Image Coding Using Self-Supervised Backpropagation Neural Network

Wei Gong, IEEE student member, K. R. Rao and M. T. Manry, IEEE senior member
Department of Electrical Engineering, University of Texas at Arlington,
P.O. Box 19016, Arlington, Tx 76019

Abstract
In this paper, image data compression is implemented using a self-supervised backpropagation neural network. First, a backpropagation discrete cosine transform (BPDCT) is developed and then used for transform coding. Secondly, to alleviate edge distortion, classification techniques are applied to transform image coding. The classification technique is based on the edge detection since that the human visual system is more sensitive to edges. Simulation results are given.

1 Introduction
Many researchers are interested in image data compression since it is widely used in many applications, such as teleconferencing, telebrowsing, videotext, electronic shopping in mail order, accessing large databases, security services, HDTV, and so on [1]. A fundamental objective of image compression is to represent an image using a minimum number of bits, while maintaining good fidelity. Many approaches [2] are used to reach this goal. One of the methods is transform coding [3,4,5,6]. Compression is obtained by transforming the input image into a domain where a large amount of information is packed into a few coefficients. The advantages of transform coding are that (1) it maps statistically dependent or correlated image data into independent or uncorrelated coefficients and (2) the channel error within each block does not propagate into the other blocks since each block's coefficients are processed independently.

Neural networks have become popular recently because of their parallelism, mapping ability, and learning ability [7,8]. One is the popular backpropagation neural network (BPNN). It transfers input signals from one domain into another domain. The input-output transform is determined by supervised adjustment of the system parameters through training. Conventional transforms can be performed by the BPNN. The performance of transforms for a given task, can be improved by the BPNN's training. The issues addressed above have led us to begin this research, that is to map the discrete cosine transform (DCT) into BPNN.

Section 2 gives a short description of the BPNN. Section 3 develops the BPDCT. Section 4 shows the implementation of image coding using the BPDCT with simulation results. Section 5 gives the classification technique which is applied to image coding using the BPDCT. Conclusions are given in section 6.

2 Backpropagation Neural Network
Figure 1 shows the architecture of a simple three-layer BP neural network (BPNN) [8,12,13]. The general description of the BPNN is given in the following. The output of each unit in the BPNN, is

\[ O(i,j) = f(\text{net}(i,j)) \]

\[ \text{net}(i,j) = \sum_{Ls=Ld(i)}^{Ls=Ld} \sum_{Ns=1}^{N_s} w_s(i,Ls,j,Ns)O(Ls,Ns) \]

where \( O(i,j) \) is the output of unit \( j \) at layer \( i \). \( f(\bullet) \) is the activation function. \( Lf(i) \) is the number of the first layer which connects to layer \( i \). \( N_s(i) \) is the number of units in layer \( i \) for \( i=1,\ldots,NL \). \( NL \) is the number of layers. \( w_s(i,Ls,j,Ns) \) is the weight connecting unit \( Ns \) of layer \( Ls \) to unit \( Nd \) of layer \( Ld \). \( Ld \) is between 2 and \( NL \), \( Ls \) is between \( Lo(Ld) \) and \( Ld-1 \), where \( LO(i) \) is the last layer which the layer \( i \) feeds. \( \text{net}(i,j) \) is a weighted sum of signals received at unit \( j \) of layer \( i \). It is used to calculate the output of unit \( j \) at layer \( i \).

In general, the outputs of units in the output layer are different from the target output \( Tu(j) \), where \( j=1,\ldots,Nu(NL) \). The error between the \( O(NL,i,j) \) and \( Tu(j) \) is measured by the mean-square error,

\[ \epsilon = \frac{1}{2} \sum_{j=1}^{Nu(NL)} [ Tu(j) - O(NL,i,j) ]^2 \]

In supervised learning, the goal is to minimize the error \( \epsilon \) with respect to the weights \( w_s(i,Ls,j,Ns) \). Rumelhart and others [8] have developed the backpropagation algorithm. The basic goal is to minimize \( \epsilon \) with respect to \( w_s(Ld,Ls,j,k) \).

3 Modified Discrete Cosine Transform Using Self-Supervised BPNN
The DCT is widely used in image compression since
it leads to high data compression rates [1]. In practice there are a few problems: (1) For different input vectors, the distribution of energy among the DCT coefficients changes significantly, (2) Although the DCT concentrates energy well, it is not optimal like the KLT. BPNN can be used in an attempt to alleviate these problems.

