Week 6

Detection & Segmentation

[ECEA0649/ECE40049] Deep Learning for Image Processing | Spring 2026

Kihyun Na (Research Professor)

BK21 AI Project Group & Institute for Information and Communication Technology,
Handong Global University

HANDONG GLOBAL UNIVERSITY

Curriculum

** Adjusted based on survey results*

Wk 1	OT + Introduction	**Wk 9**	Foundation Models (CLIP, SAM) + Paper #1
Wk 2	DL Fundamentals Review	**Wk 10**	Diffusion Models + Paper #2
Wk 3	CNN	**Wk 11**	Conditional Generation + Paper #3
Wk 4	From Sequence Modeling to Transformer	**Wk 12**	Vision-Language Models + Paper #4
Wk 5	Transformer in Vision	**Wk 13**	VLM Applications + Paper #5
Wk 6	Detection & Segmentation (Today)	**Wk 14**	Video Understanding + Paper #6
Wk 7	Self-Supervised Learning	**Wk 15**	Embodied AI & Robot Vision + Paper #7
Wk 8	Review Literacy + Role Explanation	**Wk 16**	Miniconference (Final Project)

Lecture Lecture + Paper Miniconference

Today's Agenda

01 **Task Definition** — 10 min

Vision task spectrum, Evaluation metrics

02 **Detection** — 35 min

R-CNN → Faster-RCNN → FPN → YOLO → DETR → Grounding DINO

03 **Segmentation** — 35 min

FCN → U-Net → Mask R-CNN → Mask2Former → SAM → Grounded SAM

04 **Wrap-Up** — 10 min

Timeline, resources, preview of Wk7 SSL & Wk9 Foundation

Part 1

Task Definition

From bounding boxes to pixel-level classification

One Scene, Five Questions

What can we ask a model to do with an image?

Classification

What is in the image?

Single label per image

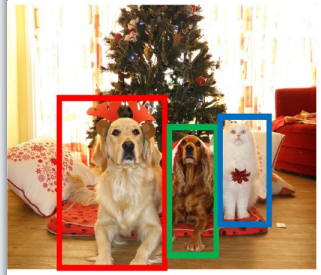
CAT

Object Detection

Where are objects?

Bounding boxes + labels

"cat at (x,y,w,h)"



DOG, **DOG**, **CAT**

Instance Segmentation

Separate each object

Per-instance masks

"cat #1 / cat #2"

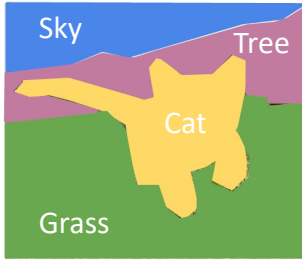
DOG, **DOG**, **CAT**

Semantic Segmentation

Label every pixel

No instance distinction

"sky / road / tree"



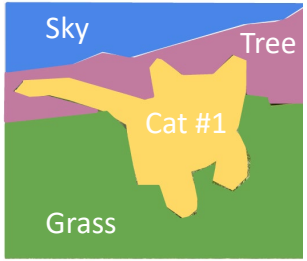
GRASS, **CAT**, **TREE**, **SKY**

Panoptic Segmentation

Instance Seg. + Semantic Seg.

Things (Countable) + Stuff

"cat #1+sky +tree"



GRASS, **CAT**, **TREE**, **SKY**

Benchmarks & Datasets

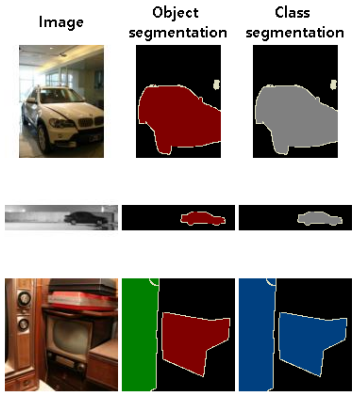
PASCAL VOC (2007–12)

20 classes, ~10K images

Usage: Det. + Seg.

Metric: AP50

R-CNN → YOLO v1 era



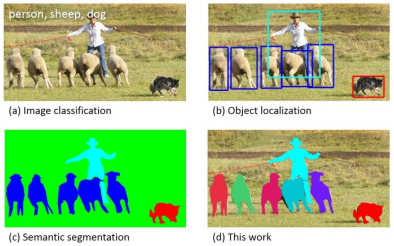
MS COCO (2014–17)

133 classes (80 things + 53 stuff), 118K train images

Usage: Det. + Seg. + Panoptic

Metric: AP@[.50:.95]

Current standard since 2015



Objects365 / LVIS

[Objects365]
365 classes, 2M images, 30 M Bounding Boxes

Usage: Large-scale Det. pretraining

[Large Vocabulary Instance Segmentation (LVIS)]
1203 classes on COCO images

Usage: Long-tail / Rare-class evaluation

SA-1B → SA-Co (2023–25)

SA-1B: 11M images, 1.1B masks

SA-V: 51K videos, 643K masklets

SA-Co: 4M concept labels

Foundation model scale

270K unique concepts vs

COCO's 80 classes

Part 2

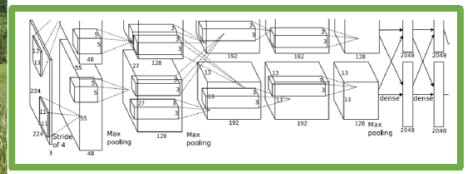
Object Detection

From sliding windows to end-to-end prediction

Object Detection: Single Object

This image is CC0 public domain

Often pretrained on ImageNet (Transfer learning)



Vector: 4096

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

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

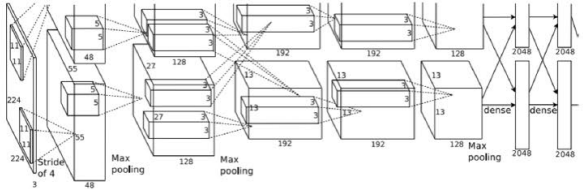
"Where"

Treat localization as a regression problem!

Problem: Images can have more than one object!

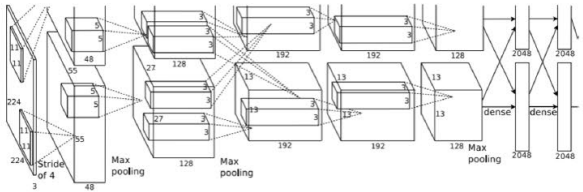
Object Detection: Multiple Objects

Need different numbers of outputs per image



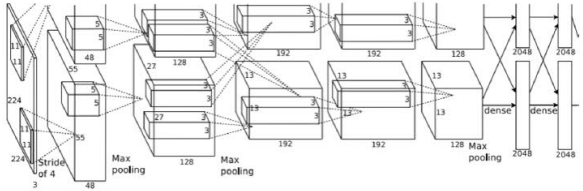
CAT: (x, y, w, h)

4 numbers



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

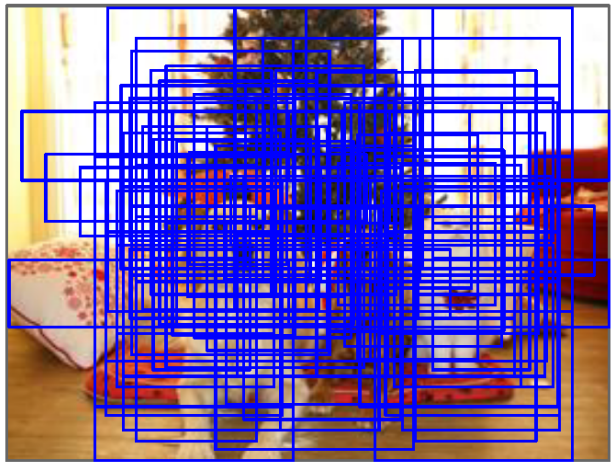
12 numbers



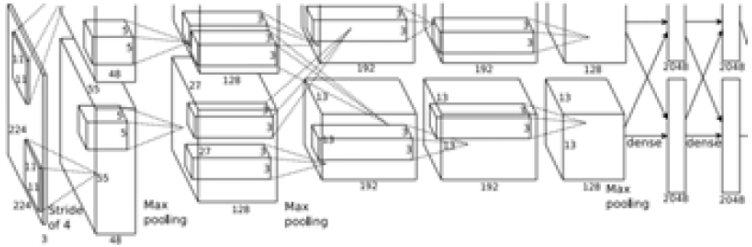
DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Many numbers!

Detecting Multiple Objects



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

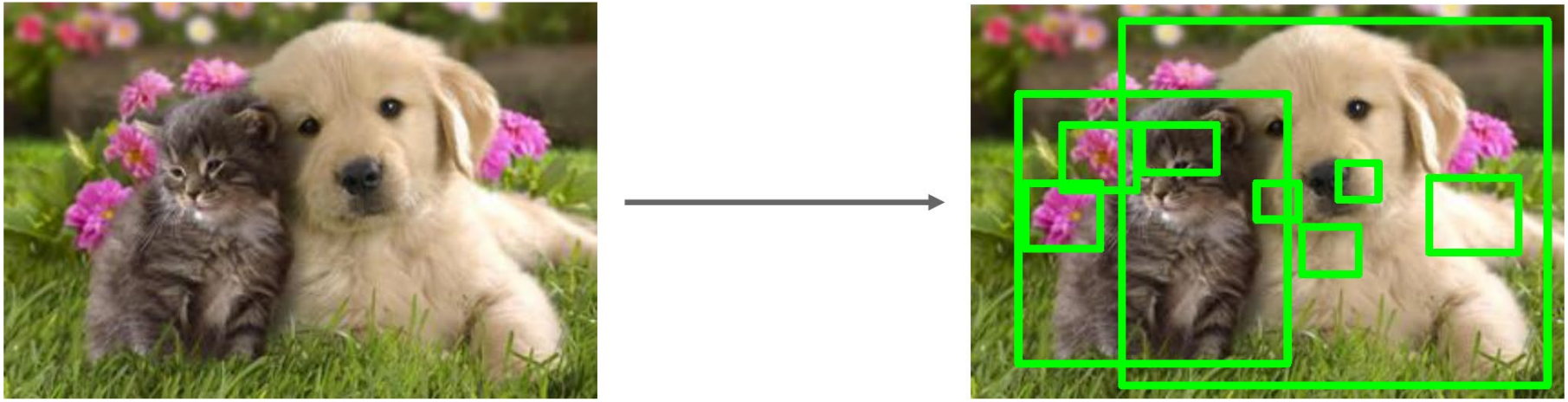


Dog? NO
Cat? YES
Background? NO

Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

Region Proposals: Selective Search

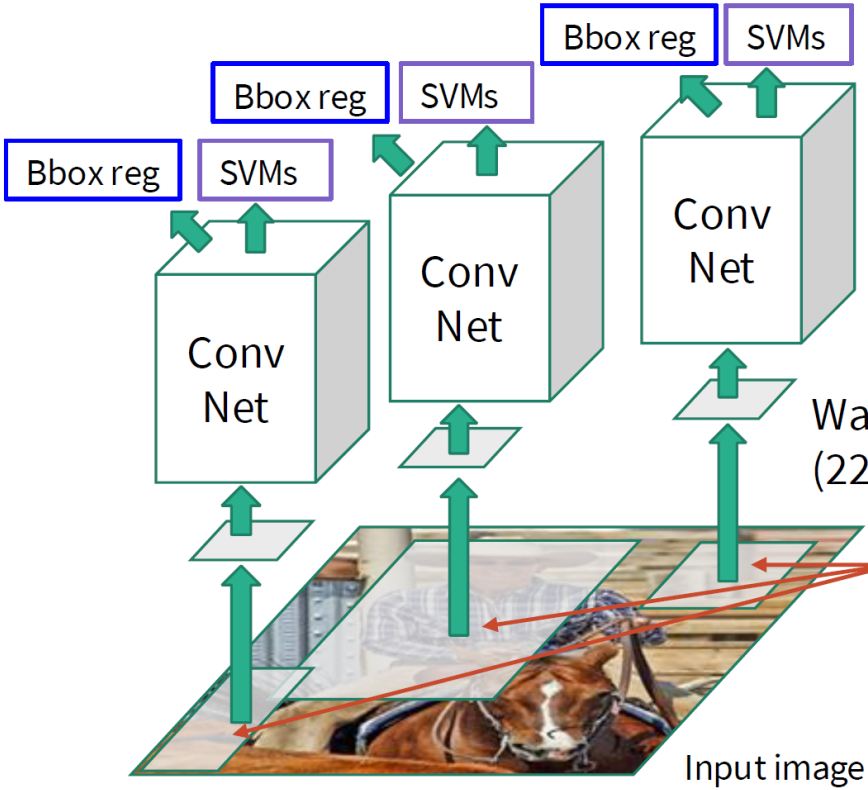
- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU



Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012
Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013
Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014
Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

R-CNN (2014)

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!

