A Switching Predictor for Lossless Image Coding [*]

Lih-Jen Kau and Yuan-Pei Lin
Dept. Elec. and Control Engr., National Chiao Tung Univ.
1001 Ta Hsueh Road, Hsinchu, Taiwan, R.O.C.
u8912801@cc.nctu.edu.tw, ypl@cc.nctu.edu.tw

Abstract – *In this paper, we propose a switching adaptive predictor (SWAP) with automatic context modeling for lossless image coding. In the SWAP system, two predictors are used. For areas with edges, estimates of coding pixels are obtained using texture context matching (TCM). For all other areas, an adaptive neural predictor (ANP) is used. The SWAP encoder switches between the two predictors ANP and TCM depending on the neighborhood of the coding pixel. The switching predictor allows statistical redundancy to be removed effectively. On the other hand, it is known that prediction can be further refined using error compensation. For this, we propose the use of a modified fuzzy clustering, which leads to a modeling of errors that adapts itself to the input statistics. Experiments show that the proposed context clustering is very useful in modeling error for prediction refinement. Comparisons of the proposed system to existing state-of-the-art predictive coders will be given to demonstrate its coding efficiency.*

Keywords: lossless image compression, context modeling, adaptive prediction, neural network, fuzzy clustering.

1 Introduction

Many of the recent advances in lossless image coding are based on predictive coding with context modeling [1]-[3][6]-[9][11]. The CALIC coding system [11], a state-of-the-art lossless predictor proposed for JPEG-LS, is a gradient adjusted predictor (GAP). Based on the gradient of neighboring pixels, one out of a set of seven predictors is chosen. The LOCO-I coder [9], an algorithm motivated by CALIC, uses a median edge detector (MED) to choose one of three predictors for current prediction. The LOCO-I system has been standardized into JPEG-LS.

[*]This work was supported in parts by the National Science Council, Taiwan, R. O. C., under NSC 90-2213-E-002-097 and 90-2213-E-009-108, Ministry of Education, Taiwan, R. O. C, under Grant # 89E-FA06-2-4, and the Lee and MTI Center for Networking Research.

In [8], Slyz invents the idea of estimating coding pixel based on context matching. For each coding pixel, a causal area of dimensions 30 by 30 is used for context matching. From this area, 11 pixels are chosen and averaged to form an estimate of the coding pixel. The histogram of the prediction errors corresponding to the 11 candidates is calculated. By computing the variance of the 11 prediction errors, one out of a set of 37 Laplacian distribution which matches the histogram the best is chosen. The prediction error of the coding pixel is then coded using a conditional arithmetic coder corresponding to the chosen Laplacian model. In [11], Wu demonstrated that prediction error can be further refined through error compensation. The compensated error has a narrower histogram and hence a lower first order entropy. In CALIC [11], 576 compound contexts are used for error modeling. In the LOCO-I system [9], 365 contexts are used for error modeling.

In the context of optimal predictors, the minimum mean square error estimate of Y given observations $X_1, X_2, \cdots, X_n$ is $E\{Y|X_1, X_2, \cdots, X_n\}$, generally a nonlinear function. There have been many results using neural networks as nonlinear estimators [1][2][6]. Dony and Haykin proposed neural approaches to predictive image compression in [1]. Neural predictors based on multi-layered perceptrons are used in [2][6]. In [2][6], adaptive prediction is achieved by updating the network weights using the prediction errors of coded pixels. Nonlinear predictors using neural networks, though perform well in slowly varying areas, can have large prediction error around boundaries [2]. The result can be improved using additional hidden layers or hidden neurons, but this incurs a drastic increase in complexity [4].

In this paper, we propose a prediction scheme for lossless image coding, called **SWAP** (SWitching Adaptive Predictor). The coder switches between two predictors: ANP (Adaptive Neural Predictor) and TCM (Texture Context Matching). For pixels around edges, TCM is used; otherwise ANP is used. When TCM is used, pixels with contexts similar to that of current pixel are averaged for the current prediction. The ANP is a three-layered neural network. We will see that the TCM pro-

