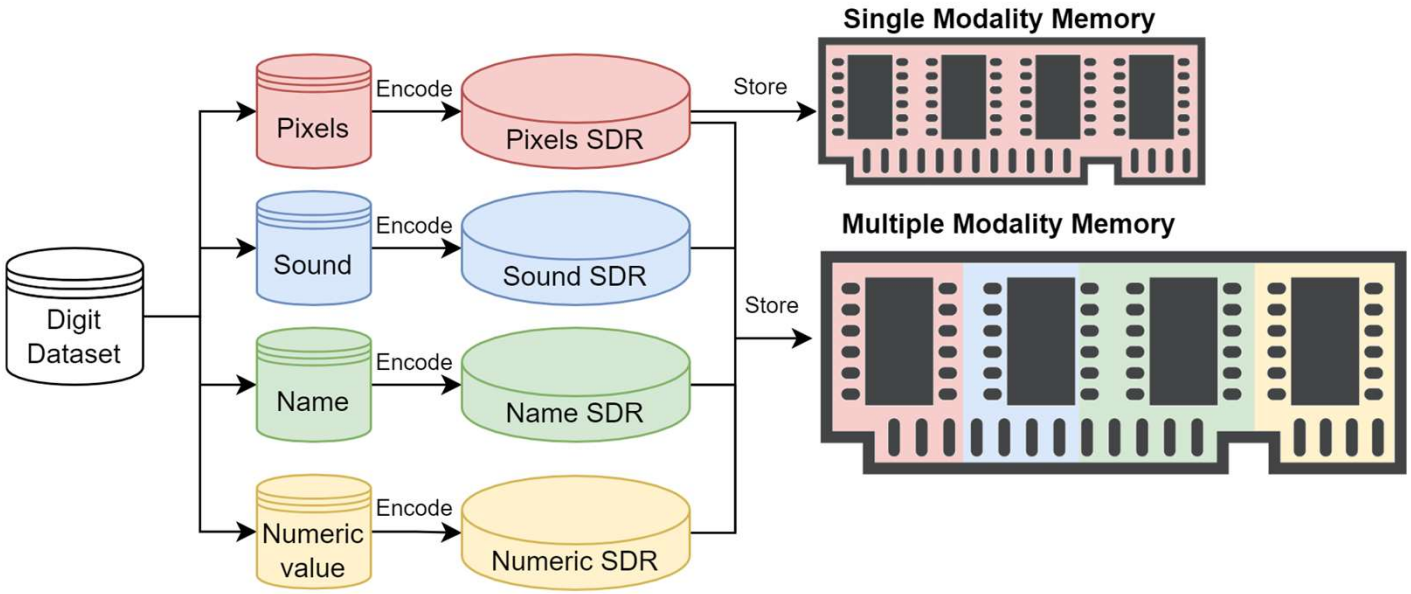
Multi-modal Associative Memory:
A framework for Artificial Intelligence

Thesis to obtain the Master of Science Degree in:
Information Systems and Computer Engineering

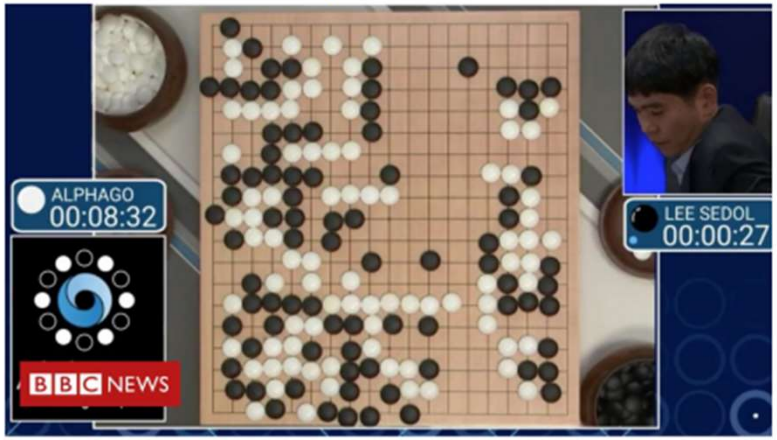


Student: Rodrigo Simas
Supervision: Prof. Andrzej Wichert, Luís Sá Couto
Arguer: Prof. João Pereira
Jury: Prof. Daniel Gonçalves (President)

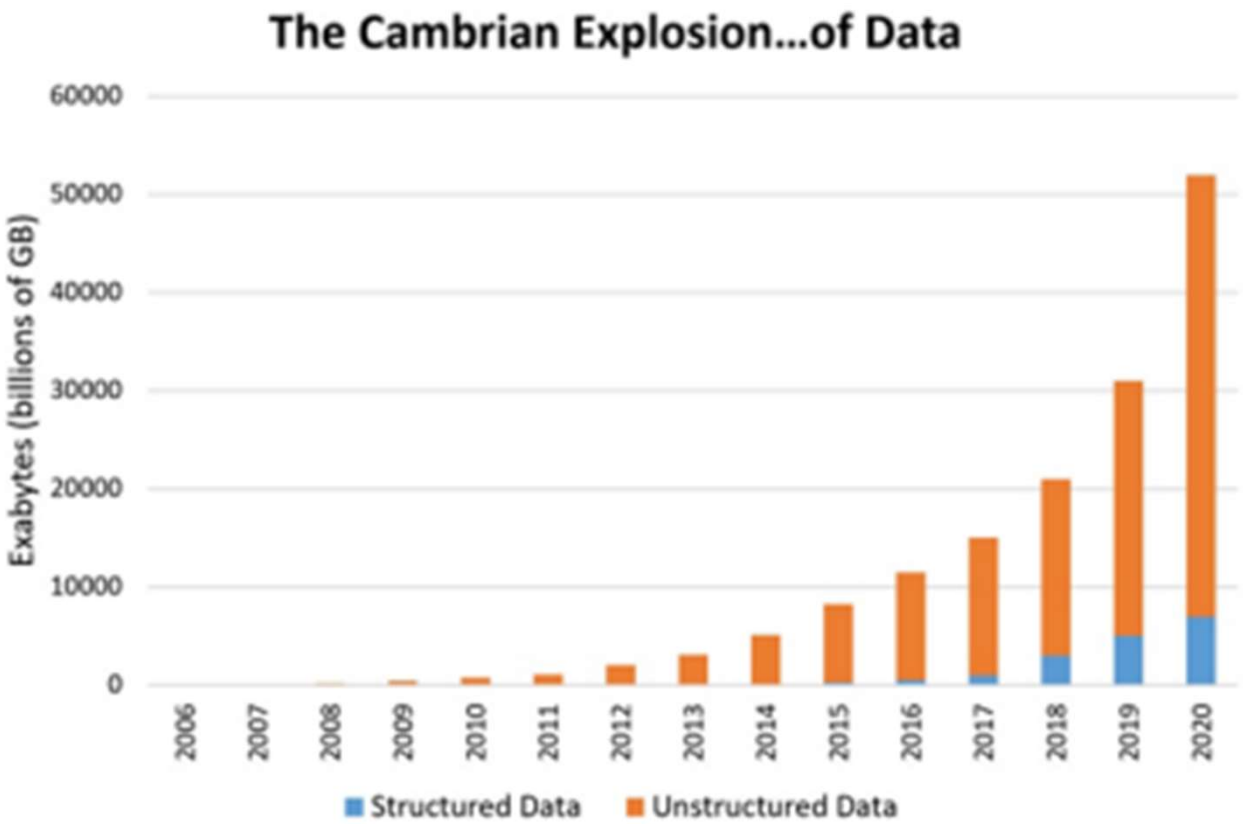
Artificial Neural Networks have made remarkable progress due to backprop!

Self-driving cars

Alpha Go



But they have some limitations

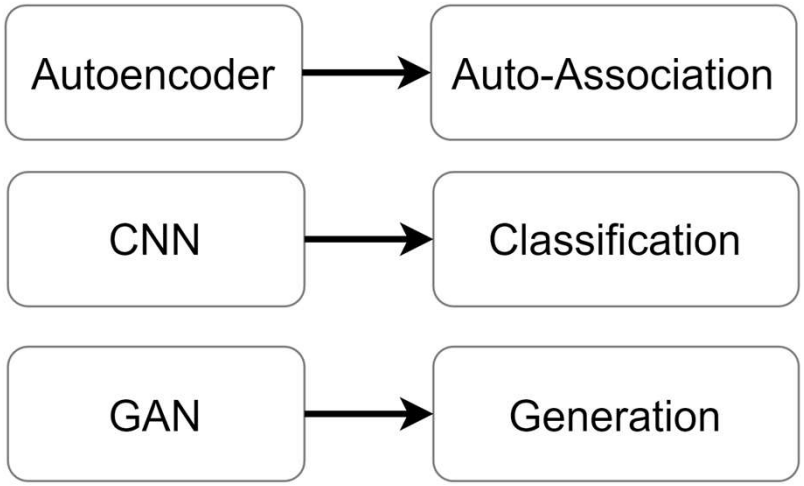


Require large datasets

But they have some limitations

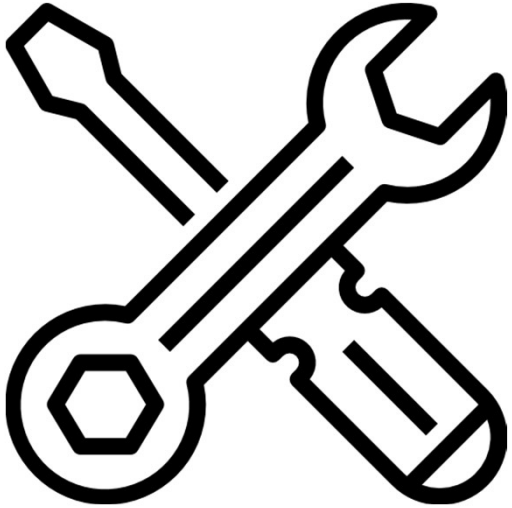
Unexplainable adversarial examples

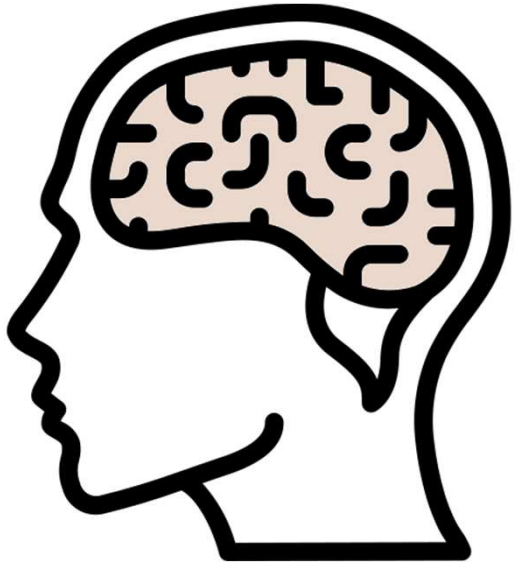
But they have some limitations



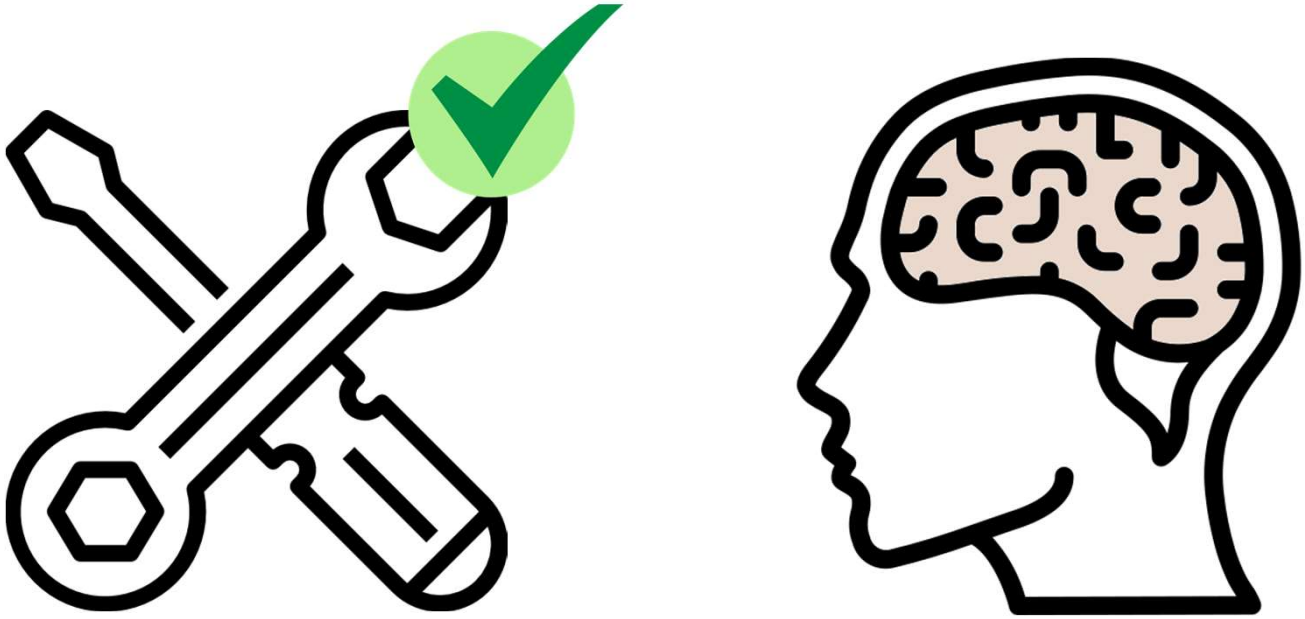
Specialization on a specific task and domain

The approach of Deep Learning

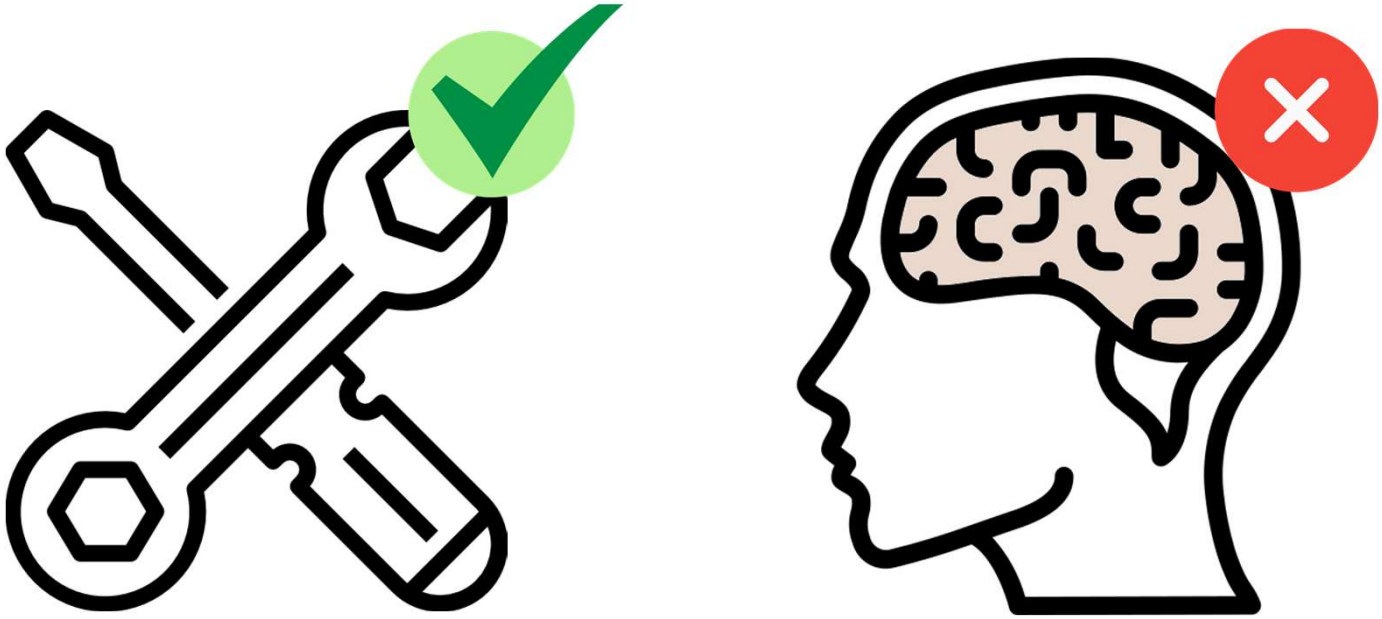




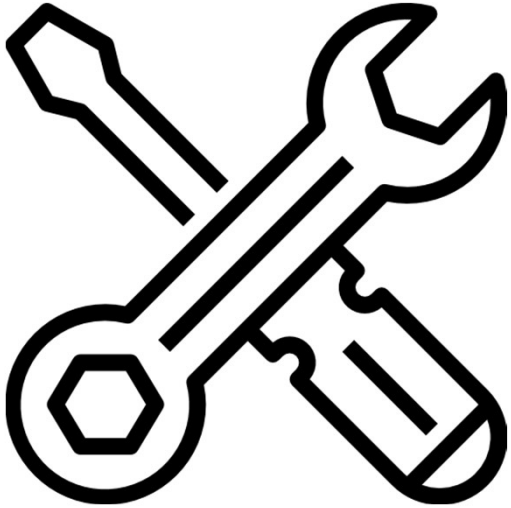
Deep Learning is a Great engineering tool

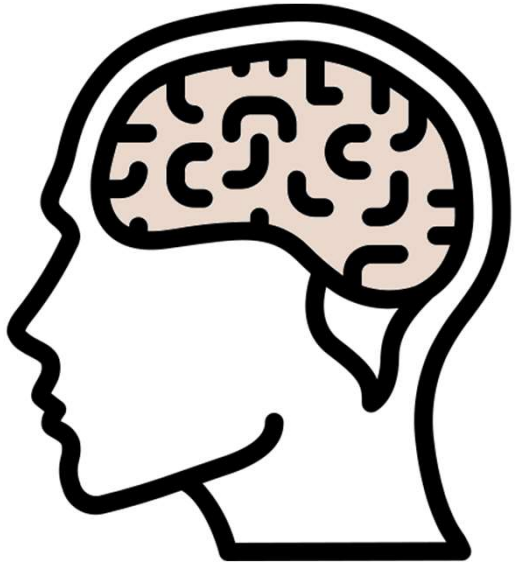


but nothing like the biological neural networks...

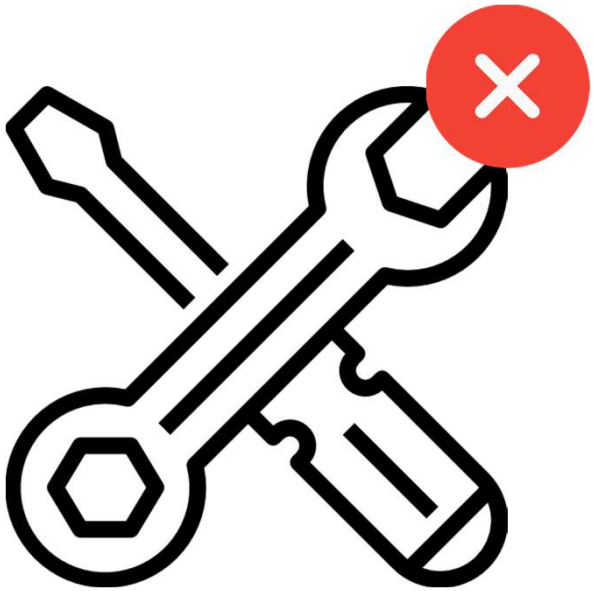


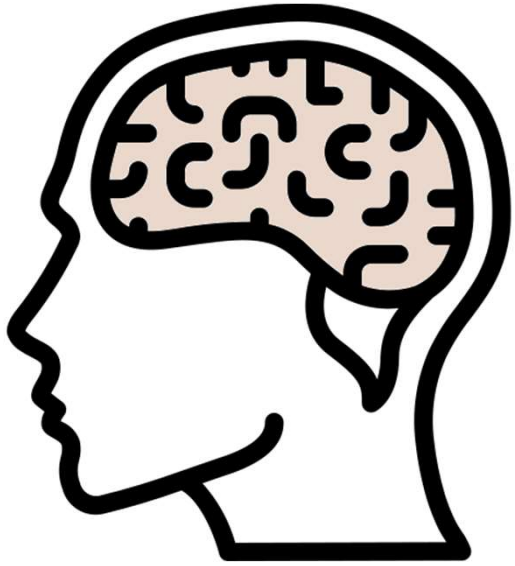
In Computational Neuroscience we take the opposite approach:



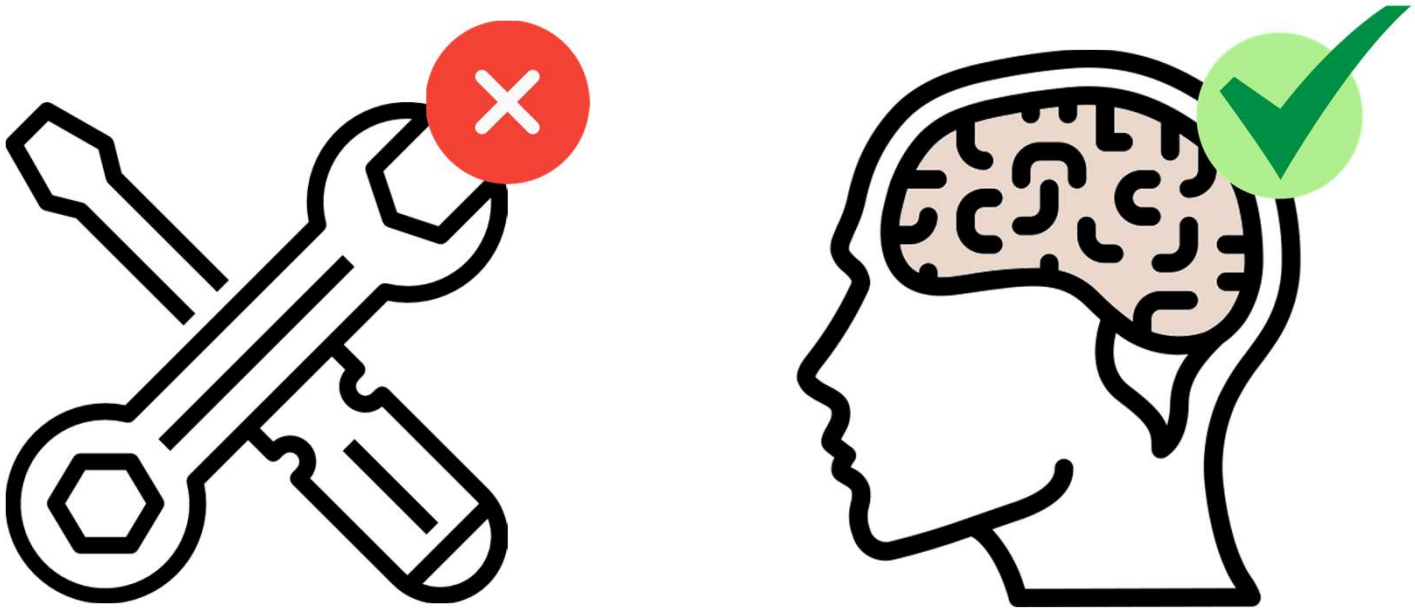


We give less emphasis to the application:

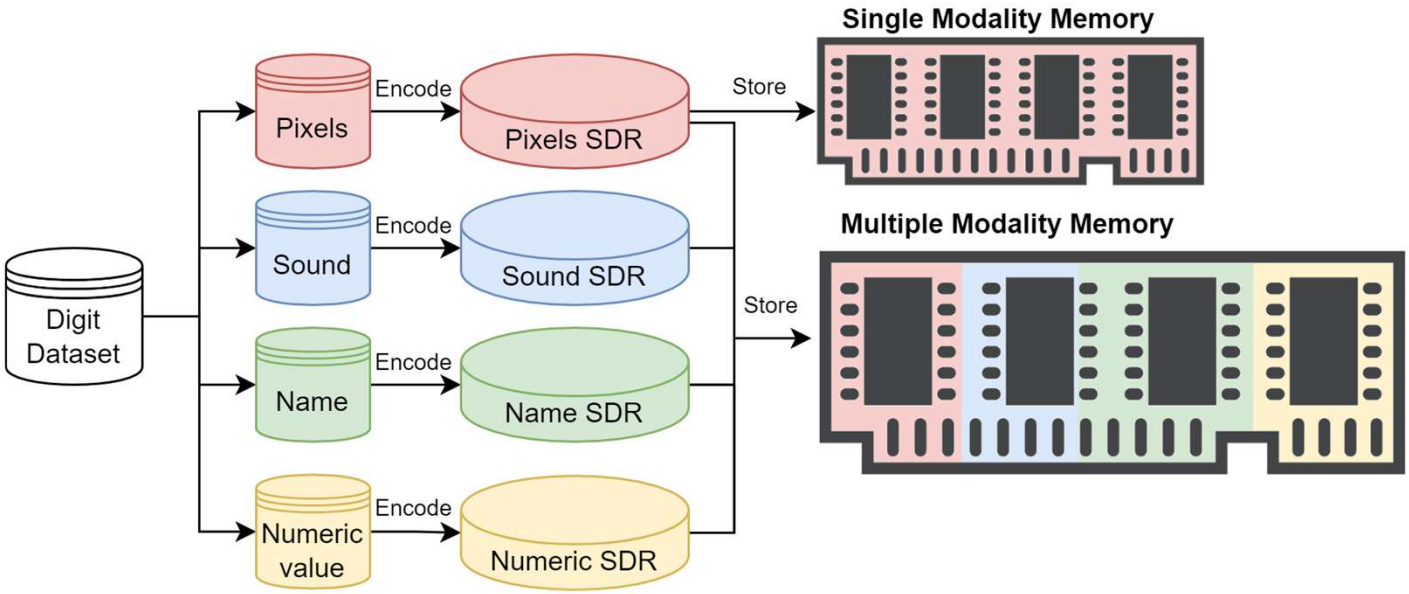




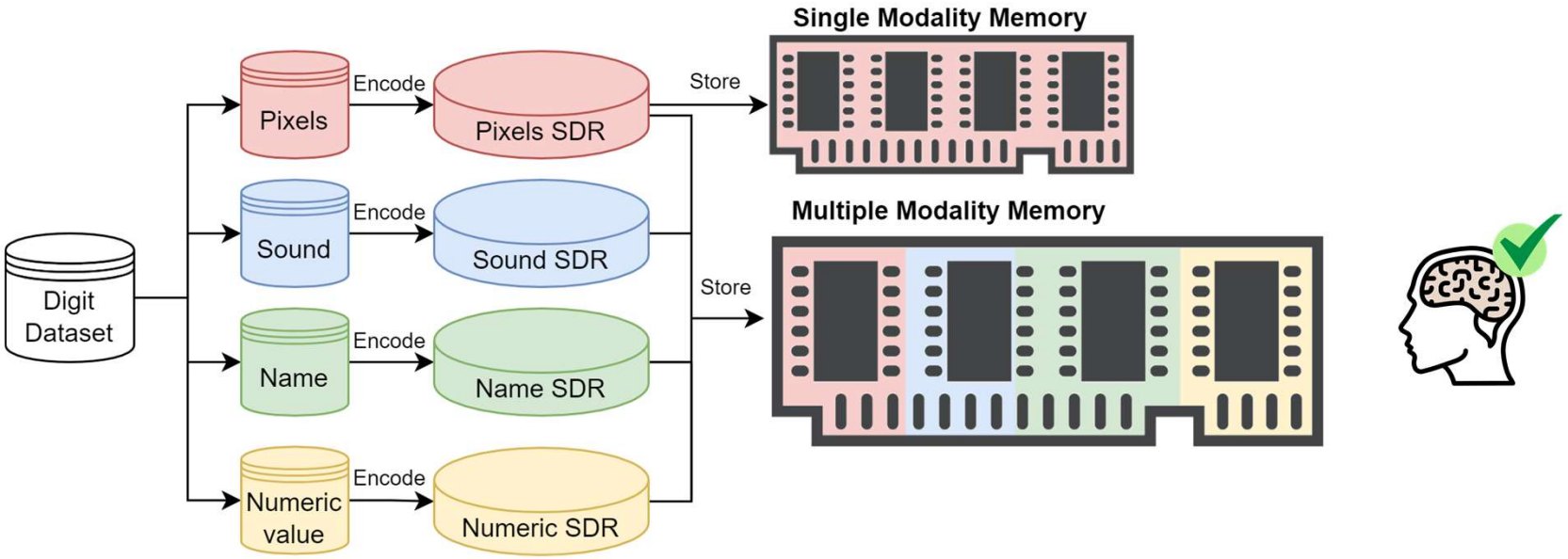
And instead focus on biology as a key part of the model:



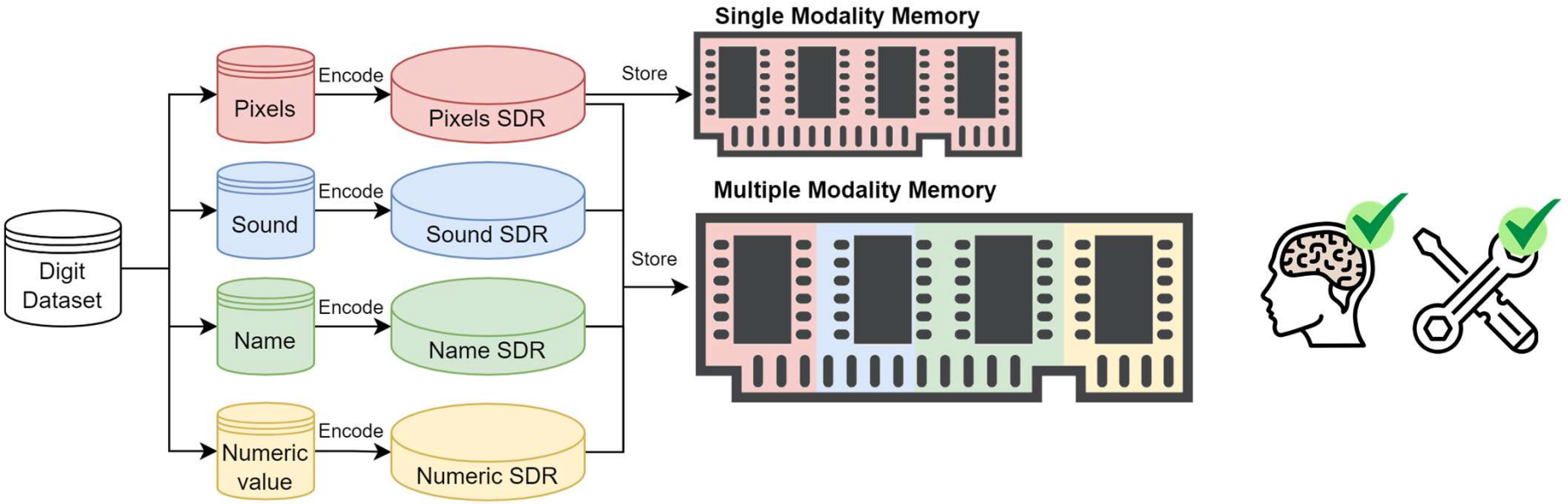
We propose a new architecture:
The Multi-modal Associative Memory



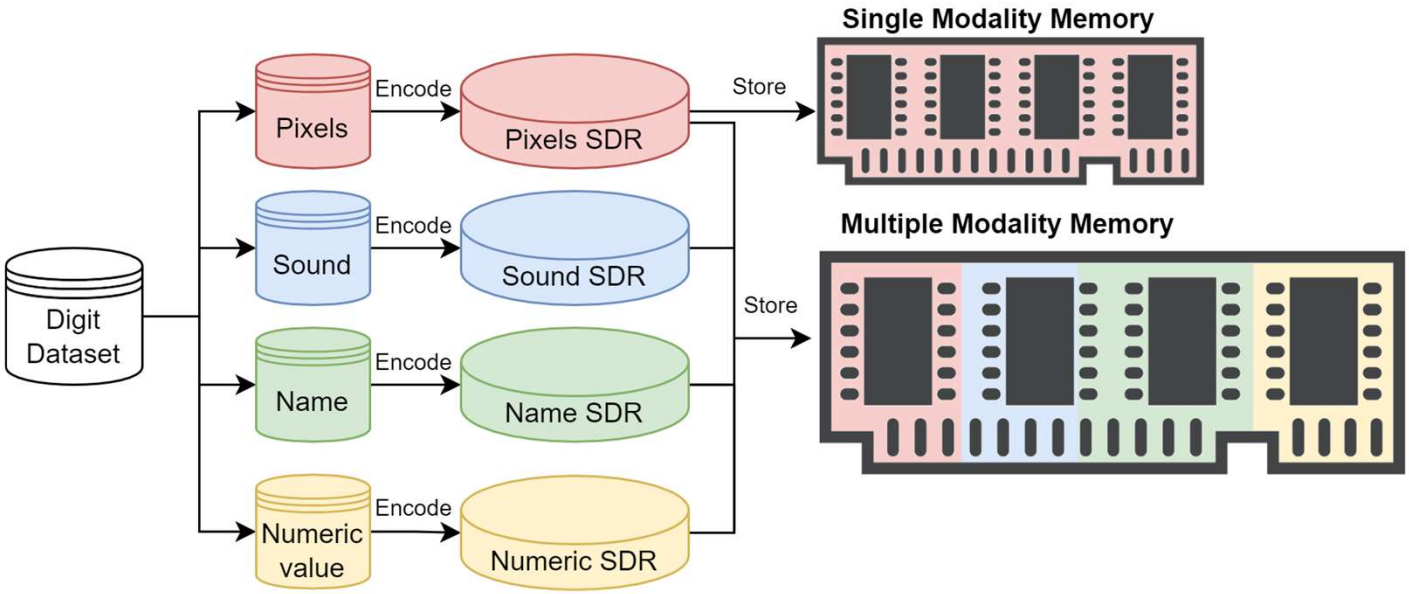
Based on a Biologically-Plausible Model



But also usefull in practice



Main advantages of the proposed Model



- 1) **Flexible** (does completion, classification, generation) with a single model.
- 2) Is **Biologically Plausible** (Uses Hebb's Postulate to train).
- 3) Has a very **efficient** training rule (single Pass training).

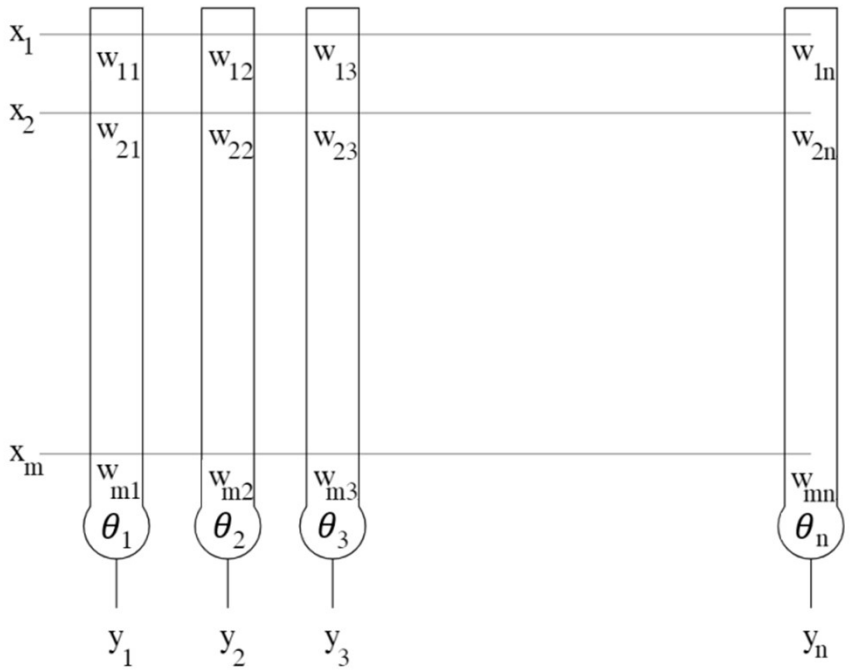
Outline

- Background
- Analysis of the Willshaw Network on visual data
- Multi-Modal Willshaw Network
- Conclusion

Outline

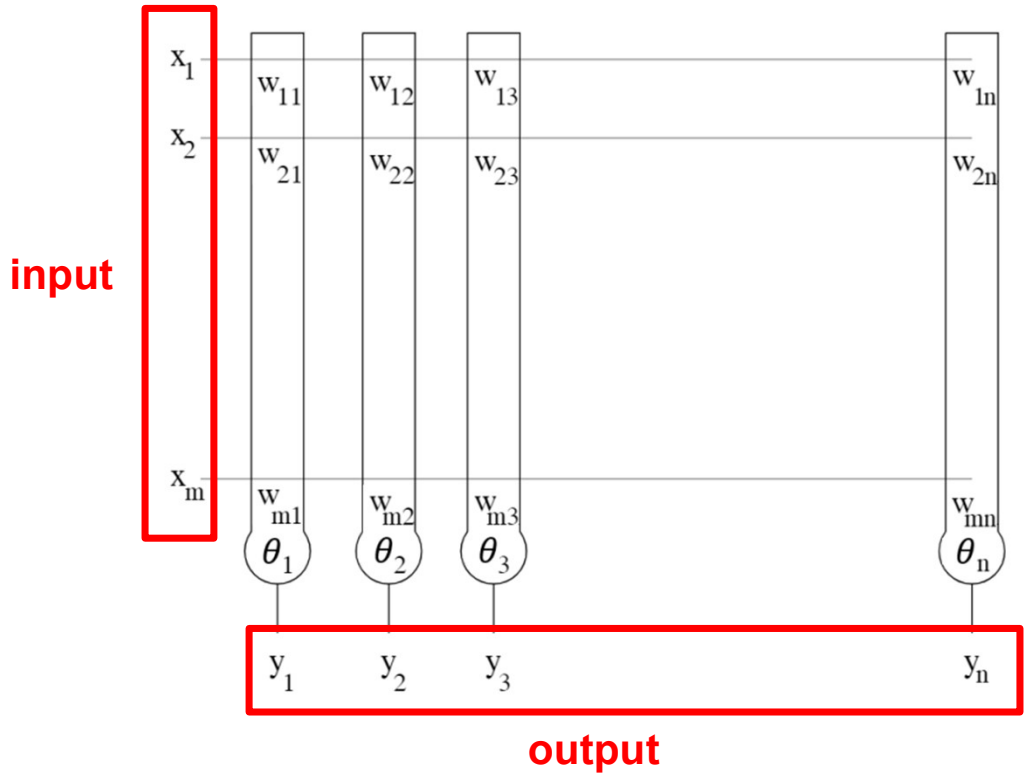
- Background
- Analysis of the Willshaw Network on visual data
- Multi-Modal Willshaw Network
- Conclusion

Meet our model: The Willshaw Network

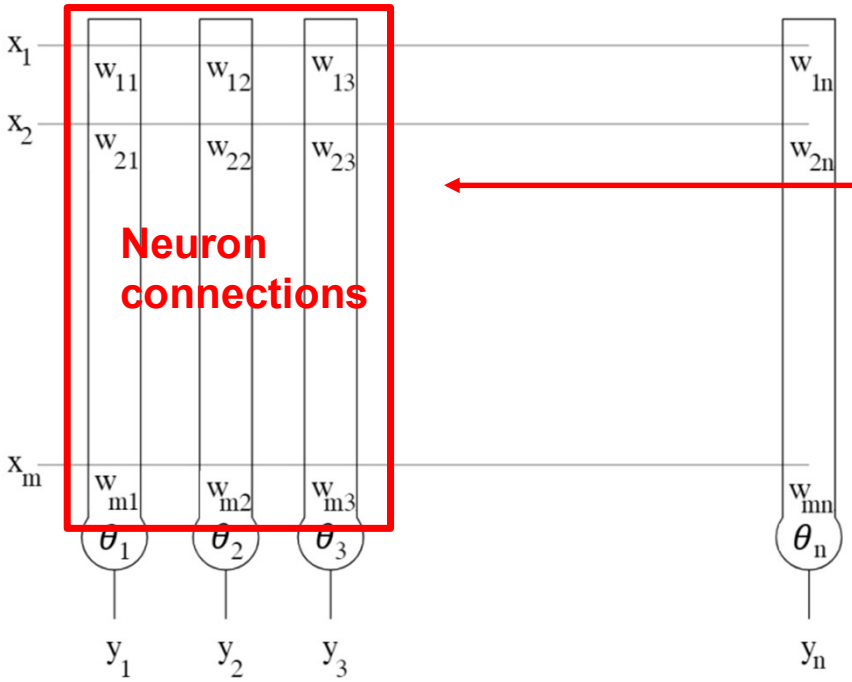


Wichert, A.M.: Principles of Quantum Artificial Intelligence: Quantum Problem Solving and Machine Learning. World scientific (2020)

A single-layered Network with binary inputs-outputs



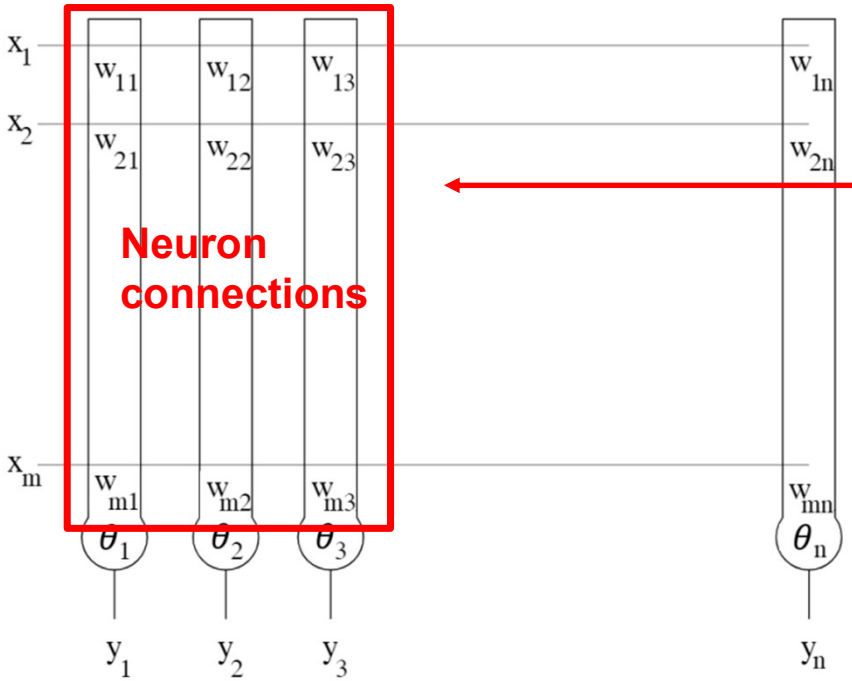
And binary weights



Hebbian Training Rule

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

And learns extremely fast

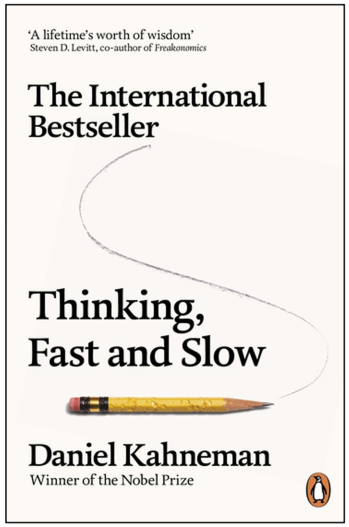


Hebbian Training Rule

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

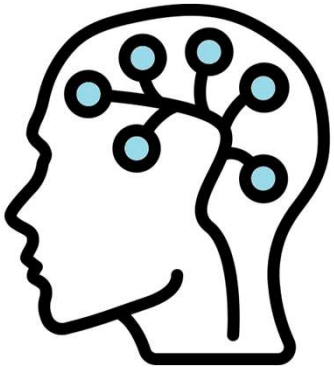
In a single pass through the data

The WN is an Associative Memory (AM)



Biological AM

MIMICS



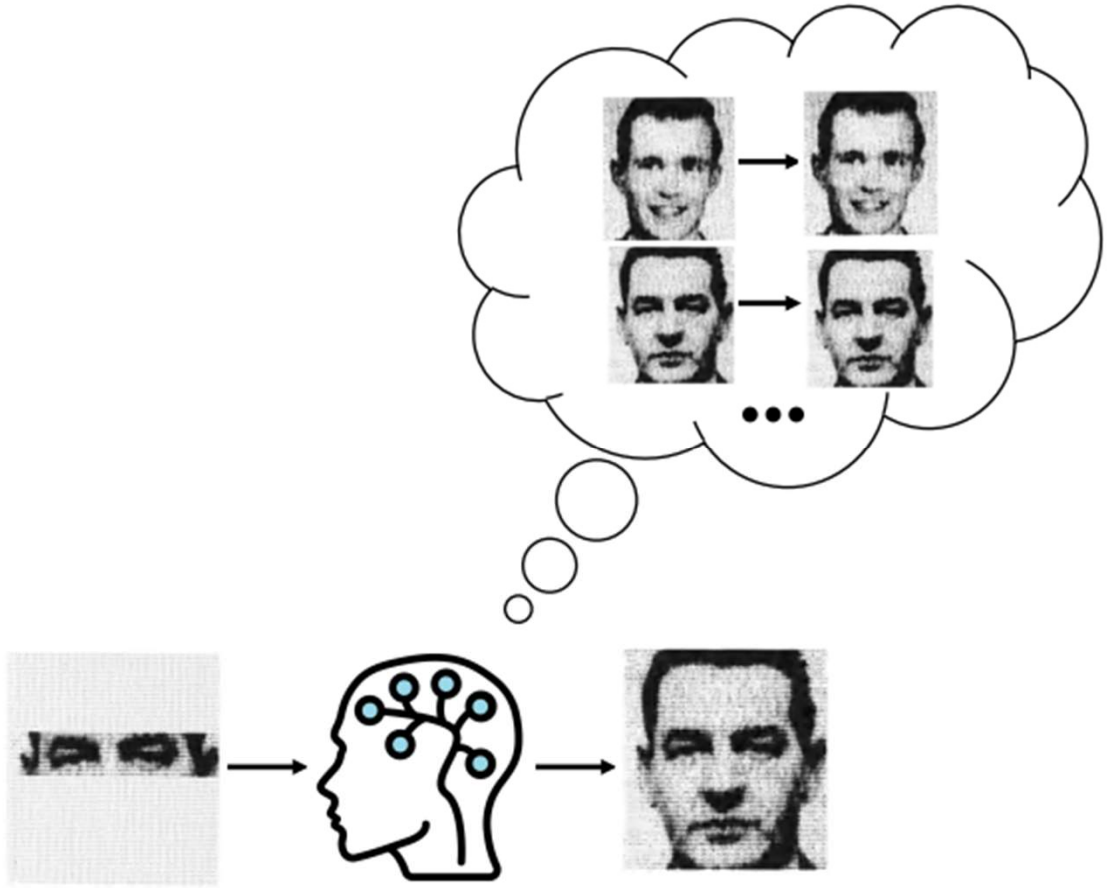
Artificial AM

*"Psychologists think of ideas as nodes in a vast network, called **associative memory**, in which each idea is linked to many others."*

