Deep Learning Models: Modern Approaches

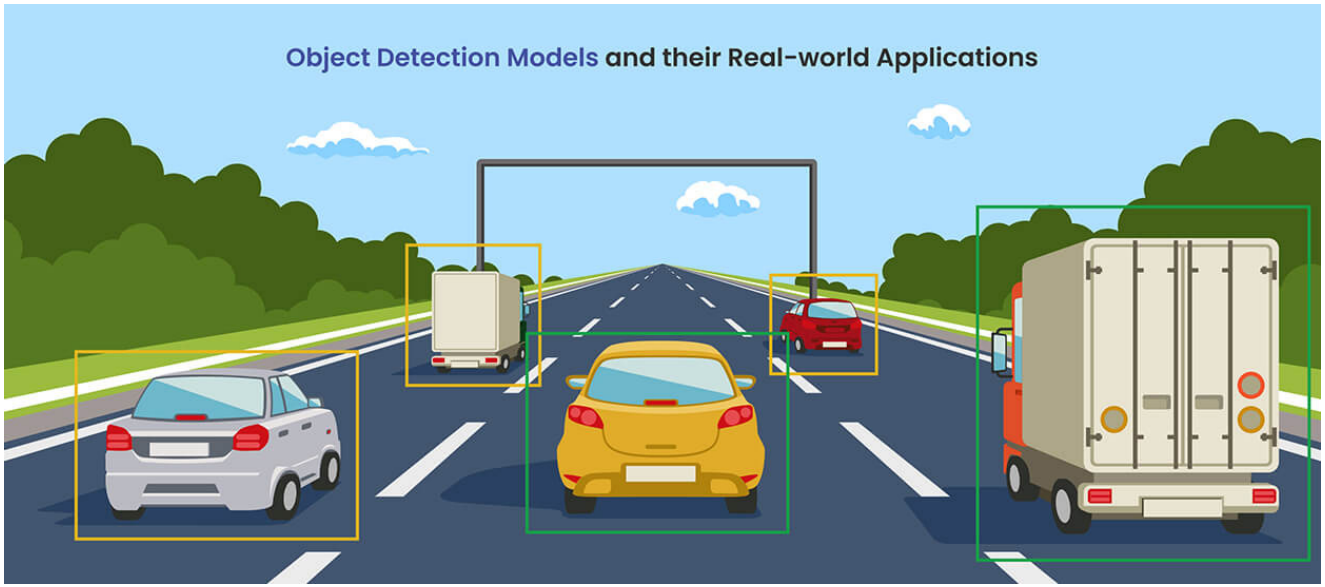
Michelle Marchesini Vanegas

Earlham College

April 18th

Introduction

Deep Learning uses multi-layer neural networks to automatically learn features from images. This has revolutionized computer vision, achieving high accuracy on tasks that were hard for earlier methods. Deep learning models now excel at image classification, object detection, and segmentation. They learn to recognize patterns (like edges, shapes, textures) directly from data, eliminating the need for manual feature design. In short, deep learning provides the foundation that makes modern object detection possible.



Object Detection vs. Image Classification

- **Image Classification:** Predicts a single label for a whole image. For example, a classifier might label an image as "cat" or "dog" based on the main subject. It does not indicate where in the image that object is.
- **Object Detection:** Goes a step further by not only classifying objects but also localizing them. It outputs multiple labels and draws bounding boxes around each object it finds. For instance, in a street photo, detection can find all cars, pedestrians, etc., and mark their positions.
- **Classification** = "What is in this image?"
- **Detection** = "What is in this image and where are they?" (usually via bounding boxes).

Key Challenges in Object Detection

- **Scale Variations:** Objects can appear in many sizes (a distant person vs. a close-up face). A detector must recognize objects at vastly different scales. This requires multi-scale feature processing so that both small and large objects are detected.
- **Occlusion and Clutter:** Often objects are partially hidden behind others or against busy backgrounds. Occlusion (one object blocking another) causes loss of visual information, making detection ambiguous. Detecting each object in a crowded or cluttered scene is much harder than in a clean image.

