# Integración de memristores en redes neuronales para la clasificación del dataset MNIST utilizando TensorFlow

1st Ricardo Aldair Tirado Torres
*Centro de Investigación en Computación*
*Instituto Politécnico Nacional*
Ciudad de México, México
rtiradot2023@cic.ipn.mx

2nd Ricardo Barrón Fernández
*Centro de Investigación en Computación*
*Instituto Politécnico Nacional*
Ciudad de México, México
rbarron@cic.ipn.mx

***Resumen*—El cómputo neuromórfico, con sus redes neuronales pulsantes, el aprendizaje STDP y los memristores, representa un paradigma revolucionario en la informática. La integración de estos conceptos en la tecnología CMOS no solo promete un salto significativo en la eficiencia y capacidad de las máquinas, sino que también nos acerca un paso más a replicar la asombrosa complejidad y eficiencia del cerebro humano. Esta tecnología abre un abanico de posibilidades en el campo de la inteligencia artificial y más allá, con el potencial de transformar la forma en que interactuamos con las máquinas y el mundo que nos rodea. En este trabajo se utiliza al nodo tecnológico de SKY130, un proceso de fabricación que integra los elementos necesarios para diseñar una red neuronal pulsante que integra el concepto del aprendizaje STDP no supervisado.**

***Index Terms*—Redes neuronales artificiales, sinapsis memristiva, clasificación de patrones, memristor HP.**

## I. Introducción

Las redes neuronales artificiales (ANNs, por sus siglas en inglés) son modelos computacionales inspirados en la estructura y el funcionamiento del cerebro humano, diseñados para realizar tareas de reconocimiento de patrones, clasificación y predicción. Estas redes consisten en capas de neuronas artificiales conectadas entre sí, donde cada conexión tiene un peso asociado que se ajusta durante el proceso de entrenamiento. El entrenamiento de una ANN implica la optimización de estos pesos mediante algoritmos de aprendizaje, siendo el descenso de gradiente uno de los métodos más comunes. Una vez entrenada, la red puede realizar inferencias sobre datos nuevos, aplicando los pesos optimizados para generar predicciones o clasificaciones.

En el ámbito de la inteligencia artificial, las ANNs se utilizan extensamente para la clasificación de patrones, un proceso crucial en aplicaciones como el reconocimiento de imágenes, el procesamiento del lenguaje natural y la detección de fraudes. El aprendizaje profundo, una subrama del aprendizaje automático que se basa en ANNs con múltiples capas (redes neuronales profundas), ha revolucionado estos campos, logrando avances significativos en precisión y eficiencia.

Los memristores han emergido como una tecnología prometedora para implementar pesos sinápticos en redes neuronales artificiales. Un memristor es un componente electrónico cuya resistencia puede ser ajustada y memorizada, comportándose de manera análoga a una sinapsis biológica. En el contexto del aprendizaje máquina y el aprendizaje profundo, los memristores ofrecen una solución eficiente para el almacenamiento y ajuste de pesos sinápticos, permitiendo la creación de hardware neuromórfico que puede ejecutar tareas de inferencia y entrenamiento de manera más rápida y con menor consumo de energía en comparación con los sistemas tradicionales basados en transistores.

El memristor de HP, desarrollado por Hewlett-Packard, es uno de los primeros y más conocidos dispositivos de este tipo, capaz de recordar su estado resistivo incluso después de apagar la energía. Este atributo lo hace ideal para su uso en ANNs, donde puede representar y ajustar eficientemente los pesos sinápticos, facilitando así la implementación de redes neuronales en dispositivos de hardware especializados.

Para evaluar y entrenar redes neuronales, se utilizan conjuntos de datos estandarizados que permiten medir el rendimiento y la precisión del modelo. Uno de los conjuntos de datos más utilizados en la investigación de ANNs es el MNIST (Modified National Institute of Standards and Technology), que contiene 60,000 imágenes de entrenamiento y 10,000 imágenes de prueba de dígitos escritos a mano, cada una con una resolución de 28x28 píxeles. Este conjunto de datos ha sido fundamental para el desarrollo y benchmarking de nuevas arquitecturas de redes neuronales y técnicas de aprendizaje automático.

