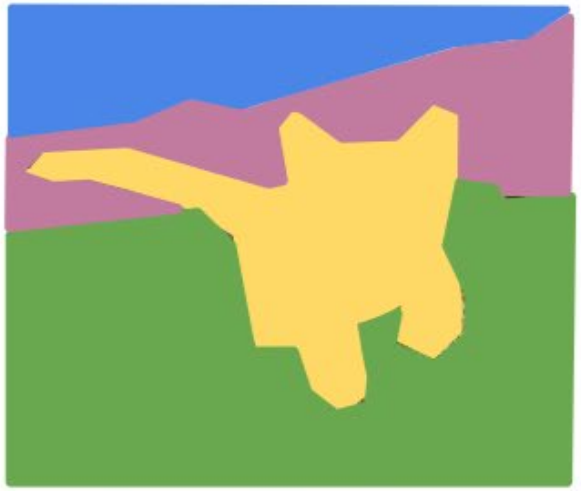
Object Detection & Image Segmentation

Prof. Kuan-Ting Lai

2022/5/28

Computer Vision Tasks

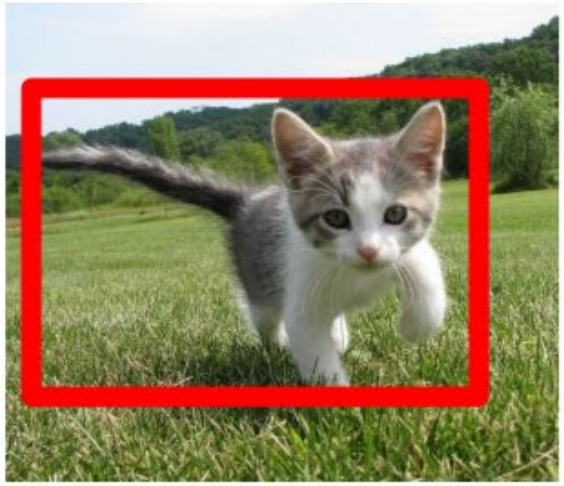
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

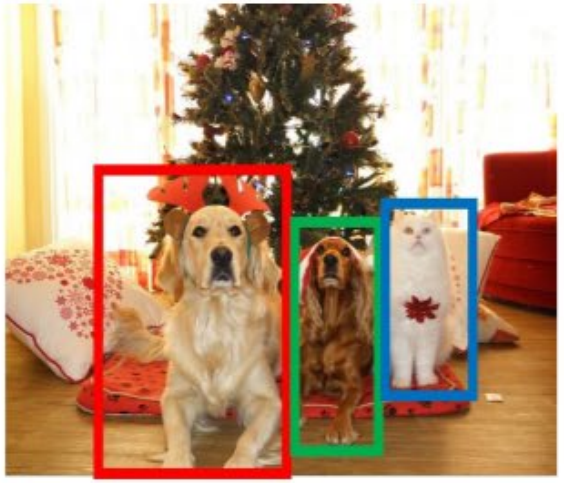
Classification + Localization



CAT

Single Object

Object Detection



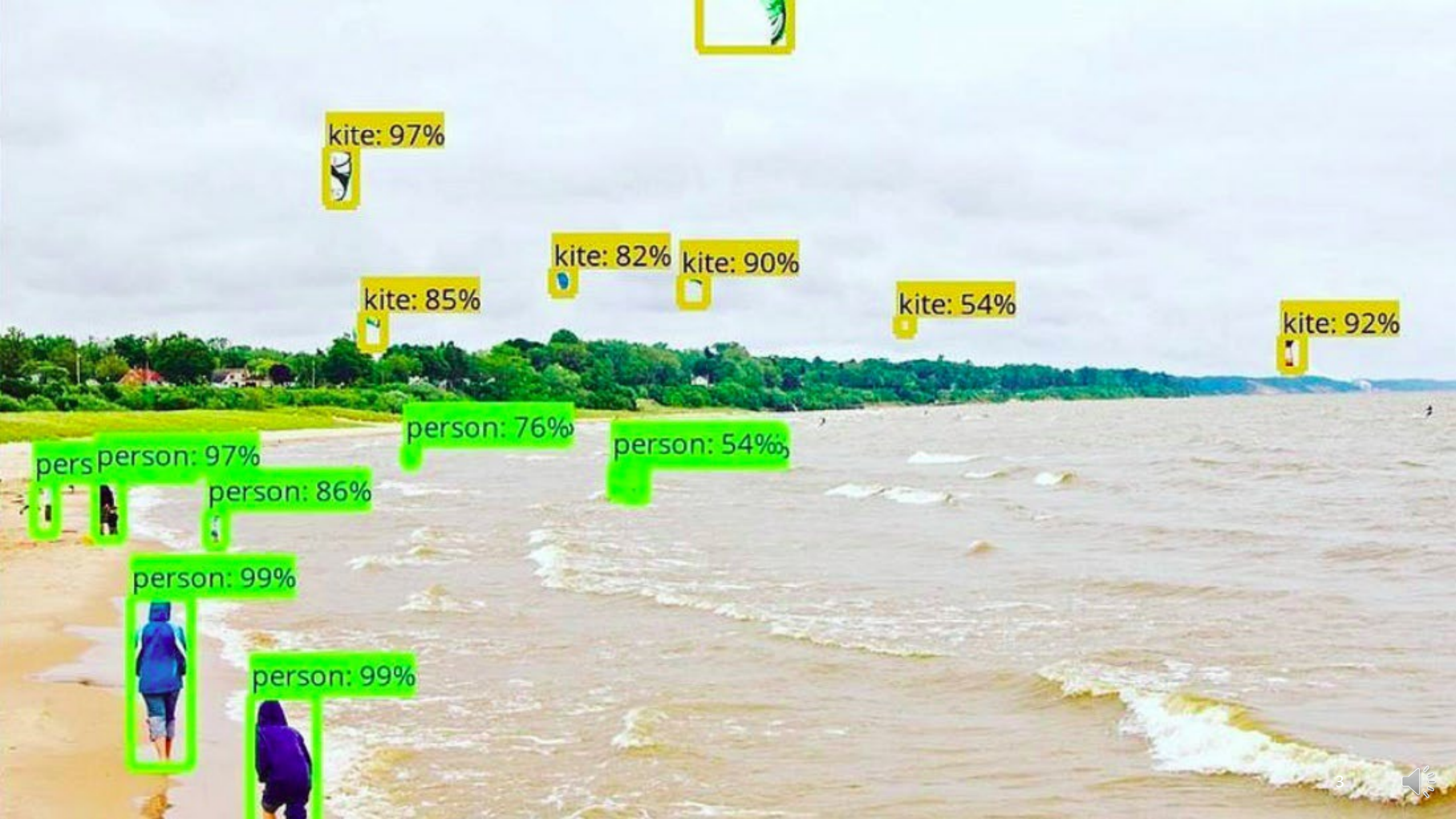
DOG, DOG, CAT

Instance Segmentation

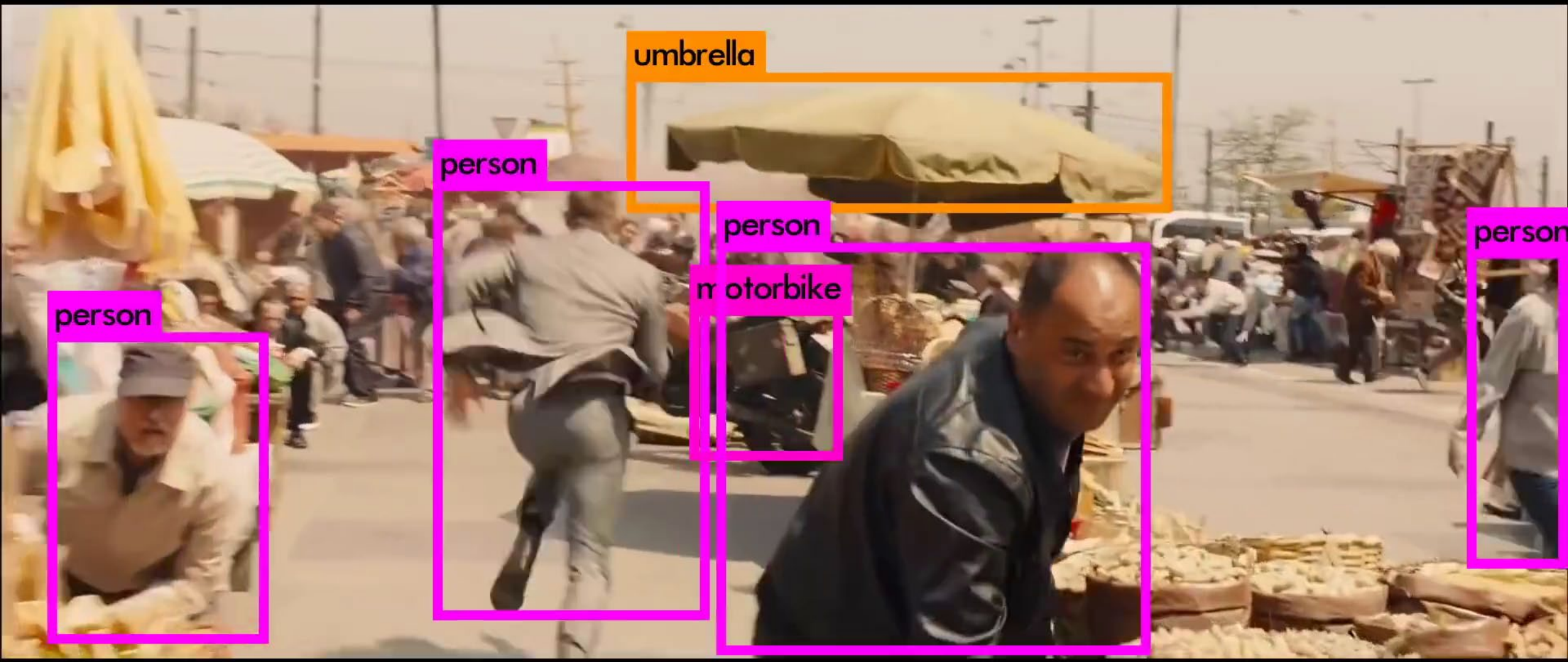
DOG, DOG, CAT

Multiple Object

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf

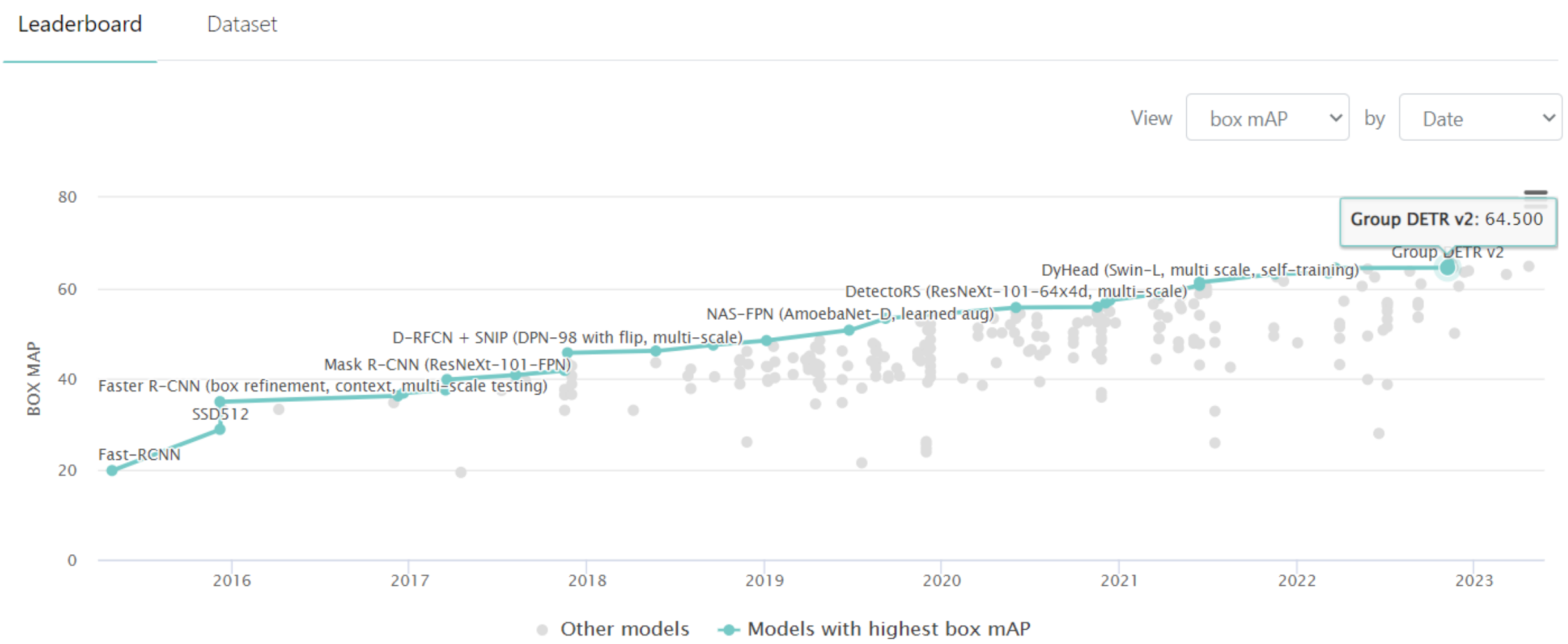

kite: 97%
kite: 82%
kite: 90%
kite: 85%
kite: 54%
kite: 92%
person: 76%
person: 54%
pers
person: 97%
person: 86%
person: 99%
person: 99%

YOLO v2



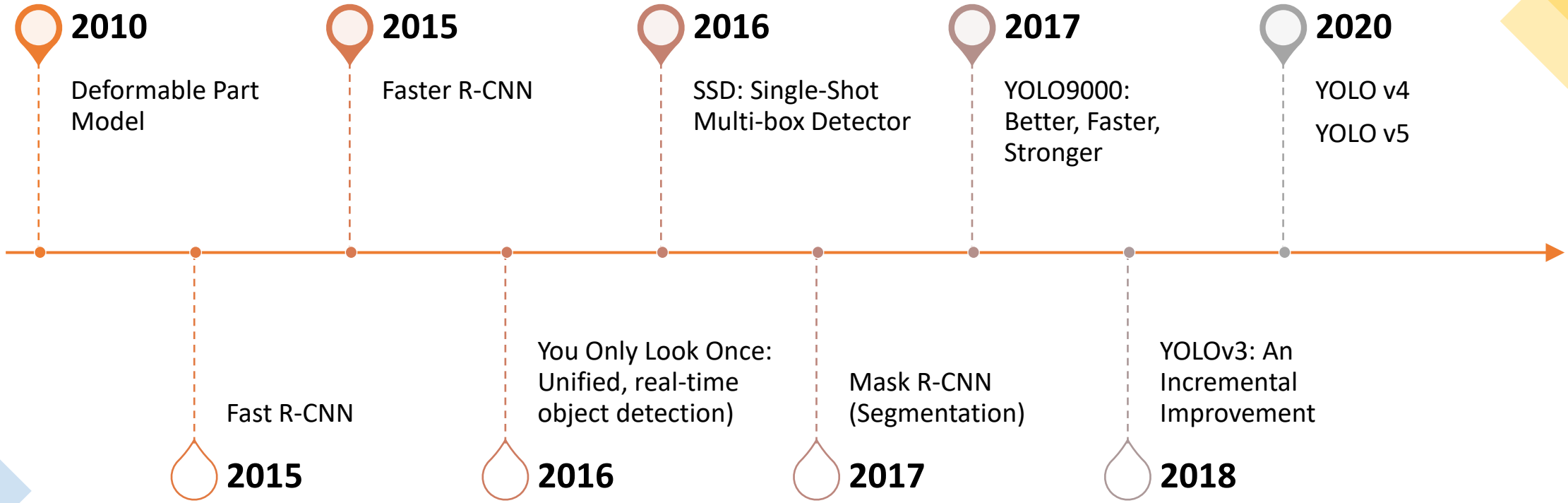
https://www.youtube.com/watch?v=VOC3huqHrss

Object Detection Ranking on COCO Test-dev

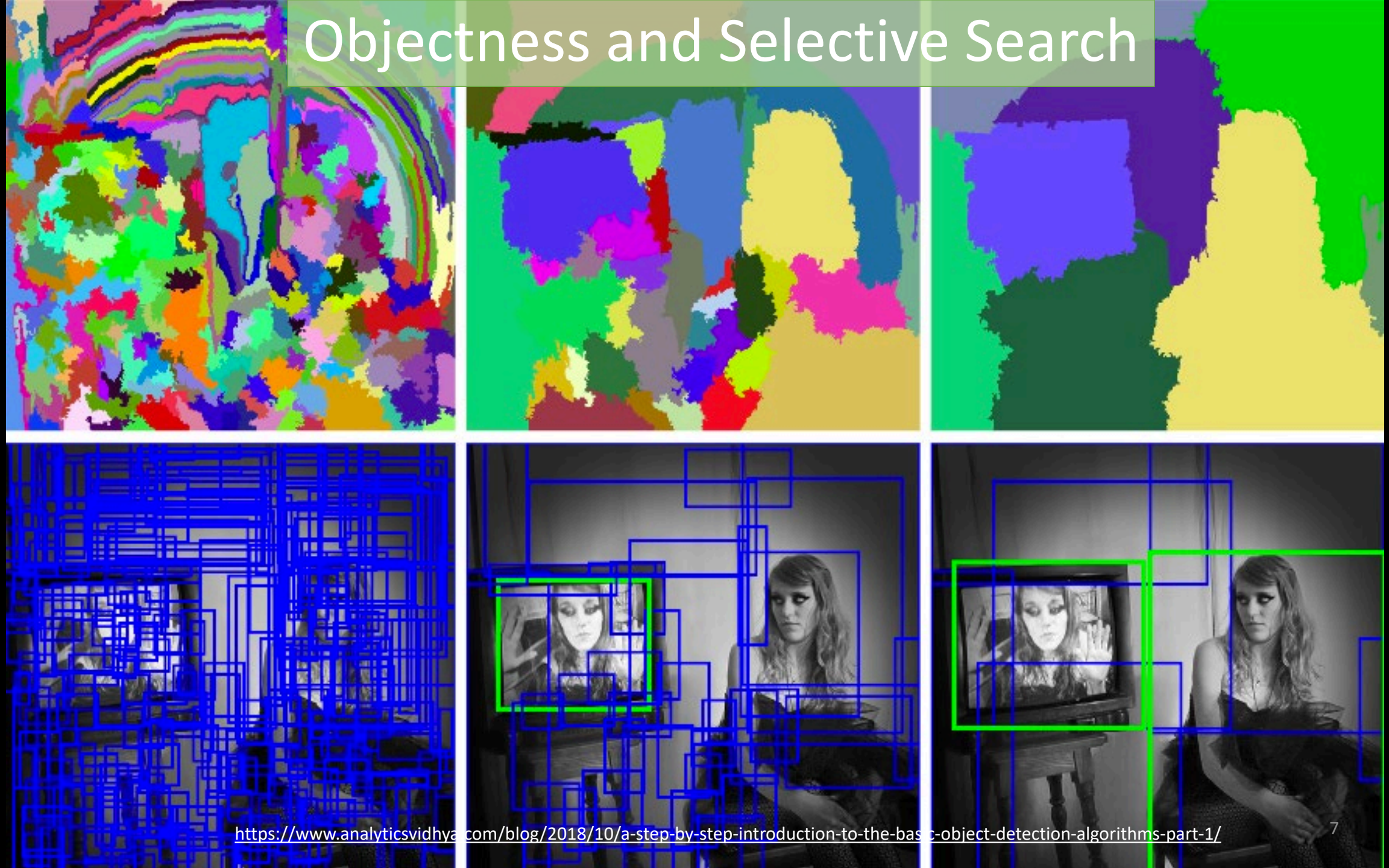


https://paperswithcode.com/sota/object-detection-on-coco

Recent Developments of Object Detection



Objectness and Selective Search

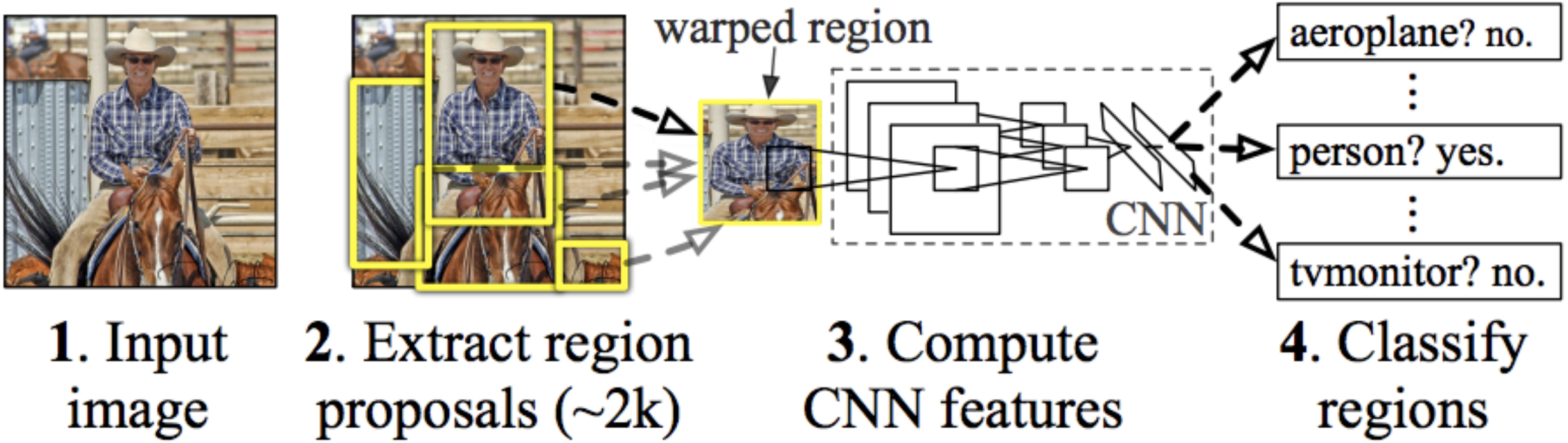


https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/

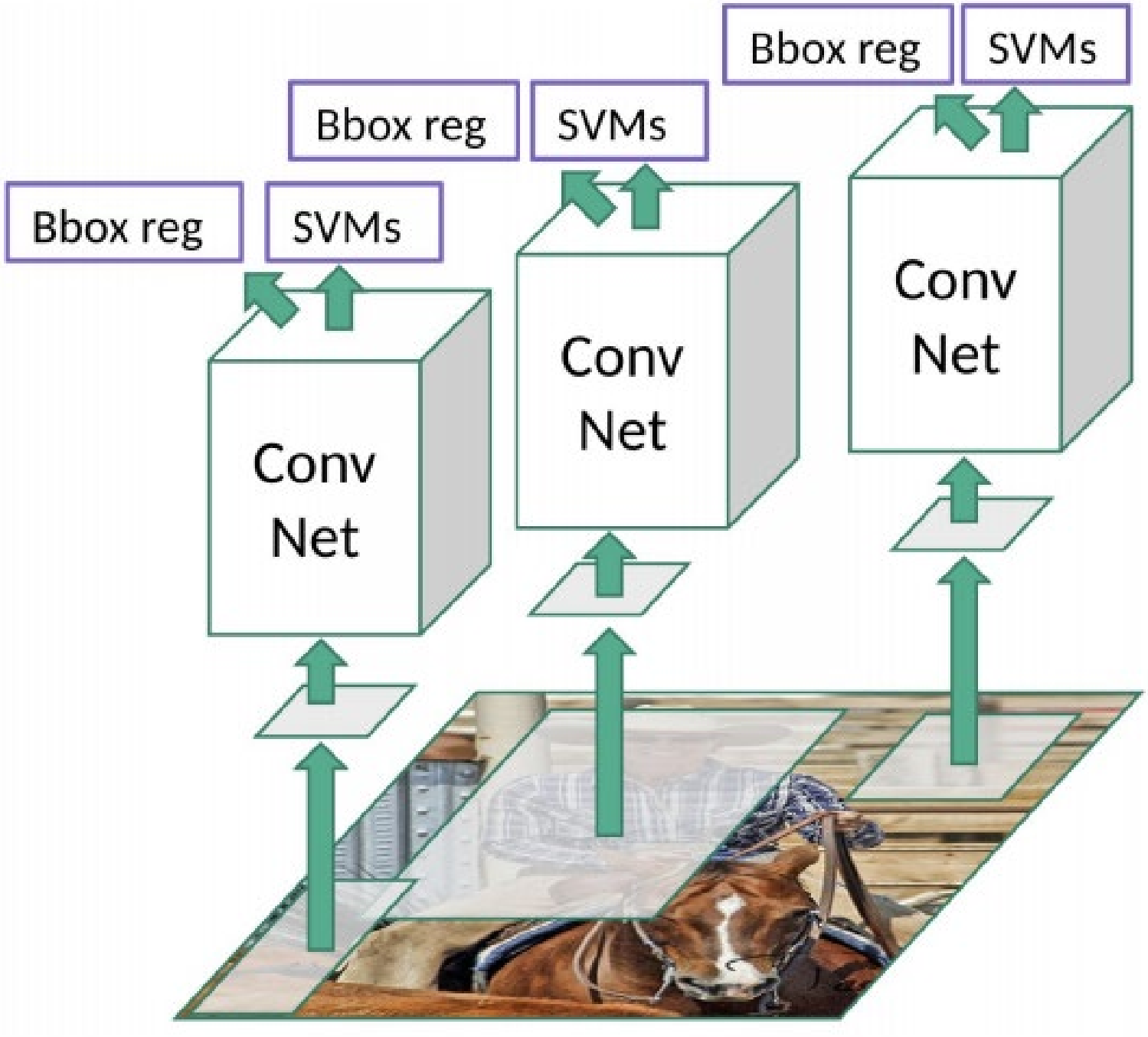
Region Proposal: Multi-scale Objectness Search

