Entropic Associative Memory for Manuscript Symbols

Rafael Morales

University of Guadalajara

Nóe Hernández

National Autonomous University of Mexico

Ricardo Cruz

Consejo Nacional de Ciencia y Tecnología

Victor D. Cruz

Posgrado en Ciencia e Ingeniería de la Computación, UNAM

Luis A. Pineda (✉ lpineda@unam.mx)

National Autonomous University of Mexico

Research Article

Keywords: Manuscript symbols, Synthesis

Posted Date: September 1st, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-846238/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Entropic Associative Memory for Manuscript Symbols

Rafael Morales¹, Noé Hernández^{2,3}, Ricardo Cruz^{2,4}, Victor D. Cruz³, and Luis A. Pineda^{2,3,*}

¹Universidad de Guadalajara, SUV, Guadalajara, 44130, México

²Universidad Nacional Autónoma de México (UNAM), IIMAS, México City, 04510, México

³Posgrado en Ciencia e Ingeniería de la Computación, UNAM, México City, 04510, México

⁴Catedrático CONACyT, México City, México

*lpineda@unam.mx

ABSTRACT

Manuscript symbols can be stored, recognized and retrieved from an entropic digital memory that is associative and distributed but yet declarative; memory retrieval is a constructive operation; symbols not contained in the memory are rejected directly without search; and memory operations can be performed through parallel computations. Manuscript symbols, both letters and numerals, are stored in Associative Memory Registers that have an associated entropy. The memory recognition operation obeys an entropy trade-off between precision and recall, and the entropy level impacts on the quality of the objects recovered through the memory retrieval operation. We discuss the operational characteristics of the system for retrieving objects with both complete and incomplete information, such as severe occlusions. The experiments reported in this paper add evidence that supports the scalability of the framework and its potential for developing practical applications. We also compare the present entropic associative memories to Hopfield's paradigm and discuss its potential for the study of natural memory.

1 Associative Memory

Natural memories¹⁻³ are associative, distributed, declarative, constructive, and reject cues of objects that have not been stored previously; and the memory register, recognition and retrieval operations are performed in parallel in a natural way. In previous work⁴ we presented an Entropic Associative Memory (EAM) that has such putative properties of natural memories. The memory system uses Associative Memory Registers (AMRs) that hold distributed representations of objects of the same class, and have an associated entropy. This parameter measures the operational characteristics of the AMRs, which conform to an entropy trade-off. EAM was tested using a database of manuscript digits for complete and severely incomplete memory cues, with satisfactory results.

This memory opposes associative memories developed within the artificial neural networks paradigm⁵⁻¹⁰ that were consolidated with Hopfield's model¹¹ and subsequent work¹²⁻¹⁷. Models in this latter framework are also associative and distributed –although do not generalize– but they are reproductive rather than constructive; and sub-symbolic rather than declarative. In these models, the memory retrieval operation is always successful, and produce the object previously stored that is most similar to the memory cue –according to some abstract similarity measure– but never rejects a cue, unless the search is limited and fails, implementing a form of the so-called closed-world assumption. On a more fundamental level, while Hopfield's model is a dynamical physical system involving sub-symbolic computations, the present proposal uses standard tables or buffers that hold representations directly, where information is registered, recognized and retrieved through declarative manipulations, without search, alike in this respect to standard Random Access Memories (RAMs) of digital computers.

In this paper we present a set of experiments in which manuscript letters and digits are stored, recognized and retrieved from AMRs with both incomplete and severely occluded cues, adding evidence of the viability of entropic associative memories for the construction of practical systems, and the development of models of natural memory.

2 Relational-Indeterminate Computing

The present associative memory system is defined with a novel mode of computing that is referred to here as Relational-Indeterminate Computing (RIC)¹⁸⁻²⁰. In this formalism, *evaluating a relation* is construed as selecting randomly one among the values associated to the given argument. In the same way that “ $f(a_i) = v_j$ ” is interpreted as stating that the value of the function f for the argument a_i is v_j , “ $r(a_i) = v_j$ ” states that the value of the relation r for the argument a_i is an object v_j that is selected randomly –with an appropriate distribution– among the values associated to the argument a_i in the relation.

The specification is taken directly from our previous work⁴ with a minor reformulation of the β operation, as follows:

Let the sets $A = \{a_1, \dots, a_n\}$ and $V = \{v_1, \dots, v_m\}$, of cardinalities n and m , be the domain and the codomain of a finite relation $r : A \rightarrow V$. The objects in the domain and codomain are referred to here as the *arguments* and the *values* respectively. For purposes of notation, for any relation r we define a function $R : A \times V \rightarrow \{0, 1\}$ –the relation in lower case and the function in upper case letters– such that $R(a_i, v_j) = 1$ or *true* if the argument a_i is related to the value v_j in r , and $R(a_i, v_j) = 0$ or *false*, otherwise.