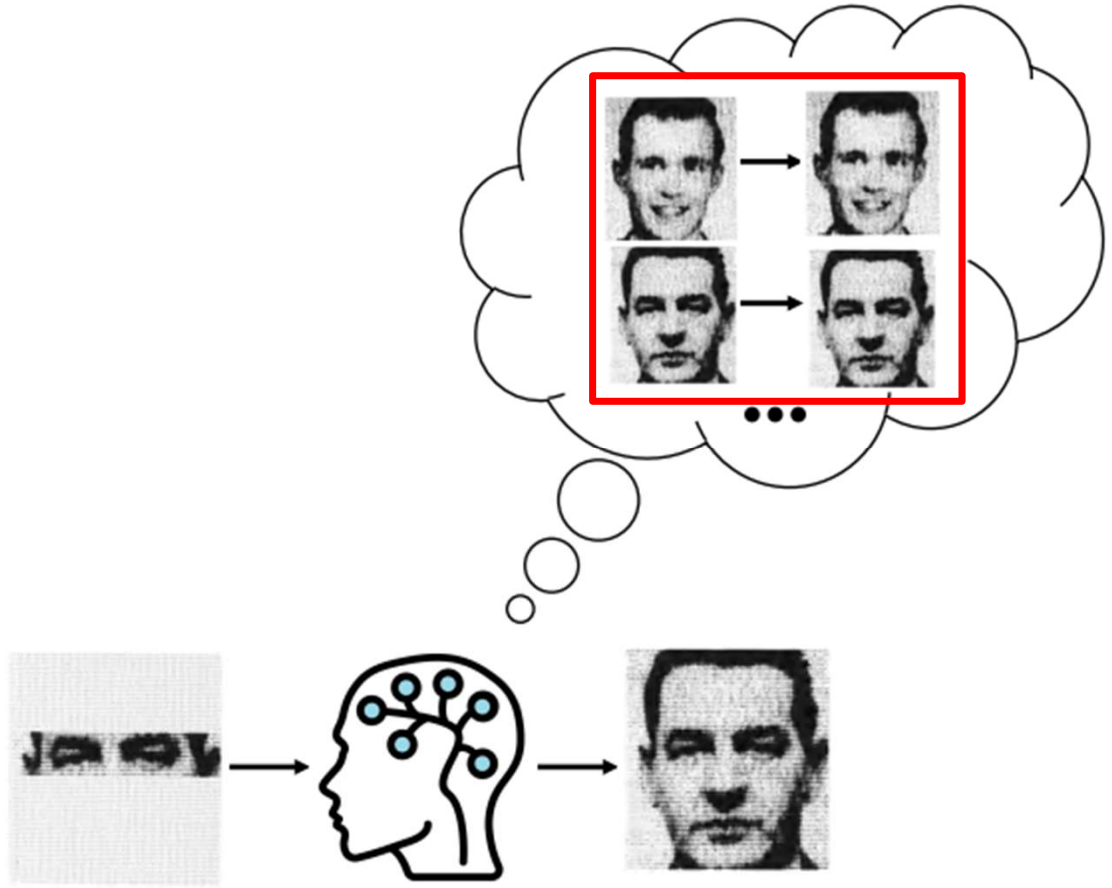
- ***Willshaw Network***
- *Hopfield Network*
- *Etc…*

Daniel, K.: Thinking, fast and slow (2017)

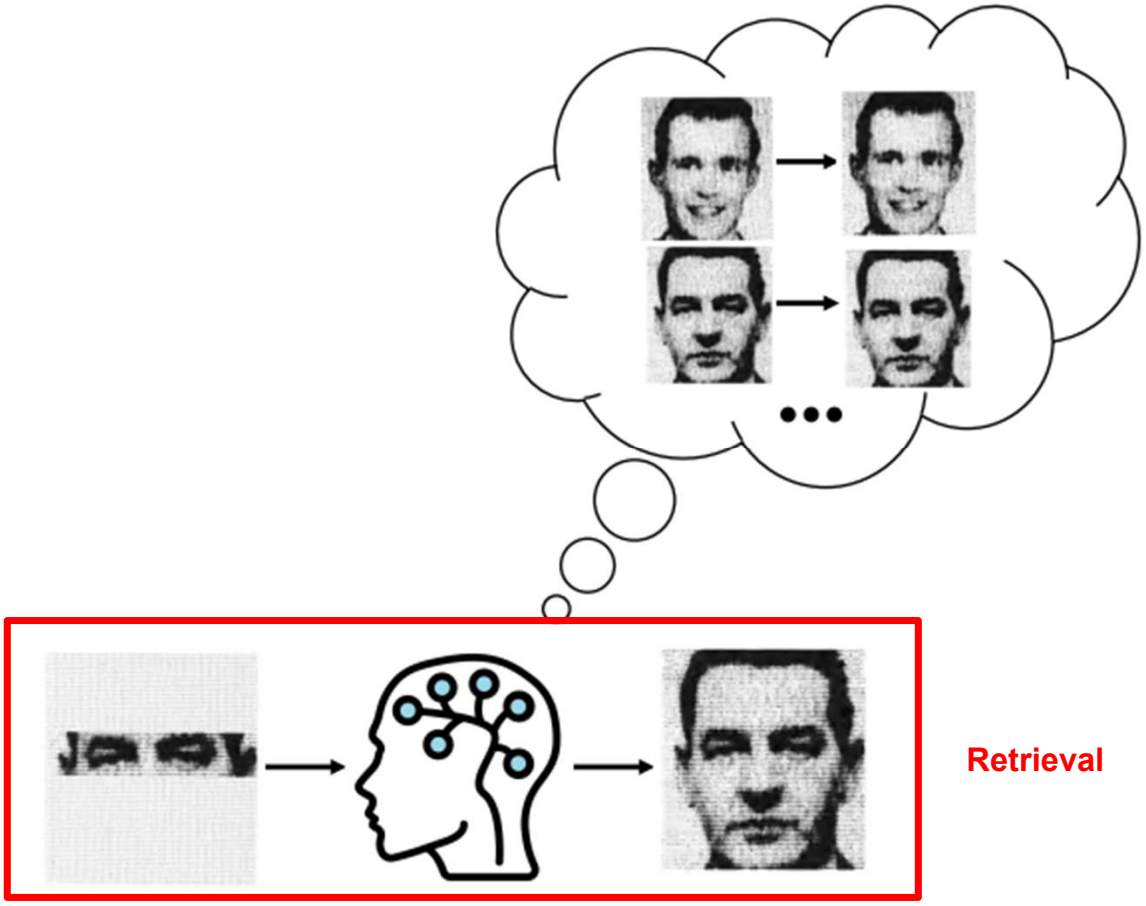
Associative Memory models in a nutshell:



They store associations Between vectors



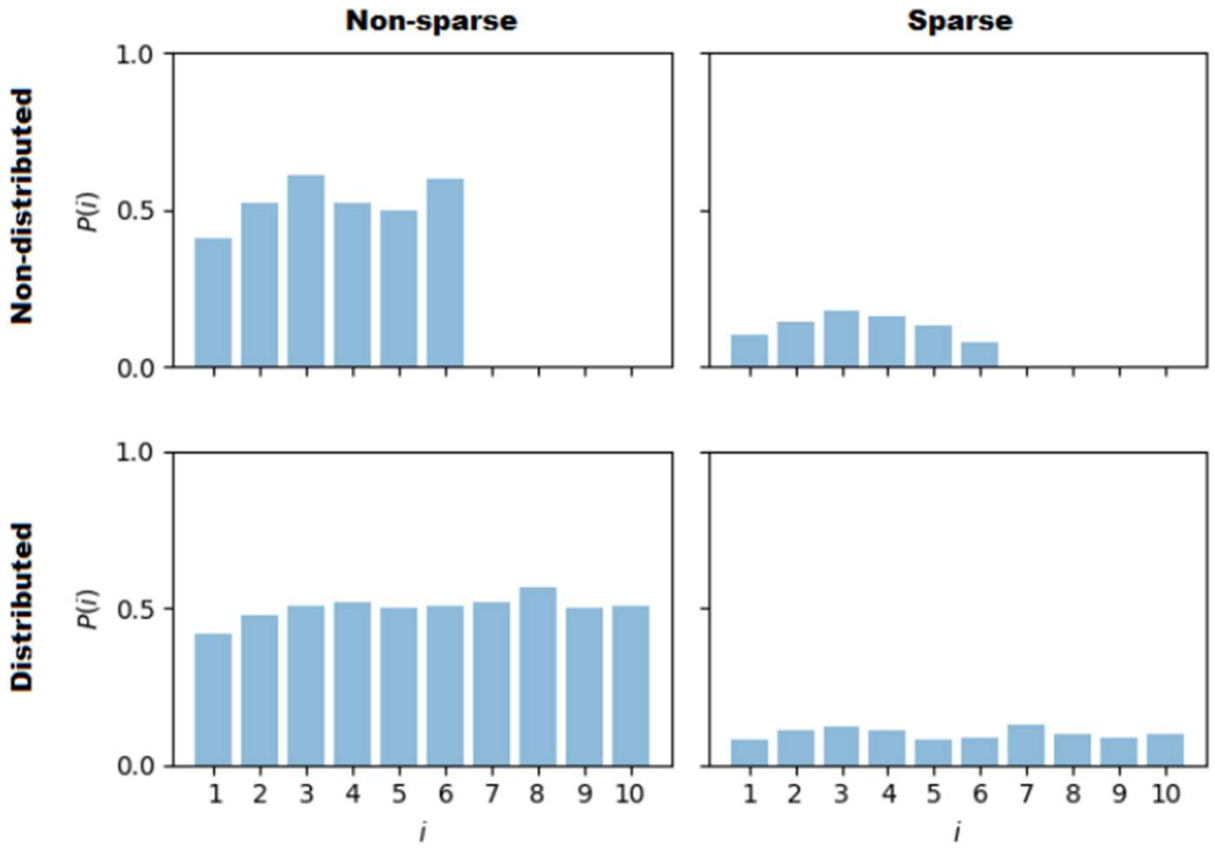
And COMPLETE missing informaion: Retrieval



Retrieval

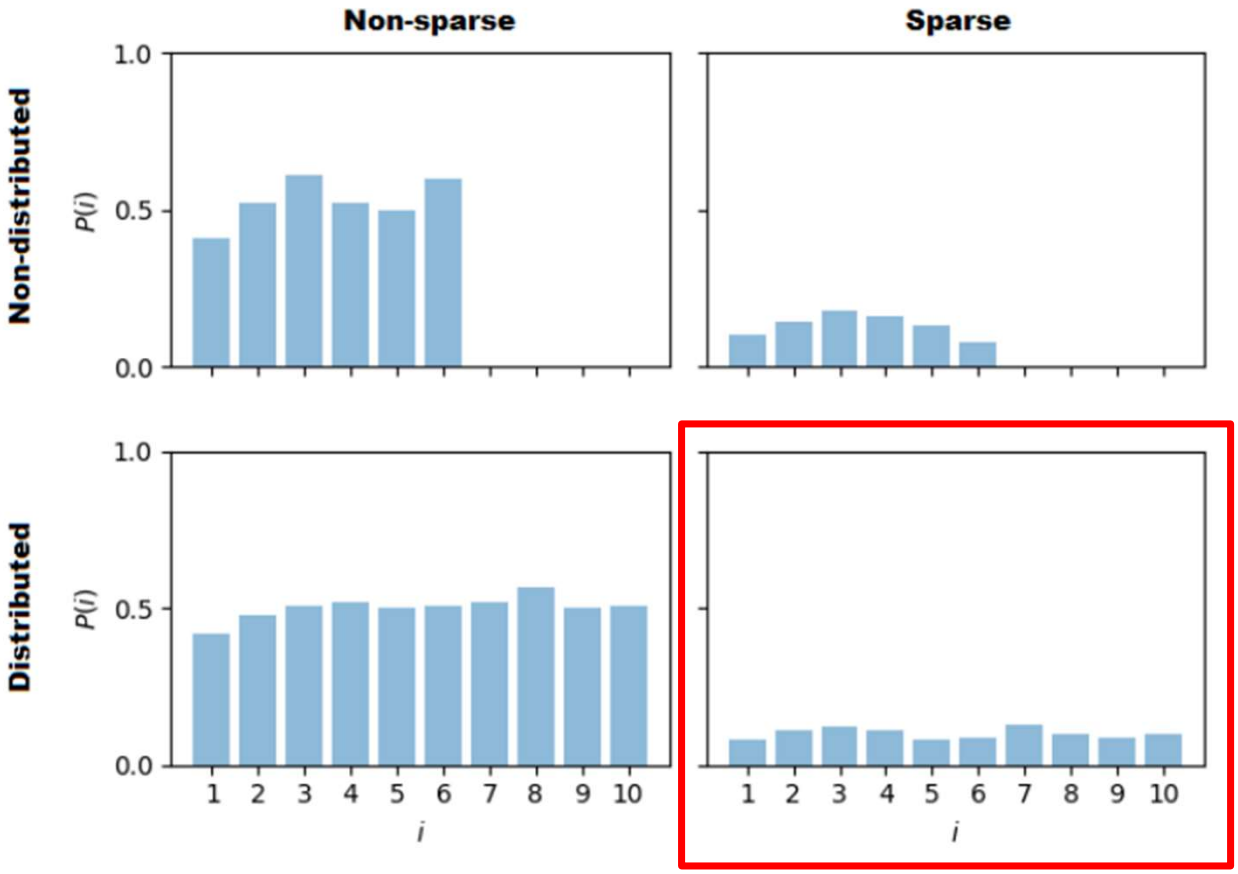
The memory is highly efficient:

$$P_i = \frac{1}{M} \sum_{\mu=1}^{M} x_i^{\mu}$$

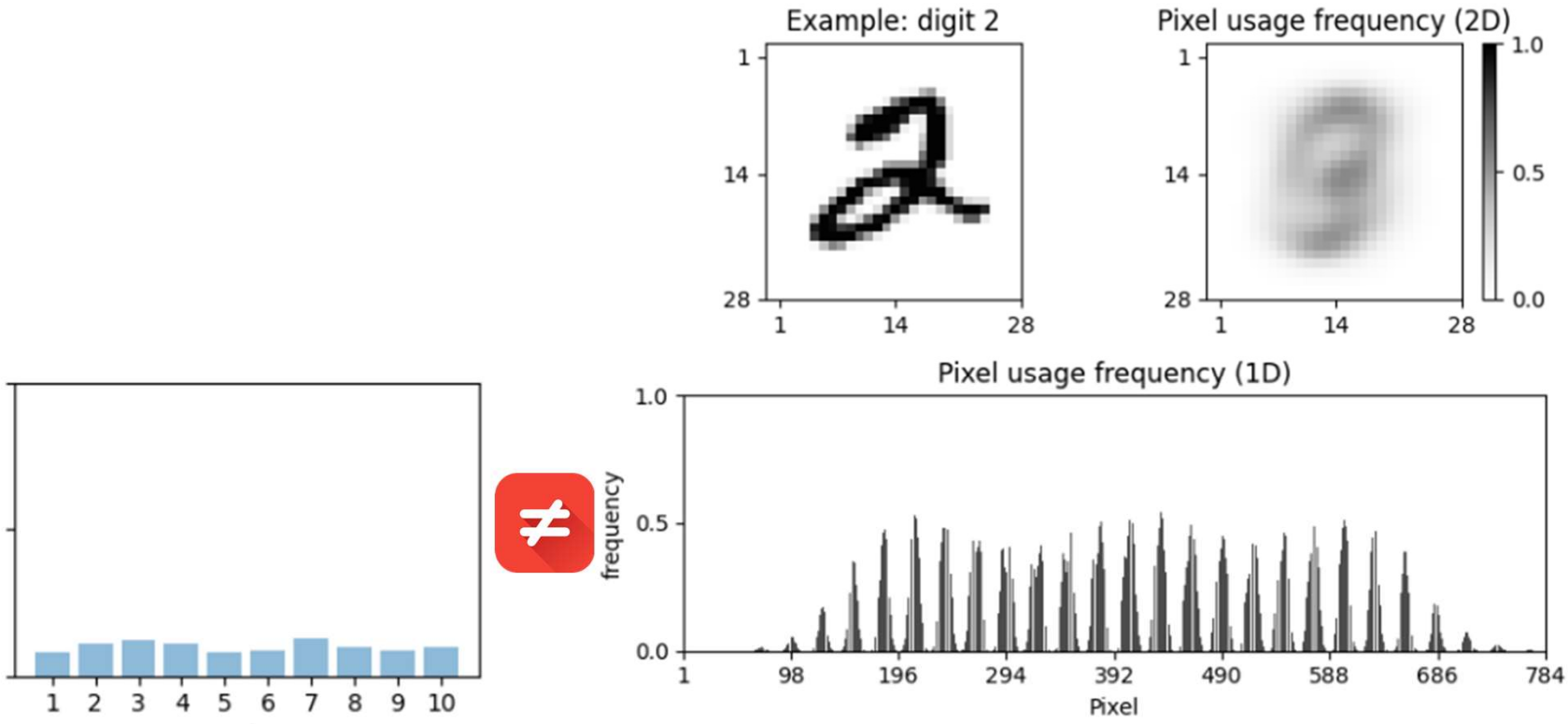


But only when its inputs are Sparse Distributes Representations (SDRs)

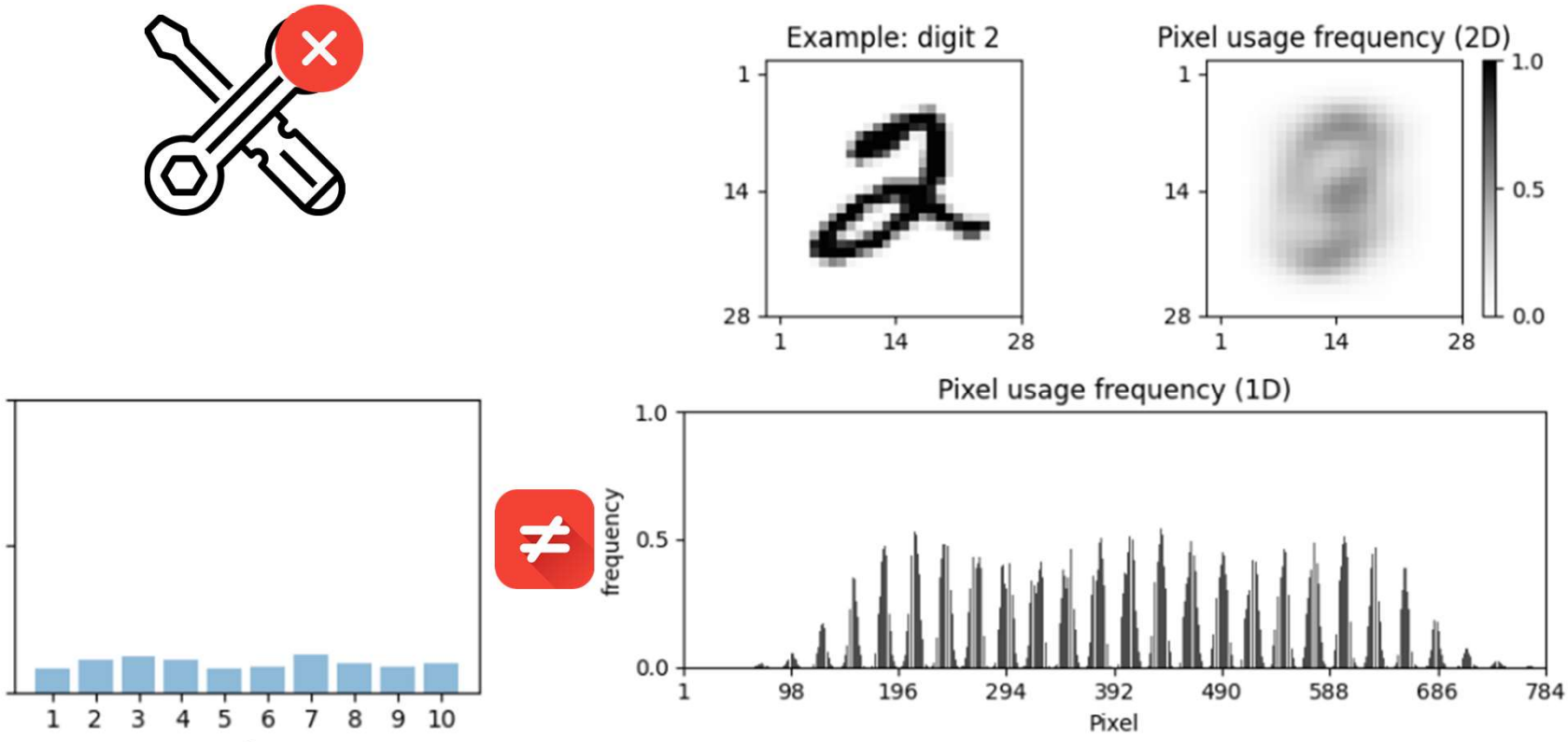
$$P_i = \frac{1}{M} \sum_{\mu=1}^{M} x_i^{\mu}$$



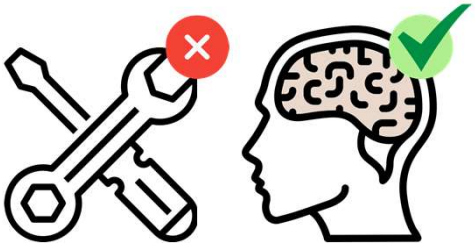
And real data is not like that



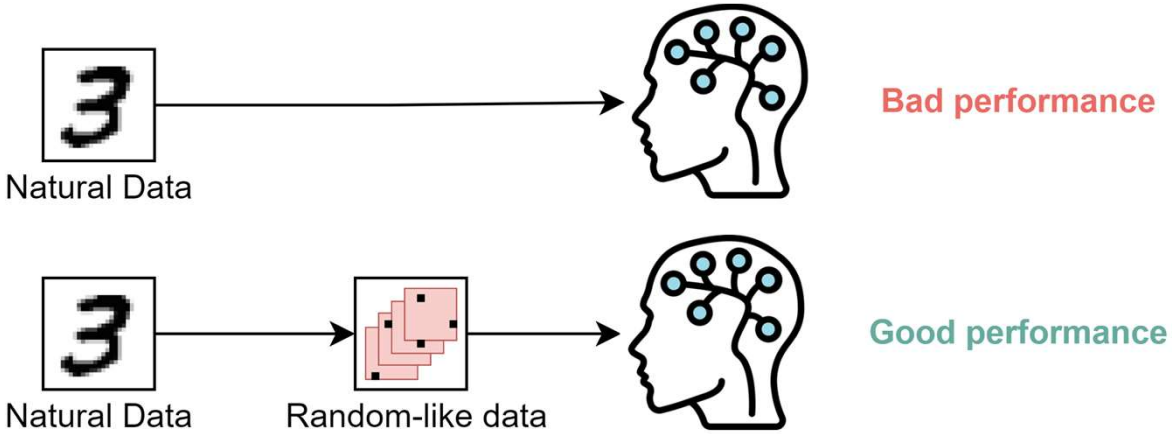
The sparse coding problema (SCP)



Associative memories have not been used in practice...

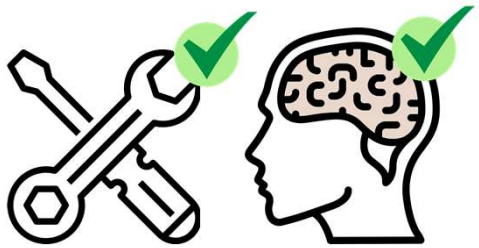


"(...) ***associative memories*** *still pose interesting questions to be addressed. As we will see,* ***their performance when dealing with natural, non-random data is extremely low****. Without special data* ***coding prescriptions****,* ***mostly undiscovered*** *yet, their full potential as a useful engineering tool remains out of reach."*



Sacramento, J, Associative Memory for Binary Image Storage and Retrieval (Unpublished)

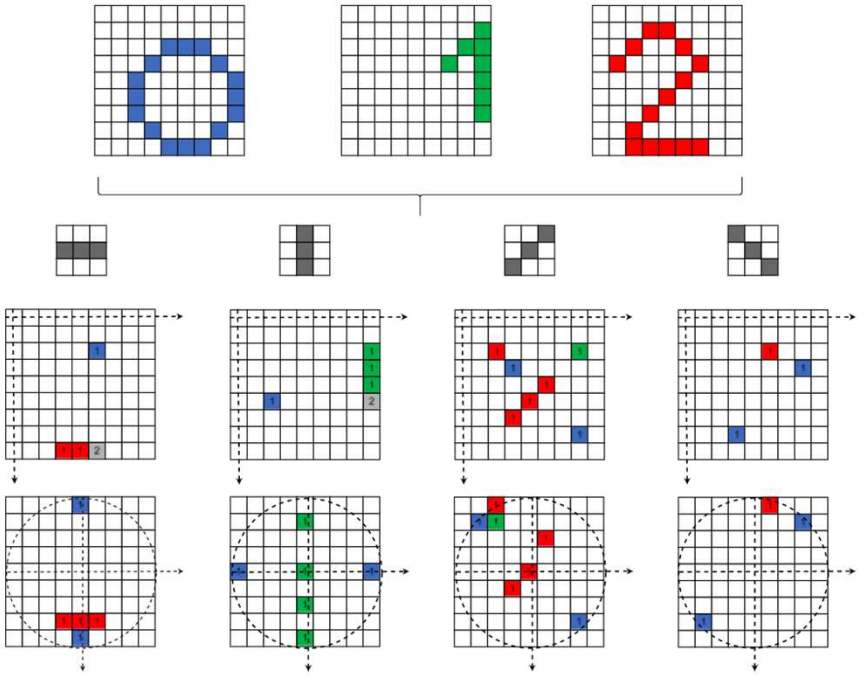
Until recently!



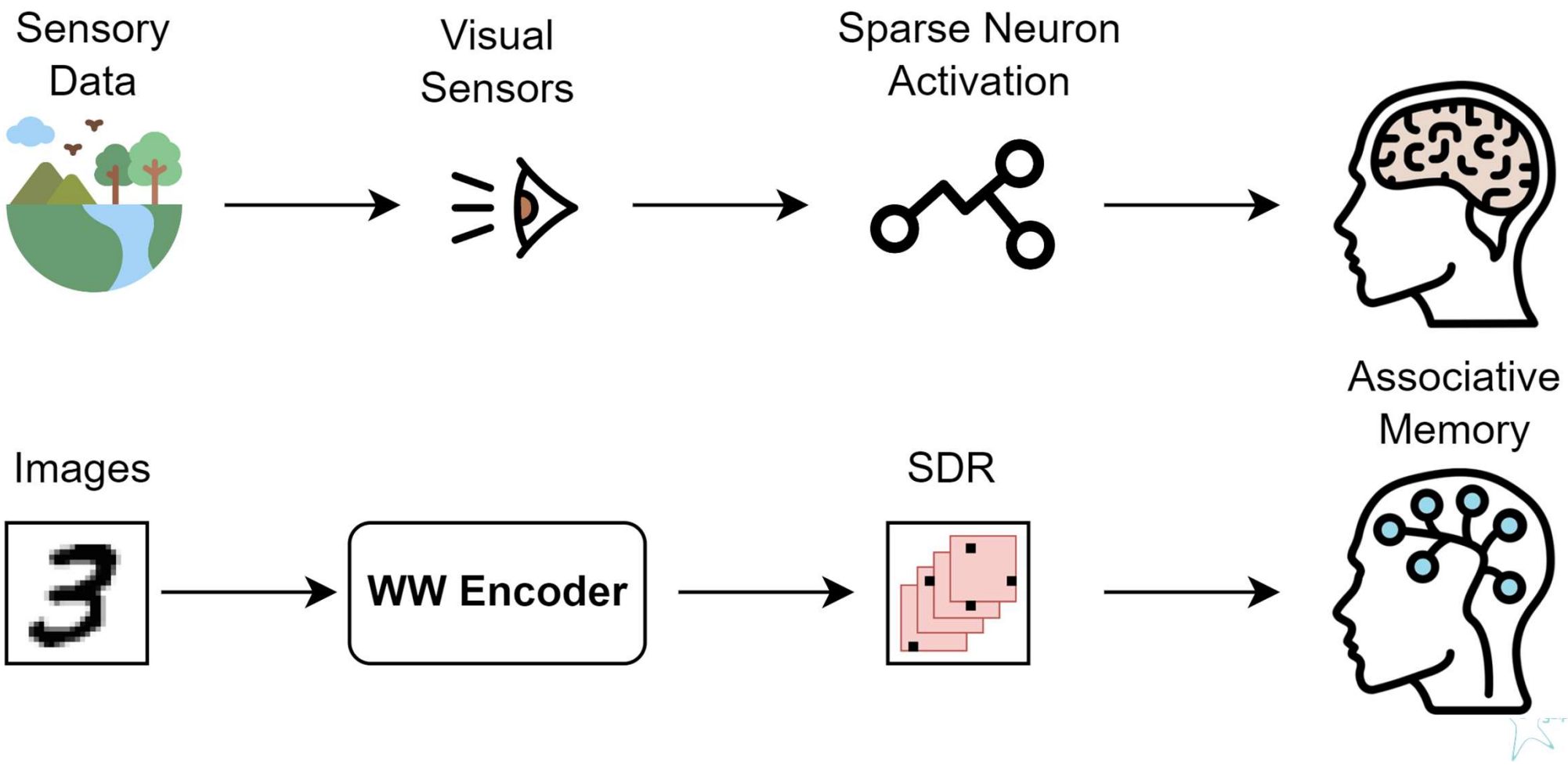
LETTER Communicated by Andreas Knoblauch

Storing Object-Dependent Sparse Codes in a Willshaw Associative Network

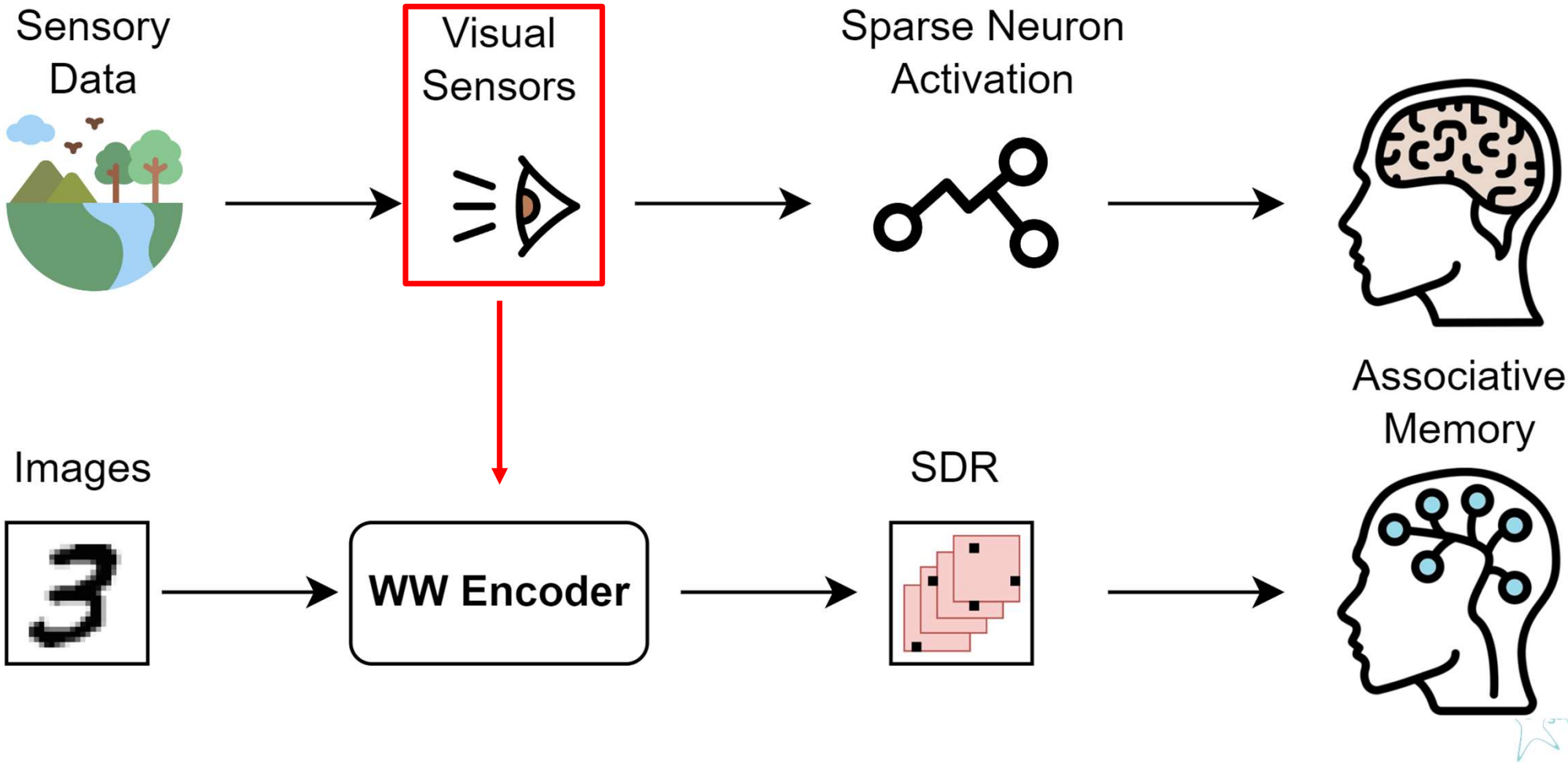
Luis Sa-Couto
luis.sa.couto@tecnico.ulisboa.pt
Andreas Wichert
andreas.wichert@tecnico.ulisboa.pt
Department of Computer Science and Engineering, INESC-ID & Instituto Superior Técnico, University of Lisbon, 2744-016 Porto Salvo, Portugal



“What-where” Encoder – Solution to the SCP



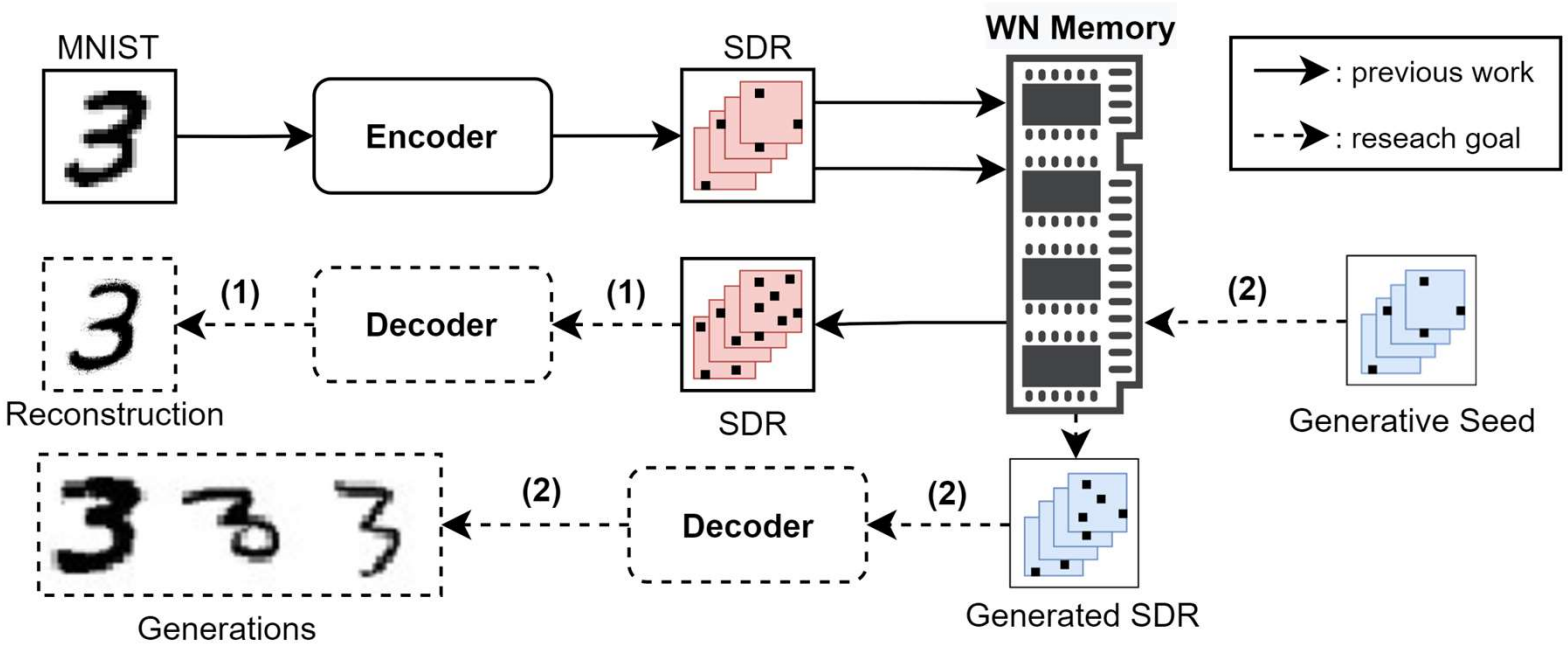
An encoder inspired in the Visual Cortex



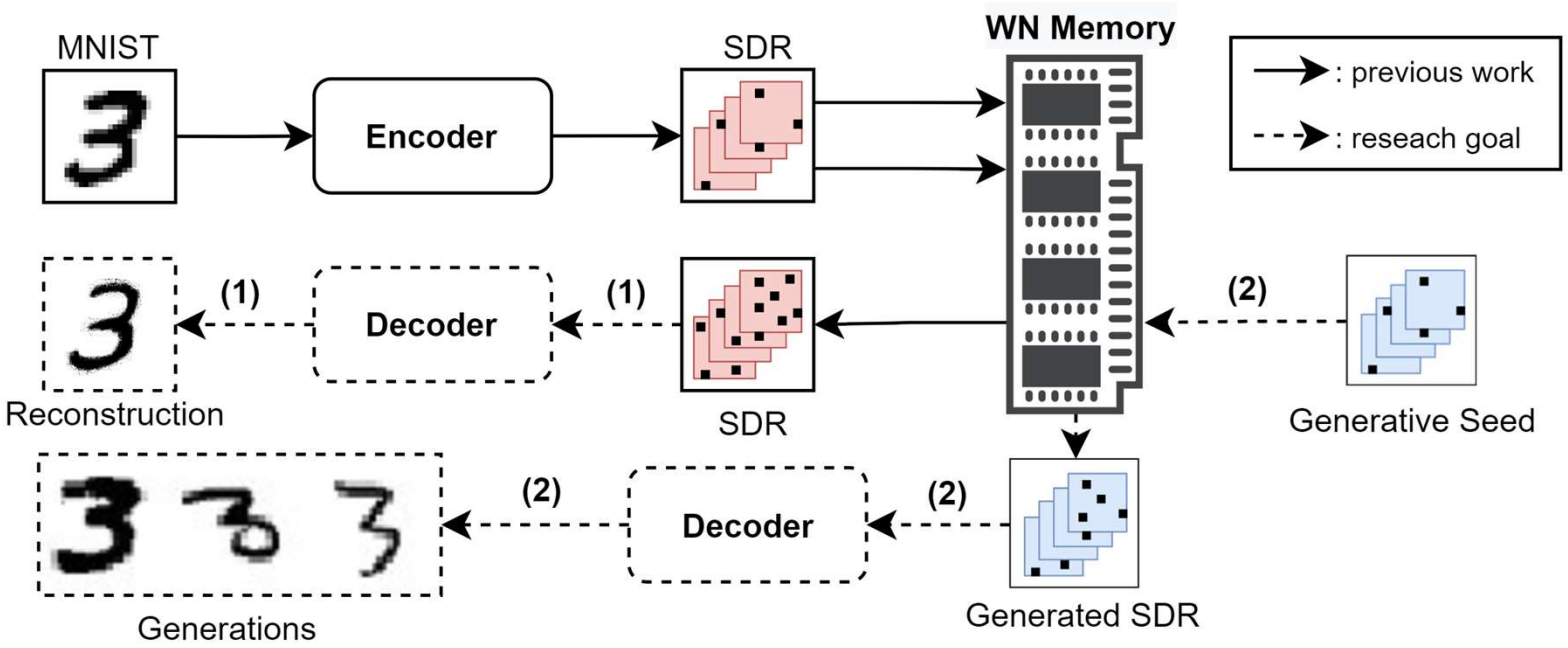
Outline

- Background
- Analysis of the Retrieval of Visual Patterns
- Multi-Modal Willshaw Network
- Conclusion

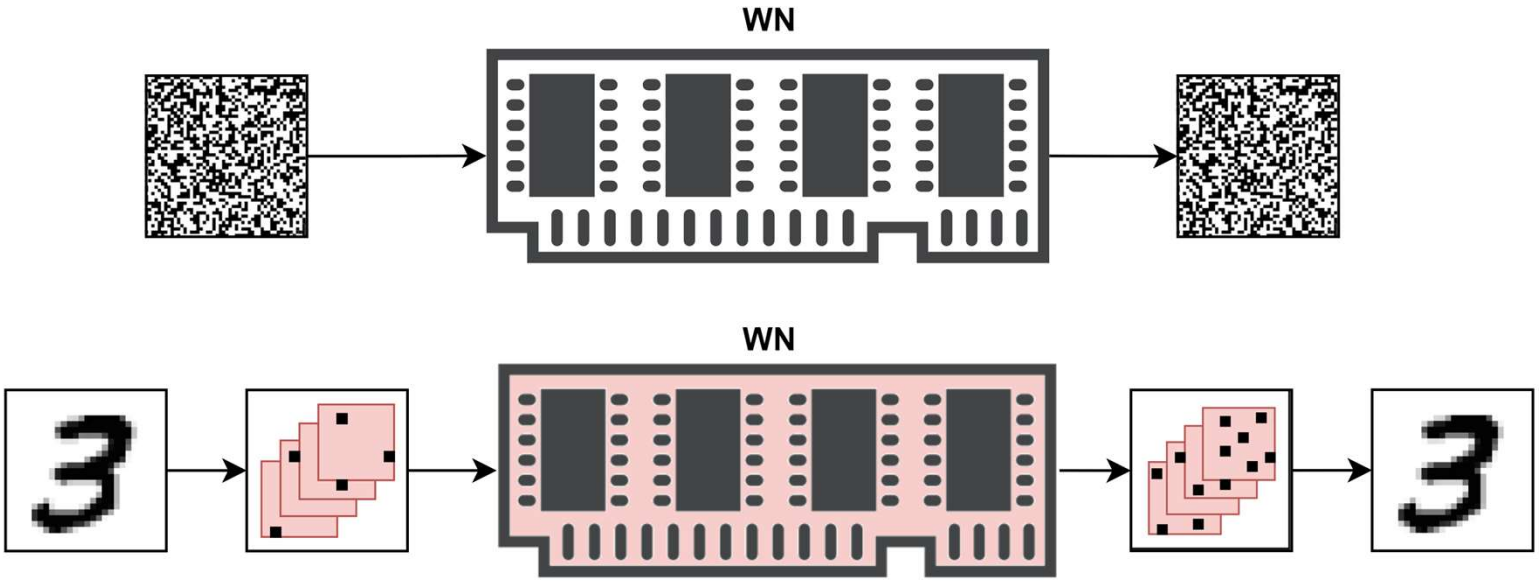
A large part of this Master Thesis



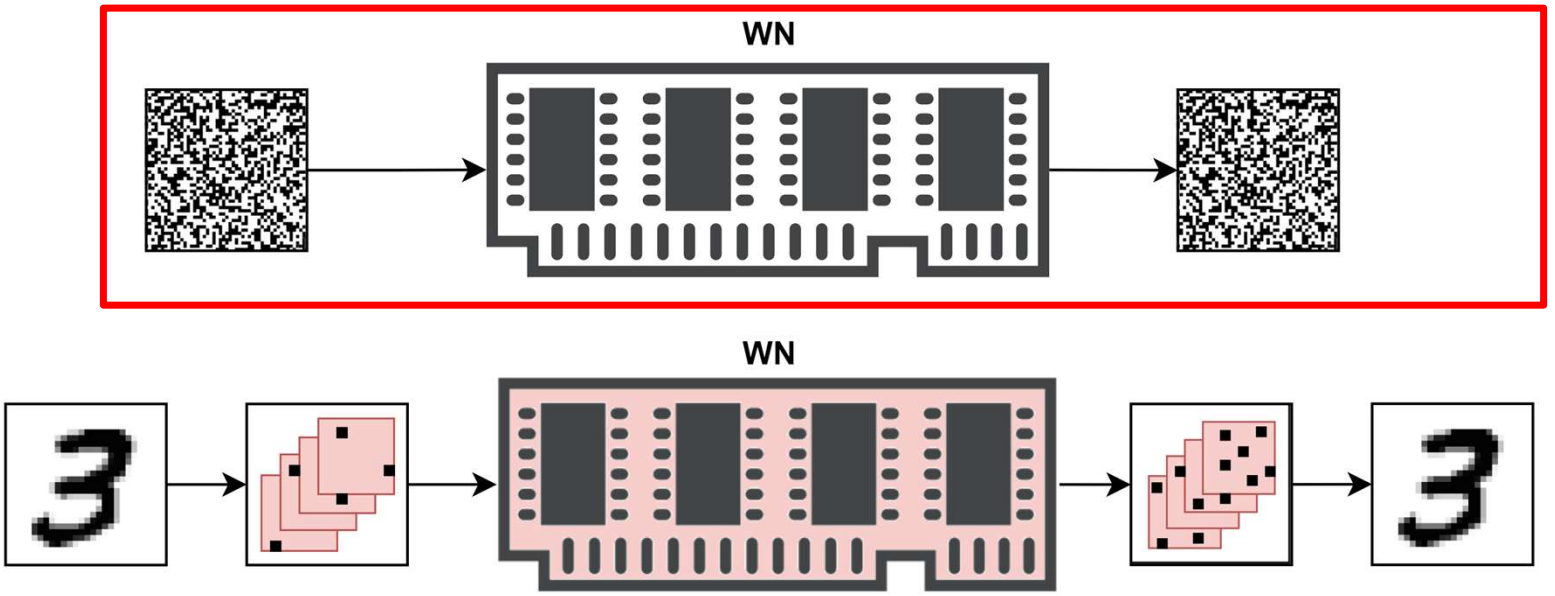
But let us focus on the main findings



The goal:

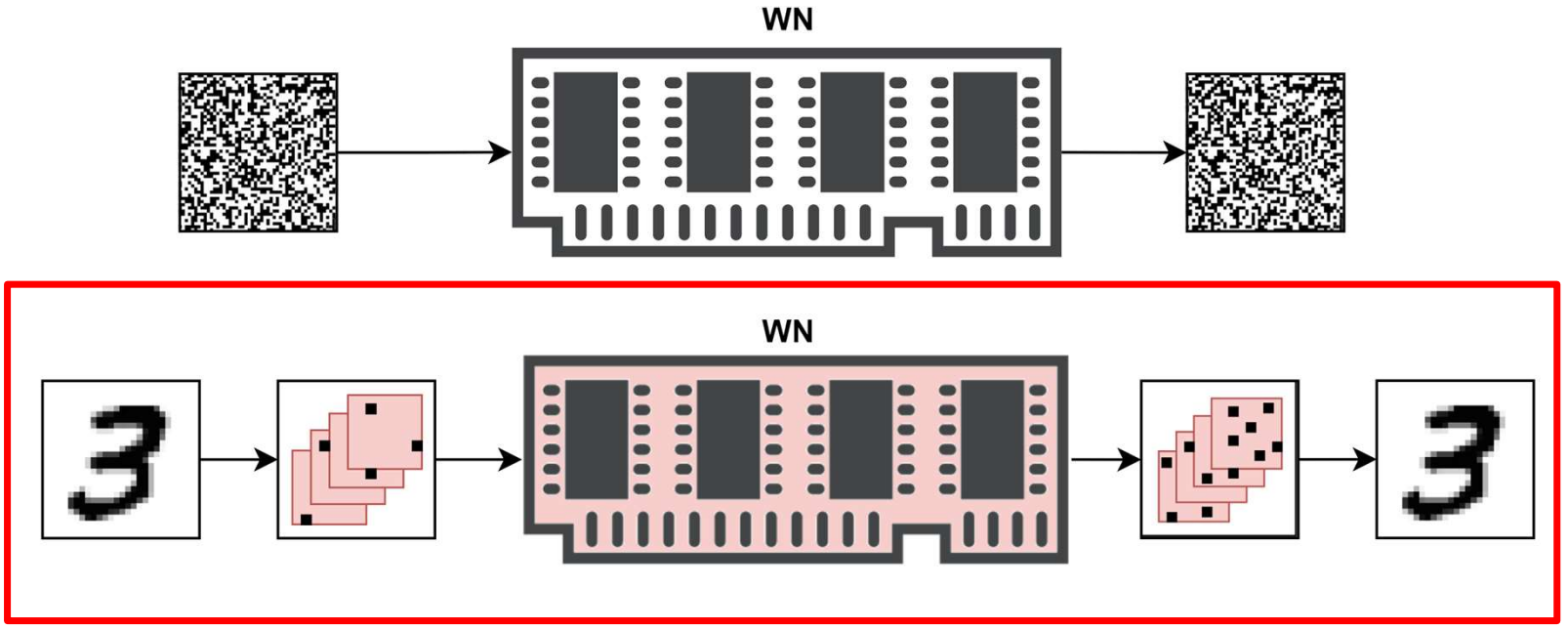


The goal:



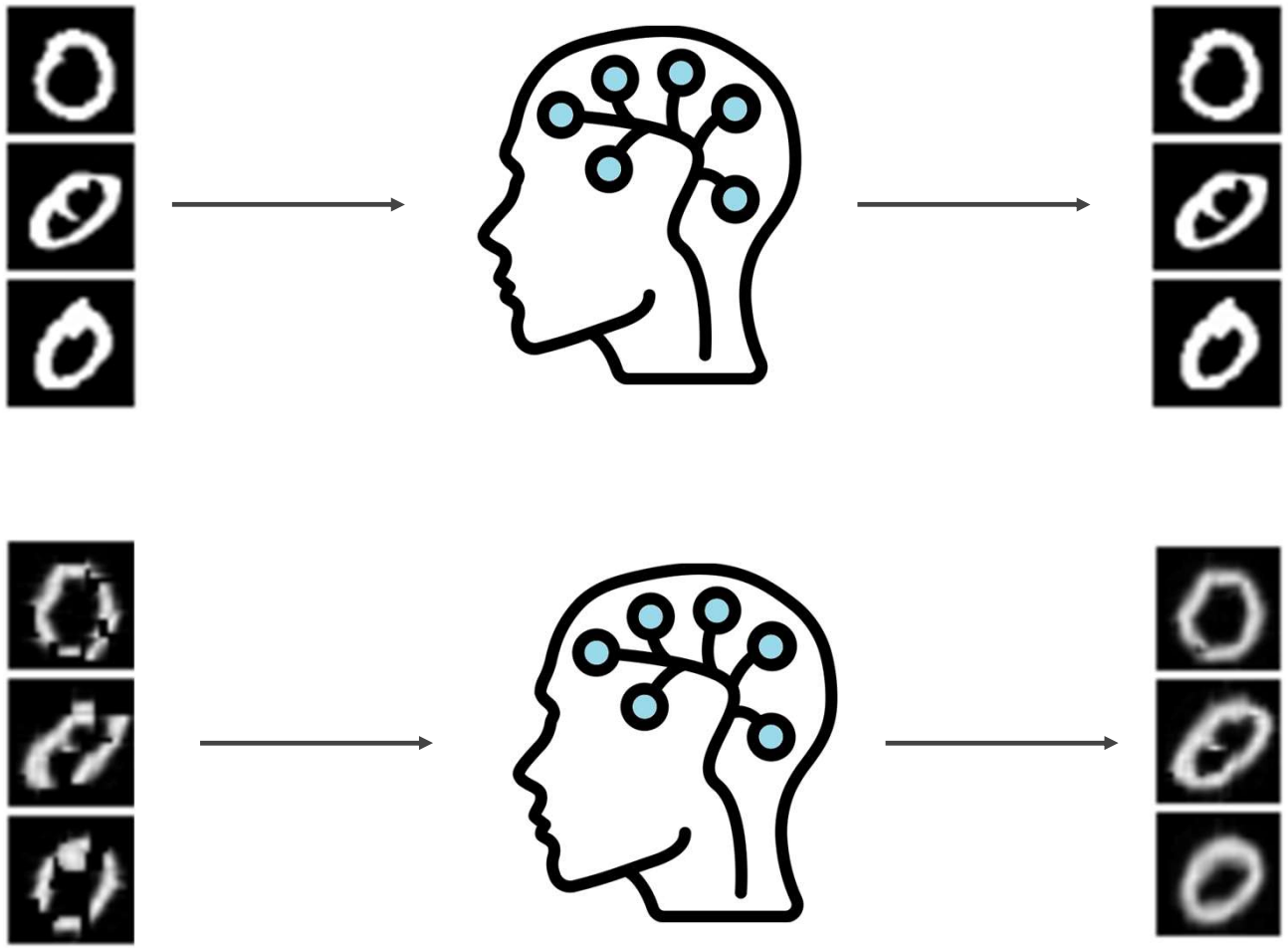
Literature focuses on theoretical scenarios

The goal:

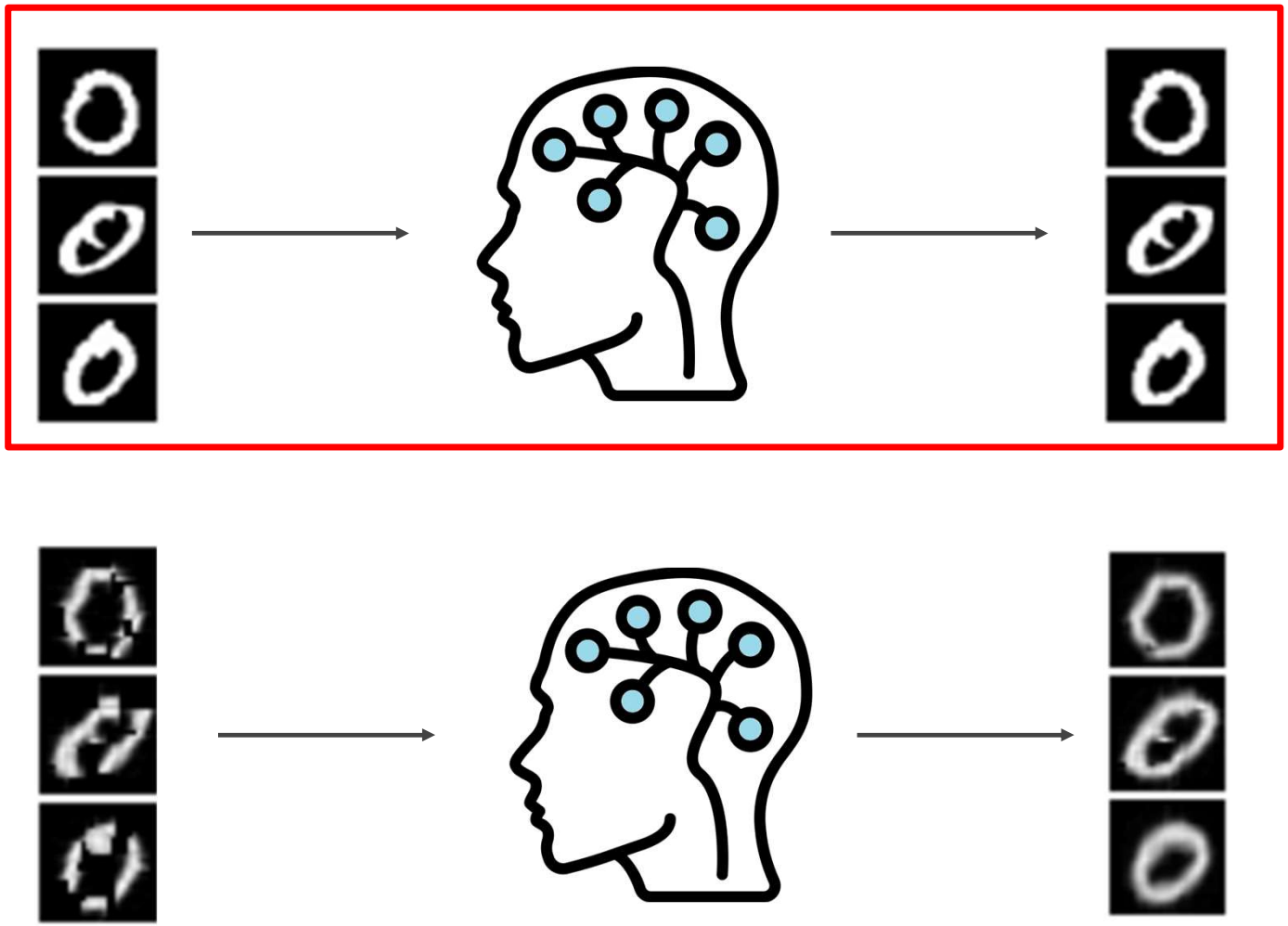


We confirm the literature with the our SDRs

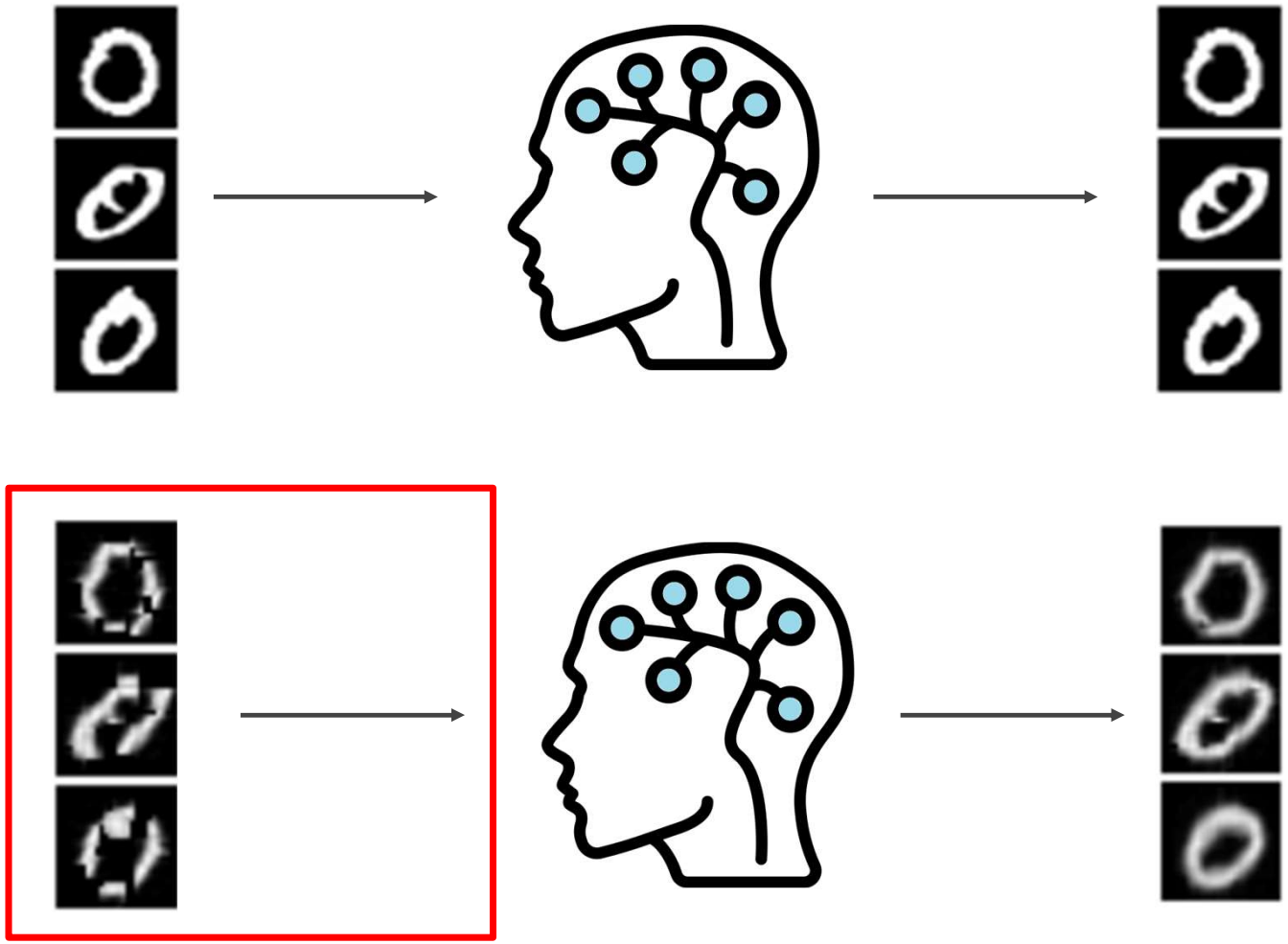
ONE Key result: The WN is great at COMPLETION



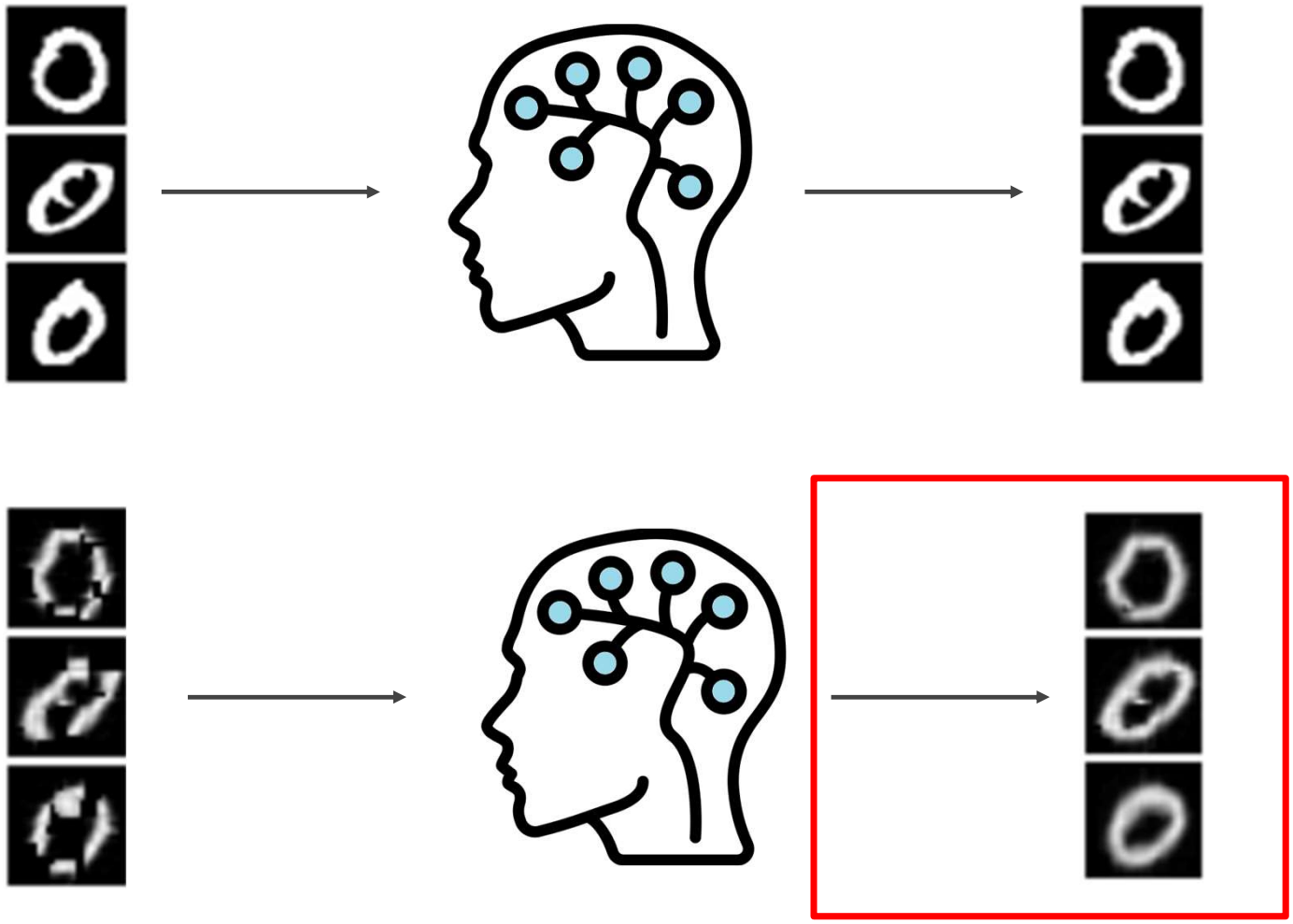
When it stores images in Auto-Association:



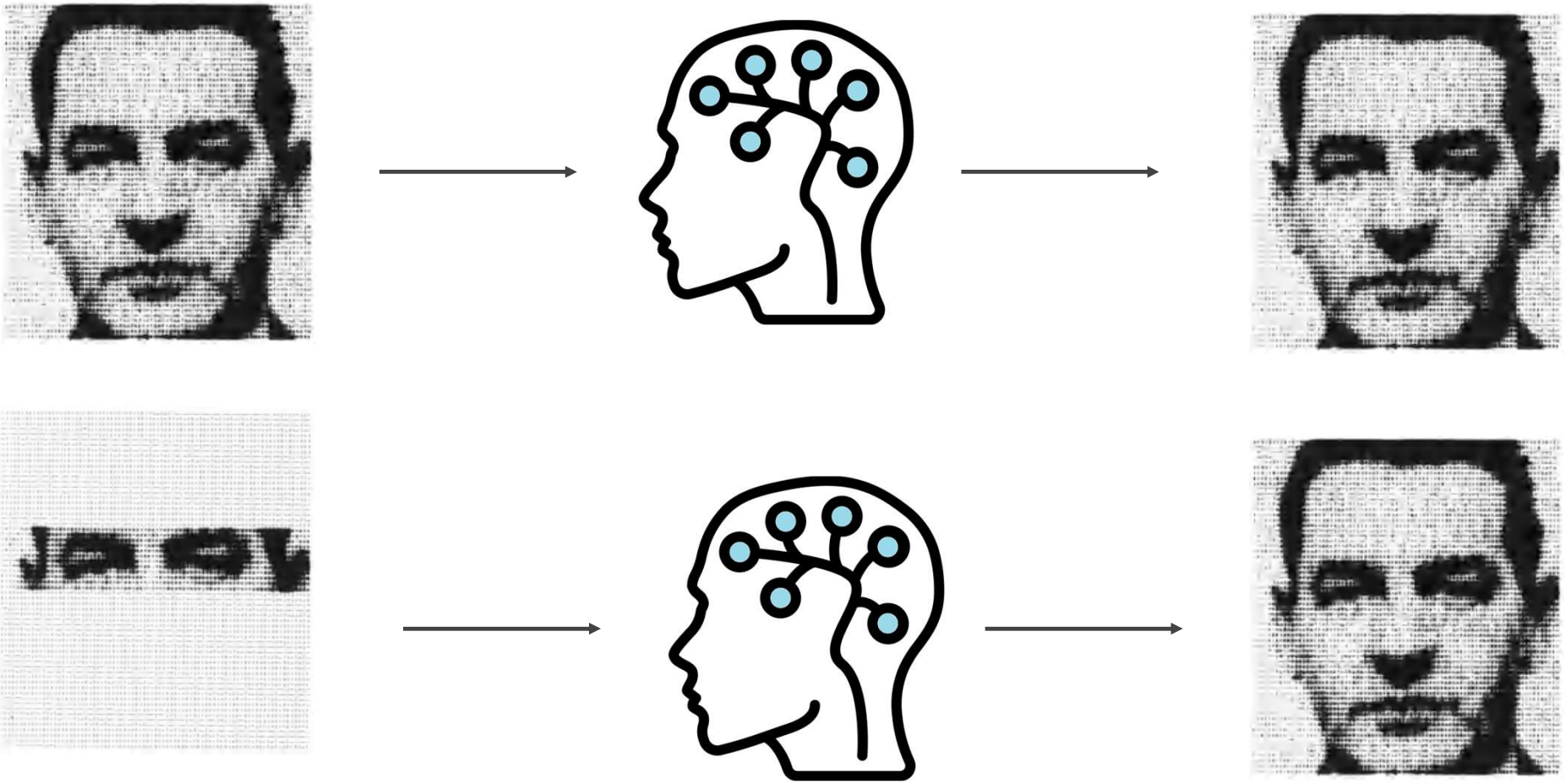
We can provide the memory with noisy cues



And the memory will complete the information



Just as suggested in the literature



Outline

- Background
- Analysis of the Retrieval of Visual Patterns
- Multi-Modal Willshaw Network
- Conclusion

The idea of MMWN emerges from the following facts:

Literature review

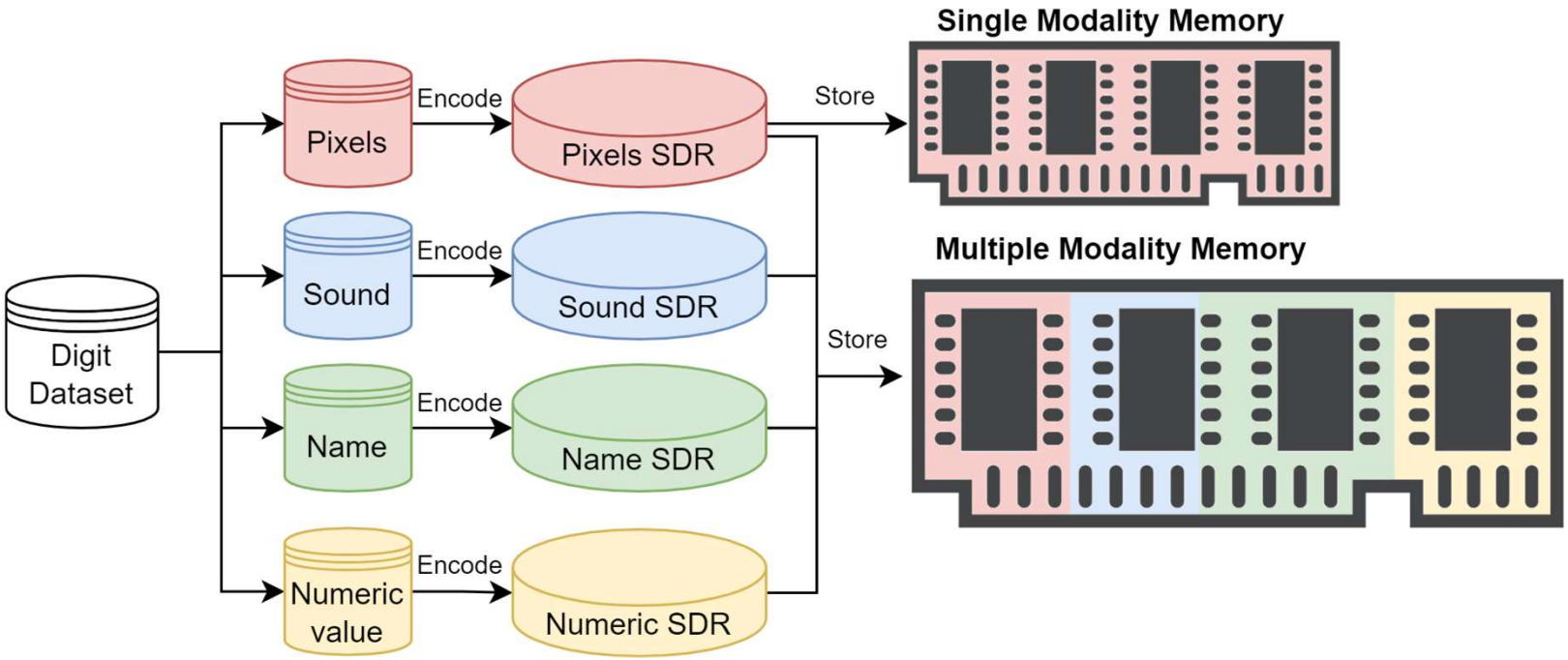
- *Active bits in an SDR simply represent the **presence of a feature**.*
- *SDRs are **compressions***

WN analysis

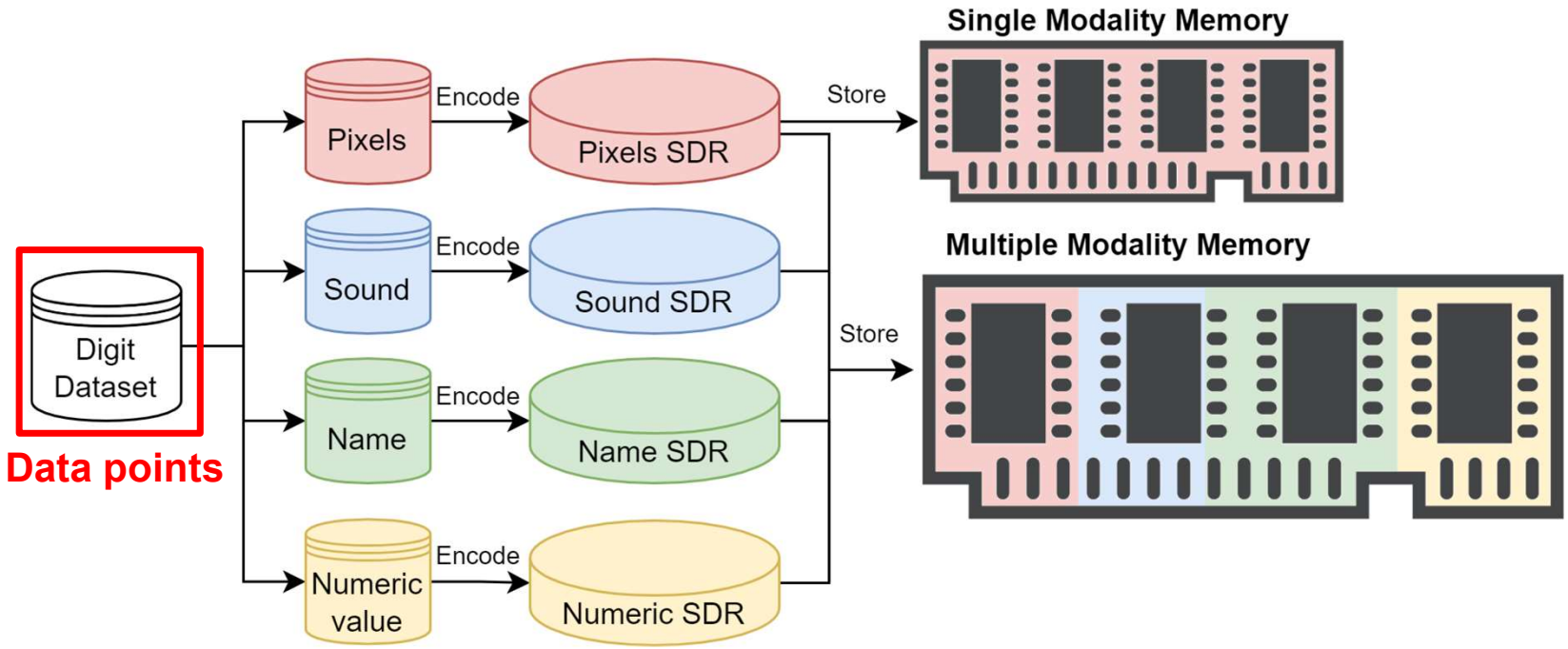
- *Memory makes **no assumption about the input.***
- *It simply interprets **each active bit** an active **feature**.*
- *Each bit is **equally important**.*

A new, simple idea

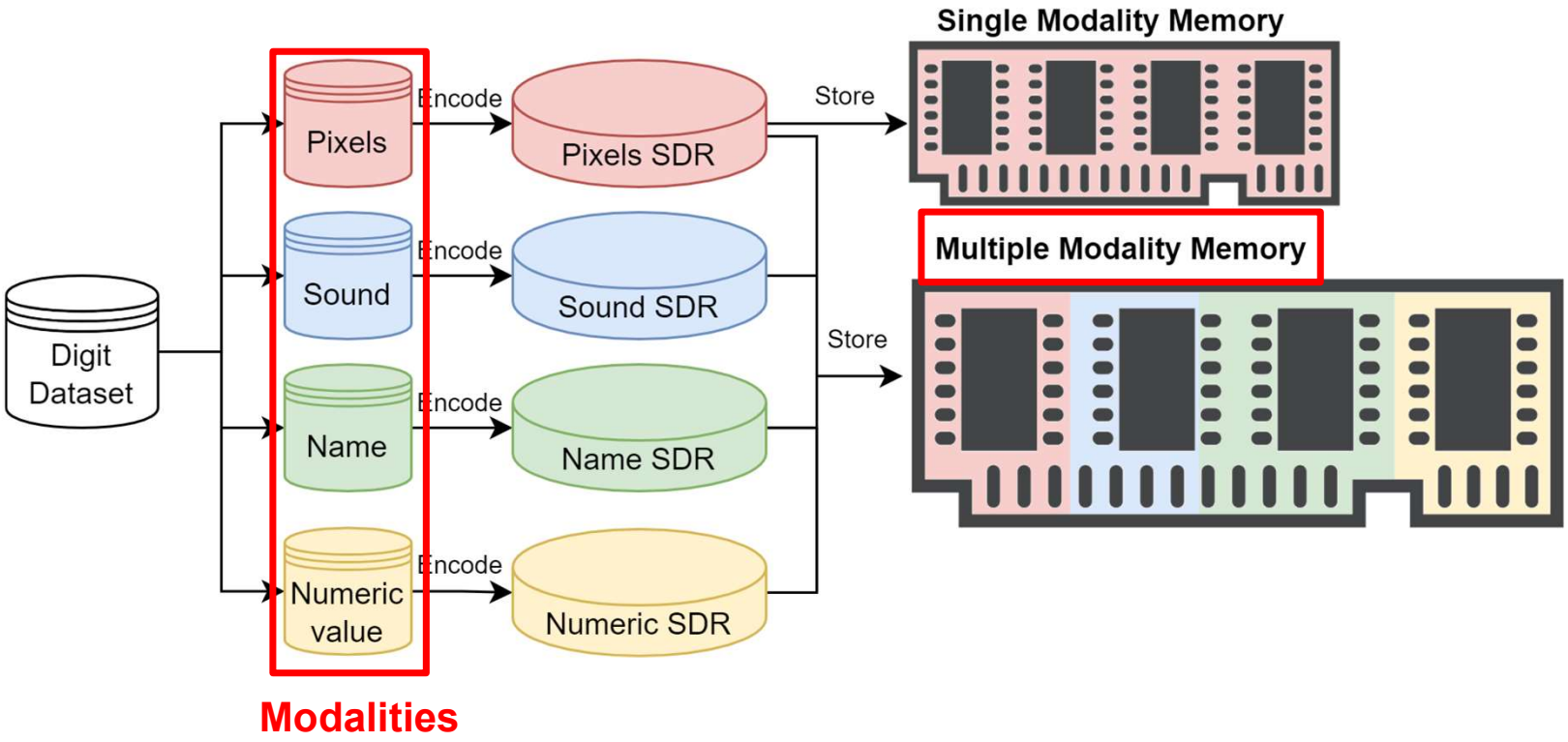
The Multi-Modal Associative memory



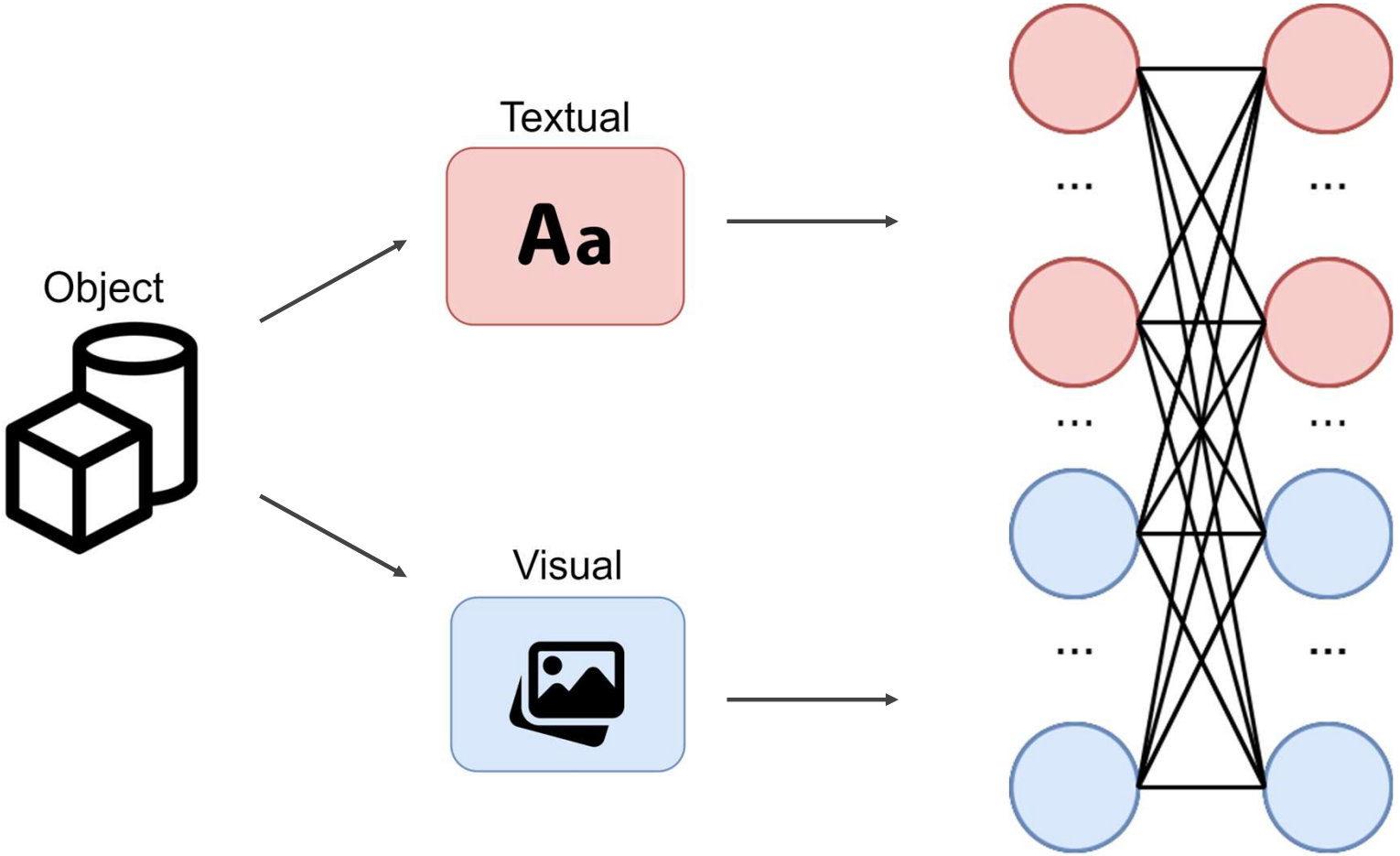
For each pattern



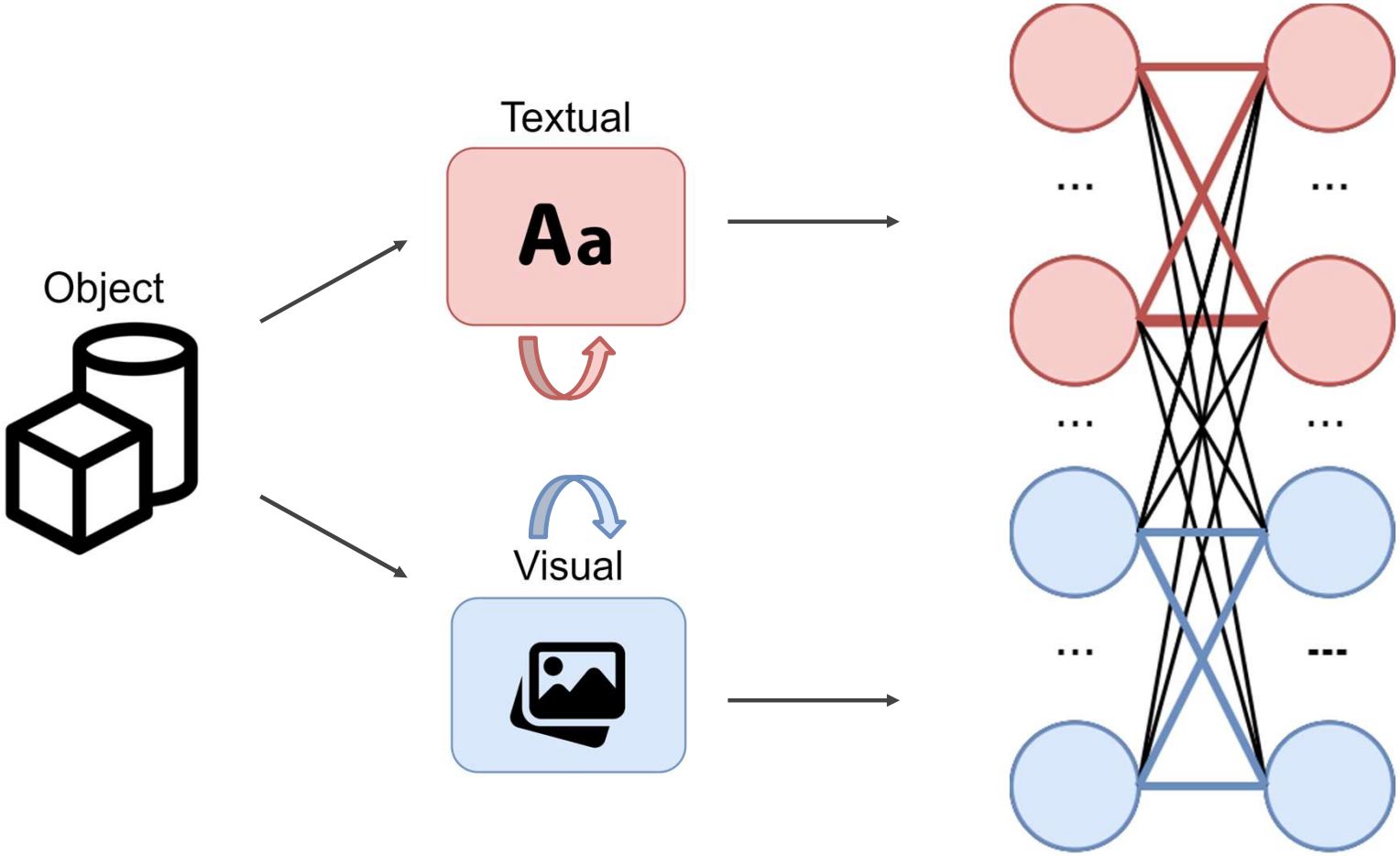
We can store multiple types of information simultaneously



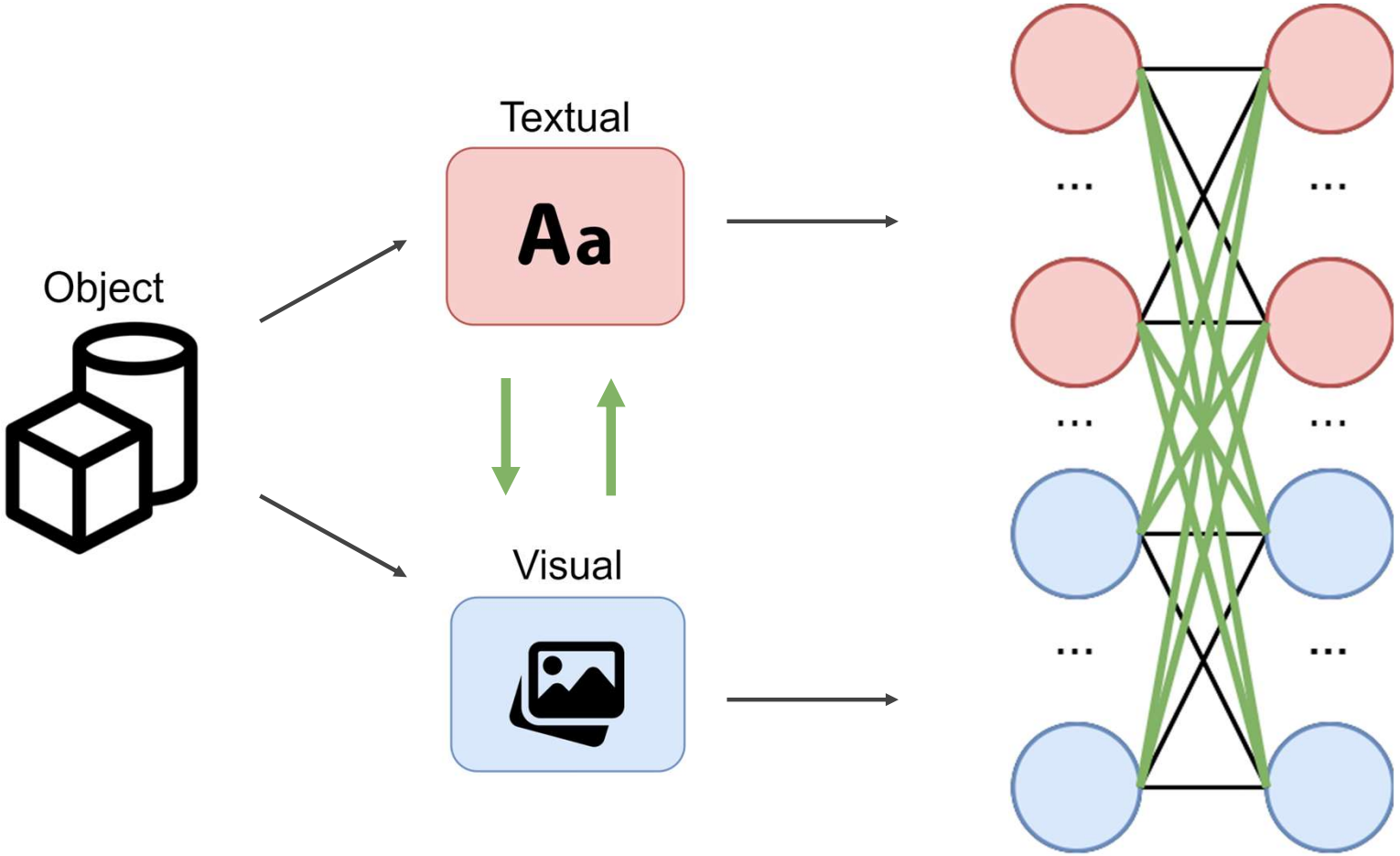
An example with two modalities



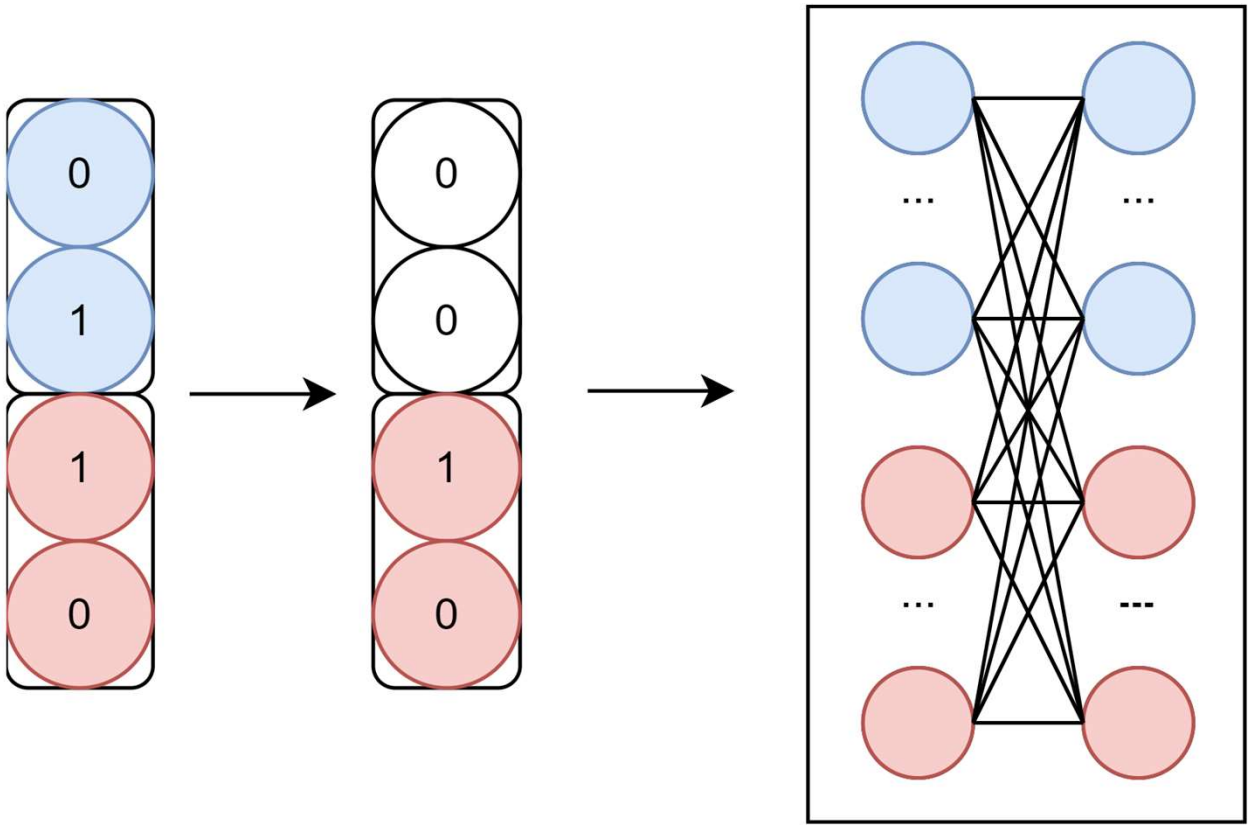
The memory will learn how to associate each modality



