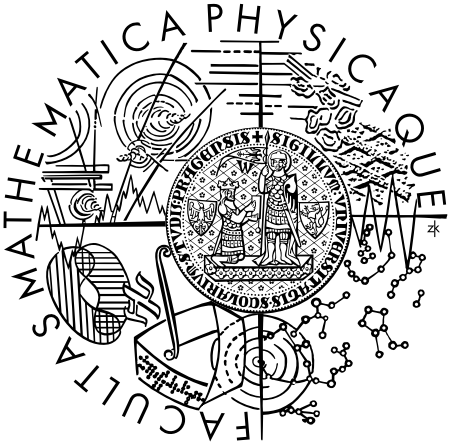
Charles University in Prague
Faculty of Mathematics and Physics

DOCTORAL THESIS



Jana Štanclová

Hierarchical associative memories

Department of Software Engineering

Advisor: RNDr. Iveta Mrázová, CSc.

Abstract

Title:	Hierarchical associative memories
Author:	RNDr. Jana Štanclová
	email: stanclova@ksi.ms.mff.cuni.cz
	phone: +420 2 2191 4260
Department:	Department of Software Engineering
	Faculty of Mathematics and Physics
	Charles University in Prague, Czech Republic
Advisor:	RNDr. Iveta Mrázová, CSc.
	email: mrazova@ksi.ms.mff.cuni.cz
	phone: +420 2 2191 4219
Mailing address (both Author and Advisor)	
	Dept. of Software Engineering, Charles University in Prague
	Malostranské nám. 25
	118 00 Prague, Czech Republic
WWW:	http://www.ksi.mff.cuni.cz/∼stanclova/

Abstract:

The progress in information technologies enables applications of artificial neural networks even in areas like navigation and geographical information systems, telecommunications, etc. This kind of problems requires robust recall of - possibly incomplete - spatial data. In this context, spatial patterns can be geographic maps, room plans, etc. Strategies for processing of spatial patterns can be based on associative memories. However, the performance of traditional associative memories (usually Hopfield-like networks) is very sensitive to the number of stored patterns and their mutual correlations.

In this thesis, we design and implement the so-called Hierarchical Associative Memory (HAM) model in order to avoid limitations imposed by processing of a large number of mutually correlated patterns. The proposed model consists of several associative memories grouped into layers. A suitable strategy applied during the network dynamics allows reliable processing of large amounts of spatial data. To evaluate the performance of the proposed HAM model, extensive simulations are carried out. The storage and recall abilities of the HAM model are studied experimentally and thoroughly compared with several other models of associative memories.

Keywords: associative memories, hierarchical associative memories, correlated patterns, processing of spatial patterns

Acknowledgments

I would like to thank all those who supported me in my doctoral study and in the work on my thesis. I very much appreciate the help and counseling received from my advisor Iveta Mrázová.

For the various help they provided me, I would also like to thank my colleagues; special thanks go to Jiří Dokulil, Filip Zavoral, David Obdržálek and Leo Galamboš. My thanks also go to Prof. František Plášil and Prof. Jaroslav Pokorný for their encouragement and empowering enthusiasm.

Last but not least, I am indebted to my parents, my husband Pavel and my daughter Terezka, whose support and patience made this work possible.

This work was partially supported by the Ministry of Education of the Czech Republic (grant MSM0021620838).

Contents

1 Introduction

An artificial neural network is an interconnected group of simple processing elements (the neurons) that uses a mathematical model for information processing. An artificial neural network is an adaptive system that changes its structure according to information that flows through the network. The original inspiration for artificial neural networks comes from examination of the central nervous system. Artificial neural networks process information in a way similar to how the human brain does. Artificial neural network applications differ greatly from other "computer-based" approaches to problem solving. Artificial neural networks are not programmed but rather they are "taught". They are used to "learn" patterns and relationships in them.

Research in the field of artificial neural networks has been attracting increasing attention in recent years. The first artificial neuron has been presented in 1943 by Warren McCulloch and Walter Pitts [47]. Their neuron has lacked the capability to learn. The next big development in artificial neural networks has been the publication of the book The Organization of Behavior by Donald Hebbs in 1949 [24] where he has described the training rule. The book has pointed out that neural pathways are strengthened each time that they are used. During the 1950's and 60's, Frank Rosenblatt has investigated the properties of mathematically described artificial neural networks.

Since then, new and more sophisticated proposals have been developed. In 1982, John Hopfield has developed the idea of an associative memory [30]. He has formulated the principle of storing and recalling information in an associative memory.

Although the original motivation for artificial neural networks has been to model the central nervous systems, today artificial neural networks are being applied in such fields as function approximation, regression analysis, classification (pattern and sequence recognition), data processing (filtering, clustering, compression), etc. The emerging development in information technologies enables the use of artificial neural networks also in areas like navigation, cartography or telecommunication. These research areas require processing of high-dimensional spatial information about the external world that corresponds in this context to spatial (geographical) plans, plans of buildings, etc.

1.1 Problem statement

Our research is focused on associative memories that represent a model of artificial neural networks applicable to information storage and recall. Associative memories are capable of recalling "partial" or "erroneous" information. They can be robust to noise and perturbation.

The inspiration for our research of associative memories comes from the robot operation and control. When designing an autonomous robot control, different information sources are usually taken into consideration [14]: stimulae from bumpers and other sensors, odometer's information, videocamera signal, beacon signals and many others. Most of the methods used to handle incoming measured data are burdened with errors and cumulative errors. The cor-

rect localization information usually cannot be maintained for longer time with sufficient precision using only a single information source. For example, the odometer's information originating from a car-like robot wheels cannot cope with wheel slides on a slippery surface or with robot position change if external forces shift the robot sideways. Thus, to localize the robot in its virtual world, overwhelming majority of implementations uses a combination of more methods of processing the source data.

Associative memories could be used to widen the palette of methods, which enable the control mechanisms of the robot to determine the robot location in the real world well. They could be used for example to compare or match against previously acquired (or learnt) "world map". They could help to eliminate or at least decrease the error coming from other localization algorithms.

Let us imagine a control and planning mechanism that has to guide a robot from one room in a building to another one. As the robot moves along the corridors, it has to turn on the junctions to follow the journey plan. Before it can actually turn, the robot must detect a junction and decide that junction exactly it is at. Associative memories could be used to process data originally coming from a camera attached to the robot. Thus, we could acquire a part of the scenery that is in fact a prediction what the robot will see in a near future (e.g. "behind a corner").

By means of associative memories, we could provide quite significant information to the robot control mechanism before it actually comes to the junction or the turn (and detects it using regular sensors). The control mechanism could have more time to compute its decision, which is helpful especially when computing power of the decision unit is not big. Regular processes would lead to a robot that stops every now and then to decide its next steps. Using associative memories, the control mechanism could start manoeuvring before it detects the junction via other sensors. It could make its move smoother or even possible at all (Figure 1).

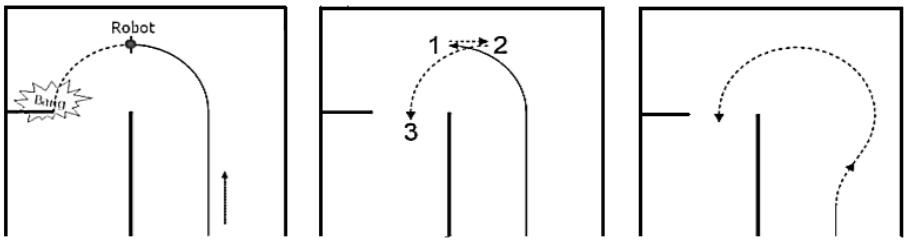


Figure 1: Inspiration for our research of associative memories: associative memories could help to guide a robot in a building.

When associative memories are used to help to navigate through the world, information processed by associative memories can be in the form of spatial patterns. Spatial patterns can correspond e.g. to fragments of sceneries.

The main limitations of traditional associative memories (usually Hopfield-like networks [30]) are related to their storage process. The storage capacity of traditional associative memories is usually very limited and the models are not

effective or even capable of recalling similar patterns. Applications can require processing a huge number of spatial patterns. In addition, real spatial patterns tend to be similar (correlated) to each other. These requirements greatly reduce the possibility of applying traditional associative memories in practice.

Therefore, we have decided to focus our research on the models of associative memories that allow handling a large number of (correlated) spatial patterns. Our research has been inspired by the models of cascade associative memories proposed by Hirahara et al. [26]. Unfortunately, the storage capacity of cascade associative memories is still insufficient for real applications where larger numbers of (correlated) patterns need to be processed. In our research, we develop a model of associative memories with an emphasis on the necessity to process a huge number of (correlated) patterns. We also implement the proposed model. The performance of the model is evaluated by means of experimental simulations.

1.2 Goals of the thesis

The main goal of the work is to design, implement and test an associative memory model. The new associative memory model should be developed with respect to allow processing a large number of (correlated) patterns. The new model should be implemented and its behaviour studied experimentally. The model should be compared with a standard associative memory and a cascade associative memory.

The goal of the work will be achieved by means of the following objectives:

1. **Study of associative memories and their properties**
   Associative memories as a kind of artificial neural networks can be used to information storage and recall. A standard associative memory (Hopfield-like network) and its properties will be studied.

   Application of a standard associative memory in practice is limited, as it cannot cope with the need for a high number of patterns to be stored and the fact that storing patterns can be correlated. Therefore, related models of associative memories will be studied. Four possible approaches to associative memories will be discussed: models improving training rules, models improving dynamics, cascade associative memories and models applicable to navigation problems.

2. **Study of associative memory approaches to spatial pattern processing**
   Various models of associative memories can be applied to spatial pattern processing. Several strategies for the storage and recall of spatial patterns will be outlined. The strategies will be discussed with respect to be applied in practice.

3. **Design (and implementation) of a hierarchical associative memory**
   Traditional associative memories have robust recall abilities, but their performance often depends on the number of stored patterns and it is sensitive

to dealing with correlated patterns. Therefore, a new model of associative memories (called hierarchical associative memory) will be developed with an emphasis on the necessity to process large amounts of (correlated) data. Since our research has been inspired by cascade associative memories [26], our model will be composed of associative memories grouped into several layers. The training and recall process of the new model will be investigated as well. The new model will be implemented.

4. **Performance analysis of the proposed associative memory model** To evaluate the performance of the proposed model, extensive simulations will be carried out. The experiments will show abilities of the designed model with respect to the recall of the stored patterns and the recall of incomplete patterns (patterns where a part of the information is unknown). The proposed model will be also compared with a standard associative memory and a cascade associative memory.

   The time complexity of the proposed model will be sketched. The theoretical number of associative memories in the proposed model will be derived. Theoretical results will be compared with experimental results.

1.3 Structure of the thesis

We describe the structure of the work in order to help with navigating through the thesis. The thesis starts with chapters explaining basic notions, various associative memory models and their properties (Chapters 2 - 4). These chapters do not present any novel work and summarize the state of art. A following chapter (Chapter 5) describes the problem of spatial pattern processing by means of associative memories. The work continues with the description of a new model of associative memories (the hierarchical associative memory) and experimental results (Chapters 6 - 7). The final part of the work is an evaluation and a conclusion of the thesis (Chapters 8 - 9).

Chapter 2 defines the basic notions from the area of artificial neural networks. In Chapter 3, a standard associative memory and its basic properties are described. Chapter 4 provides an overview of associative memory models that have been developed by various researchers to extend a standard associative memory model.

In Chapter 5, the goals of the thesis are elaborated. The problem of spatial pattern processing is described in detail. The requirements imposed on the associative memory models applicable to spatial pattern processing are presented. Various types of associative memories are discussed with respect to spatial pattern processing.

Chapter 6 presents our hierarchical associative memories (the HAM models). The training process and the recall process of the hierarchical associative memories are proposed. Our implementation of the hierarchical associative is outlined as well. The implemented hierarchical associative memory is to be found on the enclosed CD.

Chapter 7 is devoted to the analysis and the experimental results of the developed HAM models. The chapter is organized into five subsections. The

subsection 7.1 is focused on data processed in our experimental simulations. The subsection 7.2 describes the HAM models used in our experimental simulations. The subsections 7.3 and 7.4 are devoted to experimental results focused on abilities of the HAM models to recall the stored and incomplete patterns, respectively. Our models of the hierarchical associative memory are compared with the standard associative memory [30] and the cascade associative memory [26] with respect to recall the stored and incomplete patterns. The subsection 7.5 sketches the time complexity of the HAM models. The theoretical number of associative memories in the HAM models is derived and compared with experimental results.

Chapter 8 evaluates the requirements imposed on the developed hierarchical associative memories. The chapter also concludes the comparison of the hierarchical associative memories with the standard associative memory and the cascade associative memory.

Chapter 9 concludes the thesis and its main results. The main goals of the thesis are evaluated. This chapter also discusses possible directions for the future work.

1.4 Contributions and publications

The results presented in this work have already been published in proceedings of both international and local conferences.

Standard associative memories and several existing models proposed by various researchers to improve abilities of standard associative memories have been studied and reviewed in papers [81] and [80]. Since the performance of standard associative memories is very limited, our model of hierarchical associative memory (denoted as the HAM1 model) has been introduced in [55]. Our hierarchical associative memories consist of associative memories grouped into several layers.

The abilities of the proposed HAM1 model and suitable strategies applied during the training process have been discussed in papers [76] and [78]. The possible parallel implementation of the HAM1 model has been described in papers [77] and [79]. The results of extensive experiments with respect to the storage and recall abilities of the HAM1 model have been published in [73]. Papers [74] and [75] have been focused on experiments with the HAM1 model performed so far for their possible use for mobile autonomous robots.

And finally, the HAM2 model where patterns are hierarchically grouped according to the "previous layer" information has been presented in [71]. The paper has also shown the storage and the recall abilities of the HAM2 model in comparison with the previous HAM1 model. Other results of experimental simulations focused on the performance of the HAM2 model have been published in [72].

The earlier work in the area of preprocessing patterns to be stored in associative memories (marginally mentioned in this thesis but tied with the associative map recall problem) has been published in [54].

Articles published in proceedings of world-wide conferences

[71] Štanclová, J.: The Associative Recall of Spatial Correlated Patterns, Pro-

ceedings of CIARP 2006, Cancun, Mexico, Copyright (C) Springer-Verlag, Berlin, LNCS4225, ISSN 0302-9743, ISBN 978-3-540-46556-0, pp. 539-548, 2006

[73] Štanclová, J., Zavoral, F.: Hierarchical Associative Memories: The Neural Network for Prediction in Spatial Maps, Proceedings of ICIAP 2005, Cagliary, Italy, Copyright (C) Springer-Verlag, Berlin, LNCS3617, ISSN-0302-9743, ISBN 3-540-28869-4, pp. 786-793, 2005

[74] Štanclová, J., Obdržálek, D.: Map Recall based on Hierarchical Associative Memories, Proceedings of DARH 2005, Yverdon-les-Bains, Switzerland, ISBN 2-8399-0085-8, pp. 44-49, 2005

[76] Tesková (Štanclová), J.: Hierarchical Associative Memories for Path Prediction, Proceeding of SSGRR 2003, L'Aquila, Italy, Telecom Italia Learning Services S.p.A., ISBN 88-85280-74-9, 2003

[77] Tesková (Štanclová), J.: Hierarchical Associative Memories: The Parallel Implementation of The Model, Proceedings of IPSI-2003, Sveti Stefan, Serbia and Montenegro, ISBN 86-7466-117-3, 2003

Articles published in proceedings of other conferences

[72] Štanclová, J.: Asociativní paměti pro ukládání korelovaných vzorů (in czech), Proceedings of ITAT 2006, Bystrá dolina, Slovakia, ISBN 80-969184-4-3, pp. 173-178, 2006

[75] Štanclová, J., Obdržálek, D.: Použití Hierarchického asociativního modelu neuronové sítě pro zlepšení lokalizace autonomního robota (in czech), Proceedings of ITAT 2005, Račkova dolina, Slovakia, ISBN 80-7097-609-8, pp. 157-166, 2005

[78] Tesková (Štanclová), J.: Associative Recall by Means of Hierarchical Associative Memories, Proceedings of CALCI 2003, Stará Lesná, Slovakia, ISBN 80-89066-64-X, pp. 369-376, 2003

[55] Mrázová, I. , Tesková (Štanclová), J.: Hierarchical Associative Memories for Storing Spatial Patterns, Proceedings of ITAT 2002, Malinô Brdo, Slovakia, ISBN 80-7097-499-0, pp. 143-154, 2002

[79] Tesková (Štanclová), J.: Implementation of Hierarchical Associative Memories, Proceedings of WDS 2002, Praha, Czech Republic, MatfyzPress ISBN 80-85863-88-X, pp. 9-16, 2002

[80] Tesková (Štanclová), J.: Associative Memories for Path Prediction, Proceedings of WDS 2001, Praha, Czech Republic, ISBN 80-85863-73-1, pp. 198-204, 2001

[81] Tesková (Štanclová), J.: Associative Recall of Spatial Maps by Means of Hopfield Model, Proceedings of Nostradamus 2001, Zlín, Czech Republic, ISBN 80-7318-030-8, 2001

[54] Mrázová, I., Tesková (Štanclová), J.: Path Prediction in Geographic Maps, Proceedings of Nostradamus 2000, Zlín, Czech Republic, ISBN 80-214-1668-8, pp. 190-195, 2000

2 Basic notions

In this chapter, we define several basic notions to be used further in the thesis. First, a mathematical model of a neuron is introduced. The second part of the chapter is devoted to artificial neural networks (called neural networks, for short). It shows the structure of neural networks, the training process and the recall process.

2.1 Neuron

Definition 2.1 *A neuron with the weight vector* $\vec{w} = (w_1, \ldots, w_n)$, $w_1, \ldots, w_n \in \mathbb{R}$ *and the transfer function* $f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ *is an abstract device capable of computing the output* $y \in \mathbb{R}$ *as the value of the transfer function* f*:*

$$y = f[\vec{w}](\vec{z}) \tag{1}$$

for any input vector $\vec{z} = (z_1, \ldots, z_n)$, $z_1, \ldots, z_n \in \mathbb{R}$*;* $\mathbb{R}$ *denotes the set of all real numbers.*

Unless explicitly stated otherwise, we consider the hard-limiting transfer function *sgn* with values equal to -1 or 1:

$$y = f[\vec{w}](\vec{z}) = \mathrm{sgn}(\xi) = \begin{cases} 1 & \text{if } \xi > 0 \\ -1 & \text{otherwise.} \end{cases} \tag{2}$$

In the above relationship, ξ denotes the so-called neuron potential value that can be determined as

$$\xi = \sum_{i=1}^{n} z_i w_i, \tag{3}$$

where w_i and z_i $(1 \leq i \leq n)$ denote the elements of the neuron's weight and input vector, respectively. The structure of a neuron is shown in Figure 2.

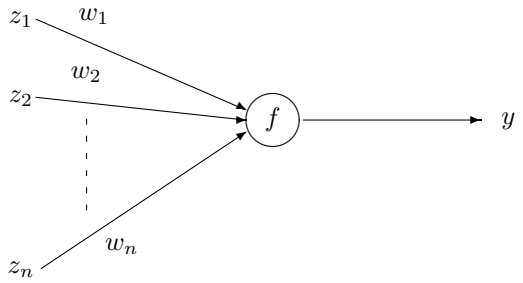


Figure 2: The model of a neuron with the input vector $\vec{z} = (z_1, \ldots, z_n)$, the weight vector $\vec{w} = (w_1, \ldots, w_n)$, the transfer function f and the output y.

Definition 2.2 *Let $\vec{x} = (x_1, \ldots, x_n)$ be $n-$dimensional vector. The vector $\vec{x}$ is bipolar if $x_i \in \{-1, +1\}$ for all $i = 1, \ldots, n$.*

Unless explicitly stated otherwise, we restrict our consideration to the bipolar vectors.

Definition 2.3 *Let $\vec{x} = (x_1, \ldots, x_n) \neq \vec{0}$ and $\vec{y} = (y_1, \ldots, y_n) \neq \vec{0}$ be $n-$dimensional vectors.*

- *The vectors $\vec{x}$ and $\vec{y}$ are orthogonal if*

$$\frac{1}{n} \sum_{i=1}^{n} x_i y_i = 0. \tag{4}$$

- *The vectors $\vec{x}$ and $\vec{y}$ are correlated if*

$$\frac{1}{n} \sum_{i=1}^{n} x_i y_i \neq 0. \tag{5}$$

For two bipolar vectors $\vec{x} = (x_1, \ldots, x_n) \neq \vec{0}$ and $\vec{y} = (y_1, \ldots, y_n) \neq \vec{0}$, it holds

$$| \frac{1}{n} \sum_{i=1}^{n} x_i y_i | \leq 1 \tag{6}$$

where n $(n > 0)$ is the dimension of vectors. The term

$$| \frac{1}{n} \sum_{i=1}^{n} x_i y_i | \tag{7}$$

corresponds to the correlation between the vectors $\vec{x}$ and $\vec{y}$. Two bipolar vectors $\vec{x}$ and $\vec{y}$ are strongly correlated when the correlation term is high.

Definition 2.4 *Let $\vec{x} = (x_1, \ldots, x_n)$ and $\vec{y} = (y_1, \ldots, y_n)$ be $n-$dimensional vectors. A Hamming distance Hamm of $\vec{x}$ and $\vec{y}$ is a function given by the formula*

$$Hamm(\vec{x}, \vec{y}) = \sum_{i=1}^{n} \Theta(| x_i - y_i |) \tag{8}$$

where $\Theta(u)$ is the step function

$$\Theta(u) = \begin{cases} 1 & \text{if } u > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

The symbol $| \cdot |$ denotes the absolute value.

2.2 Neural networks

Definition 2.5 *A neural network M is a 5-tuple $M = (N, C, I, O, w)$, where*

- *N is a finite non-empty set of neurons,*
- *$C \subseteq N \times N$ is a set of oriented inter-connections among neurons,*
- *$I \subseteq N$ is a non-empty set of input neurons,*
- *$O \subseteq N$ is a non-empty set of output neurons and*
- *$w : C \to \mathbb{R}$ is a weight function.*

The pair (N, C) forms an oriented graph that is called the topology of M.

Outputs of all neurons in the neural network define the state of the neural network. Further, we denote weights of all neurons in the neural network as a neural network configuration. Two different calculations in the neural network are distinguished: the training process and the recall process.

The recall process is given by the initial neural network state and the way the neural network state updates over time. In the initial neural network state, the outputs of all input neurons are set according to the presented input vector (the input pattern) and others are set to an initial value. After the initialization, the neural network state is usually updated only at discrete time steps $1, 2, 3, \ldots$. In general, a continuous time update can be considered as well. At every time step, one or more neurons are selected to update their output according to the corresponding neuron's inputs. The way of selecting the neurons to be updated depends on a given model of the neural network.

The output of all output neurons is called the neural network output (the output pattern). The output of the neural network can change over time. The recall process of the neural network can be defined in such a way that the neural network output does not vary after some time.

The training process is defined by the initial neural network configuration and the way the neural network configuration updates over time. First, the initial neural network configuration is chosen (e.g. the weights can be set to random values). After the initialization, the neural network configuration is updated. Generally, a continuous time update of the neural network configuration is considered. In practice, the configuration can be updated only at discrete time steps. The way of updating the neural network configuration depends on a given model of the neural network.

The aim of the training process is to find the appropriate neural network configuration such that the neural network produces the desired output. In order to update the neural network configuration, a training set is usually available. It characterizes the desired performance of the neural network.

There are two main training paradigms: supervised and unsupervised. In the supervised training, the network is provided with a desired output pattern for every input pattern. The training set consists of pairs of the input pattern and the corresponding output pattern. In contrast, the unsupervised training

does not require a desired output pattern associated with every input pattern. In this case, only input patterns form the training set.

3 Associative memories

Associative memories are a class of artificial neural networks that have been the subject of research since the early seventies. The recent interest in associative memories has been spurred by the seminal work of Hopfield in the early eighties. Hopfield [30] has formulated the physical principle of storing and recalling information in an associative memory through the collective computing. Since then, a number of important contributions have appeared addressing various issues of associative memories - training algorithms, capacity, recall dynamics, etc. Associative memories have been also studied as possible models of biological associative phenomena, cognition and categorical perception.

In generally, associative memories are treated as nonlinear dynamical systems where the information recall is realized as an evolution of the system's state in a high-dimensional state space. Associative memories are used to associate one set of patterns with another set of patterns; input and output patterns, respectively. The aim of an associative memory is to recall the associated pattern whenever the input pattern is presented to the memory.

Associative memories utilize training algorithms to store patterns. The stored patterns should correspond to stable states of the memory. The recall of stored patterns starts after initializing the associative memory with an input pattern. The associative memory performs a collective computing to recall the output pattern the best associated with the input pattern. The associative memory may, however, also recall patterns that are not associated with any of the stored patterns. An important property of the associative memory is the ability to recall a stored pattern, given a reasonable subset of information content of that pattern.

The main area of application of associative memories involves recalling of noisy, distorted or incomplete version of the stored patterns. They can be used in applications as pattern recognition to recover data when the available information is imprecise. Associative memories can have a natural mapping onto parallel hardware. They can be used for the information retrieval from e.g. heterogenous databases. Associative memories can help to understand the information processing in the cortex. Associative memories can be also applied to solve optimization problems, such as the combinatoric best route for a traveling salesman.

3.1 Types of associative memories

Associative memories are neural networks that map a set of input patterns to a set of output patterns. We may distinguish two basic types of associative memories - heteroassociative and autoassociative memories. In an autoassociative memory, the recalled pattern should correspond to the stored pattern that most closely resembles the presented pattern. A heteroassociative memory recalls the pattern that is, in general, different from the input pattern (not only in "content" but possibly also in "type" and "format").

In more detail, a heteroassociative memory maps m input patterns $\vec{x}^1, \ldots, \vec{x}^m$ to m output patterns $\vec{y}^1, \ldots, \vec{y}^m$, so that $\vec{x}^p \rightarrow \vec{y}^p$ $(p = 1, \ldots, m)$. Moreover,

if $\|\vec{x}' - \vec{x}^p\|^2 < \varepsilon$ ($\varepsilon > 0$) then $\vec{x}' \to \vec{y}^p$ ($1 \leq p \leq m$). Autoassociative memories are a special subset of heteroassociative memories. In an autoassociative memory, each pattern is associated with itself, i.e. $\vec{y}^p = \vec{x}^p$ for $p = 1, \ldots, m$.

Associative memories can be implemented as neural networks with or without feedback. In the simplest kind of the feedback, the output of a network is used repetitively as a new input until a concrete condition is fulfilled (e.g. no further changes in the output of the memory). This kind of the feedback is utilized e.g. in autoassociative memories of the Hopfield type [30].

Another kind of the feedback is used e.g. in bidirectional associative memories (BAM) [42]. The BAM models are made up of two sets of neurons where the information is sent recursively between them. The neurons in the first set receive the input pattern and compute their output. The output of the first set is transfered to the second set of neurons, which then send the output of their computation back to the first set using the same inter-connections. This process is repeated until a concrete condition is fulfilled (e.g. the information sent back and forth in the memory does not change further).

Further in the text, we mainly focus on autoassociative memories of the Hopfield type, that we call associative memories unless explicitly stated otherwise.

3.2 Structure of associative memories

Definition 3.1 *An associative memory $AM = (N, C, I, O, w)$ with n ($n > 0$) neurons is a neural network for which*

- *all neurons are input and output neurons simultaneously ($N = I = O$),*
- *there are oriented inter-connections among all neurons ($C = N \times N$),*
- *the weights are symmetric ($w_{ij} = w_{ji} \;\; \forall\, i, j \in N$),*
- *each neuron is connected to all other neurons except itself ($w_{ii} = 0 \;\; \forall\, i \in N$).*

An input pattern $\vec{x} = (x_1, \ldots, x_n)$ is presented to all (input) neurons of AM. An output pattern of AM is a vector $\vec{y} = (y_1, \ldots, y_n)$ formed by the outputs of all (output) neurons of AM. An output pattern is also called a state of AM. The structure of an associative memory is shown in Figure 3.

A weight matrix W of AM is $n \times n$ matrix $W = (w_{ij})$, where w_{ij} denotes the weight between the neurons i and j in AM ($1 \leq i, j \leq n$). Unless explicitly stated otherwise, we restrict our considerations to processing bipolar patterns.

Definition 3.2 *Let AM be an associative memory with n neurons. A training set T is a finite non-empty set of m patterns $\vec{x}^p$ ($p = 1, \ldots, m$):*

$$T = \{\vec{x}^1, \ldots, \vec{x}^m \mid \vec{x}^p \in \{+1, -1\}^n,\ p = 1, \ldots, m\}.$$

The pattern $\vec{x}^p$ is called a training pattern.

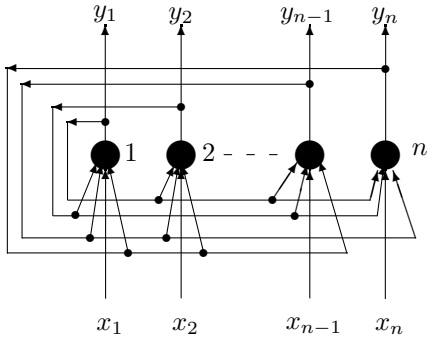


Figure 3: The structure of an associative memory with n neurons, the input pattern $(x_1, \ldots, x_n)$ and the output pattern $(y_1, \ldots, y_n)$

3.3 Training process

During training, associative memories adjust weights of all neurons with the aim to store each training pattern. To train associative memories the so-called Hebbian training can be applied. This training rule has been proposed by the psychologist Donald Hebb in 1949 [24]. The idea is based on the observation that two neurons that are simultaneously active should develop a higher degree of mutual interaction than those neurons whose activities are not correlated. In the latter case, the interaction between the neurons should be very low or zero [51].

Now, we describe in more detail the training process of an associative memory with n $(n > 0)$ neurons. Before the training process starts, the weight matrix $W = (w_{ij})$ of the associative memory AM is initially set to zero. Afterwards, a training pattern $\vec{x}^p \in T$, $\vec{x}^p = (x_1^p, \ldots, x_n^p)$, $1 \leq p \leq m$ is presented to the associative memory AM. For each weight w_{ij} of AM, the weight update $\Delta^p w_{ij}$ is given by

$$\Delta^p w_{ij} = \begin{cases} x_i^p x_j^p & \text{for } i \neq j \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where $1 \leq i, j \leq n$ [30]. The increment $\Delta^p w_{ij}$ measures the correlation between the elements x_i^p and x_j^p of the training pattern $\vec{x}^p$.

After presenting all training patterns, the elements of the weight matrix $W = (w_{ij})$ can be written in the form

$$w_{ij} = \begin{cases} \sum\limits_{p=1}^{m} \Delta^p w_{ij} = \sum\limits_{p=1}^{m} x_i^p x_j^p & \text{for } i \neq j \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

for $1 \leq i, j \leq n$. This is equivalent to

$$W = W^1 + \ldots + W^m$$

$$
\begin{aligned}
&= \Big((\vec{x}^1)^T\vec{x}^1 - I\Big) + \ldots + \Big((\vec{x}^m)^T\vec{x}^m - I\Big) \\
&= (\vec{x}^1)^T\vec{x}^1 + \ldots + (\vec{x}^m)^T\vec{x}^m - mI \qquad (12)
\end{aligned}
$$

where I denotes the $n \times n$ identity matrix. Hence, the weight matrix W of the associative memory AM is symmetric with zero diagonal.

Now, we discuss the situation when the stored pattern $\vec{x}^p \in T$ $(1 \leq p \leq m)$ is presented to already trained associative memory AM. The output of all neurons in AM corresponds to

$$
\begin{aligned}
\mathrm{sgn}(\vec{x}^pW) &= \mathrm{sgn}\Big(\vec{x}^p(W^1 + \ldots + W^m)\Big) \\
&= \mathrm{sgn}\Big(\vec{x}^p(\vec{x}^1)^T\vec{x}^1 \ + \ldots + \vec{x}^p(\vec{x}^m)^T\vec{x}^m - m\vec{x}^pI\Big). \qquad (13)
\end{aligned}
$$

Since $\vec{x}^p(\vec{x}^p)^T = n$ for any $n-$dimensional bipolar pattern $\vec{x}^p$, the output of all neurons can be written in the form

$$
\mathrm{sgn}(\vec{x}^pW) \quad = \quad \mathrm{sgn}\Big((n-m)\vec{x}^p \ + \ \sum_{\substack{i=1 \\ i\neq p}}^{m} \vec{x}^p(\vec{x}^i)^T\vec{x}^i\Big). \qquad (14)
$$

Hence, the pattern $\vec{x}^pW$ is equal to the presented pattern $\vec{x}^p$ multiplied by a constant $(n-m)$ plus an additional term

$$
\sum_{\substack{i=1 \\ i\neq p}}^{m} \vec{x}^p(\vec{x}^i)^T\vec{x}^i \qquad (15)
$$

called a crosstalk. The associative memory AM produces the output pattern $\vec{x}^p$ (i.e. $\vec{x}^p = \mathrm{sgn}(\vec{x}^pW)$) when $m < n$ and the crosstalk is zero [4]. This is the case whenever the number of training patterns m is smaller than the number of neurons n and the training patterns $\vec{x}^1, \ldots, \vec{x}^m$ are pairwise orthogonal.

Moreover, this mapping can still work even if the crosstalk differs from zero. In this case, the crosstalk is sufficiently smaller than $(n-m)\vec{x}^p$. If $m < n$, it must hold

$$
\mathrm{sgn}(\vec{x}^pW) = \mathrm{sgn}\left(\vec{x}^p + \sum_{\substack{i=1 \\ i\neq p}}^{m} \frac{\vec{x}^p(\vec{x}^i)^T}{n-m}\vec{x}^i\right). \qquad (16)
$$

In order to produce the pattern $\vec{x}^p$, we must get $\vec{x}^p = \mathrm{sgn}(\vec{x}^pW)$. It means that the absolute value of all elements of the term

$$
\sum_{\substack{i=1 \\ i\neq p}}^{m} \frac{\vec{x}^p(\vec{x}^i)^T}{n-m}\vec{x}^i \qquad (17)
$$

is smaller than 1 (for $m < n$).

In the above-mentioned cases, the Hebbian training rule leads to an efficient set of weights in the associative memory AM with n neurons [4].

3.4 Recall process

In this section, we focus on the recall process in more detail. We describe the recall dynamics of associative memories. We also show the convergence of the recall process.

The recall process proceeds in associative memories iteratively. First, a pattern $\vec{x} = (x_1, \ldots, x_n)$ is presented to the associative memory AM. The pattern $\vec{x}$ represents an input pattern of AM. Before the recall process starts ($t = 0$), the initial output $\vec{y}(0) = (y_1(0), \ldots, y_n(0))$ of the associative memory is identical to the presented input pattern $\vec{x}$ (i.e. $\vec{y}(0) = \vec{x}$). In the following time steps, the actual outputs of all neurons in the network represent a new input to the neurons. In every time step t ($t > 0$), one or more neurons i ($1 \leq i \leq n$) are selected to update their output $y_i(t)$ according to the rule

$$y_i(t) = \operatorname{sgn}\Big(\sum_{j=1}^{n} w_{ji} y_j(t-1) \Big) \tag{18}$$

where $y_j(t-1)$ is the output of the neuron j at the time step $t-1$ and w_{ij} is the weight between the neurons j and i. The output of other neurons remains unchanged until they are selected for an update (the so-called asynchronous dynamics of the network). Anyway, all neurons in the associative memory should be updated approximately at the same average rate [64]. Often, a uniformly random update schedule is applied. Hence, the time evolution of the network output pattern depends on the chosen update schedule [64].

In the first time step, the associative memory computes the output pattern $\vec{y}(1) = (y_1(1), \ldots, y_n(1))$ that can be written as

$$y_i(1) = \begin{cases} \operatorname{sgn}\Big(\sum\limits_{j=1}^{n} w_{ji} y_j(0) \Big) & \text{if the neuron } i \text{ is selected for an update} \\ y_i(0) & \text{otherwise} \end{cases} \tag{19}$$

for $1 \leq i \leq n$.

After t iterations (the time steps), the associative memory produces a sequence of $t+1$ outputs $\vec{y}(0), \ldots, \vec{y}(t)$ for which

$$y_i(\tau) = \begin{cases} \operatorname{sgn}\Big(\sum\limits_{j=1}^{n} w_{ji} y_j(\tau-1) \Big) & \text{if the neuron } i \text{ is selected for an update} \\ & \text{at the time step } \tau \\ y_i(\tau-1) & \text{otherwise} \end{cases} \tag{20}$$

where $\vec{y}(\tau) = (y_1(\tau), \ldots, y_n(\tau))$, $1 \leq i \leq n$ and $0 < \tau \leq t$.

A recalled pattern $\vec{y}$ is defined as an output pattern $\vec{y}(t^*)$, $t^* \geq 0$ that fulfills the condition

$$\vec{y}(t^* + \tau) = \vec{y}(t^*) \tag{21}$$

where $\tau = 1, 2, \ldots$. Hence, the asynchronous update is continued until there are no further changes in the output of the associative memory [64]. The recalled

pattern corresponds to the stable state of the associative memory. In generally, the dynamics of the associative memory does not necessarily lead to stable states. It can lead to oscillations and output patterns can change cyclically [64].

In order to characterize the behaviour of associative memories in more detail, the concept of the so-called energy function can be applied.

Definition 3.3 *Let W denote the weight matrix of an associative memory AM with n ($n > 0$) neurons. Let $\vec{y} = (y_1, \ldots, y_n)$ be a state of the associative memory AM. The energy $E(\vec{y})$ of the associative memory AM at the network state $\vec{y}$ is given by*

$$E(\vec{y}) = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} y_i y_j. \tag{22}$$

It can be shown that associative memories converge to stable states [30].

Theorem 3.4 *An associative memory AM with n ($n > 0$) neurons and asynchronous dynamics, which starts from any given network state, eventually reaches a stable state at a local minimum of the energy function.*

Proof. [30] The proof of the theorem relies on analysis of the new value of the energy function after each neuron update. The energy function of the associative memory AM at the state $\vec{y} = (y_1, \ldots, y_n)$ is given by Equation 22.

If the neuron k ($1 \leq k \leq n$) is selected for an update and does not change its output, then the energy $E(\vec{y})$ of the associative memory does not change either. If the output of the neuron k is changed, the associative memory AM reaches a new network state $\vec{y}' = (y_1, \ldots, y_{k-1}, y'_k, y_{k+1}, \ldots, y_n)$ for which the new energy is $E(\vec{y}')$. The difference between $E(\vec{y})$ and $E(\vec{y}')$ is given by all terms in the summation (Equation 22) that contain y_k and y'_k. It is

$$E(\vec{y}) - E(\vec{y}') = \Big(-\sum_{i=1}^{n} w_{ik} y_i y_k \Big) - \Big(-\sum_{i=1}^{n} w_{ik} y_i y'_k \Big). \tag{23}$$

The factor $1/2$ disappears from the computation because the terms $w_{ik} y_i y_k$ appear twice in the double sum of Equation 22. Because of the absence of an inter-connection from each neuron to itself (i.e. $w_{ii} = 0$), Equation 23 can be rewritten as

$$E(\vec{y}) - E(\vec{y}') = -(y_k - y'_k) \sum_{i=1}^{n} w_{ik} y_i. \tag{24}$$

Equation 24 can be expressed as

$$E(\vec{y}) - E(\vec{y}') = -(y_k - y'_k) \xi_k, \tag{25}$$

where ξ_k denotes the potential of the neuron k (Equation 3). The potential ξ_k has a different sign from the term $(y_k - y'_k)$, because otherwise the output of the neuron k would not have been changed. This means that the product $-(y_k - y'_k)\xi_k$ is positive and therefore

$$E(\vec{y}) - E(\vec{y}') \;\; > \;\; 0. \tag{26}$$

It shows that every time the output of a neuron is altered, the total energy of the associative memory AM is reduced. Since there is only a finite set of possible network states, the associative memory AM must eventually reach a state where the energy cannot be reduced further. It is a stable state of the network, as we wanted to prove.

□

For (auto)associative memories with symmetric weights and no inter-connection from a neuron to itself, the recall process eventually reaches a stable state at a local minimum of the energy function after a finite number of time steps. That the memory eventually settles in a stable state means that the probability of not reaching such a state approaches zero as the number of iterations increases. If the weight matrix of the memory does not contain a zero diagonal (i.e. the inter-connection from a neuron to itself), the recall process does not necessarily lead to stable states (an example e.g. in [64]). The dynamics of an (auto)associative memory where a weight matrix contains a zero diagonal can also lead to oscillations in a case where the weights are not symmetric [30]. The symmetry of the weight matrix and a zero diagonal are thus necessary for the convergence of an (auto)associative memory [64].

In heteroassociative memories with the simplest feedback (i.e. the models where an output of a memory is used as a new input), the weights are generally asymmetric. The recall process cannot be regarded as a monotonically decreasing process in the energy function. Hence, the convergence of the recall process cannot be guaranteed [64]. When another kind of the feedback is used in heteroassociative memories (e.g. the BAM models), the memory can reach a stable state after a finite number of time steps.

During the recall process, an associative memory does not always go from a given network state to the nearest stable state (with the distance measured by the Hamming distance). Sometimes, it goes to a stable state that is farther away than the nearest stable state. No way how to fix this problem is known (this problem probably cannot, in general, be fixed) [57].

Definition 3.5 *Let AM denote an associative memory. A basin of attraction of a pattern $\vec{x}$ in AM is a region of the input space, in which all included patterns converge to the same stable state of the pattern $\vec{x}$ when recalled by AM.*

In associative memories, the number of stable states increases exponentially with the number of neurons [20]. An associative memory approaches a stable state that corresponds to one of the stored patterns or the so-called spurious pattern. A spurious pattern is a stable state that differs from any stored pattern. Unfortunately, it is not possible to determine without the information about the stored patterns, which stored pattern is reached.

3.5 Storage capacity

The basins of attraction of stored patterns deteriorate every time when a new pattern is stored in an associative memory. If the crosstalk becomes too large, it can even happen that previously stored patterns are lost [4]. To reduce the

probability that the stored patterns are lost, some limits are put to the number of stored patterns m (in comparison with the number of neurons n).

Definition 3.6 *Let AM be an associative memory with n ($n > 0$) neurons.*

- *A loading rate α_r of AM is a ratio between the number of stored patterns m and the number of neurons n (i.e. $\alpha_r = \frac{m}{n}$).*
- *A storage capacity α_c of AM is defined as a ratio between the maximum number of patterns stored in AM as stable states of the energy function and the number of neurons n.*

By computer simulations, Hopfield has demonstrated that the storage capacity α_c of an associative memory is about 0.15 in a case when orthogonal (or close to orthogonal) patterns are stored in it [30]. Anyway, storing correlated patterns can cause serious problems both during the training and recall processes. Previously stored training patterns can even become lost after presenting other patterns correlated to the stored ones. This effect is caused by large values of the crosstalk that changes the sign of computed potentials [4].

Assume that neurons in an associative memory are selected to update their outputs independently with equal probability. During recalling a pattern, a neuron in an associative memory becomes stable if its output is not going to change anymore. The probability P that a neuron in an associative memory is stable can be estimated as

$$P = \frac{1}{2} - \frac{1}{\sqrt{\pi}} \int_0^{\sqrt{n/2m}} e^{-x^2} dx. \tag{27}$$

for sufficiently large numbers n, m of neurons and stored training pattern, respectively [25]. For example, it is expected that for $\alpha_r = 0.185$ the number of unstable neurons in the associative memory does not exceed 1% [70].

However, results do not say anything about the output pattern of an associative memory in the next time step when the respective unstable neuron is updated (e.g. it does not guarantee that the memory reaches a stable state close to the underlying stored pattern). A similar analysis concerning the recall of most or all stored patterns (unlike the recall of only a particular neuron) has shown that the storage capacity of the associative memories is asymptotically proportional to $n/\log n$ [88] [48].

However, associative memories should possess more than the storage and recall of all training patterns, but they should also recall "incomplete" or "imprecise" patterns. A more detailed analysis has shown that all stored patterns correspond to local minima of the energy function for $\alpha_r \leq 0.138$ [3] [4] [63]. For $\alpha_r > 0.138$, the local minima corresponding to the stored patterns disappear. Furthermore, for $\alpha_r < 0.05$ the stored patterns correspond to global minima of the energy function [3].

4 Models of associative memories

The storage capacity α_c of a standard associative memory is about 0.15 when the stored patterns are (nearly pairwise) orthogonal [30]. If the number of stored patterns does not exceed this limit, the model has robust ability to recall the stored patterns. Disadvantage of the standard associative memory is the fact that the stored patterns have to be (almost) orthogonal to one another. These limitations reduce the possibility to apply the standard associative memory in practice.

To overcome this problem, various modified models of associative memories have been proposed to improve these abilities: e.g. Feigelman and Ioffe [15], Gutfreund [22], Morita [53], Kakeya and Kindo [35], Hirahara, Oka and Kindo [26], [27], [28], Parga and Rolls [62], Elliffe et al. [13], Kimoto and Okada [39], Itoh and Shimizu [32], [33], [19]. We focus on four possible approaches how researchers have extended the standard associative memory model. The approaches are devoted to:

- models improving training rules
- models improving dynamics
- cascade models of associative memories
- models applicable to navigation

In this chapter, we provide a short overview of several models of associative memories. Similar overview can be found in our previous papers [81], [80]. This chapter does not present any novel work.

4.1 Models improving training rules

Models with various training rules have been designed to improve the storage process of the standard associative memory. Such models involve a much more complicated dependance of the weights on stored patterns (e.g. self-coupling, cross-correlation, etc.). We briefly discuss such models and their abilities.

4.1.1 Models for storing biased patterns

Amit et al. [4] have proposed a model of associative memory for storing and recall of the so-called biased patterns. Every element x_i of the bipolar pattern $\vec{x} = (x_1, \ldots, x_n)$ is chosen independently with the probability

$$P(x_i) = \frac{1}{2}(1+a)\delta(x_i - 1) + \frac{1}{2}(1-a)\delta(x_i + 1) \tag{28}$$

where $\delta(u) = 1$ for $u = 0$, and zero otherwise. Such patterns are characterized by a bias parameter a $(-1 < a < +1)$.

The patterns $\vec{x}^1, \ldots, \vec{x}^m$ with the finite bias a $(-1 < a < +1)$ are stored in the model by modifying the training rule to

$$w_{ij} = \frac{1}{n}\sum_{k=1}^{m}(x_i^k - a)(x_j^k - a) \tag{29}$$

where $\vec{x}^k = (x_1^k, \ldots, x_n^k)$ is a bipolar pattern and $i, j = 1, \ldots, n$. The symbol n ($n > 0$) denotes the number of neurons and a is the bias parameter.

The main virtue of this model is that it retains the simplicity of the associative memory training rule. Amit et al. [4] have shown that the stored patterns are recalled with a small fraction of errors, up to a storage level of $\alpha_c(a) \cdot n$. However, the storage capacity decreases sharply with a. As the bias a increases, the number of spurious patterns increases and they become the absolute minima of the energy function. Although the stored patterns correspond to the network stable states below $\alpha_c(a)$ (up to a small fraction of errors), the model is suited for treating patterns with the small bias [4].

4.1.2 Associative memories with the self-coupling

Training rules for traditional associative memories are generally used to obtain the off-diagonal elements of the weight matrix. The diagonal elements represent the neuron self-interaction (the self-coupling). They are generally considered to be a nuisance [25] and are set to zero. Several analyses have shown that, in contrast to general belief, the presence of the self-interaction may even improve the performance the associative memory (e.g. [18], [6], [68]).

Fontanari et al. [18] have carried out the equilibrium analysis of the associative memory with the synchronous dynamics in the presence of the self-coupling. In the synchronous dynamics, all neurons in the network evaluate their inputs and compute outputs simultaneously. It has been shown that the self-coupling could be used to control the occurrences of cycles in the associative memories [18]. The synchronous dynamics is much more stable to noise than the asynchronous one for the same loading rate, provided that neurons have a sufficiently large self-coupling [68]. The self-coupling of an appropriate magnitude can cause the suppression of spurious patterns in the associative memory.

In a standard associative memory, the stable states appear only in two distinct regions of the state space: in a narrow "recall band" and in a "spurious band" [20]. In the recall band, the stable states are strongly correlated with the stored patterns. The spurious band is centered around the state that has no "macroscopic" overlap with the chosen stored pattern. Two bands are disjointed from each other only below a certain critical value of the loading rate $\alpha_r = 0.113$ [20]. Singh [68] has found the deterioration in the recall process of the associative memory in the presence of a negative self-coupling (the basin of attraction of stored patterns decreases and the recall time increases). On the other hand, a positive self-coupling of an appropriate magnitude has a positive effect in the recall process (i.e. the enhancement in the basin of attraction of the stored patterns and the storage capacity) [68].

4.1.3 Associative memories with cross-correlation terms

An associative memory is made to store the training patterns as a result of the training process. Not only the stored patterns but also patterns generated by mixing of arbitrary stored patterns can be recalled by the model. These patterns are called mixed states, and they are not simply a side effect unnecessary for information processing.

Parga and Rolls [62] have proposed a transform-invariant recognition model based on an associative memory. Their model of associative memory uses a modified Hebbian training method that includes cross-correlation terms between various training patterns belonging to the same object. When the strength of the cross-correlation terms is small, the recalled pattern corresponds to one of the stored patterns itself. When the strength exceeds a certain limit, the recalled pattern becomes the mixed state, which is generated by all stored patterns of the same object, independent of the particular stored pattern presented to the model. Hence, the model becomes able to recognize the object by observing the drawn mixed state regardless of the coordinate transformation [62].

Elliffe et al. [13] have added a sparse encoding scheme to the model proposed by Parga and Rolls [62] since it has been thought to be used in the brain (e.g. [59]). An encoding scheme characterizes the values of elements in patterns processed by the model. An encoding scheme (given by the parameter ζ called a firing rate) is said to be sparse, when the number of elements with the value of 1 in a pattern is very small in comparison with the pattern dimension. Elliffe et al. [13] have shown how pattern sparseness interacts with the number of views of each object to give predictable recall performance. Kimoto and Okada have shown that the storage capacity of this model with the hard-limiting neuron transfer function sgn does not diverge for $\zeta \to 0$ [39]. The advantage of using the sparse encoding scheme is thus reduced by half [39].

Kimoto and Okada [39] have used a step transfer function ($\Theta(u) = 1$ for $u \geq 0$ and $\Theta(u) = 0$ otherwise) in neurons of an associative memory with cross-correlation terms. They have analyzed whether the mixed state could be recalled by the model for $\zeta \to 0$ [38], [39]. It has been found that only the stored pattern and the so-called OR mixed state (generated by the OR operation through each element of the stored patterns) can be recalled for $\zeta \to 0$ [39]. Under the condition of a fixed firing rate ζ, the storage capacity of the stored patterns decreases and that of the OR mixed states increases, as the strengths of the cross-correlation terms are increased. For $\zeta \to 0$, the storage capacity of the stored pattern and of the OR mixed state diverges, and that of the other mixed states becomes zero [39].

4.1.4 Models for hierarchical patterns in the cortex

The associative memory models in the studies of Parga and Rolls [62], Elliffe et al. [13], and Kimoto and Okada [39] store uncorrelated patterns. However, a relationship between the objects encoded in the training patterns can be represented using the correlation coefficient. Such patterns are called hierarchically correlated patterns. Sugase et al. [69] have recorded single neuron activity in the cortex of macaque monkeys while showing them various visual stimuli. Matsumoto et al. [43] [45] have analyzed how these neuron activities change over time. They have found that the activity vectors of neurons for different visual stimuli can be grouped into the clusters corresponding to rough categories in the early phase ($90 - 140$ ms) and that each cluster can expand to form subclusters corresponding to finer categories in the later phase ($140 - 190$ ms). They have also found that the activity vectors of the later phase are hierarchically

correlated and that the activity vectors of the initial phase are mixed states of the activity vectors of the later phase.

Toya et al. [83] have analyzed an associative memory model storing hierarchically correlated patterns. The model shows dynamics similar to Sugase's physiological finding [83]. Matsumoto and Okada [44] have enhanced Toya's model by using the so-called excitatory and inhibitory neurons to make it more realistic.

Physiological findings (e.g. [59]) have indicated that a sparse encoding scheme is used in the brain. Thus, Kimoto and Okada [40] have added a sparse encoding scheme to an associative memory model storing hierarchically correlated patterns. They have analyzed the coexistence properties of the stored patterns and the mixed states. Moreover, they have found that the stored patterns and one of those mixed states could become the stable state of the model. Thus, the stored patterns and the mixed states can coexist in this model. This finding should contribute to research on the cortex of monkeys [40].

4.2 Models improving dynamics

Since the storage capacity of a standard associative memory is relatively small, researchers have studied associative memory models with more complicated neuron dynamics (e.g. the non-monotonic transfer function, the chaos etc.). Various mechanisms how to enhance the storage properties of associative memory models have been proposed. Here, we review several associative memory models with various kinds of dynamics.

4.2.1 Models with non-monotonic dynamics

Morita [53] has proposed a model that enhances the performance of the associative memory by replacing the neuron hard-limiting transfer function with the so-called non-monotonic transfer function. Dynamics of a neuron with the non-monotonic transfer function can be expressed as a first order differential equation [53]. Morita has shown by the computer simulations that the storage capacity of this model is about 0.32.

Yoshizawa et al. [89] have used a piecewise linear model of the non-monotonic neuron (i.e. neuron with the transfer function $f(u) = sgn(u) - ku$). They have presented a geometrical theory to investigate the existence and stability of stable states in the model. They have proved that the storage capacity of this model is larger than the storage capacity of the standard associative memory [89]. It has been also shown that spurious patterns disappear from the model. When the model fails to recall a stored pattern correctly, the dynamics of the model shows a chaotic behaviour instead of falling into a spurious pattern [89].

4.2.2 Models with forgetting process

Hopfield has suggested training schemes where new patterns are stored in an associative memory at the expense of gradually forgetting previously stored patterns [30]. Nadal et al. [56] have explored the marginalist scheme, in which the contribution of each pattern to the weights decays exponentially with "age".

The scheme can be accomplished by an iterative process that is biologically realistic. Mézard et al. [50] have formulated and analyzed a general training rule with forgetting mechanism that incorporated both the model proposed by Hopfield and the marginalist training scheme. Okada et al. [58] have analyzed this model by the statistical neurodynamics.

Mimura et al. [52] have investigated the properties of the associative memory with forgetting process and the piecewise linear non-monotonic neurons. They have shown that the recently stored patterns remain retrievable though the memory is overloaded. It has been shown that the optimal value of forgetting rate at which the storage capacity is maximized for the given by the neuron non-monotonicity. The maximum storage capacity increases with non-monotonicity whereas the optimal forgetting rate decreases with it [52].

4.2.3 Associative memories with dynamic synapses

In most models of associative memory, the connection between neurons is modeled as a constant weight [30], [4]. In biological systems, the synaptic weight is a dynamic quantity that strongly depends on the presynaptic neural activity (e.g [1], [85]). Several studies have been focused on the performance of associative memories with the so-called dynamic synapses (e.g. [10], [60], [82], [46]).

Bibitchkov et al. [10] have found that the dynamic synapses do not change stable states of the model. However, the stability of these states is affected considerably. They have determined that the synaptic depression reduces the storage capacity of the associative memory [10]. Pantic et al. [60] have shown that the associative memory with depressing synapses is able to recall stored patterns but the model dynamics switches between stored patterns with an intermittent type of behavior. This phenomenon might reflect the flexibility of real neural systems with respect to receive and respond to novel and changing inputs [60].

In the brain, a large variety of values for the depression and recovery time constants are encountered. Torres et al. [82] have shown that only the ratio of these constants can affect the storage capacity of the associative memory with dynamic synapses. They have determined the dependance of the storage capacity on this ratio. The storage capacity is strongly reduced by the degree of the synaptic depression in an infinite number of patterns [82].

Matsumoto at al. [46] have investigated how the synaptic depression influences basins of attraction. They have found that basins of attraction are enlarged while the storage capacity does not change. Thus, the synaptic depression may incorporate the activity control mechanism necessary for the pattern recall [46].

4.2.4 Associative memories with chaos

In associative memories, the energy function is shown to decrease monotonically and one energy minimum corresponds to a single stored pattern [31]. However, if the energy function reaches a minimum corresponding to any spurious pattern, it cannot then escape. One approach how to solve this problem can be based on the chaos. Shimizu [67] has proposed a chaotic neural network model consisting

of Brownian neurons. The time evolution of the neuron can be described by the Brownian motion with a chaotic force. The model has been shown to recall the stored patterns and reversed ones systematically and very quickly. Moreover, the model is able to store and recall the number of patterns comparable to the number of neurons [67].

Itoh and Shimizu [32] have studied neurons with the chaos behaving as a simple harmonic oscillator driven by a chaotic force. Later, they have focused on an associative memory consisting of neurons with the chaos interacting via the chaotic force [33].

Itoh et al. [34] have proposed a model of mutually coupled associative memories with the chaos. If two associative memories are not coupled and one pattern is stored in each memory, the energy of each memory has only one minimum corresponding to the pattern. If they are coupled, each network can recall the pattern stored in the other network. Hence, different patterns can be stored in each network and the model is able to recall them [34]. They have also implemented the proposed model in a robot and confirmed the ability of the robot to recall optimum output in response to an input pattern [34].

4.2.5 Sequential associative memories

In a standard (auto)associative memory, the weights are symmetric and the model is not able to recall patterns sequentially (i.e. $\vec{x}^1 \rightarrow \vec{x}^2 \rightarrow \ldots \rightarrow \vec{x}^p \rightarrow \vec{x}^1$). The sequential associative memory is a model of heteroassociative memories that stores a sequence of patterns by an asymmetric modification of weight matrix. It has been shown that the storage capacity of the sequential associative memory is $\alpha_c \approx 0.27$ [12]. The storage capacity is thus larger than that of a standard associative memory ($\alpha_c \approx 0.15$ [30]).

Kawamura et al. [36] [37] have focused on the sequential associative memories with the non-monotonic transfer function of neurons. The chaotic behavior has been shown in the model [36]. Moreover, the chaos occurs only when it has some degree of frustration [36]. By means of bifurcation diagrams, it has been shown that the disappearance of the chaos is due to the temperature effect [37].

4.3 Cascade models

A standard associative memory does not work well if stored patterns are correlated [4]. It can correspond to situations where objects encoded in patterns are similar to one another. More realistic models of associative memories have to confront the presence of correlated patterns. In the following sections, we describe several models of associative memories that allow processing a special kind of correlated patterns (the so-called hierarchically correlated patterns).

4.3.1 Hierarchically correlated patterns

The model proposed by Amit et al. [4] is suited to processing the biased patterns (Section 4.1.1). Gutfreund [22] has generalized this approach to a set of the hierarchically correlated patterns. The organization of objects with well-defined

relations of similarity into a hierarchical tree seems to be natural in e.g. data classification and analysis.

Gutfreund [22] has proposed a procedure for generating hierarchical correlated patterns. In the following paragraphs, the process of generating patterns with the two-level hierarchy is described (as it has been presented by Gutfreund [22]). The two-level hierarchy can be preferred for an easy geometrical interpretation.

At the first level of the hierarchy, p_1 bipolar patterns $\vec{x}^u = (x_1^u, \ldots, x_n^u)$, $u = 1, \ldots, p_1$ are generated ($n > 0$). Each element x_i^u ($i = 1, \ldots, n$) takes the value of $+1$ or -1 independently (Equation 28). These patterns are called ancestors. The ancestors depends on a bias parameter a ($-1 < a < 1$). At the second level of the hierarchy, p_2 bipolar patterns $\vec{x}^{uv} = (x_1^{uv}, \ldots, x_n^{uv})$, $v = 1, \ldots, p_2$ are generated for each pattern $\vec{x}^u$ ($u = 1, \ldots, p_1$). The patterns $\vec{x}^{uv}$ are called descendants. Each element x_i^{uv} ($i = 1, \ldots, n$) takes the value of $+1$ or -1 with the probability

$$P(x_i^{uv}) = \frac{1}{2}(1 + x_i^u b)\delta(x_i^{uv} - 1) + \frac{1}{2}(1 - x_i^u b)\delta(x_i^{uv} + 1) \tag{30}$$

where b ($0 < b < 1$) is a correlation parameter between the patterns $\vec{x}^u$ and $\vec{x}^{uv}$. Geometrically, the patterns are grouped into clusters with large correlations between patterns within the same cluster and smaller correlations between patterns in different clusters. The descendants $\vec{x}^{uv}$ ($1 \leq v \leq p_2$) belonging to the same ancestor $\vec{x}^u$ ($1 \leq u \leq p_1$) are distributed around $\vec{x}^u$ with equal distance of $1 - b$. The distance is defined by

$$1 - \frac{1}{n}\sum_{i=1}^{n} x_i^{uv}\, x_i^u \tag{31}$$

where n denotes the dimension of patterns [28]. This situation is schematically depicted in Figure 4.

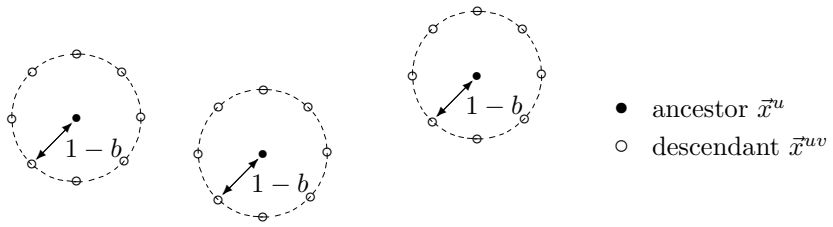


Figure 4: Hierarchically correlated patterns having uniform correlations b. The descendants $\vec{x}^{uv}$ belonging to the same ancestor $\vec{x}^u$ are distributed around $\vec{x}^u$ with equal distance of $1 - b$.

For $a = 0$, there is no correlation between patterns positioned in different clusters ($u \neq u'$) and only patterns within the same cluster are correlated. The patterns have on the average the same number or $+1$ and -1 elements [22].

The process of pattern generating can be extended to an arbitrary number of levels. The k th ($k > 0$) level of the hierarchy is given by a correlation parameter

a_k. For each of the patterns $\vec{x}^{u_1 \ldots u_{k-1}}$ from the level $k-1$, p_k patterns $\vec{x}^{u_1 \ldots u_k}$ are generated at the level k.

4.3.2 Gutfreund's model

Amit et al. [4] have proposed the model of associative memory suited to processing the biased patterns. Gutfreund [22] has generalized their approach to allow processing the hierarchically correlated patterns. He has investigated a model based on a set of associative memories - one for each level of the hierarchy. Gutfreund's research has been restricted to two-levels hierarchy [22]. Gutfreund's model is composed of two autoassociative memories; each of them consists of n ($n > 0$) neurons (Figure 5). The first autoassociative memory (AM1) and the second one (AM2) store the ancestors and their descendants, respectively.

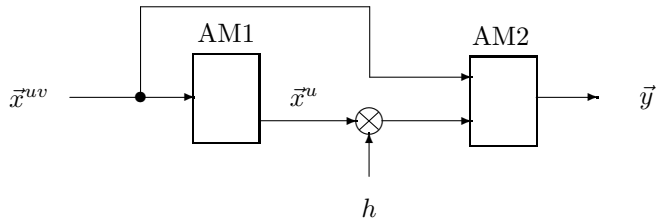


Figure 5: The structure of the Gutfreund's model is composed of two autoassociative memories AM1 and AM2. When a target pattern $\vec{x}^{uv}$ is an input pattern to the model, AM1 recalls the ancestor $\vec{x}^{u}$. The dynamics of AM2 processes the input pattern $\vec{x}^{uv}$, the recalled ancestor $\vec{x}^{u}$ and the external parameter h to give the output pattern $\vec{y}$.

The ancestors are stored in AM1 by means of the modified training rule (Equation 29). The total number of descendants to be stored in AM2 is $p = p_1 p_2$ (p_1 clusters; p_2 patterns in each cluster). The training rule for AM2 is

$$w_{ij}^{(2)} = \frac{1}{n} \sum_{u=1}^{p_1} \sum_{v=1}^{p_2} (x_i^{uv} - x_i^u b)(x_j^{uv} - x_j^u b) \tag{32}$$

for $1 \leq i, j \leq n$. The descendant $\vec{x}^{uv} = (x_1^{uv}, \ldots, x_n^{uv})$ $(1 \leq v \leq p_2)$ and the corresponding ancestor $\vec{x}^{u} = (x_1^{u}, \ldots, x_n^{u})$ $(1 \leq u \leq p_1)$ are bipolar patterns. The parameter b determines the correlation between $\vec{x}^{u}$ and $\vec{x}^{uv}$. The storage capacity of AM2 depends on the total number of descendants and not on how they are grouped into the different clusters [22]. It does not depend on the biased parameter a of ancestors, and is determined only by the parameter b, which represents the correlation among the descendants [22].

The recall of a particular descendant is achieved in the second memory AM2 with the help of the corresponding ancestor recalled by AM1. An input pattern $\vec{x}^{uv}$ is presented to the first associative memory AM1 that recalls the corresponding ancestor. Gutfreund [22] has shown that for a wide range of parameters a and b the input pattern $\vec{x}^{uv}$ approaches (usually very fast) the stable state close to $\vec{x}^{u}$ (in the Hamming distance). It is required that b is not too small, so that a

is not too large, to ensure a clear separation from the "attraction" of the other ancestors [22].

Once $\vec{x}^u$ is recalled, the recalled ancestor restricts the dynamics of AM2 and enables AM2 to recall the descendant. Hence, the recalled ancestor $\vec{x}^u$ is used to form a threshold bias $h_i = h \cdot x_i^u$ of each neuron i $(1 \leq i \leq n)$ of AM2. The parameter h represents the external parameter. If AM2 is not overloaded, there is a fairly broad range of values of h that are appropriate for a range of values of b [22]. The recall process of Gutfreund's model is shown in Figure 5.

Unfortunately, this model is very uneconomical because the same size of the memory is required to store p_1 ancestors at the first level as is used at the highest level to store $p_1 p_2 \ldots p_k$ patterns (in k level hierarchy) [22].

4.3.3 Cascade associative memories CASM1

The model proposed by Gutfreund [22] seems to be attractive in organizing the hierarchical memory structure. Unfortunately, the storage capacity α_c ($\alpha_c \approx 0.15$) is insufficient for its application. In order to overcome this problem, Hirahara et al. [26] have presented a model of cascade associative memory (called CASM1) based on the Gutfreund's model [22]. In CASM1, the second associative memory AM2 does not store descendants but the so-called difference patterns, which contain only the information on the differences between ancestors and the corresponding descendants.

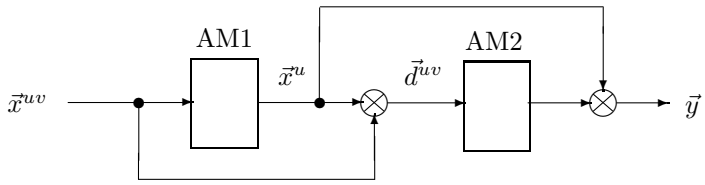


Figure 6: The structure of CASM1 is composed of two autoassociative memories: AM1 storing the ancestors $\vec{x}^u$ and AM2 storing the difference patterns $\vec{d}^{uv}$ of the recalled ancestor $\vec{x}^u$ and the descendant $\vec{x}^{uv}$.

The model of CASM1 is composed of two autoassociative memories, AM1 and AM2 (Figure 6). The first autoassociative memory AM1 stores p_1 ancestors. The second autoassociative memory AM2 stores the difference patterns. Hirahara et al. [26] assume that every descendant $\vec{x}^{uv} = (x_1^{uv}, \ldots, x_n^{uv})$ is in the basin of attraction around its ancestor $\vec{x}^u = (x_1^u, \ldots, x_n^u)$ $(1 \leq u \leq p_1$ and $1 \leq v \leq p_2)$. Based on this assumption, the difference pattern $\vec{d}^{uv} = (d_1^{uv}, \ldots, d_n^{uv})$ is defined by

$$d_i^{uv} = x_i^{uv} \cdot x_i^u = \begin{cases} 1 & \text{if } x_i^u = x_i^{uv} \\ -1 & \text{otherwise} \end{cases} \tag{33}$$

for $i = 1, \ldots, n$. Therefore, each pattern $\vec{d}^{uv}$ contains only the information on the differences between the descendant $\vec{x}^{uv}$ and the corresponding ancestor $\vec{x}^u$. Hence, the difference patterns $\vec{d}^{uv}$ $(1 \leq u \leq p_1$ and $1 \leq v \leq p_2)$ are biased patterns whose bias is equal to the correlation b. The difference patterns are stored in the weight matrix of AM2.

During the recall process of CASM1, a pattern $\vec{x}'^{uv}$ is presented to the CASM1 model. The input pattern $\vec{x}'^{uv}$ represents an input pattern to AM1 and AM1 converges to a stable state $\vec{x}^{(1)}(\vec{x}'^{uv}) = (x_1^{(1)}(\vec{x}'^{uv}), \ldots, x_n^{(1)}(\vec{x}'^{uv}))$. Recalling AM2 starts after AM1 converges to the stable state $\vec{x}^{(1)}(\vec{x}'^{uv})$. The left circle "$\otimes$" in Figure 6 produces the difference pattern $\vec{d}^{uv}$ (Equation 33) that is the input to AM2. When AM2 converges to a stable state $\vec{x}^{(2)}(\vec{x}'^{uv}) = (x_1^{(2)}(\vec{x}'^{uv}), \ldots, x_n^{(2)}(\vec{x}'^{uv}))$, the model of CASM1 also converges to the stable state $\vec{y}(\vec{x}'^{uv}) = (y_1(\vec{x}'^{uv}), \ldots, y_n(\vec{x}'^{uv}))$ given by

$$y_i(\vec{x}'^{uv}) = x_i^{(1)}(\vec{x}'^{uv}) \cdot x_i^{(2)}(\vec{x}'^{uv}) \tag{34}$$

for $i = 1, \ldots, n$. The pattern $\vec{y}(\vec{x}'^{uv})$ is produced by the right circle "$\otimes$" shown in Figure 6 and represents the output (the recalled pattern) of the CASM1 model.

The difference patterns are biased patterns and become sparser with increasing b (b is a correlation parameter between the descendant $\vec{x}^{uv}$ and the corresponding ancestor $\vec{x}^{u}$). The storage capacity of CASM1 is larger than that of the Gutfreund's model [26]. The storage capacity increases with increasing b [2]. The basins of attraction in CASM1 become larger with decreasing b. Their size does not depend on the loading rate α_r ($\alpha_r = \frac{p_1 p_2}{n}$) and is almost constant for a fixed value of b [26]. Hirahara et al. have shown that for $\alpha_r = 0.1$ CASM1 has larger basins of attraction than the standard associative memory storing uncorrelated patterns [26]. Furthermore, the basins of attraction in the standard associative memory become larger with decreasing α_r, and are larger than those in CASM1 when $\alpha_r < 0.08$ [59].

4.3.4 Cascade associative memories CASM2

To apply the model of CASM1 in practice, it has been important to enlarge the size of the basins of attraction further. To approach this problem, Hirahara et al. [27] have improved the original CASM1 and proposed the CASM2 models. In CASM1, difference patterns are uniformly stored in the same weight matrix of AM2, although hierarchically correlated patterns form a tree structure (i.e. the descendants belonging to the same ancestor are distributed in the cluster whose center is the ancestor). CASM2 has a hierarchical memory structure organized in the weight matrix of AM2. AM2 in CASM2 is a heteroassociative memory associating the descendants with their difference patterns.

The model of CASM2 consists of two associative memories: an autoassociative memory AM1 and a heteroassociative memory AM2. The structure of CASM2 is shown in Figure 7.

During the training process, the autoassociative memory AM1 stores p_1 ancestors $\vec{x}^{u}$ ($u = 1, \ldots, p_1$). AM2 in CASM2 is a heteroassociative memory that stores $p_1 \cdot p_2$ pairs $(\vec{x}^{uv}, \vec{d}^{uv})$ composed of the descendant $\vec{x}^{uv}$ and the corresponding difference pattern $\vec{d}^{uv}$ (Equation 33) for $u = 1, \ldots, p_1$ and $v = 1, \ldots, p_2$.

During the recall process of CASM2, a pattern $\vec{x}'^{uv}$ is presented to the model. The input pattern represents the input pattern to AM1 that recalls the pattern

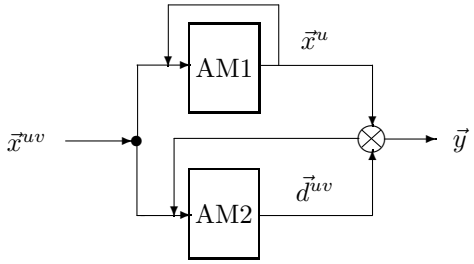


Figure 7: The structure of CASM2 is composed of two associative memories: the autoassociative memory AM1 storing the ancestors $\vec{x}^u$ and the heteroassociative memory AM2 associating every descendant $\vec{x}^{uv}$ with its difference pattern $\vec{d}^{uv}$.

$\vec{x}^{(1)}(\vec{x}'^{uv})$. Recalling in AM2 starts after AM1 converges to the stable state $\vec{x}^{(1)}(\vec{x}'^{uv})$. The input pattern $\vec{x}^{(2)}(0)$ of AM2 is identical to the input pattern $\vec{x}'^{uv}$ of CASM2 (i.e. $\vec{x}^{(2)}(0) = \vec{x}'^{uv}$). The dynamics of AM2 uses the dynamic threshold realized by the overlap between the recalled ancestor $\vec{x}^{(1)}(\vec{x}'^{uv})$ and the actual input $\vec{x}^{(2)}(t)$ (for $t > 0$). As shown in Figure 7, AM1 and AM2 in CASM2 cooperate to recall the target pattern $\vec{x}^{uv}$. In AM2, the weights are asymmetric (i.e. $w_{ij} \neq w_{ji}$), so that it cannot be regarded the recalling in AM2 as a monotonically decreasing process of the energy function [27]. Hirahara et al. [27] have shown that although the convergence of the dynamics in AM2 is not guaranteed, the input pattern can approach the descendant $\vec{x}^{uv}$ with some fluctuations, and substantially converges to $\vec{x}^{(2)}(\vec{x}^{uv}) = \vec{x}^{uv}$.

The dynamics of AM2 combines the asymmetric matrix and the recalled ancestor and works as a switching mechanism to activate the covariance matrix responsible for recalling only p_2 difference patterns for the given cluster [27]. The crosstalk generated by the other difference patterns is suppressed [27]. It has been also shown that CASM2 can work better than the associative memory with the non-monotonic dynamics proposed by Morita [53] (Section 4.2.1) for large b (i.e. the correlation parameter between the descendant and the corresponding ancestor) [27].

The storage capacity α_c of AM2 in CASM2 is larger than that in CASM1 [27]. The upper limit of the storage and recall abilities of CASM2 is bounded by that of AM1 (where the storage capacity remains 0.15) [27].

4.3.5 Cascade associative memories CASM3

In previous models of cascade associative memories, correlations between ancestors and the corresponding descendants are assumed to be uniform. In this way, the descendants are distributed around their ancestors with equal distances (Figure 4). However, this assumption can be unnatural. In real problems, the descendants can be distributed around their ancestors with various distances (Figure 8), so the previous models can become inapplicable. To overcome this, Hirahara et al. [28] have proposed the CASM3 model for storing hierarchically

correlated patterns with various correlations.

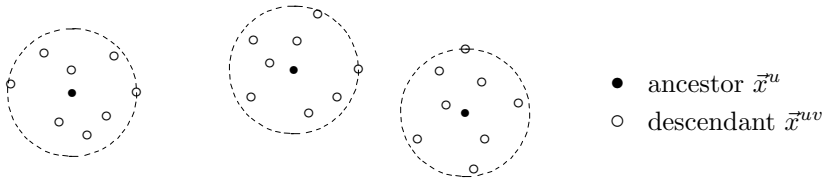


Figure 8: Hierarchically correlated patterns having various correlations.

The structure of CASM3 is identical to that of CASM2 (Figure 7). The difference between CASM2 and CASM3 is only in the training algorithm of AM2 that is the generalization of that in AM2 in CASM2. In CASM3, the recurrent heteroassociative memory AM2 stores $p_1 p_2$ pairs $(\vec{x}^{uv}, \vec{d}^{uv})$ composed of the descendant $\vec{x}^{uv} = (x_1^{uv}, \ldots, x_n^{uv})$ and the corresponding difference pattern $\vec{d}^{uv} = (d_1^{uv}, \ldots, d_n^{uv})$ for $1 \leq u \leq p_1$ and $1 \leq v \leq p_2$. The training process of AM2 is given by

$$w_{ij}^{(2)} = \frac{1}{n} \sum_{u=1}^{p_1} \sum_{v=1}^{p_2} \frac{1}{1 - b_{uv}^2} (d_i^{uv} - b_{uv})(x_j^{uv} - b_{uv} x_j^u) \tag{35}$$

for $1 \leq i, j \leq n$ where $\vec{x}^u = (x_1^u, \ldots, x_n^u)$ is the ancestor corresponding to the descendant $\vec{x}^{uv}$. The parameter b_{uv} is a correlation between $\vec{x}^{uv}$ and $\vec{x}^u$

$$b_{uv} = \frac{1}{n} \sum_{i=1}^{n} x_i^{uv} \, x_i^u \tag{36}$$

for $1 \leq i, j \leq n$.

The CASM3 model deals with hierarchically correlated patterns with various correlations. The recall process in AM2 depends on the correlation between the descendant and the corresponding ancestor. As the loading rate of AM2 increases, the stored descendants are successively destroyed in descending order of their correlations. Even if the stored descendants having small correlations are destroyed at a certain loading rate of AM2, the other descendants having large correlations can be still retrievable [28]. This is due to fact that recall of a descendant having a small correlation induces large variance in the crosstalk noise. Hirahara et al. [28] have observed that the basins of attraction depends on the range of correlations between stored descendants and their ancestors. As the correlation range with a fixed width is shifted to lower levels, the basin size becomes larger [28].

4.4 Models applicable to navigation

Models of associative memories can be also used in the area of navigation where the spatial information has to be processed. In this section, we sketch three approaches based on associative memories. The spatial information can be

processed by means of the so-called cognitive maps [87], [9]. In a different approach, the model based on an associative memory is used to emulate the spatial memory [19].

4.4.1 Hierarchical cognitive maps

The research in spatial cognition has shown that human beings build internal representations of the environments they explore (e.g. in [29], [49]). These internal representations, called cognitive maps, have many properties including distortion, fragmentation and hierarchical structure.

Voicu [87] has proposed a model based on experimental studies conducted with human participants. The model builds and uses a hierarchical cognitive map of a large environment represented by clusters of landmarks. The hierarchical cognitive map consists of three heteroassociative memories AM1, AM2 and AM3 that create associations between landmarks. The structure of the hierarchical cognitive map is shown in Figure 9.

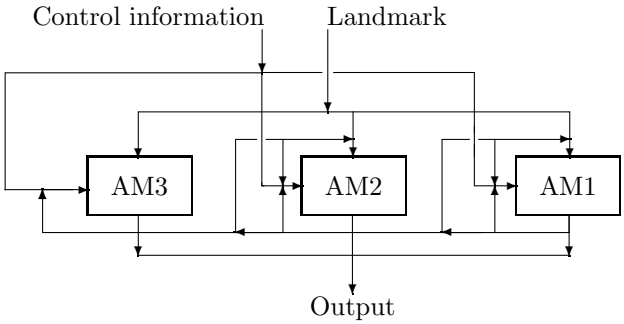


Figure 9: The structure of the hierarchical cognitive map. It contains three heteroassociative memories AM1, AM2 and AM3 that store the connections between landmarks at the three different levels

Each heteroassociative memory stores spatial information about the environment at a different resolution. The first level heteroassociative memory AM1 stores the associations between all landmarks in the environment. At the second level, AM2 stores the associations between landmarks that have the largest number of associations at the first level. The third level memory AM3 stores the associations between landmarks that have the largest number of associations at the second level.

For navigation, the model uses the goal-activation method [86]. The goal position (the landmark) in the cognitive map is activated. Then, the activity from the goal position is spread through the cognitive map, using all levels of the hierarchy, until the current position (the landmark) is reached. The activation is attenuated as it spreads from one landmark to another. The landmarks closer to the goal positions are more active than the further ones. The output of the hierarchical cognitive map is used to produce “move” action upon the environment.

4.4.2 Cognitive maps based on hippocampo-cortical systems

Banquet et al. have described a model applicable to processing the spatial information. The model is based on a cascade of increasingly complex associative processes (relevant for hippocampo-cortical systems) [9]. The model comprises three levels of associations of different types and of increasing complexity:

1. the object location level that computes the landmarks from the sensory inputs (the object location associations are combined to form the local views);

2. the subject location level that computes the place fields by means of the local views and the movement-related information;

3. the spatiotemporal level that computes place transitions from the contiguous place fields (used for the training process).

The model provides multimodal representations of inputs. The characteristics of the model is a dynamical spatiotemporal (and not just spatial) representation of the environment through the computation and encoding of the transitions. It provides a solution to the switch from spatial cognitive map to its motor implementation during the goal-oriented navigation [8].

The model has been implemented and tested on a robot moving in a real environment. There are encoded two types of coexisting maps. The first is "universal" context independent map [66]. The second type is a context dependent map computed by an associative memory based on the place field transitions. While the first type of map is spatial and stable, the second type is tempospatial and dynamic. Universal and contextual maps are modulated by the "head direction system", thus achieving external coherence aligned to the external world, as well as internal coherence by the alignment of views from multiple directions. The model provides interpretations and predictions for largely unaccounted structural differences between hippocampo-cortical systems [9].

4.4.3 The approach of Fukushima et al.

Fukushima et al. have proposed a model that emulates the spatial memory in the brain [19]. The model can store and recall the spatial information. The spatial information is processed in the form of fragmentary spatial patterns (e.g. the spatial maps). The scenery of a wide area can be recalled by a continuous chain process of recalling spatial patterns from the memory of the model.

The model is based on an autoassociative memory. The structure of the model is shown in Figure 10. A "geographic map" is divided into fragmentary spatial patterns overlapping each other. The spatial patterns are stored in the autoassociative memory.

