In this article we present the so-called continuous classifying associative memory, able to store continuous patterns avoiding the problems of spurious states and data dependency. This is a memory model based on our previously developed classifying associative memory, which enables continuous patterns to be stored and recovered. We will also show that the behavior of this continuous classifying associative memory may be adjusted to some predetermined goals by selecting some internal operating functions. © 2002 Wiley Periodicals, Inc.

1. INTRODUCTION

Associative memories generally have a low storage capacity as a result of the appearance of spurious patterns (patterns that do not correspond to any of those stored) and data dependency (the imposition of conditions that can be stored).

The classifying associative memory (CLAM) solves the problem of capacity by avoiding those factors that restrict it: spurious states and data dependency. It does so by not limiting the type of data that can be stored, while preventing interference between the stored patterns that could result in spurious patterns.

However, the CLAM is restricted to storing bipolar patterns. In many cases we might need patterns, the components of which are continuous. These patterns can be memorized in a CLAM if they have previously been coded in a bipolar representation using the incremental discretization coding (CDI) method.

Although the bipolar representation of the patterns enables us to store and recover patterns correctly, the input patterns are classified using the Hamming distance. This means that the distances used in the classification are not the natural distances between the continuous patterns, but rather those which exist between their bipolar representations.

In this article, we present the continuous classifying associative memory. This is a CLAM-based model that allows patterns with continuous components to be stored, and then recovered and classified, by the nearest neighbor according to the measurement required.

*Author to whom all correspondence should be addressed: e-mail: bailon@decsai.ugr.es.
2. CLASSIFYING ASSOCIATIVE MEMORY

The CLAM\textsuperscript{1–3} is an associative memory with two layers of processing elements (PEs) that classify bipolar patterns by the criterion of the nearest neighbor according to the Hamming distance (see Figure 1).

The topology of the memory is dynamic because the number of output PEs and the number of connections vary according to the number of patterns memorized.

The input/output layer comprises the same number of bipolar PEs as components as the patterns to be memorized. The input pattern is presented in this layer, and after the recovery process, the closest stored pattern is obtained.

The output layer is competitive and made up of bipolar PEs that represent the classes by which the patterns will be classified. For each pattern memorized, an associated PE will be created in the output layer. A CLAM that has not yet memorized any pattern will not have any PEs in this layer.

The two layers are linked by connections, the weights of which are represented in a matrix with the same number of rows as the number of patterns stored, and with the same number of columns as the number of pattern components.

This operates in the following way:

- A possibly incomplete and/or incorrect pattern arrives at the input/output layer.
- The activation signal spreads from the input/output layer by means of the weights matrix.
- The PEs of the output layer compete so that only the one that received the maximum activation signal remains active. This PE represents the stored pattern closest to the input pattern. If after competition there is more than one active PE, this means that these received the maximum signal. In this case, it is necessary to select one of them by following some criterion and to deactivate the remaining ones.
- The signal spreads towards the input/output layer by means of the weights matrix.
- In the input/output layer, the stored pattern that is closest to the input pattern is recovered.
The main features of the model are:

- The dynamic topology means it adapts to the number of patterns to be stored.
- All the stored patterns are recovered perfectly.
- It classifies incomplete or incorrect patterns by the nearest stored pattern according to the Hamming distance.
- It does not present spurious states; that is, it will never recover a pattern which has not been stored before.
- It does not impose any type of condition on the patterns to be stored so as to guarantee that they are memorized correctly.

3. CONTINUOUS CLAM

The continuous classifying associative memory (CCLAM) is an associative memory model that stores arbitrary spatial patterns. The topology of the CCLAM (see Figure 2) is different from the CLAM in that all the PEs are continuous instead of bipolar and binary. The activation signal received by the PEs does not need to exceed the threshold for the element to be activated, and its activation function is:

\[
    f(x) = \begin{cases} 
        0 & \text{if } x < 0 \\
        x & \text{if } 0 \leq x \leq 1 \\
        1 & \text{if } x > 1 
    \end{cases}
\]

(1)

There are as many PEs in the $F_A$ layer as there are components of the patterns to store. The input patterns will be presented and the stored patterns will be recovered in this layer.

The PEs of the $F_B$ layer represent the stored patterns. The number of these is variable and changes dynamically with the learning, in such a way that, as in the discrete CLAM, there will always be as many PEs in the $F_B$ layer as there are

![Figure 2. Continuous classifying associative memory (CCLAM).](image-url)
stored patterns. The activation state of these indicates the degree to which the pattern associated with each of them is recognized.

