Image Compression Using a Multilayer Neural Network

S. P. LUTTRELL
Royal Signals and Radar Establishment, St Andrews Road, Malvern, WORCS, WR14 4NL, UK

We demonstrate that a topographic neural network model (Kohonen, 1984) may be used to data
compress synthetic aperture radar (SAR) images by up to a factor of 8.

I. INTRODUCTION

A property of neural networks is their ability to con-
struct feature detectors as a result of supervised or unsup-
vised training. We demonstrate that a class of neural
networks which produces topographic mappings \( \Pi \) may
be used to data compress SAR images. In Section II we
summarise Kohonen’s network learning algorithm, and
we present an improved version of the algorithm in Sec-
tion III. In Section IV we generalise the method to mul-
tilayer mappings and indicate how such networks might
be implemented as table look-up operations. In Section
V we apply such a multilayer network to the problem of
data compressing SAR images.

II. KOHONEN’S NETWORK LEARNING
ALGORITHM

Define a mapping \( T(x_{in}) \) from a \( d_{in} \)-dimensional input
vector \( x_{in} \) to a \( d_{out} \)-dimensional \( (d_{out} \leq d_{in}) \) output
vector \( x_{out} \) as

\[
x_{out} \equiv T(x_{in})
\]

A vector quantiser representation of \( T(x_{in}) \) may be con-
bructed by defining a set of code vectors \( \{ \nu_{in,j} : j = 1,2,\cdots,m \} \)
together with the \( m \) \( L_2 \) norms

\[
D_j(x_{in}) \equiv \| x_{in} - \nu_{in,j} \|^2 \quad j = 1,2,\cdots,m
\]

and forming a mapping \( x_{in} \rightarrow j_0 \) such that \( j = j_0 \)
minimises \( D_j(x_{in}) \) with respect to \( j \). The index \( j_0 \) then
plays the role of the output vector \( x_{out} \), and \( T^{-1}(x_{out}) \)
becomes the pseudo-inverse \( j_0 \rightarrow x_{in,j_0} \).

Kohonen has presented an algorithm for unsupervised
training of the set of \( \nu_{in,j} \) from examples \( x_{in} \) drawn from
the pattern space \( \Pi \). The algorithm is very similar to
the K-means clustering algorithm, and can also be shown
to minimise the mean square reconstruction error in cer-

tain cases. In its simplest form the algorithm may be
implemented as two update rules for each presentation
of a pattern \( x_{in} \):

\[
\nu_{in,j_0} \rightarrow \nu_{in,j_0} + \varepsilon_0 (x_{in} - \nu_{in,j_0}) \quad (2.3)
\]

\[
\nu_{in,N(j_0)} \rightarrow \nu_{in,N(j_0)} + \varepsilon_1 (x_{in} - \nu_{in,N(j_0)}) \quad (2.4)
\]

where \( 1 > \varepsilon_0 > \varepsilon_1 > 0 \), and \( N(j_0) \) ranges over some
neighbourhood set of \( j_0 \). The set of neighbourhoods de-
defines the topology of the discrete output space which is
usually chosen to be some regular lattice in one or two
dimensions. Equation 2.3 by itself will lead to a vector
quantiser, but if, in addition, Equation 2.4 is imposed
then the vector quantiser approximates a topographic
mapping \( \Pi \). This arises because Equation 2.4 tends to
force \( \nu_{in,N(j_0)} \) to lie close to \( \nu_{in,j_0} \) in the pattern space,
so the pseudo-inverses of \( j_0 \) and \( N(j_0) \) are close together.
The parameters \( \varepsilon_0 \) and \( \varepsilon_1 \) are functions of the number of
training cycles which have elapsed, and \( \varepsilon_1 \) also depends
on the neighbourhood distance \( j_0 - N(j_0) \). However we
shall see that all such dependencies can be removed for
the purpose of SAR image compression.

III. AN IMPROVED LEARNING ALGORITHM

For simplicity we shall explain our improved algorithm
for the case \( d_{out} = 1 \). Thus we train initially with a small
value of \( m \) \( (m = 2, \text{say}) \) with nearest neighbour interac-
tions in Equation 2.3 until the code vectors equilibrate.
We then increase \( m \) to \( 2m - 1 \) by inserting additional
code vectors according to the prescription

\[
\nu_{in,j} = \frac{1}{2}(\nu_{in,j} + \nu_{in,j+1}) \quad j = 1,2,\cdots,m - 1 \quad (3.1)
\]

and continue training as before using nearest neighbour
interactions only. We repeat the cycle of training followed
by insertion of new code vectors until \( m \) is sufficiently
large that the pseudo-inverse \( j_0 \rightarrow \nu_{in,j_0} \) produces an
acceptable reconstruction. When \( d_{out} > 1 \) an appro-
priate multidimensional neighbourhood set of \( \nu_{in,j} \) used
on the right hand side of Equation 3.1.

