Variable Rate Self Organizing Neural Networks for Video Compression

K.S. Thyagarajan & Daniel Erickson*
Department of Electrical and Computer Engineering
San Diego State University, San Diego, CA 92182
*GDE Systems Inc.
San Diego, CA.

Abstract

This paper describes a design of Kohonen's self-organizing neural networks as learning vector quantizers (LVQ) to compress video images. Both fixed rate and variable rate LVQs have been designed. For fixed rate LVQs, both full search and tree-structured codebooks are designed. Further, this paper describes the design of variable rate LVQs. Variable rate LVQs, structured as unbalanced trees, are found to provide improved performance of up to 3 dB peak SNR over comparable fixed rate LVQs.

1 Introduction

Artificial neural networks are made up of a large number of primitive processing elements (PEs) that can be programmed in a parallel fashion to perform such tasks as pattern recognition, noise filtering, etc. A distinctive feature of artificial neural networks is that they are model-free function estimators contrary to classical statistical techniques. Therefore, artificial neural nets can be used to solve a wide range of problems with the same network architecture. There are essentially a few basic architectures for neural nets. They can be classified on the basis of the training mechanism used, such as supervised and unsupervised nets. Supervised nets use feedback for learning the correct response, as in back propagation methods. The unsupervised nets, on the other hand, start with undifferentiated data and classify the data into a desired number of classes on the basis of nearest neighbor or K-means clustering procedures. Kohonen's so called *self-organizing* net belongs to this class. Kohonen's net is based on the conjecture that certain regions in the cerebral cortex form topographical maps of the world of stimuli [Kohonen 84]. In image coding, one can precompute templates of pixel patterns (finite in number) based on a suitable algorithm. Then a given image can be coded by simply substituting the templates for the actual pixel patterns in an optimal way. This process is akin to the self-organizing net in the sense of spatial ordering and hence, is highly suitable for image compression [Kohonen 84, Thyagarajan 90]. This paper, therefore , deals with methods to form two-dimensional topographic maps of image pixel patterns using Kohonen's network architecture. This paper describes procedures to design fixed rate and variable rate learning vector quantizers (LVQs). The design of variable rate LVQs utilizes a greedy tree growing algorithm which minimizes the distortion-rate function. The tree is grown to an average depth of a certain level and can be pruned back to the desired average rate of coding using a procedure due to Breiman,Friedman, Olshen, and Stone (BFOS) to further optimize the LVQ. It is found that pruned LVQs out-perform fixed rate counterparts.

The next section describes the architecture of a Self-organizing net. Section 3 describes the network design procedure. Section 4 describes the greedy tree growing and pruning techniques. Section 5 describes the complete results obtained with the LVQs.

2 Self-organizing Network Architecture

This section describes the architecture and the learning process of the self-organizing network. Kohonen's self-organizing net or LVQ is a feedforward neural net with an input layer, a processing layer and no hidden layers. The processing layer is a two-dimensional layer of processing elements (nodes). Each input node is fully connected to every output node. When the net is presented with input patterns, a spatial ordering takes place in the weight vectors

1058-6393/95 $4.00

corresponding to the input patterns. The spatial extent of this ordering is usually limited to a finite set of nodes called the neighborhood. In neural networks, the word learning signifies changes in the weight vectors. Each time an input pattern is presented to the network, the weight vectors are updated according to a predetermined rule. Learning is complete when two consecutive patterns result in no changes in weight values. When the learning is complete, the spatial ordering reflects the probability density function (pdf) of the input patterns presented. The weight vectors of the network form the set of centroids of the partitioning of the input vector space , these are the codewords in the LVQ codebook. Interested reader may refer to [Kohonen 84, Thyagarajan 92, Erickson 92] for the description of the network training algorithm. When encoding an image using the self-organizing network, only the indices of the final weights (codewords) corresponding to the best matching weight vectors are transmitted to obtain compression. The receiver uses a replica of the codebook used at the transmitter, and the received indices to decode the image.

The computational complexity in training an LVQ is found to be [Erickson 92]

$$pN(M+1) \approx PNM \quad Multiplications \qquad (1)$$

and pMN additions, where p equals the number of times the training set is presented, N the number of vectors in the training set and M the number of nodes in the LVQ network. This complexity is the same as that of the algorithm commonly used in VQ codebook construction, namely, the modified Lloyd algorithm also known as the LBG algorithm after Linde, Buzo and Gray [Gersho & Gray 92]. Thus, both the VQ and the LVQ have the same computational complexity if the codebook construction is implemented serially. However, if the algorithm is performed on a parallel architecture machine with M processors, the complexity can be reduced to: pN multiplications and pN additions. The LVQ codebook construction is significantly faster than the LBG algorithm if performed in a parallel fashion. Parallel processing is inherent in neural networks and therefore they are more suitable for vector processing. Design of the LVQ consists first of selecting an appropriate training set, codebook size, appropriate LVQ parameters, and second, determining the weight vectors using the algorithm mentioned above. These parameters include the network learning rate and the neighborhood function. The learning rate is a parameter in the learning equation which controls the rate at which the learning process takes place. The neighborhood function is a parameter which controls the number of network nodes which may adjust their weight values after each training vector is presented. After determining optimal network parameters, the network must then be trained using an appropriate training set [Erickson 92].

