Estudio Comparativo de los Modelos de Memoria LAM, BAM y Hopfield para el Reconocimiento de Letras del Alfabeto Latino bajo Ruido y Variación de Funciones de Activación

M. A. Pérez Ávila[1] y A. M. Sánchez Serratos[2]
Departamento de Computación,
Instituto Tecnológico y de Estudios Superiores de Monterrey
Ciudad de México, México

[1]A01369908@tec.mx
[2]A01771843@tec.mx

Resumen **- Este trabajo presenta un análisis experimental de redes neuronales asociativas Hopfield, LAM y BAM, evaluando el impacto de la función de activación y la naturaleza de las perturbaciones en la capacidad de recuperación. Los resultados muestran que Hopfield y BAM con activación simétrica, y LAM con activación asimétrica, alcanzan alta robustez frente a ruido inducido, mientras que todas las arquitecturas presentan vulnerabilidad ante perturbaciones externas que alteran la estructura de los patrones. Se observó además que la saturación inherente a la dinámica de cada red puede limitar la diferenciación de patrones. Estos hallazgos proporcionan bases sólidas para futuras investigaciones orientadas a mejorar la generalización y la eficiencia de las memorias asociativas.**

Índice de Términos **- BAM, función de activación, Hopfield, LAM, memoria asociativa, reconocimiento de patrones, redes neuronales.**

I. INTRODUCCIÓN

El reconocimiento de patrones constituye una capacidad fundamental en campos interdisciplinarios como la visión por computadora, los sistemas de seguridad biométrica y la robótica autónoma. En estos dominios, la capacidad de identificar y clasificar información a partir de datos imperfectos o corruptos resulta esencial para aplicaciones prácticas. Las memorias asociativas (MA), como paradigma dentro de las redes neuronales, emergen como herramientas computacionales diseñadas específicamente para abordar este desafío mediante el almacenamiento y recuperación eficiente de patrones, incluso ante entradas parciales o distorsionadas.

Tres arquitecturas destacadas en este ámbito son la Memoria Asociativa Lineal (LAM), la Memoria Asociativa Bidireccional (BAM) y la Red de Hopfield. La LAM se caracteriza por su diseño basado en la regla de Hebb, donde las asociaciones entre patrones se almacenan mediante la acumulación de correlaciones entre entradas y salidas. La BAM introduce capacidades de asociación bidireccional entre dos dominios distintos de patrones, mientras que la Red de Hopfield emplea dinámicas no lineales y recurrentes para alcanzar estados estables que corresponden a los patrones memorizados.

A pesar de su base teórica bien establecida, el desempeño práctico de estos modelos se ve afectado significativamente por dos factores críticos: la presencia de ruido en los datos de entrada y la selección de la función de activación neuronal. El ruido, inherente a procesos de captura y transmisión de datos en escenarios reales, compromete la capacidad de recuperación de las redes. Paralelamente, la función de activación es el núcleo que dicta la dinámica de la red. Mientras que las funciones tradicionales como las funciones escalón simétrico y asimétrico son omnipresentes en la literatura clásica, su impacto en las MA no ha sido suficientemente explorado de manera comparativa.

La literatura actual carece de un análisis comparativo sistemático que evalúe de manera conjunta la eficacia de estos tres modelos bajo la influencia combinada de estos factores. La mayoría de los estudios se centran en un solo modelo o no consideran el impacto de variar la función de activación más allá de la elección estándar.

Este trabajo busca llenar este vacío mediante la realización de un estudio comparativo. El objetivo principal es evaluar y comparar el rendimiento de las redes LAM, BAM y Hopfield en la tarea de reconocimiento y recuperación de letras del alfabeto latino, representadas como patrones binarios. La evaluación se enfoca en dos dimensiones críticas: 1) Robustez al Ruido, donde se cuantifica la capacidad de cada modelo para recuperar patrones correctamente a medida que se

introducen errores de bits aleatorios de creciente cantidad en los vectores de entrada, y 2) Variación de la Función de Activación, donde se analiza la sensibilidad del desempeño de cada modelo al emplear diferentes funciones de activación.

II. MARCO TEÓRICO

A. *Redes Neuronales*

Las redes neuronales artificiales (RNA) constituyen un paradigma de computación inspirado en la estructura y funcionamiento de los sistemas neuronales biológicos. Estos sistemas están diseñados para procesar información a través de la interconexión de unidades computacionales elementales denominadas neuronas artificiales, las cuales colaboran para resolver tareas complejas de aprendizaje automático. La arquitectura fundamental de una RNA se organiza en capas, comprendiendo una capa de entrada, una o múltiples capas ocultas y una capa de salida, donde cada capa contiene un conjunto de neuronas que transforman progresivamente los datos de entrada [1].

El funcionamiento fundamental de una RNA se basa en el procesamiento secuencial de información a través de múltiples capas. Cada neurona recibe señales de entrada, las procesa mediante una combinación lineal ponderada seguida de una transformación no lineal, y genera una señal de salida que se propaga a las neuronas subsiguientes. Matemáticamente, la operación de una neurona individual en la capa j se expresa como:

$$y_j = \varphi\left(\sum_{i=1}^{n} w_{ij} x_i + b_j\right) \tag{1}$$

donde las variables x_i representan las señales de entrada, los parámetros w_{ij} corresponden a los pesos sinápticos que determinan la importancia relativa de cada conexión entre la neurona i de la capa anterior y la neurona j de la capa actual, el término b_j constituye el sesgo que ajusta el umbral de activación de la neurona j, y la función φ aplica una transformación no lineal que introduce la capacidad de modelar relaciones complejas en los datos.

El proceso de aprendizaje se realiza predominantemente mediante el algoritmo de retropropagación, que ajusta iterativamente los pesos y sesgos para minimizar una función de error que mide la discrepancia entre las predicciones de la red y los valores reales. Este algoritmo calcula el gradiente del error con respecto a cada parámetro mediante la regla de la cadena, permitiendo actualizaciones proporcionales que reducen progresivamente el error de predicción. La regla general de actualización de parámetros sigue la formulación del descenso de gradiente:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \cdot \nabla_\theta \mathcal{L}(\theta^{(t)}) \tag{2}$$

donde θ representa el conjunto de parámetros (pesos w_{ij} y sesgos w_{ij}), η es la tasa de aprendizaje que controla el tamaño del paso de optimización, y $\nabla_\theta \mathcal{L}$ denota el gradiente de la función de pérdida $\mathcal{L}$ con respecto a los parámetros.

Cabe destacar que, aunque la retropropagación es el método más ampliamente utilizado, existen alternativas como los algoritmos evolutivos y el aprendizaje por refuerzo, cada uno con aplicaciones específicas en distintos contextos de redes neuronales. Particularmente, para el caso específico de las memorias asociativas como LAM, BAM y Hopfield, el aprendizaje se basa típicamente en reglas hebbianas o métodos de pseudo-inversa matricial, lo que las distingue de las redes que utilizan retropropagación. Esta diferencia es crucial para comprender el comportamiento y las limitaciones de estos modelos en tareas de recuperación de patrones bajo condiciones de ruido.

B. *Memorias Asociativas*

Las memorias asociativas representan una categoría fundamental dentro del campo de las redes neuronales artificiales, caracterizadas por su capacidad única para almacenar y recuperar información mediante principios de asociación directa entre patrones. A diferencia de las redes neuronales de retroalimentación convencionales que requieren procesos iterativos de optimización basados en gradientes descendientes, las memorias asociativas operan mediante mecanismos algebraicos que permiten el almacenamiento inmediato y la recuperación eficiente de patrones incluso en condiciones de información incompleta o corrupta.

Conceptualmente, una memoria asociativa puede definirse como un sistema de procesamiento de información que implementa una transformación no lineal $\Phi: \mathbb{R}^n \rightarrow \mathbb{R}^m$, donde $\mathbb{R}^n$ representa el espacio de entrada de dimensiones n y $\mathbb{R}^m$ el espacio de salida de dimensiones m. Este sistema está diseñado para almacenar un conjunto finito de P patrones de entrenamiento, denotados como $\{\mathbf{x}^{(1)}, \mathbf{y}^{(1)}\}$, $\{\mathbf{x}^{(2)}, \mathbf{y}^{(2)}\}$, ..., $\{\mathbf{x}^{(P)}, \mathbf{y}^{(P)}\}$, donde cada par $(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$ representa una asociación específica entre un patrón de entrada y su correspondiente patrón de salida. La operación fundamental de la memoria consiste en recuperar el patrón almacenado $\mathbf{y}^{(k)}$ cuando se presenta una entrada $\mathbf{x}$ que constituye una versión aproximada, parcial o distorsionada del patrón original $\mathbf{x}^{(k)}$.

Las memorias asociativas se clasifican en dos categorías principales según la naturaleza de la transformación que realizan. Las memorias autoasociativas, caracterizadas por la relación $\Phi: \mathbb{R}^n \rightarrow \mathbb{R}^n$, operan dentro del mismo espacio dimensional, es decir, los patrones de entrada y salida pertenecen al mismo dominio espacial. Esta categoría está específicamente diseñada para tareas de recuperación y

completado de patrones, donde el objetivo es reconstruir la versión completa y correcta de un patrón a partir de una representación parcial o corrupta. En contraste, las memorias heteroasociativas, denotadas por Φ: $\mathbb{R}^n \rightarrow \mathbb{R}^m$, establecen correspondencias entre espacios dimensionales diferentes, permitiendo la asociación de patrones de entrada con patrones de salida que pueden exhibir diferentes características y dimensionalidades [2].

El almacenamiento de información en memorias asociativas puede formalizarse mediante dos aproximaciones fundamentales. La primera se sustenta en la regla de Hebb, la cual encuentra su origen en principios neurobiológicos. Donald Hebb postuló que la fuerza de conexión entre dos neuronas aumenta cuando ambas presentan actividad de manera simultánea [3]. Este principio, trasladado al ámbito matemático, se implementa mediante la construcción de una matriz de pesos sinápticos W, definida como:

$$W = \sum_{k=1}^{P} y^{(k)}(x^{(k)})^T \quad (3)$$

En esta formulación, cada par de patrones de entrada $x^{(k)}$ y salida $y^{(k)}$ aporta de manera acumulativa a la construcción de la matriz de pesos W. Es decir, el aprendizaje se realiza de forma incremental: cada nueva asociación (x, y) refuerza las conexiones existentes, dejando su propia huella en la matriz. Este enfoque resulta intuitivo y fácil de implementar, reflejando de manera aproximada el comportamiento de las conexiones sinápticas biológicas. Sin embargo, cuando el número de patrones almacenados es muy grande o los patrones se parecen entre sí, las huellas pueden superponerse y competir por el mismo espacio de representación, generando interferencias que reducen la precisión de la recuperación de los patrones originales.

La segunda aproximación para construir memorias asociativas utiliza la pseudo-inversa de Moore–Penrose, que constituye una generalización de la inversa de una matriz. A diferencia de la inversa convencional, que solo existe para matrices cuadradas y no singulares, la pseudo-inversa se puede calcular para cualquier matriz, ya sea rectangular o singular. Esto es especialmente útil en redes asociativas, donde la matriz de entradas X puede tener más patrones que dimensiones.

En el contexto de memorias asociativas, la pseudo-inversa permite formular la relación entrada–salida como un problema de mínimos cuadrados. Esto significa que, dada una matriz de entradas X (donde cada columna es un patrón $x^{(k)}$ y una matriz de salidas Y (donde cada columna es el patrón correspondiente $y^{(k)}$), se busca una matriz de pesos W que cumpla:

$$WX \approx Y \quad (4)$$

Esta aproximación corresponde a una optimización global. La pseudo-inversa encuentra la matriz W que minimiza la suma de los cuadrados de las diferencias entre las salidas deseadas y las salidas generadas. En otras palabras, se busca que el error global entre Y y WX sea lo más pequeño posible.

Matemáticamente, la solución óptima se obtiene como:

$$W = YX^{+} \quad (5)$$

A diferencia de la regla de Hebb, que acumula asociaciones de manera local y aditiva, la pseudo-inversa calcula los pesos considerando todos los patrones a la vez. Esto permite reducir la interferencia entre patrones, incluso cuando muchos patrones se solapan o la red tiene alta densidad de información.

C. Modelos de Memoria Asociativa

1) Memoria Asociativa Lineal (LAM)

La Memoria Asociativa Lineal (LAM, por sus siglas en inglés) constituye un modelo de memoria heteroasociativa dentro del ámbito de las redes neuronales, cuyo objetivo principal es almacenar y recuperar patrones de manera que, dada una entrada aproximada o parcial, se pueda obtener la salida asociada correcta. La arquitectura de la LAM consiste en dos capas de neuronas: una capa de entrada con N neuronas y una capa de salida con M neuronas. Cada neurona de la capa de salida está conectada a todas las neuronas de la capa de entrada mediante pesos sinápticos, formando una matriz de pesos $W \in \mathbb{R}^{M \times N}$. La LAM no posee capas ocultas ni dinámicas recurrentes. Su estructura es completamente feedforward, lo que permite realizar el mapeo lineal de patrones de entrada a patrones de salida de manera directa. Cada neurona de salida puede además incorporar un sesgo que ajusta su nivel basal de activación, es decir, la tendencia inherente de la neurona a activarse incluso sin recibir entradas [4].

