NEURAL NETWORKS FOR DISCRETE TOMOGRAPHY

K.J. Batenburg [a] W.A. Kosters [b]

[a] *Mathematical Institute, Universiteit Leiden, and CWI, Amsterdam, The Netherlands;*
kbatenbu@math.leidenuniv.nl
[b] *Leiden Institute of Advanced Computer Science (LIACS), Universiteit Leiden, The Netherlands;*
kosters@liacs.nl

Abstract

Discrete tomography deals with the reconstruction of binary images from their projections in a small number of directions. In this paper we consider possible neural network approaches to this tomographic reconstruction problem. In particular we are interested in methods that can compute reconstructions in real-time and make efficient use of prior knowledge about the images, even when this knowledge is difficult to model by hand. We propose both a feed-forward back-propagation network method and a Hopfield network method for solving the reconstruction problem.

1 Introduction

Tomography deals with the reconstruction of an object from its projections in several directions. Figure 1 shows the basic principle. By measuring the attenuation of a parallel beam (e.g., X-rays) passing through the object, one obtains a projection of the "density" distribution in the object in the direction of the beam.

The main problem of tomography is to compute the density of the object from the measured projections. The most prominent use of tomography can be found in medical imaging, where X-ray CT scanners are used for non-invasive imaging of patients. In this application the various tissue types exhibit a continuous spectrum of X-ray absorption coefficient (e.g., densities), resulting in a reconstructed image that has a fine-grained spectrum of gray levels. To obtain sufficient reconstruction quality, more than 100 projections are required.

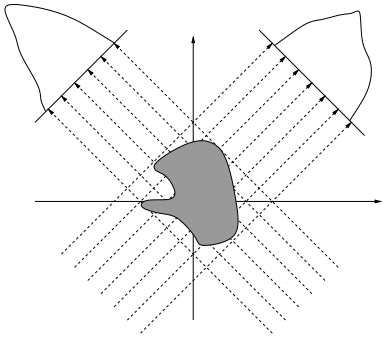

Figure 1: The basic principle of tomography

In *discrete tomography* one focuses on the reconstruction of objects having a very small, discrete set of densities; see [6]. Many objects scanned in industry, for example, are made of a single homogeneous material which results in a reconstruction that uses only two gray levels. Moreover, the gray levels are known in advance. By using this fact as a priori knowledge, it is possible to dramatically reduce the number of projections that is required to obtain an accurate reconstruction.

An important application of discrete tomography, which was a driving force behind initial research on the subject, is atomic resolution electron microscopy. Having the ability to make 3D reconstructions of crystalline solids at atomic resolution from electron microscopic images can be considered the "holy grail" in materials science. Recently, microscope resolution has improved considerably, making it possible to count the number of atoms in each column of a crystal, along several directions. Discrete tomography can be used to reconstruct the position of all individual atoms,

yielding an accurate 3D reconstruction, see [2, 8]. For this application, using few projections is a necessity, since the electron beam damages the sample after a few images have been recorded.

Reconstructing binary images from three or more projections is NP-hard [5]. Several algorithms have been proposed for computing a reconstruction when the image satisfies certain a priori assumptions, such as convexity, see, e.g., [1, 3, 6]. Although some of these algorithms are capable of computing high quality reconstructions for certain types of images, they usually have a considerable running time.

In this paper we focus on a specific problem in discrete tomography: reconstructing an $n \times n$ binary (black-and-white) image from its horizontal, vertical, diagonal and anti-diagonal projections or linesums, i.e., the total number of black pixels on those lines. Our results can be generalized to other tomography settings.

We number the pixels sequentially from 1 to n^2. There are n horizontal lines, n vertical lines, $2n-1$ diagonal lines and $2n-1$ anti-diagonal lines (see Figure 2). We denote the total number of lines by $m = 6n-2$. The problem input consists of m integers describing the number of black pixels (having pixel value 1) on each line. The equations relating the pixel values (either 0 or 1) to the projected linesums can be described in matrix-notation by the system $Ax = b$ where x is a column vector with n^2 elements, and the $m \times n^2$ matrix A contains a row for each horizontal, vertical, diagonal and anti-diagonal line of the image x: $A_{ki} = 1$ if line k contains pixel i and 0 otherwise. The integer $m \times 1$ vector b contains the number of black pixels for each line.

