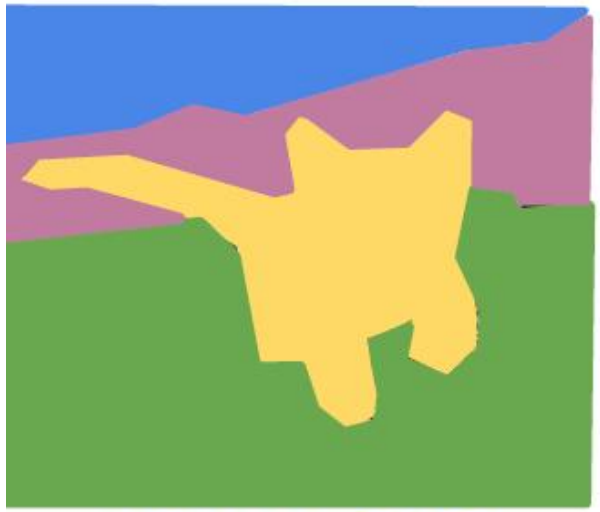
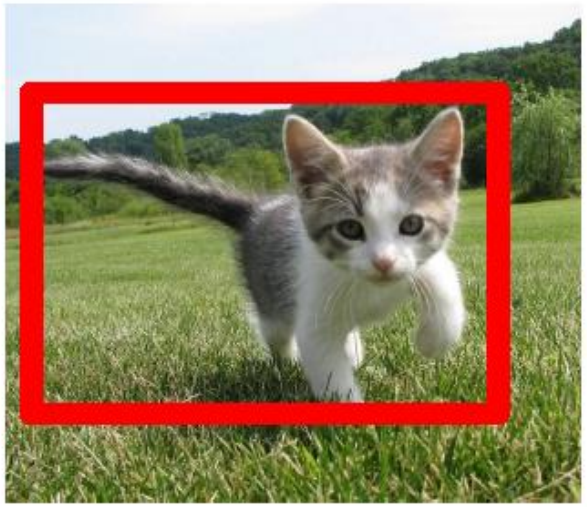
CLASSIFICATION AND LOCALIZATION

Classification + Localization



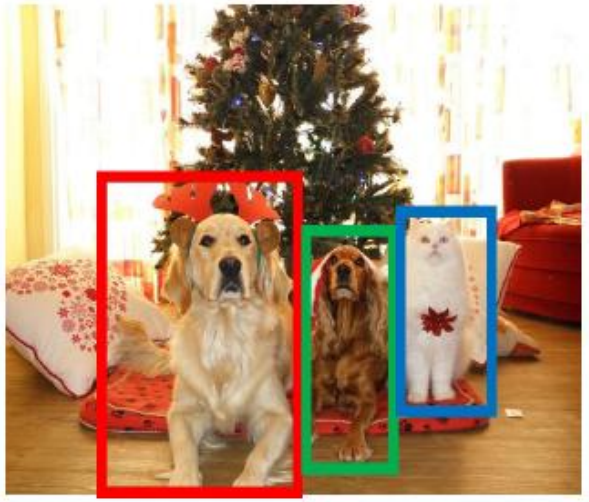
GRASS, CAT, TREE, SKY

No objects, just pixels

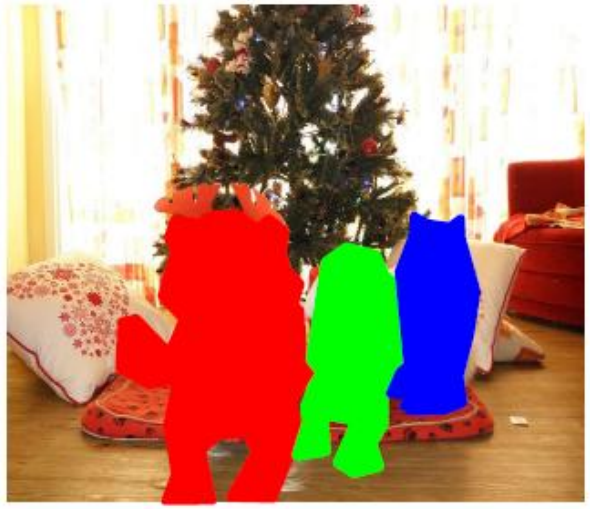


CAT

Single Object



DOG, DOG, CAT



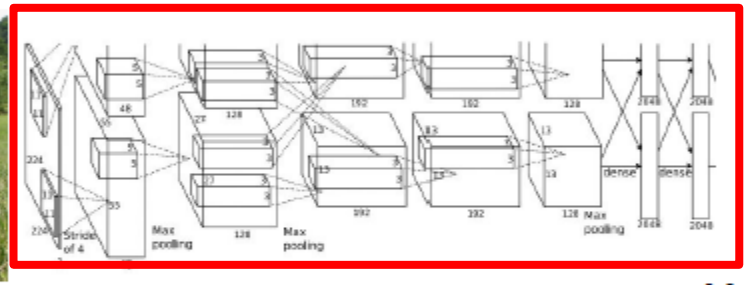
DOG, DOG, CAT

Multiple Object

Source: cs231n

Classification + Localization

This image is CC0 public domain



Often pretrained on ImageNet
(Transfer learning)

Vector:
4096

Fully Connected:
4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Correct label:
Cat

Multitask Loss

+

Loss

Fully Connected:
4096 to 4

Box Coordinates
(x, y, w, h)

L2 Loss

Correct box:
(x', y', w', h')

Treat localization as a regression problem!

Source: CS231n

Aside: Human Pose Estimation

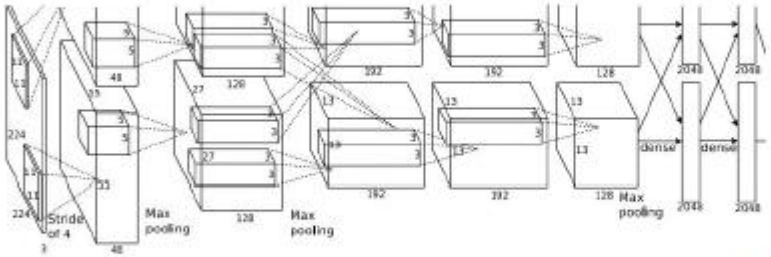
Represent pose as a set of 14 joint positions:

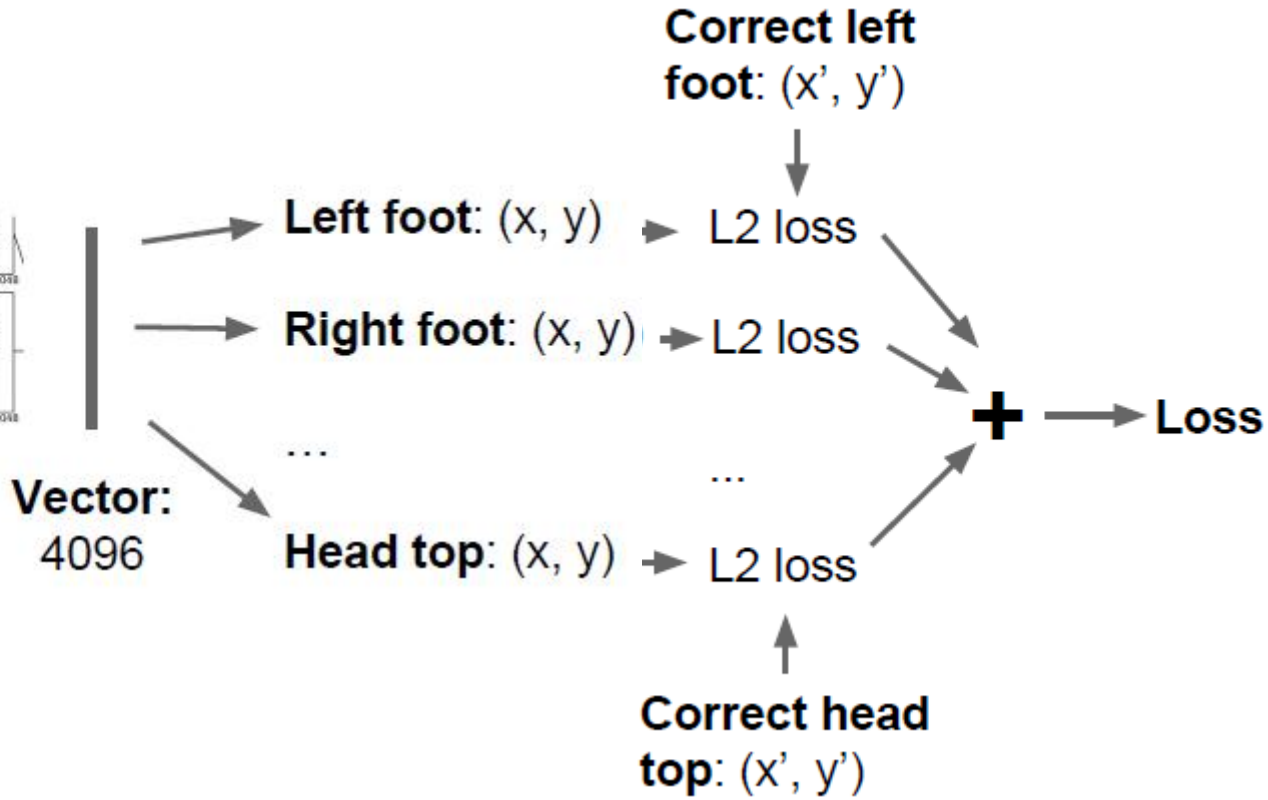
Left / right foot
Left / right knee
Left / right hip
Left / right shoulder
Left / right elbow
Left / right hand
Neck
Head top

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

Source: cs231n

Human Pose Estimation





Toshev and Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks", CVPR 2014
CS231n

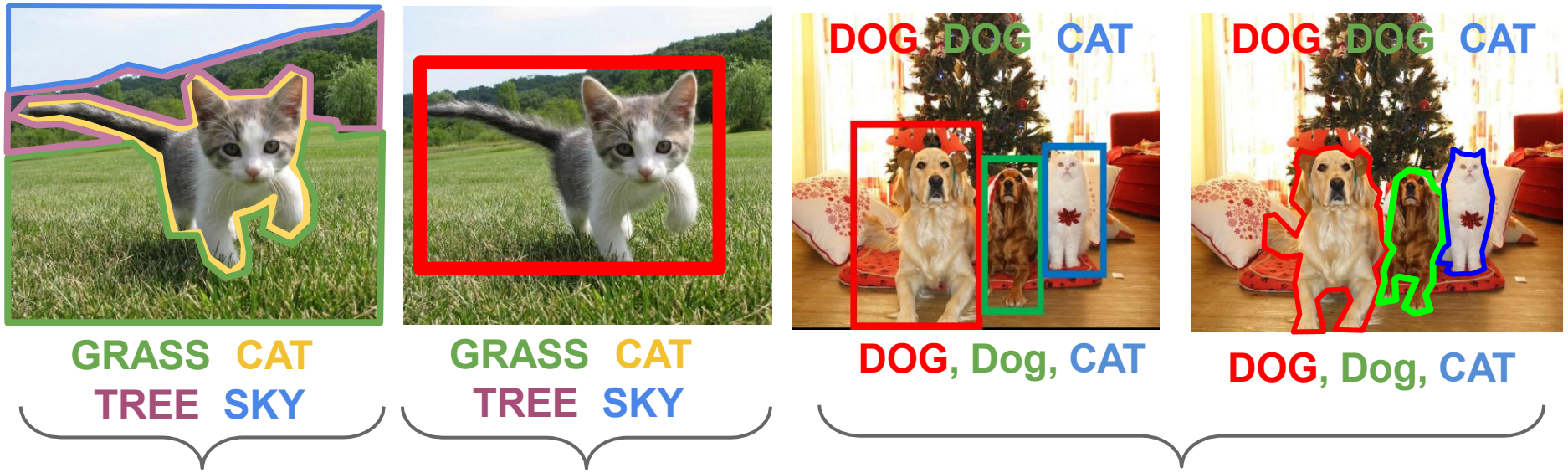
Source: cs231n

OBJECT DETECTION: OVERVIEW

http://deeplearning.csail.mit.edu/

Object Detection

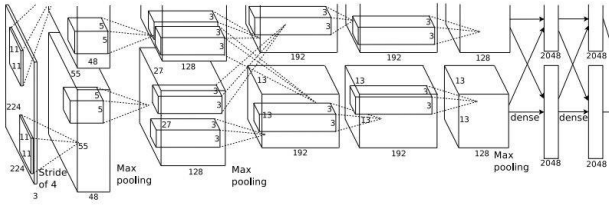
- Object Detection
  - Object location (with bounding box coordinates)
  - Object classification



Source: CS231n

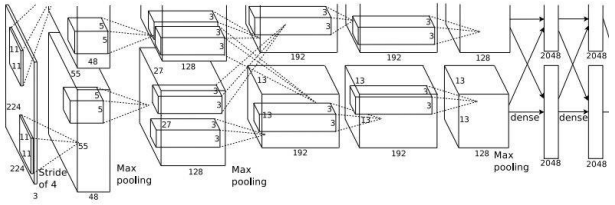
Object Detection as Regression?

Each image needs a different number of outputs!



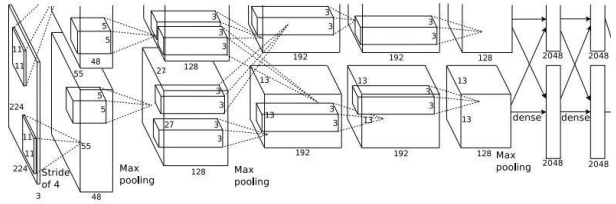
CAT: (x, y, w, h)

4 numbers



DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

16 numbers



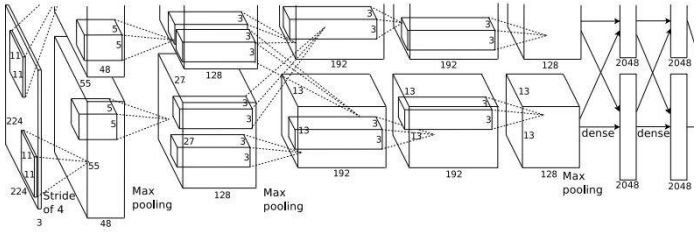
DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Many numbers

Source: cs231n

Object Detection as Classification: Sliding Window

- Apply a CNN to many different crops of the image, CNN classifies each crop as object or background
  - Multi-scale windows



Dog? NO
Cat? NO
Background? YES

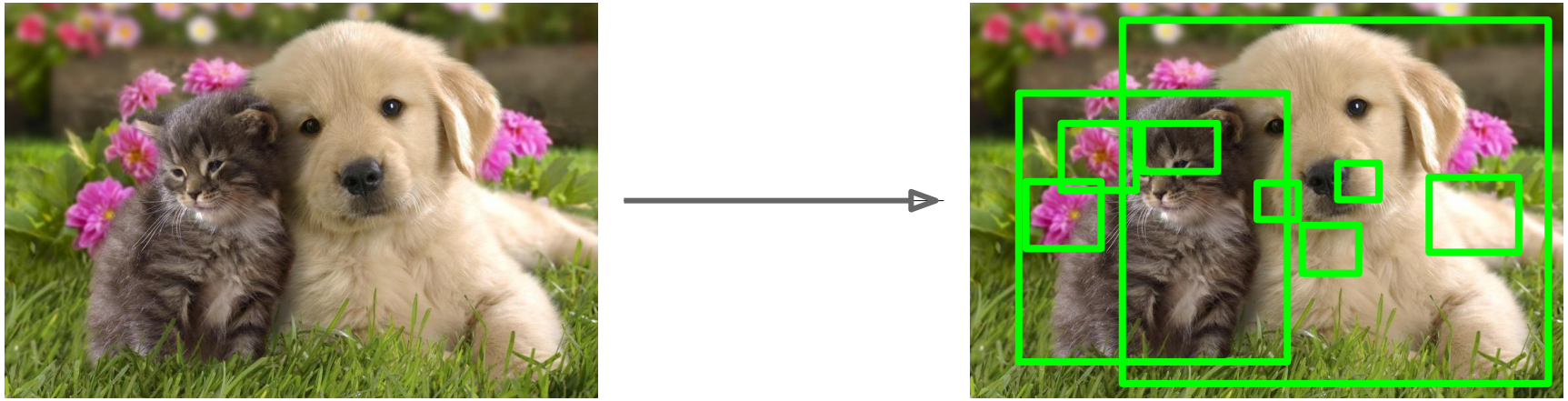
Dog? NO
Cat? Yes
Background? YES

Problem: Need to apply CNN to huge number of locations and scales, very computationally expensive!

Source: cs231n

Region Proposals

- Find “blobby” image regions that are likely to contain objects
  - Relatively fast to run; e.g. *Selective Search* gives 1000 region proposals in a few seconds on CPU

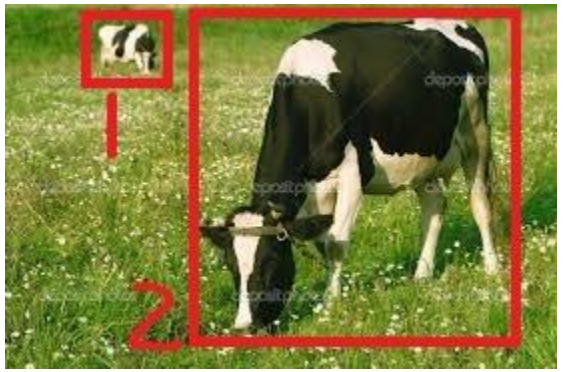


Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012
Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013
Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014
Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

Source: cs231n

Note. Selective Search

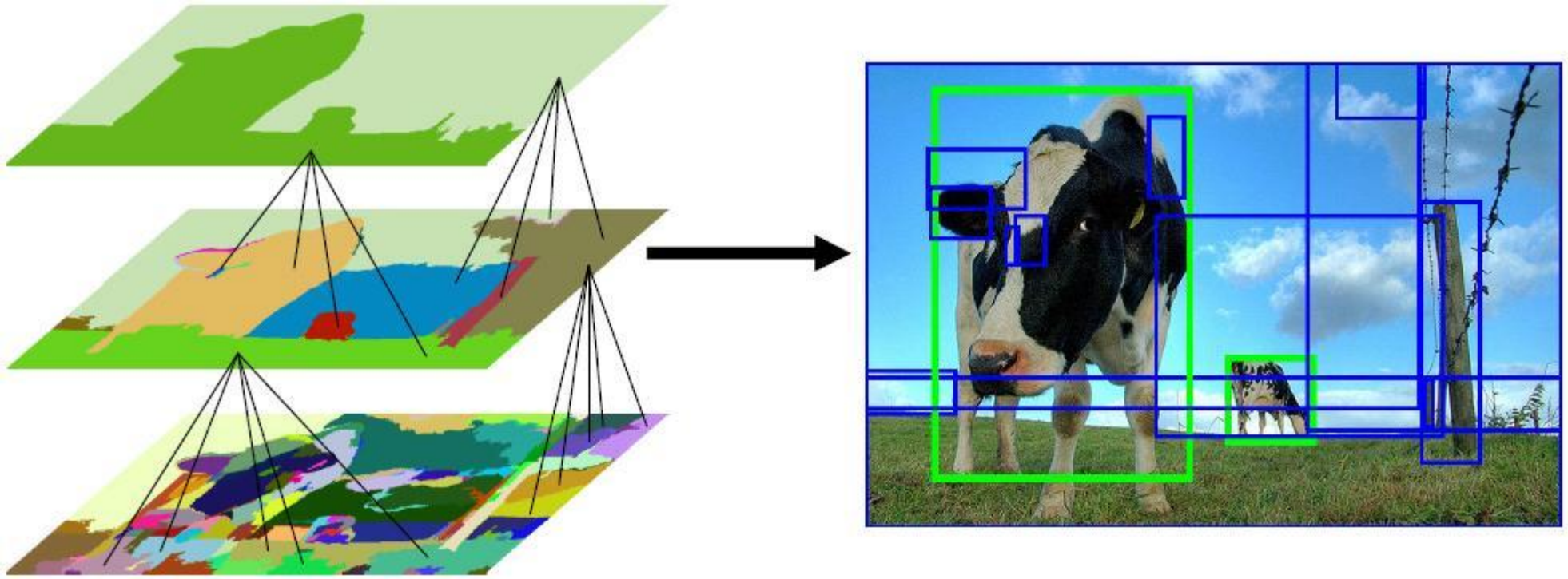
- Challenges of region proposal
  - Multi-scales => segmentation + hierarchical grouping
  - Variety of reasons for a region
    - Texture/color/hierarchical
  - Speed



hierarchical color texture Enclosed object

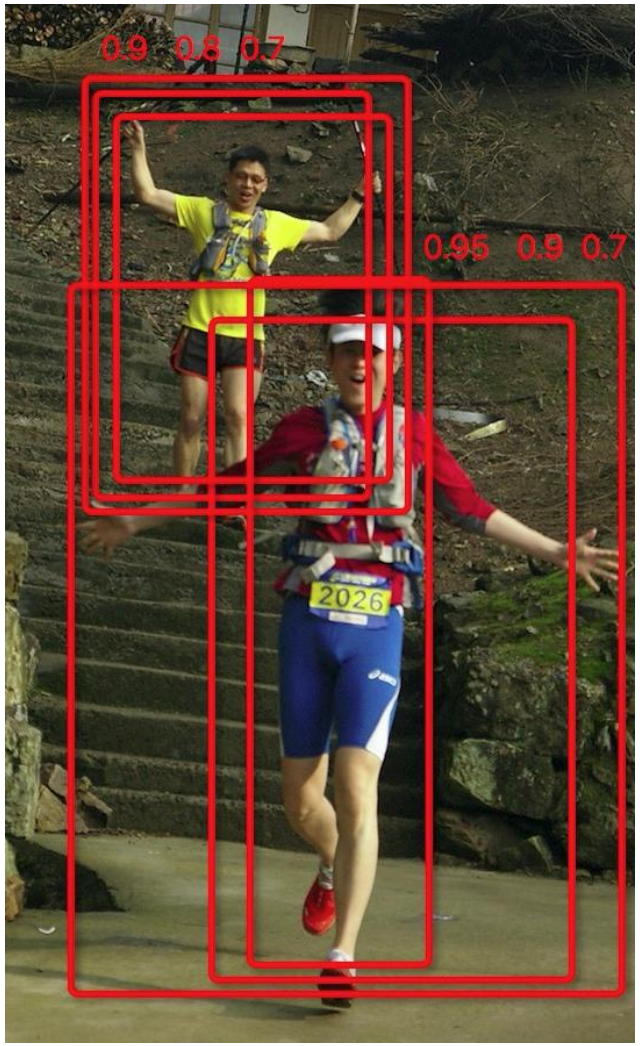
Note. Selective Search

- Hierarchical grouping
- Diversification strategies
  - Color/texture/color space/….



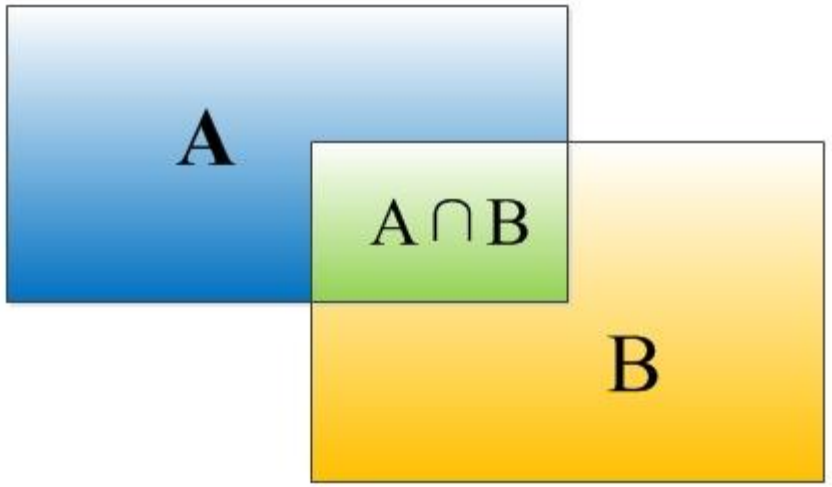
Non-maximum suppression非極大值抑制

- Example: 6 boxes with confidence scores
- How to remove the duplicated one?
  - 按信心機率排序: 0.95, 0.9, 0.9, 0.8, 0.7, 0.7
  - 取機率最大0.95的框為一個物體框
  - 剩餘5個框中，去掉與0.95框重疊率大於0.6(可以另行設置)，則保留0.9, 0.8, 0.7三個框
  - 重複上面的步驟，直到沒有框了，0.9為一個框
  - 選出來的為: 0.95, 0.9

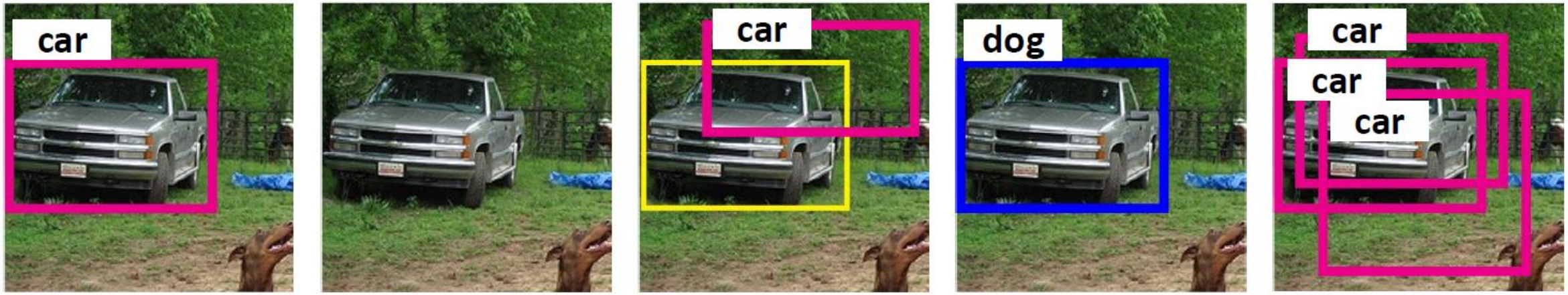


Note. Terms

- IOU: intersection of unit
  - Define accuracy of localization
  - IOU=(A∩B)/(A∪B)

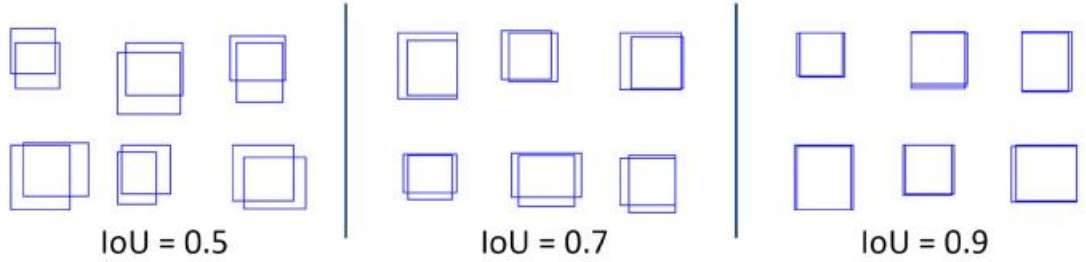


Object Detection: Evaluation



correct (true positive)	missed (false negative)	mislocalized (false positive)	misrecognized (false positive)	duplicated (false positive)

define by an IoU threshold



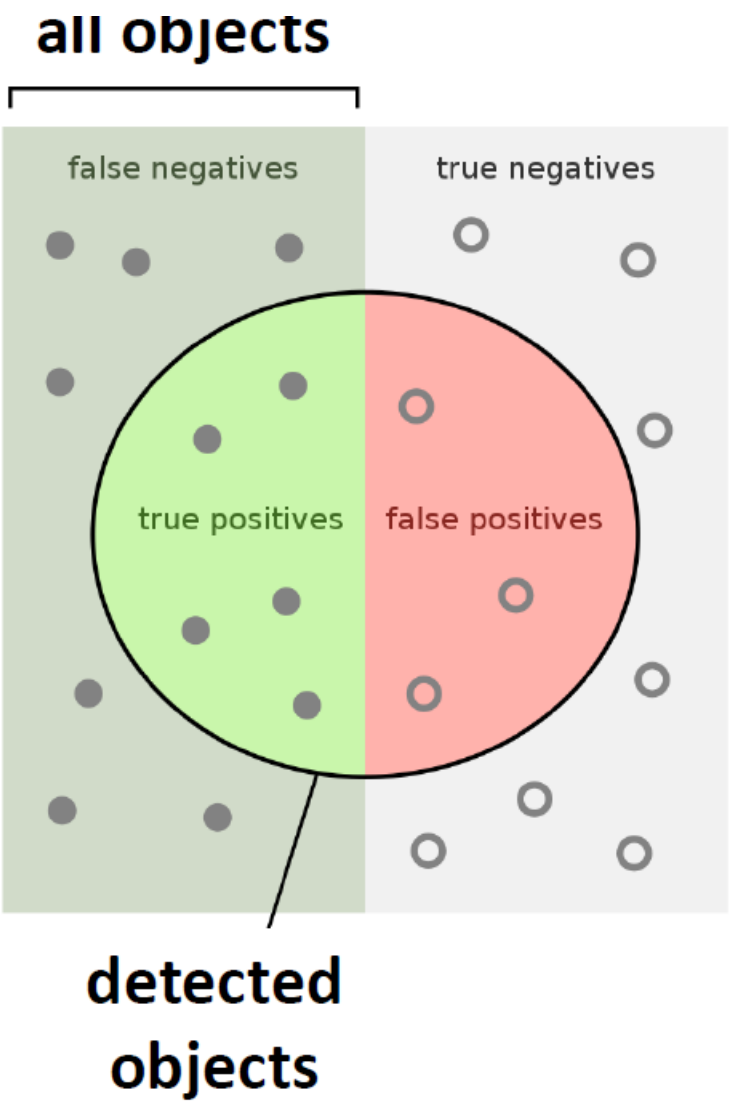
Object Detection: Evaluation

- **Precision**: What % of detected objects are correct?

$$\textbf{Precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}}$$

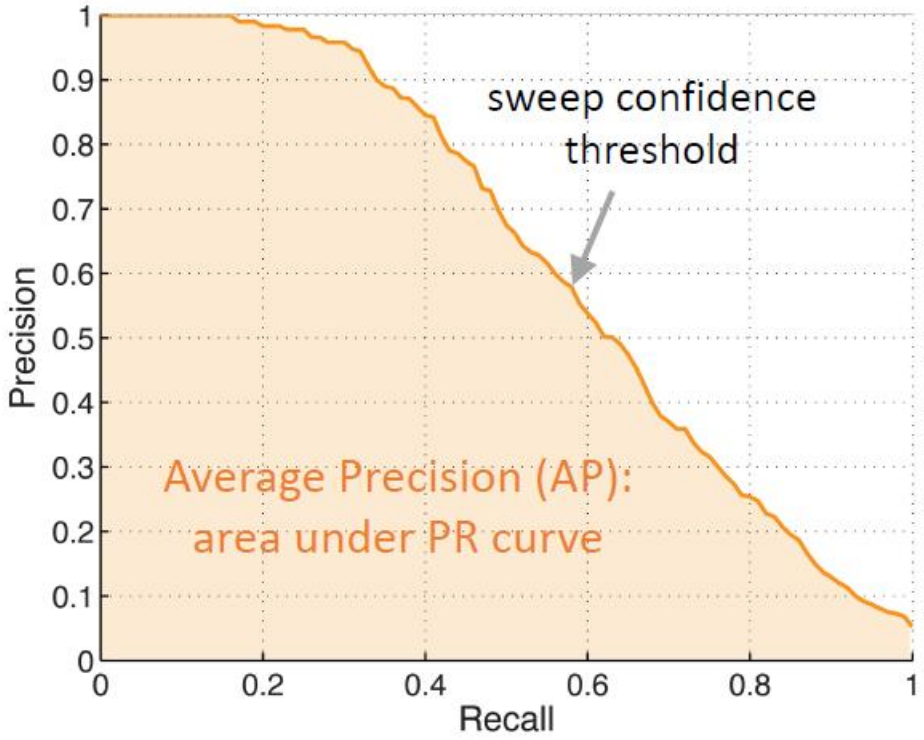
- **Recall**: What % of objects are detected?

$$\textbf{Recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}}$$



Object Detection: Evaluation

- Precision-Recall Tradeoff



Life is a precision-recall tradeoff.

threshold = **0.4**: Recall ↑, Precision ↓

threshold = **0.7**: Recall ↓, Precision ↑

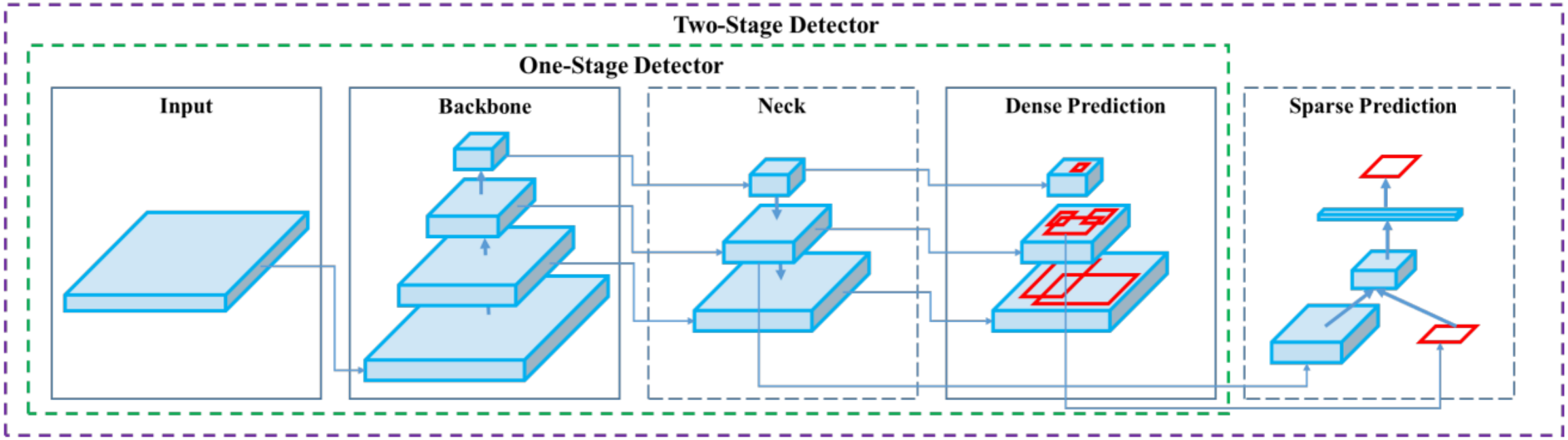
Note. Bounding Box Regression

- Regression used to find bounding box parameters
- Applied in one of two ways
  - Bounding box refinement
  - Complete object detection

Example Loss Function

$$\sum_{i \ in \ I} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2]$$

I is the set of all matching bounding boxes (Highest IoU with ground truth)



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

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

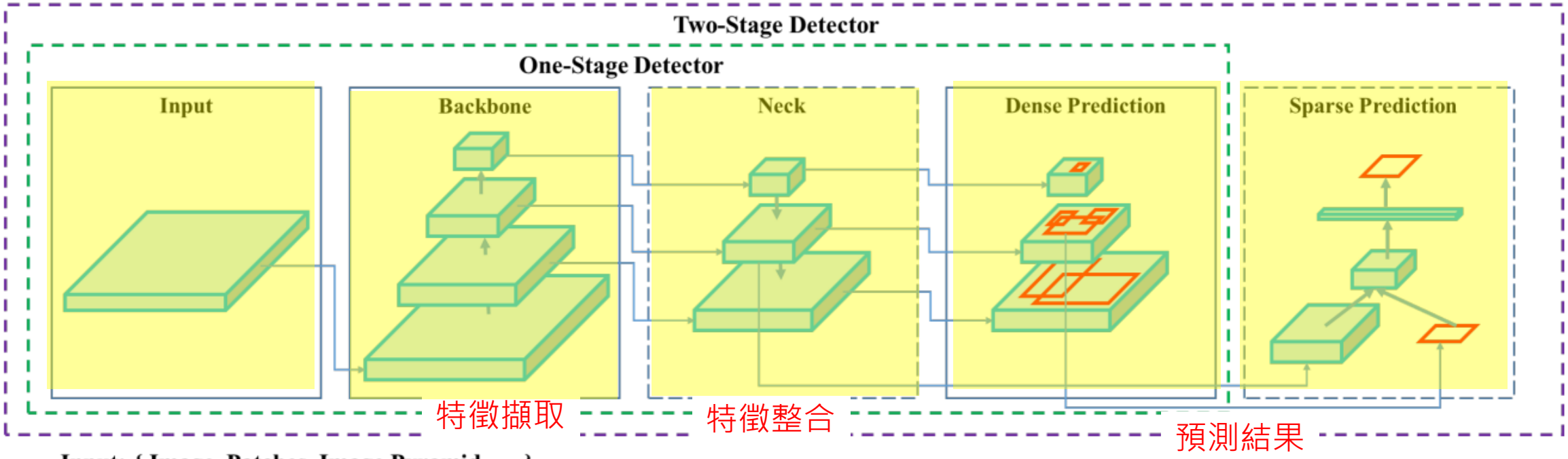
Neck: { FPN [44], PANet [49], Bi-FPN [77], … }

Head:

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

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

YOLOv4: Optimal Speed and Accuracy of Object Detection https://www.youtube.com/watch?v=HdQqAF-rMKc&t=5113s



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

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

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

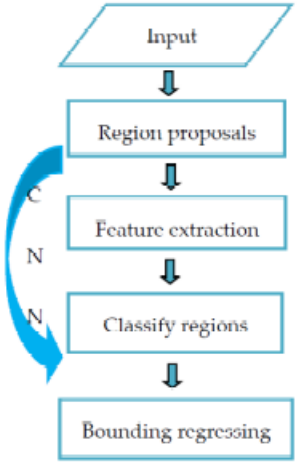
Head:

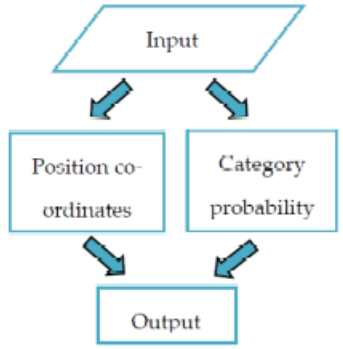
Dense Prediction: { RPN [64], YOLO [61, 62, 63], SSD [50], RetinaNet [45], FCOS [78], … }

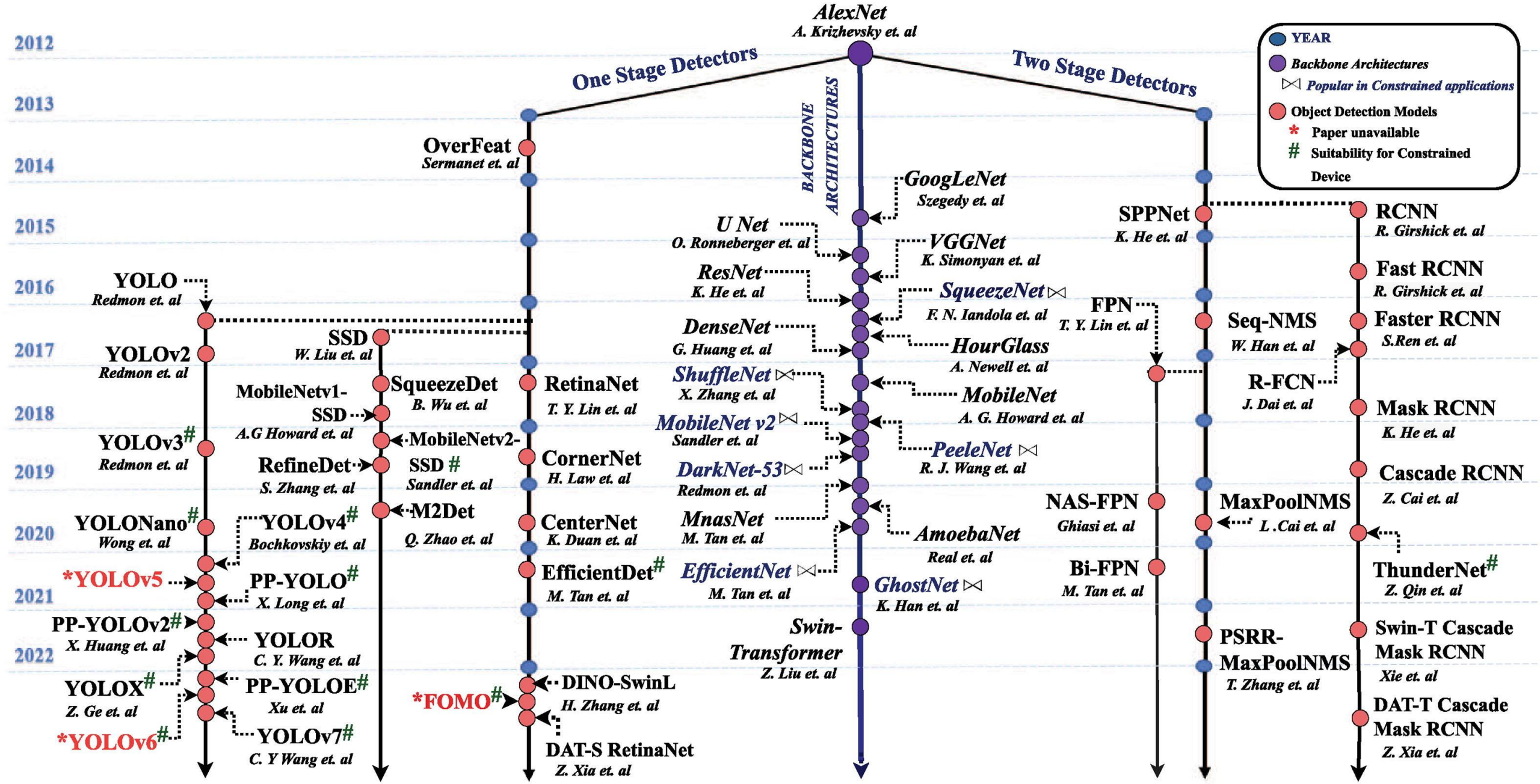
Sparse Prediction: { Faster R-CNN [64], R-FCN [9], … }

Predict class and bounding box

YOLOv4: Optimal Speed and Accuracy of Object Detection https://www.youtube.com/watch?v=HdQqAF-rMKc&t=5113s

Method Category	Typical Work	Characteristics	Advantages and Disadvantages	Basic Framework
Two-stage	R-CNN [20] SPPNet [21] Fast R-CNN [22] Faster R-CNN [23] Cascade R-CNN [24] FPN [25]	(1) Propose a large number of regions by selective searching; (2) Detect objects in the region proposals; (3) Change the way of extracting features from samples and solve the problem that a CNN model is hard to be trained with a small number of samples.	Advantages: High detection accuracy Disadvantages: Multistage pipeline training Slow object detection	

Method Category	Typical Work	Characteristics	Advantages and Disadvantages	Basic Framework
One-stage	YOLO-V1 [26] SSD [27] RetinaNet [28] YOLO-V2 [29] YOLO-V3 [30] FCOS [31] GC-YOLOv3 [32] AlignDet [33] CenterNet [34]	(1) No region proposal stage and processing of the entire picture during training and prediction; (2) End-to-end detection with a single CNN model so it can directly output object positions and categories from the input image.	Advantages: Fast object detection Disadvantages: Relatively low detection accuracy	



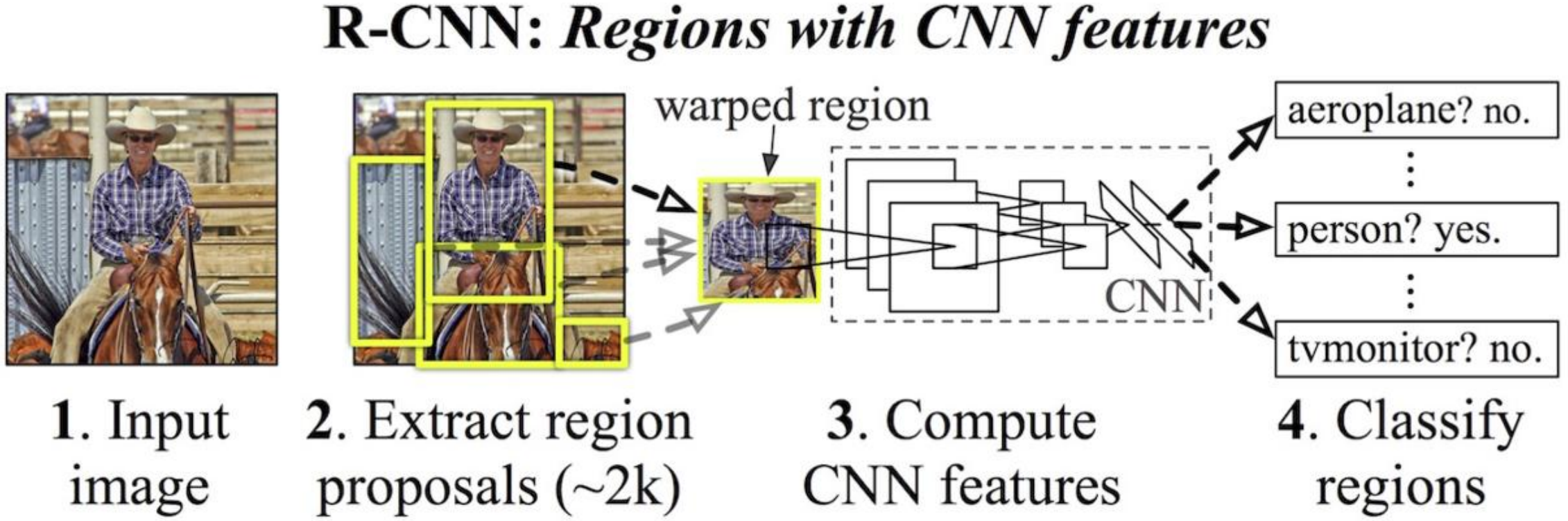
Deep learning based object detection for resource constrained devices: Systematic review, future trends and challenges ahead

OBJECT DETECTION: TWO-STAGE APPROACH

http://deeplearning.csail.mit.edu/

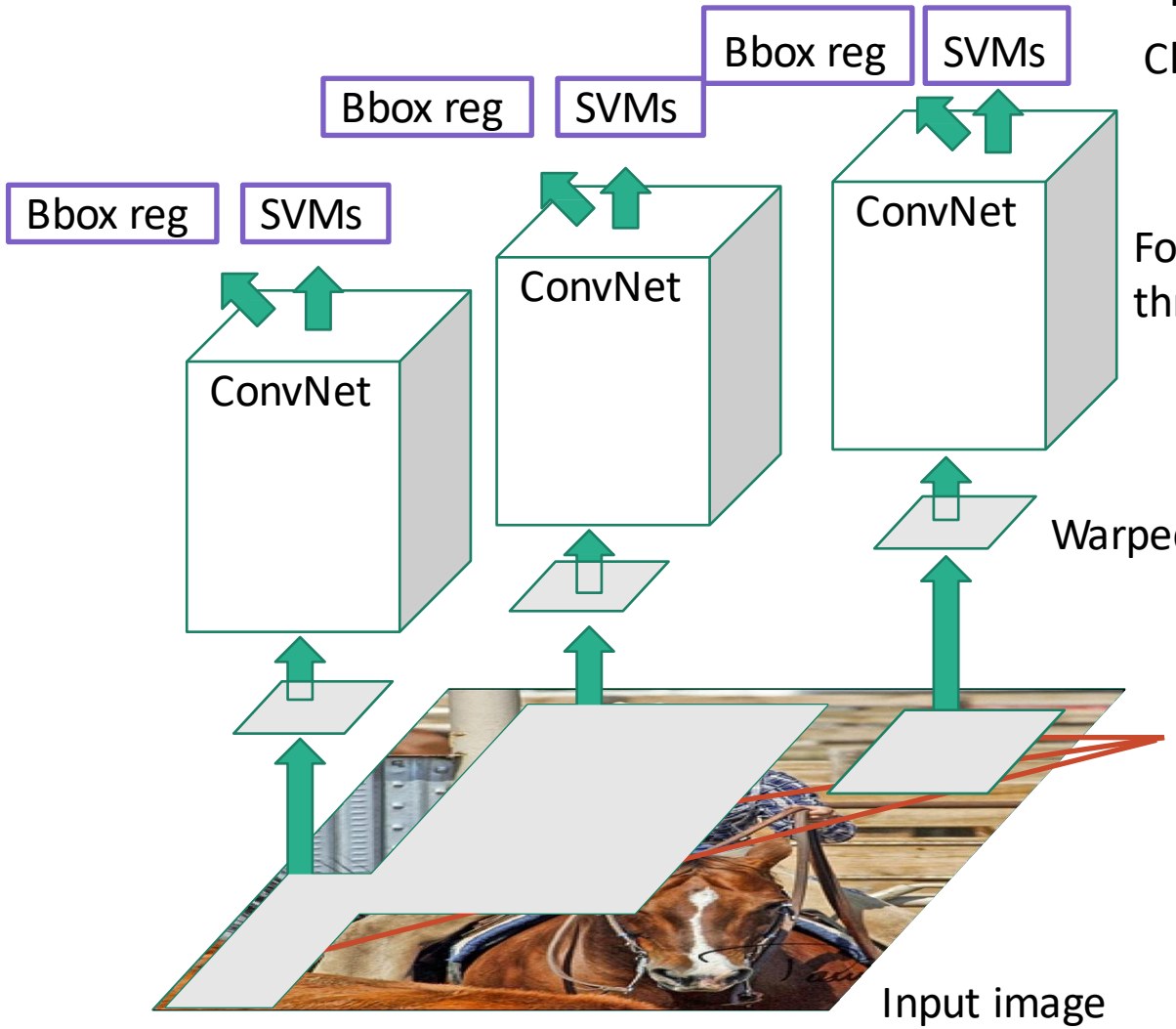
R-CNN

- Warped region: scale all proposal size to 227x227 for CNN
- CNN: AlexNet with finetuning
- Bounding box regression + SVM



Source: 2013, Rich feature hierarchies for accurate object detection and semantic segmentation

R-CNN



Apply bounding box regressors

Classify regions with SVMs

Forward each region through ConvNet

Warped image regions

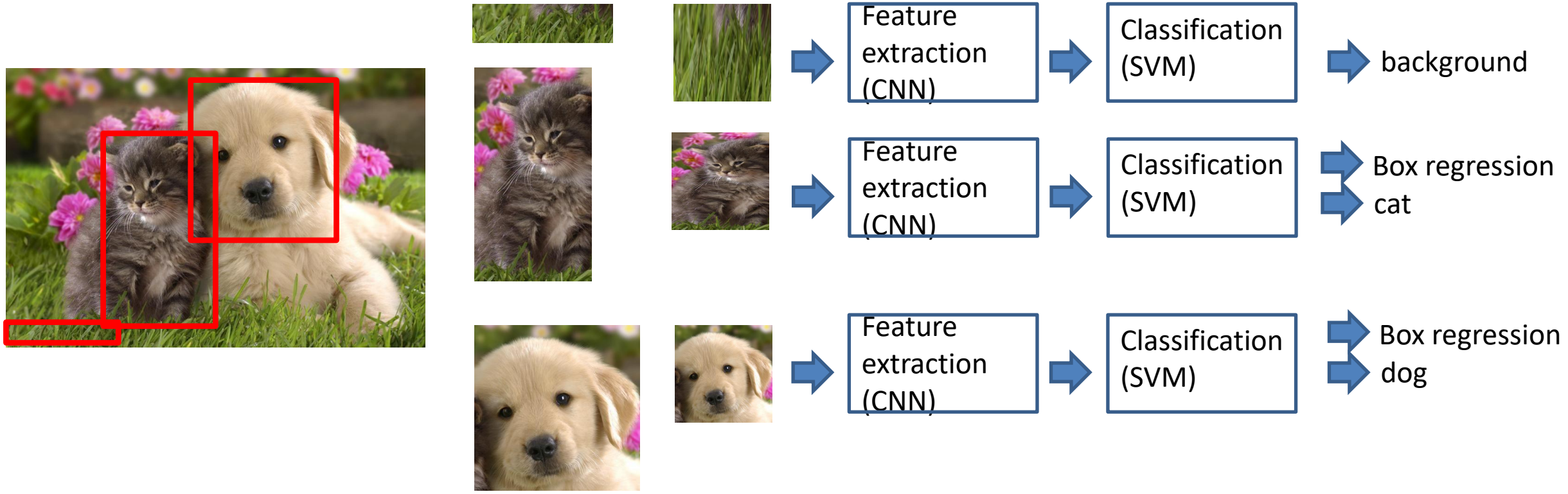
Regions of Interest (RoI) from a proposal method (~2k)

Input image

Girshick et al. CVPR14.

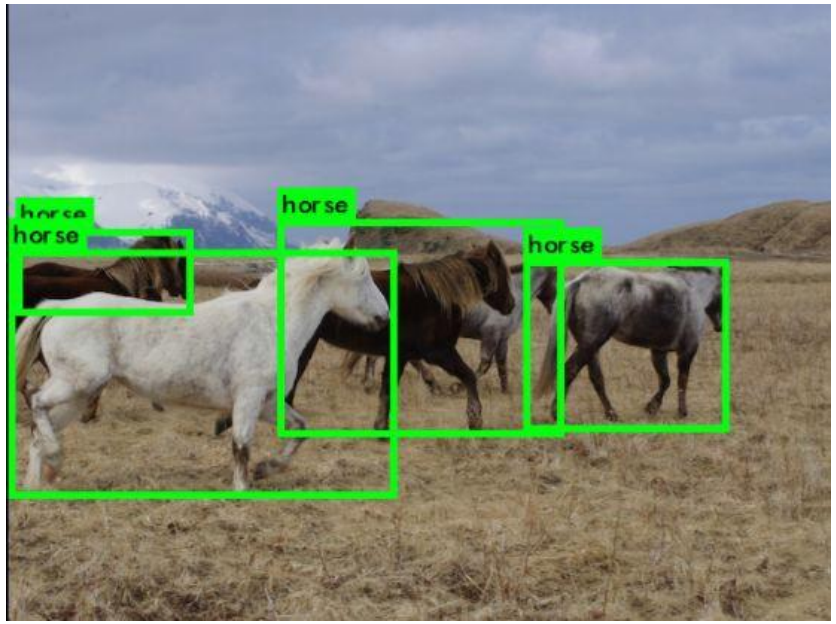
Post hoc component

R-CNN


Feature extraction (CNN)
Classification (SVM)
background
Feature extraction (CNN)
Classification (SVM)
Box regression
cat
Feature extraction (CNN)
Classification (SVM)
Box regression
dog
Input image
Region proposal
Warp
Detection and bounding box regression

R-CNN

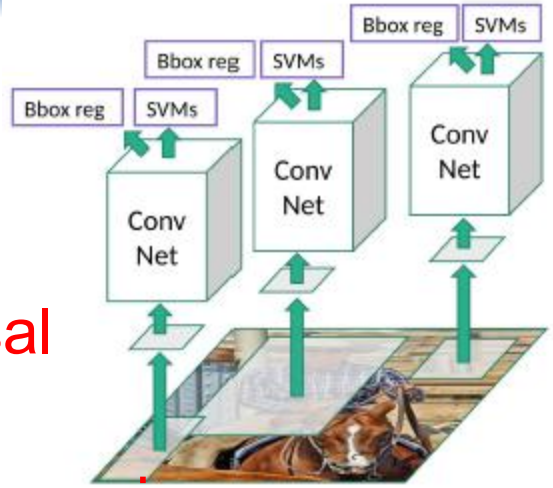
We expect



We get

Non Maximum Suppression + score threshold

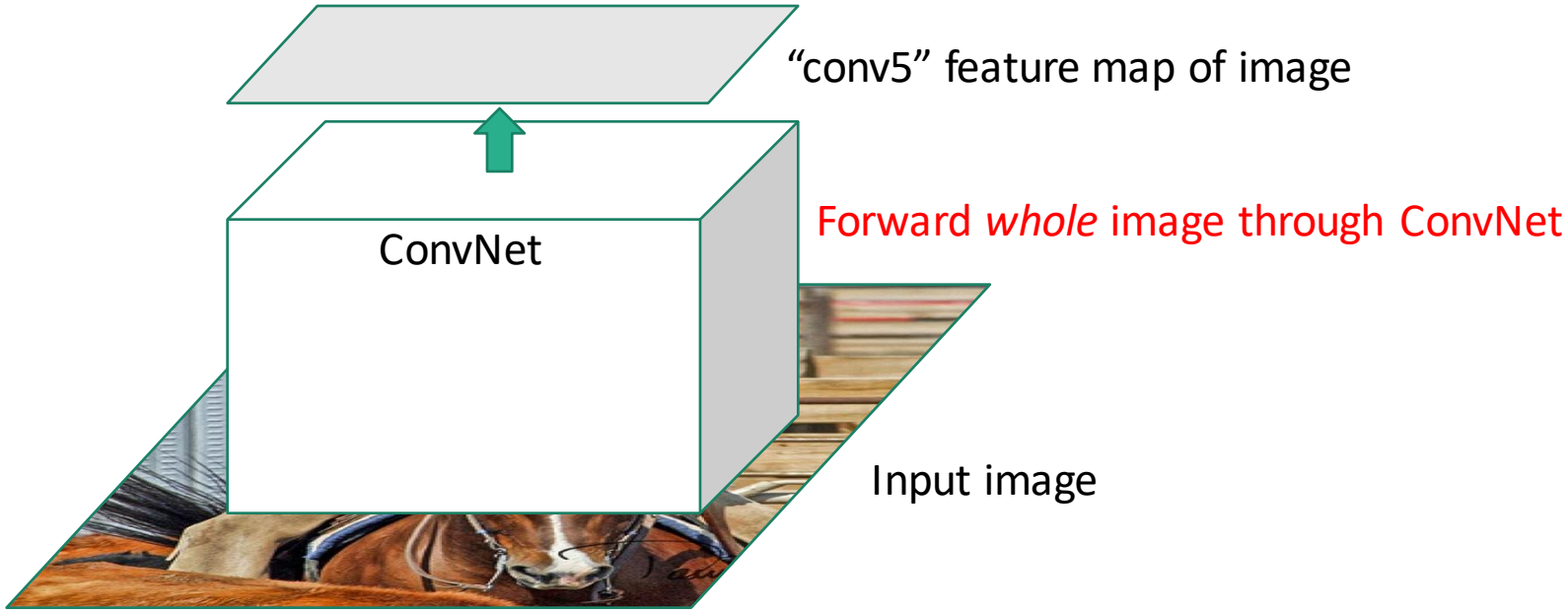
R-CNN: Problems



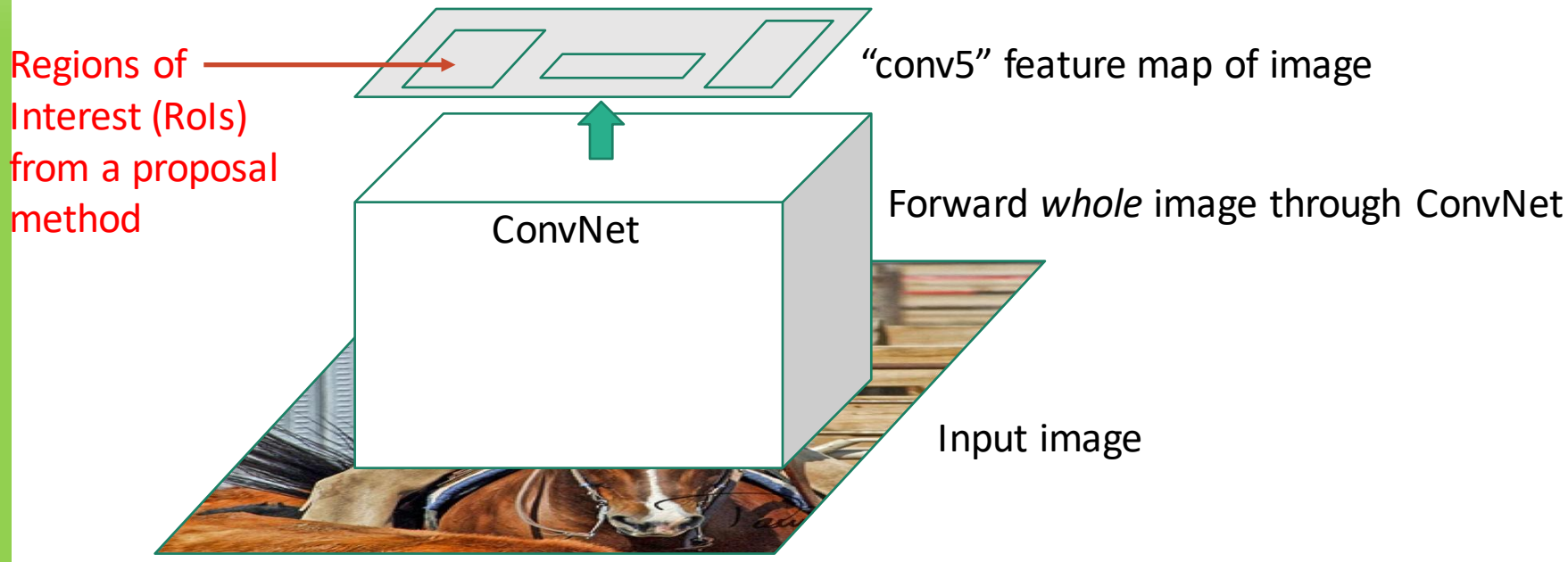
- Slow at test time
  - need to run full forward pass of CNN for each region proposal
  - ~20K proposal => 2000 ConvNet forward passes per image
  - Solution: Share computation of convolutional layers between region proposals for an image
- Ad hoc training objectives
  - Fine-tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressions (least squares)
  - CNN features not updated in response to SVMs and regressors
  - Can we co-optimize them all?
- Training is slow (84h), takes a lot of disk space
  - Complex multistage training pipeline
- Solution: Train it all at together end-to-end

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

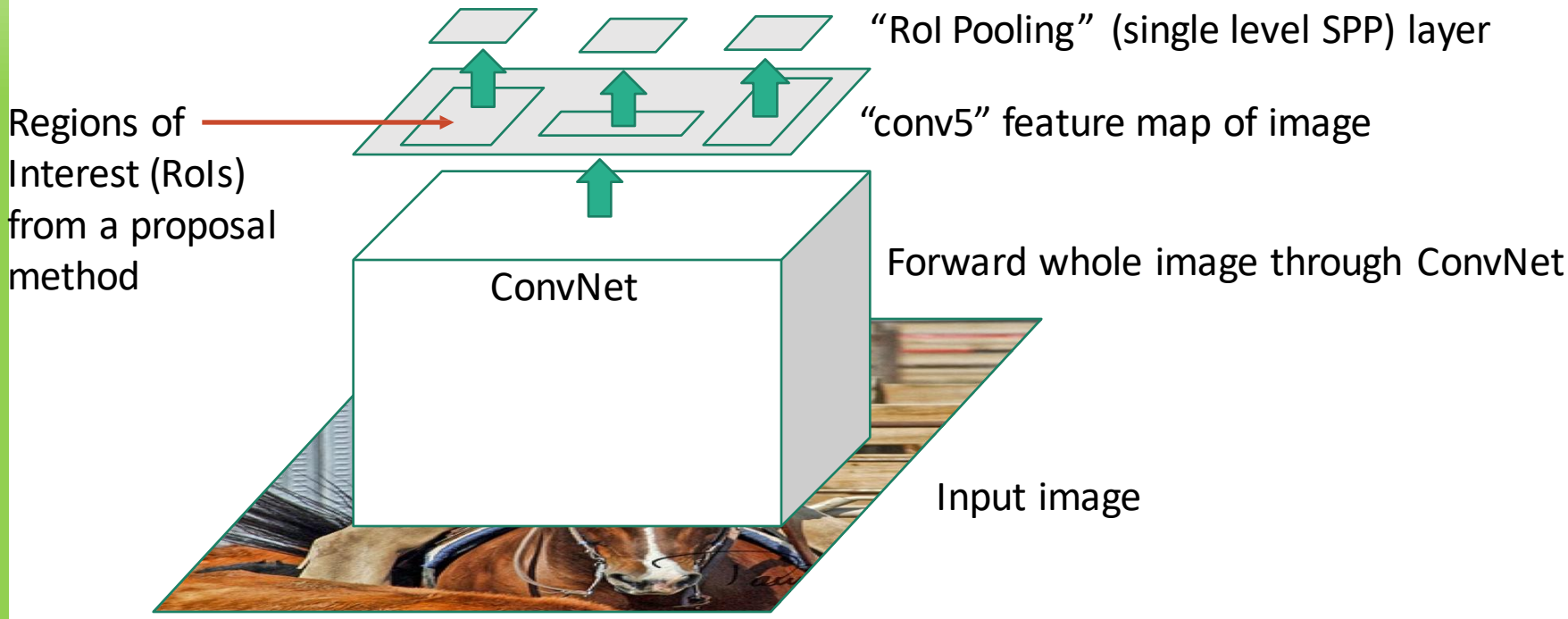
Fast R-CNN


“conv5” feature map of image
ConvNet
Forward *whole* image through ConvNet
Input image

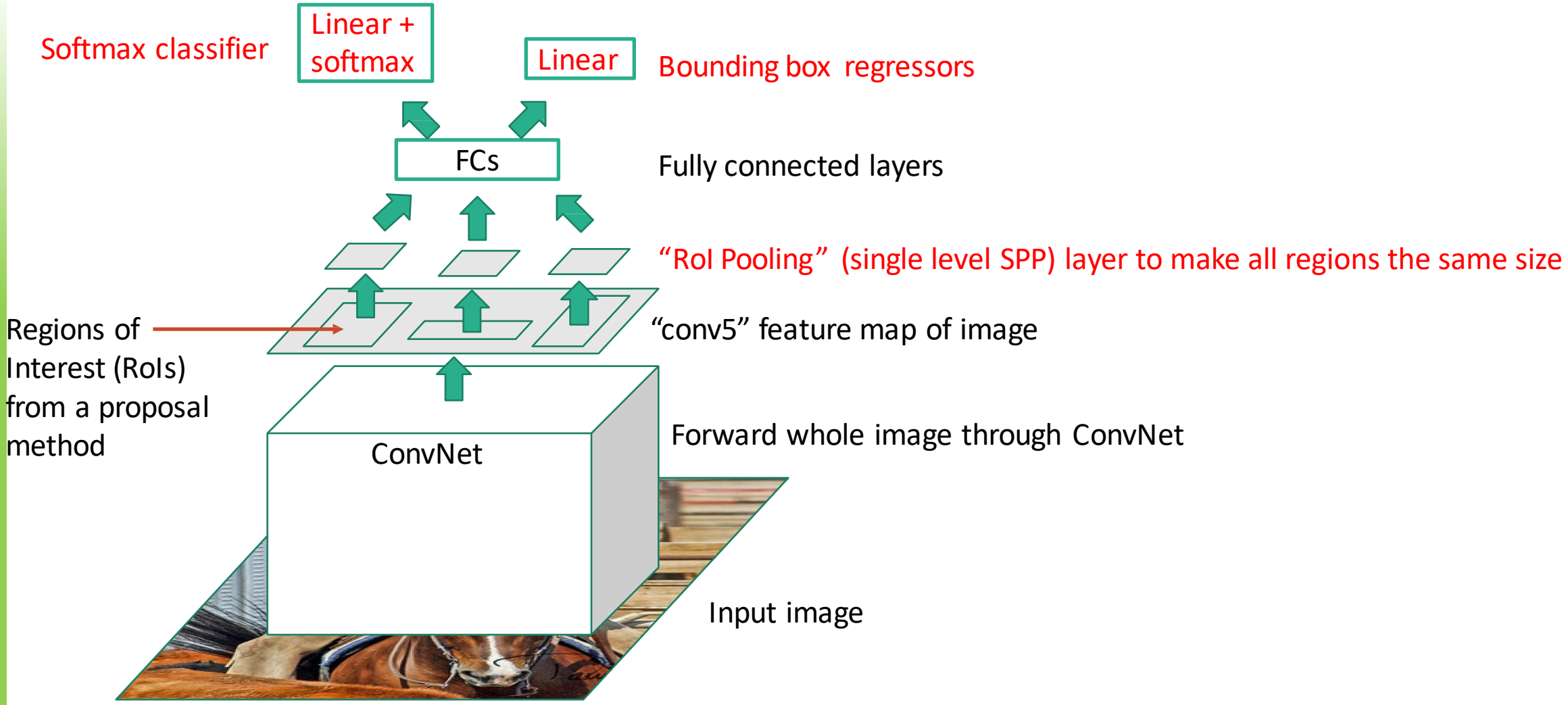
Fast R'CNN (test time)



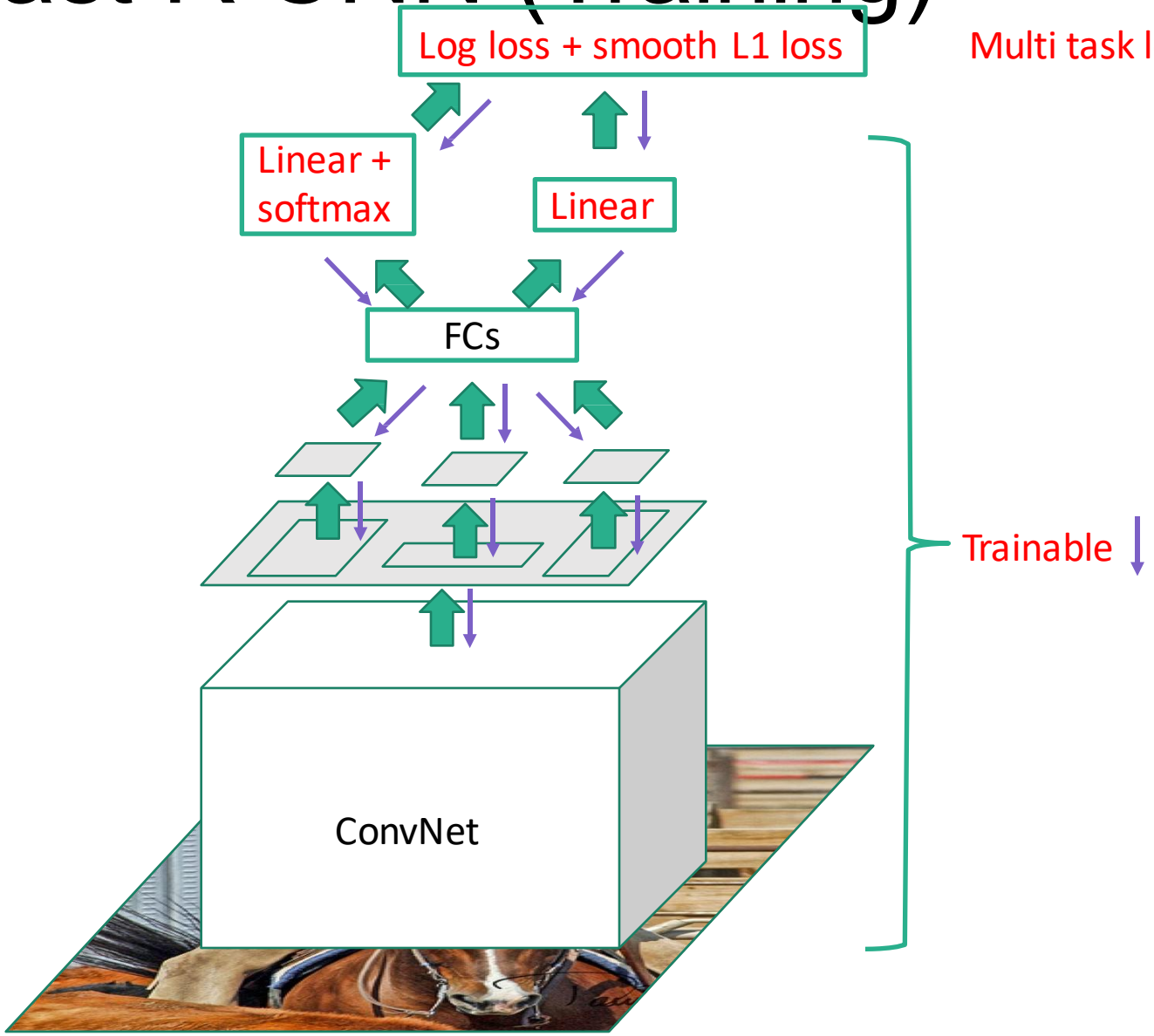
Fast R'CNN (test time)



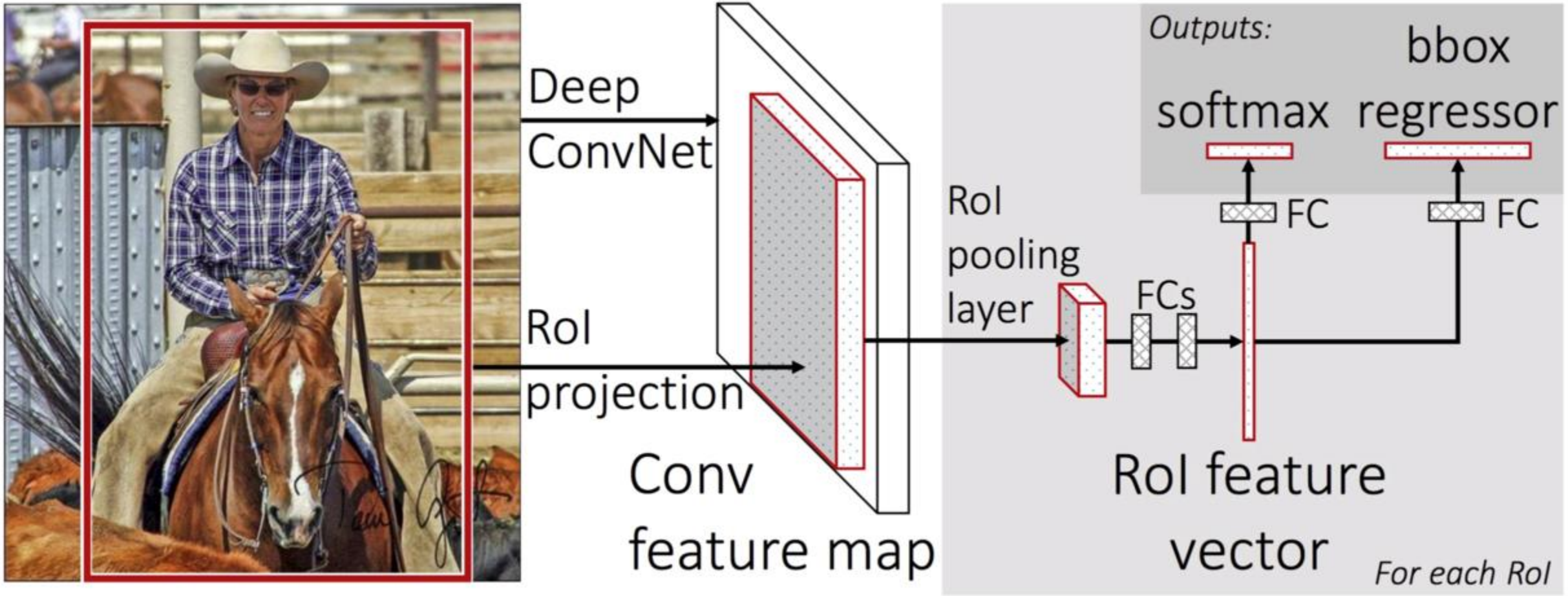
Fast R'CNN (test time)



Fast R'CNN (Training)



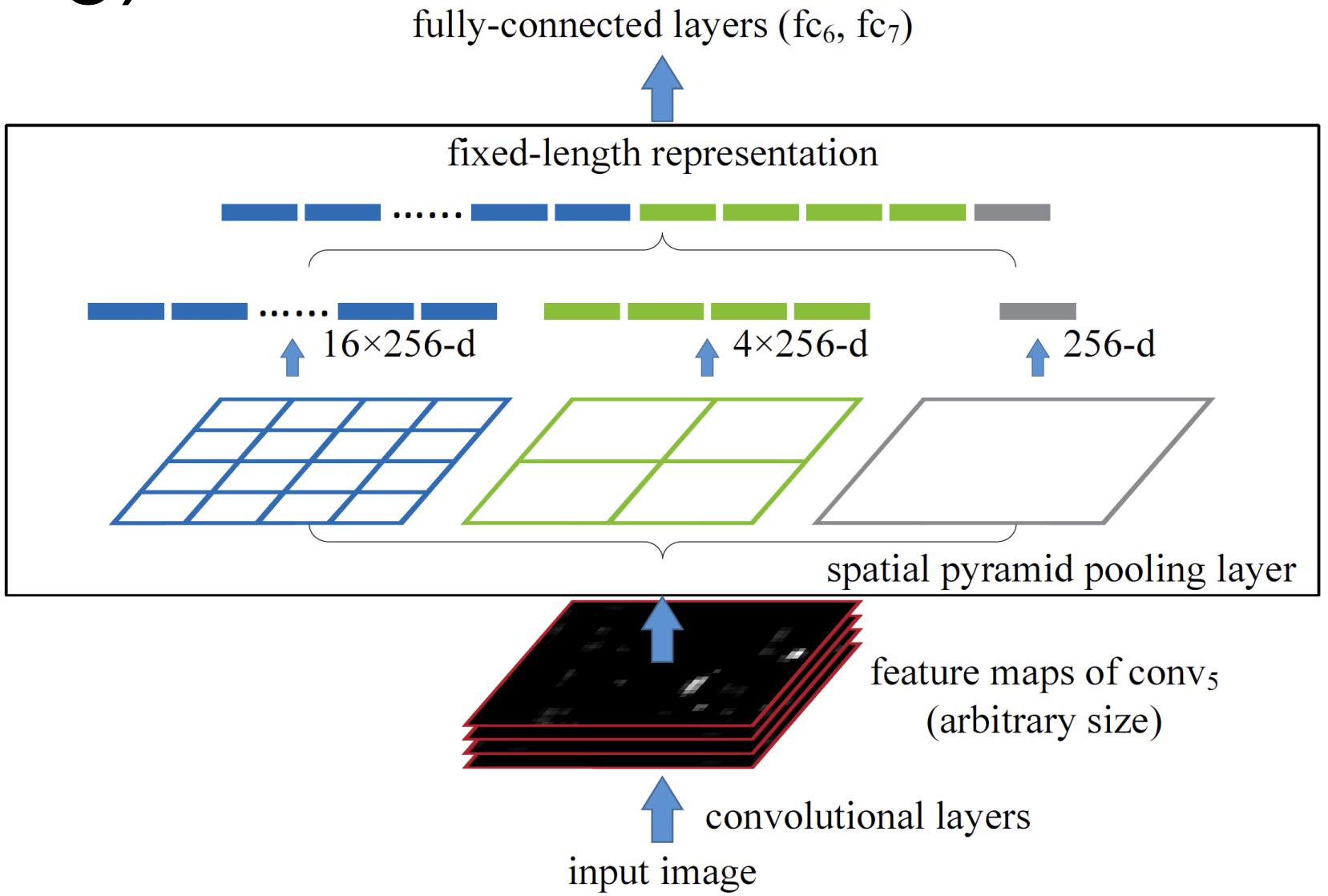
Fast R-CNN



Problem: Out'of'network region proposals

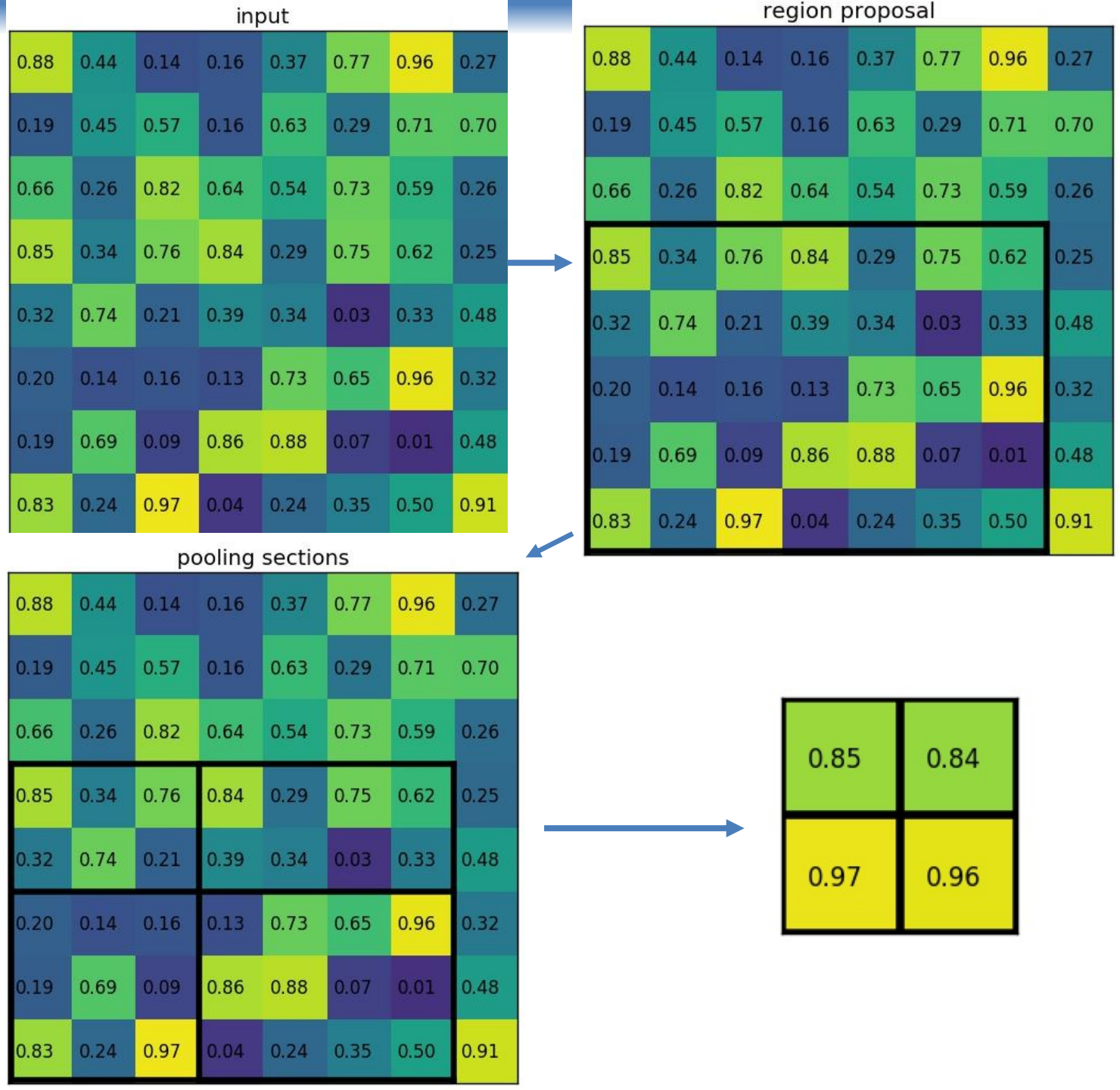
ROI Pooling: Special Case of SPP (Spatial Pyramid Pooling)

- Map different size of feature map to fixed size
  - To fit fixed size needs of FC layers

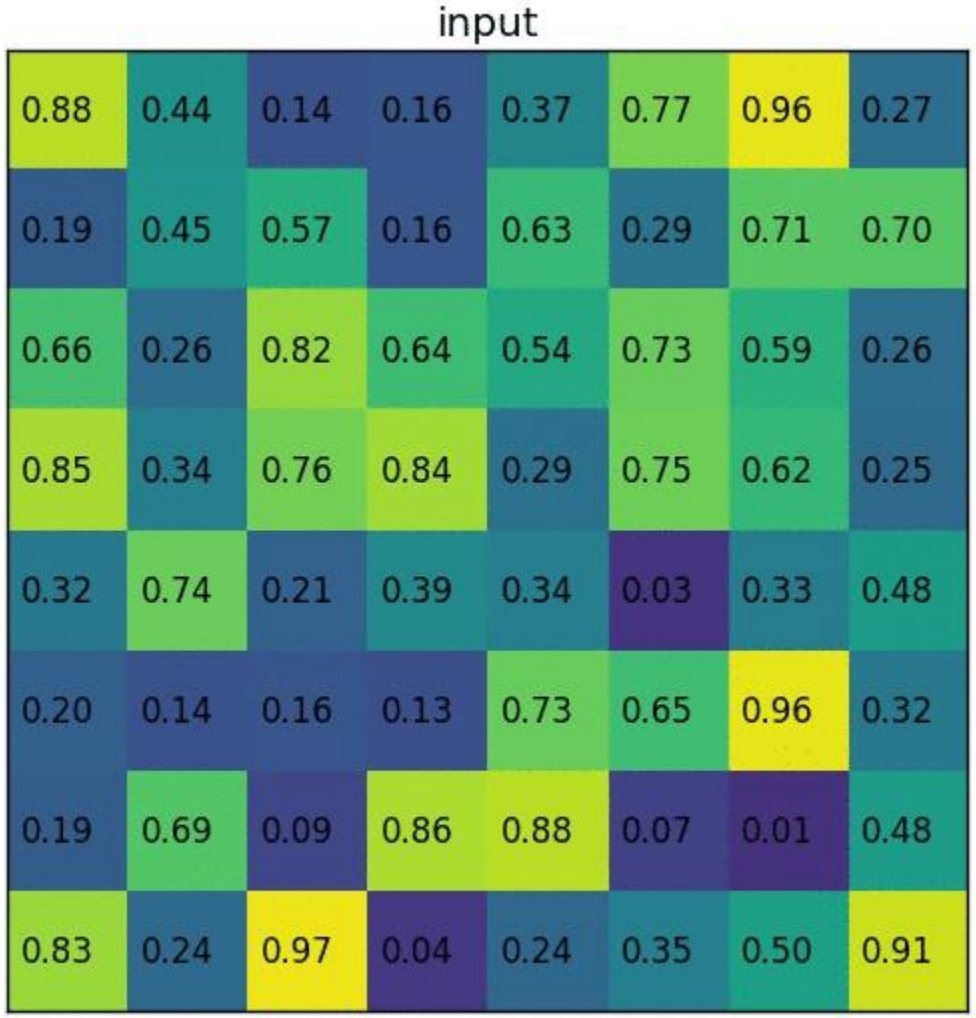


ROI Pooling

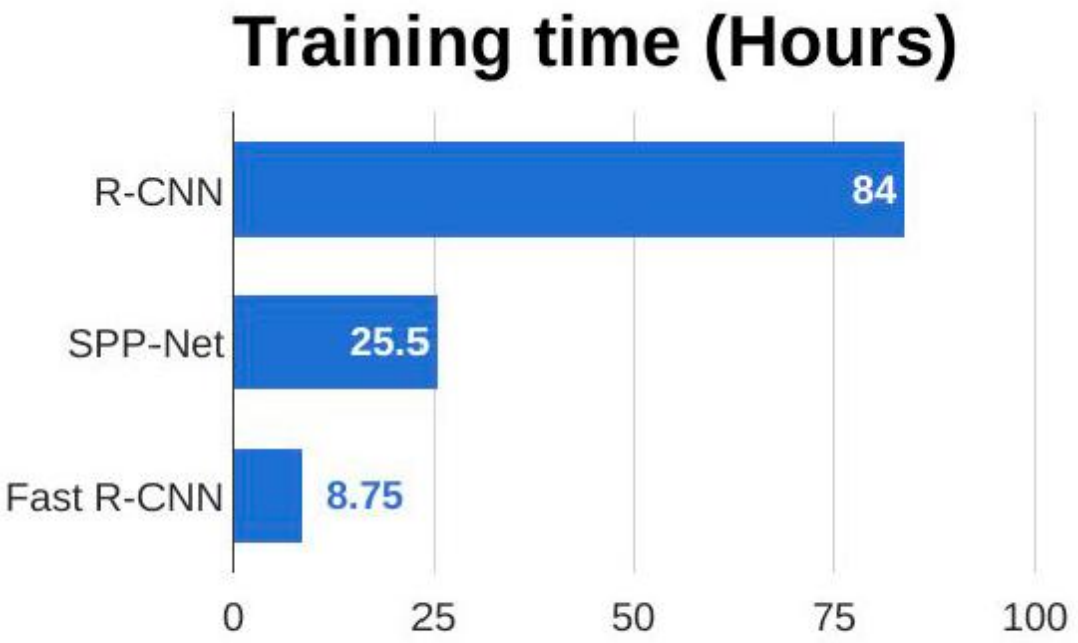
- An feature map
  - 8x8
- Region proposal
  - (0, 3), (7, 8)
- Divided into desired output size
  - (2x2)
- Max pooling

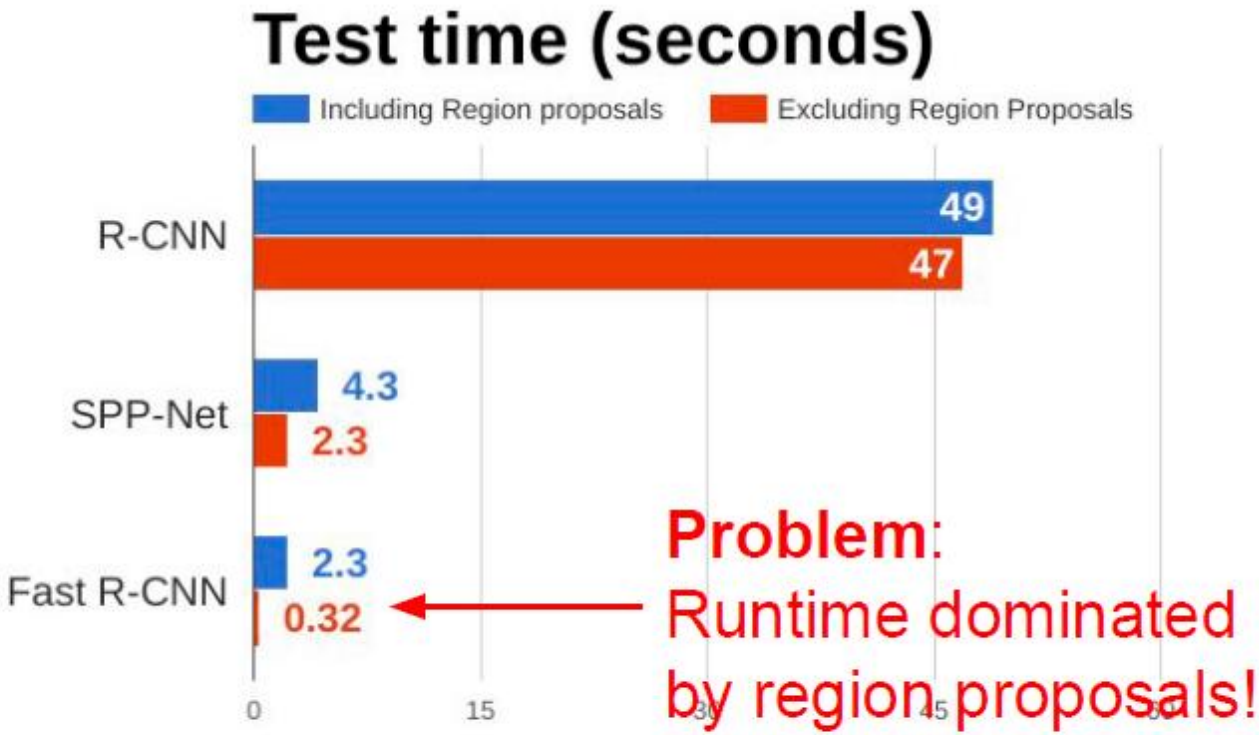


ROI Pooling Example



R-CNN v.s. SPP v.s. Fast R-CNN



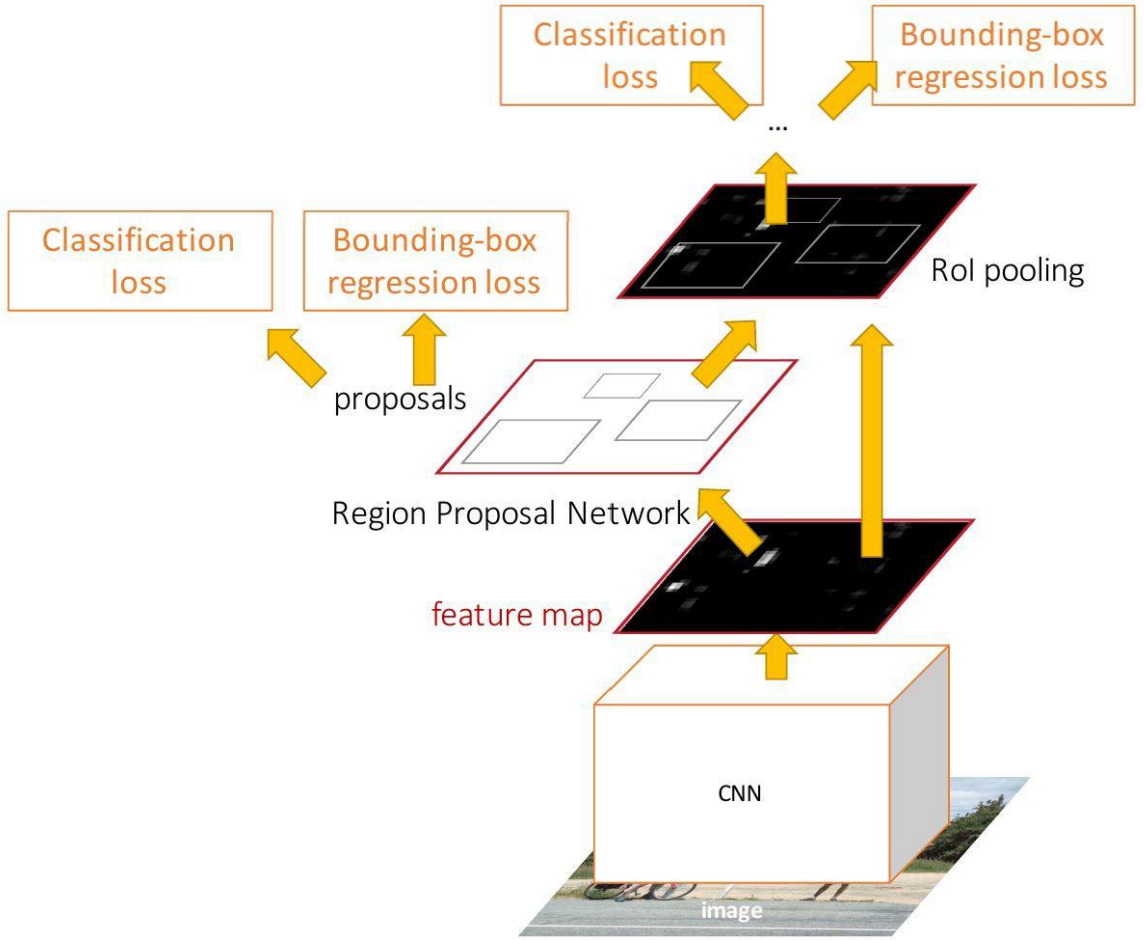


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

Source: CS231n

Faster R-CNN: Make CNN do proposals!

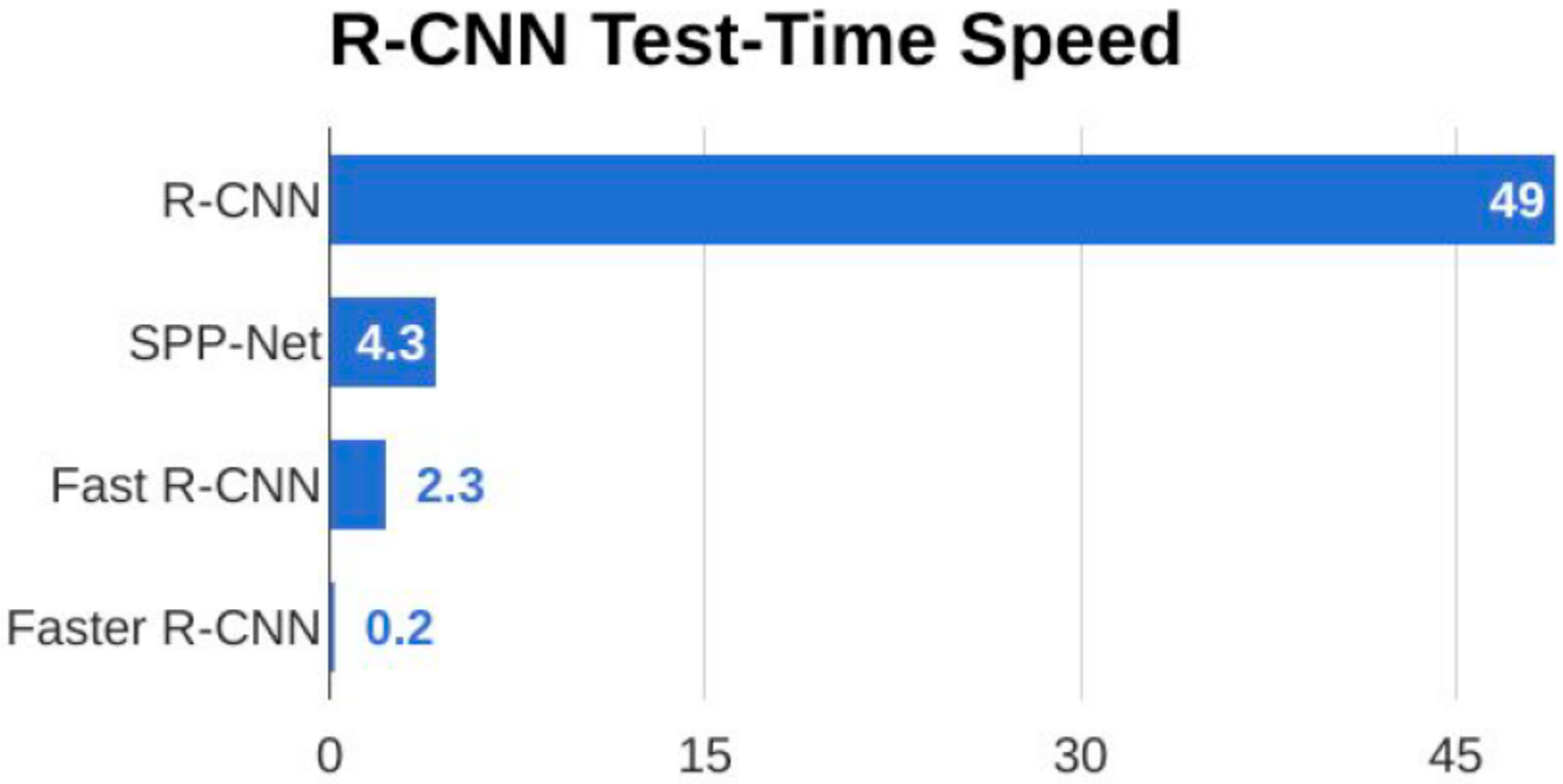
- Add a region proposal network to predict proposal from features
  - Share common feature maps with classification network
  - RPN serves as the 'attention' of this unified network.
- Jointly train with 4 losses:
  - 1. RPN classify object / not object
  - 2. RPN regress box coordinates
  - 3. Final classification score (object classes)
  - 4. Final box coordinates



Shaoqing Ren, Kaiming He, Ross Girshick & Jian Sun. "Faster R'CNN: Towards Real'Time Object Detection with Region Proposal Networks." NIPS 2015.

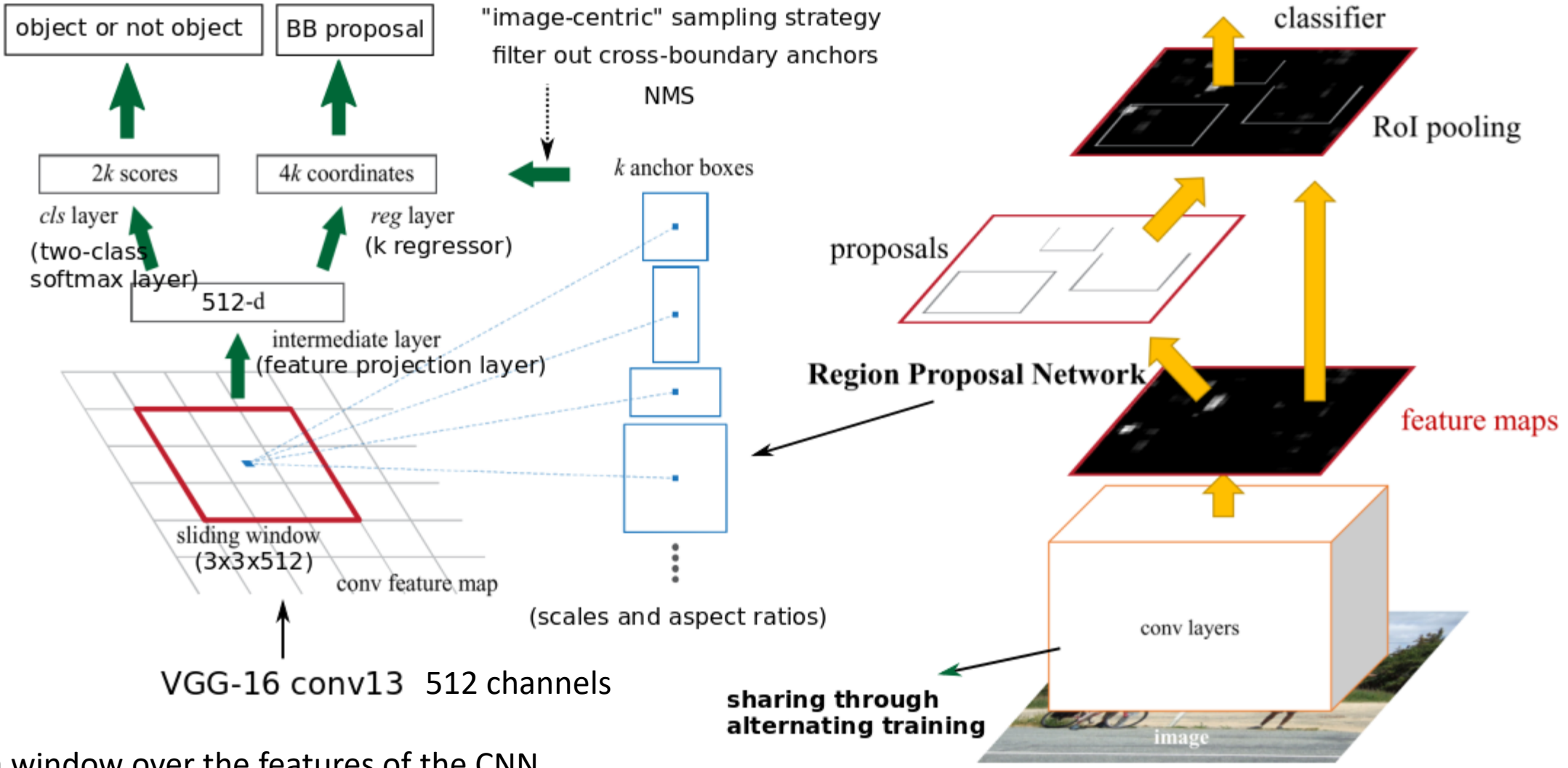
Faster R-CNN

- Make CNN do proposals



Source: CS231n

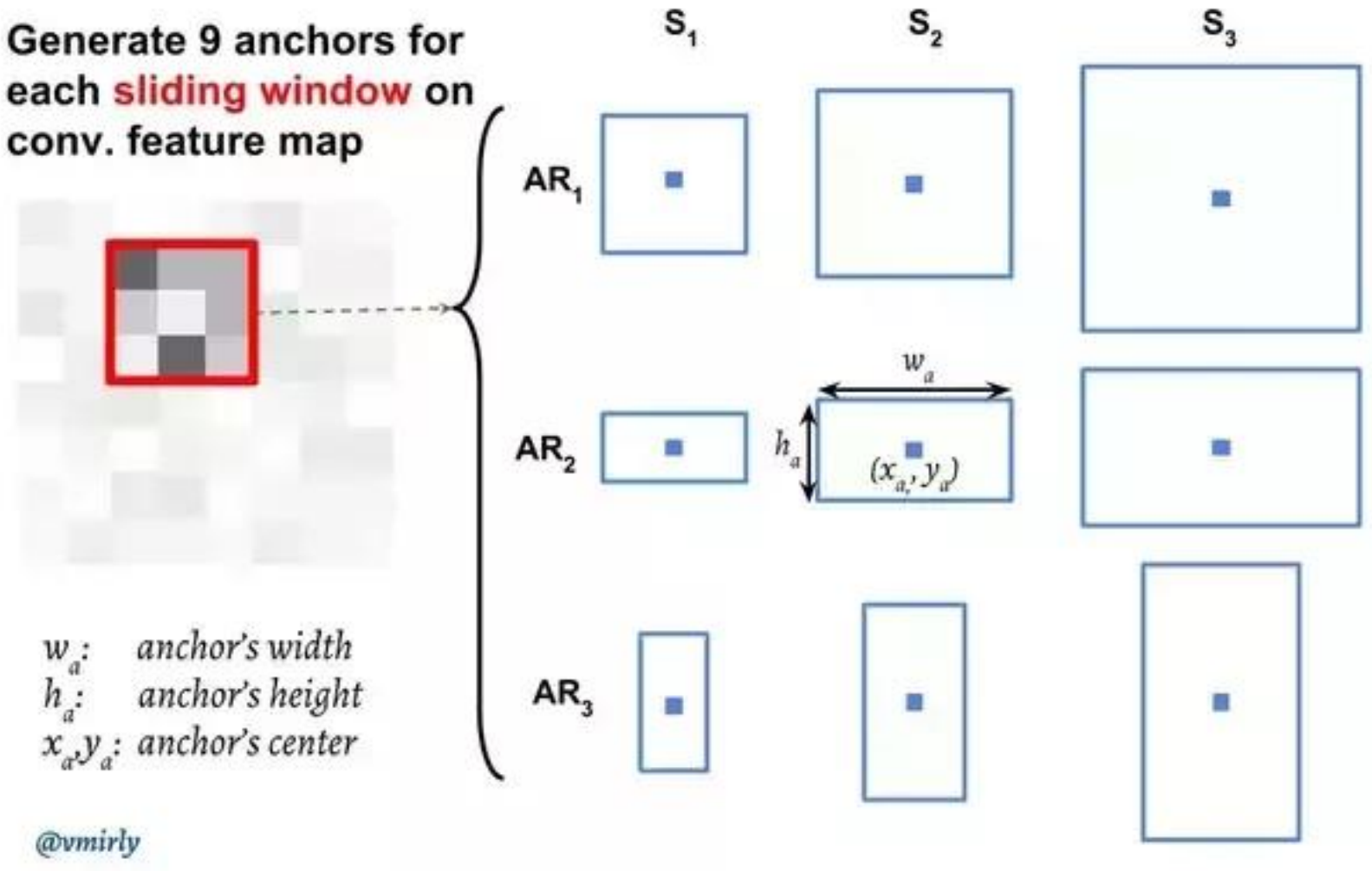
Region Proposal Network



RPN slides a window over the features of the CNN

Faster R-CNN: Anchor generation

- 9 anchors to detect objects with multi-scale, multi-aspect ratios



Faster R-CNN: Training

4-step training (note. Co-adaptive effect of RPN and detector)

- 1. Train RPN
  - Initialized with ImageNet-pre-trained model
- 2.train a detector network by fast R-CNN using the proposal generated by the step 1
  - Initialized with ImageNet-pre-trained model
- 3. Use the detector to initialize RPN training, but fix the shared convolutional layers and only fine tune the layers unique to RPN
- 4.keeping the shared convolutional layers fixed, we fine-tune the unique layers of fast R-CNN

Multi-Scale and Size Detection in Faster R-CNN

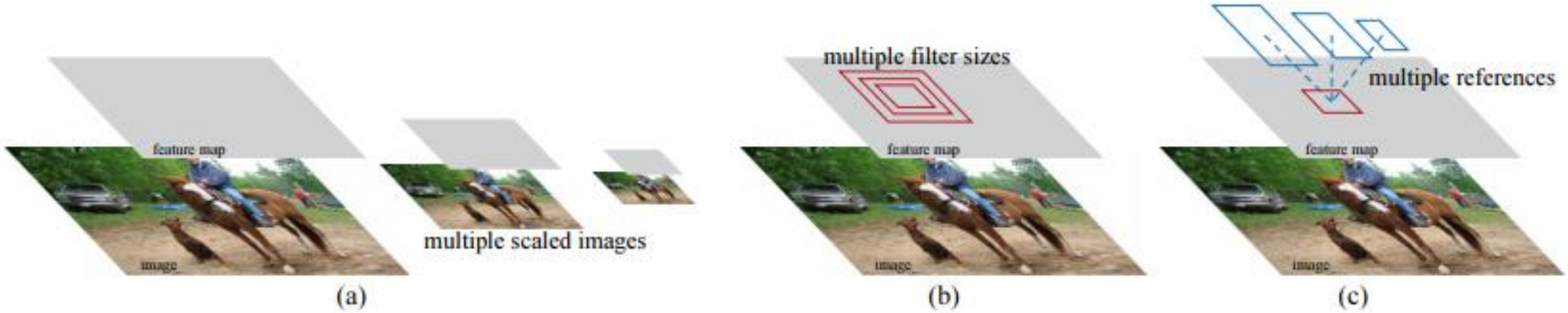


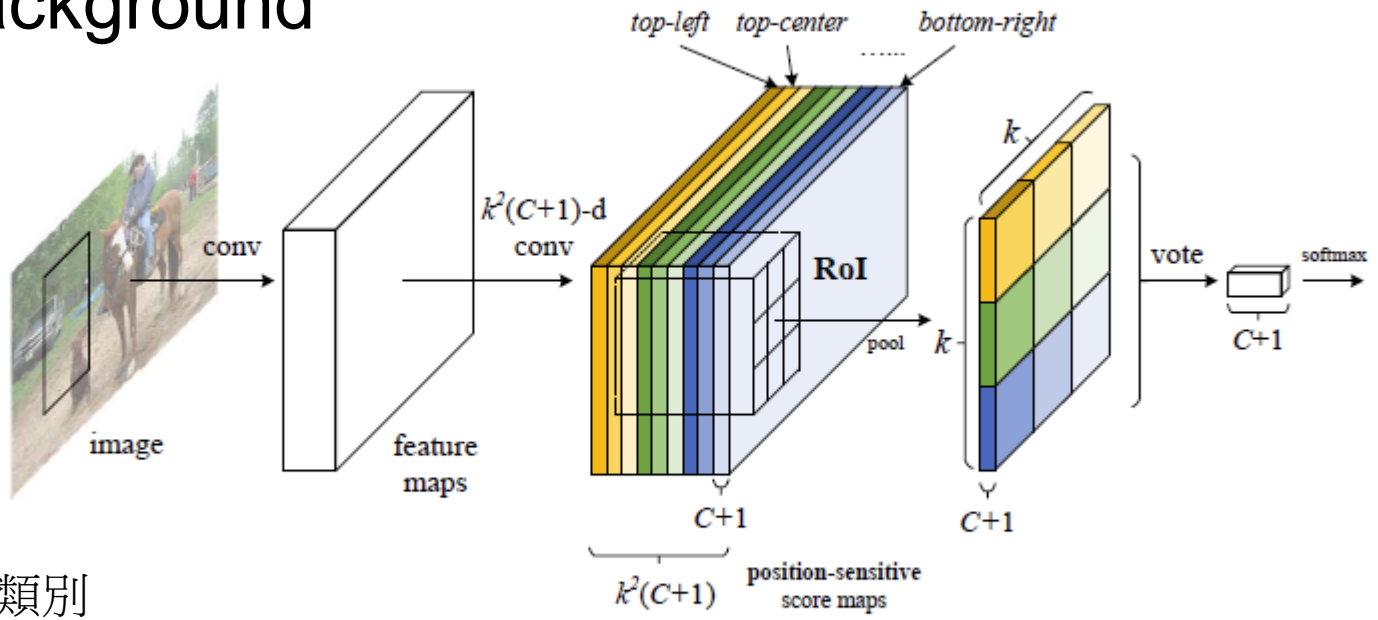
Figure 1: Different schemes for addressing multiple scales and sizes. (a) Pyramids of images and feature maps are built, and the classifier is run at all scales. (b) Pyramids of filters with multiple scales/sizes are run on the feature map. (c) We use pyramids of reference boxes in the regression functions.

