Object Detection

Naeemullah Khan
naeemullah.khan@kaust.edu.sa



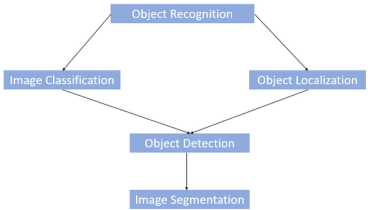
جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

KAUST Academy
King Abdullah University of Science and Technology

August 9, 2026



Source: Picsellia, “Segmentation vs Detection vs Classification in Computer Vision: A Comparative Analysis”, 2023



Source: GeeksforGeeks, “Object Detection vs Object Recognition vs Image Segmentation”

Table of Contents

1. Learning Outcomes
2. Introduction to Object Detection
3. Intersection over Union (IoU)
4. Detecting a Single Object
5. Detecting Multiple Objects
    1. Sliding Window
6. Region Proposals
7. Most Popular Methods
8. R-CNN Family
    1. R-CNN
    2. Fast R-CNN
    3. Faster R-CNN

Table of Contents (cont.)

4. Mask R-CNN

9. Single Shot Detections

10. YOLO Family
    1. YOLO (v1)
    2. YOLOv2 / YOLO9000
    3. YOLOv3
    4. YOLOv4
    5. YOLOv5
    6. YOLOv6 and YOLOv7
    7. YOLOv8
    8. Beyond v8: YOLOv9–YOLOv12

11. Industry Use Case

Table of Contents (cont.)

12. Limitations and Future Directions

⇒ Learning Outcomes

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

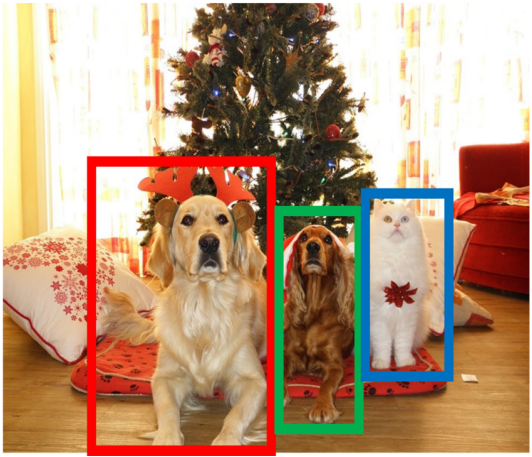
- Know why object detection matters and where it's used
- Score a predicted box against a ground-truth box with IoU, and remove duplicate detections with non-max suppression
- Understand how the R-CNN family (R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN) and YOLO work
- Explain what makes a detector two-stage or single-shot, and what each design trades away
- See how YOLO evolved and what each version solved
- Recognise current limitations and future innovation paths

Object Detection: **Introduction**

Object Detection

- **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.)



Source: UMich EECS 498 (2022), Lecture 13: Object Detection, slide 32

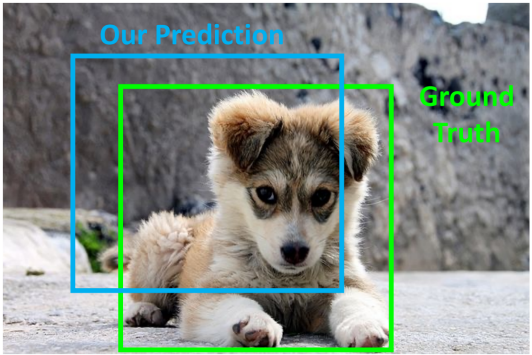
Object Detection: Challenges

- **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?

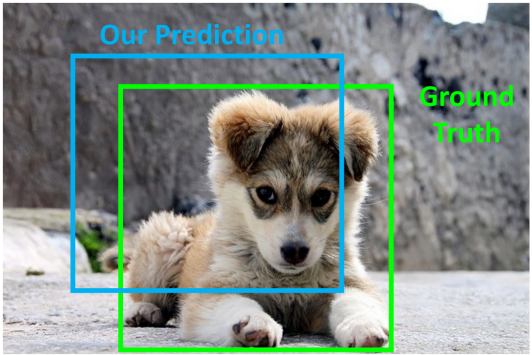


Source: UMich EECS 498 (2022), Lecture 13: Object Detection, slide 39

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"):

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



Source: UMich EECS 498 (2022), Lecture 13: Object Detection, slide 39

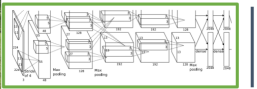
Object Detection: **Detecting a Single Object**

Detecting a Single Object

Often pretrained on ImageNet (Transfer learning)



This image is CC0 public domain



Vector: 4096

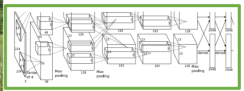
Treat localization as a regression problem!

Source: UMich EECS 498 (2022), Lecture 13: Object Detection, slide 45

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!

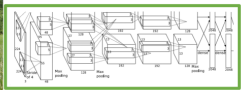
Source: UMich EECS 498 (2022), Lecture 13: Object Detection, slide 46

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

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

"Where"

L2 Loss

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

Treat localization as a regression problem!

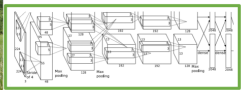
Source: UMich EECS 498 (2022), Lecture 13: Object Detection, slide 47

Detecting a Single Object (cont.)



This image is CC0 public domain

Often pretrained on ImageNet (Transfer learning)



Vector: 4096

Treat localization as a regression problem!

Fully Connected: 4096 to 1000

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

"What"

Correct label: Cat

Softmax Loss

Multitask Loss

Weighted Sum

Loss

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

Fully Connected: 4096 to 4

Box Coordinates
(x, y, w, h)

L2 Loss

"Where"

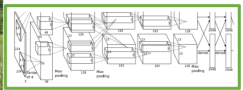
Correct box:
(x', y', w', h')

Source: UMich EECS 498 (2022), Lecture 13: Object Detection, slide 49

Detecting a Single Object (cont.)



This image is CC0 public domain



Often pretrained on ImageNet (Transfer learning)

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

L2 Loss

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

Multitask Loss

Weighted Sum

Loss

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

Treat localization as a regression problem!

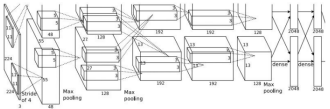
Problem: Images can have more than one object!

Source: UMich EECS 498 (2022), Lecture 13: Object Detection, slide 50

Object Detection: **Detecting Multiple Objects**

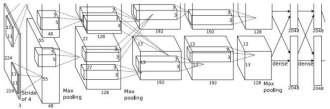
Multiple Objects





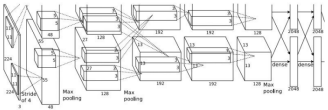
CAT: (x, y, w, h)





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





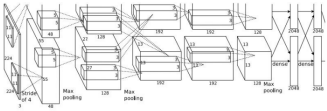
DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Source: Stanford CS231n (2021), Lecture 15: Detection and Segmentation, slide 41

Multiple Objects (cont.)

Each image needs a different number of outputs!

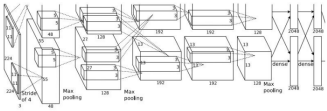




CAT: (x, y, w, h)

4 numbers

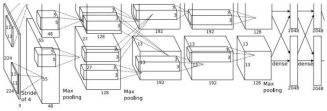




DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

12 numbers





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

Many numbers!

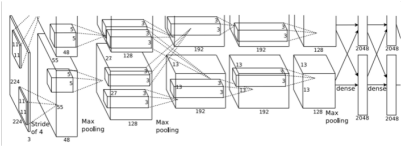
Source: Stanford CS231n (2021), Lecture 15: Detection and Segmentation, slide 42

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

Detecting Multiple Objects: Sliding Window



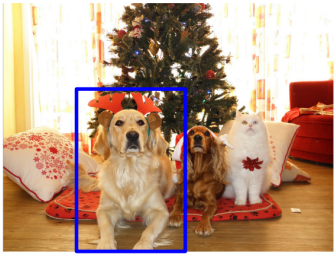
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



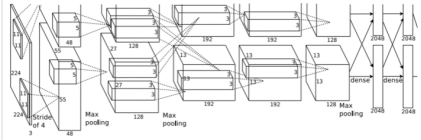
Dog? NO
Cat? NO
Background? YES

Source: UMich EECS 498 (2022), Lecture 13: Object Detection, slide 52

Detecting Multiple Objects: Sliding Window (cont.)



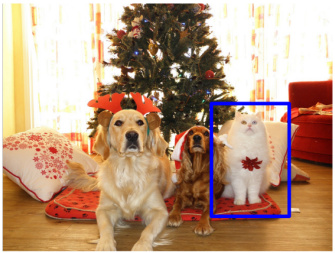
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



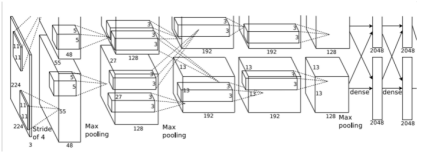
Dog? YES
Cat? NO
Background? NO

Source: UMich EECS 498 (2022), Lecture 13: Object Detection, slide 53

Detecting Multiple Objects: Sliding Window (cont.)



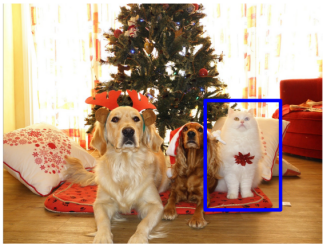
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? YES
Background? NO

Source: UMich EECS 498 (2022), Lecture 13: Object Detection, slide 55

Detecting Multiple Objects: Sliding Window (cont.)

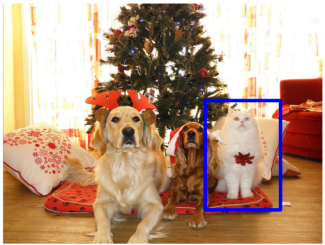


Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

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

Source: UMich EECS 498 (2022), Lecture 13: Object Detection, slide 56

Detecting Multiple Objects: Sliding Window (cont.)



