International Journal of Computational Intelligence and Applications
Vol. 1, No. 4 (Special Issue on Neural Networks at IJCAI-2001) 427–438

LEARNING ITERATIVE IMAGE RECONSTRUCTION IN THE NEURAL ABSTRACTION PYRAMID

SVEN BEHNKE

Freie Universität Berlin, Institut für Informatik
Takustr. 9, 14195 Berlin, Germany
behnke@inf.fu-berlin.de

Received July 2001

Successful image reconstruction requires the recognition of a scene and the generation of a clean image of that scene. We propose to use recurrent neural networks for both analysis and synthesis.

The networks have a hierarchical architecture that represents images in multiple scales with different degrees of abstraction. The mapping between these representations is mediated by a local connection structure. We supply the networks with degraded images and train them to reconstruct the originals iteratively. This iterative reconstruction makes it possible to use partial results as context information to resolve ambiguities.

We demonstrate the power of the approach using three examples: superresolution, fill-in of occluded parts, and noise removal / contrast enhancement. We also reconstruct images from sequences of degraded images.

Keywords: Neural Abstraction Pyramid; Image Reconstruction; Backpropagation Through Time; Superresolution; Occlusion; Contrast Enhancement

1. Introduction

The quality of captured real world images is frequently not sufficient for the application at hand. The reasons for this can be found in the image formation process (e.g. occlusions) and in the capturing device (e.g. low resolution, sensor noise).

The goal of the reconstruction process is to improve the quality of measured images, e.g. by suppressing the noise. To separate noise from objects, models of the noise and the objects present in the images are needed. Then, the scene can be recognized and a clean image of that scene can be generated.

Hierarchical image decompositions using wavelets have been successfully applied to image denoising.[1,2] The image is transformed into a multiscale representation and the statistics of the coefficients of this representation are used to threshold them. The back-transformed images are then less noisy. Problematic with these approaches is that the choice of the wavelet transformation is usually fixed and the thresholding ignores dependencies between neighboring locations within a scale and between scales.

The recently proposed VISTA approach to learning low-level vision uses Markov random fields to model images and scenes. The parameters of these graphical models can be trained, e.g. for a superresolution task.[3] However, the models have no hidden variables and the inference via belief propagation is only approximate.

Sven Behnke, IJCIA 1(4), pp. 427-438, 2001.

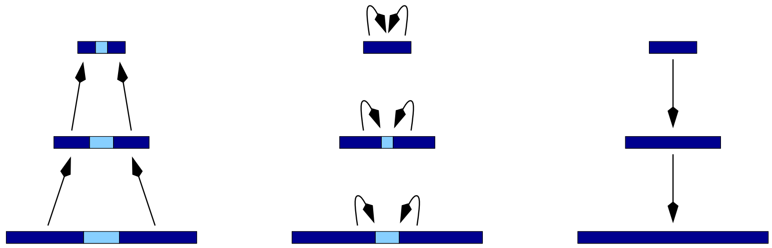

Fig. 1. Iterative image reconstruction. Partial image interpretations are used as context to resolve ambiguity.

Continuous attractor networks have been proposed to complete images with occlusions. [4] For digits belonging to a common class, a two-layer recurrent network was trained using gradient descent to reconstruct the original. The network had many adaptable parameters, since no weight sharing was used. Further, it was not demonstrated that the reconstruction is possible if the digit class is unknown. We extend the approach by adding lateral connections, weight sharing, and more layers to the network and train it to reconstruct digits from all classes without presenting the class label.

A common problem with image reconstruction is that it is difficult to decide locally about the interpretation of an image part. For example in a digit binarization task, it might be impossible to decide whether or not a pixel belongs to the foreground by looking only at the pixel's intensity. If contrast is low and noise is present, it could be necessary to bias this decision with the output of a line-detector for that location.

In general, to resolve such local ambiguities, a large context is needed, but feed-forward models that consider such a large context have many free parameters. They are therefore expensive to compute and difficult to train.

We propose to iteratively transform the image into a hierarchical representation and to use partial results as context. Figure 1 illustrates the propagation of information from regions that are interpreted easily to ambiguous regions. Further, we describe the reconstruction problem using examples of degraded images and desired output images and train a recurrent neural network of suitable structure to do the job.

The remainder of the paper is organized as follows: In the next section, the hierarchical architecture of the proposed recurrent networks is introduced. Section 3 discusses the supervised training of such networks. Experiments on four image reconstruction tasks are presented in Section 4.

