Object Detection Models: Balancing Speed, Accuracy and Efficiency

Sage Elliott

AI Engineer

Union.ai

UNION

Object Detection 101: The Basics

- What is object detection?
  - Detects and classifies multiple objects in an image or video by drawing bounding boxes around them and assigning labels.
- What data is needed?
  - A dataset of labeled images, where each object is annotated with a bounding box and a class label.

Why Object Detection Matters

Real-world use cases (sample):

- Autonomous Vehicles & Robotics
- Security
- Mobile Apps

Real-world systems must balance:

- Speed — fast enough for real-time use (if needed)
- Accuracy — reliable predictions under noise & variation
- Efficiency — low compute cost, especially on edge devices

Evolution of Object Detection

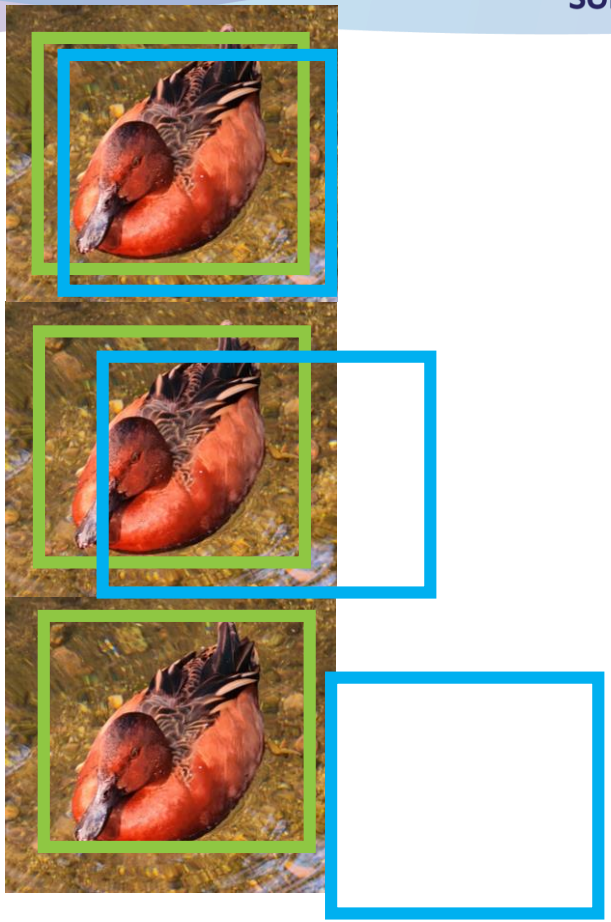
- **Traditional CV:** Handcrafted features (edge detectors, HOG) + classical classifiers (SVMs, decision trees)
- **Deep learning:** Learns features and patterns directly from raw image data using convolutional neural networks (CNNs)
  - **Transformers:** Recent models (DETR, DINO) use attention mechanisms for end-to-end object detection

Measuring Object Detection

Measuring object detection is more complex than classification.

- Intersection over union (IoU)
- Average precision (AP) on a threshold
- Mean average precision (mAP) | .50 - .95
- mAP on Coco dataset a common metric

mAP may seem low on models. But they can have very high AP on higher thresholds.



Major Architectures for Deep Learning Object Detection

- Two-stage detectors (Faster R-CNN, DETR)
  - High accuracy
  - Slower, more compute-intensive
- One-stage detectors (YOLO, SSD)
  - Fast, real-time friendly
  - Historically less accurate (but improving)

Two-Stage Models: Overview

- **Stage 1: Region proposal**
  - Identify potential object locations (using Region Proposal Network (RPN)
- **Stage 2: Classification + box refinement**
  - Predict class label and refine bounding box coordinates for each proposal
- High accuracy, especially in complex scenes with small or overlapping objects
- Slower inference time, higher computational cost

Non-Max Suppression (NMS)
With confidence threshold

Spotlight: RCNN / Faster R-CNN

- Uses a CNN backbone (ResNet) for feature extraction
- Generates region proposals with an RPN
- Classifies each region and refines bounding box coordinate
- Higher-accuracy applications where speed is less critical

One-Stage Models: Overview

- Detection and classification happen in one forward pass
- Predict class labels and bounding boxes directly from feature maps
- Designed for speed and efficiency, suitable for real-time applications
- Typically faster, but may trade off some accuracy in complex scenes

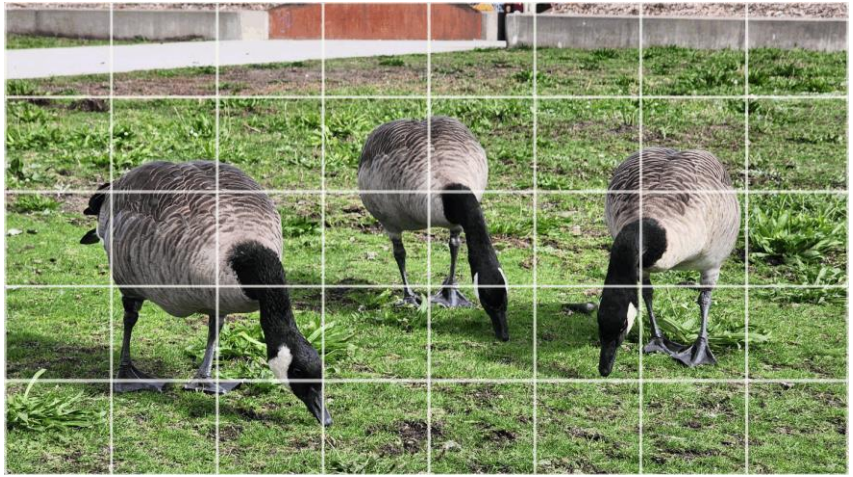
Spotlight: SSD (Single Shot Detector)

- Uses anchor boxes to detect objects and class labels in a single forward pass
- Balances speed and accuracy—faster than two-stage models like Faster R-CNN
- Well-suited for edge deployments where performance and reliability matter
- Leverages multi-scale feature maps to detect objects of varied sizes effectively

Default SSD has 8732 Boxes

Spotlight: YOLO (You Only Look Once)

- Predicts bounding boxes and class labels in one forward pass
- Evolved from YOLOv1 to YOLOv9 with major improvements in speed and accuracy
- Extremely fast and efficient—ideal for real-time applications and edge devices
- Strong balance of speed and decent accuracy across a wide range of tasks



Spotlight: DETR (DEtection TRansformer)

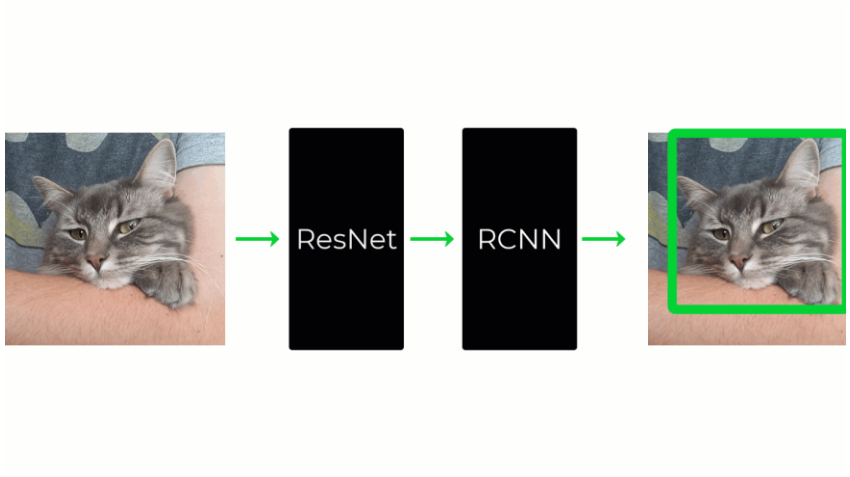
- Uses a CNN backbone to extract image features
- Applies a Transformer encoder-decoder to model global relationships
- Predicts a fixed set of objects using learned object queries
- Complex visual scenes or vision-language tasks where global reasoning and a simplified detection pipeline are beneficial

Images from DETR Paper

Attention-based model – we can visualize what the network is looking at to make predictions.

Classification "Backbones" for Object Detection

- Most object detection models use a classification backbone to extract visual features
- These can be swapped to optimize for speed, accuracy, or efficiency
  - MobileNet
  - EfficientNet
  - ResNet50
  - Swin Transformers



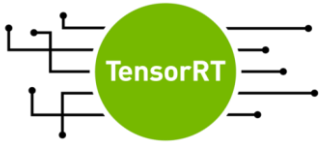
How to Choose the Right Model

- What's your available hardware (CPU, GPU, edge device)?
- How important is accuracy vs. inference speed?
- Do you need real-time performance, or can you afford latency?
- Are you detecting simple objects or complex scenes?
- Experiment

Other Deployment Considerations

Beyond model choice, optimize for speed, size, and deployment environment:

- Model quantization & pruning
- Formats: ONNX, TensorRT, LiteRT (TFlite)
- Serving frameworks: TorchServe, OpenVINO, Ray, Union.ai



Summary & Final Thoughts

- Trade-offs are everywhere— there's no one-size-fits-all model
- Know your constraints — **or** figure them out fast
- Optimize for real-world use — not just benchmarks
- Build an efficient pipeline — make it easy to test and compare models quickly

Resources

R-CNN

https://arxiv.org/abs/1506.01497v3

DETR

https://arxiv.org/abs/2005.12872v3

YOLO

https://arxiv.org/abs/1506.02640

SSD

https://arxiv.org/abs/1512.02325

Get these links & examples:

https://github.com/sagecodes

Thank You and Stay Connected!

- Thank you!
- I help build efficient AI pipelines at Union.ai (including computer vision)
- Check out Union.ai and RSVP for my upcoming object detection workshop!