Prácticas: CIFAR-10

Ana Valentina López Chacón

Redes Neuronales Artificiales
MIARFID, UPV

Enero, 2025

1. Descripción de la Práctica

Se realizaron diferentes experimentos sobre el *dataset* de CIFAR-10 [1], enfocados en la tarea de clasificación de imágenes. Se exploraron diversas arquitecturas, técnicas de aumento de datos y métricas de evaluación para medir la calidad de los resultados. El objetivo fue desarrollar un *pipeline* completamente en `Pytorch`, estructurado en distintos `Jupyter Notebooks` con experimentos progresivos.

1.1. Objetivos

Se plantearon los siguientes objetivos:

- Evaluar el impacto del cambio del *learning rate* e incluir aumento de datos para una CNN.
- Explorar diferentes arquitecturas de red, con capas convolucionales y bloques residuales.
- Alcanzar niveles de precisión satisfactorios en la tarea de clasificación, destacando umbrales de 86 %, 88 %, 90 % y 92 % como *benchmarks*.

2. Análisis Descriptivo de los Datos

El *dataset* CIFAR-10 contiene imágenes RGB de 32×32 píxeles organizadas en 10 categorías: avión, automóvil, pájaro, gato, venado, perro, rana, caballo, barco y camión (ver Fig. 1). Se realizó un análisis inicial para visualizar la distribución de las muestras en cada una de las clases para las particiones de train (50000 muestras) y test (10000 muestras) (ver Fig. 2). Este conjunto de datos presenta imágenes con una mayor variabilidad en cuanto a iluminación, ángulos y fondo, lo que hace que la tarea de clasificación sea más desafiante.

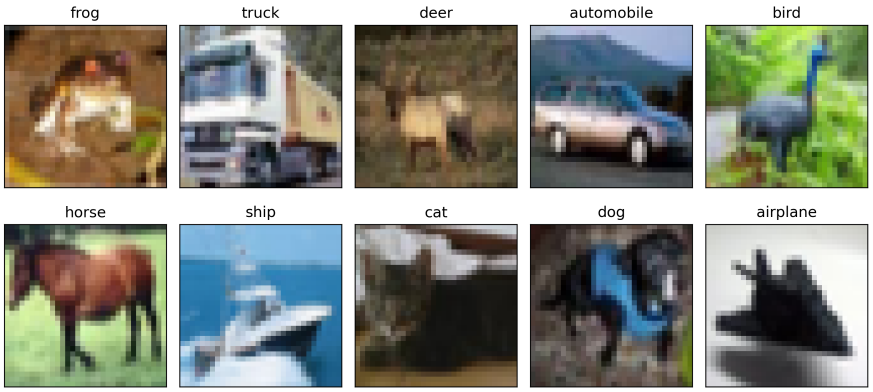

Figura 1: Ejemplo de Imágenes por Categoría.

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 una Red Convolucional (CNN) con cinco capas convolucionales con su respectiva normalización y *pooling* con activaciones `ReLU` y dos capas completamente conexas para generar la salida de la clasificación. Como función de pérdida se tomó entropía cruzada 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 100 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 **82.11 %** en la época 81.

Con el objetivo de mejorar la capacidad de generalización y reducir el sobreajuste del modelo, se realizaron experimentos a partir de la CNN base. En primer lugar, se implementaron técnicas de aumento de datos (*Data Augmentation*) incluyendo flip horizontal, rotaciones de hasta $\pm 5°$, desplazamientos horizontales y verticales de hasta el 20 % del tamaño de la imagen. También, se incorporó un ajuste adaptativo de la tasa de aprendizaje o *Learning Rate Annealing* (LRA) utilizando la estrategia *Reduce on Plateau*, con 10 épocas de paciencia y un factor de 0.1, alcanzó una precisión de **87.5 %** en la época 69.

