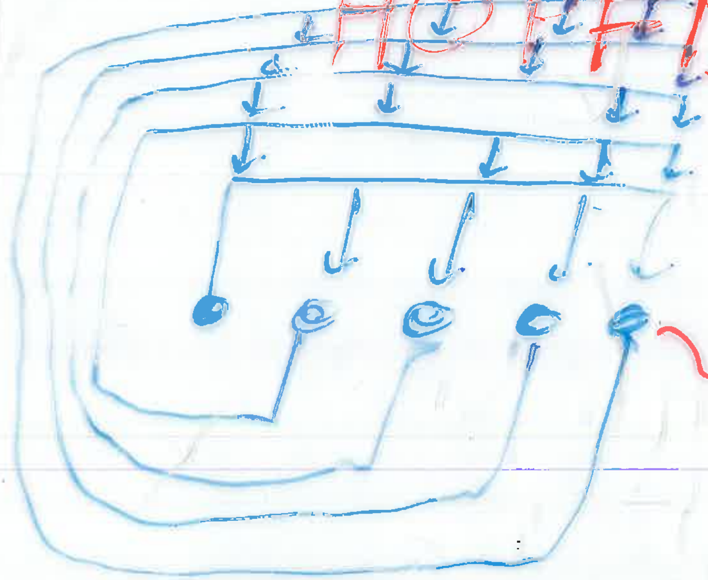
HOPFIELD NET



HOPFIELD NET
(recurrent)

perceptrons (thresholds)

- Converges to Stable State
(Proof via Lyapunov (Energy) Function)

- Stores "memories" in weights
(Associative – Fault-Tolerant)

- $T_W = \vec{X}_1 \vec{X}_1^T + \vec{X}_2 \vec{X}_2^T \dots + \vec{X}_n \vec{X}_n^T$

$w_{ii} = 0$ outer-product

Bipolar Example: Store $(1,-1,-1)$ and $(-1,1,1)$

$$T_W = \begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix} (1\ -1\ -1) + \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix} (-1,1,1) = \begin{pmatrix} 2 & -2 & -2 \\ -2 & 2 & 2 \\ -2 & 2 & 2 \end{pmatrix}$$

$$\boxed{T_W = \begin{pmatrix} 0 & -2 & -2 \\ -2 & 0 & 2 \\ -2 & 2 & 0 \end{pmatrix}}$$

$$T_W \begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 4 \\ -4 \\ -4 \end{pmatrix} \longrightarrow \begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix}$$

- "Spurious" States (e.g. $\begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix}^c = \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}$)

Proof of Convergence of Hopfield Networks

(1) Activation function

$$f(x_j) = \begin{cases} -1 & \sum_i w_{ij} x_i - \theta < 0 \\ x_j & \sum w_{ij} x_i - \theta = 0 \\ 1 & \sum w_{ij} x_i - \theta > 0 \end{cases}$$

Define "Energy" or Lyapunov function as

$$E(\vec{x}) = -\sum_{i\neq j} x_i w_{ij} x_j \left(+ \sum_i \theta_i x_i \right) \text{ (for thresholds.)}$$

Assume: $w_{ii} = 0$, $w_{ij} = w_{ji}$. asynchronous update.

To Show: Regardless of $\vec{x}^{(0)}$, eventually $\vec{x}^{(new)}_i = \vec{x}^{old}$.

Proof: (1) $E(\vec{x})$ is bounded from below

Pf: $E(\vec{x}) < \sum_{i\neq j} |w_{ij}|$

(2) Each change is finite. If $x_i^{new} \neq x_i^{old}$

Pf: $E^{new} - E^{old}$

$$= -\sum_{i\neq j} x_i^{new} w_{ij} x_j^{(new)} - \sum_{i,j} x_i^{old} w_{ij} x_j^{old}$$

$$= -\sum_j x_i^{new} w_{ij} x_j - \sum_j x_i^{old} w_{ij} x_j$$

$$= -\sum_j (x_i^{new} - x_i^{old}) w_{ij} x_j \quad \text{(Take smallest such over } j\text{)}$$

(3) Always decreases:

$$E^{new} - E^{old} = -(x_i^{new} - x_i^{old}) \left(\sum_j w_{ij} x_j - \theta_i \right)$$

If $x_i^{new} \neq x_i^{old}$ then:

