Trabajo práctico 1
Redes Neuronales y Aprendizaje Profundo

Ignacio Ezequiel Cavicchioli
Padrón 109428
icavicchioli@fi.uba.ar

10/9/2025

Índice

1. Introducción

Este documento presenta el desarrollo de las consignas del trabajo práctico N°2 de la materia de **Redes Neuronales y Aprendizaje Profundo**. El código correspondiente fue realizado en *Jupyter notebooks*, *Python*, adjuntados a la entrega en formato PDF. Toda imagen o implementación requeridas para el análisis se explicitarán en el presente archivo, por lo que la lectura del código en sí queda a discreción del lector. La teoría relevante será presentada y discutida en la sección pertinente.

2. Ejercicio 1

2.1. Consignas

Implemente un perceptrón simple que aprenda la función lógica AND y la función lógica OR, de 2 y de 4 entradas. Muestre la evolución del error durante el entrenamiento. Para el caso de 2 dimensiones, grafique la recta discriminadora y todos los vectores de entrada de la red.

2.2. Desarrollo

Lo primero que se hizo fue generar el dataset para entrenar los perceptrones, que no es más que la tabla de verdad de la función lógica que se quiere aprender.

Las figuras 1, 2,4,5 muestran el error en función de la iteración (de entrenamiento) para los perceptrones que emulan las funciones AND y OR de 2 y 4 entradas. Como se discutió en la teórica, el perceptrón es capaz de aprender estas funciones lógicas simples, por lo que el error final del entrenamiento es cero.

La figura 6 muestra la frontera de decisión del perceptrón que aprendió la función OR, y la 3 muestra la frontera para la compuerta AND. Los puntos del mismo color son de la misma clase (0 o 1), y son adecuadamente segregados por la frontera. Contrario a SVM, el perceptrón simple no tiene la distancia desde la frontera hasta las muestras en su función de costo, por lo que la recta que separa las clases no es única.

Para encontrar la fórmula de la recta que hace de frontera de decisión se puede partir de la expresión de la sumatoria de los X con sus respectivos pesos y el bias, y suponer que $Y = 0$, lo que significa que estás parado sobre la frontera, tu muestra no es ni de una clase ni de la otra.

$$X1 \cdot w1 + X2 \cdot w2 + b = 0$$

Y si suponemos que $X1$ es nuestra abscisa que vamos a barrer y $X2$ es la ordenada, la recta toma la forma de:

$$X2 = \frac{w1 \cdot X1 + b}{-w2}$$

Y entonces $m = -w1/w2$ y b $= -b/w2$

2.3. Análisis

Los perceptrones lograron aprender sus funciones lógicas designadas sin error en una cantidad finita de iteraciones del algoritmo.

El ejercicio demuestra la capacidad del perceptrón de aprender, además de familiarizarnos con un algoritmo de gradiente descendiente, por el cual los pesos se actualizan en la dirección que minimiza el error.

Todo esto es fundamental para las consignas siguientes.

