CHAPTER VII
FINDING IFS USING NEURAL NETWORKS

7.1 Introduction

Fractal image coding takes advantage of image self-similarities on different scales. Most fractal algorithms, beginning with Jacquin's implementation [1][2], operate on an image segmentation consisting of non-overlapping square regions, called ranges. Each range block is encoded by a non-expansive transformation \( T \) operating on the whole image and mapping a domain block. This basic idea is followed in this IFS coding using the principles of Neural Networks. Several-sophisticated image matching or pattern-matching techniques based on statistical and AI methods are available. But, these techniques consume a lot of time when it is adopted in fractal IFS algorithm. Invariably, the existing fractal image compression algorithms suffer in performance mainly because of the lack of an efficient image classification/sorting method [140].

In a typical fractal image compression algorithm, there will be 'D' domain blocks which have to be compared with 'R' range blocks in the original image 'I' and \( D \times R \) block matching are to be performed. If \( D \times R \) block matching is performed with \( N \) number of possible geometric and intensity transformations, the process will consume a lot of time obviously. Implementing image compression by using an appropriate Neural Network technique [149] can minimize this IFS coding time.

Although, the proposed fractal image compression algorithm needs simple and fast image classification procedures, most of the existing image classification and recognition works are meant for face recognition, fingerprint classification and general-purpose image identification. The common global search mechanisms consume a lot of time since a lot of geometric transformations and image comparison operations are involved in that process. The possibilities of applying Neural Network based image block
classification/sorting algorithm [146] to find the near optimal solution in fast and efficient manner has been explored in this thesis.

In principle, a popular Neural Network can be trained to recognize image blocks directly. However, a simple network can be very complex and difficult to train for recognizing a number of image blocks involved in a typical fractal image compression scenario. To find a suitable match for a range block image, it has to be compared with all the domain block images with all possible transformations. Obviously it will require a lot of computation time. Further, the time will increase rapidly even for a little increase in the number of domain blocks or range blocks or for both.

In this Neural Network based model knowledge is usually distributed throughout the nodes and is stored in the structure of the topology as the weights of the links. The network is organized by automated training methods which greatly simplify the development of specific methods in this application. There are two advantages of using these models. In situations where no clear set of logical rules are given, the first advantage allows the classical logic rules in ordinary AI systems to replace the vague conclusions and associative rules (exact match versus best match). The inherent fault tolerance of connective models is another advantage.

Furthermore, Neural Networks can be made tolerant against noise in the input. Generally it exhibits graceful performance degradation with increased noise and the quality of the output degrades at a slow pace. The Neural Network can be trained to do this classification in a fast and an efficient manner. And four steps are involved in the training process:

- Assemble the training data from the pixel values of the domain Blocks.
- Create the network object
- Train the network with the data for predefined results.
- Simulate the network response to new inputs
7.2 Neural Network in Image compression

Neural networks are composed of simple elements operating parallely. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements. It can train a Neural Network to perform a particular function by adjusting the values of the connections (weights) between elements. Generally, the Neural Networks are adjusted, or trained, so that a particular input leads to a specific target output as shown in Figure 7.1. And, the network is adjusted based on a comparison of the output and the target until the network output matches the target.

![Figure 7.1: A Neural Network Diagram](image)

A typical image recognition network requires \( N = m \times n \) input neurons, one for each of the pixels in an \( m \times n \) image. For example, if the image is 128x128 size, the number of inputs of the network would be 16,384. In order to reduce the complexity, Cottrell and Fleming used two Back Propagation nets. The first net operates in the auto-association mode and extracts features. The second net operates in the more common classification mode [3]. Generally, the techniques used in the Neural Network systems
depend on the application of the system. The goal of this research is to identify the suitable network for image block classification and to use it in fractal image compression.

7.3 Multi-Layer Neural Network Architecture

In the implementation, the proposed multi-layer neural network design used for fractal IFS coding is explained. It is a simple feed forward network with three layers in the neural network. In the input layer there will be P neurons. The number of neurons in the input layer will be equal to the number of pixels in a domain block. The number of neuron in the second layer (Hidden Layer) will be decided after testing the accuracy of classification. It can be decided based on the number of domain blocks. In the output layer, there will be D neurons representing the D domain blocks. Once the network weights and biases have been initialized, the network is ready for training. The network can be trained for function approximation (nonlinear regression), pattern association, or pattern classification. The training process requires a set of examples of proper network behavior with input neurons ‘p’ and target outputs ‘t’. The figure 7.2 shows the proposed multilayer neural network architecture.

Generalizing the Widrow-Hoff learning rule to multilayer networks and nonlinear differentiable transfer functions created back propagation. Input vectors and the corresponding target vectors are used to train a network until it can approximate a function, associate input vectors with specific output vectors, or classify input vectors in an appropriate way as defined. Networks with biases, a sigmoid layer, and a linear output layer are capable of approximating any function with a finite number of discontinuities. Standard back propagation is a gradient descent algorithm, and the network weights are moved along the negative of the gradient of the performance function as in Widrow-Hoff learning rule.
The term back propagation refers to the manner in which the gradient computed for nonlinear multilayer networks. A number of variations are there in the basic algorithm based on other standard optimization techniques such as conjugate gradient and Newton methods that are used. Properly trained back propagation networks tend to give reasonable answers when presented with unseen inputs. Typically, a new input leads to an output similar to the correct output for input vectors used in training. This generalization property makes it possible to train a network on a representative set of input/target pairs and get good results without training the network on all possible input/output pairs.
The simplest implementation of back propagation learning updates the network weights and biases in the direction in which the performance function decreases the negative of the gradient rapidly.