- Scan all possible locations and scales for objects

Region Proposal + CNN = R-CNN (2013)



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

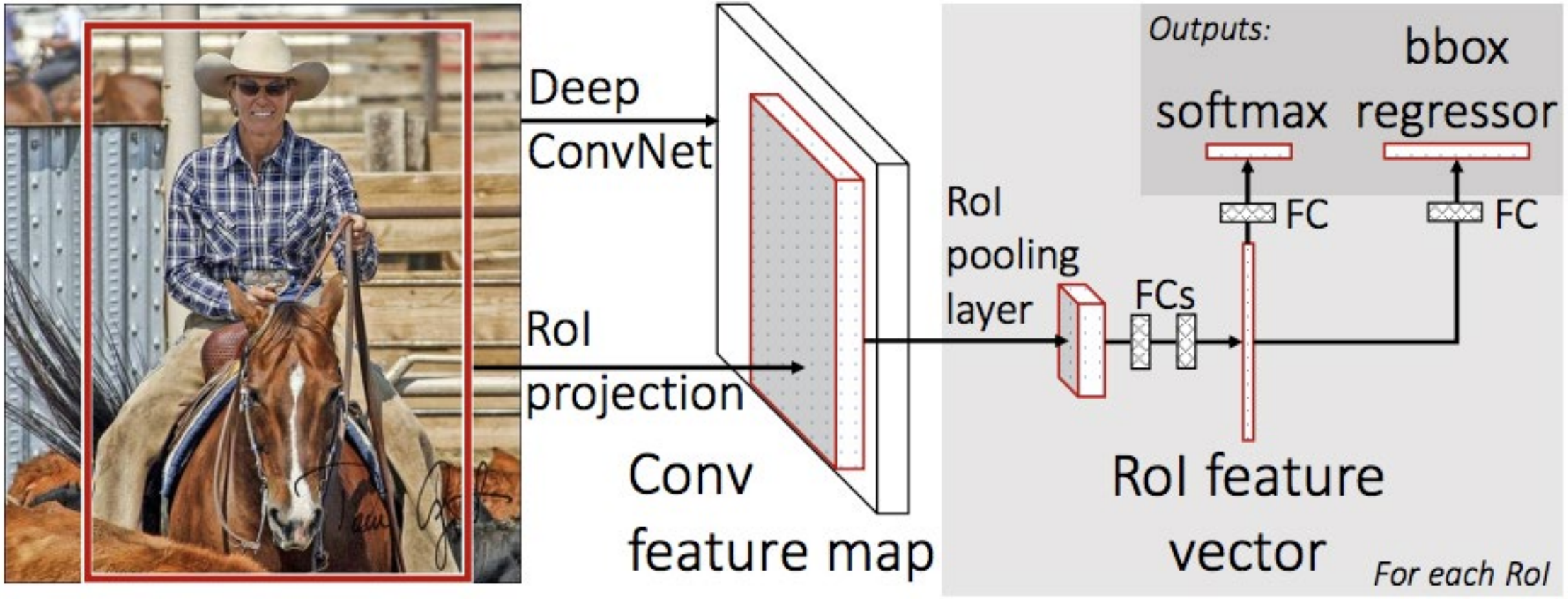

Bbox reg
SVMs
Bbox reg
SVMs
Bbox reg
SVMs
Conv Net
Conv Net
Conv Net

Problems with R-CNN

- 2000 region proposals per image
- It takes around 47 seconds for testing one image (Nvidia K40)
- The selective search algorithm is a fixed algorithm using shallow architecture

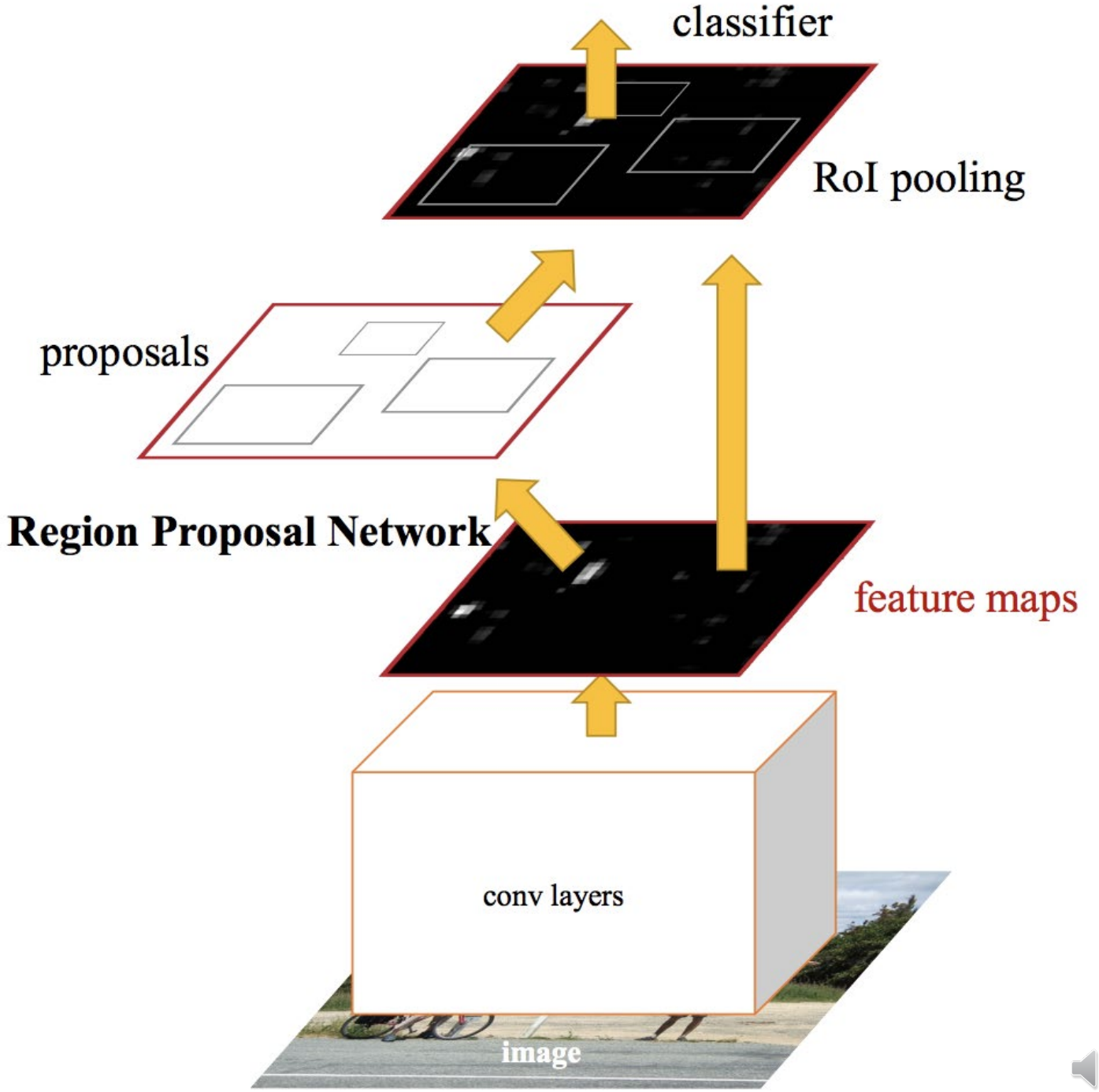
Fast R-CNN (2015)

- Instead of running a CNN 2,000 times per image, run just once per image and get all the regions of interest (RoI)

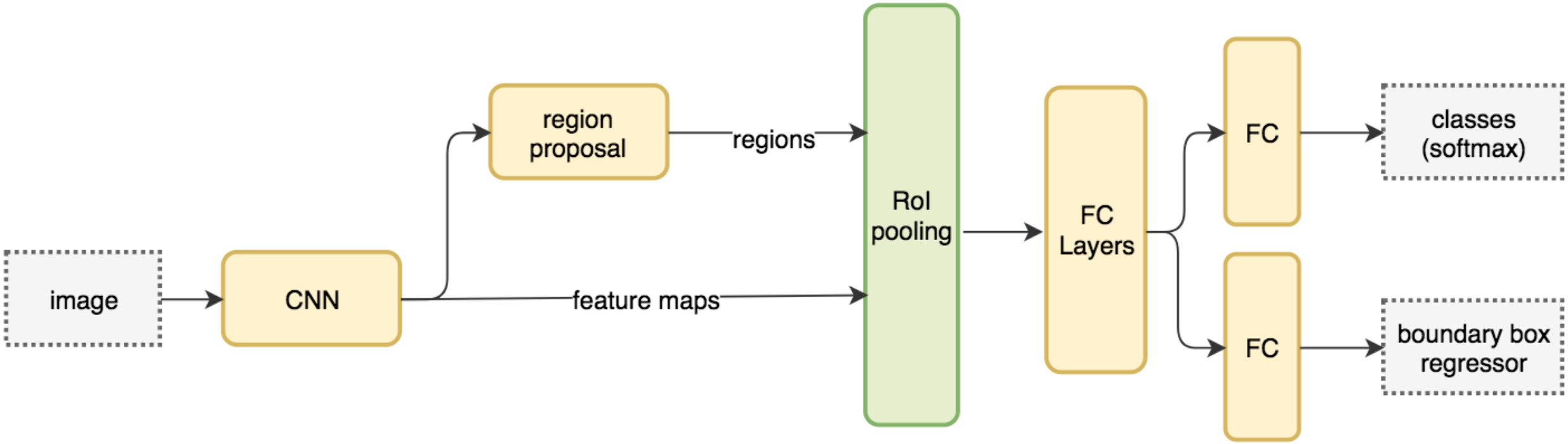


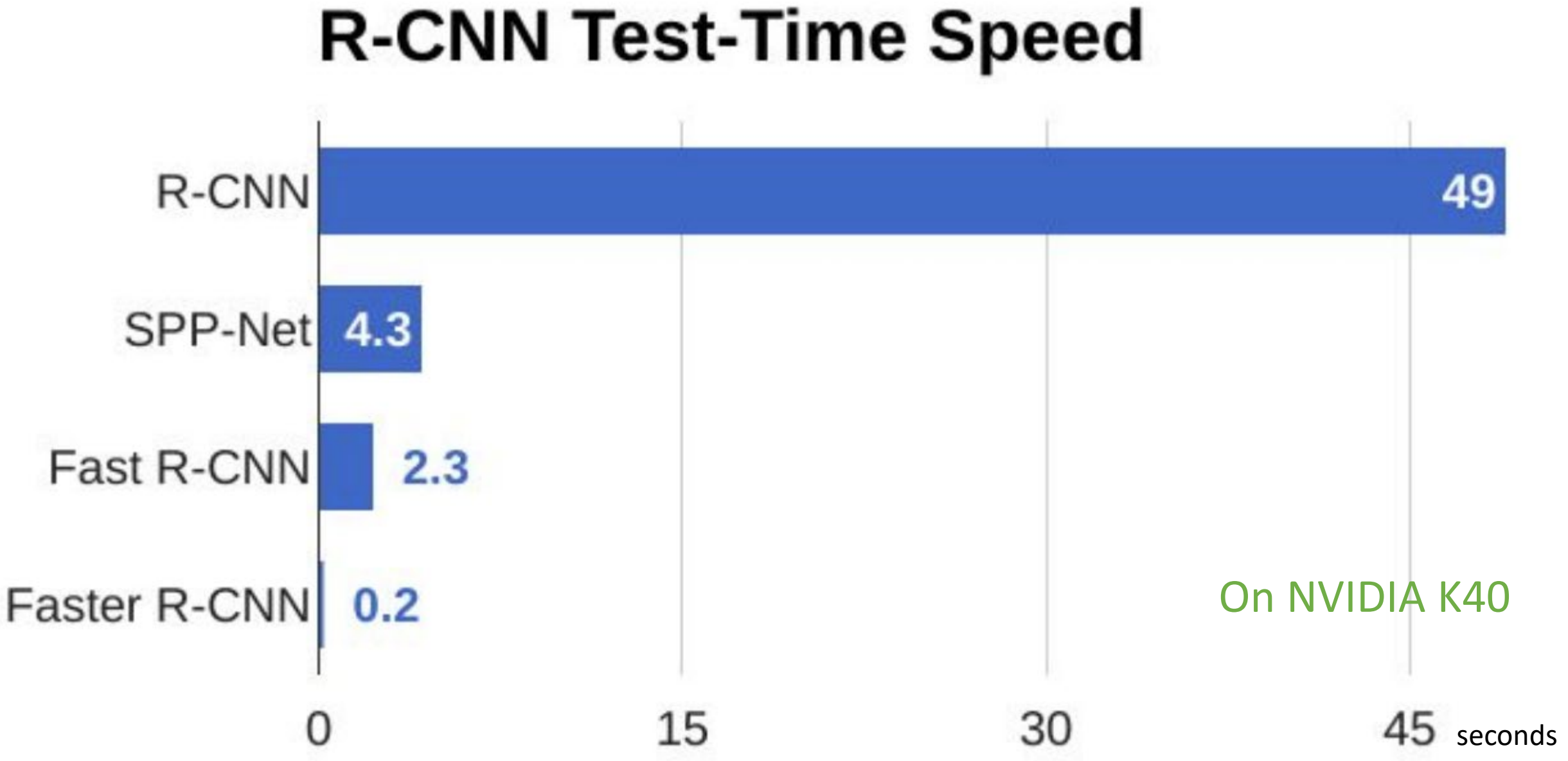
Faster R-CNN

- Replace Selective Search with neural networks



Faster R-CNN Architecture

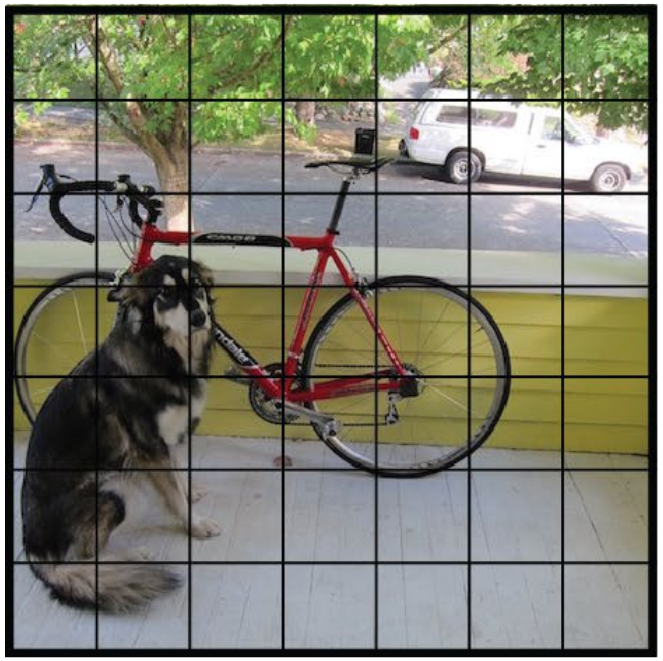


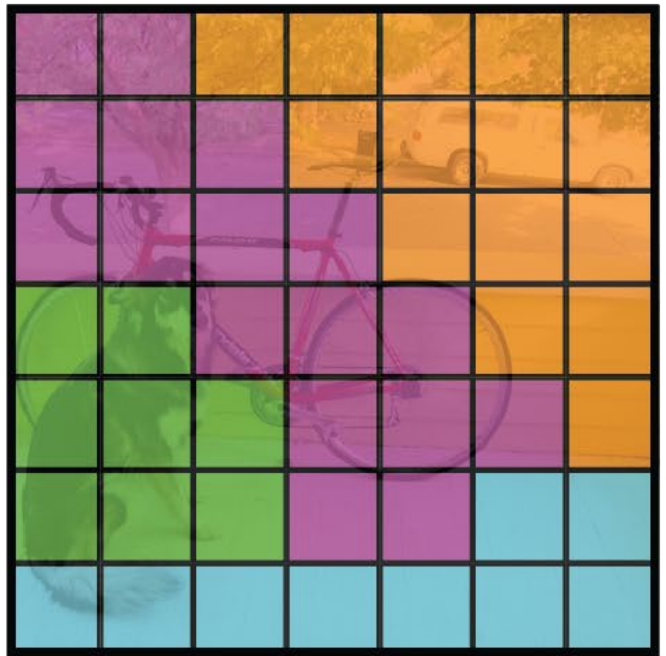

R-CNN Test-Time Speed
R-CNN
49
SPP-Net
4.3
Fast R-CNN
2.3
Faster R-CNN
0.2
On NVIDIA K40
0
15
30
45
seconds

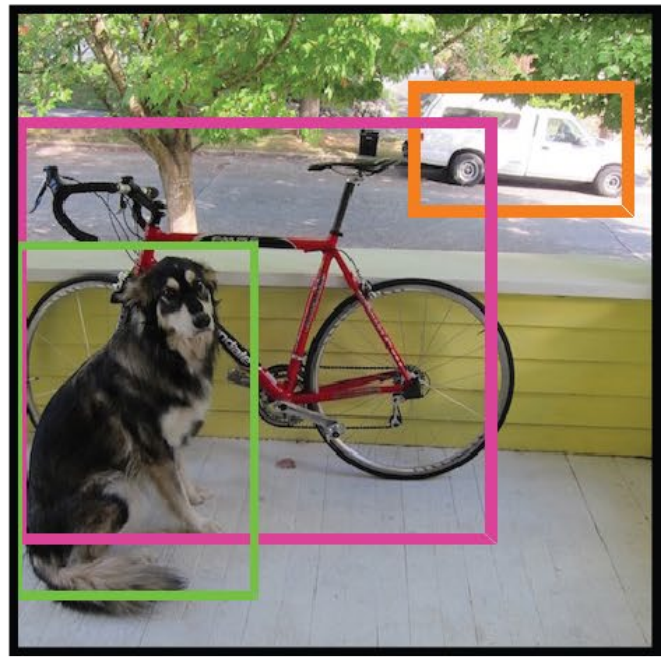
Summary of RCNN Trilogy

Algorithm	Features	Prediction time (Nvidia K40)	Limitations
RCNN	• Use selective search to find object candidate regions • Generate around 2000 regions from each image • Extract CNN features for all the 2000 regions	40-50 secs	High computation time as each region is passed to the CNN separately
Fast RCNN	• Only extract CNN features once for each image • Use selective search to find object candidate regions	2 secs	Selective search is slow and hence computation time is still high.
Faster RCNN	• Replace the selective search with region proposal network.	0.2 sec	Object proposal takes time

YOLO – You Only Look Once (2016)







You Only Look Once:
Unified, Real-Time Object Detection

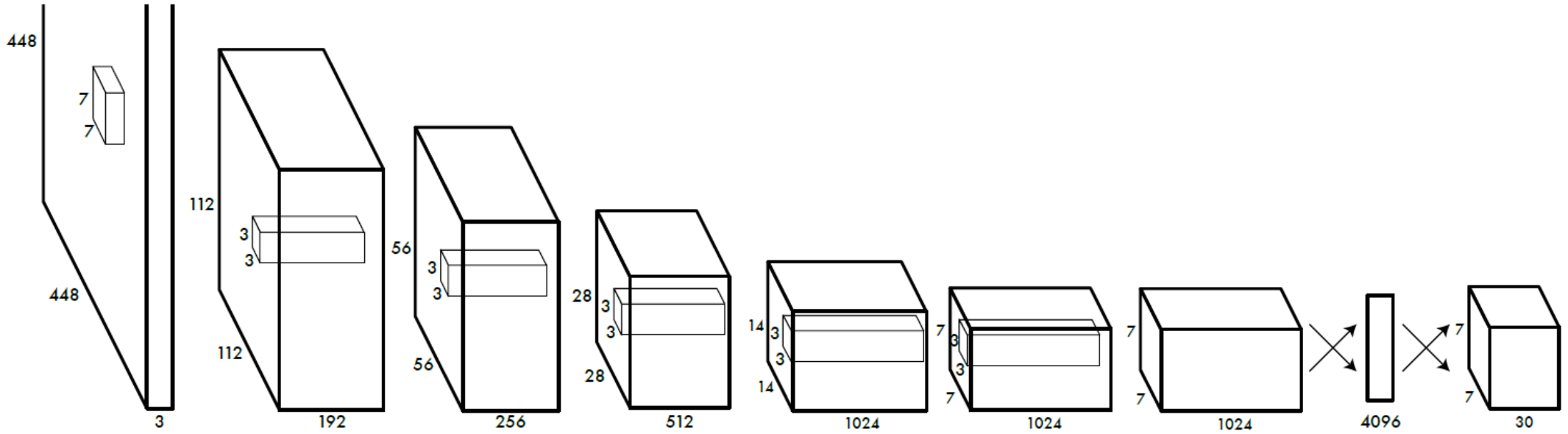
Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶
http://pjreddie.com/yolo/

YOLO v1

- One neural network predicts bounding boxes and class probabilities
- Divide an image into S x S grid, each grid predicts B bounding boxes and C class probabilities
  - bounding box vector: (x, y, w, h, confidence)
  - Final prediction vector size: S x S x (B*5 + C)

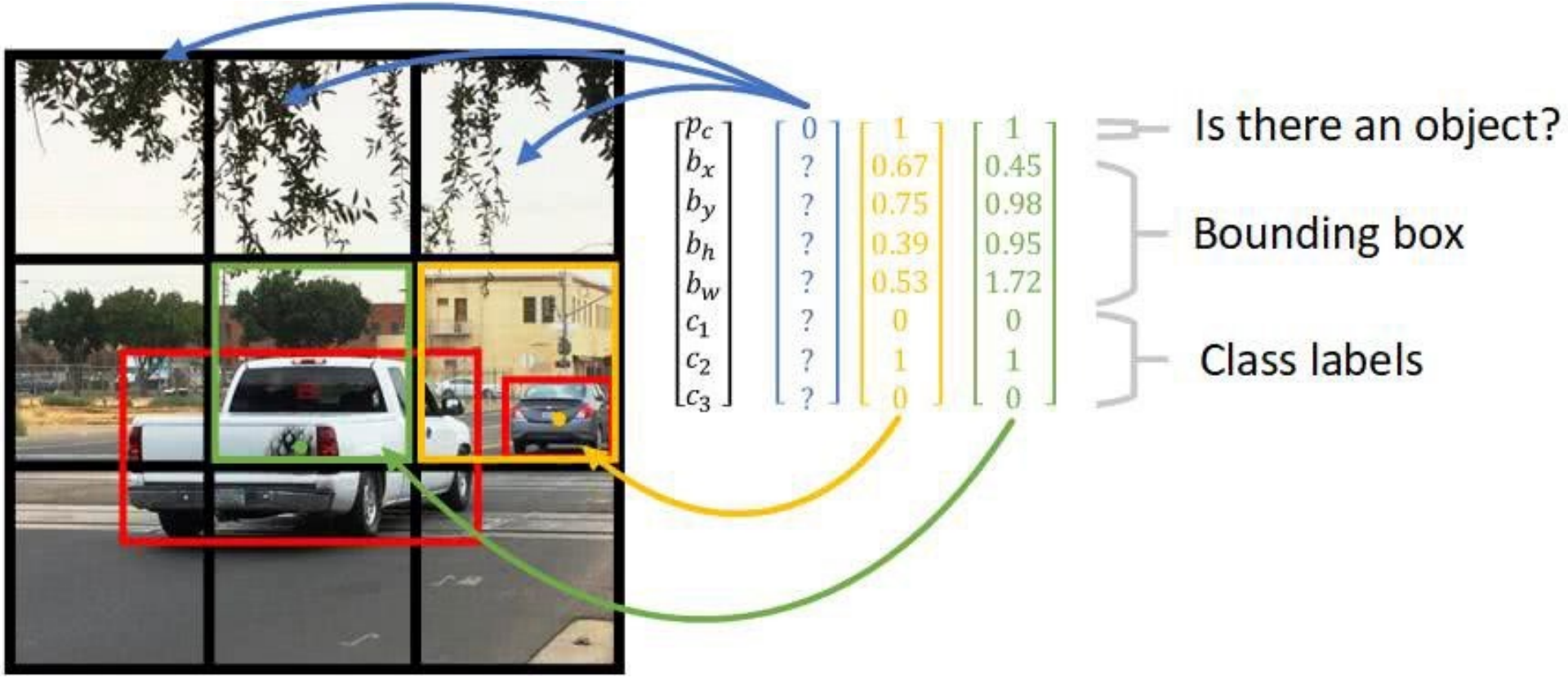
CNN Architecture of YOLO v1

- YOLO v1 network for PASCAL VOC dataset. The authors set S = 7, B = 2. 20 labelled classes C = 20.
- Final prediction is a 7 * 7 * (2*5 + 20) tensor.