2. Hierarchical Architecture

The neural abstraction pyramid architecture that has been introduced by Behnke and Rojas is a suitable framework for iterative image reconstruction. [5] Its main features are:

Pyramidal shape: Layers of *hypercolumns* are arranged vertically to form a pyramid (see Fig. 2). Each hypercolumn consists of a set of neural processing elements (*nodes*) with overlapping receptive fields that extract different features or *quantities*. The number of nodes per hypercolumn increases and the number of hypercolumns per layer decreases towards the top of the pyramid.

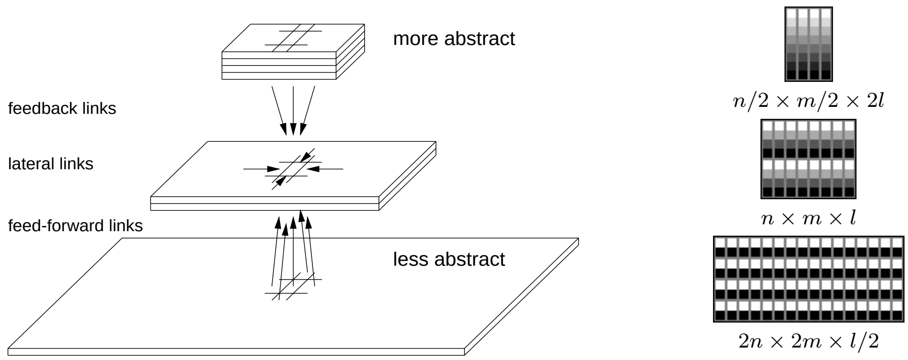

Fig. 2. Sketch of the recurrent network. When going up two layers the number of quantities per hypercolumn increases from $l/2$ to l to $2l$ as the spatial resolution decreases from $2n \times 2m$ to $n \times n$ to $n/2 \times m/2$.

Analog representation: Each layer describes an image in a two-dimensional representation where the level of abstraction increases with height, while spatial resolution decreases. The bottom layer stores the given image (a signal). Subsymbolic representations are present in intermediate layers, while the highest layers contain almost symbolic descriptions of the image content. These representations consist of quantities that have an *activity value* for each hypercolumn that is computed by a corresponding node.
Local interaction: Each node is connected to some nodes from its neighborhood via directed *weighted links*. The shared weights of all nodes that represent the same quantity are described by a common *template*. Links can be classified as:

- *feed-forward links:* perform feature extraction,
- *lateral links:* for consistent interpretation,
- *feedback links:* provide interpretation hypotheses.

Discrete time computation: A node's value for time step t depends only on the input values at $(t-1)$. All nodes are updated in parallel at each time step.

We use Σ-units as neural processing elements that compute the weighted sum of their inputs and apply a nonlinear output function. The update of the value $v_{x,y,z,q}$ of a unit at hypercolumn (x, y) in layer z for quantity q is done as follows:

$$v^{t+1}_{x,y,z,q} = \sigma \left[\sum_{j \in \mathcal{L}(i)} \mathcal{W}(j)\, v^t_{\mathcal{X}(j,x),\mathcal{Y}(j,y),\mathcal{Z}(j,z),\mathcal{Q}(j)} + \mathcal{B}(i) \right]. \tag{1}$$

The template $i = \mathcal{T}(z, q)$ is associated with quantity q at layer z. $\mathcal{L}(i)$ is the set of links of that template and $\mathcal{B}(i)$ is the template bias. $(\mathcal{X}(j,x), \mathcal{Y}(j,y), \mathcal{Z}(j,z), \mathcal{Q}(j))$ describe location and quantity of the input value for link j, and $\mathcal{W}(j)$ is the link weight. The output function $\sigma(x) = 1/(1+e^{-x})$ is here a sigmoidal function that limits the values to the interval $[0,1]$. In addition , a start value $\mathcal{V}^0(i)$ for initialization at $t = 0$ is needed for each template. The value of an input node at position (x, y, z, q) is set to a copy of the corresponding component $\mathcal{I}(x, y, z, q)$ of the input vector $\mathbf{x}_k$ of the current example k:

$$v^t_{x,y,z,q} = x^t_{k,\mathcal{I}(x,y,z,q)}, \text{ if } i = \mathcal{T}(z,q) \text{ is an input template.} \tag{2}$$

The feed-forward inputs of a node come from all quantities in a small window at the corresponding position in the layer $(z-1)$ directly below that node. Lateral connections link to all quantities in its neighborhood, including the node itself. Feedback links originate from the units in the layer above that correspond to the same position.

3. Training Recurrent Networks

Behnke proposed an unsupervised learning algorithm for the neural abstraction pyramid. [6] It learns a hierarchy of increasingly abstract representations of the image content that could be used to improve the quality of the images. Here, we apply supervised training to learn exactly the features that are needed to achieve the desired image reconstruction.

