Image Compression using Radial Basis Function Networks
OREST VASCAN*, IONEL-BUJOREL PAVALOIU+
Crescendo International*, Department of Engineering in Foreign Languages+
University POLITEHNICA of Bucharest
Spl. Independenţei 313, Bucharest
ROMANIA
orest.vascan@crescendo.ro, bujor.pavaloiu@ing.pub.ro

Abstract: - In this paper it is proposed an image compression method based on the idea of fitting a set of Neural Networks (NNs) outputs to the image surface, which is a three-dimensional surface where the pixel values are considered as heights (z-values) defined on the x–y ground plane. An image is divided into subimages (blocks) using a quad tree, according to the complexity of the image surface. Individual block surfaces are fitted using a radial-basis function network (RBFN), and the parameters of the RBFN are stored as image representation in a compressed form.

Key-Words: - Image compression, Neural Networks, Radial Basis Function Network, Gaussian Approximation

1 Introduction
Neural Networks are basic tools used in image and signal processing for operations that include pattern recognition and template matching.

In this work neural networks are used for image compression.

Image compression, the reduction of the space used to store an image, can be lossless or lossy, when the reconstruction is not identical with the original.

Among the various techniques of lossy image compression, we have to mention the transform based methods. Discrete Cosine Transform (DCT) is used in JPEG Standard and Discrete Wavelet Transform (DWT) is used in JPEG2000. They transform blocks of the image (subimages) into a frequency-like space, followed by quantization and entropy coding [1].

Some of the tools that are used in image compression belong to the class of NNs [2], [3].

There are two large approaches in this trend. The first one uses unsupervised NNs for the clustering and quantization of data obtained through DCT or DWT [4], [5]. The other uses a special type of Feed Forward NN, with supervised learning, that has the same number of inputs and outputs and a single hidden layer with a smaller number of elements.

The network is trained with pixel values from the blocks of the image, trying to have the same output as the input.

The information from the image is in a compressed form at the level of the hidden layer, because of the smaller number of units.

Instead of this usual schemes, we will use the fact that NNs in general and RBFNs in particular are appropriate solutions to high-dimensional surface fitting problems. We are using a similar approach with [6], where the image intensity value in point (x,y) is considered to be the height z(x,y) of a point belonging to the image surface.

We are using the maximum number of elements (neurons) for a RBFN as the main control parameter and that will set the compression ratio and the quality for the image.

2 RBFN Representation of Images
The RBFN is a linear combination of Radial Basis Function (RBF) units.

A RBF unit has the output dependent only on the distance from the input to a given center. We will use the Gaussian as a basic form of RBF.

The Gaussian G can be written as:

\[
G(x, y) = A_G e^{-\frac{(x-x_G)^2+(y-y_G)^2}{2\sigma_G^2}}
\] (1)

where the pair (x,y) are a pixel coordinates, \(A_G\) is the amplitude of the Gaussian, the pair \((x_G,y_G)\) represents the center of G and \(\sigma_G\) is the standard deviation.

In this work, a RBFN with N units will be defined as:

\[
I(x, y) = A_0 + \sum_{i=1}^{N-1} A_i e^{-\frac{(x-x_i)^2+(y-y_i)^2}{2\sigma_i^2}}
\] (2)

where the first unit delivers a constant value (DC) and the rest of the units are Gaussian RBFs.
In Fig. 1 it is shown a RBFN with M units that is used in this paper. The image is split in blocks and the intensity of a pixel is the sum of the outputs for that pixel corresponding to the Gaussian RBFs of the RBFN assigned to that block.

![Fig. 1 RBFN for image surface representation](image)

Gaussian representation is very interesting for image compression. One has to store just the parameters of the RBFNs and to reconstruct each subimage by adding the surfaces corresponding to the RBFs from the given RBFN.

In [6], the authors use RBFNs with centers disposed in specific patterns, making it easier to compute the parameters of the RBFN. We will use a different approach, trying to find a minimal RBFN (with minimum number of units) that will fit the subimage surface.

Finding the minimal Gaussian decomposition is not a trivial challenge, with few attempts to solve the problem. The paper of Goshtasby and O’Neill [7] made remarkable initial steps in dealing with the problem. In [8] and [9], the classic Marquardt algorithm [10] is improved and complexity issues are taken into account. The authors present in [11] and [12] the construction of RBFN with different topologies.

### 3 Image Compression using RBFN

The number of RBFs used to encode a subimage can be far fewer than the number of data points, allowing reduction of data size.

The image is split into subimage blocks in a quad-tree manner, reducing the complexity whenever the representation is not possible.

For a given Gaussian RBF, one has to store the coordinates of the center and the dispersion as parameters belonging to the first layer in the network, and the amplitude, equal with the weight corresponding to the neuron connection to the output.

