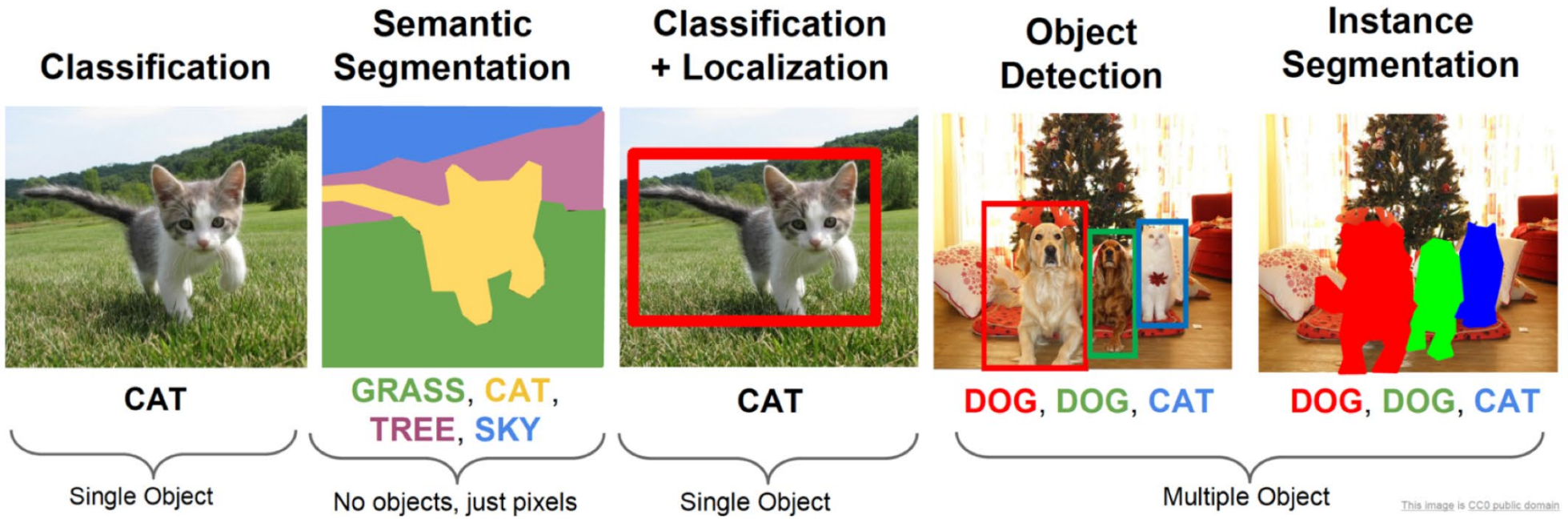
Deep Learning Optimization - Object Detection

March 15, 2023

Eunhyeok Park

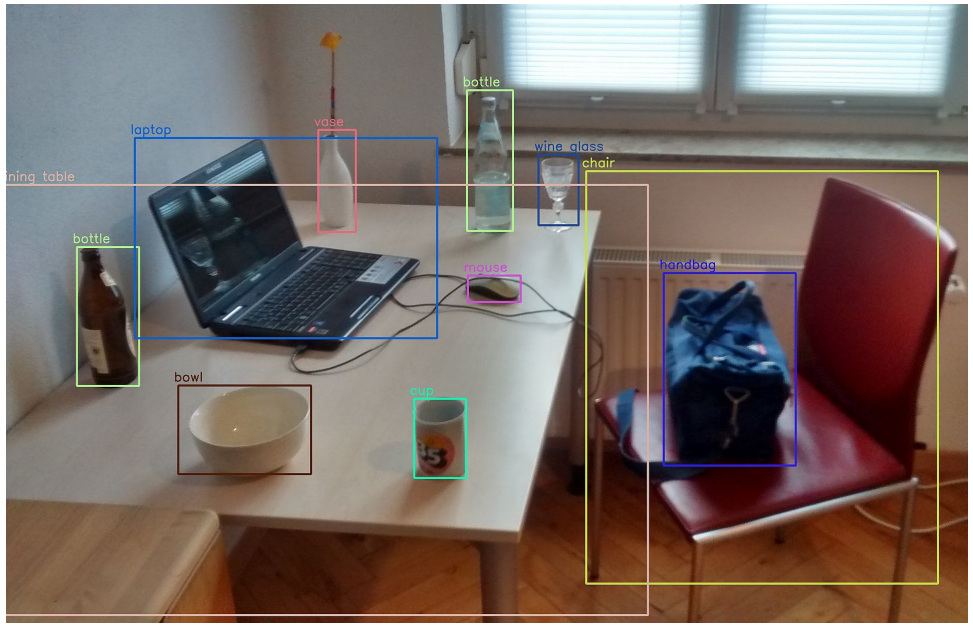
Various Vision Tasks

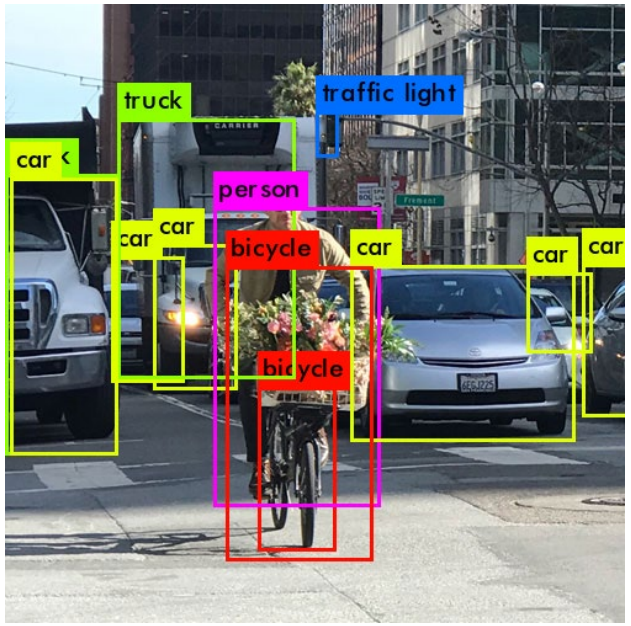
- There are enormous vision tasks having different characteristics
  - # of instances per output
  - Desired output format, i.e. classification / regression
  - Local information / global information



Object Detection

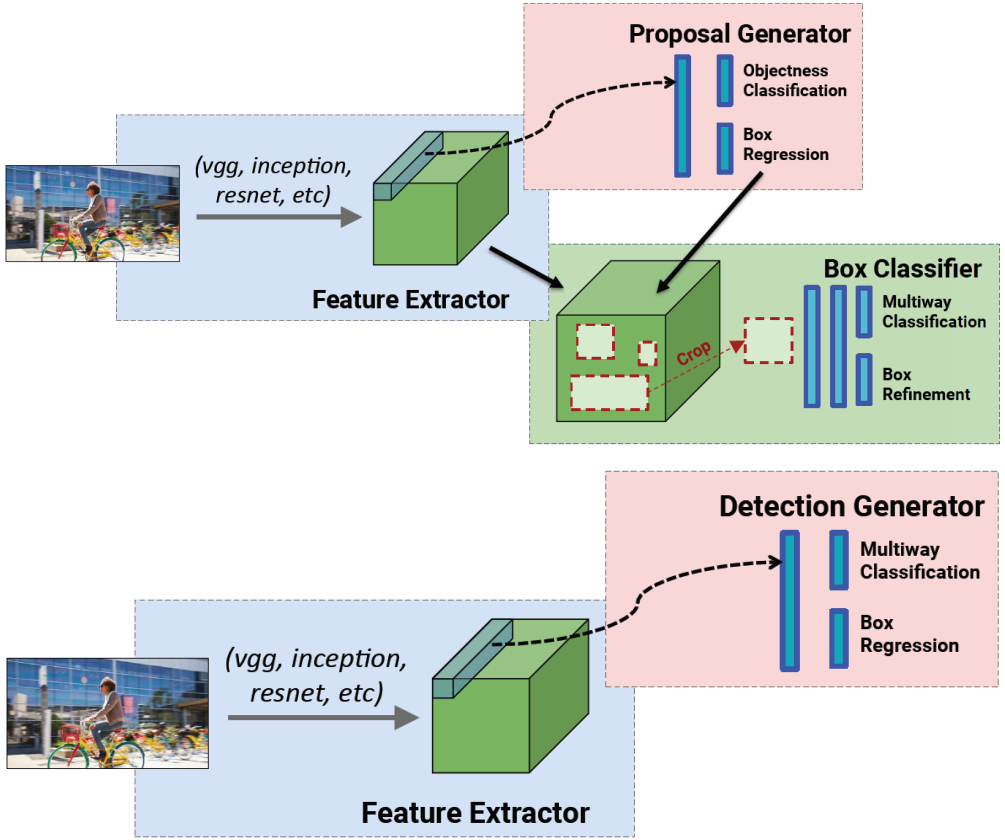
- Object detection is an image processing task that detecting objects in certain class
  - Classification (object class) + regression (object position)
  - Multi-instance & multi-object
  - In general, find out the bounding box with the smallest area while covering the entire object





Object Detection Methods

- Two-step methods
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
  - Mask R-CNN
  - Light-Head R-CNN
  - Pyramid Networks / FPN
  - Oriented R-CNN
  - G-RCNN
- Single shot methods
  - YOLO
  - SSD
  - YOLO2
  - DSSD
  - RetinaNet
  - YOLOv3-v8
  - M2Det
  - Sparse R-CNN
- Anchor-free methods
  - CornerNet
  - CenterNet
  - FCOS
  - CenterNet++



[Huang, 2017]

Selective Search

- Use bottom-up grouping of image regions to generate a hierarchy of small to large regions.
  1. Generate initial sub-segmentation
  2. Recursively combine similar regions into larger ones
  3. Use the generated regions to produce candidate object locations

[S. Smith]

Similarity Measure

- Final metric:
  - Considering various aspects of visual similarity
  - Linear combination of the four metrics:
  - $s(r_i, r_j) =$

$$a_1 s_{colour}(r_i, r_j) + a_2 s_{texture}(r_i, r_j) + a_3 s_{size}(r_i, r_j) + a_4 s_{fill}(r_i, r_j)$$

[S. Smith]

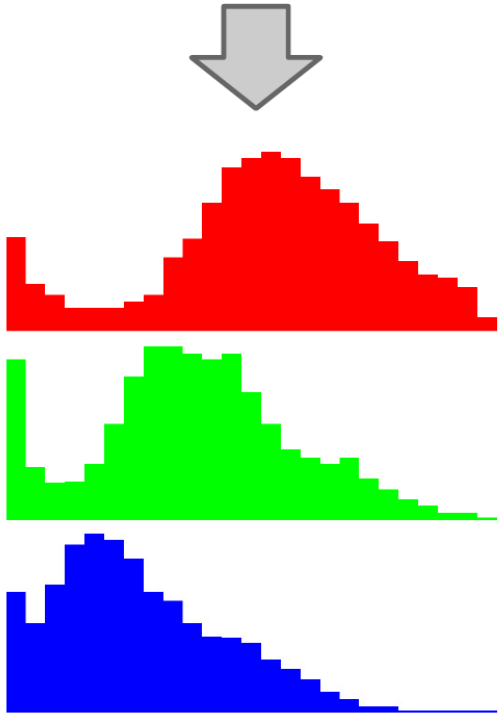
Color Similarity

Create a color histogram *C* for each channel in region *r*.

In the paper, 25 bins were used, for 75 total dimensions.

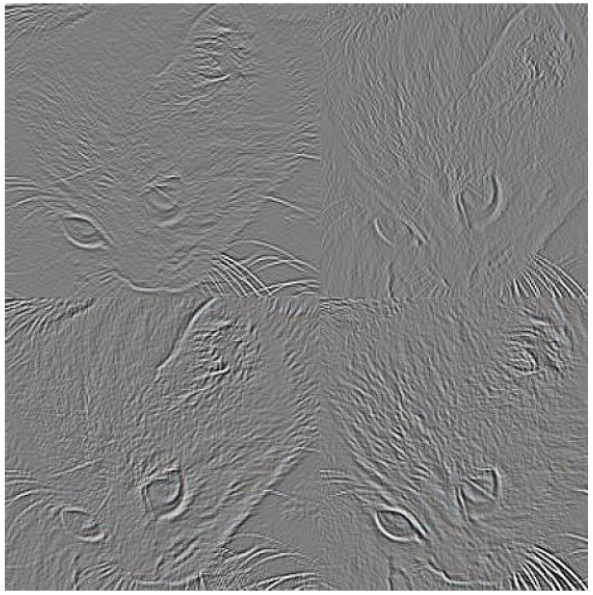
We can measure similarity with histogram intersection:

$$s_{colour}(r_i, r_j) = \sum_{k=1}^{n} \min(c_i^k, c_j^k)$$



[S. Smith]

Texture Similarity

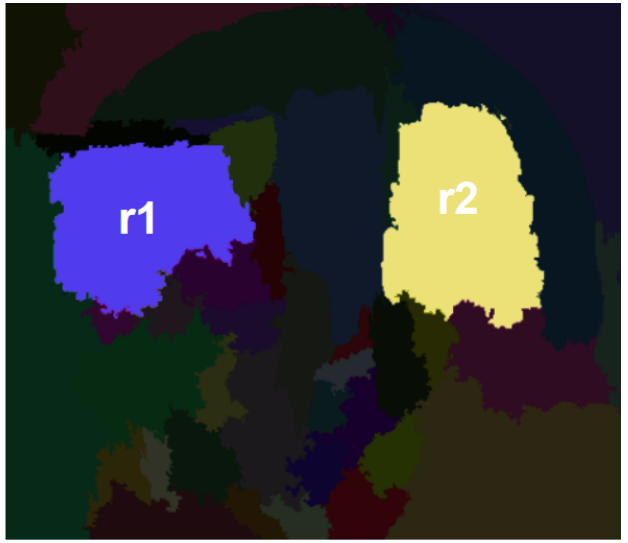


Can measure textures with a HOG-like feature:

1. Extract gaussian derivatives of the image in 8 directions and for each channel.
2. Construct a 10-bin histogram for each, resulting in a 240-dimensional descriptor.

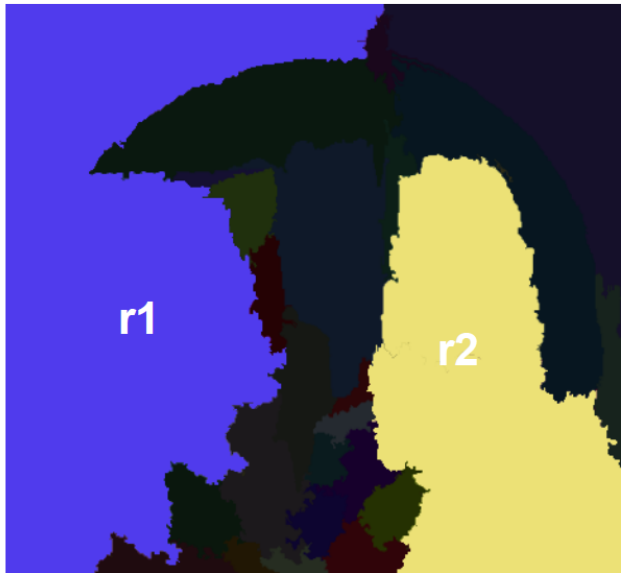
[S. Smith]

Size Similarity



We want small regions to merge into larger ones, to create a balanced hierarchy.

Solution: Add a size component to our similarity metric, that ensures small regions are more similar to each other.



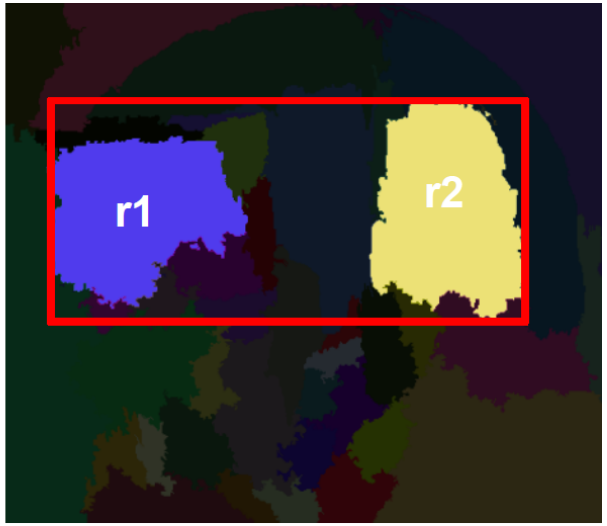
$$s_{size}(r_i, r_j) = 1 - \frac{\text{size}(r_i) + \text{size}(\text{r}_\text{j})}{\text{size}(im)}$$

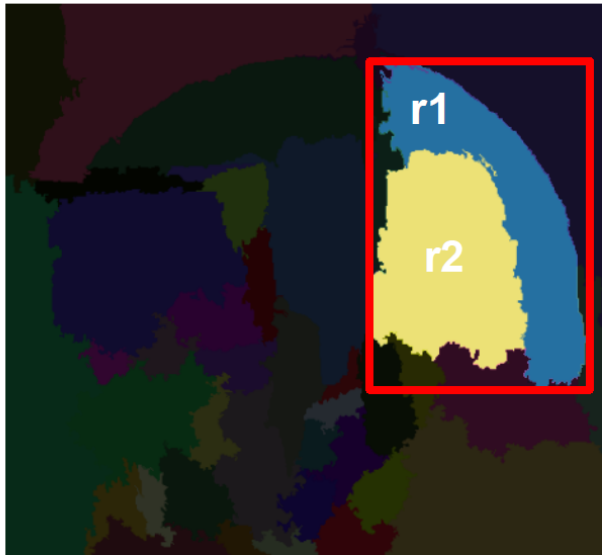
[S. Smith]

Shape Compatibility

We also want our merged regions to be cohesive, so we can add a measure of how well two regions "fit together".

$$fill(r_i, r_j) = 1 - \frac{\text{size}(BB_{ij}) - \text{size}(r_i) - \text{size}(r_i)}{\text{size}(im)}$$





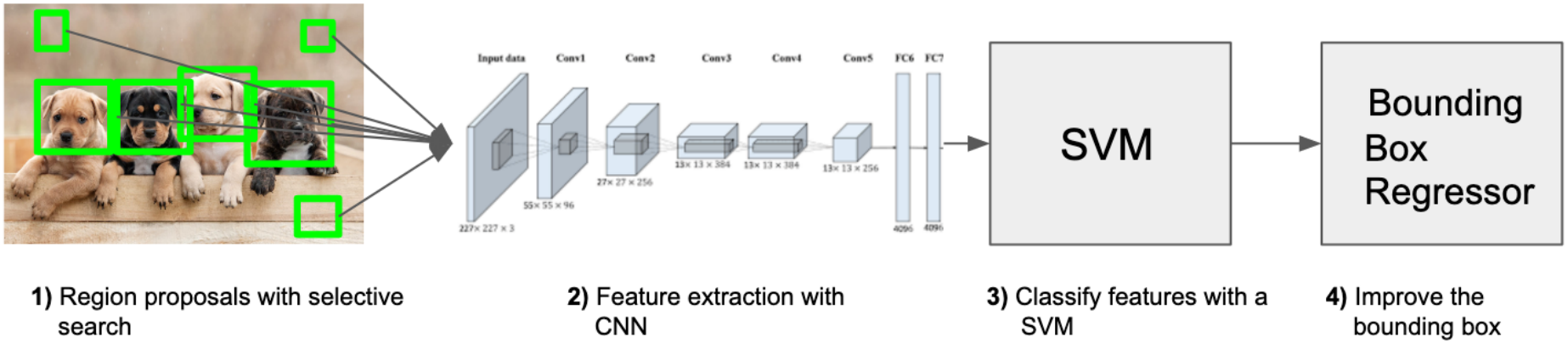
[S. Smith]

Selective Search

- Use bottom-up grouping of image regions to generate a hierarchy of small to large regions.
  1. Generate initial sub-segmentation
  2. Recursively combine similar regions into larger ones
  3. Use the generated regions to produce candidate object locations

[S. Smith]

Object Detection in R-CNN



1) Region proposals with selective search

2) Feature extraction with CNN

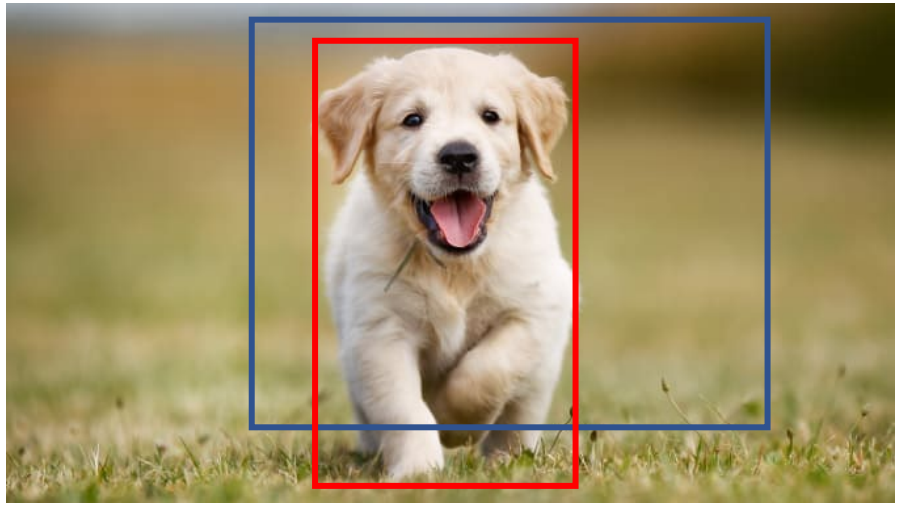
3) Classify features with a SVM

4) Improve the bounding box

- CNN extracts features from the inputs corresponding to the box proposed by selective search (2000 proposals)
- Features are fed in to an SVM to classify the presence of object
  - If bounding box regression is used, MLP is attached to predict the box coordinate

Box Regression in R-CNN

- Box Regression offers additional 3 % accuracy improvement
- Proposal generated by selective search $P^i = (P_x^i, P_y^i, P_w^i, P_H^i)$
- Ground truth of an object
- $G = (G_x, G_y, G_w, G_h)$
- Center point + width & height



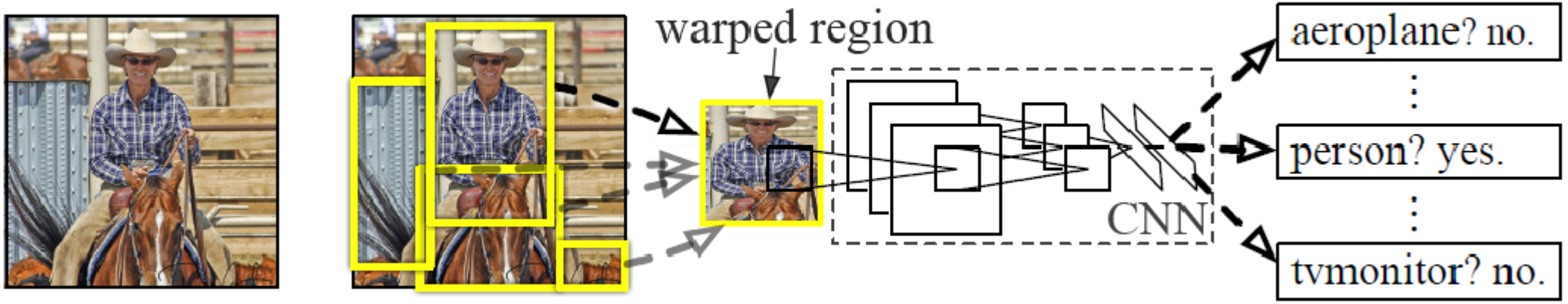
Regression Target

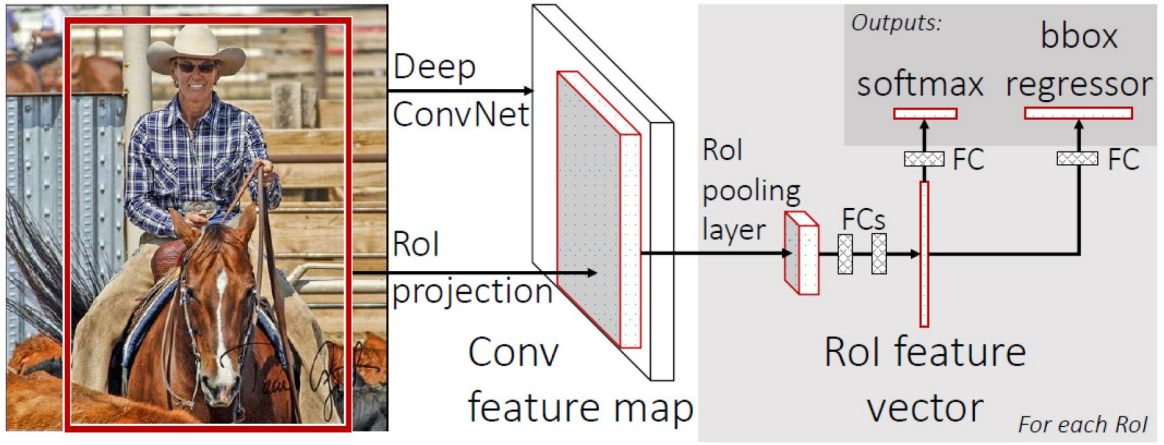
$$t_x = \frac{G_x - P_x}{P_w}, \qquad t_y = \frac{G_y - P_y}{P_h}$$

$$t_w = \log\left(\frac{G_w}{P_w}\right), \qquad t_h = \log\left(\frac{G_h}{P_h}\right)$$

R-CNN vs Fast R-CNN

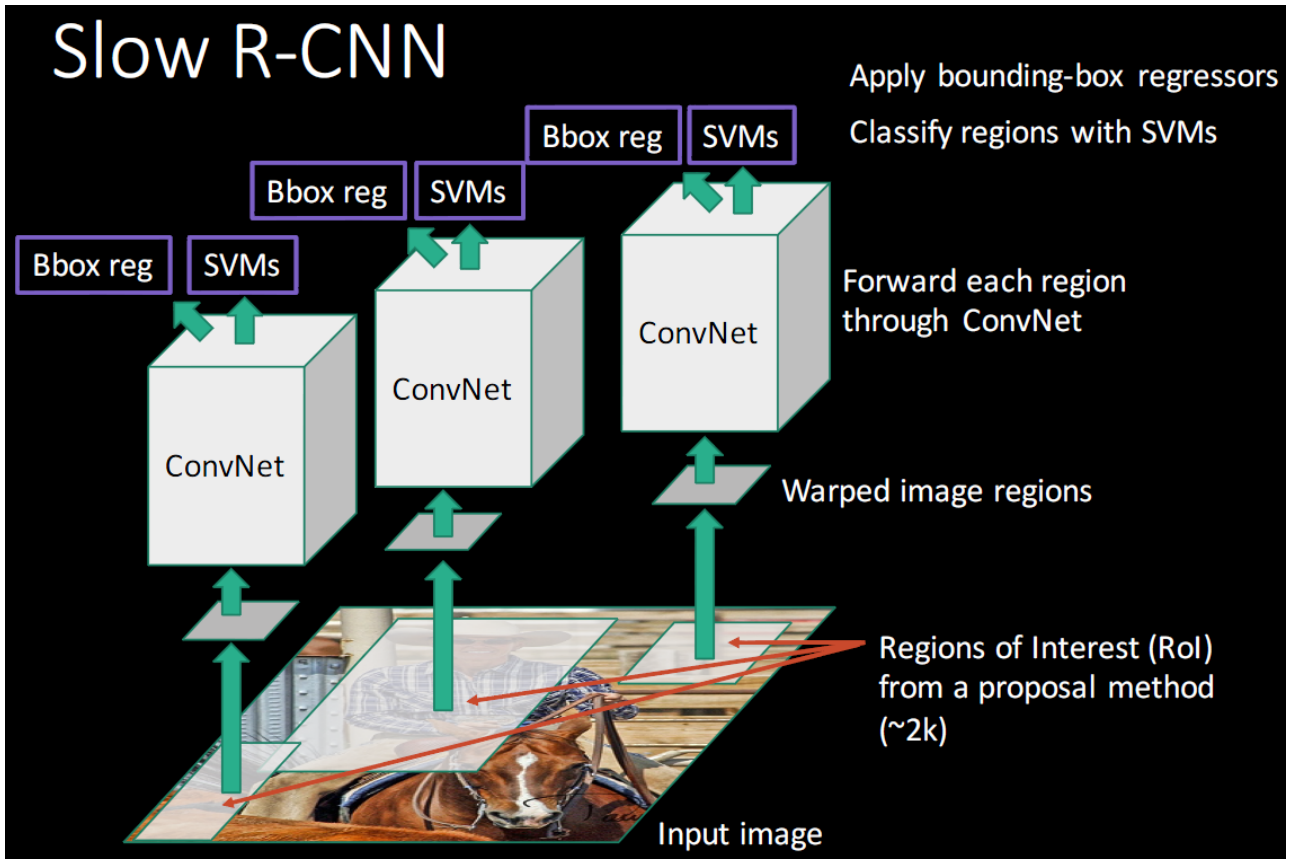
- Bottleneck of R-CNN: multiple CNN runs for 2k proposed regions
  - Fast R-CNN: Adopt RoI pooling and reuse the output feature map several times

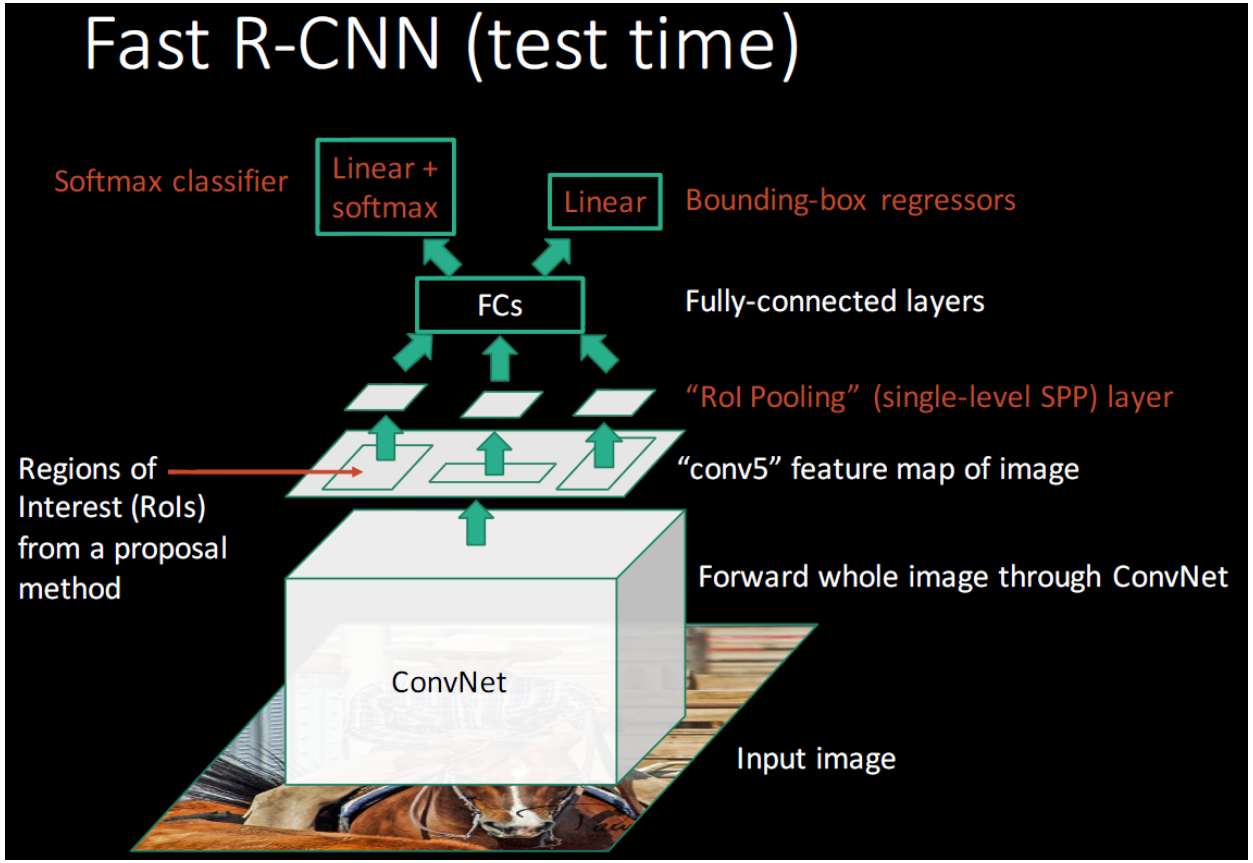




R-CNN vs Fast R-CNN

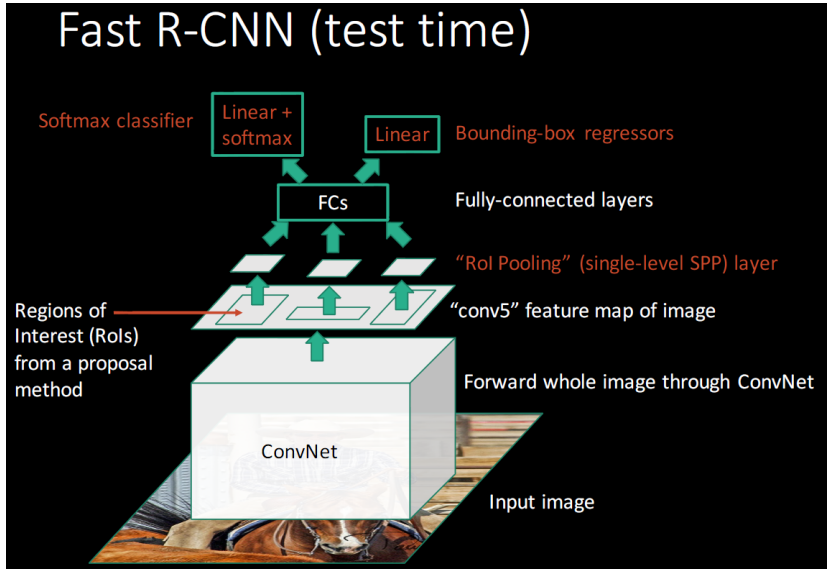
- Bottleneck of R-CNN: multiple CNN runs for 2k proposed regions
  - Adopt RoI pooling and reuse the output feature map several times

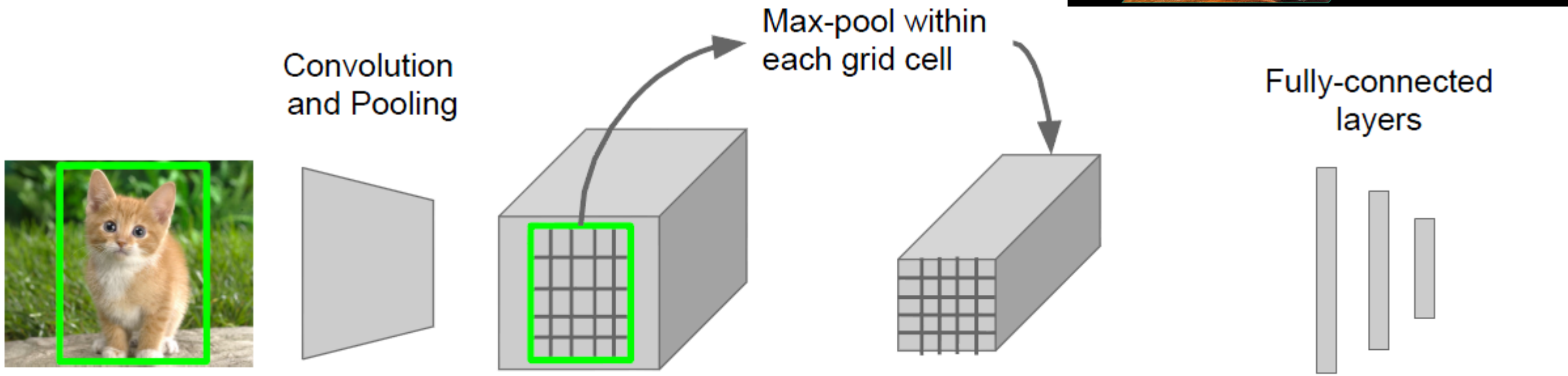




RoI Pooling

- Crop & max-pool for RoI-covered features
- Apply linear layers for the low-resolution features





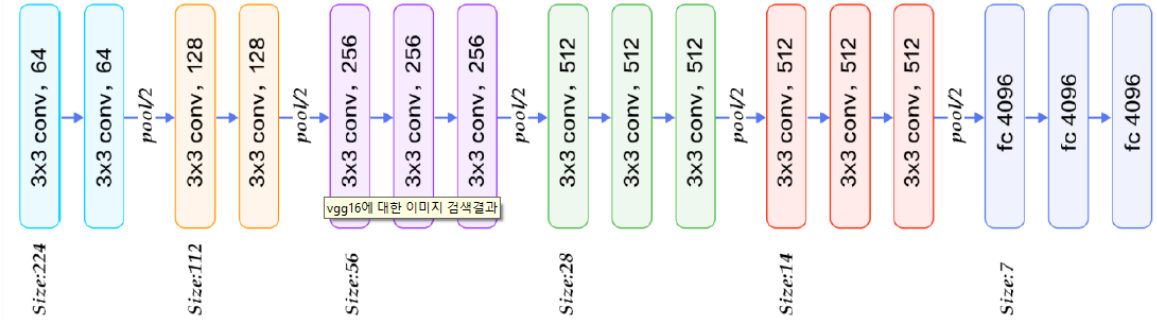
Hi-res input image: 3 x 800 x 600 with region proposal

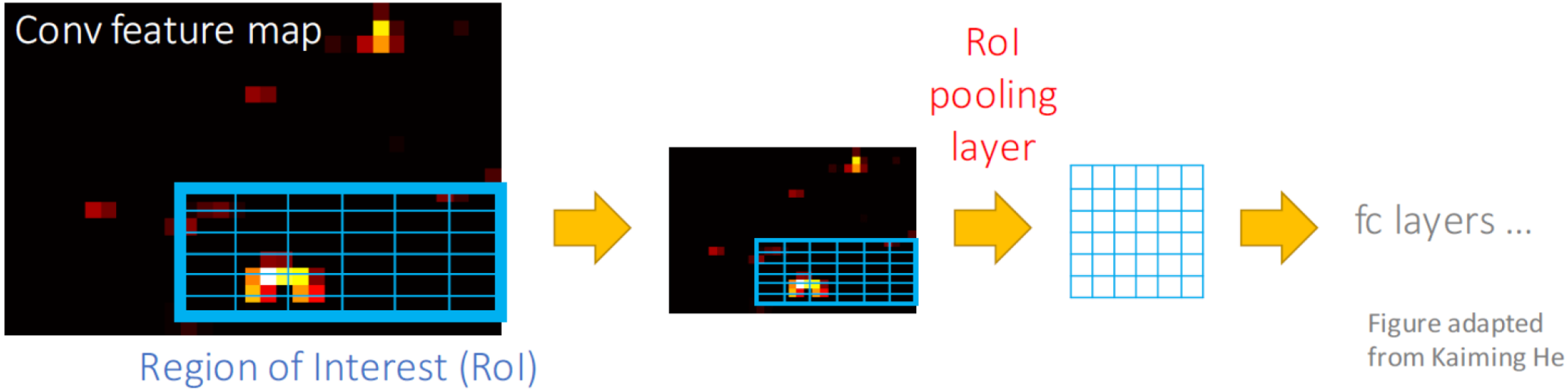
Hi-res conv features: C x H x W with region proposal

RoI conv features: C x h x w for region proposal

Fully-connected layers expect low-res conv features: C x h x w

ROI Pooling



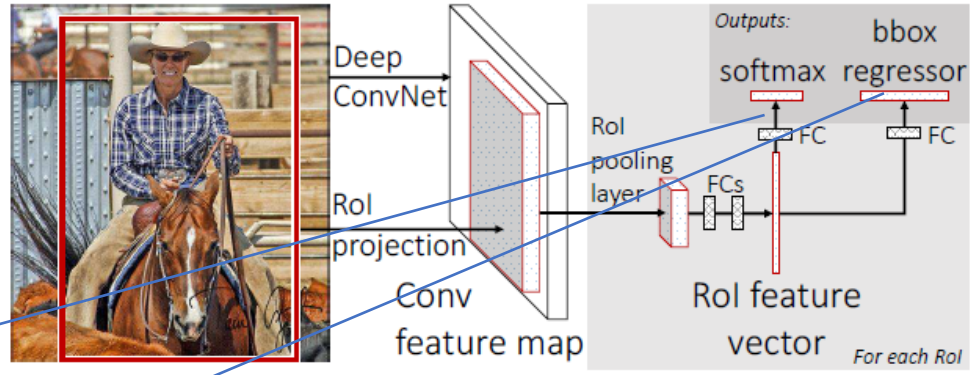


Just a special case of the SPP layer with one pyramid level

RoI in Conv feature map : 21x14 → 3x2 max pooling with stride(3, 2) → output : 7x7
RoI in Conv feature map : 35x42 → 5x6 max pooling with stride(5, 6) → output : 7x7

Training: Multi-task Loss Function

1. Takes an input and a set of bounding boxes
2. Extract features using convolution backbone
3. For each bbox, get a fixed-length feature using ROI pooling layer
4. Apply loss function with two information
   1. Class labels
   2. Bounding box locations



