A feedback network with useful invariant properties

M. Fang, G. Häusler
Physikalisches Institut, University of Erlangen, FRG

Received July 3, 1987

Abstract
A feedback Network with useful invariant properties. A feedback network is described, that is invariant under a shift of the input pattern. The network combines a space invariant transform with binary input and output, and a special coupling matrix. The space invariant transform can be implemented within the network.

Inhalt

1. Introduction
One of the fascinating properties of neural networks is associativity. Hopfield [1] introduced a model for neural networks that restores an incomplete or disturbed binary input pattern within a few feedback cycles. Very similar networks have been already described and analyzed by Kohonen [2], Nakano [3], Amari [4, 5]. The output pattern of the Hopfield network is in general that pattern which is most similar to one of the learned input patterns. The feedback loop of this network consists of a coupling matrix which carries the information about the learned patterns and a nonlinear mapping function, as shown in fig. 1.

This network can be characterized by the following iterative algorithm:

\[ V_i(k+1) = NL \left( \sum_{j=1}^{N} C_{ij} V_j(k) \right) \]  

where \( V \) is the \( k \)-the iteration of a bipolar binary vector \((-1,1)\) of length \( N \). \( NL \) is a nonlinear mapping function, for example, the signum function. \( C_{ij} \) is the element of the coupling matrix with size \( N \times N \).

The system can learn \( Q \) input patterns \( V_q, q = 1, \ldots, Q \), by the following construction of the coupling matrix:

\[ C_{ij} = \sum_{q=1}^{Q} V_i^q V_j^q \quad \text{for} \ i \neq j \]  

\[ C_{ii} = 0 \]  

For the application of the Hopfield model the following features have to be considered:

1. The Hopfield model is shift variant. If the input pattern is disturbed by noise, the system can restore it even with a noise level of 50% [6]. However, even small shifts of a few pixels usually cause failure. This is due to the fact that recognition is essentially based on the inner product of the stored vectors and the input vector [7].

2. The initial Hopfield model can learn and recognize about 0.15 \( N \) uncorrelated binary patterns. However, generally patterns are correlated, hence costly orthogonalization procedures are necessary [8, 9]. It is possible to directly store correlated patterns, in a modified Hopfield model, as was shown by Kinzel [6]. However only about \( \ln(N) \) patterns can be stored, which is much smaller than 0.15 \( N \), for large \( N \).

3. The autoassociative restoration of disturbed signals, performed by the Hopfield network, is a useful feature. However, frequently we want to output the name of the recognized object or of its class. This “hetero association” or classification requires further comparison of the output to each of the learned patterns if it matches an exemplar exactly. Hetero association and classification are not explicitly performed by the Hopfield network.

In the following we want to demonstrate a feedback network that can overcome some of the above mentioned problems of the Hopfield network.

2. Basic idea

In order to overcome space variance it is obvious to perform a space invariant transform of the input vector \( V \) before feeding it into the network. However, some problems arise: First, we get the transformed data \( \hat{V} \) as an

![Fig. 1. Block diagram of the Hopfield network.](image-url)
Fig. 2. Block diagram of a shift invariant feedback network.

The output of this system is an inverse transformation to reproduce the original data $V$ is generally not possible, since space invariant transformations are not invertible without additional information. Second, we must find a suitable invariant transform which can be easily implemented within the network.

The first problem can be solved without an inverse transform. The principle is sketched in fig. 2. The series of the operators is: space invariant transform $T$, coupling matrix $C_{ij}$ and nonlinearity $NL$. The coupling matrix is different from Hopfield’s rule, eq. (2). It is now trained with the original patterns $V_0$ and the corresponding transform patterns $V_q$:

$$C_{ij} = \frac{1}{q} \sum_{q=1}^{Q} V_{i}^q \cdot \bar{V}_{j}^q. \quad (3)$$

The system equation can be written as

$$V(k+1) = NL \left\{ \sum_{j=1}^{N} C_{ij} \bar{V}_{j}(k) \right\}$$

or

$$V(k+1) = NL \left\{ \sum_{q=1}^{Q} \left[ \sum_{j=1}^{N} \bar{V}_{j}^q \bar{V}_{j}(k) \right] V_{i}^q \right\}. \quad (5)$$

The $[\cdot \cdot \cdot]$-term is the inner product of the transformed learned vector $V_i^q$ and the transformed input vector. For uncorrelated learned patterns this term is large only for that input vector close to a learned vector. Hence, summation over $q$ and clipping yields essentially that learned vector which is most similar to the input vector.