R-FCN: Object Detection via Region-based Fully Convolutional Networks

- Drawbacks of Faster R-CNN
  - Subnetwork computation per region
  - Dilemma of translation invariance for classification v.s. translation variance for object detection
  - Can we use fully convolutional network for object detection as well?
  - Aka. Share all computations?
- Hypothesis
  - deeper convolutional layers in an image classification network are less sensitive to translation
  - construct a set of position-sensitive score maps by using a bank of specialized convolutional layers as the FCN output

R-FCN

- Concept: position sensitive ROI pooling
  - To explicitly encode position information into each RoI
- Divide each RoI rectangle into k × k bins by a regular grid
- C: class number, +1: background



Kxk bins: encode position information
channel數目為c+1的: 每個channel 是一個類別

Figure 1: Key idea of **R-FCN** for object detection. In this illustration, there are $k \times k = 3 \times 3$ position-sensitive score maps generated by a fully convolutional network. For each of the $k \times k$ bins in an RoI, pooling is only performed on one of the k^2 maps (marked by different colors).

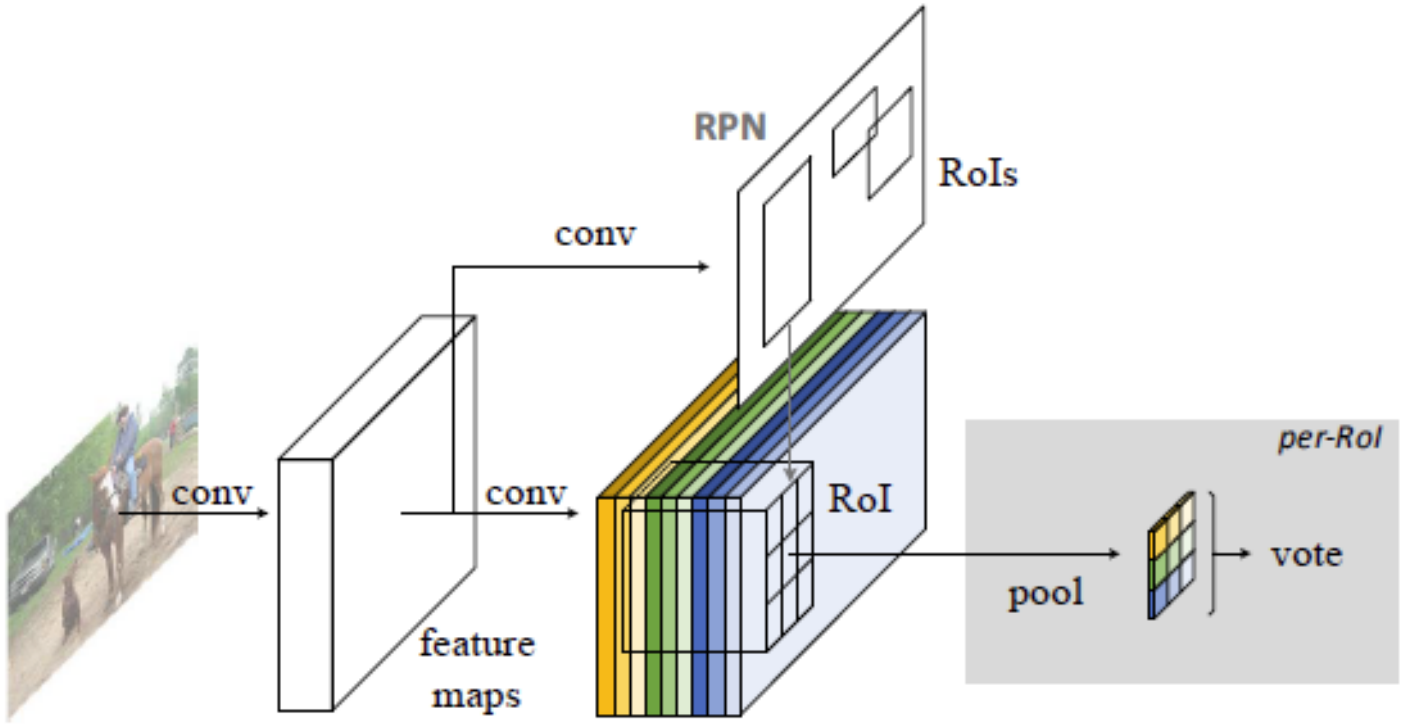


Figure 2: Overall architecture of R-FCN. A Region Proposal Network (RPN) [18] proposes candidate RoIs, which are then applied on the score maps. All learnable weight layers are convolutional and are computed on the entire image; the per-RoI computational cost is negligible.

Position Sensitive ROI Pooling

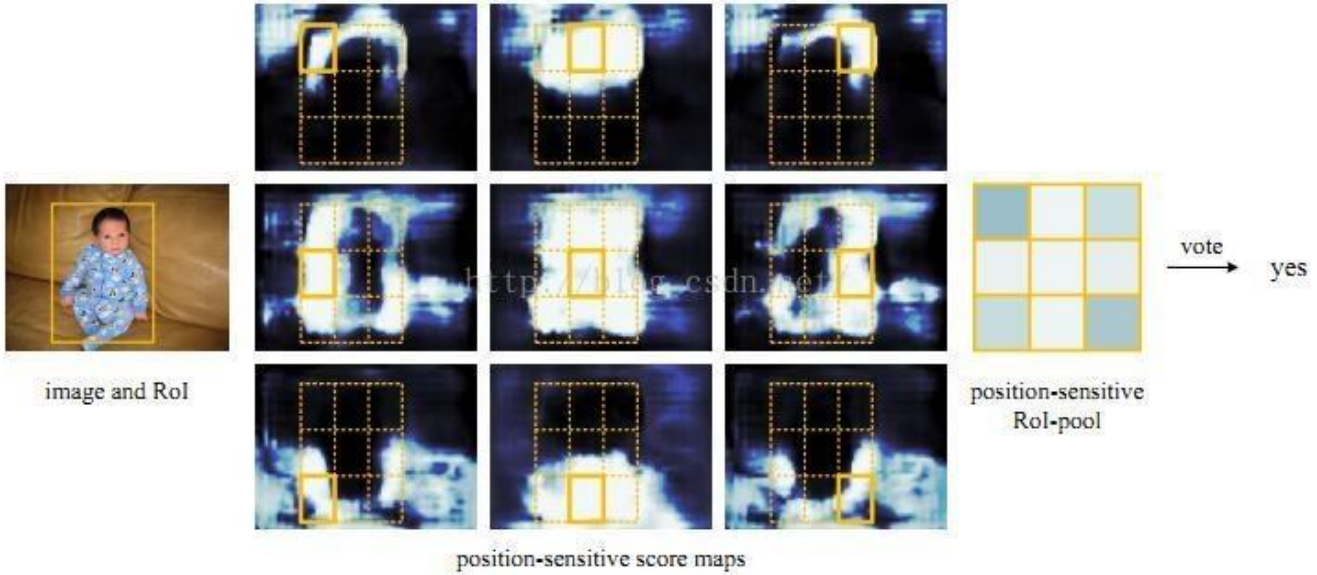


Figure 3: Visualization of R-FCN ($k \times k = 3 \times 3$) for the *person* category.

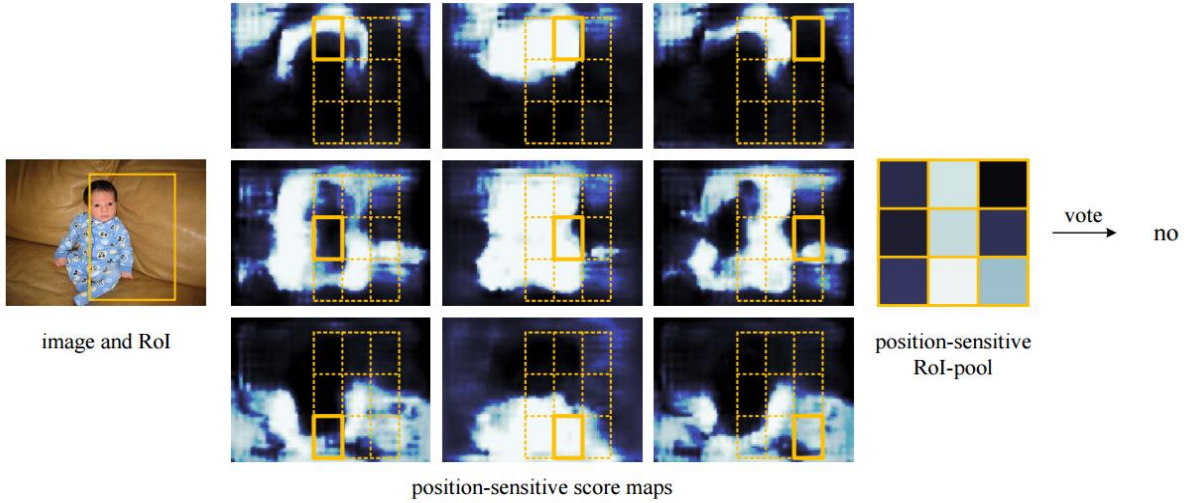


Figure 4: Visualization when an RoI does not correctly overlap the object.

位置敏感分數圖：
對feature map，將RoI矩形分成k*k個網格，針對每一個位置為（i，j）處分數的計算，主要不僅針對該處的softmax響應的分數，同時結合了其相對於RoI的位置

為了簡化計算，對於一個RoI區域內計算均值作為該RoI的vote

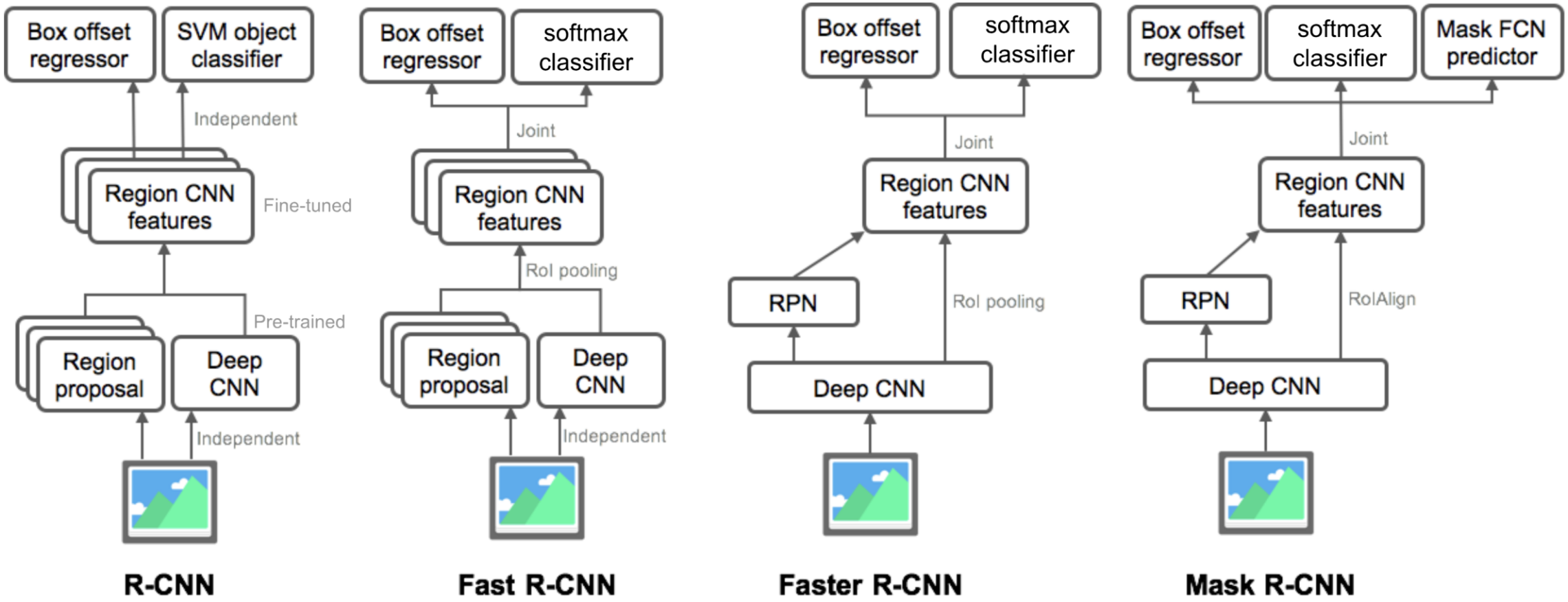
R-FCN: Faster Results

Table 3: Comparisons between Faster R-CNN and R-FCN using ResNet-101. Timing is evaluated on a single Nvidia K40 GPU. With OHEM, N RoIs per image are computed in the forward pass, and 128 samples are selected for backpropagation. 300 RoIs are used for testing following [18].

	depth of per-RoI subnetwork	training w/ OHEM?	train time (sec/img)	test time (sec/img)	mAP (%) on VOC07
Faster R-CNN	10		1.2	0.42	76.4
R-FCN	0		0.45	0.17	76.6
Faster R-CNN	10	✓ (300 RoIs)	1.5	0.42	79.3
R-FCN	0	✓ (300 RoIs)	0.45	0.17	**79.5**
Faster R-CNN	10	✓ (2000 RoIs)	2.9	0.42	*N/A*
R-FCN	0	✓ (2000 RoIs)	0.46	0.17	79.3

Table 6: Comparisons on MS COCO dataset using **ResNet-101**. The COCO-style AP is evaluated @ IoU $\in [0.5, 0.95]$. AP@0.5 is the PASCAL-style AP evaluated @ IoU $= 0.5$.

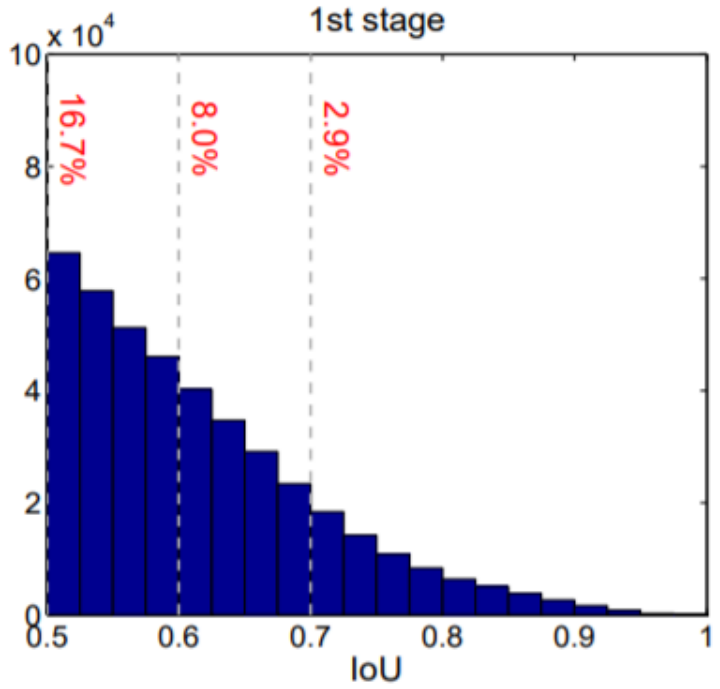
	training data	test data	AP@0.5	AP	AP small	AP medium	AP large	test time (sec/img)
Faster R-CNN [9]	train	val	48.4	27.2	6.6	28.6	45.0	0.42
R-FCN	train	val	48.9	27.6	8.9	30.5	42.0	0.17
R-FCN multi-sc train	train	val	49.1	27.8	8.8	30.8	42.2	0.17
Faster R-CNN +++ [9]	trainval	test-dev	**55.7**	**34.9**	15.6	38.7	50.9	3.36
R-FCN	trainval	test-dev	51.5	29.2	10.3	32.4	43.3	0.17
R-FCN multi-sc train	trainval	test-dev	51.9	29.9	10.8	32.8	45.0	0.17
R-FCN multi-sc train, test	trainval	test-dev	**53.2**	**31.5**	14.3	35.5	44.2	1.00

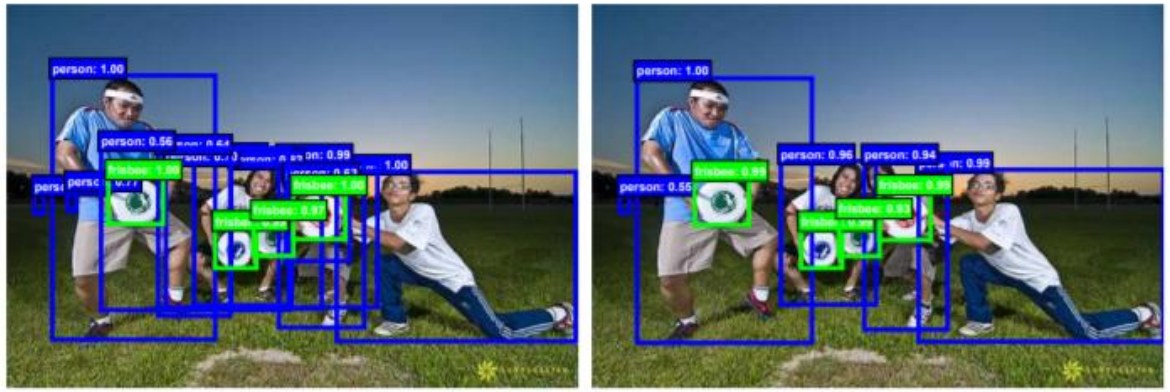

Box offset regressor
SVM object classifier
Independent
Region CNN features
Fine-tuned
Pre-trained
Region proposal
Deep CNN
Independent
R-CNN
Box offset regressor
softmax classifier
Joint
Region CNN features
RoI pooling
Region proposal
Deep CNN
Independent
Fast R-CNN
Box offset regressor
softmax classifier
Joint
Region CNN features
RPN
RoI pooling
Deep CNN
Faster R-CNN
Box offset regressor
softmax classifier
Mask FCN predictor
Joint
Region CNN features
RPN
RoIAlign
Deep CNN
Mask R-CNN

Cascade R-CNN

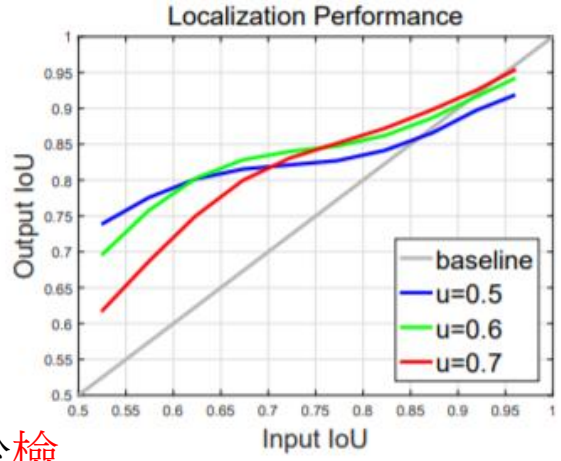
- R-CNN
  - Training 用低IoU閾值來train (靠GT選出300 proposals)，
    - training階段的輸入proposals質量更高(被採樣過，IoU>threshold)，
  - 但inference階段沒有辦法對這些proposals採樣（inference階段肯定不知道gt的，也就沒法計算iou）
    - inference階段的輸入proposals質量相對較差（沒有被採樣過，可能包括很多IoU<threshold的）
- 如果提高IoU閾值，選高質量proposal，則會導致兩個問題：
  - IoU提高，訓練中positive samples的數量會指數級減少，導致overfitting
  - 預測和訓練用不同閾值，會導致mismatch。
- Cascade R-CNN
  - 採取cascade的方式能夠讓每一個stage的detector都專注於檢測IOU在某一範圍內的proposal，因為輸出IOU普遍大於輸入IOU，因此檢測效果會越來越好。

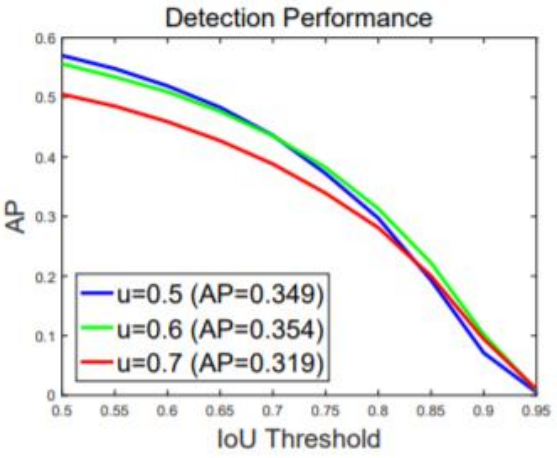
Cascade R-CNN: Delving into High Quality Object Detection





(a) Detection of $u = 0.5$ (b) Detection of $u = 0.7$



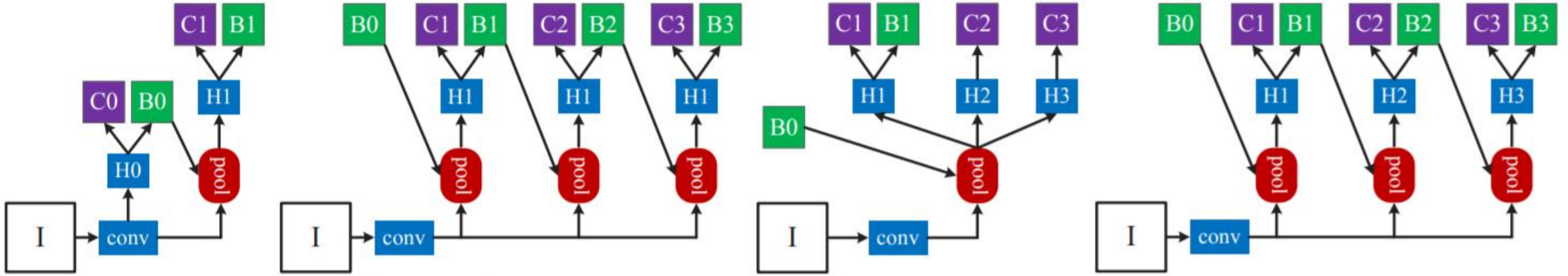


(c) Regressor (d) Detector

Figure 1. The detection outputs, localization and detection performance of object detectors of increasing IoU threshold u.

採取cascade的方式能夠讓每一個stage的detector都專注於檢測IOU在某一範圍內的proposal，因為輸出IOU普遍大於輸入IOU，因此檢測效果會越來越好。

Cascade R-CNN: Delving into High Quality Object Detection



(a) Faster R-CNN (b) Iterative BBox at inference (c) Integral Loss (d) Cascade R-CNN

Figure 3. The architectures of different frameworks. "I" is input image, "conv" backbone convolutions, "pool" region-wise feature extraction, "H" network head, "B" bounding box, and "C" classification. "B0" is proposals in all architectures.

- H0"可以看作是RPN 網絡，產生初步的proposal——"B0"，
  - "H1"，"H2"，"H3"是R-CNN網絡，進一步對產生的proposal進行finetune，"B1 "，"B2"，"B3"是每個級聯的檢測網絡的bbox輸出
- （b）是迭代式的bbox回歸 (refinement, but H1 from initial training)
  - 主要思想就是前一個檢測模型回歸得到的bbox坐標初始化下一個檢測模型的bbox，然後繼續回歸，這樣迭代三次後得到結果。
- （c）是Integral Loss (integrate different IOU threshold results)
  - 表示對輸出bbox的標籤界定採取不同的IOU閾值，因為當IOU較高時，雖然預測得到bbox很準確，但是也會丟失一些bbox。檢測模型基於不同的IOU閾值訓練得到
- （d）就是本文提出的cascade-R-CNN。(H1, H2, H3, refined model)

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
YOLOv2 [26]	DarkNet-19	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [23]	ResNet-101	31.2	50.4	33.3	10.2	34.5	49.8
RetinaNet [22]	ResNet-101	39.1	59.1	42.3	21.8	42.7	50.2
Faster R-CNN+++ [16]*	ResNet-101	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [21]	ResNet-101	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN w FPN+ (ours)	ResNet-101	38.8	61.1	41.9	21.3	41.8	49.8
Faster R-CNN by G-RMI [17]	Inception-ResNet-v2	34.7	55.5	36.7	13.5	38.1	52.0
Deformable R-FCN [5]*	Aligned-Inception-ResNet	37.5	58.0	40.8	19.4	40.1	52.5
Mask R-CNN [14]	ResNet-101	38.2	60.3	41.7	20.1	41.1	50.2
AttractioNet [10]*	VGG16+Wide ResNet	35.7	53.4	39.3	15.6	38.0	52.7
Cascade R-CNN	ResNet-101	**42.8**	**62.1**	**46.3**	**23.7**	**45.5**	**55.2**

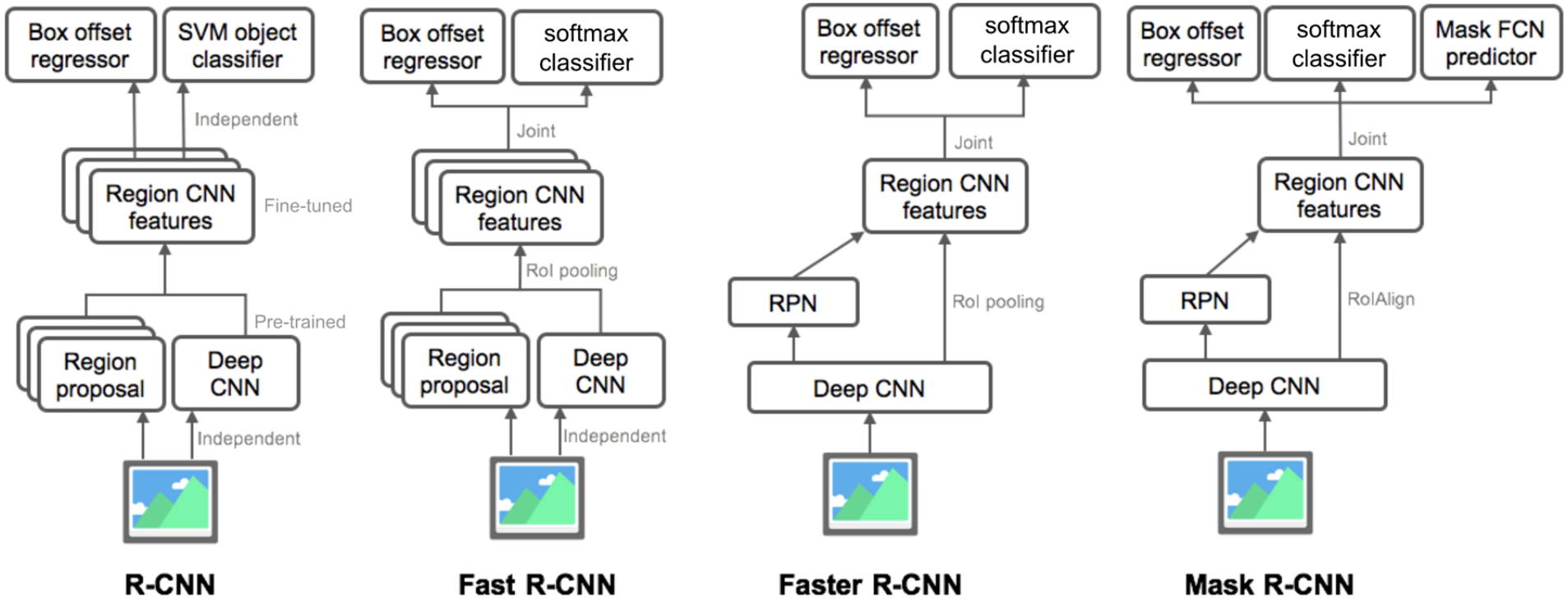
Table 5. Comparison with the state-of-the-art *single-model* detectors on COCO `test-dev`. The entries denoted by "*" used bells and

2到4個點的AP提升

OBJECT DETECTION: SINGLE STAGE APPROACH

Problems with 2 step detection

- Complex Pipeline
- Slow (Cannot run in real time)
- Hard to optimize each component



YOLO (YOU ONLY LOOK ONCE)

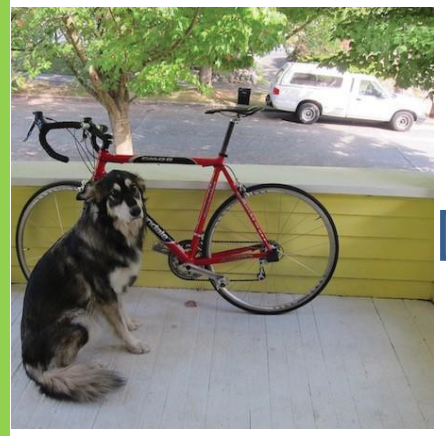
YOLO

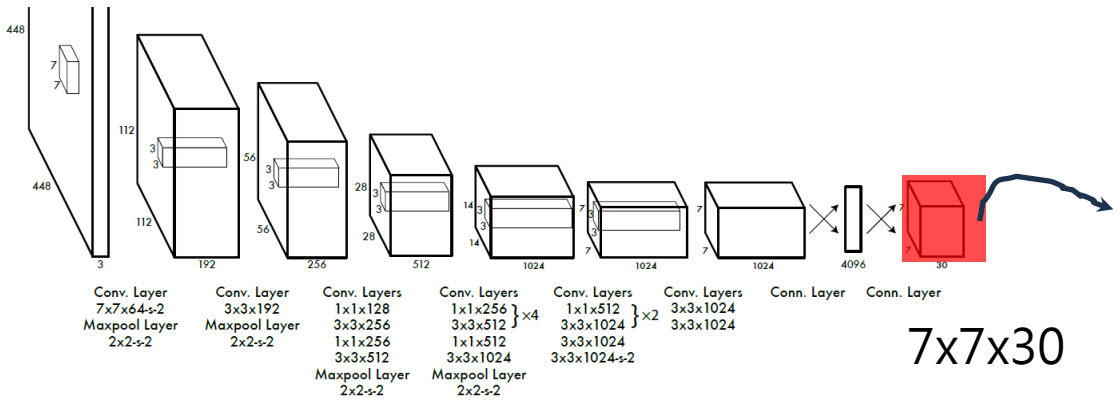
- Treat object detection as an regression problem
- Output of CNN: S x S x (5 * B + C) tensor of predictions

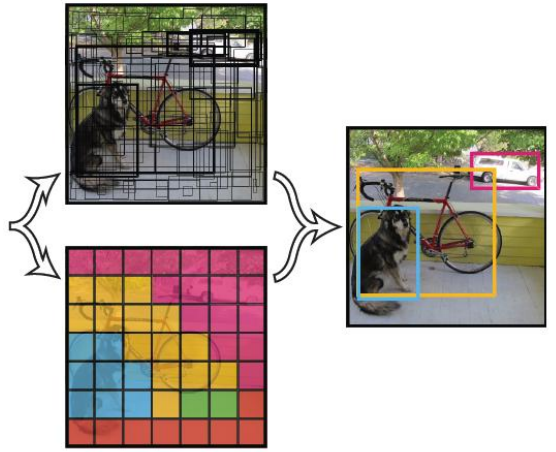
Resize to 448x448

CNN Pretrained on ImageNet

分類與定位





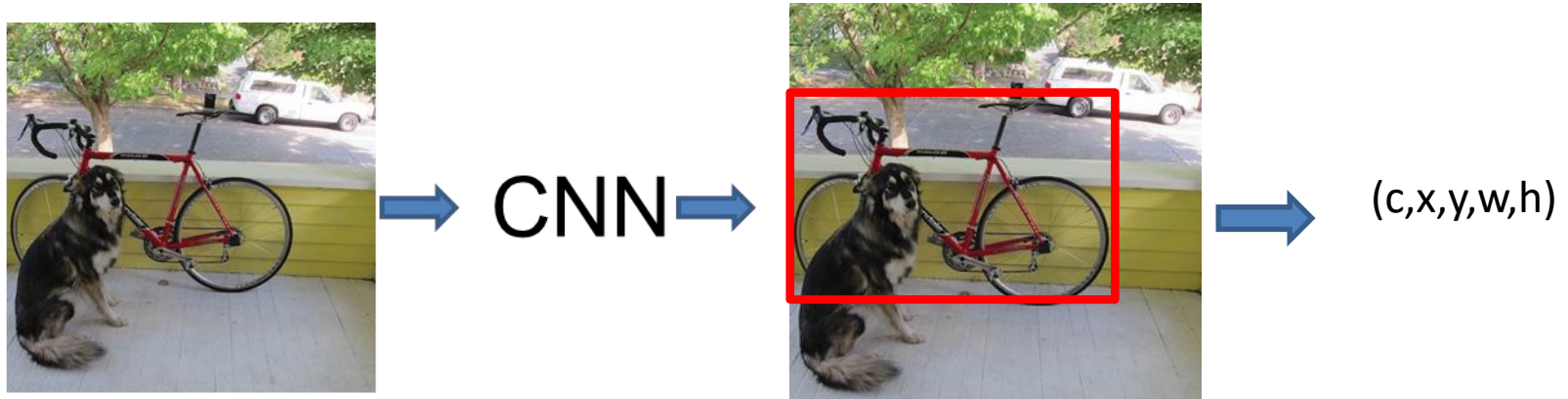


Joseph Redmon, Santosh Divvala, Ross Girshick , Ali Farhadi. "You Only Look Once: Unified, Real-Time Object Detection" . CVPR 2016.

- RCNN: 用分類的算法
  - Sliding window
  - 輸入是一個框的內容，輸出是(前景/背景)
    - 第1個問題：
      - 框有不同的大小，對於不同大小的框，輸入到相同的二分類器中嗎？
      - 是的。要先把不同大小的input歸一化到統一的大小 (warping)。
    - 第2個問題：
      - 背景圖片很多，前景圖片很少：二分類樣本不均衡
      - Region proposal

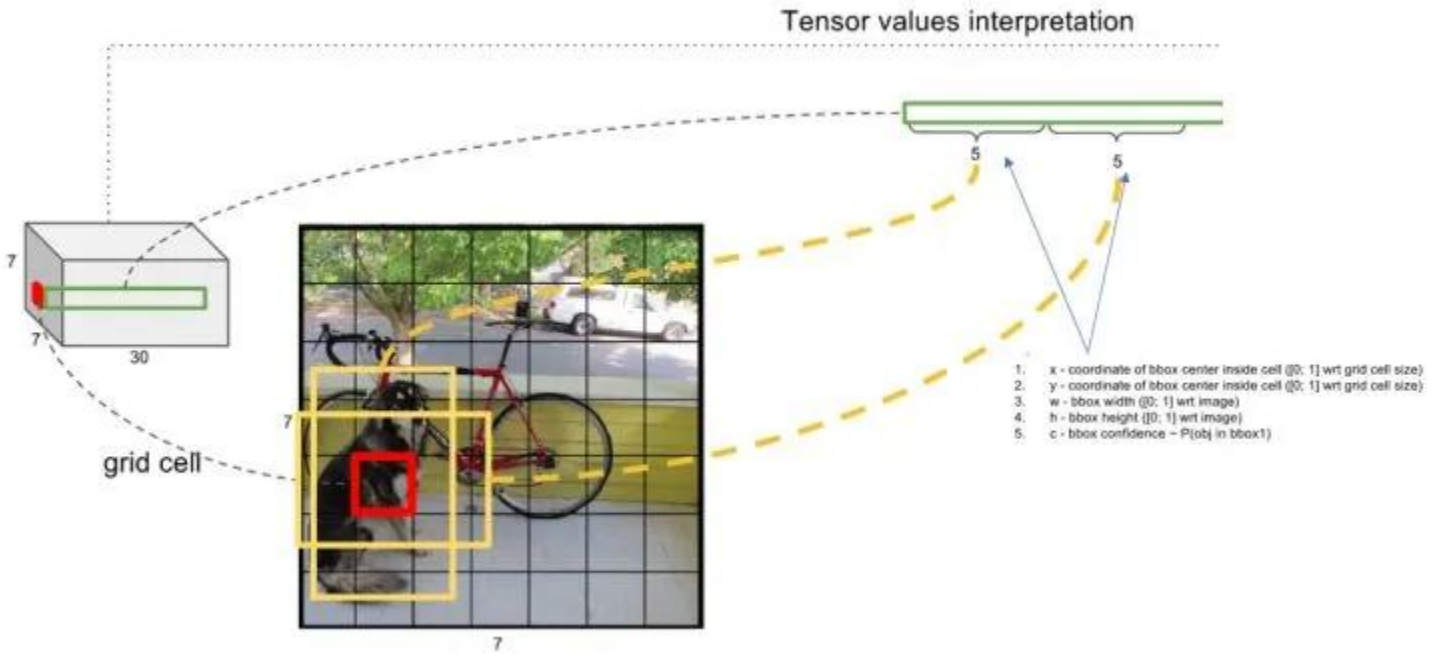
(bg/fg)

- YOLO:問題轉化成一個回歸問題，直接回歸出Bounding Box的位置
  - 找1000張圖片，把label設置為(1,x, y, w, h)。有了數據和標籤，就完成
  - 輸出換成(c,x,y,w,h)，c表示confidence置信度

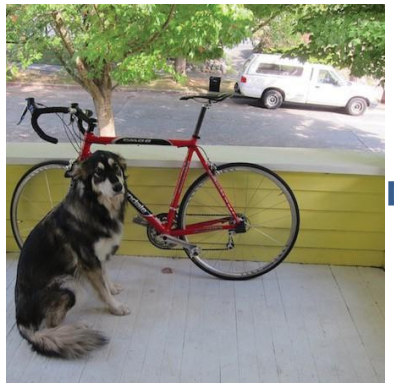


YOLOv1

- 需求1：naïve YOLO 只能輸出一個目標，那多個目標怎麼辦呢？
  - 為了保證所有目標都被檢測到，我們應該輸出盡量多的目標。
  - 這種方法也不是最優的，用一個(c,x,y,w,h)去負責image某個區域的目標。
    - 比如說圖片設置為49個區域，每個區域用1個(c,x,y,w,h)去負責：

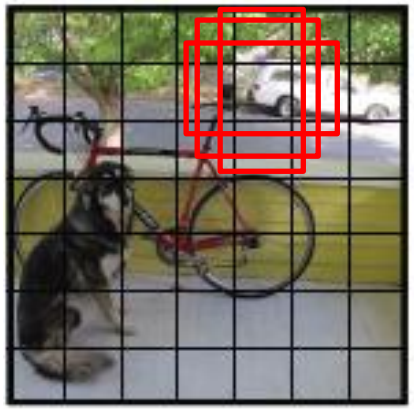


YOLO v1



Input image

CNN



Divide into grid S x S

each grid cell

B Bounding box + confidence

找出object 位置(定位)

Joseph Redmon, Santosh Divvala, Ross Girshick , Ali Farhadi. “You Only Look Once: Unified, Real-Time Object Detection”. CVPR 2016.

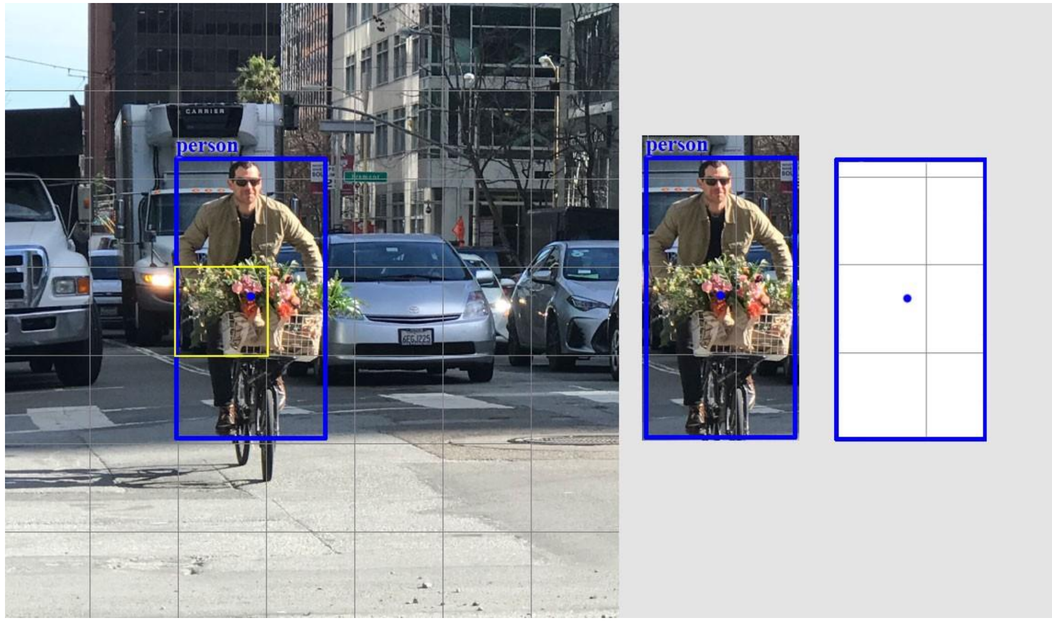
Bounding Box + Confidence物體定位

- bounding boxes 座標起始點 dx, dy
  - bounding box 的中心與 grid cell 方格邊界的相對值
- bounding boxes 長寬 dh, dw

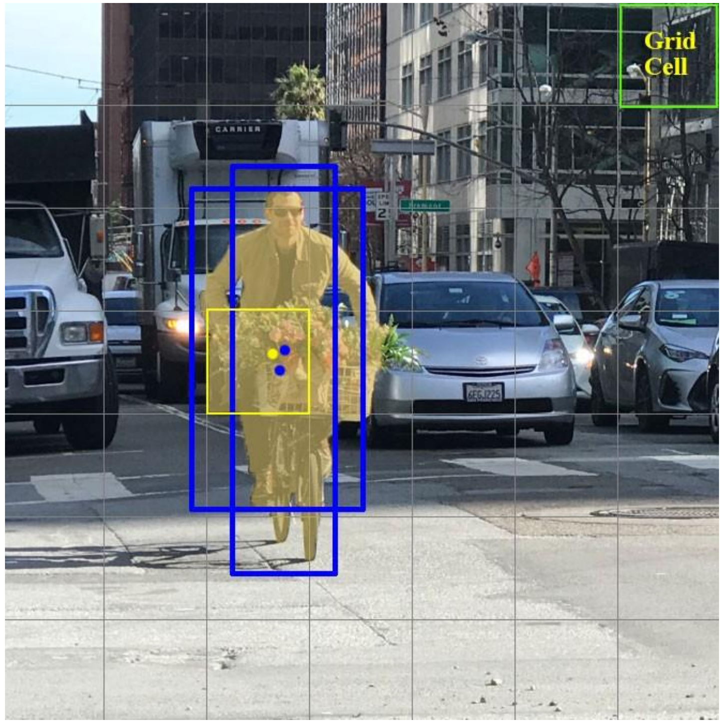
- 正式的信心分數 $Pr(Object)*IOU_{truthpred}$

Joseph Redmon, Santosh Divvala, Ross Girshick , Ali Farhadi. “You Only Look Once: Unified, Real-Time Object Detection”. CVPR 2016.

- 物體跨多區域，c的真值該怎麼設置呢？
  - 假設物件跨了4個區域(grid)，但只能某一個grid的c=1，其他的c=0。那麼該讓哪一個grid的c=1呢？就看**物件中心**落在了哪個grid裡面。
- 某一個grid裡面有2個目標，c的真值該怎麼設置呢？
  - Grid 切更細，或者更大，使區域更密集，就可以緩解多個目標的問題，但無法從根本上去解決。
- 按上面的設計你檢測得到了49個框，可是假設圖片上只有2個物件，怎麼從49個結果中篩選出2個我們要的呢？
  - NMS(非極大值抑制)。2個框重合度很高，大概率是一個目標，那就只取一個框。解決了提出的多個目標的問題



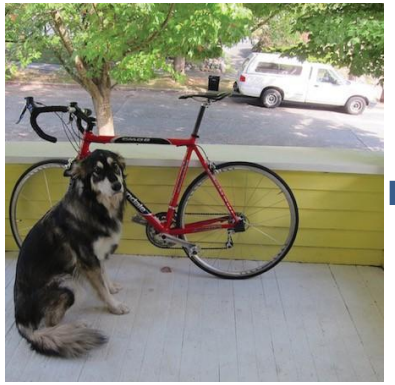
Each grid cell detects only one object.



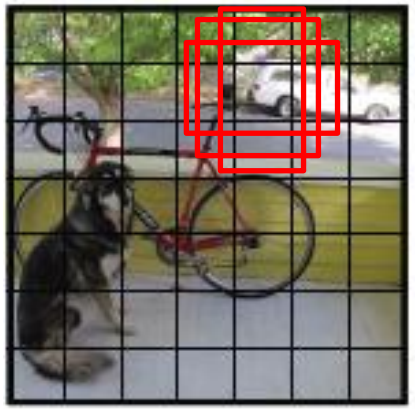
Each grid cell make a fixed number of boundary box guesses for the object.

YOLO may miss objects that are too close. For the picture , there are 9 Santas in the lower left corner but YOLO can detect 5 only.

each grid cell



CNN



找出object 位置(定位)

Input image

Divide into grid S x S

Bounding box + confidence

對物體作分類

Class probability map

Joseph Redmon, Santosh Divvala, Ross Girshick , Ali Farhadi. “You Only Look Once: Unified, Real-Time Object Detection”. CVPR 2016.

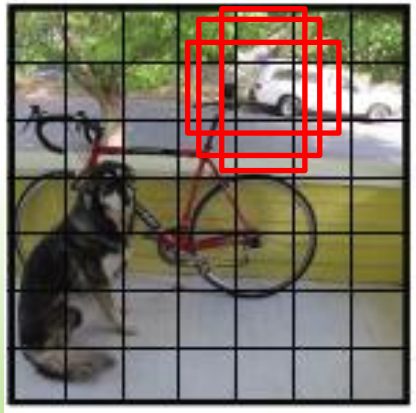
Class Probability Map

- $Pr(Class_i|Object)$
  - 是物體的條件下，是某一類物體如人或車的機率為多少
  - 不管方格grid cell 中包含的 bounding box 有多少個，每個方格grid cell 只 predict 每個類別的 conditional probabilities
- PASCAL VOC
  - S =7
  - B=2
  - C =20

在PASCAL VOC這個dataset中，YOLO對影像分成7x7方格，每個方格取2個bounding box，20類物體

Joseph Redmon, Santosh Divvala, Ross Girshick , Ali Farhadi. “You Only Look Once: Unified, Real-Time Object Detection”. CVPR 2016.

Within each grid cell:



Bounding box + confidence

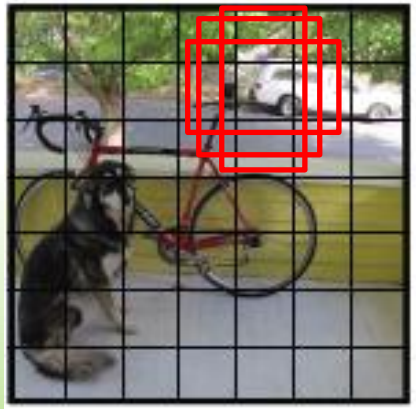
NMS
Score threshold

Class probability map

將每個 grid cell 的 conditional class probabilities 與每個 bounding box 的 confidence 相乘

Joseph Redmon, Santosh Divvala, Ross Girshick , Ali Farhadi. “You Only Look Once: Unified, Real-Time Object Detection”. CVPR 2016.

Within each grid cell:



$$Pr(Object) * IOU_{pred}^{truth}$$

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

$$Pr(Class_i|Object)$$

是不是物體的機率和
物體種類的條件機率，相乘以後
得到每個 bounding box 的每個類別的信心分數 confidence score。

這樣就把 bounding box 中預測的物體種類的機率
與是否為物體合在一起，同時有物體定位和物體分類的功能

Joseph Redmon, Santosh Divvala, Ross Girshick , Ali Farhadi. “You Only Look Once: Unified, Real-Time Object Detection”. CVPR 2016.

