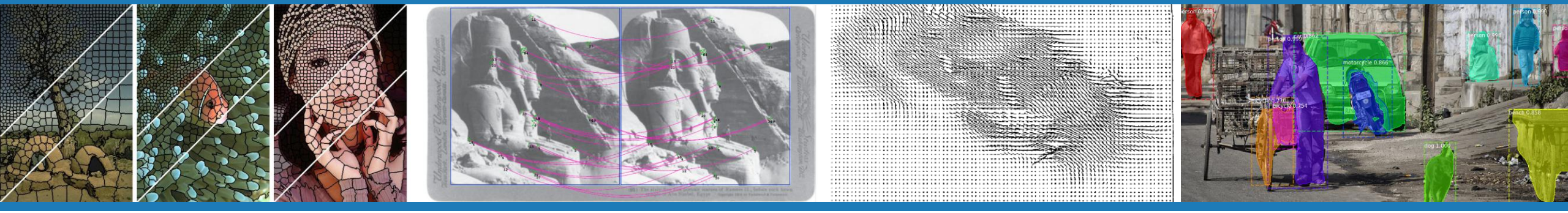
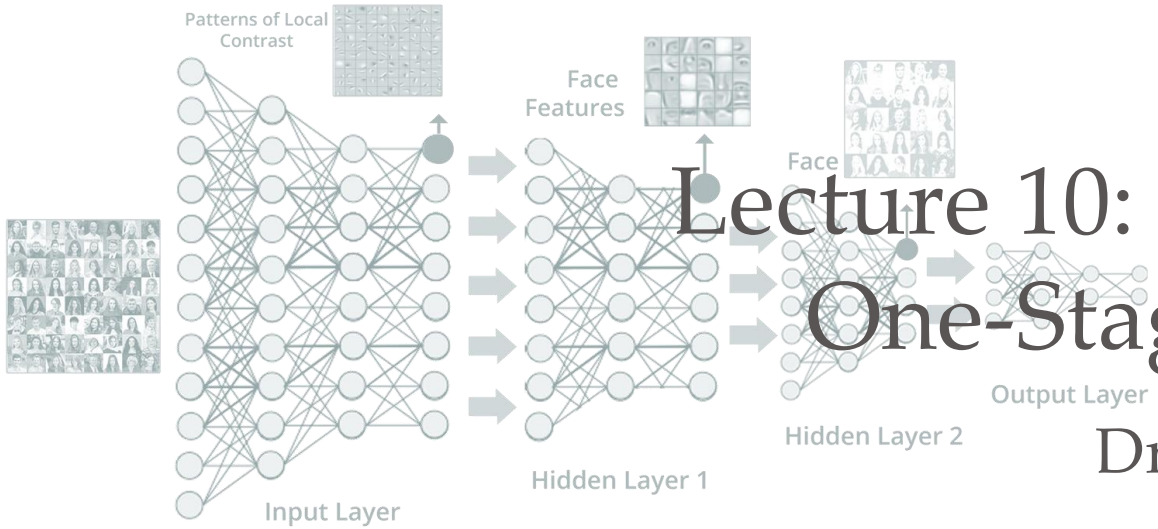
INSTITUTE FOR MACHINE LEARNING AND ANALYTICS



Computer Vision



Lecture 10: Object Detection - One-Stage Approaches

Dr. Xiao Zhao

EMI Elektrotechnik, Medizintechnik und Informatik

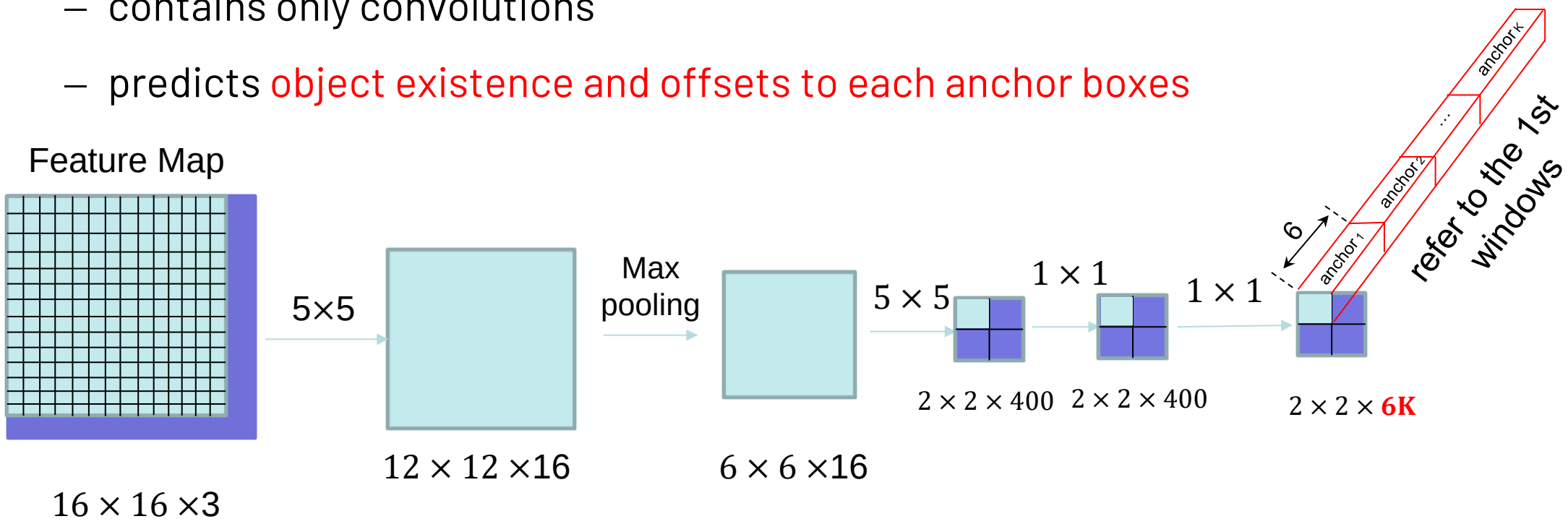
5. Juni 2025

Content

- Introduction
- YOLO v1, v2, v3
- SSD
- Feature pyramid network
- Retina network
- FCOS

Recall: Region Proposal Network (RPN)

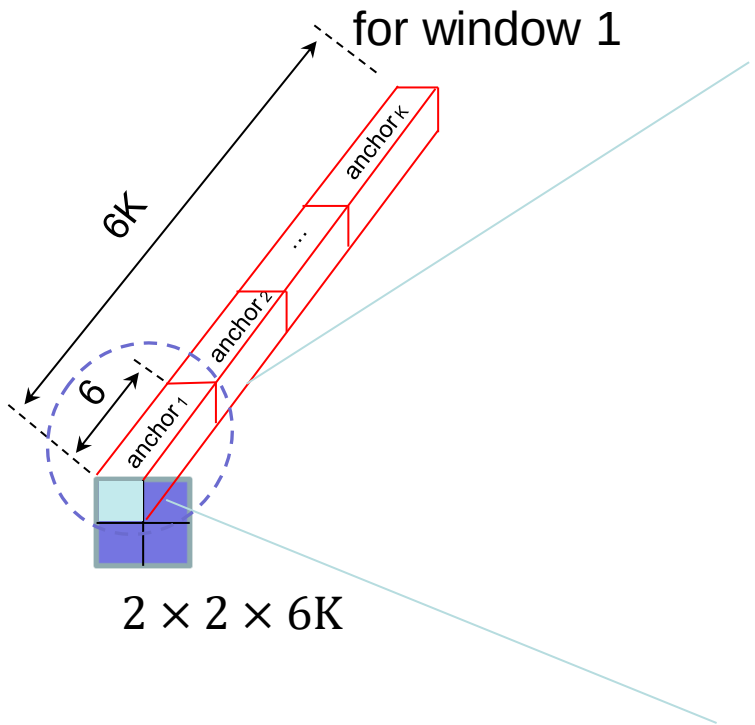
- **Idea**: Multi-scale anchors as regression's reference
- Consider that **in each subgroup/window** K anchor boxes are selected
- Region Proposal Network:
  - contains only convolutions
  - predicts object existence and offsets to each anchor boxes



K: the number of anchors for each window

Recall: Region Proposal Network

- Output of Region Proposal Network

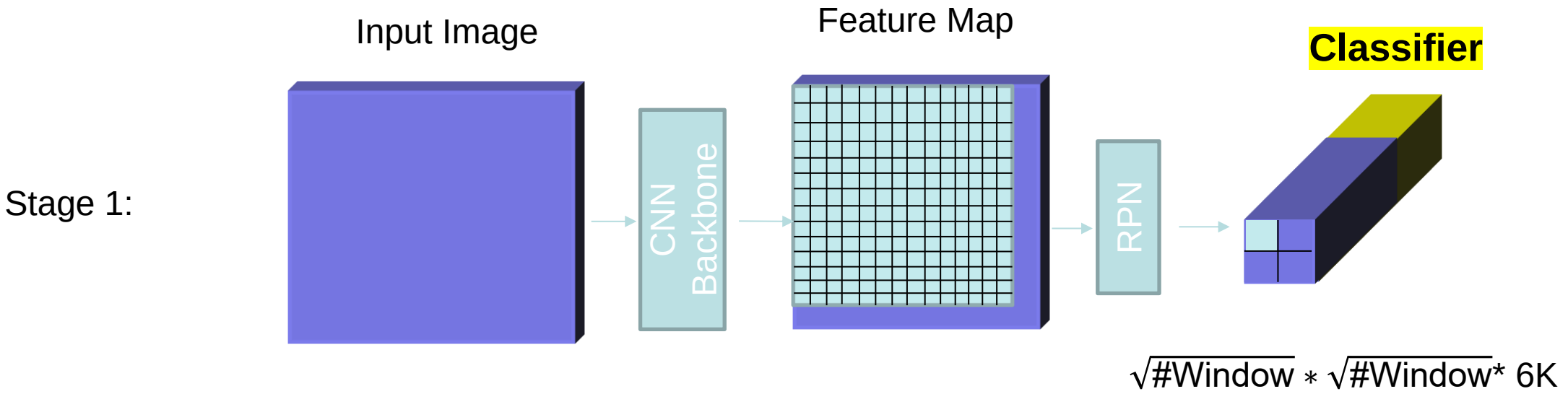


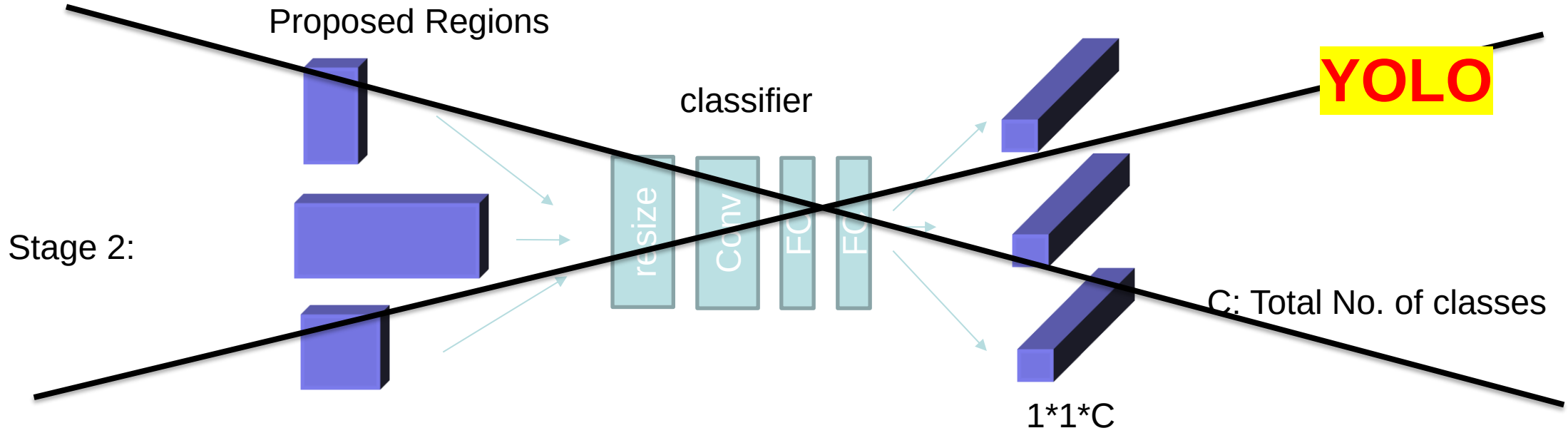
Exist= 0.94	NotE. =0.06	Dx=10	Dy= -5	Dw =8	Dh=-6

existente of an object in anchor 1 (classification)

Object´s offsets to anchor 1 (regression)

Recall: Faster R-CNN, Two-stage Approach





YOLO v1

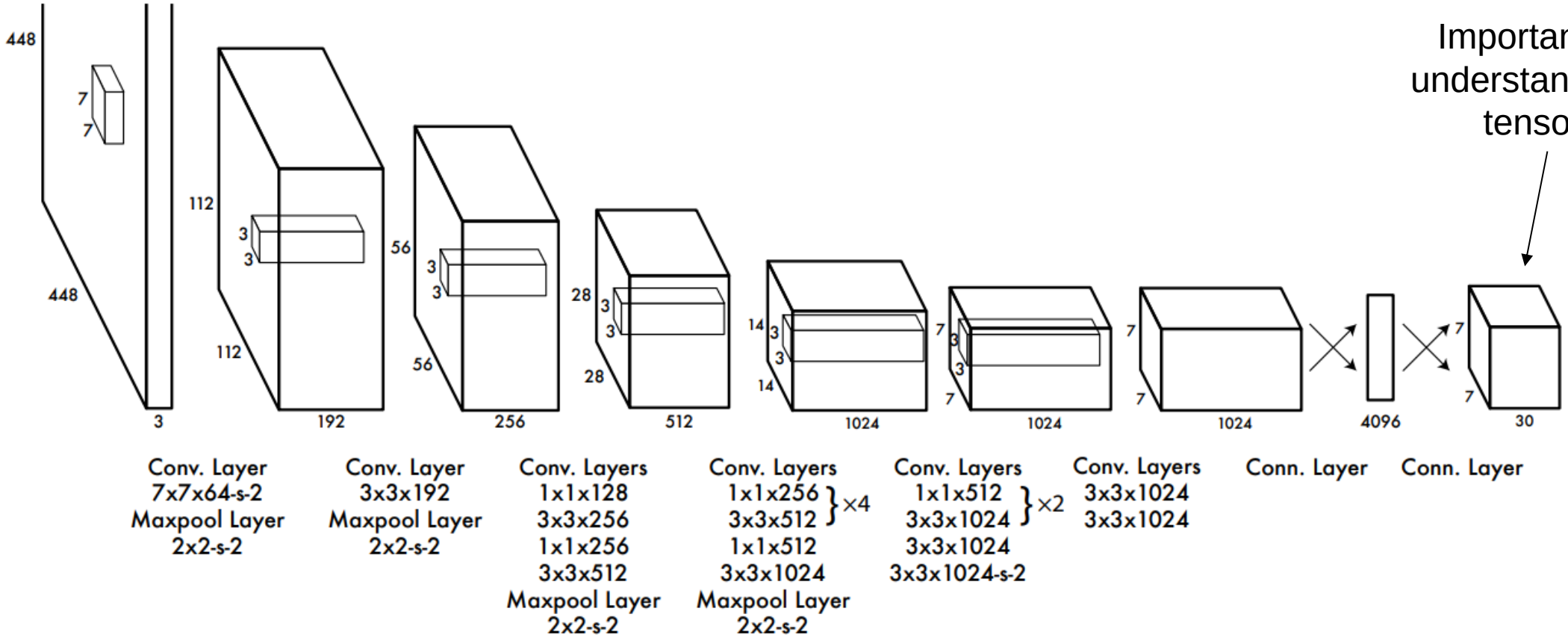
YOLO V1 [1]: „You Only Look Once“

- Motivation:
  - R-CNN, Fast R-CNN, Faster R-CNN are two-stage systems, difficult to train
  - Want to have a single-stage „end-to-end“ trainable network
- Idea:
  - Similar to Faster R-CNN:
    - Keep 1st stage of Faster R-CNN
    - Remove 2nd stage of Faster R-CNN
  - Postprocessing: Non-max supression

[1] Redmon, Divvala, Girshick, Farhadi, You only look once: Unified, real-time object detection, arXiv, 2015

