Multiple-Modality Associative Memory: a framework for Learning

Rodrigo Simas, Luis Sa-Couto, and Andreas Wichert

Instituto Superior Técnico, University of Lisbon

Abstract. Drawing from memory the face of a friend you have not seen in years is a difficult task. However, if you happen to cross paths, you would easily recognize each other. The biological memory is equipped with an impressive compression algorithm that can store the essential, and then infer the details to match perception. Willshaw's model of Associative memory is a likely candidate for a computational model of this brain function, but its application on real-world data is hindered by the so-called *Sparse Coding Problem*. Due to a recently proposed sparse encoding prescription [31], which maps visual patterns into binary feature maps, we were able to analyze the behavior of the Willshaw Network (WN) on real-world data and gain key insights into the strengths of the model. To further enhance the capabilities of the WN, we propose the Multiple-Modality architecture. In this new setting, the memory stores several modalities (e.g., visual, or textual) simultaneously. After training, the model can be used to infer missing modalities when just a subset is perceived, thus serving as a flexible framework for learning tasks. We evaluated the model on the MNIST dataset. By storing both the images and labels as modalities, we were able to successfully perform pattern completion, classification, and generation with a single model.

Keywords: Associate Memory; Willshaw Network; Generation; Classification

1 Introduction

Deep Artificial Neural networks have made remarkable progress in recent years, being able to solve extremely complex tasks at a human, or even "superhuman" level [15]. Despite their original inspiration in biological systems [27], these models no longer attempt to implement a general information processing system like that of the brain. Instead, these models serve as effective engineering tools that solve specific tasks with high accuracy. On the flip side, the field of Computational Neuroscience concerns itself with building more general models that are constrained by biological principles, not focused on specific tasks. Examples of these constraints include Hebb's postulate for the synaptic plasticity of cell assemblies [9], and the neural energy efficiency of sparse representations [20, 21].

The basis for this work is the Willshaw model [34] of associative memory. This shallow artificial neural network is a likely candidate for a computational model of brain functions [9, 24]: With extreme neural energy efficiency [13, 16, 17], this associate memory is able to efficiently store a tremendous amount of information [18, 33, 23, 1]. Unfortunately, the usage of this model in practical settings is hindered by the so-called *Sparse Coding Problem* [12]. In this paper, we propose the Multiple-Modality framework, a new architecture for the Willshaw Model which leverages the information-completion capabilities of the model. In this new setting, the model can be used to solve a variety of learning tasks simultaneously. Furthermore, our solution achieves the reported results with a very efficient Hebbian Training rule, which requires a single pass through the dataset to train the model for all the tasks.

Although this work is evaluated on the MNIST dataset for demonstration purposes, its performance on individual tasks should not be compared with state-of-the-art image processing systems. The goal of the Multiple-Modality architecture is not to solve a particular task, but instead to provide a flexible framework such as the brain-like systems proposed by Hawkins [8]. All in all, the proposed architecture is not meant to be an accurate replica of the brain, but a simple framework that exhibits the flexibility to perform several complex tasks simultaneously, such as pattern completion, classification, and generation, under biological constraints.

The rest of this paper is organized as follows. The remainder of Section 1 is dedicated to providing the necessary background on artificial associative memories and the Willshaw Model. On Section 2, we highlight the main advantages of the model and illustrate them with experiments on the MNIST dataset. Section 3 presents the new Multiple-Modality framework, and Section 4 analyses the performance of the new architecture as a classifier and generative model. Finally, on Section 5 we discuss the results and reflect on future possibilities enabled by this work.

1.1 Associative Memory

Associative Memories (AMs) are a family of Biologically-Inspired Artificial Intelligence models that imitate the mechanisms of biological memories [4]. These

models learn by storing associations between pairs of patterns: Correlated features form synaptic connections between the memory's neurons. This way, a pattern is represented in the memory by the activation of a population of neurons. Each individual neuron can participate in many populations, thus making these models extremely efficient [25]. A trained memory can then be queried with a pattern: the neurons in the network will fire according to their learned connections, and the resulting population of active neurons will correspond to the memory's response to the query. This task is known as *Retrieval*, and the fact that both the query and the answer are patterns, means that the memory is employing Content-Addressability.

1.2 Willshaw network

The Willshaw Network (WN) [34] is a shallow, feed-forward artificial neural network that stores a set of associations between question vectors x, and answer vectors y. The model implements a content-addressable AM that establishes a mapping ($\mathrm{x} \rightarrow \mathrm{y}$).

The model is simply composed of n neurons, each having m binary connections to the input. Therefore, the model is fully represented by a binary matrix $W_{ij} \in \{0,1\} : i = 1,\ldots,m; j = 1,\ldots,n$, where the dimensions m and n are given by the fixed sizes of the *question* and *answer* vectors, respectively. This matrix defines the absence/presence of correlations between positions of the stored associations: A connection $W_{ij} = 1$ is formed during training when a correlation between the positions i of a *question* vector, and the position j of an *answer* vector is detected. Formally, given a set of M pairs (x, y), the weight matrix W is computed as:

$$W_{ij} = \min\left(1, \sum_{\mu=1}^{M} x_i^\mu y_j^\mu\right). \tag{1}$$

Notice how the learning rule requires a single pass through the training set ($\sum_{\mu=1}^{M}$), and how a local Hebbian Rule [9] is employed ($x_i^\mu y_j^\mu$).

When a cue $\tilde{x}$ is shown as input to a trained memory, each neuron will fire according to its learned connections. Then, the memory will output its resulting state $\hat{y}$ which is denoted as the retrieved vector. The memory's response to the cue reflects the learned mapping ($\mathrm{x} \rightarrow \mathrm{y}$), and since this mapping is implicitly stored through the connections of the network, the model is naturally able to generalize for novel or noisy cues. This process, denoted retrieval, can be formulated in two steps. First, the dendritic potential s, is computed for each neuron:

$$s_j = \sum_{i=1}^{m} W_{ij}\tilde{x}_i. \tag{2}$$

Then, the state of the network $\hat{y}$ is determined by applying the Heaviside step activation function H to the potential of each neuron:

$$\hat{y}_j = H\left(s_j - \theta_j\right). \tag{3}$$

The threshold parameter θ_j will determine the sensitivity of the network. In this work we employ the *soft threshold* strategy [25, 22] where $\theta_j = \max_{1 \leq i \leq n} s_i$.

A particularly interesting use case of the WN, is when we teach the model to learn the mapping ($x \rightarrow x$), which is denoted *auto-association*. In such cases, the memory learns how to map a vector back to itself, similarly to an Autoencoder [3]. In this setting, the memory has great practical uses such as reconstructing the whole pattern when only part of it is presented to the memory (Fig. 1).

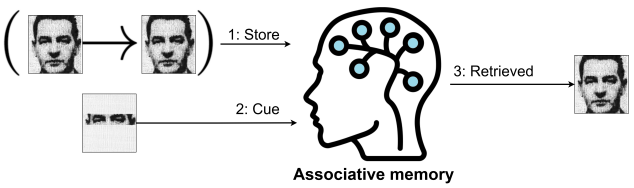


Fig. 1. Auto-association example. (1): An artificial Associative Memory model is used to store visual patterns of face in auto-association ($x \rightarrow x$). **(2):** When an incomplete version of a stored pattern is shown to the memory as a retrieval cue ($\tilde{x}$), **(3):** the memory will retrieve ($\hat{y}$) the original image.

The goal of an Associative Memory model is to store as much information as possible and retrieve it in a fault-tolerant manner. Despite the simplicity of the WN, it has been shown that this model, under specific conditions, is capable of storing a tremendous number of associations. More specifically, this is the case when the patterns that the model stores are Sparse Distributed Representations (SDRs) [2, 23, 24, 26, 10, 6, 21]. This type of representation, which closely resembles the selective neuron firing phenomenon of the human neocortex [7, 21, 19], corresponds to large binary vectors where only a subset of its bits are active (sparse), and where all of the positions of the vectors are quasi-uniformly used across a dataset (distributed). The main issue in the field of Associative Memories is the fact that real-world data is not naturally in the form of SDRs (Fig. 2).

Without the use of sparse compression prescriptions, mostly undiscovered, these models are unable to be useful in practical settings. Again, an analogy with biological systems can be drawn. The neocortex does not handle information in its raw format, instead, it relies on lower functional regions to transform sensory data into the sparse activation of neurons. For instance, the visual area of the brain is divided into several hierarchical regions: V1, V2, V4, and IT. The V1 area of the brain is responsible for detecting low-level visual features such as edges, and basic color [5]. The detection of such features results in the activation of a collection of neurons that forms a signal. This signal is passed onto the V2 area which will apply a similar process. After passing through all the regions, the final result will be an SDR that represents what is being perceived [8].

In our recent work [28, 30], a sparse encoding prescription for visual patterns, coined the What-Where (WW) encoder, was proposed. This prescription is in-

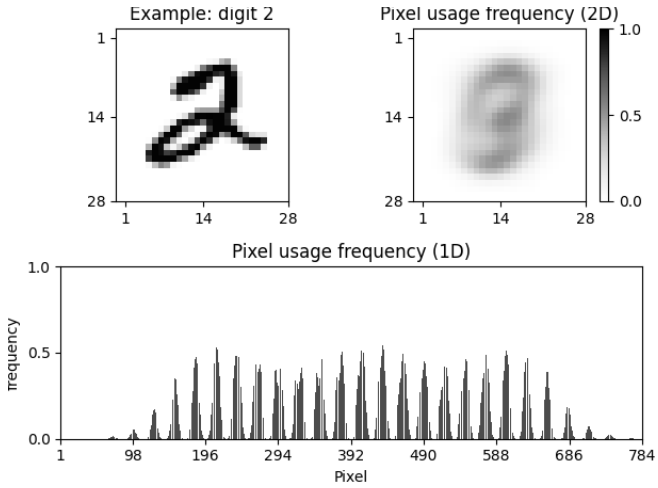


Fig. 2. The sparse coding problem. Here, the MNIST dataset [14] is used. The dataset contains 70,000 images of handwritten digits, drawn by 250 different writers. Each image is 28 pixels wide and 28 pixels high, totaling 784 pixels per image. **(top-left)** One single example from the dataset. **(top-right)** The relative frequency (encoded with color) at which each pixel is used. **(bottom)** The same pixel frequency depicted in the top-right picture, visualized in a single horizontal axis. This dataset is neither sparse, since many of the pixels are active with high probability ($f \approx 0.5$), nor distributed since pixels are not used equally. This type of dense representation will lead to a poor performance when stored on an Associative Memory.

spired by elements of the mammalian visual cortex [11] to produce informative binary compressions. Preliminary results on the MNIST dataset [14] have illustrated the quality of the WW codes in the context of Associative Memories: Storing large amounts of these SDR in the Willshaw Network allowed for error-tolerant retrieval [29], thus opening up the opportunity to finally use this model on practical settings.