Case 1: $x_i^{new} = 1$, $x_i^{old} = -1$ So $x_i^{new} - x_i^{old} > 0$

$\sum_j w_{ij} x_j > 0$ (θ) so both +

$\therefore \Delta E < 0$

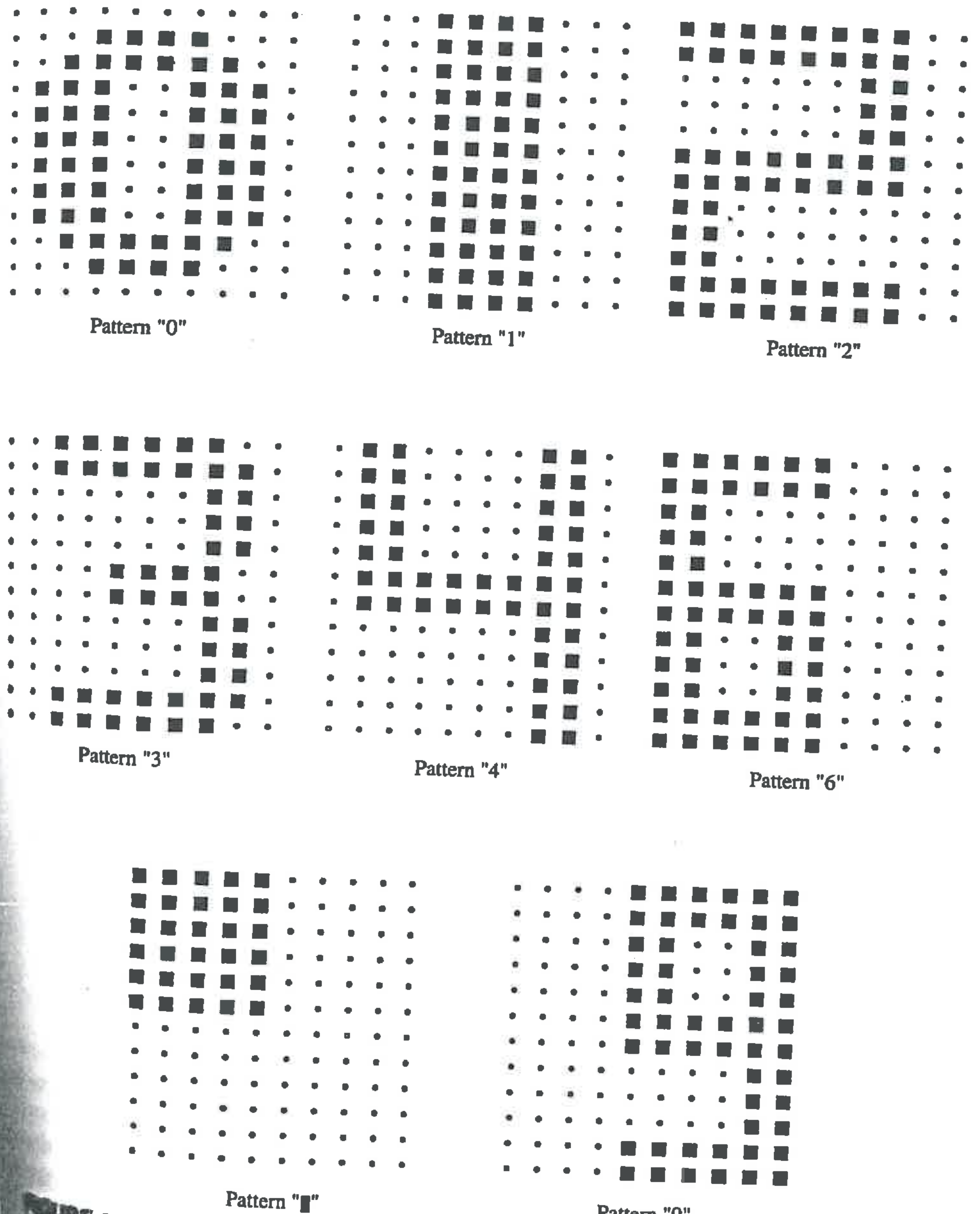
Case 2: $x_i^{new} = -1$, $x_i^{old} = 1$ $x_i^{new} - x_i^{old} < 0$

$\therefore \sum_j w_{ij} x_j < 0$ (θ) so both −

$\therefore \Delta E < 0.$

…orage (learning) phase of the Hopfield network to create the synaptic …which was done using Eq. (8.9). The retrieval phase of the network's …formed asynchronously, as described in Section 8.3.

…rst stage of the retrieval part of the experiment, the fundamental memories …ented to the network to test its ability to recover them correctly from the …ation stored in the synaptic weight matrix. In each case, the desired pattern was …duced by the network after one iteration.



…URE 8.7 Set of handcrafted patterns for computer experiment on the Hopfield network.