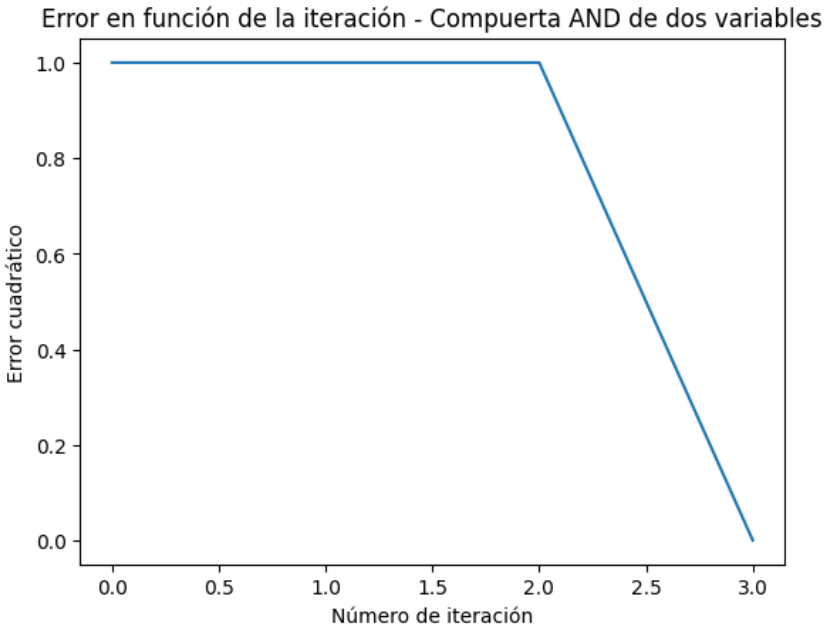

Figura 1: Error de entrenamiento en el tiempo para compuerta AND de 2 entradas

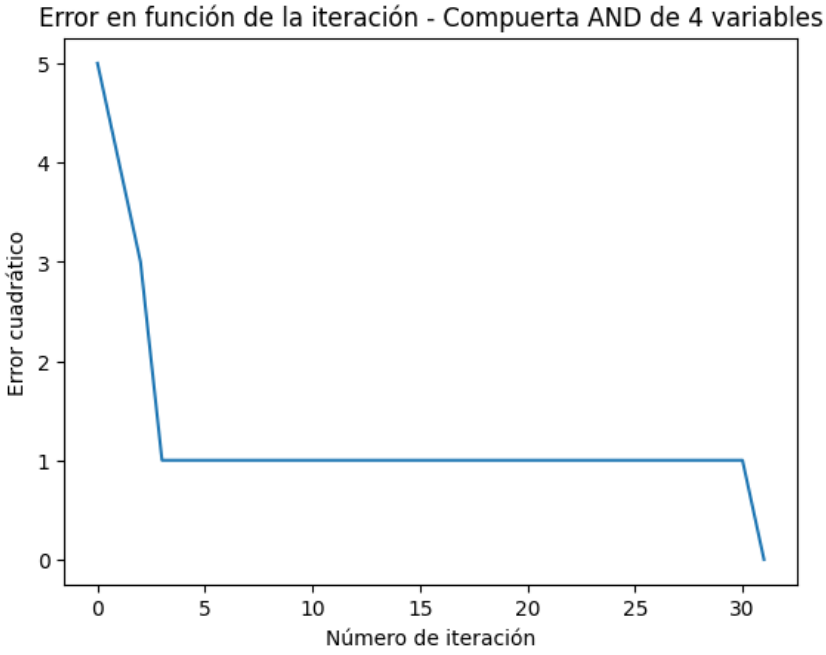

Figura 2: Error de entrenamiento en el tiempo para compuerta AND de 4 entradas

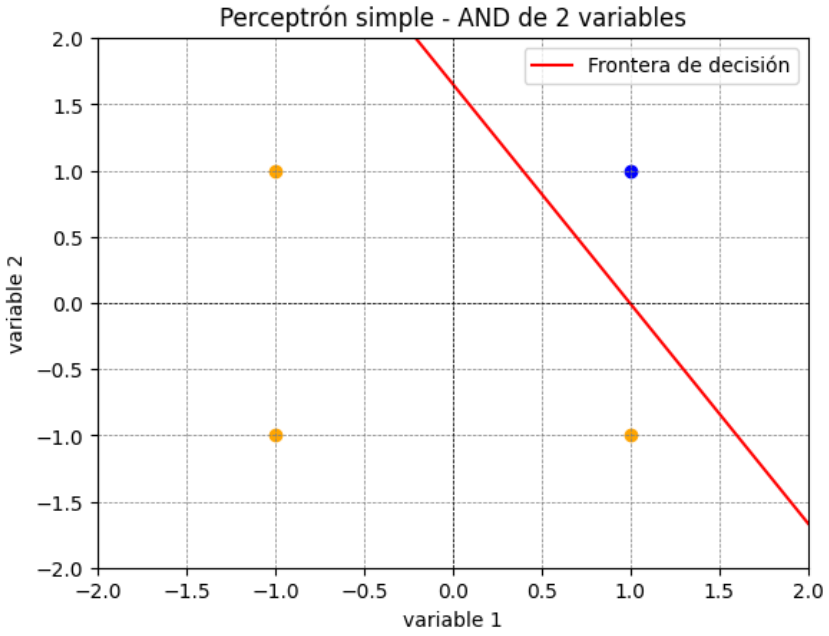

Figura 3: Frontera encontrada para compuerta AND de 2 entradas

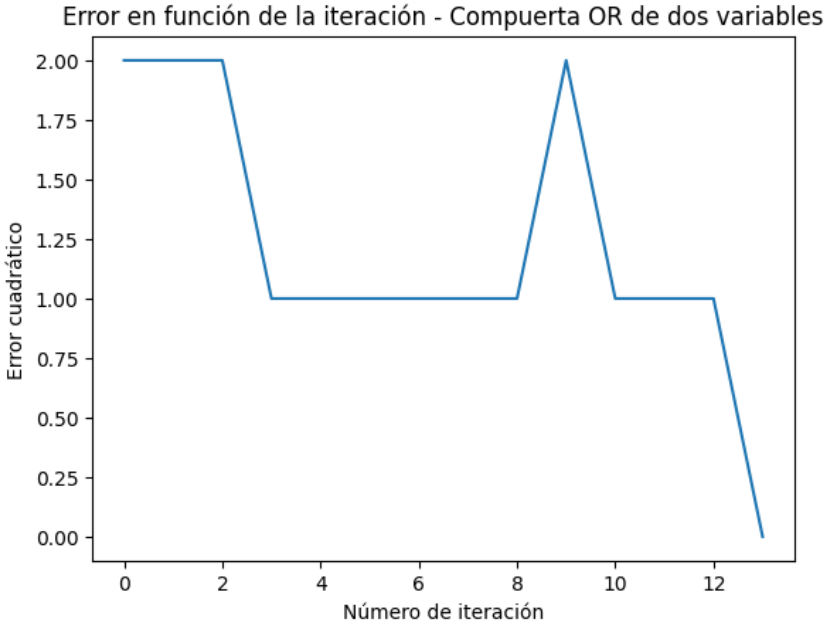

Figura 4: Error de entrenamiento en el tiempo para compuerta OR de 2 entradas

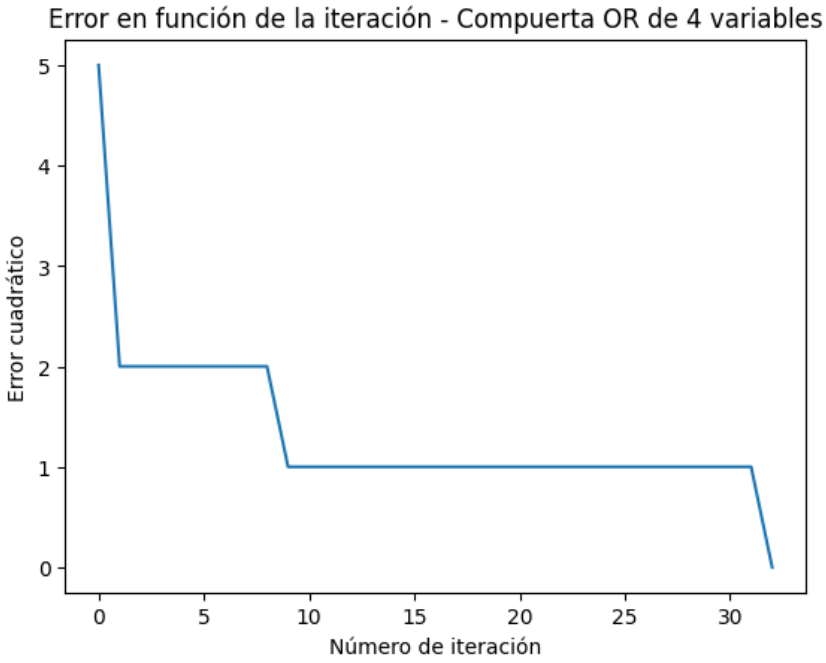

