Discrete Tomography: A Neural Network Approach

Jonathan K. Vis [a] Walter A. Kosters [a] K. Joost Batenburg [b,c]

[a] *Leiden Institute of Advanced Computer Science, Universiteit Leiden, The Netherlands*
[b] *Centrum Wiskunde & Informatica, Amsterdam, The Netherlands*
[c] *Vision Lab, Universiteit Antwerpen, Belgium*

Abstract

Tomography tries to reconstruct an object from a number of projections in multiple directions. There are many obvious application domains, but we will focus on high throughput applications, and will therefore try to reduce the number of necessary projections, while being able to generate good quality reconstructions. We apply several forms of Neural Networks, an Artificial Intelligence method. These networks are especially suited for solving underdetermined problems, and therefore well suited to our problem.

Many different variants of Neural Networks are developed since its introduction; some simple, while other architectures can consist of many nodes in many hidden layers increasing the training complexity. We will here focus on the simpler forms of Neural Networks: feedforward (multilayer) perceptrons.

We show, for both artificial and real-life data, that these networks are capable of creating good quality reconstructions from a limited set of projections, while avoiding image artifacts that are often present in traditional approaches.

1 Introduction

Tomography, or more especially computed tomography, is a technique used in a broad variety of research areas: from medical to industrial, and archaeological to material studies. It can be applied to investigate (non-invasively) the internal structure of many different types of objects and materials. Probably the most recognized application is the X-ray CT scanner, for diagnostic purposes, as is found in many hospitals. The main idea of tomography is to be able to visualize and analyze the internal structure of an object.

The object is examined in various orientations leading to so-called *projections*, and when put together, a reconstruction is made. Typically, there is a need for a large number of projections (more than 100) to reconstruct, with adequate quality, an object. This approach, however, has many drawbacks. In areas where a high throughput is required the time needed to generate the projection data is limited, and when examining organic tissue only a limited dose of radiation might be administered without the risk of affecting the tissue. Therefore, we notice a need of good quality reconstructions based on a limited set of projections.

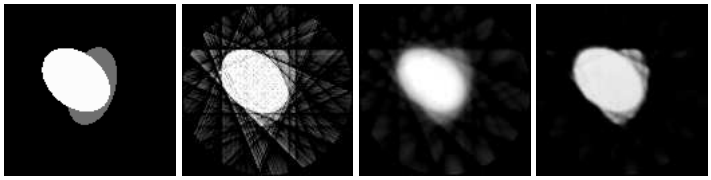


Figure 1: Examples of tomographic reconstructions. From left to right: original 128×128 image, filtered back projection reconstruction, linear perceptron, and multilayer reconstruction with 8 projection angles.

Traditional reconstruction methods tend to generate better quality reconstructions when increasing the number of projections. Furthermore, they are static and general, i.e., they cannot be adapted to a specific application domain. Here, we propose a different strategy. We apply Neural Networks, an Artificial Intelligence

method [12, 5], for generating reconstructions. In Figure 1 some examples of tomographic reconstructions are shown. Neural Networks must be trained (which implies a one time increase in effort), but carry the advantage of being capable of reconstructing specific images, and improving themselves. In this paper we show the potential of single-pixel networks for this purpose, in particular for situations with very few projection angles, also reducing image artifacts.

The paper is organized as follows. Section 2 contains related work. In Section 3 we introduce the theory of (discrete) tomography. Section 4 describes the various Neural Network topologies and approaches. We present the experiments in Section 5 and the conclusions to the study in Section 6.

2 Related Work

Computed tomography is a well studied field, and there are many publications describing many of its aspects. In [8, 9, 4, 7] the fundamentals of computed tomography, as well as many technical aspects are covered.

The application of Neural Networks is a relatively new approach. In literature they are introduced as a reconstruction technique in [10, 11]. In general, Neural Networks seem an uninteresting strategy for the general problem of tomography due to their nature of dealing with large numbers of projections, and consequently the large number of variables. It seems quite hard to outperform traditional reconstruction techniques. However, we will here focus on tomography problems consisting of a small set of projections resulting in an underdetermined problem. Neural Networks are well-known for their successful application to underdetermined problems.

