Object Detection

CoE197Z/EE298Z (Deep Learning)

Rowel Atienza

University of the Philippines

rowel@eee.upd.edu.ph

Outline

- Single-stage detectors
    - SSD
    - RetinaNet
    - Anchor-free
        - FSAF
        - FCOS
- Two-stage detectors
    - Fast RCNN
    - Faster RCNN
    - FPN

Single-Shot Detector (SSD)

Liu, Wei, et al. "Ssd: Single shot multibox detector." European conference on computer vision. Springer, Cham, 2016.

Detection = Identification + Localization

Objective : Identify and localize objects in an image

Identify: Soda can

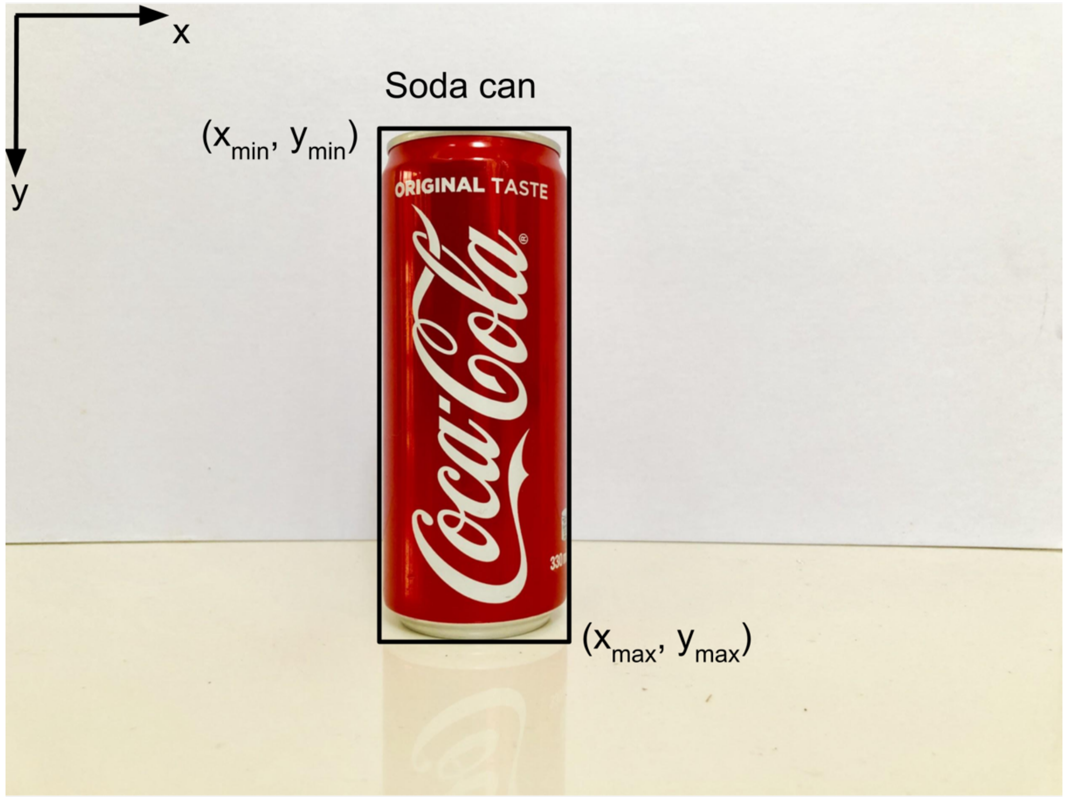
- Also known as **class id**

Localize: (x_{min}, y_{min}) , (x_{max}, y_{max})

- Also known as **bounding box**

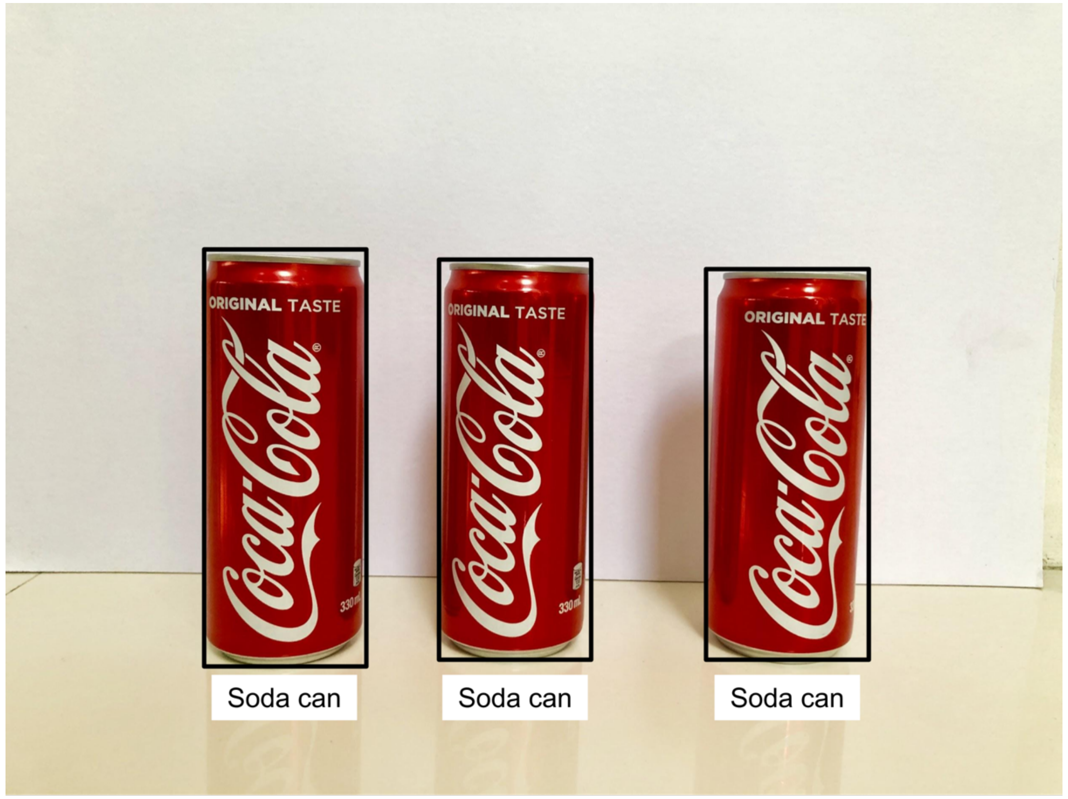
Coordinate system

- Origin at upper-left corner



Detection

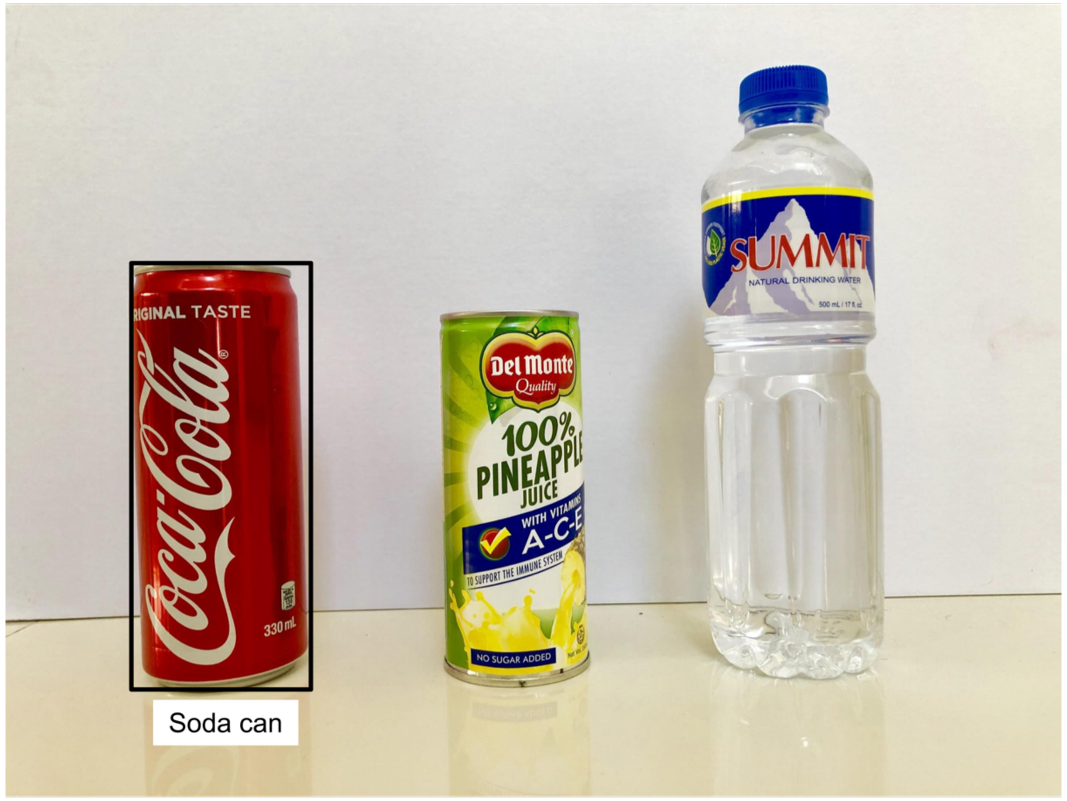
Can detect multiple objects at the same time



Detection

Objects not trained on will be ignored

Example: Detector of Soda ignores all other objects

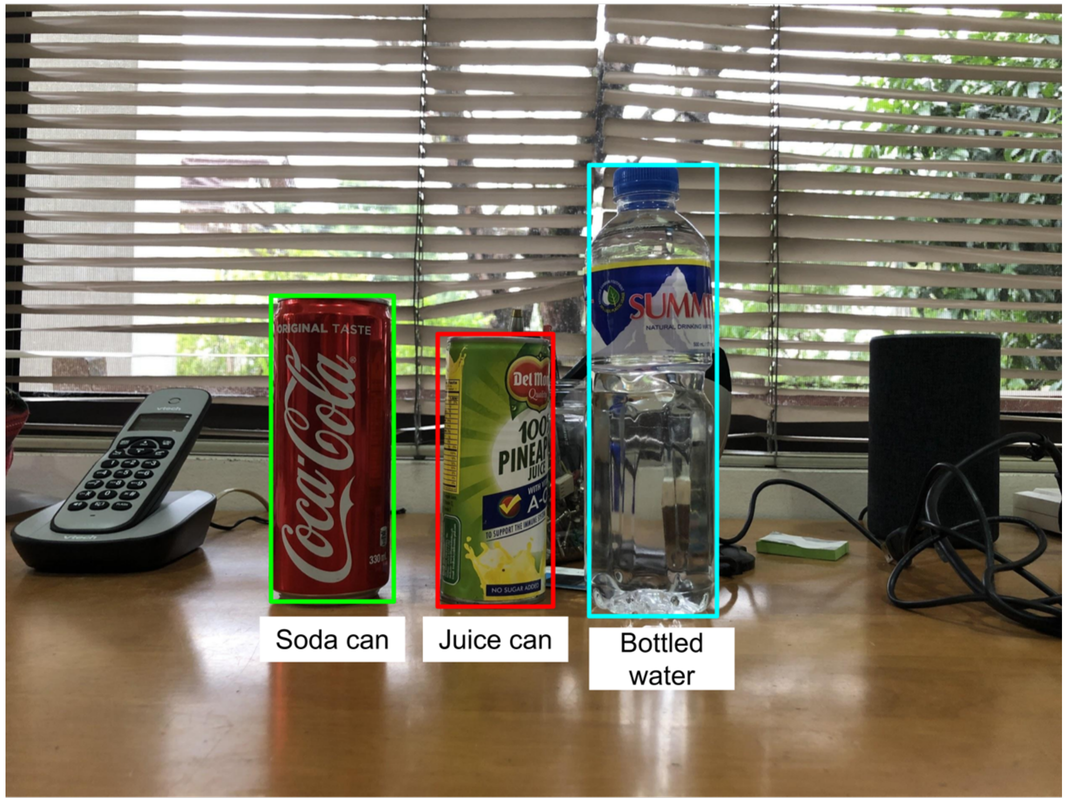


Detection

Can be trained to detect multiple objects

Even at varying conditions

- Illumination
- Distortion
- Orientation



Single-Shot Detection – Key Idea

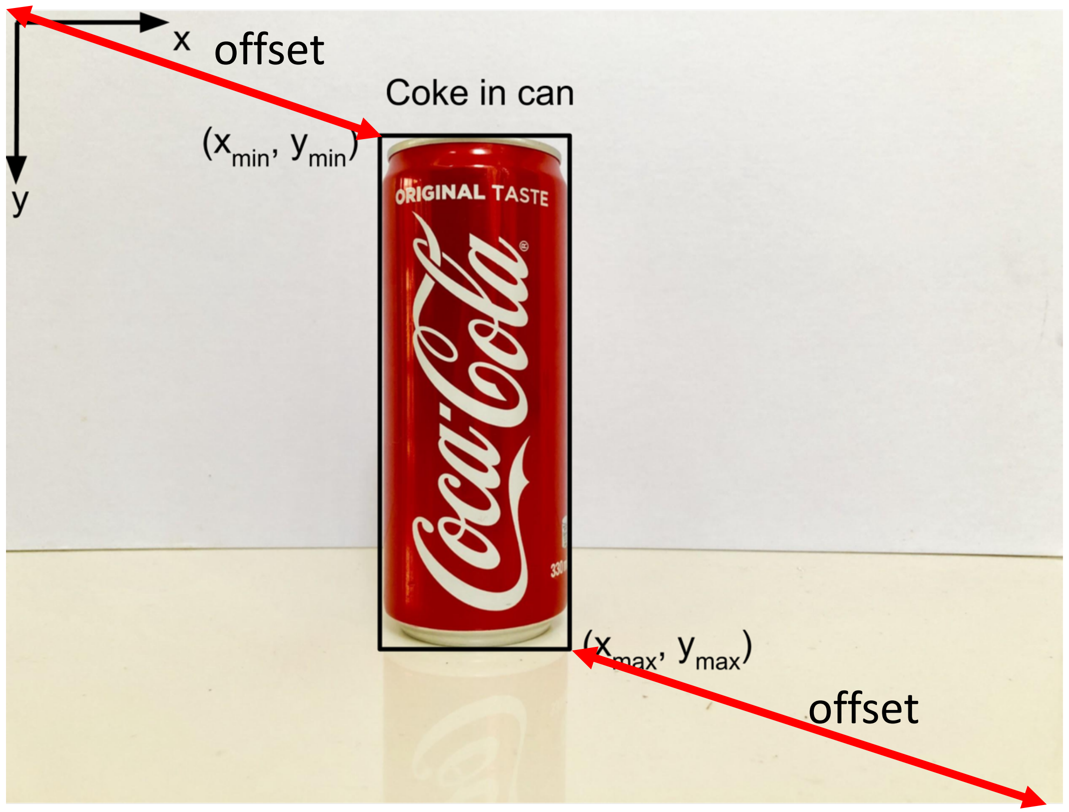
- Divide an image into regions
  - A region is called an **anchor box**
- For each region, determine its class (background or object)
- For each region, determine the offset of the bounding box
  - Instead of bounding box coordinates (high variance), the offsets with respect to anchor box is estimated (low variance)
  - Predict offsets only if the region is classified as non-background

What is the size of anchor box?

If the entire image is used:

Offsets: (x_{min}, y_{min}) and $(x_{max} - w, y_{max} - h)$

Assume 640 x 480 RGB Image

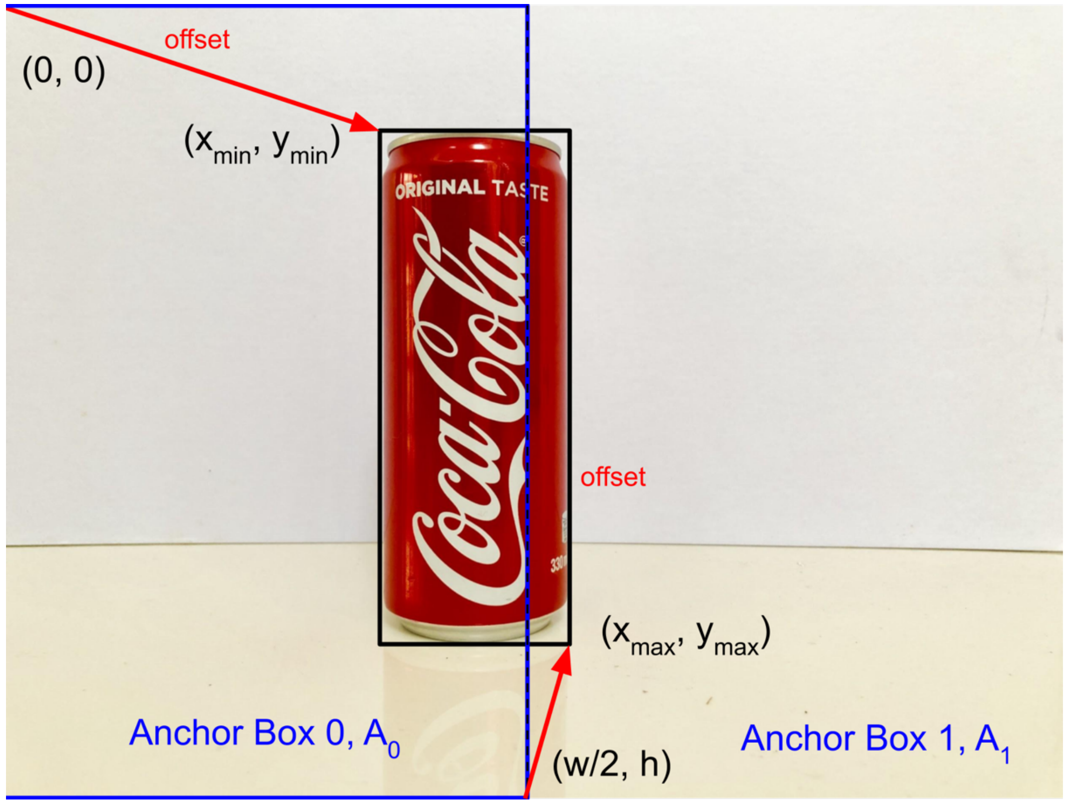


Anchor : 2 x 1

Offset: (x_{min}, y_{min}) and $(x_{max} - {}^{w}/_{2}\,, y_{max} - h)$

This is smaller

Anchor box size: $({}^{w}/_{2}\,, h)$

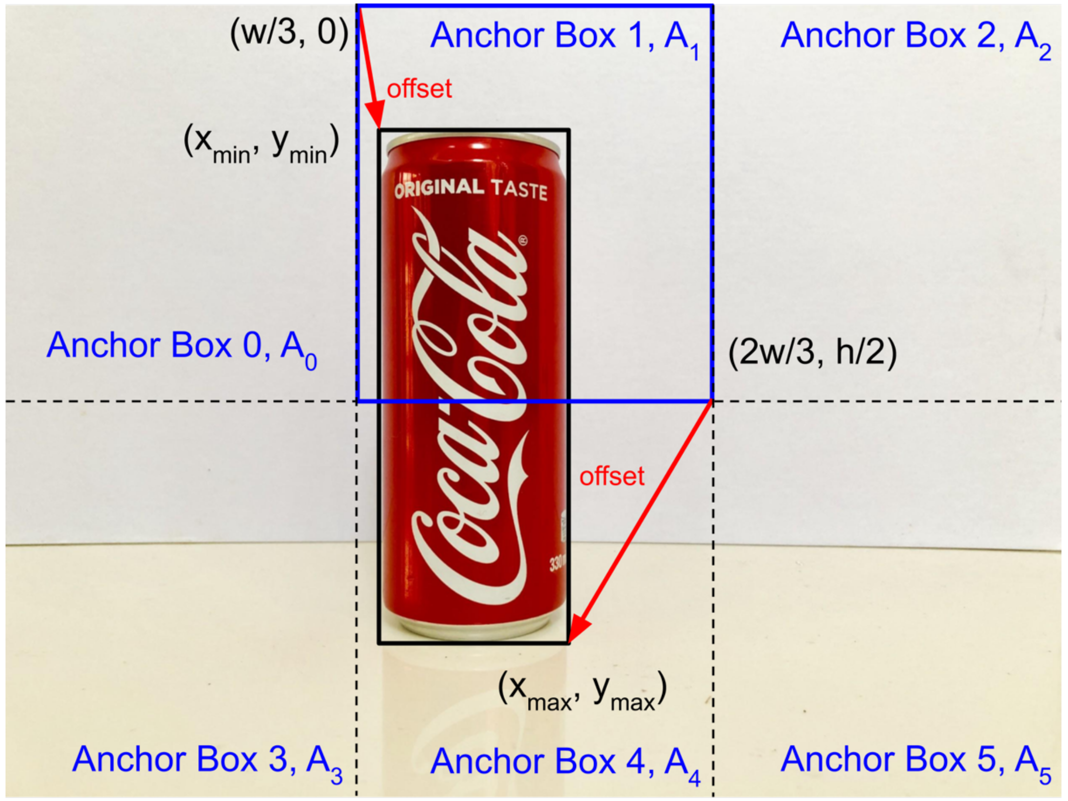


Anchor : 3 x 2

Offsets: $(x_{min} - {}^{w}/_{3}, y_{min})$ and $(x_{max} - {}^{2w}/_{3}, y_{max} - {}^{h}/_{2})$

This is the smallest so far

Anchor box size: $({}^{w}/_{3}, {}^{h}/_{2})$



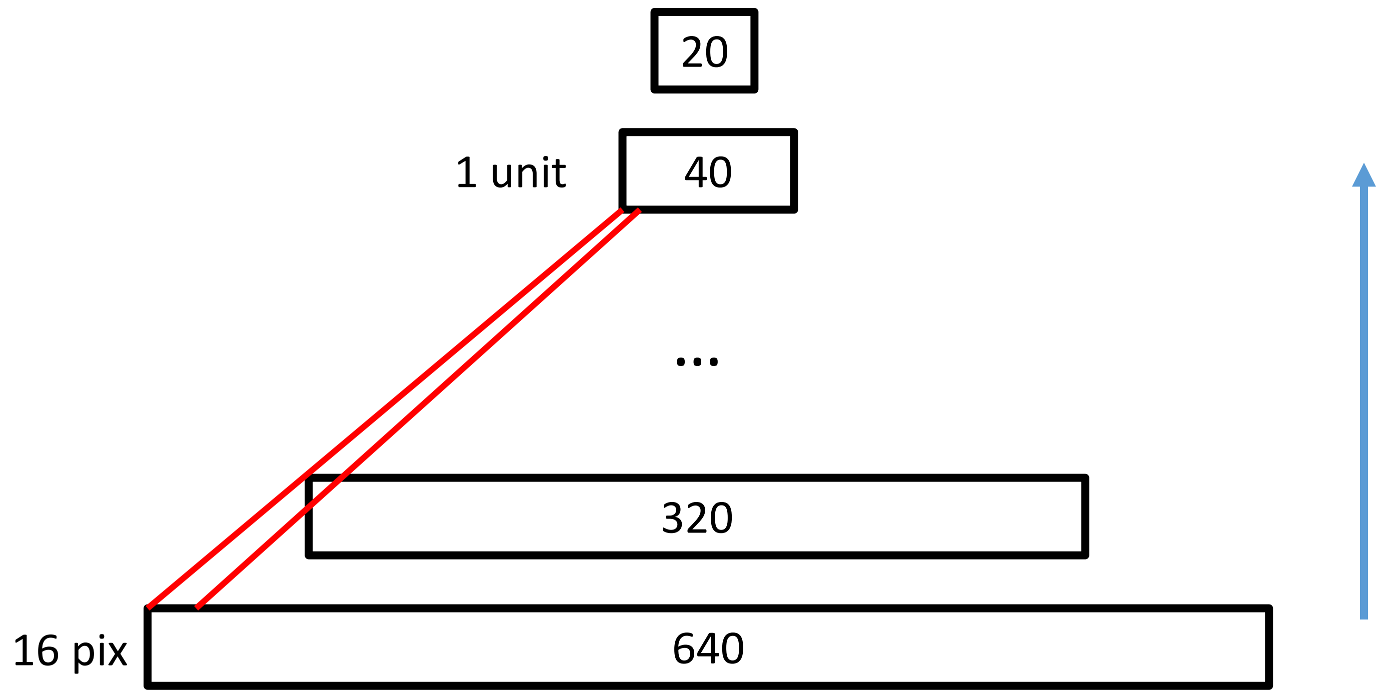
Available anchor box sizes: Total 1608

2×1 grid of anchor boxes each with dimensions $({}^{w}/_{2}\,, h)$.
3×2 grid of anchor boxes each with dimensions $({}^{w}/_{3}\,, {}^{h}/_{2})$
5×4 grid of anchor boxes each with dimensions $({}^{w}/_{5}\,, {}^{h}/_{4})$
10×8 grid of anchor boxes each with dimensions $({}^{w}/_{10}\,, {}^{h}/_{8})$
20×15 grid of anchor boxes each with dimensions $({}^{w}/_{20}\,, {}^{h}/_{15})$
40×30 grid of anchor boxes each with dimensions $({}^{w}/_{40}\,, {}^{h}/_{30})$
In summary: $s = \left[\left(\frac{1}{2}, 1\right), \left(\frac{1}{3}, \frac{1}{2}\right), \left(\frac{1}{5}, \frac{1}{4}\right), \left(\frac{1}{10}, \frac{1}{8}\right), \left(\frac{1}{20}, \frac{1}{15}\right), \left(\frac{1}{40}, \frac{1}{30}\right)\right]$
In SSD, s is not used. Instead, linearly spaced scales from $[0.2, 0.9]$ is used

40×30 grid

With 40×30 grid, the anchor box covers a $\frac{640}{40}\times\frac{480}{30}$ or 16×16 pixels patch

Also known as the *receptive field*



Convolution and pooling (4x)
Backbone or base network
VGG, ResNet, ResNext

Aspect Ratio

Bounding box approximation can be better approximated if we allow different aspect ratios other than 1.0

Aspect Ratio: $a = \left[1, 2,\ 3, \frac{1}{2}, \frac{1}{3}\right]$

This results to anchor box dimensions:

$$(w_i,\ \ h_i) = \left(w s_{xj}\sqrt{a_i},\ \ h s_{yj}\frac{1}{\sqrt{a_i}}\right).$$

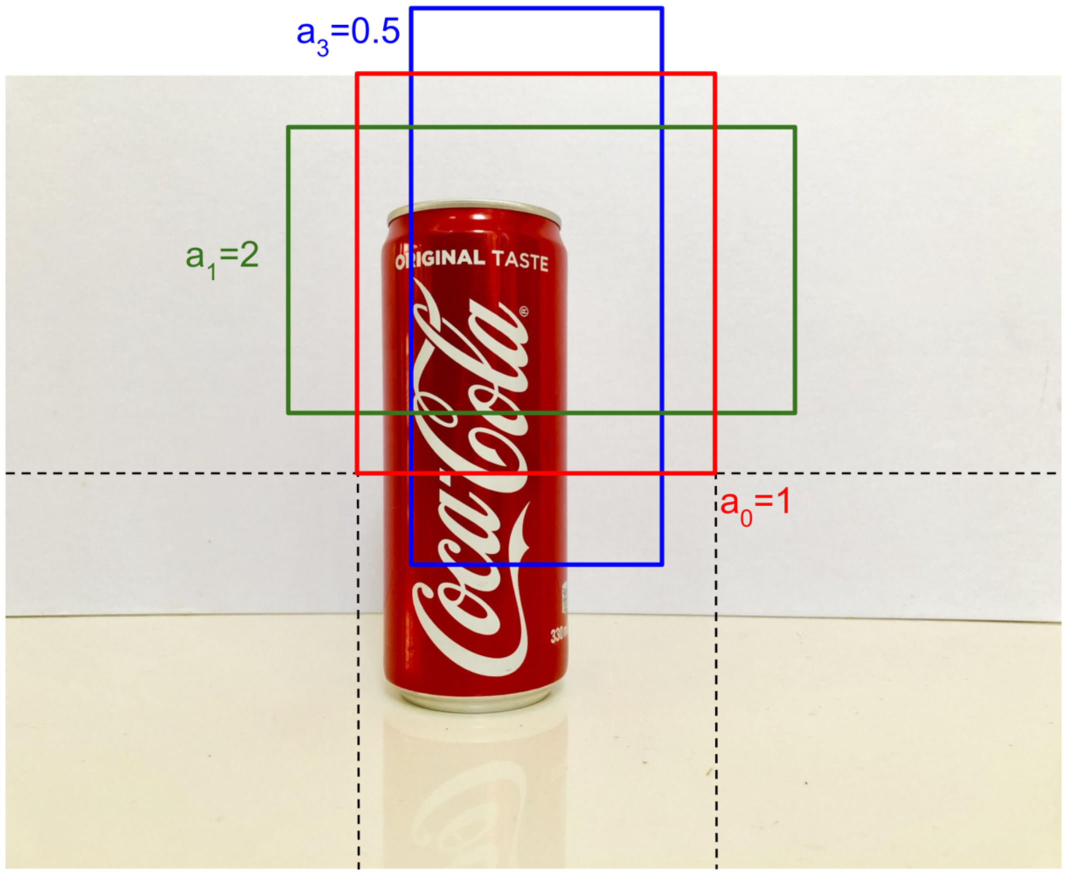
$\left(s_{xj},\ \ s_{yj}\right)$ is the $j - th$ scaling factor

New total number of anchor boxes: $1608 \times 5 = 8{,}040$

Further recommendations

SSD recommends an additional anchor box with dimensions $(w_5, h_5) = \left(w\sqrt{s_j s_{j+1}},\ \ h\sqrt{s_j s_{j+1}}\right)$ for aspect ratio of 1.