El funcionamiento de la LAM se basa en el principio de la suma ponderada de las contribuciones de cada patrón de entrenamiento, siguiendo la regla de Hebb. Para implementar la LAM, los patrones de entrada y salida se codifican en formato bipolar mediante las transformaciones $x^k = 2a^{(k)} - 1$ para la entrada y para la salida $y^k = 2b^{(k)} - 1$, donde $a^{(k)}$ y $b^{(k)}$ representan los valores binarios originales del patrón k. Esta codificación bipolar permite que los valores de las neuronas tomen valores en $\{-1, +1\}$, facilitando la operación lineal de la memoria y la convergencia de los cálculos de recuperación.

Formalmente, la matriz de pesos W de la LAM se calcula mediante la regla de Hebb de forma vectorizada como:

$$W = \sum_{k=1}^{P} y^{(k)} (x^{(k)})^T \tag{6}$$

donde P es el número total de patrones almacenados. Cada fila de W representa los pesos que conectan todas las neuronas de entrada con una neurona de salida, mientras que cada columna representa las conexiones de una neurona de entrada con todas las neuronas de salida.

Para mejorar la flexibilidad del modelo, se introduce un vector de sesgo $b \in \mathbb{R}^{N\times 1}$, que permite ajustar el nivel basal de activación de cada neurona de salida y compensar desplazamientos en los patrones de entrada. De forma explícita, para la neurona de salida i, el sesgo se calcula como:

$$b_i = -\frac{1}{2}\sum_{j=1}^{N} w_{ij} \tag{7}$$

donde w_{ij} es el peso que conecta la neurona de entrada j con la neurona de salida i, y N es el número de neuronas de entrada. En forma vectorizada, esto puede expresarse como:

$$b = -\frac{1}{2} W 1_N \tag{8}$$

donde $1_N \in \mathbb{R}^{N\times 1}$ es un vector columna de unos.

De esta manera, la recuperación de un patrón de salida $\hat{y}$ dado un patrón de entrada x se realiza mediante la operación lineal:

$$z = Wx + b \tag{9}$$

y posteriormente se aplica la función de activación signo elemento a elemento para obtener valores discretos en $\{-1, +1\}$, restaurando la codificación bipolar original:

$$\hat{y} = \varphi(Wx + b) \tag{10}$$

donde φ representa la función de activación signo, definida como:

$$\varphi(z) = \begin{cases} +1 & si\ z \geq 0 \\ -1 & si\ z < 0 \end{cases} \tag{11}$$

Esta operación garantiza que cada neurona de salida considere simultáneamente la información combinada de todas las neuronas de entrada y que el sesgo ajuste adecuadamente el umbral de activación.

La Memoria Asociativa Lineal se distingue por su capacidad de establecer asociaciones directas y deterministas entre conjuntos de patrones de entrada y salida, operando bajo un esquema de codificación bipolar y aprendizaje hebbiano. Esta característica la convierte en un modelo particularmente adecuado para aplicaciones que requieren recuperación exacta o aproximada de información en condiciones controladas. En términos de aplicaciones, la Memoria Asociativa Lineal (LAM) se ha utilizado principalmente en entornos académicos y experimentales como un modelo de referencia para el estudio de memorias asociativas heteroasociativas. Su formulación matemática basada en operaciones lineales facilita el análisis teórico y la implementación computacional, lo que la hace útil para evaluar la recuperación de patrones bajo condiciones controladas de ruido. En particular, la LAM ha sido aplicada en tareas de asociación de patrones binarios, como el mapeo entre representaciones de caracteres y símbolos discretos, así como en sistemas de comunicación digital para la corrección básica de errores en señales codificadas. Si bien su adopción en aplicaciones prácticas a gran escala es limitada debido a restricciones en capacidad de almacenamiento y sensibilidad a altas tasas de ruido, la LAM mantiene relevancia como herramienta pedagógica y de investigación para explorar principios fundamentales de las memorias asociativas.

Una de las principales bondades de la LAM es su simplicidad estructural y computacional. Al basarse en una suma ponderada de patrones codificados y en la regla de Hebb para el aprendizaje de la matriz de pesos, el modelo evita la necesidad de algoritmos iterativos de retropropagación, lo que reduce significativamente el tiempo de entrenamiento y facilita la implementación en hardware o sistemas embebidos.

2) *Memoria Asociativa Bidireccional (BAM)*

La Memoria Asociativa Bidireccional (BAM, por sus siglas en inglés) se estructura típicamente en dos capas de neuronas correspondientes a dos dominios de patrones distintos, denotados como X y Y. Cada capa contiene un conjunto de neuronas, donde cada neurona de la capa X está conectada a todas las neuronas de la capa Y mediante pesos sinápticos que representan las asociaciones almacenadas. Estas conexiones son bidireccionales, de manera que los patrones pueden propagarse de X a Y y viceversa. La arquitectura carece de conexiones internas dentro de cada capa, lo que simplifica la dinámica y garantiza que la recuperación de patrones dependa únicamente de las interacciones entre capas. Cada neurona opera con una función de activación signo, típica de redes de codificación bipolar, que permite mantener las señales

en valores $\{-1, +1\}$ y facilita la convergencia iterativa del sistema [5].

El aprendizaje en la BAM se basa en la regla de Hebb, la cual establece que la fuerza de conexión entre dos neuronas aumenta proporcionalmente a la coincidencia de sus activaciones. Sea un conjunto de P patrones de entrenamiento $(x^{(k)}, y^{(k)})$, donde $x^{(k)} \in \{-1, +1\}^n$ y $y^{(k)} \in \{-1, +1\}^m$ son vectores codificados en formato bipolar, la matriz de pesos W que define la asociación entre los conjuntos X y Y se calcula como:

$$W = \sum_{k=1}^{P} y^{(k)} (x^{(k)})^T \tag{6}$$

La dimensión de la matriz de pesos W es $m \times n$, donde n es el número de neuronas en la capa X y m el número de neuronas en la capa Y. Esta estructura garantiza que cada neurona de X tenga conexión con cada neurona de Y y viceversa, permitiendo la recuperación bidireccional de los patrones almacenados.

La recuperación de patrones en la BAM se fundamenta en la propagación bidireccional de las señales a través de la matriz de pesos. Formalmente, a partir de un patrón inicial de entrada x, el sistema genera la salida asociada aplicando la transformación lineal seguida de la función de activación signo:

$$\hat{y} = \varphi(Wx) \tag{12}$$

Del mismo modo, si se parte de un patrón de salida y, es posible recuperar la entrada asociada mediante la operación inversa:

$$\hat{x} = \varphi(W^T y) \tag{13}$$

donde φ representa la función de activación signo.

En la formulación original de Bart Kosko, la Memoria Asociativa Bidireccional fue concebida como un sistema dinámico recurrente. El mecanismo de recuperación se implementa a través de un proceso iterativo en el cual los vectores de entrada y salida se actualizan de manera alternada hasta alcanzar un estado estable. Formalmente, dado un par inicial $(x^{(0)}, y^{(0)})$, las actualizaciones se definen como:

$$\hat{y}^{(t+1)} = \varphi\left(W^T x^{(t)}\right) \qquad \hat{x}^{(t+1)} = \varphi\left(W^T y^{(t)}\right) \tag{14}$$

Donde $x^{(t)}$ y $y^{(t)}$ representan los estados de los vectores en la iteración t. El proceso continúa hasta que se alcanza un punto fijo, es decir, un par (x^*, y^*) que cumple la condición de invarianza:

$$\hat{x}^* = \varphi(W^T y^*) \qquad \hat{y}^* = \varphi(W^T x^*) \tag{15}$$

En este estado, las actualizaciones sucesivas no modifican los vectores y el sistema ha convergido a una asociación estable almacenada en la memoria. Dichos estados fijos constituyen los patrones de equilibrio de la BAM, y pueden interpretarse como los pares de patrones más consistentes con la estructura de la matriz de pesos.

No obstante, en implementaciones prácticas orientadas al reconocimiento de patrones discretos, donde la interferencia entre asociaciones es reducida, no resulta necesario mantener la dinámica iterativa. En tales casos, una única aplicación de las transformaciones es suficiente para recuperar la asociación correspondiente. Bajo este esquema, el modelo opera en modo directo, lo que elimina la necesidad de convergencia sucesiva y disminuye el costo computacional del proceso de recuperación. Para los propósitos del presente estudio, se adopta precisamente esta estrategia, es decir, cada entrada se proyecta una sola vez a través de la matriz de pesos, obteniéndose el patrón asociado de salida sin recurrir a actualizaciones alternadas.

Una de las particularidades más relevantes de la Memoria Asociativa Bidireccional es su capacidad de establecer correspondencias recíprocas entre dos dominios de patrones, lo que le permite recuperar un vector de salida a partir de un vector de entrada y, de manera simétrica, reconstruir la entrada correspondiente a partir de la salida. Esta bidireccionalidad le otorga una ventaja respecto a otros modelos como la LAM, que opera únicamente de forma unidireccional, o la red de Hopfield, cuya naturaleza es autoasociativa. Gracias a esta propiedad, la BAM se adapta de manera eficiente a escenarios donde la información puede recibirse o consultarse en cualquiera de los dos dominios asociados.

El funcionamiento de la BAM puede realizarse mediante actualizaciones sucesivas, conocidas como dinámica iterativa, lo que permite que el sistema evolucione hacia un estado de equilibrio estable que corresponde al par de patrones más cercano almacenado en la memoria. Este proceso confiere al modelo cierta tolerancia al ruido y a la distorsión, ya que entradas incompletas o alteradas pueden refinarse progresivamente hasta coincidir con una asociación válida.

En términos de aplicaciones, la BAM ha sido explorada principalmente en escenarios de investigación relacionados con el reconocimiento de patrones y la recuperación de información. Gracias a su capacidad bidireccional, se considera un modelo adecuado para tareas como la correspondencia entre caracteres manuscritos y sus representaciones digitales, la asociación entre símbolos pertenecientes a diferentes alfabetos o sistemas de codificación, y la recuperación de secuencias en entornos de comunicación donde los datos pueden llegar con ruido o distorsión. Asimismo, tiene potencial en dominios como la verificación de identidad, donde diferentes modalidades de representación (por ejemplo, imágenes) requieren ser

asociadas de manera consistente. Aunque en muchos de estos campos su uso ha permanecido en un nivel experimental y académico, la estructura de la BAM la posiciona como una herramienta conceptual valiosa para problemas donde la relación entre dos conjuntos de datos debe preservarse de manera recíproca y tolerante a perturbaciones.

3) Red de Hopfield

La red de Hopfield es un modelo clásico de memoria autoasociativa dentro de las redes neuronales artificiales, introducido por John Hopfield en 1982. Su objetivo principal es almacenar un conjunto de patrones binarios y recuperarlos a partir de versiones incompletas o ruidosas de los mismos. La arquitectura de la red consiste en n neuronas binarias (−1 o +1), totalmente conectadas entre sí, de manera que cada neurona se comunica directamente con todas las demás, pero no consigo misma. Esta conectividad se denomina recurrente completa [6].

Una característica fundamental de la red es la simetría de los pesos sinápticos, es decir, el peso de conexión entre la neurona i y la neurona j es igual al de la conexión inversa $(w_{ij} = w_{ji})$, y $w_{ii} = 0$. Esta simetría garantiza que el sistema puede definirse mediante una función de energía global, la cual disminuye o permanece constante con cada actualización de la red, asegurando la convergencia hacia estados estables denominados atractores.

El aprendizaje en una red de Hopfield se realiza mediante la regla de Hebb, que fortalece las conexiones entre neuronas que se activan simultáneamente en los patrones a almacenar. Sea un conjunto de p patrones binarios $x^{\mu} \in \{-1, +1\}^n$, $\mu = 1, \dots, p$ la matriz de pesos se calcula de la siguiente forma vectorizada:

$$W = \sum_{\mu=1}^{P} x^{\mu}(x^{\mu})^T \tag{16}$$

De manera explícita, elemento por elemento:

$$w_{ij} = \sum_{\mu=1}^{P} x_i{}^{\mu} x_j{}^{\mu} \quad con\ w_{ii} = 0 \tag{17}$$

Por otro lado, la recuperación de un patrón se realiza mediante la evolución del estado de las neuronas a partir de un vector de entrada inicial. Cada neurona actualiza su estado según la suma ponderada de las entradas recibidas:

$$x_i{}^{(t+1)} = \varphi\left(\sum_{j=1}^{n} w_{ij} x_j{}^{(t)}\right) \tag{18}$$

donde la función φ devuelve +1 si el argumento es positivo y −1 si es negativo.

La dinámica de la red de Hopfield se basa en la actualización de los estados de las neuronas, un proceso que define cómo la red evoluciona desde un vector de entrada inicial hacia un patrón almacenado o un mínimo de energía. Este proceso puede implementarse de dos formas principales: síncrona y asíncrona.

En la actualización síncrona, todas las neuronas de la red se actualizan simultáneamente en cada paso de tiempo discreto. Para ilustrar este proceso, consideremos un ejemplo con una red de tres neuronas $(n = 3)$ y un vector de estado inicial:

$$x(t) = \begin{bmatrix} +1 \\ -1 \\ +1 \end{bmatrix} \qquad w = \begin{bmatrix} 0 & 1 & -1 \\ 1 & 0 & 1 \\ -1 & 1 & 0 \end{bmatrix}$$

La actualización síncrona se realiza multiplicando el vector de estado por la matriz de pesos y aplicando la función signo:

$$Wx(t) = \begin{bmatrix} 0 & 1 & -1 \\ 1 & 0 & 1 \\ -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} +1 \\ -1 \\ +1 \end{bmatrix} = \begin{bmatrix} (0)(+1) + (1)(-1) + (-1)(+1) \\ (1)(+1) + (0)(-1) + (1)(+1) \\ (-1)(+1) + (1)(-1) + (0)(+1) \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \\ -2 \end{bmatrix}$$

Aplicando la función signo:

$$x^{(t+1)} = \varphi\left(\begin{bmatrix} -2 \\ 2 \\ -2 \end{bmatrix}\right) = \begin{bmatrix} -1 \\ +1 \\ -1 \end{bmatrix}$$

Como se observa, todas las neuronas cambian de estado simultáneamente. Este enfoque permite evaluar el efecto global de todas las conexiones en un solo paso, pero en redes más grandes puede generar conflictos temporales o ciclos si varias neuronas cambian al mismo tiempo.

