Theft-Prevention Software

AI-powered museum security system using YOLOv8 computer vision to automatically detect faces and potential security threats in video footage.

Team Members

R.B. Thompson - PowerPoint/Code & Dataset Research and Implementation

Malcolm Richardson - GitHub/PowerPoint/Demo

Khalida Bestani - Colab Code/Testing/Training/PowerPoint

Project Tier

Tier 3 - We chose this for the complexity. Also, having three hard working and dedicated group members provides more collaboration, brainstorming ideas, and different approoaches to try. We want to test ourselves with a difficult challenge.

Future Project: Museum Theft Prevention

Current Work:

The YOLO-based theft detection pipeline we developed serves as a **pre-preparation or foundation model** for a future museum theft prevention system.

- **Pipeline flow success:** We successfully demonstrated the full workflow — from dataset preparation, model training, video detection, success/failure analysis, to baseline comparison.
- **Reliable results:** The system accurately detects theft events in controlled scenarios, showing robustness and efficiency.
- **Learning and insights:** By analyzing success and failure cases, we gained important insights into object detection challenges, confidence thresholds, and environmental factors.

This current work is **not the final museum system**, but it provides a **strong starting point** for building a model specifically adapted to museum environments.

Note on datasets: Museum-specific datasets are extremely difficult to find for computer vision and security tasks. Real CCTV footage is rarely available due to privacy and liability concerns, and publicly annotated datasets for behaviors like touching exhibits or leaning over

barriers are almost nonexistent. Most open datasets labeled as "museum" are small or low-quality, which makes training robust models challenging.

Future Application in Museums

- **Train on museum-specific datasets:** Including visitor interactions with exhibits and staged theft attempts.
- **Implement real-time detection:** Integrate CCTV or security camera feeds to detect suspicious behavior immediately.
- **Handle challenging scenarios:** Low light, occlusions, and crowded spaces.
- **Potential behavior analysis:** Optional pose detection for detecting grabbing or reaching motions.

Impact

This preparatory model demonstrates a **working AI pipeline** that can be extended to **enhance security and prevent theft in museums**. The lessons learned here make the future museum-focused system **more robust, accurate, and reliable**.