RIC has three basic operations: *abstraction*, *containment* and *reduction*. Let r_f and r_a be two arbitrary relations from A to V , and f_a be a function from A into V . The operations are defined as follows:

- Abstraction: $\lambda(r_f, r_a) = q$, such that $Q(a_i, v_j) = R_f(a_i, v_j) \vee R_a(a_i, v_j)$ for all $a_i \in A$ and $v_j \in V$ –i.e., $\lambda(r_f, r_a) = r_f \cup r_a$.
- Containment: $\eta(r_a, r_f)$ is true if $R_a(a_i, v_j) \rightarrow R_f(a_i, v_j)$ for all $a_i \in A$ and $v_j \in V$ (i.e., material implication), and false otherwise.
- Reduction: $\beta(f_a, r_f) = f_v$ such that if $\eta(f_a, r_f)$ holds $f_v(a_i) \in r_f(a_i)$ for all a_i , where $f_v(a_i)$ is selected randomly, using a distribution centered around f_a . If $\eta(f_a, r_f)$ does not hold, $\beta(f_a, r_f)$ is undefined –i.e., $f_v(a_i)$ is undefined– for all a_i .

Relations have an associated entropy, which is defined as the average indeterminacy of the relation r . Let μ_i be the number of values assigned to the argument a_i in r ; let $v_i = 1/\mu_i$ and n the number of arguments in the domain. In case r is partial, we define $v_i = 1$ for all a_i not included in r . The *computational entropy* $e(r)$ –or the entropy of a relation– is defined here as:

$$e(r) = -\frac{1}{n} \sum_{i=1}^n \log_2(v_i).$$

A function is a relation that has at most one value for any of its arguments, and its entropy is zero. Partial functions do not have a value for all the arguments, but this is fully determined and the entropy of partial functions is also zero.

3 Memory Operations

The associative memory consists on a set of *Associative Memory Registers* (AMRs) and the machinery to register, recognize and retrieve the information from and to the external environment. AMRs consist of tables with n columns and m rows, containing a relation between a set of arguments $A = \{a_1, \dots, a_n\}$ and a set of values $V = \{v_1, \dots, v_m\}$. Let $[R_k]^t$ be the content of the register R_k at time t and $\leftarrow$ an assignment operator such that $R_k \leftarrow R_j$ assigns $[R_j]^t$ to $[R_k]^{t+1}$, where j and k may be equal. The machine also includes the conditional operator *if* relating a condition *pred* to the operations op_1 and op_2 –i.e., if *pred* then op_1 else op_2 , where op_2 is optional. The initialization of a register R such that all its cells are set to 0 or to 1 is denoted $R \leftarrow 0$ or $R \leftarrow 1$ respectively, and $R \leftarrow f$ denotes that a function $f : A \rightarrow V$ is input into the register R . The system also includes the operators λ , η and β for computing the corresponding operations. These are all the operations in table computing.

Let R_k be an associative memory register and $R_{k-i/o}$ an auxiliary input and output register, both of size $n \times m$. The memory register, recognition and retrieval operations are implemented by the algorithms:

- *Memory_Register*(f_i, R_k):

1. $R_{k-i/o} \leftarrow f_i$
2. $R_k \leftarrow \lambda(R_k, R_{k-i/o})$
3. $R_{k-i/o} \leftarrow 0$

- *Memory_Recognize*(f, R_k):

1. $R_{k-i/o} \leftarrow f$
2. If $\eta(R_{k-i/o}, R_k)$ then $(R_{k-i/o} \leftarrow 1)$

- *Memory_Retrieve*(f, R_k):

1. $R_{k-i/o} \leftarrow f$
2. If $\eta(R_{k-i/o}, R_k)$ then $R_{k-i/o} \leftarrow \beta(R_{k-i/o}, R_k)$ else $R_{k-i/o} \leftarrow 0$

Figure 1. Training the Analysis and Synthesis modules

4 Analysis and Synthesis

The information registered, recognized and retrieved from the memory is presented to the AMRs as functions with n arguments whose codomain is a set with cardinality m . The input and output information is placed in modality specific buffers and mapped into its corresponding abstract representation and vice versa through an analysis and a synthesis module, respectively. The mapping is performed through an autoencoder^{21,22} including an encoder and a decoder²³, that implement the analysis and synthesis modules, respectively.

The encoder is implemented with a VGG5-like neural network with ten convolutional layers²⁴, and the decoder with a transposed convolutional neural network with four layers. The classifier is a fully connected neural network (FCNN) with two layers, mapping sets of features output by the encoder into their corresponding c classes. The purpose of the classifier is to establish a base-line to assess the performance of the AMRs, but it plays no active role in the memory operations and could be removed. The input to the encoder consists of p features, that correspond to the information in the input buffer, as illustrated in Figure 1. The encoder maps such concrete representation into n outputs, which constitutes the input to the AMRs directly –i.e., a function of n arguments with their corresponding values. This set of features is also the input to the decoder, that computes an approximation of the inverse function of the encoder, and renders p features, regenerating the concrete image. The encoder, the decoder and the classifier are trained simultaneously in a supervised manner by standard back-propagation, and the latter is removed once the autoencoder has been trained. Autoencoders were originally proposed to reduce the dimensionality of the data^{21,22}, and the present use constitutes a novel application of such technology.

The analysis and synthesis modules support the memory register, memory recognition and memory retrieve algorithms, which are carried on directly on the AMRs and their corresponding auxiliary registers in two or three computing step, as follows:

- *Memory_Register*(f_i, R_k): The register AMR_k is set on, and the remaining AMRs are set off; the input symbol is read through the analysis module; the *Memory_Register* operation is performed.
- *Memory_Recognize*(f_i, R_k): All AMRs are set on; the memory cue is read through the analysis module; the *Memory_Recognize* operation is performed; if no AMR's recognition operation is successful the object is rejected.
- *Memory_Retrieve*(f_i, R_k): All AMRs are set on; the memory cue is read through the analysis module; the *Memory_Retrieve* operation is performed; all AMRs but the one selected by the cue are set off; the recovered object is placed on the output buffer through the synthesis module.

5 A Visual Memory for Hand Written Manuscript Symbols

The associative memory system was tested in previous work⁴ through the construction of a visual memory for storing and retrieving distributed representations of hand written digits from “0” to “9”. The system was built and tested using the MNIST

Class number	Class name	Class instances	Class number	Class name	Class instances	Class number	Class name	Class instances
0	0	000	16	G	GGG	32	W	WWW
1	1	111	17	H	HHH	33	X	XXX
2	2	222	18	I	III	34	Y	YYY
3	3	333	19	J	JJJ	35	Z	ZZZ
4	4	444	20	K	KKK	36	a	aaa
5	5	555	21	L	LLL	37	b	bbb
6	6	666	22	M	MMM	38	d	ddd
7	7	777	23	N	NNN	39	e	eee
8	8	888	24	O	OOO	40	f	fff
9	9	999	25	P	PPP	41	g	ggg
10	A	AAA	26	Q	QQQ	42	h	hhh
11	B	BBB	27	R	RRR	43	n	nnn
12	C	CCC	28	S	SSS	44	q	qqq
13	D	DDD	29	T	TTT	45	r	rrr
14	E	EEE	30	U	UUU	46	t	ttt
15	F	FFF	31	V	VVV			

Figure 2. EMNIST-47 Alphabet

database available at <http://yann.lecun.com/exdb/mnist/>. In the present work we add evidence of the potential of the framework for the construction of practical applications. We use now the EMNIST²⁵ database including manuscript capital and lower case letters, and also the 10 digits, increasing the number of classes in relation to MNIST from 10 to 62; however, capital and lower case letters with very similar visual shapes are further merged, rendering 47 classes. Here we define the alphabet EMNIST-47, as shown in Figure 2. As can be seen, there are 11 lower case letters that can be clearly distinguished from their upper case counterparts –i.e., classes 36 to 46– and have independents entries. For the experiments we used the EMNIST Balanced segment including 2,800 instances of each class.

In previous work we also showed that an Associative Memory Register can hold the representation of more than one class and yet the system has a satisfactory performance. Here we capitalize such functionality and also model a memory in which capital and lower case letters of the same type that have very different visual shapes are held in the same memory register. As a result, classes 36 to 46 of EMNIST-47 are dropped, and the total number of classes is reduced to 36. This second alphabet is referred to as EMNIST-36, and it is shown in Figure 3.

The corpus was partitioned for the experiments in both settings in three disjoint sets. The partitions were rotated through a standard 10-fold cross-validation procedure. The partitions names and the assigned amount of data are as follows:

- Training Corpus (*TrainCorpus*): For training the analysis and synthesis modules (57 %).
- Remembered Corpus (*RemCorpus*): For filling in the Associative Memory Registers (33 %).
- Test Corpus (*TestCorpus*): For testing (10 %).

Preliminary experiments were performed to find out the optimal value of n , i.e., the number of inputs of the AMRs, using integers powers of 2. This was set to 64 for all the experiments. Developing on our previous work, the entropic associative memory for manuscript symbols was tested with four experiments as follows:

1. Experiment 1: Define an associative memory system including an AMR for holding the distributed representation of each one of the forty seven manuscript symbols of EMNIST-47. Determine the recognition precision and recall of the individual AMRs, and of the overall system, for AMRs of size $n \times 2^m$. The experiment was performed for $0 \leq m \leq 9$. Identify the parameter m of the AMRs with satisfactory performance. Finally, determine the precision and recall of the memory recognition when the AMRs contain different amounts of remembered instances and, consequently, different level of entropy.
2. Experiment 2: Determine the value of m of AMRs holding the distributed representation of capital and lower case letters of EMNIST-36. Determine the recognition precision and recall of the individual AMRs and of the overall system, for

Class number	Class name	Number of instances	Class instances	Class number	Class name	Number of instances	Class instances
0	0	2,800		18	I	2,800
1	1	2,800		19	J	2,800
2	2	2,800		20	K	2,800
3	3	2,800		21	L	2,800
4	4	2,800		22	M	2,800
5	5	2,800		23	N	5,600
6	6	2,800		24	O	2,800
7	7	2,800		25	P	2,800
8	8	2,800		26	Q	5,600
9	9	2,800		27	R	5,600
10	A	5,600		28	S	2,800
11	B	5,600		29	T	5,600
12	C	2,800		30	U	2,800
13	D	5,600		31	V	2,800
14	E	5,600		32	W	2,800
15	F	5,600		33	X	2,800
16	G	5,600		34	Y	2,800
17	H	5,600		35	Z	2,800

Figure 3. EMNIST-36 Alphabet

$0 \leq m \leq 9$ as before. Determine the precision and recall of the memory recognition when the AMRs contain different level of entropy, as in experiment 1.