One iteration of this algorithm can be written as

$$x_{k+1} = x_k - \alpha_k g_k$$

Where $x_k$ is a vector of current weights and biases, $g_k$ is the current gradient and is the learning rate. This gradient descent algorithm can be implemented in incremental mode and batch mode. In the incremental mode, the gradient is computed and the weights are updated after each input is applied to the network. In the batch mode, all the inputs are applied to the network before the weights are updated.

In the proposed algorithm, the search space is reduced before the intensive search which involves a lot of transformation. For reducing the search space, the redundant blocks are removed from the decomposed image using RBRS mechanism. For finding the obviously similar blocks, a simple block matching technique based on RMSE value [48] is used. During this process, no geometric or intensity transformation is considered for the block matching operation. The RBRS strategy of this algorithm will consume very low time when compared to the overall time for finding IFS using all the possible transformations.
In figure 7.3, the original image ‘A’ is first decomposed into blocks of various sizes using quadtree decomposition as in ‘B’. Then the image blocks are separated as two groups namely smooth blocks ‘C’ and rough blocks ‘D’ [112]. After performing the compression operation using the proposed multi layer Neural Network architecture, the IFS code is created. Later, appropriate decompression operation is carried out and the
image in ‘E’ is produced. The quality is measured for both the original and decompressed image by calculating PSNR in db.

7.4 Proposed Compression Algorithm

[1] Take an input image I and crop as a square image of particular size which will be suitable for quadtree decomposition.

[2] Decompose the given image into a number of non-overlapping blocks of various sizes based on its features and details using quadtree decomposition.

[3] Remove the redundant blocks from all size groups by leaving only one seed block (domain block) as smaller seed blocks are to be used for coding the image.

[4] Separate smooth blocks and rough blocks based on the values of Variance of each block from the remaining blocks.

[5] Store the rough blocks as well as the smooth blocks as it is.

[6] Consider the image as 'S_n' smooth blocks as domain blocks (seeds) of uniform sizes.

[7] Consider R_n = (Total blocks - Rough blocks) as the range blocks of various sizes to be coded. Then there will be S_n domain blocks and R_n range blocks.

[8] Construct a Multi-layer Neural Network as in figure 7.2 and train it with the S_n number of domain blocks. Repeat the training for a predefined criterion.

[9] Select a range block as the input for the trained network for classification.

[10] Register the result (in this case it will be the index of the domain block which can be substituted for that range block.).

[11] Repeat all the steps from Step 9 for all the range blocks.

[12] Record the IFS code and stop.
7.5 Implementation Results

Figure 7.4, illustrates the various stages of images of this proposed algorithm. In each row, the Column one is the original image, Column two is the image after applying quadtree decomposition, Column three is the rough blocks of image, Column four is the smooth blocks of image, and Column five is the reconstructed image from the proposed Neural Network based IFS coding. The implementation results in terms of time and quality is given in Table 7.1 and Table 7.2. The results of Neural Network algorithm have been compared with the results of Normal IFS algorithm.

The experiments have been done using Neural Networks algorithm to measure performance based on two parameters such as compression time and quality in PSNR. The experiments have been done on various images of 128 x 128 and 256 x 256 size. The execution results for various images of the size 128 x 128 are presented in Table 7.1. The similar experiment is performed on 256 x 256 size images and the results are presented in Table 7.2. The obtained results are significant and the time taken for IFS coding is very minimum, but the quality of the decompressed image is comparatively poor.

Initially, the input image is decomposed into a number of blocks using quadtree decomposition method. Then, the redundant blocks are removed to reduce the search space for the block matching operation. In the remaining blocks, further division is made and the blocks with 4 x 4 size are taken for performing block matching operation to generate IFS coding [30]. For training the network, the image blocks of 2 x 2 size from the original 4 x 4 size are used. The blocks after the decomposition are named as smooth blocks and rough blocks. In fractal image compression, these blocks are called domain blocks (seed blocks) and range blocks.
FIGURE 7.4 various stages of Face, Lena and Nature Images of size 128 x 128 after applying Neural Networks Algorithm.
### TABLE 7.1
EXPERIMENTAL RESULTS IN TERMS OF TIME AND PSNR USING NEURAL NETWORKS ALGORITHM FOR 128 X 128 SIZE IMAGES