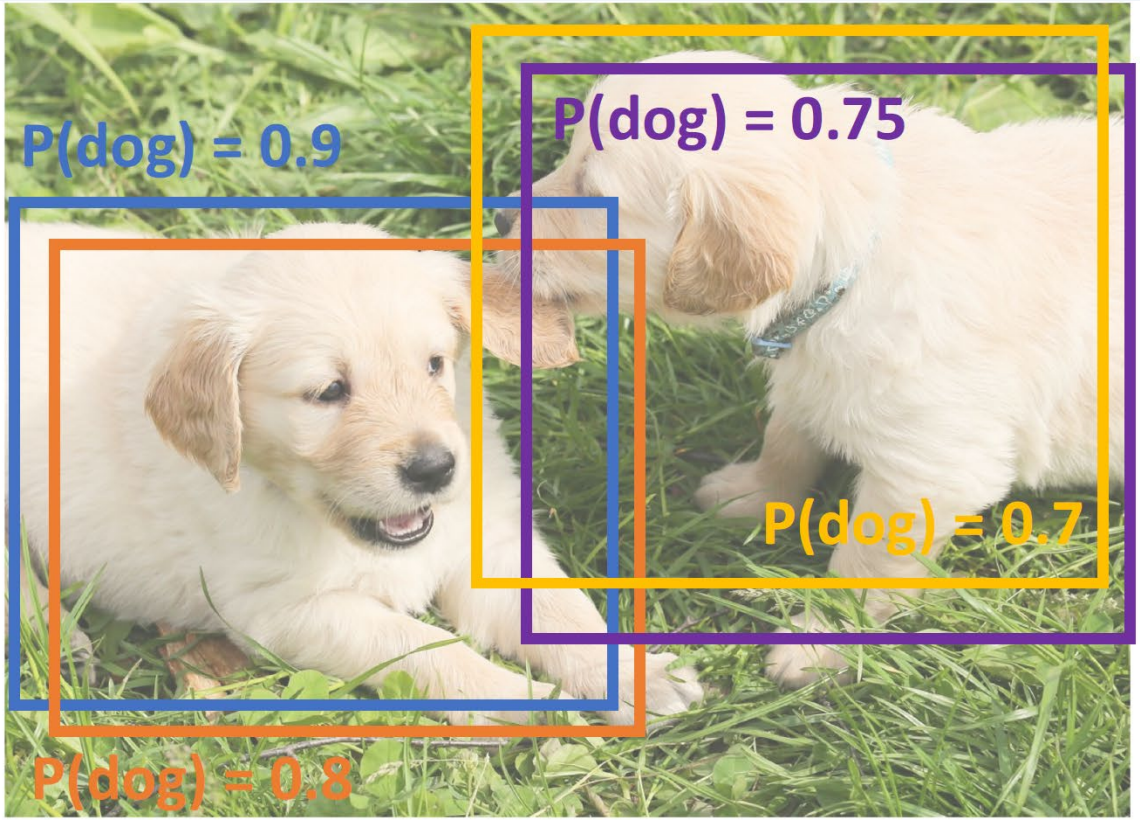
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Overlapping Boxes: Non-maximum Suppression

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



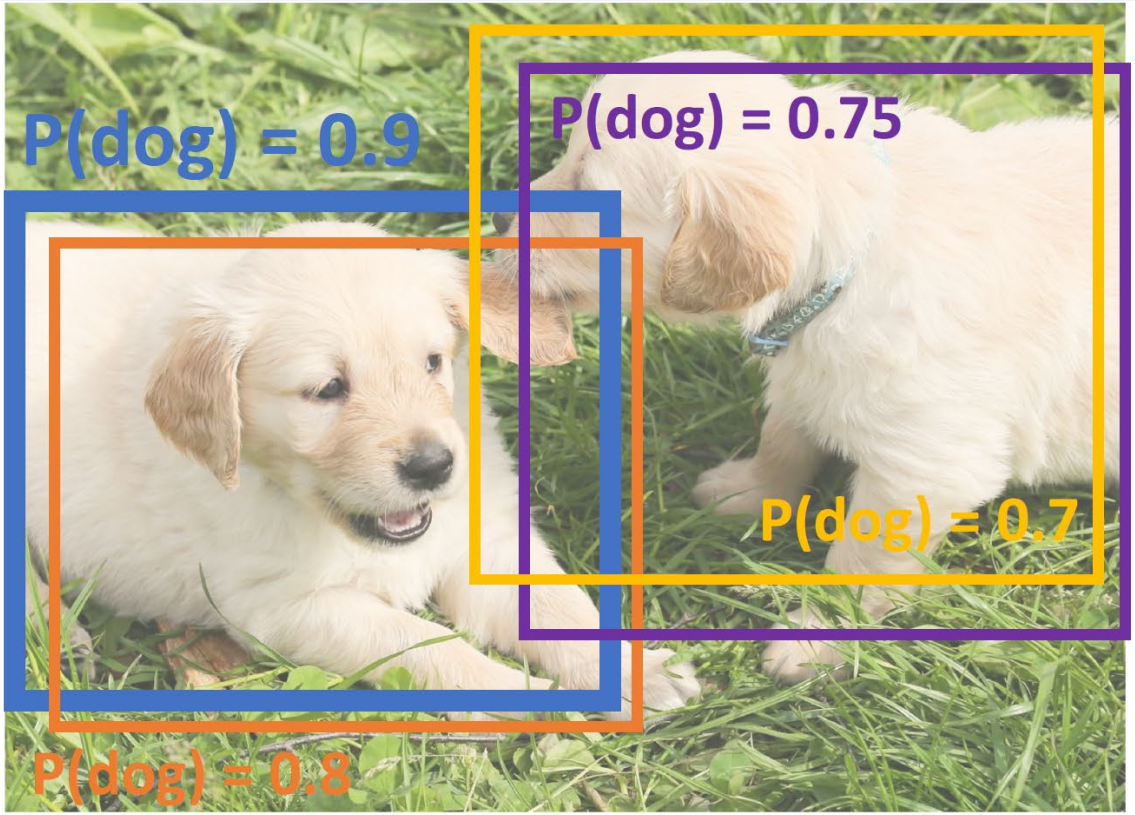
Overlapping Boxes: Non-maximum Suppression

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

IoU(■, ■) = **0.78**
IoU(■, ■) = 0.05
IoU(■, ■) = 0.07



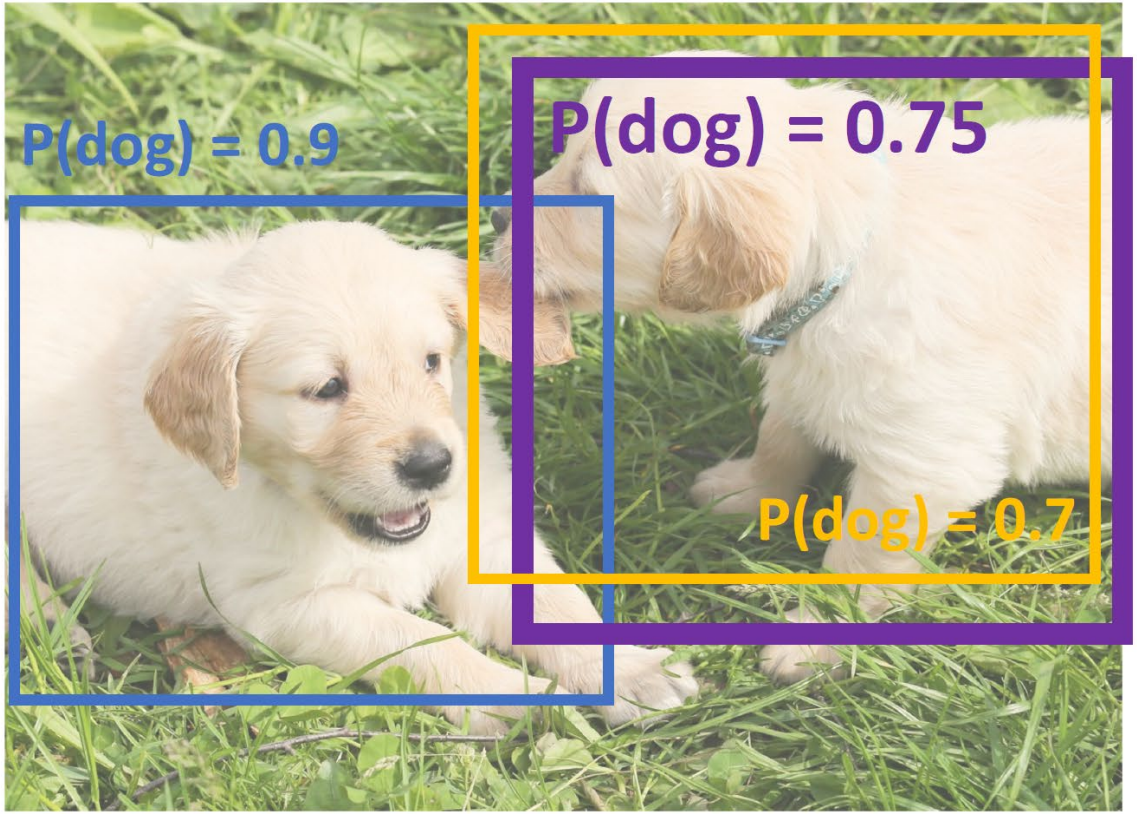
Overlapping Boxes: Non-maximum Suppression

Problem: Object detectors often output many overlapping detections:

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

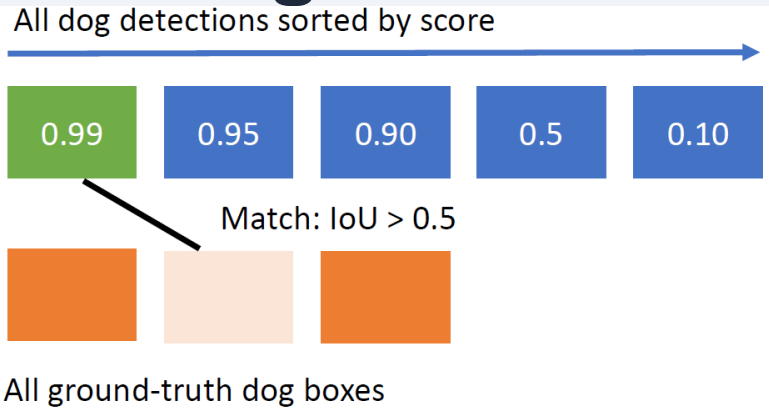
1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

IoU(■, ■) = **0.74**

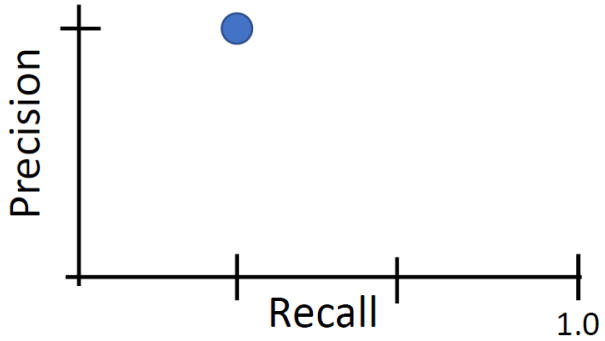


Evaluation Metric: Mean Average Precision

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve

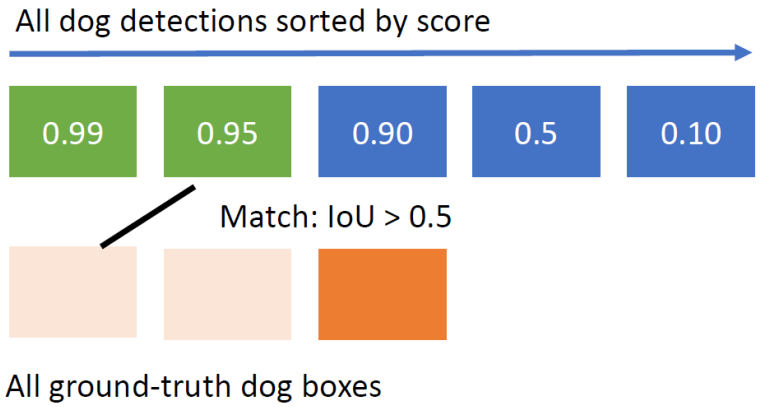


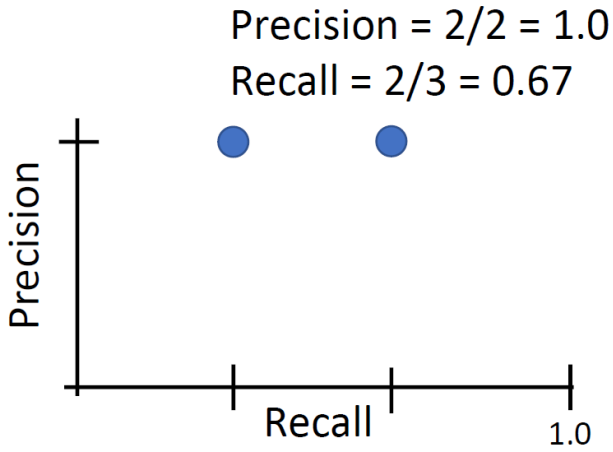
Precision = 1/1 = 1.0
Recall = 1/3 = 0.33