2 Single Modality Willshaw Network

This section is dedicated to highlighting one of the main strengths of the Willshaw Network which is its ability to complete missing information. This feature of the basic architecture of the model is a key component of the Multiple-Modality architecture proposed in this paper, which will be presented in the upcoming section.

Our previous analysis [29] has focused on the retrieval process at the SDR level, where we compared the bits of the retrieval cues and retrieved vector to measure the memory's performance. The main conclusions of this study were:

- The memory can store large amounts of SDRs. Even when the retrieval becomes imperfect as the memory fills up, the noisy retrieved patterns keep the relevant information about the pattern.
- The memory can effectively retrieve noisy cues where active bits are randomly deleted.

Here we complete the analysis by including a decoder module in the pipeline [32] (Fig. 3). This way we can get a visual representation of the SDRs in the several steps of the storage and retrieval process, and evaluate the memory with measures such as the Mean Squared Error (MSE) between the original patterns P, and reconstructions $\hat{P}$: $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{784}\sum_{j=1}^{784}\left(P_j^{(i)} - \hat{P}_j^{(i)}\right)^2\right)$, where 784 is the number of pixels in each image, and n is the number of patterns.

2.1 Retrieval

Let us start with the simplest of cases. Here, we present the memory with cues that are identical to those that the memory stored in its learning stage. Recall that since the memory is implementing an auto-associative mapping ($\text{x} \rightarrow \text{x}$), the goal of the memory is to output a copy of the cue.

The Willshaw Network does not retrieve its stored patterns perfectly. In fact, the average number of bits in the retrieved vector increases by a factor of 4 (Fig. 4(a)) as we store the entire MNIST dataset in the memory. Despite this phenomenon, the MSE between the original images and the reconstructions only increases by a factor of 1.2 (Fig. 4(b)). This result shows that the noise introduced by the memory is non-random and that the retrieved vectors keep the information relevant to the pattern (as originally proposed in [29]). This can be confirmed by visually inspecting the examples of Fig. 5.

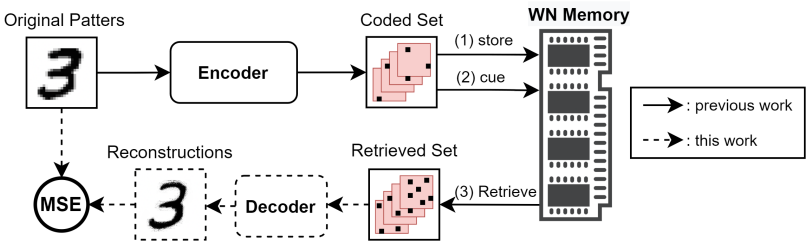
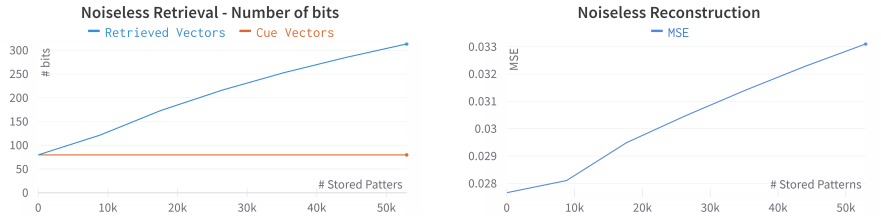


Fig. 3. Encoder Decoder and Memory pipeline. First, we encode the visual patterns of the MNIST dataset into SDRs and store them in a Willshaw Network in auto-association. We then cue the trained memory (with either the original WW codes or noisy versions of it) to obtain the retrieved codes. Finally, the WW decoder can be used to obtain a visual representation of the memory's content.



(a) Number of bits **(b)** Mean Squared Error

Fig. 4. Noiseless Retrieval. Here we analyse the retrieval process of the best performing sparse codes of [29]. We measure the nuber of active bits in the retrieval cue, and retrieved vector **(a)**; and the Mean Squared Error between the original MNIST patterns and the reconstructions of the retrieved vectors. Notice how, despite the heavy increse in the information in the retrieved vector (a), the reconstruction error does not increase significantly (b).

(a) Original Patterns **(b)** Noiseless cues

(c) Retrieved (10.000 stored) **(d)** Retrieved (50.000 stored)

Fig. 5. Noiseless retrieval. Notice how the decodings in (b) are very similar to the original patterns (a), with only a few pixels missing due to the lossiness of the encoding process. As the memory feels up (c) the memory adds noise to cues. Most of the noise is around the active pixels of the digit, but some noisy spots start to appear when the memory becomes fuller (d).

2.2 Completion

Let us now consider a more practical scenario. Consider that the retrieval cues are incomplete versions of the patterns that are stored in the memory. Formally, the cues $\tilde{X}$ are obtained from the stored patterns X by stochastically deleting each active bit of X with probability P_{del}. The goal is to test the content-addressable capability of the model. If the memory has learned a robust auto-associative mapping, it will be able to match the subset of information in the cue with the corresponding stored pattern.

To have a more detailed analysis, we subdivide the MSE into its negative and positive part: the Mean Squared Error (MSE) due to lost information (when $P_j^{(i)} > \hat{P}_j^{(i)}$), and the MSE due to extra information (when $P_j^{(i)} < \hat{P}_j^{(i)}$) (Fig. 6).

(a) MSE due to lost information **(b)** MSE due to extra information

Fig. 6. WW Decoder Reconstruction Error - Noisy (type zero). Here we plot the MSE between the original patterns and the reconstructions. When $x = 0$ we are measuring the MSE of the cue reconstructions. On the rest we measure the quality of the **reconstructions** that utilize the memory. The green lines correspond to a run without noise, the remaining plots correspond to runs where the retrieval cues are altered by adding **zeros** to the code.

