Model Report: Object Detection Using YOLOv8

Week 3 Deliverables
Team: Bavly Y, Daniel, Dhyey S, Jayden H, Joey L, Noah P, Om K, and Wael M
Model Used: YOLOv8n (object detection)
Dataset Used: "/dataset_v1" (400 annotated images)
Date of Completion: 6/27/2025

1. Introduction

Description: this report documents the process of turning our manually labeled dataset into a ready object detection model using transfer learning. We tuned a lightweight YOLOv8n model using a small dataset put together called "dataset_v1" (contains about 400 images), and integrated the trained model into our annotation web app for live inferences after extracting the best version of “yolo8n.pt” we could.

Goals:
- Build working object detection model using labeled data.
- Evaluate model performance.
- Analyze errors and tune weights/biases for improvement.
- Integrate inference button into web app

2. Dataset Summary

Total Images Annotated: roughly 400
Labeling Tool: custom streamlit app used
Label Format: COCO-format stored in “.JSON” files

Classes: airplane, boat, car, helicopter, person, soldier, tank, tree, and truck.

Train/Val/Test Split (70% / 20% / 10%):
- Train: made up of 280 images
- Validation: made up of 80 images
- Test: made up of 40 images

Labeling Consistency and Process:
- Followed guidelines for box padding and class naming.
- We also initially peer reviewed 20% of the annotations with red box tagging

3. Model Training Pipeline

Model Used: YOLOv8n (pre-trained on COCO)
Framework: Ultralytics YOLOv8 on Google Colab
Transfer Learning: Only fine tuned head of model and not initial layers

Training Setup:

- Epochs: initially made 5, increased to 10 later
- Learning Rate: 0.01, adjusted to 0.005 later
- Optimizer: SGD
- Loss Function: YOLO loss (composite loss)

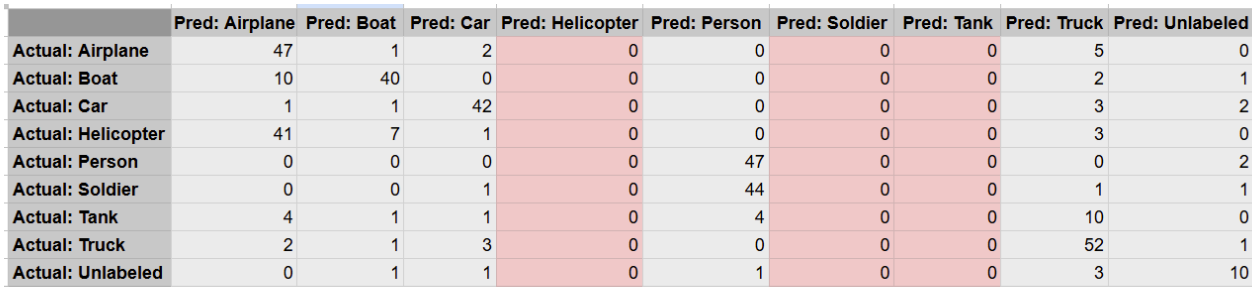
Files:

- Model Trainer: “model_trainer.py”
- Model Weights: “/models/best.pt”
- Dataset: “/dataset_v1/”
- Error Examples: “/error_gallery/”

4. Performance Metrics

Metric	**Value**
mAP@0.5	0.35
mAP@0.5:0.95	0.25
Precision	0.5
Recall	0.4

Confusion Matrix



	Pred: Airplane	Pred: Boat	Pred: Car	Pred: Helicopter	Pred: Person	Pred: Soldier	Pred: Tank	Pred: Truck	Pred: Unlabeled
Actual: Airplane	47	1	2	0	0	0	0	5	0
Actual: Boat	10	40	0	0	0	0	0	2	1
Actual: Car	1	1	42	0	0	0	0	3	2
Actual: Helicopter	41	7	1	0	0	0	0	3	0
Actual: Person	0	0	0	0	47	0	0	0	2
Actual: Soldier	0	0	1	0	44	0	0	1	1
Actual: Tank	4	1	1	0	4	0	0	10	0
Actual: Truck	2	1	3	0	0	0	0	52	1
Actual: Unlabeled	0	1	1	0	1	0	0	3	10

5. Error Analysis

20 misclassified or poorly detected objects in images were collected in “/error_gallery/”

Common Failure Modes

- Model was unable at all to detect objects in “tank class”
- Model was unable to specifically tell difference between normal civilians and soldiers and instead had a broader classification for both of them under class “person”‘

- Model was unable to detect difference between “jets” and “helicopters” putting both under the “airplane” class

Planned Fixes

- Add more labeled images involving tanks
- Improve quality of manual annotations and potentially redo them

6. Integration into Web App

- Added a “Run Inference” button to Streamlit app
- When button is clicked the model runs a prediction on the selected image using the “best.pt” in the “models” folder
- Model inference uses the same bounding box format as the labeling tool in the app

7. Lessons Learned

- The pre-trained model allowed us to reduce training time and improve guess performance to a certain extent
- Labeling consistency is important for model accuracy and for our specific use case sometimes the model had random issues with labeling classes that were alike
- Visual tools like confusion matrices and error galleries are good for helping us improve the model and make it more accurate
- Lightweight models like YOLOv8n work well even on small datasets when trained correctly like through transfer learning but can sometimes have a decent amount of error with more complex images

8. Significant Files

App: “app.py” (in Github repository)
Model: “model.py” (in Github repository)
Dataset: “dataset v1/” (in Github repository)
Original Dataset Used: “xview_sample/” (in Github repository)
Final Model Iteration: “models/best.pt”