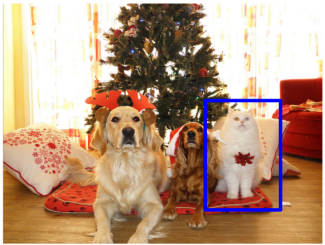
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

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)

Source: UMich EECS 498 (2022), Lecture 13: Object Detection, slide 57

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)

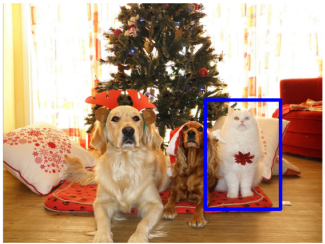
Total possible boxes:

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

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

Source: UMich EECS 498 (2022), Lecture 13: Object Detection, slide 58

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:

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

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

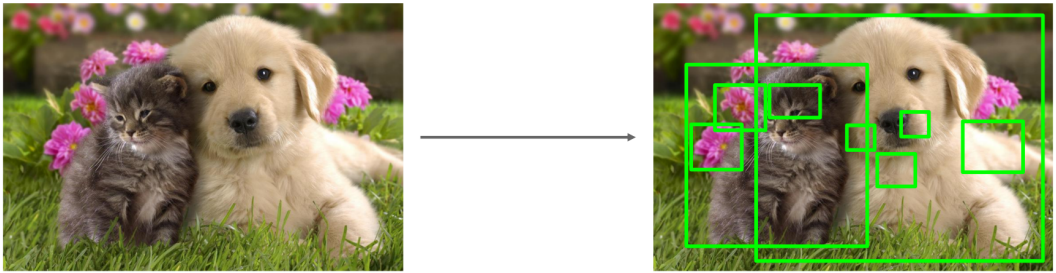
Source: UMich EECS 498 (2022), Lecture 13: Object Detection, slide 59

[0]Correction: the figure's own formula $\frac{H(H+1)}{2}\cdot\frac{W(W+1)}{2}$ gives $\approx 5.8\times10^{10}$ boxes, i.e. about 58 *billion*, not 58 million.

Object Detection: **Region Proposals**

Region Proposals

- 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

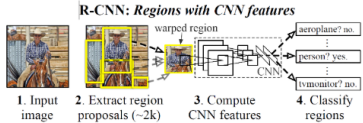


Source: UMich EECS 498 (2022), Lecture 13: Object Detection, slide 60

Object Detection: **Most Popular Methods**

Object Detection – Most Popular Methods

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



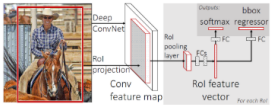


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax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

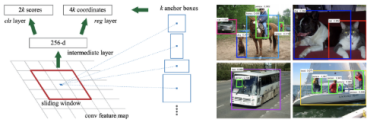


Figure 3: **Left**: Region Proposal Network (RPN). **Right**: Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.

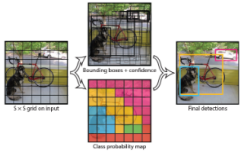
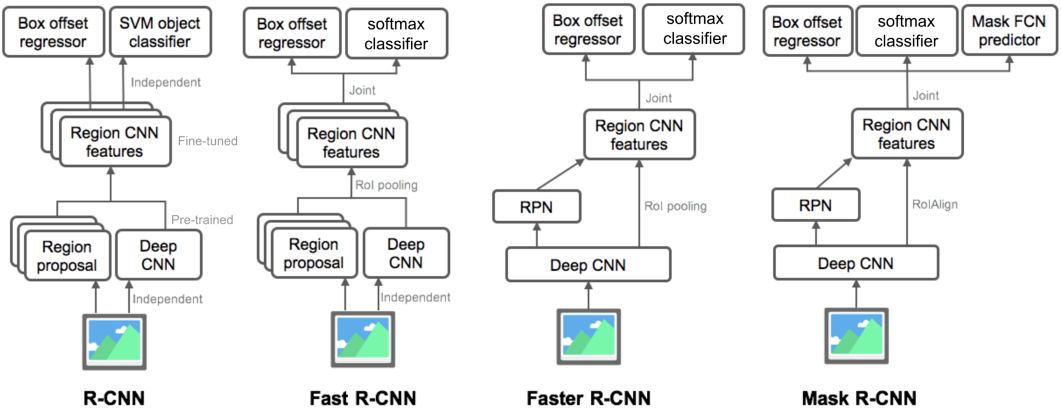


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

Sources: R-CNN strip, Girshick et al., CVPR 2014, Fig. 1; Fast R-CNN, Girshick, ICCV 2015, Fig. 1; RPN, Ren et al., NIPS 2015, Fig. 3; YOLO, Redmon et al., CVPR 2016, Fig. 2.

Object Detection: **R-CNN Family**

R-CNN Family



Source: Lilian Weng, "Object Detection for Dummies Part 3: R-CNN Family", lilianweng.github.io, 2017.

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.

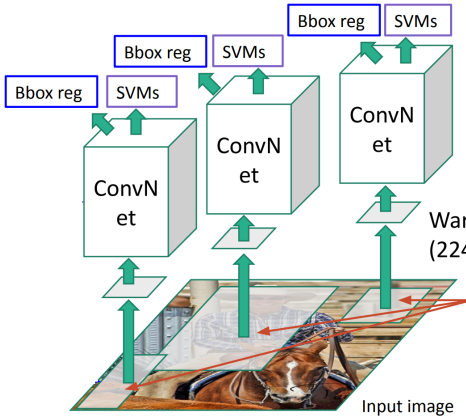
Object Detection: **Regions with Convolutional Neural Networks (R-CNN)**

R-CNN

- **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)



Classify regions with SVMs

Forward each region through ConvNet

Warped image regions (224x224 pixels)

Regions of Interest (RoI) from a proposal method (~2k)

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:

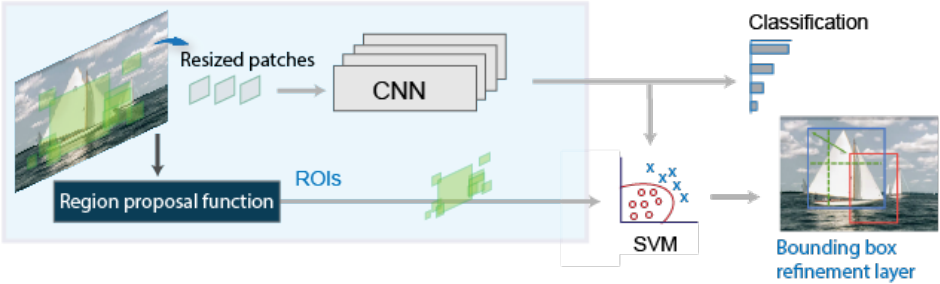
1. **Region Proposal (Selective Search)**
   - Instead of using a brute-force sliding window approach, R-CNN applies **Selective Search**, an algorithm that identifies around **2000 candidate object regions** (called region proposals).
   - This significantly reduces the number of regions to process,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

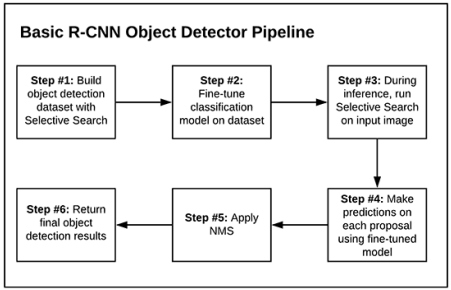
3. **Object Classification & Bounding Box Refinement**
   - The extracted features are passed through a **Support Vector Machine** (SVM) classifier to determine the object category.
   - A **bounding box regression** model is used to fine-tune the position of the detected objects.

R-CNN (cont.)



Source: MathWorks, "Getting Started with R-CNN, Fast R-CNN, and Faster R-CNN", MATLAB Computer Vision Toolbox documentation.

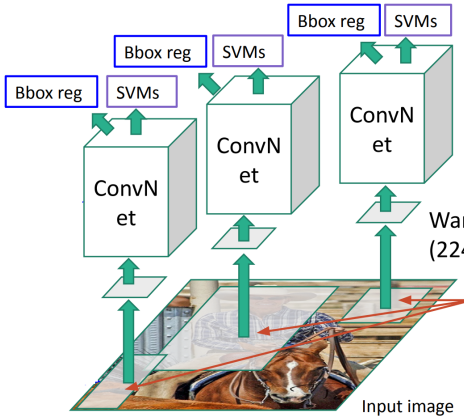
R-CNN (cont.)



Step 5 applies non-max suppression, which collapses the many overlapping boxes R-CNN produces down to one per object. We come back to it in detail later. Source: A. Rosebrock, "R-CNN object detection with Keras, TensorFlow, and Deep Learning", PyImageSearch, 2020.

R-CNN: Problem

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



Classify regions with SVMs

Forward each region through ConvNet

Warped image regions (224x224 pixels)

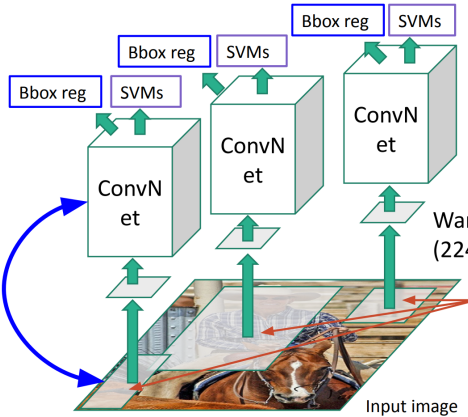
Regions of Interest (RoI) from a proposal method (~2k)

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.)

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



Classify regions with SVMs

Forward each region through ConvNet

Warped image regions (224x224 pixels)

Regions of Interest (RoI) from a proposal method (~2k)

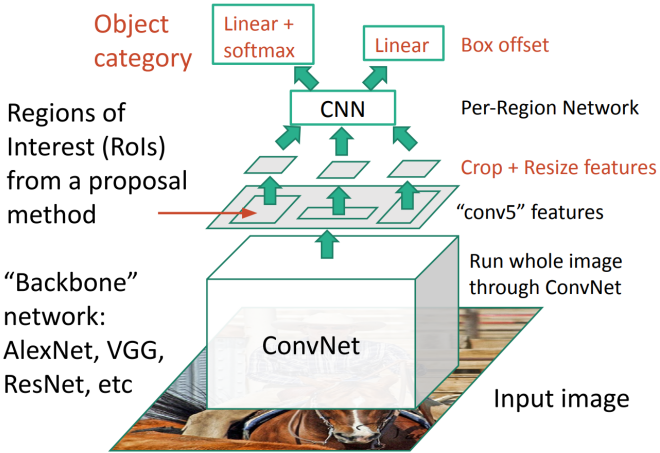
Problem: Very slow! Need to do ~2k independent forward passes for each image!

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.