Note that in general there are 2^{n^2} possible $n \times n$ images. The projections give $m = 6n-2$ values in $\{0, 1, \ldots, n\}$, needing at most $6n \log_2 n$ bits to encode (some values cannot occur for the (anti-)diagonals). Also taking into account that not all bit sequences originate from proper images, this shows that a lot of information is lost during projection. Also notice that different images might generate the same series of projections (in case of only horizontal and vertical projections one can think of a chessboard, and interchange white and black). So the problem is in general underdetermined, but if we adhere to special images we can perhaps achieve acceptable reconstructions. In this paper we consider the reconstruction of images containing a fixed number of filled ellipses. An ellipse is specified by its center, the lengths of its two axes and an angle. So e ellipses in an $n \times n$ image can be encoded by some $5e \log_2 n$ bits, $\log_2 n$ for each parameter.

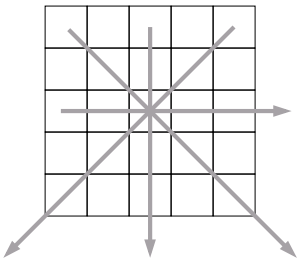

Figure 2: Four directions

In this paper we consider two neural network approaches for the reconstruction problem. The first approach (see Section 2), based on feed-forward back-propagation networks, has two key advantages. Prior knowledge on the set of images is specified implicitly, as it is learned by the network. In addition, when training of the network is complete it can compute a reconstruction extremely fast.

The second approach (Section 3) uses a Hopfield network to solve the reconstruction problem. The computations of the Hopfield network can be performed with a very high degree of parallelism, resulting in fast reconstructions. There is no training involved, the weights of the network are computed directly from the projection data. The tomography problem is solved by letting the Hopfield network converge to a local minimum of its energy function.

In Section 4 we describe several experiments, showing the feasibility of both approaches. In particular we show that our proposed topology for the feed-forward network is very effective. We conclude in Section 5.

2 Back-propagation network approach

In this section we will discuss the application of feed-forward back-propagation networks (see, e.g., [4]) to the discrete tomography problem.

The input nodes of the networks contain the values of the projections, the output nodes contain the pixel values. So the number of input nodes equals the number of projections, i.e., m, while the number of output nodes equals the size of the picture, i.e., n^2. The projection values are translated into equidistant points in the unit interval. For instance, the values $\{0, 1, 2, 3, 4\}$ are coded

as $\{1/6, 2/6, 3/6, 4/6, 5/6\}$. Output values are interpreted as gray values, yielding the gray level reconstruction. If necessary, these values can be rounded to 0/1-values for a crisp reconstruction.

The networks have one hidden layer, that consists of two types of nodes:

global nodes that are connected to all input nodes and all output nodes; they can keep track of the global constraints;

local nodes that are connected to a few input nodes and output nodes; they keep track of the constraints that affect a pixel and its immediate neighbours.

There are n^2 local nodes, each corresponding with a unique pixel. Such a local node receives input from the 4 line projections that intersect with the pixel, and is connected to the 9 output nodes corresponding with the pixel and its immediate neighbours (6 or 4 near the boundaries). The number of global nodes, k, plays a key role. The total number of weights, including those from the bias nodes, is $(k+15)n^2 + (6k-12)n - (k-4)$ (for the bias nodes: $2n^2 + k$; for the global nodes: $k(n^2+6n-2)$; for the local nodes: $13n^2 - 12n + 4$), which is approximately $(k+15)n^2$ or $(k+1)n^2$ if no local nodes are used. The network architecture is depicted in Figure 3.