Por otro lado, en la actualización asíncrona, cada neurona se actualiza de manera individual y secuencial. La selección de la neurona puede realizarse aleatoriamente o siguiendo un orden predefinido. La actualización asíncrona garantiza que cada cambio de neurona disminuya estrictamente la función de energía o la mantenga constante, eliminando la posibilidad de oscilaciones debido a conflictos entre neuronas.

Este método es particularmente útil en redes grandes, donde la convergencia síncrona podría tardar más o generar patrones inestables, pero requiere más pasos de actualización y un control cuidadoso del orden de selección de las neuronas. Para ilustrar, considerando el mismo vector de estado y matriz de pesos del ejemplo anterior:

Si se actualiza primero la neurona 1:

$$x_1{}^{(t+1)} = \varphi(w_{12}x_2 + w_{13}x_3) = \varphi\big((1)(-1) + (-1)(+1)\big) = \varphi(-2) = -1$$

Posteriormente, la neurona 2:

$$x_2{}^{(t+1)} = \varphi\big(w_{21}x_1{}^{(t+1)} + w_{23}x_3\big) = \varphi\big((1)(-1) + (1)(+1)\big) = \varphi(0) = +1$$

Y finalmente la neurona 3:

$$x_3^{(t+1)} = \varphi\left(w_{31}x_1^{(t+1)} + w_{32}x_2^{(t+1)}\right) = \varphi\big((-1)(-1) + (1)(+1)\big) = \varphi(2) = +1$$

Como se observa, cada neurona se actualiza individualmente, reduciendo la posibilidad de conflictos que podrían surgir si todas cambian simultáneamente. Cada actualización asegura que la función de energía disminuya o se mantenga constante, favoreciendo la convergencia estable de la red. Sin embargo, en el presente estudio, se opta por la actualización síncrona, ya que permite una implementación más sencilla y facilita la evaluación del efecto de diferentes niveles de ruido y funciones de activación sobre la recuperación de patrones, sin necesidad de iterar de forma individual por neurona.

La convergencia en la red de Hopfield se produce cuando el vector de estados $x(t)$ deja de cambiar tras sucesivas actualizaciones, es decir, cuando cada neurona alcanza un estado estable que no varía en pasos posteriores. Matemáticamente, para cada neurona i, el estado de equilibrio se define x_i^* como:

$$x_i^* = \varphi\left(\sum_{j=1}^{n} w_{ij} x_j^*\right) \qquad (19)$$

En esta expresión, x_i^* representa el estado final de la neurona i, mientras que x_j^* son los estados de todas las neuronas que ejercen influencia sobre i a través de los pesos sinápticos w_{ij}. La doble notación i y j indica que se evalúa la contribución de todas las neuronas $j = 1,2,\dots,n$ sobre la neurona i para determinar su estado final. Si la suma ponderada de las influencias es positiva, la neurona se activa, si es negativa, se desactiva. La diagonal de la matriz de pesos se establece en cero $(w_{ii} = 0)$ para evitar que una neurona se influya a sí misma.

Los estados estables que cumple esta condición se clasifican en dos tipos: los atractores de memoria, que corresponden a los patrones originalmente almacenados y permiten la recuperación correcta de información, y los mínimos espurios, que son configuraciones estables no deseadas originadas por la superposición de memorias y que pueden capturar estados iniciales parcialmente ruidosos. La convergencia garantiza que, incluso ante entradas incompletas o parcialmente perturbadas, la red tenderá a estabilizarse en un estado cercano al patrón original más coherente, funcionando como un corrector de errores natural.

El comportamiento dinámico de la red puede analizarse mediante la función de energía de Hopfield, que asigna un valor escalar a cada configuración de la red:

$$E(x) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} x_i x_j \qquad (20)$$

La función de energía en la red de Hopfield asigna un valor escalar a cada posible configuración de estados de la red, permitiendo cuantificar la estabilidad de dichas configuraciones. Conceptualmente, se puede imaginar cada estado de la red como un punto en un paisaje de energía, donde la altura del punto corresponde al valor de $E(x)$. Los patrones almacenados se representan como pozos profundos en este paisaje, es decir, son configuraciones altamente estables a las que la dinámica de la red tiende naturalmente. Por su parte, los mínimos espurios son configuraciones también estables, pero menos profundas, lo que significa que pueden atraer algunos estados iniciales, pero representan soluciones no deseadas o incorrectas.

Cada actualización de una neurona, ya sea síncrona o asíncrona, tiene el efecto de disminuir o mantener constante la energía total de la red. Este principio garantiza que la red evolucione de manera monótona hacia un mínimo local de energía y que no entre en ciclos infinitos de actualización, evitando oscilaciones continuas entre estados. En otras palabras, la función de energía actúa como un criterio de estabilidad, donde los estados con menor energía son más resistentes a cambios y representan los atractores de la red. Así, la red realiza un descenso automático en este paisaje energético hasta que alcanza un estado estable, sea un patrón deseado o un mínimo espurio.

Cabe destacar que la red presenta limitaciones inherentes en cuanto a la capacidad de almacenamiento. Para una red con n neuronas, se estima que la recuperación confiable se mantiene hasta aproximadamente $0.13n$ patrones. Cuando se supera este umbral, la superposición de memorias incrementa la probabilidad de que la red converja a mínimos espurios, produciendo recuperaciones incorrectas. Estos mínimos representan configuraciones estables, pero no deseadas, y constituyen uno de los principales factores que limitan la eficiencia de la red en escenarios con alta densidad de patrones.

A pesar de estas restricciones, la red de Hopfield ofrece ventajas significativas. Entre ellas se encuentra su capacidad para corregir errores de manera natural, funcionando como un sistema de recuperación de información frente a entradas ruidosas. La red también proporciona un marco teórico sólido para analizar la dinámica de sistemas recurrentes y la estabilidad de estados en modelos de memoria distribuidos. Su función de energía permite visualizar el comportamiento de la red, facilitando la comprensión de la convergencia y de la relación entre estados estables y patrones almacenados.

En términos de aplicaciones, la red de Hopfield ha sido utilizada principalmente en tareas de reconocimiento y recuperación de patrones binarios, donde permite identificar configuraciones completas a partir de entradas parciales o

parcialmente ruidosas. Otras posibles aplicaciones, como la recuperación de datos corruptos en secuencias binarias o la resolución de problemas de optimización combinatoria, incluyendo el problema del viajante (TSP), la asignación de tareas o la planificación de rutas, se han explorado principalmente en contextos académicos y experimentales, pero no constituyen implementaciones prácticas estándar debido a limitaciones en la capacidad de almacenamiento y a la aparición de mínimos espurios. Más allá de estas aplicaciones, la red de Hopfield sigue siendo ampliamente empleada como modelo conceptual en estudios de neurociencia computacional y teoría de redes recurrentes, sirviendo como referencia para el desarrollo de arquitecturas más complejas de memoria asociativa y sistemas de recuperación de información distribuidos.

D. Funciones de Activación en Memorias Asociativas

En memorias asociativas, la función de activación determina la salida de cada neurona a partir de la suma ponderada de sus entradas, condicionando directamente la dinámica de actualización, la estabilidad de los estados y la capacidad de recuperación de patrones. La selección de la función de activación tiene implicaciones importantes sobre la convergencia de la red y su tolerancia frente a perturbaciones o entradas parcialmente alteradas. En este estudio se consideran dos funciones escalón: una simétrica y una asimétrica, elegidas por su relevancia práctica y compatibilidad con distintas codificaciones de patrones.

La función escalón simétrica, también conocida como función signo, asigna a cada neurona un valor de salida −1 o +1 dependiendo del signo de su entrada ponderada.

$$y_i = \begin{cases} 1 & si \sum_{j=1}^{n} w_{ij}x_j \geq 0 \\ -1 & si \sum_{j=1}^{n} w_{ij}x_j < 0 \end{cases} \qquad (21)$$

Esta función es la más empleada en memorias asociativas binarias, como LAM, BAM y Hopfield, ya que mantiene coherencia con la codificación bipolar de los patrones. Su actualización determinista asegura que la red evolucione hacia estados estables que corresponden a los patrones almacenados. La simplicidad computacional de esta función permite realizar actualizaciones rápidas y consistentes. No obstante, al ser estrictamente discreta, no permite gradaciones intermedias en la activación, lo que puede limitar la tolerancia de la red frente a entradas muy ruidosas o altamente distorsionadas.

Por otra parte, la función escalón asimétrica mapea las entradas a un rango $[0,1]$, manteniendo la estructura discreta pero adaptada a codificaciones binarias estándar no bipolares:

$$y_i = \begin{cases} 1 & si \sum_{j=1}^{n} w_{ij}x_j \geq 0 \\ 0 & si \sum_{j=1}^{n} w_{ij}x_j < 0 \end{cases} \qquad (22)$$

Esta función permite representar activaciones binarias en entornos donde los patrones se codifican como 0 y 1, y ofrece una transición determinista similar a la función signo. Su principal diferencia radica en el rango de salida, lo que puede ser ventajoso en implementaciones que requieren compatibilidad con sistemas discretos estándar o hardware digital basado en lógica binaria convencional. Al igual que la función signo, la función escalón asimétrica no admite activaciones intermedias, y su tolerancia frente a perturbaciones depende de la arquitectura de la red y de la separación entre patrones almacenados.

La elección entre la función escalón simétrica y la escalón asimétrica tiene implicaciones directas sobre la codificación de los patrones y la interpretación de la salida de la red. Mientras que la función simétrica es adecuada para codificaciones bipolares y garantiza convergencia rápida hacia atractores de memoria, la función asimétrica es más apropiada para representaciones binarias estándar y puede facilitar la integración con sistemas digitales. En este estudio, la comparación de ambas funciones permitirá evaluar su efecto sobre la capacidad de recuperación, la robustez frente a ruido y la estabilidad de los estados almacenados en redes LAM, BAM y Hopfield.

E. Ruido y Robustez en Redes Neuronales

En memorias asociativas, la presencia de ruido en los patrones de entrada representa un factor crítico que afecta la capacidad de la red para recuperar correctamente la información almacenada. Se entiende por ruido cualquier alteración que modifique los valores originales de un patrón, comprometiendo la integridad de los datos y la convergencia de la red hacia los atractores de memoria deseados. En aplicaciones de reconocimiento de patrones visuales, como letras del alfabeto latino codificadas en forma binaria o bipolar, el ruido puede originarse por errores de adquisición, transmisión defectuosa, degradación de la imagen o representaciones parciales de los caracteres.

El ruido puede clasificarse según su naturaleza. El ruido aleatorio (salt-and-pepper) altera individualmente ciertos elementos del patrón, cambiando valores de 0 a 1 o de −1 a +1, simulando errores discretos de captura o transmisión. El ruido gaussiano añade perturbaciones continuas con distribución normal, modelando variaciones graduales; en patrones binarios, esto se traduce en una probabilidad de que cada bit se invierta. Por último, el ruido estructural o correlacionado modifica bloques completos del patrón, como trazos incompletos o segmentos borrados de caracteres, y es especialmente relevante en reconocimiento de escritura manual o impresiones deterioradas, donde los errores no son aleatorios sino concentrados en regiones específicas del patrón.

La introducción del ruido en los patrones se realiza mediante un parámetro de probabilidad p que determina la proporción de neuronas afectadas en cada patrón. Para un patrón binario $x \in \{0,1\}^n$, cada elemento x_i tiene una probabilidad p de invertirse, es decir, de cambiar de 0 a 1 o de 1 a 0. En codificación bipolar, el procedimiento es equivalente: cada neurona $x \in \{-1,+1\}^n$ tiene una probabilidad p de cambiar de signo, pasando de −1 a +1 o de +1 a −1. Este método permite controlar de manera precisa la intensidad del ruido aplicado, desde perturbaciones leves que afectan solo unas pocas neuronas hasta alteraciones más extensas que comprometen gran parte del patrón.

A partir de esta definición de ruido, surge de manera natural el concepto de robustez, que en el presente estudio se referirá a la capacidad de un modelo de memoria asociativa para mantener su desempeño y recuperar correctamente los patrones almacenados a pesar de la presencia de perturbaciones en la entrada. En otras palabras, la robustez mide cuán tolerante es la red frente a datos incompletos, distorsionados o parcialmente corruptos. Esta propiedad depende de factores como la estructura de la matriz de pesos, la codificación de los patrones, la densidad de almacenamiento y la función de activación utilizada, ya que todos influyen en cómo los estados iniciales ruidosos evolucionan hacia los atractores de memoria.

La robustez se evalúa mediante métricas objetivas, siendo la más común el porcentaje de recuperación correcta, definido como la proporción de neuronas cuyo estado final coincide con el patrón almacenado después de la actualización de la red:

$$Recuperación = \frac{1}{n}\sum_{i=1}^{n} \delta(x_i{}^{orig}, x_i{}^{rec}) \times 100\% \qquad (23)$$

donde $\delta(a,b)$ es la función indicadora que vale 1 $a = b$ y 0 en caso contrario, y n es el número de neuronas. De manera complementaria, se puede analizar la tasa de convergencia, es decir, el número de pasos requeridos para que un patrón ruidoso alcance un estado estable, ya sea un patrón almacenado o un mínimo espurio. Estas medidas permiten comparar objetivamente la robustez de diferentes modelos como LAM, BAM y Hopfield frente a distintos niveles y tipos de ruido.