$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L_1}(t_i^u - v_i),$$

True box coordinates

Predicted box coordinates

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1] L_{\text{loc}}(t^u, v)$$

Predicted class scores

True class scores

Log loss

Smooth L1 loss

in which

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

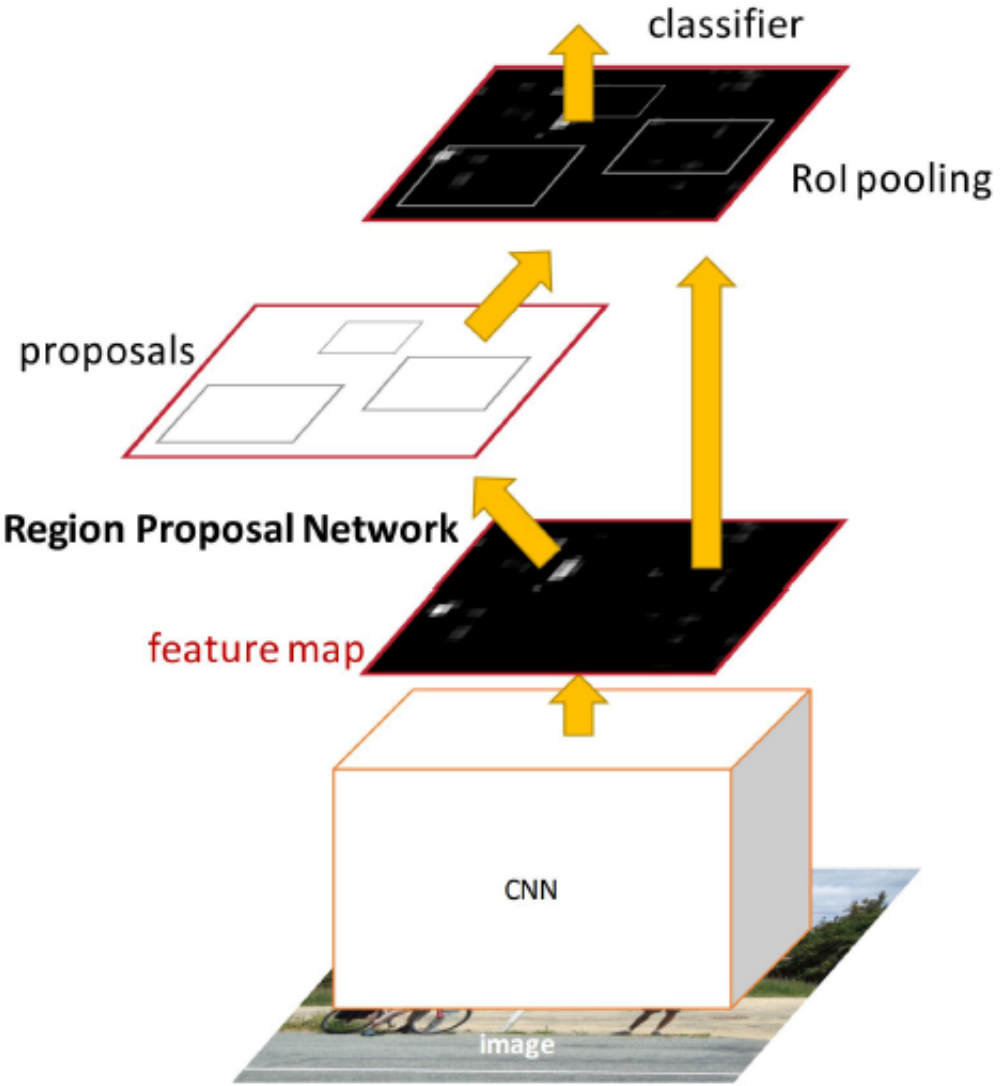
We Need to Go Deeper…

- Remained bottleneck: region proposal
- How to obtain good region proposals faster?

	R-CNN	**Fast R-CNN**
Test time per image	47 seconds	**0.32 seconds**
(Speedup)	1x	**146x**
Test time per image with Selective Search	50 seconds	**2 seconds**
(Speedup)	1x	**25x**

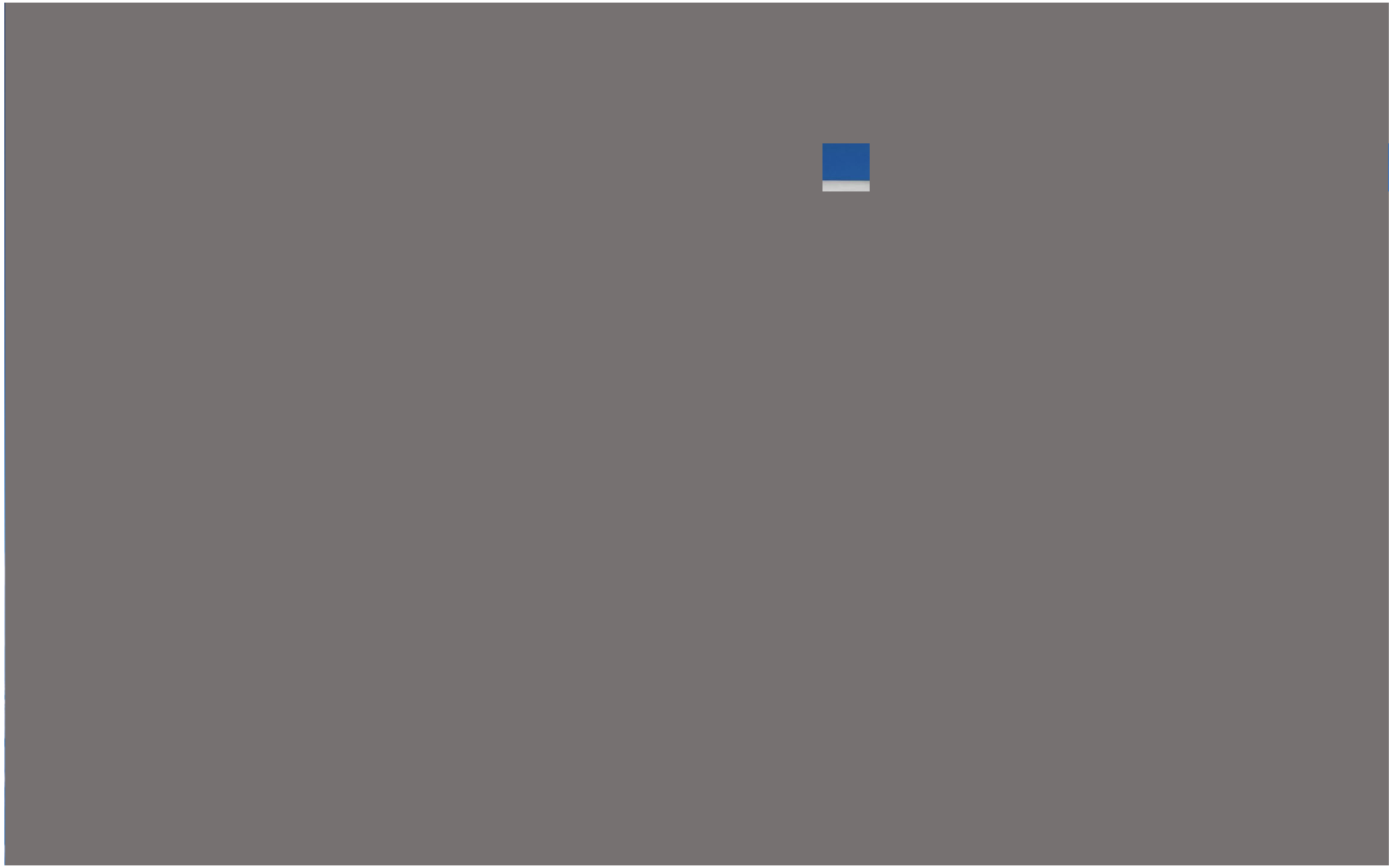
Faster R-CNN

- Replace selective search by region proposal network (RPN)
- Selective search is not used
- Region proposal network (RPN)
  - Runs on conv feature maps
  - Anchor-based proposal (multi-scale support with multi-scale anchor)
  - Gives region proposals
- ROI pooling
  - Max pooling → trainable



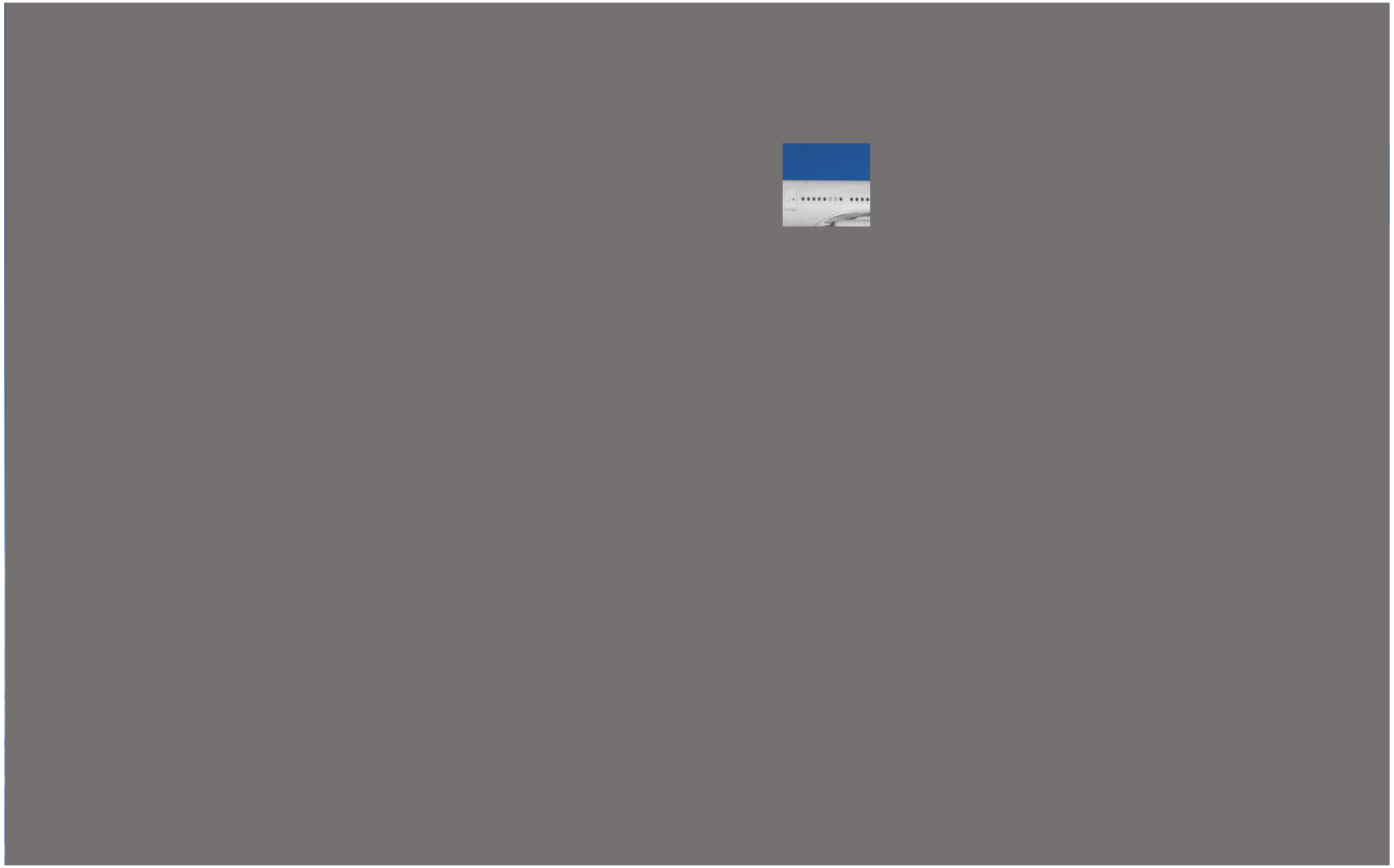
[Girshick, 2015]

Flow Example: What's This?



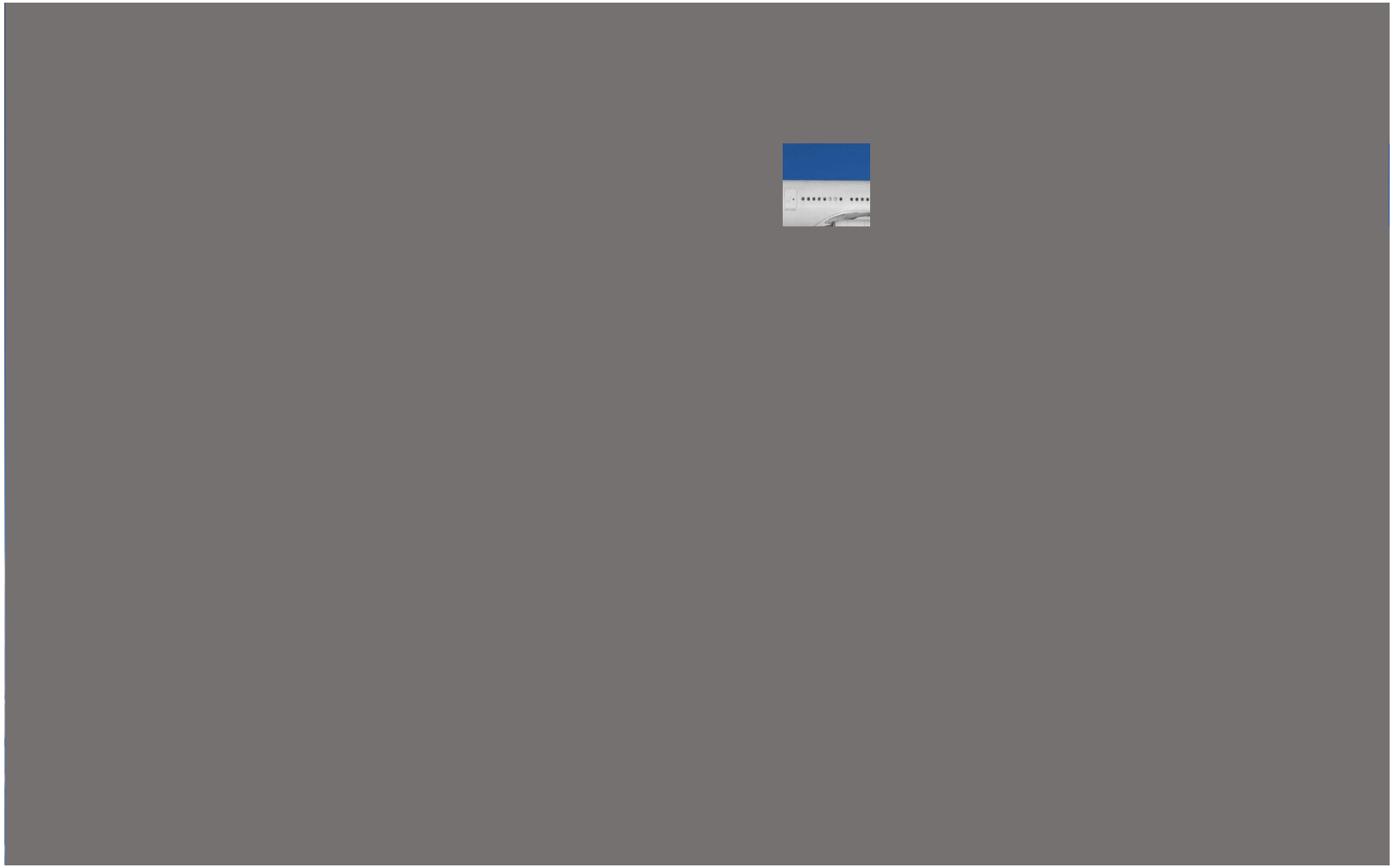
[S.Yoo]

Flow Example: Maybe We Need a Bigger Peephole. What's This?



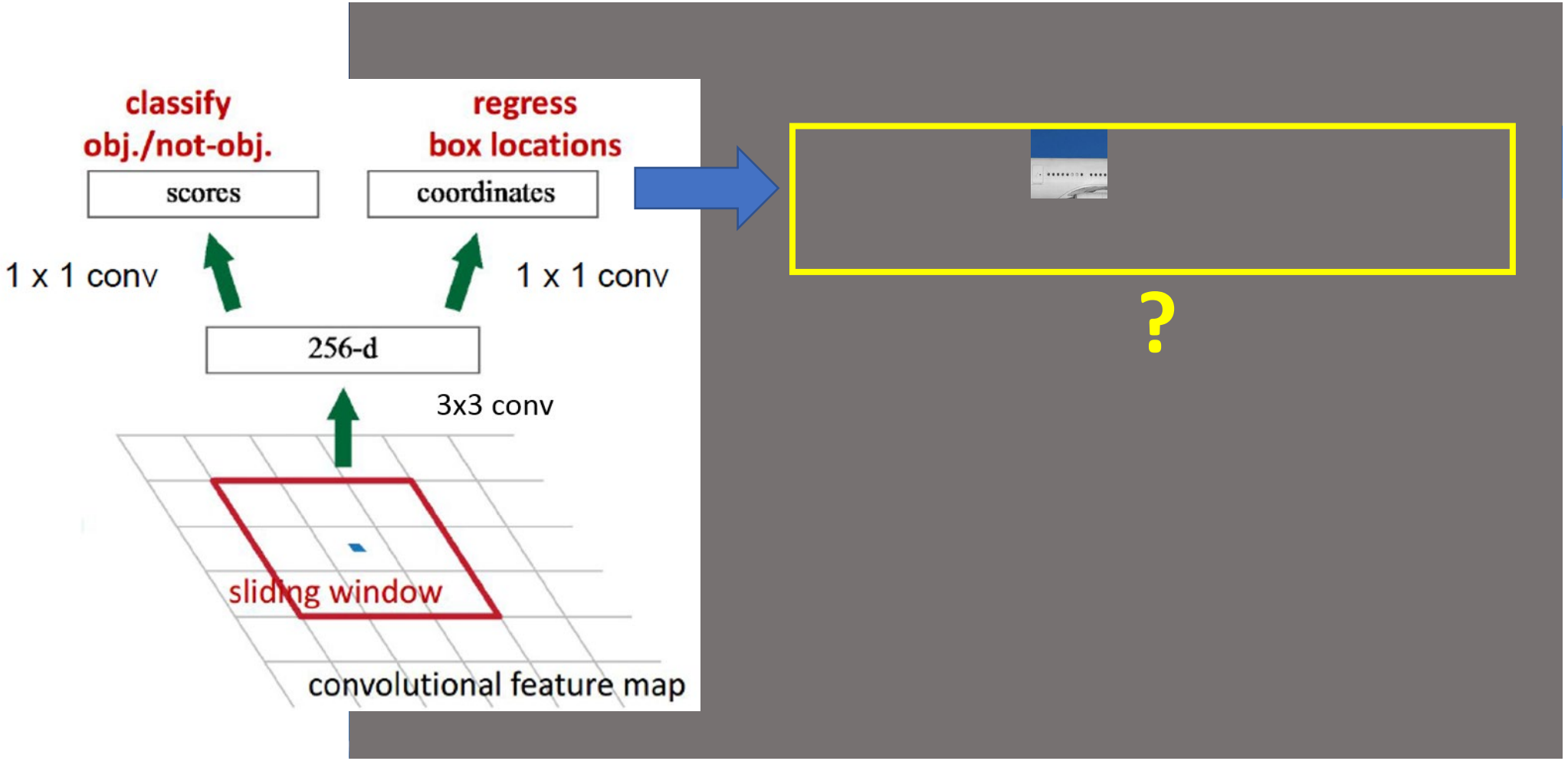
[S.Yoo]

Flow Example: Plane?! How Big Can It Be?



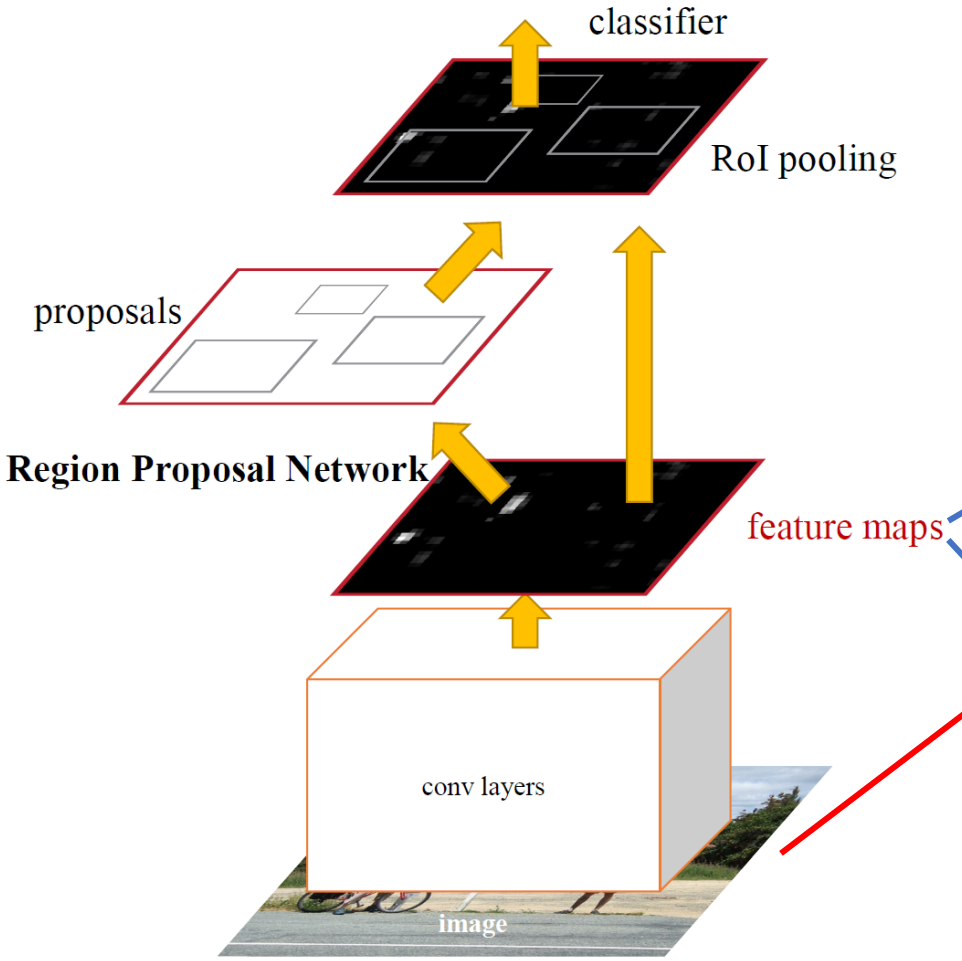
[S.Yoo]

Flow Example: Plane?! How Big Can It Be?



