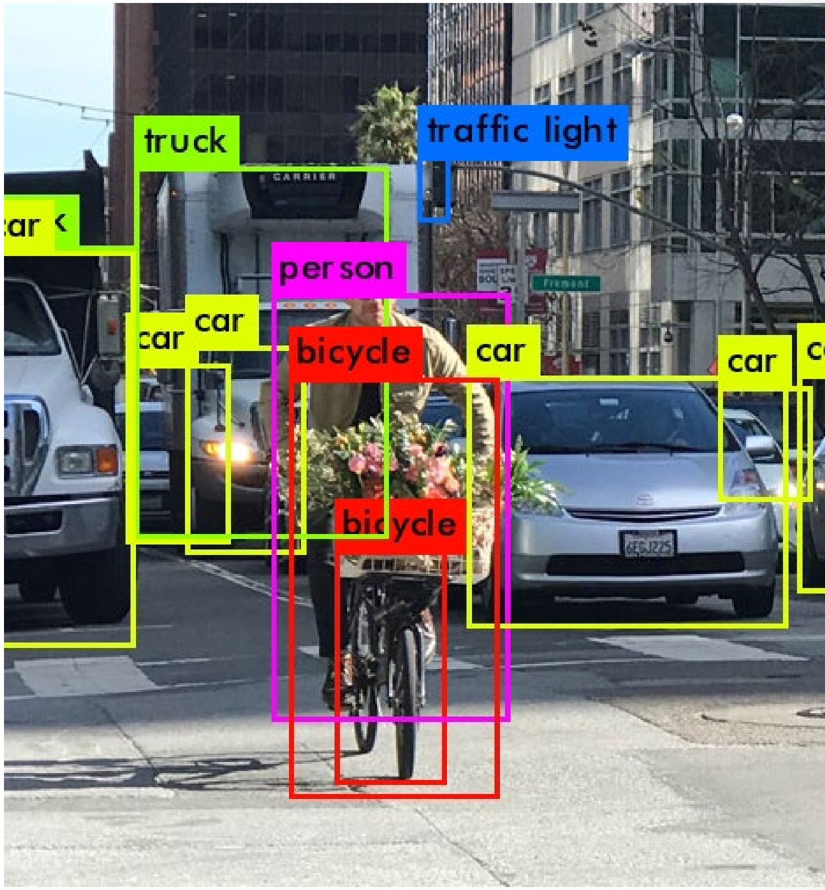
OBJECT DETECTION

Chih-Chung Hsu (許志仲)
Assistant Professor
ACVLab, Institute of Data Science
National Cheng Kung University
https://cchsu.info



