Image compression using a self-organized neural network

Qiang Ji
Dept. of Electrical Engineering
University of Washington

ABSTRACT
In the research described by this paper, we implemented and evaluated a linear self-organized feedforward neural network for image compression. Based on the Generalized Hebbian Learning Algorithm (GHA), the neural network extracts the principle components from the auto-correlation matrix of the input images. To do so, an image is first divided into mutually exclusive square blocks of size m x m. Each block represents a feature vector of m² dimension in the feature space. The input dimension of the neural net is therefore m² and the output dimension is m. Training based on GHA for each block then yields a weight matrix with dimension of m x m², rows of which are the eigenvectors of the auto-correlation matrix of the input image block. Projection of each image block onto the extracted eigenvectors yields m coefficients for each block. Image compression is then accomplished by quantizing and coding the coefficients for each block.

To evaluate the performance of the neural network, two experiments were conducted using standard IEEE images. First, the neural net was implemented to compress images at different bit rates using different block sizes. Second, to test the neural network's generalization capability, the sets of principle components extracted from one image was used for compressing different but statistically similar images. The evaluation, based on both visual inspection and statistical measures (NMSE and SNR) of the reconstructed images, demonstrates that the network can yield satisfactory image compression performance and possesses a good generalization capability.

Keywords: image compression, neural networks

1. INTRODUCTION
A self-organized feedforward neural network for principle component analysis (PCA) consists of an input layer and output layer. The dimension reduction is realized by having a larger input dimension than the output dimension. A linear activation function is used for computing the output of each output node. The GHA learning mechanism is used for network training. According to the GHA, the weight matrix update is given by the following iterative equation:

\[ W_{ji}(n+1) = W_{ji}(n) + \mu \sum_{k=1}^{j} W_{ki} y_k(n) \]  (1)

where \( W_{ji} \), \( y_j(n) \), \( x_i(n) \), and \( \mu \) represent the weight matrix, output, input, and the learning rate respectively.

Sanger proved that if \( W_{ji} \) is initialized to random weights, eq. (1) will converge, and \( W \) will approach a matrix whose rows are the first m eigenvectors of the input correlation matrix, ordered by decreasing eigenvalues.

2. NETWORK TRAINING AND PCA
To generate the training data, input images were divided into mutually exclusive square blocks of size m x m. Each block represents a feature vector of m² dimension in the feature space. The input dimension of the neural net is therefore m² and the output dimension m. During training, the input image is raster scanned block-by-block several iterations to provide enough time for the network to converge. Due to the self-amplification property of the Hebbian learning, a small learning rate \( \mu \) was selected to prevent the weights from being saturated and in the meantime to provide a good convergence speed. Training takes an average of 3-4 iterations. After training, rows of the resulting weight matrix are the eigenvectors of the auto-covariance matrix of the input image. These eigenvectors are then used for image compression as discussed below.

3. IMAGE COMPRESSION AND RECONSTRUCTION
For this project, image compression consists of three steps, namely, coefficient generation, bit allocation, and coefficient quantization. To generate coefficients for each block, each image block is projected onto the extracted m eigenvectors, yielding m coefficients for each image block. For example, the coefficients \( C_i \) for the block starting at position y, x in the image I are thus given by
\[ C_{y,x}^i = \sum_{p=1}^{m} \sum_{q=1}^{m} W_{i,p+m,q} T_{y+p,x+q} \]  

where \( m \) is the size of the block and \( i=1,2,\ldots,m \).

The coefficients \( C_i \) are then uniformly quantized. The quantization level used for coding each coefficient is approximately proportional to the log of the variance of that coefficient over the whole image. Codewords are subsequently assigned to each quantized coefficient using Huffman coding.

To reconstruct the image from the quantized coefficients, each image block is reformed by adding together all the weights, weighted by their quantized coefficients. For example, the estimated intensities for pixels in a block starting at \( y, x \) are computed as follows:

\[ I_{y+p,x+q} = \sum_{i=1}^{m} W_{i,p+m,q} C_i \]

Repeatedly applying eq(3) to each block, we will recover all pixel values of the image.

4. PERFORMANCE ANALYSIS

To evaluate the performance of the above mentioned neural network, we conducted two experiments. First, the neural net was implemented to compress an image at different bit rates using different block sizes. Figures 1, 2, and 3 give the results. Second, to test the network's generalization capability, the sets of principle components extracted from one image was used for compressing a different but statistically similar image as shown in Figures 4 and 5. Figure 4 gives the test image (the "peppers" image). It was coded using the weight masks obtained from the image in fig. 3. Figure 5 shows the image after it has been reconstructed from quantized coefficients derived from the set of 8 masks in Figure 3.

The reconstructed images in figures 2 and 3 demonstrate that the network can yield a satisfactory image compression performance. As evidenced by both visual inspection and statistical measures (NMSE and SNR), the intelligibility of the reconstructed images improve with the decrease of the block size. However, the image quality improves at the expense of the compression ratio. A balance can be reached between the compression ratio and the loss of image fidelity. If image quality is more important, network should be trained with smaller block sizes, resulting in more bits allocation for each pixel. On the other hand, if high compression is required, bigger block size should be employed. It is clear from figure 5 that quality of the reconstructed image is reasonably good considering the fact that figure 4 was never used to train the neural net and it was coded using the coefficients from another different image. This demonstrates that the neural net trained by GHA has a good generalization capability.

5. CONCLUSIONS

The introduction of the generalized Hebbian learning algorithm for image compression and its satisfying performance demonstrate the value of the technique in this specific area. The beauty of this method is that we do not need to compute the correlation matrix in advance for the compression of image because the eigenvectors are derived directly from the data. For large images, the performance of the neural network in PCA is superior to that of Karhunen-Loeve transform (KLT) method. Further, the generalization capability of the neural nets enables us to quickly compress images using the principle components extracted from other images. This can greatly accelerate the compression process. Finally, like other transform-based coding techniques such as Fourier transform, the reconstructed images from neural nets suffer from blocking effect and blurry. Blocking effect may be minimized by using overlapping blocks.

REFERENCES

Fig. 1 Original 256x256 (8bit) test image (the ‘lena’ girl)

Figure 2. Image of fig. 1 coded at 1.75 bits/pixel. Block size is 2x2. Input/output dimension of NN is 4 and 2. NMSE=0.0061 and SNR=22.135 dB.

Figure 3. Image of fig. 1 coded at 0.422 bits/pixel. Block size is 8x8. Input/output dimension of NN is 64 and 8. NMSE=0.027 and SNR=15.68 dB.

Figure 4. Another test image (the “peppers” image) for testing NN’s generalization capability.
Figure 5. Coding of the image in fig. 4 at 0.609 bits/pixel. The image in fig. 4 is coded using the coefficients derived from image in Fig. 3. Block size is 8x8, NMSE=0.0109, and SNR=19.63.