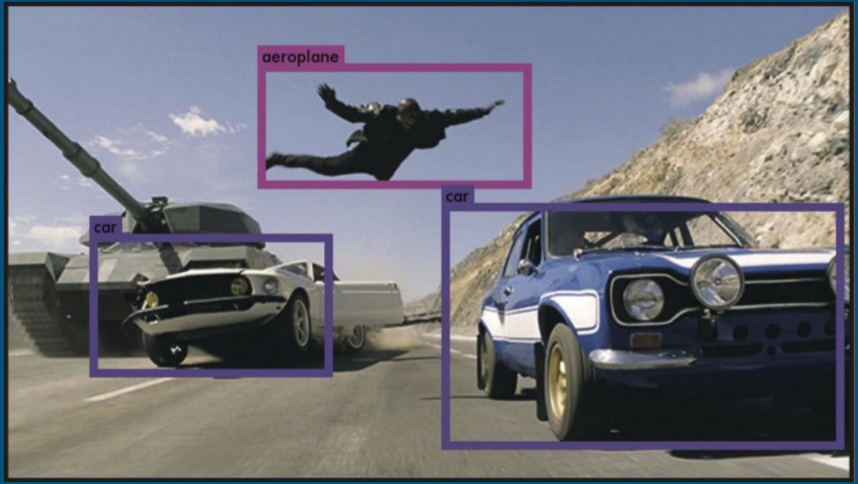
YOLO v1, v2, v3

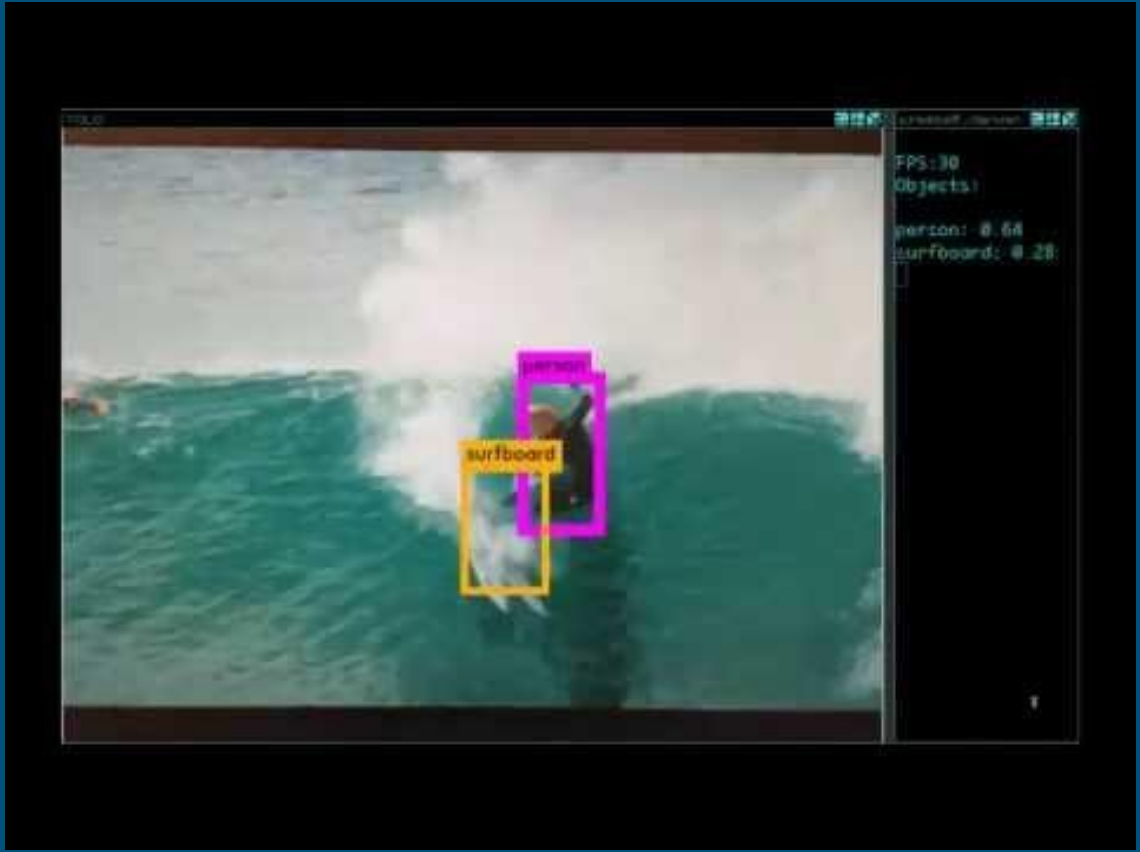
You only look once

Tørres Lande

What is special about YOLO?