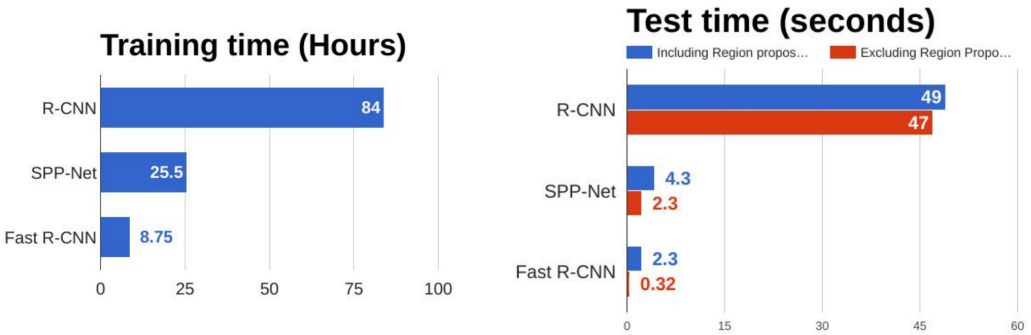
Object Detection: **Fast R-CNN**

Fast R-CNN



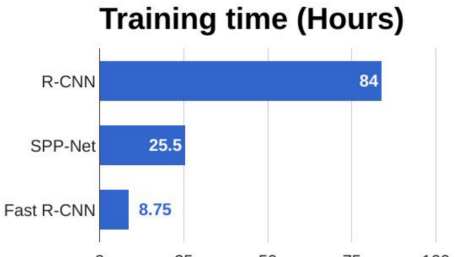
Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

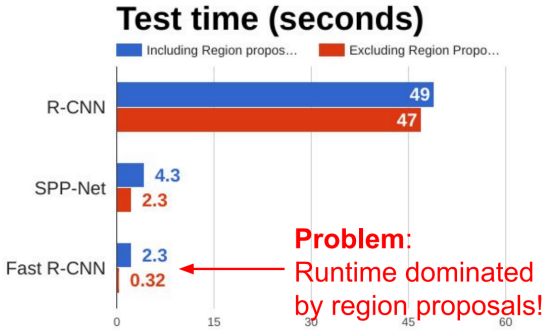
Fast R-CNN (cont.)



Source: Stanford CS231n (2021), Lecture 15: Detection and Segmentation, slide 75

Fast R-CNN (cont.)





Source: Stanford CS231n (2021), Lecture 15: Detection and Segmentation, slide 76

Object Detection: **Faster R-CNN**

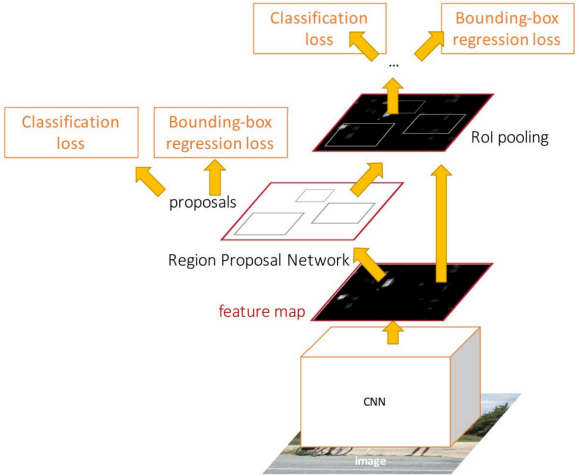
Faster R-CNN

- 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



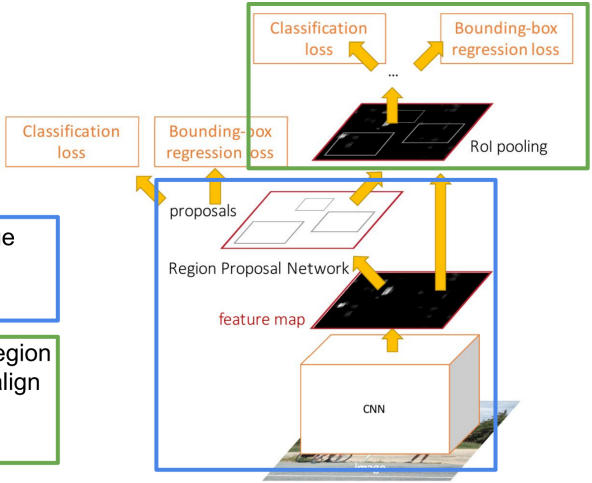
Source: Stanford CS231n (2021), Lecture 15: Detection and Segmentation, slide 77

Faster R-CNN:
Make CNN do proposals!

Faster R-CNN is a
Two-stage object detector

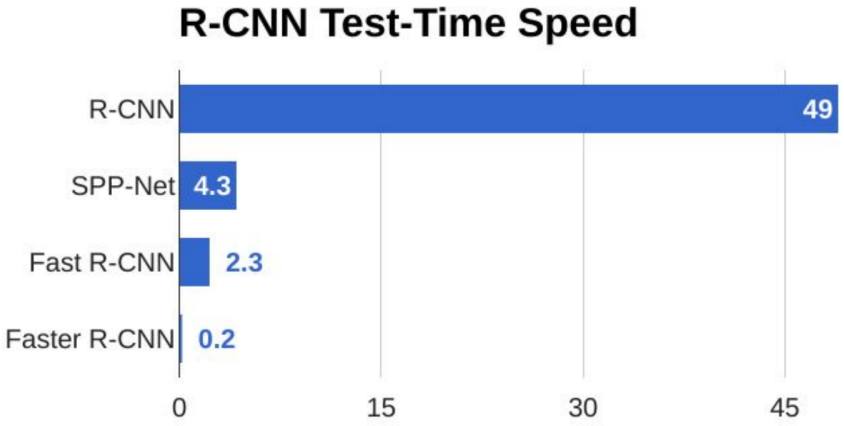
First stage: Run once per image
- Backbone network
- Region proposal network

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



Source: Stanford CS231n (2021), Lecture 15: Detection and Segmentation, slide 87

Faster R-CNN (cont.)



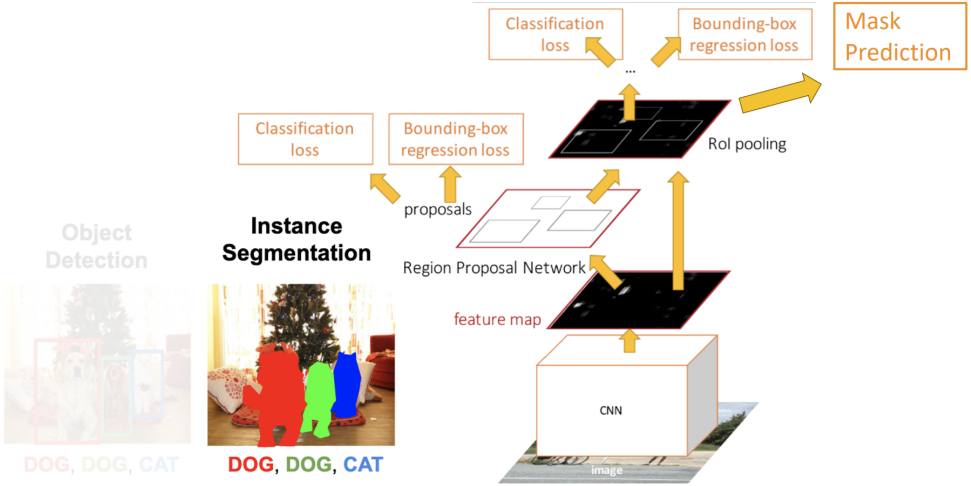
Source: Stanford CS231n (2021), Lecture 15: Detection and Segmentation, slide 85

Instance Segmentation: **Mask R-CNN**

Mask R-CNN Overview

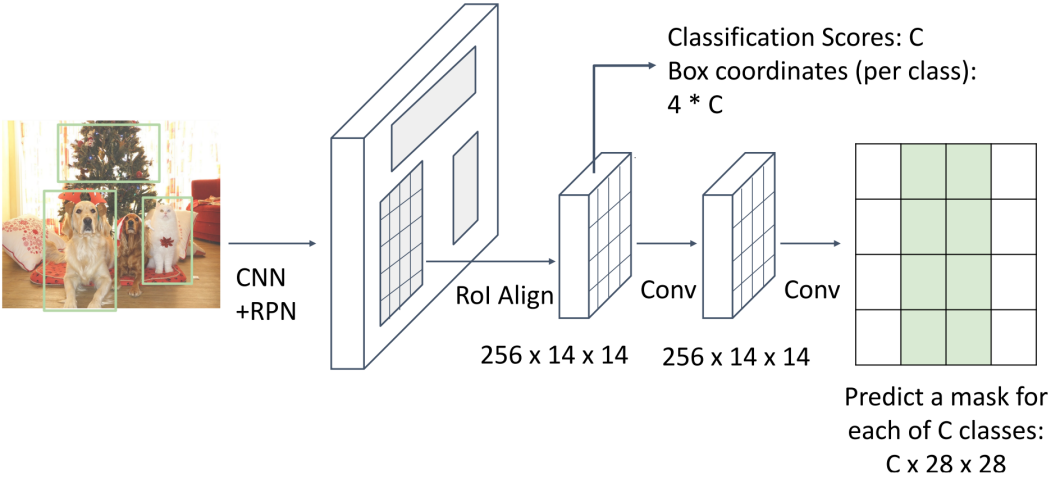
- **Developed on top of Faster R-CNN**
- Faster R-CNN outputs:
  - Class label
  - Bounding-box offset
- Mask R-CNN adds a third branch:
  - Predicts object mask for each candidate
- Performs **Instance Segmentation** (per-object masks; extensible to keypoint/pose estimation)

Mask R-CNN (cont.)



He et al., "Mask R-CNN", ICCV 2017.

Mask R-CNN (cont.)

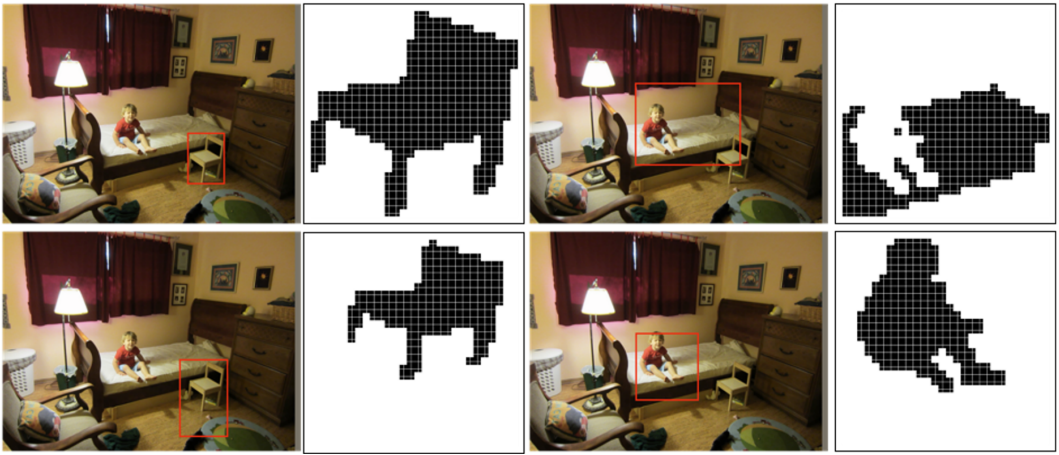


Source: UMich EECS 498 (2022), Lecture 15: Image Segmentation, slide 79 (figure from He et al., "Mask R-CNN", ICCV 2017)

Advantages of Mask R-CNN

- **Simplicity**: Simple to train
- **Performance**: Outperforms previous methods, at 5 fps (195 ms per image on a Tesla M40)
- **Efficiency**: Very efficient, small overhead compared to Faster R-CNN
- **Flexibility**: Can perform detection and estimation tasks simultaneously

Mask R-CNN: Example Training Targets



Source: UMich EECS 498 (2022), Lecture 15: Image Segmentation, slide 83 (figure from He et al., "Mask R-CNN", ICCV 2017)

Mask R-CNN: Very Good Results!

