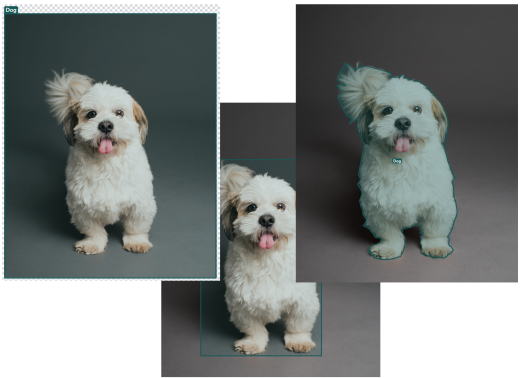
Object Detection

Naeemullah Khan
naeemullah.khan@kaust.edu.sa

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

LMH
Lady Margaret Hall

July 25, 2025


Dog
Dog

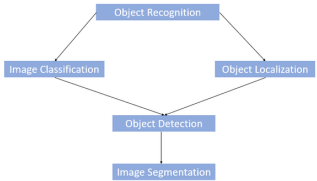

Object Recognition
Image Classification
Object Localization
Object Detection
Image Segmentation

Table of Contents

Table of Contents (cont.)

⇒ Learning Outcomes

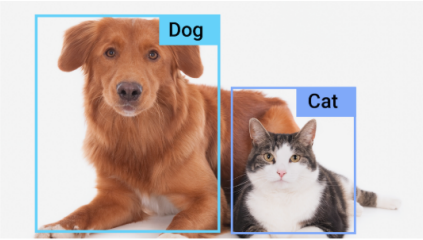
By the end of this session, you will be able to:

- Know why object detection matters and where it's used
- Understand how the R-CNN family (R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN) and YOLO work
- See how YOLO evolved—what each version solved
- Recognise current limitations and future innovation paths

Object Detection: **Introduction**

Object Detection

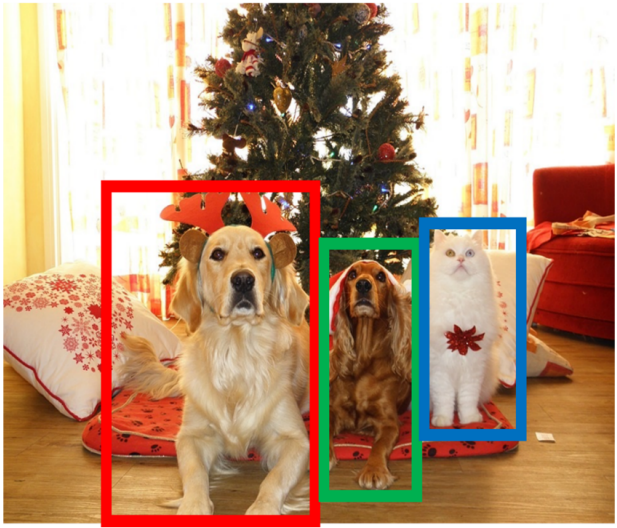
- Object detection involves:
  - localising, and
  - classifying objects within an image or video
- It aims to:
  - identify specific objects of interest, and
  - provide their bounding boxes
- Crucial for tasks like:
  - object tracking, and
  - scene understanding



Object Detection (cont.)

- **Input:** Single RGB Image
- **Output:** A set of detected objects. For each object predict:
  - Category label (from fixed, known set of categories)
  - Bounding box (four numbers: x, y, width, height)

Object Detection (cont.)



Object Detection – Components

Object detection comprises key components such as:

- Region Proposal Networks (RPNs) for generating potential object proposals,
- Feature extraction networks for analyzing proposals, and
- Object classification networks for assigning class labels.

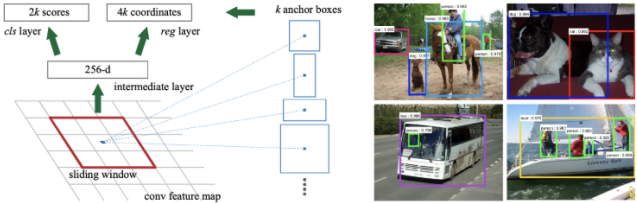


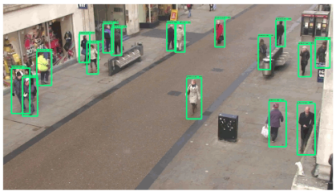
Figure 3: **Left**: Region Proposal Network (RPN). **Right**: Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.

From the paper: **Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks**

Object Detection – Application

Applications in various fields, including:

- Video surveillance for identifying and tracking individuals or objects
- Agriculture for crop monitoring and pest detection, and
- Retail analytics for customer behavior analysis.



(a) Flower sample (b) Seedpod sample

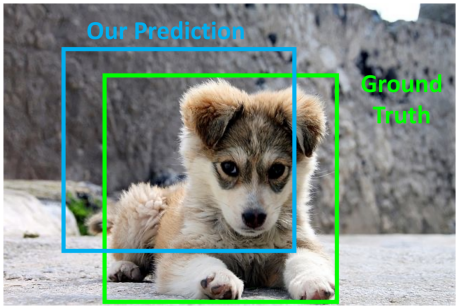
Object Detection: Challenges

- **Multiple outputs:** Need to output variable numbers of objects per image
- **Multiple types of output:** Need to predict "what" (category label) as well as "where" (bounding box)
- **Large images:** Classification works at 224x224; need higher resolution for detection, often $\sim 800 \times 600$

Object Detection: **Intersection over Union (IoU)**

Comparing Boxes: Intersection over Union (IoU)

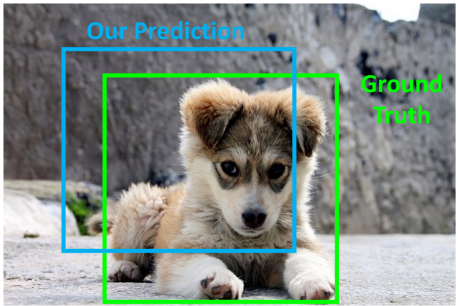
- How can we compare our prediction to the ground-truth box?



Comparing Boxes: Intersection over Union (IoU)

- How can we compare our prediction to the ground-truth box?
- **Intersection over Union** (IoU) (Also called "Jaccard similarity" or "Jaccard index"):

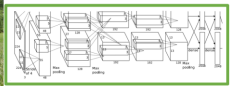
$$\frac{\text{Area of Intersection}}{\text{Area of Union}}$$



Object Detection: **Detecting a Single Object**

Detecting a Single Object

Often pretrained on ImageNet (Transfer learning)



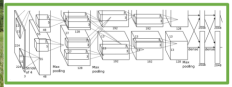
This image is CC0 public domain

Vector: 4096

Treat localization as a regression problem!

Detecting a Single Object (cont.)

This image is CC0 public domain



Often pretrained on ImageNet (Transfer learning)

Vector: 4096

Fully Connected: 4096 to 1000

"What"

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Correct label: Cat

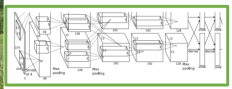
Softmax Loss

Treat localization as a regression problem!

Detecting a Single Object (cont.)

This image is CC0 public domain

Often pretrained on ImageNet (Transfer learning)



Vector: 4096

Fully Connected: 4096 to 1000

"What"

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Correct label: Cat

Fully Connected: 4096 to 4

"Where"

Box Coordinates
(x, y, w, h)

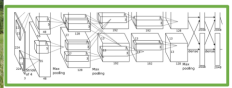
L2 Loss

Correct box: (x', y', w', h')

Treat localization as a regression problem!

Detecting a Single Object (cont.)

This image is CC0 public domain



Often pretrained on ImageNet (Transfer learning)

Treat localization as a regression problem!

Vector: 4096

Fully Connected: 4096 to 1000

Fully Connected: 4096 to 4

"What"

"Where"

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Box Coordinates
(x, y, w, h)

Correct label:
Cat

Softmax Loss

Multitask Loss

Weighted Sum → **Loss**

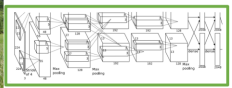
$$L = L_{cls} + \lambda L_{reg}$$

L2 Loss

Correct box:
(x', y', w', h')

Detecting a Single Object (cont.)

This image is CC0 public domain



Often pretrained on ImageNet (Transfer learning)

Treat localization as a regression problem!

Vector: 4096

Fully Connected: 4096 to 1000

Fully Connected: 4096 to 4

"What"

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Box Coordinates
(x, y, w, h)

"Where"

Correct label:
Cat

Softmax Loss

Multitask Loss

Weighted Sum → **Loss**

$$L = L_{cls} + \lambda L_{reg}$$

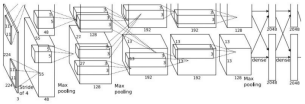
L2 Loss

Correct box:
(x', y', w', h')

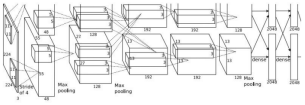
Problem: Images can have more than one object!

Object Detection: **Detecting Multiple Objects**

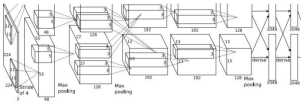
Multiple Objects



CAT: (x, y, w, h)



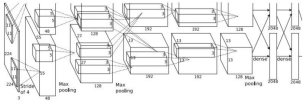
DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)



DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

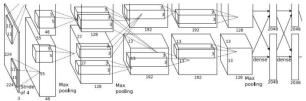
Multiple Objects (cont.)