Luego, se sustituyeron los últimos dos bloques convolucionales por bloques residuales y se incluyó en el aumento de datos un recorte aleatorio de relleno y un ajuste de color, se obtuvo como precisión un **88.86** % en la época 60 y al añadir una capa de *average pooling* de 4×4 después del último bloque residual se sube a **89.65** % para la época 74 de este nuevo experimento. Para los siguientes experimentos se emplearon unicamente bloques residuales, en el primer caso se aplicaron 3 bloques residuales luego una capa de *max pooling* de 4×4 seguido por dos bloques residuales más y una capa final de *average pooling* de 8×8 donde se registró una precisión del **91.74** % en la época 99, sin embargo con mucho sobreajuste dado que solo se presento una mejora del 0.04 en las últimas 50 épocas.

Por último, se optó por cambiar la capa *max pooling* de 4×4 por dos capas de 2×2 después del tercer y cuarto bloque residual respectivamente, adicionalmente se modificó el aumento de datos quitando por completo el ajuste de color e incluyendo en el recorte aleatorio también un cambio de dimensiones (ver Fig. 3). Estos cambios permitieron un entrenamiento mucho más estable y que alcanzara el último *benchmark* planteado en los objetivos inicialmente con una precisión del **92.64** % en la época 81.

Figura 3: Visualización del Aumento de Datos.

Un resumen de todos los resultados obtenidos se consignó en la siguiente tabla 1.

3.1. Resumen de Resultados

	Arquitectura con Cambios/Adiciones	Parámetros	Precisión
CNN	(`Conv2d`, `BatchNorm2d`, `MaxPool2d`) $\times$ 5 `ReLU`, `SGD(0.01, 1e-6, 0.9)`	1838346	82.11 %

+DA +LRA	`RandomHorizontalFlip()`
	`RandomAffine(5, (0.2, 0.2))`
	`ReduceLROnPlateau(o, 0.1, 10, 1e-6)`
		1838346	87.5 %

+ResNet	`ConvBlock` $\times$ 3, `ResBlock` $\times$ 2
	`RandomCrop(32, 4)`
	`ColorJitter(0.2, 0.2, 0.2, 0.1)`
		8886538	88.86 %

+ AvgPool	`ConvBlock` $\times$ 3, `ResBlock` $\times$ 2 `AvgPool2d`(4×4)	4954378	89.65 %
+ MaxPool	`ResBlock` $\times$ 3, `MaxPool2d`(4×4) `ResBlock` $\times$ 2, `AvgPool2d`(8×8)	5159146	91.74 %
ResNet Final	`ResBlock` $\times$ 3, `MaxPool2d`(2×2) `ResBlock`, `MaxPool2d`(2×2) `ResBlock`, `AvgPool2d`(8×8) `RandomResizedCrop(32, (0.8, 1.0))`	5159146	**92.64** %

Cuadro 1: Resumen de Resultados para cada Experimento.

4. Conclusiones

Para finalizar este informe podemos destacar las siguientes conclusiones:

- Se observó que el uso de métodos de regularización como aumento de datos, métodos de ajuste de la tasa de aprendizaje como *ReduceLROnPlateau* y técnicas de reducción de dimensionalidad como *pooling* generaron un salto significativo en los valores de precisión y redujeron considerablemente el número de parámetros pero manteniendo la información lo suficientemente relevante para generar un resultado satisfactorio mejorando la convergencia y estabilidad del entrenamiento.
- Ajustes en las técnicas de regularización como sustituir la capa de *max pooling* de 4×4 por dos capas de tamaño 2×2 permitió redistribuir la información de los datos de forma más adecuada al problema lo que le permitió al modelo extraer más características para realizar la inferencia final donde se alcanzó el valor de precisión máximo superando el último *benchmark* establecido.
- Se optó por cambiar de un arquitectura híbrida (CNN + ResNet) a una que utilizó exclusivamente bloques residuales, con el fin de explorar redes mucho más profundas, con mayor estabilidad de gradiente y que fuera más eficiente

a la hora de extraer y emplear características. Como resultado se obtuvo un modelo que generaliza mejor a datos que no ha visto en su entrenamiento.

Referencias

[1] Krizhevsky, A., & Hinton, G. (2009). *Learning multiple layers of features from tiny images.*