Figura 5: Error de entrenamiento en el tiempo para compuerta OR de 4 entradas

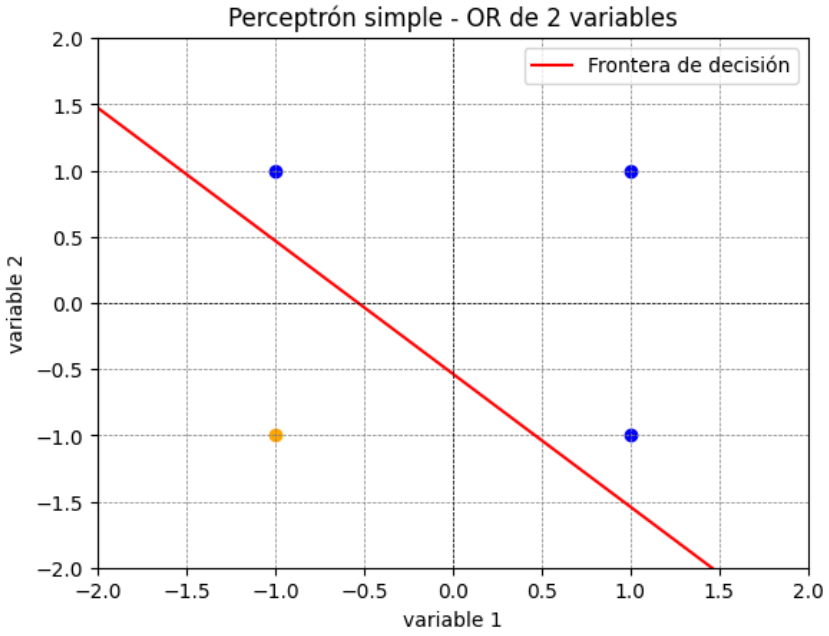

Figura 6: Frontera encontrada para compuerta OR de 2 entradas

3. Ejercicio 2

3.1. Consignas

Determine numéricamente cómo varía la capacidad del perceptrón simple en función del número de patrones enseñados.

3.2. Desarrollo

La capacidad de un perceptron está definida en la p.111 del libro "Introduction to the theory of neural computation". El experimento se debe hacer para para una cantidad particular de unidades de entrada, que sería la dimensión de nuestro espacio de features. A medida que este número N aumenta ($N \to$ ínf), se demuestra que la capacidad de almacenamiento máxima del perceptron converge a $p_{max} = 2N$, con p la cantidad de patrones.

La idea es tomar un perceptron de cierto tamaño y alimentarlo con cada vez más patrones, registrando los resultados. En el ejercicio, la capacidad se ensayó de la siguiente forma:

los resultados para $N = 30$ se pueden ver en la figura 7.

