**Introduction**

This project focuses on implementing a robust pipeline for video processing and live object tracking using the YOLOv5 (You Only Look Once) model and the Deep SORT (Simple Online and Realtime Tracking) algorithm. The goal is to demonstrate real-time person detection and tracking capabilities using both pre-recorded videos from YouTube and live webcam input.

**Approach**

The approach consists of several key components:

1. **Downloading the Video**: The first step involves downloading a video from YouTube using the yt-dlp library. This library facilitates easy retrieval of online video content for analysis.
2. **Object Detection Model**: The YOLOv5 model is employed for detecting objects in video frames. Specifically, the yolov5s variant is chosen for its balance between speed and accuracy, making it suitable for real-time applications.
3. **Video Processing**: A video processing function reads frames from the downloaded video and performs the following:
   * Resizes each frame for consistency and faster processing.
   * Converts the frame from BGR to RGB color space, which is required for YOLOv5.
   * Utilizes the YOLOv5 model to detect persons in each frame, filtering out the detection results to focus on human targets.
4. **Object Tracking**: The Deep SORT algorithm is integrated to maintain the identity of detected persons across frames. It assigns unique IDs to each detected individual, allowing for consistent tracking throughout the video.
5. **Live Webcam Testing**: After processing the video, the program transitions to a live webcam test, where it utilizes the same detection and tracking pipeline to monitor persons in real time.

**1. Installing Required Libraries**

A screen shot of a computer program

Description automatically generated

**Purpose**: This code installs the necessary libraries to run the video processing and object detection tasks.

* torch and torchvision: Libraries for deep learning and computer vision.
* opencv-python: OpenCV library for video processing and image manipulation.
* yolov5: The YOLO (You Only Look Once) object detection model, specifically a version that can detect multiple objects in real-time.
* deep\_sort\_realtime: A tracking algorithm to keep track of detected objects across frames.
* yt-dlp: A tool to download videos from YouTube and other platforms.
* ffmpeg-python: A wrapper for the FFmpeg tool, which is used for handling video and audio processing.

**Steps to Install FFmpeg:**

**Windows Installation:**

* **Download FFmpeg**:
  + Go to the official FFmpeg download page: <https://ffmpeg.org/download.html>.
  + Choose the "Windows builds from gyan.dev" option or go directly to the Windows FFmpeg Build.
  + Download the latest **full release** zip file under the "Release builds" section (e.g., ffmpeg-git-full.7z).
* **Extract the Zip File**:
  + Extract the downloaded .7z file to a folder on your system. You can use a tool like **7-Zip** or **WinRAR** for extraction.
* **Set the FFmpeg Path in System Environment Variables**:
  + Open the folder where you extracted FFmpeg.
  + Copy the path of the bin directory inside the extracted folder (e.g., C:\ffmpeg\bin).
  + Right-click on **This PC** (or **My Computer**) and choose **Properties**.
  + Click on **Advanced system settings** > **Environment Variables**.
  + Under **System Variables**, scroll down and select **Path**, then click **Edit**.
  + In the **Edit Environment Variable** dialog, click **New** and paste the path you copied (e.g., C:\ffmpeg\bin).
  + Click **OK** to close all dialog boxes.
* **Verify Installation**:
  + Open a new Command Prompt and type ffmpeg. If FFmpeg is installed correctly, you will see output with the FFmpeg version and build details.

2. **Downloading a YouTube Video**A screen shot of a computer

Description automatically generated

**Purpose**: This block of code downloads a YouTube video.

* The yt\_dlp library is imported for downloading.
* ydl\_opts specifies output settings:
  + outtmpl: Filename for the downloaded video.
  + ffmpeg\_location: Path to the FFmpeg binary, necessary for video processing.
* The YouTube URL is defined, and the video is downloaded using the YoutubeDL class.
* **Note**: Using r'' for paths prevents escape sequences from being interpreted.

3. **Importing Necessary Libraries**A screen shot of a computer

Description automatically generated

* **Purpose**: Importing libraries for functionality.
  + warnings: To manage warnings in the code.
  + torch: For loading and using the YOLOv5 model.
  + cv2: OpenCV for video processing.
  + DeepSort: To track objects across frames.
  + time: For measuring execution time.
  + cv2\_imshow and display: For showing images in a Jupyter environment.