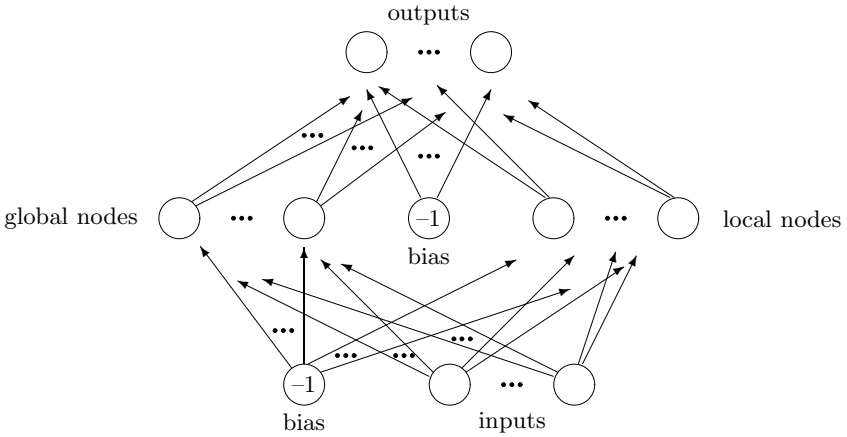

Figure 3: Neural network architecture

Training is performed as follows. In each epoch, a number (50,000, say) of random pictures with their projections are presented to the network. The pictures are sampled from a certain distribution, see later. Note that samples are used only once, unless they are by chance regenerated. The weights are adapted using the normal back-propagation rule. After each epoch the learning rate α is somewhat decreased. The nodes all use the standard sigmoid $\sigma\colon x \mapsto 1/(1+e^{-x})$.

3 Hopfield network approach

A Hopfield network is a fully connected recurrent neural network with neurons taking values from the set $\{-1, +1\}$. When using the Hopfield network for discrete tomography, the network has n^2 neurons; each neuron corresponds to a pixel of the image which can be either black $(+1)$ or white (-1).

The net input h_i of neuron i (with activation s_i) is given by $h_i = \sum_{j \neq i} w_{ij} s_j + \theta_i$, where w_{ij} is the weight associated with the connection between neuron i and neuron j, and θ_i is a real constant for neuron i: the threshold value. The weights in a Hopfield network are always symmetric, i.e., $w_{ij} = w_{ji}$ for all i, j. The deterministic dynamics of the network are given by the following update rule:

$$s_i' = \begin{cases} 1 & \text{if } h_i > 0 \\ -1 & \text{if } h_i < 0 \\ s_i & \text{if } h_i = 0 \end{cases} \tag{1}$$

Hopfield networks are well-known for their ability to act as an associative memory when combined with Hebbian learning. They are also capable of solving combinatorial optimization problems,

using the concept of an *energy function* E associated with the network (first proposed in [7]):

$$E = -\sum_{i<j} w_{ij} s_i s_j - \sum_i \theta_i s_i \tag{2}$$

The energy function has the property that every single neuron update either decreases the energy or leaves the energy unchanged. This property ensures that the network will converge to a local minimum of the energy function in a finite number of steps regardless of the initial state, when the neuron updates are carried out asynchronously.

For the discrete tomography problem it is possible to construct a similar energy function (containing terms that are at most quadratic in the variables of the problem) such that a minimum of the energy function over the set of possible states of the Hopfield network corresponds to an optimal solution of the tomography problem. Let $\delta(i)$ be the 4-neighbourhood of pixel i, i.e., the set of direct horizontal and vertical neighbours of pixel i. Define (using the notation from Section 1)

$$E = \sum_{k=1}^{m} \Big(\{ \sum_{i=1}^{n^2} A_{ki} x_i \} - b_k \Big)^2 + \lambda \sum_{i=1}^{n^2} \sum_{j \in \delta(i)} (x_i - x_j)^2 \quad , \tag{3}$$

where λ is a positive constant ($\lambda = 4$ in the experiments). The variables x_i in the energy function are the binary pixel values of the image. The first term corresponds to the difference of the projections Ax of an image x to the prescribed projections b, the second term corresponds to the "local smoothness" of the image x: as a form of regularization, we prefer reconstructions that are locally smooth.

The *network neighbourhood* of neuron i, i.e., the set of all neurons j for which w_{ij} is nonzero, is star-shaped, see Figure 4. It consists of the neurons that correspond to pixels which either share a projected line with pixel i or are in the 4-neighbourhood of pixel i.

The pixel values x_i are binary variables. These variables can be converted to variables s_i having domain $\{-1, +1\}$ by taking $x_i = (s_i + 1)/2$. This yields an expression of the form (2), from which all weights w_{ij} and thresholds θ_i of the Hopfield network can be found.