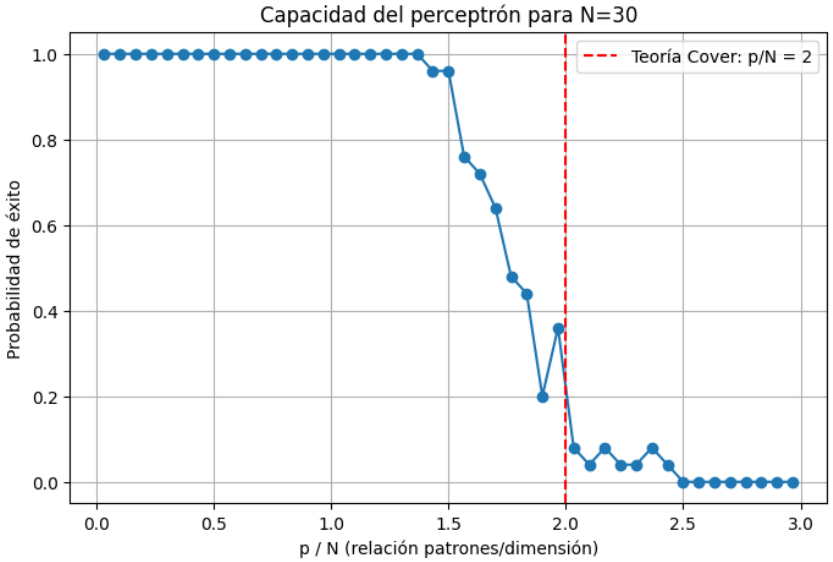

Figura 7: Ensayo de capacidad $N = 30$

3.3. Análisis

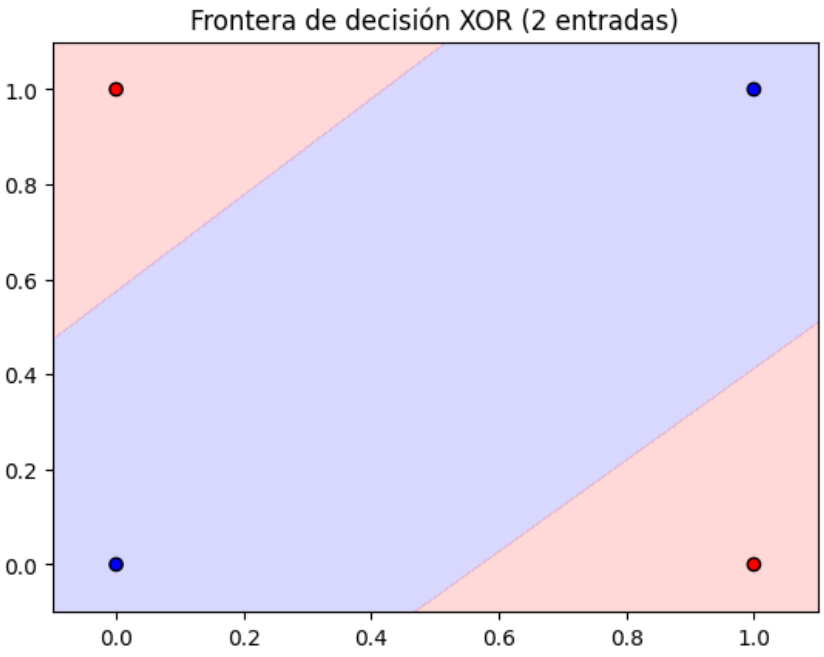

Figura 8

4. Ejercicio 3

4.1. Consignas

Implemente un perceptrón multicapa que aprenda la función lógica XOR de 2 y de 4 entradas (utilizando el algoritmo Backpropagation y actualizando en batch). Muestre cómo evoluciona el error durante el entrenamiento.