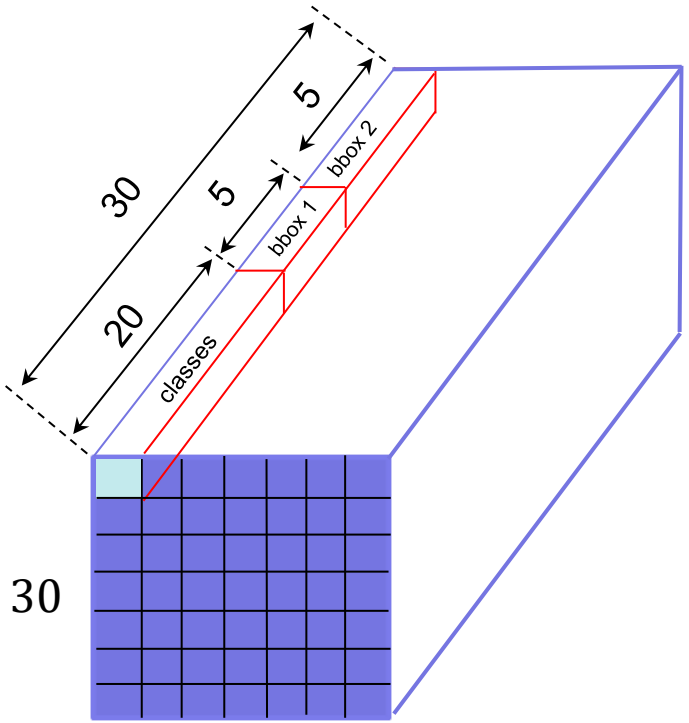
YOLO v1: Network Structure

- Only convolution layers
- Input: image 448*448*3
- Output: tensor: 7*7*30



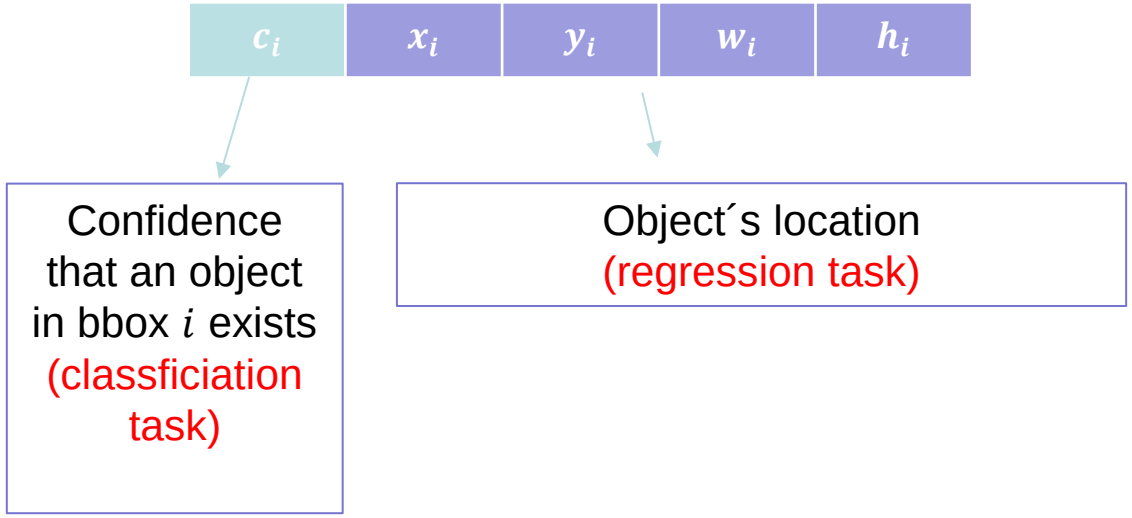
Last Tensor 7*7*30

- No. of predicted bounding boxes: B=2
- No. of classes: S=7

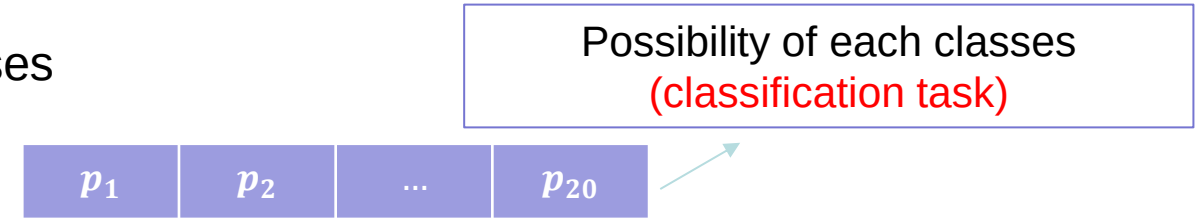


$7 \times 7 \times 30$

- Bbox i:



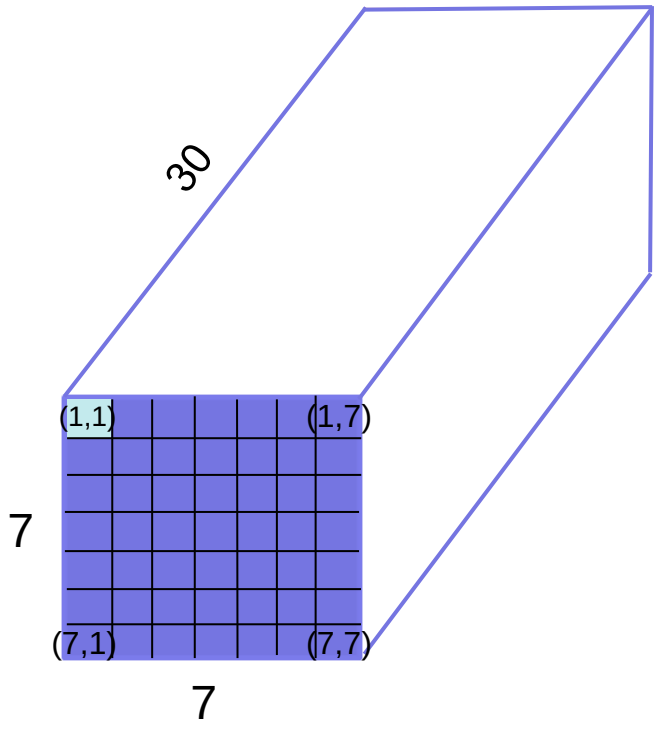
- Classes



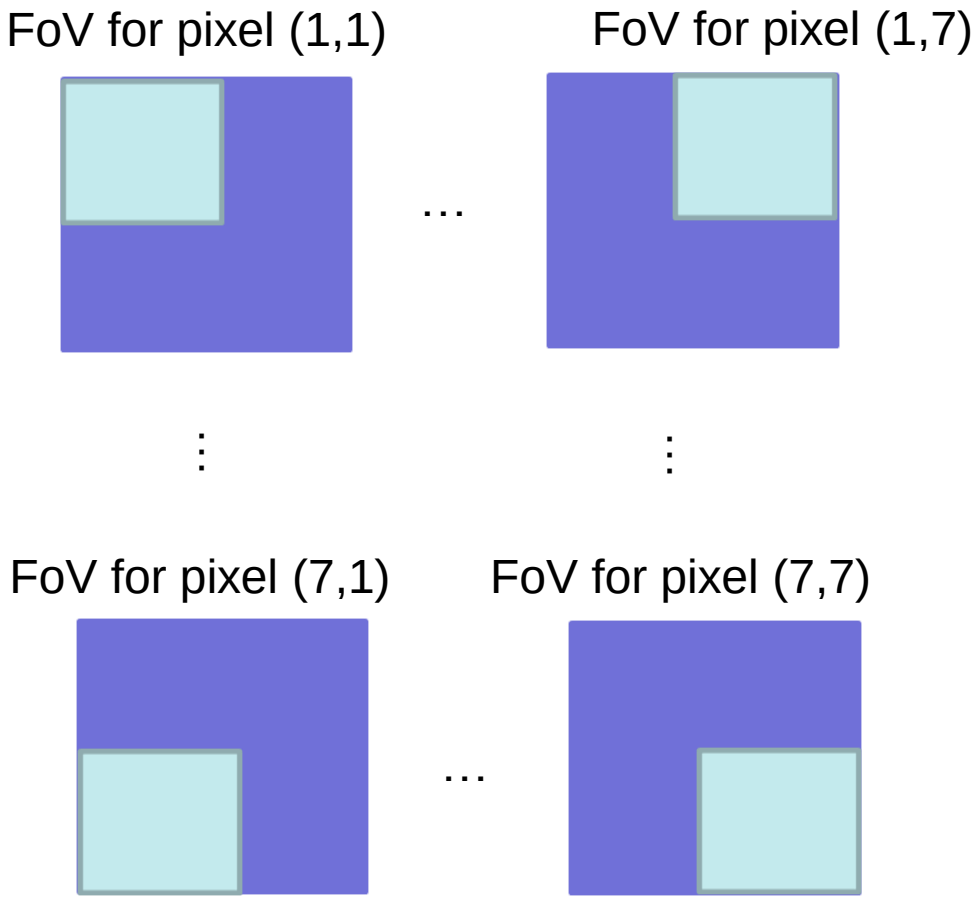
Idea is very similar to Faster R-CNN! However, not explicitly based on anchor.

Last Tensor 7*7*30

- Last Tensor

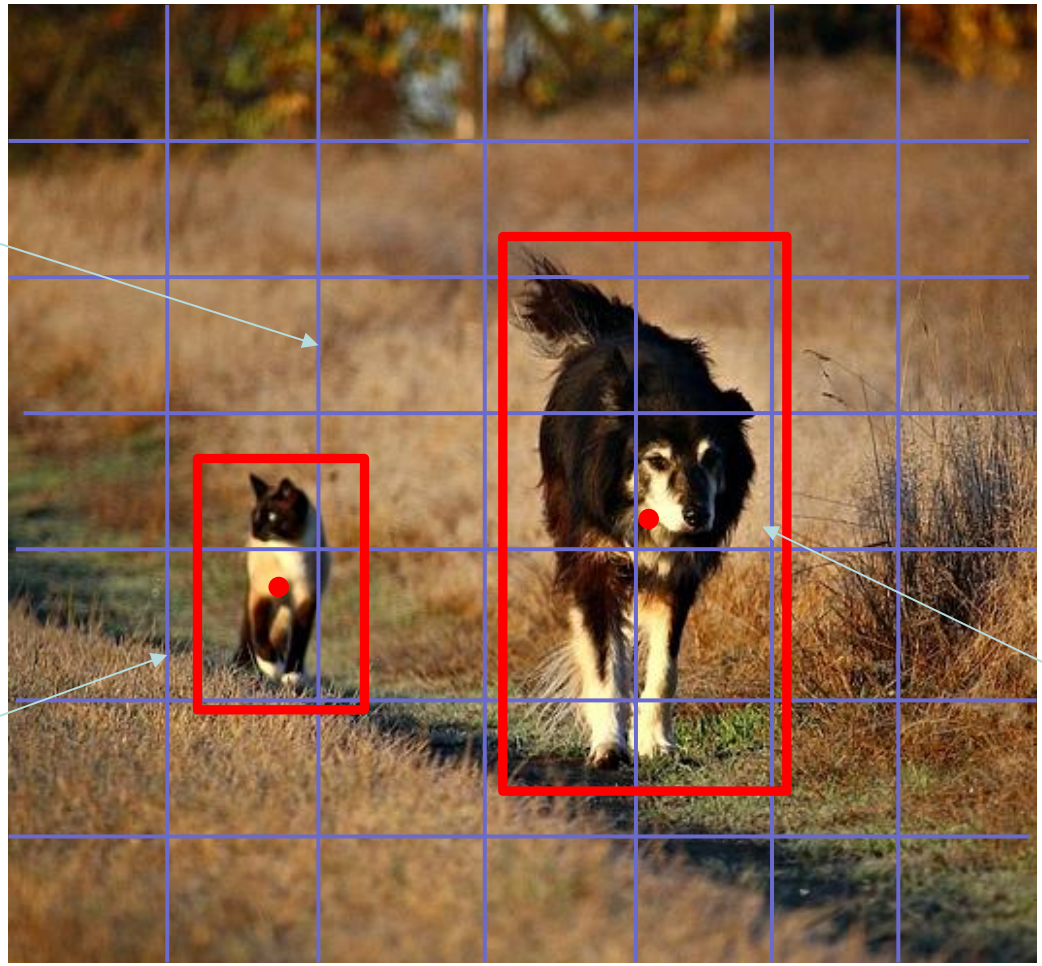


- FoV of each pixel in the output tensor



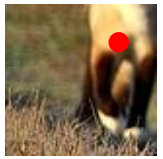
GT of last tensor

$$\begin{bmatrix} p_{dog} \\ p_{cat} \\ c_1 \\ x_1 \\ y_1 \\ w_1 \\ h_1 \end{bmatrix} = \begin{bmatrix} - \\ - \\ 0 \\ - \\ - \\ - \\ - \end{bmatrix}$$



- Seperate image into 7*7 patches
- If the center of a GT Bounding box locates inside a patch, this patch is responsible for detecting this object
- (x, y, w, h) are offsets, not absolute values
- Offsets are easier to learn

$$\begin{bmatrix} p_{dog} \\ p_{cat} \\ c_1 \\ x_1 \\ y_1 \\ w_1 \\ h_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0.8 \\ 0.2 \\ 1.0 \\ 1.8 \end{bmatrix}$$



w.r.t. the patch

$$\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0.1 \\ 0.8 \\ 2.0 \\ 4.0 \end{bmatrix}$$

GT of last tensor

- For two bbox prediction at each position (B=2)

„YOLO predicts multiple bounding box per grid cell. At training time, we only want one bounding box predictor to be responsible for each object. We assign one predictor to be „respoinsible“ for predicting an object based on which prediction has the highest current IOU with the ground truth.“ [1]

$$\begin{bmatrix} p_{dog} \\ p_{cat} \\ c_1 \\ x_1 \\ y_1 \\ w_1 \\ h_1 \\ c_2 \\ x_2 \\ y_2 \\ w_2 \\ h_2 \end{bmatrix} \begin{matrix} \\ \\ \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} \text{Bbox 1} \\ \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} \text{Bbox 2} \end{matrix}$$

[1] Redmon, Divvala, Girshick, Farhadi, You only look once: Unified, real-time object detection, arXiv, 2015

Training: Objective Function

Loss for location of each predicted bbox

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

Loss for confidence (exitence of an object)

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

Loss for classification (dog or cat)

- $1_i^{obj}(\cdot)$: if the center of an GT bbox locate in i-th patch
- $1_{ij}^{obj}(\cdot)$: if j-th bbox of patch i is responsible for predict

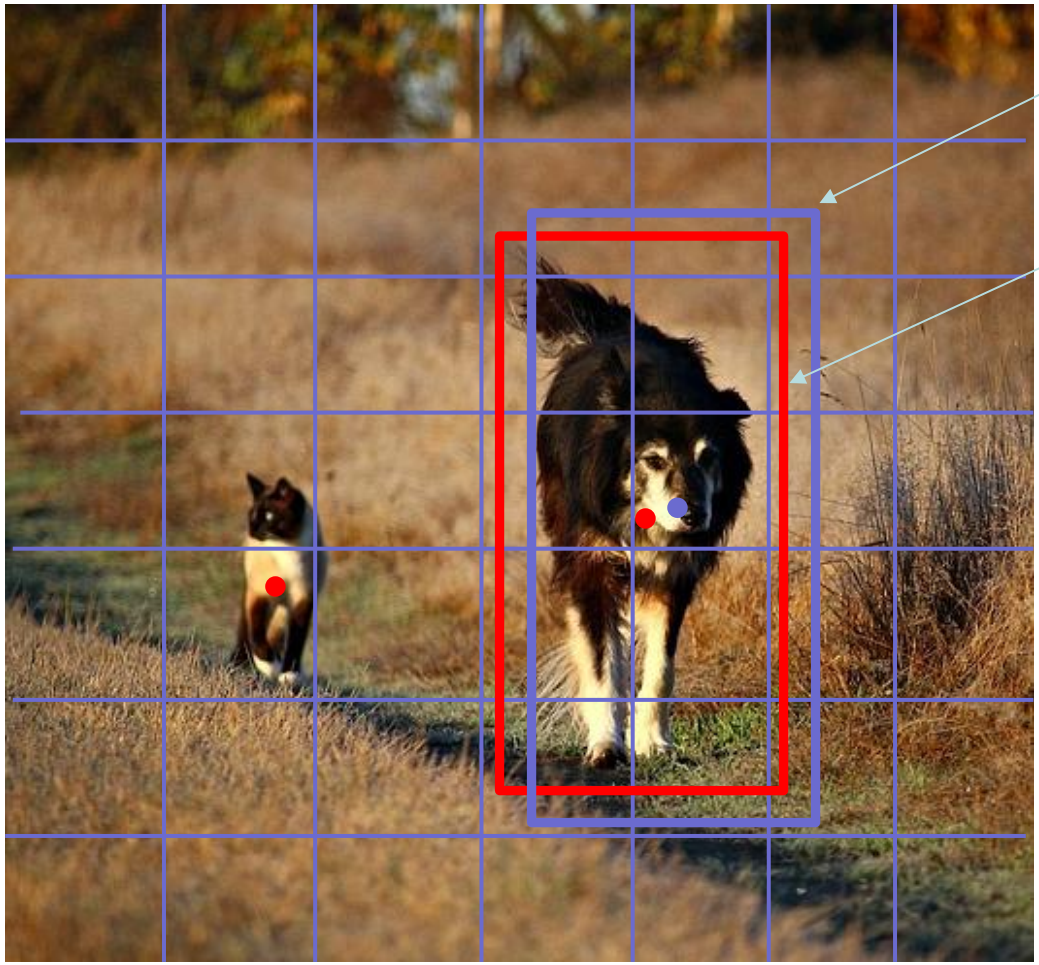
Note:

- C_i: Confidence of patch i
- MSE is used for classfication

At Prediction Time …

- At each position, multiple bboxes (B=2) may be predicted.

e.g. $\begin{bmatrix} p_{dog} \\ p_{cat} \\ c_1 \\ x_1 \\ y_1 \\ w_1 \\ h_1 \\ c_2 \\ x_2 \\ y_2 \\ w_2 \\ h_2 \end{bmatrix} = \begin{bmatrix} 0.95 \\ 0.05 \\ 0.9 \\ 0.12 \\ 0.85 \\ 2.2 \\ 4.1 \\ 0.8 \\ 0.1 \\ 0.8 \\ 2.2 \\ 4.3 \end{bmatrix}$ Bbox 1 Bbox 2

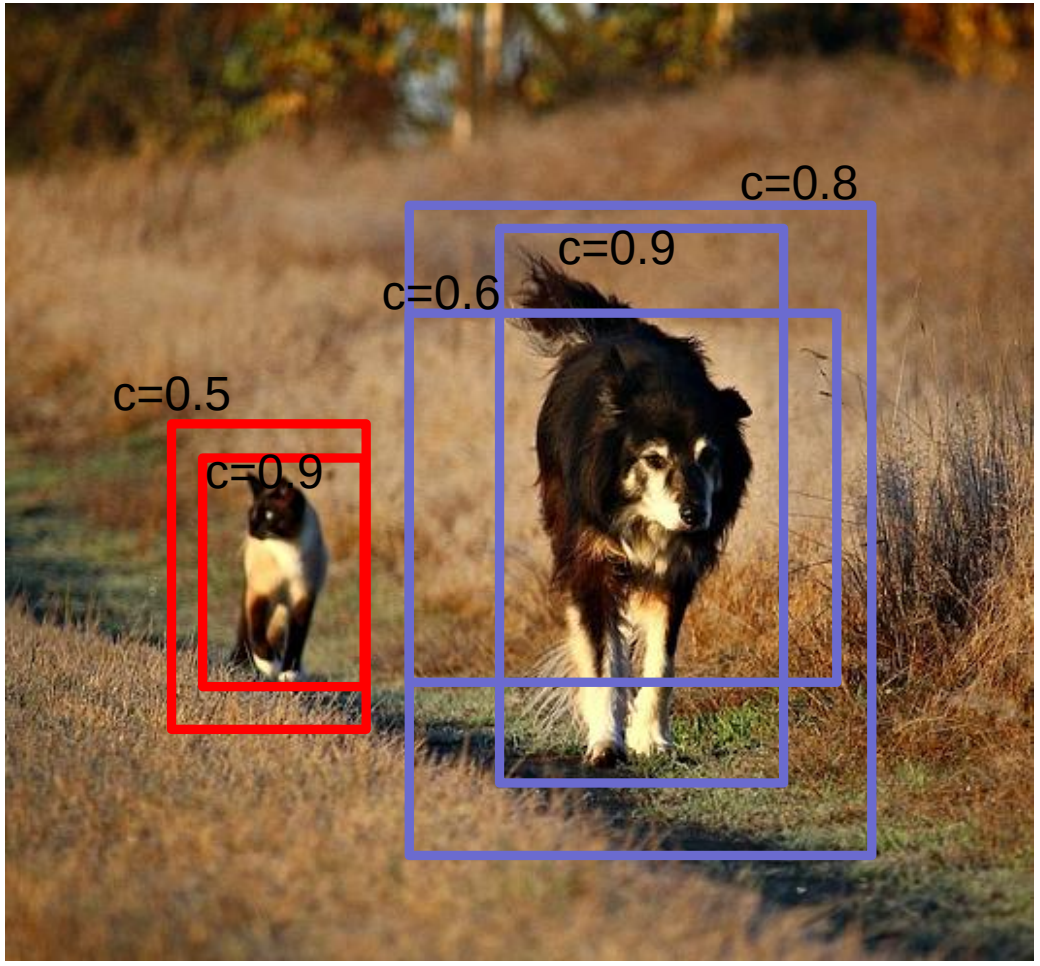


Bbox 1 for position (5, 4)