So far: Image Classification

This image is CC0 public domain

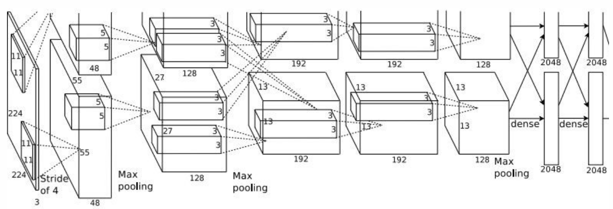


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Vector: 4096

Fully-Connected: 4096 to 1000

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

OBJECT DETECTION / LOCALIZATION

Object Detection

Classification

CAT

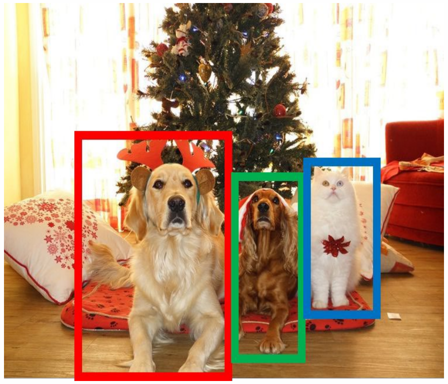
No spatial extent

Semantic Segmentation

GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection



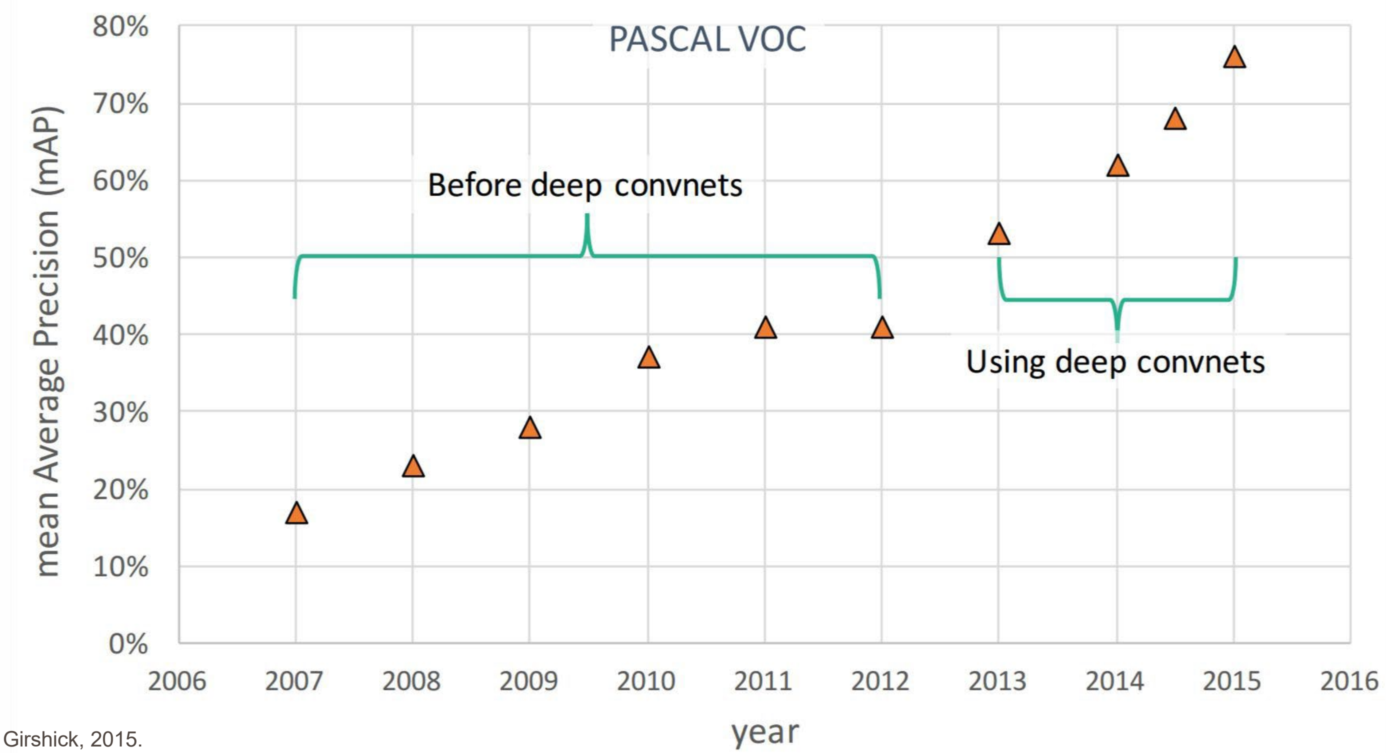
DOG, DOG, CAT

Instance Segmentation

DOG, DOG, CAT

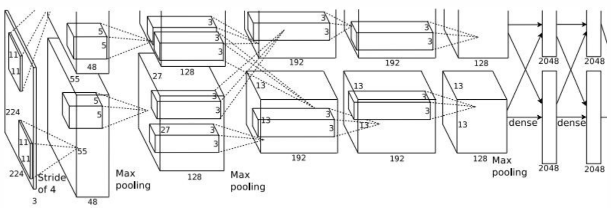
Multiple Object

Object Detection: Impact of Deep Learning



Object Detection: Single Object (Classification + Localization)

This image is CC0 public domain



Vector: 4096

Fully Connected: 4096 to 1000

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

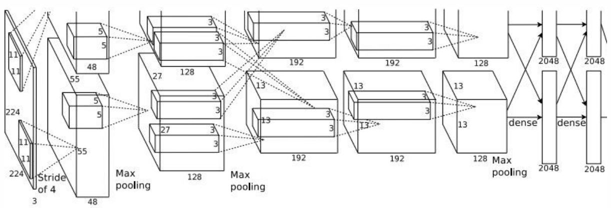
Fully Connected: 4096 to 4

Box Coordinates
(x, y, w, h)

Treat localization as a regression problem!

