UNIVERSITY OF JYVÄSKYLÄ

Lecture 5: Computer Vision (part 2)

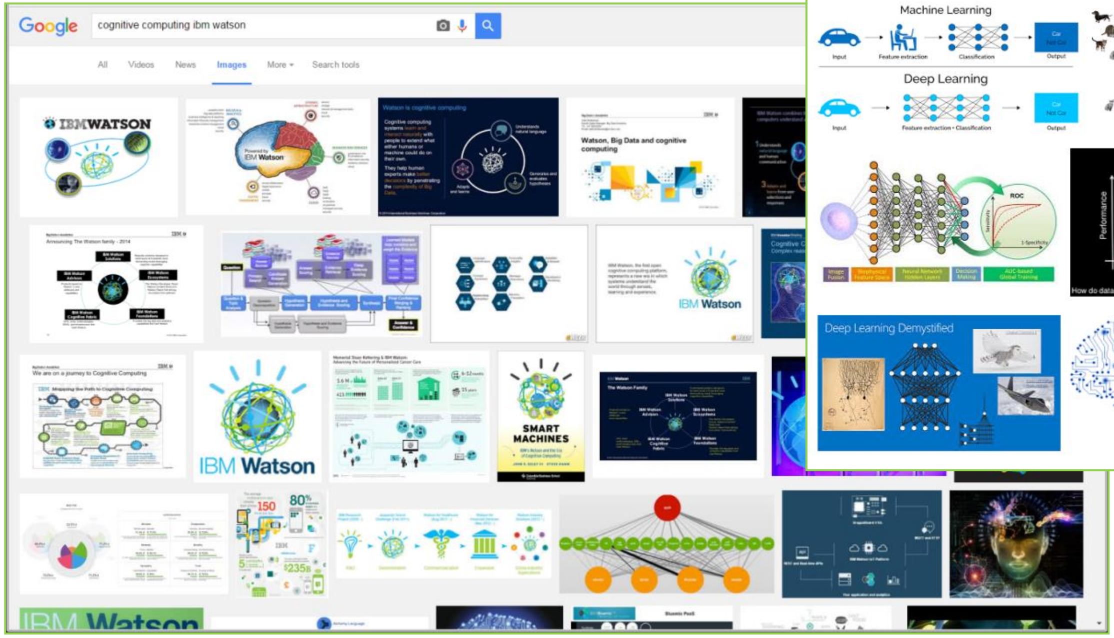
Object Detection

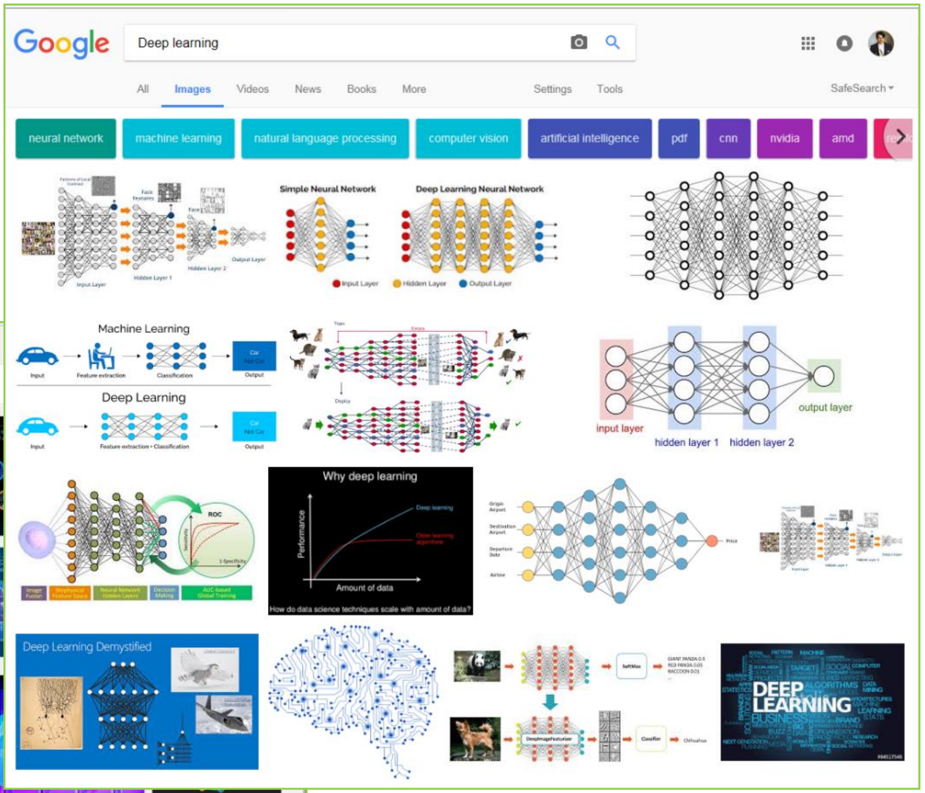
TIES4911 Deep-Learning for Cognitive Computing for Developers
Spring 2024

by:
Dr. Oleksiy Khriyenko
IT Faculty
University of Jyväskylä

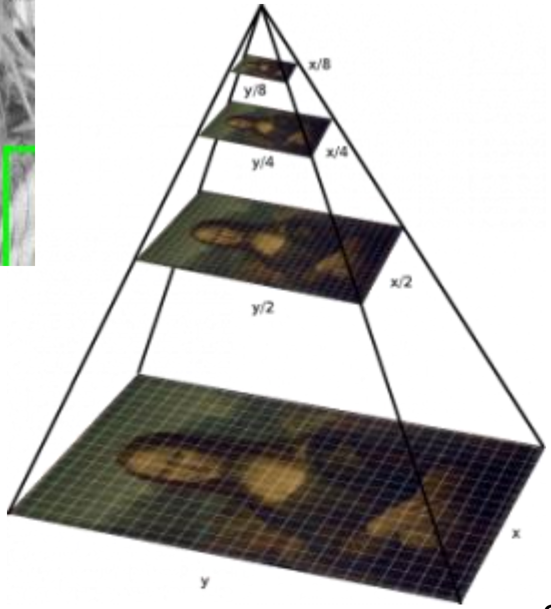
Acknowledgement

I am grateful to all the creators/owners of the images that I found from Google and have used in this presentation.





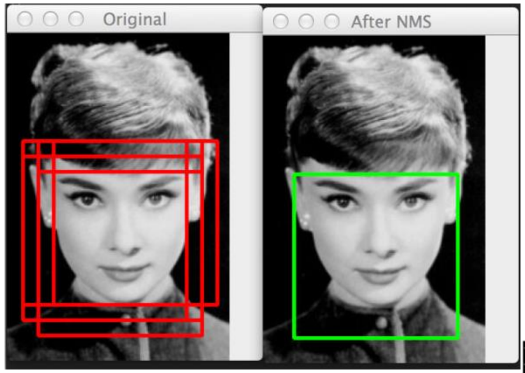
Object Detection

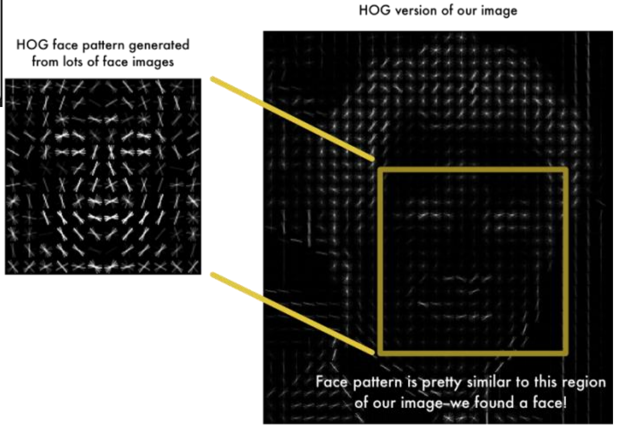


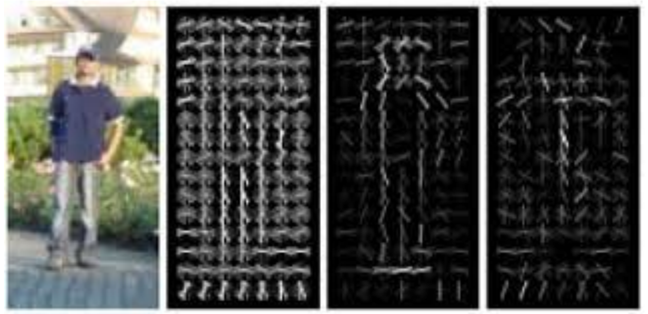
Object Detection

Before **Deep Learning** …

The ***Histogram of Oriented Gradients (HOG)*** method suggested *by Navneet Dalal* and *Bill Triggs* in their seminal 2005 paper (*http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf*) demonstrated that the Histogram of Oriented Gradients (HOG) image descriptor and a Linear Support Vector Machine (SVM) could be used to train highly accurate object classifiers and object detectors.




HOG face pattern generated from lots of face images
HOG version of our image
Face pattern is pretty similar to this region of our image–we found a face!

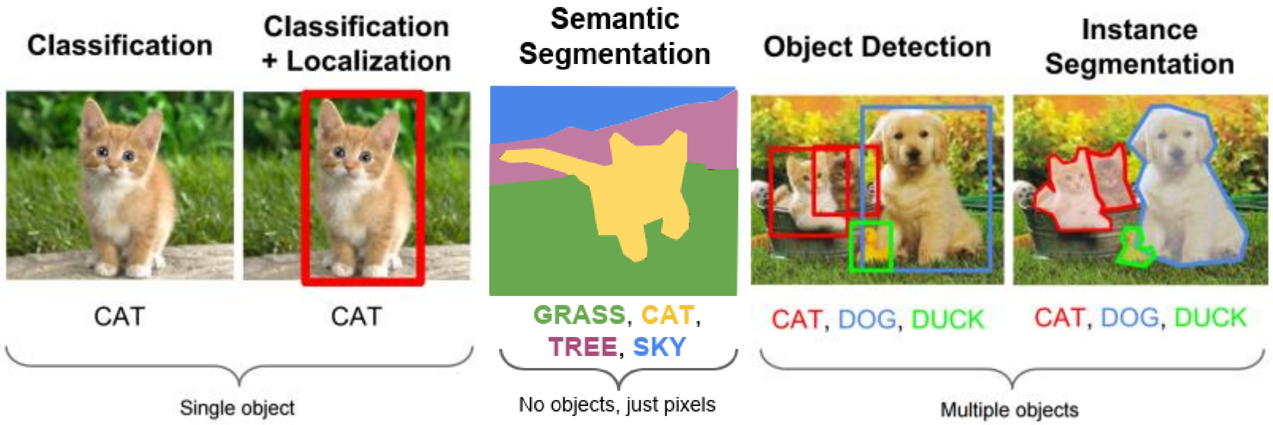


Relevant links:
https://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/
https://cs.brown.edu/~pff/papers/lsvm-pami.pdf

Object Detection

CNN is used not only for image classification tasks, but as well as for ***Object Detection***, ***Object Localization*** and ***Instance Segmentation***.

- R-CNN *(2013)*
- Fast R-CNN *(2015)*
- Faster R-CNN *(2015)*
- Mask R-CNN *(2017)*
- Yolo *(2015)*
- SSD *(2015)*
- …



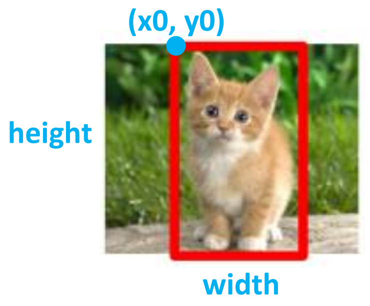
With the first ***R-CNN*** paper, *Ross Girshick and his group at UC Berkeley* created one of the most impactful advancements in computer vision. Further, ***Fast R-CNN***, ***Faster R-CNN, Mask R-CNN,*** and ***other models*** worked to make the model faster and better suited for modern object detection tasks.

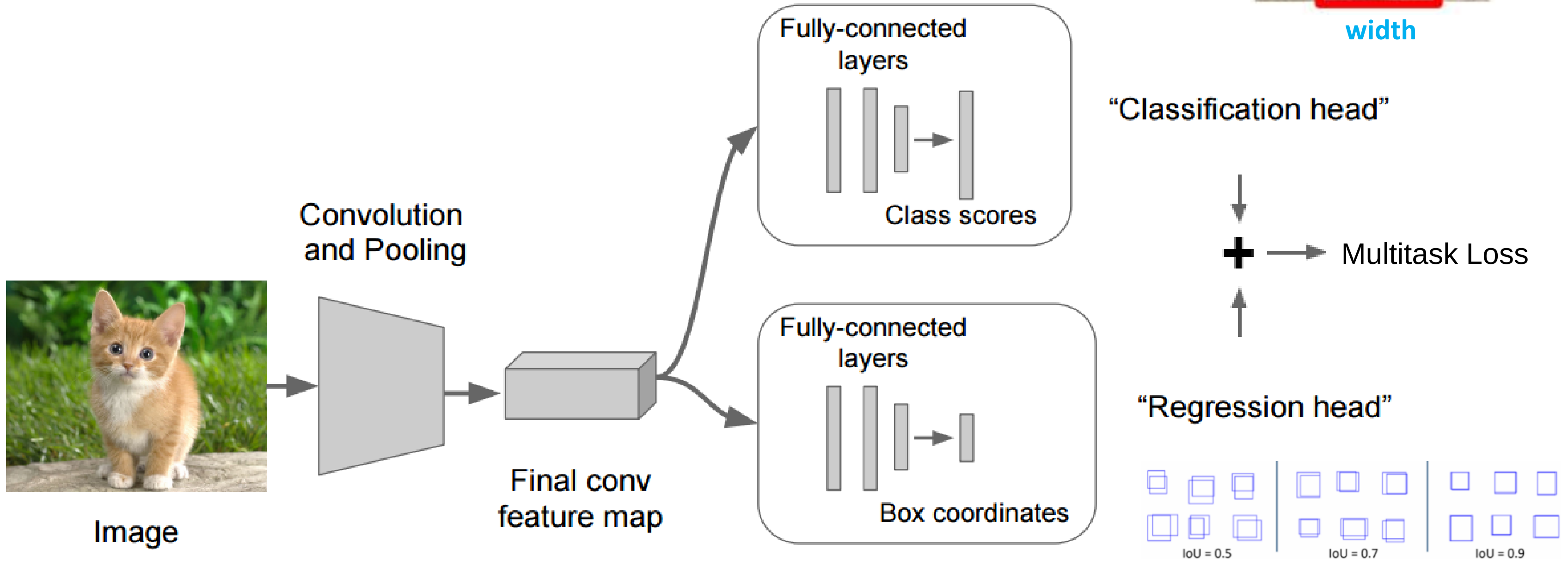
Relevant links:
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf
http://openaccess.thecvf.com/content_cvpr_2017/papers/Huang_SpeedAccuracy_Trade-Offs_for_CVPR_2017_paper.pdf
MultiBox detection (http://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Erhan_Scalable_Object_Detection_2014_CVPR_paper.pdf)
Bayesian Optimization: (http://web.eecs.umich.edu/~honglak/cvpr15-cnn-detection.pdf)
Multi-region: (http://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Gidaris_Object_Detection_via_ICCV_2015_paper.pdf)
RCNN Minus R (http://www.robots.ox.ac.uk/~vedaldi/assets/pubs/lenc15rcnn.pdf)
Image Windows (http://calvin.inf.ed.ac.uk/wp-content/uploads/Publications/alexe12pami.pdf)
Semantic Seg (http://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Long_Fully_Convolutional_Networks_2015_CVPR_paper.pdf)
Unconstrained Video (http://calvin.inf.ed.ac.uk/wp-content/uploads/Publications/papazoglouICCV2013-camera-ready.pdf)
Shape Guided (https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/Borenstein06.pdf)
Object Regions (http://crcv.ucf.edu/papers/cvpr2013/VideoObjectSegmentation.pdf)
Shape Sharing (http://www.cs.utexas.edu/~grauman/papers/shape-sharing-ECCV2012.pdf)

Classification + Localization

Object Classification and ***Localization*** (with Regression).

Train the Neural Net with an image and a ground truth bounding box with 4 numbers (x0, y0, width, height), and use L2 (Euclidean) distance to calculate the loss between the predicted bounding box and the ground truth. For this purpose, just attach another fully connected layer on the last convolution layer…

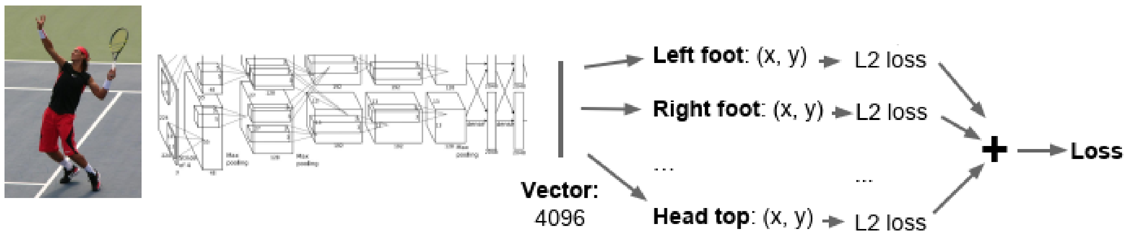


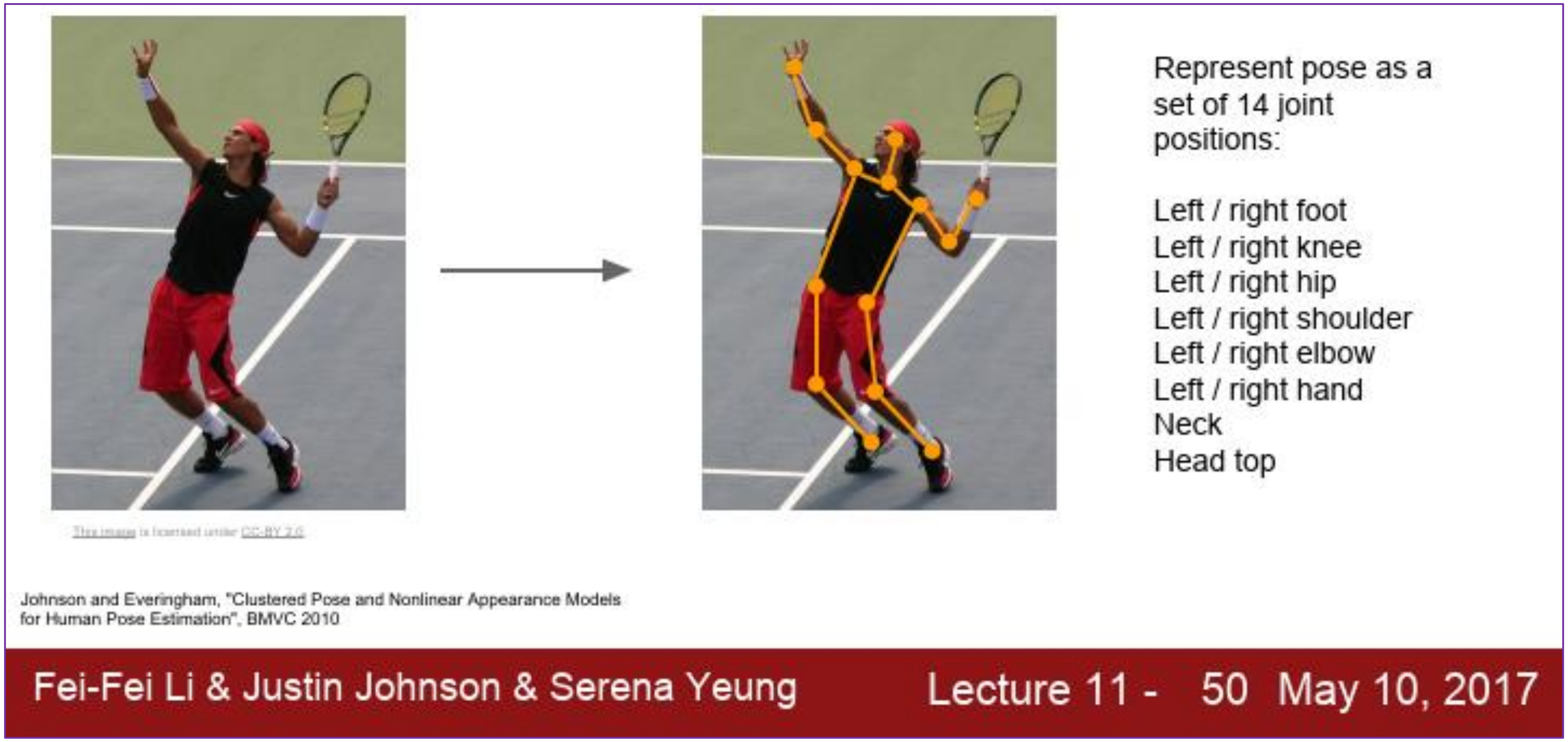


Supports only one object…

Localization

Human Pose Estimation


Vector:
4096
Left foot: (x, y) → L2 loss
Right foot: (x, y) → L2 loss
...
Head top: (x, y) → L2 loss
+ → Loss



Represent pose as a set of 14 joint positions:

Left / right foot
Left / right knee
Left / right hip
Left / right shoulder
Left / right elbow
Left / right hand
Neck
Head top

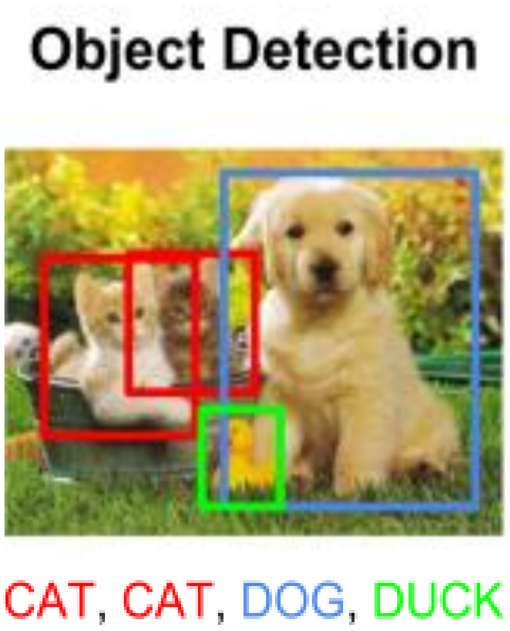
This image is licensed under CC-BY 2.0

Johnson and Everingham, "Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation", BMVC 2010

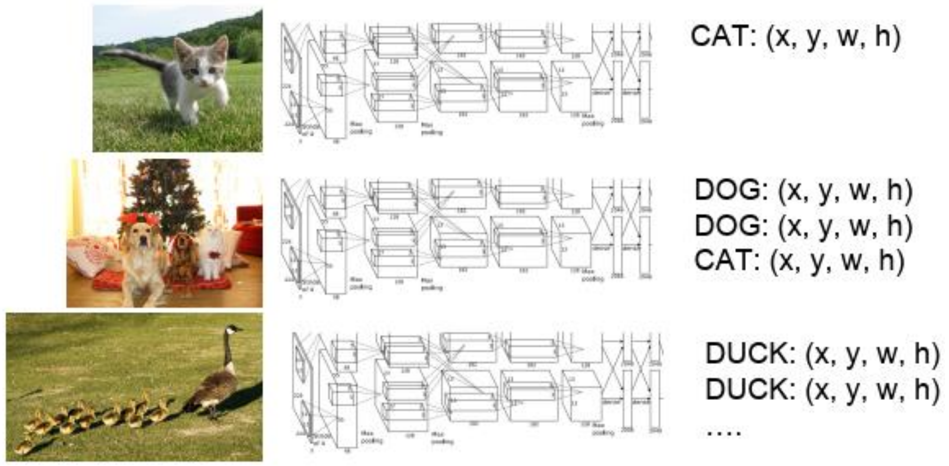
Fei-Fei Li & Justin Johnson & Serena Yeung   Lecture 11 - 50   May 10, 2017

Relevant links:
https://viso.ai/deep-learning/pose-estimation-ultimate-overview/

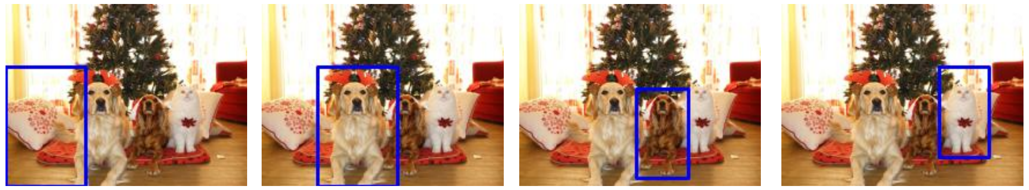
Object Detection

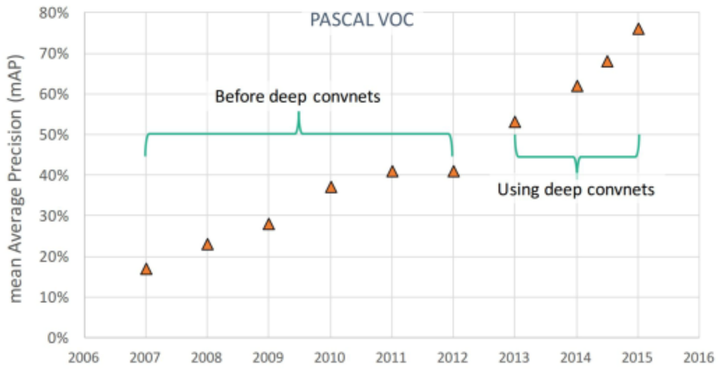


Can we approach Object Detection with Regression?...