New total: 9,648



Predictions

We need Ground Truth Labels and Predictions

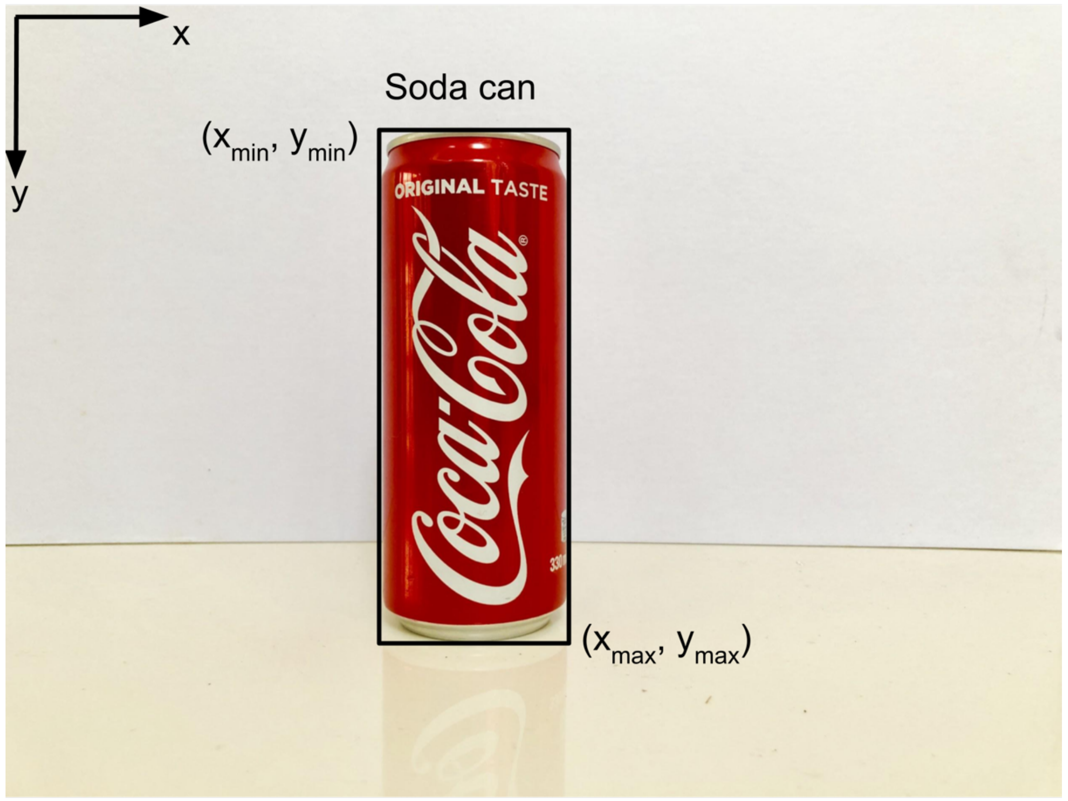
Ground Truth

Given for each object:

- 1 for class id
- 4 bounding box coordinates

Goal:

- Assign each anchor box ground truth labels:
  - Class id (0 if background)
  - 4 offsets wrt to bounding box coordinates
  - No need to assign offsets if class id is 0



Ground Truth

For each labelled object, assign its class id and bounding box offsets to one of over thousands of anchor boxes.

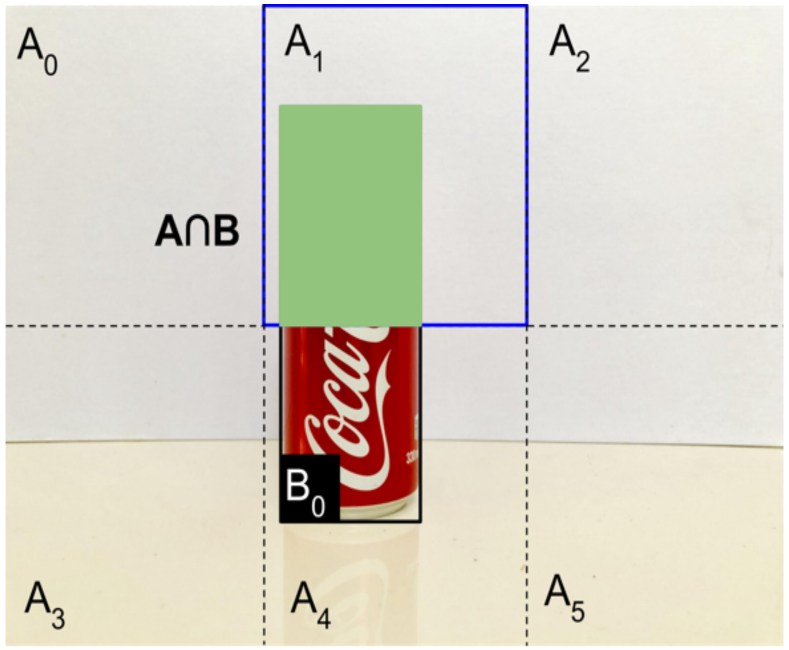
The rest of anchor boxes will be assigned class id of 0 (background). No need to assign bounding box offsets for background.

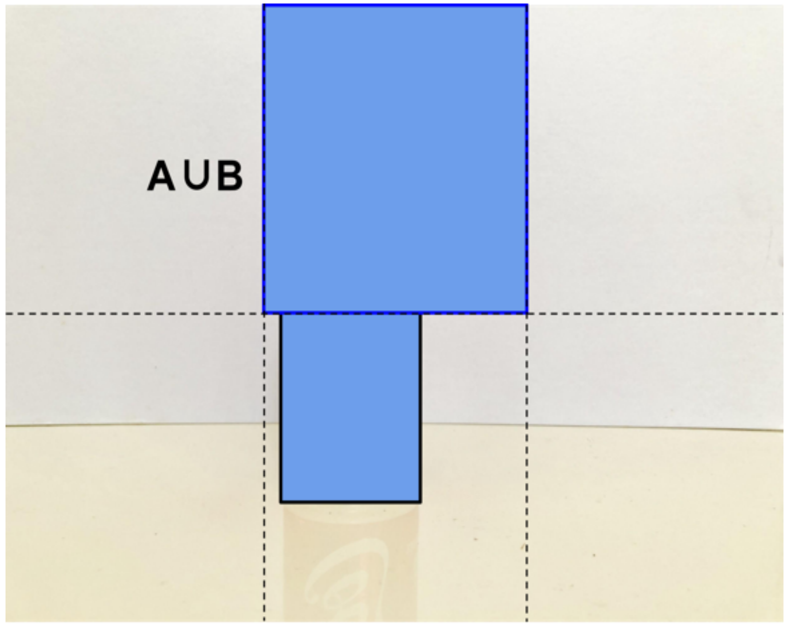
Ground truth based on IoU

Intersection over Union (IoU)

IoU is also known as *Jaccard index*

$$IoU = \frac{A \cap B}{A \cup B}$$





Which anchor box?

For each bounding box, search for the anchor box with the maximum IoU.

$$A_{j(gt)} = \max_{j} IoU\left(B_i, A_j\right)$$

Make this anchor box, the ground truth anchor box, $A_{j(gt)}$, with

- Class id equal to bounding box class id
- Offsets measured wrt to bounding box coordinates
- This anchor box is called **positive anchor box**

Therefore, for N objects, there are N ground truth or positive anchor boxes

What about the rest of anchor boxes?

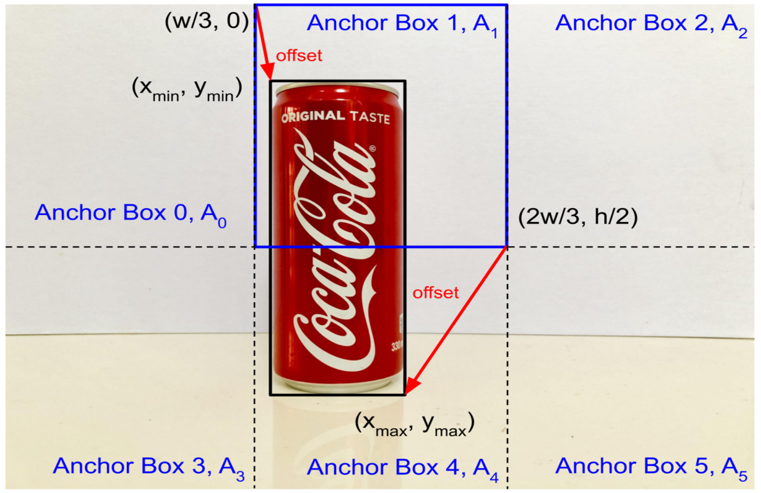
Second chance – the rest of the anchor boxes are given 2nd chance

If an anchor box, A_i has IoU > threshold with ground truth bounding box, B_j, it is also chosen as **positive anchor box** with ground truth labels (class id and offsets) from B_j

- In other words, 1 ground truth bounding box can be assigned to multiple anchor boxes

Everything else is considered **negative anchor box**

- Class id is 0
- Offsets is (0,0) (0,0) – no need to assign



The Task of Detection is to Predict Class and Offsets

y_{cls} or the category or class in the form of a one-hot vector

$y_{off} = \left((x_{offmin}, y_{offmin}), (x_{offmax}, y_{offmax})\right)$ or the offsets in the form of pixel coordinates relative to $y_{anchor} = \left((x_{amin}, y_{amin}), (x_{amax}, y_{amax})\right)$.

- For computational convenience, the offsets are better expressed in the form $y_{pred} = y_{off} = (x_{offmin}, x_{offmax}, y_{offmin}, y_{offmax})$.

Ground Truth Labels (Supervised)

y_{label} or class label of each object to detect

$y_{gt} = \left(x_{gtmin}, x_{gtmax}, y_{gtmin}, y_{gtmax}\right) = (x_{bmin} - x_{amin}, x_{bmax} - x_{amax}, y_{bmin} - y_{amin}, y_{bmax} - y_{amax})$ or the ground truth offset of the object bounding box $y_{box} = \left((x_{bmin}, y_{bmin}), (x_{bmax}, y_{bmax})\right)$ relative to $y_{anchor} = \left((x_{amin}, y_{amin}), (x_{amax}, y_{amax})\right)$.

Loss Functions

$\mathcal{L}_{cls}$ - Categorical cross-entropy loss for y_{cls}

$\mathcal{L}_{off}$ - L1 or L2 for y_{off}.

Note that only positive anchor boxes contribute to $\mathcal{L}_{off}$.

Total Loss Function:

$$\mathcal{L} = \frac{1}{N}\left(\mathcal{L}_{cls} + \alpha[u \geq 1]\mathcal{L}_{off}\right)$$

Where N is the number of matched default boxes. Iow, get the mean value.

$[u \geq 1]$ Iverson bracket means only positive anchors contribute

Recommended: $\alpha = 1$

Normalized ground truth offsets (parameterization)

$$y_{box} = \left((x_{bmin}, y_{bmin}), (x_{bmax}, y_{bmax})\right) \rightarrow \left(c_{bx}, c_{by}, w_b, h_b\right)$$
$$y_{anchor} = \left((x_{amin}, y_{amin}), (x_{amax}, y_{amax})\right) \rightarrow \left(c_{ax}, c_{ay}, w_a, h_a\right)$$

where:

$$\left(c_{bx}, c_{by}\right) = \left(x_{min} + \frac{x_{max} - x_{min}}{2}, y_{min} + \frac{y_{max} - y_{min}}{2}\right)$$
$$(w_b, h_b) = (x_{max} - x_{min}, y_{max} - y_{min})$$

$$y_{gt} = \left(\frac{c_{bx} - c_{ax}}{w_a}, \frac{c_{by} - c_{ay}}{h_a}, \log\frac{w_b}{w_a}, \log\frac{h_b}{h_a}\right)$$

Normalized ground truth offsets using standard deviation

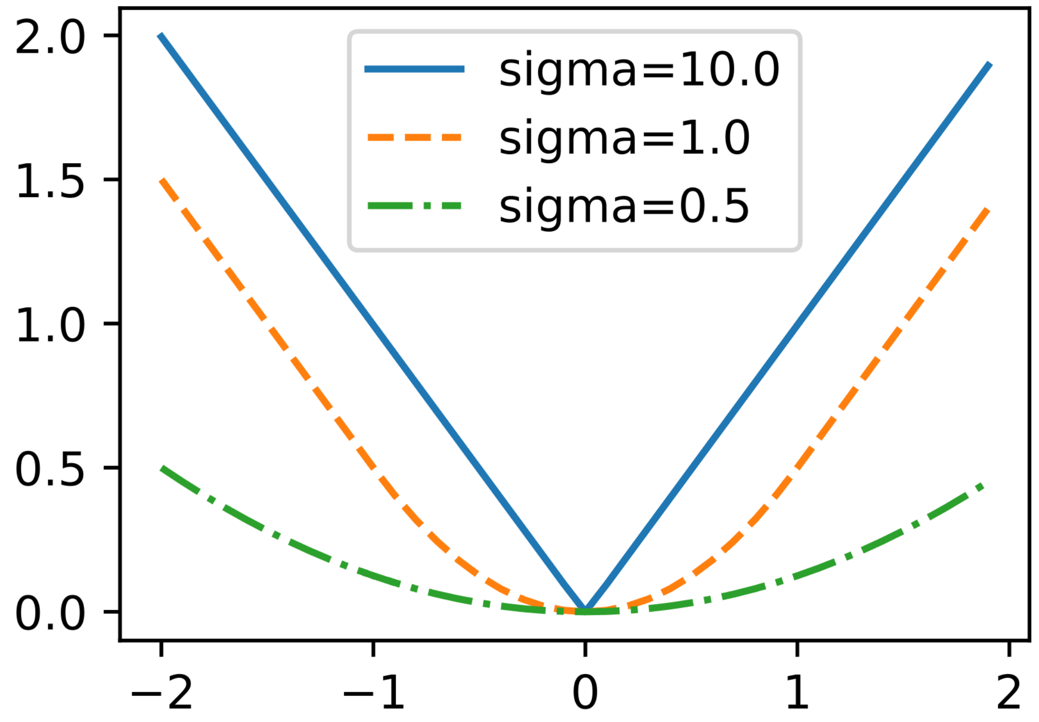
$$y_{gt} = \left(\frac{\frac{c_{bx} - c_{ax}}{w_a}}{\sigma_x}, \frac{\frac{c_{by} - c_{ay}}{h_a}}{\sigma_y}, \frac{\log \frac{w_b}{w_a}}{\sigma_w}, \frac{\log \frac{h_b}{h_a}}{\sigma_h} \right)$$

The recommended values are: $\sigma_x = \sigma_y = 0.1$ and $\sigma_w = \sigma_h = 0.2$.

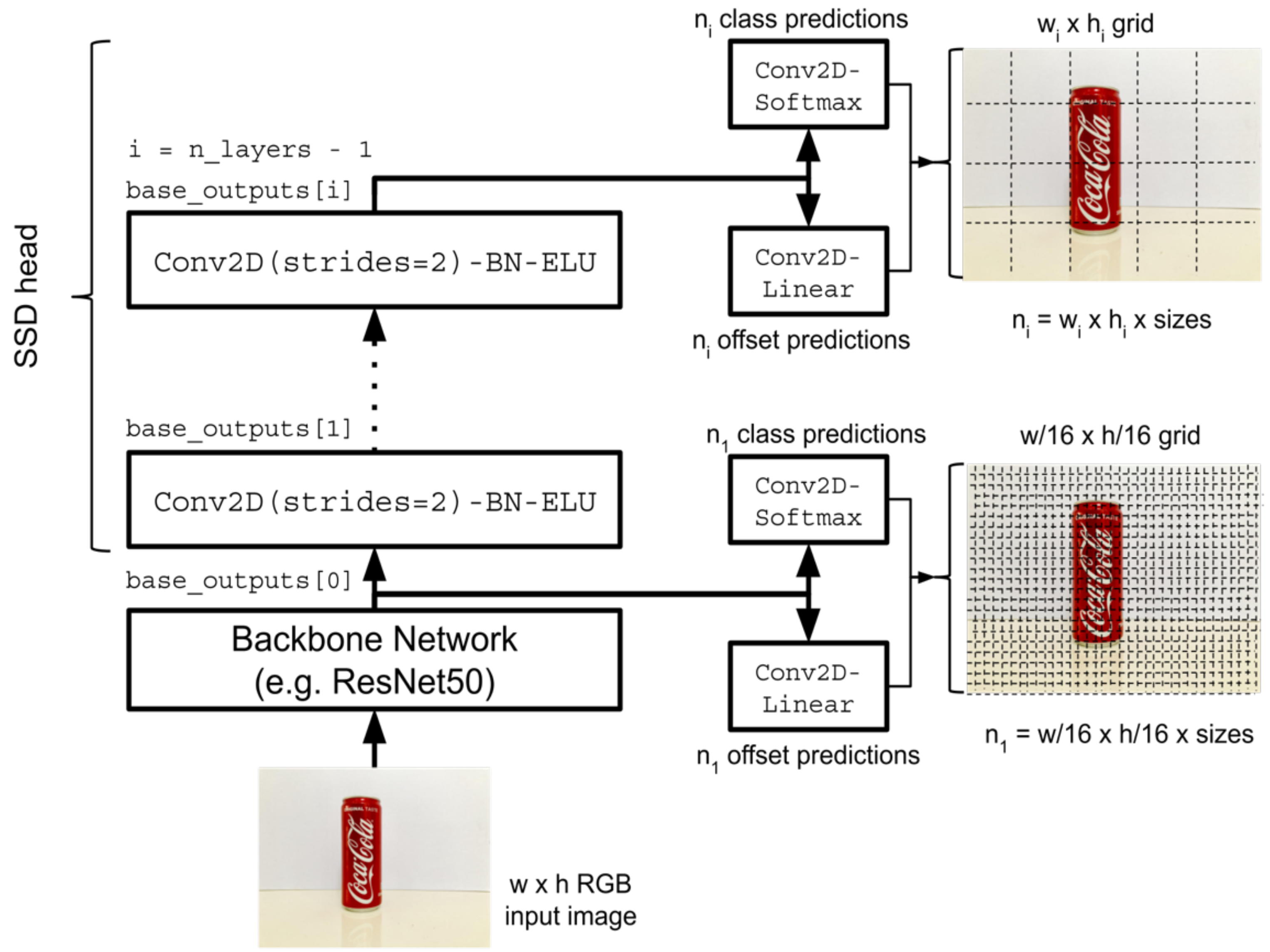
- About 10% of w_a or h_a

Smooth L1 compared to L1 yields better results

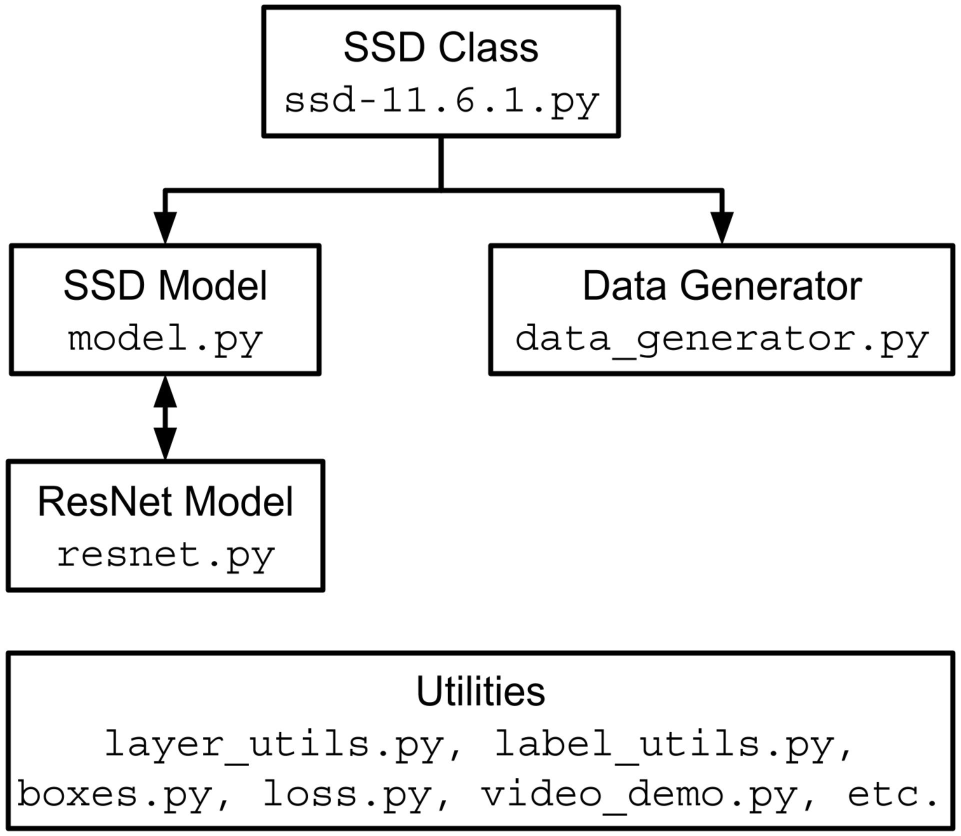
$$\mathcal{L}_{off} = L1_{smooth}\left(y_{pred} - y_{gt}\right) = L1_{smooth}(u) = \begin{cases} \frac{(\sigma u)^2}{2} & if\ |u| < \frac{1}{\sigma^2} \\ |u| - \frac{1}{2\sigma^2} & otherwise \end{cases}$$



SSD Model Architecture



Keras



https://github.com/PacktPublishing/Advanced-Deep-Learning-with-Keras/tree/master/chapter11-detection

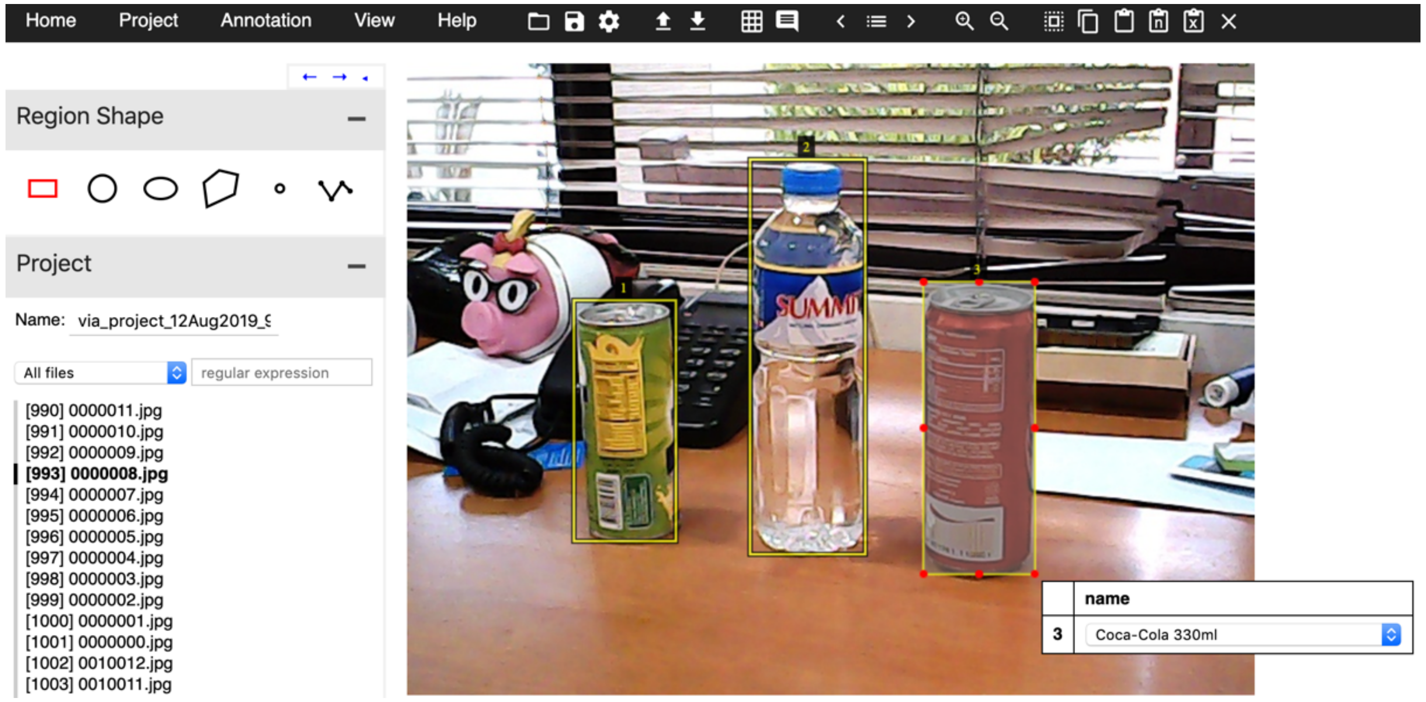
Example Dataset and Labelling using VIA

1k train images

50 test images

Annotated using VIA

- VGG Image Annotator
  http://www.robots.ox.ac.uk/~vgg/software/via/



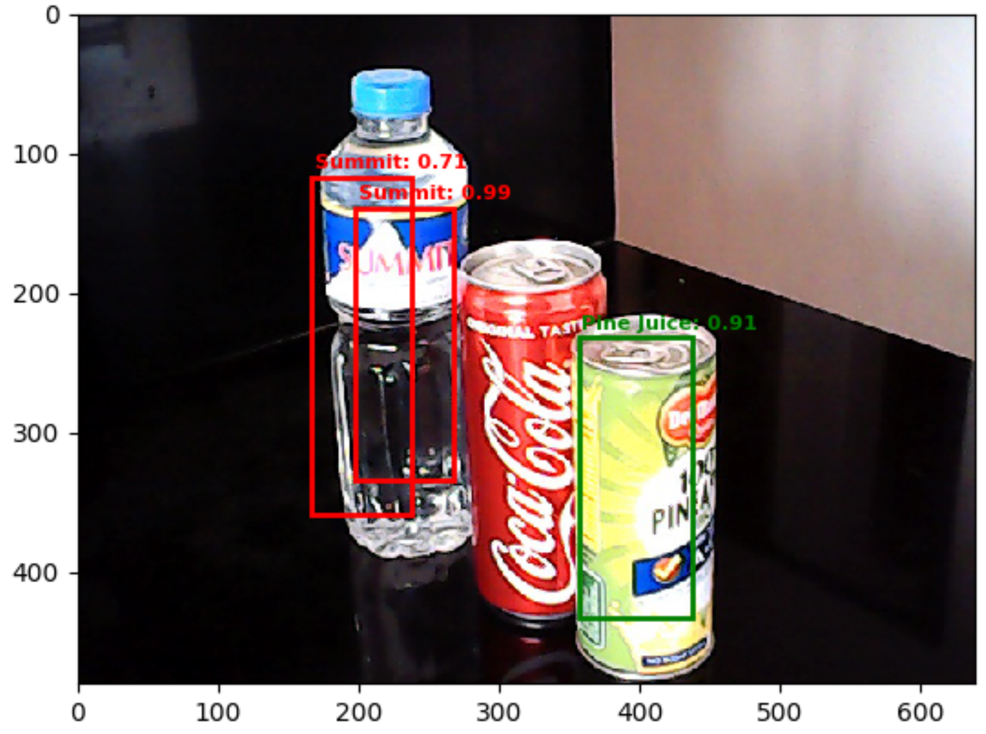
Name: via_project_12Aug2019_

[990] 0000011.jpg
[991] 0000010.jpg
[992] 0000009.jpg
[993] 0000008.jpg
[994] 0000007.jpg
[995] 0000006.jpg
[996] 0000005.jpg
[997] 0000004.jpg
[998] 0000003.jpg
[999] 0000002.jpg
[1000] 0000001.jpg
[1001] 0000000.jpg
[1002] 0010012.jpg
[1003] 0010011.jpg

	name
3	Coca-Cola 330ml

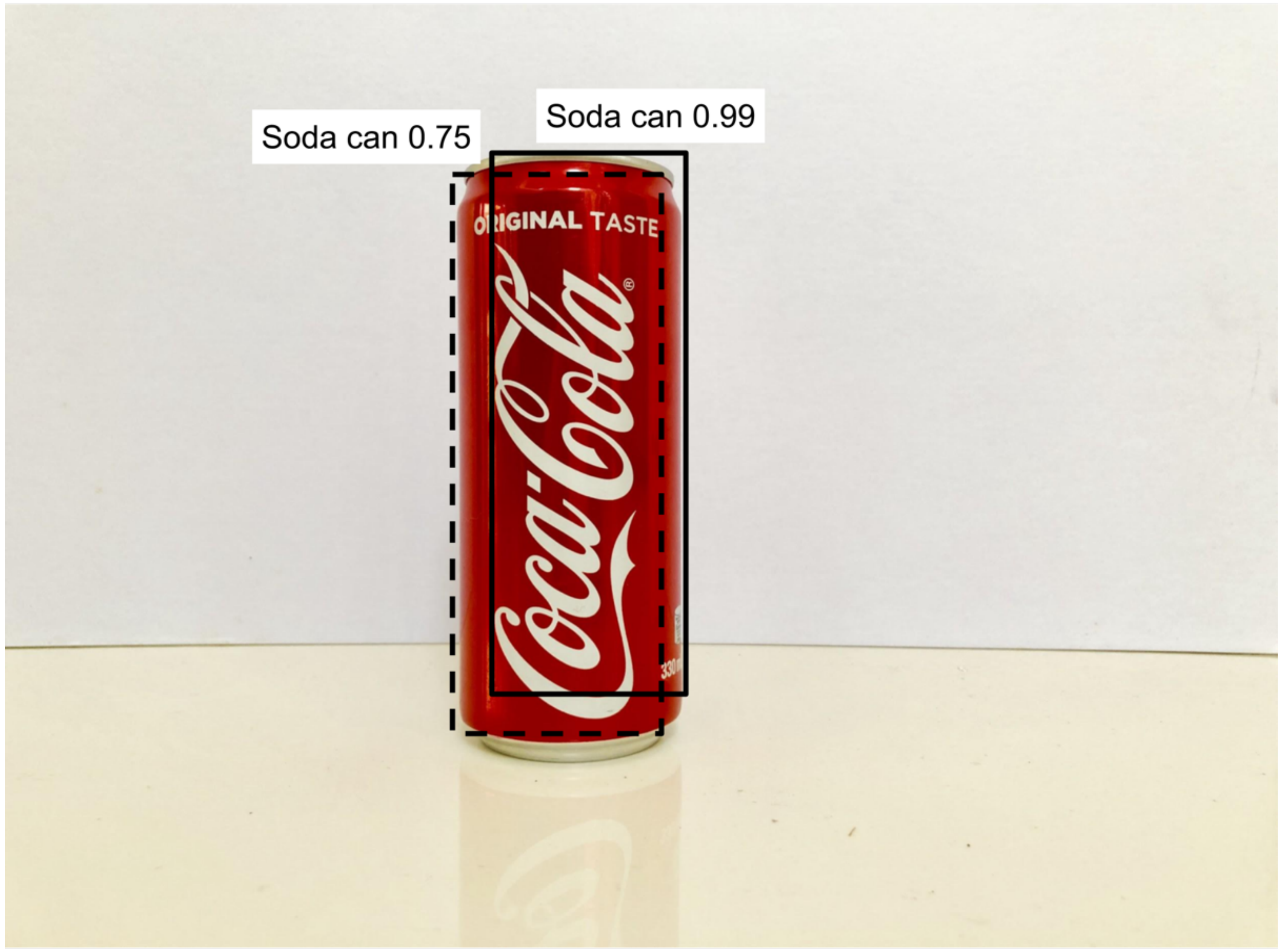
Non-Maximum Suppression (NMS) and Soft NMS

During prediction, multiple anchor boxes might generate predictions for the same object



NMS

The network predicted two overlapping bounding boxes for the *Soda can* object. Only one valid bounding box is chosen and that is the one with the higher score of 0.99.



