AI OBJECT DETECTION

A PROJECT REPORT

by

Snigdha Pratap (202410116100209)

Tanishka Singh (202410116100219)

Tanisha (202410116100216)

Shruti Sagar (202410116100203)

Session:2024-2025 (II Semester)

Under the supervision of

Ms. Komal Salgotra

KIET Group of Institutions, Delhi-NCR, Ghaziabad

DEPARTMENT OF COMPUTER APPLICATIONS

KIET GROUP OF INSTITUTIONS,
DELHI-NCR, GHAZIABAD-201206

(2024- 2025)

Introduction

AI Object Detection is a cutting-edge technology that enables machines to **locate and identify objects within digital images or video streams** using artificial intelligence. Unlike simple image classification, which assigns a single label to an entire image, object detection goes a step further by identifying **what objects are present** and **where they are located** using bounding boxes.

At the core of this technology are **deep learning models**, especially Convolutional Neural Networks (CNNs), which learn patterns from vast amounts of labeled data. These models can recognize a wide variety of objects—from people and vehicles to animals and everyday items—with impressive speed and accuracy.

AI object detection plays a crucial role in many modern applications such as **autonomous vehicles**, **surveillance systems**, **healthcare diagnostics**, and **retail analytics**. By combining **object localization** and **classification**, it allows machines to better understand and interact with their environment in real-time.

Objective

The main objective of AI Object Detection is to **automatically identify and localize multiple objects within an image or video frame** using advanced deep learning techniques. It aims to:

- **Detect the presence of objects** in various visual inputs.
- **Draw bounding boxes** around detected objects to pinpoint their exact locations.
- **Classify objects** into predefined categories (e.g., person, car, dog).
- Enable **real-time and accurate recognition** to support intelligent decision-making in applications such as autonomous driving, surveillance, robotics, and smart cities.

Ultimately, AI Object Detection seeks to enhance machines’ understanding of the visual world to make them more interactive, responsive, and intelligent.

Scope

The scope of AI Object Detection is broad and continues to expand with advancements in deep learning and computer vision. It encompasses a wide range of real-world applications and industries, including:

- **Autonomous Vehicles:** Detects pedestrians, vehicles, road signs, and obstacles for safe navigation.
- **Surveillance and Security:** Monitors public spaces for suspicious activities and intrusions.
- **Healthcare:** Assists in medical imaging analysis, such as tumor detection in X-rays or MRIs.
- **Retail and E-commerce:** Tracks customer movement, product placement, and shelf management.
- **Agriculture:** Identifies crops, pests, and diseases from drone or satellite imagery.
- **Manufacturing:** Detects defects on assembly lines for quality control.
- **Smart Cities:** Enables traffic monitoring, parking management, and public safety systems.

With real-time processing capabilities and high accuracy, AI Object Detection has the potential to **transform how machines perceive and interact with the world**, making it a vital tool in the era of automation and intelligent systems.

Methodology

The methodology of AI Object Detection involves several key steps that combine data processing, deep learning, and evaluation techniques. Below is a typical workflow:

1. **Data Collection and Annotation**
    - Large datasets of labeled images are collected.
    - Each object in an image is manually annotated with bounding boxes and class labels.
    -

2. **Preprocessing**
    - Images are resized, normalized, and augmented (flipped, rotated, etc.) to improve model generalization.
3. **Model Selection**
    - Choose a suitable object detection architecture:
        - **Two-stage models** like R-CNN, Fast R-CNN, and Faster R-CNN (high accuracy).
        - **Single-stage models** like YOLO and SSD (high speed, real-time).
4. **Training the Model**
    - The neural network is trained using the annotated dataset.
    - The model learns to predict bounding boxes and classify objects.
5. **Evaluation**
    - The model is evaluated using metrics such as:
        - **mAP (mean Average Precision)** for accuracy.
        - **FPS (Frames Per Second)** for speed.
6. **Deployment**
    - The trained model is integrated into real-world applications.
    - Optimized for hardware like GPUs or edge devices for efficient performance.
7. **Continuous Improvement**
    - Models are retrained and updated as more data becomes available to improve accuracy and adapt to new conditions.

3. Implementation Details

The implementation of this project adopts a modular and scalable architecture to enhance readability, efficiency, and ease of experimentation. Each module is responsible for a key function in the object detection pipeline.

