<u>**Shivam Goyal**</u>

Overview:

Project Name	Task Type	Computation (GPU/CPU)	Model Used	Dataset Used	Key Functions
DETR Object Detection	Object Detection & Tracking	GPU	facebook/detr-resnet-50	None (Pretrained Model Used)	Object detection on video, heatmap generation
DeepSORT RPC Tracking	Object Detection & Tracking	GPU	MobileNetV2 (α=0.35), DeepSORT	Retail Product Checkout (RPC) Dataset (COCO)	Object detection, tracking using DeepSORT
YOLOv8 & SORT Analysis	Object Detection, Heatmap Generation, Customer Analysis (Charts)	CPU	YOLOv8, SORT, ByteTrack	CCTV Footage (Custom Data)	Object detection, tracking, heatmap creation, customer behavior analysis with charts

Dataset Description:

Retail Product Checkout (RPC) Dataset

- **Total Images:** 83,739
- **Product Categories:** 200
- **Sub-Categories:** 17 (e.g., Puffed Food)
- **Dataset Split:**
  - **Training Set:** 53,739 single-product images (controlled environment)
  - **Testing Set:** 30,000 multi-product images (real-world checkout simulation)
  - **Difficulty Levels:** Easy, Medium, Hard (based on product count & variety)

Class Distribution

- 200 product categories with varying image counts
- Training set: Single-product images
- Testing set: Multi-product images (mimics checkout scenarios)

Annotation Details

- **Bounding Boxes:** Product location in images
- **Category Labels:** Product class identification
- **Instance Counts:** Number of each product in an image

The RPC dataset supports object detection, recognition, and counting, aiding research in automatic checkout systems.

CPU Based Task:

- Designed for object detection and tracking using YOLOv8 and SORT tracker.
- Generates heatmaps for analyzing customer interactions.
- Optimized for CPU execution for retail analytics applications.

Dependencies and Setup

- Installs required packages: ultralytics (YOLO), filterpy (tracking), supervision (heatmap generation).
- Downloads and integrates the SORT tracking algorithm.

Configuration Parameters (CFG Class)

- Defines key parameters such as model weights, confidence thresholds, video paths, and tracking settings.

Object Detection and Tracking Process

Step 1: Load YOLO Model and Tracker

- Initializes the YOLOv8 model and the SORT tracker for object detection and tracking.

Step 2: Extract Video Properties

- Retrieves properties like frame rate, resolution, and codec for efficient processing.

Step 3: Perform Object Detection and Tracking

- YOLO detects objects, and SORT assigns unique tracking IDs.
- Applies bounding boxes and labels to detected objects.

Heatmap Generation

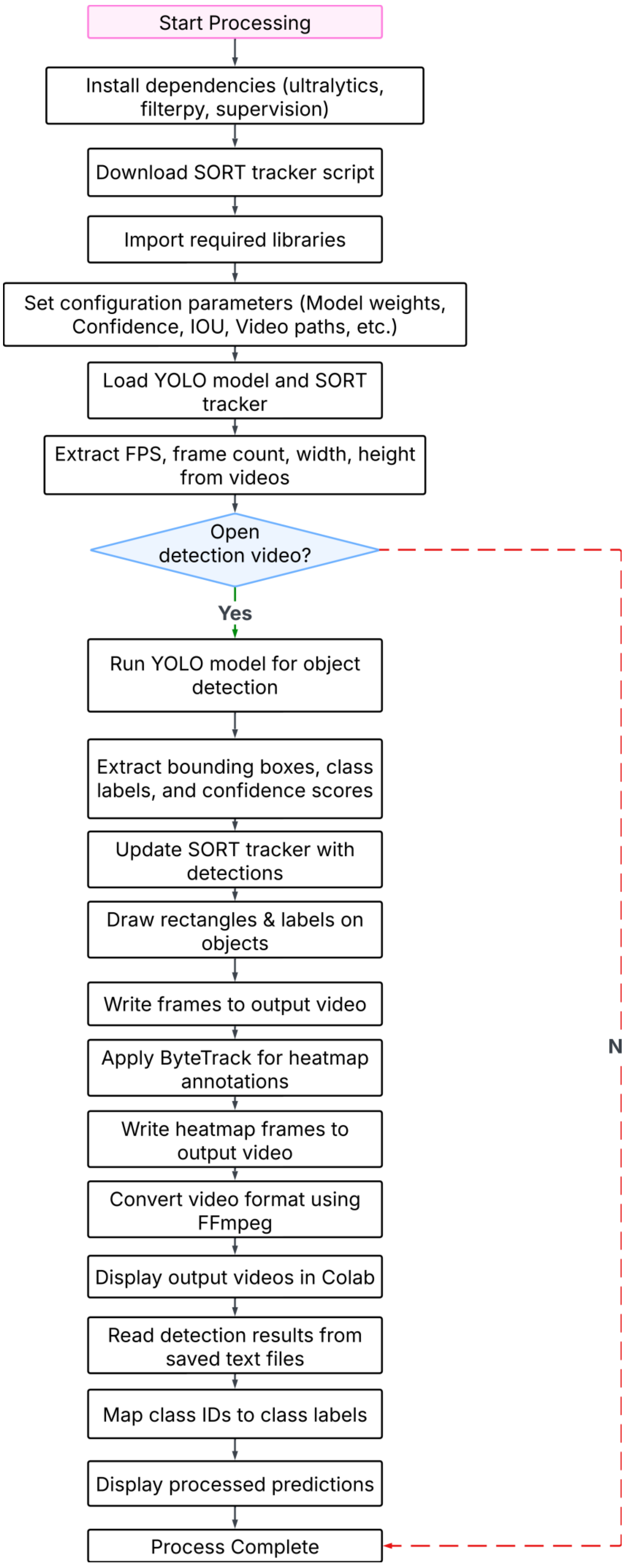
- Uses supervision library to generate heatmaps.
- Refines tracking using ByteTrack for smoother detection.
- Overlay heatmaps visualize object movement over time.

