Proyecto 1 - Deep Learning

Objetivo

Implementar un clasificador robusto en PyTorch para CIFAR-10, manejando desbalanceo de clases y aplicando técnicas avanzadas para combatir overfitting.

Instrucciones Generales

1. Formato: Notebook ejecutable (.ipynb) con código y análisis en celdas Markdown.
2. Dataset: CIFAR-10 (modificado para desbalanceo).
3. Prohibido: Uso de APIs de alto nivel para definir el modelo o código generado con modelos de lenguaje.
4. Entrega: 4 de abril a las 11:00 pm

Parte 1: Preparación de Datos con Desbalanceo Inducido (20%)

1. Cargar CIFAR-10 y dividir en entrenamiento (70%), validación (15%), prueba (15%).
2. Desbalanceo Artificial: En el conjunto de entrenamiento, reducir las muestras de 3 clases seleccionadas al 10% de su tamaño original (ej: aviones, barcos, ranas).
3. Aumento de Datos Básico: Solo para entrenamiento, aplicar normalización (utilice la media (0.5, 0.5, 0.5) y la std (0.5, 0.5, 0.5)).
4. Muestra un gráfico de barras con la distribución de clases antes/después del desbalanceo.

```
# Ejemplo de desbalanceo (deben implementarlo):
# Clases a desbalancear: [0, 8, 6] (avión, barco, rana)
# Para cada clase seleccionada, mantener solo el 10% de las muestras.
```

Parte 2: Modelo Base Sobrealimentado (25%)

1. Arquitectura:
    a. Red densamente conectada con mínimo 5 capas ocultas (ej: 3072 → 2048 → 1024 → 512 → 256 → 10).
    b. Activaciones ReLU, dropout NO permitido en esta etapa.
2. Entrenamiento:
    a. Optimizador: SGD con momentum (sin regularización L2).
    b. Función de pérdida: Cross-Entropy sin pesos de clases.
    c. Debe escoger las métricas para la evaluación.
    d. Entrenar por 100 épocas.

3. Análisis:
    a. Gráficas de pérdida y métricas (entrenamiento vs validación).
    b. Matriz de confusión para las 3 clases minoritarias.
    c. Explique: ¿Por qué el modelo probablemente tendrá bajo recall en las clases minoritarias?

Parte 3: Mitigación de Desbalanceo y Regularización (35%)

1. Pérdida Ponderada (10%)
    a. Calcula pesos de clases inversamente proporcionales a su frecuencia.
    b. Re-entrena el modelo base usando CrossEntropyLoss(weight=class_weights).
2. Técnicas Avanzadas (25%)
    a. Dropout + BatchNorm: Añade capas de Dropout (p=0.5) y BatchNorm después de cada capa oculta.
    b. Aumento de Datos Agresivo: Rotaciones aleatorias (±30°), volteo horizontal, ajustes de brillo/contraste.
    c. Investigue sobre Early Stopping: Implemente una estrategia de early stopping en el entrenamiento si la pérdida de validación no mejora en 10 épocas.
3. Análisis:
    a. Compara las curvas de entrenamiento vs las del modelo base.
    b. ¿Cómo afectó BatchNorm a la velocidad de convergencia? (Justifique con métricas).
    c. Para una clase minoritaria, calcule la mejora en F1-score tras aplicar los pesos.

Parte 4: Evaluación Final y Métricas (20%)

1. Métricas en Test:
    a. Reporte precisión, recall y F1-score por clase.
    b. Analice los resultados obtenidos.
    c. Seleccione la clase con peor performance y proponga una estrategia específica para mejorarla.