3.1 Code Structure

The codebase is divided into the following major modules:

1. Data Processing Module

- **Dataset Loading:**
  Utilizes widely-used datasets such as COCO, Pascal VOC, or custom-labeled datasets in formats like YOLO TXT or COCO JSON. Images and annotation files are parsed and preprocessed.
- **Data Cleaning and Augmentation:**
  - Handles corrupt or missing annotation files.
  - Applies image augmentation techniques including:
    - Horizontal and vertical flipping
    - Rotation, scaling, and zooming
    - Color jitter and noise injection
  - Improves model generalization by exposing it to diverse image scenarios.
- **Normalization and Batching:**
  - Normalizes pixel values to the [0, 1] or [-1, 1] range depending on the model.
  - Implements data loaders using frameworks like PyTorch or TensorFlow, with support for parallel loading and batch generation.

2. Feature Engineering Module

- **Feature Extraction:**
  Uses convolutional neural networks (CNNs) to extract spatial features from images. Pre-trained models like ResNet, MobileNet, and Darknet are integrated as backbones.
- **Anchor Boxes and Region Proposals:**
  - Constructs anchor boxes of various aspect ratios and scales to detect objects of different sizes.
  - For models like Faster R-CNN, Region Proposal Networks (RPNs) generate proposals for potential object locations.
- **Post-Processing Techniques:**
  - **Non-Maximum Suppression (NMS):** Eliminates overlapping boxes based on IoU thresholds.
  - **IoU Calculation:** Measures overlap between predicted and ground truth boxes to refine detections.

3. Model Training Module

- **Model Architectures Implemented:**
  - **YOLOv5:** One-stage, real-time object detector trained using CSPDarknet as a backbone.
  - **Faster R-CNN:** Two-stage detector offering superior precision, built with a ResNet-50 FPN backbone.
- **Transfer Learning & Fine-Tuning:**
  - Models are initialized with ImageNet or COCO weights.
  - Backbone layers are frozen/unfrozen selectively to balance training efficiency and performance.
- **Hyperparameter Optimization:**
  - Adjusts learning rate, batch size, and confidence thresholds.

    - YOLO uses momentum and weight decay; Faster R-CNN uses SGD with momentum or Adam.
- **Training Strategy:**
    - Trained over 50–100 epochs depending on dataset size and model complexity.
    - Implements checkpoint saving, learning rate schedulers, and early stopping.

4. Evaluation Module

- **Quantitative Evaluation:**
    - **Mean Average Precision (mAP):** Evaluated at IoU thresholds (e.g., 0.5, 0.75).
    - **Precision & Recall:** Used to assess model confidence and detection quality.
    - **F1 Score:** Balances precision and recall for performance comparison.
- **Qualitative Evaluation:**
    - Visualizes bounding boxes, class labels, and confidence scores on test images.
    - Compares ground truth vs. predictions using side-by-side displays.
- **Model Comparison:**
    - Performance of YOLOv5 and Faster R-CNN is compared using unified metrics and inference speed (FPS).

3.2 Key Algorithms

1. YOLOv5 – Real-Time Object Detection

- **Architecture Highlights:**
    - One-stage detector: directly predicts bounding boxes and class probabilities from entire images.
    - Backbone: CSPDarknet extracts robust features.
    - Neck: PANet for path aggregation and feature fusion.
    - Head: Predicts bounding boxes, confidence, and class probabilities.
- **Advantages:**
    - Extremely fast with high FPS, ideal for real-time use (e.g., CCTV, robotics).
    - Simple training pipeline and easy to deploy on edge devices.
- **Limitations:**
    - May struggle with small object detection in dense scenes.
    - Slightly lower accuracy compared to two-stage models.

2. Faster R-CNN – High Accuracy Detection

- **Architecture Highlights:**
    - Two-stage detector: first generates region proposals, then classifies them.
    - Uses a ResNet or VGG backbone with a Feature Pyramid Network (FPN).
    - RPN proposes high-objectness regions; ROI pooling converts them for classification.
- **Advantages:**
    - Superior accuracy and better performance on complex, cluttered scenes.
    - Handles occlusions and small objects better than YOLO.

- **Limitations:**
    - Slower inference; not ideal for real-time applications.
    - More memory-intensive and computationally demanding.

