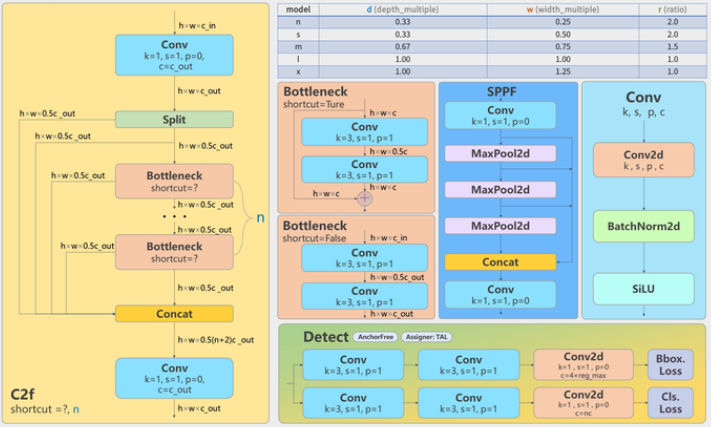
**Detection and Tracking: YOLOv8 & DeepSORT**

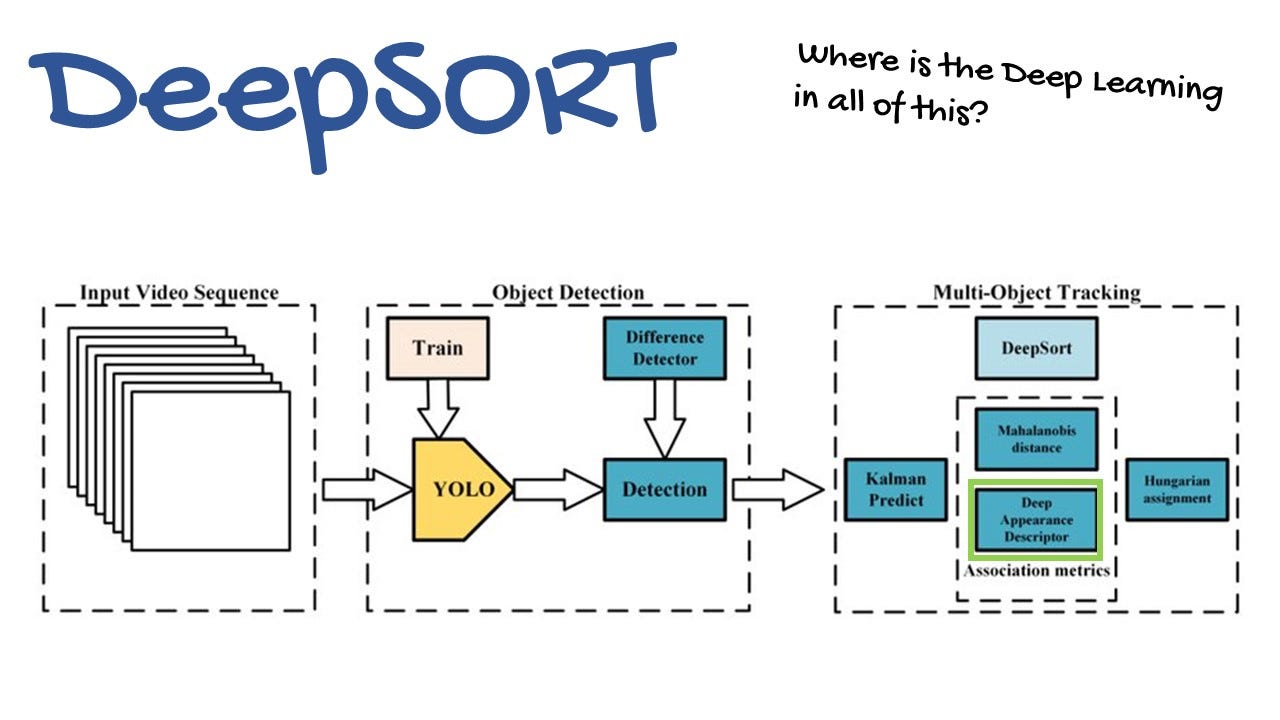
* **Choice of Detection Method:**
  + YOLOv8 was selected for object detection due to its real-time performance and accuracy. YOLOv8 is a model for object detection, capable of recognizing objects within images or video frames and providing bounding box coordinates.
  + This model is effective for applications like video processing, where high-speed object detection is essential. YOLOv8 also benefits from its ability to handle various object classes, which in this case is limited to tracking the "person" class (classes=[0]).