F. Estudios Comparativos Previos

El análisis comparativo de memorias asociativas constituye un área de investigación extensa, donde diversos autores han evaluado el rendimiento, la robustez frente a ruido y las limitaciones teóricas de modelos como la Linear Associative Memory (LAM), la Bidirectional Associative Memory (BAM) y la red de Hopfield. Estas comparaciones permiten identificar no solo las fortalezas particulares de cada arquitectura, sino también los escenarios donde su desempeño es más confiable.

En el caso de la BAM, Wang y Jiang realizaron un estudio exhaustivo en el que compararon su desempeño con el de la red de Hopfield discreta y la red de propagación contraria (Counterpropagation Network, CPN) en el procesamiento de datos ruidosos. Los autores observaron que tanto BAM como Hopfield presentaban resultados similares cuando el nivel de ruido en los patrones de entrada era bajo, logrando recuperar correctamente las asociaciones almacenadas. Sin embargo, bajo condiciones de ruido más intenso, la BAM mostró mayor capacidad de recuperación, debido a que su mecanismo bidireccional permitía estabilizarse en patrones coherentes incluso cuando la entrada se encontraba significativamente perturbada. Los experimentos de Wang y Jiang también destacaron que la implementación de la BAM mediante descenso de gradiente mejora su eficacia, ampliando el rango de perturbaciones tolerables sin perder precisión en la recuperación. Además, estudios posteriores sobre la sensibilidad al ruido en BAM han demostrado que su inmunidad depende de dos factores clave: el valor absoluto mínimo de las entradas netas (Minimum Absolute Value, MAV) y la varianza de los pesos sinápticos. Esto implica que redes con pesos mejor distribuidos y valores mínimos suficientemente altos presentan mayor robustez frente a la corrupción de datos, lo que convierte a la BAM en un modelo flexible y ajustable a escenarios prácticos donde el ruido es inevitable [7].

En cuanto a la red de Hopfield, múltiples investigaciones han explorado tanto sus limitaciones teóricas como sus extensiones modernas. Es bien conocido que la capacidad máxima de almacenamiento de este modelo se encuentra en el orden de $0.13n$ patrones para una red de n neuronas, lo que restringe su uso en tareas con grandes volúmenes de datos. Asimismo, se ha observado que la presencia de ruido sináptico o de sobrecarga en el almacenamiento produce la aparición de mínimos espurios, es decir, configuraciones estables que no corresponden a patrones entrenados y que pueden atraer estados iniciales ruidosos hacia recuperaciones incorrectas. Sin embargo, Hu y colaboradores propusieron un enfoque basado en redes de Hopfield dispersas (sparse Hopfield networks), en las que la conectividad entre neuronas no es completa, sino limitada a un subconjunto reducido. Este diseño permitió alcanzar memorias robustas incluso con un número relativamente pequeño de patrones de entrenamiento, reduciendo la interferencia entre asociaciones y aumentando la tolerancia a entradas corruptas [8]. Más recientemente, el desarrollo de Dense Associative Memories (DAM) ha supuesto un avance significativo sobre el modelo original de Hopfield. Estas memorias introducen no linealidades más pronunciadas tanto en la función de energía como en las funciones de activación, lo que permite superar las limitaciones clásicas de capacidad y alcanzar un almacenamiento superlineal o incluso exponencial. Los resultados experimentales sugieren que este enfoque no solo mejora la cantidad de patrones que pueden recuperarse, sino que también incrementa la robustez frente al ruido, al suavizar

el paisaje de energía y reducir la probabilidad de caer en mínimos espurios [9].

En relación con la LAM, la investigación también ha aportado resultados relevantes que permiten entender su papel como modelo autoasociativo lineal. Cherkassky y colaboradores analizaron variantes de LAM bajo la presencia de ruido en los patrones de entrenamiento, concluyendo que el modelo Correlation Matrix Memory (CMM) es más robusto que los métodos basados en pseudo-inversa, ya que aprovecha mejor la correlación entre patrones y reduce la amplificación del ruido durante la recuperación [10]. Por otra parte, Dayan y Willshaw exploraron el uso de la regla de covarianza como método de aprendizaje en LAM, mostrando que, bajo condiciones de codificación dispersa, esta regla mejora la relación señal/ruido y, por tanto, la calidad de recuperación [11].

En conjunto, los resultados de estos estudios evidencian que cada una de estas arquitecturas (LAM, BAM y Hopfield) ofrece ventajas y limitaciones específicas. LAM destaca por su sencillez y aplicaciones en restauración de señales, BAM sobresale en escenarios donde se requiere flexibilidad bidireccional y mayor tolerancia frente a ruido, y Hopfield sigue siendo un referente teórico y práctico para el análisis de memorias autoasociativas y la optimización combinatoria. Estos antecedentes proporcionan una base sólida para comparar sus desempeños en este estudio, particularmente en el contexto del reconocimiento de letras del alfabeto latino bajo condiciones de perturbación y con diferentes funciones de activación.

III. IMPLEMENTACIÓN

La implementación de los modelos de memoria asociativa se realizó utilizando Python 3.10, apoyándose en la biblioteca NumPy para el manejo de matrices y operaciones vectorizadas. Cada modelo se desarrolló siguiendo las reglas de aprendizaje de Hebb, adaptadas según la naturaleza específica de la red. A continuación, se describe detalladamente la construcción de cada arquitectura, la formulación matemática y el pseudocódigo correspondiente.

A. Implementación de la Red de Hopfield

La red de Hopfield se implementó mediante una matriz de pesos W de dimensión $n \times n$ donde n representa el número de neuronas, equivalente a la longitud de los vectores de entrada. La matriz de pesos se calculó utilizando la regla de Hebb evitando auto conexiones, de forma que:

$$W_{ij} = \begin{cases} 0, & i = j \\ \frac{i}{n} \sum_{m=1}^{M} x_i^{(m)} x_j^{(m)}, & i \neq j \end{cases} \quad (24)$$

Donde es el número de patrones de entrenamiento, y el factor normaliza la magnitud de los pesos para evitar saturación en redes grandes. Esta formulación es estándar en implementaciones discretas de Hopfield y asegura una dinámica de convergencia estable.

1) Pseudocódigo de Entrenamiento

- Entrada: matriz de pesos $X(M \times n)$
- Salida: matriz de pesos $W\ (n \times n)$

```
1. Inicializar W como matriz de ceros de tamaño n x n
   # Esto asegura que inicialmente no haya influencia entre neuronas.

2. Para cada neurona i = 1 hasta n:
       Para cada neurona j = 1 hasta n:
           Si i ≠ j:
               Calcular W[i,j] = (1/n) * sumatoria de X[m][i] * X[m][j] para
m = 1 a M
               # Aplicar la regla de Hebb, acumulando la correlación entre
las neuronas i y j.
           Sino:
               W[i,j] = 0
               # No se permiten autoconexiones.

3. Retornar W
   # Devuelve la matriz de pesos entrenada.
```

2) Pseudocódigo de Recuperación

- Entrada: matriz de pesos W, vector de entrada x_0 , función de activación f_{act}, máximo de iteraciones $\max\_iter$.
- Salida: vector final x.

```
1. Inicializar x = x0
   # Tomar el patrón de entrada inicial, que puede estar ruidoso o
incompleto.

2. Inicializar iter = 0
   # Contador de iteraciones para evitar bucles infinitos.

3. Mientras iter < max_iter:
       Calcular x_new = f_act(W * x)
       # Aplicar la función de activación al producto de la matriz de pesos
y el vector actual.


       Si x_new == x:
           Romper
           # Si no hay cambios, se alcanzó un estado estable.


       Actualizar x = x_new
       Incrementar iter

4. Retornar x
   # Devolver el patrón final convergente.
```

B. Implementación de la Memoria Asociativa Lineal (LAM)

La LAM se implementó mediante la construcción de una matriz de pesos y un vector de sesgos, siguiendo codificación bipolar para transformar $0 \rightarrow -1$y $1 \rightarrow +1$:

$$W_{ij} = \sum_{k=1}^{M} (2x_i^{(k)} - 1)(2y_j^{(k)} - 1) \quad (25)$$

$$b_j = -\frac{1}{2}\sum_{i=1}^{n} W_{ij} \quad (7)$$

1) Pseudocódigo de Entrenamiento

- Entrada: matrices $X(M \times n)$, $Y(M \times m)$.
- Salida: matriz de pesos $W(n \times m)$, vector de sesgos $\theta(m)$.

```
1. Inicializar W como matriz de ceros de tamaño n × m
   # Esto asegura que inicialmente no haya influencia entre neuronas.

2. Para i = 1 hasta n:                # Iterar sobre neuronas de entrada
       Para j = 1 hasta m:            # Iterar sobre neuronas de salida
           W[i,j] = 0                 # Inicializar acumulador
           Para k = 1 hasta M:        # Iterar sobre todos los patrones de
entrenamiento
               W[i,j] += (2*X[k][i]-1) * (2*Y[k][j]-1)
           # Acumula la correlación entre entrada y salida para todos los
patrones

3. Para j = 1 hasta m:                # Calcular sesgos
       theta[j] = -(1/2) * Σ_{i=1}^{n} W[i,j]
       # Ajusta el nivel basal de activación de cada neurona de salida

4. Retornar W, theta
```

2) Pseudocódigo de Recuperación

- Entrada: vector de entrada x, matriz de pesos W, vector de sesgos θ, función de activación f_{act}
- Salida: vector de salida y.

```
1. y_intermedio = W^T * x + theta      # Multiplicar la
transpuesta de W por x y sumar el sesgo
2. y = f_act(y_intermedio)             # Aplicar función de
activación para obtener salida discreta
3. Retornar y
```

C. Implementación de la Memoria Asociativa Bidireccional (BAM)

La BAM se implementó con una matriz de pesos $W(n \times m)$, calculada como la suma de productos externos de cada par de patrones de entrada y salida:

$$W = \sum_{k=1}^{M} y^{(k)}(x^{(k)})^T \quad (6)$$

No se utilizó sesgo y la recuperación se realizó mediante una sola iteración directa para simplificar la comparación entre modelos.

1) Pseudocódigo de Entrenamiento

- Entrada: matrices $X(M \times n)$, $Y(M \times m)$.
- Salida: matriz de pesos $W(n \times m)$.

```
1.  Inicializar W como matriz de ceros de tamaño n × m
# Esto asegura que al inicio no haya influencia entre neuronas.

Para i = 1 hasta n:                   # Iterar sobre neuronas de entrada
    Para j = 1 hasta m:               # Iterar sobre neuronas de salida
        Inicializar acum = 0          # Acumulador para sumar contribuciones
de todos los patrones
        Para k = 1 hasta M:           # Iterar sobre todos los patrones de
entrenamiento
            acum += X[k][i] * Y[k][j]# Sumar la contribución de este patrón
        W[i, j] = acum                # Asignar el valor acumulado a la
matriz de pesos

Retornar W
```

2) Pseudocódigo de Recuperación

- Entrada: vector de entrada x, matriz de pesos W, función de activación f_{act}
- Salida: vector de salida y.

```
1. y_intermedio = W^T * x              # Multiplicar la
transpuesta de W por el vector de entrada
2. y = f_act(y_intermedio)             # Aplicar función de
activación para obtener salida discreta
3. Retornar y
```

IV. Experimentación y Resultados

Con el propósito de evaluar y comparar el desempeño de las tres arquitecturas de memoria asociativa consideradas en este estudio, se utilizó un conjunto de datos de imágenes manuscritas correspondientes a las letras del alfabeto latino (a–z). Cada imagen posee una resolución de 105 × 105 píxeles y está representada en formato RGB. Este conjunto de datos fue obtenido de la plataforma Kaggle, disponible en el repositorio de Wate Soyan [12].

Previo a las pruebas se realizó un preprocesamiento de las imágenes para darles un formato óptimo y lograr su interpretación por parte de los 3 algoritmos de red. Se utilizó la librería de Python openCV para realizar la lectura de las imágenes y extraer su interpretación matricial de 105x105x3, por lo que en principio se hizo la conversión de codificación RGB a escala de grises, reduciendo la dimensión de las imágenes a 105 × 105 × 1, posteriormente se realizó un escalado de la imagen al 0.7% de su proporción original convirtiendo las imágenes a una dimensión de 74x74x1. Como paso siguiente se cambió el formato de las matrices (interpretación de las imágenes) por una traducción de sus valores según la función de activación deseada, donde en caso de elegir la función escalón simétrica se leen los pixeles de la imagen y en donde se presente el valor en escala de grises 255 se intercambiará por –1 y para valores de 0 se colocará un 1.

Lo anterior simplemente se decidió de manera arbitraría ya que se están utilizando valores binarios donde únicamente existen dos posibilidades, ya sea 0 a 1 para escalón asimétrica o –1 a 1 para escalón simétrica. Una vez teniendo las matrices traducidas al formato correcto según el caso, se procede a aplicar un aplanado de la matriz para obtener un único vector por cada imagen, en este caso se formaron vectores de tamaño 74x74= 5476.