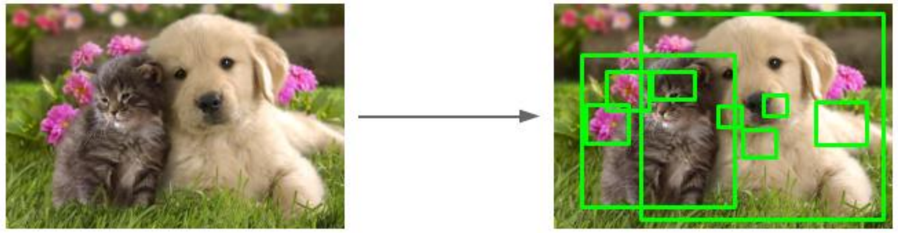


Can we approach Object Detection with Sliding Window?...

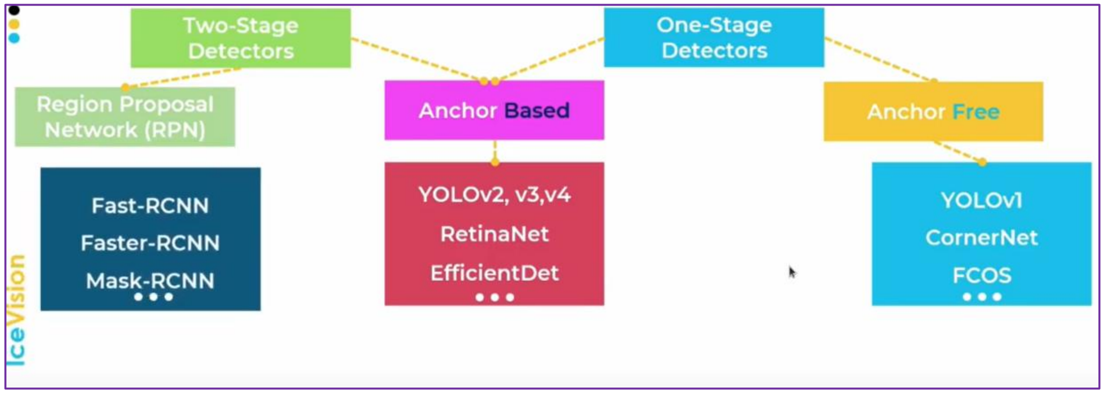


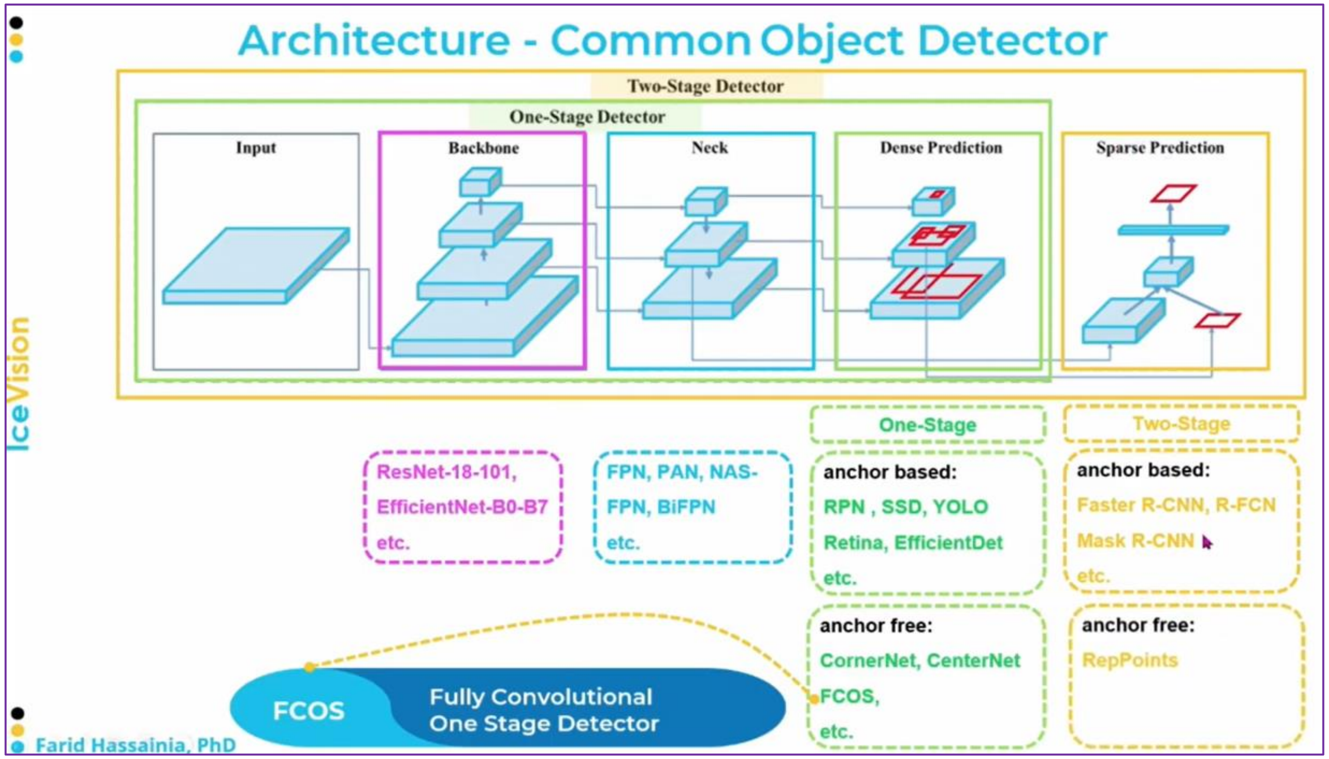


Let's use Region Proposals!



Object Detector



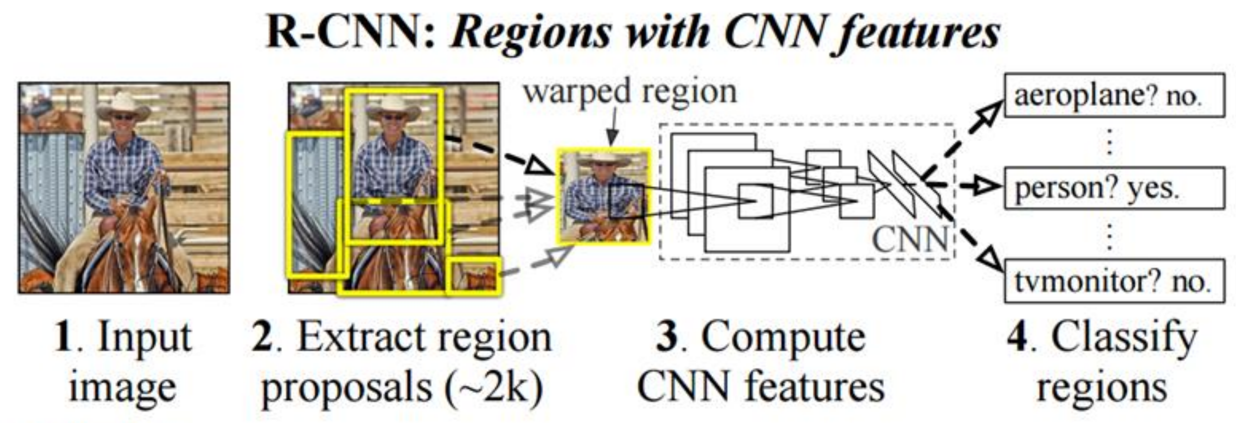

Architecture - Common Object Detector
Two-Stage Detector
One-Stage Detector
Input
Backbone
Neck
Dense Prediction
Sparse Prediction
ResNet-18-101, EfficientNet-B0-B7 etc.
FPN, PAN, NAS-FPN, BiFPN etc.
One-Stage
anchor based: RPN , SSD, YOLO Retina, EfficientDet etc.
anchor free: CornerNet, CenterNet FCOS, etc.
Two-Stage
anchor based: Faster R-CNN, R-FCN Mask R-CNN etc.
anchor free: RepPoints
FCOS
Fully Convolutional One Stage Detector
IceVision
Farid Hassainia, PhD

Relevant links:
https://www.youtube.com/watch?v=_ADYE6QaAAY
https://thesai.org/Downloads/Volume12No11/Paper_19-Deep_Learning_based_Neck_Models_for_Object_Detection.pdf

R-CNN architecture

R-CNN (Region proposals + CNN) is a method that relies on an external region proposal system.

- The ***Selective Search*** performs the function of generating 2000 different region proposals for bounding boxes that have the highest probability of containing an object.
- The proposals are "warped/deformed" into an image size that can be fed into a trained CNN (e.g. AlexNet) that extracts a feature vector for each region.
- The feature vector is used as the input to a set of linear ***Support Vector Machines (SVMs)*** that are trained for each class and output a classification. Alternatively, ***ANN*** could be used for multi-class classification purpose as well.
- The vector also gets fed into a bounding box "regressor" (a linear regression model) to obtain the tighter (most accurate) coordinates for the box once the object has been classified.



R-CNN workflow

Selective Search looks through windows of multiple scales and looks for adjacent ***pixels that share textures***, ***colors***, or ***intensities***.

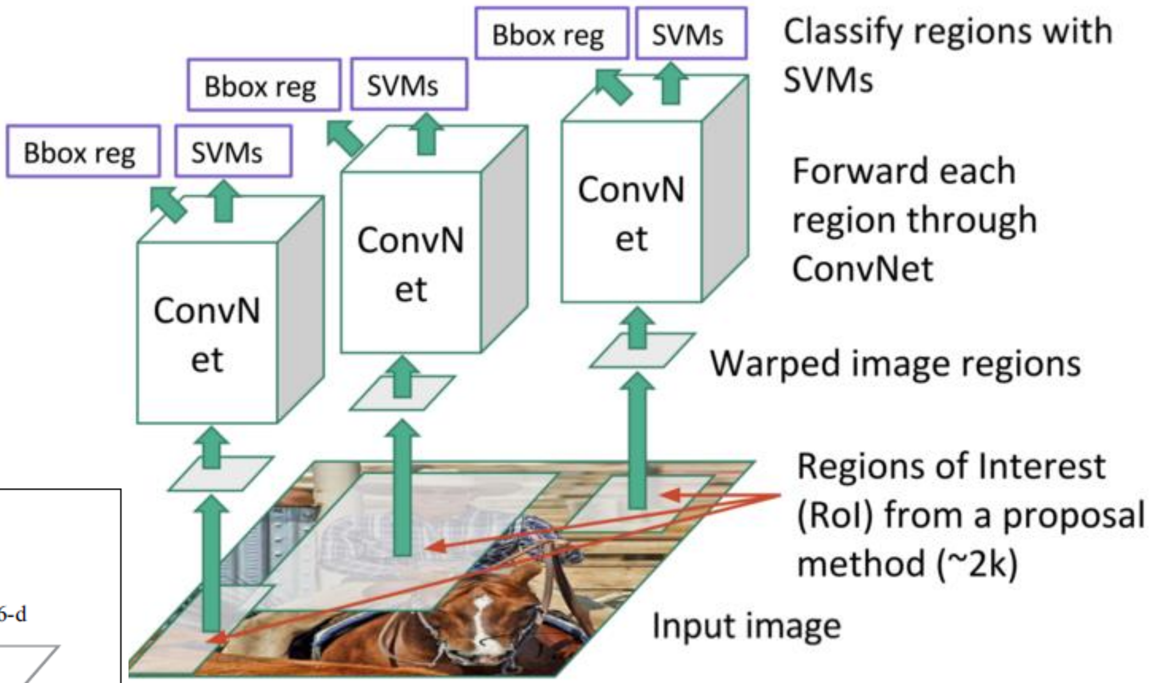
Relevant links:
https://arxiv.org/abs/1311.2524
https://www.quora.com/Where-can-I-find-a-nice-tutorial-for-Regional-based-Convolution-neural-Network
Selective search: https://ivi.fnwi.uva.nl/isis/publications/2013/UijlingsIJCV2013/UijlingsIJCV2013.pdf
SVM: https://en.wikipedia.org/wiki/Support_vector_machine
https://www.sciencedirect.com/science/article/pii/S1110866512000345

R-CNN architecture

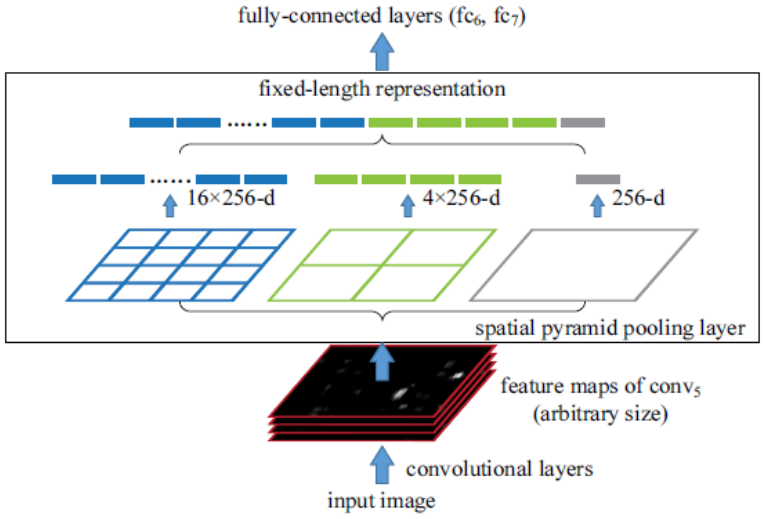
R-CNN Linear Regression for bounding box offsets…

- Cropping may loss some information about the object;
- Warping may change the object's appearance

check ***SPP-Net*** with ***Spatial Pyramid Pooling Layer***



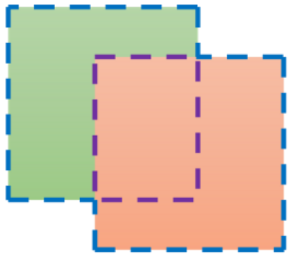
Bbox reg
SVMs
Classify regions with SVMs
ConvNet
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image



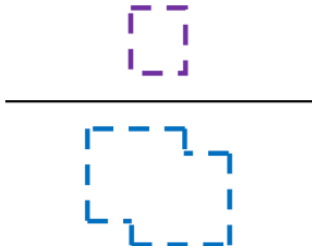
fully-connected layers (fc_6, fc_7)
fixed-length representation
16×256-d
4×256-d
256-d
spatial pyramid pooling layer
feature maps of $conv_5$ (arbitrary size)
convolutional layers
input image

B-box selection

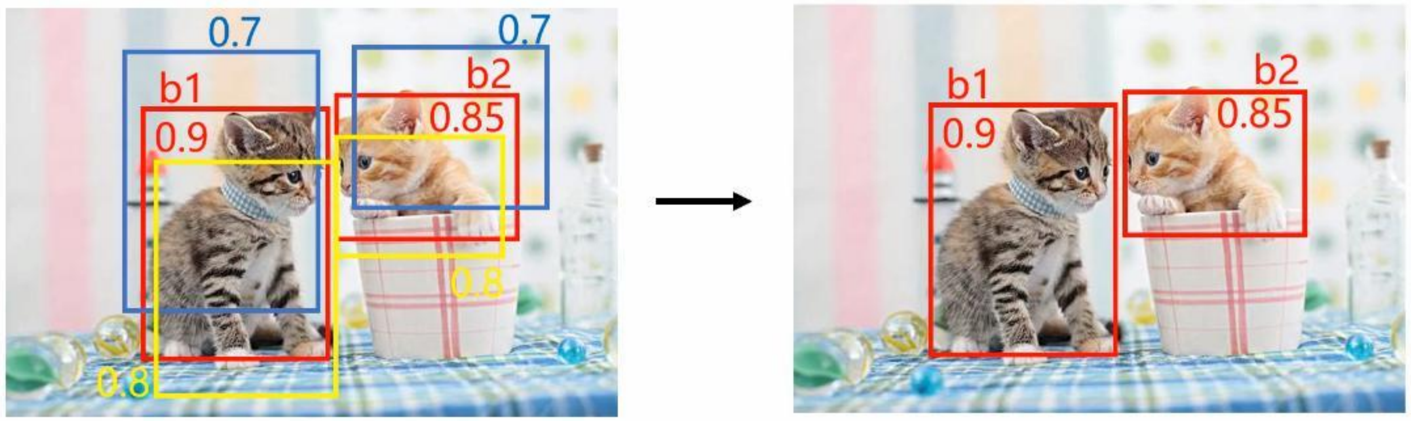
IoU – Intersection over Union



$$IOU = \frac{surface\ of\ intersection}{surface\ of\ union} =$$

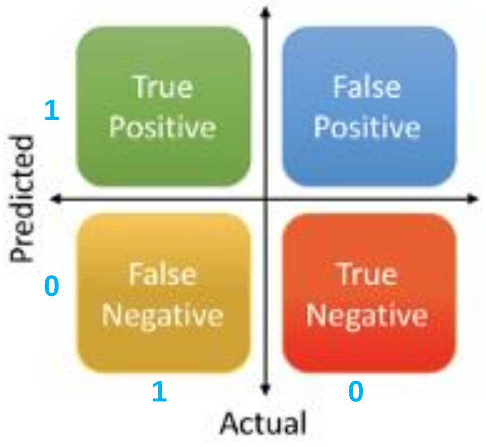


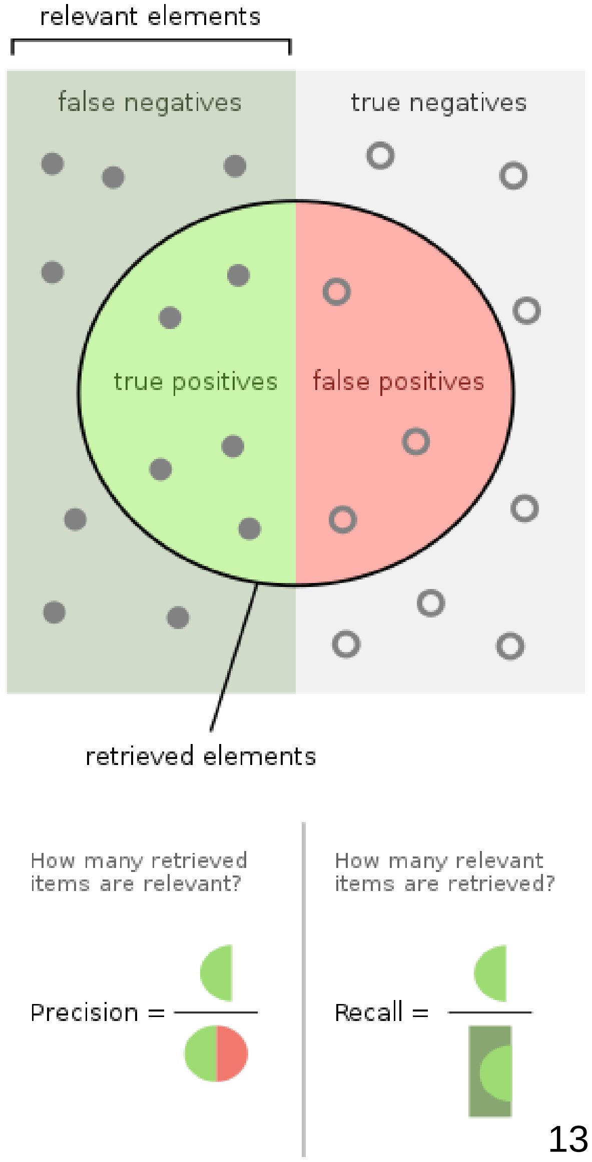
NMS – Non-Max Suppression



Open challenge is to handle very "crowdy" images…

Evaluation of Object Detection Model





percentage of the results which are relevant…

$$\text{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \text{ or } \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

percentage of total relevant results correctly classified by the algorithm…

$$\text{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \text{ or } \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

percentage of correct predictions out of total cases…

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{Total}}$$

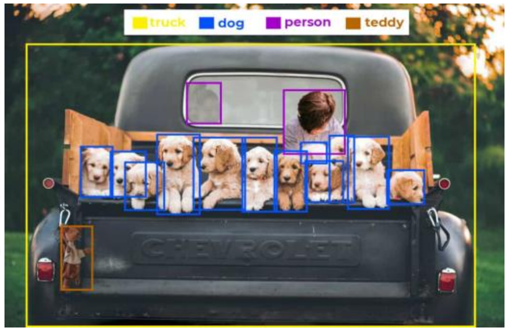
The F1 score is simply the harmonic mean of Precision and Recall…

$$F_1 = 2 * \frac{precision * recall}{precision + recall} = \frac{tp}{tp + \frac{1}{2}fp + fn}$$

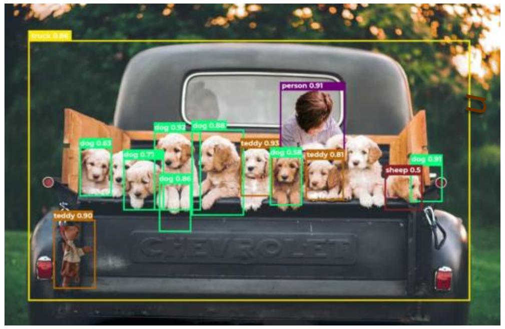
Relevant links:
https://en.wikipedia.org/wiki/Precision_and_recall
https://medium.com/@shrutisaxena0617/precision-vs-recall-386cf9f89488

Evaluation of Object Detection Model

Ground Truth



12 dogs, etc.



