



**UNIVERSIDAD TECNOLÓGICA CENTROAMERICANA**

**CLASE:**

Redes Neuronales y aprendizaje profundo

**SECCIÓN:**

38

**DOCENTE:**

Ing. Ivan De Jesus Deras

**PRESENTADO POR:**

Ana Melissa Cabrera Izaguirre 22141111

**SAN PEDRO SULA, CORTÉS, HONDURAS**

Jueves, 27 de marzo de 2025

## Red Neuronal para la Clasificación de MNIST

### Metodología

En este proyecto se implementó una red neuronal completamente conectada para la clasificación de dígitos manuscritos en el conjunto de datos MNIST. La red neuronal diseñada cuenta con una capa de entrada de 784 neuronas, correspondientes a los píxeles de cada imagen aplanada, una capa oculta con 128 neuronas que emplea la activación ReLU y una capa de salida con 10 neuronas que utiliza la función Softmax para generar probabilidades asociadas a cada clase del 0 al 9.

Para entrenar la red, se aplicó un preprocesamiento a los datos que incluyó la normalización de los valores de los píxeles al rango  $[0,1]$  y la conversión de las etiquetas a formato one-hot. Se optó por el uso de **mini-batch gradient descent**, estableciendo un tamaño de batch de 64 muestras, lo que permite una actualización más estable de los pesos y un entrenamiento eficiente.

El modelo fue optimizado utilizando, **Adam**, que combina momentum con RMSprop para ajustar de manera adaptativa la tasa de aprendizaje. Adicionalmente, se incorporó la **regularización L2** en la función de pérdida. El entrenamiento se llevó a cabo durante **10, 20, 50 y 100 épocas**, calculando en cada iteración la pérdida y precisión del modelo para evaluar su desempeño.

### Resultados

Durante el entrenamiento, se observaron mejoras progresivas en la precisión del modelo. Se generaron gráficos de pérdida y precisión para visualizar la evolución del aprendizaje y evaluar el comportamiento del entrenamiento. Con la configuración óptima de hiperparámetros y el uso de **Adam** junto con **regularización L2**, la red alcanzó una precisión final de aproximadamente **92.89%** en el conjunto de entrenamiento después de 50 épocas.

Sin embargo, en la fase de predicción se detectó un problema importante: la red estaba asignando la misma predicción para todas las imágenes de prueba, lo que indicaba un error en la propagación hacia atrás o en la actualización de los pesos. Esto llevó a una revisión detallada de la implementación del cálculo de gradientes y del optimizador para asegurar que los parámetros de la red estuvieran ajustándose correctamente.