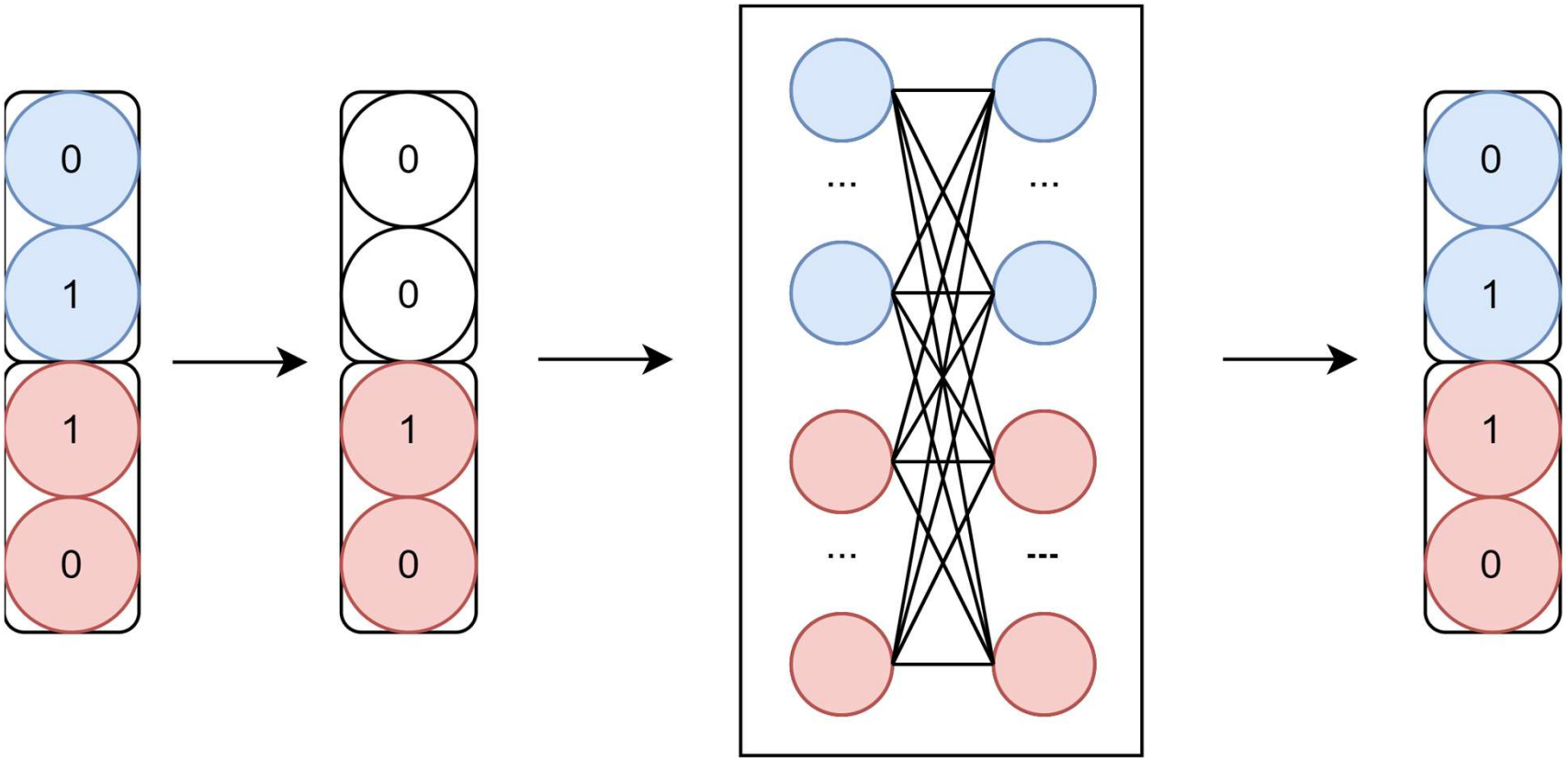
But also how to associate modalities with each other



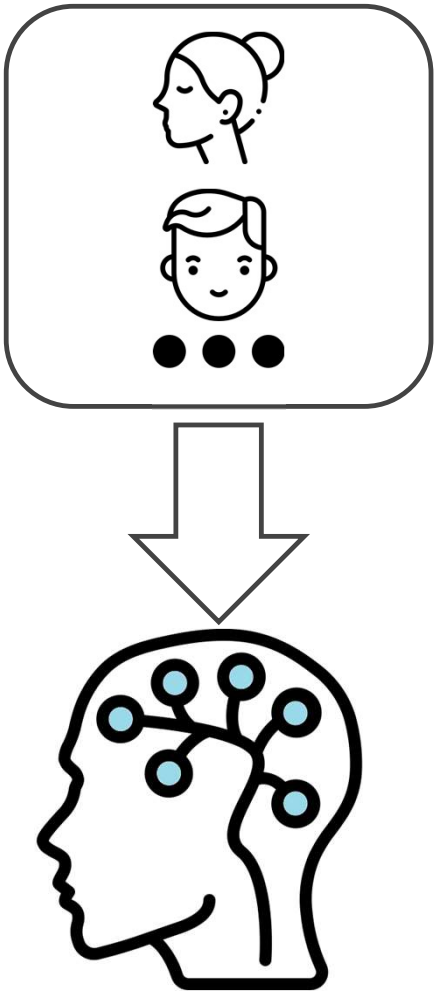
We can then leverage the COMPLETION capabilities



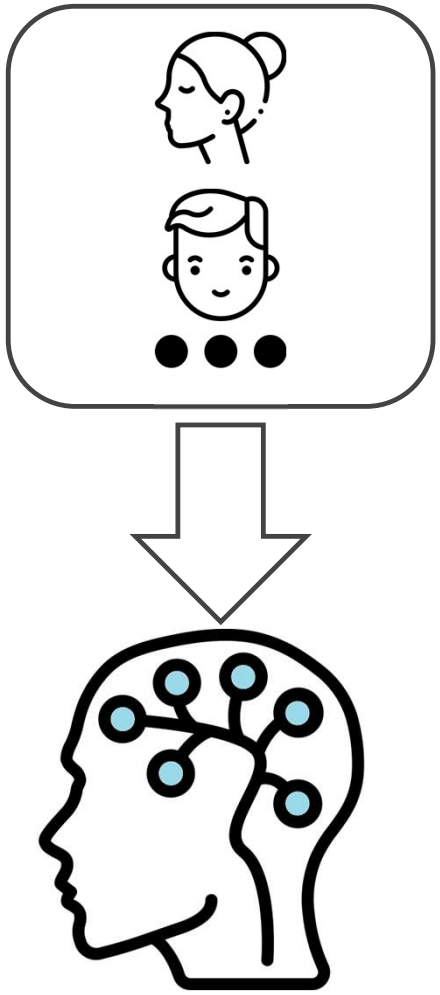
To do a variety of tasks



For instance if we store information about friends



And we use two modalities

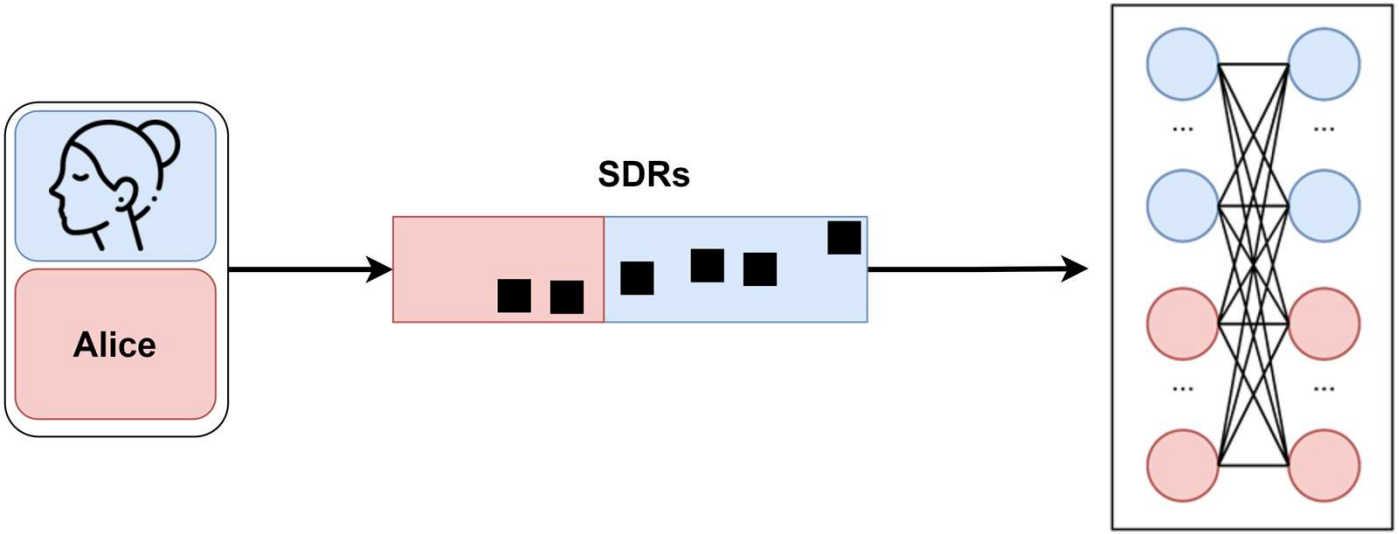


Aa

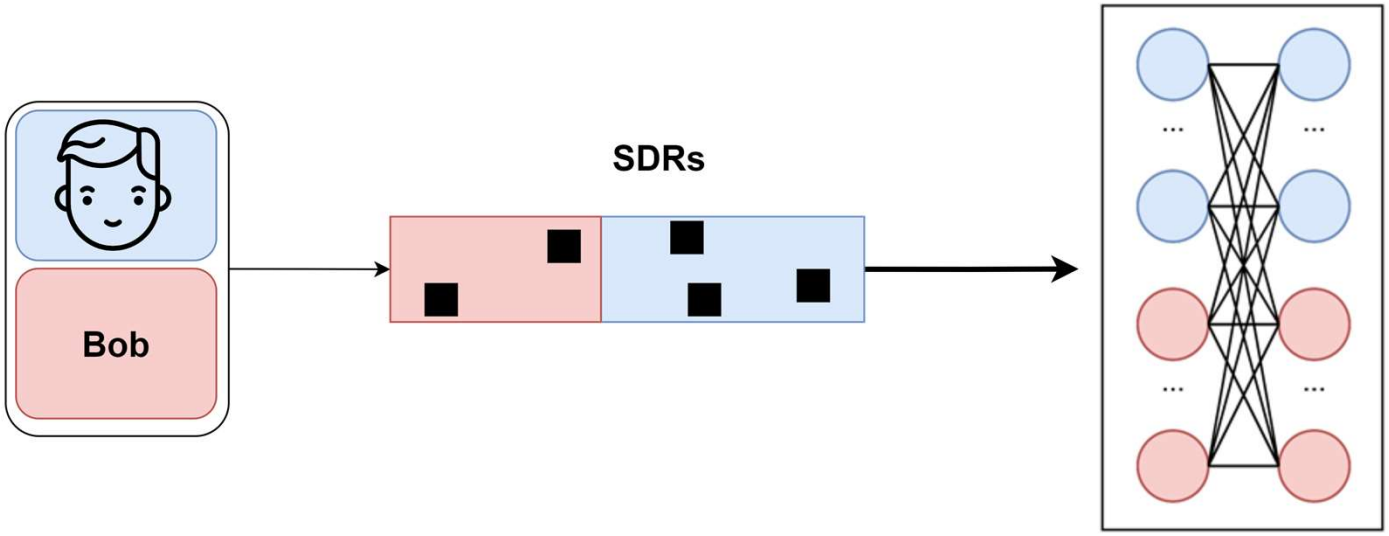
Textual : Name

Visual: a picture of them

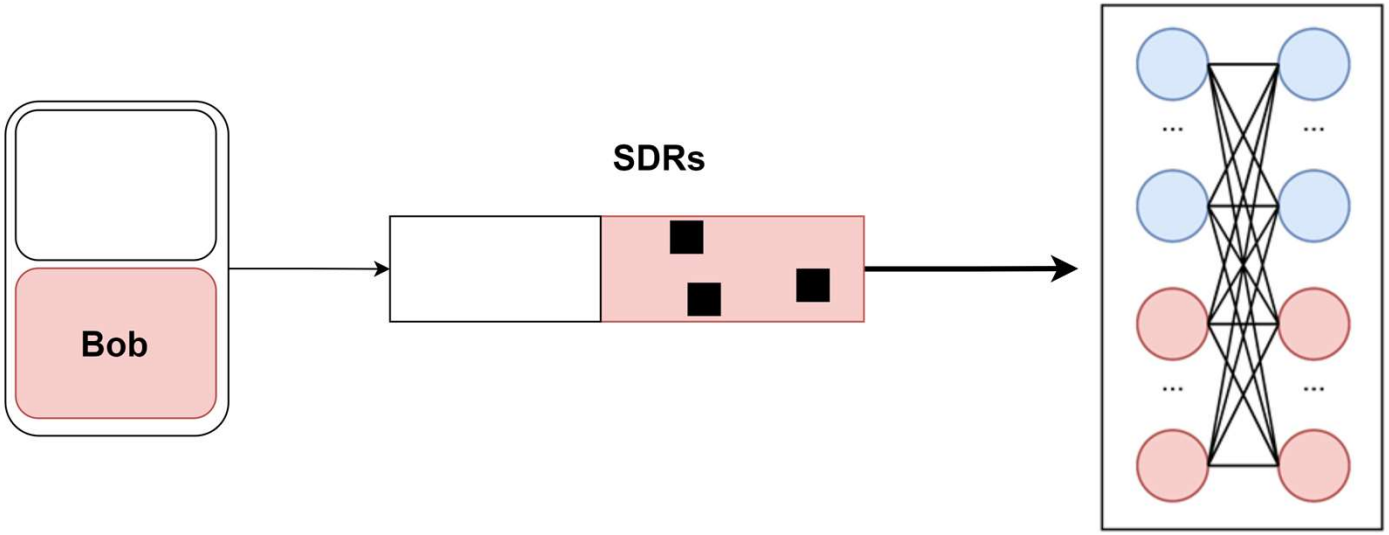
We create SDRs of both this modalities, and train a memory



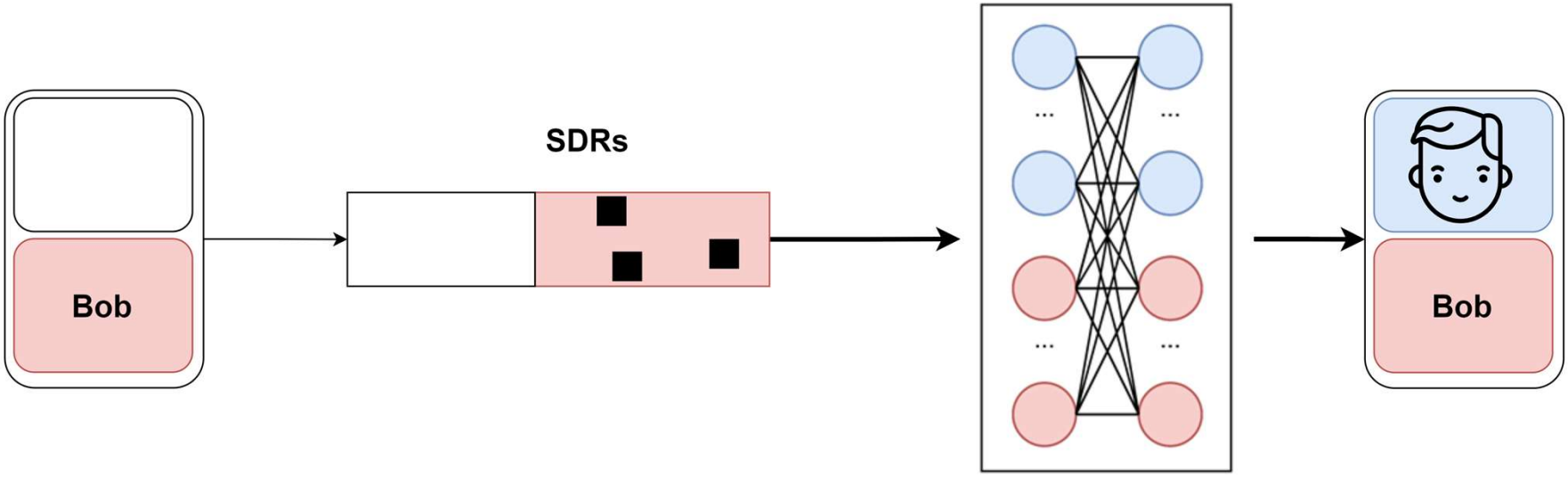
For several people...



We can give the memory a name...

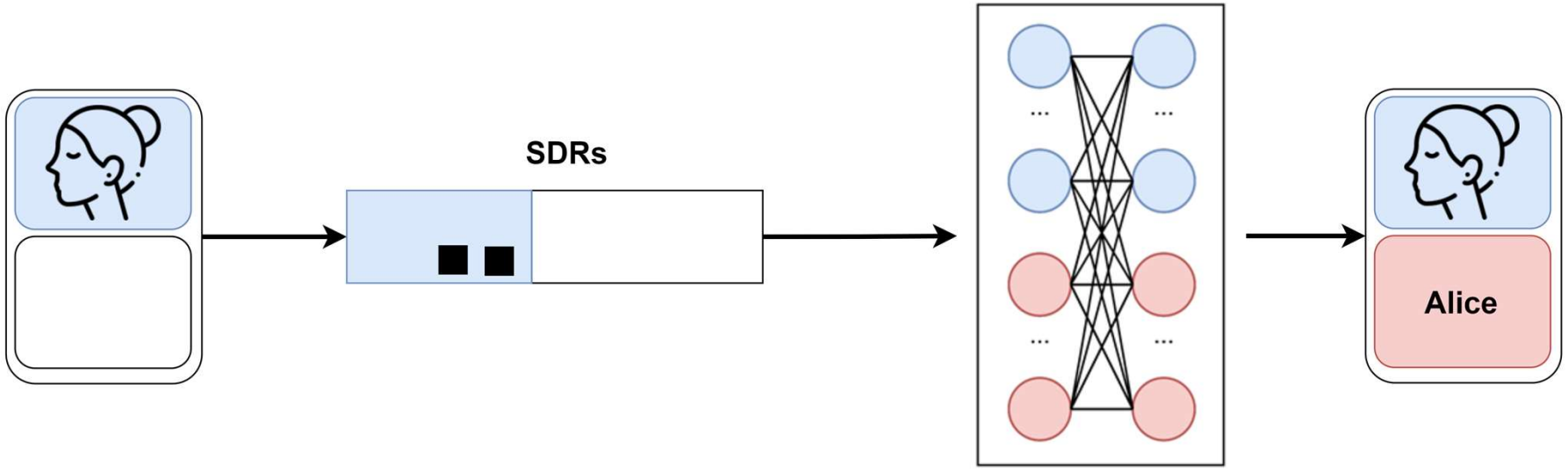


And the picture will be completed



Or we can delete the name...

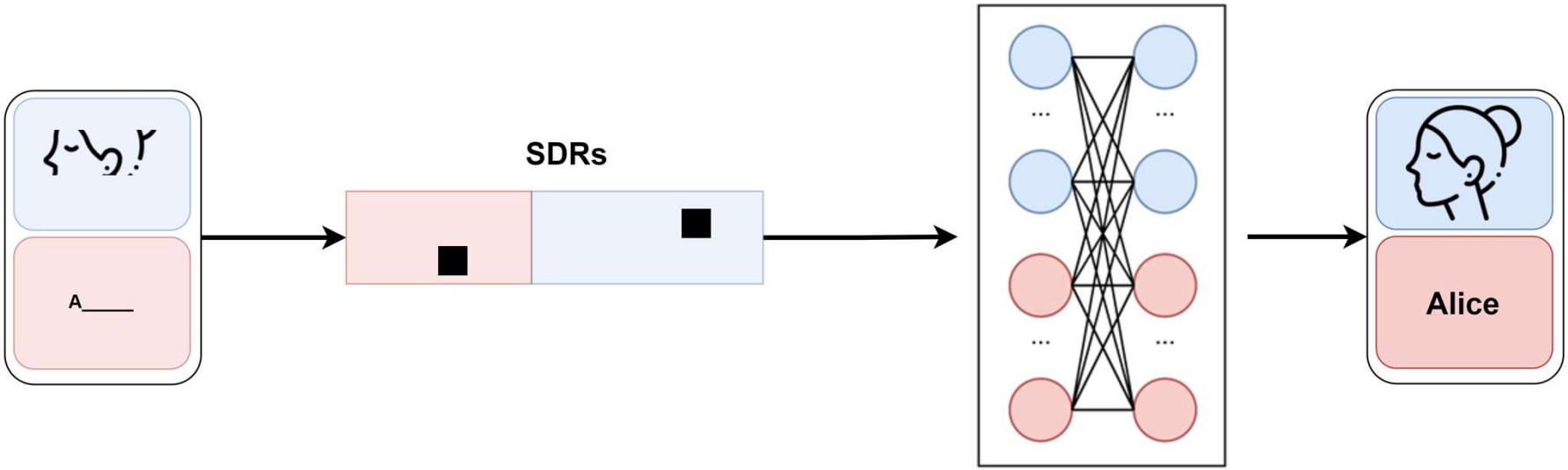
And use the memory to get it back



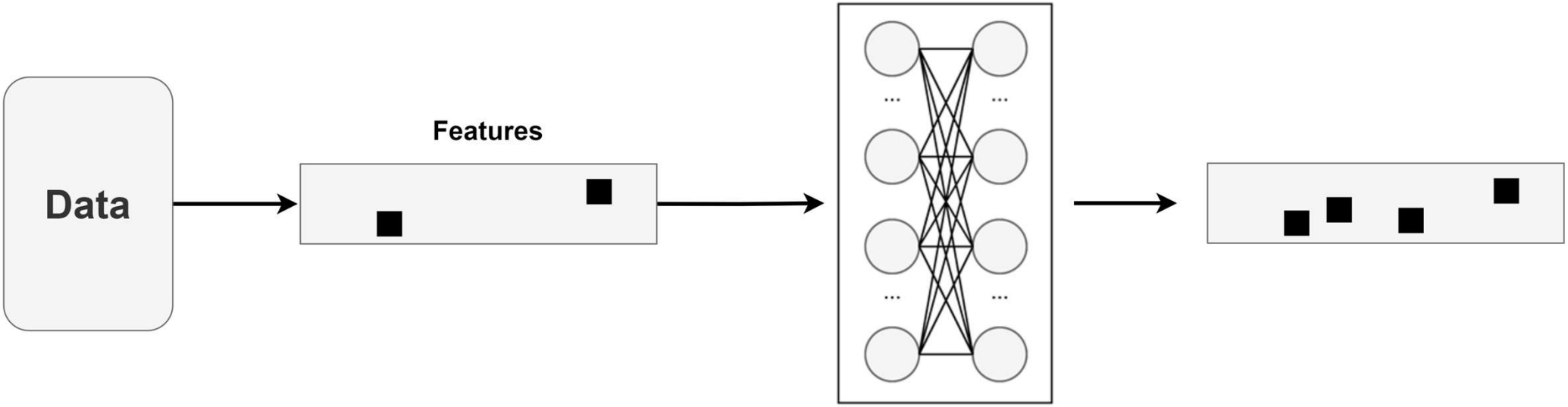
We could even do a mix:

- Long hair
- Name starts with “A”

We could even do a mix:



The modalities are simply an ABSTRACTION



The memory simply COMPLETES the missing information

Experimental Methodology

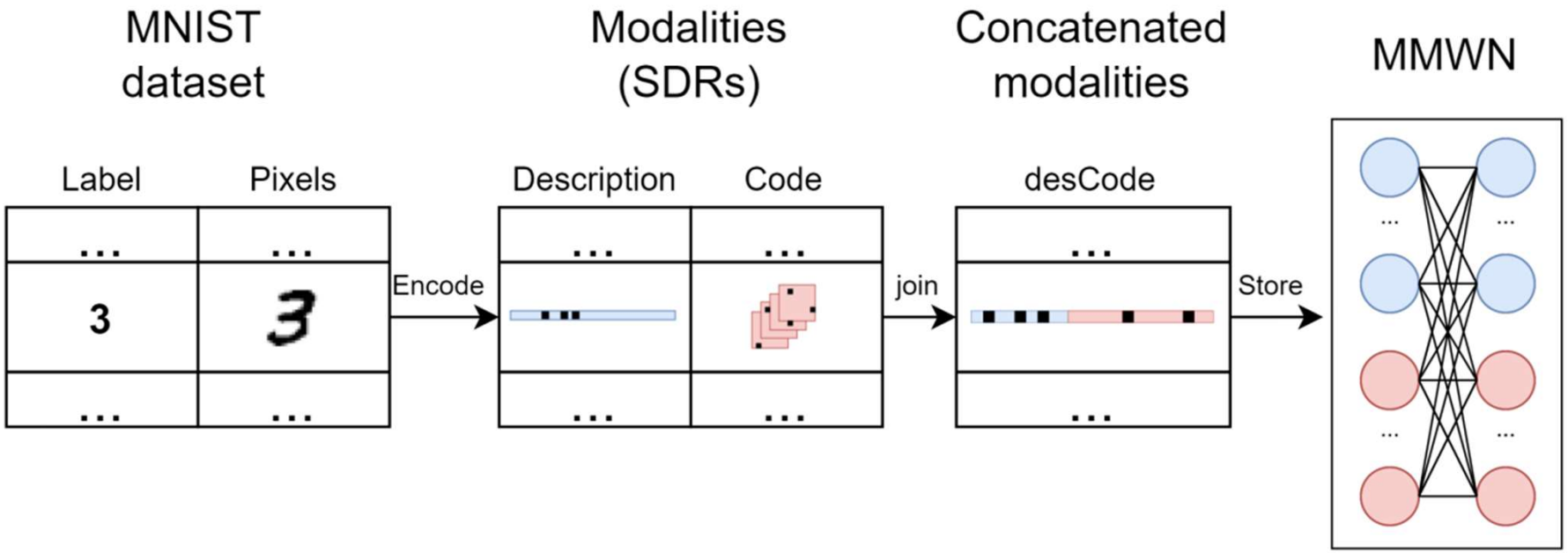
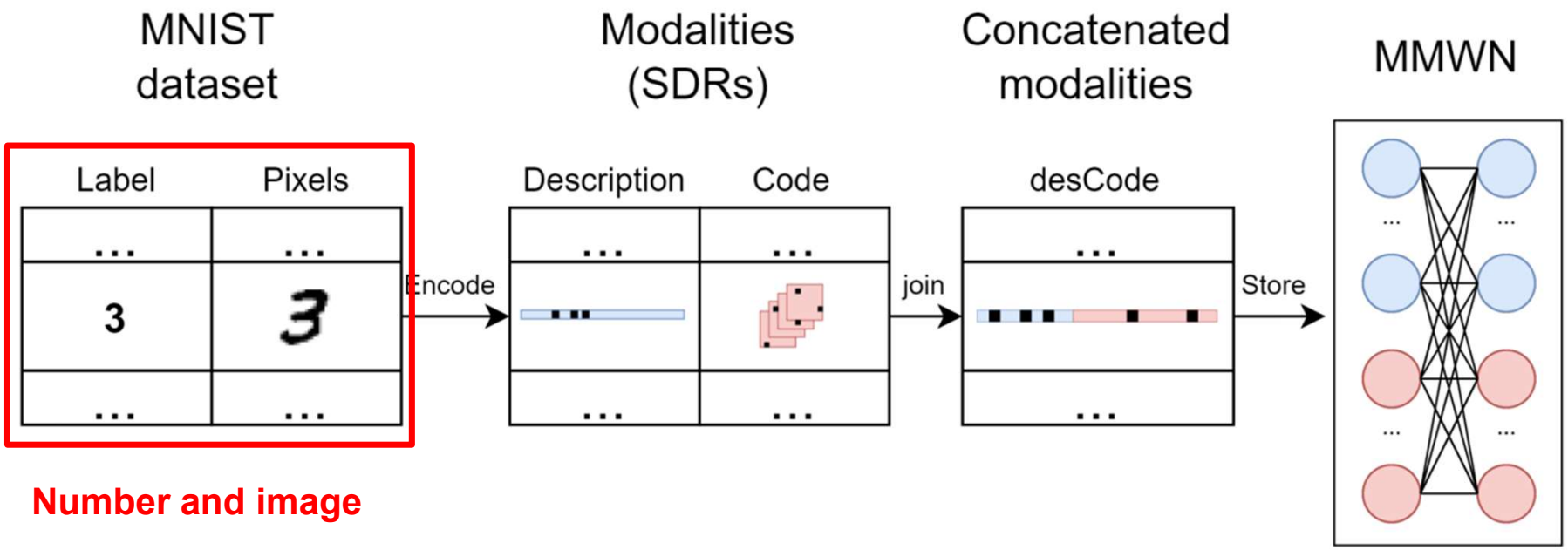


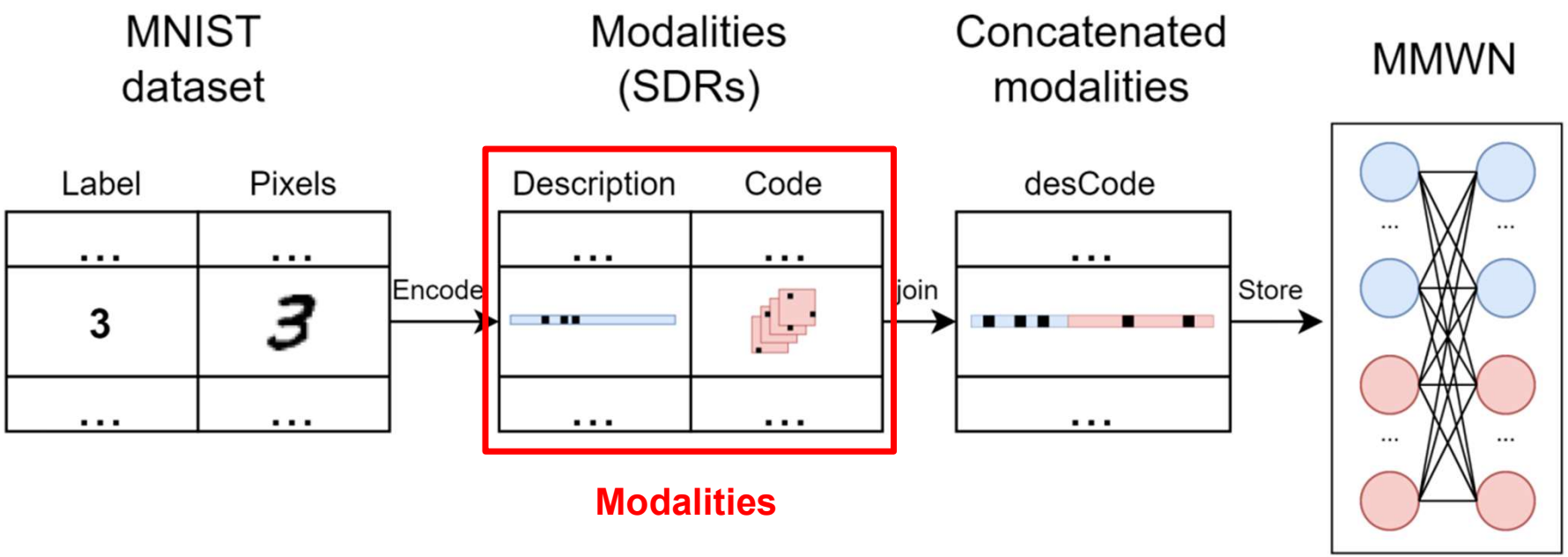
Figure 5.3: Training Step in a MMWN.

MNIST dataset

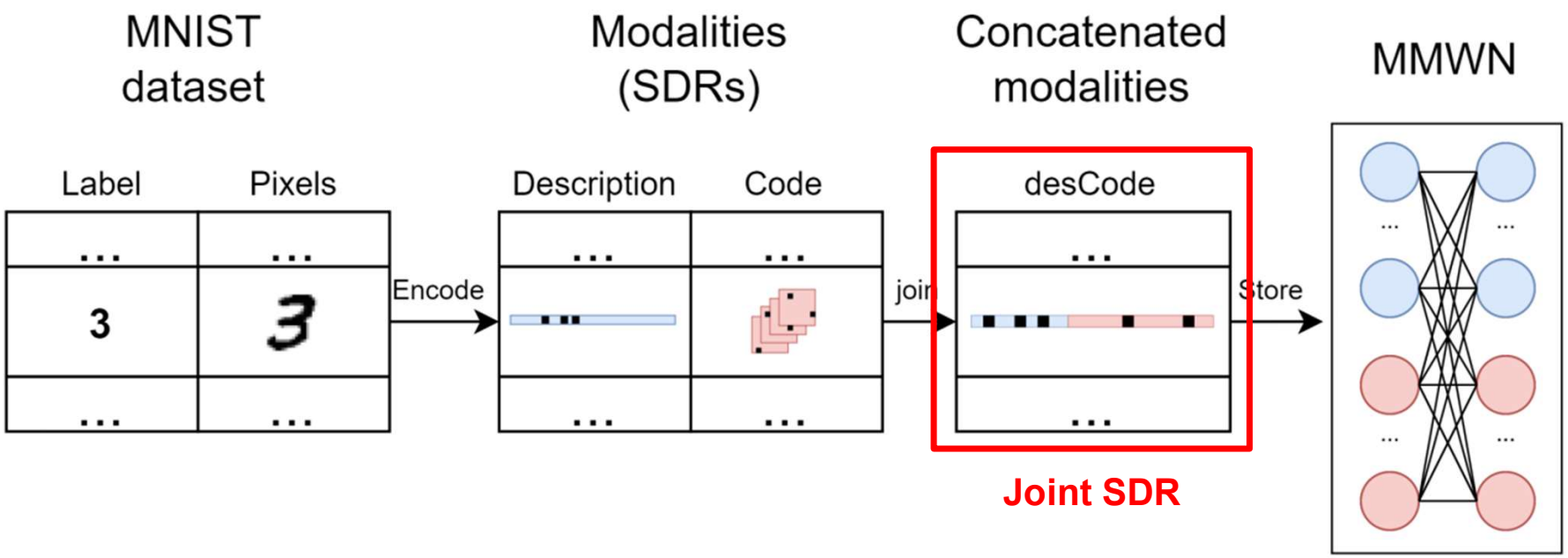


Number and image

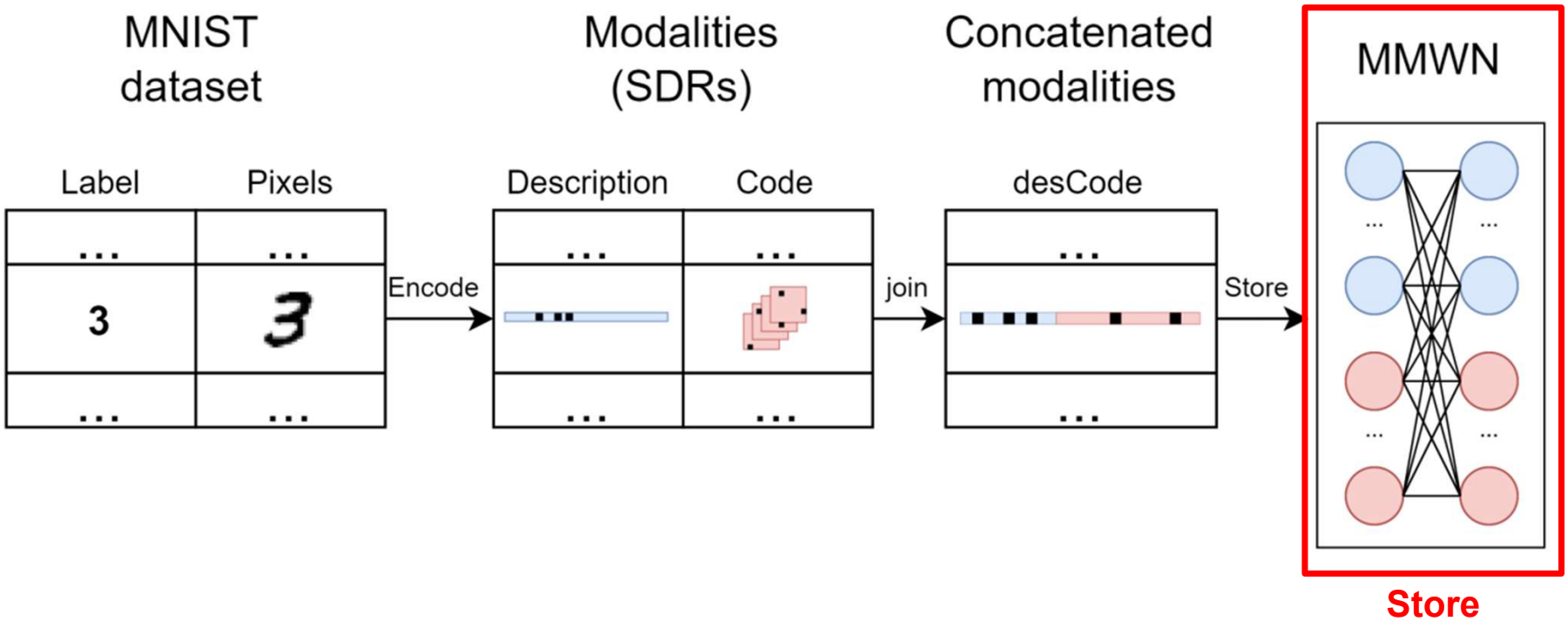
SDRs


MNIST dataset
Label
Pixels
3
Encode
Modalities (SDRs)
Description
Code
join
Concatenated modalities
desCode
Store
MMWN
Modalities

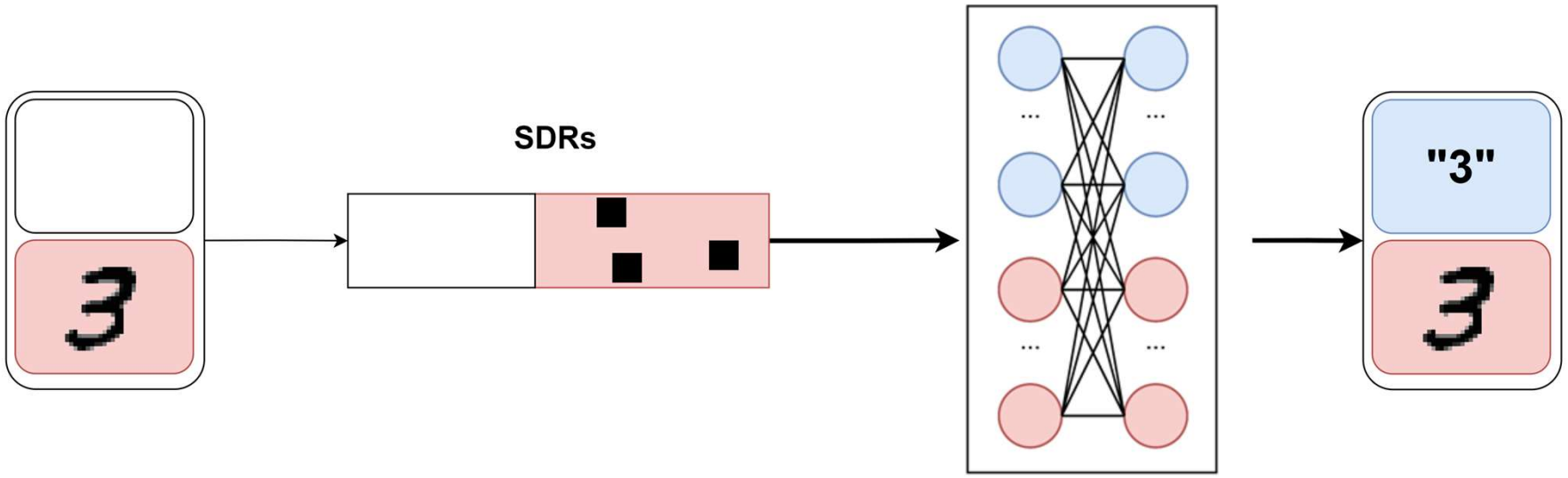
Join the two modalities



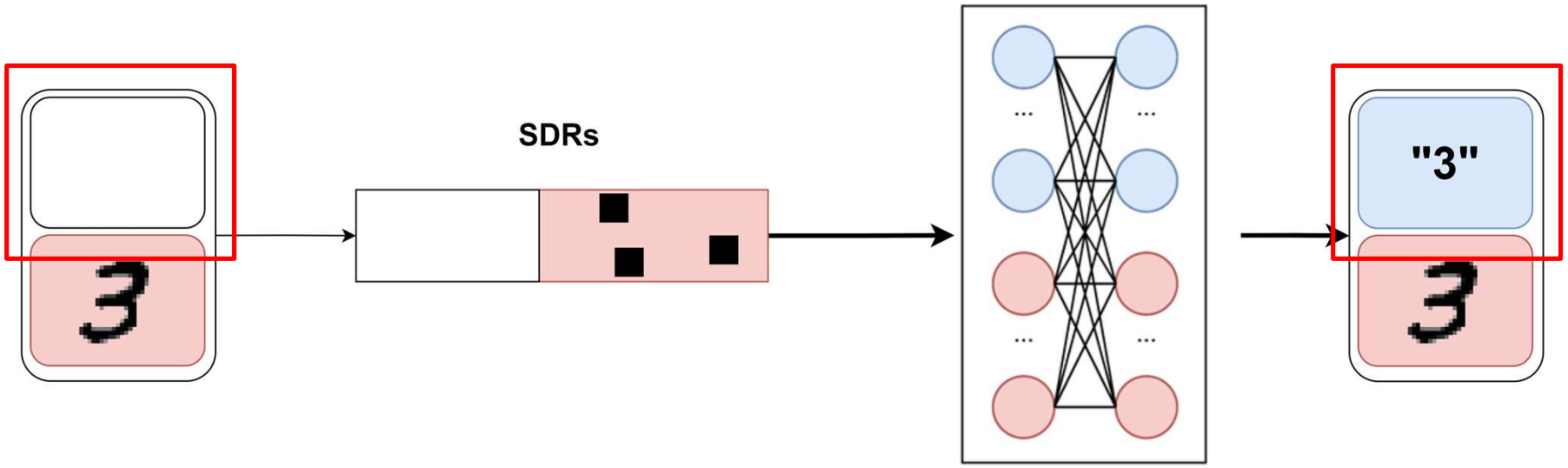
Store in the Multi-Modal Memory



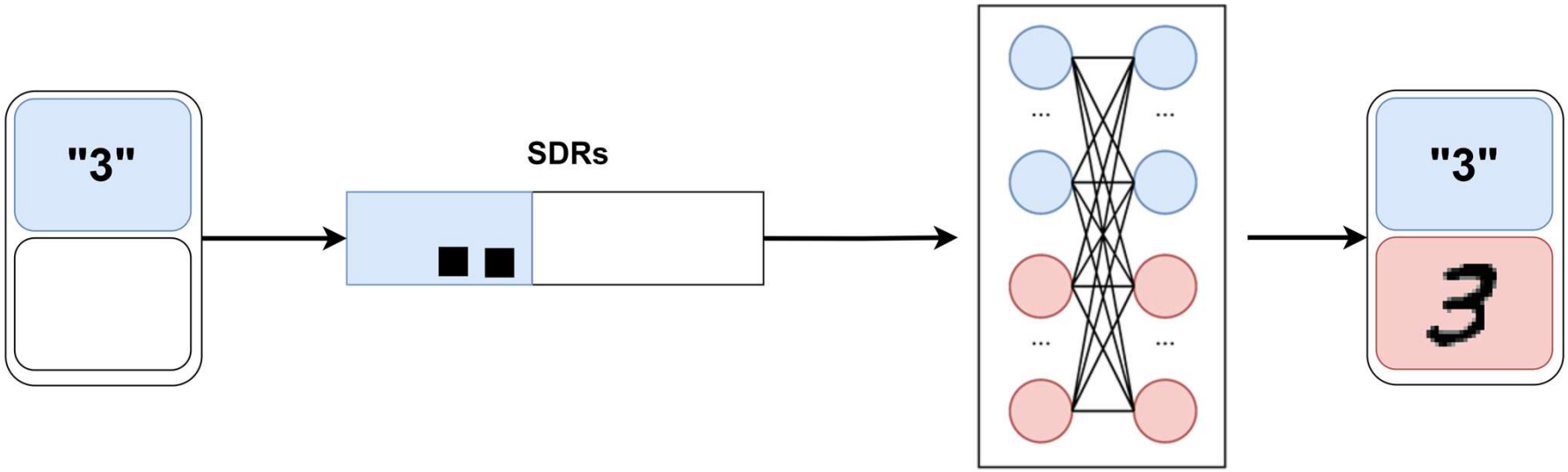
We can then: Delete the number



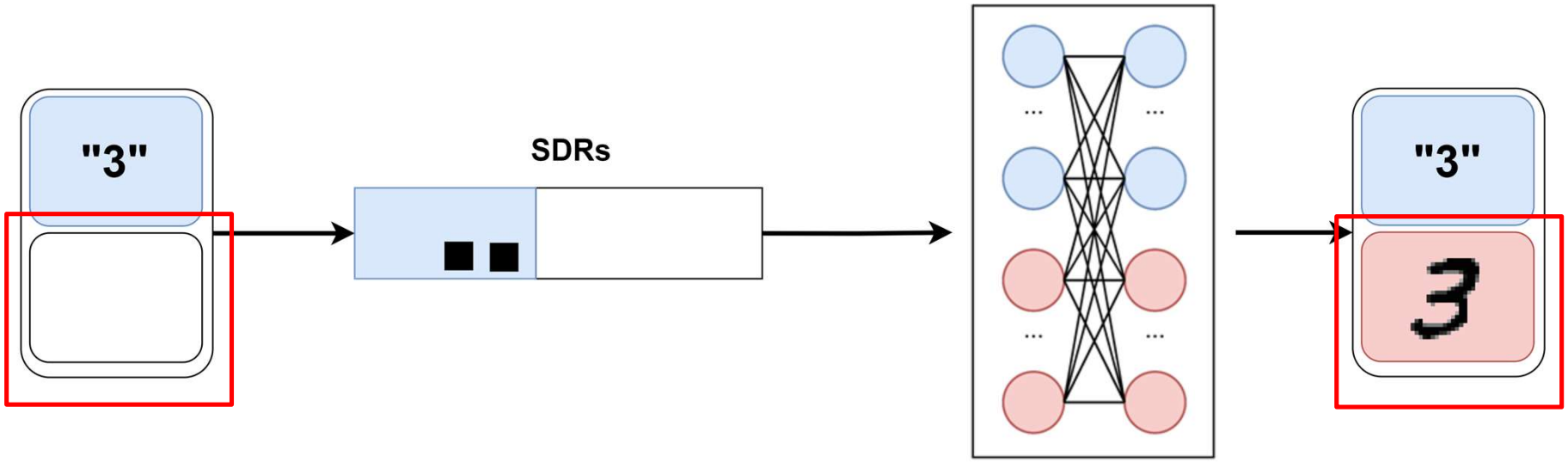
And the memory will complete it: Classification