- 需求2：多類的目標怎麼辦呢？
  - 輸出變成**多分類**[c,x,y,w,h,one-hot]*N
- 需求3：小目標檢測怎麼辦呢？
  - 小目標總是檢測不佳，所以我們**專門設計神經元去擬合小目標**。
  - 每個區域，我們用2個五元組(c,x,y,w,h)，**一個負責回歸大目標，一個負責回歸小目標**，同樣添加one-hot vector，one-hot就是[0, 1],[1,0]這樣子，來表示屬於哪一類

YOLO

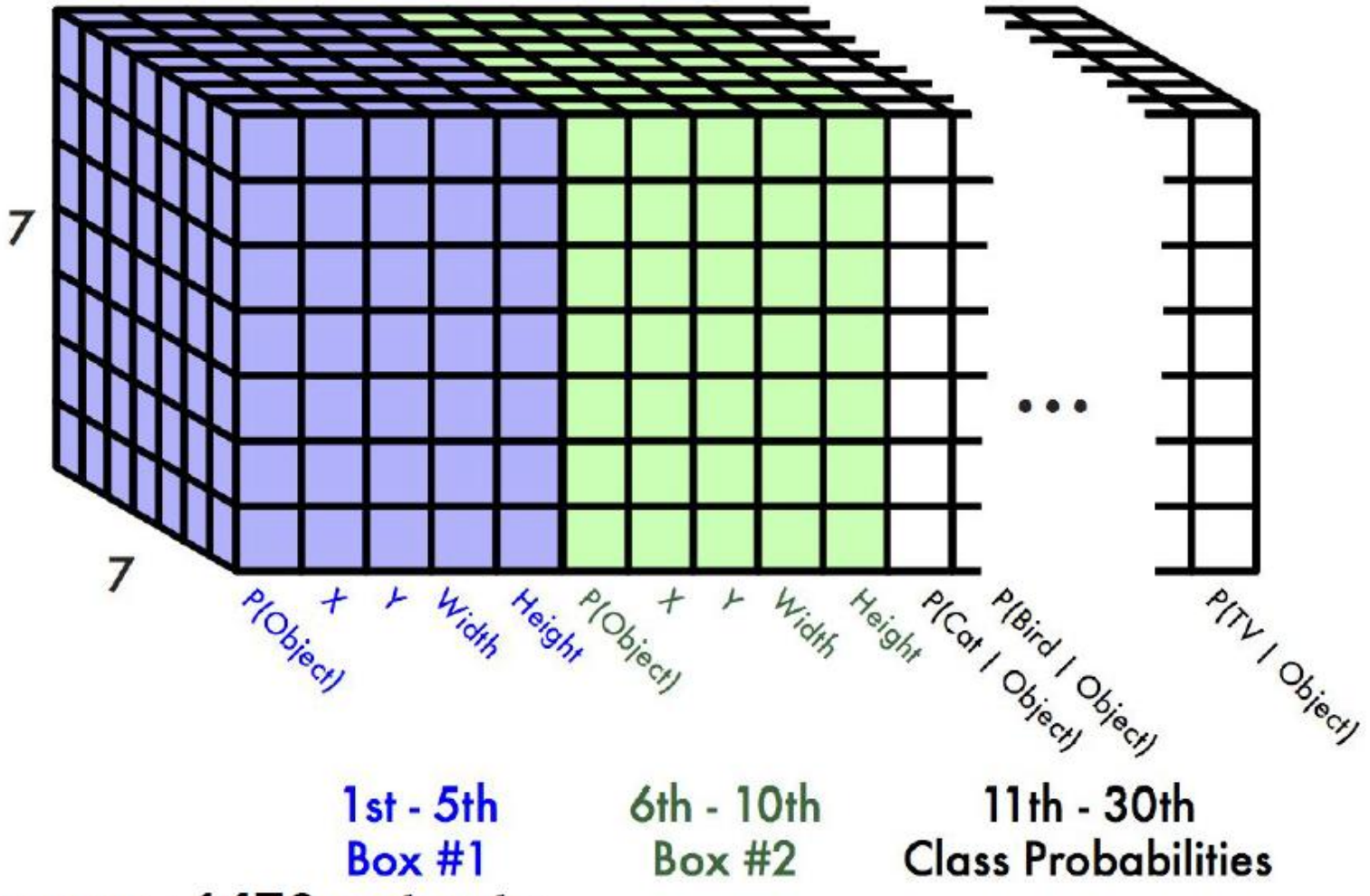
$$30 = 5 * 2(c_{big}, x_{big}, y_{big}, w_{big}, h_{big}, c_{small}, x_{small}, y_{small}, w_{small}, h_{small}) + 20 classes$$

Each cell predicts:

- For each bounding box:
  - 4 coordinates (x, y, w, h)
  - 1 confidence value
- Some number of class probabilities

For Pascal VOC:

- 7x7 grid
- 2 bounding boxes / cell
- 20 classes



7 x 7 x (2 x 5 + 20) = 7 x 7 x 30 tensor = **1470 outputs**

YOLO: Training

- train的時候用的小圖片，檢測的時候用的是大圖片
- Linear activation in final layer, leaky ReLU at all others
- Loss function
  - Only penalize classification error if there is an object at the grid cell
  - Only penalize bounding box error if that predictor is “responsible”
  - Weighting
    - to balance classification/localization error
  - Use square root of width/height
    - To balance sensitivity of large/small box

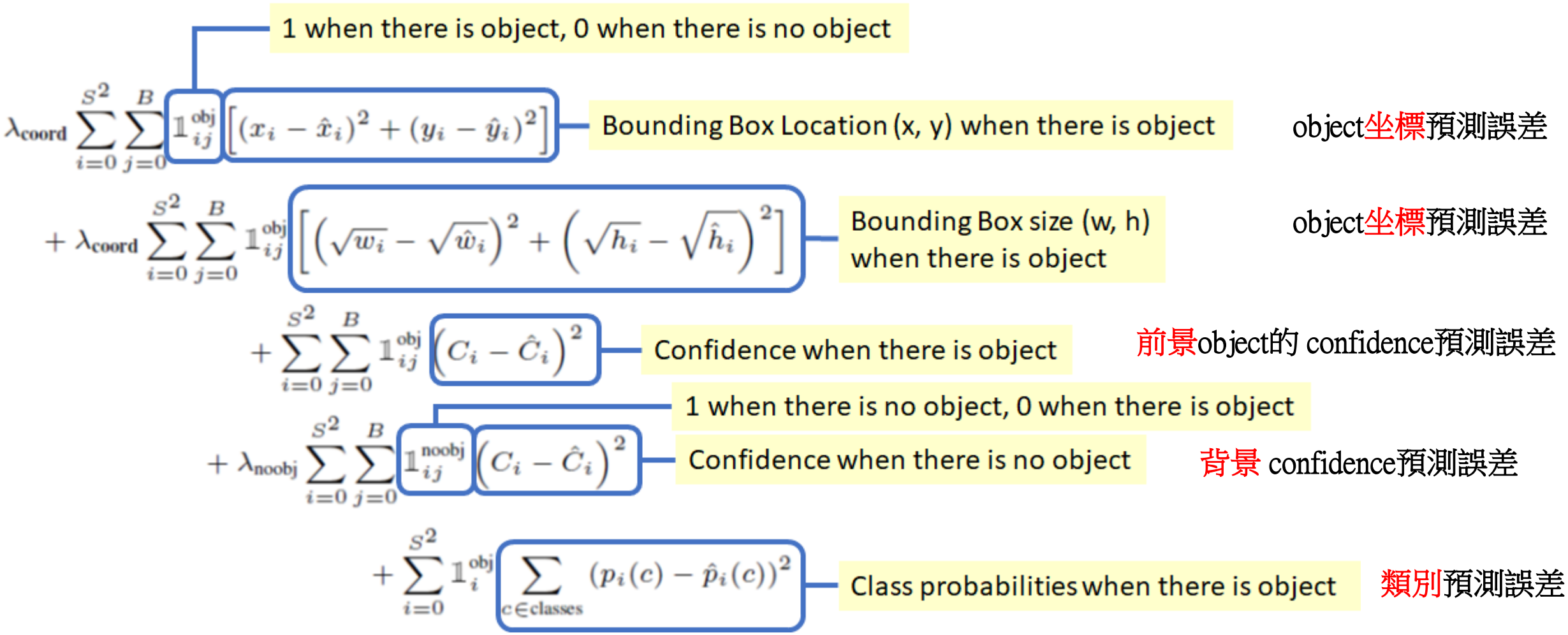
$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

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

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

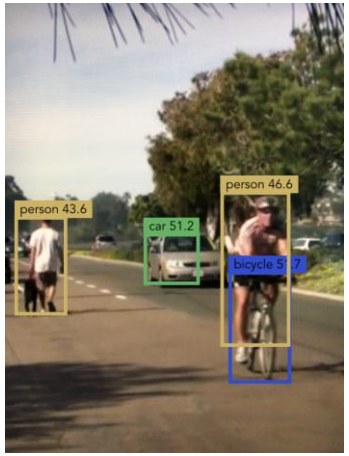
$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

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


1 when there is object, 0 when there is no object
Bounding Box Location (x, y) when there is object
object坐標預測誤差
Bounding Box size (w, h) when there is object
object坐標預測誤差
Confidence when there is object
前景object的 confidence預測誤差
1 when there is no object, 0 when there is object
Confidence when there is no object
背景 confidence預測誤差
Class probabilities when there is object
類別預測誤差

YOLO

- Very fast
  - Simple pipeline and regression problem formulation
  - End-to-end training
  - 45fps (standard version), 150fps (fast version with fewer CNN layers)
- Less background errors
  - Whole image processing enables better encoding of whole object context information, class information, and appearance information
- Learn more generalized representation of objects
  - Better at artwork than R-CNN
- Main source of error is localization error
  - lower performance than state-of-the-art
  - Strong constraints on bounding box predictions
  - Bad for small objects and unusual aspect ratios
- Improvements: YOLOv2, YOLO9000



deepsystems.io:Illustration of YOLO

- YOLO
  - Less BG error
  - Major location error

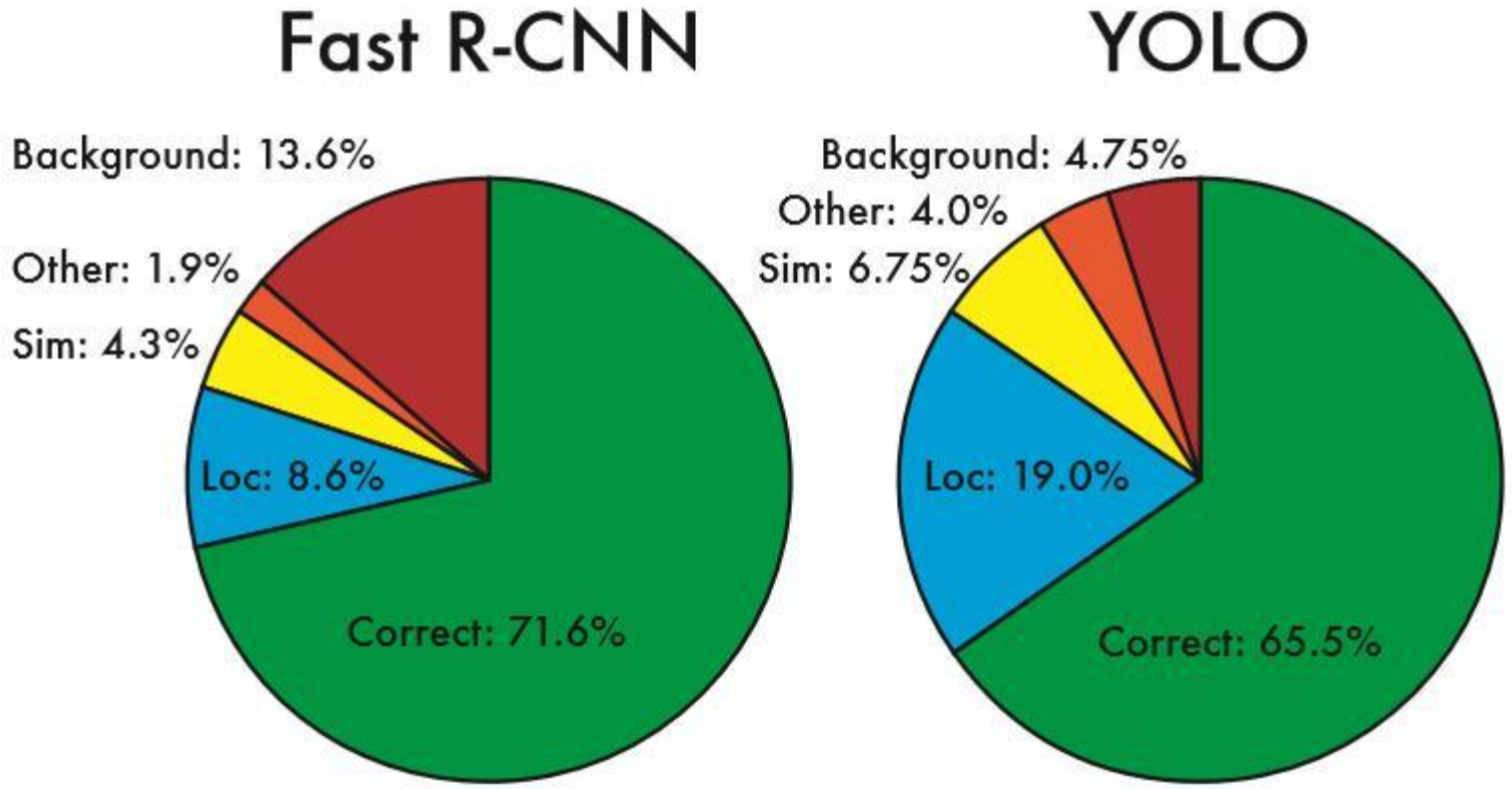


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

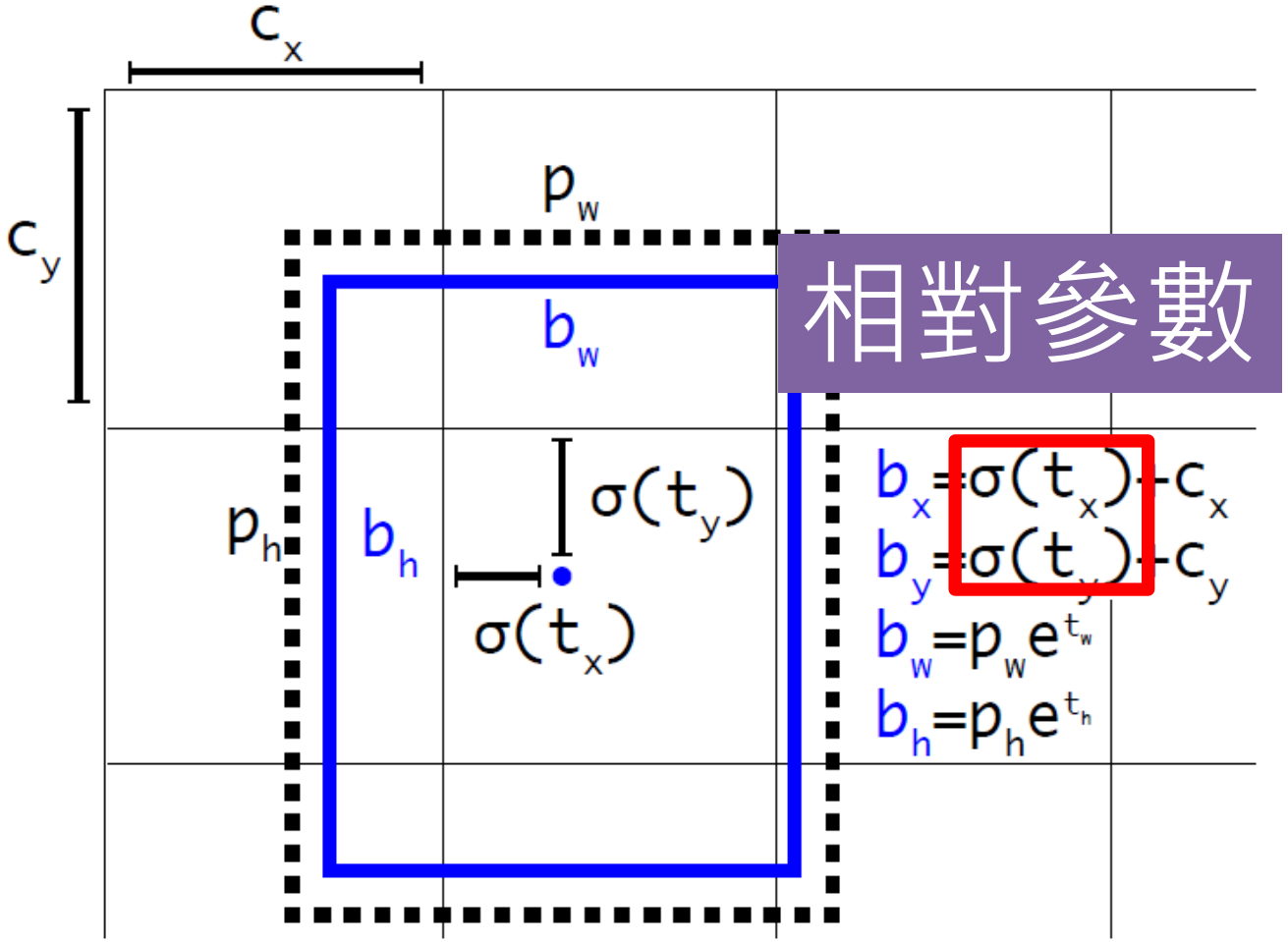
YOLO v2

- 檢測頭的改進：
  - YOLO v1雖然快，但是預測的框不准確，很多目標找不到：

- 預測的框不准確：準確度不足。
  - R-CNN，人家預測的是偏移量
  - 從YOLOV1直接預測位置改為預測一個偏移量，基於Anchor框的寬和高和grid的先驗位置的偏移量，得到最終目標的位置，這種方法也叫作location prediction
  - 這裡還涉及到一個尺寸問題，對這些值**歸一化，值很小，有利於神經網絡的學習**
  - YOLOV1直接預測位置會導致神經網絡在一開始訓練時不穩定，使用偏移量會使得訓練過程更加穩定，性能指標提升了5%左右
  - 位置上不使用Anchor框，寬高上使用Anchor框。

YOLO v2

預測邊界框中心點相對於對應cell左上角位置的相對偏移值，為了將邊界框中心點約束在當前cell中，使用sigmoid函式處理偏移值，這樣預測的偏移值在(0,1)範圍內



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

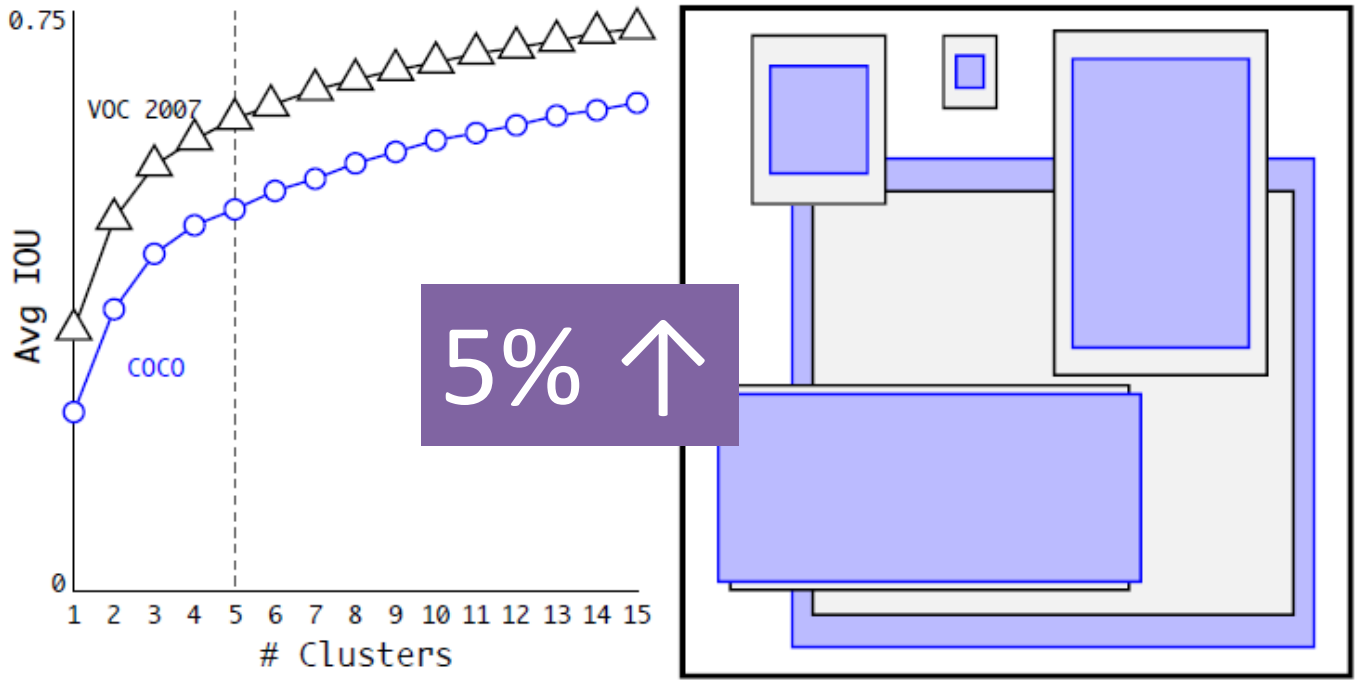
- 引入RCNN 的Anchor概念
  - 預先定義好的框，它的位置，寬高都是已知的，是一個參照物，供我們預測時參考
- C_x, C_y :
  - grid的左上角坐標，如圖所示。
- p_w, p_h :
  - Anchor的寬和高，這裡的anchor是人為定好的一個框，寬和高是固定的
- b_x, b_y, b_w, b_h
  - 模型最終得到的的檢測結果
- t_x, t_y, t_w, t_h :
  - 模型要預測的offset值。

每個cell的尺度為1

- 很多目標找不到：recall不足
  - YOLO v1一次能檢測多少個目標嗎？答案是49個目標。98個框，並且2個框對應一個類別，可以是大目標也可以是小目標。
  - YOLO v2首先把7x7個區域改為13x13個區域，每個區域有5個anchor，且每個anchor對應著1個類別
  - **每個區域的5個anchor是如何得到的？**
    - anchor是從數據集中統計得到的
- 但是小目標檢測仍然是YOLO v2的痛
  - 直到ResNet出現，backbone可以更深了，所以darknet53誕生 => YOLOv3

YOLO v2

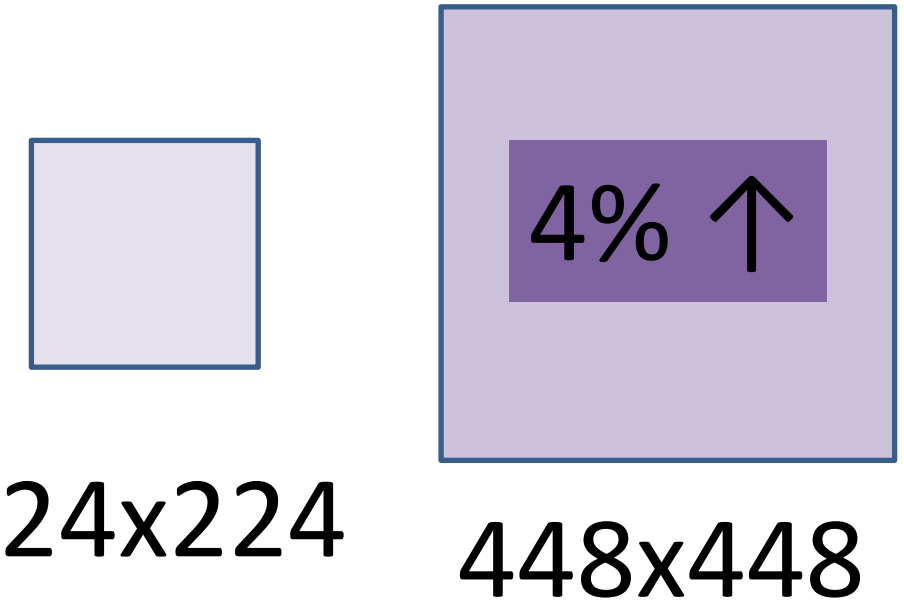
- 使用 K-Means 求 anchor box 比例



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

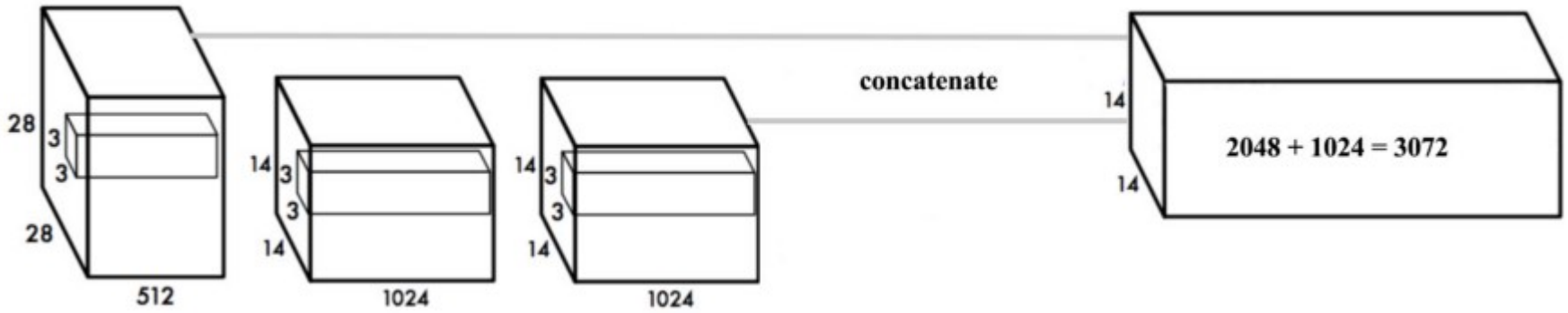
YOLO v2

- 高解析度分類器
  - 之前是訓練用224x224，偵測時用448x448，現在直接用448x448 資料做10個epoch fine tuning



Multi-Scale Training

- Fine grained features with passthrough like SSD
  - reshape the 28 × 28 × 512 layer to 14 × 14 × 2048
  - Concatenates



- Multi-scale training with data augmentation
  - During training, YOLO takes images of size 320×320, 352×352, … and 608×608 (with a step of 32).
  - For every 10 batches, YOLOv2 randomly selects another image size to train the model

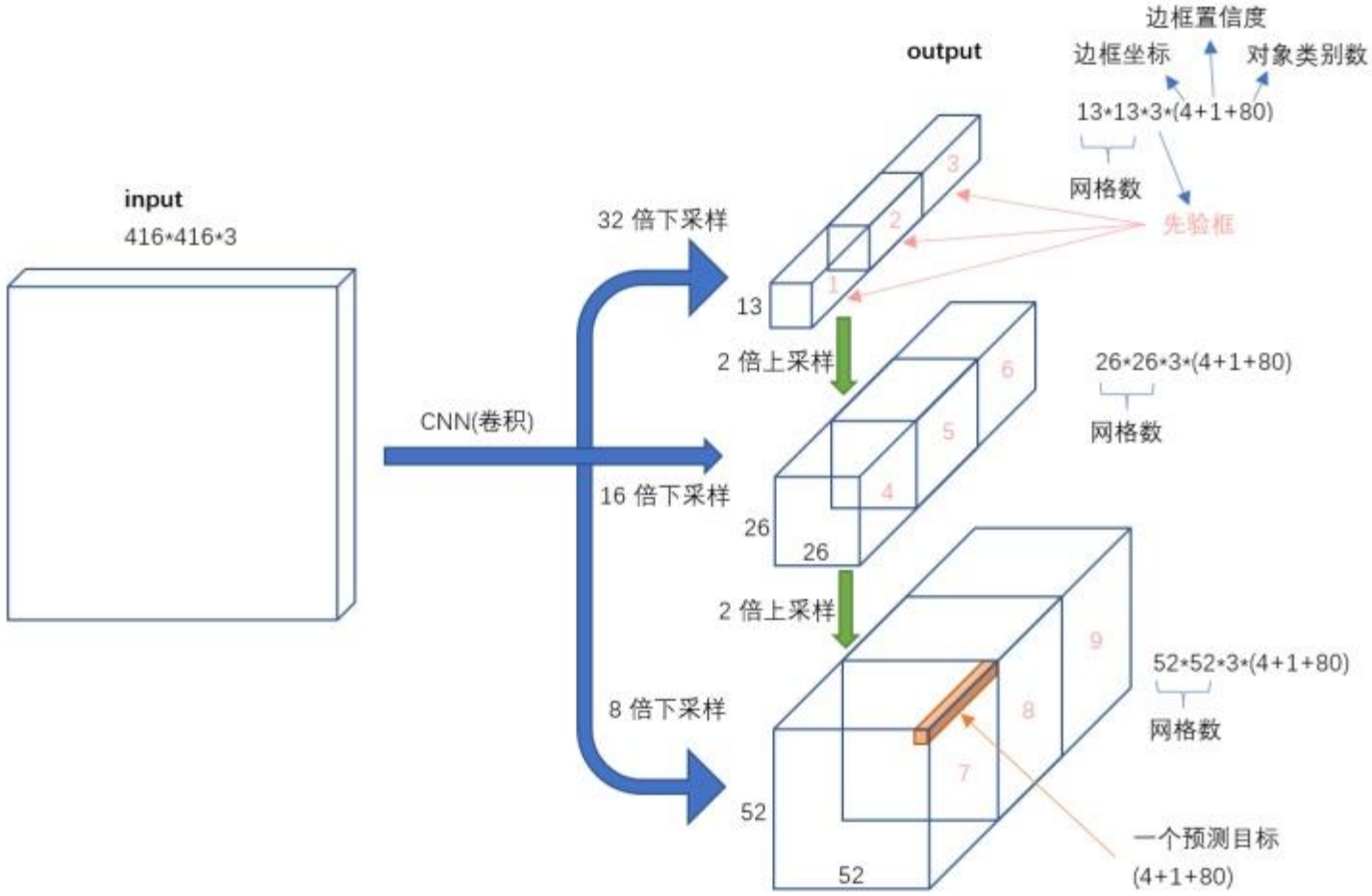
YOLO v2

	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**

15.2% ↑

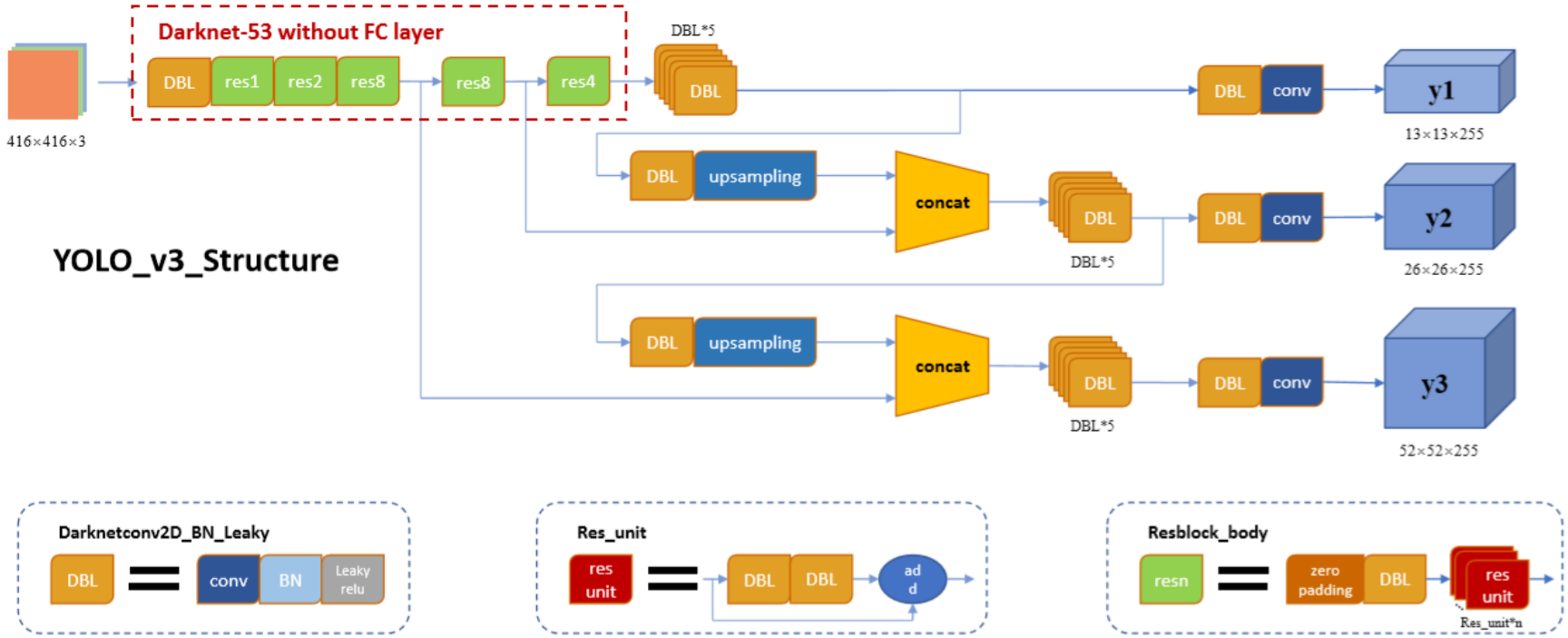
YOLO v3

- **檢測頭的改進**
  - 小目標檢測仍然是YOLO v2的痛，YOLO v3是如何改進的呢？
  - YOLO v3檢測頭分叉了，分成了3部分：
  - 13*13*3*(4+1+80)
  - 26*26*3*(4+1+80)
  - 52*52*3*(4+1+80)



– 3個分支分別為**32倍下採樣，16倍下採樣，8倍下採樣**，分別取預測**大，中，小目標**

- **32倍下採樣**每個點感受野更大，所以去預測**大目標**，**8倍下採樣**每個點感受野最小，所以去預測**小目標**。**專人專事**。

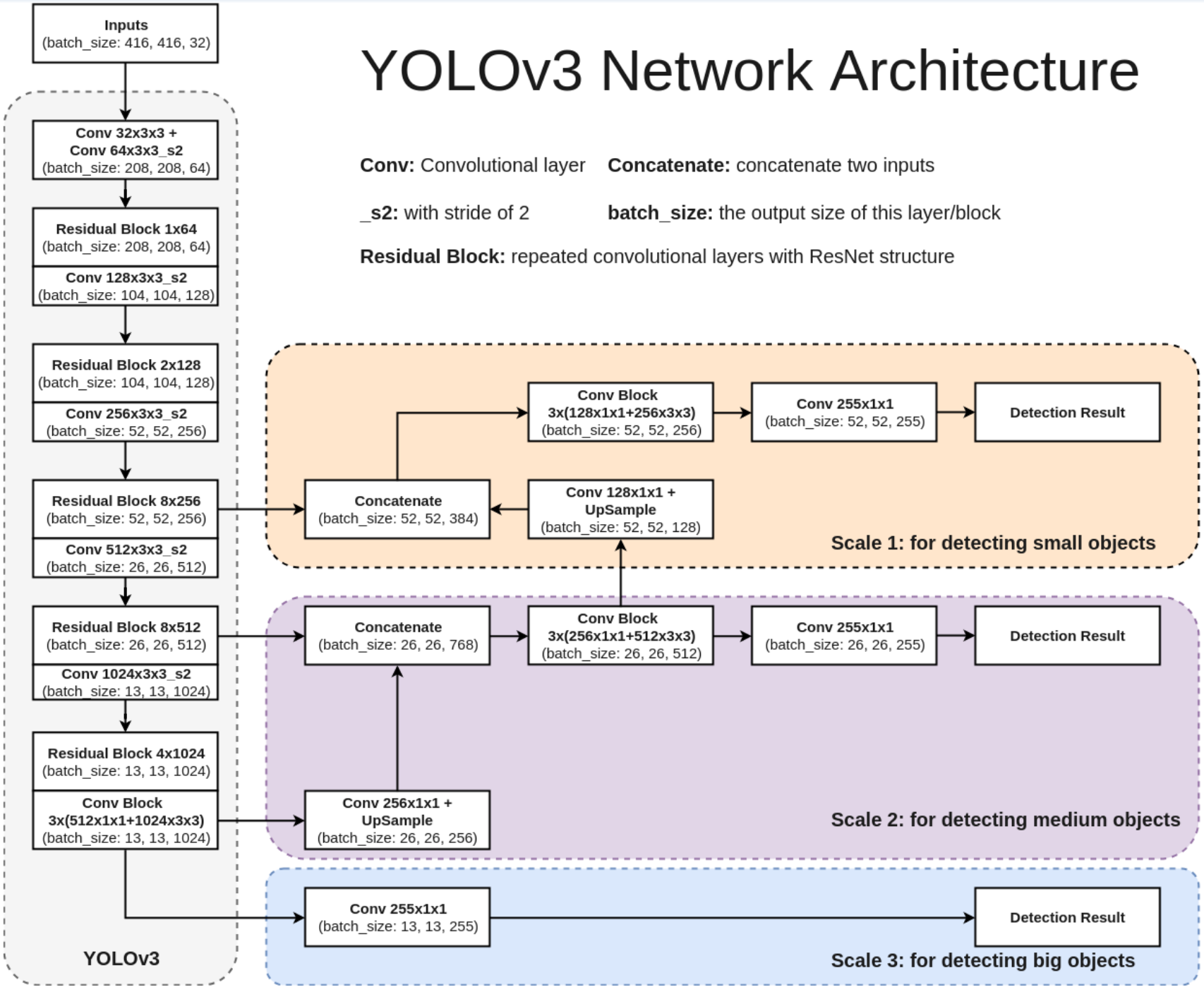


YOLOv3 Network Architecture

Conv: Convolutional layer **Concatenate:** concatenate two inputs

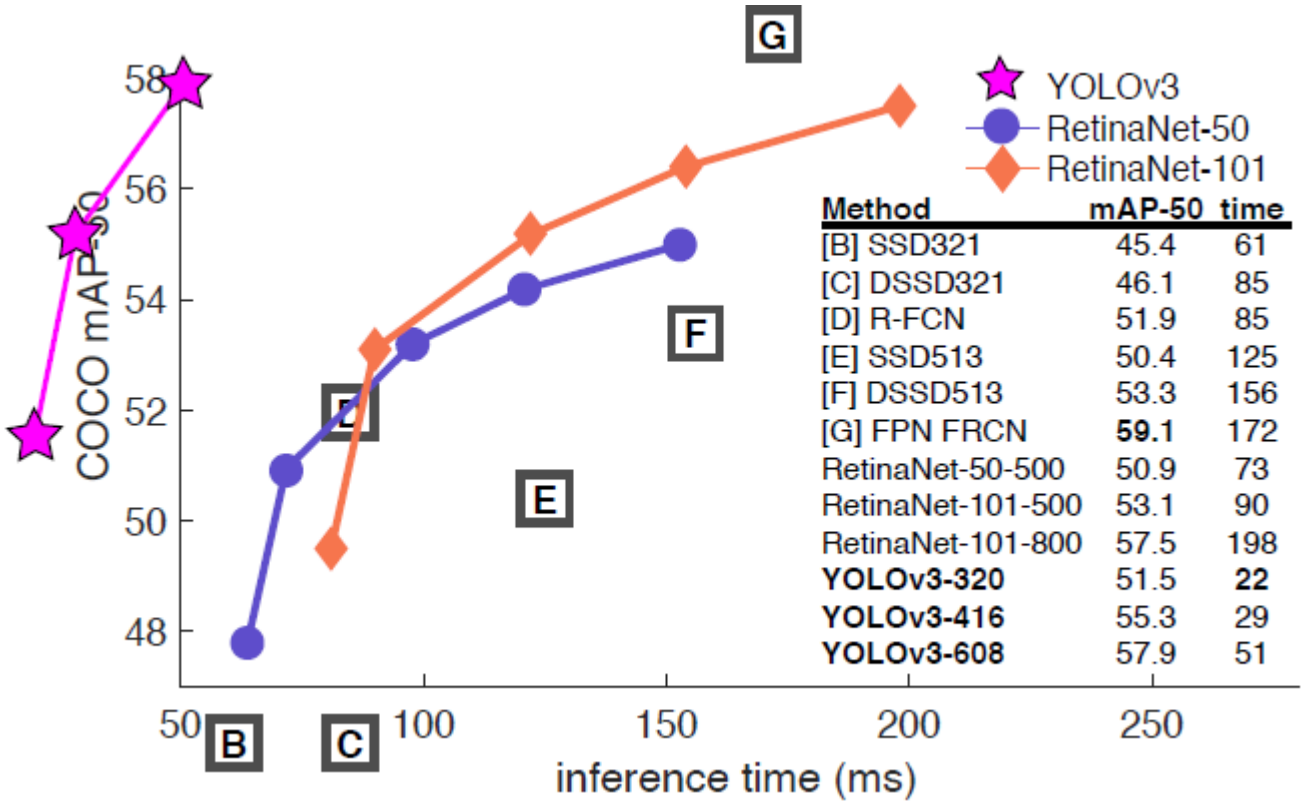
_s2: with stride of 2 **batch_size:** the output size of this layer/block

Residual Block: repeated convolutional layers with ResNet structure



Residual structure
Successive 3x3 and 1x1 + shortcut

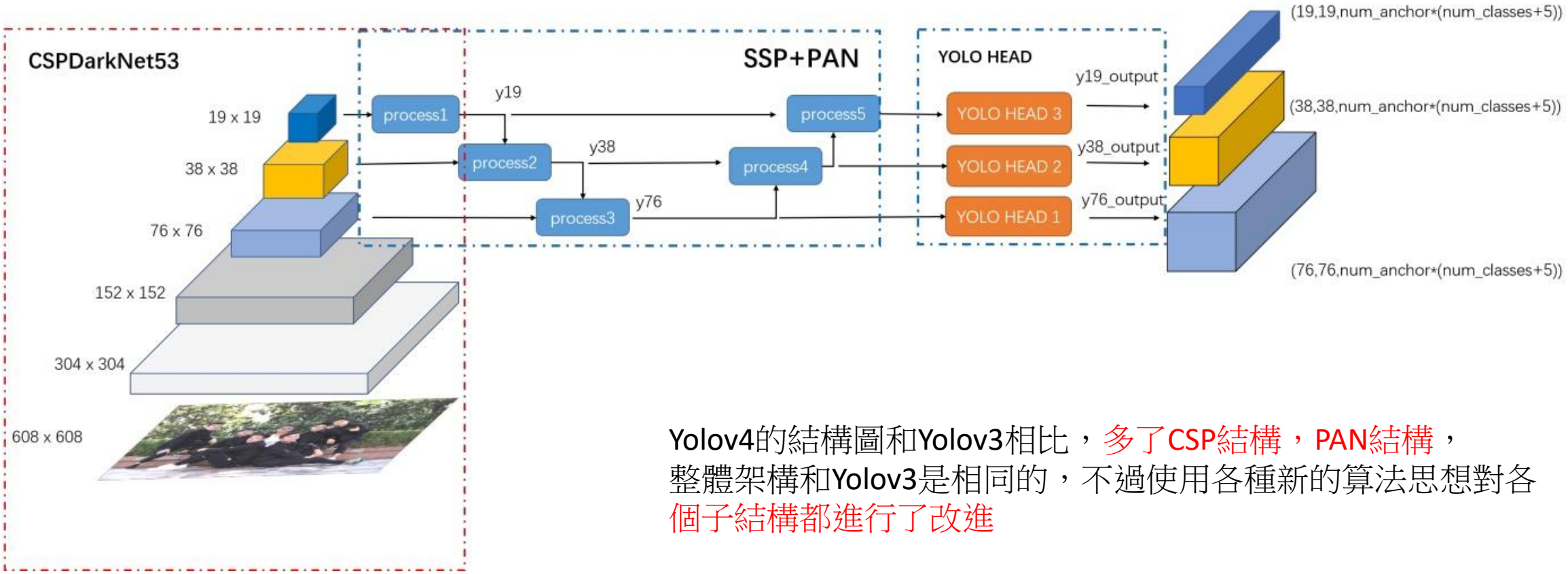
YOLO v3



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Joseph Redmon & Ali Farhadi. "YOLOv3: An Incremental Improvement." CoRR abs/1804.02767 (2018)

YOLO v4



Yolov4的結構圖和Yolov3相比，多了CSP結構，PAN結構，
整體架構和Yolov3是相同的，不過使用各種新的算法思想對各個子結構都進行了改進

YOLOv4: Optimal Speed and Accuracy of Object Detection

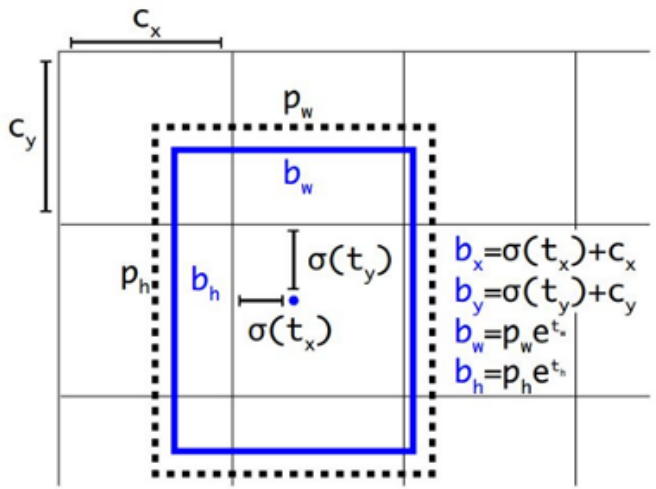
- Yolov4的結構圖和Yolov3相比，
  - 因為多了**CSP結構，PAN結構**，
    - 如果單純看可視化流程圖，會覺得很繞，不過在繪製出上面的圖形後，會覺得豁然開朗，其實整體架構和Yolov3是相同的，不過使用各種新的算法思想對各個子結構都進行了改進。
- **Yolov4的五個基本組件：**
  - **CBM：** Yolov4網絡結構中的最小組件，由Conv+Bn+Mish激活函數三者組成。
  - **CBL：**由Conv+Bn+Leaky_relu激活函數三者組成。
  - **Res unit：**借鑒Resnet網絡中的殘差結構，讓網絡可以構建的更深。
  - **CSPX：**借鑒CSPNet網絡結構，由三個卷積層和X個Res unint模塊Concate組成。
  - **SPP：**採用1×1，5×5，9×9，13×13的最大池化的方式，進行多尺度融合。

YOLOv4: head

- 檢測頭
  - 總的來說還是多尺度的，3個尺度，分別負責大中小目標。只不過多了一些細節的改進
  - **Using multi-anchors for single ground truth**
    - 之前的YOLO v3是1個anchor負責一個GT，YOLO v4中用多個anchor去負責一個GT。方法是：對於GT來說，只要IoU(anchor, GT) > threshold，就讓anchor去負責GT。
    - 這就相當於你anchor框的數量沒變，但是選擇的正樣本的比例增加了，就緩解了正負樣本不均衡的問題。
  - Eliminate grid sensitivity

Bx 沒辦法到grid 邊界，因為sigmoid (0, 1)
改為乘上一個大於一的數字解決

$$b_x = 1.1 \cdot \sigma(t_x) + c_x$$



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

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

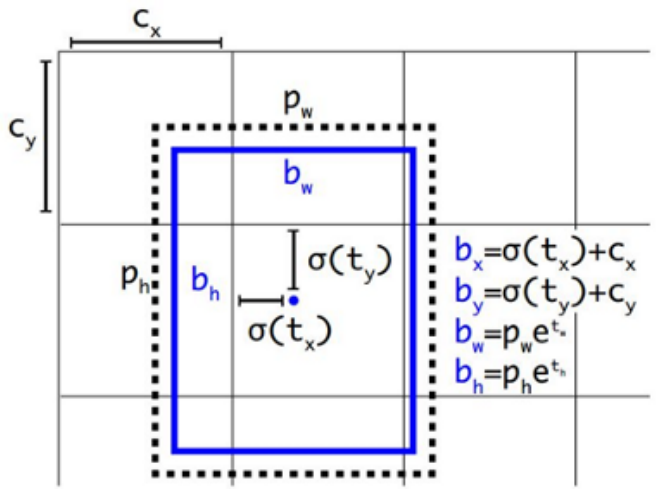
- Yolov4的結構圖和Yolov3相比，
  - 因為多了**CSP結構，PAN結構**，
    - 如果單純看可視化流程圖，會覺得很繞，不過在繪製出上面的圖形後，會覺得豁然開朗，其實整體架構和Yolov3是相同的，不過使用各種新的算法思想對各個子結構都進行了改進。
- **Yolov4的五個基本組件：**
  - **CBM：** Yolov4網絡結構中的最小組件，由Conv+Bn+Mish激活函數三者組成。
  - **CBL：**由Conv+Bn+Leaky_relu激活函數三者組成。
  - **Res unit：**借鑒Resnet網絡中的殘差結構，讓網絡可以構建的更深。
  - **CSPX：**借鑒CSPNet網絡結構，由三個卷積層和X個Res unint模塊Concate組成。
  - **SPP：**採用1×1，5×5，9×9，13×13的最大池化的方式，進行多尺度融合。

YOLOv4: head

- 檢測頭
  - 總的來說還是多尺度的，3個尺度，分別負責大中小目標。只不過多了一些細節的改進
  - **Using multi-anchors for single ground truth**
    - 之前的YOLO v3是1個anchor負責一個GT，YOLO v4中用多個anchor去負責一個GT。方法是：對於GT來說，只要IoU(anchor, GT) > threshold，就讓anchor去負責GT。
    - 這就相當於你anchor框的數量沒變，但是選擇的正樣本的比例增加了，就緩解了正負樣本不均衡的問題。
  - Eliminate grid sensitivity

Bx 沒辦法到grid 邊界，因為sigmoid (0, 1)
改為乘上一個大於一的數字解決

$$b_x = 1.1 \cdot \sigma(t_x) + c_x$$



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

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

YOLO v4: Input, Mosaic Data Augmentation

Figure 3: Mosaic represents a new method of data augmentation.

- Mosaic
  - 參考CutMix，但採用了4張圖片，隨機縮放、隨機裁剪、隨機排布
- 優點
  - 豐富數據集：隨機使用4張圖片，隨機縮放，再隨機分佈進行拼接，大大豐富了檢測數據集，特別是隨機縮放增加了很多小目標，讓網絡的魯棒性更好。
  - 減少GPU：可能會有人說，隨機縮放，普通的數據增強也可以做，但作者考慮到很多人可能只有一個GPU，因此Mosaic增強訓練時，可以直接計算4張圖片的數據，使得Mini-batch大小並不需要很大，一個GPU就可以達到比較好的效果。

小目標的定義是目標框的長寬0×0\~32×32之間的物體。但在整體的數據集中，小、中、大目標的佔比並不均衡。
如上表所示，Coco數據集中小目標佔比達到41.4\%，數量比中目標和大目標都要多。
但在所有的訓練集圖片中，只有52.3\%的圖片有小目標，而中目標和大目標的分佈相對來說更加均勻一些。

YOLO v4: backbone- Cross Stage Partial Network

- CSPNet

Table 1: Parameters of neural networks for image classification.

Backbone model	Input network resolution	Receptive field size	Parameters	Average size of layer output (WxHxC)	BFLOPs (512x512 network resolution)	FPS (GPU RTX 2070)
CSPResNext50	512x512	425x425	20.6 M	**1058 K**	31 (15.5 FMA)	62
CSPDarknet53	512x512	725x725	**27.6 M**	950 K	52 (26.0 FMA)	**66**
EfficientNet-B3 (ours)	512x512	**1311x1311**	12.0 M	668 K	11 (5.5 FMA)	26

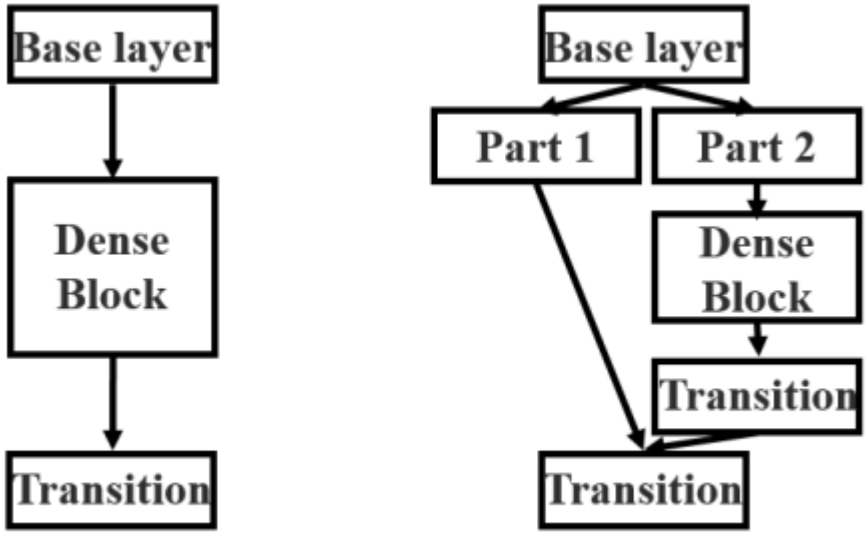
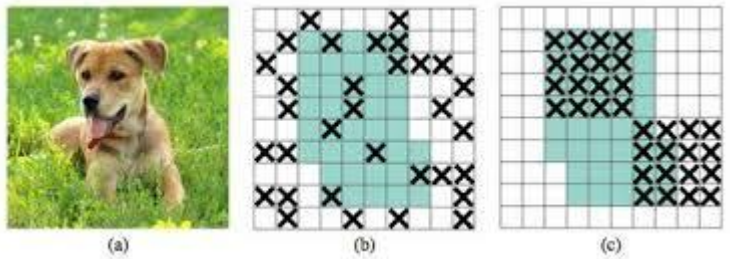


Table 3: Influence of BoF and Mish on the CSPDarknet-53 classifier accuracy.