In [2, 3, 1] the authors introduce Neural Networks successfully for reconstruction binary images (i.e., black and white). Two different network topologies are investigated: a full-image network, and a single-pixel network. The first variant tries to reconstruct a complete image at once from all projection data. The second variant reconstructs one pixel from a selection of the projection data. Based on their conclusions we will here focus on the single-pixel network topology. The networks used in [3] are quite large, consisting of 50–200 hidden nodes. Here, we will use much smaller networks. The disadvantage of applying a single-pixel network is its reduced ability to be trained for specific image classes. This property is, to a much greater extend, available in full-image networks at the expense of increased computational complexity.

3 Tomography

In *tomography*, we try to reconstruct an object from a number of *projections* in multiple directions. Here, we will focus on projections obtained by parallel beams through a finite object. We assume that this object is contained in the disc

$$A = \left\{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \leq R^2\right\} \tag{1}$$

with radius $R > 0$, see Figure 2. The object is an image described by the real-valued grayscale mapping $f : A \to [0, 1]$ where 0 is black and 1 is white; intermediate values can be interpreted as shades of gray.

The attenuations of the beams are measured on an infinite *detector*. Different projections are generated by rotating the detector around the object. The construction of the projections is performed by the so-called *Radon transform*, which is the integral transform of the function f over straight lines L:

$$R_f(L) = \int_L f(\ell)\ d\ell. \tag{2}$$

For angle θ we define: $L_{\theta,\tau} = \{(x, y) \in A : \tau = t\}$, with $t = x\cos\theta + y\sin\theta$. The Radon transform P_f of the function f is defined as:

$$P_f(\theta, \tau) = \int_{L_{\theta,\tau}} f(x, y)\ ds \qquad \text{for } \theta \in [0, \pi), \tau \in \mathbb{R}. \tag{3}$$

The reconstruction of the original image from its projections is obtained from applying the *inverse Radon transform* [6]:

$$f(x, y) = \int_{\theta=0}^{\pi}\int_{\tau=-\infty}^{\infty} h(\tau - t)\, P_f(\theta, \tau)\ d\tau\, d\theta, \tag{4}$$

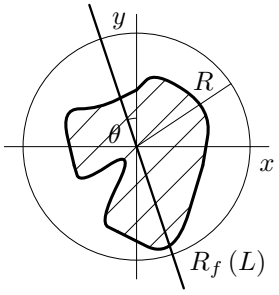


Figure 2: The basic principle of tomography. The hatched area only defines the outline of the object, and not its internal structure.

where h is a suitable weight or *kernel* function acting as a filter. Several kernel functions can be used. Often the so-called *Ram-Lak* kernel, only defined in the integer domain, or *ramp* filter is applied:

$$h(\sigma) = \begin{cases} \frac{\pi}{4} & \text{if } \sigma = 0, \\ -\frac{1}{\pi^2 \sigma^2} & \text{if } \sigma \text{ is odd}, \\ 0 & \text{otherwise}, \end{cases} \tag{5}$$

where $\sigma \in \mathbb{Z}$. Note that the kernel is symmetric around 0, as expected, see Figure 7(b) in Section 5.

Discrete tomography focuses on the reconstruction of images, which are reconstructed using a small, discrete set of pixel values, e.g., a binary image.

To find a discrete approximation, we substitute the integrals for summations. First, we choose a fixed number of angles k (equally dividing the 0 to π semicircle). And secondly, we approximate the remaining integral by choosing a finite detector size, D, so $h(\tau') = 0$ when $|\tau'| > D$, where $\tau' = \tau - t$:

$$f(x, y) = \sum_{\tau'=-D}^{D} h(\tau') \sum_{d=1}^{k} P_f(\theta_d, \tau' + t). \tag{6}$$

In tomography the calculation of Equation (6) is usually performed by the *filtered back projection* algorithm. In practical applications, filtered back projection is implemented by calculation via the *frequency domain*, or more especially the *Fourier domain*. In the Fourier domain, the convolution operator translates to a much simpler multiplication, and therefore reduces the computational complexity.

