Prácticas: MNIST

Ana Valentina López Chacón

Redes Neuronales Artificiales
MIARFID, UPV

Enero, 2025

1. Descripción de la Práctica

Se buscó realizar diferentes experimentos sobre el *dataset* de MNIST [1], para la tarea de traducción de clasificación de imágenes. Se exploraron diferentes arquitecturas, funciones de activación, técnicas de aumento de datos y métricas de evaluación que permitieron medir la calidad de los resultados. Con el fin de progresivamente generar un *pipeline* desarrollado por completo en `Pytorch` con experimentos distribuidos a lo largo de diferentes `Jupyter Notebooks`.

1.1. Objetivos

Se plantearon los siguientes objetivos:

- Explorar diferentes arquitecturas de red sin emplear bloques convolucionales.
- Realizar una exploración de parámetros pertinente para valores de tasas de aprendizaje.
- Generar diversos tipos de aumento de datos mediante transformaciones.
- Obtener porcentajes de precisión o *accuracy* satisfactorios destacando como *benchmarks* los umbrales de 98.8 %, 99 %, 99.2 % y 99.4 %.

2. Análisis Descriptivo de los Datos

El dataset MNIST contiene imágenes de dígitos escritos a mano en escala de grises, con tamaño 28×28 píxeles (ver Fig. 1). Se realizó un análisis inicial de los datos para visualizar la distribución de las muestras en cada una de las clases para las particiones del *dataset* de train (60000 muestras) y test (10000 muestras) (ver Fig. 2). Dado que el objetivo original de este conjunto de datos era evaluar modelos mas no ajustar parámetros, no se cuenta con la partición de validation.

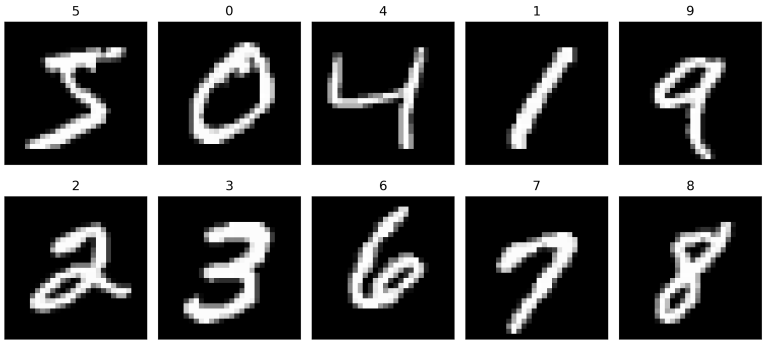

Figura 1: Gráfica de una Imagen por Categoría.

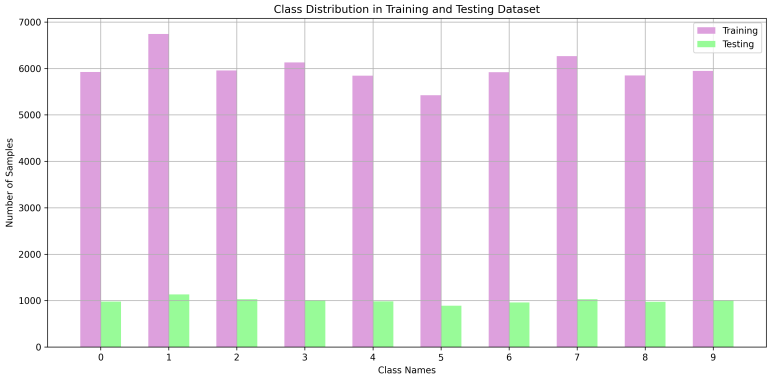

Figura 2: Distribución de Muestras por Categoría.

3. Resultados

Todos los experimentos se desarrollaron con el módulo `Pytorch` de `Python` y se organizaron de tal forma que se evidencie un progreso en las métricas de evaluación. Se estableció como *baseline* de los experimentos un *Multilayer Perceptron* (MLP) sencillo con tres capas ocultas de 1024 neuronas, función de activación `ReLU` y una capa de salida de clasificación. Como función de pérdida se tomó entropía cruzada, que combina una activación tipo *softmax* con un *negative log-likelihood loss*, y algoritmo de optimización *Stochastic Gradient Descent* (SGD) con una tasa de aprendizaje de 0.01, *momentum* de 0.9 y una erosión de pesos del 0.000001 que sirve como factor de regularización.

Para el proceso de entrenamiento se carga el modelo a GPU y se itera a lo largo de 45 épocas con un tamaño de *Batch* de 100, para cada una de estas hay una fase de entrenamiento, donde la red actualiza sus pesos usando SGD, y una fase de prueba, donde el modelo entrenado se evalúa utilizando data que no fue parte de su entrenamiento. El mejor modelo se guarda cuando se obtiene el mayor valor de precisión en test que para este caso inicial fue **98.39 %**.

Con el fin obtener un mejor resultado de clasificación se realizaron experimentos a partir del MLP base, en primer lugar, se incluyeron capas de normalización o *Batch Normalization* (BN) para estabilizar y acelerar el entrenamiento entre cada capa lineal y su respectiva función de activación, obteniendo una nueva precisión del 98.66 %. Luego, se incorporó al entrenamiento un método de enfriamiento de tasa de aprendizaje o *Learning Rate Annealing* (LRA), donde al llegar a las épocas 20 y 35 se disminuye de forma escalonada el valor del LR, lo que permite un aprendizaje rápido inicial y que conforme avanza el entrenamiento se afinen resultados, logrando una precisión del **98.65 %**.