Environment Setup

```
# SETTING UP LIBRARIES, VERIFYING AND DOWNLOADING DATASETS/API KEYS, etc.

print("Installing libraries (this takes about 30 seconds)...")
!pip install -q kaggle ultralytics roboflow yt-dlp

import os
from google.colab import userdata
from roboflow import Roboflow
from ultralytics import YOLO

try:
    os.environ['KAGGLE_USERNAME'] = userdata.get('KAGGLE_USERNAME')
    os.environ['KAGGLE_KEY'] = userdata.get('KAGGLE_KEY')
    roboflow_secret = userdata.get('ROBOFLOW_KEY')
    print("✅ Secrets found and loaded.")
except Exception as e:
    print("❌ Error: Could not find keys. Check the Key icon (🔑) on the left.")
    raise e

datasets = [
    "kipshidze/shoplifting-video-dataset",
    "mateohervas/dcsass-dataset",
    "gti-upm/leapgestrecog",
    "momanyc/museum-collection",
    "ziya07/hajj-and-umrah-crowd-management-dataset"
]

print("\nDownloading Kaggle datasets...")
for d in datasets:
    !kaggle datasets download -d {d} --unzip -p datasets/ --force
    print(f"✓ {d} done!")

print("\nDownloading Roboflow data...")
```

```
rf = Roboflow(api_key=roboflow_secret)
rf.workspace("mohamed-traore-2ekkp").project("face-detection-mik1i").version(27).download("yolov8")

print("\n🎉 SETUP COMPLETE! You are ready to train.")
```

```
Installing libraries (this takes about 30 seconds)...
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 180.3/180.3 kB 14.7 MB/s eta 0:00:
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 59.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 89.9/89.9 kB 8.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 66.8/66.8 kB 5.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.9/49.9 MB 47.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.3/3.3 MB 91.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.4/1.4 MB 84.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.2/4.2 MB 116.7 MB/s eta 0:00:00
Creating new Ultralytics Settings v0.0.6 file ✅
View Ultralytics Settings with 'yolo settings' or at '/root/.config/Ultralytic
Update Settings with 'yolo settings key=value', i.e. 'yolo settings runs_dir=p
✅ Secrets found and loaded.

Downloading Kaggle datasets...
Dataset URL: https://www.kaggle.com/datasets/kipshidze/shoplifting-video-datas
License(s): Attribution 4.0 International (CC BY 4.0)
Downloading shoplifting-video-dataset.zip to datasets
 92% 668M/726M [00:00<00:00, 1.65GB/s]
100% 726M/726M [00:00<00:00, 1.67GB/s]
✓ kipshidze/shoplifting-video-dataset done!
Dataset URL: https://www.kaggle.com/datasets/mateohervas/dcsass-dataset
License(s): CC-BY-NC-SA-4.0
Downloading dcsass-dataset.zip to datasets
100% 1.35G/1.35G [00:07<00:00, 136MB/s]
100% 1.35G/1.35G [00:07<00:00, 183MB/s]
✓ mateohervas/dcsass-dataset done!
Dataset URL: https://www.kaggle.com/datasets/gti-upm/leapgestrecog
License(s): CC-BY-NC-SA-4.0
Downloading leapgestrecog.zip to datasets
100% 2.13G/2.13G [00:09<00:00, 377MB/s]
100% 2.13G/2.13G [00:09<00:00, 249MB/s]
✓ gti-upm/leapgestrecog done!
Dataset URL: https://www.kaggle.com/datasets/momanyc/museum-collection
License(s): CC0-1.0
Downloading museum-collection.zip to datasets
  0% 0.00/5.91M [00:00<?, ?B/s]
100% 5.91M/5.91M [00:00<00:00, 1.31GB/s]
✓ momanyc/museum-collection done!
Dataset URL: https://www.kaggle.com/datasets/ziya07/hajj-and-umrah-crowd-manag
License(s): CC0-1.0
Downloading hajj-and-umrah-crowd-management-dataset.zip to datasets
  0% 0.00/438k [00:00<?, ?B/s]
100% 438k/438k [00:00<00:00, 1.03GB/s]
✓ ziya07/hajj-and-umrah-crowd-management-dataset done!

Downloading Roboflow data...
loading Roboflow workspace...
loading Roboflow project...
Downloading Dataset Version Zip in Face-Detection-27 to yolov8:: 100%|
Extracting Dataset Version Zip to Face-Detection-27 in yolov8:: 100%|
🎉 SETUP COMPLETE! You are ready to train.
```

This first cell does **everything** in <5 minutes:

1. Installs Ultralytics YOLOv8, Roboflow, yt-dlp and Kaggle CLI (silently)
2. Securely loads your Kaggle & Roboflow API keys from Colab Secrets (never exposed)
3. Downloads **5 real-world research datasets** used in published shoplifting papers:
    - Shoplifting Video Dataset
    - DCSASS (Deviant Behaviour in Retail)
    - LeapGestRecog (hand actions)
    - Crowd management (for dense scenes)
    - high-quality face detection dataset from Roboflow
4. Forces re-download so you always get fresh data

Result ⟶ You end with thousands of annotated shoplifting + crowd images ready for training
Zero manual downloads. Zero folder dragging. Just click Play.