The [YOLOv8 architecture](https://yolov8.org/yolov8-architecture/) can be broadly divided into three main components:

* Backbone: This is the convolutional neural network (CNN) responsible for extracting features from the input image. YOLOv8 uses a custom CSPDarknet53 backbone, which employs cross-stage partial connections to improve information flow between layers and boost accuracy.
* Neck: The neck, also known as the feature extractor, merges feature maps from different stages of the backbone to capture information at various scales. YOLOv8 Architecture utilizes a novel C2f module instead of the traditional Feature Pyramid Network (FPN). This module combines high-level semantic features with low-level spatial information, leading to improved detection accuracy, especially for small objects.
* Head: The head is responsible for making predictions. YOLOv8 employs multiple detection modules that predict bounding boxes, objectness scores, and class probabilities for each grid cell in the feature map. These predictions are then aggregated to obtain the final detections.



* **Tracking Method:**
  + Tracking is handled using YOLOv8's built-in tracking functionality (model.track). This method is optimized DeepSORT to track detected objects across frames. It assigns unique IDs to each object (in this case, persons) and uses the Intersection over Union (IoU) threshold to determine how objects are matched across frames.
  + The IoU (Intersection over Union) threshold is set to 0.6 to control how tightly bounding boxes must overlap to be considered the same object across frames. A higher IoU value would result in more strict tracking, possibly causing missed tracks for objects with slight motion.
  + The confidence threshold is set to 0.25 (conf=0.25), ensuring that only detections with a confidence level above this value are considered for tracking.



**Optimizations and Key Decisions:**

1. **File Handling and Output Folder Management:**
   * The function clear\_and\_create\_output\_folder() ensures that the output folder (runs/detect/track) is cleaned up before starting a new video processing session. This avoids accumulation of files from previous runs.
   * Error handling is implemented to skip files that cannot be deleted due to permission issues, ensuring robustness in case of access-related errors.
2. **Efficient Video Processing:**
   * Real-time processing is enabled by streaming video frames during detection (stream=True), which allows for faster processing and output generation, preventing memory overload from loading the entire video into memory.
   * Metrics Calculation:
     + The total frames processed and the frames per second (FPS) are calculated, providing a measure of processing speed and efficiency.
     + Continuity issues are tracked by comparing object IDs across frames, which can help in identifying potential tracking errors or interruptions in continuity.
3. **Optimized Video Output:**
   * The output video is saved in .avi format (output\_filename = f"{os.path.splitext(filename)[0]}.avi"), ensuring compatibility with most media players and easier handling in further processing steps.
4. **Streamlined User Experience with Flask:**
   * Flask is used to serve the application, with a simple HTML interface allowing users to upload a video, track persons, and view or download the processed video.
   * The processed video and relevant metrics (e.g., total time, frames, FPS) are displayed on the webpage, ensuring users can quickly evaluate the results.
5. **Error Handling and Robustness:**
   * Basic error handling for file permissions and missing files ensures that the application remains stable and responsive even when unexpected issues arise during file deletion or video processing.

**Potential Improvements / Future Enhancements:**

* Tracking Continuity Improvements: The current implementation counts continuity issues simply when an object ID appears multiple times, but more sophisticated methods, such as tracking algorithms (e.g., Kalman filtering), could be added to improve tracking accuracy and handle occlusions.
* Scalability: The system could be optimized for large-scale video processing or deployed with a queue for processing multiple videos simultaneously, utilizing distributed computing resources.
* Customization of Tracking: Currently, tracking is restricted to "person" detection. Future versions could allow users to choose other classes for tracking or detect multiple objects concurrently.