Bbox 2 for position (5, 4)

At Prediction Time …

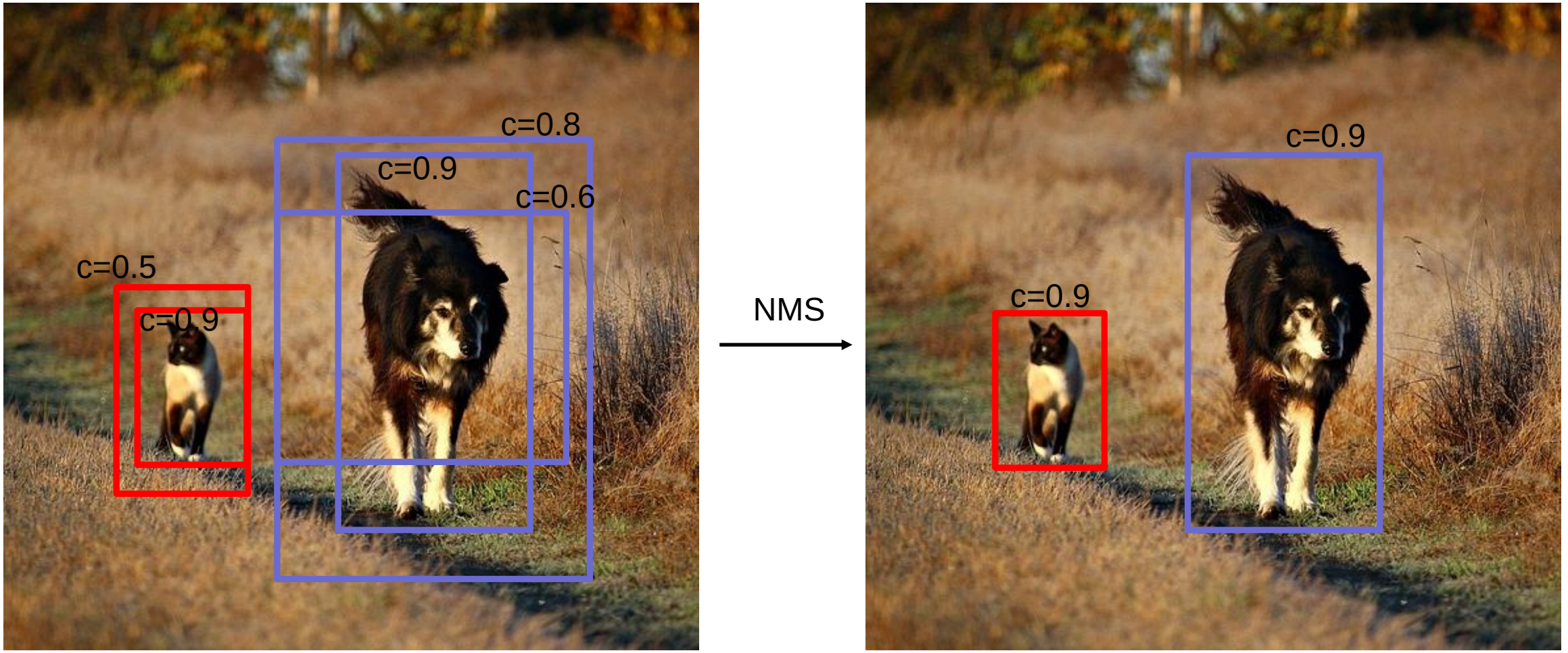
- Multiple Bboxes may be predicted for a single object



What to do?

Non-Maximum Suppression (NMS)

- NMS clean up redundant bboxes by IoU



Non-Maximum Suppression (NMS)

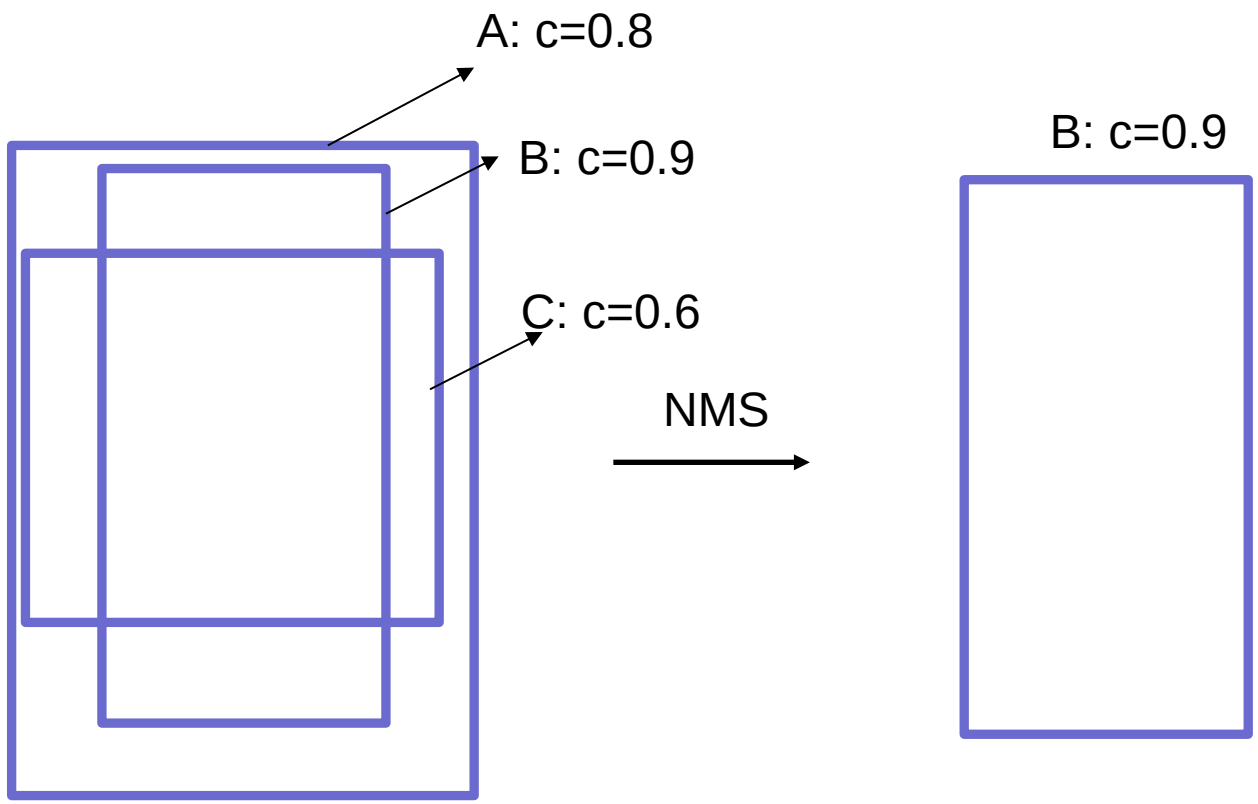
Steps (for each class):

1. Discard bboxes, if confidence is too low, e.g ≤ 0.5

2. Select a bbox with highest confidence
   → B is selected

3. Depress the other bboxes, if IoU≥0.5
   → IoU(B,A) = 0.8
   → IoU(B,C) = 0.6
   → A, C are depressed

4. In the rest bboxes, select a bbox with the highest confidence and repeat



YOLO V1: Summary

- In one forward path: prediction of all candidate bounding boxes for an image!!

  →„You only look once“

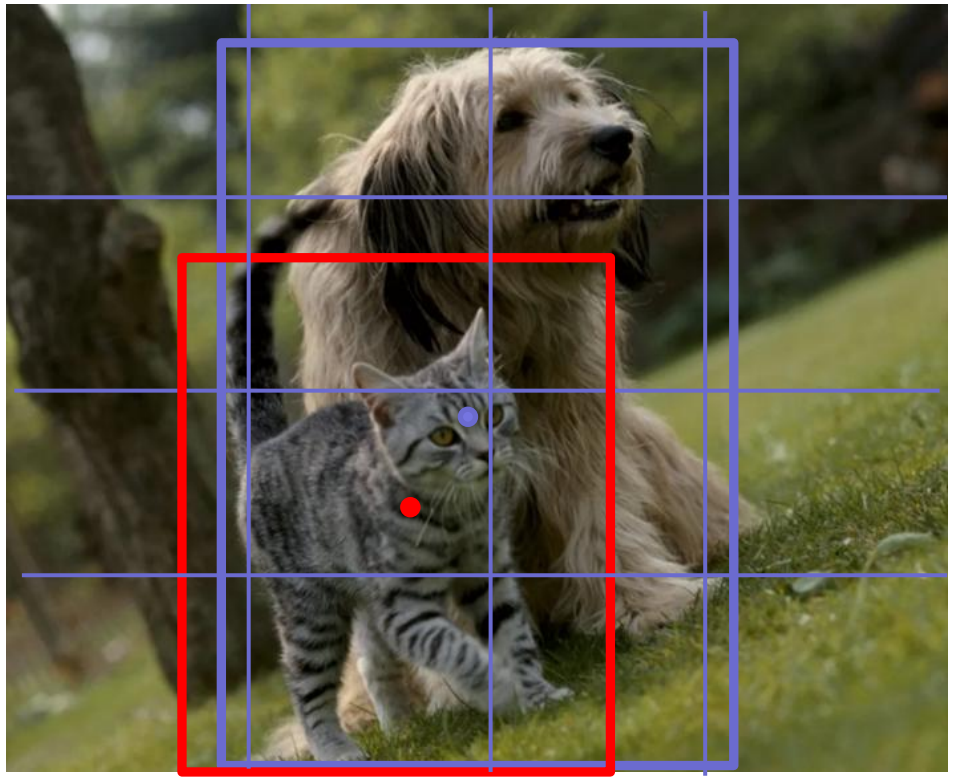
  →„end-to-end” trainable

- NMS as Post-processing
- An extended version of Region Proposal Network
  - Object Classification is added to RPN
- Limitation:
  - Fixed total number of identifiable objects: 7*7*B
  - Each patch has only 1 class -> All B bboxes of this patch can only have the same class

YOLO V1: Summary

- Limitation:
  - If two objects have their centers in a single patch, difficult to encode them in the GT last tensor
  - No anchor used. (x, y, w, h) is refer to the size of each patch

    → Not feasible in identifying objects in different scales and/or different h/w ratio



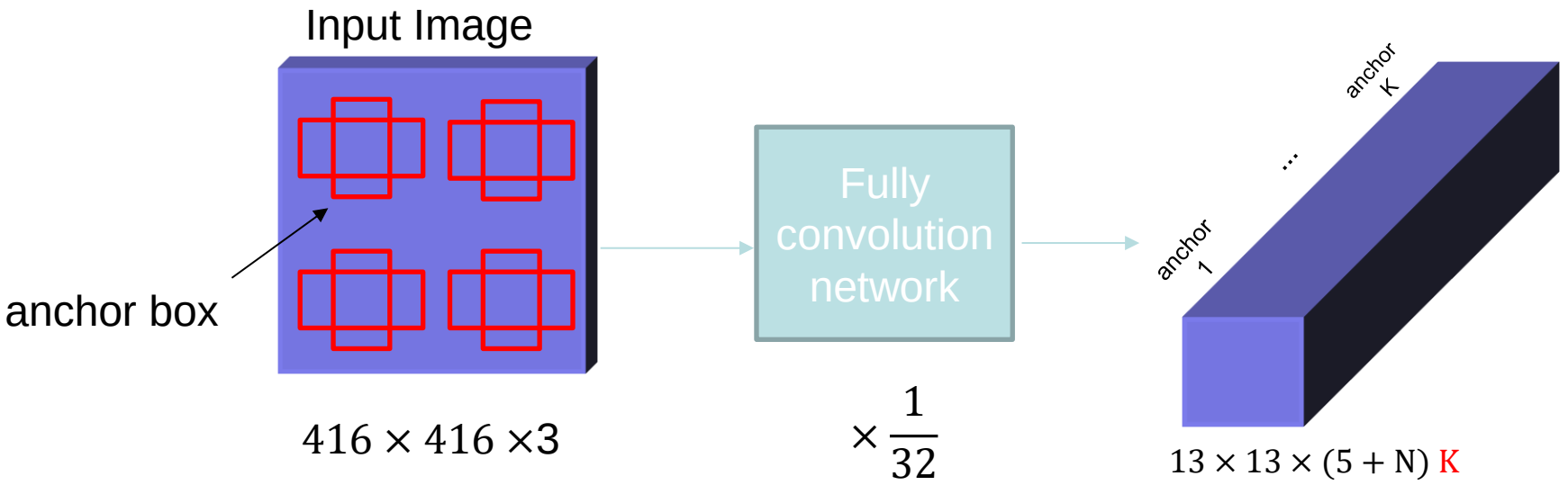
YOLO v2, v3

YOLO V2 [2]

- Yolo V1:
  - Do not use anchor box *directly*
  - 7*7 „grid cell“ can be viewed as anchor boxes
  - Every "grid cell" in YOLO V1 can predict only 1 class
- Yolo V2:
  - Use anchor box
  - Each „grid cell“ in Yolo V2 can predict multiple classes

[2] Redmon, Farhadi, YOLO9000: better, faster, stronger, arXiv, 2016

YOLO V2 [2]



Every anchor: $[c_i, x_i, y_i, w_i, h_i, p_1, p_2, \ldots, p_N]$

confidence | location | class

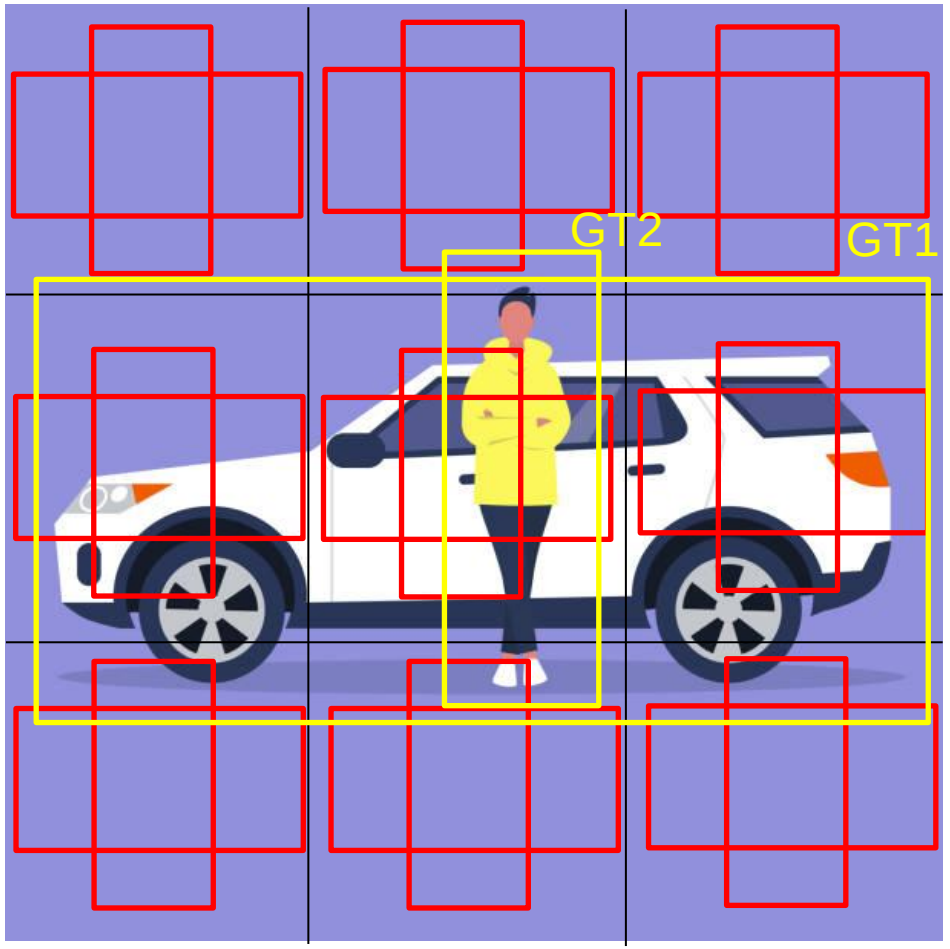
- Predict class and confidence for each anchor box!!
- Why?
  - → Avoid problem that centers of multiple classes inside a single patch
  - → Much more number of objects can be predicted
    - YOLO v1: 98, YOLO v2: >1000

Create GT Tensor: Example

- Assume:
  - Last tensor has shape: 3*3***(5+N)K**
  - 2 anchor boxes at each position
  - Exist 2 objects and their GT bbox are known

- We have:
  - in total 3*3*2=18 anchors

Which anchor to encode which GT Box?



