted for feature extraction with SOM and mapped into the Hopfield neural network. In this simulation the input images are preprocessed with FFT and ED and compressed using SOM before storing them into the Pattern storage network. The training set of L patterns is feed through a sequential algorithm into the self-organizing map of dimension 10 × 10.
These sample examples are presented with in the form of SOM grid as shown in figure 4.5a and 4.5b.

Figure 4.5 a: SOM Grid networks for FFT preprocessed features.
Figure 4.5b: SOM Grid networks for ED preprocessed features.
The performance of SOM can consider with following issues.

1. **Quality measure of SOM**: Although some optimal map always exists for the input data, choosing the right parameters from start is a tricky task. Since different parameters and initializations gives rise to different maps, it is important to know whether the map has properly adapted itself to the training data. This is usually done with a cost function that explicitly defines the optimal solution but since it has been shown that the SOM algorithms not the gradient of any cost function, at least in the general case, other quality measures has to be used. Two commonly used quality measures that can be used to determine the quality of the map and helping in choosing suitable learning parameters and map sizes are the average quantization error and the topographic error.

2. **Average quantization error**: Average quantization error is as a measure of how good the map can fit the input data and the best map is expected to yield the smallest average quantization error between the BMNs $m_c$ and the input vectors $x$. The mean of $||x_i - m_c||$ defined via inputting the training data once again after learning, is used to calculate as:

$$E_q = \frac{1}{N} \sum_{i=1}^{N} ||x_i - m_c||$$

where $N$ is the number of input vectors used to train the map. A SOM with a lower average error is more accurate than a SOM with higher average error.

3. **Topographic error**: Topographic error measures how well the topology is preserved by the map. Unlike the average quantization error, it considers the
structure of the map. For each input vector, the distance of the BMN and the second BMN on the map is considered; if the nodes are not neighbors, then the topology is not preserved. This error is can computed with the following method:

\[
E_t = \frac{1}{N} \sum_{k=1}^{N} u(x_k)
\]

where \(N\) is the number of input vectors used to train the map and \(u(x_k)\) is 1 if the first and second BMN of \(x_k\) are not direct neighbors of each other. Otherwise \(u(x_k)\) is 0.

The SOM method is constructing the feature map for the pattern. The input to the SOM is the input stimulus which is preprocessed from ED and FFT method of feature extraction. The final quantization and topographic error with both the methods from SOM network is represented in table 4.3 as:

<table>
<thead>
<tr>
<th></th>
<th>ED Method</th>
<th>FFT Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Final Quantization</td>
<td>Final Topographic</td>
</tr>
<tr>
<td></td>
<td>Error</td>
<td>Error</td>
</tr>
<tr>
<td>Original Data</td>
<td>2.276</td>
<td>0.038</td>
</tr>
<tr>
<td>10% Error Data</td>
<td>1.703</td>
<td>0.052</td>
</tr>
<tr>
<td>20% Error Data</td>
<td>1.996</td>
<td>0.054</td>
</tr>
<tr>
<td>30% Error Data</td>
<td>2.180</td>
<td>0.048</td>
</tr>
<tr>
<td>40% Error Data</td>
<td>2.260</td>
<td>0.041</td>
</tr>
<tr>
<td>50% Error Data</td>
<td>2.303</td>
<td>0.042</td>
</tr>
</tbody>
</table>

Table 4.3: Summary of final quantization and topographic error for SOM method.
4.4.2 Implementation of GA:

In our second phase of experiment, the Hebbian learning rule is used to store the pattern in the Hopfield Neural Network and the genetic algorithms is used for recalling the patterns. Thus we start the process of recalling for any presented any noisy prototype patterns of static original images. The process of recalling is started with sub optimal weight matrix obtained from the network.

The square matrix as shown in equation (4.23) is known as parent or suboptimal weight matrix for storing the given input patterns because it represents the storing of input patterns. Hopfield suggested that the maximum limit for the storage is 0.15N in a network with N neurons, if a small error in recalling is allowed. Later, this was theoretically calculated as 0.14N by using replica method [189]. Wasserman [190] showed that the maximum number of memories ‘m’ that can be stored in a network of ‘n’ neurons and recalled exactly is less that \( cn^2 \) where ‘c’ is a positive constant greater than one. It has been observed that the possibility of false minima may occur during the recalling of memorized patterns. Kumar and Singh investigated [191] that the GA of the evolutionary algorithm is much suitable choice to reduce the effect of false minima from the Hopfield neural network during the recalling of memorized patterns.