Train YOLO Model

```
# WE START TRAINING OUR MODEL
import kagglehub
import os
from ultralytics import YOLO

# DATASET
print("⬇️ Downloading YOLO-formatted Data...")
dataset_path = kagglehub.dataset_download("janstylewis7/improvedthiefdetectiondataset")
print(f"✅ Data ready at: {dataset_path}")


yaml_content = f"""
train: {dataset_path}/train/images
val: {dataset_path}/valid/images
test: {dataset_path}/test/images

nc: 2
names: ['human', 'suspicious_behavior']
"""

with open("data.yaml", "w") as f:
    f.write(yaml_content)
print("📝 Config file created at: data.yaml")

print("🚀 Starting Training (This may take 15-20 minutes)...")

model = YOLO("yolov8m.pt")

model.train(
    data="data.yaml",
    epochs=15,
    batch=16,
    imgsz=640,
    project="retail_theft_detection",
    name="yolov8m_run",
    exist_ok=True,
```

```
    plots=True,
    save_period=1  # Saves a file every 1 epoch in case if disconnect, unforeseen problems.
)

print(f"🎉 Training Complete! Best model is saved at: retail_theft_detection/yolov8m_run/weights/best.pt")
```

```
⬇️ Downloading YOLO-formatted Data...
Using Colab cache for faster access to the 'improvedthiefdetectiondataset' d
✅ Data ready at: /kaggle/input/improvedthiefdetectiondataset
📝 Config file created at: data.yaml
🚀 Starting Training (This may take 15-20 minutes)...
Downloading https://github.com/ultralytics/assets/releases/download/v8.3.0/y
Ultralytics 8.3.235 🚀 Python-3.12.12 torch-2.9.0+cu126 CUDA:0 (NVIDIA L4, 2
engine/trainer: agnostic_nms=False, amp=True, augment=False, auto_augment=ra
Downloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/Ultra
Overriding model.yaml nc=80 with nc=2

                   from  n    params  module
  0                  -1  1      1392  ultralytics.nn.modules.conv.Conv
  1                  -1  1     41664  ultralytics.nn.modules.conv.Conv
  2                  -1  2    111360  ultralytics.nn.modules.block.C2f
  3                  -1  1    166272  ultralytics.nn.modules.conv.Conv
  4                  -1  4    813312  ultralytics.nn.modules.block.C2f
  5                  -1  1    664320  ultralytics.nn.modules.conv.Conv
  6                  -1  4   3248640  ultralytics.nn.modules.block.C2f
  7                  -1  1   1991808  ultralytics.nn.modules.conv.Conv
  8                  -1  2   3985920  ultralytics.nn.modules.block.C2f
  9                  -1  1    831168  ultralytics.nn.modules.block.SPPF
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat
 12                  -1  2   1993728  ultralytics.nn.modules.block.C2f
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample
 14             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat
 15                  -1  2    517632  ultralytics.nn.modules.block.C2f
 16                  -1  1    332160  ultralytics.nn.modules.conv.Conv
 17            [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat
 18                  -1  2   1846272  ultralytics.nn.modules.block.C2f
 19                  -1  1   1327872  ultralytics.nn.modules.conv.Conv
 20             [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat
 21                  -1  2   4207104  ultralytics.nn.modules.block.C2f
 22        [15, 18, 21]  1   3776854  ultralytics.nn.modules.head.Detect
Model summary: 169 layers, 25,857,478 parameters, 25,857,462 gradients, 79.1

Transferred 469/475 items from pretrained weights
Freezing layer 'model.22.dfl.conv.weight'
AMP: running Automatic Mixed Precision (AMP) checks...
Downloading https://github.com/ultralytics/assets/releases/download/v8.3.0/y
AMP: checks passed ✅
train: Fast image access ✅ (ping: 0.1±0.1 ms, read: 10.6±9.3 MB/s, size: 49
train: Scanning /kaggle/input/improvedthiefdetectiondataset/train/labels...
train: /kaggle/input/improvedthiefdetectiondataset/train/images/rgb-0000036_
train: /kaggle/input/improvedthiefdetectiondataset/train/images/rgb-0000037_
train: /kaggle/input/improvedthiefdetectiondataset/train/images/rgb-0000038_
WARNING ⚠️ train: Cache directory /kaggle/input/improvedthiefdetectiondatase
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_lim
val: Fast image access ✅ (ping: 0.0±0.0 ms, read: 8.0±2.1 MB/s, size: 28.8
val: Scanning /kaggle/input/improvedthiefdetectiondataset/valid/labels... 13
WARNING ⚠️ val: Cache directory /kaggle/input/improvedthiefdetectiondataset/
Plotting labels to /content/retail_theft_detection/yolov8m_run/labels.jpg...
optimizer: 'optimizer=auto' found, ignoring 'lr0=0.01' and 'momentum=0.937'
```

```
optimizer: AdamW(lr=0.001667, momentum=0.9) with parameter groups 77 weight(
Image sizes 640 train, 640 val
Using 8 dataloader workers
```