Una vez concluyendo el procesamiento de los datos se generaron los conjuntos de entrenamiento y de prueba conteniendo los vectores de las 26 muestras del alfabeto latín con tamaño 5476 para cada vector según el preprocesamiento anteriormente descrito. A continuación, se muestra un ejemplo de muestra de la categoría "k" contenido en el conjunto de entrenamiento principal:

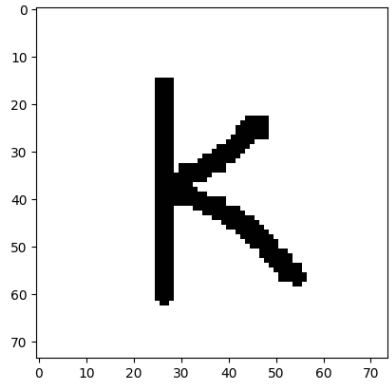


Figura 1. Muestra clasificada en la categoría" k" del conjunto de entrenamiento principal.

El conjunto de prueba está dividido a su vez en 3 subconjuntos. El primer subconjunto contiene muestras de prueba que presentan ruido en la forma/diseño de cada categoría manuscrita, tal como se muestra en la **Figura 2**:

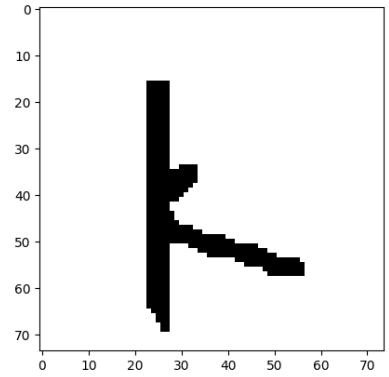


Figura 2. Muestra clasificada en la categoría" k" del conjunto de pruebas con ruido de forma.

El segundo subconjunto de prueba contiene muestras de prueba que presentan un tipo de ruido inducido con un factor aleatorio para la activación y desactivación de pixeles a lo largo del espacio vectorial de la imagen. Este segundo subconjunto presenta ruido inducido con un factor de 0.2, tal y como se visualiza en la **Figura 3**:

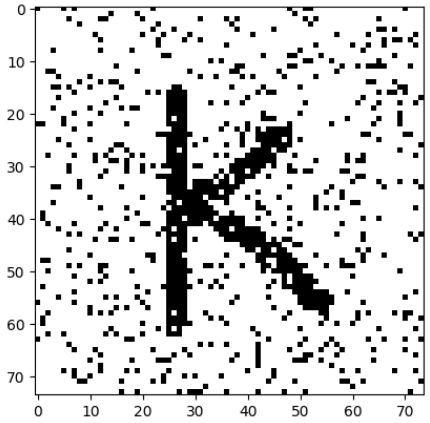


Figura 3. Muestra clasificada en la categoría" k" del conjunto de pruebas con ruido inducido con un factor de 0.2.

Finalmente, el tercer subconjunto de prueba presenta un ruido inducido con un factor de 0.75, lo anterior se puede apreciar en la siguiente muestra:

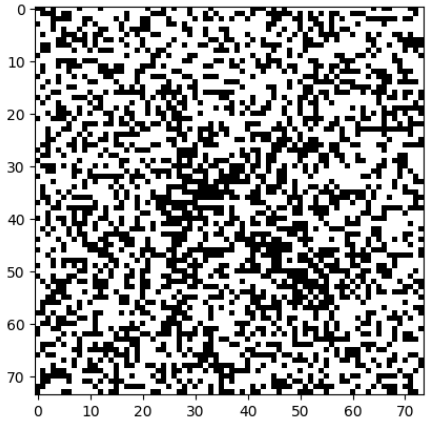


Figura 4. Muestra clasificada en la categoría" k" del conjunto de pruebas con ruido inducido con un factor de 0.75.

El conjunto de entrenamiento principal de muestras se utilizará de forma directa para la implementación de la red Hopfield ya que no se requiere una etiqueta relacionada con el vector, solo las muestras que alimentarán la memoria de la red. En caso contrario (para LAM y BAM), se asociará cada muestra del alfabeto con una etiqueta generada manualmente en donde se indica en una expresión binaria de 5 bits el índice de la clase a la que pertenece cada muestra, tal y como se muestra en la siguiente tabla:

Etiqueta	Valor
00000	a
00001	b
00010	c
00011	d
00100	e
00101	f

00110	g
00111	h
01000	i
01001	j
01010	k
01011	l
01100	m
01101	n
01110	o
01111	p
10000	q
10001	r
10010	s
10011	t
10100	u
10101	v
10110	w
10111	x
11000	y
11001	z

Tabla 1. Asociaciones entre Vectores de Etiquetas y Categorías.

Posterior a este procesamiento se redujo el conjunto de muestras a únicamente 4, en este caso del índice 10 al 14, siendo estos las categorías “k”, “l”, “m” y “n”. La decisión anterior se tomó después de algunas ejecuciones realizadas al utilizar una cantidad mayor de muestras del alfabeto, siendo que se observó como resultado que las 3 redes identifican al menos 2 categorías con el diseño más caprichoso y caen en un sesgo donde utilizan estas 2 categorías para clasificar o identificar todos los patrones restantes. Lo anterior se le puede atribuir a que la técnica de vectorización utilizada para procesar las muestras es un simple aplanado de matrices por lo que las redes no son capaces de memorizar contornos ni detalles más precisos, por lo que generalizar se vuelve complicado para los algoritmos.

A continuación, se muestra la experimentación realizada para cada algoritmo de red neuronal asociativa:

A. Red Neuronal de Hopfield.

1) Función de Activación Escalón Simétrico

a) Predicción Utilizando Muestras de Entrenamiento

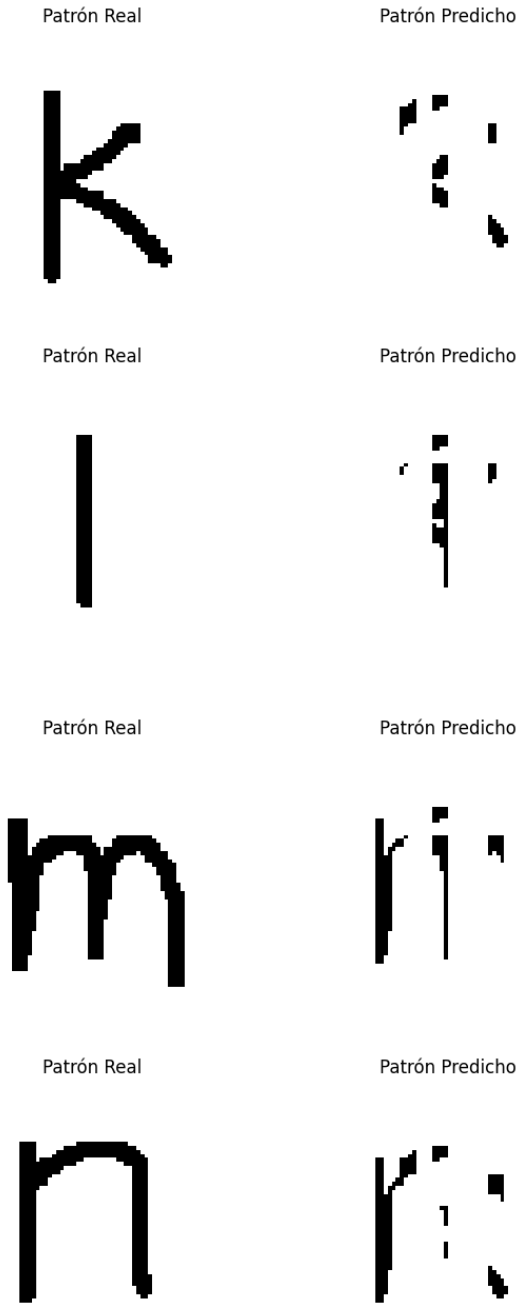


Figura 5. Resultados de la predicción para muestras del conjunto de entrenamiento.

En los resultados anteriores se puede observar que el desempeño de la red de Hopfield fue medianamente aceptable, siendo que, para cada una de las letras, logró asociar las características memorizadas más significativas para cada una de ellas, aunque no haya predicho el patrón completo y enteramente similar al original. Se puede observar este hallazgo en la letra “m” donde la red logró asociarla con un patrón que presenta los detalles más característicos de la letra,

tales como la extensión superior izquierda y la inferior del medio.

b) Predicción Utilizando Muestras de Prueba (Ruido de Forma)

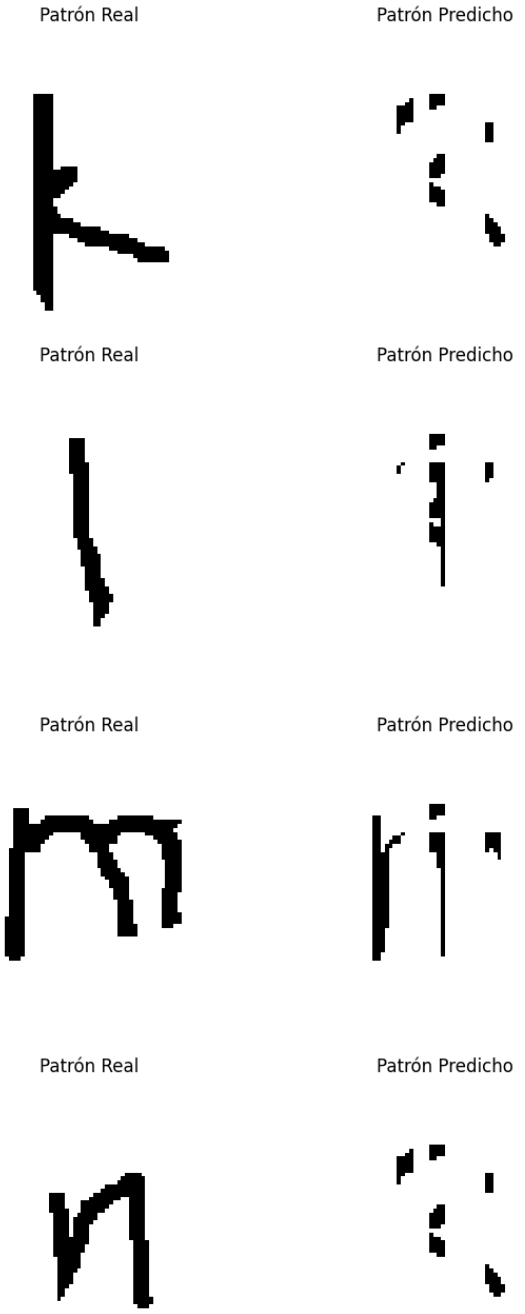
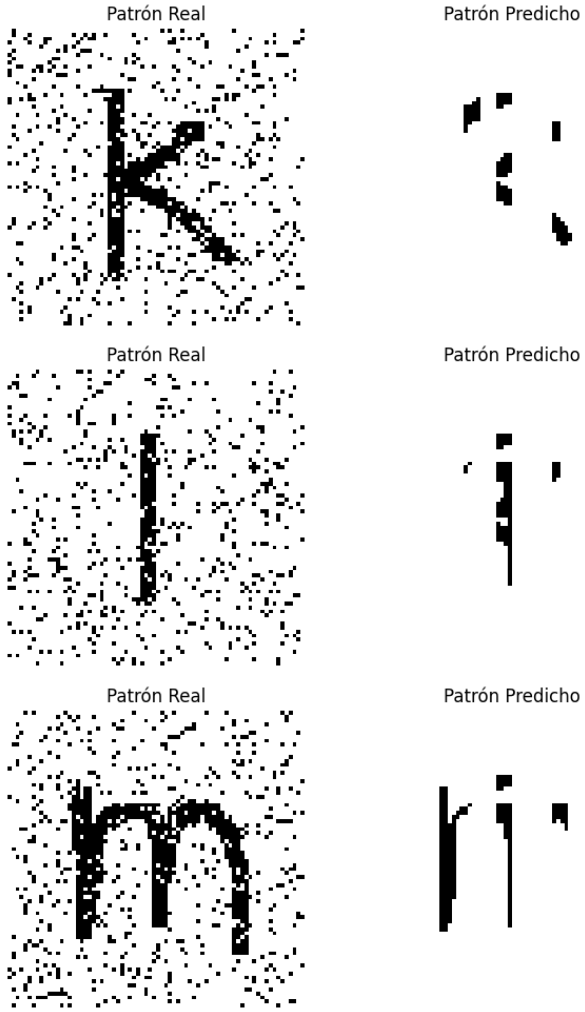


Figura 6. Resultados de la predicción para muestras del conjunto de prueba (Ruido de Forma).

En este caso, se puede observar un comportamiento similar al del experimento anterior, siendo que la red intentó asociar las muestras ruidosas con su carácter original más parecido, en algunos casos logrando mostrar sus características más significativas. Tal es el ejemplo de la letra “k”, para la cual logro identificar las dos extensiones diagonales de forma muy general. Sin embargo, hay mayor grado de confusión para letras más caprichosas en su ruido como el ejemplo de la letra “n”.