[S.Yoo]

Convolutional Layers in Faster RCNN



ZF

	input channel	input width	input height	kernel size	stride	padding	group	output channel	output width	output height
conv	3	1000	600	7	2	1	1	96	498	298
pool	96	498	298	3	2	0	1	96	249	149
conv	96	249	149	5	2	0	1	256	123	73
pool	256	123	73	3	2	0	1	256	61	36
conv	256	61	36	3	1	1	1	384	61	36
conv	384	61	36	3	1	1	1	384	61	36
conv	384	61	36	3	1	1	1	256	61	36
pool	256	61	36	3	2	0	1	256	30	18

VGG16

	input channel	input width	input height	kernel size	stride	padding	group	output channel	output width	output height
conv	3	1000	600	3	1	1	1	64	1000	600
conv	64	1000	600	3	1	1	1	64	1000	600
pool	64	1000	600	2	2	0	1	64	500	300
conv	64	500	300	3	1	1	1	128	500	300
conv	128	500	300	3	1	1	1	128	500	300
pool	128	500	300	2	2	0	1	128	250	150
conv	128	250	150	3	1	1	1	256	250	150
conv	256	250	150	3	1	1	1	256	250	150
conv	256	250	150	3	1	1	1	256	250	150
pool	256	250	150	2	2	0	1	256	125	75
conv	256	125	75	3	1	1	1	512	125	75
conv	512	125	75	3	1	1	1	512	125	75
conv	512	125	75	3	1	1	1	512	125	75
pool	512	125	75	2	2	0	1	512	62	37
conv	512	62	37	3	1	1	1	512	62	37
conv	512	62	37	3	1	1	1	512	62	37
conv	512	62	37	3	1	1	1	512	62	37
pool	512	62	37	2	2	0	1	512	31	18

[S.Yoo]

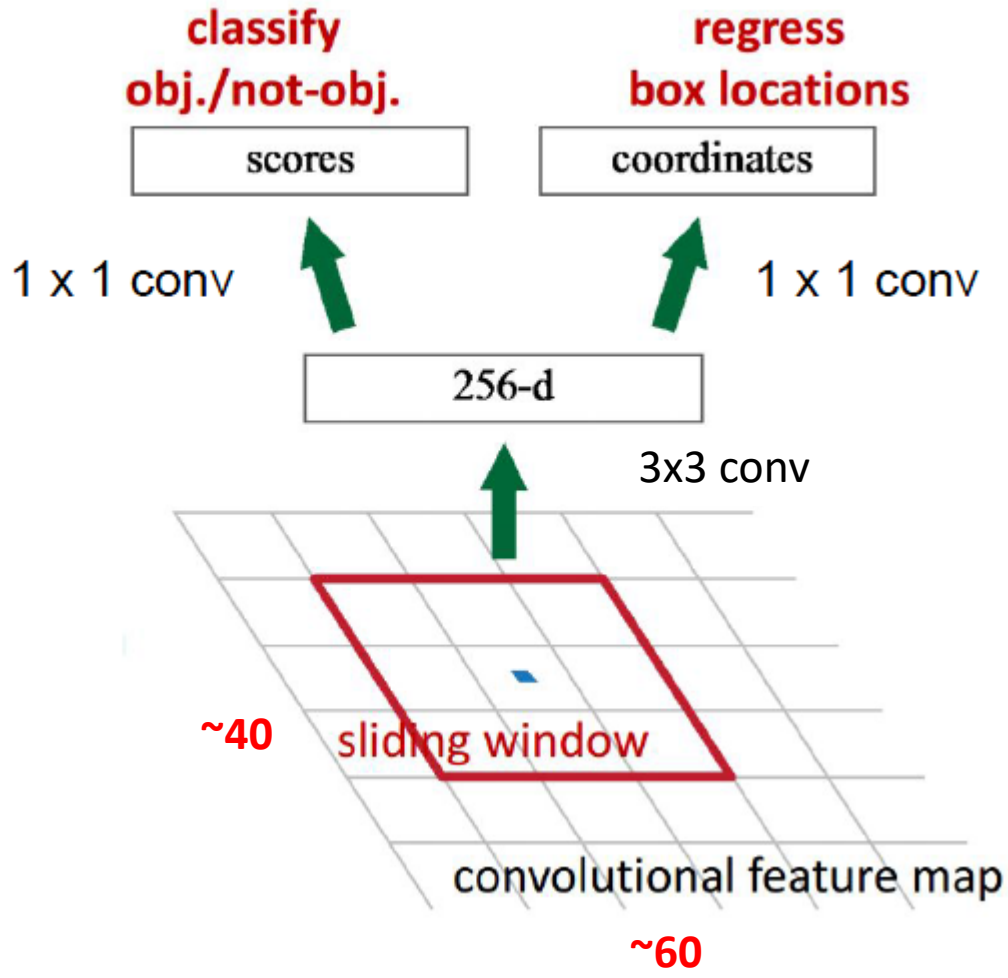
Region Proposal Network (RPN)

Slide a small window on the feature map
3x3?

Build a small network for:
- classifying object or not-object, and
- regressing bbox locations

Position of the sliding window provides localization information with reference to the image

Box regression provides finer localization information with reference to this sliding window



[Girshick, 2015]

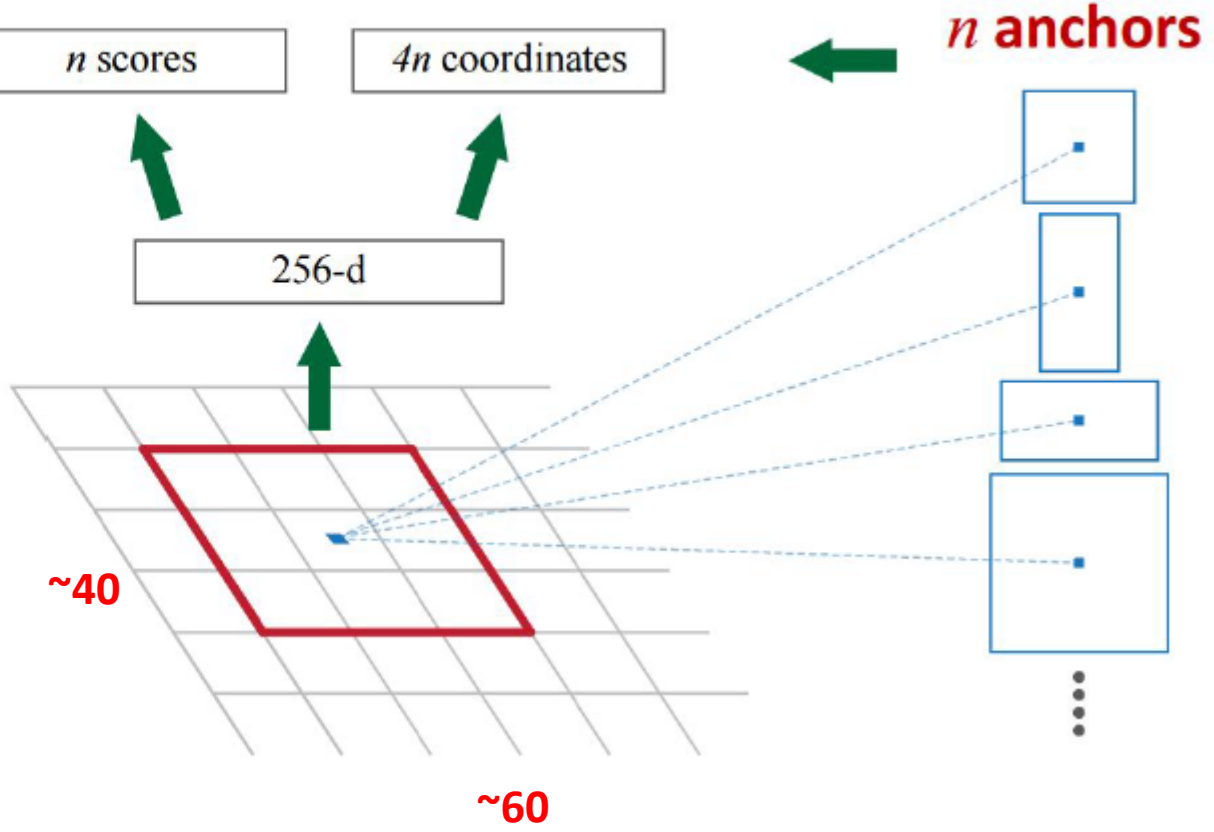
Anchors at Multiple Scales

Use **N anchor boxes** at each location

Anchors are **translation invariant**: use the same ones at every location

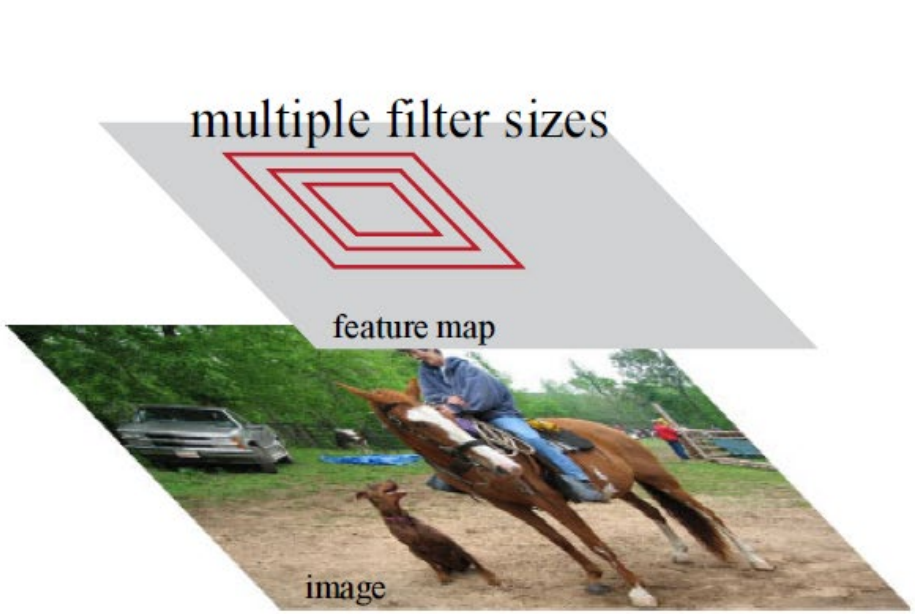
Regression gives offsets from anchor boxes

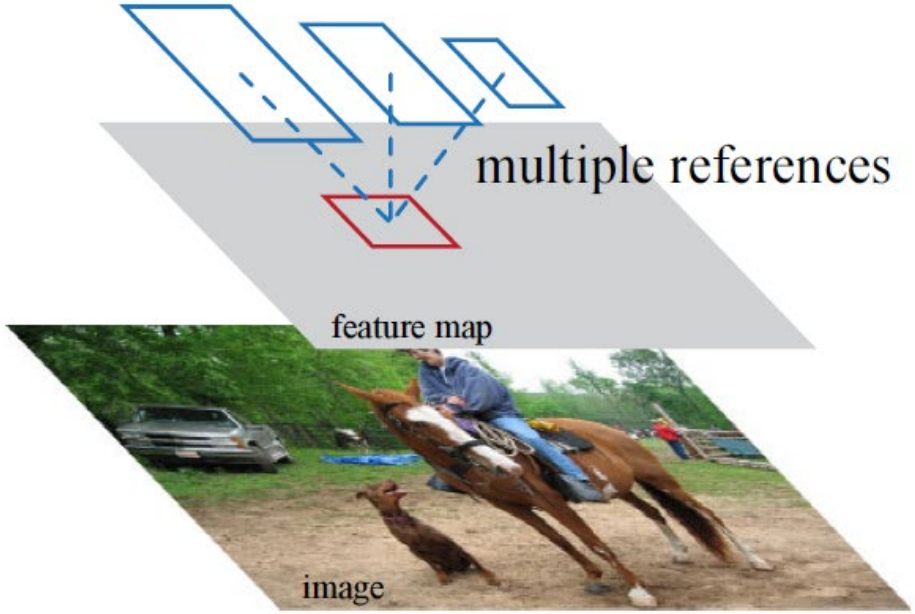
Classification gives the probability that each (regressed) anchor shows an object



[Girshick, 2015]

Multiple Filters or References (Anchors)?



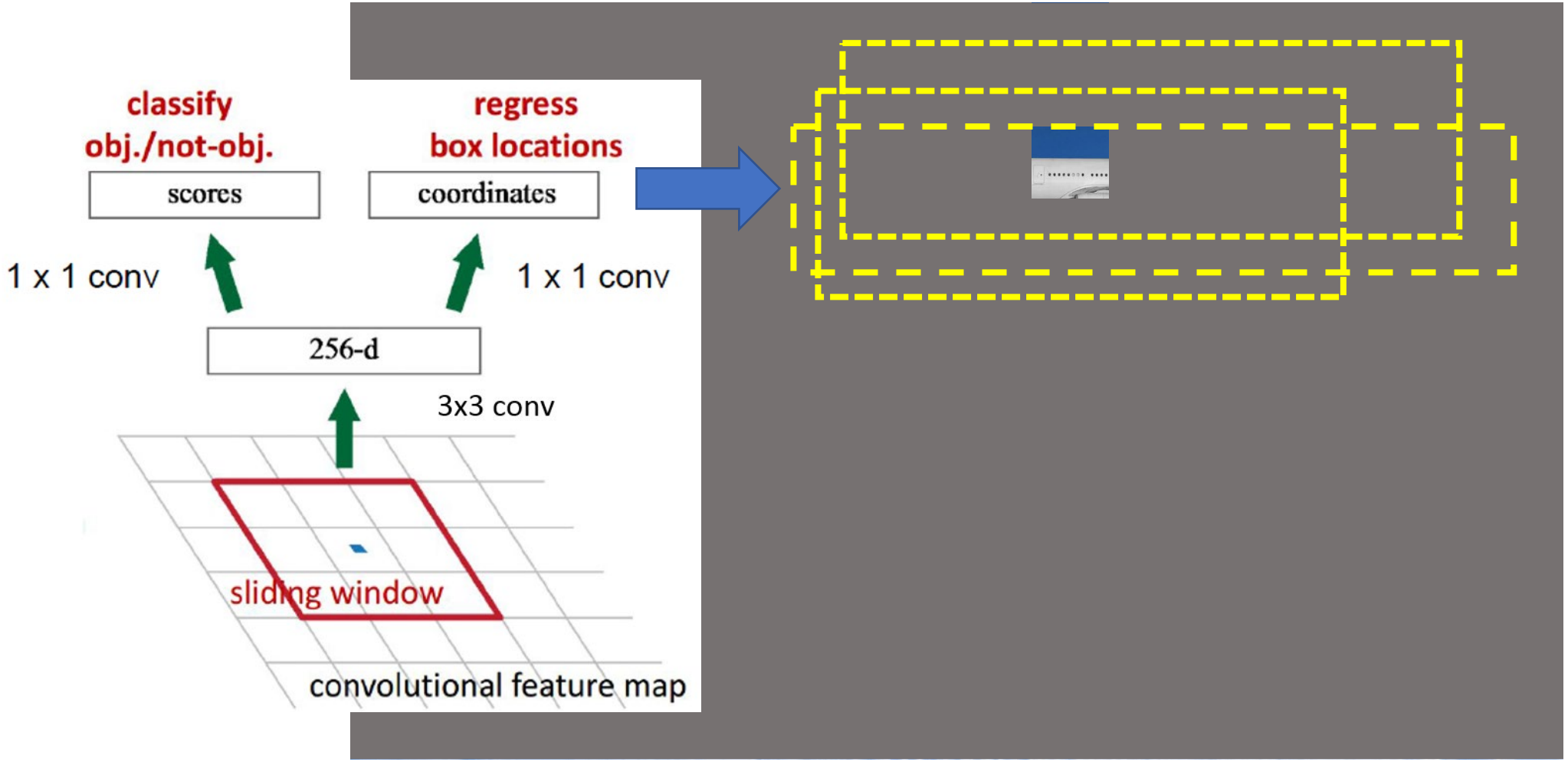


Frequently used method
- Good for locating objects with different scales
- Bad due to computation overhead

Proposed anchor (=reference) method
- Good for small computation overhead
(only one feature is enough for different scales)
- Accuracy? Proven in this paper!

[Girshick, 2015]

Each Grid Gives Multiple (=9) Region Proposals: One RP for an Anchor

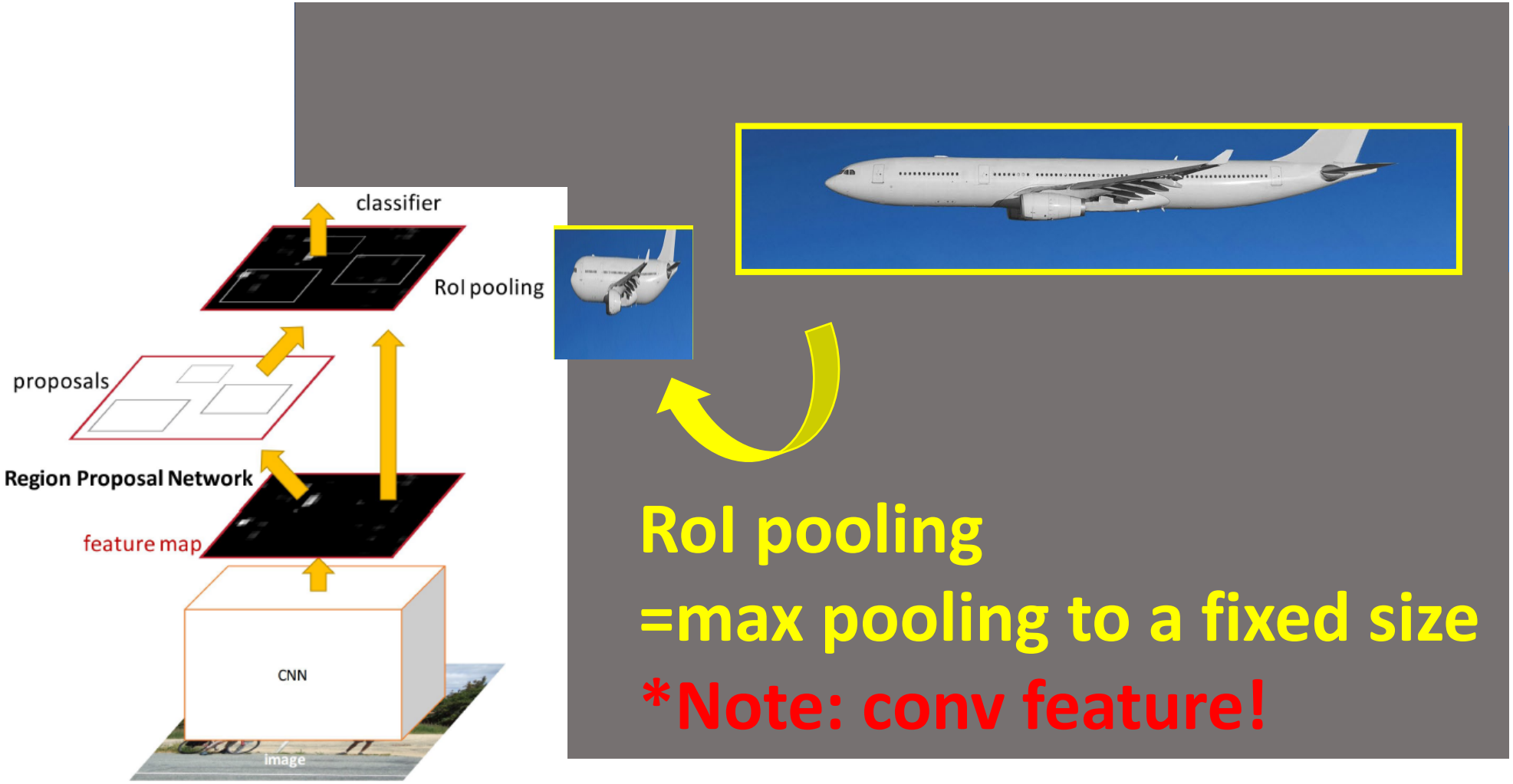


[S.Yoo]