However, most patterns and their transform patterns are correlated. In order to avoid orthogonalization procedures we use the method of Kinzel [6], which is based on the destruction of synapses. The coupling matrix can be constructed in this case with the following rules:

$$C_{ij} = V_{i}^1 \cdot \bar{V}_{j}^1 \quad \text{for } q = 1 \quad (6a)$$

$$C_{ij} = 0, \quad \text{if } C_{ij} V_{i}^q \bar{V}_{j}^q < 0 \quad \text{for any } q > 1. \quad (6b)$$

It was shown [6] that by this method about $\ln(N)$ correlated patterns can be stored and recalled.

One remark about the convergence of the algorithm has to be added: Since our matrix $C_{ij}$ is not symmetric, one of the conditions of Hopfield for convergence is not satisfied. However this condition is not a necessary one and in our experiments we never observed divergence.

The invariance of our network depends on the invariance of the transform $T$. In general, one can achieve all desirable invariant properties, if a suitable transform can be found. For a network with a coupling matrix trained by the rule of eq. (3), the transform pattern $V_i^q$ must not be binary. However, if we train the coupling matrix by using eqs. (6a) and (6b), the transform patterns should be bipolar binary ($-1,1$).

This will now be explained: It can easily be shown that the network cannot learn two different bipolar binary patterns if the transform patterns are unipolar (in this case the coupling matrix displays many ‘zero rows’, hence corresponding vector components cannot be reconstructed). Therefore some well-known invariant transforms, such as the Fourier amplitude spectrum, the rapid transform, invariant moments, etc., are not suitable. Moreover, looking closer to the Kinzel learning rule eqs. (6a) and (6b), it turns out that only the sign of the transform pattern is used. Hence it is economic to use a

Fig. 3. (a) The signal flow diagram of the generalized rapid transform (GRT) for an input vector with four pixels. (b) A feed forward network with $\ln(N)$ layers for the implementation of the generalized rapid transform. The weights are all unqul. The nonlinear mapping function are $F_1$ and $F_2$. 

$a_j$ and $a_i$ are bipolar ($-1, 1$).
bipolar binary transform and not to waste storage capacity by unused information. A binary transform has a further advantage: it can be easily implemented within the neural network itself.

There is a space invariant transform satisfying the desired conditions: The generalized rapid transform defined by Wagh and Kanetkar [10] as a special example of an invariant transform for binary patterns. In the following section we briefly review this transform.

3. Generalized rapid transform (GRT)

The generalized rapid transform [10] (GRT) is similar to the rapid transform (RT) introduced by Reitboeck and Brody [11]. The signal flow graph of both transforms is shown in fig. 3a, left side. Wagh and Kanetkar have generalized the rapid transform by using any two commutative operators instead of the two special commutative operators “a1 + a2” and “[a1 – a2]” applied to two pixels ai and aj of the input pattern in the RT. Two special commutative operators for binary input and output may be max(ai, aj) and min(ai, aj) as shown in fig. 3a. Such a transform with two commutative operators shows the desired cyclic shift invariance [10]. This class of transforms has a computational complexity of $O(n \log n)$; n is the length of the input vector.

For our purpose, the GRT shows one more interesting property. It can be easily implemented by a feed forward network. This is illustrated in fig. 3b. The weights of the interconnections are all set to 1. The thresholds of the nonlinear operators F1 and F2 have to be set to small negative (0 – d) and positive (0 + d) values as shown in fig. 3b.

4. Networks with data compression

Before proceeding to the experiments we want to explain two additional ideas which have not necessarily to be implemented, but which improve the efficiency.

Fig. 4. Block diagram of the invariant network with an additional data compression step (DC).

Fig. 5. Some characters and their transform pattern. The binary values +1 and –1 are represented by “#” and “.” respectively. Only the parts of the transform patterns indicated with rectangles carry the useful information.

The first idea is an additional data compression step DC, after the transformation, as shown in fig. 4. We illustrate data compression with some characters. Each character is represented by an 8 by 16 binary matrix. Fig. 5 shows the characters and the corresponding transform patterns.

From fig. 5 follows that only those pixels inside of the marked rectangles carry the useful information. The other pixels are identical for all patterns and can be seen as a non-relevant background. The background can be omitted without loss of useful information.

Only if the original patterns are disturbed with noise, this background in the transform patterns is also disturbed and therefore not identical for each pattern. However the background carries in this case only noise information.