Soft NMS

Soft NMS proposes that instead of outright removal from the list, the score of the overlapping bounding box is decreased at a negative exponential rate in proportion to the square of its IoU with the object of biggest confidence.

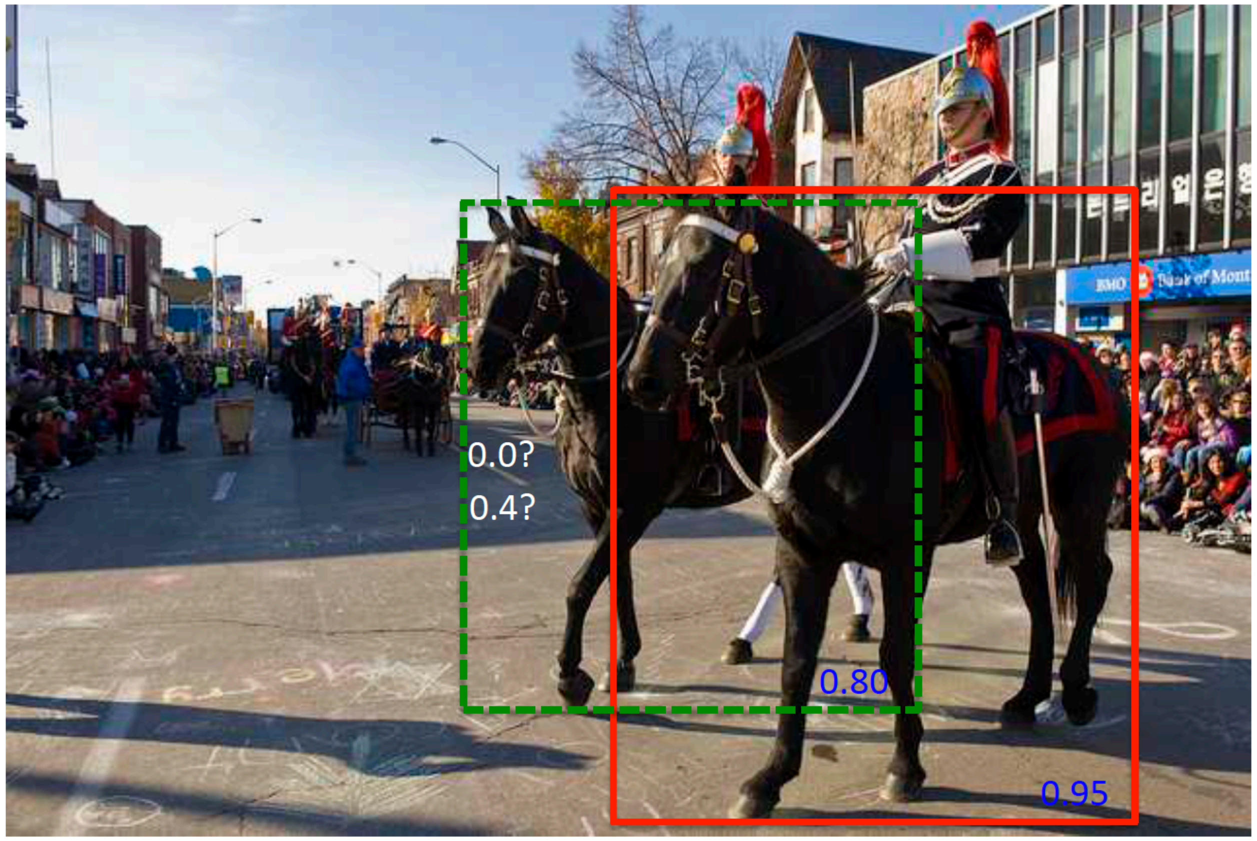


Figure 1. This image has two confident horse detections (shown in red and green) which have a score of 0.95 and 0.8 respectively. The green detection box has a significant overlap with the red one. Is it better to suppress the green box altogether and assign it a score of 0 or a slightly lower score of 0.4?

Algorithm 11.12.1 **NMS and Soft NMS**

```
1.  *Require:* Bounding box predictions: $\mathcal{B} = \{b_1, b_2, \dots, b_n\}$.
2.  *Require:* Bounding box class confidence or scores: $\mathcal{P} = \{p_1, p_2, \dots, p_n\}$.
3.  *Require:* NMS minimum IoU threshold: $N_t$.
4.  $\mathcal{D} \leftarrow \{\ \}; \mathcal{S} \leftarrow \{\ \}$
5.  **while** $\mathcal{B} \neq empty$ **do**
6.    $m \leftarrow argmax\ \mathcal{P}$
7.    $\mathcal{M} \leftarrow b_m; \mathcal{N} \leftarrow p_m,$
8.    $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{M}; \mathcal{B} \leftarrow \mathcal{B} - \mathcal{M}; \mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{N}; \mathcal{P} \leftarrow \mathcal{P} - \mathcal{N};$
9.    **for** steps $b_i\ in\ \mathcal{B}$ **do**
10.     **if** $soft\_NMS = True$ **then**
11.       $p_i = p_i e^{-\frac{IoU(\mathcal{M}, b_i)^2}{\sigma}}$
12.     **elif** $IoU(\mathcal{M}, b_i) \geq N_t$ **then**
13.       $\mathcal{B} = \mathcal{B} - b_i\ ; \mathcal{P} = \mathcal{P} - p_i$
14.     **end**
15.   **end**
16. **end**
17. return $\mathcal{D}, \mathcal{S}$
```

NMS and Soft-NMS remove redundant predictions based on confidence and IoU

Simple Evaluation Metric

Mean IOU:

$$mIoU = \frac{1}{n_{box}} \sum_{i \in \{1,2,..n_{box}\}} \max_{j \in \{1,2,..n_{pred}\}} IoU(b_i, d_j)$$

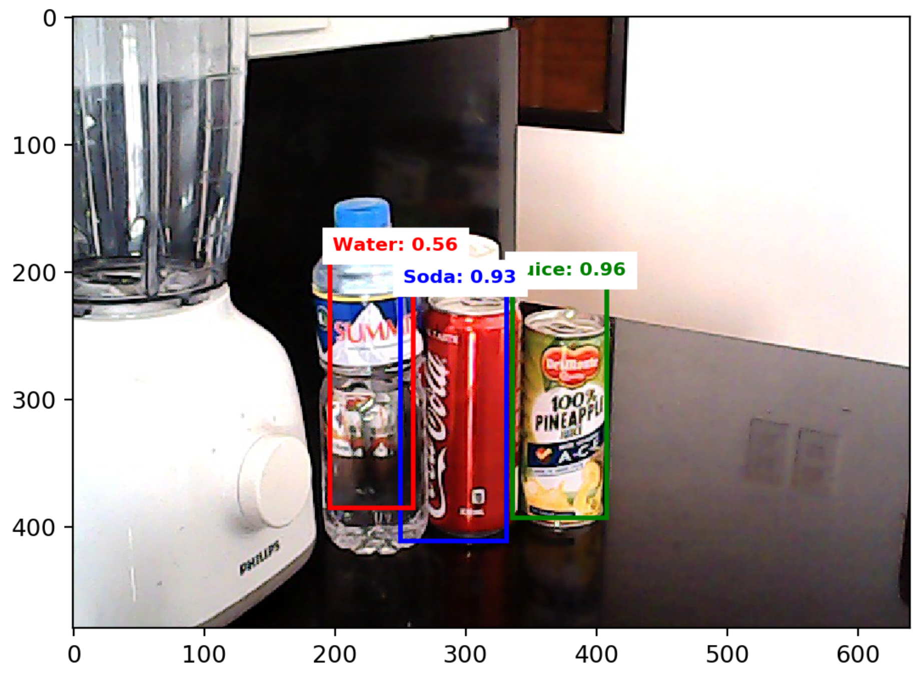
Precision:

$$precision = \frac{tp}{tp + fp} = \frac{True\_Positive}{True\_Positive + False\_Positive} = \frac{tp}{total\_positive}$$

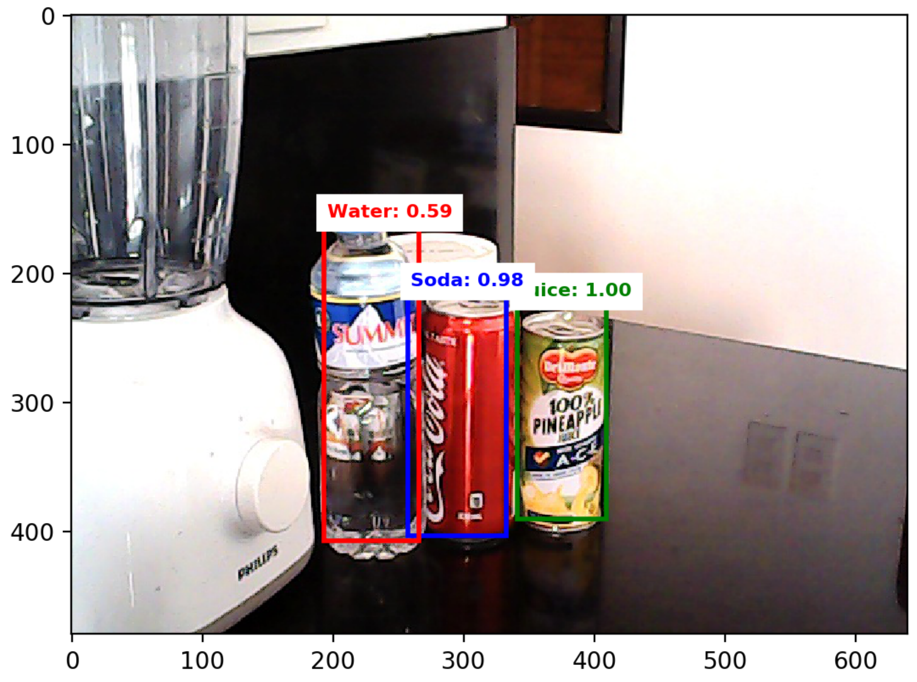
Recall:

$$recall = \frac{tp}{tp + fn} = \frac{True\_Positive}{True\_Positive + False\_Negative}$$
$$= \frac{tp}{true\_total\_positive}$$

Sample Results

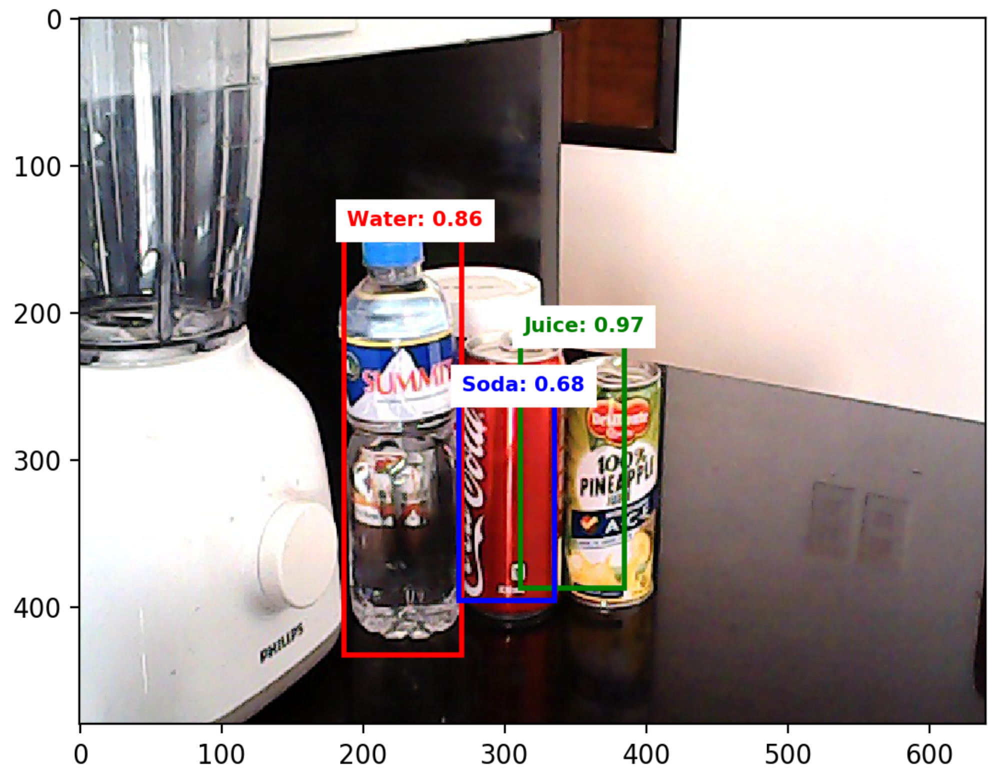


Unnormalized Offsets

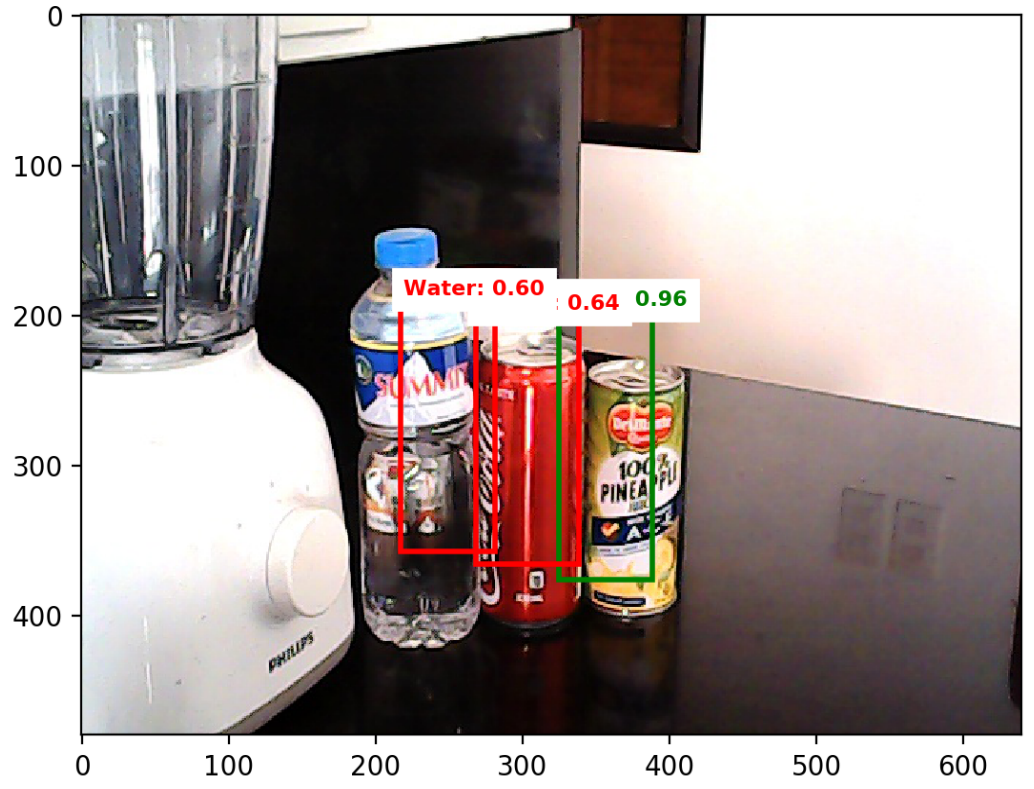


Unnormalized Offsets
with Smooth L1

Sample Results



Normalized Offsets



Normalized Offsets
with Smooth L1

Sample Evaluation

	Un-normalized offsets	Un-normalized offsets, smooth L1	Normalized offsets	Normalized offsets, smooth L1	Normalized offsets, smooth L1, focal loss
mIoU	0.64	0.61	0.53	0.50	0.51
Average precision	0.87	0.86	0.90	0.85	0.85
Average recall	0.87	0.85	0.87	0.83	0.83

Performance

VOC2007: 68.0 mAP, SSD 300 (300 x 300 input)

VOC2007: 71.6 mAP, SSD 512 (512 x 512 input)

Pascal VOC 2007:

> The data has been split into 50% for training/validation and 50% for testing. The distributions of images and objects by class are approximately equal across the training/validation and test sets. In total there are 9,963 images, containing 24,640 annotated objects.

Average Precision (AP)

AP is the average precision over recall values from 0 to 1 linearly spaced by 11

$$AP = \frac{1}{11} \sum_{r \epsilon \{0, 0.1, \ldots, 1.0\}} P_r$$

P_r is the precision at a given recall r

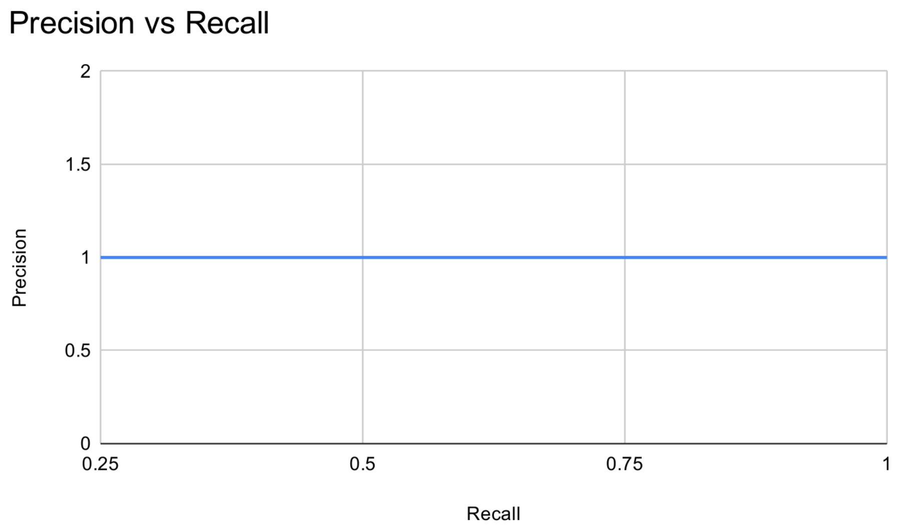
Positive prediction if $IoU \geq 0.5$ (VOC2007)

mean Average Precision or **mAP** score is calculated by taking the mean AP over all classes and/or over all IoU thresholds

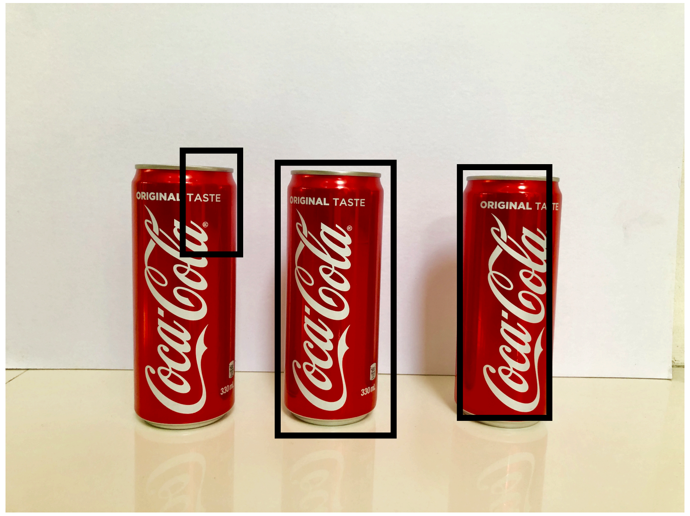
- For PASCAL VOC2007 Challenge, there is only one threshold = 0.5 and there are 20 classes
- For the COCO 2017 challenge, the mAP was averaged over all 80 object categories and all 10 IoU thresholds.

Average Precision (AP)

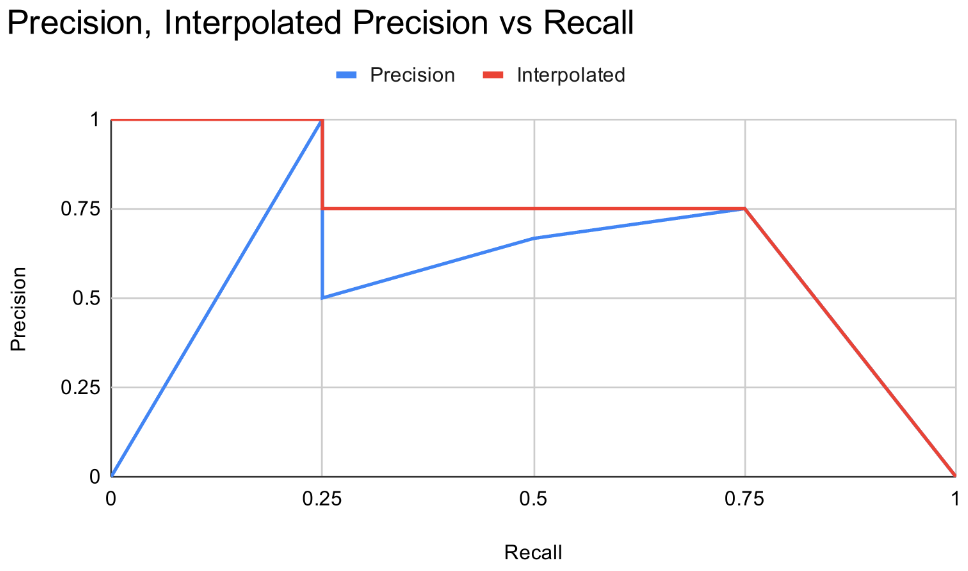
Rank	Prediction	Precision	Recall
1	TP	1/1=1.0	¼=0.25
2	TP	2/2=1.0	2/4=0.5
3	TP	3/3=1.0	¾=0.75
4	TP	4/4=1.0	4/4=1.0
	AP	1.0=(11x1.0)/11	

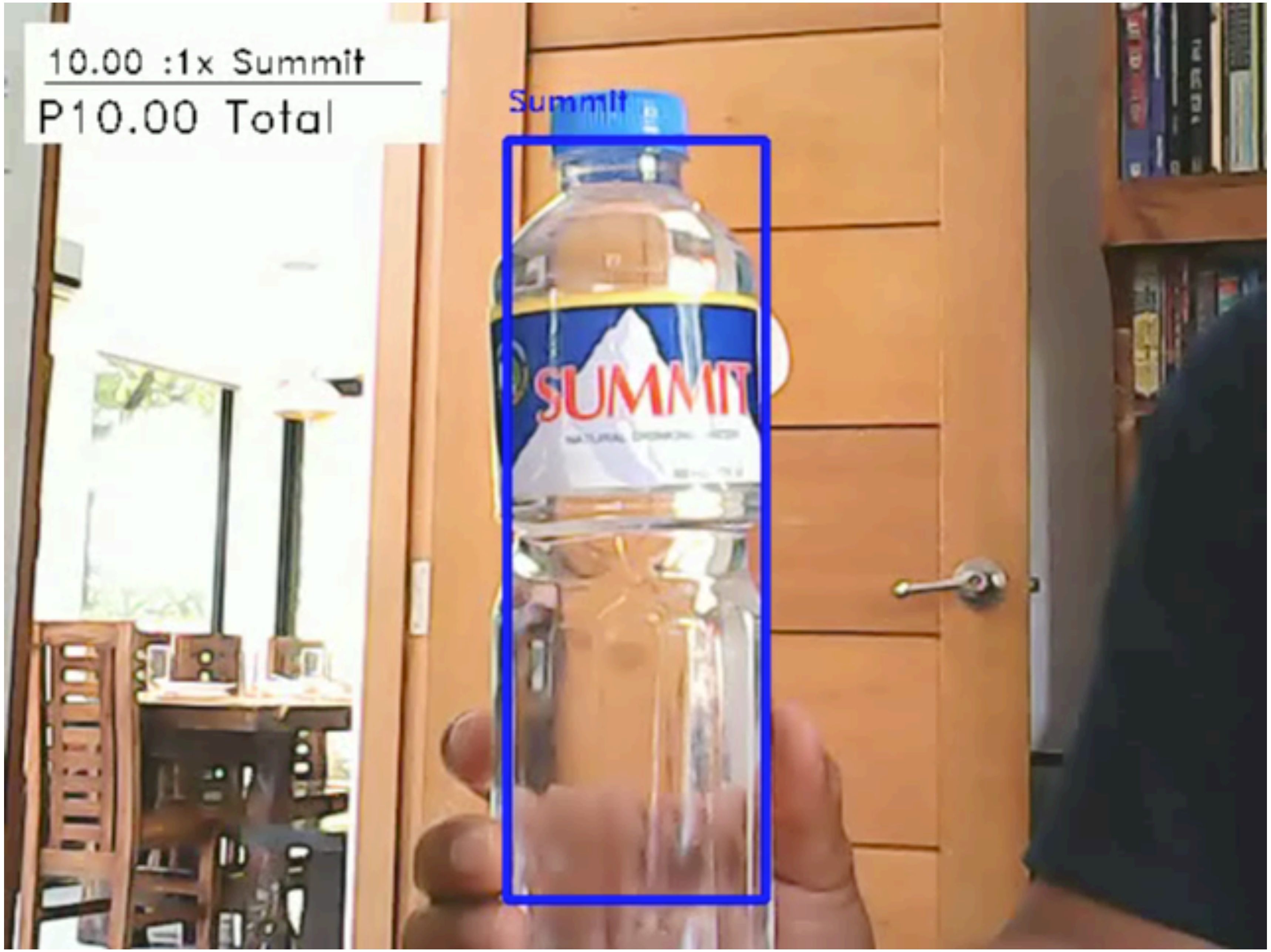


Average Precision (AP)



Rank	Prediction	Precision	Recall
1	TP	1/1=1.0	¼=0.25
2	FP	1/2=0.5	1/4=0.25
3	TP	2/3=0.67	2/4=0.5
4	TP	3/4=0.75	3/4=0.75
	AP	(1x3+0.75x5+?x3)/11	





```
10.00 :1x Summit
P10.00 Total
```

Sample Outputs (SSD512 on COCO2015)

Fast R-CNN

Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE international conference on computer vision. 2015.

RCNN is slow

1st stage: Classifier with SVM.
Classifier is not parallel!
Process object proposal one at a time.

2nd stage: Bounding box regressor (not shown)
Region proposals – from classical selective search algorithm

R-CNN: Region-based Convolutional Network

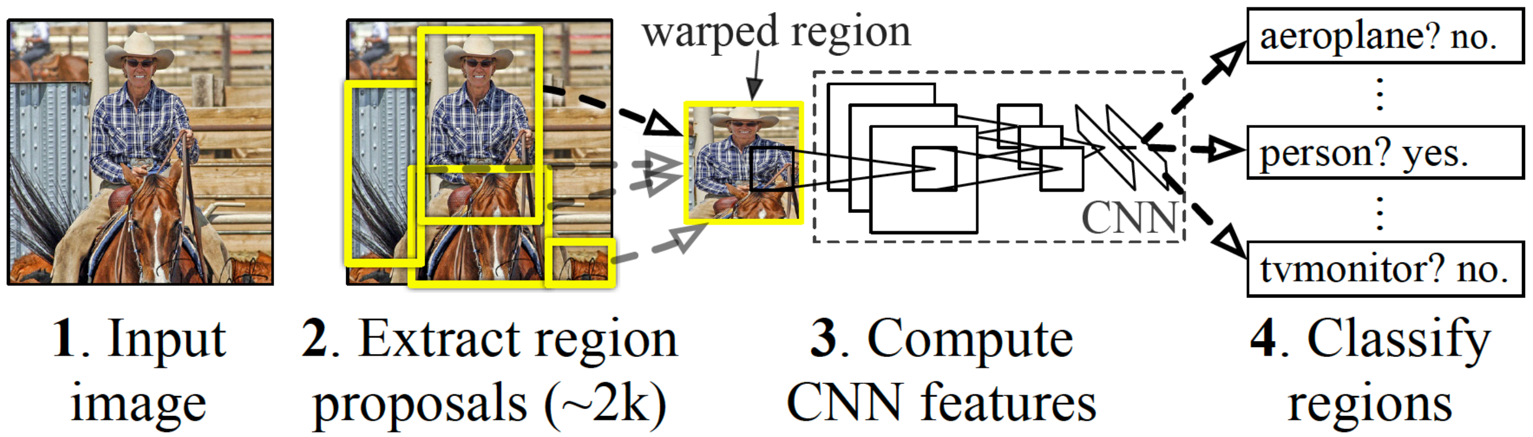


Fig. 1. Object detection system overview. Our system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional network (CNN), and then (4) classifies each region using class-specific linear SVMs. We trained an R-CNN that achieves a mean average precision (mAP) of 62.9% on PASCAL VOC 2010. For comparison, [21] reports 35.1% mAP using the same region proposals, but with a spatial pyramid and bag-of-visual-words approach. The popular deformable part models perform at 33.4%. On the 200-class ILSVRC2013 detection dataset, we trained an R-CNN with a mAP of 31.4%, a large improvement over OverFeat [19], which had the previous best result at 24.3% mAP.

