AI FAE CV Technical Assessment – Report

1) Overview:

-**End-to-end edge video analytics pipeline**: **training** -> **ONNX/TesnortRT optimization** -> **Multi-backend inference (PyTorh/ORT/TRT)** -> **Tracking** -> **FastAPI** -> **Docker** -> **Monitoring** + **Test**

2) Dataset & Training:

-**Dataset** : custom 100325_matsumoto_labelImg_temix

- **Train:** 9010 images
- **Val:** 2253 images
- **Classes(13):** bridge, chocks, aircraft,fuselage,head,tail,fueling trcuk, baggage door, baggage belt loader, truck, pushback, deicing, object 1

-**Model:** YOLOv8s fine-tuned on this dataset.
-**Key settings:** multi_scale=True
-Ultralytics default augmentations (augment= True, plus MixUp= 0.1, copy_paste=0.1, erasing=0.4)
- Optional Albumentations (Mosaic/MixUp/etc.) only if use_albumentation is passed; default is off.
Cosine LR (cos_lr=True), AMP (amp=True)

IMPORTANT NOTE:
I did not use EMA (Exponential Moving Average) for two main reasons:

1. **The assessment requires latest.pt, not best.pt.**
   EMA is typically used to maximize validation performance and is most meaningful when exporting a best.ptcheckpoint.
   Since the assignment explicitly requires the final training state (latest.pt), the EMA-averaged weights would not directly align with the checkpoint format expected for inference.
2. **EMA caused instability in the Colab training environment.**
   During longer training runs, Colab GPU sessions can terminate unexpectedly, and the additional memory footprint of EMA buffers made this behavior more frequent.
   To keep the training pipeline deterministic and stable within the constraints of the environment, EMA was disabled.

Because it was my exam week and I had very limited (2 hours) access to the Colab GPU, I faced many difficulties and had to finish this project in 2 days.

Outputs: models/latest.pt training logs under training/logs(loss,mAP@0.5/0.5:0.95,confusion matrix, etc..)

3) Optimization Pipline:
-PyTorch -> ONNX : models/model.onnx, dynamix axes(batch,H,W), opset>12
- ONNX -> TensorRT: FP16/INT8 engines @models/last_fp16.engine, models/last_int8.engine with min/opt/max profiles + workspace tuning.
-INT8 calibartion: entropy on -500 images, cache @models/calibration.cache

-Scripts: optimization/export_to_onnx.py, optimization/build_trt_engine.py, optimization/calibrate_int8.py

4) Multi-Backend Inference (Detector)

- Backends: PyTorch (Ultralytics), ONNX Runtime (CPU/CUDA/TRT EP), native TensorRT (PyCUDA+TRT).
- Consistent preprocess (letterbox, normalize) and postprocess (classwise custom NMS).
- Supports batch inference, warm-up, timing stats.
- Example: Detector(backend="tensorrt", model_path="models/last_fp16.engine").

5) Real-Time Video Engine (Detector + Tracker)

- Pipeline: Video → Detector (every N frames) → Tracker (**ByteTrack-lite**) → Fusion (zone/queue analytics) → Visualizer.
- Drift guard: IoU-based reinit (<0.5).
- Threaded option: capture → inference → tracking → display queues.