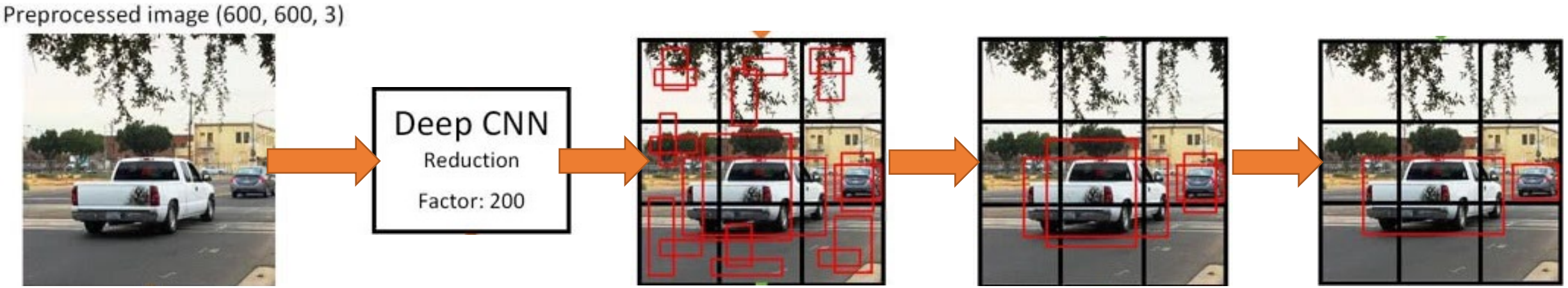
Example: Detecting Objects in the Grids

- 3 classes: pedestrian, car, motorcycle
- 1 bounding box per grid



https://heartbeat.fritz.ai/gentle-guide-on-how-yolo-object-localization-works-with-keras-part-2-65fe59ac12d

YOLO Detection Flow



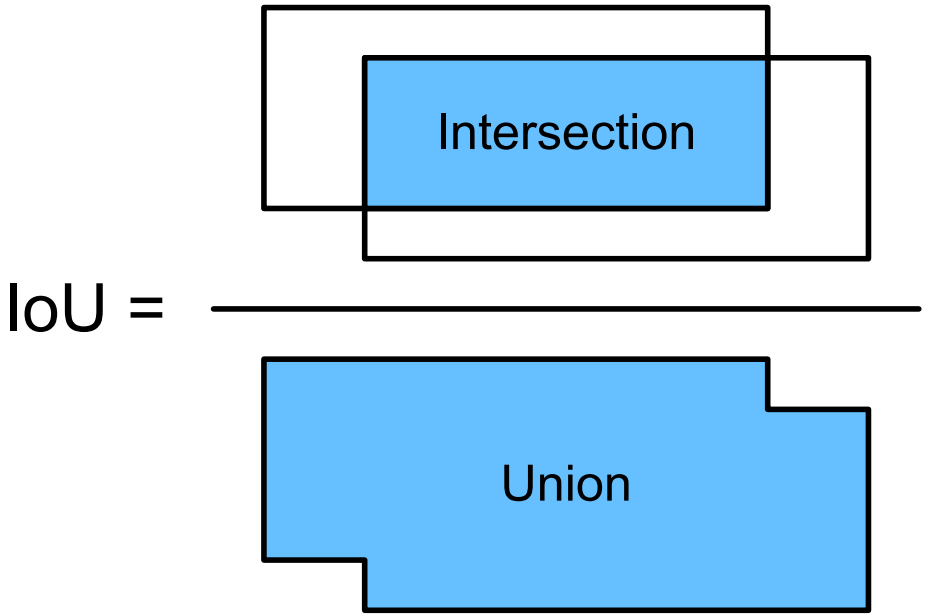
Filer boxes by class scores

Non-max suppression

https://heartbeat.fritz.ai/gentle-guide-on-how-yolo-object-localization-works-with-keras-part-2-65fe59ac12d

Intersection over Union (IoU)

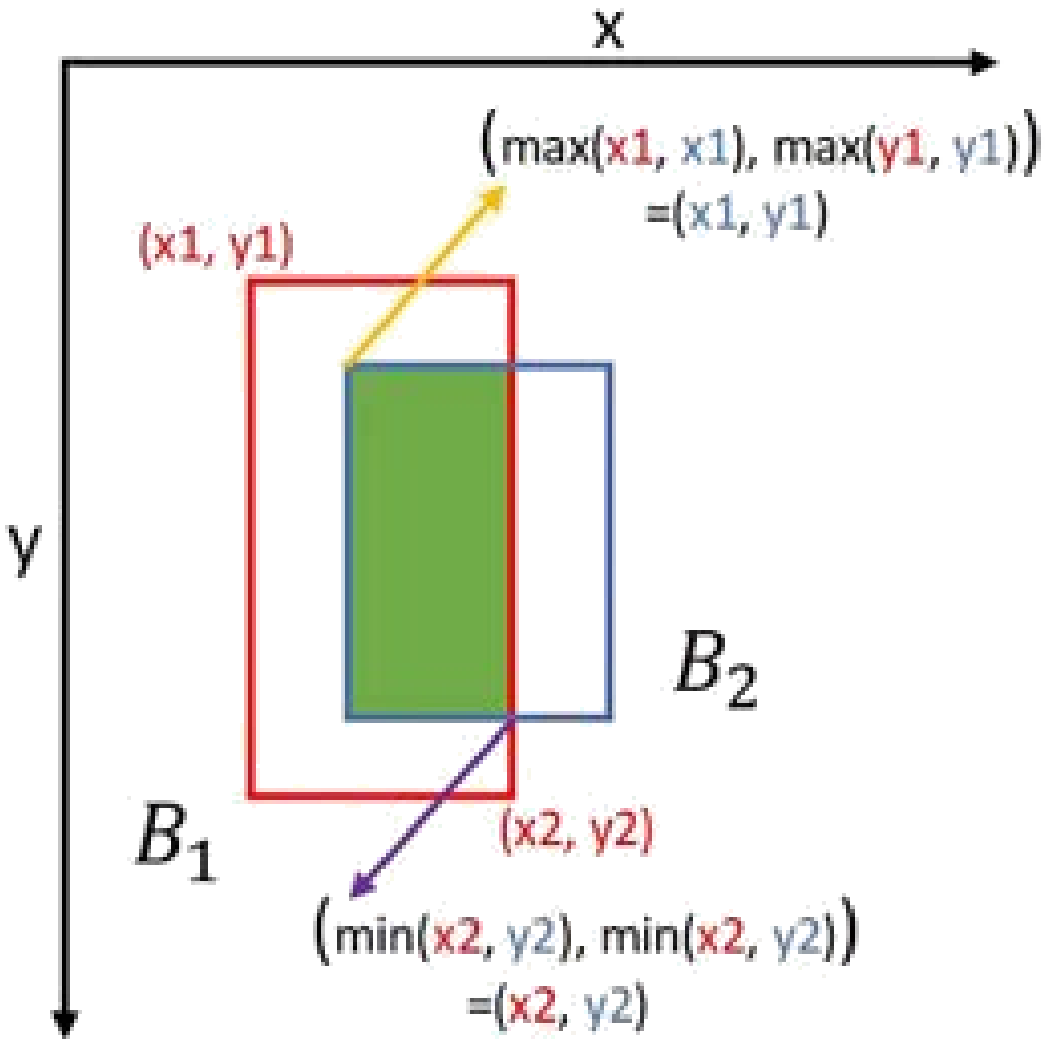
- Measure the similarity between the predicted box and the ground-truth bounding box



https://d2l.ai/chapter_computer-vision/anchor.html

Checking Intersections

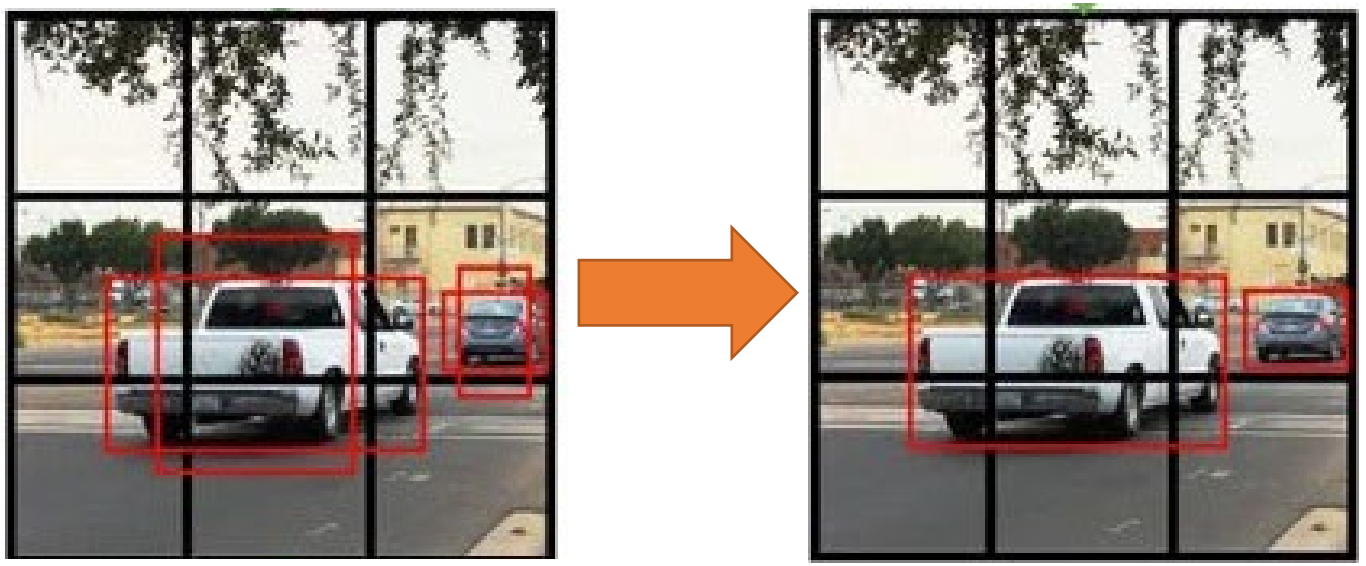
- Define bounding boxes using two corners: upper left (x1, y1) and lower right (x2, y2)



https://heartbeat.fritz.ai/gentle-guide-on-how-yolo-object-localization-works-with-keras-part-2-65fe59ac12d

Non-maximum Suppression

- Combine multiple bounding boxes
  - Discard all boxes with confidence less or equal to 0.6.
  - Pick the box with the largest confidence output as a prediction.
  - Discard any remaining box with IoU greater than or equal to 0.5.



DarkNet

- On ImageNet
  - VGG (30.69 billion FLOPS)
  - GoogLeNet (8.52 billion FLOPS)
  - DarkNet (5.58 billion FLOPS)

- DarkNet uses mostly 3 × 3 filters to extract features and 1 × 1 filters to reduce output channels

Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2/2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2/2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
~~Convolutional~~	1000	1 × 1	7 × 7
Avgpool		Global	1000
Softmax			

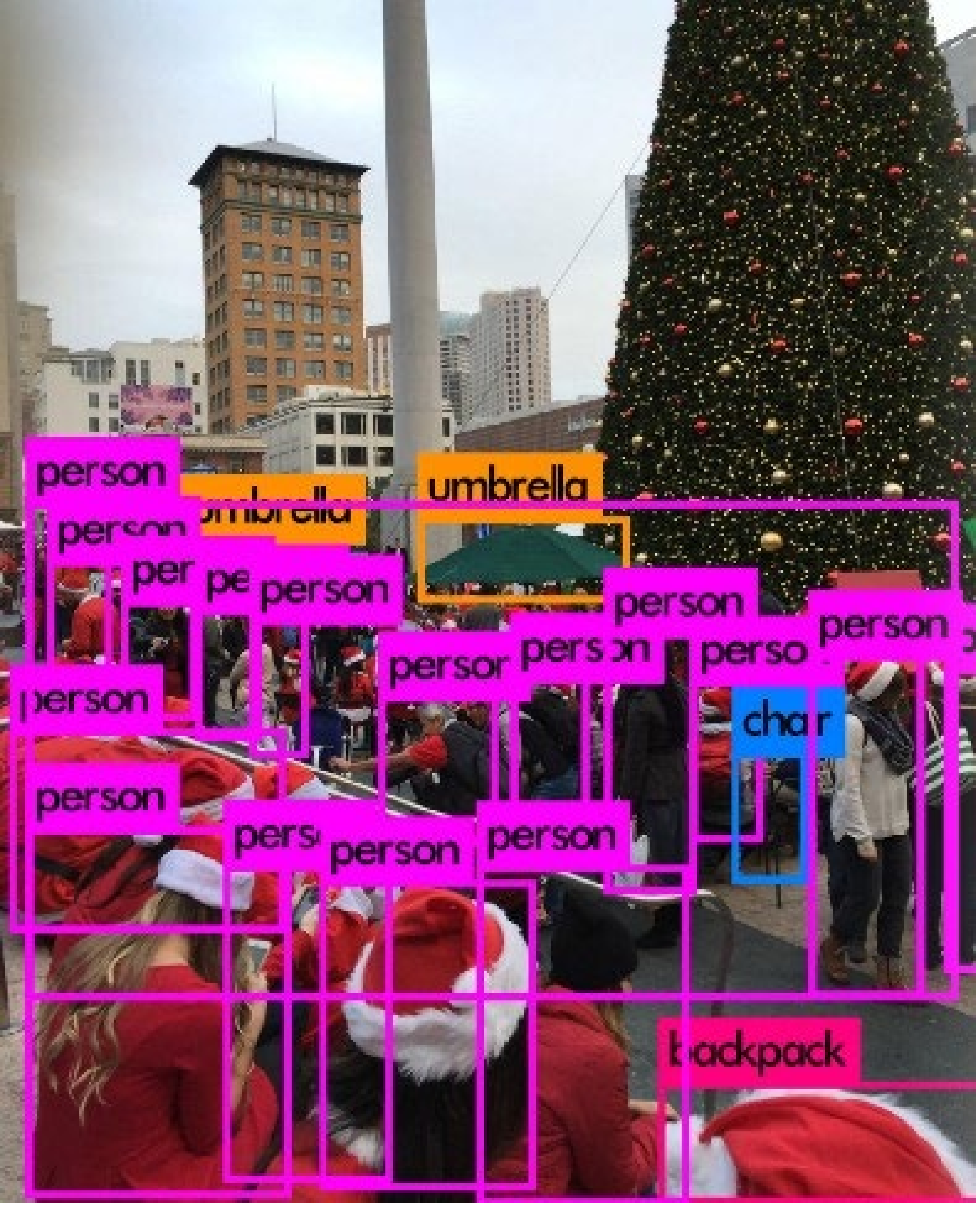
Real-Time Systems on PASCAL VOC 2007

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

← 9 conv. layers

← 24 conv. layers

Running on Nvidia Titan X



Limitations of YOLO

- Lower recall rate and higher localization error compared to Faster R-CNN.
- Each grid can only find 1 object (with 2 bounding-boxes proposal).
- Struggle to detect small objects.

YOLO v2 – YOLO 9000: Better, Faster, Stronger

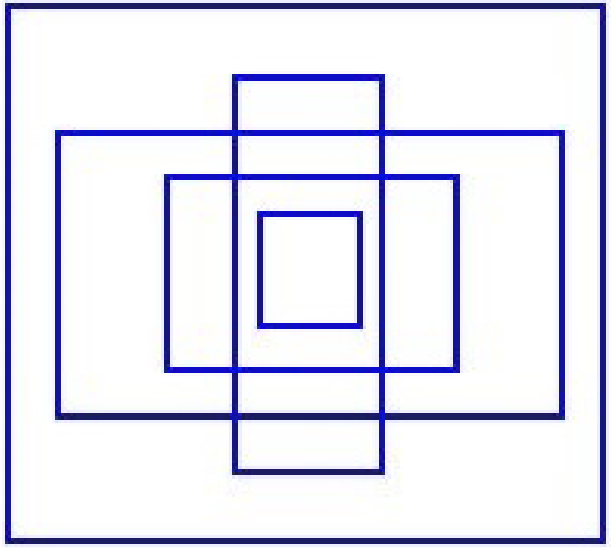
- Batch normalization
- High-resolution pre-trained CNN (224*224 -> 448*448)
- Convolutional with Anchor Boxes

	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**

(Adding anchor boxes led to same accuracy but higher recall rate)

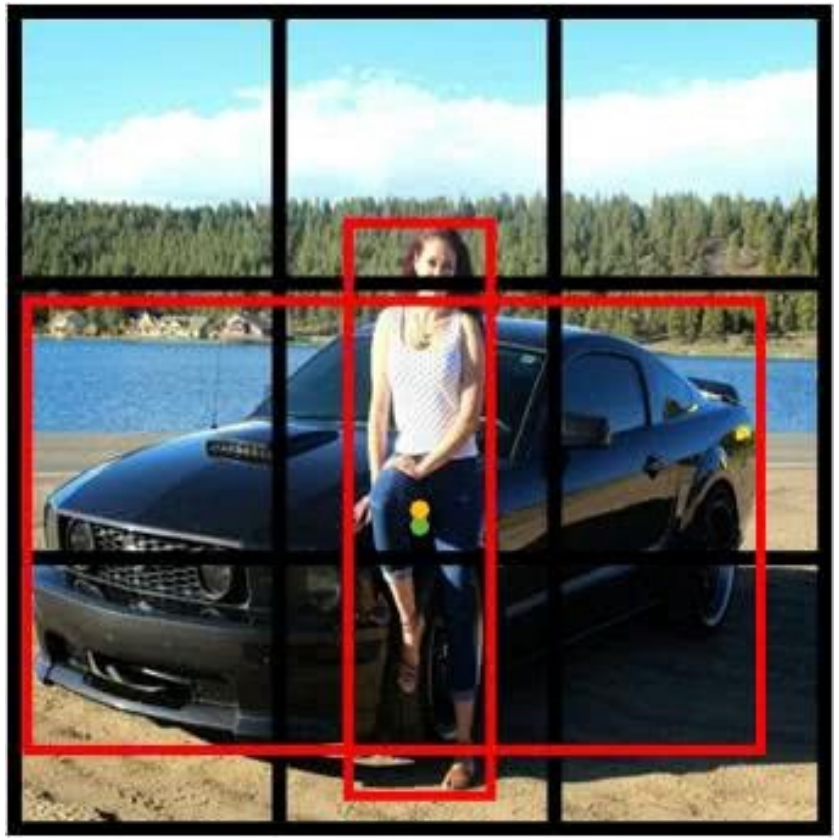
Anchor Boxes

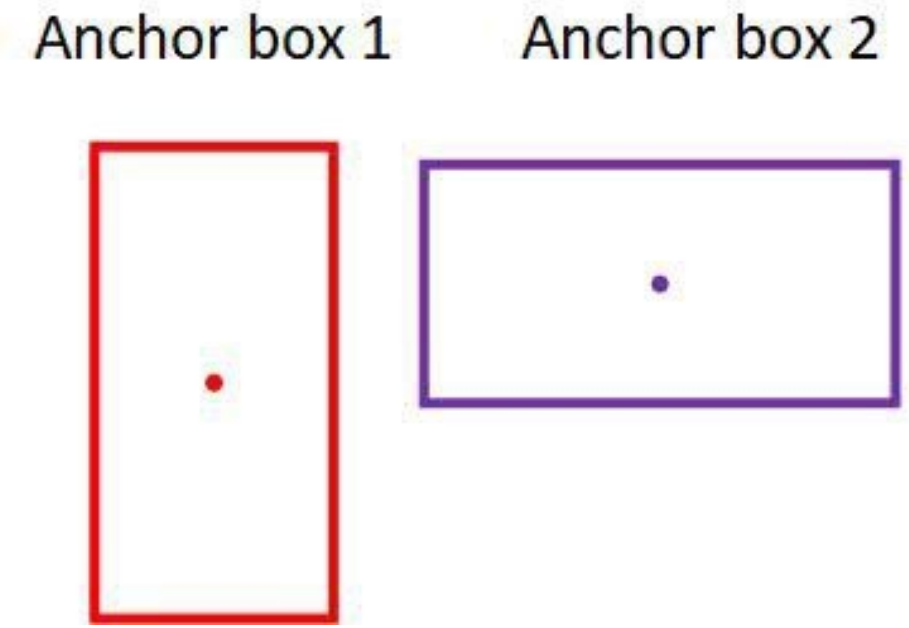
- Detect objects with different shapes
- Detect overlapping shapes



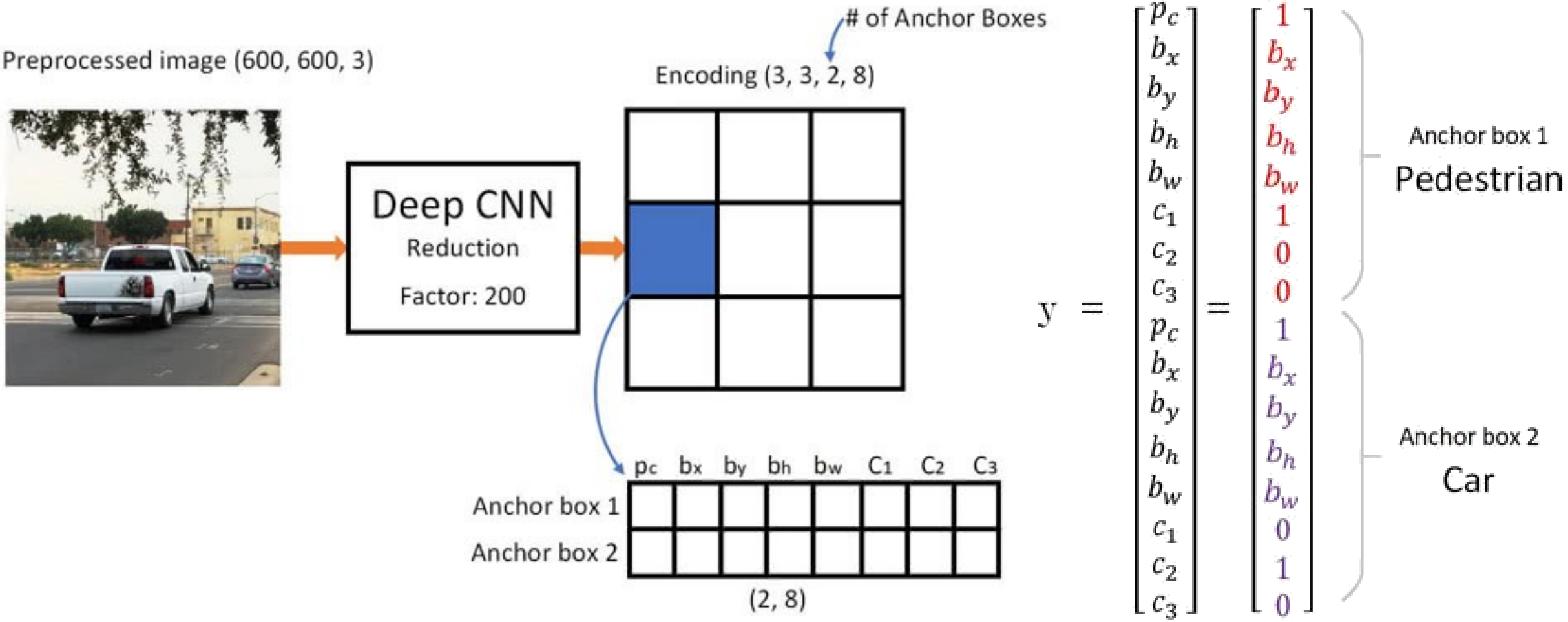
https://heartbeat.fritz.ai/gentle-guide-on-how-yolo-object-localization-works-with-keras-part-2-65fe59ac12d