When the CCLAM operates, we can distinguish two processes: classification and recovery. The classification process receives an input pattern in the $F_A$ layer, and in each element of $F_B$ it obtains a value that indicates the degree to which the input pattern resembles the stored pattern presented in $F_B$. The recovery process receives the degrees to which we refer to each of the stored patterns in $F_B$, and it recovers a combination of these patterns in $F_A$.

The $F_B$ layer is competitive. The weights of the competition signals between the PEs of the $F_B$ layer are represented in a $\beta_{n \times n}$ matrix, with $n$ being the number of patterns stored, and $\beta_{ij}$ being the weight of the connection that exists between $b_i$ and $b_j$. The connection weight indicates the degree of competition, so that a connection from the element $i$ to $j$ with zero weight indicates that the element $i$ does not compete with the element $j$; if the weight is one, the element $i$ competes with the element $j$ with a maximum degree indicated by its activation state. If all the competition connections have a unitary weight, we obtain a competition in which the same importance is given to all the patterns; if all the weights are zero, we cancel the competition.

For the competition, we must establish whether the winning PE will retain its activation state or take the value 1 as its new state. Let $Y_i$ be the activation state of the elements of $B_i$. If after competition, the winning element retains its state, the activation value of $B_i$ will be:

$$
\begin{cases}
Y_i & \text{if } Y_i \geq Y_j \beta_{ji} \quad \forall j \\
0 & \text{otherwise}
\end{cases} \quad (2)
$$

If the winner has the value 1, the new value of $B_i$ will be:

$$
\begin{cases}
1 & \text{if } Y_i \geq Y_j \beta_{ji} \quad \forall j \\
0 & \text{otherwise}
\end{cases} \quad (3)
$$

If there is more than one winning element, one of them will be selected following some pre-established selection criterion.

As the competition established in $F_B$ may be different in the classification and recovery phases, we can really consider two weight matrices, where $\beta_{ij}^C$ is the competition weight of $i$ towards $j$ in the classification process, and $\beta_{ij}^R$ is the competition weight of $i$ towards $j$ in the recovery process.

During the classification process, we present an input pattern in $F_A$, and in the $F_B$ layer we either obtain the degree of compatibility that exists between each of the stored patterns and the input pattern, or we find that the only active element is the one representing the pattern that is most compatible with the input pattern. This will depend on the type of competition that is established.

In the recovery process, we present the degrees to which we refer to each of the stored patterns in $F_B$, and we recover a combination of these patterns in $F_A$. 
3.1. Learning

During the learning process, memorization of the pattern involves creating a new PE in $F_B$, and the appropriate adjustment of the weights of the new connections that are made between the elements in $F_A$ and the new PE.

The learning process consists of the following steps:

1. Arrival at $F_A$ of a pattern to be memorized.
2. Verification that the pattern has not been previously memorized.
3. Creation of a new PE in $F_B$ associated with the pattern.
4. Adjustment of the competition weights of the $F_B$ layer.
5. Adjustment of the weights of $F_A$ towards the new PE.

3.1.1. Verification That the Pattern Has Not Been Previously Memorized

In order to check that the pattern has not yet been stored in the memory, the classification process is carried out with this pattern as the input of the memory, and the result obtained is checked.

The classification process enables us to specify two functions, $\Upsilon^C$ and $\Psi^C$, so that the memory behavior can be adapted to our needs. In a further section we will discuss that choice. To check prior pattern memorization, the following functions will be used:

$$\Upsilon^C(x, y) = 1 - |x - y|$$
$$\Psi^C = \text{arithmetic mean}$$

With this configuration, the input pattern is compared with each of those stored, component by component. The signal received by the PEs of the $F_B$ layer can only have a value of one if the input pattern and the stored pattern coincide.

In the $F_B$ layer, we ensure that all the weights are equal to one and that the winning element retains its activation state during the competition phase. In this way, if any element with a value equal to one remains after the classification process, this implies that the input pattern coincides with one of those stored, and therefore the learning of this should not be repeated.