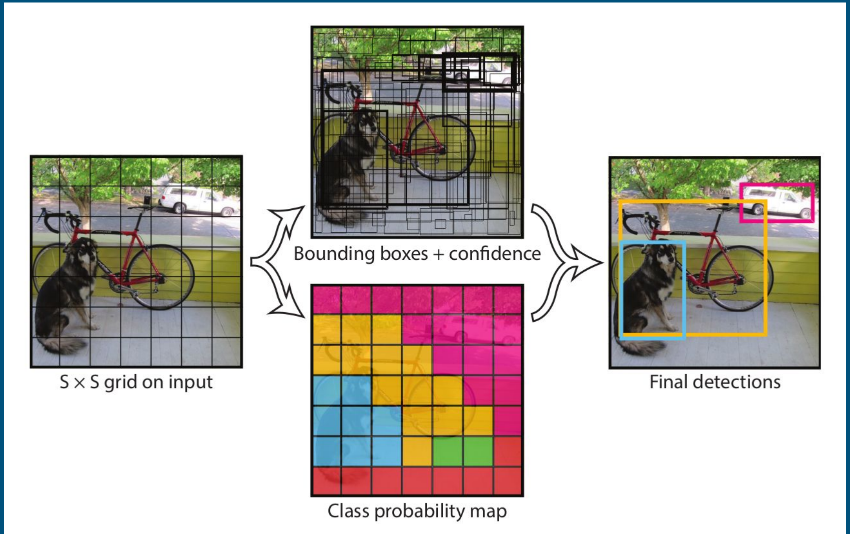
- Fast yolo can run in 155 fps, mest video is 30 fps.
- It have a global scope, so it can base it detection on context.
- YOLO is good at generalizing images.
- R-CNN have better accuracy.




FPS:30
Objects:
surfboard
person

How does YOLO work?

- Split the image into S x S grid
- Each cell have 5 parameters
- (x,y) = position
- (h,w) = height, weight
- c = confidence
- Each cell predict two bounding boxes and a confidence score
- predict conditional class probabilities p(Classi | Object)
- Output = S x S (B x 5 + num classes)

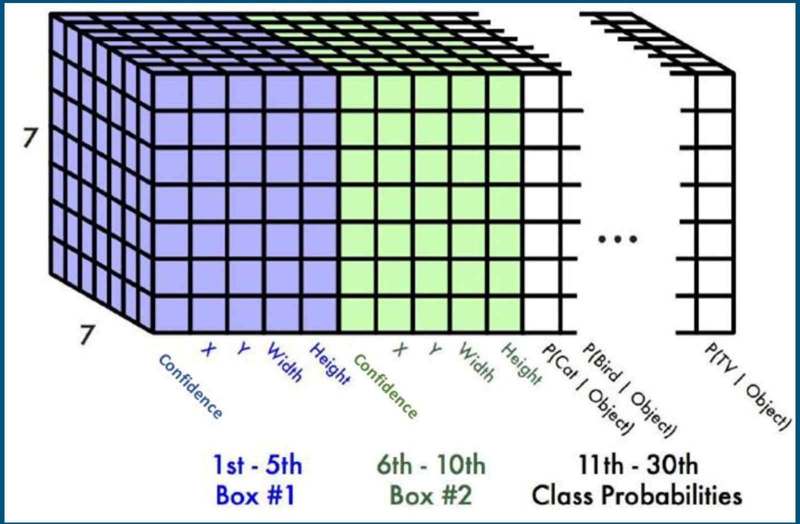


Output

- (x,y) = position
- (h,w) = height, weight
- c = confidence

Pascal VOC

- 7 x 7 grid
- 2 bounding boxes
- 20 classes
- 7 x 7 x (2 x 5 + 20) = 7 x 7 x 30 tensor = 1470 outputs