Detecting Objects using Multi Anchor Boxes

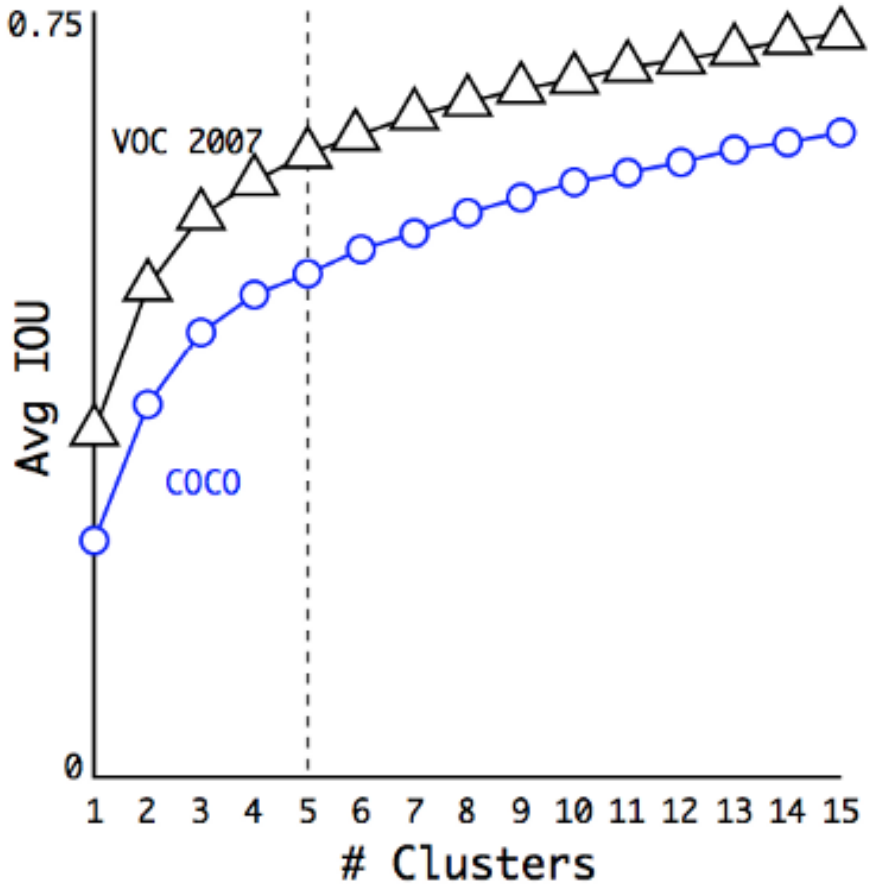


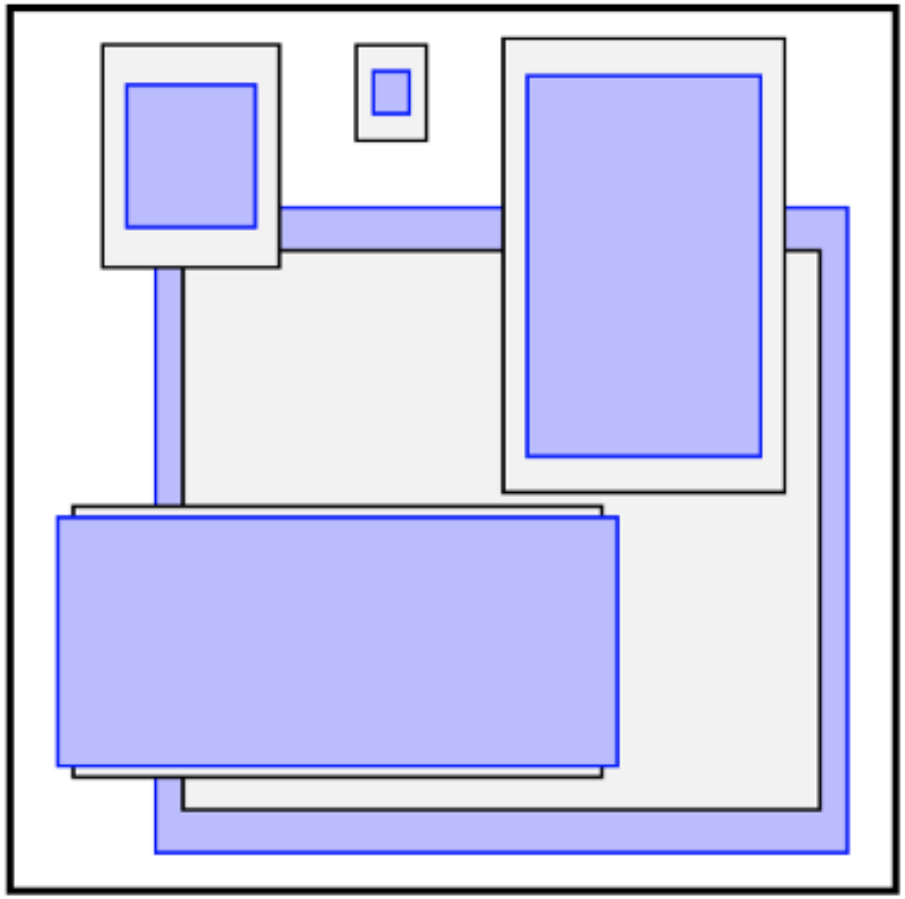


Each Grid has Multiple Anchor Box Predictions

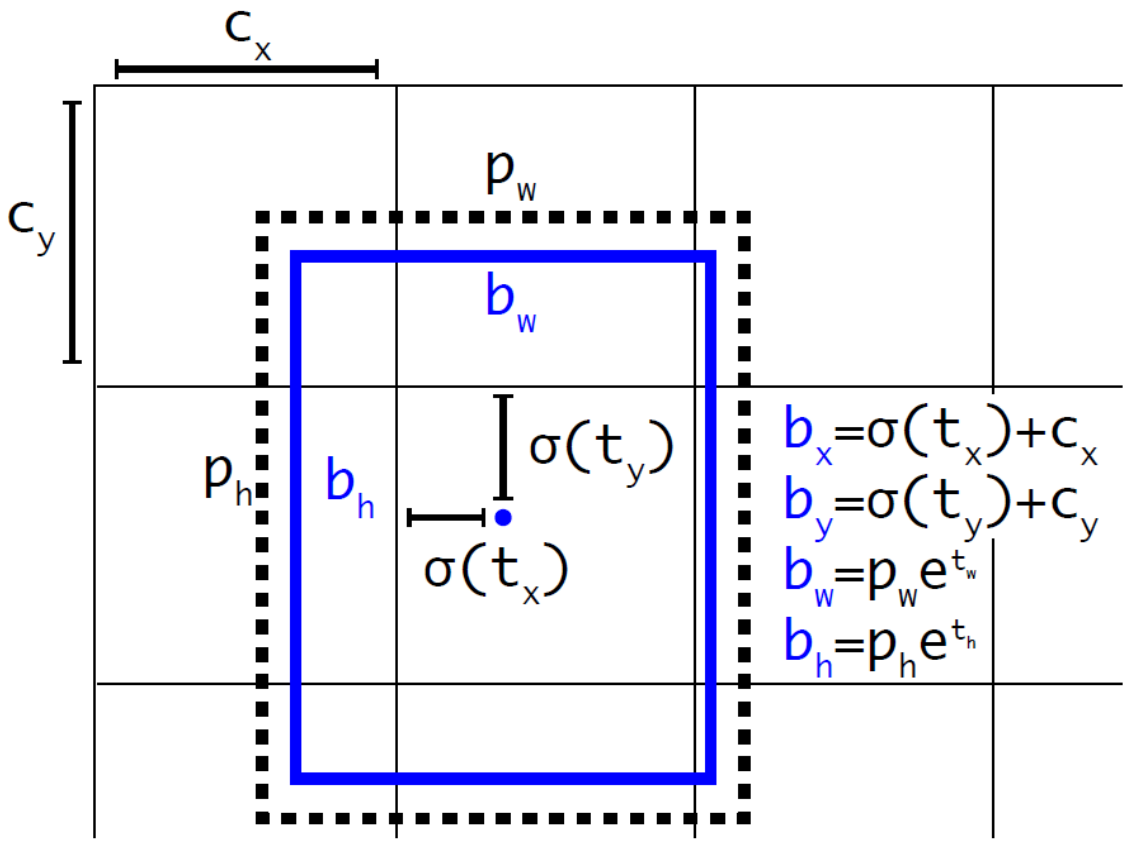


Using K-means Clustering to Find Anchor Boxes





Bounding Box Prediction with Priors

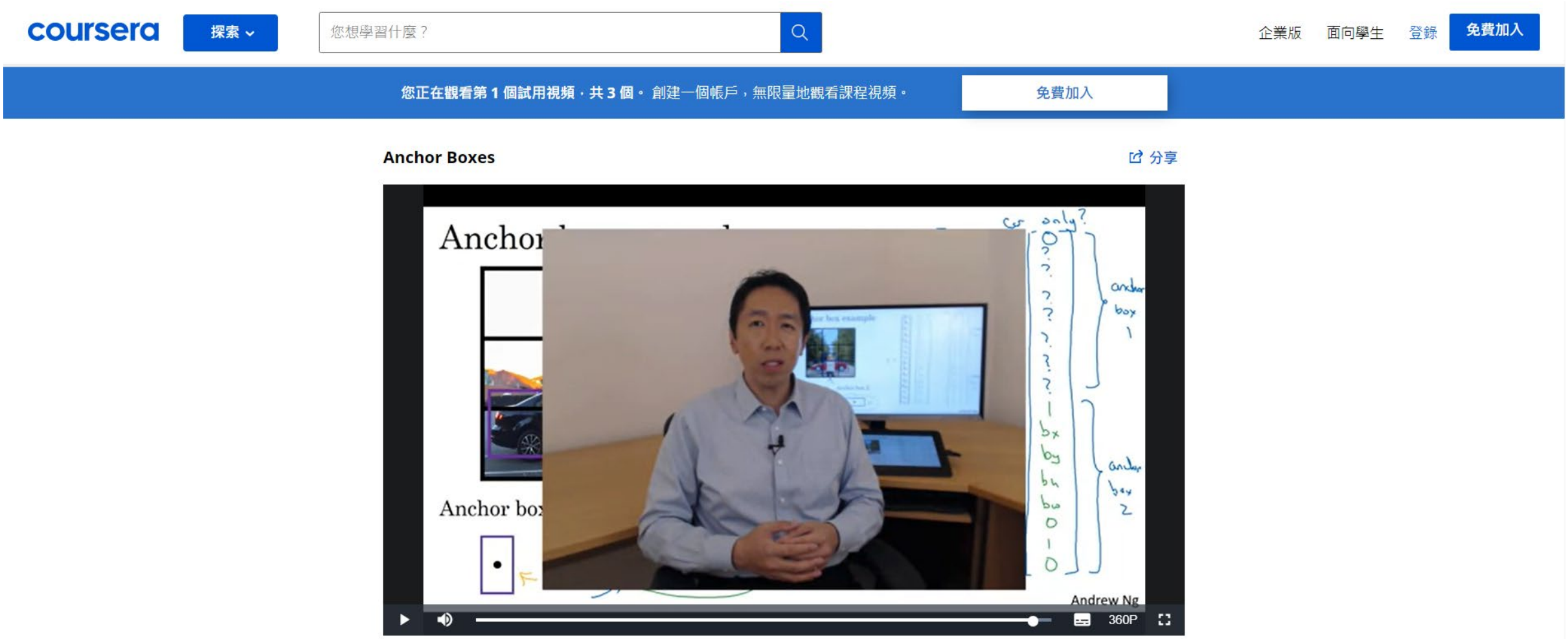


- Predict t_x, t_y, t_w, t_h, t_o
- Cell offset c_x, c_y

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$

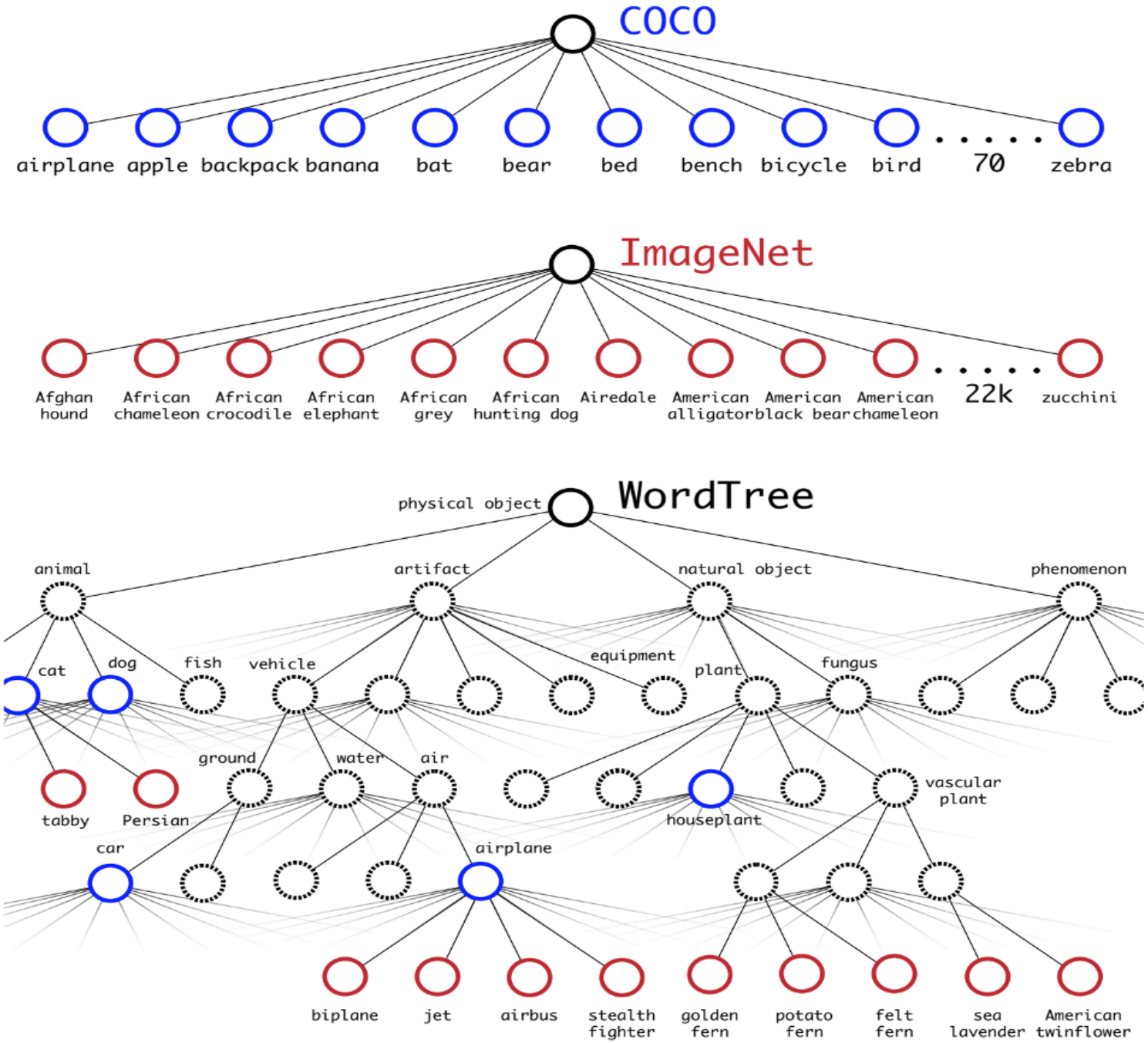
$$Pr(\text{object}) * IOU(b, \text{object}) = \sigma(t_o)$$

Introduction to Anchor Boxes by Andrew Ng



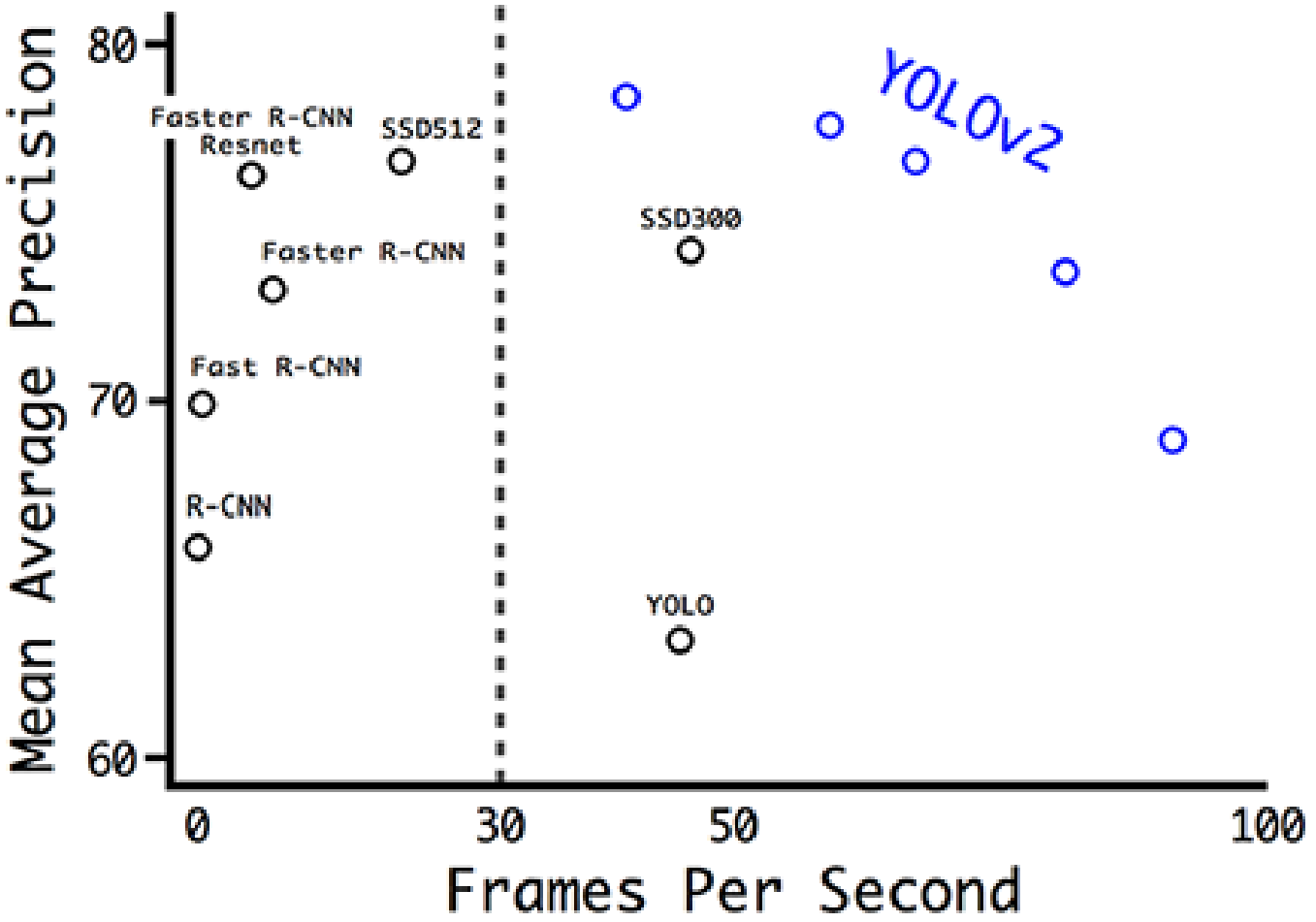
https://www.coursera.org/lecture/convolutional-neural-networks/anchor-boxes-yNwO0

Hierarchical Classification



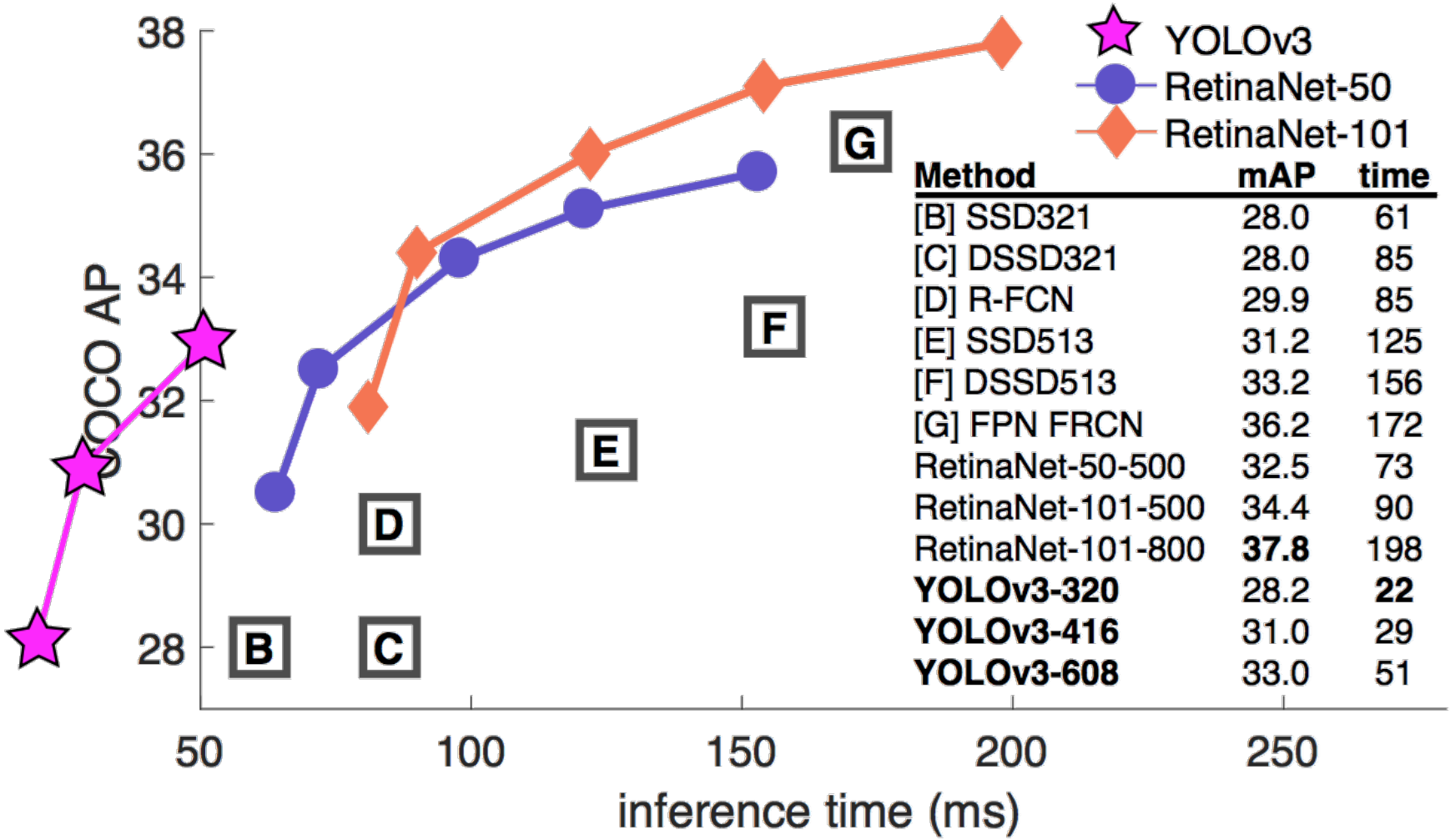
Performance of YOLOv2 on VOC 2007

- Running on Nvidia GTX Titan X



YOLO v3: An Incremental Improvement

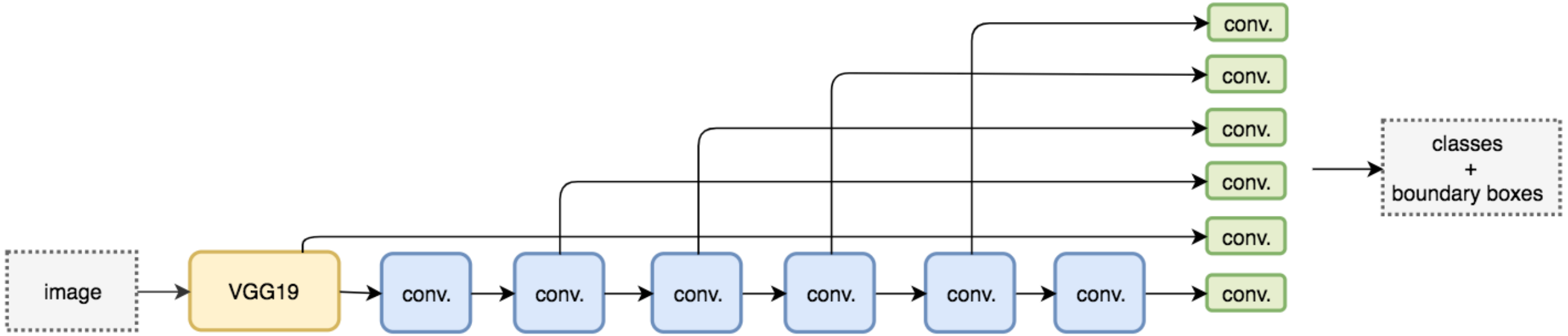
- Using Feature Pyramid with 3 scales; 8 clusters for COCO dataset



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

Single-Shot Multi-Box Object Detection (SSD)

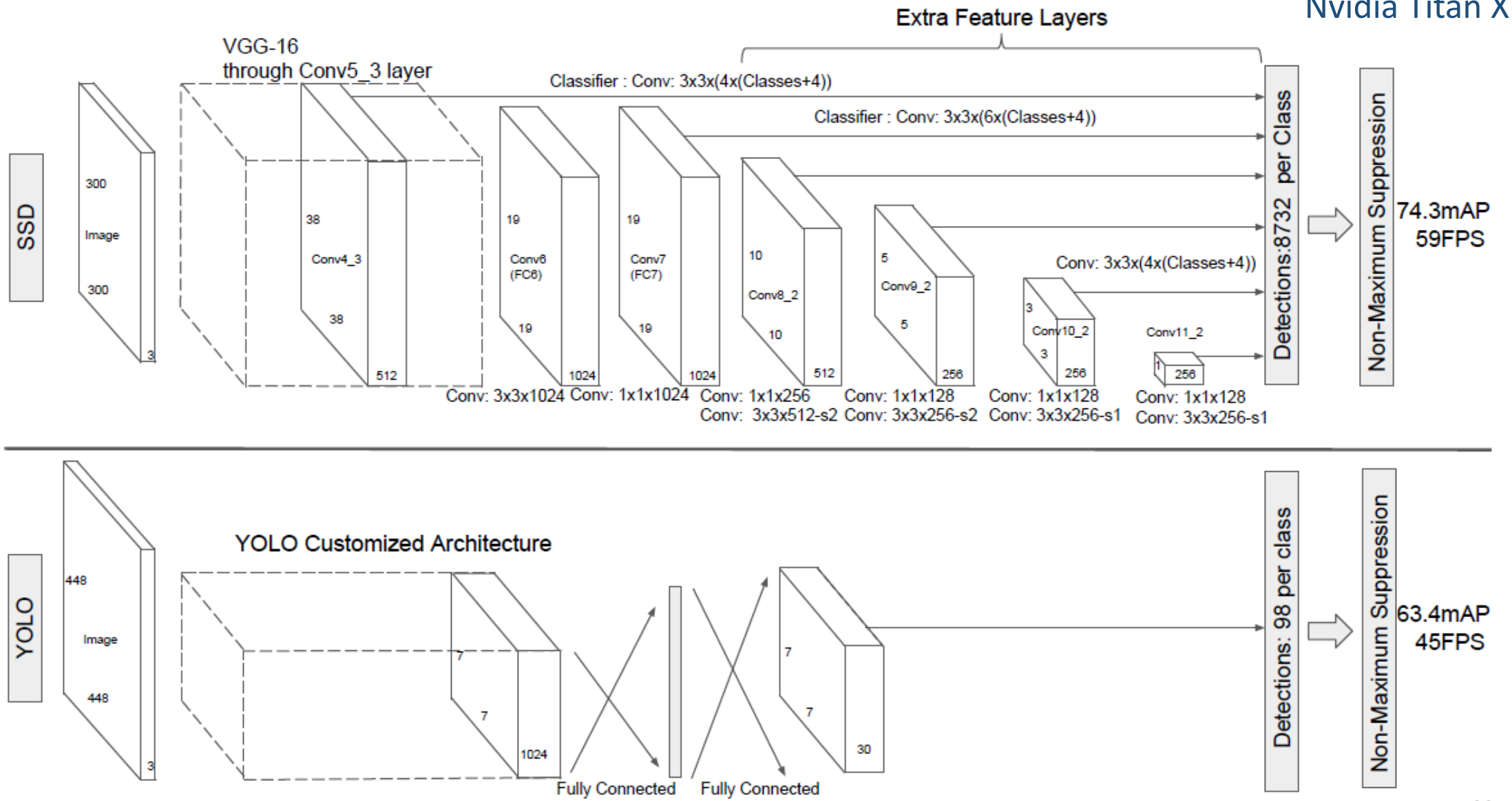
- W. Liu et al., “SSD: Single Shot MultiBox Detector,” 2016



https://arxiv.org/pdf/1512.02325.pdf

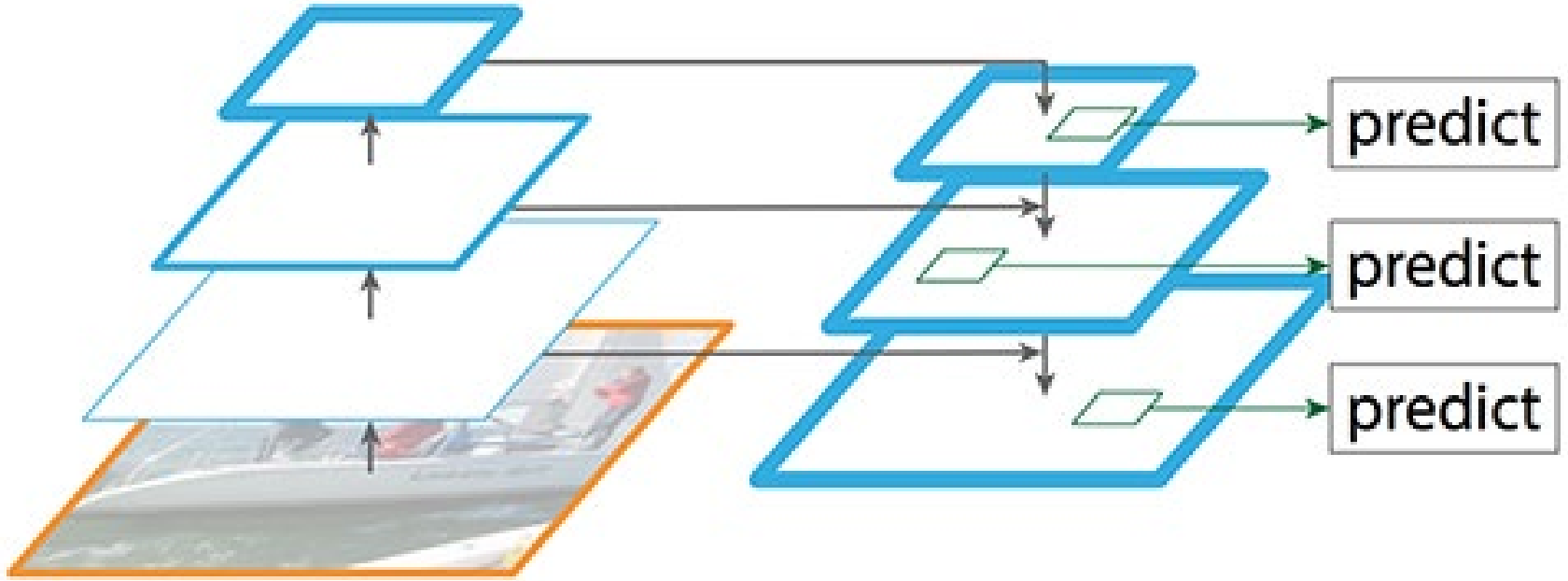
SSD vs. YOLO

VOC 2007 dataset
Nvidia Titan X



Feature Pyramid Networks (FPN)