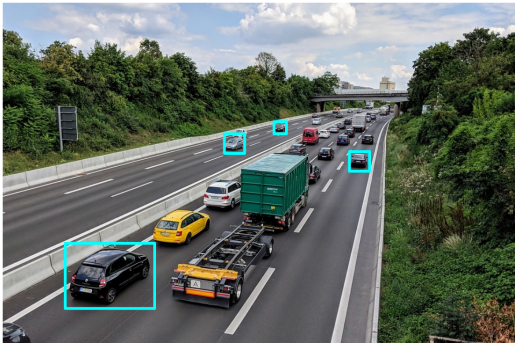


Source: UMich EECS 498 (2022), Lecture 15: Image Segmentation, slide 84 (figure from He et al., "Mask R-CNN", ICCV 2017)

Object Detection: **Single Shot Detections**

Dealing with Scale

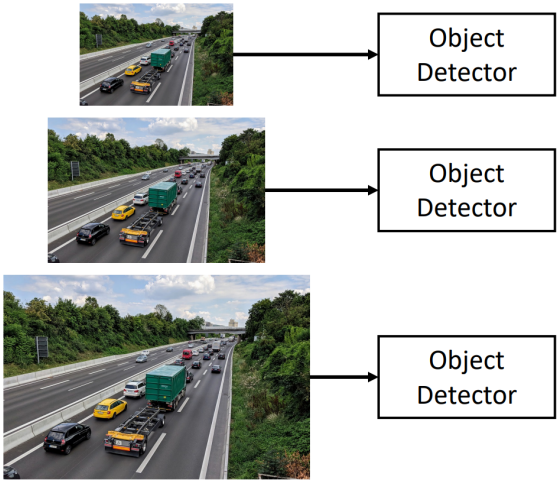
- We need to detect objects of many different scales.
- How to improve scale invariance of the detector



Source: UMich EECS 498 (2022), Lecture 14: Object Detectors, slide 73

Dealing with Scale: Image Pyramid

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

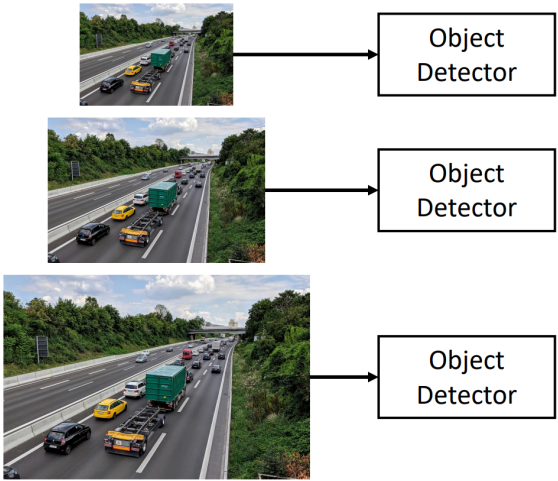


Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

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.

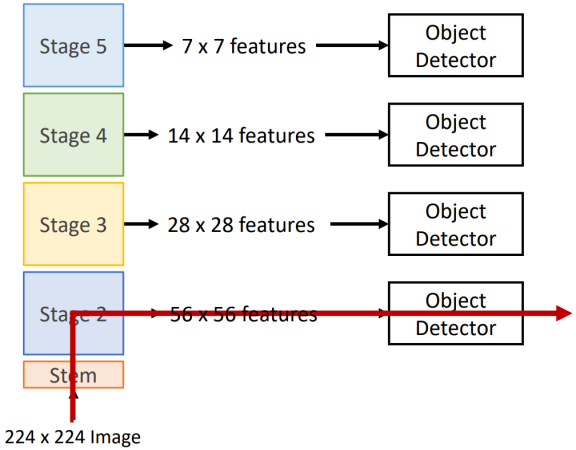
Problem: Expensive! Don't share any computation between scales



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



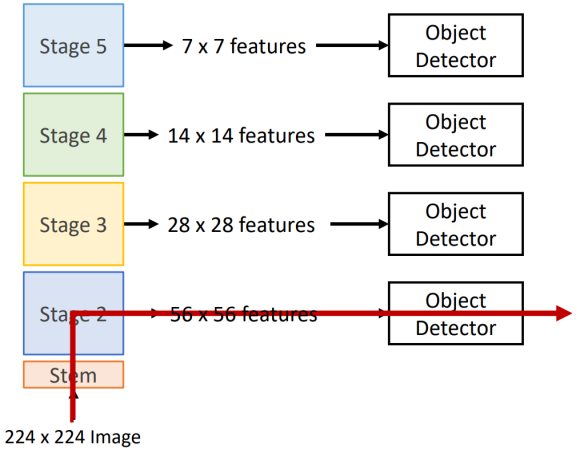
Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

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

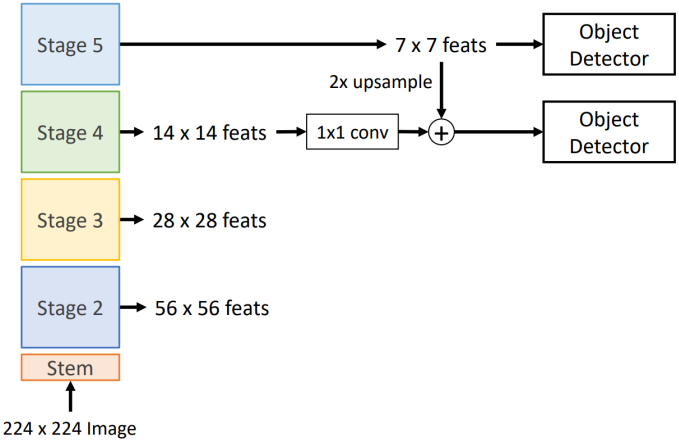
Problem: detector on early features doesn't make use of the entire backbone; doesn't get access to high-level features

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



Dealing with Scale: Feature Pyramid Network

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

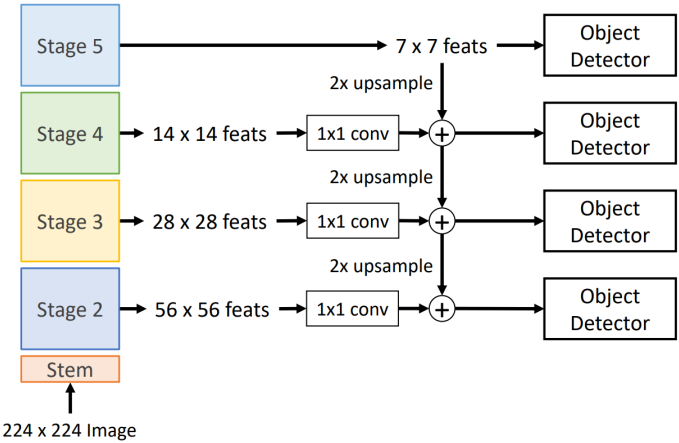


Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Feature Pyramid Network (cont.)

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



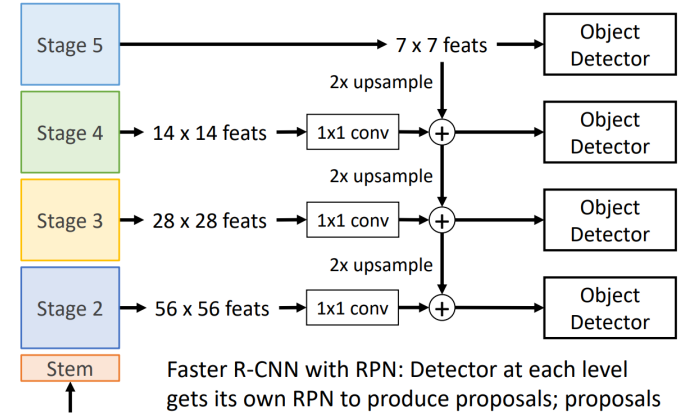
Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Feature Pyramid Network (cont.)

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

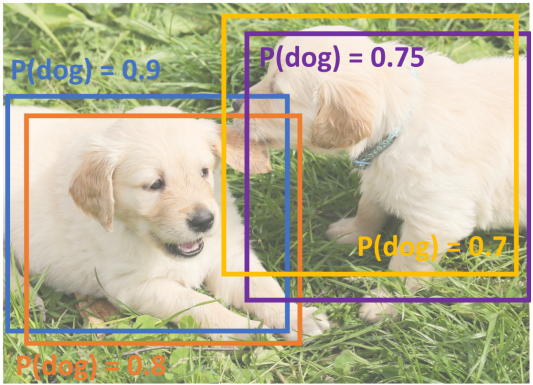


Faster R-CNN with RPN: Detector at each level gets its own RPN to produce proposals; proposals from all levels route to a shared second stage

[0]Correction: the venue printed inside these figures is wrong. Feature Pyramid Networks appeared at **CVPR** 2017, not ICCV 2017: Lin, Dollár, Girshick, He, Hariharan & Belongie, "Feature Pyramid Networks for Object Detection".

Overlapping Boxes: Non-Max Suppression (NMS)

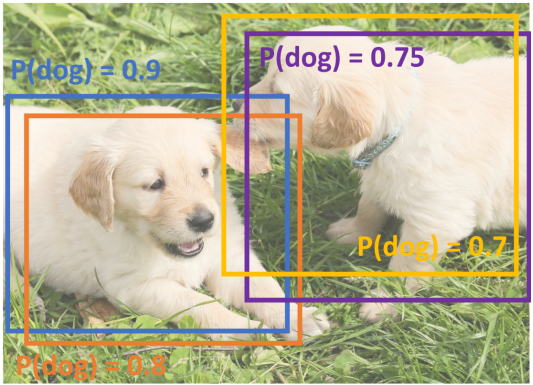
- **Problem:** Object detectors often output many overlapping detections



Source: UMich EECS 498 (2022), Lecture 13: Object Detection, slide 78

Overlapping Boxes: Non-Max Suppression (NMS)

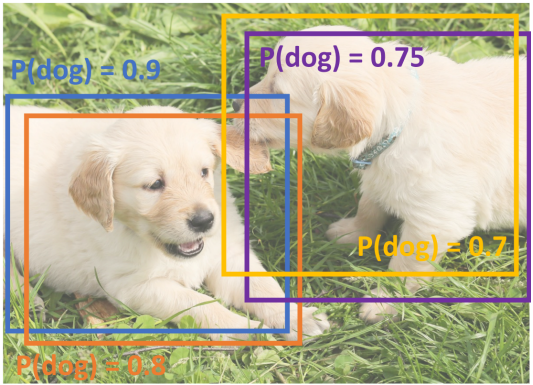
- **Problem:** Object detectors often output many overlapping detections
- **Solution:** Post-process raw detections using Non-Max Suppression (NMS)



Source: UMich EECS 498 (2022), Lecture 13: Object Detection, slide 78

Overlapping Boxes: Non-Max Suppression (NMS)

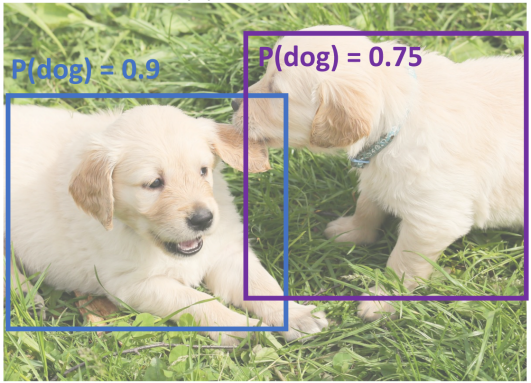
- **Problem:** Object detectors often output many overlapping detections
- **Solution:** Post-process raw detections using Non-Max Suppression (NMS)
  1. Select next highest-scoring box
  2. Eliminate lower-scoring boxes with IoU $>$ threshold (e.g. 0.7)
  3. If any boxes remain, GOTO 1



Source: UMich EECS 498 (2022), Lecture 13: Object Detection, slide 78

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 with IoU $>$ threshold (e.g. 0.7)
  3. If any boxes remain, GOTO 1



Source: UMich EECS 498 (2022), Lecture 13: Object Detection, slide 82

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 with IoU $>$ threshold (e.g. 0.7)
  3. If any boxes remain, GOTO 1
- **Problem:** NMS may eliminate "good" boxes when objects are highly overlapping... no good solution =(

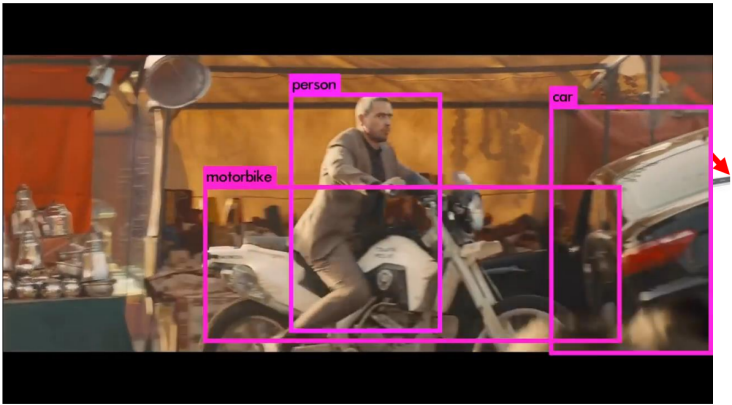


Source: UMich EECS 498 (2022), Lecture 13: Object Detection, slide 83

Two Stages or One?

- **Two-stage** (the R-CNN family): stage one proposes a few thousand candidate regions, stage two runs a classifier and a box regressor *once per region*. Every proposal gets a second look, and the cost of inference grows with the number of proposals.
- **Single shot**: one network predicts a class score and a box offset for every location in a dense, fixed grid of cells or anchors, in a single forward pass. There is no proposal stage, so the runtime does not depend on how many objects are in the image.
- The two families that established this design are **SSD** (Single Shot MultiBox Detector), which predicts from several feature maps at different scales, and **YOLO**, which we turn to next.
- **The trade-off:** single-shot detectors are far faster, but they must score a huge number of locations of which nearly all are background, and they historically lag two-stage detectors in accuracy, particularly on small objects.

Single Shot Object Detection



Single Shot:
SSD, YOLO ...

Fast
High false rate

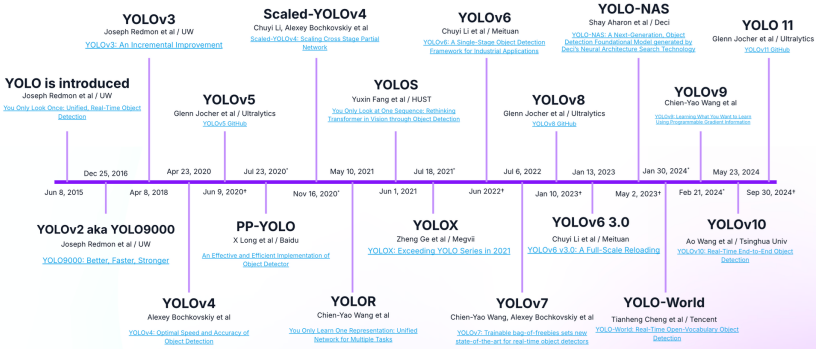
Source: WAIC DL4CV, "Object Detection and Segmentation" (L. Yariv, 2022), slide 5

Object Detection: **YOLO Family**

YOLO: One Network, One Pass

- **Motivation:** the region-proposal detectors above run a classifier once per proposal, which is too slow for real time. YOLO predicts every box and class in a single forward pass.
- **Impact:** became the standard starting point for real-time detection on video and embedded hardware.

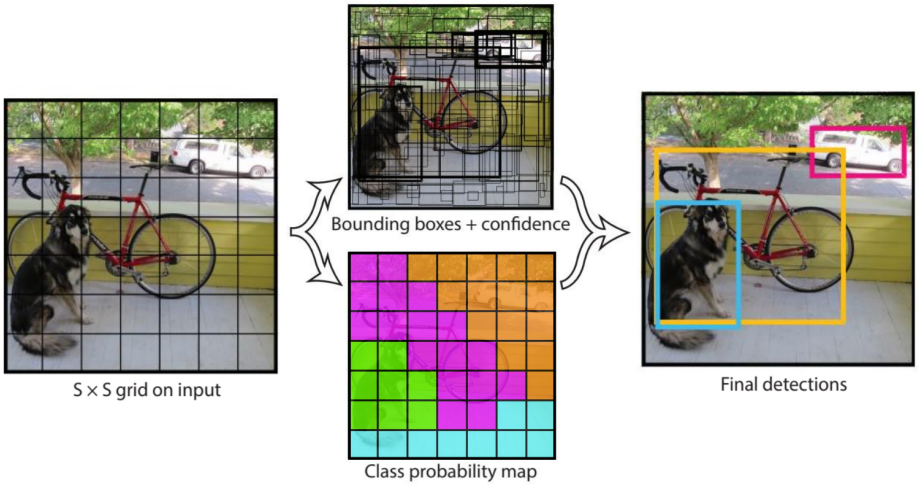
YOLO Family (cont.)



Source: Roboflow, "YOLO Greatest Hits" timeline, from "What Is YOLO? The Guide to YOLO Models", blog.roboflow.com (cropped).

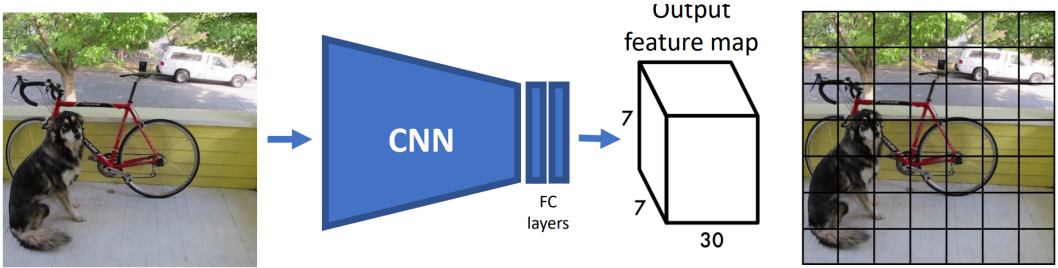
Object Detection: **YOLO (v1)**

YOLO - Overview



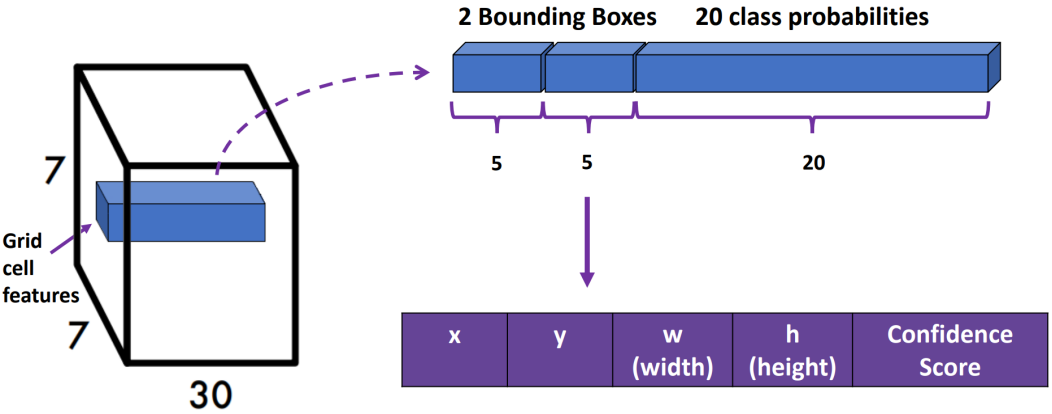
Source: WAIC DL4CV, "Object Detection and Segmentation" (Yariv, 2022), sl. 6; image: Redmon et al., CVPR 2016

YOLO - Overview

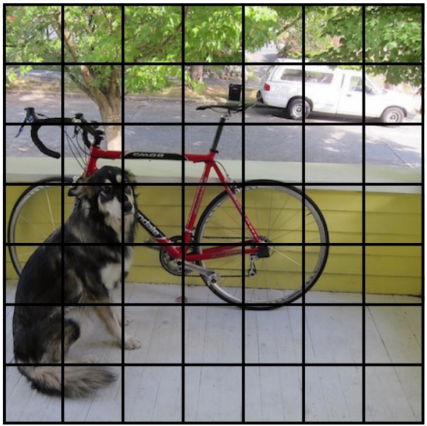


Source: WAIC DL4CV, "Object Detection and Segmentation" (Yariv, 2022), sl. 7; image: Redmon et al., CVPR 2016

YOLO - Overview

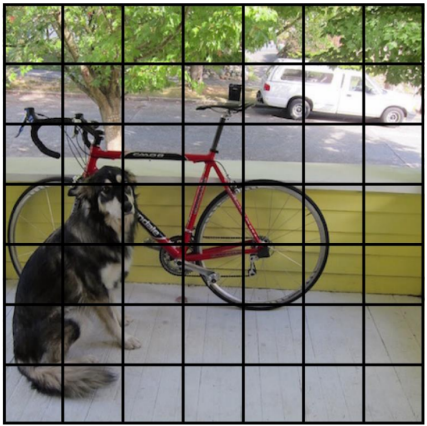


Source: WAIC DL4CV, "Object Detection and Segmentation" (Yariv, 2022), sl. 12; image: Redmon et al., CVPR 2016



Source: WAIC DL4CV, “Object Detection and Segmentation” (Yariv, 2022), sl. 8; image: Redmon et al., CVPR 2016

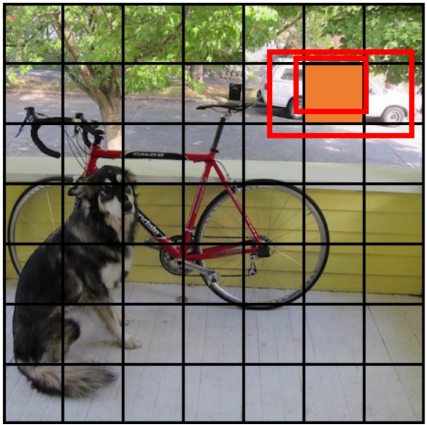
Each cell predicts



Source: WAIC DL4CV, “Object Detection and Segmentation” (Yariv, 2022), sl. 8; image: Redmon et al., CVPR 2016

Each cell predicts

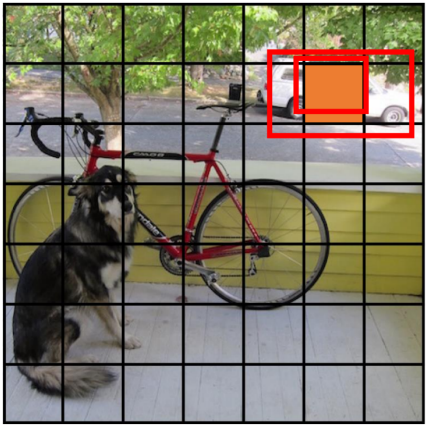
- $B = 2$ bounding boxes (x, y, w, h)+ confidence score



Source: WAIC DL4CV, "Object Detection and Segmentation" (Yariv, 2022), sl. 10; image: Redmon et al., CVPR 2016

Each cell predicts

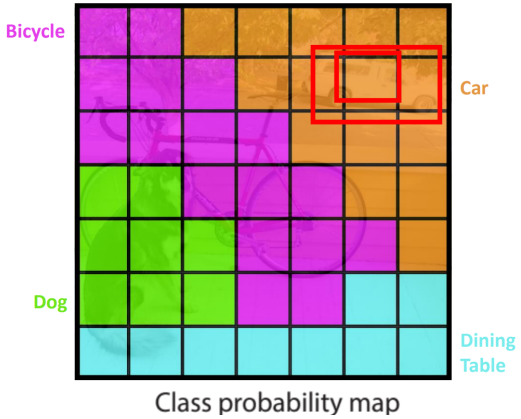
- $B = 2$ bounding boxes (x, y, w, h)+ confidence score
- $C = 20$ class probabilities



Source: WAIC DL4CV, "Object Detection and Segmentation" (Yariv, 2022), sl. 10; image: Redmon et al., CVPR 2016

Each cell predicts

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



Source: WAIC DL4CV, "Object Detection and Segmentation" (Yariv, 2022), sl. 11; image: Redmon et al., CVPR 2016

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 Bboxs)



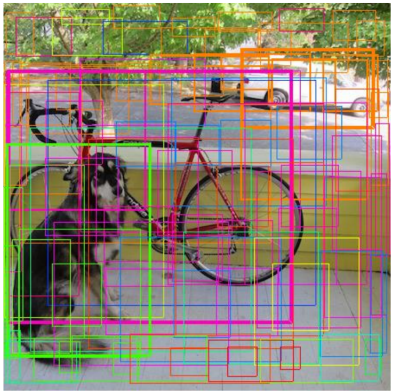
S × S grid on input

Source: WAIC DL4CV, "Object Detection and Segmentation" (Yariv, 2022), sl. 13; image: Redmon et al., CVPR 2016

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 Bboxs)

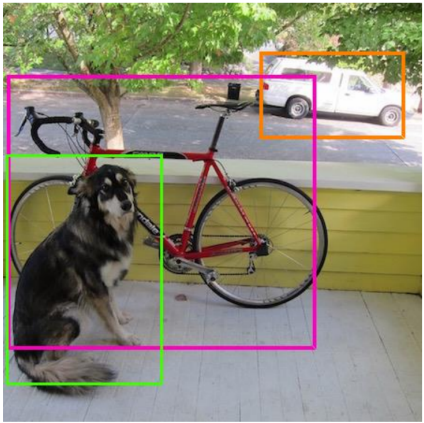


Source: WAIC DL4CV, "Object Detection and Segmentation" (Yariv, 2022), sl. 14; image: Redmon et al., CVPR 2016

Each cell predicts

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

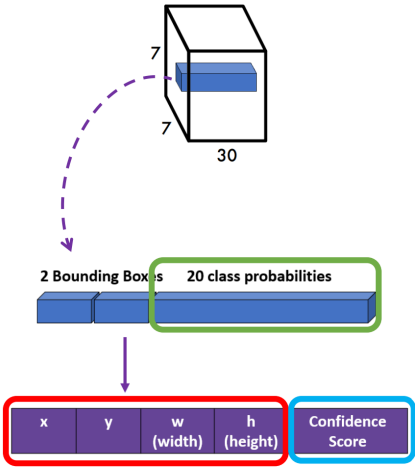
- Apply Non-Maximum Suppression



Source: WAIC DL4CV, "Object Detection and Segmentation" (Yariv, 2022), sl. 15; image: Redmon et al., CVPR 2016

YOLO – Loss function

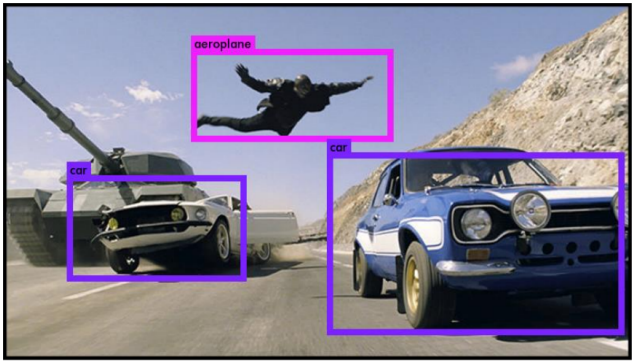
$$\mathcal{L} = \mathcal{L}_{Localization\ Loss} + \mathcal{L}_{Confidence\ Loss} + \mathcal{L}_{Classification\ Loss}$$



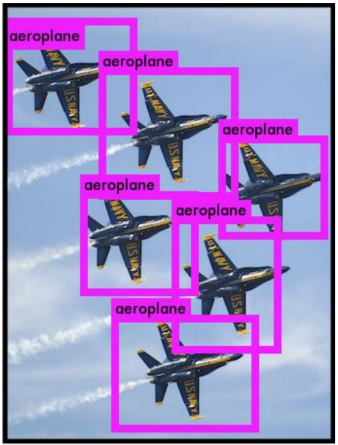
Source: WAIC DL4CV, "Object Detection and Segmentation" (Yariv, 2022), sl. 19; image: Redmon et al., CVPR 2016

YOLO - Benefits

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

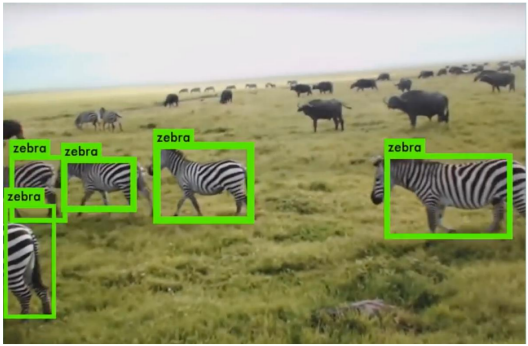


Source: WAIC DL4CV, "Object Detection and Segmentation" (Yariv, 2022), sl. 29; image: Redmon et al., CVPR 2016



Source: WAIC DL4CV, "Object Detection and Segmentation" (Yariv, 2022), sl. 29; image: Redmon et al., CVPR 2016

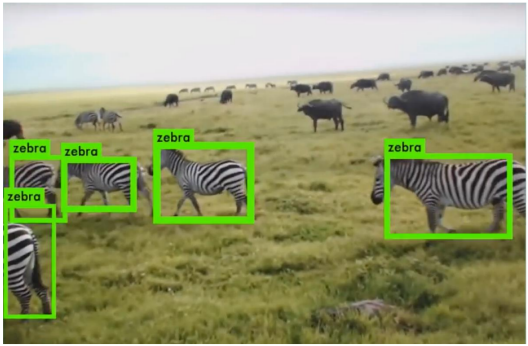
YOLO - Limitations



Source: WAIC DL4CV, "Object Detection and Segmentation" (Yariv, 2022), sl. 30; image: https://pjreddie.com/darknet/yolov1/

YOLO - Limitations

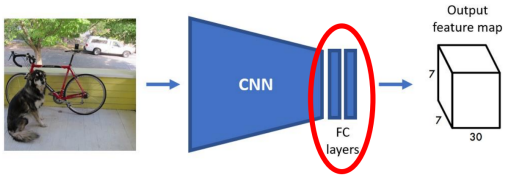
- Difficult to detect small objects
- Coarse predictions



Source: WAIC DL4CV, "Object Detection and Segmentation" (Yariv, 2022), sl. 30; image: https://pjreddie.com/darknet/yolov1/

YOLO - Limitations

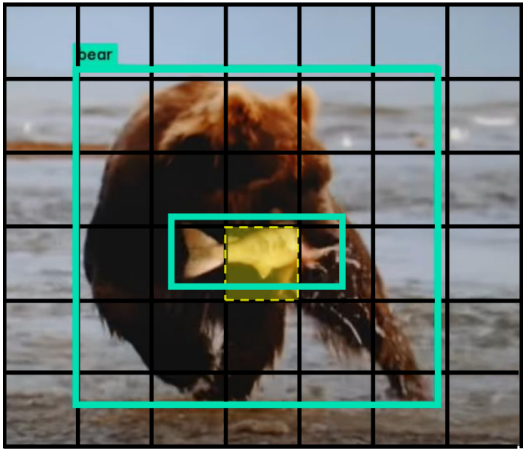
- Difficult to detect small objects
- Coarse predictions
- Fixed input size