Our results show how the memory can very effectively complete the missing information from the noisy cues. Notice how, the MSE due to loss information sharply drops once we used the memory (compare $x = 0$, with $x = 9.000$ of Fig. 6(a), which corresponds to the first stored batch). The MSE due to extra information increases as the memory is loaded, which is the natural tendency of the retrieval process (notice the same trend in Fig. 4(b)). Fig. 7 illustrates the information completion capabilities of the model.

3 Multiple-Modality Architecture

The WN makes no assumption about the data that it stores. Each bit of a pattern that is stored in a WN is interpreted as an informative feature. The position of a particular feature within the pattern, and its meaning/interpretation are irrelevant to the memory. As a result, we can fill the memory with heterogeneous patterns; i.e., patterns containing multiple modalities/types of information, as

(a) Noisy cue (Decoding) **(b)** Reconstruction of noisy cues.

Fig. 7. Completion - examples. The reconstruction of the noisy cues. **(b):** The reconstruction of the outputs of a memory that stores 10.000 patterns. Notice the effect of the noise ($P_{del} = 0.75$) on the cues: the digits are missing a lot of pixels (a). Using the memory's retrieval to complete the missing information leads to an improvement in the reconstructions (b). Notice also some of the spurious retrievals that occur due to uncertainty introduced by the noise (bottom-most 4 converges to a 9).

seen in Figure 8. In such cases, we are building a Multiple-Modality Willshaw Network (MMWN).

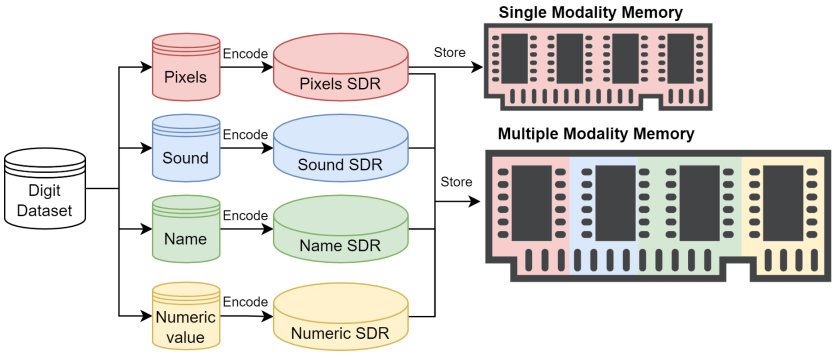


Fig. 8. Multiple vs Single modality memory. In this example, we are storing digits in both memories. If we consider a single type of information about the digit, then we require a Single Modality AM (top). If we consider multiple types of information about each pattern, we are using a Multiple Modality AM (bottom).

As we have seen before, one of the main strengths of the WN is the ability to complete missing information. A MMWN benefits immensely from this property: During training, the memory will learn how all the different features, from all modalities, are correlated. On retrieval, all the different modalities contribute to the memory's response. If the information from one of the modalities is missing in the retrieval cue, the memory can complete it using the remaining modalities.

Let us illustrate the strength of this idea with a simple example:

3.1 Example of multi-modality retrieval

Consider two patterns (x^1 and x^2), each represented in two different modalities (a and b):

- $x_a^1 = (0, 1),\ x_b^1 = (0, 0, 1, 1)$
- $x_a^2 = (1, 0),\ x_b^2 = (1, 1, 0, 0)$

Using the Willshaw training rule (Eq. (1)), we could train two distinct WNs (one for each modality) to store the two distinct modalities of the patterns in auto-association, yielding:

$$W_a = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ W_b = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Alternatively, we can concatenate the two modalities of each pattern,

- $x_{ab}^1 = (x_a^1 | x_B^1) = (0, 1 | 0, 0, 1, 1) = (0, 1, 0, 0, 1, 1)$
- $x_{ab}^2 = (x_a^2 | x_B^2) = (1, 0 | 1, 1, 0, 0) = (1, 0, 1, 1, 0, 0)$

and store the concatenated patterns in auto-association, yielding a single multi-modality matrix:

$$W_{ab} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Notice two things about the resulting matrix:

- The individual matrices from modalities a and b (in blue and red, respectively) are displayed along the diagonal of the bigger matrix. Each of these matrices represents the **intra-modality correlations**.
- The rest of the matrix (green) has non-zero entries, which represent the **inter-modality correlations**.

The single multi-modal memory has all the capabilities of several single-modal ones. But the additional **inter-modality correlations** provide the memory with extra flexibility.

For instance, if we delete one of the modalities from each pattern to form the following retrieval cues:

- $\tilde{x}_{ab}^1 = (x_a^1 | 0) = (0, 1, 0, 0, 0, 0)$
- $\tilde{x}_{ab}^2 = (0 | x_b^2) = (0, 0, 1, 1, 0, 0)$

And then compute the memory's response using the two-step retrieval rule (Eqs. (2) and (3)), we get:

- $y^1_{ab} = (0, 1, 0, 0, 1, 1) = x^1_{ab}$
- $y^2_{ab} = (1, 0, 1, 1, 0, 0) = x^2_{ab}$

The memory can fully retrieve the missing modality in both cases, which would not be possible if we used W_a and W_b separately.