MixUp	CutMix	Mosaic	Bluring	Label Smoothing	Swish	Mish	Top-1	Top-5
							77.2%	93.6%
	✓	✓		✓			**77.8%**	**94.4%**
	✓	✓		✓		✓	**78.7%**	**94.8%**

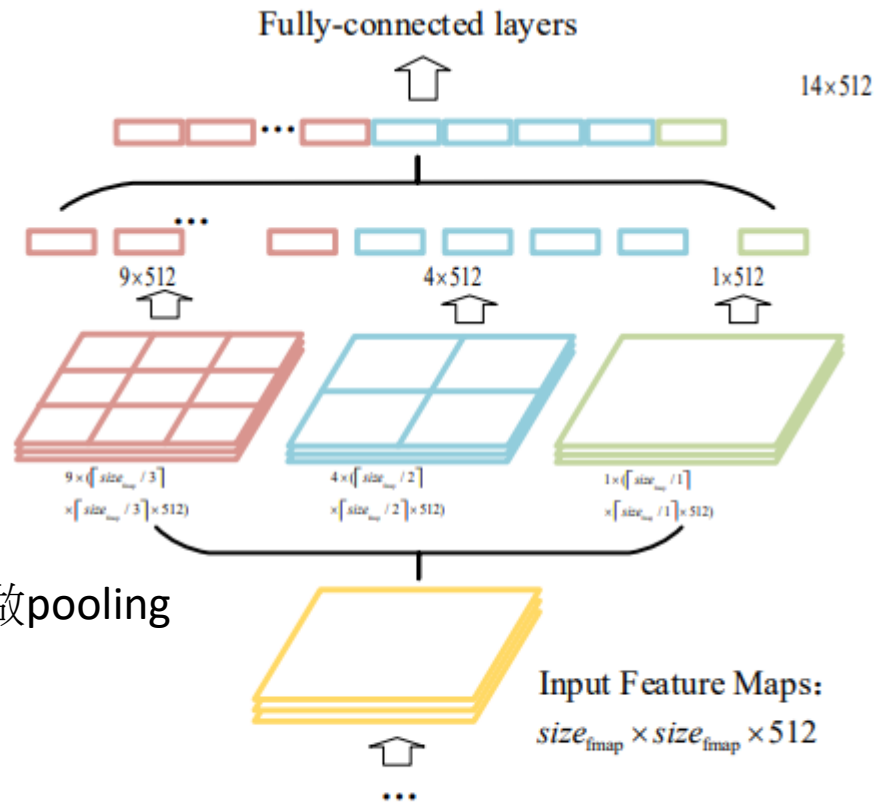


dropblock

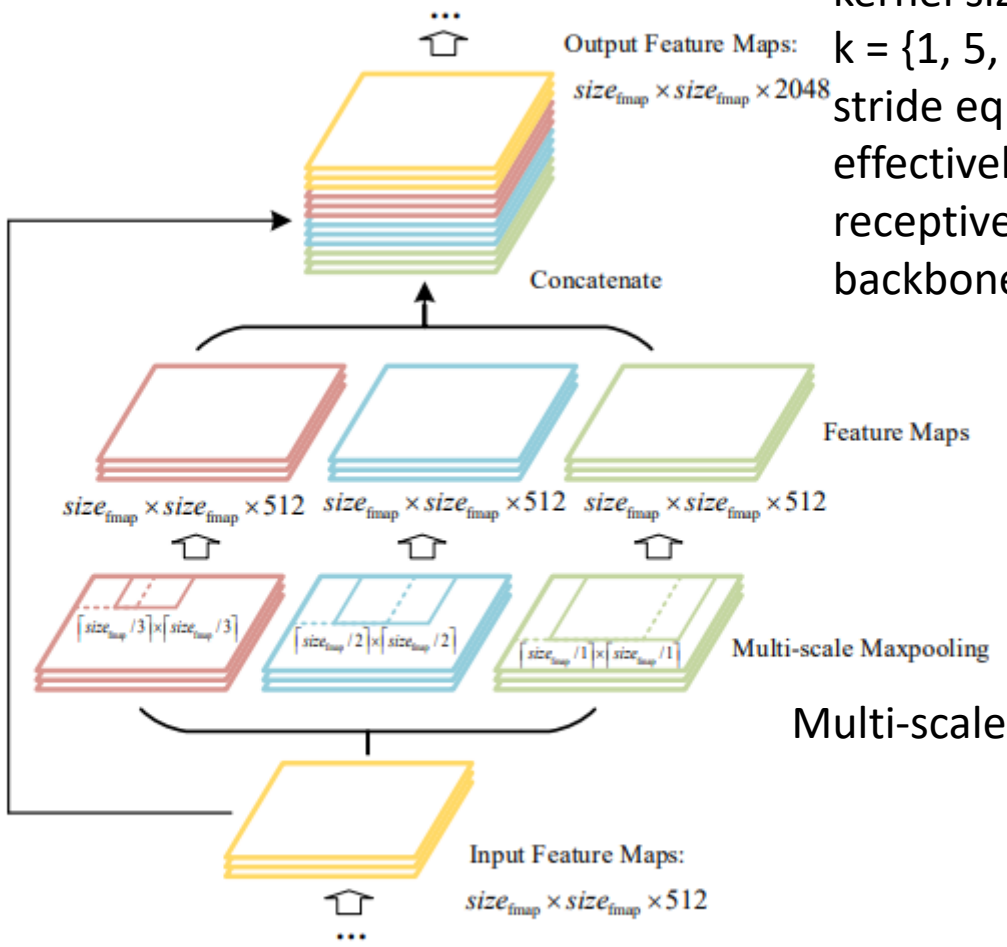
CSPNet: A New Backbone that can Enhance Learning Capability of CNN

YOLO v4: Neck- SPP

concatenation of max-pooling outputs with kernel size k × k, where k = {1, 5, 9, 13}, and stride equals to 1. effectively increase the receptive field of backbone feature



直接做pooling



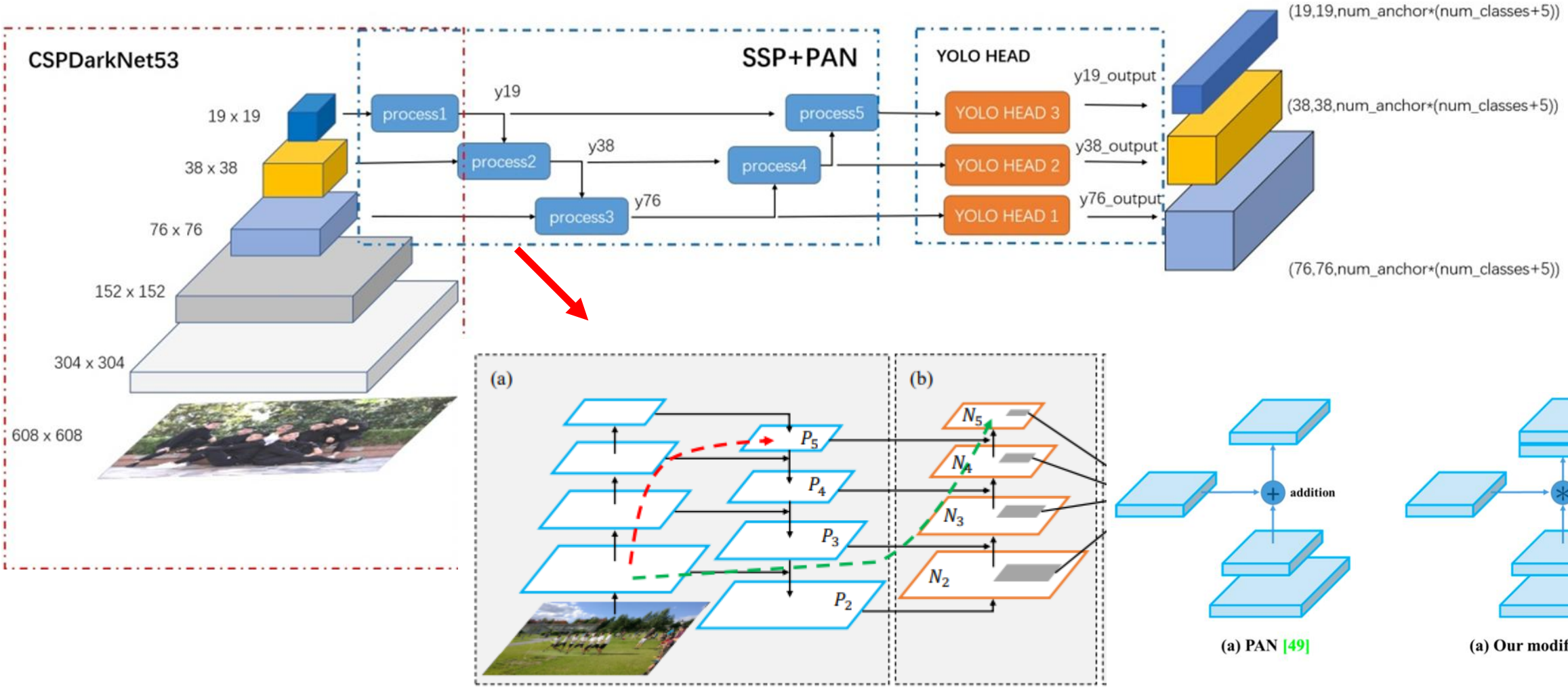
Multi-scale pooling

(a) (b)

The Spatial Pyramid Pooling (a) and the Improved Spatial Pyramid Pooling (b)

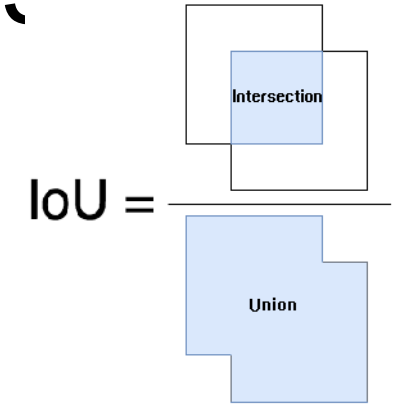
DC-SPP-YOLO: Dense Connection and Spatial Pyramid Pooling Based YOLO for Object Detection

YOLO v4: Neck - PAN



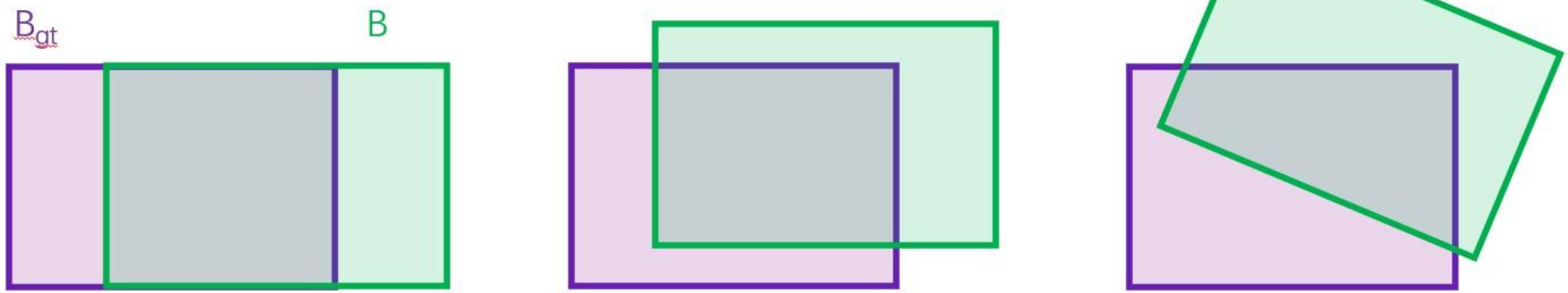
Path Aggregation Network for Instance Segmentation

YOLO v4: Loss Function – CIoU Loss



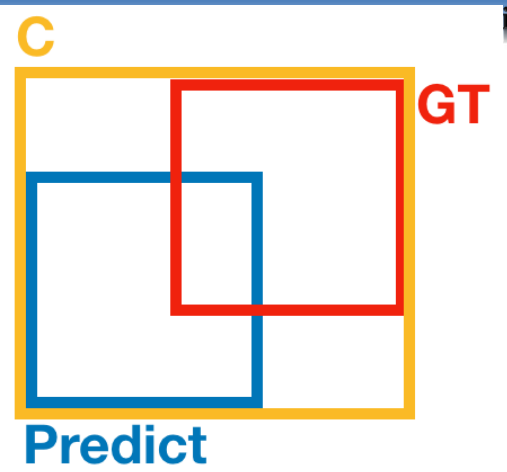
- **CIoU-loss for BOX regression**
  - MSE Loss ->IoU Loss ->GIoU Loss ->DIoU Loss ->CIoU Loss
- IoU Loss
  - 反映預測檢測框與真實檢測框的檢測效果
  - 問題也很多：不能反映兩者的距離大小（重合度）。IoU無法辨別不同方式的對齊，比如方向不一致等。同時因為loss=1(沒有重疊)，沒有梯度回傳，無法進行學習訓練

$$L_{IoU} = 1 - \frac{|B \cap B_{gt}|}{|B \cup B_{gt}|}$$



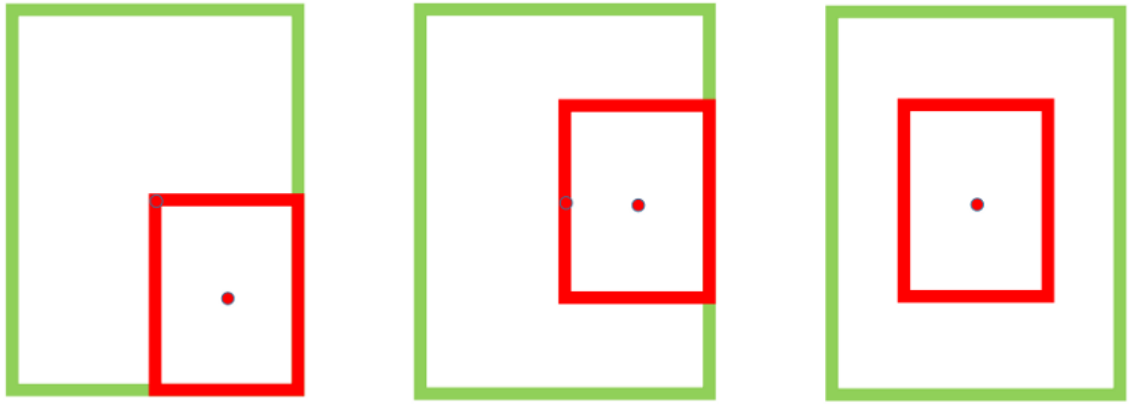
IoU Loss不能反映兩者的距離大小

$$L_{GIoU} = 1 - IoU + \frac{|C - B \cup B_{gt}|}{|C|}$$



- GIoU loss
  - C 為同時包含了預測框和真實框的最小框的面積
  - IOU-loss只在predict box 與 ground box有重疊的時候才會有gradient, 在沒有重疊的情況下並不會有gradient(因為loss 都是 1), 故G-IOU對於此問題提出了一個懲罰項：
  - GIoU Loss可以解決上面IoU Loss對距離不敏感的問題。但是GIoU Loss存在訓練過程中發散等問題。

GIOU Loss都是一樣



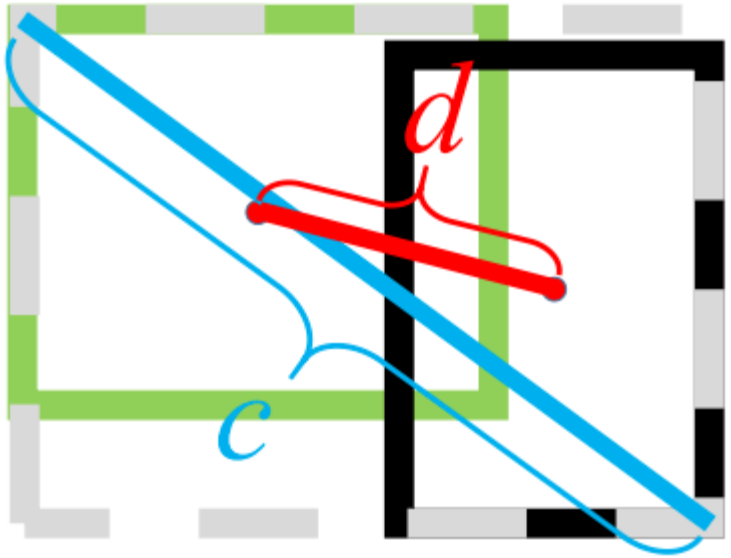
$\mathcal{L}_{IoU} = 0.75$ $\mathcal{L}_{GIoU} = 0.75$ $\mathcal{L}_{DIoU} = 0.81$

$\mathcal{L}_{IoU} = 0.75$ $\mathcal{L}_{GIoU} = 0.75$ $\mathcal{L}_{DIoU} = 0.77$

$\mathcal{L}_{IoU} = 0.75$ $\mathcal{L}_{GIoU} = 0.75$ $\mathcal{L}_{DIoU} = 0.75$

$$L_{DIoU} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2}$$

- DIoU loss
  - 其中，b，bgt分別代表了預測框和真實框的中心點，且ρ代表的是計算兩個中心點間的歐式距離。c代表的是能夠同時包含預測框和真實框的最小閉包區域的對角線距離。
  - 提出懲罰項如下來最小化predict box 與 target box 正規化後的距離



$$\mathcal{R}_{DIoU} = \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2},$$

分子的部分為predict box 與 ground truth box 的l2-distance, c 為包含predict box 與 ground truth box最小Box的Diagonal length.
DIOU則是直接最小化中心點距離.

- CIoU Loss: **Complete-IoU Loss**
  - Box regression的三大核心(overlap area, central point distance, aspect ratio)
  - GIoU為了歸一化坐標尺度，利用IOU並初步解決了IoU為0無法優化的問題。
  - DIoU損失在GIoU Loss的基礎上考慮了邊界框的重疊面積和中心點距離。
  - Anchor的長寬比和目標框之間的長寬比的一致性。=>CIoU Loss。

$$\mathcal{L}_{CIoU} = 1 - IoU + \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} + \alpha v.$$

$$v = \frac{4}{\pi^2}(arctan\frac{w^{gt}}{h^{gt}} - arctan\frac{w}{h})^2.$$ 衡量consistency aspect ratio

$$\alpha = \frac{v}{(1 - IoU) + v},$$

Distance-IOU Loss : Faster and Better Learning for Box Regression

Table 1: Quantitative comparison of **YOLOv3** (Redmon and Farhadi 2018) trained using $\mathcal{L}_{IoU}$ (baseline), $\mathcal{L}_{GIoU}$, $\mathcal{L}_{DIoU}$ and $\mathcal{L}_{CIoU}$. (D) denotes using DIoU-NMS. The results are reported on the test set of PASCAL VOC 2007.

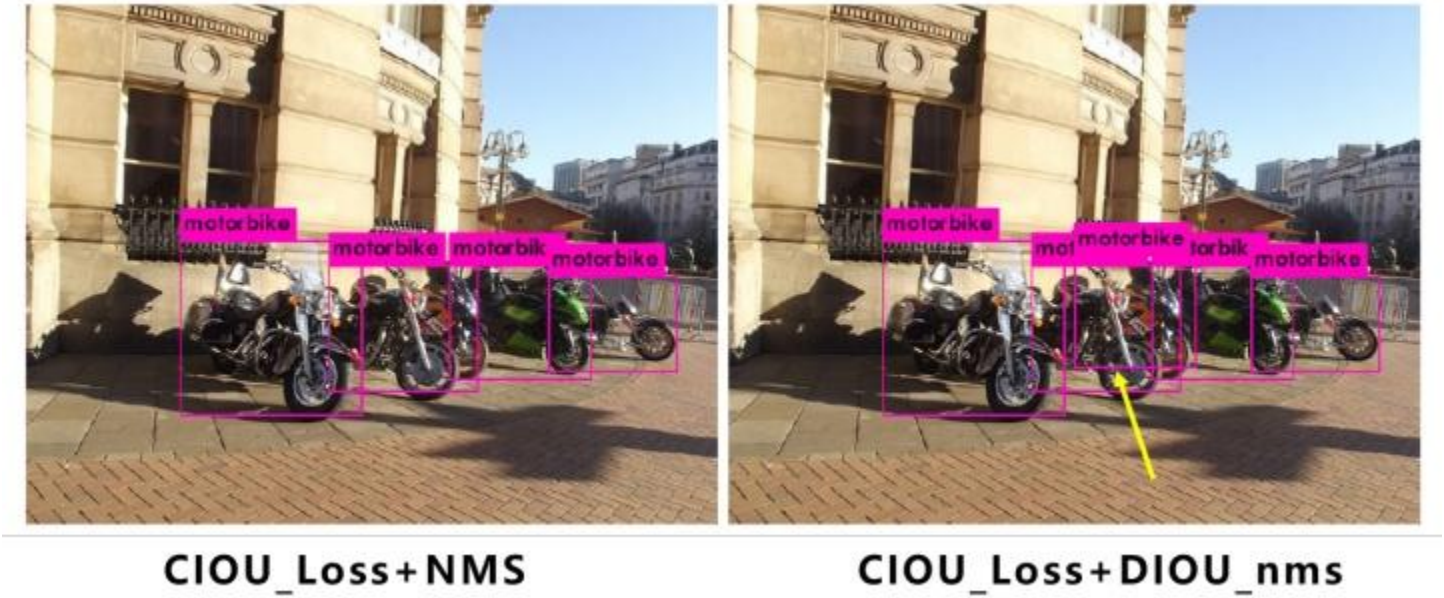
Loss / Evaluation	AP		AP75	
	IoU	**GIoU**	**IoU**	**GIoU**
$\mathcal{L}_{IoU}$	46.57	45.82	49.82	48.76
$\mathcal{L}_{GIoU}$	47.73	46.88	52.20	51.05
Relative improv. %	2.49%	2.31%	4.78%	4.70%
$\mathcal{L}_{DIoU}$	48.10	47.38	52.82	51.88
Relative improv. %	3.29%	3.40%	6.02%	6.40%
$\mathcal{L}_{CIoU}$	**49.21**	**48.42**	**54.28**	**52.87**
Relative improv. %	**5.67%**	**5.67%**	**8.95%**	**8.43%**
$\mathcal{L}_{CIoU}(D)$	**49.32**	**48.54**	**54.74**	**53.30**
Relative improv. %	**5.91%**	**5.94%**	**9.88%**	**9.31%**

Table 3: Quantitative comparison of **Faster R-CNN** (Ren et al. 2015) trained using $\mathcal{L}_{IoU}$ (baseline), $\mathcal{L}_{GIoU}$, $\mathcal{L}_{DIoU}$ and $\mathcal{L}_{CIoU}$. (D) denotes using DIoU-NMS. The results are reported on the validation set of MS COCO 2017.

Loss / Evaluation	AP	AP75	APsmall	APmedium	APlarge
$\mathcal{L}_{IoU}$	37.93	40.79	21.58	40.82	50.14
$\mathcal{L}_{GIoU}$	38.02	41.11	21.45	41.06	50.21
Relative improv. %	0.24%	0.78%	-0.60%	0.59%	0.14%
$\mathcal{L}_{DIoU}$	38.09	41.11	**21.66**	41.18	50.32
Relative improv. %	0.42%	0.78%	**0.31%**	0.88%	0.36%
$\mathcal{L}_{CIoU}$	**38.65**	**41.96**	21.32	**41.83**	**51.51**
Relative improv. %	**1.90%**	**2.87%**	-1.20%	**2.47%**	**2.73%**
$\mathcal{L}_{CIoU}(D)$	**38.71**	**42.07**	21.37	**41.93**	**51.60**
Relative improv. %	**2.06%**	**3.14%**	-0.97%	**2.72%**	**2.91%**

YOLO v4: DIOU_NMS

- NMS with DIOU loss
  - CIOU loss needs ground truth (i.e. for training only)
  - CIOU loss – penalty => DIOU loss



YOLOv4

- We assume that such universal features include
  - Weighted-Residual-Connections (WRC),
  - Cross-Stage-Partial-connections (CSP),
  - Cross mini-Batch Normalization (CmBN),
  - Self-adversarial-training (SAT)
  - Mish-activation.
  - Mosaic data augmentation,
  - DropBlock regularization,
  - CIoU loss,
  - combine some of them to achieve state-of-the-art results: 43.5% AP (65.7% AP50) for the MS COCO dataset at a realtime speed of ~65 FPS on Tesla V100

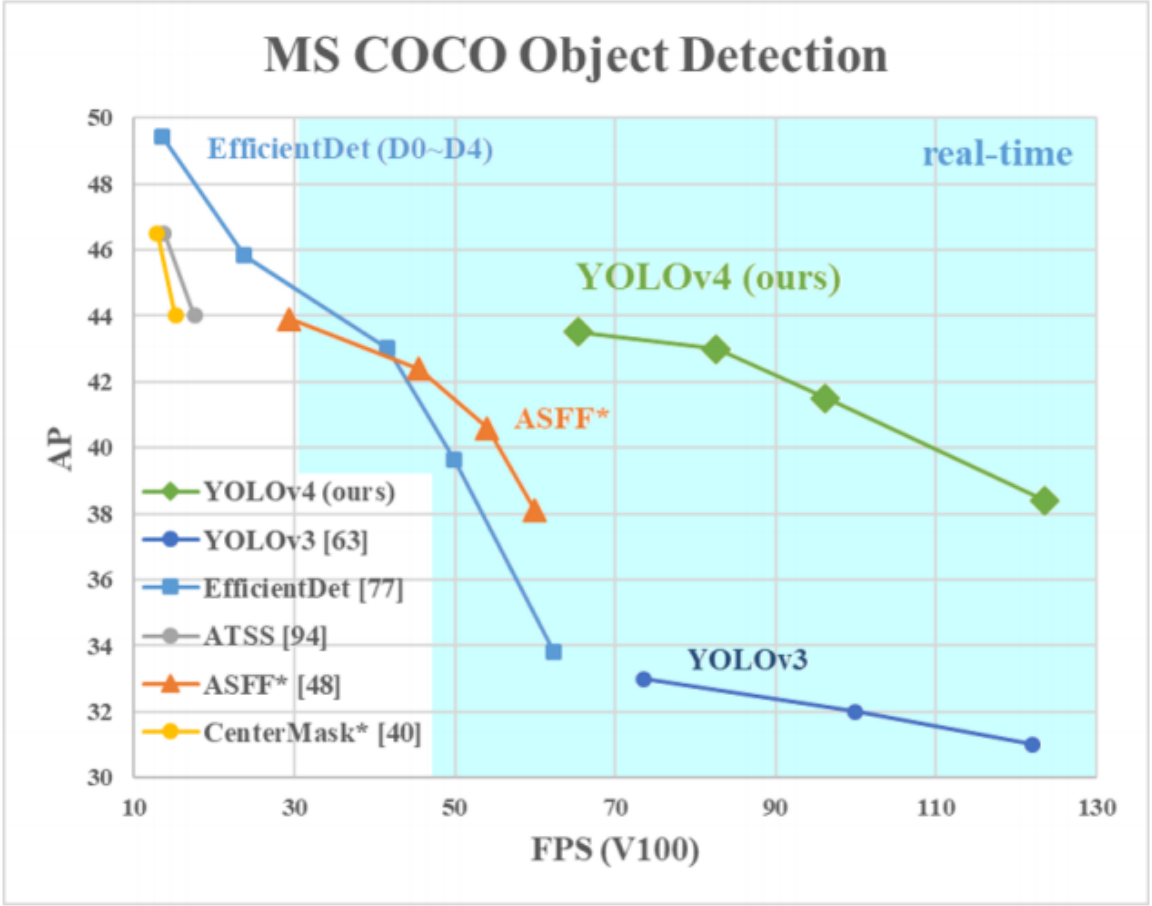
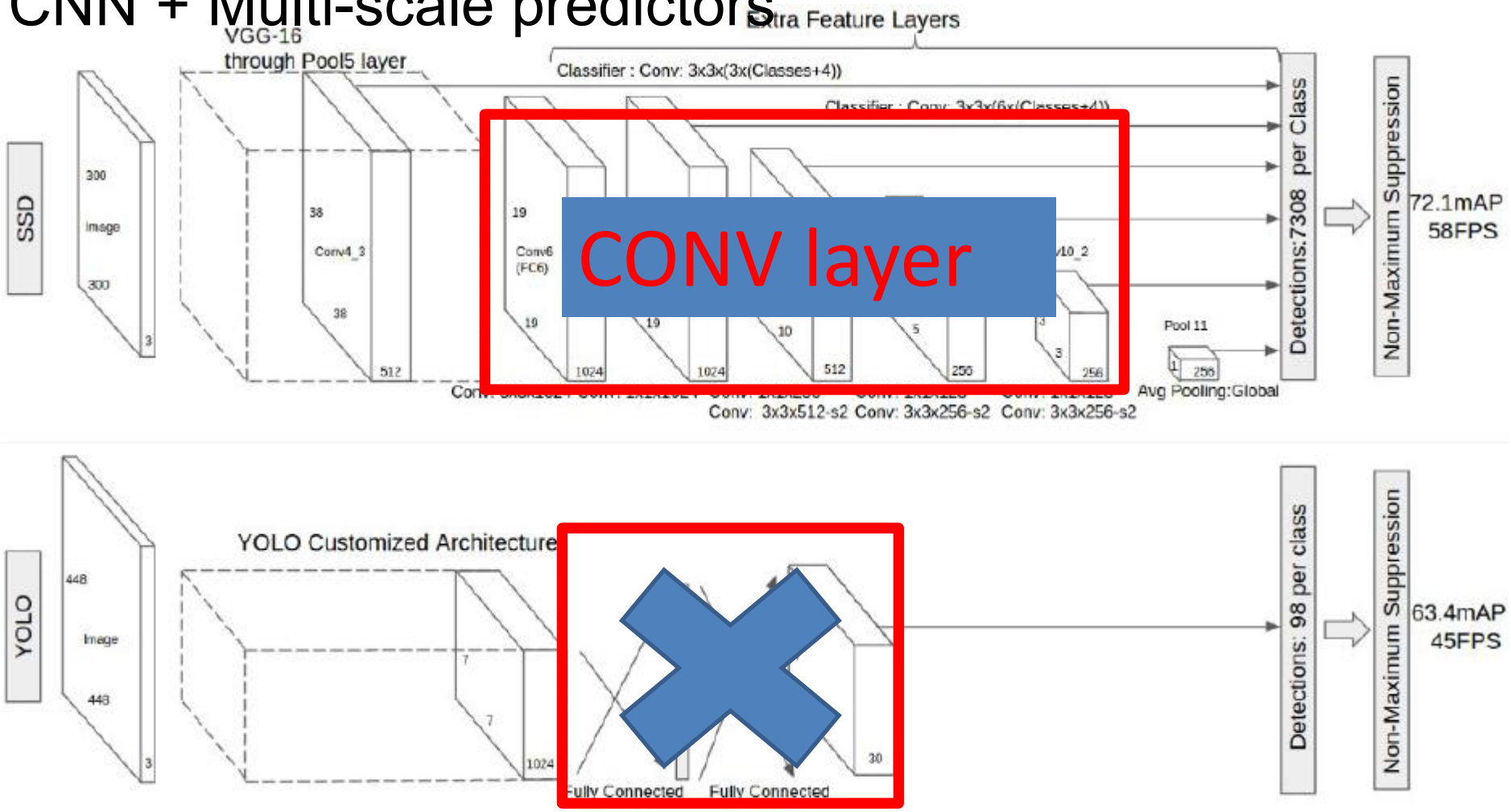


Figure 1: Comparison of the proposed YOLOv4 and other state-of-the-art object detectors. YOLOv4 runs twice faster than EfficientDet with comparable performance. Improves YOLOv3's AP and FPS by 10% and 12%, respectively.

SSD

SSD: Single Shot MultiBox Detector

- CNN + Multi-scale predictors

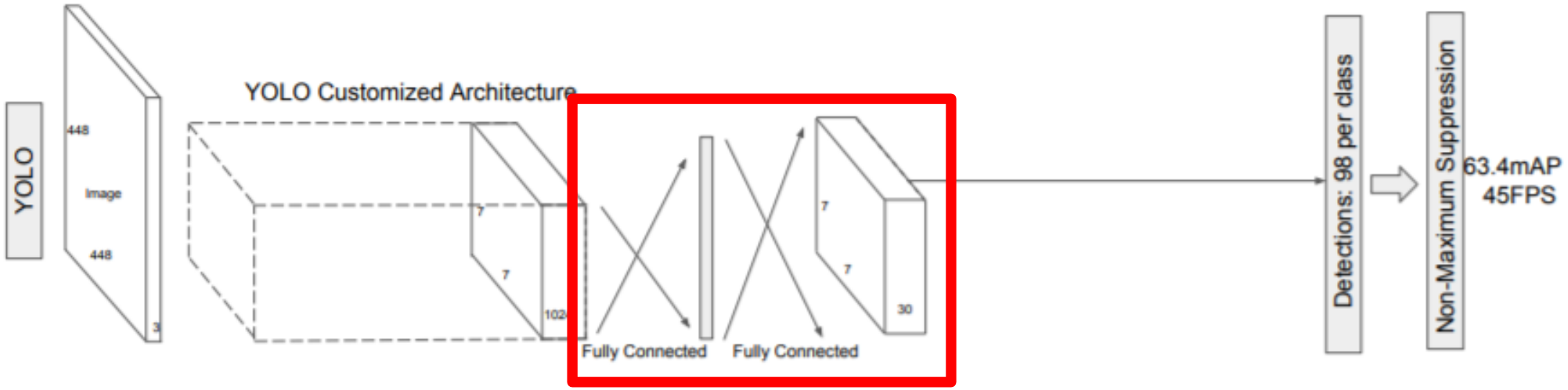


SSD

- Base network +
- Multi-scale feature maps for detection
  - Allow predictions of detections at multiple scales
- Convolutional predictors for detection
  - produce a fixed set of detection predictions using a set of convolutional filters
    - predictions: a score for a category, or a shape offset relative to the default box
- Default boxes and aspect ratios

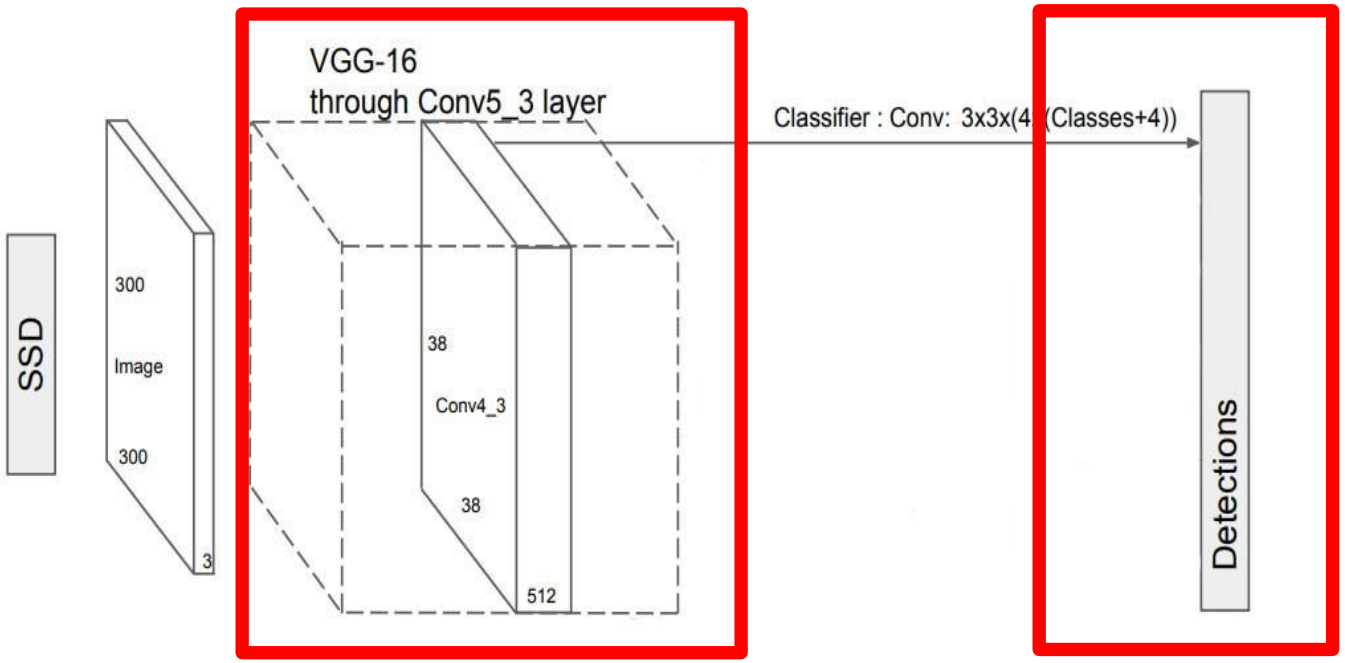
- Training
  - Extensive data argumentation
  - Hard negative mining

SSD: Single Shot MultiBox Detector

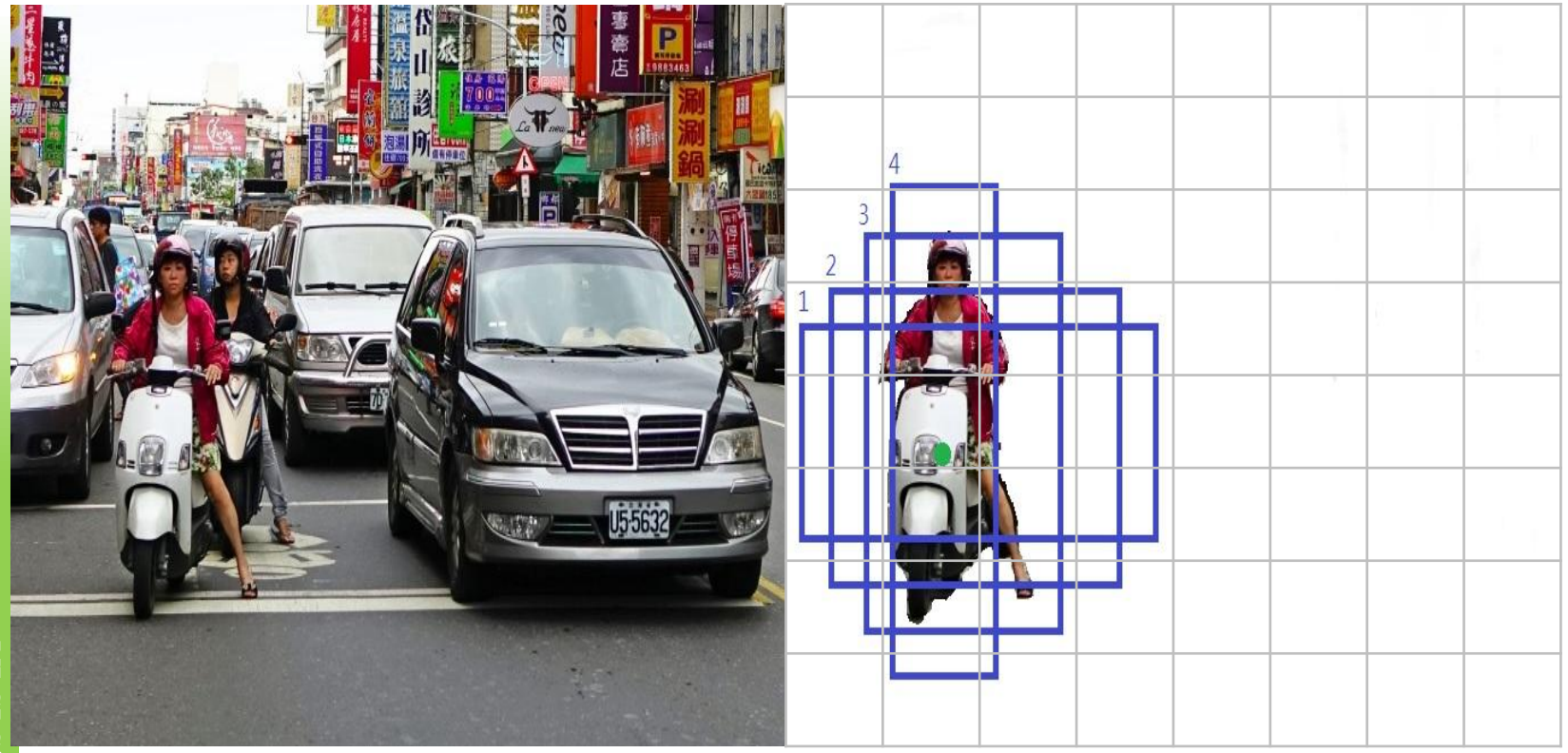


YOLO有一個缺點，就是對小物體的偵測效果不佳
因為YOLO在 7x7 的框架下識別物體，遇到大量小物體時，難以處理

SSD: Single Shot Multibox Detector



這裡的classifier 用3x3 conv filter 而不是用FC層



用類似YOLO的方法，對feature map 分成不同的cell，但是每個cell進行4個不同大小的object預測

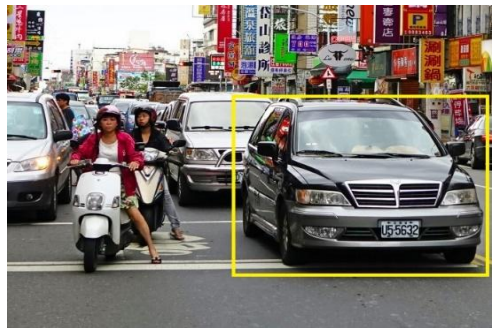
SSD

(Δcx,Δcy,w,h) + person:0.8 scooter:0.6 ... car:0.5

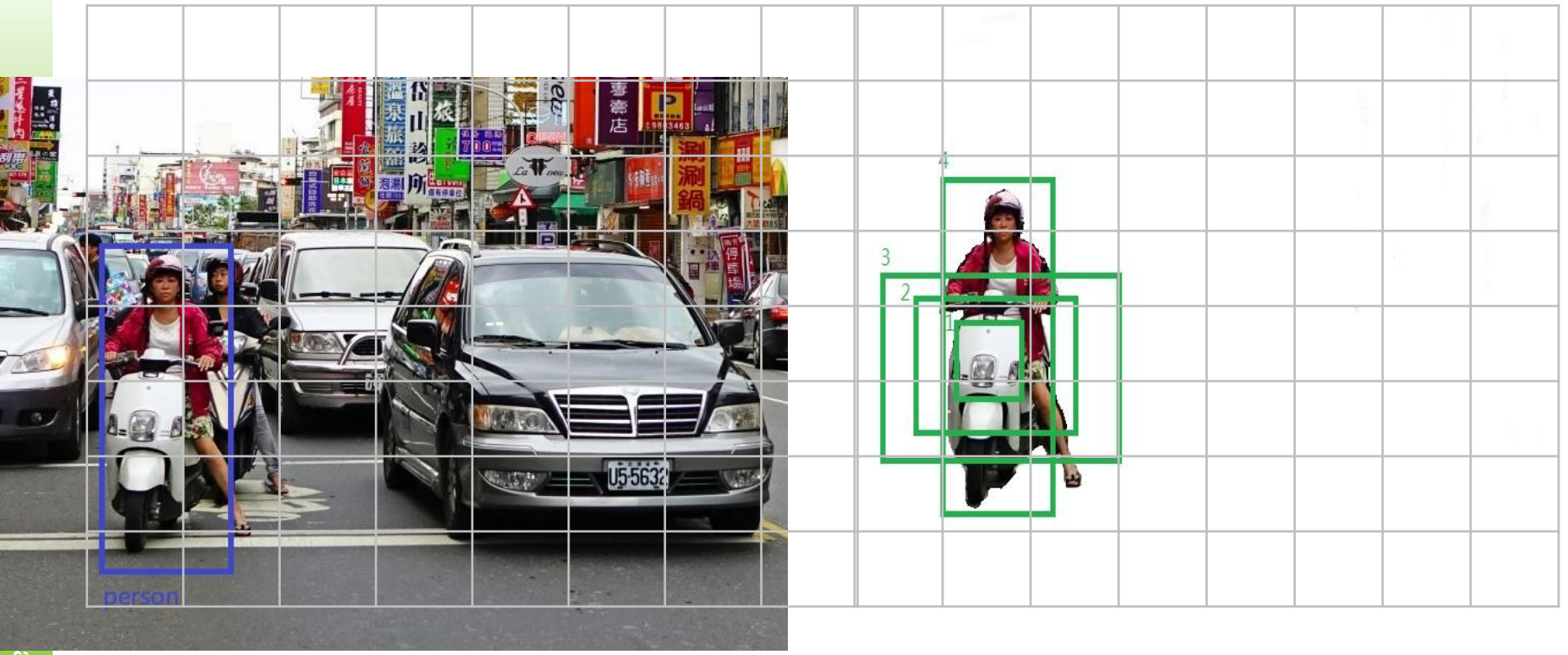
(Δcx,Δcy,w,h) + car:0.7 scooter:0.2 ... person:0.1

• • •

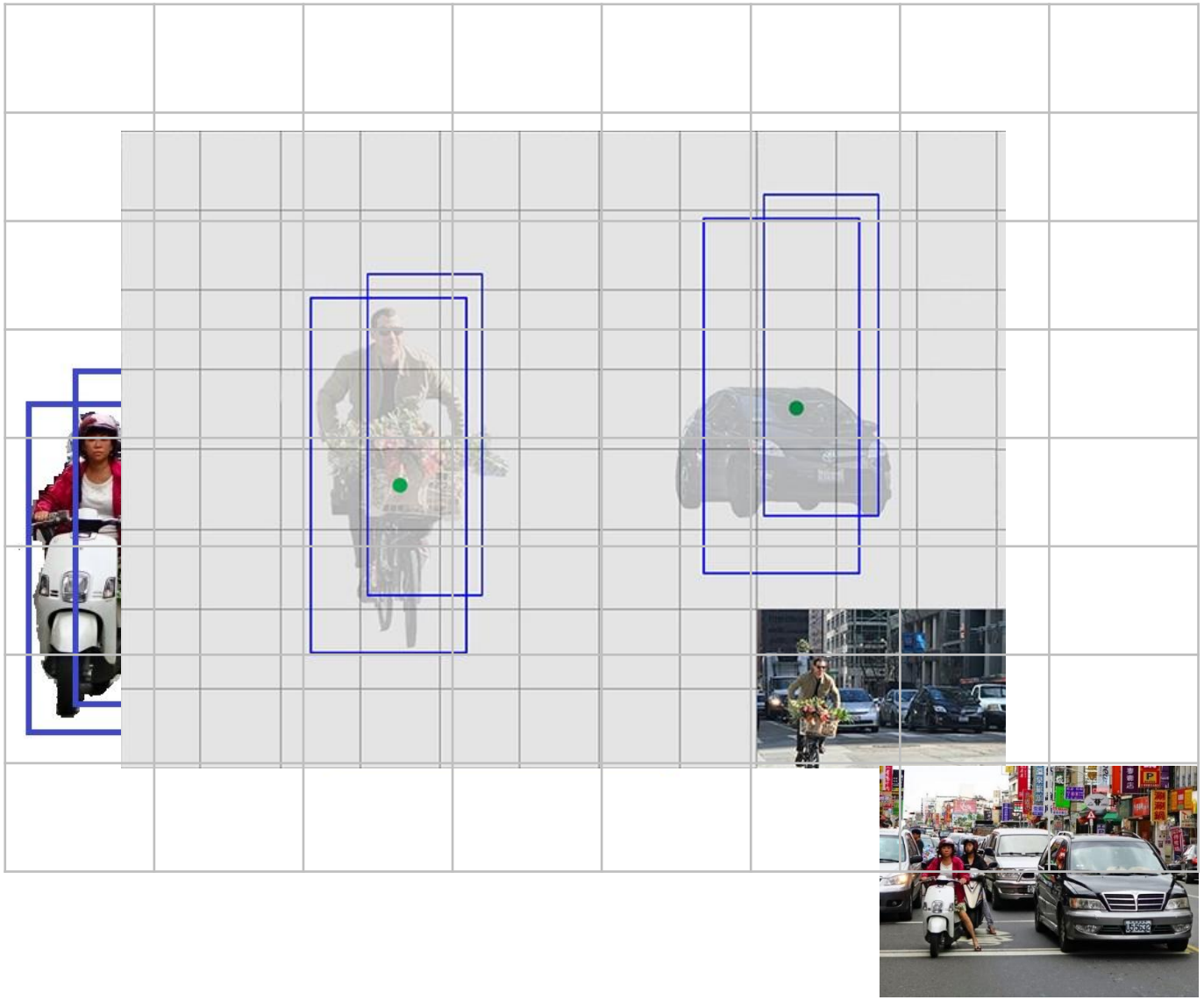
(Δcx,Δcy,w,h)



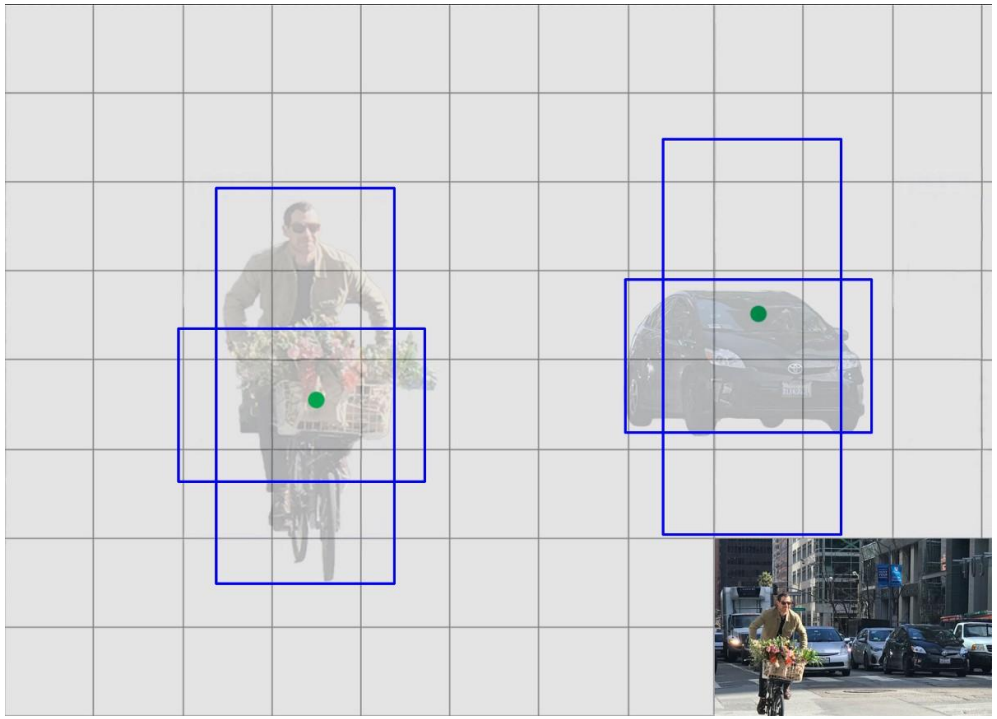
每個預測包括一個bounding box 和21個類的21個分數（一類是背景）(PASCAL VOC dataset)



SSD 採用幾個預先設定好的bounding box 形狀，不同長寬比和大小

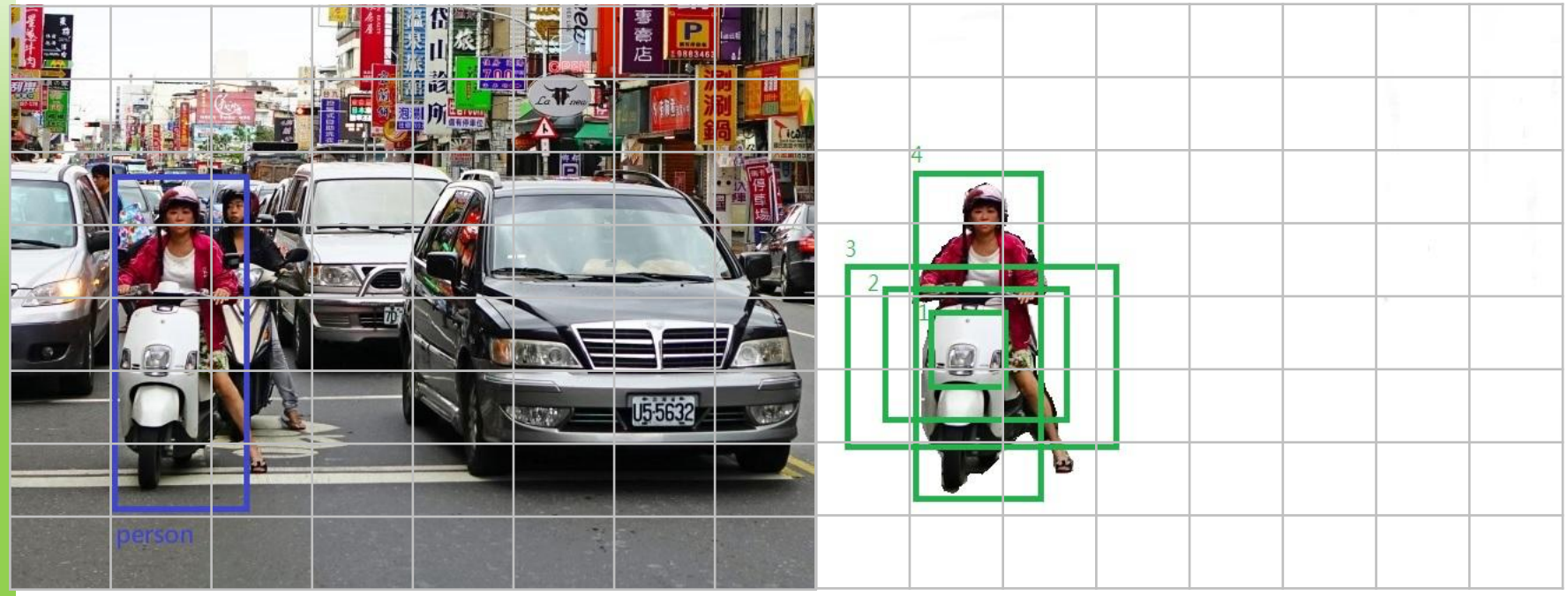


如果讓bounding box從隨機預測開始去訓練，實驗證明，一開始的訓練因為不同形狀的bounding box 互相干擾，訓練
能非常不穩定。而bounding box形狀若是不夠多樣，則模型將無法執行。