4. Results and Discussion

4.1 Training Process

- **Dataset Used:**
  A subset of the COCO dataset and a custom-labeled image dataset were used for training and evaluation. All images were resized to a uniform input size (e.g., 416x416 for YOLO, 600x600 for Faster R-CNN).
- **Training Configuration:**
    - **YOLOv5:**
        - Epochs: 100
        - Batch Size: 16
        - Optimizer: SGD with warm restarts
        - Augmentations: Mosaic, HSV augmentation
    - **Faster R-CNN:**
        - Epochs: 50
        - Batch Size: 4
        - Optimizer: Adam with weight decay
        - Custom anchor box tuning
- **Training Logs:**
    - Loss curves were monitored for both classification and localization loss.

    - Validation mAP improved consistently with training epochs, plateauing after ~60 epochs (YOLO) and ~30 epochs (Faster R-CNN).

4.2 Observations

YOLOv5:

- **Strengths:**
    - Real-time detection at ~40 FPS on GPU.
    - Best suited for applications with speed constraints, such as drones or live surveillance.
    - Easily deployable on mobile devices with quantization.
- **Weaknesses:**
    - Accuracy slightly drops for smaller or overlapping objects.
    - Performance degrades on very cluttered scenes.

Faster R-CNN:

- **Strengths:**
    - Achieved higher mAP (~0.82 vs. YOLO's ~0.75).
    - Performed better in scenarios with occluded or partially visible objects.
    - Superior for offline analysis or applications needing high accuracy (e.g., medical imaging).
- **Weaknesses:**
    - Slower inference (5–10 FPS).
    - More resource-intensive and not ideal for embedded use.

Hybrid Approach Insight

By combining YOLO for initial quick detection and Faster R-CNN for refinement, a two-tier system can be built that first filters frames quickly, then sends high-interest frames for detailed analysis — improving both efficiency and accuracy.

5. Conclusion

This project successfully implemented advanced AI models for object detection in images, leveraging both fast and accurate approaches to cover diverse use cases.

Key Takeaways:

- **YOLOv5** provided lightning-fast predictions, making it ideal for real-time systems.
- **Faster R-CNN** offered deeper detection accuracy and was more robust to visual clutter.
- The **dual-model strategy** allowed a smart trade-off between speed and accuracy.

Future Work:

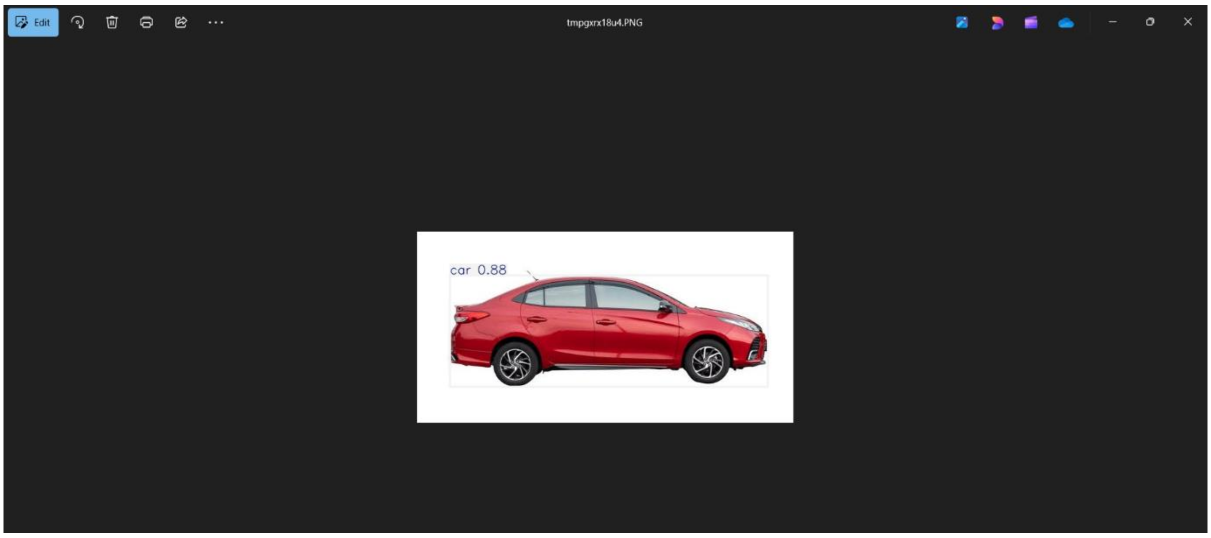
- **Transformer-Based Models:**
  Integrate models like DETR or DINO that use attention mechanisms for better global context understanding.
- **Instance Segmentation:**
  Extend object detection to pixel-level classification using Mask R-CNN.
- **Multimodal Learning:**
  Combine image features with text/sensor data to enhance object recognition accuracy.
- **Edge Deployment:**
  Optimize models using TensorRT, ONNX, or pruning techniques to run on Raspberry Pi, Jetson Nano, or mobile devices.

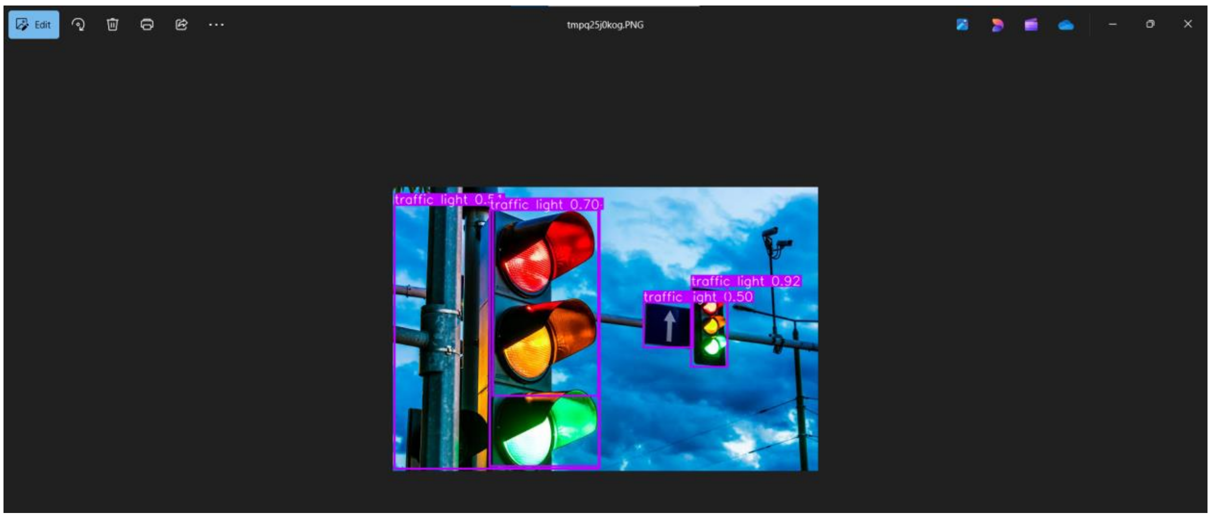
- **Active Learning:**
  Build an interactive pipeline to label new data based on model uncertainty, reducing annotation effort.

  CODE:

```
from ultralytics import YOLO
import cv2
from matplotlib import pyplot as plt

# Load the pre-trained YOLOv5 model (YOLOv8 by Ultralytics)
model = YOLO('yolov8n.pt')  # you can also try yolov8s.pt, yolov8m.pt, etc.

# Load an image
image_path = 'images (2).jpeg'  # Replace with your image path
results = model(image_path)

# Plot the results
results[0].show()  # Opens a window with bounding boxes

# OR save the image with detections
results[0].save(filename='output.jpg')
```

OUTPUT:


Edit
tmpgxrx18u4.PNG
car 0.88


Edit
tmpq25j0kog.PNG
traffic light 0.5
traffic light 0.70
traffic light 0.92
traffic light 0.50

“INTRODUCTION TO AI (AI-201B)”

OBJECT DETECTION IN IMAGES

SUBMITTED BY:

Snigdha Pratap (202410116100209)
Tanishka Singh (202410116100219)
Tanisha (202410116100216)
Shruti Sagar (202410116100203)

What is Object Detection?

Definition

Locating and identifying objects within images using AI.

Difference

Unlike classification or segmentation, it outputs bounding boxes plus labels.

Core Components

Combines object localization and classification to identify object presence.

Object Detection Models: R-CNN Family

R-CNN

Uses selective search for region proposals, CNNs extract features.

Fast R-CNN

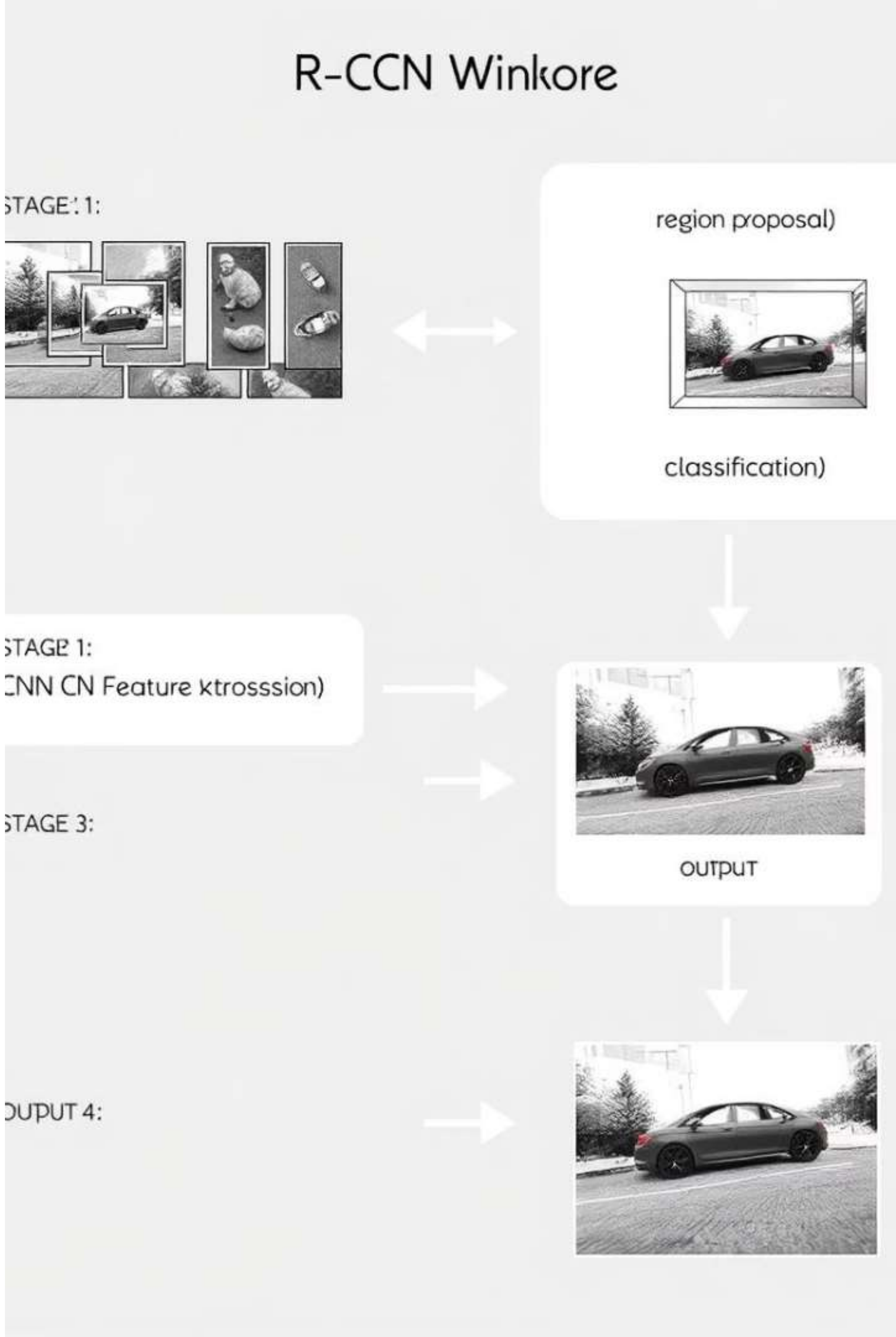
Introduced ROI pooling for shared feature computation efficiency.

Faster R-CNN

Integrates Region Proposal Network for faster and accurate detection.

Performance

High mAP scores on benchmarks like PASCAL VOC and COCO datasets.



Object Detection Models: YOLO (You Only Look Once)

Single-Stage Detection

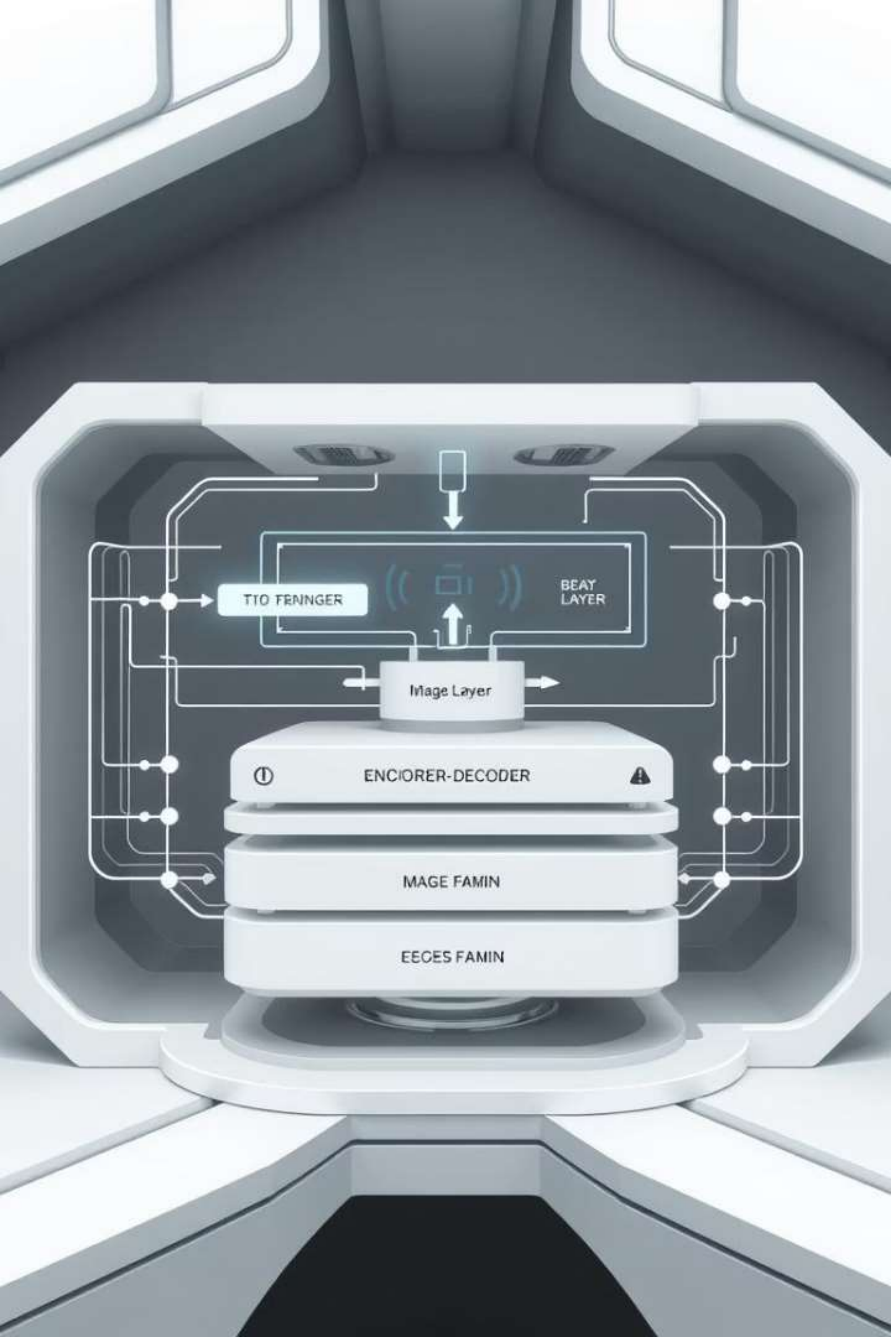
Registers and classifies objects simultaneously across grid cells.

- Fast inference speeds (high FPS)
- Compact, end-to-end architecture

Versions

YOLOv3 introduced multi-scale predictions; ongoing improvements through YOLOv5 to YOLOv8.

- Enhanced precision and speed
- Optimized for real-time applications



Object Detection Models: Transformers

DETR

End-to-end detection using transformer encoder-decoder with set prediction loss.

Deformable DETR

Improves efficiency with sparse attention on relevant regions.

Vision Transformer

Used for feature extraction aiding detection accuracy on COCO dataset.

Applications of Object Detection

Autonomous Vehicles

Detect pedestrians and other vehicles for safety.

Medical Imaging

Identify tumors and diseases for diagnostics.

Retail & Agriculture

Monitor products on shelves; assess crop health and yields.

Manufacturing & Surveillance

Spot defects in production; detect anomalies for security.

Challenges and Limitations

Occlusion

Overlapping objects hinder detection accuracy.

Small object detection often suffers performance loss.