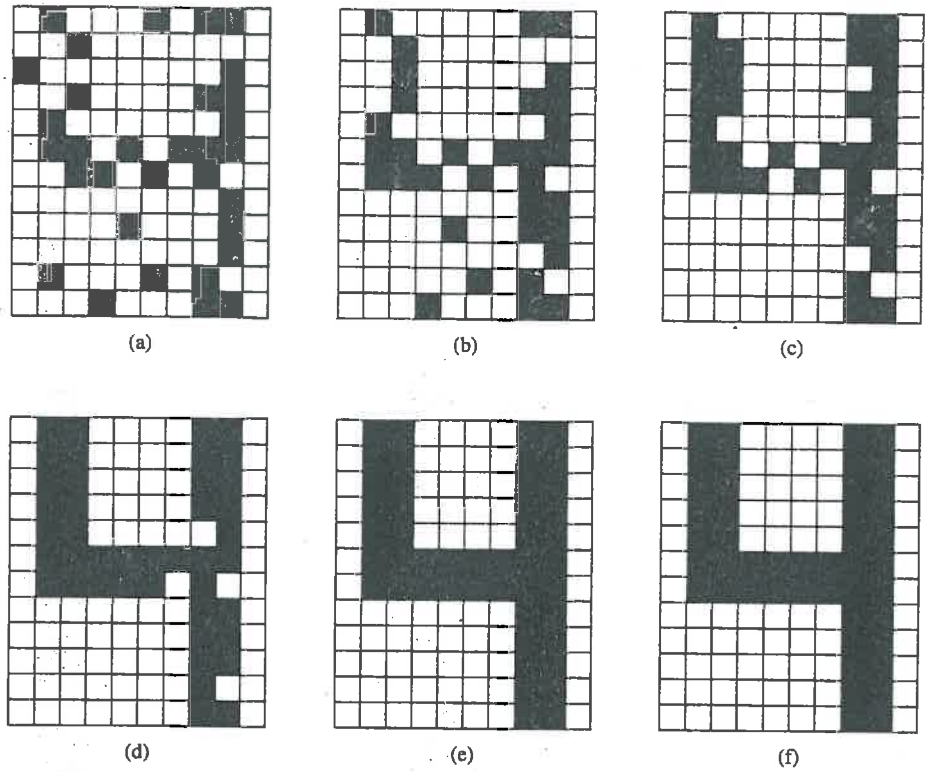


Figure 5.2 Example of recursive asynchronous update of corrupted digit 4: (a) $k = 0$, (b) $k = 1$, (c) $k = 2$, (d) $k = 3$, (e) $k = 4$, and (f) $k = 5$.

Example snapshots of neural network transitions were given in Figure 1.8. Another example of convergence of a 10×12 neuron network is illustrated in Figure 5.2. It shows the 10×12 bit map of black and white pixels representing the digit 4 during the update as defined in (5.3). Figure 5.2(a) has been made for $k = 0$, and it shows the initial, distorted digit 4 with 20% of the pixels randomly reversed. Consecutive responses are shown in Figure 5.2(b) through (f). It can be seen that the updates continue until $k = 4$ as in Figure 5.2(e), and for $k \geq 5$ no changes are produced at the network output since the system arrived into one of its stable states (5.8b).

It has been shown in the literature that the synchronous state updating algorithm can lead to persisting cyclic states that consist of two complementary patterns (Kamp and Hasler 1990). Consider the 2×2 weight matrix with zero diagonal and off-diagonal entries of -1, and the synchronous updates of the output vector $\mathbf{v}^0 = \begin{bmatrix} -1 & -1 \end{bmatrix}^t$. By processing the signal $\mathbf{v}^0$ once we obtain

$$\mathbf{v}^1 = \Gamma \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} \operatorname{sgn}(1) \\ \operatorname{sgn}(1) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Next, to demonstrate the error-correcting capability of the Hopfield networ
of interest was distorted by randomly and independently reversing each pixel of
from $+1$ to -1 and vice versa with a probability of 0.25, and then using th
pattern as a probe for the network. The result of this experiment for digit 6 i
in Fig. 8.8. The pattern in the top left hand-corner of this figure represents
version of digit 6, which is applied to the network at zero time. The pattern
by the network after 5, 10, 15, 20, 25, 30, and 37 iterations are presented in
the figure. As the number of iterations is increased, we see that the resembl