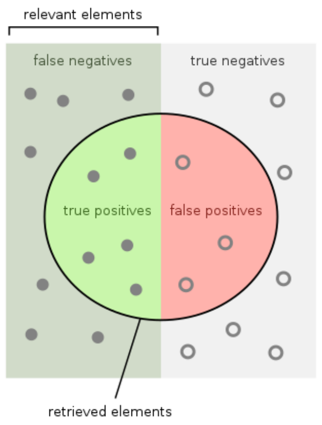
Predictio n

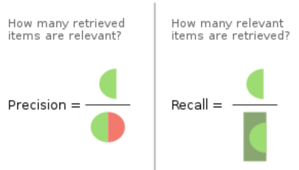
7 dogs, etc.

Detections							
Conf.	0.63	0.77	0.92	0.86	0.88	0.58	0.91
Matches GT by IoU?	TP	TP	TP	FP	TP	TP	FP

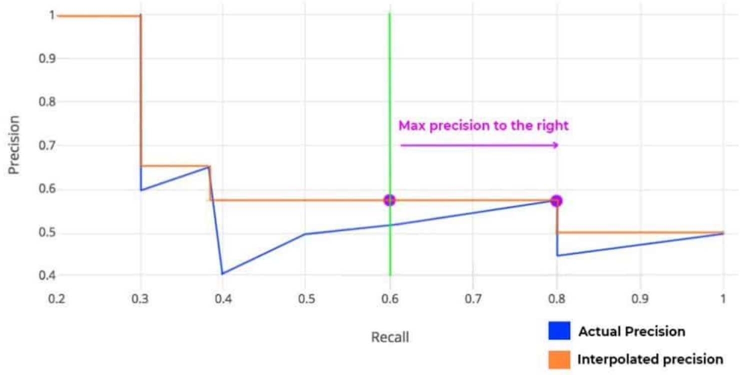
Precision = Cumulative TP / (Cumulative TP + Cumulative FP)
Recall = Cumulative TP / Total Ground Truths

Preds.	Conf.	Matches	Cumulative TP	Cumulative FP	Precision	Recall
	0.92	TP	1	0	**1/(1+0) = 1**	**1/16 = 0.08**
	0.91	FP	1	1	**1/(1+1) = 0.5**	**1/16 = 0.08**
	0.88	TP	2	1	**2/(2+3) = 0.66**	**2/16 = 0.16**
	0.86	FP	2	2	**0.5**	**0.16**
	0.77	TP	3	2	**0.6**	**0.25**
	0.63	TP	4	2	**0.66**	**0.33**
	0.58	TP	5	2	**0.71**	**0.41**



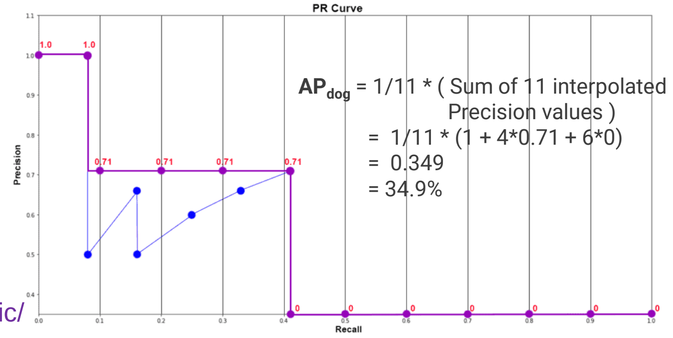

How many retrieved items are relevant?
How many relevant items are retrieved?
Precision =
Recall =

Average Precision (AP) – area under ***Precision vs Recall Curve***, which is calculated class-wise.



Mean Average Precision (mAP)

mAP = 1/n * sum(AP), where ***n*** is the number of predicted classes.



$\textbf{AP}_{\textbf{dog}}$ = 1/11 * (Sum of 11 interpolated Precision values)
= 1/11 * (1 + 4*0.71 + 6*0)
= 0.349
= 34.9%

Relevant links:
https://learnopencv.com/mean-average-precision-map-object-detection-model-evaluation-metric/

Evaluation of Object Detection Model

- 11 Point Interpolation AP was introduced in 2007.
- All Point AP was adopted later in 2010.
- PASCAL VOC and ImageNET challenge used AP and mAP calculated at 0.5 IoU.
- At present, MS COCO 101 point Average Precision (AP) is accepted as the standard metric.
- COCO mAP is an averaged mAP@[0.5:.05:.95] *(a set of 10 different IoU thresholds with the step 0.05)*.

Relevant links:
https://learnopencv.com/mean-average-precision-map-object-detection-model-evaluation-metric/

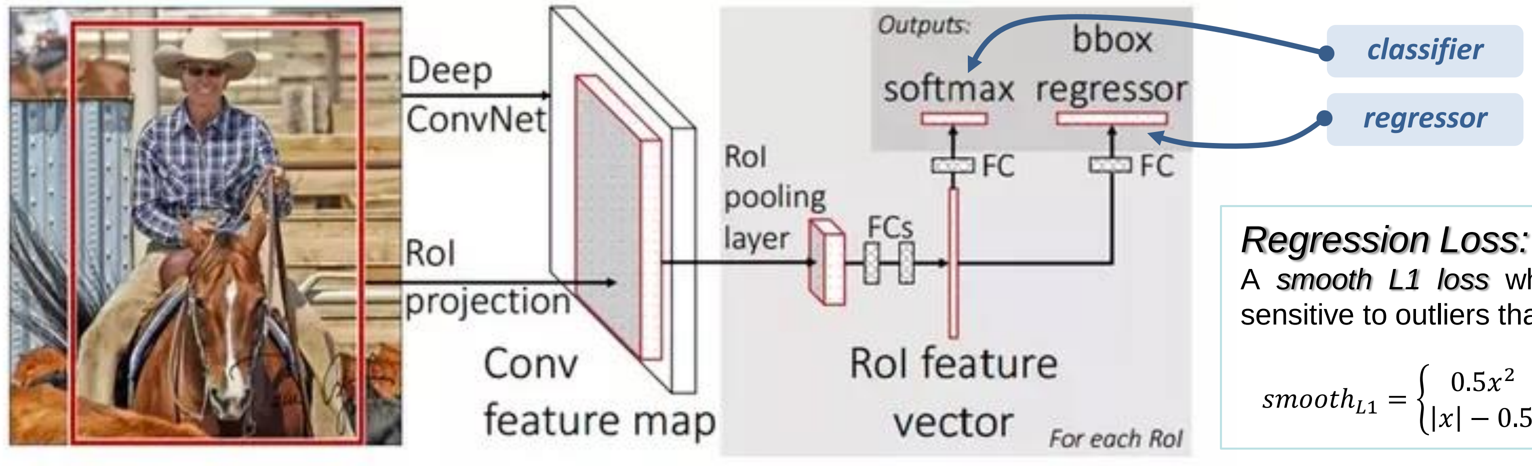
Fast R-CNN architecture

Fast R-CNN

R-CNN works really well, but is computationally expensive and extremely slow due to a few reasons:

- It requires a forward pass of the CNN for every single region proposal for every single image (*that's around 2000 forward passes per image!*).
- It has to train three different models separately - the CNN to generate image features, the classifier that predicts the class, and the regression model to tighten the bounding boxes that makes the pipeline extremely hard to train.

A big drawback of SPP net - it is not trivial to perform back-propagation through spatial pooling layer (net is trained only partially).



Fast R-CNN workflow

Regression Loss:
A ***smooth L1 loss*** which is less sensitive to outliers than ***L2 loss***.

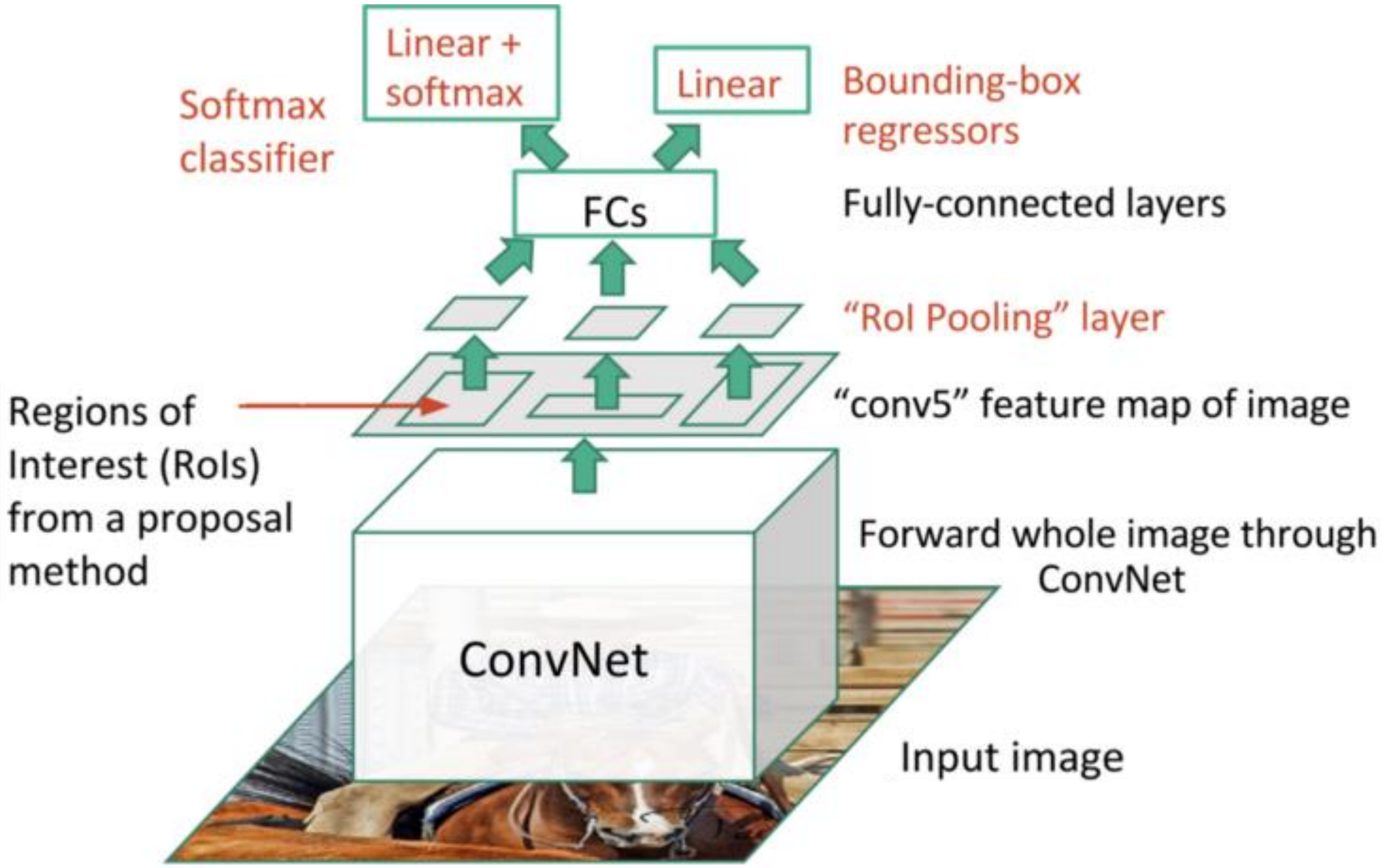
$$smooth_{L1} = \begin{cases} 0.5x^2 & if|x| < 1 \\ |x| - 0.5 & otherwise, \end{cases}$$

In 2015, Ross Girshick (the first author of R-CNN) solved both these problems, leading to the second algorithm ***of joint training framework Fast R-CNN*** that jointly trains the CNN, classifier, and bounding box regressor in a single model. It simplifies and improves the speed by sharing computation of the conv layers between different proposals (swapping the steps of region proposals generation and running the CNN). This technique known as Region of Interest Pooling (RoIPool)...

Relevant links:
https://arxiv.org/abs/1504.08083
https://www.quora.com/Where-can-I-find-a-nice-tutorial-for-Regional-based-Convolution-neural-Network

Fast R-CNN architecture

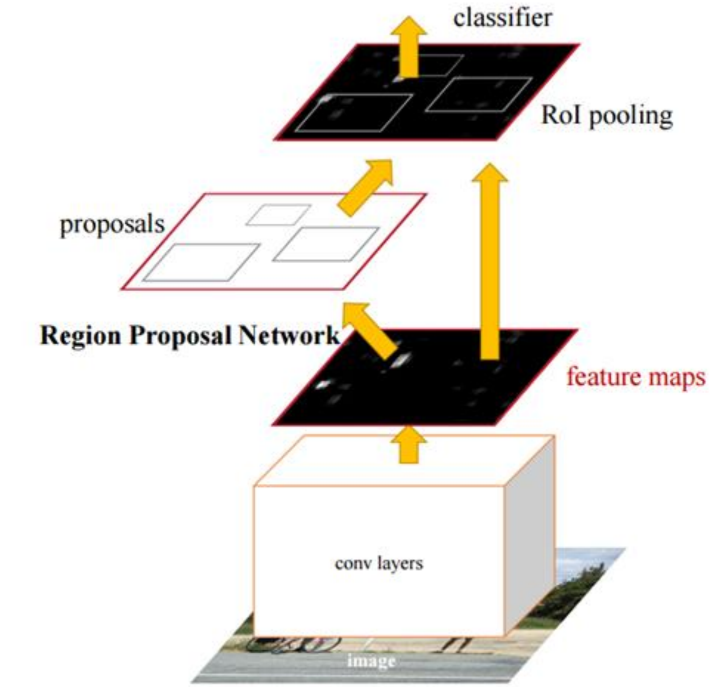
Fast R-CNN



Faster R-CNN architecture

Faster R-CNN (team led by Jian Sun, a principal researcher at Microsoft Research)

*Fast R-CNN process has still one remaining bottleneck—the **region proposer**. In 2015, a team at Microsoft Research found a way to make the region proposal step almost cost free…*

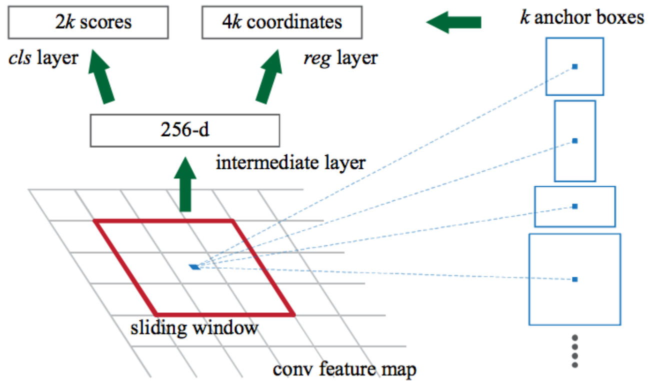


Faster R-CNN workflow

The insight of Faster R-CNN is that region proposals depended on features of the image calculated with CNN. Faster R-CNN reuses CNN results for region proposals instead of running a separate selective search algorithm. "Our observation is that the convolutional feature maps used by region-based detectors, like Fast R- CNN, can also be used for generating region proposals…"

Faster R-CNN adds a Fully Convolutional Network on top of the features of the CNN creating what's known as the ***Region Proposal Network***.

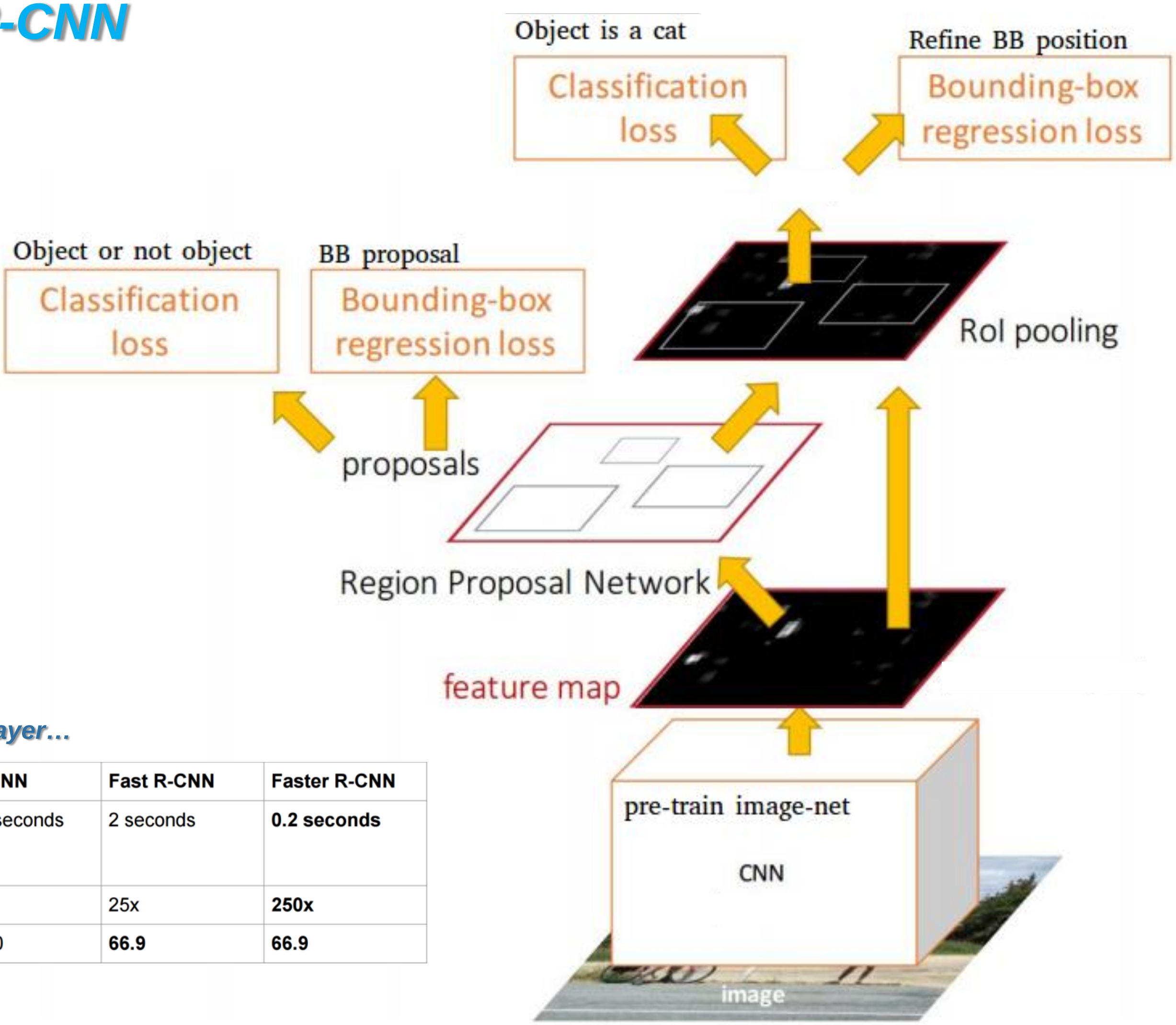
The Region Proposal Network works by passing a sliding window over the CNN feature map and at each window, outputting *k* potential bounding boxes and scores for how good each of those boxes is expected to be. (e.g. We know that the bounding boxes for people tend to be rectangular and vertical).



