Informe Técnico del Sistema de Descripción Automática de Imágenes

Resumen del Proyecto

Este proyecto implementa un sistema de **Image Captioning** (descripción automática de imágenes) utilizando una arquitectura **Encoder-Decoder** con las siguientes características:

- **Encoder**: Extrae características visuales usando **EfficientNetB3** (pre-entrenado en ImageNet).
- **Decoder**: Genera descripciones usando una **LSTM con atención** (Bahdanau Attention).
- **Dataset**: **Flickr8k** (8,000 imágenes con 5 descripciones cada una).

El sistema convierte imágenes en texto describiendo su contenido en lenguaje natural.

Partes Clave del Código

1. Configuración Inicial

- **Descarga de dependencias**: TensorFlow, NLTK, NumPy, etc.
- **Descarga del dataset Flickr8k**:
  - Imágenes (`Flickr8k_Dataset.zip`).
  - Descripciones (`Flickr8k_text.zip`).
- **Preparación de directorios**: Organiza las imágenes y textos en carpetas estructuradas.

2. Preprocesamiento de Datos

Texto (Descripciones)

- **Limpieza**: Elimina puntuación y convierte a minúsculas.
- **Tokenización**: Usa NLTK para dividir las frases en palabras.
- **Vocabulario**:
  - Se construye contando palabras frecuentes (`min_count=3`).

    - Incluye tokens especiales: `<start>`, `<end>`, `<pad>`, `<unk>`.
    - Ejemplo: Si la palabra "dog" aparece 10 veces, se incluye; si "cat" aparece 2 veces, se descarta.

Imágenes

- **Extracción de características**:
    - Usa **EfficientNetB3** para convertir imágenes en vectores de 512 dimensiones.
    - **Aumento de datos**: Rotaciones, zoom y cambios de brillo para mejorar generalización.
- **Normalización**: Las imágenes se redimensionan a 300x300 y se normalizan (valores entre 0 y 1).

3. Arquitectura del Modelo

Encoder (EfficientNetB3)

- **Función**: Extrae patrones visuales (forma, color, objetos).
- **Capas congeladas**: Solo las últimas 6 capas se entrenan (para evitar sobreajuste).

Decoder (LSTM con Atención)

- **Embedding**: Convierte palabras en vectores densos (256 dimensiones).
- **LSTM**: Procesa secuencias de texto (512 unidades).
- **Mecanismo de atención (Bahdanau)**:
    - Permite al modelo "fijarse" en partes relevantes de la imagen al generar cada palabra.
    - Ejemplo: Si la imagen tiene un perro, el modelo pondrá más atención en esa región al escribir "dog".
- **Salida**: Capa densa con activación `softmax` para predecir la siguiente palabra.

4. Entrenamiento

Hiperparámetros

- **Batch size**: 64.
- **Épocas**: 35 (entrenamiento base) + 2 (fine-tuning).

- **Optimizador**: Adam con learning rate ajustable (planificación dinámica).
- **Función de pérdida**: `SparseCategoricalCrossentropy` (para problemas de clasificación).

Callbacks

- **Early Stopping**: Detiene el entrenamiento si no hay mejora en 5 épocas.
- **ReduceLROnPlateau**: Reduce el learning rate si el loss no mejora.
- **TensorBoard**: Monitorea métricas en tiempo real.

Proceso

1. **Entrenamiento inicial**:
    a. Learning rate alto (5e-4) para convergencia rápida.
2. **Fine-tuning**:
    a. Descongela capas del encoder para ajuste fino.
    b. Learning rate bajo (1e-5) para refinamiento.

5. Evaluación y Resultados

- **Métricas**:
    - **Accuracy**: Precisión en la predicción de palabras.
    - **Loss**: Pérdida (cross-entropy).
- **Gráficos**:
    - Se observa convergencia estable en training/validation.
    - Ejemplo: `val_accuracy` ~60% (depende del tamaño del vocabulario).
- **Guardado del modelo**:
    - Se exporta el modelo en formato `.keras`.
    - Se guarda el vocabulario (`vocab_final.pkl`) para inferencia futura.

Problemas y Soluciones

1. **Vocabulario muy grande**:
    a. Se filtraron palabras raras (`min_count=3`).
2. **Overfitting**:
    a. Aumento de datos (rotaciones, zoom) + dropout (0.5).
3. **Lentitud en entrenamiento**:
    a. Uso de `tf.data.Dataset` para carga eficiente.

Conclusión

El sistema es capaz de generar descripciones coherentes para imágenes, aunque su calidad depende de:

- Tamaño del vocabulario.
- Complejidad del mecanismo de atención.
- Cantidad de datos de entrenamiento.

Mejoras futuras:

- Usar transformers (ViT + GPT) en lugar de LSTM.
- Dataset más grande (COCO, Flickr30k).