- Experiment 3: Retrieve objects out of a cue for different levels of entropy and generate their corresponding images –with the best AMRs found in experiment 1. Assess the similarity between the cue and the recovered object at different levels of entropy.
- Experiment 4: Retrieve letters and digits of significantly occluded objects with the same AMR size used in Experiment 3. Assess the precision and recall of the memory retrieval operation and the quality of the generated images.

The source code for replicating the experiments, including the detailed results and the specifications of the hardware used, are available in Github at <https://github.com/eam-experiments/EMNIST>.

5.1 Experiment 1

Compute the characteristics of AMR of size 64×2^m for $0 \leq m \leq 9$:

- Register the totality of *RemCorpus* in their corresponding register through the *Memory_Register* operation;
- Test the recognition performance of all the instances of the test corpus through the *Memory_Recognize* operation;
- Compute the average precision, recall and entropy of individual memories.
- Select a unique object to be recovered by the *Memory_Retrieve* operation; compute the average precision and recall of the integrated system when this choice has been made.
- Select the parameter m with the best trade-off between precision and recall. Determine the performance of the system with such memory size $n \times 2^m$, for different amounts of the *RemCorpus* and entropy levels.

The average precision, recall and entropy of the ten AMRs are shown in Figure 4 (a). Precision and recall are computed in the standard way as follows: $Precision = TP/(TP + FP)$; $Recall = TP/(TP + FN)$, where TP , FP and FN stand for the total positives, the false positives and the false negatives, respectively. As can be seen, precision is very low for small values of m but grows very quickly with the number of rows, i.e., 2^m . Recall, on its part, is very high for low values of m , as the information is confused when the number of rows is very low, and all instances are accepted. However, when the value of m is increased, the grid is made finer, true instances are missed, and the recall is lowered. The optimal value of m is a trade-off between the precision and recall graphs. Figure 4 (a) shows that there is a good compromise at $m = 6$ and $m = 7$, i.e., for 64 and 128 rows. The figures also show the entropy at the bottom bar. It is increased almost linearly with the AMRs size, starting from 0 where the relations have only one value and is maximal for the largest value of 2^m .

Figure 4 (b) shows the precision and the recall as a function of the number of rows of the system as a whole. In this case if an instance is rejected by all AMRs, it counts as a false negative for the memory system, and lowers the total recall. In case the instance is accepted by only one register, but of the wrong class, the instance is a false positive for such register and a false negative for the register holding the representation of its right class, and counts both as a false positive and a false negative for the system as a whole. Finally, if more than one AMRs accept the instance, the AMRs with the lowest entropy is selected; if it is of the wrong class it counts as a false positive and a false negative for the system as a whole, as in the previous case. Figure 4 (b) shows that the precision and recall show a very similar pattern, as both grow from a very low to a higher value according to the increase of the number of rows. However, when the grid is too fine, then true instances are reject and recall starts to decrease as before. Figures 4 (a) and (b) are coherent, and both show that there is a good compromise at 64 and 128 rows. Figure 4 (c) shows the average number of accepting AMRs for each instance per AMR size. As can be seen this number goes from 47 for AMRs with one row to 1 for AMRs with 32 or more rows. This effect is further illustrated in Figure 4 (d).

The overall purpose of experiment 1 was to investigate the performance of AMRs with satisfactory operational characteristics in relation to its entropy or information content. These are the AMRs with sizes 64×64 and 64×128 . Figures 4 (e) and 4 (g) show the respective average performance of these AMRs when they are filled up with varying proportions of the *RemCorpus* –1%, 2%, 4%, 8%, 16%, 32%, 64% and 100%– using the best AMRs size; and Figures 4 (f) and 4 (h) show the performance of the system as whole for the corresponding register sizes, as before.

These latter figures show another aspect of the entropy trade-off. If the number of remembered instances is very low, the entropy is also very low –in the limiting case, if there is only one instance, the entropy is zero. In this condition there are very few objects stored in the AMRs, and if the cue to the memory recognition operation is accepted, precision is very high; however, even cues that are very similar to the stored objects will be rejected; hence recall is very low. Precision decreases slightly with the increase of remembered information and the consequent entropy increase, but recall increases quite rapidly, until a satisfactory compromise between precision and recall is reached. However, if the entropy is increased even further, then the precision starts to decrease, but the recall continues to increase. In this case the AMRs are saturated, most instances are accepted and recall is very high, but the retrieved instances are poor and may be ambiguous.

Figure 4 (e) shows that the best trade-off between precision and recall for AMRs of size 64×64 occurs when the percentage of the remembered corpus included in the memory is 64% where the corresponding graphs intersect. The graphs varies slightly for AMRs of size 64×128 where they do not intersect, and the best performance occurs when the totality of the remembered corpus is used, as shown in Figure 4 (g). The performance of the system as a whole, shown respectively in Figures 4 (f) and (h), shows that the precision remains high when the totality of the remembered corpus is used, but the recall is lower and the graphs do not intersect. To asses the performance of the system it also has to be considered the cost of the memory resource which is doubled for the largest register.