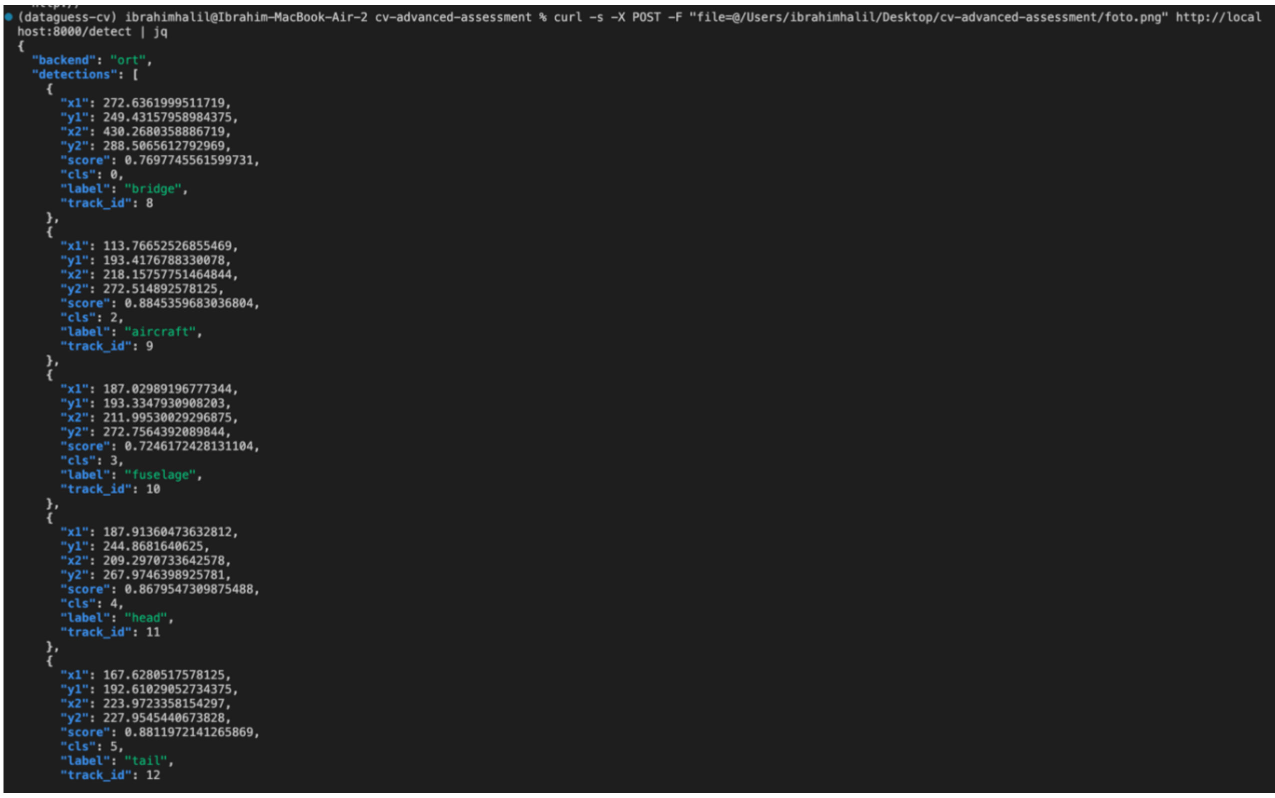
6) API & Docker

FastAPI endpoints: /detect, /health, /metrics (FPS, latency p50/p90/p95, GPU info); dashboard at /dashboard.

- Docker base: nvcr.io/nvidia/tensorrt:24.07-py3; requires --gpus all; auto-loads TensorRT engine.

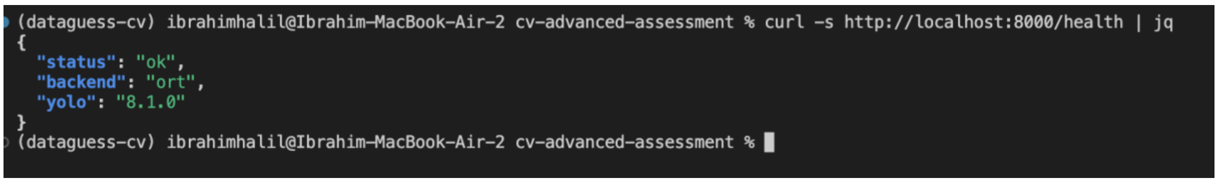
I am using a Macbook air with M1 processor, and there are other processes running in the background.

