

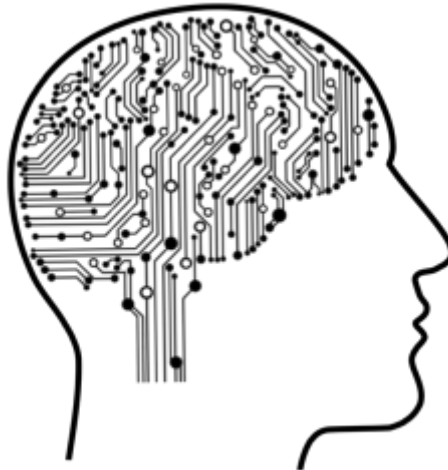


Instituto Politécnico Nacional
Escuela Superior de Computo
Ingeniería en sistemas computacionales



Pattern Recognition

Implementación de Memorias Morfológicas
Autoasociativas.



Profesor: Dra. María Elena Cruz Meza.

Alumno: Cruz Villalba Edwin Bernardo.

Fecha de entrega: 8 de mayo del 2025

Introducción

El reconocimiento de imágenes es una tarea fundamental en inteligencia artificial, con aplicaciones que van desde la identificación de objetos hasta el procesamiento de señales visuales en entornos ruidosos.

Las memorias morfológicas autoasociativas son un modelo de memoria artificial basado en morfología matemática, diseñado para almacenar y recuperar patrones binarios, especialmente en presencia de ruido. Este trabajo documenta la implementación de una memoria morfológica utilizando matrices de máximo W y mínimo M , con el objetivo de evaluar su capacidad para corregir ruido artificial y manejar patrones a mano alzada.

En particular, las **memorias morfológicas autoasociativas**, desarrolladas a partir de los principios de la morfología matemática, destacan por su capacidad para recuperar patrones con ruido aditivo, sustractivo y mixto, superando limitaciones de modelos clásicos como la red de Hopfield. Este proyecto implementa una memoria morfológica autoasociativa, utilizando los modelos **máx** y **mín**, para abordar el problema de reconocimiento de imágenes bajo diferentes niveles de ruido (1%, 25%, 50% y 90%).

Descripción del problema

El problema consiste en analizar el desempeño de una memoria morfológica autoasociativa para el reconocimiento y recuperación de patrones binarios, específicamente dígitos del 0 al 9 representados como imágenes binarias. El análisis se centra en dos escenarios principales: la recuperación de patrones fundamentales (CFP) alterados con ruido artificial y la evaluación con patrones de prueba dibujados a mano alzada sin ruido adicional.

Objetivos

- Implementar una memoria autoasociativa.
- Evaluar la capacidad de la memoria para corregir ruido artificial (aditivo, sustractivo y mixto) en el CFP, aplicado en porcentajes del 1% al 90%
- Determinar si puede reconocer patrones a mano alzada que presentan variaciones estructurales significativas.

Marco teórico

Memorias asociativas

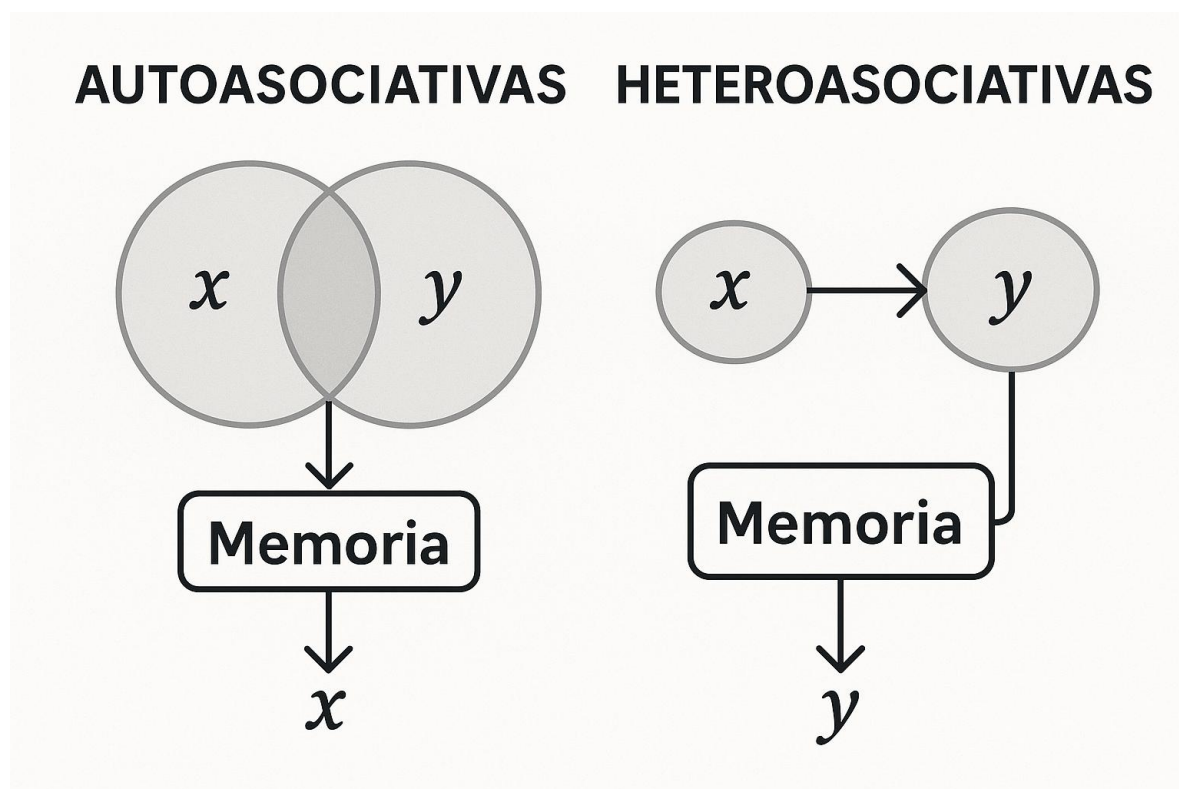
El término “asociación”, de manera general, se refiere a establecer relación entre personas, ideas u objetos. Los organismos superiores poseen la capacidad de establecer una conexión mental entre ideas e imágenes de acuerdo con su relación. Un cerebro orgánico funciona en base a estas asociaciones, de tal manera que relaciona imágenes, sonidos u otros estímulos físicos. [1]