Object Detection: Single Object (Classification + Localization)

This image is CC0 public domain



Vector: 4096

Fully Connected: 4096 to 1000

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

Correct label: Cat

Softmax Loss

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

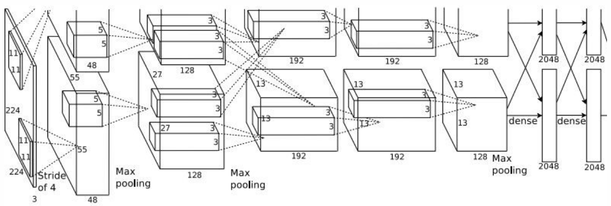
L2 Loss

Correct box: (x’, y’, w’, h’)

Treat localization as a regression problem!

Object Detection: Single Object (Classification + Localization)

[This image](#) is [CC0 public domain](#)



Vector: 4096

Fully Connected: 4096 to 1000

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

Correct label: Cat

Softmax Loss

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

L2 Loss

Correct box: (x’, y’, w’, h’)

Multitask Loss

\+ **Loss**

Treat localization as a regression problem!

Object Detection: Single Object (Classification + Localization)

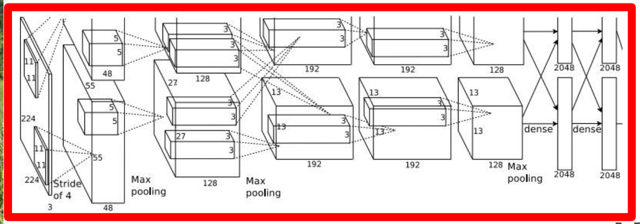
Correct label:
Cat

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

Fully Connected:
4096 to 1000

Softmax Loss

This image is CC0 public domain



Often pretrained on ImageNet (Transfer learning)

Vector: 4096

Fully Connected:
4096 to 4

Box Coordinates
(x, y, w, h)

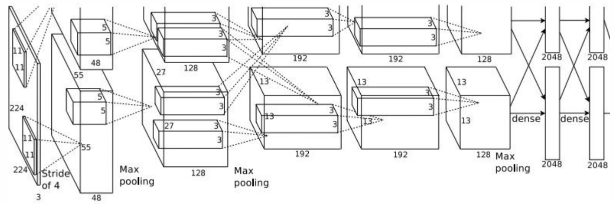
L2 Loss

+ **Loss**

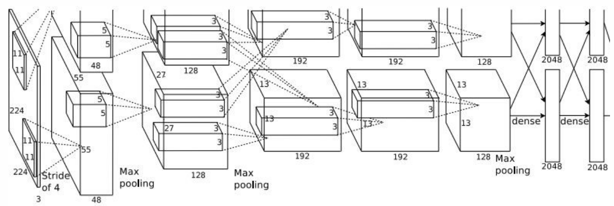
Correct box:
(x’, y’, w’, h’)

Treat localization as a regression problem!

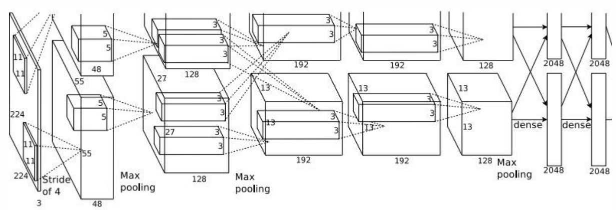
Object Detection: Multiple Objects



CAT: (x, y, w, h)

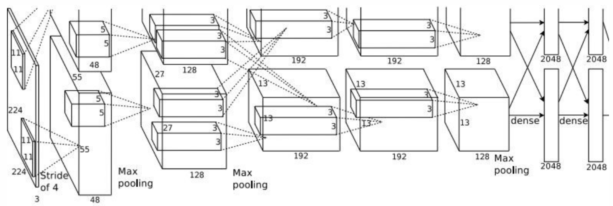


DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

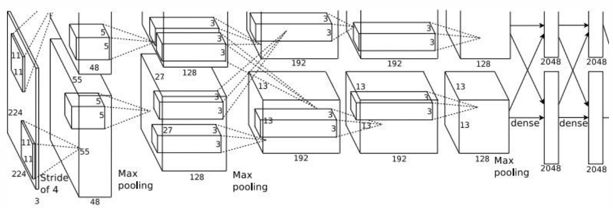


DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

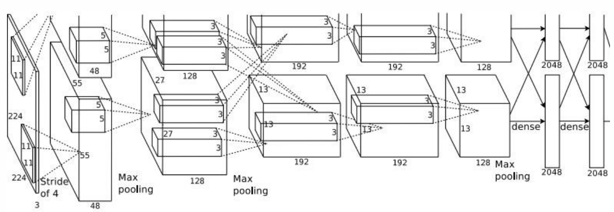
Each image needs many outputs!