This technique of building up the number of code vec-
tors converges more rapidly than Kohonen’s original algo-
rithm, because our code vector insertion procedure seeds
the code vectors with good positions, and thus reduces the
overall amount of computation which is required. We
find that the consequential increase in the rate of conver-
gence far outweighs the decrease due to adjustment of
old code vectors after new ones have been inserted. This
is because our technique ensures that the code vectors
tend not to get stuck in ‘knotted’ configurations from
which they have a low probability of escaping, and also
because we do not waste resources attempting to com-
pute the optimum locations of multiply correlated code
vector positions during the early (approximate) stages of
the training process, as in Kohonen’s original scheme.
IV. MULTILAYER MAPPINGS

We have described how to map a \( d_{in} \)-dimensional input space into a \( d_{out} \)-dimensional output space. For image compression this is not feasible because \( d_{in} \) is enormous (it is equal to the number of pixels in the image). We must therefore partition the image into a large number of blocks of pixels each of which is separately processed. Unfortunately this is computationally very intensive. If the number of bits which is used to represent \( x_{in} \) is not too large and a particular set of code vectors is to be used for many images, then the processing time may be reduced by explicitly tabulating the mapping \( x_{in} \rightarrow j_{0} \) for all \( x_{in} \). The maximum feasible table size is around 1 Megabyte using current memory technology, which limits the number of input bits to 20 assuming that 0 < \( j_{0} < 255 \) (i.e. 1 byte per table entry). This limitation on \( d_{in} \) is unacceptable for image compression, so we must resort to multilayer mappings to factorise the overall data compression into calculable pieces.

For convenience we shall use the following notation

\[
\begin{align*}
   x^{(k)} & = \text{state of } k\text{-th network layer} \\
   x_{b}^{(k)} & = \text{state of } b\text{-th block of } k\text{-th network layer} \\
   \{x_{j}^{(k)} : j = 1, 2, \ldots, m_{k}\} & = \text{code vectors for } k\text{-th network layer} \\
   j_{b}^{(k)} & = \text{result of mapping } x_{b}^{(k)} \\
\end{align*}
\]

where each layer is partitioned into an array of nonoverlapping blocks of identical shape and orientation such as \( 2 \times 1 \) or \( 2 \times 2 \) pixel regions. We define the input image as \( x^{(0)} \), which we assume is represented as an array of \( w \)-bit unsigned integers. For simplicity we constrain \( m_{k} < 2^{w} \) so that \( j_{b}^{(k)} \) may also be represented as a \( w \)-bit word, and we use the same value \( m_{k} = m \) for each layer \( k \). The mapping from layer \( k \) to \( k + 1 \) is accomplished by computing the \( m \) norms \( \|x_{b}^{(k)} - \nu_{j}^{(k)}\|^{2} \) \((j = 1, 2, \ldots, m)\) for block \( b \), and selecting as the compressed version of \( x_{b}^{(k)} \) the value \( j = j_{b}^{(k)} \) which gives the minimum norm. We show part of this process for the case \( m = 7 \) in Figure 1 where a 2-tuple of pixels with value \( x_{b}^{(k)} = (x_{b,1}^{(k)}, x_{b,2}^{(k)}) \) is mapped (via the code vectors) to a 1-tuple \( j_{b}^{(k)} \) (\( 0 \leq j_{b}^{(k)} \leq 7 \) in this case), which becomes the value of the \( b \)-th pixel in layer \( k + 1 \). This process may be iterated by constructing blocks in layer \( k + 1 \) to yield the \( x_{b}^{(k+1)} \) which are then mapped to layer \( k + 2 \) using another set of code vectors, and so on.

![Figure 1: Stages of coding layer k to produce layer k+1.](image)

The learning algorithm may be generalised to multilayer mappings by training the \( \nu_{j}^{(0)} \) on a single image \( x^{(0)} \) followed by mapping \( x^{(0)} \) to \( x^{(1)} \) (via the \( j_{b}^{(1)} \)). The \( \nu_{j}^{(1)} \) are then similarly trained on \( x^{(1)} \), and so on. The \( x_{b}^{(k)} \rightarrow j_{b}^{(k)} \) mappings may then be tabulated for future use. Various improvements to this scheme are possible, such as simultaneous optimisation of all layers of the mapping, but we do not report them here. In our work we have used \( w = 8 \), \( m = 2^{w} = 256 \), and block sizes \( 1 \times 2 \) and \( 2 \times 1 \), so a 65 kilobyte (\( 2 \times 8 \) bit address space, 8 bit entries) look-up table suffices for each set of code vectors.