In order to recall the scenery of a wide area, the chain process of recalling spatial patterns is repeated. During recalling, the actual (or initial) spatial pattern is presented to the model. The aim of the recall process is to recall the "neighbouring" spatial pattern in a given direction. Thus, the actual spatial pattern is "shifted" relatively to the given direction. As the spatial pattern is shifted, a vacant region appears in the "shifted" spatial pattern. The "shifted"

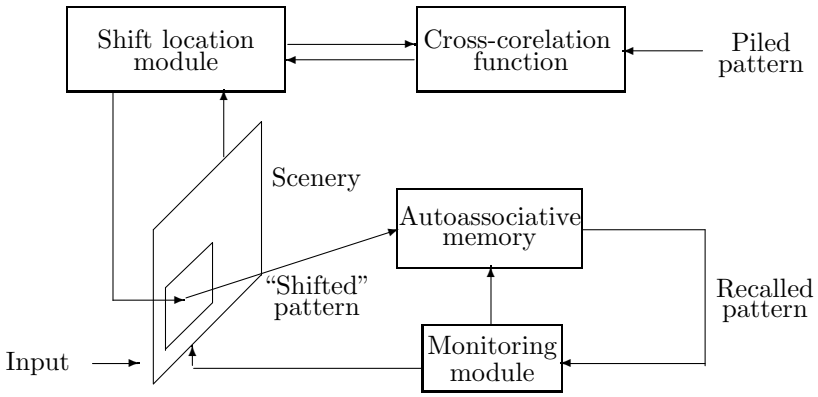


Figure 10: The structure of the model proposed by Fukushima et al. [19]

spatial pattern is partially filled by the actual spatial pattern and partially is left empty (to represent the unknown part of the "neighbouring" spatial pattern). The "shifted" spatial pattern with a vacant region is presented to the autoassociative memory of the model. The recall process of the autoassociative memory is expected to fill the vacant region of the pattern. The situation is depicted in Figure 11. In the figure, the dark filled area of the "shifted" pattern corresponds to the vacant region.

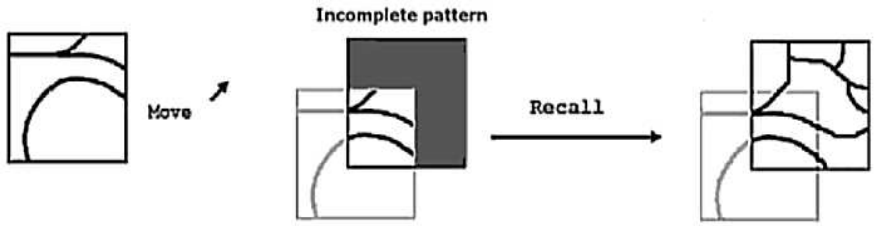


Figure 11: Principle of recalling the "neighbouring" spatial pattern in a given direction

Since an autoassociative memory by itself does not accept shifts in the location of the spatial pattern, it is necessary to shift the pattern at the same location as one of the stored spatial patterns. In the model of Fukushima et al., the problem is solved by means of the cross-correlation function between the "expected shifted" pattern and the so-called piled pattern [19].

The recall process can sometimes fail (i.e. the error in the non-vacant part of the pattern is too high). Hence, the model contains a monitoring module that detects these failures. If a failure is detected, the recalled pattern is discarded and the recall process is repeated using the "shifted" pattern in the different position [19].

5 Associative memories and spatial pattern processing

In this chapter, we describe the problem of spatial pattern processing. Associative memories can be applied to this problem. We make a short overview of various models of associative memories and their applicability to our problem. The main goal of the thesis is a design of an associative memory model applicable to processing a large number of spatial patterns. We describe the requirements imposed on the proposed model and sketch the main principles of the designed model. Various strategies to processing the spatial patterns are discussed as well.

5.1 Problem statement elaborated

In our research, processing the spatial patterns is inspired by Fukushima et al. [19]. The principle can be demonstrated by a simple example.

A traveller moves through a familiar scenery. In such a situation, he can usually only see his close surroundings. Even if the traveller is not able to draw a map of the scenery from the memory, he can often find the correct way to a destination when he actually goes there. Based on his previous knowledge about the whole scenery, he might be able to recall the scenery (or the geographical features) that he cannot see yet but will see soon as he moves. The newly recalled scenery in his mind can trigger other associations and he is able to recall another part of the scenery further ahead of him. Thus, he can in principle imagine the scenery of a wide area by a chain of the iterative recall processes. The situation is depicted in Figure 12.

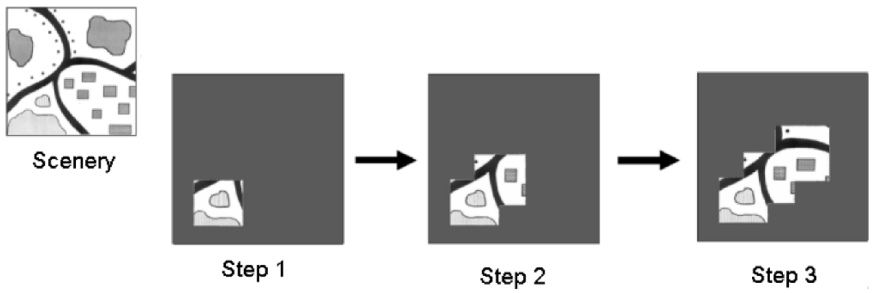


Figure 12: The principle of recalling the scenery of a wide area by a chain of the iterative recall processes

This principle leads to the idea that the human brain stores fragments of sceneries without any further information representing their exact location [19]. Fragments of sceneries can be represented in the form of spatial patterns. For storing and recall of spatial patterns, associative memory models can be applied.

When associative memories are used to help one to move through an environment, the associative memory has to be able to recall also the so-called incomplete patterns. One part of the incomplete pattern is known (e.g. the scenery seen from a given place) and the rest of the pattern is unknown (e.g. the scenery not seen yet). The incomplete pattern is presented to an associative memory to be recalled. After recall, the unknown part of the pattern is expected to be filled according to spatial patterns stored in the associative memory.

Our research is focused on models of associative memories with an emphasis to store and recall of a large number of spatial patterns. Applying the associative memory models to processing the spatial patterns, robust recall of (possibly incomplete) spatial patterns is required.

5.2 Goals elaborated

In Chapter 3, a standard associative memory and its properties have been discussed. The standard associative memory is able to reliably recall patterns as long as the number of stored patterns is relatively small ($0.15n$ where n is the dimension of stored patterns) and stored patterns are almost orthogonal (e.g. [30]). Storing correlated patterns can cause serious problems and previously stored patterns can even become lost because the crosstalk does not average to zero [4].

To overcome these limitations, many researchers have extended the standard associative memory model to improve its abilities. In chapter 4, several related models of associative memories proposed by various researchers have been described. The discussed approaches have been aimed at:

- **Models modifying the training rule**
  Models involve a much more complicated dependance of weights on the stored patterns. The standard associative memory model has been extended to allow storing the biased patterns [4]. The storage capacity of the model depends on the bias parameter. The model is suitable for processing patterns with the small bias parameter [4]. As spatial patterns are expected to be similar, this model of an associative memory is not very applicable to processing larger amounts of spatial patterns.

  In a standard associative memory, the diagonal elements of a weight matrix (the self-coupling) are set to zero. It has been shown that a positive self-coupling of an appropriate magnitude can improve the storage capacity of the model [68]. Unfortunately, the appropriate value of the self-coupling depends on data and it does not have to be easily determined. It reduces applying the model in practice.

  Associative memory models can also include cross-correlation terms. Patterns stored in these models (e.g. [62], [13], [39]) are assumed to be mutually orthogonal. Since such an assumption can be very limited, the models have been also extended to process a special type of correlated patterns (e.g. [83], [40]). A relationship between patterns is represented using correlation coefficients. Unfortunately, the current research in this area is focused mainly on theoretical analyses of coexistence properties between

stored patterns and the so-called mixed states. These models seem to be interesting but they are not very applicable in practice as spatial patterns do not usually follow the desired probability distributions.

- **Models improving the network dynamics**
  Since the storage capacity of a standard associative memory is relatively small, various mechanisms have been adopted to the associative memory dynamics. The performance of the standard associative memory has been improved by replacing the neuron hard-limiting transfer function with the non-monotonic or piecewise linear transfer function (e.g. [53], [89]). In such models, the storage capacity is larger than that of the standard model. Even the storage capacity is increased, it is still insufficient for the problems where huge amounts of patterns need to be stored.

  When the standard associative memory is overloaded, stored patterns become "forgotten" [30]. In order to avoid such a memory deterioration, forgetting mechanisms have been introduced (e.g. [50], [30], [52]). Even the number of stored patterns is high, the model with the forgetting mechanism is still able to recall the recently stored patterns. Such an ability seems to be useful in the problems where larger numbers of patterns need to be handled. Unfortunately, the maximum number of retrievable patterns is not very high [52]. Thus, it reduces applying these models in processing the spatial patterns in which all stored spatial patterns are important and have to be retrievable.

  Associative memories can use dynamic synapses. Depressing synapses have a negative effect on the storage capacity of the model [10]. The storage capacity is reduced by the degree of the synaptic depression. The models have been investigated with respect to model biological processes in real neurons. The models are not very applicable to processing the spatial patterns because the storage ability is lower than that of the standard model. Moreover, the model performance is affected by a parameter of the synaptic depression.

  Associative memories with chaos (e.g. [67], [32], [33]) allow storing and recall the number of patterns comparable to the number of neurons. The models are also able to recall stored patterns very quickly. Such abilities are desirable in handling the spatial patterns. Drawbacks of these models are connected with relatively complicated dynamics and many parameters that it is necessary to set.

- **Cascade associative memories**
  The standard associative memory has been extended to allow storing and recalling of special kinds of correlated patterns. Cascade associative memories are a combination of several associative memories (the researchers' treatment has been restricted to two associative memories).

  Cascade associative memories retain the simplicity of the training process in the standard associative memory. The recall of a pattern is achieved in the second associative memory and is assisted by the recall of the preceding associative memory.

The Gutfreund's model [22] is very attractive in terms of organizing the model structure. Since the storage capacity of the model is insufficient, the model is not suitable for processing larger amounts of patterns. To increase the storage capacity, the CASM1 model [26] has been presented. Although the structure of the CASM1 model is similar to the Gutfreund's model [22], the CASM1 model does not have a parameter on which the storage capacity depends. Later, the CASM2 model [27] has been developed to enlarge the basins of attraction.

In the CASM1 and CASM2 models, the uniform correlation between the stored patterns is assumed. It reduces the possibility of applying the models to processing the spatial patterns, as the similarity between spatial patterns can vary. The CASM3 model [28] is an extension of the CASM2 model with respect to allow processing patterns with various correlations. It has been shown that the CASM3 model has the same storage capacity as the CASM2 model does.

The cascade associative memories seem to be very attractive to processing the spatial patterns, as they are able to deal with correlated patterns. The disadvantage of the presented models is their insufficient storage capacity in cases where larger amounts of spatial patterns need to be stored.

All models discussed above have been proposed by other researchers. Although the storage capacity of several discussed models is higher than that of the standard associative memory, it is still insufficient for handling the large amounts of spatial patterns.

Moreover, most of the models discussed above assume that the stored patterns are orthogonal or very close to orthogonal (Definition 2.3). Several models have been developed to deal with special types of correlated patterns. In practice, spatial patterns (e.g. the spatial maps) tend to be similar to each other. Such spatial patterns are correlated rather than to be orthogonal. Thus, most of traditional associative memories are not applicable to the problems where spatial patterns mutually correlated has to be processed.

The main goal of the thesis is a design, implementation and testing of an associative memory model applicable to processing large numbers of spatial patterns. Our requirements for the model refer in particular to:

1. **Storage of a huge amount of patterns**
   Associative memories can be used to store spatial patterns. The number of spatial patterns to be stored depends on a concrete application. In practice, a large number of spatial patterns needs to be stored in the model. Thus, the first requirement represents the ability of the model to store larger amounts of spatial patterns.

2. **Storage of correlated patterns**
   Spatial patterns can represent the fragmentary spatial maps of the scenery. Spatial patterns cannot be assumed mutually orthogonal in a general case as they often are very similar. Storing correlated patterns limits the capacity of most traditional models. Thus, our second requirement is connected

with the ability of the model to store and recall correlated patterns as well as orthogonal ones.

3. **Reliability of the storage process**
   The third requirement is adopted to the reliability of the storage process. When all patterns are stored in the model (i.e. the storage process is finished), the model should be able to recall all stored patterns without any "error". It means that the recalled pattern should coincide with the pattern originally stored in the model. The storage process is reliable if the ratio of patterns recalled without any "error" is very high.

4. **Reliability of the recall process**
   This requirement refers to recall of incomplete patterns. Incomplete patterns represent patterns unknown in some part of their surface (Figure 11). When the associative memory is used to help one to move through an environment, incomplete patterns can represent e.g. the "foreseen" scenery. Incomplete patterns should be "completed" during the recall process. It means that the recalled patterns should coincide with the patterns originally stored in the model. The reliability of the recall process is very important ability of the model in spatial information processing (especially in applications focused on the navigation problems).

In order to address the requirements imposed on a model applicable to processing larger amounts of spatial patterns, we propose the so-called hierarchical associative memory model. Our model is inspired by the concept of cascade associative memories. Our goal is to use the concept more generally by allowing an arbitrary number of layers with more associative memories grouped in each layer. The overview of our hierarchical associative memory is depicted in Figure 13.

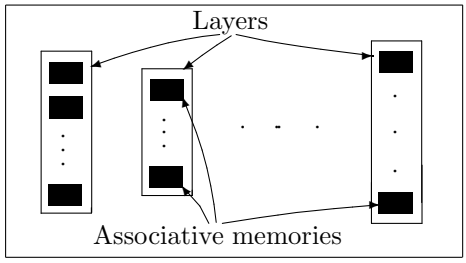


Figure 13: The overview of the developed hierarchical associative memory model

To avoid the capacity limitation of traditional associative memories, a suitable strategy for the storage process of the hierarchical associative memory is proposed. Associative memories are dynamically added to our hierarchical associative memory model according to the incoming spatial patterns to be stored in it. Thus, the structure of the hierarchical associative memory depends on the processing data and their mutual correlations. As the associative memories are

automatically added to the hierarchical associative memory, there is no need to state the initial number of associative memories before the storage process starts.

In practice, a robust recall of (possibly incomplete) spatial patterns is required. The experimental simulations carried out with the proposed hierarchical associative memory model are focused on

- the ability to process patterns stored in the model,
- the ability to process incomplete patterns,
- the memory complexity (i.e. the number of associative memories in the model).

The obtained experimental results are analyzed and compared with the standard associative memory and the cascade associative memory. In the memory complexity, the experimental results are compared with the derived theoretical results with respect to the number of associative memories in the hierarchical associative memory model.

5.3 Strategies for spatial pattern processing

Neurophysiological and psychological experiments have suggested that the spatial information about the external world is represented in the brain in the form of topologically ordered spatial patterns (e.g. [21]). Experiments have also suggested that the human brain stores spatial patterns of the scenery without any further information representing the exact location of the pattern in the scenery [19].

When human beings recall the image of the scenery around them, the recalled scenery is represented as a spatial activity pattern of neurons in some area of the brain. The spatial activity pattern preserves the topology of the geometrical features of the external world. The image recalled at one time is limited in size because of the limited number of neurons. Furthermore, the spatial information stored at one time in the brain is limited because of the limited size of the human visual field. However, human beings are able to combine two stored adjacent spatial patterns and create one unified image in the mind [19].

Now, we propose several approaches of how to store and recall spatial patterns with respect to allow processing larger amounts of (correlated) spatial patterns. In addition, it is required the robust recall of presented (possibly incomplete) patterns. The first approach is based on the simple idea to store the whole image of the scenery. Other approaches use associative memories to process the spatial patterns. At the end of the chapter, we sketch the problem of preprocessing the spatial patterns as a way how to further improve the abilities of the associative memory models.

5.3.1 Storing the whole image of the scenery

A simple approach to processing the spatial information could be to memorize a whole image of the scenery (instead of storing the fragmentary spatial patterns). The scheme is depicted in Figure 14.

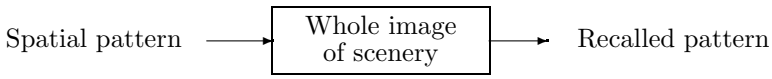


Figure 14: The spatial information recall based on storing a whole image of the scenery

This approach depends on the size of the whole scenery image. In case of large sceneries, this kind of memorizing the scenery would be connected with high memory costs. The recall process would probably be time-consuming, too.

The main disadvantage of this approach is the inability to recall "damaged" sceneries (i.e. to recall images that differ from the stored information). The "damaged" image does not coincide with any part of the stored scenery image or it can coincide with a different part of the image. This greatly reduces the possibility of applying this approach in practice. Therefore, it seems to be better to use associative memories for storing fragmentary spatial patterns.

5.3.2 Autoassociative memory approach

A standard autoassociative memory (of the Hopfield type) is one of the associative memory models that could be applied to processing spatial patterns. The autoassociative memory is able to recall spatial patterns (even the "damaged" patterns or the patterns incomplete in some parts). The scheme of this approach is depicted in Figure 15.

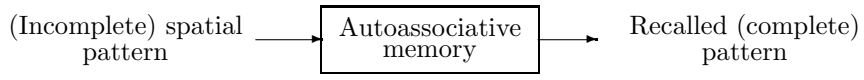


Figure 15: The spatial pattern recall based on the autoassociative memory

Unfortunately, the performance of the standard autoassociative memory depends on the number of patterns stored in the memory and is very sensitive to mutual correlations of the stored patterns (Chapter 3). As it is necessary to process large numbers of spatial patterns (mostly correlated), these limitations greatly reduce the possibility of applying the standard autoassociative memory in practice.

The standard autoassociative memory requires higher memory costs for storing spatial patterns in comparison with memorizing the whole image of a scenery. The storage capacity of the autoassociative memory corresponds to 0.15 when the stored spatial patterns are (almost) orthogonal [30]. The memory costs $MemCost$ for the autoassociative memory (AutoAM) with n ($n > 0$) neurons is given by the formula

$$MemCost(AutoAM) = \frac{n(n-1)}{2}. \tag{37}$$

The formula corresponds to the fact that the weight matrix of the autoassociative memory is $n \times n$ symmetric matrix with zero diagonal (Chapter 3.3).

The number of neurons n is given by the dimension of the stored spatial patterns. The memory costs corresponds to the number of elements above the main diagonal in the weight matrix of an autoassociative memory.

In memorizing the whole image of a scenery, the memory costs $MemCost$ required to store $0.15n$ spatial patterns of the dimension n would be

$$MemCost(WholeSceneryImage) = 0.15n^2. \tag{38}$$

Thus, the ratio ρ between the memory costs required by a "conventional" storing method and the autoassociative memory to store $0.15n$ spatial patterns of the dimension n approaches

$$\rho = \frac{MemCost(WholeSceneryImage)}{MemCost(AutoAM)} \sim 0.3 \tag{39}$$

for large-dimensional patterns.

It would be possible to increase the storage capacity of the autoassociative memory by increasing the dimensionality of spatial patterns to be stored in it (having now proportionally more neurons). But at the same time, the number of network's weights would raise quadratically with the number of network's neurons. For this reason, the recall process would require higher computational costs as well. Moreover, real spatial patterns (e.g. the spatial maps) tend to be correlated rather than nearly orthogonal. This also limits the capacity of the autoassociative memory in spatial pattern processing.

5.3.3 Hierarchical associative memory approach

A standard autoassociative memory has the robust recall ability when the number of stored patterns is relatively small and patterns are almost orthogonal to each other (Chapter 3.5). It limits the possibility of applying the standard autoassociative memory in the problems where larger amounts of spatial patterns need to be processed.

To avoid (at least to a certain extent) these limitations, our research has been focused on models composed of several (auto)associative memories. The inspiration for our research comes from cascade associative memories that consist in principle of two associative memories (e.g. [26], [27], [28]). Our main goal has been to use the idea of cascade associative memories more efficiently.

We have develop the so-called hierarchical associative memories (the HAM model) that consist of (auto)associative memories grouped into several layers. Since our model has been developed with a stressed necessity to process huge amounts of spatial patterns, (auto)associative memories are added dynamically to the proposed model according to the structure of storing spatial patterns. Using this strategy, hierarchical associative memories can avoid the capacity limitation of traditional associative memories.

The scheme of the spatial pattern recall by means of our hierarchical associative memories is similar to the spatial pattern recall using by the standard (auto)associative memory. The scheme is depicted in Figure 16.

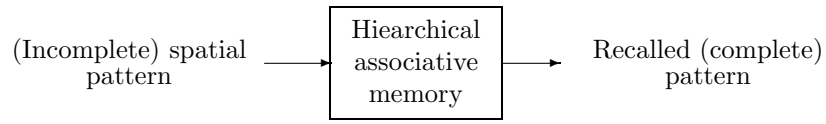


Figure 16: The spatial pattern recall based on our hierarchical associative memories

5.3.4 Heteroassociative memory approach

Another approach to processing spatial patterns could be based on heteroassociative memories. When the memory is used to help one to move in a scenery, a heteroassociative memory could be used to recall the "foreseen" spatial pattern in the direction of the "next" movement. It means that the "foreseen" spatial pattern is recalled based on the available spatial information (e.g. the actual scenery). The input pattern of the heteroassociative memory is the spatial pattern with the available information (i.e. the actual spatial pattern). The output of the memory represents the "foreseen" spatial pattern (Figure 17). In such a case, no incomplete pattern recall is needed.

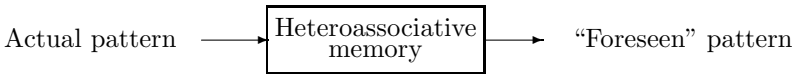


Figure 17: The spatial pattern recall based on the heteroassociative memory

During the storage process, training patterns are in the form of ordered pairs: < input pattern – output "foreseen" pattern>. Training pairs are stored in the weight matrix of the heteroassociative memory. During recall, the actual spatial pattern is an input of the heteroassociative memory and the model should recall the corresponding "foreseen" spatial pattern.

This approach has several disadvantages. The weight matrix of the heteroassociative memory with a simple feedback is in general asymmetric, so that the convergence of the recall process cannot be guaranteed [64]. This problem could be solved when the heteroassociative memory with more complicated feedback is used (e.g. the bidirectional associative memories). Another disadvantage is the necessity to define the unique output "foreseen" pattern for every input pattern. The "foreseen" spatial pattern is an output of the heteroassociative memory after presenting the input spatial pattern. In reality, one spatial pattern usually has several "foreseen" spatial patterns according to all possible move directions. It reflects the fact that from one place it is possible to move to various directions. Thus, there are several training pairs: <input pattern – output "foreseen" pattern> where the input patterns are identical. Storing training pairs with the same input pattern greatly reduces the recall ability of the heteroassociative memory. There is no direct way to recall the desired "foreseen" pattern after presenting an input spatial pattern.

This problem could be solved by processing the additional information about

the "foreseen" spatial pattern to be recalled. It could contain the information about e.g. the move direction. In this case, the information encoding the move direction is added into the input pattern. The input pattern is composed of two parts: the actual spatial pattern and the move direction. The heteroassociative memory is extended with several new neurons processing the move direction information. During recall, the actual spatial pattern is combined with the encoded move direction to produce the corresponding "foreseen" spatial pattern. The scheme is depicted in Figure 18.

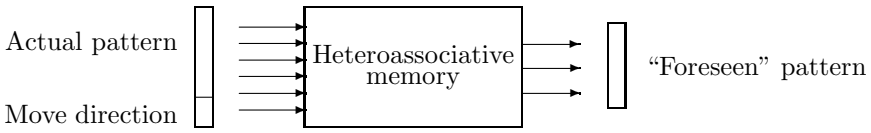


Figure 18: The spatial pattern recall based on the heteroassociative memory with additional information encoding the move direction

5.3.5 Preprocessing of spatial patterns

Now, we can imagine another situation. A person visits a museum where he was a long time ago. Since his last visit the museum arrangement has been changed - e.g. some of the exhibits have been moved to different places, several showcases have been rotated or enlarged etc. Hence, the visitor's previous memories about the museum do not fully correspond with the reality because of rotation, shift or change in scale of the objects contained in the scenery.

The situation is shown in Figure 19. The previous room scheme (a) has been changed to the actual room scheme (b). During the room re-arrangement, the object D has been enlarged and the objects A and C have been rotated. The "foreseen" scenery to be recalled by the visitor's "old memories" appears to be deformed. Even though the human eye (brain) recognizes this "deformed" scenery as the original one.

Figure 19: (a) The previous room scheme (b) The actual room scheme

Presenting the "deformed" scenery pattern to an associative memory can lead to serious problems during the recall. For example, the known part of the "deformed" pattern does not coincide with any part of the corresponding stored pattern. The reason is that the "deformed" patterns represent completely different input patterns for an associative memory than those stored in it "without

any deformation". The performance of associative memories is very sensitive to changes in scale, shift and rotation of objects encoded in the patterns.

In order to eliminate these drawbacks, the spatial patterns can be preprocessed before they are presented to an associative memory. The preprocessing can be based also on artificial neural networks (namely the Kohonen networks). In our paper [54], we have proposed the so-called Feature Weighting Kohonen Network to ensure more invariance of an associative memory with respect to small deformations.

During training, our Feature Weighting Kohonen Network should "approximate the overall structure" and "preserve the topographic ordering" encoded in the presented pattern. Additionally, the network is forced to distribute its neurons in such a way that the important "features" attract more neurons than the other ones (e.g. the important features can correspond to object borders).

During such kind of preprocessing, the occurrence of object important "features" is enhanced (e.g. the object borders are widened). The following associative memory recall of the spatial patterns with the enhanced important "features" is expected to be more robust against the above-mentioned deformations. Hence, this kind of preprocessing could ensure a recall invariant (at least partially) to minor deformations of spatial patterns. The more detailed description of the network can be found in [54]. The problem of spatial pattern preprocessing can be covered by our future work.

6 Hierarchical associative memories (HAM)

The performance of a standard associative memory is limited by the number of patterns that can be stored in it and the fact that the storing patterns have to be orthogonal. The standard associative memory cannot deal with correlated patterns as the perturbation noise generated during recall by other stored patterns does not average to zero [4]. These problems greatly reduce the possibility of applying the standard associative memory in the field of processing large amounts of (correlated) spatial patterns.

To avoid (at least to a certain extent) these limitations, we have developed the so-called Hierarchical Associative Memories (the HAM model). The hierarchical associative memories represent a generalization of cascade associative memories proposed by Hirahara et al. [26]. Cascade associative memories comprise in principle two associative memories. Thus, the number of patterns to be stored in it is limited. Since our research is focused on processing larger amounts of (correlated) patterns, our goal is to use the idea of cascade associative memories more efficiently. The hierarchical associative memories are composed of more associative memories grouped into several layers. A suitable strategy applied during training and recall can lead to a more appropriate processing of huge amounts of spatial data often containing mutually correlated patterns.

6.1 Structure of the HAM model

Definition 6.1 *A hierarchical associative memory $\mathcal{H}$ with L layers ($L > 0$) is an ordered $L-$tuple $\mathcal{H} = (M_1, \ldots, M_L)$ where $M_1, \ldots, M_L$ are finite non-empty sets of autoassociative memories; each one having the same number of neurons n ($n > 0$).*

The sets $M_1, \ldots, M_L$ are called layers of the network $\mathcal{H}$. The autoassociative memory in the layer M_k ($1 \leq k \leq L$) is called the local associative memory. $\mid M_k \mid$ denotes the number of local associative memories of the layer M_k ($1 \leq k \leq L$). The structure of the hierarchical associative memory is depicted in Figure 20.

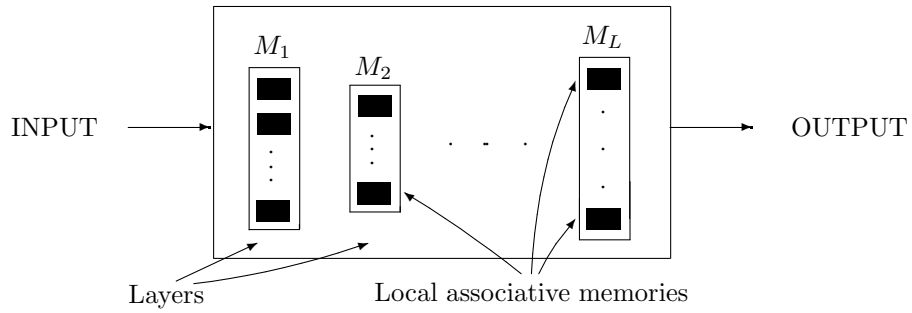


Figure 20: The structure of the hierarchical associative memory with L layers $M_1, \ldots, M_L$

Definition 6.2 *Let $\mathcal{H} = (M_1, \ldots, M_L)$ be a hierarchical associative memory with L ($L > 0$) layers. A training tuple $\mathcal{T}$ of $\mathcal{H}$ is an ordered $L-$tuple $\mathcal{T}$*

$$\mathcal{T} = (T_1, \ldots, T_L)$$

where T_k ($1 \leq k \leq L$) is a finite non-empty set of training patterns ${}^k\vec{x}^1, \ldots, {}^k\vec{x}^{m_k}$ ($m_k \in \mathbb{N}$) for the layer M_k

$$T_k = \{{}^k\vec{x}^1, \ldots, {}^k\vec{x}^{m_k} \mid {}^k\vec{x}^p \in \{+1, -1\}^n,\ p = 1, \ldots, m_k\}.$$

The symbol m_k denotes the number of training patterns for the layer M_k and $\mathbb{N}$ denotes the set of all natural numbers.

We have proposed two models of hierarchical associative memories. Our original model (called the HAM1 model) stores the patterns in "any suitable" local associative memory. The HAM1 model does not use the information recalled by the "previous" layer to find the appropriate local associative memory at the layer. Later, we have extended our original HAM1 model and presented the HAM2 model. In the HAM2 model, the local associative memories form a tree structure. The HAM2 model groups the patterns hierarchically according to the information recalled by the "previous" layer. Now, we describe in more detail the HAM1 and HAM2 models.

6.2 The HAM1 model

Our original model HAM1 has been developed with an emphasis on processing large amounts of (correlated) data. The HAM1 model consist of several local associative memories grouped into layers. The structure of the HAM1 model is shown in Figure 20.

First, we describe the training process of the HAM1 model. Let $\mathcal{T} = (T_1, \ldots, T_L)$ be a training tuple of the HAM1 model $\mathcal{H}$. Training patterns from the training tuple $\mathcal{T}$ are to be stored in the model separately for different layers. It means that training patterns from the set T_k ($1 \leq k \leq L$) are stored in the local associative memories of the layer M_k. The training process of the whole network $\mathcal{H}$ consists in training each layer M_k ($k = 1, \ldots, L$) separately. We call it layer training. In the HAM1 model, training the layer M_k ($1 \leq k \leq L$) is independent on training other layers. Hence, all layer training processes can be done in parallel.

Let M_k ($1 \leq k \leq L$) be a layer of the network $\mathcal{H}$. Now, we describe the layer training process of the layer M_k in more detail. During the training the layer M_k, the training patterns from the set T_k are presented to the layer M_k sequentially. Each training pattern $\vec{x}$ ($\vec{x} \in T_k$) is stored in such a local associative memory of the layer M_k where the pattern $\vec{x}$ (or the "noisy" version of the pattern $\vec{x}$ [1]) is recalled correctly. If more local associative memories fulfill this condition, the patterns remains stored in one of them. If there is no "suitable" local associative memory in the layer M_k, a new local associative memory is created and added to the layer M_k. In this case, the training pattern $\vec{x}$ is stored in the newly

[1] a pattern, in which a certain number of randomly selected elements change their value

created local associative memory. We describe the so-called DLT1-algorithm (the dynamical layer training algorithm) in the formal way.

The DLT1-algorithm (for the layer M_k):

Step 1: The weight matrices of all local associative memories of the layer M_k are set to zero.

Step 2: A training pattern $\vec{x}$ from T_k is presented to the layer M_k.

Step 3: The pattern $\vec{x}$ is stored in all local associative memories of the layer M_k (e.g. according to the Hebbian training rule - Equation 10).

Step 4: The pattern $\vec{x}$ (or its "noisy" version) is recalled by all local associative memories of M_k. We denote $\vec{y}^i$ the output of the i-th local associative memory of M_k $(i = 1, \ldots, \mid M_k \mid)$.

Step 5: The Hamming distance d^i of the training pattern $\vec{x}$ and the recalled output $\vec{y}^i$ $(i = 1, \ldots, \mid M_k \mid)$ is computed

$$d^i = \text{Hamm}(\vec{x}, \vec{y}^i)$$

for all local associative memories of the layer M_k.

Step 6: The minimum Hamming distance d^{min} is found

$$d^{min} = \min_{i=1,\ldots,|M_k|} \{d^i\}.$$

The parameter min is set to the index of the local associative memory of M_k with satisfying d^{min}. If there exist more than one local associative memories of M_k with the same minimum Hamming distance d^{min}, the parameter min is set to the lowest index of the local associative memory with satisfying d^{min}. Thus, the index min characterizes only one local associative memory of M_k.

Step 7: The pattern $\vec{x}$ is unlearnt from every local associative memory of M_k that fulfills the condition

$$d^i \neq 0 \;\text{ or }\; i \neq min$$

where i is the index of the local associative memory in the layer M_k. The process of unlearning the pattern $\vec{x} = (x_1, \ldots, x_n)$ is the operation inversed to the pattern training. In a case of Hebbian training rule, it can be written in the form

$$w^i_{rs} = \begin{cases} w^i_{rs} - x_r x_s & \text{for } r \neq s \\ \\ w^i_{rs} & \text{otherwise} \end{cases} \tag{40}$$

where $W^i = (w^i_{rs})$ is the weight matrix of the i-th local associative memory of M_k, $1 \leq i \leq \mid M_k \mid$ and $1 \leq r, s \leq n$.

Step 8: If the pattern $\vec{x}$ is unlearnt from all local associative memories of the layer M_k, a new local associative memory is created and added to the layer M_k. The pattern $\vec{x}$ is stored in this newly created local associative memory (e.g. according to the Hebbian training rule).

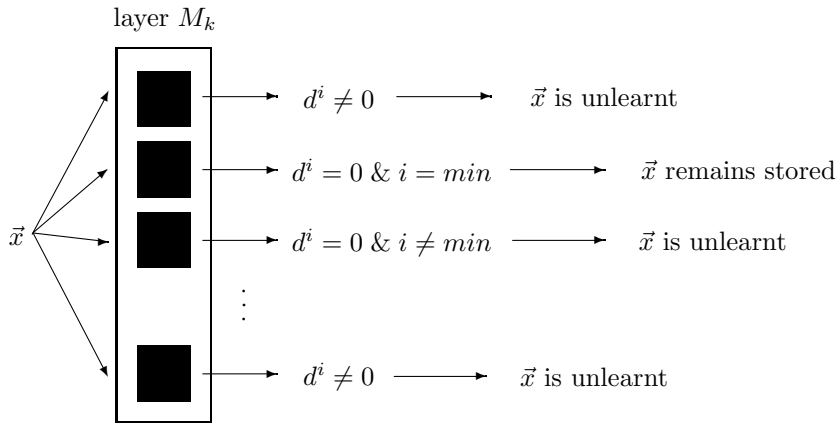


Figure 21: The scheme of the layer training algorithm in the HAM1 model

Step 9: If there is any other training pattern in T_k, **Step 2**.

The idea of the DLT1-algorithm is depicted in Figure 21. In the training algorithm, we use a simple heuristics to choose the local associative memory for storing a pattern. The heuristics is quick, simple and easy to implement, but it is not optimal. A pattern remains stored in such a local associative memory where the pattern or its "noisy version" is recalled correctly. However, using this training algorithm we cannot predict anything about recalling previously stored patterns. After storing a pattern, some of the previously stored patterns can be recalled incorrectly or even become lost.

In the model, there is no need to state the number of the local associative memories before the training process starts because the local memories are automatically added to the corresponding layers during training. The training process usually starts with one local associative memory in every layer. Other local associative memories are added to the HAM1 model during the training process according to the incoming patterns. Hence, the number of local associative memories depends on the incoming patterns.

During recall, an input pattern $\vec{x}$ is presented to the HAM1 model. The input pattern $\vec{x}$ represents an input for the first layer M_1 (i.e. ${}^1\vec{x} = \vec{x}$). In the following time steps k ($1 \leq k \leq L$), the layer M_k produces the output ${}^k\vec{y}$. The output ${}^k\vec{y}$ can be combined with the output of the "previous" layer M_{k-1} (e.g. using the difference pattern calculation). The pattern is used as the input ${}^{k+1}\vec{x}$ of the "next" layer M_{k+1}. The output ${}^L\vec{y}$ of the "last" layer M_L (combined with the output of the layer M_{L-1}) represents the output $\vec{y}$ of the whole HAM1 model.

The recall process of the HAM1 model is depicted in Figure 22. The function of the circle "$\otimes$" in Figure 22 is to produce the input to the "next" layer according to the output and the "previous-layer" output. In our implementation, difference pattern calculation is used; other function can be used as well.

Now, we focus on the recall process in a layer in more detail. The recall of

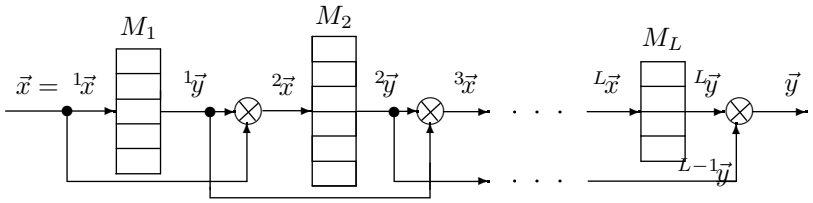


Figure 22: The recall process in the HAM1 model with L layers $M_1, \ldots, M_L$

the layer M_k $(1 \leq k \leq L)$ starts when the input pattern ${}^k\vec{x}$ is presented to the layer M_k. The input pattern ${}^k\vec{x}$ is propagated to all local associative memories of the layer M_k. Each of the local associative memories recalls the corresponding output ${}^k\vec{y}^i$ $(i = 1, \ldots, \mid M_k \mid)$. The symbol i denotes the index of the local associative memory in the layer M_k. The output ${}^k\vec{y}$ of the layer M_k is such a recalled output that is "the most similar" to the presented input pattern ${}^k\vec{x}$. We measure the similarity between patterns using the Hamming distance, but other metrics may be considered too. We describe the so-called LR1-algorithm (the layer recall algorithm) for recall in the layer of the HAM1 model in formal way.

The LR1-algorithm (for the layer M_k):

Step 1: The input pattern ${}^k\vec{x}$ is presented to all local associative memories in the layer M_k.

Step 2: The pattern ${}^k\vec{x}$ is recalled by all local associative memories in M_k. We denote ${}^k\vec{y}^i$ output of the i-th local associative memory in M_k $(i = 1, \ldots, \mid M_k \mid)$.

Step 3: The Hamming distance d^i of the input pattern ${}^k\vec{x}$ and the recalled output ${}^k\vec{y}^i$ is computed

$$d^i = \text{Hamm}({}^k\vec{x},\ {}^k\vec{y}^i)$$

for every local associative memory of M_k $(i = 1, \ldots, \mid M_k \mid)$.

Step 4: The minimum Hamming distance d^{min} is found

$$d^{min} = \min_{i=1,\ldots,|M_k|} \{d^i\}.$$

The parameter min is set to the index of the local associative memory of M_k with satisfying d^{min}. If there exist more than one local associative memories of M_k with the same minimum Hamming distance d^{min}, the parameter min is set to the lowest index of the local associative memory with satisfying d^{min}.

Step 5: The output ${}^k\vec{y}$ of the layer M_k is the output ${}^k\vec{y}^{min}$

$${}^k\vec{y} = {}^k\vec{y}^{min}.$$

The scheme of the LR1-algorithm in the layer M_k is shown in Figure 23.

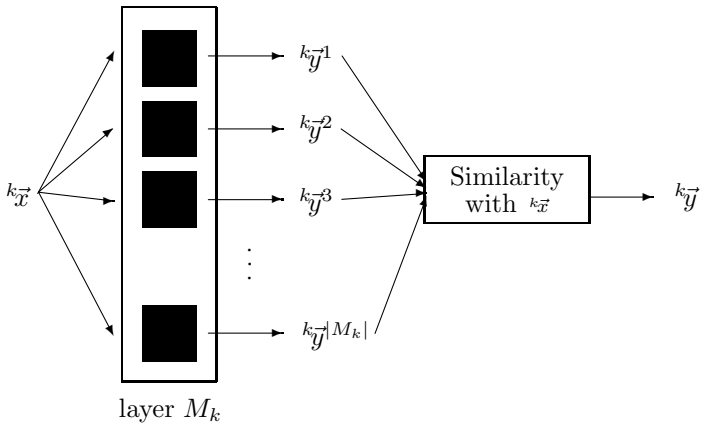


Figure 23: The scheme of the recall process in the layer M_k of the HAM1 model

6.3 The HAM2 model

A basic concept of the HAM2 model is similar to the HAM1 model. Our original HAM1 model does not use the information recalled by the "previous" layer to find an appropriate local associative memory of the layer. The local associative memories of the HAM2 model form a tree structure. The local associative memories of every layer are grouped into disjoint subsets. Every subset of the layer is assigned to one local associative memory of the "previous" layer. Thus, the number of subsets in the layer corresponds to the number of local associative memories in the "previous" layer. All local associative memories of one subset are assigned to the same local associative memory of the "previous" layer.

The first layer is composed of one subset. Thus, all local associative memories of the first layer belong to the same subset. The tree structure of the HAM2 model is depicted in Figure 24.

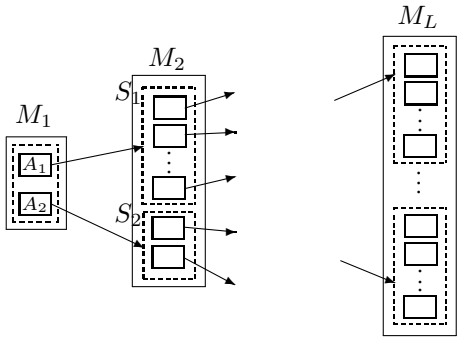


Figure 24: The tree structure of the HAM2 model with L ($L > 0$) layers $M_1, \ldots, M_L$. The local associative memories of the layer M_k are grouped into disjoint subsets. In figure, the local associative memories of the layer M_2 are grouped into two subsets S_1 and S_2. The subsets S_1 and S_2 are assigned to the local associative memories A_1 and A_2 of the "previous" layer M_1, respectively.

The hierarchical structure of the HAM2 model is reminiscent of the hierarchical organization of cortical areas in the visual system of the owl monkey [7]. In the owl monkey's brain, neurons in the lowest areas respond to information such as shape and motion, while the response in the areas at higher layers is more specific.

Now, we describe the training process of the HAM2 model. Let $\mathcal{T} = (T_1, \ldots, T_L)$ be a training tuple of the HAM2 model $\mathcal{H}$ with L ($L > 0$) layers. The training patterns from the training tuple $\mathcal{T}$ are to be stored in the model separately for the corresponding layers. All layers $M_1, \ldots, M_k$ of the HAM2 model are trained sequentially. The layer M_1 is trained first, then the training process is proceeded in the layer M_2 etc. For training the layer M_k ($1 < k \leq L$), the "previous" already trained layers $M_1, \ldots, M_{k-1}$ are used. In comparison with the HAM1 model, the layer training processes in the HAM2 model cannot be done in parallel.

During the training of the layer M_k ($1 < k \leq L$), the training patterns from the set T_k are presented to the layer M_k sequentially. The training of the pattern $\vec{x}$ ($\vec{x} \in T_k$) consists of two steps:

- The pattern $\vec{x}$ is recalled by already trained part of the model (i.e. the model composed of the layers $M_1, \ldots, M_{k-1}$). The recall of $\vec{x}$ is restricted only to the layers $M_1, \ldots, M_{k-1}$. The pattern $\vec{x}$ is presented to the layer M_1 and the recall process starts. The recall process is finished when the output $\vec{y}$ of the layer M_{k-1} is computed. We denote A the local associative memory that produces the output $\vec{y}$ of the layer M_{k-1}.

  The local associative memory A determines the subset S in the "next" layer M_k. The subset S is such a subset of M_k that is assigned to the local associative memory A of M_{k-1}. Thus, the output of the layer M_{k-1} provides the information about the subset S of M_k (for the pattern $\vec{x}$). The situation is depicted in Figure 25.

- When the corresponding subset S is detected, the pattern $\vec{x}$ can be stored in the layer M_k. The pattern $\vec{x}$ is stored in the "suitable" local associative memory in the subset S of M_k. If there is no "suitable" local memory in the subset S of M_k, a new local associative memory is created and added to the subset S of the layer M_k. In this case, the training pattern $\vec{x}$ is stored in the newly created associative memory of the subset S.

The training process of the first layer M_1 is very simple. Since all local associative memories are grouped into one subset, there is no need to detect the corresponding subset. Now, we describe the DLT2-algorithm in the formal way.

The DLT2-algorithm (for the layer M_k):

Step 1: The weight matrices of all local associative memories of the layer M_k are set to zero.

Step 2: A training pattern $\vec{x}$ is selected from T_k.

Step 3: If $M_k = M_1$, **Step 6**. If $M_k \neq M_1$, **Step 4**.

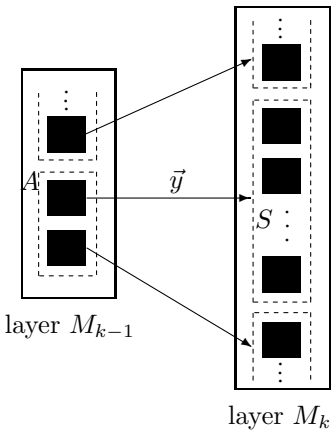


Figure 25: The scheme of the training algorithm in the layer M_k in the HAM2 model. In the layer M_{k-1}, the local associative memory A produces the output pattern $\vec{y}$. The output $\vec{y}$ of the layer M_{k-1} provides the information about the subset S of the layer M_k.

Step 4: The pattern $\vec{x}$ is recalled by the model composed only of the layers $M_1, \ldots, M_{k-1}$ (according to the recall algorithm described later). We denote $\vec{y}$ the output of the layer M_{k-1} (for the pattern $\vec{x}$) and A the local associative memory of the layer M_{k-1} that produces the output pattern $\vec{y}$.

Step 5: The local associative memory A in M_{k-1} determines the corresponding subset S of local associative memories in the layer M_k. All local associative memories of the subset S (of M_k) belong to the same local associative memory A in M_{k-1} because of the HAM2 model tree structure (Figure 25).

Step 6: The pattern $\vec{x}$ is stored in all local associative memories belonging to the subset S of M_k (e.g. according to the Hebbian training rule - Equation 10).

Step 7: The pattern $\vec{x}$ (or the "noisy" version of the pattern) is recalled by all local associative memories belonging to the subset S of M_k. We denote $\vec{y}^i$ the output of the i-th local associative memory of S in the layer M_k ($i = 1, \ldots, \mid S \mid$). The symbol $\mid S \mid$ denotes the number of local associative memories in the subset S of M_k.

Step 8: The Hamming distance d^i of the training pattern $\vec{x}$ and the recalled output $\vec{y}^i$ ($i = 1, \ldots, \mid S \mid$) is computed

$$d^i = \text{Hamm}(\vec{x}, \vec{y}^i)$$

for all local associative memories in the subset S of M_k.

Step 9: The minimum Hamming distance d^{min} is found

$$d^{min} = \min_{i=1,\ldots,|S|} \{d^i\}.$$

The parameter min is set to the index of the local associative memory in the subset S (of M_k) with satisfying d^{min}. If there exist more than one local associative memories belonging to the subset S of M_k with the same minimum Hamming distance d^{min}, the parameter min is set to the lowest index of the local associative memory with satisfying d^{min}. The index min determines only one local associative memory of the subset S (of M_k).

Step 10: The pattern $\vec{x}$ is unlearnt from every local associative memory of the subset S (of M_k) that fulfills the condition

$$d^i \neq 0 \text{ or } i \neq min$$

where i is the index of the local associative memory in the subset S (of M_k). The process of unlearning the pattern $\vec{x} = (x_1, \ldots, x_n)$ is the operation inversed to the pattern training (Equation 40).

Step 11: If the pattern $\vec{x}$ is unlearnt from all local associative memories of the subset S (of M_k), i.e.

$$\nexists i, 1 \leq i \leq \mid S \mid \text{ for which } d^i = 0,$$

a new local associative memory is created and added to the subset S of the layer M_k. The pattern $\vec{x}$ is stored in this newly created local associative memory (e.g. according to the Hebbian training rule).

Step 12: If there is any other training pattern in T_k, **Step 2**.

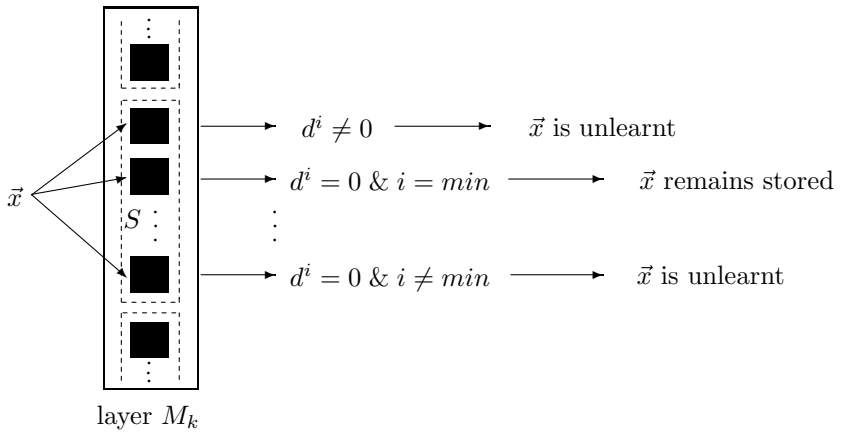


Figure 26: The scheme of the layer training algorithm in the HAM2 model. The pattern $\vec{x}$ is stored in the "suitable" local associative memory in the subset S of M_k (if it exists).

The scheme of the DLT2-algorithm is illustrated in Figure 26. There is no need to state the number of local associative memories before the HAM2 training process starts because the local memories are automatically added to the corresponding layers during training. The training process can start with one local associative memory in one subset in every layer. In this case, the

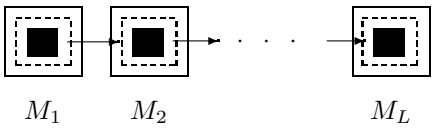


Figure 27: The initial structure of the HAM2 model (in the form of a path)

initial structure of the HAM2 model forms a path (i.e. the degenerated tree) - Figure 27. Other local associative memories are added to the HAM2 model during the training process according to incoming patterns. The number of local associative memories in the HAM2 model varies with the structure of incoming data. The number of subsets in every layer depends on the structure of incoming patterns, too.

The DLT2-algorithm (as well as the DLT1-algorithm in the HAM1 model uses the heuristics to select the local associative memory for storing the presented pattern. The pattern remains stored in the local associative memory of the corresponding subset where the pattern (or its "noisy version") is recalled correctly. Anyway, some previously stored patterns can be recalled incorrectly (after storing some other patterns) or can even become lost.

The recall process in the HAM2 model is similar to the recall process of the HAM1 model, but it takes advantage of the tree structure of the HAM2 model. The recall process is proceeded sequentially in the layers of the HAM2 model. During recall, a pattern $\vec{x}$ is presented to the HAM2 model. The pattern $\vec{x}$ represents an input to the first layer M_1 (i.e. ${}^1\vec{x} = \vec{x}$). At every time step k ($1 \leq k \leq L$), the layer M_k receives the input ${}^k\vec{x}$ and the recall process of this layer can start.

First, the output ${}^{k-1}\vec{y}$ of the "previous" layer M_{k-1} ($k > 1$) gives the information about the subset ${}^kS$ of the layer M_k. The recall process of the pattern ${}^k\vec{x}$ is restricted to the local associative memories belonging to the subset S. Then, the pattern ${}^k\vec{x}$ is presented to the local associative memories of the subset ${}^kS$ (of M_k). Each of these local associative memories recalls the outputs ${}^k\vec{y}^i$ ($i = 1, \ldots, \mid S \mid$). The output ${}^k\vec{y}$ of the layer M_k is the output ${}^k\vec{y}^i$ that is "the most similar" to the input pattern ${}^k\vec{x}$. We use the Hamming distance to measure the similarity between patterns, but other metrics can be used as well.

The output ${}^k\vec{y}$ can be combined with the output ${}^{k-1}\vec{y}$ of the "previous" layer M_{k-1} (the circle "$\otimes$" in Figure 28) in order to compute the input ${}^{k+1}\vec{x}$ to the "next" layer M_{k+1}. In our implementation, the difference pattern calculation is used to produce the input ${}^{k+1}\vec{x}$. Other functions can be considered as well. The output of the "last" layer M_L is combined with the output of the layer M_{L-1} to produce the output of the whole HAM2 model. The structure of the recall process in the HAM2 model is depicted in Figure 28.

We describe the so-called LR2-algorithm (the layer recall algorithm) for recall in one layer of the HAM2 model in formal way.

The LR2-algorithm (for the layer M_k):

Step 1: The input pattern ${}^k\vec{x}$ is presented to the layer M_k.

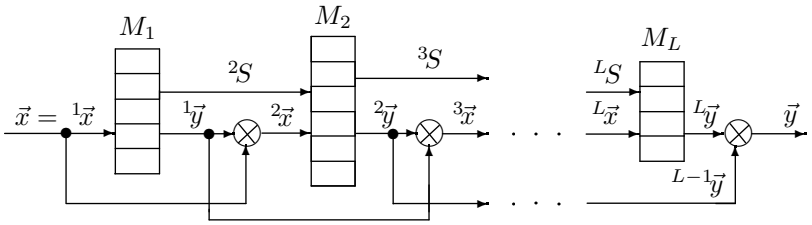


Figure 28: The recall process of the HAM2 model with L layers $M_1, \ldots, M_L$

Step 2: If $M_k = M_1$, **Step 4**. If $M_k \neq M_1$, **Step 3**.

Step 3: The recall process of the "previous" layer M_{k-1} provides the information about the subset ${}^kS$ that is responsible for the recall process in the layer M_k (according to the local associative memory that recalls the pattern ${}^{k-1}\vec{y}$).

Step 4: The pattern ${}^k\vec{x}$ is recalled by all local associative memories in the subset ${}^kS$ in M_k. We denote ${}^k\vec{y}^i$ output of the i-th local associative memory in the subset ${}^kS$ of M_k $(i = 1, \ldots, |\, {}^kS\,|)$.

Step 5: The Hamming distance d^i of the input pattern ${}^k\vec{x}$ and the recalled output ${}^k\vec{y}^i$ is computed

$$d^i = \mathrm{Hamm}({}^k\vec{x},\ {}^k\vec{y}^i)$$

for every local associative memory belonging to the subset ${}^kS$ of M_k $(i = 1, \ldots, |{}^k\,S\,|)$.

Step 6: The minimum Hamming distance d^{min} is found

$$d^{min} = \min_{i=1,\ldots,|{}^kS|} \{d^i\}.$$

The parameter min is set to the index of the local associative memory in the subset ${}^kS$ of M_k with satisfying d^{min}. If there exist more than one local associative memories in the subset ${}^kS$ in M_k with the same minimum Hamming distance d^{min}, the parameter min is set to the lowest index of the local associative memory with satisfying d^{min}.

Step 7: The local associative memory with the index min determines the subset ${}^{k+1}S$ in the "next" layer M^{k+1} (for $k < L$) according to the tree structure of the HAM2 model.

Step 8: The output ${}^k\vec{y}$ of the layer M_k is the output ${}^k\vec{y}^{min}$

$${}^k\vec{y} = {}^k\vec{y}^{min}.$$

The scheme of the LR2-algorithm in the layer M_k is shown in Figure 29.

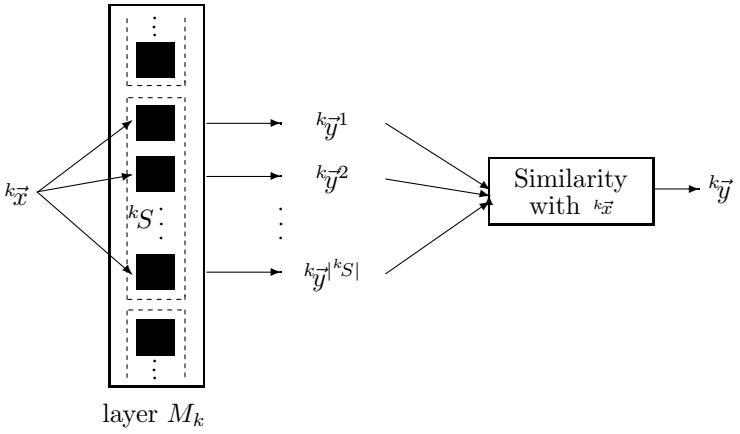


Figure 29: The scheme of the recall process in the layer M_k of the HAM2 model. The pattern $^k\vec{x}$ is recalled by local associative memories of the subset ^kS. The output $^k\vec{y}$ of the layer M_k is the output $^k\vec{y}^i$ that is "the most similar" to the input pattern $^k\vec{x}$.

.

6.4 Implementation of HAM model

Here, we sketch the main features of our implementation of the hierarchical associative memories. The system of the hierarchical associative memory is implemented as the console application for the operating system Windows (win32). It is written in the language C/C++. The implemented hierarchical associative memory is to be found on the enclosed CD.

The hierarchical associative memory is composed of local associative memories grouped into several layers. First, the implementation of the local associative memory is outlined. Then, the basic concept of hierarchical associative memory implementation is presented.

Local associative memory

First, we focus on the implementation of the local associative memory that is represented by the autoassociative memory. Our ultimate goal has been to propose such an implementation of the local associative memory that could be further used in the implementation of our hierarchical associative memory. The local associative memory is implemented as a class `AssMemNet`. Below, we list the most important characteristics (data and methods) of this class in pseudocode:

```
class AssMemNet {
  data:
    // Weight matrix of the associative memory
    TMatrix weight;
    // Number of neurons in the associative memory
    int numberNeuron;
  methods:
    // Method for training the pattern 'pattern'
    virtual void learnPattern(TPattern* pattern);
    // Method for unlearning the pattern 'pattern'
    virtual void unLearnPattern(TPattern* pattern);
    // Method for recalling the pattern 'unknPattern'
    // 'recPattern' is the recalled pattern
    virtual int recalling(TPattern* unknPattern,
                          TPattern* recPattern);
}
```

The above-listed methods correspond to main functions of the local associative memory: i.e. storing a pattern (`learnPattern`), unlearning a pattern (`unLearnPattern`) and recalling a pattern (`recalling`). In the HAM-models, these basic functions can be further combined together in order to achieve a more reliable behaviour.

In our system, patterns processed by the network are represented as dynamical arrays `TPattern` (in order to enable a variant pattern dimension and working with various types of pattern elements). Each element of the structure

`TPattern` corresponds to one element of the presented pattern. The weight matrices of the local associative memory are also represented as dynamical matrices in order to allow processing various types of elements.

The method `learnPattern` for storing a pattern is very simple and corresponds to the training rule (e.g. the Hebbian training rule):

```
void learnPattern(TPattern* pattern) {
  for all items [i,j] in the weight matrix {
    if (i<>j) {
      update the weight w[i,j] according to the presented pattern;
    }
  }
}
```

The whole training process (i.e. storing all training patterns) represents a loop and in each iteration the method `learnPattern` is called for storing one training pattern. Since, there are known many methods for training associative memories, other training methods can be implemented by inheriting the class `AssMemNet` and redefining the corresponding methods.

The function `recalling` implements the recall algorithm in the local associative memory.

```
int recalling(TPattern* unknPattern, TPattern* recPattern) {
  memoryOutput = unknPattern;
  repeat
    // one iteration
    for all neurons in the memory {
      update the output of all neurons according to
      its potential value
    }
    recPattern = memoryOutput;
  until (recPattern and recPattern from the previous iteration
         are the same)
}
```

The neurons in the local associative memory are selected for an update uniformly (for-loop). The recall process stops when no neuron changes its output during two sequential iterations. Then, `recPattern` is the output of the local associative memory (i.e. output of all neurons in the memory). If recall process of a pattern is successful, then the function returns the value 1. Otherwise, the function returns the value 0.

Further data and methods not mentioned explicitly refer to control and monitoring of the performance of the network etc.

Hierarchical associative memories

Now, we sketch the principle of our implementation of the hierarchical associative memory. The main structure of our implementation of the hierarchical associative memory is shown in Figure 30.

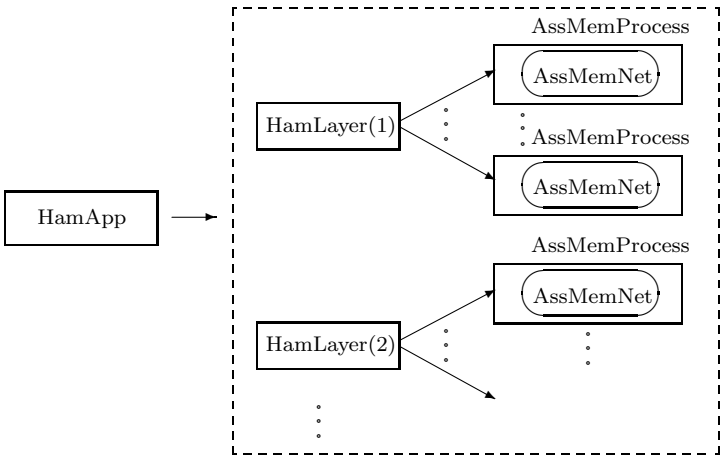


Figure 30: The structure of our implementation of the hierarchical associative memory

`HamApp` is the main class of the system. It serves as the manager of the whole system. `HamApp` controls running of the hierarchical associative memory. It reads the data from files (training patterns, patterns to be recalled), checks results, writes outputs to files (recalled patterns, run-time messages, etc), "communicates" with the hierarchical associative memory.

The basic functions of the manager `HamApp` are:

- Creating the hierarchical associative memory.
  The hierarchical associative memory with the initial structure (e.g. one local associative memory in every layer) is created. The local associative memories are allocated and their constructors are called.
- Reading the training patterns from files.
- Training the hierarchical associative memory.
  All training patterns are sequentially stored in the hierarchical associative memory according to the corresponding training algorithm (described in the previous sections).
- Reading the patterns to be recalled from files.
- Recalling the patterns by the hierarchical associative memory.
  All patterns to be recalled are presented to the hierarchical associative memory in the sequential order. The hierarchical associative memory recalls the presented patterns according to the recall algorithm (described in the previous sections).
- Writing results to files.
  The results (recalled patterns, errors, etc) acquired during the recall process are written to files. Files with results can be processed by other programs.

- Destroying the hierarchical associative memory.
  All local associative memories of the hierarchical associative memory are destroyed. The destructors are called and the memory is deallocated.

The hierarchical associative memory is composed of local associative memories grouped into layers. The layer of the hierarchical associative memory is implemented as the class `HamLayer` (Figure 30). The class contains the array of local associative memories (belonging to the layer) and the number of local associative memories in the layer. Since the local associative memories are dynamically added to the layer during the training, the class has the method for the array extension. The constructor of the class `HamLayer` creates the layer with the initial structure, i.e. one local associative memory (in one subset). The destructor destroys all existing local associative memories of the layer.

The local associative memory is implemented as the class `AssMemNet`. The class `AssMemNet` has only attributes and methods related to the associative memory performance. In the implementation of the hierarchical associative memory, it is needed to know more information about every local associative memory (e.g. its position in the layer, the corresponding local associative memory of the "previous" layer etc.). For this reason, the class `AssMemNet` is encapsulated into the class `AssMemProcess`. The situation is depicted in Figure 30.

The class `AssMemProcess` contains the instance of the class `AssMemNet` and the methods providing the access to the class `AssMemNet`. In addition, the class `AssMemProcess` contains the attributes necessary for operating within the whole hierarchical associative memory. The basic methods of the class `AssMemProcess` are:

- Creating the local associative memory
  The constructor of the class `AssMemProcess` creates the instance of the class `AssMemNet` and sets all attributes to their initial values.

- Storing the pattern in the local associative memory
  The method `learnPattern` of the class `AssMemNet` is called in order to store the presented training pattern in the memory.

- Unlearning the pattern from local associative memory
  The method `unLearnPattern` of the class `AssMemNet` is called to unlearn the pattern from the local associative memory.

- Recalling the pattern by the local associative memory
  The method `recalling` of the class `AssMemNet` is called and the presented pattern is recalled by the memory. The recalled pattern is sent back (to the manager `HamApp`).

- Storing and recalling the pattern by the local associative memory
  The method `learnPattern` of the class `AssMemNet` is called in order to store the presented pattern. Afterwards, the actually stored pattern is recalled by the local associative memory (the method `recalling` of the class `AssMemNet`). The result of the recall process is sent back (to the manager `HamApp`).

- Recalling the incomplete pattern by the local associative memory
  The method `recalling` of the class `AssMemNet` is called and the presented incomplete pattern is recalled by the memory. The recalled (complete) pattern is sent back to `HamApp`.
- Destroying the local associative memory
  The destructor of the class `AssMemProcess` destroys the instance of the class `AssMemNet` and deallocates the memory.

7 Analysis and experimental results

The key points in neural network performance are the storage and recall abilities. In applications of processing large amounts of spatial patterns (e.g. to help to move in an environment), it is required robust abilities of the model with respect to

1. processing of stored patterns
2. processing of incomplete patterns.

The incomplete patterns represent patterns unknown in some parts of their surface.

Our extensive experimental simulations carried out are focused on the analyses of the HAM1 and HAM2 model abilities in the comparison with the standard associative memory model [30] and the model of cascade associative memories [26]. At the end of this chapter, the memory- and time- complexity of the HAM models are sketched.

7.1 Data for experimental simulations

Our experimental simulations are restricted to the two-layer hierarchy of patterns. In this scheme, the first-layer patterns and the second-layer patterns are called ancestors and descendants, respectively. We outline the possible interpretation of such patterns. Then, generation of patterns processed in the experimental simulations is described.

7.1.1 Data interpretation

Two-layer hierarchy of patterns can have a simple geometrical interpretation. All patterns are distributed in an input space in clusters. The centers of clusters correspond to the representative patterns. They could form the first-layer patterns. Intrinsic patterns distributed in the clusters around the representative patterns could form the second-layer patterns. Patterns in one cluster are expected to be strongly correlated to each other. Moreover, the patterns in a cluster are expected to be correlated with the corresponding representative pattern situated in its center. The situation is shown in Figure 31.

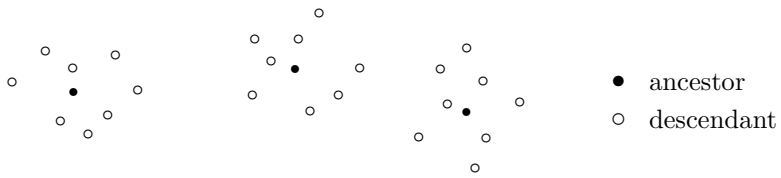


Figure 31: One possible interpretation of the two-layer data

The generalization of this scheme to an arbitrary number of layers is straightforward but the geometrical interpretation does not have to be so simple. Such

a data organization can frequently appear in real problems and neural networks in the human brain memorize and recall them flexibly [27].

7.1.2 Data generation

For experiments, we generate q ($q = 100$) sets of m ($m = 100$) randomly generated bipolar patterns; each of size 15×15 elements ($n = 225$).

The procedure for generating one pattern element is very simple and can be written (in pseudocode) as

```
if ((rand() mod 2) == 0){
   val = 1;
}
else {
   val = -1;
}
```

The value `val` corresponds to the generated value of the pattern element. The `rand` function returns a pseudorandom integer in the range 0 to `RANDMAX`.

After generating the patterns, every experiment is run on its set of patterns independently on other data sets. Experiments are repeated for every data set. In our experiments, we process "relatively small" patterns, as it is necessary to perform a huge number of experiments to analyze the storage/recall ability of our hierarchical associative memory and compare our model with the standard associative memory [30] and the cascade associative memory [26].

We focus on experiments processing the randomly generated patterns because of an easiness of mass generation. Though the patterns are randomly generated, we can assume that they are correlated (according to Definition 2.3). Thus, the randomly generated patterns are sufficient for testing the model abilities with respect to processing a large number of (correlated) patterns.

After pattern generation, the patterns of every data set with the smallest cumulative correlation between the respective patterns are chosen to be the first-layer patterns (the ancestors). The remaining patterns are used to form the second-layer patterns (the descendants). In this way, the first- and second-layer patterns do not have to fully correspond with the interpretation depicted in Figure 31.

In order to fulfill the above mentioned interpretation, the patterns have to be separated into the first- and second- layer patterns using more sophisticated methods (the clustering methods - e.g. [11] - or fuzzy clustering methods - e.g. [65]). Unfortunately, most clustering methods need to know the total number of clusters in advance. In data processing, this knowledge is usually not known because it is available only a sequence of spatial patterns (and the underlying data structure is unknown).

Every data set is composed of the ancestors and the descendants. We denote *anc* and *desc* the number of ancestors and descendants in the data set, respectively. It holds

$$m = anc + desc \tag{41}$$

where m is the total number of patterns in every data set. In the experiments, the number of ancestors and descendants does not have to be expressed explicitly. It is possible to express only the ratio between the number of ancestors and the number of descendants in the data set. The ratio is called the pattern rate.

Definition 7.1 *Let anc denote the number of ancestors and desc denote the number of descendants in the data set D. A pattern rate r of data set D is a ratio between the number of ancestors and the number of descendants*

$$r = \frac{anc}{desc}$$

in the data set D.

7.2 The HAM models for experimental simulations

Here, we describe the hierarchical associative memories that are used in the experimental simulations. Since two-layer patterns (called ancestors and descendants) are processed in the simulations, the hierarchical associative memories are composed of two layers.

7.2.1 Difference patterns

In two-layer hierarchy, we can assume that the descendants are expected to be correlated with their ancestors. Therefore, the second layer of the hierarchical associative memory does not store the descendants themselves but the so-called difference patterns. The difference patterns contain only the information on the differences between the descendant and the corresponding ancestor. They are in the form of bipolar patterns as well. In this way, the difference patterns become sparser with increasing correlation between the descendants and their ancestors, which is in general expected to allow a higher storage capacity of the hierarchical associative memory.

Now, we define the difference pattern in the formal way.

Definition 7.2 *Let $\vec{y} = (y_1, \ldots, y_n)$ be $n-$dimensional bipolar pattern of a descendant ($n > 0$). Let $\vec{x} = (x_1, \ldots, x_n)$ be $n-$dimensional bipolar pattern of an ancestor corresponding to the descendant $\vec{y}$. A difference pattern $\vec{d} = (d_1, \ldots, d_n)$ is $n-$dimensional bipolar pattern defined by*

$$d_i = \begin{cases} 1 & \text{if } y_i = x_i \\ \\ -1 & \text{otherwise} \end{cases} \tag{42}$$

for $i = 1, \ldots, n$.

7.2.2 Training process

During the training of the hierarchical associative memory, ancestors are stored in the first layer of the model according to the layer training algorithm. Afterwards, the corresponding ancestor is found for every descendants. This is

achieved by recalling the descendant in the first - already trained - layer of the model. The difference pattern of the descendant and the recalled ancestor is calculated and the pattern stored in the second layer according to the layer training algorithm.

In the implemented hierarchical associative memory, the modified Hebbian training rule is used for storing the patterns (the ancestors and the difference patterns) in the corresponding local associative memory. The weight update $\Delta^p w_{ij}$ after presenting the $n-$ dimensional bipolar pattern $\vec{x}^p = (x_1^p, \ldots, x_n^p)$ is calculated as

$$\Delta^p w_{ij} = \begin{cases} (x_i^p - \overline{x}^p)(x_j^p - \overline{x}^p) & \text{for } i \neq j \\ \\ 0 & \text{otherwise} \end{cases} \tag{43}$$

where $x_i^p \in \{+1, -1\}$ and $i, j = 1, \ldots, n$. The symbol $\overline{x}^p$ is denotes the average value of all elements in $\vec{x}^p$:

$$\overline{x}^p = \frac{1}{n} \sum_{i=1}^{n} x_i^p. \tag{44}$$

The local associative memory in the HAM model is the standard autoassociative memory. The storage capacity of the local associative memory is 0.15 when the patterns stored in it are orthogonal (or close to orthogonal) [30]. Thus, the maximum number of patterns that can be stored in the local associative memory as stable states corresponds to $0.15 \cdot n$ where n $(n > 0)$ is the dimension of patterns.

With respect to the robust storage and recall ability requirement, we define the so-called capacity coefficient c $(0 < c \leq 1)$ that reduces the maximum number of patterns stored in every local associative memory. The maximum number of patterns that can be stored in every local associative memory is given by the formula

$$\max = 0.15 \cdot c \cdot n, \tag{45}$$

where n $(n > 0)$ is the pattern dimension.

The capacity coefficient c takes the value of 1 for the standard autoassociative memory. For decreasing capacity coefficient c, the maximum number of patterns that can be stored in the local associative memory is reduced. In applications based on processing the patterns, it is necessary to recall all stored patterns "without any error". We expect that the abilities of the HAM model with respect to the recall the stored patterns will rise when the maximum number max of patterns stored in every local associative memory is decreased.

7.2.3 Recall process

During the recall, an input pattern is presented to the first layer of the hierarchical associative memory to recall the corresponding ancestor. Once the ancestor is recalled, the corresponding difference pattern is calculated by combining the

recalled ancestor with the input pattern. The difference pattern is propagated to the second layer of the model to be recalled. The output of the model is produced by combining the recalled difference pattern with its recalled ancestor (by the operation inversed to the difference pattern calculation - Equation 42).

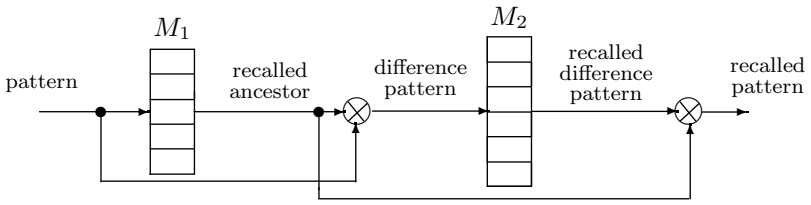


Figure 32: The recall process in the implemented hierarchical associative memory with two layers M_1 and M_2

Since patterns are stored in the local associative memories using modified Hebbian training rule (Equation 43), it is necessary to modify also the recall process for every local associative memory of the hierarchical associative memory. The recall process of the local associative memory is given by

$$y_i(t+1) = \mathrm{sgn}\Big(\sum_{j=1}^{n} w_{ji}\big(y_j(t) - \overline{y}(t)\big)\Big) \tag{46}$$

where $\vec{y}(t+1) = (y_1(t+1), \ldots, y_n(t+1))$ represents the output of the local associative memory at time $t+1$ ($t > 0$) and $i = 1, \ldots, n$. The symbol $\overline{y}(t)$ is the average value of all elements in the output $\vec{y}(t)$ (Equation 44).

7.3 Processing of stored patterns

In this chapter, we focus the so-called storage ability of the hierarchical associative memory. The storage ability measures the ability of the model to recall the stored patterns. The patterns from every data set are stored in the model according to the corresponding training algorithm. Afterwards, all patterns from the data set are presented to the model to be recalled (according to the concrete recall algorithm).

To determine the storage ability of our hierarchical associative memory, we measure the ratio of patterns recalled correctly in every data set. Afterwards, we focus on the patterns that are recalled with an error (i.e. incorrectly). We analyze the magnitude of the error in the patterns recalled incorrectly. In the following experiments, we analyze the magnitude of the error for the pattern elements.

With respect to the storage ability analysis, we define several terms used further in the text. We also sketch the theory used in the analysis.

Definition 7.3 *Let AM be a model of an associative memory. Let $\vec{x}$ be n-dimensional pattern ($n > 0$) presented to the model AM. Let $\vec{y}$ be n-dimensional pattern recalled by the model AM after presenting the input pattern $\vec{x}$.*

- *The pattern $\vec{x}$ is recalled correctly, if the recalled pattern $\vec{y}$ coincides with the presented pattern $\vec{x}$, i.e.*

$$Hamm(\vec{x}, \vec{y}) = 0.$$

- *The pattern $\vec{x}$ is recalled with error k ($0 < k \leq n$), if the recalled pattern $\vec{y}$ varies from the presented pattern $\vec{x}$ in k elements, i.e.*

$$Hamm(\vec{x}, \vec{y}) = k.$$

If the pattern $\vec{x}$ is recalled with the error k and $k \neq 0$, we say that the pattern $\vec{x}$ is recalled incorrectly.

Let X be a random variable that corresponds to the number of patterns recalled correctly in one data set consisting of m patterns ($m = 100$). Suppose that X has a binomial distribution since the number of pattern recalled correctly by an associative memory does not depend on processing other data sets. A mean EX of the distribution is

$$EX = m \cdot p_{ok} \tag{47}$$

and a variance $varX$ is given by the formula

$$varX = m \cdot p_{ok} \cdot (1 - p_{ok}) \tag{48}$$

[5], where p_{ok} is the probability that the pattern is recalled correctly by an associative memory.

A relative number of patterns recalled correctly in one data set is a random variable $Y = X/m$. The random variable Y has the distribution with the mean EY

$$EY = p_{ok} \tag{49}$$

and the variance $varY$

$$varY = p_{ok} \cdot (1 - p_{ok})/m \tag{50}$$

[5]. The Central Limit Theorem says that the distribution Y can be approximated by a normal distribution with the same parameters

$$Y \sim N\Big(p_{ok}, p_{ok} \cdot (1 - p_{ok})/m\Big) \tag{51}$$

for large values of the parameter m (e.g. [84]).

Unfortunately, the probability p_{ok} (the pattern is recalled correctly by the associative memory) is unknown. The mean EY and the variance $varY$ are unknown too. In this case, the so-called sample mean and sample variance are used as point estimates for the unknown mean EY and variance $varY$, respectively.

We compute the sample mean and the sample variance of the distribution Y by means of the experimental results $Y_1, \ldots, Y_q$. The symbol Y_i denotes the

ratio between the number of stored patterns recalled correctly and the number of stored patterns in the i-th experiment ($1 \leq i \leq q$). Every experiment handles its data set; the i-th experiment deals with the i-th data set. All experiments are mutually independent (i.e. results of one experiment are not affected by other experiments). The sample mean $\overline{Y}_q$ is given by the formula

$$\overline{Y}_q = \frac{1}{q} \cdot \sum_{i=1}^{q} Y_i \tag{52}$$

where q ($q > 0$) is the total number of the experiments. The sample variance for q experiments is defined by

$$S_q^2 = \frac{1}{q-1} \cdot \sum_{i=1}^{q} (Y_i - \overline{Y}_q)^2. \tag{53}$$

The symbol S_q corresponds to the standard error (i.e. the estimated standard deviation).

In our analyses, we are not only interested in computing the point estimates for the mean and the variance, but also in determining how accurate the point estimates are. We use the so-called confidence intervals that quantify the uncertainty in the experimental results (i.e. in the point estimates).

A confidence interval gives an estimated range of values that is likely to include an unknown parameter. The estimated range is calculated based on the experimental results. A confidence level $\alpha \in (0, 1)$ of a confidence interval gives the probability $1 - \alpha$ that the interval produced by the method employed includes the true value of the parameter.

In our analyses, the confidence level $\alpha = 0.05$ is chosen. It corresponds to the 95% confidence interval. We calculate the confidence intervals for the mean EY and the variance $varY$ by means of $q = 100$ different experiments.

First, we focus on the confidence interval for the mean EY. The symbols $Y_{min,q}$ and $Y_{max,q}$ denote the lower and upper limit of the confidence interval for the mean using the q experiments, respectively. The $(1 - \alpha)\%$ confidence interval can be expressed as

$$Pr\Big(Y_{min,q} < EY < Y_{max,q}\Big) = 1 - \alpha \tag{54}$$

where α is the confidence level, q denotes the number of experiments and $Pr(\cdot)$ denotes the probability.

When the variance $varY$ is not known, it can be replaced by the sample variance S_q^2. Then, the random variable T

$$T = \frac{\overline{Y}_q - EY}{S_q/\sqrt{q}} \tag{55}$$

has the Student's $t-$distribution with $q-1$ degrees of freedom [5]. Consequently,

$$Pr\Big(\mid T \mid \leq \ t_{q-1}(1 - \alpha/2)\Big) = 1 - \alpha \tag{56}$$

where t_{q-1} are the quantiles of the Student's $t-$distribution with $q-1$ degrees of freedom [5]. From the previous formula, we get the $(1-\alpha)\%$ confidence interval for the mean EY

$$Pr\Big(\overline{Y}_q - t_{q-1}(1-\alpha/2)\cdot\frac{S_q}{\sqrt{q}} < EY < \overline{Y}_q + t_{q-1}(1-\alpha/2)\cdot\frac{S_q}{\sqrt{q}}\Big) = 1-\alpha. \quad (57)$$

Hence, the lower and upper limits of the $(1-\alpha)\%$ confidence interval are given by

$$Y_{min,q} = \overline{Y}_q - t_{q-1}(1-\alpha/2)\cdot\frac{S_q}{\sqrt{q}} \quad (58)$$

$$Y_{max,q} = \overline{Y}_q + t_{q-1}(1-\alpha/2)\cdot\frac{S_q}{\sqrt{q}}. \quad (59)$$

The values of the quantiles t_{q-1} of the Student's $t-$distribution with $q-1$ degrees of freedom can be found in the literature (e.g. [5]).

Now, we construct the confidence interval for the variance $varY$ of the normal distribution Y. The statistics Z

$$Z = \frac{q-1}{varY}\cdot S_q^2 \quad (60)$$

has the χ^2-distribution with $q-1$ degrees of freedom [5]. Consequently, the $(1-\alpha)\%$ confidence interval for the variance $varY$ is

$$Pr\left(\frac{(q-1)\cdot S_q^2}{\chi^2_{q-1}(1-\alpha/2)} < varY < \frac{(q-1)\cdot S_q^2}{\chi^2_{q-1}(\alpha/2)}\right) = 1-\alpha \quad (61)$$

where χ^2_{q-1} are the quantiles of the χ^2-distribution with $q-1$ degrees of freedom. We denote $S^2_{min,q}$ and $S^2_{max,q}$ the lower and upper limit of the confidence interval for the variance, respectively. The lower and upper limit of the $(1-\alpha)\%$ confidence interval can be expressed as

$$S^2_{min,q} = \frac{(q-1)\cdot S_q^2}{\chi^2_{q-1}(1-\alpha/2)} \quad (62)$$

$$S^2_{max,q} = \frac{(q-1)\cdot S_q^2}{\chi^2_{q-1}(\alpha/2)} \quad (63)$$

[5]. The values of the quantiles χ^2_{q-1} of the χ^2-distribution with $q-1$ degrees of freedom can be found in the literature (e.g. [5]).