The genetic algorithm considers this parent weight matrix given in equation (4.23) as its initial solution. In order to implement the genetic algorithm we consider a population of sub optimal solution which is modified through uniform random mutation, discrete cross over and fitness function. In genetic algorithms implementation we consider the cycle of generating the new population with better
individuals and restarting the search is repeated until an optimum solution is found. In this process one fitness evaluation functions has been used. The fitness function is evaluating the best matrices of the weights population on the basis of the settlement of network in the stable state corresponding to the stored pattern on the presentation of the already stored pattern as the input pattern. It indicates that the stable states of the network will be used for the evaluation of the weights population. Thus in the recalling process, stable state of the network corresponding to the stored pattern should be retained for the selected weight vector on the presentation of prototype input pattern.

In regard of the fitness functions, we wish to say that the fitness evaluation function determines the suitable weight matrices which are responsible to generate the correct recalling of the stored pattern for the input pattern that has been used in the training set. It means, at the first level of filtering only those weight matrices will be selected which provide the correct pattern association for the training pattern set. Thus, at this level we will not use any test pattern, which involves the noise in the original pattern. It represents only those weight matrices which exhibit the pattern association during the training of the network and should carry in the next generation of population.

The population which is filtered from fitness function is further used to generate new population with uniform cross over operation. Each newly generated population is also evaluated from same fitness criteria and passes for the next generation. The detailed genetic algorithm is described in following sub sections.
4.4.2.1 The Mutation Operator:

The mutation operator produces the population of N weight matrices or chromosomes of same order as the original parent matrix on applying it N times. Thus, each chromosome is having a fixed length of $N \times N$ genes or alleles. In this process of mutation, we select all non-zero genes, i.e., $s_i^r$, $s_j^r$ from parent chromosome, where $r$ is the position of the gene in the parent chromosome or weight matrix. We have calculated the different values for mutation (vm) like minimum (min), maximum (max), max + min, max-min, average of min & max, etc. of the all genes of the same parent chromosome and multiple arithmetic operation (op) like add, subtract, multiply are applied on the random selected genes $s_i^r$, $s_j^r$ with these such values (vm) as below.

For $i=0$ to $N$

For $j=0$ to $N$

If $(w_{ij}^{old} \neq 0)$ then

$$w_{ij}^{new} = (\sum_{i,j=0}^{N} w_{ij}^{old}) \text{(op)} \text{ vm}$$

Else

$$w_{ij}^{new} = w_{ij}^{old}$$

End if

where op is any operator like $\ast$, $/$, `-` and vm is value for mutation operation.

In this way we get n population of mutated weights like

$$W_{mn} = \{ w_{m1}, w_{m2}, w_{m3} \ldots w_{mn} \}$$

4.36

-(118)-
4.4.2.2 Crossover:

In the proposed methodology we divide all the N population matrices into quadrants of fix size $S \times S$. Each quadrant matrix will contain chromosomes. We can apply crossover in two forms. First one is the Local Crossover and the Second is Global Crossover.

In the local crossover operation, the uniform crossover will apply by exchanging the one quadrant with another one to produce a new population of weight matrix of size $m \times n$. In global crossover operation, we select randomly two populations of weight matrices from the population N. From these two selected weight matrices, we again randomly select any quadrant of order $s \times s$ from one matrix and exchange it with other randomly selected quadrant of another matrix with same size to produce new population of weight matrix of size $m \times n$.

$$w_{N \times N}^i = \begin{bmatrix} w_{11}^i & w_{12}^i \\ w_{21}^i & w_{22}^i \end{bmatrix} \text{ where } i= 1, 2, 3, \ldots, N$$

For local crossover:

$$w_{1,N \times N}^{new} = w_{s \times s}^{R1} \leftarrow w_{s \times s}^{R2} \text{ where } R1, R2 \text{ are any random selected quadrant from same weight matrix.}$$

$$w_{2,N \times N}^{new} = w_{s \times s}^{R1} \leftarrow w_{s \times s}^{R2}$$

$$\ldots$$

$$w_{N,N \times N}^{new} = w_{s \times s}^{R1} \leftarrow w_{s \times s}^{R2}$$

-(119)-
Here $W_{1,N×N}^{new}$, $W_{2,N×N}^{new}$ and $W_{N,N×N}^{new}$ are newly generated weight matrices after local crossover operator, $W_{s×s}^{R1}$ and $W_{s×s}^{R2}$ are two sub weight matrices selected randomly from old mutated population to perform the local crossover operation for exchange. Now $W_{newL}$ is

For global crossover

$$W_{1,N×N}^{new} = W_{s×s}^{iR_1R_2} \leftarrow W_{s×s}^{jR_1R_2} \text{ for } i \neq j \text{ and } i, j = 1, 2, 3, ... N$$

$$W_{2,N×N}^{new} = W_{s×s}^{iR_1R_2} \leftarrow W_{s×s}^{jR_1R_2}$$

..........................

