Universidad Nacional de Colombia - Sede Manizales

GCPDS

Modelos de Detección y Segmentación en Entornos Móviles Android

Este repositorio contiene implementaciones y modelos optimizados para detección y segmentación de objetos utilizando **TFLite** en dispositivos Android.

¿Qué es TFLite?

TensorFlow Lite (**TFLite**) es un framework ligero diseñado para ejecutar modelos de aprendizaje automático en dispositivos móviles y embebidos. Ofrece baja latencia y un rendimiento destacado en entornos con recursos limitados. Este framework se integra fácilmente con Android Studio, permitiendo desplegar modelos de **deep learning** sin la necesidad de hardware costoso o especializado.

Estructura del Repositorio

```
.
├── LICENSE
├── NOTICE
├── README.md
├── TransformToTFLITE/
│   ├── transformar.ipynb
├── tree.py
├── YOLOv10-Object-Detector-Android-Tflite/
│   ├── app/
│   │   ├── build.gradle.kts
│   │   ├── proguard-rules.pro
│   │   └── src/
│   │       ├── androidTest/
│   │       ├── main/
│   │       └── test/
```

```
│   ├── build.gradle.kts
│   ├── gradle/
│   │   ├── libs.versions.toml
│   │   └── wrapper/
│   │       ├── gradle-wrapper.jar
│   │       └── gradle-wrapper.properties
├── YOLOv8-Image-Classification-Android-Tflite/
│   ├── app/
│   │   ├── build.gradle.kts
│   │   ├── proguard-rules.pro
│   │   └── src/
│   │       ├── androidTest/
│   │       ├── main/
│   │       └── test/
├── YOLOv8-Instance-Segmentation-Android-Tflite/
│   ├── app/
│   │   ├── build.gradle.kts
│   │   ├── proguard-rules.pro
│   │   └── src/
│   │       ├── androidTest/
│   │       ├── main/
│   │       └── test/
├── YOLOv9-Object-Detector-Android-Tflite/
│   ├── app/
│   │   ├── build.gradle.kts
│   │   ├── proguard-rules.pro
│   │   └── src/
│   │       ├── androidTest/
│   │       ├── main/
│   │       └── test/
└── YOLOv8-Object-Detector-Android-Tflite/
    ├── app/
    │   ├── build.gradle.kts
    │   ├── proguard-rules.pro
    │   └── src/
    │       ├── androidTest/
    │       ├── main/
    │       └── test/
    │
```

Cada subdirectorio contiene un proyecto Android Studio con los archivos necesarios para implementar modelos TFLite de diferentes versiones de YOLO, como YOLOv8, YOLOv9 y YOLOv10.

Tutoriales

La carpeta **Tutoriales** incluye guías paso a paso para instalar, configurar y ejecutar los modelos.

¿Cómo Ejecutar los Modelos?

Requisitos Previos

1. **Instalar Android Studio**

   Descarga la última versión de Android Studio e instálala en tu sistema operativo.

2. **Clonar el Repositorio**

   Utiliza el siguiente comando:

   ```
   git clone https://github.com/tu-repositorio.git
   ```

3. **Abrir el Proyecto**

   En Android Studio, selecciona `File > Open` y navega hasta la carpeta del modelo correspondiente (por ejemplo, `YOLOv8-Object-Detector-Android-Tflite/`).

Una vez cargado el proyecto, estará listo para ser compilado y ejecutado en un dispositivo o emulador Android.

Instrucciones para Conversión a TFLite

El directorio `TransformToTFLITE` contiene un cuaderno llamado `transformar.ipynb`, que describe el proceso para convertir modelos preentrenados al formato TFLite:

1. **Abrir el Notebook**

   Ejecuta el archivo utilizando Jupyter Notebook o Google Colab.

2. **Pasos en el Notebook**

   Sigue las instrucciones detalladas para:

- Preparar los modelos.
- Optimizarlos para dispositivos móviles.
- Exportarlos al formato `.tflite`.

Este flujo permite ajustar los modelos a las necesidades específicas de la aplicación.

Uso de los Proyectos y Selección de Modelos

Para trabajar con un proyecto (por ejemplo, `YOLOv10-Object-Detector-Android-Tflite`), realiza los siguientes pasos:

1. **Abrir el Proyecto**

   Abre la carpeta del proyecto en Android Studio (`File > Open`).

2. **Actualizar el Modelo `.tflite`**

   Cambia el archivo del modelo en la carpeta `assets` dependiendo de la versión y el tipo de cuantización deseada:

   ```
   YOLOv10-Object-Detector-Android-Tflite\app\src\main\assets\yolov10n_float16.tflite
   YOLOv10-Object-Detector-Android-Tflite\app\src\main\assets\yolov10n_int8.tflite
   ```

   Configura el archivo correcto en el código para que el intérprete cargue el modelo adecuado.

Diferencia entre int8 y float16

- **float16**

  Cuantización en punto flotante de 16 bits. Proporciona un equilibrio entre precisión y tamaño del modelo. Es ideal para aplicaciones que requieren mantener buena precisión pero también buscan reducir el tamaño del modelo.

- **int8**

  Cuantización a enteros de 8 bits, lo que reduce significativamente el tamaño y aumenta la velocidad de inferencia. Útil para dispositivos con recursos limitados o aplicaciones que priorizan baja latencia.

La elección dependerá de las capacidades del dispositivo y los requisitos de la aplicación.

Explicación del Código (Detector.kt)

El archivo `Detector.kt` define una clase que procesa imágenes y ejecuta inferencias utilizando modelos TFLite. Algunos aspectos clave son:

1. **Inicialización del Intérprete TFLite**

   Se configura el intérprete para ejecutar el modelo `.tflite` y, opcionalmente, se habilita la aceleración GPU utilizando `GpuDelegate`.

2. **Procesamiento de Imagen**

   La imagen de entrada se redimensiona y normaliza utilizando un `ImageProcessor`. Esto asegura que los datos cumplan con los requisitos del modelo.

3. **Parámetros Clave**

   - `tensorWidth` y `tensorHeight`: Tamaño esperado de la imagen de entrada.
   - `numChannel` y `numElements`: Dimensiones del tensor de salida.
   - `CONFIDENCE_THRESHOLD`: Umbral de confianza para filtrar detecciones con baja probabilidad.

4. **Método** `detect(frame: Bitmap)`

   - Redimensiona la imagen al tamaño requerido por el tensor.
   - Ejecuta la inferencia con el intérprete.
   - Filtra y retorna las detecciones con mayor confianza.

5. **Reinicio con GPU**

   La función `restart(isGpu: Boolean)` reinicia el intérprete y habilita la GPU en dispositivos compatibles para mejorar el rendimiento.

Personalización del Código

Para ajustar el rendimiento, puedes modificar:

- `CONFIDENCE_THRESHOLD`: Ajusta la sensibilidad de las detecciones.
- `setNumThreads`: Incrementa o reduce el número de hilos utilizados.
- `GpuDelegate`: Habilita o deshabilita la aceleración por GPU.

Estas opciones permiten optimizar el modelo según las capacidades del dispositivo y los requisitos del proyecto.