La capacidad de los cerebros orgánicos para realizar estas “asociaciones” ha sido objeto de múltiples investigaciones para desarrollar modelos de redes neuronales artificiales que actúen como memorias asociativas. La diferencia entre una memoria como un dispositivo electrónico digital y

una memoria asociativa consiste en que en la primera los datos que se almacenan pueden recuperarse a través de una dirección determinada. [1]

Las memorias asociativas artificiales imitan la capacidad de un cerebro orgánico de recuperar información completa o casi completa a partir de información parcial. Por tanto, esta clase de memorias poseen la característica de ser tolerantes a errores y distorsiones, siendo capaces de recuperar información inclusive cuando se les presentan datos incompletos. [2]

El propósito fundamental de una memoria asociativa es recuperar patrones completos a partir de patrones de entrada que pueden estar alterados con ruido aditivo, sustractivo o combinado. De acuerdo con esta afirmación, una memoria asociativa M puede formularse como un sistema de entrada y salida [2]



Las memorias asociativas se clasifican en **autoasociativas** y **heteroasociativas**. Una memoria autoasociativa es aquella que recupera un patrón y , previamente almacenado, cuando se le presenta un patrón similar y ; de tal manera que el patrón de entrada x pertenece al mismo conjunto que el patrón de salida y . Por otra parte, una memoria heteroasociativa recupera un patrón y cuando se le presenta un patrón de entrada x tal que los patrones de entrada y salida pertenecen a dos conjuntos diferentes. [1]

Por su naturaleza, el problema inherente al funcionamiento de las memorias asociativas se escinde en dos fases claramente distinguibles: [2]

1. Fase de aprendizaje (generación)

2. Fase de recuperación (operación)

La memoria asociativa **M** se representa mediante una matriz cuya componente ij -ésima es m_{ij}

$$\mathbf{x} \rightarrow \boxed{\mathbf{M}} \rightarrow \mathbf{y}$$

Donde el patrón de entrada estará presentado por un vector columna denotado por \mathbf{x} y el patrón de salida, por el vector columna denotado por \mathbf{y} . La notación para una asociación es similar a la de una pareja ordenada (\mathbf{x}, \mathbf{y}) . [3]

A un patrón de entrada \mathbf{x}^1 le corresponderá un patrón de salida \mathbf{y}^1 , y ambos formarán la asociación $(\mathbf{x}^1, \mathbf{y}^1)$; del mismo modo, para un número entero positivo k específico, la asociación correspondiente será $(\mathbf{x}^k, \mathbf{y}^k)$. [3]

La matriz **M** se genera a partir de un conjunto finito de asociaciones conocidas de antemano: éste es el **conjunto fundamental de asociaciones**. Se denota por p la cardinalidad del conjunto fundamental de asociaciones. [3]

Si μ es un índice, el conjunto fundamental se representa de la siguiente manera: [3]

$$\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu = 1, 2, \dots, p\}$$

A los patrones que conforman las asociaciones del conjunto fundamental, se les llama **patrones fundamentales**. [3]

Para diferenciar un patrón alterado del correspondiente patrón fundamental, usaremos la tilde en la parte superior; así, el patrón $\tilde{\mathbf{x}}^k$ es una versión alterada del patrón fundamental \mathbf{x}^k , y el tipo de alteración que representa $\tilde{\mathbf{x}}^k$. [3]

Si al presentarle a la memoria **M** un patrón alterado $\tilde{\mathbf{x}}^\omega$ como entrada ($\omega \in \{1, 2, \dots, p\}$), **M** responde con el correspondiente patrón fundamental de salida \mathbf{y}^ω , se dice que la recuperación es perfecta. [3]