5.2 Experiment 2

This experiment shows that an AMR can hold the distributed representation of objects of different classes, adding evidence on previous work. In this case, the EMNIST-36 Alphabet is used. The procedure is analogous to experiment 1. The results are shown in the corresponding graphs in Figure 5. The experiment shows that the performance of the two settings are analogous, with the only difference that the entropy of the AMRs holding capital and lower case letters is slightly larger than the entropy of the corresponding AMR holding only one, and the average entropy of all AMRs, and for the system as a whole, is larger in this latter setting.

Experiments 1 and 2 show that the performance of the systems is mostly similar for AMRs of sizes 64×64 and 64×128 ; they also show that the system's performance is similar for the alphabets EMNIST-47 and EMNIST-36; as the latest abstracts over capital letters and uses only 36 AMRs. We use AMRs of size 64×64 with the EMNIST-36 for investigating further the quality of the memory system in experiments 3 and 4.

5.3 Experiment 3

This experiment assesses the performance of the memory retrieval operation, using the configuration and entropy levels in Figures 5 (e) and (f). The memory retrieval operation is constructive in opposition to photography memories, and all recovered

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

Figure 4. Results of Experiment 1 using EMNIST-47

Figure 5. Results of Experiment 2 using EMNIST-36

objects are constructed randomly by the β operation using a triangular distribution. We study three conditions for memory retrieval, which are illustrated in the tables of Figures 6, 7 and 8. In the tables there is a column for every type of symbol of the EMNIST-36 alphabet. The different rows show the symbol that is recovered from the memory using the same cue at all different amount of corpus and levels of entropy. White cells indicate that the memory cue was rejected at the corresponding entropy level. The top row shows the cue to the memory retrieval operation. The second row shows the symbol that is produced by the autoencoder feeding the output of the encoder directly into the decoder, i.e., bypassing the memory. As the decoder computes the inverse function that is computed by the encoder, the symbols in this row should be exact copies of corresponding cues in the top row. However, the decoder computes only an approximation and the symbols are slightly different. The encoder and decoder produced always an object, which is the most proximate to the cue, as neural networks never reject an object, despite that the quality of the cue may be very poor; and that the rendered object may be of the wrong class. The 3rd to the 10th row show the symbols recovered at the eight different levels of entropy of the *RemCorpus*. We study three conditions of the memory retrieval operation, as follows:

Figure 6. EMNIST-36 Symbols recovered with high quality cues as a function of the entropy

Figure 7. EMNIST-36 Symbols recovered with moderate quality cues as a function of the entropy

Figure 8. EMNIST-36 Symbols recovered with very poor cues as a function of the entropy

1. When the quality of the cue to the memory retrieval operation is very high and symbols are recovered at very low levels of entropy. In this condition it is expected that the symbol will be recovered at higher levels too, which it is in fact the case as shown in Figure 6. The instances chosen were recovered at a minimal entropy level, and appear in all subsequent levels down below. The table shows that the symbols at most entropy levels are quite similar to the cue, although the quality of the constructed object diminishes slightly at the higher entropy levels.
2. When the quality of the cue is not very good, as is common in calligraphy, and the symbols are not recovered at very low entropy levels, but could be retrieved at higher levels. In this condition most cues are rejected at low entropy levels,

but they are recovered at higher levels, when the recall is increased, although to the expense of the precision, and the recovered objects may be of the wrong class. This condition is illustrated in Figure 7 using randomly selected instances; in which most low entropy rows are blanks, indicating that the cue was rejected. The images the figure were selected randomly.

3. When the quality of the manuscript symbols were too poor, and could not be interpreted, even by people. In this case, the memory cues were rejected at all entropy levels. This condition is illustrated in Figure 8

5.4 Experiment 4

In this experiment, we investigated the memory retrieval operation using severely occluded cues. In this condition, cues should be rejected at most entropy levels, as they are very different of the corresponding stored objects. This is reflected in the recognition recall, which is very low. However, there may be situations in which recovering tentative objects may be useful if they can be further processed in relation to contextual information, and the right object may still be recovered, as is common in visual interpretation. Despite severe occlusions, the visible part of the cue has some amount of structure that is reflected in its abstract representation, and recognition failure may be due to the rejection of a small number of features, as shown in our previous work⁴. The definition of the η recognition operation states that in the present setting, all 64 features of the abstract representation of the cue must be included in the AMR of the corresponding class, which is a very strong test. A means to increase the recognition recall is to relax the test and allow that recognition is successful if a small number of features are allowed to fail. In particular, we tested the increase of recall with the relaxation of 1, 2 and 3 features.

We tested the memory retrieval operation when the bottom-half of the manuscript symbols were occluded, and also when the symbols were occluded by horizontal bars covering more than half of the total area in which the symbol is placed. The results are shown in Figures 9 and 10, respectively. Figures (a) at the top in both conditions show the objects recovered with no tolerance at all levels of entropy, and Figures (b), (c) and (d) show the relaxations of 1, 2 and 3 features, respectively. As expected, recall is increased but precision is lowered according to the amount of relaxation. As recall is very low in all these conditions only a small amount of symbols in the test data are recovered.