Here, we will not use the Fourier domain, but rather calculate the convolution of the kernel with the projection data in the *spatial domain*. As is suggested in Equation (6) we choose the kernel to be static, and consequently shift the projection data relative to the center of the kernel. Therefore we introduce a *shift operator* which aligns the projection data for a certain image pixel x, y with corresponding τ', and a certain angle θ_d. The shift amount is denoted by t. This implies, on a finite detector, that some projection data will be shifted outside the detector range, and some "new" projection data is shifted onto the detector. We will deal with this phenomenon as follows: the data shifted out of range is discarded, and the new data is treated as being 0, as it would be on an infinite detector.

Some difficulties arise in applying the discrete version of the Radon transform. In general, a pixel will not be mapped to a single pixel on the detector, it will instead be mapped between two pixels. Now, we have to decide how the image pixel will contribute to the possible pixels on the detector. Several strategies can be applied. Roughly, they can be divided into two categories. First, a single pixel on the detector receives the full contribution of an image pixel, for example, the *nearest neighbor* approximation. Secondly, we can distribute the contribution of an image pixel over the detector pixels, as is, for instance, done by *linear interpolation*.

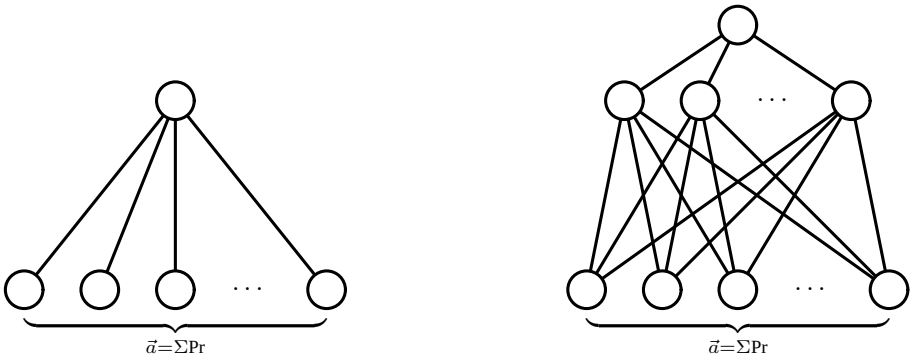
In order to increase the accuracy of the projections further, and, ultimately, the quality of the reconstruction, we apply a *subsampling* technique. This strategy is especially beneficial for images of small dimensionality. Each pixel is divided into $m \times m$ *subpixels*, where m is the *sampling rate*, with integer $m > 1$. Typically, $m = 2$ or $m = 3$. Each subpixel is then projected onto the detector.

4 Neural Networks

An *(Artificial) Neural Network* is a computational model that is inspired by the structure of a biological neural network such as the human brain [12, 5]. It consists of interconnected neurons passing information to each other. The structure is often adaptive based on internal or external data. This concept is referred to as *learning*. Many different forms exist today. Here, we will first focus on a simple form of a feedforward network called a *perceptron*.

4.1 Topologies

The computational properties of a linear perceptron, see Figure 3(a), show a remarkable similarity with the computations in Equation (6). We expect a linear perceptron to be able to simulate these computations. The



(a) Linear perceptron with N input nodes, and one output node. The total number of weights is $N + 1$. The bias node and weight are not depicted, but assumed to be present within the output node.

(b) Multilayer perceptron with N input nodes, h hidden nodes, and one output node. The total number of weights is $h\,(N + 1) + h + 1$ (including weights for the bias nodes). The bias nodes and weight are assumed to be present in each hidden and output node.

Figure 3: Two Neural Network topologies. Left: a linear perceptron and right: a multilayer perceptron.

weights should form the kernel function, and, for many projections, we expect the weight vector to be very similar to the Ram-Lak kernel.

The first topology consists of only one output node capable of reconstructing one pixel of the image. In order to reconstruct the complete image we apply the perceptron N^2 times. The input for reconstructing a single pixel is heavily dependent on that pixel. Therefore, we preprocess the Radon projection data depending on a certain pixel. The projection data is "shifted" such that the projection is centered around τ. Missing values are replaced by zeros (as it would be on an infinite detector). The input vector is $\vec{a} = \Sigma\text{Pr}$, the symbolic notation for the precomputed summation for all projection angles d according to Equation (6). Note that the emphasis on the single pixel topology and the aggregation of the projection data partly eliminates the expected benefits of the classification power of the Neural Network. The input vector for a certain pixel carries little information about other pixels in the image, thereby increasing the difficulty of learning image class specific features.

