Chapter 1

Introduction to Soft Computing & Image Processing
Abstract:

This chapter presents an overview of the biological network, Hopfield neural network, Image extraction methods and genetic algorithm used in my research work. This chapter contains various models, learning methods of neural network and the detailed discussion about the architecture and types of neural networks. It also contains different models of neurons with their types, learning & recalling (generalization) capability of the neural network, various supervised/un-supervised learning algorithms and applications of the Artificial Neural Networks. A brief introduction about images and various steps in its processing is also given. In last, the problem definition, objectives and motivation of this research work along with the organization of thesis is presented.
1.1 Introduction:

Soft computing, an innovative approach to construct computationally intelligent systems, has recently gained popularity and wide spread use. It is being realized that complex real world problems require intelligent systems that combine knowledge, techniques and methodologies from various sources. These intelligent systems are supposed to possess human-like expertise within a specific domain and the ability to adapt and learn in changing environments. To achieve this complex goal, a single computing paradigm or solution is not sufficient.

Soft computing is a wide ranging term encompassing such varied techniques as Fuzzy Systems, Rough Sets, Neural Networks, Genetic Algorithms, Simulated Annealing, DNA Computing, Quantum Computing, Membrane Computing etc. While some of these techniques are still in the nascent stage, the rest of them have found wide spread use in the area of Pattern recognition, Classification, Image Processing, Voice Recognition, Data Mining etc. Each of these methodologies have their own strength. The seamless integration of these methodologies to create intelligent systems, forms the core of soft computing.

Soft computing is an approach to computing which parallels the remarkable ability of the human mind to reason and learn in an environment of uncertainty and imprecision. In an attempt to find out reasonably useful solutions, soft computing-based methods [1, 2] acknowledge the presence of imprecision and uncertainty present in machining. Soft computing techniques such as Fuzzy Logic, Neural Network, Genetic Algorithms, Simulated Annealing, and Ant Colony Optimization
have received a lot of attention of researchers due to their potentials to deal with highly nonlinear, multidimensional, and ill-behaved complex engineering problems.

1.2 Application of Soft Computing:

- **Automobile applications:** Ford Motor Co., General Motors, and other automobile manufacturers are currently researching the possibility of widespread use of neural networks in automobiles and in automobile production.

- **Biomedical applications:** Neural networks are rapidly finding diverse applications in the biomedical sciences. They are being used widely in research on amino acid sequencing in RNA and DNA, ECG and EEG waveform classification, prediction of patient’s reactions to drug treatments, prevention of anesthesia-related accidents, and classification of medical images, lung nodule detection, and diagnosis of hepatic masses and the study of interstitial lung disease.

- **Control of sound and vibration:** Active control of vibration and noise is accomplished by using an adaptive actuator to generate equal and opposite vibration and noise. This is being used in air-conditioning systems, in automotive systems, and in industrial applications.

- **Cursive handwriting recognition:** Neural networks have proved useful in the development of algorithms for on-line cursive handwriting recognition [3]. A recent startup company in Palo Alto, Lexicus beginning with this basic technology has developed an impressive PC based cursive handwriting system.
- Financial forecasting and portfolio management: Neural networks are used for financial forecasting at a large number of investment firms and financial entities including Merill Lynch & Co., Salomon Brothers, Shearson Lehman Brothers Inc., Citibank, and the World Bank.

- Hand-printed character recognition: Hecht-Nielsen Corp’s Quickstrokes Automated Data Entry System is being used to recognize handwritten forms at Avon’s order-processing center and at the state of Wyoming’s Department of revenue.

- Machine-printed character recognition: Commercial products performing machine-printed character recognition have been introduced by a large number of companies and have been described in the literature.

- Petroleum exploration: Oil companies including Arco and Texaco are using neural networks to help determine the locations of underground oil and gas deposits.

- Quality control in manufacturing: Neural fuzzy networks are being used in a large number of quality control and quality assurance programs throughout industry. Applications include contaminant-level detection from spectroscopy data at chemical plants and loudspeaker defect classification by CTS Electronics.

- Speech recognition: The Stanford Research Institute is currently involved in research combining neural networks with hidden Markov models and other technologies in a highly successful speaker independent speech recognition system. The technology will most likely be licensed to interested companies once perfected.
1.3 Neural Network: Biological & Artificial Paradigm:

The human brain is a source of natural intelligence and a truly remarkable parallel computer. The brain is made up of vast network of neurons, which are coupled with receptors and effectors. The brain processes incomplete information obtained by perception at an incredibly rapid rate. Nerve cells function about 10^6 times slower than electronic circuit gates, but human brain process visual and auditory information much faster than modern computers.

The brain learns from experience and acquires knowledge. It is natural proof that some problems that are beyond the scope of current computers are indeed solvable by small energy efficient packages. This brain modeling also promises a less technical way to develop machine solutions. This new approach to computing also provides a more graceful degradation during system overload than its more traditional counterparts.

1.3.1 Biological Neuron:

Basically, a biological neuron receives inputs from other sources, combines them in some way, performs a generally nonlinear operation on the result, and then outputs the final result. Figure 1.1 shows the relationship of these four parts.

Within humans there are many variations on basic type of neuron, yet, all biological neurons have the same four basic components. They are known by their biological names – cell body (soma), dendrites, axon, and synapses.
- Cell body (Soma): The body of neuron cell contains the nucleus and carries out biochemical transformation necessary to the life of neurons.

- Dendrite: Each neuron has fine, hair like tubular structures (extensions) around it. They branch out into tree around the cell body. They accept incoming signals.

- Axon: It is a long, thin, tubular structure which works like a transmission line.

- Synapse: Neurons are connected to one another in complex spatial arrangement.

When axon reaches its final destination it branches again called as terminal arborization. At the end of axon are highly complex and specialized structures called synapses. Connection between two neurons takes place at these synapses.

Dendrites receive the input through the synapses of other neurons. The soma processes these incoming signals over time and converts that processed value into an output, which is sent out to other neurons through the axon and the synapses.
1.3.2 Artificial Neuron:

Artificial neuron is a basic building block of every artificial neural network. Its design and functionalities are derived from observation of a biological neuron that is basic building block of biological neural networks (systems) which includes the brain, spinal cord and peripheral ganglia. Similarities in design and functionalities can be seen in Figure 1.2, which represents an artificial neuron with its inputs, weights, transfer function, bias and outputs.