Evaluation Metric: Mean Average Precision

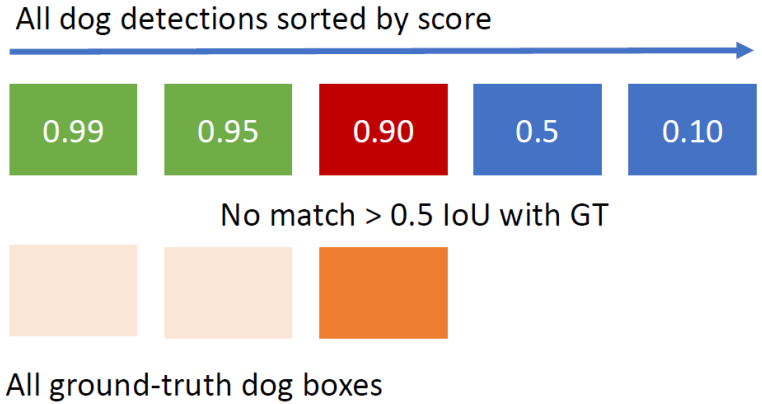
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve

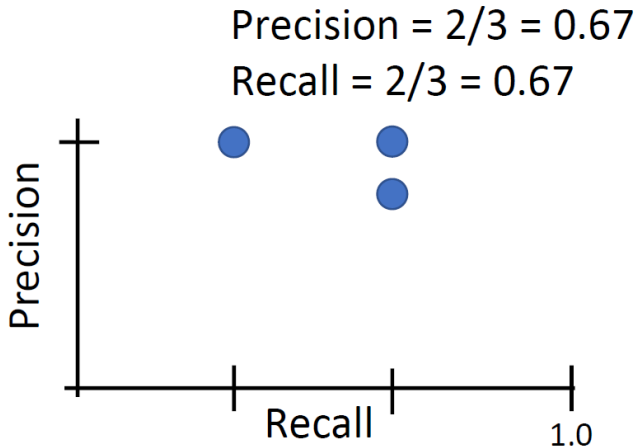




Evaluation Metric: Mean Average Precision

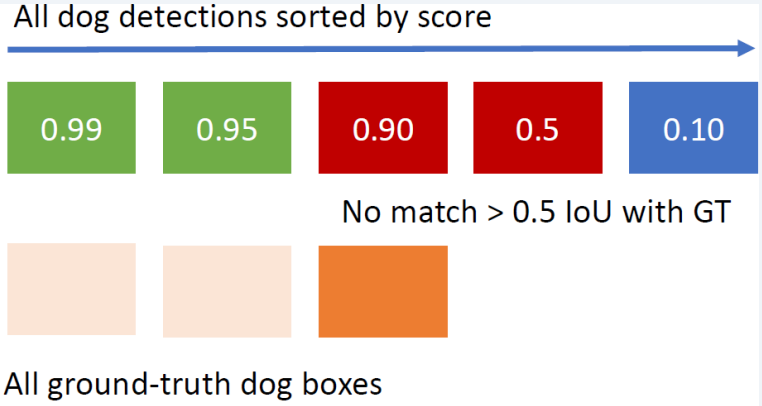
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve

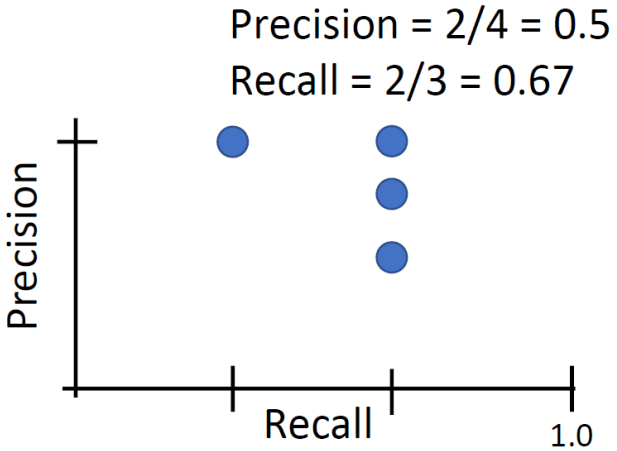




Evaluation Metric: Mean Average Precision

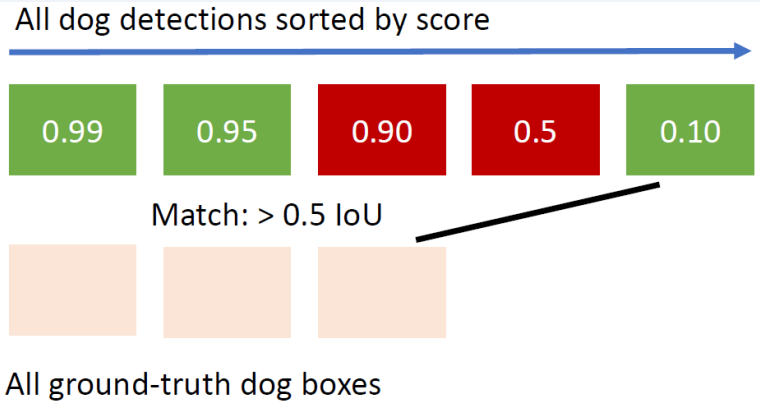
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve

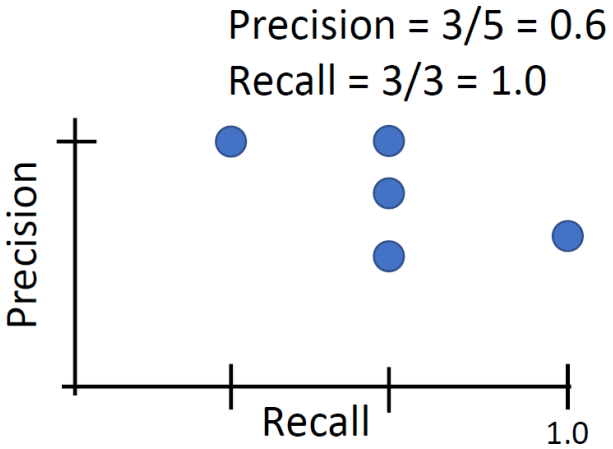




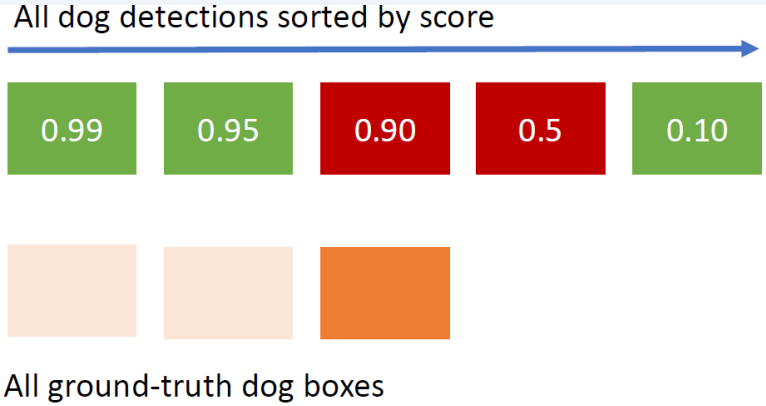
Evaluation Metric: Mean Average Precision

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve



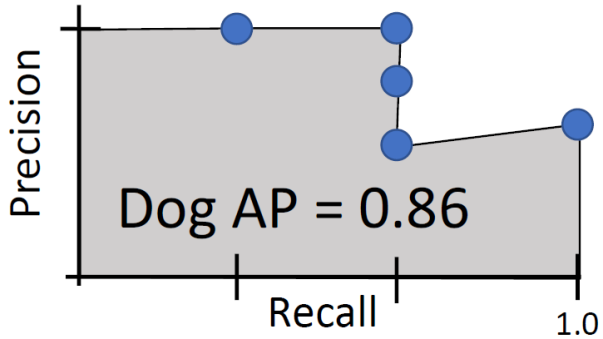


Evaluation Metric: Mean Average Precision



1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve
    2. Average Precision (AP) = area under PR curve

How to get AP = 1.0: Hit all GT boxes with IoU > 0.5, and have no "false positive" detections ranked above any "true positives"



Mean Average Precision (mAP), COCO mAP

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve
    2. Average Precision (AP) = area under PR curve
3. Mean Average Precision (mAP) = average of AP for each category
4. For "COCO mAP": Compute mAP@thresh for each IoU threshold (0.5, 0.55, 0.6, …, 0.95) and take average

mAP@0.5 = 0.77

mAP@0.55 = 0.71

mAP@0.60 = 0.65

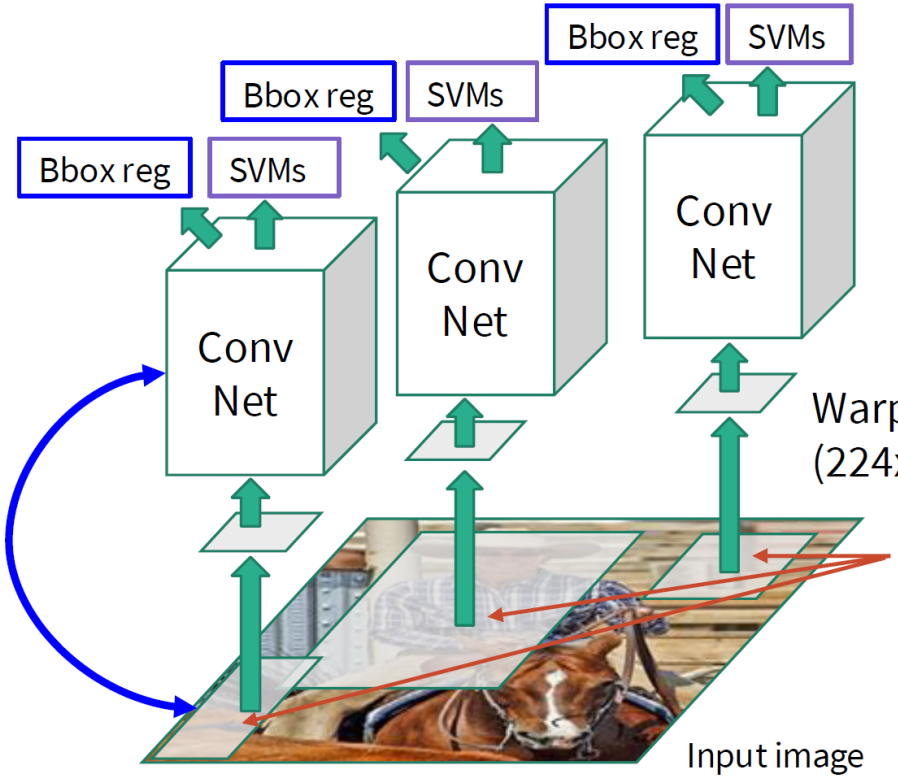
…

mAP@0.95 = 0.2

COCO mAP = 0.4

“Slow” R-CNN (2014)

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)

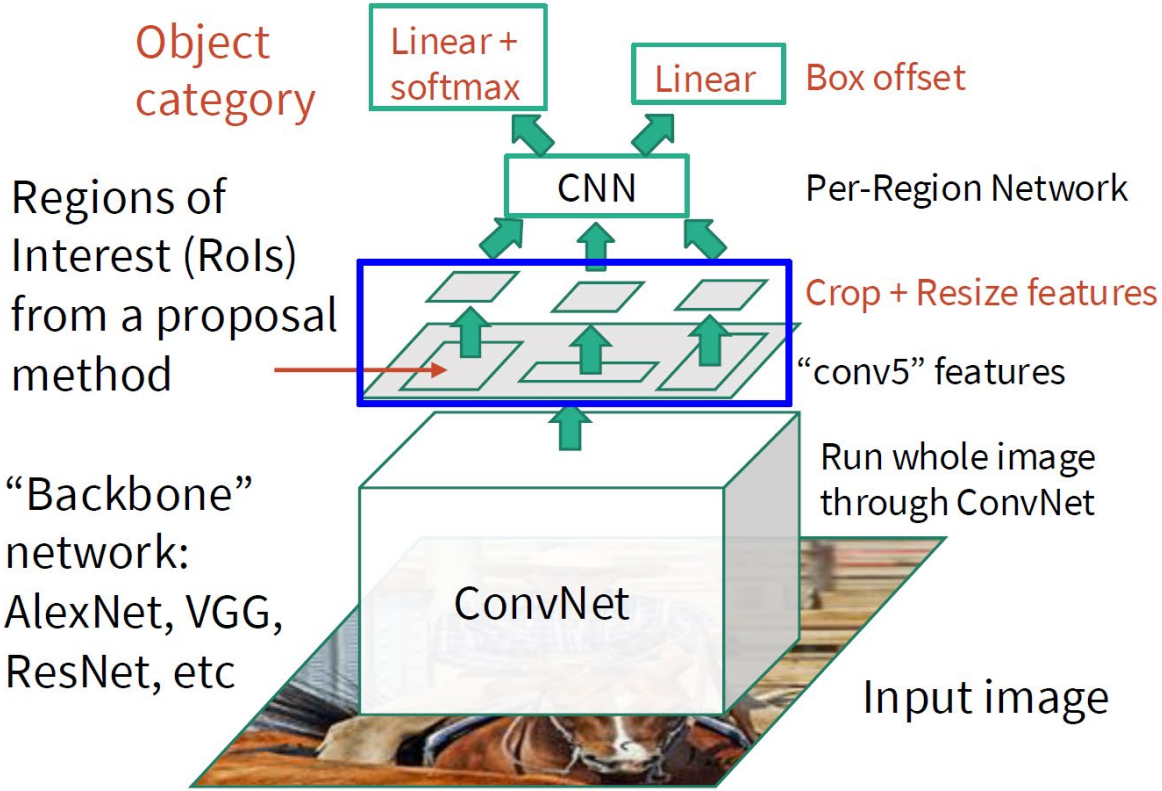
Regions of Interest (RoI) from a proposal method (~2k)

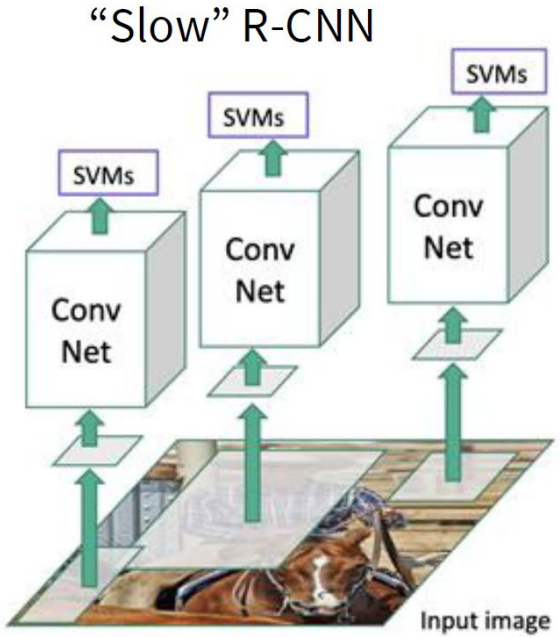
Problem: Very slow! Need to do ~2k independent forward passes for each image!