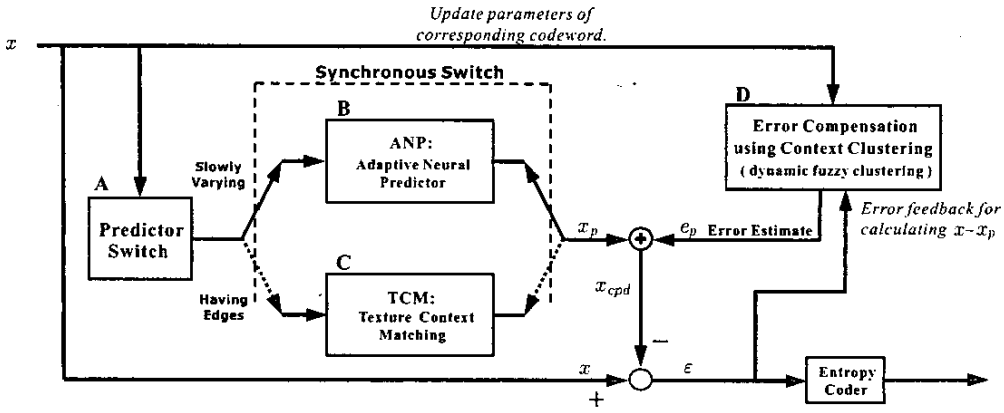

Figure 1: Proposed **SWAP** System.

vides a nice complement of ANP. Very good predictions can be obtained for pixel around edges, where conventional predictors tend to have large prediction errors.

For prediction error refinement, automatic error modeling is achieved in SWAP using a modified version of the UFCL (Unsupervised Fuzzy Competitive Learning) algorithm in [5]. It does not require training data to be available all at once, and it can be used for sequential encoding. Furthermore, the number of clusters is variable. The combination of a switching structure and automatic error modeling renders the proposed SWAP coder highly adaptable and very low bit rates can be achieved.

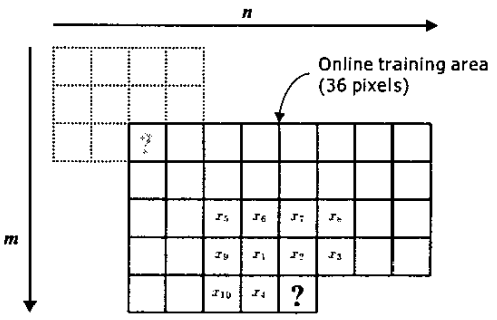

Figure 2: **The texture context of the coding pixel and online training regions for ANP.**

2 Proposed SWAP System

The proposed SWAP system is made up of four components as shown in Fig. 1. It has two predictors, ANP and TCM. The ANP is an adaptive three-layered back propagation network that is updated on the fly using causal pixels as training patterns. The TCM looks for pixels in a predefined causal area that have textures similar to that of the coding pixel and using these pixels to estimate the coding pixel. The "*Predictor Switch*" block (Fig. 1) determines whether the current pixel is around an edge. If it is, TCM will be used as the predictor; otherwise ANP will be used. With the "*Predictor Switch*" block, the encoder switches automatically between ANP and TCM.

The prediction is further refined through error compensation. That is, the output x_p from ANP or TCM is added by e_p to get a compensated prediction $x_{cpd} = x_p + e_p$. The amount of compensation e_p is determined through error modeling based on a modified UFCL (Unsupervised Fuzzy Competitive Learning) [5]. The error signal $\varepsilon = x - x_{cpd}$ can then be entropy encoded using conditional arithmetic coding [10]. In addition, the encoder uses only causal pixels for estimating the coding pixels; no additional side information needs to be transmitted. It is noted that the proposed SWAP coder is symmetric, meaning that the decoder has the same predictor switch as the encoder, and perform ANP/TCM prediction and error compensation just like the encoder. Details of the individual components of the system are introduced in subsequent sections.