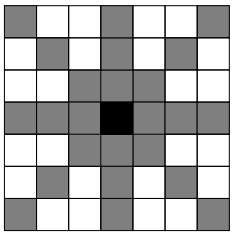

Figure 4: Network topology

If we start the Hopfield network with the given weights and let it converge to a local minimum of the energy function, it will find a locally optimal solution to the tomography problem. There is no guarantee that the network will converge to a global optimum, however.

When the updates of several neurons are carried out simultaneously (i.e., synchronous updates), convergence is no longer guaranteed. The network state may also reach a cycle. In the experimental results we will show that when the neuron activations are split into several random subsets (e.g., 10 subsets) and each subset is updated synchronously, the network converges almost as fast as for the asynchronous case, for all test cases.

4 Experiments

In this section we present reconstruction results for images of size 10×10, 25×25 and 50×50. All test images consist of 5 ellipses. Overlapping parts of ellipses can be either "or-ed" (yielding a union of ellipses) or "xor-ed" (pixels being black when contained in an odd number of ellipses), the latter version leading to more complex images. We performed all tests with both classes of images.

We start by giving some illustrative results for the feed-forward network of Section 2 for 25×25 images from the "or-class". The network used had 600 global hidden nodes and 625 local hidden nodes. After 50 epochs, each using 50,000 random training examples, the average number of incorrect pixels for the crisp reconstructions in a random test set of 100 pictures was 18.97, the mean squared gray level error being 15.05. The training took around 10 hours on a Pentium 4 2.8 GHz. The best crisp reconstruction from the test set (Figure 5, left) had an error of 3 pixels, the worst (Figure 5, right) achieved an error of 48 pixels. In Figure 6 we see two more "common" reconstructions: left an image with crisp error 14, right an image with crisp error 15.

We used the same settings for the "xor-class", which also led to the "same" test set, except for the fact that the ellipses are now "xor-ed".

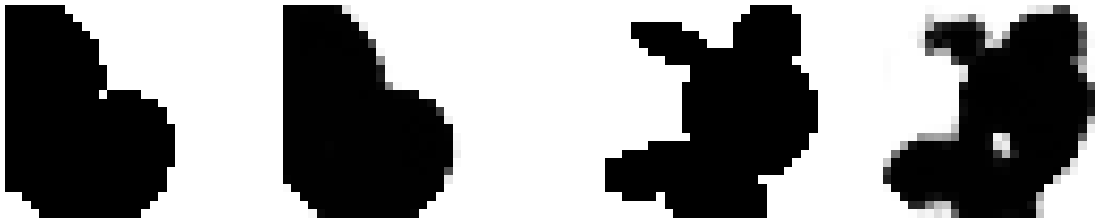

Figure 5: From left to right: original and gray level reconstruction with squared error 3.09; original and gray level reconstruction with squared error 36.81

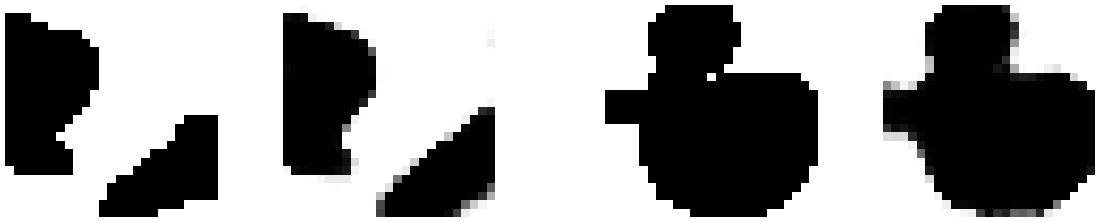

Figure 6: From left to right: original and gray level reconstruction with squared error 12.06; original and gray level reconstruction with squared error 14.24

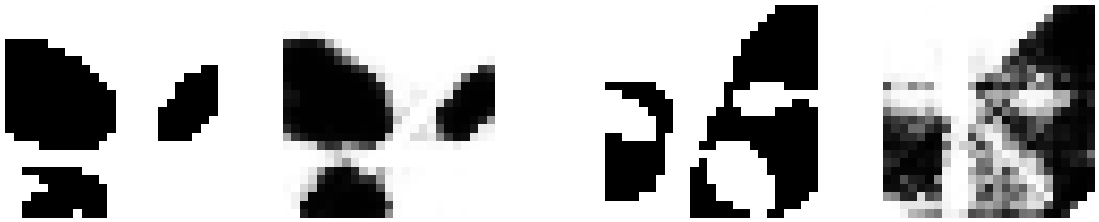

Figure 7: From left to right: original and gray level reconstruction with squared error 10.67; original and gray level reconstruction with squared error 65.96