3 Greedy-tree Growing and Pruning Algorithms

To improve the performance of a full search LVQ, one should use finer quantization of edge details and coarser quantization of shades since the human visual system is more sensitive to edges than shades. One way to accomplish this is to assign more bits of quantization to edges than to shades so that the average rate is within the budget. Another way to achieve improved performance is to use a variable-rate LVQ which is usually implemented as an unbalanced binary tree. In this case, less frequently used codewords are stored deeper in the tree and thus indexed with more bits. This paper describes constant vector dimension variable rate LVQ. Two methods are investigated: the greedy tree structured LVQ (GTSLVQ) and the pruned greedy tree structured LVQ (PGTSLVQ). Greedy tree growing algorithm for the design of variable rate tree structured vector quantizers have been studied by Riskin [Riskin 84] and Makhoul [Makhoul 85]. Both works utilize the modified Lloyd algorithm to construct the tree-nodes in the codebook. In this study, the self-organizing network is utilized in conjunction with Riskin's greedy-tree algorithm to design variable rate LVQs. According to Riskin, et al, the decision to split a tree node depends on a *goodness of split* criterion. The node that results in the largest decrease in distortion due to a given rate increase is split. This is, therefore, optimal from a rate-distortion view point while the method due to Makhoul is not. For instance, let t_R and t_L represent the left and right children of node t, and let α denote the change in distortion to rate ratio. Riskin shows that α is given by:

$$\alpha = -\Delta D/\Delta R = p(t)[d(t) - p_L * d(t_L) - p_R * d(t_R)]/p(t) \qquad (2)$$

where ΔD and ΔR are the changes in distortion and rate, respectively, resulting from splitting node t, p(t) is the probability of node t, p_L and p_R are the proportion of samples in t that go to t_L and t_R respectively, that is:

$$p_L = p(t_L)/p(t) \qquad (3)$$

$$p_R = p(t_R)/p(t) \qquad (4)$$

and d(t) is the distortion at node t. The node that achieves the maximum value for α should be split further. The algorithm will proceed by designing new nodes one at a time, and splitting the nodes with the best distortion-rate tradeoff until the desired average coding rate is reached. At this point the tree growing procedure is terminated. For full details on greedy tree growing algorithm, refer to [Riskin 84, Gersho & Gray, 92].

3.1 Pruning of LVQ

Greedy Tree LVQs (GTSLVQ) as described above, are in general not optimal for a given average rate since they are constructed without taking into account the distortion due to the entire tree. That is, only the current state of the tree is considered when splitting any node. A GTSLVQ can be pruned to obtain an optimal tree structured LVQ, called a Pruned Greedy Tree LVQ (PGTSLVQ). The BFOS algorithm as referred to in this work is a generalization of an algorithm by Breiman, Friedman, Olshen and Stone (hence the name BFOS), [Breiman 84] and is used for optimally pruning a tree by trading off the number of leaves for distortion in source coding of signals [Gersho & Gray 92].

Starting with the greedy tree T, branches are successively pruned off, based upon the rate-distortion of their subtrees. Branches which, by removing, result in the smallest increase in distortion to a given decrease in the rate are pruned off. The algorithm continues until the desired average data rate is obtained. The complexity of the entire algorithm [Gersho & Gray 92] has been shown to be: O(NloghN) or for a binary tree: O(Nlog2N) where N= Number of nodes in tree, h= degree of tree (h=2 for a binary tree).

4 Experimental Results

The performances of fixed rate full and tree structured LVQs were reported earlier in another paper by the authors [Thyagarajan 90, Erickson 92]. We summarize salient results on fixed rate LVQs. Full search and tree structured LVQs were designed for vector dimensions of 64 (8x8 blocks), 16 (4x4 blocks) and 4 (2x2 blocks). At a given encoding bit rate the LVQ performance is always best at as large a vector size as possible. However, for a given encoding rate, a larger vector size implies a larger codebook and thus more encoding complexity. For example, at an encoding rate of 0.25 bit/pixel an LVQ with vector dimension of 16 has only 16 codevectors (indexed by 4 bits) while at the same encoding rate an LVQ with vector dimension 64 has 65,536 codevectors (indexed by 16 bits). Thus again there is a tradeoff of LVQ performance and LVQ encoding complexity