Training recurrent networks is difficult due to the non-linear dynamics of the system. Several supervised training methods have been proposed in the literature. Real-time recurrent learning (RTRL) [7] is suitable for continuously running networks, but very resource intensive. The backpropagation through time algorithm (BPTT) [8] unfolds the network in time and applies the backpropagation idea to compute the gradient of an error function. Its computational costs are linear in the number of time steps the error is propagated back.

For image reconstruction, we present a static input $\mathbf{x}_k$ to the network and train it to quickly reach a fixed point that coincides with the desired output $\mathbf{y}_k$. Thus, the network runs only a few iterations and the gradient can be computed efficiently. No artificial truncation of history is necessary.

3.1. *Objective Function*

The goal of the training is to produce the desired output $\mathbf{y}_k$ as quickly as possible. To achieve this, the network is updated for a fixed number T of iterations. The output vector $\mathbf{v}_k^t$ collects the output units of the network in an appropriate order. The output error δ_k^t, the difference between the activity of the output units $\mathbf{v}_k^t$ and the desired output $\mathbf{y}_k$, is not only computed at the end of the sequence, but after every update step. In the error function we weigh the squared differences progressively, as the number of iterations increases:

$$E = \sum_{k=1}^{K} \sum_{t=1}^{T} t^2 \|\mathbf{y}_k - \mathbf{v}_k^t\|^2. \tag{3}$$

A quadratic weight t^2 has proven to give the later differences a large enough advantage over the earlier differences, such that the network prefers a longer approximation phase if the final approximation to the desired output is closer. The contribution of intermediate output values to the error function makes a slight modification to the original backpropagation rule necessary. At all copies of the output units $v^t_{x,y,z,q}$ for $t < T$ the difference $\delta^t_{k,\mathcal{I}(x,y,z,q)}$ is computed and added to the backpropagated component of the gradient.

3.2. *Robust Gradient Descent*

Minimizing the error function with gradient descent faces the problem that the gradient in recurrent networks either vanishes or grows exponentially in time depending on the magnitude of gains in loops.[9] It is therefore very difficult to determine a learning constant that allows for both stability and fast convergence.

For that reason, we decided to employ the RPROP algorithm,[10] that maintains a learning constant for each weight and uses only the sign of the gradient to determine the weight change. The learning rates are initialized to a moderate value, increased when consecutive steps have the same direction, and decreased otherwise. We modify not only the weights in this way, but adapt the biases and start values as well.

The RPROP training method proved experimentally to be much more stable than gradient descent with a fixed learning rate. However, to compute the gradient, all training examples have to be presented to the network, which is slow for large training sets. To accelerate the training we implemented the following modification. We use as batch only a small working set of training examples that is initialized at random. After each weight update, a small fraction of the examples is replaced with randomly chosen examples to ensure a stable estimate for the gradient that takes over time all training examples into account. With a working set of 1% of 60.000 training examples we achieved a speedup of two orders of magnitude, as compared to the batch method, without compromising convergence.

4. Experimental Results

We conducted a series of experiments with images of handwritten digits to demonstrate the power of the proposed approach for iterative image reconstruction. The reason for choosing digits was that large datasets are publicly available and that the images contain multiscale structure which can be exploited by the learning algorithm. Clearly, if there were no structure to learn, the training would not help. We degraded the digits by subsampling, occlusion, or noise and trained recurrent networks to reconstruct the originals.

4.1. *Superresolution*

For our first experiment we used the original NIST images of segmented binarized handwritten digits.[11] The digits are given in a 128×128 window, but their bounding box is typically much smaller. For this reason, we centered the bounding box in a 64×64 window to produce the desired output Y. The input X to the network consists of 16×16 subsampled versions of the digits that have been produced by averaging 4×4 pixels.

The superresolution network has three layers, as shown in Figure 3. The low resolution image is input to the rightmost layer. Four 32×32 quantities represent the digit in the middle layer. They are connected to their 3×3-neighborhoods, to 2×2 windows of the output units, and to a single input node. The leftmost layer contains only the output units of the network, connected to four nodes in the middle layer and to their 3×3-neighborhoods.