A *multilayer perceptron* is a feedforward network organized in multiple layers (an input and an output layer, as well as (several) hidden layers), see Figure 3(b). Each layer is fully connected to the next. In contrast to the perceptron model each node has a nonlinear activation function. In this case we will use the logistics function $\phi\,(x) = 1/\left(1 + e^{-\beta x}\right)$, with $\beta = 1$. It can be shown that each multilayer perceptron using only linear activation functions has an equivalent perceptron model.

4.2 Initialization

The initialization is not trivial and may have a huge impact in the convergence of the Neural Network. For linear perceptrons this effect is, generally, not too severe as the magnitude of the adjustment of the weights is always the same. A wrongly initialized network is likely to converge eventually. Note that initialization outside the interval $[-1, 1]$ may result in divergent behavior. In all cases, a suitable distribution must be chosen. Two distributions are commonly used; the uniform distribution, and the normal distribution. We will use the normal distribution. For a linear perceptron we fix $\mu = 0.5$, and $\sigma = 0.25$, where μ is the aimed for mean, and σ the standard deviation.

For nonlinear perceptrons, and consequently, multilayer perceptrons the initialization problem is harder. The introduction of nonlinearity is the main culprit. For commonly used activation functions (hyperbolic tangent and logistics) the behavior is similar. The use of the first derivative in updating the weights results in a very small update when the input of a node is very large or very small. Only in a small interval, controlled by the parameter β, the weight adjustments are similar to the linear case. Besides the danger of divergent behavior we face an additional hazard: being trapped in an inescapable region of the activation function. A good strategy is to assure that we start (from initialization) in the region where we have a linear-like behavior. We must be careful with adjusting parameter β as we might end up with a completely linear perceptron, and thereby reducing the additional capabilities of a nonlinear perceptron. We again fix $\mu = 0$ and $\sigma = 0.25$ for nonlinear perceptrons.

5 Experiments

In this section we describe several experiments, highlighting some examples. For a more comprehensive set of experiments see [13]. The first experiment aims to illustrate the statement that a linear perceptron is capable of simulating Equation (6). The second experiment describes a real-life case study. In all cases we use a custom developed C++ framework.

An *incremental training* approach is used. A single image of a particular image class is generated from which a number of pixels are selected randomly. Each pixel is a training instance. The weights are updated after each training instance. We call the usage of an image an *epoch*. In each epoch a number of pixels are selected as training instance. We refer to the number of selected pixels as *iterations*. Here, the number of epochs is fixed to $10{,}000$, and the number of iterations is fixed to $3{,}000$ (largely avoiding the problem of overfitting). A set of 50 predetermined images (for each image class) serves as *validation set*. The reported errors are measured on the validation set. The average absolute error is calculated as the absolute difference between each pixel in the original image with its corresponding reconstructed pixel, and averaged over the total number of pixels within disc A. Furthermore, the errors are averaged over the number of images in the validation set.

5.1 Artificial Image Classes

We construct a number of artificial image classes to experiment on. For all image classes we fix the dimensions to 128×128 pixels. The 2 ELLIPSES (OVERLAY) image class consists of two ellipses of random intensities (maybe with the same intensity) which may or may not (partially) overlap each other. The ellipses are drawn in a nondeterministic order. The last drawn ellipse determines the ultimate intensity. In [13], other artificial image classes are defined including random noise. The features of these images are always cropped to the disc A. The first three classes resemble objects consisting of larger homogeneous areas of constant graylevel, while the random noise images are mainly used to validate results. Samples of the image classes are presented in Figure 4.

Figure 4: Example images from five different image classes named (from left to right): 2 ELLIPSES (OVERLAY), 20 SMALL ELLIPSES (OVERLAY), 5 CONCENTRIC ELLIPSES (OVERLAY), RANDOM NOISE (1000), and RANDOM NOISE (10000).

5.2 Simulating Filtered Back Projection

The first experiment aims to prove the statement: A linear perceptron should be able to reconstruct an image from its Radon projections (for a large number of angles) as good as Equation (6).

As a training set we generate a set of $10{,}000$ (the number of epochs) random 128×128 images from the image class 2 ELLIPSES (OVERLAY). For each image we offer the Radon transform projections for $k = 32$, and $3{,}000$ randomly chosen pixels as target outputs for a total of $30{,}000{,}000$ training examples. In Figure 5, the differences in the reconstructions between filtered back projection and a linear perceptron are shown.