4.2. Desarrollo

4.3. Análisis

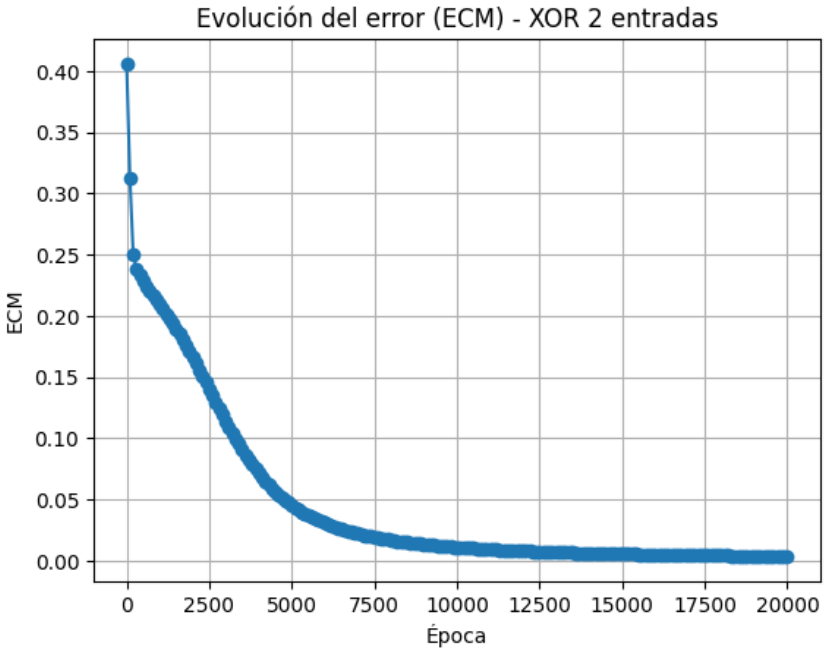

Figura 9

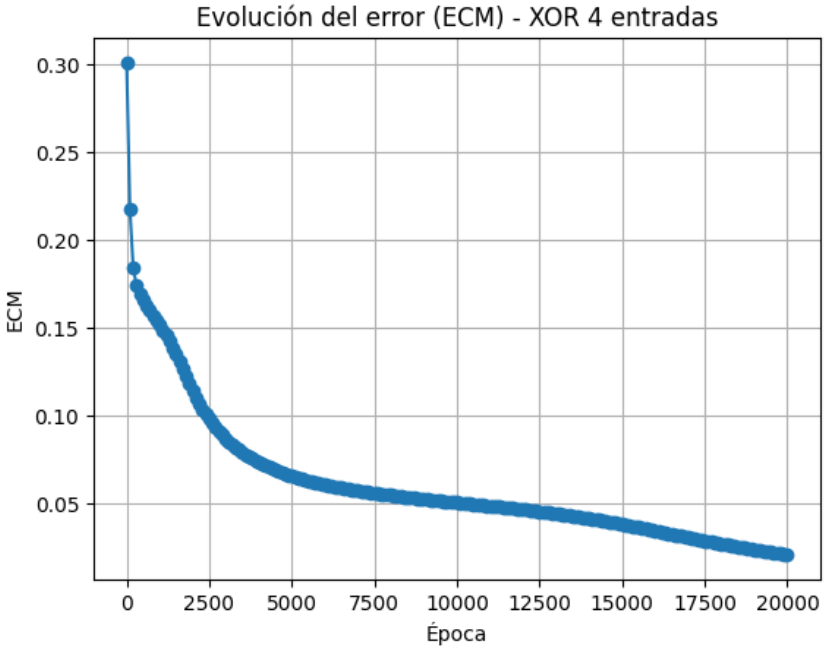

Figura 10

5. Ejercicio 4

5.1. Consignas

a - Implemente una red con aprendizaje Backpropagation que aprenda la siguiente función:

$$f(x, y, z) = \sin(x) + \cos(y) + z$$

donde: x e $y \in [0, 2\pi]$ y $z \in [-1, 1]$.

Para ello construya un conjunto de datos de entrenamiento y un conjunto de evaluación. Muestre la evolución del error de entrenamiento y de evaluación en función de las épocas de entrenamiento.

b - Estudie la evolución de los errores durante el entrenamiento de una red con una capa oculta de 30 neuronas cuando el conjunto de entrenamiento contiene 40 muestras. ¿Que ocurre si el minibatch tiene tamaño 40? ¿Y si tiene tamaño 1?

5.2. Desarrollo

5.3. Análisis

6. Ejercicio 5

6.1. Consignas

Siguiendo el trabajo de Hinton y Salakhutdinov (2006), entrene una máquina restringida de Boltzmann con imágenes de la base de datos MNIST. Muestre el error de reconstrucción durante el entrenamiento, y ejemplos de cada uno de los dígitos reconstruidos.

6.2. Desarrollo

6.3. Análisis

7. Ejercicio 6

7.1. Consignas

Entrene una red convolucional para clasificar las imágenes de la base de datos MNIST. ¿Cuál es la red convolucional más pequeña que puede conseguir con una exactitud de al menos 90 % en el conjunto de evaluación? ¿Cuál es el perceptrón multicapa más pequeño que puede conseguir con la misma exactitud?

7.2. Desarrollo

7.3. Análisis

8. Ejercicio 7

8.1. Consignas

Entrene un autoencoder para obtener una representación de baja dimensionalidad de las imágenes de MNIST. Use dichas representaciones para entrenar un perceptrón multicapa como clasificador. ¿Cuál es el tiempo de entrenamiento y la exactitud del clasificador obtenido cuando parte de la representación del autoencoder, en comparación con lo obtenido usando las imágenes originales?