Relevant links:
https://arxiv.org/abs/1506.01497
https://www.quora.com/Where-can-I-find-a-nice-tutorial-for-Regional-based-Convolution-neural-Network
https://blogs.microsoft.com/ai/microsoft-researchers-win-imagenet-computer-vision-challenge/#sm.00017fqnl1bz6fqf11amuo0d9ttdp

Faster R-CNN architecture

Faster R-CNN

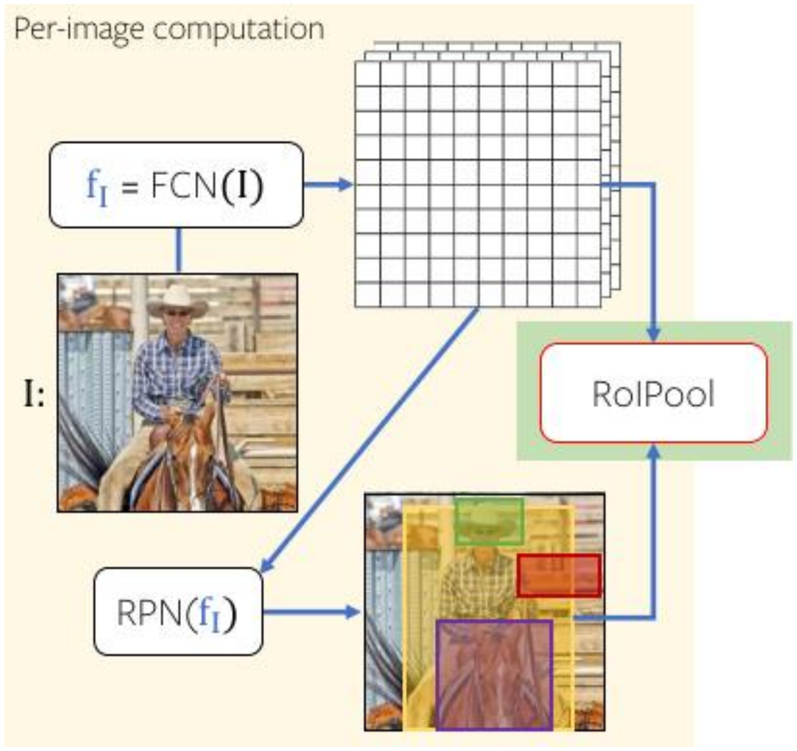


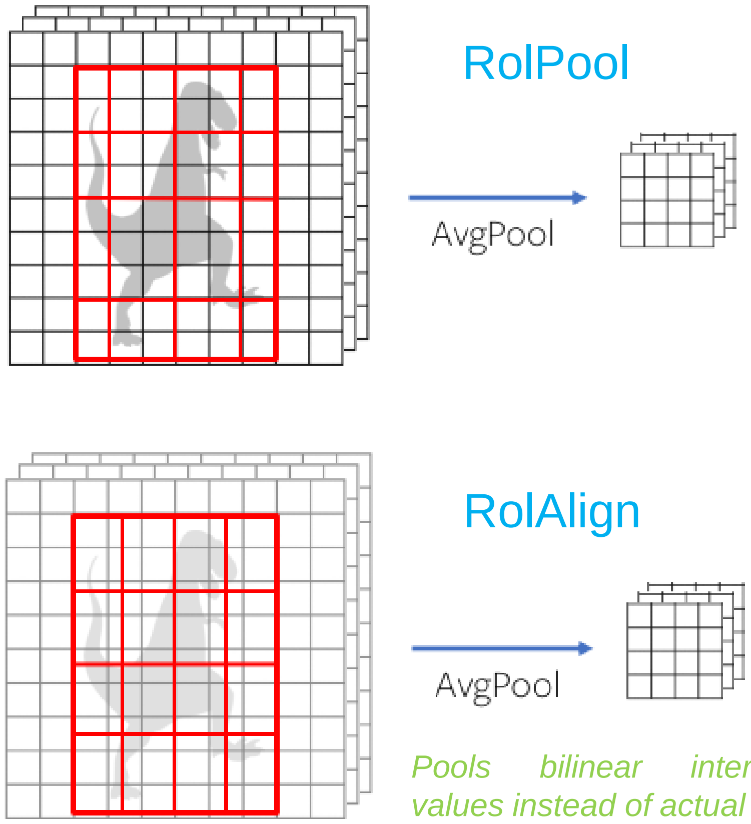
With ResNet 101 layer…

	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image (with proposals)	50 seconds	2 seconds	**0.2 seconds**
(Speedup)	1x	25x	**250x**
mAP (VOC 2007)	66.0	**66.9**	**66.9**

Faster R-CNN architecture

RoIPool extracts same size feature map for each proposal

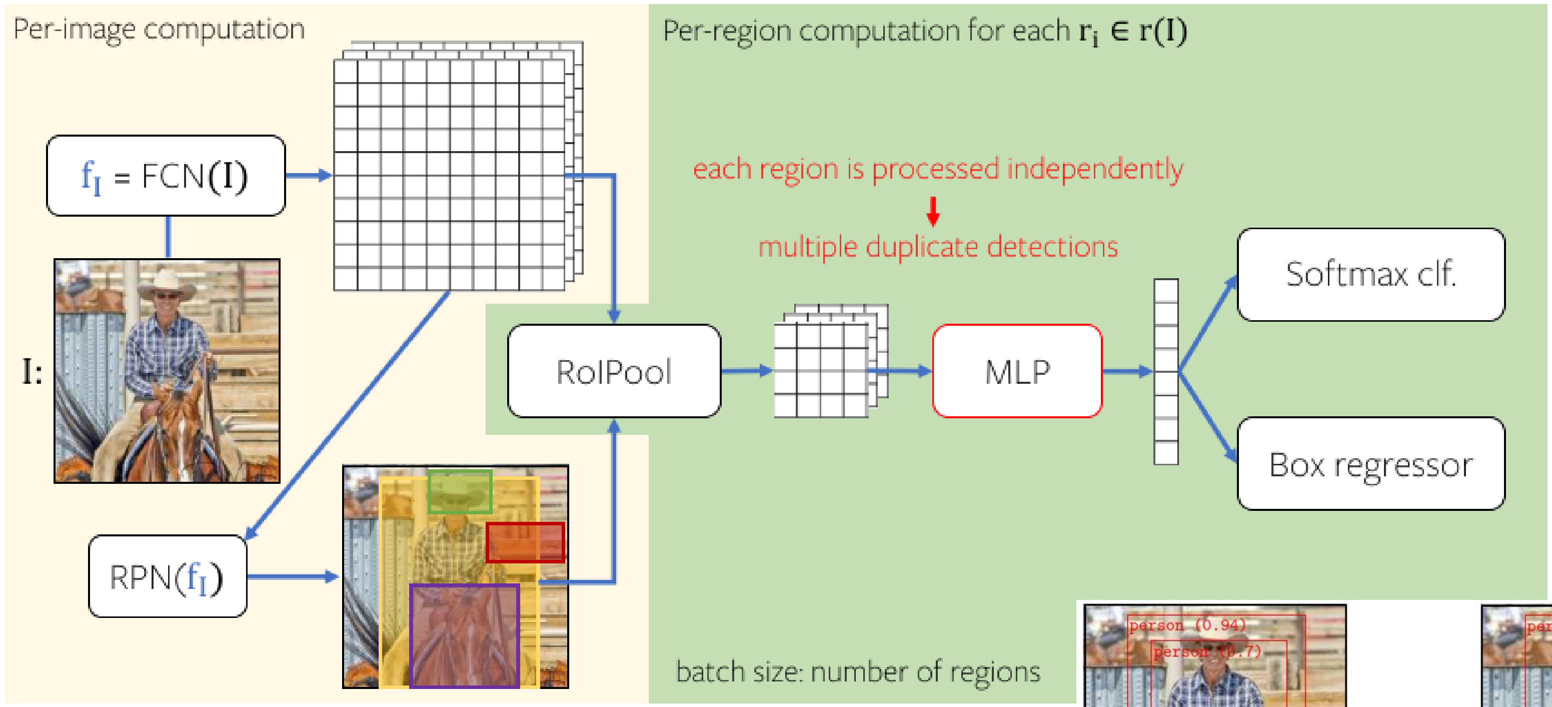


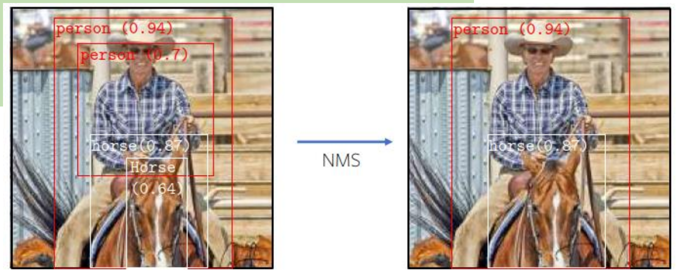


Pools bilinear interpolated values instead of actual pixels

Relevant links:
https://www.youtube.com/watch?v=5bWG3ySf4fI
https://arxiv.org/pdf/1506.01497.pdf
https://arxiv.org/abs/1703.06870

Faster R-CNN architecture


Per-image computation
f_I = FCN(I)
I:
RPN(f_I)
Per-region computation for each $r_i \in r(I)$
each region is processed independently
multiple duplicate detections
RoIPool
MLP
Softmax clf.
Box regressor
batch size: number of regions

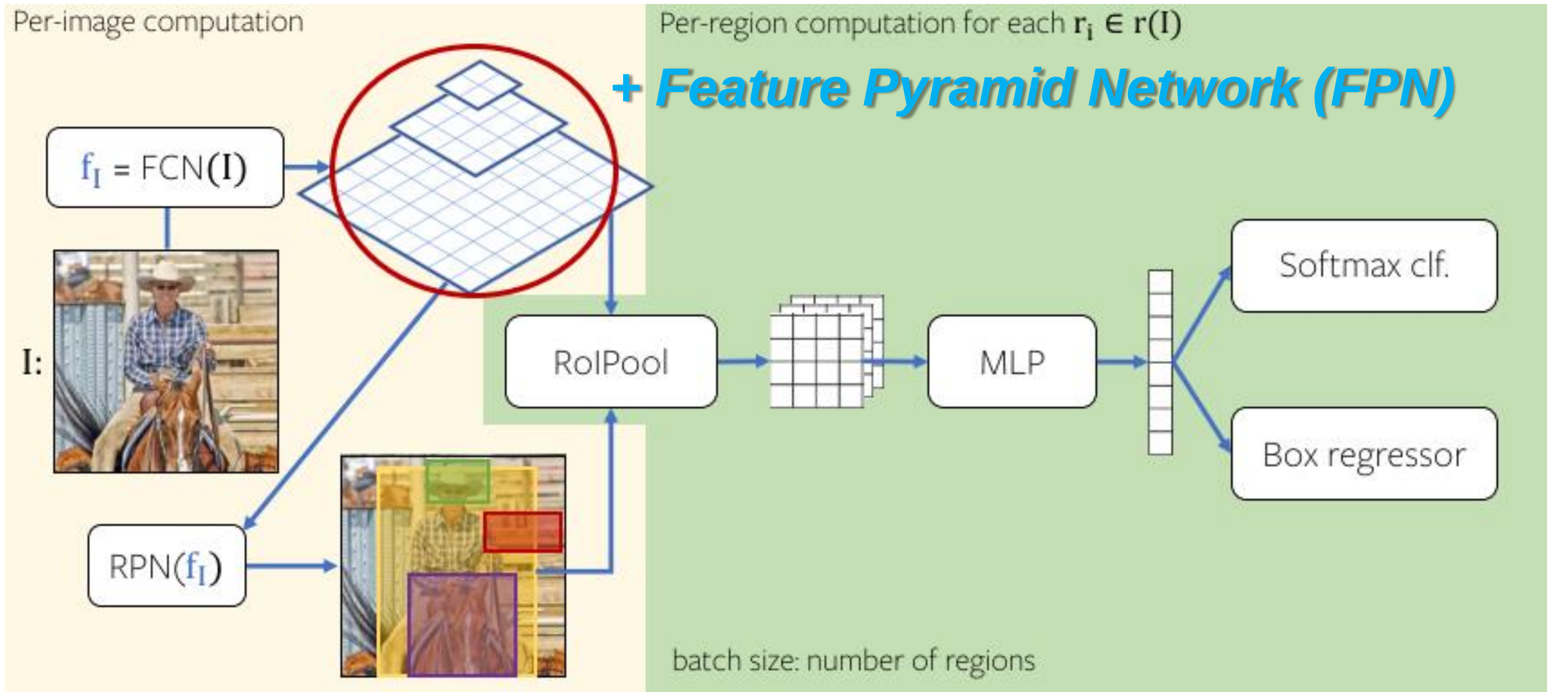


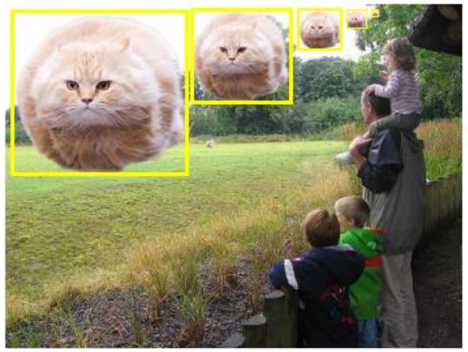
Non-maximum suppression (NMS)

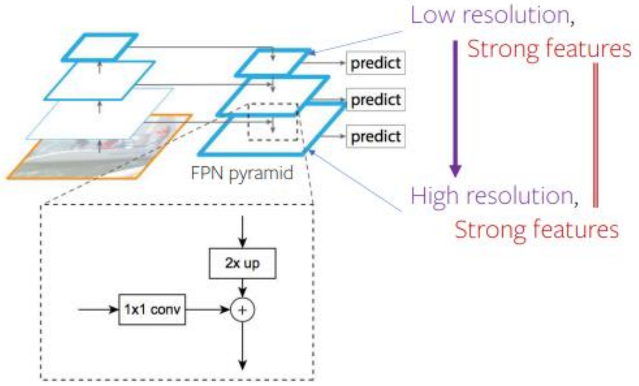
post-processing heuristic to remove duplicate detections

Relevant links:
https://www.youtube.com/watch?v=5bWG3ySf4fI
https://arxiv.org/pdf/1506.01497.pdf

Faster R-CNN architecture

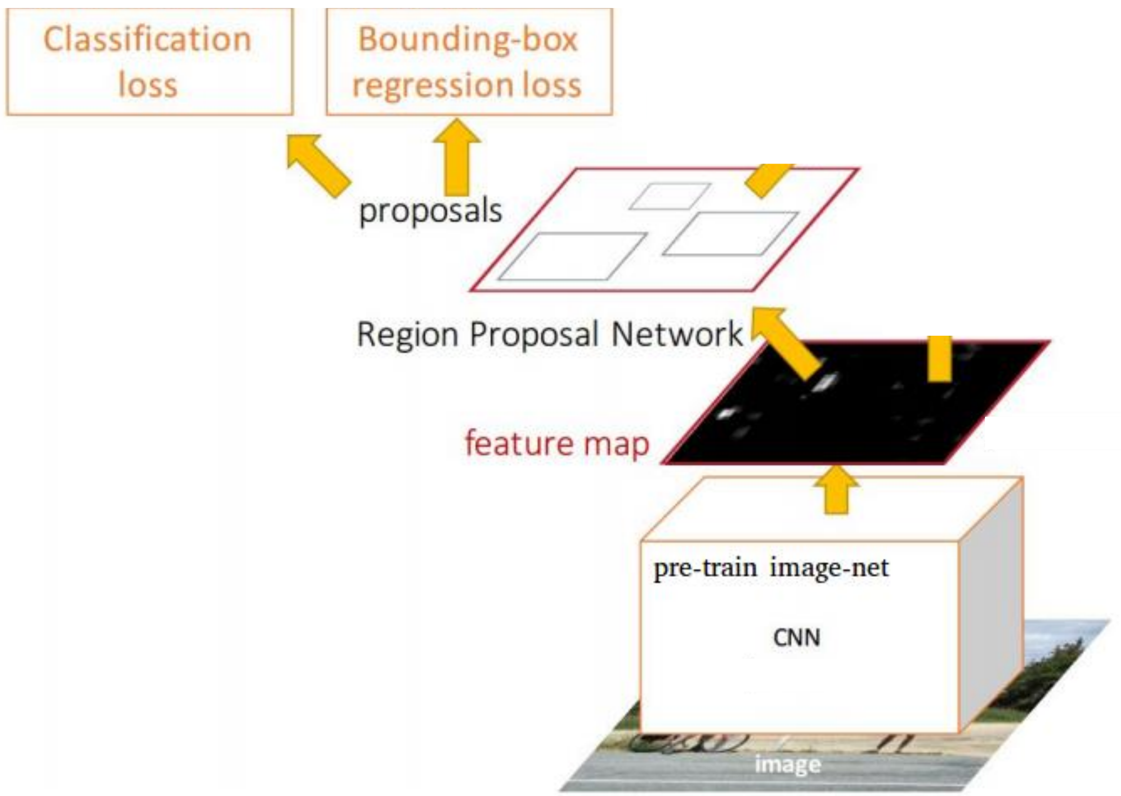

Per-image computation
Per-region computation for each $\mathbf{r_i} \in \mathbf{r(I)}$
+ Feature Pyramid Network (FPN)
$\mathrm{f_I}$ = FCN(I)
I:
RPN($\mathrm{f_I}$)
RoIPool
MLP
Softmax clf.
Box regressor
batch size: number of regions





Relevant links:
https://www.youtube.com/watch?v=5bWG3ySf4fI
https://arxiv.org/pdf/1506.01497.pdf
https://arxiv.org/abs/1612.03144

Single-Stage: Fully Convolutional Detector



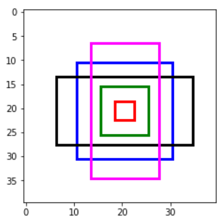
SSD

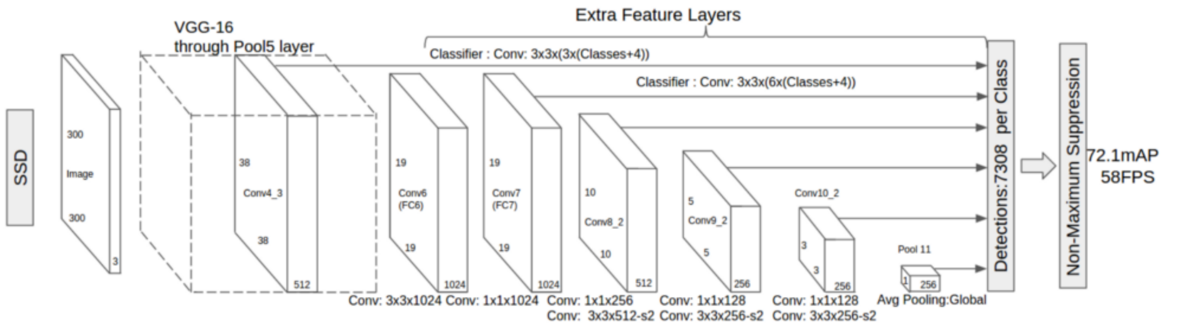
SSD

SSD (Single-Shot Detector) provides enormous speed gains over Faster R-CNN, but does so in a markedly different manner. In contrast to previous models that used a region proposal network to generate regions of interest and further either fully-connected layers or position-sensitive convolutional layers to classify those regions, SSD does the two in a “single shot”, simultaneously predicting the bounding box and the class as it processes the image.

Without region proposal SSD skips filtering step where we get object regions (with certain confidence). SSD classifies and draws bounding boxes from *every single position in the image*, using *multiple different shapes*, at *several different scales*. As a result, a much greater number of bounding boxes are generated, and nearly all of the them are negative examples. To fix this imbalance, SSD does:

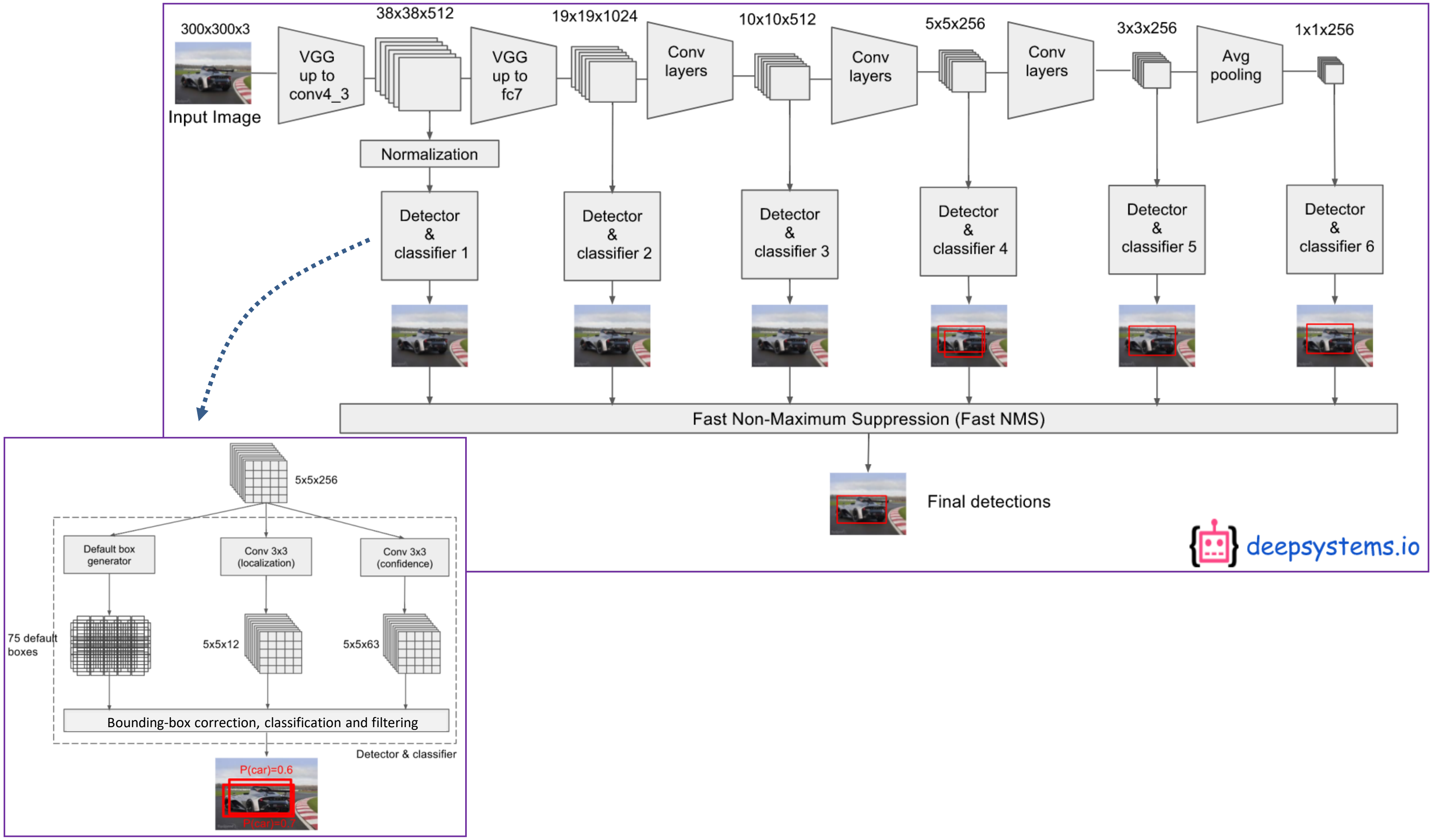
- ***non-maximum suppression*** to group together highly-overlapping boxes into a single box. In other words, if four boxes of similar shapes, sizes, etc. contain the same dog, NMS would keep the one with the highest confidence and discard the rest.
- ***hard negative mining*** to balance classes during training, only a subset of the negative examples with the highest training loss (i.e. false positives) are used at each iteration of training. SSD keeps a 3:1 ratio of negatives to positives.