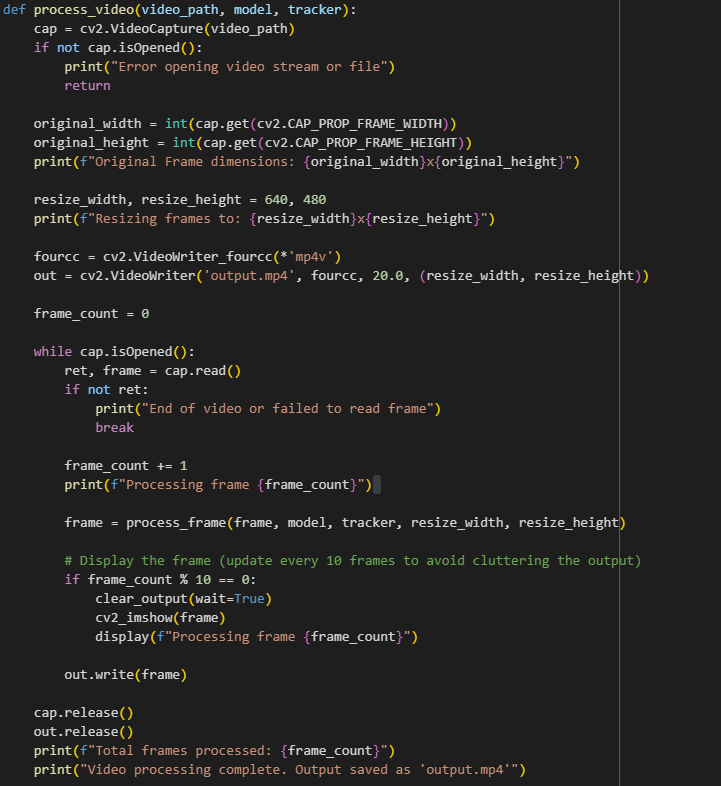
**4. Suppressing Warnings**

****

* **Purpose: Suppresses specific warnings in the output to keep the console clean.**

**5. Video Processing Function**

* **Purpose: Processes the downloaded video frame by frame.**
  + **cv2.VideoCapture(video\_path): Opens the video file.**
  + **Checks if the video file opened successfully.**
  + **Gets the original video dimensions.**
  + **Sets new dimensions for resizing frames.**
  + **Initializes the cv2.VideoWriter to save the processed output.**
  + **Enters a loop to read frames until the video ends or fails:**
    - **Reads a frame and processes it using process\_frame.**
    - **Displays the processed frame every 10 frames.**
    - **Writes the processed frame to the output video.**
  + **Releases resources after processing.**

****

**6. Frame Processing Function**

**A computer screen shot of a program

Description automatically generated**

* **Purpose: Processes individual frames for object detection and tracking.**
  + **Resizes the frame and converts it to RGB for YOLO model input.**
  + **Uses the YOLOv5 model to get detections from the frame.**
  + **Filters out detections for persons (class 0).**
  + **If there are person detections, updates the DeepSort tracker with those detections.**
  + **Draws bounding boxes around detected persons with their IDs.**
  + **Calculates and displays frames per second (FPS).**

**7. Live Webcam Testing Function**

**A screen shot of a computer program

Description automatically generated**

* **Purpose: To perform live object detection using the webcam.**
  + **Initializes the webcam capture.**
  + **Resizes the webcam frames.**
  + **Continuously reads frames from the webcam:**
    - **Processes each frame similar to video processing.**
    - **Displays the processed frame in a window.**
    - **Exits when 'q' is pressed.**

**8. Main Function**

**A computer screen shot of text

Description automatically generated**

* **Purpose**: The entry point of the script.
  + Loads the YOLOv5 model using torch.hub.load to access pre-trained models directly from the Ultralytics repository.
  + Uses the smaller model yolov5s for faster inference at the cost of some accuracy.
  + Checks for CUDA availability to utilize GPU if available.
  + Initializes the DeepSort tracker.
  + Specifies the path of the downloaded video and processes it.
  + Finally, starts live webcam testing.

**Why YOLOv5?**

* **Real-Time Performance**: YOLOv5 is optimized for speed and can process frames quickly enough for real-time applications, making it suitable for tasks like surveillance and autonomous driving.
* **Ease of Use**: With pre-trained models available, it's easy to get started with object detection tasks without extensive training time.
* **Accuracy**: YOLOv5 balances accuracy and speed effectively, making it a popular choice for many applications.
* **Community Support**: Being part of the Ultralytics community means continuous updates and improvements, as well as extensive documentation.

**Findings**

The implementation successfully processes the downloaded YouTube video, detecting and tracking persons throughout the frames. Key findings include:

* The YOLOv5 model exhibited high efficiency in detecting objects, even in a dynamically changing environment, with minimal latency between frames.
* The Deep SORT tracker effectively maintained the identity of detected persons, demonstrating robustness in tracking even when individuals crossed paths or were momentarily occluded.
* During the live webcam testing, the system performed reliably, providing a real-time view of detected individuals.

**Conclusion**

This project demonstrates the potential of integrating state-of-the-art object detection and tracking algorithms to develop a comprehensive system capable of real-time person monitoring. The combination of YOLOv5 for object detection and Deep SORT for tracking results in an efficient and effective solution for various applications, such as security surveillance and human-computer interaction. Future enhancements could include optimizing the model for accuracy improvements, incorporating multi-object tracking in crowded environments, and extending the system to detect other classes of objects beyond humans. Overall, this implementation provides a solid foundation for further exploration and development in the field of computer vision.