An instance of each type of symbol recovered at the lowest possible entropy level was selected, and the same instance was picked at larger entropy levels, if the symbol was indeed recovered. The none-relaxation condition for the two occluding settings is shown in figures 9 (a) and 10 (a), respectively. In this condition most cells are blanks at low entropy levels and there are columns with only blank cells. The relaxation of one feature, illustrated in (b) in figures 9 and 10, shows that objects are recovered at lower levels of entropy, and that the blanks are filled-up from bottom to top; this tendency is continued with the relaxation of two and three features as shown in (c) and (d) in both figures, respectively.

6 Experimental Setting

The programming for all the experiments was carried out in Python 3.8 on the Anaconda distribution. The neural networks were implemented with TensorFlow GPU 2.4.1, and most of the graphs were produced using Matplotlib. The experiments were run on an Alienware Aurora R5 with an Intel Core i7-6700 Processor, 64 GBytes of RAM and an NVIDIA GeForce GTX 1080 graphics card.

7 Discussion

The present memory system is associative because the AMRs are accessed through their contents, which are codified as discrete functions represented in a tabular format. The concrete representations of manuscript symbols are placed on modality specific input and output buffers, and these are translated to their abstract representations, which are input and output to the AMRs through an encoder and a decoder, respectively. The abstract representations are placed on a bus that is accessed directly by the AMRs and the autoencoder; and oppose to RAM memories of standard digital computers, that are accessed through their addresses.

The representation is distributed because the individual instances of the stored objects are represented as functions that are overlapped within the corresponding AMR. Hence the cells of the register's table can contributed to the representation of more than one object –all the objects whose representation share the same value for the same argument– and the representation of an object –a function– shares memory cells with the representations of other objects, and the relation between memory units and their contents is *many-to-many* in opposition of local representations in which such relation is *one-to-one*²⁶. This format differs from the form of distributed representations in Hopfield's associative memories, where the memory is a numerical matrix in which the stored patterns are codified implicitly.

Memory register is conceptualized as an abstraction between the representation of the object to be stored and the content of the memory, and is implemented with the standard inclusive logical disjunction between the arguments and values of the object

Figure 9. Memory retrieval with occlusions of 50% at the bottom with relaxations of 1, 2 and 3 features out of 64

to be stored and its corresponding arguments and values in the memory, which can be construed as the micro-features of the representations.

Memory recognition is conceptualized as the inclusion relation of the object to be recognized in relation to the content of the memory, and is implemented through the logical material implication between the corresponding micro-features of the object to be recognized and the content of the memory. Failure in the recognition test implements a strong negation directly, without search.

Memory retrieval is conceptualized as a constructive operation that renders a novel object always. The retrieval operation is conditioned by the recognition test, and the rendered object is built by selecting randomly –with a distribution centered on the memory cue– the values associated to the all arguments of the function representing the retrieved object.

The objects of computing in the three memory operations are the micro-features constituting the representations. The computations can be performed in parallel in a natural way taking very few computing steps, if the appropriate hardware is made available.

The memory operations conform to the main properties of human memory that emerged from the paradigmatic studies carried on by Bartlett¹; and differ fundamentally from the corresponding operations in memories developed within Hopfield's paradigm. In this latter framework, the memory is implemented through matrix operations involving the cue as a matrix of weights. In particular, memory retrieval and recognition are not distinguished, and involve an iterative process that is not guaranteed to converge in most models. These systems always return a previously stored pattern using an arbitrary measure of similarity between the memory cue and the recovered object, and fail only if there is a limit on the time allowed for the operation; for this, rejecting a memory cue, if implemented at all, is a form of negation as failure, and may render the wrong answer due to the halting problem. In addition, while EAM generalizes on the basis of the information explicitly stored, which is a property of distributed representations in general²⁶, Hopfield's model lacks such property, and is similar to local representations in this respect.

The logical nature of the memory operations, including the random element, highlights the declarative aspect of the memory. Although the representations are confused within the AMRs, they stand in a *one-to-one* relation with the represented object, and AMRs hold the representations of classes of objects, that are built through the abstraction of the individual instances that were used in their construction. Representations of individual objects lose their identity within the representation of the class, but their identity is reestablished when the objects are recovered and can be individual objects of attention, thought and action.

AMRs hold relations that have a certain amount of indeterminacy. Relational-Indeterminate Computing and the EAM made explicit such property and measure it with the computational entropy. A function is a fully determined relation and its entropy is zero. The entropy is related to the number of objects stored in the memory; when there are few, the entropy is very low, and successful retrieval operations are very precise but recall is very low; conversely, when the number of stored objects is high, so is the entropy, and recall may be very high, but precision decreases significantly. The entropy depends on the interactions between such functions, i.e., the number of arguments that share the same value for a number of functions: the greater the interactions the lower of the entropy. More generally, the operational capacity of AMRs conforms to an entropy trade-off, to the effect that the objects recovered through the memory retrieval operation in relation to the memory cue are “photographic reproductions”, “appropriate reconstructions”, “imaged objects” and noise, according to the increase of the entropy, from very low to very high, respectively. The trade-off between precision and recall is satisfactory for moderate entropy values.