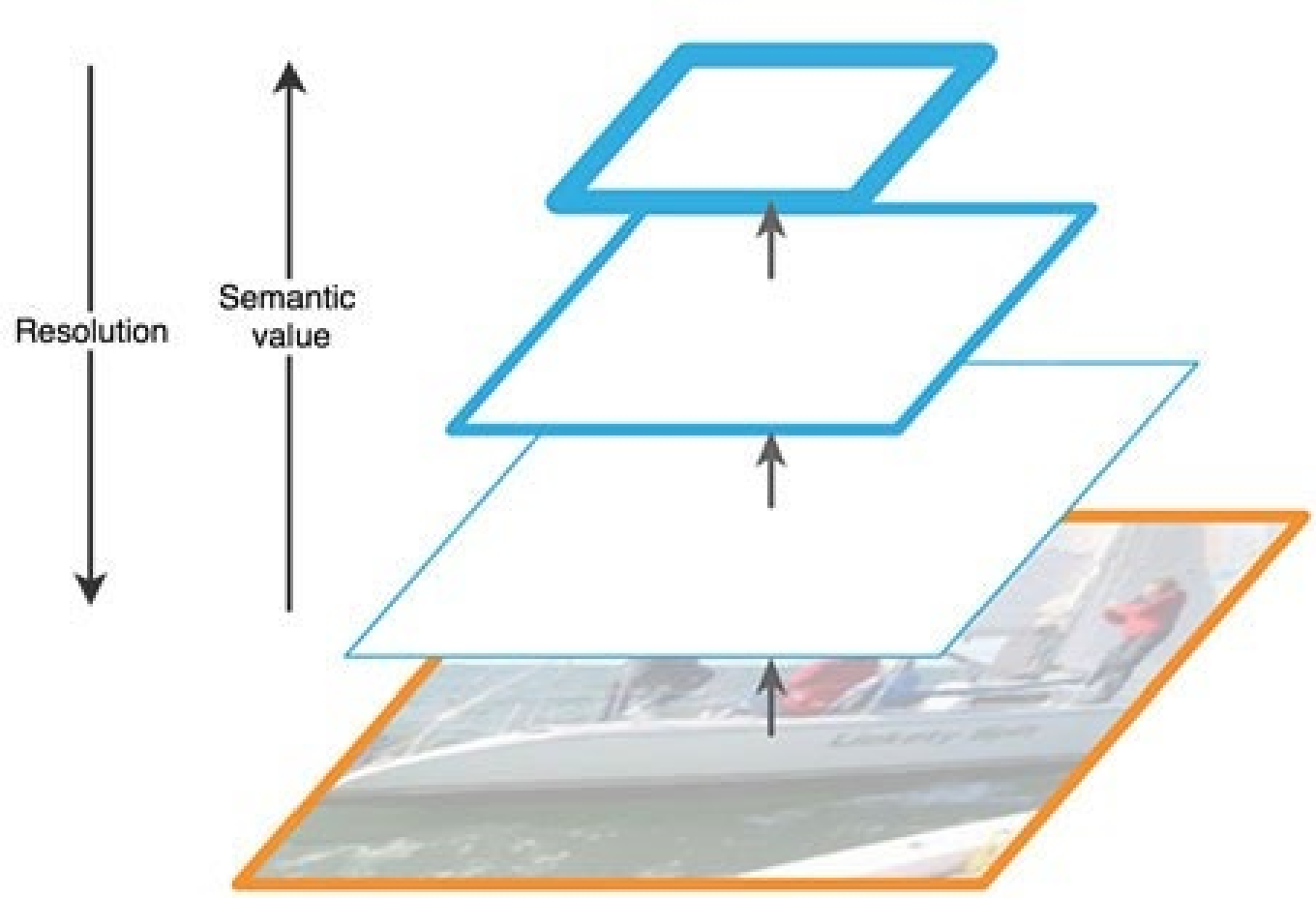
- Traditional featured Image pyramid can increase accuracy, but cost too much memory



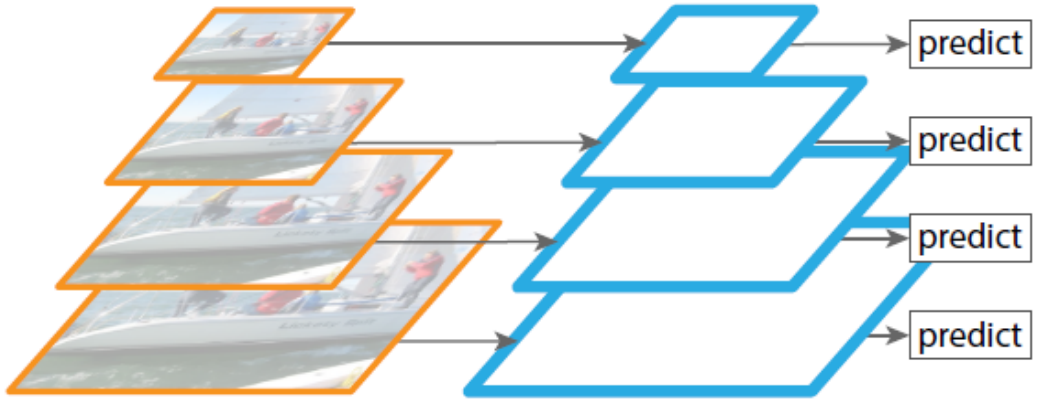
Tsung-Yi Lin, “Feature Pyramid Networks for Object Detection,” 2017

Using CNN Pyramidal Features

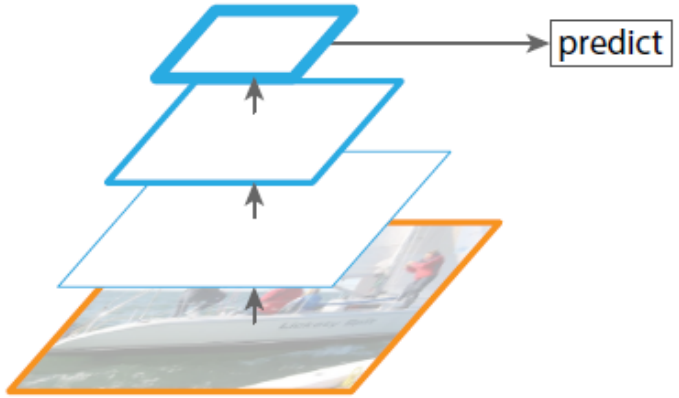
- ConvNets create feature pyramid naturally



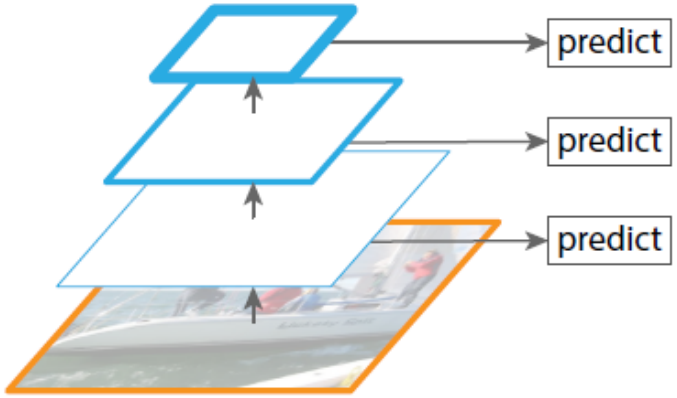
Feature Pyramids Comparison



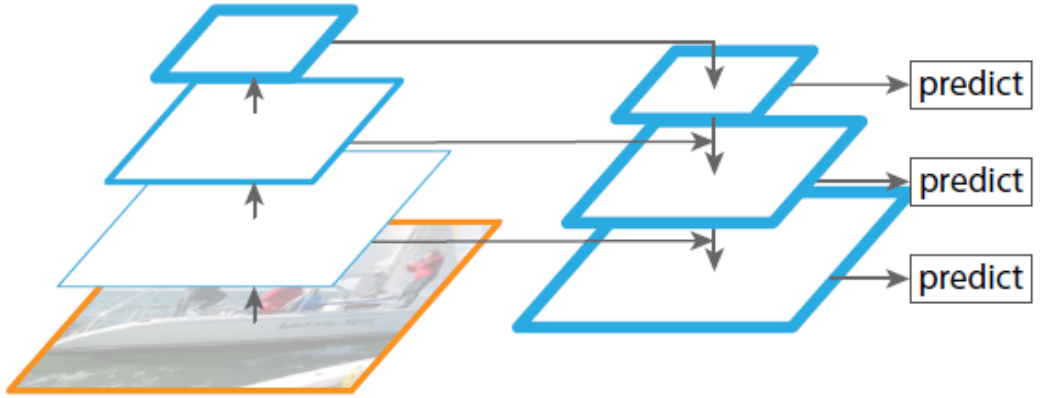
(a) Featurized image pyramid



(b) Single feature map



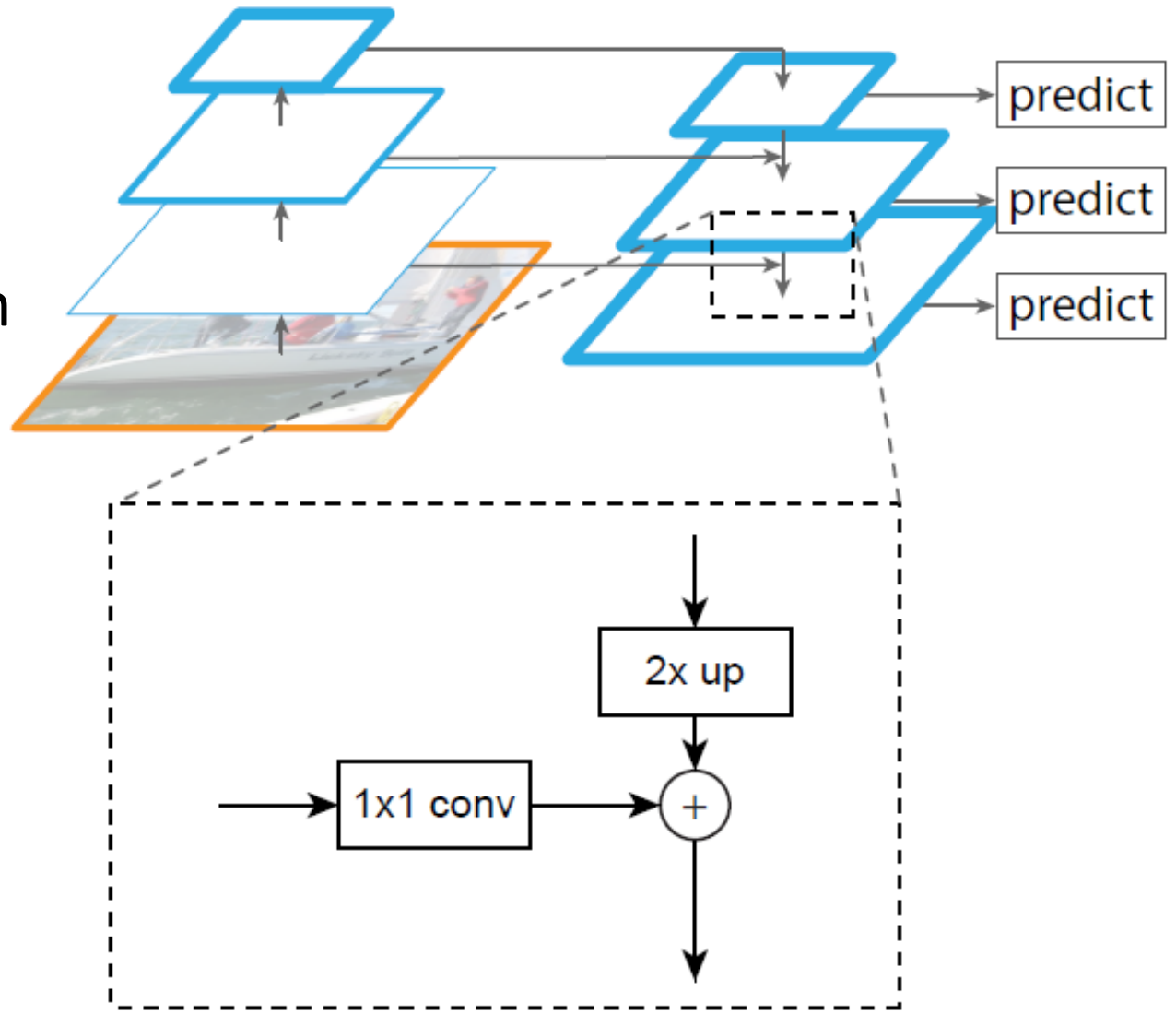
(c) Pyramidal feature hierarchy



(d) Feature Pyramid Network

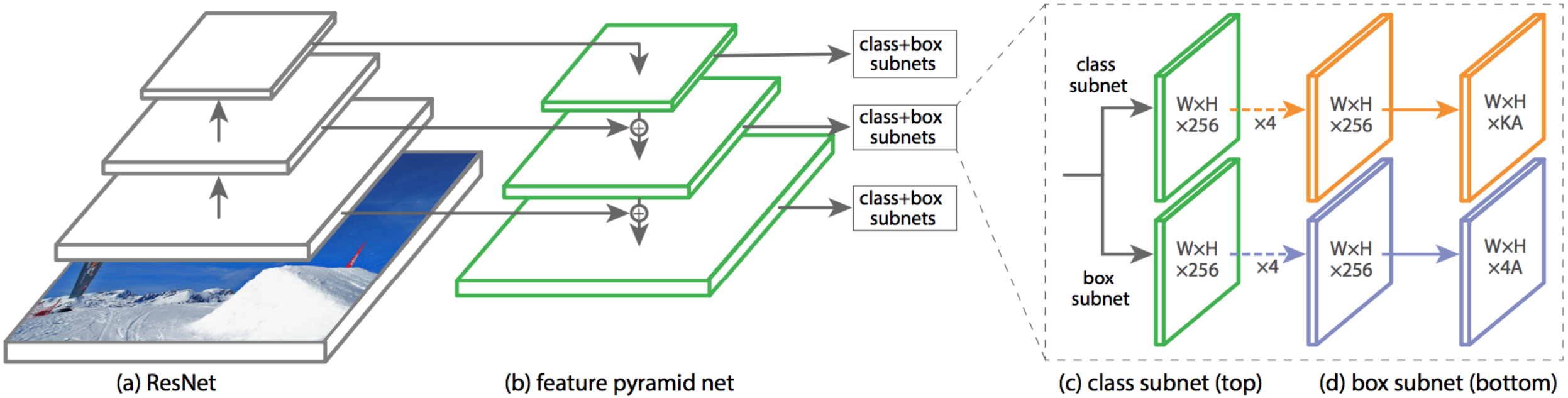
FPN (Top-Down)

- FPN is accurate but slow
- FPN-based Faster R-CNN system on a single NVIDIA M40 GPU
  - 6.76 FPS for ResNet-50
  - 5.81 FPS for ResNet-101.



RetinaNet

2018	**Focal Loss for Dense Object Detection**
Tsung-Yi Lin Priya Goyal Ross Girshick Kaiming He Piotr Dollár Facebook AI Research (FAIR)	



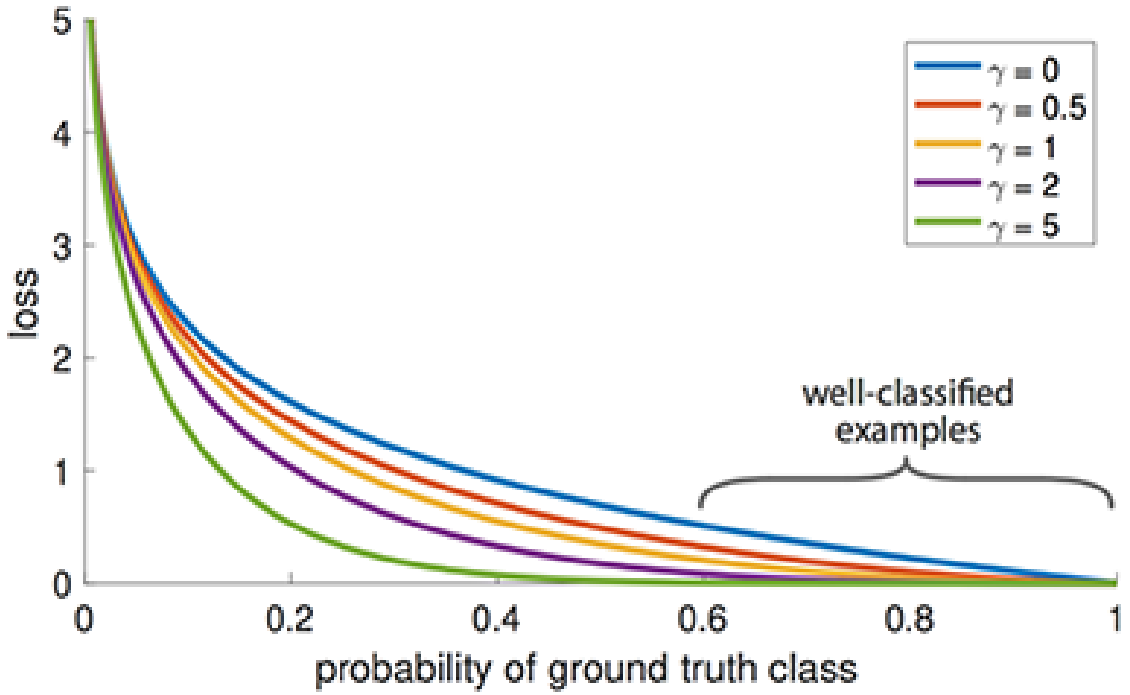
(a) ResNet (b) feature pyramid net (c) class subnet (top) (d) box subnet (bottom)

https://arxiv.org/pdf/1708.02002.pdf

Focal Loss

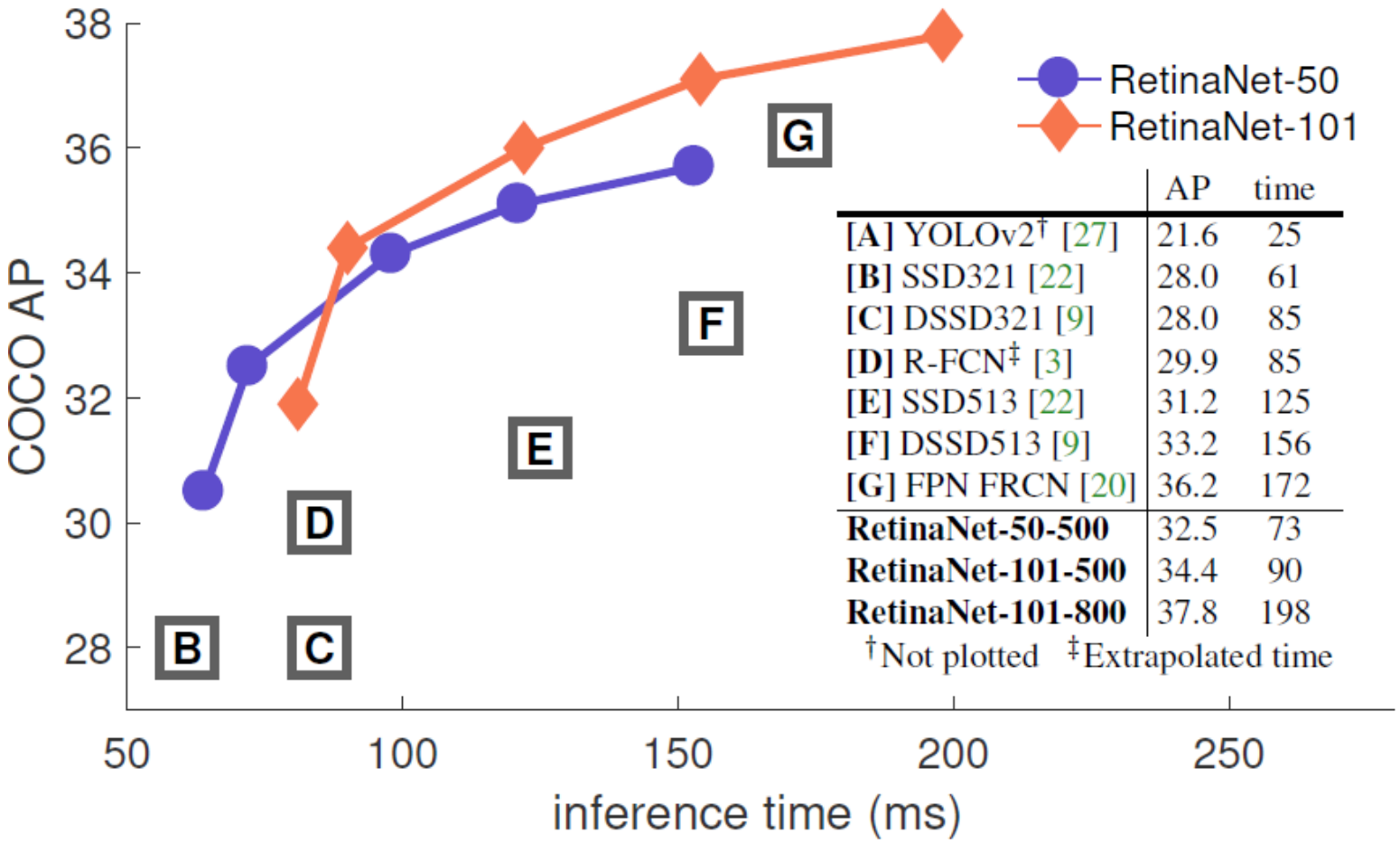
- Solve class imbalance problem by reducing loss for well-trained class