Una memoria perfecta es aquella que realiza recuperaciones perfectas para todos los patrones fundamentales. [3]

Entonces si \mathbf{y}^3 es un patrón fundamental, entonces $\tilde{\mathbf{y}}^3$ representa una versión alterada de \mathbf{y}^3 . [3]

Fase de aprendizaje

Encontrar los operadores adecuados y una manera de generar una matriz **M** que almacene las p asociaciones del conjunto fundamental.

$$\{(\mathbf{x}^1, \mathbf{y}^1), (\mathbf{x}^2, \mathbf{y}^2), \dots, (\mathbf{x}^p, \mathbf{y}^p)\} \text{ donde } \mathbf{x}^\mu \in A^n$$

Si $\exists \mu \in \{1, 2, \dots, p\}$ tal que $\mathbf{x}^\mu \neq \mathbf{y}^\mu$ la memoria es **heteroasociativa**.

Si $m = n$ y $\mathbf{x}^\mu = \mathbf{y}^\mu \forall \mu \in \{1, 2, \dots, p\}$ la memoria es **autoasociativa**. [3]

Fase de recuperación

Hallar los operadores adecuados y las condiciones suficientes para obtener el patrón fundamental de salida \mathbf{y}^μ , cuando se opera la memoria \mathbf{M} con el patrón fundamental de entrada \mathbf{x}^μ ; lo anterior para todos los elementos del conjunto fundamental y para ambos modos: autoasociativo y heteroasociativo. Exhibir y caracterizar, además, el ruido que puede soportar la memoria en el patrón de entrada $\tilde{\mathbf{x}}^\omega$, para entregar una salida perfecta \mathbf{y}^ω [3].

Código

La implementación se realizó en Python con NumPy y Matplotlib, y las pruebas

La memoria morfológica fue implementada como una clase MorphologicalMemory:

```
class MorphologicalMemory:
    def __init__(self, n):
        self.n = n
        self.W = None
        self.M = None
```

Funcion de entrenamiento:

```
def train(self, patterns):
    p = len(patterns)
    self.W = np.zeros((self.n, self.n))
    self.M = np.zeros((self.n, self.n))

    for i in range(self.n):
        for j in range(self.n):
            self.W[i, j] = max([patterns[mu][i] - patterns[mu][j] for mu in range(p)])
            self.M[i, j] = min([patterns[mu][i] - patterns[mu][j] for mu in range(p)])
```

El entrenamiento genera las matrices W y M mediante las siguientes definiciones:

$$M = \bigvee_{\mu=1}^p [x^\mu \Delta (-x^\mu)^t] = [m_{ij}]_{m \times n}$$
$$W = \bigwedge_{\mu=1}^p [x^\mu \nabla (-x^\mu)^t] = [w_{ij}]_{n \times n}$$

Funcion de recuperación con memoria min.

```
def recall_min(self, pattern):
    pattern = pattern.flatten()
    y = np.zeros(self.n)
    for i in range(self.n):
        y[i] = max([self.M[i, j] + pattern[j] for j in range(self.n)])
    y = (y >= 0.5).astype(int)
    height = int(np.sqrt(self.n))
    return y.reshape(height, height)
```

y se implementó una función para agregar ruido artificial (aditivo, sustractivo, o mixto) en porcentajes del 1% al 90%.

```
def add_noise(pattern, percentage, noise_type='additive'):
    noisy = pattern.copy().flatten()
    n_pixels = len(noisy)
    n_change = int(n_pixels * percentage / 100)
    indices = np.random.choice(n_pixels, n_change, replace=False)

    if noise_type == 'additive':
        noisy[indices] = np.random.choice([0, 1], n_change)
    elif noise_type == 'subtractive':
        noisy[indices] = 0
    elif noise_type == 'mixed':
        half = n_change // 2
        noisy[indices[:half]] = np.random.choice([0, 1], half)
        noisy[indices[half:]] = 0

    return noisy.reshape(pattern.shape)
```