3.1.2. Adjustment of the Competition Weights of the $F_B$ Layer

In the $F_B$ layer, the weight of the $(i,j)$ connection indicates the degree to which the activation signal of the PE associated with the $i$th pattern is compared with that of the $j$th. If the weight of the connection is one, the two activation states of the connected PEs are compared; if the weight is zero, this means that there is no competition between the connected PEs; other intermediate values indicate different degrees of influence.

As the $F_B$ layer is dynamic, for each stored pattern, it is necessary to assign the weights of the connections that go to the new PE that represents the pattern.

The weight of the competition connection between two PEs can be seen as the relative importance that is given to the two patterns represented by the connected
PEs. We will generally find that all the competition connections have the weight \( \beta_{ij} = 1 \), to reflect that the same importance is given to all the patterns.

The weights established for the competition in the classification process can be different from those of the recovery process. \( \beta_{ij}^C \) will be the weight of the competition that exists between the PEs \( i \) and \( j \) during the classification, and \( \beta_{ij}^R \) the weight during the recovery.

### 3.1.3. Adjustment of the Weights of \( F_A \) Towards the New PE

The number of PEs in the \( F_B \) layer is dynamic, so that when a new pattern is stored, a new PE is created in the \( F_B \) layer associated with this pattern.

When the pattern \( X = (x_1, \ldots, x_p) \) is stored, the value \( m_{pi} = x_i \) is assigned to the weights of the connections of the PEs in the \( F_A \) layer towards the new \( b_p \) PEs in the \( F_B \) layer.

A CCLAM that has stored \( p \) patterns of \( n \) components will have an associated weights matrix that represents the weights of the connections between the PEs in the \( F_A \) and \( F_B \) layers.

\[
M_{n \times p} = \begin{pmatrix}
m_{11} & \cdots & m_{1p} \\
\vdots & \ddots & \vdots \\
m_{n1} & \cdots & m_{np}
\end{pmatrix}
\]  

(5)

Therefore, the storage of the set of patterns \( X_1, \ldots, X_n \) with \( X_i = (x_{i1}, \ldots, x_{ip}) \) will result in the weights matrix:

\[
M = \begin{pmatrix}
x_{11} & \cdots & x_{1p} \\
\vdots & \ddots & \vdots \\
x_{n1} & \cdots & x_{np}
\end{pmatrix}
\]  

(6)

For each new stored pattern, a row is added to the weights matrix with the weights of the connections of the PEs in the \( F_A \) layer towards the new PE in \( F_B \) that represents the new pattern.

The weights of the connections between the two layers in the classification process may be different from those of the recovery process. In order to differentiate between them, we will call the weights matrix in the classification process \( M^C \), and the weights matrix in the recovery process \( M^R \).

### 3.2. Classification

In the classification process, the PEs send activation signals from the \( F_A \) layer towards the \( F_B \) layer. When an input pattern is presented in the \( F_A \) layer, activation states are obtained in the \( F_B \) layer that represent the similarity that exists between the pattern and each of the stored patterns.

The classification process has the following steps:

1. A pattern \( X = (x_1, \ldots, x_p) \) arrives at the \( F_A \) layer.
(2) The signal propagates towards the $F_B$ layer so that the $b_i$ PE receives the activation signal:

$$y_i = \Psi^C \left\{ \Upsilon^C (x_i, m^C_{ji}) \right\}_{j=1,...,p}$$

$$= \Psi^C \left( \Upsilon^C (x_1, m^C_{11}), \Upsilon^C (x_2, m^C_{12}), \ldots, \Upsilon^C (x_p, m^C_{ip}) \right)$$  \(7\)

where $\Upsilon^C$ and $\Psi^C$ are functions bounded to the interval $[0, 1]$. The choice of these functions depends on the behavior that we would like the memory to present according to our needs (see Figure 3).

(3) The PEs in the $F_B$ layer compete with the intensity reflected in the weights matrix $\beta^C$.

(4) We obtain the result of the classification process in the $F_B$ layer. The activation state obtained in the PEs in the $F_B$ layer indicates the degree to which the input pattern is classified as belonging to each of the classes represented by the stored patterns.

### 3.3. Recovery

In the recovery process, the information flows in the opposite direction to the classification process. In the $F_B$ layer, an input pattern is presented that represents the degree of recovery of each stored pattern. Recovery has the following steps:

(1) A pattern $Y = (y_1, \ldots, y_n)$ arrives at the $F_B$ layer.