Model

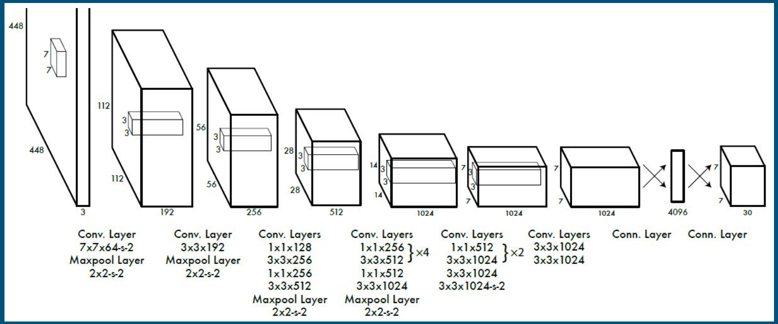
Yolo

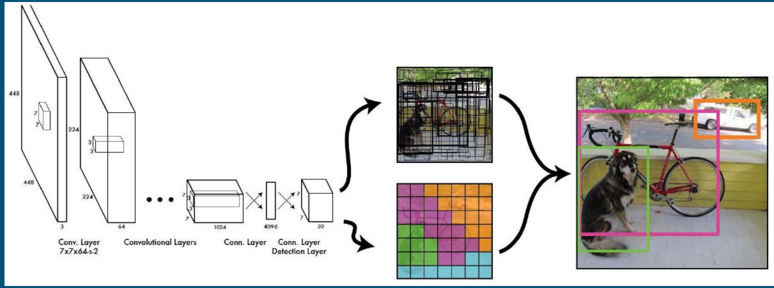
- 24 convolution layers

Fast Yolo

- 9 convolution layers

Both are inspired by googLeNet, that uses 1x1 convolution layer.



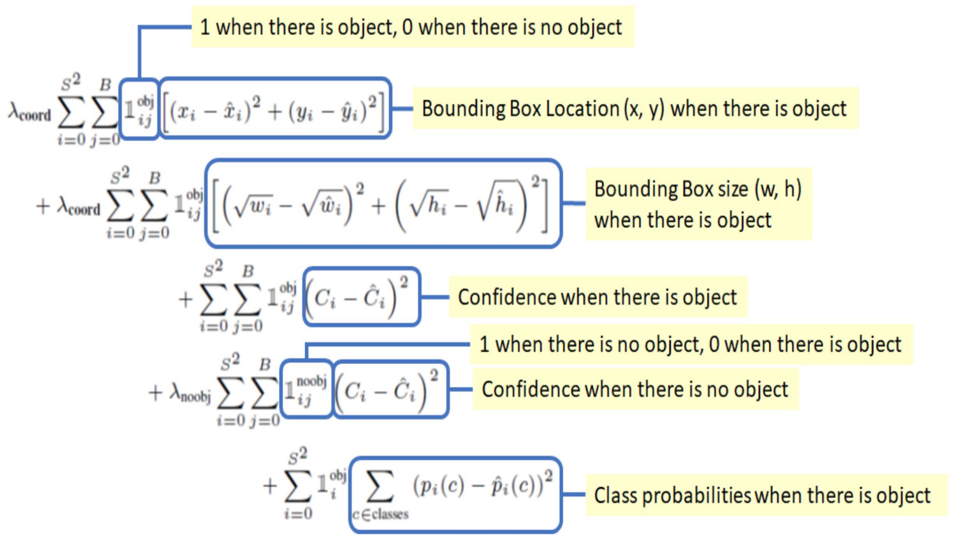


Training

- Trained for one week
- pre-trained on 224x224 for image classification, then 448x448 becuse image detection need more fine grained images.
- Leaky ReLU

Loss function

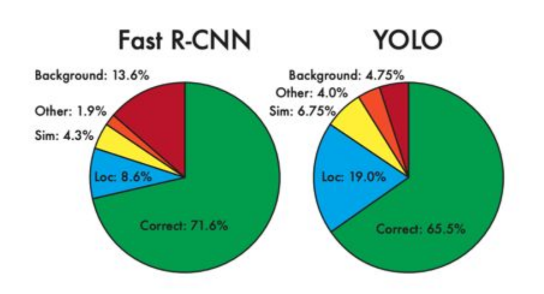
- location of the bounding boxes
- (w, h) helps the loss function punishes small errors in big boxes less than i small boxes.
- C = Confidence
- p = probabilities
- λ = weight



Loss Function

Comparison

- YOLO v1 is faster then R-CNN but not more accurate
- Yolo is more general, it outperform R-CNN on art work.
- Can combine them to get higher accuracy, since they are good at different aspects.