Please note that the Multiple-Modality framework is just an abstraction: the model is simply storing and retrieving larger binary feature vectors using its basic rules, without any notion of modality. As a consequence, the model is not restricted to the use case of this example, where one modality is completely deleted, and the other used to retrieve it. One could, for instance, provide partial information of both modalities, and use the memory to complete both simultaneously.

Many possible use cases leverage the mechanism of completion. In fact, typical learning tasks (e.g., prediction, classification, and generation) can be seamlessly mapped to a completion problem. For this reason, the proposed model should be seen as a framework and not a tool for specific applications.

In the following section, we will show two examples of typical learning tasks which can be solved under the Multi-Modal framework.

4 Experimental analysis

To demonstrate that the MMWN can work in practice, let us use the MNIST dataset to train a MMWN with two modalities and use it for practical applications. Each pattern in the MNIST dataset has two attributes: the label represented by an integer; and a picture of the handwritten digit represented as an array of pixels. Our previous work [29] has shown that a normal WN can efficiently store and retrieve the images of the MNIST dataset. Here, we will use a MMWN that stores both the images and the labels of the MNIST dataset. If the MMWN manages to store both modalities efficiently, we can leverage the information-completion properties of the model for practical applications such as the generation of new patterns and classification.

4.1 A simple SDR for integers

To use the label of the MNIST digit (which is an integer between 0 and 9) as a modality, we must first transform it into a binary code. Furthermore, this binary code should be suitable for a WN; i.e., be an SDR.

4.1.1 Noisy X-Hot Encoder Our proposed encoding strategy is the Noisy X-Hot (NXH) encoding, which is a stochastic version of the X-Hot encoding strategy (a more general version of the well-known 1-Hot codes). The main idea is that X bits in the code will randomly activate with high probability, while the remaining bits will be active with low probability. The result is an X-Hot encoding with some added noise/randomness, hence the name.

Formally, for a label l in the range $\{0, \ldots, L-1\}$, its Noisy-X-hot encoding $e(l)$, with X bits per class, and probabilities P_{class}, and P_{rest} (with $P_{class} \gg P_{rest}$), is a binary array of size $L \times X$ given by:

$$e_i(l) = \begin{cases} 1 & \text{with } P_{class} \text{ probability, if } i \in \{z \in \mathbb{Z} \mid lX \leq z < (l+1)X\} \\ 1 & \text{with } P_{rest} \text{ probability, if } i \notin \{z \in \mathbb{Z} \mid lX \leq z < (l+1)X\} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The resulting code can be thought of as a collection of L populations of with X neurons, one for each of the L integer values we are encoding. For a given label l, its NXH code will have most of its activity on the l^{th} interval. Furthermore, two distinct patterns with identical labels ($l_1 = l_2$) will likely have different NXH encodings ($e(l_1) \neq e(l_2)$) due to the stochastic nature of the NXH. In this way, the NXH code is not only an encoding of the labels but also a **description** which allows us to differentiate between different patterns with the same label.

4.1.2 Noisy X-Hot Decoder For practical purposes, we must have a mechanism that maps the NXH codes back into the integer value of the label. We can achieve this, by looking at the total activity of all the L intervals of the NXH code and picking the interval with the most activity.
Formally, a NXH code e, can be mapped back into its corresponding label l with:

$$l = \arg\max_i \left(\sum_{n=iX}^{(i+1)X} e_n \right) \tag{5}$$

4.2 Methodology

There are two key steps when operating an AM: the learning step and the retrieval step. The same is true for a MMWN.

The learning step consists in: (1) encoding the labels and pixels into the descriptions and What-Where codes, respectively; (2) concatenating the description and code into what we refer to as a desCode (DC) (short for description and code); and (3) storing the DCs in auto-association, as depicted in Fig. 9

The Retrieval Step simply consists in showing a cue DC to the trained memory and obtaining a retrieved DC back. As shown in the example of Section 3.1, the MMWN can be used to complete missing modalities using information from another. The flexibility of this framework allows us to manipulate the retrieval cues in different ways to perform different learning tasks.

4.3 Classification

If one shows, to a trained MMWN, cues where the description modality has been deleted, the MMWN will complete the missing information, essentially working as a classifier.

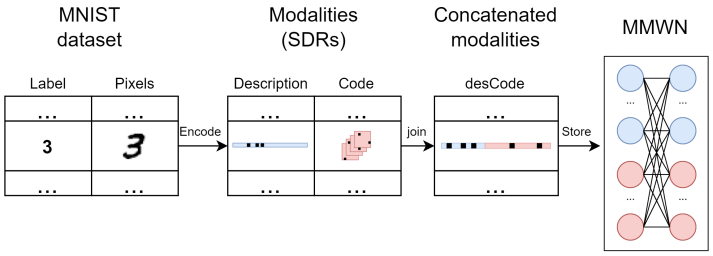


Fig. 9. Training Step in a MMWN.

To test the MMWN's performance when it comes to classification we defined three tasks of increasing complexity: Auto-association, classification of stored patterns, and classification of unseen patterns (see Fig. 10). Results are reported in Fig. 11.

4.4 Generation

Inversely to classification, generation with a MMWN is performed by creating a set of cues where the description modality remains intact but the visual modality is deleted. The memory's response will complete the missing information in the visual modality, essentially generating a pattern from a description.

Providing a trained MMWN with a cue where the visual modality has been completely set to zero yields the results in Fig. 12(b). The description modality alone is not able to generate unique patterns that resemble examples from the original dataset, such as those in Fig. 12(a). Instead, "blobs" with no detail, which looks as prototypes of each class, are obtained.

