Home Assignment: Real-Time Video Analysis Pipeline – Captain's Eye

# Objective

Assess your proficiency in Computer Vision.

Your mission is to build an end-to-end pipeline that reads video from an RTSP source, applies a given PyTorch model, and writes the results to a JSON file.

The model should be an Object Detection model – Specifically detecting objects of class “person”. If the model you are using has more classes – don’t write them to the output file. Please do not train your own model, the task is to use as many tools as possible to produce a working pipeline.

# Requirements

1. Python 3.x
2. OpenCV (For reading RTSP video)
3. PyTorch (For model inference)
4. Any JSON handling library
5. Conda environment

# Task

1. **Stream RTSP Video**: Gather frames data from an RTSP video address.
2. **Model Inference**: Pre-process the captured frame and perform inference using a given PyTorch model. You may use a pre-trained model for demonstration purposes.
3. **Result Processing**: Post-process the model's output and organize the data you wish to save. Needed output is YOLO relative format.
4. **Write to JSON**: In a separate thread from the inference thread, write the model's output and any meta-data (e.g., timestamp of the frame) to a JSON file. The schema is not important.
5. **CUDA Support**: Make your pipeline auto-detect whether CUDA is available and use it if so.
6. **Automation**: The pipeline should run continuously, processing one frame at a time and appending the results to the same JSON file.

# Guidelines

* Include a `main.py` file, that has a CLI command accepting an RTSP address and launching your pipeline.
* There is no need to handle batches.
* Use only PyTorch models.
* Feel free to use any library or open-source model available on the internet. There are no restrictions. Recommended: Use YOLOv8.
* Include a `requirements.txt` file specifying the necessary packages for your project.
* Add a README.md that outlines:
  + Clear instructions on how to install and run your project.
  + Any assumptions or simplifications you made.
  + An explanation of the choices you made in building your pipeline.
  + Answers (in English) for the questions in the “Questions” section of this document

# Submission

Please send a zip file containing all the required files:

1. README.md
2. Requirements.txt
3. Main.py
4. Any additional source code

# Evaluation

Your assignment will be evaluated based on the most important value – a working code as described. In addition:

* Code Quality
* Efficiency of your pipeline
* Simplicity and readability
* Proper error handling and edge case coverage
* Compliance with best practices

# Questions

1. How would you improve your code to handle multiple stream sources?
2. What is the advantage of using a GPU for inference? Can the GPU use for decoding as well?
3. For each following SDK/Python Package please write a short sentence describing it’s purpose and use cases, only if you are familiar with it. If you are familiar with it, describe how would it be relevant to your implementation:
   1. asyncio
   2. GStreamer
   3. Nvidia DeepStream SDK
   4. Docker