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



RetinaNet Performance

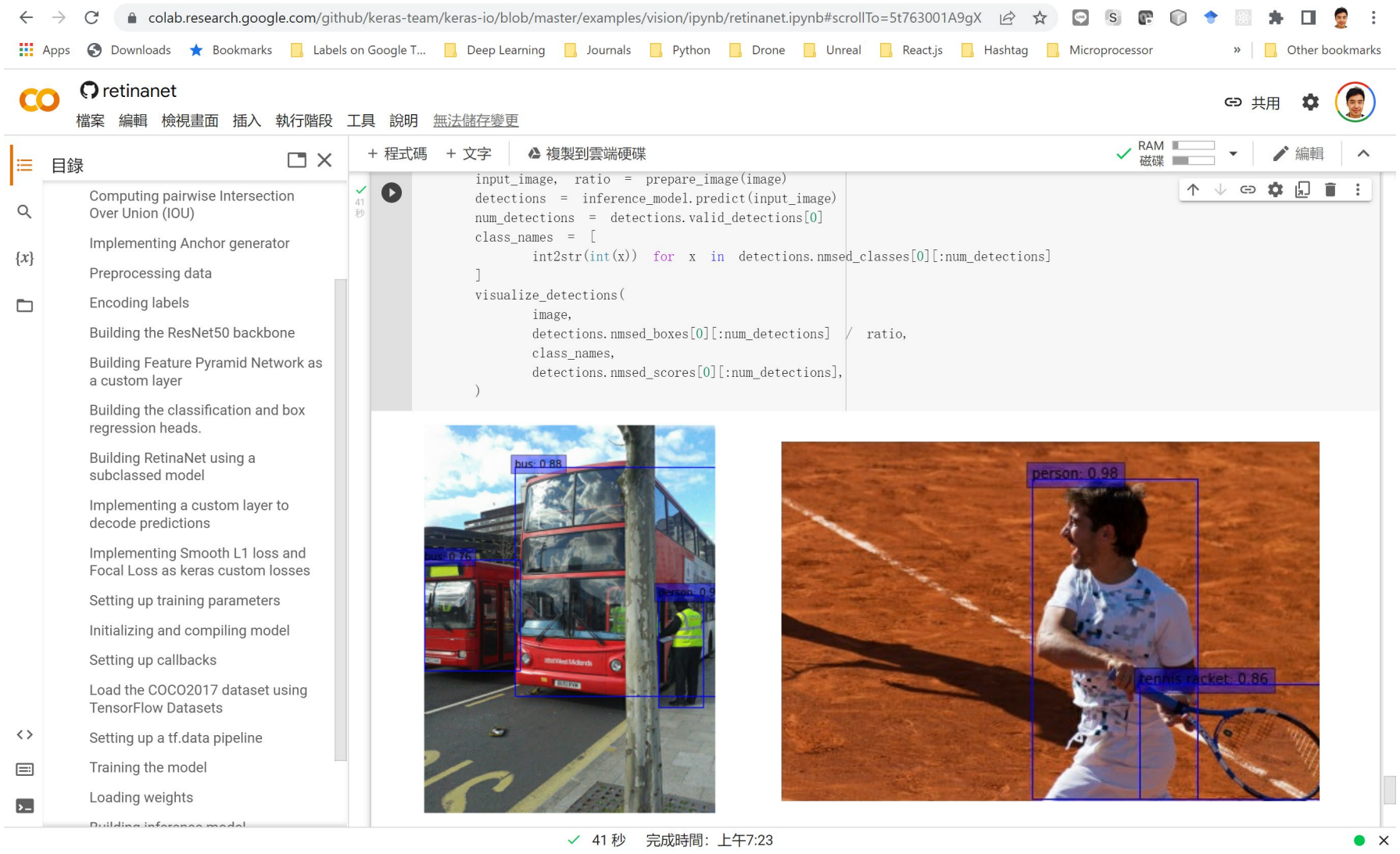
- Nvidia M40



	AP	time
[A] YOLOv2† [27]	21.6	25
[B] SSD321 [22]	28.0	61
[C] DSSD321 [9]	28.0	85
[D] R-FCN‡ [3]	29.9	85
[E] SSD513 [22]	31.2	125
[F] DSSD513 [9]	33.2	156
[G] FPN FRCN [20]	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	37.8	198

†Not plotted ‡Extrapolated time

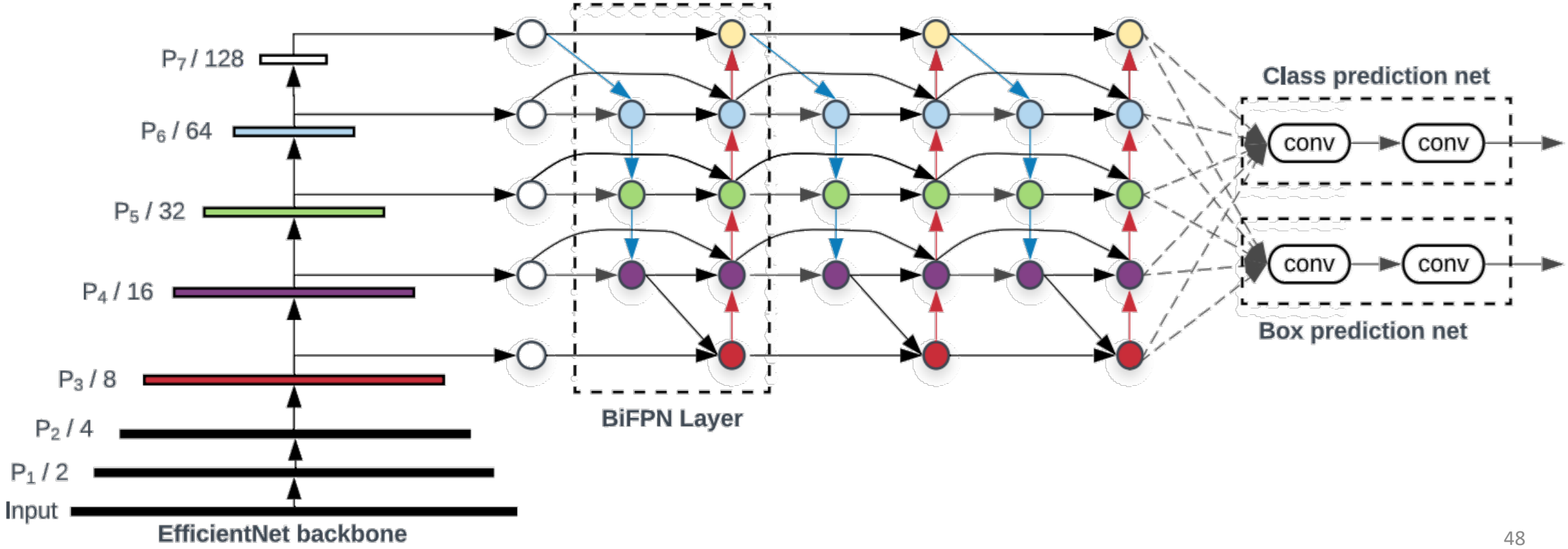
Keras: Object Detection with RetinaNet



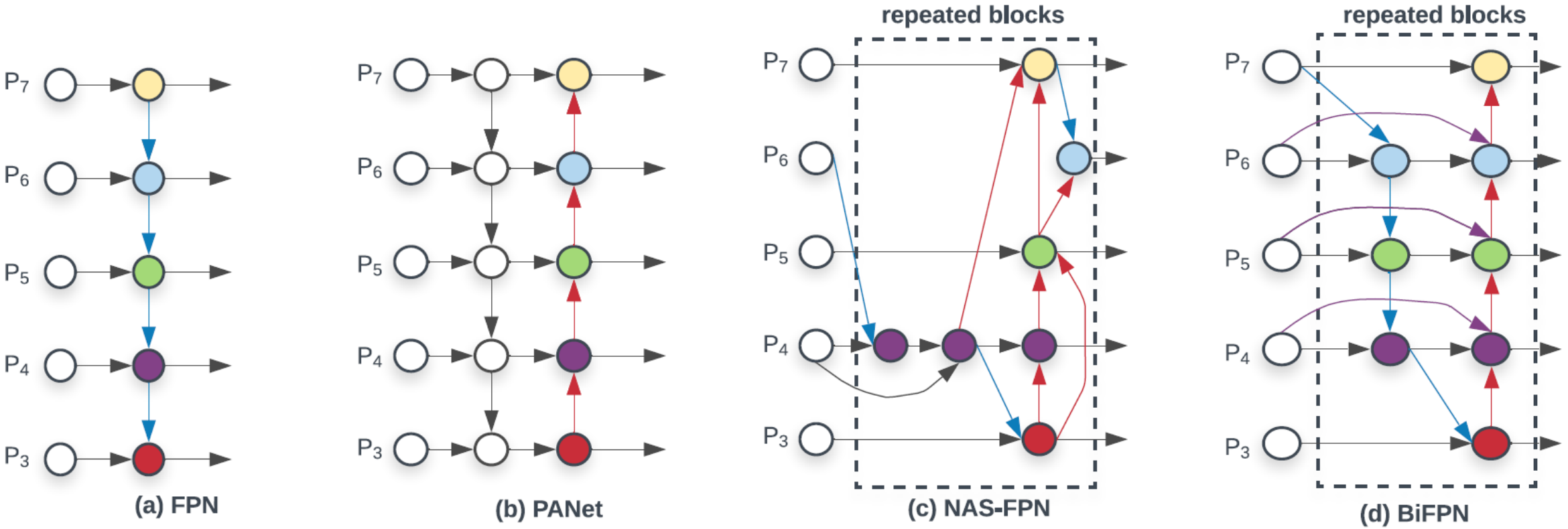
https://keras.io/examples/vision/retinanet/

EfficientDet (2020)

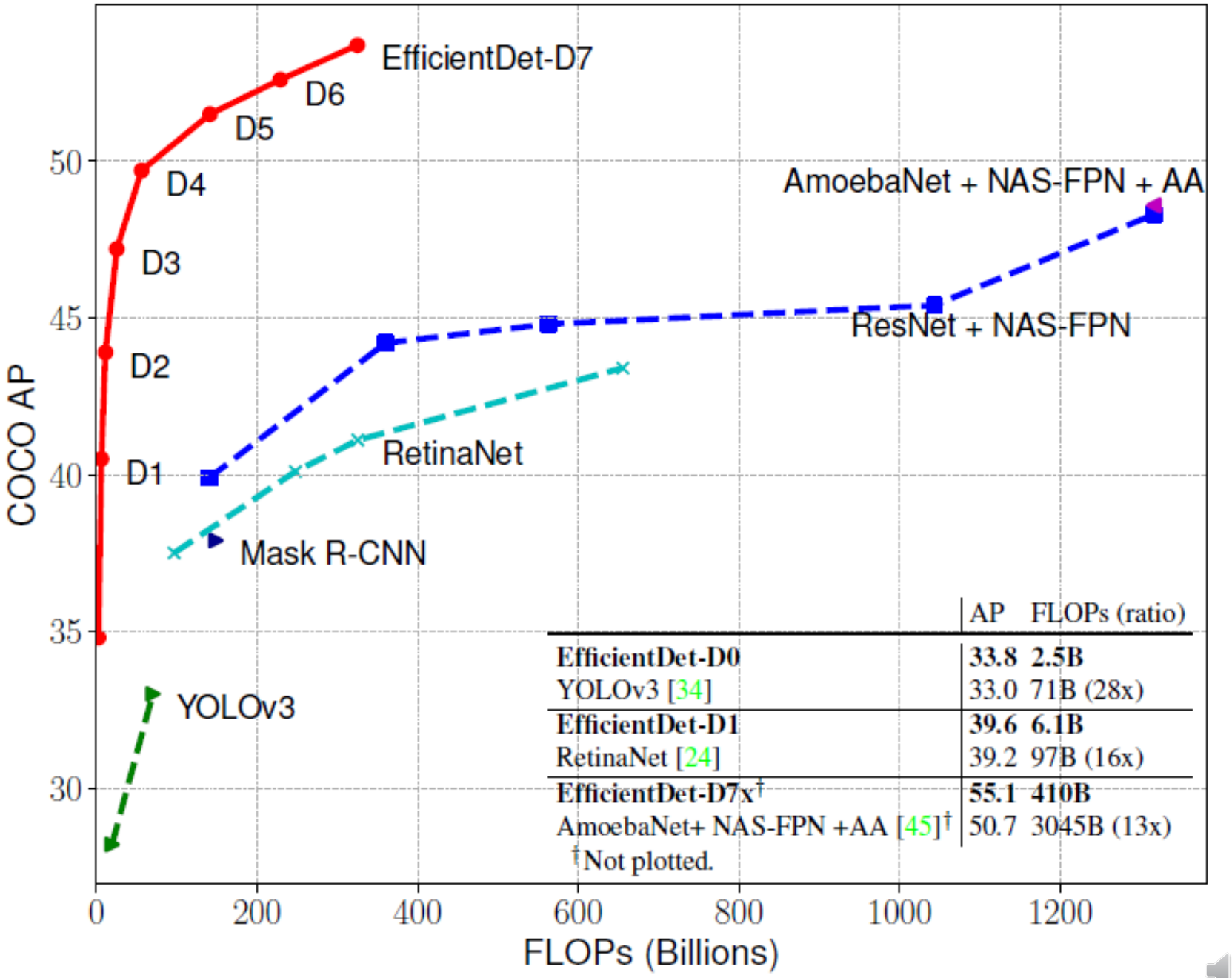
- Based on EfficientNet
  - Mingxing Tan Ruoming Pang Quoc V. Le, "EfficientDet: Scalable and Efficient Object Detection", Google Research, Brain Team



Feature Network Design



Performance of EfficientDet on MS COCO

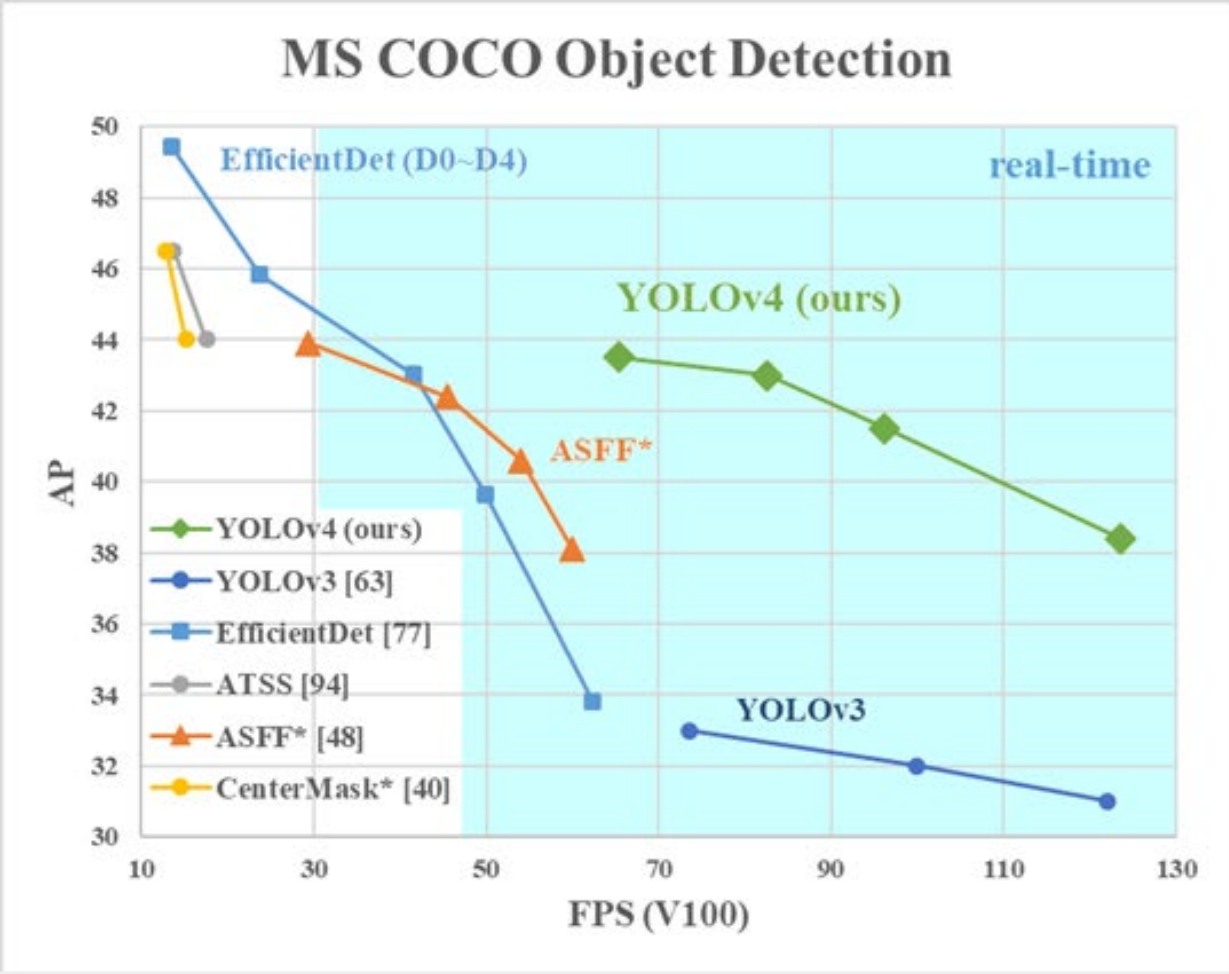


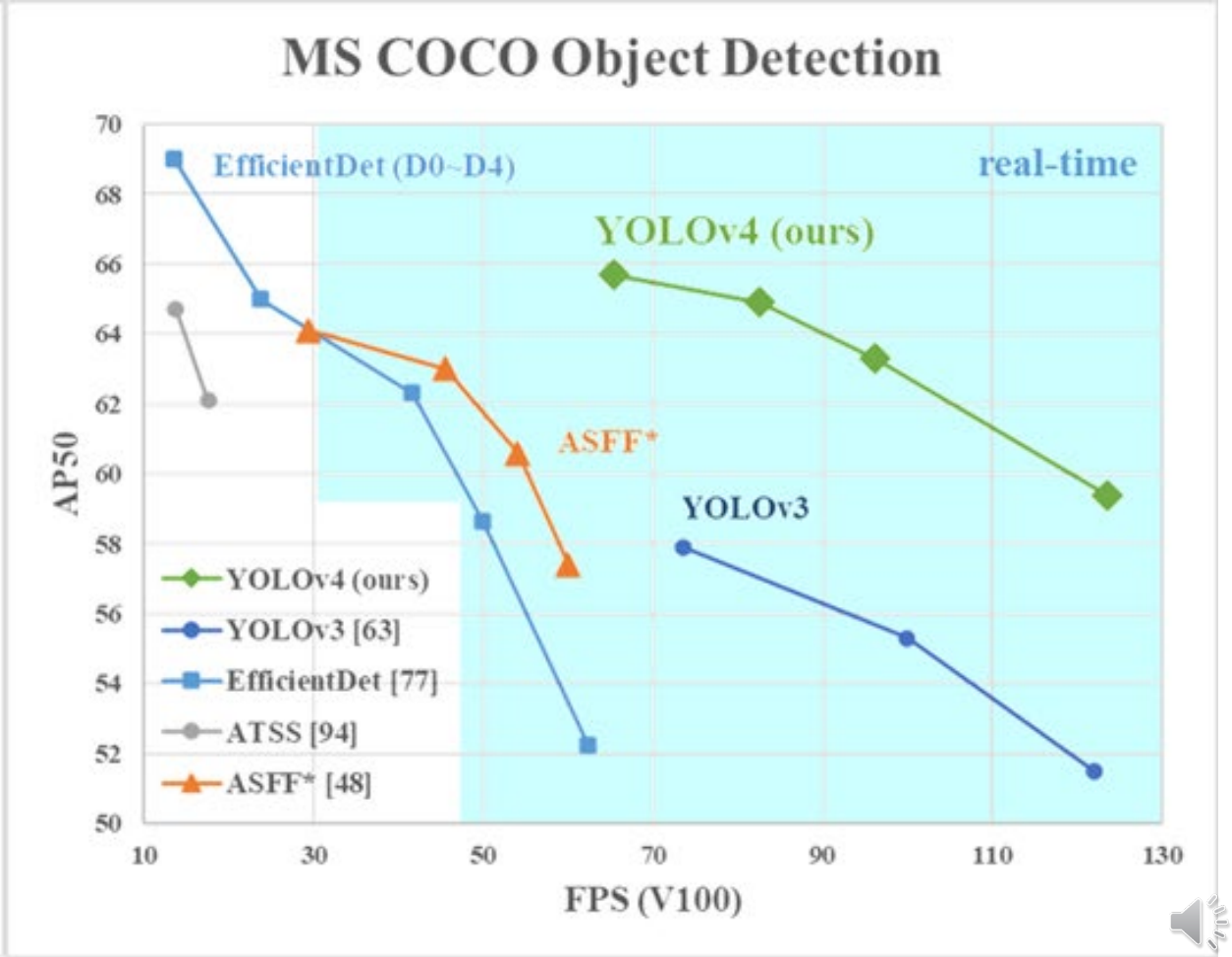
	AP	FLOPs (ratio)
EfficientDet-D0	**33.8**	**2.5B**
YOLOv3 [34]	33.0	71B (28x)
EfficientDet-D1	**39.6**	**6.1B**
RetinaNet [24]	39.2	97B (16x)
EfficientDet-D7x†	**55.1**	**410B**
AmoebaNet+ NAS-FPN +AA [45]†	50.7	3045B (13x)

†Not plotted.

YOLO v4

- A. Bochkovskiy, C.-Y. Wang, H.-Y. Mark Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection”, 2020
- https://github.com/AlexeyAB/darknet

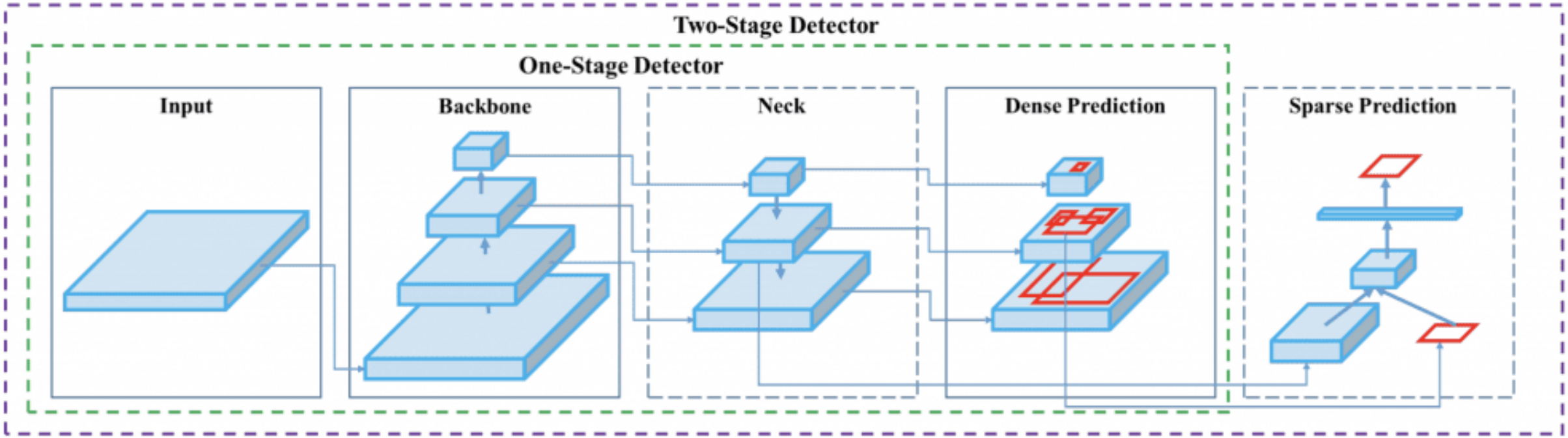




Me and YOLO v4 Authors

- From left to right
  - Me, Dr. Mark Liao, Dr. Wang

Architecture of Modern Object Detectors



Input: { Image, Patches, Image Pyramid, … }

Backbone: { VGG16 [68], ResNet-50 [26], ResNeXt-101 [86], Darknet53 [63], … }

Neck: { FPN [44], PANet [49], Bi-FPN [77], … }

Head:

Dense Prediction: { RPN [64], YOLO [61, 62, 63], SSD [50], RetinaNet [45], FCOS [78], … }

Sparse Prediction: { Faster R-CNN [64], R-FCN [9], … }

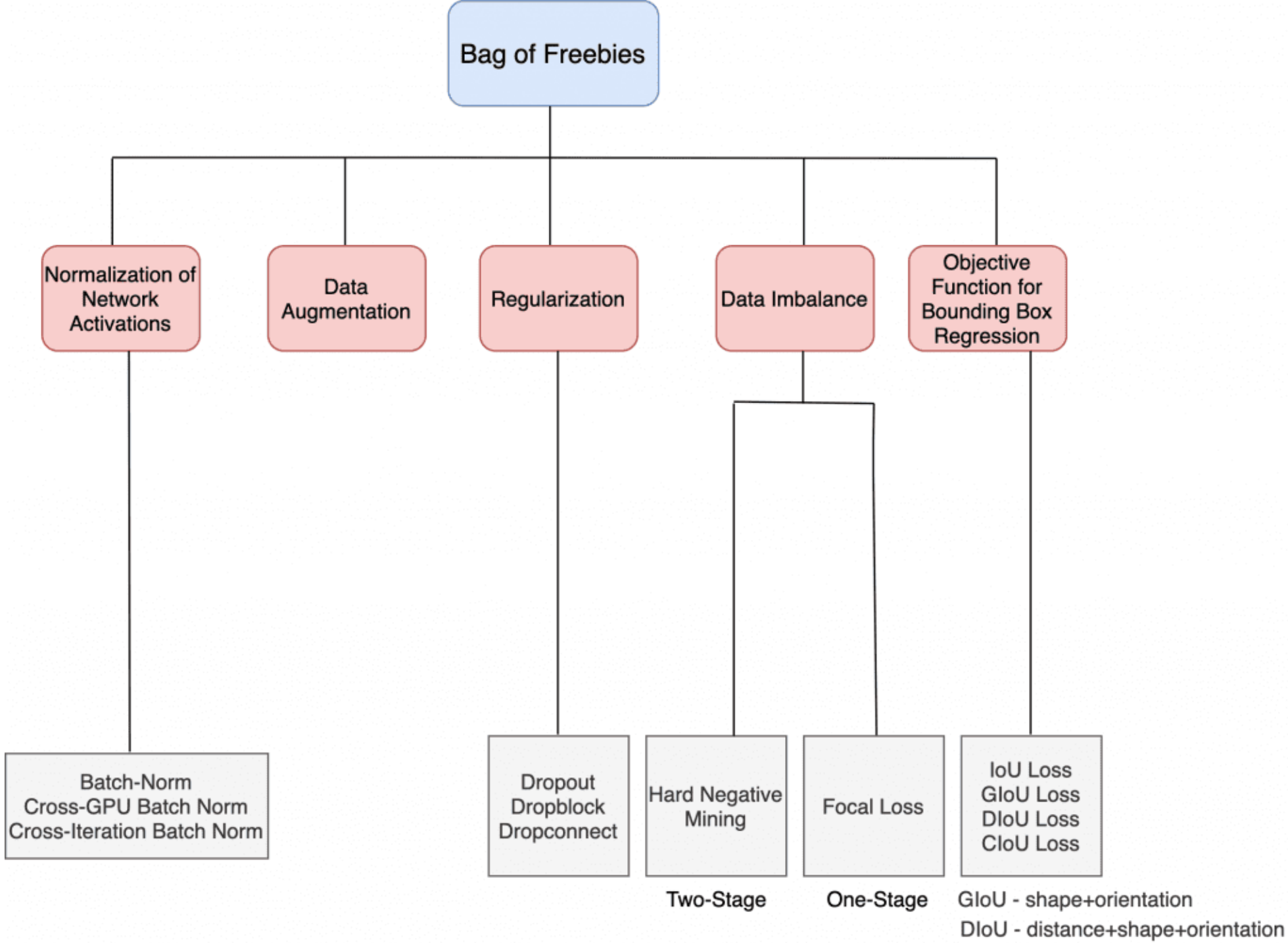
Object Detection Workflow (**Bochkovskiy et al., 2020**)