To move away from "blobs" and create more realistic generations, we cannot provide zero information in the visual modality. Instead, we must "seed" the generation process by adding some visual information to the retrieval cue, so that the memory can leverage its information completion capabilities. Furthermore, the visual information provided in the seed should be in accordance with the description that we are generating from (e.g., if we generate from class "zero", we should seed the generation process with some visual features that often occur in patterns of the zero class). The question then becomes how do we create these class-dependent generation seeds.

In an ideal scenario, with access to the probability distributions of the visual features for each class, one could sample from these distributions to create artificial cues, and retrieve them in an MMWN to obtain generations. A much more elegant and self-contained alternative is to use the blobs (Fig. 12(b)) as an approximation for the probability distribution of each class: These blobs are very rich in information, containing many more active bits than the average pattern that is stored in memory. Essentially, they are a collection of all the features that are commonly found in examples of the class that originated the blob. If one deletes most of the bits from a blob, the result is a small set of visual features

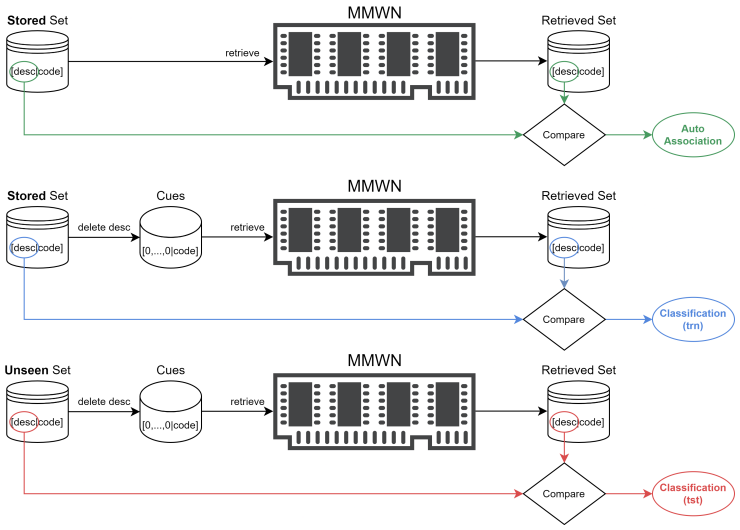


Fig. 10. MMWN Classification methodology. To evaluate the MMWN's ability to perform classification we define three distinct tasks of increasing difficulty: **(Top):** Auto-Association, **(Middle):** Classification of stored patterns, and **(Bottom):** Classification of unseen patterns. In all cases, we compare the original description of the pattern (before memory) with the description in the retrieved vector (after memory). For all three tasks, the accuracy score is determined by comparing the integer value of the label before and after the memory which is achieved by decoding the description with Eq. (5).

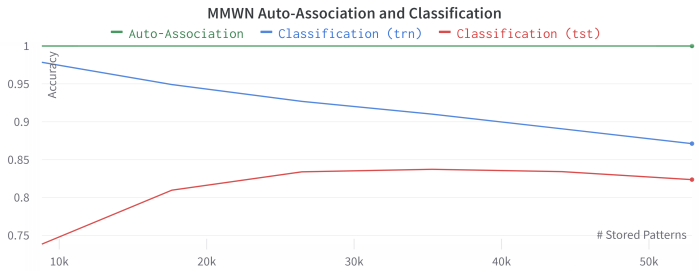


Fig. 11. Auto-Association and Classification Results with the MMWN. Here we follow the methodology of Fig. 10 to report the scores of the three tasks as the memory is filled with patterns. For the WW encoder, we used the parameters of the best performer in [29]. For the NXH encoder we used: $X = 500$, $P_{class} = 0.5$, and $P_{rest} = 0.0$. **(Green):** Auto-Association score. Here we are simply measuring the memory's ability to auto-associate the description of DCs. The accuracy is perfect even when the memory is full, indicating that the descriptions are tolerant to the noise that the memory introduces. **(Blue):** Classification of stored patterns. This score is high but decreases monotonically as the memory is filled up, which is expected since the memory gradually loses the ability to perfectly retrieve the patterns that it stores. **(Red):** The classification accuracy for unseen patterns improves as the memory stores more information because the memory naturally becomes better at generalizing as it stores more information (peeking at 84.04%). However, once the memory becomes too full, this score gets worse.

(a) MNIST examples **(b)** Blobs

Fig. 12. MNIST examples (a) and Drawings from memory (blobs) (b). The blobs are obtained when we provide a MMWN with cues that contain the description, but are missing the visual modality.

that can be used as a retrieval cue for generation. Let us denote this process of sampling from the memory's output as **sparsification**.

4.5 Iterative Generation

To use an MMWN as a generative model, an iterative approach is followed. The general idea is to iteratively retrieve and sparsify the inputs and outputs of the retrieval process, respectively [32]. In the beginning, the visual modality of the generation cue is empty, but with each iteration, we improve the generation seed and provide the memory with more information. This, way the memory will gradually get closer to a pattern similar to those that it stores. The process stops once the number of bits in the generation is in the predefined acceptance interval (see Fig. 13).