![Figure 1.2: Artificial Neuron](image)

In case of biological neuron information comes into the neuron via dendrite, soma processes the information and passes it on via axon. In case of artificial neuron the information comes into the body of an artificial neuron via inputs that are weighted (each input can be individually multiplied with a weight). The body of an artificial neuron then sums the weighted inputs, bias and “processes” the sum with a transfer function. At the end an artificial neuron passes the processed information via output(s).
1.3.3 Characteristics of ANN:

The overall functioning of neural networks is divided into two stages: **learning** (training) and **generalization** (recalling). Network training is done by giving the input samples, and network parameters are adapted using a learning method. This can be done in an online or offline manner. Once the network is trained to accomplish the desired performance, the learning process is terminated. For real-time applications, a neural network is required to have a constant processing delay regardless of the number of input nodes, and a minimum number of layers. As the number of input nodes increases, the size of the network layers should grow at the same rate without additional layers.

The Artificial neural networks are characterized by the network architecture, node characteristics, and learning rules. Neural networks are usually biologically motivated. Each neuron is a computational node, which represents a nonlinear function. Neural networks possess the following advantages:

Neural networks have strong learning capability. They can adapt themselves by changing the network parameters in a surrounding environment. Powerful learning algorithms make this capability possible. Learning and generalization are the most salient features of neural networks. A well-trained neural network has superior generalization capability. This can be attributed to the bounded and smooth nature of the hidden-unit responses. The bounded-unit response localizes the nonlinear effects of the individual hidden units in a neural network and allows for the approximations in different regions of the input space to be independently tuned.
[4]. In contrast, in conventional curve-fitting methods, the polynomials and other functions have a potential divergence nature.

Some neural networks such as the SOM [5] and competitive learning based neural networks have a self-organization property. The training of these networks is based on the unsupervised learning algorithms.

Neural networks have robustness and fault-tolerant capability. A neural network can easily handle imprecise, fuzzy, noisy, and probabilistic information. It is a distributed information system, where information is stored in the whole network in a distributed manner by the network structure. Some neural networks such as the Hopfield network can be used as associative storage of information. When a noisy or incomplete pattern is presented to a trained network, it will help to find the correct pattern; that is, the trained network is fault tolerant.

1.4 Neural Network Models:

Most of the models of the neural networks are well known out of which some of them are given below for their formal description in order to understand their functionality and application for neural networks.

1.4.1 Feed Forward Neural Network Architecture:

This was the first and arguable simplest type of artificial neural network devised. Artificial neural network with feed-forward topology is called Feed-Forward artificial neural network and as such has only one condition: information must flow from input to output in only one direction with no back-loops. There are
no limitations on number of layers, type of transfer function used in individual artificial neuron or number of connections between individual artificial neurons. The simplest feed-forward artificial neural network is a single perceptron that is only capable of learning linear separable problems.

1.4.2 Feed Back Neural Network Architecture:

In this type of neural network, there are connections from output to input neurons; such a network keeps a memory of its previous states, and the next state depends not only on the input signal but also on the previous state of the network. In this the information moves in the backward direction. We can see this type of neural network in Hopfield network.

1.4.3 Auto-associative Neural Network Architecture:

In this architecture, all units are interconnected with each other. Signals can pass both ways on every connection. There is no distinction between input and output unit because there is no explicit, external feedback or reinforcement. These networks are sometime called unsupervised. Technically it means that, they can be used to implement unsupervised learning where the network improves its performance over time without an explicit teaching mechanism. Some of these types of system are also called self-organizing systems. The Hopfield network is an auto associative type of network.
1.4.4 Hetero-associative Neural Network Architecture:

The hetero-associative neural network (HANN) associates a spatial pattern with another pattern which may or may not be the same as the input pattern. The BAM, multilayer perceptron network (MLP), the Kohonen network, the counter propagation network etc. are examples of the hetero-associative architecture.

1.4.5 Neural Network as Associative Memories:

Pattern association is the process of memorizing input-output patterns in a hetero-associative network architecture, or input patterns only in an auto-associative network, in order to recall the patterns when a prototype input pattern is presented. It is not required that the new input pattern be exactly the same as one that is memorized. Three exemplar patterns are shown in Figure 1.3. After memorizing them in a system, a new pattern is presented, a corrupted (noisy) variant of the pattern 3. An associative memory system should associate this pattern with one of the memorized patterns. This is a task for auto-associative network architecture.

Auto-associative neuronal architectures can be realized either by feedforward, or feedback networks. Hopfield networks and Boltzmann machines are two examples of auto associative networks. Other types of pattern associators are the hetero-associative network architectures. One of them is the bidirectional associative memory (BAM) [6].
Figure 1.3: The pattern association task, illustrated with three class patterns and a new one, corrupted, which has to be associated to one from the class patterns that is most similar.

1.4.6 Recurrent Neural Networks:

Artificial neural network with the recurrent topology is called Recurrent Artificial Neural Network. It is similar to feed-forward neural network with no limitations regarding back-loops. In these cases information is no longer transmitted only in one direction but it is also transmitted backwards. This creates an internal state of the network which allows it to exhibit dynamic temporal behavior. Recurrent artificial neural networks can use their internal memory to process any sequence of inputs. Figure 1.4 shows small Fully Recurrent artificial neural network and complexity of its artificial neuron interconnections.

Figure 1.4: Recurrent Artificial Neural Networks
The most basic topology of recurrent artificial neural network is fully recurrent artificial network where every basic building block (artificial neuron) is directly connected to every other basic building block in all direction. Other recurrent artificial neural networks such as Hopfield, Elman, Jordan, bi-directional and other networks are just special cases of recurrent artificial neural networks.

Recurrent neural networks have feedback connections from neurons in one layer to neurons of a previous layer. Different modified versions of such networks developed and explored. A typical recurrent network is that in which the nodes whose output values feedback as inputs to the network. So the next state of a network not only depends upon the connection weights and the currently presented input signals but also on the previous states of the network. The network leaves a trace of its behavior; the network keeps a memory of its previous states.

Depending on the architecture of the feedback connections, there are two general models of recurrent networks: (1) Partially recurrent, and (2) Fully recurrent.

1.4.6.1 Partially Recurrent Neural Networks:

The neural networks of the first type have carefully selected specific feedback connections which are meaningful. They represent (possibly time-) dependence between concepts represented in the network. Feedback connections to the input layer can be established from hidden nodes or output nodes.