Source: WAIC DL4CV, "Object Detection and Segmentation" (Yariv, 2022), sl. 32; image: Redmon et al., CVPR 2016

YOLO - Limitations

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

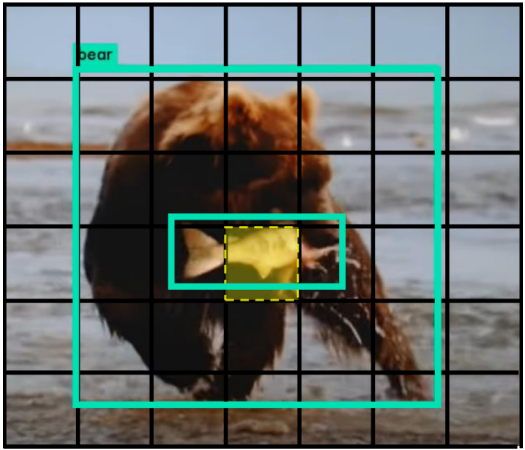


Source: WAIC DL4CV, "Object Detection and Segmentation" (Yariv, 2022), sl. 33; image: https://pjreddie.com/darknet/yolov1/

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)

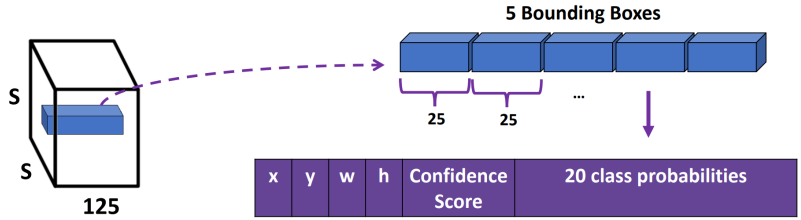


Source: WAIC DL4CV, "Object Detection and Segmentation" (Yariv, 2022), sl. 33; image: https://pjreddie.com/darknet/yolov1/

Object Detection: **YOLOv2 / YOLO9000**

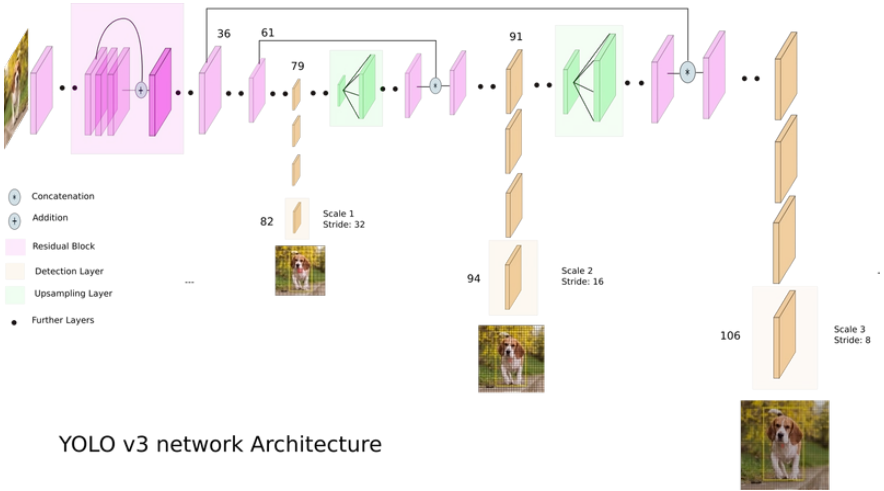
YOLOv2 / YOLO9000 (2017)

- Introduced **anchor boxes**, **batch normalization**, and **high-resolution classifier fine-tuning** (448×448); detection runs at **416×416** to give an odd 13×13 grid
- Removed fully connected layers for a fully convolutional architecture
- Each grid cell predicts class probabilities for each anchor box
- Achieved a significant mAP improvement: ∼13-point increase to 76.8% on PASCAL VOC 2007 (at 416×416)
- Still struggles with small objects: anchor boxes help, but not enough



Source: WAIC DL4CV, "Object Detection and Segmentation" (Yariv, 2022), sl. 35; image: Redmon & Farhadi, "YOLO9000", CVPR 2017

Object Detection: **YOLOv3**



YOLO v3 network Architecture

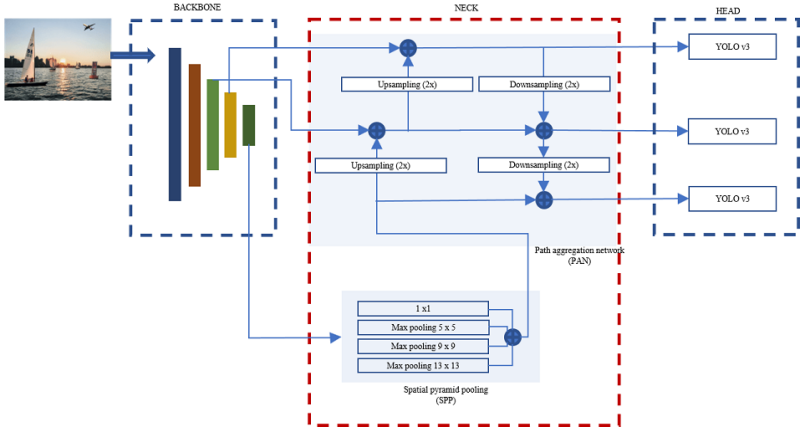
Source: A. Kathuria, "What's new in YOLO v3?", Towards Data Science, 2018.

YOLOv3 (cont.)

- **YOLOv3 (2018):**
  - Uses Darknet-53 backbone (deeper, more powerful CNN)
  - Multi-scale detection via Feature Pyramid Networks (FPN)
  - Achieves ∼33 AP (57.9 AP@50) on the 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: **YOLOv4**

YOLOv4



Source: MathWorks, "Getting Started with YOLO v4", MATLAB Computer Vision Toolbox documentation.

YOLOv4 (cont.)

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 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.

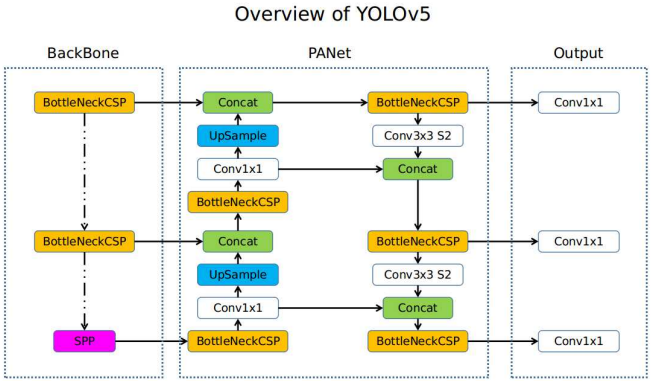
YOLOv4 (cont.)

Performance:

- 43.5% AP (65.7% AP@50) on COCO.
- ∼65 FPS on a Tesla V100, i.e. real-time.

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

Object Detection: **YOLOv5**



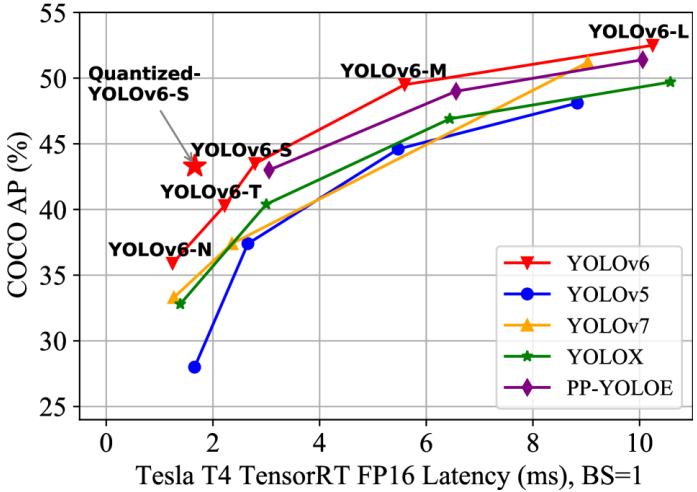
Source: "Overview of YOLOv5" diagram by GitHub user seekFire, ultralytics/yolov5 issue #280, 2020.

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: **YOLOv6 and YOLOv7**

YOLOv6 and YOLOv7



Source: Li et al., "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications", arXiv:2209.02976, Fig. 1.

YOLOv6 and YOLOv7 (cont.)

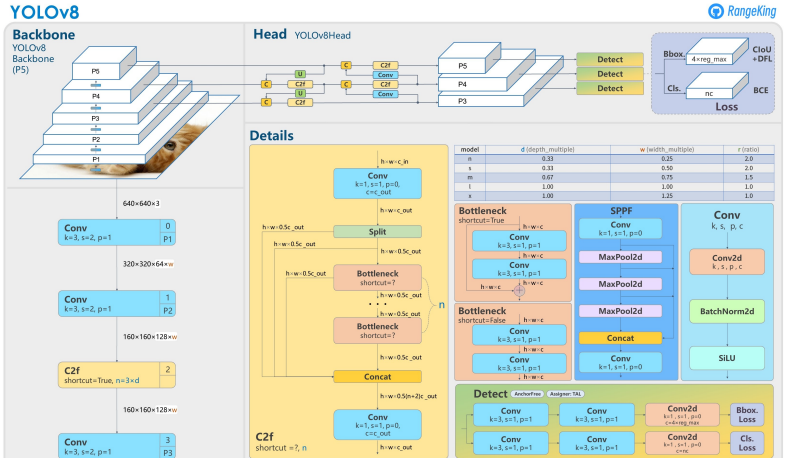
YOLOv6 (2022) Highlights:

- Released by Meituan with industrial deployment as the stated target, not by the original YOLO authors.
- **Anchor-free:** boxes are predicted from anchor points, so the hand-tuned anchor priors of v2–v5 are gone.
- **Backbone and neck:** EfficientRep for the small models, CSPStackRep blocks for the larger ones, with a Rep-PAN neck. Both re-parameterise into plain convolutions at inference time.
- **Head:** an efficient decoupled head, with Task Alignment Learning deciding which prediction is matched to which ground-truth box.
- **Losses:** VariFocal for classification, SIoU or GIoU for box regression, plus self-distillation on both tasks.
- **Performance:** YOLOv6-N reaches 35.9% AP on COCO at 1234 FPS on a Tesla T4; YOLOv6-S 43.5% AP at 495 FPS; YOLOv6-M/L 49.5%/52.3% AP.