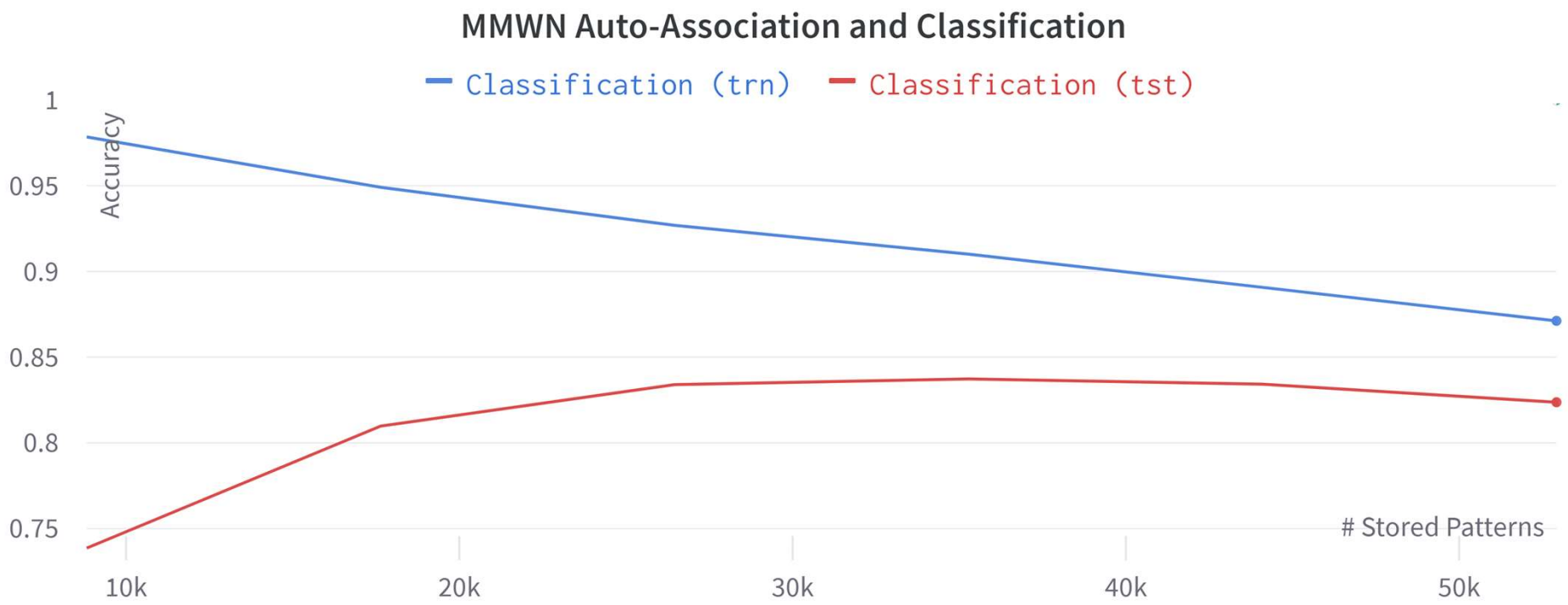
Or delete the visual modality:



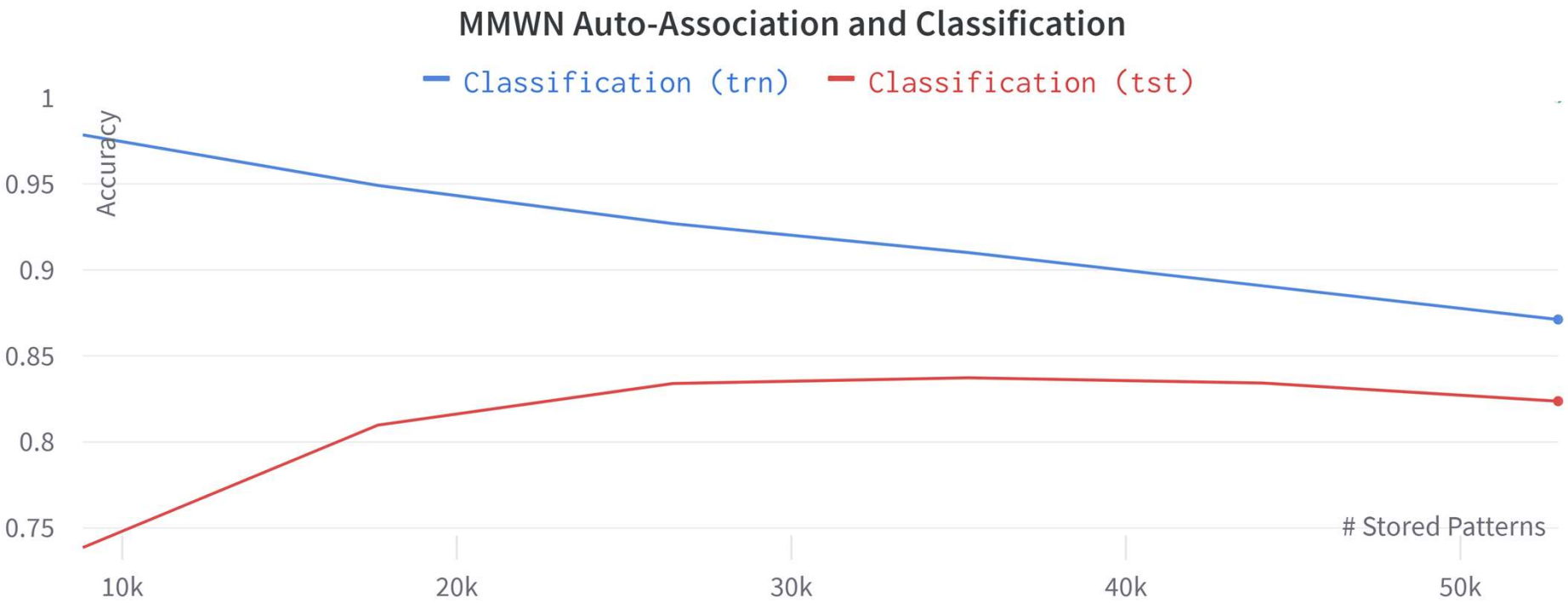
Forcing the memory to COMPLETE the image: Generation



Results: Classification

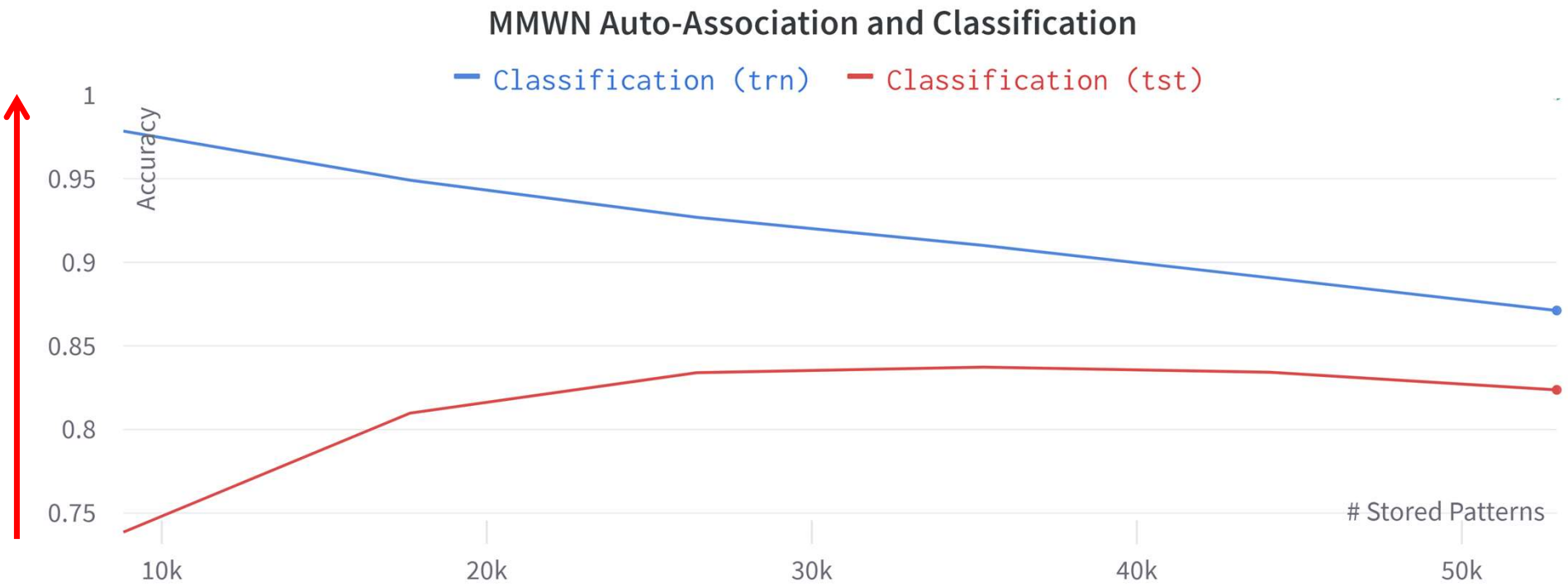


As we store more information in the memory



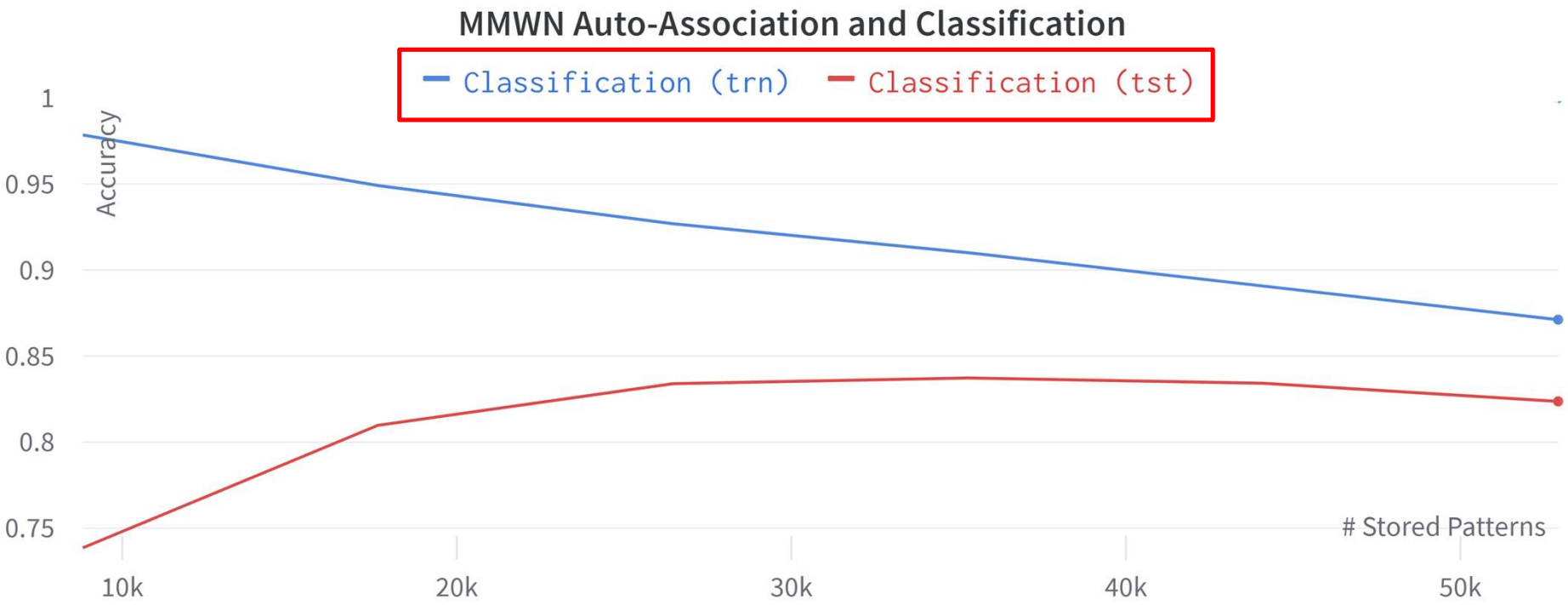
Training

We measure the classification accuracy

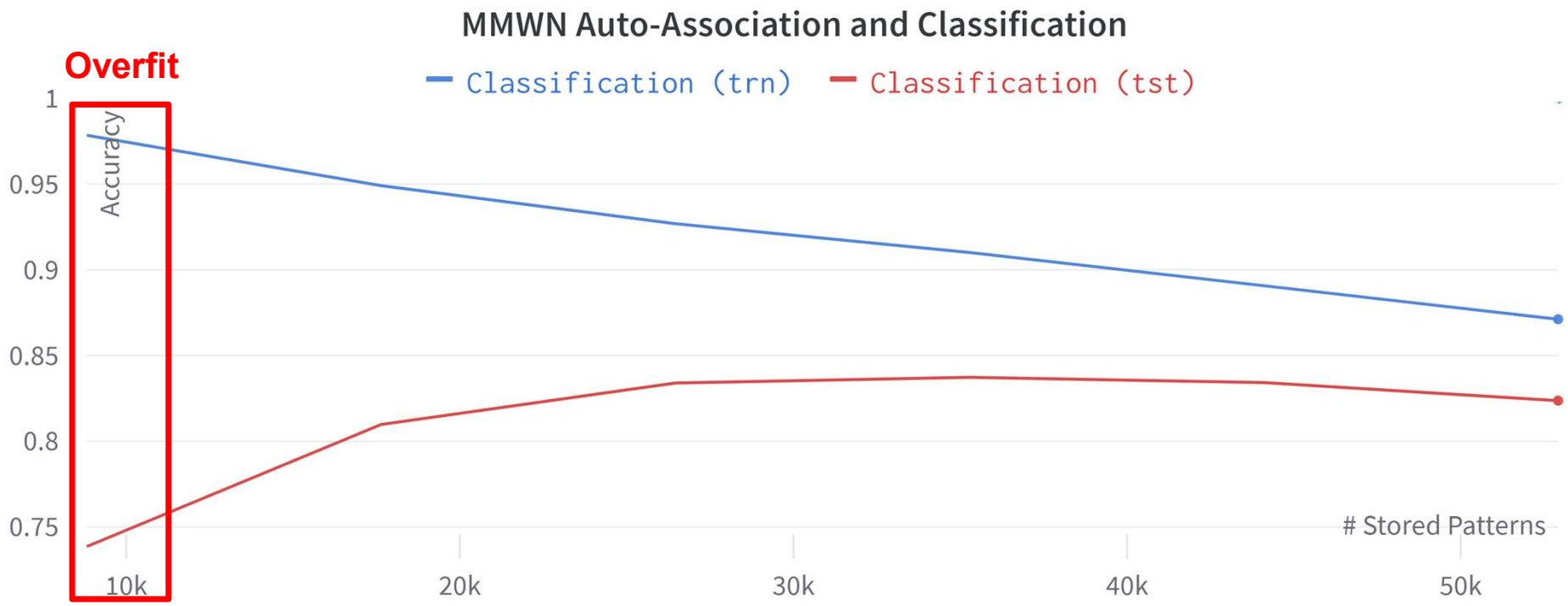


Accuracy

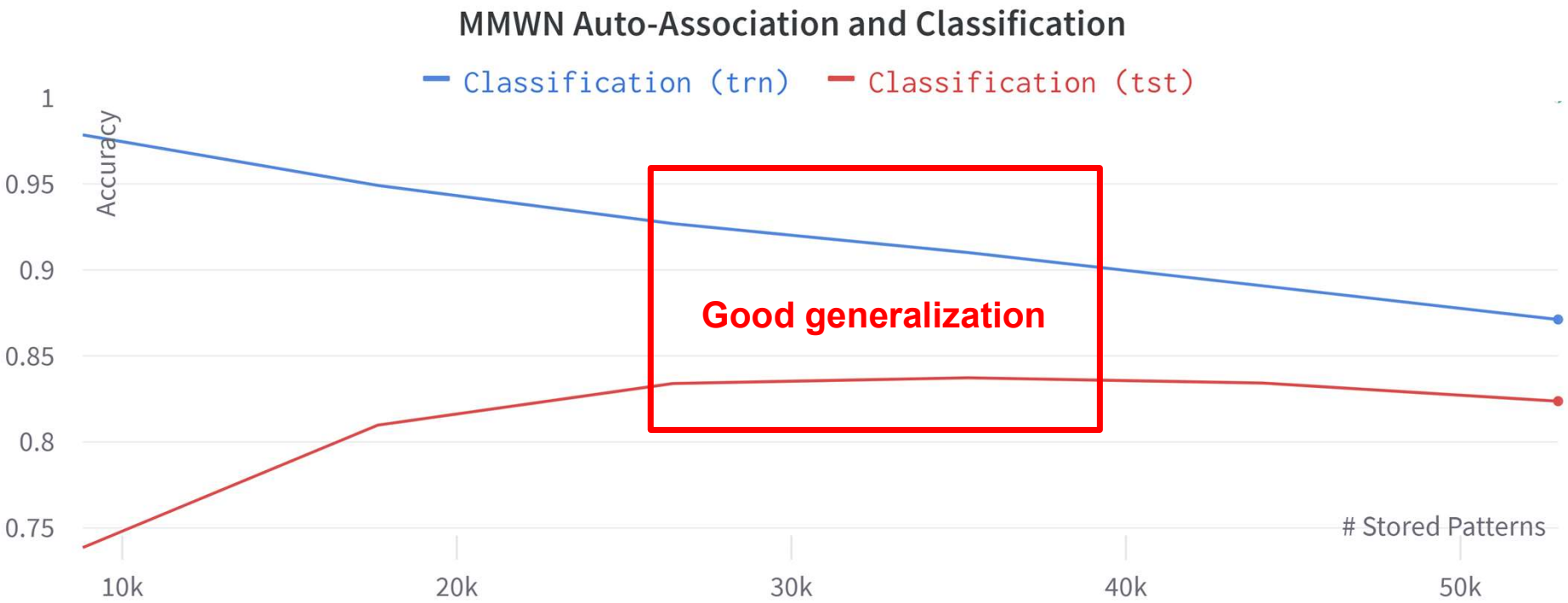
On known, and unknown data



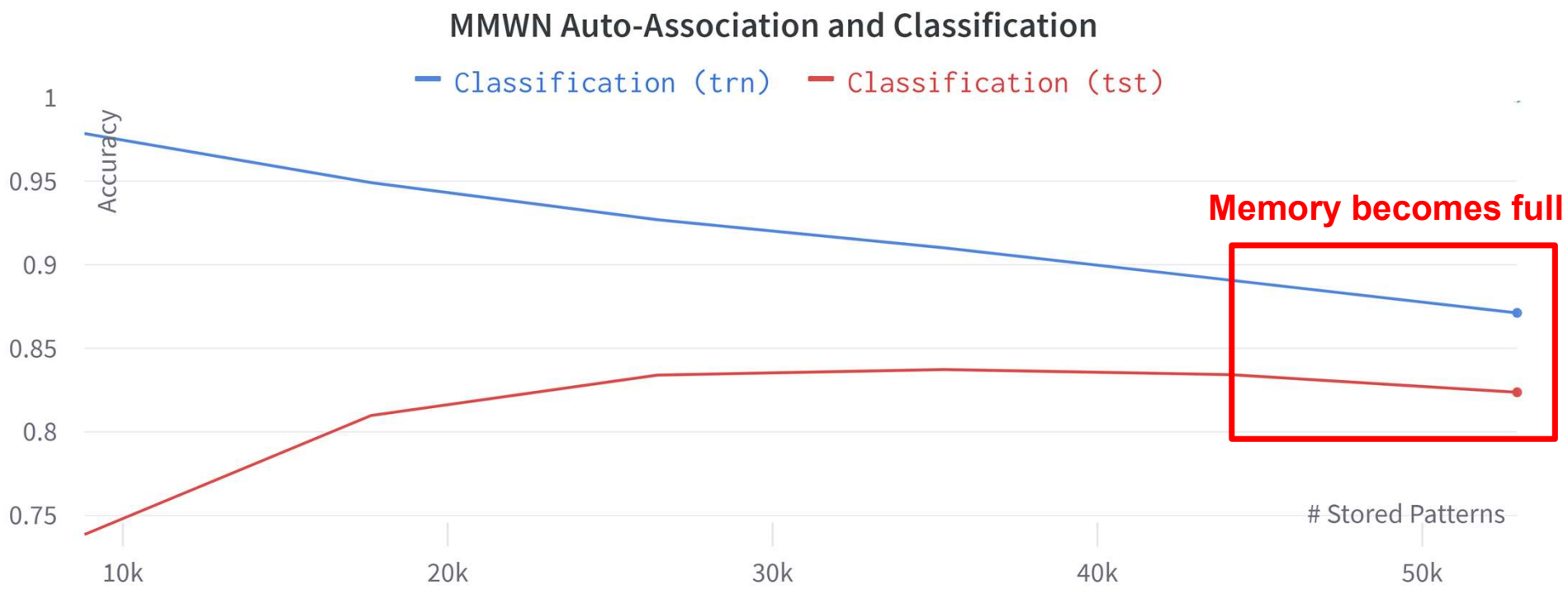
The memory memorizes with little information



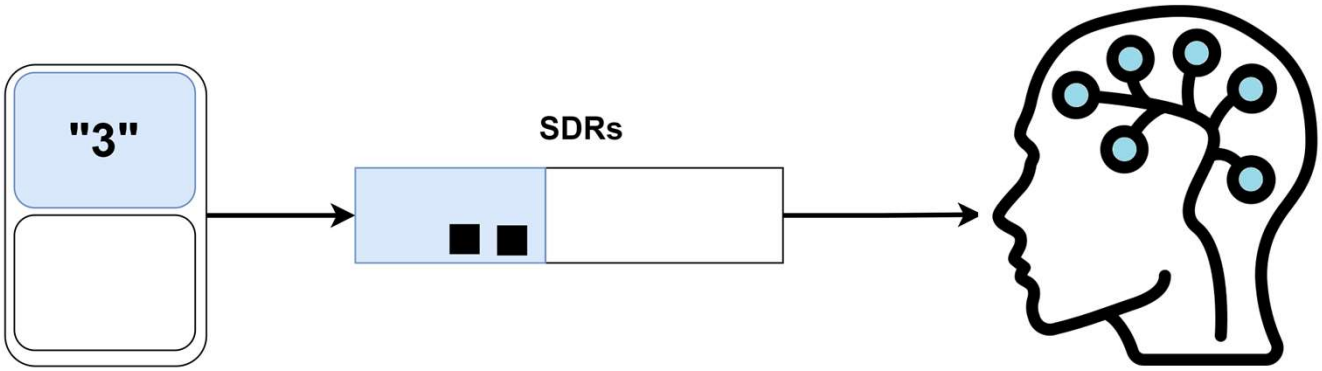
But learns to generalize as it trains



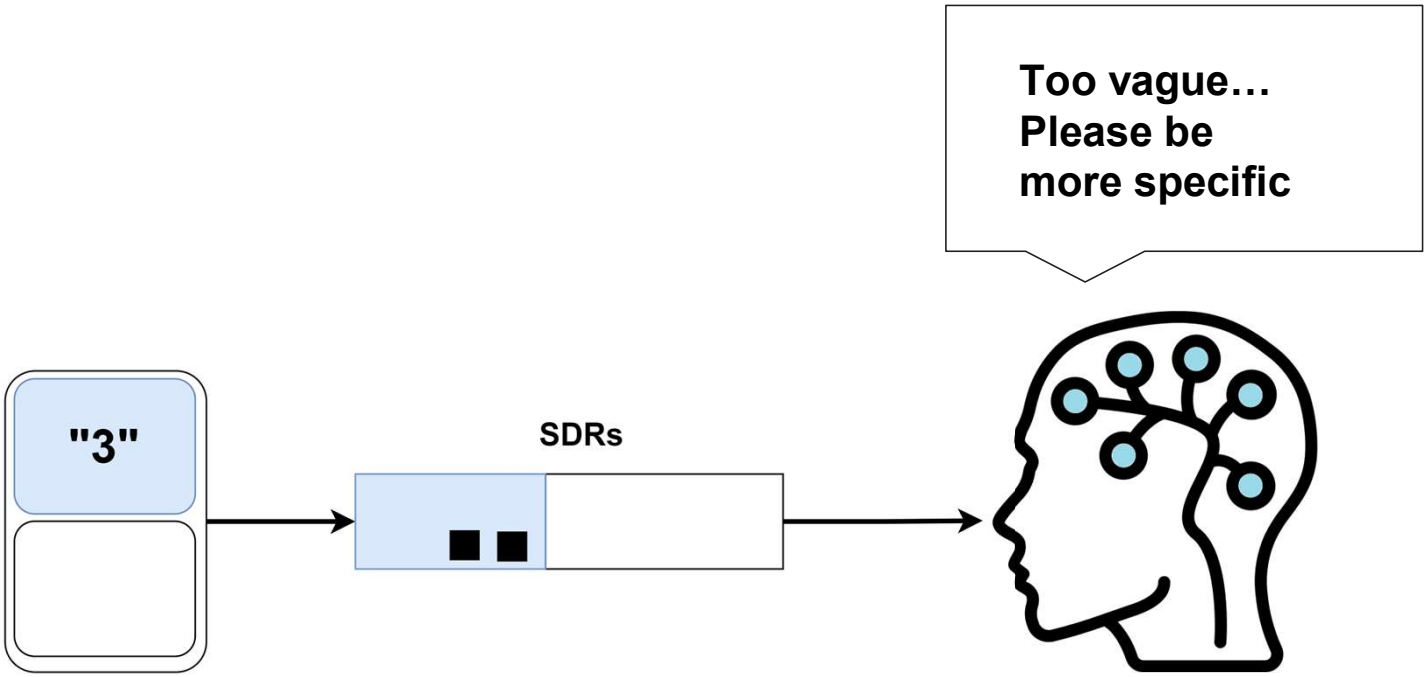
But toot much information is not ideal



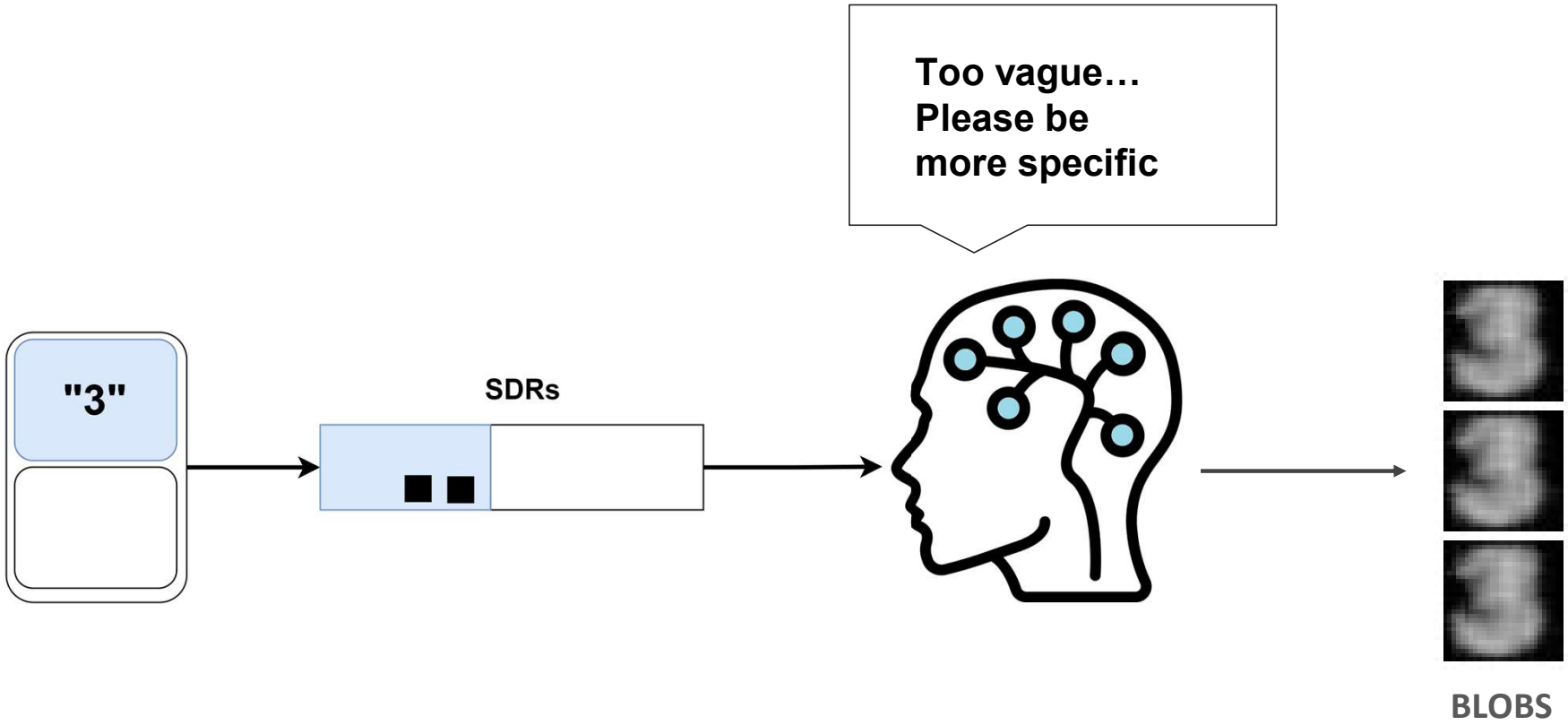
Generation is not so simple...



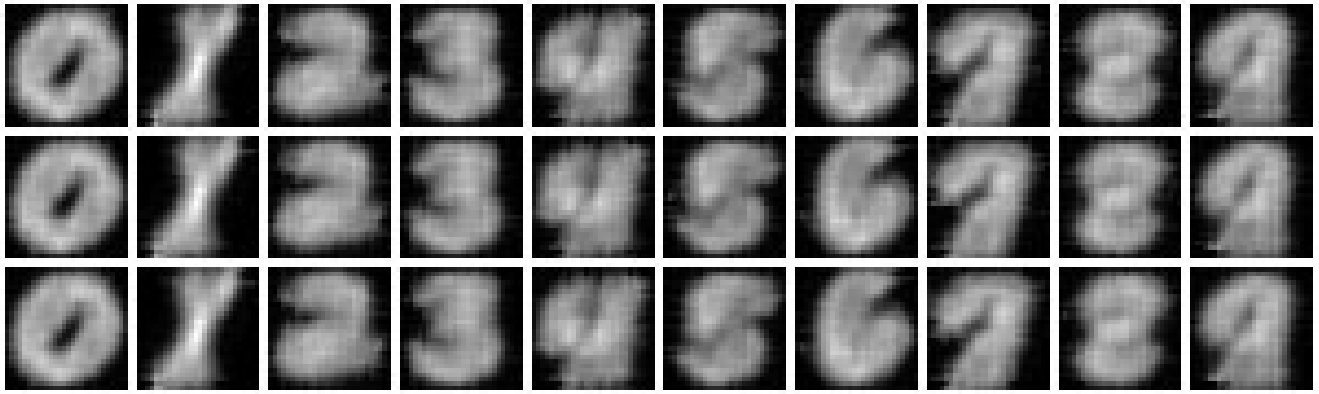
Just the number is not enough information



Leading to “vague” examples

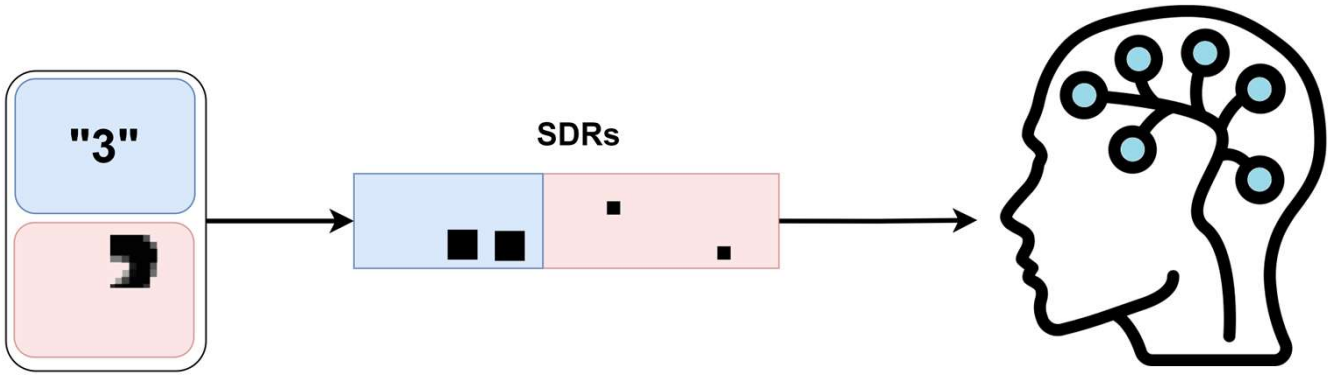


“Blobs” – prototype of the class

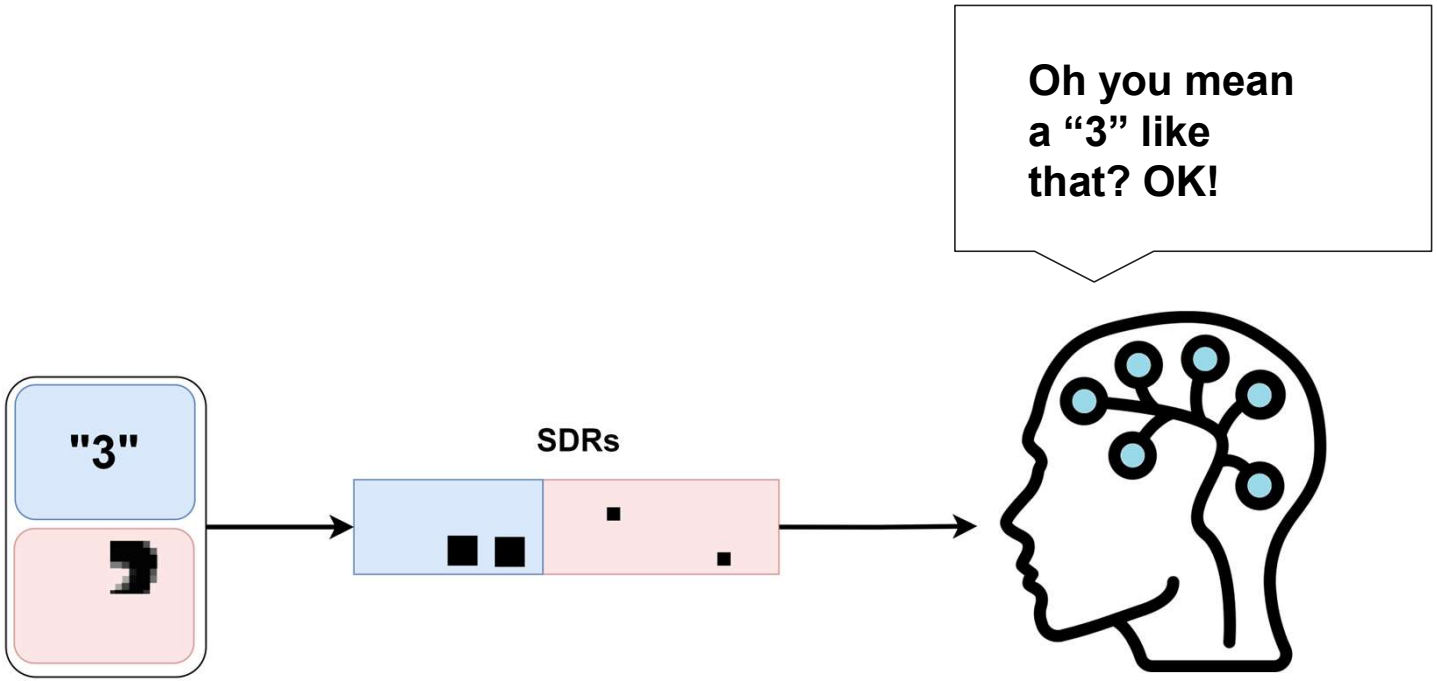


Like drawings from memory: without detail

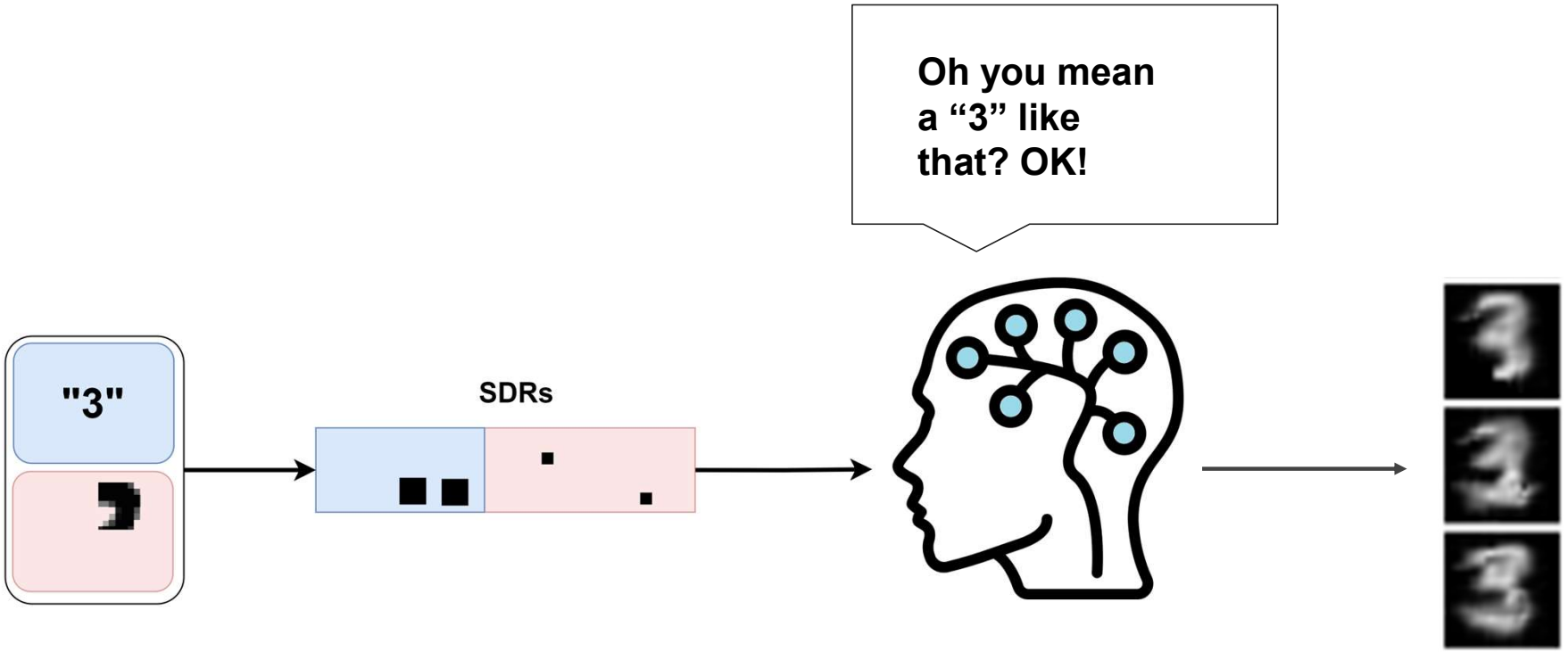
Solution: guide the memory with some information



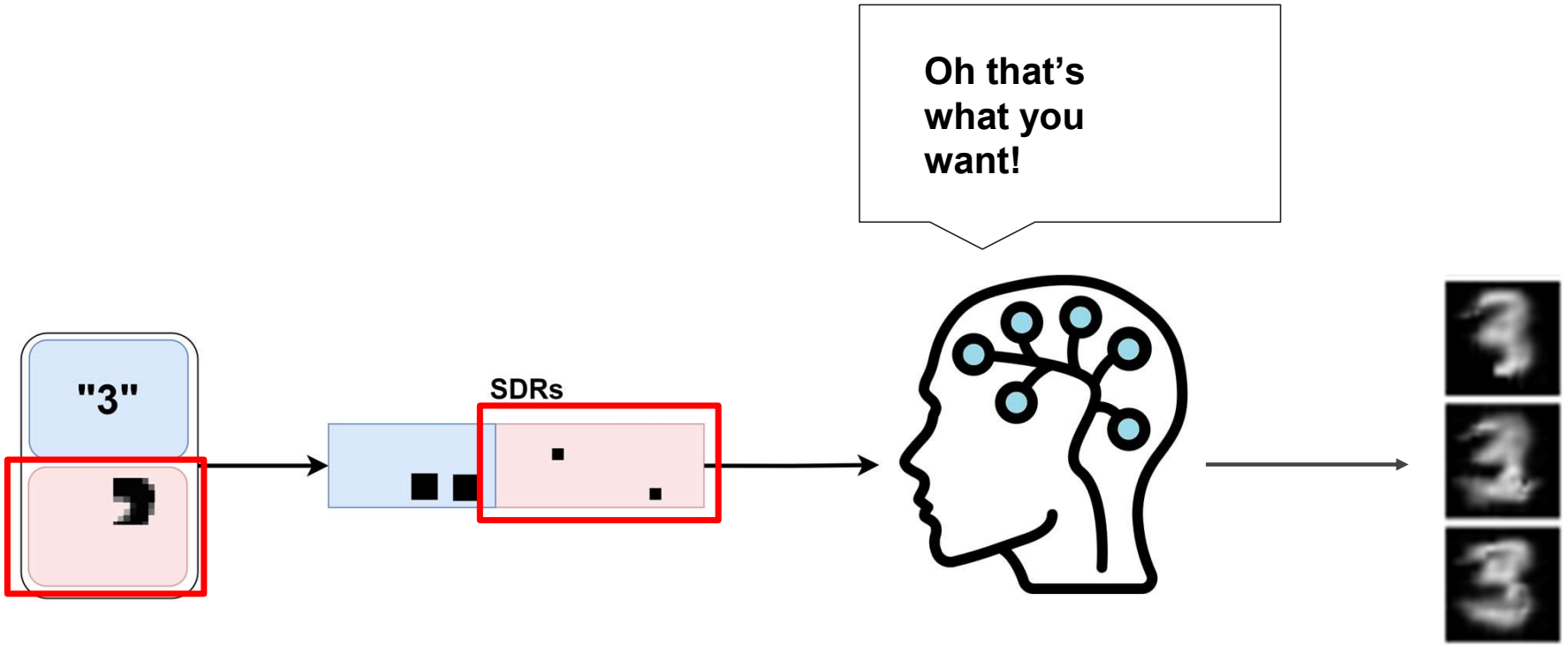
To “seed” the generation process



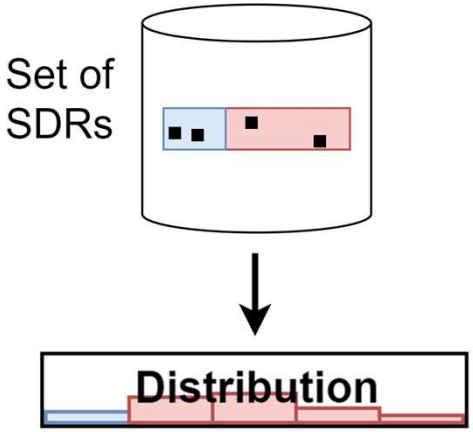
And give more realistic results



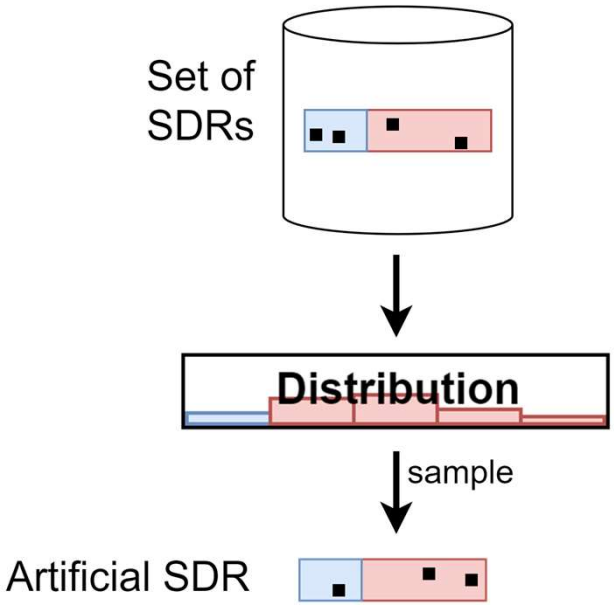
But how do we create this “seeds”?