The weights assigned to the feedback connections may not be adjustable. In this case a standard MLP training algorithm (e.g., the back propagation algorithm)
can be used to adjust the feed forward connection weights. In addition to the feedback connections, which make possible use of a previous-time moment output value for calculating the next one, a buffer of output values for more than one step back in time can be used as an additional input buffer. Such recurrent networks are called buffered networks [7, 8]. It was shown that a recurrent network which keeps a track of its k-previous states can be represented as unfolded MLP with k layers of connections. Such networks were called back propagation through time [9, 10]. The idea is to duplicate the nodes in space in order to achieve time-dependence.

1.4.6.2 Fully Recurrent Neural Networks:

In fully recurrent networks any node may be connected to any other. This is the case with the Hopfield network.

Recurrent neural network can model a finite state automation. Difficulties arise to deal with, when using recurrent neural network, are

- Synchronization is required in order to achieve proper timing when propagating the signals through the network.
- It is difficult to express in a linguistic form or in a formula the time-dependence learned in a recurrent network after training, i.e. the balance which the network has achieved between forgetting previous states and remembering new ones.
- Recurrent networks may manifest chaotic behavior, and therefore learning might be difficult.
Recurrent networks suit time-series prediction problems, speech recognition problems, and many others, where in order to recognize current state previous states have to be considered, as in language understanding and speech processing the semantic meaning of a word is recognized after taking into account the previously recognized words and possibly some of the following words.

1.4.7 Stochastic Artificial Neural Network:

Stochastic artificial neural networks are a type of an artificial intelligence tool. They are built by introducing random variations into the network, either by giving the network's neurons stochastic transfer functions, or by giving them stochastic weights. This makes them useful tools for optimization problems, since the random fluctuations help it escape from local minima. Stochastic neural networks that are built by using stochastic transfer functions are often called Boltzmann machine.

1.5 Learning (Training) and Recalling (Generalization) of ANN:

Typically, any type of learning is little bit slow process. Almost everybody is familiar with this term learning or training. In any neural network, it is also a slow process, and the samples containing a pattern to train a network, may have to be presented several times before the pattern information is captured by the weights of the network. A large number of samples are normally needed for the network to learn the pattern implicit in the samples. Pattern information is distributed across all the weights, and it is difficult to relate the weights directly to the training samples.
After finalizing the architecture of the neural network for a given application, the training or learning is required for getting the desired output from the network. Training or learning of a neural network is an optimization process that produces an output that is as close as possible to the desired output by adjusting network parameters. This kind of parameter estimation is also called learning or training algorithm. Neural networks are usually trained by epoch. An epoch is a complete run when all the training examples are presented to the network and are processed using the learning algorithm only once. After learning, a neural network represents a complex relationship, and possesses the ability for generalization. When a new input is presented to the trained neural network, a reasonable output is produced.

Learning methods are conventionally divided into supervised, unsupervised, reinforcement, and evolutionary learning. Supervised learning is widely used in pattern recognition, approximation, control, modeling and identification, signal processing, and optimization. Reinforcement learning is usually used in control. Unsupervised learning schemes are mainly used for pattern recognition, clustering, vector quantization, signal coding, and data analysis. Evolutionary computation is a class of optimization techniques, which can be used to search for the global minima/maxima of an objective function. Evolutionary learning is used for adjusting neural network architecture and parameters using an evolutionary algorithm (EA), and can also be used to optimize the control parameters in a supervised or unsupervised learning algorithm.
1.5.1 Supervised Learning Methods:

Supervised learning is based on a direct comparison between the actual network output and the desired output. Network parameters (weights) are adjusted by a combination of the training pattern set and the corresponding errors between the desired output and the actual network response. The errors first calculated then propagated back through the system, causing the system to adjust the weights, which evolve the learning process. The pattern set, which enables the learning, is called the "training set." During the learning of a network the same set of data is processed many times as the connection weights are ever refined. So supervised learning can be defined as a closed-loop feedback system, where the error is the feedback signal. The trained network is used to emulate the system. To control a learning process, a criterion is needed to decide the time for terminating the process. For supervised learning, an error measure, which shows the difference between the network output and the output from the training samples, is used to guide the learning process. The error measure is usually defined by the mean squared error and calculated by the error function:

$$E = \frac{1}{N} \sum_{p=1}^{N} [z_p - \hat{z}_p]^2$$  \hspace{1cm} (1.1)

Where the $N$ is the total number of patterns pair from a sample training set, $z_p$ is the actual output and $\hat{z}_p$ is the output calculated by the network for $p^{th}$ pair of sample of training set. This function is also known as the objective function to optimize the network. The error $E$ is calculated a new after each epoch. This process of network training is terminated when $E$ is sufficiently small or a failure criterion is met. To minimize the error up to the non-significant value, a gradient-descent procedure is usually applied. The LMS [11] and back propagation algorithms are two early, but
most popular, supervised learning algorithms. Both of them are derived using a gradient-descent procedure. When finally, the system has been correctly learned, and no further learning is needed, the weights can, if desired, be "frozen"[12]. In some systems, this finalized network is then turned into hardware so that it can be fast. Other systems don't lock themselves in but continue to learn while in production use.

1.5.2 Unsupervised Learning methods:

Unsupervised learning involves no target values. It tries to auto associate information from the inputs to decide what features it will use to group the input data. Unsupervised learning is solely based on the correlations among the input data, and is used to find the significant patterns or features in the input data without any supervision. A criterion is needed to terminate the learning process. Without a termination criterion, a continuous learning process continues even when a pattern, which does not belong to the training patterns set, is presented to the network. The network is adapted according to a constantly changing environment. Hebbian learning [13], competitive learning [14], and Kohonen’s SOM [15,16] are the three mostly used unsupervised learning approaches. In general the unsupervised learning is slow to settle into stable conditions. In Hebbian learning [13], learning is a purely local phenomenon, involving only two neurons and a synapse. The synaptic weight change is proportional to the correlation between the pre and postsynaptic signals. The C-means algorithm is a popular competitive learning-based clustering method [17]. By using the correlation of the input vectors, the learning rule changes the network weights to group the input vectors into clusters. The Boltzmann machine [18] uses a kind of stochastic training technique known as SA [19], which can been treated as a special type of unsupervised learning based on the inherent property of a
physical system. Tuevo Kohonen, an electrical engineer at the Helsinki University of Technology developed a self-organizing network [20], sometimes called an autoassociator that learns without the benefit of knowing the right answer. It is an unusual looking network in that it contains one single layer with many connections. The weights for those connections have to be initialized and the inputs have to be normalized. The neurons are set up to compete in a winner-take-all fashion. The other most common algorithm of unsupervised learning is the Hopfield neural network model [21, 22] of associative memory. The Hopfield neural network suggested a feedback system in which the used energy estimation function relates the network to other physical systems. Hopfield network is fully interconnected network with symmetric weights, no self-feedback and asynchronous update of the state of processing elements.