7.3.1 Storage ability of the HAM1 and HAM2 model

To analyze the storage ability of the hierarchical associative memory, we measure the ratio Y_i between the number of stored patterns recalled correctly and the

number of all stored patterns in every data set D_i $(1 \leq i \leq q)$. [2] We make a number of experiments using various parameters:

- q $(q = 100)$ different data sets $D_1, \ldots, D_{100}$,
- four different pattern rates $r \sim 1/5$, $r = 1/3$, $r \sim 1/2$, $r = 1$,
- nine different capacity coefficients $c = 1$, $c = 0.9$, ..., $c = 0.2$.

In our experiments, the symbols $r \sim 1/5$ and $r \sim 1/2$ correspond to the pattern rates $r = 17/83$ and $r = 33/67$, respectively. The sample mean $\overline{Y}_q$ (Equation 52) and the sample variance S_q^2 (Equation 53) are calculated based on the results obtained for the corresponding parameters r (Section 7.1.2) and c (Section 7.2.2). Afterwards, we find the 95% confidence interval for the mean (Equation 58, Equation 59) and the variance (Equation 62, Equation 63).

The same experiments are performed by the HAM1 model and the HAM2 model. The sample mean and the corresponding confidence interval for the HAM1 model are summarized in Table 1. For the HAM2 model, the results are summarized in Table 2. The experiments are repeated for the different values of the pattern rate r and the different values of the capacity coefficient c.

Table 1 and Table 2 show the relative number of stored patterns recalled correctly by the HAM1 model and the HAM2 model, respectively. It represents the storage ability of the HAM models.

With the increasing pattern rate r, the storage ability does not have to be improved. The storage ability is affected by the number of local associative memories in the layer. Decrease in the storage ability reflects the reduction in the number of local associative memories (see Table 11). In this case, the local associative memories are almost overloaded. It leads to decrease in the storage ability because more stored patterns are recalled with any error.

The results for the HAM1 and HAM2 models are compared and discussed in more detail in the following section 7.3.2.

[2] The symbol Y_i does not have any connection with the symbol y_i that denotes the output of the neuron i in a network

The HAM1 model: The mean				
	$r \sim 1/5$	$r = 1/3$	$r \sim 1/2$	$r = 1$
c	$\overline{Y}_q$ $\langle Y_{min,q}, Y_{max,q} \rangle$	$\overline{Y}_q$ $\langle Y_{min,q}, Y_{max,q} \rangle$	$\overline{Y}_q$ $\langle Y_{min,q}, Y_{max,q} \rangle$	$\overline{Y}_q$ $\langle Y_{min,q}, Y_{max,q} \rangle$
1	0.7794 $\langle 0.7661, 0.7927 \rangle$	0.8106 $\langle 0.7974, 0.8238 \rangle$	0.826 $\langle 0.8135, 0.8385 \rangle$	0.8167 $\langle 0.8058, 0.8276 \rangle$
0.9	0.7904 $\langle 0.7782, 0.8026 \rangle$	0.8118 $\langle 0.7988, 0.8248 \rangle$	0.8213 $\langle 0.8118, 0.8308 \rangle$	0.8525 $\langle 0.8412, 0.8638 \rangle$
0.8	0.8326 $\langle 0.8239, 0.8414 \rangle$	0.8273 $\langle 0.8163, 0.8383 \rangle$	0.8437 $\langle 0.8353, 0.8521 \rangle$	0.8861 $\langle 0.8765, 0.8957 \rangle$
0.7	0.8823 $\langle 0.8756, 0.8890 \rangle$	0.8905 $\langle 0.8833, 0.8977 \rangle$	0.8756 $\langle 0.8677, 0.8835 \rangle$	0.8992 $\langle 0.8915, 0.9069 \rangle$
0.6	0.9421 $\langle 0.9371, 0.9471 \rangle$	0.928 $\langle 0.9224, 0.9336 \rangle$	0.943 $\langle 0.9383, 0.9477 \rangle$	0.952 $\langle 0.9474, 0.9566 \rangle$
0.5	0.9814 $\langle 0.9784, 0.9844 \rangle$	0.9715 $\langle 0.9681, 0.9749 \rangle$	0.9797 $\langle 0.9765, 0.9829 \rangle$	0.9739 $\langle 0.9702, 0.9776 \rangle$
0.4	0.9925 $\langle 0.9909, 0.9941 \rangle$	0.992 $\langle 0.99, 0.9940 \rangle$	0.9918 $\langle 0.9901, 0.9935 \rangle$	0.992 $\langle 0.9901, 0.9939 \rangle$
0.3	0.9997 $\langle 0.9994, 1 \rangle$	0.9995 $\langle 0.9991, 0.9999 \rangle$	0.9999 $\langle 0.9997, 1.0001 \rangle$	1 $\langle 1, 1 \rangle$
0.2	1 $\langle 1, 1 \rangle$	1 $\langle 1, 1 \rangle$	1 $\langle 1, 1 \rangle$	0.9993 $\langle 0.9987, 0.9999 \rangle$

Table 1: The storage ability of the HAM1 model: the sample mean $\overline{Y}_q$ and the confidence interval $\langle Y_{min,q}, Y_{max,q} \rangle$ using various capacity coefficients c and pattern rates r (the confidence level $\alpha = 0.05$ and the number of experiments $q = 100$)

The HAM2 model: The mean				
	$r \sim 1/5$	$r = 1/3$	$r \sim 1/2$	$r = 1$
c	$\overline{Y}_q$ $\langle Y_{min,q}, Y_{max,q} \rangle$	$\overline{Y}_q$ $\langle Y_{min,q}, Y_{max,q} \rangle$	$\overline{Y}_q$ $\langle Y_{min,q}, Y_{max,q} \rangle$	$\overline{Y}_q$ $\langle Y_{min,q}, Y_{max,q} \rangle$
1	0.7794 $\langle 0.7661, 0.7927 \rangle$	0.8116 $\langle 0.7995, 0.8237 \rangle$	0.8383 $\langle 0.8255, 0.8512 \rangle$	0.8688 $\langle 0.8609, 0.8767 \rangle$
0.9	0.7904 $\langle 0.7782, 0.8026 \rangle$	0.8169 $\langle 0.8058, 0.8280 \rangle$	0.869 $\langle 0.8599, 0.8781 \rangle$	0.9074 $\langle 0.9, 0.9148 \rangle$
0.8	0.8326 $\langle 0.8239, 0.8414 \rangle$	0.8302 $\langle 0.8195, 0.8409 \rangle$	0.9002 $\langle 0.892, 0.9084 \rangle$	0.9201 $\langle 0.9119, 0.9283 \rangle$
0.7	0.8823 $\langle 0.8756, 0.8890 \rangle$	0.8983 $\langle 0.8911, 0.9055 \rangle$	0.9168 $\langle 0.9099, 0.9237 \rangle$	0.9516 $\langle 0.9457, 0.9575 \rangle$
0.6	0.9421 $\langle 0.9371, 0.9471 \rangle$	0.9416 $\langle 0.9371, 0.9461 \rangle$	0.9523 $\langle 0.9477, 0.9569 \rangle$	0.9732 $\langle 0.9695, 0.9769 \rangle$
0.5	0.9782 $\langle 0.9752, 0.981 \rangle$	0.9747 $\langle 0.9714, 0.978 \rangle$	0.9787 $\langle 0.9758, 0.9815 \rangle$	0.9813 $\langle 0.9781, 0.9845 \rangle$
0.4	0.9937 $\langle 0.992, 0.9954 \rangle$	0.9923 $\langle 0.9905, 0.9941 \rangle$	0.9926 $\langle 0.9909, 0.9943 \rangle$	0.9951 $\langle 0.9937, 0.9965 \rangle$
0.3	0.9996 $\langle 0.9991, 1.0001 \rangle$	0.9995 $\langle 0.999, 1.0000 \rangle$	0.9995 $\langle 0.999065, 0.9999 \rangle$	0.9998 $\langle 0.999521, 1.0001 \rangle$
0.2	1 $\langle 1, 1 \rangle$	1 $\langle 1, 1 \rangle$	1 $\langle 1, 1 \rangle$	0.9998 $\langle 0.9995, 1.0001 \rangle$

Table 2: The storage ability of the HAM2 model: the sample mean $\overline{Y}_q$ and the confidence interval $\langle Y_{min,q}, Y_{max,q} \rangle$ using various capacity coefficients c and pattern rates r (the confidence level $\alpha = 0.05$ and the number of experiments $q = 100$)

Now, we construct the 95% confidence interval for the variance $varY$. This interval represents the most likely distribution of the result variances, given the result's size and the variance.

The sample variance and the corresponding 95% confidence intervals for the HAM1 model and the HAM2 model are summarized in Table 3 and Table 4, respectively. The experiments are performed for the different values of the pattern rate r and the capacity coefficient c.

The HAM1 model: The variance				
	$r \sim 1/5$	$r = 1/3$	$r \sim 1/2$	$r = 1$
c	S_q^2 $\langle S_{min,q}^2, S_{max,q}^2 \rangle$	S_q^2 $\langle S_{min,q}^2, S_{max,q}^2 \rangle$	S_q^2 $\langle S_{min,q}^2, S_{max,q}^2 \rangle$	S_q^2 $\langle S_{min,q}^2, S_{max,q}^2 \rangle$
1	0.0045 $\langle 0.0035, 0.0061 \rangle$	0.0044 $\langle 0.0034, 0.006 \rangle$	0.004 $\langle 0.0031, 0.0054 \rangle$	0.003 $\langle 0.0023, 0.0041 \rangle$
0.9	0.0038 $\langle 0.0029, 0.0051 \rangle$	0.0043 $\langle 0.0033, 0.0058 \rangle$	0.0023 $\langle 0.0018, 0.0031 \rangle$	0.0032 $\langle 0.0025, 0.0044 \rangle$
0.8	0.0019 $\langle 0.0015, 0.0026 \rangle$	0.0031 $\langle 0.0024, 0.0041 \rangle$	0.0018 $\langle 0.0014, 0.0024 \rangle$	0.0023 $\langle 0.0018, 0.0031 \rangle$
0.7	0.0011 $\langle 0.0009, 0.0016 \rangle$	0.0013 $\langle 0.001, 0.0018 \rangle$	0.0016 $\langle 0.0012, 0.0021 \rangle$	0.0015 $\langle 0.0012, 0.002 \rangle$
0.6	0.0006 $\langle 0.0005, 0.0009 \rangle$	0.0008 $\langle 0.0006, 0.0011 \rangle$	0.0006 $\langle 0.0004, 0.0008 \rangle$	0.0005 $\langle 0.0004, 0.0007 \rangle$
0.5	0.0002 $\langle 0.0002, 0.0003 \rangle$	0.0003 $\langle 0.0002, 0.0004 \rangle$	0.0003 $\langle 0.0002, 0.0004 \rangle$	0.0003 $\langle 0.0003, 0.0005 \rangle$
0.4	0.0001 $\langle 0.0001, 0.0001 \rangle$	0.0001 $\langle 0.0001, 0.0001 \rangle$	0.0001 $\langle 0.0001, 0.0001 \rangle$	0.0001 $\langle 0.0001, 0.0001 \rangle$
0.3	0 $\langle 0, 0 \rangle$	0 $\langle 0, 0 \rangle$	0 $\langle 0, 0 \rangle$	0 $\langle 0, 0 \rangle$
0.2	0 $\langle 0, 0 \rangle$	0 $\langle 0, 0 \rangle$	0 $\langle 0, 0 \rangle$	0 $\langle 0, 0 \rangle$

Table 3: The storage ability of the HAM1 model: the sample variance S_q^2 and the confidence interval $\langle S_{min,q}^2, S_{max,q}^2 \rangle$ using various capacity coefficients c and pattern rates r (the confidence level $\alpha = 0.05$ and the number of experiments $q = 100$)

The HAM2 model: The variance				
	$r \sim 1/5$	$r = 1/3$	$r \sim 1/2$	$r = 1$
c	S_q^2 $\langle S_{min,q}^2, S_{max,q}^2 \rangle$	S_q^2 $\langle S_{min,q}^2, S_{max,q}^2 \rangle$	S_q^2 $\langle S_{min,q}^2, S_{max,q}^2 \rangle$	S_q^2 $\langle S_{min,q}^2, S_{max,q}^2 \rangle$
1	0.0045 $\langle 0.0035, 0.0061 \rangle$	0.0037 $\langle 0.0029, 0.005 \rangle$	0.0042 $\langle 0.0032, 0.0057 \rangle$	0.0016 $\langle 0.0012, 0.0022 \rangle$
0.9	0.0038 $\langle 0.0029, 0.0051 \rangle$	0.0031 $\langle 0.0024, 0.0042 \rangle$	0.0021 $\langle 0.0016, 0.0028 \rangle$	0.0014 $\langle 0.0011, 0.0019 \rangle$
0.8	0.0019 $\langle 0.0015, 0.0026 \rangle$	0.0029 $\langle 0.0022, 0.0039 \rangle$	0.0017 $\langle 0.0013, 0.0023 \rangle$	0.0017 $\langle 0.0013, 0.0023 \rangle$
0.7	0.0011 $\langle 0.0009, 0.0016 \rangle$	0.0013 $\langle 0.001, 0.0018 \rangle$	0.0012 $\langle 0.0009, 0.0016 \rangle$	0.0009 $\langle 0.0007, 0.0012 \rangle$
0.6	0.0006 $\langle 0.0005, 0.0009 \rangle$	0.0005 $\langle 0.0004, 0.0007 \rangle$	0.0005 $\langle 0.0004, 0.0007 \rangle$	0.0003 $\langle 0.0003, 0.0005 \rangle$
0.5	0.0002 $\langle 0.0002, 0.0003 \rangle$	0.0003 $\langle 0.0002, 0.0004 \rangle$	0.0002 $\langle 0.0002, 0.0003 \rangle$	0.0003 $\langle 0.0000, 0.0003 \rangle$
0.4	0.0001 $\langle 0.0001, 0.0001 \rangle$	0.0001 $\langle 0.0001, 0.0001 \rangle$	0.0001 $\langle 0.0001, 0.0001 \rangle$	0.0001 $\langle 0, 0.0001 \rangle$
0.3	0 $\langle 0, 0 \rangle$	0 $\langle 0, 0 \rangle$	0 $\langle 0, 0 \rangle$	0 $\langle 0, 0 \rangle$
0.2	0 $\langle 0, 0 \rangle$	0 $\langle 0, 0 \rangle$	0 $\langle 0, 0 \rangle$	0 $\langle 0, 0 \rangle$

Table 4: The storage ability of the HAM2 model: the sample variance S_q^2 and the confidence interval $\langle S_{min,q}^2, S_{max,q}^2 \rangle$ using various capacity coefficients c and pattern rates r (the confidence level $\alpha = 0.05$ and the number of experiments $q = 100$)

7.3.2 Comparison of the HAM1 and HAM2 model

Now, we compare our proposed HAM1 and HAM2 models with respect to the storage ability. The complete tables with the measured results are described in the previous section. The number of patterns recalled correctly by the HAM1 and HAM2 models is shown in the two graphs (Figure 33 and Figure 34).

In the figures, the curve represents the sample mean $\overline{Y}_q$ (i.e. the average number of patterns recalled correctly). The different curves are parametrized by the pattern rates r. Figure 33 shows the storage ability of the HAM1 model. The storage ability of the HAM2 model are depicted in Figure 34.

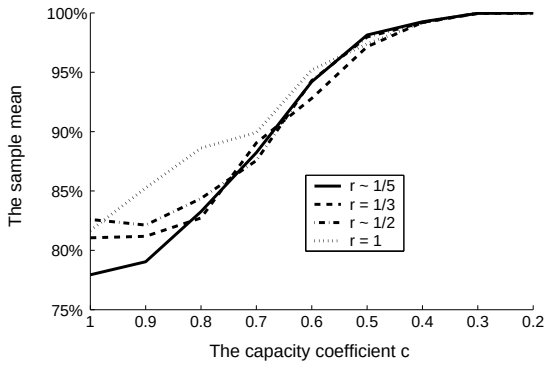


Figure 33: The number of patterns recalled correctly by the HAM1 model (the sample mean $\overline{Y}_{100}$) with respect to the various capacity coefficients c and the various pattern rates r

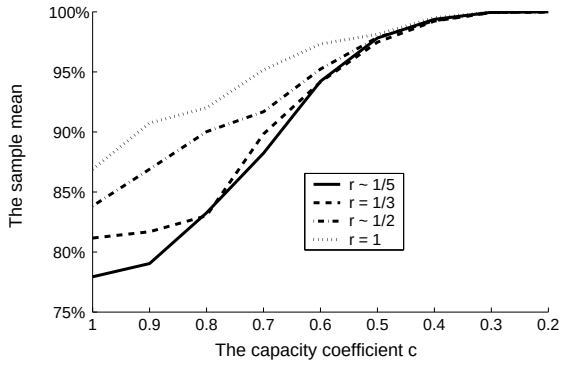


Figure 34: The number of patterns recalled correctly by the HAM2 model (the sample mean $\overline{Y}_{100}$) with respect to the various capacity coefficients c and the various pattern rates r

If $c \leq 0.5$, the results are very similar in both models (for every pattern rate r). In this case, more than 97% stored patterns are recalled correctly. As the capacity coefficient c and the pattern rate r are increased, the differences between the HAM1 and HAM2 model become more significant.

For small values of the pattern rate r, the number of local associative memories in the first layer is small. It causes that the structure of the HAM1 and HAM2 model is very similar. In this case, the tree structure of the HAM2 model has almost no influence on the storage ability of the HAM2 model.

The increasing pattern rate r induces the increasing number of ancestors in every data set. With the increasing number of ancestors in data sets, the number of local associative memories in the first layer rises too. The training algorithms of the first layer are the same in the HAM1 as well as the HAM2 model. Therefore, the ancestors are stored in the same local associative memories in both models. The descendants are stored in the second layer of the models. In the HAM1 model, the descendants are stored in any "suitable" local associative memory of the second layer. In the HAM2 model, a disjoint subset of local associative memories of the second layer is assigned to every local associative memory in the first layer (the tree structure). Thus, the descendants are stored in the HAM2 model in the different local associative memories than that of the HAM1 model.

For large values of the pattern rate r (especially for $r > 1/2$), the differences between the HAM1 model and the HAM2 model are significant. In this case, the storage ability is enhanced in the HAM2 model due to the influence of the tree structure. The descendants are "separated" to the local associative memories of the second layer in more natural way (it can retain the hierarchical structure of data). It causes the improvement of the storage process in the HAM2 model, since the patterns recalled incorrectly are mostly descendants.

The previous experiments examine only the patterns recalled correctly in every data set. Such patterns are recalled by the model without any error. Now, we focus on the patterns that are recalled incorrectly (i.e. at least one pattern element is recalled incorrectly). We analyze the distribution of the patterns recalled incorrectly by means of an error function. The error function measures the number of erroneously recalled elements in every pattern (recalled incorrectly).

We construct the histogram of the error in all patterns recalled incorrectly (in data sets). Figure 35 shows the histogram of the error in the patterns recalled incorrectly by the HAM1 model. The same experiments are performed for the HAM2 model and the corresponding histogram is depicted in Figure 36. In the figures, the horizontal axis corresponds to the number of erroneously recalled elements in a pattern. The vertical axis shows the number of patterns recalled with a given error (in relation to the total number of patterns). Different scalings are used in the vertical axes in Figure 35 and Figure 36. The patterns recalled correctly (i.e. error = 0) are not depicted in the figures.

The shapes of the curves are similar for both HAM1 and HAM2 models. The number of patterns recalled incorrectly by the HAM1 model is higher (in comparison with the HAM2 model) and thus the number of patterns recalled

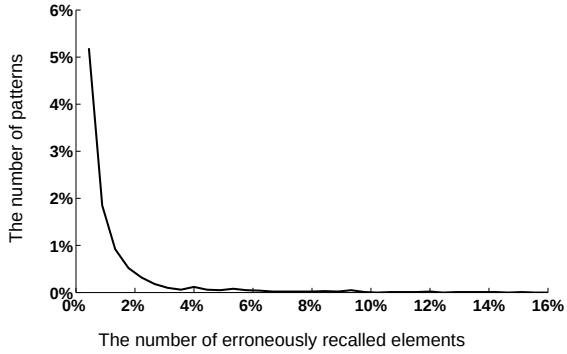


Figure 35: The histogram of the error in the patterns recalled incorrectly by the HAM1 model (the capacity coefficient $c = 0.7$ and the pattern rate $r = 1$)

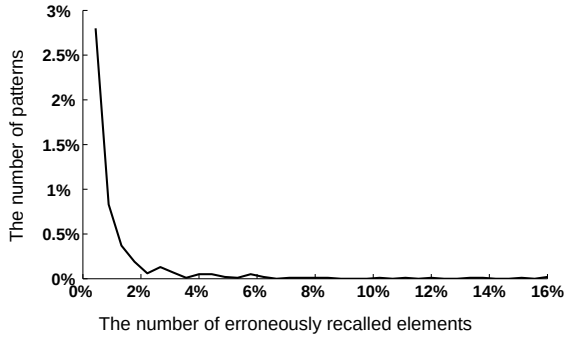


Figure 36: The histogram of the error in the patterns recalled incorrectly by the HAM2 model (the capacity coefficient $c = 0.7$ and the pattern rate $r = 1$)

with a given error is increased proportionally. The total number of patterns recalled incorrectly by the HAM1 model (with $c = 0.7$ and $r = 1$) is 10.08%. The HAM2 model with the same parameters recalls incorrectly only 4.84% patterns. In most patterns recalled incorrectly, the error does not exceed 2%. It means that only few pattern elements are recalled erroneously (e.g. the error 2% corresponds to 4 pattern elements recalled erroneously). The number of patterns in which the error is higher than 2% is diminutive.

The depicted results are acquired using the HAM1 and HAM2 models with the capacity coefficient $c = 0.7$ and the pattern rate $r = 1$. The results performed for other capacity coefficients c (and the pattern rate $r = 1$) are shown in Appendix of the thesis (Figure 55 - Figure 72). For other values of the pattern rate r, the histograms of the error in the patterns recalled incorrectly are almost identical. The differences are diminutive.

In the following experiments, we focus on the patterns that are recalled incorrectly and we analyze the magnitude of the error for the pattern elements. Figure 37 shows the magnitude of the error for the pattern elements in the HAM1 model. The results for the HAM2 model are summarized in Figure 38. The tested patterns are composed of 15×15 elements; it corresponds to the size of matrix in the figures. The values in matrix represent the average error occurrence for the corresponding pattern element (in hundredths of percent).

18	20	18	17	22	12	21	12	23	24	22	21	11	14	23
24	21	18	24	20	21	17	12	20	15	16	19	14	22	25
20	24	17	23	21	16	15	19	24	17	21	17	28	19	21
21	16	20	18	21	23	13	22	20	14	21	17	13	15	17
16	16	21	22	13	20	16	28	21	22	18	20	18	19	21
21	21	16	12	14	29	14	20	18	24	24	20	14	17	16
22	25	17	24	20	17	15	19	18	19	21	24	15	16	21
13	18	13	17	21	18	18	14	23	14	14	14	18	30	22
20	18	12	21	27	20	12	23	15	13	20	20	22	20	18
21	17	24	22	17	19	20	12	21	18	14	15	17	17	22
19	27	12	17	20	18	19	13	19	15	11	23	20	17	21
20	17	16	20	14	18	22	21	17	15	20	17	17	20	22
17	18	16	20	21	20	15	18	18	15	20	18	19	16	17
23	20	17	16	19	23	19	18	25	16	22	22	13	21	15
21	19	15	14	20	29	15	16	21	21	21	13	23	15	12

Figure 37: The average error occurrence for a given pattern element (hundredths of percent): the HAM1 model with the capacity coefficient $c = 0.7$ and the pattern rate $r = 1$

9	7	7	7	11	11	12	2	5	3	8	11	8	5	10
9	5	4	6	6	11	4	6	7	13	10	10	12	9	6
5	7	5	6	9	5	5	7	12	9	9	8	17	6	9
6	4	9	9	6	7	8	5	5	4	10	7	4	10	10
7	8	7	9	12	5	7	7	6	8	8	8	6	6	7
6	11	5	2	4	5	3	6	9	8	4	6	4	9	9
10	9	8	9	5	7	5	10	9	9	5	8	5	8	6
6	6	7	8	4	6	9	3	7	5	13	5	6	11	8
3	6	8	8	6	6	11	8	8	2	3	6	6	5	7
4	9	7	5	9	9	8	4	4	7	8	7	4	7	5
7	9	5	8	10	6	8	11	10	6	9	8	9	5	2
10	9	7	8	3	9	7	12	5	9	4	5	11	4	8
8	4	4	7	5	5	4	7	5	4	9	6	11	7	9
7	5	2	9	2	10	10	6	6	10	7	8	5	8	2
6	6	4	5	14	16	8	5	5	10	10	8	9	4	4

Figure 38: The average error occurrence for a given pattern element (hundredths of percent): the HAM2 model with the capacity coefficient $c = 0.7$ and the pattern rate $r = 1$

Since the processed patterns are randomly generated, there is no position in a pattern where the error occurrence differs greatly. The erroneously recalled el-

ements are distributed almost uniformly over the whole area of patterns. Small fluctuation refers to pattern structure (even if the patterns are randomly generated). The experiments are carried out for the HAM models with the capacity coefficient $c = 0.7$ and the pattern rate $r = 1$. The HAM models with other values of parameters have been also tested and the results are very similar.

The higher number of patterns recalled incorrectly by the HAM1 model induces the higher error in the recalled pattern elements (in comparison with the HAM2 model).

7.3.3 Comparison with the standard associative memory

In the previous chapter, we have compared our proposed HAM1 and HAM2 models with respect to the storage ability. Now, we perform the same experiments with the model of the standard associative memory (AM) [30]. We compare the results obtained from the standard associative memory with the results coming from our HAM models.

The theoretical storage capacity of the standard associative memory is 0.15 [30]. Thus, the theoretical maximum number of patterns that can be stored in the standard associative memory is $0.15 \cdot n$ when the patterns are (almost) orthogonal. Our experiments are focused on processing larger numbers of patterns. We deal with $m = 100$ patterns in every data set; each pattern of the dimension $n = 225$. Since the tested patterns are randomly generated, the patterns do not have to be necessarily orthogonal.

In the first experiment, we measure the ratio Y_i between the number of stored patterns recalled correctly and the total number of stored patterns in every data set D_i ($1 \leq i \leq q$, $q = 100$). We calculate the sample mean $\overline{Y}_q$ (Equation 52). From the experimental results, we get $Y_i = 0$ for every data set D_i. In means that the standard associative memory does not recall correctly any of the stored patterns. The sample mean $\overline{Y}_q$ is zero for the standard associative memory.

The theoretical limit for the number of 225−dimensional patterns to be stored in the memory is

$$max = 0.15 \cdot n = 0.15 \cdot 225 \doteq 34. \tag{64}$$

Every data set consists of $m = 100$ patterns to be stored in the model. Since the number of stored patterns exceeds the theoretical limit of the standard associative memory, the standard associative memory built in our test does not recall any of the stored patterns without any error. Our experimental results are in the accordance with the theory.

In the second experiment, we analyze the distribution of the error in the patterns recalled incorrectly. The histogram of the error in the patterns recalled incorrectly is shown in Figure 39. The horizontal axis represents the number of erroneously recalled elements in a pattern (in percents). The vertical axis shows the number of patterns (in percents) that are recalled with a given error. Since no pattern is recalled correctly by the standard associative memory, all patterns are depicted in the figure.

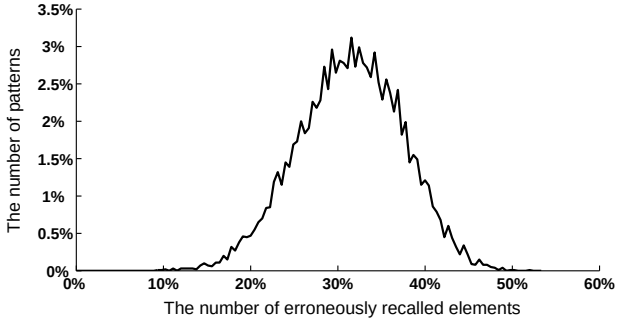


Figure 39: The histogram of the error in the patterns recalled incorrectly by the standard associative memory

The corresponding results for the HAM1 and HAM2 models are summarized in Figure 35 and Figure 36, respectively. The HAM models are able to recall correctly majority part of patterns. The average error in the incorrectly recalled patterns does not exceed 2%. On the other side, the standard associative memory recalls incorrectly 100% patterns. The average error in one pattern is higher $(10\% - 50\%)$.

31	31	32	31	32	31	32	31	32	31	31	31	32	31	32
31	31	32	32	31	31	31	33	31	31	31	32	31	31	31
32	32	31	32	31	32	32	32	31	31	32	31	31	32	31
31	31	31	32	31	32	32	31	31	31	31	31	31	32	32
32	32	31	31	31	32	31	32	31	31	32	31	32	31	32
31	32	31	32	31	32	32	31	31	32	32	32	31	32	31
31	31	32	31	31	32	31	32	32	31	32	31	32	31	31
31	31	32	31	31	31	31	32	32	31	30	31	31	32	32
31	31	32	31	32	31	32	31	31	31	31	32	32	31	32
31	32	31	31	32	31	31	31	31	31	32	31	31	32	31
31	31	32	32	32	32	31	31	31	31	31	31	32	31	31
31	31	31	31	32	31	31	31	32	32	31	32	31	32	31
32	31	32	32	31	31	31	32	31	31	31	31	31	31	32
32	32	31	31	32	31	32	32	32	31	32	31	31	32	32
31	32	32	31	32	32	32	32	31	32	32	32	32	32	31

Figure 40: The average error occurrence for a given pattern element (in percents) in the standard associative memory

In the next experiment, we analyze the magnitude of the error in pattern elements. The results for the standard associative memory are summarized in Figure 40. For the comparison, the equivalent results proceeded by the HAM models are in Figure 37 and Figure 38. In Figure 40, the values in matrix represent the average error occurrence for a corresponding pattern element in

percents (in the HAM1 and HAM2 models the values correspond to hundredths of percent).

As in the HAM model results, the erroneously recalled elements are distributed uniformly over the whole area of a pattern. Experiments with the standard associative memory shows that every pattern element is recalled erroneously in one third of the patterns.

7.3.4 Comparison with the cascade associative memory

The cascade associative memories (CASM) have been proposed by Hirahara et al. Since our HAM models are based on the concept of the CASM model, we conduct the same experiments (as in the previous two sections) also for the CASM model. We focus on the CASM1 model [26] as this model of cascade associative memory is the most similar in its structure to our HAM models.

The CASM1 model is composed of two autoassociative memories. The first autoassociative memory (AM1) and the second one (AM2) store the ancestors and the corresponding difference patterns, respectively. The detailed description of the CASM1 model can be found in Chapter 4.3.3. Since AM1 and AM2 are autoassociative memories, the theoretical maximum number of patterns stored in each of them is $0.15 \cdot n$, when the patterns to be stored in it are (almost) orthogonal [26]. In our experiments, the dimension of patterns is $n = 225$, so the theoretical maximum number of patterns of AM1 and AM2 can be expressed as

$$max_{AM1} = max_{AM2} = 0.15 \cdot n = 0.15 \cdot 225 \doteq 34. \tag{65}$$

In the first experiment, we measure the ratio Y_i between the number of stored patterns recalled correctly and the total number of stored patterns in every data set D_i ($1 \leq i \leq q$, $q = 100$) with $m = 100$ randomly generated patterns (Section 7.1.2). The experimental simulations for the CASM1 model are repeated for four different pattern rates: $r \sim 1/5$, $r = 1/3$, $r \sim 1/2$ and $r = 1$. [3] The pattern rate r determines a ratio between the number of ancestor *anc* and descendants *desc*. The sample mean $\overline{Y}_q$ is calculated according to Equation 52 and the results are summarized in Table 5.

The CASM1 model: The mean			
$r \sim 1/5$	$r = 1/3$	$r \sim 1/2$	$r = 1$
$\overline{Y}_q$	$\overline{Y}_q$	$\overline{Y}_q$	$\overline{Y}_q$
$\langle Y_{min,q}, Y_{max,q} \rangle$	$\langle Y_{min,q}, Y_{max,q} \rangle$	$\langle Y_{min,q}, Y_{max,q} \rangle$	$\langle Y_{min,q}, Y_{max,q} \rangle$
0.1698	0.2373	0.2407	0.0612
$\langle 0.1695, 0.1701 \rangle$	$\langle 0.2351, 0.2395 \rangle$	$\langle 0.2345, 0.2469 \rangle$	$\langle 0.0565, 0.066 \rangle$

Table 5: The storage ability of the CASM1 model: the sample mean $\overline{Y}_q$ and the confidence interval $\langle Y_{min,q}, Y_{max,q} \rangle$ for various pattern rates r (the confidence level $\alpha = 0.05$ and the number of experiments $q = 100$)

[3] The symbols $r \sim 1/5$ and $r \sim 1/2$ denote the pattern rates $r = 17/83$ and $r = 33/67$, respectively

For $r = 1$, both associative memories AM1 and AM2 of the CASM1 model are overloaded as the number of ancestors and descendants (difference patterns) exceeds the theoretical limit max_{AM1} and max_{AM2}, respectively. In this case, only 6% patterns are recalled correctly by the CASM1 model (the last column in Table 5).

The decreasing pattern rate r corresponds to the decreasing number of ancestors *anc* and the increasing number of descendants *desc*. If $r \sim 1/2$, AM1 is fully loaded and several ancestors are not recalled correctly (because of possible correlation between ancestors). For $r < 1/2$, the number of ancestors is lower than the theoretical limit max_{AM1} and AM1 recalls correctly most of the ancestors. At the same time, the number of descendants *desc* exceeds the theoretical limit max_{AM2} and no descendant is recalled correctly by AM2.

The number of patterns recalled correctly by the CASM1 model corresponds mainly to the number of ancestors recalled correctly. Our experimental results with the CASM1 model (Table 5) are in accordance with the theory. The storage ability of the CASM1 model is not very robust with respect to storing larger amounts of data (in comparison to the HAM models).

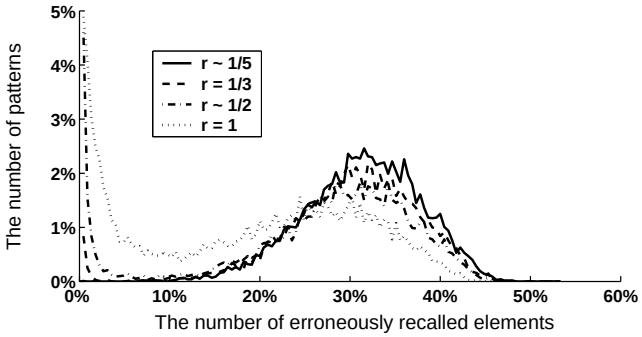


Figure 41: The histogram of the error in the patterns recalled incorrectly by the CASM1 model for four different pattern rates: $r \sim 1/5$ (the solid line), $r = 1/3$ (the dashed line), $r \sim 1/2$ (the dash-dotted line), and $r = 1$ (the dotted line)

In the next experiments, we analyze the distribution of the error in patterns recalled incorrectly. The histogram of the error is depicted in Figure 41. The horizontal axis denotes the number of erroneously recalled elements in a pattern (in percents). The vertical axis shows the relative number of patterns recalled with a given error (in percents). The patterns recalled with zero error are not considered in the graph. The shape of the curves is similar to the histogram of the error in the standard associative memory (Figure 39). The differences can be found in cases where the error is less than 10%.

For $m = 100$ stored patterns, the standard associative memory is "greatly" overloaded. Patterns recalled by the memory usually correspond to spurious patterns generated by all stored patterns. The error in such patterns is expected

to be relatively high. The similar situation is detected in the CASM1 model with $r < 1/2$. The AM2 is "greatly" overloaded and the CASM1 model recalls descendants with a relatively high error. The AM1 is not fully loaded and most ancestors are recalled with zero error.

For the CASM1 models with the higher values of the pattern rate ($r \geq 1/2$), the AM1 is fully loaded (or even slightly overloaded). In these cases, two (or more) similar ancestors can fall to the same basin of attraction corresponding to the spurious pattern generated by the similar ancestors. It probably causes the increasing number of patterns recalled with a small error.

In the third experiment, the magnitude of the error in pattern elements is analyzed. The average error occurrence for the pattern elements is summarized in Table 6.

The average error occurrence of pattern elements		
	min	max
AM	30.29%	32.63%
CASM1, $r \sim 1/5$	24.91%	26.83%
CASM1, $r = 1/3$	21.93%	23.76%
CASM1, $r \sim 1/2$	18.97%	20.78%
CASM1, $r = 1$	16.81%	18.31%
HAM1, $r = 1$, $c = 0.7$	0.11%	0.29%
HAM2, $r = 1$, $c = 0.7$	0%	0.17%

Table 6: The minimum and maximum average error occurrence of pattern elements in various associative memory models: the standard associative memory (AM), the CASM1 model with various pattern rates ($r \sim 1/5$, $r = 1/3$, $r \sim 1/2$, $r = 1$), the HAM1 model ($r = 1, c = 0.7$) and the HAM2 model ($r = 1, c = 0.7$)

In the standard associative memory, every pattern element is recalled erroneously in 30% patterns. Every pattern element is recalled erroneously by the CASM1 model approximately in 20% patterns. Although the CASM1 model has been proposed to deal with correlated patterns, the performance of the model is sensitive to the number of stored patterns. The average error occurrence of pattern elements in our HAM models is very small (less than 1%). It is caused by the fact that the most of patterns are recalled by the HAM models with zero error (Figure 33, Figure 34) or very small error (Figure 35, Figure 36).

7.4 Processing of incomplete patterns

An application of the proposed HAM models for solving the real problems (e.g. to help to move in an environment) requires a robust recall of presented (possibly incomplete) patterns. In the previous chapter, we consider the ability of the model to recall the stored patterns. In this chapter, we focus on the so-called recall ability of our HAM1 and HAM2 models (in comparison with the

standard associative memory and the CASM1 model). The recall ability measures the ability of the models to recall incomplete patterns (e.g. the images). The incomplete pattern elements contain the information that differs from the information encoded in original patterns.

Definition 7.4 *Let $\vec{x} = (x_1, \ldots, x_n)$ be $n-$dimensional pattern.*

- *The $n-$ dimensional pattern $\vec{y} = (y_1, \ldots, y_n)$ is called the incomplete pattern corresponding to the pattern $\vec{x}$ if $y_i \neq x_i$ for at least one pattern element i $(1 \leq i \leq n)$.*
- *The pattern element y_i $(1 \leq i \leq n)$ of the incomplete pattern $\vec{y}$ is called the unknown element if $y_i \neq x_i$.*
- *The pattern element y_i $(1 \leq i \leq n)$ of the incomplete pattern $\vec{y}$ is called the known element if $y_i = x_i$.*

All unknown elements of the incomplete pattern form the so-called unknown area. The known elements of the incomplete pattern form the so-called known area. Our experiments are focused on processing the incomplete patterns with the "continuous" unknown area.

The information encoded in the unknown elements is expected "not to be set". In practice, the unknown elements are filled with the predefined value (e.g. the average value of the elements in the original pattern or the average value of the known pattern elements).

During our experiments, the patterns from every data set are stored in the model according to the corresponding training algorithm. Afterwards, the incomplete patterns are generated and they are presented to the model to be recalled. The incomplete patterns contain the unknown area that should be "filled" during the recall process. We detect the ability of the models to recall the unknown area of the presented incomplete patterns. The magnitude of the error in the recalled patterns determines the recall ability of the models.

Definition 7.5 *Let AM be a model of an associative memory. Let $\vec{x}$ be n-dimensional incomplete pattern. Let $\vec{y}$ be n-dimensional pattern recalled by the model AM after presenting the input pattern $\vec{x}$. The incomplete pattern $\vec{x}$ is recalled with error k $(0 < k \leq n)$, if the recalled pattern $\vec{y}$ varies from the presented pattern $\vec{x}$ in k elements located in the unknown area of $\vec{x}$.*

We evaluate the number of the incomplete patterns recalled correctly in every data set. Then, we focus on the incomplete patterns recalled with an error and we analyze the magnitude of the error in every pattern.

7.4.1 Generation of incomplete patterns

For experiments, we create three groups of incomplete patterns:

- with 13% unknown elements (the shift by 1),
- with 25% unknown elements (the shift by 2),
- with 36% unknown elements (the shift by 3).

Each of the groups corresponds to the diagonal shift of the patterns (e.g. the images) by 1, 2, or 3 points, respectively. Hence, every pattern element is shifted by 1, 2 or 3 points in the diagonal direction. We distinguish four types of the diagonal directions:

- northwest $\nwarrow$,
- northeast $\nearrow$,
- southwest $\swarrow$,
- southeast $\searrow$.

