Adaptive Image Pyramidal Representation

Valeriy Cherkashyn\textsuperscript{1}, Roumen Kountchev\textsuperscript{2}, Dong-Chen He\textsuperscript{1}, Roumiana Kountcheva\textsuperscript{3}

\textsuperscript{1} CARTEL, Université de Sherbrooke, Quebec, J1K 2R1, Canada; valeriy.cherkashyn@usherbrooke.ca; dong-chen.he@usherbrooke.ca
\textsuperscript{2} T&K Engineering, Sofia, 1712, Pob.12, Bulgaria, kountcheva_r@yahoo.com
\textsuperscript{3} Dept. of RCVT, Technical University-Sofia, Sofia 1000, Kl. Ohridski 8, Bulgaria, rkountch@tu-sofia.bg

Abstract. New adaptive method for image compression based on pyramid decomposition with neural networks with error back propagation (BPNN) is presented in this paper. The processed image is divided in blocks and then each is compressed in the space of the hidden layers of 3-layer BPNNs, which build the so-called Inverse Difference Pyramid. The results of the new method modeling are compared with these, obtained using the image compression standards JPEG and JPEG2000.

Key words: Image representation and compression, Image pyramidal decomposition, Neural networks.

I. INTRODUCTION

The demands towards the efficiency of the methods for image representation in compressed form are getting higher together with their wide application and this is the basis for further elaboration and development. The classic methods [1-6] could be classified in the following main groups: deterministic and statistical orthogonal linear transforms (DFT, DCT, WHT and KLT, SVD, PCA correspondingly); discrete wavelet transforms (DWT, Embedded Zerotree Wavelet, etc.); transforms based on prediction (LP, Adaptive LP, etc.); vector quantization (VQ, Adaptive VQ, Multistage Predictive VQ, etc.); fractal transforms (IFS, Quadtree Partitioned IFS, etc.) and decompositions, based on various pyramids, such as GP/LP, RLP, RSP/RDP, IDP, etc.

The analysis of these methods shows that for image compression are usually used deterministic orthogonal transforms and linear prediction with fixed coefficients [1,4,6]. For example, in the standard JPEG is used discrete cosine transform (DCT) and linear prediction (LP), and the standard JPEG2000 [3] is based on the discrete wavelet transform (DWT). The transformed image data are compressed with entropy coding [6], implemented as a combination of various methods for lossless coding (RLC, AC, LZW, etc.). The transforms DCT, DWT and LP are deterministic and the values of the coefficients of their matrices do not depend on the processed image content. For this reason, the compression efficiency is low when the correlation between the image content and the corresponding transform functions is low. The statistical transforms [1,4] are more efficient than the deterministic ones, but they have higher computational complexity. Similar disadvantage have the fractal methods [2,6], which together with this are not enough efficient when images with unclear texture regions are processed. The famous pyramidal image decompositions are usually implemented with digital filters with fixed
decimation and interpolation coefficients [2,4] or use some kind of transform, such as for example the Multiple DCT [5], i.e. these methods are not well conformed to the image content.

A group of methods for image representation, based on the use of artificial neural networks (NN) [7-14] had recently been developed. Unlike the classic methods, this approach is distinguished by higher compression ratios, because together with the coding, NN training is performed. The results already obtained show that these methods can not successfully compete the still image compression standards, JPEG and JPEG2000 [3]. For example, the Adaptive Vector Quantization (AVQ), based on SOM NN [8, 13], requires the use of code books of too many vectors, needed to ensure high quality of the restored image and this results in lower compression.

In this paper is offered new adaptive method for pyramidal representation of digital images with 3-layer BPNNs. The results obtained with the method modeling show significant visual quality enhancement for the restored images in comparison with the standards for image compression. The paper is arranged as follows: in Section II is described the method for adaptive pyramidal image representation, in Section III is given the algorithm simulation; in Section IV are given some experimental results, and Section V is the Conclusion.