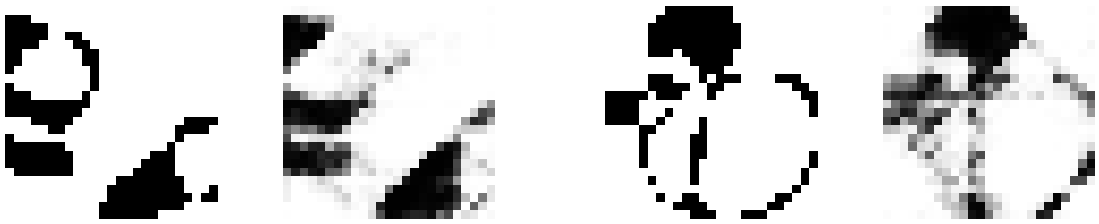

Figure 8: From left to right: original and gray level reconstruction with squared error 33.12; original and gray level reconstruction with squared error 45.19

After 50 epochs the average crisp error was 47.89. The mean squared error for the gray level reconstructions was 35.21. The best crisp reconstruction from the test set (Figure 7, left) had an error of 12 pixels, the worst (Figure 7, right) achieved an error of 92 pixels. In Figure 8 we see reconstructions of the "same" images (now with "xor-ed" ellipses) as in Figure 6: left an image with crisp error 48, right an image with crisp error 59.

Table 1 shows more extensive experimental results. For each test case, we performed three training runs of the network. The table shows the average results over the three runs, using a test set of 100 images. The column "#global" shows the number of global nodes in the network. The column "local?" indicates if local nodes are used. The column "p.e." shows the average error per pixel in the unrounded network output. The column "c.p.e." shows the average error in the crisp output and the column "sqr.pr.e." shows the sum over all projected lines of the squared error in the projection of the crisp output. Note that the integer valued projections were re-scaled to reals in $[0, 1]$.

			or			xor		
size	#global	local?	p.e.	c.p.e.	sqr. pr.e.	p.e.	c.p.e.	sqr. pr.e.
10×10	10	yes	0.026	0.020	0.0018	0.084	0.061	0.0044
	10	no	0.086	0.063	0.0079	0.206	0.140	0.0194
	30	yes	0.018	0.014	0.0012	0.069	0.051	0.0033
	30	no	0.035	0.027	0.0025	0.103	0.075	0.0066
	60	yes	0.013	0.010	0.0008	0.069	0.051	0.0033
	60	no	0.014	0.011	0.0009	0.065	0.048	0.0032
	120	yes	0.008	0.006	0.0005	0.060	0.043	0.0029
	120	no	0.012	0.009	0.0008	0.066	0.048	0.0033
25×25	25	yes	0.051	0.038	0.0028	0.112	0.080	0.0057
	25	no	0.134	0.096	0.0170	0.196	0.135	0.0210
	75	yes	0.045	0.034	0.0022	0.098	0.071	0.0044
	75	no	0.085	0.061	0.0077	0.143	0.101	0.0100
	150	yes	0.041	0.030	0.0020	0.093	0.067	0.0037
	150	no	0.068	0.050	0.0049	0.124	0.088	0.0074
	300	yes	0.035	0.026	0.0017	0.083	0.060	0.0029
	300	no	0.060	0.053	0.0040	0.120	0.109	0.0066
	600	yes	0.035	0.026	0.0016	0.077	0.056	0.0024
	600	no	0.052	0.049	0.0032	0.112	0.108	0.0058
50×50	50	yes	0.055	0.041	0.0044	0.149	0.104	0.0165
	50	no	0.271	0.180	0.0787	0.270	0.177	0.0478
	150	yes	0.051	0.038	0.0035	0.128	0.092	0.0095
	150	no	0.198	0.139	0.0489	0.208	0.143	0.0265
	300	yes	0.049	0.037	0.0031	0.120	0.086	0.0071
	300	no	0.114	0.096	0.0176	0.179	0.125	0.0178
	600	yes	0.043	0.032	0.0024	0.110	0.080	0.0057
	600	no	0.085	0.078	0.0095	0.146	0.139	0.0109