Each image needs a different number of outputs!



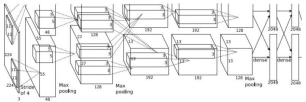
CAT: (x, y, w, h)

4 numbers



DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

12 numbers



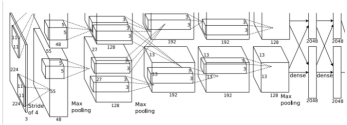
DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
….

Many numbers!

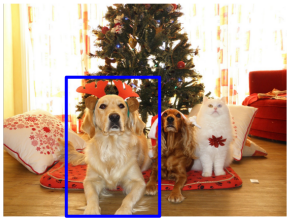
Object Detection: **Detecting Multiple Objects: Sliding Window**

Detecting Multiple Objects: Sliding Window

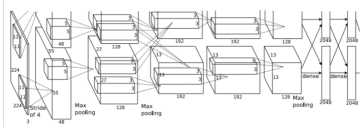
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? NO
Background? YES



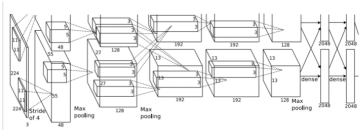
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

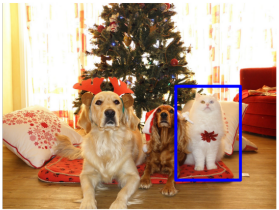
Detecting Multiple Objects: Sliding Window (cont.)

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? YES
Background? NO

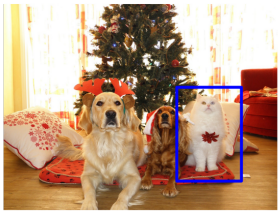
Detecting Multiple Objects: Sliding Window (cont.)



Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

Question: How many possible boxes are there in an image of size H x W?

Detecting Multiple Objects: Sliding Window (cont.)

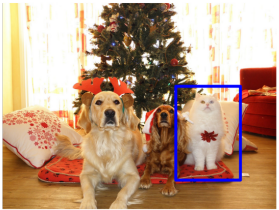


Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

Question: How many possible boxes are there in an image of size H x W?

Consider a box of size h x w:
Possible x positions: W – w + 1
Possible y positions: H – h + 1
Possible positions:
(W – w + 1) * (H – h + 1)

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Question: How many possible boxes are there in an image of size H x W?

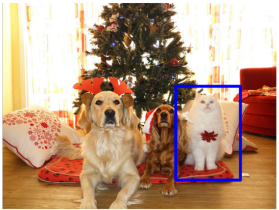
Consider a box of size h x w:
Possible x positions: W – w + 1
Possible y positions: H – h + 1
Possible positions:
(W – w + 1) * (H – h + 1)

Total possible boxes:

$$\sum_{h=1}^{H}\sum_{w=1}^{W}(W-w+1)(H-h+1)$$

$$=\frac{H(H+1)}{2}\frac{W(W+1)}{2}$$

Detecting Multiple Objects: Sliding Window (cont.)



Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

800 x 600 image has ~58M boxes! No way we can evaluate them all

Question: How many possible boxes are there in an image of size H x W?

Consider a box of size h x w:
Possible x positions: W – w + 1
Possible y positions: H – h + 1
Possible positions:
(W – w + 1) * (H – h + 1)

Total possible boxes:

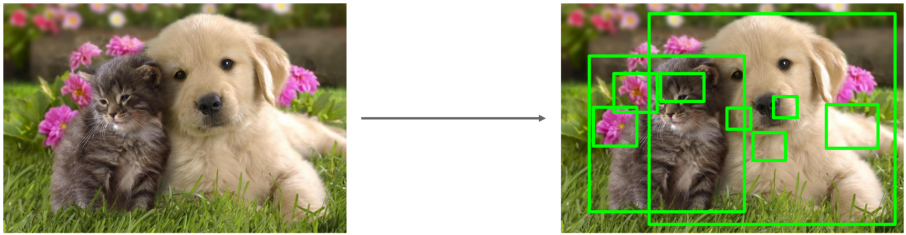
$$\sum_{h=1}^{H}\sum_{w=1}^{W}(W-w+1)(H-h+1)$$

$$=\frac{H(H+1)}{2}\frac{W(W+1)}{2}$$

Object Detection: **Regions Proposal**

Regions Proposal

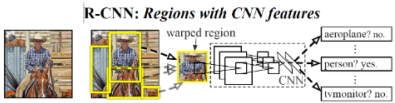
- Find a small set of boxes that are likely to cover all objects
- Often based on heuristics: e.g. look for "blob-like" image regions
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU



Object Detection: **Most Popular Methods**

Object Detection – Most Popular Methods

- R-CNN (2014)
- Fast RCNN (2015)
- Faster RCNN (2015)
- YOLO (2016)
- Mask RCNN (2017)



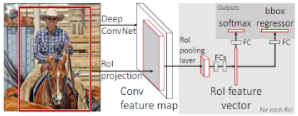


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

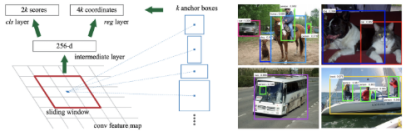


Figure 3: **Left**: Region Proposal Network (RPN). **Right**: Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.

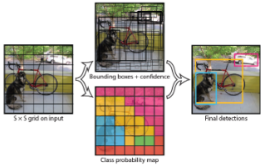
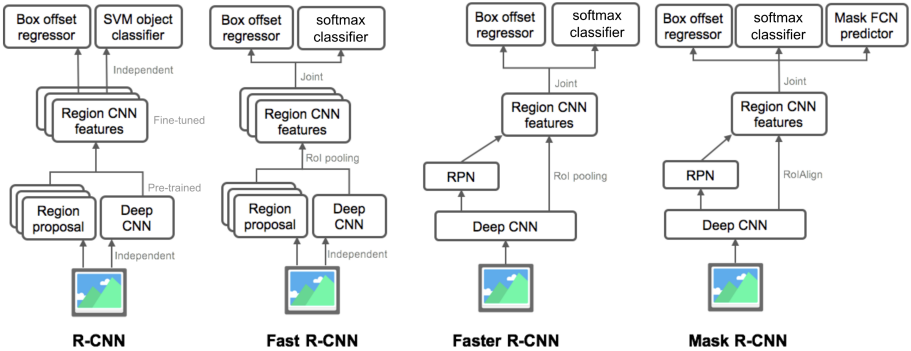


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

Object Detection: **R-CNN Family**

R-CNN Family


Box offset regressor
SVM object classifier
Independent
Region CNN features
Fine-tuned
Pre-trained
Region proposal
Deep CNN
Independent
R-CNN
Box offset regressor
softmax classifier
Joint
Region CNN features
RoI pooling
Region proposal
Deep CNN
Independent
Fast R-CNN
Box offset regressor
softmax classifier
Joint
Region CNN features
RPN
RoI pooling
Deep CNN
Faster R-CNN
Box offset regressor
softmax classifier
Mask FCN predictor
Joint
Region CNN features
RPN
RoIAlign
Deep CNN
Mask R-CNN

R-CNN Family (cont.)

1. **R-CNN**: Uses selective search to propose regions, then classifies each region using a CNN.

2. **Fast R-CNN**: Improves R-CNN by sharing computation across regions, using a single CNN for the entire image.

3. **Faster R-CNN**: Introduces a Region Proposal Network (RPN) to generate region proposals, making it faster and more efficient.

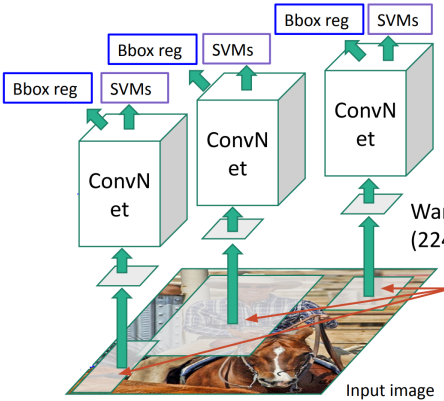
4. **Mask R-CNN**: Extends Faster R-CNN to also predict segmentation masks for each detected object.

Object Detection: **Regions with Convolutional Neural Networks (R-CNN)**

- **R-CNN** stands for *Regions with Convolutional Neural Networks*.
- Proposed by Ross Girshick et al. in 2014.
- Combines region proposals with CNNs for object detection.
- Steps:
  1. Generate region proposals (e.g., using Selective Search).
  2. Warp each region and extract features using a CNN.
  3. Classify each region and refine bounding boxes.
- Achieved significant improvements over traditional methods (e.g., sliding windows, hand-crafted features).

R-CNN (cont.)

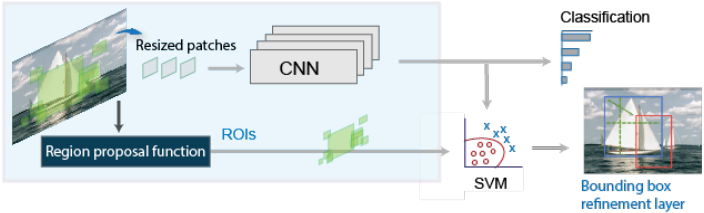
Predict “corrections” to the RoI: 4 numbers: (dx, dy, dw, dh)



Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

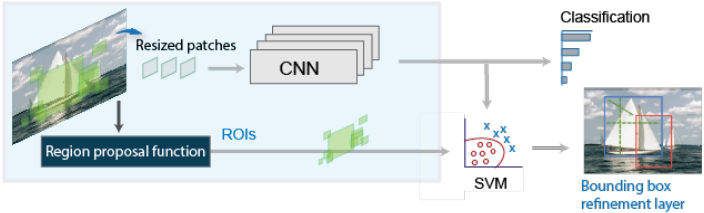
R-CNN follows a **three-step pipeline** for object detection:

1. **Region Proposal (Selective Search)**
   - Instead of using a brute-force sliding window approach, R-CNN applies **Selective Search**, an algorithm that identifies around **2000 candidate object regions** (called region proposals).
   - This significantly reduces the number of regions to process, making the model more efficient.