(2) The PEs in the $F_B$ layer compete with each other with the $\beta^R$ weights matrix. The new activation states form the pattern $Y' = (y'_1, \ldots, y'_n)$.

(3) The signal propagates from the $F_B$ layer towards the $F_A$ layer so that the $a_j$ PE receives the activation signal:

$$x_j = \Psi^R \left\{ \Upsilon^R (y'_j, m^R_{ji}) \right\}_{j=1,...,n}$$

$$= \Psi^R \left( \Upsilon^R (y'_1, m^R_{11}), \Upsilon^R (y'_2, m^R_{12}), \ldots, \Upsilon^R (y'_n, m^R_{1n}) \right)$$  \(8\)

where $\Upsilon^R$ and $\Psi^R$ are functions bounded to the interval $[0, 1]$. The choice of these functions depends on the behavior that we would like the memory to present according to our needs.
functions depends on the behavior that we would like the memory to present (see Figure 4).

(4) In the $F_A$ layer, we obtain the pattern resulting from the recovery process. This may be one of the stored patterns or a combination of them.

4. CONFIGURATION

The CCLAM enables us to choose the $\psi^C$, $\gamma^C$, $\psi^R$, and $\gamma^R$ functions, and the competition with which the classification and recovery processes will operate, so that their behavior meets our needs.

In this section (and to demonstrate the previous statement), we are going to examine how, by using the appropriate functions, we can:

- Classify the stored patterns by the nearest neighbor
- Recover every stored pattern

4.1. Classification by the Nearest Neighbor

As a measure of the distance that exists between the patterns, we will use the generalized distance $L_m$ (Minkowski distance), defined as:

$$L_m(X, Y) = \left( \sum_{i=1}^{p} |x_i - y_i|^m \right)^{1/m}$$

The greater the value of $m$, the greater the importance given to the particular difference that exists between the components. The Manhattan distance is calculated
with \( m = 1 \), the Euclidean distance with \( m = 2 \), and the Chebychev or maximum distance is calculated as the limit when \( m \to \infty \).

The selected functions for the classification are:

\[
\beta^C = \begin{pmatrix}
1 & 1 & \ldots & 1 \\
1 & 1 & \ldots & 1 \\
\vdots & \vdots & \ddots & \vdots \\
1 & 1 & \ldots & 1
\end{pmatrix}
\]

(10)

\[
\psi^C(x_1, \ldots, x_p) = 1 - \left( \frac{1}{p} \sum_{i=1}^{p} x_i^m \right)^{1/m}
\]

\[
\gamma^C(x, y) = |x - y|
\]

The only active element that the classification process will obtain in \( F_B \) is the one representing the stored pattern closest to the one present in \( F_A \). The signal received by the \( F_B \) elements indicates the proximity of the input pattern and the stored pattern.

Let us suppose that a CCLAM has stored the set of patterns:

\[
\{A_1, A_2, \ldots, A_n\} \quad A_i = (a_{i1}, a_{i2}, \ldots, a_{ip}) \quad i = 1, \ldots, n
\]

(11)

Let us also suppose that the \( b_i \) element in the \( F_B \) layer is associated with each stored \( A_i \) pattern.

Although there will be competition in the classification phase in which all the elements compete with the same weight, there will be no competition in the recovery phase.

The weights matrixes \( M^C \) and \( M^R \) of the memory after storing the patterns will be:

\[
M^C = M^R = \begin{pmatrix}
a_{11} & a_{12} & \ldots & a_{1p} \\
a_{21} & a_{22} & \ldots & a_{2p} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & \ldots & a_{np}
\end{pmatrix}
\]

(12)

On presenting the pattern \((x_1, \ldots, x_p)\) in the \( F_A \) layer, the signal received by the \( b_k \) PE in the \( F_B \) layer is:

\[
y_k = \psi^C \{ \gamma^C \{a_{ei}, m^C_k\}_i\}_i = 1, \ldots, p
\]

(13)

Bearing in mind the form of \( \psi^C \) and \( \gamma^C \) and the distance definition \( L_m \), we can express the signal received by \( b_k \) as:

\[
y_k = 1 - \left( \frac{1}{p} \sum_{i=1}^{p} |x_i - a_{ki}|^m \right)^{1/m} = 1 - \left( \frac{1}{p} \right)^{1/m} \cdot L_m(x, A_k)
\]