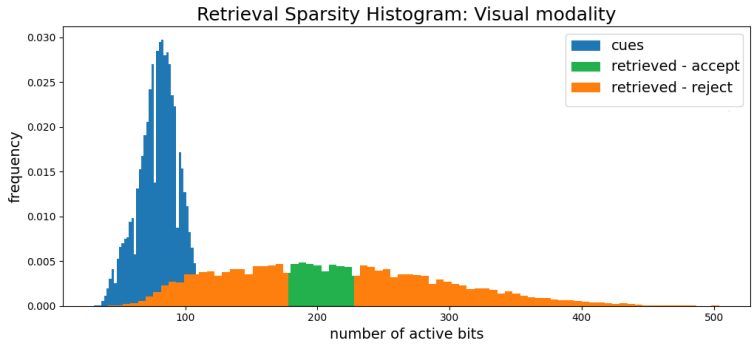
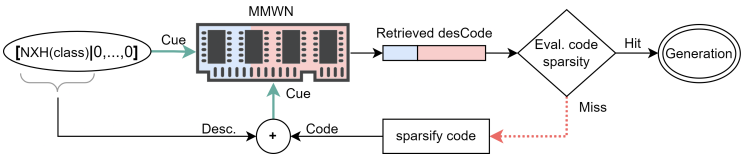
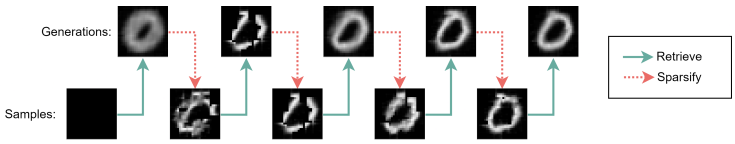


Fig. 13. Retrieval Acceptance Interval Here we used a trained memory to retrieve noisy cues. The number of bits in the cues, and retrieved vectors is measured and plotted. We can see that both the cues (blue) and retrieved vectors (orange/green) appear to follow a normal distribution. The number of bits in the retrieved vector is indicative of the quality of the retrieval process: Values close to the center of the distribution (green) are a good indicator of a successful retrieval. Values in the tails of the distribution (orange) correspond to poor retrievals, where the cue led to a lack or excess of detected correlation by the memory.

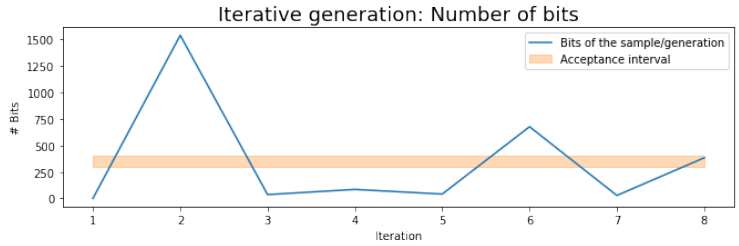
An overview of the architecture of the iterative generation method can be seen in Fig. 14(a). Additionally, an illustrative example is provided in Figs. 14(b) and 14(c). The first shows how the visual patterns evolve throughout the generation process, while the second monitors the number of active bits. The results of the iterative generation method can be seen in Fig. 15.



(a) Method overview



(b) Iterative generation Example: Images



(c) Iterative generation Example: Sparsity

Fig. 14. Iterative Generation: Three perspectives. A Multi-modal Willshaw network is used to generate patterns. A description and an empty code are given to the memory as a retrieval cue. The memory returns a generation, and its sparsity is evaluated. If the sparsity of the generation is within a predefined interval, then the generation process ends; otherwise, the generation is "sparsified" (i.e. we delete bits from it until a certain level of activity is reached). The sparsified code is then fed back to the memory, and we repeat the process. **(a):** The process begins by providing the multi-modal memory with a desCode (DC) where the description is a new noisy-X-Hot (NXH) encoding of the class we want to generate from, and the code is empty (all zeros). Then we take the code from the retrieved desCode and evaluate its sparsity. If the sparsity is a "hit", the process ends here. Otherwise, we "sparsify" the retrieved code, join it with the original description, and repeat the process. **(b):** The patterns on the bottom row are retrieval cues that act as "samples" in the generation process. The memory retrieves (green arrows) these samples and produces generations which are displayed on the top row. If the generation's sparsity is outside the acceptance interval, we randomly delete bits from the generated code (dashed red arrows) to obtain a new sample. **(c):** Here we plot the number of bits throughout the iterative generation process. This graph corresponds to the same example as Fig. 14(b). The first sample has zero active bits since it corresponds to an empty code. The sparsity of the encoding process will oscillate (blue line): When we "sparsify" we bring the number of bits down. On retrieval, the memory will complete the missing information and add more bits. When the output of the memory falls within the acceptance interval (orange) the generation process ends.

Fig. 15. Iterative generation examples. Here we follow the iterative method described in Fig. 14(a) to generate 5 examples from each MNIST class with the MMWN. All generations look like realistic examples from the class that they belong to. Furthermore, there is some variance between generations of the same class. These results were obtained with an acceptance interval of $[400, 500]$ bits.

5 Conclusion

In this paper, we propose the Multiple-Modality framework and demonstrate its applicability. The resulting system is a computational model of brain functions which learns by storing several modalities (e.g., visual, textual, etc.) of each pattern simultaneously. After storage, the model's completion capabilities can be leveraged to infer missing modalities when just a subset is perceived, thus unlocking a framework for learning tasks. To demonstrate the idea, we evaluated the model on the MNIST dataset of handwritten digits. Using sparse encoding prescriptions for the image [30] and numeric value (Section 4.1.1) of the patterns, we were able to perform several learning tasks which are crucial in intelligent systems, namely Retrieval, Completion, Classification, and Generation. Furthermore, we achieved the reported results using an extremely efficient Hebbian rule, which trains the memory in a single pass through the data.

Despite being evaluated on traditional Machine Learning tasks, the model's performance on individual tasks should not be compared with state-of-the-art deep architectures. After all, the proposed model is intended to be a general information processing framework, not designed for specific applications.