II. METHOD FOR ADAPTIVE PYRAMIDAL IMAGE REPRESENTATION

2.1. Pyramidal Decomposition Selection

The basic advantage of the pyramidal decomposition in comparison with the other methods for image compression is the ability to perform “progressive” transfer (or storage) for every consecutive decomposition layer. In result, the image could be restored with high compression ratio and gradually improving quality. The classic approach for progressive image transfer is based on the Laplasian pyramid (LP) [4] combined with the Gaussian (GP).

In this paper is offered new approach for pyramidal image representation, based on the so-called Adaptive Inverse Difference Pyramid (AIDP). Unlike the non-adaptive Inverse Difference Pyramid (IDP) [5] it is built in the non-linear transformed image space using a group of NNs. The AIDP is calculated starting the calculation from the pyramid top, placed down, and continues iteratively with the next pyramid
layers. The AIDP built this way, has some important advantages when compared to the LP: easier implementation of the progressive image transfer and compression enhancement in the space of the hidden layers of the corresponding NN.

2.2. Description of the Inverse Difference Pyramid

Mathematically the digital image is usually represented as a matrix of size $H \times V$, whose elements $b(x, y)$ correspond to the image pixels; $x$ and $y$ define the pixel position as a matrix row and column and the pixel brightness is $b$. The halftone image is then defined as:

$$ [B(x, y)] = \begin{bmatrix} b(0,0) & b(0,1) & \cdots & b(0,H-1) \\ b(1,0) & b(1,1) & \cdots & b(1,H-1) \\ \vdots & \vdots & \ddots & \vdots \\ b(V-1,0) & b(V-1,1) & \cdots & b(V-1,H-1) \end{bmatrix} $$

(1)

In order to make the calculation of the pyramidal image decomposition easier, the matrix is divided into $K$ blocks (sub-images) of size $m \times m$ $(m=2^n)$ and on each is then built a multi-layer IDP. The number $p$ of the IDP layers for every block is in the range $0 \leq p \leq n-1$. The case $p=n-1$ corresponds to complete pyramidal decomposition of maximum number of layers, for which the image is restored without errors (all decomposition components are used).

The IDP top (layer $p=0$) for a block of size $2^n \times 2^n$ contains coefficients, from which after inverse transform is obtained its worse (coarse) approximation. The next IDP layer for the same block (the layer $p=1$) is defined from the difference between the block matrix and the approximation, divided into $4$ sub-matrices of size $2^{n-1} \times 2^{n-1}$ in advance. The highest IDP layer (layer $p=n-1$) is based on the information from the pixels in all the $4^{n-1}$ difference sub-matrices of size $2 \times 2$, obtained in result of the $(n-1)$-time division of the initial matrix into sub-matrices.

In correspondence with the described principle, the matrix $[B_{k_0}]$ of one image block could be represented as a decomposition of $(n+1)$ components:

$$ [B_{k_0}] = \{B_{k_0}\} + \sum_{p=1}^{n-1} \{E_{k_p}\} + \{E_{k_{n-1}}\} $$

(2)

for $k_p=1,2,\ldots,4^K$ and $p=0,1,\ldots,n-1$.

Here $k_p$ is the number of the sub-matrices of size $m_p \times m_p$ $(m_p=2^{n-p})$ in the IDP layer $p$; the matrices $[\tilde{B}_{k_p}]$ and $[\tilde{E}_{k_p}]$ are the corresponding approximations of $[B_{k_p}]$ and $[E_{k_p}]$; $[E_{k_{n-1}}]$ is the matrix, which represents the decomposition error in correspondence with Eq. (2), for the case, when only the first $n$ components are used.

The matrix $[E_{k_{p-1}}]$ of the difference sub-block $k_{p-1}$ in the IDP layer $p$ is defined as:

$$ [E_{k_{p-1}}] = [E_{k_p}] - [\tilde{E}_{k_p}] $$

(3)

for $p=2,3,\ldots,n-1$. In this case $p=1$:

$$ [E_{k_1}] = [B_{k_1}] - [\tilde{B}_{k_1}] $$

(4)

The matrix $[E_{k_{p-1}}]$ of the difference sub-block in the layer $p$ is divided into $4^K$ sub-matrices $[E_{k_p}]$ and for each is then calculated the corresponding approximating matrix $[\tilde{E}_{k_p}]$. The submatrices $[\tilde{E}_{k_p}]$ for $k_p=1,2,\ldots,4^K$ define the next decomposition component $(p+1)$, represented by Eq. (2). For this is necessary to calculate the new difference matrix and then to perform the same operations again following the already presented order.