Image reconstruction is achieved by backpropagation from the last layer to the zeroth layer of the network. The inverse mapping from a pixel value \( j_{b}^{(k)} \) in layer \( k + 1 \) to an image block \( x_{b}^{(k)} \) in layer \( k \) is given by the pseudo-inverse \( j_{b}^{(k)} \rightarrow \nu_{j}^{(k)} \) where the \( \nu_{j}^{(k)} \) are real-valued quantities. It is necessary to round each component of \( \nu_{j}^{(k)} \) to the nearest integer in order to recover an \( x_{b}^{(k)} \) which may be used in the next stage of the backpropagation from layer \( k \) to layer \( k - 1 \).

V. DATA COMPRESSION OF SAR IMAGES

In a recent study [2] some statistical measures of the quality of SAR image reconstructions from the output of predictive compression systems were presented. We shall adopt the same measures, which are:

- **Mean:**

\[
m \equiv \frac{1}{N} \sum_{n=1}^{N} x_{n} \quad (5.1)
\]
Variance:
\[ v \equiv \frac{1}{N} \sum_{n=1}^{N} (x_n - m)^2 \]  
(5.2)

Skewness coefficient:
\[ s \equiv \frac{1}{N} \sum_{n=1}^{N} \frac{(x_n - m)^3}{v^{3/2}} \]  
(5.3)

Kurtosis coefficient:
\[ k \equiv \frac{1}{N} \sum_{n=1}^{N} \frac{(x_n - m)^4}{v^2} - 3 \]  
(5.4)

Autocorrelation at delay one:
\[ a \equiv \frac{1}{N - 1} \sum_{n=1}^{N-1} (x_n - m) (x_{n+1} - m) \]  
(5.5)

where \( x_n \) is the grey level (0 \( \leq x_n \leq 255 \)) of pixel \( n \), and \( N \) is the number of image pixels (1 \( \leq n \leq N \)). In our work we use 256 \( \times \) 256 pixel images so \( N = 65536 \). The autocorrelation at delay one in Equation (5.5) is written in one-dimensional form; the two-dimensional version is analogous.

Our approach to SAR image compression is different from the model based predictive approach of [2]. A priori, the only constraint that we place on the image compression is the number of bits per pixel \( m \) (and hence the number of code vectors per layer in our scheme), and the particular choice of block size and shape to use in each layer of the network. Everything else is deduced from the statistical structure of the training set, which means that, for instance, our approach will attempt to preserve information about image speckle. It requires additional prior knowledge (i.e. a data model) to reject speckle: this is an area of future research.

The network is trained on one or more SAR images which contain the types of features for which the learning algorithm must form code vectors. Typically we use urban images which have many small bright features, and rural images which have characteristic textural correlations. In all cases we globally scale the image pixel values to normalise their mean value to some (approximate) standard.

We use a standard network structure parameterised by \( w = 8, m = 2^w = 256 \), and block sizes \( 1 \times 2 \) and \( 2 \times 1 \), where we alternate the \( 1 \times 2 \) and \( 2 \times 1 \) blocks from layer to layer of the network. Each network layer thus provides a compression factor of 2. The SAR images which we use are auto focussed to remove the blurring effect of anomalous sensor motion; this results in a single look fully speckled image with a resolution of 1.5 metres. For our purposes we then average the moduli of all contiguous non-overlapping \( 2 \times 2 \) pixel blocks to produce a smoother image (with \( 1/4 \) of the number of pixels of the original image). This is a trade-off which entails some loss of resolution, but which at the same time reduces image speckle (as in multi-look SAR processing).

In Figure 2(a) we show a typical urban SAR image. We use this image as a training set together with update parameter values \( \varepsilon_0 = 0.1 \) and \( \varepsilon_1 = 0.02 \). For each network layer a training cycle consists of selecting \( 32m \) image blocks at random in order to train the \( m \) code vectors, starting with \( m = 2 \) and performing insertions upon equilibration of the code vectors until \( m = 257 \) is reached after 9 training cycles. We finally eliminate one code vector chosen at random and perform a final training cycle to achieve equilibration of \( m = 256 = 2^8 \) code vectors. Using these code vectors we forward propagate the image to the next network layer, and we repeat the training process to recover the next set of code vectors, and so on. We do not vary the values of the \( \varepsilon_0 \) and \( \varepsilon_1 \) parameters at any stage.

In Figure 2(b), Figure 2(c) and Figure 2(d) we show the image reconstructions which we obtain from layers 1, 2, 3 of a network which was trained on Figure 2(a). These correspond to compression factors of 2, 4, and 8 respectively. We also present the values of the statistics for both the original image and its reconstruction in Table 1 where \( R_k \) denotes the reconstruction from layer \( k \) (\( R_0 \) is the original image), \( a(r) \) denotes the azimuth (horizontal) autocorrelation and \( a(t) \) denotes the range (vertical) autocorrelation at delay one.