We initialized the network's 235 free parameters randomly and trained it for ten time steps using 200 randomly chosen examples. As a test set, we used 200 different randomly chosen examples. Figure 4 shows for the first five test digits, how the output of the network develops over time. After two iterations the input can influence the output, but no further interactions are possible yet. In the following iterations the initial reconstruction

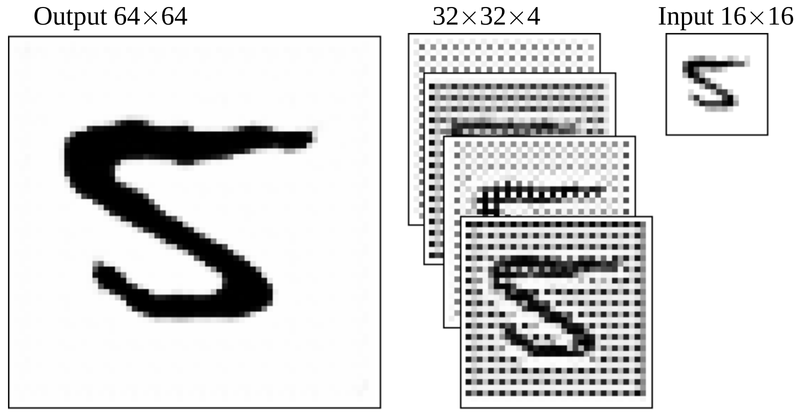

Fig. 3. Network for superresolution. Each pixel corresponds to a node that is connected to its neighbors. The topmost layer, shown on the right, contains the low resolution input. The middle layer contains four intermediate quantities. The bottom layer (left) contains the high resolution output quantity. Gray values show activities after the recurrent network has iteratively reconstructed the high resolution image.

is refined. The network tries to concentrate the gray that is present in the input images at black lines with smooth borders. To illustrate this behavior, we presented uniform pixel noise to the network. The stable response after ten time steps is shown in Figure 5. The network synthesizes smooth black lines at positions where many dark pixels are present.

We also trained a larger version of the recurrent network (RNN) that had eight hidden quantities in the middle layer as well as two feed forward neural networks (FFNN) with four and eight quantities. The units of the FFNNs looked at 3×3 windows of the previous layer such that the networks had a similar number of adjustable parameters as the corresponding RNNs. Figure 6 shows for the next five test digits the output of these four networks after 10 iterations. In general, the reconstructions are good approximations to the high resolution targets, given the low resolution inputs. In Figure 7 the mean square error of the networks is displayed. The test set reconstruction error of the recurrent networks decreases quickly and remains below the error of the corresponding FFNN after six time steps. At iterations 9 and 10 the small RNN outperforms even the large FFNN. When iterated beond the trained cycles, the reconstruction error slightly increases again.

4.2. *Fill-In of Occluded Parts*

For the second reconstruction experiment we used the MNIST database of handwritten digits.[12] The NIST digits have been scaled to the size 20×20 and centered in an 28×28 image. We set an 8×8 square to the value 0.125 (light gray) to simulate an occlusion. The square was placed randomly at one of 12×12 positions, leaving a 4 pixel wide border that was never modified.

The recurrent reconstruction network consisted of four layers, as illustrated in Figure 8. The first layer (28×28) contains the input image and the output units of the network. In the second layer four quantities with resolution 14×14 look at overlapping 4×4 windows

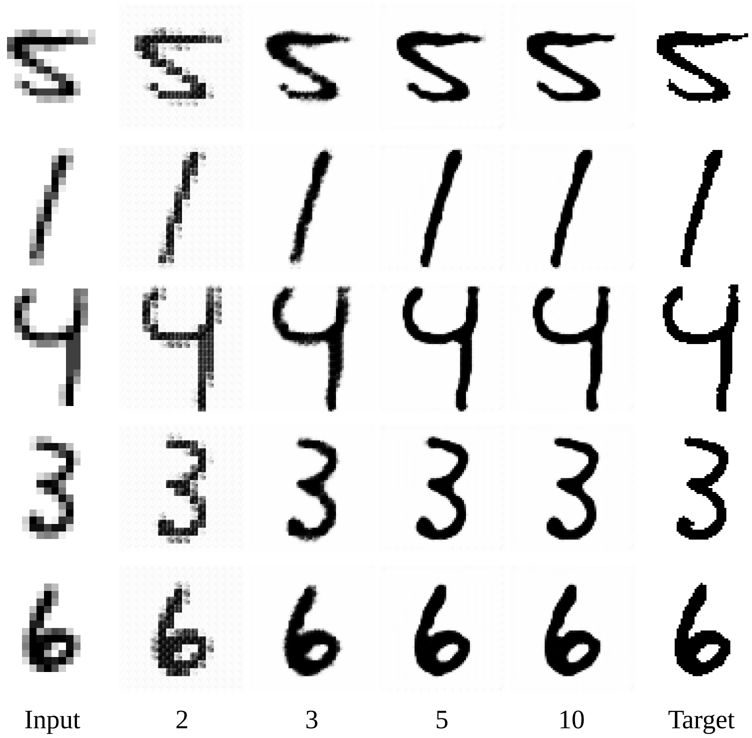

Fig. 4. Iterative superresolution.