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.

Fast R-CNN (2015)



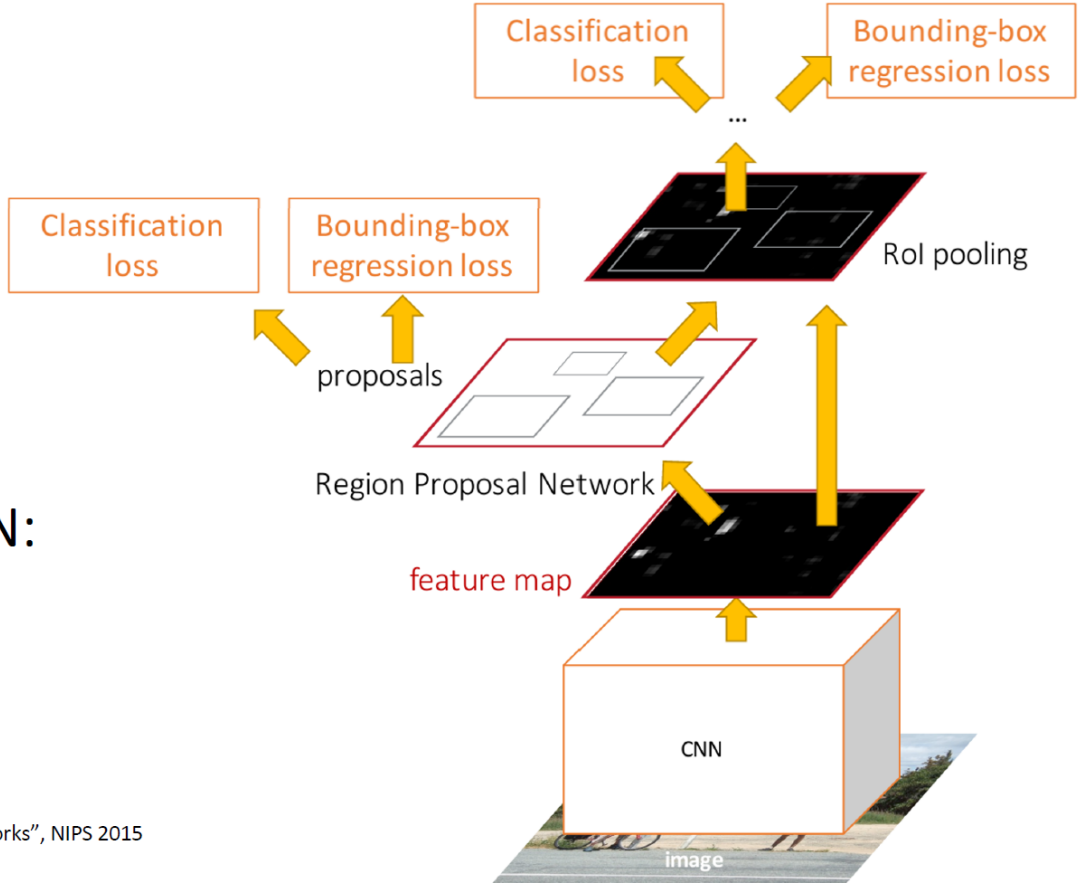


Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Faster R-CNN (2015)

Insert **Region Proposal Network (RPN)** to predict proposals from features

Otherwise same as Fast R-CNN: Crop features for each proposal, classify each one



Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

Region Proposal Network

Imagine an anchor box of fixed size at each point in the feature map

Input Image
(e.g. 3 x 640 x 480)

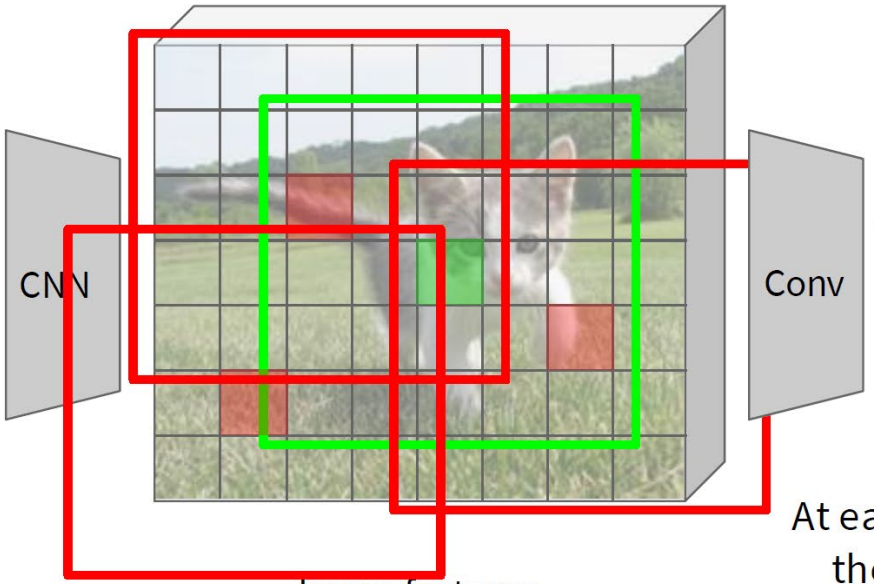


Image features
(e.g. 512 x 20 x 15)

Anchor is an object?
1 x 20 x 15

At each point, predict whether the corresponding anchor contains an object (binary classification)

Region Proposal Network

Imagine an anchor box of fixed size at each point in the feature map

Input Image
(e.g. 3 x 640 x 480)

CNN

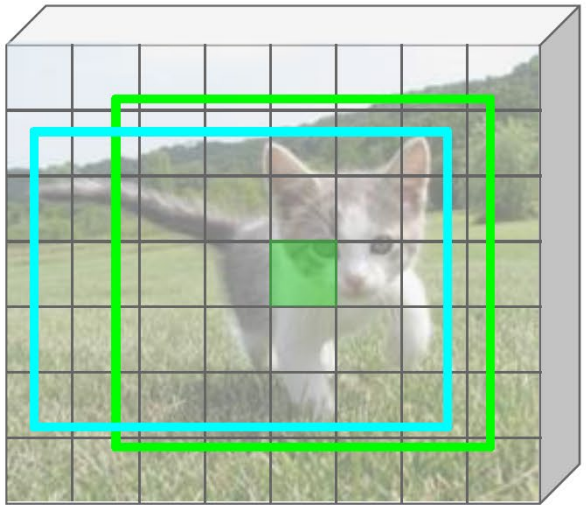


Image features
(e.g. 512 x 20 x 15)

Conv

Anchor is an object?
1 x 20 x 15

Box corrections
4 x 20 x 15

For positive boxes, also predict a corrections from the anchor to the ground-truth box (regress 4 numbers per pixel)

Region Proposal Network

In practice use K different anchor boxes of different size / scale at each point

Input Image
(e.g. 3 x 640 x 480)

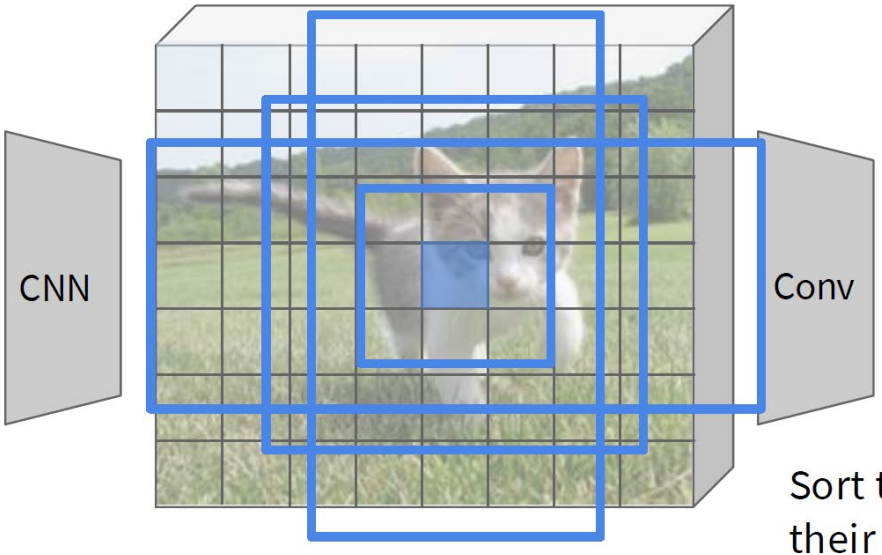


Image features
(e.g. 512 x 20 x 15)

Anchor is an object?
K x 20 x 15

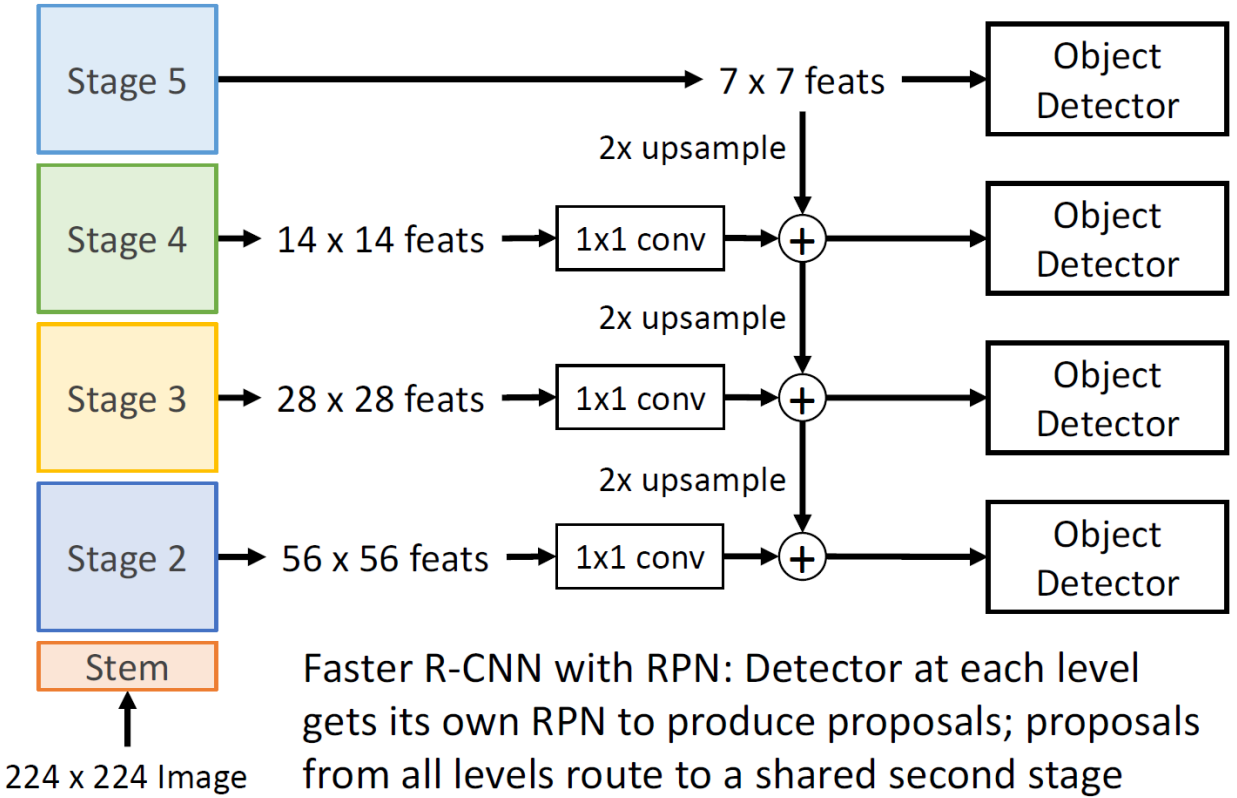
Box transforms
4K x 20 x 15

Sort the K*20*15 boxes by their "objectness" score, take top ~300 as our proposals