The location of the unknown area in the incomplete pattern is opposite to the diagonal direction. Four types of incomplete patterns according to the different diagonal directions are shown in Figure 42. The unknown area is drawn in black. Since the experiments focused on the incomplete patterns are relatively time-consuming, we restrict our simulations to four selected directions. In practice, other directions are also possible. The size of the unknown area (i.e. the number of unknown elements in a pattern) is characterized by the size of the shift.

(a) (b) (c) (d)

Figure 42: The four types of incomplete patterns according to the diagonal direction: (a) southwest direction, (b) southeast direction, (c) northeast direction, (d) northwest direction. The unknown area is shown in black. The "known" part of the pattern is in white.

During generation of the incomplete patterns, twelve incomplete patterns are generated for every pattern from the data set. We get three different groups of incomplete patterns according to the size of the shift (the shift by 1, the shift by 2 and the shift by 3). In every group, there are four types of incomplete patterns with respect to the different diagonal directions (northwest, northeast, southwest, southeast). The generated incomplete patterns are presented to the model to be recalled. The unknown area of the pattern is filled during the recall process. As we know the original pattern, we can determine the error in the recalled pattern. The error is measured in the unknown area (already filled) of the recalled pattern.

7.4.2 Comparison of the HAM1 and the HAM2 model

We analyze the ability of our HAM1 and HAM2 models to recall the incomplete patterns. First, we focus on the recall of incomplete patterns with respect to the different size of the unknown area. The size of the unknown area is characterized by the size of the diagonal shift.

In our experiments, we distinguish twelve types of incomplete patterns: three different size of the shift (the shift by 1, the shift by 2 and the shift by 3) and

four different diagonal directions ($\searrow$, $\nearrow$, $\swarrow$ and $\nwarrow$). The experiments are performed for every diagonal direction and every size of the shift separately. The results are averaged for the corresponding size of the shift.

As the number of experiments to be performed is relatively high, we decide to perform the experiments for the HAM1 and HAM2 models with the following parameters [4]:

- the pattern rate $r = 1/3$ and the capacity coefficient $c = 0.5$
- the pattern rate $r \sim 1/2$ and the capacity coefficient $c = 0.8$
- the pattern rate $r = 1$ and the capacity coefficient $c = 0.7$

For the pattern rate r, the capacity coefficient c is chosen in such a way that the differences between the storage ability of the HAM1 model and the HAM2 model (with the corresponding parameters r and c) are maximum. For $r \sim 1/5$, the number of ancestors is very small and the influence of the tree structure of the HAM2 model is relatively small. The results for the HAM1 and HAM2 models with the pattern rate $r \sim 1/5$ and the capacity coefficient $c = 0.4$ are shown in Appendix of the thesis (Table 12, Table 13, Table 14) [5].

In every data set, we measure the ratio between the number of incomplete patterns recalled correctly and the total number of incomplete patterns in the data set. The recall ability of the model corresponds to the sample mean that is calculated from all data set results (Equation 52). It shows the relative number of incomplete patterns recalled correctly by the model.

Unfortunately, we expect that the recall ability of the model decreases with the increasing number of unknown elements in incomplete patterns. In practice, it is not necessary to recall the incomplete patterns without any error and a small error is also acceptable (the so-called acceptable error). The acceptable error e_{accept} corresponds to the maximum number of elements in a pattern that can be recalled erroneously. We define that the incomplete pattern is recalled correctly if the number of erroneously recalled elements in the recalled pattern is less than or equal to the acceptable error.

We show the relative number of incomplete patterns recalled without any error and also the relative number of incomplete patterns recalled correctly with respect to the acceptable error. The results of the HAM1 and HAM2 models processing the incomplete patterns with the diagonal shift by 1 point are summarized in Table 7. Here, the incomplete patterns contain 13% unknown elements. Table 8 shows the recall ability of the HAM1 and HAM2 models dealing with the incomplete patterns containing 25% unknown elements (i.e. the diagonal shift by 2 points). The ability of the HAM1 and HAM2 models to recall the incomplete patterns containing 36% unknown elements (i.e. the diagonal shift by 3 points) are summarized in Table 9. The results are in percents.

The structure of Table 7, Table 8 and Table 9 is very similar. The tables show the relative number of incomplete patterns, in which the error is less than or equal to the corresponding acceptable error level. The first column represents

[4] The symbol $r \sim 1/2$ denotes the pattern rate $r = 33/67$

[5] The symbol $r \sim 1/5$ denotes the pattern rate $r = 17/83$

the acceptable error level e_{accept}. The next three columns show the results of the HAM1 model. The last three columns correspond to the results of the HAM2 model. The parameters r and c of the HAM1 and HAM2 models are inscribed in every column. The small numbers in the square brackets in the third row of the table show the storage ability $\overline{Y}_{100}$ of the model with the given parameters (to make comparison with the recall ability). If $e_{accept} = 0$, the recall ability covers only the incomplete patterns recalled without any error. For $e_{accept} > 0$, the incomplete patterns recalled with a small error are also considered.

The shift by 1						
	The HAM1 model			The HAM2 model		
The acceptable error e_{accept}	$r = 1/3$ $c = 0.5$ [97.15%]	$r \sim 1/2$ $c = 0.8$ [84.37%]	$r = 1$ $c = 0.7$ [89.92%]	$r = 1/3$ $c = 0.5$ [97.47%]	$r \sim 1/2$ $c = 0.8$ [90.02%]	$r = 1$ $c = 0.7$ [95.16%]
$\leq 0\%$	41.70%	41.06%	52.97%	41.68%	41.85%	53.53%
$\leq 1\%$	45.83%	46.67%	57.82%	45.90%	47.51%	58.52%
$\leq 2\%$	50.95%	52.41%	62.58%	51.02%	53.57%	63.58%
$\leq 3\%$	56.06%	57.91%	66.57%	56.04%	59.48%	67.89%
$\leq 4\%$	63.30%	65.77%	71.65%	63.30%	67.45%	72.88%
$\leq 5\%$	69.04%	71.50%	75.55%	68.71%	72.67%	76.40%
$\leq 6\%$	76.89%	79.17%	81.68%	76.25%	79.61%	82.12%
$\leq 7\%$	86.85%	88.33%	89.57%	86.21%	88.06%	89.44%
$\leq 8\%$	97.35%	97.56%	97.82%	97.02%	97.33%	97.69%
$\leq 9\%$	99.53%	99.50%	99.56%	99.44%	99.47%	99.58%
$\leq 10\%$	99.96%	99.95%	99.96%	99.94%	99.94%	99.96%
$\leq 11\%$	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%

Table 7: The recall ability of the HAM1 and HAM2 models for the diagonal shift by 1 point: the relative number of incomplete patterns recalled with the error less than or equal to the acceptable error e_{accept}

The shift by 2						
	The HAM1 model			The HAM2 model		
The acceptable error e_{accept}	$r = 1/3$ $c = 0.5$ [97.15%]	$r \sim 1/2$ $c = 0.8$ [84.37%]	$r = 1$ $c = 0.7$ [89.92%]	$r = 1/3$ $c = 0.5$ [97.47%]	$r \sim 1/2$ $c = 0.8$ [90.02%]	$r = 1$ $c = 0.7$ [95.16%]
$\leq 0\%$	36.04%	35.88%	49.70%	35.98%	36.87%	50.43%
$\leq 1\%$	37.75%	39.28%	52.54%	37.85%	39.98%	53.15%
$\leq 2\%$	39.41%	41.12%	54.14%	39.46%	41.95%	54.92%
$\leq 3\%$	41.29%	43.26%	55.97%	41.33%	44.14%	57.08%
$\leq 4\%$	44.53%	46.65%	58.87%	44.59%	47.88%	60.23%
$\leq 5\%$	46.95%	49.12%	60.62%	47.02%	50.62%	62.29%
$\leq 6\%$	49.41%	51.63%	62.39%	49.47%	53.54%	64.40%
$\leq 7\%$	51.90%	54.18%	64.08%	52.04%	56.32%	66.26%
$\leq 8\%$	55.30%	57.96%	66.48%	55.43%	60.08%	68.62%
$\leq 9\%$	57.64%	60.54%	68.13%	57.56%	62.46%	70.08%
$\leq 10\%$	60.50%	63.62%	70.24%	60.30%	65.18%	71.81%
$\leq 11\%$	64.93%	68.13%	73.55%	64.39%	69.05%	74.45%

Table 8: The recall ability of the HAM1 and HAM2 models for the diagonal shift by 2 points: the relative number of incomplete patterns recalled with the error less than or equal to the acceptable error e_{accept}

The shift by 3						
	The HAM1 model			The HAM2 model		
The acceptable error e_{accept}	$r = 1/3$ $c = 0.5$ [97.15%]	$r \sim 1/2$ $c = 0.8$ [84.37%]	$r = 1$ $c = 0.7$ [89.92%]	$r = 1/3$ $c = 0.5$ [97.47%]	$r \sim 1/2$ $c = 0.8$ [90.02%]	$r = 1$ $c = 0.7$ [95.16%]
$\leq 0\%$	32.63%	32.24%	46.90%	32.74%	33.48%	47.69%
$\leq 1\%$	33.98%	35.94%	50.52%	34.09%	36.83%	51.10%
$\leq 2\%$	34.75%	37.17%	51.49%	34.85%	38.05%	52.09%
$\leq 3\%$	35.70%	38.15%	52.28%	35.79%	39.06%	53.05%
$\leq 4\%$	37.30%	39.73%	53.61%	37.42%	40.75%	54.66%
$\leq 5\%$	38.39%	40.79%	54.63%	38.51%	41.98%	55.83%
$\leq 6\%$	39.69%	42.04%	55.65%	39.72%	43.39%	57.02%
$\leq 7\%$	40.96%	43.33%	56.65%	41.04%	44.93%	58.24%
$\leq 8\%$	43.40%	45.52%	58.25%	43.44%	47.38%	60.07%
$\leq 9\%$	45.02%	47.04%	59.31%	45.10%	49.01%	61.42%
$\leq 10\%$	46.60%	48.72%	60.40%	46.59%	50.72%	62.63%
$\leq 11\%$	48.00%	50.35%	61.56%	48.08%	52.42%	63.89%

Table 9: The recall ability of the HAM1 and HAM2 models for the diagonal shift by 3 points: the relative number of incomplete patterns recalled with the error less than or equal to the acceptable error e_{accept}

The results in Table 7, Table 8 and Table 9 show that the recall abilities of the HAM2 model are higher than the recall abilities of the HAM1 model but the differences are just minor. With increasing error acceptance e_{accept} the number of patterns recalled correctly (i.e. the error is less than or equal to the acceptable error) increases. When the unknown area is small and $e_{accept} \leq 5\%$, the HAM models have relatively robust recall abilities (70% or even more) with respect to the incomplete pattern recall.

If the unknown area is small, all incomplete patterns can be recalled correctly for the acceptable error $e_{accept} \leq 11\%$. As the unknown area enlarges, the number of incomplete patterns recalled correctly decreases rapidly. When the incomplete patterns are shifted by 3 points, more than one third of all pattern elements are unknown and the recall abilities of the HAM models are relatively low $(30\% - 60\%)$. With the enlarging unknown area, the incomplete patterns represent for the model different patterns than the patterns stored originally in the model. During the recall, the incomplete patterns that differ from their patterns stored originally in the model can easily fall into the basin of attraction of other stored patterns. Thus, the recalled patterns do not correspond to the presented incomplete patterns and they are considered to be recalled incorrectly.

As the pattern rate r is increased, the recall abilities of the HAM models are improved. The ancestor recall is crucial for the whole process of the incomplete pattern recall. If the ancestor is recalled incorrectly by the first layer of the model, the output of the model is often incorrect, too. When the corresponding ancestor is recalled incorrectly, the incomplete pattern is classified into the wrong group of stored patterns. Hence, the recalled pattern does not coincide with the presented incomplete pattern.

During the recall of stored patterns, the information necessary to recall the corresponding ancestors is available. Hence, the ancestors are probably recalled correctly. During the incomplete pattern recall, the part of the information is missing and it decreases the probability of the correct ancestor recall. As the pattern rate r rises, the number of local associative memories of the first layer is increased. It induces the low number of ancestors recalled incorrectly. Thus, the HAM models with increasing pattern rates r give the better results with respect to the incomplete pattern recall.

In next experiments, we analyze the incomplete patterns that are recalled with some error. We construct the histogram of the error in the recalled incomplete patterns. It means that we measure the number of incomplete patterns that are recalled with the same error. The histogram of the error in the incomplete patterns recalled by the HAM1 model with the pattern rate $r = 1$ and the capacity coefficient $c = 0.7$ is depicted in Figure 43. The same experiment is performed also by the HAM2 model with the same parameters and the results are shown in Figure 44.

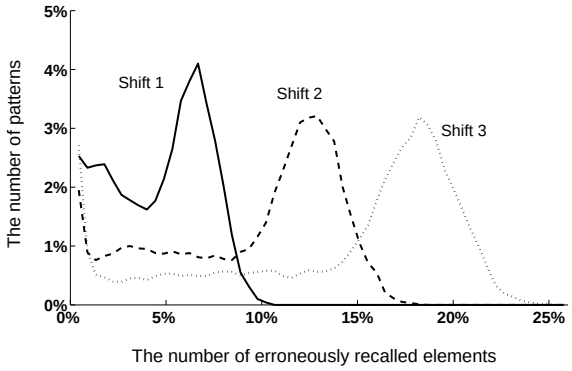


Figure 43: The histogram of the error in the incomplete patterns recalled by the HAM1 model ($c = 0.7$ and $r = 1$) for three types of the diagonal shift: the shift by 1 (the solid line), the shift by 2 (the dashed line) and the shift by 3 (the dotted line)

The horizontal axis in Figure 43 and Figure 44 corresponds to the number of erroneously recalled elements in the unknown area of incomplete patterns (in percents). The vertical axis denotes the number of incomplete patterns recalled with the given error in relation to the total number of incomplete patterns. The incomplete patterns recalled without any error are not depicted.

The shapes of the graphs for the HAM1 and the HAM2 model are very similar. The maxima are located in the same positions for both HAM models. The maximum values are lower for the HAM2 but the differences are diminutive. The number of incomplete patterns recalled with error $> 4\%$ is lower for the HAM2 model (in comparison with the HAM1 model).

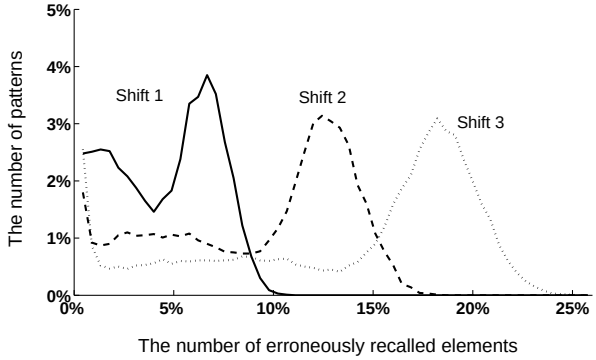


Figure 44: The histogram of the error in the incomplete patterns recalled by the HAM2 model ($c = 0.7$ and $r = 1$) for three types of the diagonal shift: the shift by 1 (the solid line), the shift by 2 (the dashed line) and the shift by 3 (the dotted line)

The experiments are also performed for the HAM1 and HAM2 models with parameters $c \sim 1/2$, $r = 0.8$ and $c = 1/3$, $r = 0.5$ and $c \sim 1/5$, $r = 0.4$. The results are depicted in Appendix of the thesis (Figure 73 - Figure 78). For the tested HAM1 and HAM2 models, the maxima of the histograms are located at similar positions: the position of 7% for the shift 1, 12% for the shift 2 and 18% for the shift 3. The incomplete patterns with the shift 1 (or 2 or 3) contains 13% (or 25% or 36%) unknown elements. Hence, the histogram maxima correspond to the incomplete patterns, in which one half of the unknown elements are recalled erroneously. The encoding of the incomplete pattern (because of the unknown elements) represents for the model partially different input pattern than that of the pattern stored originally in the model. During the recall, the incomplete patterns can easily fall into the basin of attraction of the different patterns. The falling into other basins of attraction leads to the increasing error in the recalled incomplete patterns.

For all tested HAM models, the shapes of the histograms are very similar (Figure 43, Figure 44, Figure 73 - Figure 78). Since the recall abilities of the HAM models with $r < 1$ are lower than the recall abilities of the HAM models with $r = 1$, the total number of incomplete patterns recalled with non-zero error is higher for the HAM models with $r < 1$. It corresponds to the larger area under the histogram curve for the HAM models with $r < 1$.

The recall abilities of the HAM models with the parameters $r \sim 1/2$ and $r = 1/3$ are similar, but the maximum values of the histograms are slightly different. Moreover, the number of incomplete patterns recalled with the error below 2% is lower for the HAM models with $r = 1/3$ than that for the HAM models with $r > 1/3$. It means the recalled incomplete patterns contain more erroneously recalled elements.

7.4.3 Comparison with the standard associative memory

Here, we compare our HAM models with the model of the standard associative memory [30]. We perform the same experiments as described in the previous section 7.4.2 to determine the ability of the standard associative memory to recall the incomplete patterns.

First, we measure the number of incomplete patterns recalled without any error. Because of the model incapability to recall the stored patterns without any error, the standard associative memory is not able to recall the incomplete patterns [6] without any error either.

In the next experiment, we analyze the number of incomplete patterns, in which the error is less than or equal to the acceptable error. These incomplete patterns are considered to be recalled correctly with respect to the acceptable error level. Figure 45 shows the recall ability of the standard associative memory in comparison with the HAM1 and HAM2 models for the incomplete patterns with 13% unknown elements (the diagonal shift by 1). The recall results for the incomplete patterns corresponding to the diagonal shift by 2 and the diagonal shift by 3 are summarized in Figure 46 and Figure 47, respectively.

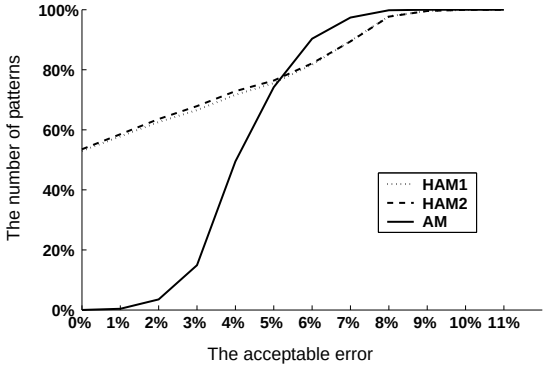


Figure 45: The relative number of incomplete patterns recalled by the standard associative memory (the solid line), the HAM1 model with $r = 1$, $c = 0.7$ (the dotted line) and the HAM2 model with $r = 1$, $c = 0.7$ (the dashed line) where the error is less than or equal to the acceptable error. The unknown area corresponds to the diagonal shit by 1.

[6] The incomplete patterns are generated based on the stored pattern

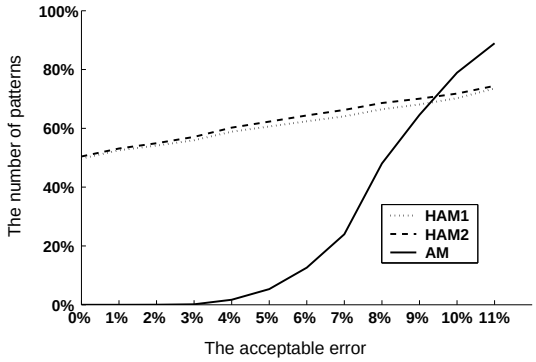


Figure 46: The relative number of incomplete patterns recalled by the standard associative memory (the solid line), the HAM1 model with $r = 1$, $c = 0.7$ (the dotted line) and the HAM2 model with $r = 1$, $c = 0.7$ (the dashed line) where the error is less than or equal to the acceptable error. The unknown area corresponds to the diagonal shit by 2.

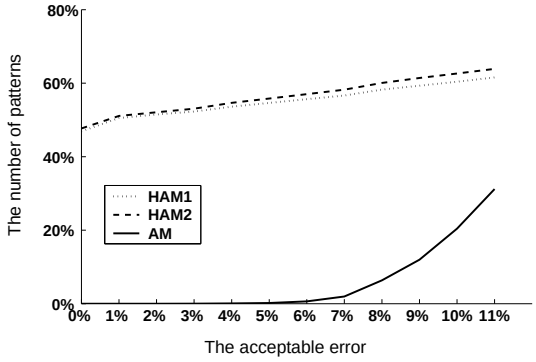


Figure 47: The relative number of incomplete patterns recalled by the standard associative memory (the solid line), the HAM1 model with $r = 1$, $c = 0.7$ (the dotted line) and the HAM2 model with $r = 1$, $c = 0.7$ (the dashed line) where the error is less than or equal to the acceptable error. The unknown area corresponds to the diagonal shit by 3.

The horizontal axis in Figure 45, Figure 46 and Figure 47 corresponds to the acceptable error e_{accept}. The vertical axis shows the relative number of incomplete patterns, in which the recall error is is less than or equal to the corresponding acceptable error level. The results are obtained from the HAM1 and the HAM2 models with the capacity coefficient $c = 0.7$ and the pattern rate $r = 1$.

In the HAM models, more than 50% incomplete patterns with the diagonal shift by 1 are recalled without any error. More than 60% incomplete patterns are recalled with the error below 2%. When the error is high, we expect (due to the robust storage ability) that another pattern is recalled by the model. It means that the recalled pattern does not correspond to the original pattern, from which the incomplete pattern has been generated. The incomplete pattern falls into the basin of attraction of the different ancestor and thus the first layer is likely to recall another stored ancestor. The recalled ancestor combined with the incomplete pattern is presented to the second layer of the HAM models. When the "incorrect" ancestor is recalled, the output pattern of the HAM model can easily correspond the "incorrect" stored pattern.

For $e_{accept} \leq 4\%$, the recall abilities of our HAM models are superior to the standard associative memory. When the acceptable error is low ($e_{accept} \leq 2\%$), the standard associative memory is not able to recall correctly almost any incomplete pattern. As the standard associative memory is overloaded, it recalls spurious patterns instead of the patterns stored originally in the model. The spurious pattern is generated by all patterns stored in the model. With the increasing error acceptance the recall ability of the standard associative memory grows rapidly. It could be implied by the fact that the Hamming distance between the recalled spurious pattern and the original pattern falls below the increasing acceptable error level. Hence, the recalled spurious pattern is considered to be recalled correctly.

When the error acceptance is greater than 6%, the standard associative memory shows the higher recall abilities than the HAM models for the incomplete patterns corresponding to the diagonal shift by 1. In our opinion, the standard associative memory recalls the spurious pattern that is "more similar" to the originally stored pattern than the pattern recalled by the HAM models. Thus, the pattern recalled by the standard associative memory is considered to be recalled correctly and the pattern recalled by the HAM models is not. The similarity is measured by the Hamming distance.

When the unknown area in the incomplete patterns is large, the recall ability of the standard associative memory is inferior to the HAM models. For $e_{accept} \leq 5\%$, the standard associative memory does not recall almost any incomplete pattern with the error less than or equal to the acceptable error.

We also analyze the incomplete patterns, in which at least one pattern element is recalled erroneously. We determine the number of incomplete patterns recalled with the corresponding error. The obtained results are summarized in the histogram graph (Figure 48).

The horizontal axis denotes the number of erroneously recalled elements in one incomplete pattern. The vertical axis shows the relative number of incom-

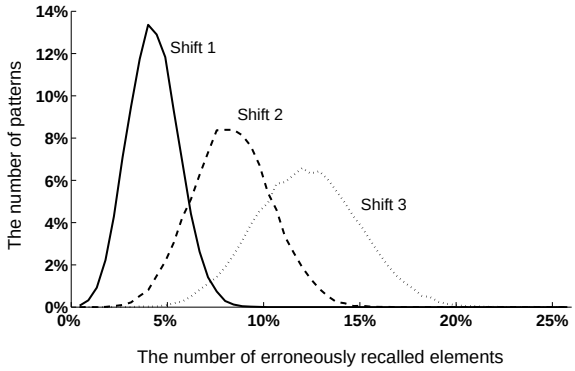


Figure 48: The histogram of the error in the incomplete patterns recalled by the standard associative memory for three types of the diagonal shift: the shift by 1 (the solid line), the shift by 2 (the dashed line) and the shift by 3 (the dotted line)

plete patterns recalled with the given error (in percents). As no incomplete pattern is recalled without any error by the standard associative memory, this information can be easily drawn in the figure, too.

The shape of the graph is similar to the corresponding graphs for the HAM models (Figure 43 and Figure 44). The pattern recalled by the standard associative memory is probably the spurious pattern because of the low storage ability of the model. On the other side, the pattern recalled by the HAM model could be another stored pattern. We can expect that the spurious pattern is "more similar" to the presented incomplete pattern than to another stored pattern. Hence, the "most frequent" error of the patterns recalled by the standard associative memory is lower than the error in the patterns recalled by the HAM models. It corresponds to the graph peaks in Figure 48 (the standard associative memory), Figure 43 (the HAM1 model) and Figure 44 (the HAM2 model).

The histogram maxima for the standard associative memory are positioned near the values of 4%, 8% and 12% for every tested diagonal shift. The peaks of the HAM model histograms are close to the values of 7%, 12% and 18% for the corresponding diagonal shifts. Although the shapes of the histograms are very similar, the vertical axes are scaled in different ways (for the HAM models and the standard associative memory). The maximum values of the standard associative memory histograms are higher than that of the HAM models. It shows the higher number of incomplete patterns recalled by the standard associative memory with the corresponding error. The area under the histogram curve is also larger for the standard associative memory as the total number of incomplete patterns recalled with any error is rapidly higher.

7.4.4 Comparison with the cascade associative memory

In this section, we conduct the experiments described in Section 7.4.2 for the cascade associative memory in order to compare the recall ability of this model with our HAM model. The recall ability measures the ability of the model to process incomplete patterns. Our experiments are focused on the CASM1 model [26] as it is the most similar in its structure to our HAM model.

First, we analyze the number of incomplete patterns, in which the recall error is less than or equal to the acceptable error. The acceptable error defines the maximum number of elements in a pattern that can be recalled erroneously. Figure 49 shows the recall ability of the CASM1 model in comparison with the standard associative memory (AM) and our HAM2 model in a case where incomplete patterns contain 13% unknown elements (the diagonal shift by 1). Simulations are repeated for the CASM model with four different pattern rates ($r = 1$, $r \sim 1/2$, $r = 1/3$, $r \sim 1/5$) [7]. The HAM1 results are not depicted as the differences between the HAM1 model and HAM2 model are only minor (Section 7.4.2). The similar experiments are performed also for the incomplete patterns corresponding to the diagonal shift by 2 and the diagonal shift by 3. The results are summarized in Figure 50 and Figure 51, respectively.

The horizontal axis in Figure 49, Figure 50 and Figure 51 shows the acceptable error level. The vertical axis measures the relative number of incomplete patterns recalled correctly (i.e. the error is not above the corresponding acceptable error).

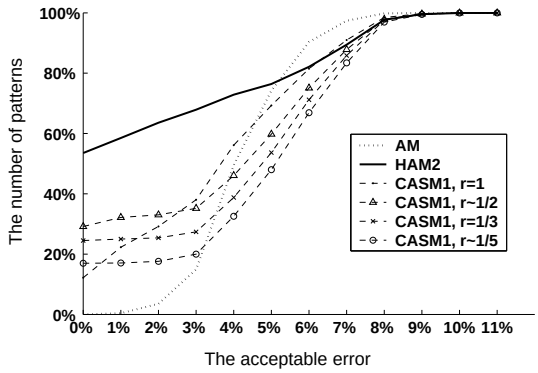


Figure 49: The relative number of incomplete patterns recalled by the standard associative memory AM (the dotted line), the HAM2 model with $r = 1$ and $c = 0.7$ (the solid line) and the CASM1 model with four different pattern rates r (the dashed lines) where the error is not above the acceptable error. The unknown area corresponds to the diagonal shit by 1.

[7] The symbols $r \sim 1/5$ and $r \sim 1/2$ denote the pattern rates $r = 17/83$ and $r = 33/67$, respectively

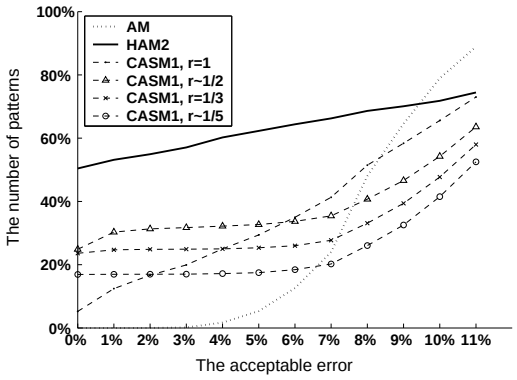


Figure 50: The relative number of incomplete patterns recalled by the standard associative memory AM (the dotted line), the HAM2 model with $r = 1$ and $c = 0.7$ (the solid line) and the CASM1 model with four different pattern rates r (the dashed lines) where the error is not above the acceptable error. The unknown area corresponds to the diagonal shit by 2.

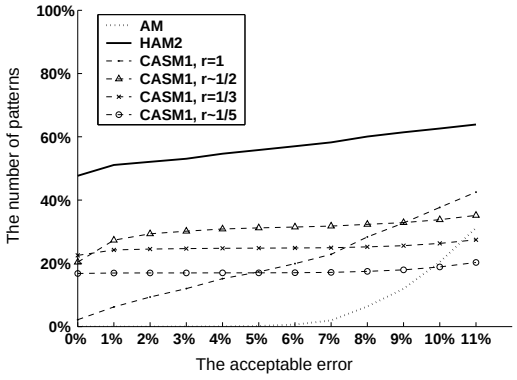


Figure 51: The relative number of incomplete patterns recalled by the standard associative memory AM (the dotted line), the HAM2 model with $r = 1$ and $c = 0.7$ (the solid line) and the CASM1 model with four different pattern rates r (the dashed lines) where the error is not above the acceptable error. The unknown area corresponds to the diagonal shit by 3.

For zero acceptable error, the recall ability of the CASM1 model with $r \sim 1/5$ is similar to its storage ability. Since the first associative memory AM1 of the CASM1 model is not fully loaded, the model is able to recall all stored ancestors without any error (even when the part of the ancestor is unknown). At the same time, none of descendants is recalled without any error as the second associative memory AM2 of the CASM1 model is overloaded.

The results show that the recall ability of the CASM1 model with $r > 1/5$ can be higher than the storage ability (Table 5) when the acceptable error is zero. This phenomenon is caused by the fact that the error is measured only in the unknown area of recalled (incomplete) patterns. The error is expected to increase if the error in recalled (incomplete) patterns would be measured over the whole area of patterns.

If the error acceptance is small, the recall abilities of the CASM1 model are superior to the standard associative memory model. Moreover, our HAM models are more robust than the CASM1 model with respect to processing the incomplete patterns. These results are in accordance with the structure of the discussed models.

If the acceptable error level grows and the unknown area in the incomplete patterns is not too large, the patterns recalled by the standard associative memory are "more similar" [8] to the original patterns than the patterns recalled by the CASM1 model or the HAM model. Since the standard associative memory is "greatly" overloaded in the conducted experiments, it recalls spurious patterns generated by all stored patterns. The error measured in the recalled spurious patterns can easily fall below the acceptable error level. In the same situation, the HAM model and CASM1 model recall patterns with the higher error. It could be caused by the layered structure of these models. If the first layer of the model recalls "incorrect" ancestor, the pattern recalled by the whole model is probably "incorrect" too. Thus, the error in the recalled pattern can be higher than that of the standard associative memory.

In the next experiments, we focus on the histograms of the error in the incomplete patterns. We measure the average number of incomplete patterns recalled with the corresponding error in every data set. The results are obtained from the CASM1 model with four different pattern rates ($r = 1$, $r \sim 1/2$, $r = 1/3$ and $r \sim 1/5$) and summarized in Figure 52.

The horizontal axis denotes the error (i.e. the number of erroneously recalled elements in one incomplete pattern). The vertical axis shows the relative number of incomplete patterns recalled with the given error (in percents). Incomplete patterns recalled by the CASM1 model with a zero error are not depicted in Figure 52.

The area under every histogram curve corresponds to the total number of incomplete patterns recalled with non-zero error. The maximum values of the histograms in the CASM1 model increase with the decreasing pattern rate r. In addition, with the increasing maximum values the number of incomplete patterns recalled with very small error is growing.

The shape of curves in Figure 52 are similar to the histograms obtained

[8] Hamming distance

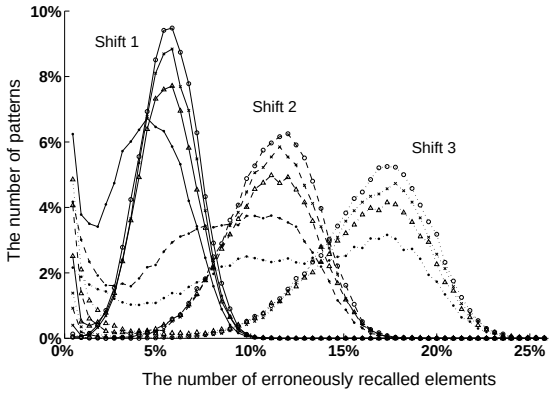


Figure 52: The histogram of the error in the incomplete patterns recalled by the CASM1 model for three types of the diagonal shift: the shift by 1 (the solid lines), the shift by 2 (the dashed lines) and the shift by 3 (the dotted lines). The CASM1 model with four different pattern rates $r = 1$ (the point marker), $r \sim 1/2$ (the triangle marker), $r = 1/3$ (the cross marker) and $r \sim 1/5$ (the circle marker) are considered.

from the HAM models (Figure 43 and Figure 44) and the standard associative memory (Figure 48). The histogram maxima for the CASM1 model are located close to the histogram maxima of the HAM model (6%, 12% and 17% for the diagonal shift by 1, 2 and 3, respectively). The peaks of the histograms of the CASM1 model are higher than that of the HAM models and lower than that of the standard associative memory.

7.5 Complexity of the HAM models

In this chapter, we discuss the memory- and the time- complexity of our HAM models. The HAM models are composed of local associative memories grouped into layers. In the experiments, the HAM models with two layers are considered.

The memory complexity of the HAM models is characterized by the number of local associative memories. The theoretical number of local associative memories in the HAM models is derived. The theoretical results are compared with the experimental results. At the end of the chapter, the time complexity of the models is shortly sketched.

7.5.1 Memory complexity

The memory costs of the network corresponds to the number of elements necessary for storing the network in the computer memory. Since the HAM model is composed of local associative memories grouped into (two) layers, the memory costs of the HAM model is characterized by the total number of local associative

memories (in all layers of the HAM model) and the memory costs of the local associative memory.

The local associative memory is an autoassociative memory. The memory costs $MemCost$ of the local associative memory (LocalAM) with n ($n > 0$) neurons can be expressed as

$$MemCost(LocalAM) = \frac{n(n-1)}{2} \tag{66}$$

where the number of neurons n corresponds to the dimension of the patterns processed by the memory. The memory costs of the local associative memory is given by the number of elements necessary for storing the weight matrix of the memory in the computer memory as a vector of numbers. The weight matrix of the autoassociative memory with n ($n > 0$) neurons is $n \times n$ symmetric matrix with a zero diagonal (Chapter 3.5).

In order to express the memory costs of the HAM model, it is necessary to determine the number of local associative memories in the model. First, we derive the theoretical minimum number of local associative memories in the HAM model with respect to the parameters of the HAM models (the pattern rate r and the capacity coefficient c) and the number of stored patterns m. Then, we compare the theoretical results with our experimental results.

7.5.2 Number of local associative memories

The maximum number of patterns that can be stored in every local associative memory of the HAM model is restricted to

$$max = 0.15 \cdot c \cdot n$$

(Equation 45) where c ($0 < c \leq 1$) is the capacity coefficient. If every local associative memory of the HAM model is maximum loaded, the minimum number of local associative memories needed for storing m patterns with the dimension n is given by the expression

$$\frac{m}{max} = \frac{m}{0.15 \cdot c \cdot n} = \frac{20 \cdot m}{3 \cdot c \cdot n}. \tag{67}$$

We say that the local associative memory is maximum loaded when the number of patterns stored in the local memory is very close to its maximum number max. Hence, m/max is the minimum number of local associative memories that are necessary for storing m patterns.

We analyze the theoretical number of local associative memories that are necessary for storing m patterns in the HAM model (with two layers). In the analysis, the hierarchical structure of the HAM model is considered. The patterns to be stored in the HAM model are divided into two sets: ancestors and descendants. The ancestors are stored in the first layer of the HAM model and the descendants (difference patterns) are stored in the second layer of the HAM model (Section 7.2).

We denote *anc* and *desc* the numbers of ancestors and descendants, respectively. For storing *anc* ancestors in the HAM model, the minimum number of local associative memories in the first layer can be expressed as

$$\#_{anc} = \left\lceil \frac{20 \cdot anc}{3 \cdot c \cdot n} \right\rceil. \tag{68}$$

where $\lceil z \rceil$ denotes the smallest integer number that is not less than z. Equivalently, the minimum number of local associative memories of the second layer necessary for storing *desc* descendants (difference patterns) is

$$\#_{desc} = \left\lceil \frac{20 \cdot desc}{3 \cdot c \cdot n} \right\rceil. \tag{69}$$

The minimum number of local associative memories in the HAM model corresponds to

$$\#_m = \#_{anc} + \#_{desc} = \left\lceil \frac{20 \cdot anc}{3 \cdot c \cdot n} \right\rceil + \left\lceil \frac{20 \cdot desc}{3 \cdot c \cdot n} \right\rceil \tag{70}$$

when m patterns are stored in the HAM model ($m = anc + desc$).

The derived results are summarized in the Table 10 with respect to various values of the pattern rate r and the capacity coefficient c. The parameters r and c are chosen as in Chapter 7.3 [9]. In Table 10, every table element corresponds to the derived number of local associative memories in the HAM model with the given pattern rate r and the capacity coefficient c. The first number in the square bracket shows $\#_{anc}$ (Equation 68) and the second number in the square bracket represents $\#_{desc}$ (Equation 69). The sum of these two numbers is the theoretical minimum number of local associative memories when $m = 100$ patterns ($m = anc + desc$) are stored in the HAM model with the pattern rate r ($r = anc/desc$) and the capacity coefficient c.

The presented results are the lower estimations for the number of local associative memories in the HAM models. In the derived results, the full loading of every local associative memory is considered. In practice, the number of local associative memories is expected to be higher as it depends also on the structure of incoming data. Based on the corresponding layer training algorithm of the HAM model, the local associative memories do not have to be fully loaded.

[9] The symbols $r \sim 1/5$ and $r \sim 1/2$ denote the pattern rates $r = 17/83$ and $r = 33/67$, respectively

The theoretical number of LocalAM				
c	$r \sim 1/5$	$r = 1/3$	$r \sim 1/2$	$r = 1$
1	4	4	3	4
	$[1+3]$	$[1+3]$	$[1+2]$	$[2+2]$
0.9	4	4	5	4
	$[1+3]$	$[1+3]$	$[2+3]$	$[2+2]$
0.8	5	4	5	4
	$[1+4]$	$[1+3]$	$[2+3]$	$[2+2]$
0.7	5	6	5	6
	$[1+4]$	$[2+4]$	$[2+3]$	$[3+3]$
0.6	6	6	6	6
	$[1+5]$	$[2+4]$	$[2+4]$	$[3+3]$
0.5	7	7	6	6
	$[2+5]$	$[2+5]$	$[2+4]$	$[3+3]$
0.4	9	8	8	8
	$[2+7]$	$[2+6]$	$[3+5]$	$[4+4]$
0.3	11	11	11	10
	$[2+9]$	$[3+8]$	$[4+7]$	$[5+5]$
0.2	16	16	15	16
	$[3+13]$	$[4+12]$	$[5+10]$	$[8+8]$

Table 10: The theoretical number of local associative memories (LocalAM) in the HAM model with respect to various values of the pattern rate r and the capacity coefficient c. m ($m = 100$) patterns to be stored in the HAM model are divided into anc ancestors and $desc$ descendants ($m = anc + desc$). The pattern rate is defined by $r = anc/desc$.

7.5.3 Comparison with the experimental results

The previous section is focused on the theoretical minimum numbers of local associative memories that are needed for storing patterns in the HAM model. Now, we describe our experimental results with respect to the number of local associative memories in the HAM models.

In the experiments, m ($m = 100$) patterns are stored in the HAM1 model and the HAM2 model with the two-layer hierarchy. According to the pattern rate r, the patterns are divided into the ancestors and the descendants; each of them is stored in the corresponding layer of the HAM model. Every experiment is repeated for every data set (Chapter 7.1). The experimental results are averaged for all data set. The obtained results are summarized in Table 11.

In Table 11, the first column shows the value of the capacity coefficient c. The next four columns correspond to the HAM1 model results (with respect to various pattern rates r). The last four columns show the HAM2 model results. The first number in the square bracket denotes the average number of local associative memories needed for storing *anc* ancestors in the first layer of the HAM1 (or the HAM2) model. The second number in the square bracket represents the average number of local associative memories that are necessary for storing *desc* descendants (difference patterns) in the second layer of the HAM1 (or the HAM2) model. The number of the greater font size describes the total number of local associative memories when m patterns ($m = anc + desc$) are stored in the HAM1 (or the HAM2) model with the given pattern rate r and the capacity coefficient c.