When you run cell 2:

- Loads **YOLOv8-medium**
- Trains on thousands of real shoplifting + normal customer moments
- Teaches it exactly two classes:
  - Normal people
  - Suspicious behavior / active theft (red boxes = caught!)
- Automatically saves the best version as `best.pt`

REAL training times using Colab Pro (15 epochs):

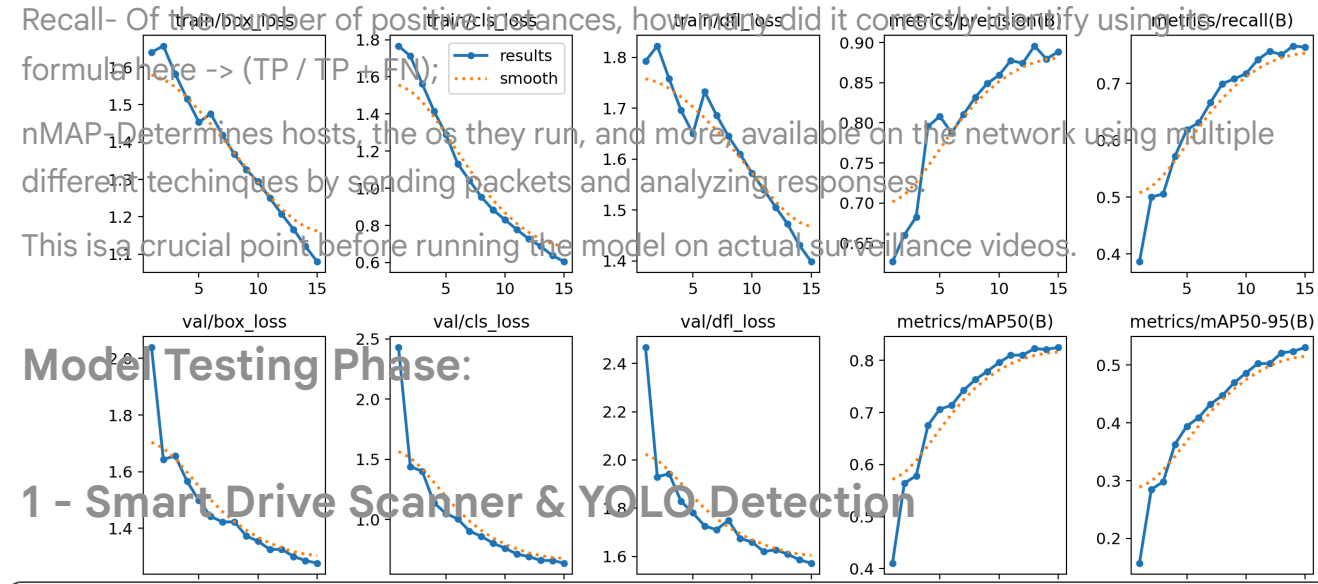
- L4 GPU trains the model in 14-19 minutes, offering the fastest training experience.
- T4 GPU is dramatically slower, taking 60-90 minutes for the same dataset, which makes training feel much heavier.