La parte inicial de entrenamiento empezamos la lectura y preparación de las imágenes de entrenamiento (CFP) que se nos fue proporcionado y las imágenes de prueba (Imágenes a mano alzada), haciendo uso de las imágenes de números del 0 al 9 las cuales serán tratados como vectores unidimensionales ya normalizados.

```
def main():
    # Parámetros configurables
    training_folder = "train_chica/" # Carpeta con imágenes de entrenamiento (CFP)
    testing_folder = "test/" # Carpeta con imágenes de prueba

    # Fase 1: Lectura y preparación de imágenes de entrenamiento (CFP)
    train_images = []
    train_filenames = []
    for filename in sorted(os.listdir(training_folder)):
        if filename.endswith(".bmp"):
            filepath = os.path.join(training_folder, filename)
            img = Image.open(filepath).convert('L')
            img_array = np.array(img)
            train_images.append(img_array)
            train_filenames.append(filename)

    train_sets = {"numeros": train_images}
    for category in train_sets:
        train_sets[category] = [binarize_image(img) for img in train_sets[category]]

    # Validación de tamaños de entrenamiento
    for category in train_sets:
        shapes = [p.shape for p in train_sets[category]]
        if len(set(shapes)) > 1:
            raise ValueError(f"Los patrones en '{category}' tienen diferentes tamaños: {shapes}")

    # Fase 2: Lectura y preparación de imágenes de prueba
    test_images = []
    test_filenames = []
    for filename in sorted(os.listdir(testing_folder)):
        if filename.endswith(".bmp"):
            filepath = os.path.join(testing_folder, filename)
            img = Image.open(filepath).convert('L')
            img_array = np.array(img)
            test_images.append(img_array)
            test_filenames.append(filename)

    test_sets = {"numeros": test_images}
    for category in test_sets:
        test_sets[category] = [binarize_image(img) for img in test_sets[category]]
```

La fase de entrenamiento será instanciar MorphologicalMemory y llamar los métodos de entrenamiento administrando la CFP.

```
# Entrenamiento de la memoria con el CFP
category = "numeros"
train_patterns = train_sets[category]
n = train_patterns[0].size
height = train_patterns[0].shape[0]
patterns_flat = [p.flatten() for p in train_patterns]
memory = MorphologicalMemory(n=n)
memory.train(patterns_flat)
```

Las pruebas se estructuraron en dos fases:

Fase 1: Recuperación del CFP con Ruido:

Se aplicaron diferentes tipos y niveles de ruido a las imágenes del CFP para evaluar la capacidad de recuperación. Esto respondió a la inquietud inicial de verificar el funcionamiento con ruido artificial.

```
# Fase 1: Prueba de recuperación con el CFP original con ruido
print("\nFase 1: Recuperación del CFP con ruido")
noise_types = ['additive', 'subtractive', 'mixed']
percentages = [1, 25, 50, 90]
for idx, (train_image, filename) in enumerate(zip(train_patterns, train_filenames)):
    print(f"Probando patrón {filename}:")
    for noise_type in noise_types:
        for percentage in percentages:
            print(f"  Ruido {noise_type}, {percentage}%:")
            noisy_image = add_noise(train_image, percentage, noise_type)
            recovered_max = memory.recall_max(noisy_image)
            recovered_min = memory.recall_min(noisy_image)
            plot_patterns(noisy_image, recovered_max, recovered_min, noise_type,
                          percentage, category, idx, 1, filename)
```

Fase 2: Prueba con Patrones sin Ruido

Se utilizaron imágenes a mano alzada sin ruido, para analizar el comportamiento frente a variaciones estructurales, que fue otro punto de duda al notar que no se recuperaban correctamente.

```
# Fase 2: Prueba con patrones de prueba sin ruido
print("\nFase 2: Prueba con patrones de prueba sin ruido")
for idx, (test_image, test_filename) in enumerate(zip(test_sets[category], test_filenames)):
    print(f"Probando patrón de prueba {test_filename}:")
    recovered_max = memory.recall_max(test_image)
    recovered_min = memory.recall_min(test_image)
    plot_patterns(test_image, recovered_max, recovered_min, None, None, category, idx, 2, test_filename)
```

Pruebas y resultados

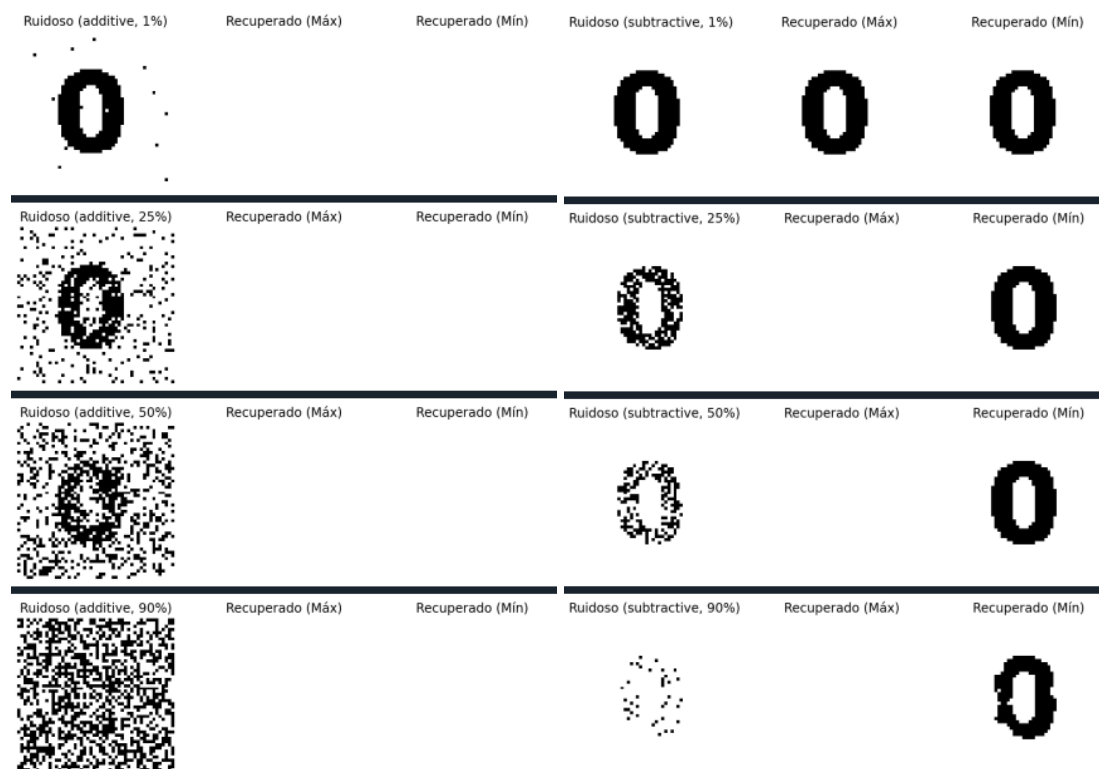
3.1 Fase 1: Recuperación del CFP con Ruido