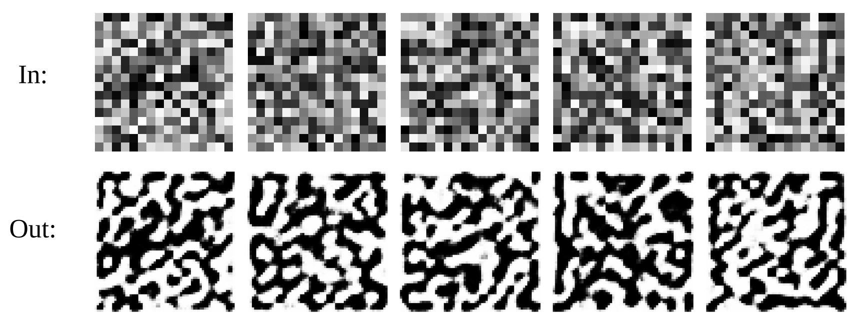

Fig. 5. Response of the superresolution network to uniform noise.

of the quantities below. The 16 quantities in the third layer have also 4×4 feed-forward connections, while in the top layer the resolution of the 64 quantities is reduced to 1×1 and the feed-forward weights are connected to all $7 \times 7 \times 16$ nodes of the third layer. The first three layers are surrounded by a one pixel wide border that is set to zero. In these layers the nodes have 3×3 lateral connections. In the fourth layer the lateral weights contact all 64 nodes. The feedback links are non-overlapping and have thus the size 2×2 between the first three layers and 7×7 between the third and the topmost layer.

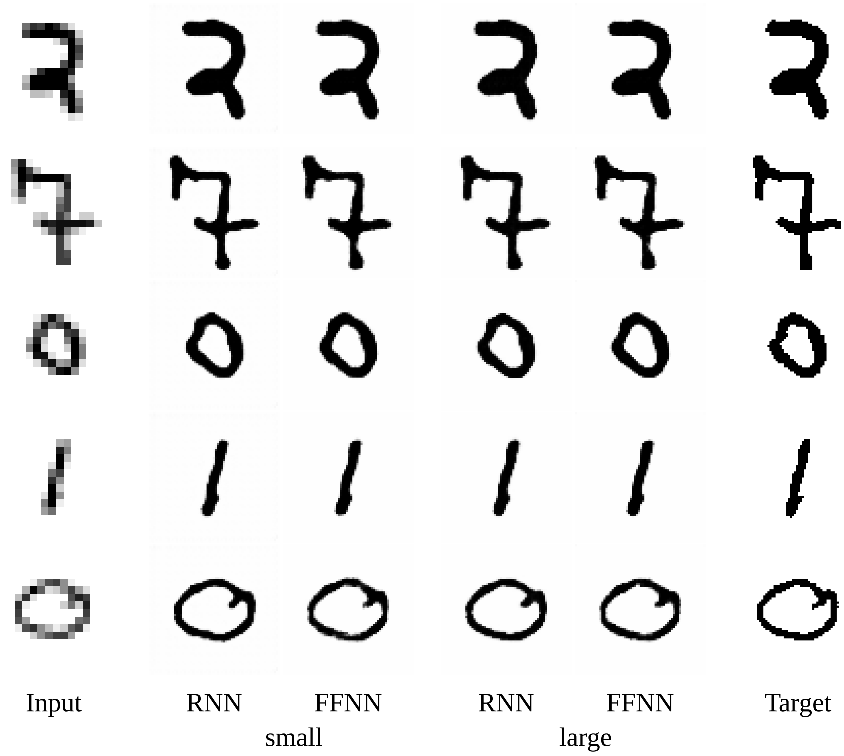

Fig. 6. Outputs of different superresolution networks.

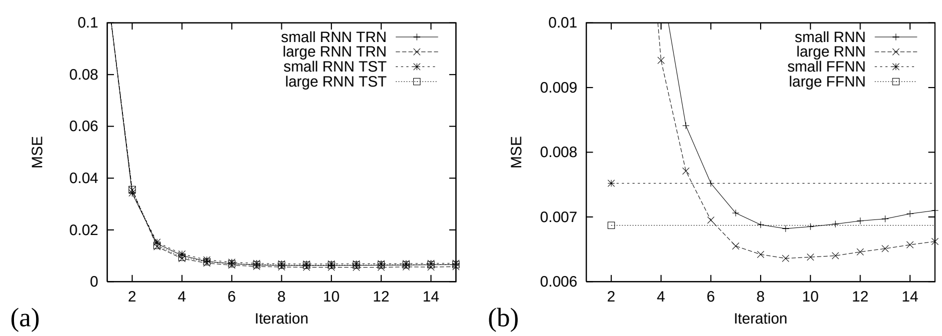

Fig. 7. Mean square error of superresolution: (a) the recurrent network on the training set and the test set; (b) detailed view of the test set performance, compared to FFNN.