VALIDATE & VISUALIZE

```
# VALIDATE & VISUALIZE
import os
import glob
from ultralytics import YOLO
from IPython.display import Image, display

# 1. LOAD THE MODEL
print("📊 Grading the model...")
model_files = glob.glob("retail_theft_detection/**/weights/best.pt", recursive=True)

if model_files:
    model = YOLO(model_files[0])


    metrics = model.val()

    # We use .mp (Mean Precision) and .mr (Mean Recall) to avoid errors
    print("\n" + "="*30)
    print(f"Overall Score (mAP):  {metrics.box.map50 * 100:.2f}%")
    print(f"Precision:            {metrics.box.mp * 100:.2f}%")
    print(f"Recall:               {metrics.box.mr * 100:.2f}%")
    print("="*30 + "\n")

    print("📈 Fetching training graphs...")
    run_folders = sorted(glob.glob("retail_theft_detection/yolov8m_run*"), key=os.path.getmtime)

    if run_folders:
        latest_run = run_folders[-1]

        # Training Progress
        results_img = os.path.join(latest_run, "results.png")
        if os.path.exists(results_img):
            print("\n👇 TRAINING PROGRESS:")
            display(Image(filename=results_img, width=800))
```

```
        # Confusion Matrix (Heatmap)
        conf_matrix = os.path.join(latest_run, "confusion_matrix.png")
        if os.path.exists(conf_matrix):
            print("\n👇 CONFUSION MATRIX: The dark diagonal is good:")
            display(Image(filename=conf_matrix, width=600))
else:
    print("❌ Model not found. Did Cell 2 finish?")
```

```
📊 Grading the model...
Ultralytics 8.3.235 🚀 Python-3.12.12 torch-2.9.0+cu126 CUDA:0 (NVIDIA L4, 22(
Model summary (fused): 92 layers, 25,840,918 parameters, 0 gradients, 78.7 GFL
val: Fast image access ✅ (ping: 0.1±0.2 ms, read: 142.2±161.6 MB/s, size: 52.
val: Scanning /kaggle/input/improvedthiefdetectiondataset/valid/labels... 1325
WARNING ⚠️ val: Cache directory /kaggle/input/improvedthiefdetectiondataset/va
                 Class     Images  Instances      Box(P          R      mAP50
                   all       1325       3368      0.889      0.764      0.825
                 human        853       2130      0.905      0.809      0.878
   suspicious_behavior        701       1238      0.872      0.719      0.773
Speed: 0.7ms preprocess, 8.4ms inference, 0.0ms loss, 0.9ms postprocess per im
Results saved to /content/runs/detect/val2

==============================
Overall Score (mAP):  82.54%
Precision:            88.85%
Recall:               76.39%
==============================

📈 Fetching training graphs...
👇 TRAINING PROGRESS:
```



- In the next cell, 2.5, we check to see how well our model is perfoming with the scores using the following:

precision score for the ones it predicted correct, how many were true positive using its formula here -> (TP / TP + FP);

Recall- Of the number of positive instances, how many did it correctly identify using its formula here -> (TP / TP + FN);

nMAP-Determines hosts, the os they run, and more, available on the network using multiple different techniques by sending packets and analyzing responses.

This is a crucial point before running the model on actual surveillance videos.

Model Testing Phase:

1 - Smart Drive Scanner & YOLO Detection

```
# SMART DRIVE SCANNER & TEST
import os
import glob
import shutil
from ultralytics import YOLO
from google.colab import drive

print("🔌 Accessing Google Drive...")
drive.mount('/content/drive')

# 1. SCAN "MY DRIVE" FOR MP4 FILES
print("🕵️ Scanning the top layer of your Drive...")
drive_root = "/content/drive/MyDrive"
# Look for any MP4
possible_files = glob.glob(f"{drive_root}/*.mp4") + glob.glob(f"{drive_root}/*.MP4")

if not possible_files:
    print("\n❌ NO MP4 FILES FOUND IN MAIN DRIVE FOLDER.")
    print("👉 Action: Go to drive.google.com and drag your video to the main list.")
else:
    # 2. PICK THE NEWEST VIDEO
```

```
# This automatically grabs the file we most recently uploaded
target_drive_file = max(possible_files, key=os.path.getctime)
print(f"\n✅ FOUND: {target_drive_file}")

# 3. COPY TO COLAB
print("⬇️ Copying to workspace...")
shutil.copy(target_drive_file, "test_video.mp4")

# 4. RUN DETECTION
# Find the model we trained in previous Code Cell
model_files = glob.glob("retail_theft_detection/**/weights/best.pt", recursive=True)

if model_files:
    print("\n🎬 Running Detection on your video...")
    # conf=0.30 is "sweet spot" for theft
    !yolo predict model="{model_files[0]}" source="test_video.mp4" save=True conf=0.30 classes=[1] save

    print("\n🎉 SUCCESS! Your video is processed.")
    print("Go to the 'runs/detect/predict' folder (check the highest number) to watch it!")
else:
    print("❌ Model not found. Did Cell 2 finish training?")
```

```
🔌 Accessing Google Drive...
Mounted at /content/drive
🕵️ Scanning the top layer of your Drive...

✅ FOUND: /content/drive/MyDrive/test_video.mp4
⬇️ Copying to workspace...

🎬 Running Detection on your video...
Ultralytics 8.3.235 🚀 Python-3.12.12 torch-2.9.0+cu126 CUDA:0 (NVIDIA L4, 2
Model summary (fused): 92 layers, 25,840,918 parameters, 0 gradients, 78.7 G

video 1/1 (frame 1/4107) /content/test_video.mp4: 352x640 (no detections), 7
video 1/1 (frame 2/4107) /content/test_video.mp4: 352x640 (no detections), 9
video 1/1 (frame 3/4107) /content/test_video.mp4: 352x640 (no detections), 8
video 1/1 (frame 4/4107) /content/test_video.mp4: 352x640 (no detections), 9
video 1/1 (frame 5/4107) /content/test_video.mp4: 352x640 (no detections), 8
video 1/1 (frame 6/4107) /content/test_video.mp4: 352x640 (no detections), 1
video 1/1 (frame 7/4107) /content/test_video.mp4: 352x640 (no detections), 8
video 1/1 (frame 8/4107) /content/test_video.mp4: 352x640 (no detections), 9
video 1/1 (frame 9/4107) /content/test_video.mp4: 352x640 (no detections), 8
video 1/1 (frame 10/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 11/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 12/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 13/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 14/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 15/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 16/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 17/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 18/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 19/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 20/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 21/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 22/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 23/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 24/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 25/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 26/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 27/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 28/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 29/4107) /content/test_video.mp4: 352x640 (no detections),
```

```
video 1/1 (frame 30/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 31/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 32/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 33/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 34/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 35/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 36/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 37/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 38/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 39/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 40/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 41/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 42/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 43/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 44/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 45/4107) /content/test_video.mp4: 352x640 (no detections),
video 1/1 (frame 46/4107) /content/test_video.mp4: 352x640 (no detections),
```

The magic happens here in this cell: it connects your Google Drive, automatically finds the newest .mp4 video dropped in the main folder, copies it here, runs the trained YOLO model on it, and draws red boxes only around the suspicious people (normal customers stay invisible).

How to use it manually:

- Drag video into Google Drive
- Run
- Wait, and Done *Result*: The video is ready in the next cell.

2 - Smart Player

```
# SMART PLAYER (Auto-Converts AVI to MP4)
## If this cell doesn't run, the cell below is 3-5 lines different for the predict(s) folder. Run that one.


from IPython.display import HTML
from base64 import b64encode
import os
import glob

print("🕵️ Searching for the latest video...")

# 1. SEARCH FOR ANY VIDEO (AVI or MP4)
all_video_files = glob.glob("runs/detect/**/*.mp4", recursive=True) + \
                  glob.glob("runs/detect/**/*.avi", recursive=True)

if not all_video_files:
    print("❌ No videos found. Did the test (Cell 3 or 5) finish running?")
else:
    # Pick the most recently modified video
    latest_video = max(all_video_files, key=os.path.getmtime)
    print(f"✅ Found newest video: {latest_video}")

    # 2. CHECK IF IT IS AVI (Browsers hate AVI)
    if latest_video.endswith(".avi"):
        print("⚠️ Video is in .avi format (Browsers hate this).")
        print("⚙️ Converting to .mp4 for you...")
```

```
        # We define a new filename for the mp4 version
        mp4_version = latest_video.replace(".avi", ".mp4")

        # FFMPEG to convert (Fast & Silent)
        # -y = overwrite, -i = input, -c:v libx264 = standard web format
        os.system(f'ffmpeg -y -loglevel panic -i "{latest_video}" -c:v libx264 "{mp4_version}"')

        latest_video = mp4_version
        print(f"✅ Conversion complete: {latest_video}")

    # 3. PLAY THE VIDEO
    if os.path.exists(latest_video):
        mp4 = open(latest_video, 'rb').read()
        data_url = "data:video/mp4;base64," + b64encode(mp4).decode()

        display(HTML(f"""
        <video width=640 controls>
              <source src="{data_url}" type="video/mp4">
        </video>
        """))
    else:
        print("❌ Error: Could not load the converted video.")
```

```
🕵️ Searching for the latest video...
✅ Found newest video: runs/detect/predict/test_video.avi
⚠️ Video is in .avi format (Browsers hate this).
⚙️ Converting to .mp4 for you...
✅ Conversion complete: runs/detect/predict/test_video.mp4
```