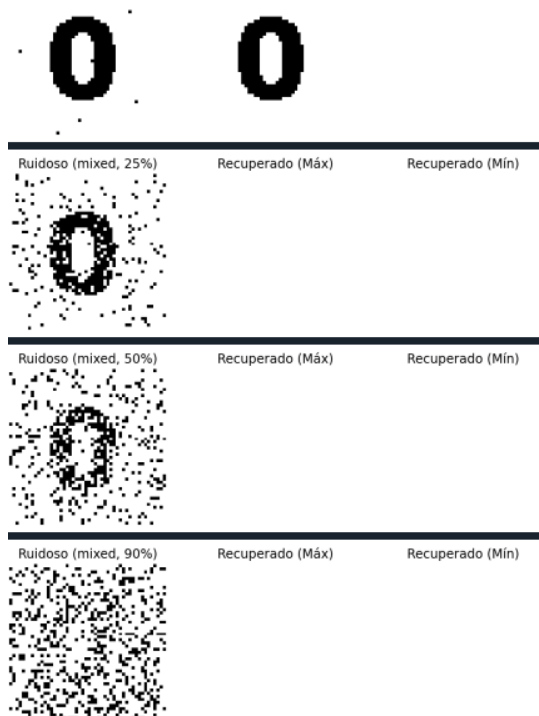
La memoria **min** recuperó con éxito los patrones originales cuando el ruido es substractivo.

En el caso de el ruido aditivo y mixto no hubo éxito alguno excepto que la memoria **max** logro obtener algo cuando el ruido es de un 1%.

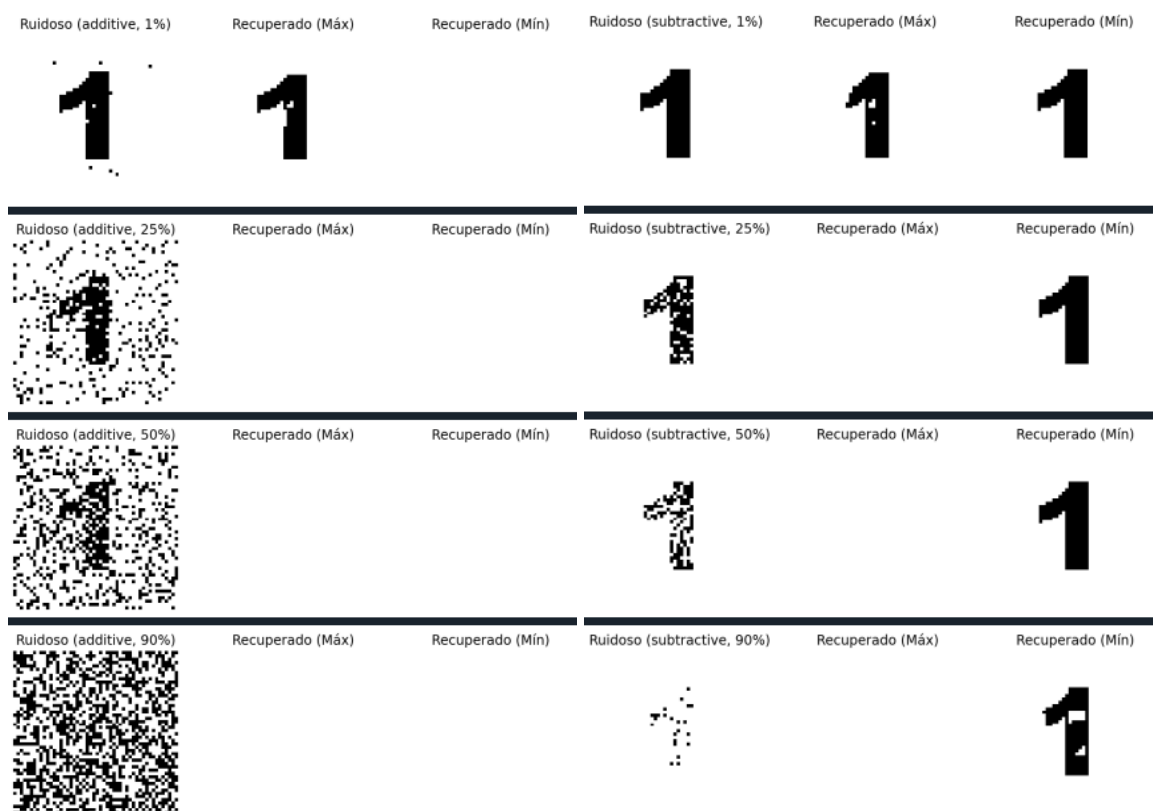
Las siguientes imágenes reflejan esta limitación, mostrando recuperaciones que no preservan la forma original.