## Pruebas cambiando tasa de aprendizaje sin utilizar optimizadores y regulador

tasa de aprendizaje = 0.1 Tasa de aprendizaje = 0.001

```

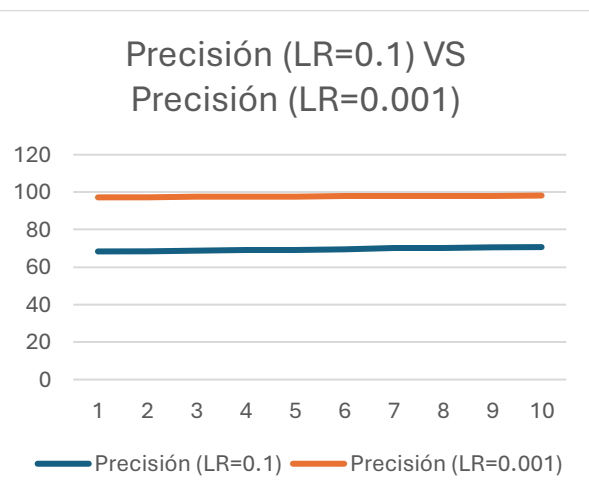
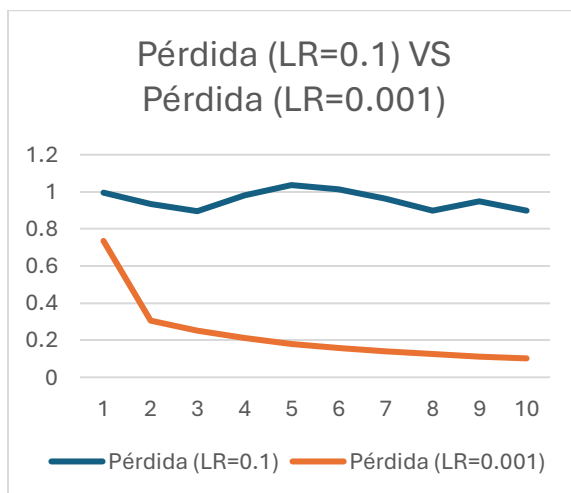
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
● melizaguirre@melizaguirre:~/ProyectoRedesNeuronales_Ana_
Época 1/20, Pérdida: 0.995647134280611
Época 2/20, Pérdida: 0.9338102903368957
Época 3/20, Pérdida: 0.8955148650268278
Época 4/20, Pérdida: 0.9810253337490071
Época 5/20, Pérdida: 1.0362215801984314
Época 6/20, Pérdida: 1.014529241829515
Época 7/20, Pérdida: 0.9645203108777949
Época 8/20, Pérdida: 0.9000618381306764
Época 9/20, Pérdida: 0.9502770654773242
Época 10/20, Pérdida: 0.8968430404096792
Época 11/20, Pérdida: 0.9791273050831255
Época 12/20, Pérdida: 0.9741239599656684
Época 13/20, Pérdida: 1.0686717221003292
Época 14/20, Pérdida: 1.015026020560911
Época 15/20, Pérdida: 1.008829512911926
Época 16/20, Pérdida: 0.9996813454825848
Época 17/20, Pérdida: 0.919120243227855
Época 18/20, Pérdida: 0.9598357401684603
Época 19/20, Pérdida: 0.9720313260886835
Época 20/20, Pérdida: 1.0228656543975656
Precisión: 68.38499999999999%
Entrenamiento finalizado.

```

```

13  entrada_dim = 784
14  oculta_dim = 128
15  salida_dim = 10
16  tasa_aprendizaje = 0.001
17  num_epochs = 10
18
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
● melizaguirre@melizaguirre:~/ProyectoRedesNeuronales_Ana_Cabrera/Proye
Época 1/10, Pérdida: 0.7352590494404772
Época 2/10, Pérdida: 0.3069791669077044
Época 3/10, Pérdida: 0.2511938588842997
Época 4/10, Pérdida: 0.21045175522576703
Época 5/10, Pérdida: 0.17994840861605768
Época 6/10, Pérdida: 0.15666789261738706
Época 7/10, Pérdida: 0.13847959244714725
Época 8/10, Pérdida: 0.12386019491789253
Época 9/10, Pérdida: 0.11192506038632696
Época 10/10, Pérdida: 0.10188009981224609
Precisión: 97.245%
Entrenamiento finalizado.

```



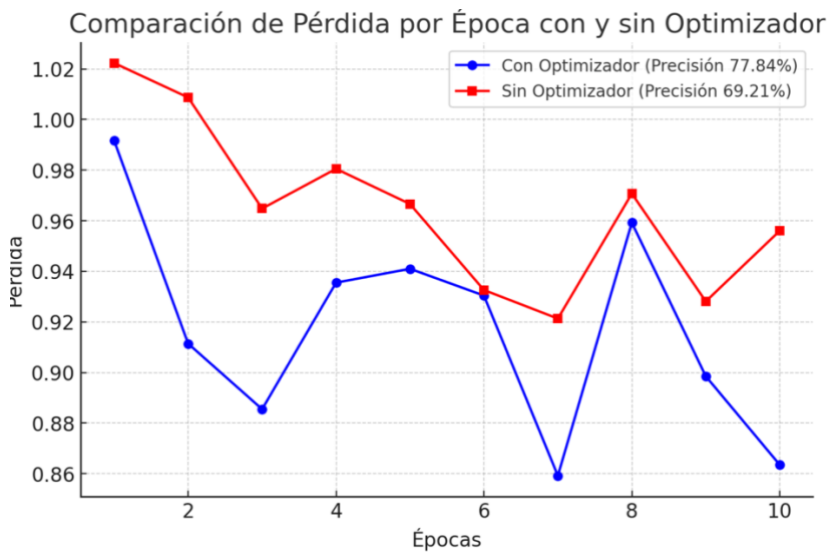
Los resultados muestran cómo la tasa de aprendizaje afecta el desempeño del modelo durante el entrenamiento. Con una tasa de 0.1, la pérdida fluctúa

significativamente entre épocas, lo que sugiere que el modelo tiene dificultades para converger de manera estable. Además, la precisión final alcanzada es de **68.38%**, lo que indica que la red no logró generalizar bien. Por otro lado, con una tasa de **0.001**, la pérdida disminuye de manera constante, indicando una convergencia más estable y controlada. La precisión final es **97.25%**, lo que demuestra que un ajuste más fino de los pesos permitió una mejor optimización.