Flow Example: Take the Box from the Image

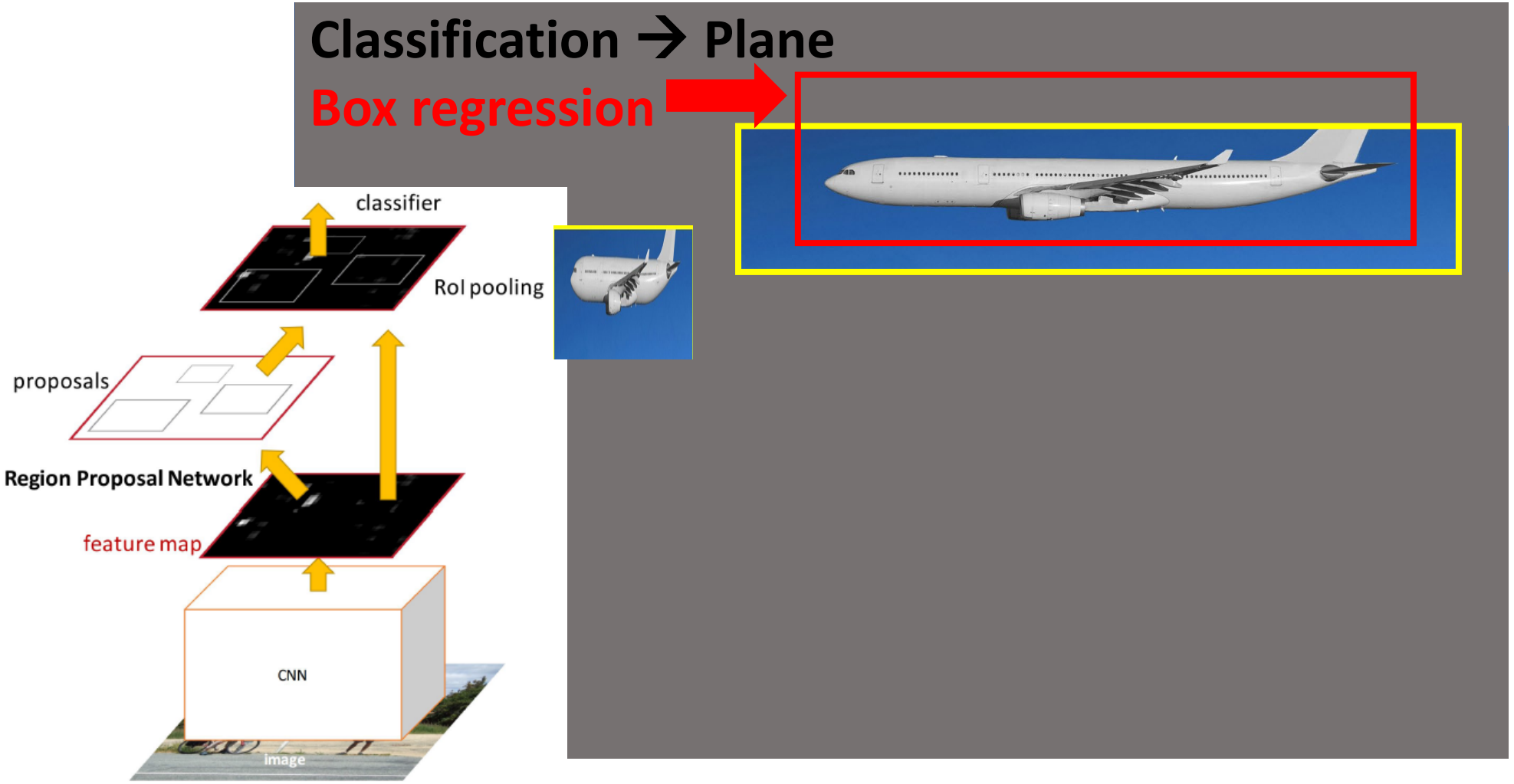
[S.Yoo]

Flow Example: Warp it into a Uniform Size (via RoI Pooling)



[S.Yoo]

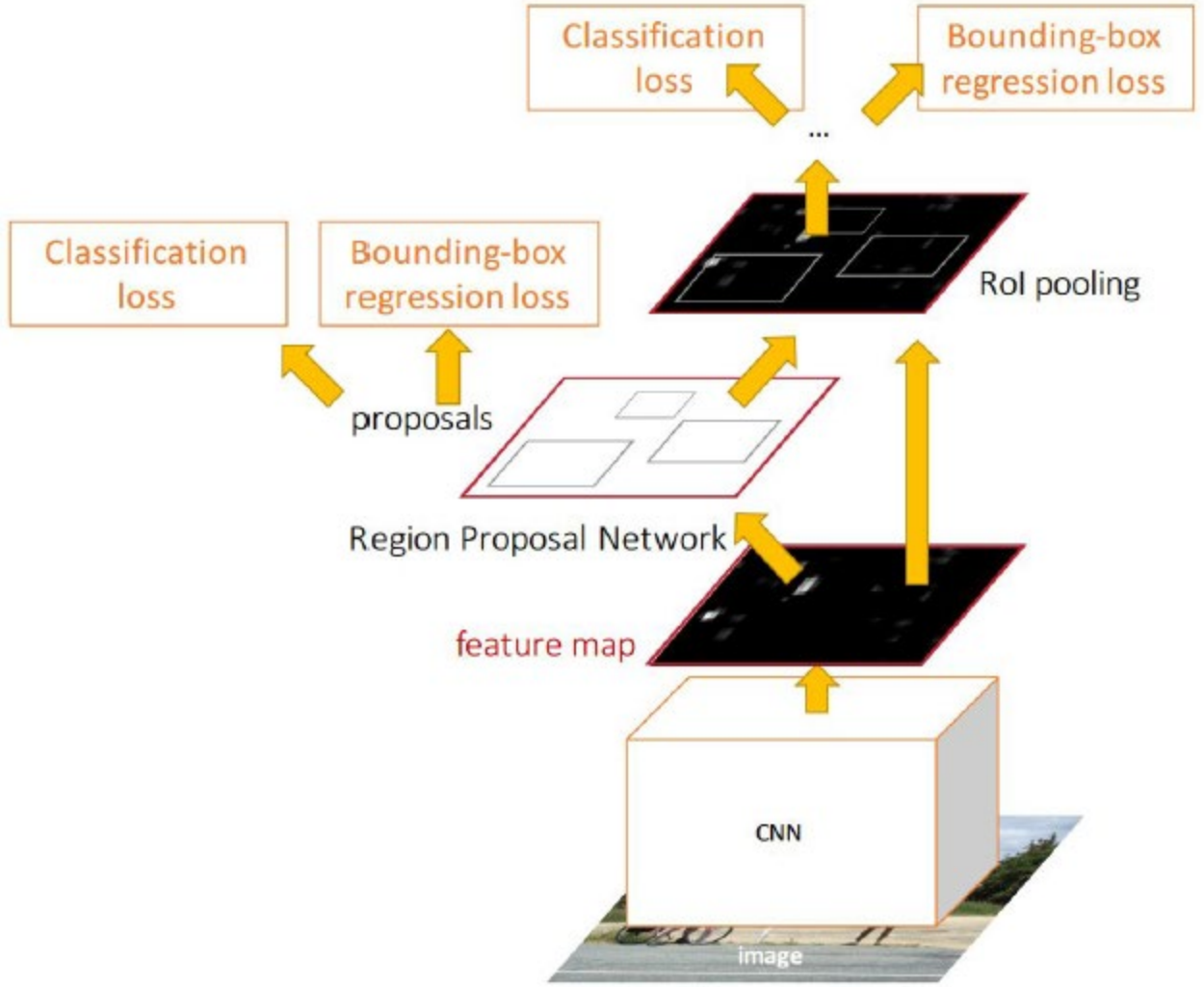
Flow Example: Fast RCNN gives Classification and Box



[S.Yoo]

Training Faster-RCNN

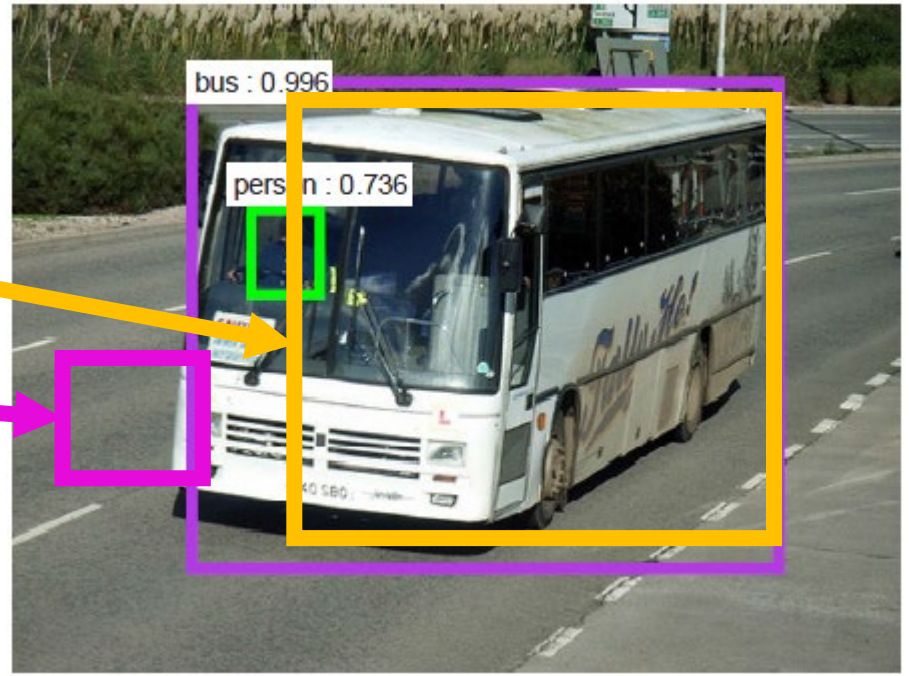
- Alternating training
  - Repeat training of RPN + CNN and that of class/BB + CNN
- Joint training
  - Combine loss from RPN and class/BB outputs
  - RoI pooling is treated as a kind of max pooling layer



[S.Yoo]

Positive and Negative Anchors in Training

- Positive anchor
  - IoU (intersection over union) with ground truth box > 0.7 or top IoU for the ground truth class
- Negative anchor
  - IoU with any ground truth box < 0.3
- 25 %/75 % ratio of positive/negative samples in a minibatch



[S.Yoo]

Loss Function to Train RPN with An Anchor

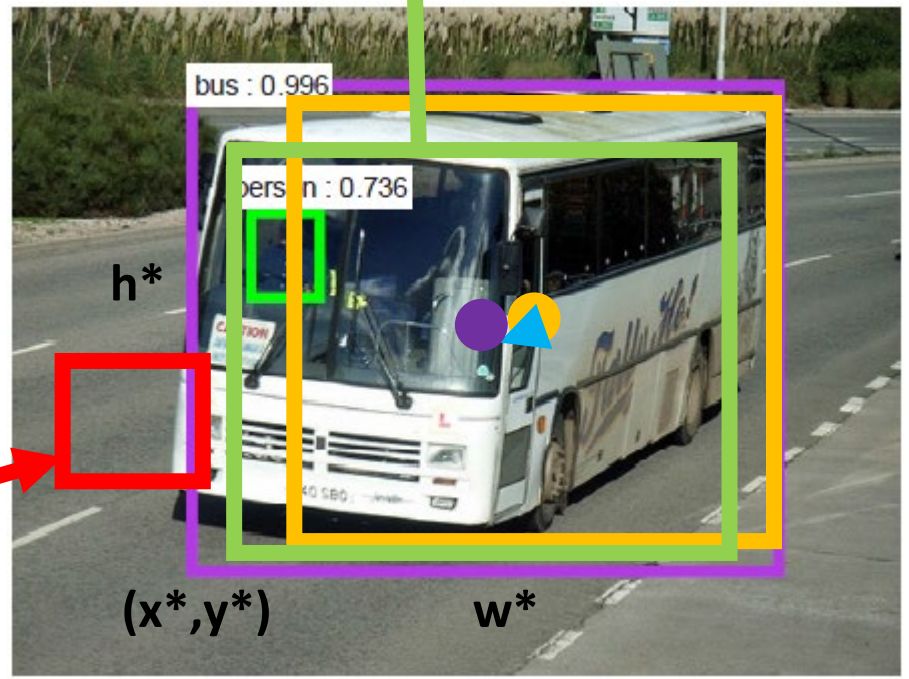
$$t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a,$$
$$t_w = \log(w/w_a), \quad t_h = \log(h/h_a),$$
$$t_x^* = (x^* - x_a)/w_a, \quad t_y^* = (y^* - y_a)/h_a,$$
$$t_w^* = \log(w^*/w_a), \quad t_h^* = \log(h^*/h_a),$$

$$L_{loc}(\ t^*,t\ ) = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L_1}(\ \ t^*\text{-}t\ \ )$$

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

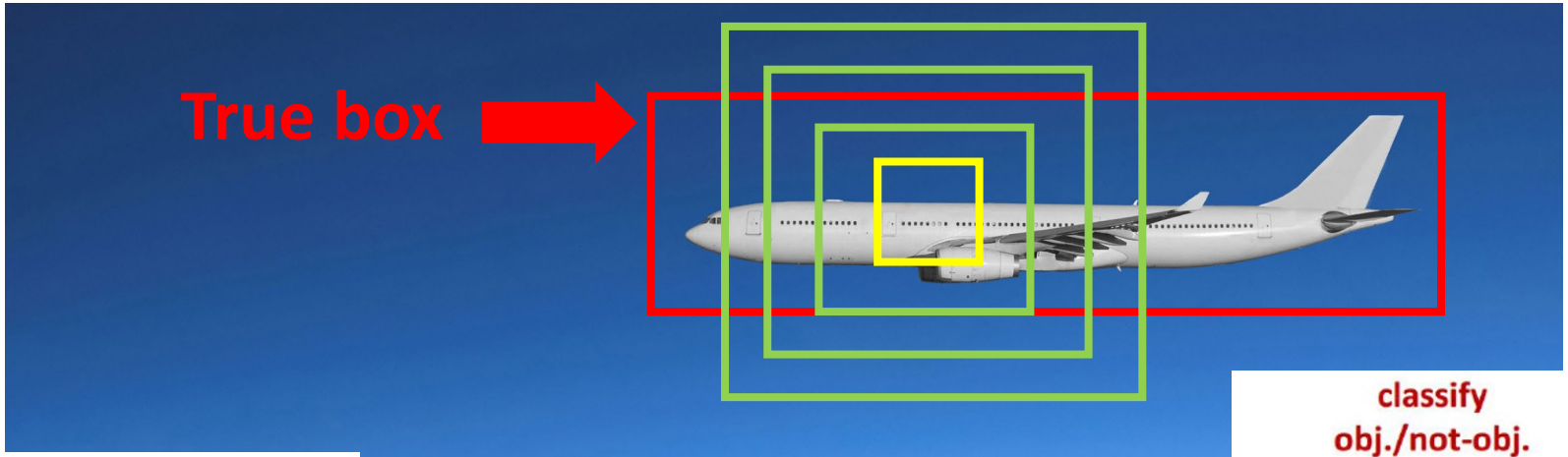
$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*)$$
$$+\lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*).$$

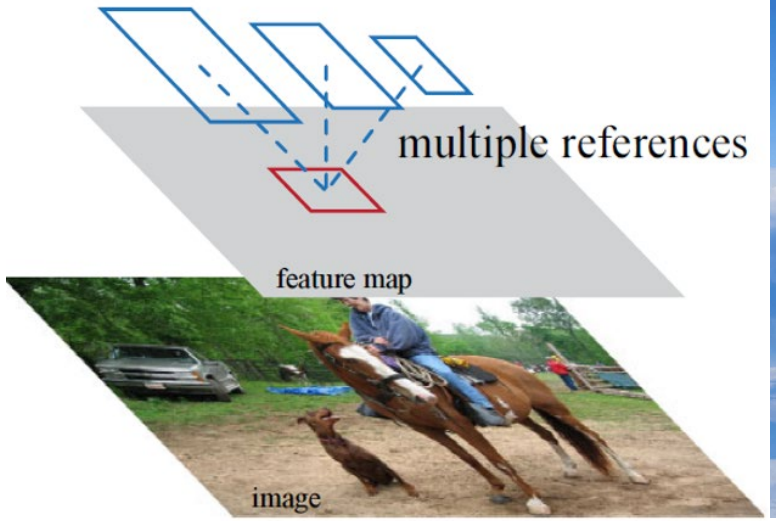
P_i^*=0
for background

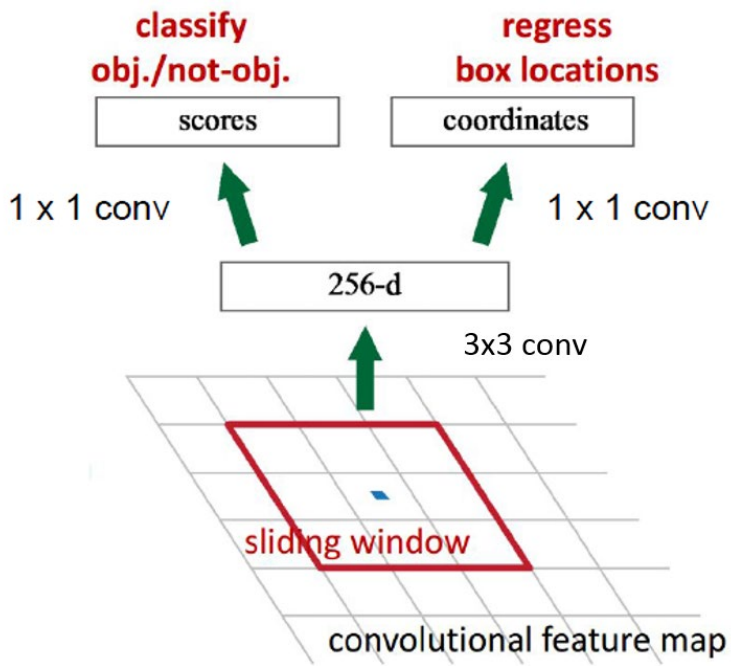


[S.Yoo]

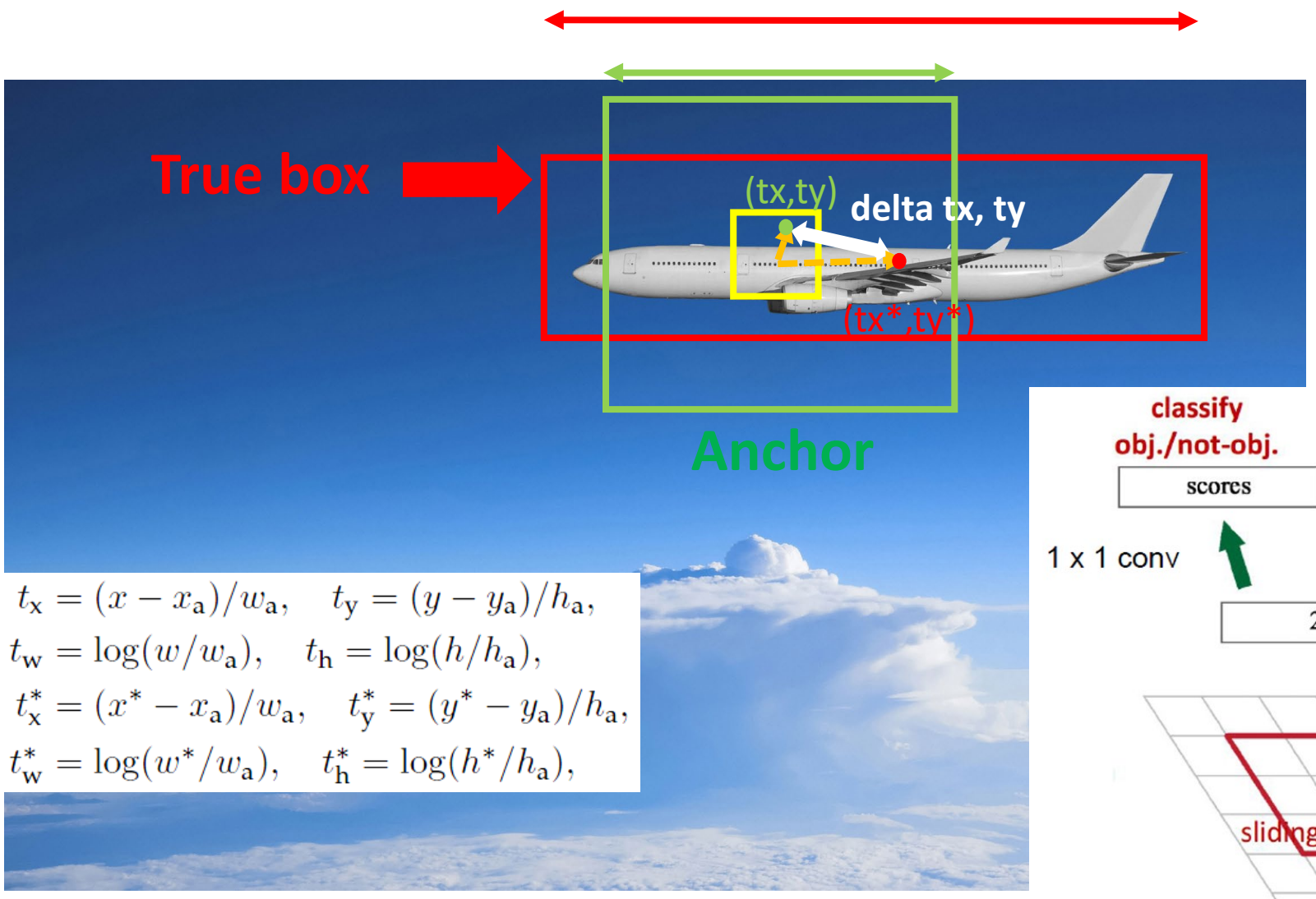
How Are Anchors Used in Training?