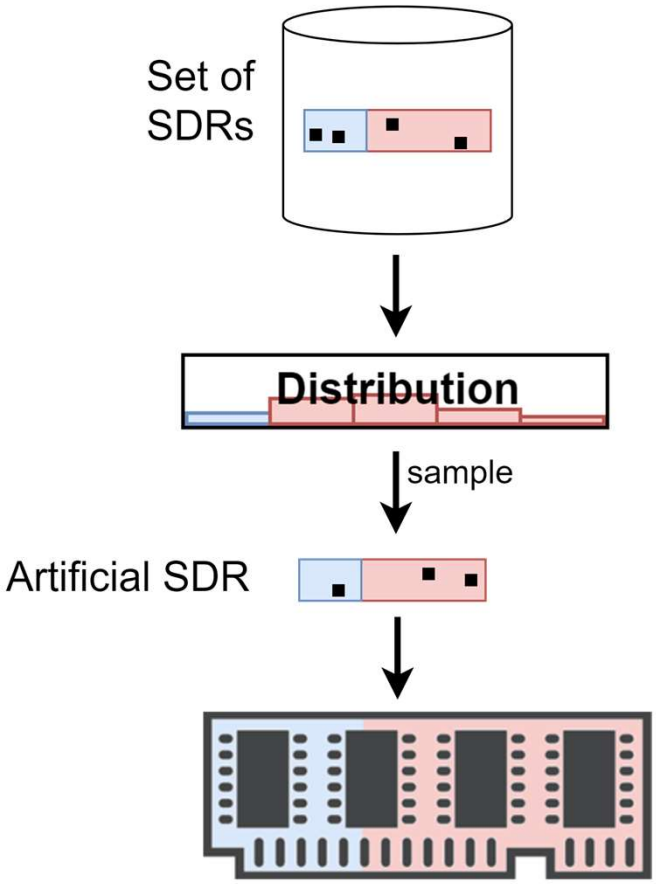
If only we had a probability distribution...



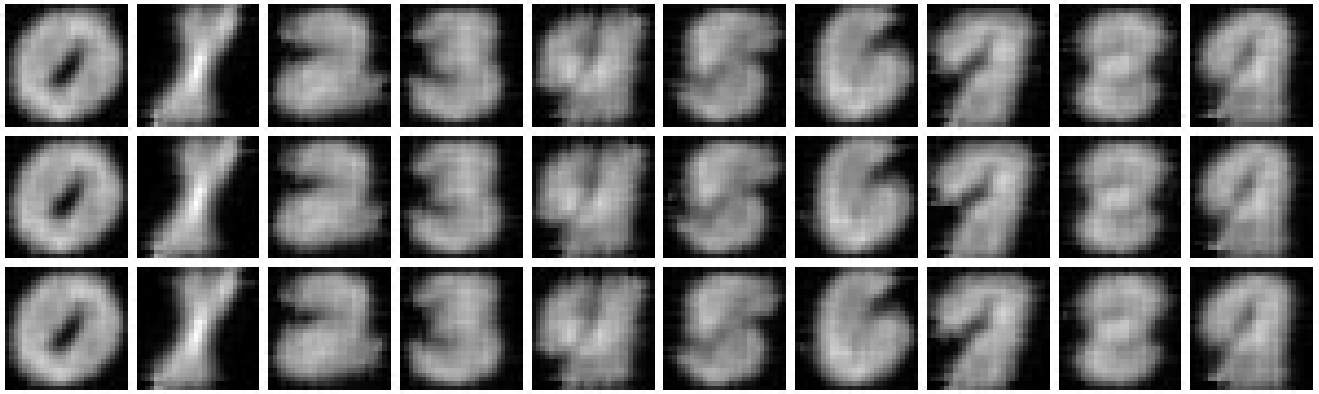
We could create generative “seeds”



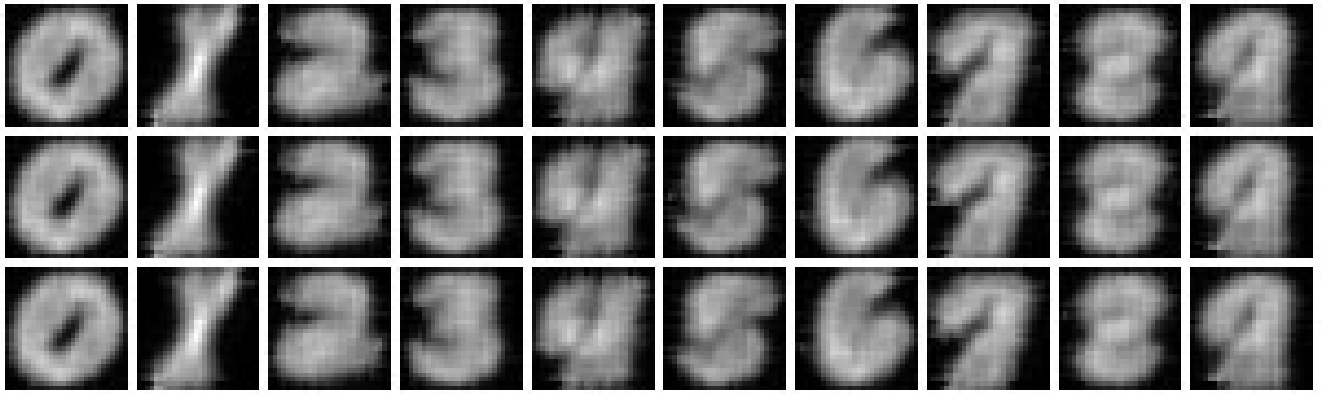
And generate...



We don’t have a distribution, but we have BLOBS!



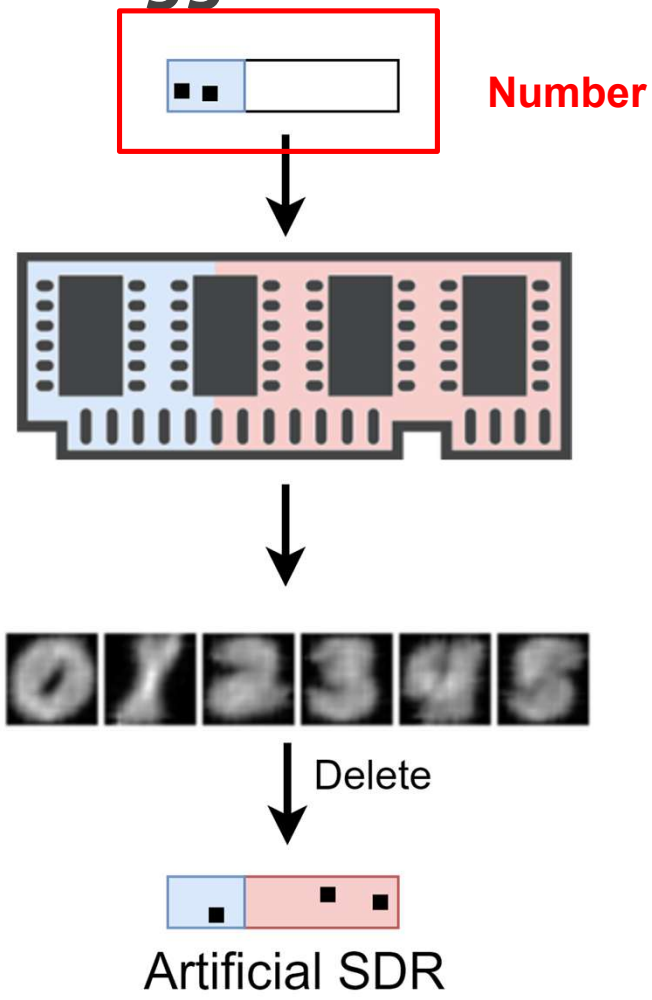
We don’t have a distribution, but we have BLOBS!



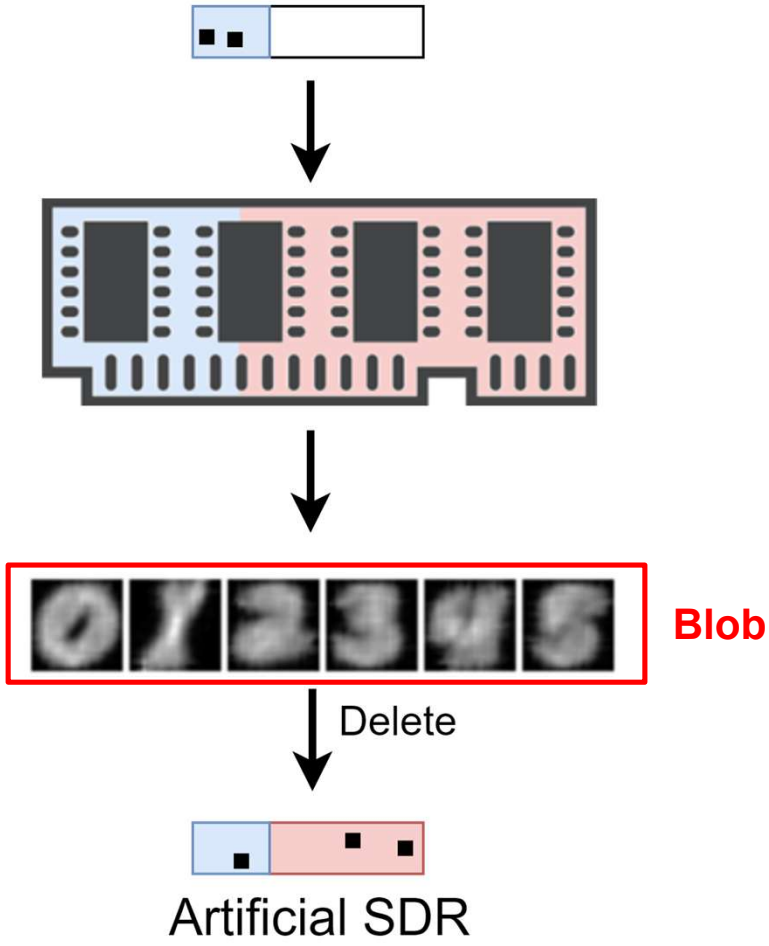
No Variance

Combination of all features in a number

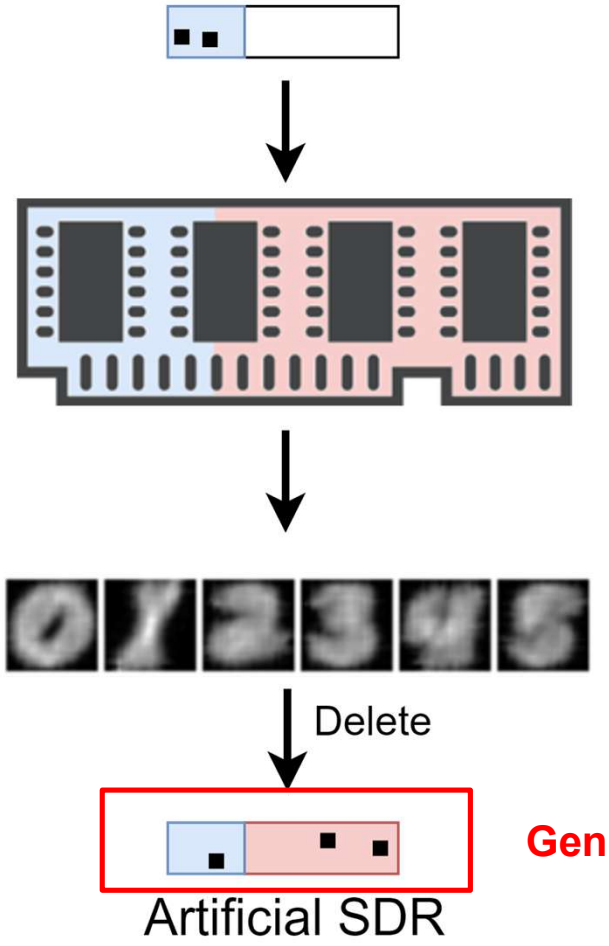
Proposed seeding strategy



We can create the blobs

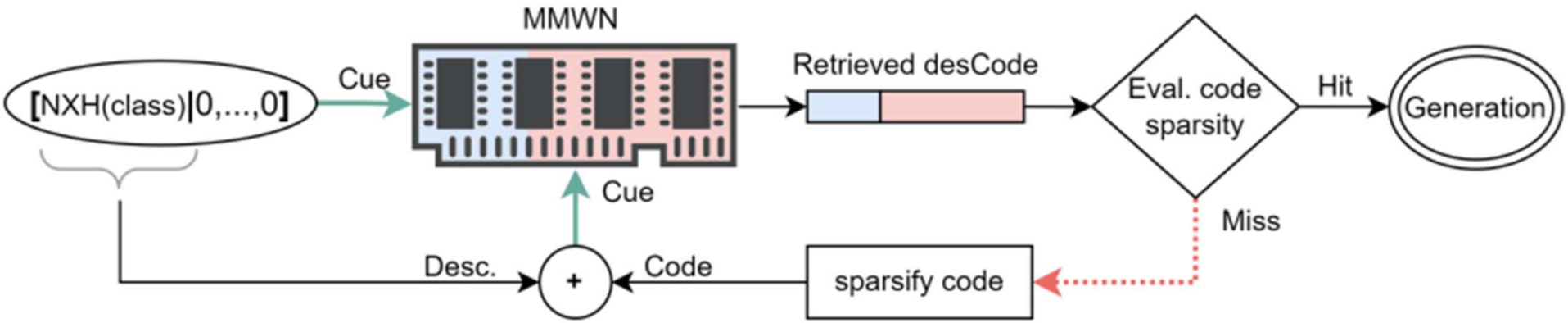


And get to a generation seed

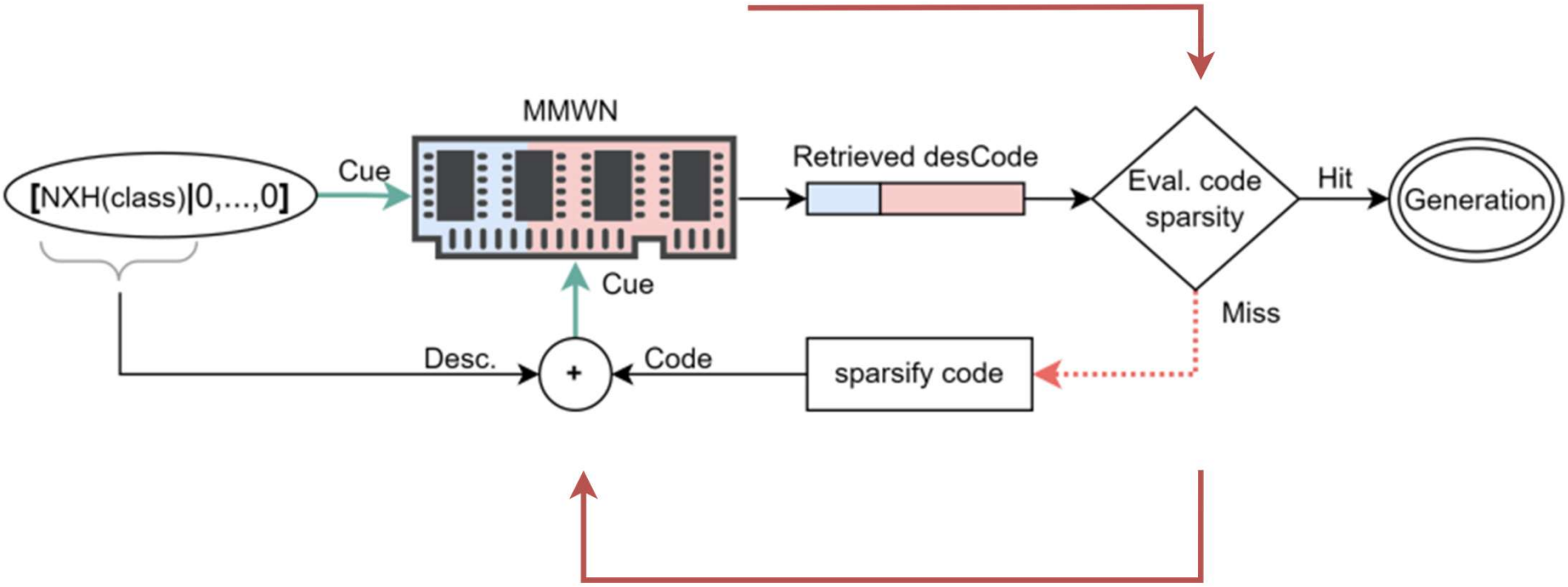


Generation Seed with class information

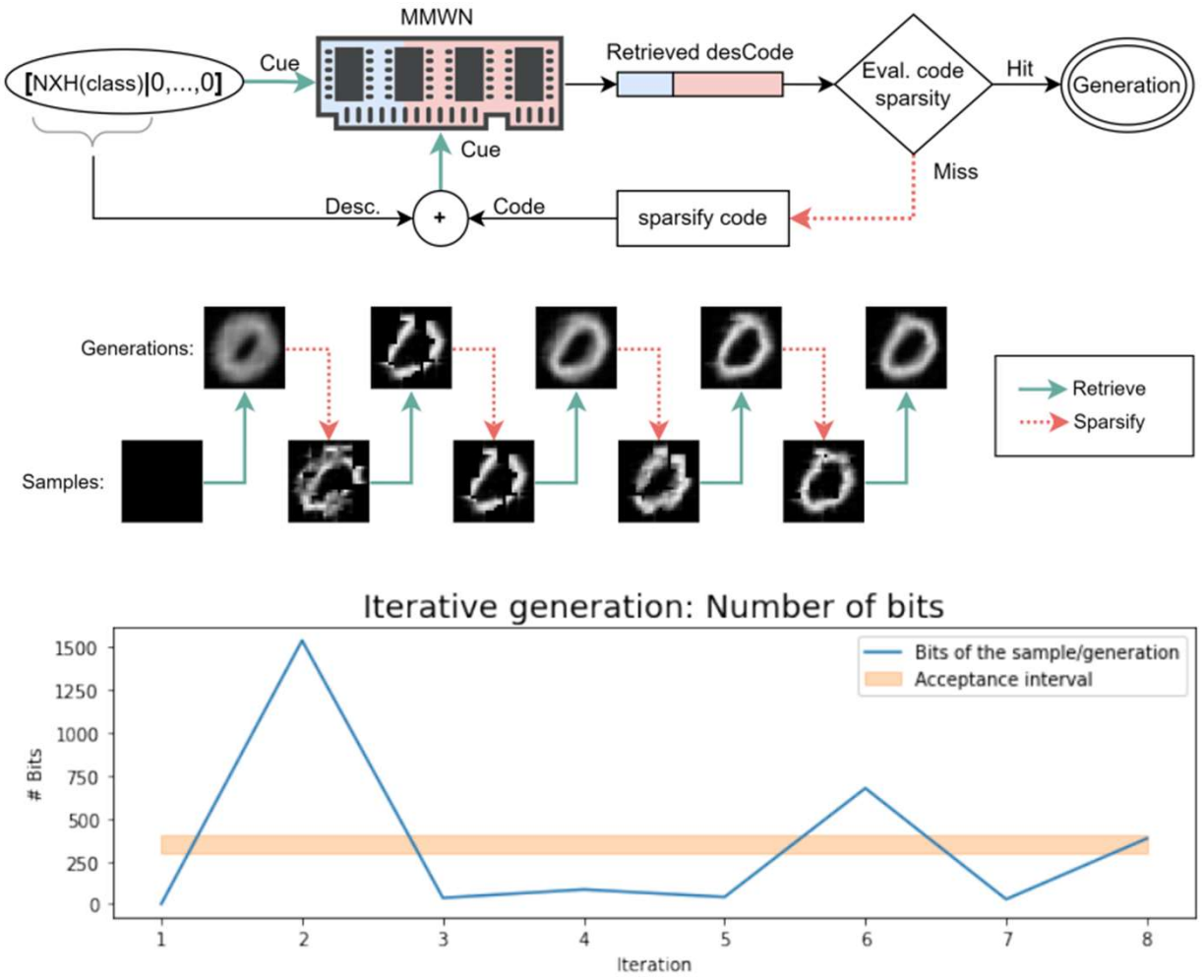
Propose Solution: Iterative Generation



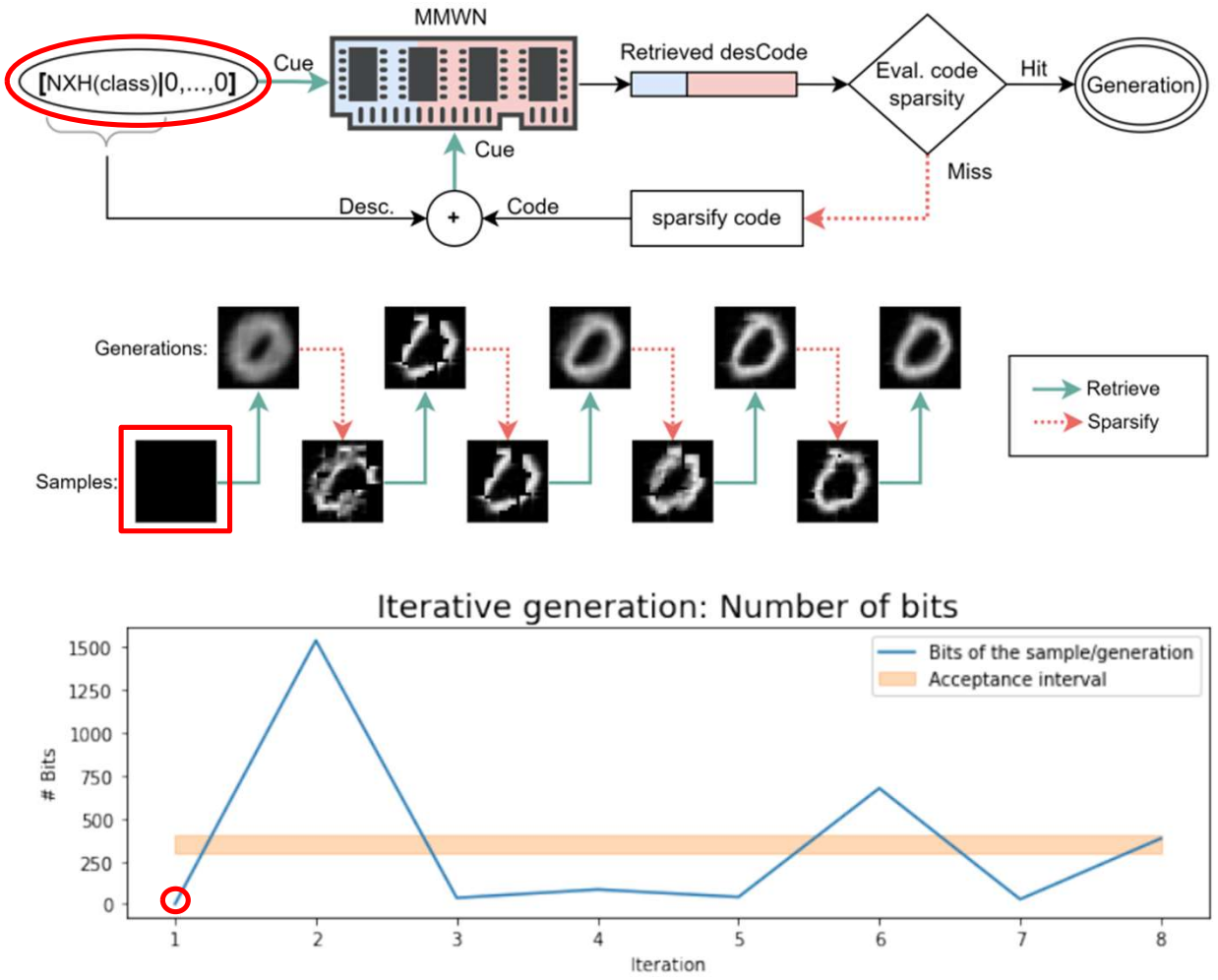
Iteratively use the memory



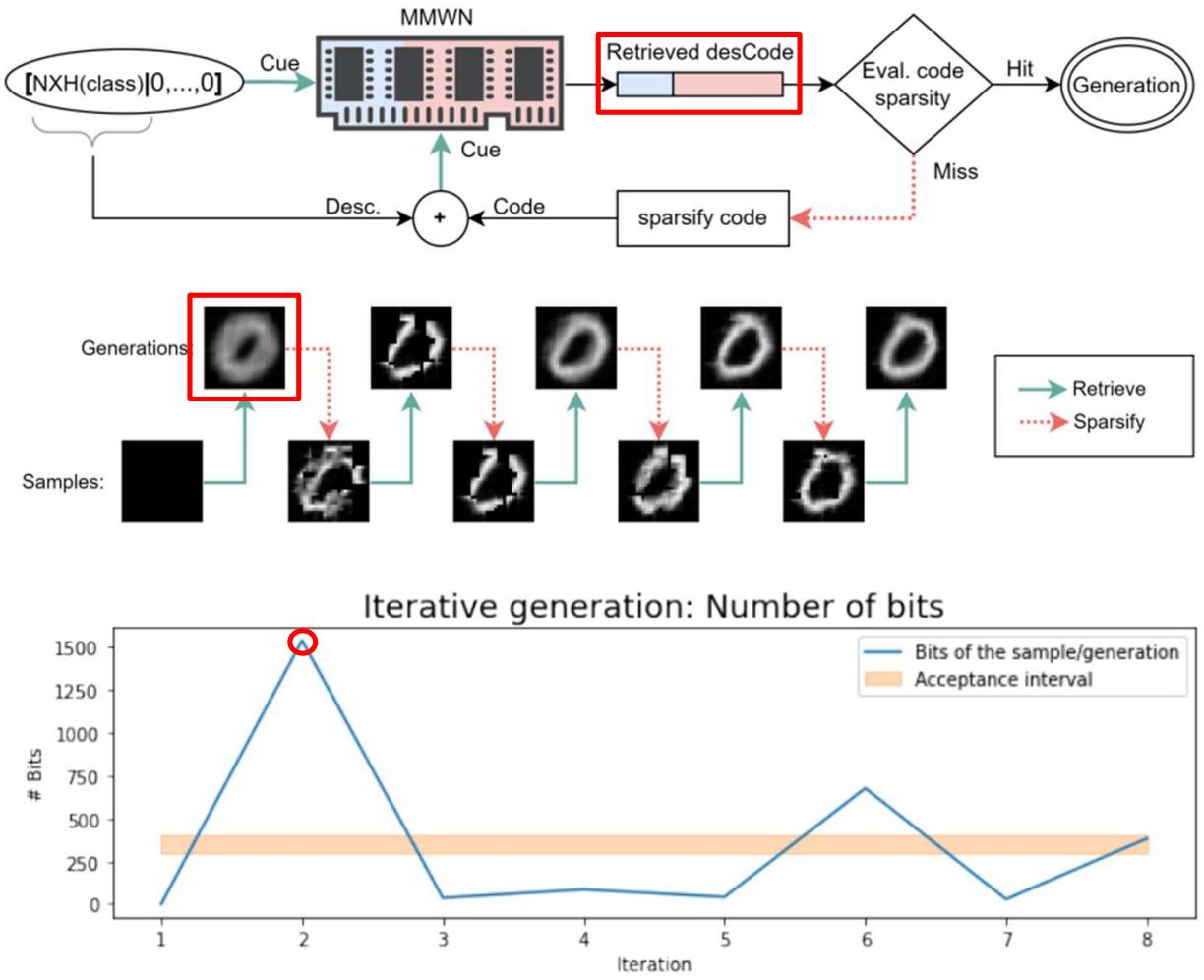
Let us illustrate with an example:



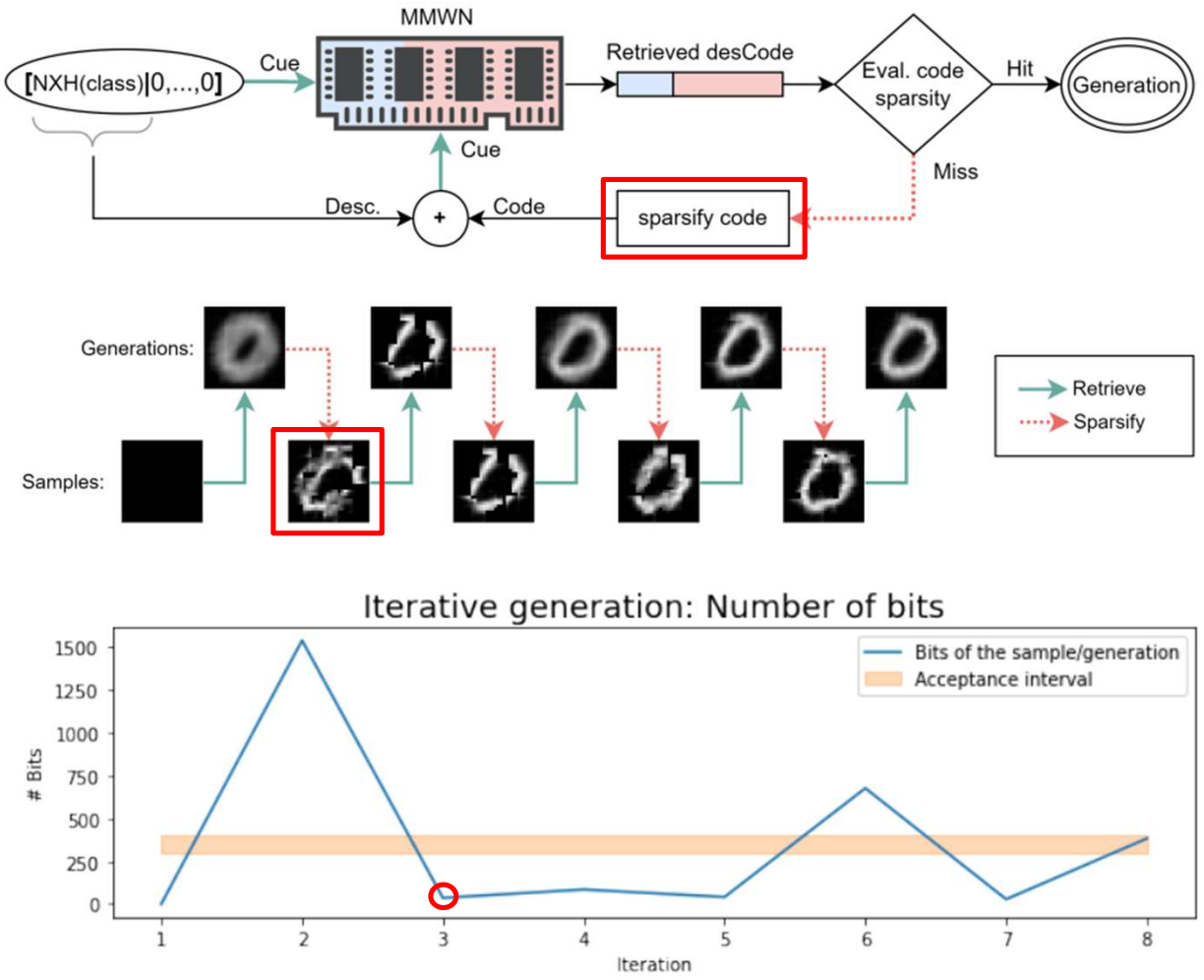
We start with a number, and no visual information



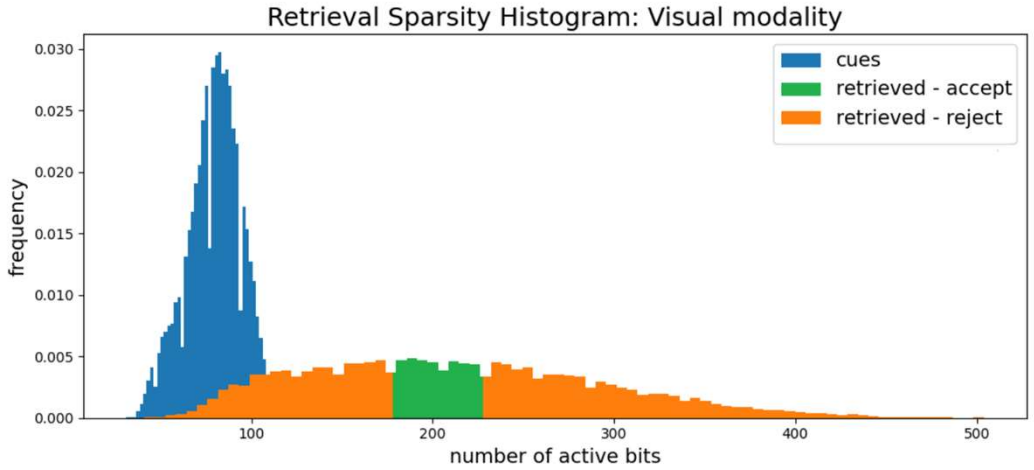
And obtain a blob

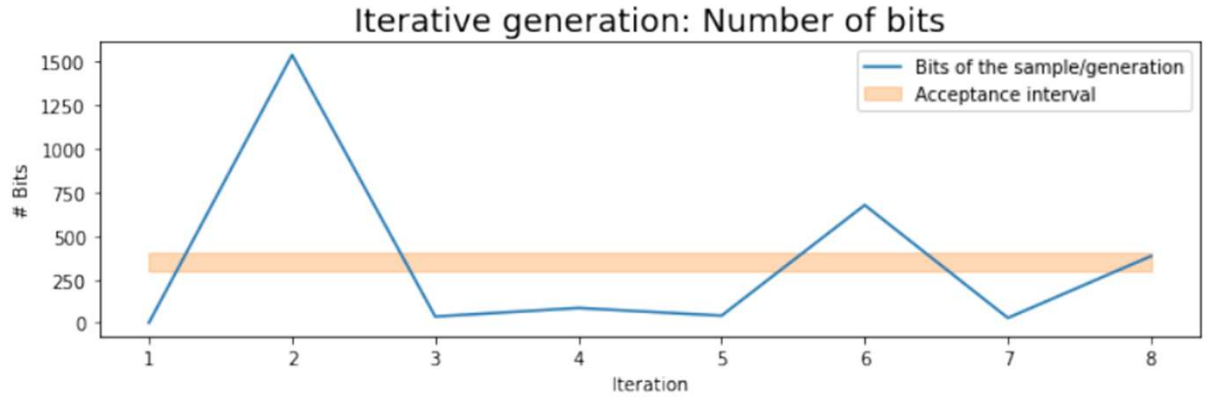


We create a seed

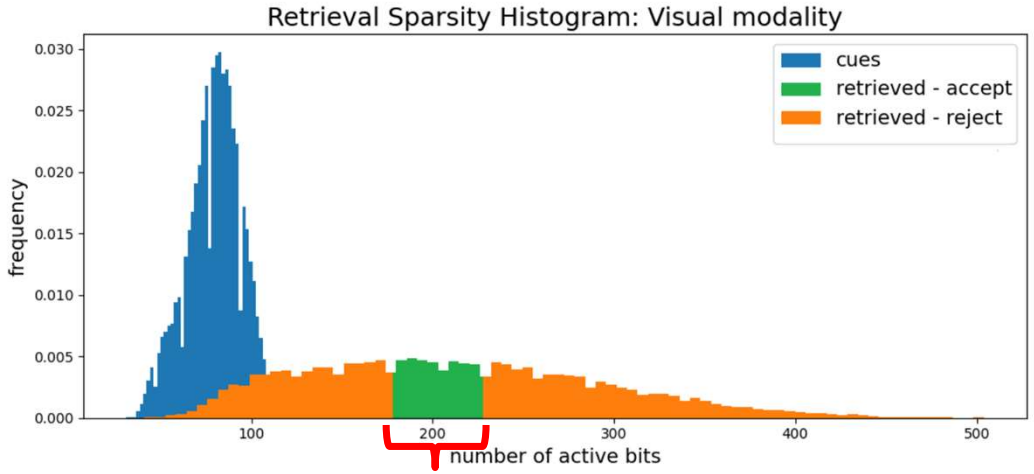


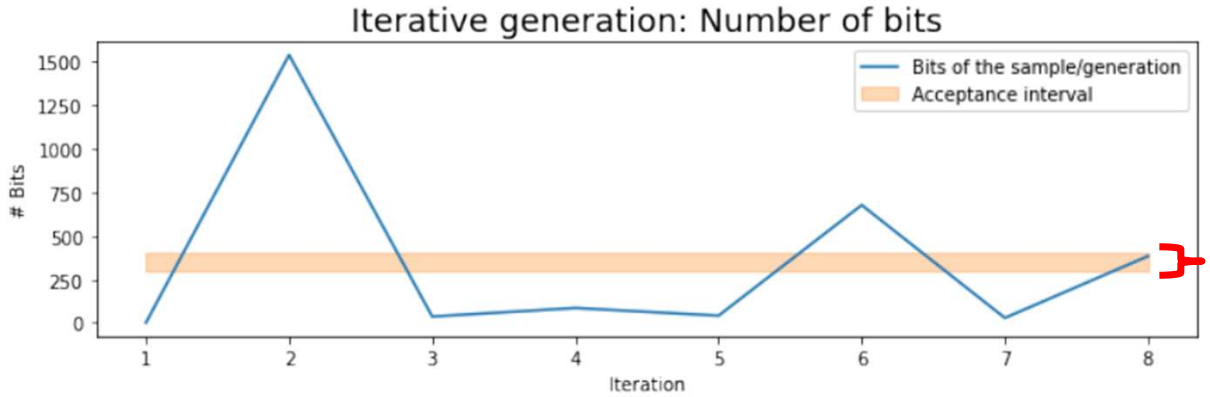
The number of bits in the output is indicative of its quality



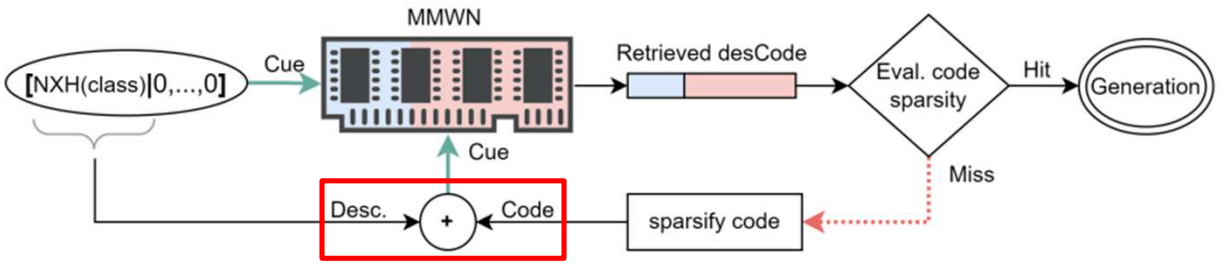


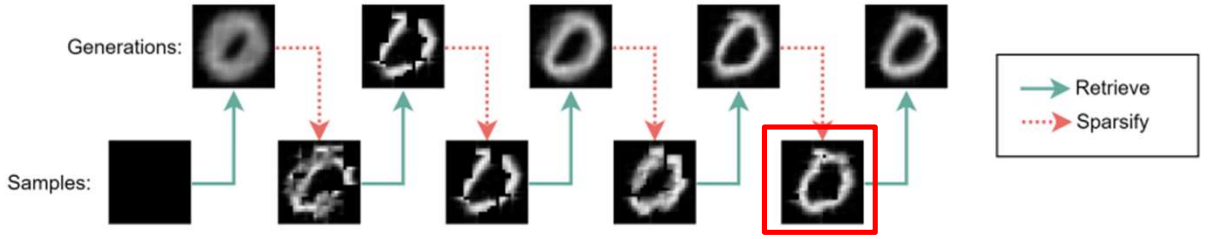
We define an acceptance interval

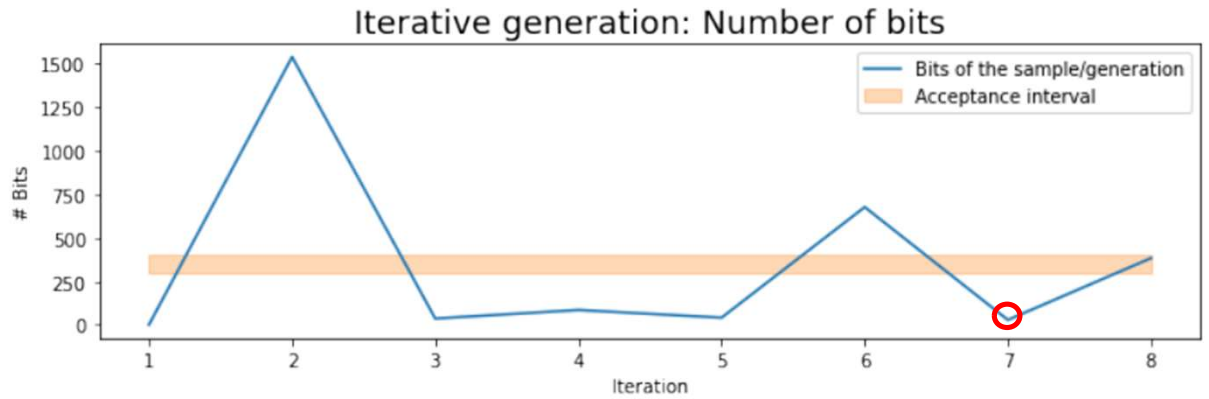




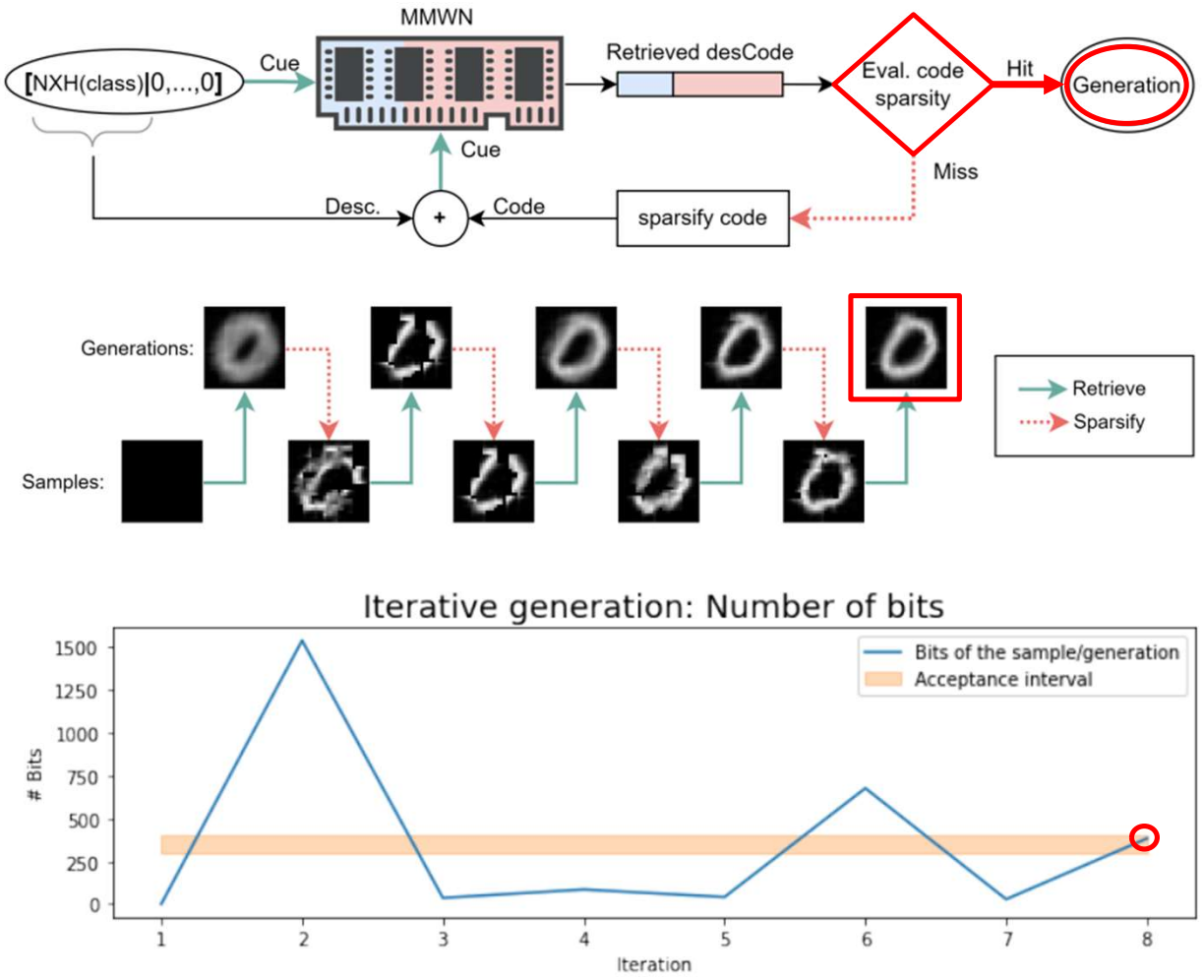
With a good seed







We fall in the acceptance interval



Iterative generation - results

Intra-class variance

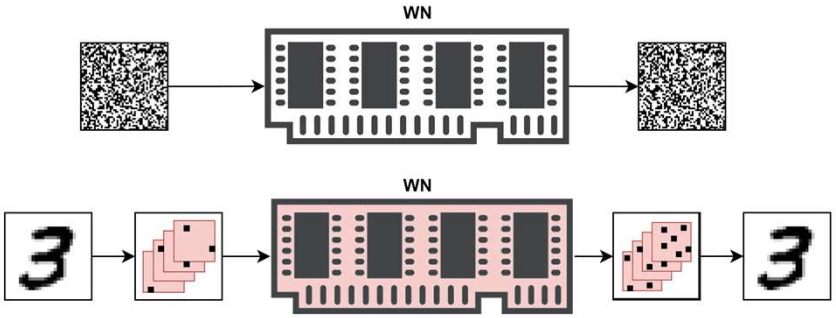
Outline

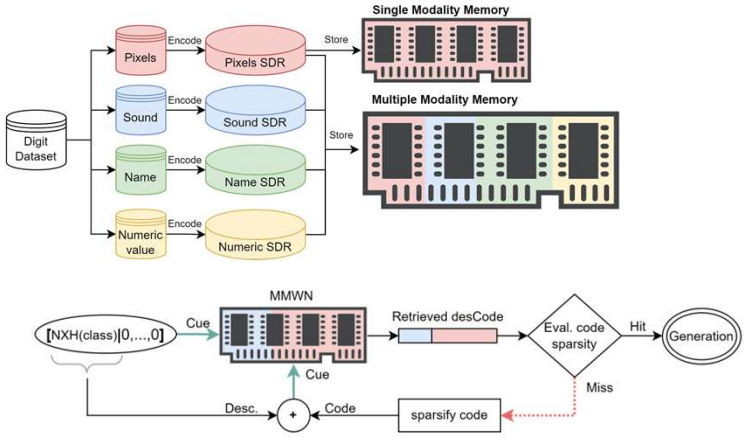
- Background
- Analysis of the Retrieval of Visual Patterns
- Multi-Modal Willshaw Network
- Conclusion

Main Contributions

1. A complete **literature review** on the topic of Associative Memory.

(1)

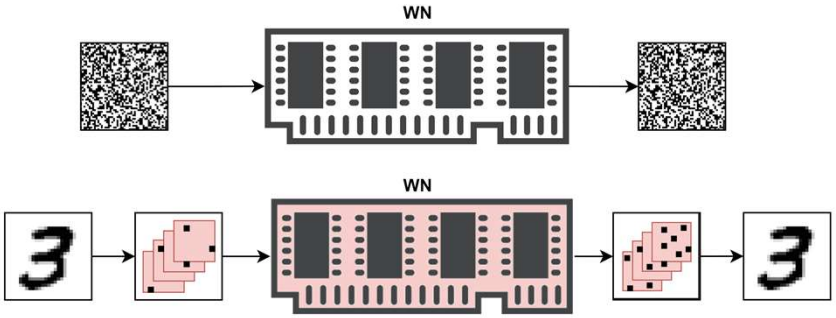


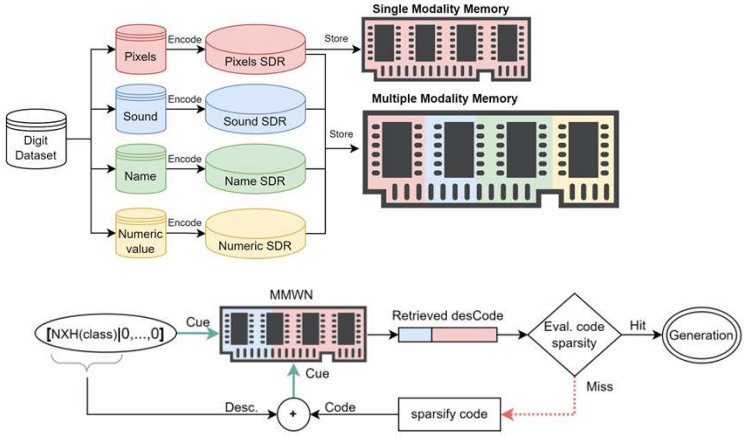


Main Contributions

1. A complete **literature review** on the topic of Associative Memory.
2. A thorough **analysis** on the WN's retrieval of **visual patterns**.