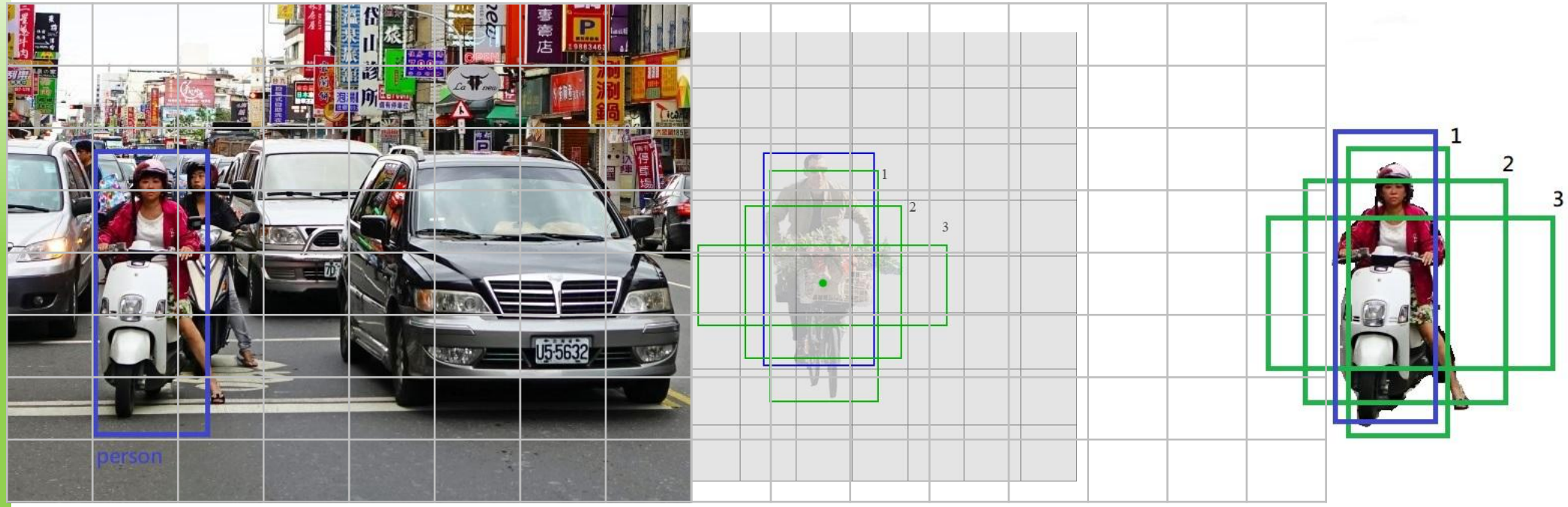


在現實生活中，**bounding box形狀**和大小不是**任意的**。
汽車的形狀相似，行人的長寬比約為0.41。

Default Box

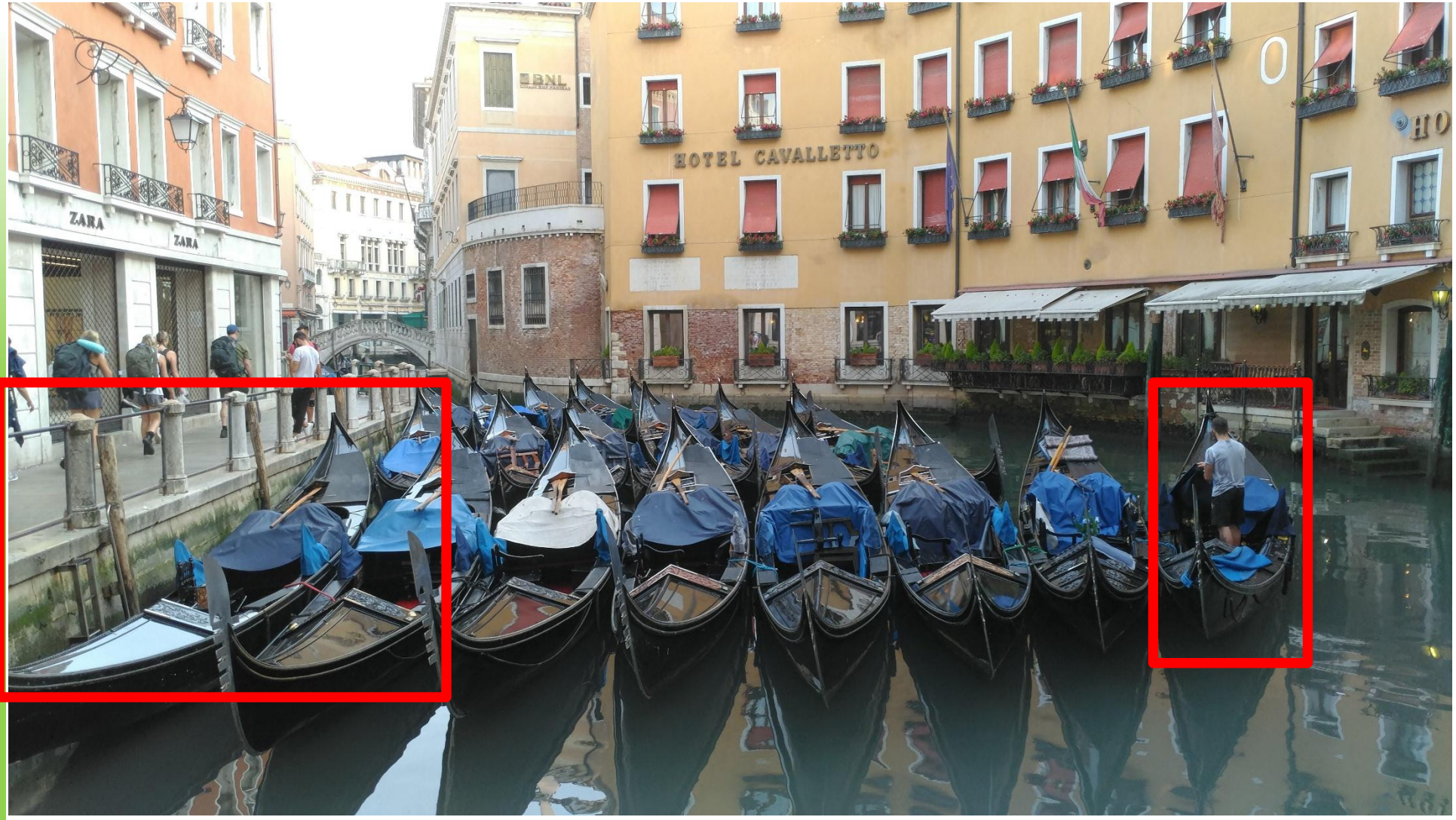


在SSD的default box長寬比有1,2,3,1 / 2和1/3，在搭配不同的大小比例去使用



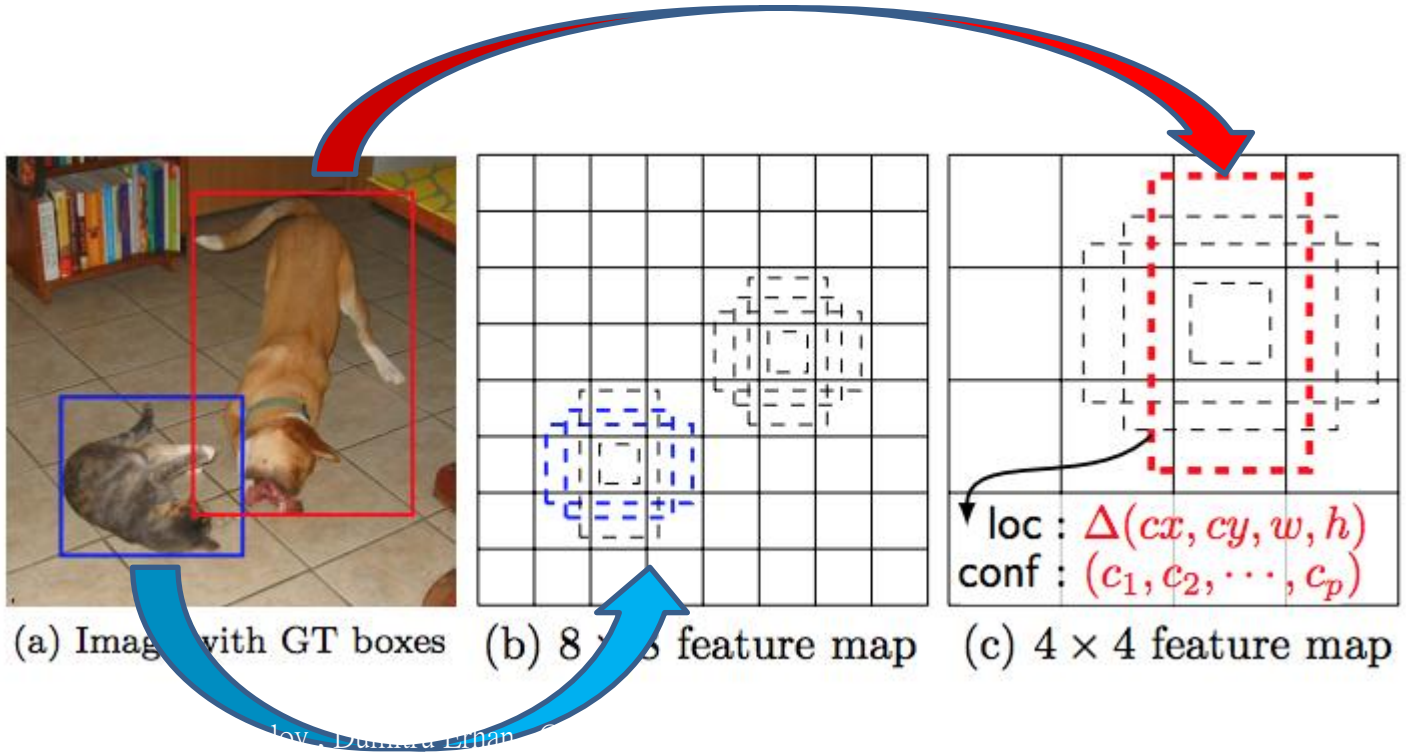
只有和ground truth 重疊超過一半的default box，才被認為有match 到
沒有match 到的，後續就不考慮

MultiScale Feature Map



改變觀看的視角時，原本同樣大小的物體，在畫面上長寬比例就會變

Multi-Scale



同一畫面，可能有大小不同的物體

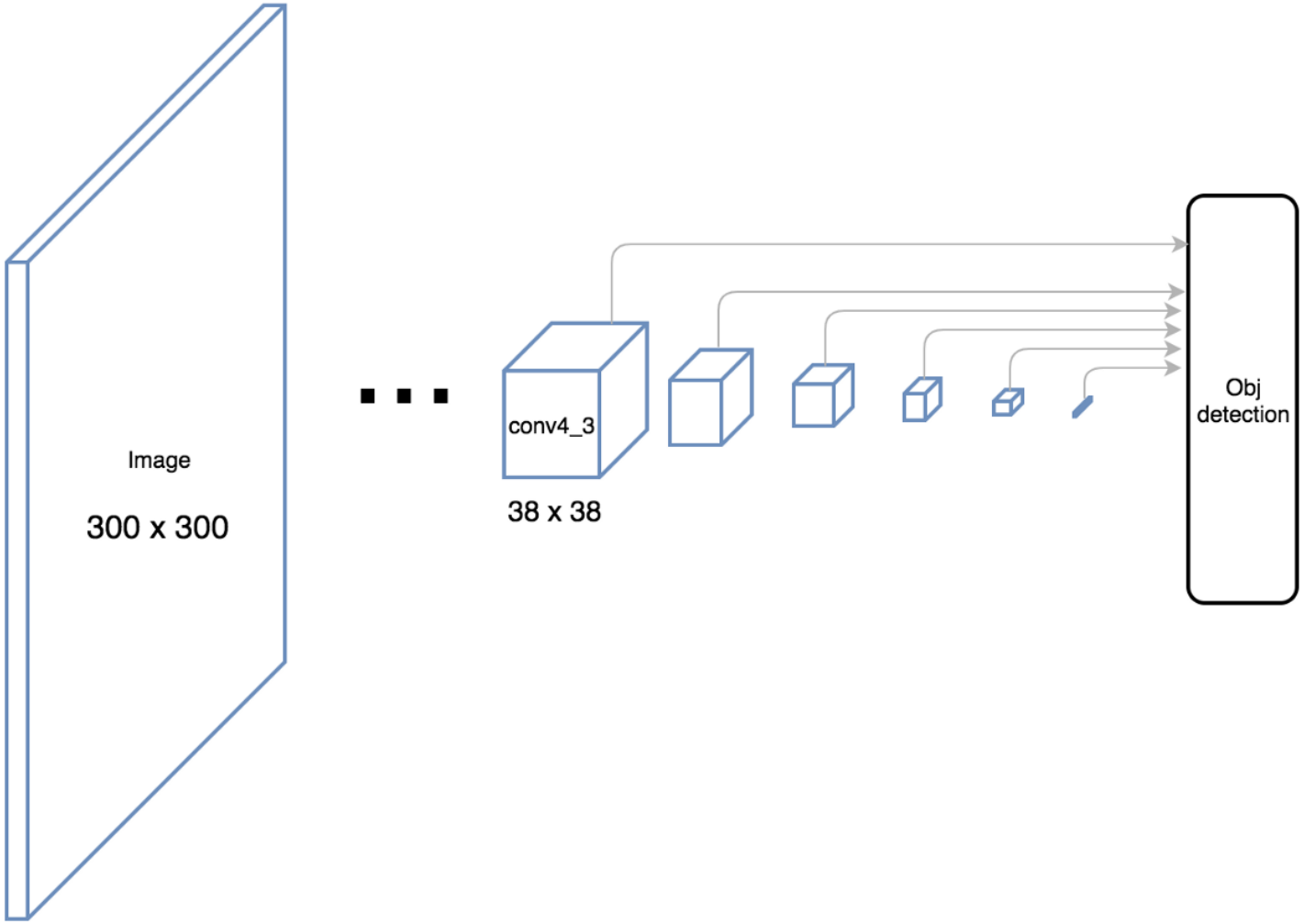


Image
300 x 300
conv4_3
38 x 38
Obj
detection

Method	data	Avg. Precision, IoU: 0.5:0.95	0.5	0.75	Avg. Precision, Area: S	M	L
Fast [6]	train	19.7	35.9	-	-	-	-
Fast [24]	train	20.5	39.9	19.4	4.1	20.0	35.8
Faster [2]	trainval	21.9	42.7	-	-	-	-
ION [24]	train	23.6	43.2	23.6	6.4	24.1	38.3
Faster [25]	trainval	24.2	45.3	23.5	7.7	26.4	37.1
SSD300	trainval35k	23.2	41.2	23.4	5.3	23.2	39.6
SSD512	trainval35k	**26.8**	**46.5**	**27.8**	**9.0**	**28.9**	**41.9**

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

SSD速度比YOLO快，而且比較準，
主要原因就是去除FC層，和multi-scale network 與多種default box 所貢獻形成

Could a simple one-stage detector achieve similar accuracy as two stage detector?

- Reasons of one-stage detector fail: inaccurate locations/features?
- Hypothesis in Focal Loss for Dense Object Detection, RetinaNet. 2017
  - Class imbalance
- How this happens?
  - Cross entropy loss CE(p_t) = -log(p_t)
  - Note total loss =

Σ CE(p_t) = loss of positive examples + loss of negative examples

  - Positive examples (object) has large loss (hard to train), but small number
  - Negative examples (background) (well classified examples) has small loss but large number => easily dominate total cost, and gradient
    - these are most on background,

Class Imbalance: Classic Problem in Object Detection

- Too few positive examples, and too many negative examples
  - Few real object boxes and many background boxes (10^4 ~10^5 for SSD)

Solution in two stage object detection

- Proposal stages
  - Limited object candidates (1~2K by selective search)
    - to filter out most background samples
- Classification stages
  - sampling heuristics
    - (fixed foreground-to-background ratio (1:3), or online hard example mining (OHEM) , bootstrapping, hard example mining) to maintain a manageable balance between foreground and background
  - inefficient as the training procedure is still dominated by easily classified background examples

RetinaNet: Focal Loss

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

- 解決方案也很直接
  - 直接按照loss decay掉那些easy example的權重，這樣使訓練更加bias到更有意義的樣本中去
  - Class with larger Pt

  ⇒easy samples

  ⇒Lower weighting to the loss

  ⇒Smaller loss

- Cross entropy CE(p_t) = -log(p_t)
- Focal loss FL(p_t) = -(1-p_t)$^\gamma$ log(p_t)
- γ =0, Focal loss == cross entropy loss
- α balanced version

$$\mathrm{FL}(p_t)=-\alpha_t(1-p_t)^\gamma\log(p_t).$$

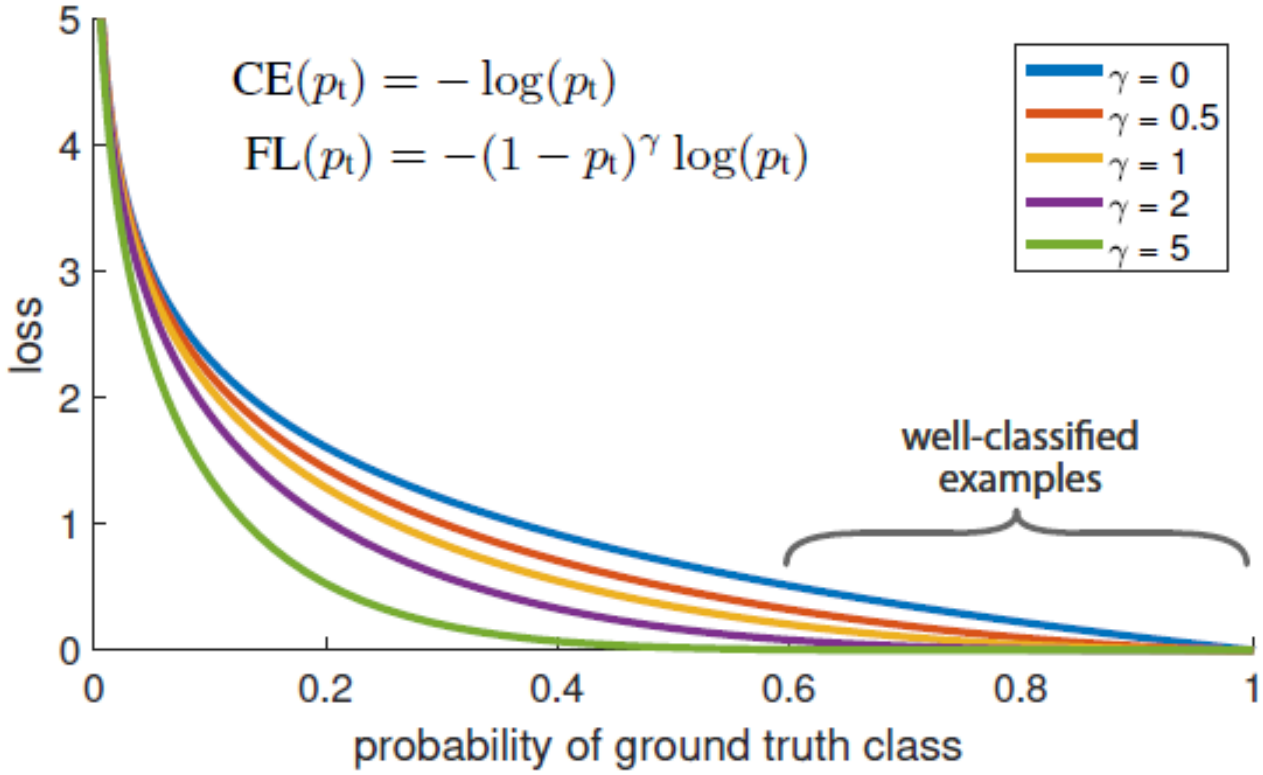
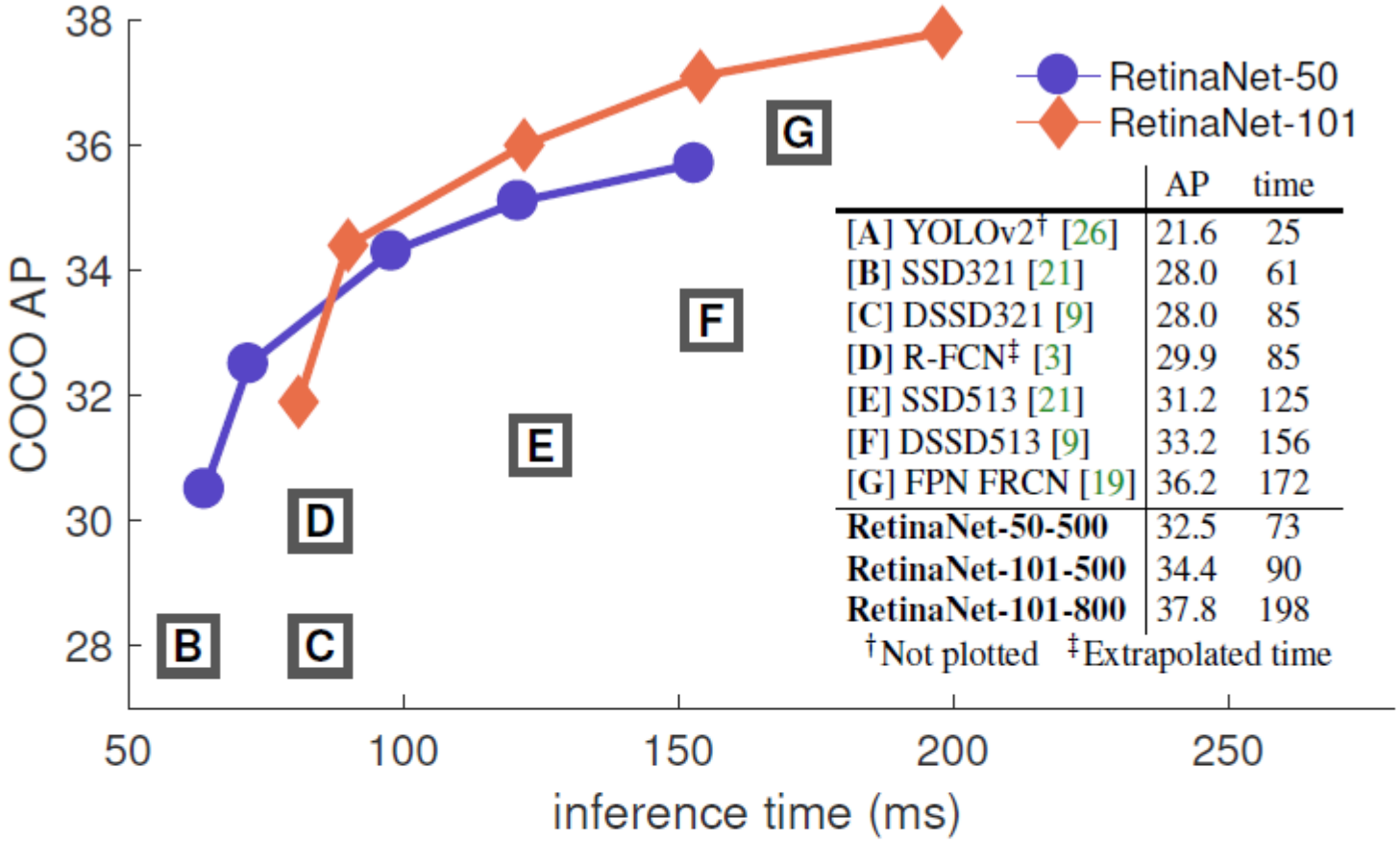


Figure 1. We propose a novel loss we term the *Focal Loss* that adds a factor $(1-p_t)^\gamma$ to the standard cross entropy criterion.



	AP	time
[A] YOLOv2† [26]	21.6	25
[B] SSD321 [21]	28.0	61
[C] DSSD321 [9]	28.0	85
[D] R-FCN‡ [3]	29.9	85
[E] SSD513 [21]	31.2	125
[F] DSSD513 [9]	33.2	156
[G] FPN FRCN [19]	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

Figure 2. Speed (ms) versus accuracy (AP) on COCO `test-dev`. Enabled by the focal loss, our simple one-stage *RetinaNet* detector outperforms all previous one-stage and two-stage detectors, including the best reported Faster R-CNN [27] system from [19]. We

RetinaNet

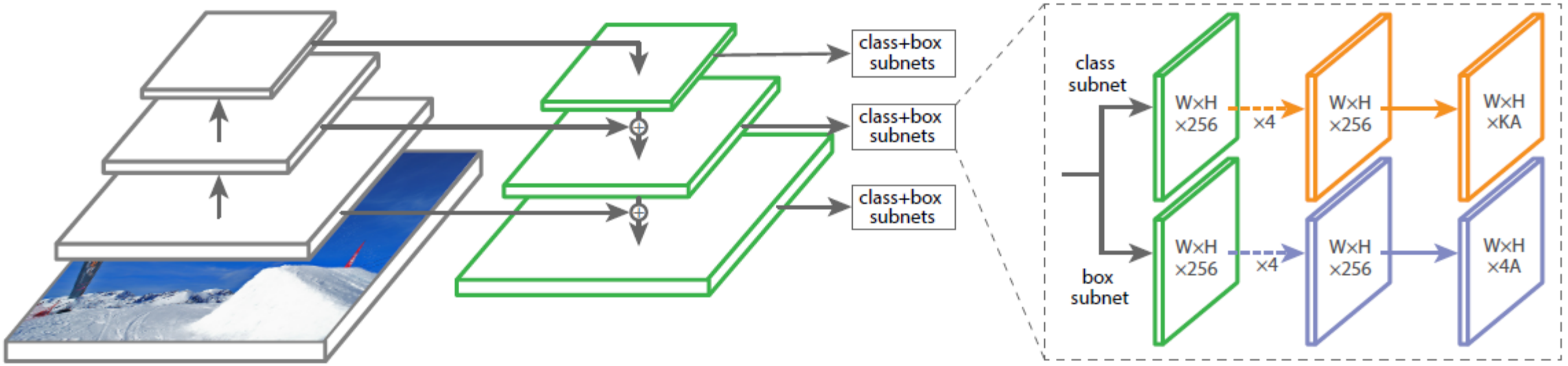
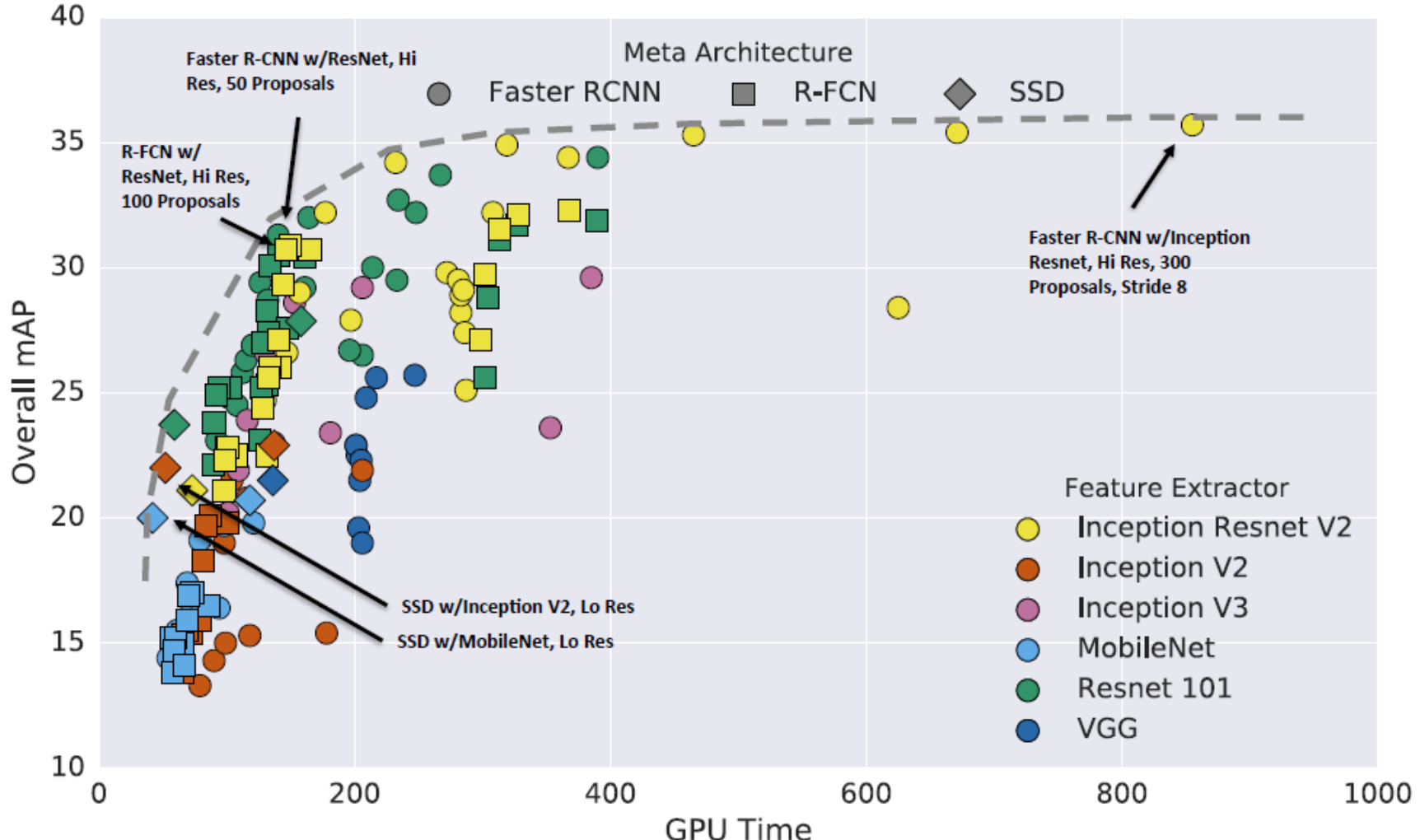


Figure 3. The one-stage **RetinaNet** network architecture uses a Feature Pyramid Network (FPN) [19] backbone on top of a feedforward ResNet architecture [15] (a) to generate a rich, multi-scale convolutional feature pyramid (b). To this backbone RetinaNet attaches two subnetworks, one for classifying anchor boxes (c) and one for regressing from anchor boxes to ground-truth object boxes (d). The network design is intentionally simple, which enables this work to focus on a novel focal loss function that eliminates the accuracy gap between our one-stage detector and state-of-the-art two-stage detectors like Faster R-CNN with FPN [19] while running at faster speeds.

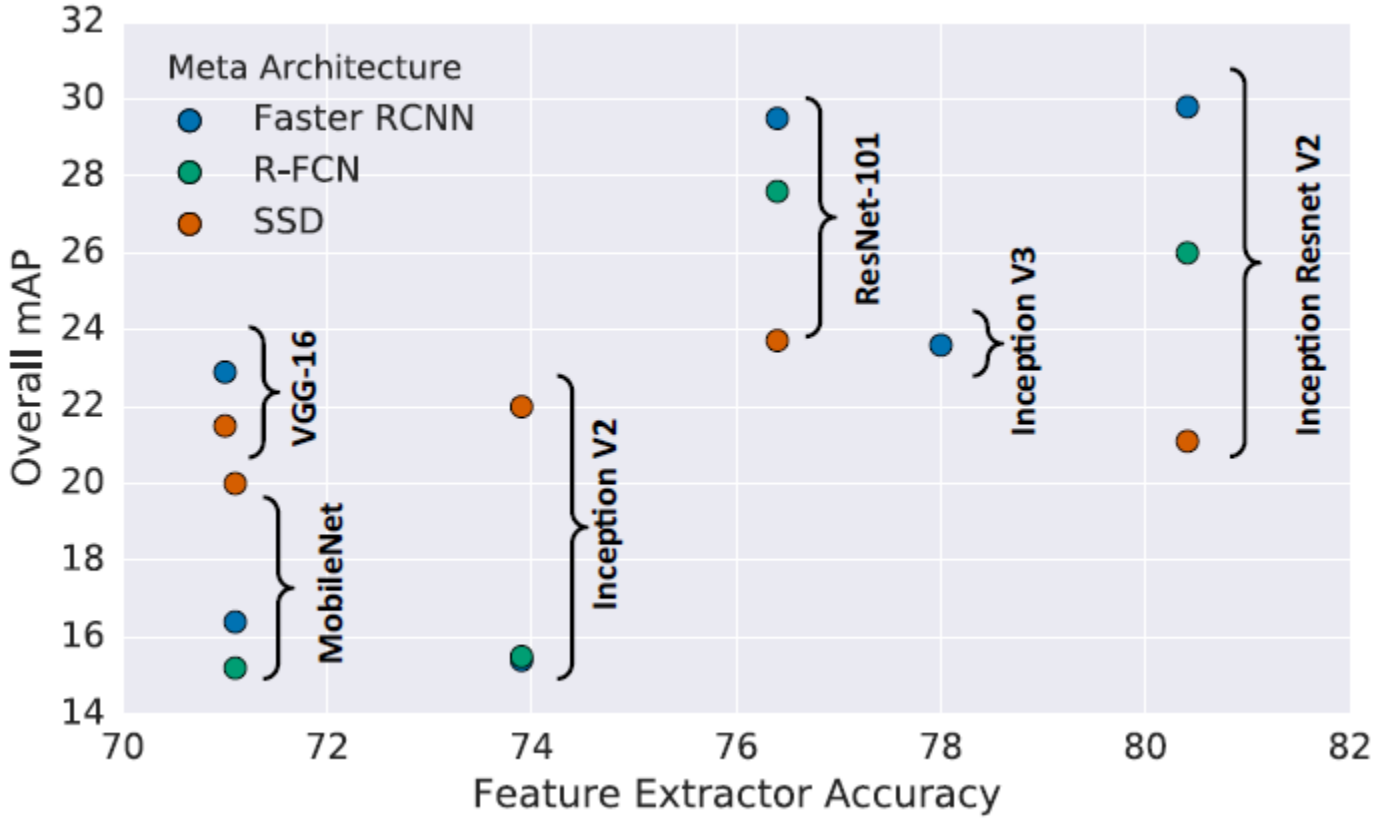
Reference

- 主流的CNN Based Detector詳細的演進與效能比較
  - Google Research的CVPR 2017論文



Model summary	minival mAP	test-dev mAP
(Fastest) SSD w/MobileNet (Low Resolution)	19.3	18.8
(Fastest) SSD w/Inception V2 (Low Resolution)	22	21.6
(Sweet Spot) Faster R-CNN w/Resnet 101, 100 Proposals	32	31.9
(Sweet Spot) R-FCN w/Resnet 101, 300 Proposals	30.4	30.3
(Most Accurate) Faster R-CNN w/Inception Resnet V2, 300 Proposals	35.7	35.6

Table 3: Test-dev performance of the "critical" points along our optimality frontier.



that SSDs performance is less sensitive to the quality of the feature extractor than Faster R-CNN and R-FCN

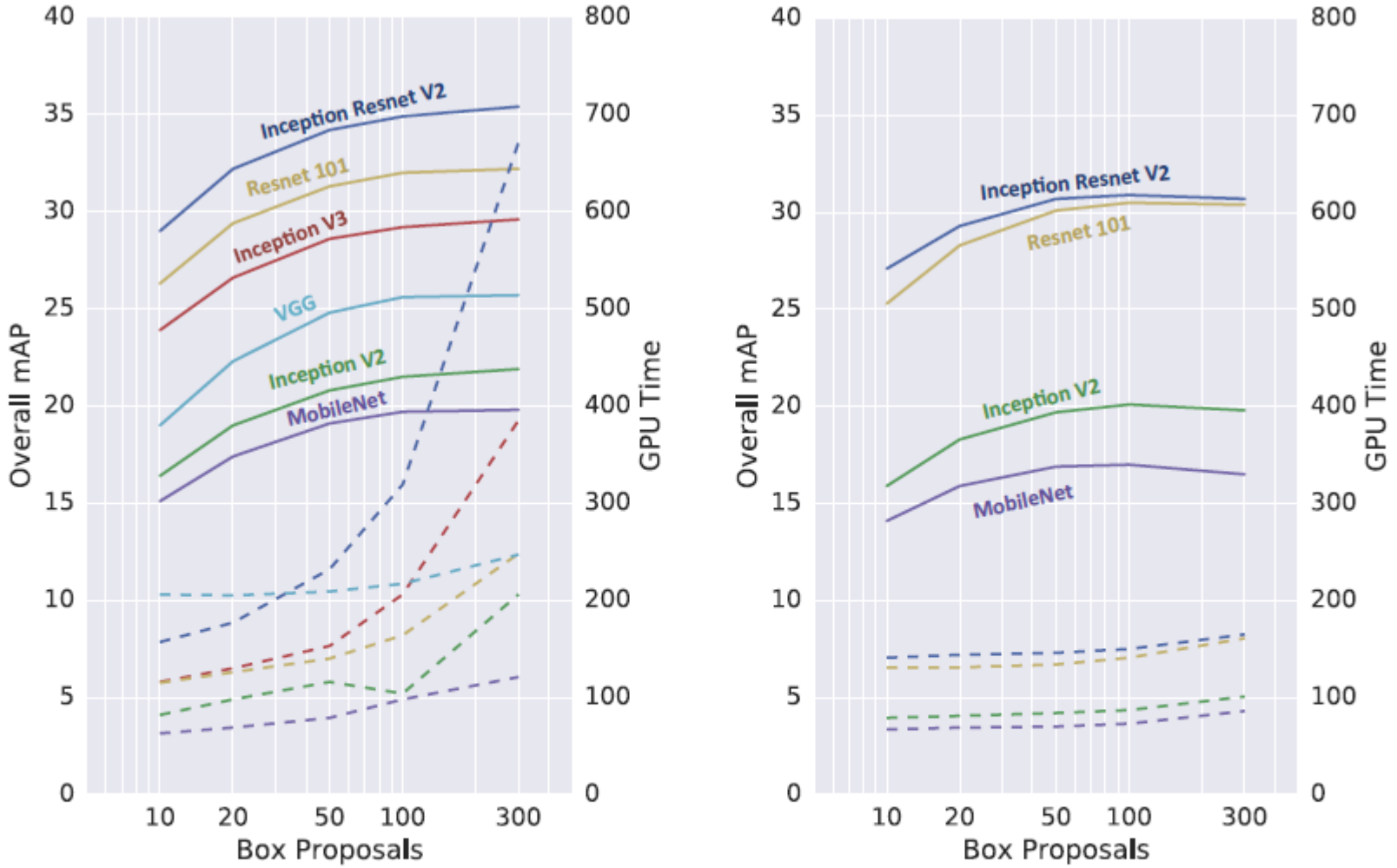
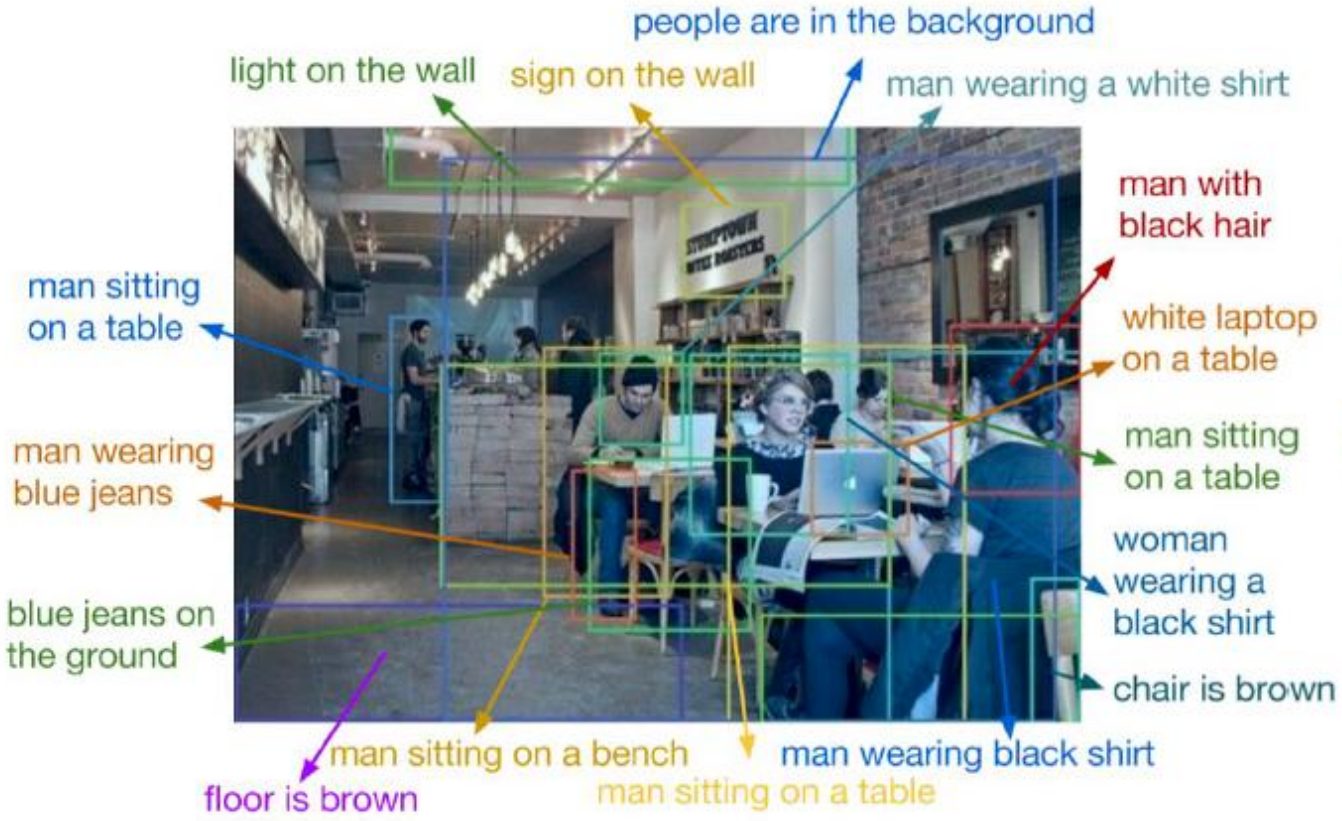
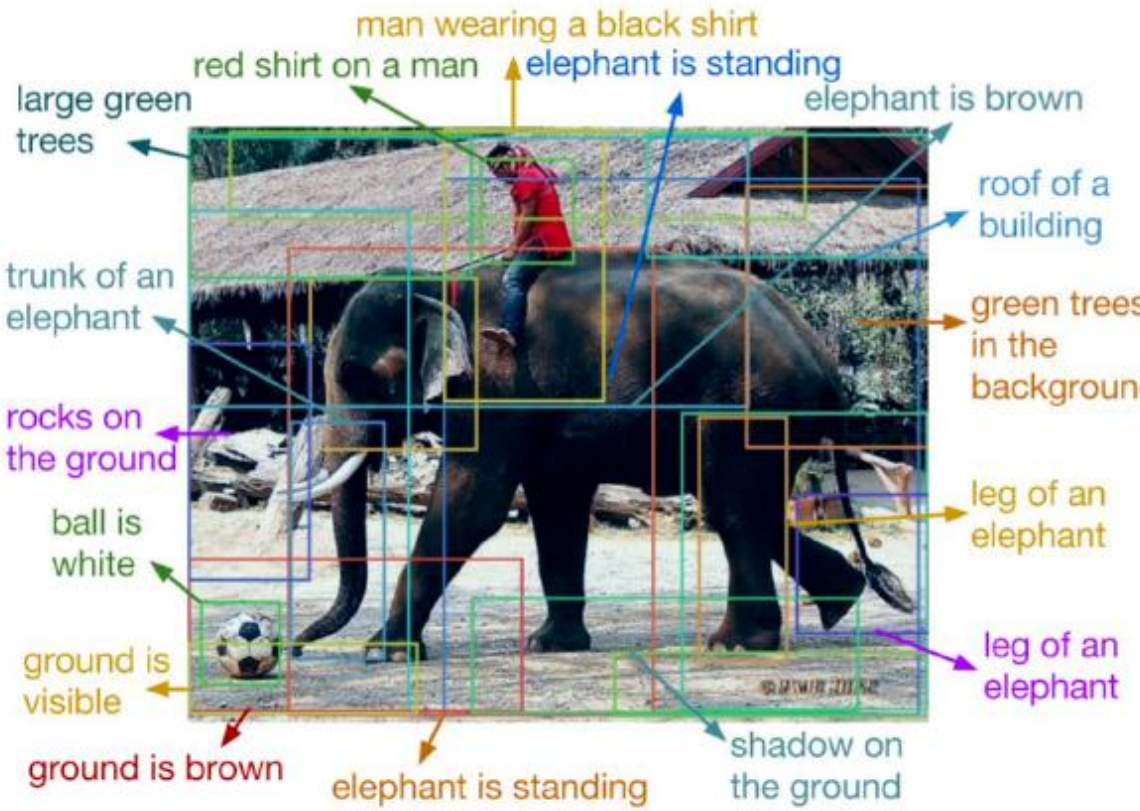


Figure 6: Effect of proposing increasing number of regions on mAP accuracy (solid lines) and GPU inference time (dotted). Surprisingly, for Faster R-CNN with Inception Resnet, we obtain 96% of the accuracy of using 300 proposals by using only 50 proposals, which reduces running time by a factor of 3.

Aside: Object Detection + Captioning = Dense Captioning





Johnson, Karpathy, and Fei-Fei, "DenseCap: Fully Convolutional Localization Networks for Dense Captioning", CVPR 2016
Figure copyright IEEE, 2016. Reproduced for educational purposes.

Source: cs231n

DATASET AND SUMMARY

Table 4
Performance comparison of various object detectors on MS COCO and PASCAL VOC 2012 datasets at similar input image size. Rows colored gray are real-time detectors (>30 FPS).

Model	Year	Backbone	Size	$AP_{[0.5:0.95]}$	$AP_{0.5}$	FPS
R-CNN*	2014	AlexNet	224	-	58.50%	~0.02
SPP-Net*	2015	ZF-5	Variable	-	59.20%	~0.23
Fast R-CNN*	2015	VGG-16	Variable	-	65.70%	~0.43
Faster R-CNN*	2016	VGG-16	600	-	67.00%	5
R-FCN	2016	ResNet-101	600	31.50%	53.20%	~3
FPN	2017	ResNet-101	800	36.20%	59.10%	5
Mask R-CNN	2018	ResNeXt-101-FPN	800	39.80%	62.30%	5
DetectoRS	2020	ResNeXt-101	1333	53.30%	71.60%	~4
YOLO*	2015	(Modified) GoogLeNet	448	-	57.90%	45
SSD	2016	VGG-16	300	23.20%	41.20%	46
YOLOv2	2016	DarkNet-19	352	21.60%	44.00%	81
RetinaNet	2018	ResNet-101-FPN	400	31.90%	49.50%	12
YOLOv3	2018	DarkNet-53	320	28.20%	51.50%	45
CenterNet	2019	Hourglass-104	512	42.10%	61.10%	7.8
EfficientDet-D2	2020	Efficient-B2	768	43.00%	62.30%	41.7
YOLOv4	2020	CSPDarkNet-53	512	43.00%	64.90%	31
DeTR	2020	ResNet-101	-	43.50%	63.80%	20
Swin-L	2021	HTC++	-	57.70%	-	-

Models marked with * are compared on PASCAL VOC 2012, while others on MS COCO.

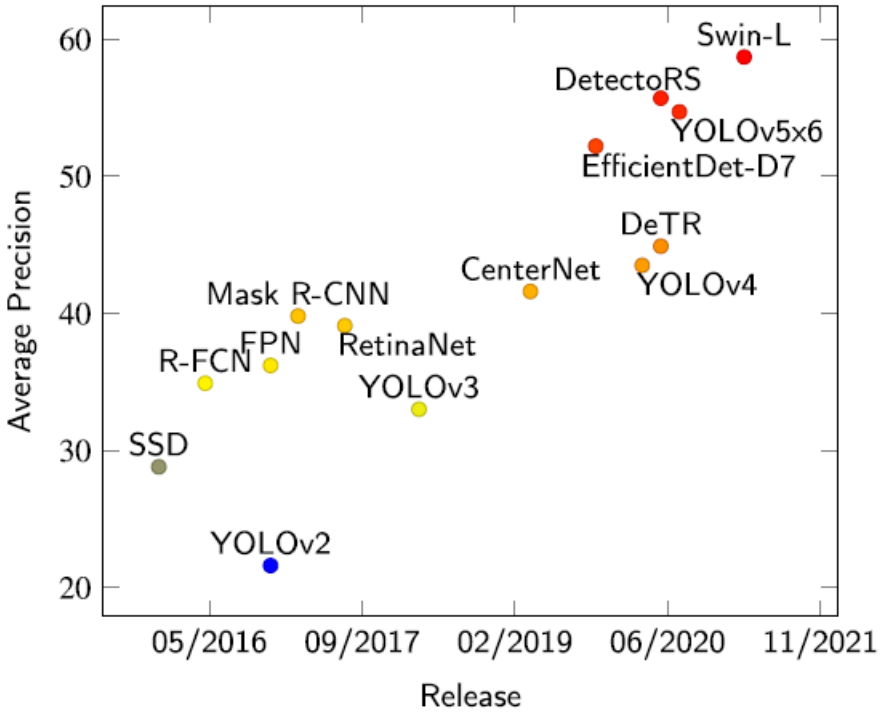


Fig. 10. Performance of object detectors on MS COCO dataset.

Table 2
Comparison of various object detection datasets.

Dataset	Classes	Train			Validation			Test
		Images	Objects	Objects/Image	Images	Objects	Objects/Image	
PASCAL VOC 12	20	5,717	13,609	2.38	5,823	13,841	2.37	10,991
MS-COCO	80	118,287	860,001	7.27	5,000	36,781	7.35	40,670
ILSVRC	200	456,567	478,807	1.05	20,121	55,501	2.76	40,152
OpenImage	600	1,743,042	14,610,229	8.38	41,620	204,621	4.92	125,436

(a) PASCAL VOC 12 (b) MS-COCO (c) ILSVRC (d) OpenImage

Fig. 2. Sample images from different datasets.

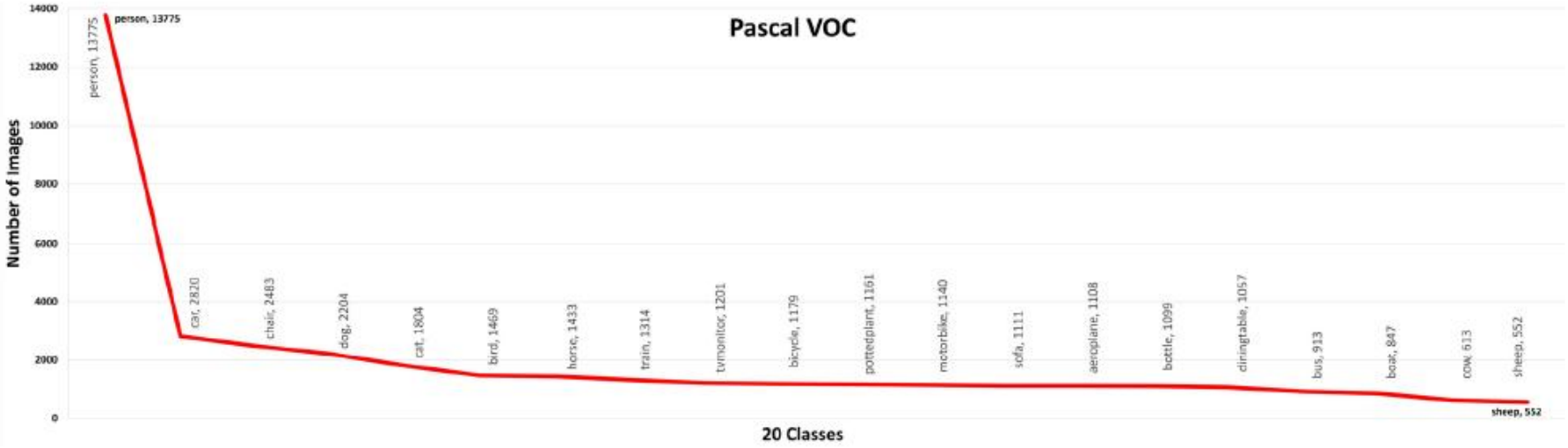


Fig. 3. (This image is best viewed in PDF form with magnification.) Number of images for different classes annotated in the PascalVOC dataset [38].

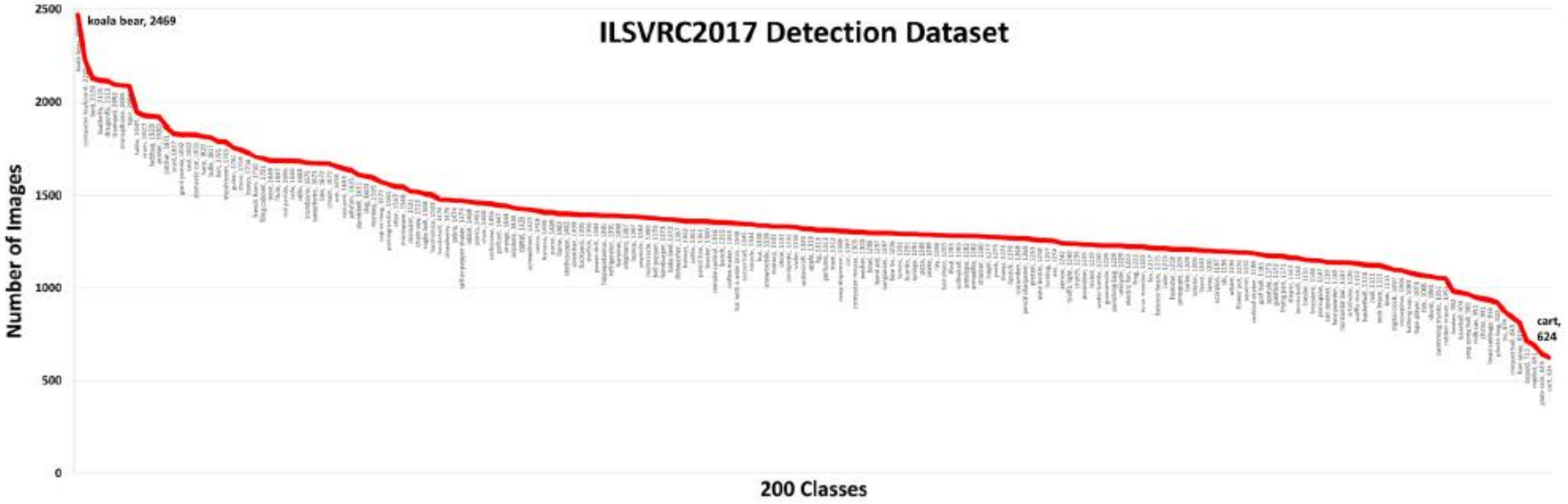


Fig. 4. (This image is best viewed in PDF form with magnification.) Number of images for different classes annotated in the ImageNet dataset [38].

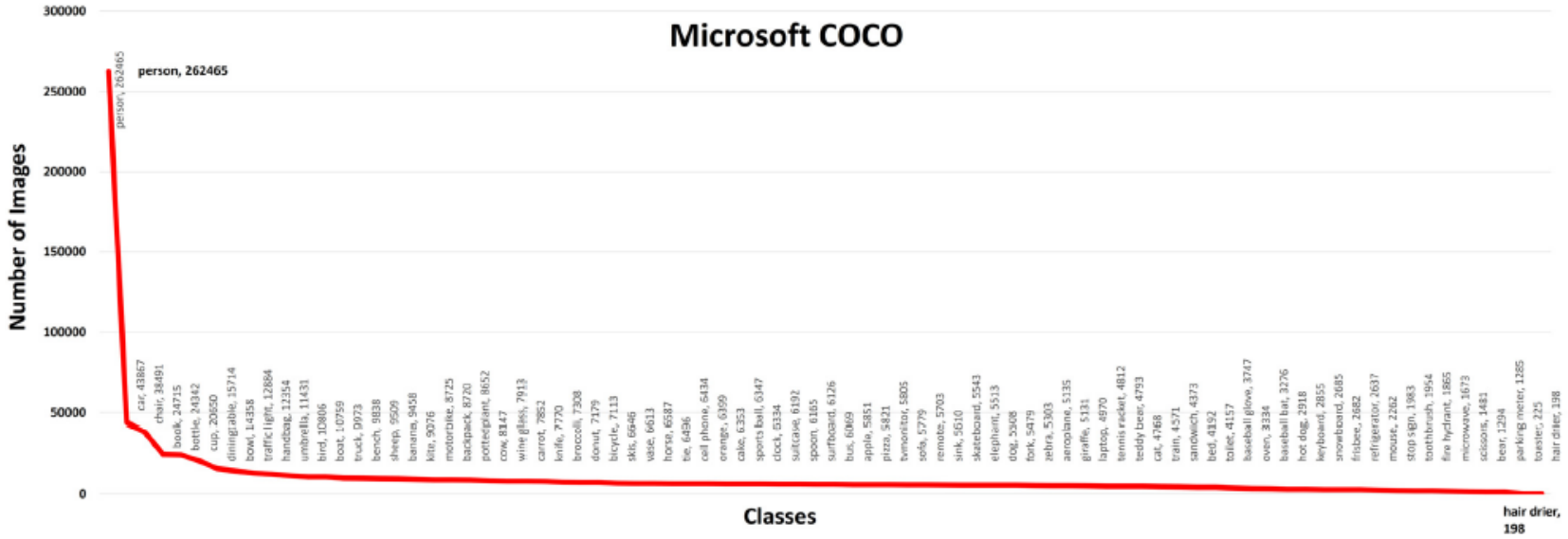
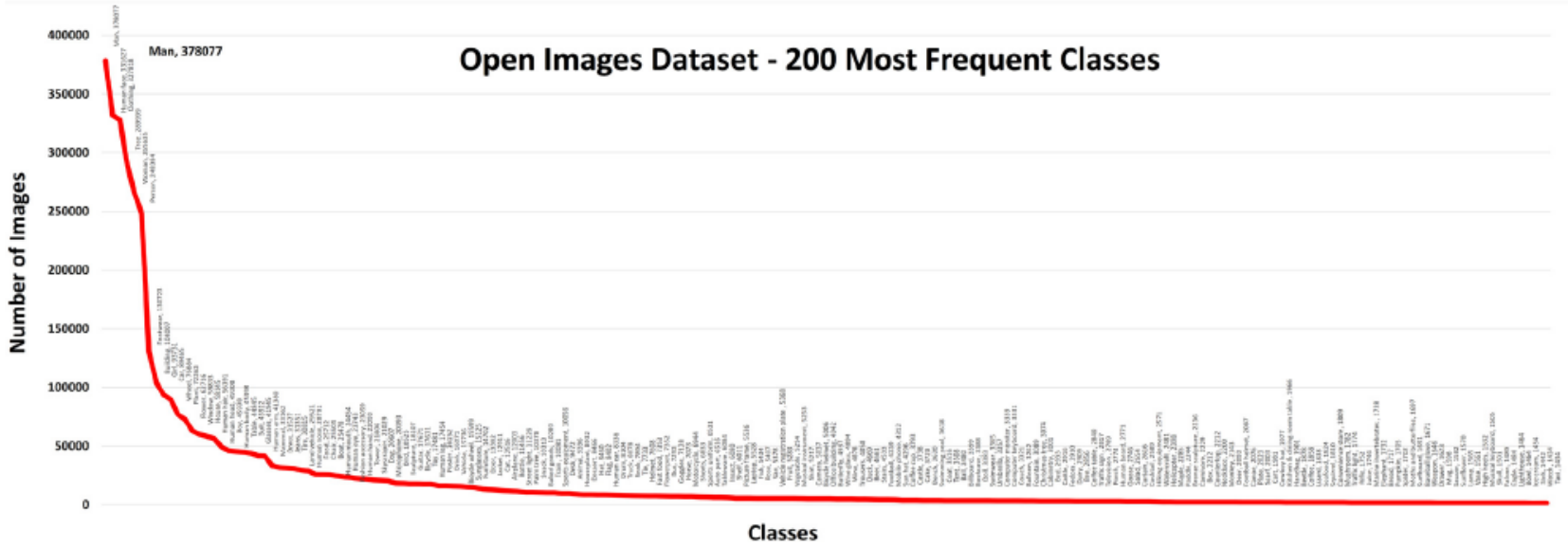


Fig. 5. (This image is best viewed in PDF form with magnification.) Number of images for different classes annotated in the MS-COCO dataset [38].