The results obtained from the HAM1 model (Table 11) are slightly higher than the theoretical minima (Table 10). It shows that the local associative memories in the HAM1 model are almost fully loaded. The number of local associative memories in the HAM2 models is higher than that of the HAM1 model (and the theoretical minima as well). It is due to the tree structure of the HAM2 model. The tree structure of the HAM2 model needs more local associative memories in every layer. The improvement of the storage ability of the HAM2 model leads to the higher number of local associative memories in the model.

When the capacity coefficient c ($0 < c \leq 1$) decreases, the number of local associative memories in the HAM model is higher as the maximum number of patterns that to be stored in every local associative memory is more restricted. For some values of r, the number of local associative memories can be notably lower than the number of memories in the HAM model with the same value of c but the higher or lower values of r (e.g. the HAM model with the parameters $c = 0.5$ and $r = 1/3$).

In these cases, the first layer of the HAM model is able to store more patterns than it is stored in it (the local associative memories are not fully loaded). When the pattern rate r is adequately increased (i.e. the number of ancestors is higher), the local associative memories in the first layer are used more efficiently while the number of memories in the first layer remains the same. At the same time, the number of patterns to be stored in the second layer of the HAM model is lower (than that of the HAM model with the lower pattern rate r). It can lead to decrease in the number of local associative memories in the HAM model.

The number of LocalAM								
	The HAM1 model				The HAM2 model			
c	$r \sim 1/5$	$r = 1/3$	$r \sim 1/2$	$r = 1$	$r \sim 1/5$	$r = 1/3$	$r \sim 1/2$	$r = 1$
1	4.68 [1 + + 3.68]	4.83 [1.15 + + 3.68]	5.67 [1.84 + + 3.83]	4.86 [2.02 + + 2.84]	4.68 [1 + + 3.68]	5.04 [1.15 + + 3.89]	6.83 [1.84 + + 4.99]	5.69 [2.02 + + 3.67]
0.9	4.81 [1 + + 3.81]	4.86 [1.15 + + 3.71]	5.13 [2 + + 3.13]	4.95 [2.09 + + 2.86]	4.81 [1 + + 3.81]	5.1 [1.15 + + 3.95]	6.17 [2 + + 4.17]	5.88 [2.09 + + 3.79]
0.8	5 [1 + + 4]	5.04 [1.15 + + 3.89]	5.22 [2 + + 3.22]	5.49 [2.32 + + 3.17]	5 [1 + + 4]	5.24 [1.15 + + 4.09]	6.13 [2 + + 4.13]	6.78 [2.32 + + 4.46]
0.7	5.03 [1 + + 4.03]	6.39 [2 + + 4.39]	5.48 [2 + + 3.48]	6 [3 + + 3]	5.03 [1 + + 4.03]	7.1 [2 + + 5.1]	6.19 [2 + + 4.19]	7.25 [3 + + 4.25]
0.6	6 [1 + + 5]	6.32 [2 + + 6.32]	6 [2 + + 4]	6.01 [3 + + 3.01]	6 [1 + + 5]	7.03 [2 + + 5.03]	6.54 [2 + + 4.54]	7.27 [3 + + 4.27]
0.5	9.5 [2 + + 7.5]	7.01 [2 + + 5.01]	9 [3 + + 6]	8 [4 + + 4]	12.5 [2 + + 10.5]	7.66 [2 + + 5.66]	12.5 [3 + + 9.5]	10 [4 + + 6]
0.4	9 [2 + + 7]	8 [2 + + 6]	9 [3 + + 6]	8.02 [4 + + 4.02]	9.36 [2 + + 7.36]	8.73 [2 + + 6.73]	9.51 [3 + + 6.51]	9.52 [4 + + 5.52]
0.3	12 [2 + + 10]	12 [3 + + 9]	12 [4 + + 8]	12 [6 + + 6]	12.13 [2 + + 10.13]	12.68 [3 + + 9.68]	13.27 [4 + + 9.27]	14 [6 + + 8]
0.2	17 [3 + + 14]	18.2 [5 + + 13.2]	18 [6 + + 12]	18.01 [9 + + 9.01]	18.05 [3 + + 15.05]	22.18 [5 + + 17.18]	19.56 [6 + + 13.56]	21.27 [9 + + 12.27]

Table 11: The obtained average number of local associative memories (LocalAM) in the HAM1 and HAM2 model with respect to various values of the pattern rate r and the capacity coefficient c. m ($m = 100$) patterns are divided into anc ancestors and $desc$ descendants ($m = anc + desc$) and stored in the HAM1 and the HAM2 model. The pattern rate r is defined as the ratio between the number of ancestors and descendants; i.e. $r = anc/desc$.

With the increasing pattern rate r, the storage ability of the HAM model is improved. In addition, the storage ability is improved for the decreasing capacity coefficient c. At the same time, the number of local associative memories can be increased as well. The experiments show that the better results are achieved for higher values of the pattern rate r.

For $c = 0.6$, the storage ability of the HAM models is greater than 94%. It seems to be acceptable level of the storage ability in practice. At the same time, the number of local associative memories is reasonable high. The storage ability of the HAM models with $c < 0.6$ is further improved but the gains are marginal. Moreover, the number of local associative memories grows rapidly if $c < 0.6$.

Another approach to processing larger amounts of (correlated) patterns could be based on a sequence of (auto)associative memories. The number of (auto)associative memories in the sequence is expected to be lower than the total number of local associative memories in the HAM model (especially in the HAM2 model). With the increasing correlation between patterns, the storage capacity of the (auto)associative memory decreases. Thus, the number of (auto)associative memories in the sequence is expected to grow rapidly.

When the patterns are stored in the sequence of (auto)associative memories, the processing of incomplete patterns can lead to problems, especially when the unknown area is large. In these cases, the incomplete patterns can easily fall into the basin of attractions of patterns different from the pattern originally stored in the model. These patterns can be "more similar" to the incomplete patterns that the original patterns.

The HAM models are expected to process the (correlated) patterns more efficiently than the sequence of (auto)associative memories. The hierarchical structure of the HAM models is suitable for dealing with the correlated data. In the HAM models, all layers process the different information. The first layer processes the "basic" information encoded in the patterns (e.g. centers of the clusters or representative patterns). Other layers process the more detailed information (e.g. intrinsic patterns of clusters). The information recalled in the layer is used to help to recall the pattern in the "next" layer.

The information stored in the first layer is expected to be as orthogonal as possible. Even for the incomplete patterns with the larger unknown area, the first layer patterns (i.e. the "representative pattern") can be recalled correctly. In the "next" layer, the processed information is based on the presented incomplete pattern as well as on the information recalled by the first layer. Thus, the recall process can deal more efficiently.

7.5.4 Time complexity

Our HAM models are composed of local associative memories grouped into layers. Every local associative memory is a model of an (auto)associative memory [30]. Associative memories operate in two dynamics: the training (storage) process and the recall process. In this section, we review the time complexity of autoassociative memories. Based on the autoassociative memory results, we derive the time complexity of the HAM models.

The training process

The training process of an autoassociative memory with n ($n > 0$) neurons consists in storing the training patterns in $n \times n$ weight matrix of the memory. It corresponds to an update of every matrix element in the weight matrix according to the presented training pattern and the concrete training rule. As the weight matrix of the autoassociative memory is symmetric with zero diagonal, it is necessary to update only $n \cdot (n-1)/2$ matrix elements. Let T_{store} denote the amount of time that storing the pattern in the autoassociative memory takes. It holds

$$T_{store} = \frac{n \cdot (n-1)}{2}. \tag{71}$$

In the HAM models, all layers (composed of local associative memories) are trained separately. The layers are trained in a sequential order as they use outputs recalled by the previous already trained layers (e.g. the tree structure in the HAM2 model or creating difference patterns). The number of operations of the training process in the HAM model is given by the number of operations of training process in all layers [10].

Now, we derive the number of operations that the layer training process in the HAM model takes. The process of storing the pattern $\vec{x}$ in the layer of the HAM model comprises in general

1. storing the pattern $\vec{x}$ in the corresponding local associative memories of the layer,
2. recalling the pattern $\vec{x}$ in the corresponding local associative memories of the layer,
3. finding the "suitable" local associative memory where the pattern $\vec{x}$ remains stored,
4. unlearning the pattern $\vec{x}$ from all "unsuitable" local associative memories of the layer,
5. storing the pattern $\vec{x}$ in the newly created local associative memory of the layer when no "suitable" local associative memory is found.

[10] The process of gathering the information necessary for storing a pattern in the layer is not considered in the assessment of the time complexity (e.g. detection of the appropriate subset of local associative memories in the HAM2 model)

The complete layer training algorithms for the HAM1 and HAM2 models are described in Chapter 6. Now, we discuss the above-mentioned steps in more detail. We assume that all patterns processed by the model are $n-$dimensional ($n > 0$).

1. Storing the pattern in the local associative memory of the HAM model corresponds to the process of storing the pattern in the autoassociative memory. The number of operations is

$$T_{store} = \frac{n \cdot (n-1)}{2}$$

   (Formula 71), where n is the number of neurons in the memory.

2. Let T_{recall} denote the amount of time that the recall of the pattern in the local associative memory takes. It corresponds to the amount of time that the pattern recall in the autoassociative memory takes. The recall process of the autoassociative memory is discussed in more detail in the next subsection. Now, we show the obtained results - Formula 81 and Formula 84. The convergence time T_{recall} is

$$T_{recall} \leq 2 \cdot n^2 \cdot m_{patt}$$

   for the local associative memory with the Hebbian training rule (Formula 10). For the local associative memory with the modified training rule (Formula 43), the convergence time is

$$T_{recall} \leq \frac{2}{\varepsilon} \cdot n^2 \cdot m_{patt}$$

   where ε is the least possible amount of energy decrease achievable in one time step of the recall process in the local associative memory. The symbol m_{patt} denotes the number of patterns stored in the memory.

3. This step consists in finding the "suitable" local associative memory in the layer. The "suitable" local associative memory is found according to the Hamming distances between the presented pattern and the patterns recalled by the corresponding local associative memories of the layer. Computing the Hamming distance between two $n-$dimensional patterns requires

$$T_{hamm} = n$$

   operations. The "suitable" local associative memory is selected according to the computed Hamming distances. Let T_{find} denote the number of operations that finding the "suitable" local associative memory takes. The upper bound of the number of operations T_{find} is

$$T_{find} \leq s$$

   where s denotes the number of local associative memories, in which the pattern is processed. In the HAM1 model, the pattern is processed by all

local associative memories of the layer. In the HAM2 model, the patten is processed by the local associative memories of the corresponding subset (Chapter 6).

4. Unlearning the pattern is an operation inversed to the pattern training process (e.g. Formula 40). It performs the same number of operations as the pattern training does, i.e.

$$T_{unlearn} = \frac{n \cdot (n-1)}{2}.$$

5. If the pattern is unlearnt from all corresponding local associative memories of the layer, the pattern is stored in a newly created local associative memory with n neurons. It requires

$$T_{storenew} = \frac{n \cdot (n-1)}{2}$$

operations.

The total number of operations T_{DLT} of the layer training process (after presenting the pattern) can be written as

$$T_{DLT} = s \cdot T_{store} + s \cdot T_{recall} + s \cdot T_{hamm} + T_{find} + (s-1) \cdot T_{unlearn}, \quad (72)$$

if the pattern remains stored in one "suitable" local associative memory. The symbol s denotes the number of local associative memories, in which the pattern is processed. In the HAM model, the number of patterns m_{patt} stored in every local associative memory is limited to $m_{patt} \leq 0.15 \cdot c \cdot n$ (Formula 45), where c $(0 < c \leq 1)$ is the capacity coefficient. Therefore, we get

$$\begin{aligned} T_{DLT} &\leq s \cdot \frac{n \cdot (n-1)}{2} + s \cdot 2 \cdot n^2 \cdot m_{patt} + s \cdot n + s + (s-1) \cdot \frac{n \cdot (n-1)}{2} \\ &\leq s \cdot (2 \cdot n^2 \cdot m_{patt} + n^2 + 1) - \frac{n \cdot (n-1)}{2} \\ &\leq s \cdot (0.3 \cdot n^3 + n^2 + 1) - \frac{n \cdot (n-1)}{2} \\ &= O(s \cdot n^3), \end{aligned} \quad (73)$$

when the Hebbian training rule (Formula 10) is used to store patterns in the local associative memory. If the modified training rule (Formula 43) is used to store patterns in the local associative memory, it holds

$$\begin{aligned} T_{DLT} &\leq s \cdot \frac{n \cdot (n-1)}{2} + s \cdot \frac{2}{\varepsilon} \cdot n^2 \cdot m_{patt} + s \cdot n + s + (s-1) \cdot \frac{n \cdot (n-1)}{2} \\ &\leq s \cdot \left(\frac{2}{\varepsilon} \cdot n^2 \cdot m_{patt} + n^2 + 1\right) - \frac{n \cdot (n-1)}{2} \\ &\leq s \cdot \left(0.3 \cdot \frac{1}{\varepsilon} \cdot n^3 + n^2 + 1\right) - \frac{n \cdot (n-1)}{2} \\ &= O(s \cdot \frac{1}{\varepsilon} \cdot n^3), \end{aligned} \quad (74)$$

where ε is the least possible amount of energy decrease achievable in one time step of the recall process in the local associative memory.

In a case, the pattern is unlearnt from all corresponding local associative memories (and it is stored in the newly created memory), the total number of operations T_{DLT} of the layer training process (after presenting the pattern) is

$$T_{DLT} = s \cdot T_{store} + s \cdot T_{recall} + s \cdot T_{hamm} + T_{find} + s \cdot T_{unlearn} + T_{storenew}. \quad (75)$$

The symbol s denotes the number of local associative memories, in which the pattern is processed. The number of patterns m_{patt} stored in every local associative memory is limited to $0.15 \cdot c \cdot n$ $(0 < c \leq 1)$. If patterns are stored in the local associative memory according to the Hebbian training rule, the total number of operations T_{DLT} is

$$\begin{aligned} T_{DLT} &\leq s \cdot \frac{n \cdot (n-1)}{2} + s \cdot 2 \cdot n^2 \cdot m_{patt} + s \cdot n + s + s \cdot \frac{n \cdot (n-1)}{2} \\ &\quad + \frac{n \cdot (n-1)}{2} \\ &\leq s \cdot (2 \cdot n^2 \cdot m_{patt} + n^2 + 1) + \frac{n \cdot (n-1)}{2} \\ &\leq s \cdot (0.3 \cdot n^3 + n^2 + 1) + \frac{n \cdot (n-1)}{2} \\ &= O(s \cdot n^3). \end{aligned} \quad (76)$$

If the modified training rule is used to store patterns in the local associative memories of the HAM models, we get

$$\begin{aligned} T_{DLT} &\leq s \cdot \frac{n \cdot (n-1)}{2} + s \cdot \frac{2}{\varepsilon} \cdot n^2 \cdot m_{patt} + s \cdot n + s + s \cdot \frac{n \cdot (n-1)}{2} \\ &\quad + \frac{n \cdot (n-1)}{2} \\ &\leq s \cdot \left(\frac{2}{\varepsilon} \cdot n^2 \cdot m_{patt} + n^2 + 1\right) + \frac{n \cdot (n-1)}{2} \\ &\leq s \cdot \left(0.3 \cdot \frac{1}{\varepsilon} \cdot n^3 + n^2 + 1\right) + \frac{n \cdot (n-1)}{2} \\ &= O(s \cdot \frac{1}{\varepsilon} \cdot n^3) \end{aligned} \quad (77)$$

where ε is the least possible amount of energy decrease achievable in one time step of the recall process in the local associative memory.

The layer training process depends mainly on the pattern dimensionality. The number of local associative memories of the layer is not so crucial as most of the described processes can be run in parallel in all corresponding local associative memories of the layer (except the process of finding the "suitable" local associative memory - Step 3). In practice, the pattern dimension n is expected to be greater than the number of local associative memories in the layer.

The recall process

Here, we focus on the recall process. First, we review the convergence time of the autoassociative memory. Afterwards, we derive the complexity of the recall process in the HAM model.

The recall process of the autoassociative memory is an iterative process. Hopfield [30] has shown that the autoassociative memory (with symmetric weights and a zero diagonal) using an asynchronous dynamics eventually converges to the stable state of the energy function. Parberry [61] has shown that any productive sequential recall process of the autoassociative memory with the integer weights ($w_{ij} \in \mathbb{Z}$ for $1 \leq i, j \leq n$) converges in time

$$T_{recall} \leq 2 \cdot \sum_{i,j} |w_{ij}|. \tag{78}$$

The symbol $\mathbb{Z}$ denotes the set of all integer numbers, w_{ij} is the weight between the neuron i and the neuron j and n ($n > 0$) is the number of neurons. The recall process is sequential if a single neuron updates its state within each time step. The recall process is productive if at least one unstable neuron (i.e. the neuron that changes its state) is updated in each time step. Since the autoassociative memory has n neurons, Formula 78 can be rewritten as

$$T_{recall} \leq 2 \cdot \sum_{i,j} |w_{ij}| \leq 2 \cdot n^2 \cdot \max_{i,j} |w_{ij}|. \tag{79}$$

During the training process, m_{patt} ($m_{patt} > 0$) bipolar patterns are stored in the autoassociative memory. Therefore,

$$\max_{i,j} |w_{ij}| \leq m_{patt}. \tag{80}$$

Thus, the upper bound of the convergence time is

$$T_{recall} \leq 2 \cdot \sum_{i,j} |w_{ij}| \leq 2 \cdot n^2 \cdot \max_{i,j} |w_{ij}| \leq 2 \cdot n^2 \cdot m_{patt}. \tag{81}$$

It has been shown that the autoassociative memory with large weights may indeed require an exponential time to converge (e.g. [16]). An autoassociative memory with asynchronous dynamics requiring exponential time to converge has been demonstrated e.g. in [23]. Nevertheless, in practice the number of time steps required for the convergence is typically at most slightly greater than the Hamming distance between the presented input and output patterns [16]. This phenomenon has been also explored analytically [41].

In the implemented HAM model, patterns are stored in the local associative memories according to the modified training rule (Formula 43). In this case, the weights are in general real numbers ($w_{ij} \in \mathbb{R}$ for $1 \leq i, j \leq n$). Fogelman et al. [17] have shown that the autoassociative memory with the real weights ($w_{ij} \in \mathbb{R}$) converges in time

$$T_{recall} \leq \frac{1}{2 \cdot \varepsilon} \cdot \sum_{i,j} |w_{ij}| \leq \frac{1}{2 \cdot \varepsilon} \cdot n^2 \cdot \max_{i,j} |w_{ij}|, \tag{82}$$

where ε is the least possible amount of energy decrease achievable in one time step and n denotes the number of neurons in the memory. When m_{patt} ($m_{patt} > 0$) bipolar patterns are stored in the local associative memory according to the modified training rule (Formula 43), it holds

$$\max_{i,j} |w_{ij}| \leq 4 \cdot m_{patt}. \tag{83}$$

Therefore,

$$T_{recall} \leq \frac{1}{2 \cdot \varepsilon} \cdot \sum_{i,j} |w_{ij}| \leq \frac{1}{2 \cdot \varepsilon} \cdot n^2 \cdot \max_{i,j} |w_{ij}| \leq \frac{2}{\varepsilon} \cdot n^2 \cdot m_{patt}. \tag{84}$$

In the HAM models, the number of patterns that can be stored in every local associative memory is limited to $0.15 \cdot c \cdot n$ (Formula 45), where c ($0 < c \leq 1$) is the capacity coefficient and n ($n > 0$) is the pattern dimension. Thus, Formula 81 can be rewritten as

$$T_{recall} \leq 2 \cdot n^2 \cdot m_{patt} \leq 0.3 \cdot c \cdot n^3 \leq 0.3 \cdot n^3. \tag{85}$$

The dependance of the upper bound of the convergence time upon the pattern dimension n is shown in Figure 53.

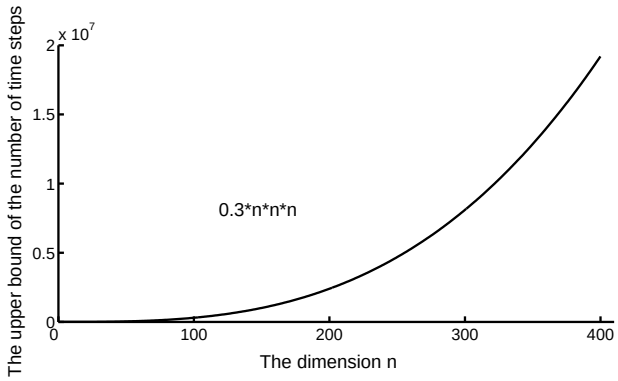


Figure 53: The theoretical upper bound of the convergence time in the autoassociative memory with the Hebbian training rule with respect to the pattern dimension n

In a similar way, Formula 84 can be expressed as

$$T_{recall} \leq \frac{2}{\varepsilon} \cdot n^2 \cdot m_{patt} \leq 0.3 \cdot \frac{1}{\varepsilon} \cdot n^3, \tag{86}$$

where ε is the least possible amount of energy decrease achievable in one time step.

We conduct the experimental simulation in order to determine the number of changing update steps of the autoassociative memory in practice (to make comparison with the derived upper bound). We measure the average number of changing update steps during the pattern recall for various pattern dimensions n: $n = 6 \cdot 6, 7 \cdot 7, \ldots, 20 \cdot 20$. The experimental results are depicted in Figure 54. The solid line with filled circle markers corresponds to the results of the autoassociative memory (AM) with the Hebbian training rule (Formula 10). The dashed line with empty circle markers corresponds to the results of the autoassociative memory (AM) with the modified training rule (Formula 43).

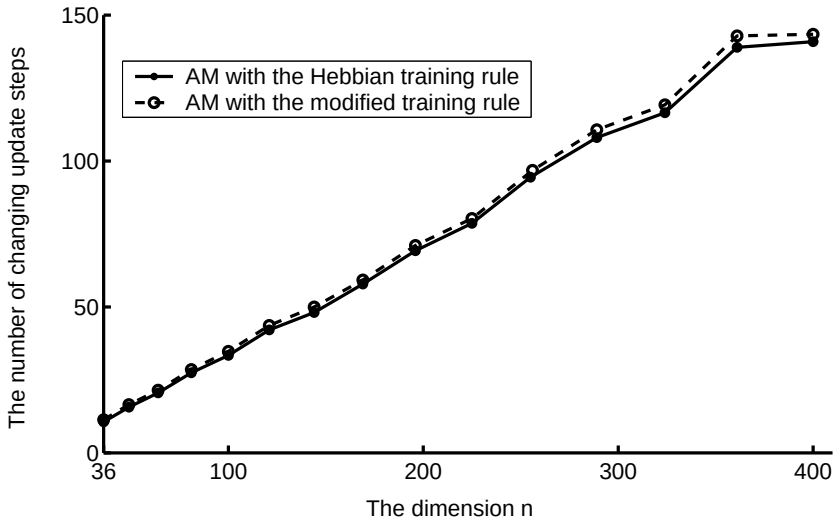


Figure 54: The average number of changing update steps of the recall process in the autoassociative memory (AM) with respect to the pattern dimension n: the autoassociative memory with the Hebbian training rule (the solid line) and the autoassociative memory with the modified training rule (the dashed line). In the experimental simulation, the training patterns with 30% different pattern elements are recalled.

For a given pattern dimension n, every simulation is repeat for 100 different data sets. The results for the corresponding pattern dimension are averaged. In the experiment, $0.15 \cdot n$ training patterns are stored in the autoassociative memory. The symbol n denotes the pattern dimension. During the recall process, the autoassociative memory recalls the training patterns, in which 30% elements have changed their value. The Hamming distance between the input and output patterns corresponds to $0.3 \cdot n$, if the pattern is recalled correctly.

The experimental results (Figure 53) do not exceed the theoretical upper bound (Figure 54). The theoretical bound is the worst-case upper bound and the simulation results show only slightly higher number of changing update steps than the Hamming distance between the input and output pattern. The

theoretical bound is significantly higher than the simulation results.

The obtained experimental results (Figure 54) are in accordance with [16]: the number of time steps required for the convergence is in practice typically at most slightly greater than the Hamming distance between the input and output patterns. Theexperimental results of the autoassociative memory with the Hebbian training rule and the autoassociative memory with the modified rule are very similar. The differences are diminutive.

Now, we focus on the recall process in the HAM models. In our HAM models, the recall process is done sequentially in the layers as the output pattern of the layer forms the input pattern to the "next" layer [11]. The number of operations of the recall process in the HAM model is given by the number of operations of the recall processes in all layers and gathering the information necessary for recalling the pattern in the layer.

During the layer recall process, recalling the pattern in the layer comprises in generally

1. recalling the pattern in the corresponding local associative memories of the layer,
2. finding the recalled pattern, which is "the most similar" to the presented pattern.

The complete layer recall algorithms for the HAM1 and HAM2 models are described in Chapter 6. Now, we discuss the steps in more detail.

1. Every local associative memory in the layer is the autoassociative memory with n neurons. Recalling the pattern in the local associative memory requires

   $$T_{recall} \leq 2 \cdot n^2 \cdot m_{patt} \leq 0.3 \cdot n^3$$

   operations (Formula 85), if the Hebbian training rule is used. In a case, patterns are stored in the local associative memory according to the modified training rule, we get

   $$T_{recall} \leq \frac{2}{\varepsilon} \cdot n^2 \cdot m_{patt} \leq 0.3 \cdot \frac{1}{\varepsilon} \cdot n^3$$

   (Formula 86). The symbol m_{patt} denotes the number of patterns stored in the local associative memory. In the HAM model, the number of patterns stored in the local associative memory is limited to $m_{patt} \leq 0.15 \cdot c \cdot n$ where c $(0 < c \leq 1)$ is the capacity coefficient.

2. In order to find the pattern recalled by the layer, the Hamming distance is computed between the patterns recalled by all corresponding local associative memories and the presented pattern. It requires

   $$T_{hamm} = n$$

[11] e.g. to determine the corresponding subset of local associative memories in the HAM2 model

operations, where n denotes the pattern dimension. The pattern recalled by the layer corresponds to the recalled pattern that is the "most similar" to the presented pattern (measured in the Hamming distance). The number of operations depends on the number of local associative memories (denoted as s), in which the pattern is processed. Therefore,

$$T_{find} \leq s.$$

In the HAM1 model, the pattern is recalled in all local associative memories of the layer. In the HAM2 model, the patten is recalled in the local associative memories of the corresponding subset.

The total amount of time T_{LR} of the layer recall process (after presenting the pattern) can be written as

$$T_{LR} = s \cdot T_{recall} + s \cdot T_{hamm} + T_{find} \tag{87}$$

where s denotes the number of local associative memories processing the pattern. If patterns are stored in the local associative memory according to the Hebbian training rule (Formula 10), we get

$$\begin{aligned} T_{LR} &\leq s \cdot 0.3 \cdot n^3 + s \cdot n + s \\ &= O(s \cdot n^3). \end{aligned} \tag{88}$$

If the modified training rule (Formula 43) is used to store patterns in the local associative memory, the number of time steps is given by

$$\begin{aligned} T_{LR} &\leq s \cdot 0.3 \cdot \frac{1}{\varepsilon} \cdot n^3 + s \cdot n + s \\ &= O(s \cdot \frac{1}{\varepsilon} \cdot n^3) \end{aligned} \tag{89}$$

where ε is the least possible amount of energy decrease achievable in one time step and n denotes the pattern dimension.

The layer recall process of the pattern depends mainly on the pattern dimensionality. The number of local associative memories of the layer is not so crucial as most of the time-consuming operations (e.g. recalling the pattern in the local associative memory) can be run in parallel for all corresponding local associative memories in the layer.

8 Evaluation and related work

Techniques for processing the spatial patterns can be based on associative memory models. However, traditional associative memories are not suitable to situations where many patterns are similar to one another. Their storage capacity is usually very limited and they are not effective or even capable of storing/recalling correlated patterns. To avoid these limitations, this thesis presents a new model of associative memory called Hierarchical associative memory (HAM). The HAM model consists of local associative memories grouped into several layers. The developed training and recall algorithms of the HAM model allow processing of huge amounts of spatial data often containing mutually correlated patterns.

8.1 Evaluation of the HAM model

In this section, we evaluate all requirements, as imposed on the developed hierarchical associative memory model in Chapter 5.

8.1.1 Storage of a huge amount of patterns

The first requirement is focused on the number of patterns that can be stored in the model. The requirement is fulfilled if the model is able to store a huge number of patterns.

For the storage process of the HAM model, the dynamic layer training (DLT) algorithm is developed. The algorithm is based on a simple idea. A pattern is stored in one of the existing local associative memories of the layer. When it is not possible to store the pattern in any corresponding local associative memory of the layer (e.g. the local associative memories are overloaded), a new local associative memory is automatically created and added to the layer. Applying the DLT algorithm, the HAM model is able to store an arbitrary number of (correlated) patterns. Thus, the HAM model can avoid any capacity limitation of traditional associative memories.

The advantage of the developed DLT algorithm is no need to state the initial number of local associative memories in every layer before the storage process starts. The initial structure of the HAM model can be very simple - one local associative memory in every layer. Other local associative memories are added automatically to corresponding layers during the training process according to the incoming patterns. Thus, the structure of the HAM models is adapted to the incoming patterns and their mutual correlation.

8.1.2 Storage of correlated patterns

Spatial pattern processing is needed in applications like navigation, cartography etc. Spatial patterns can correspond to the fragmentary maps of the scenery. Since they can be very similar to each other, spatial patterns cannot be assumed mutually orthogonal (or even almost orthogonal). This requirement is adopted to the ability of the model to store any kind of patterns (orthogonal/correlated).

The proposed DLT algorithm for the HAM model uses simple heuristics to select the local associative memory, in which the pattern is stored. The pattern remains stored in the local associative memory where the pattern or its "noisy version" are recalled correctly (i.e. without any error). The DLT algorithm is able to process any kind of patterns (correlated/uncorrelated). When the pattern is correlated with the patterns previously stored in the local associative memory, we expect that the memory would not be able to store and recall the pattern correctly. Thus, the pattern remains stored in another local associative memory, in which the previously stored patterns are more mutually orthogonal. In cases that no "suitable" local associative memory is found in the corresponding layer, the pattern is stored in a newly created local associative memory of the layer.

Anyway, in the DLT algorithm we cannot predict anything about recalling the previously stored patterns. Previously stored patterns can be recalled incorrectly after storing some other patterns. One solution to prevent "forgetting" the previously stored patterns could be based on gathering the redundant information (e.g. a list of all patterns stored in every local associative memory). We improve the performance of the HAM model with respect to storing correlated patterns in a different way. We artificially restrict the maximum number of patterns to be stored in every local associative memory (using the capacity coefficient c). When the maximum number of patterns in every local associative memory is decreased, the probability of "forgetting" the previously stored patterns is reduced as well. The experimental results carried out confirm the legitimacy of this restriction (for $c < 0.6$ more than 97% stored patterns are recalled correctly).

8.1.3 Reliability of the storage process

This requirement is adopted to the ability of the model to recall patterns stored in it (i.e. the storage ability). The experiments show the robust storage ability of our HAM1 and HAM2 models with two-layer hierarchy. The performance of the models is influenced by the pattern rate r that determines the ratio between the number of patterns stored in the first- and second- layer of the HAM model, respectively. The storage ability of the HAM model can be further improved by restricting the storage capacity of every local associative memory (using the capacity coefficient c).

The right choice of a local associative memory for storing the pattern represents a crucial point of the successful overall performance of the HAM models. The HAM1 model stores the pattern in any local associative memory of the corresponding layer. In the HAM2 model, the patterns are grouped hierarchically according to the information recalled by the "previous" layer. The "previous-layer" information determines the corresponding subset of local associative memories, in which the pattern can be stored.

The experimental results show the significant differences between the HAM1 and HAM2 models for the larger values of the pattern rate r. In this case, the storage ability is enhanced in the HAM2 model (in contrast to the HAM1 model) due to the influence of the tree structure. The experiments also show that the

most processed patterns can be stored and recalled correctly (for $c < 0.6$ more than 97% patterns). Several patterns are recalled with any error, but the error in the patterns usually does not exceed 2%.

8.1.4 Reliability of the recall process

This requirement refers to the ability of models to recall incomplete patterns (i.e. the recall ability). Our experimental results show that the recall ability of the HAM2 model is higher than that of the HAM1 model but the differences are just minor. We also define the error acceptance like the maximum number of elements in a pattern that can be recalled erroneously. In such cases, a pattern is considered to be recalled correctly if the number of erroneously recalled elements does not exceed the error acceptance level. In accordance to our opinion, the number of incomplete patterns recalled correctly is increasing with the increasing error acceptance.

The HAM models have relative robust recall abilities with respect to processing the incomplete patterns, whenever the unknown area is sufficiently small and the acceptable error is about 5%. Recalling the incomplete patterns with the enlarging unknown area leads to the increasing error in the recalled patterns, as the incomplete patterns can easily fall into the basins of attraction of other patterns.

The experiments carried out also confirm that the appropriately recalled "previous-layer" pattern represents the crucial point in the HAM model performance. In the case of two layer hierarchy, the appropriately recalled "first-layer" pattern should represent "well enough" the corresponding group of the "second-layer" patterns. As the pattern rate r rises, recalling the "first-layer" patterns is improved. It increases the overall performance of the HAM model.

8.2 Related work

In the following paragraphs, we discuss two related models of associative memories: the standard associative memory and the cascade associative memory. Our discussions of related works are done in consideration of processing larger amounts of (correlated) patterns.

8.2.1 Standard associative memory

A standard associative memory is not able to store and recall correctly a large number of patterns. If the number of stored patterns exceeds the theoretical limit [12] or the patterns are not orthogonal, the stored patterns can be destroyed and become lost [30], [4]. Our experiments carried out confirm this theoretical knowledge.

The experiments are focused on processing larger amounts of (correlated) patterns. Since the number of processed patterns ($m = 100$) exceeds tripply the theoretical limit of the standard associative memory, no pattern is recalled correctly (i.e. without any error). The obtained results also show the relatively

[12] $0.15 \cdot n$ where n ($n > 0$) is the dimension of patterns

large number of erroneously recalled elements in every pattern (10% – 50% erroneously recalled elements in every pattern).

Since the standard associative memory is not able to recall any of the stored patterns without any error, it is not able to process the incomplete patterns without any error either. We also focus on the situations, in which the incomplete patterns do not have to be recalled without any error and a small error is acceptable (the acceptable error). Even for the small values of the error acceptance level ($< 2\%$), the standard associative memory does not recall any incomplete patterns correctly. It is caused by overloading of the standard associative memory and/or correlations between the patterns. The standard associative memory recalls spurious patterns instead of the patterns stored originally in the memory.

The experimental results are in accordance with our opinion that the standard associative memory is not suitable for applications where processing of a large number of (correlated) patterns is needed. The HAM models are developed with an emphasis on the necessity to process large amounts of patterns. Thus, the storage and recall abilities of the proposed HAM models are superior to the standard associative memory with respect to processing larger amounts of data.

8.2.2 Cascade associative memory

The cascade associative memories have been proposed by Hirahara et al. to handle a special kind of correlated patterns. They are composed of two associative memories. To make the comparison with our HAM model, we focus on the CASM1 model [26] as its structure is the most similar to our HAM models.

The CASM1 model consists of two autoassociative memories AM1 and AM2. Since AM1 and AM2 are models of the standard (auto)associative memory, the theoretical maximum number max of $n-$dimensional patterns to be stored in each of them is limited. In our experiments, we set $n = 225$ and thus $max \doteq 34$. The performance of the CASM1 model is determined with respect to processing larger amounts of patterns. We focus on the performance of the CASM1 model with respect to four different pattern rates $r = 50/50$ ($r = 1$), $r = 33/67$ ($r \sim 1/2$), $r = 25/75$ ($r = 1/3$) and $r = 17/83$ ($r \sim 1/5$). The pattern rate r denotes the ratio between the number of patterns stored in AM1 and AM2, respectively.

If the number of patterns stored in AM1 does not exceed the theoretical limit (i.e. $r < 1$), the CASM1 model is able to recall without any error most patterns stored in AM1. The patterns stored in AM2 are usually recalled with a non-zero error, as their number exceeds the theoretical limit of AM2. If the number of patterns stored in AM1 is higher than the theoretical limit (i.e. $r = 1$), the storage ability of the CASM1 model is relatively poor. AM1 is overloaded and it recalls the spurious patterns instead of the original patterns. If AM1 recalls the spurious patterns, the patterns recalled by the CASM1 model do not have to correspond to the original patterns.

In comparison to our HAM models, the storage ability of the CASM1 model is not very robust. The conducted experiments show that the upper limit of the

CASM1 storage ability is bounded by that of AM1. The obtained results are in accordance with the results presented by the authors of the CASM1 model (Hirahara et al. [26]).

The recall ability of the CASM1 model is affected by the storage ability. For a zero error acceptance, the recall ability of the CASM1 model is similar to the storage ability. In this case, the incomplete patterns recalled correctly by the CASM1 model corresponds mainly to the patterns stored originally in AM1. With the increasing error acceptance level, the recall ability of the CASM1 model is growing. The experimental results show that the recall ability of the CASM1 model are lower than that of the HAM models.

The CASM1 model is not very applicable to problems where a huge number of patterns needs to be processed. Our HAM model represents a generalization of the CASM1 model developed with a stressed necessity to handle larger amounts of data. The experiments show that the storage and recall abilities of the HAM model are superior to the CASM1 model.

8.3 Summary

Our research in the area of associative memories is focused on models applicable to processing the larger amounts of (correlated) patterns. We develop the model of hierarchical associative memory (HAM), which improves the abilities of the traditional associative memories. In comparison with traditional models, the main advantage of the developed HAM model is

- ability to process any patterns (correlated/uncorrelated)
- no limit for the number of patterns to be stored in the model.

We would like to highlight the adaptability of our HAM model to incoming data. The structure of the model is adapted to the presented patterns. There is no need to state the number of local associative memories before the storage process starts. The local associative memories are automatically added to the corresponding layers of the HAM model during the storage process.

The storage and recall abilities of the HAM model can be further improved by restricting the storage capacity of every local associative memory. The experimental results confirm the legitimacy of the proposed HAM model and show the promising results for the area of processing a large number of (correlated) patterns. However, for real applications (e.g. in robot navigation) it will be necessary to further improve the robustness of the hierarchical associative memories with respect to the incomplete pattern recall.

9 Conclusion and future work

This chapter summarizes the work achieved in this thesis and its main results. In addition, several possible directions for further research are discussed.

9.1 Main results

Artificial neural networks can be applied in areas like navigation or cartography, in which processing large amounts of spatial data is required. Artificial neural networks can help to guide the robot in buildings and recall the scenery "behind a corner" or they can help to drive through a scenery where the traveller has been before etc. In this context, spatial patterns processed by artificial neural networks can be geographic maps, room plans, etc. Strategies for processing spatial patterns can be based on artificial neural networks called associative memories. In this thesis, we have focused on associative memory models applicable to storage and recall of large numbers of spatial patterns.

One goal of the thesis has been to study associative memory approaches to processing spatial patterns. We have presented and discussed several strategies how to store and recall spatial patterns (i.e. storing the whole image of the scenery, the autoassociative memory approach, the hierarchical associative memory approach and the heteroassociative memory approach). The main differences between the presented strategies consist in the applied models of associative memories. We have also discussed their applicability to problems, in which larger numbers of spatial patterns need to be processed.

Another goal of our work has been to study models of associative memories that could be applicable to storage and recall of spatial patterns. Traditional models of associative memories have robust storage/recall abilities, but they often suffer from too strong memory constraints for the storage process. The performance is very sensitive to the number of stored patterns and their mutual correlations. As the model should enable robust retrieval of large numbers of spatial patterns, traditional associative memory models are not very applicable to the problem.

Based on the studied associative memory models and their properties, we have proposed a new model of associative memories, called Hierarchical Associative Memory (HAM). The HAM model has been developed with a stressed necessity to process large amounts of spatial patterns. Inherently, the HAM model is able to handle an arbitrary number of patterns; uncorrelated as well as correlated. The structure of the HAM model varies with the incoming patterns and their mutual correlations.

Our hierarchical associative memories consist of associative memories grouped into several layers. In the thesis, we have presented two models of hierarchical associative memories - the HAM1 model and the HAM2 model. The HAM1 model stores patterns in "any suitable" associative memory of a layer. In the HAM2 model, patterns are hierarchically grouped and processed according to the information recalled by the "previous" layer. We have also implemented the HAM models. In experiments carried out, we have demonstrated the applicability of the proposed HAM models in tasks where larger amounts of (correlated)

patterns need to be processed.

The robust storage ability [13] has been observed while experimenting with the storage ability of the developed models. Most of all stored patterns have been recalled without any error. The storage ability has been enhanced in the HAM2 model due to the influence of the tree structure. Very good results have been achieved in comparison with traditional models of associative memories.

The recall ability has been analyzed with respect to the incomplete pattern recall. In our experiments, the incomplete patterns have been recalled with a small fraction of errors. With the increasing acceptable error, the number of incomplete patterns recalled correctly has been increased. When the "unknown" area in incomplete patterns has been sufficiently small, the HAM models have had relatively robust recall abilities. The experiments have also shown that the recall ability of the HAM2 model is higher than that of HAM1 model, but the differences are just minor.