GT of the last Tensor

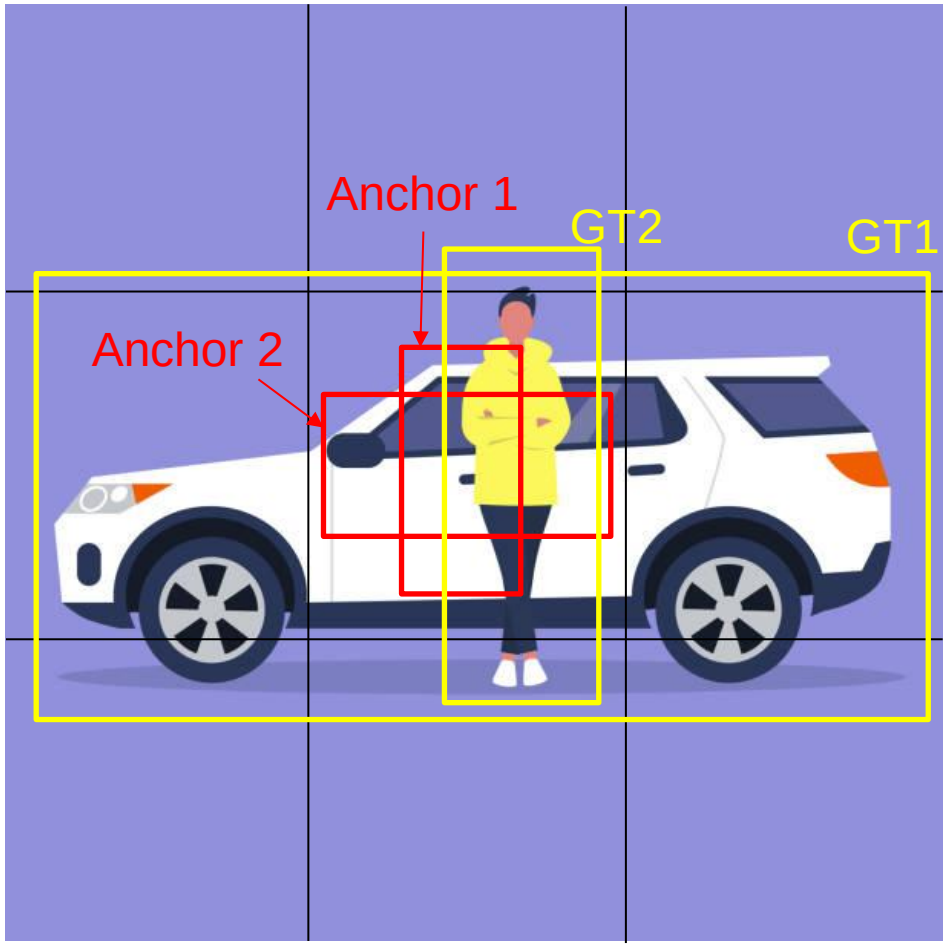
- **Rule 1**: Each object is assigned to a grid cell, which contains the center point of this object

- **Rule 2**: Each object is assigned to an anchor box of the above grid cell, which has the max IoU w.r.t. the other anchor boxes of that grid cell

Create GT Tensor: Example

- By Rule 1:
  - Center of GT1 locates in patch (2, 2)
  - Center of GT2 locates in patch (2, 2)
  - Therefore, patch (2, 2) is responsible for predicting GT1 & GT2

- By Rule 2:
  - IoU(GT2, anchor1) > IoU(GT2, anchor2)
  - IoU(GT1, anchor1) ≤ IoU(GT1, anchor2)
  - Therefore, anchor 1 is responsible for predicting GT2
  - anchor 2 is responsible for predicting GT1



Create GT Tensor: Example

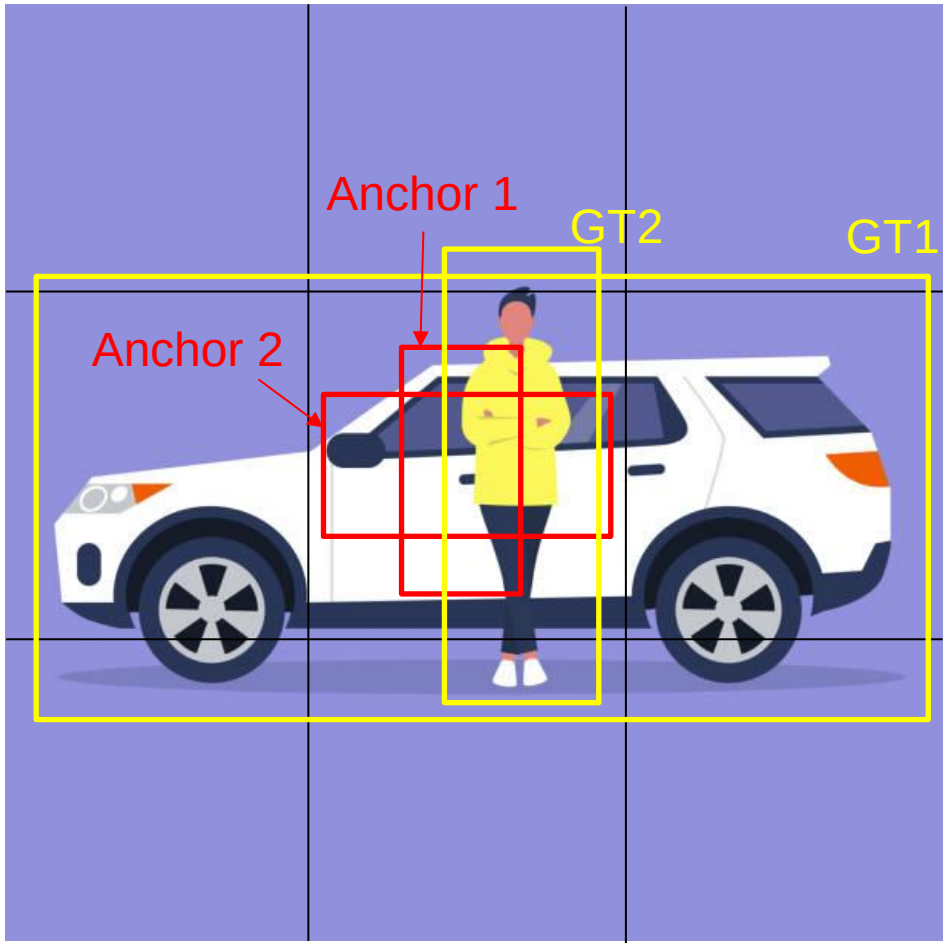
(GT)

Anchor 1 $\begin{bmatrix} c_{per} \\ c_{car} \\ p_1 \\ x_1 \\ y_1 \\ w_1 \\ h_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ x_1 \\ y_1 \\ w_1 \\ h_1 \end{bmatrix}$

Anchor 2 $\begin{bmatrix} c_{per} \\ c_{car} \\ p_2 \\ x_2 \\ y_2 \\ w_2 \\ h_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ x_2 \\ y_2 \\ w_2 \\ h_2 \end{bmatrix}$

(GT)

Other Anchors: $\begin{bmatrix} c_{per} \\ c_{car} \\ p_i \\ x_i \\ y_i \\ w_i \\ h_i \end{bmatrix} = \begin{bmatrix} ? \\ ? \\ \mathbf{0} \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$

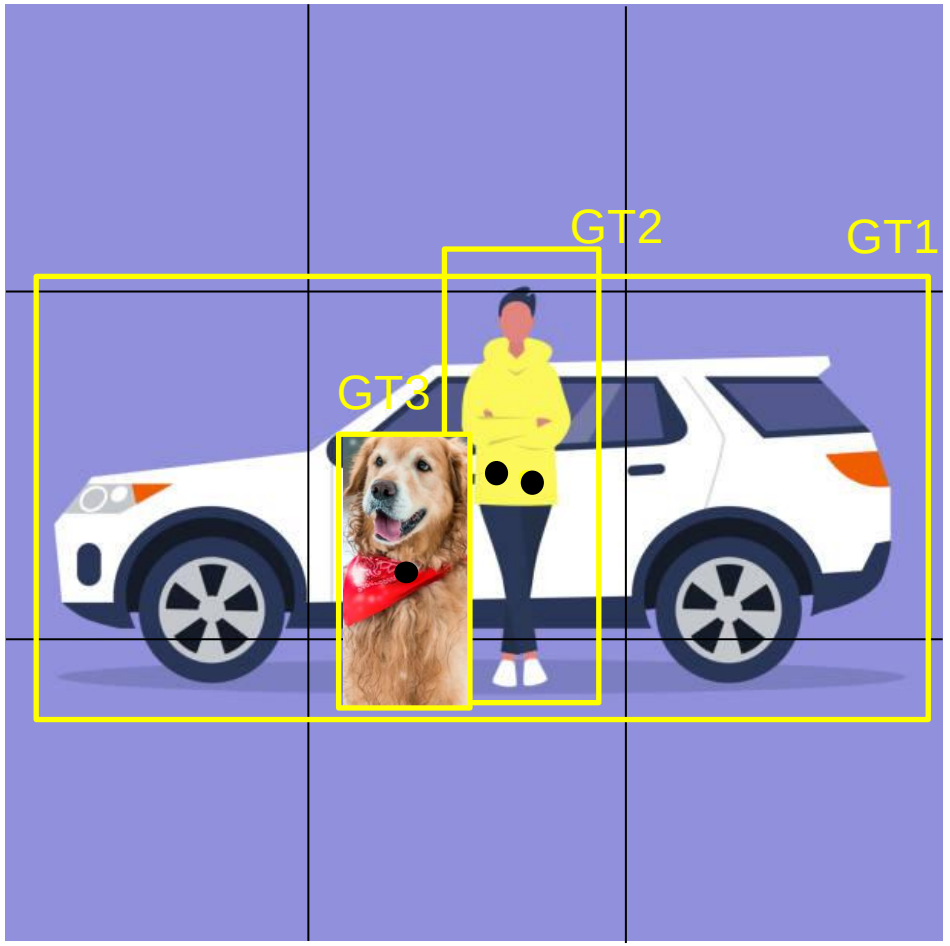


Create GT Tensor: Example

- What happens, if a third object (e.g. a dog) exists?

→ Not possible to encode the GT bbox for the dog in the last tensor, because K=2

→ All anchors for patch (2,2) are used



YOLO V2 [2]: Summary

- YOLO V1: no anchor box
- YOLO V2: use anchor box
- Fully convolution network
- For each anchor at each position, predict:

  confidence + **class** + **bbox position**

- YOLO v1: mAP = 63.4
- YOLO v2: mAP = 76.8

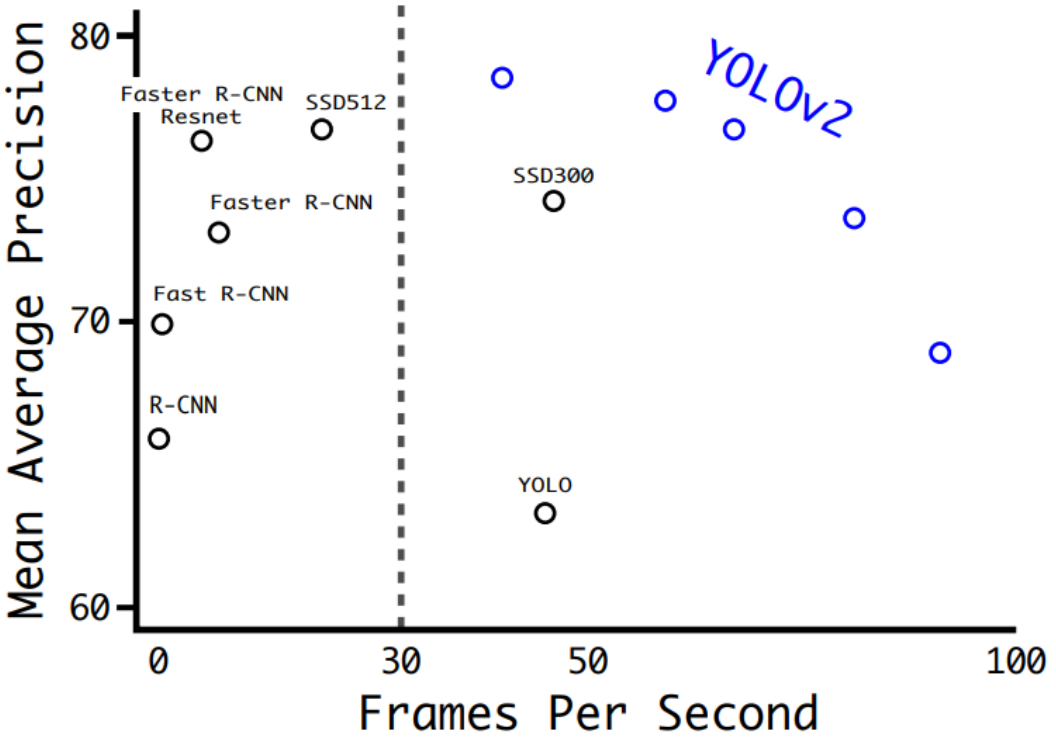
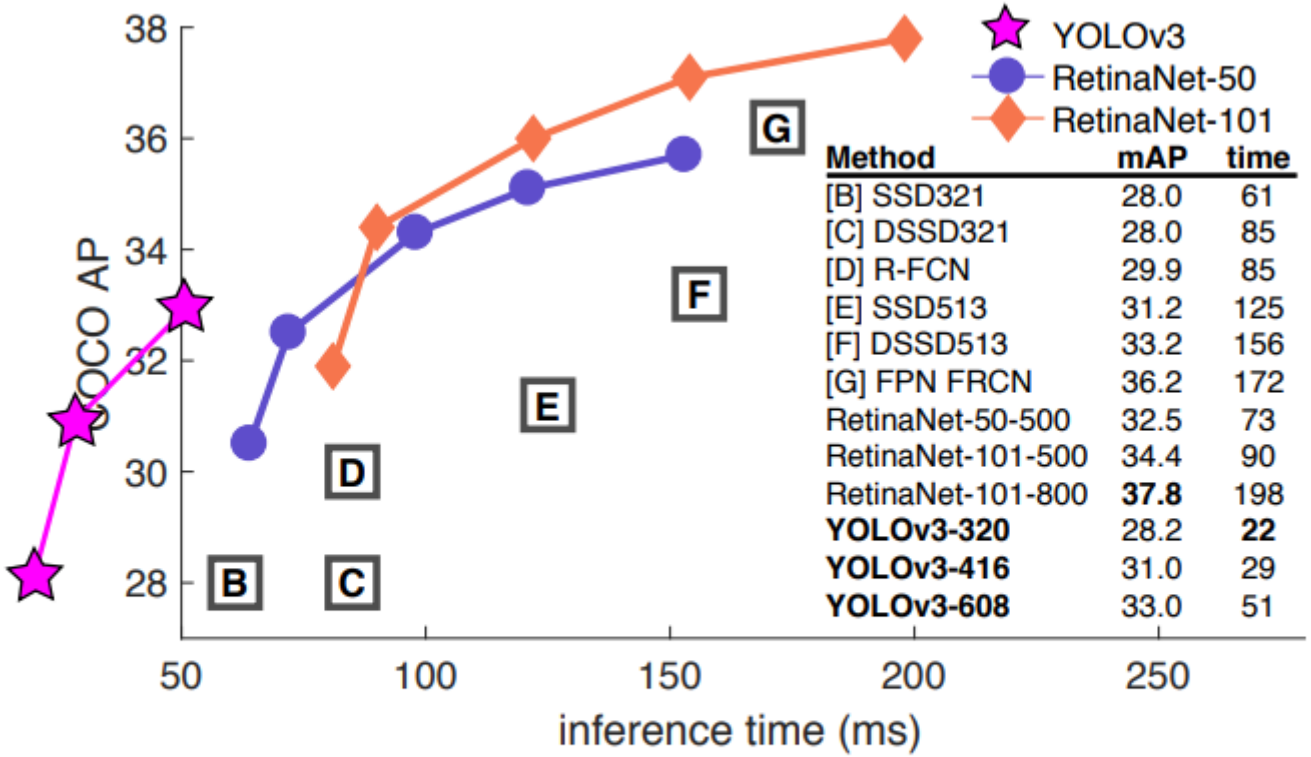


Figure 4: Accuracy and speed on VOC 2007.

[2] Redmon, Farhadi, YOLO9000: better, faster, stronger, arXiv, 2016

YOLO V3 [3]



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

Figure 1. We adapt this figure from the Focal Loss paper [9]. YOLOv3 runs significantly faster than other detection methods with comparable performance. Times from either an M40 or Titan X, they are basically the same GPU.