Fast Region-based Convolutional Network method (Fast R-CNN)

- Classic object detector
- Pre-computed region proposals or Region of Interests (ROIs)
- Faster than classic R-CNN
    - R-CNN uses SVM for classification
    - Slow at 47s/image
    - Slow training due to SVM
    - Intermediate results are written on a disk

Fast R-CNN

Contributions:

- Higher detection quality (mAP) than R-CNN
- Training is single-stage, using a multi-task loss
- Training can update all network layers
- No disk storage is required for feature caching

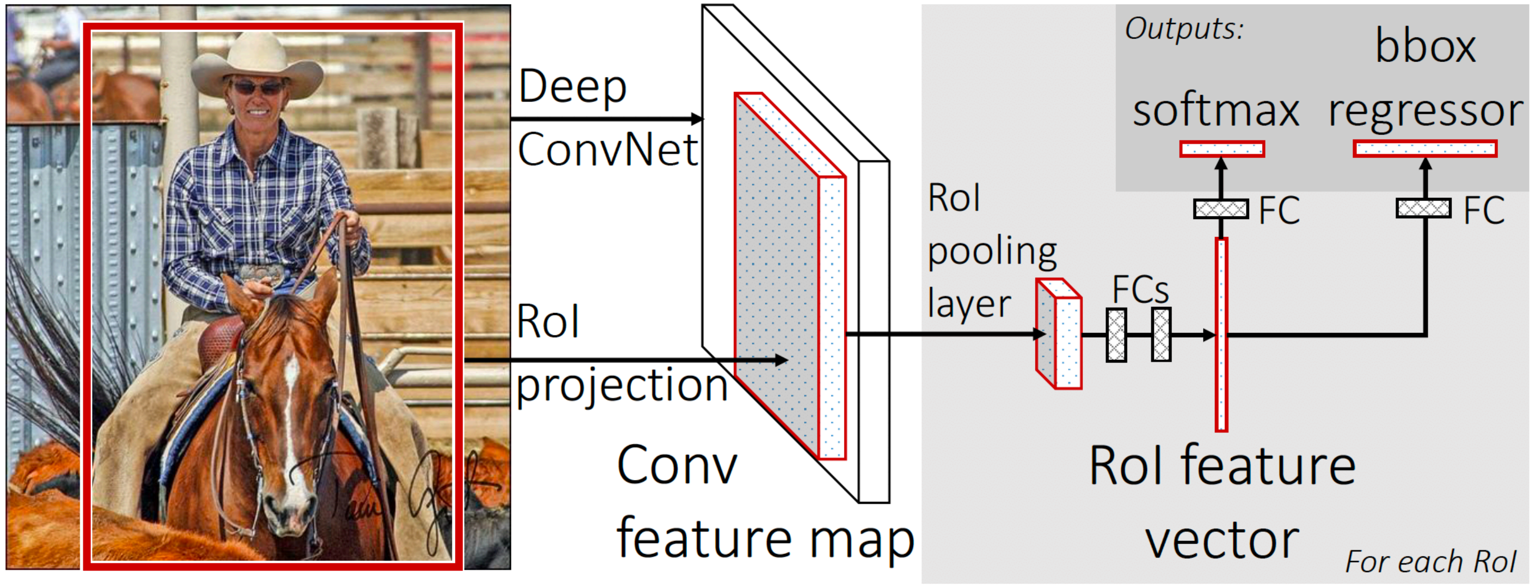
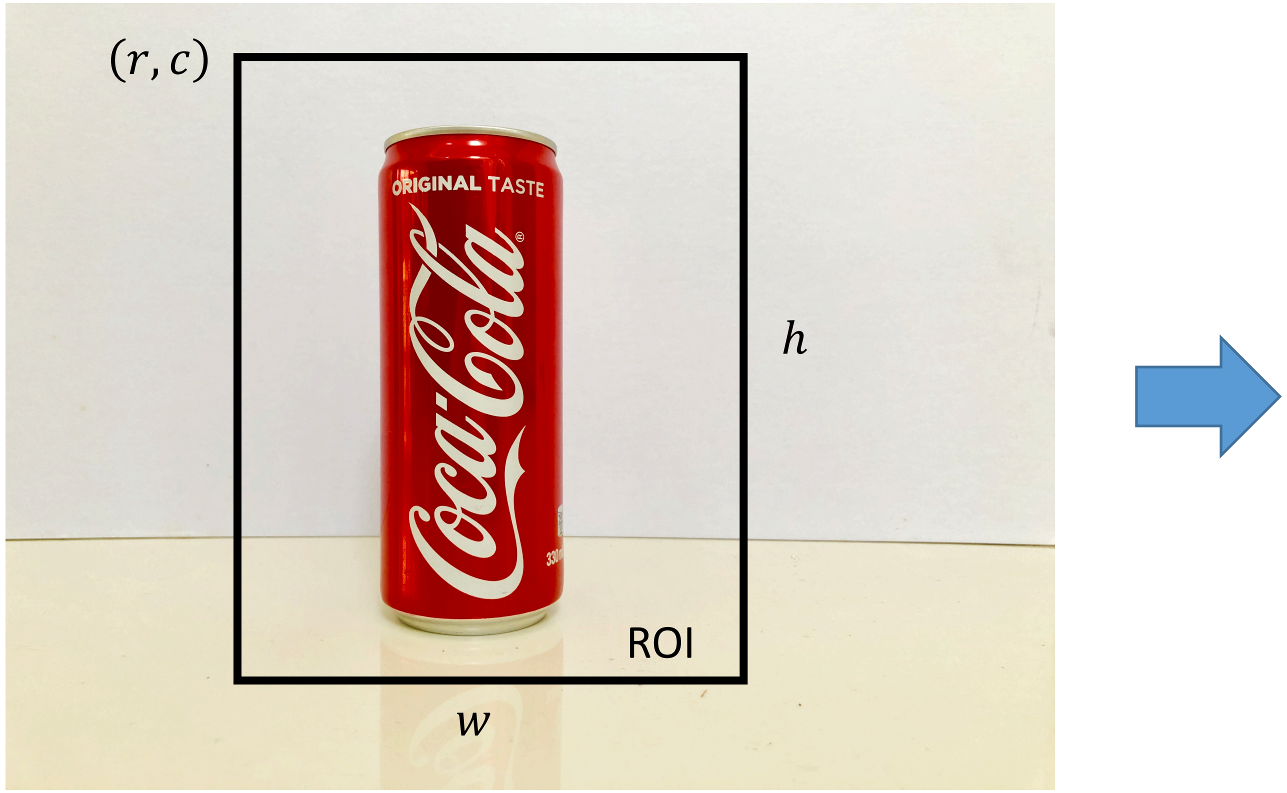


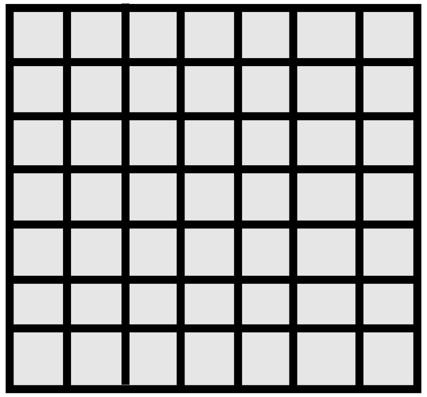
Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

Fast R-CNN Pooling Layer

Input: (h, w) ROI



Target: (H, W) Feature Map

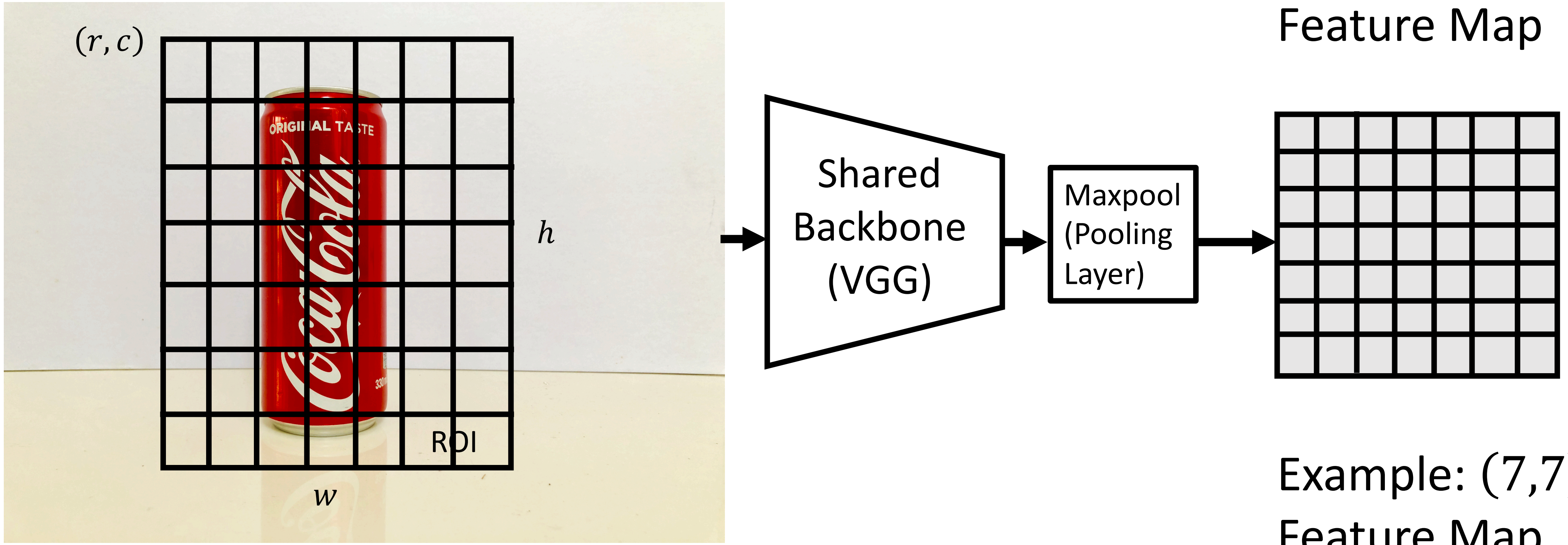


Example: $(7,7)$ Feature Map

Fast R-CNN Pooling Layer

Input: (h, w) ROI

Target: (H, W)
Feature Map



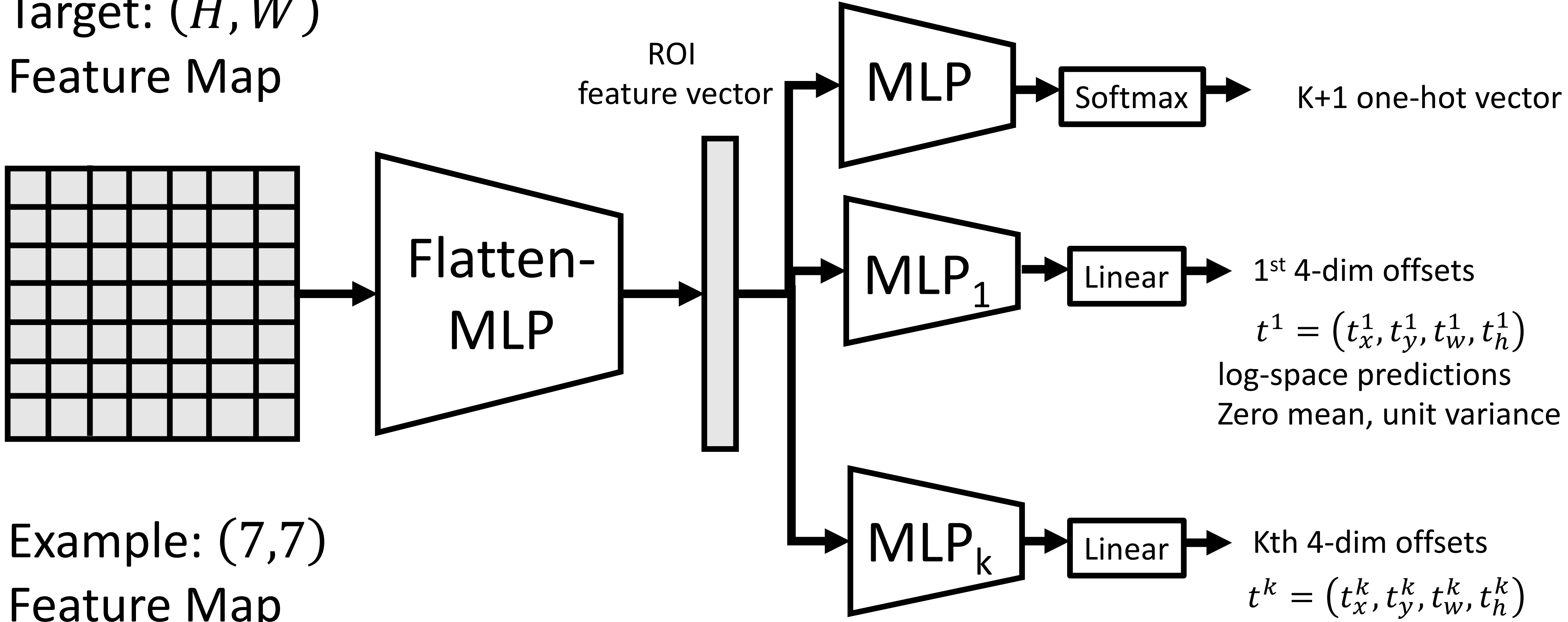
Example: $(7,7)$
Feature Map

ROI is divided into $\left(\frac{w}{W}, \frac{h}{H}\right)$ regions

Note: ROI is ROI projection map (feature map)

Fast R-CNN Outputs per ROI

Target: (H, W)
Feature Map



Example: $(7,7)$
Feature Map

Fast R-CNN Other Network Details

- Backbone or base network – VGG 16 pre-trained on ImageNet 1000-category classifier
- Last MaxPool of VGG is replaced by ROI Pooling Layer
- Classifier layer is replaced by a sibling network
    - K + 1 –dim Object classifier
    - K 4-dim Offsets predictors

Training

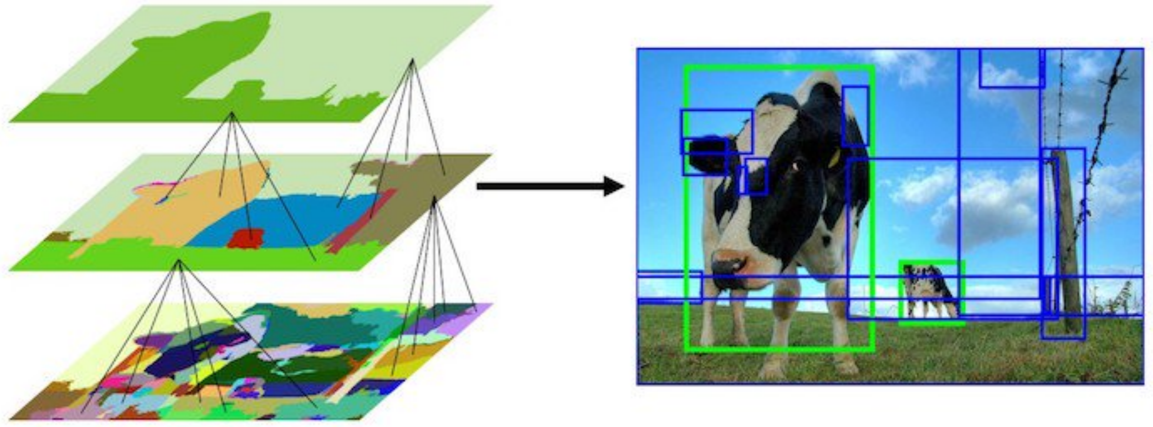
- Input: Image
- Output ground truth data:
    - Region of Interest (ROI)
        - Object class
        - Bounding box coordinates

Loss Functions, Ground Truth, Augmentation, ROI

- See SSD Loss Function
- Total Loss Function: $\mathcal{L} = \mathcal{L}_{cls} + \alpha[u \geq 1]\mathcal{L}_{off}$
  - $\mathcal{L}_{cls} = -\log p_u$, is the log loss for the true class
  - $[u \geq 1]$ Iverson bracket means only positive anchors contribute
- Ground truth:
  - $IoU \geq 0.5$ of ROI with ground truth $[u \geq 1]$
  - $0.1 \leq IoU < 0.5$ are labelled $u = 0$
- Augmentation: Image flipping
- ROI: 2000 generated by Selective Search algorithm

Selective Search

- Use for generating ROIs
- Based on hierarchical grouping of similar regions based on color, texture, size and shape
  - Hand-crafted feature extractions
- Over-segment an image:
  - Each segmented object is an ROI
  - Merge similar neighboring regions to form another ROI
  - Also known as merging of superpixels



https://www.koen.me/research/selectivesearch/

Training details

- Backbone is VGG pretrained on ImageNet
- Shortest side of image is 600
- End to end training

Performance

- VOC2007: 66.9 mAP
- VOC2010: 66.1 mAP
- VOC2012: 65.7 mAP

Faster R-CNN

Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems. 2015.

Faster R-CNN

- ROIs generation in Fast R-CNN is the slowest part of object detection pipeline
  - Selective Search is 2sec per image on CPU
  - Actual object detection is only 0.2sec
- Introduces Region Proposal Network (RPN) to replace ROIs
  - Faster R-CNN = RPN + Fast R-CNN

Faster R-CNN

2) Train Fast R-CNN.
Goto 1.

1) Train RPN.
Freeze. Goto 2.

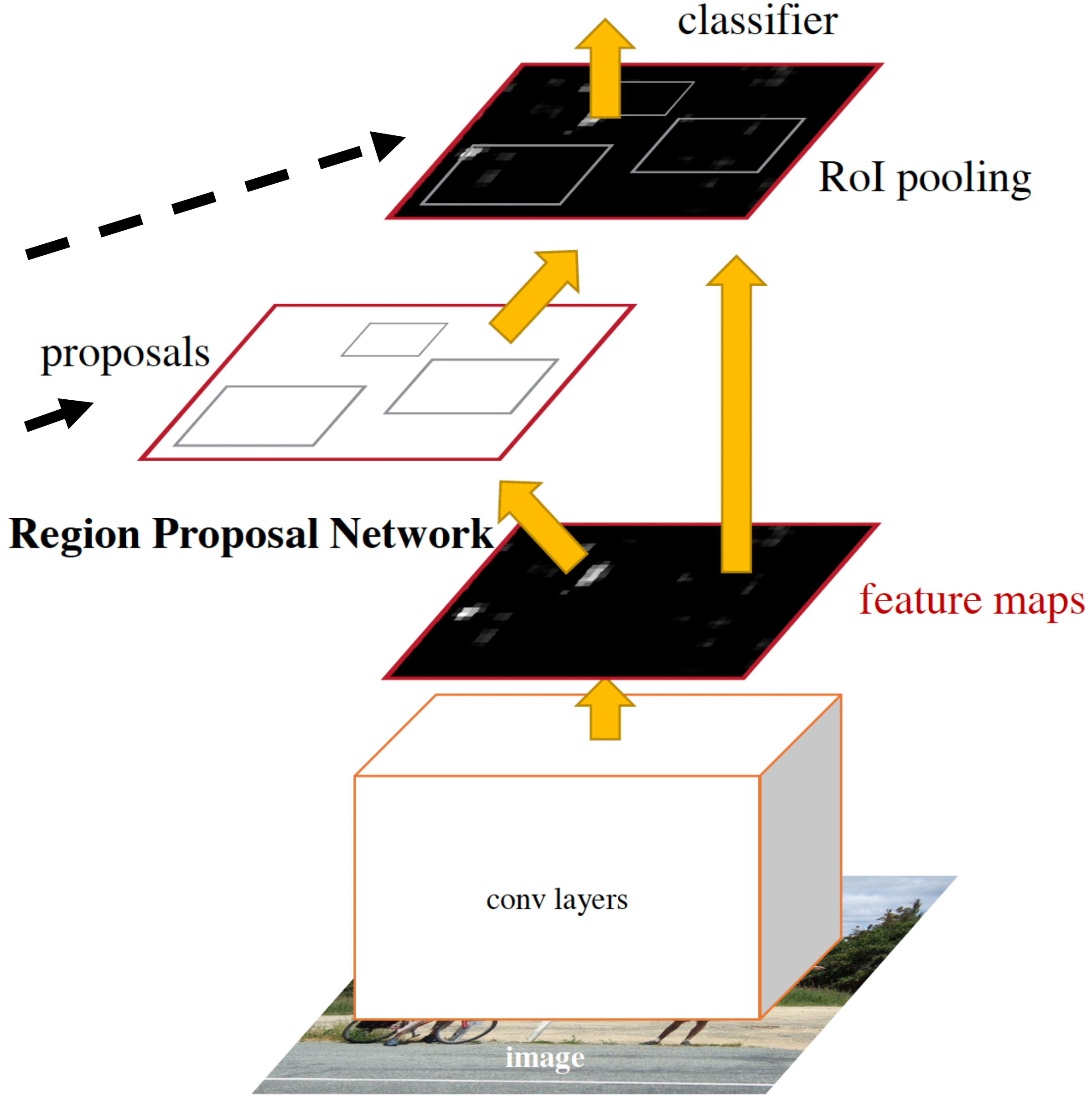


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

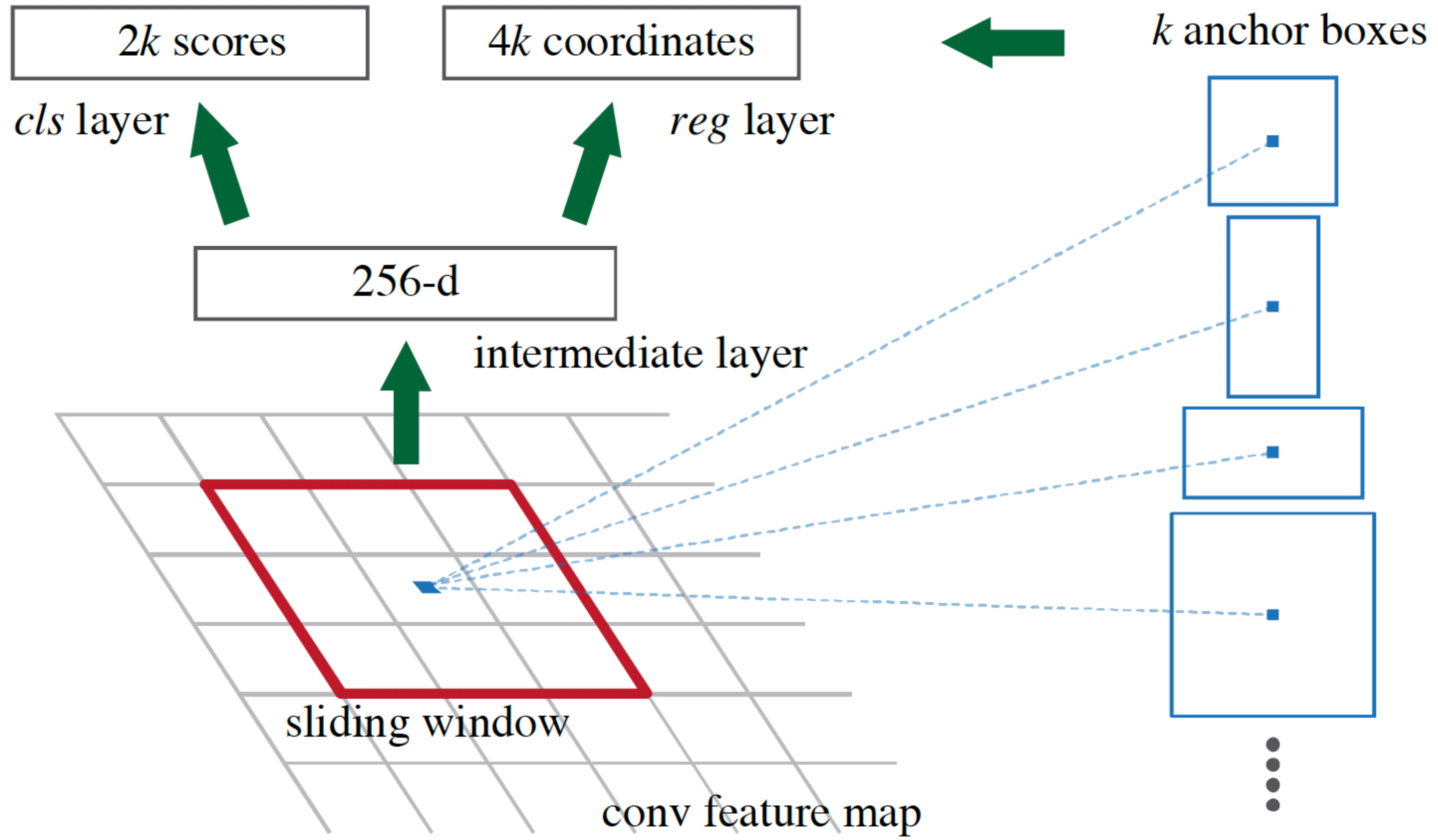
Region Proposal Network (RPN)

- Input: Image
- Outputs: Rectangular object proposals
- Method
  - Slide a network over the last feature map
  - The network takes $n \times n$ input window (e.g. $n = 3$)
  - The network output is a 256-dim or 512-dim code
  - The code is input to object classifier and bounding box coordinates regression layer
    - 2-dim Classifier : object or no object
    - 4-dim Box coordinates (proposed region coordinates if object)

RPN

- The maximum number of proposals per window location is k
  - Output: $2k$ classifier (object or no object)
  - Output: $4k$ box coordinates
- Each proposal is parameterized with reference to a box called ***Anchor*** centered at the window location.
- 3 scales and 3 aspect ratios generate $k = 9$ anchors
- Therefore, for a $W \times H$ feature map, there are WHk anchors
  - Typically $W \times H \approx 2400$

RPN


2k scores
4k coordinates
k anchor boxes
cls layer
reg layer
256-d
intermediate layer
sliding window
conv feature map

Multi-Scale ROI using Pyramid of Anchors

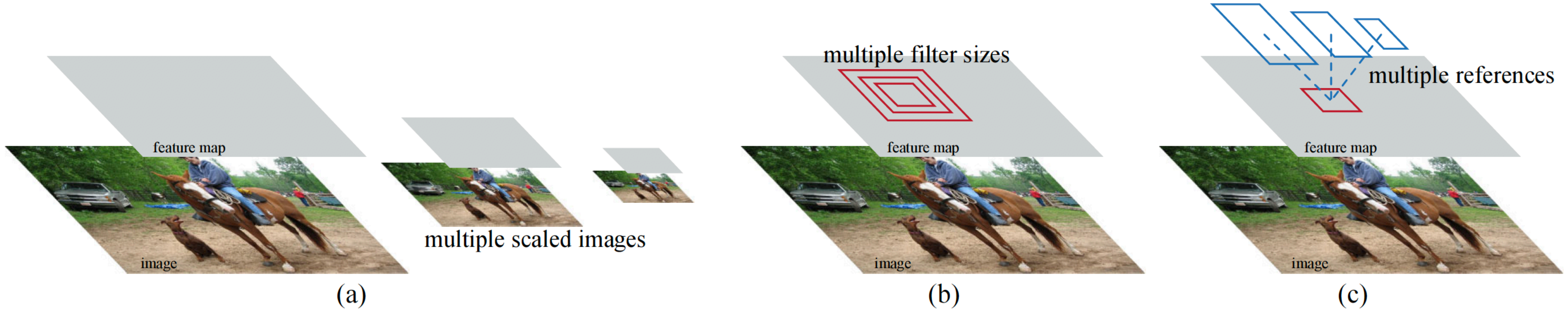


Figure 1: Different schemes for addressing multiple scales and sizes. (a) Pyramids of images and feature maps are built, and the classifier is run at all scales. (b) Pyramids of filters with multiple scales/sizes are run on the feature map. (c) We use pyramids of reference boxes in the regression functions.

RPN generates an anchor box as reference at multiple scales and aspect ratios

Anchor Box Ground Truth Label

- Anchor with the highest IoU for a given ground truth bounding box is tagged as positive anchor box
- Anchor with $IoU > 0.7$ with any ground truth bounding box is tagged as positive anchor box
- A ground truth bounding box may assign positive labels to multiple anchor boxes
- Anchor with $IoU < 0.3$ for all ground truth boxes is tagged as negative anchor box
- Anchors that are neither positive nor negative do not contribute to the training objective

Loss Function

- See SSD Loss Function
- Total Loss Function: $\mathcal{L} = \frac{1}{N_{cls}} \mathcal{L}_{cls} + \frac{1}{N_{reg}} \lambda [u \geq 1] \mathcal{L}_{off}$
- $[u \geq 1]$ Iverson bracket means only positive anchors contribute
- $\mathcal{L}_{cls}$ is log loss over 2 classes (object, not object)
- N_{cls} is the mini-batch size (~256)
- N_{reg} is the number of anchor locations (~2,400)
- λ is weighting factor (set to 10 to equalize contributions of $\mathcal{L}_{cls}$ and $\mathcal{L}_{off}$)
- $\mathcal{L}_{off} = smooth\_L1\left(y_{pred} - y_{gt}\right)$ is smooth L1 loss with coordinates parameterized (same as SSD)

$$y_{gt} = \left(\frac{c_{bx} - c_{ax}}{w_a}, \frac{c_{by} - c_{ay}}{h_a}, \log \frac{w_b}{w_a}, \log \frac{h_b}{h_a}\right)$$

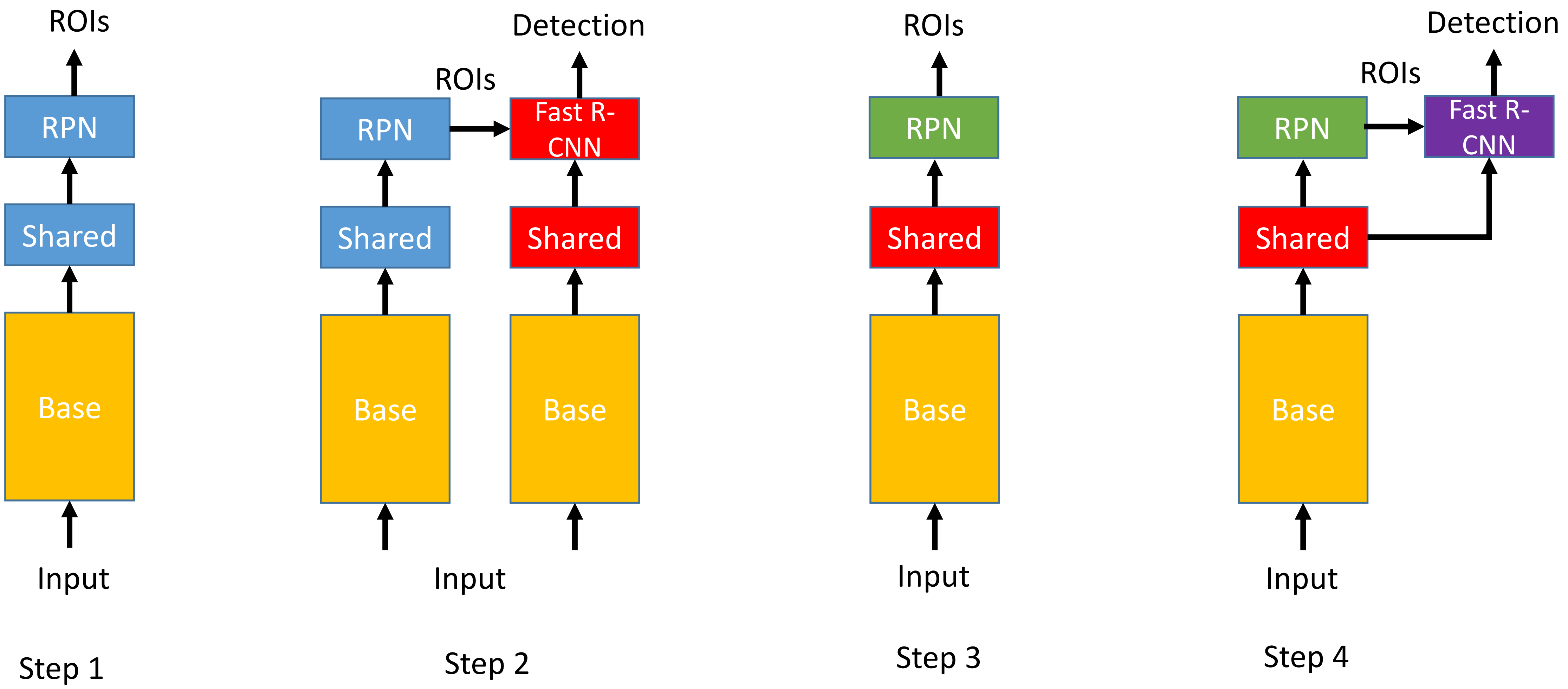
RPN Training

- Not all anchors are used since there is positive-negative anchor boxes imbalance
- 256 anchors are sampled with about 1:1 positive:negative anchors ratio
- Backbone: Pre-trained on ImageNet VGG 16

RPN + Fast R-CNN Training

- Alternating training is used
  - Train RPN to generate region proposals
  - Train Fast R-CNN on these region proposals
- 4-step alternating training
  - Step 1: Train RPN alone end-to-end
  - Step 2: Train a separate network for Fast R-CNN but with regions coming from RPN in Step 1
  - Step 3: Train RPN again but first copying the weights for the shared convolutional layer. The shared weights are frozen. Train only layers unique to RPN.
  - Step 4: With trained RPN generating proposals, train Fast R-CNN again. The shared weights are frozen. Train only layers unique to Fast R-CNN.

4-step Training



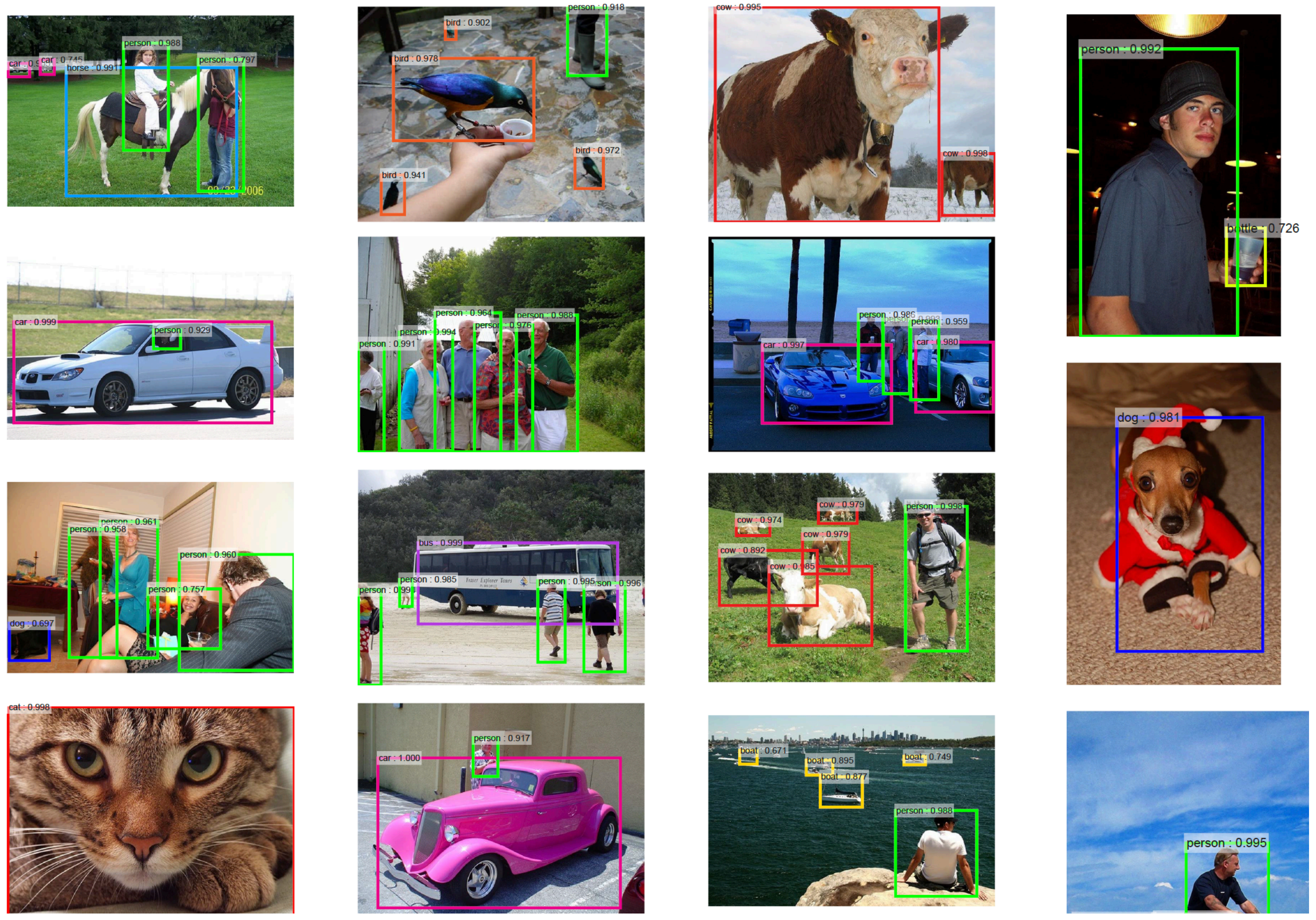
Other training details

- 3 Scales: 128^2, 256^2, 512^2
- 3 Aspect Ratios: 1:1, 1:2, 2:1
- Shortest side of image is 600
- Total stride of backbone network is 16
- Cross-boundary anchors are ignored (anchors exceeding image boundaries). Net anchors is about 6,000.

Performance

- VOC2007: 69.9 mAP
- VOC2012: 67.0 mAP

Sample Outputs (VOC2007)



FPN

Lin, Tsung-Yi, et al. "Feature pyramid networks for object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

Feature Pyramid Network (PFN)

- Fast/Faster R-CNN achieves multi-scale by using different anchor sizes and aspect ratios
- Fast/Faster R-CNN is using only one feature scale due to memory limits
  - Not taking advantage of different feature scales to learn object features that come in different sizes
- SSD and YOLO take advantage of the feature pyramids but features are considered independent from one another
- FPN addresses the issue by downsampling, upsampling and merging feature maps of different sizes
  - FPN is simple, scale invariant and has low computational overhead

Different Approaches to Feature Pyramids

Idea of PFN is to combine semantically strong low-resolution features with semantically weak high-resolution features

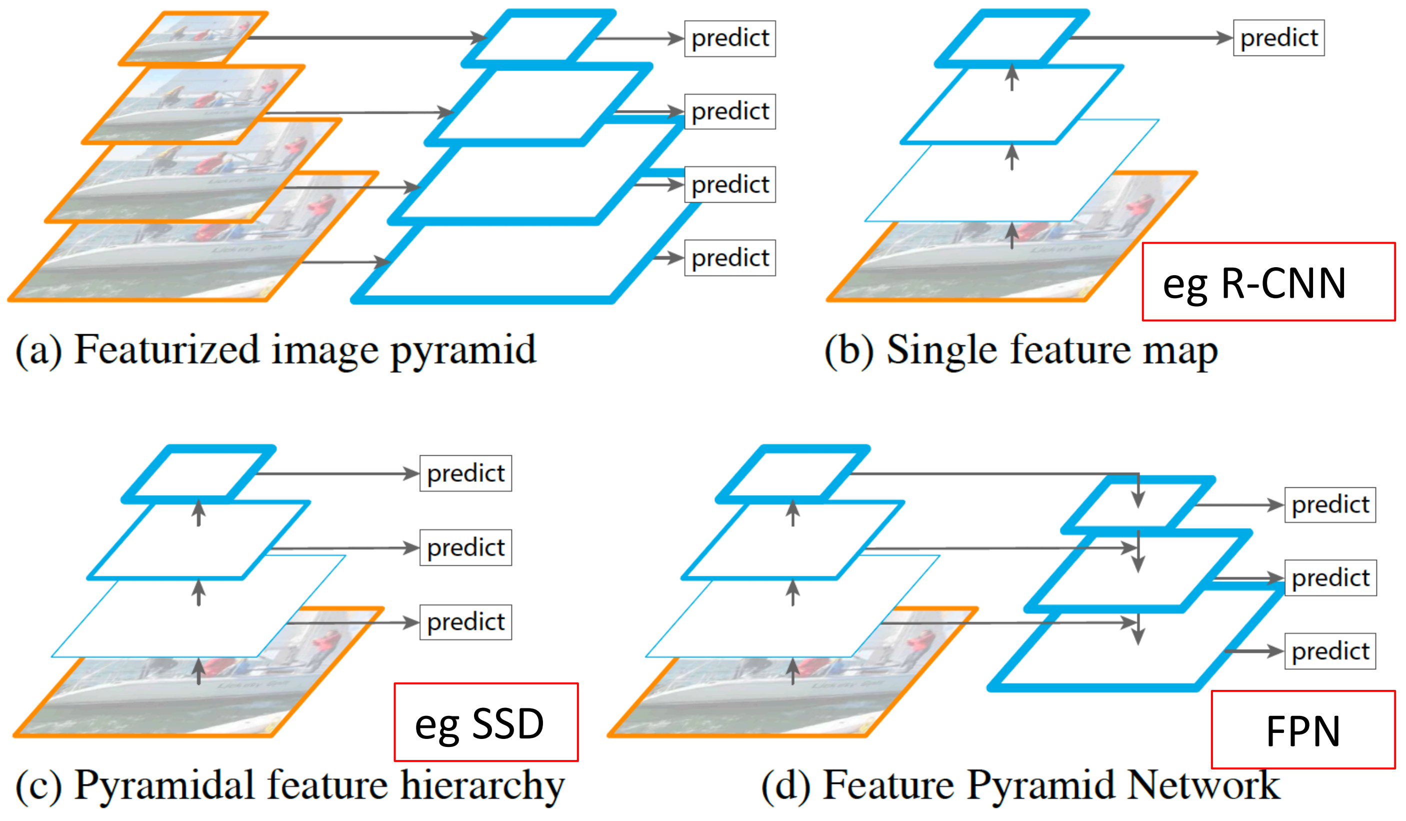


Figure 1. (a) Using an image pyramid to build a feature pyramid. Features are computed on each of the image scales independently, which is slow. (b) Recent detection systems have opted to use only single scale features for faster detection. (c) An alternative is to reuse the pyramidal feature hierarchy computed by a ConvNet as if it were a featurized image pyramid. (d) Our proposed Feature Pyramid Network (FPN) is fast like (b) and (c), but more accurate. In this figure, feature maps are indicate by blue outlines and thicker outlines denote semantically stronger features.

Top-down Feature Pyramids

SharpMask

PFN

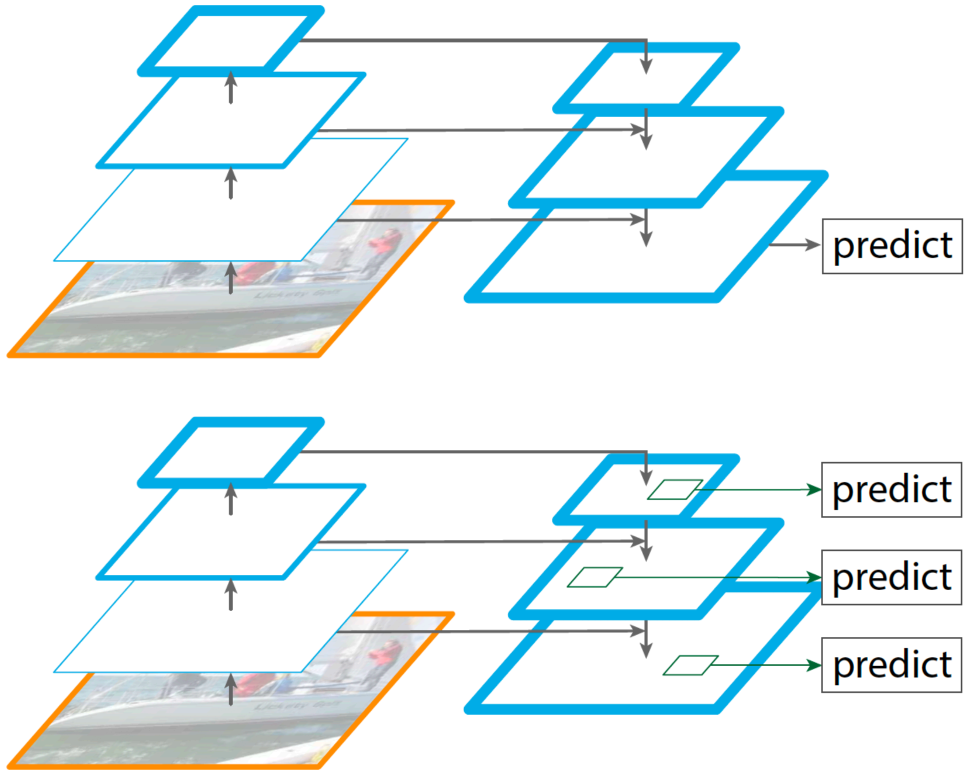
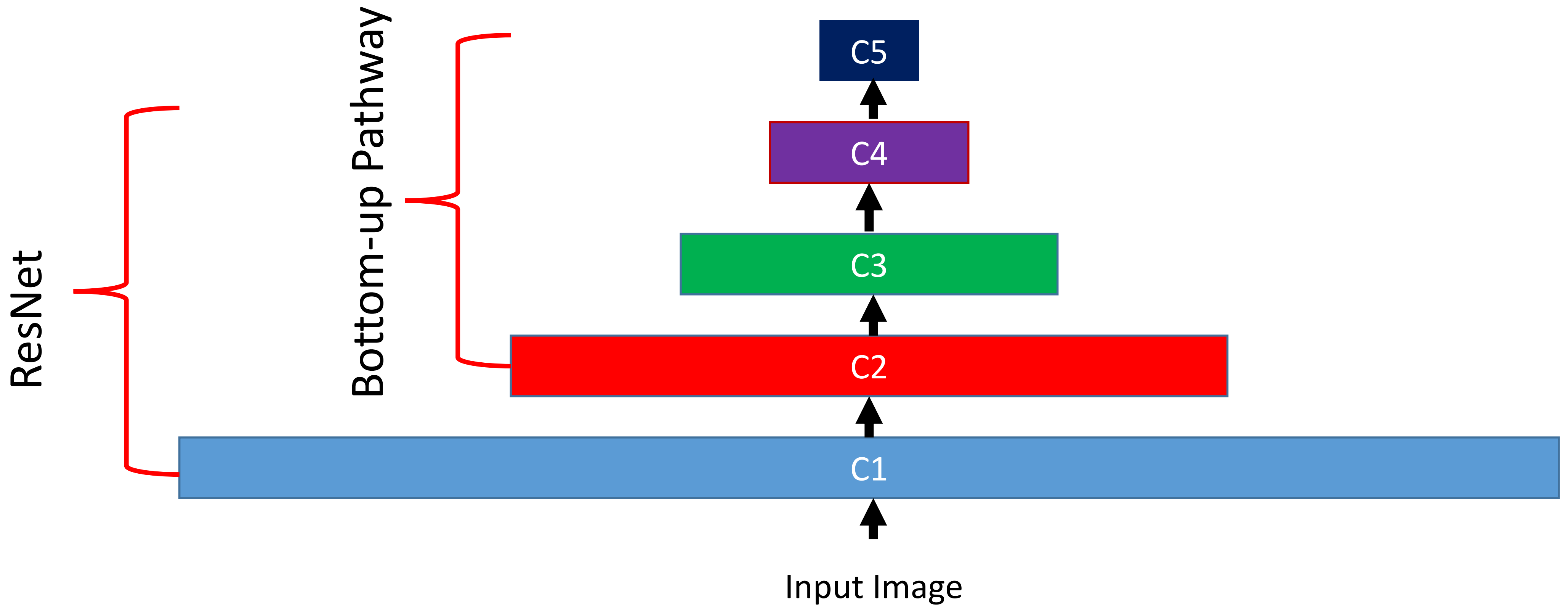
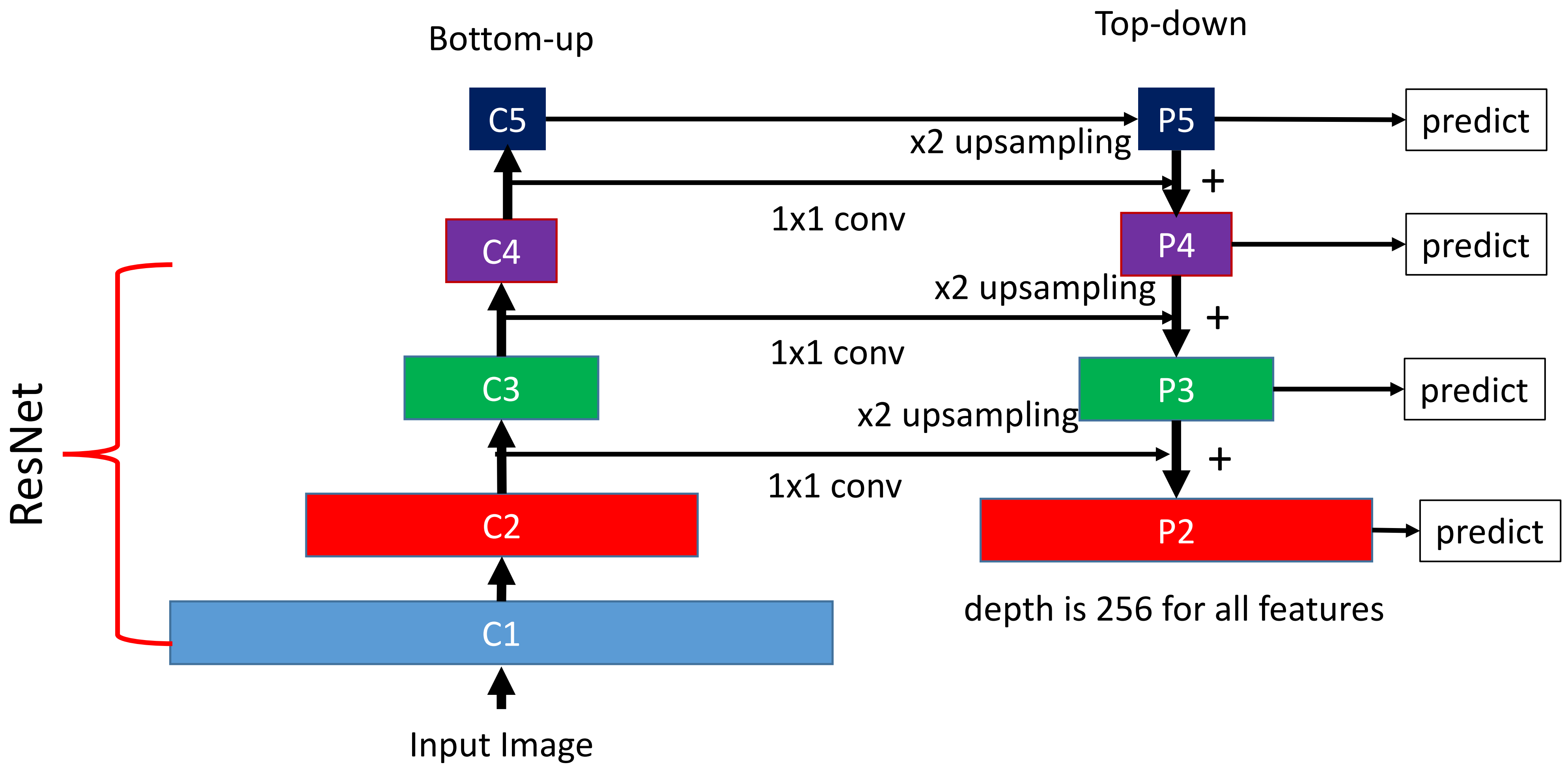


Figure 2. Top: a top-down architecture with skip connections, where predictions are made on the finest level (*e.g.*, [28]). Bottom: our model that has a similar structure but leverages it as a *feature pyramid*, with predictions made independently at all levels.

Bottom-up Pathway



Bottom-up and Top-down Pathways



Lateral Connection

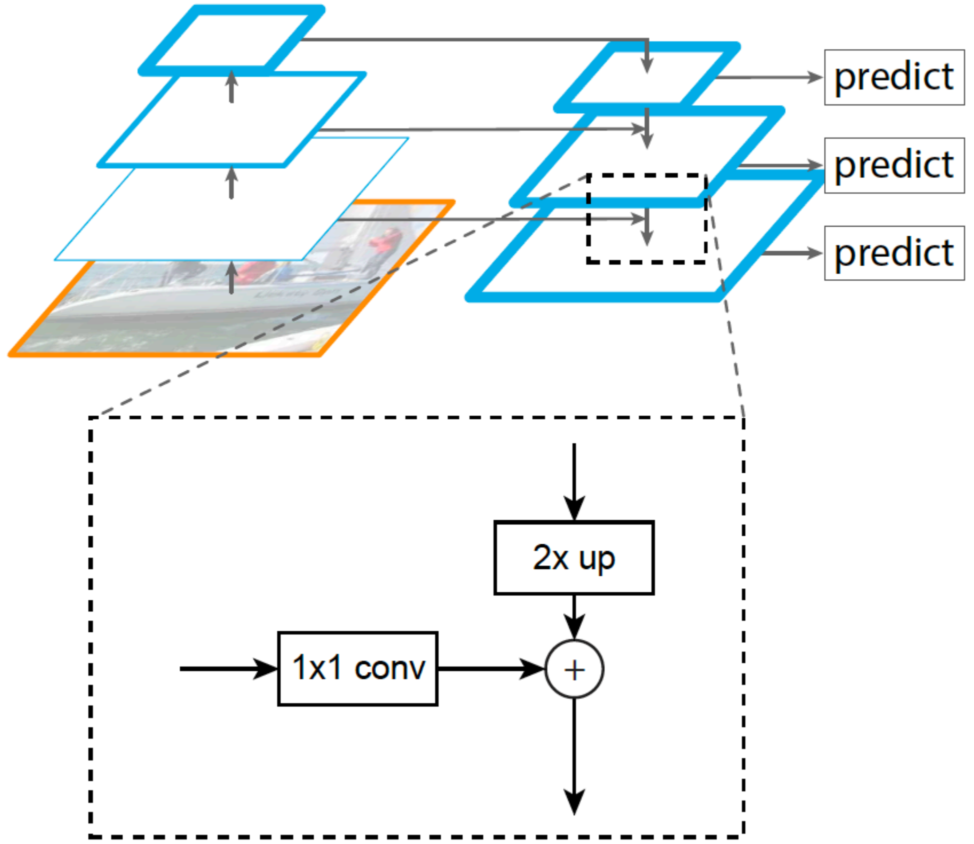
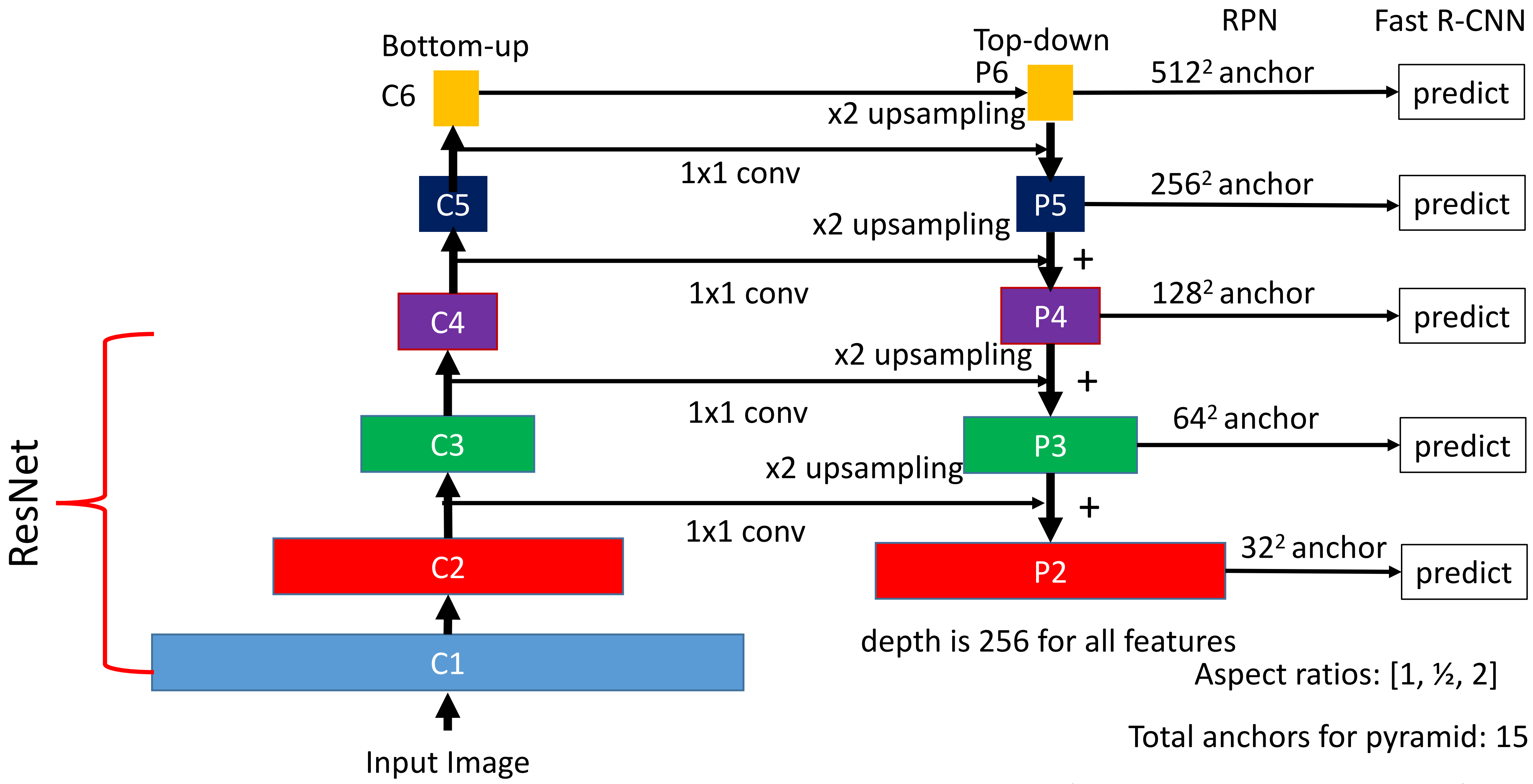


Figure 3. A building block illustrating the lateral connection and the top-down pathway, merged by addition.

PFN for RPN in Faster R-CNN



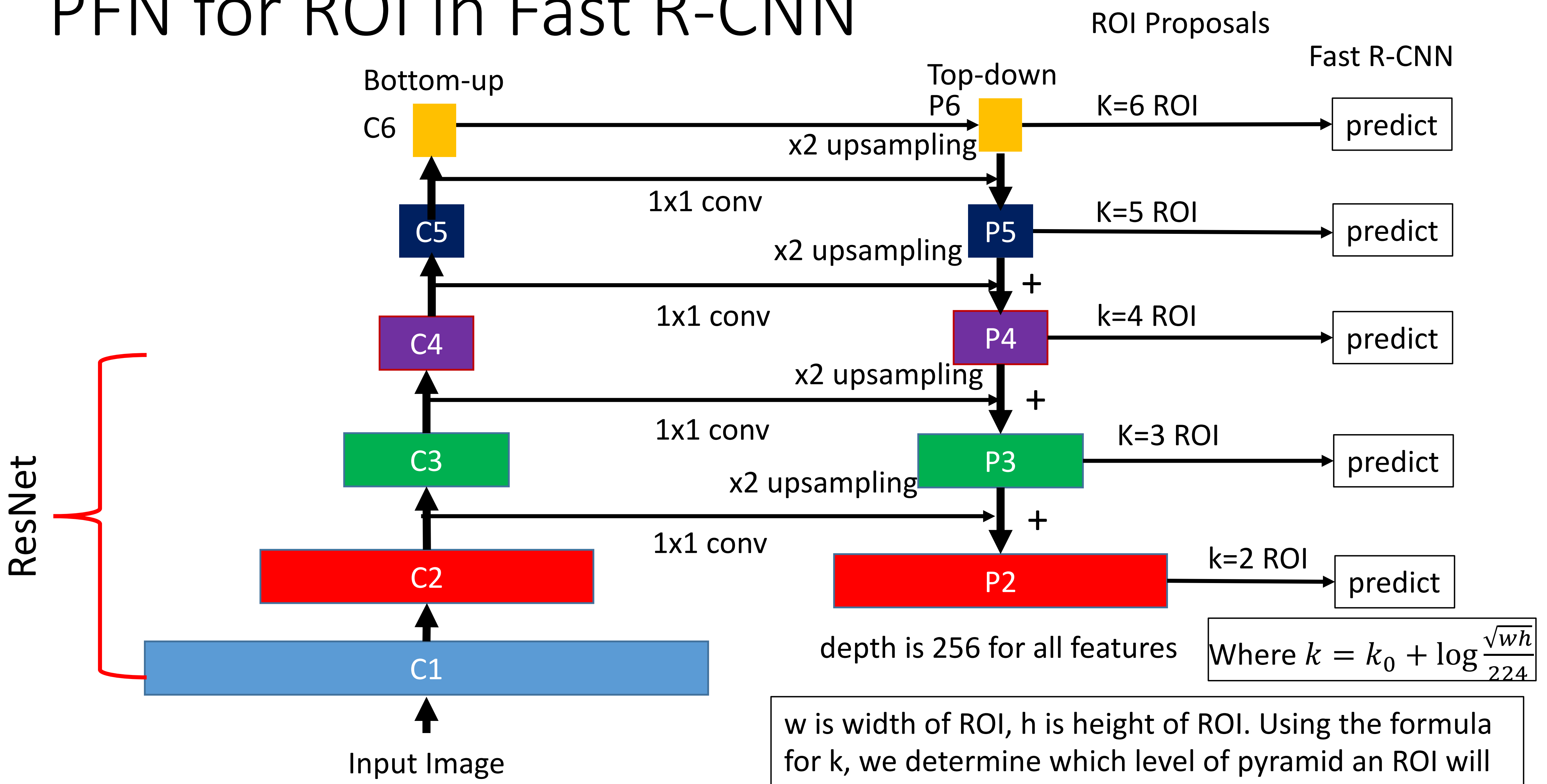
depth is 256 for all features

Aspect ratios: [1, ½, 2]

Total anchors for pyramid: 15

Positive anchors if IoU>0.7, negative anchors if IoU<0.3

PFN for ROI in Fast R-CNN



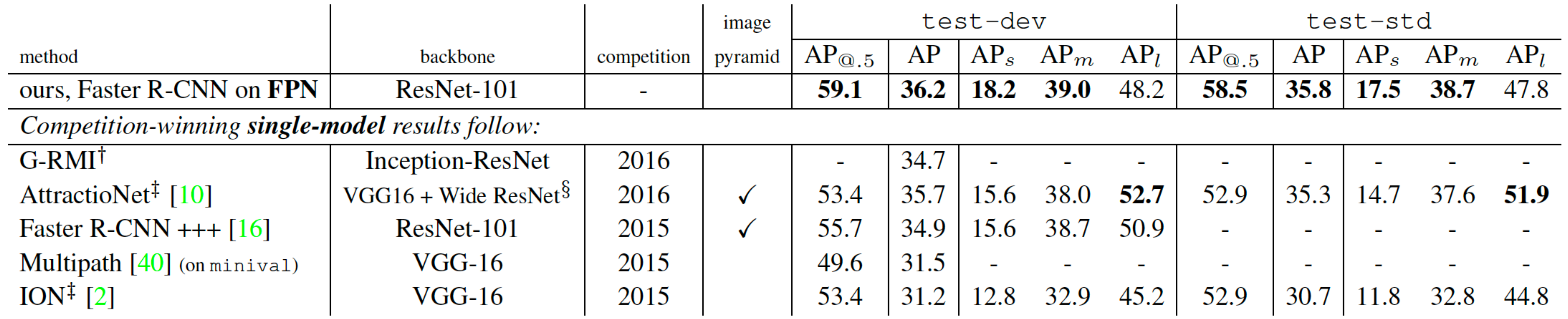
depth is 256 for all features

Where $k = k_0 + \log \frac{\sqrt{wh}}{224}$

w is width of ROI, h is height of ROI. Using the formula for k, we determine which level of pyramid an ROI will be assigned to. $k_0 = 4$

Performance

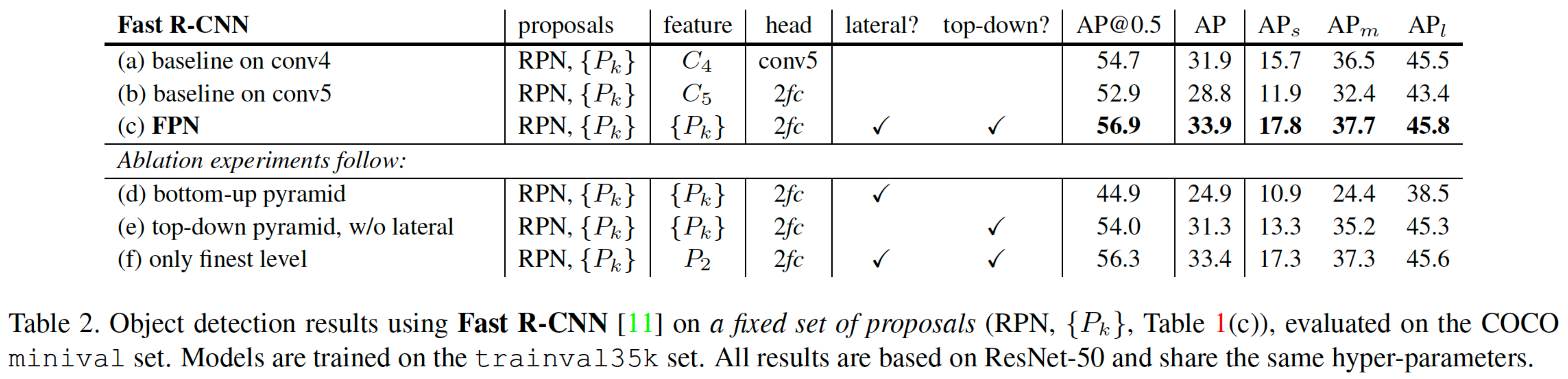
- Coco test-dev : 36.2 AP (Faster R-CNN)



method	backbone	competition	image pyramid	test-dev $AP_{@.5}$	test-dev AP	test-dev AP_s	test-dev AP_m	test-dev AP_l	test-std $AP_{@.5}$	test-std AP	test-std AP_s	test-std AP_m	test-std AP_l
ours, Faster R-CNN on **FPN**	ResNet-101	-		**59.1**	**36.2**	**18.2**	**39.0**	48.2	**58.5**	**35.8**	**17.5**	**38.7**	47.8
*Competition-winning **single-model** results follow:*													
G-RMI[†]	Inception-ResNet	2016		-	34.7	-	-	-	-	-	-	-	-
AttractioNet[‡] [10]	VGG16 + Wide ResNet[§]	2016	✓	53.4	35.7	15.6	38.0	**52.7**	52.9	35.3	14.7	37.6	**51.9**
Faster R-CNN +++ [16]	ResNet-101	2015	✓	55.7	34.9	15.6	38.7	50.9	-	-	-	-	-
Multipath [40] (on minival)	VGG-16	2015		49.6	31.5	-	-	-	-	-	-	-	-
ION[‡] [2]	VGG-16	2015		53.4	31.2	12.8	32.9	45.2	52.9	30.7	11.8	32.8	44.8

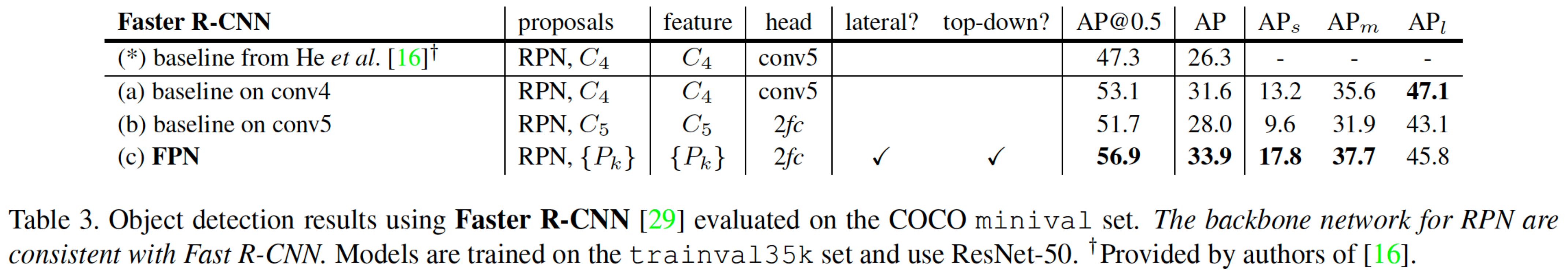
Table 4. Comparisons of **single-model** results on the COCO detection benchmark. Some results were not available on the test-std set, so we also include the test-dev results (and for Multipath [40] on minival). [†]: http://image-net.org/challenges/talks/2016/GRMI-COCO-slidedeck.pdf. [‡]: http://mscoco.org/dataset/#detections-leaderboard. [§]: This entry of AttractioNet [10] adopts VGG-16 for proposals and Wide ResNet [39] for object detection, so is not strictly a single-model result.

Improvement over Baseline Network



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

Table 2. Object detection results using **Fast R-CNN** [11] on *a fixed set of proposals* (RPN, $\{P_k\}$, Table 1(c)), evaluated on the COCO `minival` set. Models are trained on the `trainval35k` set. All results are based on ResNet-50 and share the same hyper-parameters.



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

Table 3. Object detection results using **Faster R-CNN** [29] evaluated on the COCO `minival` set. *The backbone network for RPN are consistent with Fast R-CNN.* Models are trained on the `trainval35k` set and use ResNet-50. †Provided by authors of [16].

RetinaNet

Lin, Tsung-Yi, et al. "Focal loss for dense object detection." Proceedings of the IEEE international conference on computer vision. 2017.

Focal Loss and RetinaNet

- In single-stage detectors (SSD, YOLO), there is big class imbalance between background-foreground anchor boxes
- In two-stage detectors (RCNN, Fast RCNN, Faster RCNN), there is no imbalance due to ROIs that filter out most background regions
- Easy negatives (ie background) should have less contribution to the total loss
  - New loss function: **Focal Loss**
- Create a new single-stage network based on PFN using Focal Loss
  - Result: Fast and high-performing SSD called **RetinaNet**

Approaches on Class Imbalance

- ROI/RPN in RCNN
  - Hard negative mining
- Sampling heuristics (Selective Search, EdgeBoxes)
- Keep a constant background:foreground ratio (eg 1:3)

Problems with SSD

- A large number of anchor boxes : 10k to 100k depending on input resolution and number of levels in the feature pyramid
- Applying sampling heuristics is inefficient
- Category loss function, $\mathcal{L}_{cls}$, is dominated by background or negative anchor boxes drowning foreground or positive anchor boxes

Focal Loss

- Introduce a weighting factor on the loss
- Down weight easy anchor boxes (eg background)
- Focus on hard examples (eg foreground)
- Typical class imbalance is 1:1000

Focal Loss

- Binary Cross-entropy loss (assume bg & fg are equally likely):

$$CE(p, y) = \begin{cases} -\log p & y = 1\ or\ foreground \\ -\log(1-p) & y = -1\ or\ background \end{cases}$$

Where $p \in [0, 1]$ is the model probability estimate when $y = 1$.

$$p_t = \begin{cases} p & y = 1\ or\ foreground \\ (1-p) & y = -1\ or\ background \end{cases}$$

$$CE(p, y) = CE(p_t) = -\log p_t$$

Focal Loss

- For $p_t \gg 0.5$, CE contribution is not negligible. When summed up for a given large number of easy samples, it overwhelms the sparse loss from hard samples (ie object instances)

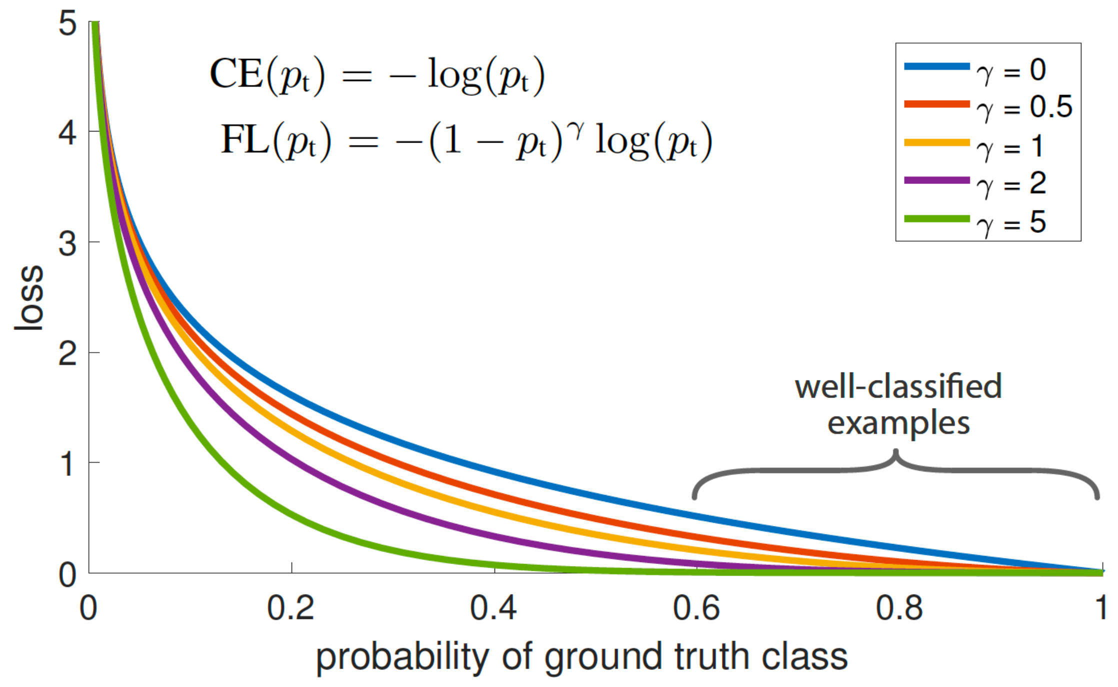


Figure 1. We propose a novel loss we term the *Focal Loss* that adds a factor $(1 - p_t)^\gamma$ to the standard cross entropy criterion. Setting $\gamma > 0$ reduces the relative loss for well-classified examples ($p_t > .5$), putting more focus on hard, misclassified examples. As our experiments will demonstrate, the proposed focal loss enables training highly accurate dense object detectors in the presence of vast numbers of easy background examples.

Balanced Loss Function

$$CE(p_t) = -\alpha_t \log p_t$$

Where $\alpha_t = \begin{cases} 1 & y = 1 \; or \; foreground \\ (1-\alpha) & y = -1 \; or \; background \end{cases}$

Where $\alpha \in [0, 1]$ is inversely proportional to frequency of a sample. Value is small for background (more frequent) and large for foreground (less frequent)

Focal Loss

- Balanced loss function can not differentiate easy from hard samples

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

Where $(1-p_t)^{\gamma}$ is a modulating factor and $\gamma \geq 0$. Empirically, $\gamma = 2$ works best. CE is equal to the normal cross-entropy loss when $\gamma = 0$.

- Balanced focal loss function:

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

α=0.25 works best for $\gamma = 2$

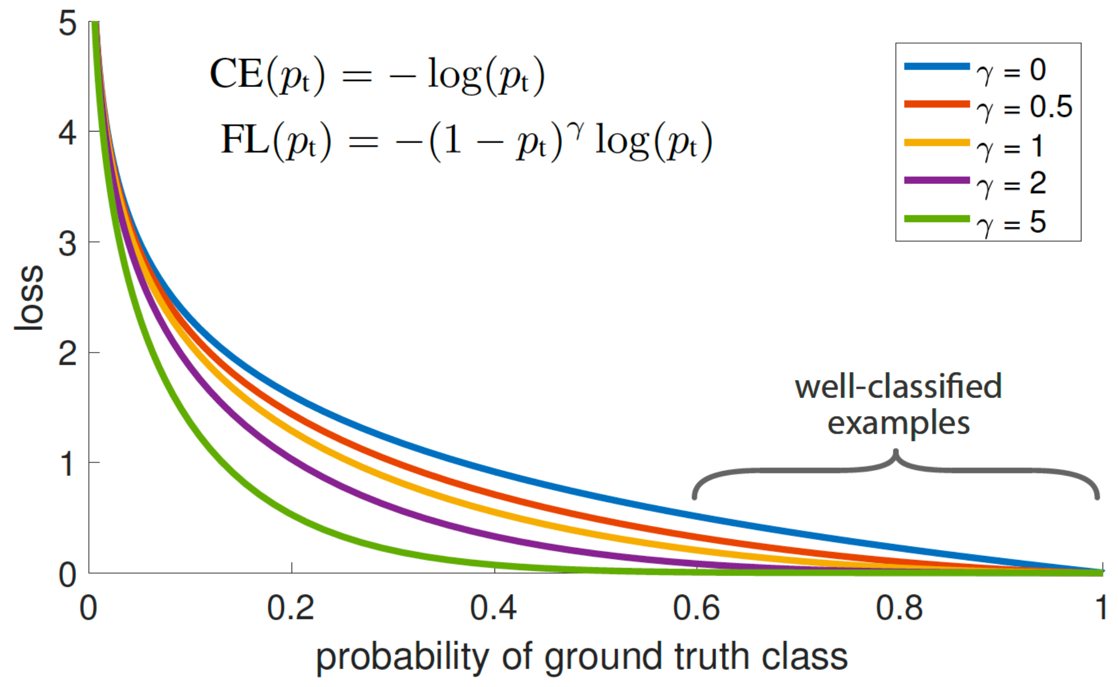


Figure 1. We propose a novel loss we term the *Focal Loss* that adds a factor $(1-p_t)^{\gamma}$ to the standard cross entropy criterion. Setting $\gamma > 0$ reduces the relative loss for well-classified examples ($p_t > .5$), putting more focus on hard, misclassified examples. As our experiments will demonstrate, the proposed focal loss enables training highly accurate dense object detectors in the presence of vast numbers of easy background examples.

RetinaNet

Pyramids are from P_3 to P_7. Only P_3 to P_5 are shown.

K classes, A anchor boxes

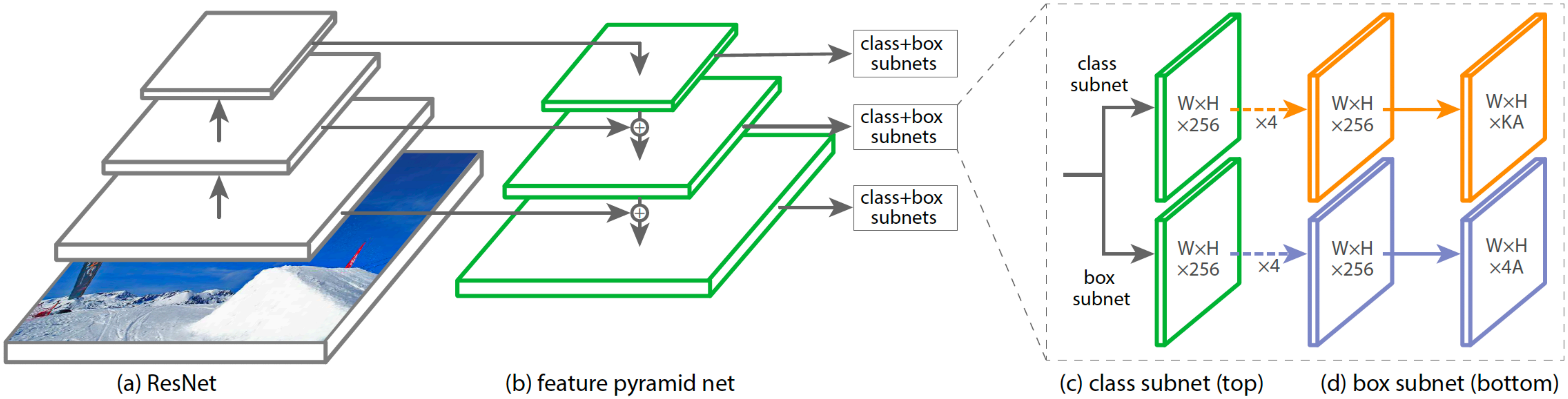


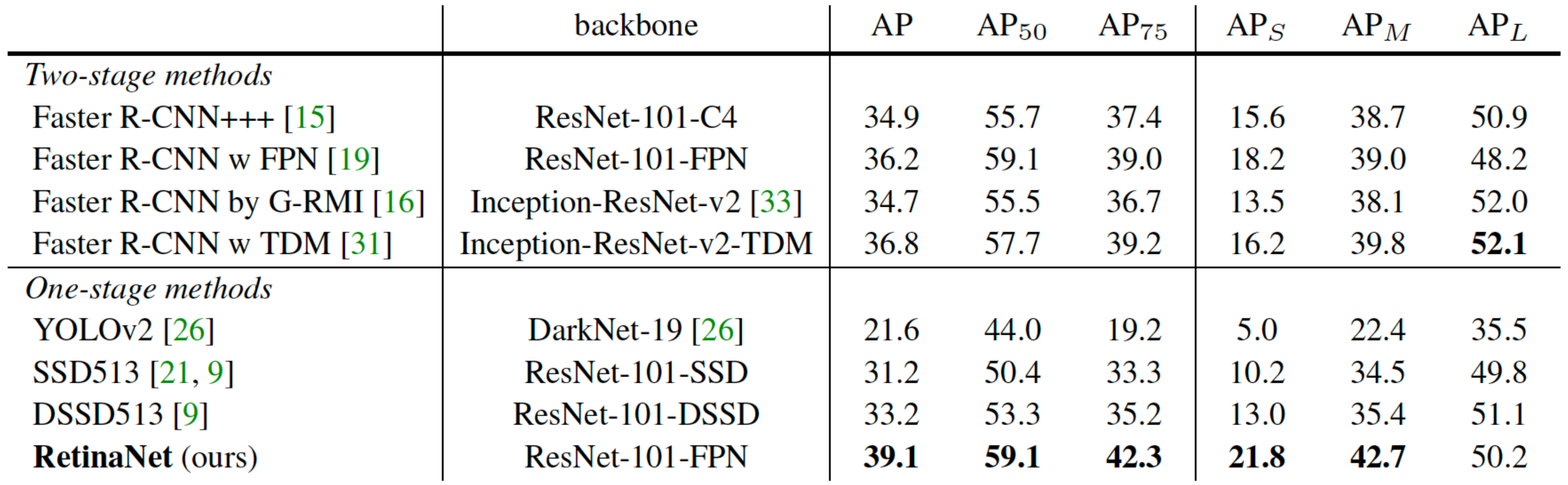
Figure 3. The one-stage **RetinaNet** network architecture uses a Feature Pyramid Network (FPN) [19] backbone on top of a feedforward ResNet architecture [15] (a) to generate a rich, multi-scale convolutional feature pyramid (b). To this backbone RetinaNet attaches two subnetworks, one for classifying 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.

RetinaNet

- Last layer set to:
  - Bias: $-\log\frac{1-\pi}{\pi}$, where $\pi = 0.01$ to give foreground objects small positive probabilities and prevent unstable training
- All other biases are set to: 0.0
- All weights: Gaussian with $\sigma = 0.01$

Performance

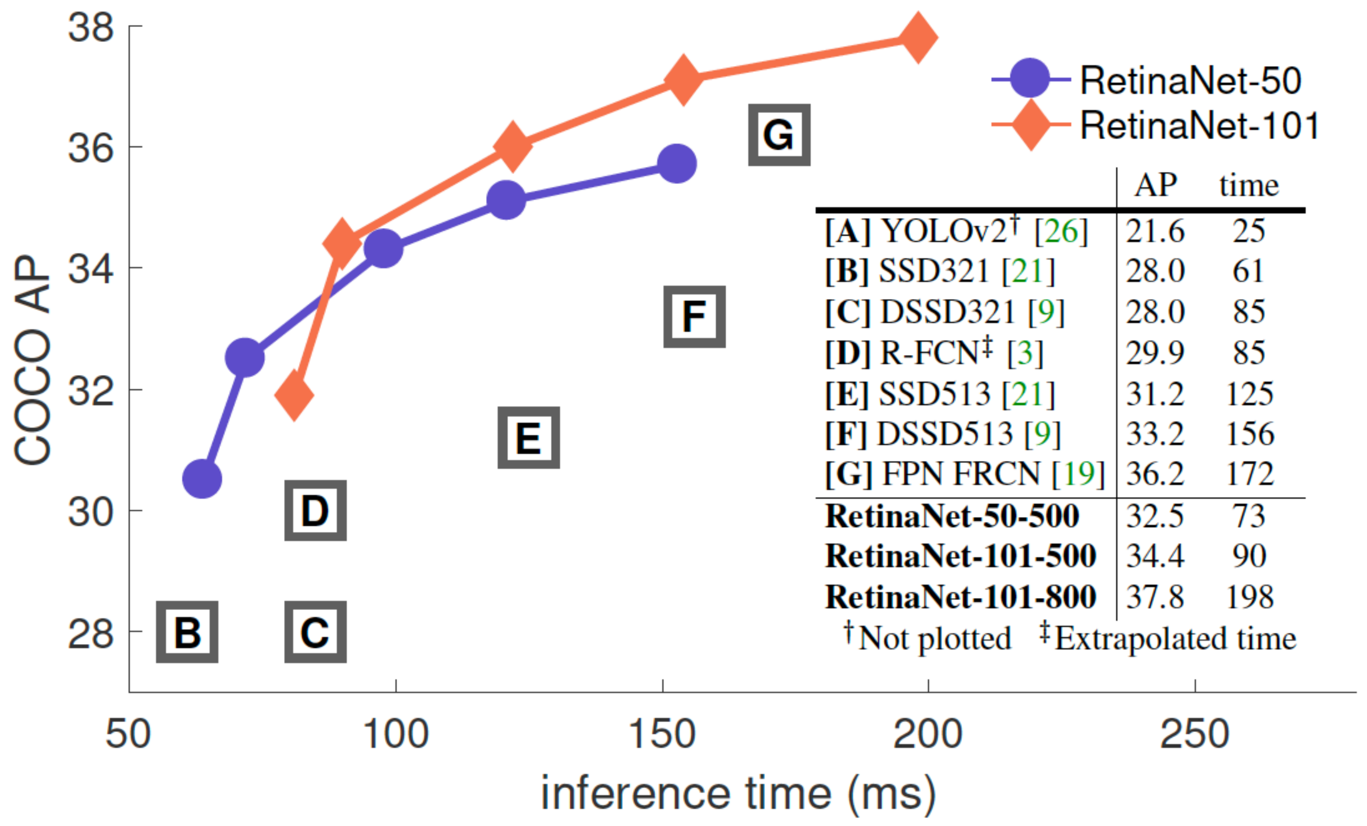
- Coco test-dev : 39.1 AP at 5fps



	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [15]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [19]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [16]	Inception-ResNet-v2 [33]	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**
One-stage methods							
YOLOv2 [26]	DarkNet-19 [26]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [21, 9]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [9]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet (ours)	ResNet-101-FPN	**39.1**	**59.1**	**42.3**	**21.8**	**42.7**	50.2

Table 2. **Object detection** *single-model* results (bounding box AP), *vs.* state-of-the-art on COCO `test-dev`. We show results for our RetinaNet-101-800 model, trained with scale jitter and for 1.5$\times$ longer than the same model from Table 1e. Our model achieves top results, outperforming both one-stage and two-stage models. For a detailed breakdown of speed versus accuracy see Table 1e and Figure 2.

