Real-Time Human Detection and Tracking System on Edge Devices

Overview

This documentation provides a detailed account of the real-time object detection and tracking system developed using YOLOv4 and Deep SORT, optimized for deployment on embedded edge devices like the Jetson Nano. The system efficiently detects and tracks multiple objects in a video stream with real-time performance.

---

1. Model Selection and Development

YOLOv4

Justification:

- YOLOv4 was chosen for its superior balance between speed and accuracy for object detection.
- It supports efficient real-time inference while maintaining competitive performance compared to other state-of-the-art models.

Deep SORT

Justification:

- Deep SORT provides robust tracking by associating detected objects across frames.
- The lightweight `mars-small128.pb` model for feature extraction enables efficient tracking on resource-constrained devices.

---

2. Training Methodology and Dataset Description

Dataset

- Pre-trained weights for YOLOv4 and Deep SORT were utilized to speed up development and optimize performance.

- Custom fine-tuning was not required, as the existing models performed well for the target object categories.

---

3. Optimization Techniques for Edge Deployment

Model Conversion

- The YOLOv4 model was converted to TensorFlow Lite (TFLite) format for edge compatibility.
    - This reduced model size and improved inference speed.

Performance Optimizations

- Reduced input resolution to balance detection accuracy and speed.
- Capped the input frame rate at 15 FPS for optimal real-time performance.

Deep SORT Model

- Used the lightweight `mars-small128.pb` model to minimize computational overhead.

---

4. System Architecture and Design

Core Components

- **Object Detection:** YOLOv4 (TFLite)
- **Object Tracking:** Deep SORT using `mars-small128.pb`
- **Communication:** Flask app for video stream management

Workflow

1. Input video frames are captured from the camera.
2. YOLOv4 detects objects within each frame.
3. Deep SORT associates detected objects between frames for tracking.
4. The processed frames are displayed in real-time.

---

5. GUI Design and Functionality

Flask Application

The web-based GUI provides the following functionalities:

- **Start Video Streaming:** Initiates the video stream and starts object detection and tracking.
- **Stop Video Streaming:** Stops the video stream and terminates tracking.

---

6. Instructions for Setting Up and Running the Application

Prerequisites

- Ubuntu 20.04
- Python 3.8
- TensorFlow Lite runtime
- Flask framework

Installation Steps

1. **Set up Jetson Nano:**
    - Flash the SD card with JetPack OS.
    - Boot the Nano and set up network connectivity.

Install Dependencies:
sudo apt-get update
sudo apt-get install python3-pip

2. **Deploy the Application:**
    - Clone the project repository.
    - Navigate to the project directory.
    - Python3 -m venv env
    - . env/bin/activate
    - pip install -e requirements.txt
3. python app.py
4. **Access the Application:**
    - Open a web browser and navigate to `http://<Jetson_IP>:5000`.

Usage Instructions

1. Click **Start Video Streaming** to begin object detection and tracking.

2. Monitor the real-time video feed.
3. Click **Stop Video Streaming** to end the process.

---

7. Performance Considerations

- Ensure that the frame resolution is optimized for the Jetson Nano.
- For best results, maintain moderate lighting conditions.
- Use the TensorRT optimization feature for improved inference speed.

This documentation should assist developers and stakeholders in understanding the architecture, functionality, and deployment process of the real-time object detection and tracking system.