Training is done with a working set of 600 of the 60.000 examples for twelve time steps. Fig. 9 displays the reconstruction process for the first ten digits of the test set. One can observe that the images change mostly at occluded pixels. This shows that the network recognized the occluding square. Further, the change is such that a reasonable guess is produced how the digit could look like behind the square. The network connects lines

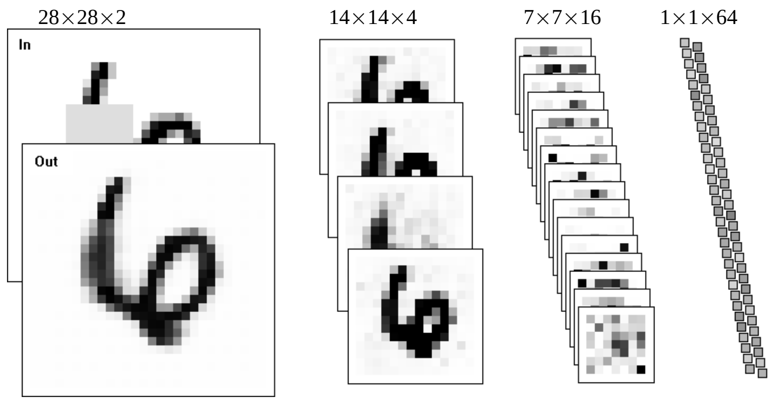

Fig. 8. Network for fill-in of occluded parts. Also used for contrast enhancement / noise reduction and for reconstruction from sequences of degraded images.

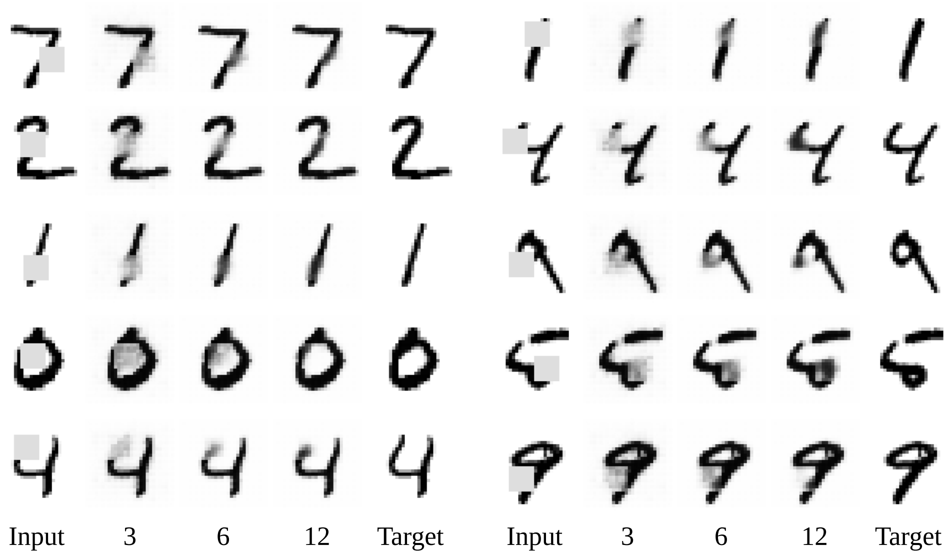

Fig. 9. Fill-in of occluded parts.

again that have been interrupted by the square. It is also able to extend shortened lines and to close opened loops. In most cases, the reconstructions are very similar to the originals.

4.3. *Noise Removal and Contrast Enhancement*

We used the same network architecture and the same MNIST digits to learn noise removal and contrast enhancement. We scaled the pixel intensities of the input digits to $[0.25, 0.75]$, added a random background level that was uniformly distributed in the range $(-0.25, 0.25)$, and added uniform pixel noise in the range $(-0.25, 0.25)$. Finally, we clipped the pixel values at $[0, 1]$. The first column of Figure 10 shows the first ten digits of the test set that have been degraded in this way. The network was trained on a working set of 600 out of 60.000 digits for twelve time steps.

The reconstruction process is also shown in Figure 10. One can observe that the network is able to detect the dark lines, to complete them, to remove the background clutter, and to enhance the contrast. The interpretation at most locations is decided quickly by the network. Ambiguous locations are kept for some iterations at intermediate values, such that the decision can be influenced by neighboring nodes. The reconstructed digits are very similar to the originals.