Este reporte explorará en detalle los conceptos mencionados, comenzando con una revisión de las redes neuronales artificiales, seguido de una discusión sobre el papel de los memristores como pesos sinápticos, destacando el memristor

de HP, y finalizando con un análisis del conjunto de datos MNIST y su relevancia en el campo de la inteligencia artificial, el aprendizaje máquina y el aprendizaje profundo.

A lo largo de este trabajo, discutimos la teoría subyacente de los memristores, detallamos la arquitectura de nuestra red neuronal y evaluamos el desempeño de nuestro modelo en la tarea de clasificación de imágenes. Los resultados obtenidos muestran que la integración de memristores en redes neuronales es una estrategia prometedora que puede contribuir significativamente al desarrollo de sistemas de IA más avanzados y eficientes.

## II. Preliminares

### *II-A. Redes neuronales artificiales*

Las redes neuronales artificiales son modelos computacionales inspirados en la estructura y el funcionamiento del cerebro humano. Estas redes están compuestas por unidades interconectadas llamadas neuronas artificiales, distribuidas típicamente en tres tipos de capas:

- *Capa de entrada*: Recibe las señales iniciales del entorno externo. En el caso de la clasificación de imágenes, cada neurona de la capa de entrada representa un píxel de la imagen.
- *Capas ocultas*: Procesan las señales de entrada mediante una serie de transformaciones no lineales. Estas capas están compuestas por neuronas que aplican funciones de activación a sus entradas ponderadas.
- *Capa de salida*: Genera la salida final del modelo. Para la clasificación, cada neurona en la capa de salida puede representar una clase diferente.

Cada neurona en una red neuronal artificial realiza una operación matemática sobre sus entradas ponderadas y una función de activación. La salida de una neurona $j$ se puede representar matemáticamente como:

$$y_j = \sigma\left(\sum_i w_{ij} x_i + b_i\right) \tag{1}$$

Donde:

- $y_j$ es la salida de la neurona .
- $\sigma$ es la función de activación (por ejemplo, ReLU, sigmoide, tanh).
- $w_{ij}$ es el peso sináptico entre la neurona $i$ de la capa anterior y la neurona $j$.
- $x_i$ es la entrada desde la neurona $i$ de la capa anterior.
- $b_j$ es el sesgo de la neurona $j$

El proceso de entrenamiento de una red neuronal implica ajustar los pesos sinápticos $w_{ij}$ y los sesgos $b_j$ para minimizar una función de pérdida, que mide la diferencia entre las predicciones de la red y los valores reales. Este ajuste se realiza mediante un algoritmo de optimización, como el gradiente descendente, junto con un proceso de retropropagación del error.

Propagación hacia Adelante: Los datos de entrada se propagan a través de la red para generar una predicción. Cálculo de la Pérdida: Se calcula la función de pérdida que mide la discrepancia entre la predicción y la etiqueta real. Retropropagación: El error se propaga hacia atrás a través de la red para calcular los gradientes de la función de pérdida respecto a los pesos y sesgos. Actualización de Pesos: Los pesos y sesgos se actualizan utilizando el algoritmo de optimización.

Propagación hacia Adelante:

Las entradas se pasan a través de la capa de entrada. Las salidas de cada neurona de la capa de entrada se multiplican por los pesos correspondientes y se pasan a las neuronas de la primera capa oculta. Cada neurona de la capa oculta suma sus entradas ponderadas y aplica una función de activación (como ReLU, sigmoide o tanh) para calcular su salida. Este proceso se repite para todas las capas ocultas hasta llegar a la capa de salida. La capa de salida produce la predicción final del modelo. Cálculo de la Pérdida:

La salida de la red se compara con la etiqueta verdadera utilizando una función de pérdida (como la entropía cruzada esparsa para problemas de clasificación). La función de pérdida cuantifica la discrepancia entre la predicción del modelo y el valor verdadero. Retropropagación del Error:

El error calculado se propaga hacia atrás a través de la red. Se calculan los gradientes de la función de pérdida respecto a los pesos y sesgos utilizando el algoritmo de retropropagación. Estos gradientes indican la dirección y magnitud en la que deben ajustarse los pesos para reducir la pérdida. Actualización de Pesos:

Los pesos y sesgos se actualizan utilizando un algoritmo de optimización, como el gradiente descendente o una de sus variantes (por ejemplo, Adam, RMSprop). La actualización de los pesos se realiza para minimizar la función de pérdida y mejorar la precisión del modelo.