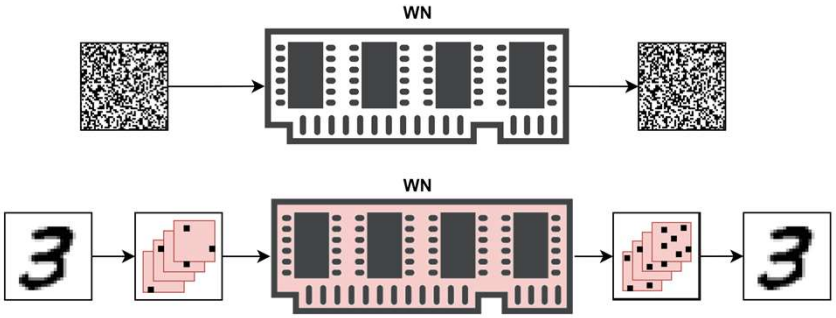
(2)

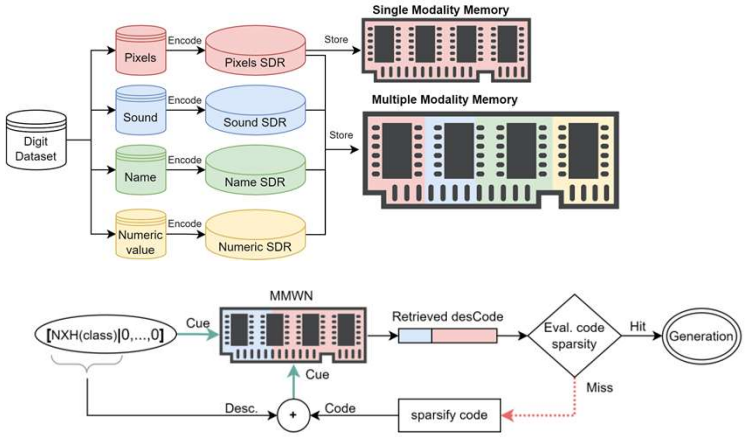




Main Contributions

1. A complete **literature review** on the topic of Associative Memory.
2. A thorough **analysis** on the WN's retrieval of **visual patterns**.
3. The proposal of a **new architecture:** the Multi-Modal Associative memory.

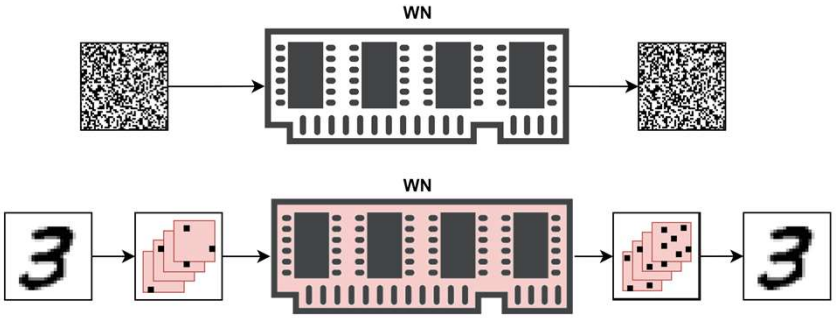


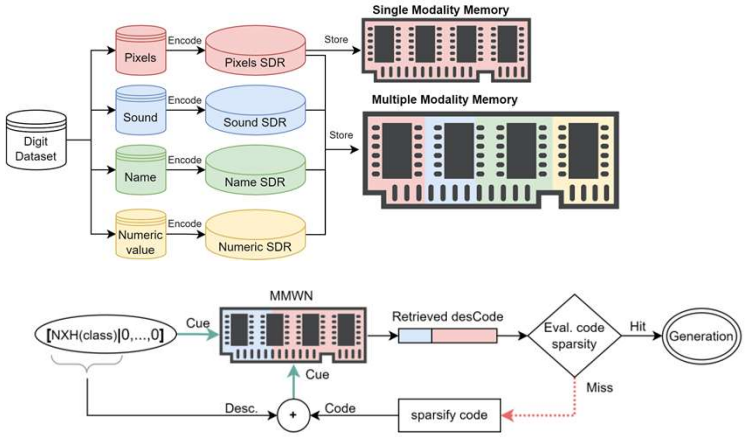


(3)

Main Contributions

1. A complete **literature review** on the topic of Associative Memory.
2. A thorough **analysis** on the WN's retrieval of **visual patterns**.
3. The proposal of a **new architecture:** the Multi-Modal Associative memory.
4. The proposal of a **new algorithm** for the WN: iterative retrieval.

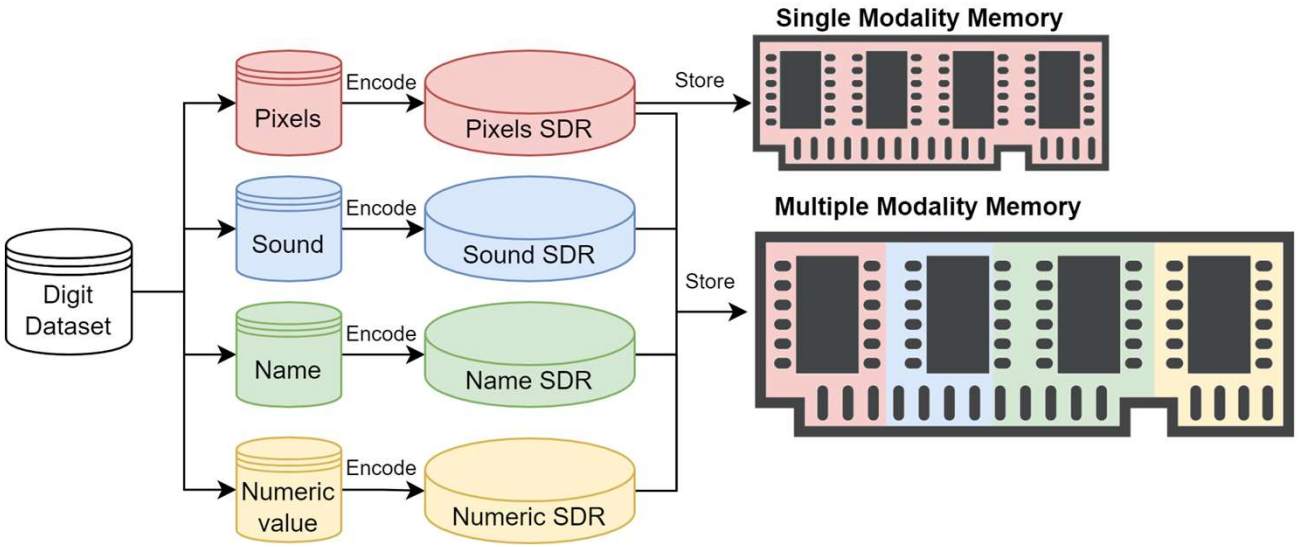




(4)

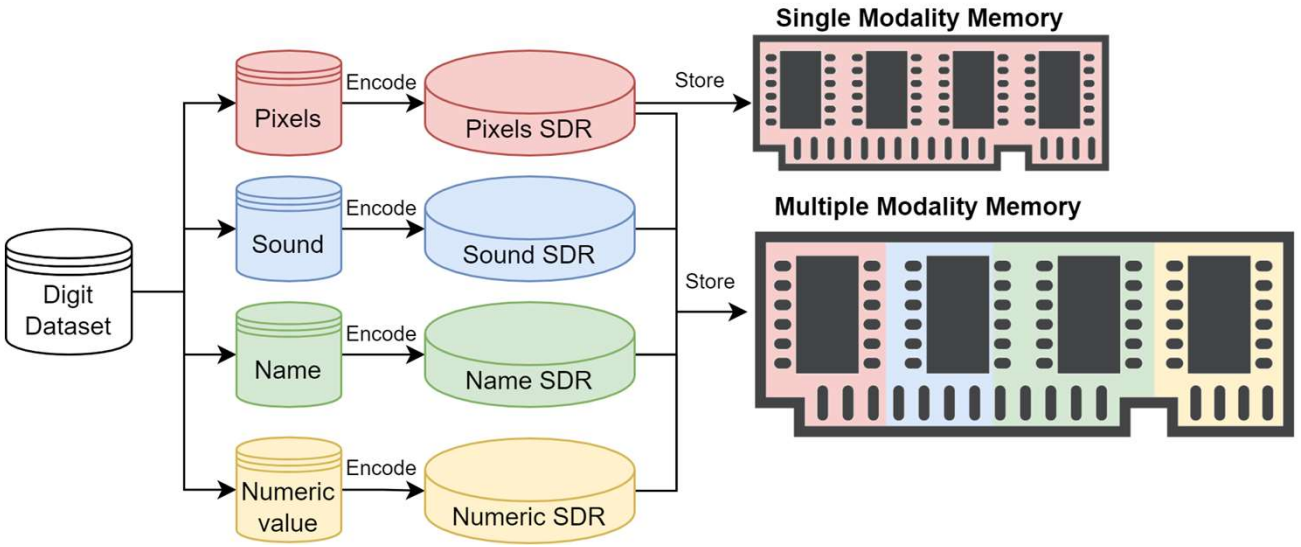
Proposed Architecture Advantages

1. **Biologically** Constrained (Hebbian Leaning, etc…)



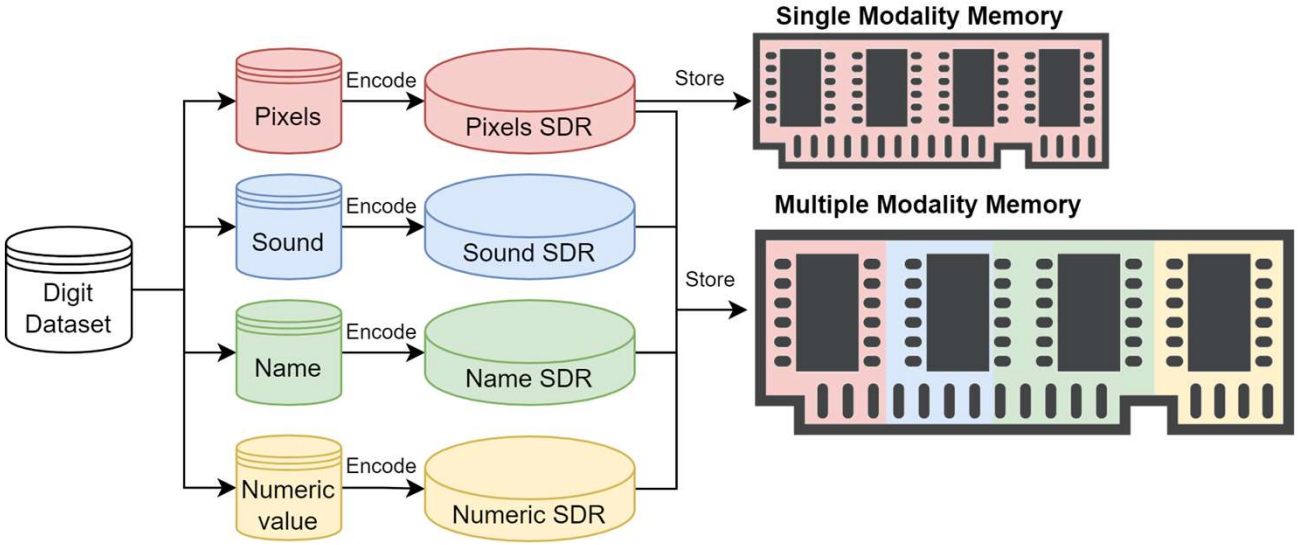
Proposed Architecture Advantages

1. **Biologically** Constrained (Hebbian Leaning, etc…)
2. **Efficiency** (single epoch training)



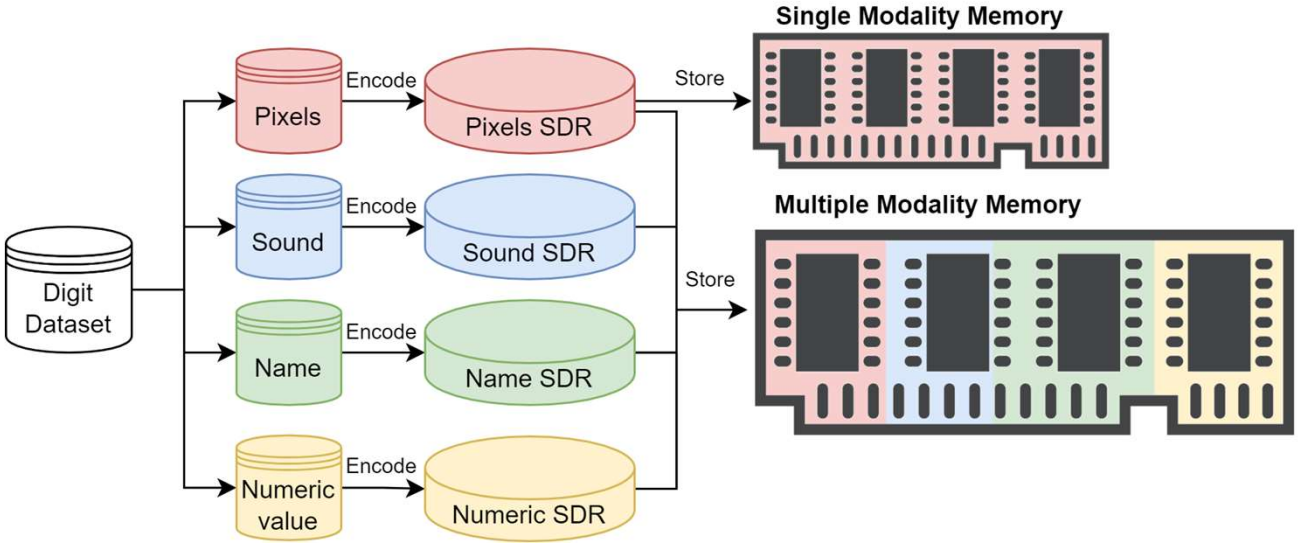
Proposed Architecture Advantages

1. **Biologically** Constrained (Hebbian Leaning, etc…)
2. **Efficiency** (single epoch training)
3. **Flexibility** (Classification, Generation, Retrieval)



Proposed Architecture Advantages

1. **Biologically** Constrained (Hebbian Leaning, etc…)
2. **Efficiency** (single epoch training)
3. **Flexibility** (Classification, Generation, Retrieval)



Multi-modal Associative Memory:

A framework for Artificial Intelligence

Paper coming soon...

Multi-Modal Associative Memory: A framework for Artificial Intelligence

Rodrigo Simas, *Instituto Superior Técnico, Lisbon, Portugal,*

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 the perception. The Willshaw Network (WN), is a model that aims to mimic these mechanisms of its biological counterpart. The usage of this model in practical applications is hindered by the so-called *Sparse Coding Problem*. To advance, prescriptions that transform raw data into sparse distributed representations are required. In this work, we use a recently proposed prescription [1] that maps visual patterns into binary feature maps. We analyze the behavior of the WN on real-world data and gain key insights into the strengths and weaknesses of this model. To further enhance the capabilities of the WN, we propose the Multi-Modal framework. 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 unlocking many practical applications. As a proof of concept, we perform experiments on the MNIST dataset. By storing both the images and labels as modalities, we were able to successfully perform retrieval, classification, and generation with a single model. Our results highlight the flexibility of the Multi-Modal framework to perform various classical Machine Learning tasks with a single network and provide a big hint on how the field of Associative Memories can advance in practical settings.

Index Terms

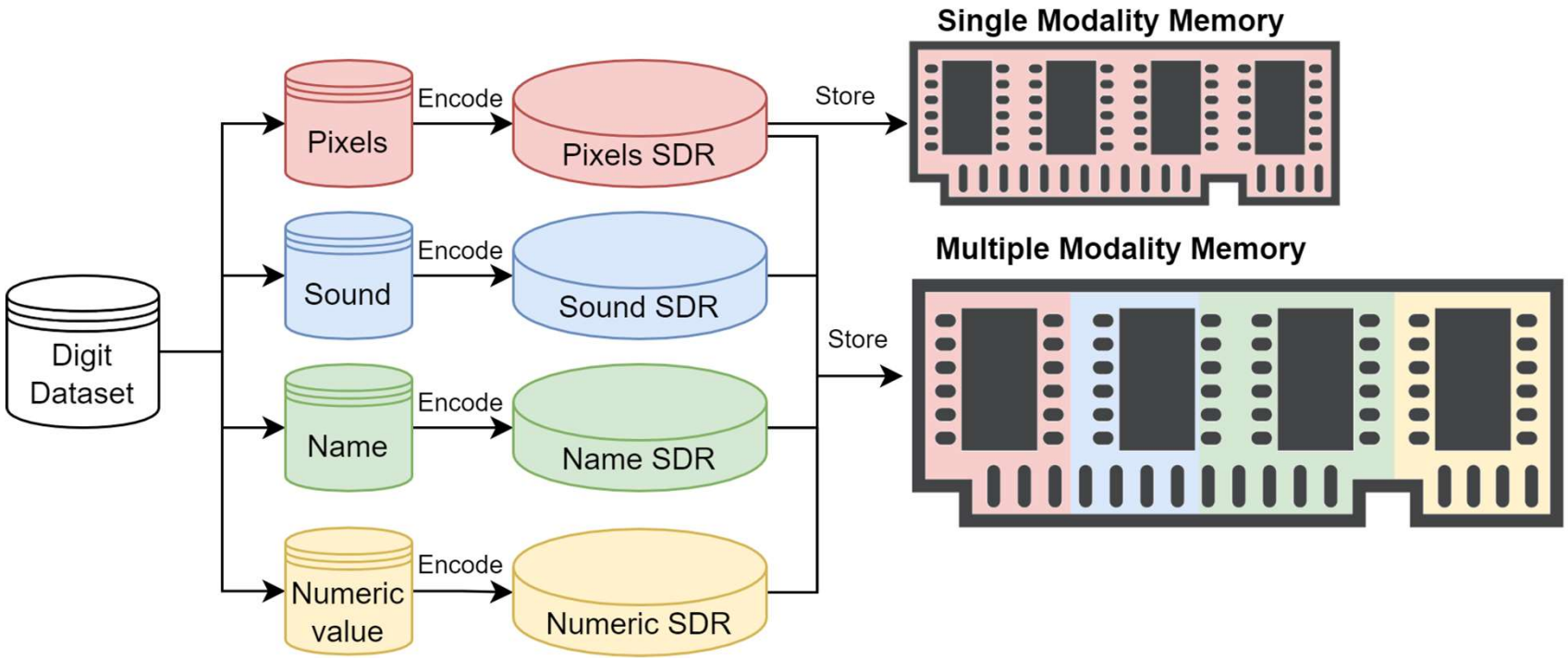
Associate Memory, Auto-Association, Willshaw Network, Generation,s Classification

I. Introduction

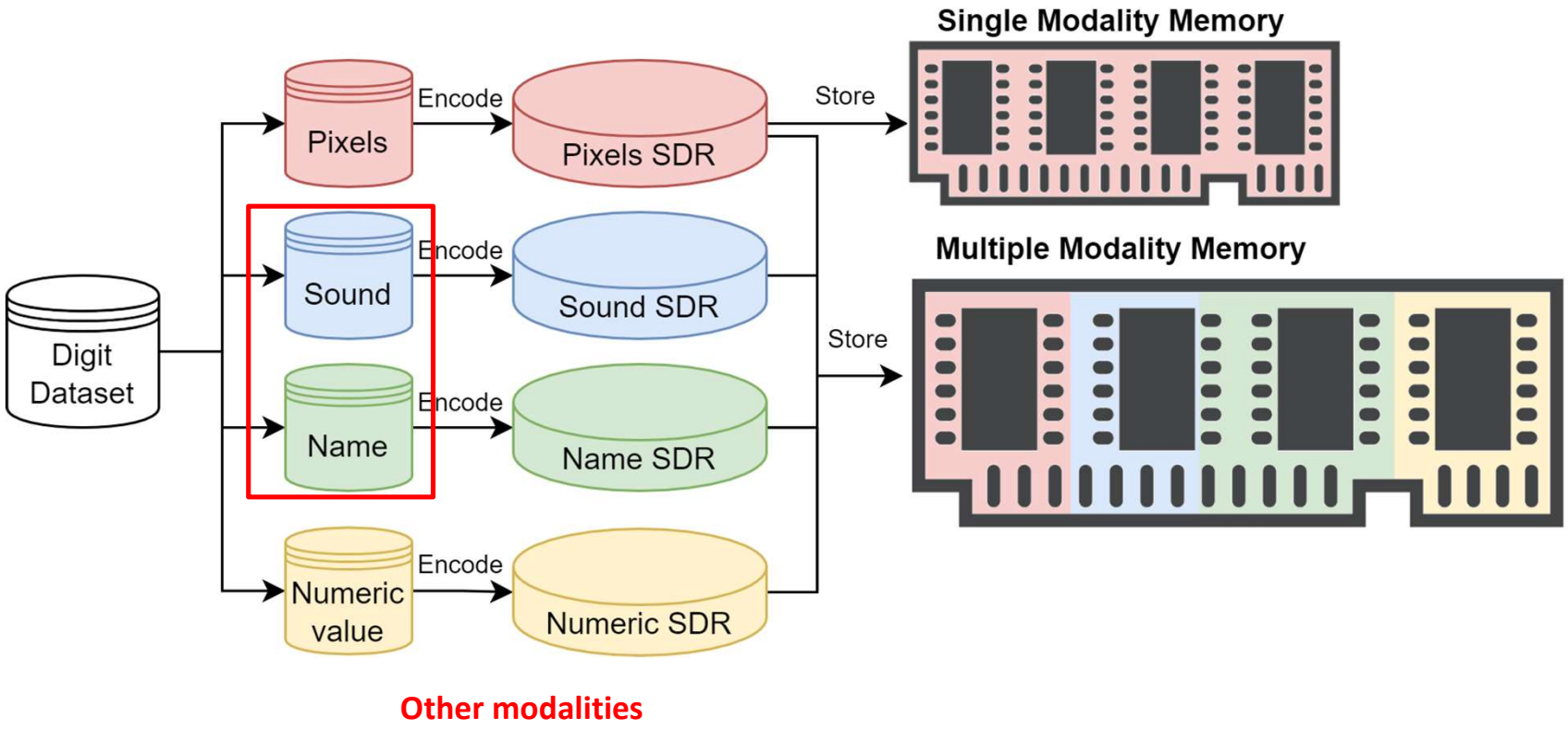
TO create intelligent machines, understanding the human brain and its mechanisms is a fundamental step. For this reason, the focus of this work was the field of AMs: a family of Biologically-Inspired Artificial Intelligence models that imitate biological memories [2].

These models train by storing associations between pairs of patterns: Correlated features form synaptic connections between

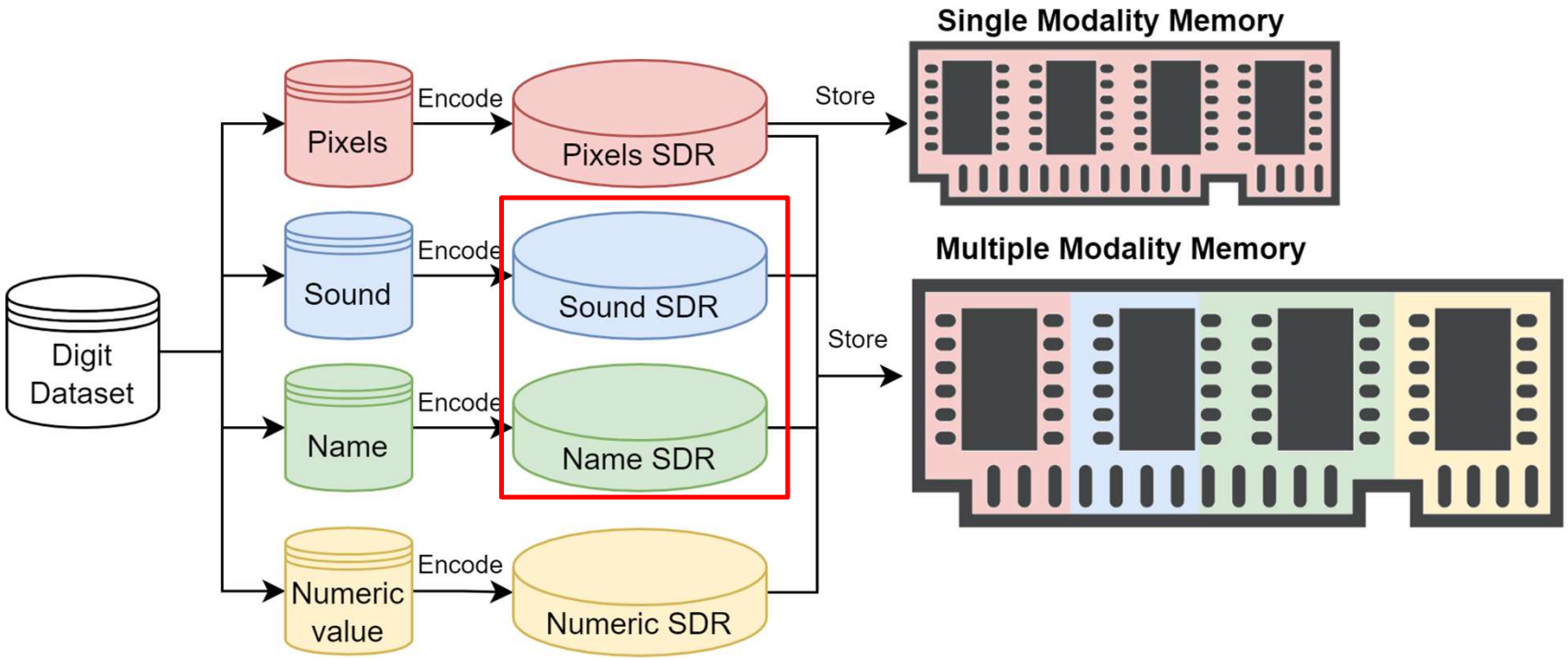
Future work: Scale the framework



Future work: Scale the framework

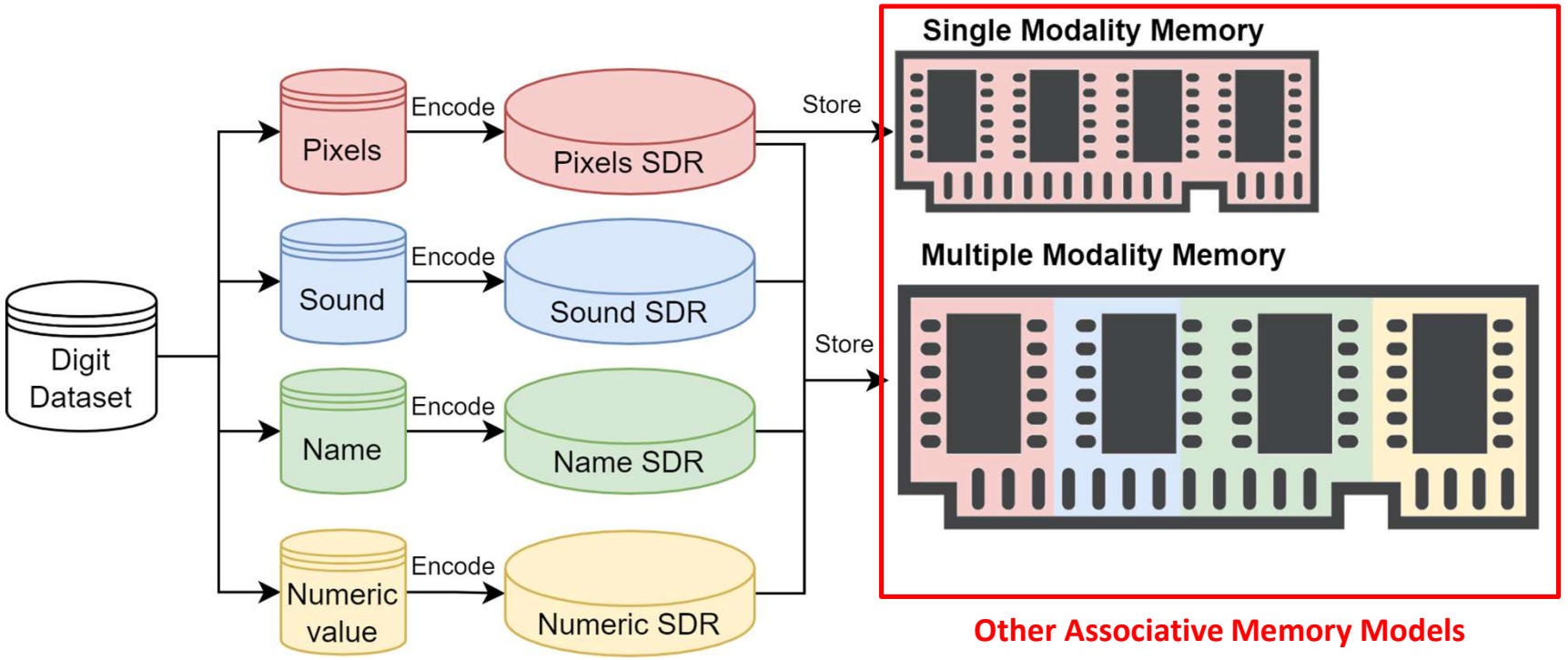


Future work: Scale the framework

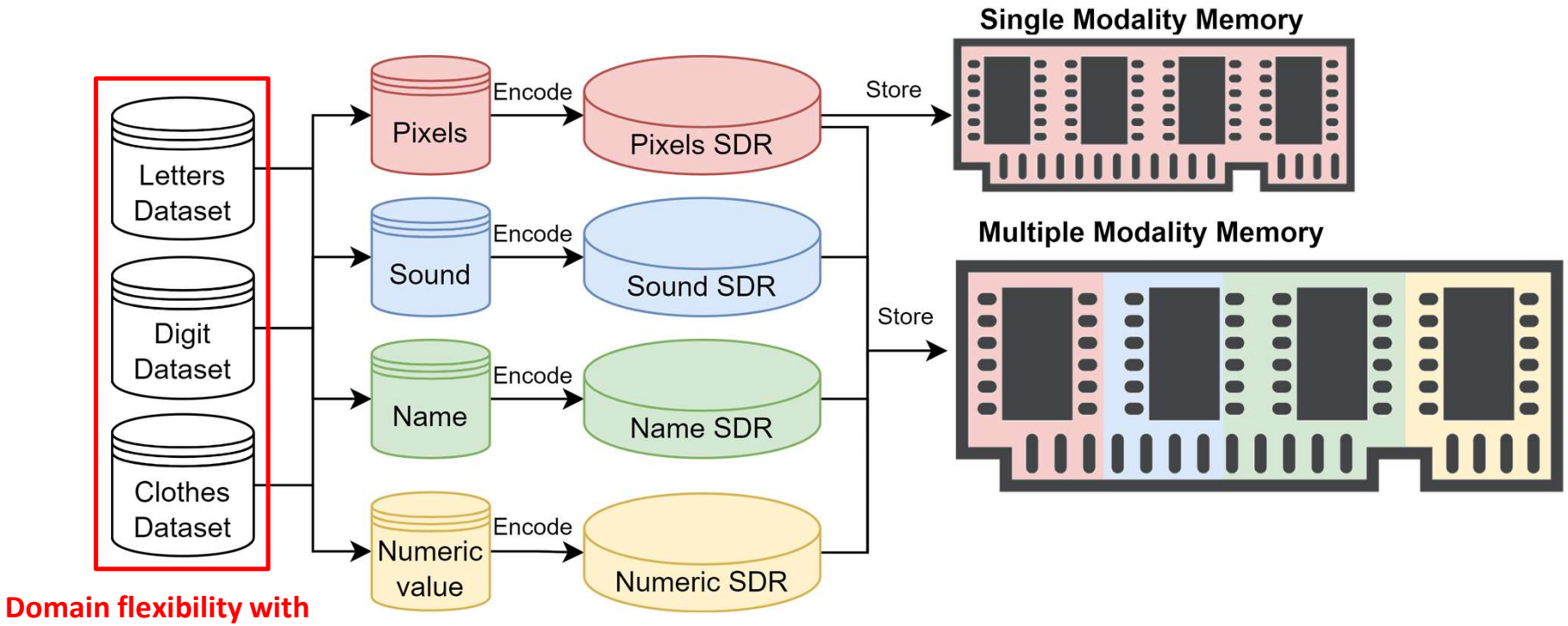


With other Encoders and Decoders

Future work: Scale the framework



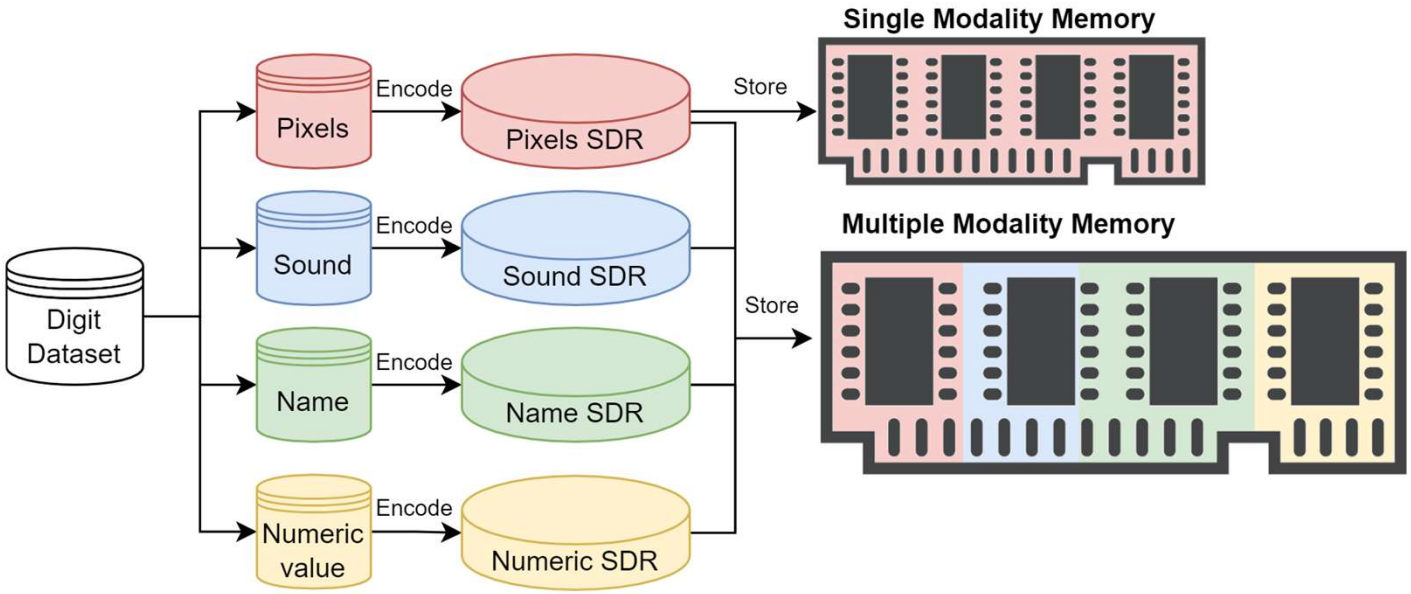
Future work: Scale the framework



Domain flexibility with shared modalities

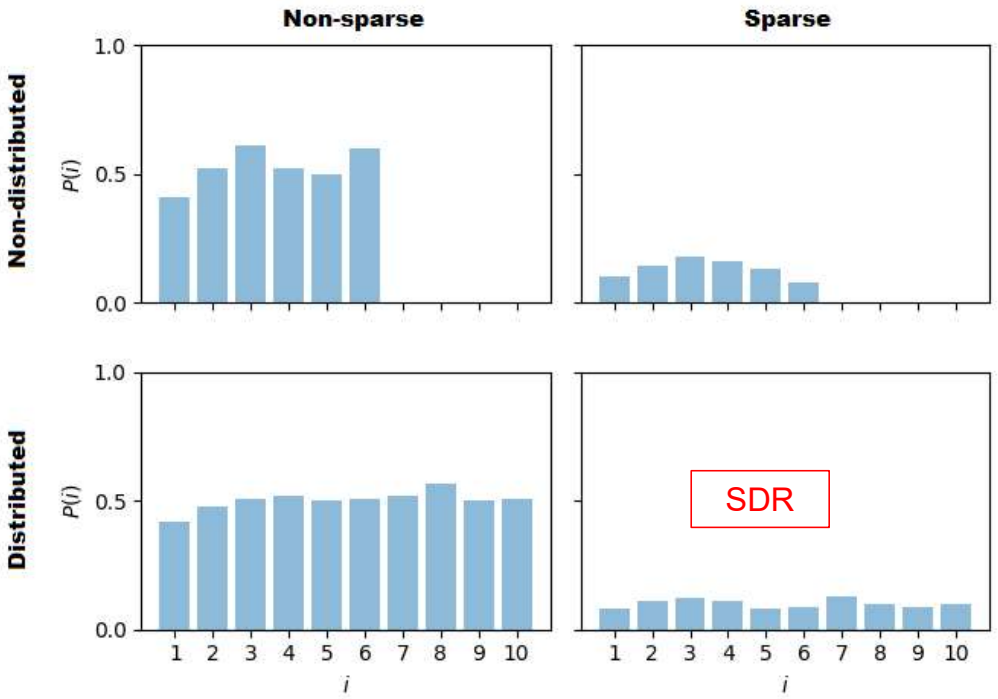
Thank you for your attention!