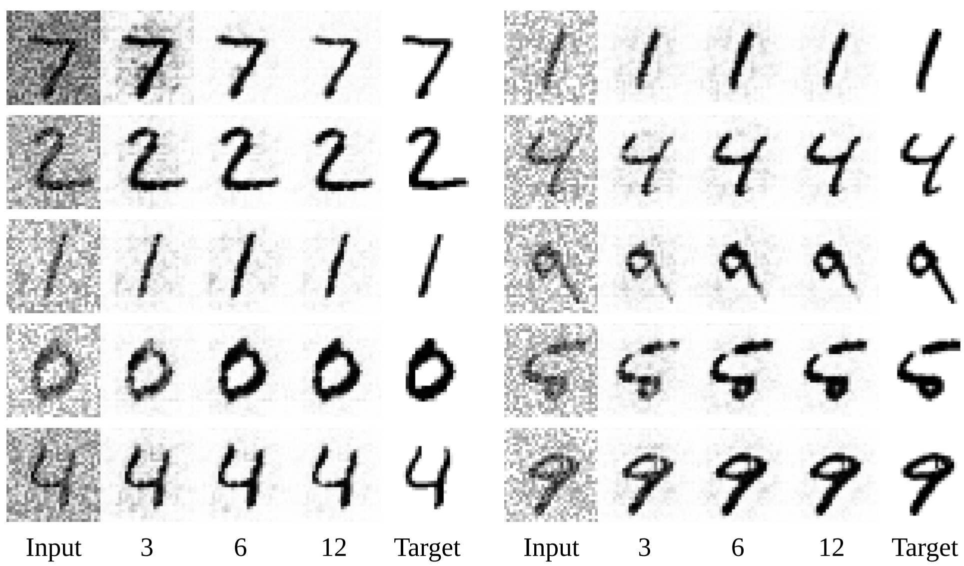

Fig. 10. Noise removal and contrast enhancement.

4.4. *Digit Reconstruction from a Sequence of Degraded Images*

Since the network is recurrent, it is able to integrate information over time. To test the digit reconstruction from a sequence of degraded images, we combined a random background level, contrast reduction, occlusion, and pixel noise to produce the inputs shown in Figure 11. For each sequence, a random movement direction is selected for the occluding square that is reflected near the image borders. Thus, most digit parts are visible at some point of the sequence and the network has the chance to remember the parts seen before occlusion. The network was trained for 16 time steps with a working set of size 200 chosen from 10000 randomly selected training set examples. For the first ten digits of the test set, the output sequence is shown in Figure 11. One can see that the network is able to produce good reconstructions of the originals (see Fig. 10) given the input sequences. The less ambiguous parts are reconstructed faster than the parts that are occluded. Towards the end of the sequence the output changes are small.