The purpose of experiments 1 and 2 was to identify the size of operational AMRs for the application domain. Our previous work with digits showed that AMRs with 64 features – the cardinality of the domain of the functions stored– offer a good compromise between performance and memory size and, consequently, cost⁴. Further experiments showed that this choice is appropriate for the present case study too. Figures 1 and 2 show the performance of the memory for different sizes or discrete levels of the functions' codomains. This choice is also essential for the system performance and its cost. The tests were performed with ten values of the parameter m where the number of rows of the AMRs is 2^m . The graphs from (a) to (d) in experiments 1 and 2 show that there is a good compromise between precision and recall at $m = 6$ and $m = 7$ with entropy from 4.6 to 5.4 in both experiments.

Then we measured the performance of the system when the operational registers have different amount of information and hence entropy levels using the test corpus. The graphs (e) and (f) show the performance of the AMRs with size 64×64 and the graphs (g) and (h) show the corresponding figures for AMRs of size 64×128 . As can be seen, a satisfactory performance is achieved with a significant amount of memorized information, and both sizes of memory would benefit for a larger amount of stored information.

In our previous work we showed that AMRs can hold the representation of more than one class, with the only penalty of an increase of the entropy. In the present study we capitalized such property to represent capital and lower case letters in the same AMRs, reducing the number of registers from 47 to 36. As can be seen in the figures, the reduction in the number of classes compensated the entropy increase of the abstracted classes, and the performance of the system using the EMNIST-47 and the EMNIST-36 is practically the same. Also, the precision and recall are very similar when the AMRs are filled up with a

substantial amount of corpus, but considering that the memory cost is twice as much for registers of larger size i.e., 64×128 , we selected the EMNIST-36 corpus and alphabet, with AMRs of size 64×64 for the memory retrieval experiments.

Experiment 3 confirmed our previous memory retrieval experiments using high, middle and poor quality memory cues, as shown in Figures 6, 7 and 8, respectively. The quality of the cue impacts on the recovery: high quality cues retrieve objects with good quality at most entropy levels, but when their quality is lowered the memory retrieval operation renders blanks at low entropy levels; and when the quality of the cue is very low the memory renders blanks at most entropy levels. This corresponds to the interpretation of calligraphic symbols by people, that cannot be understood when the hand writing is very poor.

However, when the output of the encoder is fed into the decoder directly, the cue is rendered always, independently of its quality. If the cue is poor or very poor, the shape rendered by the decoder is due to the prototype patterns codified implicitly in the networks that implement the analysis and synthesis modules, rather than to the form of the cue itself.

Experiment 4 addresses recovering objects with incomplete information. In this case, with severely occluded objects. This task has been an important motivation for the study of associative memories. In Hopfield's paradigm all objects to be remembered are input explicitly to the memory beforehand. If the cue is complete, the task is reduced to match it with an object in the memory, and memory retrieval is reduced to memory recognition. However, in most realistic situations, where objects are seen from different orientations and distances, and there is usually noise, such as occlusions, poor lighting conditions, or impaired vision, the cue is incomplete, and it has to be approximated to the object recovered in the memory. If the cue is too poor the retrieved object may be the wrong one, and the agent would have no means to realize such a failure, unless the object is subjected to further contextual interpretation. Hopfield's memory models proceed in such fashion and select a complete object using incomplete information, but it is not the case that construct novel objects.

This is a salient contrast of EAM in which there is no such search process or approximation, as all objects retrieved from the memory are novel, and built randomly on the basis of the cue and all previously stored objects. We showed in our previous work that incomplete information of the cue is reflected in the values of a small number of features of the function representing the object, which are the cause of the rejection by the recognition test. Hence, our strategy to recover objects with incomplete cues consists simply of relaxing the recognition test. Figures 9 and 10 show the performance of the retrieval operation with very severely occluded objects in two settings –covering the bottom-half of the figure and placing the figure behind wide bars. These conditions are very hard to interpret even for people, and constitute a very strong test for memory systems in general.

The results show another aspect of the entropy trade-off. Most objects severely occluded are likely to be rejected at low entropy levels, as shown in the tables (a) in Figures 9 and 10. However, a small relaxation of 1, 2 and 3 features increases the recall proportionally, and objects may be recovered at lower levels of entropy, as shown from the tables in (b), (c) and (d), reducing the gap of blanks at low entropy levels. As before, the price to pay for the increase of the recall is a decrease of the precision, and the objects recovered may be of the wrong class. Nevertheless, having some candidate interpretations that can be processed further in relation to a context is better than having none.

The present research introduces a novel application to autoencoders developed within the deep-neural networks paradigm. Such systems were introduced to reduce the number of dimensions of large features spaces for the construction of classifiers and predictors, but not to produce abstract representations of concrete objects and vice versa, as is performed by the analysis and synthesis modules of the EAM architecture, respectively. The construction of this architecture required to make such modules independent, and make available the output of the encoder and the input to the decoder, which are the objects of the memory operations. Hence, the present study provides a novel application to standard autoencoders, and contributes to such technology by making such systems modular.