3 Predictor Switch

This section introduces the switching criterions of the proposed SWAP system. The TCM is used in nonuniform regions having edges while ANP predictor is used in all other cases. We observe that the variance of an area that contains an edge is usually large. Furthermore, the histogram of such an area tends to have 2 peaks, one on each side of the mean value. We will use these two observations to determine whether ANP or TCM should be used. We define the *texture context* κ of a coding pixel as the collection of the ten causal pixels $x_1, x_2, \cdots, x_{10}$ in Fig. 2,

$$\kappa = \{x_1, x_2, \cdots, x_{10}\}. \quad (1)$$

The mean $\bar{x}$ and variance σ^2 of the *texture context* are respectively,

$$\bar{x} = \frac{1}{10}\sum_{i=1}^{10} x_i, \qquad \sigma^2 = \frac{1}{10}\sum_{i=1}^{10}(x_i - \bar{x})^2. \quad (2)$$

The ten pixels can be divided into two groups, the pixels with gray levels higher than $\bar{x}$ in one group κ_h and the rest in another κ_l. We also compute the mean $\bar{x}_h$ and variance σ_h^2 of those pixels in κ_h,

$$\bar{x}_h = \frac{1}{N_h}\sum_{x_j \in \kappa_h} x_j, \qquad \sigma_h^2 = \frac{1}{N_h}\sum_{x_j \in \kappa_h}(x_j - \bar{x}_h)^2, \quad (3)$$

where N_h is the number of elements in κ_h. Similarly, we compute the mean $\bar{x}_l$ and variance σ_l^2 of those pixels in κ_l,

$$\bar{x}_l = \frac{1}{N_l}\sum_{x_j \in \kappa_l} x_j, \qquad \sigma_l^2 = \frac{1}{N_l}\sum_{x_j \in \kappa_l}(x_j - \bar{x}_l)^2, \quad (4)$$

where N_l is the number of elements in κ_l.

A histogram with 2 peaks is likely to have a large σ^2 but small σ_h^2 and σ_l^2. We determine whether the coding pixel is around an edge if the following 2 conditions are satisfied,

$$\sigma^2 > \gamma_1, \quad \text{and} \quad \frac{\sigma^2}{0.01 + \sigma_h^2 + \sigma_l^2} \geq \gamma_2, \quad (5)$$

where 0.01 is added so that the denominator of (5) does not become 0 when σ_h^2 and σ_l^2 are both zero. We will use $\gamma_1 = 25$ and $\gamma_2 = 10$ in this paper. The TCM predictor is used whenever the conditions in (5) are satisfied; otherwise the ANP predictor is used.

4 Adaptive Neural Predictor

The ANP predictor is a three-layered back propagation neural network. There are 10 neurons in the input layer, 5 neurons in the hidden layer, and 1 neuron in the output layer. The output is the prediction value of the coding pixel. The number of hidden neurons is chosen empirically. We have found 5 to be a proper choice. Our experiments show that increasing the number of hidden neurons leads to only marginal improvement in entropy or bit rates, but will increase complexity dramatically.

We use the ten causal pixels $x_1, x_2 \cdots, x_{10}$ (Fig. 2) in *texture context* (1) as the predictor inputs. The ANP predictor adapts itself to the varying statistics by applying gradient descent method continuously on the fly with the 36 causal pixels in Fig. 2 as training patterns. The online updating process is performed by iterative learning on the 36 training pixels. The learning rate is set to 0.9 within the first three training cycles to avoid being trapped in the local minima and is set to 0.1 after that to avoid the oscillation problem [4][6]. In addition, a momentum term 0.5 is used to accelerate convergence speed [4]. Updated weights are used for current prediction and passed on to the next coding pixel as initial weights.

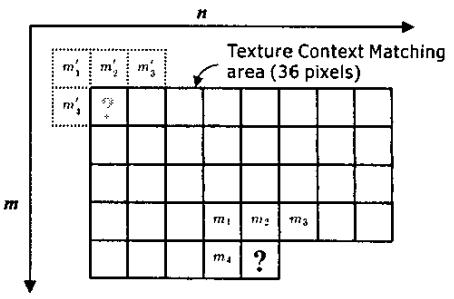

Figure 3: **The area for Texture Context Matching.**

5 Texture Context Matching

The ANP predictor can achieve very good prediction in slowly varying areas. However, for areas containing edges, convergence of the network weights is slow and the prediction error is large. For such areas, the encoder switches to *Texture Context Matching* and estimate the coding pixel using pixels with similar contexts. To reduce the complexity of texture matching, we will use a *shortened texture context.*

Shortened Texture Context. The 4 pixels marked by m_1, m_2, m_3, m_4 as shown in Fig. 3 constitute the *shortened texture context* for TCM. They are the same pixels x_1, x_2, x_3, x_4 in section 3 but are renamed as $m_1, \cdots m_4$ for the convenience of explanation. As similar contexts are most likely to appear in the vicinity of the coding pixel, there is no need to perform an exhaustive search or near global search in the causal area. The pixels to be used for context matching is shown in Fig. 3, the same 36 pixels used for the online training area of ANP.