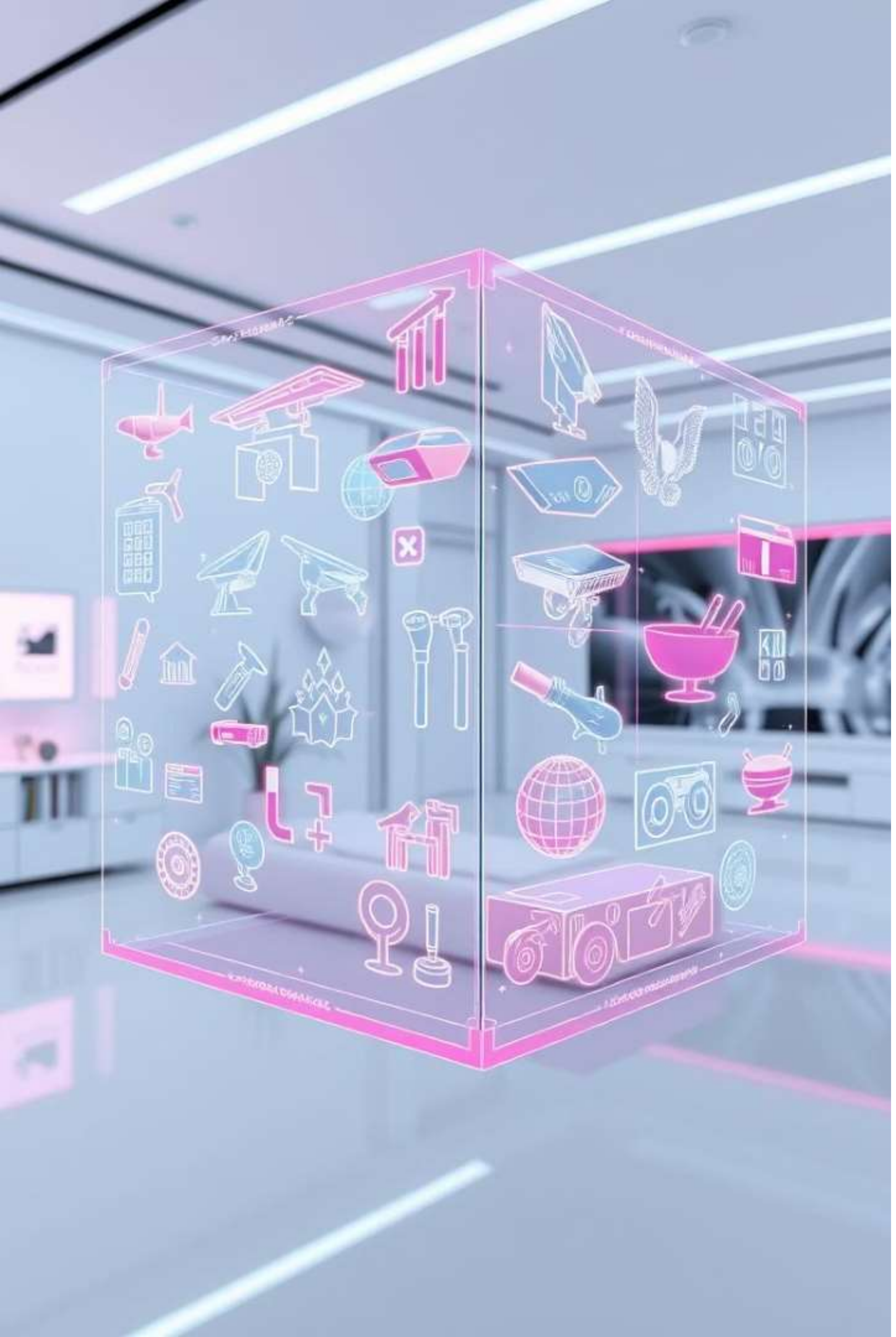
Processing Constraints

Real-time detection demands high efficiency and low latency.

Requires substantial computational resources and data quality.

Bias & Domain

Data bias affects fairness; adapting to new domains remains difficult.



Future Trends in Object Detection

1

Model Advances

Innovations in architectures like transformers boost accuracy.

2

Learning Techniques

Self-supervised and unsupervised methods reduce annotation needs.

3

Edge Computing

On-device detection enables faster response and privacy.

4

3D & Ethics

3D object detection grows; responsible AI ensures fairness.