Javier Gil-Terrón Rodríguez
MIARFID 2021-2022

Redes neuronales artificiales

1. MNIST – Perceptrón multicapa

Para la realización de la práctica, tanto en esta tarea como la siguiente basada en redes neuronales convolucionales, se ha seguido el esquema propuesto en el github de la asignatura y usando PyTorch.

De esta manera, se ha implementado de manera progresiva añadiendo elementos hasta alcanzar el resultado final propuesto. Así pues, la primera configuración, teniendo la estructura básica de la red, se compone por 3 capas densas (o de operación lineal con la sintaxis de PyTorch) con funciones de activación ReLU y softmax para la salida (definido de serie al establecer criterio CrossEntropyLoss) y optimizador SGD (descenso por gradiente estocástico) con factor de aprendizaje 0.1. Además, se ha empleado un tamaño de batch de 100 y se ha limitado a 75 epochs (estas cifras serán constantes en todas las pruebas realizadas).

Esta primera configuración ha obtenido una precisión máxima de 98.35%, a la cual, la primera adición que se hizo fue la de normalización de batch (BN) en cada capa, incrementando la precisión hasta 98.67%.

La siguiente prueba fue la de añadir ruido gaussiano (GN) tanto en la entrada como en las capas, y se alcanzó una precisión de 98.8%. Tras esto, se incluyó learning rate scheduler (LRS) modificando el factor de aprendizaje de manera fija según la epoch en que se encuentre, y aumentando la precisión máxima hasta 98.85%.

Finalmente, se incluyó data augmentation (DA) con traslación del 10% tanto en ancho como en altura, siendo esta la opción que más impacto ha tenido sobre la precisión final, llegando a alcanzar el 99.3% propuesto, aunque es cierto que se alcanzó únicamente en un par de epochs y no de manera consistente.

Con esto en mente, se reemplazó el learning rate scheduler por la opción ReduceLROnPlateau con paciencia de 5 epochs y factor multiplicativo 0.5, además de añadir rotación de 3 grados en la fase de data augmentation. Con estos cambios, se logró el 99.3% de manera consistente.

También se probaron algunas otras modificaciones menores pero que no mejoraron el resultado, como probar otros algoritmos de

optimización como Adam, pero no se reflejarán en la tabla de resultados.
Todo lo anteriormente citado se puede sintetizar en la siguiente tabla de resultados donde se refleja la precisión obtenida según la configuración. Estas configuraciones son acumulativas, es decir, cada una será la anterior añadiendo un elemento (a excepción de la última, que será una modificación de su inmediatamente anterior según lo explicado anteriormente).

Básico	**+ BN**	**+ GN**	**+ LRS**	**+ DA**	**Final**
98.35%	98.67%	98.8%	98.85%	99.2%	99.3%

Tabla 1 – Valores de Accuracy para la tarea MNIST en función de la configuración

2. CIFAR10 – Convolucionales

En esta tarea no se ha seguido la misma lógica que en la anterior, (partiendo de una estructura base e ir modificándola hasta alcanzar el objetivo) sino que se han probado estructuras de red diferentes hasta encontrar una que pareciera prometedora y realizar pruebas sobre ella.

Para este caso, la estructura base de la red estaba compuesta por cinco bloques de capas convolucionales con función de activación ReLu y MaxPool; pero en este caso ya partimos en esta primera aproximación con batch normalization (al igual que todas las pruebas de esta sección). El algoritmo de optimización también será SGD con factor de aprendizaje 0.1 y se limitará a 75 epochs con 100 de tamaño de batch.

Con esta primera configuración se consiguió un accuracy de 81.68%, a la que se añadió learning factor scheduler (fijo en función del número de epoch) y data augmentation con 20% de traslación en cada eje, con lo que se aumentó la precisión a 86.94%.

La estructura anterior se modificó para tratar de alcanzar mejores resultados, con lo que se reemplazó el learning factor scheduler ReduceLROnPlateau con paciencia de 5 epochs y factor multiplicativo 0.5 y al data augmentation se le añadieron rotación de 20 grados, además de normalizarlo. Con esto se logró un accuracy de 89.17%.

Para llegar al 92% propuesto en la tarea se probó a cambiar totalmente la topología de la red, probando a continuación una red residual

(Resnet18), en este caso, formada por 8 bloques de 2 covoluciones con batch normalization y un average pooling tras los 8 bloques. Tanto el learning factor scheduler como el data augmentation se mantuvieron idénticos al anterior. En conjunto, se llegó a una precisión de 92.12%.

No obstante, ya que esta red de partida ofrecía tan buenos resultados para lo que se pedía en la práctica, decidí probar otras estructuras de red más sencillas e incluir otros mecanismos para llegar al 92%. Bajo esta premisa se implementó una VGG11 compuesta por dos convoluciones iniciales seguidas de un maxpool cada una, y tres bloques de dos convoluciones y un maxpool para finalizar con tres capas lineales con Dropout entre ellas (las funciones de activación siguen siendo ReLu y se mantiene batch normalization). Con esta estructura se logró un 89.34% de accuracy tras 75 epochs, que ya mejora el 89.17% obtenido con la estructura básica, pues se sigue empleando la misma configuración de Data augmentation y Learning rate scheduler.

Puesto que este resultado parecía aún insuficiente, se modificó esta red añadiendo más capas hasta implementar una VGG16 compuesta por dos bloques de dos convoluciones y un maxpool y tres bloques de tres convoluciones y un maxpool finalizando una vez más con tres capas lineales con Dropout entre ellas (mantenemos ReLus, batch normalization y mismo Data augmentation y Learning factor scheduler). Con esta configuración se llegó a 91.51% de accuracy tras 75 epochs, que hubiera alcanzado el 92% objetivo con más epochs.

A esta estructura que parece más prometedora se le incluyó la técnica de cutout como Data augmentation, recortando aleatoriamente un fragmento de ocho píxeles en las imágenes, con lo que se alcanzó un 92.59% de accuracy.

Ya que esta técnica supuso una mejora consistente respecto a la anterior se probó a sustituirla por mixup (mezclar imágenes aleatoriamente) para observar el impacto que tendría sobre el accuracy, mejorando incluso los resultados obtenidos con cutout y llegando a 92.81%.

Además, se probó a juntar ambas técnicas, tanto cutout como mixup, pero los resultados fueron peores que los obtenidos usando estas técnicas por separado, logrando un 92.32%. No obstante, existe otra manera de integrar estos mecanismos: en lugar de utilizarlos a la vez, se puede recortar un fragmento de una imagen e insertarlo en otra, lo que recibe el nombre de CutMix, con lo que se mejoró los resultados anteriores alcanzando el máximo hasta ahora: 92.88% tras 75 epochs.

*Cabe citar que en todas las pruebas realizadas se probó a incluir ruido gaussiano, pero en ninguno de los casos de esta práctica obtuvo mejores resultados.

Los resultados de esta tarea se resumen en las siguientes tablas:

Básico	**+ LR y DA**	**DA*normalizado**	**Resnet18**
82.68%	86.94%	89.17%	92.12%

Tabla 2 – Valores de Accuracy para la tarea CIFAR10 en función de la configuración

VGG11	**VGG16**	**+Cutout**	**+Mixup**	**+Mixup y cutout**	**+CutMix**
89.34%	91.54%	92.59%	92.81%	92.32%	92.88%

Tabla 3 – Valores de Accuracy para la tarea CIFAR10 en función de la configuración