Table 1: Experimental results for the feed-forward network approach

Table 2 shows the results of the Hopfield network for the same test set, for the case of asynchronous updates and for the case of synchronous updates. The column "parallel?" shows if neuron updating is performed sequentially or in parallel (each time updating 1/10th of all neurons). The column "%perfect" indicates the percentage of perfectly reconstructed images in the test set. The number of iterations of the network (in one iteration all neurons are updated once) is shown in the column "it.".

		or			xor		
size	parallel?	%perfect	it.	p.e.	%perfect	it	p.e.
10×10	no	87	4.56	0.0052	28	5.96	0.0640
10×10	yes	87	4.56	0.0052	27	6.58	0.0594
25×25	no	71	8.88	0.0075	5	11.40	0.0902
25×25	yes	72	10.00	0.0056	1	12.34	0.0812
50×50	no	84	14.36	0.0028	0	18.44	0.1157
50×50	yes	79	17.13	0.0078	0	24.18	0.1050

Table 2: Experimental results for the Hopfield network approach

5 Conclusions and future research

For the "or-class" of test cases, both approaches perform well. The Hopfield network is even capable of finding a perfect reconstruction in most of the test cases, due to the smoothness of the images. The average pixel error is also extremely small for the Hopfield network.

The "xor-class" of test cases yields quite different results. For this class the Hopfield network is not very effective, it tends to get stuck in a local optimum of the energy function. This can be explained by the fact that images in the "xor-class" are far less smooth than those in the "or-class". The preference for smooth images that is used in the energy function of the Hopfield network is hardly justified in this case. A drawback of the Hopfield approach is the inability to adjust to new classes of images by learning. The feed-forward network, however, is capable of computing much better reconstructions for the "xor-class". Still, the error is significantly larger compared to the "or-class".

The results show clearly that using local nodes in the feed-forward network has a very positive effect on its reconstruction performance. Adding local nodes hardly affects the training time of the network, because each of these nodes only has a constant, small number of connections to the input and output layer. Apparently, significant benefits can be gained by changing the network topology from the standard, fully-connected topology. In future research, we intend to perform experiments with several alternative topologies, and different types of real-world images. The fact that once the network is fully trained (which may take very long), a nearly instantaneous reconstruction method is available (no retraining is necessary), makes the approach very interesting for applications where real-time reconstructions are important, such as medical imaging.

For the Hopfield network, the results of the sequential implementation are not very different from those of the parallel implementation, showing that the approach allows a high degree of parallelisation. It is well suited for a hardware implementation, where a single (simple) hardware processor can be assigned to each neuron.

References

[1] K.J. Batenburg, An Evolutionary Algorithm for Discrete Tomography, *Discrete Applied Mathematics*, 2005 (to appear).

[2] K.J. Batenburg, J.R. Jinschek and C. Kisielowski, Atomic Resolution Electron Tomography on a Discrete Grid: Atom Count Errors, *Microscopy and Microanalysis*, Vol. 11, Suppl. 2, 2005.

[3] K.J. Batenburg and W.A. Kosters, A Discrete Tomography Approach to Japanese Puzzles, *Proceedings of BNAIC 2004*, 243–250.

[4] C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[5] R.J. Gardner, P. Gritzmann and D. Prangenberg, On the Computational Complexity of Reconstructing Lattice Sets from their X-rays, *Discrete Math.* 202, 1999, 45–71.

[6] G.T. Herman and A. Kuba, eds., *Discrete Tomography: Foundations, Algorithms and Applications*, Birkhäuser, Boston, 1999.

[7] J.J. Hopfield and D.W. Tank, Neural Computation of Decisions in Optimization Problems, *Biological Cybernetics* 52, 1985, 141–152.

[8] J.R. Jinschek, H.A. Calderon, K.J. Batenburg, V. Radmilovic and C. Kisielowski, Discrete Tomography of Ga and InGa Particles from HREM Image Simulation and Exit Wave Reconstruction, *MRS Proceedings* 839, 2004, 4.5.1–4.5.6.