Dealing with Scale: Feature Pyramid Network

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



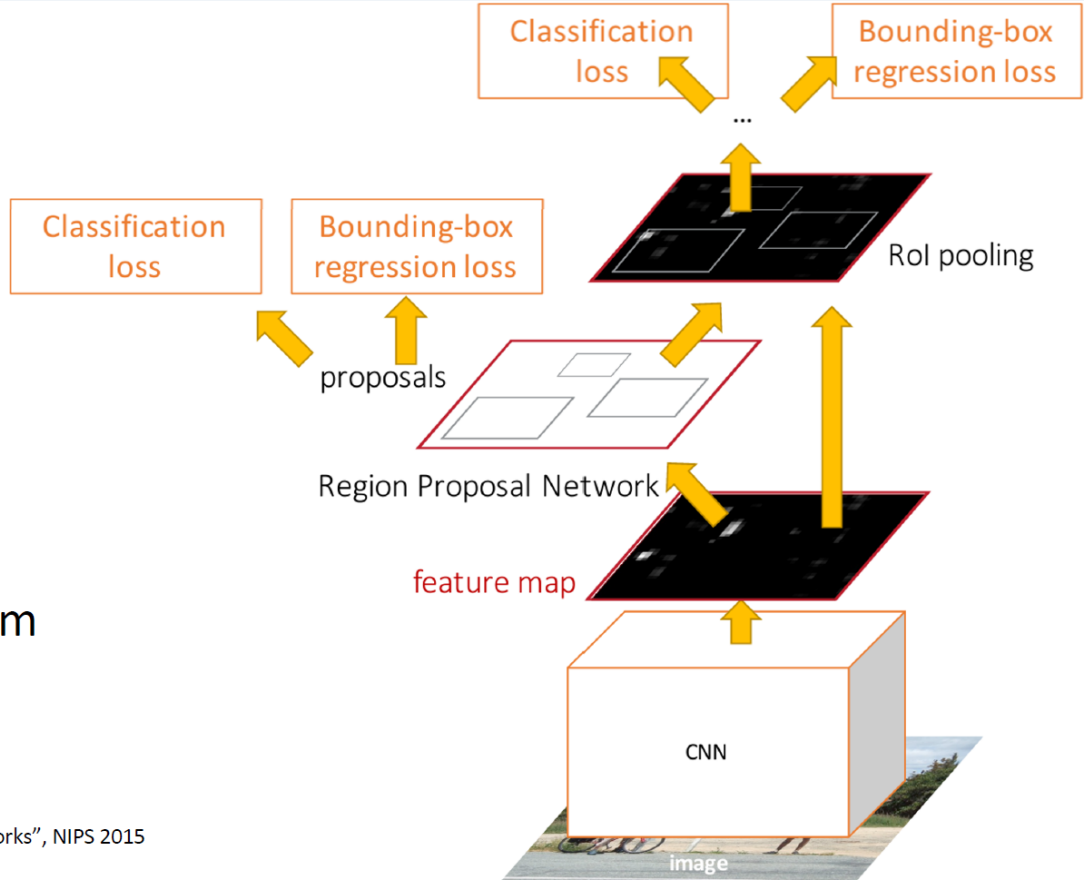
Faster R-CNN with RPN: Detector at each level gets its own RPN to produce proposals; proposals from all levels route to a shared second stage

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

Faster R-CNN: Learnable Region Proposals

Jointly train with 4 losses:

1. **RPN classification**: anchor box is object / not an object
2. **RPN regression**: predict transform from anchor box to proposal box
3. **Object classification**: classify proposals as background / object class
4. **Object regression**: predict transform from proposal box to object box



Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

R-CNN Family: Three Generations

All use ImageNet-pretrained backbone → fine-tune for detection (transfer learning is the norm)

R-CNN (2014)

Selective Search → ~2000 proposals
CNN feature extraction per region
SVM classifier + bbox regressor

Problem: Very slow
~47s per image (GPU)

Fast R-CNN (2015)

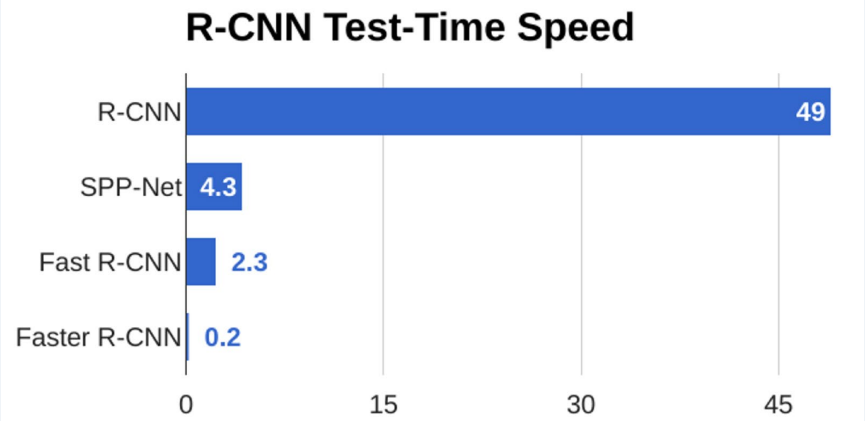
Shared CNN backbone
RoI Pooling from feature map
Multi-task loss (cls + bbox)

Problem: Still uses Selective Search
~2s per image

Faster R-CNN (2016)

Region Proposal Network (RPN)
Anchors replace Selective Search
End-to-end trainable

Breakthrough: ~0.2s per image
Real-time possible



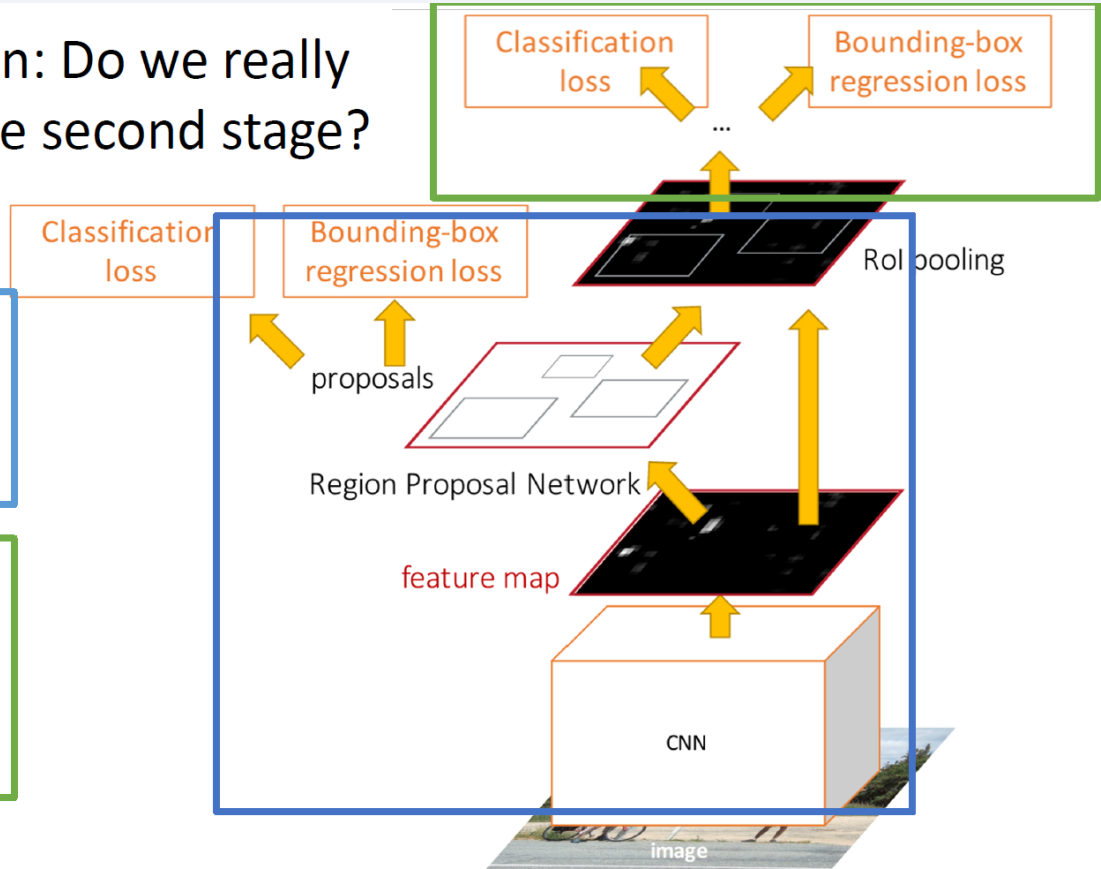
Faster R-CNN: Two-stage Object Detector

Question: Do we really need the second stage?

Faster R-CNN is a **Two-stage object detector**

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

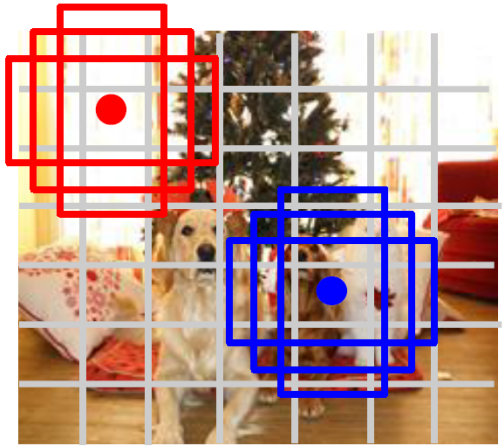
Second stage: Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset



Single-Stage Object Detectors

YOLO, SSD, RetinaNet, FCOS, ...

Input image
3 x H x W



Divide image into grid
7 x 7

Image a set of base boxes centered at each grid cell
Here B = 3

Within each grid cell:

- Regress from each of the B base boxes to a final box with 5 numbers: (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)
- Looks a lot like RPN, but category-specific!

Output:
7 x 7 x (5 * B + C)

Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016
Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

YOLO(2016): You Only Look Once

Core Concept & Loss

Single-shot: one forward pass, no proposals

Divide image into S×S grid

Each cell predicts B boxes + confidence + C classes

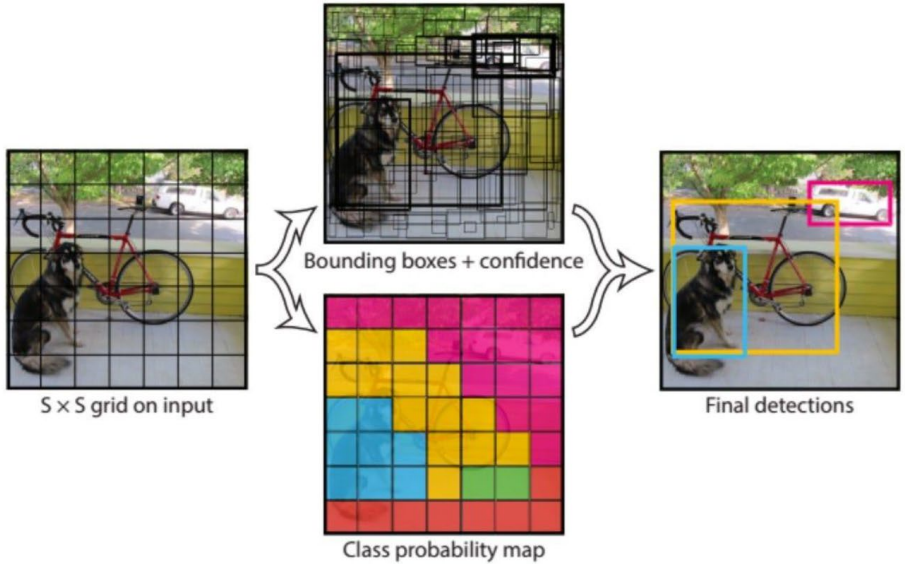
Multi-part loss:

λ_{coord} · (box position + size)

+ confidence (obj / no-obj)

+ classification

Detection as regression, not proposal+classify



Redmon et al. "You only look once: unified, real-time object detection (2015)."

YOLO(2016): You Only Look Once

Core Concept & Loss

Single-shot: one forward pass, no proposals

Divide image into S×S grid

Each cell predicts B boxes + confidence + C classes

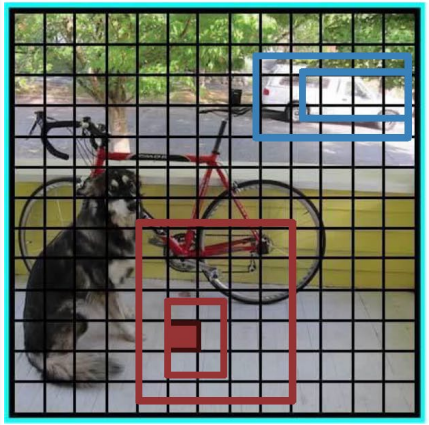
Multi-part loss:

λ_{coord} · (box position + size)

+ confidence (obj / no-obj)

+ classification

Detection as regression, not proposal+classify



SxS Grid

For each box output:

- P(object): probability that the box contains an object
- B bounding boxes (x, y, h, w)
- P(class): probability of belonging to a class

B=2

Redmon et al. "You only look once: unified, real-time object detection (2015)."