## Pruebas utilizando optimizador ADAM y regulador L2

### Con y Sin optimizador

```
melizaguirre@melizaguirre:~/ProyectoRedesNeuronales_Ana_Cabrera/Proyecto$ python3 entren_optimizadores.py
Epoca 1/10, Pérdida: 0.9916014369817306
Epoca 2/10, Pérdida: 0.9114418627812315
Epoca 3/10, Pérdida: 0.8854956651777778
Epoca 4/10, Pérdida: 0.9355212029360082
Epoca 5/10, Pérdida: 0.9410787972003757
Epoca 6/10, Pérdida: 0.9304576790446667
Epoca 7/10, Pérdida: 0.8592430148083824
Epoca 8/10, Pérdida: 0.9592324465935121
Epoca 9/10, Pérdida: 0.8985618235209495
Epoca 10/10, Pérdida: 0.8635297349678904
Precisión: 77.84333333333333%
Entrenamiento finalizado.
melizaguirre@melizaguirre:~/ProyectoRedesNeuronales_Ana_Cabrera/Proyecto$ python3 entrenamiento.py
Epoca 1/10, Pérdida: 1.0223511439056034
Epoca 2/10, Pérdida: 1.008794073287437
Epoca 3/10, Pérdida: 0.9648602089833502
Epoca 4/10, Pérdida: 0.9805362505910922
Epoca 5/10, Pérdida: 0.9666838187416456
Epoca 6/10, Pérdida: 0.9326011597099839
Epoca 7/10, Pérdida: 0.9213535862223784
Epoca 8/10, Pérdida: 0.9707690596144949
Epoca 9/10, Pérdida: 0.9281362313655376
Epoca 10/10, Pérdida: 0.956024065894127
Precisión: 69.20833333333333%
Entrenamiento finalizado.
```



El uso del optimizador en el entrenamiento de la red neuronal ha demostrado ser beneficioso, ya que logró una menor pérdida y una mayor precisión (77.84% frente

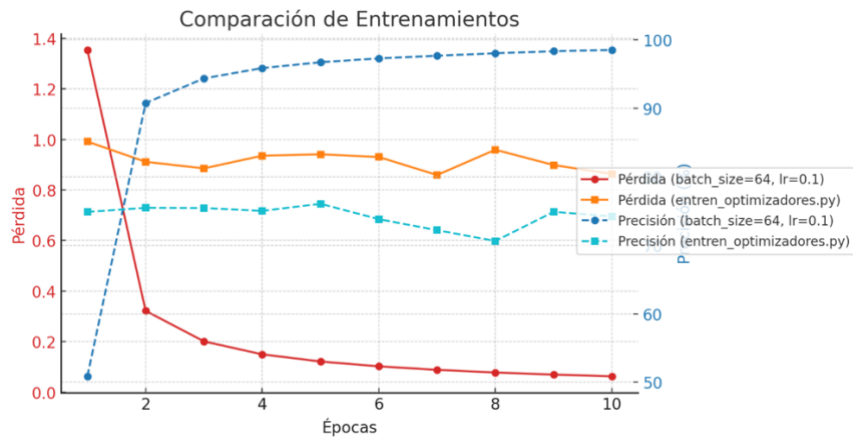
a 69.21%) en comparación con el entrenamiento sin optimizador. Además, la curva de pérdida con optimizador muestra una mayor estabilidad, mientras que sin él se observan fluctuaciones más notables.