1.5.3 Reinforcement Learning in Neural Network:

Reinforcement learning [23] is a special case of supervised learning, where the exact desired output is unknown. It is based only on the information as to whether or not the actual output is close to the estimate. Explicit computation of derivatives is not required. This, however, presents a slower learning process. Reinforcement learning is a learning procedure that rewards the neural network for its good output result and punishes it for the bad output result. It is used in the case when the correct output for an input pattern is not available and there is need for developing a certain output. The evaluation of an output as good or bad depends on the specific problem and the environment. For a control system, if the controller still works properly after an input, the output is judged as good; otherwise, it is considered as bad. The evaluation of the output is binary, and is called external
reinforcement. Thus, reinforcement learning is a kind of supervised learning with the external reinforcement as the error signal. Reinforcement learning can learn the system structure by trial-and-error, and is suitable for online learning [24-27].

1.5.4 Evolutionary Learning in Neural Network:

Evolutionary learning approach is attractive since it can handle the global search problem better on a vast, complex, multimodal, and no differentiable surface. It is not dependent on the gradient information of the error (or fitness) function, and thus is particularly appealing when this information is unavailable or very costly to obtain or estimate. Evolutionary Algorithms can be used to search for the optimal control parameters in supervised as well as unsupervised learning by optimizing their respective objective functions. It can also be used as an independent training method for network parameters by optimizing the error function. Evolutionary Algorithms are widely used for training neural networks and tuning fuzzy systems, and are generally much less sensitive to the initial conditions. They always search for a globally optimal solution, while supervised and unsupervised learning algorithms can only find a local optimum in a neighborhood of the initial solution [28].

1.5.5 Learning Algorithms of ANN:

There are several methods of learning. Some of the well-known learning algorithms for the ANN are discussed as below.
1.5.5.1 Hebbian Learning:

The basis for the class of Hebbian learning is that the changes in the synaptic strength is proportional to the correlation between the firing of the post- and pre-synaptic neurons [29]. The synaptic dynamics equation is given by a decay term and a correlation term \(\{s_is_j\}\) as,

\[
w_{ij}(t) = -w_{ij} + s_is_j
\]  \hspace{1cm} (1.2)

where \(s_is_j\) is the product of the post-synaptic and pre-synaptic neuronal variables for the \(i^{th}\) unit. These variables could be activation values \((s_is_j = x_i(t) x_j(t))\), or an activation value and an external input \((s_is_j = x_i(t)a(t))\), or an output signal and an external input \((s_is_j = f_i(x_i(t))a(t))\), or output signals from two units \((s_is_j = f_i(x_i(t)) f_j(x_j(t)))\), or some other parameters related to the post-synaptic and pre-synaptic activity.

1.5.5.2 Competitive Learning:

Learning laws which modulate the difference between the output signal and the synaptic weight belong to the category of competitive learning. The general form of competitive learning is given by the following expression for the synaptic dynamics [30].

\[
w_{ij}(t) = s_i[s_j - w_{ij}(t)]
\]  \hspace{1cm} (1.3)

where, \(s_i = f_i(x_i(t))\) is the output signal of the unit \(i\), and \(s_j = f_i(x_j(t))\) is the output signal of the unit \(j\). This is also called the deterministic competitive learning law. It can be written as
1.5.5.3 Error Correction Learning:

Error correction learning uses the error between the desired output and the actual output for a given input pattern to adjust the weights. Rosenblatt's perceptron learning [31] uses the instantaneous misclassification error to adjust the weights is given by

\[ w_{ij}(t) = \eta (b_i - s_i) a_j \]  

(1.5)

where \( b_i \) is the desired output from the \( i \)th output unit for an input pattern \( a(n) = (a_1, a_2, ..., a_n) \), \( a_j \) is the \( j \)th component of the input pattern to the \( i \)th unit, \( s_i \) is the actual output of the \( i \)th unit given \( s_i = \text{sgn}(\sum_j w_{ij} a_j) \) and \( \eta \) is a small positive learning constant.

1.5.5.4 Stochastic Learning:

Stochastic learning involves adjustment of weights of a neural network in a probabilistic manner [32]. The adjustment uses a probability law, which in turn depends on the error. The error for a network is a positive scalar defined in terms of the external input, desired output and the weights connecting the units. In the learning process, a random weight change is made and the resulting change in the error is determined. If the resulting error is lower, then accept the random weight change. If the resulting error is not lower, then accept the random weight change with a pre decided probability distribution. The acceptance of random change of
weights despite increase in the error from the network allows the network to escape local minima in the search for the global minimum of the error surface.

1.5.5.5 Self Organizing Map (SOM):

As we have discussed, Artificial neural networks (ANNs) has been studied and used for information processing systems which is inspired from biological neural structures. They not only provide solutions with improved performance in comparison with traditional problem-solving methods, but it also gives a deep understanding of human cognitive abilities. Among various existing neural network architectures and learning algorithms, Kohonen’s self-organizing map (SOM) [33] is one of the most popular neural network models. Developed for an associative memory model, it is an unsupervised learning algorithm with a simple structure and computational form, and is motivated by the retina-cortex mapping.

Self-organization in general is a fundamental pattern recognition process, in which intrinsic inter- and intra-pattern relationships among the stimuli and responses are learnt without the presence of a potentially biased or subjective external influence. The SOM can provide topologically preserved mapping from input to output spaces. Although the computational form of the SOM is very simple, numerous researchers have already examined the algorithm and many of its problems, nevertheless research in this area goes deeper and deeper – there are still many aspects to be exploited.

Kohonen’s self-organizing map (SOM) is an abstract mathematical model of topographic mapping from the (visual) sensors to the cerebral cortex. Modeling and
analyzing the mapping are important to understanding how the brain perceives, encodes, recognizes and processes the patterns it receives and thus, if somewhat indirectly, are beneficial to machine-based pattern recognition. Kohonen's SOM is a neural network architecture which can be efficiently used for vector quantization.