RetinaNet Performance



	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 show variants of RetinaNet with ResNet-50-FPN (blue circles) and ResNet-101-FPN (orange diamonds) at five scales (400-800 pixels). Ignoring the low-accuracy regime (AP<25), RetinaNet forms an upper envelope of all current detectors, and a variant trained for longer (not shown) achieves 39.1 AP. Details are given in §5.

FSAF

Zhu, Chenchen, Yihui He, and Marios Savvides. "Feature Selective Anchor-Free Module for Single-Shot Object Detection." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019.

Feature Selective Anchor-Free Module

- Anchor-based detections have 2 limitations:
    - Heuristic-guided feature selection
    - Overlap-based (IoU) anchor sampling
- The selected feature level may not be optimal to train the object instance
- Why not learn the feature level assignment as well?

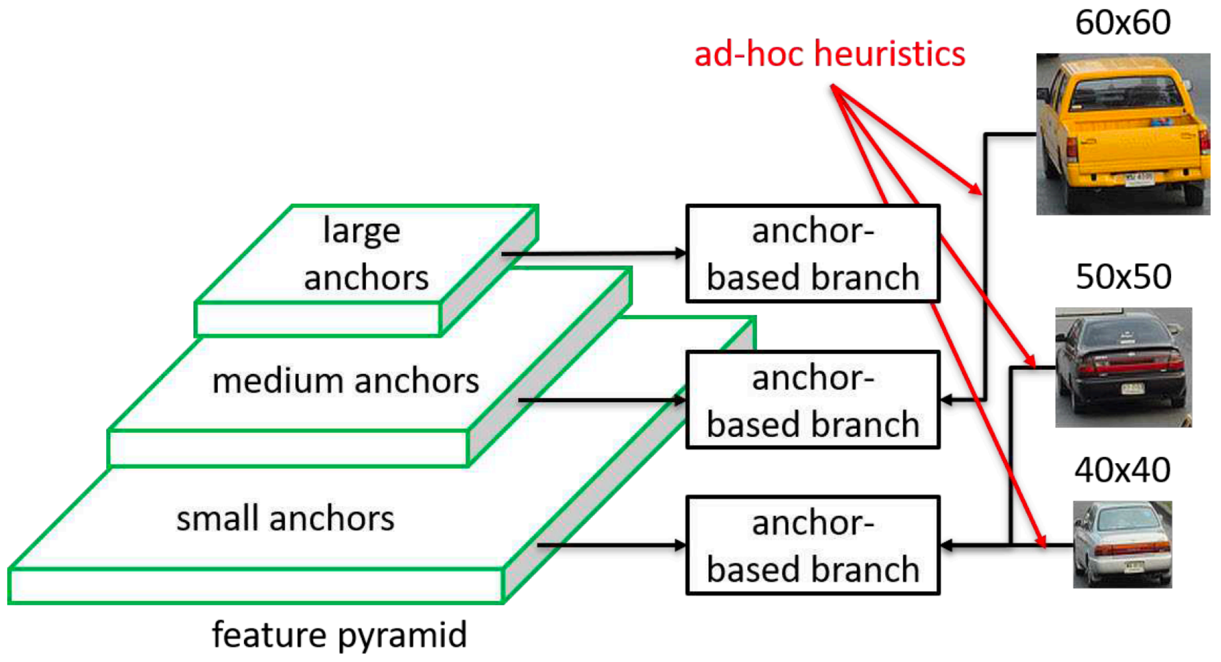


Figure 2: Selected feature level in anchor-based branches may not be optimal.

40x40 and 50x50 car share the same feature level, may not be optimal

FSAF

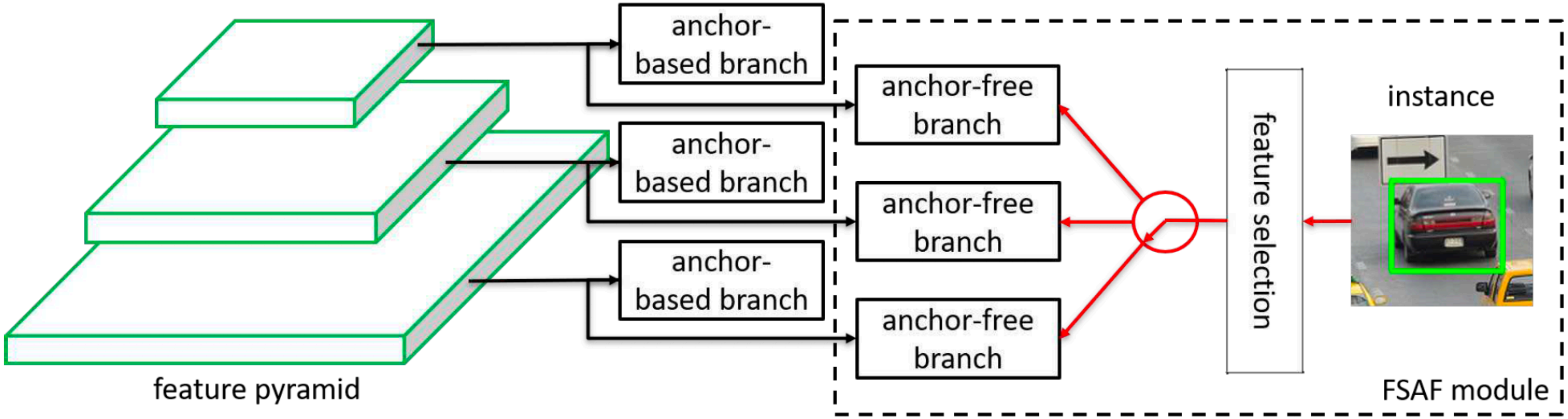


Figure 3: The general concept of our FSAF module plugged into conventional anchor-based detection methods. During training, each instance is assigned to a pyramid level via online feature selection for setting up supervision signals.

An instance is not assigned to an anchor box. Instead, the feature selection module selects the best feature level for a given instance. Note: In the feature pyramid, only 3 (P_3 to P_5) of the 5 feature levels (P_3 to P_7) are shown.

RetinaNet + FSAF

Predicts the probability in each spatial location for K objects

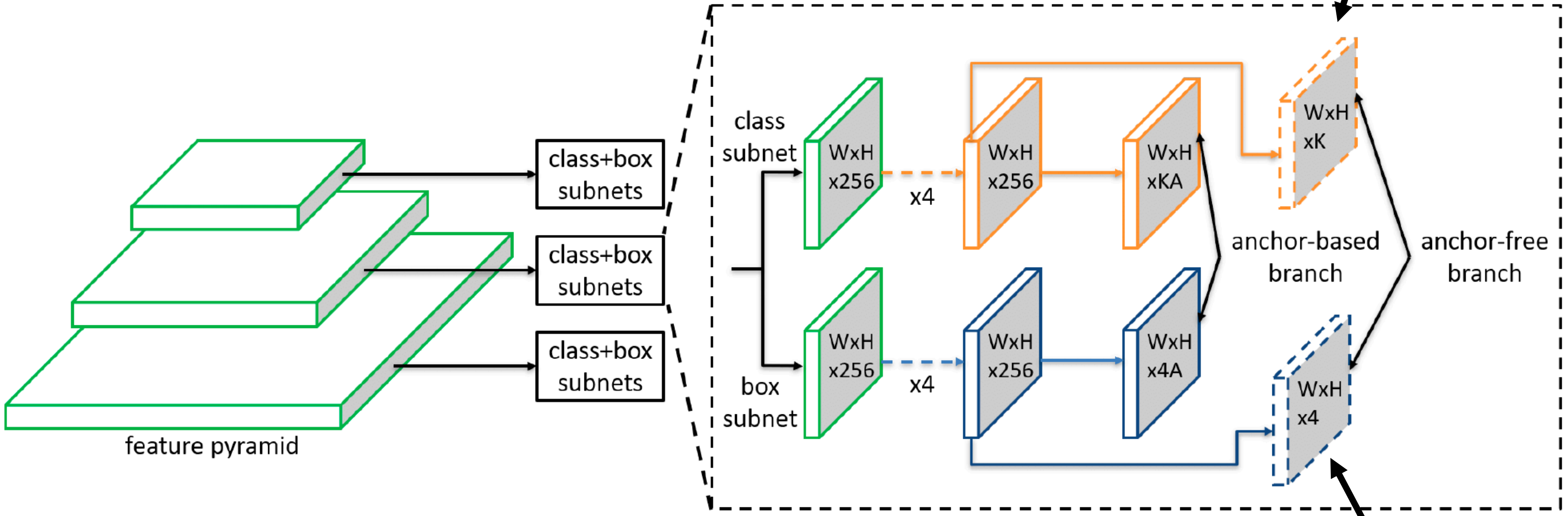
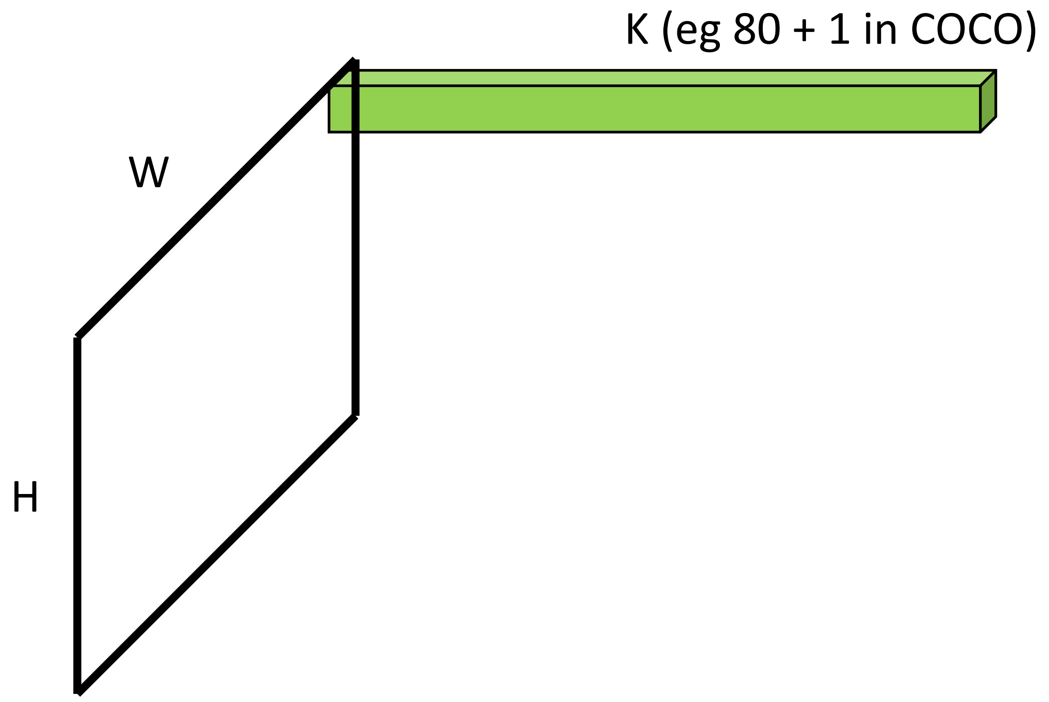


Figure 4: Network architecture of RetinaNet with our FSAF module. The FSAF module only introduces two additional conv layers (dashed feature maps) per pyramid level, keeping the architecture fully convolutional.

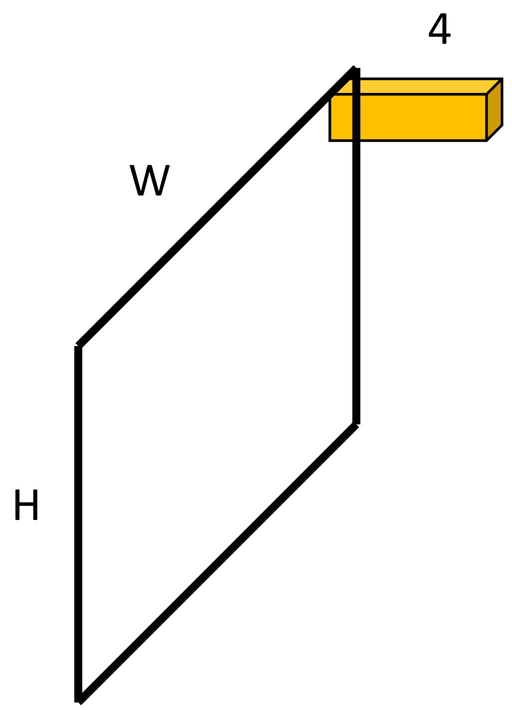
Only 3 out of 5 levels are shown for clarity

Predicts box offsets in each spatial location

FSAF Modules



Class Predictions



Offset Predictions

Ground Truths

- Class: k
- Bounding box: $b = [x, y, w, h]$ where (x, y) is the center and (w, h) is the width and height
- Given pyramid level P_l, the projection of b to l is $b_p^l = \left[x_p^l, y_p^l, w_p^l, h_p^l\right]$ where $b_p^l = \frac{b}{2^l}$
- Introduce effective box: $b_e^l = [x_e^l, y_e^l, w_e^l, h_e^l] = \left[x_p^l, y_p^l, \epsilon_e w_p^l, \epsilon_e h_p^l\right]$which is a box proportional to b_p^l by ϵ_e with typical value of 0.2
  - When two b_e^l overlap, the smaller one is considered
- Introduce ignoring box: $b_i^l = \left[x_i^l, y_i^l, w_i^l, h_i^l\right]$= $\left[x_p^l, y_p^l, \epsilon_i w_p^l, \epsilon_i h_p^l\right]$ which is a box proportional to b_p^l by ϵ_i with typical value of 0.5
  - If this exists, there is also corresponding b_i^{l-1} and b_i^{l+1} for the $l-1$ and $l+1$ layers respectively (we ignore those objects as well)

White: effective region
Gray: (Ignoring minus effective) region
Black: Not included in loss computation

Class Ground (GT) Truth

- The output has K maps
- A ground truth instance (eg "car") affects the k-th feature map (eg feature corresponding to car)
- GT is on non-ignoring/effective regions (White)
- Focal loss is applied
  - Normalized by the number of pixels on non-ignoring regions

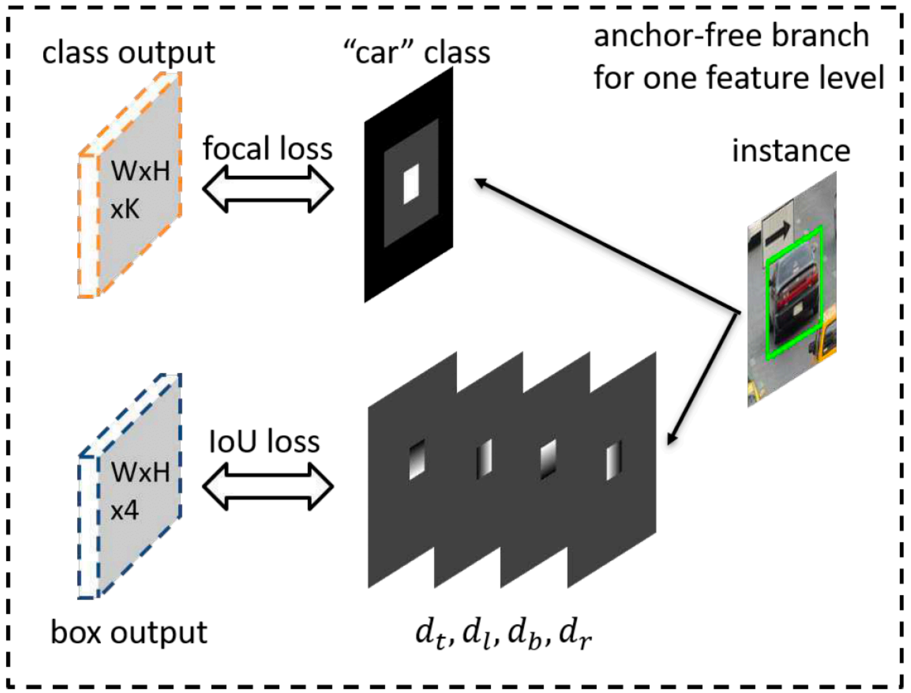
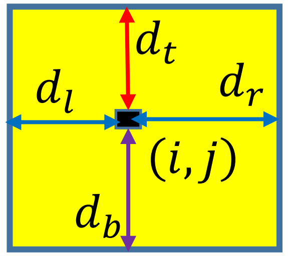


Figure 5: Supervision signals for an instance in one feature level of the anchor-free branches. We use focal loss for classification and IoU loss for box regression.

White: effective region
Gray: (Ignoring – effective) region
Black: Not included in loss computation

Offset Ground (GT) Truth

- GT is 4 offset maps
- GT on the non-ignoring/effective regions (White)
- For each (i,j) in GT, the projected box b_p^l is represented as 4-dim:
    - $\mathbf{d}_{i,j}^l = \left[\mathrm{d}_{t_{i,j}}^l, \mathrm{d}_{l_{i,j}}^l, \mathrm{d}_{b_{i,j}}^l, \mathrm{d}_{r_{i,j}}^l\right]$ where $\mathrm{d}_{t_{i,j}}^l, \mathrm{d}_{l_{i,j}}^l, \mathrm{d}_{b_{i,j}}^l, \mathrm{d}_{r_{i,j}}^l$ are the distances from top, left, bottom and right boundaries of b_p^l relative to (i,j)
    - GT: $\frac{\mathbf{d}_{i,j}^l}{S}$, where $S = 4.0$
- IoU loss is applied



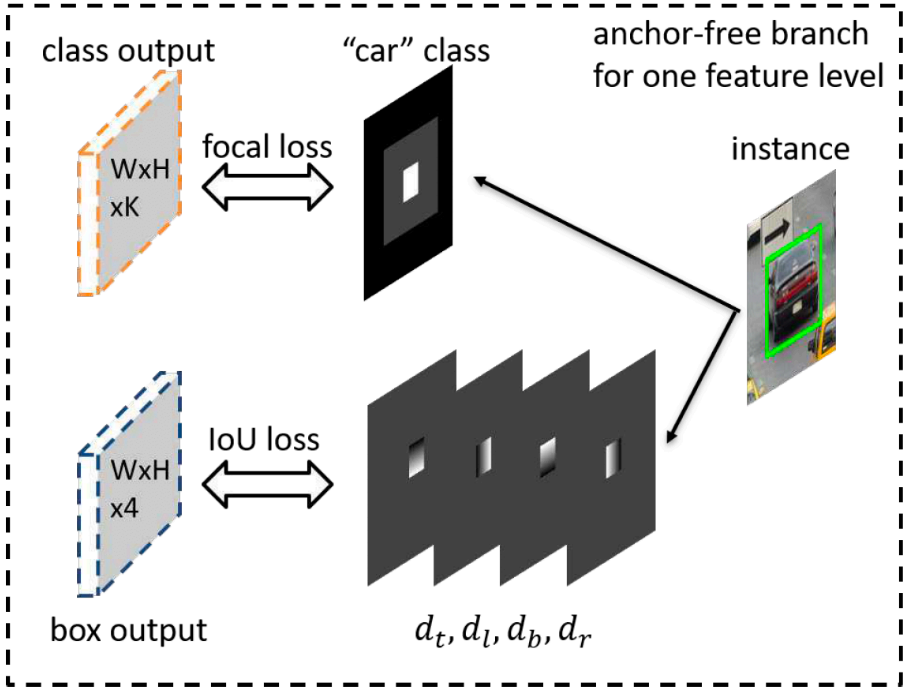
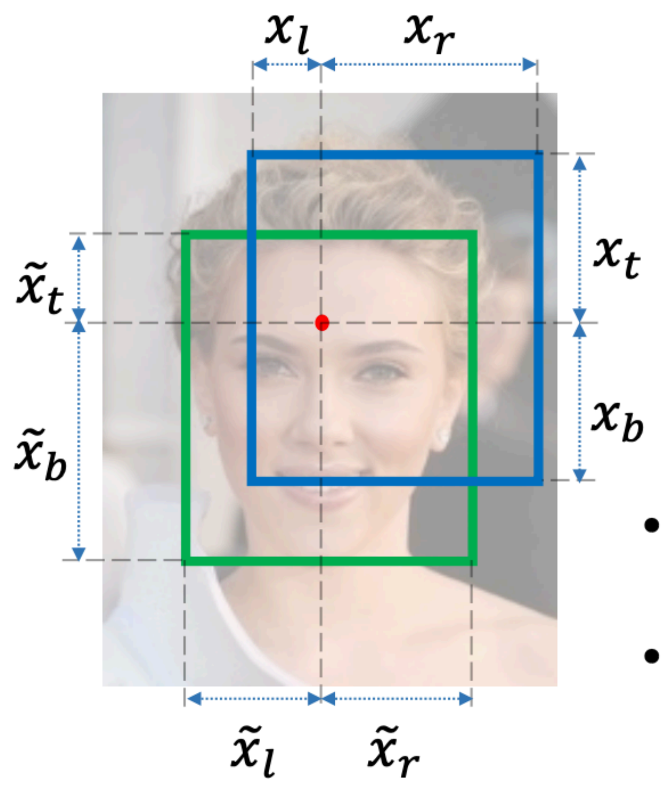


Figure 5: Supervision signals for an instance in one feature level of the anchor-free branches. We use focal loss for classification and IoU loss for box regression.

IoU Loss

Yu, Jiahui, et al. "Unitbox: An advanced object detection network." *Proceedings of the 24th ACM international conference on Multimedia*. ACM, 2016.



🟩 Ground truth: $\tilde{x} = (\tilde{x}_t, \tilde{x}_b, \tilde{x}_l, \tilde{x}_r)$

🟦 Prediction: $x = (x_t, x_b, x_l, x_r)$

- ℓ_2 *loss* $= ||$ 🟦 $-$ 🟩 $||_2^2$
- *IoU loss* $= -\ln \dfrac{Intersection(\text{🟦},\text{🟩})}{Union(\text{🟦},\text{🟩})}$

Figure 1: Illustration of IoU loss and ℓ_2 loss for pixel-wise bounding box prediction.

Offset Prediction

- For (i, j), prediction is: $\boldsymbol{o}_{i,j}^{l} = \left[o_{t_{i,j}}^{l}, o_{l_{i,j}}^{l}, o_{b_{i,j}}^{l}, o_{r_{i,j}}^{l}\right]$
- Actual predicted box (upper left corner, lower right corner):

$$\widehat{\boldsymbol{d}}_{p}^{l} = \left[i - So_{t_{i,j}}^{l}, j - So_{l_{i,j}}^{l}, i + So_{b_{i,j}}^{l}, j + So_{r_{i,j}}^{l}\right] 2^{l}$$

Where 2^{l} is the scaling factor of the pyramid level.

- The score and class are dictated by the K-dim vector prediction at (i, j)

Feature Selection

To find the optimal feature level, FSAF module selects the best P_l based on the smallest loss on the instance content, instead of the size of instance box as in anchor-based methods.

Only the feature with lowest loss is selected for training

Similar to heuristic selector of PFN $k = k_0 + \log \frac{\sqrt{wh}}{224}$ (Slide 74)

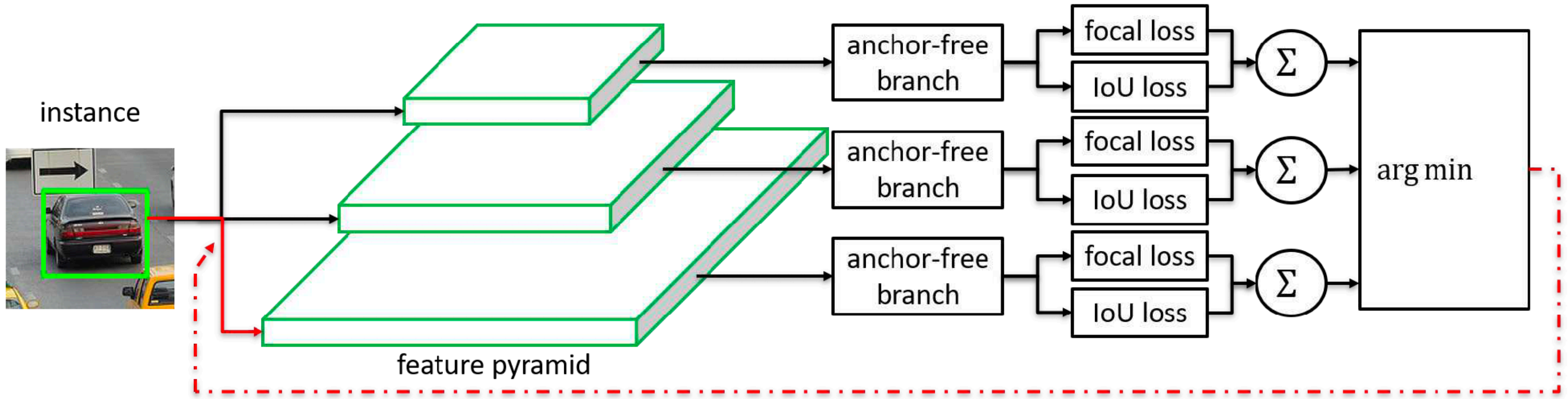


Figure 6: Online feature selection mechanism. Each instance is passing through all levels of anchor-free branches to compute the averaged classification (focal) loss and regression (IoU) loss over effective regions. Then the level with minimal summation of two losses is selected to set up the supervision signals for that instance.

Loss Functions

- Given an instance I and pyramid level P_l
- Classification Focal Loss: $L_{FL}^{I}(l) = \frac{1}{N\left(b_e^l\right)} \sum_{i,j \in b_e^l} FL(l,i,j)$
- Regression IoU Loss: $L_{IoU}^{I}(l) = \frac{1}{N\left(b_e^l\right)} \sum_{i,j \in b_e^l} IoU(l,i,j)$
- Where $N\left(b_e^l\right)$ is the number of pixels in effective region b_e^l
- For all levels, the best feature for instance I is therefore:

$$l^* = arg \min_l L_{FL}^{I}(l) + L_{IoU}^{I}(l)$$

Joint Inference and Training

- RetinaNet + FSAF
- Inference:
    - For FSAF, decode only top 1k scoring locations with $thresh = 0.05$ on each level.
    - Outputs of FSAF are merged with outputs of RetinaNet
    - Both outputs are filtered by NMS with $thresh = 0.5$
- Initialization: Same as RetinaNet except non-classification layer bias is set to 0.1, Backbone pre-trained on ImageNet1k
- Loss Function: RetinaNet Loss + λFSAF Loss
    - RetinaNet Loss: L^{ab}
    - FSAF Loss: $L^{af}_{cls} + L^{af}_{reg}$
    - λ=0.5
- Augmentation: Flipping

Performance

- Coco test-dev : 42.8 AP (ResNet 101), 44.6 AP (ResNeXt-64x4d-101-FPN)

Method	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Multi-shot detectors							
CoupleNet [42]	ResNet-101	34.4	54.8	37.2	13.4	38.1	50.8
Faster R-CNN+++ [28]		34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w/ FPN [21]		36.2	59.1	39.0	18.2	39.0	48.2
Regionlets [35]		39.3	59.8	n/a	21.7	43.7	50.9
Fitness NMS [31]		41.8	60.9	44.9	21.5	45.0	57.5
Cascade R-CNN [3]		42.8	62.1	46.3	23.7	45.5	55.2
Deformable R-FCN [4]	Aligned-Inception-ResNet	37.5	58.0	n/a	19.4	40.1	52.5
Soft-NMS [2]		40.9	62.8	n/a	23.3	43.6	53.3
Deformable R-FCN + SNIP [30]	DPN-98	45.7	67.3	51.1	29.3	48.8	57.1
Single-shot detectors							
YOLOv2 [27]	DarkNet-19	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [24]	ResNet-101	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [8]		33.2	53.3	35.2	13.0	35.4	51.1
RefineDet512 [37] (single-scale)		36.4	57.5	39.5	16.6	39.9	51.4
RefineDet [37] (multi-scale)		41.8	62.9	45.7	25.6	45.1	**54.1**
RetinaNet800 [22]		39.1	59.1	42.3	21.8	42.7	50.2
GHM800 [18]		39.9	60.8	42.5	20.3	43.6	**54.1**
Ours800 (single-scale)		40.9	61.5	44.0	24.0	44.2	51.3
Ours (multi-scale)		**42.8**	**63.1**	**46.5**	**27.8**	**45.5**	53.2
CornerNet511 [17] (single-scale)	Hourglass-104	40.5	56.5	43.1	19.4	42.7	53.9
CornerNet [17] (multi-scale)		42.1	57.8	45.3	20.8	44.8	56.7
GHM800 [18]	ResNeXt-101	41.6	62.8	44.2	22.3	45.1	**55.3**
Ours800 (single-scale)		42.9	63.8	46.3	26.6	46.2	52.7
Ours (multi-scale)		**44.6**	**65.2**	**48.6**	**29.7**	**47.1**	54.6

Table 3: Object detection results of our best *single* model with the FSAF module vs. state-of-the-art single-shot and multi-shot detectors on the COCO `test-dev`.