2.3. Image representation with AIDP-BPNN

The new method for image representation is based on the IDP decomposition, in which the direct and inverse transforms in all layers are performed using 3-layer neural networks with error back propagation (BPNN) [7].

The general BPNN structure in AIDP was chosen to be a 3-layer one of the kind $m^2 \times n \times m^2$, shown in Fig. 1. The input layer is of $m^2$ elements, which correspond to the input vector components; the hidden layer is of $n$ elements for $n<m^2$, and the output layer is of $m^2$ elements as well, which correspond to the output vector components. The output $m^2$-dimensional vector is obtained in result of the transformation of the elements $m_0$ of each image block of size $m \times m$ into one-dimensional massif of length $m^2$ using the “meander” scan, shown in Fig. 2.

![Fig. 1. Three-layer BPNN with $n < m^2$ neurons in the hidden layer and $m^2$ neurons in the input and in the output layer](image-url)

In order to obtain better compression the processed image is represented by the sequence of $m^2$-dimensional vectors $\tilde{X}_1, \tilde{X}_2, \ldots, \tilde{X}_K$, which are then transformed in the $n$-dimensional vectors $\tilde{h}_1, \tilde{h}_2, \ldots, \tilde{h}_K$ correspondingly. The components of the vectors $\tilde{h}_k$ for $k=1,2,\ldots,K$ represent the neurons in the hidden layer of the trained 3-layer BPNN with $m^2 \times n \times m^2$ structure. In the output NN layer the vector $\tilde{h}_K$ is transformed back into the $m^2$-dimensional output vector $\tilde{Y}$, which approximates the corresponding input vector $X$.

The approximation error depends on the training algorithm and on
the participating BPNN parameters. The training vectors $\tilde{X}_1, \tilde{X}_2, \ldots, \tilde{X}_K$ at the BPNN input for the AIDP layer $p=0$ correspond to the image blocks. For the training was chosen the algorithm of Levenberg-Marquardt (LM) [7,8], which ensures rapidity in cases, when high accuracy is not required and as a result is suitable for the presented approach. One more reason is that the data necessary for the training has significant volume and information redundancy, but this does not make worse the training with the LM algorithm and influences only the time needed (i.e. it becomes longer).

The normalization is for range sigmoid function, defined by the relation:

$$r = \frac{1}{1 + e^{-x}}.$$  \hfill (5)

where $W$ is a matrix of weight coefficients of size $m \times n$, which is used for the linear transform of the input vector $\tilde{X}_k$; $\tilde{b}_j$ is the $n$-dimensional vector of the threshold coefficients in the hidden layer, and $f(x)$ is a linear activating sigmoid function, defined by the relation:

$$f(x) = \frac{1}{1 + e^{-x}}.$$  \hfill (6)

In result the network performance becomes partially non-linear and this influence is stronger when $x$ is outside the range [-1.5, +1.5].

The relation between the $n$-dimensional vector $\tilde{h}_k$ of the hidden layer and the $m^2$-dimensional BPNN vector $\tilde{Y}_k$ from the AIDP layer $p=0$, which approximates $\tilde{X}_k$, is defined in accordance with Eq. (5) as follows:

$$\tilde{Y}_k = f([W]_2 \tilde{h}_k + \tilde{b}_2) \quad \text{for} \quad k=1,2,\ldots,K,$$  \hfill (7)

where $[W]_2$ is a matrix of size $n \times m^2$ representing the weight coefficients used for the linear transform in the hidden layer of the vector $\tilde{h}_k$, and $\tilde{b}_2$ is the $m^2$-dimensional vector of the threshold coefficients for the output layer. Unlike the pixels in the halftone images, whose brightness is in the range $[0, 255]$, the components of the input and output BPNN vectors are normalized in the range $x_i(k), y_i(k) \in [0,1]$ for $i=1,2,\ldots,m^2$. The components of the vector which represents the neurons in the hidden layer $h_j(k) \in \{0,1\}$ for $j=1,2,\ldots,n$ are placed in the same range, because they are defined by the activating function $f(x) \in [0,1]$. The normalization is necessary, because it enhances the BPNN efficiency [8].

