



Object Detection

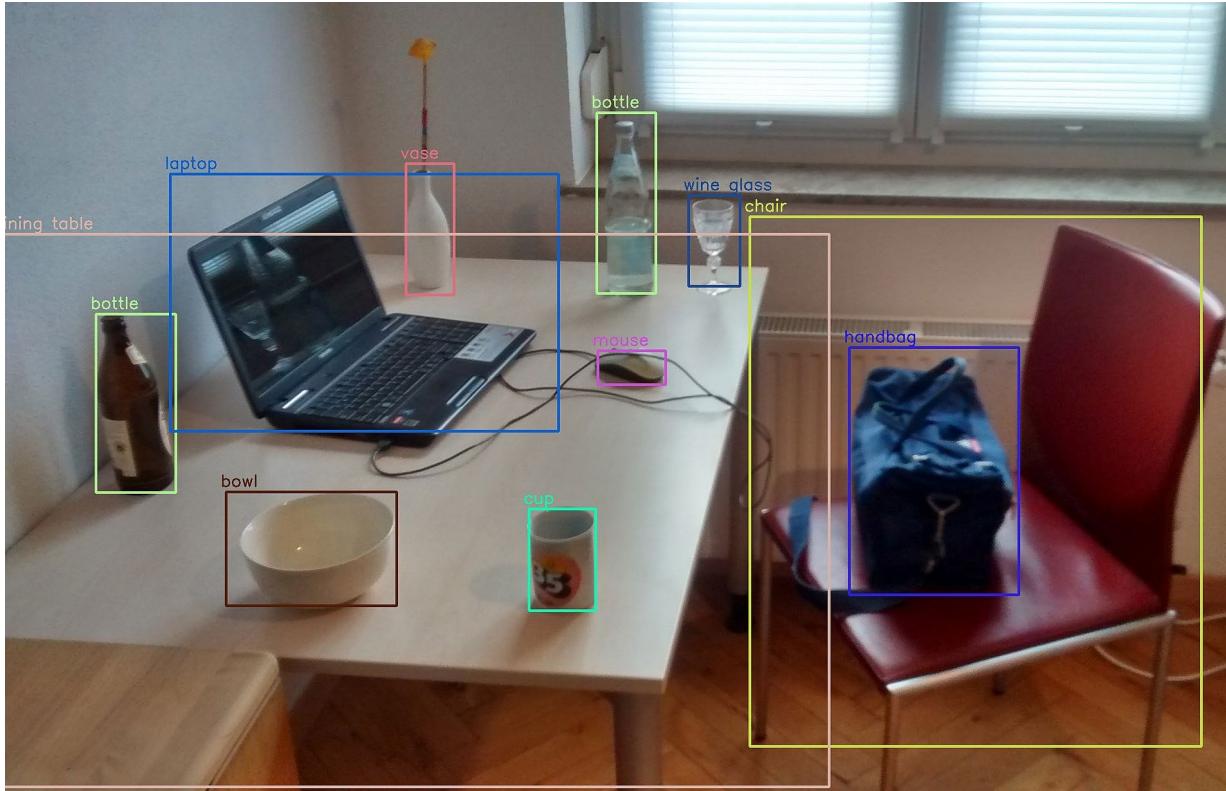
CS 6384 Computer Vision

Professor Yu Xiang

The University of Texas at Dallas

Object Detection

- Localize objects in images and classify them



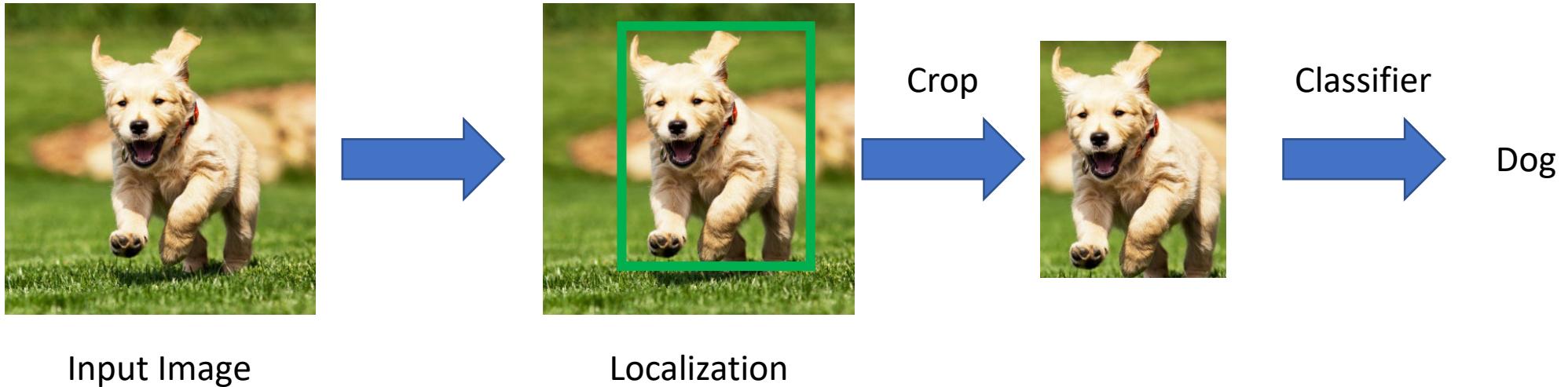
Wikipedia

Why using bounding boxes?

- Easy to store
 - (x, y, w, h) : box center with width, height
 - (x_1, y_1, x_2, y_2) : top left corner and bottom right corner
- Easy for image processing
 - Crop a region