The unsupervised learning feature of SOM gives user even more freedom of dimension reduction while preserving the topological property of input sequences. Therefore, it is one of most commonly used neural network approaches for image processing. Self-Organizing Maps (SOMs) are a data visualization technique invented by Professor Teuvo Kohonen which reduces the dimensions of data through the use of self-organizing neural networks.

Self- Organizing Feature Map is characterized as unsupervised learning algorithm, in which the output network neurons compete among themselves to be activated; in result only one output neuron or one neuron per layer wins the competition [34-36].

On the basis of the above discussion, below table provides a compact summary of various neural network models, their important characteristics and applications [37].
A summary of various neural network models, their important characteristics and applications

<table>
<thead>
<tr>
<th>Model</th>
<th>Learning Algorithm</th>
<th>Architecture</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perceptron</td>
<td>Supervised, error-correction</td>
<td>Single-node, feedforward</td>
<td>Pattern classification</td>
</tr>
<tr>
<td>Adaline</td>
<td>Supervised, gradient descent</td>
<td>Single-node, feedforward</td>
<td>Regression</td>
</tr>
<tr>
<td>Multilayer perceptron</td>
<td>Supervised, gradient descent</td>
<td>Multilayer, feedforward</td>
<td>Function approximation</td>
</tr>
<tr>
<td>Reinforcement learning</td>
<td>Supervised, reward-punishment</td>
<td>Multilayer</td>
<td>Control</td>
</tr>
<tr>
<td>Support vector machines</td>
<td>Supervised quadratic optimization</td>
<td>Multilayer kernel based</td>
<td>Classification, regression</td>
</tr>
<tr>
<td>Radial basis function net</td>
<td>Supervised, gradient descent</td>
<td>Multilayer distance based</td>
<td>Interpolation, classification, regression</td>
</tr>
<tr>
<td>Hopfield network</td>
<td>Outer product correlation</td>
<td>Single layer feedback</td>
<td>CAM, Optimization</td>
</tr>
<tr>
<td>Boltzmann machine</td>
<td>Stochastic gradient descent</td>
<td>Two layer, feedback</td>
<td>Optimization</td>
</tr>
<tr>
<td>BSB</td>
<td>Outer product correlation</td>
<td>Single layer feedback</td>
<td>Clustering</td>
</tr>
<tr>
<td>Bidirectional associative memory</td>
<td>Outer product correlation</td>
<td>Two layer, feedback</td>
<td>Associative memory</td>
</tr>
<tr>
<td>Adaptive resonance theory</td>
<td>Unsupervised competitive</td>
<td>Two layered</td>
<td>Clustering, optimization</td>
</tr>
<tr>
<td>Vector quantization</td>
<td>Supervised, unsupervised</td>
<td>Single layer feedback</td>
<td>Quantization, clustering</td>
</tr>
</tbody>
</table>
Before closing neural networks discussion, I summarize some important properties of neural networks that have made them popular for real world applications. Therefore, neural networks are well known for their

- Ability to approximate functions
- Automatic similarity based generalization
- Capability to perform associative recall of memory and fill-in incomplete information.
- Robustness, graceful degradation of performance, and fault tolerance.

### 1.6 Genetic Algorithm (GA):

The genetic algorithm (GA) is a randomized search and optimization technique guided by the principle of natural genetic systems. Genetic algorithms follow the process of natural evolution by incorporating the “survival of the fittest” philosophy [38]. Recently, there has been a great deal of interest in GAs and their application to various engineering fields. The GA is also being applied to a wide range of optimization and learning problems in many domains. Genetic algorithms

<table>
<thead>
<tr>
<th>Model</th>
<th>Characteristics</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mexican hat net</td>
<td>None fixed weights</td>
<td>Single layer, feedback</td>
</tr>
<tr>
<td>Kohonen SOFM</td>
<td>Unsupervised soft- competitive</td>
<td>Single layer</td>
</tr>
<tr>
<td>Pulsed neuron models</td>
<td>None</td>
<td>Single/multilayer layer</td>
</tr>
</tbody>
</table>

Table 1.1: A summary of various neural network models, their important characteristics and applications
also lend themselves well to power system optimization problems, since they are known to exhibit robustness, require no auxiliary information, and can offer significant advantages in solution methodologies and optimization performance. Genetic algorithms solve problems using principles inspired by natural population genetics. They maintain populations of knowledge structures that represent candidate solutions, and let those populations evolve over time through competition and controlled variation. The main advantage of the genetic algorithms formulation is that fairly accurate results may be obtained using a very simple algorithm.

The genetic algorithm maintains a set of possible solutions (population) represented as strings of, typically, binary numbers (0/1). New strings are produced in each and every generation by the repetition of a two-step cycle. This involves first decoding each individual string and assessing its ability to solve the problem. Each string is assigned fitness values depending on how well it has performed in an environment.

In the second stage, the fittest string is preferentially chosen for recombination, which involves the selection of two strings, and the switching of the segments to the right of the meeting point of the two strings. This is called crossover. Another genetic operator is mutation. It is used to maintain genetic diversity within a small population of strings. There is a small probability that any bit in a string will be flipped from its present value to its opposite (e.g., 0 to 1), this prevents certain hits from becoming fixed at a specific value due to every string in the population having the same value, often causing premature convergence to a non-optimal solution.
An additional common feature of the GA is the automatic inclusion of the best performing string of the parent generation in the new offspring generation. This procedure prevents a good string from being lost by the probabilistic nature of reproduction and speeds convergence to a good solution.

The genetic algorithm goes through the following cycle: Evaluate, Select and Mute, and Mutate until some kind of stopping criteria are reached. One criterion is to let the GA run for a certain number of cycles. A second one is to allow the genetic algorithms to run until a reasonable solution is found. Although GA seems to be a good method for solving optimization problems, sometimes the solution obtained is only a near global optimum solution.

The idea behind GAs is to do what nature does. Before we take a closer look at the structure of a genetic algorithm, let us have a quick look at the history of genetics. Genetic algorithms use vocabulary borrowed from natural genetics. We will talk about individuals (Or genotypes and structures) in a population; quite often these individuals are called strings or chromosomes. Chromosomes are made of units—genes (also features, characters, or decoders)—arranged in linear succession: every gene controls the inheritance of one or more characters.