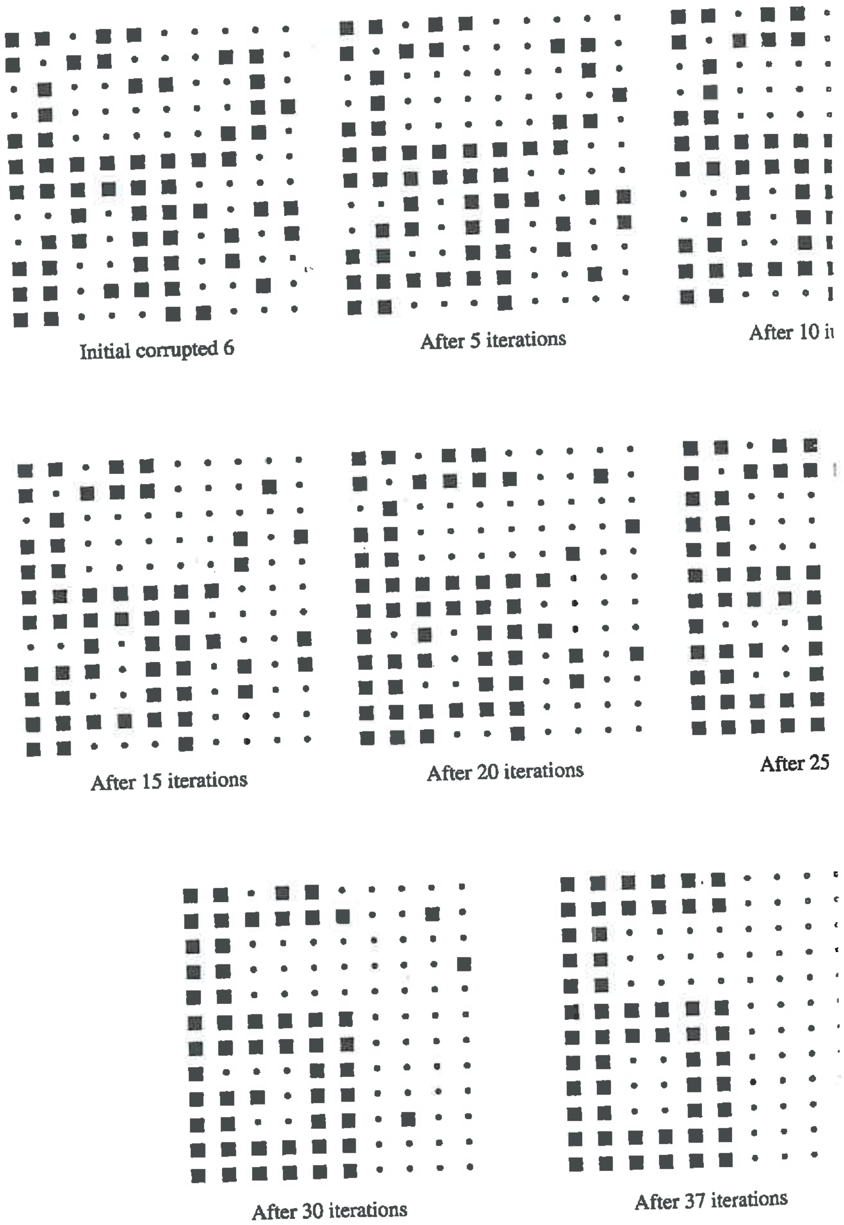
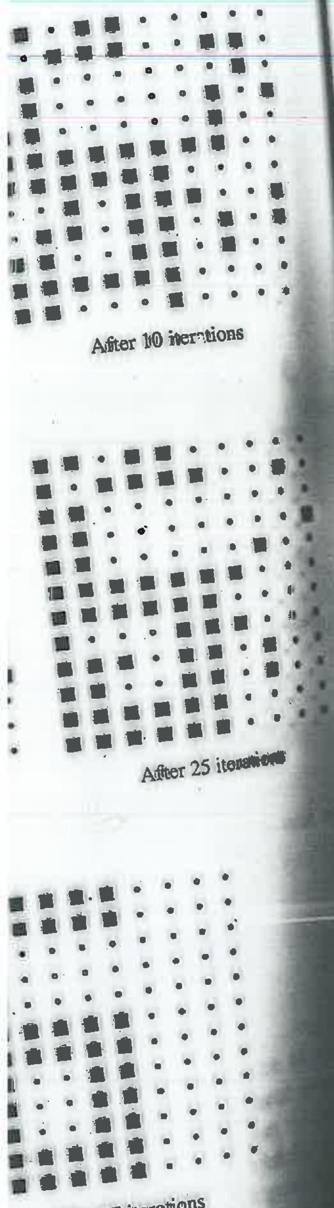


FIGURE 8.8 Correct recollection of corrupted pattern 6.

ld network, a pattern
ch pixel of the pattern
n using the corrupted
or digit 6 is presented
represents a corrupted
The patterns produced
presented in the rest of
the resemblance of the



of corrupted pattern 6.