(14)
In the competition phase established in the $F_B$ layer, the PE that received the maximum signal from the $F_A$ layer is selected. In order to be able to express this selection, we define the following functions:

$$\max_k \{ m_k \} = r \iff \forall j \ m_r \geq m_j$$

$$\min_k \{ m_k \} = r \iff \forall j \ m_r \leq m_j$$

Let $c > 0$ be a constant. The previous functions have the following properties:

$$\max_k \{ m_k + c \} = \max_k \{ m_k \}$$
$$\max_k \{ m_k - c \} = \max_k \{ m_k \}$$

$$\max_k \{ cm_k \} = \max_k \{ m_k \}$$
$$\max_k \{ -m_k \} = \min_k \{ m_k \}$$

These properties allow selecting the PE that received the greatest signal from $F_A$. The only PE in the $F_B$ layer that remains active after the $b_s$ competition is the one that received the greatest signal from $F_A$; that is:

$$s = \max_k \{ y_k \}$$

Therefore:

$$s = \max_k \left\{ 1 - \left( \frac{1}{p} \right)^{1/m} L_m(X, A_k) \right\}$$

Applying the previously mentioned properties of the function, we obtain:

$$s = \min_k \{ L_m(X, A_k) \}$$

This means that the winning element in the competition in the $F_B$ layer is the one that represents the stored pattern that minimizes the distance to the input pattern.

The functions and competition selected for the recovery process are:

$$\beta^R = \begin{pmatrix} 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 \end{pmatrix}$$

$$\gamma^R(a, b) = \min(a, b)$$

$$\Psi^R(y_1, \ldots, y_n) = \max(y_1, \ldots, y_n)$$

This configuration allows the recovery of the stored pattern represented by the $F_B$ element that receives the greatest signal.
As there is no competition in the recovery phase, the signal obtained in $F_B$ propagates towards $F_A$ so that the $a_k$ element receives the signal:

$$x_k = \max \{ \min (y_i, a_{ik}) \}_{i=1,\ldots,n} = a_s$$  \hspace{1cm} (21)

The pattern recovered in the $F_A$ layer is therefore $A_s$, which is the one closest to the input pattern according to the distance $L_m$.

### 4.2. Recovery of All the Stored Patterns

In the previous section, we saw that the selected $\Psi^C$, $\Upsilon^C$, $\Psi^R$, and $\Upsilon^R$ functions allowed an input pattern to be classified as belonging to the class represented by the stored pattern that is closest according to the distance $L_m$. The recovered pattern $A_s$ therefore verifies:

$$L_m(X, A_s) \leq L_m(X, A_i) \quad \forall i \neq s$$  \hspace{1cm} (22)

A corollary can be drawn from this result: as one of the stored patterns is presented at the input, the pattern recovered is the same. It is therefore only necessary to bear in mind two properties in general, and, of course, the distance $L_m$ in particular:

- $L_m(X, Y) \geq 0$
- $L_m(X, Y) = 0 \iff X = Y$

The same pattern is prevented from being memorized more than once in the learning process. So when the $A_e$ pattern is presented as the input in the $F_A$ layer, the pattern recovered will be $A_e$, because the following is satisfied:

$$L_m(A_e, A_e) = 0 < L_m(A_e, A_i) \quad \forall i \neq e$$  \hspace{1cm} (23)

With a correct choice of the $\Psi^C$, $\Upsilon^C$, $\Psi^R$, and $\Upsilon^R$ functions, all the patterns stored in the CCLAM can therefore be recovered.

### 5. EXPERIMENTATION

As we have mentioned above, the careful selection of the $\Psi^C$, $\Upsilon^C$, $\Psi^R$, and $\Upsilon^R$ functions, together with the appropriate establishment of the parameters that determine the competition, enable the CCLAM to correctly store and recover the stored patterns.

On many occasions the classification or recovery processes will begin with a stimulus that codes information that was not stored previously. In these cases, the correct configuration of the memory will enable the closest stored pattern to be recovered.