CAT: (x, y, w, h) — 4 numbers



DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h) — 12 numbers



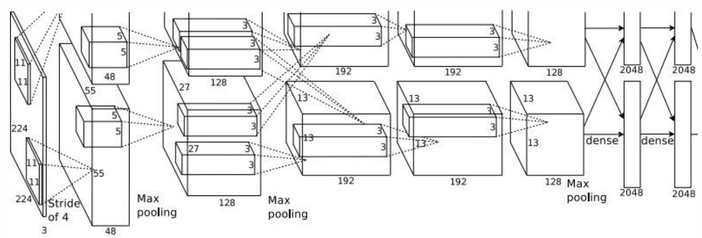
DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
….

Many numbers!

Object Detection: Multiple Objects

Object Detection: Multiple Objects

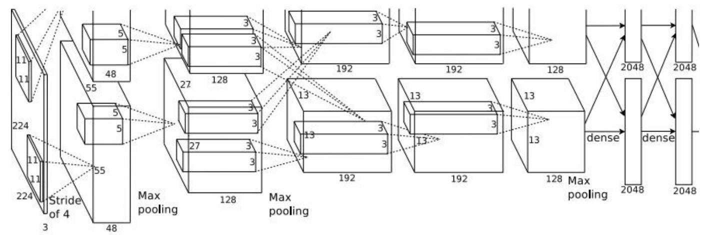
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? NO
Background? YES

Object Detection: Multiple Objects

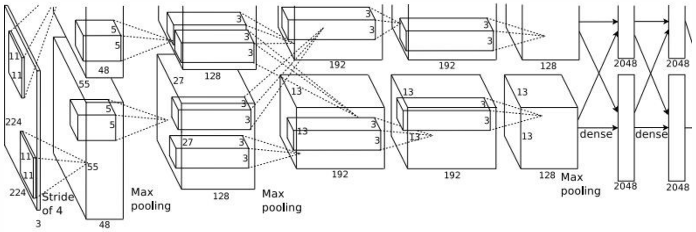
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Object Detection: Multiple Objects

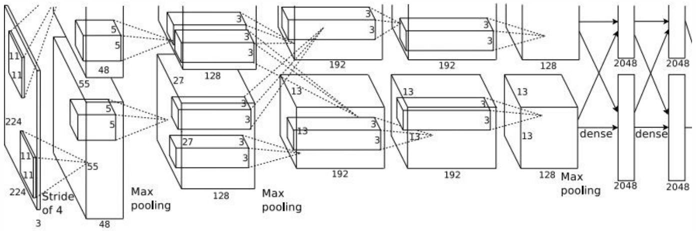
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