YOLO(2016): You Only Look Once

Core Concept & Loss

Single-shot: one forward pass, no proposals

Divide image into S×S grid

Each cell predicts B boxes + confidence + C classes

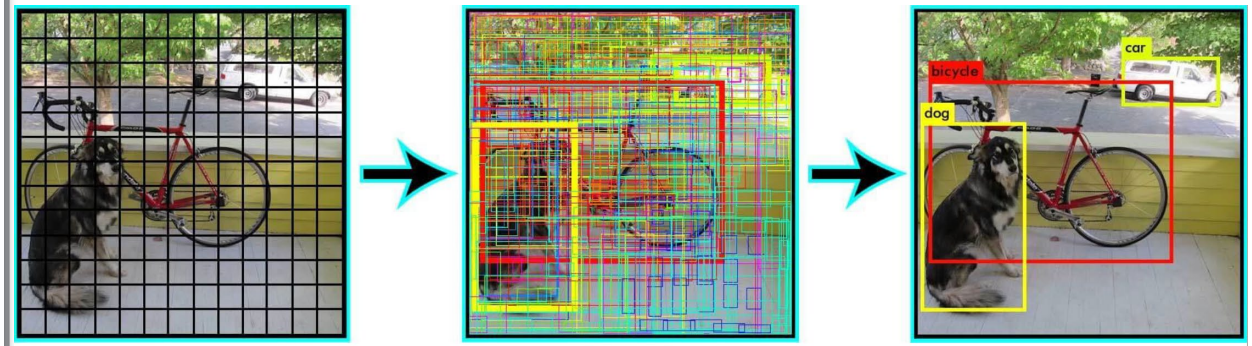
Multi-part loss:

λ_{coord} · (box position + size)

+ confidence (obj / no-obj)

+ classification

Detection as regression, not proposal+classify



SxS Grid

Redmon et al. "You only look once: unified, real-time object detection (2015)."

One-Stage vs Two-Stage & Focal Loss

Two-Stage (Faster R-CNN)

Stage 1: RPN proposes regions

Stage 2: Classify + refine each

Pros:

- High accuracy
- Rich per-region features

Cons:

- Slower (two passes)
- Complex pipeline

One-Stage (YOLO, SSD)

Single pass: predict all boxes at once

Pros:

- Fast (real-time)
- Simple architecture

Cons:

- Class imbalance: ~100K anchors, only ~10 are positive
- → Easy negatives dominate loss
- → Accuracy gap vs two-stage

RetinaNet + Focal Loss

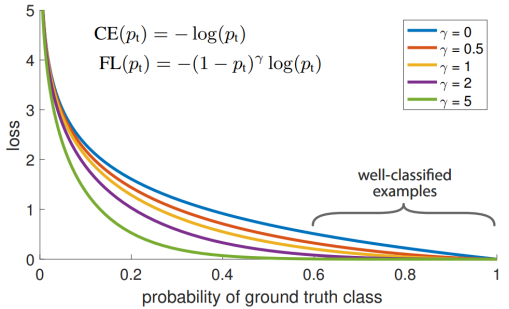
Focal Loss (Lin et al., ICCV 2017)

Down-weights easy negatives, focuses training on hard examples

$\gamma = 2$: ~100× reduction at $p \approx 0.9$

RetinaNet = FPN backbone + Focal Loss

→ First one-stage detector to match two-stage accuracy



Beyond Anchors: Why DETR?

Anchor-Based Limits

Hand-designed anchor sizes/ratios
Scale-specific assignment rules

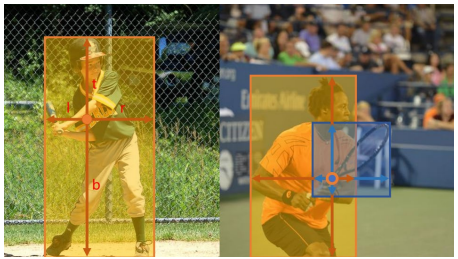
Huge number of anchors (~100K)
Mostly negative → class imbalance
(→ Focal Loss helps but doesn't eliminate)

Many hyperparameters to tune per dataset

NMS & Anchor-Free Attempts

NMS (Non-Max Suppression):
Sort boxes by score → keep highest
→ suppress boxes with IoU > threshold
Sequential, non-differentiable

Anchor-free detectors tried to help:
FCOS (2019): predict from each pixel
→ Simpler, but still need NMS



What if...?

What if we could:

✓ Remove anchors entirely?
✓ Remove NMS?
✓ Use a fixed set of predictions?
✓ Let the model learn to avoid duplicates?

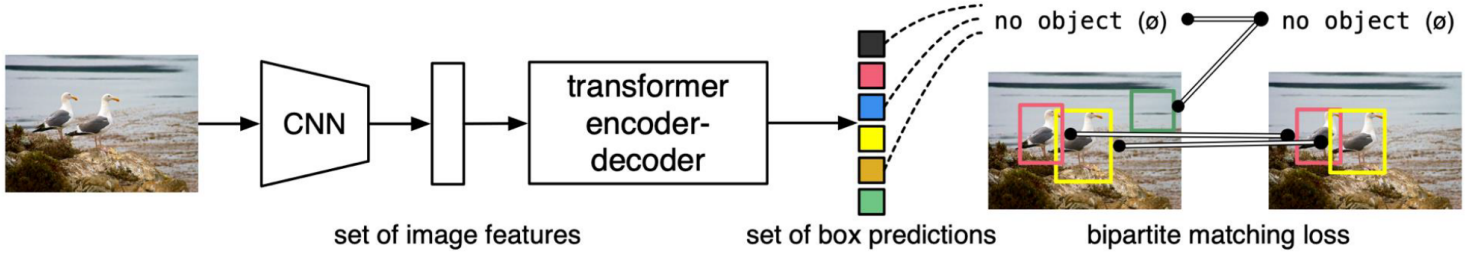
→ This is exactly what DETR does.

DETR(2020): Detection Transformer

Simple object detection pipeline: directly output a set of boxes from a Transformer

No anchors, no regression of box transforms

Match predicted boxes to GT boxes with bipartite matching; train to regress box coordinates



Key Takeaway Set prediction with Hungarian matching eliminates anchors and NMS → the first truly end-to-end object detector.

DETR Successors

Deformable DETR (2021)

✓ Training Speed
✓ Small Object

* DETR converges slowly (300-500 epochs)
* Deformable attention:
sparse sampling points
10× faster convergence

DINO-DETR (2022)

✓ Accuracy

* Contrastive denoising
Mixed query selection
Look forward twice
* First COCO SOTA in DETR variants

RT-DETR (2024)

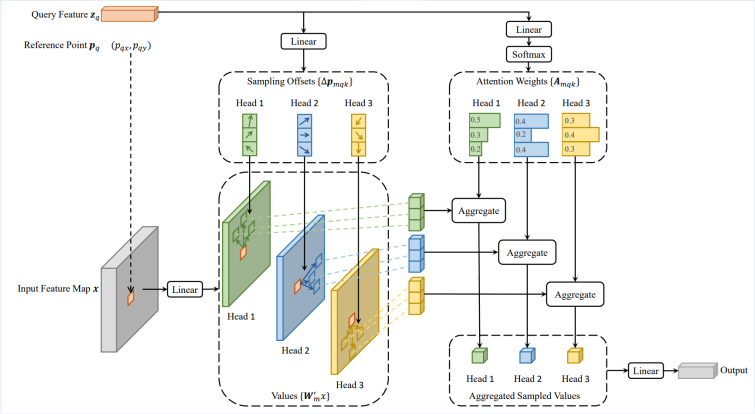
✓ Inference Speed
✓ Beats YOLO

* Hybrid encoder Beats YOLO at similar speed

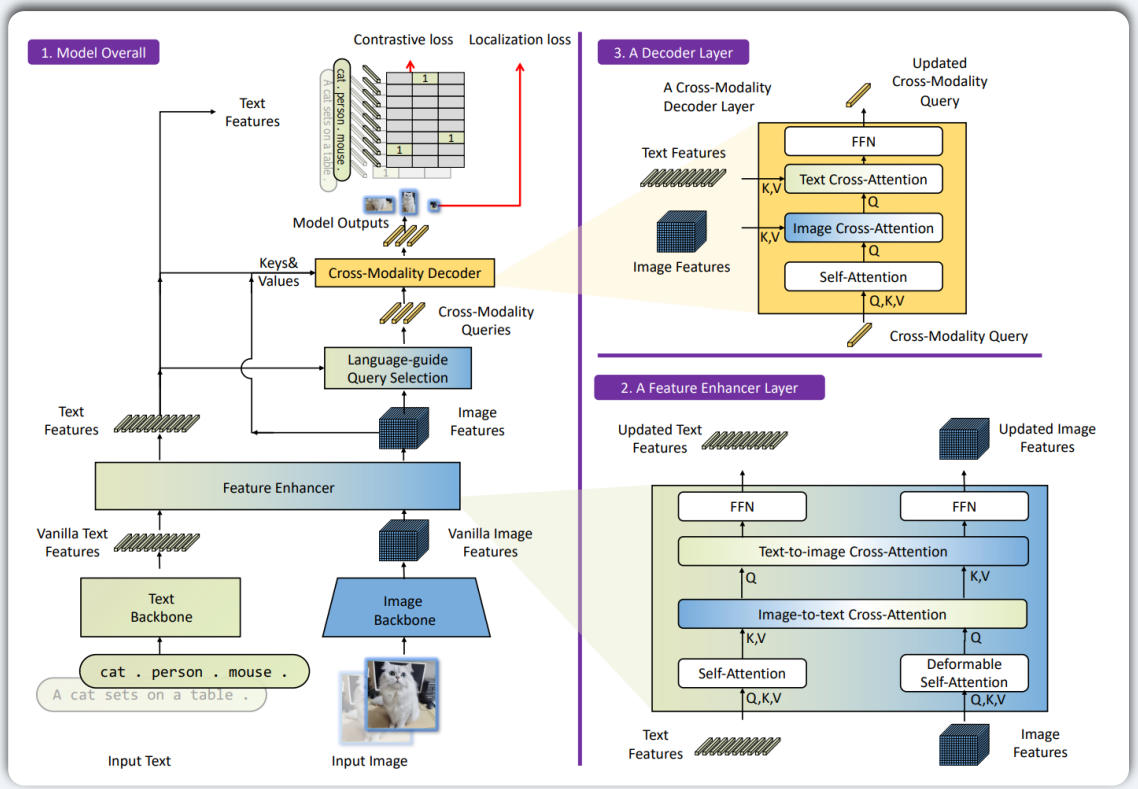
RF-DETR (2025)

✓ Accuracy + Speed
* DINOv2 (SSL) backbone
+ Deformable DETR decoder
* First real-time 60+ mAP on COCO
* SSL pretrain → detection

Deformable Attention



Grounding DINO: Open-Vocabulary Detection



Key Observations

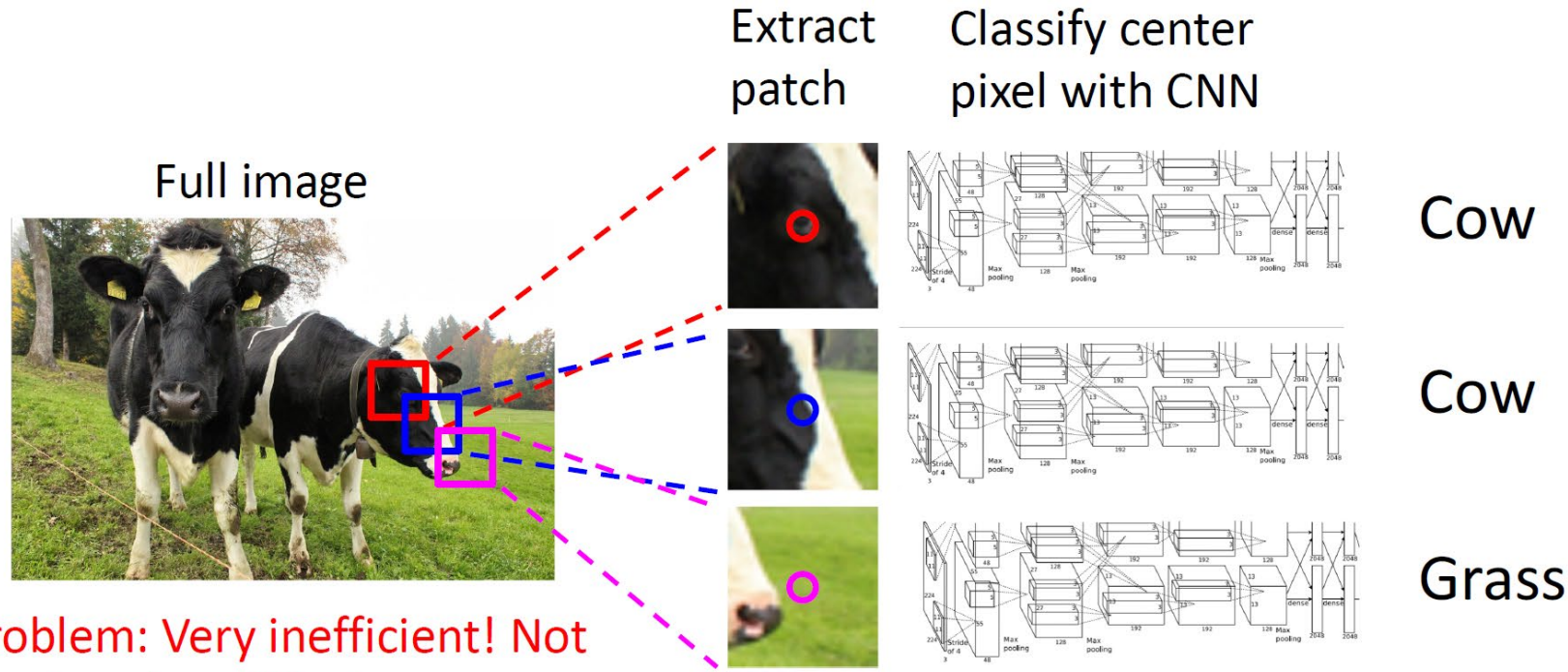
- Text prompt → detect any object (open vocabulary)
- Fuses DINO-DETR detector with BERT text encoder
- Cross-modality fusion via feature enhancer layers
- Demo: try at huggingface.co/IDEA-Research/grounding-dino