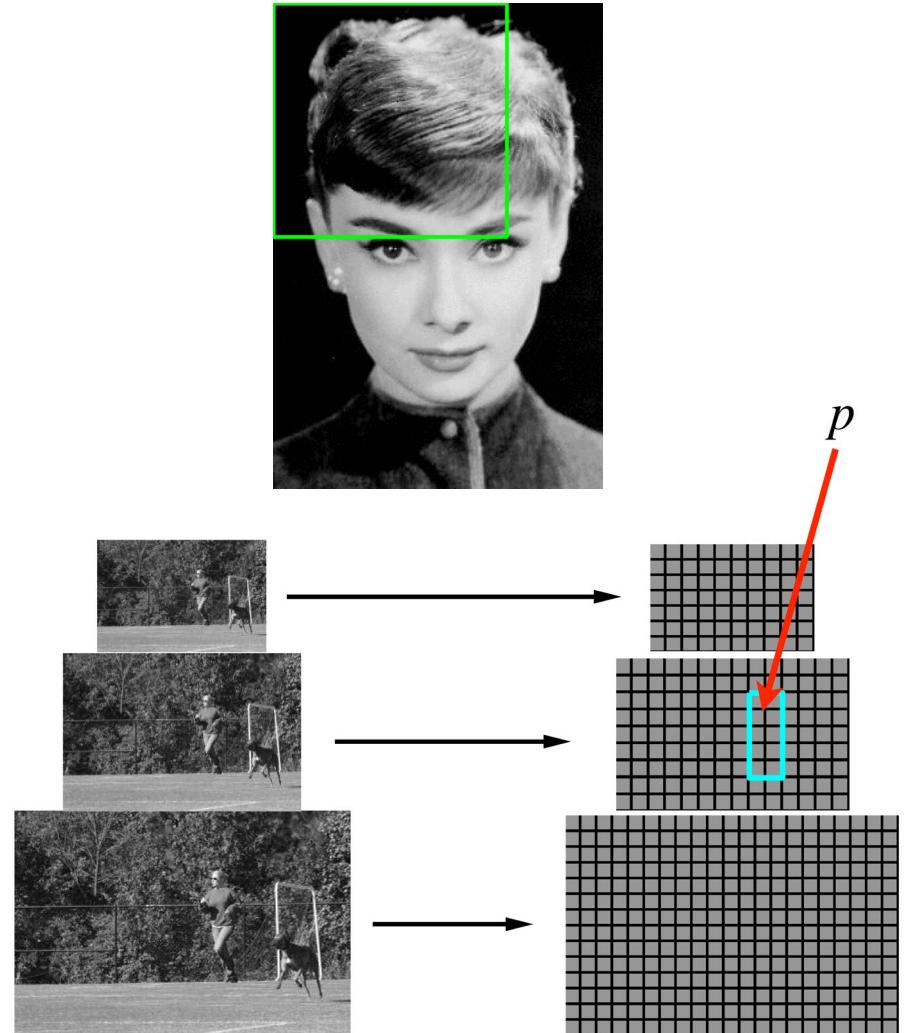
Object Detection

- Localization + Classification



Localization: Sliding Window

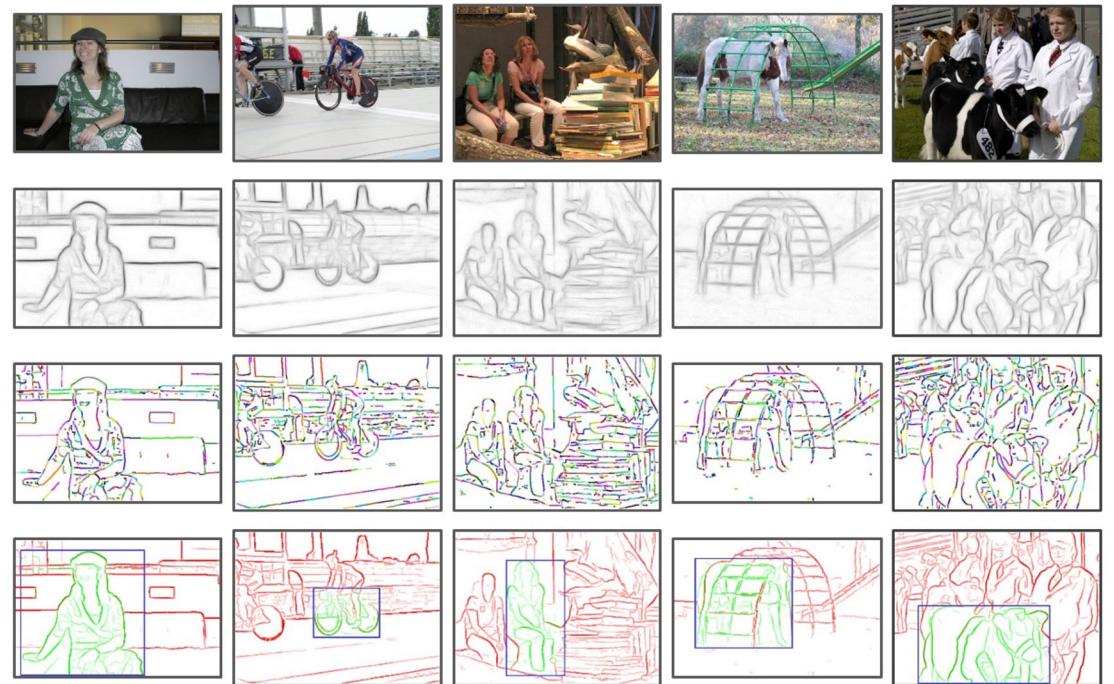
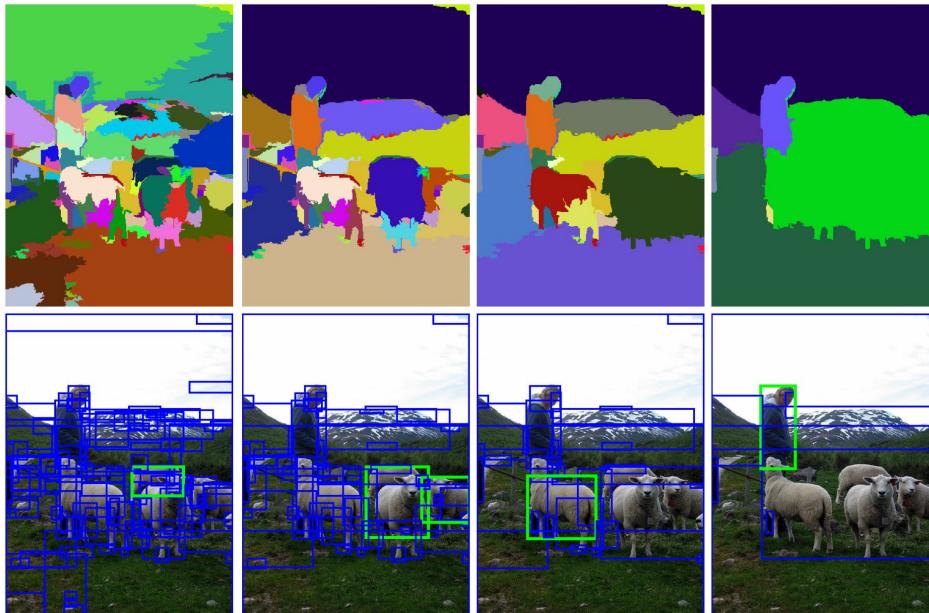
- Select a window with a fixed size
- Scan the input image with the window (bounding box)
- How to deal with different object scales and aspect ratios?
 - Use boxes with different aspect ratios
 - Image pyramid



<https://cvexplained.wordpress.com/tag/sliding-windows/>

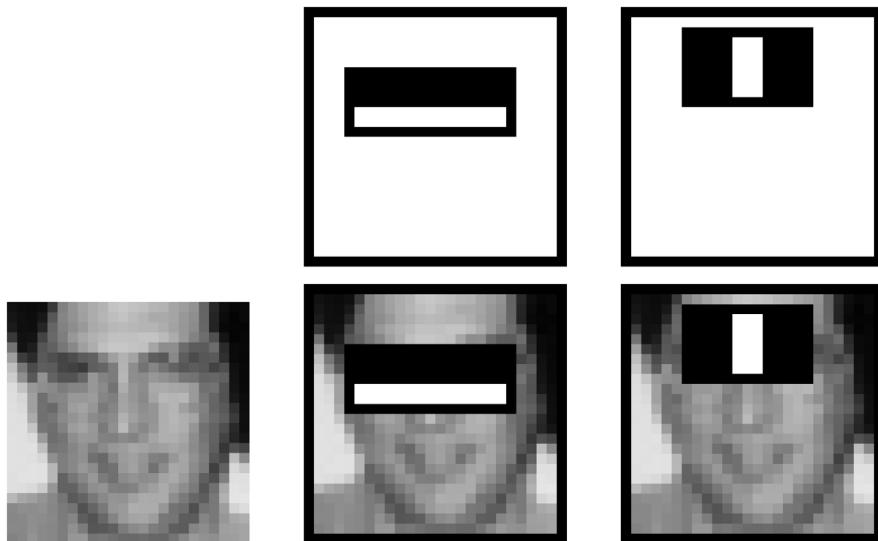
Localization: Region Proposal

- Leverage methods that can generate regions with high likelihood of containing objects
 - E.g., bottom-up segmentation methods, using edges

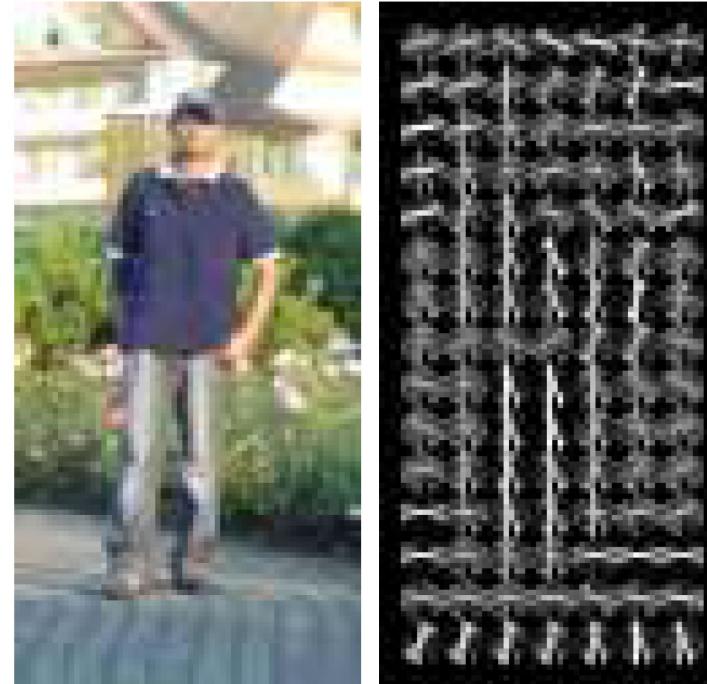


Classification: Features

- Traditional methods: Hand-crafted features
- Deep learning methods: learned features in the network



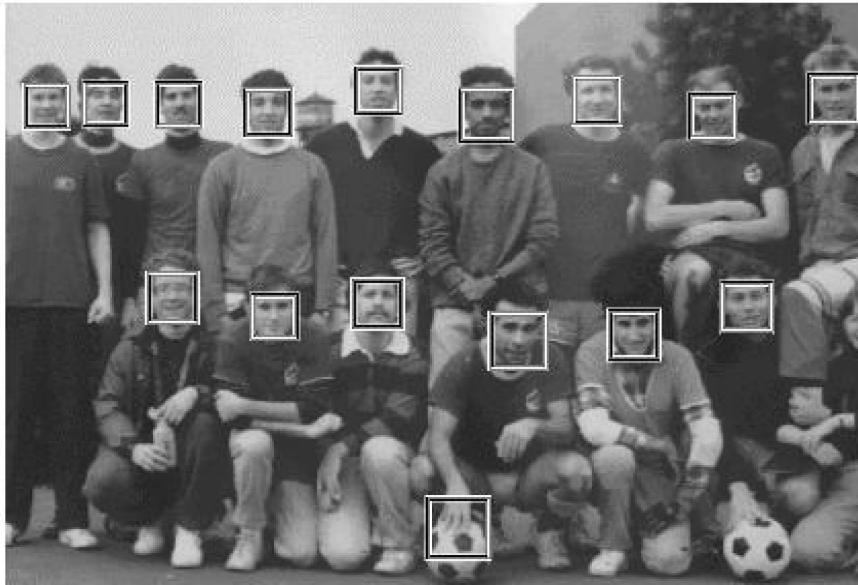
Viola and Jones: rectangle features



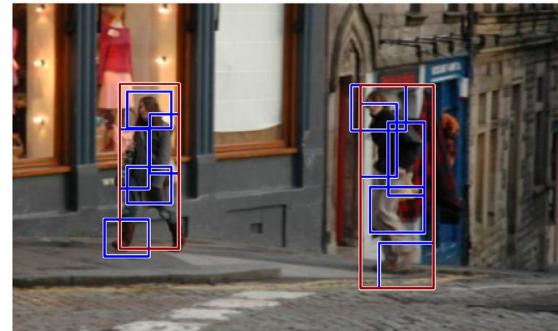
Dadal & Triggs: Histograms of Oriented Gradients

Classification: Classifiers

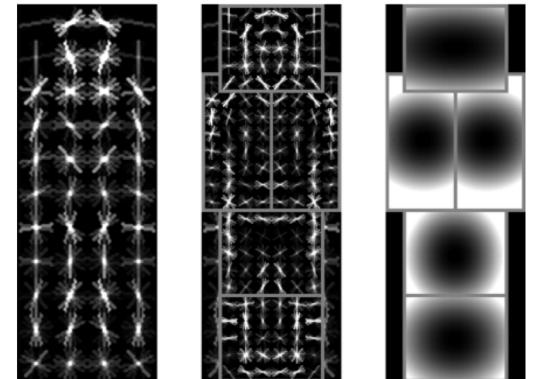
- Traditional methods
 - AdaBoost
 - Support vector machines (SVMs)
- Deep learning methods
 - Neural networks



Viola and Jones: AdaBoost
Robust Real-time Object Detection. IJCV, 2001.



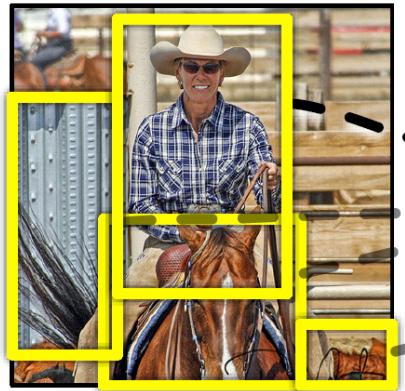
Felzenszwalb et al: SVM
Object detection with discriminatively trained part-based models . TPAMI, 2009.



R-CNN



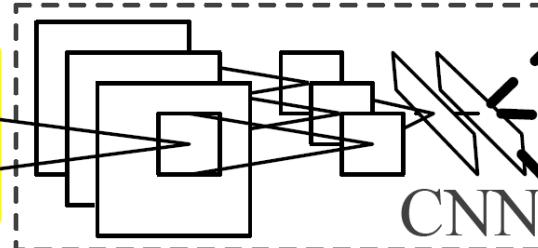
1. Input
image



2. Extract region
proposals (~2k)

Selective Search

warped region



3. Compute
CNN features

CNN

aeroplane? no.

person? yes.

tvmonitor? no.

4. Classify
regions

SVM

Rich feature hierarchies for accurate object detection and semantic segmentation. Girshick et al., CVPR, 2014

R-CNN

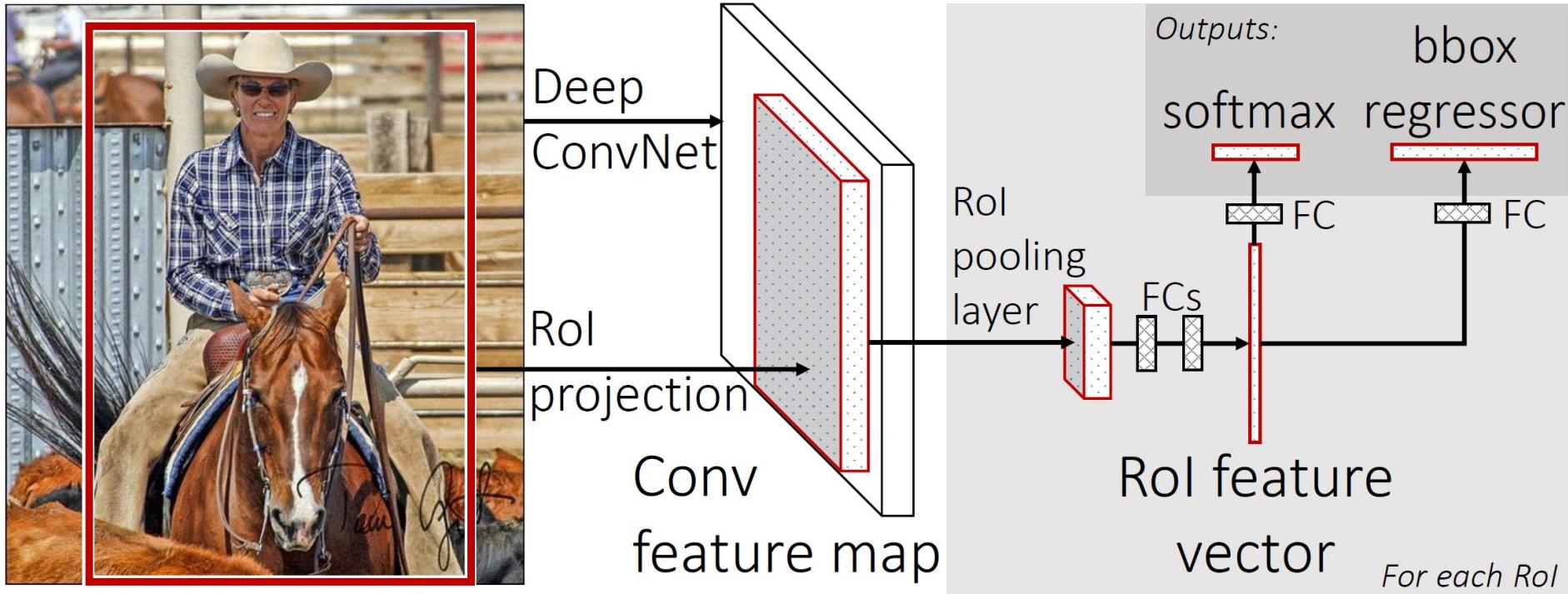
VOC 2007 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
R-CNN pool ₅	51.8	60.2	36.4	27.8	23.2	52.8	60.6	49.2	18.3	47.8	44.3	40.8	56.6	58.7	42.4	23.4	46.1	36.7	51.3	55.7	44.2
R-CNN fc ₆	59.3	61.8	43.1	34.0	25.1	53.1	60.6	52.8	21.7	47.8	42.7	47.8	52.5	58.5	44.6	25.6	48.3	34.0	53.1	58.0	46.2
R-CNN fc ₇	57.6	57.9	38.5	31.8	23.7	51.2	58.9	51.4	20.0	50.5	40.9	46.0	51.6	55.9	43.3	23.3	48.1	35.3	51.0	57.4	44.7
R-CNN FT pool ₅	58.2	63.3	37.9	27.6	26.1	54.1	66.9	51.4	26.7	55.5	43.4	43.1	57.7	59.0	45.8	28.1	50.8	40.6	53.1	56.4	47.3
R-CNN FT fc ₆	63.5	66.0	47.9	37.7	29.9	62.5	70.2	60.2	32.0	57.9	47.0	53.5	60.1	64.2	52.2	31.3	55.0	50.0	57.7	63.0	53.1
R-CNN FT fc ₇	64.2	69.7	50.0	41.9	32.0	62.6	71.0	60.7	32.7	58.5	46.5	56.1	60.6	66.8	54.2	31.5	52.8	48.9	57.9	64.7	54.2
R-CNN FT fc ₇ BB	68.1	72.8	56.8	43.0	36.8	66.3	74.2	67.6	34.4	63.5	54.5	61.2	69.1	68.6	58.7	33.4	62.9	51.1	62.5	64.8	58.5
DPM v5 [20]	33.2	60.3	10.2	16.1	27.3	54.3	58.2	23.0	20.0	24.1	26.7	12.7	58.1	48.2	43.2	12.0	21.1	36.1	46.0	43.5	33.7
DPM ST [28]	23.8	58.2	10.5	8.5	27.1	50.4	52.0	7.3	19.2	22.8	18.1	8.0	55.9	44.8	32.4	13.3	15.9	22.8	46.2	44.9	29.1
DPM HSC [31]	32.2	58.3	11.5	16.3	30.6	49.9	54.8	23.5	21.5	27.7	34.0	13.7	58.1	51.6	39.9	12.4	23.5	34.4	47.4	45.2	34.3