Probando patrón 0:









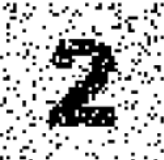

























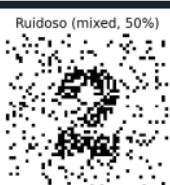

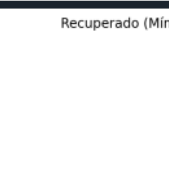



Probando patrón 1:



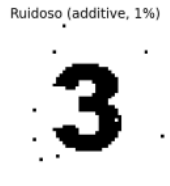


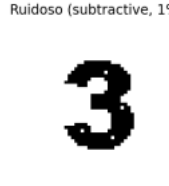
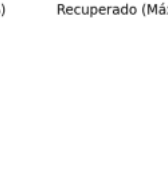
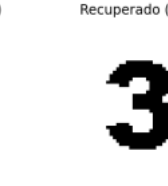
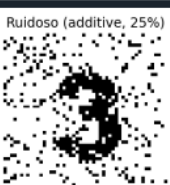


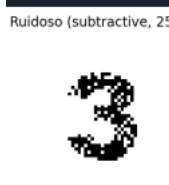
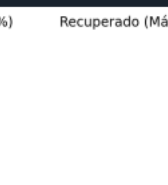

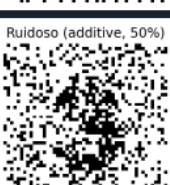
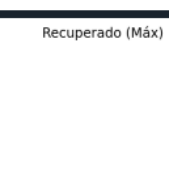

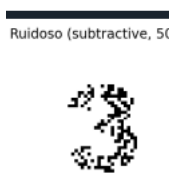
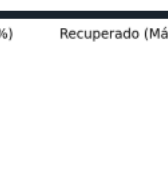


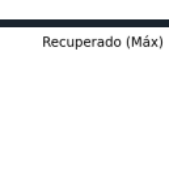
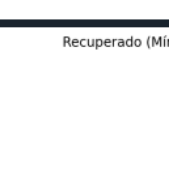
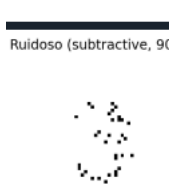
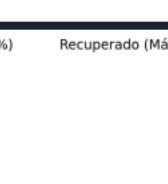
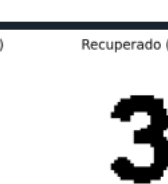
Ruidoso (mixed, 1%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 25%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 50%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 90%)	Recuperado (Máx)	Recuperado (Mín)
		



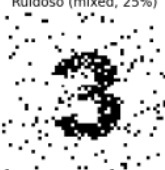


Probando patrón 2:

Ruidoso (additive, 1%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 1%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 25%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 25%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 50%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 50%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 90%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 90%)	Recuperado (Máx)	Recuperado (Mín)
					















Ruidoso (mixed, 1%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 25%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 50%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 90%)	Recuperado (Máx)	Recuperado (Mín)
		




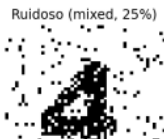
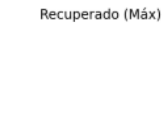
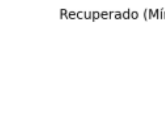
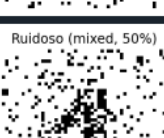
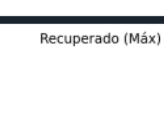
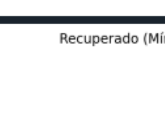

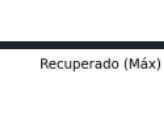
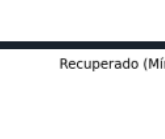
Probando patrón 3:

Ruidoso (additive, 1%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 1%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 25%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 25%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 50%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 50%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 90%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 90%)	Recuperado (Máx)	Recuperado (Mín)
					

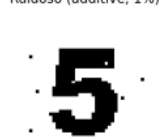






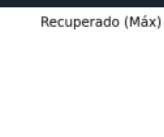
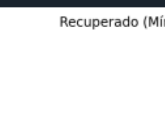
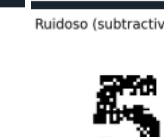
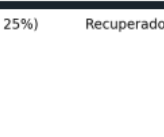