We also perform a visual comparison of \( R_0 \) and \( R_k \) (\( k > 0 \)) by ‘flicker photometry’ in which the images are rapidly interchanged on a display screen. \( R_1 \) is virtually indistinguishable from \( R_0 \). The only obvious difference between \( R_2 \) and \( R_0 \) is that the grey level of some pixels is substantially different, but the resolution of fine detail...
Table I: Statistics for the urban images in Figure 2(a)-Figure 2(d).

<table>
<thead>
<tr>
<th></th>
<th>R_0</th>
<th>R_1</th>
<th>R_2</th>
<th>R_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>117.4</td>
<td>116.9</td>
<td>116.2</td>
<td></td>
</tr>
<tr>
<td>v</td>
<td>3736</td>
<td>3696</td>
<td>3619</td>
<td>3594</td>
</tr>
<tr>
<td>a(a)</td>
<td>1701</td>
<td>1681</td>
<td>1649</td>
<td>1555</td>
</tr>
<tr>
<td>a(r)</td>
<td>1823</td>
<td>1811</td>
<td>1736</td>
<td>1515</td>
</tr>
<tr>
<td>s</td>
<td>0.805</td>
<td>0.820</td>
<td>0.859</td>
<td>0.876</td>
</tr>
<tr>
<td>k</td>
<td>-0.147</td>
<td>-0.146</td>
<td>-0.032</td>
<td>-0.014</td>
</tr>
</tbody>
</table>

is preserved. \( R_3 \) shows the same effect to a greater extent, and probably represents the limit to which our data compression scheme can be pushed for this type of SAR imagery. The degradation of the \( a(a) \) and \( a(r) \) statistics is a major contributor to the visual differences between the reconstructions and the original.

In order to test the possibility of using a single set of code vectors to compress a variety of SAR images, we apply the same code vectors to a rural SAR image containing a wood and some other cultural features. The original image is produced in the same way as the urban image and is shown in Figure 3(a). The reconstructions in Figure 3(b), Figure 3(c) and Figure 3(d) correspond to those in Figure 2(b), Figure 2(c) and Figure 2(d) and we present the corresponding statistics in Table II.

Table II: Statistics for the rural images in Figure 3(a)-Figure 3(d).

<table>
<thead>
<tr>
<th></th>
<th>R_0</th>
<th>R_1</th>
<th>R_2</th>
<th>R_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>136.2</td>
<td>135.6</td>
<td>135.2</td>
<td>140.0</td>
</tr>
<tr>
<td>v</td>
<td>3622</td>
<td>3605</td>
<td>3660</td>
<td>4024</td>
</tr>
<tr>
<td>a(a)</td>
<td>1457</td>
<td>1443</td>
<td>1461</td>
<td>1396</td>
</tr>
<tr>
<td>a(r)</td>
<td>1854</td>
<td>1841</td>
<td>1760</td>
<td>1546</td>
</tr>
<tr>
<td>s</td>
<td>0.436</td>
<td>0.448</td>
<td>0.518</td>
<td>0.567</td>
</tr>
<tr>
<td>k</td>
<td>-0.643</td>
<td>-0.649</td>
<td>-0.559</td>
<td>-0.668</td>
</tr>
</tbody>
</table>

Table III: Statistics for the rural images using matched network.

<table>
<thead>
<tr>
<th></th>
<th>R_0</th>
<th>R_1</th>
<th>R_2</th>
<th>R_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>136.2</td>
<td>135.7</td>
<td>134.9</td>
<td>134.6</td>
</tr>
<tr>
<td>v</td>
<td>3622</td>
<td>3583</td>
<td>3591</td>
<td>3594</td>
</tr>
<tr>
<td>a(a)</td>
<td>1457</td>
<td>1438</td>
<td>1393</td>
<td>1206</td>
</tr>
<tr>
<td>a(r)</td>
<td>1854</td>
<td>1843</td>
<td>1763</td>
<td>1598</td>
</tr>
<tr>
<td>s</td>
<td>0.436</td>
<td>0.450</td>
<td>0.403</td>
<td>0.396</td>
</tr>
<tr>
<td>k</td>
<td>-0.643</td>
<td>-0.652</td>
<td>-0.695</td>
<td>-0.690</td>
</tr>
</tbody>
</table>

VI. CONCLUSIONS

We have demonstrated the ability of a neural network to learn a feature space suitable for SAR image compression and reconstruction. The ability of a single network to compress a variety of SAR images suggests that a universal feature set might also be used for other low-level SAR image processing techniques.