BB: bounding box regression

Features from AlexNet

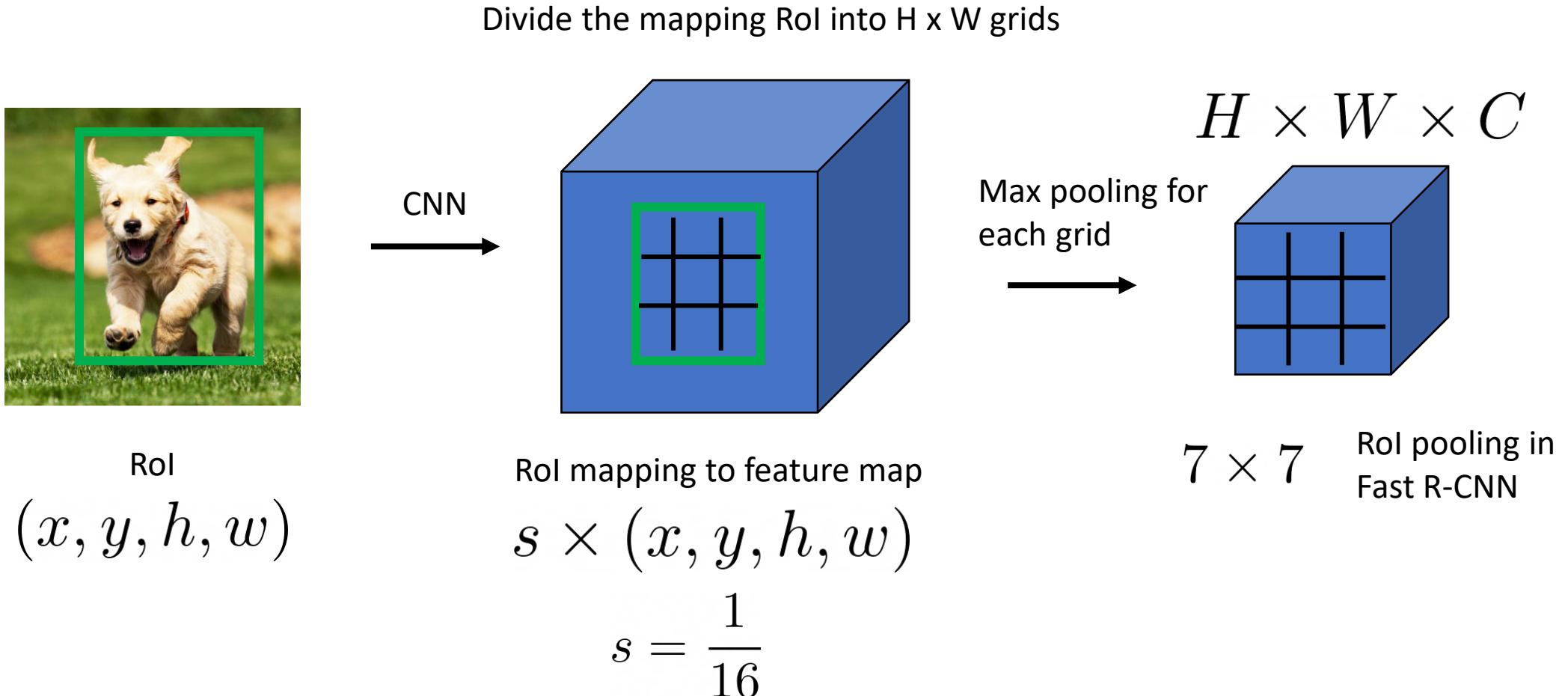
Rich feature hierarchies for accurate object detection and semantic segmentation. Girshick et al., CVPR, 2014

Fast R-CNN



Fast R-CNN. Girshick, ICCV, 2015

RoI Pooling



Bounding Box Regression

- Predict bounding box regression offset for K object classes

$$t^k = (t_x^k, t_y^k, t_w^k, t_h^k)$$

$$t_x = (G_x - P_x)/P_w$$

$$t_y = (G_y - P_y)/P_h$$

$$t_w = \log(G_w/P_w)$$

$$t_h = \log(G_h/P_h).$$

$$\hat{G}_x = P_w d_x(P) + P_x$$

$$\hat{G}_y = P_h d_y(P) + P_y$$

$$\hat{G}_w = P_w \exp(d_w(P))$$

$$\hat{G}_h = P_h \exp(d_h(P)).$$

G: ground truth, P: input RoI

Fast R-CNN

- Loss function

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1] L_{\text{loc}}(t^u, v)$$

Softmax classification probabilities

$$p = (p_0, \dots, p_K)$$

True class label

Bounding box regress target

Bounding box regress prediction

$$t^u = (t_x^u, t_y^u, t_w^u, t_h^u)$$

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x}, \text{y}, \text{w}, \text{h}\}} \text{smooth}_{L_1}(t_i^u - v_i)$$

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

Fast R-CNN

	Fast R-CNN			R-CNN			SPPnet
	S	M	L	S	M	L	$\dagger L$
train time (h)	1.2	2.0	9.5	22	28	84	25
train speedup	18.3×	14.0×	8.8×	1×	1×	1×	3.4×
test rate (s/im)	0.10	0.15	0.32	9.8	12.1	47.0	2.3
▷ with SVD	0.06	0.08	0.22	-	-	-	-
test speedup	98×	80×	146×	1×	1×	1×	20×
▷ with SVD	169×	150×	213×	-	-	-	-
VOC07 mAP	57.1	59.2	66.9	58.5	60.2	66.0	63.1
▷ with SVD	56.5	58.7	66.6	-	-	-	-

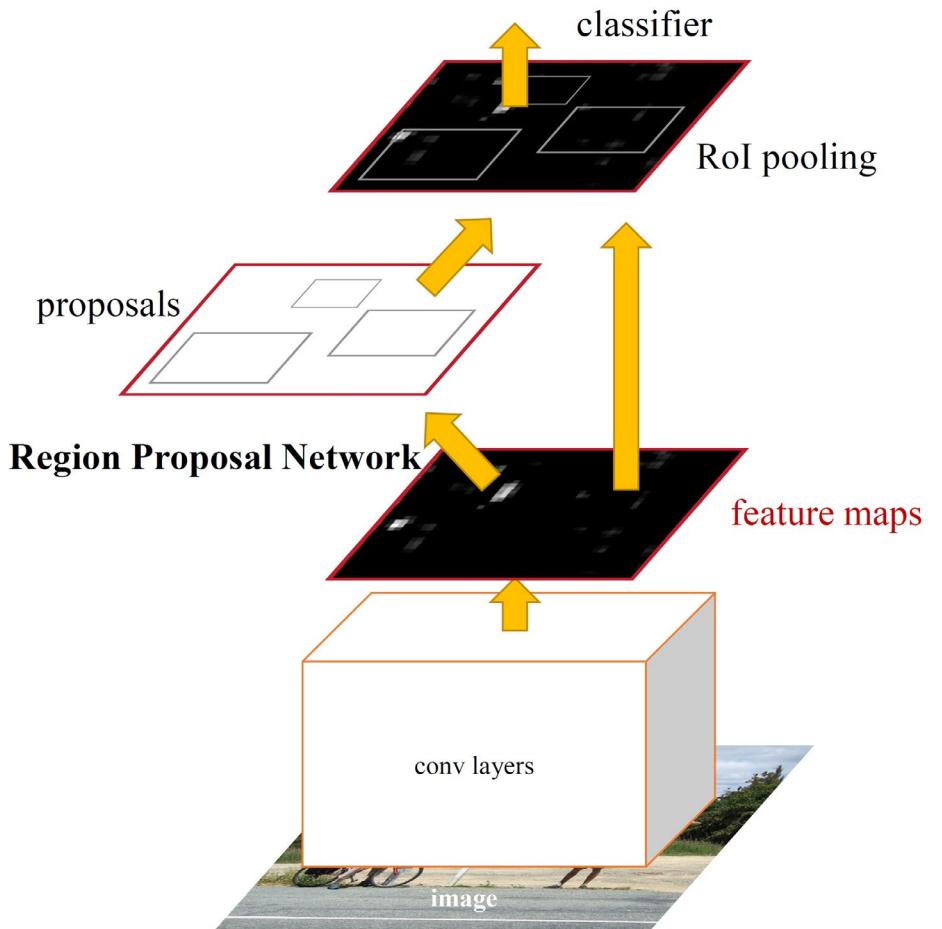
S: AlexNet, M: VGG, L:
deep VGG
SVD for FCs layers