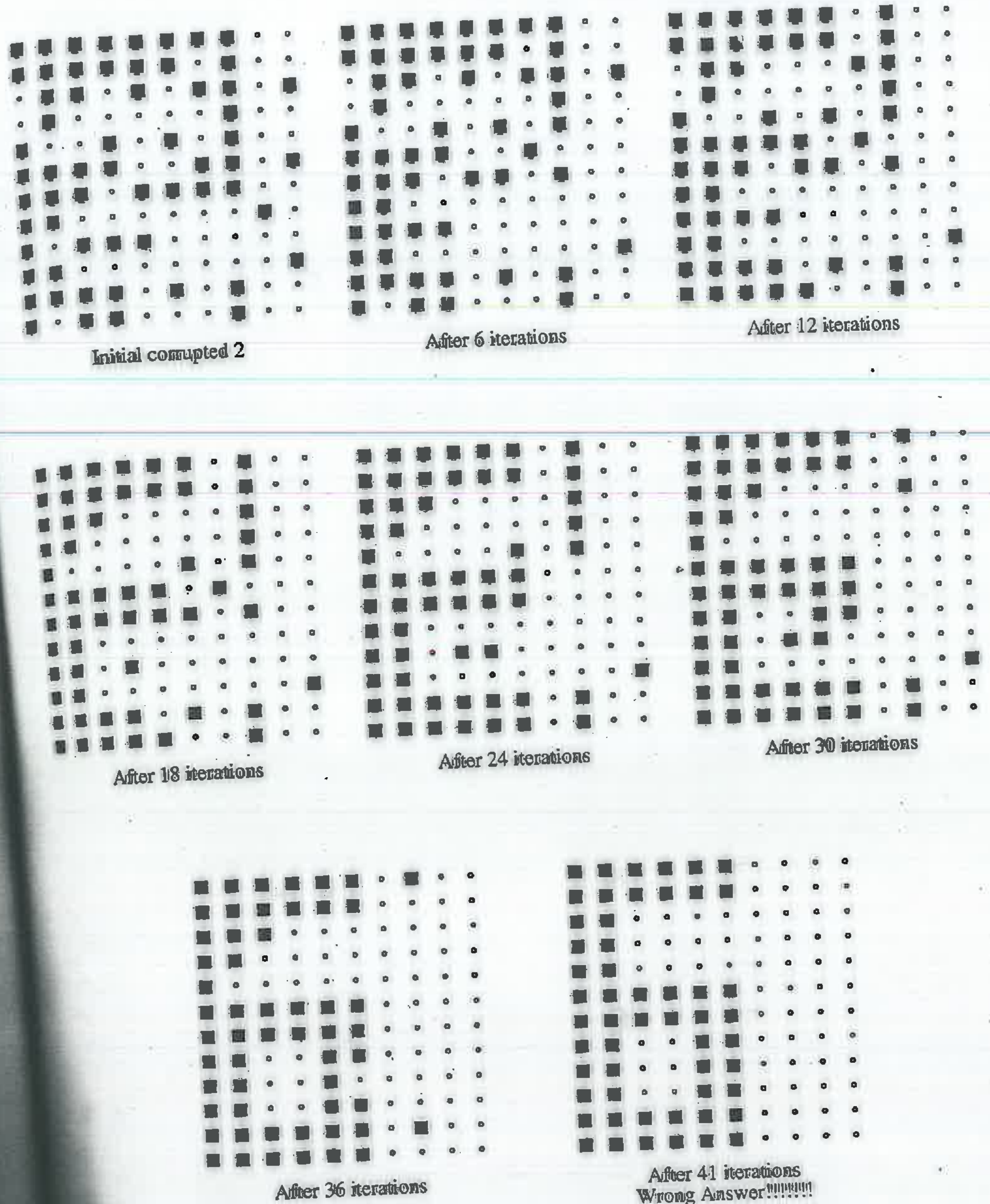


FIGURE 8.9 Incorrect recollection of corrupted pattern 2.

network output to digit 6 is progressively improved. Indeed, after 37 iterations, the network converges onto the correct form of digit 6.

Since, in theory, one-quarter of the 120 neurons of the Hopfield network end up changing state for each corrupted pattern, the number of iterations needed for recall, on average, is 30. In our experiment, the number of iterations needed for the recall of the different patterns from their corrupted versions were as follows: