Image Compression Using Neural Networks

Sumit Chhikara & Mrs. Reema Sachdeva
1M.Tech student of Department of Computer Science and Engg.
Sat Kabir Institute of Technology and Management Bahadurgarh, Haryana - 124507
2A.P in CSE Deptt. Sat Kabir Institute of Technology and Management Ladrawan, Bahadurgarh – Haryana – 124507

Abstract: The aim is to design and implement image compression using neural network to achieve better compression levels. The compression is first obtained by modeling the neural network in MATLAB. Image compression addresses the problem of reducing the amount of data required to represent a digital image.

Keywords: Image compression: image processing, image enhancement, image restoration, compression standards

1. INTRODUCTION

Uncompressed multimedia such as audio, video, graphics requires considerable storage capacity and transmission bandwidth. Main goal of image compression is to represent input signal using minimal amount of resources. Performance measurement of image compression is done by its efficiency, complexity and distortion measurement.

- Fast multiplication, leading to fast update of the neural network.
- Flexibility, because different network architectures are possible.
- Scalability, as the proposed hardware architecture can be used for arbitrary large network, considered by the no. of neurons in one layer. The availability of user-configurable, digital field programmable gate arrays, which can be used for experiments.

1.1 Image processing: various techniques used for image processing are bitmap, encapsulated postscript and tagged image file format.

<table>
<thead>
<tr>
<th></th>
<th>GIF</th>
<th>JPEG</th>
<th>TIFF</th>
<th>PNG</th>
</tr>
</thead>
<tbody>
<tr>
<td>When to Use</td>
<td>For images with few colors for the Web, animation, or multimedia.</td>
<td>For photographic images for the Web, e-mail or multimedia.</td>
<td>For scanned images and for desktop publishing—printed images.</td>
<td>For photos or single images (not animations) on the Web.</td>
</tr>
<tr>
<td>Compression</td>
<td>Lossy. Changes original image during conversion.</td>
<td>Lossy. Changes original image during conversion.</td>
<td>Lossless. Detail and clarity are retained when the image is edited or saved.</td>
<td>Lossless. Detail and clarity are retained when the image is edited or saved.</td>
</tr>
<tr>
<td>Resolution for 4”x5.5” or 4”x6” Print Size</td>
<td>Set the resolution (pixels) to at least 600x900 and the compression (file size) to 300KB.</td>
<td>Set the resolution (pixels) to at least 600x900 and the compression (file size) to 1.5MB.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table – 1 various compression standards

2. LITERATURE REVIEW

P. Srikala 1, Shaik and Faruq in 2012 proposed an approach for image compression using MLFF neural network with error propagation training algorithm. S.L Pinjare and M Arun Kumar presented the implementation of trainable artificial neural network which can be trained to implement certain functions. S. Anna Durai and E. Anna Saro proposed an approach of mapping the pixels.

In the research work, a detailed study is done about the compression mechanisms through a literature review. During the thesis period our priority will be to study and compare the techniques earlier developed for the image compression as well as the implementation
technique to achieve high recognition accuracy is tried. We are going to show 2 compression mechanisms – through Back Propagation Neural Network for Grey Scaled Images and through Wavelet based mechanism for Colored images. As back propagation algorithm provides an easy platform to do supervised learning with various learning techniques. Back propagation algorithm will be used with multiple training techniques to compare and test the results based on recognition accuracy, no. of epochs taken and time required for each technique. The techniques are illustrated in detail and implemented. Stepwise operations are shown here.

3. PROBLEM FORMULATION

The goal of thesis is to create an image compression mechanism that use an artificial neural network as the backend to solve the compression problem for gray scaled images. The problem inherent to any digital image is the large amount of bandwidth required for storage. In this research a three layered back propagation neural network will be designed for building image compression. The next problem is how to train the neural network. Back propagation algorithm will be used with multiple training techniques to compare and test the results based in recognition accuracy.

4. ALGORITHM USED AND IMPLEMENTATION

A feed-forward back propagation network is created using function newff. It requires basically three arguments and returns the network object. The first argument is a matrix of sample R-element input vectors. The second argument is a matrix of sample S-element target vectors. The sample inputs and outputs are used to set up network input and output dimensions and parameters. The third argument is an array containing the sizes of each hidden layer. (The output layer size is determined from the targets). More optional arguments can be provided. For instance, the fourth argument is a cell array containing the names of the transfer functions to be used in each layer. The fifth argument contains the name of the training function to be used. If only three arguments are supplied, the default transfer function for hidden layers is tansig and the default for the output layer is purelin. The default training function is trainlm. newff command creates the network object and also initializes the weights and biases of the network; therefore the network is ready for training. The command used in this work is given as under:

\[
\text{net} = \text{newff} \text{arguments} \]

Step 1: Initialization: Assuming that no prior information is available, pick the synaptic weights and thresholds from a uniform distribution whose mean is zero and whose variance is chosen to make the standard deviation of the induced local fields of the neurons lie at the transition between the linear and saturated parts of the sigmoid activation function.

Step 2: Presentation of Training examples: Present the network with an epoch of training examples. For such example in the set, ordered in some fashion, perform the sequence of forward and backward computations.

Step 3: Forward Computation: Let a training example in the epoch be denoted by \((x(n),d(n))\), with the input vector \(x(n)\) applied to the input layer of sensory nodes and the desired response vector \(d(n)\) presented to the output layer of computation nodes. Compute the induced local fields and function signals of the network by proceeding forward through the network, layer by layer. The induced local field \(v_j(n)\) for neuron \(j\) in layer \(l\) is:

\[
v_j(n) = \sum w_{jl}(n) y_l(n)\]
where $y_i(n)$ is the output (function) signal of neuron $i$ in the previous layer $l-1$ at iteration $n$ and $w_{ji}(n)$ is the synaptic weight of neuron $j$ in layer $l$ that is fed from neuron $i$ in layer $l-1$. For $i=0$, we have $y_i(n) = +1$ and $w_{ji}(n) = b_j(n)$ is the bias applied to neuron $j$ in layer $l$.

Assuming the use of a sigmoid function, the output signal of neuron $j$ in layer $l$ is

$$y_j(n) = \phi(v_j(n))$$

If neuron $j$ is in the first hidden layer (i.e. $l=1$), set $y_j(n) = o_j(n)$

Compute the error signal:

$$e_j(n) = d_j(n) - o_j(n)$$

where $d_j(n)$ is the $j$th element of desired response vector $d(n)$.

**Step 4: Backward Computation:** Compute the $\delta_j$ (i.e. local gradients) of the network, defined by

$\delta_j(n) = e_j(n) \phi'(v_j(n))$ for neuron $j$ in output layer $L$.

$\delta_j(n) = \phi'(v_j(n)) \sum \delta_k(n) w_{kj}(n)$ for neuron $j$ in hidden layer $l$.

where the prime in $\phi'(.)$ denotes differentiation with respect to the argument. Adjust the synaptic weights of the network in layer $l$ according to the generalized delta rule:

$$w_{ji}(n+1) = w_{ji}(n) + \eta \left[ w_{ji}(n-1) + \eta \delta_j(n) y_i(n) \right]$$

where $\eta$ is the learning rate parameter and $\alpha$ is the momentum constant.

**Step 5: Iteration:** Iterate the forward and backward computations under steps 3 and 4 by presenting new epochs of training examples to the network until the stopping criterion is met.

In the research work, we have applied various image processing techniques and then neural network to train and test the compression of an image. The ability of neural networks to learn by example makes them very flexible and powerful. Also to perform color image compression, we have implemented a wavelet based image compression algorithm.

**REFERENCES**