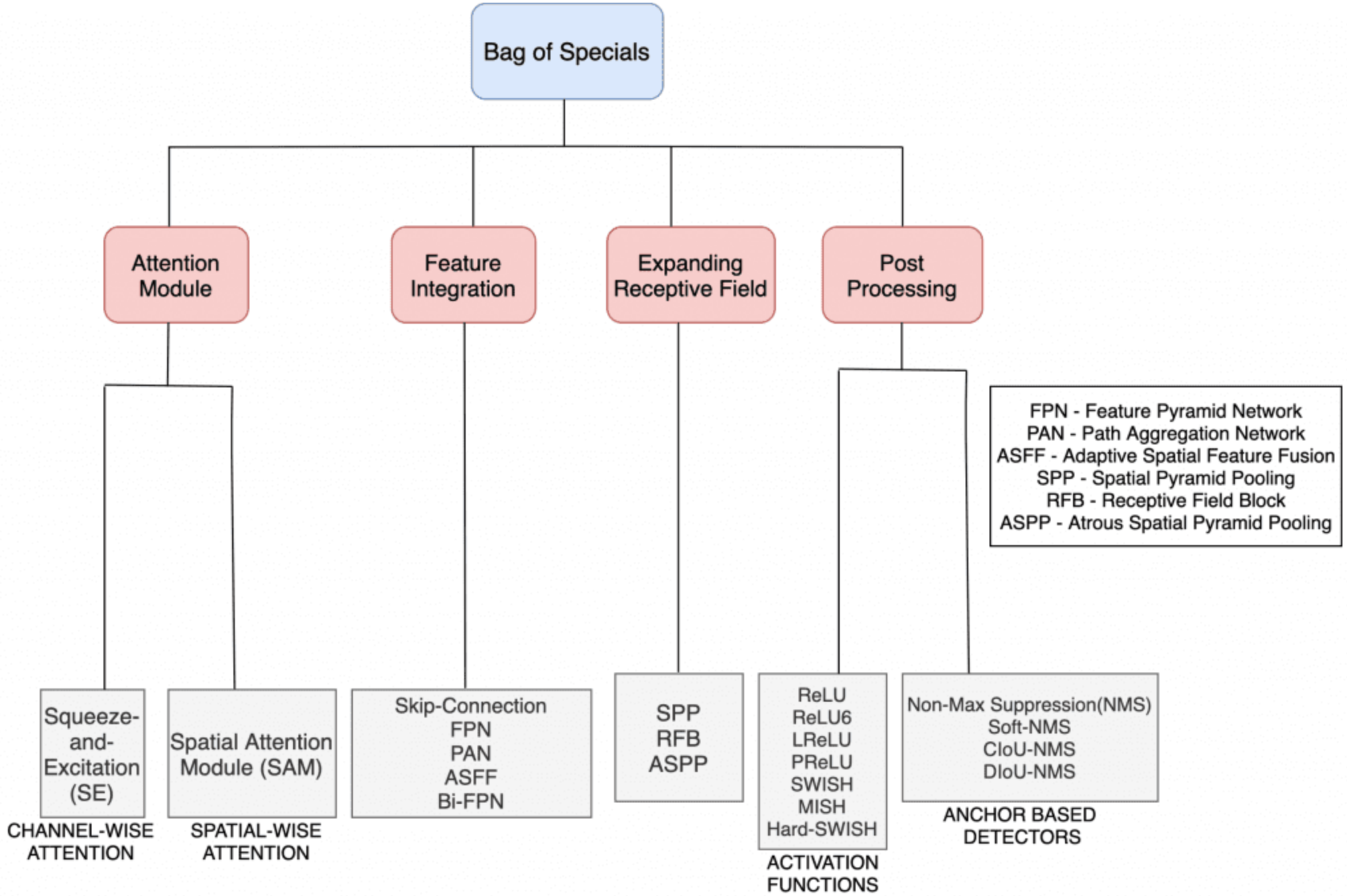
New Techniques Adopted in YOLO v4

- Bag of Freebies (Training)
  - Self-adversarial Training (SAT)
  - Data augmentation (cutmix, mixup, mosaic,…)
  - Cross mini-Batch, Normalization (CmBN)

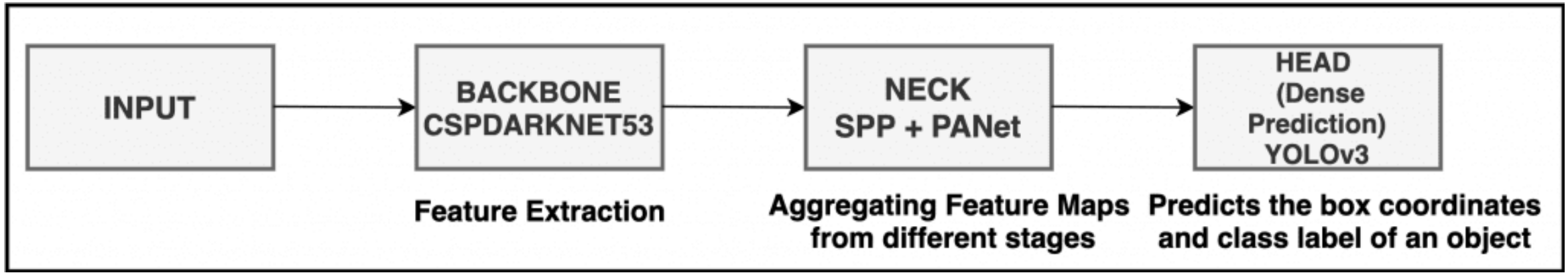
- Bag of Specials
  - Weighted Residual Connections (WRC)
  - Cross-Stage Partial Connections (CSP)
  - Mish-activation
  - …

https://pyimagesearch.com/2022/05/16/achieving-optimal-speed-and-accuracy-in-object-detection-yolov4/


Bag of Freebies
Normalization of Network Activations
Data Augmentation
Regularization
Data Imbalance
Objective Function for Bounding Box Regression
Batch-Norm
Cross-GPU Batch Norm
Cross-Iteration Batch Norm
Dropout
Dropblock
Dropconnect
Hard Negative Mining
Focal Loss
IoU Loss
GIoU Loss
DIoU Loss
CIoU Loss
Two-Stage
One-Stage
GIoU - shape+orientation
DIoU - distance+shape+orientation


Bag of Specials
Attention Module
Feature Integration
Expanding Receptive Field
Post Processing
FPN - Feature Pyramid Network
PAN - Path Aggregation Network
ASFF - Adaptive Spatial Feature Fusion
SPP - Spatial Pyramid Pooling
RFB - Receptive Field Block
ASPP - Atrous Spatial Pyramid Pooling
Squeeze-and-Excitation (SE)
Spatial Attention Module (SAM)
Skip-Connection
FPN
PAN
ASFF
Bi-FPN
SPP
RFB
ASPP
ReLU
ReLU6
LReLU
PReLU
SWISH
MISH
Hard-SWISH
Non-Max Suppression(NMS)
Soft-NMS
CIoU-NMS
DIoU-NMS
CHANNEL-WISE ATTENTION
SPATIAL-WISE ATTENTION
ACTIVATION FUNCTIONS
ANCHOR BASED DETECTORS

Final Architecture of YOLO v4



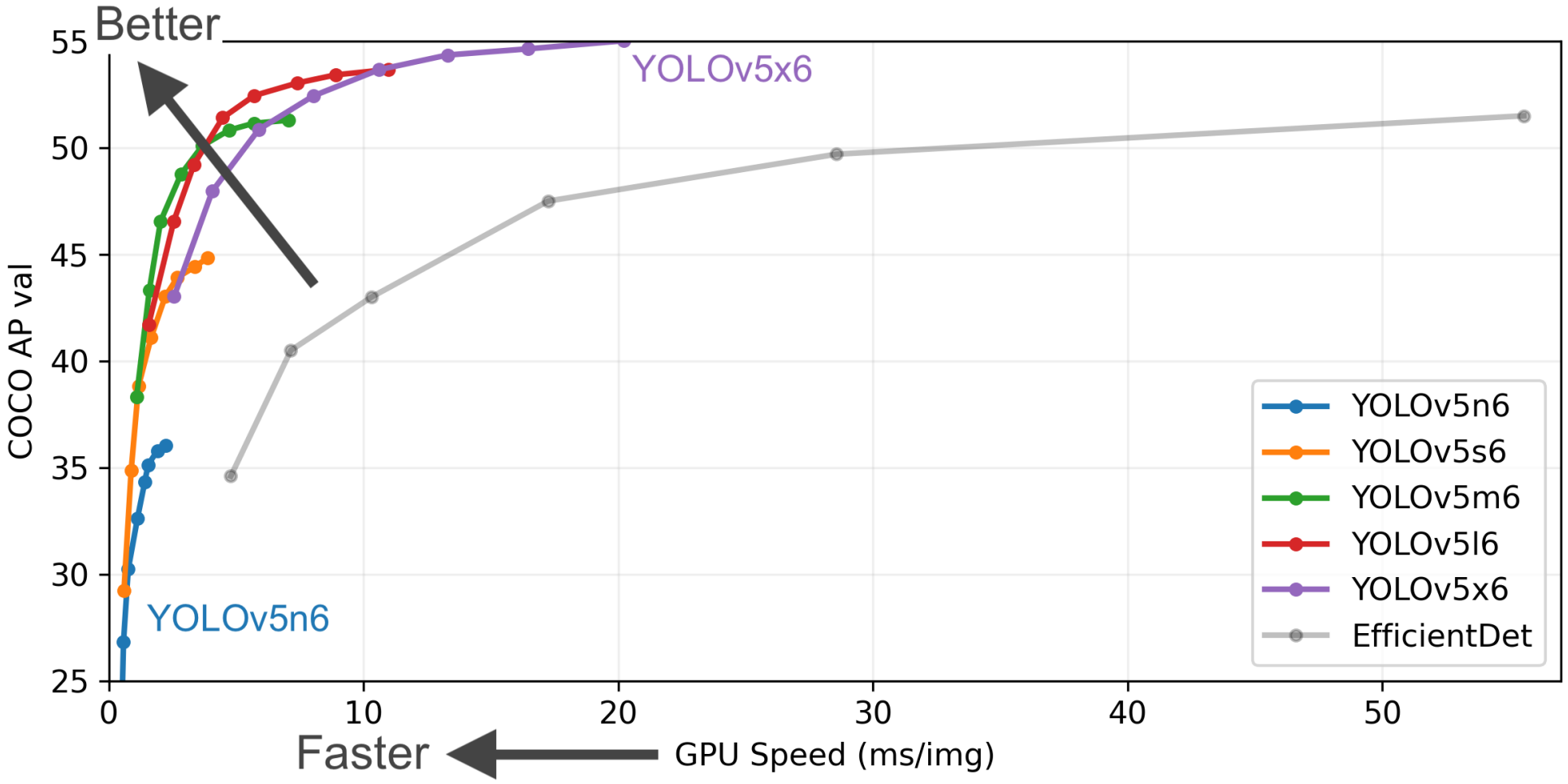
YOLOv5 by

Download on the App Store

Coming Soon on Google Play

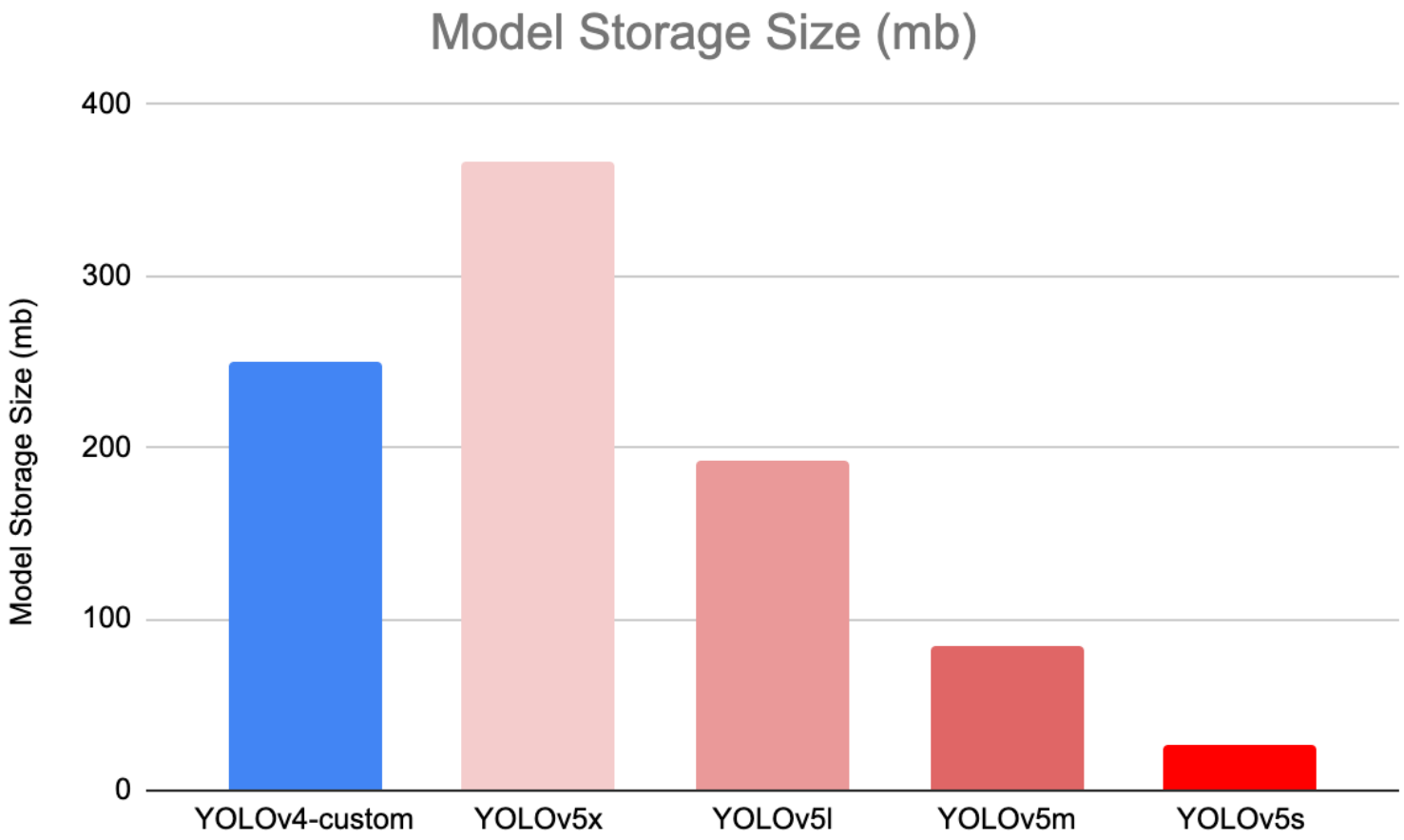
YOLO v5

- No published paper. Created by Glenn Jocher. Implemented in PyTorch
- https://github.com/ultralytics/yolov5



YOLO v4 vs. YOLO v5

- https://blog.roboflow.com/yolov4-versus-yolov5/
- Similar accuracy but smaller model size

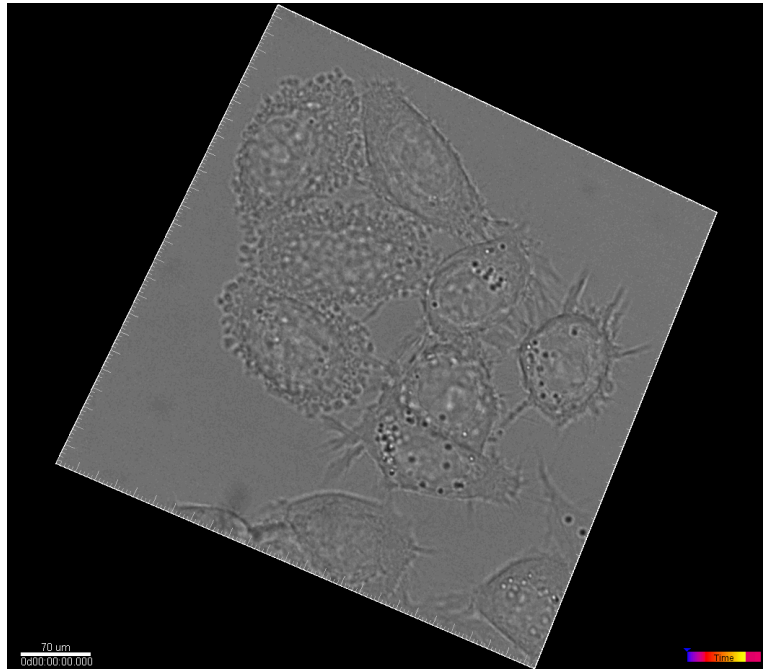


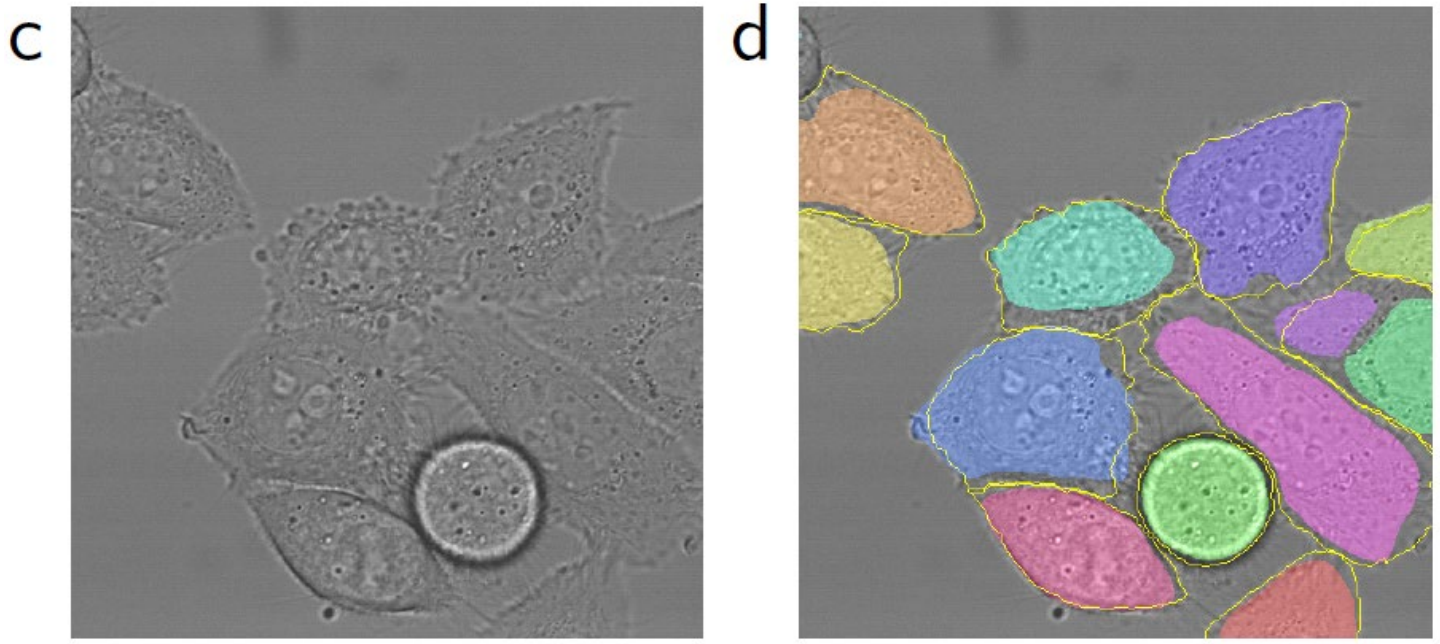
Segmentation

https://www.analyticsvidhya.com/blog/2019/07/computer-vision-implementing-mask-r-cnn-image-segmentation/

U-Net: CNN for Biomedical Image Segmentation

- Ronneberger et al. 2015
- One of the best models for image segmentation





https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/

U-Net Architecture

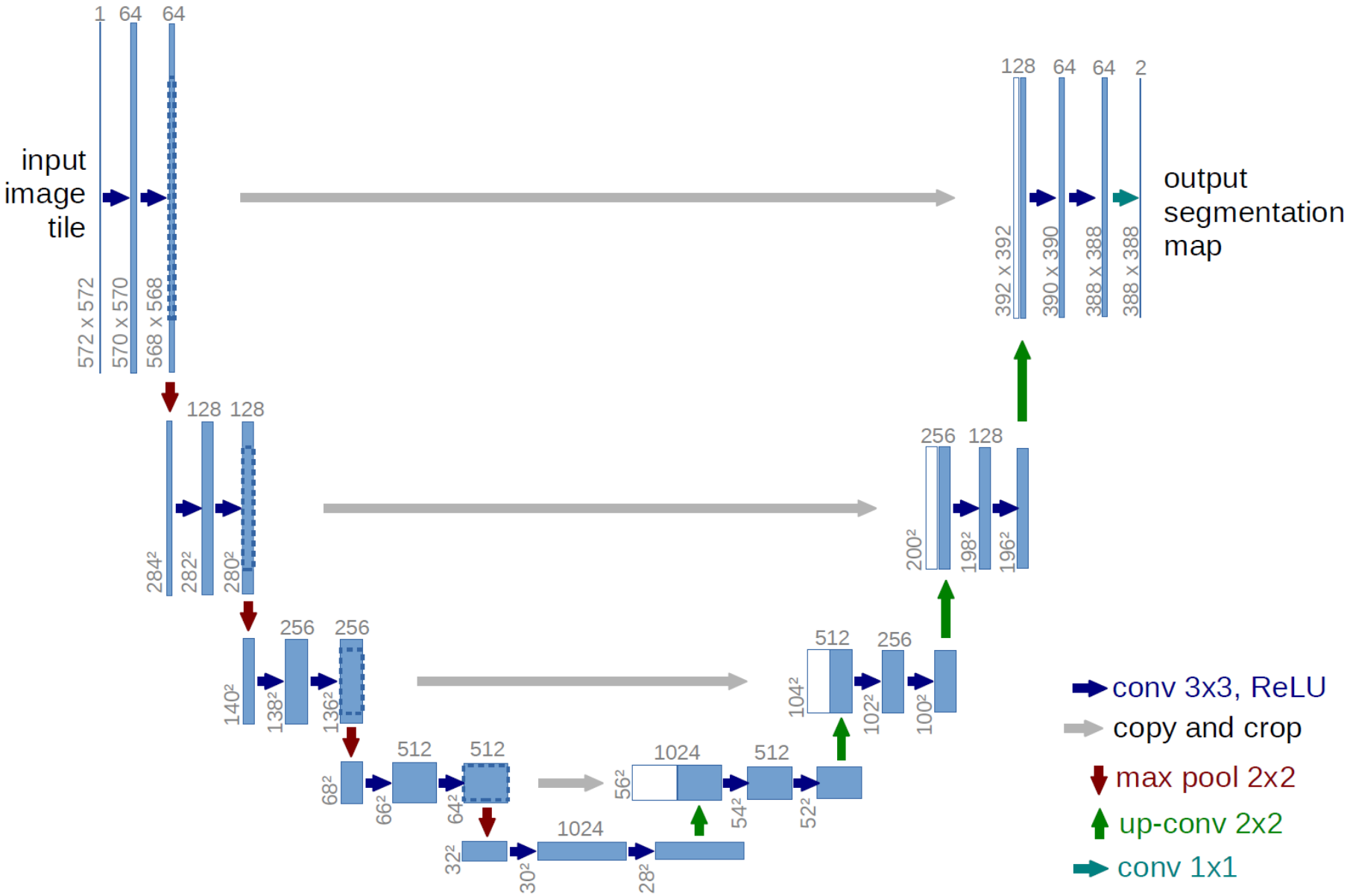
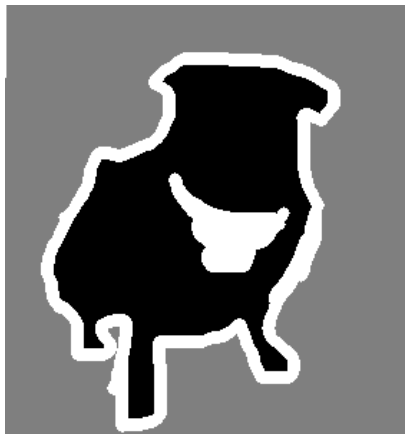