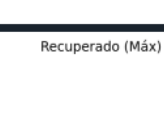
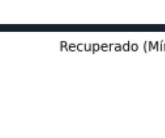
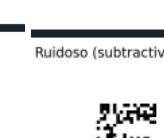
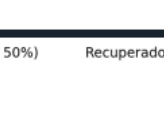



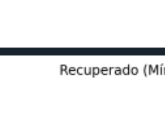
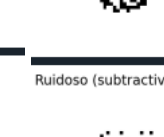
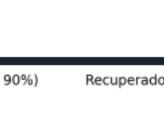

Ruidoso (mixed, 1%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 25%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 50%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 90%)	Recuperado (Máx)	Recuperado (Mín)
		



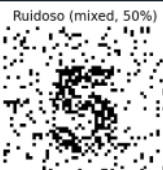

Probando patrón 4:

Ruidoso (additive, 1%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 1%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 25%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 25%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 50%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 50%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 90%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 90%)	Recuperado (Máx)	Recuperado (Mín)
					

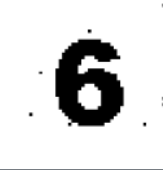




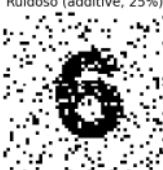


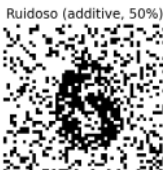



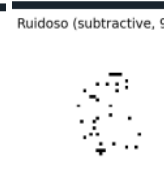

Ruidoso (mixed, 1%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 25%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 50%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 90%)	Recuperado (Máx)	Recuperado (Mín)
		


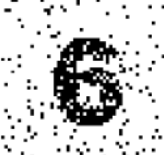


Probando patrón 5:

Ruidoso (additive, 1%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 1%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 25%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 25%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 50%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 50%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 90%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 90%)	Recuperado (Máx)	Recuperado (Mín)
					

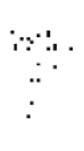

Ruidoso (mixed, 1%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 25%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 50%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 90%)	Recuperado (Máx)	Recuperado (Mín)
		




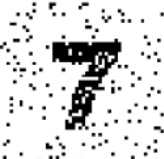








Probando patrón 6:

Ruidoso (additive, 1%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 1%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 25%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 25%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 50%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 50%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 90%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 90%)	Recuperado (Máx)	Recuperado (Mín)
					

























Ruidoso (mixed, 1%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 25%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 50%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 90%)	Recuperado (Máx)	Recuperado (Mín)
		






Probando patrón 7:

Ruidoso (additive, 1%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 1%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 25%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 25%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 50%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 50%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 90%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 90%)	Recuperado (Máx)	Recuperado (Mín)
					















Ruidoso (mixed, 1%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 25%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 50%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 90%)	Recuperado (Máx)	Recuperado (Mín)
		

Probando patrón 8:

Ruidoso (additive, 1%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 1%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 25%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 25%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 50%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 50%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 90%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 90%)	Recuperado (Máx)	Recuperado (Mín)
					

Ruidoso (mixed, 1%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 25%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 50%)	Recuperado (Máx)	Recuperado (Mín)
		
Ruidoso (mixed, 90%)	Recuperado (Máx)	Recuperado (Mín)
		

Probando patrón 9:

Ruidoso (additive, 1%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 1%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 25%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 25%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 50%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 50%)	Recuperado (Máx)	Recuperado (Mín)
					
Ruidoso (additive, 90%)	Recuperado (Máx)	Recuperado (Mín)	Ruidoso (subtractive, 90%)	Recuperado (Máx)	Recuperado (Mín)
					

Ruidoso (mixed, 1%)



Recuperado (Máx)



Recuperado (Mín)

Ruidoso (mixed, 25%)



Recuperado (Máx)

Recuperado (Mín)

Ruidoso (mixed, 50%)



Recuperado (Máx)