Thesis to obtain the Master of Science Degree in:
Information Systems and Computer Engineering

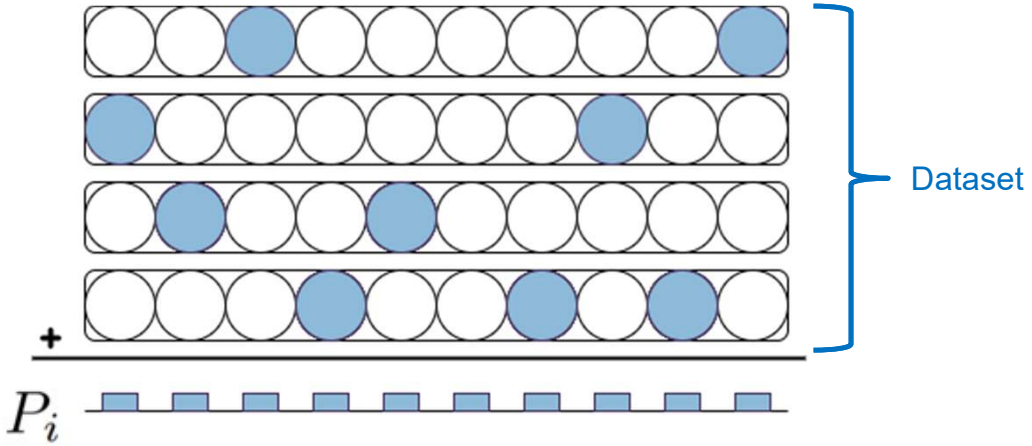


Student: Rodrigo Simas
Supervision: Prof. Andrzej Wichert, Luís Sá Couto
Arguer: Prof. João Pereira
Jury: Prof. Daniel Gonçalves (President)

Sparse Distributed Representations



$$P_i = \frac{1}{M} \sum_{\mu=1}^{M} x_i^{\mu}$$



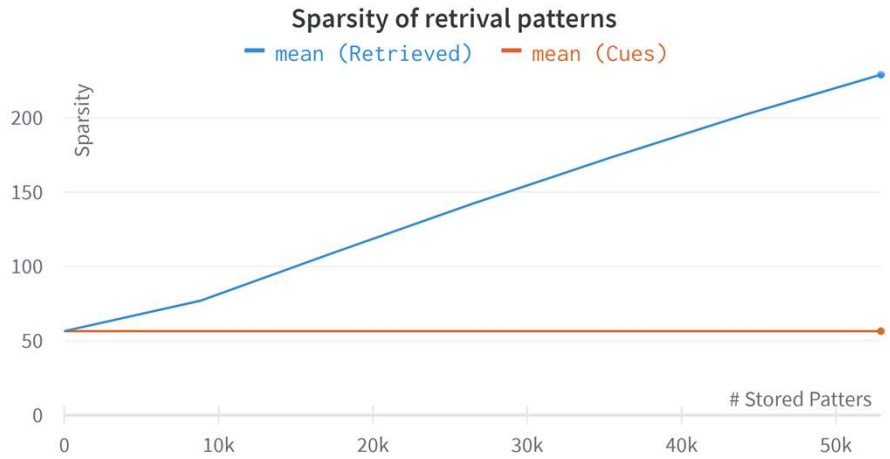
(a) Stored vector

(b) Retrieved: memory with 160 vectors

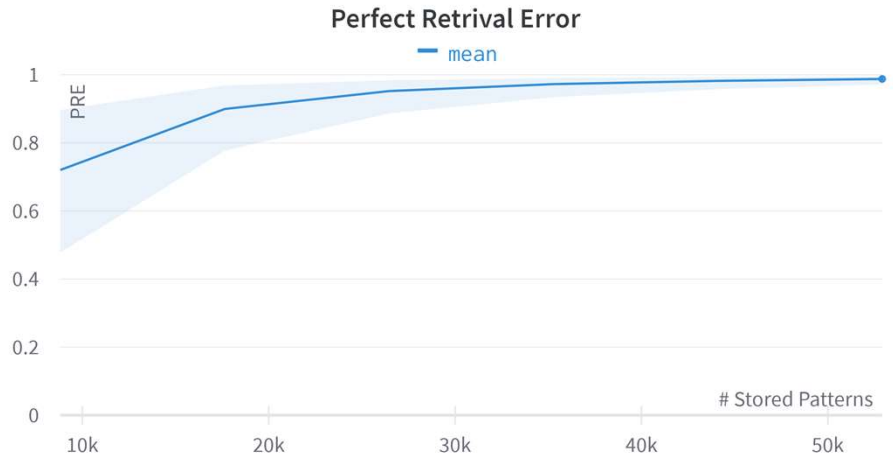
(c) Retrieved: memory with 500 vectors

- **Hand picked to ensure Orthogonality**
- **Just 500**

Retrieval- The Bad

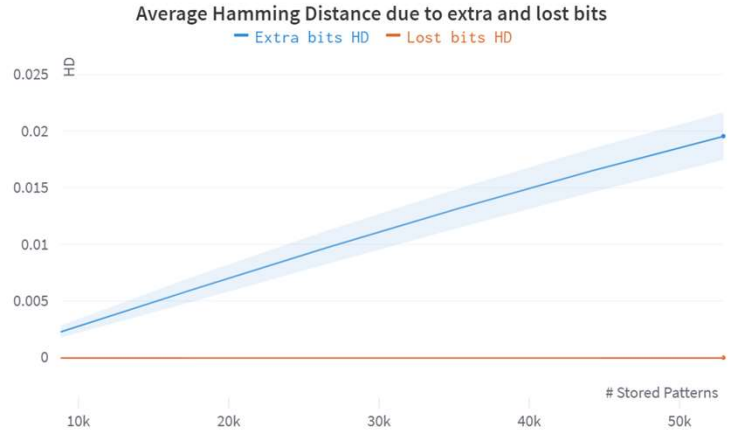


- Memory adds noise

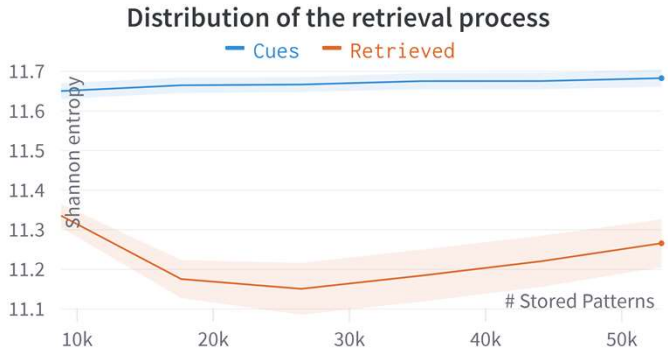


- Retrieval is imperfect

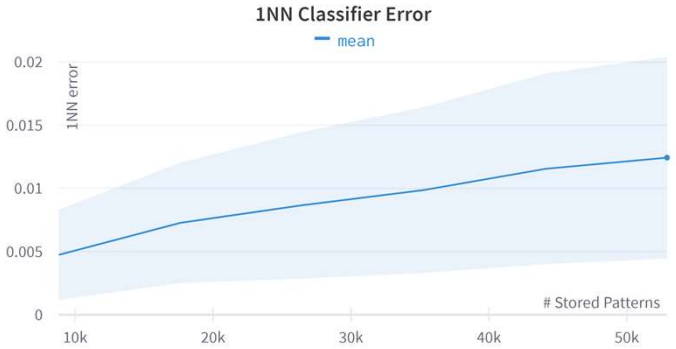
Retrieval - The good



- Memory does not delete information



- Noise is non-random

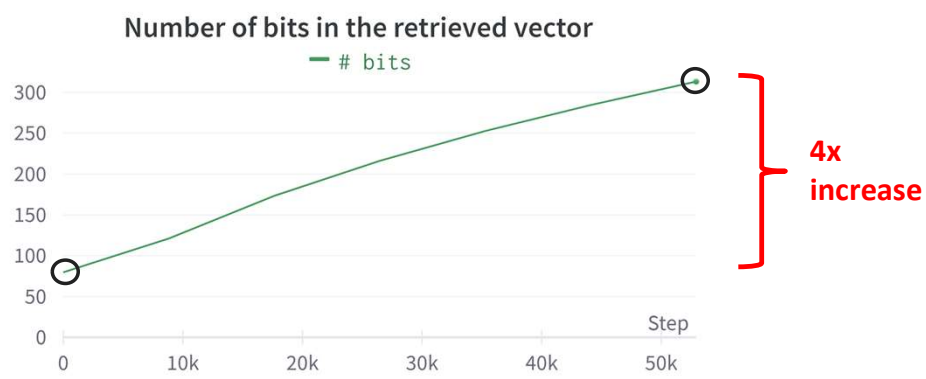


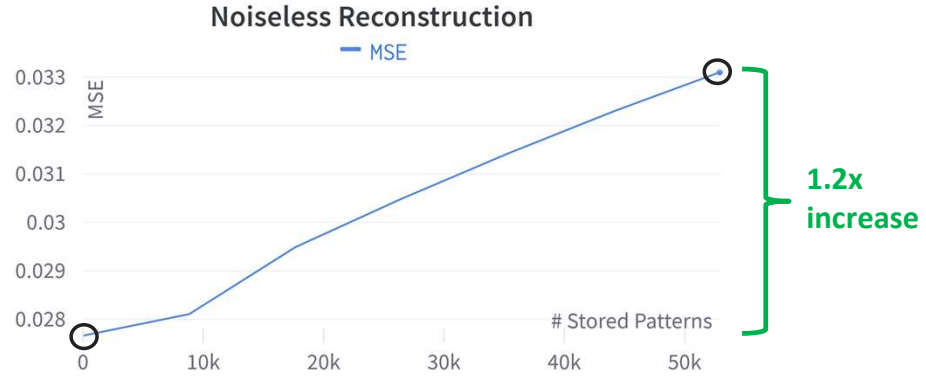
- Noisy patterns retain class information

Reconstructions – noiseless cues

(a) Decodings

(c) Reconstruction (50k stored)





Distribution

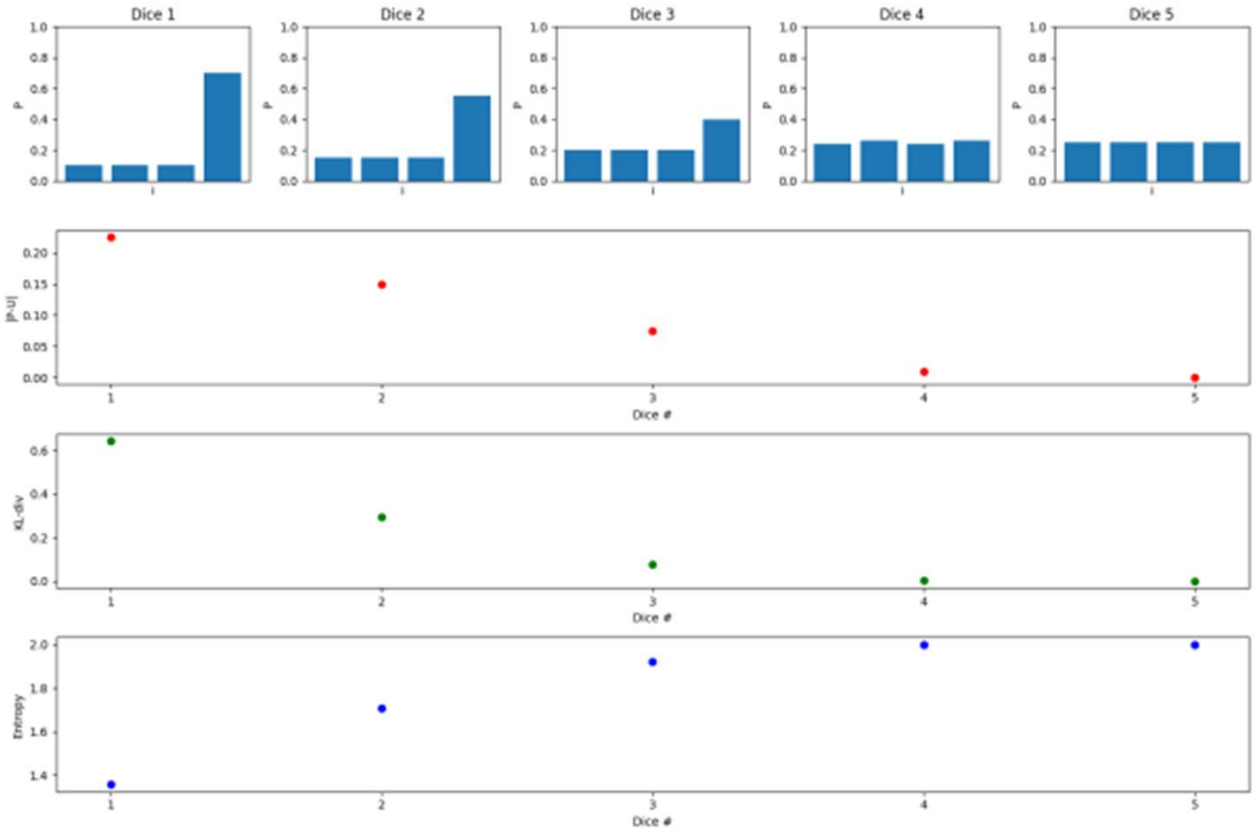


Figure A.2: Distribution measurements: Toy example. **(top row):** Here we present five distinct 4-faced dices that become increasingly more uniform from left to right. Underneath, we measure three distinct distribution measurements from Section 3.2.2 for the different dices: **(second row):** The average norm between P and U; **(third row):** The KL-divergence between P_n and U. **(bottom row):** The Shannon entropy of P. We can see that all measurements have a similar behaviour

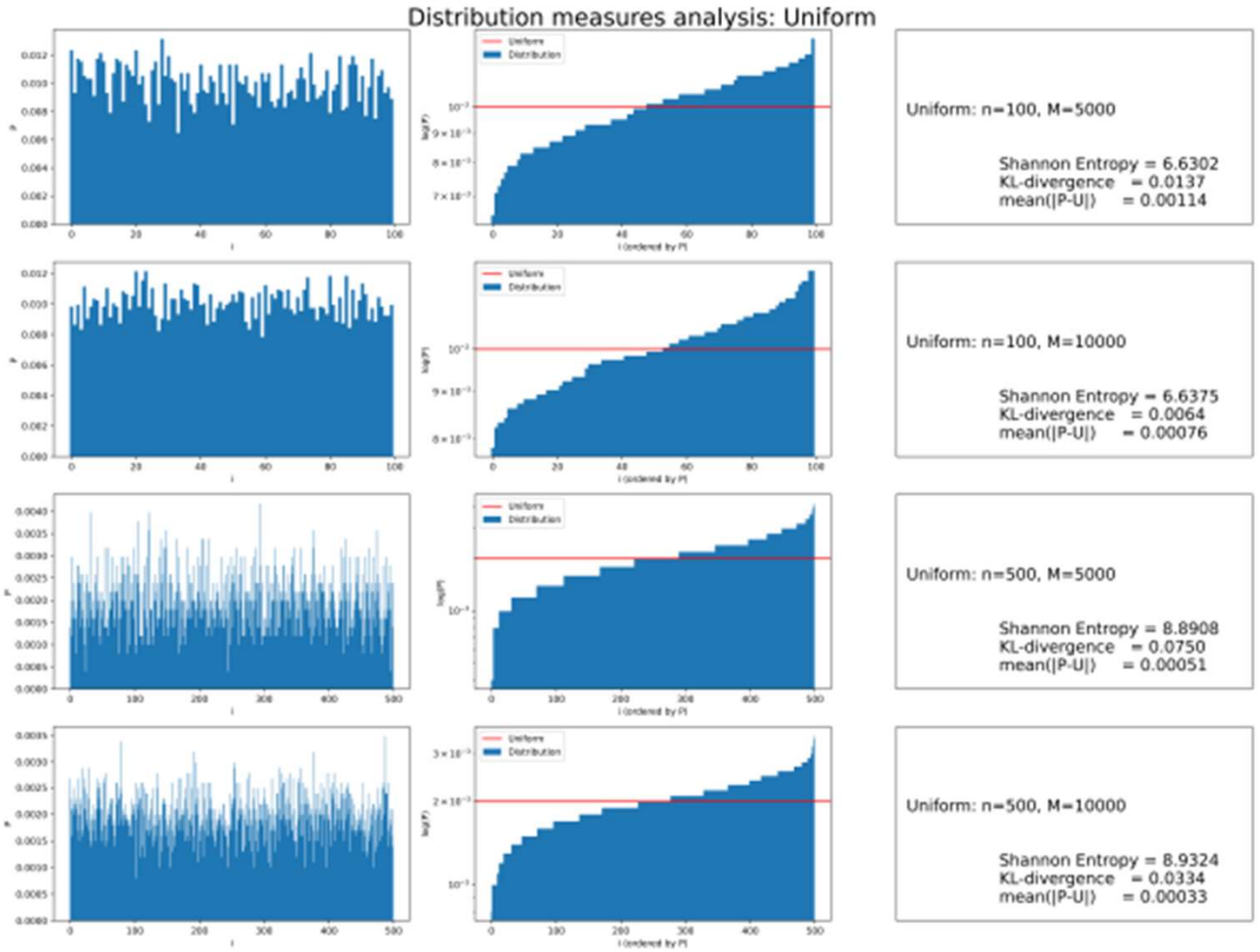


Figure A.4: Distribution measurements: Uniform. Here we randomly sample from the uniform distribution to create artificial binary datasets. On different rows, we use different values for the size of the dataset M, and for the size of the patterns n. In different columns represent: **(Left column:)** P(i) of the set, as defined in Eq. (2.13). **(Middle Column):** The same information as the left column, but the x-axis is sorted, the y-axis has a logarithmic scale, and the uniform probability is plotted so that visual inspection is easier. **(Right column):** Information about the dataset and distribution measurements.

(a) Decodings

(b) Reconstruction (10k stored)

(c) Reconstruction (50k stored)

Figure 4.3: WW **Decoder examples - Noiseless**. Here we follow the methodology of Fig. 4.1. **(a):** The direct decodings of the WW codes. **(b):** The reconstruction of the outputs of a memory that stores 10.000 patterns. **(c):** The reconstruction of the outputs of a memory that stores 60.000 patterns. Notice how the decodings in (a) are very similar to the original patters (Fig. 4.2), with only a few pixels missing due to the lossiness of the encoding process. As the memory feels up (b) the memory adds noise to cues (Fig. 3.7). Most of the noise is around the active pixels of the digit, but some noisy spots start to appear when the memory becomes fuller (c).

(a) Decodings

(b) Reconstruction (10k stored)

(c) Reconstruction (50k stored)

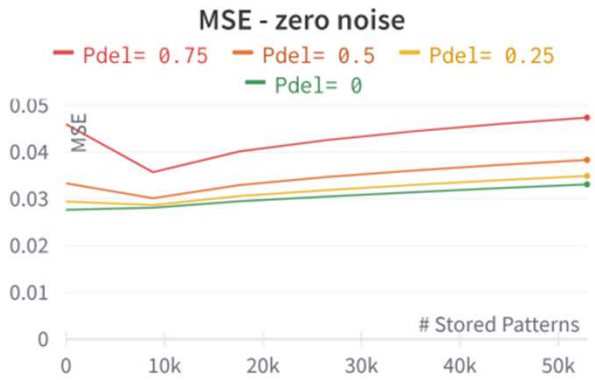
Figure 4.5: WW **Decoder examples - Noisy (type Zero)**. Here we follow the methodology of Fig. 4.1. **(a):** The direct decodings of the WW codes. **(b):** The reconstruction of the outputs of a memory that stores 10.000 patterns. **(c):** The reconstruction of the outputs of a memory that stores 60.000 patterns. Notice the effect of the noise ($P_{del} = 0.75$) on the decodings: 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), However, as the memory gets fuller, retrieval becamos to noisy and the reconstructions become blurry (c).

(a) Decoding

(b) Reconstruction (10k)

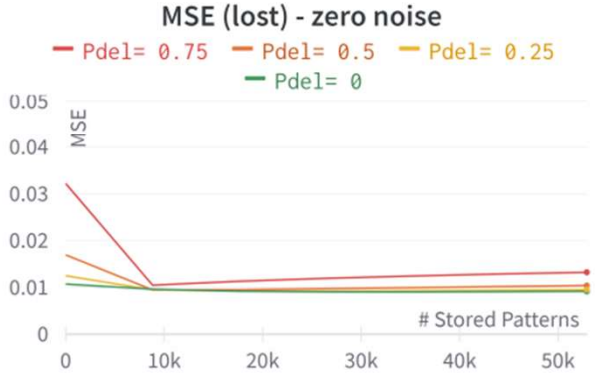
(c) Reconstruction (50k)

Figure 4.7: WW **Decoder examples - Noisy (type One)**. Here we follow the methodology of Fig. 4.1. **(a):** The direct decodings of the WW codes. **(b):** The reconstruction of the outputs of a memory that stores 10.000 patterns. **(c):** The reconstruction of the outputs of a memory that stores 60.000 patterns. Notice the effect of the noise ($P_{rep} = 0.05$) on the decodings: the patterns have some white stops randomly placed around the image (a). The memory is terrible at delaing with the added information, and the retrieval process deletes most of the information of the patterns(b). However, as the memory gets fuller, retrieval becomes better (c).

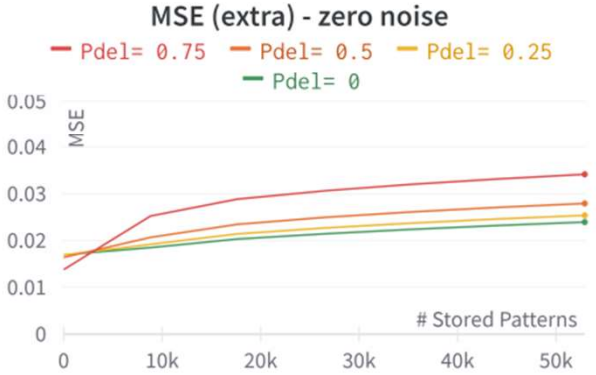


(a) Total MSE

- Memory completes missing information



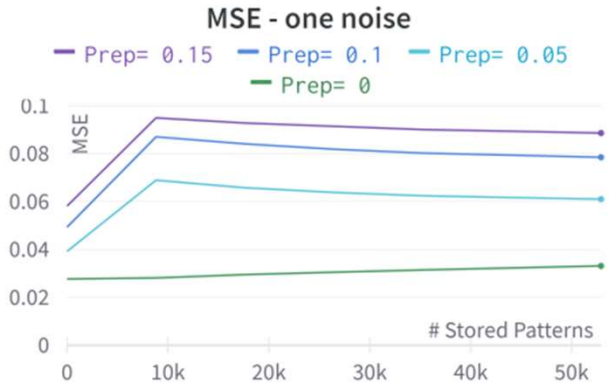
(b) MSE due to lost information



(c) MSE due to extra information

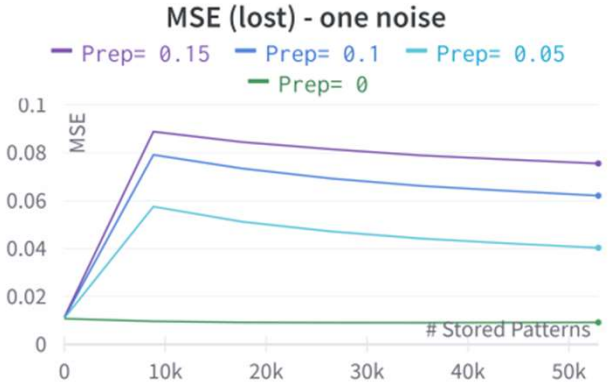
- Memory over-completes

Reconstructions – noisy cues (addition)

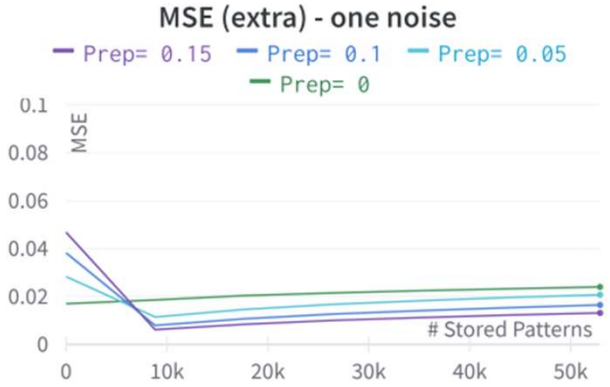


(a) MSE

- Memory **loses** a lot of **information**



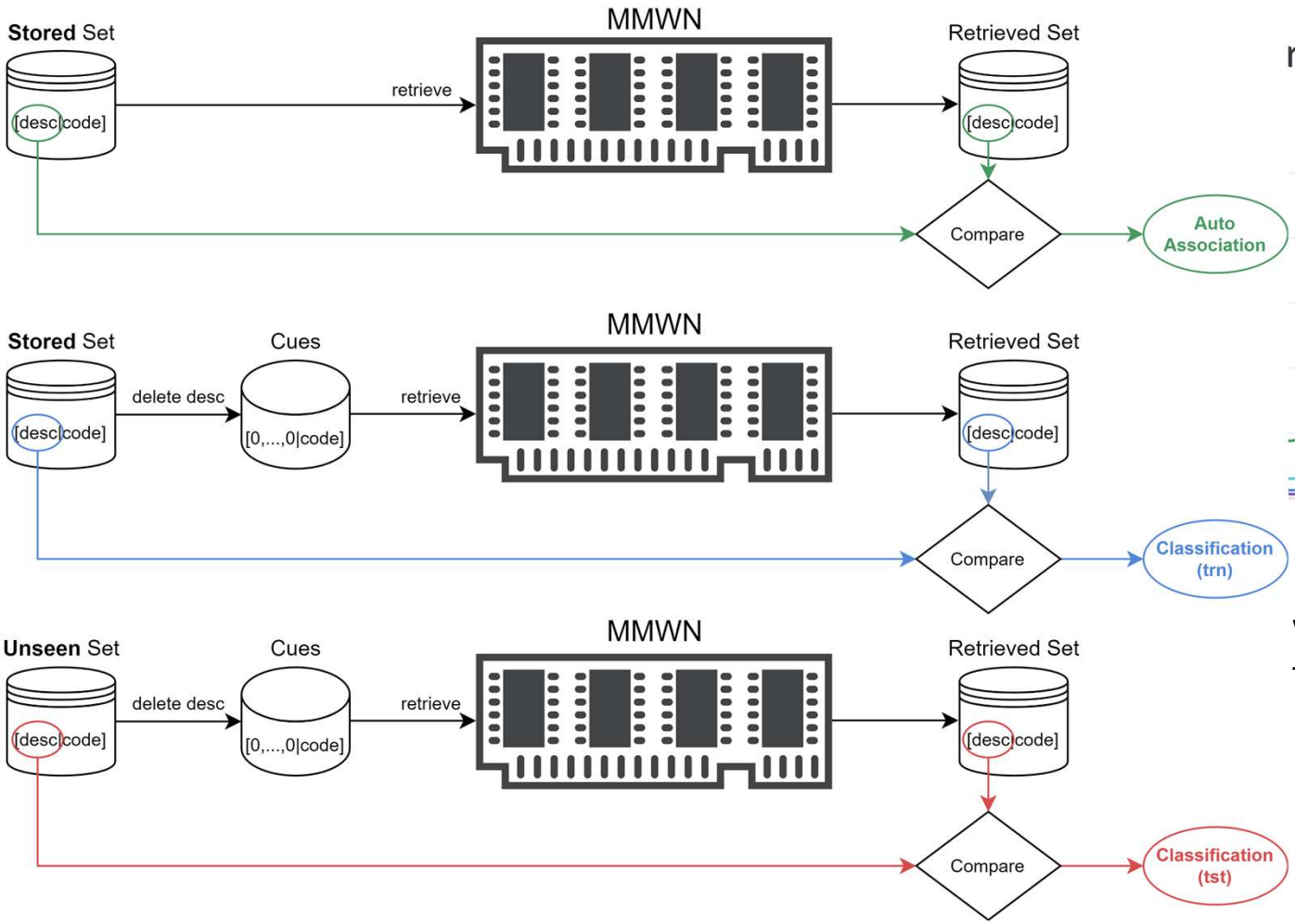
(b) MSE (lost)

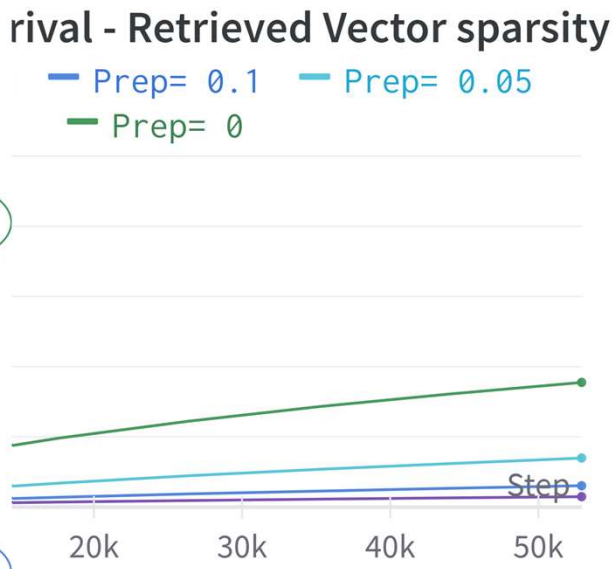


(c) MSE (extra)

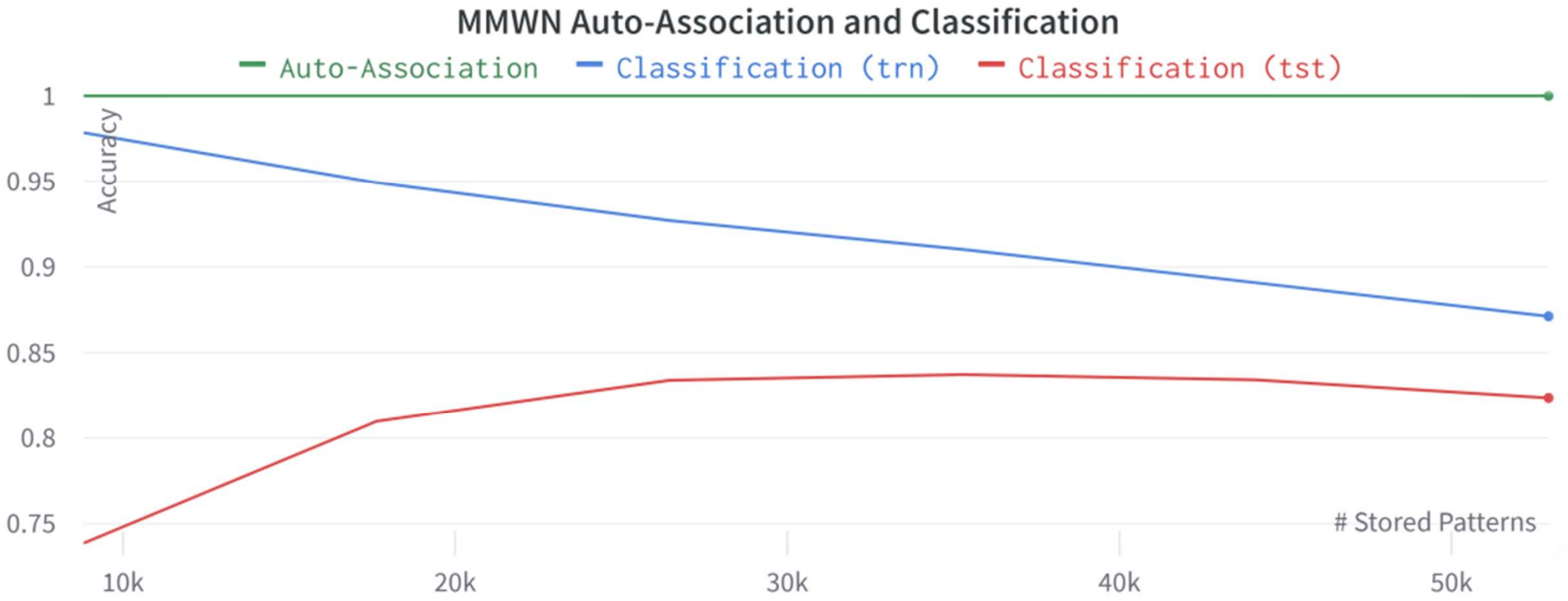
- Patterns are **corrupted**

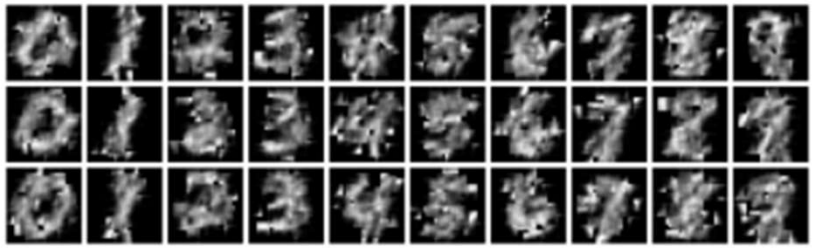
Sparsity - Response to noisy cues



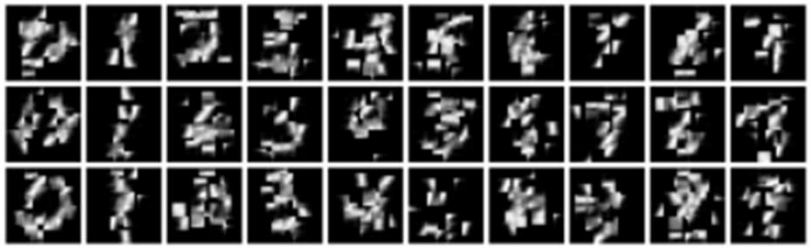


we add to the cue…
the memory deletes.

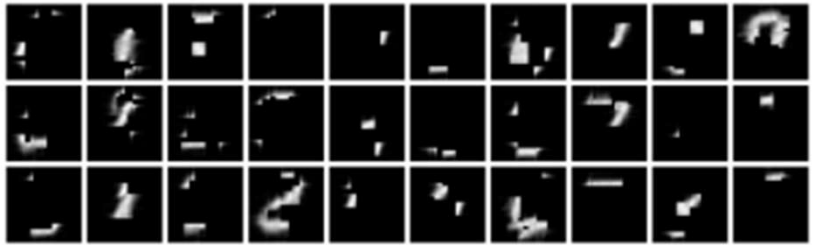

MMWN Auto-Association and Classification
Auto-Association
Classification (trn)
Classification (tst)
1
0.95
0.9
0.85
0.8
0.75
Accuracy
Stored Patterns
10k
20k
30k
40k
50k



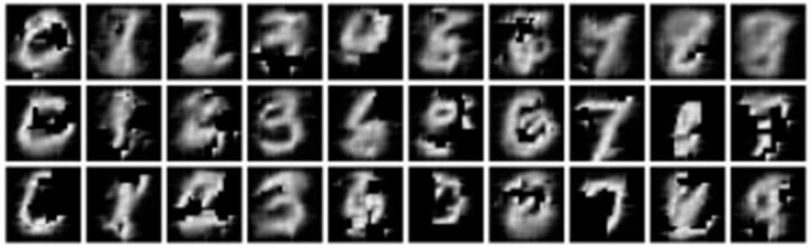
(a) Noiseless Decodings



(b) Noisy Decodings ($P_{del} = 0.75$)



(c) Noiseless Reconstructions



(d) Noisy Reconstructions ($P_{del} = 0.75$)

Figure 4.9: Baseline Generation with the WW decoder. Here we are generating 3 examples for each of the 10 classes of the MNIST dataset. Two experiments are performed: noiseless generation **(left column)** and noisy generation **(righ column)**. For each experiment, we plot the outputs of the WW decoder: without using the WN memory **(top row)**, and using a memory filled with 30.000 codes **(bottom row)**.

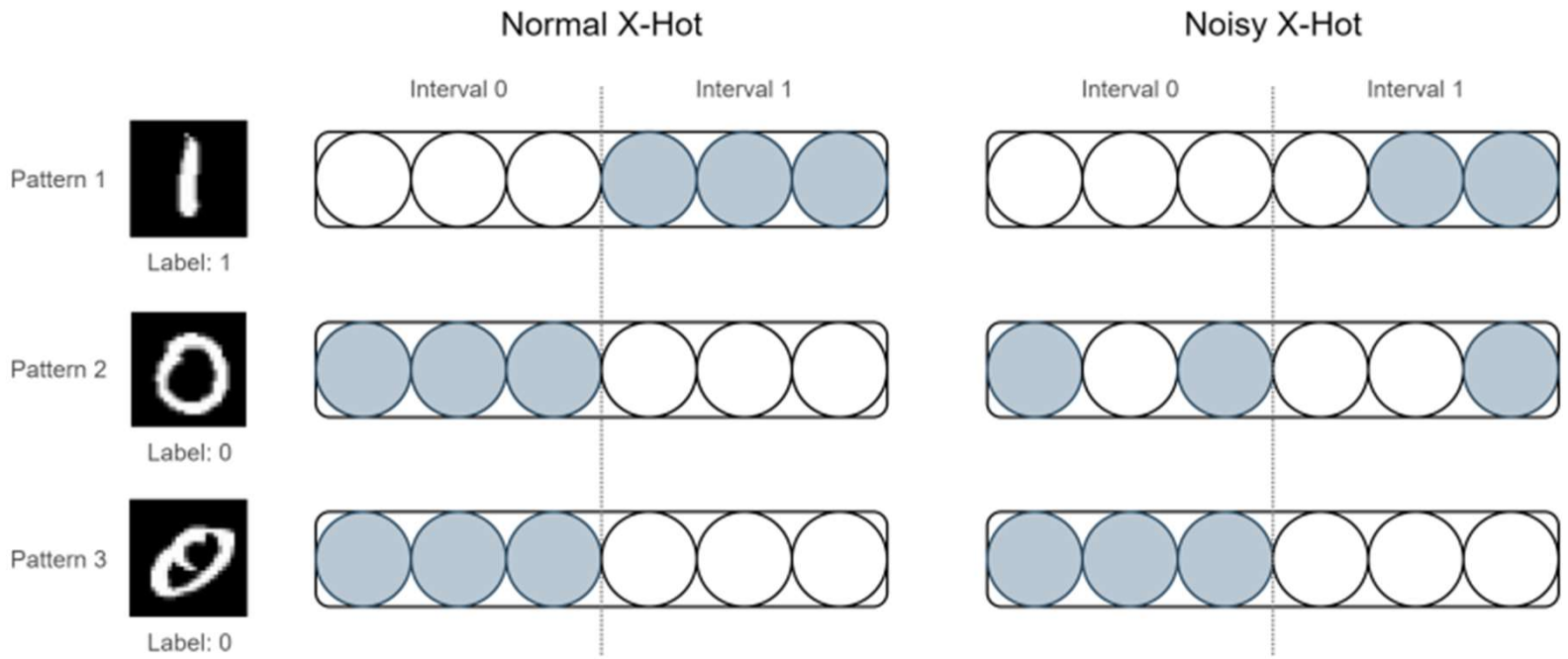
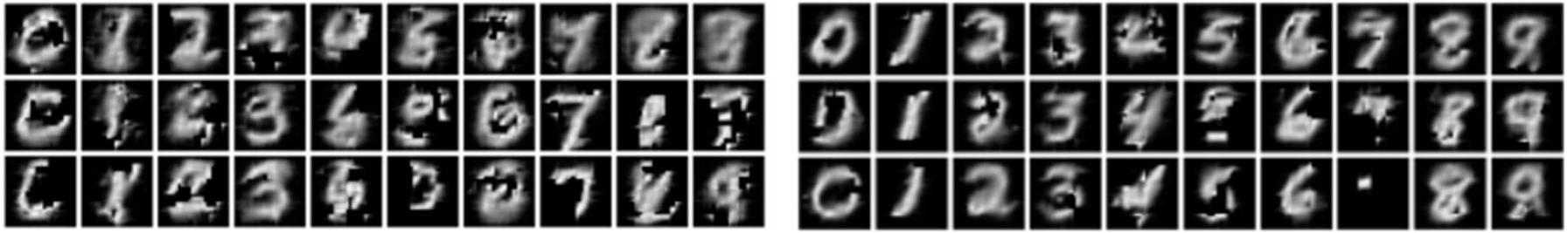


Figure 5.2: Comparison between X-Hot and Noisy X-Hot codes. Consider a dataset of digits with just two classes: 0 and 1. Consider also three distinct patterns from this dataset. Here we encode all three patterns with two distinct encoding prescriptions. **(Left):** A regular X-Hot encoding where the 3 bits assigned to the class are activated with certainty. **(right):** A NXH encoding where the 3 bits in the class interval are activated with $P_{class} = 0.5$, and the rest of the bits are activated with $P_{rest} = 0.1$. Notice that, by chance, both encoding strategies can lead to the same code (bottom-most pattern). Notice also how two patterns with the same label can end up with different NXH encodings (middle and bottom-most pattern).



(a) Baseline Generation with Single Modality WN (Section 4.3.5)

(b) Sampling-based Generation with a MMWN

Figure 5.7: Sampling-based Generation: Single vs Multiple Modality Memory. Here we are generating images by sampling from the class distribution, retrieving the sample in a full memory, and reconstructing the memory's output with the What-Where decoder. **(a):** The memory is a Single-Modality Willshaw Network. **(b):** A Multi-Modal Willshaw Network (MMWN) that stores descriptions and visual codes is used. Please note that in both (a) and (b) we delete bits from the sample ($P_{del} = 0.5$). Using the new modality clearly improves the quality of the generations. However, the memory will, occasionally, corrupt the sample.

(a) Sampling-based Generation with a MMWN

(b) Trial and error sampling-based Generation with a MMWN

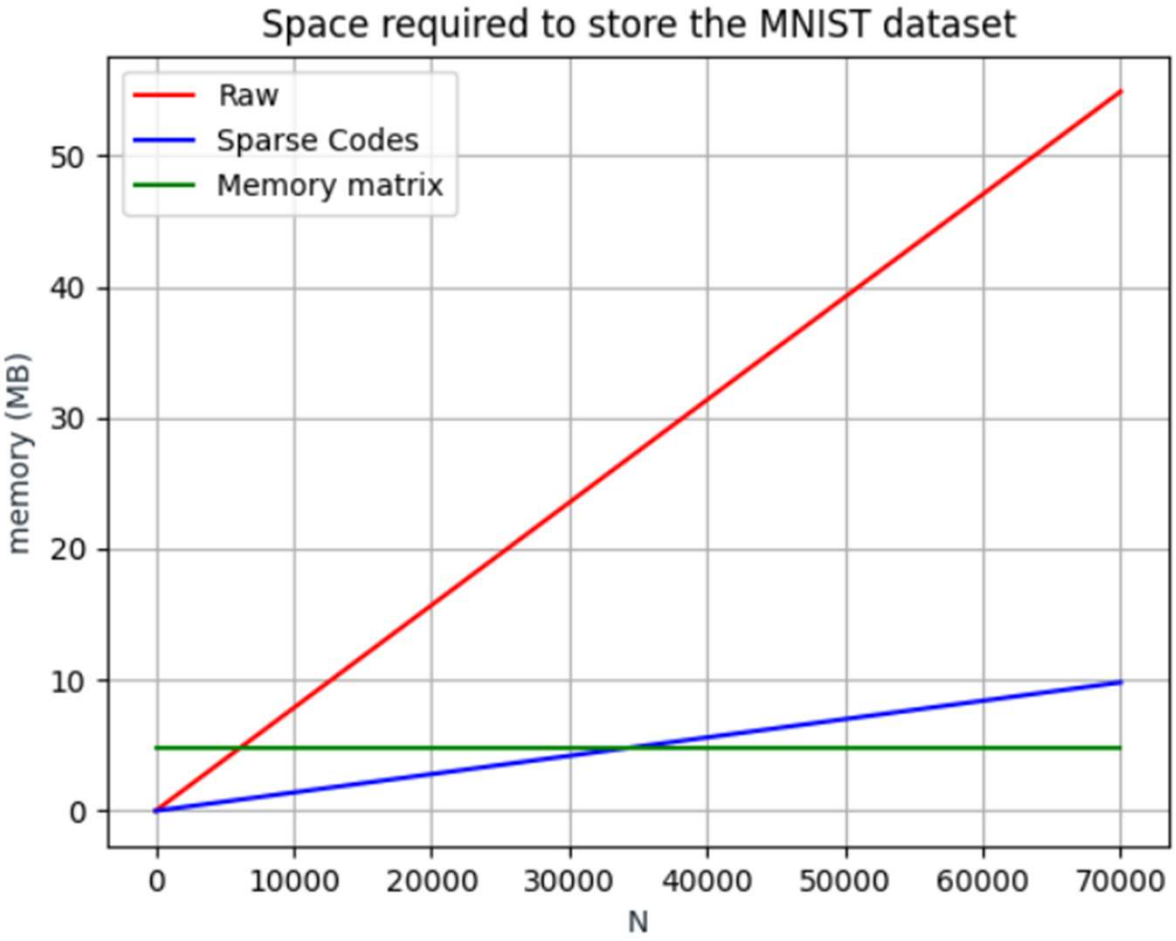
Figure 5.9: Sampling-based Generation with a MMWN: Simple vs Trial-and-error approach. Here we are generating images with the MMWN. First, we create a descriptor of the class and create an artificial code by sampling from the class distribution. Then we provide the artificial desCode to a full MMWN and reconstruct the memory's output with the What-Where decoder. **(a):** Simple generation, for each generation in this plot we used one sample and performed retrieval once. **(b):** Trial and Error Generation. Here we monitor the sparsity of the generations: if the number of bits of the retrieved vector is not within a predefined interval, we discard the generation and try again with a new sample. In this experiment, the generation process took an average of 4.63 attempts per generation.

```python
def iterative_generation(
    MMWN, # trained memory
    desc, # description
    S_init, # initial value for the sparsification function
    S_inc, # increment value for the sparsification function
    accept, # acceptance interval
    max_iter # maximum number of iterations
):
    visual= [0, ..., 0]  # empty visual modality
    cue = desc + visual
    S = S_init # initialize sparsification level

    for _ in range(max_iter):
        gen = MMWN.retrieve(cue)                    # (1) - retrieve
        if gen.visual.num_bits in accept:           # (2) - Assess Sparsity
            break
        else:
            Pdel = 1 - (S / gen.visual.num_bits)
            cue.visual = sparsify(gen, P=Pdel) # (3) - Sparsify
            S = S + S_inc

    return gen
```

Listing 5.1: Iterative Generation Pseudo-Code. The function sparsify(code, P) receives a binary code and stochasticaly transforms the ones in the code into zeros with probability P.

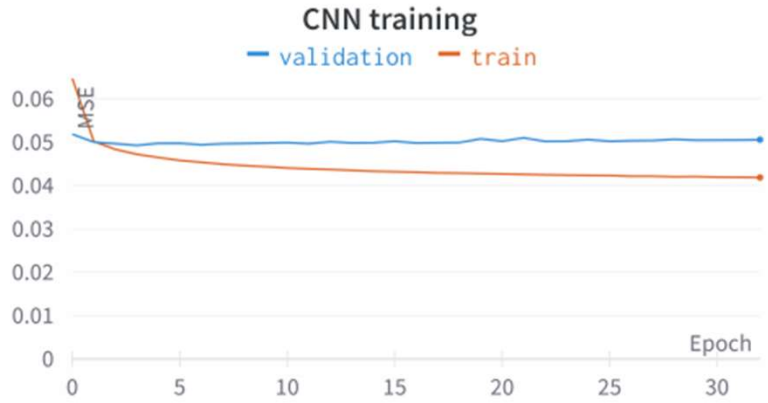

Space required to store the MNIST dataset
Raw
Sparse Codes
Memory matrix
memory (MB)
0
10
20
30
40
50
0
10000
20000
30000
40000
50000
60000
70000
N

(a) Training

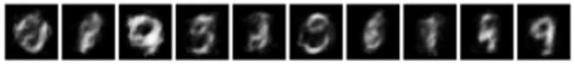
(b) Output

(c) Target

Figure A.6: MLP Decoder Results for WW codes. Here we show the results of an MLP trained with backpropagation to decode the WW sparse codes back into the MNIST images that they represent. The architecture of this network was composed of: an input layer with the size of the WW codes (8820 units), a hidden layer with 6000 units, and an output layer with 784 units (size of the MNIST images). The training was done with the ADAM optimization algorithm [73], without the K-folds method. Hyper-parameters for the training were: learning rate of 0.001, batch size of 32, error measure was the Mean Squared Error, early-stopping with a patience level of 10, and delta of 0.0001. **(a):** The validation and train accuracy of the training process. **(b):** The output of the network for 10 examples from the validation set on the early stopping checkpoint (best validation score). **(c)**: The target outputs for the given examples. Training is short as the network quickly overfits the training data. The training loss keeps decreasing but the validation loss stagnates at around 0.05. The network overfits some patterns and gives a general prototype of the class in other cases. The reconstruction quality is overall poor. This network was the best performing one in a grid search where the number of hidden layers varied between 0 and 3, and the number of hidden units per layer varied between 100 and 6000.



(a) Training



(b) Output

(c) Target

Figure A.7: CNN Decoder Results for WW codes. Here we show the results of a CNN trained with backpropagation to decode the WW sparse codes back into the MNIST images that they represent. The architecture of this network was composed of: an input layer with the size of the WW codes (8820 units), a convolution layer with 20 kernels of size 5 and same-padding, followed by a ReLU activation function and a Max-pooling layer with a 2×2 kernel, and a fully connected output layer with 784 output units (size of the MNIST images). The training was done with the ADAM optimization algorithm [73], without the K-folds method. Hyper-parameters for the training were: learning rate of 0.001, batch size of 32, error measure was the Mean Squared Error, early-stopping with a patience level of 10, and delta of 0.0001. **(a):** The validation and train accuracy of the training process. **(b):** The output of the network for 10 examples from the validation set on the early stopping checkpoint (best validation score). **(c)**: The target outputs for the given examples. Training slightly overfits the training data. The training loss keeps decreasing but the validation loss stagnates at around 0.05 (similar values to the MLP). The network gives a general prototype of the class as reconstructions. The reconstruction quality is overall poor. This network was the best performing one in a grid search where the number of hidden layers varied between 0 and 3, and the number of hidden units per layer varied between 100 and 6000.