<table>
<thead>
<tr>
<th>Image</th>
<th>Range Blocks</th>
<th>Domain Blocks</th>
<th>QT and Training</th>
<th>Time (in Sec.)</th>
<th>PSNR in db</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face1</td>
<td>733</td>
<td>184</td>
<td>3.46</td>
<td>3.71</td>
<td>7.86</td>
</tr>
<tr>
<td>Face2</td>
<td>811</td>
<td>283</td>
<td>3.43</td>
<td>4.12</td>
<td>7.96</td>
</tr>
<tr>
<td>Face3</td>
<td>745</td>
<td>174</td>
<td>3.57</td>
<td>3.46</td>
<td>7.03</td>
</tr>
<tr>
<td>Lena</td>
<td>943</td>
<td>310</td>
<td>3.06</td>
<td>4.97</td>
<td>8.03</td>
</tr>
<tr>
<td>Nature</td>
<td>631</td>
<td>133</td>
<td>3.09</td>
<td>2.91</td>
<td>6.00</td>
</tr>
</tbody>
</table>

### TABLE 7.2
EXPERIMENTAL RESULTS IN TERMS OF TIME AND PSNR USING NEURAL NETWORKS ALGORITHM FOR 256 X 256 SIZE IMAGES

<table>
<thead>
<tr>
<th>Image</th>
<th>Range Blocks</th>
<th>Domain Blocks</th>
<th>QT and Training</th>
<th>Time (in Sec.)</th>
<th>PSNR in db</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face1</td>
<td>2053</td>
<td>628</td>
<td>13.11</td>
<td>12.03</td>
<td>25.14</td>
</tr>
<tr>
<td>Face2</td>
<td>2344</td>
<td>968</td>
<td>12.99</td>
<td>12.38</td>
<td>25.36</td>
</tr>
<tr>
<td>Face3</td>
<td>1969</td>
<td>675</td>
<td>12.93</td>
<td>10.82</td>
<td>23.75</td>
</tr>
<tr>
<td>Lena</td>
<td>2253</td>
<td>977</td>
<td>12.23</td>
<td>15.72</td>
<td>27.96</td>
</tr>
<tr>
<td>Nature</td>
<td>1864</td>
<td>725</td>
<td>13.03</td>
<td>9.70</td>
<td>22.73</td>
</tr>
</tbody>
</table>

**7.5.1 The Comparison of Performance in terms of Speed**

The Table 7.1 shows the performance of proposed Neural Network method for the images of size 128 x 128 in terms of speed and quality. To find the suitability of the algorithm for image compression operation, it is compared with the performance of the Normal IFS algorithm. The study shows clearly that Neural Network method provides minimum compression time for IFS coding but poor decompressed image quality. The comparative study of this result is shown in Graphs 7.1 and 7.2.
Comparative study of Time for Normal IFS and Neural Network based IFS

**GRAPH 7.1**
Analysis in terms of Time for 128 x 128 size images using Normal IFS and Neural Networks Algorithm

Comparative study of Normal IFS and Neural Networks IFS algorithm for 256 x 256 size images

**GRAPH 7.2**
Analysis in terms of Time for 256 x 256 size images using Normal IFS and Neural Networks Algorithm
7.5.2 The Comparison of Performance in terms of Quality

The quality of a decompressed image is the other important criteria measured and analyzed for both Normal IFS algorithm and Neural Networks algorithm. The execution results of all sizes of images have been compared and the study on performance for one size is shown in the Graph 7.3.

![Graph 7.3](image)

**GRAPH 7.3**
Analysis in terms of Quality (PSNR) for 128 x 128 size images using Normal IFS and Neural Networks Algorithm

7.6 Results of High Resolution images

The main finding of this research is that the Neural Network based implementation consumes less time for IFS coding for all sizes of images invariably, but the quality of decompressed image is very poor. For example, the Lena image of 1024 X 1024 size consumes 415 seconds for IFS coding and the calculated PSNR is 25.43 db. The results of this study reveal that Neural Network based implementation needs further enhancement in terms of decompressed image quality. Hence, the results of experiments on higher resolution images using neural network based implementation are not presented
here. The further exploration can be done on improving the quality of decompressed image with the mentioned framework for the implementation.

7.6 Summary

The proposed Neural Network based approach is considered to be a good approach in fractal based image compression. Though the performance in terms of quality is poor when compared to the Normal IFS method and other proposed methods mentioned in this research, further exploration is needed to improve the quality aspects. The Neural Network architecture designed has some shortcomings due to the direct use of image pixel gray values. As a result, the system becomes sensitive to sudden changes in individual pixel values and needs a beforehand pre-processing step [43]. Satisfactory classification performances could be reached by adopting suitable image filtering techniques and dimensionality reduction techniques such as PCA at the pre-processing and post-processing stages. The results of Neural Network based approach seem to be significant in terms of time compared to the Normal IFS method. During the testing process, the training is given to 300 epochs to improve the performance (encoding time) of the algorithm. The system performs well at very low training also. A training of 10-100 epochs [88] is enough to produce similar results.

In this work, the Neural Network based system is found good only for the performance in terms of speed. Eventhough the performance of the system in terms of quality is not satisfactory when compared to other methods proposed here, the possibilities of improvements in image quality without sacrificing the existing speed may be explored in future works. In this implementation, only a portion of the selected seed blocks is used for coding (smooth blocks). The remaining blocks are stored as rough blocks. In future implementations, all the suitable blocks from the two categories can be used for coding [88].