UNIVERSIDAD DEL VALLE DE GUATEMALA
CC3084 – Data Science
Sección 20
Luis Furlán

Laboratorio 2

Redes Neuronales con MNIST

Andre Jo, 22199
Héctor Penedo, 22217

GUATEMALA, 31 de julio del 2025

Link del repositorio

https://github.com/DANdelion-0908/Lab2-Data_Science

1. Modificación del Ancho de la Red (8 puntos)

Con un tamaño de capa escondida de 50 neuronas, el modelo es capaz de alcanzar una precisión del 97% con un tiempo de entrenamiento de 1 segundo por cada una de las épocas para un total de 5 segundos con 5 épocas. Al realizar el cambio a 200 neuronas, la precisión escaló a un 98% y aumentó el tiempo de entrenamiento a 6 segundos con 2 segundos para la primera época y 1 segundo para todas las demás.

Se realizaron pruebas con distintos tamaños de capa escondida y se obtuvieron los siguientes resultados:

Tamaño de capa	Precisión	Tiempo de entrenamiento
50	96.60%	5 segundos (1s por época).
100	97.55%	5 segundos (1s por época).
200	97.65%	7 segundos (2s, 2s, 1s, 1s, 1s).
300	97.88%	10 segundos (2s por época).
500	97.64%	15 segundos (3s por época).

Se determinó que el modelo con un tamaño de capa de 300 neuronas ofrece el mejor rendimiento al tener la precisión más alta tomando un tiempo de entrenamiento razonable, a diferencia de tamaños superiores, que toman una mayor cantidad de tiempo y obtienen una precisión menor.

2. Modificación de la Profundidad de la Red (12 puntos)

Se agregó una capa oculta adicional con un tamaño de 50 neuronas añadiendo la variable tamanio_capa_escondida_2 y agregándola al atributo Dense de tensorflow con una función de activación RelU en el modelo. Utilizando ambas capas ocultas, el modelo tuvo un tiempo de entrenamiento de 11 segundos (1 segundo más lento que el mejor modelo) y obtuvo una precisión del 97.55% (un 0.33% peor que el mejor modelo). Se probaron otros tamaños de capa, y 300 + 50 obtuvo el mejor rendimiento.

3. Redes Profundas (12 puntos)

Se realizaron arquitecturas con distinta cantidad de capas ocultas y se configuraron con distintos tamaños. Se obtuvieron los siguientes resultados:

Capa 1	Capa 2	Capa 3	Capa 4	Capa 5	Precisión	Tiempo
300	0	0	0	0	97.88%	10 s
300	200	100	50	25	97.40%	16 s
500	300	200	100	50	96.95%	22 s
512	256	128	64	32	97.61%	22 s
520	300	250	130	60	97.95%	23 s

Luego de probar distintos valores de capas, se determinó que la configuración 520 + 300 + 250 + 130 + 60 es capaz de mejorar la precisión del modelo de una sola capa de 300 neuronas, aunque toma más del doble de tiempo. También se encontró que existe

una relación de proporcionalidad directa entre la profundidad de las capas con el tiempo de ejecución.

4. Funciones de Activación I (8 puntos)

Se aplicó la función de activación ‘sigmoid’ a todas las capas utilizando la configuración con el mejor rendimiento hasta el momento. Esta ejecución mostró una precisión del 95.78% y tuvo un tiempo de entrenamiento de 20 segundos. Siendo ligeramente peor que la ejecución del modelo con la función de activación 'relus' en cada capa.

5. Funciones de Activación II (8 puntos)

Se aplicaron las funciones de activación 'relu' y 'tanh' a la primera y segunda capa respectivamente, mientras que se dejó la función 'sigmoid’ al resto de capas. Comparado a la ejecución anterior, el modelo logró una precisión del 97.48% y tuvo un tiempo de entrenamiento de 19 s. Teniendo un rendimiento considerablemente superior a la ejecución del modelo exclusivamente con la aplicación de la función de activación 'sigmoid’.

Función	Ventajas	Desventajas
ReLu (Rectified Linear Unit)	Actualmente es la función de activación más utilizada para capas convolusionales y deep learning. Resulta ser muy simple computacionalmente. Presenta un comportamiento lineal, lo que facilita su optimización.	No es diferenciable en el punto cero. Todos los valores negativos se vuelven cero.
Sigmoid	Su output es 0 o 1, lo que resulta especialmente útil en problemas de clasificación binaria. Funciona bien en redes con pocas capas.	La gradiente se vuelve pequeña en valores positivos y negativos, lo que ralentiza el aprendizaje.
Tanh	Su output está centrado en 0, mapea inputs positivos como fuertemente positivos e inputs negativos como fuertemente negativos.	Sufre de desvanecimiento de gradiente cuando el input presenta una alta magnitud.