The image representation with AIDP-BPNN is performed in two consecutive stages: 1) BPNN training and 2) coding of the obtained output data.

For the BPNN training in the AIDP layer $p=0$, the vectors $\tilde{X}_k$ are used as input and reference ones, with which are compared the corresponding output vectors. The comparison result is used to correct the weight and the threshold coefficients so that to obtain minimum MSE. The training is repeated until the MSE value for the output vectors becomes lower than predefined threshold.

For the training of the 3-layer BPNN in the next ($p>0$) AIDP layers are used the vectors obtained after the dividing of the difference block $[E_{k-1}]$ (or sub-block) into $4K$ sub-blocks and their transformation into corresponding vectors. The BPNN training for each layer $p>0$ is performed in the way already described for the layer $p=0$.

In the second stage the vectors in the hidden BPNN layers for all AIDP layers are coded losslessly with entropy coding. The coding is based on two methods: Run-Length Coding (RLC) and variable length Huffman coding [6]. The block diagram of the pyramid decomposition for one block of size $m \times m$ with 3-layer BPNN for the layers $p=0,1,2$ and entropy coding/decoding is shown in Fig. 3. When the BPNN training is finished, for each layer $p$ are defined the corresponding output weight matrix $[W]_p$ and the threshold vector $[\tilde{b}]_p$. The entropy coder (EC) compresses the data transferred to the decoder for the layer $p$, i.e.:  

- The vector of the threshold coefficients for the neurons in the output NN layer (common for all blocks in the layer $p$);  
- The matrix of the weight coefficients of the relations between the neurons in the hidden layer towards the output BPNN layer (common for all blocks in the layer $p$);  
- The vector of the neurons in the hidden BPNN layer, personal for each block in the layer $p$.

In the decoder is performed the entropy decoding (ED) of the compressed data. After that the BPNN in the layer $p$ is initialized setting the values of the threshold coefficients for the neurons in the output layer and of the weight coefficients for the neurons, connecting the hidden and the output layers.

At the end of the decoding the vector of the neurons in the hidden BPNN layer for each block is transformed into corresponding output vector. The obtained output vectors are used for the restoration of the processed image.
III. SIMULATION OF THE AIDP-BPNN ALGORITHM

For the simulation of the AIDP-BPNN algorithm, it is necessary to perform the following operations: transformation of the input data into a sequence of vectors; selection of the BPNN structure; BPNN creation and initialization of its parameters; BPNN training using the input vectors so that to obtain the required output vectors; testing of the AIDP-BPNN algorithm with various test images and evaluation of their quality after restoration (objective and subjective).

The AIDP-BPNN algorithm consists of the following steps:

**Step 1.** The input halftone image is represented as a matrix of size $H \times V$, 8bpp (in case that $H$ and $V$ are not multiples of 2 the matrix is expanded with zeros, until the required size is obtained);

**Step 2.** The input image matrix is divided into $K$ blocks of size $m \times m$ ($m=2^n$). The value of $m$ is selected so that to retain as much as possible the correlation between the block pixels (for big images of size $1024 \times 1024$ or larger the block is usually $16 \times 16$ or $32 \times 32$, and for smaller images it is $8 \times 8$);

**Step 3.** The AIDP layer numbers $p$ are set, starting with $p=0$;

**Step 4.** The matrix of every block (sub-block) of $m^2/2^p$ elements in the layer $p$ is transformed into input vector of $\text{size}(m^2/2^p) \times 1$. The so obtained $4^p K$ input vectors constitute a matrix of $\text{size}(m^2/2^p) \times 4^p K$, which is used for the BPNN training and as a matrix of the reference vectors, which are then compared with the BPNN output vectors;

**Step 5.** The matrix for used for the BPNN training is normalized transforming its range $[0,255]$ into $[0,1]$;

**Step 6.** The working function of the BPNN is set: the $\text{mse}$ function (mean square error);

**Step 7.** The criterion for the BPNN training end is defined setting the deviation value $(0.01)$;

**Step 8.** The maximum number of training cycles $(5000)$ is set, after which the training ends;

**Step 9.** Iterative BPNN tuning is performed, using the function which follows the error gradient. After that the following information is saved in a special file:
- The neurons of the hidden layer, which in general are different for every block (sub-block);
- The threshold coefficients for the output layer;
- The matrix of the weight coefficients between the hidden and the output BPNN layers.

**Step 10.** The data, described in Step 9 is losslessly coded using RLC and Huffman code and is saved in a special file, which contains the compressed data for the layer $p$;

**Step 11.** The compression ratio $\text{Cr}(p)$ for the data of the AIDP-BPNN layer $p$ is calculated;

**Step 12.** The layer number $p$ is increased ($p=p+1$): in case that it is lower than the maximum $(p_{\text{max}} \leq n)$ is performed step 3, else the processing continues with Step 13.

**Step 13.** One common file is generated, where is stored the data from all layers: $p = 0,1,\ldots,p_{\text{max}}$;

**Step 14.** The global compression ratio obtained using the AIDP-BPNN representation is calculated;

**Step 15.** The decoder receives the sequentially transferred data for the AIDP-BPNN layers $p = 0,1,\ldots,p_{\text{max}}$;

**Step 16.** For every layer $p$ are decoded the values of the neurons in the hidden layer for each block (sub-block), the
threshold coefficients and the matrix of the weight coefficients for the corresponding output BPNN layer;

Step 17. The decoded data for each AIDP-BPNN layer are set for the corresponding BPNN in the decoder;

Step 18. The components of the vector for each block (sub-block) in the output BPNN layer are restored;

Step 19. The output BPNN vector is transformed into the block (sub-block) matrix;

Step 20. The range [0,1] of the matrix elements is transformed back into [0,255];

Step 21. The restored image is visualized for visual evaluation of its quality;

Step 22. The parameters $MSE$ and $PSNR$ are calculated for the objective evaluation of the restored image;

For the image representation in accordance with the AIDP-BPNN method was developed new format, which contains the 3 main BPNN components for every layer. The new structure is as follows:

- The vector of the values of the neurons in the hidden layer – personal for each block/sub-block;
- The vector of the threshold coefficients for the output layer – common for all blocks/sub-blocks;
- The matrix of the weight coefficients for the output layer - common for all blocks/sub-blocks.

All coefficients are calculated with accuracy “double word”.

IV. EXPERIMENTAL RESULTS

The experiments with the AIDP-BPNN algorithm were performed with test images of size 512×512, 8 bpp (i.e. 262 144B). In the ADP layer $p=0$ the image is divided into $K$ blocks of size 8×8 pixels, (K=4096). At the BPNN input for the layer $p=0$ is passed the training matrix of the input vectors of size 64×4096=262 144. In the hidden BPNN layer the size of each input vector is reduced from 64 to 8. The restoration of the output vector in the decoder is performed using these 8 components, together with the vector of the threshold values and the matrix of the weight coefficients in the BPNN output layer. For the layer $p=0$ the size of the data obtained is 266 752 B, i.e. larger than that of the original image (262 144 B). As it was already pointed out, the data has high correlation and is efficiently compressed with entropy coding. For example, the compressed data size for the same layer ($p=0$) of the test image Tracy is 4374 B (the result is given in Table 1). Taking into account the size of the original image, is calculated the compression ratio $Cr=59.93$. The Peak Signal to Noise Ratio for the test image Tracy is 4374 B (the result is given in Table 1). Taking into account the size of the original image, is calculated the compression ratio $Cr=59.93$. The Peak Signal to Noise Ratio for the test image Tracy for $p=0$ (Table 1) is $PSNR=35.32$ dB. In the same table are given the compression ratios obtained with AIDP-BPNN for other 7 test images of same size (512×512). It is easy to see that for the mean compression ratio $Cr=60$ is obtained $PSNR=30$ dB, i.e. the visual quality of the restored test images is good enough for various applications.

In Fig.4 are shown the graphic relations $PSNR=f(Cr)$ for each of the 8 test images, compressed in accordance with the AIDP-BPNN method for the layers $p=0,1,2$.

In Table 2 are given the results for the same 8 test images obtained using 6 methods: AIDP-BPNN, 3 versions of the standards JPEG and JPEG2000, implemented with MATLAB, and 2 versions of the software product LuraWave SmartCompress (www.algovision-luratech.com). The results obtained show that AIDP-BPNN surpasses 4 of the compared methods and is comparable with Lura JPEG2000 for 6 test images. For the visual comparison in Fig. 5 are given the restored images BOY, obtained using the already mentioned 6 methods for image compression for relatively equal PSNR.

The visual evaluation of the restored images quality shows that AIDP-BPNN is better than the remaining 5 methods.

In Table 1 are given the results for the same 8 test images after AIDP-BPNN compression using layer $p=0$

<table>
<thead>
<tr>
<th>Име на файл</th>
<th>$Cr$</th>
<th>$PSNR$ (dB)</th>
<th>$RMSE$</th>
<th>Брой битове/пиксел (bpp)</th>
<th>Компресиран файл</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOY</td>
<td>60.40</td>
<td>29.05</td>
<td>9.22</td>
<td>0.1324</td>
<td>4340</td>
</tr>
<tr>
<td>FRUIT</td>
<td>60.29</td>
<td>32.89</td>
<td>5.79</td>
<td>0.1326</td>
<td>4348</td>
</tr>
<tr>
<td>Tracy</td>
<td>59.93</td>
<td>35.32</td>
<td>4.37</td>
<td>0.1334</td>
<td>4374</td>
</tr>
<tr>
<td>Vase</td>
<td>60.18</td>
<td>26.83</td>
<td>11.62</td>
<td>0.1329</td>
<td>4356</td>
</tr>
<tr>
<td>Clown</td>
<td>60.01</td>
<td>31.81</td>
<td>6.55</td>
<td>0.1333</td>
<td>4368</td>
</tr>
<tr>
<td>Peppers</td>
<td>60.23</td>
<td>30.94</td>
<td>7.24</td>
<td>0.1328</td>
<td>4352</td>
</tr>
<tr>
<td>Text</td>
<td>60.23</td>
<td>18.69</td>
<td>29.65</td>
<td>0.1328</td>
<td>4352</td>
</tr>
<tr>
<td>Lena 512</td>
<td>59.57</td>
<td>29.15</td>
<td>8.89</td>
<td>0.1334</td>
<td>4400</td>
</tr>
</tbody>
</table>

Fig. 5. The restored test image BOY after compression with 6 methods.
The NN architecture used for the experiments comprises for the zero level 64 neurons in the input layer, 8 neurons in the hidden layer, and 64 neurons in the output layer. The chosen proportion for the input vectors was correspondingly: 80% for Training; 10% for Validation and 10% for Testing.

V. CONCLUSION

In this paper is presented one new approach for still image adaptive pyramid decomposition based on the AIDP-BPNN algorithm. The algorithm modeling was used to compare it with 5 versions of the image compression standards JPEG and JPEG2000. The results obtained show that for same conditions it ensures higher visual quality of the restored images. For example, for the images of the kind “portrait” (Lena and Tracy) it ensures high image quality using the zero pyramid level only: 59.93 dB and 32.89 dB versus 59.57 dB and 29.15 dB correspondingly.

The AIDP-BPNN is asymmetric (the coder is more complicated than the decoder) and this determines it mostly in application areas which do not require real time processing i.e. applications, for which the training time is not crucial.

The hardware implementation of the method is beyond the scope of this work. The experiments for the AIDP-BPNN algorithm were performed with sub-blocks of size 8x8 pixels.

The computational complexity of the method was compared with that of JPEG and the investigation proved that AIDP-BPNN complexity is comparable with that of JPEG. In general, the computational complexity of the method depends on the training method selected.

The new method offers wide opportunities for application areas in the digital image processing, such as the progressive transfer via Internet, saving and searching in large image databases, etc.

ACKNOWLEDGEMENT

This paper was supported by the National Fund for Scientific Research of the Bulgarian Ministry of Education and Science, Contract VU-I 305.

REFERENCES