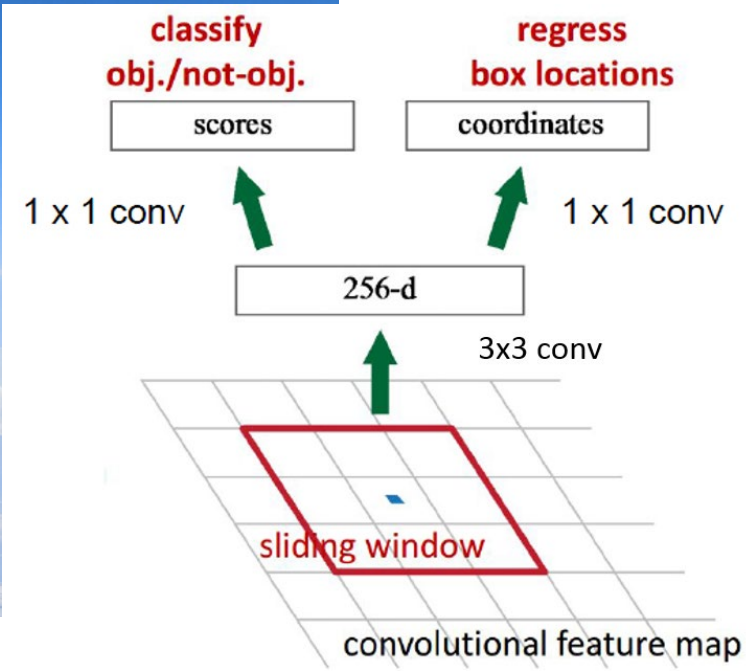




How Are Anchors Used in Training?



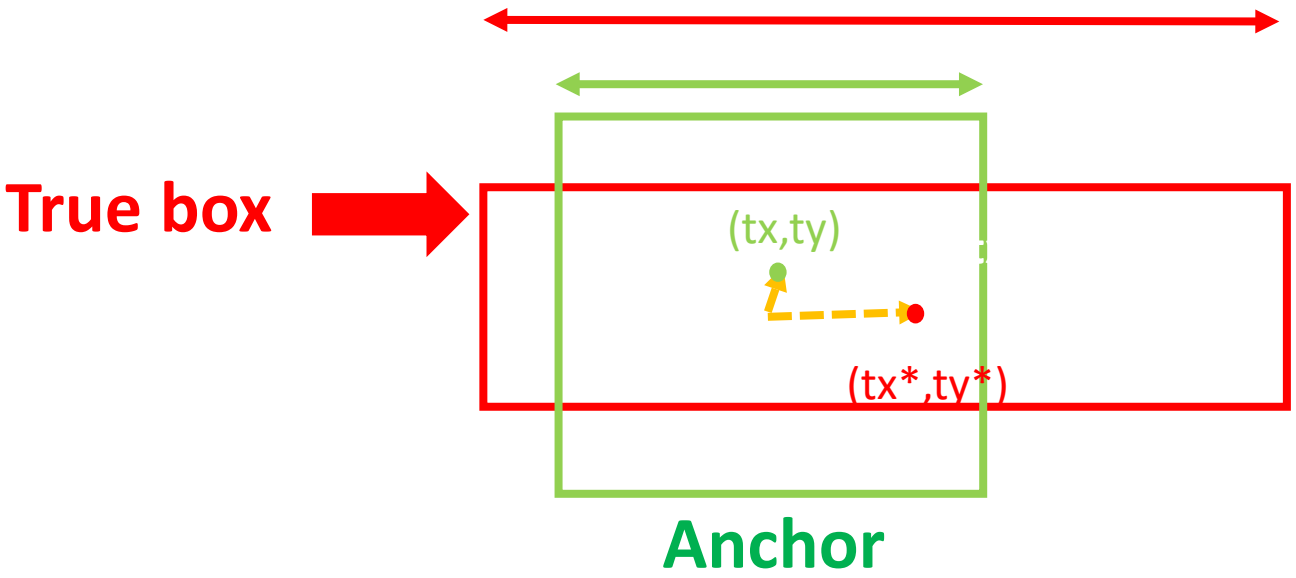
$$t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a,$$
$$t_w = \log(w/w_a), \quad t_h = \log(h/h_a),$$
$$t_x^* = (x^* - x_a)/w_a, \quad t_y^* = (y^* - y_a)/h_a,$$
$$t_w^* = \log(w^*/w_a), \quad t_h^* = \log(h^*/h_a),$$



Residual Training

- Insight
  - The smaller the function gives, the easier to learn the function

- Residual network
  - Y = X + **f(X)**
- Anchor-based box prediction
  - **W** = W'/W_{anchor}



[S.Yoo]

Faster Object Detection by Reducing Runtime for Region Proposal

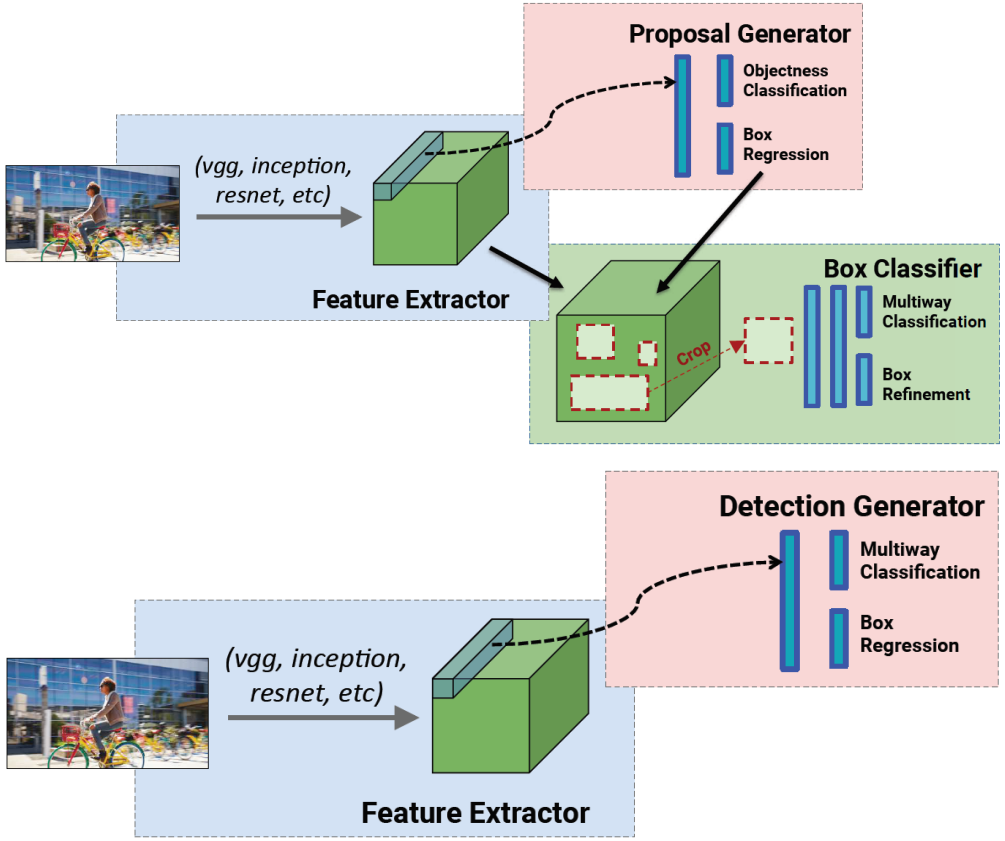
- RPN takes ~50% of conv layer runtime
  - ZF conv layer case

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

[S.Yoo]

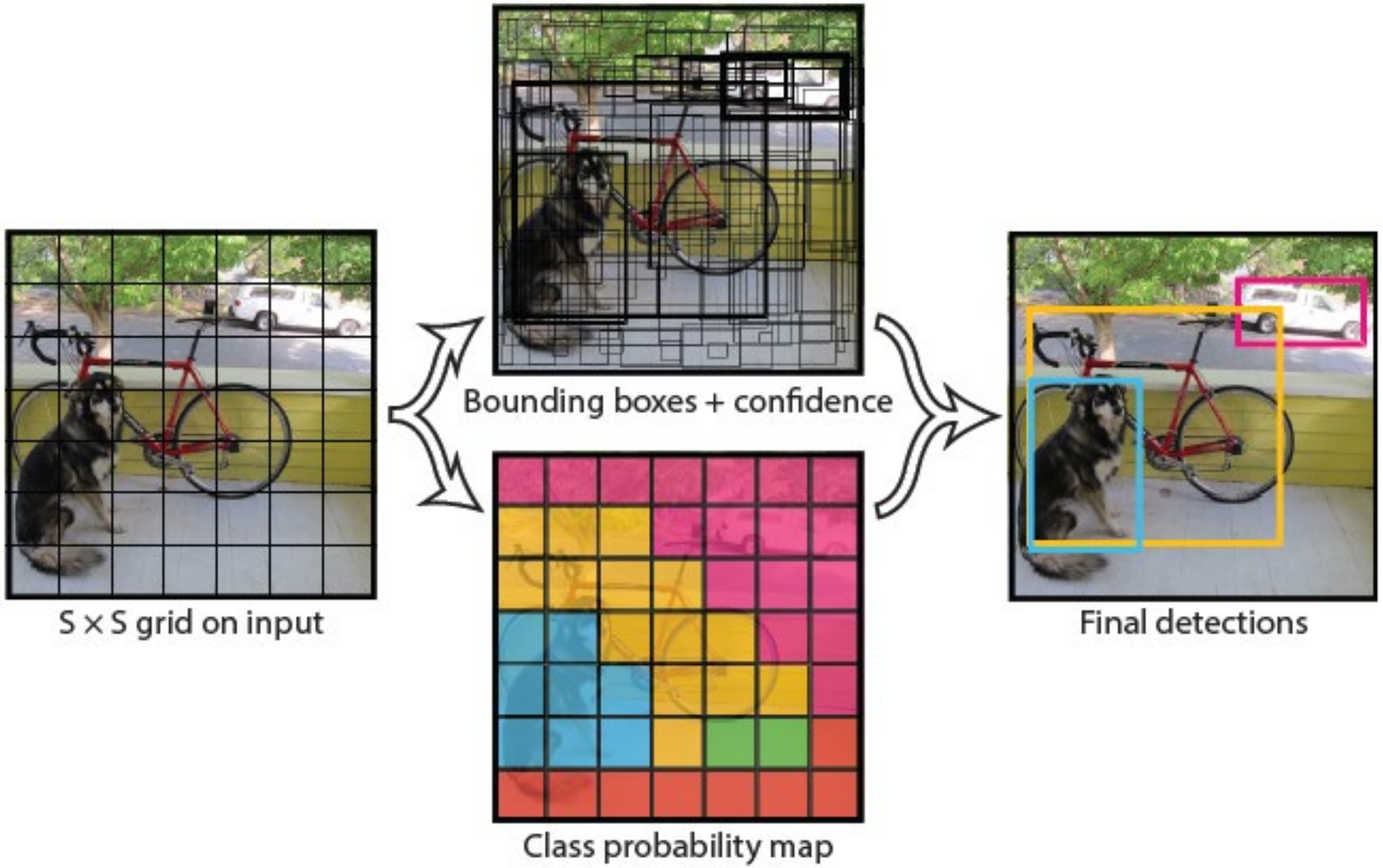
Object Detection Methods

- Two-step methods
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
  - Mask R-CNN
  - Light-Head R-CNN
- Single shot methods
  - YOLO
  - SSD
  - YOLO2
  - DSSD
  - RetinaNet
  - YOLOv3
- Anchor-free methods
  - CornerNet
  - CenterNet



[Huang, 2017]

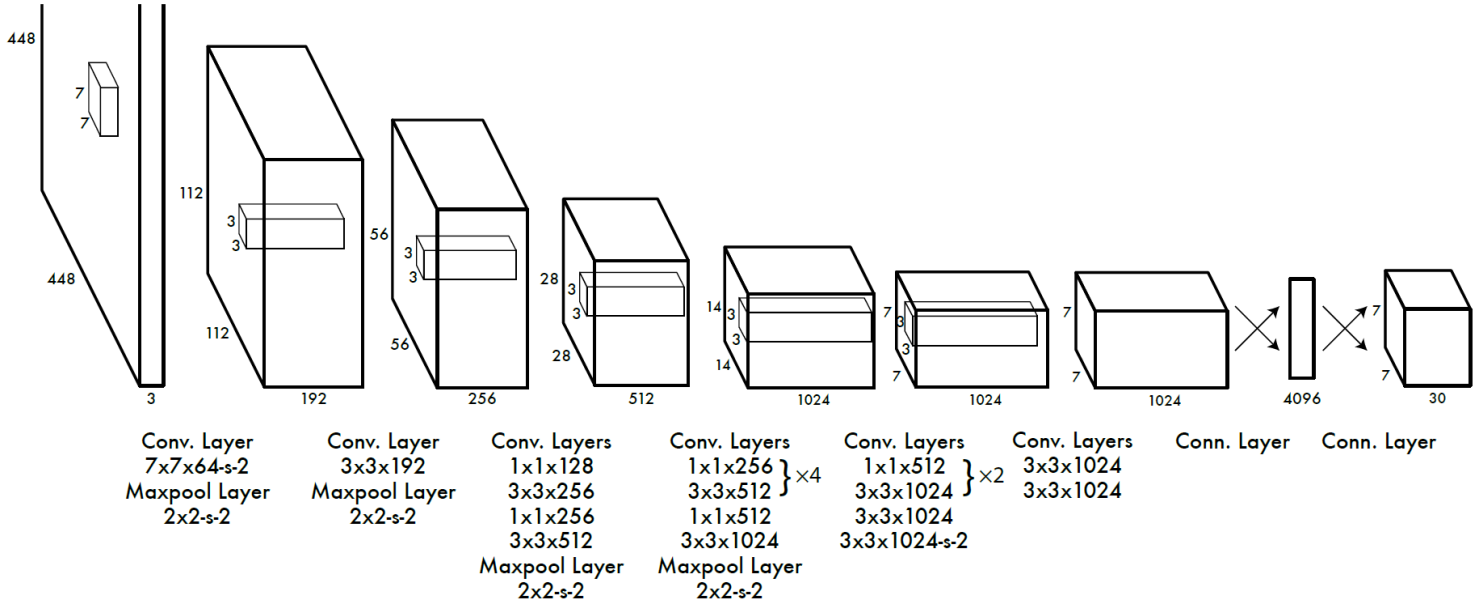
YOLO (You Look Only Once)

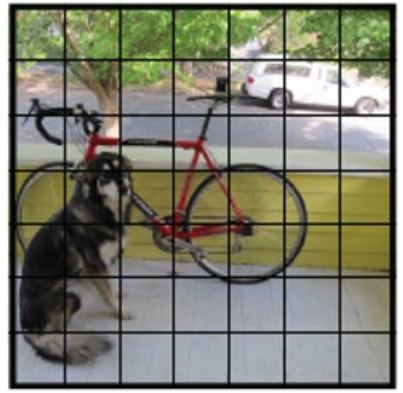


[YOLO]

Network Design

- 24 CONV (9 Conv in Fast YOLO) + 2 FC layers
- Each of 7x7 grid cells gives following outputs:
  - B bounding boxes with coordinates (4-tuple) with confidence score
    - $confidence\ score = \Pr(containing\ an\ object) \times IoU(pred, truth)$
  - Probability of each class $C_i, \Pr(object \in C_i \mid containing\ an\ object)$



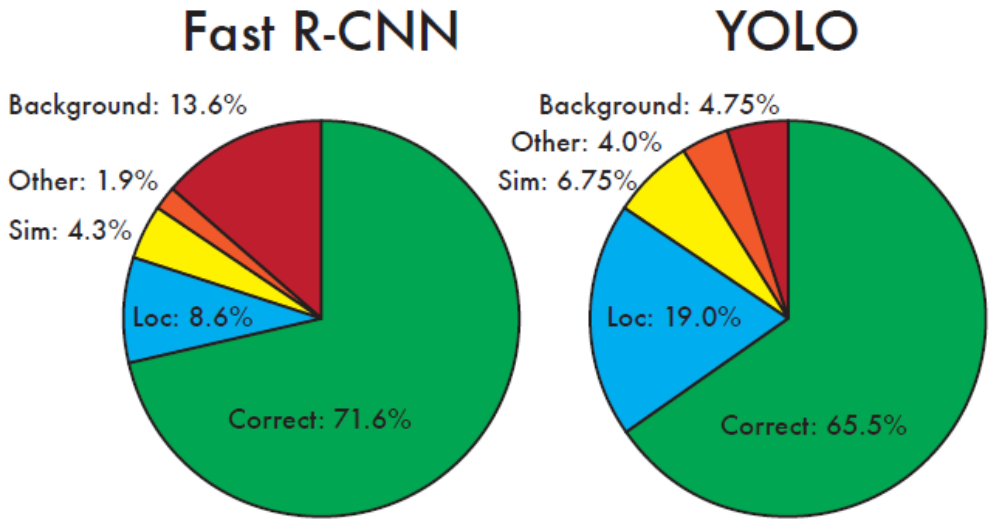


[YOLO]

Performance & Limitation

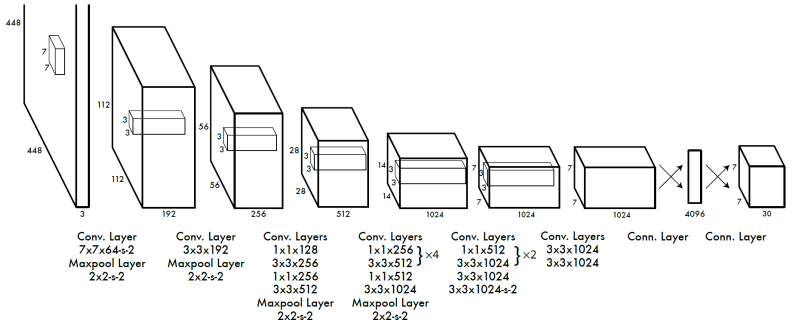
- High location error
- Difficulty with small objects, e.g., groups of birds

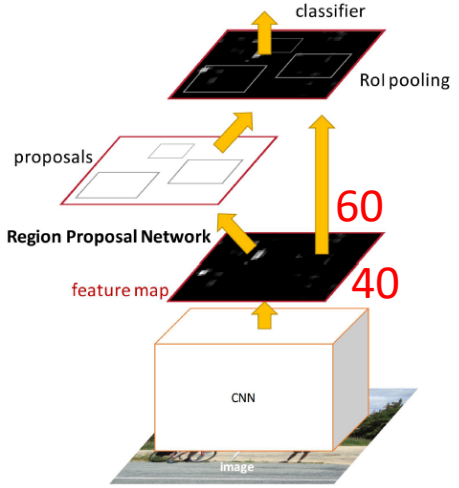
Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21



[YOLO]

Faster RCNN vs. YOLO

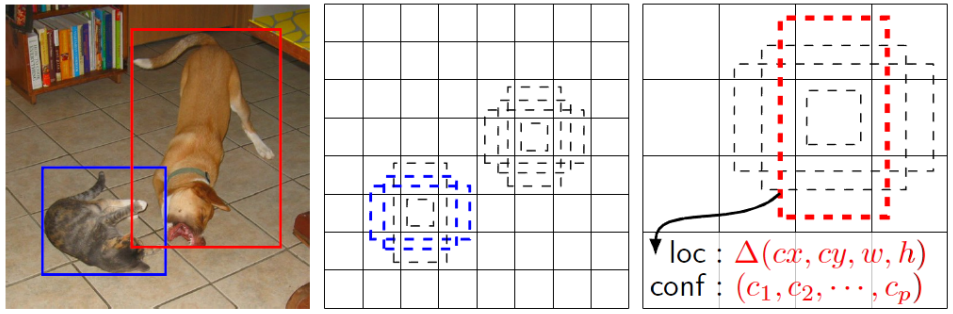




- Global or local information
  - YOLO predicts boxes in a single step with global info
  - Faster RCNN utilizes local feature maps (after ROI pooling) to predict boxes
- Spatial resolution
  - 7x7 vs 40x60
  - Low resolution may be one of key reasons in YOLO's poor performance with small objects
- # box proposals
  - YOLO: 2 box proposals / grid cell * 7x7 = 98 box proposals
  - Faster RCNN
    - 9 box proposals at each of 40x60 points → 300 box proposals are selected among them