Example Results

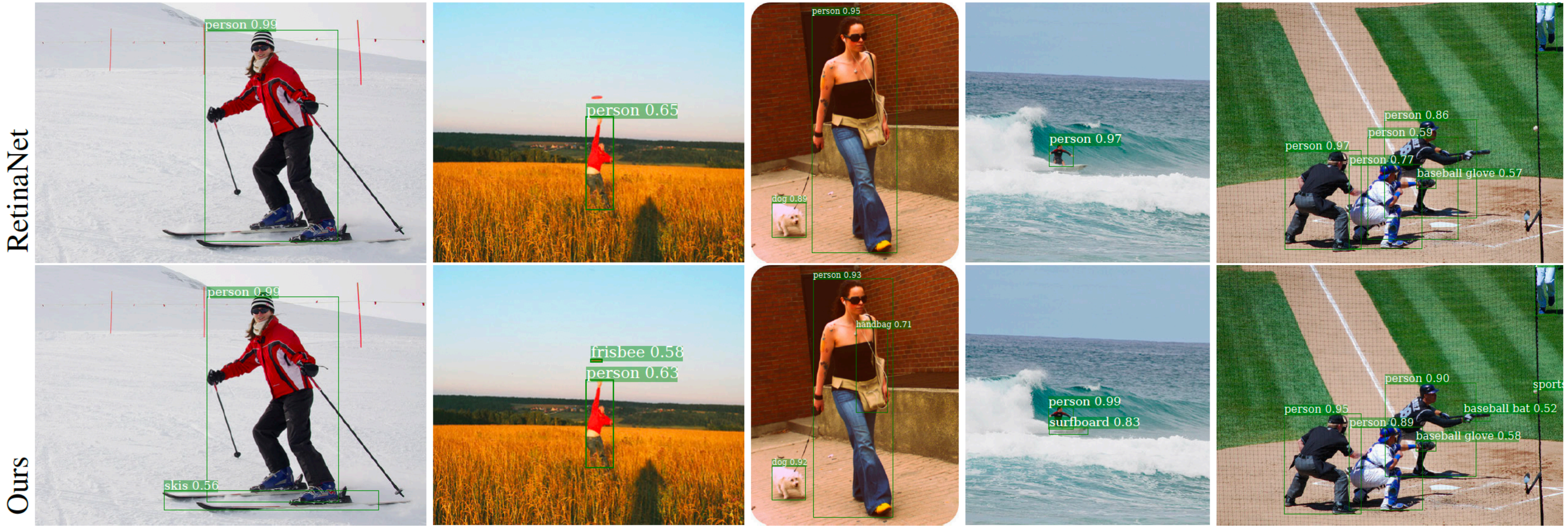


Figure 7: More qualitative comparison examples between anchor-based RetinaNet (top, Table 1 1st entry) and our detector with additional FSAF module (bottom, Table 1 5th entry). Both are using ResNet-50 as backbone. Our FSAF module helps finding more challenging objects, such as thin objects (skis, frisbee, handbag, surfboard) and small objects (baseball).

FCOS

FCOS: Fully Convolutional One-Stage Object Detection;
Tian Zhi, Chunhua Shen, Hao Chen, and Tong He;
In: Proc. Int. Conf. Computer Vision (ICCV), 2019.
arXiv preprint arXiv:1904.01355

FCOS

- Anchor-based detectors’ performance is dependent on hyperparameters (# of anchor boxes, aspect ratios, sizes)
- Anchor-based detectors could not handle small objects or objects with large shape variations
- To achieve high recall rate, anchor-based detectors densely pack anchor boxes on the image (~180k for PFN) causing more background:foreground class imbalance
- Anchor-based detectors involve complex IoU computations
- FCOS: no anchor boxes, regress 4D vector offset to bounding box, similar idea to FCN in segmentation

FCOS

- FCOS does not use anchor boxes but directly regresses the bounding box offsets

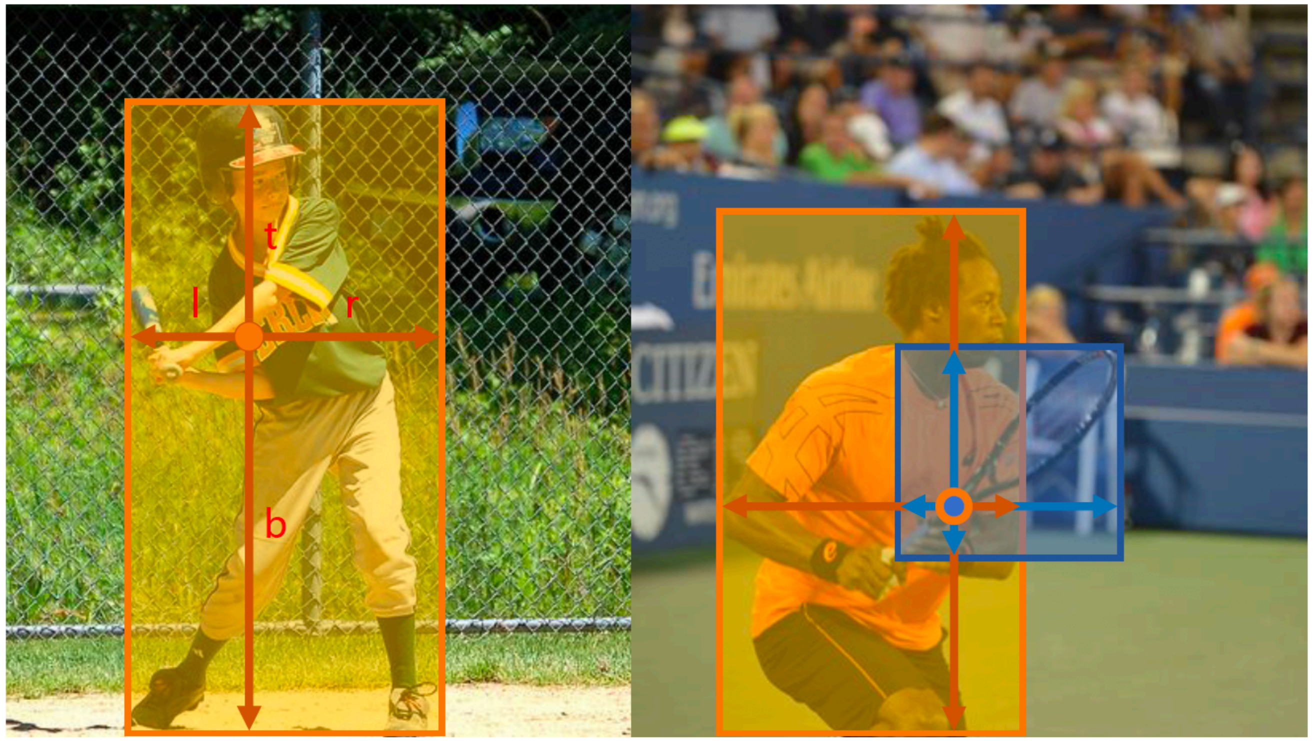


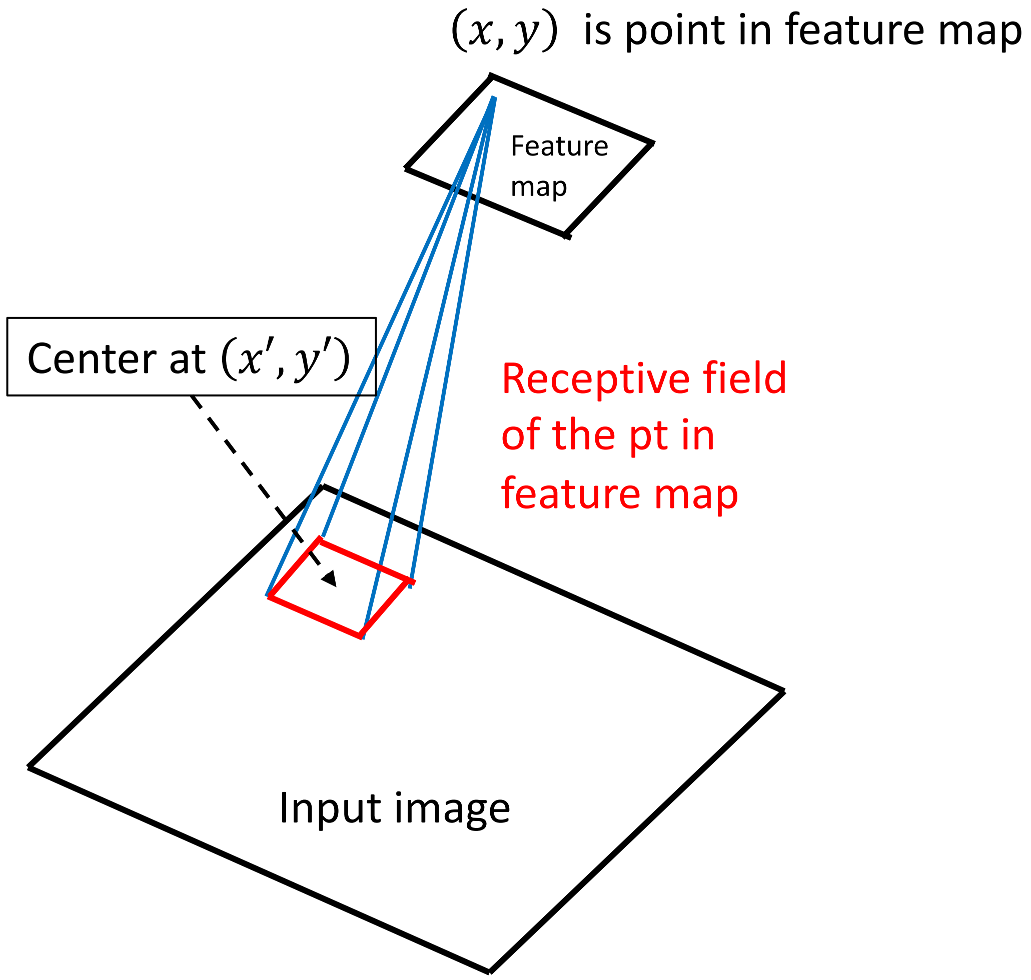
Figure 1 – As shown in the left image, FCOS works by predicting a 4D vector (l, t, r, b) encoding the location of a bounding box at each foreground pixel (supervised by ground-truth bounding box information during training). The right plot shows that when a location residing in multiple bounding boxes, it can be ambiguous in terms of which bounding box this location should regress.

FCOS

- Let $F_i \in \mathbb{R}^{H \times W \times C}$ be feature map at layer i
- s is the stride
- Ground truth bounding boxes $\{B_i\}$ where $B_i = \left(x_0^{(i)}, y_0^{(i)}, x_1^{(i)}, y_1^{(i)}, c^{(i)}\right) \in \mathbb{R}^4 \times \{1, 2, \dots, C\}$
  - $\left(x_0^{(i)}, y_0^{(i)}\right)$ - top left corner
  - $\left(x_1^{(i)}, y_1^{(i)}\right)$ - bottom right corner
  - $c^{(i)}$ is the class and C is the number of classes ($C = 80$ for MS-COCO)

Receptive Field

- Each point (x, y) in the feature map can be mapped back on the input image at location $(x', y') = \left(\left\lfloor \frac{s}{2} \right\rfloor + xs, \left\lfloor \frac{s}{2} \right\rfloor + ys\right)$ near the center of the receptive field
- There can be many points backprojecting to the same object instance
  - Improves performance compared to IoU-based positive anchors

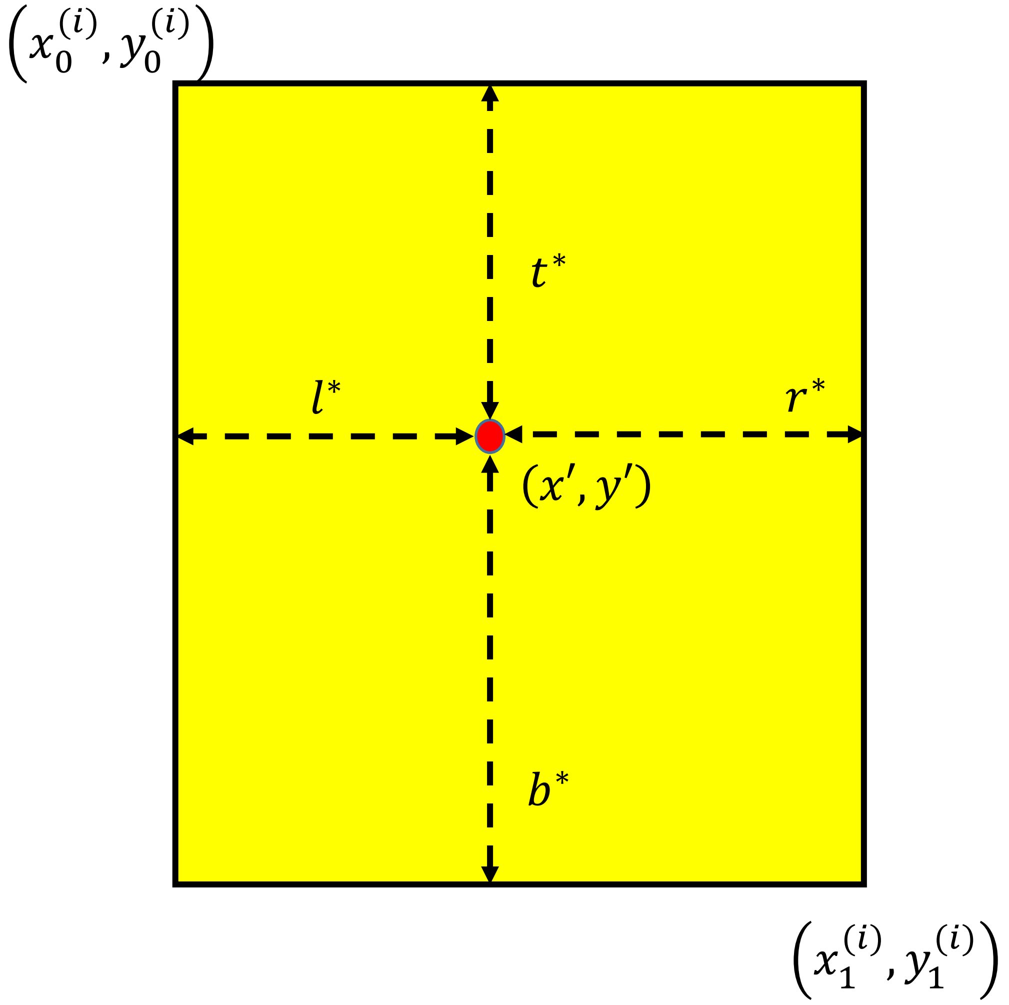

(x, y) is point in feature map
Feature map
Center at (x', y')
Receptive field of the pt in feature map
Input image

Ground Truth

- Each point (x, y) in the feature map is considered positive sample if when back projected it falls inside the bounding box and is labelled with class c^* or the class of the bounding box
    - In addition, a 4D vector $\boldsymbol{t}^* = (l^*, t^*, r^*, b^*)$ bounding box regression target is assigned corresponding to the offset of (x', y') relative to the left, top, right, and bottom sides of the bounding box
- Else, (x, y) is considered negative sample with label $c^* = 0$ (background)
- If (x', y') falls into multiple bounding boxes, it is considered an ambiguous sample
    - The smallest bounding box will be considered the ground truth

Ground Truth

- FCOS target:
  - $l^* = x' - x_0^{(i)}$
  - $t^* = y' - y_0^{(i)}$
  - $r^* = x_1^{(i)} - x'$
  - $b^* = y_1^{(i)} - y'$


$\left(x_0^{(i)}, y_0^{(i)}\right)$
t^*
l^*
r^*
(x', y')
b^*
$\left(x_1^{(i)}, y_1^{(i)}\right)$

FCOS Network

Outputs:
Classifier: 80D $\boldsymbol{p}$; C+1-dim softmax or C sigmoids
Bounding box: 4D bounding box coordinates $\boldsymbol{t} = (l, t, r, b)$
Bounding box outputs are terminated by exp() to map outputs to positive real numbers

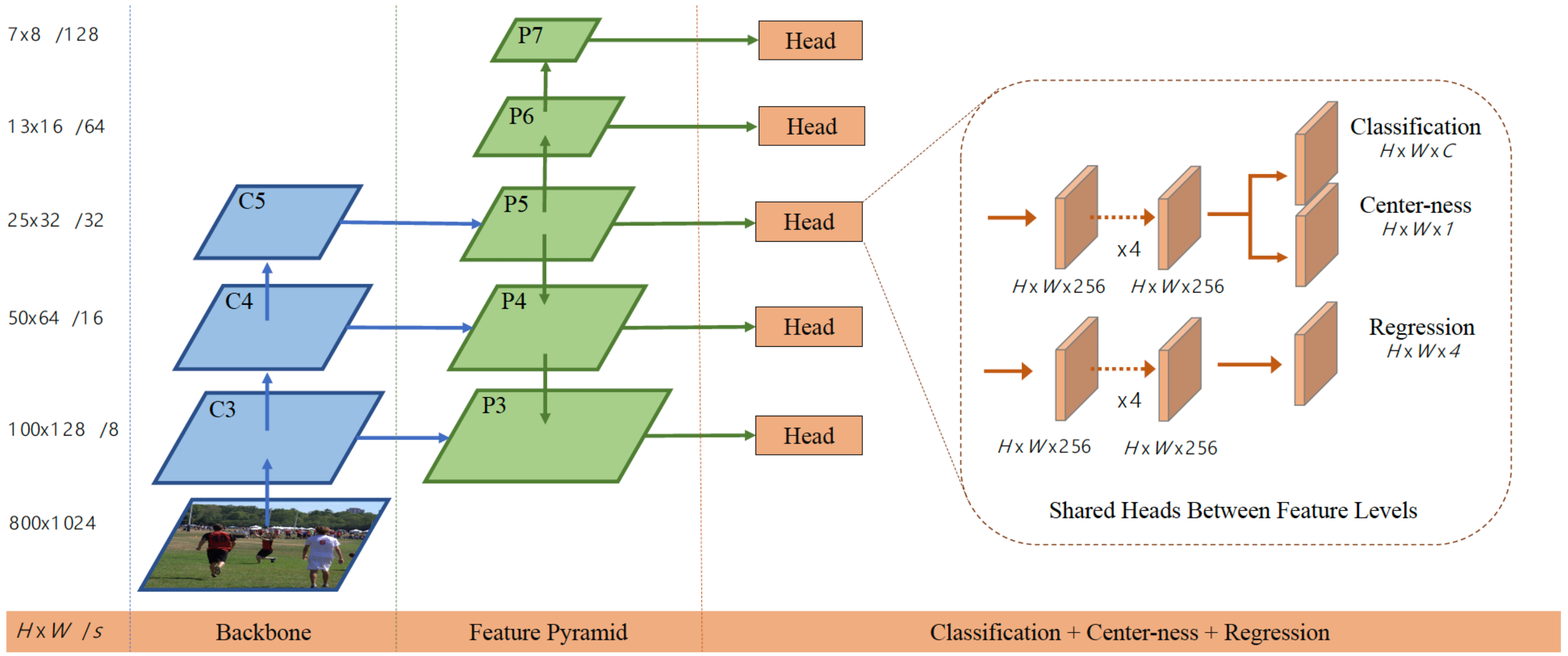


Figure 2 – The network architecture of FCOS, where C3, C4, and C5 denote the feature maps of the backbone network and P3 to P7 are the feature levels used for the final prediction. $H \times W$ is the height and width of feature maps. '/s' ($s = 8, 16, ..., 128$) is the down-sampling ratio of the feature maps at the level to the input image. As an example, all the numbers are computed with an 800×1024 input.

Loss Functions

$$L(\{\boldsymbol{p}_{x,y}\},\{\boldsymbol{t}_{x,y}\}) = \frac{1}{N_{pos}}\sum_{x,y} L_{cls}(\boldsymbol{p}_{x,y}, c^*_{x,y}) + \frac{\lambda}{N_{pos}}\sum_{x,y}[c^*_{x,y} \geq 1] L_{reg}(\boldsymbol{t}_{x,y}, \boldsymbol{t}^*_{x,y})$$

Where:

L_{cls} is the Focal Loss

L_{reg} is the IoU Loss

λ is the weighting factor (equal to 1)

The sum is done for all points in each Feature map F_i

Resolving Ambiguous Samples

- Consider pyramid feature levels $P_3, P_4, P_5, P_6\ and\ P_7$ with strides 8, 16, 32, 64, 128
- Given $\boldsymbol{t}^* = (l^*, t^*, r^*, b^*)$ for each point in the feature map at each pyramid level feature map
- If a point satisfies $\max(\boldsymbol{t}^*) > m_i$ or $\min(\boldsymbol{t}^*) < m_{i-1}$, it is tagged as negative sample
    - $(m_2, m_3, m_4, m_5, m_6, m_7) = (0, 64, 128, 256, 512, \infty)$

Centerness

- Many low-quality bounding box due to predictions that are far away from the center
- Centerness measure is introduced:

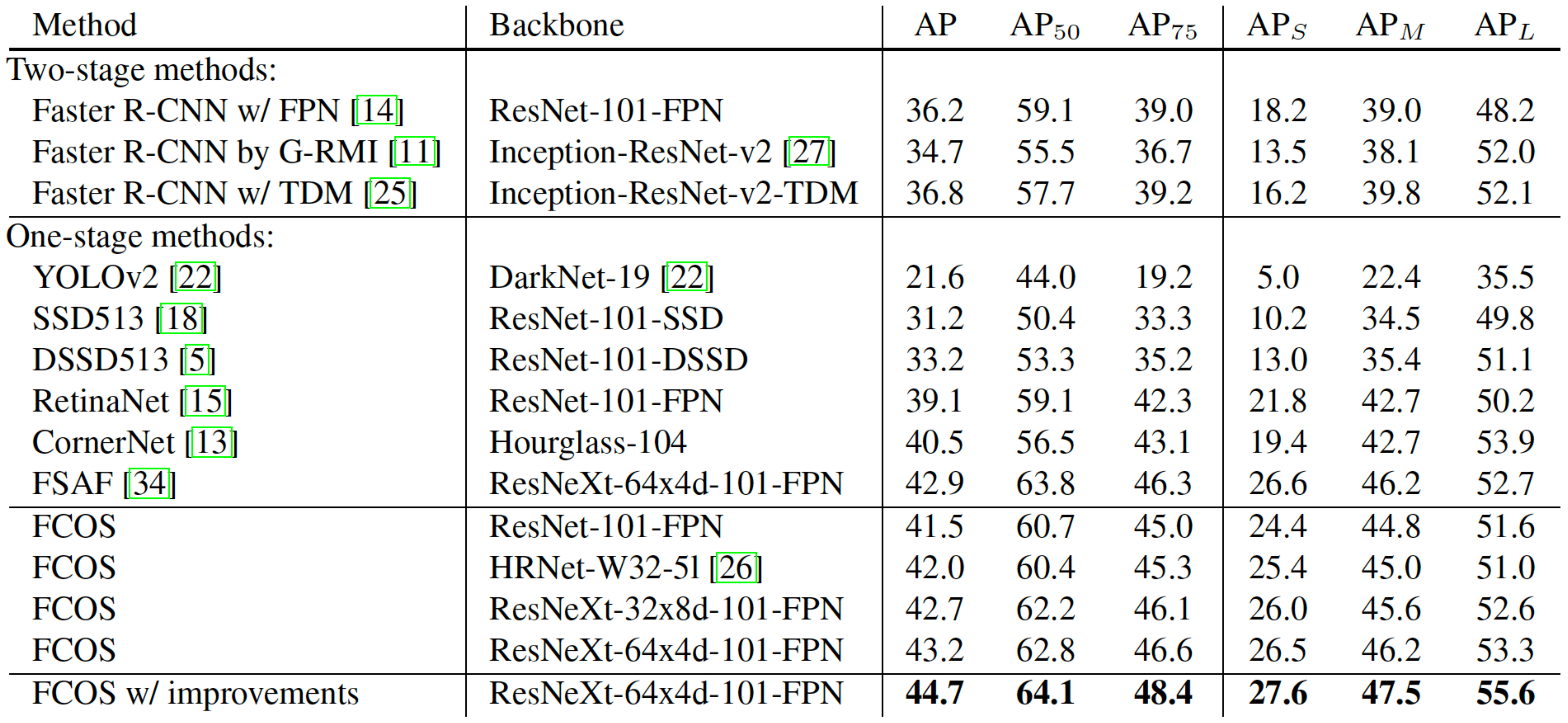
$$centerness = \sqrt{\frac{\min(l^*, r^*)}{\max(l^*, r^*)} \times \frac{\min(t^*, b^*)}{\max(t^*, b^*)}}$$

$$centerness \in [0,1]$$

- Additional layer: sigmoid output
- Additional loss function: binary cross entropy
- Inference: The classifier probability output is multiplied by centerness to down-weight the confidence score of low quality bounding boxes. Low-confidence bounding boxes will be filtered out by NMS.

Performance

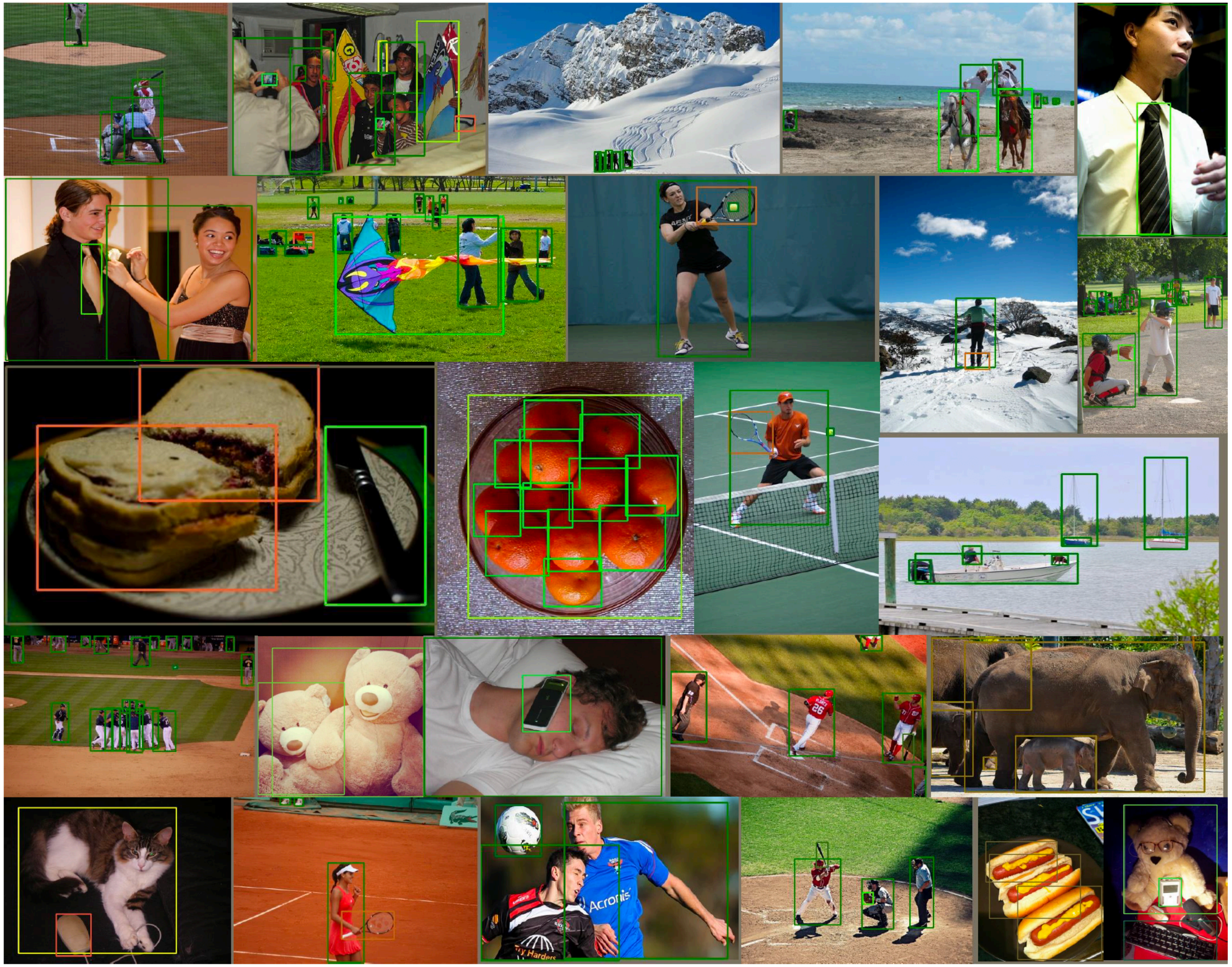
- Coco test-dev : 44.7 AP



Method	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods:							
Faster R-CNN w/ FPN [14]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [11]	Inception-ResNet-v2 [27]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w/ TDM [25]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
One-stage methods:							
YOLOv2 [22]	DarkNet-19 [22]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [18]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [5]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [15]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
CornerNet [13]	Hourglass-104	40.5	56.5	43.1	19.4	42.7	53.9
FSAF [34]	ResNeXt-64x4d-101-FPN	42.9	63.8	46.3	26.6	46.2	52.7
FCOS	ResNet-101-FPN	41.5	60.7	45.0	24.4	44.8	51.6
FCOS	HRNet-W32-51 [26]	42.0	60.4	45.3	25.4	45.0	51.0
FCOS	ResNeXt-32x8d-101-FPN	42.7	62.2	46.1	26.0	45.6	52.6
FCOS	ResNeXt-64x4d-101-FPN	43.2	62.8	46.6	26.5	46.2	53.3
FCOS w/ improvements	ResNeXt-64x4d-101-FPN	**44.7**	**64.1**	**48.4**	**27.6**	**47.5**	**55.6**

Table 5 – FCOS vs. other state-of-the-art two-stage or one-stage detectors (*single-model and single-scale results*). FCOS outperforms the anchor-based counterpart RetinaNet by 2.4% in AP with the same backbone. FCOS also outperforms the recent anchor-free one-stage detector CornerNet with much less design complexity. Refer to Table 3 for details of "improvements".

FCOS Sample Outputs



End