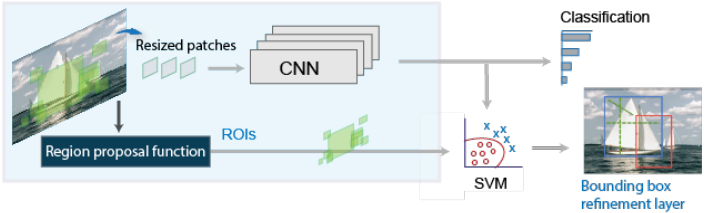
2. Feature Extraction with CNN

- Each region proposal is cropped from the image and resized to a fixed size (e.g., 224x224 pixels).
- The cropped regions are passed through a **pre-trained CNN** (like AlexNet or VGG16) to extract deep features.
- These features are then stored for classification.

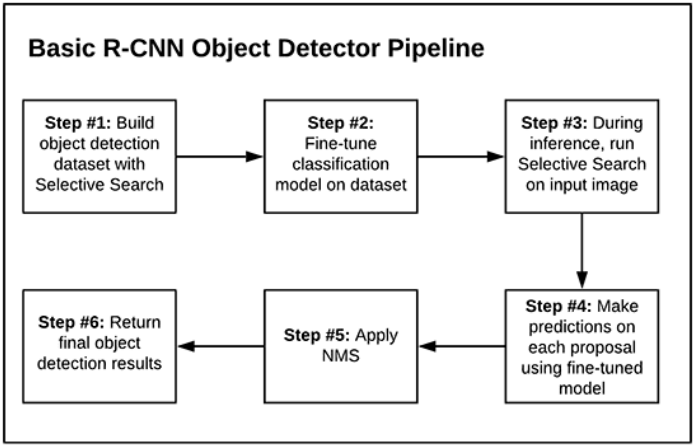


3. Object Classification & Bounding Box Refinement

- The extracted features are passed through a **Support Vector Machine** (SVM) classifier to determine the object category.
- A **bounding box regression** model is used to fine-tune the position of the detected objects.

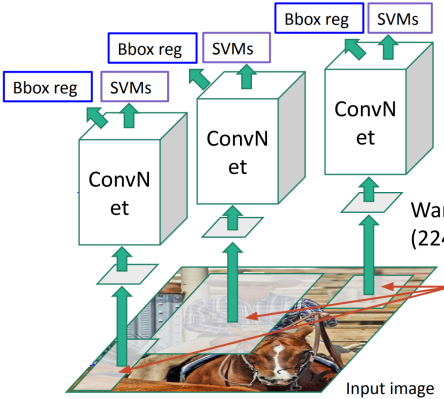


R-CNN (cont.)


Basic R-CNN Object Detector Pipeline
Step #1: Build object detection dataset with Selective Search
Step #2: Fine-tune classification model on dataset
Step #3: During inference, run Selective Search on input image
Step #4: Make predictions on each proposal using fine-tuned model
Step #5: Apply NMS
Step #6: Return final object detection results

R-CNN: Problem

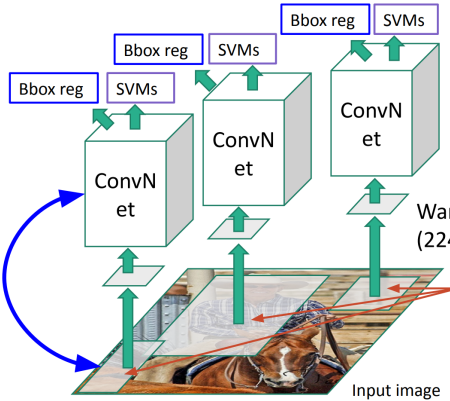
Predict "corrections" to the RoI: 4 numbers: (dx, dy, dw, dh)



Problem: Very slow! Need to do ~2k independent forward passes for each image!

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN: Problem (cont.)



Problem: Very slow! Need to do ~2k independent forward passes for each image!

Idea: Pass the image through convnet before cropping! Crop the conv feature instead!

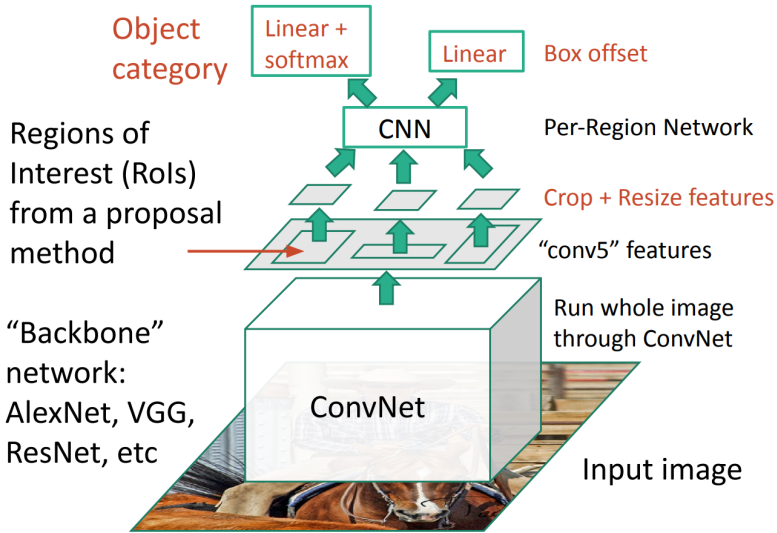
Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Evolution of R-CNN

- **R-CNN** inspired a series of improved models:
  - **Fast R-CNN**: Performs feature extraction for all region proposals in a single forward pass, greatly increasing speed and efficiency.
  - **Faster R-CNN**: Introduces a **Region Proposal Network (RPN)** to generate region proposals, making the process end-to-end and even faster.
  - **Mask R-CNN**: Extends Faster R-CNN by adding a branch for predicting segmentation masks, enabling instance segmentation.
- These advancements have made deep learning-based object detection both faster and more accurate.

Object Detection: **Fast R-CNN**

Fast R-CNN



Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Object Detection: **Faster R-CNN**

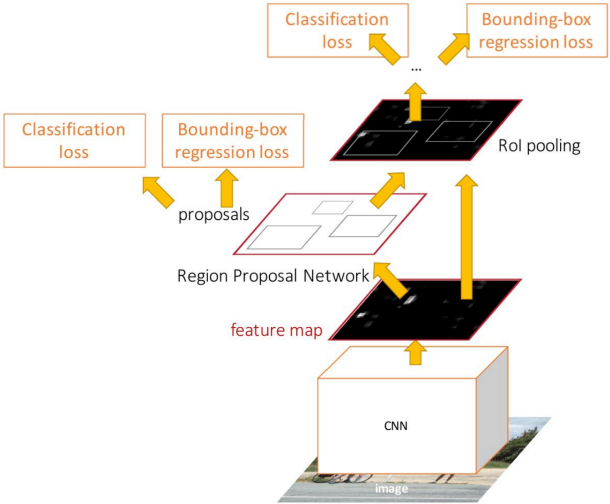
Faster R-CNN

- Make CNN do proposals!
- Insert Region Proposal Network (RPN) to predict proposals from features

Faster R-CNN

- Make CNN do proposals!
- Insert Region Proposal Network (RPN) to predict proposals from features
- Jointly train on 4 losses:
  - **RPN classification:** anchor box is object / not an object
  - **RPN regression:** predict transform from anchor box to proposal box
  - **Object classification:** classify proposals as background / object class
  - **Object regression:** predict transform from proposal box to object box

Faster R-CNN


Classification loss
Bounding-box regression loss
...
Classification loss
Bounding-box regression loss
RoI pooling
proposals
Region Proposal Network
feature map
CNN
image

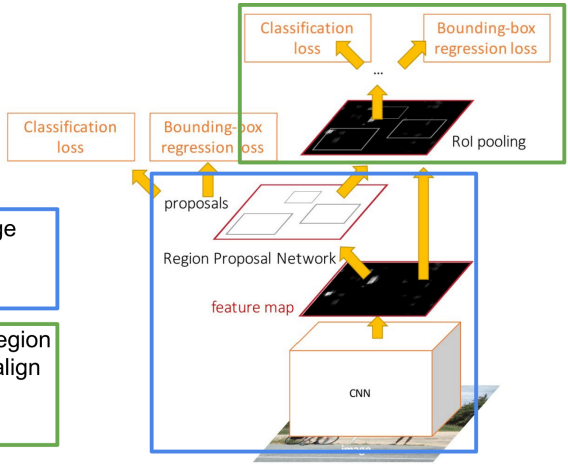
Faster R-CNN:

Make CNN do proposals!

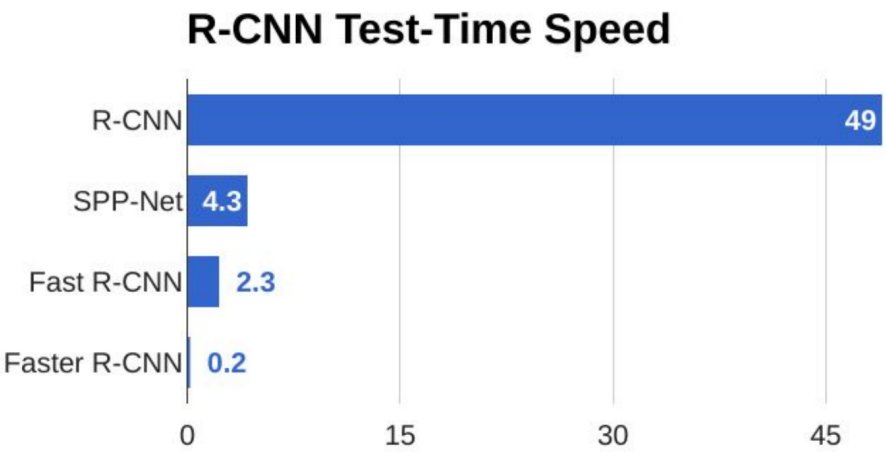
Faster R-CNN is a
Two-stage object detector

First stage: Run once per image
- Backbone network
- Region proposal network

Second stage: Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset



Faster R-CNN (cont.)

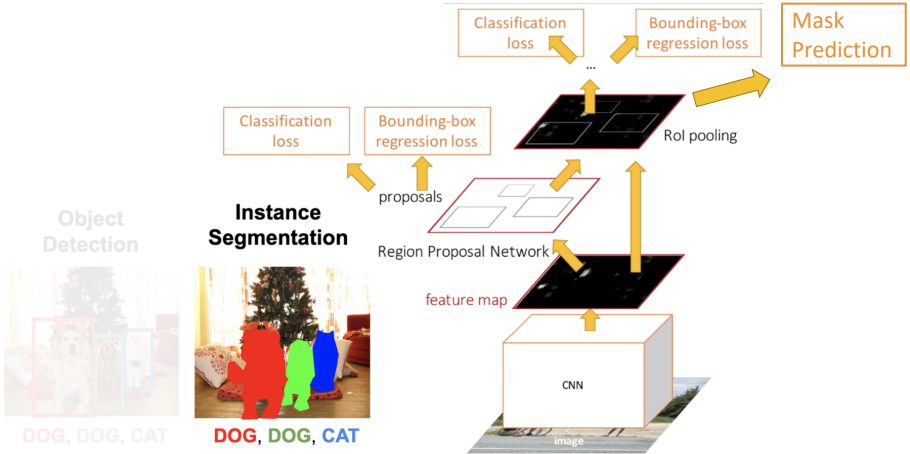


Instance Segmentation: **Mask R-CNN**

Mask R-CNN Overview

- **Developed on top of Faster R-CNN**
- Faster R-CNN outputs:
  - Class label
  - Bounding-box offset
- Mask R-CNN adds a third branch:
  - Predicts object mask for each candidate
- Performs both **Semantic** and **Instance Segmentation**

Mask R-CNN (cont.)



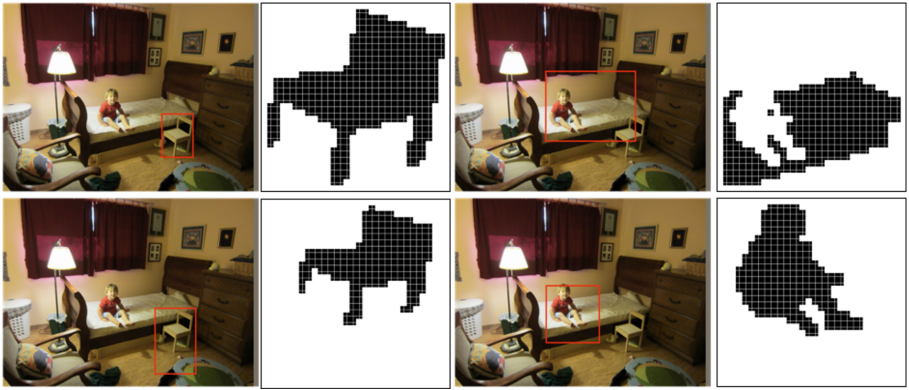
Advantages of Mask R-CNN

- **Simplicity**: Simple to train
- **Performance**: Outperforms previous methods, works almost in real-time
- **Efficiency**: Very efficient, small overhead compared to Faster R-CNN
- **Flexibility**: Can perform detection and estimation tasks simultaneously

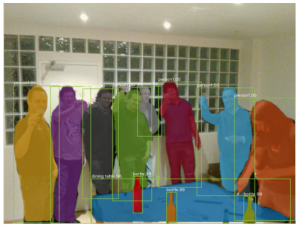
[0]He et al, "Mask R-CNN", ICCV 2017

Mask R-CNN: Example Training Targets

Mask R-CNN: Example Training Targets



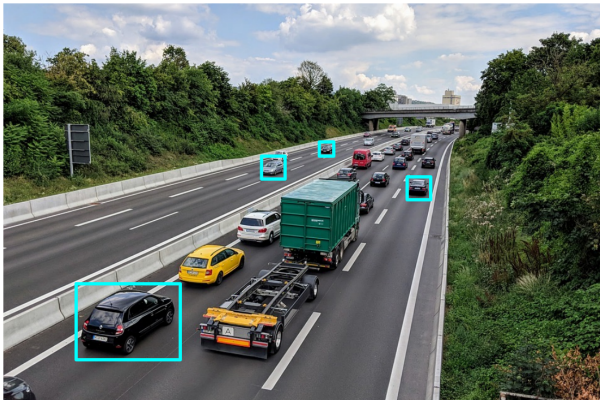
Mask R-CNN: Very Good Results!



Object Detection: **Single Shot Detections**

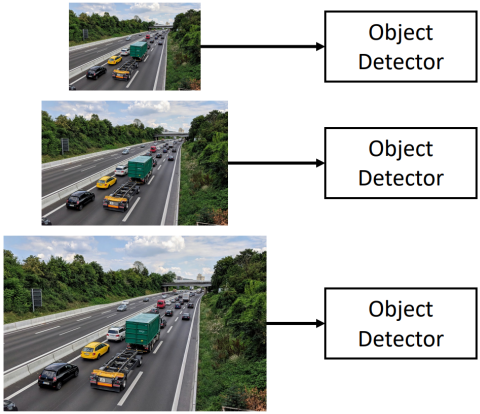
Dealing with Scale

- We need to detect objects of many different scales.
- How to improve scale invariance of the detector



Dealing with Scale: Image Pyramid

Classic idea: build an *image pyramid* by resizing the image to different scales, then process each image scale independently.

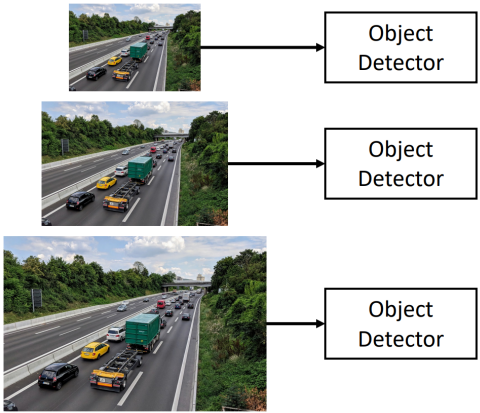


Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Image Pyramid (cont.)

Classic idea: build an *image pyramid* by resizing the image to different scales, then process each image scale independently.

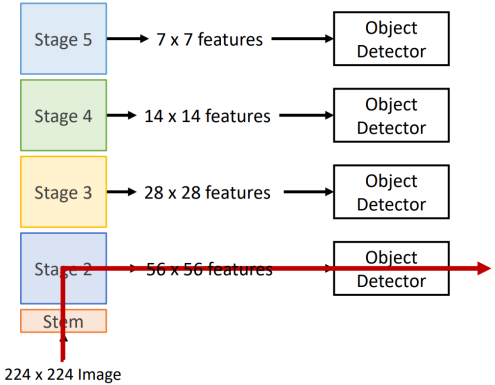
Problem: Expensive! Don't share any computation between scales



Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Image Pyramid

CNNs have multiple *stages* that operate at different resolutions. Attach an independent detector to the features at each level



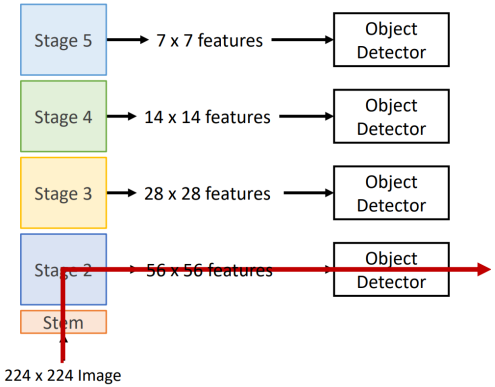
Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Image Pyramid (cont.)

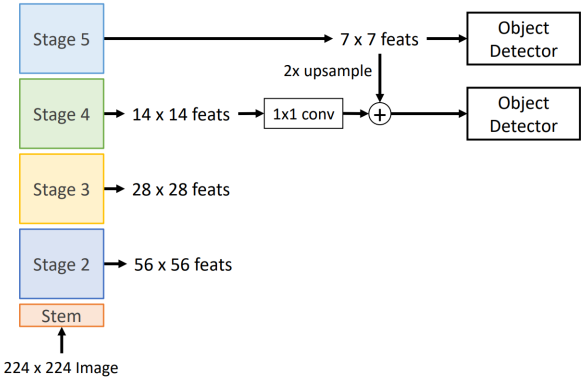
CNNs have multiple *stages* that operate at different resolutions. Attach an independent detector to the features at each level

Problem: detector on early features doesn't make use of the entire backbone; doesn't get access to high-level features

Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017



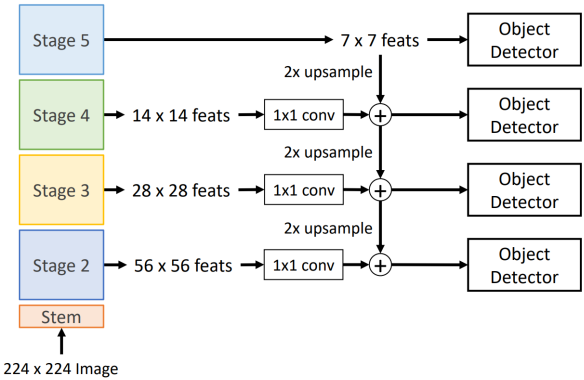
Add *top down connections* that feed information from high level features back down to lower level features



Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Add *top down connections* that feed information from high level features back down to lower level features

Efficient multiscale features where all levels benefit from the whole backbone! Widely used in practice

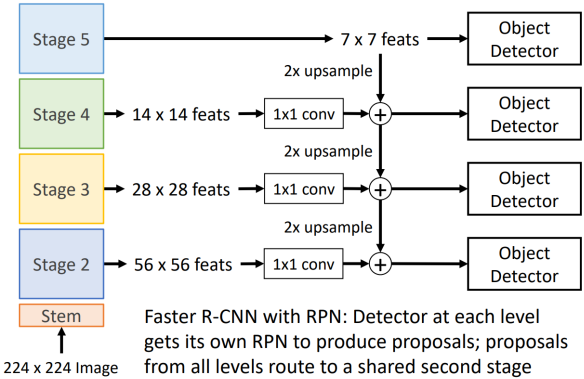


Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Add *top down connections* that feed information from high level features back down to lower level features

Efficient multiscale features where all levels benefit from the whole backbone! Widely used in practice

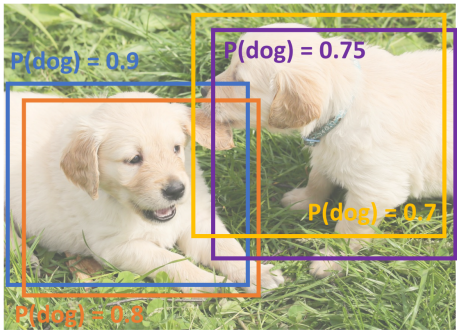
Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017



Faster R-CNN with RPN: Detector at each level gets its own RPN to produce proposals; proposals from all levels route to a shared second stage

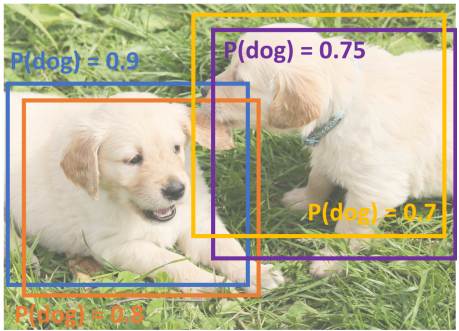
Overlapping Boxes: Non-Max Suppression (NMS)

- **Problem:** Object detectors often output many overlapping detections



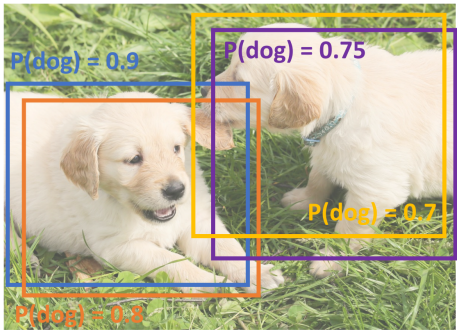
Overlapping Boxes: Non-Max Suppression (NMS)

- **Problem:** Object detectors often output many overlapping detections
- **Solution:** Post-process raw detections using Non-Max Suppression (NMS)



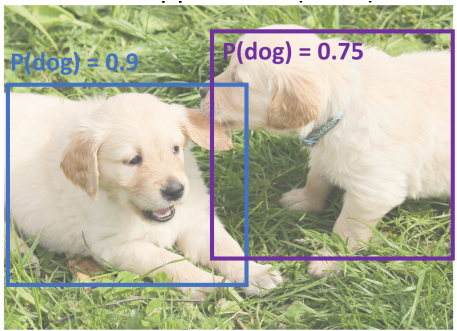
Overlapping Boxes: Non-Max Suppression (NMS)

- **Problem:** Object detectors often output many overlapping detections
- **Solution:** Post-process raw detections using Non-Max Suppression (NMS)
  1. Select next highest-scoring box
  2. Eliminate lower-scoring boxes
  3. with IoU $>$ threshold (e.g. 0.7)
  4. If any boxes remain, GOTO 1



Overlapping Boxes: Non-Max Suppression (NMS)

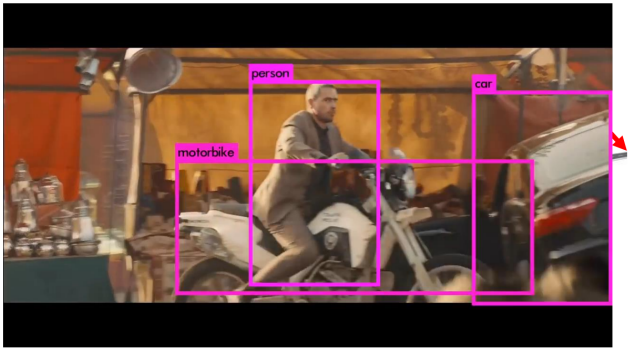
- **Problem:** Object detectors often output many overlapping detections
- **Solution:** Post-process raw detections using Non-Max Suppression (NMS)
  1. Select next highest-scoring box
  2. Eliminate lower-scoring boxes
  3. with IoU $>$ threshold (e.g. 0.7)
  4. If any boxes remain, GOTO 1



Overlapping Boxes: Non-Max Suppression (NMS)

- **Problem:** Object detectors often output many overlapping detections
- **Solution:** Post-process raw detections using Non-Max Suppression (NMS)
  1. Select next highest-scoring box
  2. Eliminate lower-scoring boxes
  3. with IoU $>$ threshold (e.g. 0.7)
  4. If any boxes remain, GOTO 1
- **Problem:** NMS may eliminate "good" boxes when objects are highly overlapping... no good solution =(

Single Shot Object Detection



Single Shot:
SSD, YOLO ...

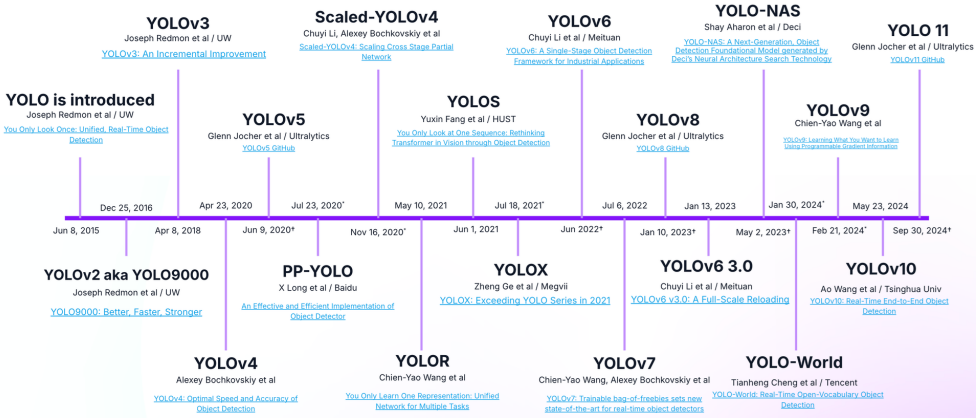
Fast
High false rate

Object Detection: **YOLO Family**

YOLO: A Game-Changer

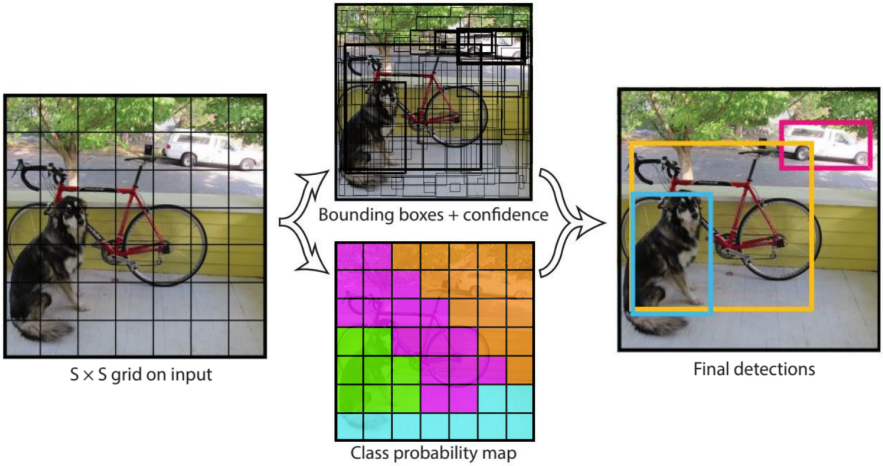
- **Motivation:** All previous methods were too slow for real time. YOLO does detection in one pass.
- **Impact:** Became the basis for real-time vision everywhere.

YOLO Family (cont.)

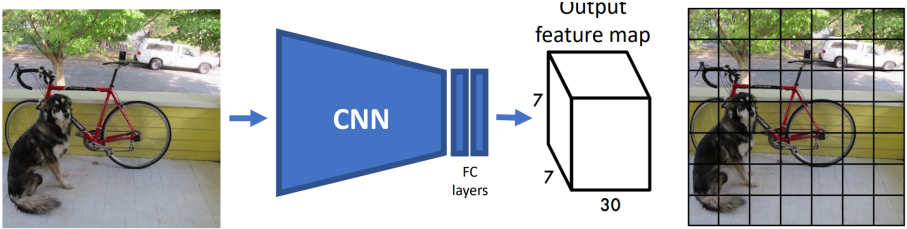


Object Detection: **YOLO (v1)**

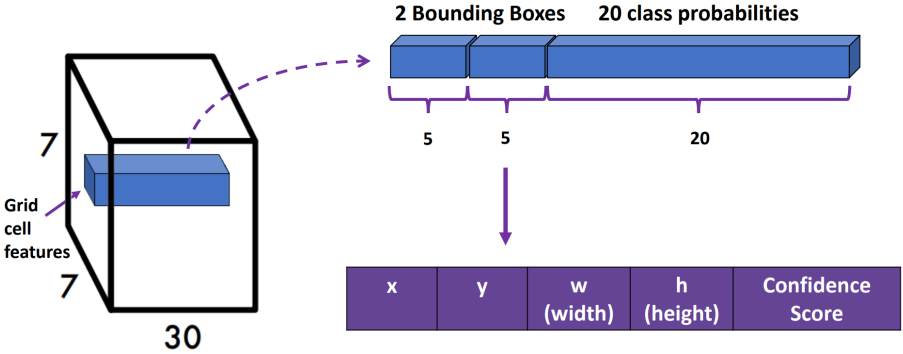
YOLO - Overview

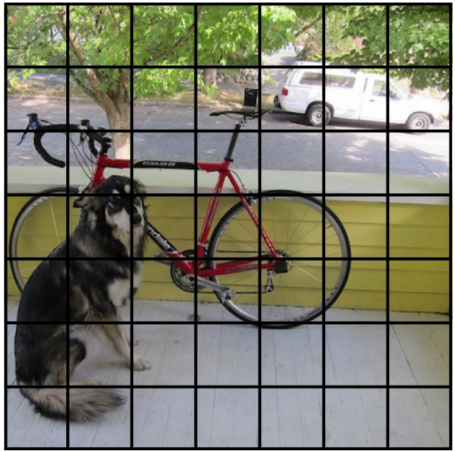


YOLO - Overview

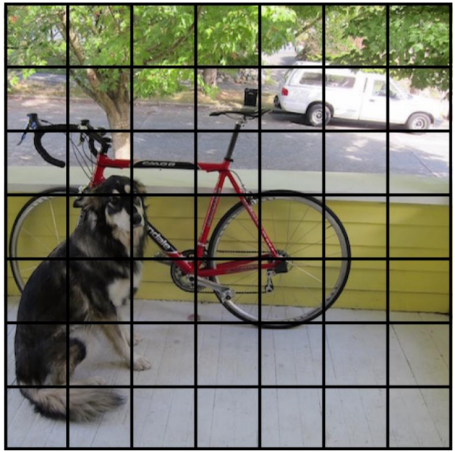


YOLO - Overview



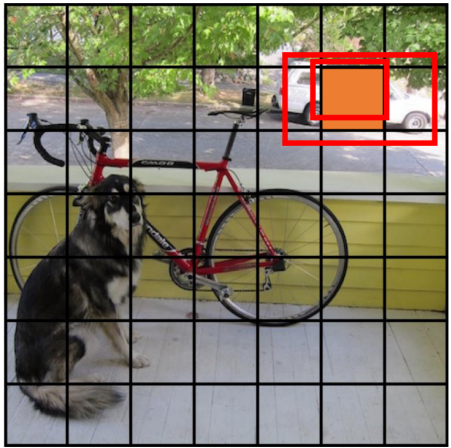


Each cell predicts



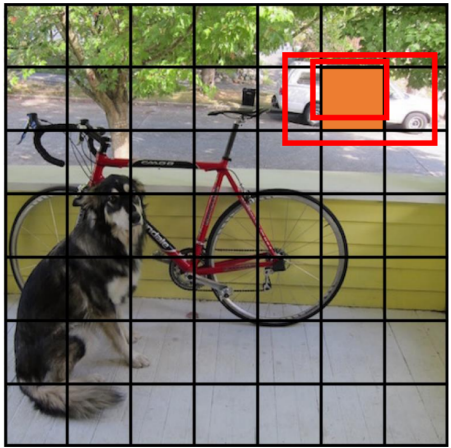
Each cell predicts

- $B = 2$ bounding boxes (x, y, w, h)+ confidence score



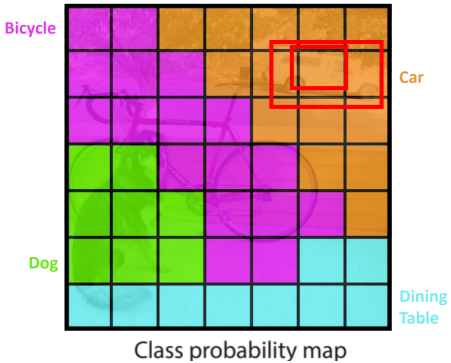
Each cell predicts

- $B = 2$ bounding boxes (x, y, w, h)+ confidence score
- $C = 20$ class probabilities



Each cell predicts

- $B = 2$ bounding boxes (x, y, w, h)+ confidence score
- $C = 20$ class probabilities



Each cell predicts

- $B = 2$ bounding boxes (x, y, w, h)+ confidence score
- $C = 20$ class probabilities

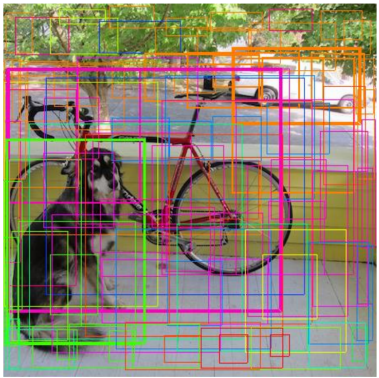
SxSxB Bounding-Boxes (S=7,B=2 → 96 Bboxes)

S × S grid on input

Each cell predicts

- $B = 2$ bounding boxes (x, y, w, h)+ confidence score
- $C = 20$ class probabilities

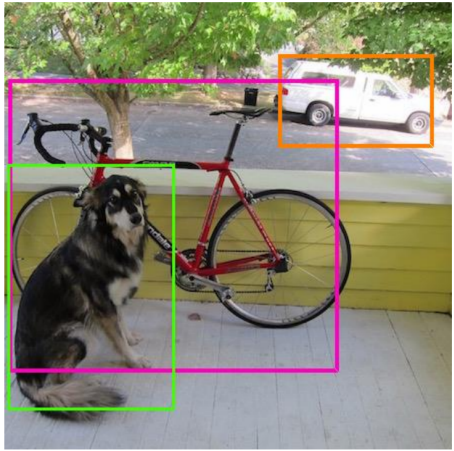
SxSxB Bounding-Boxes (S=7,B=2 → 96 Bboxes)



Each cell predicts

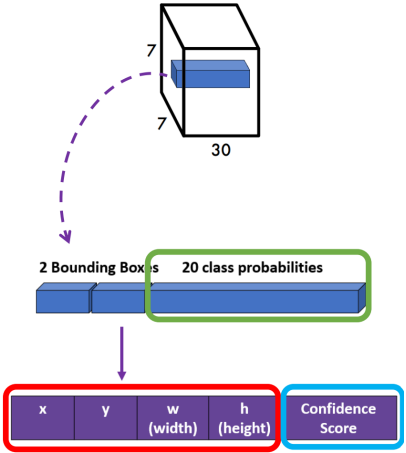
- $B = 2$ bounding boxes (x, y, w, h)+ confidence score
- $C = 20$ class probabilities

- Apply Non-Maximum Suppression



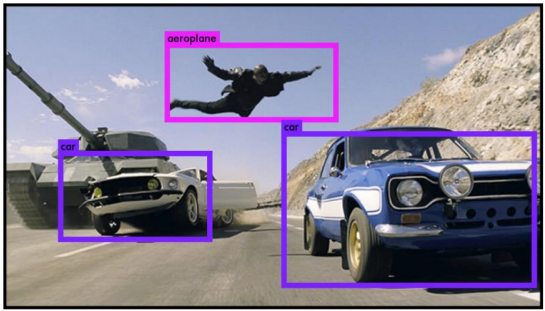
YOLO – Loss function

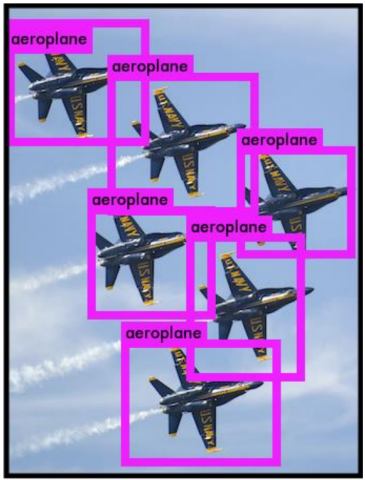
$$\mathcal{L} = \mathcal{L}_{Localization\ Loss} + \mathcal{L}_{Confidence\ Loss} + \mathcal{L}_{Classification\ Loss}$$



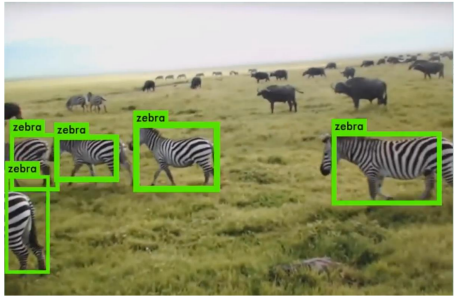
YOLO - Benefits

- Fast. Good for real-time processing
- End-to-end training



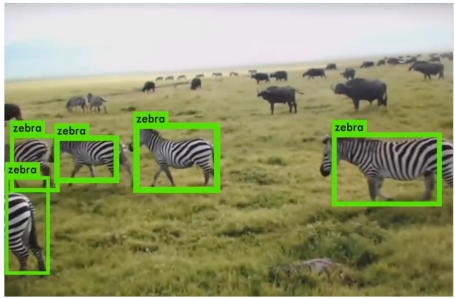


YOLO - Limitations



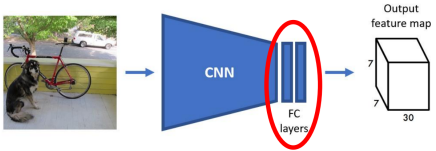
YOLO - Limitations

- Difficult to detect small objects
- Coarse predictions



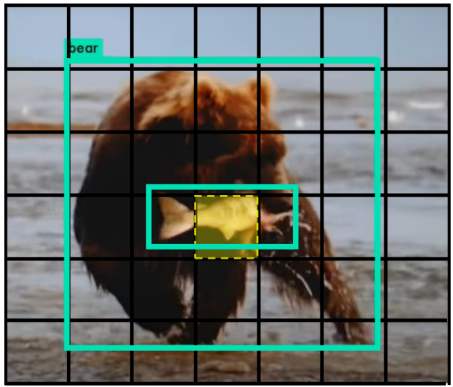
YOLO - Limitations

- Difficult to detect small objects
- Coarse predictions
- Fixed input size



YOLO - Limitations

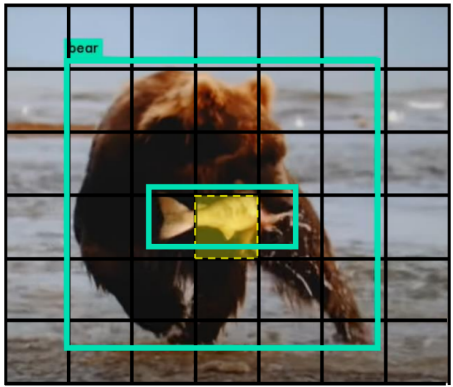
- Difficult to detect small objects
- Coarse predictions
- Fixed input size
- A grid cell can predict only one class



YOLO - Limitations

- Difficult to detect small objects
- Coarse predictions
- Fixed input size
- A grid cell can predict only one class

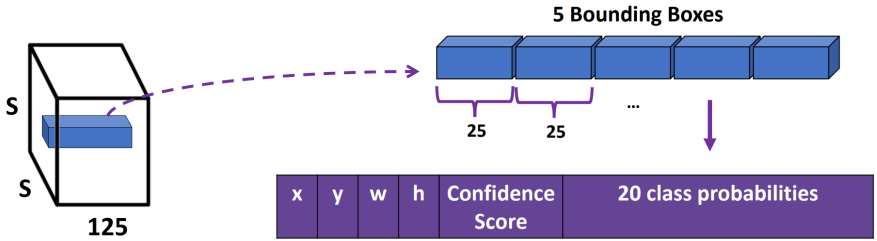
- Solutions:
  - Remove fc layers!
  - Predict class per bbox (not per cell)



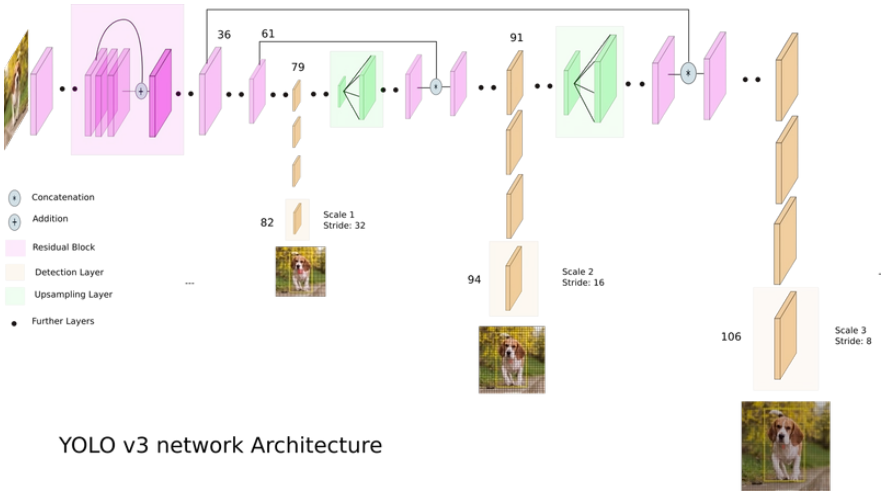
Object Detection: **YOLOv2 / YOLO9000**

YOLOv2 / YOLO9000 (2017)

- Introduced **anchor boxes**, **batch normalization**, and higher input resolution (**416×416**)
- Removed fully connected layers for a fully convolutional architecture
- Each grid cell predicts class probabilities for each anchor box
- Achieved a significant mAP improvement: ∼10% increase to ∼73% on PASCAL VOC
- Still struggles with small objects—anchor boxes help, but not enough



Object Detection: **YOLOv3**



YOLO v3 network Architecture

YOLOv4 (2020): A community-driven upgrade to YOLOv3, introducing several enhancements:

- **Backbone:** CSPDarknet53 for improved feature extraction.
- **Activation:** Mish activation function for better non-linearity.
- **Neck:** Spatial Pyramid Pooling (SPP) and PANet for robust multi-scale feature aggregation.
- **Loss:** CIoU loss for more accurate bounding box regression.
- **Training Tricks:** Bag-of-Freebies and Bag-of-Specials to boost performance without increasing inference cost.

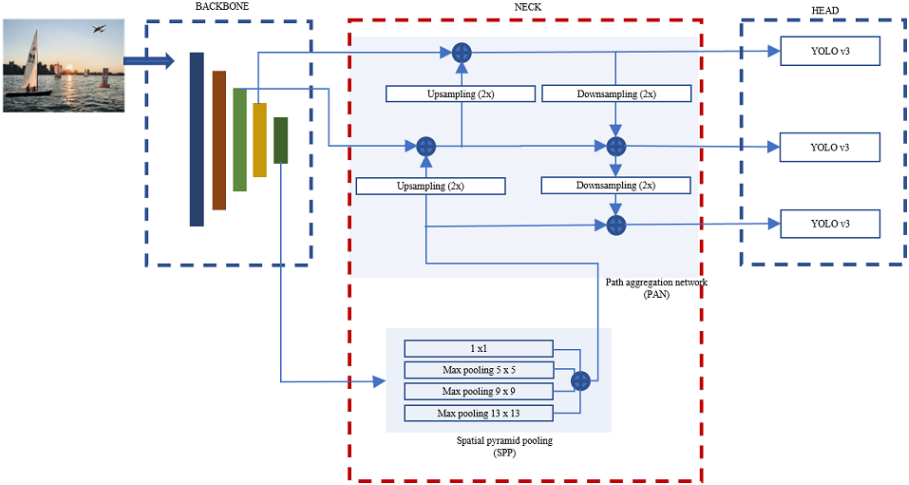
Performance:

- ∼43.5 mAP on COCO dataset.
- ∼65 FPS on GPU (real-time).

Limitation: Relies on many training and architectural tricks, making it more complex to fully understand and implement.

Object Detection: **YOLOv4**

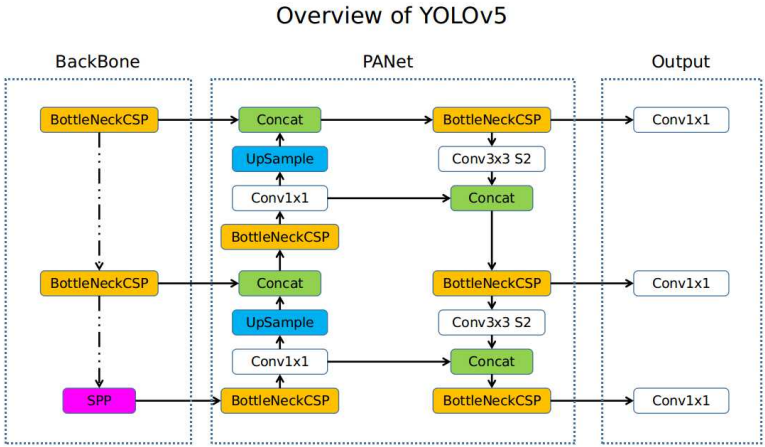
YOLOv4


BACKBONE
NECK
HEAD
Upsampling (2x)
Downsampling (2x)
Upsampling (2x)
Downsampling (2x)
YOLO v3
YOLO v3
YOLO v3
Path aggregation network
(PAN)
1 x1
Max pooling 5 x 5
Max pooling 9 x 9
Max pooling 13 x 13
Spatial pyramid pooling
(SPP)

YOLOv4 (cont.)

- **YOLOv3 (2018):**
  - Uses Darknet-53 backbone (deeper, more powerful CNN)
  - Multi-scale detection via Feature Pyramid Networks (FPN)
  - Achieves ∼80% mAP on COCO dataset
  - Improved small-object detection compared to previous YOLO versions
- **Limitations:**
  - Still misses some overlapping objects
  - Less accurate than two-stage detectors (e.g., Faster R-CNN)

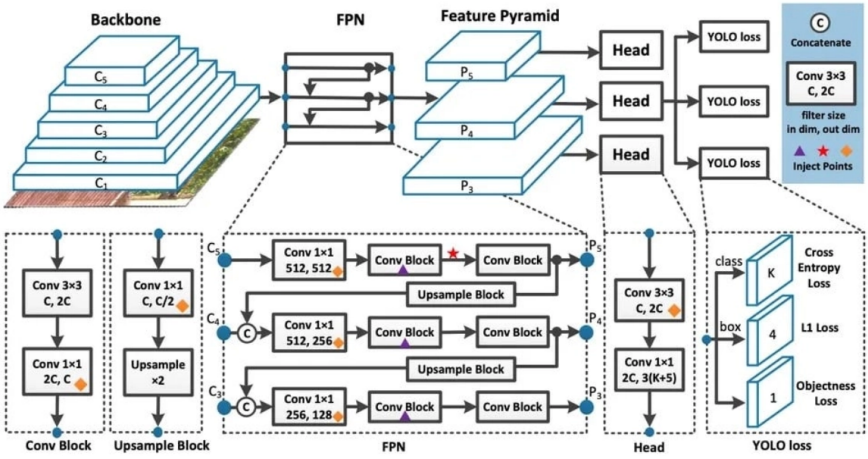
Object Detection: **YOLOv5**


Overview of YOLOv5
BackBone
PANet
Output
BottleNeckCSP
BottleNeckCSP
SPP
Concat
UpSample
Conv1x1
BottleNeckCSP
Concat
UpSample
Conv1x1
BottleNeckCSP
BottleNeckCSP
Conv3x3 S2
Concat
BottleNeckCSP
Conv3x3 S2
Concat
BottleNeckCSP
Conv1x1
Conv1x1
Conv1x1

YOLOv5 (2020) Highlights:

- Developed by Ultralytics with a PyTorch-based implementation.
- Offers multiple model sizes: nano (n), small (s), medium (m), large (l), extra-large (x).
- Introduces mosaic data augmentation for improved generalization.
- Achieves ∼49 mAP (COCO), ∼140 FPS (V100).
- Strengths: Easy to train, use, and deploy.
- Limitations: Still anchor-based; licensing under AGPL has faced scrutiny.

Object Detection: **YOLOv8**


Backbone
FPN
Feature Pyramid
Head
Head
Head
YOLO loss
YOLO loss
YOLO loss
Concatenate
Conv 3×3
C, 2C
filter size
in dim, out dim
Inject Points
Conv 3×3
C, 2C
Conv 1×1
2C, C
Conv 1×1
C, C/2
Upsample
×2
Conv 1×1
512, 512
Conv Block
Conv Block
Upsample Block
Conv 1×1
512, 256
Conv Block
Conv Block
Upsample Block
Conv 1×1
256, 128
Conv Block
Conv Block
Conv 3×3
C, 2C
Conv 1×1
2C, 3(K+5)
class
box
K
4
1
Cross
Entropy
Loss
L1 Loss
Objectness
Loss
Conv Block
Upsample Block
FPN
Head
YOLO loss

YOLOv8 (2023) Highlights:

- **Anchor-free** detection architecture
- CSP backbone, FPN+PAN neck
- Supports object detection, segmentation, and pose estimation
- High performance: ∼52.5 mAP, ∼280 FPS
- Strengths: multi-task capabilities, ease of use, segmentation support
- Challenges: small/clustered object detection remains difficult

Object Detection: **Beyond v8: YOLOv9 – YOLOv12**

Beyond v8: YOLOv9 – YOLOv12

YOLOv9 (2024):

- Introduces PGI (Progressive Gradient Integration) and GELAN (Gradient Enhanced Lightweight Aggregation Network) for improved gradient flow.
- Achieves better parameter efficiency and accuracy.

YOLOv10 (May 2024):

- NMS-free detection using dual-assignment strategy.
- Incorporates partial self-attention for enhanced feature representation.
- Achieves 1.8× faster inference than RT-DETR-R18.

YOLOv11 (Sept 2024):

- Utilizes C3k2/C2PSA blocks and dynamic head architecture.
- Supports oriented bounding boxes and segmentation tasks.

- Delivers ∼2% mAP gain over YOLOv8 with fewer parameters.

YOLOv12 (Feb 2025):

- Attention-centric design with Area Attention and R-ELAN modules.
- Achieves 1.64 ms latency (light version) and excels in poor lighting and tracking scenarios.
- Reports ∼40.6% mAP (light version).

Object Detection: **Limitations and Future Directions**

Limitations Still to Solve

- **Small or Clustered Object Detection**: Current models struggle to accurately detect small or closely packed objects.
- **Robustness in Adverse Conditions**: Performance drops significantly under poor lighting or challenging weather conditions.
- **Dataset Bias and Labeling Cost**: Datasets may not represent real-world diversity, and manual annotation remains expensive.
- **On-device Deployment Complexity**: Running detection models on edge devices is challenging due to resource constraints and licensing issues.

What's Next? Future Directions

- **Transformer-first Models**: Approaches like DETR and RT-DETR are surpassing YOLO in both speed and accuracy.
- **Hybrid Architectures**: Models such as MambaNeXt combine global context modeling with edge efficiency.
- **Weak-supervised & Prompt-driven Models**: Examples like YOLO-World enable zero-shot detection with minimal supervision.
- **Ongoing Challenges**: Focus areas include improving detection of small/clustered objects, open-vocabulary detection, and developing models suitable for edge deployment.

Object Detection: **References & Resources**

Further Reading

[1] R-CNN to Faster R-CNN: Girshick et al. 2014–2015
R-CNN: https://arxiv.org/abs/1311.2524
Fast R-CNN: https://arxiv.org/abs/1504.08083
Faster R-CNN: https://arxiv.org/abs/1506.01497

[2] Mask R-CNN: He et al. (2017)
https://arxiv.org/abs/1703.06870

[3] YOLO Original Series (v1–v3): Redmon et al.
YOLOv1: https://arxiv.org/abs/1506.02640
YOLOv2: https://arxiv.org/abs/1612.08242
YOLOv3: https://pjreddie.com/media/files/papers/YOLOv3.pdf

[4] YOLOv4: Bochkovskiy et al. 2020
https://arxiv.org/abs/2004.10934

Further Reading (cont.)

[5] YOLOv5: Ultralytics (2020)
https://github.com/ultralytics/yolov5

[6] YOLOv6: Li et al. (2022)
https://arxiv.org/abs/2209.02976

[7] YOLOv7: Wang et al. (2022)
https://arxiv.org/abs/2207.02696

[8] YOLOv8: Ultralytics (2023)
https://github.com/ultralytics/ultralytics

[9] YOLOv10: Wang et al. (May 2024)
https://arxiv.org/abs/2405.12345

[10] YOLOv11: Architectural update Sept 2024

Further Reading (cont.)

[11] YOLOv12: Attention-centric Feb 2025

[12] MambaNeXt-YOLO (June 2025)

[13] RT-DETR: Zhao et al. (April 2023)
https://arxiv.org/abs/2304.08069

[14] DETR (Transformers): https://arxiv.org/abs/2005.12872

[15] YOLO-World zero-shot model
https://arxiv.org/abs/2311.15620

[16] Fei-Fei Li, Yunzhu Li & Ruohan Gao, Stanford CS231n: Deep Learning for Computer Vision
http://cs231n.stanford.edu/

Further Reading (cont.)

[17] Assaf Shocher, Shai Bagon, Meirav Galun & Tali Dekel, WAIC DL4CV Deep Learning for Computer Vision: Fundamentals and Applications
https://www.weizmann.ac.il/math/AssafShocher/WAIC-DL4CV/

[18] Justin Johnson, UMich EECS 498.008/598.008: Deep Learning for Computer Vision
https://web.eecs.umich.edu/~justincj/teaching/eecs498/

- https://github.com/pjreddie/darknet
- https://github.com/ultralytics/yolov5
- https://github.com/facebookresearch/detectron2
- https://github.com/ultralytics/ultralytics
- Wikipedia: https://en.wikipedia.org/wiki/You_Only_Look_Once
- Springer: https://link.springer.com/article/10.1007/s11263-015-0816-y

Further Reading (cont.)

- Reddit: https://www.reddit.com/r/MachineLearning/
- CapaLearning: https://capalearning.com/

Credits

Dr. Prashant Aparajeya

Computer Vision Scientist — Director(AISimply Ltd)

p.aparajeya@aisimply.uk

This project benefited from external collaboration, and we acknowledge their contribution with gratitude.