Uno de los tipos más simples y ampliamente utilizados de redes neuronales artificiales son las redes neuronales *feed-forward* (FFNN). En una FFNN, las conexiones entre las unidades no forman ciclos, y la información fluye solo en una dirección: desde las capas de entrada, a través de las capas ocultas, y finalmente hacia la capa de salida.

Existen varios tipos de redes neuronales artificiales

### II-B. Memristores

Los memristores (resistores de memoria) son componentes eléctricos pasivos que han sido teorizados durante décadas y que finalmente se desarrollaron experimentalmente en los últimos años. Estos dispositivos se caracterizan por su capacidad para recordar la cantidad de carga que ha pasado a través de ellos, lo que les permite retener un estado resistivo específico incluso después de que se ha eliminado la fuente de energía. Esta propiedad de memoria los hace especialmente adecuados para aplicaciones en almacenamiento de datos no volátil y procesamiento neuromórfico.

La idea de los memristores fue propuesta por primera vez por Leon Chua en 1971. Chua predijo la existencia de un cuarto elemento pasivo fundamental en los circuitos eléctricos, junto con el resistor, el capacitor y el inductor. Este cuarto elemento, el memristor, tiene una relación directa entre el flujo magnético y la carga eléctrica, lo que le confiere sus propiedades únicas. El memristor se define matemáticamente por la ecuación 2 y 3.

$$V = R(w) \cdot I \tag{2}$$

$$\frac{dw}{dt} = f(I) \tag{3}$$

Donde:

- $V$ es la tensión a través del memristor.
- $I$ es la corriente que pasa a través del memristor.
- $R(w)$ es la resistencia dependiente del estado $\omega$, que es una función de la carga histórica.

La capacidad de retener un estado resistivo específico sin necesidad de energía continua. Analogía con las sinapsis: Su comportamiento es similar al de las sinapsis neuronales, donde la resistencia del memristor puede ajustarse de manera análoga a la fuerza sináptica en las redes biológicas. Densidad y eficiencia energética: Ofrecen alta densidad de almacenamiento y baja energía de operación en comparación con los dispositivos convencionales.

La integración de memristores en redes neuronales artificiales ofrece una solución prometedora para mejorar la eficiencia energética y la densidad de almacenamiento de estos sistemas. Los memristores pueden emular las sinapsis biológicas al proporcionar una forma no volátil y analógica de almacenamiento y procesamiento de información. En la siguiente sección, describimos nuestro enfoque para integrar un modelo de memristor de HP en una red neuronal artificial para la clasificación del dataset MNIST, utilizando TensorFlow.

## III. Sistema propuesto

El sistema propuesto integra un modelo de memristor de HP en una red neuronal artificial para la clasificación del dataset MNIST utilizando la herramienta TensorFlow. Este enfoque busca aprovechar las propiedades únicas de los memristores para mejorar la eficiencia energética y la capacidad de almacenamiento del sistema de aprendizaje automático. A continuación, se detalla la arquitectura de la red neuronal, el funcionamiento del programa, el modelo de memristor utilizado y el algoritmo de entrenamiento.

### III-A. Arquitectura de la Red Neuronal

La red neuronal utilizada en este trabajo es una red neuronal feedforward (FFNN) con las siguientes características:

Capa de Entrada: Una capa de entrada que toma imágenes de 28x28 píxeles del dataset MNIST, aplanando cada imagen en un vector de 784 dimensiones. Capa Oculta Personalizada con Memristores: Una capa oculta personalizada que simula el comportamiento de un memristor de HP, con 128 unidades (neuronas). Capa de Activación: Una capa de activación ReLU (Rectified Linear Unit) que introduce no linealidad en el modelo. Capa de Salida: Una capa densa (fully connected) con 10 neuronas, cada una correspondiente a una de las 10 clases de dígitos (0-9) del dataset MNIST. Modelo de Memristor de HP El modelo de memristor de HP se integra en la capa oculta personalizada. La actualización de los pesos sinápticos en esta capa se realiza mediante una ecuación diferencial simplificada que describe la dinámica de los memristores:

$$R(w) = R_{on} \cdot w + R_{off} \cdot (1 - w) \tag{4}$$

$$\frac{dw}{dt} = \mu \cdot \frac{I}{D} \tag{5}$$

Donde:

- $R_{on}$ y $R_{off}$ son las resistencias mínima y máxima del memristor.
- $w$ es una variable de estado que representa la proporción de la capa de $TiO_2$ dopada.
- $\mu$ es la movilidad de los vacantes de oxígeno.
- $D$ es el espesor de la película de $TiO_2$.