[3] Redmon, Farhadi, YOLOv3: An Incremental Improvement, arXiv, 2018

SSD: Single Shot Multibox Detector

SSD: Single Shot Multibox Detector

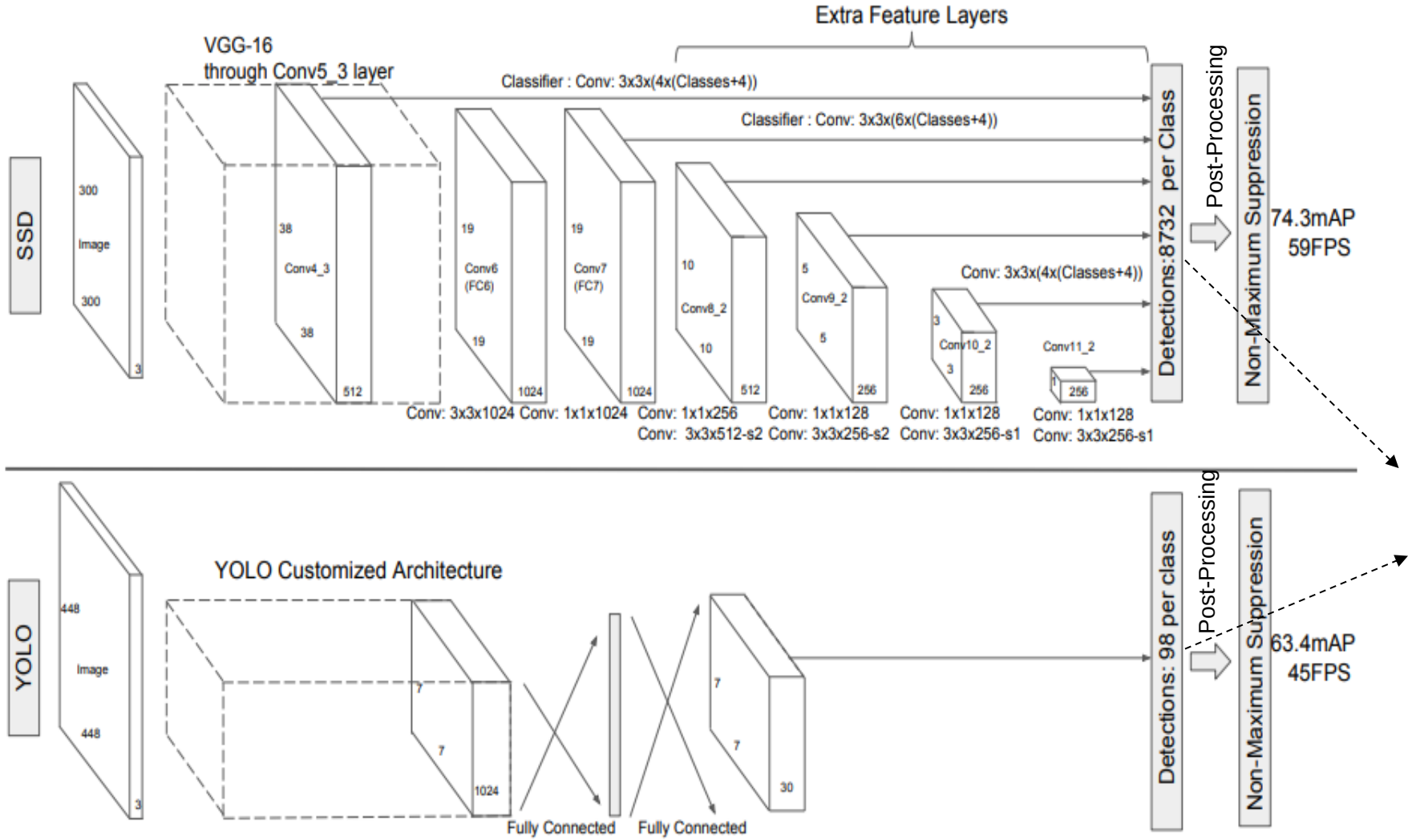
- What is new?
  - Using multiple feature maps for prediction at different scales
  - A previous work of FPN (feature pyramid network) [5]

- Performance [4]:
  - Faster R-CNN: 7 FPS, mAP=73.2%
  - YOLO: 45 FPS, mAP=63.4%
  - SSD: 59 FPS, mAP=74.3

FPS: frames per second

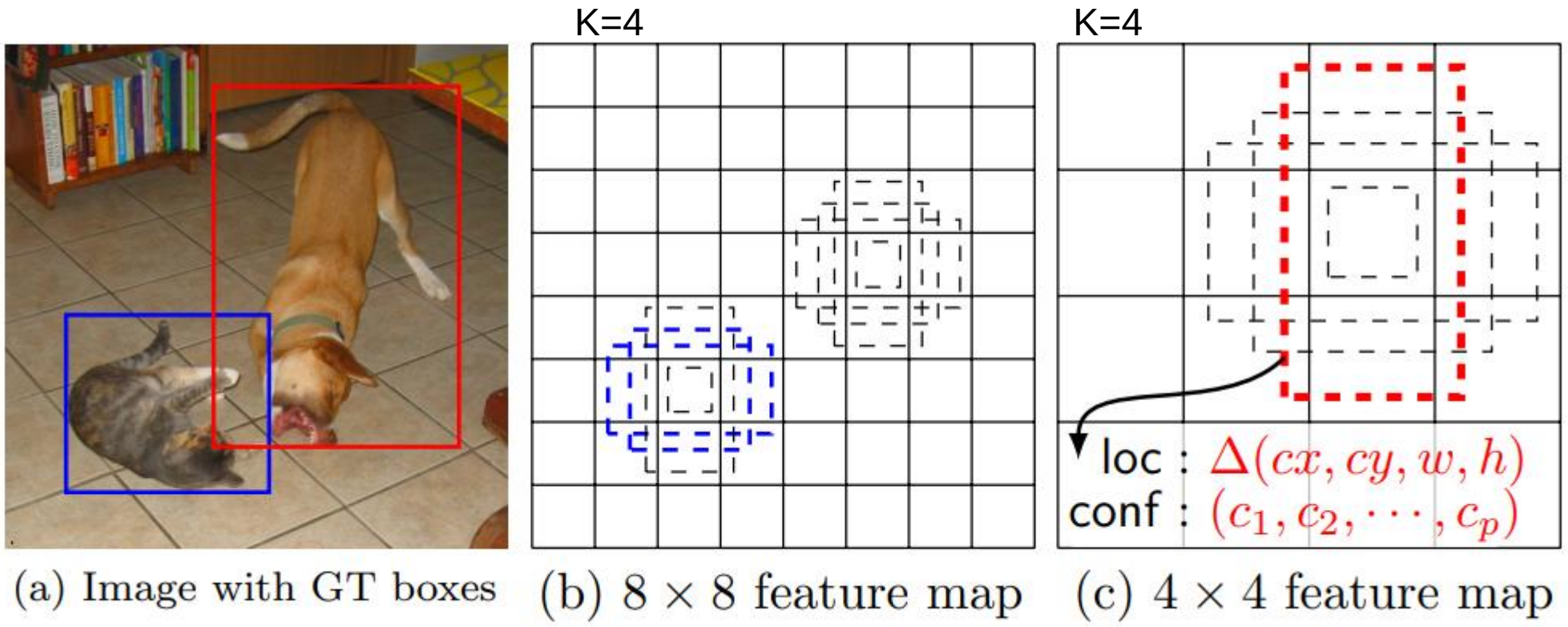
[4] Liu, et al., SSD: single, shot multibox detector, arXiv, 2016
[5] Lin, et al., Feature pyramid networks for object detection, arXiv, 2017

SSD: Single Shot Multibox Detector



- Use multi-scale feature maps from different layers
- Predict Classes+4* parameters for each anchor (same as YOLO)
- Post-processing (only during prediction): NMS
- SSD: 8732 anchor boxes
- YOLO: 98 anchor boxes

SSD: Single Shot Multibox Detector



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

- #Anchor=8*8*K
- Tensor of GT: 8*8*K*(c+4)

- #Anchor=4*4*K
- Tensor of GT: 4*4*K*(c+4)

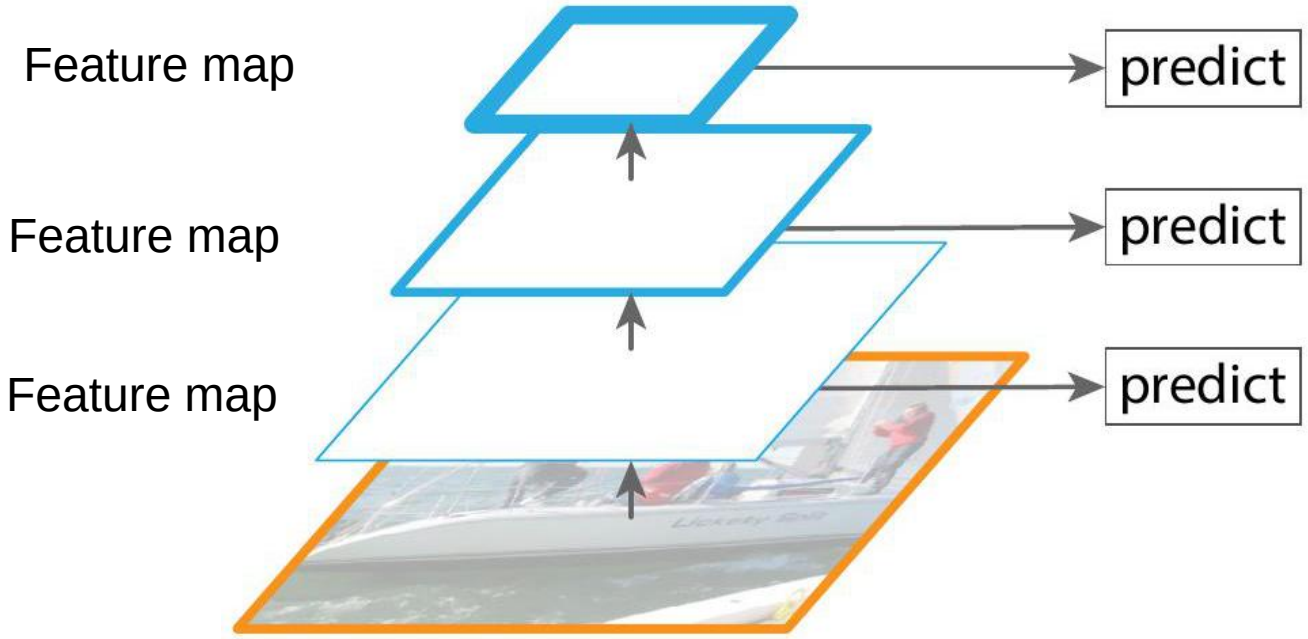
SSD: Performance

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

Table 7: **Results on Pascal VOC2007 `test`.** SSD300 is the only real-time detection method that can achieve above 70% mAP. By using a larger input image, SSD512 outperforms all methods on accuracy while maintaining a close to real-time speed.

SSD: Summary

- Important: use feature map from different layers!
- More anchor boxes are used compared to YOLO
- Anchor-based approach

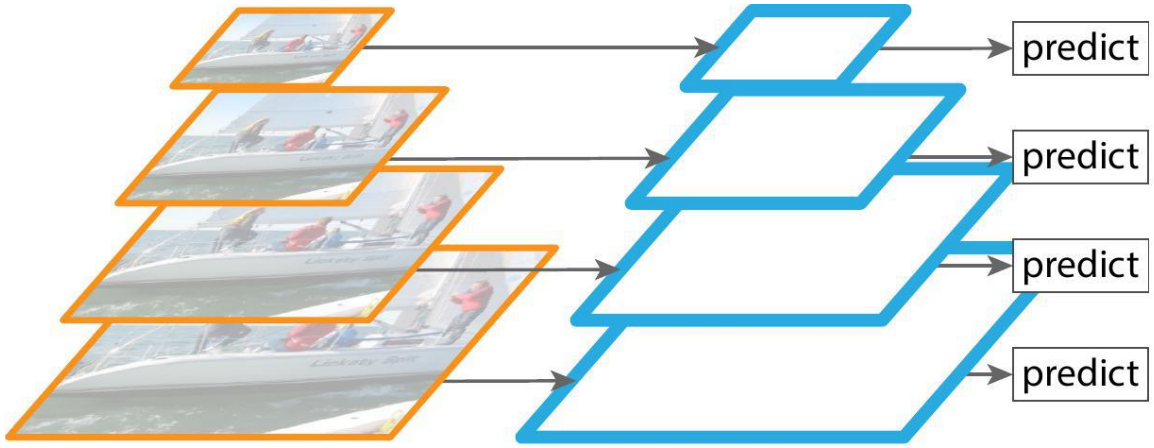


Network structure of SSD

Feature Pyramid Network (FPN)

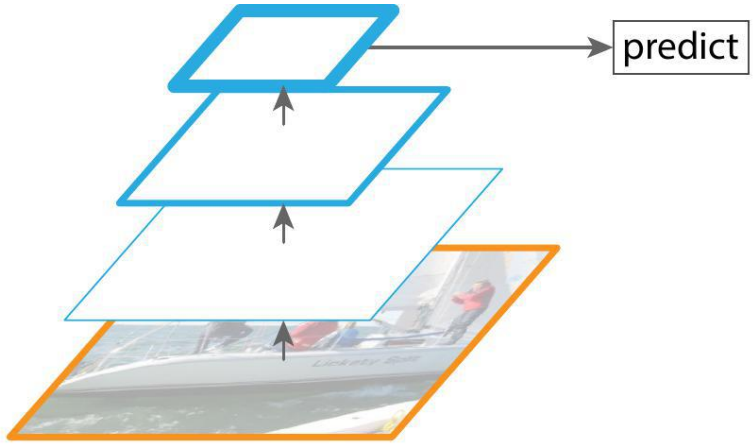
Feature Pyramid Network (FPN)

- Recognizing objects at different scale is a fundamental challenge [5]
- 4 ways to treat object in different scales:



(a) Featurized image pyramid

- Image is rescaled
- Computational slow

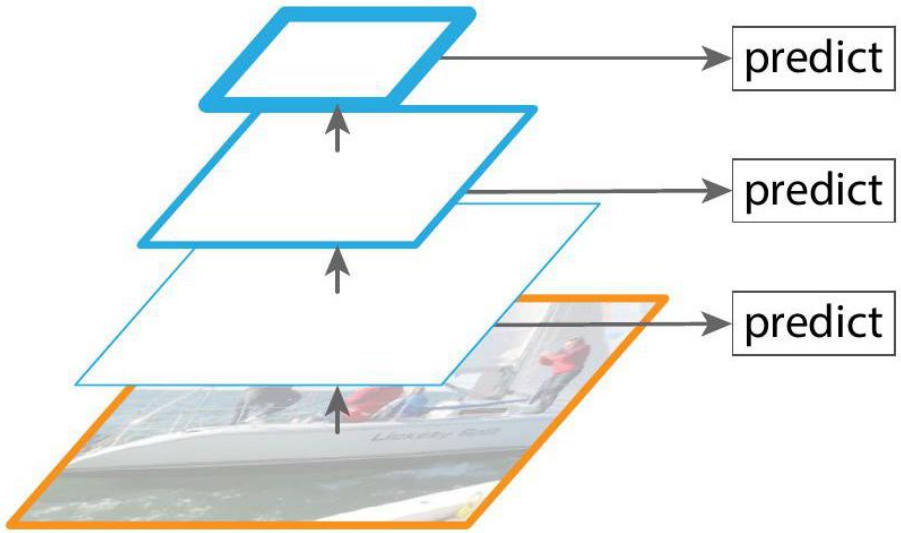


(b) Single feature map

- Classical way
- E. g. VGG16

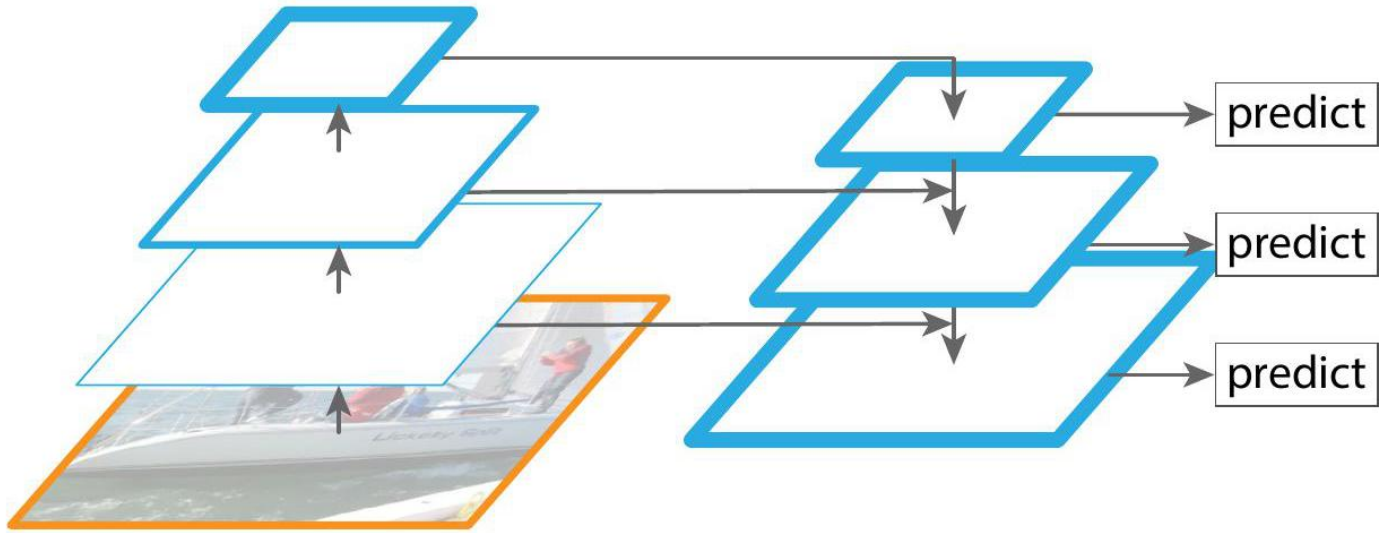
[5] Lin, et al., Feature pyramid networks for object detection, arXiv, 2017

Feature Pyramid Network (FPN)



(c) Pyramidal feature hierarchy

- Multiple feature maps are used, e. g. SSD
- It misses the opportunity to reuse higher-resolution maps of the feature hierarchy [5]



(d) Feature Pyramid Network

- Rich semantics at all levels
- Better performances for various systems [5]
- Similar to U-net

FPN: Idea and structure

- Idea:
  - Features at higher levels contribute th predictions using features at lower levels by:
    - Bottom-up pathway
    - Top-down pathway
    - Lateral pathway
  - A general idea to treat objects in different scales

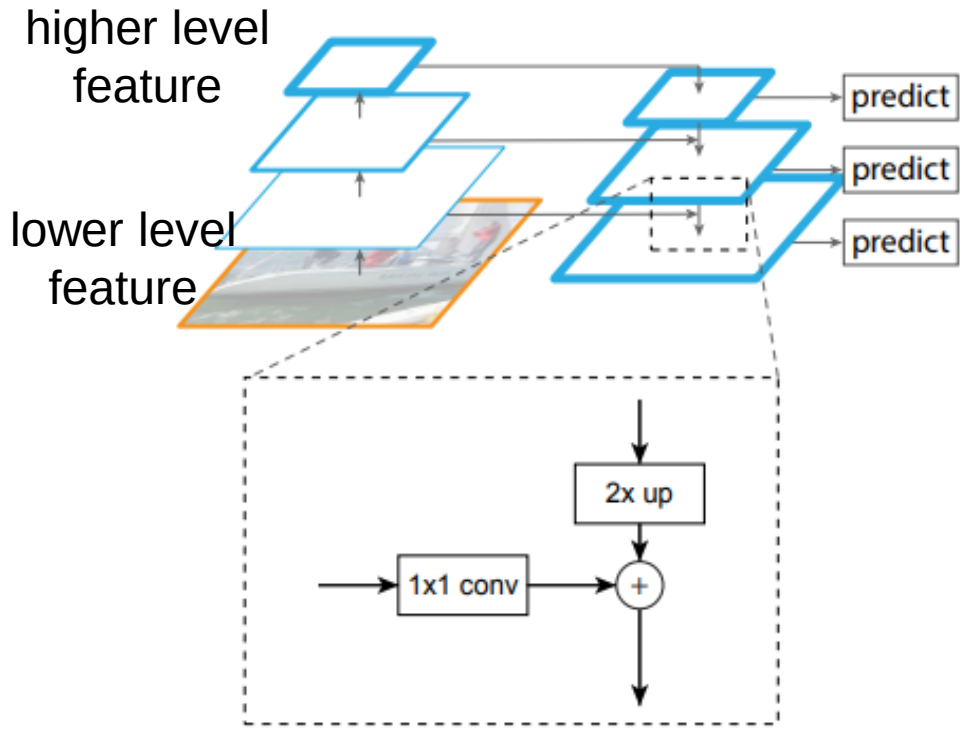


Figure 3. A building block illustrating the lateral connection and the top-down pathway, merged by addition.