..........................

$$W_{n,N×N}^{new} = W_{s×s}^{iR_1R_2} \leftarrow W_{s×s}^{jR_1R_2}$$

where $W_{s×s}^{iR_1R_2}$, $W_{s×s}^{jR_1R_2}$ are two any random selected quadrant from different weight matrices.

So using this global crossover, we will find a new weight matrix like as below

$$W_{N×N}^{newG} = \{W_{s×s}^{1R_1R_2}, W_{s×s}^{2R_1R_2}, ...... .... W_{s×s}^{nR_1R_2} \}$$

After $m$ times of crossover operation

-(120)-
Where \( w_n \) are mutations after N+1 time,

\[
\begin{align*}
\mathbf{W}_{\text{local}} = & \bigcup_{n=1}^{N} \mathbf{W}_n \bigcup_{k=1}^{m} \mathbf{W}_{k,N\times N}^{\text{newL}} \bigcup_{t=1}^{P} \mathbf{W}_{t,N\times N}^{\text{newG}} \\
\end{align*}
\]

\[
\begin{align*}
\mathbf{W}_{\text{newL}} = \{ \mathbf{W}_{1,N\times N}^{\text{new}}, \mathbf{W}_{2,N\times N}^{\text{new}}, \mathbf{W}_{3,N\times N}^{\text{new}}, \ldots, \mathbf{W}_{m,N\times N}^{\text{new}} \} \\
\end{align*}
\]

\[
\begin{align*}
\mathbf{W}_{k,N\times N}^{\text{new}} = & \mathcal{R}_i (\mathbf{W}_{S\times S}^k) \iff \mathcal{R}_j (\mathbf{W}_{S\times S}^k) \\
\end{align*}
\]

where \( i \neq j \) and \( \mathcal{R}_i, \mathcal{R}_j \) are random selected matrices of order \( S\times S \).

**4.4.2.3 Fitness Function:**

The next step of our sub optimal genetic algorithm processing is to determine an efficient fitness function for selecting the better next generation of weight matrices as new population. The fitness function (f) is supposed to evaluate with individual population of weight matrices carried in out during mutation and crossover operation. The fitness function (f) is constructed for whole pattern samples \( P \) (shown in equation no. 4.18) for each weight matrix. Each weight matrix is assigned to the network for the given pattern set or prototype pattern set (which contains the erroneous sample pattern) and the recalled pattern set are obtained. The performance of the network for this recalling is measured in form of the regression value between desired pattern which is supposed to be recalled and actual pattern which is recalled.
The population of weight matrices evaluated those will consider for next generation evaluation of the fitness evaluation function is

\[ f_1(w) = \text{Avg}[R_{w_1}, R_{w_2}, \ldots R_{w_m}, R_{w_{m1}}, R_{w_{m2}}, \ldots R_{w_{mn}}] \]  

Where \( R_{w_1}, R_{w_2}, \ldots R_{w_m} \) are the generated weight matrices from local and global crossover and \( w_{m1}, w_{m2}, \ldots w_{mn} \) are the population of weight matrices after the mutation.

The fitness evaluation function \( f(w) \) is defined as:

\[ f(w) = \begin{cases} 
1 \text{ if } R_{w_i} > f_1(w) & \forall i = 1 \text{ to } 2m \text{ and } 2m + 1 \text{ to } mn. \\
0 \text{ if } R_{w_i} \leq f_1(w) & \forall i = 1 \text{ to } 2m \text{ and } 2m + 1 \text{ to } mn. 
\end{cases} \]

The weight populations for which the network performance for pattern recalling is less than or equal to average of regression value are discarded and do not continue for the further iteration of genetic algorithm in next population. Thus, the weight matrices are selected for the next population contains the fitness value \( f(w) = 1 \).