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

YOLO v2

YOLO used 448 x 448

YOLOv2 tested om multiple sizes

YOLOv2 480x480 > YOLO 448x448

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288×288	2007+2012	69.0	91
YOLOv2 352×352	2007+2012	73.7	81
YOLOv2 416×416	2007+2012	76.8	67
YOLOv2 480×480	2007+2012	77.8	59
YOLOv2 544×544	2007+2012	**78.6**	40

VOC 2007 for YOLOv2

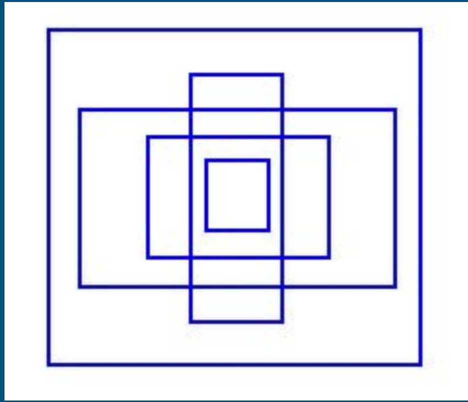
What did they do?

Batch normalization: mAP 2%

Train classifier with 224x224 and 448x448: mAP 4%

Anchor boxes: more stable training: mAP -0.3%, recall 7%

- YOLO make arbitrary guesses on boundary boxes.
- YOLOv2 create 5 anchor boxes, that focuses on shapes that most likely will fit a object
- Dimentionsion cluster to find anchor points

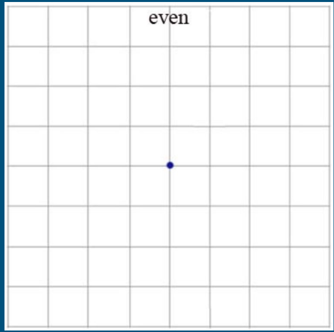


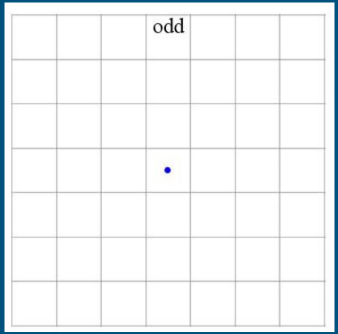
What did they do?

Odd number grid 448x448 -> 416x416: higher chance of detecting big images in the middle