Posteriormente, se exploraron técnicas de aumento de datos o *Data Augmentation* (DA) con el objetivo de reducir sobreentrenamiento y generar un modelo más robusto. En este caso, tomando en cuenta transformaciones que no comprometieran la interpretación de los datos, se aplicó una rotación aleatoria de 3° y 2° al igual que procesos de traslación y cambios de escala, los datos de prueba se mantienen iguales y se obtuvo una precisión del **98.86 %**. Luego, se optó por realizar una mayor alteración de los datos modificando los argumentos de las transformaciones aplicadas, obteniendo mejores resultados de precisión. Estas modificaciones de los datos se pueden apreciar en la siguiente imagen (ver Fig. 3) cuyo resultado fue del **99.26 %**. Para este punto en la experimentación se habían logrado muy buenos resultados, superando tres de los cuatro *benchmarks* planteados inicialmente.

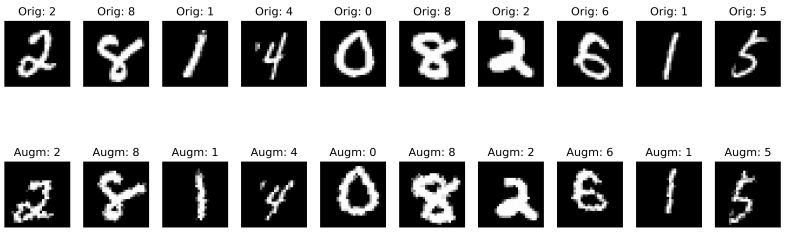

Figura 3: Visualización del Aumento de Datos.

Se observó que al incluir el último aumento de datos la mayor precisión al evaluar el modelo se reportó en las últimas épocas, lo cual nos llevó a incrementar el tiempo de entrenamiento y ajustar el cambio de la tasa de aprendizaje de forma acorde. Sin embargo, permitir una tasa de aprendizaje tan pequeña no estaba dando resultados satisfactorios para el experimento con 60 épocas donde su mejor precisión se obtiene en la época 47 con un porcentaje de **99.34 %** y permanece oscilando hasta terminar su entrenamiento. Es debido a esta observación que para el último experimento se optó por inicializar la tasa de aprendizaje con un valor mayor a los casos anteriores con una distribución y número de épocas más equitativo logrando el mejor valor de precisión global con un **99.42 %**. El progreso de cada experimento se registró en la siguiente tabla 1.

3.1. Resumen de Resultados

	Red	Hiperparams	Épocas	Transformaciones	Precisión
MLP	(1024 `ReLU`) × 3	lr = 0.01 BS = 100	45		98.39 %
+ BN	(1024 `BatchNorm1d` `ReLU`) × 3	lr = 0.01 BS = 100	45		98.66 %
+ LRA	(1024 `BatchNorm1d` `ReLU`) × 3	lr = 0.01, 0.001, 0.0001 BS = 100	20, 35, 45		98.65 %
+ DA	(1024 `BatchNorm1d` `ReLU`) × 3	lr = 0.01, 0.001, 0.0001 BS = 100	20, 35, 45	`RandomRotation(3)` `RandomAffine(2, (0.002,0.001), (0.001, 1.64) )`	98.86 %
+ DA	(1024 `BatchNorm1d` `ReLU`) × 3	lr = 0.01, 0.001, 0.0001 BS = 100	20, 35, 45	`RandomRotation(10)` `RandomAffine(20, (0.1, 0.1), (0.9, 1.1) )`	99.3 %
+ DA	(1024 `BatchNorm1d` `ReLU`) × 3	lr = 0.01, 0.001, 0.0001 BS = 100	25, 45, 60	`RandomRotation(10)` `RandomAffine(20, (0.1, 0.1), (0.9, 1.1) )`	99.34 %
+ DA	(1024 `BatchNorm1d` `ReLU`) × 3	lr = 0.1, 0.01, 0.001 BS = 100	25, 50, 75	`RandomRotation(10)` `RandomAffine(20, (0.1, 0.1), (0.9, 1.1) )`	**99.42 %**

Cuadro 1: Resumen de Resultados para cada Experimento.

4. Conclusiones

Para finalizar este informe podemos destacar las siguientes conclusiones:

- La tarea de clasificación para MNIST no demostró tener una complejidad muy alta, debido al tamaño y diversidad de los datos obteniendo resultados muy buenos con técnicas relativamente sencillas, dado que no se incluyeron capas convolucionales ni bloques residuales, que habría sido lo más natural para un problema con imágenes.
- El mayor cambio en la precisión reportada se dio al aumentar la magnitud del aumento de datos, generando así muestras mucho más variadas y complejas de predecir, lo que pone al modelo en una situación adversa mejorando así su capacidad de generalización y su adaptabilidad a muestras que nunca ha visto en su entrenamiento.
- Al inicializar la tasa de aprendizaje con mayor magnitud en el experimento final se le permitió al modelo explotar su capacidad de aprendizaje en iteraciones iniciales y afinar sus resultados posteriormente, sin llegar a sobreajustarse.

Referencias

[1] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). *Gradient-based learning applied to document recognition.* Proceedings of the IEEE, 86(11), 2278-2324.