RNA

Anna Llinares Llinares

February 6, 2025

1 MNIST

En el conjunto de datos MNIST, primero llevamos a cabo experimentaciones basadas en el boletín de la práctica 1, ajustando variables como el learning rate (0.1, 0.01, 0.001, 0.0001), el número de neuronas en las capas ocultas (1024, 512, 256) y el tipo de optimizador (SGD, Adam, AdamW, Adadelta). Además, se experimentó con la sustitución de la activación ReLU por GeLU. De estas configuraciones, las pruebas realizadas con 1024 neuronas mostraron ser las más efectivas. Con todo esto se alcanzó un 98.43, en concreto se obtuvieron los siguientes resultados con ReLU:

	SGD	Adadelta	Adam	AdamW
0.1	98.32	98.18	11.35	11.35
0.01	98.16	93.90	96.39	96.66
0.001	94.91	71.96	98.36	98.33
0.0001	74.65	19.18	98.18	98.24

Table 1: Resultados de las configuraciones de modelos con MNIST y ReLU, en base a la práctica 1

Y para GeLU:

	SGD	Adadelta	Adam	AdamW
0.1	98.38	97.45	11.35	41.57
0.01	97.68	91.76	97.43	97.29
0.001	92.36	50.09	98.36	98.29
0.0001	35.05	15.27	98.06	98.01

Table 2: Resultados de las configuraciones de modelos con MNIST y GeLU, en base a la práctica 1

A partir de la segunda práctica, simplemente ejecutando el notebook del boletín de prácticas, logramos un accuracy del 99.01% (0) [1]. Posteriormente, decidimos cambiar al optimizador Adam, ya que había mostrado buenos resultados en experimentaciones anteriores, obteniendo una mejora hasta 99.13% (1). Para continuar decidimos añadir la transformación Color Jitter, lo que nos permitió alcanzar un 99.17% (2). Dado el buen rendimiento del modelo usando GeLU en lugar de ReLU, probamos esta activación junto con el optimizador Adam e incrementando *RandomRotation* a 15, lo que nos llevó a conseguir un 99.23% (3), superando así el tercer hito.

Después, añadimos la transformación *RandomResizedCrop*, logrando un incremento del 99.31% (4). Siguiendo esta tendencia, añadimos *RandomPerspective*, aunque el rendimiento se mantuvo en 99.30% (5).

Al observar que no conseguíamos superar el cuarto hito, llevamos a cabo varias experimentaciones ajustando los parámetros de las transformaciones de *RandomAffine*, *RandomResizedCrop*, *RandomPerspective* y *Color Jitter*. Sin embargo, el rendimiento no mejoraba, lo que nos llevó a la conclusión de que la configuración era excesivamente compleja y, en algunos casos, redundante. Tras optimizar los parámetros, añadir una capa de *Dropout* y utilizar *ReduceLROnPlateau*, logramos un accuracy del 99.39

Finalmente, incrementamos el número de epochs de 75 a 100, lo que nos permitió alcanzar un 99.52% (7), logrando así superar el objetivo final.

[1] (X) corresponde al número del Jupyter Notebook en el que se llevó a cabo este experimento, dentro de la carpeta con toda la experientación llevada a cabo

Configuración	Activación	Data Augmentation	Accuracy
Modelo Inicial (SGD)	ReLU	-	99.01
+ Adam	ReLU	-	99.13
+ Color Jitter	ReLU	Color Jitter	99.17
+ GeLU + Randomrotation(15)	GeLU	Rotation	99.23
+ RandomResizedCrop	GeLU	Rotation + ResizedCrop	99.31
+ RandomResizedCrop	GeLU	Rotation + ResizedCrop + Perspective	99.30
+ DropOut + ReduceLROnPLateau	GeLU	Rotation + ResizedCrop + Perspective	99.39
+ 100 epochs	GeLU	Rotation + ResizedCrop + Perspective	99.52

Table 3: Resultados de diferentes configuraciones de modelos con MNIST, en base a la práctica 2

2 CIFAR-10

En el conjunto de datos CIFAR, inicialmente superamos el primer hito sin realizar cambios en el boletín de prácticas, logrando un error del 12.06% (0). Al implementar el parámetro Color Jitter en las transformaciones para el data augmentation, obtuvimos un error de 12.05% (1). Después, decidimos optar por una arquitectura con tres capas convolucionales sin Color Jitter, lo que resultó en una mejora significativa, alcanzando un error del 8.31% (2) y superando así el segundo y tercer hito. Observando la notable mejora con el modelo de tres capas convolucionales, se decidió entrenar esta misma configuración añadiendo la transformación Color Jitter. Esto resultó en un error de 8.06% (3), acercándonos significativamente a nuestro objetivo. Posteriormente, experimentamos con cuatro capas convolucionales, el error volvió a empeorar en un 8.4% (4). También decidimos entrenar un modelo de cinco capas convolucionales, lo que consiguió disminuir el error a un 8.15% (5). Luego, intentamos añadir al modelo anterior una transformación *RandomVerticalFlip* pero el error empeoró significativamente a un 12.36% (6), por lo que no dudamos en descartar esta transformación. Con base a los resultados obtenidos previamente con el conjunto de datos MNIST, donde la sustitución de ReLU por GELU mejoró el rendimiento, aplicamos GELU en nuestro modelo de tres capas convolucionales con Color Jitter, logrando reducir el error a un 8.18% (7), aunque siguió por debajo de los resultados obtenidos con ReLU. Finalmente, cambiamos la capa de pooling de Max Pooling a Average Pooling, utilizando un modelo de cinco capas convolucionales GELU y Color Jitter, lo que nos permitió alcanzar un error de 7.27% (8) y lograr nuestro objetivo. Después de observar que el mejor modelo antes de alcanzar el hito 4, fue uno con tres capas ReLU, decidimos probar cambiando la capa de pooling de Max Pooling a Average Pooling para ver si lograba mejorar el resultado anterior. Sin embargo, el cambio empeoró el rendimiento, resultando en un error del 8.23% (9).

Configuración	Convoluciones	Activación	Data Augmentation	Pooling	Error
Modelo Inicial	1	ReLU	-	MaxPooling	12.06
+ Color Jitter	1	ReLU	Color Jitter	MaxPooling	10.25
3 CNN	3	ReLU	Color Jitter	MaxPooling	8.31
3 CNN + CJ	3	ReLU	-	MaxPooling	8.27
4 CNN	4	ReLU	Color Jitter	MaxPooling	8.31
5 CNN	5	ReLU	Color Jitter	MaxPooling	8.18
+ Vertical Flip	5	ReLU	Color Jitter + Vertical Flip	MaxPooling	12.36
3 CNN GELU	3	GELU	Color Jitter	MaxPooling	7.27
+ AveragePooling	3	ReLU	Color Jitter	AveragePooling	8.23

Table 4: Resultados de diferentes configuraciones de modelos con CIFAR-10