To measure the closeness between two patterns, we configure the CCLAM to compute the Euclidean distance, and to recover the closest pattern with the following selection of the $\Psi^C$, $\Upsilon^C$, $\Psi^R$, and $\Upsilon^R$ functions and the competition with which the
classification and recovery processes will operate:

\[
\beta^C = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix}
\]

\[
\Psi^C(x_1, \ldots, x_p) = 1 - \sqrt{\frac{1}{p} \sum_{i=1}^{p} x_i^2}
\]

\[
\gamma^C(x, y) = |x - y|
\]

\[
\beta^R = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}
\]

\[
\gamma^R(a, b) = \min(a, b)
\]

\[
\Psi^R(y_1, \ldots, y_n) = \max(y_1, \ldots, y_n)
\]

In order to check the classification of unstored patterns and the recovery of the closest stored pattern, we will experiment with four test sets that we will store in four CCLAM, and we will proceed to recover the patterns when they have been affected by:

- Partial loss of the value of the pattern components
- Replacement of the value of some components by random values
- Appearance of additive noise of varying intensity

5.1. Test Sets

The experiments will be carried out with sets of 40 patterns of 25, 100, 400, and 1,600 components, which we will represent in square matrixes of 5 × 5, 10 × 10, 20 × 20, and 40 × 40 components, respectively.

In Figure 5, the stored patterns with 400 components are represented. The value of the components is shown with a shade of grey that ranges from black for the value zero to white for the value one.

The experiments consists of the following stages:

- Storage of the test set in a CCLAM.
- For each pattern X of the set:
  - Alter the pattern X obtaining X′.
  - Present X′ as stimulus in F_R.
  - Proceed with the classification and recovery processes to obtain the pattern Z.
  - If the original pattern X and the recovered one Z are the same, then the test is successful.
- Present the ratio of successfully recovered patterns.
To achieve more reliable results, the test is repeated 1,000 times to compute the mean of the obtained ratios.

5.2. Recovery from Incomplete Patterns

When information is lost in the communication channel, we have no choice but to recover information from incomplete patterns.

Experiments have been carried out to check the behavior of the CCLAM when it is a question of recovering a pattern which has lost part of the information. To
Table I. Experimental results with incomplete patterns.

<table>
<thead>
<tr>
<th>Size</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 × 5</td>
<td>99.9</td>
<td>99.3</td>
<td>97.2</td>
<td>88.0</td>
<td>64.0</td>
<td>28.3</td>
<td>15.3</td>
<td>4.5</td>
<td>4.2</td>
</tr>
<tr>
<td>10 × 10</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>83.7</td>
<td>61.6</td>
<td>15.9</td>
<td>5.0</td>
<td>4.8</td>
</tr>
<tr>
<td>20 × 20</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>90.2</td>
<td>62.3</td>
<td>10.0</td>
<td>8.7</td>
<td>7.3</td>
</tr>
<tr>
<td>40 × 40</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>96.8</td>
<td>36.0</td>
<td>5.0</td>
<td>5.0</td>
<td>5.0</td>
</tr>
</tbody>
</table>

Figure 6. Incomplete patterns.

Figure 7. Patterns contaminated by random noise.

do so, we have ascribed the value zero to the corresponding components for which information has been lost, and we have checked whether the memory is able to recover the original pattern (see Figure 6).

The success rate of recovering incomplete patterns in terms of the percentage of information loss is shown in Table I. The results obtained with the different pattern sizes used are shown in each row.

5.3. Recovery of Patterns with Incorrect Components

At times, the information received about a pattern can be partly wrong. For example, this can occur when a mistake occurs in a measuring instrument, or when the information comes from a badly informed source. Consequently, the value obtained from the affected component does not bear any relation to its real value.

We will simulate this situation by substituting the value of some components with random values. We will check the recovery capacity of the original pattern according to the quantity of erroneous components (see Figure 7).

Table II shows the success rate in recovering patterns according to the percentage of altered components. The results obtained with the different pattern sizes used are shown in each row.
### Table II. Experimental results with incorrect patterns.

<table>
<thead>
<tr>
<th>Size</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 × 5</td>
<td>100</td>
<td>99.7</td>
<td>98.8</td>
<td>95.5</td>
<td>86.4</td>
<td>68.9</td>
<td>45.4</td>
<td>23.4</td>
<td>9.0</td>
</tr>
<tr>
<td>10 × 10</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>99.6</td>
<td>92.5</td>
<td>61.5</td>
<td>19.6</td>
</tr>
<tr>
<td>20 × 20</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>99.8</td>
<td>75.8</td>
<td>10.7</td>
</tr>
<tr>
<td>40 × 40</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>98.9</td>
<td>8.6</td>
</tr>
</tbody>
</table>

### 5.4. Recovery of Patterns Affected by Noise

When we take a measurement, we err as a result of the precision of the measuring instrument, the number of measurements made, and the human ability to appreciate the value indicated on the meter. The information may also be disturbed in the communication channel for various reasons.