Key Challenges in Object Detection

- **Pose and Viewpoint Changes:** Objects can be viewed from different angles or orientations (imagine a car front-on vs. side view). Such pose variations mean the detector must generalize to objects looking very different
- **Real-Time Processing:** Some applications (e.g. self-driving cars or live video analysis) require fast detection. The challenge is to make models accurate and efficient enough to run in real time. There's often a trade-off between speed and accuracy.
- **Data and Class Variability:** Object detectors usually need a lot of labeled examples of each object class. They can struggle with rare classes or unexpected images. If the test scenario differs from training data (domain shift), performance may drop. Ensuring robustness in all conditions remains difficult.

Components of Object Detection Models

- **Backbone CNN:** Extracts high-level feature maps from the input image
- **Detection Head:** Takes feature maps and predicts bounding boxes + class scores
- **Region Proposal Network (RPN) (two-stage only):** Quickly suggests candidate object regions
- **Single-Shot Predictor (one-stage only):** Directly regresses boxes and labels in one pass

One-Stage vs Two-Stage Architectures

One-Stage Detectors

- Single network predicts boxes + classes all at once
- real-time speed, simpler pipeline
- historically a slight accuracy trade-off (narrowed in modern versions)

Two-Stage Detectors

- Stage 1: generate region proposals
- Stage 2: classify and refine those regions
- higher accuracy, especially on small/overlapping objects
- slower inference (extra processing)

Introduction to Faster R-CNN

Faster R-CNN is a classic two-stage deep learning detector that became a benchmark for accuracy. Introduced in 2015 by researchers at Microsoft, it built upon earlier R-CNN models and significantly sped up detection by integrating the proposal step into the network.

The original R-CNN (2014) and Fast R-CNN (2015) methods showed that using CNNs for object detection could work well, but they were slow because they relied on external region proposals. Faster R-CNN's key innovation was the Region Proposal Network (RPN), which generates proposals inside the CNN itself, making the system nearly end-to-end trainable. As the name suggests, it's "faster" than its predecessors thanks to this integration.

Faster R-CNN

- **Feature Extraction:** CNN backbone produces a feature map
- **Region Proposal (RPN):** Slides over features, outputs candidate boxes + objectness scores
- **RoI Pooling/Align:** Extracts fixed-size feature regions for each proposal
- **Detection Head:** Classifies each region and refines box coordinates
- **Non-Max Suppression:** Removes overlapping/duplicate detections

Faster R-CNN

- **Unified Architecture:** Faster R-CNN still follows the two-stage paradigm: it has an RPN module to propose regions and then a second stage that classifies these regions and fine-tunes the bounding boxes. Both stages share the same backbone CNN features, which makes it efficient. This model demonstrated that high-quality object detection could be done in near real-time (with a good GPU), and it became the foundation for many subsequent advances (like Mask R-CNN for segmentation).

- **Usage:** Faster R-CNN is known for its accuracy. It often serves as a strong baseline in research and is used in applications where detecting small or subtle objects is important (e.g. medical images or detailed scene analysis), despite not being the fastest model.

YOLO

YOLO (You Only Look Once) is a family of one-stage object detectors known for extreme speed. Its name highlights that the model looks at the image only once – i.e., it performs detection in a single evaluation of a neural network, rather than multiple passes.