Con este experimento se puede comprobar que la red de Hopfield se basa de forma precisa en los patrones que guarda en su memoria, y aunque es capaz de asociar muestras con patrones similares, cuando estas difieren en características significativas con los patrones memorizados, ya no logran ser correctamente clasificados.

c) Predicción Utilizando Muestras de Entrenamiento Induciendo Ruido Bajo un Factor de Aleatoriedad 0.2



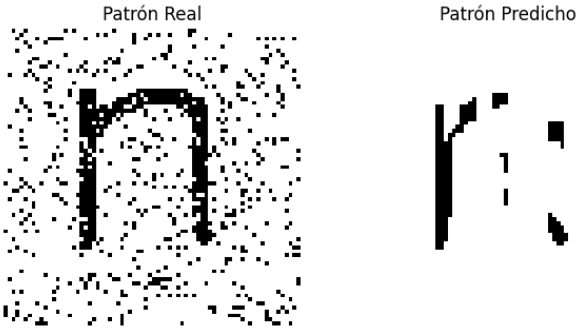
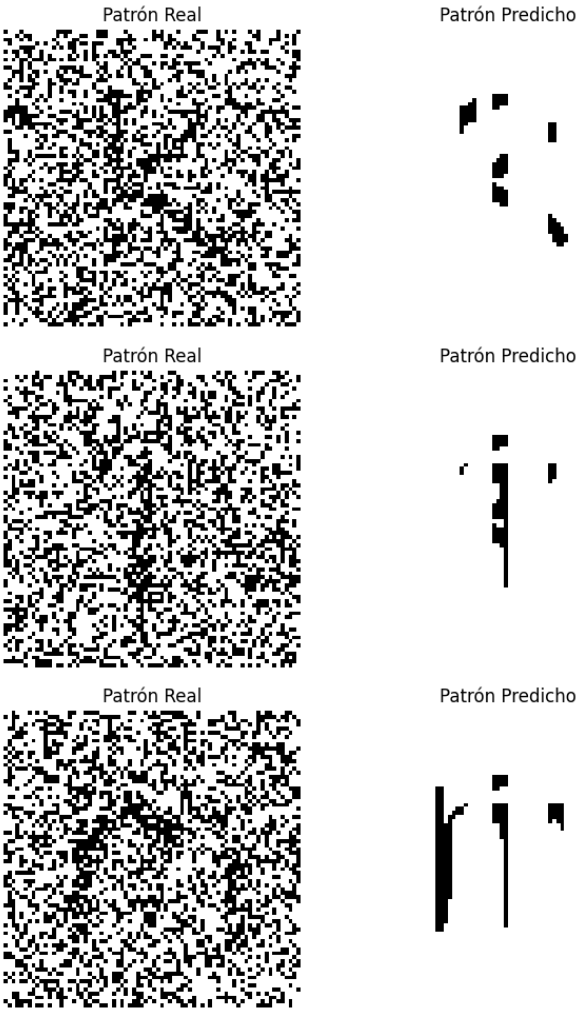


Figura 7. Resultados de la predicción para muestras del conjunto de prueba (Ruido Inducido con un Factor de 0.2).

Figura 8. Resultados de la predicción para muestras del conjunto de prueba (Ruido Inducido con un Factor de 0.75).

Para el experimento con muestras ruidosas, se puede apreciar como la red sigue obteniendo los mismos resultados que el primer experimento (sin ruido). Esto demuestra la alta capacidad de la red para identificar patrones significativos a pesar del ruido a bajo nivel.

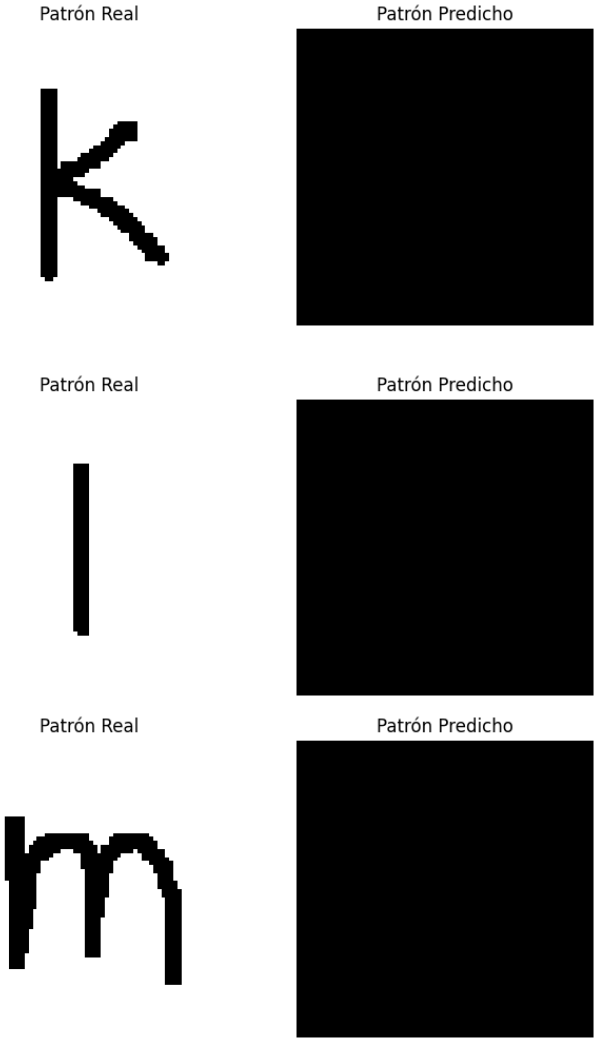
d) Predicción Utilizando Muestras de Entrenamiento Induciendo Ruido Bajo un Factor de Aleatoriedad 0.75



Para el experimento con muestras ruidosas con un factor de 0.75, se puede apreciar como la red continúa obteniendo los mismos resultados que el primer experimento (sin ruido) y que el experimento anterior (ruido inducido con factor de 0.2). Esto demuestra la alta capacidad de la red para identificar patrones significativos a pesar del ruido a muy alto nivel.

2) Función de Activación Escalón Asimétrico

a) Predicción Utilizando Muestras de Entrenamiento



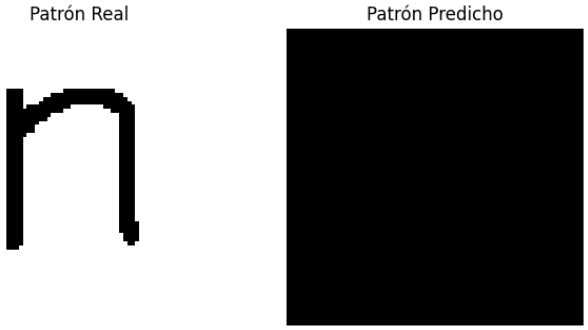


Figura 9. Resultados de la predicción para muestras del conjunto de entrenamiento.

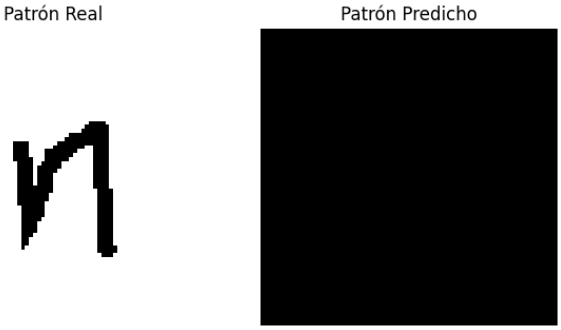
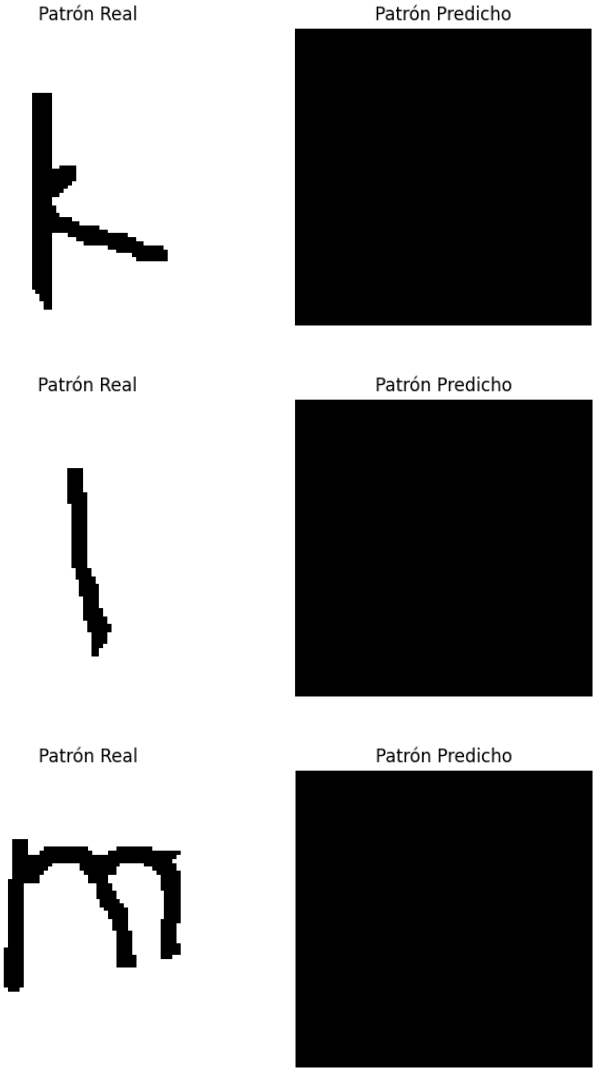
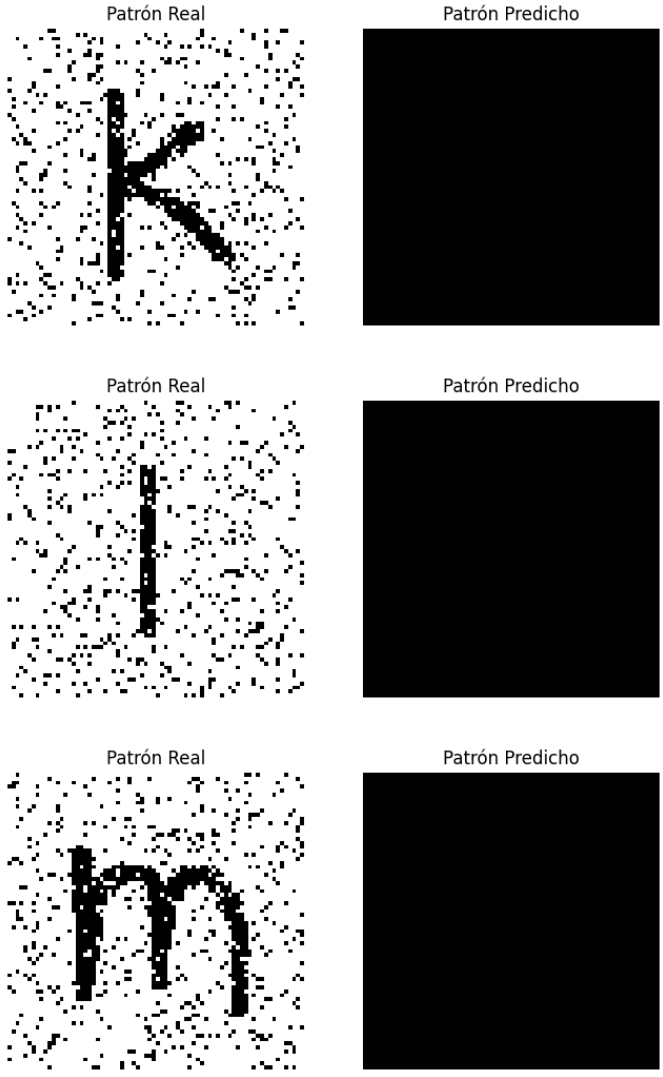


Figura 10. Resultados de la predicción para muestras del conjunto de prueba (Ruido de Forma).

b) Predicción Utilizando Muestras de Prueba (Ruido de Forma)



c) Predicción Utilizando Muestras de Entrenamiento Induciendo Ruido Bajo un Factor de Aleatoriedad 2.0



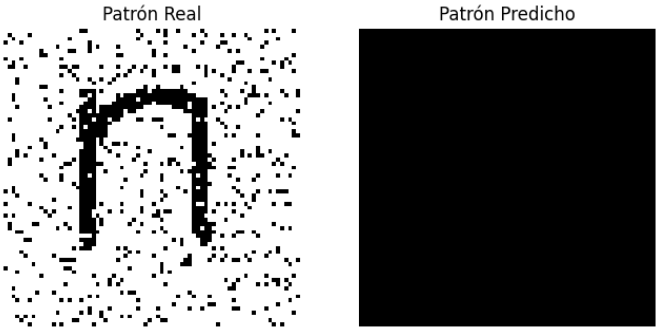


Figura 11. Resultados de la predicción para muestras del conjunto de prueba (Ruido Inducido con un Factor de 0.2).