Because of this type of error, some components of the patterns are altered so that the value obtained is some distance from the real value, which depends on the intensity of the noise suffered or the range of the error made.

To simulate this situation, we will attempt to recover patterns affected by additive noise. To specify the noise present in the patterns, we will use two parameters: the percentage of components affected by noise and the noise intensity. The noise intensity will be indicated by the maximum range of error that is present in the patterns (see Figure 8).

The results obtained are shown in Tables III–VI. Each row corresponds to a percentage of components affected by noise, and each column corresponds to a different noise intensity.

![Patterns contaminated by additive noise.](image-url)
### Table III. Experimental results with $5 \times 5$ components noisy patterns.

<table>
<thead>
<tr>
<th></th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>20%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>99</td>
</tr>
<tr>
<td>30%</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
</tr>
<tr>
<td>40%</td>
<td>94</td>
<td>94</td>
<td>94</td>
<td>94</td>
<td>93</td>
<td>93</td>
<td>93</td>
<td>93</td>
<td>93</td>
<td>93</td>
</tr>
<tr>
<td>50%</td>
<td>84</td>
<td>83</td>
<td>83</td>
<td>83</td>
<td>82</td>
<td>83</td>
<td>83</td>
<td>83</td>
<td>83</td>
<td>83</td>
</tr>
<tr>
<td>60%</td>
<td>68</td>
<td>66</td>
<td>65</td>
<td>65</td>
<td>64</td>
<td>65</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>70%</td>
<td>46</td>
<td>43</td>
<td>43</td>
<td>43</td>
<td>42</td>
<td>42</td>
<td>42</td>
<td>42</td>
<td>42</td>
<td>42</td>
</tr>
<tr>
<td>80%</td>
<td>26</td>
<td>24</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>23</td>
<td>22</td>
<td>23</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>90%</td>
<td>12</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>100%</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

### Table IV. Experimental results with $10 \times 10$ components noisy patterns.

<table>
<thead>
<tr>
<th></th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>20%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>30%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>40%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>50%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>60%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>70%</td>
<td>98</td>
<td>97</td>
<td>96</td>
<td>96</td>
<td>96</td>
<td>96</td>
<td>96</td>
<td>96</td>
<td>96</td>
<td>96</td>
</tr>
<tr>
<td>80%</td>
<td>86</td>
<td>83</td>
<td>82</td>
<td>81</td>
<td>81</td>
<td>81</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>90%</td>
<td>56</td>
<td>51</td>
<td>50</td>
<td>49</td>
<td>49</td>
<td>48</td>
<td>48</td>
<td>48</td>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>100%</td>
<td>22</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>

### Table V. Experimental results with $20 \times 20$ components noisy patterns.

<table>
<thead>
<tr>
<th></th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>20%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>30%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>40%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>50%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>60%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>70%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>80%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>90%</td>
<td>86</td>
<td>79</td>
<td>77</td>
<td>76</td>
<td>75</td>
<td>74</td>
<td>74</td>
<td>73</td>
<td>74</td>
<td>74</td>
</tr>
<tr>
<td>100%</td>
<td>23</td>
<td>17</td>
<td>14</td>
<td>14</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
</tbody>
</table>

### Table VI. Experimental results with $40 \times 40$ components noisy patterns.

<table>
<thead>
<tr>
<th></th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>20%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>30%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>40%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>50%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>60%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>70%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>80%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>90%</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>100%</td>
<td>62</td>
<td>42</td>
<td>36</td>
<td>34</td>
<td>32</td>
<td>31</td>
<td>30</td>
<td>29</td>
<td>29</td>
<td>29</td>
</tr>
</tbody>
</table>

406
6. CONCLUSIONS AND FURTHER REMARKS

- The CCLAM does not impose any type of restriction on the patterns that are memorized, which is why there is no data dependency. Consequently, it allows any pattern to be stored.
- The dynamic topology of the CCLAM allows as many patterns as required to be stored without limiting the capacity by a static number of PEs.
- The CCLAM is more efficient than other models, so although fewer resources are used, better results are obtained.
- The correct choice of the functions that define the behavior of the memory enables the configuration of the CCLAM to adapt to our information storage and recovery needs.