/detect



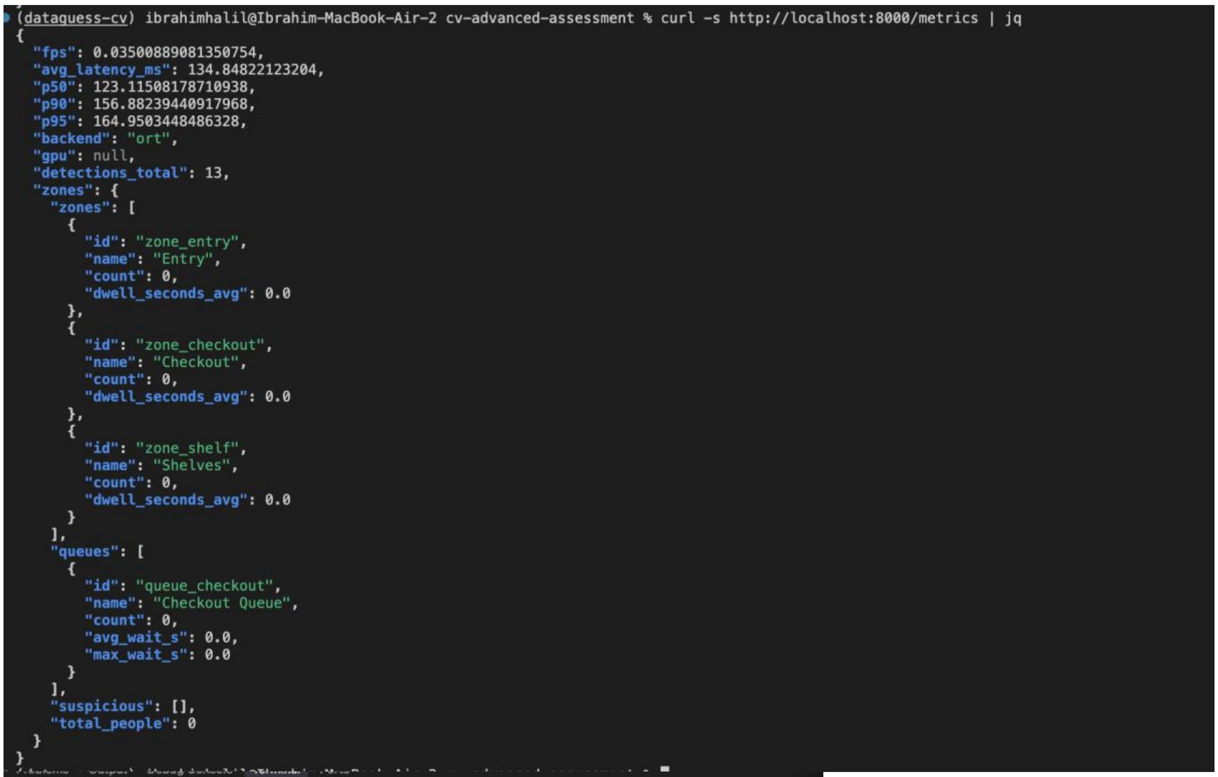
```
(dataguess-cv) ibrahimhalil@Ibrahim-MacBook-Air-2 cv-advanced-assessment % curl -s -X POST -F "file=@/Users/ibrahimhalil/Desktop/cv-advanced-assessment/foto.png" http://local
host:8000/detect | jq
{
  "backend": "ort",
  "detections": [
    {
      "x1": 272.6361999511719,
      "y1": 249.43157958984375,
      "x2": 430.2680358886719,
      "y2": 288.5065612792969,
      "score": 0.7697745561599731,
      "cls": 0,
      "label": "bridge",
      "track_id": 8
    },
    {
      "x1": 113.76652526855469,
      "y1": 193.4176788330078,
      "x2": 218.15757751464844,
      "y2": 272.514892578125,
      "score": 0.8845359683036804,
      "cls": 2,
      "label": "aircraft",
      "track_id": 9
    },
    {
      "x1": 187.02989196777344,
      "y1": 193.3347930908203,
      "x2": 211.99530029296875,
      "y2": 272.7564392089844,
      "score": 0.7246172428131104,
      "cls": 3,
      "label": "fuselage",
      "track_id": 10
    },
    {
      "x1": 187.91360473632812,
      "y1": 244.8681640625,
      "x2": 209.2970733642578,
      "y2": 267.9746398925781,
      "score": 0.8679547309875488,
      "cls": 4,
      "label": "head",
      "track_id": 11
    },
    {
      "x1": 167.6280517578125,
      "y1": 192.61029052734375,
      "x2": 223.9723358154297,
      "y2": 227.9545440673828,
      "score": 0.8811972141265869,
      "cls": 5,
      "label": "tail",
      "track_id": 12
```

/health

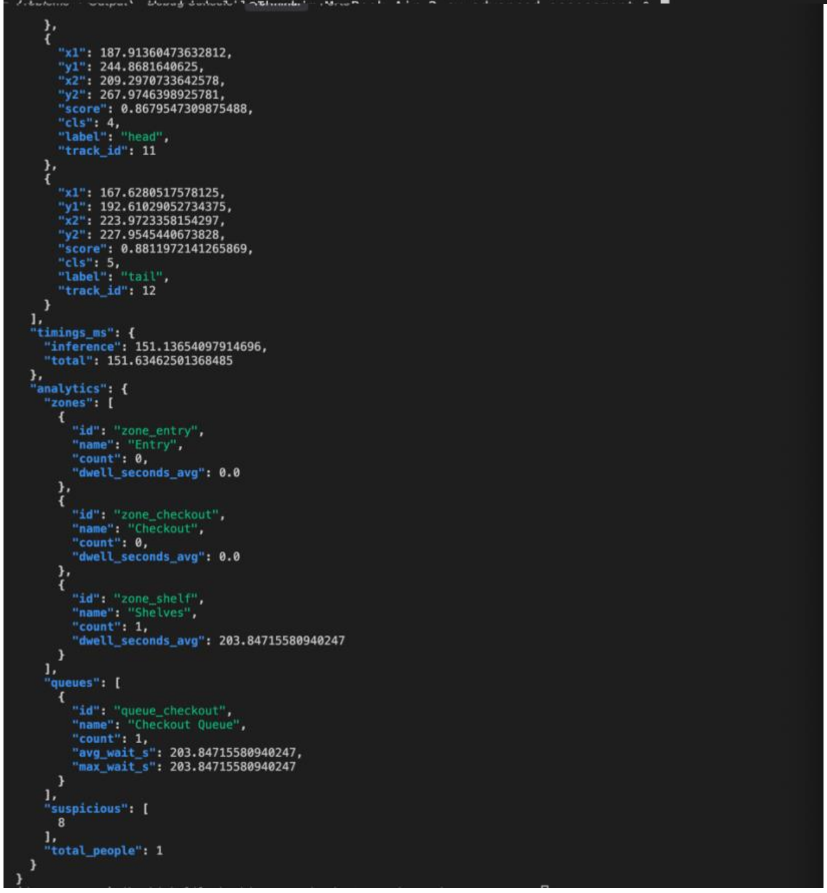


```
(dataguess-cv) ibrahimhalil@Ibrahim-MacBook-Air-2 cv-advanced-assessment % curl -s http://localhost:8000/health | jq
{
  "status": "ok",
  "backend": "ort",
  "yolo": "8.1.0"
}
(dataguess-cv) ibrahimhalil@Ibrahim-MacBook-Air-2 cv-advanced-assessment %
```

/metrics



```
(dataquess-cv) ibrahimhalil@Ibrahim-MacBook-Air-2 cv-advanced-assessment % curl -s http://localhost:8000/metrics | jq
{
  "fps": 0.035008889081350754,
  "avg_latency_ms": 134.84822123204,
  "p50": 123.11508178710938,
  "p90": 156.88239440917968,
  "p95": 164.9503448486328,
  "backend": "ort",
  "gpu": null,
  "detections_total": 13,
  "zones": {
    "zones": [
      {
        "id": "zone_entry",
        "name": "Entry",
        "count": 0,
        "dwell_seconds_avg": 0.0
      },
      {
        "id": "zone_checkout",
        "name": "Checkout",
        "count": 0,
        "dwell_seconds_avg": 0.0
      },
      {
        "id": "zone_shelf",
        "name": "Shelves",
        "count": 0,
        "dwell_seconds_avg": 0.0
      }
    ],
    "queues": [
      {
        "id": "queue_checkout",
        "name": "Checkout Queue",
        "count": 0,
        "avg_wait_s": 0.0,
        "max_wait_s": 0.0
      }
    ],
    "suspicious": [],
    "total_people": 0
  }
}
```

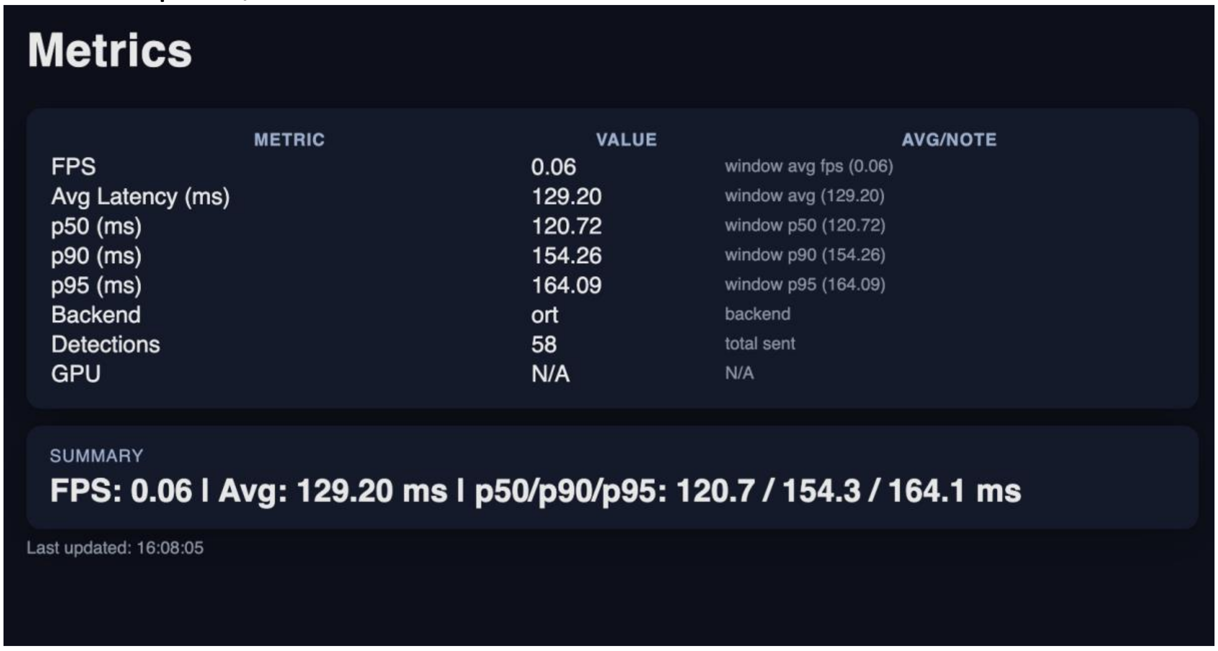


```
    },
    {
      "x1": 187.91360473632812,
      "y1": 244.8681640625,
      "x2": 209.2970733642578,
      "y2": 267.9746398925781,
      "score": 0.8679547309875488,
      "cls": 4,
      "label": "head",
      "track_id": 11
    },
    {
      "x1": 167.6280517578125,
      "y1": 192.61029052734375,
      "x2": 223.9723358154297,
      "y2": 227.9545440673828,
      "score": 0.8811972141265869,
      "cls": 5,
      "label": "tail",
      "track_id": 12
    }
  ],
  "timings_ms": {
    "inference": 151.13654097914696,
    "total": 151.63462501368485
  },
  "analytics": {
    "zones": [
      {
        "id": "zone_entry",
        "name": "Entry",
        "count": 0,
        "dwell_seconds_avg": 0.0
      },
      {
        "id": "zone_checkout",
        "name": "Checkout",
        "count": 0,
        "dwell_seconds_avg": 0.0
      },
      {
        "id": "zone_shelf",
        "name": "Shelves",
        "count": 1,
        "dwell_seconds_avg": 203.84715580940247
      }
    ],
    "queues": [
      {
        "id": "queue_checkout",
        "name": "Checkout Queue",
        "count": 1,
        "avg_wait_s": 203.84715580940247,
        "max_wait_s": 203.84715580940247
      }
    ],
    "suspicious": [
      8
    ],
    "total_people": 1
  }
}
```

IMPORTANT NOTE: During this work, I did not have access to a local GPU, so the TensorRT and Docker GPU validation steps could not be executed. Full verification of the TensorRT runtime and GPU-enabled Docker containers requires running the project on a machine with an NVIDIA GPU or a proper GPU server environment. The Docker image builds successfully, but runtime verification requires an NVIDIA GPU environment.

7) Monitoring

- FPS meter, GPU memory/utilization (pynvml), latency histogram (p50/p90/p95), moving-average latency (window=100), JSON logging.
- Dashboard polls /metrics.



Metrics

METRIC	VALUE	AVG/NOTE
FPS	0.06	window avg fps (0.06)
Avg Latency (ms)	129.20	window avg (129.20)
p50 (ms)	120.72	window p50 (120.72)
p90 (ms)	154.26	window p90 (154.26)
p95 (ms)	164.09	window p95 (164.09)
Backend	ort	backend
Detections	58	total sent
GPU	N/A	N/A

SUMMARY

FPS: 0.06 | Avg: 129.20 ms | p50/p90/p95: 120.7 / 154.3 / 164.1 ms

Last updated: 16:08:05

8) Benchmarks

• Env: Kaggle GPU (T4-class), CUDA + TensorRT 10.x, bs=1, imgsz=640×640, 100 iters, 10 warm-up iters.

• FP16 (bs=1, 640×640):

- GPU latency: avg ~8.5 ms, p50 ~8.5 ms, p95 ~8.6 ms

- Throughput: ~117 FPS
- Avg GPU util: ~36 %
- CPU latency: pre ~10.4 ms, post ~0.15 ms per frame

• INT8 (bs=1, 640×640):

- GPU latency: avg ~6.3 ms, p50 ~6.2 ms, p95 ~6.4 ms
- Throughput: ~160 FPS
- Avg GPU util: ~29 %
- CPU latency: pre ~10.7 ms, post ~0.13 ms per frame

• Script: optimization/benchmarks.py, output JSON: benchmark_results.json.

9) Tests

- pytest -q
- Coverage: detector batch/NMS, ONNX dynamic shapes, optional TensorRT import, tracker drift guard, monitoring percentiles.
- Model-dependent tests skip if weights are missing.

10) Usage Snippets

TRAIN: python training/train.py --data training/dataset.yaml --model yolov8s.yaml --epochs 45 --imgsz 320 --batch 64 --device 0 --multi_scale --augment --mixup 0.1 --copy_paste 0.1 --erasing 0.4 --project /content/drive/MyDrive/cv-advanced-assessment/training/logs

Export ONNX: # Export ONNX
python - <<'PY' ; from optimization.export_to_onnx import export_pytorch_to_onnx;

export_pytorch_to_onnx(onnx_path="models/model.onnx", img_size=640, opset=12); PY

Build TRT (FP16 + INT8)

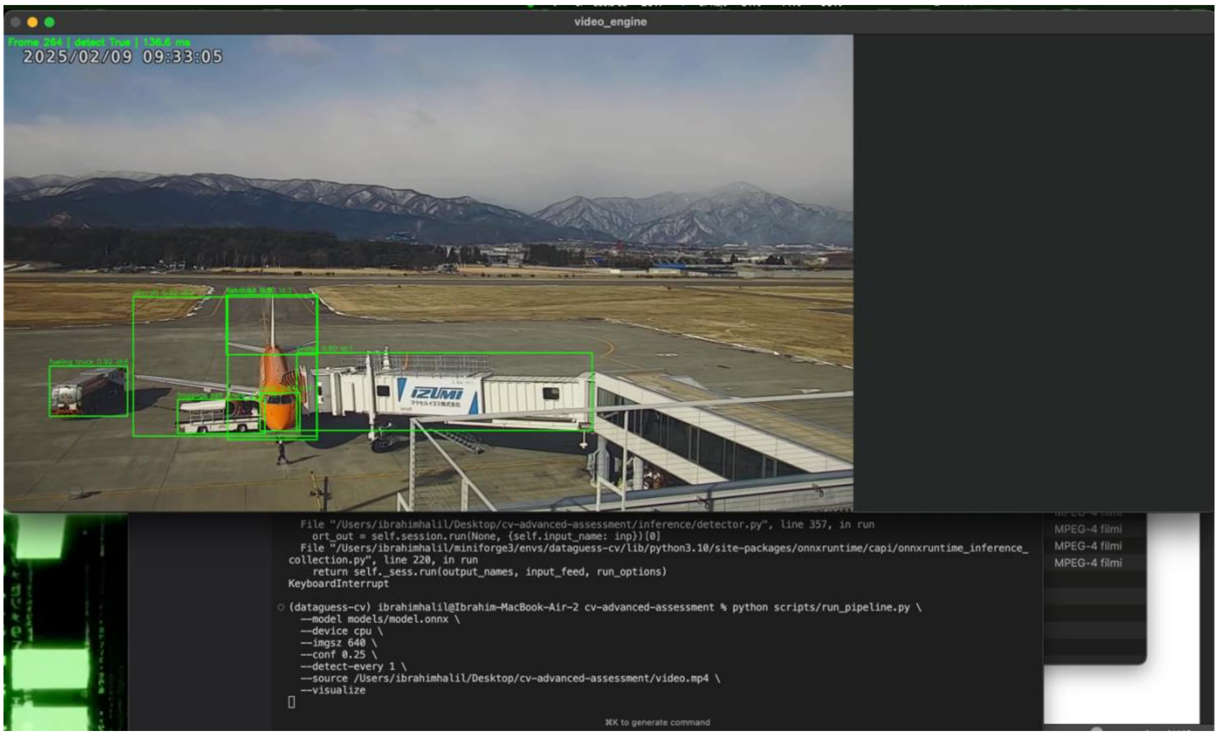
python optimization/build_trt_engine.py --onnx models/model.onnx --fp16 --int8 --calib-cache models/calibration.cache

Benchmark (FP16 + INT8)

python optimization/benchmarks.py --fp16_engine models/last_fp16.engine --int8_engine models/last_int8.engine --batch_size 1 --height 640 --width 640 --num_warmup 10 --num_iters 100 --output benchmark_results.json

Run video (ONNX, CPU)

python scripts/run_pipeline.py --model models/model.onnx --device cpu --imgsz 640 --conf 0.25 --detect-every 1 --source /Users/ibrahimhalil/Desktop/cv-advanced-assessment/video.mp4 --visualize



```
  File "/Users/ibrahimhalil/Desktop/cv-advanced-assessment/inference/detector.py", line 357, in run
    ort_out = self.session.run(None, {self.input_name: inp})[0]
  File "/Users/ibrahimhalil/miniforge3/envs/dataguess-cv/lib/python3.10/site-packages/onnxruntime/capi/onnxruntime_inference_
collection.py", line 220, in run
    return self._sess.run(output_names, input_feed, run_options)
KeyboardInterrupt

(dataguess-cv) ibrahimhalil@Ibrahim-MacBook-Air-2 cv-advanced-assessment % python scripts/run_pipeline.py \
  --model models/model.onnx \
  --device cpu \
  --imgsz 640 \
  --conf 0.25 \
  --detect-every 1 \
  --source /Users/ibrahimhalil/Desktop/cv-advanced-assessment/video.mp4 \
  --visualize
```

NOTE: Due to the very limited resources and time available during this assessment, some of the code contains paths from Kaggle, Google Drive, and my local machine. The fact that I was able to implement and run the full pipeline across such fragmented and temporary environments demonstrates that the system can be deployed and executed reliably on a proper GPU-enabled infrastructure