Informe del Proyecto: Clasificación de Imágenes CIFAR-10 con CNNs

Estudiante: [Jaime Jorge Rico Steegmann]
Fecha: [17/11/2025]
Asignatura: [IA]

1. Introducción

Este informe detalla el desarrollo y evaluación de modelos de Redes Neuronales Convolucionales (CNNs) para la clasificación de imágenes en el conjunto de datos CIFAR-10. Se exploran diversas arquitecturas y técnicas de regularización y optimización para obtener un modelo robusto y eficiente.

2. Metodología

2.1. Dataset: CIFAR-10

CIFAR-10 consiste en 60,000 imágenes a color de 32x32 píxeles en 10 clases (6,000 imágenes por clase). Se realizó una normalización de píxeles a [0,1] y un split estratificado 80/20 para entrenamiento y validación. Las etiquetas se convirtieron a formato one-hot.

2.2. Modelos Evaluados

1. **Baseline MLP:** Una red densa simple para establecer un punto de control sin sesgo inductivo espacial.
2. **CNN Simple (2 Bloques):** Introducción de capas convolucionales y de pooling para capturar características espaciales.
3. **CNN Profunda (3 Bloques):** Exploración de mayor capacidad del modelo.
4. **Técnicas de Regularización:**
    - **L2 Weight Decay:** Penalización de los pesos para controlar la complejidad.
    - **Dropout:** Desactivación aleatoria de neuronas para prevenir co-adaptación.
    - **Data Augmentation:** Generación de nuevas muestras de entrenamiento mediante transformaciones (flips, rotaciones, zooms).

- **Early Stopping:** Detención temprana del entrenamiento para evitar sobreajuste.

5. **Estrategias de Optimización:**
    - **Adam con ReduceLROnPlateau:** Adaptación del learning rate durante el entrenamiento.
    - **SGD con Momentum y CosineDecay:** Optimización alternativa con scheduler de LR predefinido.

2.3. Trazabilidad

Se utilizó Git para el control de versiones, y se registraron metadatos de los datos (data_meta.json), parámetros de los experimentos (params.yaml), historias de entrenamiento (history.csv), y métricas clave (metrics.json) para cada corrida. Todas las figuras se guardaron con identificadores de commit y timestamp.

3. Resultados y Análisis

3.1. Baseline MLP vs CNNs

Modelo	Parámetros	Tiempo/Época (s)	Val Acc (%)	Test Acc (%)
MLP	[X]	[Y]	[Z]	[W]
CNN (2 Bloques)	[X']	[Y']	[Z']	[W']
CNN (3 Bloques)	[X'']	[Y'']	[Z'']	[W'']
(Rellenar con valores de tus experimentos)				

La CNN (3 bloques) superó consistentemente al MLP, demostrando la eficacia de las capas convolucionales en la extracción de características espaciales. A pesar de tener un número similar o incluso menor de parámetros, las CNNs aprovechan el sesgo inductivo espacial.

3.2. Impacto de la Regularización

(Insertar figura con curvas de entrenamiento/validación de los modelos con y sin regularización. Comentar sobre la reducción del gap train/val y mejora en test acc.)

3.3. Ablación de Técnicas

Experimento	Descripción	Val Acc (%)	Test Acc (%)

A_Control	Todo activo	[A_val]	[A_test]
B_NoAugment	Sin Data Augmentation	[B_val]	[B_test]
C_NoL2	Sin Regularización L2	[C_val]	[C_test]
D_NoDropout *(Rellenar con valores de tus experimentos)*	Sin Dropout	[D_val]	[D_test]

El análisis de ablación mostró que la **Data Augmentation** tuvo el mayor impacto en la test_acc, indicando su importancia crítica para la generalización en este dataset. Le siguieron el Dropout y la regularización L2.

3.4. Matriz de Confusión y Errores Típicos

(Insertar figura de la matriz de confusión del mejor modelo. Comentar sobre las clases más confundidas, p. ej., 'cat' vs 'dog', 'automobile' vs 'truck'. Insertar también la figura de las 12 imágenes con errores.)