Part 3

Segmentation

From per-pixel classification to universal segmentation

Semantic Segmentation: Sliding Window



Problem: Very inefficient! Not reusing shared features between overlapping patches

Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

FCN: Fully Convolutional Networks

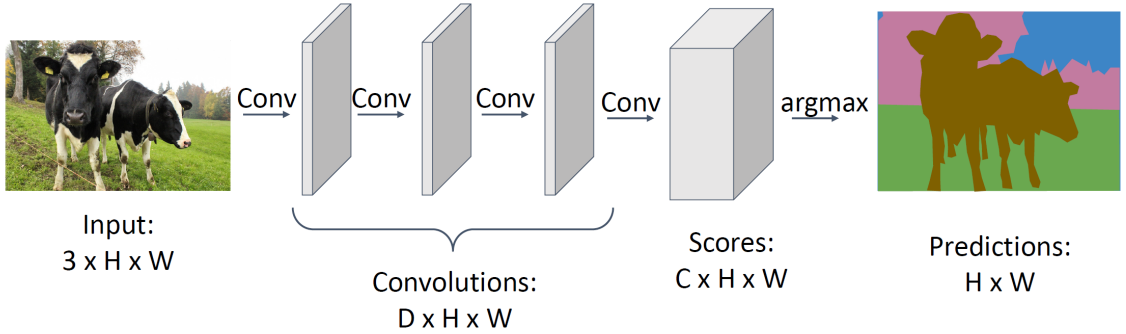
Long et al., CVPR 2015 — The foundation of modern segmentation

Convolutionalization

Replace FC layers with 1×1 conv layers

→ Output is a spatial map, not a single vector

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



Loss function: Per-Pixel cross-entropy

Long et al, "Fully convolutional networks for semantic segmentation", CVPR 2015

Why It Matters

Before FCN:

Sliding window → classify each patch

Extremely slow

After FCN:

Single forward pass → full segmentation map

Opened the door to:

- U-Net (2015)
- DeepLab (2015)
- PSPNet (2017)

FCN: Fully Convolutional Networks

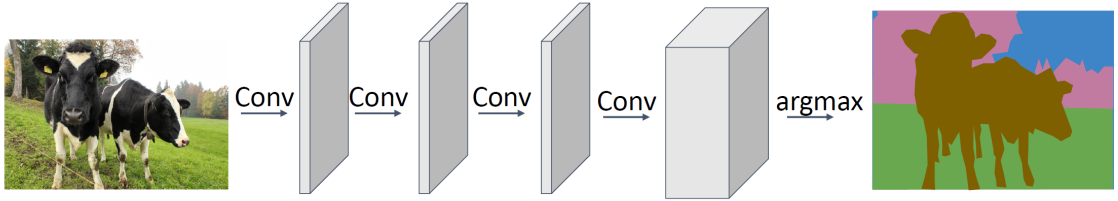
Long et al., CVPR 2015 — The foundation of modern segmentation

Convolutionalization

Replace FC layers with 1×1 conv layers

→ Output is a spatial map, not a single vector

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



Input:
3 x H x W

Problem #1: Effective receptive field size is linear in number of conv layers: With L 3x3 conv layers, receptive field is 1+2L

Problem #2: Convolution on high res images is expensive! Recall ResNet stem aggressively downsamples

Long et al, "Fully convolutional networks for semantic segmentation", CVPR 2015

Why It Matters

Before FCN:

Sliding window → classify each patch

Extremely slow

After FCN:

Single forward pass → full segmentation map

Opened the door to:

- U-Net (2015)
- DeepLab (2015)
- PSPNet (2017)

FCN: Fully Convolutional Networks

Long et al., CVPR 2015 — The foundation of modern segmentation

Convolutionalization

Replace FC layers with 1×1 conv layers

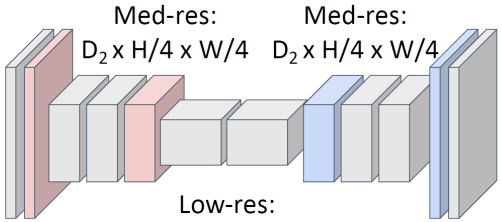
→ Output is a spatial map, not a single vector

Downsampling: Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling: linterpolation, transposed conv

Input:
3 x H x W



High-res:
D1 x H/2 x W/2

High-res:
D1 x H/2 x W/2

Predictions:
H x W

Loss function: Per-Pixel cross-entropy

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Why It Matters

Before FCN:

Sliding window → classify each patch

Extremely slow

After FCN:

Single forward pass → full segmentation map

Opened the door to:

- U-Net (2015)
- DeepLab (2015)
- PSPNet (2017)

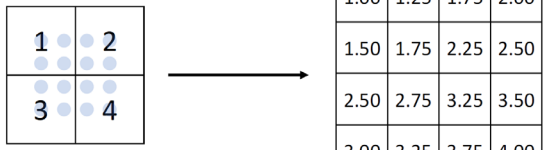
Upsampling & Skip Connections

Bilinear Upsampling

Fixed interpolation (no learnable params)

Fast, memory-efficient

Smooth but may lack detail



1.00	1.25	1.75	2.00
1.50	1.75	2.25	2.50
2.50	2.75	3.25	3.50
3.00	3.25	3.75	4.00

Input: C x 2 x 2

Output: C x 4 x 4

$$f_{x,y} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - i|) \max(0, 1 - |y - j|) \quad i \in \{\lfloor x \rfloor - 1, \ldots, \lceil x \rceil + 1\} \quad j \in \{\lfloor y \rfloor - 1, \ldots, \lceil y \rceil + 1\}$$

Use two closest neighbors in x and y to construct linear approximations

Transposed Convolution

Learnable upsampling

Also called "deconvolution" (misnomer)

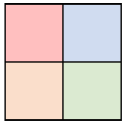
Can produce checkerboard artifacts

if kernel size not divisible by stride

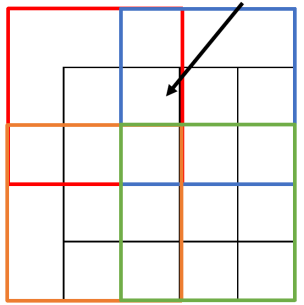
3 x 3 **convolution transpose**, stride 2

This gives 5x5 output – need to trim one pixel from top and left to give 4x4 output

Sum where output overlaps



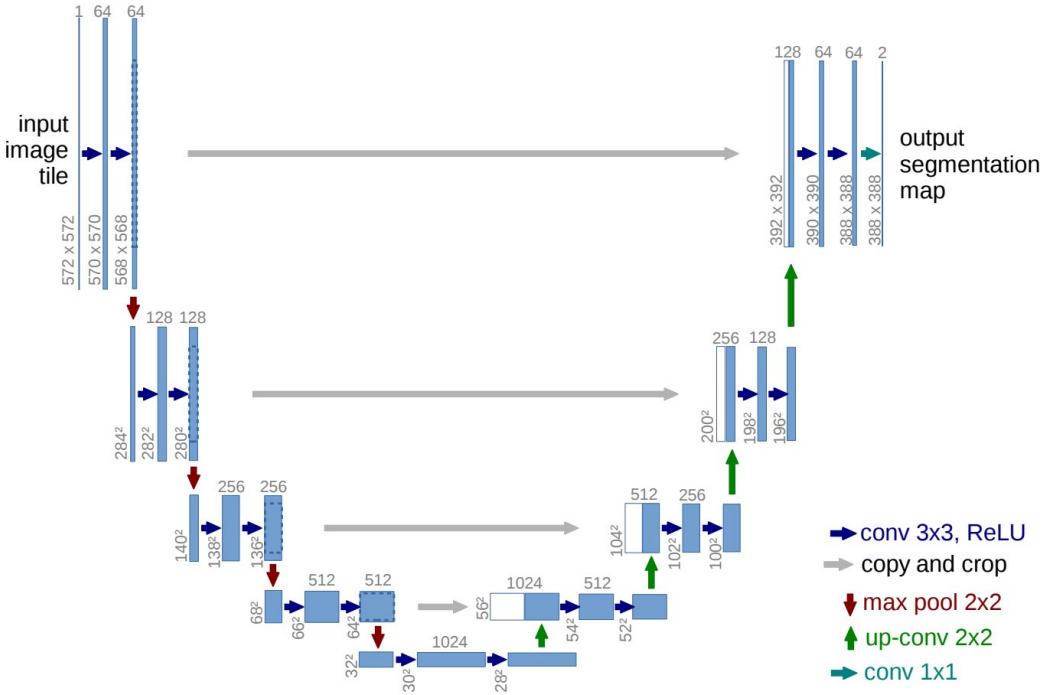
Weight filter by input value and copy to output



Input: 2 x 2

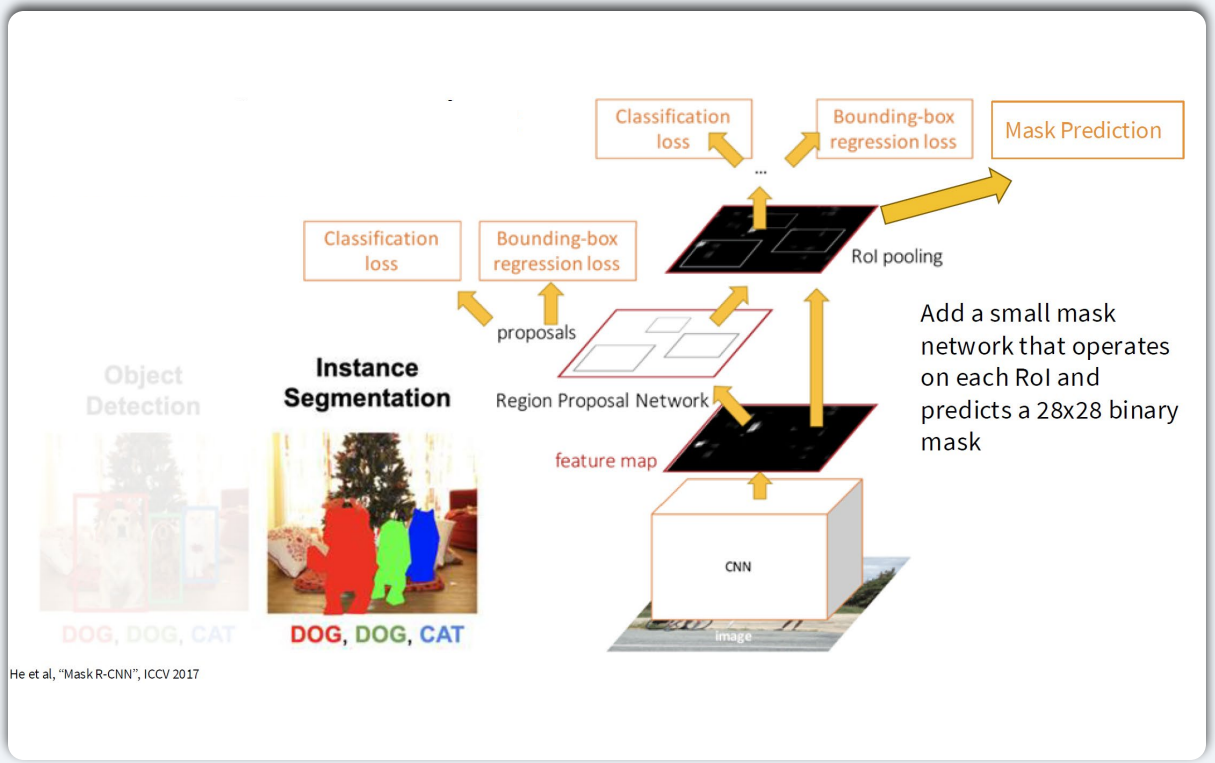
Output: 4 x 4

U-Net: Encoder–Decoder with Skip Connections



Key Takeaway U-Net's encoder-decoder + skip connection design became the backbone of diffusion models (→ Wk10).