Finally, the present experiments show that the entropic associative memory has a satisfactory performance for storing, recognizing and retrieving manuscript letters and digits; and can potentially be used in practical applications. The present theory and experiments also show a novel conception of memory that differs in several respects from Hopfield's paradigm. In particular, it uses a more transparent notion of distributed representation; makes an explicit use of the entropy; and shows that memory conforms to the entropy trade-off. The memory operations use the logical disjunction, the material implication, and the strong negation directly, which operate on the micro-features of representations, and can be computed in parallel in a very reduced number of steps if the appropriate hardware is provided. This suggests the conjecture that such logical functions have their roots on the basic operations of associative memory. The random element suggests that imagination, creativity and free will have their roots, at least in part, in associative memory too. The memory is constructive as opposed to reproductive, and resembles better the properties of natural memory within constructivist approaches to knowledge and learning.

8 Acknowledgments

Luis A. Pineda acknowledges the partial support of grant PAPIIT-UNAM IN112819, Mexico.

References

1. Bartlett, F. C. *Remembering: A study in experimental and social psychology* (Cambridge University Press, 1932).
2. Anderson, J. R. & Bower, G. H. *Human Associative Memory: A Brief Edition* (Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey, 1980).
3. Kosslyn, S. M., Thomson, W. L. & Ganis, G. *The Case for Mental Imagery* (Oxford University Press, 2006).
4. Pineda, L. A., Fuentes, G. & Morales, R. An entropic associative memory. *Sci. Reports* **11**, 6948 (2021).
5. Steinbuch, K. Die lernmatrix. *Kybernetik* **1**, 36–45 (1961).
6. Willshaw, D. J., Buneman, O. P. & Longuet-Higgins, H. C. Non-holographic associative memory. *Nature* **222**, 960–962 (1969).
7. Kohonen, T. Correlation matrix memories. *Comput. IEEE Trans. on* **C-21**, 353 – 359, DOI: <https://doi.org/10.1109/TC.1972.5008975> (1972).
8. Palm, G. On associative memory. *Biol. Cybern.* **36**, 19–36, DOI: [10.1007/BF00337019](https://doi.org/10.1007/BF00337019) (1980).
9. Aleksander, I., Thomas, W. & Bowden, P. Wisard, a radical new step forward in image recognition. *Sens. Rev* **4**, 120–124 (1984).
10. Aleksander, I. *An introduction to neural computing* (Chapman and Hall, London, 1990).
11. Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proc. National Acad. Sci. USA* **79**, 2554–2558 (1982).
12. Kosko, B. Bidirectional associative memories. *IEEE Trans. on Syst. man, Cybern.* **18**, 49–60 (1988).
13. Ritter, G. X., Sussner, P. & DiazdeLeon, J. L. Morphological associative memories. *IEEE Trans. on Neural Networks* **9**, 281–293 (1998).
14. Ritter, G. X., DiazdeLeon, J. L. & Sussner, P. Morphological bidirectional associative memories. *Neural Networks* **12**, 851–867 (1999).
15. Sussner, P. & Valle, M. E. Implicative fuzzy associative memories. *IEEE Trans. on Fuzzy Syst.* **14**, 793–807 (2006).
16. Sussner, P. & Schuster, T. Interval-valued fuzzy morphological associative memories: Some theoretical aspects and applications. *Inf. Sci.* **438**, 127–144, DOI: doi.org/10.1016/j.ins.2018.01.042 (2018).
17. Ramsauer, H. *et al.* Hopfield networks is all you need. *arXiv preprint* (2020). [2008.02217](https://arxiv.org/abs/2008.02217).
18. Pineda, L. A. Entropy, computing and rationality. *arXiv preprint* (2020). [2009.10224](https://arxiv.org/abs/2009.10224).
19. Pineda, L. A. The mode of computing. *CoRR* **abs/1903.10559** (2019). [1903.10559](https://arxiv.org/abs/1903.10559).
20. Pineda, L. A. *Racionalidad Computacional* (Academia Mexicana de Computación, A. C., Ciudad de México, 2021).
21. Hinton, G. E. & Salakhutdinov, R. R. Reducing the Dimensionality of Data with Neural Networks. *Science* **313**, 504–507, DOI: [10.1126/science.1125249](https://doi.org/10.1126/science.1125249) (2006).
22. Masci, J., Meier, U., Cireşan, D. & Schmidhuber, J. Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction. In Honkela, T., Duch, W., Girolami, M. & Kaski, S. (eds.) *Artificial Neural Networks and Machine Learning – ICANN 2011*, vol. 6791 of *Lecture Notes in Computer Science*, 52–59, DOI: [10.1007/978-3-642-21735-7_7](https://doi.org/10.1007/978-3-642-21735-7_7) (Springer, 2011).
23. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444, DOI: [10.1038/nature14539](https://doi.org/10.1038/nature14539) (2015).
24. Kweon, M. MNIST Competition Tensorflow KR Group (2017).
25. Cohen, G., Afshar, S., Tapson, J. & van Schaik, A. EMNIST: an extension of MNIST to handwritten letters. *arXiv preprint arXiv:1702.05373* (2017).
26. Hinton, G. E., McClelland, J. L. & Rumelhart, D. E. Distributed representations (chapter 3). In Rumelhart, D. E. & McClelland, J. L. (eds.) *Parallel Distributed Processing, Explorations in the Microstructure of Cognition, Vol.1: Foundations* (The MIT Press, Cambridge, Mass., 1986).