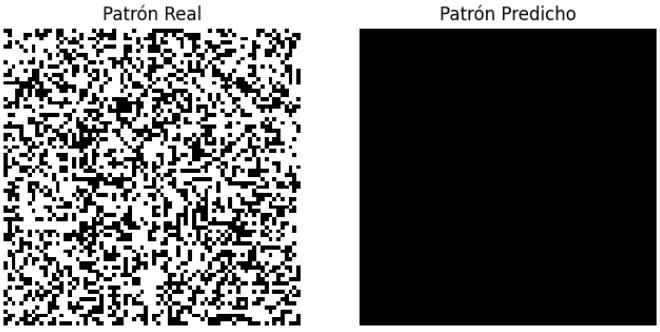
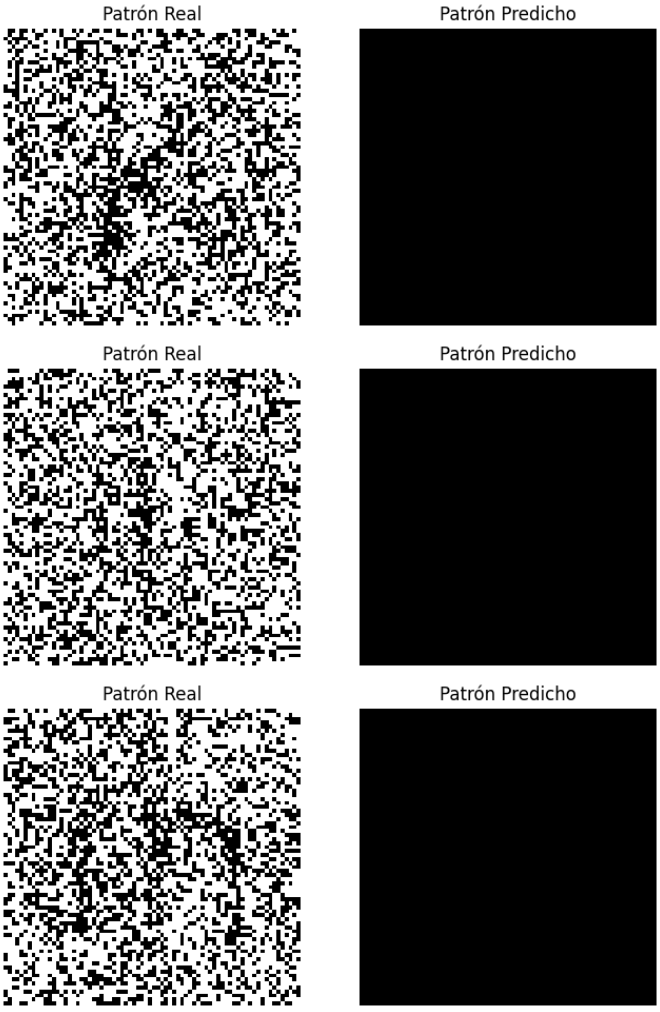


Figura 12. Resultados de la predicción para muestras del conjunto de prueba (Ruido Inducido con un Factor de 0.75).

d) Predicción Utilizando Muestras de Entrenamiento Induciendo Ruido Bajo un Factor de Aleatoriedad 0.75



Los resultados para todos los experimentos realizados con la Red de Hopfield, aplicando la función de activación escalón asimétrica, fueron completamente ineficientes. Siendo que absolutamente todas las predicciones resultaron en vectores contenidos con pixeles activos por lo que las imágenes resultantes fueron enteramente monocromáticas y sin sentido.

De estos experimentos se puede concluir que la red de Hopfield es de completa inutilidad para muestras activadas con la función escalón asimétrico (utilizando las condiciones propuestas para este proyecto).

La teoría propuesta para explicar este fenómeno es que, al tener únicamente valores positivos 0 o 1 en las muestras y siguiendo la fórmula para construir la matriz de pesos para la red de Hopfield, esta matriz siempre va a contener valores positivos. Posteriormente, al intentar clasificar algún patrón sobre la memoria de la red y utilizar la fórmula u = W*input.T, se le aplicará la función de activación (escalón asimétrica) la cual activará los valores resultantes del producto matricial obteniendo únicamente vectores con 1 para todos sus valores, siendo que según la notación de la función de activación escalón asimétrica, cualquier valor presentado mayor o igual a 0, se activará como 1.

B. Red Neuronal LAM (Memoria Asociativa Lineal)

1) Función de Activación Escalón Simétrico

a) Predicción Utilizando Muestras de Entrenamiento

```
================= Test:  0
Tag Real...
[-1  1 -1  1 -1]
k

Tag Predicho...
[-1.  1. -1. -1. -1.]
i
================= Test:  1
Tag Real...
[-1  1 -1  1  1]
l

Tag Predicho...
[-1.  1. -1. -1. -1.]
```

```
i
================= Test:  2
Tag Real...
[-1  1  1 -1 -1]
m

Tag Predicho...
[-1.  1. -1. -1. -1.]
i
================= Test:  3
Tag Real...
[-1  1  1 -1  1]
n

Tag Predicho...
[-1.  1. -1. -1. -1.]
i
```

b) Predicción Utilizando Muestras de Prueba (Ruido de Forma)

```
================= Test:  0
Tag Real...
[-1  1 -1  1 -1]
k

Tag Predicho...
[-1.  1. -1. -1. -1.]
i
================= Test:  1
Tag Real...
[-1  1 -1  1  1]
l

Tag Predicho...
[-1.  1. -1. -1. -1.]
i
================= Test:  2
Tag Real...
[-1  1  1 -1 -1]
m

Tag Predicho...
[-1.  1. -1. -1. -1.]
i
================= Test:  3
Tag Real...
[-1  1  1 -1  1]
n

Tag Predicho...
[-1.  1. -1. -1. -1.]
i
```

c) Predicción Utilizando Muestras de Entrenamiento Induciendo Ruido Bajo un Factor de Aleatoriedad 0.2

```
================= Test:  0
Tag Real...
[-1  1 -1  1 -1]
k

Tag Predicho...
[-1.  1. -1. -1. -1.]
i
================= Test:  1
Tag Real...
[-1  1 -1  1  1]
l

Tag Predicho...
[-1.  1. -1. -1. -1.]
i
================= Test:  2
Tag Real...
[-1  1  1 -1 -1]
m

Tag Predicho...
[-1.  1. -1. -1. -1.]
i
================= Test:  3
Tag Real...
[-1  1  1 -1  1]
n

Tag Predicho...
[-1.  1. -1. -1. -1.]
i
```

d) Predicción Utilizando Muestras de Entrenamiento Induciendo Ruido Bajo un Factor de Aleatoriedad 0.75

```
================= Test:  0
Tag Real...
[-1  1 -1  1 -1]
k

Tag Predicho...
[-1.  1. -1. -1. -1.]
i
================= Test:  1
Tag Real...
[-1  1 -1  1  1]
l

Tag Predicho...
[-1.  1. -1. -1. -1.]
i
================= Test:  2
Tag Real...
[-1  1  1 -1 -1]
m

Tag Predicho...
[-1.  1. -1. -1. -1.]
i
================= Test:  3
Tag Real...
[-1  1  1 -1  1]
n
```

Tag Predicho...
[-1. 1. -1. -1. -1.]
I

Resultados Finales para LAM Simétrico

Condición de Entrada	Patrones Correctos	Total Patrones	Exactitud (%)
Entrenamiento (sin ruido)	0	4	0
Prueba (ruido de forma)	0	4	0
Entrenamiento (ruido 0.2)	0	4	0
Entrenamiento (ruido 0.75)	0	4	0

Tabla 2. Desempeño de LAM con Función Escalón Simétrico

2) Función de Activación Escalón Asimétrico

a) Predicción Utilizando Muestras de Entrenamiento

========== Test: 0
Tag Real...
[0 1 0 1 0]
k

Tag Predicho...
[0. 1. 0. 1. 0.]
k
========== Test: 1
Tag Real...
[0 1 0 1 1]
l

Tag Predicho...
[0. 1. 0. 1. 1.]
l
========== Test: 2
Tag Real...
[0 1 1 0 0]
m

Tag Predicho...
[0. 1. 1. 0. 0.]
m
========== Test: 3
Tag Real...
[0 1 1 0 1]
n

Tag Predicho...
[0. 1. 1. 0. 1.]
n

b) Predicción Utilizando Muestras de Prueba (Ruido de Forma)

========== Test: 0
Tag Real...
[0 1 0 1 0]
k

Tag Predicho...
[0. 1. 0. 1. 1.]
l
========== Test: 1
Tag Real...
[0 1 0 1 1]
l

Tag Predicho...
[0. 1. 0. 1. 1.]
l
========== Test: 2
Tag Real...
[0 1 1 0 0]
m

Tag Predicho...
[0. 1. 1. 0. 1.]
n
========== Test: 3
Tag Real...
[0 1 1 0 1]
n

Tag Predicho...
[0. 1. 0. 1. 1.]
l

c) Predicción Utilizando Muestras de Entrenamiento Induciendo Ruido Bajo un Factor de Aleatoriedad 0.2

========== Test: 0
Tag Real...
[0 1 0 1 0]
k

Tag Predicho...
[0. 1. 0. 1. 0.]
k
========== Test: 1
Tag Real...
[0 1 0 1 1]
l

Tag Predicho...
[0. 1. 0. 1. 1.]
l
========== Test: 2

Tag Real...
[0 1 1 0 0]
m

Tag Predicho...
[0. 1. 1. 0. 0.]
m
========== Test: 3
Tag Real...
[0 1 1 0 1]
n

Tag Predicho...
[0. 1. 1. 0. 1.]
n

d) Predicción Utilizando Muestras de Entrenamiento Induciendo Ruido Bajo un Factor de Aleatoriedad 0.75

========== Test: 0
Tag Real...
[0 1 0 1 0]
k

Tag Predicho...
[0. 1. 0. 1. 0.]
k
========== Test: 1
Tag Real...
[0 1 0 1 1]
l

Tag Predicho...
[0. 1. 0. 1. 1.]
l
========== Test: 2
Tag Real...
[0 1 1 0 0]
m

Tag Predicho...
[0. 1. 1. 0. 0.]
m
========== Test: 3
Tag Real...
[0 1 1 0 1]
n

Tag Predicho...
[0. 1. 1. 0. 1.]
N

Resultados Finales para LAM Asimétrico

Condición de Entrada	**Patrones Correctos**	**Total Patrones**	**Exactitud (%)**
Entrenamiento (sin ruido)	4	4	100
Prueba (ruido de forma)	1	4	25
Entrenamiento (ruido 0.2)	4	4	100
Entrenamiento (ruido 0.75)	4	4	100

Tabla 3. Desempeño de LAM con Función Escalón Asimétrico

C. Red Neuronal BAM (Memoria Asociativa Bidireccional)

1) Función de Activación Escalón Simétrico

a) Predicción Utilizando Muestras de Entrenamiento

========== Test: 0
Tag Real...
[-1 1 -1 1 -1]
k

Tag Predicho...
[-1. 1. -1. 1. -1.]
k
========== Test: 1
Tag Real...
[-1 1 -1 1 1]
l

Tag Predicho...
[-1. 1. -1. 1. 1.]
l
========== Test: 2
Tag Real...
[-1 1 1 -1 -1]
m

Tag Predicho...
[-1. 1. 1. -1. -1.]
m
========== Test: 3
Tag Real...
[-1 1 1 -1 1]
n

Tag Predicho...
[-1. 1. 1. -1. 1.]
n

b) Predicción Utilizando Muestras de Prueba (Ruido de Forma)

========== Test: 0
Tag Real...
[-1 1 -1 1 -1]
k

Tag Predicho...

```
[-1.  1. -1.  1.  1.]
l
 ================= Test:  1
Tag Real...
[-1  1 -1  1  1]
l

Tag Predicho...
[-1.  1. -1.  1.  1.]
l
 ================= Test:  2
Tag Real...
[-1  1  1 -1 -1]
m

Tag Predicho...
[-1.  1.  1. -1.  1.]
n
 ================= Test:  3
Tag Real...
[-1  1  1 -1  1]
n

Tag Predicho...
[-1.  1. -1.  1.  1.]
l
```

c) Predicción Utilizando Muestras de Entrenamiento Induciendo Ruido Bajo un Factor de Aleatoriedad 2.0

```
================= Test:  0
Tag Real...
[-1  1 -1  1 -1]
k

Tag Predicho...
[-1.  1. -1.  1. -1.]
k
================= Test:  1
Tag Real...
[-1  1 -1  1  1]
l

Tag Predicho...
[-1.  1. -1.  1.  1.]
l
================= Test:  2
Tag Real...
[-1  1  1 -1 -1]
m

Tag Predicho...
[-1.  1.  1. -1. -1.]
m
 ================= Test:  3
Tag Real...
[-1  1  1 -1  1]
n

Tag Predicho...
[-1.  1.  1. -1.  1.]
n
```

d) Predicción Utilizando Muestras de Entrenamiento Induciendo Ruido Bajo un Factor de Aleatoriedad 0.75

```
================= Test:  0
Tag Real...
[-1  1 -1  1 -1]
k

Tag Predicho...
[-1.  1. -1.  1. -1.]
K
================= Test:  1
Tag Real...
[-1  1 -1  1  1]
l

Tag Predicho...
[-1.  1. -1.  1.  1.]
l
================= Test:  2
Tag Real...
[-1  1  1 -1 -1]
m

Tag Predicho...
[-1.  1.  1. -1. -1.]
m
 ================= Test:  3
Tag Real...
[-1  1  1 -1  1]
n

Tag Predicho...
[-1.  1.  1. -1.  1.]
n
```

Resultados Finales para BAM Simétrico

Condición de Entrada	**Patrones Correctos**	**Total Patrones**	**Exactitud (%)**
Entrenamiento (sin ruido)	4	4	100
Prueba (ruido de forma)	1	4	25
Entrenamiento (ruido 0.2)	4	4	100
Entrenamiento (ruido 0.75)	4	4	100