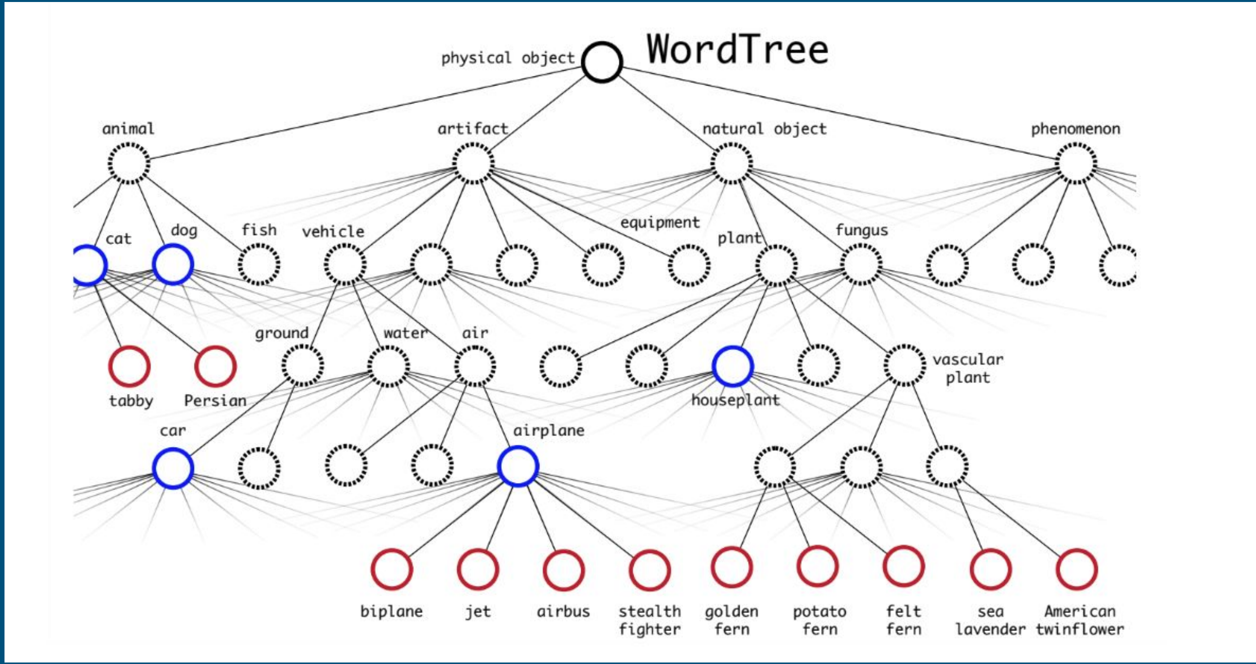
Multiscale training

- remove fully connected layer
- can use multiple image sizes





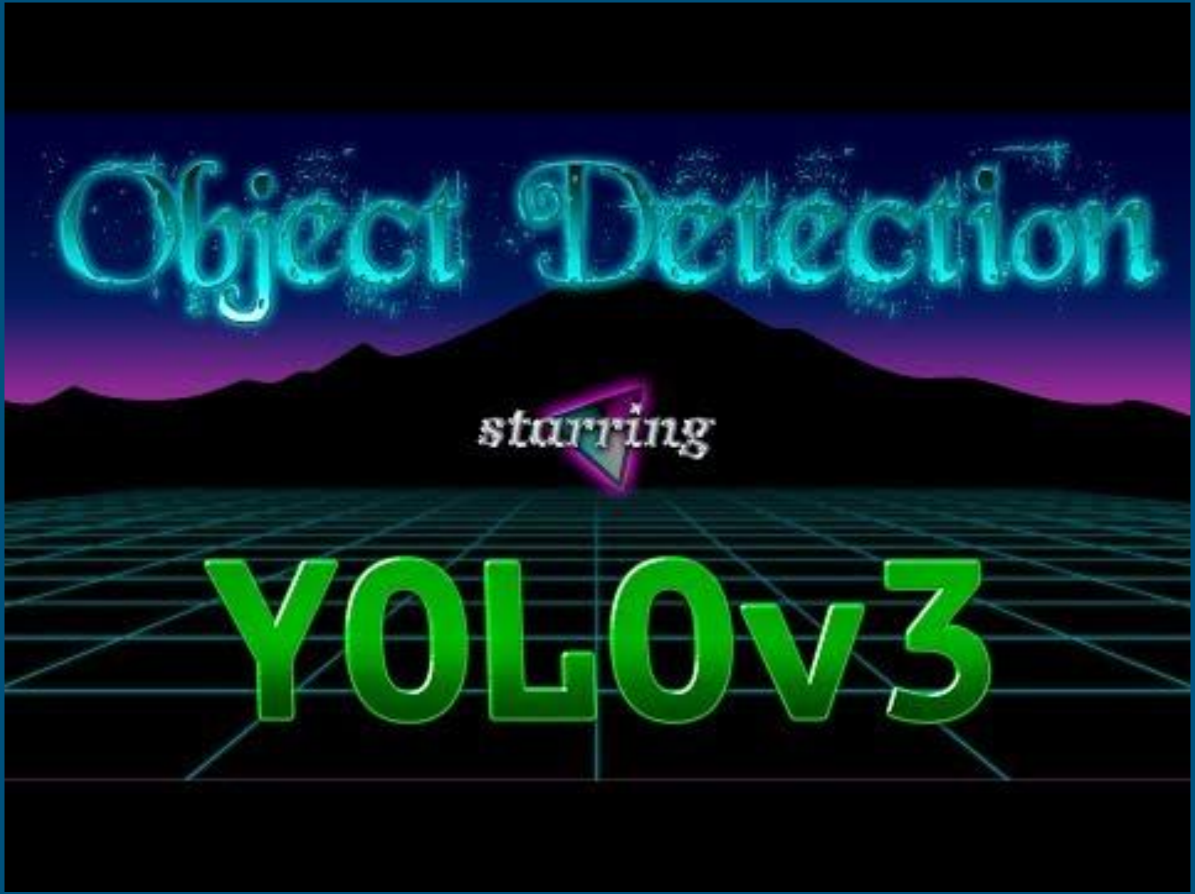
Classification



	YOLO								YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

YOLO V3

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [5]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [8]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [6]	Inception-ResNet-v2 [21]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [20]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [15]	DarkNet-19 [15]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [11, 3]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [3]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [9]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [9]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608×608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9


Object Detection
starring
YOLOv3

Sources

https://towardsdatascience.com/yolov1-you-only-look-once-object-detection-e1f3ffec8a89

https://www.youtube.com/watch?list=PLrrmP4uhN47Y-hWs7DVfCmLwUACRigYyT&v=NM6lrxy0bxs

https://arxiv.org/pdf/1506.02640.pdf

https://medium.com/adventures-with-deep-learning/yolo-v1-part-1-cfb47135f81f

https://medium.com/@divakar_239/yolo-v1-part-2-bfc686ae5560

https://medium.com/adventures-with-deep-learning/yolo-v1-part3-78f22bd97de4

https://hackernoon.com/understanding-yolo-f5a74bbc7967

https://pjreddie.com/media/files/papers/YOLOv3.pdf

https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

Questions?