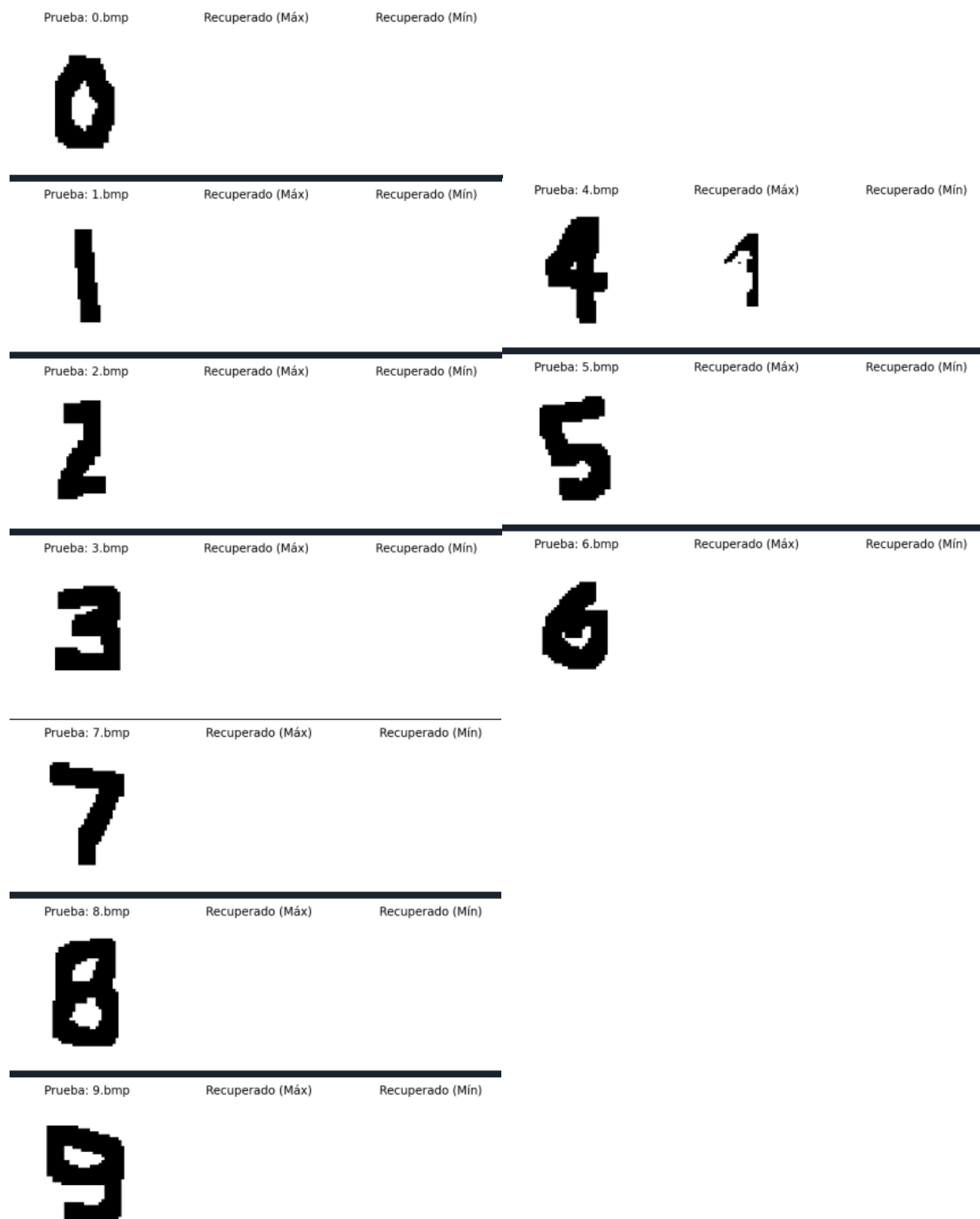
Recuperado (Mín)

Ruidoso (mixed, 90%)



3.2 Fase 2: Prueba con Patrones sin Ruido

Las imágenes a mano alzada no fueron reconocidas correctamente. Esto se atribuye a que las memorias morfológicas están diseñadas para ruido artificial, no para variaciones estructurales, como se planteó inicialmente al notar que no regresaba el patrón original con estas imágenes.



Existe una ligera recuperación del número 4 pero no fue del todo exitosa. Esto confirma que las memorias morfológicas son efectivas para corregir ruido aditivo y sustractivo, como se esperaba según la teoría [3].

Conclusiones

Una inquietud inicial surgió al observar que la memoria funcionaba bien con ruido artificial, pero no con imágenes dibujadas a mano, lo que motivó un análisis detallado de sus limitaciones.

A pesar de su relevancia, las memorias asociativas, y en especial las morfológicas, ocupan un nicho especializado en la literatura de inteligencia artificial, opacadas por el auge del aprendizaje profundo (deep learning). Sin embargo, los trabajos de investigadores del Instituto Politécnico Nacional (IPN) [3], como Yáñez-Márquez han demostrado su valor en aplicaciones específicas, gracias a su robustez y capacidad de recuperación perfecta.

Este proyecto explora la implementación de dichas memorias, evaluando su desempeño frente a ruido y destacando su pertinencia en un campo dominado por enfoques más generalizados.

Referencias

- [1] J. C. V. Navarro, «INAOE,» Septiembre 2007. [En línea]. Available: <https://inaoe.repositorioinstitucional.mx/jspui/bitstream/1009/668/1/ValdiviezoNJC.pdf>. [Último acceso: 1 MAYO 2025].
- [2] C. Y. Márquez, «scielo,» 20 Marzo 2002. [En línea]. Available: <https://www.scielo.org.mx/pdf/cys/v6n4/v6n4a7.pdf>. [Último acceso: 1 Mayo 2025].
- [3] J. L. D. d. L. S. Cornelio Yáñez Márquez, «Memorias Morfológicas Autoasociativas,» n° 58, p. 22, 2001.