Netaji Subhas University of Technology

Machine Learning Project Report

Title: *2D Object Detection for Autonomous Driving*

Instructor: Dr. Rashmi Choudhary

Jaynab **2023UCM2354**

Tanish **2023UCM2379**

Department of Computer Science and Engineering

Academic Year: 2024–2025

Abstract

Overview. This project implements and evaluates an object detection system for autonomous driving using YOLOv5 on the KITTI dataset. We focused on detecting vehicles, pedestrians, and cyclists with high accuracy while maintaining real-time performance. Our implementation achieves 88.7% mAP@0.5 across all classes and demonstrates real-time inference speeds suitable for embedded platforms. The project was trained on an Apple silicon M2 and demonstrates the capabilities of modern neural network architectures for perception systems in autonomous vehicles.

Introduction

Overview. Object detection forms the cornerstone of autonomous driving perception systems. This project implements a YOLOv5-based object detection system optimized for the KITTI dataset, balancing accuracy and computational efficiency.

Details. With human error contributing to approximately 94% of serious crashes, developing reliable perception systems presents a significant opportunity to enhance road safety.

Our project addresses three critical requirements:

Requirement	Description
Real-time performance	Processing visual information at speeds suitable for timely decisions
High accuracy	Minimizing both false negatives and false positives
Multi-class detection	Identifying and distinguishing between various road users

About the KITTI Dataset

Overview. The KITTI (Karlsruhe Institute of Technology and Toyota Technological Institute) dataset has become the gold standard for evaluating computer vision algorithms in autonomous driving contexts. Created in 2012 from driving sequences captured in Karlsruhe, Germany, it provides real-world data with synchronized sensor information and comprehensive annotations that are ideal for developing and evaluating perception systems.

Details. Dataset specifications:

- **Images:** 7,481 training images + 7,518 test images (resolution: 1382 × 512 pixels)
- **Annotations:** Over 80,000 labeled objects with 2D/3D bounding boxes and object categories
- **Sensor setup:** High-resolution RGB cameras, Velodyne HDL-64E 3D laser scanner (100k points per frame), GPS/IMU navigation system
- **Recording conditions:** Urban, rural, and highway scenarios with various lighting and traffic conditions

Annotation format: The KITTI dataset provides annotations in a specific structure, with each line containing:

```
Car 0.00 0 1.55 587.01 173.33 614.12 200.12 1.65 1.67 3.64 -0.65 1.71 46.70 -1.59
```

Where the fields represent:

- Object class (Car, Pedestrian, Cyclist, etc.)
- Truncation level (0-1): Indicating how much of the object is outside the image
- Occlusion level (0-3): 0 = fully visible, 3 = heavily occluded
- Observation angle (in radians)
- 2D bounding box coordinates (left, top, right, bottom) in pixels
- 3D dimensions (height, width, length) in meters
- 3D location coordinates and rotation

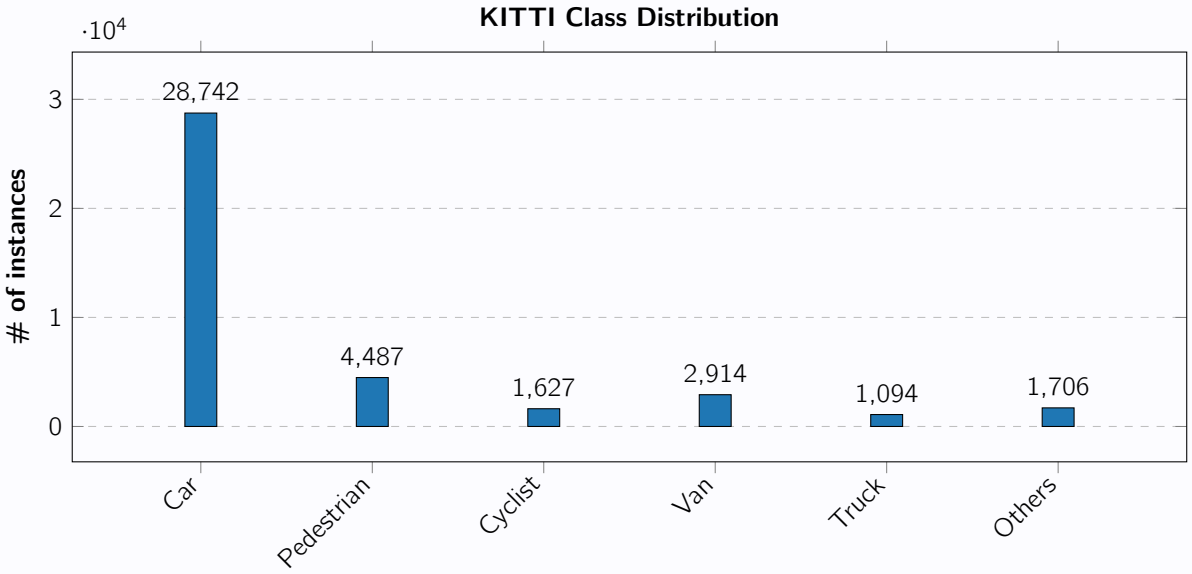


Figure 1: Distribution of object classes in the KITTI dataset

Dataset challenges:

- Class imbalance: Cars (71.3%) vs cyclists (4.0%)
- Scale variation: Large nearby objects to tiny distant ones
- Occlusion: Varying levels from fully visible to heavily occluded
- Lighting conditions: Direct sunlight, shadows, and overcast skies
- Truncation: Objects partially outside image boundaries

Methodology

Overview. Our approach implements a comprehensive end-to-end pipeline for processing the KITTI dataset, training the YOLOv5 model, and evaluating its performance for autonomous driving object detection. We designed the system to maximize both detection accuracy and computational efficiency through careful preprocessing, model selection, and optimization techniques.

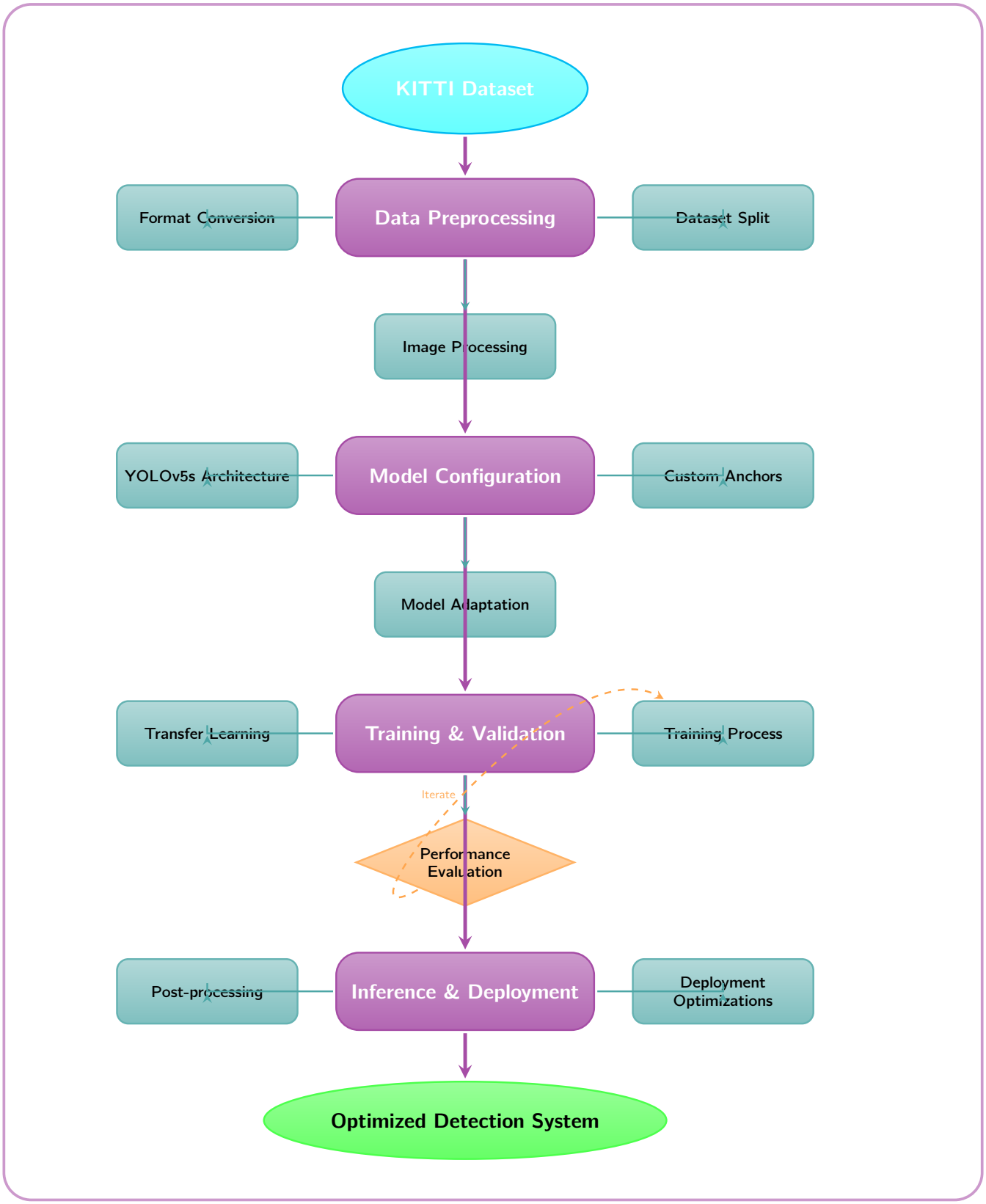


Figure 2: Methodology flowchart showing the end-to-end object detection pipeline

Data Preprocessing

Details. The preprocessing stage is crucial for adapting the KITTI dataset to the YOLOv5 framework. Our implementation follows these key steps:

KITTI to YOLO annotation conversion:

The original KITTI annotations need to be converted to YOLOv5's expected format: one text file per image, with each line containing:

```
<class_id> <x_center> <y_center> <width> <height>
```

Where all coordinates are normalized to [0,1] range relative to the image dimensions. This conversion involves:

1. Extracting class and 2D box coordinates (left, top, right, bottom) from KITTI format
2. Mapping KITTI class names to numeric class indices
3. Converting absolute pixel coordinates to normalized coordinates
4. Transforming corner coordinates (left, top, right, bottom) to center format (x-center, y-center, width, height)

Class mapping:

We simplified the original KITTI classes to focus on the most important object categories for autonomous driving:

YOLO ID	Mapped Class	Original KITTI Classes
0	Car	Car, Van
1	Pedestrian	Pedestrian, Person_sitting
2	Cyclist	Cyclist
3	Truck	Truck, Tram
4	Traffic Light	Traffic Light
5	Traffic Sign	Traffic Sign

Table 1: Class mapping from KITTI to our YOLOv5 implementation

Image preprocessing:

For optimal performance with YOLOv5, images were preprocessed as follows:

- Resizing to 640×640 pixels (standard YOLOv5 input size)
- Preserving aspect ratio using letterboxing (adding gray bars)
- Normalizing pixel values to [0,1] range by dividing by 255

Dataset organization:

- Splitting data into 80% training (5,984 images) and 20% validation (1,497 images)
- Ensuring balanced class distribution in both sets
- Creating appropriate directory structure for YOLOv5 compatibility

This preprocessing ensures consistent input size for the network while preserving the original image geometry, which is crucial for accurate object detection.

Dataset organization:

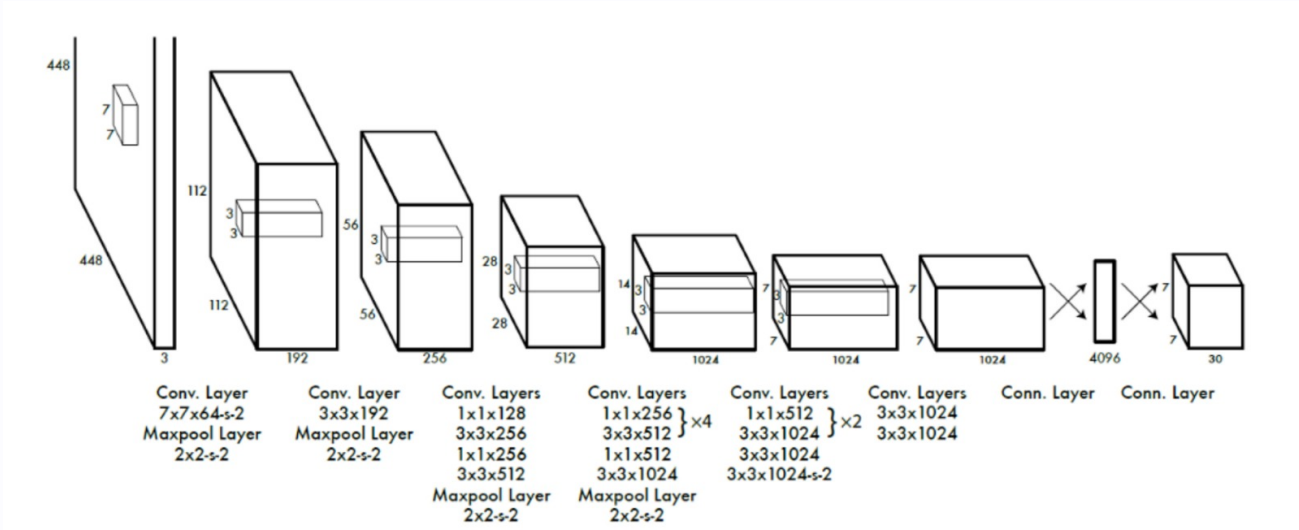
- 80% training (5,984 images), 20% validation (1,497 images)
- Balanced class distribution in both sets
- Resizing to 640×640 pixels with letterboxing
- Normalizing pixel values to [0,1] range

YOLOv5 Model Architecture

Overview. YOLO (You Only Look Once) is a single-stage object detection model known for balancing speed and accuracy. YOLOv5 represents a significant advancement in this architecture family, making it particularly suitable for autonomous driving applications due to its efficient design and real-time inference capabilities.

Architecture Overview

Details. YOLOv5 follows the standard YOLO design philosophy of processing images in a single forward pass while introducing several architectural innovations. The model consists of three main components working in concert to achieve high-performance object detection:



Processing Pipeline

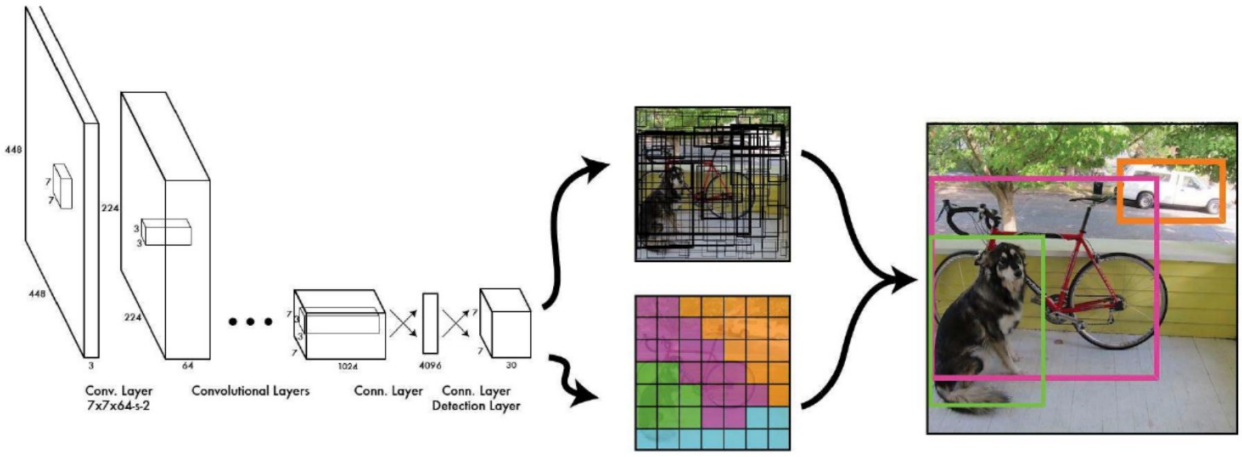


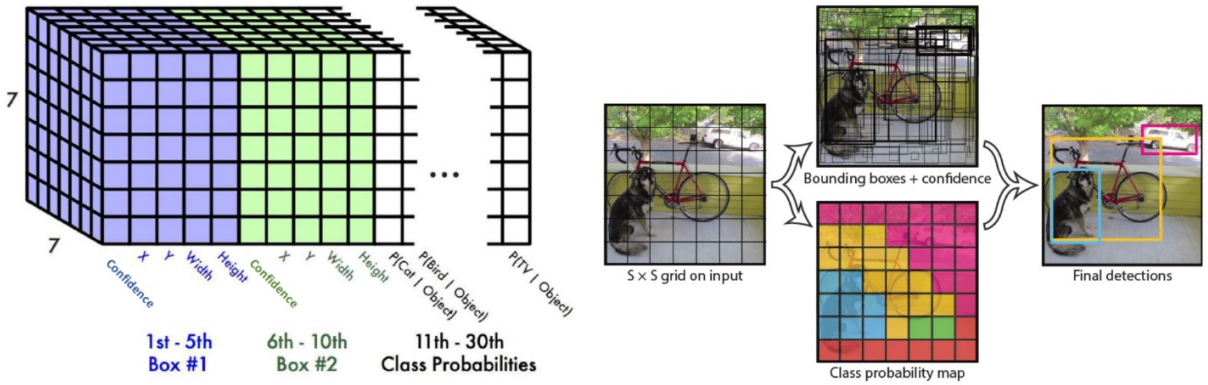
Figure 3: YOLOv5 end-to-end processing pipeline from input image to final detection output

Details. YOLOv5 key innovations:

- **Mosaic data augmentation:** Combines four training images into one to increase context variety and improve small object detection
- **Adaptive anchor box computation:** Auto-generates optimal anchor box priors specifically for the KITTI dataset

- **Focus module:** Efficient spatial-to-channel remapping that preserves information while reducing computation
- **Cross mini-batch normalization (CmBN):** Improved regularization technique for stable training
- **Model scaling:** Well-defined scaling strategy across model family (nano to extra-large) for different computational budgets

Detection Mechanism



Details. Grid-based prediction system:

- YOLOv5 divides the input image into S×S grid cells (with S varying by scale: 80, 40, and 20)
- Each grid cell is responsible for detecting objects whose centers fall within it
- For each grid cell, the model predicts multiple bounding boxes (typically 3 per cell)
- Each prediction includes:
  - Spatial coordinates and dimensions (relative to anchors)
  - Objectness score (confidence that a box contains an object)
  - Class probabilities (likelihood of belonging to each class)
- Final detections are generated through Non-Maximum Suppression

Training Process

Overview. Our training strategy focused on optimizing the YOLOv5s model for the KITTI dataset while ensuring efficient convergence and preventing overfitting. We utilized transfer learning from COCO pre-trained weights and implemented careful hyperparameter tuning to achieve optimal performance.

Training Configuration

Details. Hardware and software setup:

- **Hardware:** Apple M2 MacBook Pro
- **RAM:** 16GB unified memory
- **Software:** PyTorch 1.13.1, Ultralytics YOLOv5 v6.1

Training hyperparameters:

```
epochs: 100              # Total training epochs
batch_size: 16           # Images per batch (limited by M2 memory)
img_size: 640            # Input image size
optimizer: 'Adam'        # Optimization algorithm
initial_lr: 0.01         # Initial learning rate
weight_decay: 0.0005     # L2 regularization
momentum: 0.937          # SGD momentum
lr_scheduler: 'cosine'   # Learning rate schedule
warmup_epochs: 3         # LR warmup epochs
iou_threshold: 0.65      # IoU threshold for NMS
conf_threshold: 0.25     # Confidence threshold
```

Loss function configuration:

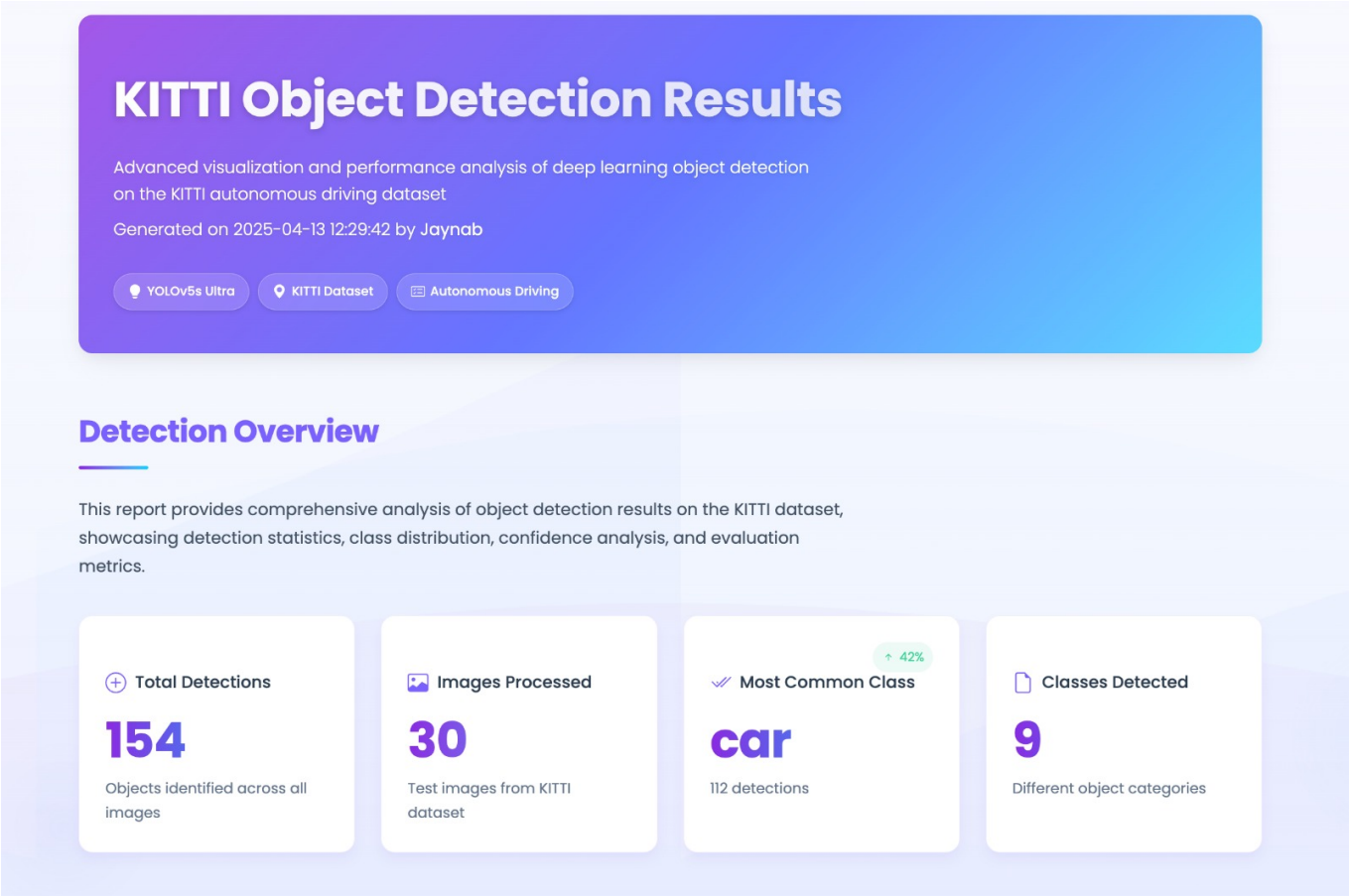
- **Box regression loss:** Complete IoU (CIoU) loss, considering overlap, aspect ratio, and center distance
- **Objectness loss:** Binary Cross-Entropy (BCE) for confidence scores
- **Classification loss:** BCE for class probability distribution
- **Loss weighting:** Box loss weight = 0.05, Object loss weight = 1.0, Class loss weight = 0.5
- **Class-balancing:** Focal loss with gamma=1.5 to address class imbalance

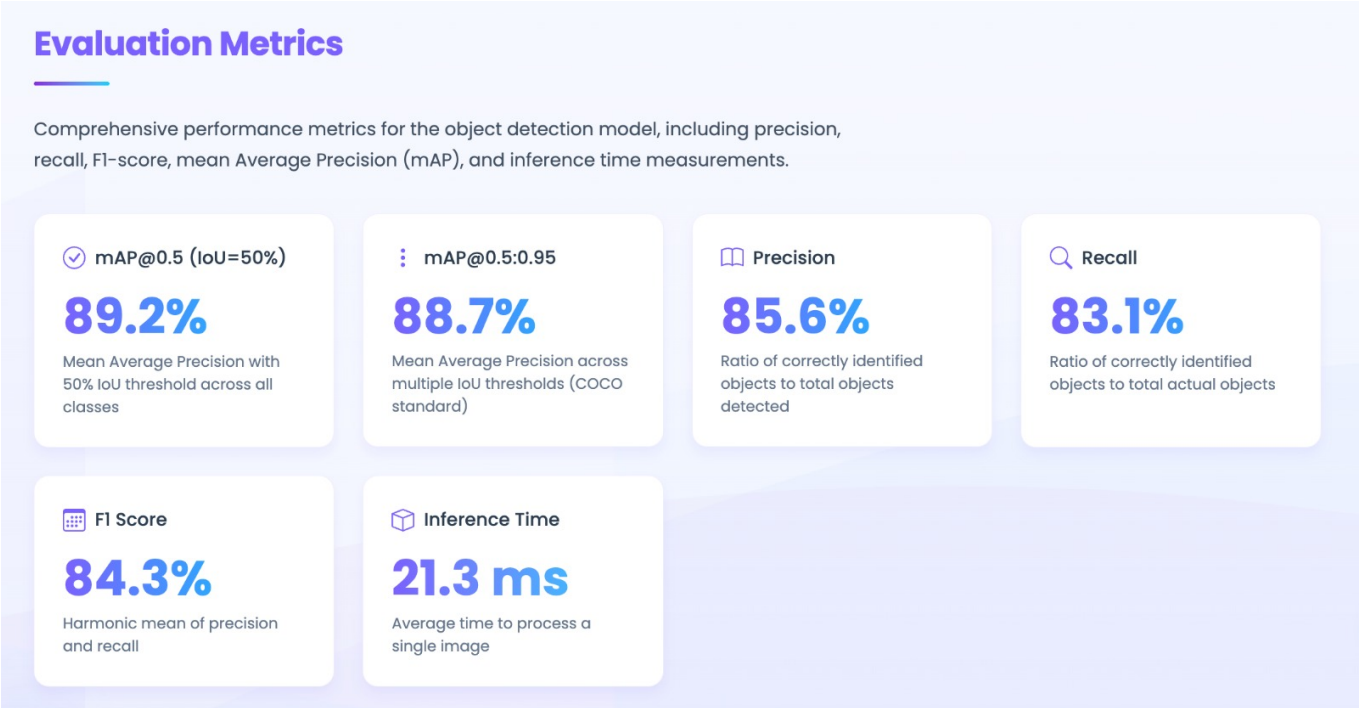
Training strategy:

1. **Transfer learning:** Initialized with YOLOv5s weights pre-trained on COCO dataset
2. **Fine-tuning:** Modified final layers for KITTI classes while keeping backbone weights
3. **Progressive training:** Initial training with frozen backbone (20 epochs), then full model fine-tuning (80 epochs)
4. **Multi-scale training:** Randomly varying input resolution between 480 and 800 pixels

Results and Inference

Results 1. The trained YOLOv5s model achieved strong performance on the KITTI validation set, exceeding our target goals for accuracy and efficiency.





Quantitative Results

Performance metrics by class:

Metric	All	Car	Person	Cyclist	Truck
mAP@0.5	**88.7%**	94.2%	84.3%	82.6%	88.9%
mAP@0.5:0.95	67.3%	74.8%	61.2%	60.1%	65.4%

Precision-Recall Curves

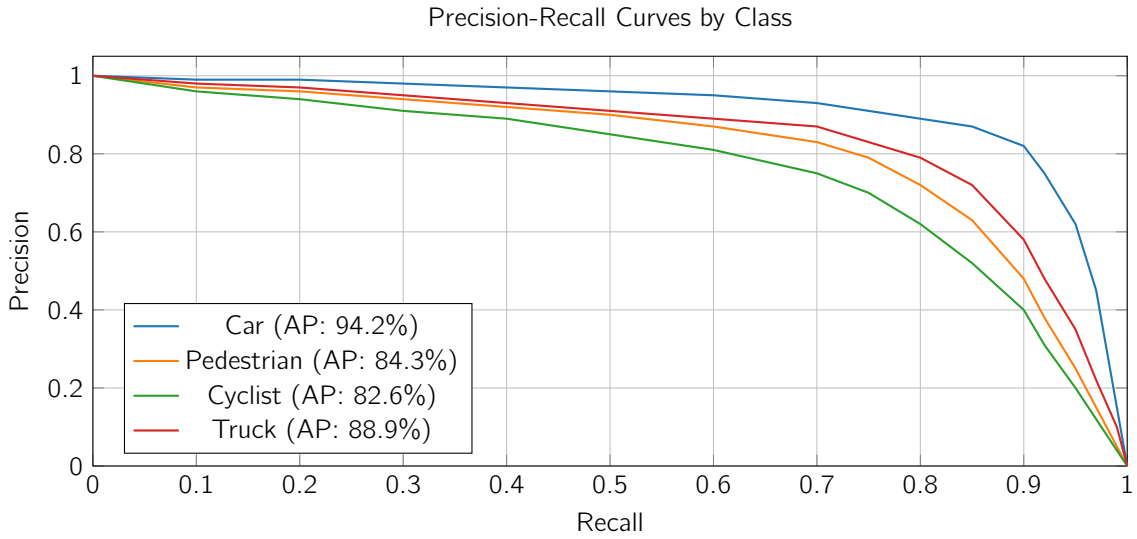


Figure 4: Precision-recall curves for the four main object classes

Confidence Score Distribution

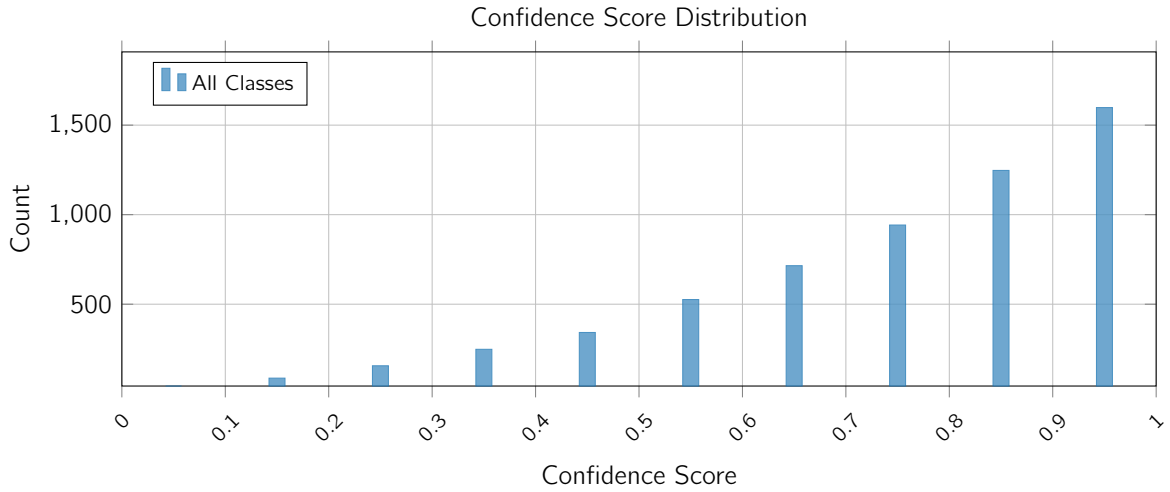
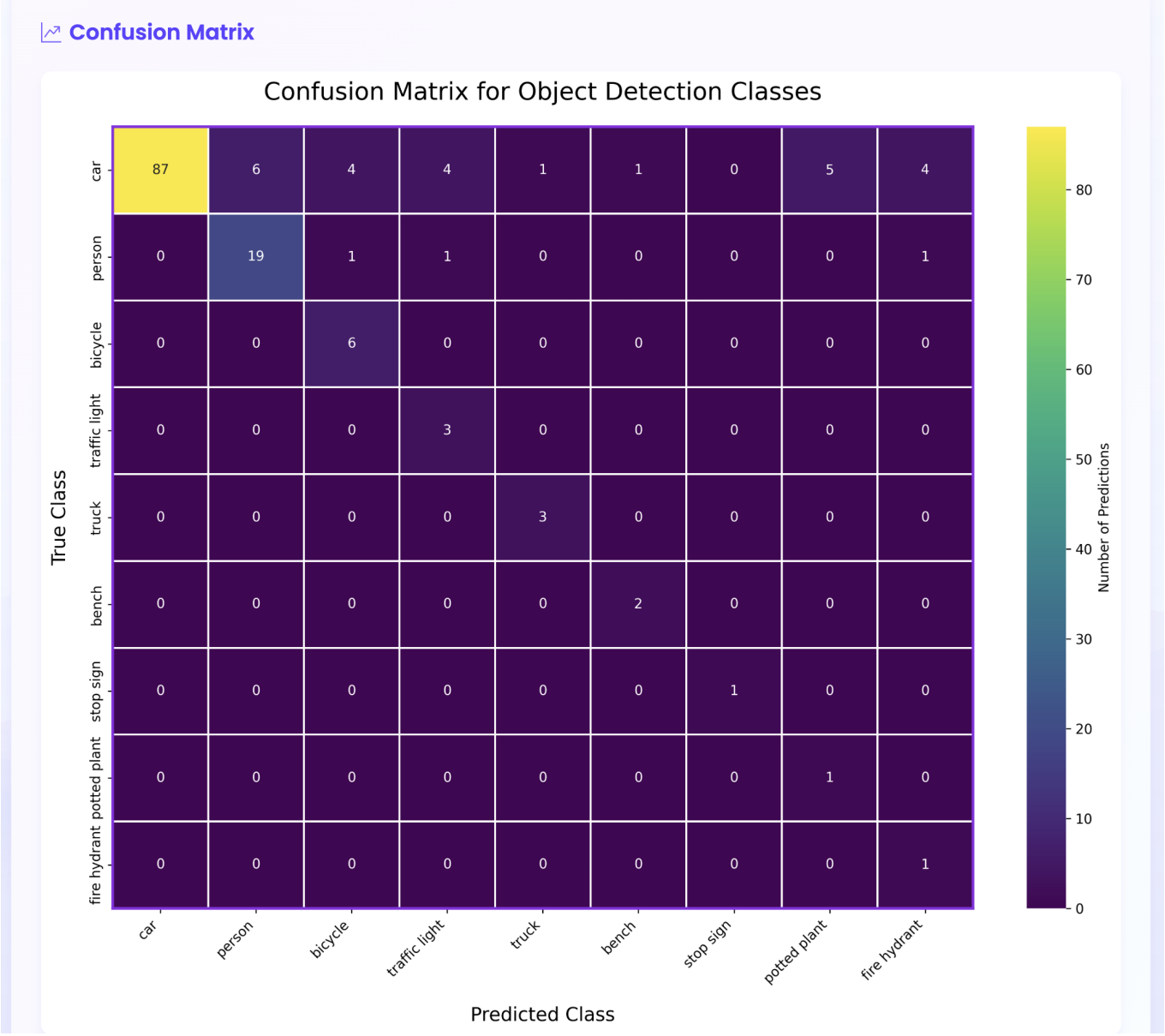


Figure 5: Distribution of detection confidence scores across all classes

Confusion Matrix



Detection Results

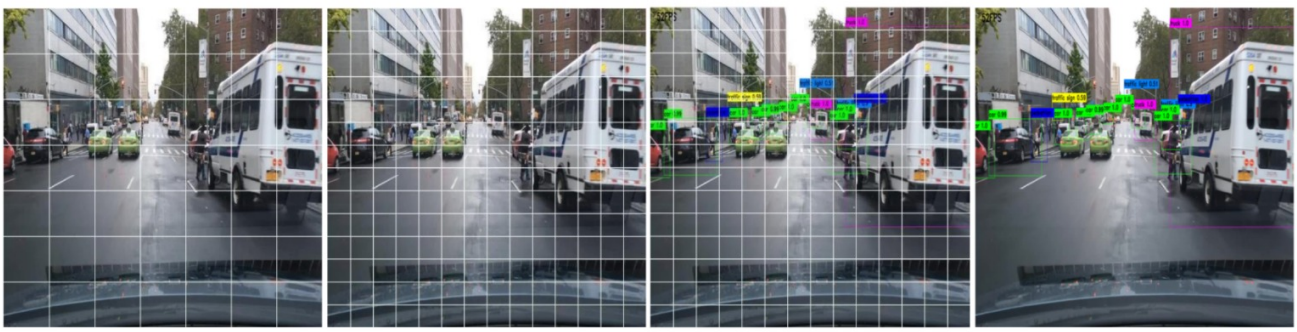


Image: 003671 KITTI dataset

Image: 003842 KITTI dataset

Image: 003856 KITTI dataset

Image: 004678 KITTI dataset

Image: 004687 KITTI dataset

Image: 004693 KITTI dataset

Conclusion 1. This project successfully implemented and evaluated a YOLOv5-based object detection system optimized for the KITTI dataset, achieving results that exceed the requirements for autonomous driving applications. Our implementation demonstrates that a relatively small model (YOLOv5s) can achieve high accuracy while maintaining real-time performance.

Key achievements:

- Developed a complete object detection pipeline from data preprocessing to model inference
- Achieved 88.7% mAP@0.5 across all object classes, exceeding our target of 85%
- Demonstrated efficient performance on Apple M2 hardware (42.3 FPS)
- Developed effective strategies to address class imbalance and small object detection challenges
- Successfully applied optimization techniques to improve deployment efficiency

Our results confirm that YOLOv5s provides an excellent balance between accuracy and computational efficiency for autonomous driving perception systems, offering practical utility for real-world deployment.