### 4.5 Result and Discussion:

The simulated results are obtained from already defined three experiments which we have done in first phase. In the result of first and second experiment of first phase, we consider the feature extraction from Edge Dilation and FFT method respectively to store pattern information of static images in the feed forward neural network of 900 processing units. In this experiment, we are analyzing the
performance evaluation of Hopfield neural network with our proposed hybrid learning rule. The recalling efficiency of Hopfield neural network is evaluated and analyzed for the prototype test pattern consist with 10%, 20%, 30%, 40%, and 50% noises in the actual sample images respectively.

Table 4.4 shows the images which are result of recalling using Edge Dilation Method for input noisy patterns from 10%, 20%, 30%, 40%, and 50%.

<table>
<thead>
<tr>
<th>Original</th>
<th>Output on 10% noisy Input</th>
<th>Output on 20% noisy Input</th>
<th>Output on 30% noisy Input</th>
<th>Output on 40% noisy Input</th>
<th>Output on 50% noisy Input</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Image" /></td>
<td><img src="image2" alt="Image" /></td>
<td><img src="image3" alt="Image" /></td>
<td><img src="image4" alt="Image" /></td>
<td><img src="image5" alt="Image" /></td>
<td><img src="image6" alt="Image" /></td>
</tr>
</tbody>
</table>
Table 4.4: Result of recalling using Edge Dilation Method from 10%, 20%, 30%, 40%, and 50%.

Table 4.5 shows the images which are result of recalling using Edge Dilation Method for input noisy patterns from 10%, 20%, 30%, 40%, and 50%.

<table>
<thead>
<tr>
<th>Original</th>
<th>Output on 10% noisy Input</th>
<th>Output on 20% noisy Input</th>
<th>Output on 30% noisy Input</th>
<th>Output on 40% noisy Input</th>
<th>Output on 50% noisy Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
</tbody>
</table>
The simulated result for recalling performance of tuned Hopfield neural network for actual sample of image and distorted or noisy sample of images can visualize from figure 4.6 and 4.8 respectively which represent the regression lines to exhibit the accuracy of recalling. The regression value is 1 for the actual images and this value starts decreasing as the percentage of noise or error increases in the original static images as shown in figure 4.6 and 4.8 respectively. This performance evaluation can also see from table 4.6, 4.7 and figure 4.7, 4.9 respectively for both experiment i.e. ED and FFT method.

The result can be summarized for ED method in table 4.6 as:

<table>
<thead>
<tr>
<th>Error in features</th>
<th>Original image</th>
<th>10 % Noisy Input</th>
<th>20 % Noisy Input</th>
<th>30 % Noisy Input</th>
<th>40 % Noisy Input</th>
<th>50 % Noisy Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression Value R</td>
<td>1.00000</td>
<td>0.95882</td>
<td>0.84518</td>
<td>0.63231</td>
<td>0.3948</td>
<td>0.3948</td>
</tr>
</tbody>
</table>

Table 4.6: Performance evaluation of Hopfield neural network for recalling from Edge Dilation method.
Figure 4.6: Regression line for a. Original image, b. 10 % error image, c. 20 % error image,
d. 30 % error image, e. 40 % error image, f. 50 % error image.
Figure 4.7: Regression value for Performance evaluation of Hopfield neural network for Recalling from Edge Dilation method.

The result can be summarized for FFT method as in below table 4.7 as:

<table>
<thead>
<tr>
<th>Error in features</th>
<th>Original image</th>
<th>10 % Noisy Input</th>
<th>20 % Noisy Input</th>
<th>30 % Noisy Input</th>
<th>40 % Noisy Input</th>
<th>50 % Noisy Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression Value R</td>
<td>1.00000</td>
<td>0.9678</td>
<td>0.8768</td>
<td>0.69351</td>
<td>0.4615</td>
<td>0.4615</td>
</tr>
</tbody>
</table>