$$W \approx U \Sigma_t V^T$$

Fast R-CNN. Girshick, ICCV, 2015

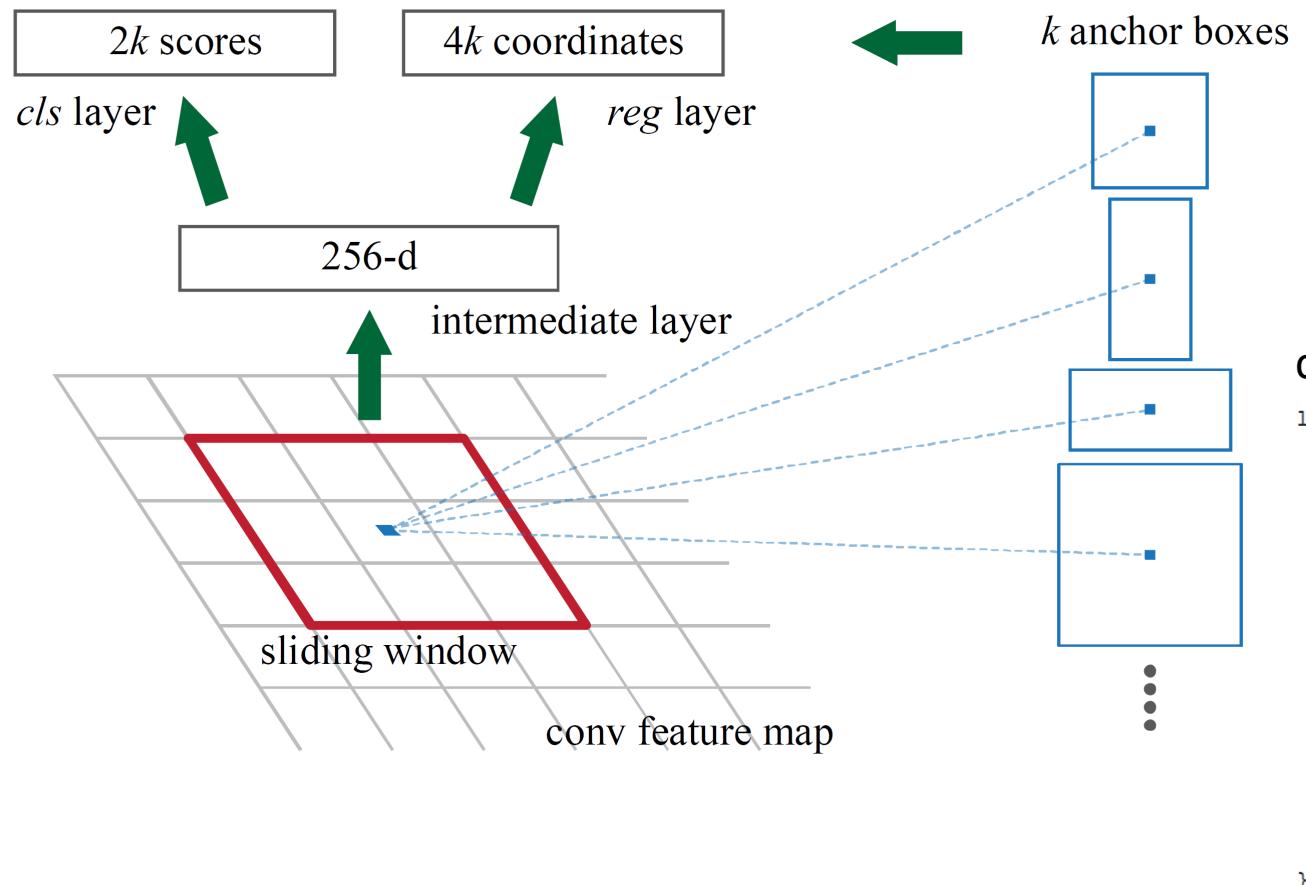
Faster R-CNN

- A single network for object detection
 - Region proposal network
 - Classification network



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

Region Proposal Network



3x3 conv layer to 256-d

```
layer {
    name: "rpn_conv/3x3"
    type: "Convolution"
    bottom: "conv5"
    top: "rpn/output"
    param { lr_mult: 1.0 }
    param { lr_mult: 2.0 }
    convolution_param {
        num_output: 256
        kernel_size: 3 pad: 1 stride: 1
        weight_filler { type: "gaussian" std: 0.01 }
        bias_filler { type: "constant" value: 0 }
    }
}
```

Bounding box regression

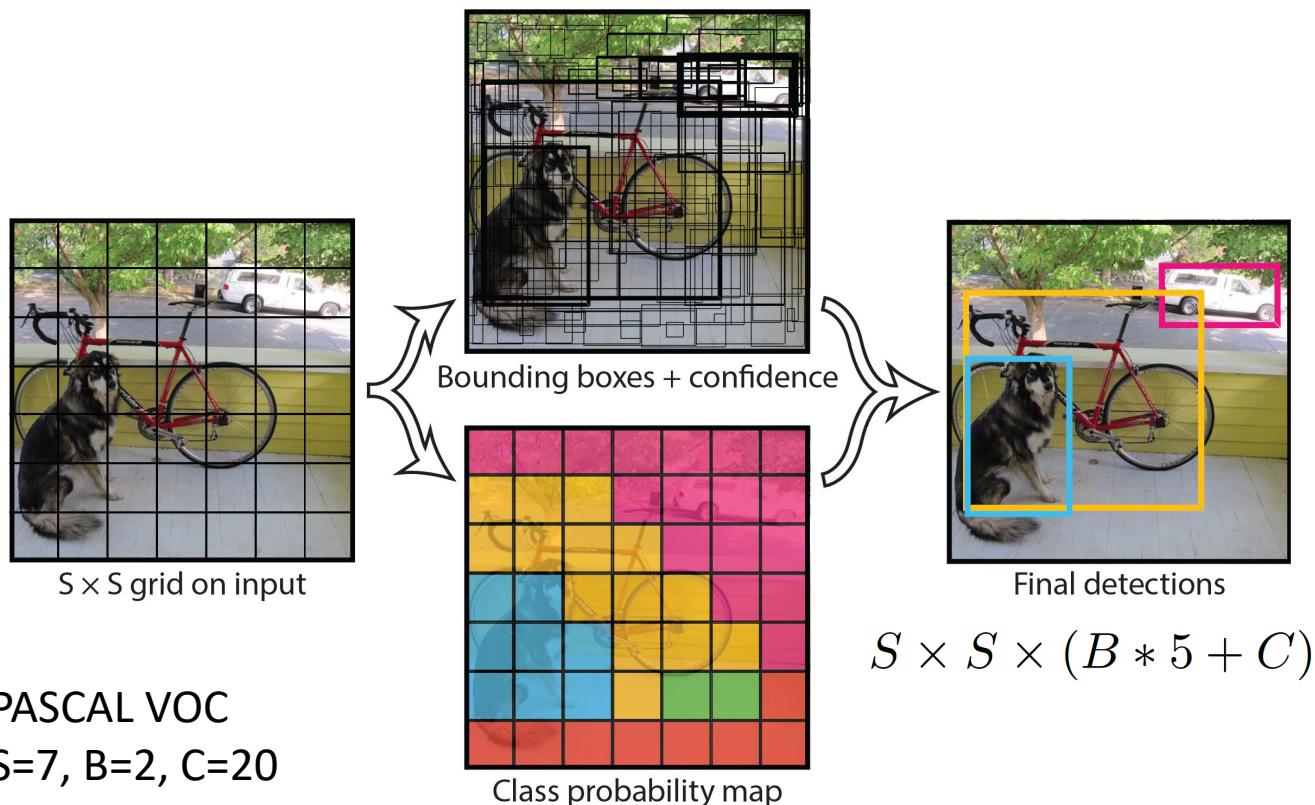
```
layer {
    name: "rpn_bbox_pred"
    type: "Convolution"
    bottom: "rpn/output"
    top: "rpn_bbox_pred"
    param { lr_mult: 1.0 }
    param { lr_mult: 2.0 }
    convolution_param {
        num_output: 36 # 4 * 9(anchors)
        kernel_size: 1 pad: 0 stride: 1
        weight_filler { type: "gaussian" std: 0.01 }
        bias_filler { type: "constant" value: 0 }
    }
}
```

Two stage vs One stage

- Two stage detection methods
 - Stage 1: generate region proposals
 - Stage 2: classify region proposals and refine their locations
 - E.g., R-CNN, Fast R-CNN, Faster R-CNN
- One stage detection methods
 - An end-to-end network for object detection
 - E.g., YOLO