We first calculate the *normalized shortened texture context,*

$$\left\{\frac{m_1}{l_m}, \frac{m_2}{l_m}, \frac{m_3}{l_m}, \frac{m_4}{l_m}\right\}, \quad (6)$$

where l_m is the length or root mean square (rms) value of the *shortened texture context.* For each of 36 pixels in the context matching area, we compute its normalized shortened texture context as in (6). We calculate the Euclidean distance $d_{mm'}$ between the *normalized shortened texture context* of the current pixel and that of each of the 36 pixels to be matched. We find the 2 pixels with the two shortest Euclidean distances. Suppose the values of these 2 pixels are y_1 and y_2, where y_i has been normalized by the length of its shortened texture context. The coding pixel value x is estimated by

$$x_p = l_m\left(\frac{y_1 + y_2}{2}\right). \quad (7)$$

For coding pixels on the boundaries of the image, matching areas are smaller. Only one pixel in the matching area is chosen to be the candidate for coding pixel estimation.

6 Error Compensation using Context Clustering

The prediction errors $x - x_p$ in the proposed SWAP system are further refined by learning from previous predictions. In the SWAP system, adaptive error modeling is achieved by designing codebooks using a modified fuzzy clustering. The context is dynamically generated and modified in the coding process. Also, the number of contexts is not fixed and will depend on the statistics of the image to be encoded.

Let e_i be the uncompensated prediction error of x_i in Fig. 2 for $i = 1, 2, 3, 4$. We define the *compound context* vector $v^{(t)}$ of a coding pixel as

$$v^{(t)} = \{x_1, x_2, \cdots, x_{10}, e_1, e_2, e_3, e_4\}, \tag{8}$$

where x_i, $i = 1, \cdots 10$ are as shown in Fig. 2 and e_i, $i = 1, \cdots 4$ are the uncompensated prediction errors corresponding to $x_1, \cdots x_4$ respectively. The pixels are encoded sequentially in raster scan order and the superscript (t) in $v^{(t)}$ denotes pixel index. We have incorporated prediction errors in codebook designs, because the amount to be compensated is likely to be related to the prediction errors of neighboring pixels. Furthermore, the compound context vector $v^{(t)}$ of a coding pixel is likely to be related to existing clusters with some membership degrees. Therefore, it is dynamically assigned to existing clusters or to a new cluster. In conventional fuzzy K-Means clustering, the vectors to be classified are available all at once, and the number of clusters is fixed [4]. Here, the compound context vectors appear sequentially. Therefore, the UFCL (Unsupervised Fuzzy Competitive Learning) algorithm [5] which is suitable for sequential input vector classification is used for the clustering process with some modifications. The number of clusters is now variable. Furthermore, we will propose a learning rate that has the desired property of approaching zero when an optimal classification is achieved [5]. In this case, the cluster updating formula is in the form of a weighted summation.

6.1 Dynamic Fuzzy Clustering

Assume we have K clusters in the codebook currently. The i^{th} codeword or i^{th} cluster center will be denoted by $c_i^{(t)}$. For an incoming compound context vector $v^{(t+1)}$ of coding pixel, the distance $d_i^{(t+1)}$ between $v^{(t+1)}$ and the existing i^{th} cluster center $c_i^{(t)}$ is calculated by

$$d_i^{(t+1)} = \left\| v^{(t+1)} - c_i^{(t)} \right\|^2, \text{ for } i = 1, 2, \cdots, K, \tag{9}$$

where $\|\cdot\|$ denotes the vector 2-norm. If the minimum distance value $d_{mim} = \min_{i=1..K} \left\{ d_i^{(t+1)} \right\}$ is greater than a predefined threshold $\beta = 15000$, a new cluster is added. Therefore, the number of codewords is increased to $K+1$, and the new cluster center $c_{(K+1)}^{(t+1)} = v^{(t+1)}$. If $d_{min} < \beta$, $v^{(t+1)}$ is used to update existing clusters based on the modified UFCL algorithm. The membership degree $A_i^{(t+1)}$ of $v^{(t+1)}$ to the i^{th} cluster is defined as [2][5],