Table 4.7: Performance evaluation of Hopfield neural network for recalling from FFT method.
Figure 4.8: Regression line for a. Original image, b. 10% error image, c. 20% error image, d. 30% error image, e. 40% error image, f. 50% error image.
In the result of third experiment of first phase, we consider the feature extraction from SOM method to store pattern information of static images in the feed forward neural network of 900 processing units. In this experiment the input stimuli for SOM network is presented in preprocessed form. This preprocessing is performed by ED and FFT methods. The preprocessed static images are presented to SOM network of 10 × 10 neighboring region. The SOM network produces a codebook for each method and grid map as shown in figure 4.5 and 4.6. These codebooks are used as pattern information for Hopfield Neural Network. The performance of recalling for the feature map of original static images and erroneous images is presented in figure 4.10 for ED preprocessed features and in figure 4.11 for FFT preprocessed features respectively.
Figure 4.10: Regression line preprocessed from ED method. a. for original image, b. for 10 % error image, c. for 20 % error image, d. for 30 % error image, e. for 40 % error image, f. for 50 % error image.
Figure 4.11: Regression line preprocessed from FFT method. a. for original image, b. for 10 % error image, c. for 20 % error image, d. for 30 % error image, e. for 40 % error image, f. for 50 % error image.
The comparative performance of pattern recalling for Original static images and erogenous images from SOM network for preprocessed input stimuli by ED and FFT methods can represent in table 4.8 and the same can visualize from figure 4.12.

<table>
<thead>
<tr>
<th>Methodology used</th>
<th>Original image</th>
<th>10% Noisy Input</th>
<th>20% Noisy Input</th>
<th>30% Noisy Input</th>
<th>40% Noisy Input</th>
<th>50% Noisy Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>R for SOM via FFT</td>
<td>1.00000</td>
<td>0.12590</td>
<td>-0.012768</td>
<td>0.17778</td>
<td>0.097176</td>
<td>0.097176</td>
</tr>
<tr>
<td>R for SOM via ED</td>
<td>1.00000</td>
<td>0.18536</td>
<td>0.15055</td>
<td>0.10368</td>
<td>0.062725</td>
<td>0.062725</td>
</tr>
</tbody>
</table>

Table 4.8: Performance evaluation of Hopfield neural network for recalling from SOM method for ED and FFT feature.

Figure 4.12: Regression value for Performance evaluation of Hopfield neural network for recalling of SOM with ED and FFT feature method.
The results of second phase are also indicating the difference in the recalling successes of Hebbian rule and sub optimal GA. The simulation results, presented in this section are demonstrating that, within the simulation framework presented above, some significant difference exists between the performance of genetic algorithm and the Hebbian rule for recalling of static images of English alphabets which have been memorized in Hopfield neural network using the Hebbian learning rule.

In this second experiments, the prototype input patterns used for recalling purpose consists the 10%, 20%, 30%, 40%, 50% error which has been generated randomly with respect to memorized patterns which are shown in tables 4.2B. Figure 4.13 shows the results of recalling as regression value for the patterns which containing 10% error with Hebbian rule and sub optimal GA as regression value. In the same way, figure 4.14, 4.15, 4.16 & 4.17 shows the 20%, 30%, 40%, 50% errors recalling as regression value for stored patterns in the Hopfield neural network. The results clearly indicate that the Hebbian rule works well for a noiseless pattern, for most of the cases, but its performance degrades substantially and recalling success degrades as noise increases in input patterns. It is also observed that in most of the cases, the performance of the suboptimal GA outperform in comparison to Hebbian rule. As we can see for 30% error improvement is approximately 2% while in 10%, 20% and 50% it is only approximately 1%. In 40% error case, there is not a significant difference.

Amit [187] claimed that the capacity of deterministic Hopfield model with the Hebbian rule is about 0.15N for the noisy prototype input patterns, where N is the number of nodes in the network. If such a network is overloaded with a number of
patterns exceeding its capacity, its performance rapidly deteriorates toward zero. Here, we are storing the 26 static images of English alphabets in a network of 900 nodes and the performance of the GA suggests that on inducing 10%, 20%, 30%, 40%, 50% error in presented prototype input pattern the network is able to recall the stored patterns for both the Hebbian rule as well as sub optimal GAs.

It implies that the network capacity has increased. Thus, the numbers of attractors are existing here and successfully explored during the recalling process. It is quite obvious to understand that the GA has searched the suitable optimal weight matrices which are responsible to generate sufficiently large number of attractions. Hence, the Hebbian rule which has been used to encode the pattern information is not the optimal weight matrix for finding the global minima of the problem due to the limited capacity of the Hopfield model. Thus capacity has been increased with GA by exploring the optimal weight matrices for the encoded patterns.