Classic Datasets

PASCAL
20 categories
6k training images
6k validation images
10k test images

COCO
80 categories
200k training images
60k val + test images

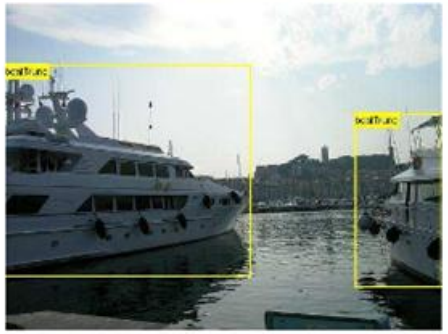
ILSVRC
200 categories
456k training images
60k validation + test images

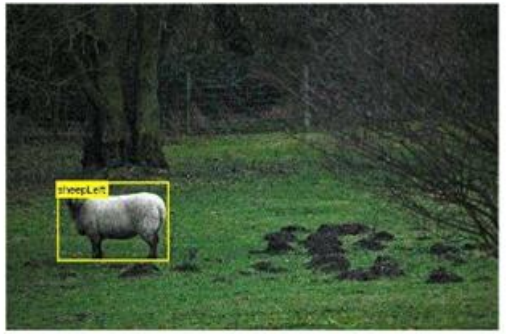
Classic Datasets

PASCAL2
Pattern Analysis, Statistical Modelling and Computational Learning



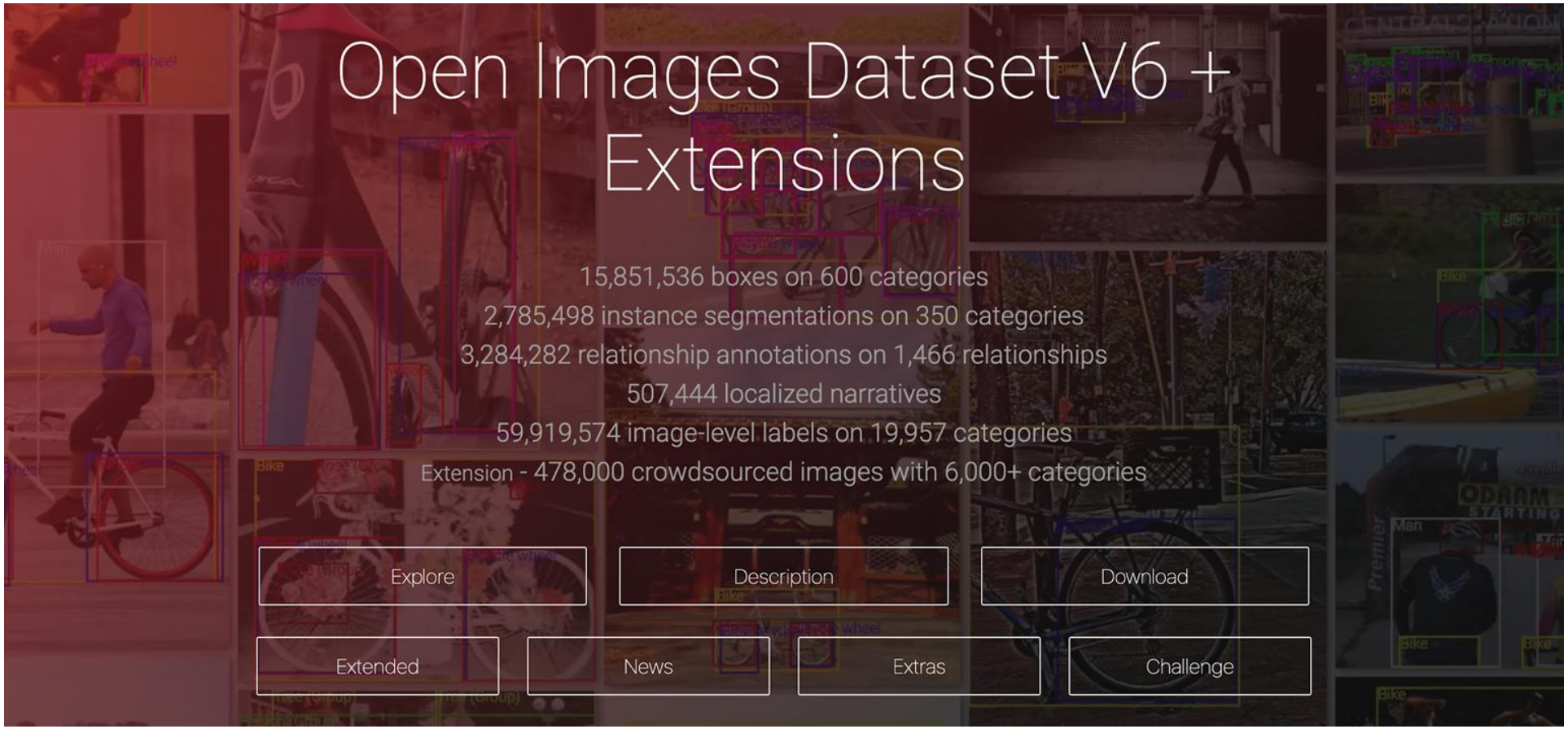



Google AI

Open Images Dataset

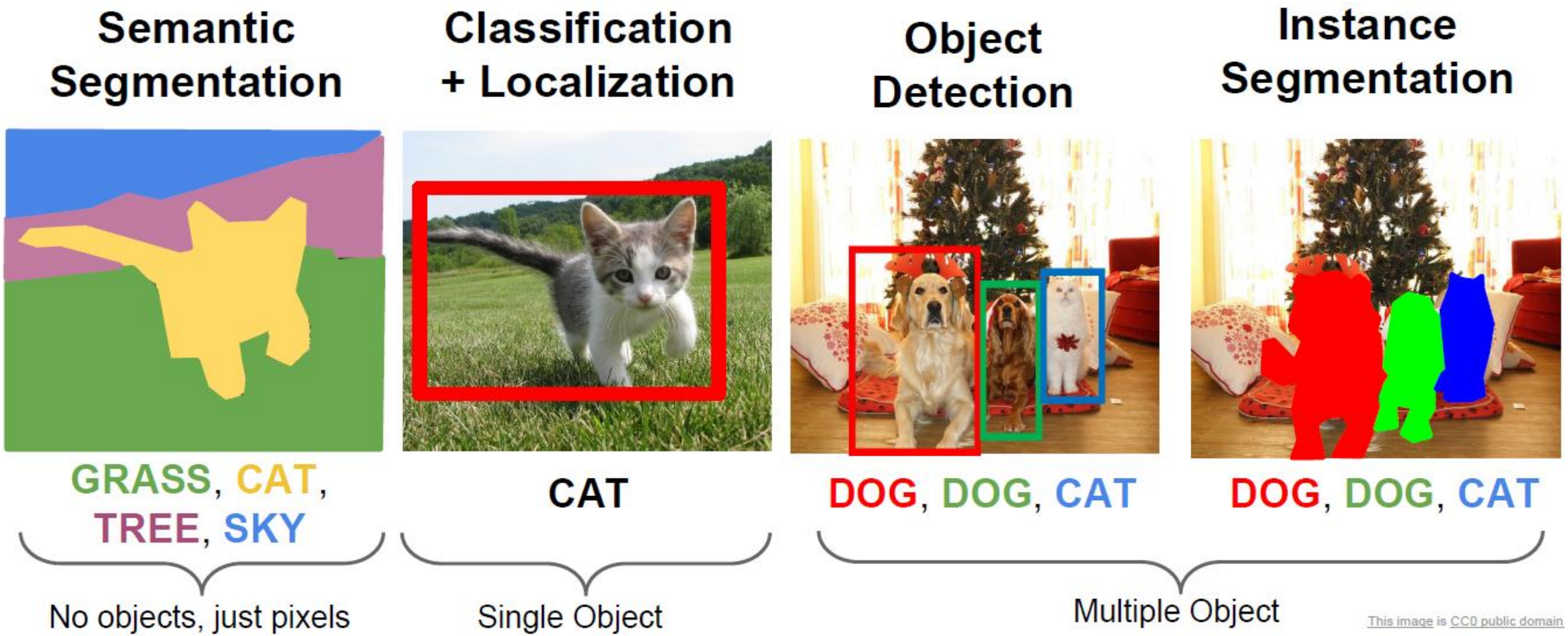


Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., ... & Ferrari, V. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. IJCV 2020. [dataset]

INSTANCE SEGMENTATION

Ref: ICCV 2017 tutorial, Mask R-CNN: A perspective on equivariance

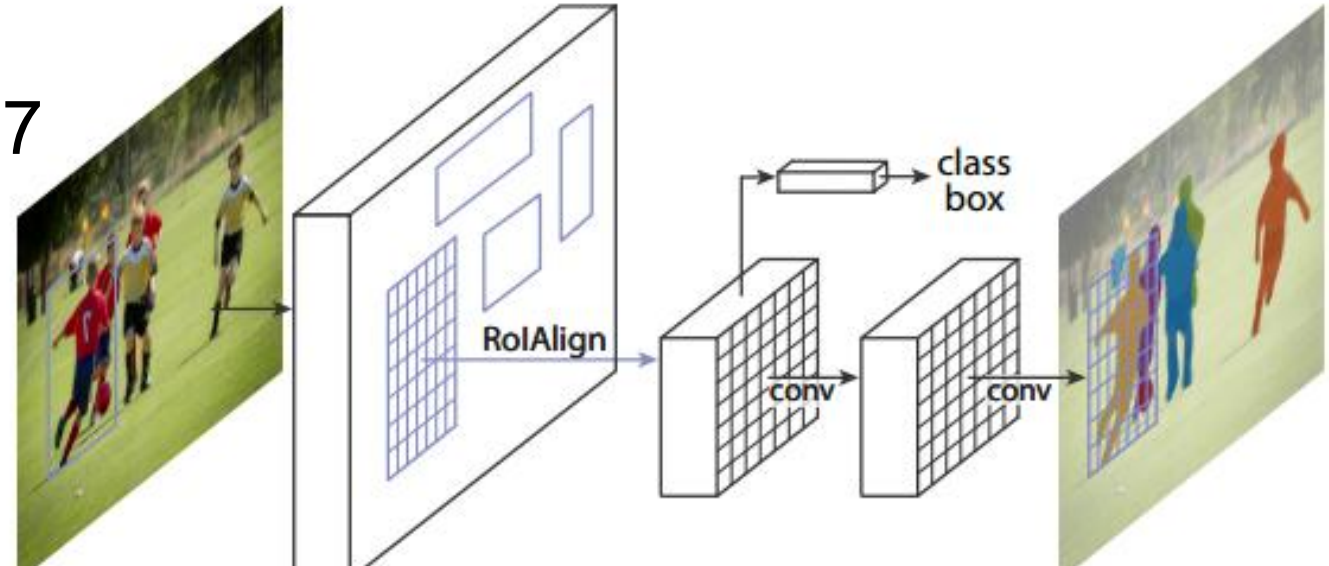
Object Detection and Semantic Segmentation



Source: cs231n

Mask R-CNN: object instance segmentation

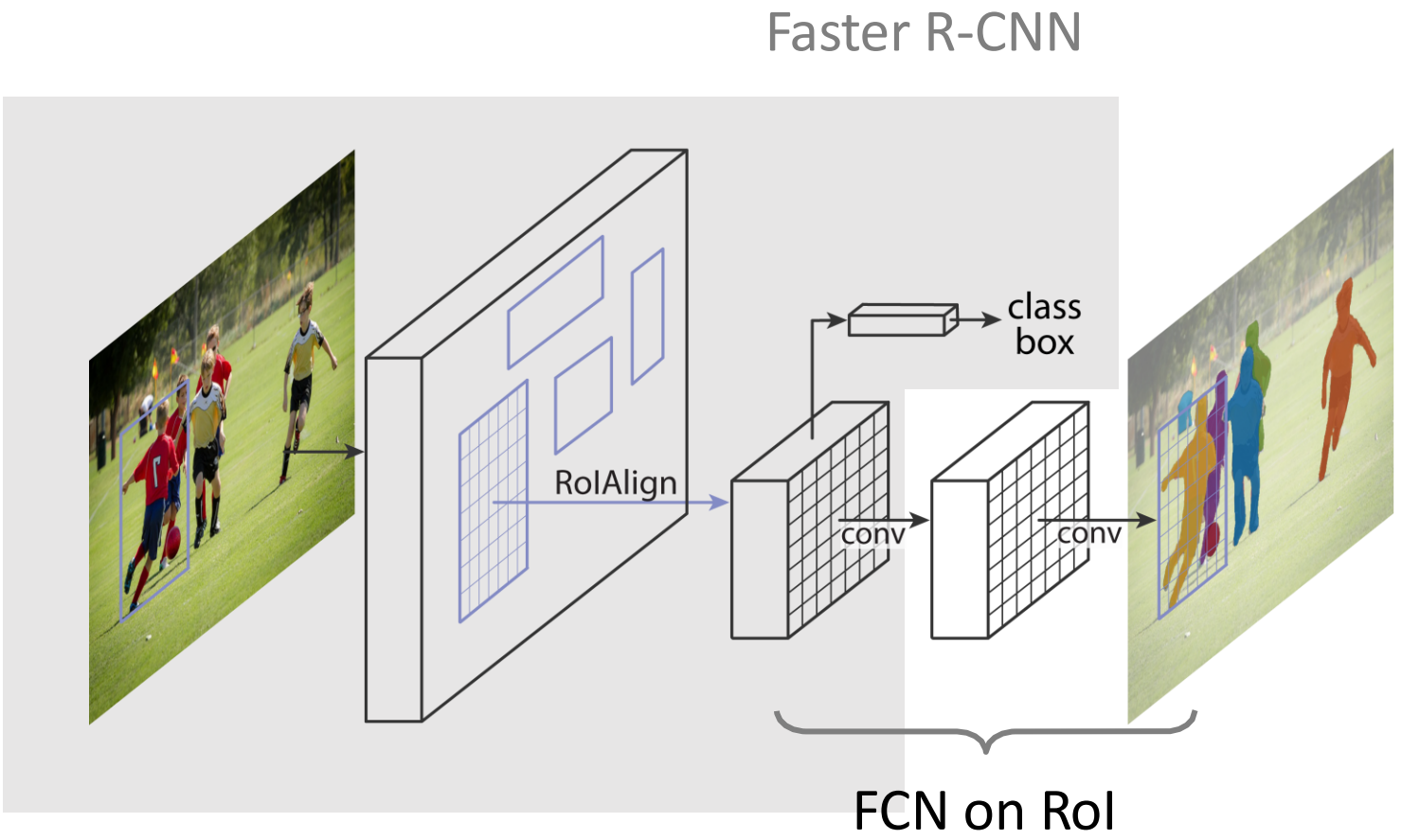
- Faster R-CNN by adding a prediction of object mask
  - Faster R-CNN (class label, bounding box offset) + object mask
  - a binary mask that says whether or not a given pixel is part of an object
  - Nearly as fast as faster R-CNN
- Easy extended to other tasks
  - instance segmentation, bounding-box object detection, and person keypoint detection.
- Best performance up to 2017



Source:2017, mask R-CNN

Mask R-CNN

- Mask R-CNN = **Faster R-CNN** with **FCN** on RoIs



Keys of MASK-RCNN

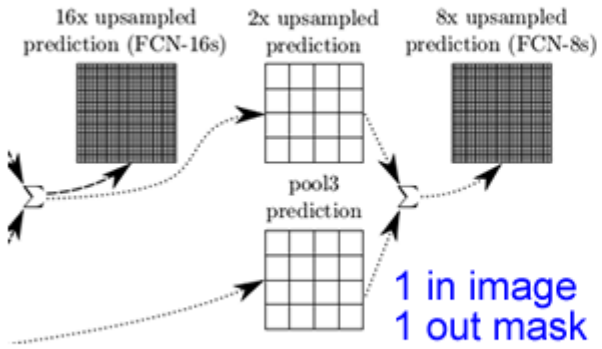
- Misalignment between image and feature map
  - ROIPool (quantization) => ROIAlign (no quantization)
  - Simple but large impact: 10~50% accuracy improvement on mask
- decouple mask prediction and class prediction
  - Binary mask prediction for each object: object or not
    - No competition among classes
    - FCN: per pixel multi-class categorization (bad for instance segmentation)
  - Class of each mask pixel depends on ROI classification result
  - Through new loss function
- Mask representation
  - Use FCN (fully convolutional network) to predict m*m mask for each ROI
    - Fewer parameters than fully connected type
  - Preserve spatial structure

Source: mask R-CNN

Training

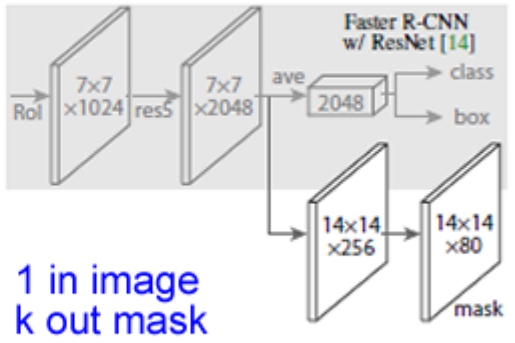
- Loss function
  - $L = L_{cls} + L_{box} + L_{mask}$
  - Classification loss + bounding box loss + mask loss
  - L_{mask} per-pixel sigmoid and a binary loss
    - average binary cross-entropy loss
  - Generate masks for every class without competition among classes

FCIS



VS

Mask R-CNN



Source: mask R-CNN

多分类的Softmax with entropy loss

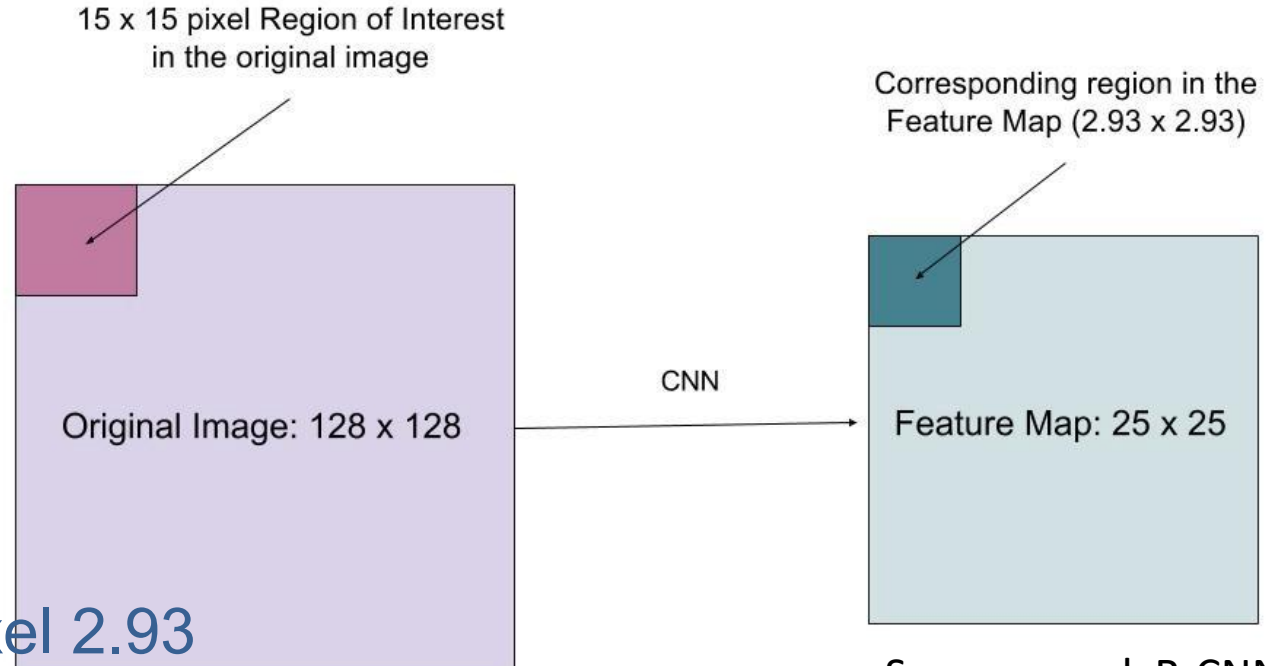
Sigmoid binary cross-entropy loss

Source: mask R-CNN

ROIPool =>ROIAlign (quantization or not)

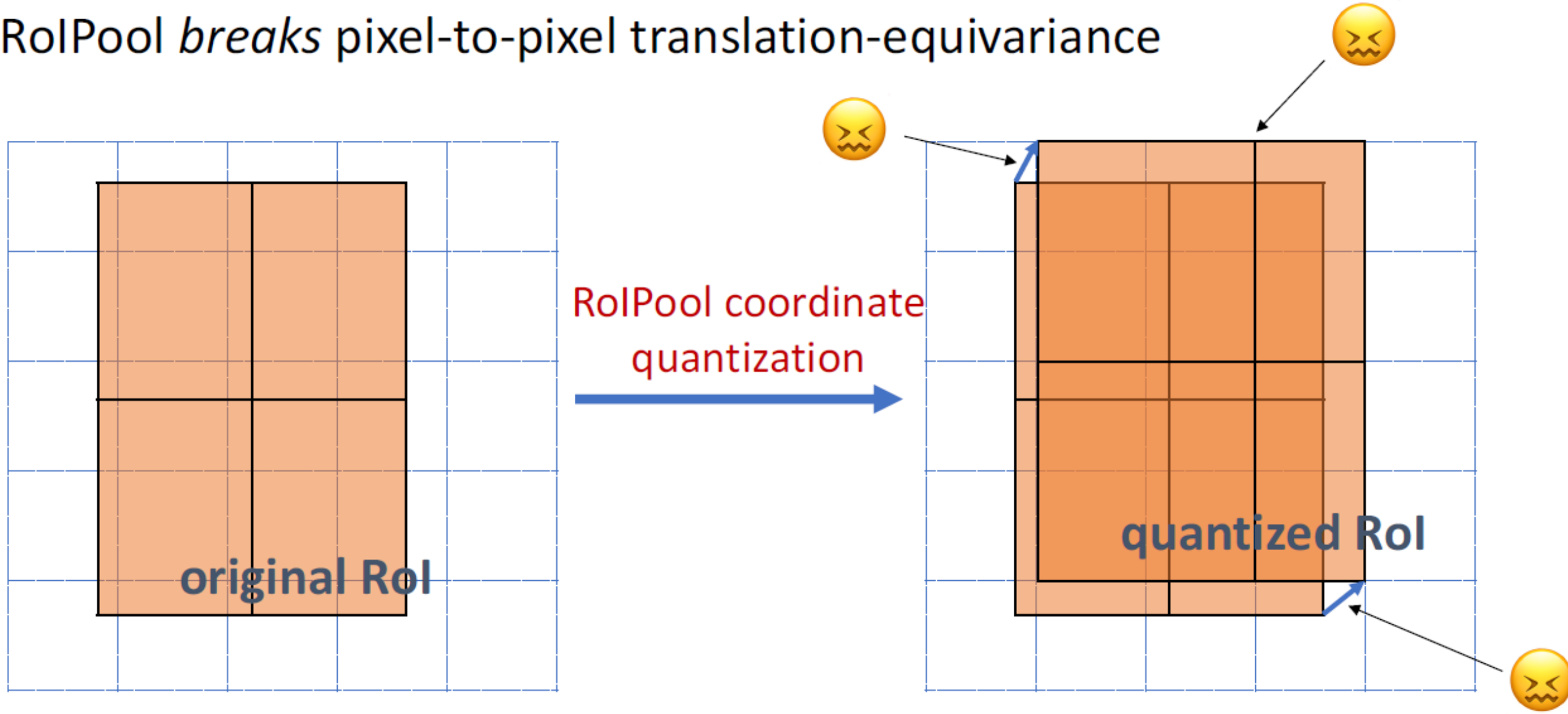
- How do we accurately map a region of interest from the original image onto the feature map?
- For left image – feature mapping example
  - 1pixel/image => ~25/128pixels in feature map
  - Select 15 pixels/image => 15*25/128 ~=2.93pixels

- ROIpool
  - Rounding to 2pixels
  - Cause slight misalignment
- ROIAlign: No rounding
  - Use bilinear interpolation to get precise idea of what would be at pixel 2.93



RoIPool v.s. RoIAlign

- RoIPool *breaks* pixel-to-pixel translation-equivariance



Ablation: RoIPool vs. RoIAlign

baseline: ResNet-50-Conv5 backbone, **stride=32**

	mask AP			box AP		
	AP	AP_{50}	AP_{75}	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}
RoIPool	23.6	46.5	21.6	28.2	52.7	26.9
RoIAlign	**30.9**	**51.8**	**32.1**	**34.0**	**55.3**	**36.4**
	+7.3	*+ 5.3*	*+10.5*	*+5.8*	*+2.6*	*+9.5*

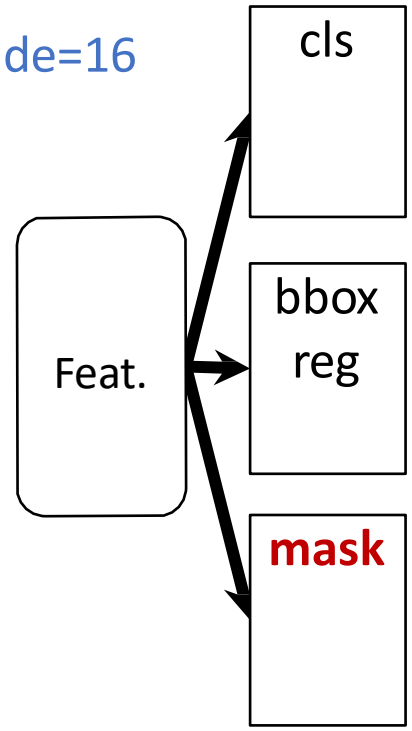
- huge gain at high IoU, in case of big stride (32)
- nice box AP without dilation/upsampling

Source: mask R-CNN

Ablation: Multinomial vs. Binary Masks

baseline: ResNet-50-Conv4 backbone, stride=16

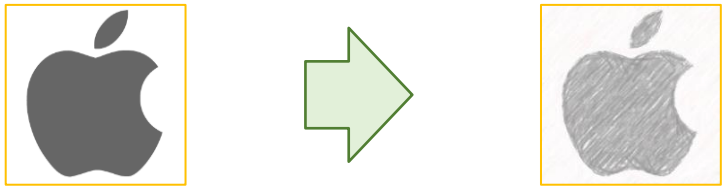
	AP	AP_{50}	AP_{75}
softmax	24.8	44.1	25.1
sigmoid	**30.3**	**51.2**	**31.5**
	+5.5	*+7.1*	*+6.4*



- cls head: did recognition

"apple"

- mask head: no need to recognize again



Decoupling via per class binary masks (sigmoid) gives large gains over multinomial masks (softmax).

每個類別獨立地預測一個binary mask，沒有引入類間競爭

Source: mask R-CNN

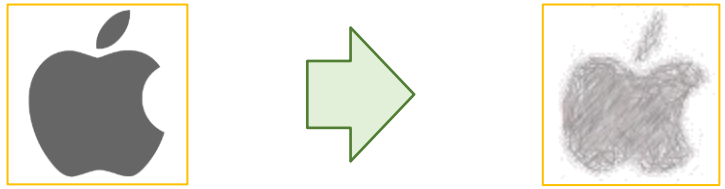
Ablation: MLP vs. FCN mask

baseline: ResNet-50-FPN backbone

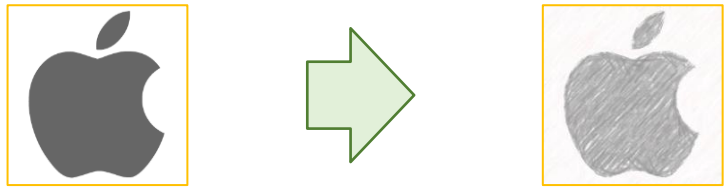
	mask branch	AP	AP_{50}	AP_{75}
MLP	fc: 1024→1024→80·28^2	31.5	53.7	32.8
MLP	fc: 1024→1024→1024→80·28^2	31.5	54.0	32.6
FCN	conv: 256→256→256→256→256→80	**33.6**	**55.2**	**35.3**

- +2.1 point

- MLP: lose “place-coded” info, too abstract



- FCN: translation-equivariant



Source: mask R-CNN

Instance Segmentation Results on COCO

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MNC [7]	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [20] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [20] +OHEM	ResNet-101-C5-dilated	33.6	54.5	-	-	-	-
Mask R-CNN	ResNet-101-C4	33.1	54.9	34.8	12.1	35.6	51.1
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	ResNeXt-101-FPN	**37.1**	**60.0**	**39.4**	**16.9**	**39.9**	**53.5**

- **2 AP better** than SOTA w/ R101, without bells and whistles
- **200ms / img**
  - benefit from better features (ResNeXt [Xie et al. CVPR'17])

Source: mask R-CNN

Object Detection Results on COCO

	backbone	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{bb}_{S}	AP^{bb}_{M}	AP^{bb}_{L}
Faster R-CNN+++ [15]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [22]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [17]	Inception-ResNet-v2 [32]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [31]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
Faster R-CNN, RoIAlign	ResNet-101-FPN	37.3	59.6	40.3	19.8	40.2	48.8
Mask R-CNN	ResNet-101-FPN	38.2	60.3	41.7	20.1	41.1	50.2
Mask R-CNN	ResNeXt-101-FPN	**39.8**	**62.3**	**43.4**	**22.1**	**43.2**	51.2

bbox detection improved by:

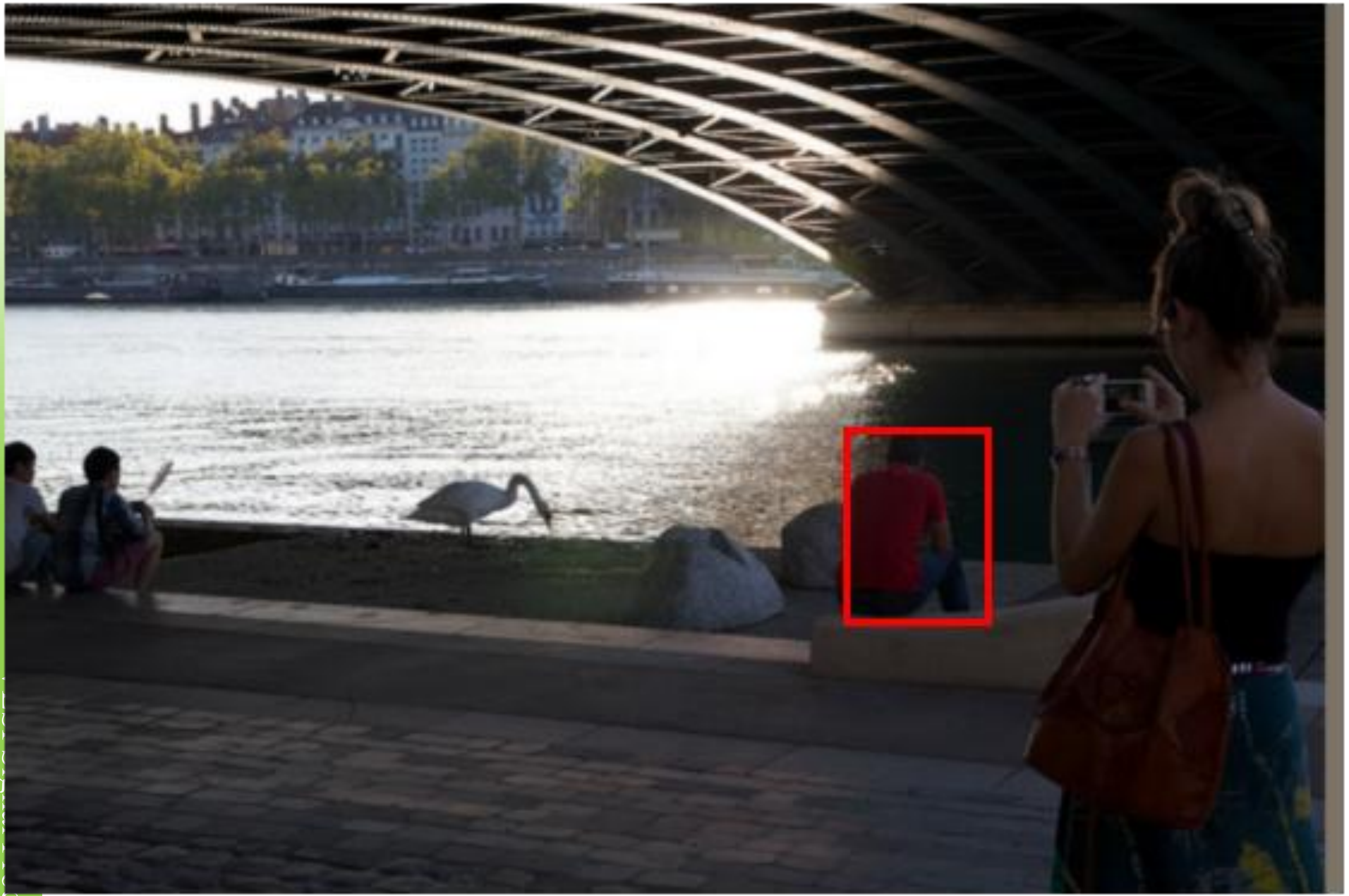
- RoIAlign
- Multi-task training w/ mask

Source: mask R-CNN

disconnected object

Mask R-CNN results on COCO

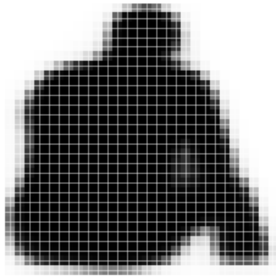
Source: mask R-CNN



Validation image with box detection shown in red

28x28 soft prediction from Mask R-CNN

(enlarged)



Soft prediction resampled to image coordinates

(bilinear and bicubic interpolation work equally well)

Final prediction (threshold at 0.5)

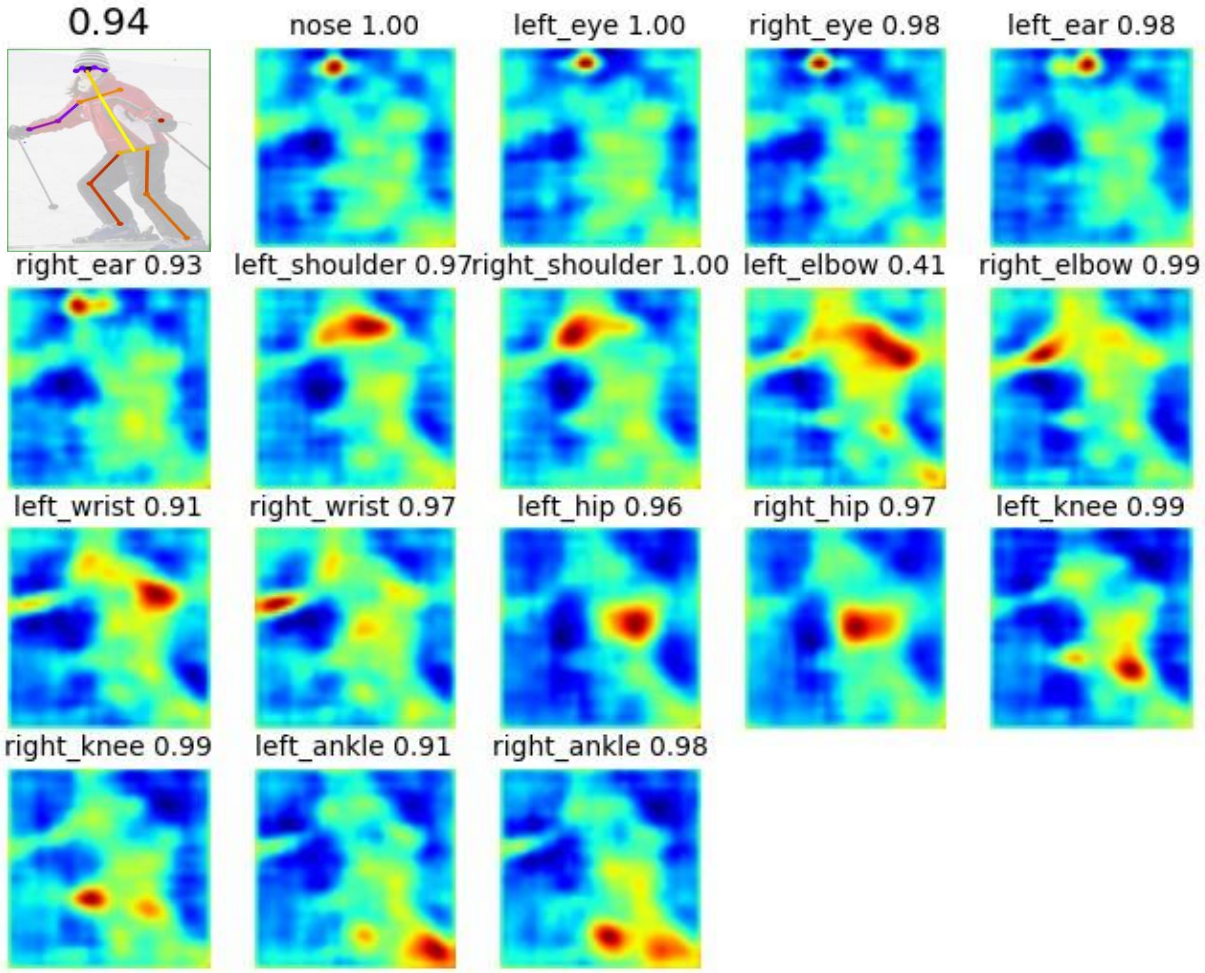
Source: mask R-CNN

Mask R-CNN: for Human Keypoint Detection

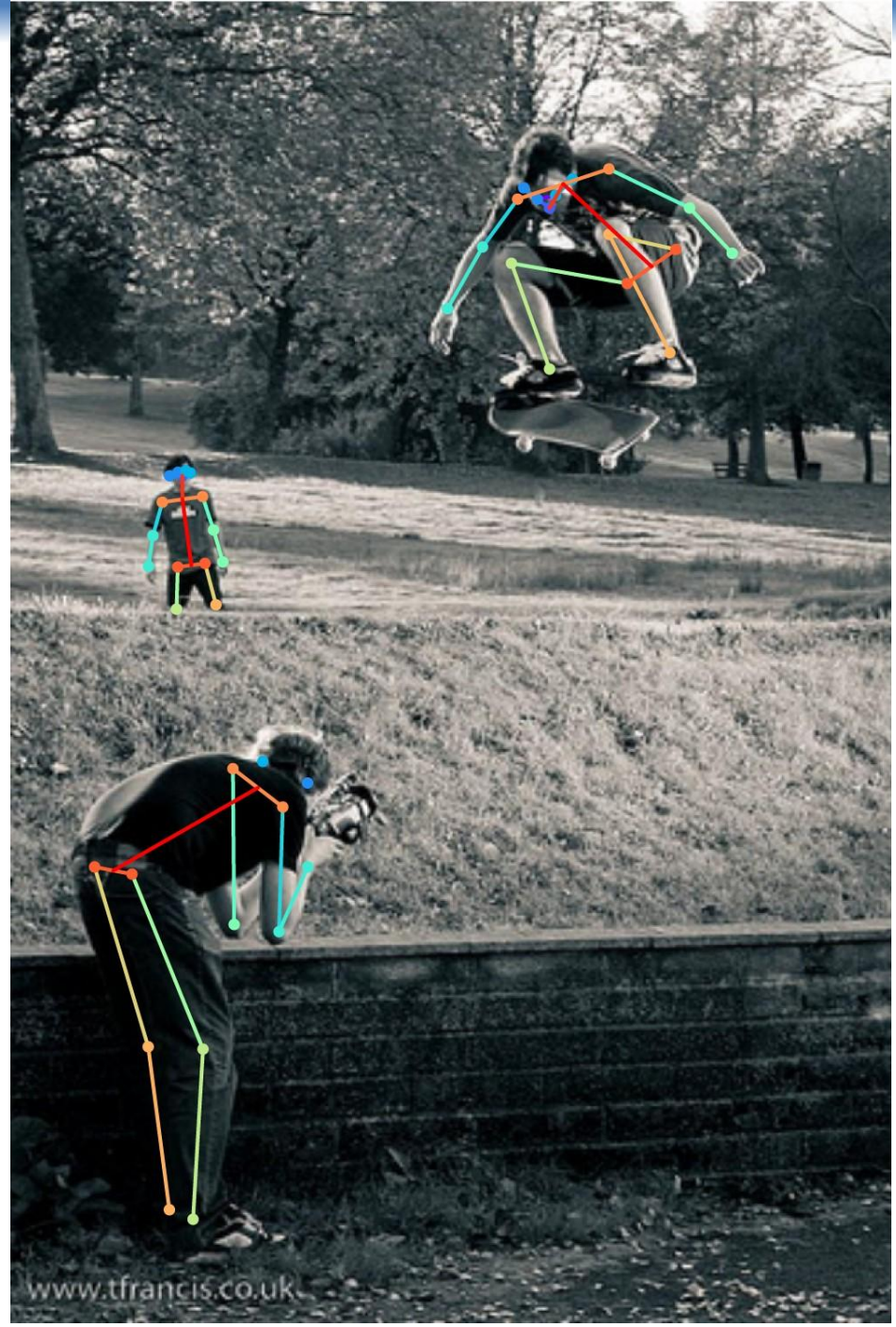
- 1 keypoint = 1-hot “mask”
- Human pose = 17 masks

- Softmax over spatial locations
  - e.g. 56^2-way softmax on 56x56

- Desire the same equivariances
  - translation, scale, aspect ratio



Source: mask R-CNN



Source: mask R-CNN

YOLACT++:You Only Look At CoefficienTs

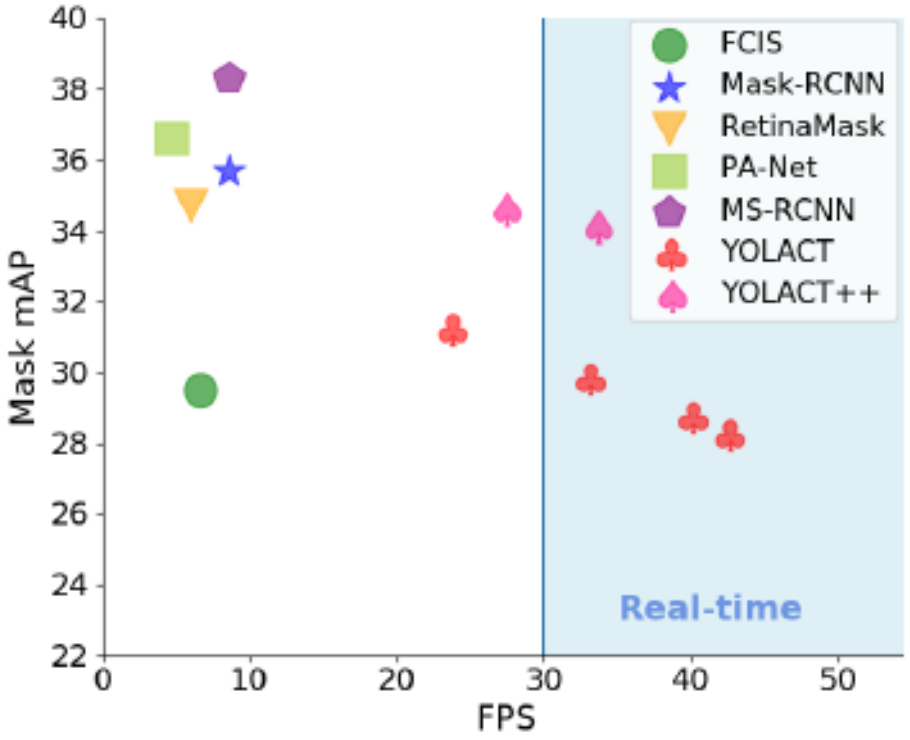


Fig. 1: Speed-performance trade-off for various instance segmentation methods on COCO. To our knowledge, ours is the first *real-time* (above 30 FPS) approach with around 30 mask mAP on COCO `test-dev`.

The prototype generation branch (protonet) predicts a set of k prototype masks for the entire image

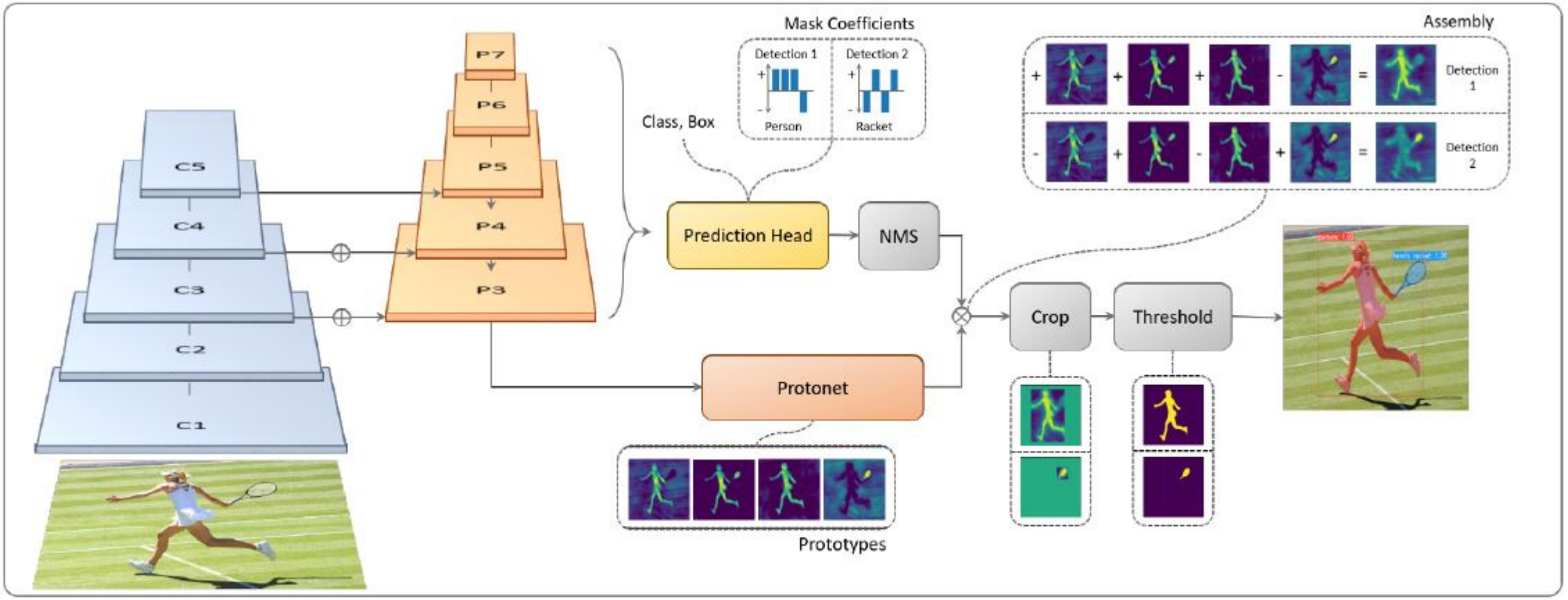


Fig. 2: **YOLACT Architecture** Blue/yellow indicates low/high values in the prototypes, gray nodes indicate functions that are not trained, and $k = 4$ in this example. We base this architecture off of RetinaNet [25] using ResNet-101 + FPN.

Use prototypes to assemble masks for instance segmentation

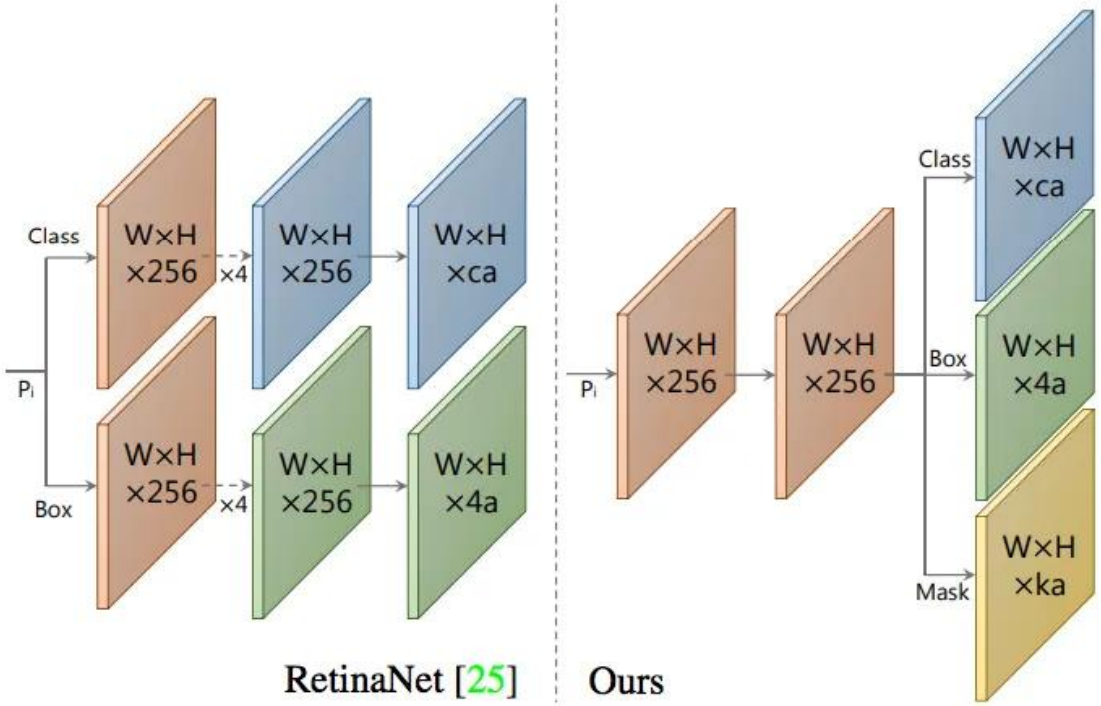


Figure 4: **Head Architecture** We use a shallower prediction head than RetinaNet [25] and add a mask coefficient branch. This is for c classes, a anchors for feature layer P_i, and k prototypes. See Figure 3 for a key.

輸出為三個

預測類別的機率 — c classes，訓練時採用c+1(背景類別)
bbox — 4 classes
mask coefficients — k classes（對應到Prototype Masks的個數）

相較於 mask coefficients 輸出都為正數，使用 mask coefficients 的輸出有 +- 的結果更好能組合出更多的選擇，因此選用 tanh()

Concept

- spatial coherence in Conv layer instead of FC layer
  - fc layer: semantic vector
  - Conv layer: mask coefficients and prototype masks

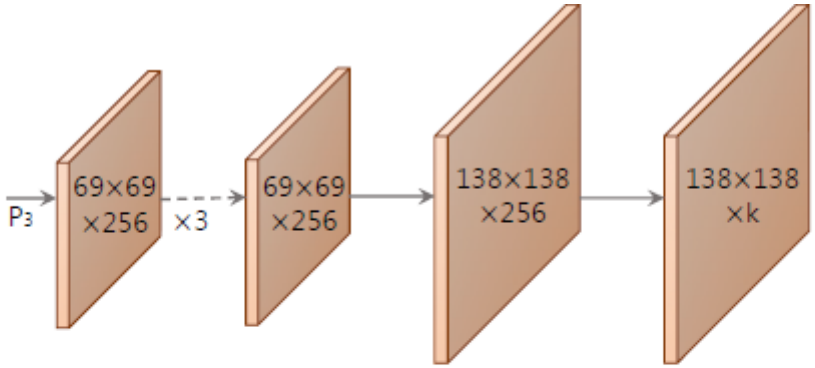


Fig. 3: **Protonet Architecture** The labels denote feature size and channels for an image size of 550 × 550. Arrows indicate 3 × 3 *conv* layers, except for the final *conv* which is 1 × 1. The increase in size is an upsample followed by a *conv*. Inspired by the mask branch in [2].

將 FPN 的 P3 輸入進 Protonet，
主要考量其一是該層是模型中最深層 layer，
能夠處理較為複雜的特徵，
其二是它有著較高的解析度，
因此對於較小的物件也會有較好的準確度。
輸出為 k 個 prototypes mask，
prototypes mask 並非與 Instance 對應，
需與 mask coefficients 搭配才能萃取出 Instance。
最終組合為 P(Prototype Masks) * C(Mask coefficients)，
再經由 Sigmoid 輸出 0 ~ 1 的 mask

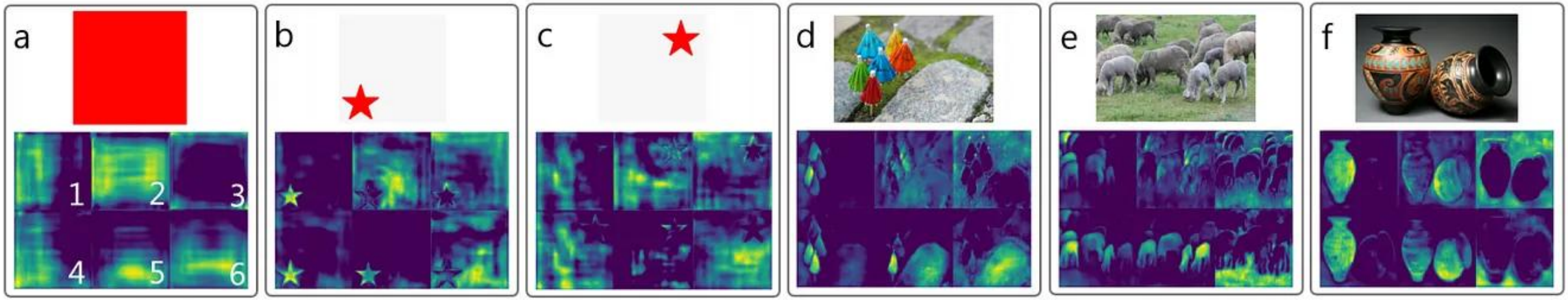


Figure 5: **Prototype Behavior** The activations of the same six prototypes across different images. Prototypes 1, 4, and 5 are partition maps with boundaries clearly defined in image a, prototype 2 is a bottom-left directional map, prototype 3 segments out the background and provides instance contours, and prototype 6 segments out the ground.