We now report the results obtained from two types of variable rate LVQs: Greedy Tree Structured LVQ (GTSLVQ) and Pruned Greedy Tree Structured LVQ (PGTSLVQ). GTSLVQs were designed and tested for vector sizes 4 (2x2 blocks), 16 (4x4 blocks) and 64 (8x8 blocks). The observed performance (PSNR) of the GTSLVQ for vector sizes 4, 16 and 64 is shown as a function of bit rate in figure 1. The number of tree leaves (codewords) in the GTSLVQ as a function of average rate shows that as average rate increases, the number of codewords grows at about the same rate as does a full search LVQ. For example, GTSLVQs (vector size 16) with an encoding rates of 0.5 and 0.75 bit/pixel have 223 codewords and 3510 codewords respectively, while full search LVQs of the same rate have 256 and 4096 codewords respectively. However, the memory requirement of a GTSLVQ includes both the codewords (tree leaves) and the tree's interior nodes, hence the GTSLVQ like the TSLVQ requires more memory than the full search LVQ for the same encoding rate.

The generalized BFOS tree pruning algorithm described above is used to prune the GTSLVQs to generate Pruned Greedy Tree Structured LVQs (PGTSLVQ). PGTSLVQs were designed and tested for vector sizes 4 (2x2 blocks), 16 (4x4 blocks) and 64 (8x8 blocks). The vector size 4 PGTSLVQs were pruned from a GTSLVQ of rate 2.5 bit/pixel, the vector size 16 PGTSLVQs were pruned from a rate 0.75 bit/pixel GTSLVQ and the vector size 64 PGTSLVQs were pruned from a GTSLVQ of rate 0.1875 bit/pixel. The PGTSLVQs rate-distortion performance is superior to the GTSLVQ from which it is derived. Figure 2 shows the observed performance of the PGTSLVQ as a function of encoding rate. The number of tree leaves (codewords) in the PGTSLVQ as a function of average rate for vector sizes 4, 16 and 64 again shows that for a given encoding rate, the number of codewords for a PGTSLVQ is significantly more than for a full search LVQ. For example , for encoding rates of 0.5 bit/pixel (vector size 16), a full search LVQ has 256 codewords while a PGTSLVQ of the same encoding rate has 1918 codewords. Thus, for a given rate, the PGTSLVQ has significantly more codewords than either the GTSLVQ or the full search LVQ, and hence has larger memory requirements. Among variable rate LVQs, the PGTSLVQ thus has superior performance but requires more memory than the GTSLVQ. The

full search LVQs, GTSLVQs and PGTSLVQs all outperform the TSLVQs, this is consistent with previous work on VQ. At lower bit rates, the full search LVQ outperforms the GTSLVQ. At higher rates, the GTSLVQ performance approaches the full search LVQ. The PGTSLVQ outperforms all other LVQs. In summary at low bit rates, where the variable rate trees are not well developed, the advantage of variable rate LVQs is not significant. Here PGTSLVQs perform only slightly better than full search LVQs, the performance of GTSLVQs is below full search LVQs and lowest performance is by TSLVQs. At higher rates, where the variable rate trees are more fully developed and contain more codewords, variable rate LVQs have a more significant advantage. Here, PGTSLVQs substantially outperform full search LVQs, GTSLVQs performance approaches that of the full search LVQs, and again the lowest performance is by TSLVQs.

The visual quality due to the variable rate coding may be seen from figures 4 and 5. Figure3 is the original image while figures 4 and 5 show images compressed by GTSLVQ and PGTSLVQ respectively. It is clearly seen from figure 5 that PGTSLVQ shows much smoother texture and better overall visual quality than the GTSLVQ. However, the visual quality of the GTSLVQ encoded image was seen to be slightly better than the full search LVQ particularly in shade areas. It was further noticed that the full search LVQ encoded image appeared visually somewhat better in dark (low intensity) areas than either of the variable rate LVQs (less patchy). When building a variable rate LVQ , distortion/rate is minimized but since low intensity areas generally produce low distortion, these areas are not refined to the extent that higher intensity areas are. In spite of this, the superiority of variable-rate LVQ over fixed-rate LVQ is apparent.

References

[1] Kohonen, T., *Self-organization and Associative Memory,* Springer-Verlag,Berlin, 1984.

[2] Gersho, A. and Gray, R., *Vector Quantization and Signal Compression,* Kluwer Academic Publishers, 1992.

[3] Riskin, E.A. and Gray, R., *A greedy tree growing algorith m for thedesign of variable rate vector quantizers,* IEEE Trans. Sig. Process., Nov. 1984.

[4] Makhoul, J.S., et al, *Vector Quantization in Speech and Coding,* Proc. IEEE, vol. 73, No.11, November 1985.

[5] Erickson, D. and Thyagarajan, K.S., *A neural network approach to image compression,* Proc. IEEE ISCAS, May 1992.