6. Tamaño de Batch Grande (5 puntos)

Se modificó el tamaño del batch a 10,000 desde 100 y se mantuvo la configuración de las capas ocultas del ejercicio anterior. El algoritmo logró una precisión del 64.22% y tuvo un tiempo de entrenamiento de 8 segundos. Esta ejecución mostró un rendimiento

drásticamente inferior a la ejecución anterior con una disminución de precisión de 31.56%.

El modelo utiliza el tamaño de batch (tanda) para crear tandas más pequeñas con la información del conjunto de datos. Dichas tandas se procesan independientemente antes de que se actualicen los parámetros del modelo. El tamaño de la tanda indica la cantidad de datos que almacenará cada tanda.

Por otro lado, si cambiamos el tamaño del batch a 1 SDG observamos que el tiempo de ejecución ante las configuraciones anteriores como el batch de 10,000 tardaron más que esa debido a que el batch size de 1 tiene que hacer una por cada mientras que el otro realiza una por 10,000 es eficiente en tiempo pero el problema es la presición del modelo puede ser mala al obtener un porcentaje de 62.39%

Tamaño del Batch	Perdida	Precisión	Tiempo de ejecución
1 SDG			
10000	38.59	79.51%	7.8

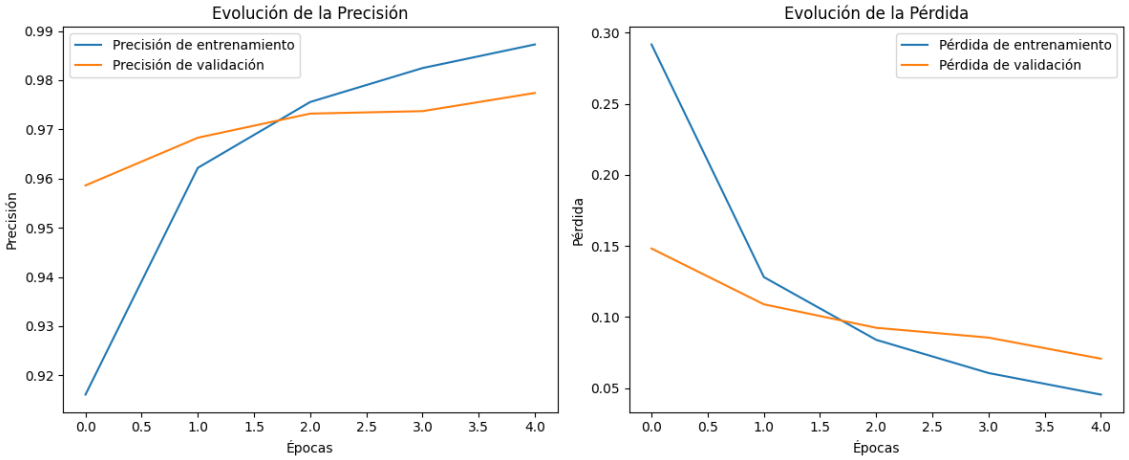
Entonces inferimos que los resultados van a ser coherentes con la teoria porque el entrenamiento fue más lento y menos estable, pero con potencial para mejor generalización debido al ruido estocástico que permite escapar de mínimos locales. En contraste, con batch de 10,000, el modelo converge más rápidamente y de forma más estable, pero con riesgo de caer en mínimos locales sin exploración y sobre ajustarse al conjunto de entrenamiento.

Al reducir la tasa de aprendizaje a 0.0001, se observó un entrenamiento mucho más lento. La pérdida disminuyó gradualmente y el modelo tardó más en comenzar a converger en comparación con configuraciones con tasas más altas. Esta lentitud es esperada, ya que debía de analizar cada paso, pero esos pasos eran pequeños lo cual mayores datos que hay más se tardaba. En términos de precisión final, en algunos casos el modelo logró una precisión al 91.50% con una pérdida de 37.74.

Se ajustó la tasa de aprendizaje a 0.02, lo cual representa un valor alto. Se observó que el modelo inicialmente convergía más rápido en las primeras épocas, reduciendo la pérdida rápidamente. También se observa que el porcentaje de accuracy sube un poco.