YOLO

- Regress to bounding box locations and class probabilities



- Each grid handles objects with centers (x, y) in it
- Each grid predicts B bounding boxes
- Each bounding box predicts (x, y, w, h) and confidence (IoU of box and ground truth box)

$$\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

- Each grid also predicts C class probabilities

$$\Pr(\text{Class}_i | \text{Object})$$

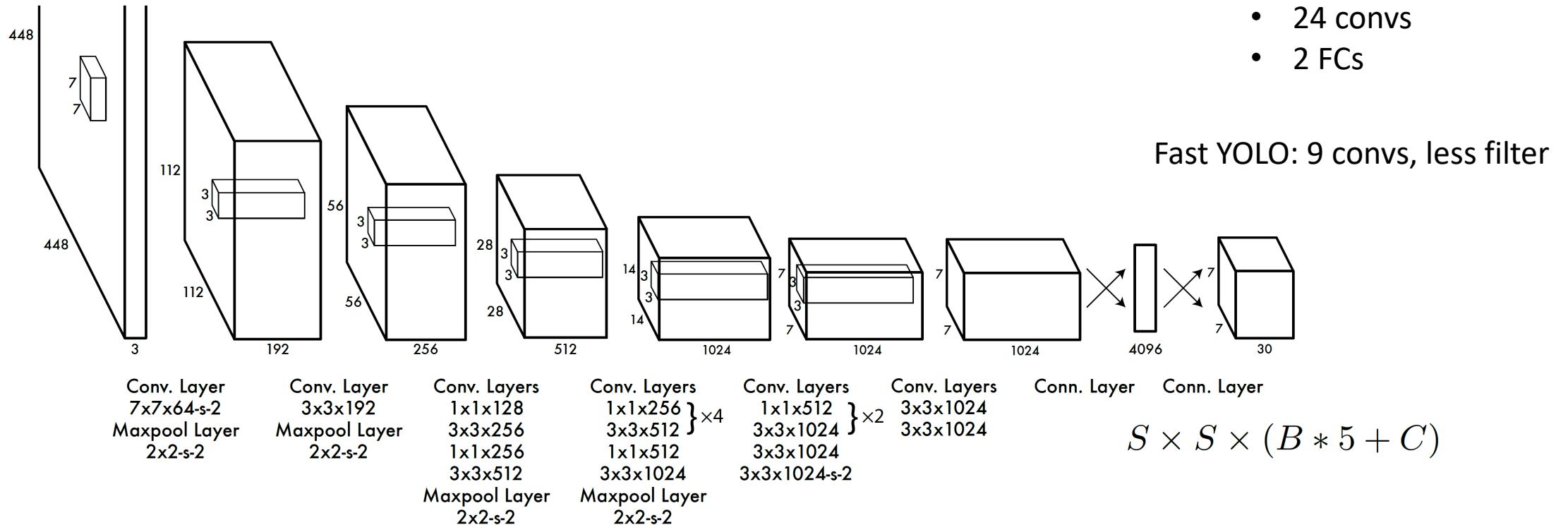
- In testing

$$\Pr(\text{Class}_i | \text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

You Only Look Once: Unified, Real-Time Object Detection. Redmon et al., CVPR, 2016

YOLO

- Regress to bounding box locations and class probabilities



You Only Look Once: Unified, Real-Time Object Detection. Redmon et al., CVPR, 2016

YOLO

- Training loss function

$\mathbb{1}_{ij}^{\text{obj}}$ jth bounding box from cell i
“responsible” for the prediction

highest current IOU with the ground truth

$\mathbb{1}_i^{\text{obj}}$ Object in cell i

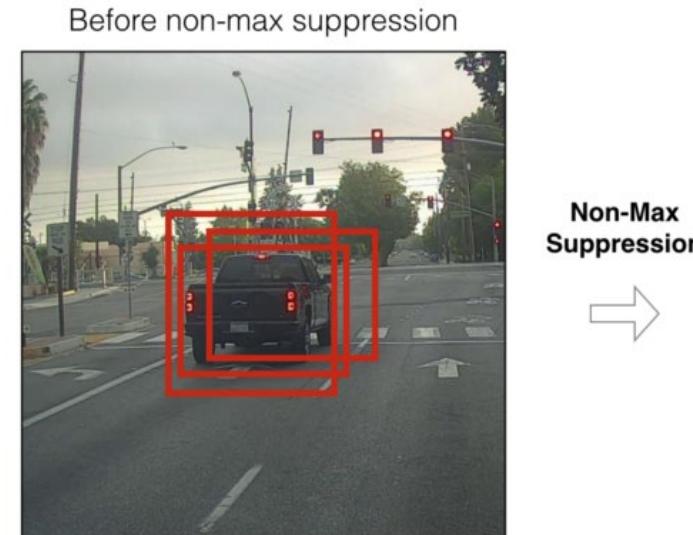
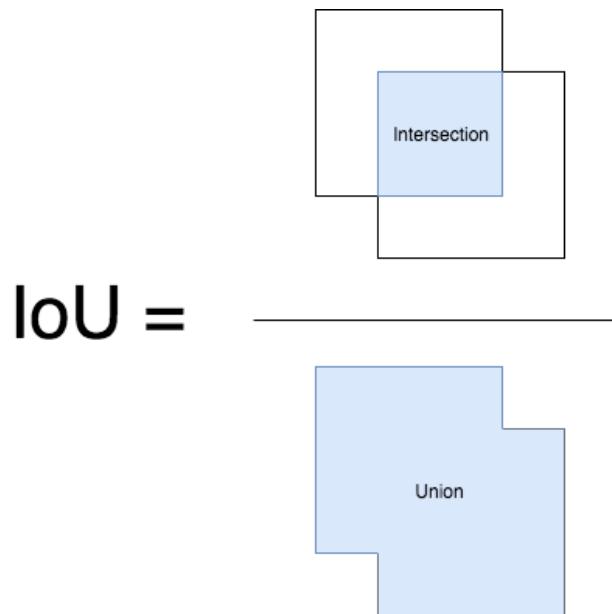
$$\lambda_{\text{coord}} = 5 \quad \lambda_{\text{noobj}} = .5$$

$$\begin{aligned} & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\ & + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\ & + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\ & + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\ & + \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \end{aligned}$$

You Only Look Once: Unified, Real-Time Object Detection. Redmon et al., CVPR, 2016

Non-maximum Suppression

- Keep the box with the highest confidence/score
- Compute IoU between this box and other boxes
- Suppress boxes with $\text{IoU} > \text{threshold}$



<https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c>

YOLO

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	155
YOLO	2007+2012	63.4	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

You Only Look Once: Unified, Real-Time Object Detection. Redmon et al., CVPR, 2016

YOLOv2 and YOLOv3

- YOLOv2
 - Batch normalization (normalization of the layers' inputs by re-centering and re-scaling)
 - High resolution classifier 416x416
 - Convolutional with anchor boxes (remove FC layers)
 - Dimension clustering to decide the anchor boxes
 - Bounding box regression
 - Multi-scale training (change input image size)
- YOLOv3
 - Binary cross-entropy loss for the class predictions
 - Prediction across scales

YOLO9000: Better, Faster, Stronger. Redmon & Farhadi, CVPR, 2017

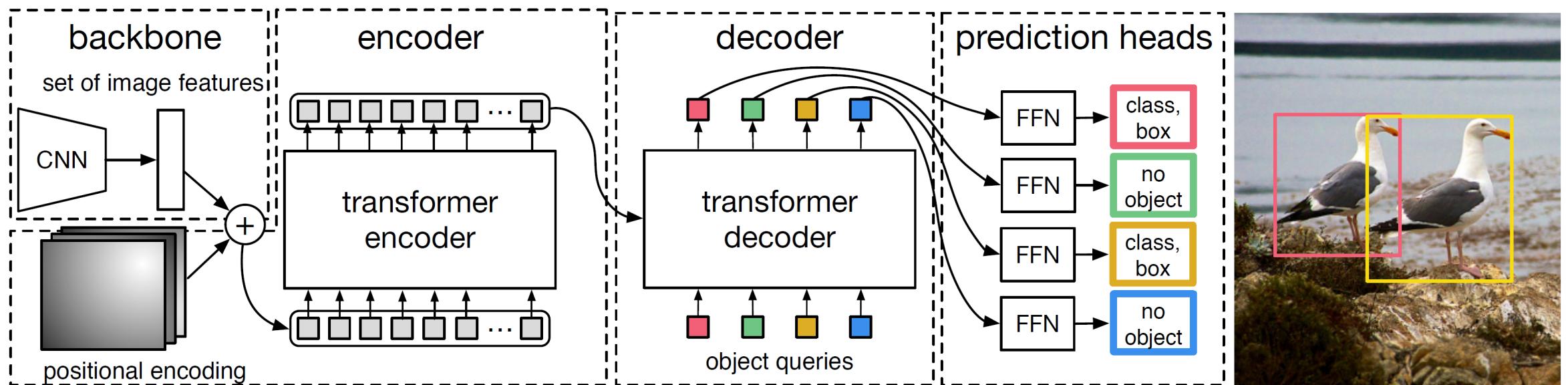
YOLOv3: An Incremental Improvement

Type	Filters	Size	Output
1x	Convolutional	32	256 × 256
	Convolutional	64	128 × 128
	Convolutional	32	1 × 1
	Convolutional	64	3 × 3
2x	Residual		128 × 128
	Convolutional	128	64 × 64
	Convolutional	64	1 × 1
	Convolutional	128	3 × 3
8x	Residual		64 × 64
	Convolutional	256	32 × 32
	Convolutional	128	1 × 1
	Convolutional	256	3 × 3
8x	Residual		32 × 32
	Convolutional	512	16 × 16
	Convolutional	256	1 × 1
	Convolutional	512	3 × 3
4x	Residual		16 × 16
	Convolutional	1024	8 × 8
	Convolutional	512	1 × 1
	Convolutional	1024	3 × 3
	Residual		8 × 8
	Avgpool	Global	
	Connected	1000	
	Softmax		

Table 1. Darknet-53.