單純以 FCN 不採用 padding 的情況下，是不會有下面那 1 ~ 6 的結果的！！
因為如果輸入一樣的話，單純使用 Conv 出來的結果也會是一樣的！因為 FCN 的特性是平移不變性。
但因為有 Padding 的動作，（舉例來說邊界 padding 0，但是整張圖片都是 1。）讓 Prototype 有辦法學會邊界的距離這項特徵， Prototype 有了平移敏感性的特性，

舉例來說 d 圖的小雨傘，
我們可以透過 prototype 5 — prototype 4 來分割出小綠傘的部分。
prototype 3 學會的是前背景分離，
prototype 6 學會的是辨認地面的部分，
而我們也能發現每個 prototype 其實擁有的特性不只一個，
舉例來說 prototype 4 不只能把左右分開，還能將左下角的物件 highlight。

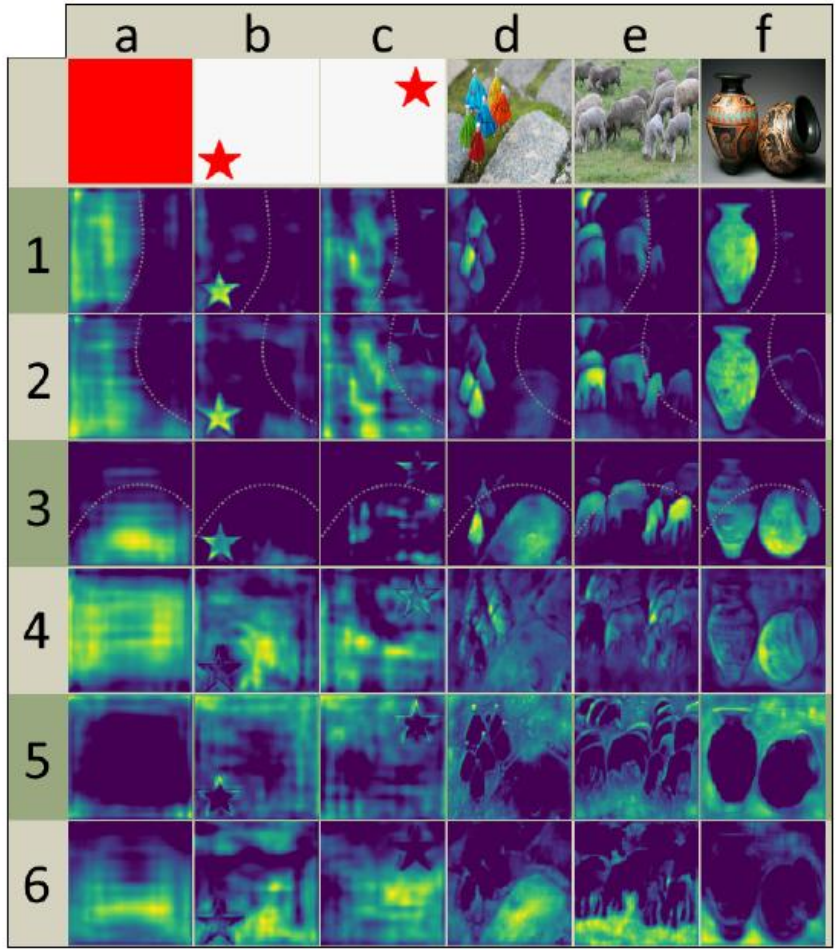


Fig. 5: **Prototype Behavior** The activations of the same six prototypes (y axis) across different images (x axis). Prototypes 1-3 respond to objects to one side of a soft, implicit boundary (marked with a dotted line). Prototype 4 activates on the bottom-left of objects (for instance, the bottom left of the umbrellas in image d); prototype 5 activates on the background and on the edges between objects; and prototype 6 segments what the network perceives to be the ground in the image. These last 3 patterns are most clear in images d-f.

YOLACT learns how to localize instances on its own via different activations in its prototypes.

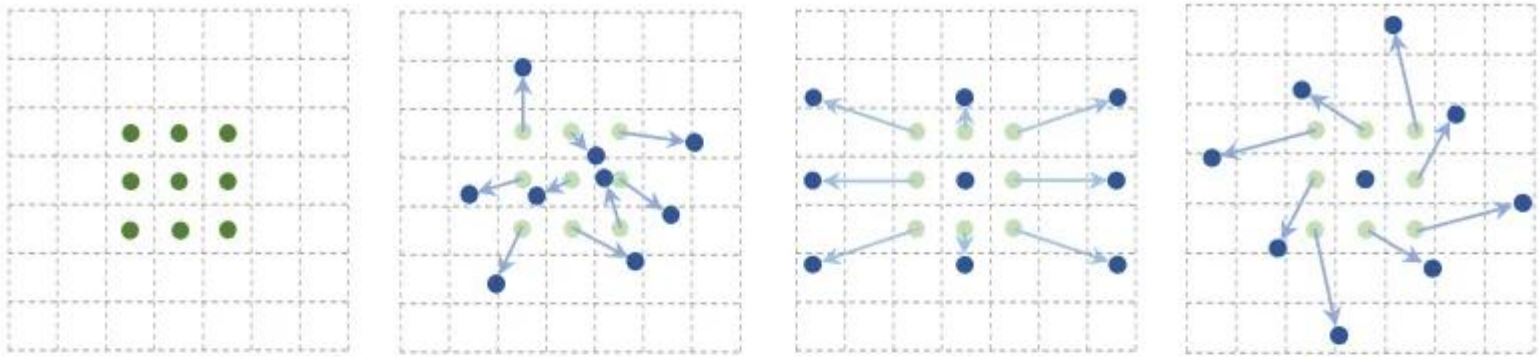
To see how this is possible, first note that the prototype activations for the solid red image (image a) in Figure 5 are actually not possible in an FCN without padding. Because a convolution outputs to a single pixel, if its input everywhere in the image is the same, the result everywhere in the *conv* output will be the same. On the other hand, the consistent rim of padding in modern FCNs like ResNet gives the network the ability to tell how far away from the image's edge a pixel is. Conceptually, one way it could accomplish this is to have multiple layers in sequence spread the padded 0's out from the edge toward the center (e.g., with a kernel like $[1, 0]$). This means ResNet, for instance, *is inherently translation variant*, and our method makes heavy use of that property (images b and c exhibit clear translation variance).

We observe many prototypes to activate on certain "partitions" of the image. That is, they only activate on objects on one side of an implicitly learned boundary. In Figure 5, prototypes 1-3 are such examples. By combining these partition maps, the network can distinguish between different (even overlapping) instances of the same semantic class; e.g., in image d, the green umbrella can be separated from the red one by subtracting prototype 3 from prototype 2.

YOLOACT++

- Fast Mask Re-Scoring Network
- Deformable Convolution with Intervals
  - having deformable convolutions in four different configurations:
  - (1) in the last 10 ResNet blocks, (2) in the last 13 ResNet blocks, (3) in the last 3 ResNet stages with an interval of 3 (i.e., skipping two ResNet blocks in between; total 11 deformable layers), and (4) in the last 3 ResNet stages with an interval of 4 (total 8 deformable layers).
- Optimized Prediction Head

Deformable Convolution with Intervals



(a) (b) (c) (d)

Figure 1: Illustration of the sampling locations in 3×3 standard and deformable convolutions. (a) regular sampling grid (green points) of standard convolution. (b) deformed sampling locations (dark blue points) with augmented offsets (light blue arrows) in deformable convolution. (c)(d) are special cases of (b), showing that the deformable convolution generalizes various transformations for scale, (anisotropic) aspect ratio and rotation.

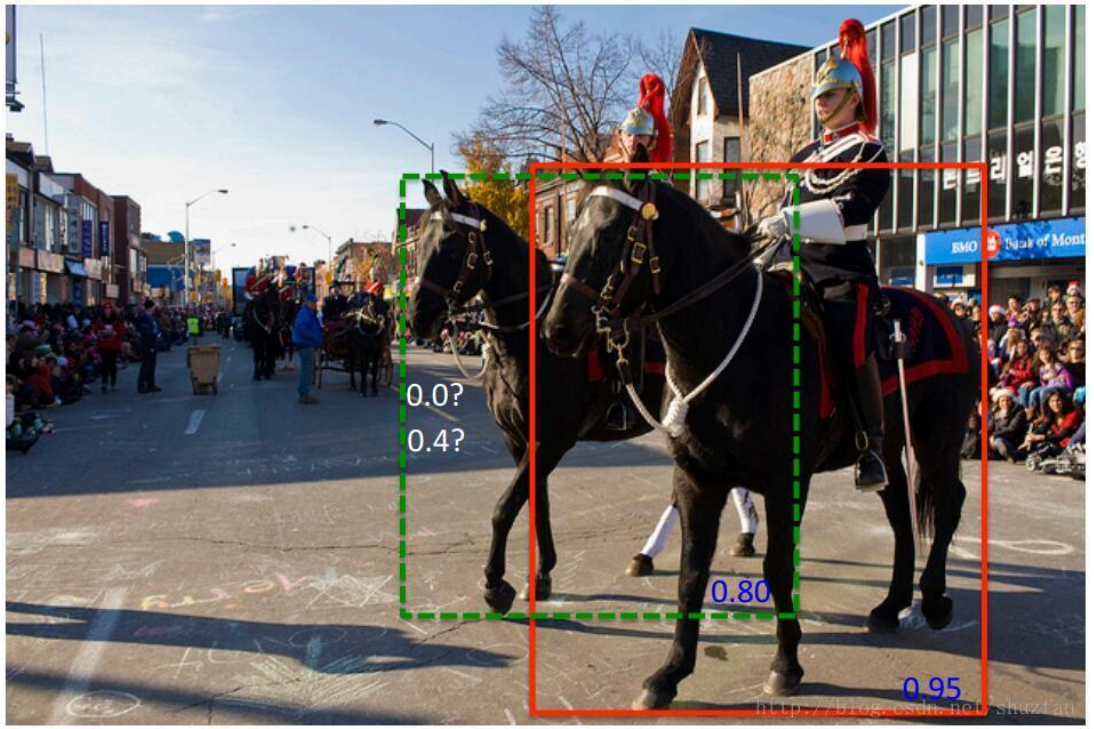
Appendix

SOFT NMS AND OTHER NMS VARIANTS

Improving Object Detection With One Line of Code

- Problem of current NMS
  - Hard to decide threshold to remove overlapped blocks
- Solution
  - Don’t remove it all
  - Use confidence level instead of directly remove highly overlapped blocks

- Red (0.95) Greed (0.8)
- Suppress 0.8 block or assign a lower score, 0.4?



- Original NMS score

$$s_i = \begin{cases} s_i, & \text{iou}(\mathcal{M}, b_i) < N_t \\ 0, & \text{iou}(\mathcal{M}, b_i) \geq N_t \end{cases}$$

- Proposed
  - Linear weighting (sudden penalty over N_t)

$$s_i = \begin{cases} s_i, & \text{iou}(\mathcal{M}, b_i) < N_t \\ s_i(1 - \text{iou}(\mathcal{M}, b_i)), & \text{iou}(\mathcal{M}, b_i) \geq N_t \end{cases},$$

  - Gaussian weighting (Continuous)

$$s_i = s_i e^{-\frac{\text{iou}(\mathcal{M}, b_i)^2}{\sigma}}, \forall b_i \notin \mathcal{D}$$

M為當前得分最高框，b_i 為待處理框，b_i 和M的IOU越大，b_i 的得分s_i 就下降的越厲害

Input : $\mathcal{B} = \{b_1, .., b_N\}, \mathcal{S} = \{s_1, .., s_N\}, N_t$
$\mathcal{B}$ is the list of initial detection boxes
$\mathcal{S}$ contains corresponding detection scores
N_t is the NMS threshold

begin
  $\mathcal{D} \leftarrow \{\}$
  while $\mathcal{B} \neq$ *empty* **do**
    $m \leftarrow$ argmax $\mathcal{S}$
    $\mathcal{M} \leftarrow b_m$
    $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{M}; \mathcal{B} \leftarrow \mathcal{B} - \mathcal{M}$
    for b_i *in* $\mathcal{B}$ **do**
      if $iou(\mathcal{M}, b_i) \geq N_t$ **then**
        $\mathcal{B} \leftarrow \mathcal{B} - b_i; \mathcal{S} \leftarrow \mathcal{S} - s_i$
      end NMS
      $s_i \leftarrow s_i f(iou(\mathcal{M}, b_i))$ Soft-NMS
    end
  end
  return $\mathcal{D}, \mathcal{S}$
end

- 1% improvement without extra training and computation burden

Method	Training data	Testing data	AP 0.5:0.95	AP @ 0.5	AP small	AP medium	AP large
R-FCN [15]	coco-minival	test-dev	31.1	52.5	14.4	34.9	43.0
R-FCN + S-NMS G	coco-minival	test-dev	**32.4**	**53.4**	**15.2**	**36.1**	**44.3**
R-FCN + S-NMS L	coco-minival	test-dev	32.2	**53.4**	15.1	36.0	44.1
F-RCNN [21]	coco-minival	test-dev	24.4	45.7	7.9	26.6	37.2
F-RCNN + S-NMS G	coco-minival	test-dev	**25.5**	46.6	**8.8**	**27.9**	**38.5**
F-RCNN + S-NMS L	coco-minival	test-dev	**25.5**	**46.7**	**8.8**	**27.9**	38.3
R-FCN [15]	0712 trainval	07 test	49.8	79.4	-	-	-
R-FCN + S-NMS G	0712 trainval	07 test	51.4	**80.0**	-	-	-
R-FCN + S-NMS L	0712 trainval	07 test	**51.5**	**80.0**	-	-	-
F-RCNN [21]	07 trainval	07 test	37.7	70.0	-	-	-
F-RCNN + S-NMS G	07 trainval	07 test	**39.4**	**71.2**	-	-	-
F-RCNN + S-NMS L	07 trainval	07 test	**39.4**	**71.2**	-	-	-

Table 1. Results on MS-COCO test-dev set and PASCAL VOC 2007 test set for off-the-shelf standard object detectors, R-FCN and Faster-RCNN (F-RCNN) which use NMS as baseline and our proposed Soft-NMS method. G denotes Gaussian weighting and L denotes linear weighting.

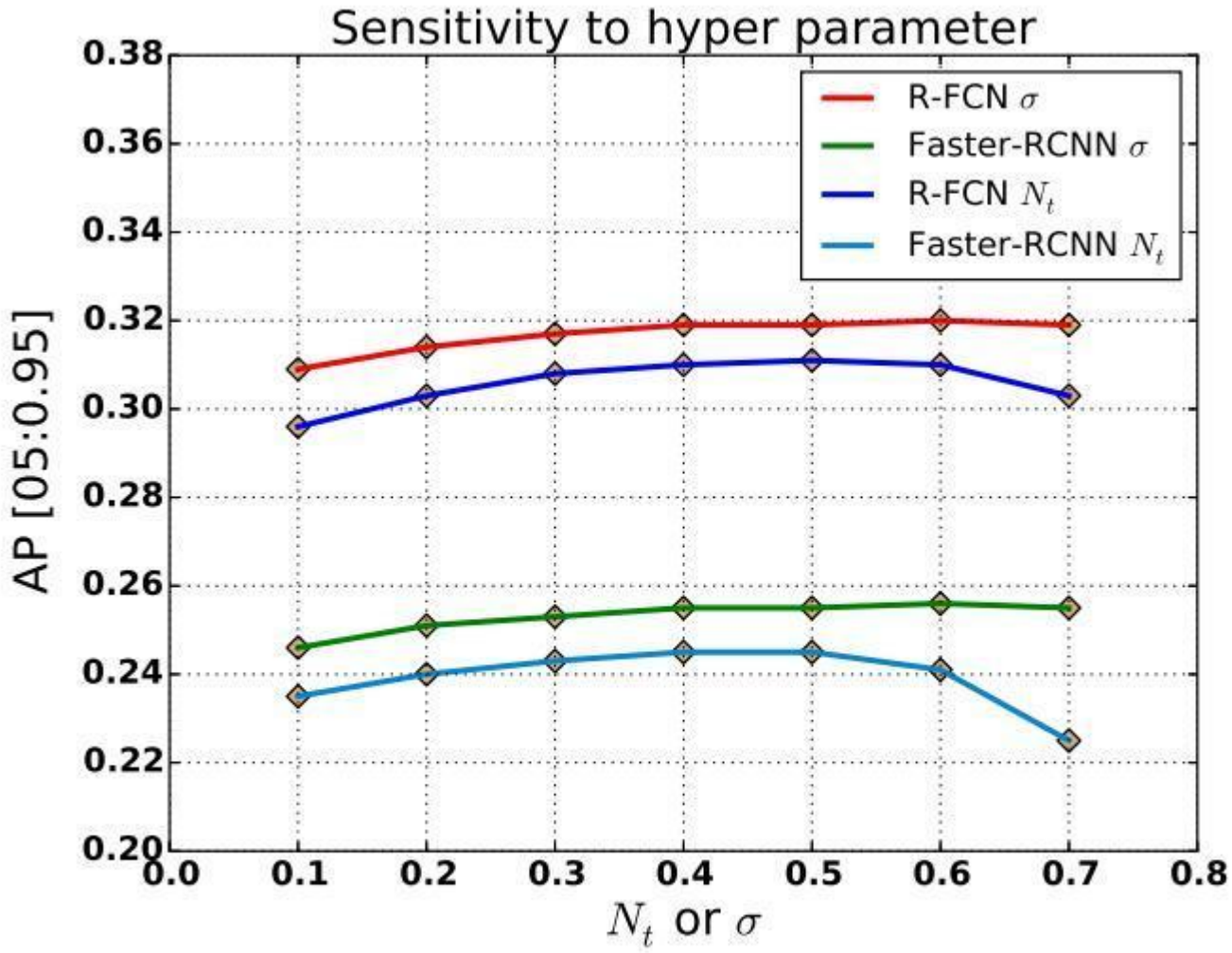
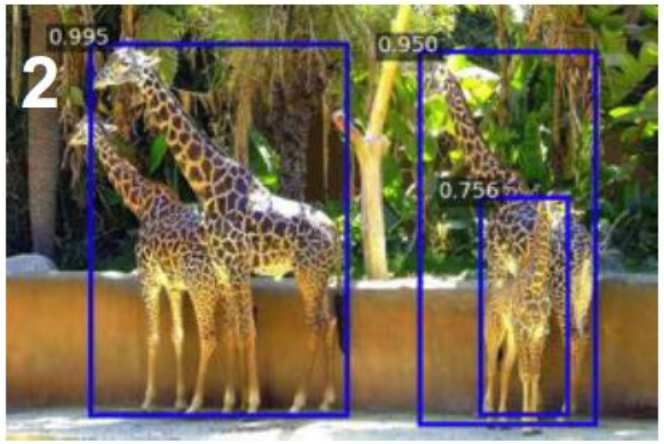
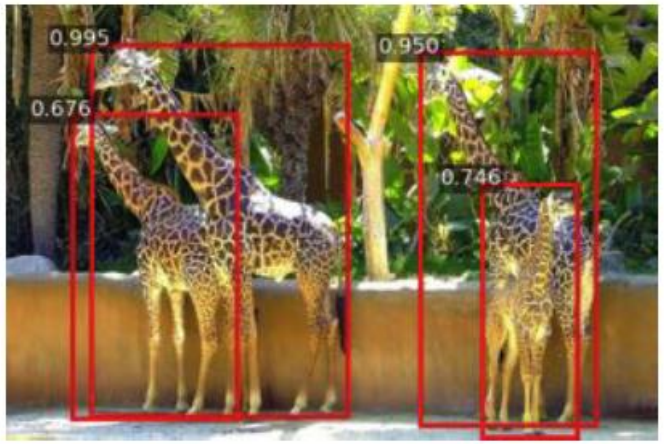
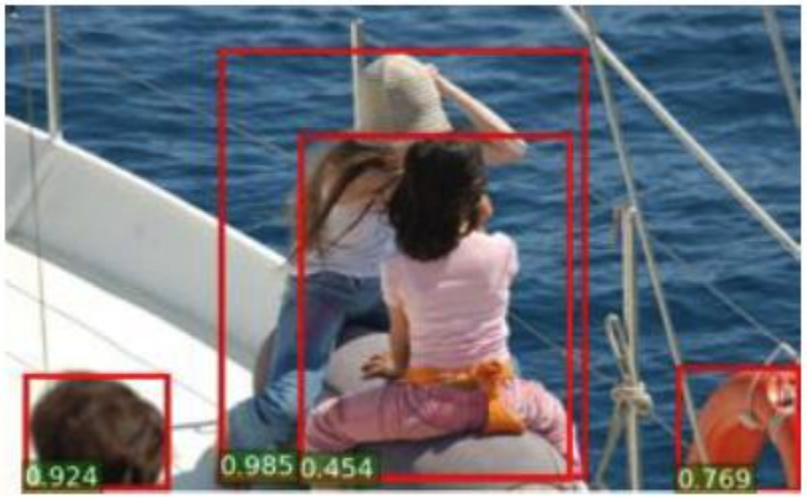


Figure 4. R-FCN Sensitivity to hyper parameters σ (Soft-NMS) and N_t (NMS)

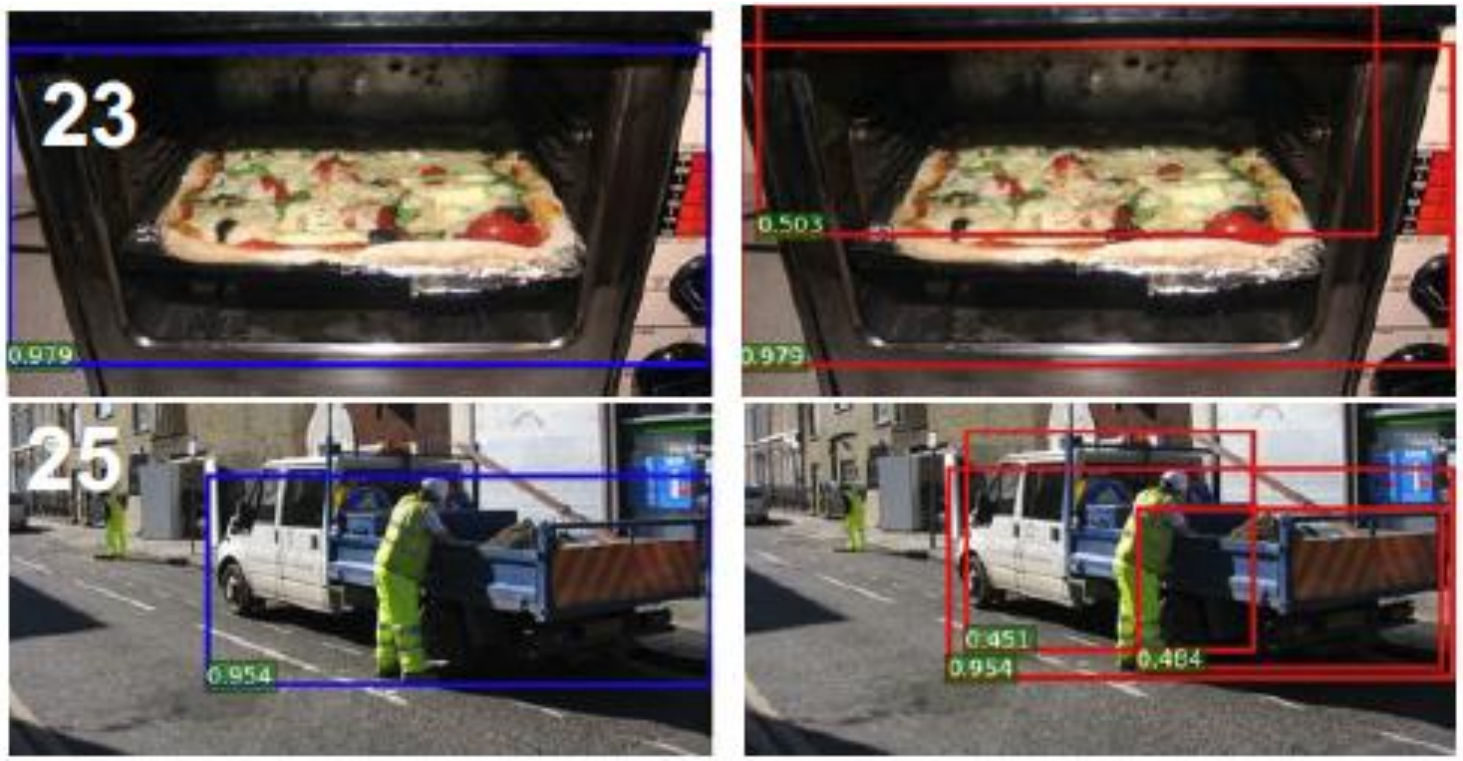
Success case





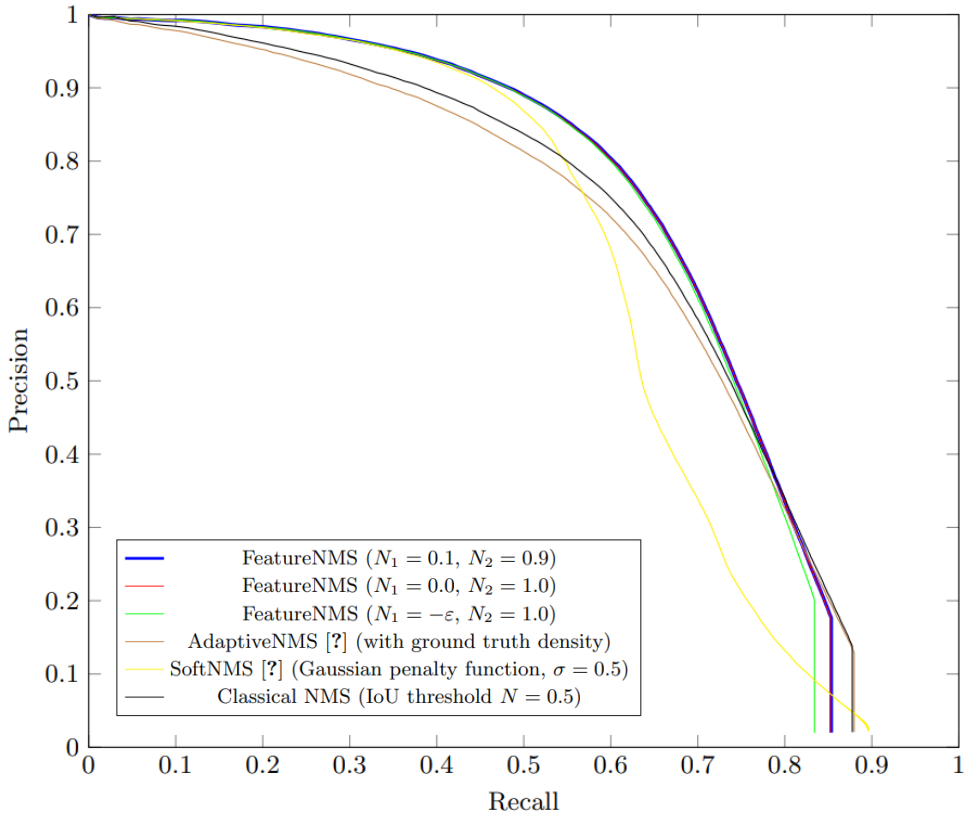


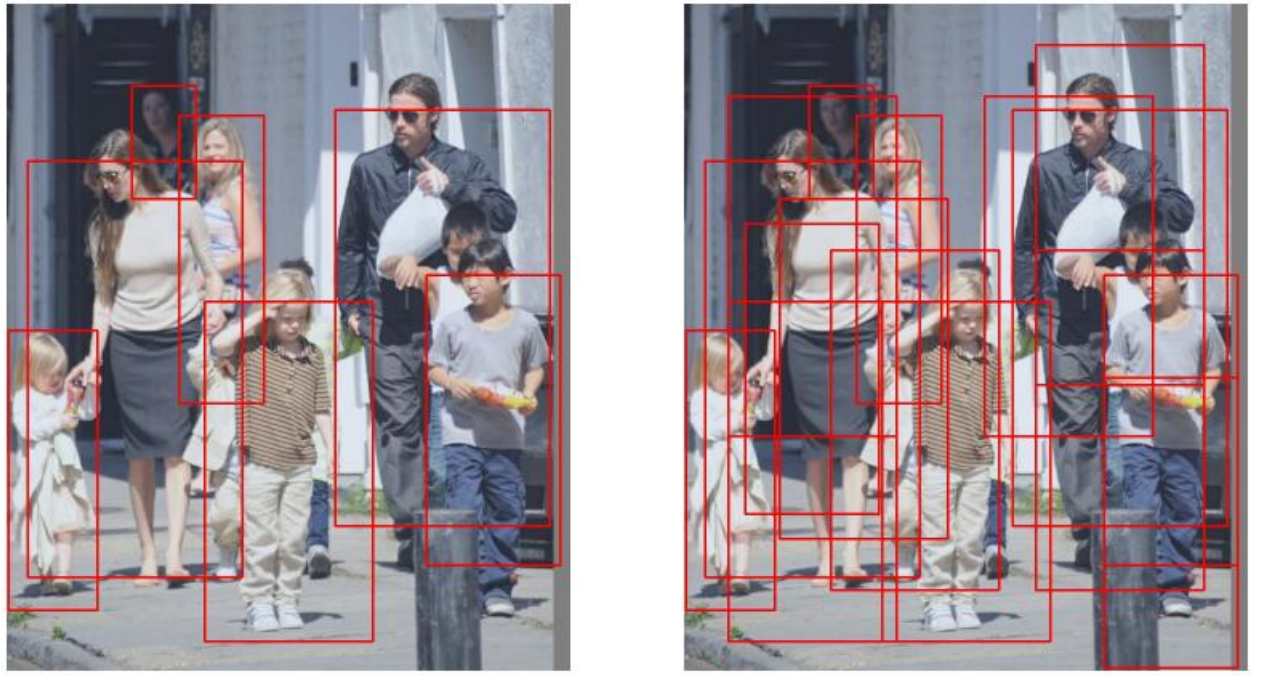
Failure case



Feature NMS

- For heavily overlapped case, compare feature vector as well when in a fuzzy region





FeatureNMS and classical NMS

Bag of Freebies for Training Object Detection Neural Networks

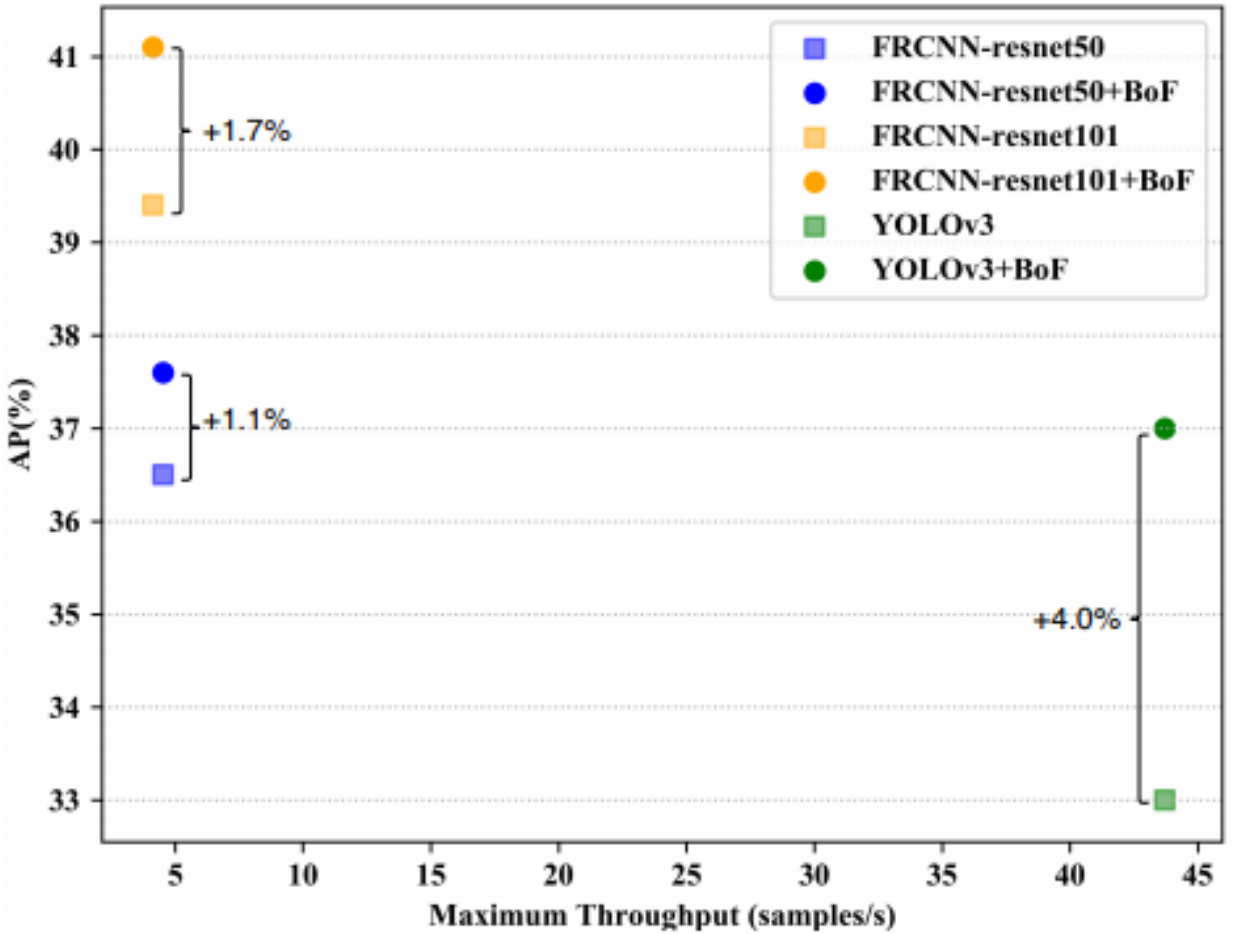


Figure 1. **Bag of Freebies dramatically improves object detector performances with no throughput penalty.**

Tricks from Image Classification

- Learning rate warm up heuristic
  - For extremely large mini-batch size
  - For object detection, a large amount of anchor size(up to 30k) is effectively contributing to batch size implicitly
- Label smoothing
  - modifies the hard ground truth labeling in cross entropy loss
- Mixup
  - alleviate adversarial perturbation
- Cosine annealing strategy for learning rate
  - Improve traditional step policy

Tricks for Object Detection

- Visually Coherent Image Mixup for Object Detection

Figure 2. Mix-up visualization of image classification with typical mixup ratio at 0.1 : 0.9.

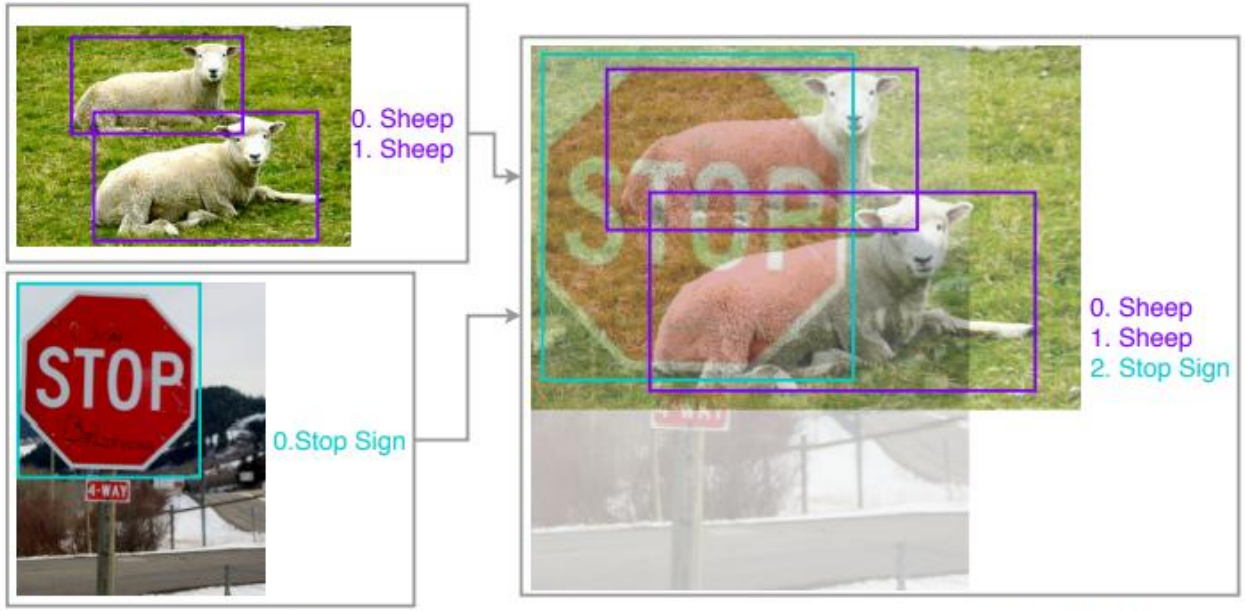


Figure 3. Geometry preserved alignment of mixed images for object detection. Image pixels are blended, object labels are merged with respect to generated new image.

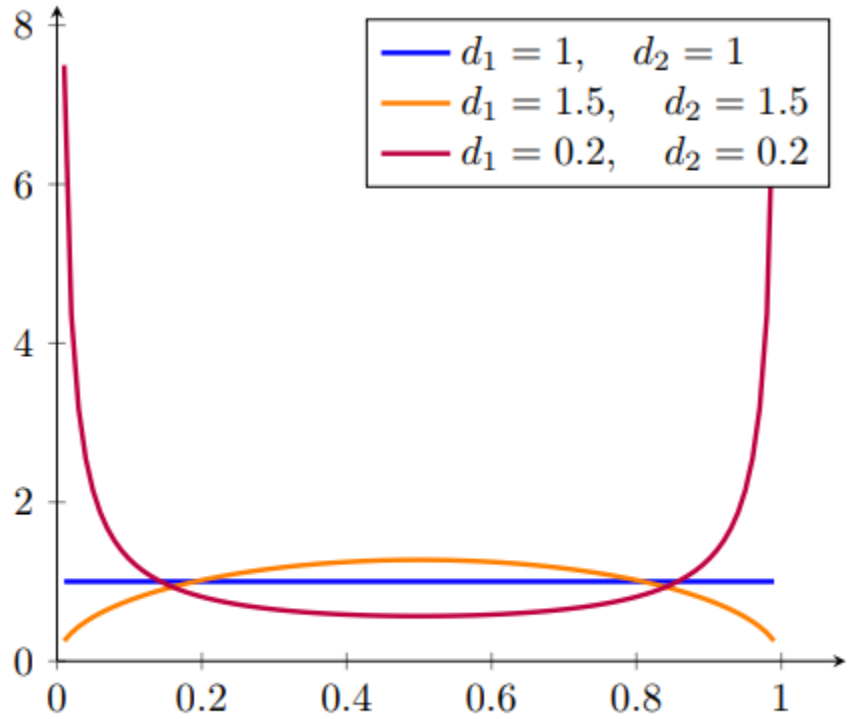


Figure 4. Comparison of different random weighted mix-up sampling distributions. Red curve indicate the typical mixup ratio used in image classification.

Model	mAP @ 0.5
baseline	81.5
0.5:0.5 evenly	83.05
beta(1.0, 1.0), weighted loss	83.48
beta(1.5, 1.5), weighted loss	83.54

Table 1. Effect of various mix-up approaches, validated with YOLOv3 [16] on Pascal VOC 2007 test set. **Weighted loss** indicates the overall loss is the summation of multiple objects with ratio 0 to 1 according to image blending ratio they belong to in the original training images.

Classification Head Label Smoothing

- smooth the ground truth distribution
  - K is the total number of classes and ε is a small constant
  - q is often a one-hot distribution, where the correct class has probability one while all other classes have zero

$$q_i' = (1 - \epsilon)q_i + \frac{\epsilon}{K},$$

- For sigmoid outputs in 0 to 1.0 as in YoLo v3
  - label smoothing is even simpler by correcting the upper and lower limit of the range of targets as above

Data Pre-processing

- Random geometry transformation
  - random cropping (with constraints), random expansion, random horizontal flip and random resize (with random interpolation)
  - More sensitive in object detection than in image classification
- Random color jittering including brightness, hue, saturation, and contrast
- Spatial variation in object detection
  - Single stage method: detection results proportional to spatial shape of an input image (due to anchor ?)
  - Two stage method: ROI candidates generated and sampled (substitute random cropping)

Training Scheduler Revamping

- Current approach use step decay
  - Faster R-CNN: x0.1 at 60K
  - YOLOv3: x0.1 at 40K/45K
  - Sharp transitions cause the optimizer to re-stabilize the learning momentum in the next few iterations
- cosine learning rate
- Warm up learning rate

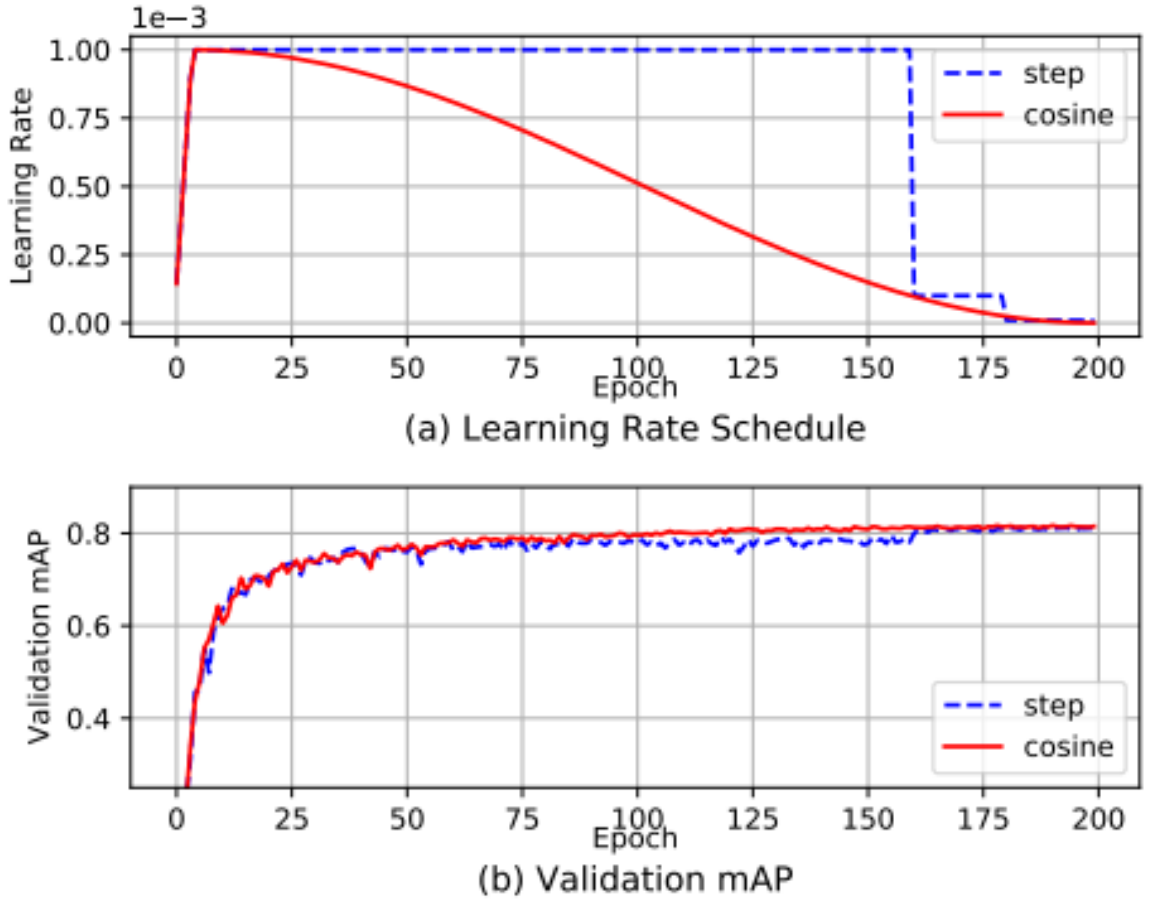


Figure 6. Visualization of learning rate scheduling with warm-up enabled for YOLOv3 training on Pascal VOC. (a): cosine and step schedules for batch size 64. (b): Validation mAP comparison curves using step and cosine learning scheduler.

Synchronized Batch Normalization

- For multi-GPUs
- Not a problem for image classification
  - batch size per device is usually large enough to obtain good statistics
  - But hurt object detection due to small size (1 per GPU)

Random shapes training for single-stage object detection networks

- a mini-batch of N training images is resized to N × 3 × H × W,
  - where H and W are multipliers of common divisor D = randint(1, k). For example, we use H = W ∈ {320, 352, 384, 416, 448, 480, 512, 544, 576, 608} for YOLOv3 training

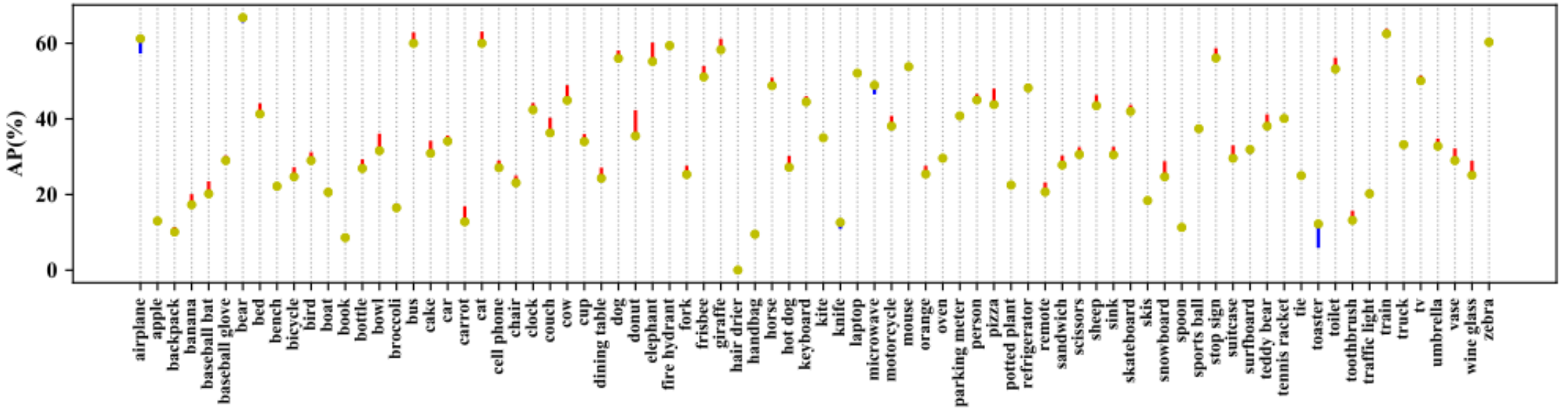


Figure 8. **COCO 80 category AP analysis with YOLOv3 [16].** Red lines indicate performance gain using BoF, while blue lines indicate performance drop.

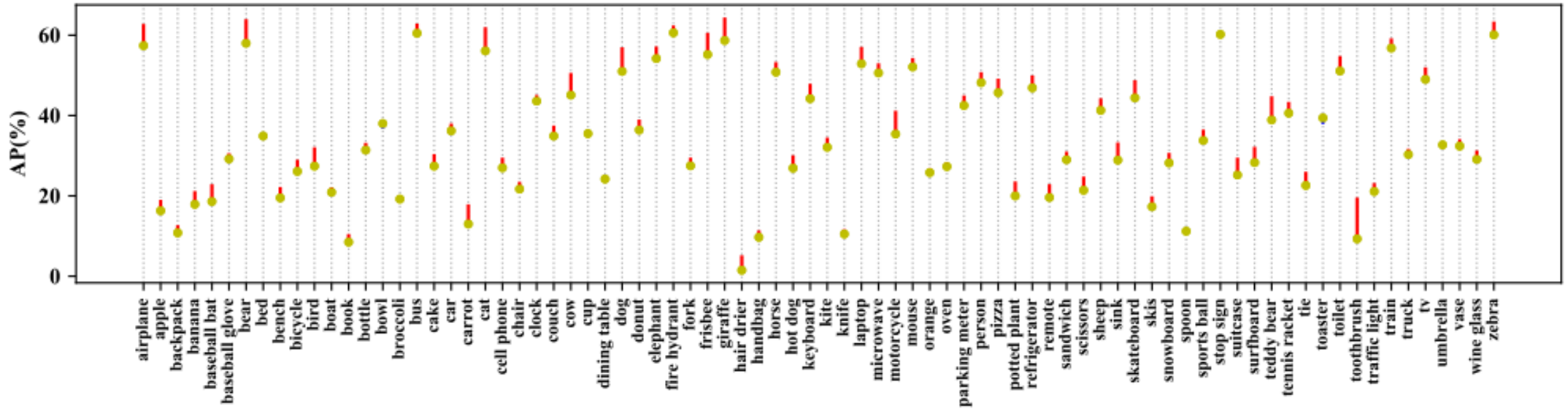
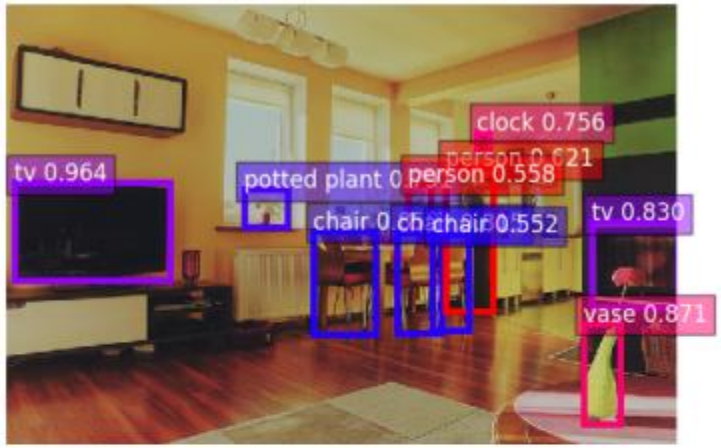
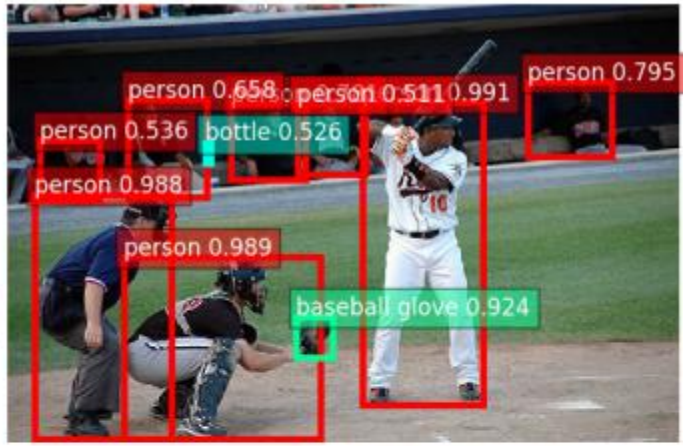
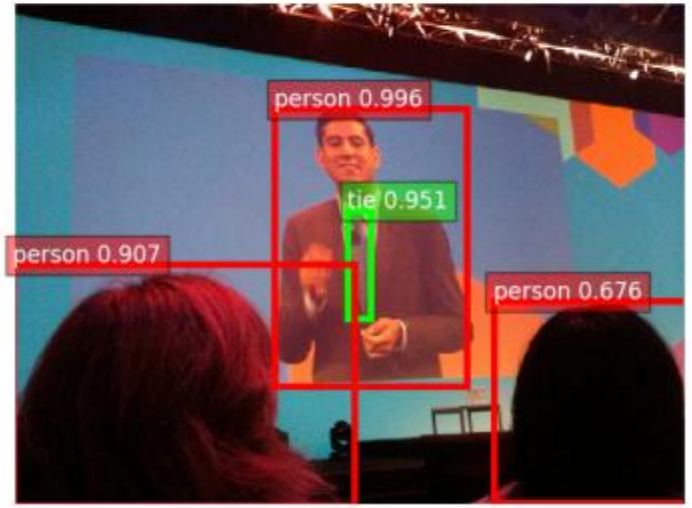


Figure 9. **COCO 80 category AP analysis with Faster-RCNN resnet50 [17].** Red lines indicate performance gain using BoF, while blue lines indicate performance drop.


clock 0.756
tv 0.964
potted plant 0
person 0.558
chair 0.552
tv 0.830
vase 0.871


person 0.658
person 0.795
person 0.536
bottle 0.526
person 0.988
person 0.989
baseball glove 0.924


person 0.988
person 0.765
person 0.897
person 0.992
person 0.939
person 0.918
person 0.981
person 0.969


person 0.996
tie 0.951
person 0.907
person 0.676

Model	mAP	Delta
- data augmentation	64.26	-15.99
baseline	80.25	0
+ synchronize BN	80.81	+0.56
+ random training shapes	81.23	+0.98
+ cosine lr schedule	81.69	+1.44
+ class label smoothing	82.14	+1.89
+ mixup	**83.68**	**+3.43**

Table 2. Training Refinements on YOLOv3, evaluated at 416×416 on Pascal VOC 2007 test set.

Model	mAP	Delta
- data augmentation	77.61	-0.16
baseline	77.77	0
+ cosine lr schedule	79.59	+1.82
+ class label smoothing	80.23	+2.46
+ mixup	**81.32**	**+3.55**

Table 3. Training Refinements on Faster-RCNN, evaluated at 600 × 1000 on Pascal VOC 2007 test set.

g of freebies(BoF), evaluated on MS COCO [11] 2017 val set.

Model	Orig $AP^{0.5:0.95}_{bbox}$	**Our BoF** $AP^{0.5:0.95}_{bbox}$	Absolute delta
Faster-RCNN R50 [5]	36.5	**37.6**	**+1.1**
Faster-RCNN R101 [5]	39.4	**41.1**	**+1.7**
YOLOv3 [16]	33.0	**37.0**	**+4.0**

Table 4. Overview of improvements achieved by applying bag of freebies(BoF), evaluated on MS COCO [11] 2017 val set.

	-Mixup YOLO3	+Mixup YOLO3
-Mixup darknet53	35.0	35.3
+Mixup darknet53	36.4	37.0

Table 5. Combined analysis of impacts of mix-up methodology for pre-trained image classification and detection network.

	-Mixup FRCNN	+Mixup FRCNN
-Mixup R101	39.9	40.1
+Mixup R101	40.1	41.1

Table 6. Combined analysis of impacts of mix-up methodology for pre-trained image classification and detection network.