The patterns are not uncorrelated. Hence we use the modified Hopfield model with destruction of synapses. Therefore this network system can only learn and recognize about ln(N) patterns. We train the matrix with original patterns and transform patterns after data compression. The transform pattern with data compression has only 40 (4 × 10) pixels (within the marked rectangles in fig. 5). In this case the network can learn $\ln(40) \approx 3.69$ patterns or three patterns in fact. However if we select only 20 (2 × 10) coefficients of the transform pattern for learning, still $\ln(20) \approx 3$ patterns can be learned. Hence, it is always efficient to use a small size of the transform after data compression for matrix training.

At a first glance the capacity of learning only 3 correlated patterns appears low. However, the efficiency of the system can be described by the following ratio: the number of the storable pixels divided by the number of necessary matrix elements. In our case, this ratio comes to $(3 \times 128)/(20 \times 128) = 15\%$: the efficiency of storage capacity in our model is the same as in the Hopfield model. However the Hopfield model can only store uncorrelated patterns while we store both correlated and uncorrelated data.

5. Heteroassociativity

With a simple heteroassociativity the network can be extended.

The character set of the learned character can be the same, it may, for example, be the set of the characters used in the character set of fig. 5. The learning feed back network itself can be extended as shown in fig. 6.

We present here only the first idea of using the desired feedback loop for data compression.
patterns with this 15% efficiency, whereas our model can store both correlated and uncorrelated patterns in this case.

5. Hetero associativity

With a modification of the matrix of eq. (3) hetero associativity can be implemented.

The output of the network described above is that learned pattern which is most similar to the input pattern. It is important that the output pattern is located at the learned position. This makes a further hetero association simple. We only have to apply an additional feed forward step, where \( \mathbf{P} \) denotes the desired heteroassociative output. The algorithm is as follows

\[
\mathbf{P}_i = NL\left\{ \sum_{j=1}^{L} A_{ij} \mathbf{V}_j \right\}
\]

for uncorrelated patterns.

For correlated patterns we again use destruction of synapses:

\[
A_{ij} = V_{ij} \quad \text{for } q = 1
\]

\[
A_{ij} = 0, \quad \text{if } A_{ij} V_{ij} < 0 \quad \text{for any } q > 1.
\]

The size \( L \) of the desired output vector must not be the same as the size \( N \) of the original pattern. The output may, for example, be the name of the object, the number of the class where it belongs to, or the ASCII code of our characters. The block diagram of the whole model including feed forward step for hetero association is displayed in fig. 6.

We point out that one has to perform feedback only in the first part of the system for associative restoration with the desired invariance. After the first step we have undisturbed patterns at the learned position. We need only one feed forward step for hetero association. Each learned pattern is a stable fixed point and the system finds the correct output without feedback.

6. Numerical experiments

In this section we investigate our network numerically. As an input pattern we choose the characters A, B and C. The original input patterns and their corresponding transform patterns are shown in fig. 5. The binary patterns with two values of “+1” and “−1” are represented in fig. 7 with bright and dark areas, respectively.

For matrix training we selected 20 coefficients from the first two columns of the rectangles shown in fig. 5. We perturbed these three patterns in the following way:

- At first patterns were shifted upwards by two pixels.
- Then a “cross” pattern was subtracted from the shifted patterns, as an example of a deterministic perturbation.
- Then random noise was added.

<table>
<thead>
<tr>
<th>Input</th>
<th>Iteration 1</th>
<th>Iteration 2</th>
<th>Iteration 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
</tbody>
</table>

Fig. 6. Block diagram of the whole invariant network including a hetero association/classification step. The network displays a desired output, for each input or for a class of inputs.

Fig. 7. Shift invariant associative restoration of disturbed characters A, B, C.
Fig. 8. The computer simulation shows the sensitivity against shift of the Hopfield model with destruction of synapses. Three patterns A, B and C are learned. The input pattern A, after training, is shifted upwards by only one pixel (a) or two pixels (b). The network cannot recall the original pattern A, even without any noise present.

Our model can restore the perturbed patterns within a few iterations as shown in fig. 7. We also tested the model with a feed-forward step for hetero association as shown in fig. 6. We trained the matrix $A_{ij}$ using eq. (9) with the original patterns and the corresponding ASCII code with 8 pixels for each character. The output of the whole system was then the corresponding ASCII code, as expected.

For comparison, it is shown in fig. 8 that the Hopfield model can be very sensitive to a small shift. Three patterns A, B and C are learned. The learned pattern A, after training, is shifted upwards by only one, respectively two pixels. The system cannot recall the original pattern A.

7. Conclusion

The proposed feedback network shows space invariance and both auto associativity and hetero associativity. The space invariant transform used can easily be implemented within a multi-step feed-forward network. The model makes data compression possible and reduces the number of necessary matrix elements.

References