Video Encoding and Display

- Uses FFmpeg for compressing and encoding output videos.
- Embeds final processed videos for review.

Results Analysis with Pandas

- Loads detection results into a Pandas DataFrame.
- Allows further post-processing, like counting objects or analyzing confidence scores.



Start Processing
Install dependencies (ultralytics, filterpy, supervision)
Download SORT tracker script
Import required libraries
Set configuration parameters (Model weights, Confidence, IOU, Video paths, etc.)
Load YOLO model and SORT tracker
Extract FPS, frame count, width, height from videos
Open detection video?
Yes
N
Run YOLO model for object detection
Extract bounding boxes, class labels, and confidence scores
Update SORT tracker with detections
Draw rectangles & labels on objects
Write frames to output video
Apply ByteTrack for heatmap annotations
Write heatmap frames to output video
Convert video format using FFmpeg
Display output videos in Colab
Read detection results from saved text files
Map class IDs to class labels
Display processed predictions
Process Complete

GPU Based Task:

- Utilizes DEtection TRansformer (DETR) for object detection.
- Designed for GPU-based processing, improving detection accuracy and efficiency.
- Generates heatmaps from CCTV footage to analyze object movement patterns.

Dependencies and Setup

- Installs Detectron2 for advanced object detection.
- Uses PyTorch, OpenCV, Transformers, and PIL for image and video processing.

Loading the DETR Model

- Loads the facebook/detr-resnet-50 model for object detection.
- Uses a corresponding image processor to format input frames.

Object Detection in Video

Step 1: Video Capture and Output Initialization

- Reads video frames using OpenCV.
- Prepares output video writer with the same frame size and frame rate.

Step 2: Object Detection with DETR

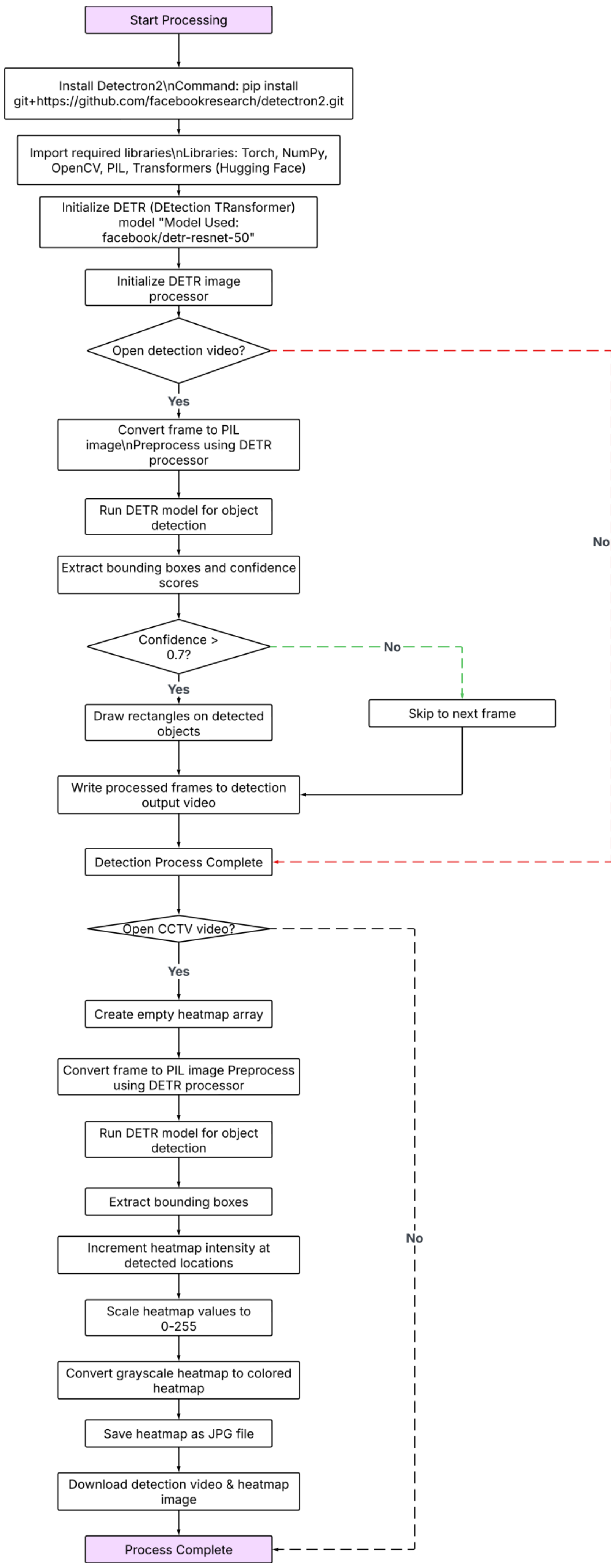
- Converts frames to PIL format for processing.
- Passes images through the DETR model for object detection.
- Extracts bounding boxes and applies confidence filtering.
- Draws bounding boxes on frames for visualization.

