Laboratorio #7 Transfer Learning

Rodrigo Reyes
Francisco Acuña

1. Modelo VGG16

Arquitectura general

- **Tipo de modelo:** Red neuronal convolucional profunda (CNN).
- **Número total de capas:** 16 capas con pesos entrenables (13 convolucionales + 3 totalmente conectadas).
- **Estructura:**
    - Bloques de convolución seguidos por capas de *max pooling*.
    - Todas las convoluciones usan filtros de tamaño **3×3**, con paso (stride) de 1 y padding de 1.
    - Cada bloque aumenta el número de filtros:
        - Bloque 1: 64 filtros
        - Bloque 2: 128 filtros
        - Bloque 3: 256 filtros
        - Bloques 4 y 5: 512 filtros
    - Al final, tres capas totalmente conectadas: dos de 4096 neuronas y una de salida (1000 clases para ImageNet).
- **Tamaño de entrada:** 224×224 píxeles con 3 canales (RGB).
- **Número aproximado de parámetros:** 138 millones.

Dataset de preentrenamiento

- VGG16 fue **preentrenado en el dataset ImageNet**, que contiene más de **1.2 millones de imágenes** distribuidas en **1000 clases** de objetos cotidianos.

Adaptación a un nuevo problema de clasificación

Para reutilizar VGG16 en un nuevo problema (por ejemplo, con CIFAR-10 o CIFAR-100):

1. **Cargar pesos preentrenados** de ImageNet.
2. **Congelar las capas convolucionales** (para conservar las características aprendidas).
3. **Reemplazar las capas finales (fully connected)** con nuevas capas adaptadas al número de clases del nuevo dataset.
    - Ejemplo: si el nuevo dataset tiene 10 clases, la última capa debe tener 10 neuronas con activación *softmax*.
4. **Reentrenar parcialmente** las últimas capas convolucionales (*fine-tuning*) para mejorar la adaptación.

2. Dataset CIFAR

Características generales

Existen dos versiones principales:

- **CIFAR-10**
    - **Número de clases:** 10 (avión, automóvil, pájaro, gato, ciervo, perro, rana, caballo, barco, camión).
    - **Número total de imágenes:** 60,000 (50,000 de entrenamiento y 10,000 de prueba).
    - **Tamaño de imágenes:** 32×32 píxeles RGB.
- **CIFAR-100**

- Igual cantidad total de imágenes, pero con **100 clases** (cada una con 600 imágenes).

Transformaciones y normalización para usar con VGG16

Como VGG16 espera imágenes de **224×224 píxeles**, se deben aplicar las siguientes transformaciones:

1. **Redimensionamiento:**
    - Escalar las imágenes de 32×32 a 224×224 píxeles.
2. **Conversión a tensor:**
    - Convertir las imágenes a tensores de PyTorch o arreglos NumPy.
3. **Normalización:**

VGG16 fue entrenado con imágenes normalizadas según las estadísticas de ImageNet:

mean = [0.485, 0.456, 0.406]

std = [0.229, 0.224, 0.225]

- Por tanto, se deben normalizar las imágenes de CIFAR con estos valores.

3. Análisis y discusión de resultados

¿Qué método alcanzó el mejor rendimiento general?

El método que obtuvo el mejor rendimiento general fue el Fine-Tuning (Caso C), con una precisión de prueba (Test Accuracy) de 94.42% y una pérdida (Loss) de 0.2159.

Esto demuestra que permitir el entrenamiento completo del modelo ajustando tanto las capas convolucionales como las densas mejora significativamente la capacidad de generalización frente al dataset CIFAR-10, en comparación con los otros dos métodos.

¿Qué diferencias se observan entre feature extraction y fine-tuning?

La principal diferencia entre ambos métodos radica en la cantidad de capas que se actualizan durante el entrenamiento y, en consecuencia, en la capacidad del modelo para adaptarse al nuevo conjunto de datos. En el caso del feature extraction, se mantienen congeladas todas las capas convolucionales del modelo VGG16, utilizando las representaciones previamente aprendidas en ImageNet, y se entrena únicamente la capa de clasificación final. Esto permite obtener un rendimiento sólido con un menor costo computacional, alcanzando una precisión del 86.84% y una pérdida de 0.45, pero limitando la capacidad del modelo para especializarse en las características específicas de CIFAR-10.

Por el contrario, el fine-tuning permite ajustar todos los parámetros del modelo, incluidas las capas convolucionales, con una tasa de aprendizaje reducida. Esto posibilita que el modelo refine sus filtros y adaptaciones internas al nuevo dominio, logrando una mayor precisión (94.42%) y una pérdida menor (0.21). En síntesis, el fine-tuning ofrece una mejor generalización y desempeño global, aunque requiere más tiempo y recursos de cómputo que el enfoque de feature extraction.

¿Cuál de las tres estrategias resulta más eficiente considerando el tiempo de entrenamiento y la cantidad de parámetros ajustados?

En términos de eficiencia, el feature extraction se posiciona como la estrategia más equilibrada, ya que logra un buen rendimiento sin necesidad de ajustar todos los parámetros del modelo. Su tiempo de entrenamiento es considerablemente menor que el del fine-tuning, dado que solo se entrena la capa final del clasificador, lo que representa una fracción mínima de los aproximadamente millones de parámetros totales de VGG16.

El fine-tuning, aunque alcanza la mejor precisión (94.42%), requiere un entrenamiento más prolongado alrededor del doble de tiempo por época y una mayor carga computacional debido a la actualización completa del modelo.

Por otro lado, el método sin entrenamiento adicional es el más rápido, pero su rendimiento (10% de precisión) es claramente insuficiente. Por ello, si se busca un equilibrio entre tiempo de ejecución, consumo de recursos y desempeño, el feature extraction resulta ser la opción más eficiente, mientras que el fine-tuning es preferible solo cuando la prioridad absoluta es maximizar la precisión del modelo.