Las clases 'cat' y 'dog' son consistentemente confundidas, al igual que 'airplane' y 'bird' o 'automobile' y 'truck'. Esto sugiere que el modelo aún lucha con distinciones finas entre categorías visualmente similares.

4. Decisiones Justificadas

1. **Normalización a [0,1]**: Se aplicó para escalar los valores de píxel, estabilizando el entrenamiento y acelerando la convergencia al colocar las entradas en un rango numérico manejable para los optimizadores.
2. **Uso de CNN sobre MLP**: Se optó por CNNs desde el principio, ya que su arquitectura convolucional explota la estructura espacial de las imágenes (traslational invariance, detección de patrones locales), superando las limitaciones de los MLPs que aplanan la imagen perdiendo información posicional.
3. **Combinación de L2, Dropout y Data Augmentation**: Estas técnicas se aplicaron conjuntamente para mitigar el sobreajuste. La ablación confirmó su contribución individual, especialmente la de Data Augmentation, permitiendo al modelo aprender características más robustas y generalizables.
4. **Early Stopping con ReduceLROnPlateau**: Esta estrategia de callbacks permitió un entrenamiento eficiente, deteniendo el proceso cuando la validación dejaba de mejorar y ajustando el learning rate para facilitar la convergencia fina. Esto ahorra recursos computacionales y previene el sobreajuste.

5. **Profundidad de 3 Bloques CNN**: Después de probar con 2 bloques, se incrementó a 3 bloques (32->64->128 filtros) para aumentar la capacidad del modelo. Esto resultó en una mejora marginal en la precisión de prueba, justificando el ligero aumento en el coste computacional por el beneficio en rendimiento.

5. Límites y Próximos Pasos

5.1. Límites del Modelo Actual

- **Capacidad Limitada:** A pesar de tener 3 bloques, el modelo sigue siendo relativamente pequeño para la complejidad de CIFAR-10.
- **Sensibilidad a Clases Similares:** El modelo sigue confundiendo clases visualmente parecidas, lo que indica que aún no ha aprendido a extraer características discriminativas suficientes.
- **Aumentos Básicos:** Los aumentos aplicados son estándar; técnicas más avanzadas podrían mejorar el rendimiento.

5.2. Próximos Pasos y Mejoras Realistas

1. **Transfer Learning:** Implementar un modelo pre-entrenado en un dataset más grande (p. ej., ImageNet) como ResNet, VGG o EfficientNet, y ajustarlo (fine-tuning) en CIFAR-10. Esto puede proporcionar una base de características mucho más rica y un rendimiento superior.
2. **Técnicas Avanzadas de Data Augmentation:** Explorar métodos como **Mixup** o **Cutout** que generan muestras de entrenamiento más diversas y robustas, ayudando al modelo a enfocarse en características intrínsecas en lugar de texturas o fondos irrelevantes.
3. **Label Smoothing:** Para las clases confundidas, aplicar Label Smoothing puede regularizar las predicciones del modelo y hacerlo menos "seguro" de sus errores, lo que a menudo mejora la generalización.

6. Recuadro de Reproducibilidad (R10)

Parámetro / Recurso	Valor
Seed Fija	seed=42
Versión Python	[Versión Python usada, p.ej. 3.9.12]
Versión TensorFlow	[Versión TensorFlow usada, p.ej. 2.10.0]
GPU	[Nombre de GPU, p.ej. Tesla T4 (Colab)]

Commit Corto	[Último commit corto de la rama principal, p.ej. a1b2c3d]
Tag de Release	v1.0-P3-CIFAR10-[TuApellido]
Data Hash	[HASH SHA-256 de data_meta.json]
Fichero params.yaml	results/params.yaml (contiene config detallada de experimentos)
Fichero metrics.json	results/metrics.json (contiene métricas finales por experimento)
Rutas Figuras	figuras/ (todas las imágenes generadas)