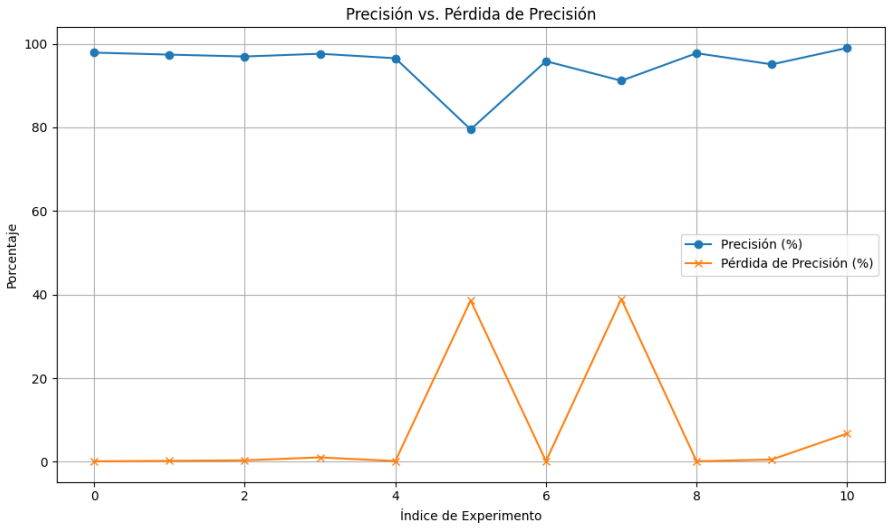
Se implementaron técnicas de regularización avanzadas para mejorar la generalización del modelo. Se incluyó Dropout entre capas densas con una tasa del 30% y se aplicó regularización L2 con una penalización de 0.001. Aunque la pérdida en el conjunto de entrenamiento fue mayor (debido a la penalización de los pesos y a la aleatoriedad de Dropout), la precisión en el conjunto de validación no mejoró significativamente. Esto

puede deberse al exceso de regularización (muchas capas con Dropout y activaciones que dificultan la convergencia como la sigmoide.



Learning Rate	Pérdida	Precisión	Tiempo de ejecución
0.02	0.11	97.74	3 m 43 s
0.001	38.93	91.16	16 m 35.4 s

Como se puede observar es mejor utilizar un learning rate moderador para reducir el tiempo de ejecución ya que igualmente aprender dato por dato en este caso no es óptimo como se observa en la diferencia de pérdida y precisión de los dos modelos. Pero observamos en la gráfica que existe sobre ajuste en el modelo de learning rate 0.02 sobreajuste.



Como se puede observar, La gráfica compara la precisión y la pérdida de precisión de un modelo a lo largo de once experimentos distintos. Se observa que el modelo logra un rendimiento consistentemente alto en la mayoría de las configuraciones, con una

precisión superior al 95% y una pérdida casi nula. Sin embargo, los experimentos 5 y 7 se destacan como fracasos, donde la precisión se desploma drásticamente y la pérdida de precisión se dispara. Este comportamiento errático subraya la importancia de la selección de hiperparámetros o la configuración del modelo, ya que un mal ajuste puede llevar a un rendimiento muy inferior, mientras que una buena configuración permite alcanzar un alto grado de eficacia. A continuación, se realizará el siguiente modelo en la cual se observará una precisión de validación mayor a 98.5% para comprar el sobreajuste de los datos y su eficiencia.

Se diseñó un modelo de red neuronal convolucional optimizado con el objetivo de alcanzar una precisión de validación superior al 98.5% lo cual podría indicar como sobreajuste; tambien debemos de minimizar el tiempo de entrenamiento y mantener una arquitectura eficiente. Se eligieron tres capas convolucionales con activación ReLU para extraer características como bordes y texturas, seguidas de capas de max pooling para reducir dimensionalidad y mejorar la generalización. La inclusión de BatchNormalization ayuda a estabilizar y acelerar el entrenamiento, mientras que la regularización mediante Dropout y penalización L2 se utilizó estratégicamente para mitigar el sobreajuste sin perder capacidad predictiva. Tras el aplanamiento, se incorporaron cinco capas densas con tamaños decrecientes para reducir gradualmente la complejidad del modelo. Se utilizó un tamaño de tanda de 64, ideal para balancear eficiencia computacional y convergencia. Finalmente, se eligió el optimizador RMSprop debido a su capacidad para adaptarse dinámicamente al ruido presente en los datos de imágenes, como las del conjunto MNIST. Esta configuración logró una precisión de prueba del 99.00% en solo 2 minutos y 6.3 segundos, con una pérdida de 6.76, cumpliendo con los objetivos planteados.

Bibliografía

Krishnamurthy, B. (2024, 26 febrero). *An Introduction to the ReLU Activation Function*. Built In. https://builtin.com/machine-learning/relu-activation-function

GeeksforGeeks. (2025, 23 julio). *Tanh vs. Sigmoid vs. ReLU*. GeeksforGeeks. https://www.geeksforgeeks.org/deep-learning/tanh-vs-sigmoid-vs-relu/

GeeksforGeeks. (2025a, julio 23). *Batch Size in Neural Network*. GeeksforGeeks.