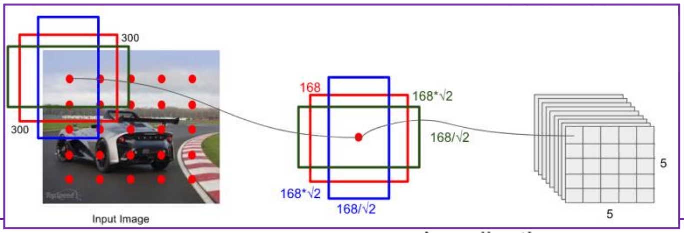
Relevant links:
https://arxiv.org/abs/1512.02325
https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab
https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11
https://towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9
https://lilianweng.github.io/lil-log/2018/12/27/object-detection-part-4.html
https://github.com/pierluigiferrari/ssd_keras
https://lambdalabs.com/blog/how-to-implement-ssd-object-detection-in-tensorflow/
https://github.com/balancap/SSD-Tensorflow
https://github.com/weiliu89/caffe/tree/ssd
http://silverpond.com.au/2017/02/17/how-we-built-and-trained-an-ssd-multibox-detector-in-tensorflow.html

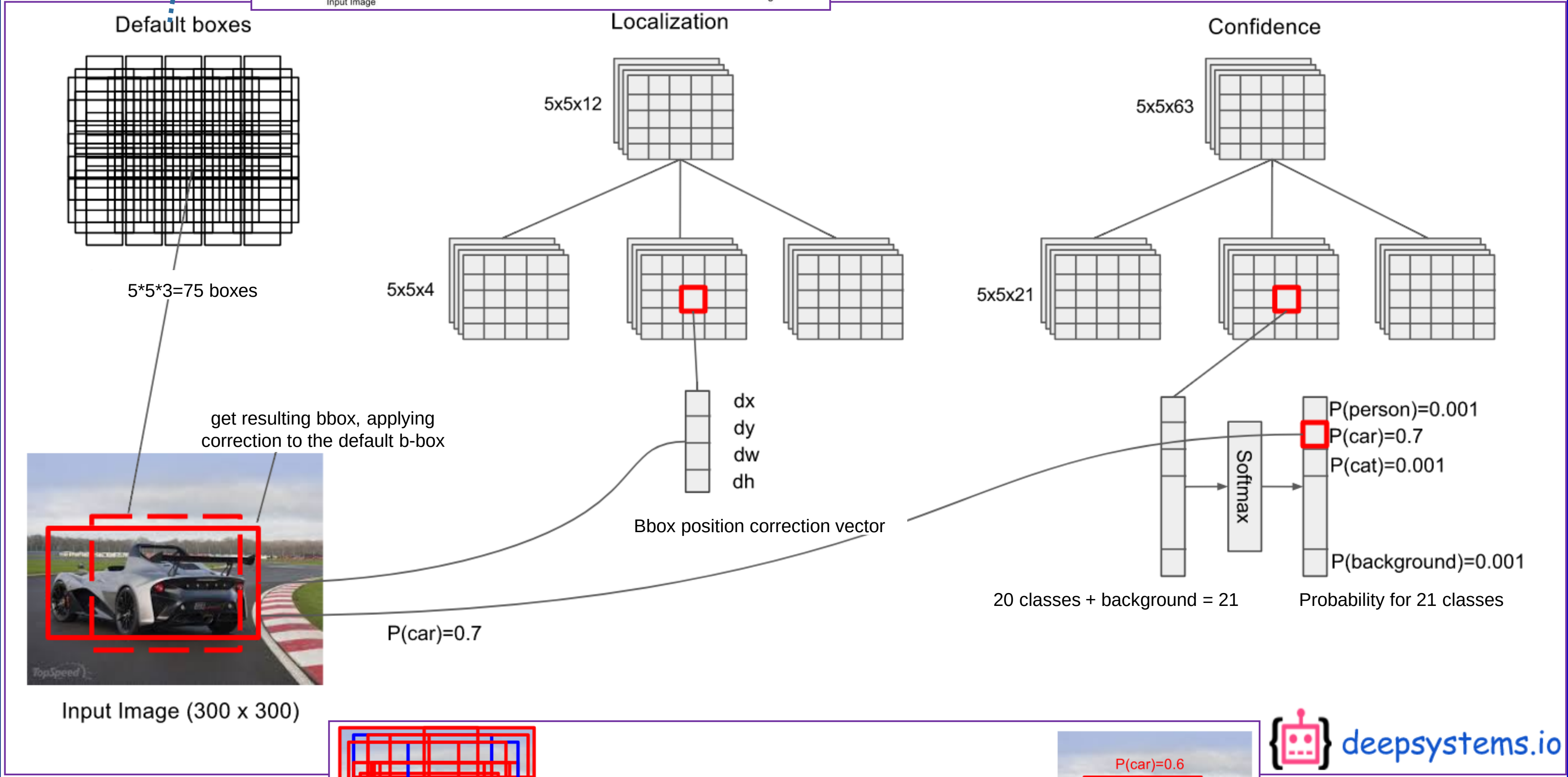
SSD

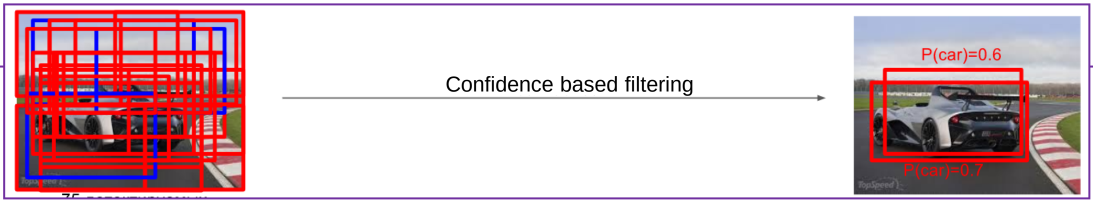


Relevant links:
https://deepsystems.ai

SSD



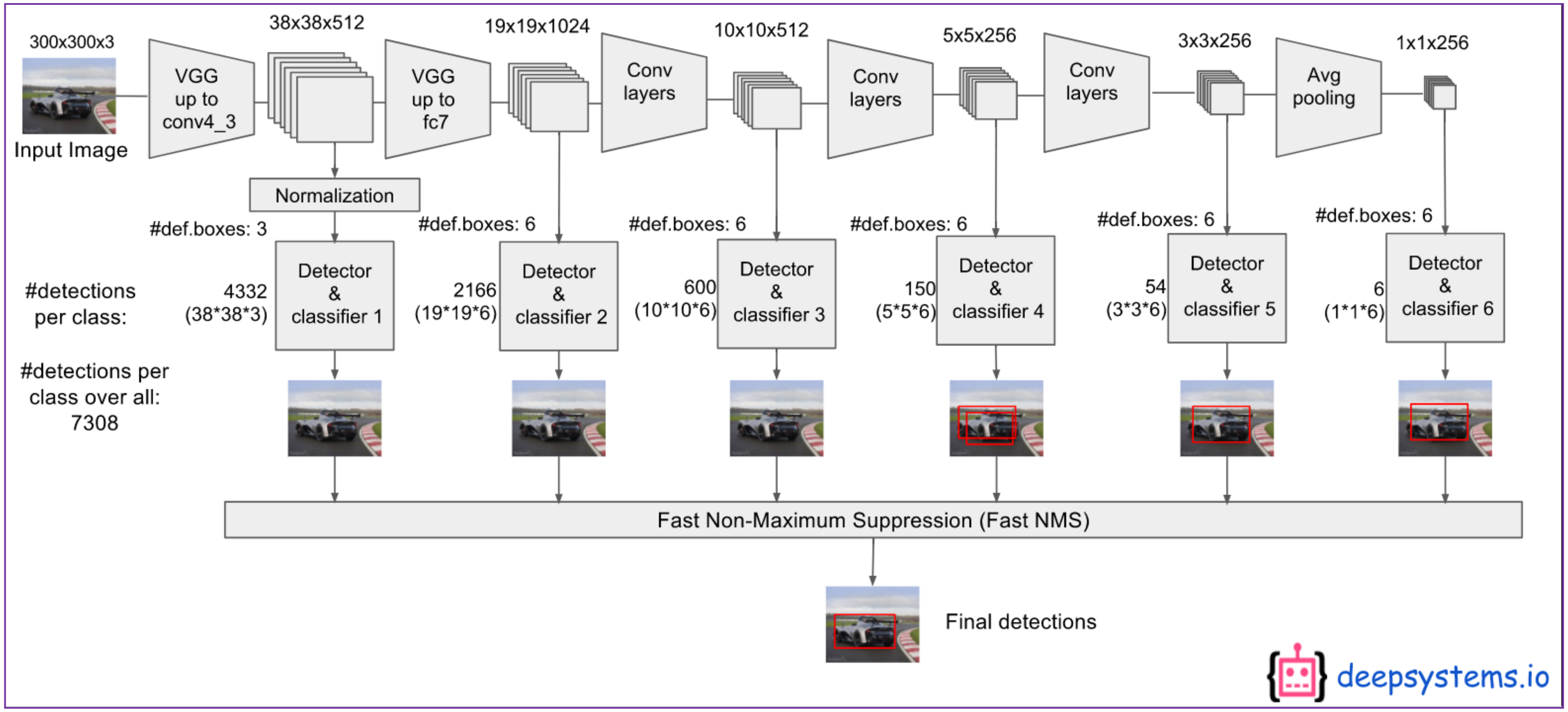




{:} deepsystems.io

Relevant links:
https://deepsystems.ai

SSD



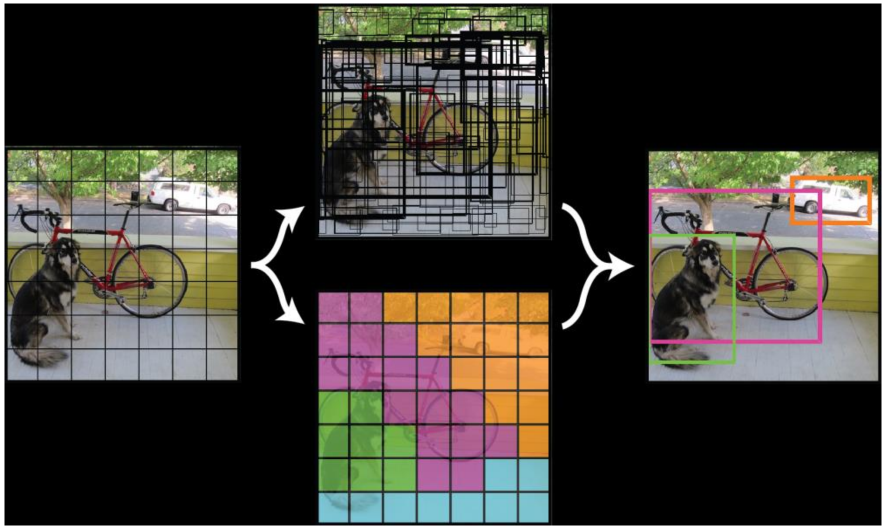
Relevant links:
https://deepsystems.ai

YOLO

YOLO ***(You Only Look Once)*** is a system for detecting objects on the *Pascal VOC 2012* dataset.
(http://host.robots.ox.ac.uk:8080/pascal/VOC/)

It can detect the 20 Pascal object classes:
- person
- bird, cat, cow, dog, horse, sheep
- aeroplane, bicycle, boat, bus, car, motorbike, train
- bottle, chair, dining table, potted plant, sofa, tv/monitor

Unlike SSD and YOLO, prior detection systems repurpose classifiers or localizers to perform detection and apply the model to an image at multiple locations and scales, and high scoring regions of the image are considered detections.

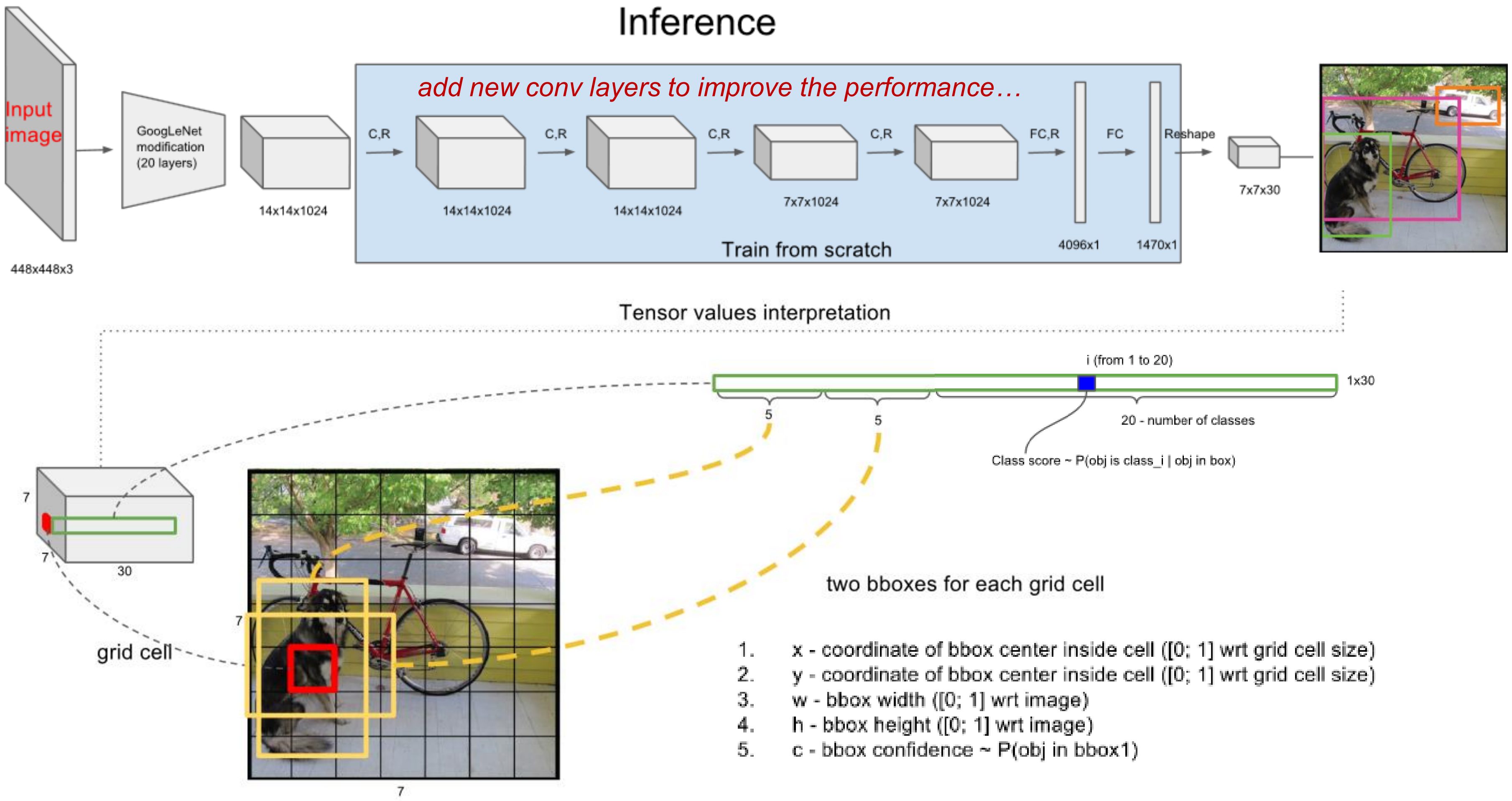


In turn, YOLO applies a single neural network to the full image, divides image into regions and predicts bounding boxes and its probabilities for each region (2 bboxes in YOLO v1, up to 5 in YOLO v2, etc.) as well as predict the class of the corresponding object. These bounding boxes are weighted by the predicted probabilities to only highlight the high scoring detections. Combined score of both (bounding box and object class) predictions finally is assessed.

Making predictions with a single network evaluation, YOLO becomes more than 1000x faster than R-CNN (which require thousands for a single image) and 100x faster than Fast R-CNN.

Relevant links:
http://arxiv.org/abs/1506.02640
https://pjreddie.com/darknet/yolo/
https://towardsdatascience.com/yolov1-you-only-look-once-object-detection-e1f3ffec8a89
https://www.youtube.com/watch?v=VOC3huqHrss
https://github.com/llSourcell/YOLO_Object_Detection
https://www.youtube.com/watch?v=4eIBisqx9_g

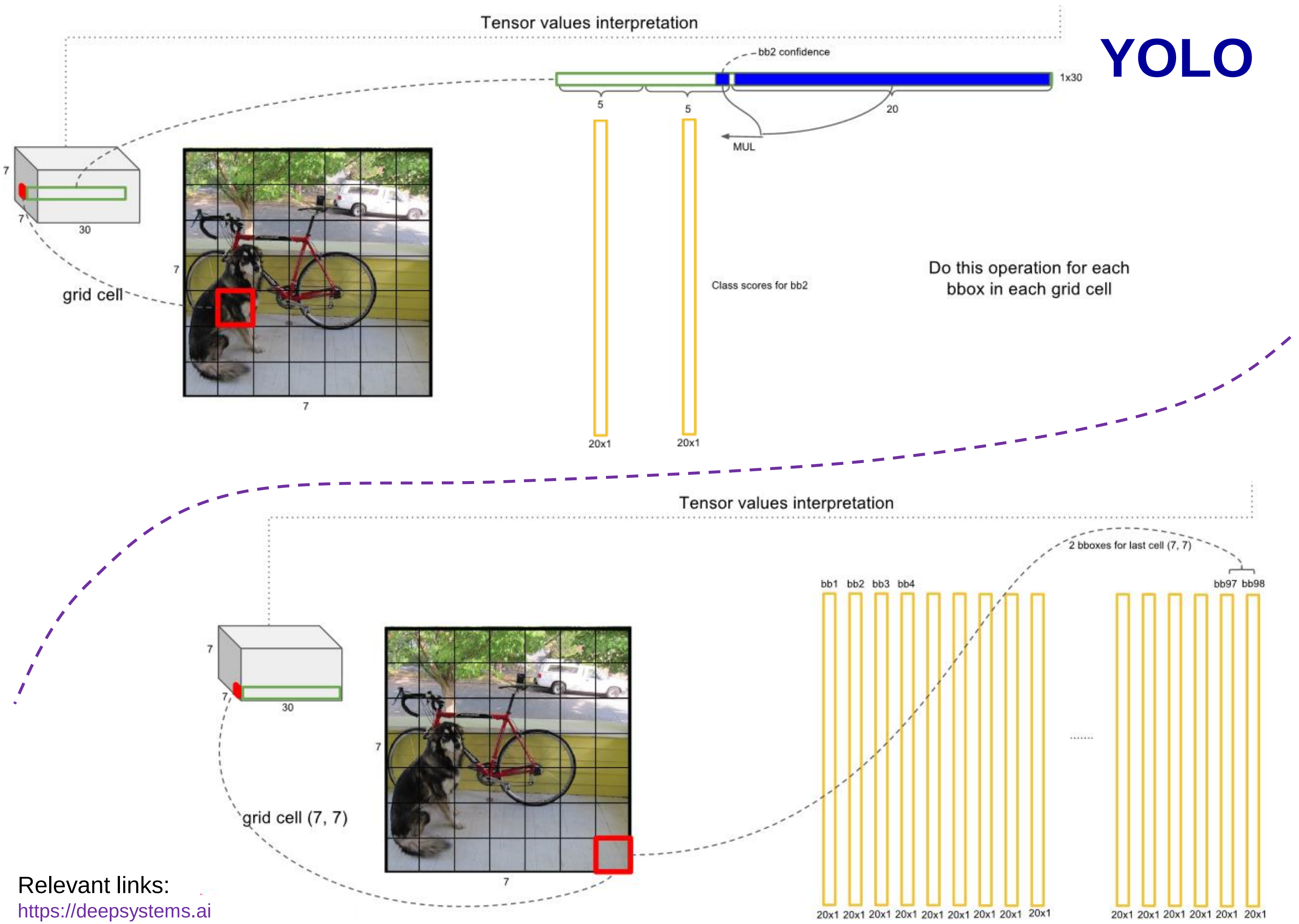
YOLO



1. x - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
2. y - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
3. w - bbox width ([0; 1] wrt image)
4. h - bbox height ([0; 1] wrt image)
5. c - bbox confidence ~ P(obj in bbox1)

Relevant links:
https://deepsystems.ai

YOLO

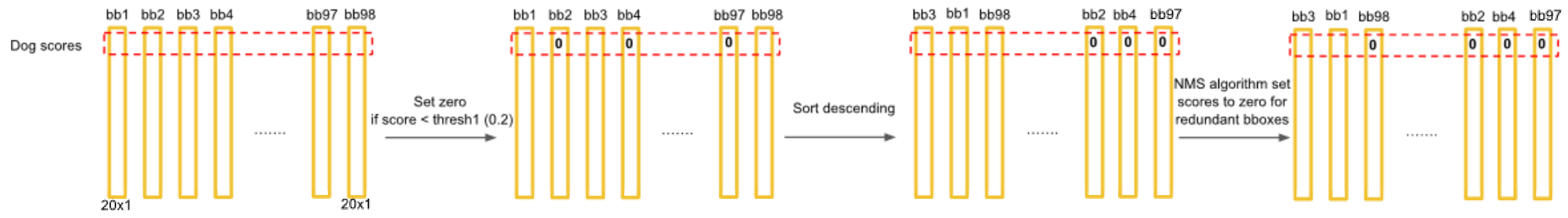

Tensor values interpretation
bb2 confidence
1x30
5
5
20
MUL
7
7
30
grid cell
7
7
20x1
20x1
Class scores for bb2
Tensor values interpretation
2 bboxes for last cell (7, 7)
bb1 bb2 bb3 bb4
bb97 bb98
grid cell (7, 7)
20x1 20x1 20x1 20x1 20x1 20x1 20x1 20x1 20x1
20x1 20x1 20x1 20x1 20x1 20x1
....