DTER

- Vision transformer-based object detection



End-to-End Object Detection with Transformers. Carion et al., ECCV, 2020

DTER

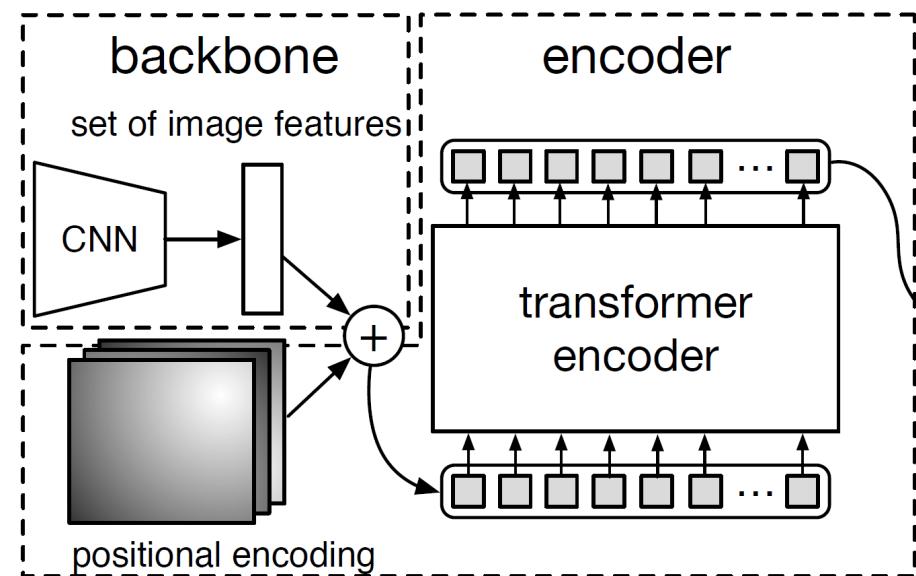
- Backbone

$$x_{\text{img}} \in \mathbb{R}^{3 \times H_0 \times W_0} \longrightarrow f \in \mathbb{R}^{C \times H \times W}$$

$$C = 2048 \quad H, W = \frac{H_0}{32}, \frac{W_0}{32}$$

- Encoder

- 1x1 conv on f $z_0 \in \mathbb{R}^{d \times H \times W}$
- $H \times W$ tokens with d -dimension each

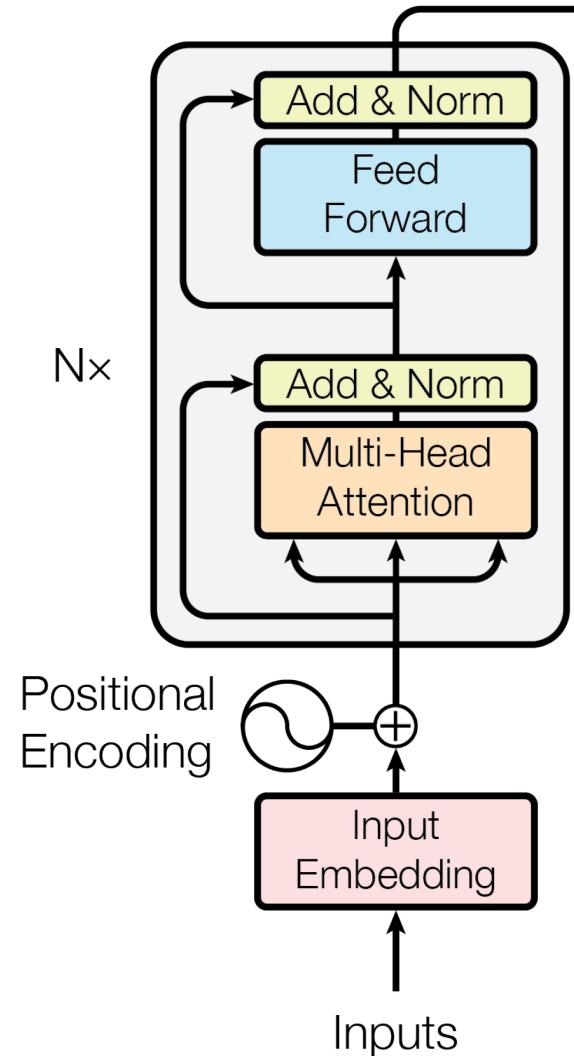


Transformer: Encoder

- Positional encoding
 - Make use of the order of the sequence
 - With dimension d_{model} for each input

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$

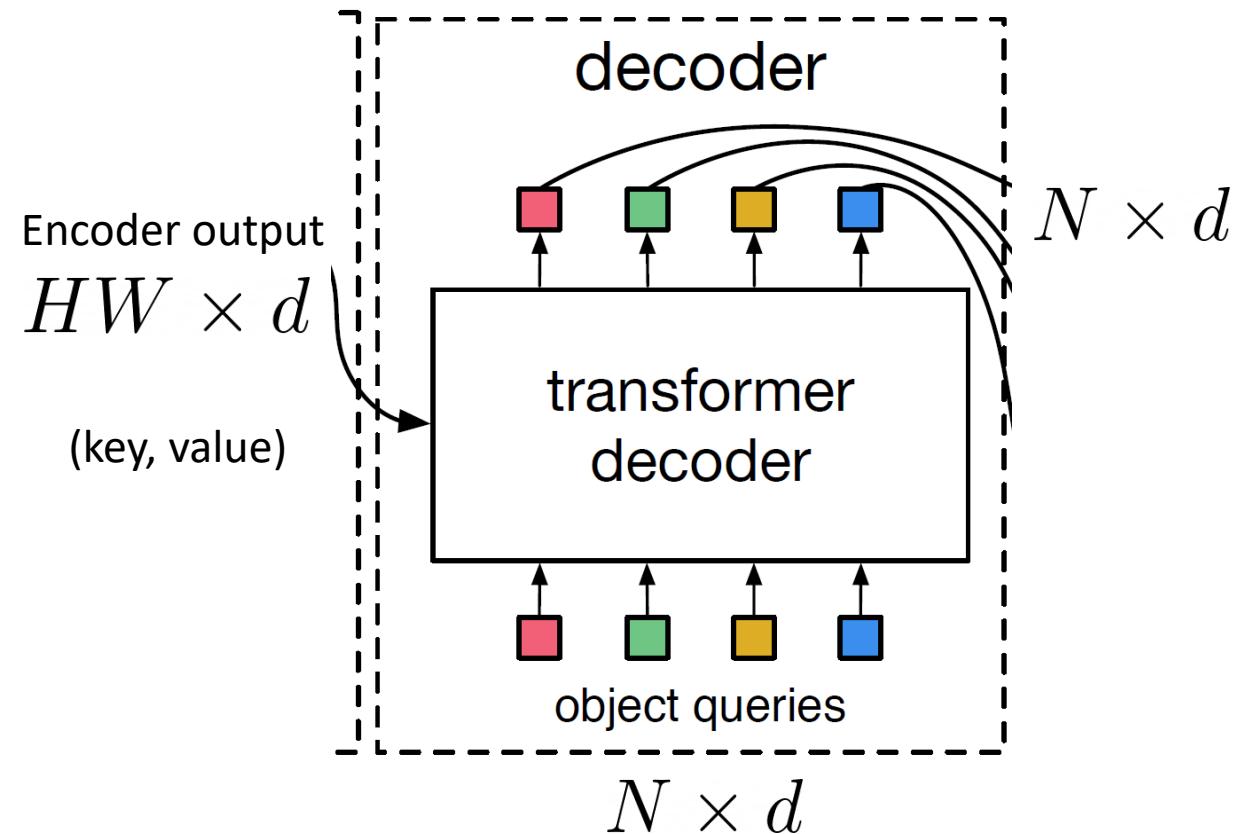
$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$