8.2. Desarrollo

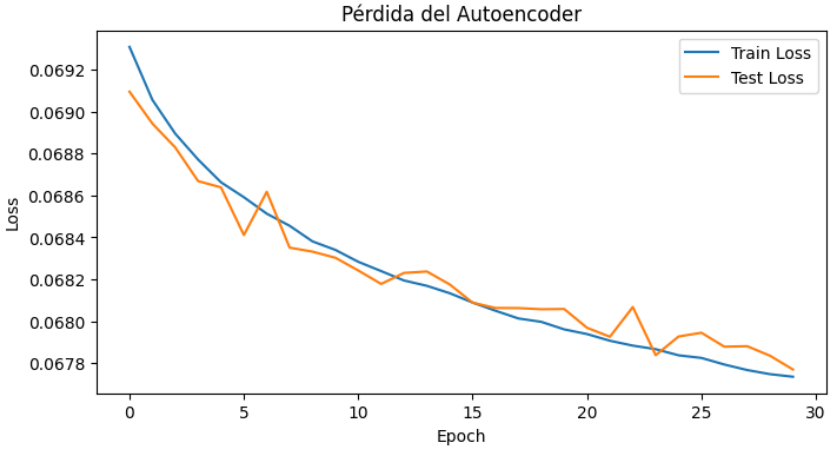

Figura 11

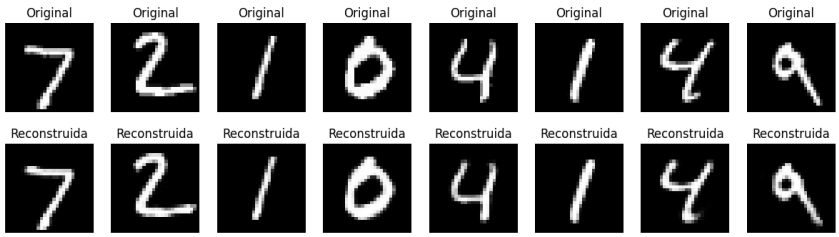

Figura 12

entrenamiento final: Epoch 1/30 — Loss=0.6624 — Acc=81.28Epoch 2/30 — Loss=0.3537 — Acc=89.86Epoch 3/30 — Loss=0.3200 — Acc=90.68Epoch 4/30 — Loss=0.3073 — Acc=90.92Epoch 5/30 — Loss=0.2996 — Acc=91.19Epoch 6/30 — Loss=0.2962 — Acc=91.22Epoch 7/30 — Loss=0.2921 — Acc=91.29Epoch 8/30 — Loss=0.2884 — Acc=91.44Epoch 9/30 — Loss=0.2884 — Acc=91.51Epoch 10/30 — Loss=0.2855 — Acc=91.53Epoch 11/30 — Loss=0.2832 — Acc=91.62Epoch 12/30 — Loss=0.2837 — Acc=91.56Epoch 13/30 — Loss=0.2823 — Acc=91.63Epoch 14/30 — Loss=0.2813 — Acc=91.79Epoch 15/30 — Loss=0.2810 — Acc=91.69Epoch 16/30 — Loss=0.2799 — Acc=91.83Epoch 17/30 — Loss=0.2785 — Acc=91.79Epoch 18/30 — Loss=0.2773 — Acc=91.82Epoch 19/30 — Loss=0.2772 — Acc=91.85Epoch 20/30 — Loss=0.2775 — Acc=91.76Epoch 21/30 — Loss=0.2751 — Acc=91.91Epoch 22/30 — Loss=0.2738 — Acc=91.90Epoch 23/30 — Loss=0.2745 — Acc=91.89Epoch 24/30 — Loss=0.2738 — Acc=91.94Epoch 25/30 — Loss=0.2735 — Acc=91.97... Epoch 28/30 — Loss=0.2736 — Acc=91.98Epoch 29/30 — Loss=0.2726 — Acc=91.98Epoch 30/30 — Loss=0.2720 — Acc=92.05Test Accuracy: 92.33

9. Conclusiones