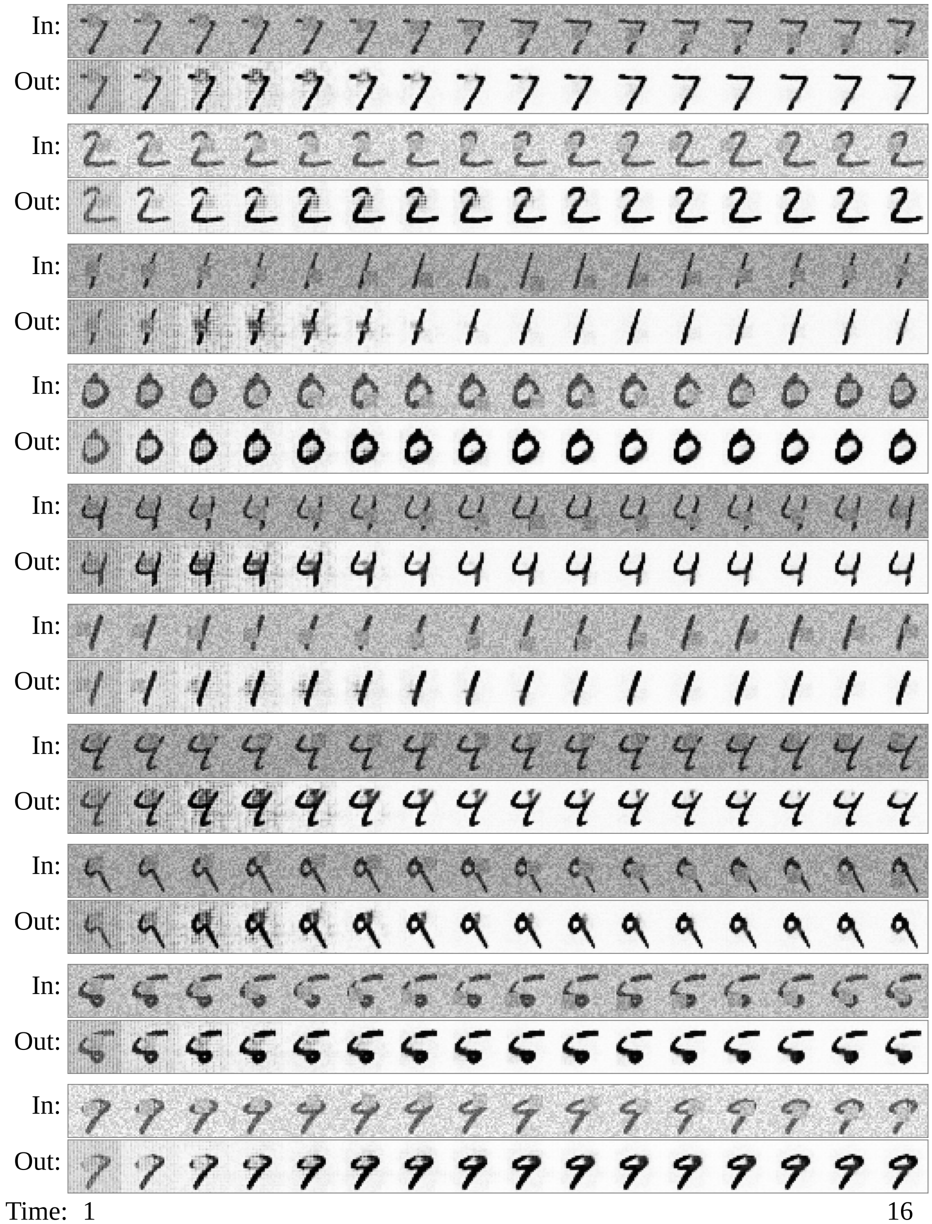

Fig. 11. Reconstruction of an image from a sequence of degraded images.

5. Discussion

The experiments demonstrated that difficult non-linear image reconstruction tasks can be learned by hierarchical neural networks with local connectivity. Supervised training of the networks was done by a combination of BPTT and RPROP.

The networks reconstruct images iteratively and are able to integrate partial results as context information for the resolution of local ambiguities. This is similar to the recently demonstrated belief propagation in graphical networks with cycles. The difference is that the proposed approach learns horizontal and vertical feedback loops that produce rich multiscale representations to model the images where current belief propagation approaches use either trees or arrays to represent the vertical or horizontal dependencies, respectively.

Further, the proposed network can be trained to compute an objective function directly, while inference in belief networks with cycles is only approximate due to multiple counting of the same evidence. Recently, Yeddia, Freeman and Weiss proposed generalized belief propagation that allows for better approximations of the inference process. [13] It would be interesting to investigate the relationship between this approach and the hierarchical recurrent neural networks.

The iterative reconstruction is not restricted to static images. We showed that the recurrent network is able to integrate information over time in order to reconstruct digits from a sequence of images that were degraded by random background level, contrast reduction, occlusion, and pixel noise. The training method allows also for a change of the desired output at each time step. Thus, the networks should be able to reconstruct video sequences.

References

1. Eero P. Simoncelli and Edward H. Adelson. Noise removal via Bayesian wavelet coring. In *Proceedings of IEEE International Conference on Image Processing (ICIP'96) – Lausanne, Switzerland*, 1996.
2. David L. Donoho and Iain M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90(432):1200–1224, 1995.
3. William T. Freeman and Egon C. Pasztor. Learning low-level vision. In *Proceedings of IEEE 7th International Conference on Computer Vision (ICCV'99) – Kerkyra, Greece*, pages 1182–1189, 1999.
4. H. Sebastian Seung. Learning continuous attractors in recurrent networks. In M.I. Jordan, M.J. Kearns, and S.A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 654–660. MIT Press, 1998.
5. Sven Behnke and Raúl Rojas. Neural Abstraction Pyramid: A hierarchical image understanding architecture. In *Proceedings of International Joint Conference on Neural Networks (IJCNN'98) – Anchorage*, volume 2, pages 820–825, 1998.
6. Sven Behnke. Hebbian learning and competition in the Neural Abstraction Pyramid. In *Proceedings of International Joint Conference on Neural Networks (IJCNN'99) – Washington, DC, paper #491*, 1999.
7. Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.
8. Ronald J. Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2(4):491–501, 1990.
9. Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.
10. Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of International Conference on Neural Networks (ICNN'93) – San Francisco*, pages 586–591. IEEE, 1993.
11. Michael D. Garris and R. Allen Wilkinson. NIST special database 3 – handwritten segmented characters. Technical Report HWSC, NIST, 1992.
12. Yann LeCun. The MNIST database of handwritten digits. http://www.research.att.com/ ∼yann/exdb/mnist, AT&T Labs, 1994.
13. Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Generalized belief propagation. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 689–695. MIT Press, 2001.