$$A_i^{(t+1)} = \left\{ \sum_{j=1}^{K} \left[\frac{d_i^{(t+1)}}{d_j^{(t+1)}} \right]^{\frac{1}{m-1}} \right\}^{-1}, \quad \text{for } i = 1, 2, \cdots, K, \tag{10}$$

where the *exponential weight* m, which controls the degree of fuzziness, is chosen to be 1.25. The membership degree $A_i^{(t+1)}$ is inversely proportional to the distance between $v^{(t+1)}$ and existing cluster centers. We define $S_i^{(t)}$, the *accumulated weight* of the i^{th} cluster as [2],

$$S_i^{(t)} = \sum_{n=1}^{t} (A_i^n)^m, \quad \text{for } i = 1, 2, \cdots, K. \tag{11}$$

Note that $S_i^{(t)}$ is very similar to the membership degree summation of vectors with respect to the same cluster in a conventional fuzzy partition matrix. Existing cluster centers are updated by [5],

$$c_i^{(t+1)} = c_i^{(t)} + \lambda^{(t+1)} (A_i^{(t+1)})^m (v^{(t+1)} - c_i^{(t)}), \tag{12}$$

where $\lambda^{(t+1)}$ is the learning rate. We will choose

$$\lambda^{(t+1)} = \frac{1}{S_i^{(t)} + (A_i^{(t+1)})^m}. \tag{13}$$

The resulting learning rate $\lambda^{(t+1)}$ will get smaller as coding proceeds and will approaches zero when an optimal classification is reached. Using (13), (12) can be re-written as,

$$c_i^{(t+1)} = \frac{S_i^{(t)} \cdot c_i^{(t)} + (A_i^{(t+1)})^m \cdot v^{(t+1)}}{S_i^{(t)} + (A_i^{(t+1)})^m}. \tag{14}$$

6.2 Error Estimate

The value e_p to be used in compensating the prediction error of the coding pixel is given by,

$$e_p = \sum_{i=1}^{K} A_i^{(t+1)} \cdot e_i^{(t)}, \tag{15}$$

where $e_i^{(t)}$ is the sample mean of prediction errors in the i^{th} cluster. We update $e_i^{(t)}$ using [5],

$$e_i^{(t+1)} = e_i^{(t)} + \lambda^{(t+1)} (A_i^{(t+1)})^m (x^{(t+1)} - x_p^{(t+1)} - e_i^{(t)}). \tag{16}$$

Similarly, using (13), (16) can be written as,

$$e_i^{(t+1)} = \frac{S_i^{(t)} \cdot e_i^{(t)} + (A_i^{(t+1)})^m \cdot \left(x^{(t+1)} - x_p^{(t+1)} \right)}{S_i^{(t)} + (A_i^{(t+1)})^m}. \tag{17}$$

We now form a more refined prediction $x_{cpd} = x_p + e_p$, where x_p is the output of predictor ANP or TCM.