The genes of certain characters are located at certain places in a chromosome, which are called loci (string positions). Any feature of an individual (such as hair color) can manifest itself in different ways—the gene is thus said to have several stun, called alleles (feature values).
Genetic algorithms differ from other optimization and search procedures in the following ways [39].

- Sometimes near optimal solutions that can be generated quickly, using GAs, are more desirable than optimal solutions which require a large amount of time.
- Using genetic algorithms problems can be modeled as optimization problems.
- GAs work with a coding of the parameter set, not the parameters themselves. Therefore, they can easily handle integral or discrete variables.
- GAs can provide globally optimal solutions.
- GAs use only objective function information, not derivatives or other auxiliary knowledge. Therefore they can deal with the non-smooth, non-continuous, and non-differentiable functions.
- GAs use probabilistic transition rules, not deterministic rules.

1.6.1: Genetic Algorithm Procedure:

Figure 1.5 shows the flow chart for GAs. A genetic algorithm (like any evolution) Particular problem must have the following five components:

1. A genetic representation for the potential solutions to the problem.
2. A way to create an initial population of potential solutions.
3. An evaluation function that plays the role of the environment, rating solutions in terms of their 'fitness'.
4. Genetic operators that alter the composition of the offspring.
5. Values for the various parameters that the genetic algorithm uses (population size, probabilities of applying genetic operators, etc.).

If the optimization problem is to minimize a function \( f \), this is equivalent to maximizing a function \( g \), where \( g = -f \), i.e.,

\[
\min \{f(x)\} = \max \{g(x)\} = \max \{-f(x)\} \quad (1.6)
\]

Moreover, we can assume that the objective function \( f \) takes only the positive values in its domain; otherwise we can add some positive constant \( C \) to \( g \), i.e.,

\[
\max \{g(x)\} = \max \{g(x) + C\} \quad (1.7)
\]

Now suppose we wish to maximize a function of \( k \) variables, \( f(x_1, x_2, ..., x_k) : \mathbb{R}^k \rightarrow \mathbb{R} \). Suppose further that each variable \( x_i \) can take values from a domain \( D_i = [a_i, b_i] \subseteq \mathbb{R} \) and \( f(x_1, x_2, ..., x_k) > 0 \) for all \( x_i \in D_i \). We wish to optimize the function \( f \) with some required precision; say, up to six decimal places for the variables. It is clear that to achieve such precision; say up to six decimal places for the variable. It is clear that to achieve such precision, each domain \( D_i \) must be cut into \((b_i - a_i) \times 10^6\) equal-sized ranges. Let \( m_i \) be the smallest integer such that \((b_i - a_i) \times 10^6 \leq 2^{m_i} - 1\).

Then, a representation having each variable \( x_i \) coded as a binary string of length \( m_i \) clearly satisfies the precision requirement. Additionally, the following formula interprets each such string:

\[
X_i = a_i + \text{decimal}(1001...001_2) \times (b_i - a_i) \quad (1.8)
\]

where \( \text{decimal(string)} \) represents the decimal value of that binary string.
Each chromosome (a potential solution) is represented by a binary string of length \( \sum_{i=1}^{k} m_i \). The first \( m_1 \) bits map onto a value in the range \([a_1, b_1]\), the next group of \( m_2 \) bits map onto a value in the range \([a_2, b_2]\) and so on; the last group of \( m_k \) bits map onto a value in the range \([a_k, b_k]\).

### 1.6.2 Genetic Algorithm Operators:

The genetic algorithms begin with a population of random strings representing design or decision variables. The population is then operated by three main operators; reproduction, crossover and mutation to create a new population of points. Genetic algorithms can be viewed as trying to maximize the fitness function, by evaluating several solution vectors. The purpose of these operators is to create new solution vectors by alteration, combination, or selection of the current solution vectors that have shown to be good temporary solutions. The new population is further evaluated and tested till termination. If the termination criterion is not met, the population is iteratively operated by the above three operators and evaluated. This procedure is continued until the termination criterion is met. One cycle of these operations and the subsequent evaluation procedure is known as a generation in genetic algorithms terminology. The operators are described in the following steps.

#### 1.6.2.1 Reproduction or Selection:

Reproduction generates a mating pool by selecting good fitness strings from the population. Although there are numerous reproduction operators reported in the literature, the essential idea in all of them is the same [41]:

-{32}-
Strings with fitness above-average are picked from the current population and strings with fitness below-average are removed from the population. This procedure may involve maintaining multiple copies of good strings in order to keep the population size fixed. The reproduction operator acts as a filtering mechanism for the selection of good strings in a population.

One commonly used reproduction operator is the *proportionate selection* operator. Here, the $i$th string in the current population is selected with a probability equal to the string's relative fitness $F_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$, where $f_i$ is the fitness of the $i$th
string, and N is the population size. The cumulative probability for all strings in the
population must necessarily add to one. Proportionate selection is commonly
implemented using a roulette-wheel with the circumference marked for each string
proportionate to the string's relative fitness. The roulette-wheel is spun N times, each
time keeping an instance of the string selected by the roulette-wheel pointer in the
mating pool. If f is the average fitness of the population, then such a roulette-wheel
mechanism can be expected to make copies of the ith string in the resulting
population. Alternatively genetic algorithms employ tournament selection for its
simplicity. In a typical tournament selection two strings are randomly chosen from
the population, and the fitter of the two is selected for insertion into the mating pool
[42].

1.6.2.2 Crossover:

The crossover operator is a core genetic operator; it is considered as an axle
for the generation of new offspring with strong exchange features of new genetic
information obtained as a result of the e individuals of the chromosomal population.
It can generally be classified into three genetic materials among the main categories:
conventional binary operators, arithmetical operators, and direction-based
operators [39, 40, 43, 44].

1.6.2.3 Mutation:

Mutation adds new information in a random way to the genetic search
process and ultimately helps to avoid getting trapped at local optima. It is an operator
that introduces diversity in the population whenever the population tends to become
homogeneous due to repeated use of reproduction and crossover operators.
The operator, mutation, is performed on a bit-by-bit basis. Another parameter of the genetic system, probability of mutation \( p_m \), gives us the expected number of mutated bits, \( p_m \times \text{pop}_\text{size} \). Every bit (in all chromosomes in the whole population) has an equal chance of undergoing mutation, i.e., changing from 0 to 1 or vice versa. So we proceed in the following way. For each chromosome in the current (i.e., after crossover) population and for each bit within the chromosome:

- Generate a random (float) number \( r \) from the range \([0, 1]\).
- If \( r < p_m \), mutate the bit.