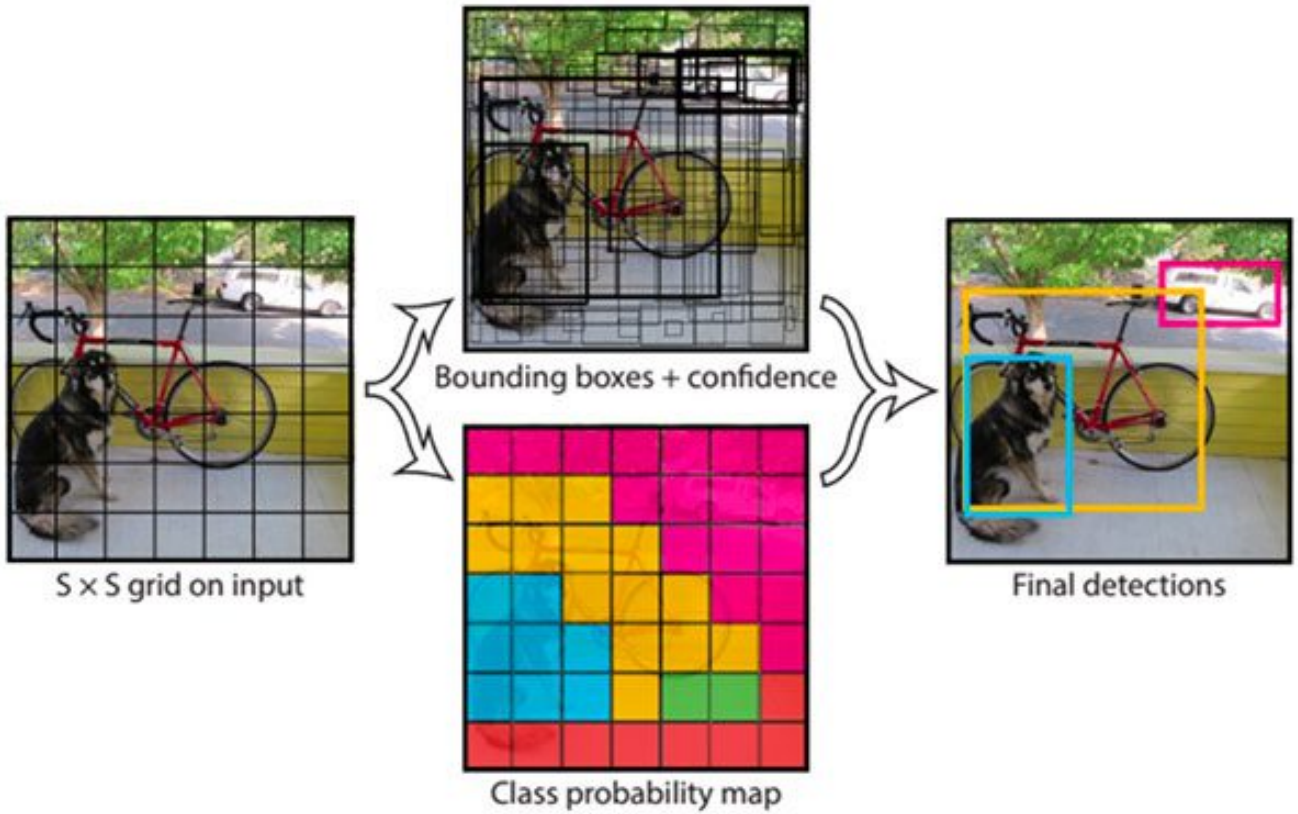


Image source: Medium

YOLO

Real-Time Detection: YOLO's big claim to fame was real-time performance. The original YOLOv1 could process images at about 45 frames per second, and a simplified version ("Fast YOLO") reached an astounding 155 FPS. This was a breakthrough, as many earlier detectors were an order of magnitude slower. YOLO demonstrated that deep learning could be used for object detection in live video and high-frame-rate applications without sacrificing too much accuracy.

YOLO

- **Single Unified Network:** Unlike two-stage methods, YOLO frames detection as a single regression problem. It uses one convolutional network to directly predict bounding boxes and class probabilities from the full image. This end-to-end approach means all parts of the model are optimized together for the detection task. Because the entire pipeline is one network, the model learns to contextualize object information globally (considering the whole image at once).
- **Trade-offs:** YOLOv1 was extremely fast but initially slightly less accurate than slower two-stage detectors; it tended to make more localization errors (boxes a bit off) though it made fewer false positives on background. However, subsequent YOLO versions have greatly improved accuracy. The simplicity and speed of YOLO have made it one of the most popular detection frameworks. It's widely used in applications that need speed (e.g. real-time video analysis, embedded systems)

SSD

SSD (Single Shot Detector) is another popular one-stage object detection model. "Single Shot" means it, like YOLO, does all the detection work in one pass of the network (no separate proposal stage). The term MultiBox refers to the approach of predicting multiple bounding boxes.

SSD uses a convolutional neural network that produces multiple feature maps at different scales and each of those maps predicts bounding boxes for objects. It does not have a region proposal network; it directly predicts the coordinates and classes of bounding boxes from feature maps in one go. The model is typically composed of:

1. A backbone CNN (e.g., VGG-16) that provides base feature maps.
2. Additional layers that progressively decrease in size, each detecting objects at a different scale. For example, early layers with high resolution detect small objects, later layers with coarser resolution detect larger objects.

Key Use Cases

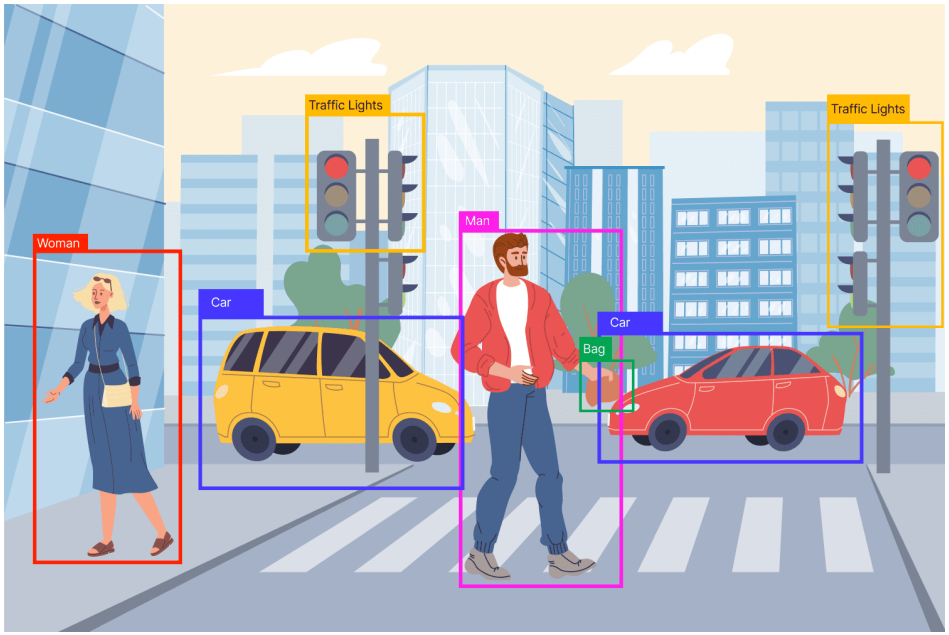
- **Autonomous Driving:** detect cars, pedestrians, lights in real time
- **Surveillance:** spot people or objects of interest on CCTV
- **Healthcare:** locate tumors or anomalies in medical scans
- **Manufacturing QC:** find defects or missing parts on assembly lines
- **Retail and Smart Cities:** inventory on shelves, traffic monitoring
- **Robotics/Drones:** obstacle avoidance; search rescue object tracking

Limitations and Challenges

- **Small Objects:** few pixels $\rightarrow$ low detection accuracy
- **Occlusion/Overlap:** hidden or crowded items get missed
- False Positives: background patterns can be misdetected
- **Pose/Lighting Variations:** unusual angles or poor light harm performance
- **Compute Constraints:** heavy models struggle on mobile/edge devices
- **Domain Shift:** models need retraining for new environments.

Recap

- Two-stage (Faster R-CNN) = accuracy
- One-stage (YOLO, SSD) = speed
- Trade-offs: speed vs. precision



How to Experiment

- Run a pre-trained YOLO or Faster R-CNN on sample images
- Explore datasets: COCO, Pascal VOC; learn annotation formats
- Dig deeper: read key papers; experiment with training on a small custom dataset
- Advance: look into Mask R-CNN (segmentation) or DETR (transformer-based detection).