The memory complexity of our hierarchical associative memory is characterized by the number of associative memories composing the model. The theoretically derived results have been compared with the experimental results. Experiments have shown that the local associative memories in the HAM1 model are used efficiently. The tree structure of the HAM2 model has increased the number of associative memories in comparison with the HAM1 model. The time complexity of the HAM models has been derived. Although the performance of the HAM models takes longer time than that of the traditional associative memories, the HAM models are applicable in practice due to their robust storage and recall abilities.

The hierarchical associative memories (in contrast to traditional associative memory models) have shown promising results for problems, in which larger amounts of patterns need to be processed. Although the proposed HAM models have been analyzed in detail only in the case of two layer hierarchy, the generalization to any number of layers is straightforward.

9.2 Future work

More research needs to be done in the development of more sophisticated methods for selecting a local associative memory, in which a pattern is stored. We have proposed a basic straightforward method that is not optimal. Some previously stored patterns can be recalled incorrectly or can even become lost after storing other patterns. A more sophisticated method could further increase the robustness of the model.

For real applications, it is necessary to further improve the recall abilities of the proposed model with respect to the incomplete pattern recall with larger unknown area. The right choice of an output pattern based on the patterns recalled by the corresponding local associative memories represents an important point in the recall process. The appropriately chosen pattern should represent "well enough" the corresponding set of local associative memories in the "next" layer of the hierarchy. In our opinion, strategies for achieving this could comprise "keys" corresponding in principle to further pattern elements artificially added

[13] The ability of the model to recall the stored patterns

to patterns. The keys introduced in such a way could then better control the recall process.

In the HAM models, the training patterns are needed for every layer. In practice, a set of spatial patterns (e.g. the scanned pictures) is available. Thus, it is necessary to process the presented patterns in order to create the corresponding training sets (one training set for one layer of the model). In the case of two layer hierarchy, patterns with the smallest cumulative correlation between the respective patterns are to be used to form training patterns for the first layer. The remaining patterns of the set are to be training patterns for the second layer. One should search for new (more sophisticated) methods of creating training sets. New methods could be based e.g. on self-organizing neural networks or clustering principles.

In future, comparison of the HAM models with other models of associative memories could be done as well.

References

[1] Abbott, L. F., Varela, J. A., Sen, K., Nelson, S. B.: Synaptic depression and cortical gain control, Science, Vol. 275, No. 5297, pp. 220-224, 1997

[2] Amari, S.: Characteristics of Sparsely Encoded Associative Memory, Neural Networks, Vol. 2, pp. 451–457, 1989

[3] Amit, D. J., Gutfreund, H., Sompolinsky, H.: Storing infinite numbers of patterns in a spin-glass model of neural networks, Physical Review Letters, Volume 55, Number 14, pp. 1530–1533, 1985

[4] Amit, D. J., Gutfreund, H., Sompolinsky, H.: Information storage in neural networks with low levels of activity, Physical Review A, 35, pp. 2293–2303, 1987

[5] Anděl, J.: Statistické metody, MATFYZPRESS, ISBN 80-86732-27-8, 2003

[6] Athithan, G., Dasgupta, C.: On the problem of spurious patterns in neural associative memory models, IEEE Transactions on Neural Networks, 8 (6), pp. 1483–1491, 1997

[7] Ballard, D. H.: Cortical connections and parallel processing: structure and function, Behavioral and Brain Sciences, 9, pp. 67–120, 1986

[8] Banquet, J. P., Gaussier, P., Quoy, P., Revel, A., Burnod, Y.: Spatial representation versus navigation through hippocampal, prefrontal and gangliobasal loops, in: International Joint Conference on Neural Networks, Budapest, Hungary, pp. 1499-1505, 2004

[9] Banquet, J. P., Gaussier, P., Quoy, P., Revel, A., Burnod, Y.: A hierarchy of associations in hippocampo-cortical systems: cognitive maps and navigation strategies, Neural Computation, 17, pp. 1339-1384, 2005

[10] Bibitchkov, D., Herrmann, J. M., Geisel, T.: Pattern storage and processing in attractor networks with short-time synaptic dynamics, Network, 13, pp. 115-129, 2002

[11] Duda, R. O., Hart, P. E., Stork, D. G.: Pattern Classification, Wiley, 2003

[12] Düring, A., Coolen, A. C. C., Sherrington, D.: Phase diagram and storage capacity of sequence processing neural network, Journal of Physics A, 31, pp. 8607–8621, 1998

[13] Elliffe, M., Rolls, E., Parga, N., Renart, A.: A recurrent model of transformation invariance by association, Neural Networks, Vol. 13, pp. 225-237, 2000

[14] Everett, H. R.: Sensors For Mobile Robots, ISBN 1-56881-048-2, A K Peters, 1995

[15] Feigelman, M. V. , Ioffe, L. B.: The augmented models of associative memory - asymetric interaction and hierarchy of patterns, International Journal of Modern Physics B, 1, pp. 51–68, 1987

[16] Floreén, P.: Computational complexity problems in neural associative memories, Doctoral dissertation, University of Helsinky, Department of Computer Science, Report A-1992-5, 1992

[17] Fogelman, F., Goles, E., Weisbuch, G.: Transient length in sequential iteration of threshold funtions, in: Discrete Applied Mathematics, 6, pp. 95–98, 1983

[18] Fontanari, J. F., Koberle, R.: Information Processing In Synchronous Neural Networks, J. Physique (France), 49, pp. 13–13, 1988

[19] Fukushima, K., Yamaguchi, Y., Okada, M.: Neural Network Model of Spatial Memory: Associative Recall of Maps, Neural Network, Vol. 10, No. 6, pp. 971–979, 1997

[20] Gardner, E.: Structure of metastable states in the Hopfield model, Journal of Physics A: Mathematical and General, Vol. 19, No. 16, 1986

[21] Guariglia, C., Padovani, A., Pantano, P., Pizzamiglio, L.: Unilateral neglect restricted to visual imagery, Nature, 364, pp. 235–237, 1993

[22] Gutfreund, H.: Neural networks with hierarchically correlated patterns, Physical Review A, Vol. 37, No. 2, pp. 570–577, 1988

[23] Haken, A.: Connectionist networks that need exponential time to stabilize, Department of Computer Science, University of Toronto, 1989

[24] Hebb, D.: The Organization of Behaviour, John Wiley, New York, 1949

[25] Hertz, J., Krogh, A., Palmer, R.: Introduction to the Theory of Neural Computation, Addison-Wesley, Redwood City, CA, 1991

[26] Hirahara, M., Oka, N., Kindo, T.: Associative memory with a sparse encoding mechanism for storing correlated patterns, in: Neural Networks, Vol. 10, pp. 1627–1636, 1997

[27] Hirahara, M., Oka, N., Kindo, T.: A cascade associative memory model with a hierarchical memory structure, Neural Networks, Vol. 13, No. 1, pp. 41–50, 2000

[28] Hirahara, M., Oka, N., Kindo, T.: Cascade associative memory storing hierarchically correlated patterns with various correlations, Neural Networks, Vol. 13, No. 1, pp. 51–61, 2000

[29] Hirtle, S. C., Jonides, J.: Evidence of hierarchies in cognitive maps, Memory and Cognition, 13(3), pp. 208–217, 1985

[30] Hopfield, J. J.: Neural Networks and Physical Systems with Emergent Collective Computational Abilities, Proceedings of the National Academy of Sciences, USA, 79, pp. 2554–2558, 1982

[31] Hopfield, J. J.: Neurons with graded response have collective computational properties like those of two-states neurons, Proceedings of the National Academy of Sciences, USA, 81, pp. 3088–3092, 1984

[32] Itoh, K., Shimizu, T.: New type of feed-back mechanism with bifurcation parameter modulation, American Institute of Physics Conference Proceedings, 519, pp. 649-651, 2000

[33] Itoh, K., Shimizu, T.: The virtual attractor in mutually coupled networks, Journal of the Korean Physical Society, 40(6), pp. 1018-1022, 2002

[34] Itoh, K., Miwab, H., Takanobud, H., Takanishia, A.: Application of neural network to humanoid robots-development of co-associative memory model, Neural Networks, Vol. 18, pp. 666-673, 2005

[35] Kakeya, H., Kindo, T.: Hierarchical concept formation in associative memory composed of neuro-window Elements, Neural Networks, Vol. 9, pp. 1095–1098, 1996

[36] Kawamura, M., Tokunaga, R., Okada, M.: Low-dimensional chaos induced by frustration in a non-monotonic system, in: Europhys. Lett., 62, pp. 657–663, 2003

[37] Kawamura, M., Tokunaga, R., Okada, M.: Bifurcation analysis in an associative memory model, in: Physical Review E, 70, 046210, 2004

[38] Kimoto, T., Okada, M.: Mixed state on a sparsely encoded associative memory model, Biological Cybernetics, Vol. 85, pp. 319–325, 2001

[39] Kimoto, T., Okada, M.: Mixed states on neural network with structural learning, Neural Networks, Vol. 17, pp. 103–112, 2004

[40] Kimoto, T., Okada, M.: Coexistence of memory patterns and mixed states in a sparsely encoded associative memory model storing ultrametric patterns, Biological Cybernetics, 90, pp. 229-238, 2004

[41] Komlós, J., Paturi, R.: Convergence results in an associative memory model, Neural networks, 1, pp. 239–250, 1988

[42] Kosko, B.: Bidirectional Associative Memories, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 18, pp. 49–60, 1988

[43] Matsumoto, N., Okada, M., Doya, K., Sugase, Y., Yamane, S.: Dynamics of face responsive neurons in the temporal cortex, Society for Neuroscience Abstracts, vol. 27, pp. 1048, 2001

[44] Matsumoto, N., Okada, M.: Neuronal mechanisms for hierarchical encoding in inferior-temporal cortex, Proceedings of CNS 2003, Alicante, Spain, 2003

[45] Matsumoto, N., Okada, M., Sugase, Y., Yamane, S., Kawano, K.: Population Dynamics of Face-responsive Neurons in the Inferior Temporal Cortex, Cerebral Cortex, Volume 15, Number 8, pp. 1103-1112, 2005

[46] Matsumoto, N., Ide, D., Watanabe, M., Okada, M.: Synaptic Depression Enlarges Basin of Attraction, Neurocomputing, 65-66, pp. 571–577, 2005

[47] McCulloch, W., Pitts, W.: A logical calculus of the ideas immanent in nervous activity, Bulletin of Mathematical Biophysics, 7, pp. 115–133, 1943

[48] McEliece, R. J., Posner, E. C., Rodemich, E. R., Venkatesh, S. S.: The capacity of the Hopfield associative memory, IEEE Transactions of Information Theory, 33, pp. 461–482, 1987

[49] McNamara, T. P., Hardy, J. K., Hirtle, S. C.: Subjective hierarchies in spatial memory, Journal of Experimental Psychology: Learning, Memory, and Cognition, 15(2), pp. 211–227, 1989

[50] Mézard, M., Nadal, J. P., Toulouse, G.: Solvable models of working memories, Journal de Physiques, 47, pp. 1457–1462, 1986

[51] Milner, P.: The Mind and Donald O. Hebb, Scientific American, Vol. 268, No.1, pp. 124–129, 1993

[52] Mimura, K., Okada, M., Kurata, K.: Associative memory model with forgetting process using nonmonotonic neurons, IEICE Trans., vol. E81-D, pp. 1298–1304, 1998

[53] Morita, M.: Associative Memory with Nonmonotone Dynamics, Neural Networks, Vol. 6, pp. 115–125, 1993

[54] Mrázová, I., Tesková (Štanclová), J.: Path Prediction in Geographic Maps, Proceedings of Nostradamus 2000, Zlín, Czech Republic, ISBN 80-214-1668-8, pp. 190–195, 2000

[55] Mrázová, I., Tesková (Štanclová), J.: Hierarchical Associative Memories for Storing Spatial Patterns, Proceedings of ITAT 2002, Malinô Brdo, Slovakia, ISBN 80-7097-499-0, pp. 143–154, 2002.

[56] Nadal, J. P., Toulouse, G., Changeux, J. P., Dehaeve, S.: Networks of formal neurons and memory palimpsests, Europhysics Letters, 1:535, 1986

[57] Hecht-Nielsen, R.: Neurocomputing, Addison Wesley, ISBN 0-201-09205-3, 1990

[58] Okada, M., Mimura, K., Kurata, K.: Associative memory with forgetting process - analysis by statistical neurodynamics, IEICE Trans., vol. J77-D-II, pp. 1178–1180, 1994

[59] Okada, M.: Notions of associative memory and sparse coding, Neural Networks, Vol. 9, pp. 1429–1458, 1996-

[60] Pantic, L., Torres, J. J., Kappen, H. J., Gielen, S. C. A. M.: Associative memory with dynamic synapses, Neural Computation, 14, pp. 2903-2923, 2002

[61] Parberry, I.: Circuit Complexity and Neural Networks, MIT Press, 1994.

[62] Parga, N., Rolls, E.: Transformation-invariant recognition by association in a recurrent network, Neural Computation, Vol. 10, pp. 1507–1525, 1998

[63] Peretto, P.: On learning rules and memory storage abilities for asymmetrical neural networks, Journal de Physique Lettres, 49, pp. 711–726, 1988

[64] Rojas, R.: Neural Networks: A systematic introduction, Springer Verlag, 1995

[65] Rutkowska, D.: Neuro-Fuzzy Architectures and Hybrid Learning, Physica-Verlag, Springer-Verlag Company, Heidelberg, New York, 2002

[66] Sharp, P.: Complementary roles for hippocampal versus subicular/entorhinal place cells in coding place, context, and events, Hippocampus, 9 (4), 432–443, 1999

[67] Shimizu, T.: Chaotic brownian network, Physica A, 256, pp. 163-177, 1998

[68] Singh, M. P.: Hopfield model with self-coupling, Physical Review E, 64, 051912, 2001

[69] Sugase, Y., Yamane, S., Ueno, S., Kawano, K.: Global and fine information coded by single neurons in the temporal visual cortex, Nature, 400, pp. 869-873, 1999

[70] Šíma, J., Neruda, R.: Teoretické otázky neuronových sítí. Praha, MATFYZPRESS, 1996

[71] Štanclová, J.: The Associative Recall of Spatial Correlated Patterns, Proceedings of CIARP 2006, Cancun, Mexico, Copyright (C) Springer-Verlag, Berlin, LNCS 4225, ISSN 0302-9743, ISBN 978-3-540-46556-0, pp. 539–548, 2006

[72] Štanclová, J.: Asociativní paměti pro ukládání korelovaných vzorů (in czech), Proceedings of ITAT 2006, Bystrá dolina, Slovakia, ISBN 80-969184-4-3, pp. 173–178, 2006

[73] Štanclová, J., Zavoral, F.: Hierarchical Associative Memories: The Neural Network for Prediction in Spatial Maps, Proceedings of ICIAP 2005, Cagliary, Italy, Copyright (C) Springer-Verlag, Berlin, LNCS3617, ISSN-0302-9743, ISBN 3-540-28869-4, pp. 786–793, 2005

[74] Štanclová, J., Obdržálek, D.: Map Recall based on Hierarchical Associative Memories, Proceedings of DARH 2005, Yverdon-les-Bains, Switzerland, ISBN 2-8399-0085-8, pp. 44–49, 2005

[75] Štanclová, J., Obdržálek, D.: Použití Hierarchického asociativního modelu neuronové sítě pro zlepšení lokalizace autonomního robota (in czech), Proceedings of ITAT 2005, Račkova dolina, Slovakia, ISBN 80-7097-609-8, pp. 157–166, 2006

[76] Tesková (Štanclová), J.: Hierarchical Associative Memories for Path Prediction, Proceeding of SSGRR 2003, L'Aquila, Italy, Telecom Italia Learning Services S.p.A., ISBN 88-85280-74-9, 2003

[77] Tesková (Štanclová), J.: Hierarchical Associative Memories: The Parallel Implementation of The Model, Proceedings of IPSI-2003, Sveti Stefan, Serbia and Montenegro, ISBN 86-7466-117-3, 2003

[78] Tesková (Štanclová), J.: Associative Recall by Means of Hierarchical Associative Memories, Proceedings of CALCI 2003, Stará Lesná, Slovakia, ISBN 80-89066-64-X, pp. 369–376, 2003

[79] Tesková (Štanclová), J.: Implementation of Hierarchical Associative Memories, Proceedings of WDS 2002, Praha, Czech Republic, MatfyzPress ISBN 80-85863-88-X, pp. 9–16, 2002

[80] Tesková (Štanclová), J.: Associative Memories for Path Prediction, Proceedings of WDS 2001, Praha, Czech Republic, ISBN 80-85863-73-1, pp. 198–204, 2001

[81] Tesková (Štanclová), J.: Associative Recall of Spatial Maps by Means of Hopfield Model, Proceedings of Nostradamus 2001, Zlín, Czech Republic, ISBN 80-7318-030-8, 2001

[82] Torres, J. J., Pantic, L., Kappen, H.J.: Storage capacity of attractor neural networks with depressing synapses, Physical Review E, 66, 061910, 2002

[83] Toya, K., Fukushima, K., Kabashima, Y., Okada, M.: Bistability of mixed states in a neural network storing hierarchical patterns, Journal of Physics A: Mathematical and General, 33, pp. 2725-2737, 2000

[84] Triola, M. F.: Elementary Statistics, The Benjamin/Cummings Publishing Company, Inc., ISBN 0-8053-0271-9, 1989

[85] Tsodyks, M. V., Markram, H.: The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability, Procedings of the National Academy of Science USA, Vol. 94, pp. 719-723, 1997

[86] Voicu, H., Schmajuk, N. A.: Exploration, navigation and cognitive mapping, Adaptive Behavior, 8(3/4), pp. 207–223, 2000

[87] Voicu, H.: Hierarchical cognitive maps, Neural Networks, Vol. 16, pp. 569–576, 2003

[88] Weisbuch, G., Fogelman-Soulié, F.: Scaling laws for the attractors of Hopfield networks, Journal de Physique Lettres, 46, pp. 623–630, Paris, 1985

[89] Yoshizawa, S., Morita, M., Amari, S.: Capacity of Associative Memory Using a Nonmonotonic Neuron Model, Neural Networks, Vol. 6, pp. 167–176, 1993

Appendix

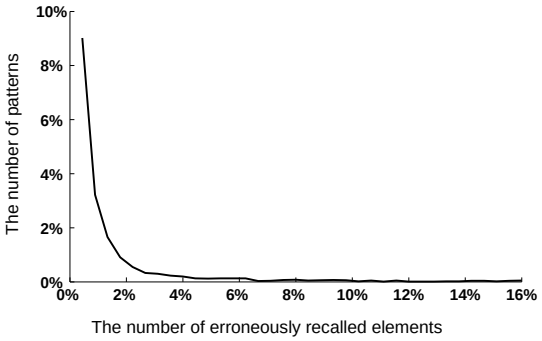


Figure 55: The histogram of the error in the patterns recalled incorrectly by the HAM1 model (the capacity coefficient $c = 1.0$ and the pattern rate $r = 1$)

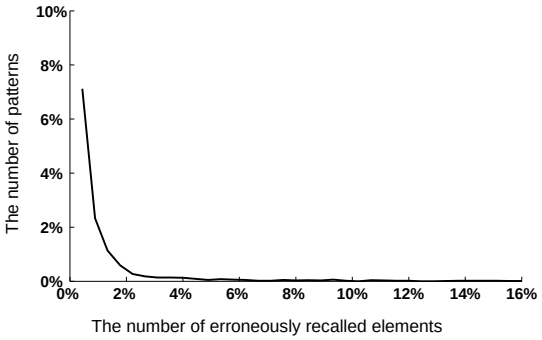


Figure 56: The histogram of the error in the patterns recalled incorrectly by the HAM2 model (the capacity coefficient $c = 1.0$ and the pattern rate $r = 1$)

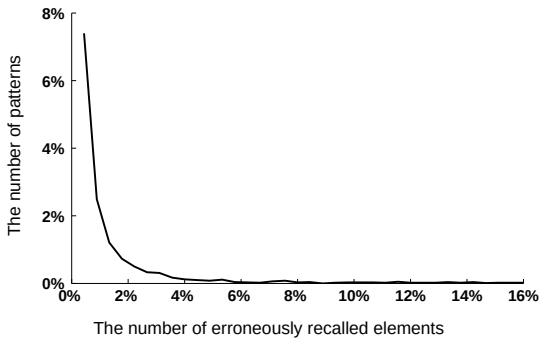


Figure 57: The histogram of the error in the patterns recalled incorrectly by the HAM1 model (the capacity coefficient $c = 0.9$ and the pattern rate $r = 1$)

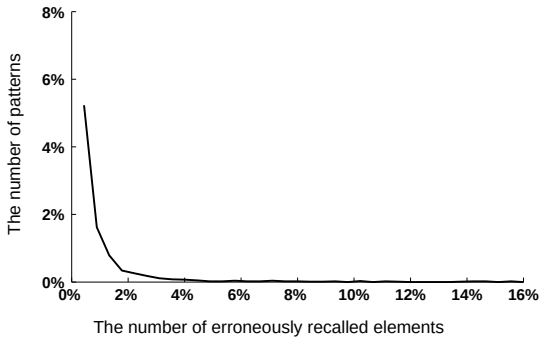


Figure 58: The histogram of the error in the patterns recalled incorrectly by the HAM2 model (the capacity coefficient $c = 0.9$ and the pattern rate $r = 1$)

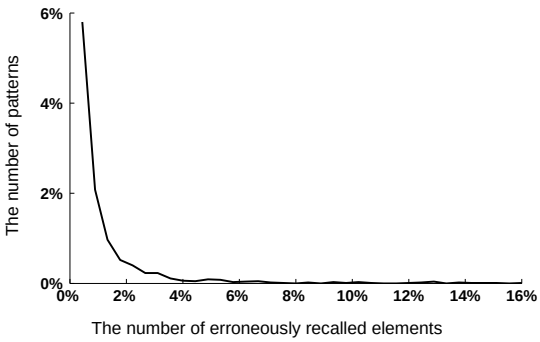


Figure 59: The histogram of the error in the patterns recalled incorrectly by the HAM1 model (the capacity coefficient $c = 0.8$ and the pattern rate $r = 1$)

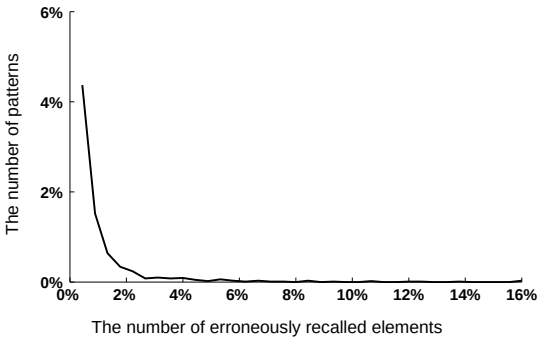


Figure 60: The histogram of the error in the patterns recalled incorrectly by the HAM2 model (the capacity coefficient $c = 0.8$ and the pattern rate $r = 1$)

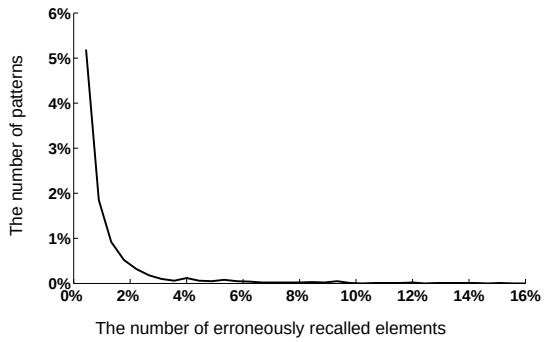


Figure 61: The histogram of the error in the patterns recalled incorrectly by the HAM1 model (the capacity coefficient $c = 0.7$ and the pattern rate $r = 1$)

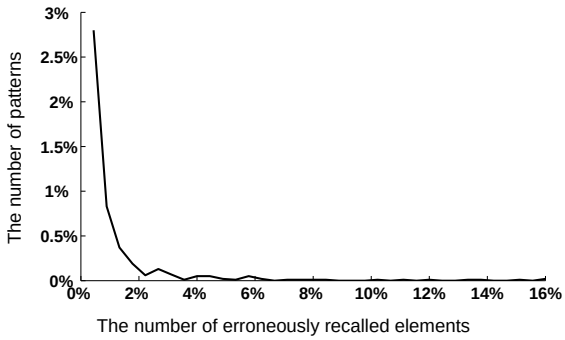


Figure 62: The histogram of the error in the patterns recalled incorrectly by the HAM2 model (the capacity coefficient $c = 0.7$ and the pattern rate $r = 1$)

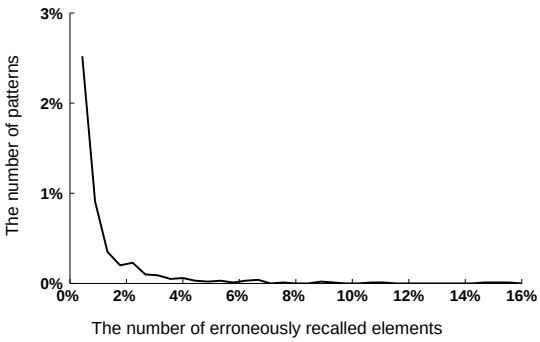


Figure 63: The histogram of the error in the patterns recalled incorrectly by the HAM1 model (the capacity coefficient $c = 0.6$ and the pattern rate $r = 1$)

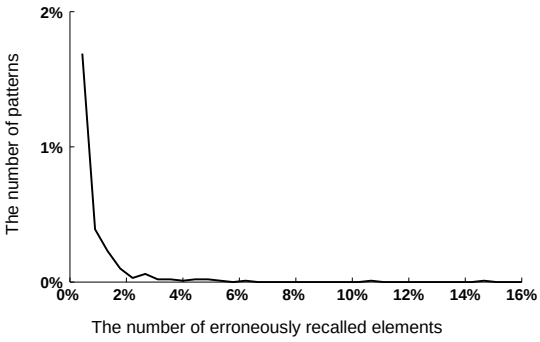


Figure 64: The histogram of the error in the patterns recalled incorrectly by the HAM2 model (the capacity coefficient $c = 0.6$ and the pattern rate $r = 1$)

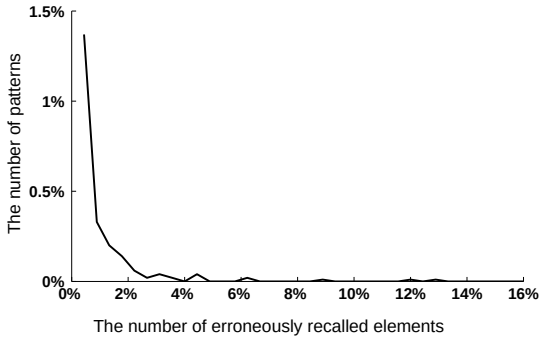


Figure 65: The histogram of the error in the patterns recalled incorrectly by the HAM1 model (the capacity coefficient $c = 0.5$ and the pattern rate $r = 1$)

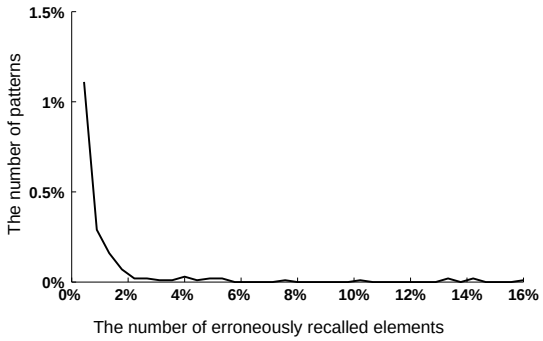


Figure 66: The histogram of the error in the patterns recalled incorrectly by the HAM2 model (the capacity coefficient $c = 0.5$ and the pattern rate $r = 1$)

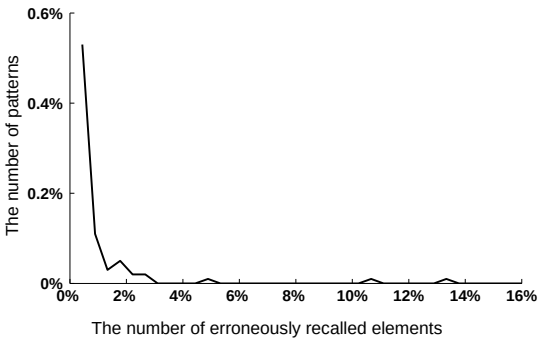


Figure 67: The histogram of the error in the patterns recalled incorrectly by the HAM1 model (the capacity coefficient $c = 0.4$ and the pattern rate $r = 1$)

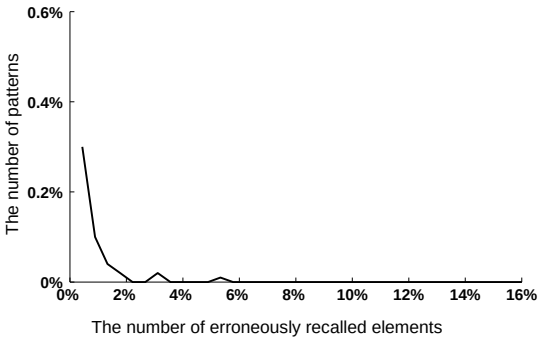


Figure 68: The histogram of the error in the patterns recalled incorrectly by the HAM2 model (the capacity coefficient $c = 0.4$ and the pattern rate $r = 1$)

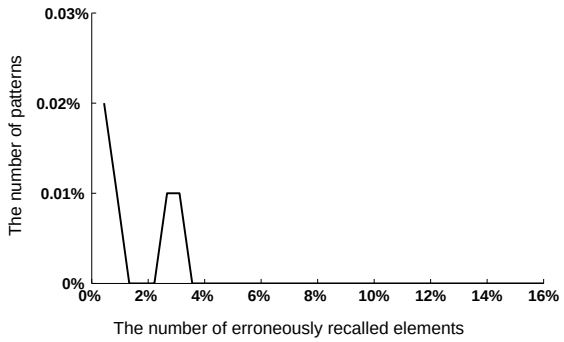


Figure 69: The histogram of the error in the patterns recalled incorrectly by the HAM1 model (the capacity coefficient $c = 0.3$ and the pattern rate $r = 1$)

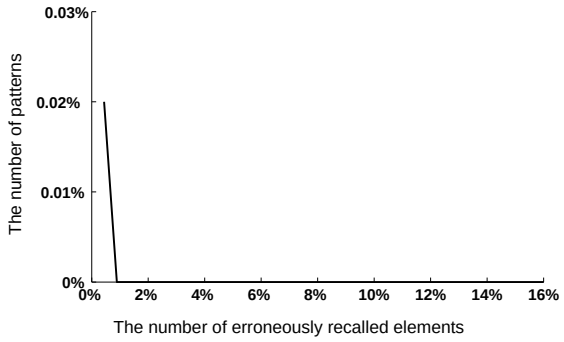


Figure 70: The histogram of the error in the patterns recalled incorrectly by the HAM2 model (the capacity coefficient $c = 0.3$ and the pattern rate $r = 1$)

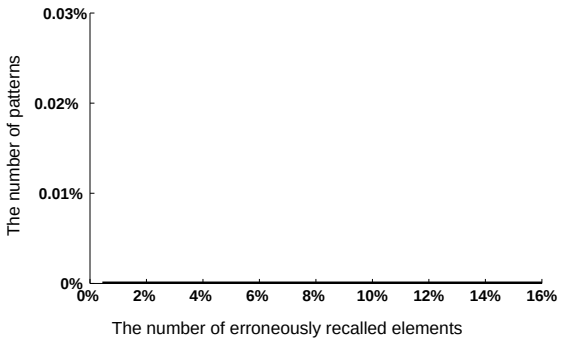


Figure 71: The histogram of the error in the patterns recalled incorrectly by the HAM1 model (the capacity coefficient $c = 0.2$ and the pattern rate $r = 1$)

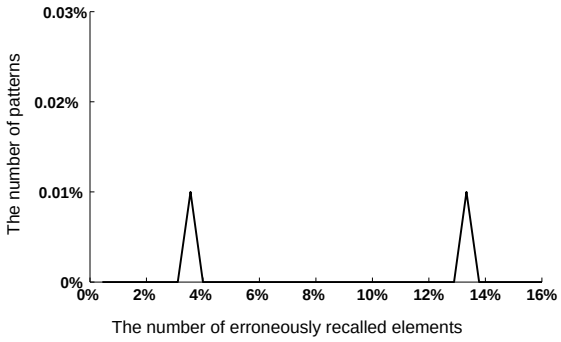


Figure 72: The histogram of the error in the patterns recalled incorrectly by the HAM2 model (the capacity coefficient $c = 0.2$ and the pattern rate $r = 1$)

The shift by 1		
	The HAM1 model	The HAM2 model
The acceptable error e_{accept}	$r \sim 1/5$ $c = 0.4$ [99.25%]	$r \sim 1/5$ $c = 0.4$ [99.37%]
$\leq 0\%$	42.18%	42.22%
$\leq 1\%$	45.82%	45.79%
$\leq 2\%$	50.69%	50.66%
$\leq 3\%$	56.10%	56.04%
$\leq 4\%$	63.74%	63.54%
$\leq 5\%$	69.41%	69.03%
$\leq 6\%$	77.68%	76.81%
$\leq 7\%$	87.27%	86.56%
$\leq 8\%$	97.40%	97.12%
$\leq 9\%$	99.50%	99.46%
$\leq 10\%$	99.96%	99.93%
$\leq 11\%$	100.00%	100.00%

Table 12: The recall ability of the HAM1 and HAM2 models for the diagonal shift by 1 point: the relative number of incomplete patterns recalled with the error less than or equal to the acceptable error e_{accept}

The shift by 2		
	The HAM1 model	The HAM2 model
The acceptable error e_{accept}	$r \sim 1/5$ $c = 0.4$ [99.25%]	$r \sim 1/5$ $c = 0.4$ [99.37%]
$\leq 0\%$	35.24%	35.19%
$\leq 1\%$	36.36%	36.29%
$\leq 2\%$	37.78%	37.80%
$\leq 3\%$	39.76%	39.72%
$\leq 4\%$	42.88%	42.91%
$\leq 5\%$	45.47%	45.53%
$\leq 6\%$	48.52%	48.53%
$\leq 7\%$	51.64%	51.64%
$\leq 8\%$	55.55%	55.51%
$\leq 9\%$	58.06%	57.87%
$\leq 10\%$	60.98%	60.63%
$\leq 11\%$	65.69%	65.07%

Table 13: The recall ability of the HAM1 and HAM2 models for the diagonal shift by 2 points: the relative number of incomplete patterns recalled with the error less than or equal to the acceptable error e_{accept}

The shift by 3		
	The HAM1 model	The HAM2 model
The acceptable error e_{accept}	$r \sim 1/5$ $c = 0.4$ [99.25%]	$r \sim 1/5$ $c = 0.4$ [99.37%]
$\leq 0\%$	30.74%	30.69%
$\leq 1\%$	31.68%	31.59%
$\leq 2\%$	32.33%	32.23%
$\leq 3\%$	33.11%	33.00%
$\leq 4\%$	34.53%	34.40%
$\leq 5\%$	35.78%	35.68%
$\leq 6\%$	37.01%	36.97%
$\leq 7\%$	38.36%	38.36%
$\leq 8\%$	41.13%	41.24%
$\leq 9\%$	43.37%	43.38%
$\leq 10\%$	45.48%	45.54%
$\leq 11\%$	47.55%	47.58%

Table 14: The recall ability of the HAM1 and HAM2 models for the diagonal shift by 3 points: the relative number of incomplete patterns recalled with the error less than or equal to the acceptable error e_{accept}

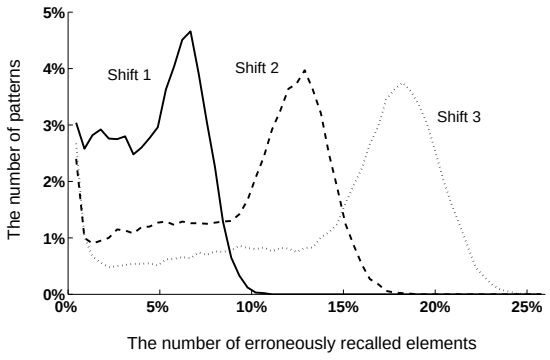


Figure 73: The histogram of the error in the incomplete patterns recalled by the HAM1 model ($c = 0.8$ and $r \sim 1/2$) for three types of the diagonal shift: the shift by 1 (the solid line), the shift by 2 (the dashed line) and the shift by 3 (the dotted line)

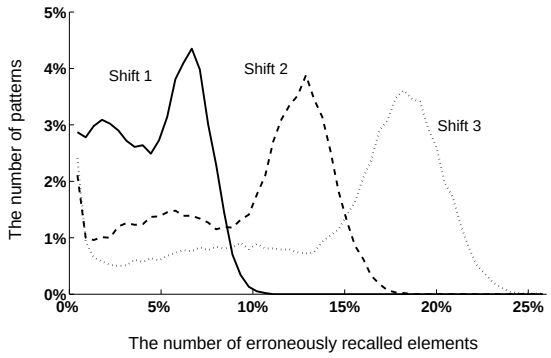


Figure 74: The histogram of the error in the incomplete patterns recalled by the HAM2 model ($c = 0.8$ and $r \sim 1/2$) for three types of the diagonal shift: the shift by 1 (the solid line), the shift by 2 (the dashed line) and the shift by 3 (the dotted line)

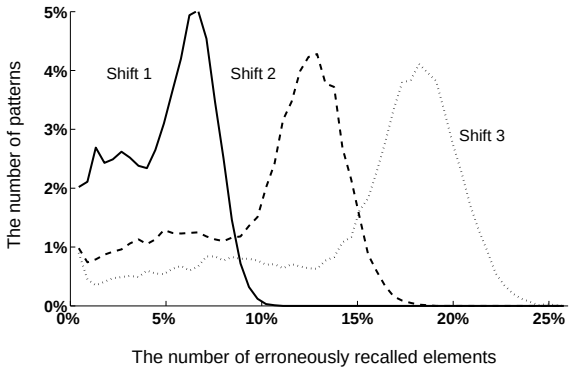


Figure 75: The histogram of the error in the incomplete patterns recalled by the HAM1 model ($c = 0.5$ and $r = 1/3$) for three types of the diagonal shift: the shift by 1 (the solid line), the shift by 2 (the dashed line) and the shift by 3 (the dotted line)

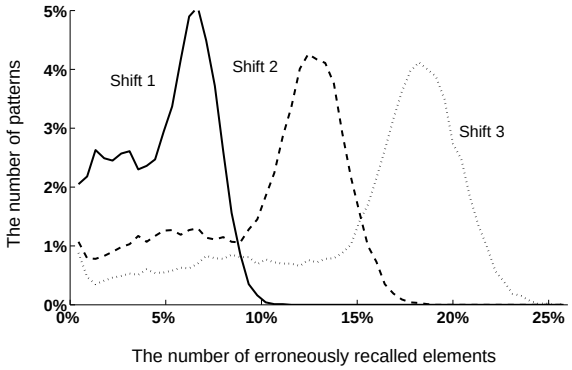


Figure 76: The histogram of the error in the incomplete patterns recalled by the HAM2 model ($c = 0.5$ and $r = 1/3$) for three types of the diagonal shift: the shift by 1 (the solid line), the shift by 2 (the dashed line) and the shift by 3 (the dotted line)

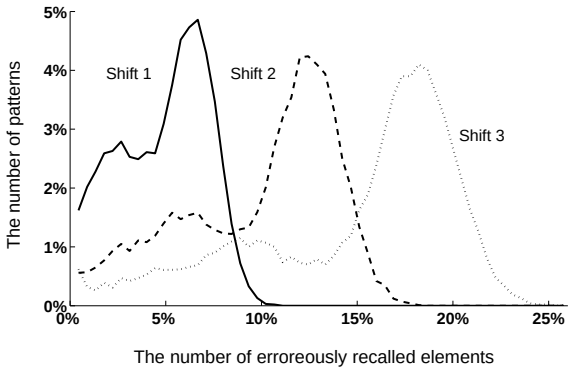


Figure 77: The histogram of the error in the incomplete patterns recalled by the HAM1 model ($c = 0.4$ and $r \sim 1/5$) for three types of the diagonal shift: the shift by 1 (the solid line), the shift by 2 (the dashed line) and the shift by 3 (the dotted line)

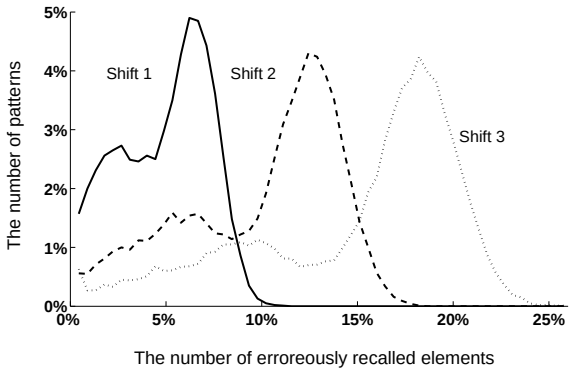


Figure 78: The histogram of the error in the incomplete patterns recalled by the HAM2 model ($c = 0.4$ and $r \sim 1/5$) for three types of the diagonal shift: the shift by 1 (the solid line), the shift by 2 (the dashed line) and the shift by 3 (the dotted line)