YOLOv8 Object Detection: Input Size Comparison Lab Report

Garbage Detection Dataset

---

1. Dataset Summary

Dataset Name: Garbage Detection (Roboflow)

Dataset Statistics:

- Total Images: 1,255
- Number of Classes: 1
  - garbage: All types of garbage/trash
- Data Split:
  - Training: 1,155 images (92%)
  - Validation: 50 images (4%)
  - Testing: 50 images (4%)

Key Dataset Characteristics:

- Format: YOLOv8 format (YOLO11 compatible)
- Image Resolution: Variable
- Annotation Format: Bounding boxes with class labels
- Challenge: Single class dataset; requires focus on detection across varying garbage types and contexts

---

2. Experimental Setup

2.1 Model Architecture

- **Model:** YOLOv8 Nano (YOLOv8n)
- **Framework:** Ultralytics YOLOv8
- **Pre-trained Weights:** YOLOv8 COCO pretrained
- **Hardware:** GPU (NVIDIA Tesla T4, 15GB VRAM)
- **Batch Size:** 32

- **Optimizer:** SGD
- **Learning Rate:** Auto-tuned

2.2 Experimental Configuration

Baseline Training (416×416):

- Input Size: 416×416 pixels
- Epochs: 10
- Training Time: 1.99 minutes
- Device: CUDA (GPU)

Experiment 1 (608×608):

- Input Size: 608×608 pixels
- Epochs: 10
- Training Time: 3.23 minutes
- Device: CUDA (GPU)

---

3. Results

3.1 Performance Metrics Comparison

Metric	416×416	608×608	Change
mAP@0.5	0.3520	0.3630	+0.0110 (+3.1%)
mAP@0.5:0.95	0.1420	0.1490	+0.0070 (+4.9%)
Precision	0.5550	0.4900	-0.0650 (-11.7%)
Recall	0.3220	0.4000	+0.0780 (+24.2%)
Inference Time (ms/image)	1.3	2.6	+1.3 (+100.0%)
Model Size (MB)	6.2	6.2	+0.0 (0.0%)
Training Time (minutes)	1.99	3.23	+1.24 (+62.3%)

3.2 Detailed Metrics (Per-Class)

Baseline (416×416):

- Precision: 0.5563

- Recall: 0.3217
- mAP@0.5: 0.3530
- mAP@0.5:0.95: 0.1427

Experiment (608×608):

- Precision: 0.4895
- Recall: 0.4000
- mAP@0.5: 0.3624
- mAP@0.5:0.95: 0.1490

---

4. Analysis & Discussion

4.1 Key Findings

1. **Recall Improvement (+24.2%):**
   - 608×608 detects significantly more garbage objects (recall: 0.32 → 0.40)
   - Critical for garbage detection applications where missing objects is problematic
   - Lower recall in 416×416 means some garbage is not detected
2. **Precision Trade-off (-11.7%):**
   - 608×608 has more false positives (precision: 0.555 → 0.490)
   - Trade-off: Better detection rate but more false alarms
   - Still acceptable for real-world deployment
3. **mAP Improvement (+3.1%):**
   - Overall accuracy improved slightly (0.352 → 0.363)
   - Consistent across both mAP@0.5 and mAP@0.5:0.95
   - More robust detections at various confidence thresholds
4. **Inference Speed Impact (+100%):**
   - 608×608 doubles inference time (1.3ms → 2.6ms per image)
   - On GPU: Still very fast for real-time applications
   - On CPU: Would be prohibitively slow (would be ~10-15ms per image)
5. **Training Efficiency:**

- 608×608 requires 62% more training time (1.99 min → 3.23 min)
- With GPU: Still very practical (3.23 minutes for 10 epochs)
- Model size remains identical (6.2 MB)

4.2 Input Size Impact Analysis

416×416 Advantages:

- Faster inference (1.3 ms/image)
- Higher precision (fewer false positives)
- Faster training time
- Better for deployment on resource-constrained devices

608×608 Advantages:

- Better recall (detects more garbage)
- Higher overall accuracy (mAP)
- More robust to object scale variations
- Better for accuracy-critical applications

4.3 GPU vs CPU Comparison

Training Time Comparison:

- **GPU (Tesla T4):** 1.99 minutes (416×416), 3.23 minutes (608×608)
- **Estimated CPU time:** 30-50 minutes each (15-25x slower)
- GPU acceleration essential for practical experimentation

Inference Speed:

- **GPU:** 1.3-2.6 ms/image (real-time capable, 380-770 FPS)
- **CPU:** 100-200 ms/image (10 FPS, not real-time)

---

5. Qualitative Analysis

Visual Comparison (5 Sample Images)

Detection results were compared on 5 test images:

Observations:

- Both models successfully detect garbage in most cases
- 608×608 shows slightly tighter bounding boxes
- 608×608 catches small garbage objects missed by 416×416
- 416×416 has fewer false positives in cluttered scenes
- Differences are subtle in visual inspection but significant in metrics

Sample Detections:

1. Trash bins with multiple garbage items - Both detect well
2. Loose garbage on ground - 608×608 performs better
3. Garbage in landfill - Both show strong performance
4. Urban litter - 608×608 catches more objects
5. Mixed debris - Similar performance, 608×608 slightly better

---

6. Conclusion

Summary

This lab successfully compared YOLOv8 performance across two input sizes (416×416 and 608×608) on the Garbage Detection dataset using GPU acceleration. Key findings demonstrate:

1. **Larger input sizes (608×608) provide better detection capability** with 24.2% higher recall, making them preferable for garbage detection applications where missing objects is costly.
2. **Input size creates a speed-accuracy trade-off:**
   - 416×416: Faster but misses more garbage
   - 608×608: Slower but catches more garbage
   - Choice depends on application requirements
3. **GPU acceleration is crucial:**
   - Enables practical experimentation with large models
   - Makes both input sizes real-time capable
   - 15-25x faster than CPU training

Recommendations

1. **For Production Deployment:** Use 608×608 with GPU for maximum accuracy
   - 24% better recall ensures garbage is detected
   - 2.6 ms inference still enables real-time processing
2. **For Edge Devices:** Use 416×416
   - Faster inference on limited hardware
   - Acceptable accuracy trade-off
3. **For Mobile Applications:** Consider model quantization
   - Reduce model size without major accuracy loss
   - Enable 416×416 on mobile with good performance
4. **Future Work:**
   - Experiment with 512×512 (compromise size)
   - Test with larger models (YOLOv8s, YOLOv8m)
   - Implement model pruning and quantization
   - Collect more diverse garbage images
   - Benchmark on CPU hardware

Impact Summary

- **Detection Rate:** +24.2% improvement with 608×608
- **Accuracy:** +3.1% improvement in mAP@0.5
- **Speed Cost:** +100% inference time (acceptable on GPU)
- **Training Cost:** +62.3% training time (still practical)
- **Deployment:** Both sizes viable; choice depends on requirements

---

7. Technical Details

Dataset Path: `/content/garbage_dataset`

Model Checkpoints:

- Baseline: `/content/runs/detect/baseline_4162/weights/best.pt`

- Experiment 1: `/content/runs/detect/experiment_608/weights/best.pt`

Environment:

- Python 3.12.12
- PyTorch 2.8.0+cu126 (CUDA)
- Ultralytics YOLOv8 8.3.225
- Hardware: NVIDIA Tesla T4 GPU (15GB VRAM)
- Runtime: Google Colab

Training Command:

```
model.train(
    data="data.yaml",
    epochs=10,
    imgsz=416/608,
    batch=32,
    device=0,
    seed=42
)
```

Appendix: Comparison with Road Signs Dataset

Previous Experiment (Road Signs - CPU):

- Hardware: Intel i5-12500H (CPU only)
- Training Time: 2.7-2.9 hours per experiment
- Model: YOLOv3
- Results: Similar input size trade-offs

Current Experiment (Garbage - GPU):

- Hardware: NVIDIA Tesla T4 (GPU)
- Training Time: 1.99-3.23 minutes per experiment
- Model: YOLOv8
- Results: Confirmed input size impact across different models and datasets

Key Takeaway: GPU acceleration and modern architectures (YOLOv8 vs YOLOv3) significantly improve experimentation speed and feasibility.