Even though mutation is performed on a bit-by-bit basis, there are still different types and ways of implementing the mutation operator. Mutation is generally classified as (a) uniform, (b) boundary, and (c) non-uniform mutation [40].

1.6.3 Fitness Function:

Genetic algorithms mimic the survival-of-the-fittest principle [45] of nature to make a search process. Therefore, genetic algorithms are naturally suitable for solving maximization problems. Maximization problems are usually transformed into maximization problem by suitable transformation. In general, a fitness function \( f(i) \) is first derived from the objective function and used in successive genetic operations. Fitness in biological sense is a quality value which is a measure of the reproductive efficiency of chromosomes. In genetic algorithm, fitness is used to allocate reproductive traits to the individuals in the population and thus act as some measure of goodness to be maximized. This means that individuals with higher fitness value will have higher probability of being selected as candidates for further
examination. Certain genetic operators require that the fitness function be non-negative, although certain operators need not have this requirement. For maximization problems, the fitness function can be considered to be the same as the objective function or \( f(i) = O(i) \). For minimization problems, to generate non-negative values in all the cases and to reflect the relative fitness of individual string, it is necessary to map the underlying natural objective function to fitness function form. A number of such transformations is possible. Two commonly adopted fitness mapping is presented below.

\[
F(x) = \frac{1}{1 + f(x)} \tag{1.9}
\]

This transformation does not alter the location of the minimum, but converts a minimization problem to an equivalent maximization problem. An alternate function to transform the objective function to get the fitness value \( f(i) \) is as below.

\[
F(i) = V - \frac{O(i)^p}{\sum_{i=1}^{P} O(i)} \tag{1.10}
\]

where, \( O(i) \) is the objective function value of \( i^{th} \) individual, \( P \) is the population size and \( V \) is a large value to ensure non-negative fitness values. The value of \( V \) adopted in this work is the maximum value of the second term of equation 1.10 so that the fitness value corresponding to maximum value of the objective function is zero. This transformation also does not alter the location of the solution, but converts a minimization problem to an equivalent maximization problem. The fitness function value of a string is known as the string fitness.
1.7 Introduction to Image Processing:

Every human have evolved with very precise visual skills: we can identify a face in an instant; we can differentiate colors; we can process a large amount of visual information very quickly. However, the world is in constant motion: stare at something for long enough and it will change in some way. Even a large solid structure, like a building or a mountain, will change its appearance depending on the time of day (day or night); amount of sunlight (clear or cloudy), or various shadows falling upon it. We are concerned with single images: snapshots, if you like, of a visual scene.

Image processing can deal with changing scenes. An image is a single picture which represents something. It may be a picture of a person, of people or animals, or of an outdoor scene, or a microphotograph of an electronic component, or the result of medical imaging. Even if the picture is not immediately recognizable, it will not be just a random blur.

Image Processing systems are becoming popular due to easy availability of image data, powerful computational devices, high capacity memory devices, graphics software etc. Image Processing is used in various applications such as pattern classification, pattern recognition, image recalling, Remote Sensing, Medical Imaging, Non-destructive Evaluation, Forensic Studies, Textiles, Material Science, Military, Film industry, Document processing, Graphic arts and Printing Industry.
1.7.1 Types of Digital images:

We shall consider four basic types of images [46] which are as under.

1.7.1.1 Binary Image:

Each pixel is just black or white. Since there are only two possible values for each pixel, we only need one bit per pixel. Such images can therefore be very efficient in terms of storage. Images for which a binary representation may be suitable include text (printed or handwriting), fingerprints, or architectural plans. In the Figure 1.6, we have only the two colors: white for the edges, and black for the background.

![Binary Image](image)

Figure 1.6: A Binary image

1.7.1.2 Grayscale Image:

Each pixel is a shade of grey, normally from 0 (black) to 255 (white). This range means that each pixel can be represented by 8 bits, or exactly one byte. This is
a very natural range for image file handling. Other greyscale ranges are used, but generally they are a power of 2. Such images arise in medicine (X-rays), images of printed works, and indeed 256 different grey levels are sufficient for the recognition of most natural objects. An example is the street scene is shown in figure 1.7.

![Grayscale Image](image1.png)

**Figure 1.7: A grayscale image**

### 1.7.1.3 True color or RGB image:

Here in an image, each pixel has a particular color; that color being described by the amount of red, green and blue in it. If each of these components has a range 0-255, this gives a total of $(256)^3 = 16777216$ different possible colors in an image. These are enough colors for any image. Since the total number of bits required for each pixel is, such images are also called 24-bit color images.
Such an image as shown in figure 1.7 may be considered as consisting of a “stack’ of three matrices; representing the red, green and blue values for each pixel. This means that for every pixel there are three corresponding values.

1.7.1.4 Indexed image:

Most color images only have a small subset of the more than sixteen million possible colors. For convenience of storage and file handling, the image has an associated color map, or color palette, which is simply a list of all the colors used in that image. Each pixel has a value which does not give its color (as for an RGB image), but an index to the color in the map.

It is convenient if an image has 256 colors or less, for then the index values will only require one byte each to store. Some image files formats (for example,
CompuServe GIF), allow only 256 colors or fewer in each image, for precisely this reason.

Figure 1.9: An indexed color image with Indices and Color Map.

Figure 1.9 shows an image with indices, rather than being the grey values of the pixels, are simply indices into the color map. Without the color map, the image would be very dark and colorless. In the figure, for example, pixels labeled 5 correspond to 0.2627, 0.2588, 0.2549, which is a dark greyish color.

1.7.2 Image Processing Techniques:

Traditional techniques from statistical pattern recognition like the Bayesian discriminant and the Parzen windows were popular until the beginning of the 1990s. Since then, neural networks (ANNs) have increasingly been used as an alternative to classic pattern classifiers and clustering techniques. Non-parametric feed-forward ANNs quickly turned out to be attractive trainable machines for feature-based segmentation and object recognition. When no gold standard is available, the self-organizing feature map (SOM) is an interesting alternative to supervised techniques. It may learn to discriminate, e.g., different textures when provided with powerful features.
The current use of ANNs in image processing exceeds the aforementioned traditional applications. The role of feed-forward ANNs and SOMs has been extended to encompass also low-level image processing tasks such as noise suppression and image enhancement. Hopfield ANNs were introduced as a tool for finding satisfactory solutions to complex (NP-complete) optimization problems. This makes them an interesting alternative to traditional optimization algorithms for image processing tasks that can be formulated as optimization problems.

