Student Maria Camila Garcia Ramirez
Teacher William David Prada Buitrago
Subject Introducción a la visión por computador

Taller en clase

Introducción

El presente informe describe el desarrollo de un sistema de entrenamiento e inferencia para modelos de **clasificación de imágenes** utilizando **Transfer Learning** en *PyTorch*, y la implementación de una **API REST** mediante *FastAPI* para realizar inferencias sobre imágenes individuales. Los modelos entrenados corresponden a las arquitecturas **VGG16** y **ResNet50**, aplicadas a los conjuntos de datos **CIFAR-10** y **CIFAR-100** respectivamente.

Dataset utilizado

Se emplearon los datasets públicos **CIFAR-10** y **CIFAR-100**, ampliamente utilizados para tareas de clasificación de imágenes. Cada uno contiene imágenes a color de 32×32 píxeles:

- **CIFAR-10:** 60,000 imágenes distribuidas en 10 clases (avión, automóvil, gato, perro, rana, etc.).
- **CIFAR-100:** 60,000 imágenes distribuidas en 100 clases más específicas (animales, objetos, frutas, paisajes, etc.).

Para efectos de experimentación y reducir tiempos de cómputo, se utilizó un subconjunto de **1,000 imágenes** en el entrenamiento de cada modelo, redimensionadas a 224×224 píxeles y normalizadas con los valores de *ImageNet*.

Entrenamiento de modelos

El entrenamiento se llevó a cabo en el entorno **Google Colab** con GPU habilitada. Se aplicó la técnica de **Transfer Learning**, congelando todas las capas del modelo preentrenado excepto la última capa de clasificación, la cual fue ajustada al número de clases correspondiente a cada dataset.

Configuración experimental

- Optimizador: `Adam`
- Tasa de aprendizaje: 1×10^{-3}
- Épocas: 5
- Tamaño de lote: 64
- Función de pérdida: `CrossEntropyLoss`
- Validación: 80% entrenamiento / 20% validación

Durante el entrenamiento se registraron las métricas por época (pérdida, exactitud y F1-score) en archivos CSV, y los pesos resultantes se almacenaron en la carpeta `modelos/`.

Resultados del entrenamiento

Los resultados obtenidos muestran un buen desempeño del modelo **VGG16** sobre CIFAR-10 y un rendimiento moderado de **ResNet50** sobre CIFAR-100, lo cual es esperable debido al mayor número de clases.

Modelo	Dataset	Val Accuracy (mejor)	Test Accuracy
VGG16	CIFAR-10	0.82	0.80
ResNet50	CIFAR-100	0.52	0.50

Table 1: Desempeño de los modelos entrenados.

Los pesos se almacenaron como:

```
modelos/
  vgg16_cifar10.pth
  resnet50_cifar100.pth
```

Ejemplo de inferencia

El módulo `inferencia_modelos.py` implementa la carga de los modelos entrenados, la normalización de las imágenes de entrada y la predicción de clases mediante la función `predict_image`.
Ejemplo de uso en Python:

```
from inferencia_modelos import load_model, predict_image

model, tfm, _ = load_model("vgg16", 10, "modelos/vgg16_cifar10.pth")
res = predict_image("ejemplo.png", model, tfm, output="full")
print(res)
```

Salida esperada:

```
{
  "prediction": {"class_index": 3, "class_name": "cat", "prob": 0.87},
  "topk": [
    {"class_index": 3, "class_name": "cat", "prob": 0.87},
    {"class_index": 5, "class_name": "dog", "prob": 0.09}
  ]
}
```

Implementación de la API

El archivo `api_inferencia.py` define un servicio REST en **FastAPI** que permite realizar inferencias a través de peticiones HTTP POST. El usuario puede seleccionar el modelo (`vgg16` o `resnet50`) y enviar una imagen en formato `.jpg`, `.png`, `.bmp` o `.webp`.

Ejecución del servidor

1. Instalar dependencias:

   ```
   pip install fastapi uvicorn torch torchvision pillow scikit-learn
   ```

2. Ejecutar el servidor desde la carpeta `src/`:

   ```
   python -m uvicorn api_inferencia:app --reload
   ```

3. Acceder al endpoint principal: **http://127.0.0.1:8000**

Ejemplo de petición

```
curl -X POST "http://127.0.0.1:8000/predict?model=vgg16&output=full" \
     -F "file=@ejemplo.png"
```

Ejemplo de respuesta:

```
{"prediction": {"class_name": "dog", "prob": 0.91}}
```

Estructura general del proyecto

```
Proyecto/
 modelos/
    vgg16_cifar10.pth
    resnet50_cifar100.pth
 src/
    api_inferencia.py
    inferencia_modelos.py
 data/
```

Conclusiones

El proyecto permitió integrar conceptos de visión por computador y aprendizaje profundo mediante el uso de **Transfer Learning** y el despliegue de un modelo como servicio. Se demostró que, incluso con un número reducido de muestras, los modelos preentrenados en *ImageNet* son capaces de generalizar adecuadamente sobre datasets más pequeños como CIFAR-10 y CIFAR-100. Además, la implementación de la **API de inferencia** en FastAPI ofrece una solución eficiente, modular y escalable para la integración de estos modelos en aplicaciones prácticas.