Tabla 4. Desempeño de BAM con Función Escalón Simétrico

2) Función de Activación Escalón Asimétrico

a) Predicción Utilizando Muestras de Entrenamiento

```
================= Test:  0
Tag Real...
[0 1 0 1 0]
k

Tag Predicho...
[1. 1. 1. 1. 1.]
Patron Nuevo
================= Test:  1
Tag Real...
[0 1 0 1 1]
l

Tag Predicho...
[1. 1. 1. 1. 1.]
Patron Nuevo
================= Test:  2
Tag Real...
[0 1 1 0 0]
m

Tag Predicho...
[1. 1. 1. 1. 1.]
Patron Nuevo
================= Test:  3
Tag Real...
[0 1 1 0 1]
n

Tag Predicho...
[1. 1. 1. 1. 1.]
Patron Nuevo
```

b) Predicción Utilizando Muestras de Prueba (Ruido de Forma)

```
================= Test:  0
Tag Real...
[0 1 0 1 0]
k

Tag Predicho...
[1. 1. 1. 1. 1.]
Patron Nuevo
================= Test:  1
Tag Real...
[0 1 0 1 1]
l

Tag Predicho...
[1. 1. 1. 1. 1.]
Patron Nuevo
================= Test:  2
Tag Real...
[0 1 1 0 0]
m

Tag Predicho...
[1. 1. 1. 1. 1.]
Patron Nuevo
================= Test:  3
Tag Real...
[0 1 1 0 1]
n

Tag Predicho...
[1. 1. 1. 1. 1.]
Patron Nuevo
```

c) Predicción Utilizando Muestras de Entrenamiento Induciendo Ruido Bajo un Factor de Aleatoriedad 0.2

```
================= Test:  0
Tag Real...
[0 1 0 1 0]
k

Tag Predicho...
[1. 1. 1. 1. 1.]
Patron Nuevo
================= Test:  1
Tag Real...
[0 1 0 1 1]
l

Tag Predicho...
[1. 1. 1. 1. 1.]
Patron Nuevo
================= Test:  2
Tag Real...
[0 1 1 0 0]
m

Tag Predicho...
[1. 1. 1. 1. 1.]
Patron Nuevo
================= Test:  3
Tag Real...
[0 1 1 0 1]
n

Tag Predicho...
[1. 1. 1. 1. 1.]
Patron Nuevo
```

d) Predicción Utilizando Muestras de Entrenamiento Induciendo Ruido Bajo un Factor de Aleatoriedad 0.75

```
================= Test:  0
Tag Real...
[0 1 0 1 0]
k

Tag Predicho...
[1. 1. 1. 1. 1.]
Patron Nuevo
```

```
================= Test:  1
Tag Real...
[0 1 0 1 1]
l

Tag Predicho...
[1. 1. 1. 1. 1.]
Patron Nuevo
================= Test:  2
Tag Real...
[0 1 1 0 0]
m

Tag Predicho...
[1. 1. 1. 1. 1.]
Patron Nuevo
================= Test:  3
Tag Real...
[0 1 1 0 1]
n

Tag Predicho...
[1. 1. 1. 1. 1.]
Patron Nuevo
```

Resultados Finales para BAM Asimétrico

Condición de Entrada	Patrones Correctos	Total Patrones	Exactitud (%)
Entrenamiento (sin ruido)	0	4	0
Prueba (ruido de forma)	0	4	0
Entrenamiento (ruido 0.2)	0	4	0
Entrenamiento (ruido 0.75)	0	4	0

Tabla 5. Desempeño de BAM con Función Escalón Asimétrico

D. Análisis Comparativo entre LAM y BAM

Así como la red de Hopfield, las redes neuronales asociativas LAM y BAM fueron evaluadas en condiciones de entrenamiento y prueba utilizando diferentes tipos de perturbaciones, a fin de analizar su robustez en la recuperación de patrones. Se consideraron patrones de entrenamiento sin ruido, patrones de entrenamiento con ruido inducido mediante factores de aleatoriedad de 0.2 y 0.75. así como patrones de prueba con ruido de forma. En todos los casos, las funciones de activación empleadas coincidieron con la codificación de entrada, de modo que los resultados reflejan propiedades inherentes a la dinámica de cada red y no incompatibilidades de representación. La métrica principal utilizada fue la exactitud de recuperación, definida como el cociente entre el número de patrones correctamente recuperados y el número total de patrones, expresado en porcentaje.

En la red BAM, la función de activación escalón simétrico (-1/+1) permitió la recuperación perfecta de los patrones de entrenamiento, tanto en ausencia de ruido como con ruido inducido en factores de 0.2 y 0.75. En todos estos casos, la exactitud alcanzó el 100%, lo que demuestra que la red posee mecanismos de corrección frente a perturbaciones internas. En contraste, cuando se aplicaron patrones de prueba con ruido de forma, la exactitud se redujo drásticamente al 25%. Este hallazgo evidencia la diferencia entre perturbaciones internas y externas. Mientras que las internas consisten en alteraciones parciales dentro de un mismo espacio de representación y pueden ser corregidas mediante la redundancia de la matriz de pesos, las perturbaciones externas modifican la estructura global del patrón, impidiendo que la dinámica iterativa converja hacia los atractores previamente establecidos.

Con la función de activación escalón asimétrico (0/1), la BAM no logró recuperar ningún patrón en ninguna condición. La exactitud fue del 0% y las salidas consistieron en vectores uniformes de unos. Este fenómeno corresponde a un caso de saturación. Debido a que la matriz de pesos en ambas arquitecturas se construyó de manera consistente con la misma codificación y función de activación empleadas en las entradas y salidas, los fenómenos observados no corresponden a incompatibilidades de representación, sino a propiedades emergentes de la dinámica de la red. En BAM con activación asimétrica, la acumulación de términos positivos en la suma ponderada genera un sesgo que induce la activación uniforme de todas las neuronas, produciendo un estado saturado compuesto exclusivamente por unos.

Los resultados en LAM mostraron un comportamiento inverso. Con activación escalón asimétrica, la red alcanzó exactitud del 100% en los patrones de entrenamiento sin ruido y con ruido inducido en factores de 0.2 y 0.75. Sin embargo, al introducir ruido de forma en las pruebas, la exactitud cayó al 25%, reflejando la misma sensibilidad observada en BAM frente a perturbaciones externas que modifican la geometría de los patrones. Por el contrario, con activación escalón simétrica, LAM no logró recuperar ningún patrón en ninguna condición, alcanzando exactitud del 0%. Todos los vectores de salida convergieron al mismo patrón dominante, correspondiente a la letra “i” en los experimentos realizados. Este resultado representa un segundo caso de saturación, en este caso caracterizado por convergencia uniforme a un único mínimo energético. La combinación de pesos lineales bipolares y función de activación simétrica establece umbrales que favorecen un estado estable global, eliminando la capacidad de diferenciar entre patrones almacenados.

La comparación entre ambos modelos confirma que la tolerancia al ruido depende directamente de la naturaleza de la perturbación. Los resultados demuestran que las perturbaciones internas, al no alterar la estructura fundamental del patrón, son corregidas eficazmente por ambas

arquitecturas bajo la activación adecuada, alcanzando exactitud del 100%. En contraste, las perturbaciones externas producen una disminución sistemática de la exactitud al 25%, lo que revela que la capacidad de generalización es limitada cuando la forma del patrón de entrada difiere significativamente de los almacenados.

Finalmente, los dos fenómenos de saturación identificados representan limitaciones críticas en la memoria asociativa. En BAM con activación asimétrica, la saturación se manifiesta en vectores uniformes producto de una dinámica que privilegia salidas positivas Por otro lado, en LAM con activación simétrica, la saturación consiste en convergencia a un único patrón dominante. Ambos casos ilustran que la elección de la función de activación no solo debe coincidir con la codificación de entrada, sino también ser congruente con la forma en que los pesos de la red distribuyen la información y definen los mínimos energéticos.

V. CONCLUSIÓN

El análisis experimental realizado sobre las redes neuronales asociativas de Hopfield, LAM y BAM permite establecer conclusiones fundamentales acerca del impacto de la función de activación, la codificación de los patrones y la naturaleza de las perturbaciones en la capacidad de recuperación.

En la red de Hopfield, la función de activación escalón simétrico permitió la asociación parcial de patrones de entrenamiento, capturando las características más relevantes de las letras aun en condiciones de ruido inducido con factores de 0.2 y 0.75. Este comportamiento confirma la robustez de Hopfield frente a perturbaciones internas de baja y alta magnitud. Sin embargo, en presencia de ruido estructural (ruido de forma), la exactitud disminuyó levemente debido a que la red se sustenta estrictamente en los patrones previamente almacenados, mostrando limitaciones sutiles frente a modificaciones externas que alteran la geometría de los vectores. En contraste, con activación escalón asimétrica, la red de Hopfield falló en todos los escenarios, generando vectores saturados en valores uno. Este resultado evidencia que, bajo la construcción de pesos bipolares, la activación asimétrica destruye la diferenciación entre patrones y anula la capacidad de recuperación.

En la red BAM, la activación simétrica produjo exactitud del 100% en patrones de entrenamiento y en entrenamiento con ruido inducido, mostrando gran tolerancia a perturbaciones internas gracias a la redundancia de la matriz de pesos. Sin embargo, la exactitud cayó al 25% con ruido de forma, confirmando que la arquitectura es sensible a perturbaciones externas que desfiguran la estructura global de los patrones. Con activación asimétrica, la red se saturó en vectores uniformes de unos, anulando su capacidad asociativa.

Por su parte, la red LAM presentó un comportamiento inverso. Con activación asimétrica alcanzó exactitud del 100% en entrenamiento sin ruido y con ruido inducido, mientras que con ruido de forma la exactitud se redujo al 25%. Con activación simétrica, la red colapsó hacia un único patrón dominante, mostrando saturación estructural producto de la interacción entre la matriz de pesos bipolares y los umbrales definidos por la activación.

En términos generales, los experimentos confirman tres principios centrales. Primero, la robustez frente a perturbaciones internas se mantiene en las tres arquitecturas bajo la configuración adecuada, alcanzando un adecuado nivel de exactitud incluso con niveles altos de ruido inducido. Segundo, la vulnerabilidad frente a perturbaciones externas es consistente en todas las redes, con reducciones sistemáticas de exactitud (siendo mayores en LAM y BAM que en Hopfield), lo que limita la capacidad de generalización frente a modificaciones estructurales no vistas en el entrenamiento. Tercero, la saturación inducida por la dinámica propia de cada red representa una limitación crítica. En Hopfield y BAM con activación asimétrica, la salida converge trivialmente hacia vectores uniformes, eliminando la capacidad de diferenciación. En LAM con activación simétrica, la red colapsa hacia un único mínimo energético, lo que genera convergencia uniforme a un patrón dominante. Estos resultados muestran que, incluso bajo codificación y activación consistentes, la interacción entre la estructura de la matriz de pesos y la dinámica impuesta por la función de activación puede conducir a pérdidas de capacidad asociativa.

En conclusión, las configuraciones más estables fueron Hopfield y BAM con activación simétrica, y LAM con activación asimétrica, las cuales demostraron robustez frente al ruido inducido y un desempeño aceptable frente al ruido estructural. Los hallazgos obtenidos evidencian los efectos de saturación inherentes a cada arquitectura y establecen las bases para futuras investigaciones orientadas a mejorar la capacidad de generalización y minimizar la pérdida de diferenciación de patrones en memorias asociativas.

REFERENCIAS

[1] S. Basak, S. Pal, and D. C. Patranabis, “Artificial Neural Network: Understanding the Basic Concepts without Mathematics,” J. Adv. Res., vol. 7, no. 5, pp. 701–713, 2016. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC6428006/

[2] A. Mazumdar, Associative Memory via a Sparse Recovery Model, University of California, San Diego, 2013. [Online]. Available: https://mazumdar.ucsd.edu/papers/Associative-final-main.pdf

[3] M. Pehlevan and D. Chklovskii, “Hebbian Learning from First Principles,” arXiv:2401.07110, Jan. 2024. [Online]. Available: https://arxiv.org/abs/2401.07110

[4] Y. Zhang and A. Willsky, “Parameter Estimation by Reduced-Order Linear Associative Memory,” IEEE Trans. Neural Netw., vol. 5, no. 3, pp. 423–432, May 1994. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/9125812/

[5] G. G. Chowdhury and S. K. Pal, “Bidirectional Associative Memories: Different Approaches,” ACM Trans. Intell. Syst. Technol., vol. 7, no. 2, pp. 1–22, Feb. 2016. [Online]. Available: https://dl.acm.org/doi/10.1145/2431211.2431217

[6] J. J. Hopfield, “Neural Networks and Physical Systems with Emergent Collective Computational Abilities,” Proc. Natl. Acad. Sci. USA, vol.

79, no. 8, pp. 2554–2558, Apr. 1982. [Online]. Available: https://www.pnas.org/doi/10.1073/pnas.79.8.2554

[7] L. Wang, M. Jiang, R. Liu, and X. Tang, "Comparison of BAM and discrete Hopfield networks with CPN for processing of noisy data," in 2008 9th International Conference on Signal Processing (ICOSP), Beijing, China, 2008, pp. 1708–1711. [Online]. Available: https://doi.org/10.1109/ICOSP.2008.4697466

[8] J. Y.-C. Hu, D. Yang, D. Wu, C. Xu, B.-Y. Chen, and H. Liu, "On Sparse Modern Hopfield Model," 2023. [Online]. Available: https://arxiv.org/abs/2309.12673

[9] D. Krotov and J. Hopfield, "Dense Associative Memory for Pattern Recognition," Advances in Neural Information Processing Systems (NeurIPS), vol. 29, pp. 1172–1180, 2016. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/file/eaae339c4d89fc102edd9dbdb6a28915-Paper.pdf

[10] V. Cherkassky, K. Fassett, and N. Vassilas, "Linear algebra approach to neural associative memories and noise performance of neural classifiers," *IEEE Trans. Comput.*, vol. 40, no. 12, pp. 1429–1435, Jan. 1992. [Online]. Available: https://www.researchgate.net/publication/3042887_Linear_Algebra_Approach_to_Neural_Associative_Memories_and_Noise_Performance_of_Neural_Classifiers

[11] P. Dayan and D. J. Willshaw, "Optimising synaptic learning rules in linear associative memories," Biological Cybernetics, vol. 65, no. 4, pp. 253–265, Aug. 1991. [Online]. Available: https://doi.org/10.1007/BF00206223

[12] W. Soyan, Omniglot handwritten Latin alphabet dataset, Kaggle. [Online]. Available: https://www.kaggle.com/datasets/watesoyan/omniglot