Image segmentation with a U-Net-like Model

- F. Chollet, https://keras.io/examples/vision/oxford_pets_image_segmentation/

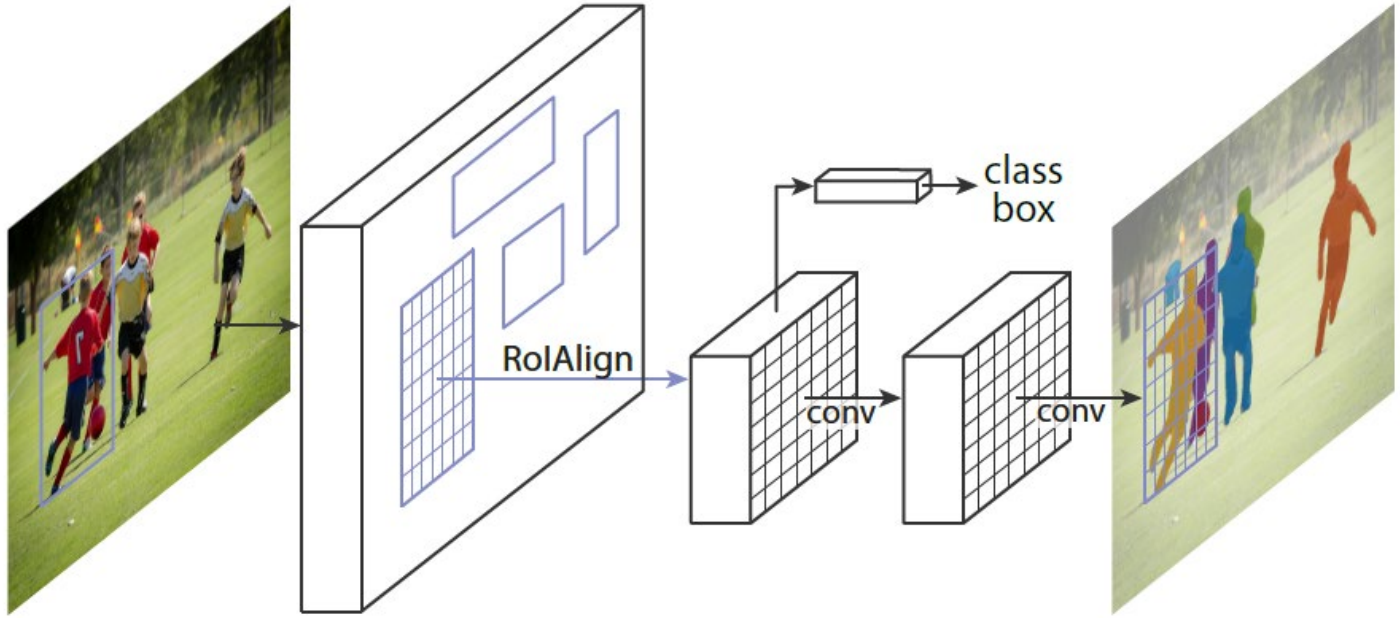


Mask R-CNN (2018)

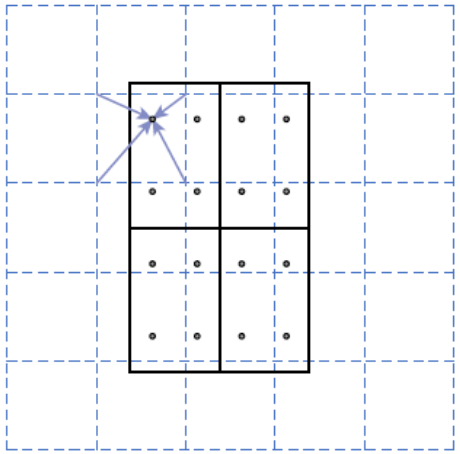
Mask R-CNN

Kaiming He Georgia Gkioxari Piotr Dollár Ross Girshick
Facebook AI Research (FAIR)

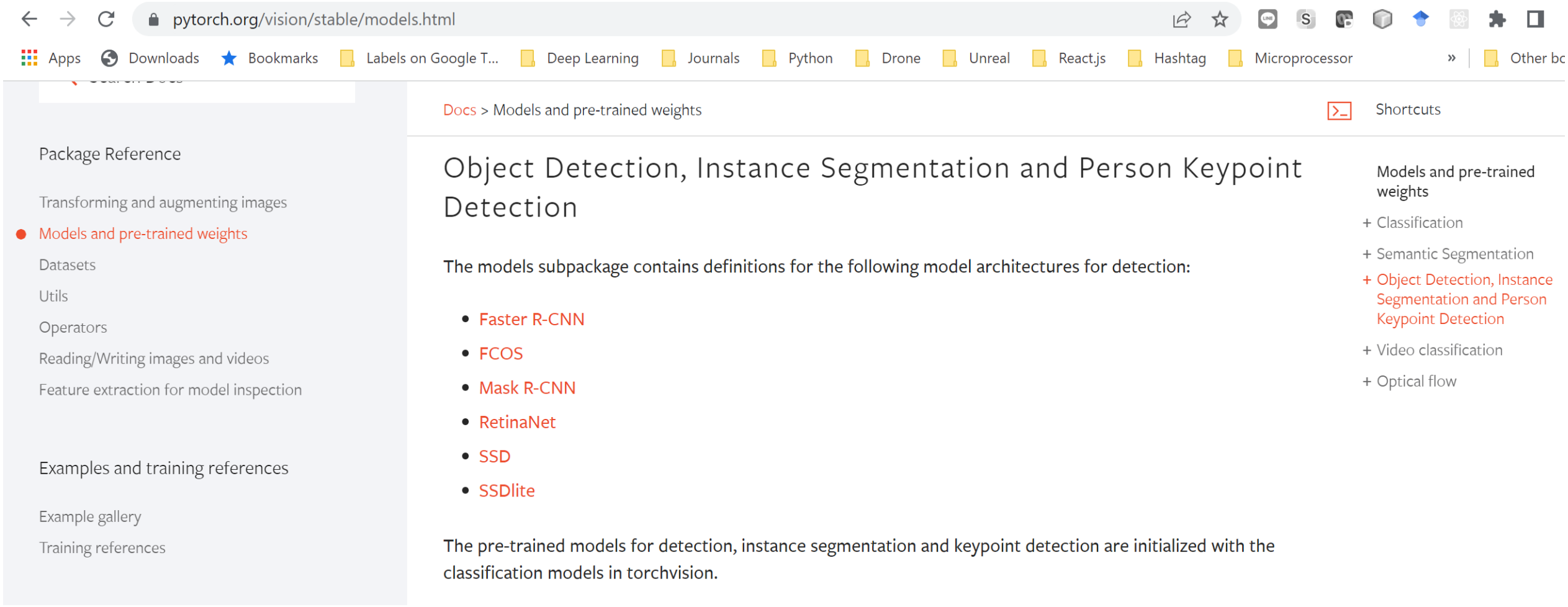
- Extend Faster R-CNN by adding a branch for predicting an object mask



RoI Align



Torch Vision

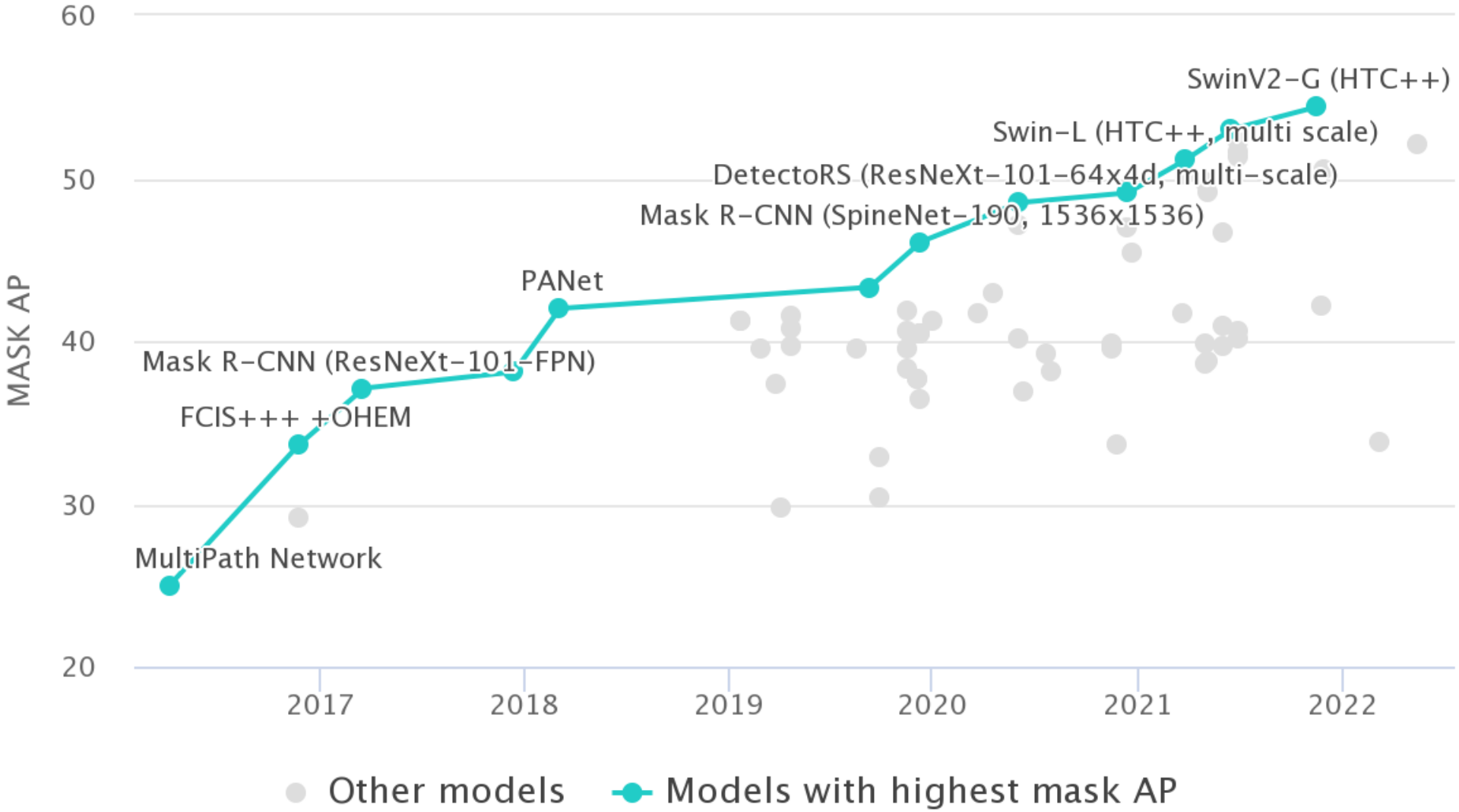


https://pytorch.org/vision/stable/models.html#object-detection-instance-segmentation-and-person-keypoint-detection

TensorFlow Mask R-CNN

https://github.com/matterport/Mask_RCNN.git

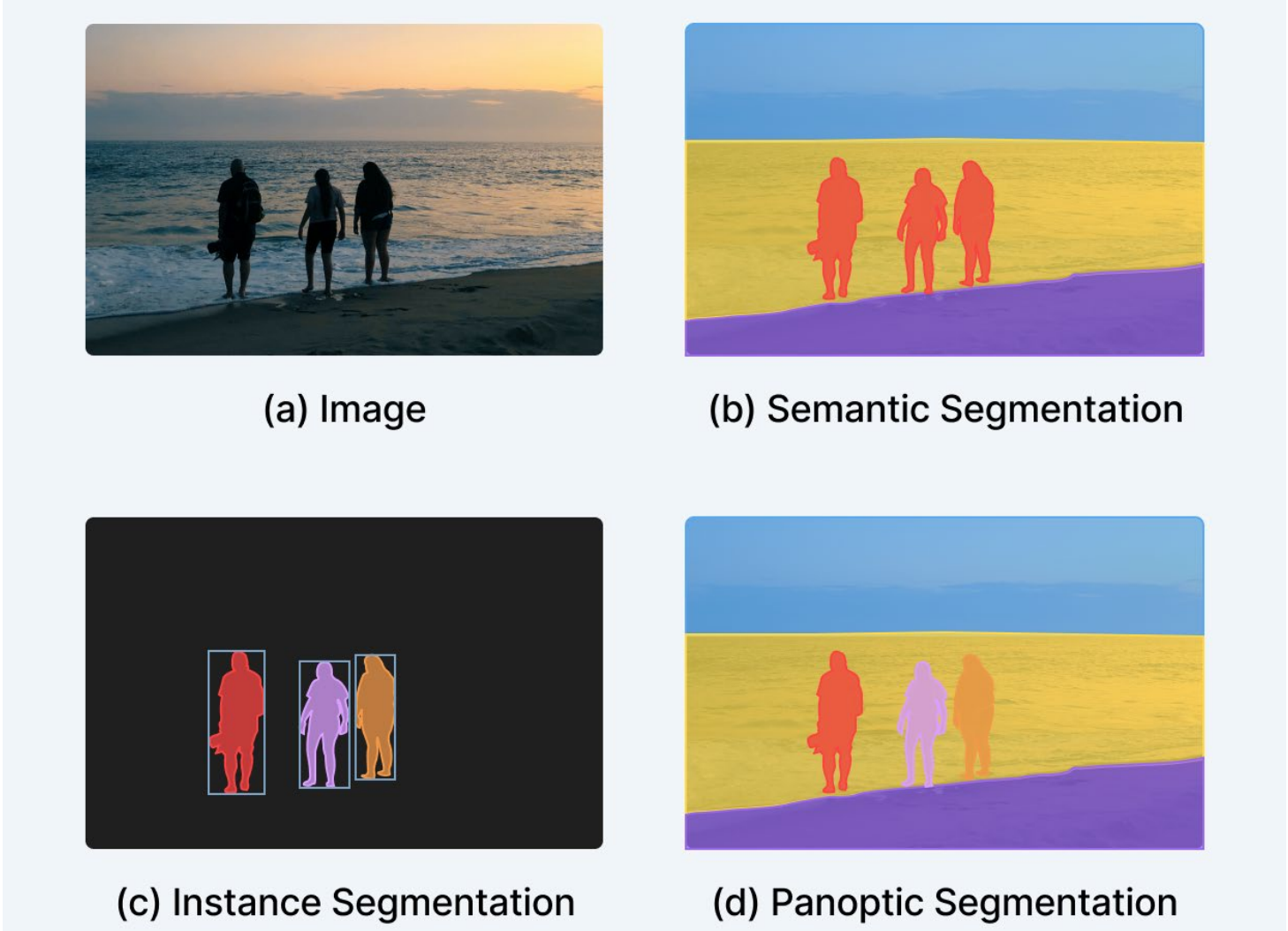
Instance Segmentation on COCO test-dev



https://paperswithcode.com/sota/instance-segmentation-on-coco

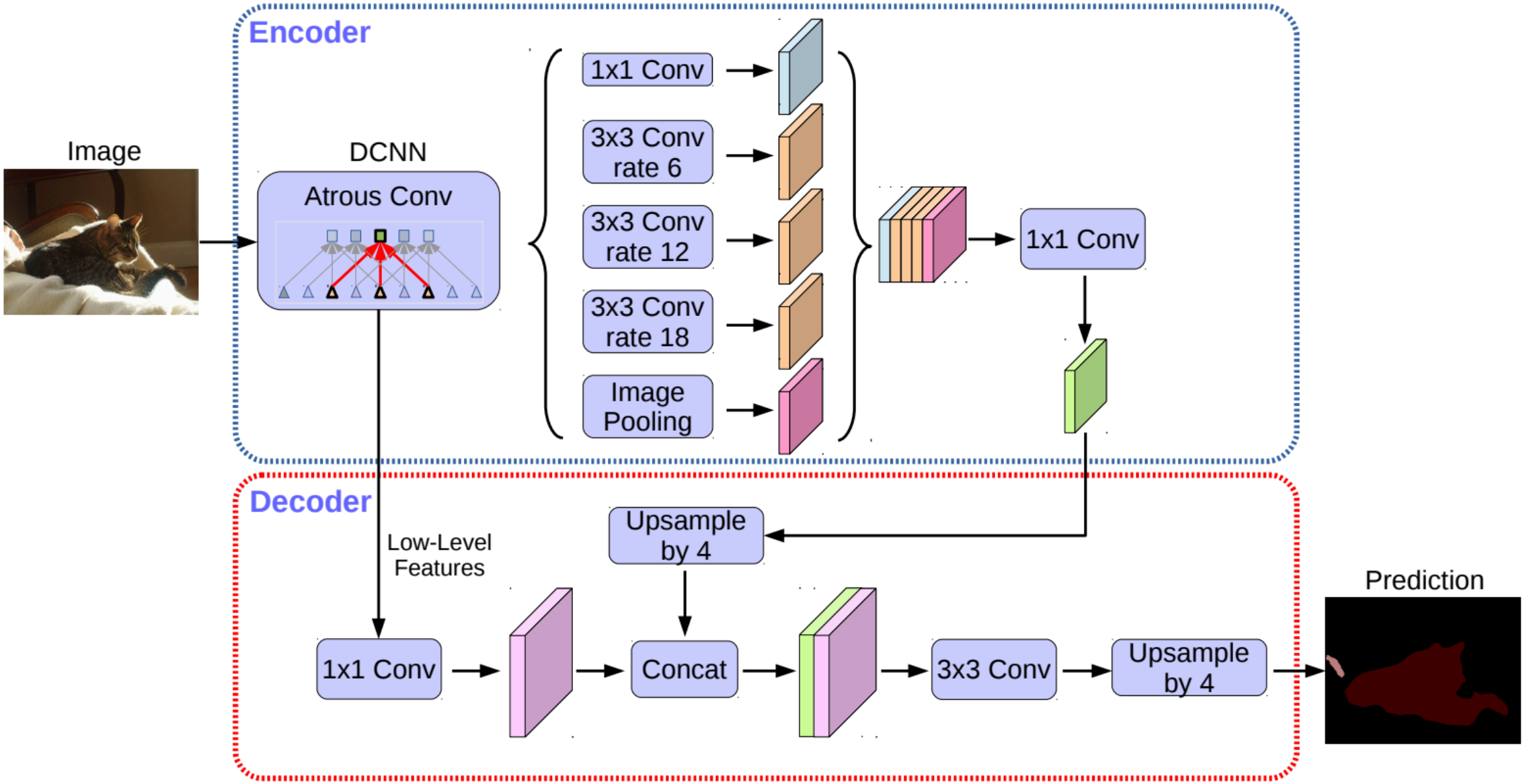
Semantic, Instance, Panoptic Segmentation

- https://www.v7labs.com/blog/panoptic-segmentation-guide

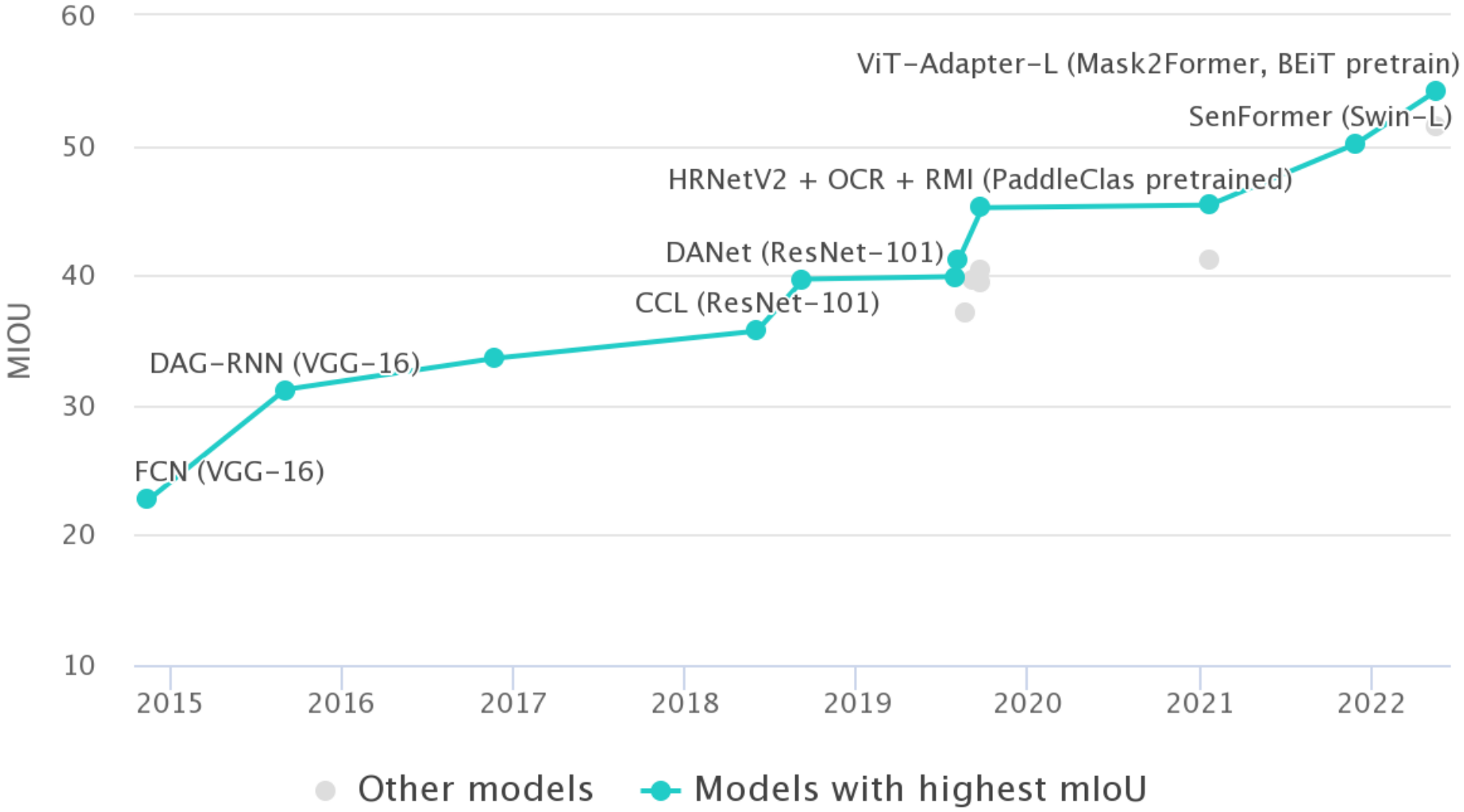


DeepLab V3+

- https://keras.io/examples/vision/deeplabv3_plus/

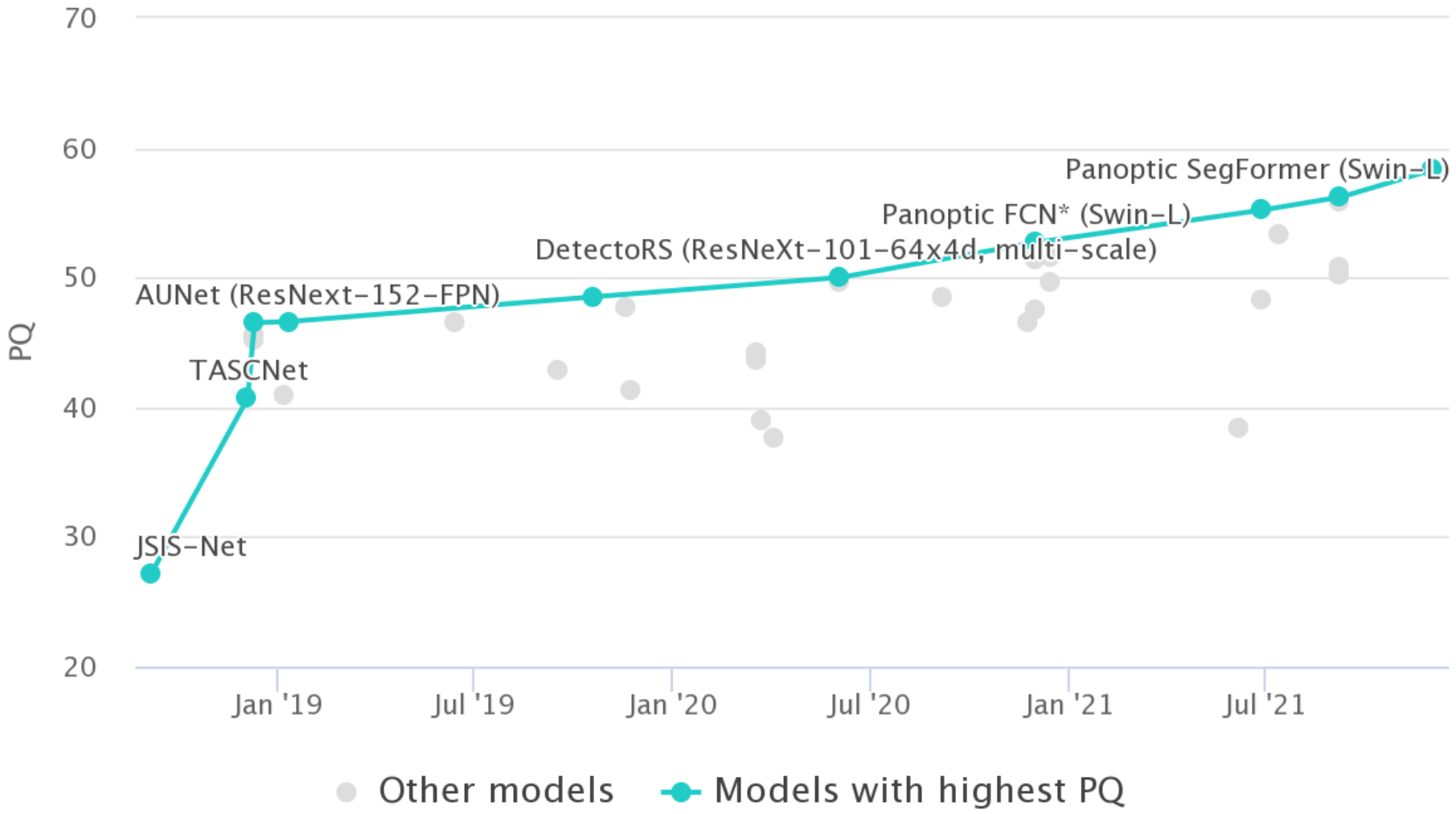


Semantic Segmentation on COCO-Stuff test



https://paperswithcode.com/sota/semantic-segmentation-on-coco-stuff-test

Panoptic Segmentation on COCO test-dev



https://paperswithcode.com/sota/panoptic-segmentation-on-coco-test-dev

Meta AI's Segment Anything Model (SAM)

- https://segment-anything.com/

Reference

- https://pjreddie.com/
- https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/
- https://heartbeat.fritz.ai/gentle-guide-on-how-yolo-object-localization-works-with-keras-part-2-65fe59ac12d
- https://towardsdatascience.com/retinanet-how-focal-loss-fixes-single-shot-detection-cb320e3bb0de
- https://medium.com/@jonathan_hui/what-do-we-learn-from-single-shot-object-detectors-ssd-yolo-fpn-focal-loss-3888677c5f4d
- https://pyimagesearch.com/2022/05/16/achieving-optimal-speed-and-accuracy-in-object-detection-yolov4/