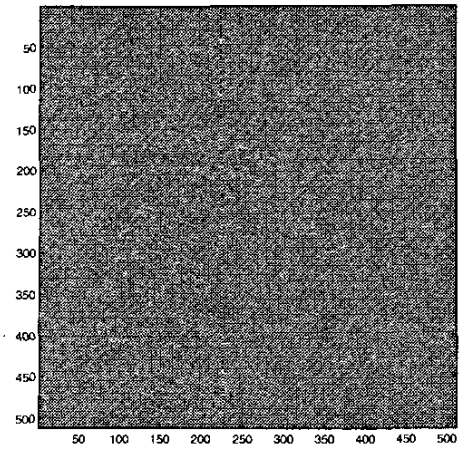

Figure 4: **Image of compensated prediction errors for "Lennagrey".**

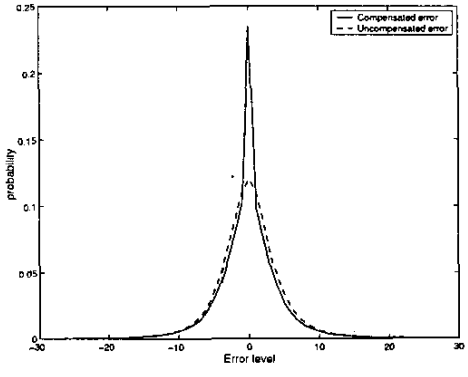

Figure 5: **Histogram of prediction errors for "Lennagrey".**

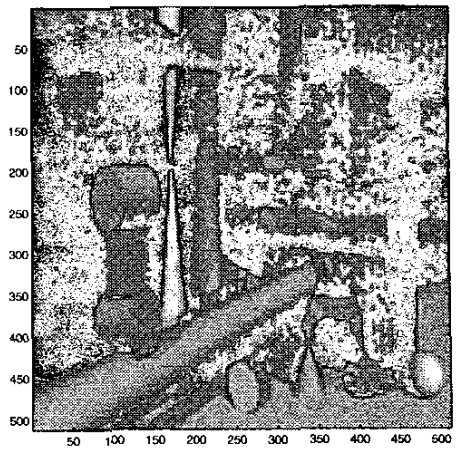

Figure 6: **The image "Shapes".**

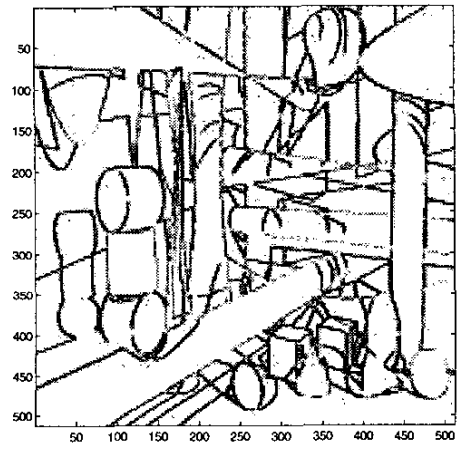

Figure 7: **Pixels for which TCM are used in the image "Shapes".**

7 Experiments

Experiment results of the proposed SWAP coder and comparisons to existing state-of-the-art linear and non-linear predictors are given in this section. All the test images used in the experiments are from the website of TMW[1] [7]. For the image "Lennagrey", we show in Fig. 4 and Fig. 5 respectively the compensated prediction error and the corresponding histogram. The statistical redundancy is removed efficiently as can be seen in Fig. 4 and Fig. 5. The usefulness of the proposed automatic context modeling for error refinment can be best observed from Fig. 5, in which the histogram of prediction errors with and without error compensation are shown. In Fig. 5, the first order entropy for compensated errors is 3.98 bits and 4.22 bits for uncompensated errors.

The usefulness of the proposed TCM predictor can be demonstrated through the following experiment. We use the image "Shapes", which is an artificial image with many edges and lines (Fig. 6). The pixels for which TCM are used are marked in Fig. 7. We can see from Fig. 7 that the "*Predictor Switch*" box has successfully picked out the pixels around edges. The histogram of uncompensated prediction errors for those pixels using TCM is shown in Fig. 8. For comparison, we also show in Fig. 8 the histogram of uncompensated prediction error if ANP were used for those pixels instead. The histogram with TCM is much narrower than that with ANP; TCM has a smaller prediction error than ANP does around edges. Indeed, the entropies corresponding to the 2 histograms in Fig. 8 are respectively 2.97 bits (TCM) and 5.38 bits (ANP).

Table 1 gives the actual bit rates by JPEG-LS [9], CALIC [11], EDP [3] and TMW [7] for of a set of 14

[1] http://www.csse.monash.edu.au/ bmeyer/tmw/

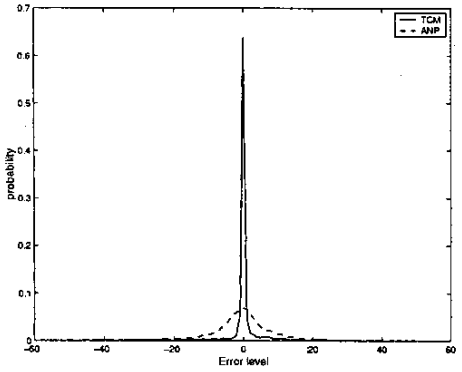

Figure 8: **Histogram of prediction errors for the pixels shown in Fig. 7.**

Table 1: Comparisons with existing lossless image coders (in bits/sample).