Funcionamiento del Programa El programa se implementa en Python utilizando TensorFlow, una biblioteca de código abierto para la construcción y entrenamiento de modelos de aprendizaje profundo. El flujo de trabajo del programa es el siguiente:

Carga y Preprocesamiento de Datos: Se carga el dataset MNIST y se normalizan las imágenes dividiendo los valores de los píxeles por 255.0 para que los valores estén en el rango [0, 1]. Definición de la Capa Personalizada: Se define una capa personalizada que simula el comportamiento del memristor de HP. Esta capa realiza la actualización de los pesos sinápticos en función de la corriente que pasa a través de ellos. Construcción del Modelo: Se construye el modelo de la red neuronal utilizando la capa personalizada junto con capas adicionales de activación y salida. Compilación del Modelo: Se compila el modelo utilizando el optimizador Adam y la función de pérdida de entropía cruzada esparsa para la clasificación. Entrenamiento del Modelo: Se entrena el modelo durante varias épocas en el conjunto de entrenamiento de MNIST. Evaluación del Modelo: Se evalúa el modelo en el conjunto de prueba de MNIST para determinar su precisión. Predicción y Visualización: Se realizan predicciones en el conjunto de prueba y se visualizan algunas de las predicciones junto con sus etiquetas verdaderas. Algoritmo de Entrenamiento El algoritmo de entrenamiento utilizado es el gradiente descendente, implementado mediante el optimizador Adam. Este algoritmo ajusta los pesos sinápticos para minimizar la función de pérdida. Los pasos del algoritmo son los siguientes:

Inicialización de Pesos y Sesgos: Los pesos y sesgos de la red se inicializan aleatoriamente. Propagación hacia Adelante: Los datos de entrada se propagan a través de la red para generar predicciones. Cálculo de la Pérdida: Se calcula la función de pérdida que mide la discrepancia entre las predicciones y las etiquetas verdaderas. Retropropagación del Error: El error se propaga hacia atrás a través de la red para calcular los gradientes de la función de pérdida respecto a los pesos y sesgos. Actualización de Pesos y Sesgos: Los pesos y sesgos se actualizan utilizando los gradientes calculados y el algoritmo de optimización Adam.

Cuadro I: Parámetros $\frac{W}{L}$ del sistema

| **Neurona** | | **Sinapsis** | |
|---|---|---|---|
| ***Parámetro*** | ***Valor*** | ***Parámetro*** | ***Valor*** |
| $W_1$ | $0{,}2\mu m$ | $W_1$ | $5\mu m$ |
| $W_2$ | $0{,}2\mu m$ | $W_2$ | $5\mu m$ |
| $W_3$ | $1\mu m$ | $W_3$ | $1{,}5\mu m$ |
| $W_4$ | $0{,}2\mu m$ | $W_4$ | $15\mu m$ |
| $W_5$ | $0{,}2\mu m$ | $W_5$ | $1{,}5\mu m$ |
| $W_6$ | $0{,}2\mu m$ | $W_6$ | $15\mu m$ |
| $W_7$ | $0{,}2\mu m$ | | |
| $W_8$ | $0{,}2\mu m$ | | |
| $W_9$ | $1\mu m$ | | |

*$L_{min} = 0{,}1\mu m$ en ambos circuitos.

## CONCLUSIÓN

Este trabajo propone una novedosa integración de memristores en redes neuronales artificiales para la clasificación del dataset MNIST. Al utilizar un modelo de memristor de HP, demostramos cómo estos dispositivos pueden mejorar la eficiencia energética y la capacidad de almacenamiento en sistemas de aprendizaje automático. Los resultados obtenidos muestran la viabilidad y el potencial de los memristores en aplicaciones de IA, abriendo el camino para futuros desarrollos en este campo emergente.

## REFERENCIAS