We present in the following the algorithm used for image compression:

**Algorithm:**

1. Consider the image to be the current subimage
2. For each subimage, until there is no one left, try to represent it as a RBFN, starting with DC node and adding nodes up to the maximal size of RBFN
3. If the representation is not possible, divide recursively the subimage into a quad of subimages and go to step 2
4. If the representation is possible, write the RBFN representation to file and go to the next subimage

The maximum number of RBFs in a RBFN determines the quality of the approximation. If one choose a large number of RBFs to be tried, there is a large probability that the decomposition will not go down to blocks with small size – for coding using DC and 3 RBFs, it will not go lower than blocks of size 2.

The image will be encoded like in Fig. 2. The header is represented by the maximum number of neurons used to encode a subimage and the number of bits used to represent the depth of a block in the QT decomposition. The blocks are represented sequentially.

<table>
<thead>
<tr>
<th>Max size of RBFN</th>
<th>No. of bits for depth</th>
<th>Block1</th>
<th>…</th>
<th>Block N</th>
</tr>
</thead>
</table>

![Fig. 2 Image representation](image)

A block starts with the its depth, that together with the given order of crossing the image and with its position in the sequence of blocks, identifies uniquely the subimage that it represents. Like in Fig. 3, it follows the code of the RBFN, equal with the number of units in the RBFN, excluding the DC node (when just the DC is present, the code is 0). For the maximum size of the RBFN having the value 4, the code it is stored on 2 bits and the RBFN can have just the DC unit or the DC unit and up to three Gaussian RBF units.

<table>
<thead>
<tr>
<th>Depth of Block</th>
<th>RBFN type code</th>
<th>DC</th>
<th>RBF₁</th>
<th>…</th>
<th>RBF₄</th>
</tr>
</thead>
</table>

![Fig. 3 Block representation](image)

The DC is stored on 8 bits, but the bit size of a RBF node is variable. A RBF structure is represented in Fig. 4 and consists on the coordinates...
of the center, the dispersion and the height (the weight of the given neuron to the output).

<table>
<thead>
<tr>
<th>$x_G$</th>
<th>$y_G$</th>
<th>$\sigma_G$</th>
<th>$h_G$</th>
</tr>
</thead>
</table>

*Fig. 4 RBF representation*

A RBF with a dispersion of $1/3$, its center in a given pixel and its output weight equal with the pixel color can represent any color/height for that pixel without influencing the neighbor pixels. It is clear this way that a subimage with size $n$ can be represented by a RBFN with $n^2$ units. In order to achieve compression a smaller number of RBF units should be chosen.

The number of bits needed for the encoding and the quantization are different for subimages of distinctive sizes. Because we use quad tree decomposition, the number of pixels $n$ on a subimage side will be a power of $2$, $n=2^k$. This allows us to use just $k$ bits for the coordinates $x_G$ and $y_G$ of the center when it is placed in the centers of the pixels placed inside the subimage. We will let the center to be positioned outside the subimage, at a maximum distance of $n$ to the center of the block on both $x$ and $y$ directions. The number of bits necessary to encode both $x_G$ and $y_G$ will be thus $k+1$.

The dispersion will be allowed to have values from $1/3$ to $2n/3$ with a linear discretization of $1/3$, requiring $k+1$ bits to store one of the $2n$ possible values. The height $h_G$ will be stored on 8 bits for normal 8 bit grayscale images. The maximum number of units in a RBFN should be a power of 2 and we tested for a DC unit and up to 3 RBF units, requiring 2 bits to store the code of the network. For a block of size 2 (4 pixels), it is not efficient to encode it with more than 2 units.

We find the parameters using genetic algorithm (GA) search with the constraints of having integer values in the bounds specified above. The former best RBFs are used for the initial population when we have to add a new RBF to the existing RBFN. Gradient methods are used to speed up the process, but it needs further improvement. Encoding of an image just with DC units give block effects, as can be seen in Fig. 5 and in the detail from Fig. 6. Using a RBFN with 4 units (neurons) insure a better representation because of natural way it represents the gradients in color.

For a quantitative analysis, for compressions of Lena image with 1.12bpp we obtained $MSE = 22.74$ and $PSNR=34.56$, in the situation where the method is still under work.
4 Conclusion
The proposed algorithm attempts to compress an image considering the intensity values of a region as a surface, and fitting a set of RBFN outputs to it.

Gaussian representation is a promising field for image compression. One has only to store the Gaussians parameters and to reconstruct the image just by adding the surfaces. This representation is also fit for data transmissions, where the coefficients of the Gaussians are to be transmitted from the ones corresponding to the large dispersions to the ones corresponding to the small values. This way, the image is transmitted and rebuilt progressively, from the coarse structure to final one by adding finer and finer details. Having an image represented as a sum of Gaussians can be extremely effective in video compression, where the Gaussians in one frame can be used to determine their “altered” version in the next one. Gaussian representation can be also used as representing features in an image, making it simple for pattern recognition.

Finding the minimal Gaussian decomposition of a surface is a necessary procedure and we are working to improve it, as well as to set better values for ranges and quantization of RBFs parameters.

References: