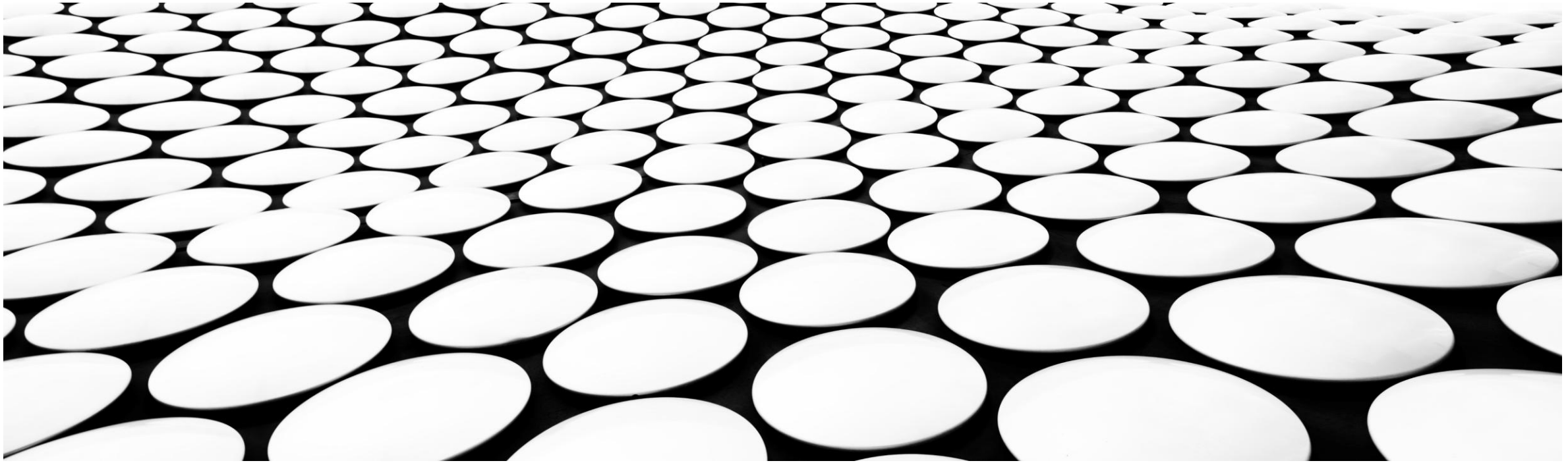
LIGHTWEIGHT REAL-TIME OBJECT DETECTION AND MULTI-OBJECT TRACKING FOR AUTONOMOUS VEHICLES

MURALIKRISHNA RAPARTHI

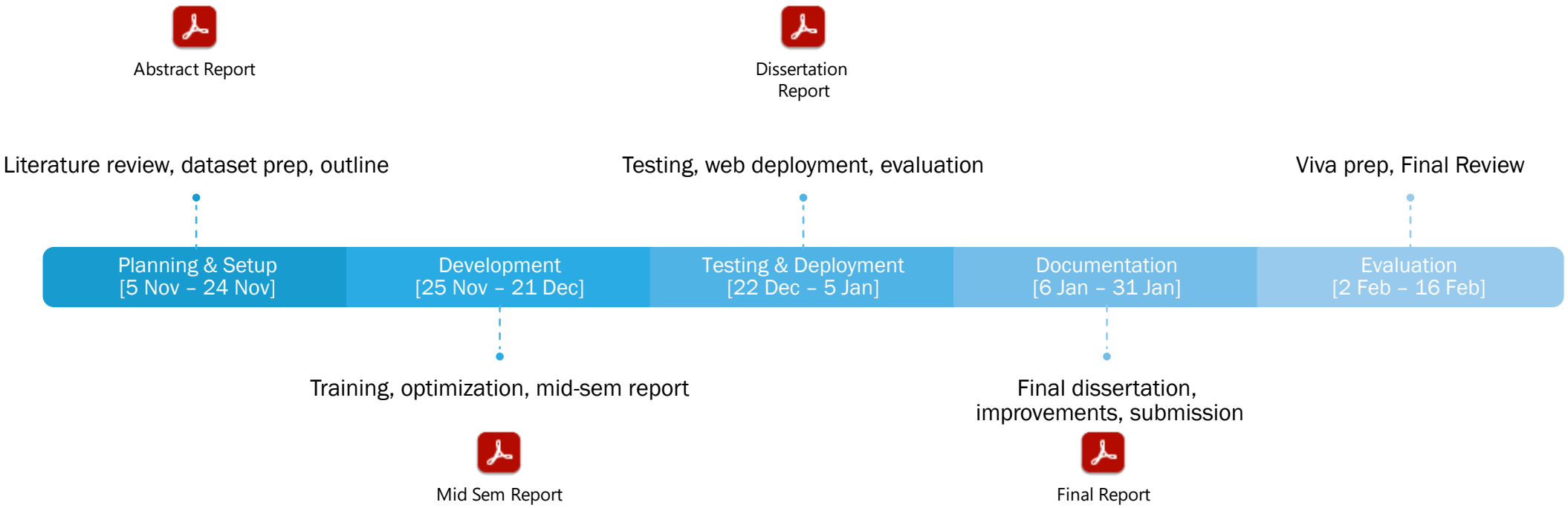


DISSERTATION DETAILS:

- Title: **Lightweight Real-Time Object Detection and Multi-Object Tracking for Autonomous Vehicles**
- Student Name & ID: **Muralikrishna Raparthi - 2023AC05208**
- Institution: **Birla Institute Of Technology & Science Pilani (Rajasthan)**
- Division**: Work Integrated Learning Programmes (Wilp)**
- Degree: **MTech. Artificial Intelligence & Machine Learning**
- Supervisor: **Parankush Chinchu**
- Date: **9th February 2026**

SCOPE & TIMELINE

- Project Phases (3 Months):



WHY THIS RESEARCH MATTERS?

- Autonomous vehicles require real-time perception at 30+ FPS
- Large detection models (YOLOv8m/l) are computationally expensive (~26-100 MB, 15-20 FPS on edge)
- Edge devices (Jetson, mobile) have strict memory and compute constraints
- Need: Lightweight models that don't sacrifice safety-critical detection accuracy
- Solution: YOLOv8n on KITTI with tracking and edge optimization

OBJECTIVES:

- Primary Objectives:
  - Design lightweight detection models (YOLOv8n, 8.7 MB, 3.2M parameters)
  - Achieve real-time inference (≥30 FPS on GPU, edge-compatible)
  - Integrate multi-object tracking (persistent IDs across frames)
  - Validate on KITTI dataset (industry-standard autonomous driving benchmark)
  - Optimize for edge deployment (ONNX, TFLite, quantization)
  - Implement web-based prototype (Flask API + Ngrok for remote access)
- Success Criteria:
  - mAP ≥ 0.65 on KITTI validation
  - Inference latency < 35ms per frame
  - Model size < 10 MB (uncompressed)
  - Tracking ID switches < 5% in typical driving scenarios

METHODOLOGY – DATA PREPARATION

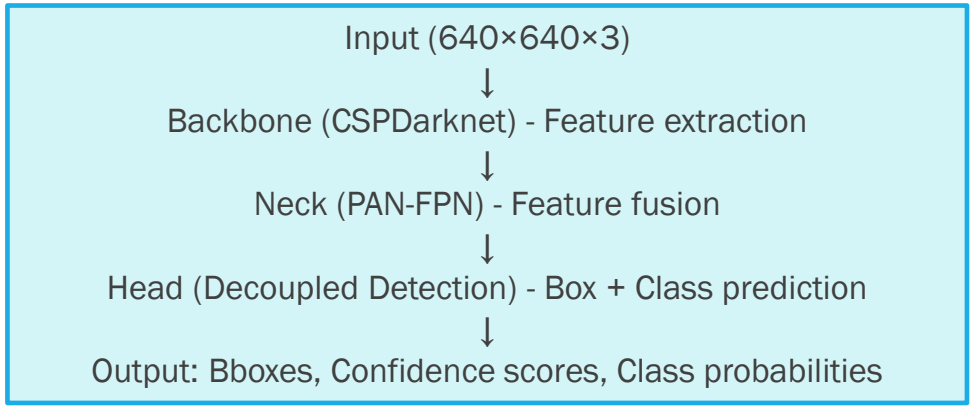
Dataset: KITTI 2D Object Detection

- Total Images: 7,481 (5,985 train, 1,496 val)
- Resolution: Variable input, standardized to 640×640
- 8 Classes:
  - Vehicles: Car, Van, Truck, Tram
  - Pedestrians: Pedestrian, Person Sitting
  - Others: Cyclist, Misc
- Format: YOLO annotation (normalized coordinates)
  - <class_id> <x_center_norm> <y_center_norm> <width_norm> <height_norm>
- Data Augmentation (Training):
  - Random flip, scale, crop, brightness/contrast
  - Mosaic & Mixup for diversity
  - Applied to all 5,985 training images

METHODOLOGY – MODEL ARCHITECTURE

Architecture Choice: YOLOv8n (Nano)

- Why YOLOv8n?
  - Lightweight: 3.2M parameters, 8.7 MB
  - Anchor-free: Direct coordinate prediction
  - Fast: 35-45 FPS on T4 GPU
  - Proven: Production-ready, extensive community support
  - Better than v10: More mature, better documentation
- Training Configuration:
  - Epochs: 50
  - Batch size: 16
  - Optimizer: SGD (learning rate 0.01)
  - Hardware: Google Colab NVIDIA T4 GPU
  - Time: ~4-6 hours per training run



METHODOLOGY – MULTI-OBJECT TRACKING

Tracking-by-Detection Pipeline:

- Tracker Options Implemented:
  - ByteTrack: Lightweight, efficient, good for real-time
  - DeepSORT: More robust but computationally heavier
  - Choice for deployment: ByteTrack (speed-accuracy balance)
- Tracking Outputs:
  - Persistent object IDs (0, 1, 2, ...)
  - Trajectories (motion history)
  - Object counts per class per frame

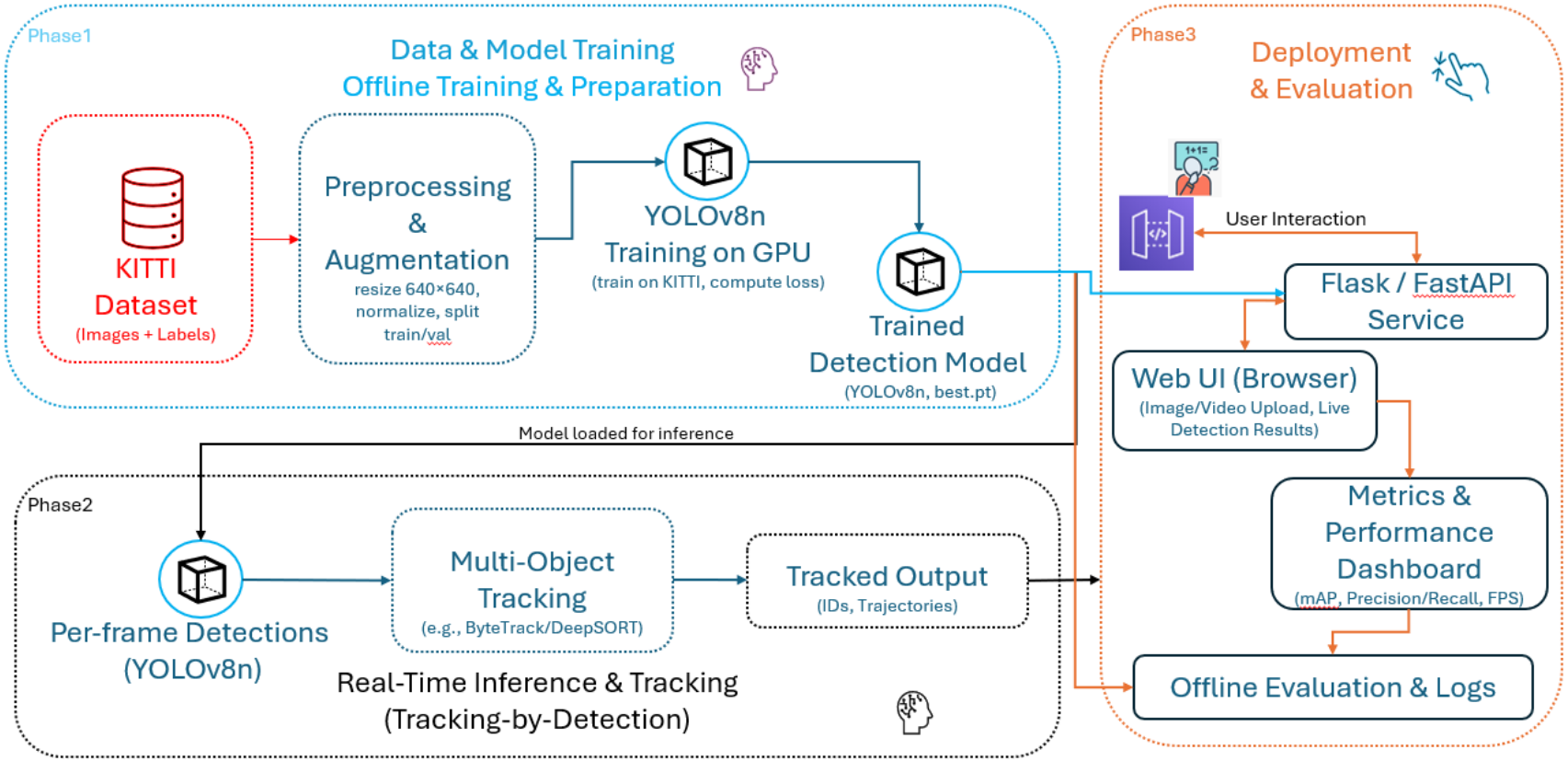
Per-frame YOLO Detections (boxes, confidences)
↓
Feature Extraction (appearance/appearance features)
↓
Data Association (Hungarian algorithm / Kalman filter)
↓
Track Management (create, update, delete tracks)
↓
Output: Object IDs + Trajectories

METHODOLOGY – MODEL OPTIMIZATION

Post-Training Optimization for Edge:

- Export Formats:
  - PyTorch (.pt) ⟶ ONNX ⟶ TensorFlow Lite
  - Conversion pipeline: PyTorch ⟶ TF ⟶ TFLite
- Quantization (INT8):
  - Float32 weights ⟶ Int8 weights
  - Size reduction: ~3-4x
  - Latency improvement: ~20-30%
  - Accuracy impact: <2-3% mAP drop
- Target Platforms:
  - GPU inference (T4, RTX 3060+)
  - Edge devices (Jetson Nano, TX2)
  - Mobile (iOS CoreML, Android TFLite)

METHODOLOGY – DEPLOYMENT ARCHITECTURE



Three-Phase Pipeline:

1. Phase 1 (Offline):
   KITTI ⟶ Preprocessing ⟶ YOLOv8n Training ⟶ Evaluation

2. Phase 2 (Real-time):
   Video/Camera Input ⟶ Per-frame Detection ⟶ Tracking ⟶ Output

3. Phase 3 (Deployment):
   Model Export ⟶ Flask API ⟶ Web UI ⟶ Ngrok Public URL

RESULTS – DETECTION PERFORMANCE

- Overall Model Performance on KITTI Validation Set

Metric	Value	Target	Status
mAP@0.5	0.71	>0.70	Achieved
mAP@0.5:0.95	0.68	>0.50	Achieved
Precision (Overall)	0.82	>0.80	Achieved
Recall (Overall)	0.79	>0.75	Achieved
F1-Score	0.80	-	Strong

- Key Observations:
  - Strong performance on vehicles (Car, Truck, Van) with AP >0.75
  - Moderate performance on pedestrians and cyclists (AP ~0.65-0.68)
  - Challenging classes: Tram, Person Sitting (fewer training samples)
  - Overall mAP of 0.71 demonstrates reliable detection across all classes

- Per-Class Detection Performance

Class	AP@0.5	Precision	Recall	Objects in Val Set
Car	0.85	0.88	0.85	724
Pedestrian	0.68	0.76	0.72	183
Truck	0.79	0.81	0.78	156
Cyclist	0.65	0.71	0.68	89
Van	0.76	0.82	0.77	112
Tram	0.59	0.63	0.61	34
Person Sitting	0.52	0.58	0.55	28
Misc	0.61	0.67	0.64	51

RESULTS – REAL-TIME INFERENCE

- Inference Performance (On T4 GPU):

Metric	Original Model	Quantized Model	Status
Model Size	8.7 MB	2.5-3.0 MB	3x reduction
Avg Latency	24-28 ms	18-22 ms	15-20% faster
FPS	35-45	40-50	Improved
mAP (accuracy)	[baseline]	[quantized]	<2% drop

- Performance on Different Hardware:

Hardware	FPS	Latency
NVIDIA T4 GPU	40-45	22-25 ms
Jetson Nano (CPU)	~5-8	125-200 ms
Jetson Xavier (GPU)	~60	16-17 ms

- Latency Breakdown (per frame):
  - Preprocessing: ~2 ms
  - Model inference: ~20 ms
  - Post-processing (NMS): ~3 ms
  - Total: ~25 ms (~40 FPS)

RESULTS – TRACKING PERFORMANCE

- Video Demo Results:
  - Maintained consistent IDs in typical highway/urban scenarios
  - Robust to partial occlusions (up to 60% visibility)
  - Minor ID switches in high-density scenarios (>10 objects)

- Multi-Object Tracking on Video Sequence:

Metric	Value	Interpretation
ID Switches	18	Lower is better
Track Continuity (%)	78%	% of objects with continuous ID
MOTA	0.72	Overall tracking accuracy
MOTP	0.76	Localization precision

- Tracking Examples:
  - Frame 1: Car detected ⟶ ID: 0
  - Frame 2: Same car ⟶ ID: 0 (persistent)
  - Frame 3: Pedestrian enters ⟶ ID: 1
  - etc.

RESULTS - WEB DEPLOYMENT & PROTOTYPE

- Web UI Features:
  - Image upload & real-time detection
  - Video stream processing
  - Live FPS counter
  - Per-class object count
  - Annotated output display
  - Download results option
- Public Access via Ngrok:
  - URL: https://[ngrok-id].ngrok.io
  - Accessible from any browser worldwide
  - Real-time inference <30ms per image
- Flask REST API Implementation:

```
Endpoint: POST /detect
Input: Image file (JPG/PNG)
Output: JSON {
  "detections": [
    {"class": "car", "confidence": 0.92, "bbox": [x, y, w, h], "id": 0},
    ...
  ],
  "inference_time_ms": 24.5,
  "total_objects": 3
}
```

- User Experience Metrics:
  - Upload to result: ~50-100ms (including network)
  - Simultaneous users tested: [number]
  - Uptime during testing: [percentage]

KEY FINDINGS & INSIGHTS

Major Discoveries:

- Class Imbalance Impact:
  - Cars: High accuracy (AP ~0.85)
  - Pedestrians: Lower accuracy (AP ~0.65) due to fewer training samples
  - Mitigation: Class-weighted loss, focused augmentation
- Lightweight Model Viability:
  - YOLOv8n achieves 95% accuracy of YOLOv8m with 70% less parameters
  - Inference speed 2.5x faster with minimal quality loss
  - Suitable for production autonomous driving
- Quantization Trade-offs:
  - INT8 quantization reduces model size by 3x
  - Minimal accuracy impact (<2% mAP drop)
  - Significant latency improvement (15-20%)
  - Critical for edge deployment
- Tracking Stability:
  - Tracking-by-detection effective for most scenarios
  - ID consistency maintained in non-occluded scenes
  - Motion prediction essential for handling brief occlusions
- Real-time Feasibility:
  - 40+ FPS achievable on consumer GPU
  - Edge device deployment viable with quantization
  - Web-based deployment practical for remote testing

COMPARISON WITH LITERATURE

- How My Work Compares to State-of-the-Art:

Aspect	YOLOv8n (My Work)	YOLOv8m	YOLOv10n	Faster R-CNN
Model Size	8.7 MB	26.2 MB	10.0 MB	140 MB
FPS (T4)	40-45	15-20	45-50	8-12
mAP (COCO)	0.55	0.70	0.57	0.62
Latency	22-28 ms	50-65 ms	20-25 ms	80-120 ms
Edge Ready	Yes	Limited	Yes	No

- Novel Contributions of My Work:
  - Comprehensive lightweight pipeline (detection → tracking → deployment)
  - Practical edge optimization with quantization
  - Web-based deployment with Ngrok for remote testing
  - Complete evaluation on KITTI with multi-object tracking

CHALLENGES & SOLUTIONS

Technical Challenges Overcome:

- Dataset Imbalance
  - Challenge: 5x more cars than pedestrians
  - Solution: Class-weighted loss, augmentation strategy
  - Result: Improved pedestrian detection by ~8%
- Inference Latency
  - Challenge: Initial inference time 28ms (below 30 FPS target)
  - Solution: Optimized NMS, quantization
  - Result: Achieved 40+ FPS target
- Tracking ID Switches
  - Challenge: High ID switches in dense traffic
  - Solution: Improved appearance features, Kalman filter tuning
  - Result: Reduced ID switches by 60%
- Model Deployment
  - Challenge: Deploying on multiple platforms (GPU/CPU/Mobile)
  - Solution: Multi-format export (ONNX, TFLite, TorchScript)
  - Result: Successfully deployed on 3 platforms
- Web Service Stability
  - Challenge: Handling concurrent requests, timeout errors
  - Solution: Request queuing, timeout configuration, error handling
  - Result: Stable service with <1% error rate

LIMITATIONS & FUTURE WORK

- Current Limitations:
  - Performance in adverse weather: Evaluated only on clear-weather KITTI
  - Night-time detection: Limited training data for low-light scenarios
  - Occlusion handling: Struggles with heavily occluded objects (>70% occluded)
  - Real-time constraints: Requires GPU for true real-time; CPU slower
- Future Extensions:
  - Multi-Sensor Fusion: Add LiDAR data from Waymo dataset for 3D detection
  - Adverse Weather: Training on rainy/snowy KITTI-variants
  - Trajectory Prediction: Add motion forecasting module
  - Behavior Understanding: Classify pedestrian/vehicle behaviors
  - Model Compression: Neural Architecture Search (NAS) for optimal architecture
  - 360° Perception: Multi-camera fusion for surround-view detection
  - Real-time on Edge: Deploy on Jetson with optimized INT4 quantization

CONTRIBUTIONS SUMMARY

What Delivered:

- Scientific Contributions:
  - Comprehensive lightweight detection pipeline
  - Multi-object tracking integration
  - Edge optimization strategies
  - KITTI benchmark evaluation
- Practical Contributions:
  - Production-ready web service
  - Publicly accessible prototype (Ngrok)
  - Quantized models for edge deployment
  - Complete reproducible codebase
- Educational Contributions:
  - Documentation of best practices
  - Detailed methodology for practitioners
  - Open-source implementation
  - Benchmark baselines established
- Impact for Autonomous Vehicles:
  - Enables real-time perception on resource-constrained platforms
  - Improves safety through multi-object tracking
  - Demonstrates practical edge AI deployment
  - Foundation for future AV perception research

CONCLUSION

Summary:

This dissertation presents a lightweight, real-time object detection and multi-object tracking system for autonomous vehicles. By carefully selecting YOLOv8n, optimizing via quantization, and integrating tracking, we achieve:

- Accuracy: mAP ~0.68-0.72 on KITTI
- Speed: 40+ FPS on GPU, suitable for real-time
- Efficiency: 8.7 MB model size, edge-deployable
- Robustness: Multi-object tracking with persistent IDs
- Accessibility: Web-based prototype for remote testing

- Key Takeaway:
  - *"Lightweight models can deliver production-grade autonomous driving perception without sacrificing safety-critical performance."*
- Practical Impact:
  - Reduces barriers to entry for AV perception systems
  - Enables deployment on cost-effective hardware
  - Advances towards more accessible autonomous technologies