The simulation of program, which is developed in MATLAB-7(R2010), to test the Hebbian rule, and the suboptimal GA, for the recalling of static images of English alphabets, stores the patterns in the Hopfield neural network of 900 neurons. It is to note that during suboptimal GA, the success is considered only if the recalling is done within 20-iterations.
Recalling on 10 % Error input using FFT
\[ R = 0.96780 \]
Recalling on 10 % Error input after GA
\[ R = 0.97533 \]

Figure 4.13: Comparison of Recalling using Hebbian Rule and GA for 10 % Error pattern.

Recalling on 20 % Error input using FFT
\[ R = 0.8768 \]
Recalling on 20 % Error input after GA
\[ R = 0.88575 \]

Figure 4.14: Comparison of Recalling using Hebbian Rule and GA for 20 % Error pattern.
Recalling on 30 % Error input using FFT

\[ R = 0.69351 \]

Recalling on 30 % Error input after GA

\[ R = 0.71774 \]

Figure 4.15: Comparison of Recalling using Hebbian Rule and GA for 30 % Error pattern.

Recalling on 40 % Error input using FFT

\[ R = 0.46150 \]

Recalling on 40 % Error input after GA

\[ R = 0.46532 \]

Figure 4.16: Comparison of Recalling using Hebbian Rule and GA for 40 % Error pattern.
Recalling on 50 % Error input using FFT  
\[ R = 0.1433 \]

Recalling on 50 % Error input after GA  
\[ R = 0.15303 \]

Figure 4.17: Comparison of Recalling using Hebbian Rule and GA for 50 % Error pattern.

<table>
<thead>
<tr>
<th>Noisy Input</th>
<th>On 10%</th>
<th>On 20%</th>
<th>On 30%</th>
<th>On 40%</th>
<th>On 50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hebbian Rule</td>
<td>FFT</td>
<td>GA</td>
<td>FFT</td>
<td>GA</td>
<td>FFT</td>
</tr>
<tr>
<td>10% Noisy</td>
<td>.96780</td>
<td>.97533</td>
<td>.87680</td>
<td>.88575</td>
<td>.69350</td>
</tr>
<tr>
<td>20% Noisy</td>
<td>.69350</td>
<td>.71774</td>
<td>.46150</td>
<td>.46532</td>
<td>.14330</td>
</tr>
<tr>
<td>30% Noisy</td>
<td>.46150</td>
<td>.46532</td>
<td>.14330</td>
<td>.15303</td>
<td>.15303</td>
</tr>
<tr>
<td>% Diff</td>
<td>1%</td>
<td>1%</td>
<td>2%</td>
<td>0%</td>
<td>1%</td>
</tr>
</tbody>
</table>

Table 4.9: Summary of Performance evaluation of Hopfield neural network for recalling from FFT and GA method.
Figure 4.18: Comparison of FFT and GA for recalling of static images.

4.6 Conclusion:

In this chapter, the performance analysis of pattern storage networks has been evaluated for efficient storage and recalling for the presented prototype patterns. Experiments are conducted with three feature extraction methods i.e. ED, FFT and SOM. The hybrid learning rule has been also proposed which has characteristic of the standard Pseudo inverse rules and Hebbian rule such as locality and incremental learning.

We have pattern storage neural network architecture of Hopfield type with 900 processing units to reflect the associative memory feature for storage and recalling of static images. The performance of SOM for preprocessed stimuli with ED exhibit more accuracy in recalling in comparison to FFT. The performance of pure FFT to
make feature extraction for pattern storage and recalling is found best over the other two methods i.e. ED and SOM.

It is being observed that the considered network is very large and the numbers of patterns are very low. Therefore the network traps in false minima during the recalling process for presented prototype input patterns. The problem of false minima is minimized by applying genetic algorithm to the network for recalling purpose. The proposed genetic algorithm starts from the weight matrix which has been constructed by the Hebbian rule. Therefore the genetic algorithm is started from suboptimal solution rather than random solution.

It has been observed from simulation results for stored pattern with Hebbian rule, genetic algorithm perhaps better for recalling of presented input prototype patterns with noise from 10%, 20%, 30%, 40% and 50%. It is also being observed that the efficiency of genetic algorithm depends on the criteria of our fitness evaluation function. We have applied genetic algorithms for optimization of Pattern storage network whose aim is to enhance the pattern association and recall efficiency of Hopfield Neural Network in such a way that false minima could eliminated. The results from the experiment are done are quite encouraging. But still there is a need of further research in various scopes and dimensions to apply such concepts in pattern recognition of different types of objects, images etc.