Step 3: Save Processed Video

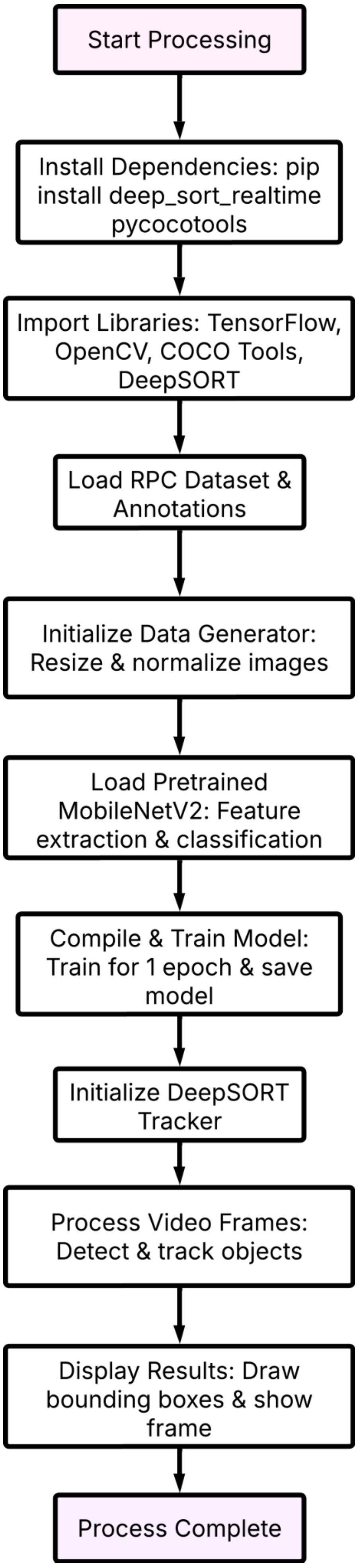
- Writes the annotated frames into an output MP4 file.

Heatmap Generation from CCTV Footage

- Reads frames from a CCTV video using OpenCV.
- Extracts object locations and overlays detections onto a heatmap matrix.
- Normalizes and colorizes the heatmap using OpenCV’s COLORMAP_JET.
- Saves the generated heatmap as an image file.


Start Processing
Install Detectron2\nCommand: pip install git+https://github.com/facebookresearch/detectron2.git
Import required libraries\nLibraries: Torch, NumPy, OpenCV, PIL, Transformers (Hugging Face)
Initialize DETR (DEtection TRansformer) model "Model Used: facebook/detr-resnet-50"
Initialize DETR image processor
Open detection video?
Yes
No
Convert frame to PIL image\nPreprocess using DETR processor
Run DETR model for object detection
Extract bounding boxes and confidence scores
Confidence > 0.7?
No
Yes
Skip to next frame
Draw rectangles on detected objects
Write processed frames to detection output video
Detection Process Complete
Open CCTV video?
Yes
No
Create empty heatmap array
Convert frame to PIL image Preprocess using DETR processor
Run DETR model for object detection
Extract bounding boxes
Increment heatmap intensity at detected locations
Scale heatmap values to 0-255
Convert grayscale heatmap to colored heatmap
Save heatmap as JPG file
Download detection video & heatmap image
Process Complete

Low Accuracy GPU Task:


Start Processing
Install Dependencies: pip install deep_sort_realtime pycocotools
Import Libraries: TensorFlow, OpenCV, COCO Tools, DeepSORT
Load RPC Dataset & Annotations
Initialize Data Generator: Resize & normalize images
Load Pretrained MobileNetV2: Feature extraction & classification
Compile & Train Model: Train for 1 epoch & save model
Initialize DeepSORT Tracker
Process Video Frames: Detect & track objects
Display Results: Draw bounding boxes & show frame
Process Complete

Reference Links:

CPU Task:

https://colab.research.google.com/drive/1BdK3E9TDG6YV5YagxL_BXHK86VPjljeI?usp=sharing

GPU Task:

https://colab.research.google.com/drive/1sQFhqeN5jG-sdq8lbyhkTeF9ul9qxjQ3?usp=sharing

Low Accuracy GPU task:

https://www.kaggle.com/code/shivam365/deepsort-rpc

Dataset Link:

https://www.kaggle.com/datasets/diyer22/retail-product-checkout-dataset