The different problems addressed in the field of digital image processing can be organized into no of steps for image processing as shown in figure 1.10.

1. **Preprocessing Filtering**: Operations that give as a result a modified image with the same dimensions as the original image (e.g., contrast enhancement and noise reduction).

2. **Data Reduction & Feature Extraction**: Any operation that extracts significant components from an image (window). The number of extracted features is generally smaller than the number of pixels in the input window.
3. **Segmentation**: Any operation that partitions an image into regions that is coherent with respect to some criterion. One example is the segregation of different textures.

4. **Object Detection & Recognition**: Determining the position and, possibly, also the orientation and scale of specific objects in an image, and classifying these objects.

5. **Image Understanding**: Obtaining high level (semantic) knowledge of what an image shows.

6. **Optimization**: Minimization of a criterion function which may be used for, e.g., graph matching or object delineation.

### 1.8 Problem Definition, Objectives & Motivation:

Human pattern recognition process is an integrated process involving the use of biological neural processing even from the stage of sensing the environment. Thus, the neural processing takes place directly on the data for feature extraction and pattern matching. Moreover, the large size of biological neural network and the inherently different mechanism of processing are attributed to our abilities of pattern recognition in spite of variability and noise in the data. Moreover, we are able to deal effortlessly with temporal patterns and also with so-called stability-plasticity dilemma as well.

These are the few reasons due to which new models of computation inspired by the structure and function of the biological neural network as natural evaluation
process are continuously evolved. Such models for computing are based on Artificial Neural Networks and Genetic Algorithms.

In this thesis we tried to implement Hybrid techniques to explore the optimal solution for real world pattern association problems. Many researchers proposed their optimal solution for pattern association problem in different domains using different technologies but not succeed in providing efficient solution as desired.

In our proposed solution, we have implemented hybrid evolutionary computing to get the best optimal solution among all available solutions. In the present research work we have proposed effective feature extraction techniques using hybrid learning rules along with existing feature extraction rules for pattern storage. We have also analyzed the performance of Hopfield Neural Network as associative memory for pattern storage and recalling of static images.

**Objectives of the Study:**

The following objectives were considered during the research:

1. To evaluate the Hopfield network and learning rules for storing and recalling the static images.
2. To develop a new hybrid learning rule so that storage capacity of Hopfield Neural Network could be enhanced.
3. To evaluate the performance of various image features extraction methods.
4. To evaluate the performance of image recalling using SOM network.
5. To optimize the recalling process with genetic algorithm for optimal result.
Therefore, to accomplish the above objectives, firstly the patterns of training set have been encoded in the neural network using conventional Hebbian learning rule, pseudoinverse rule and hybrid learning i.e. the combination of Hebbian learning and pseudoinverse rule. It is expected that all the patterns of training set has been successfully encoded as the associative memory feature of Hopfield neural network with the use of these learning rule. Now the process of recalling starts. In the process of recalling, genetic algorithm is employed for effective recalling. The effective recalling has reflected that the network should not trap any false minima and leads for the global minimum. Genetic algorithm is an evolutionary search process for obtaining global optimization solution for various problems of large search spaces. In case of genetic algorithm, the population of this approximate optimal weight matrix has been evolved using the population generation technique, crossover operator and fitness evaluation functions, until the selection of the last weight matrix or matrices has been performed. In case of genetic algorithms, we make adaptation to a obtained sub optimal synaptic weight and accept the adaptation if it improves the network performance globally. The results of various methods for recalling have been compared and analyzed.

1.9 Organization of thesis:

Basically this thesis is focusing on the Hopfield Neural Network for Hebbian learning rule, pseudoinverse rule and hybrid learning rule for storage and recalling process of the static images. In the present research work, we have used conventional effective feature extraction methods like Edge Dilation (ED), Fast Fourier Transformation (FFT) and Self Organizing Map (SOM) methods. Different patterns
sets were prepared from static images of hand written English alphabets using above feature extractions methods.

After obtaining the pattern information from the static images, the pattern vectors were constructed and training sets are prepared. These all training sets are presented to the Hopfield neural network to encode the pattern information by using different learning rules like Hebbian rule, pseudo inverse rule and hybrid learning rule and the recalling performance of Hopfield Neural Network has analyzed.

As Hopfield neural network itself cannot provide optimized result for recalling but with the help of genetic algorithms it produces comparatively better results or optimal solution i.e. genetic algorithm could increase the accuracy of recognition. To improve the recalling process we have employed sub optimal genetic algorithm on the random weights of Hopfield Neural Network. This thesis is organized as follows.

Chapter-1 introduces the fundamental concepts of soft computing, artificial neural networks and genetic algorithm. Further it includes the problem definition motivation and objectives of the proposed work. This chapter ends with organization of the thesis i.e. chapter scheme of the research work.

Chapter-2 incorporates the literature review on pattern storage, Hopfield network, associative memory and neural networks. It includes various statistical, structural, neural network, fuzzy based methods and evolutionary algorithm used for pattern recognition and classification.
Chapter-3 explains the research methodology adopted to conduct the experiments in this research work. It tells about the overall flow of the research, preprocessing of data and its execution.

Chapter-4 is talking about the outcome results of the experiments conducted on the pattern and its interpretation. Various experiments were conducted using ED, FFT and SOM method for feature extraction and pattern storage & recalling. Finally genetic algorithms have used to optimize the pattern recalling process.

Chapter-5 discussed the conclusion and recommendation for future use.

After all chapter, References are given as a separate list, then Appendix-I gives the general introduction about MatLab tool, Appendix – II which gives the details of the papers published in various international journals. At the end Appendix-III, provides the CV of Supervisor and research scholar.

1.10 Conclusions:

In this chapter we have explored the different aspects of soft computing methods those are used massively in the understanding and modeling of soft computing techniques. The potential & applicability of the soft computing techniques were discussed in detail. It is evident from the discussion that models of artificial neural network are finding their application in various walks of our life through neurocomputing. These includes pattern classification, pattern association, pattern mapping, signal processing, character recognition, various fields of forecasting, process monitoring and control, etc.
In this chapter of the thesis we have discussed various artificial neural network approaches out of which some of them are used in this thesis for the modeling of the neuro-computing models and important aspect about images, images types, image features and image processing tasks. After that we have defined problem definition, objectives and motivation and followed by organization of thesis.