Do this operation for each bbox in each grid cell

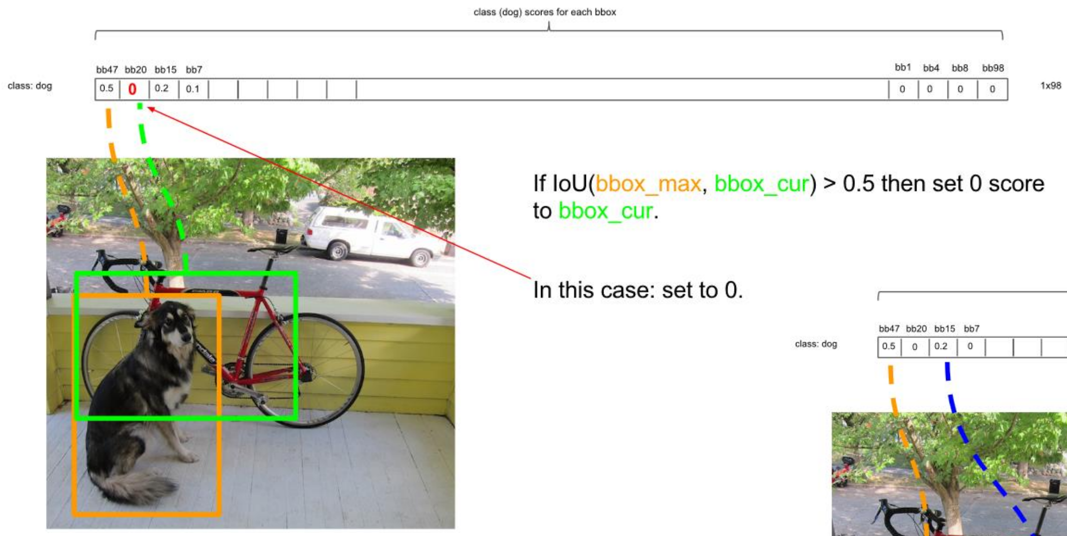
Total 7*7*2=98 bboxes

Relevant links:
https://deepsystems.ai

YOLO

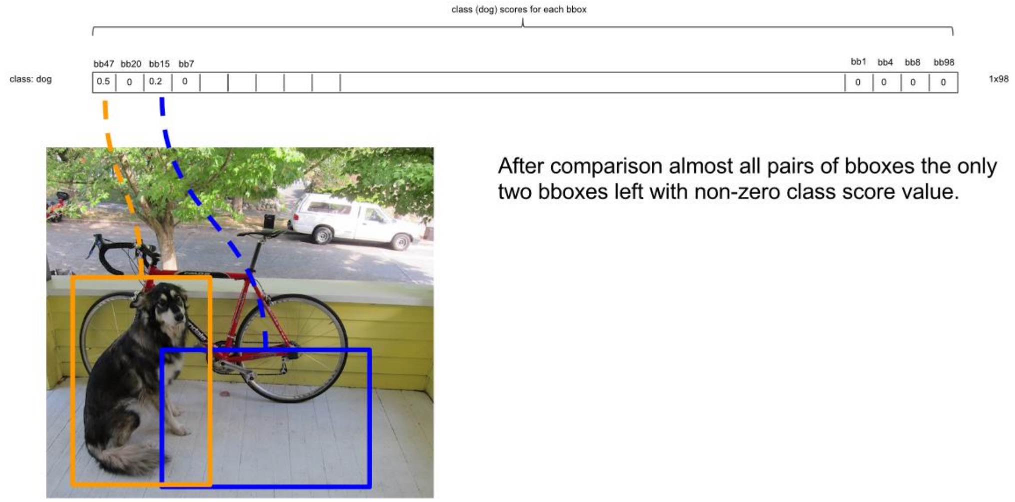

Dog scores
bb1 bb2 bb3 bb4 bb97 bb98
20x1
Set zero if score < thresh1 (0.2)
0
Sort descending
bb3 bb1 bb98 bb2 bb4 bb97
NMS algorithm set scores to zero for redundant bboxes

Non-Maximum Suppression: intuition



If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

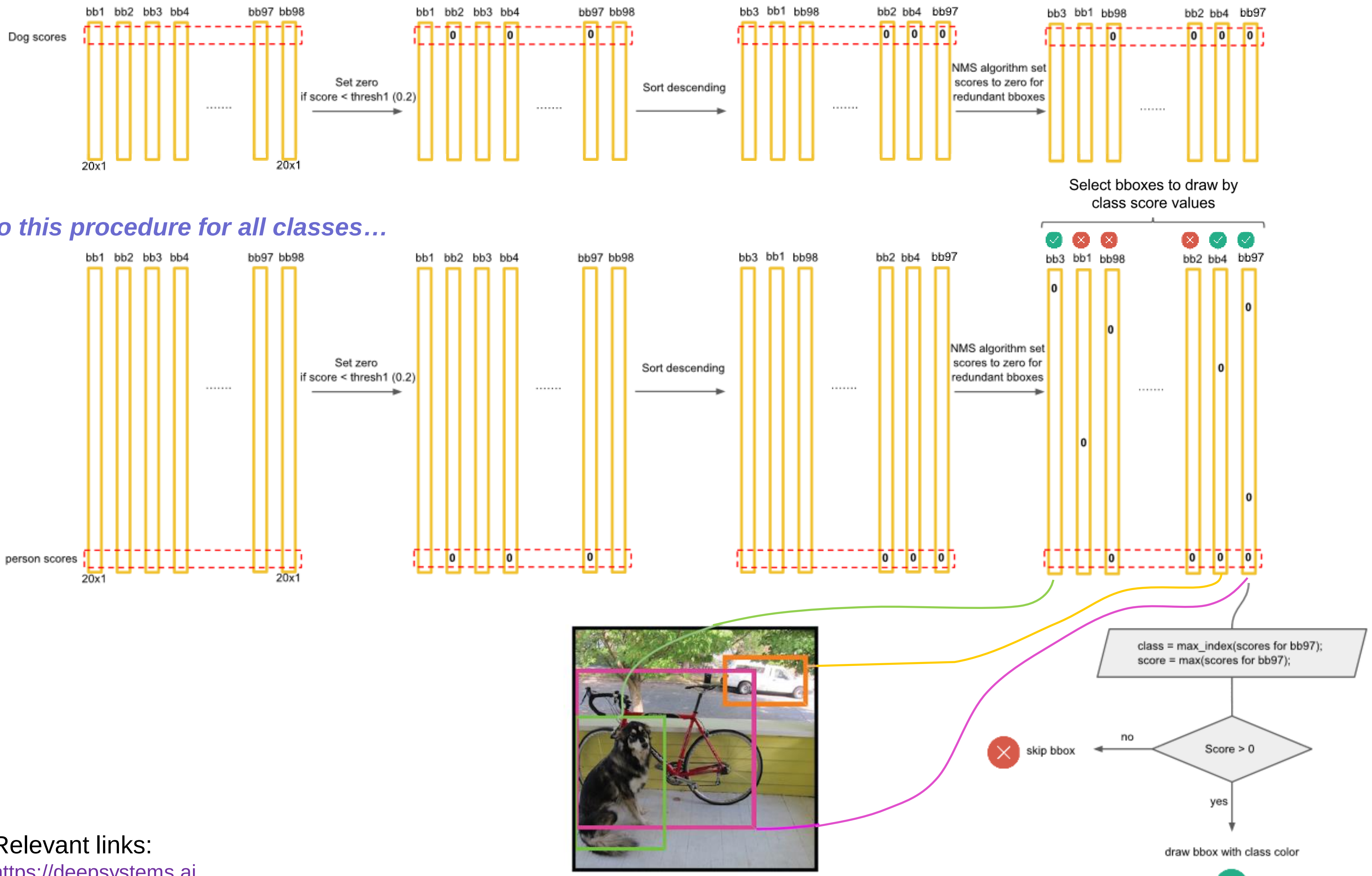
In this case: set to 0.



After comparison almost all pairs of bboxes the only two bboxes left with non-zero class score value.

Relevant links:
https://deepsystems.ai

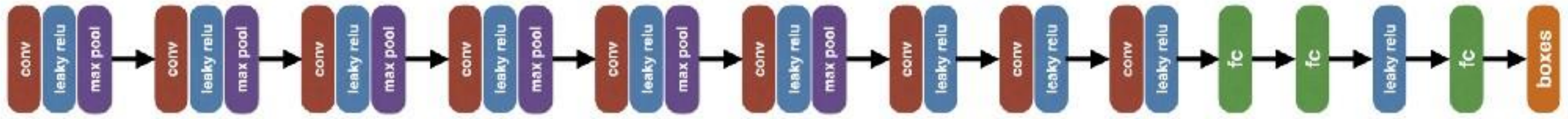
YOLO


Dog scores
bb1 bb2 bb3 bb4 bb97 bb98
20x1
Set zero if score < thresh1 (0.2)
Sort descending
bb3 bb1 bb98 bb2 bb4 bb97
NMS algorithm set scores to zero for redundant bboxes
0
Select bboxes to draw by class score values
person scores
class = max_index(scores for bb97);
score = max(scores for bb97);
Score > 0
no
skip bbox
yes
draw bbox with class color

Do this procedure for all classes…

Relevant links:
https://deepsystems.ai

YOLO



YOLO :

- Makes more localization errors, but it less likely to predict false positives on background
- Is fast, but not too precise
- Good performance in combination with fast R-CNN

In ***YOLOv1*** Image is divided to 7x7 grid where each cell predicts 2 bounding boxes. In the end, you will get a tensor value of 7*7*30. For every grid cell, you will get two bounding boxes, which will make up for the starting 10 values of the 1*30 tensor. The remaining 20 denote the number of classes.

YOLOv2 uses a few tricks to improve training and increase performance. Being trained simultaneously on several datasets (COCO detection and ImageNet classification datasets) ***YOLO9000*** (*Better, Faster, Stronger*) predicts detections for more than 9000 different object categories. The full ***YOLOv2*** model uses three times as many layers and has a slightly more complex shape, but it’s still just a regular convnet. It uses 13x13 grid where each cell predicts 5 bounding boxes. There is no fully-connected layer in YOLOv2. Last convolutional layer has a 1×1 kernel and exists to reduce the data to the shape 13×13×125.

The “tiny” version of YOLO (***Tiny YOLO***) has only these 9 convolutional layers and 6 pooling layers and can be used in mobile apps. YOLO Tiny performs 155 fps (YOLOv1 only 45 fps).

YOLO v3: *Better, not Faster, Stronger*… On top of original Darknet 53 layer network trained on Imagenet, 53 more layers are added giving us a ***106 layer fully convolutional underlying architecture*** for YOLO v3.

The architecture of Tiny YOLO:

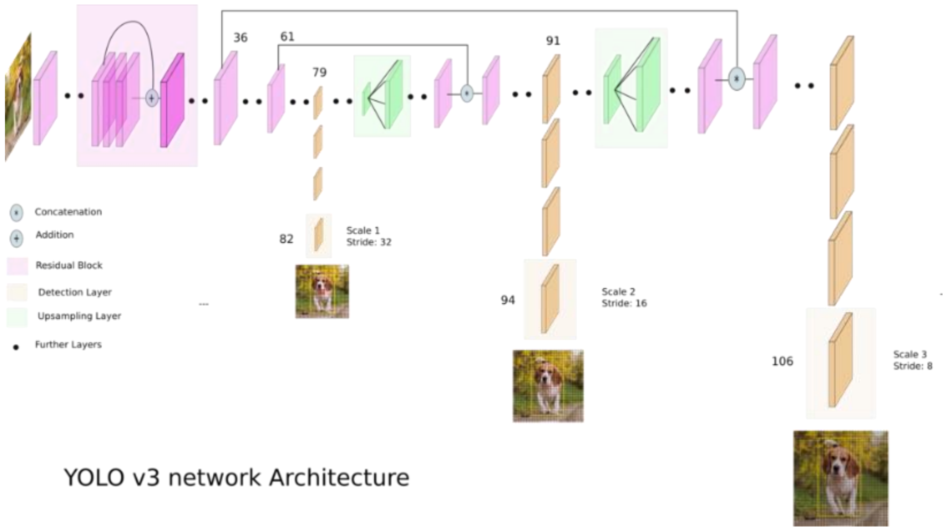
Layer	kernel	stride	output shape
Input			(416, 416, 3)
Convolution	3×3	1	(416, 416, 16)
MaxPooling	2×2	2	(208, 208, 16)
Convolution	3×3	1	(208, 208, 32)
MaxPooling	2×2	2	(104, 104, 32)
Convolution	3×3	1	(104, 104, 64)
MaxPooling	2×2	2	(52, 52, 64)
Convolution	3×3	1	(52, 52, 128)
MaxPooling	2×2	2	(26, 26, 128)
Convolution	3×3	1	(26, 26, 256)
MaxPooling	2×2	2	(13, 13, 256)
Convolution	3×3	1	(13, 13, 512)
MaxPooling	2×2	1	(13, 13, 512)
Convolution	3×3	1	(13, 13, 1024)
Convolution	3×3	1	(13, 13, 1024)
Convolution	1×1	1	(13, 13, 125)

Relevant links:
https://arxiv.org/abs/1612.08242
https://www.youtube.com/watch?v=GBu2jofRJtk
https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088
https://medium.com/diaryofawannapreneur/yolo-you-only-look-once-for-object-detection-explained-6f80ea7aaa1e
https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b
https://lilianweng.github.io/lil-log/2018/12/27/object-detection-part-4.html

YOLO

Main changes implemented in ***YOLOv3:***

- ***Logistic regression for confidence scores:*** YOLOv3 uses logistic regression to predict a confidence score for each bounding box (while YOLO and YOLOv2 uses sum of squared errors for classification terms). Linear regression of offset prediction leads to a decrease in mAP.
- ***No more softmax for class prediction:*** to predict class confidence, YOLOv3 uses multiple independent logistic classifier for each class rather than one softmax layer. It helps especially in case when one image might have multiple labels and not all the labels are guaranteed to be mutually exclusive.

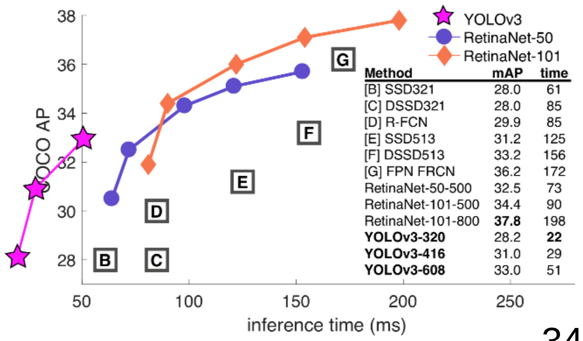


YOLO v3 network Architecture

- ***Darknet + ResNet as the base model:*** The new Darknet-53 still relies on successive 3x3 and 1x1 conv layers, just like the original dark net architecture, but has residual blocks added.
- ***Multi-scale prediction:*** Inspired by image pyramid, YOLOv3 adds several conv layers after the base feature extractor model and makes prediction at three different scales among these conv layers. Thus, it has to deal with many more bounding box candidates of various sizes overall.
- ***Skip-layer concatenation:*** YOLOv3 also adds cross-layer connections between two prediction layers (except for the output layer) and earlier finer-grained feature maps. The model first up-samples the coarse feature maps and then merges it with the previous features by concatenation. The combination with finer-grained information makes it better at detecting small objects.

Overall, ***YOLOv3*** performs better and faster than *SSD*, and worse than *RetinaNet* but 3.8x faster.

Relevant links:
https://lilianweng.github.io/lil-log/2018/12/27/object-detection-part-4.html
https://www.youtube.com/watch?v=vRqSO6RsptU

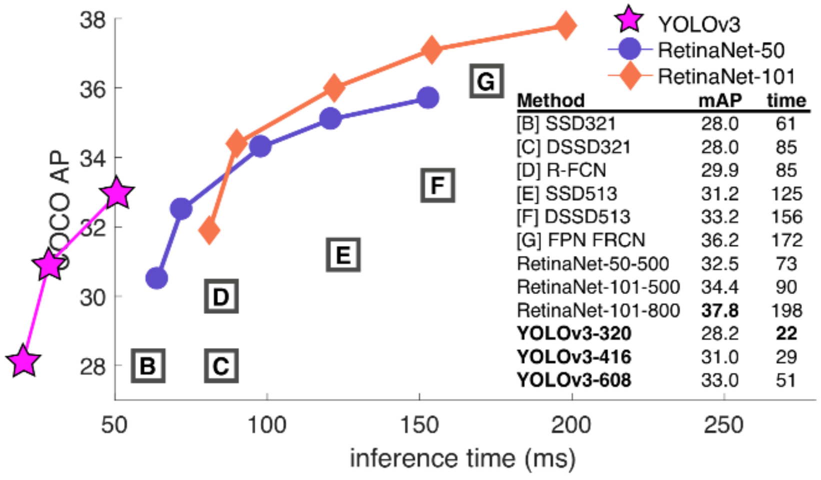


Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

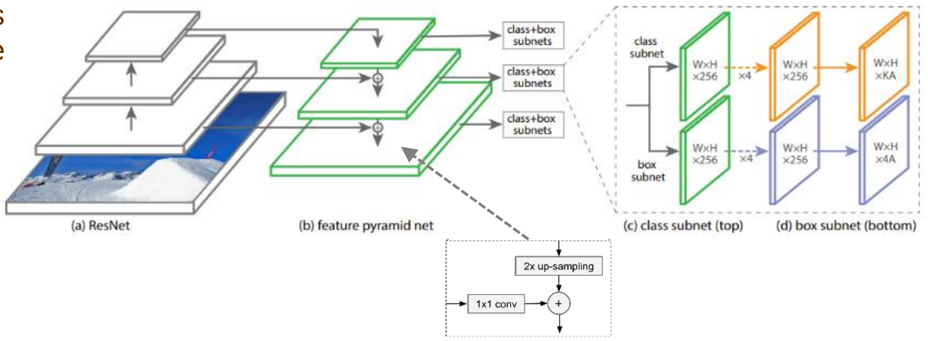
RetinaNet

RetinaNet has been formed by making two improvements over existing single stage object detection models (like *YOLO* and *SSD*) to make performance comparable to two-stage detectors:

- ***Feature Pyramid Network:*** Pyramid networks have been used conventionally to identify objects at different scales. A Feature Pyramid Network (FPN) makes use of the inherent multi-scale pyramidal hierarchy of deep CNNs to create feature pyramids.
- ***Focal Loss:*** is an improvement on cross-entropy loss that helps to reduce the relative loss for well-classified examples and putting more focus on hard, misclassified examples (since there is imbalance between regions with actual objects and variety of other "background" regions). The focal loss enables training highly accurate dense object detectors in the presence of vast numbers of easy background examples.



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



RetinaNet Object Detection (Keras):
https://keras.io/examples/vision/retinanet/

Relevant links:
https://arxiv.org/abs/1612.03144
https://arxiv.org/pdf/1708.02002.pdf
https://github.com/fizyr/keras-retinanet
https://github.com/tensorflow/tpu/tree/master/models/official/retinanet
https://towardsdatascience.com/object-detection-on-aerial-imagery-using-retinanet-626130ba2203
https://medium.com/@tabdulwahabamin/an-introduction-to-implementing-retinanet-in-keras-for-multi-object-detection-on-custom-dataset-be746024c653
https://mc.ai/object-detection-on-satellite-imagery-using-retinanet-part-1%E2%80%8A-%E2%80%8Atraining/
https://medium.com/@14prakash/the-intuition-behind-retinanet-eb636755607d
https://www.freecodecamp.org/news/object-detection-in-colab-with-fizyr-retinanet-efed36ac4af3/
https://www.tejashwi.io/object-detection-with-fizyr-retinanet/

EfficientDet and YOLO

EfficientDet and *YOLO v4 (v5)*

The ***EfficientDet*** family of models has been the preferred object detection models since it was published by the Google Research/Brain team. They released their own ConvNet model called ***EfficientNet.*** It set out to define an automatic procedure for scaling ConvNet model architectures, to optimize downstream performance given free range over depth, width, and resolution while staying within the constraints of target memory and target FLOPs. Example to train EfficientDet using a pytorch implementation on a custom dataset that has been uploaded through ***RoboFlow*** (https://colab.research.google.com/drive/1ZmbeTro4SqT7h_TfW63MLdqbrCUk_1br#scrollTo=sAs6vn4Rukct).

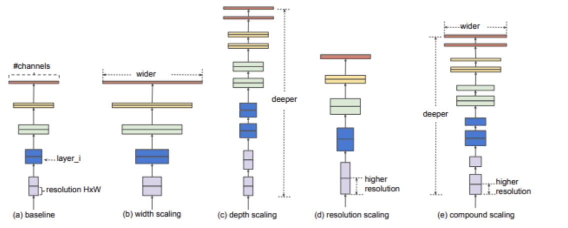


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

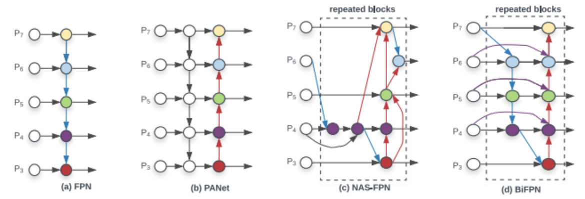
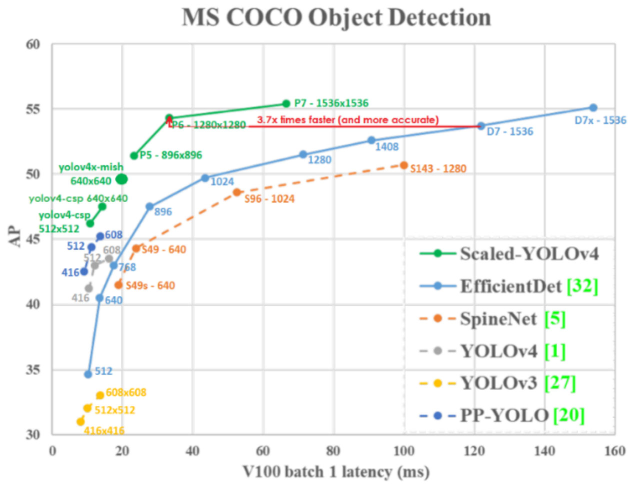


Figure 2: **Feature network design** – (a) FPN [20] introduces a top-down pathway to fuse multi-scale features from level 3 to 7 (P_3 - P_7); (b) PANet [23] adds an additional bottom-up pathway on top of FPN; (c) NAS-FPN [8] use neural architecture search to find an irregular feature network topology and then repeatedly apply the same block; (d) is our BiFPN with better accuracy and efficiency trade-offs.

When published, ***YOLOv4*** was considered one of the best models for speed and accuracy performance but did not top EfficientDet's largest model for overall accuracy on the COCO dataset.

- considers the following backbones (base networks): *CSPResNext50* and *CSPDarknet53* that are based on DenseNet, and *EfficientNet-B3*.
- considers a few options for the neck including: *FPN, PAN, NAS-FPN, BiFPN, ASFF, SFAM*.
- deploys the same YOLO head as *YOLOv3* for detection with the anchor-based detection steps and three levels of detection granularity.
- employs a *"Bag of Freebies" (most of which have to do with data augmentation)* to improve performance of the network without adding to inference time in production.
- deploys strategies called a "Bag of Specials“ to add marginal increases to inference time but significantly increase performance (e.g. Mish activation function).
- use *DIoU NMS, Cross mini-Batch Normalization (CmBN), DropBlock regularization*

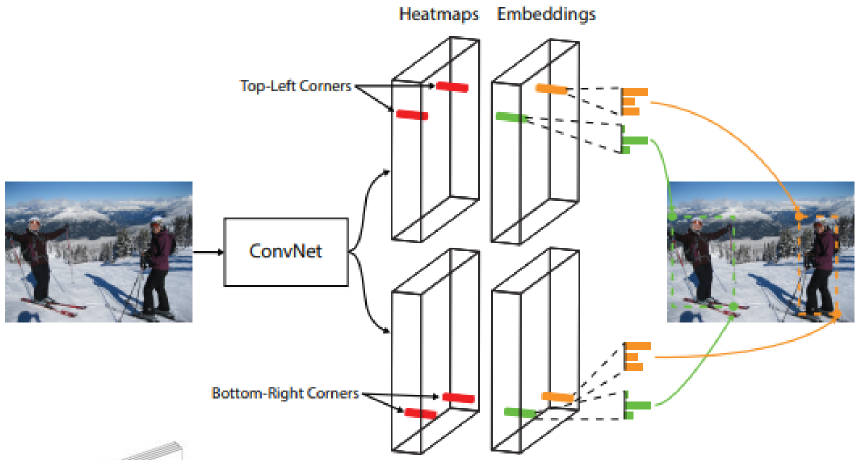


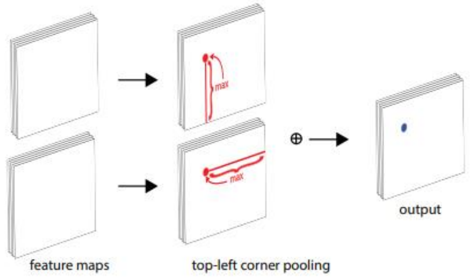
Relevant links:
https://arxiv.org/pdf/1911.09070v7.pdf
https://blog.roboflow.com/breaking-down-efficientdet/
https://jonathan-hui.medium.com/yolov4-c9901eaa8e61
https://blog.roboflow.com/a-thorough-breakdown-of-yolov4/
https://github.com/AlexeyAB/darknet
https://towardsdatascience.com/yolo-v4-or-yolo-v5-or-pp-yolo-dad8e40f7109

CornerNet and CenterNet

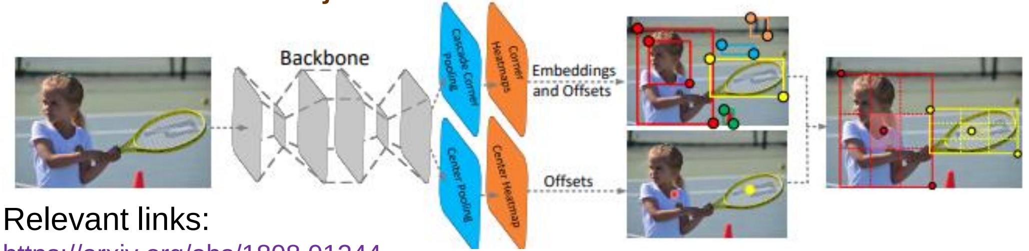
CornerNet is attempt to get rid of the anchors.

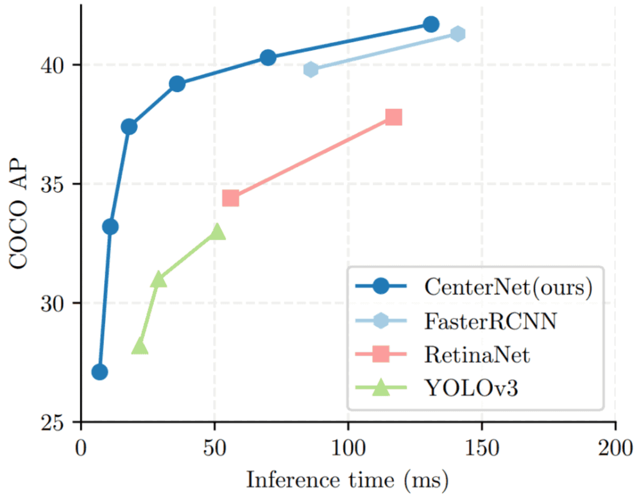
- bounding boxes are expressed with 2 points (top-left and bottom-right corners).
- a convNet outputs a heatmap for all top-left corners, a heatmap for all bottom-right corners, and an embedding vector (kind of identification signature) for each detected corner.
- the network is trained to predict similar embeddings for corners that belong to the same object.
- is based on *Hourglass Network* (originally proposed in 2016).
- applies corner pooling





CenterNet using the corners as proposals, focuses on the center of the object to infer its class and filter out outliers.

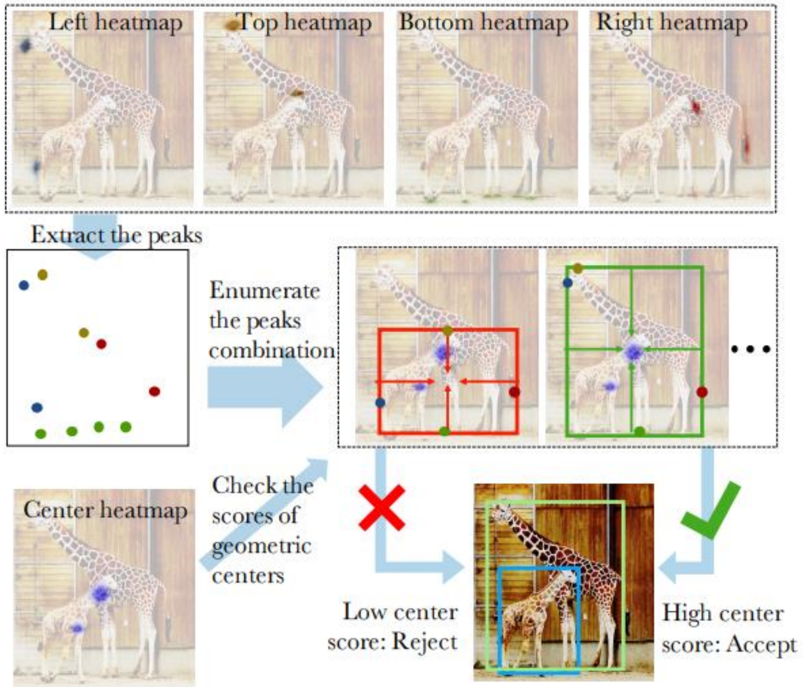




Relevant links:
https://arxiv.org/abs/1808.01244
https://arxiv.org/abs/1904.08189
https://github.com/princeton-vl/CornerNet
https://www.youtube.com/watch?v=aJnvTT1-spc

ExtremeNet

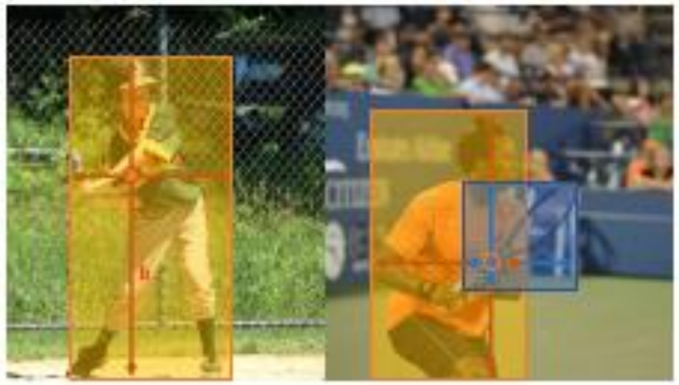
ExtremeNet follows the idea to represent objects by their extreme points instead of bounding boxes or corners. In combination with center heatmap prediction, it selects/filters the appropriate set of extreme points that belong to the object.

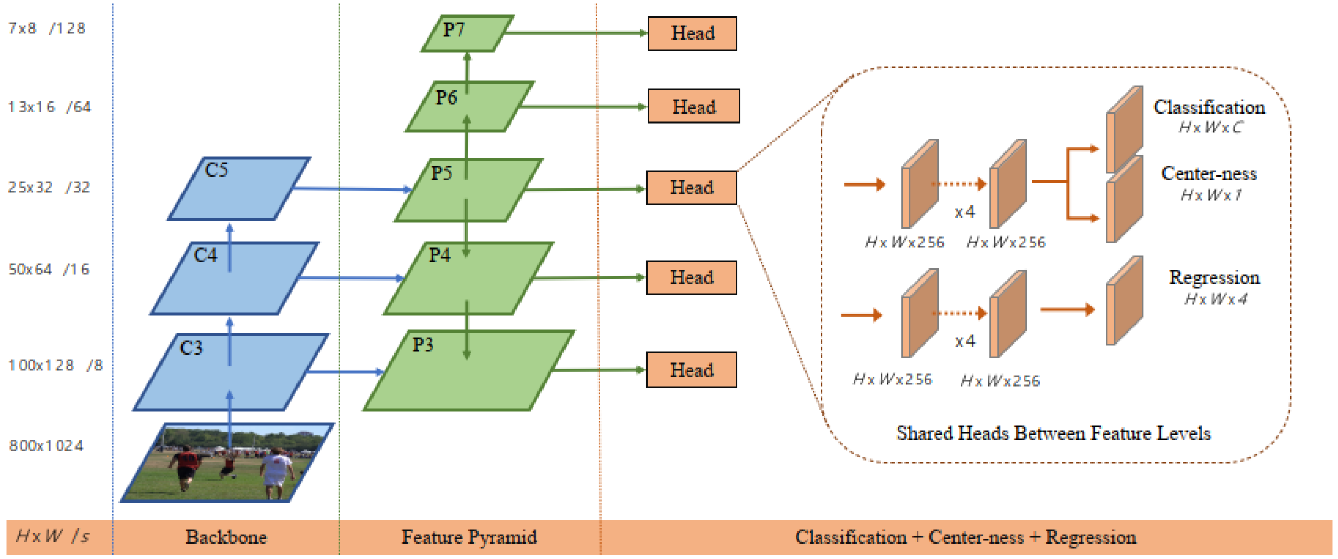


Relevant links:
https://arxiv.org/abs/1901.08043

Fully Convolutional One-Stage Object Detection

FCOS solves object detection in a per-pixel prediction fashion, analogue to semantic segmentation. In contrast to majority of state-of-the-art object detectors such as RetinaNet, SSD, YOLO (v2, v3, v4 etc.), and Faster R-CNN that rely on pre-defined anchor, FCOS detector is ***anchor box free***, as well as ***proposal free*** (without Region Proposal Network).





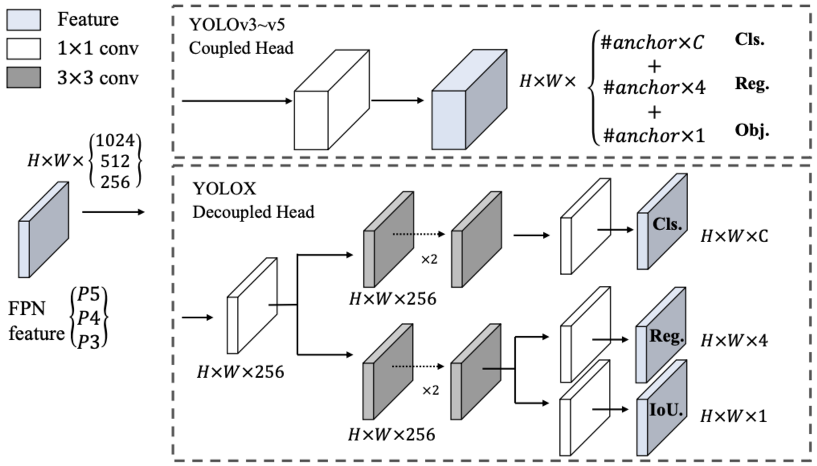
As shown in the left image, FCOS works by predicting a 4D vector (l, t, r, b) encoding the location of a bounding box at each foreground pixel (supervised by ground-truth bounding box information during training). The right plot shows that when a location residing in multiple bounding boxes, it can be ambiguous in terms of which bounding box this location should regress

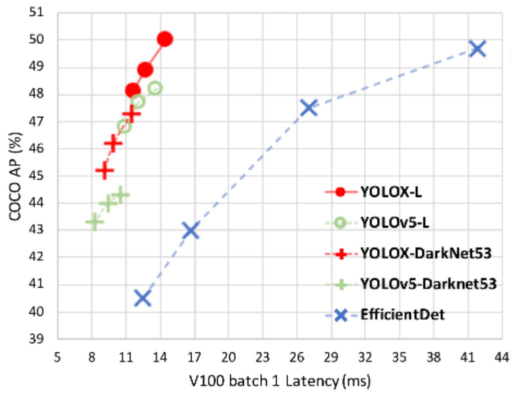
Relevant links:
https://arxiv.org/abs/1904.01355
https://github.com/tianzhi0549/FCOS
https://www.youtube.com/watch?v=_ADYE6QaAAY

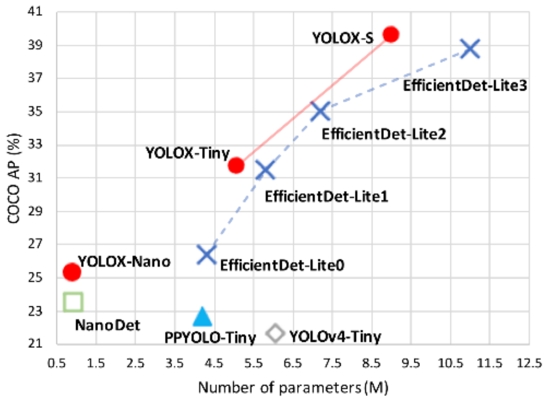
YOLOX

YOLOX is an ***anchor-free*** version of YOLO, with a simpler design but better performance! It aims to bridge the gap between research and industrial communities.

YOLOX presents improvements to YOLO series, forming a new high-performance detector. YOLO detector is switched to an anchor-free manner and conduct other advanced detection techniques, i.e., a decoupled head and the leading label assignment strategy SimOTA to achieve state-of-the-art results across a large-scale range of models…







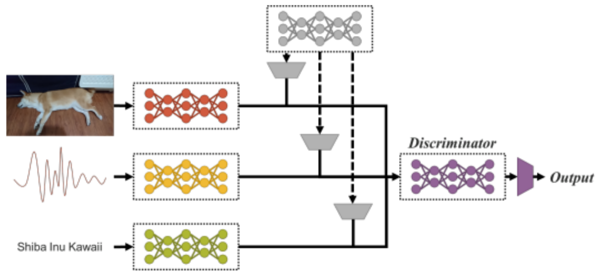
Relevant links:
https://arxiv.org/abs/2107.08430
https://github.com/Megvii-BaseDetection/YOLOX

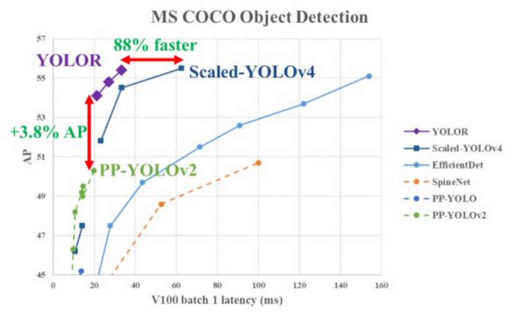
YOLOR

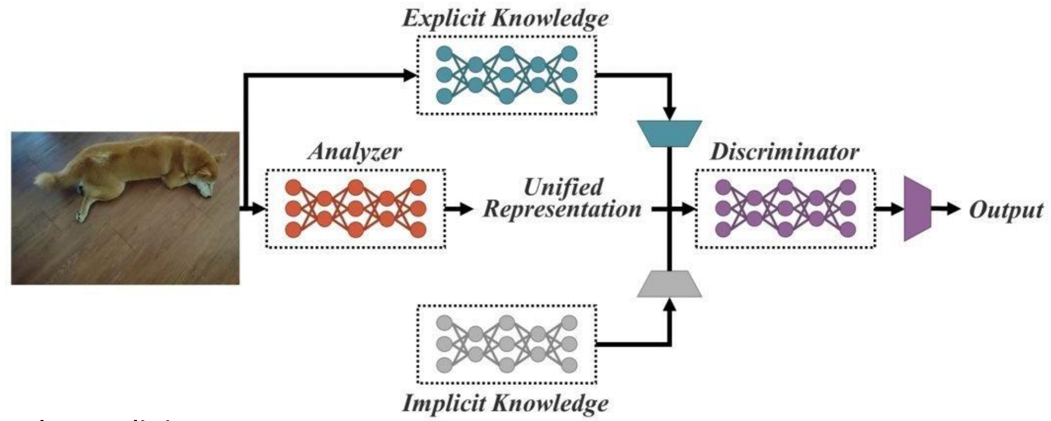
YOLOR - You Only Learn One Representation: Unified Network for Multiple Tasks

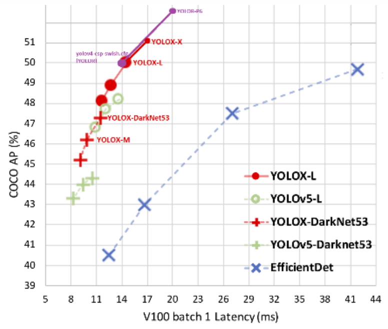
Approach behind the YOLOR combines ***explicit knowledge***, defined as learning based on given data and input, with ***implicit knowledge*** learned subconsciously. Therefore, the concept of YOLOR is based on encoding implicit and explicit knowledge together, similar to how mammalian brains process implicit and explicit knowledge in conjunction with each other. The unified network proposed in YOLOR generates a ***unified representation*** to serve a variety of tasks all at once.

YOLOR aims to give this ability to machine learning models – so that they are able to serve many tasks given one input. It is a unified model effective for multi-task (e.g. object detection, instance segmentation, key-points detection, image capturing, etc.) learning under the single model architecture. In the future, authors are going to extend the training to multi-modal and multi-task.





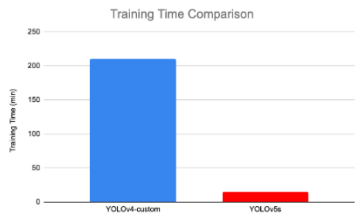




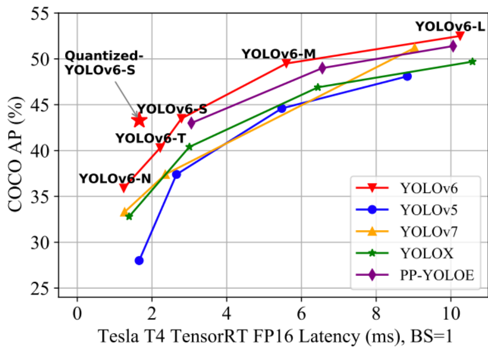
Relevant links:
https://arxiv.org/pdf/2105.04206v1.pdf
https://viso.ai/deep-learning/yolor/
https://blog.roboflow.com/train-yolor-on-a-custom-dataset/
https://medium.com/augmented-startups/yolor-vs-yolox-battle-of-the-object-detection-prodigies-ae004a5ac8d2

YOLO…

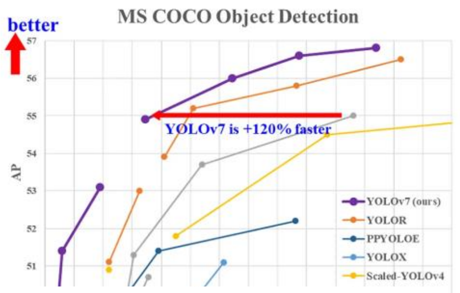
YOLOv5 Released by Glenn Jocher in June 2020, YOLOv5, similarly to YOLOv4, uses CSPDarknet53 as the backbone of its architecture. The release includes five different model sizes: YOLOv5s (smallest), YOLOv5m, YOLOv5l, and YOLOv5x (largest). One of the major improvements in YOLOv5 architecture is the integration of the *Focus layer*, represented by a single layer, which is created by replacing the first three layers of YOLOv3. Thus, it reduces the number of layers and number of parameters, increasing both forward and backward speed without any major impact on the mAP. https://blog.roboflow.com/yolov5-is-here/ , https://blog.roboflow.com/yolov4-versus-yolov5/



YOLOv6 - A Single-Stage Object Detection Framework for Industrial Applications. Dedicated to industrial applications with hardware-friendly efficient design and high performance, the YOLOv6 (MT-YOLOv6) framework was released by Meituan, a Chinese e-commerce company. Written in Pytorch, this new version was not part of the official YOLO but still got the name YOLOv6 because its backbone was inspired by the original one-stage YOLO architecture. YOLO v6 uses a variant of the EfficientNet architecture called EfficientNet-L2, introducing three significant improvements: a hardware-friendly backbone and neck design, an efficient decoupled head, and a more effective training strategy. YOLO v6 also introduces a new method for generating the anchor boxes, called "dense anchor boxes."