[6] Breiman, L., et al, *Classification and Regression Trees,* Wadworth,Belmont, 1984.

[7] Thyagarajan, K.S. and Eghbalmoghadam, A., *Design of a vector quantizer using a neural network,* AEU, Electronics & Communications,1990.

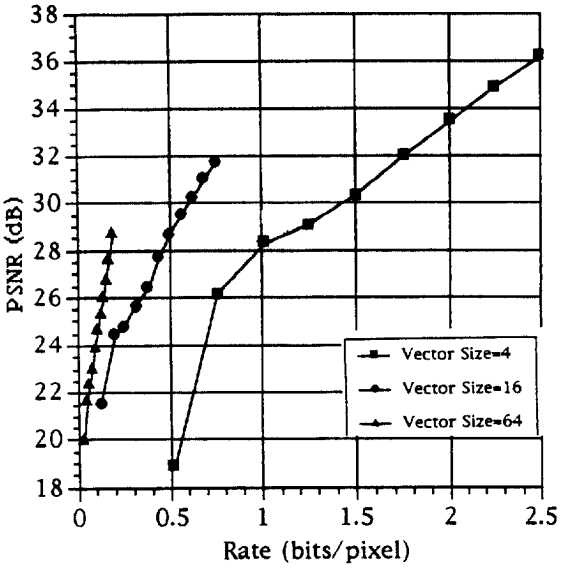

Figure 1: PSNR vs bit rate for GTSLVQ

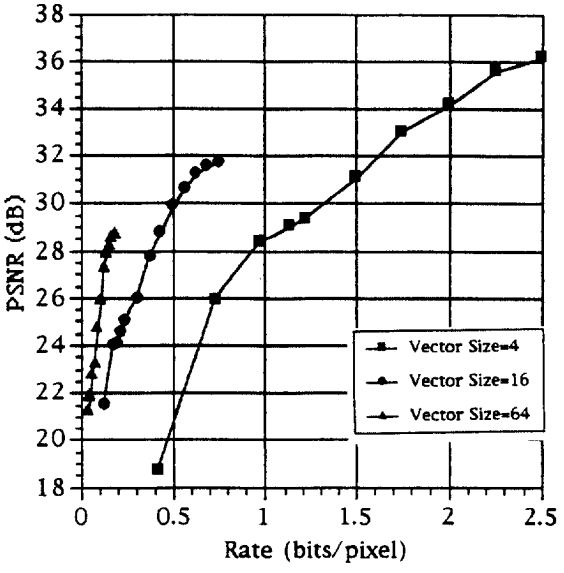

Figure 2: PSNR vs bit rate for PGTSLVQ

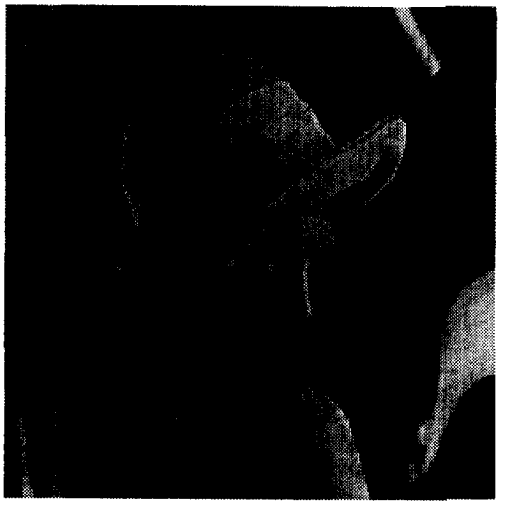

Figure 3: Original Lena Image

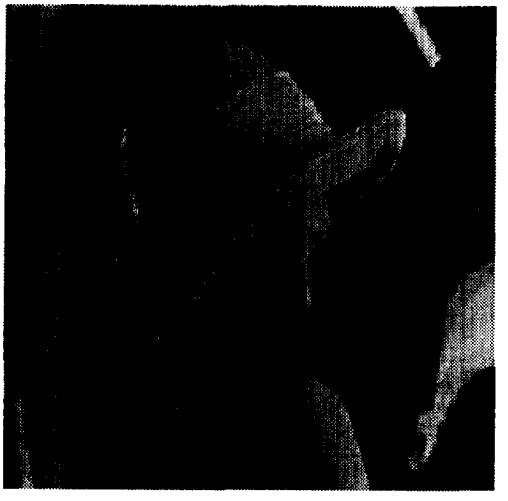

Figure 4: GTSLVQ Compressed Image at 0.5 b/pixel, 4x4 blocks, PSNR = 28.38 dB

Figure 5: PGSTLVQ compressed Lena Image @ 0.5 b/pixel, 4x4 blocks, PSNR = 29.17 dB