From Figure 5, some differences in the reconstruction images between the filtered back projection and the Neural Network approach can be observed. So-called *image artifacts* are present in the filtered back projection reconstruction giving the objects in the image a textured appearance, as well as "phantom" objects, see for example the second image from Figure 1. These artifacts are strongly oriented to the projection angles. These can be hard to eliminate in an automated way. This is especially true for $k < 32$. In the Neural Network generated reconstructions there are less image artifacts at the expense of softer boundaries of the objects. In Figure 5, we also include the multilayer (with two hidden nodes) reconstructions. However, we do not provide an extensive experimental comparison between the two network topologies. The reconstructions are much sharper defined and yield a lower average error compared to the linear perceptron case. The next section provides some arguments to prefer the latter in certain situations.

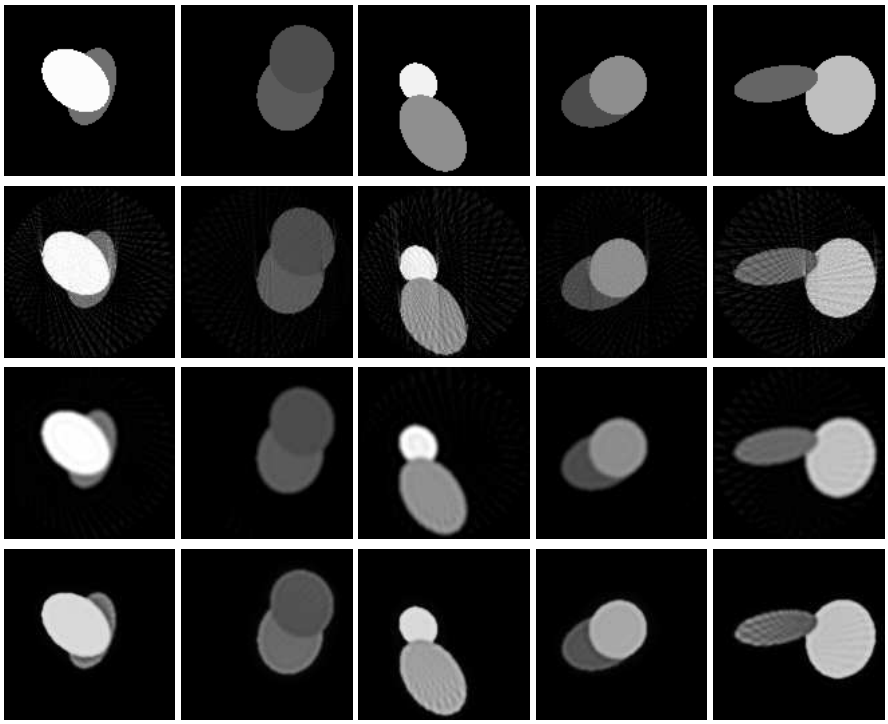


Figure 5: The original image (top row) and filtered back projection reconstruction (second row) versus a linear perceptron (third row) and a multilayer perceptron reconstruction (bottom row), all with 32 projections.

5.3 Real-life Case Study

The projection data for the real-life case study is not artificially created, but it is instead actual real-life output of a CT scanner. In this case a homogeneous crystalline object. The data set consists of 332 slices of the homogeneous crystalline object, see Figure 6. The dimensions of the images are originally 1024×1024 pixels. We reduced this dimensionality to 384×384 pixels for two reasons; first, the object is rather small compared to the full image size, and secondly, it reduces the computation times significantly. Per slice 500 projections are included (equally dividing the 0 to π semicircle).

In contrast to the earlier experiments, we have no original image to train on. We used the filtered back projection reconstruction using all 500 projections as an approximation of the original image, thereby providing a ground truth. Then we were interested in the reconstruction quality of a linear perceptron versus the traditional filtered back projection approach, of course using (much) fewer projections, e.g., 50.

We apply a linear perceptron for several reasons. First, it is the most simple topology having the fewest number of weights which makes it easy and fast to train. The initialization is easier because of its linearity. The second motivation regards the practical implementation. The weights resulting from a trained linear perceptron are assumed to be easily embedded within existing implementations.