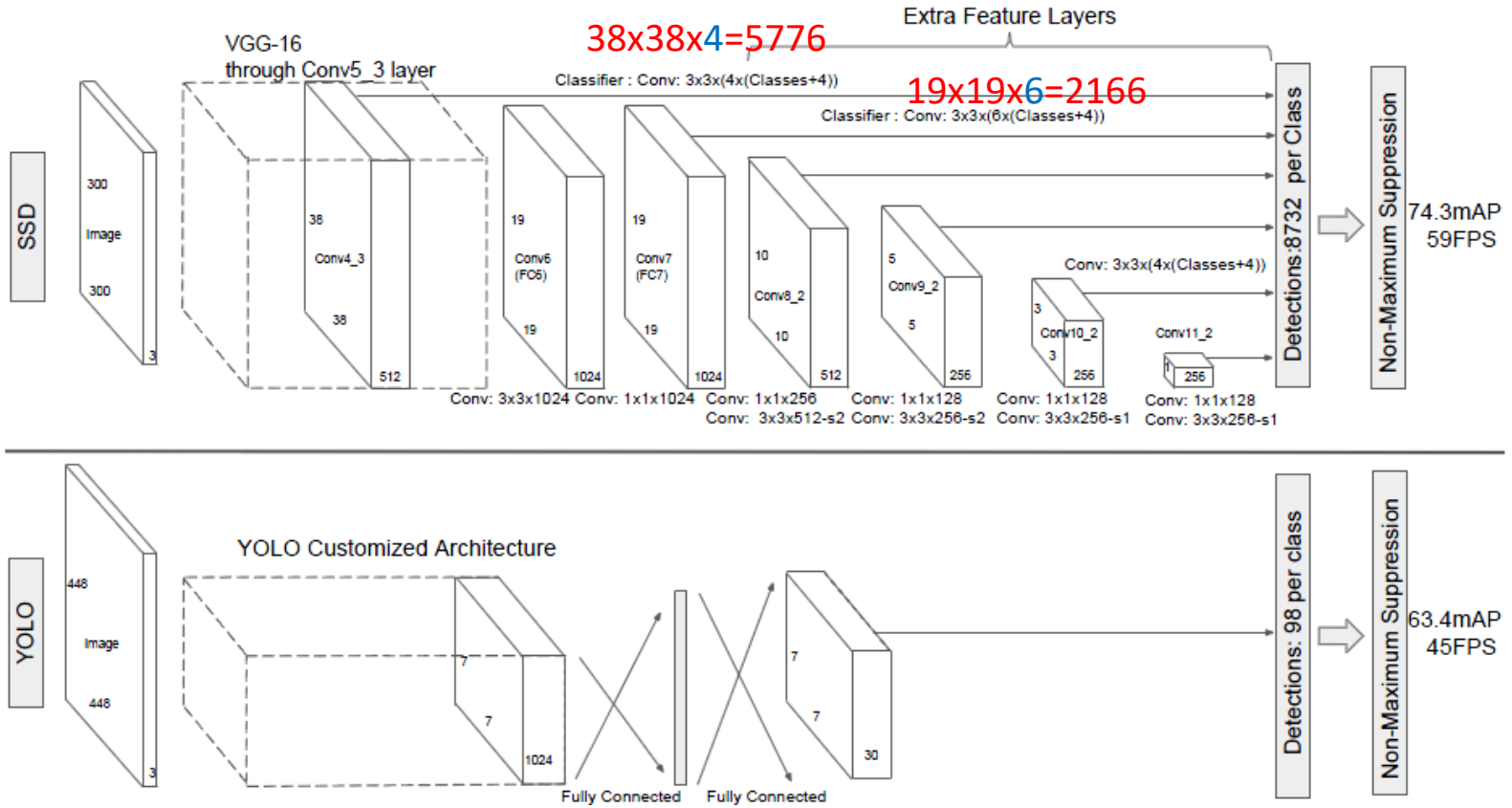
[S.Yoo]

SSD (Single Shot Detection)



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

- Anchors are applied several times for each resolution to produce numerous candidates
- Helpful to capture the objects regardless of their size



VOC 07

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

Faster RCNN w/ ResNet-101

system	net	data	mAP
baseline	VGG-16	07+12	73.2
baseline	ResNet-101	07+12	76.4
baseline+++	ResNet-101	COCO+07+12	**85.6**

	SSD300				
more data augmentation?		✔	✔	✔	✔
include $\{\frac{1}{2}, 2\}$ box?	✔		✔	✔	✔
include $\{\frac{1}{3}, 3\}$ box?	✔			✔	✔
use atrous?	✔	✔	✔		✔
VOC2007 `test` mAP	65.5	71.6	73.7	74.2	**74.3**

Prediction source layers from:						mAP use boundary boxes?	
conv4_3	conv7	conv8_2	conv9_2	conv10_2	conv11_2	Yes	No
✔	✔	✔	✔	✔	✔	74.3	63.4
✔	✔	✔	✔	✔		**74.6**	63.1
✔	✔	✔	✔			73.8	68.4
✔	✔	✔				70.7	69.2
✔	✔					64.2	64.4
	✔					62.4	64.0

[SSD]

YOLO v2

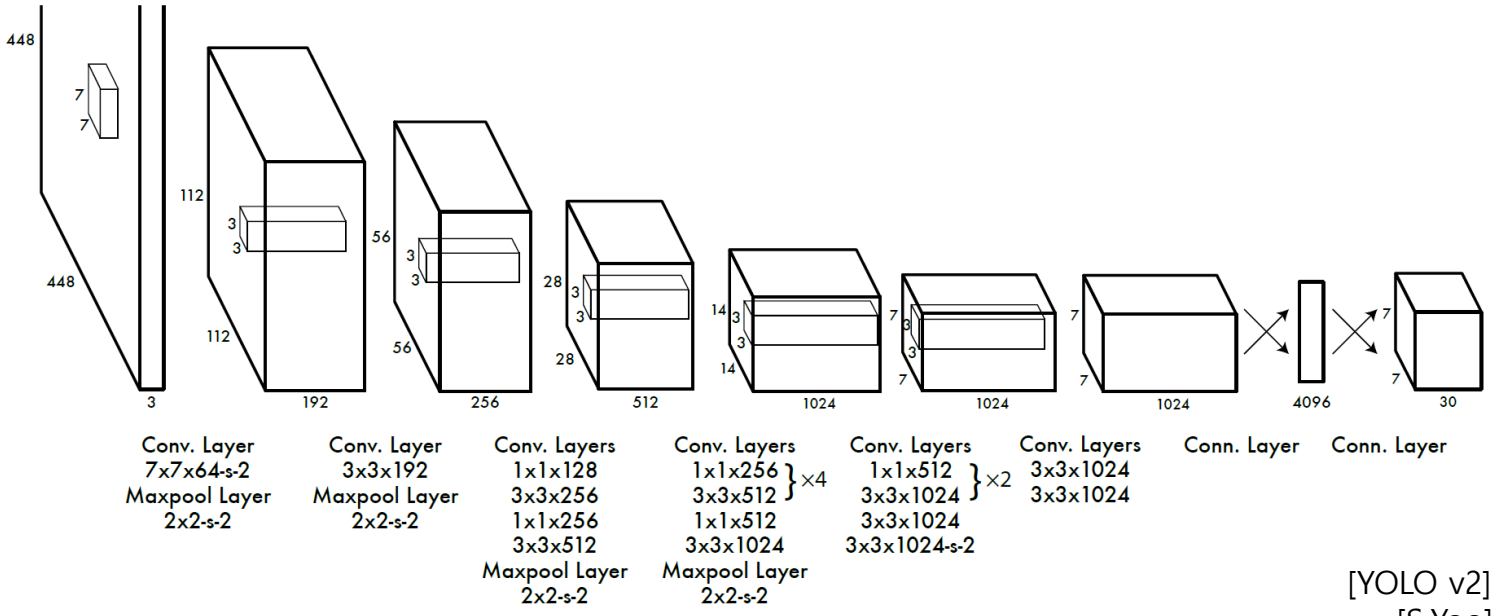
- Incremental update – mixture of existing idea & newly proposed method

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

[YOLO v2]

YOLO v2

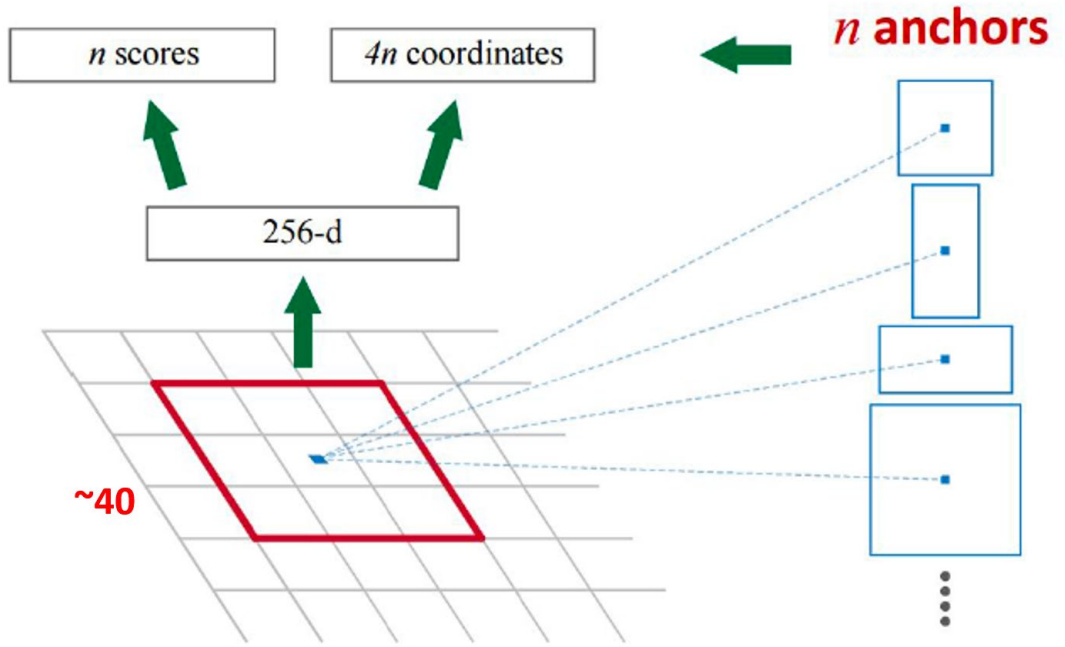
- Batch normalization: 2% mAP
- High resolution classifier: 4% mAP
  - YOLO: Training w/ 224x224 classification → Object detection w/ 448x448
  - YOLO v2: Training 224 → Fine-tuning 448 → Object detection w/ 448x448
- Multi-scale input training
  - 320x320, …, 608x608



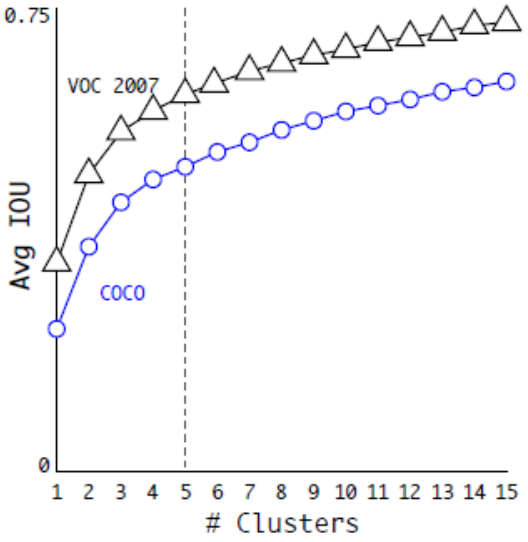
[YOLO v2]
[S.Yoo]

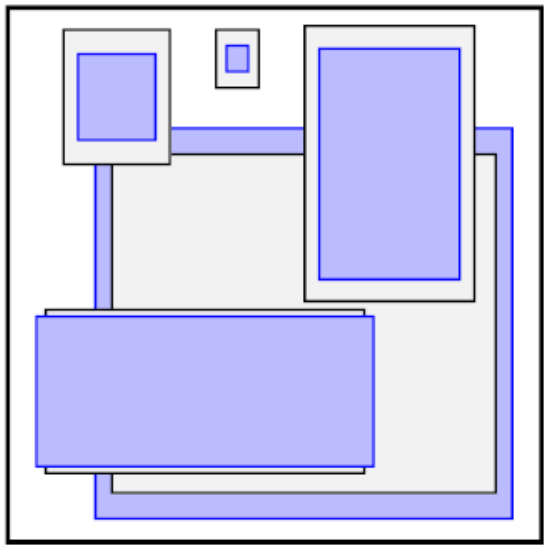
Anchor Box

- More box predictions
  - YOLO 2x7x7 boxes <-> 5x13x13 boxes
  - Per box, 20 classes + 4 coordinates + 1 IoU
  - 5 boxes x (20 + 4 + 1) = 125 outputs
- Dimension prior



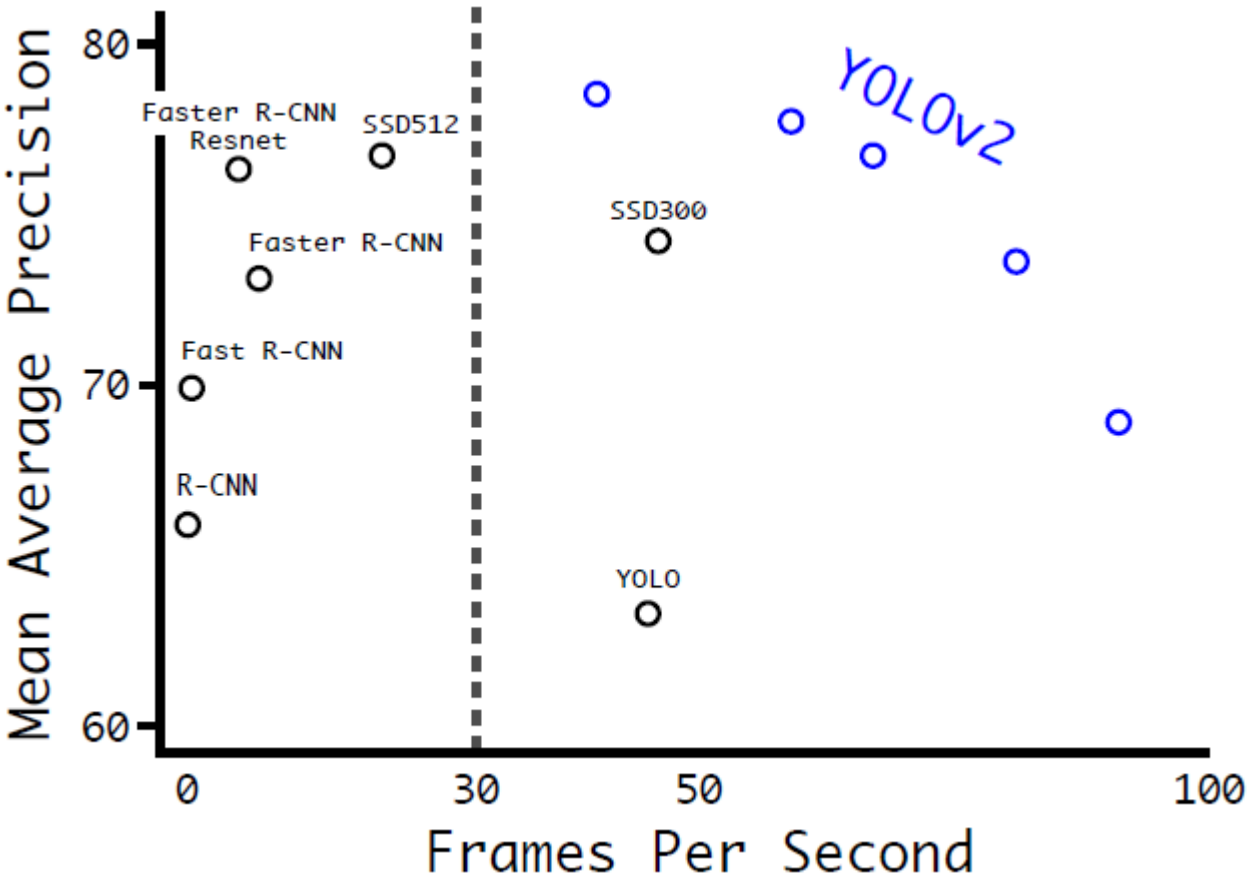
	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓
anchor boxes?				✓	✓			
new network?					✓	✓	✓	✓
dimension priors?						✓	✓	✓
location prediction?						✓	✓	✓
passthrough?							✓	✓
multi-scale?								✓
hi-res detector?								✓
VOC2007 mAP	63.4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





[YOLO v2]
[S.Yoo]

YOLOv2 vs. Others



Faster RCNN w/ ResNet-101, VOC 07

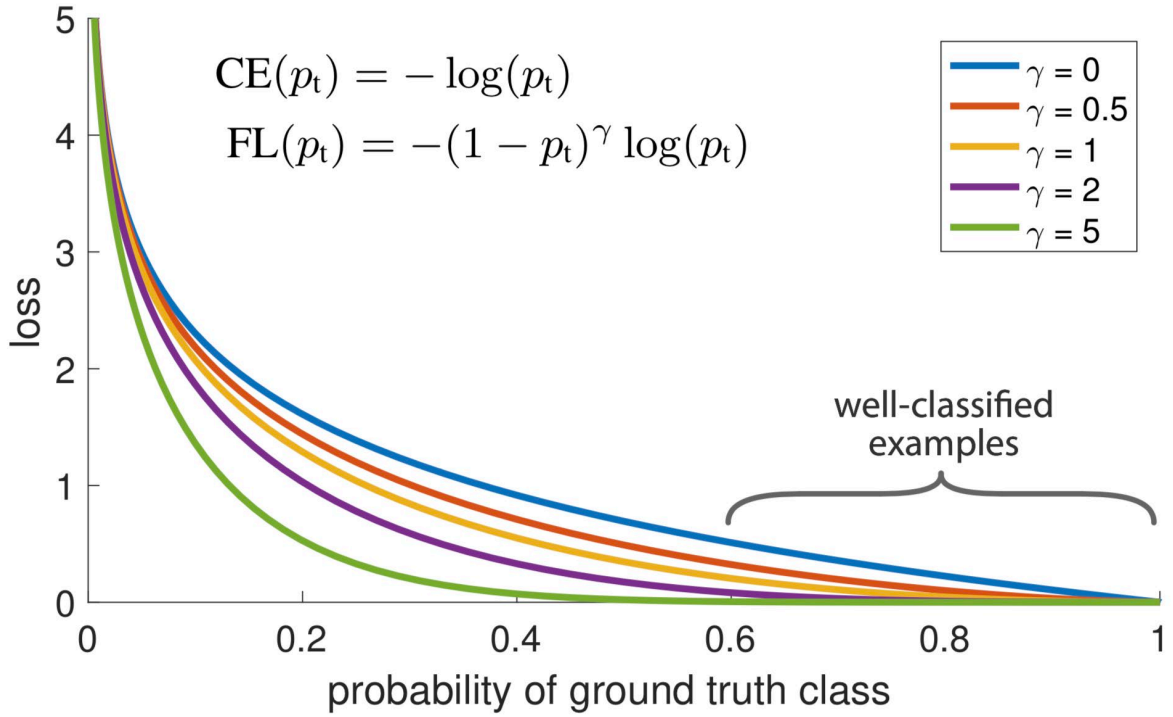
system	net	data	mAP
baseline	VGG-16	07+12	73.2
baseline	ResNet-101	07+12	76.4
baseline+++	ResNet-101	COCO+07+12	**85.6**

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

[YOLO v2]