FPN: Application [5]

- FPN can be combined with RPN (sliding window), Fast R-CNN, Faster R-CNN
- It is used in later publications, e. g. YOLO V6 [6]

RPN	feature	# anchors	lateral?	top-down?	AR^{100}	AR^{1k}	AR^{1k}_s	AR^{1k}_m	AR^{1k}_l
(a) baseline on conv4	C_4	47k			36.1	48.3	32.0	58.7	62.2
(b) baseline on conv5	C_5	12k			36.3	44.9	25.3	55.5	64.2
(c) **FPN**	$\{P_k\}$	200k	✓	✓	**44.0**	**56.3**	**44.9**	**63.4**	66.2
Ablation experiments follow:									
(d) bottom-up pyramid	$\{P_k\}$	200k	✓		37.4	49.5	30.5	59.9	**68.0**
(e) top-down pyramid, w/o lateral	$\{P_k\}$	200k		✓	34.5	46.1	26.5	57.4	64.7
(f) only finest level	P_2	750k	✓	✓	38.4	51.3	35.1	59.7	67.6

[6] Li, et al., YOLO v6: a single-stage object dection framework for industrial applications, arXiv, 2022

FPN: Application [5]

Fast R-CNN	proposals	feature	head	lateral?	top-down?	AP@0.5	AP	AP_s	AP_m	AP_l
(a) baseline on conv4	RPN, $\{P_k\}$	C_4	conv5			54.7	31.9	15.7	36.5	45.5
(b) baseline on conv5	RPN, $\{P_k\}$	C_5	*2fc*			52.9	28.8	11.9	32.4	43.4
(c) **FPN**	RPN, $\{P_k\}$	$\{P_k\}$	*2fc*	✓	✓	**56.9**	**33.9**	**17.8**	**37.7**	**45.8**
Ablation experiments follow:										
(d) bottom-up pyramid	RPN, $\{P_k\}$	$\{P_k\}$	*2fc*	✓		44.9	24.9	10.9	24.4	38.5
(e) top-down pyramid, w/o lateral	RPN, $\{P_k\}$	$\{P_k\}$	*2fc*		✓	54.0	31.3	13.3	35.2	45.3
(f) only finest level	RPN, $\{P_k\}$	P_2	*2fc*	✓	✓	56.3	33.4	17.3	37.3	45.6

Faster R-CNN	proposals	feature	head	lateral?	top-down?	AP@0.5	AP	AP_s	AP_m	AP_l
(*) baseline from He *et al.* [16]†	RPN, C_4	C_4	conv5			47.3	26.3	-	-	-
(a) baseline on conv4	RPN, C_4	C_4	conv5			53.1	31.6	13.2	35.6	**47.1**
(b) baseline on conv5	RPN, C_5	C_5	*2fc*			51.7	28.0	9.6	31.9	43.1
(c) **FPN**	RPN, $\{P_k\}$	$\{P_k\}$	*2fc*	✓	✓	**56.9**	**33.9**	**17.8**	**37.7**	45.8

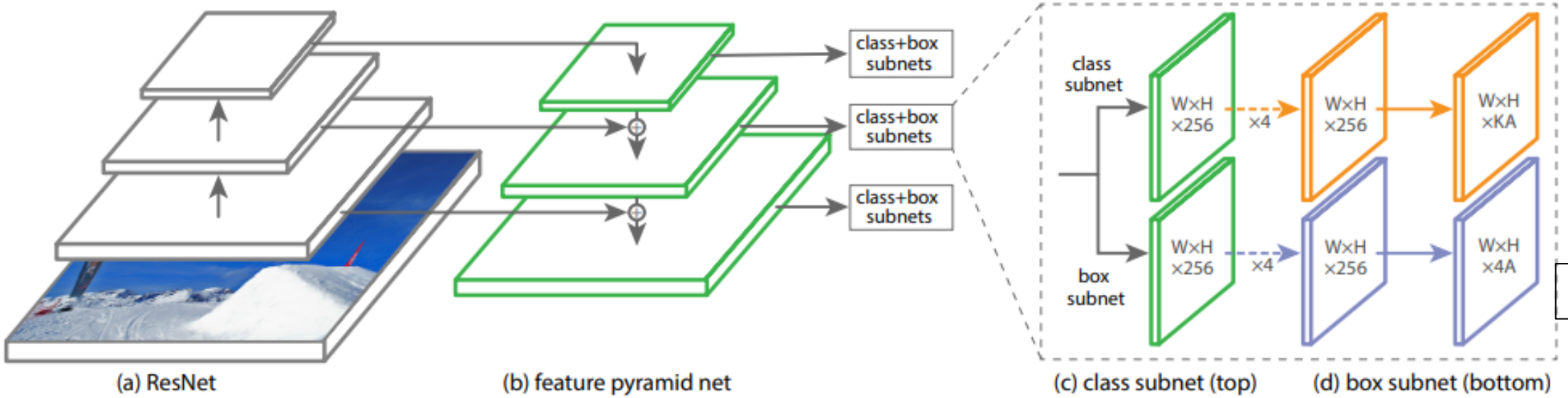
FPN improves the peformance of RPN, Fast R-CNN, Faster R-CNN

Retina Network

Retina Network [7]: Structure

backbone

2 heads / feature map



Why „4“?

- Anchor-based approach
- Ideas from FPN and SSD are applied again!

A: #anchors, A = 9 in [7]
K: #classes

[7] Lin, et al., Focal loss for dense object detection, arXiv, 2017

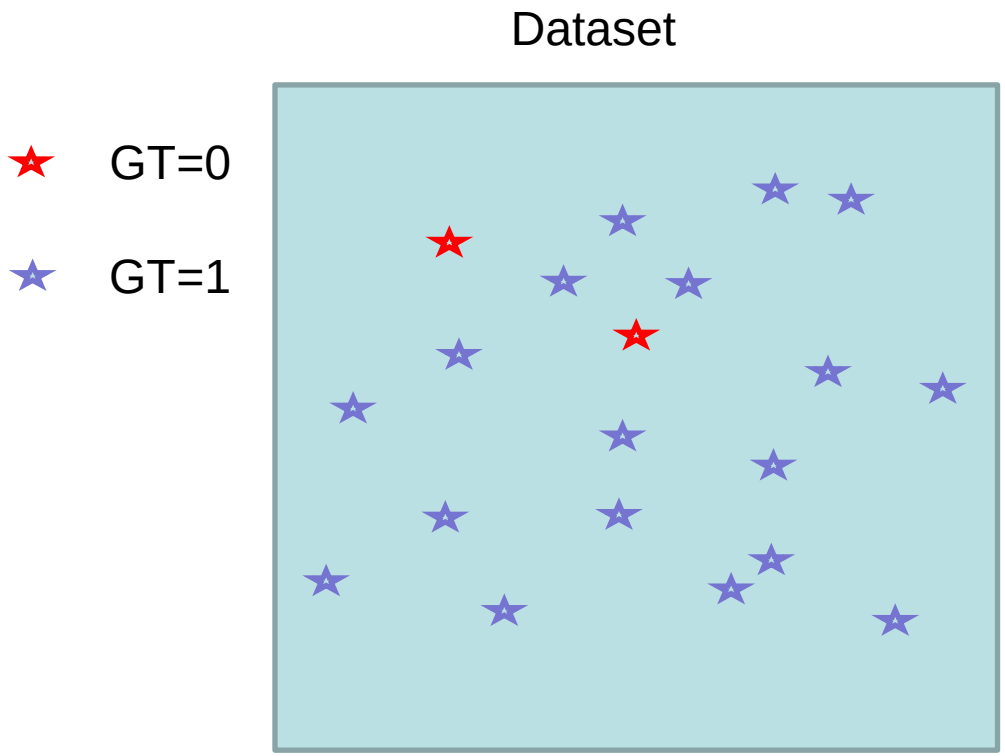
Retina Network: Focal loss

- In [7], a new loss function „focal loss“ is introducted
- Motivation: Solve **the problem of class imbalance**
- Focal loss aims to reshape the loss function to down-weight easy examples and focus training on hard examples, whose prediction is wrong
- Focal loss increases the importance of correcting misclassified examples

[7] Lin, et al., Focal loss for dense object detection, arXiv, 2017

Class Imbalance Problem

- A problem where the total number of a class of data is far less than the total number of another class of data



$$obj(x) = \sum_{i=1}^{N} l(x_i) = \underbrace{\sum_{i=red} l(x_i)}_{\text{2 red data points}} + \underbrace{\sum_{i=blue} l(x_i)}_{\text{N blue data points (N>>2)}}$$

- Term for red points will be overwhelmed by term for blue points
- Hard to optimize both terms simultaneously

Direct Idea: Balanced Cross Entropy

❑ Cross Entropy:

$$\mathrm{CE}(p, y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{otherwise.} \end{cases}$$

- y ∈ {0, 1}: GT of x_i
- p ∈ [0, 1]: predicted possiblity

❑ Idea:

$$\text{BalancedCE(p, y)} = \alpha \sum_{i=red} CE(p_i, y_i) + \beta \sum_{i=blue} CE(p_i, y_i)$$

- Assign different weights (α and β) to different classes
- E.g. α, β = inverse of class frequency
- Two classes can be minimized simultaneously and equally

❑ Example:

$$\text{BCE}(p, y) = \frac{1}{2} \sum_{i=red} CE(p_i, y_i) + \frac{1}{N} \sum_{i=blue} CE(p_i, y_i)$$

$$\text{i.e. } \alpha = \frac{1}{2}, \beta = \frac{1}{N}$$

Focal Loss [7]

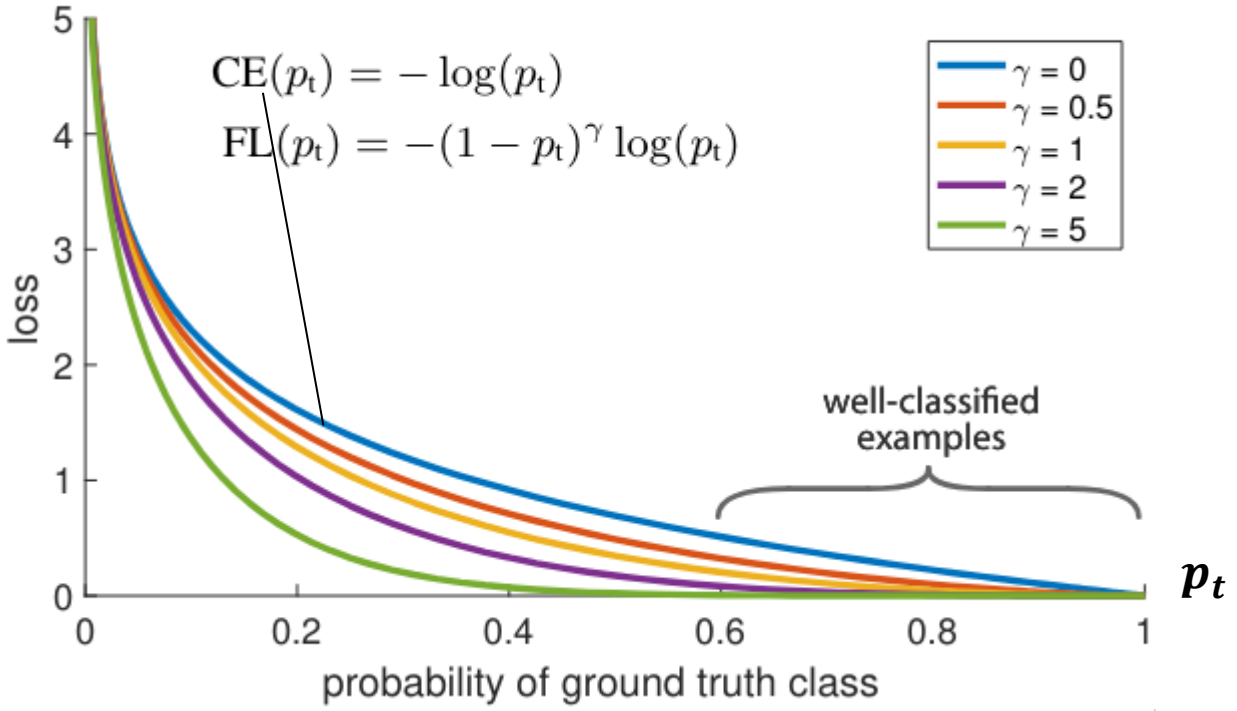
- Denote p_t as the "**Possibility of the GT class**":

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise,} \end{cases}$$

- We have: CE(p, y) = CE(p_t) = - log(p_t)
- Focal loss is introduced as:

$$\mathbf{FL}(p_t) := -(1 - p_t)^{\gamma} \log(p_t)$$

  - $\gamma > 0$ is a hyperparameter
  - If γ=0, FL = CE
- FK reduces the relative loss for well-classified examples (p_t > 0.5), putting more focus on hard, misclassified examples.



Hard examples: prediction is wrong

Easy examples: prediction is correct

Focal Loss

- In [7], α-balanced variant of the focal loss is used

$$\alpha\text{-FL}(p_t) := -\alpha_t \, (1 - p_t)^{\gamma} \log(p_t)$$

  - $\boldsymbol{\alpha_t}$: inverse frequency of classes
  - $\boldsymbol{\alpha_t}$ can be determined before training

- α-FL takes account:
  - Class imbalances (instance number of different classes)
  - Force training to focus more on hard examples (where model makes wrong predictions)
- Note: Hard examples are usually the case for **minority class**

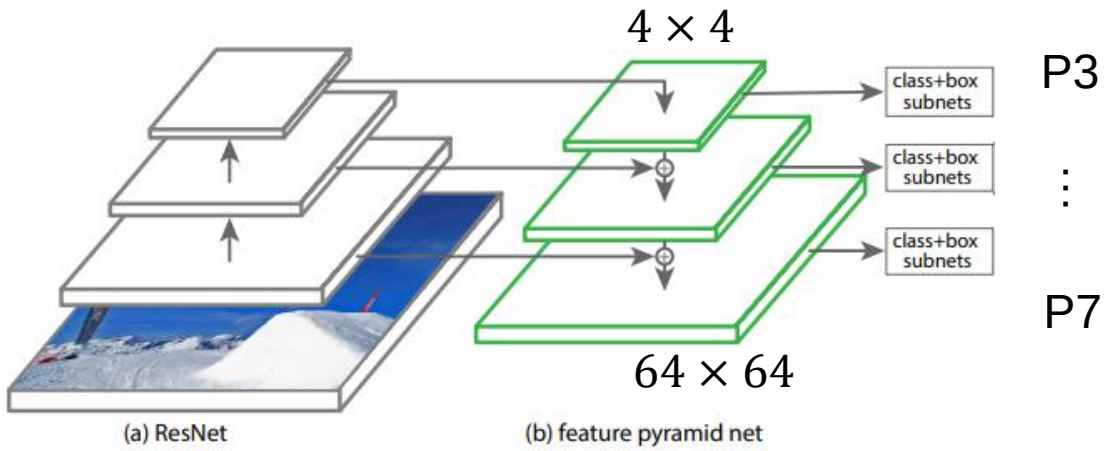
Retina Network: Why use focal loss?

- As an example:
  - # anchors = $(4^2 + 8^2 + 16^2 + 32^2 + 64^2)*9$ = 49140
  - # GT bbox <= 20 (typically)

Feature map	#locations	#anchors at each location
P3	4^2	9
P4	8^2	9
P5	16^2	9
P6	32^2	9
P7	64^2	9

→ For each anchor, the model must predict, if it is „responsible“ for predicting a bbox (y=1), or not (y=0)

→ Class imbalances!



Retina Network: Performance [7]

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [16]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [20]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [17]	Inception-ResNet-v2 [34]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [32]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [27]	DarkNet-19 [27]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [22, 9]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [9]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet (ours)	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet (ours)	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2

Retina Network: Summary

- Anchor-based approach
- Backbone + 2 heads of each feature map
- Used idea of SSD:
  - Anchors at **higher** levels are responsible for detecting **big** objects
  - Anchors at **lower** levels are responsible for detecting **small** objects
- Used idea of FPN
  - Feature pyramid
- Proposed focal loss function for treating class imbalance problem
- Performance:
  - More accurate, but slower, than YOLO V2
  - Since Retina, two-stage methods get less favored

Anchor free method: FCOS

Anchor-free Methods: Motivation

Anchor-based methods:

- Retina, SSD, YOLO V3, Faster R-CNN
- Complicated computation, e. g. calculate IoU during training
- Many hyperparameters (anchor boxes, IoU threshold, …)
- For higher recall rate, lots of anchors are needed, e. g. 180K anchor boxes für FPN [32]
- Most of anchor boxes are labeled as negative → Class Imbalance
- Need NMS (post-processing)

[32] Tian, et al., FCOS: Fully Convolutional One-Stage Object Detection, arXiv, 2019

Anchor-free Methods: Networks

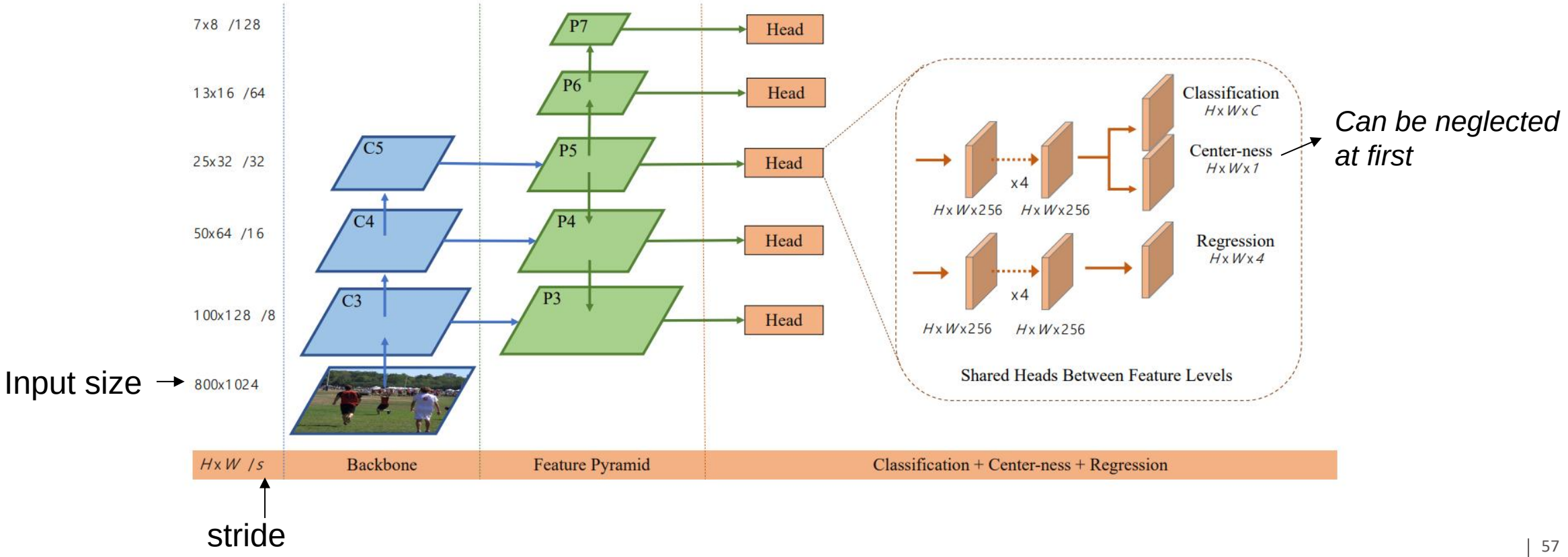
- YOLO V1 (Grid cell can be viewed as anchor!)
- **FCOS [32]**
- YOLOX
- YOLO V6
- …

The basic of anchor-based methods is to predict the coordinates **with respect to anchor boxes**. If there is no anchor boxes, what should be predicted?