For this experiment we randomly select $1{,}000$ slices, and from each selected slice we randomly select $10{,}000$ pixels as training example resulting in a total of $10{,}000{,}000$ training instances. The perceptron was trained using 50 projections equally dividing the total of 500 projections. The resulting reconstructions, as are shown in Figure 6, are softer, as is the case for the artificial image classes. In Figure 7(a) the resulting weight vector is shown. The symmetry can be clearly observed. The features are much less “sharp” as compared to for instance the Ram-Lak kernel in Figure 7(b). The resulting reconstructions, as are shown in Figure 6, are softer as well.

Table 1: Real-life average absolute errors.

	Average error	Standard deviation
Filtered back projection ($k = 50$)	0.1198	0.1262
Linear perceptron	0.0548	0.0632

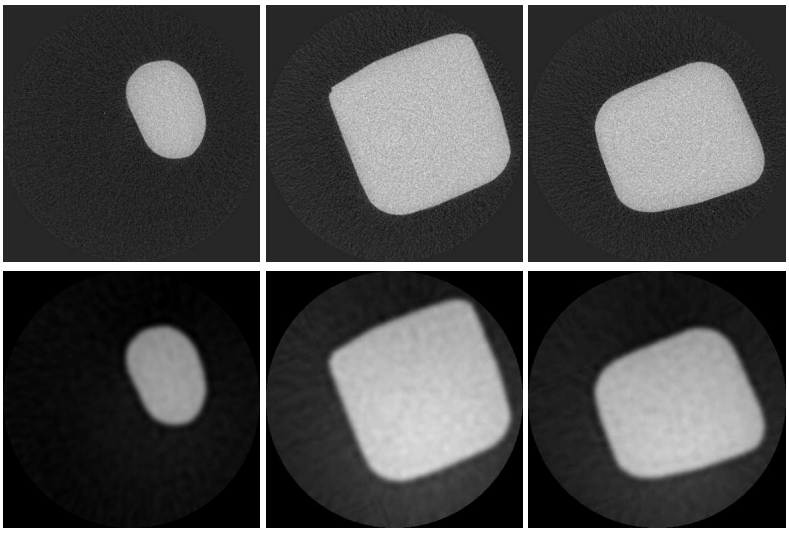
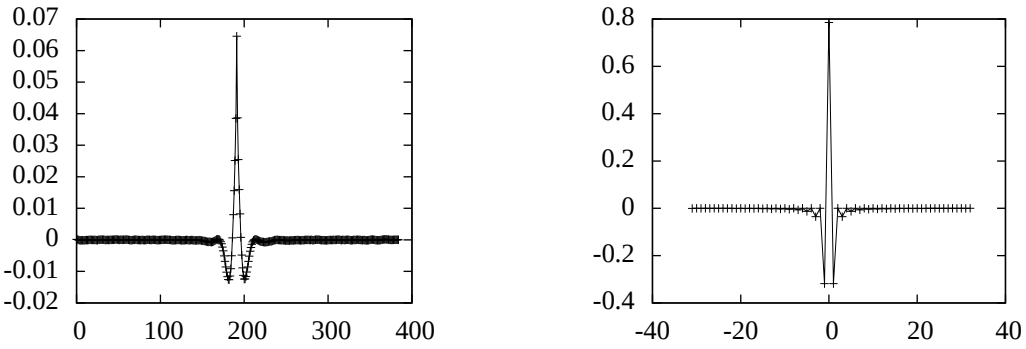


Figure 6: Top row: example images from the real-life data set: slices 50, 145, and 20 reconstructed using filtered back projection with 500 projections. Bottom row: Reconstructed image by a linear perceptron. Here 50 projection angles were used (equally dividing the semicircle), and the perceptron was offered a total of $10{,}000{,}000$ training instances. Note that only the pixels contained in disc A are reconstructed.

In Table 1 the average absolute errors are presented. The linear perceptron trained on the real-life data set performs best as can be expected. We observe about the same difference in reconstruction quality with regard to the filtered back projection algorithm as is observed on our artificial image classes reconstructions. We might conclude that a linear perceptron can be applied for the reconstruction of real-life objects.



(a) Weight vector of a linear perceptron after $10{,}000{,}000$ training instances.