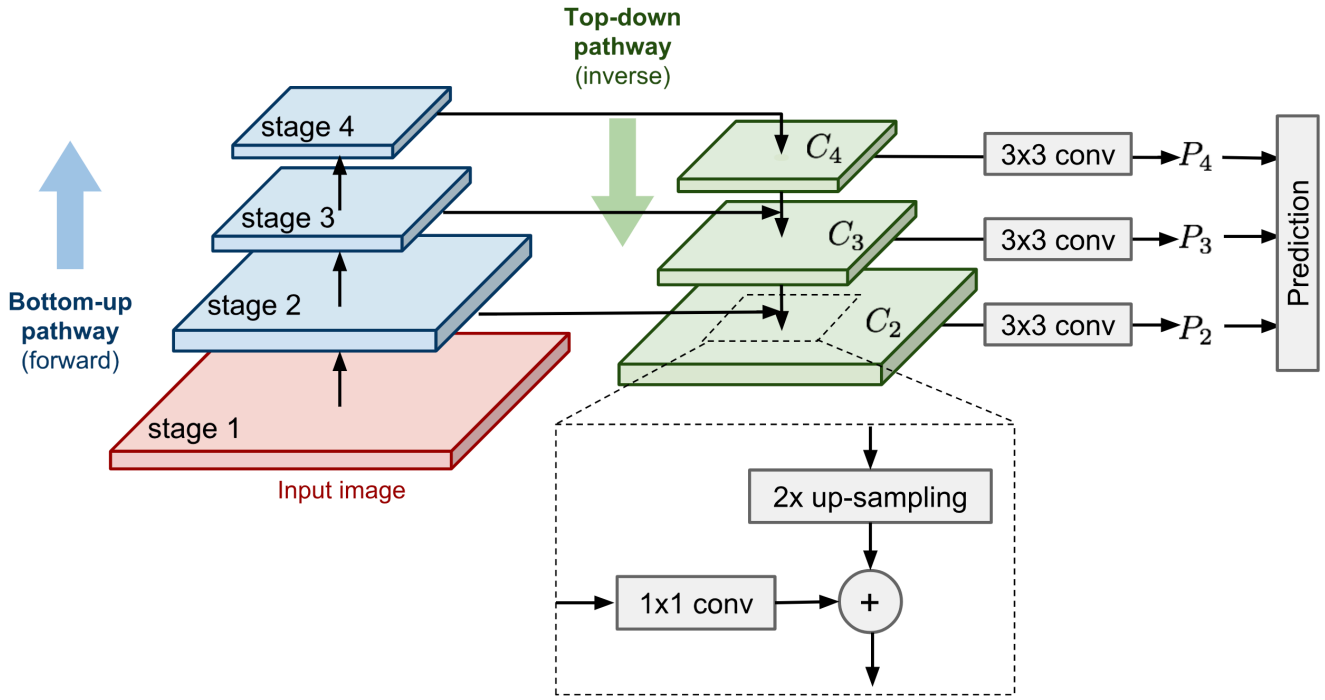
Focal Loss of RetinaNet

- Focal loss is designed to assign more weights on hard examples and to down-weight easy examples
  - $FL(p_t) = (1 - p_t)^{\gamma} \log p_t$



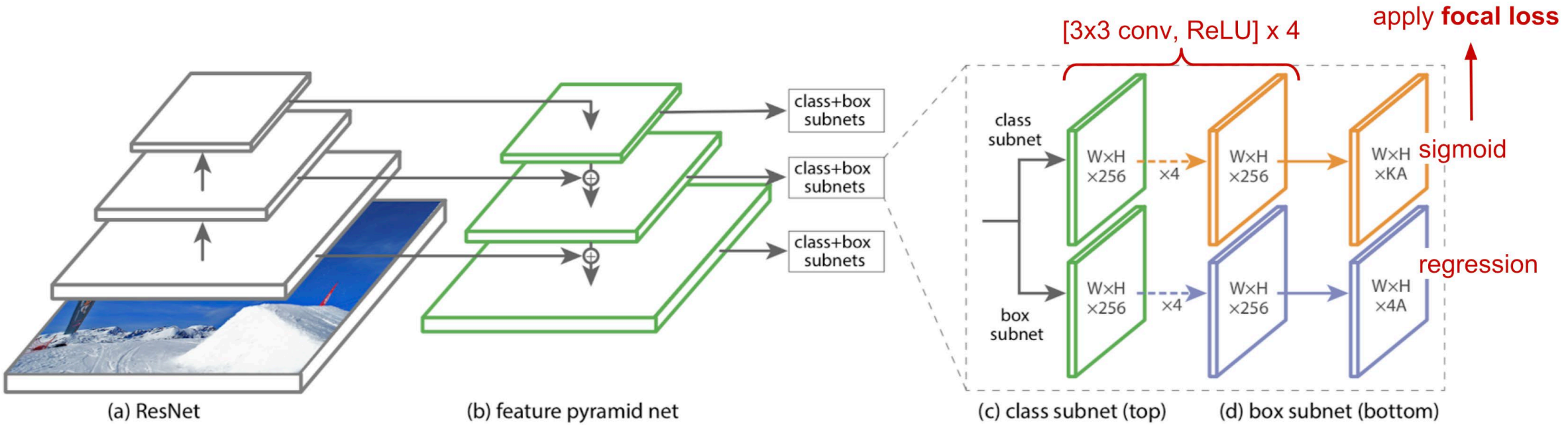
RetinaNet Architecture

- Based on featured image pyramid network (FPN)
  - Bottom-up: normal feed forward
  - Top-down: can exploit semantically compressed feature maps

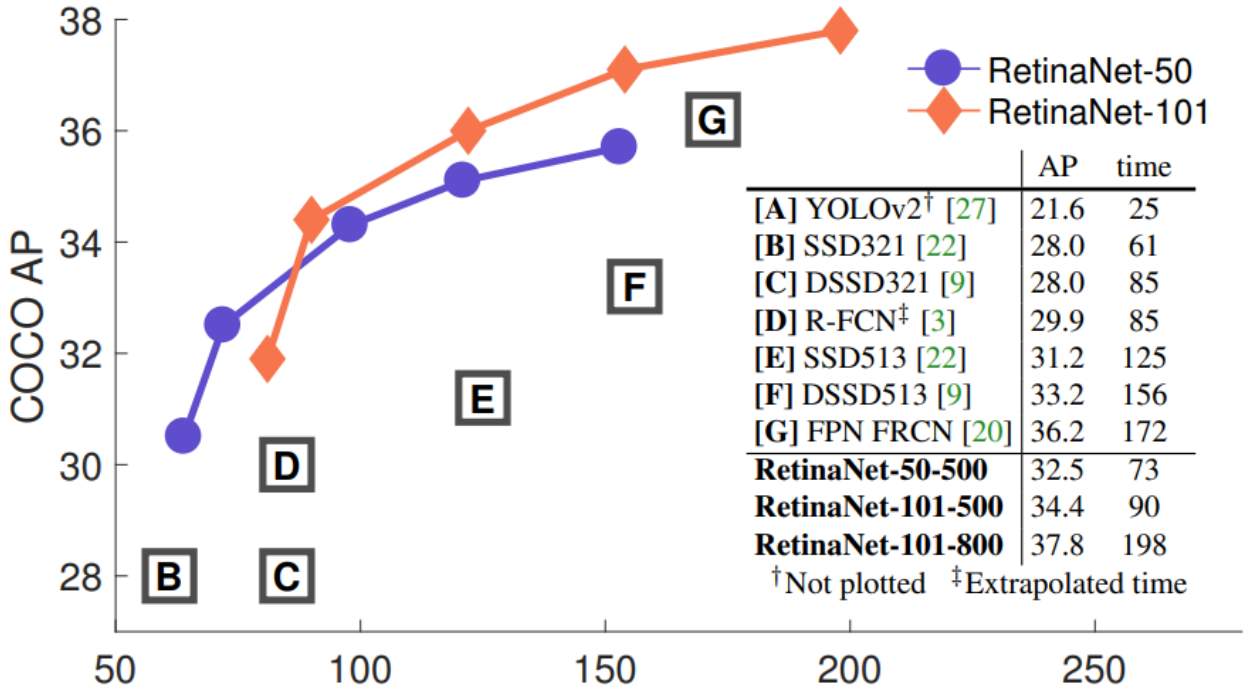


RetinaNet Architecture - 2

- FPN-based feature extractor + per-resolution prediction
  - Use 9 anchors per level
  - Simplified backbone network / increased depth of subnetwork



Results of RetinaNet



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

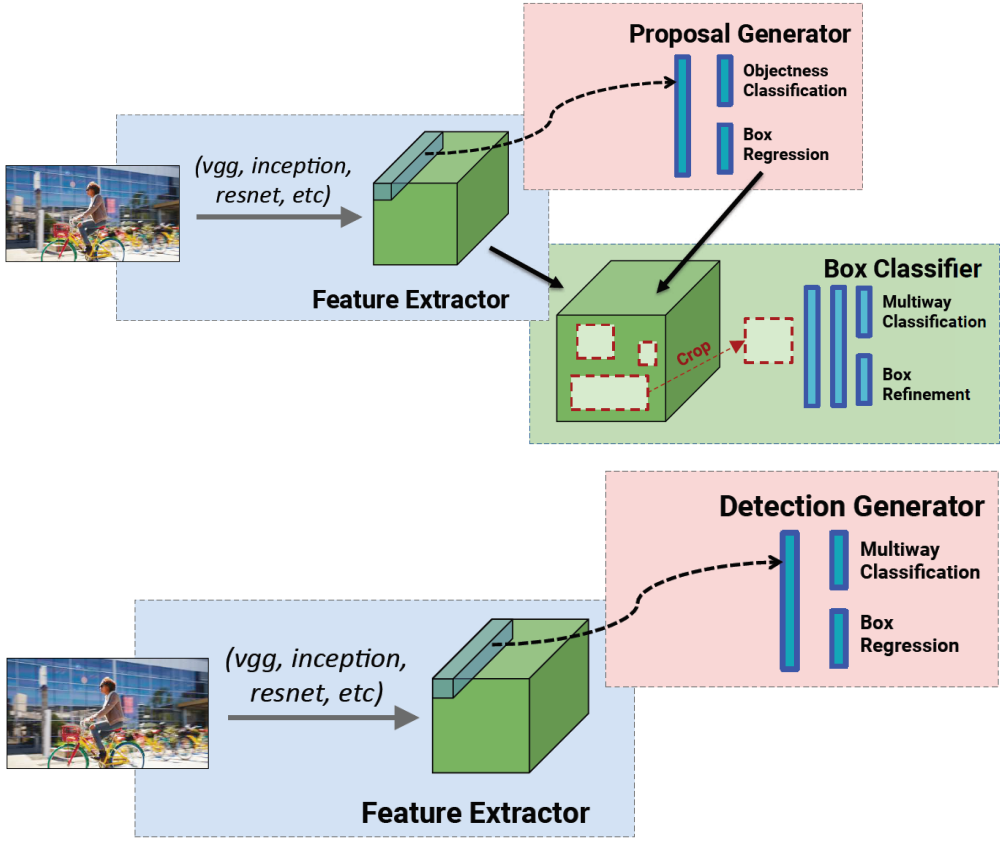
†Not plotted ‡Extrapolated time

γ	α	AP	AP_{50}	AP_{75}
0	.75	31.1	49.4	33.0
0.1	.75	31.4	49.9	33.1
0.2	.75	31.9	50.7	33.4
0.5	.50	32.9	51.7	35.2
1.0	.25	33.7	52.0	36.2
2.0	.25	**34.0**	**52.5**	**36.5**
5.0	.25	32.2	49.6	34.8

(b) **Varying γ for FL** (w. optimal α)

Object Detection Methods

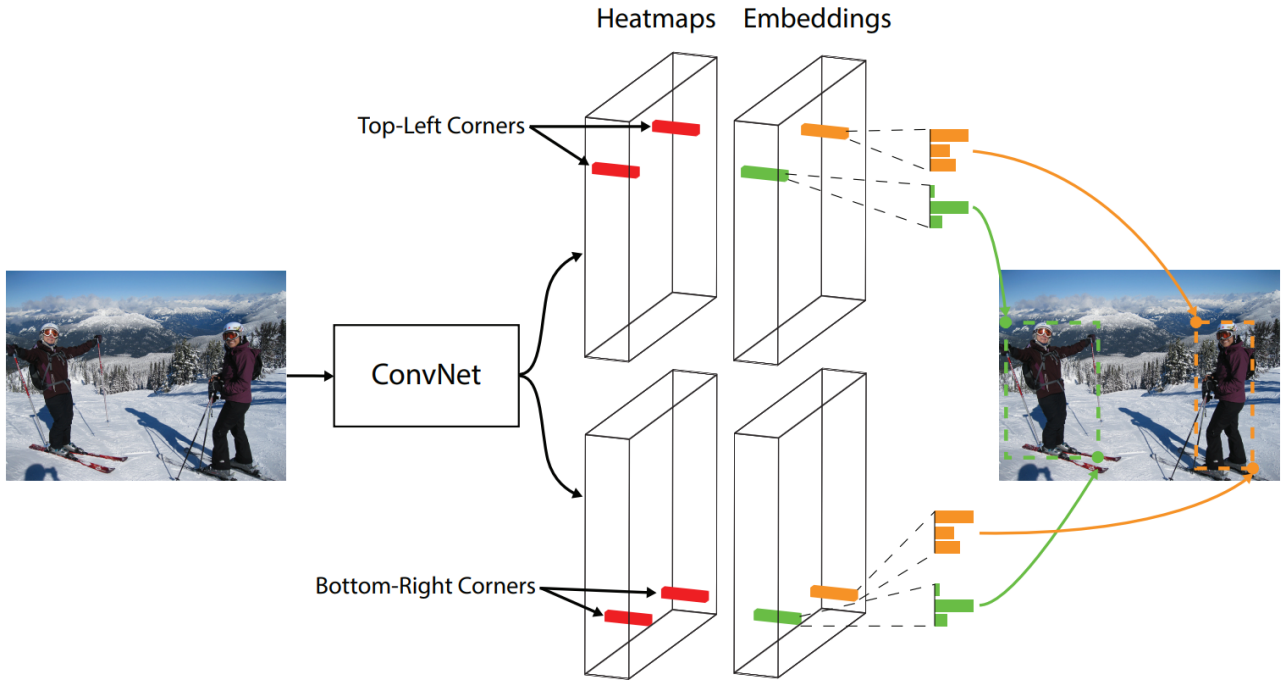
- Two-step methods
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
  - Mask R-CNN
  - Light-Head R-CNN
- Single shot methods
  - YOLO
  - SSD
  - YOLO2
  - DSSD
  - RetinaNet
  - YOLOv3
- Anchor-free methods
  - CornerNet
  - CenterNet



[Huang, 2017]

CornerNet Idea

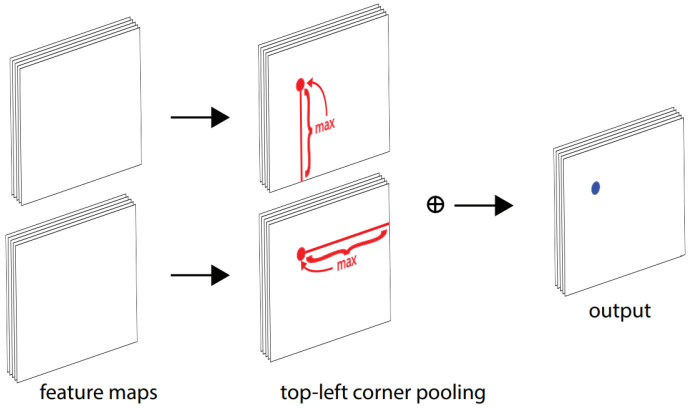
- Predict top-left & bottom-right corner instead of bounding box
  - Also predict embedding vector to identify object/corner pair
  - Create bounding box by combining two corner pair having high similarity

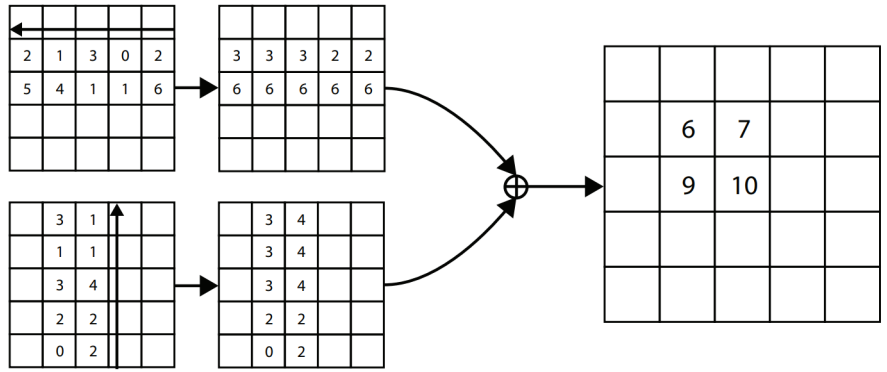


Corner Pooling

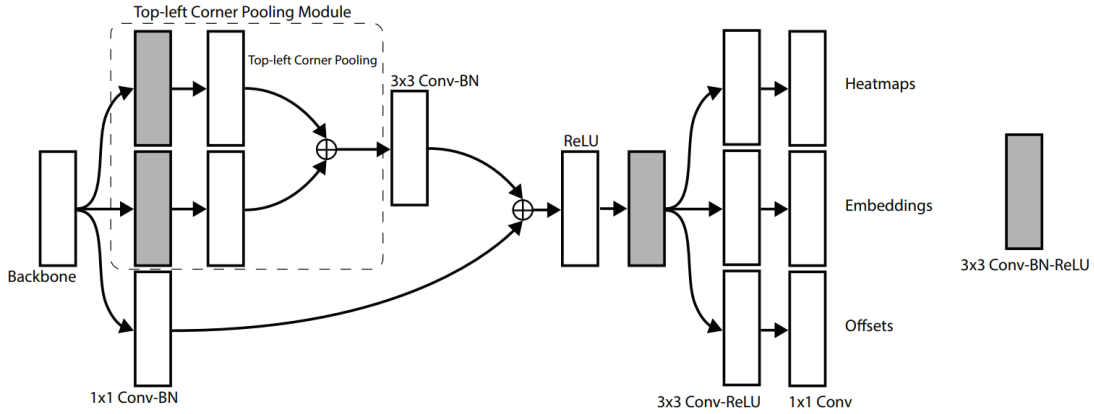
- There is no local evidence to determine the location of a bounding box corner

- Generate two features corresponding to top detection & left detection
  - Merging the features after scaning to each direction
  - Find the point where top-left having maximum value





CornerNet Training



1. Smoothing ground truth
   - Allow the neighbor to produce active heatmap, but with gaussian-based penalty
2. Use "push" loss to increase the similarity of embedding with the corners of the same object, and use "push" loss to decrease the similarity of the corners of the different object

$$L_{pull} = \frac{1}{N}\sum_{k=1}^{N}\left[(e_{t_k} - e_k)^2 + (e_{b_k} - e_k)^2\right],$$

$$L_{push} = \frac{1}{N(N-1)}\sum_{k=1}^{N}\sum_{\substack{j=1 \\ j\neq k}}^{N}\max\left(0, \Delta - |e_k - e_j|\right),$$

CornerNet beats other one-stage detectors and shows competitive results to two-stage detectors

Method	Backbone	AP	AP^{50}	AP^{75}
Two-stage detectors				
DeNet (Tychsen-Smith and Petersson, 2017a)	ResNet-101	33.8	53.4	36.1
CoupleNet (Zhu et al., 2017)	ResNet-101	34.4	54.8	37.2
Faster R-CNN by G-RMI (Huang et al., 2017)	Inception-ResNet-v2 (Szegedy et al., 2017)	34.7	55.5	36.7
Faster R-CNN+++ (He et al., 2016)	ResNet-101	34.9	55.7	37.4
Faster R-CNN w/ FPN (Lin et al., 2016)	ResNet-101	36.2	59.1	39.0
Faster R-CNN w/ TDM (Shrivastava et al., 2016)	Inception-ResNet-v2	36.8	57.7	39.2
D-FCN (Dai et al., 2017)	Aligned-Inception-ResNet	37.5	58.0	-
Regionlets (Xu et al., 2017)	ResNet-101	39.3	59.8	-
Mask R-CNN (He et al., 2017)	ResNeXt-101	39.8	62.3	43.4
Soft-NMS (Bodla et al., 2017)	Aligned-Inception-ResNet	40.9	62.8	-
LH R-CNN (Li et al., 2017)	ResNet-101	41.5	-	-
Fitness-NMS (Tychsen-Smith and Petersson, 2017b)	ResNet-101	41.8	60.9	44.9
Cascade R-CNN (Cai and Vasconcelos, 2017)	ResNet-101	42.8	62.1	46.3
D-RFCN + SNIP (Singh and Davis, 2017)	DPN-98 (Chen et al., 2017)	45.7	67.3	51.1
One-stage detectors				
YOLOv2 (Redmon and Farhadi, 2016)	DarkNet-19	21.6	44.0	19.2
DSOD300 (Shen et al., 2017a)	DS/64-192-48-1	29.3	47.3	30.6
GRP-DSOD320 (Shen et al., 2017b)	DS/64-192-48-1	30.0	47.9	31.8
SSD513 (Liu et al., 2016)	ResNet-101	31.2	50.4	33.3
DSSD513 (Fu et al., 2017)	ResNet-101	33.2	53.3	35.2
RefineDet512 (single scale) (Zhang et al., 2017)	ResNet-101	36.4	57.5	39.5
RetinaNet800 (Lin et al., 2017)	ResNet-101	39.1	59.1	42.3
RefineDet512 (multi scale) (Zhang et al., 2017)	ResNet-101	41.8	62.9	45.7
CornerNet511 (single scale)	Hourglass-104	40.6	56.4	43.2
CornerNet511 (multi scale)	Hourglass-104	42.2	57.8	45.2

Reading Assignment

- Two-step methods
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
  - Mask R-CNN
  - Light-Head R-CNN
  - Pyramid Networks / FPN
  - Oriented R-CNN
  - G-RCNN
- Single shot methods
  - YOLO
  - SSD
  - YOLO2
  - DSSD
  - RetinaNet
  - YOLOv3-v8
  - M2Det
  - Sparse R-CNN
- Anchor-free methods
  - CornerNet
  - CenterNet
  - FCOS
  - CenterNet++