## Al probar con mas epocas

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	PORTS
Epoca 1/40, Pérdida: 0.9845, Precisión: 75.41%				
Epoca 2/40, Pérdida: 0.9627, Precisión: 75.71%				
Epoca 3/40, Pérdida: 0.9467, Precisión: 74.98%				
Epoca 4/40, Pérdida: 0.9061, Precisión: 76.90%				
Epoca 5/40, Pérdida: 0.9680, Precisión: 71.14%				
Epoca 6/40, Pérdida: 0.8956, Precisión: 72.99%				
Epoca 7/40, Pérdida: 0.9156, Precisión: 72.43%				
Epoca 8/40, Pérdida: 0.8922, Precisión: 71.61%				
Epoca 9/40, Pérdida: 0.9587, Precisión: 69.58%				
Epoca 10/40, Pérdida: 0.9307, Precisión: 70.84%				
Epoca 11/40, Pérdida: 0.8968, Precisión: 71.07%				
Epoca 12/40, Pérdida: 0.9432, Precisión: 69.59%				
Epoca 13/40, Pérdida: 0.9593, Precisión: 68.36%				
Epoca 14/40, Pérdida: 0.9540, Precisión: 68.11%				
Epoca 15/40, Pérdida: 0.9309, Precisión: 68.81%				
Epoca 16/40, Pérdida: 0.9525, Precisión: 68.77%				
Epoca 17/40, Pérdida: 0.9074, Precisión: 69.71%				
Epoca 18/40, Pérdida: 0.9183, Precisión: 70.16%				
Epoca 19/40, Pérdida: 0.9489, Precisión: 69.43%				
Epoca 20/40, Pérdida: 0.9078, Precisión: 71.55%				
Epoca 21/40, Pérdida: 0.8974, Precisión: 71.17%				
Epoca 22/40, Pérdida: 0.9344, Precisión: 68.70%				
Epoca 23/40, Pérdida: 0.8755, Precisión: 71.92%				
Epoca 24/40, Pérdida: 0.8924, Precisión: 69.95%				
Epoca 25/40, Pérdida: 0.9116, Precisión: 70.00%				
Epoca 26/40, Pérdida: 0.9091, Precisión: 69.51%				
Epoca 27/40, Pérdida: 0.9278, Precisión: 68.45%				
Epoca 28/40, Pérdida: 0.8884, Precisión: 70.49%				
Epoca 29/40, Pérdida: 0.9158, Precisión: 69.27%				
Epoca 30/40, Pérdida: 0.8391, Precisión: 71.45%				
Epoca 31/40, Pérdida: 0.8660, Precisión: 70.94%				
Epoca 32/40, Pérdida: 0.8598, Precisión: 69.70%				
Epoca 33/40, Pérdida: 0.8639, Precisión: 69.16%				
Epoca 34/40, Pérdida: 0.8993, Precisión: 67.68%				
Epoca 35/40, Pérdida: 0.8835, Precisión: 68.46%				
Epoca 36/40, Pérdida: 0.9632, Precisión: 65.64%				
Epoca 37/40, Pérdida: 0.9332, Precisión: 66.32%				
Epoca 38/40, Pérdida: 0.9779, Precisión: 64.44%				
Epoca 39/40, Pérdida: 0.9693, Precisión: 64.56%				

Al aumentar el número de épocas a 40, se observa una fluctuación en la precisión a lo largo del entrenamiento, con una ligera tendencia a mejorar en las últimas iteraciones. Inicialmente, la precisión se mantiene estable en torno al 75%, pero luego experimenta altibajos, alcanzando un pico del 78.87%

Al implementar un **batch\_size** de 64, aumentar la tasa de aprendizaje a 0.1 e implementar una **segunda capa** oculta se observó una mejora significativa en la convergencia del modelo. En solo 10 épocas, la precisión pasó del 50.87% en la primera iteración al 98.52% en la última, con una pérdida decreciendo de 1.3542 a 0.0625. Esto indica que el modelo aprendió mucho más rápido en comparación con los entrenamientos anteriores.



## Comparación del Entrenamiento con Diferentes Dimensiones en la Segunda Capa Oculta

### Análisis de Resultados

- La configuración con 96 neuronas en la segunda capa oculta obtuvo una precisión promedio de 92.52%, superior a la de 64 neuronas (91.84%).
- Esto sugiere que una mayor cantidad de neuronas permitió a la red aprender mejor las representaciones de los datos.
- Con 96 neuronas, la red empezó con una mejor precisión en la primera época (56.34%) en comparación con 64 neuronas (50.87%).
- La pérdida final también fue ligeramente menor en la configuración de 96 neuronas (0.0607 vs. 0.0625), lo que indica una mejor optimización de los pesos.