(b) Ram-Lak kernel of size 64.

Figure 7: Weight vector of a linear perceptron after $10{,}000{,}000$ training instances. The discrete weights are connected by linear interpolation for better readability.

6 Conclusions and Further Research

We applied various Neural Networks to the problem of discrete tomography, where we focused on obtaining good quality reconstructions from a limited set of projections. Here, we present a summary of the conclusions from the experiments in Section 5. As a general rule, Neural Networks are capable of reconstructing very good quality images, especially when the image resolution is low, and there are only a few projections. They lose their advantage over the traditional filtered back projection technique when there are many (over 100) projections available. In theory, a linear perceptron is able to simulate the filtered back projection strategy, i.e., we can choose the weight vector to be identical to the kernel used. Training a network does not guarantee convergence to this kernel. In fact, it is quite hard to train a Neural Network (using the aggregation

approach) when many projections are used. The (aggregated) input vector shows little variation hampering its training abilities, and making the network very sensible to its initialization values.

Clearly, different network topologies are capable of reducing the average absolute error compared to the filtered back projection technique. From Figure 5, we can observe several differences of the approach of both techniques. While filtered back projection tends to recreate sharply defined boundaries around objects, the Neural Networks, especially the linear perceptron, make these edges softer. The reconstructions from filtered back projection suffer from many image artifacts, while they are almost absent in the reconstructions from the linear perceptron. It seems that the lower error values are mostly achieved by eliminating these artifacts, while losing some on the sharpness of the objects. This observation is supported by the, on average, higher error for the 20 SMALL ELLIPSES (OVERLAY) image class compared to other image classes with less ellipses, i.e., less boundaries.

The results from the real-life data set case study presented in Section 5.3 are encouraging. A linear perceptron is capable of generating high quality reconstructions of real-life data. It is very beneficial to train on the same class as the objects that will be ultimately reconstructed, however, due to the limited size of our real-life data set no hard conclusions can be drawn. The resulting weight vector from the trained linear perceptron can be easily transferred to existing practical implementations, and therefore, instantly improve reconstruction quality.

Many areas of future research remain. As we explored the behavior of simple Neural Networks (i.e., perceptrons) other topologies could be investigated, especially regarding the elimination of the aggregation operator.

References

[1] K.J. Batenburg. A Network Flow Algorithm for Reconstructing Binary Images from Discrete X-rays. *Journal of Mathematical Imaging and Vision*, 27(2):175–191, 2007.

[2] K.J. Batenburg and W.A. Kosters. Neural Networks for Discrete Tomography. In *Proceedings of the Seventeenth Belgium-Netherlands Conference on Artificial Intelligence (BNAIC)*, pages 21–27, 2005.

[3] K.J. Batenburg and W.A. Kosters. A Neural Network Approach to Real-Time Discrete Tomography. In *Combinatorial Image Analysis*, volume 4040 of *Lecture Notes in Computer Science*, pages 389–403. Springer Berlin/Heidelberg, 2006.

[4] M. Buzug. *Computed Tomography: From Photon Statistics to Modern Cone-Beam CT*. Springer Berlin/Heidelberg, 2008.

[5] S. Haykin. *Neural Networks and Learning Machines*. Prentice Hall, third edition, 2008.

[6] S. Helgason. *The Radon Transform*. Birkhäuser Boston, second edition, 1999.

[7] G.T. Herman. *Fundamentals of Computerized Tomography: Image Reconstruction from Projections*. Springer London, second edition, 2010.

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

[9] G.T. Herman and A. Kuba. *Advances in Discrete Tomography and Its Applications*. Birkhäuser Boston, 2007.

[10] J. Kerr and E. Bartlett. Neural Network Reconstruction of Single-photon Emission Computed Tomography Images. *Journal of Digital Imaging*, 8(3):116–126, 1995.

[11] J. Lampinen, A. Vehtari, and K. Leinonen. Application of Bayesian Neural Network in Electrical Impedance Tomography. In *Proceedings of the 1999 International Joint Conference on Neural Networks*, pages 3942–3947, 1999.

[12] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, third edition, 2010.

[13] J.K. Vis. Discrete Tomography A Neural Network Approach. Master's thesis, Leiden Institute of Advanced Computer Science, Universiteit Leiden, 2011.