Attention is all you need. Vaswani et al., NeurIPS'17

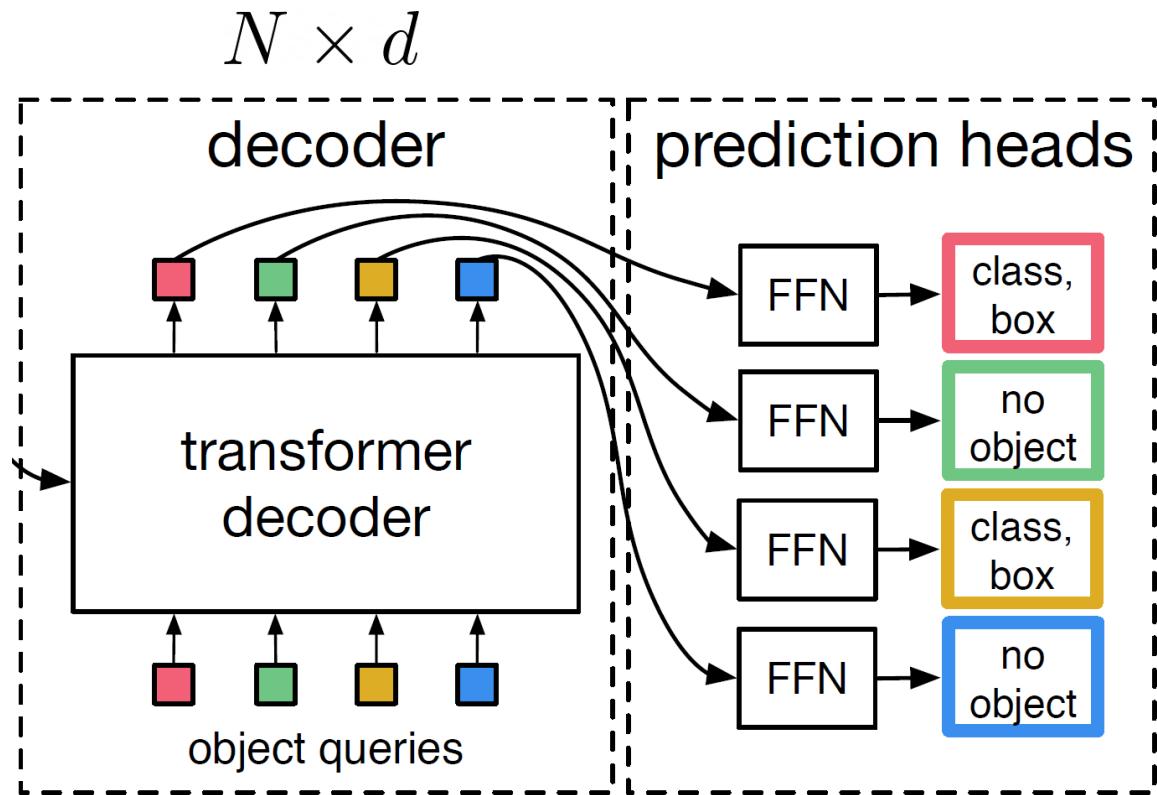
DTER

- Decoder
 - Decodes N object queries in parallel
 - Object queries: learned positional encodings (treat as weights in the network)



DTER

- Prediction heads
 - 3 FC layers
 - Box: normalized (x, y, h, w)
 - Class: softmax prediction with the “no object” class



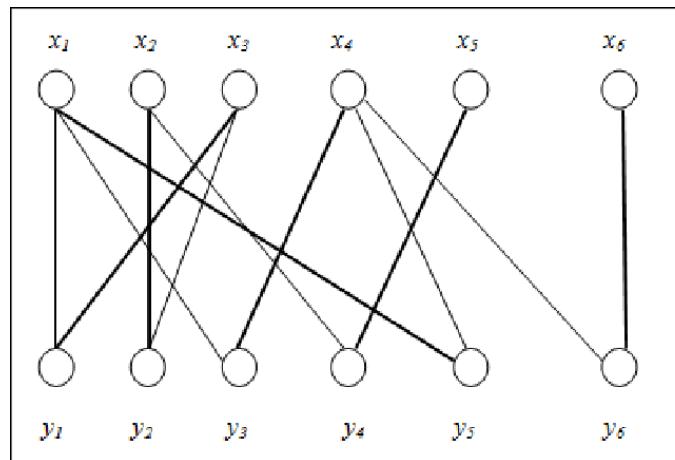
End-to-End Object Detection with Transformers. Carion et al., ECCV, 2020

DTER

- Training
 - bipartite matching between predicted and ground truth objects

Predated boxes $\hat{y} = \{\hat{y}_i\}_{i=1}^N$

Ground truth boxes $y = \{y_i\}_{i=1}^N$
padded with non-object



Hungarian algorithm

$$\mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)}) = -\mathbb{1}_{\{c_i \neq \emptyset\}} \hat{p}_{\sigma(i)}(c_i) + \mathbb{1}_{\{c_i \neq \emptyset\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\sigma(i)})$$

$$\text{Hungarian loss } \mathcal{L}_{\text{Hungarian}}(y, \hat{y}) = \sum_{i=1}^N \left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \emptyset\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}(i)}) \right] \quad \text{Based on optimal assignment}$$

End-to-End Object Detection with Transformers. Carion et al., ECCV, 2020

DETR

Model	GFLOPS/FPS	#params	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
Faster RCNN-DC5	320/16	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster RCNN-FPN	180/26	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster RCNN-R101-FPN	246/20	60M	42.0	62.5	45.9	25.2	45.6	54.6
Faster RCNN-DC5+	320/16	166M	41.1	61.4	44.3	22.9	45.9	55.0
Faster RCNN-FPN+	180/26	42M	42.0	62.1	45.5	26.6	45.4	53.4
Faster RCNN-R101-FPN+	246/20	60M	44.0	63.9	47.8	27.2	48.1	56.0
DETR	86/28	41M	42.0	62.4	44.2	20.5	45.8	61.1
DETR-DC5	187/12	41M	43.3	63.1	45.9	22.5	47.3	61.1
DETR-R101	152/20	60M	43.5	63.8	46.4	21.9	48.0	61.8
DETR-DC5-R101	253/10	60M	44.9	64.7	47.7	23.7	49.5	62.3

DC5: dilated C5 stage

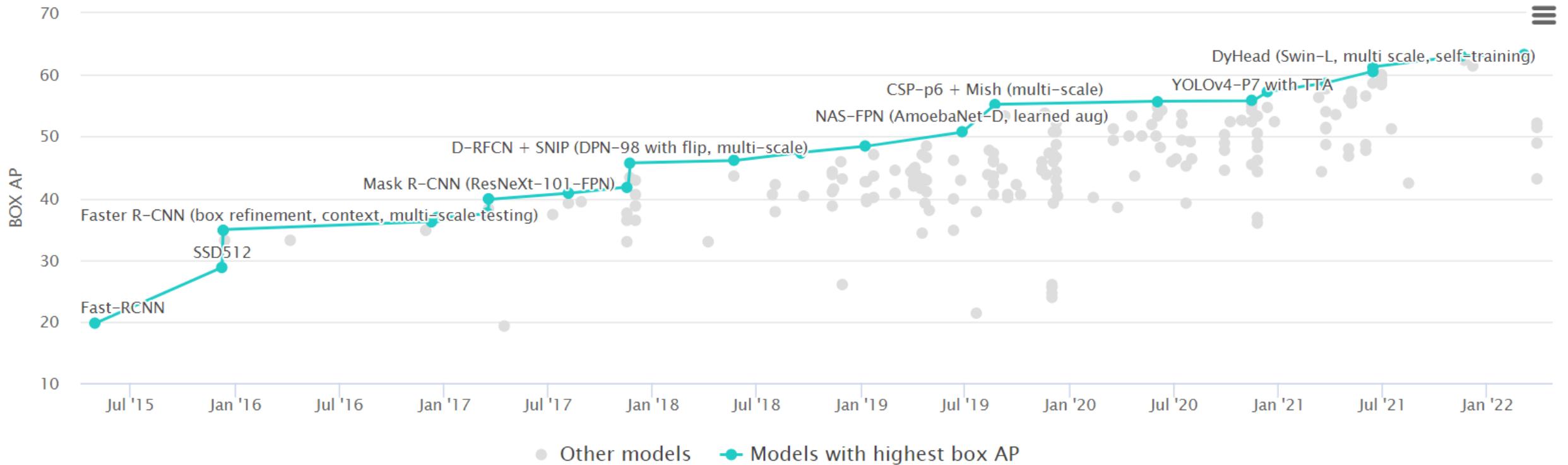
FPN: Feature pyramid networks

End-to-End Object Detection with Transformers. Carion et al., ECCV, 2020

Summary

- Two-stage detectors
 - R-CNN, Fast R-CNN, Faster R-CNN
 - Region proposal + classification
 - Good performance, slow
- One-stage detectors
 - YOLO, SSD
 - End-to-end network to regress to bounding boxes
 - Fast, comparable performance to two-stage detectors
- Transformer-based detectors
 - DTER
 - Attention-based set prediction, using object queries

Object Detection on COCO test-dev



<https://paperswithcode.com/sota/object-detection-on-coco>

Further Reading

- Viola–Jones object detection, 2001
<https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf>
- Deformable part model, 2010,
<https://ieeexplore.ieee.org/document/5255236>
- R-CNN, 2014 <https://arxiv.org/abs/1311.2524>
- Fast R-CNN, 2015 <https://arxiv.org/abs/1504.08083>
- Faster R-CNN, 2015 <https://arxiv.org/abs/1506.01497>
- YOLO, 2015 <https://arxiv.org/abs/1506.02640>
- YOLOv2, 2016 <https://arxiv.org/abs/1612.08242>
- Feature Pyramid Networks, 2017 <https://arxiv.org/pdf/1612.03144.pdf>
- DTER, 2020 <https://arxiv.org/abs/2005.12872>