Mask R-CNN: Instance Segmentation

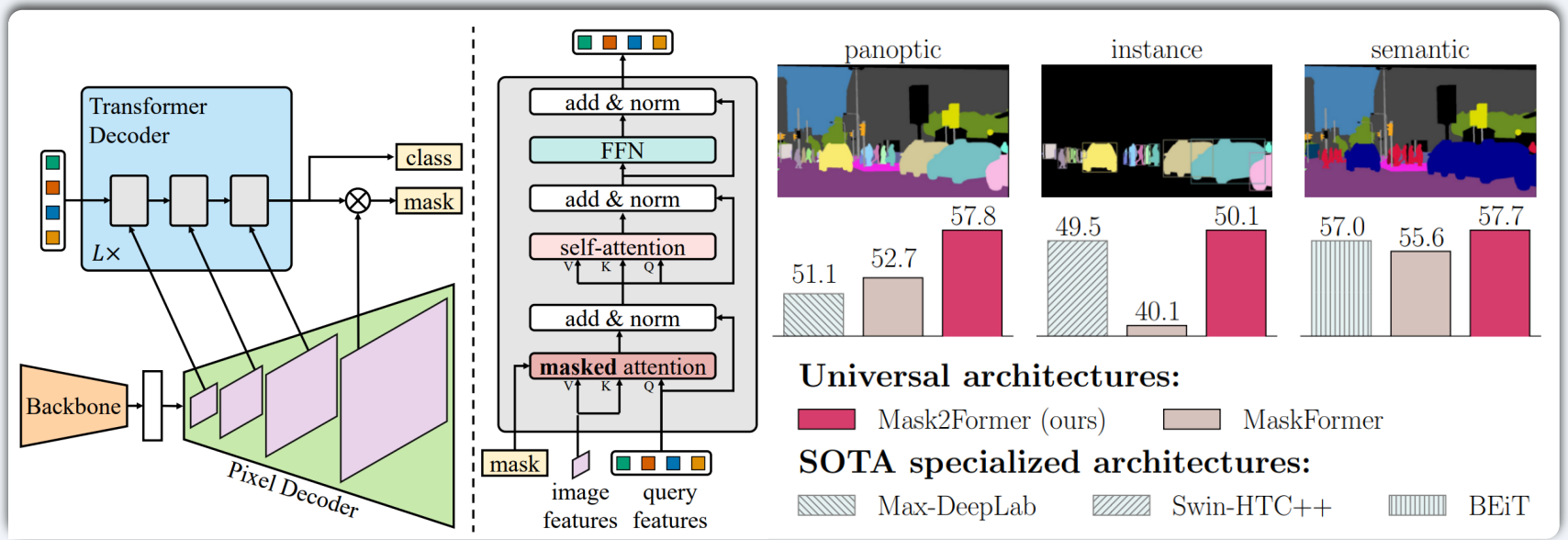


He et al, "Mask R-CNN", ICCV 2017

Key Observations

- Extends Faster R-CNN with a parallel mask prediction branch
- Small FCN applied to each RoI → binary mask per class
- RoI Align (not RoI Pool) is critical for mask quality
- Decouples mask and class prediction — key design insight

Mask2Former(2021): Universal Segmentation



Key Takeaway One architecture for all three segmentation tasks — extends DETR's object query idea to segmentation masks.

SAM: Segment Anything — Three Generations

SAM (2023)

Promptable Visual Segmentation

Prompt: points, boxes, masks
Output: one mask per prompt

ViT-H encoder + mask decoder
SA-1B: 1B masks, 11M images
Zero-shot to new domains

Kirillov et al., ICCV 2023

SAM 2 (2024)

Image + Video

Adds memory-based tracker
Propagate masks across frames

Streaming architecture
44 fps real-time

SA-V dataset: 51K videos

Ravi et al., 2024

SAM 3 (2025)

Promptable Concept Segmentation

New: text prompt + exemplar
"yellow school bus" → all instances

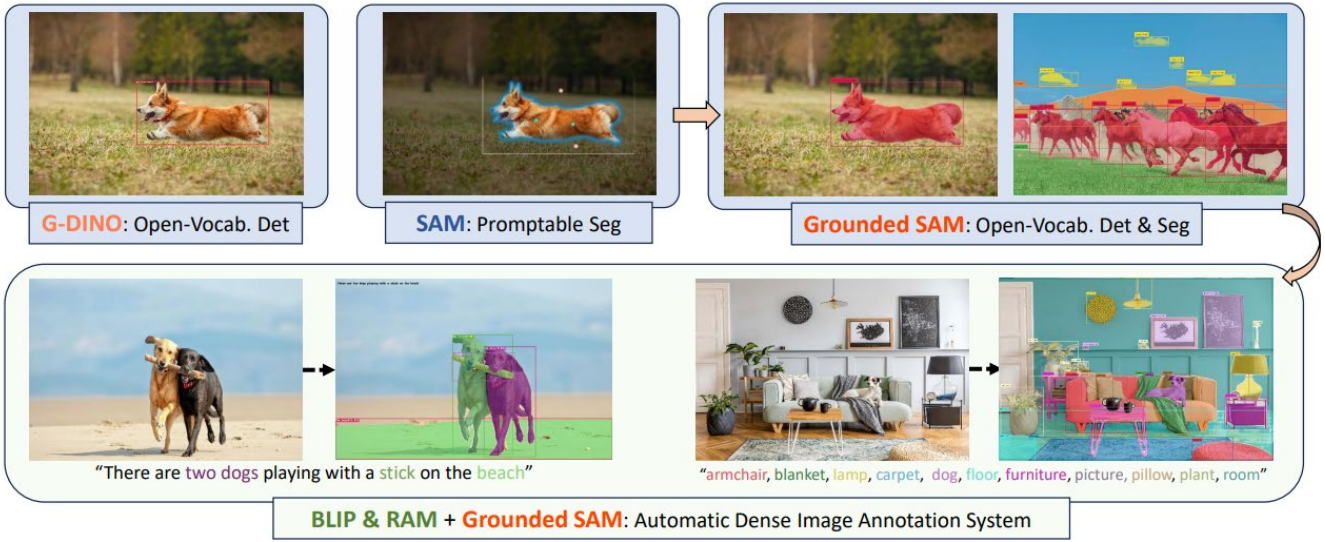
DETR-based detector + tracker
in single 848M param model

4M concepts, 2× prior SOTA

→ Detect + Segment + Track unified
Meta, Nov 2025

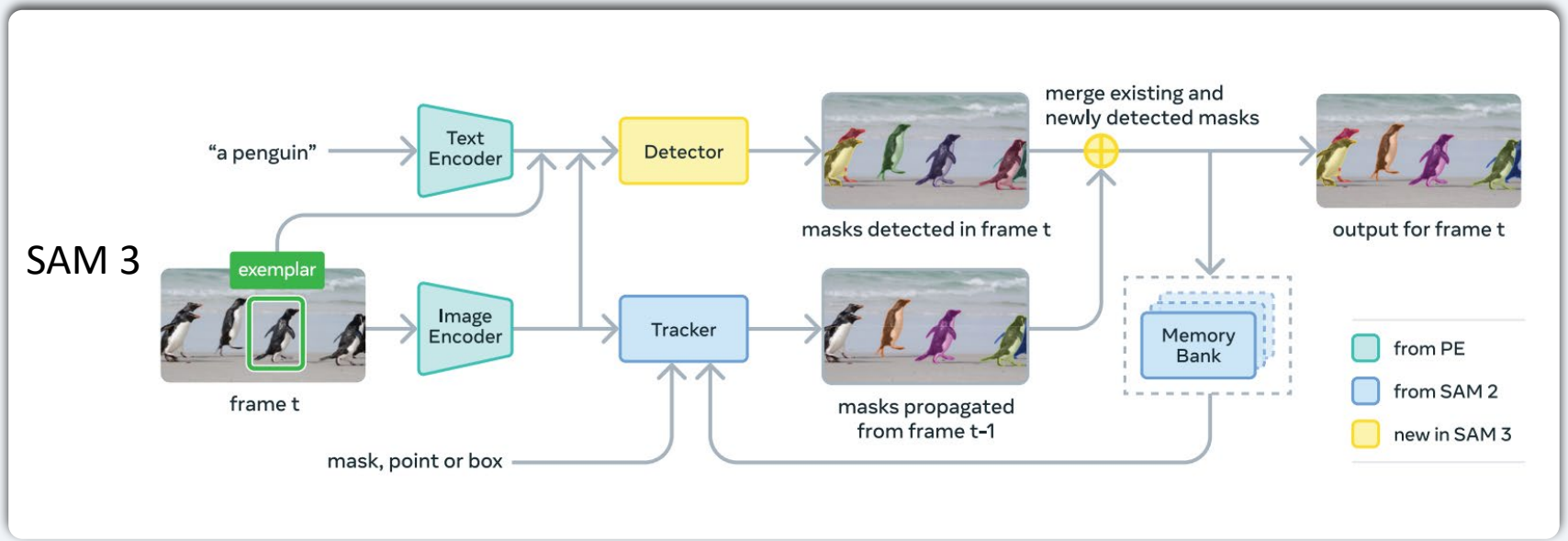
Grounded SAM (2024) → SAM 3 (2025): Pipeline to Unified Model

Grounded SAM



Key Takeaway Grounded SAM (2024): two-model pipeline for text-to-mask. SAM 3 (2025): one model does it all — the ultimate convergence of detection and segmentation.

Grounded SAM (2024) → SAM 3 (2025): Pipeline to Unified Model



Key Takeaway Grounded SAM (2024): two-model pipeline for text-to-mask. SAM 3 (2025): one model does it all — the ultimate convergence of detection and segmentation.

Detection & Segmentation Timeline

2014–15	2016–18	2020–22	2023–24	2025
Det	Det	Det	Det	Det
R-CNN → Faster 2014–15 *DL + end-to-end*	**FPN, RetinaNet** 2017 *multi-scale, Focal Loss*	**DETR** 2020 *Transformer, no NMS*	**Grounding DINO** 2023 *open-vocabulary*	**RF-DETR** 2025 *DINOv2, 60+ mAP*
	YOLO	YOLO	YOLO	YOLO
	v1 → v3 2016–18 *real-time, multi-scale*	**v5 → v8** 2020–23 *ecosystem, anchor-free*	**v11** 2024 *efficiency*	**v12** 2025 *attention-centric*
Seg	Seg	Seg	Seg	Seg
FCN, U-Net 2015 *dense prediction*	**Mask R-CNN** 2017 *instance masks*	**Mask2Former** 2022 *unified 3 tasks*	**SAM** 2023 *promptable, zero-shot*	**SAM 3** 2025 *detect + segment + track*

2025: Boundaries dissolve — YOLOv12 adopts attention, RF-DETR uses SSL backbone, SAM 3 unifies detection + segmentation + tracking

Summary & Resources

What We Covered

Vision task spectrum: Classification → Panoptic
IoU, mAP evaluation metrics

Detection: R-CNN → Faster R-CNN → FPN
YOLO (v1→v12) + RetinaNet (Focal Loss)
DETR → RF-DETR (DINOv2 backbone)
Grounding DINO: open-vocabulary detection

Segmentation: FCN → U-Net → Mask R-CNN
Mask2Former: unified segmentation
SAM → SAM 3: detect + segment + track

Further Reading & Self-Study

Michigan EECS 498 L15-16: Detection & Segmentation
web.eecs.umich.edu/~justincj/teaching/eecs498

Stanford CS231n L9: Object Detection & Segmentation
cs231n.stanford.edu

Lilian Weng: Object Detection (Parts 1-4)
lilianweng.github.io

YOLO Guide: Ultralytics Documentation
docs.ultralytics.com

Next Week Week 7: Self-Supervised Learning — How models learn without labels (SimCLR, DINO, MAE)

Acknowledgments

Some slides and figures in this course are adapted from or inspired by the following open resources.

Stanford CS231n: Deep Learning for Computer Vision (Spring 2025)

Fei-Fei Li, Ehsan Adeli, et al. | cs231n.stanford.edu

UMich EECS 498/598: Deep Learning for Computer Vision (Winter 2022)

Justin Johnson | web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

IP-Paris Master Year 2 Data Science

Olivier Grisel, Charles Ollion | https://github.com/m2dsupsdlclass/lectures-labs

MIT 6.7960: Deep Learning (Fall 2024)

Phillip Isola, Sara Beery, Jeremy Bernstein | ocw.mit.edu/courses/6-7960-deep-learning-fall-2024/

CMU 16-824: Visual Learning and Recognition (Fall 2025)

Jun-Yan Zhu | visual-learning.cs.cmu.edu

Key papers: R-CNN, Faster R-CNN, FPN, YOLO, DETR, FCN, U-Net, Mask R-CNN, Mask2Former, SAM, Grounding DINO, ...