This work is a first step, where an idea is proposed and its feasibility demonstrated. Therefore, the results reported here could certainly be improved with future research efforts, e.g., by applying the iterative retrieval method (Section 4.5) to tasks other than generation. Additional research possibilities include performing other completion-based learning tasks (e.g., regression, time-series prediction, anomaly detection, etc.) under the Multiple-Modality framework; storing several modality-sharing datasets simultaneously, which is possible since the system is not domain-bound; and scaling up the model to include additional modalities, which entails building sparse encoding prescriptions as well.

While far from a replica of a biological system, the design of our solution is heavily constrained by biology. Therefore, its successful application in artificial intelligent tasks is exciting, as it might shed some light on the mechanisms that operate in biological memories.

Acknowledgments

This work was supported by national funds through Fundação para a Ciência e Tecnologia (FCT) with reference UIDB/50021/2020 and through doctoral grant SFRH/BD/144560/2019 awarded to the second author. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

References

1. Amari, S.I.: Characteristics of sparsely encoded associative memory. Neural networks **2**(6), 451–457 (1989)
2. Barlow, H.B.: Single units and sensation: a neuron doctrine for perceptual psychology? Perception **1**(4), 371–394 (1972)
3. Bengio, Y., Goodfellow, I., Courville, A.: Deep learning, vol. 1. MIT press Massachusetts, USA: (2017)
4. Daniel, K.: Thinking, fast and slow (2017)
5. Felleman, D.J., Van Essen, D.C.: Distributed hierarchical processing in the primate cerebral cortex. Cerebral cortex (New York, NY: 1991) **1**(1), 1–47 (1991)
6. Field, D.J.: What is the goal of sensory coding? Neural computation **6**(4), 559–601 (1994)
7. Hawkins, J., Ahmad, S., Purdy, S., Lavin, A.: Biological and machine intelligence (bami) (2016), initial online release 0.4
8. Hawkins, J., Blakeslee, S.: On intelligence. Macmillan (2004)
9. Hebb, D.O.: The organization of behavior: A neuropsychological theory. Psychology Press (2005)
10. Hecht-Nielsen, R.: Neurocomputer applications. In: Neural computers, pp. 445–453. Springer (1989)
11. Hubel, D.H., Wiesel, T.N.: Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. The Journal of physiology **160**(1), 106–154 (1962)
12. Knoblauch, A., Palm, G., Sommer, F.T.: Memory capacities for synaptic and structural plasticity. Neural Computation **22**(2), 289–341 (2010)
13. Laughlin, S.B., Sejnowski, T.J.: Communication in neuronal networks. Science **301**(5641), 1870–1874 (2003)
14. LeCun, Y.: The mnist database of handwritten digits. http://yann. lecun. com/exdb/mnist/ (1998)
15. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436–444 (2015)
16. Lennie, P.: The cost of cortical computation. Current biology **13**(6), 493–497 (2003)
17. Levy, W.B., Baxter, R.A.: Energy efficient neural codes. Neural computation **8**(3), 531–543 (1996)
18. Nadal, J.P., Toulouse, G.: Information storage in sparsely coded memory nets. Network: Computation in Neural Systems **1**(1), 61–74 (1990)
19. Olshausen, B.A., Field, D.J.: Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature **381**(6583), 607–609 (1996)
20. Olshausen, B.A., Field, D.J.: Sparse coding with an overcomplete basis set: A strategy employed by v1? Vision research **37**(23), 3311–3325 (1997)
21. Olshausen, B.A., Field, D.J.: Sparse coding of sensory inputs. Current opinion in neurobiology **14**(4), 481–487 (2004)
22. Palm, G., F., Sommer, F.T., Strey, A.: Neural associative memories. Associative Processing and Processors pp. 307–326 (1997)
23. Palm, G.: On associative memory. Biological cybernetics **36**(1), 19–31 (1980)
24. Palm, G.: Neural assemblies: An alternative approach to artificial intelligence, vol. 7. Springer Science & Business Media (1982)
25. Palm, G.: On the information storage capacity of local learning rules. Neural Computation **4**(5), 703–711 (1992)

26. Palm, G., Sommer, F.T.: Associative data storage and retrieval in neural networks. In: Models of neural networks III, pp. 79–118. Springer (1996)
27. Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review **65**(6), 386 (1958)
28. Sa-Couto, L., Wichert, A.: Attention inspired network: Steep learning curve in an invariant pattern recognition model. Neural Networks **114**, 38–46 (2019)
29. Sa-Couto, L., Wichert, A.: Storing object-dependent sparse codes in a willshaw associative network. Neural Computation **32**(1), 136–152 (2020)
30. Sa-Couto, L., Wichert, A.: Using brain inspired principles to unsupervisedly learn good representations for visual pattern recognition. Neurocomputing (2022)
31. Sa-Couto, L., Wichert, A.: "what-where" sparse distributed invariant representations of visual patterns. Neural Computing and Applications pp. 1–8 (2022)
32. Simas, R.: Multi-Modal Associative Memory: A framework for Artificial Intelligence. Master's thesis, Instituto Superior Técnico, University of Lisbon (2022), https://fenix.tecnico.ulisboa.pt/cursos/meic-a/dissertacao/565303595503415
33. Steinbuch, K.: Die lernmatrix. Kybernetik **1**(1), 36–45 (1961)
34. Willshaw, D.J., Buneman, O.P., Longuet-Higgins, H.C.: Non-holographic associative memory. Nature **222**(5197), 960–962 (1969)