Item	Image	First Order Entropy	SWAP Bit Rate	JPEG-LS Bit Rate [7]	CALIC Bit Rate [6]	EDP Bit Rate [11]	TMW Bit Rate [13]
1	Airplane	6.7058	3.57	3.82	3.74	N/A	3.6
2	Baboon	7.3579	5.85	6.04	5.88	5.81	5.73
3	Balloon	7.3258	2.52	2.9	2.83	N/A	2.66
4	Barb	7.4664	4.14	4.69	4.32	4.11	4.09
5	Barb2	7.4838	4.59	4.69	4.53	4.52	4.38
6	Boats	7.095	3.64	3.93	3.83	3.8	3.61
7	Camera	7.009	4.37	4.31	4.19	N/A	4.1
8	Couple	6.3902	3.78	3.7	3.61	N/A	3.45
9	Gold Hill	7.6232	4.3	4.48	4.39	4.39	4.27
10	Lena	7.594	4.37	N/A	4.48	4.4	4.3
11	Lenna Grey	7.4473	3.96	4.24	4.11	4.02	3.91
12	Noise square	5.723	5.15	5.68	5.44	N/A	5.54
13	Peppers	7.5924	4.27	4.29	4.42	4.35	4.25
14	Shapes	6.7395	1.55	1.21	1.14	N/A	0.76

test images obtained from the website of TMW [7]. Results listed in the last four columns of Table 1 are from LOCO-I [9], TMW [7] and EDP [3]. The compensated prediction errors are coded using a conditional arithmetic coder adapted from [10]. Table 1 shows that SWAP has lower bit rates than JPEG-LS in 11 of 14 test images and outperforms CALIC [11] in 10 of 14 test images. Encouragingly, SWAP achieve bit rates lower than the highly complex TMW in 3 images "Airplane","Balloon" and "Noise square".

8 Conclusion

In this paper, a new coder called SWAP is proposed. The SWAP system switches between two predictors ANP and TCM automatically. The ANP predictor, making nonlinear prediction using a neural network, performs very well in slowly varying areas. The TCM provides a very nice complement of ANP predictor. As the simulation example has demonstrated, the TCM predictor can achieve very good prediction around edges, where ANP predictor tend to have larger prediction errors. For error refinement, automatic context modeling is achieved using the modified UFCL. The usefulness of proposed SWAP system is demonstrated through the reduction of first order entropy and actual bit rate in tested images.

References

[1] R. D. Dony, and S. Haykin, "Neural network approaches to image compression," *Proceedings of the IEEE*, Vol. 83, No. 2, pp. 288-303, Feb. 1995.

[2] Lih-Jen Kau, "Adaptive predictor with dynamic fuzzy k-means clustering for lossless image coding," *Proceedings of the IEEE International Conference on Fuzzy Systems*, pp. 944-949, 2003.

[3] Xin Li, and Michael T. Orchard, "Edge-directed prediction for lossless compression of natural images," *IEEE Transactions on Image Processing*, Vol. 10, No. 6, pp. 813-817, June 2001.

[4] Chin-Teng Lin, and C. S. George Lee, *A Neural-Fuzzy Synergism to Intelligent Systems*, Prentice Hall, 1996.

[5] Carl G. Looney, *Pattern Recognition Using Neural Networks*, Oxford, pp. 250-256, 1997.

[6] S. Marusic, and G. Deng, "A neural network based adaptive non-linear lossless predictive coding technique," *Proceedings of the International Symposium on Signal Processing and Its Applications*, Vol. 2, pp. 653-656, 1999.

[7] B. Meyer, and P. E. Tischer, "TMW-A new method for lossless image compression," *Proceedings of International Picture Coding Symposium*, Berlin, Germany, Oct. 1997.

[8] M. J. Slyz, and D. L. Neuhoff, "A nonlinear VQ-based predictive lossless image coder," *Proceedings of Data Compression Conference*, pp. 304-310, Mar. 1994.

[9] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS," *IEEE Trans. Image Processing*, Vol. 9, No. 8, pp. 1309-1324, Aug. 2000.

[10] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, Vol. 30, No. 6, pp. 520-540, June 1987.

[11] X. Wu, and N. Memon, "Context-based, adaptive, lossless image coding," *IEEE Trans. Communications*, Vol. 45, No. 4, pp. 437-444, April 1997.