Figure 2a shows the truncated 2-D DCT scheme. Input blocks of size NxN are transformed using a 2-D DCT, truncated to size MxM blocks (M ≤ N), and then inverse 2-D DCT transformed. Here, we map the above processes to the BPNN. The mapping of this 2-D DCT process is described as follows. First the 2-D forward DCT and 2-D inverse DCT are rewritten in lexicographic form. Let x(v) and x*(v) be the lexicographic form for an input block and output of the 2-D inverse DCT of size of NxN, v = 1,...,N^2 respectively, and let T(u) be the lexicographic form for the truncated 2-D DCT, u = 1,...,M^2. The wF(u,v) and wI(v,u) are kernel coefficients for the 2-D forward and inverse DCT in lexicographic form respectively. T(u) is written as

\[ T(u) = \sum_{v=1}^{N^2} wF(u,v) x(v), \text{ where } u = 1,2,...,M^2 \] (4)

The lexicographic form for the 2-D inverse DCT is

\[ x^*(v) = \sum_{u=1}^{M^2} wI(v,u) T(u), \text{ where, } v = 1,2,...,N^2 \] (5)

Map equations 4 and 5 to a three layer backpropagation neural network. The inputs of the first layer are samples of a spatial domain block, x(v), v = 1,...,N^2. The second layer outputs are samples of the 2-D forward DCT, T(u), u = 1,...,M^2. The third layer outputs are samples of the 2-D inverse DCT, x^*(v), v = 1,...,N^2. The resulting neural network is constructed as shown as Figure 2b, and is called the BP neural network DCT (BPDCT). The BPDCT is a three layer self-supervised BP network with units N(1) = N^2, N(2) = M^2, N(3) = N^2, and NL = 3.

In this BPNN, we choose DCT transform kernel coefficients, wF(u,v) and wI(v,u) as initial weights, since the DCT has good compression performance. We also use the input blocks as the desired outputs of the network (this is why it is called self-supervised BP neural network). Here, each unit’s activation function is linear, since the DCT is a linear transform. The outputs of input units are the image blocks. The outputs of hidden and output units are

\[ O(i,j) = \text{net}(i,j) = \sum_{k=1}^{N^2} wO(i, i-1,j,k) O(i-1,k) \] (6)

wF(2,1,j,k) and wI(3,2,j,k) in BPNN correspond to the DCT kernel coefficients wF(u,v) and wI(v,u) respectively. The output evaluation function is the summation of the mean-square errors over all training blocks,

\[ \xi = \frac{1}{2} \sum_{p=1}^{Npat} \sum_{j=1}^{NL} [ T_u(j) - O_p(NL,j) ]^2 \] (7)

where Npat is the total number of training blocks and the subscript p stands for the index of the training blocks. The mean-square error \( \xi \) is minimized by setting its gradient with respect to the system’s unknown weights equal to zero, and then solving for the weights. Since the DCT has a symmetry property, we only take the output weights wI(3,2,i,j) as unknown variables. The error function is defined as

\[ \xi = \frac{1}{2} \sum_{p=1}^{Npat} \sum_{j=1}^{NL} [ T_u(j) - O_p(NL,j) ]^2 = \sum_{j=1}^{NL} \epsilon_j \] (8)

where \( \epsilon_j \) is the total error in unit \( j \) over all patterns. Setting the gradient of \( \epsilon_j \) with respect to \( wI(3,2,j,k) \) to zero,

\[ \frac{\partial \epsilon_j}{\partial wI(3,2,j,k)} = 0 \] (9)

Solving the M^2 linear equations, we can get the M^2 unknown weights wI(3,2,j,k) at the output unit \( j \). Since \( j = 1,...,N^2 \), we have \( N^2 \) sets of linear equations. After solving all \( N^2 \) sets of linear equations, we get improved weights wI(3,2,j,k). These weights are then copied onto the input layer weights. The training procedure is performed.
between the original image and reconstructed image.

In Table 1, the image coding results between the truncated DCT and BPDCT are compared. The simulation results show that the average performance of the image coding using BPDCT is slightly better than using the truncated DCT. Figure 4 is original "girl" image. Figures 5 and 6 are coded "girl" images.