YOLOv7 (2022) Highlights:

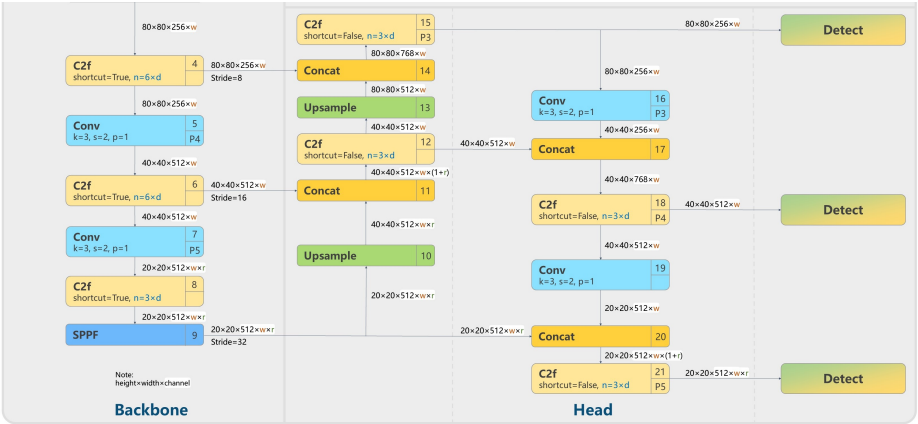
- From the YOLOv4 authors, built around *trainable bag-of-freebies*: methods that cost training time but nothing at inference.
- **E-ELAN:** an extended layer-aggregation block that widens cardinality by expand, shuffle and merge, without disturbing the gradient path.
- **Compound scaling** for concatenation-based models, scaling block depth and transition width together so the two stay matched.
- **Planned re-parameterised convolution:** RepConv with its identity branch removed, so it can be placed alongside residual and concatenation connections.
- **Deep supervision:** an extra auxiliary head is trained on coarse labels assigned by the lead head, then dropped at inference.
- **Performance:** 56.8% AP, the best of any real-time detector running at 30 FPS or more on a V100; YOLOv7-E6 gives 55.9% AP at 56 FPS.

Object Detection: **YOLOv8**



C2f replaces YOLOv5's C3; the anchor-free `Detect` head is decoupled, with no objectness branch. Source: YOLOv8 structure diagram by GitHub user RangeKing, ultralytics/ultralytics issue #189, 2023 (free reuse).

YOLOv8 (cont.)



The 22 layers in full: backbone (0–9) into the PAN-style head (10–21), feeding three `Detect` heads at strides 8, 16 and 32. Same source as the previous slide.

YOLOv8 (2023) Highlights:

- **Anchor-free** detection architecture
- CSP backbone, FPN+PAN neck
- One codebase for detection, instance segmentation, pose estimation and classification
- COCO mAP from 37.3 (YOLOv8n) to 53.9 (YOLOv8x); YOLOv8l gives 52.9 mAP at 2.39 ms per image on an A100 with TensorRT
- 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 (Programmable Gradient Information) and GELAN (Generalized Efficient Layer 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.
- YOLOv10-S runs 1.8× faster than RT-DETR-R18 at comparable AP.

Beyond v8: YOLOv9–YOLOv12 (cont.)

YOLOv11 (Sept 2024):

- Utilizes C3k2 and C2PSA blocks with a refined multi-scale detection head.
- Supports oriented bounding boxes and segmentation tasks.
- Gains between 0.5 and 2.2 mAP over the matching YOLOv8 model, with 16–42% fewer parameters.

YOLOv12 (Feb 2025):

- Attention-centric design with Area Attention and R-ELAN modules.
- YOLOv12-N reports 40.6% mAP on COCO at 1.64 ms per image on a T4.

Object Detection: **Industry Use Case**

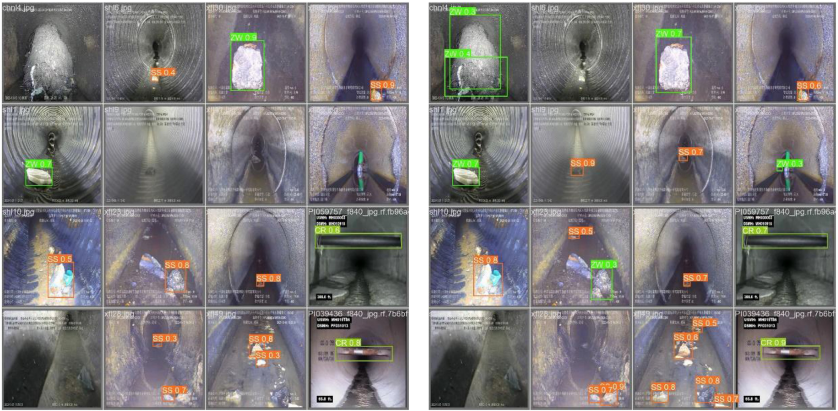
Drainage/Pipeline Internal Defect Detection

- **Architectures:**
  - Improved YOLOv5 (with enhanced CBAM attention, dilated convolution, SIoU loss)
  - EFE-SSD (with an RFB module for small defects)
  - Cycle-GAN + YOLOv5 (for data augmentation and detection)
- **Use Case:**
  - Underground drainage pipeline defect detection (gravel intrusion, obstacles, foreign objects).
  - Internal pipeline wall defect detection with image stitching.
  - Automatic defect detection in CCTV inspection videos.

Drainage/Pipeline Internal Defect Detection (cont.)

- **Benefits:**
  - Improved YOLOv5 reaches **64.49% mAP**, a **5.27%** improvement over the YOLOv5 baseline.
  - EFE-SSD reaches **92.2% mAP** across four sewer defect classes, **2.26 points** above plain SSD.
  - Cycle-GAN enables **5.5**$\times$ sample augmentation from limited data.
- **Research Papers:**
  - Zeng et al. (Dec 2024): enhanced CBAM, switchable atrous and dilated convolution, SIoU loss.
  - Shen et al. (Jun 2023): EFE-SSD, adding an RFB_s module, ECA attention and focal loss to SSD.
  - Chen et al. (2022): Cycle-GAN augmentation, transfer learning from a steel surface-defect dataset, attention mechanism.

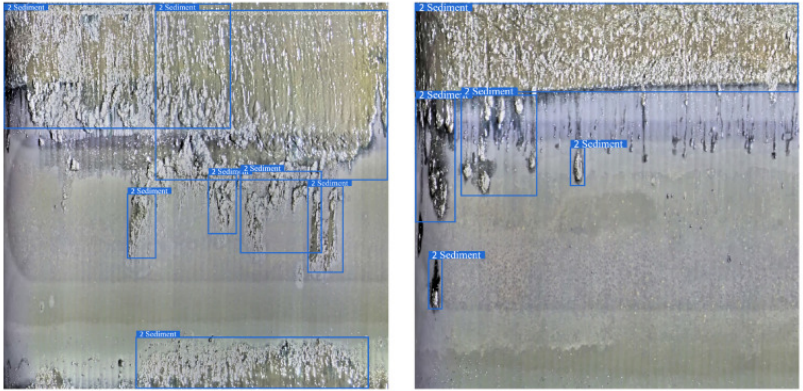
Drainage/Pipeline Internal Defect Detection (cont.)



(a) (b)

The same 16 CCTV inspection frames scored by two detectors, (a) and (b). Each box carries a defect-class code (ZW, SS, CR) and a confidence value; (b) recovers several defects that (a) misses entirely. Source: Zeng et al., "Research on Improved YOLOv5 Pipeline Defect Detection Algorithm", *J. Pipeline Syst. Eng. Pract.* 16(2):04024073, 2025, Fig. 9 (CC BY 4.0).

Drainage/Pipeline Internal Defect Detection (cont.)



YOLOv5 detecting sediment on a stitched drainage-pipe wall. Source: Chen et al., "An Automatic Defect Detection System for Petrochemical Pipeline Based on Cycle-GAN and YOLO v5", *Sensors* 22(20):7907, 2022, Fig. 20 (CC BY 4.0).

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**: RT-DETR surpasses comparable YOLOs in both speed and accuracy; DETR pioneered transformer-based, NMS-free detection.
- **Hybrid Architectures**: Models such as MambaNeXt combine global context modeling with edge efficiency.
- **Prompt-driven Models**: YOLO-World detects classes named by a text prompt with no retraining, an ability it buys with large-scale vision-language pre-training.
- **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**

References & Resources

[1] Selective Search: Uijlings et al. (2013), *IJCV* 104(2)
https://doi.org/10.1007/s11263-013-0620-5

[2] R-CNN: Girshick et al. 2014; Fast R-CNN: Girshick 2015; Faster R-CNN: Ren et al. 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

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

[4] 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

References & Resources (cont.)

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

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

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

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

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

References & Resources (cont.)

[10] YOLOv9: Wang et al. (2024)
https://arxiv.org/abs/2402.13616

[11] YOLOv10: Wang et al. (May 2024)
https://arxiv.org/abs/2405.14458

[12] YOLOv11: Khanam & Hussain (Oct 2024)
https://arxiv.org/abs/2410.17725

[13] YOLOv12: Tian et al. (Feb 2025)
https://arxiv.org/abs/2502.12524

[14] MambaNeXt-YOLO: Lei et al. (June 2025)
https://arxiv.org/abs/2506.03654

References & Resources (cont.)

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

[16] SSD (Single Shot MultiBox Detector): Liu et al. (2016), *ECCV*
https://arxiv.org/abs/1512.02325

[17] FPN (Feature Pyramid Networks): Lin et al. (2017), *CVPR*
https://arxiv.org/abs/1612.03144

[18] PP-YOLO: Long et al. (2020)
https://arxiv.org/abs/2007.12099

[19] DETR (Transformers): Carion et al. (2020)
https://arxiv.org/abs/2005.12872

References & Resources (cont.)

[20] YOLO-World: Cheng et al. (2024), Real-Time Open-Vocabulary Object Detection
https://arxiv.org/abs/2401.17270

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

[22] Assaf Shocher, Shai Bagon, Meirav Galun & Tali Dekel, WAIC DL4CV Deep Learning for Computer Vision: Fundamentals and Applications
https://dl4cv.github.io/

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

Explore Open Source

- https://github.com/pjreddie/darknet
- https://github.com/ultralytics/yolov5
- https://github.com/facebookresearch/detectron2
- https://github.com/ultralytics/ultralytics

Further Reading

- Wikipedia, *You Only Look Once*: https://en.wikipedia.org/wiki/You_Only_Look_Once
- Everingham et al., "The Pascal Visual Object Classes Challenge: A Retrospective", *IJCV* 111(1), 2015, the source of the VOC mAP numbers quoted in this lecture: https://doi.org/10.1007/s11263-014-0733-5