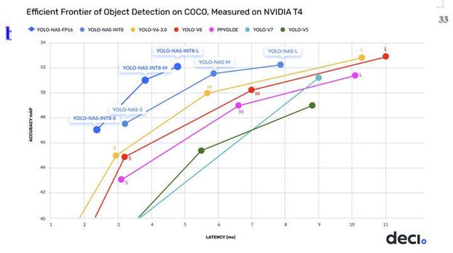


YOLOv7 was released in July 2022 in the paper "*Trained bag-of-freebies sets new state-of-the-art for real-time object detectors*". A key improvement in YOLO v7 is the use of a new loss function called "focal loss." Previous versions of YOLO used a standard cross-entropy loss function, which is known to be less effective at detecting small objects. Focal loss battles this issue by down-weighting the loss for well-classified examples and focusing on the hard examples—the objects that are hard to detect. YOLO v7 also has a higher resolution than the previous versions. https://arxiv.org/pdf/2207.02696.pdf
https://viso.ai/deep-learning/yolov7-guide/ , https://huggingface.co/spaces/akhaliq/yolov7



YOLOv8 is the latest installment in the highly influential family of models that use the YOLO (You Only Look Once) architecture. YOLOv8 was developed by Ultralytics (a team known for its work on YOLOv3 and YOLOv5) and announced in the beginning of 2023.
https://blog.roboflow.com/whats-new-in-yolov8/

YOLO-NAS is the state-of-the-art object detection model published in May 2023 by an Israel-based company Deci. It is tailored for efficiently detecting small objects and boasts enhanced localization precision. It also improves the performance-to-compute ratio, making it ideal for real-time edge-device applications.
https://learnopencv.com/yolo-nas/ , https://blog.roboflow.com/yolo-nas-how-to-train-on-custom-dataset/

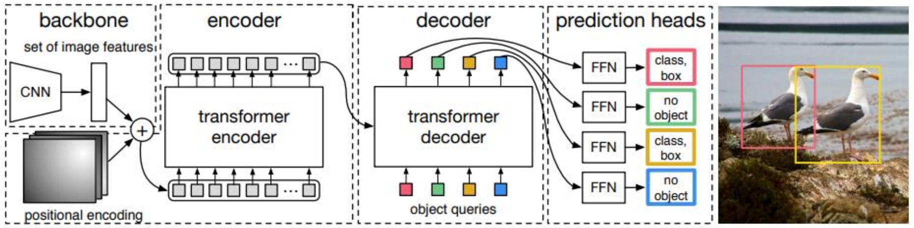


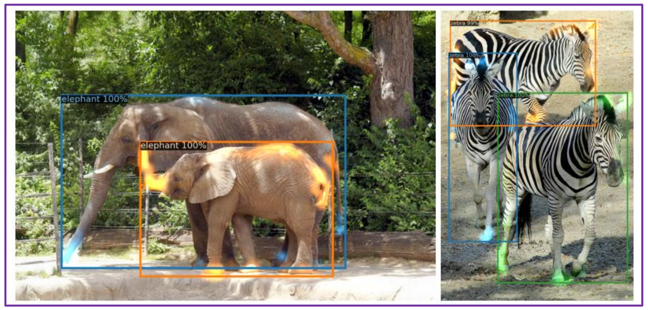
YOLO-World is a zero-shot object detection model. Shows a significant advancement in the field of open-vocabulary object detection. https://roboflow.com/model/yolo-world

Relevant links:
https://www.datacamp.com/blog/yolo-object-detection-explained
https://www.v7labs.com/blog/yolo-object-detection
https://learnopencv.com/mastering-all-yolo-models/

DEtection TRansformer (DERT)

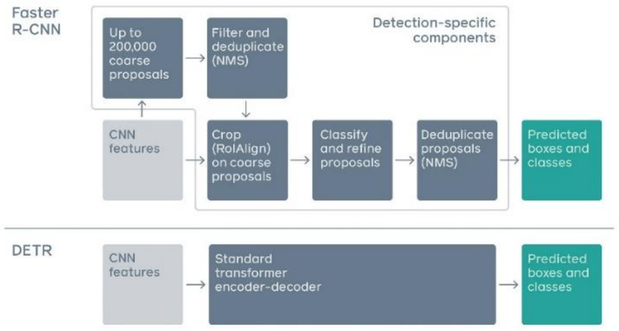
Transformers are a deep learning architecture that has gained popularity… They rely on a simple, yet powerful mechanism called *attention*, which enables AI models to selectively focus on certain parts of their input and thus reason more effectively. Even though originally transformers have been widely applied on problems with sequential data (in particular, in natural language processing (NLP) tasks such as language modeling and machine translation, as well as, extended to tasks of speech recognition, symbolic mathematics, and reinforcement learning), now this approach also shows good results for computer vision tasks like image classification and object detection… .





DETR - streamlines the detection pipeline, effectively removing the need for many hand-designed components like a non-maximum suppression procedure or anchor generation that explicitly encode our prior knowledge about the task. The main ingredients of the new framework are a set-based global loss that forces unique predictions via bipartite matching, and a transformer encoder-decoder architecture. Given a fixed small set of learned object queries, DETR reasons about the relations of the objects and the global image context to directly output the final set of predictions in parallel. The new model is conceptually simple and does not require a specialized library, unlike many other modern detectors.

DETR demonstrates accuracy and run-time performance on par with the well-established and highly-optimized Faster RCNN baseline on the challenging COCO object detection dataset. Moreover, DETR can be easily generalized to produce panoptic segmentation in a unified manner. Authors show that it significantly outperforms competitive baselines.


Faster R-CNN
Detection-specific components
Up to 200,000 coarse proposals
Filter and deduplicate (NMS)
CNN features
Crop (RoIAlign) on coarse proposals
Classify and refine proposals
Deduplicate proposals (NMS)
Predicted boxes and classes
DETR
CNN features
Standard transformer encoder-decoder
Predicted boxes and classes

A new iteration of DETR — known as ***RT-DETR*** or real-time DETR —was published in 2023, claiming superior performance in both speed and accuracy compared to all YOLO detectors of similar scale.

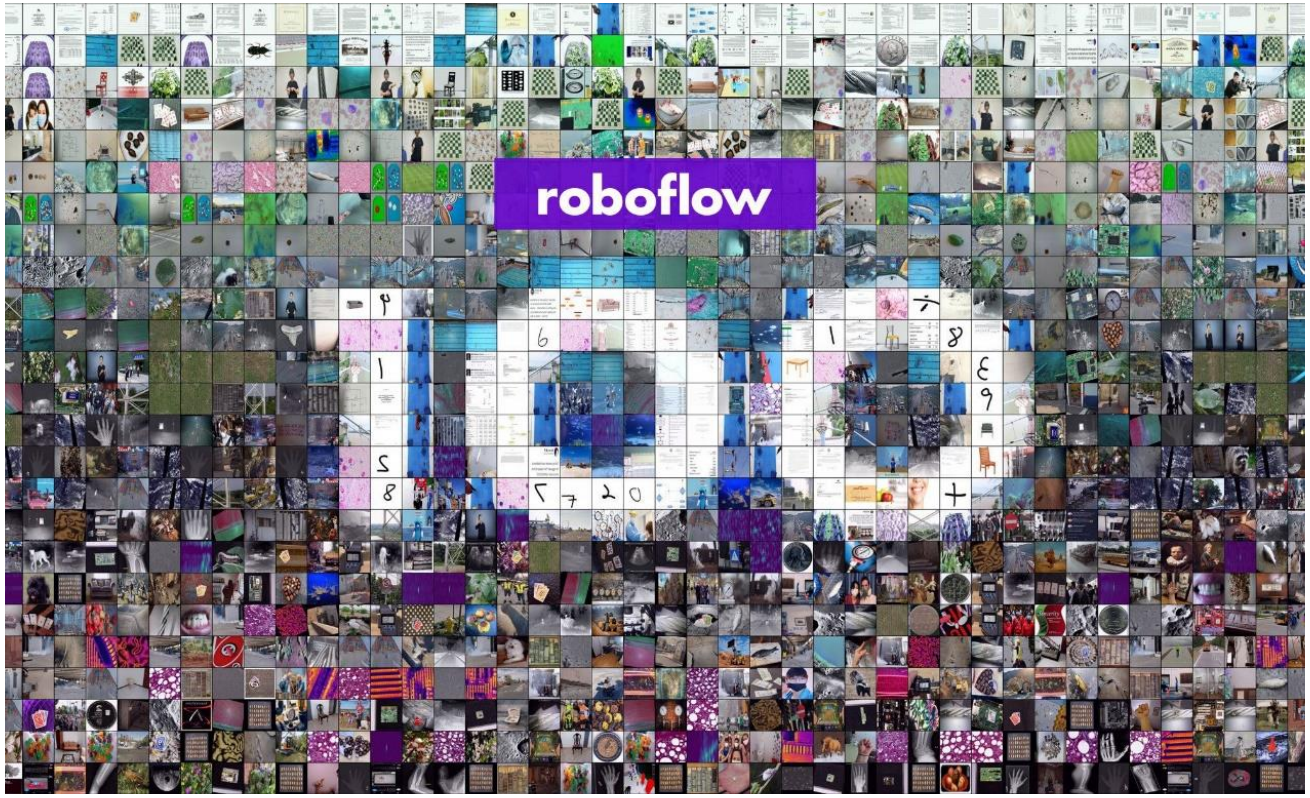
Relevant links:

https://arxiv.org/abs/2005.12872
https://arxiv.org/abs/2304.08069
https://ai.facebook.com/blog/end-to-end-object-detection-with-transformers/
https://github.com/facebookresearch/detr

Roboflow 100

Roboflow 100: A New Object Detection Benchmark advancing the state-of-the-art in object recognition with a new way to benchmark computer vision models across domains and task targets. Introduced November 2022, presented at CVPR in June 2023. https://www.rf100.org/

RF100 is a crowdsourced, open-source object detection benchmark. It consists of 100 datasets, 7 imagery domains, 224,714 images, and 829 class labels with over 11,170 labeling hours. RF100 is an Intel sponsored initiative that aims to create an accessible, transparent and open-sourced benchmark for machine learning object detection models and assert generality using in the wild crawled datasets.



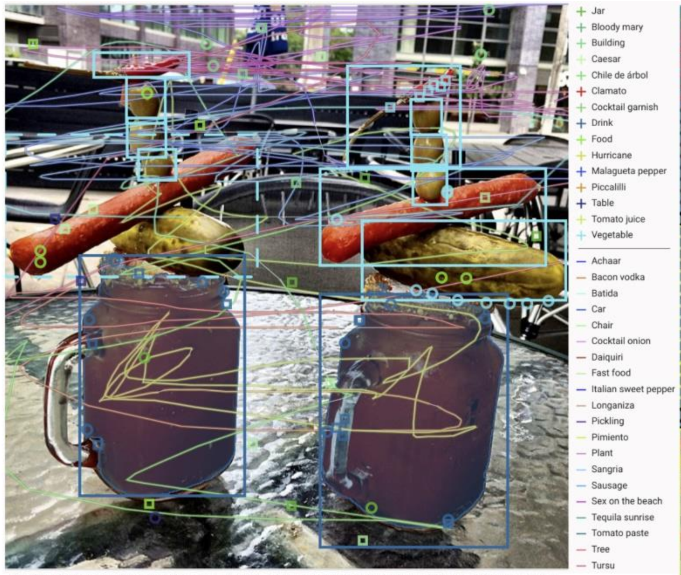
Relevant links:
https://universe.roboflow.com/
https://github.com/roboflow/roboflow-100-benchmark
https://blog.roboflow.com/roboflow-100/

Open Images Dataset

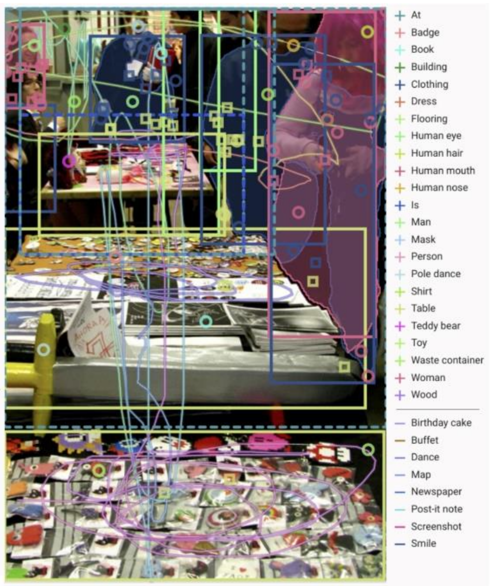
Open Images Dataset V7

Open Images is a dataset of ~9M images annotated with image-level labels, object bounding boxes, object segmentation masks, visual relationships, and localized narratives…

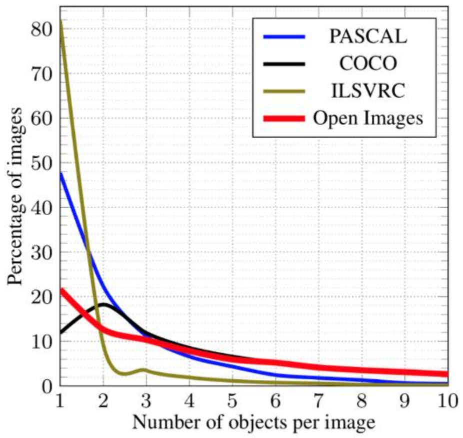
Authors believe that having a single dataset with unified annotations for image classification, object detection, visual relationship detection, instance segmentation, and multimodal image descriptions will enable to study these tasks jointly and stimulate progress towards genuine scene understanding.

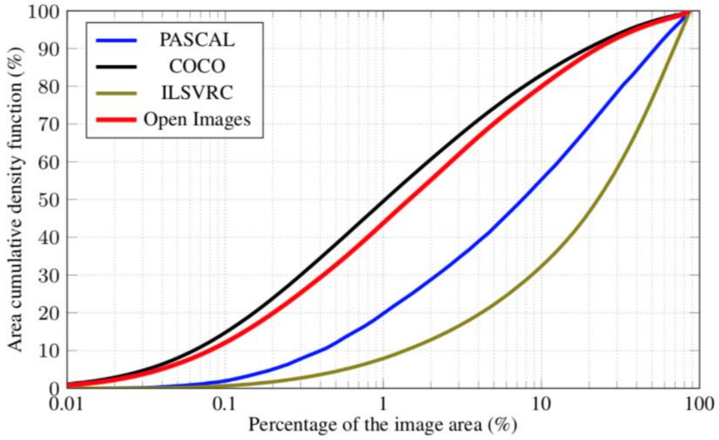


"In the foreground of the picture there are jars, drink, sausage and other food items placed on a table.
In the center of the picture there are chairs, tables and other objects.
In the background there are buildings, trees, cars, footpath and other objects."



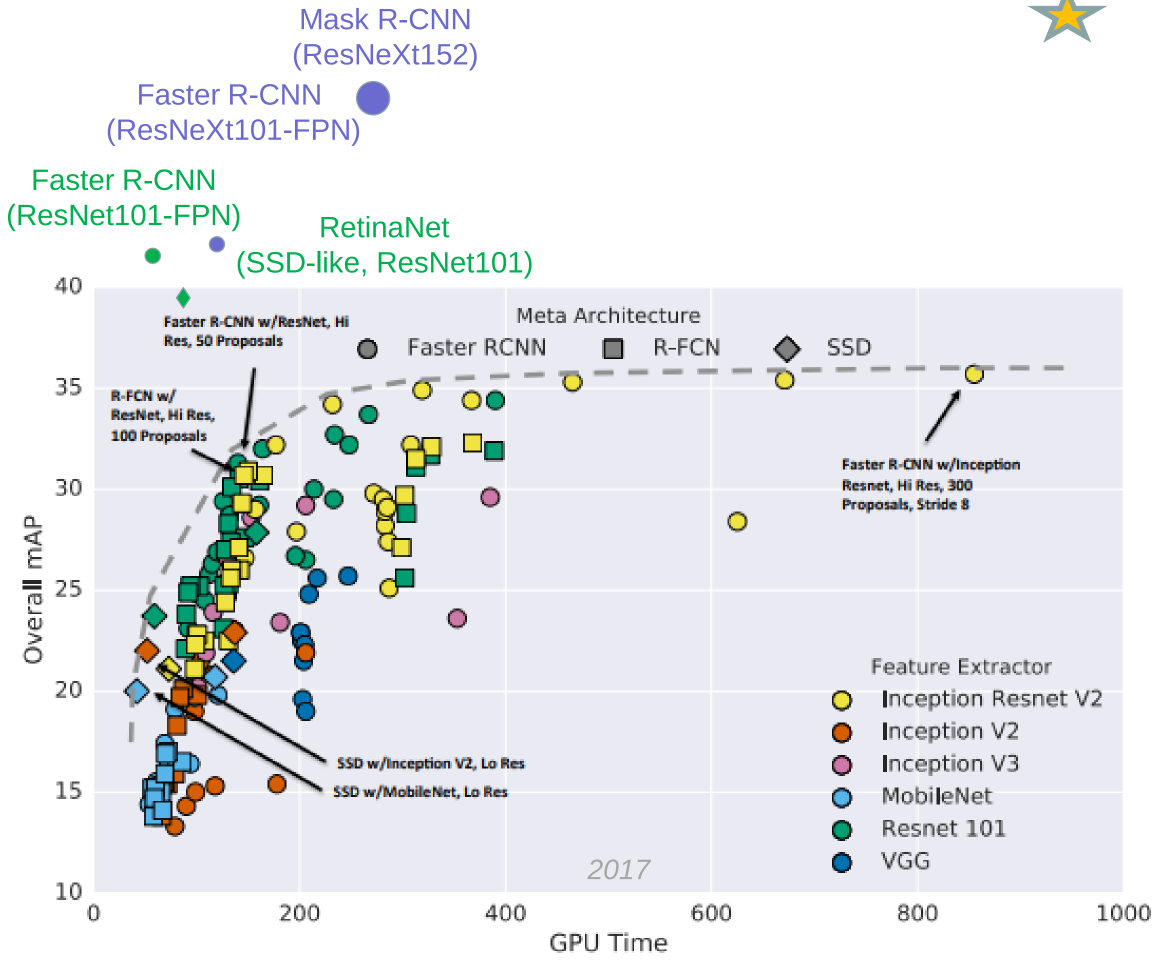
"As we can see in the image there are few people wall, mirror and tables.
On tables there are books, papers and covers."





Relevant links:
https://storage.googleapis.com/openimages/web/factsfigures_v7.html

Object detection



Extra tricks:

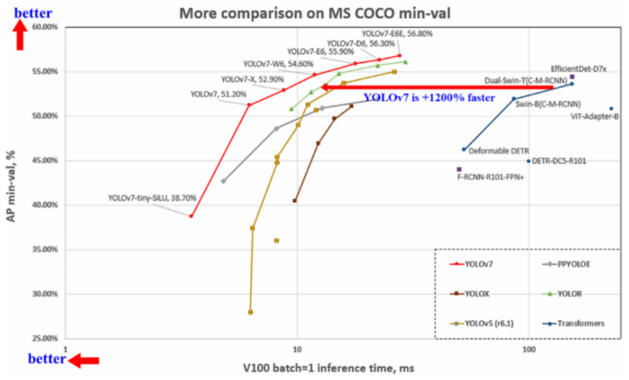
- Feature Pyramid Networks (multiscale backbone)
- Play with backbone network
- Improvement of Single-Stage methods (e.g. RetinaNet)
- Use larger models
- Test-time augmentation, big ensembles, more data, etc.

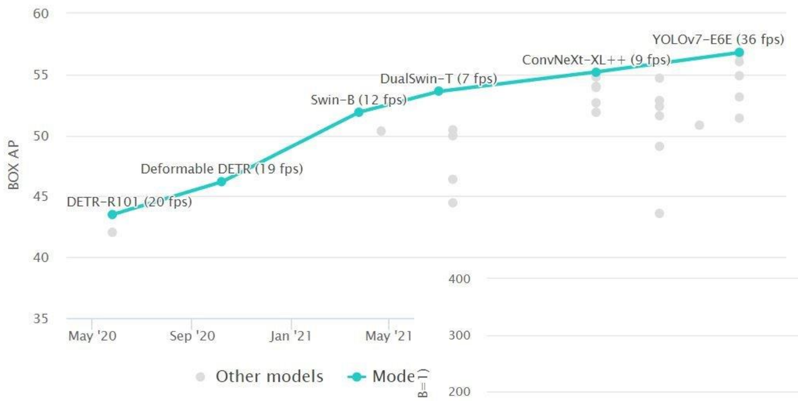
Relevant links:
https://arxiv.org/abs/1611.10012

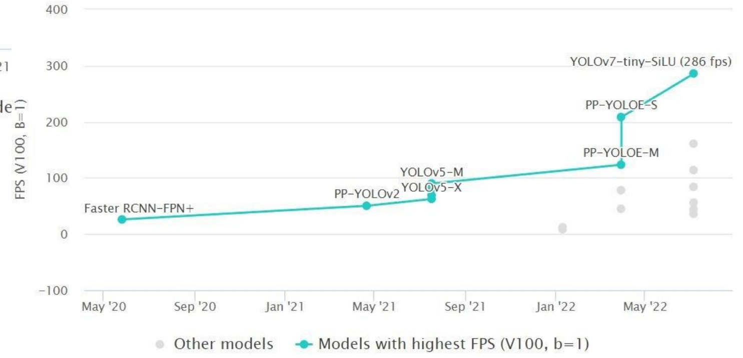
Object Detection models

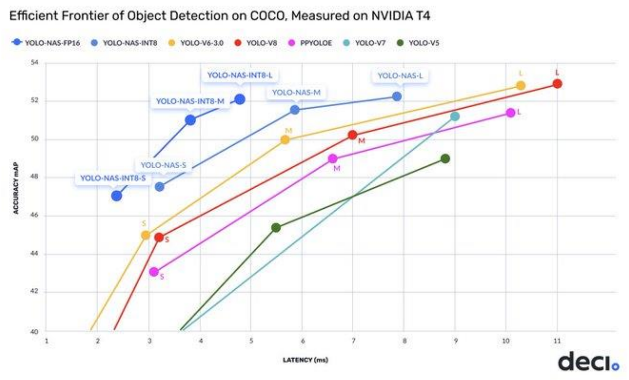
The best real-time object detection algorithm (Accuracy)

On the MS COCO dataset and based on the Average Precision (AP), the best real-time object detection algorithm in 2023 is YOLOv8, followed by YOLOv7 (2022), Vision Transformer (ViT) such as Swin and DualSwin, PP-YOLOE, YOLOR, YOLOv4, and EfficientDet…

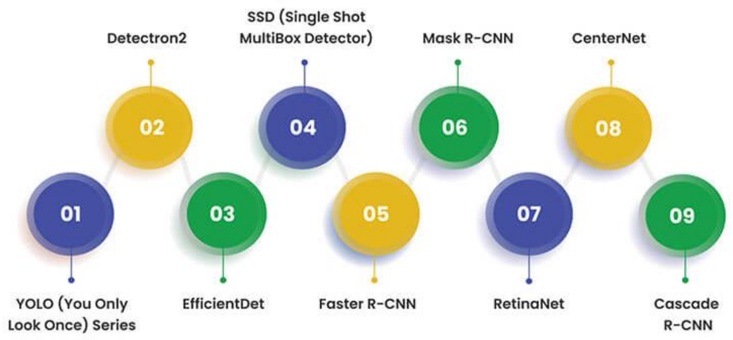








Relevant links:
https://viso.ai/deep-learning/object-detection/
https://dagshub.com/blog/best-object-detection-models/
https://www.hitechbpo.com/blog/top-object-detection-models.php
https://www.linkedin.com/pulse/top-10-object-detection-models-2023-bhavesh-kapil/



Some extra relevant links…

- https://paperswithcode.com/task/object-detection
- https://medium.com/@RaghavPrabhu/a-simple-tutorial-to-classify-images-using-tensorflow-step-by-step-guide-7e0fad26c22
- https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751
- http://openaccess.thecvf.com/content_cvpr_2017/papers/Huang_SpeedAccuracy_Trade-Offs_for_CVPR_2017_paper.pdf
- https://viso.ai/deep-learning/object-detection/