5 Classified Transform Image Coding Using BPDCT

The classification algorithm makes decisions about the activities of the image blocks. Our classification algorithm is based on edge detection since the human visual system is more sensitive to edges. The classifier is a simple rule-based classification algorithm. After the image is preprocessed, a set of representative numbers, called features, are calculated for each block. Using a set of simple rules, the classifier uses each block’s features to decide which class the block belongs to.

In the preprocessing scheme of Figure 7, the Sobel operator is used to obtain the gradient, angle and edge images. The segmentation image is obtained using an edge-guided thresholding algorithm. There are five features. In a given block, feature 1, or Nedge, is the number of edge pixels. Feature 2, or Nedge_1, is the number of edge pixel calculated for each block. Using a set of simple rules, the classifier uses each block’s features to decide which class the block belongs to.

In the preprocessing scheme of Figure 7, the Sobel operator is used to obtain the gradient, angle and edge images. The segmentation image is obtained using an edge-guided thresholding algorithm. There are five features. In a given block, feature 1, or Nedge, is the number of edge pixels. Feature 2, or Nedge_1, is the number of edge pixel calculated for each block. Using a set of simple rules, the classifier uses each block’s features to decide which class the block belongs to.

The classification algorithm makes decisions about the activities of the image blocks. Our classification algorithm is based on edge detection since the human visual system is more sensitive to edges. The classifier is a simple rule-based classification algorithm. After the image is preprocessed, a set of representative numbers, called features, are calculated for each block. Using a set of simple rules, the classifier uses each block’s features to decide which class the block belongs to.

In the preprocessing scheme of Figure 7, the Sobel operator is used to obtain the gradient, angle and edge images. The segmentation image is obtained using an edge-guided thresholding algorithm. There are five features. In a given block, feature 1, or Nedge, is the number of edge pixels. Feature 2, or Nedge_1, is the number of edge pixel calculated for each block. Using a set of simple rules, the classifier uses each block’s features to decide which class the block belongs to.
Table 2. Comparison Of CBPDCT and BPDCT

<table>
<thead>
<tr>
<th>Coding scheme</th>
<th>CBPDCT (8x8 → 4x4)</th>
<th>BPDCT (8x8 → 4x4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image</td>
<td>MSE</td>
<td>SNR(dB)</td>
</tr>
<tr>
<td>Girl</td>
<td>16.09</td>
<td>36.06</td>
</tr>
<tr>
<td>Boat</td>
<td>12.21</td>
<td>37.26</td>
</tr>
<tr>
<td>Martha</td>
<td>15.00</td>
<td>36.36</td>
</tr>
<tr>
<td>Baboon</td>
<td>251.3</td>
<td>24.13</td>
</tr>
</tbody>
</table>

locations where the gradient direction is less than or equal to 90 degrees. Feature 3, or Nedge_2, is the number of edge pixel locations where the gradient direction is greater than 90 degrees. Let Itseg be a threshold such that approximately one half the pixels in the segmentation image are less than or equal to Itseg. Itseg is obtained using the cumulative histogram of the segmentation image. In a given block in the segmentation image, feature 4, or Kseg_1, is the number of pixels having gray level greater than Itseg. Feature 5, or Kseg_2 equals \((N^2 - Kseg_1)\) where \(N\) is the block dimension. The rule-based classifier is shown in Figure 8, where \(t\) is the edge threshold.

We experimented with classified transform coding using the BPDCT (CBPDCT) using the system of Figure 9. The blocks of size \(N \times N\) (here, \(N = 8\)), are classified into four classes. For each class, the blocks are transform coded using the BPDCT, so we have four forward and inverse BPDCT's. Overhead information consists of the classification map, which increases the bit rate. At the receiver the quantized BPDCT coefficients and overhead information are decoded to get the location of blocks in the image. Then the corresponding IBPDCT is used to reconstruct the image.

We have compared the performance between image coding using BPDCT and CBPDCT. The simulation results, shown in Table 2, show that the average performance of using the CBPDCT is slightly better than image coding using BPDCT. Figure 10 is the coded "girl" image.

6 Conclusions

In this paper image coding is implemented using a self-supervised BPNN. The resulting BPDCT is sub-optimal for the given training data sets. The simulation results show that the BPDCT works better for image coding than a truncated DCT. Classification techniques improve the performance of the BPDCT.

References