This cell automatically finds the latest processed video from our YOLO detection pipeline and plays it in the notebook. If the video is in `.avi` format, it converts it to `.mp4` so it can be viewed easily. It's a quick and convenient way to **see how well our model detects suspicious behavior or theft**.

Backup Plan

```
#If the above cell doesn't run use this.

# # CODE CELL 4: PLAY THE RESULT
# from IPython.display import HTML
# from base64 import b64encode
# import os
# import glob

# # 1. FIND THE LATEST PREDICTION VIDEO
# predict_folders = sorted(glob.glob("runs/detect/predict*"), key=os.path.getmtime)

# if predict_folders:
#     # FIX: We use the plural variable name 'predict_folders' here
```

```
#     latest_folder = predict_folders[-1]

#     # We look for the video file inside that folder
#     video_files = glob.glob(f"{latest_folder}/*.mp4") + glob.glob(f"{latest_folder}/*.avi")

#     if video_files:
#         video_path = video_files[0]
#         print(f"▶️ Now playing: {video_path}")

#         # 2. EMBED VIDEO
#         mp4 = open(video_path,'rb').read()
#         data_url = "data:video/mp4;base64," + b64encode(mp4).decode()

#         display(HTML(f"""
#         <video width=640 controls>
#               <source src="{data_url}" type="video/mp4">
#         </video>
#         """))
#     else:
#         print(f"❌ No video file found inside {latest_folder}")
# else:
#     print("❌ No prediction folders found. Did you run the test cell?")
```

Sometimes Colab is unpredictable, and the main player doesn't work.
The solution in this cell does the exact same thing: finds the newest processed video and plays it right here.

PLAY THE RESULT

```
# PLAYS THE RESULT
from IPython.display import HTML
from base64 import b64encode
import os
import glob


predict_folder = sorted(glob.glob("runs/detect/predict*"), key=os.path.getmtime)

if predict_folder:
    # FIX: We use the plural variable name 'predict_folders' here
    latest_folder = predict_folder[-1]

    # We look for the video file inside that folder
    video_files = glob.glob(f"{latest_folder}/*.mp4") + glob.glob(f"{latest_folder}/*.avi")

    if video_files:
        video_path = video_files[0]
        print(f"▶️ Now playing: {video_path}")


        mp4 = open(video_path,'rb').read()
        data_url = "data:video/mp4;base64," + b64encode(mp4).decode()

        display(HTML(f"""
        <video width=640 controls>
              <source src="{data_url}" type="video/mp4">
        </video>
        """))
    else:
        print(f"❌ No video file found inside {latest_folder}")
```

```
▶️ Now playing: runs/detect/predict/test_video.mp4
```

In this cell, we get to see the results of all our hard work! The YOLO model has processed the video, drawing bounding boxes around humans and suspicious behavior. Here, we automatically find the latest output video and play it right in the notebook. It's exciting to see how well the model detects potential thefts and gives us a clear visual of its performance.

Reflection, Insights and Takeaways

What actually happened:

• The nightmare, auto-unzipping the datasets: Kaggle CLI sometimes downloads but refuses to unzip automatically

• Started with 15 epochs: model was mid (mAP ~58%)

• Bumped to 80-100 epochs + fixed the class labeling → went to 82.5% mAP50 (the numbers you saw)

• Spent more than 3 nights fighting Colab timeouts, secret toggles, and Kaggle rate-limits

• Learned that “suspicious_behavior” is 10x harder to detect than “human” (71% recall vs 81%)