[32] Tian, et al., FCOS: Fully Convolutional One-Stage Object Detection, arXiv, 2019

Anchor-free Method: FCOS [32]

- **Idea**: Solve object detection problem in a „**pixel-wise**“ prediction fassion, i.e. analogue to FCN for semantic segmentation



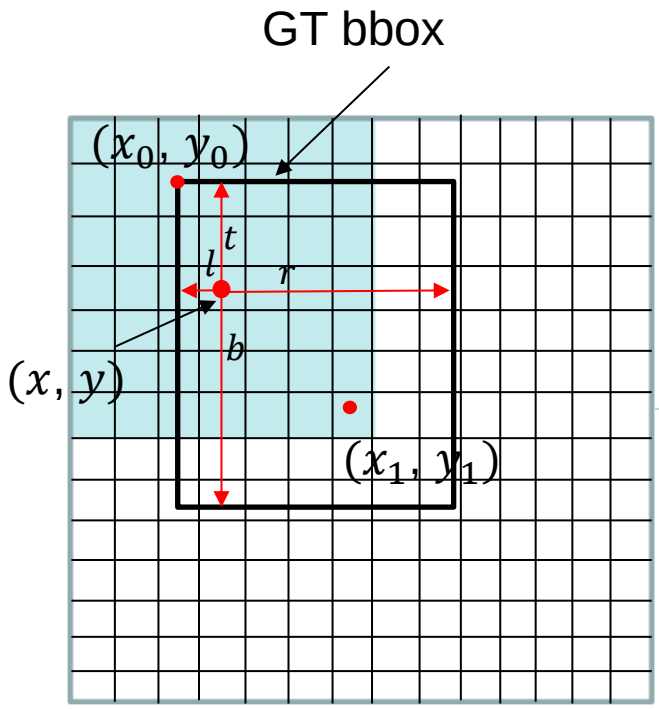
FCOS[32]: Regression Target

$$l = x - x_0$$

$$t = y - y_0$$

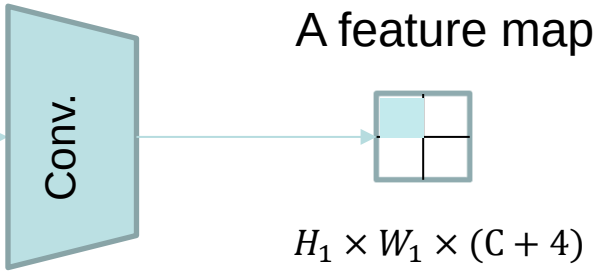
$$r = x_1 - x_0$$

$$b = y_1 - y_0$$



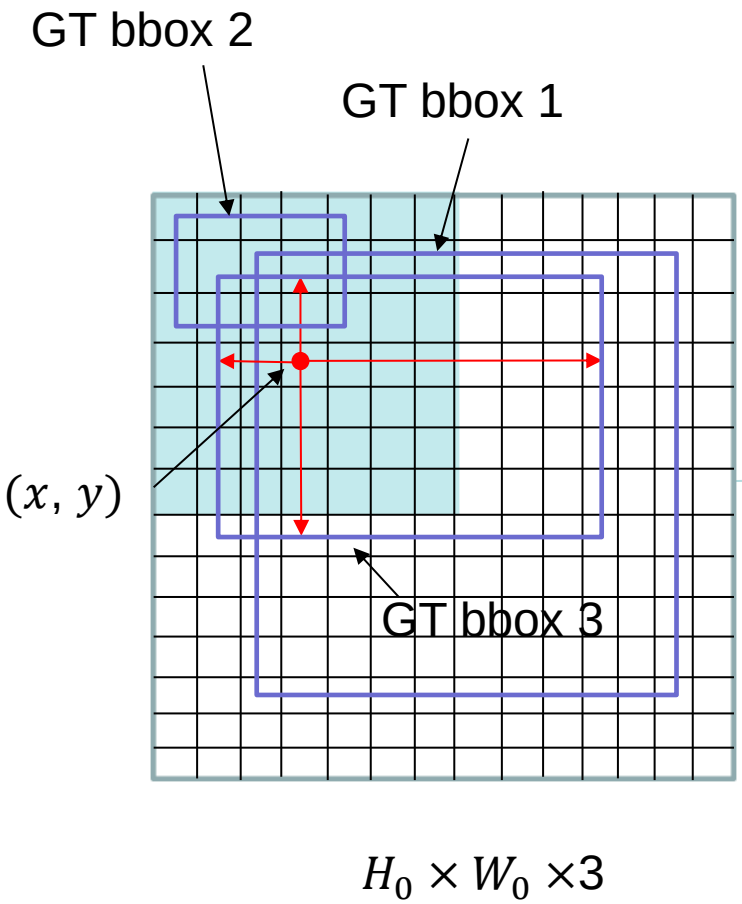
$H_0 \times W_0 \times 3$

Location (x,y) is considered as **positive**, if it falls into any GT bboxes.
Location (x, y) is also called „**anchor point**“

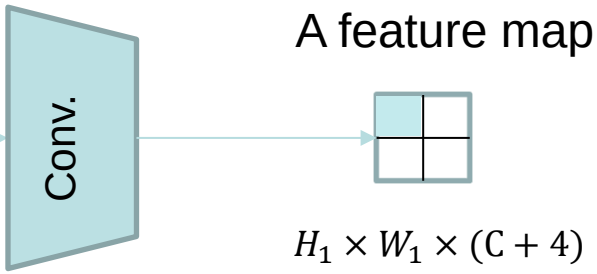


$$\Rightarrow (c_{dog}, c_{bk}, t, l, r, b) = (1.0, 0.0, 2,1,5,5)$$

FCOS [32]: Regression Target



If a location (x, y) falls into multiple bounding boxes (ambiguous sample), choose the bbox with minimal area!

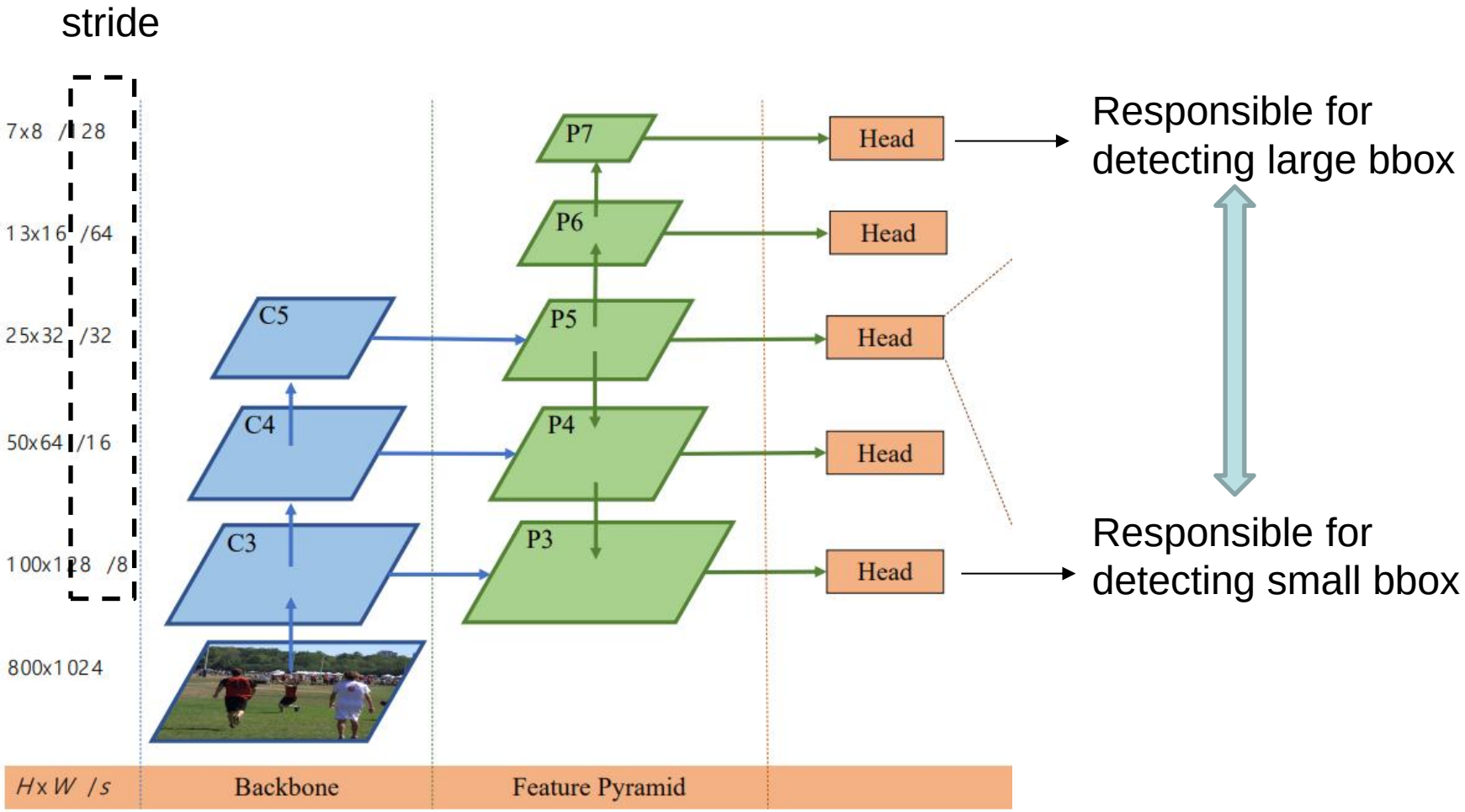


- *bbox1 and bbox3 contains location (x,y)*
- *choose bbox 3*

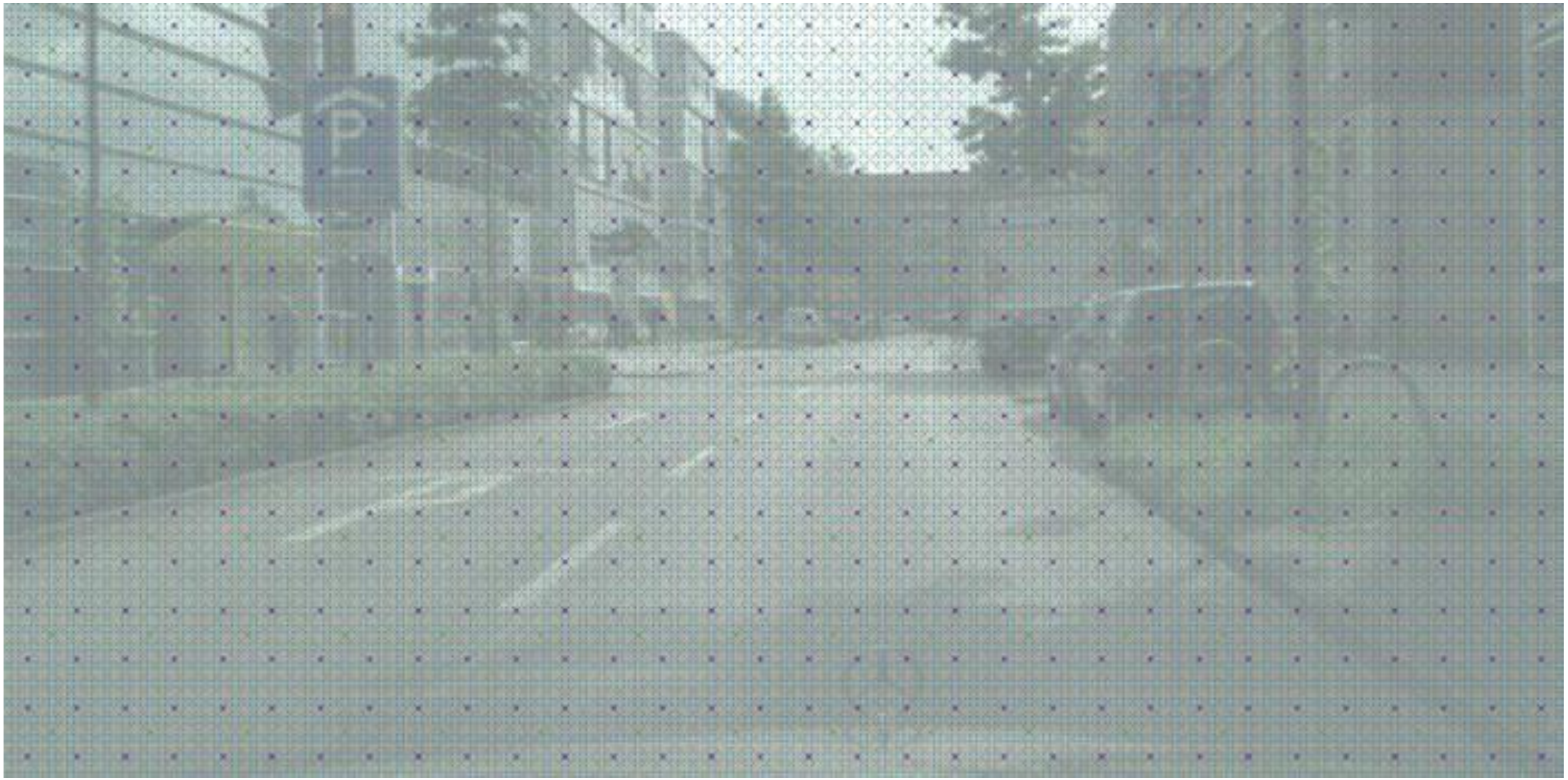
$\Rightarrow (c_{dog}, c_{bk}, t, l, r, b) = (1,0,2,2,8,4)$

- Note: Bbox 2 is too small to be encoded!
  → Because of large stride (s=7)!

FCOS [32]: Feature Pyramid



FCOS: Overlap of anchor points in feature pyramid



Source: https://medium.com/swlh/fcos-walkthrough-the-fully-convolutional-approach-to-object-detection-777f614268c

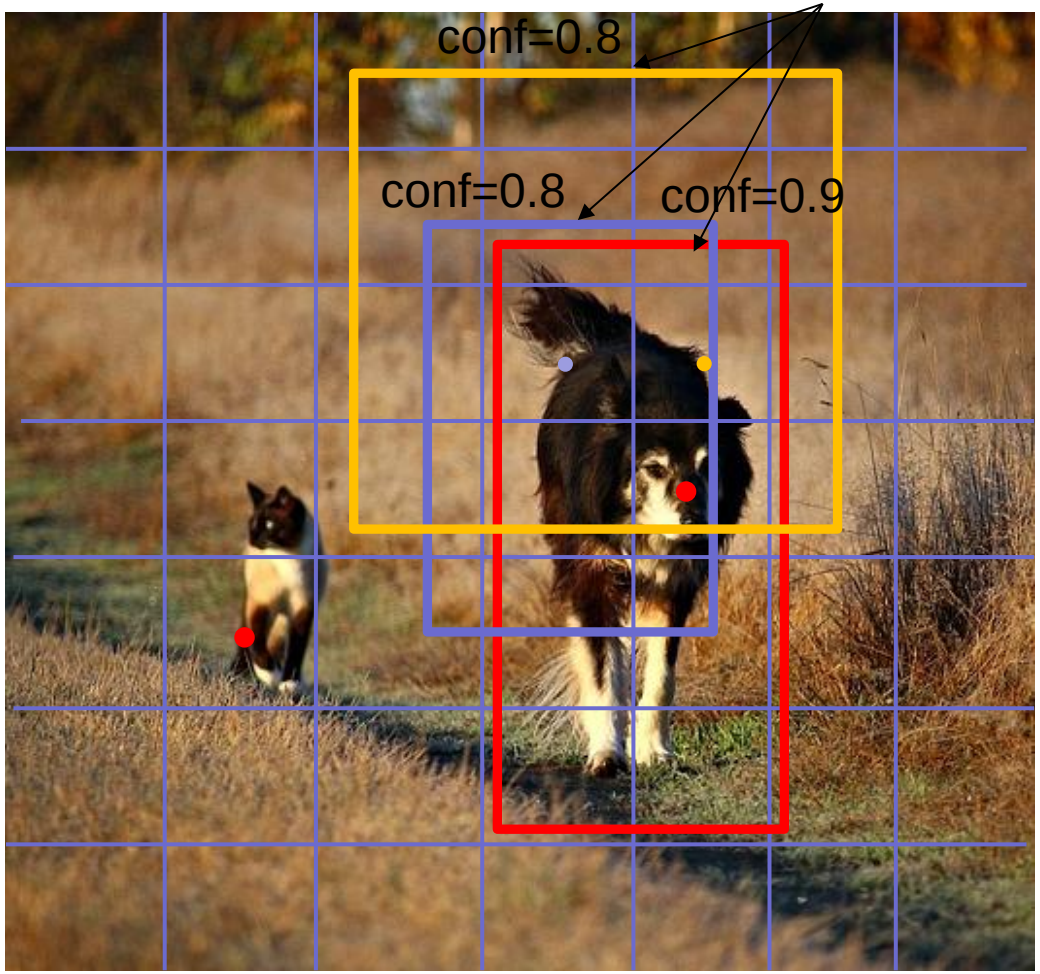
FCOS [32]: Centerness Head

Motivation:

- „A lot of low-quality predicted bbox are produced by locations far away from the center of an object“
- Try to depress these low-quality bbox

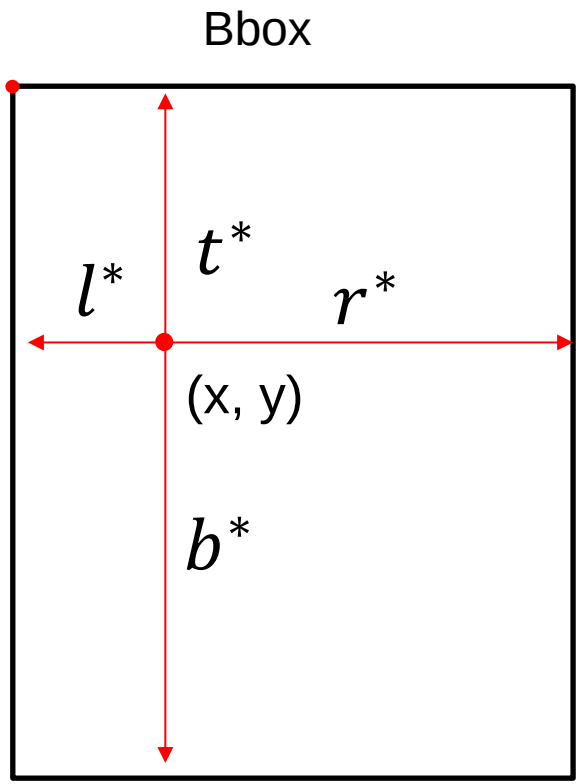
How?

- Centerness Head



- 3 bbox for 3 positions are predicted (red, blue, green)
- They can not be easily supressed by NMS, because IoU is small

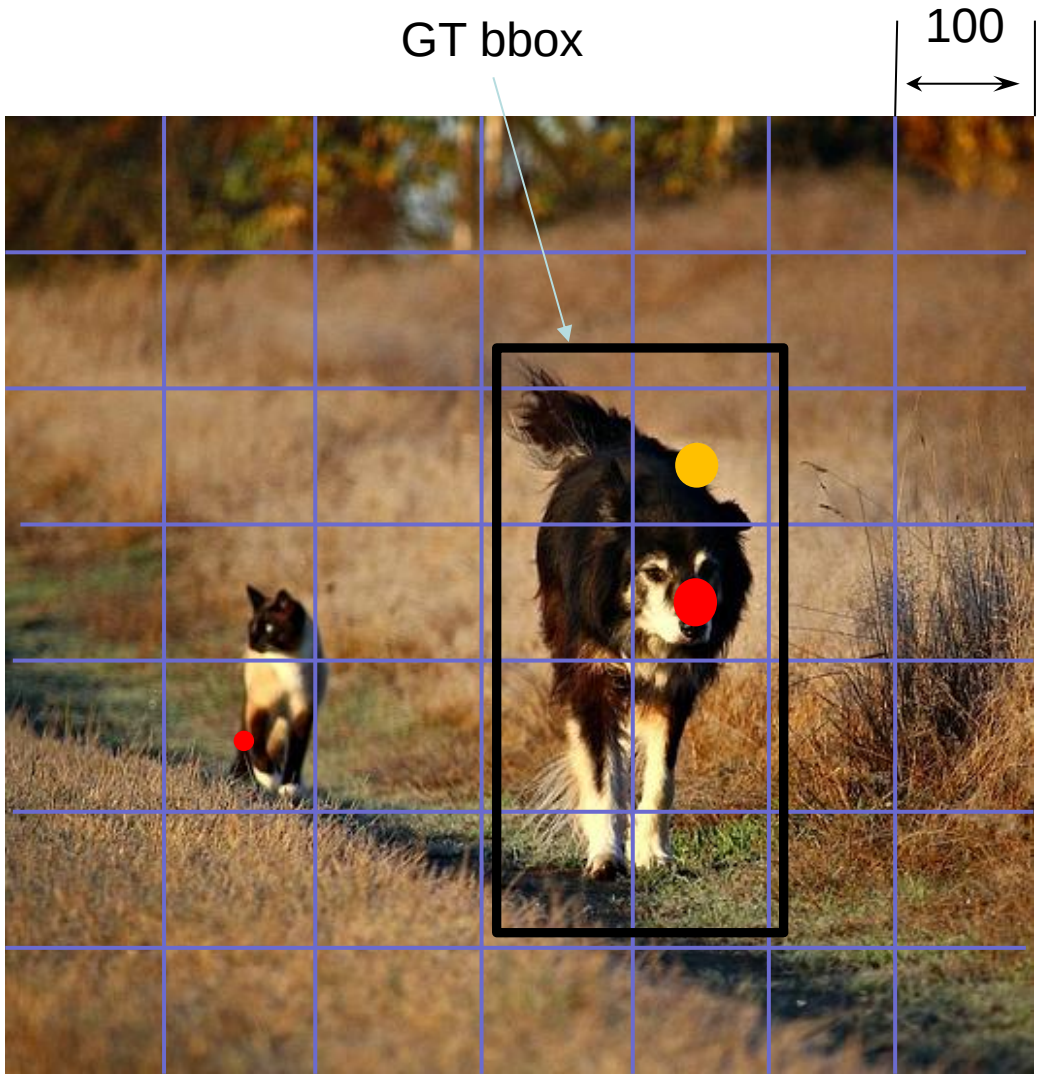
FCOS [32]: How centerness is measured?



$$\text{centerness}^* = \sqrt{\frac{\min(l^*, r^*)}{\max(l^*, r^*)} \times \frac{\min(t^*, b^*)}{\max(t^*, b^*)}}.$$

- A number ∈ [0, 1]
- If (x, y) is exactly in the middel, centerness = 1

FCOS [32]: Centerness Head @ Training time



Distances to GT bbox

$$\text{centerness}^* = \sqrt{\frac{\min(l^*, r^*)}{\max(l^*, r^*)} \times \frac{\min(t^*, b^*)}{\max(t^*, b^*)}}.$$

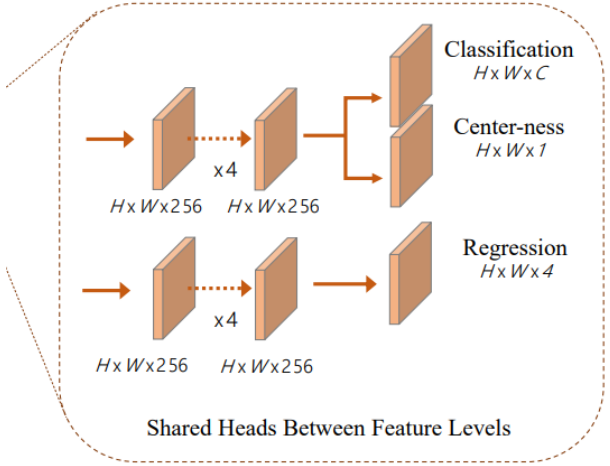
At training time:

Red location (GT):

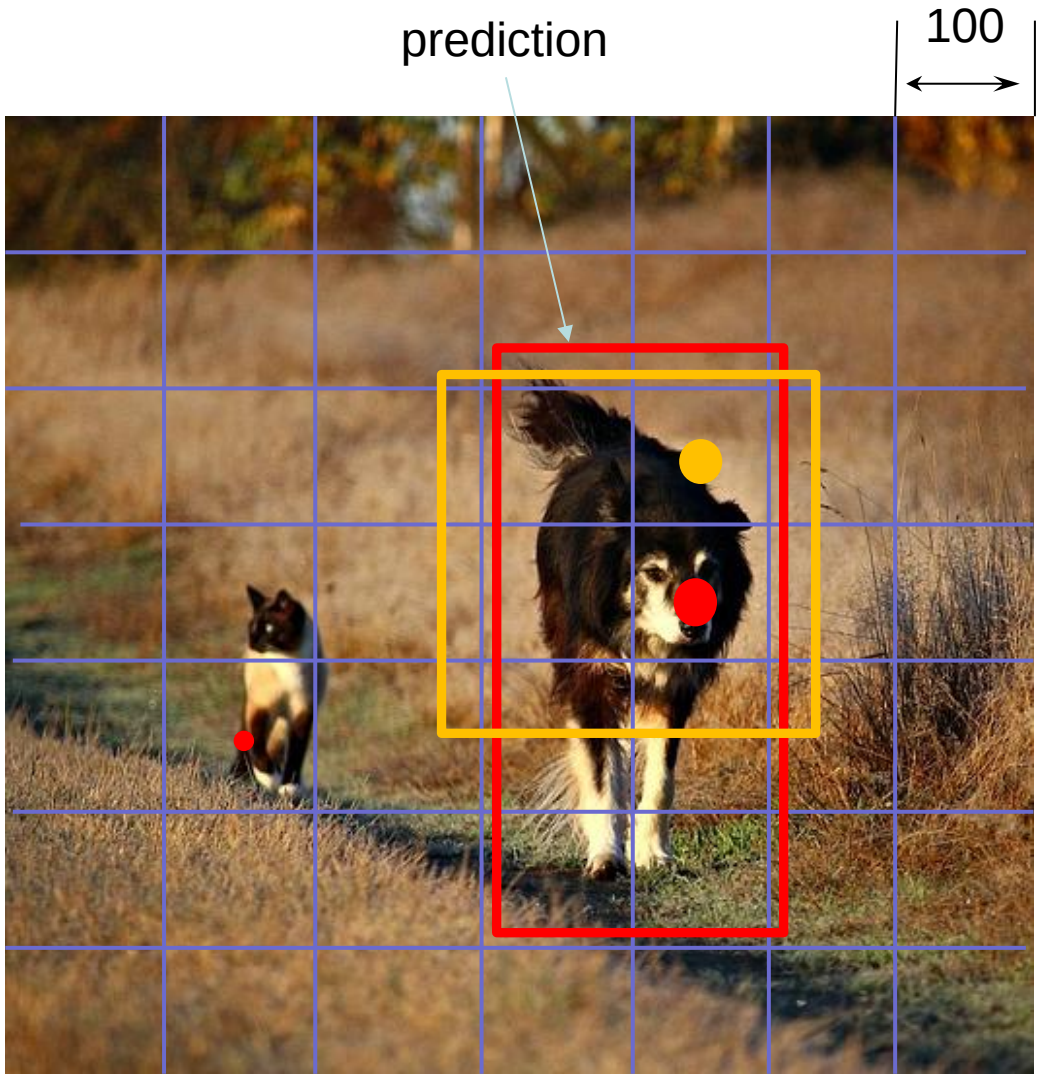
- Possiblity of dog = 1.0
- Centerness = $\sqrt{\frac{80}{140}\frac{190}{220}} = 0.70$

Green location (GT):

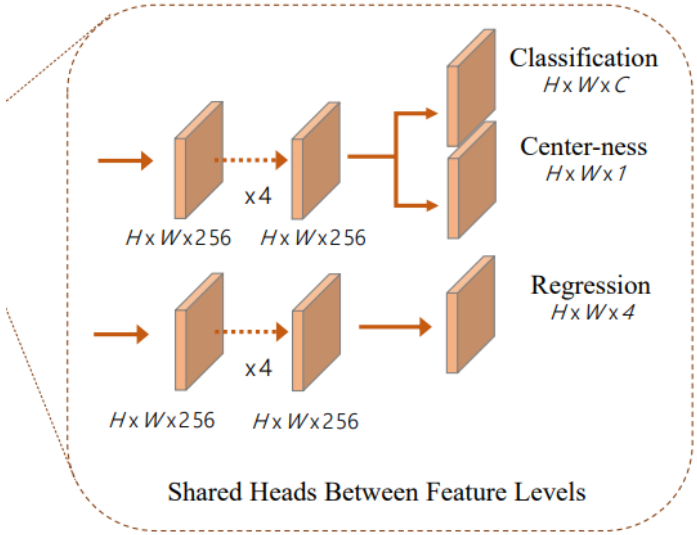
- Possiblity of dog = 1.0
- Centerness = $\sqrt{\frac{50}{120}\frac{20}{210}} = 0.20$



FCOS [32]: Centerness Head @ Prediction Time



$$\text{centerness}^* = \sqrt{\frac{\min(l^*, r^*)}{\max(l^*, r^*)} \times \frac{\min(t^*, b^*)}{\max(t^*, b^*)}}.$$



At prediction time:

Red location:

- Possibility of dog = 0.9
- Centerness = 0.69

⇒Overall score = 0.9×0.69=0.61

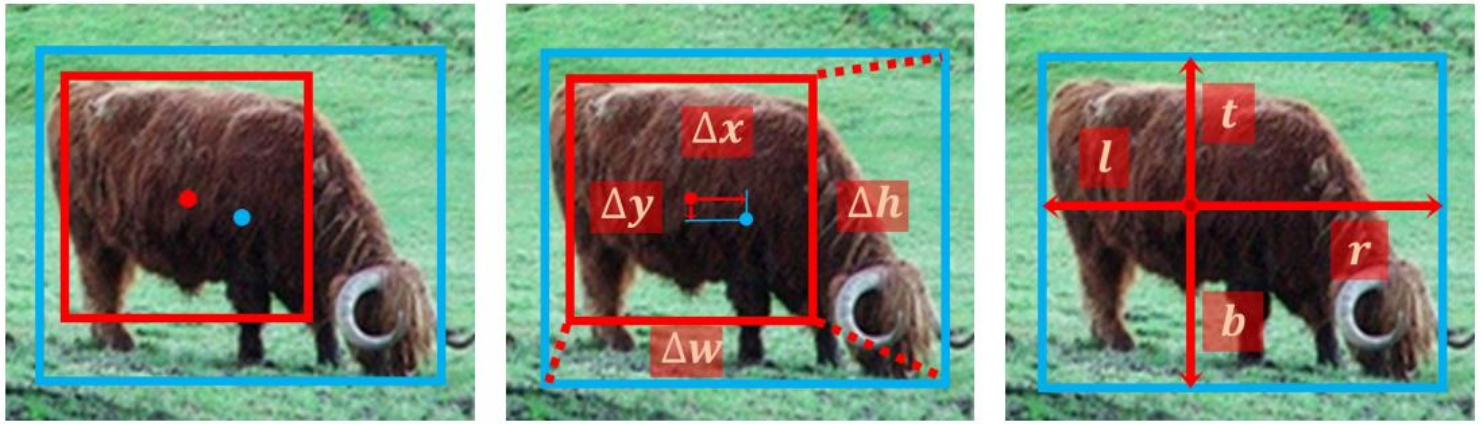
Green location:

- Possibility of dog = 0.9
- Centerness = 0.25

⇒ Overall score = 0.9×0.25=0.23

⇒ Bbox to Green location can be suppressed

FCOS compared with Retina



(a) Positive sample (b) RetinaNet (c) FCOS

Figure 2: (a) Blue point and box are the center and bound of object, red point and box are the center and bound of anchor. (b) RetinaNet regresses from anchor box with four offsets. (c) FCOS regresses from anchor point with four distances.

[36] Zhang, et al., Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection, ArXiv, 2020

FCOS [32]: Summary

- Anchor-box-free method. But still have anchor points.
- Fully convolutional network
- For each feature map, pixel-wise prediction of bbox
- Use feature pyramid to detect both small and big bboxes
- Shared heads
- Post-processing: NMS

Reference

[1] Redmon, Divvala, Girshick, Farhadi, You only look once: Unified, real-time object detection, arXiv, 2015

[2] Redmon, Farhadi, YOLO9000: better, faster, stronger, arXiv, 2016

[3] Redmon, Farhadi, YOLOv3: An Incremental Improvement, arXiv, 2018

[4] Liu, et al., SSD: single, shot multibox detector, arXiv, 2016

[5] Lin, et al., Feature pyramid networks for object detection, arXiv, 2017

[6] Li, et al., YOLOv6: a single-stage object dection framework for industrial applications, arXiv, 2022

[7] Lin, et al., Focal loss for dense object detection, arXiv, 2017

Reference

[8] Bochkovskiy, Wang, Liao, Yolov4:Optimal speed and accuracy of object detection, arXiv, 2020

[9] Liu, et al., Path aggregation network for instance segmentation, arXiv, 2018

[10] Zhong, et al., Random erasing data augmentation, arXiv, 2017

[11] DeVries and Taylor, Improved regularization of convolutional neural networks with CutOut, arXiv, 2017

[12] Singh, el al., Hide-and-Seek: A data augmentation technique for weakly-supervised localization and beyond, arXiv, 2018

[13] Chen, GridMask data augmentation, arXiv, 2020

[14] Wan, et al., Regularization of neural networks using Drop-Connect, ICML, 2013

[15] Ghiasi, et al., DropBlock: A regularization method for convolutional networks, NIPS, 2018

[16] Zhang, et al., MixUp: Beyond empirical risk minimization, arXiv, 2017.

[17] Yun, CutMix: Regularization strategy to train strong classifiers with localizable features, ICCV, 2019

[18] Geirhos, et al., ImageNet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness, ICLR, 2019.

[19] Sung, Poggio, Example-based learning for view-based human face detection, TPAMI, 1998

[20] Szegedy, et. al., Rethinking the inception architecture for computer vision, CVPR, 2016

[21] Shrivastava, et al., Training region-based object detectors with online hard example mining, CVPR, 2016

[22] Yu, et al., UnitBox: An advanced object detection network, ACM international conference on multimedia, 2016

[23] Rezatofighi, et al., Generalized intersection over union: A metric and a loss for bounding box regression, CVPR, 2019

[24] Zheng, et al., Distance-IoU loss: Faster and better learning for bounding box regression, AAAI, 2020

[25] Liu, et al., Receptive field block net for accurate and fast object detection, ECCV, 2018

[26] Hu, et al., Squeeze-and-excitation networks, CVPR, 2018

[27] Woo, et al., CBAM: Convolutional block attention module, ECCV, 2018

[28] Misra, Mish: A self regularized non-montonic neural activation function, arXiv, 2019

[29] Bodia, et al., Soft-NMS: improving object detection with one line of code, ICCV, 2017

[30] Li, et al., YOLO v6: A Single-Stage Object Detection Framework for Industrial Applications, arXiv, Sep. 2022

[31] Wang, Bochkovskiy, Liao, YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, arXiv, July 2022

[32] Tian, et al., FCOS: Fully Convolutional One-Stage Object Detection, arXiv, 2019

[33] Ge, et al., Ota:Optimal transport assignment for object detection, CVPR, 2021

[34] Ge, et al., YOLOX: Exceeding yolo series in 2021, arXiv, 2021

[35] Feng, et al., Tood: Task-aligned one-stage object detection, ICCV, 2021

[36] Zhang, et al., Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection, ArXiv, 2020

[37] Wang, Bochkovskiy, Liao, YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, arXiv, 2022