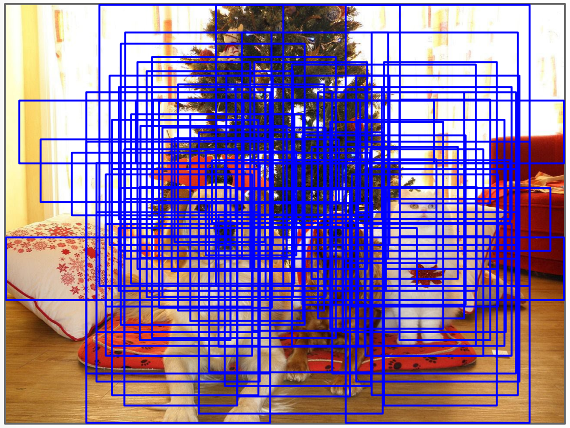
Object Detection: Multiple Objects

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

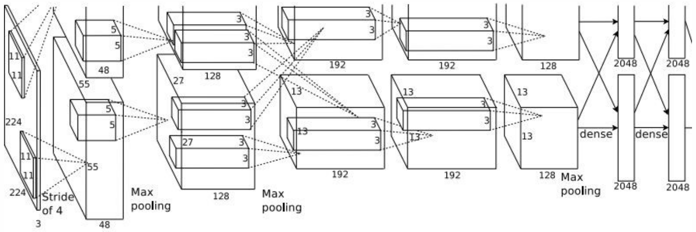


Dog? NO
Cat? YES
Background? NO

Object Detection: Multiple Objects



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

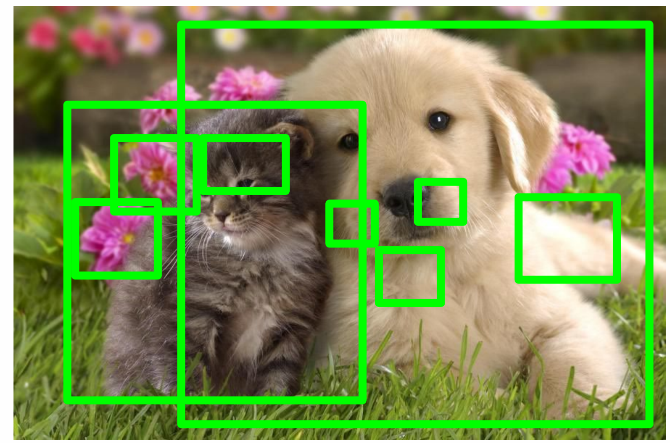


Dog? NO
Cat? YES
Background? NO

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

Region Proposals: Selective Search

- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU



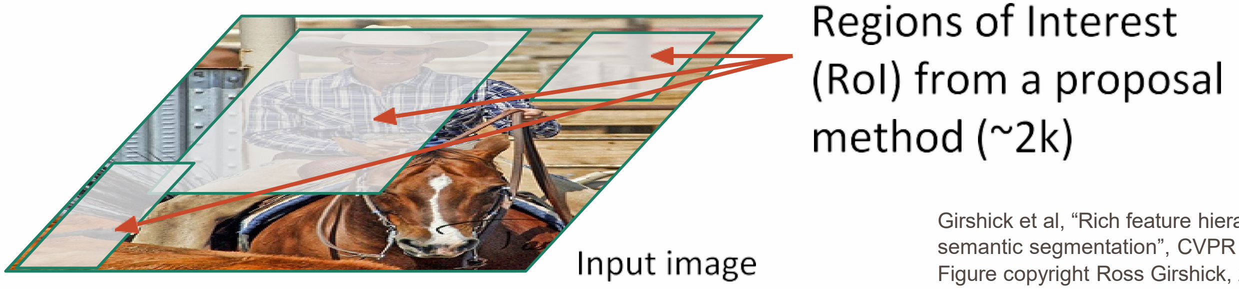
Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012
Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013
Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014
Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”,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

R-CNN

Input image

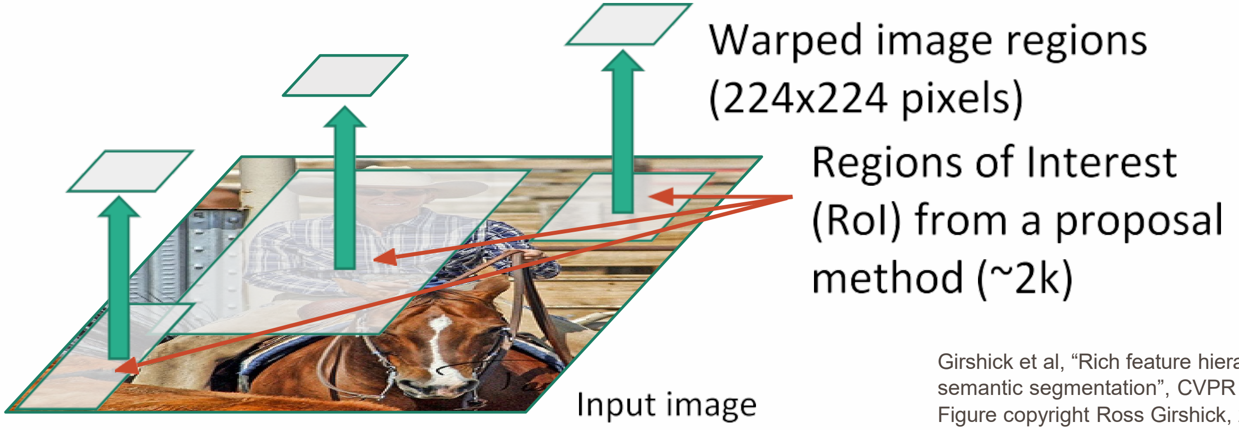
Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN



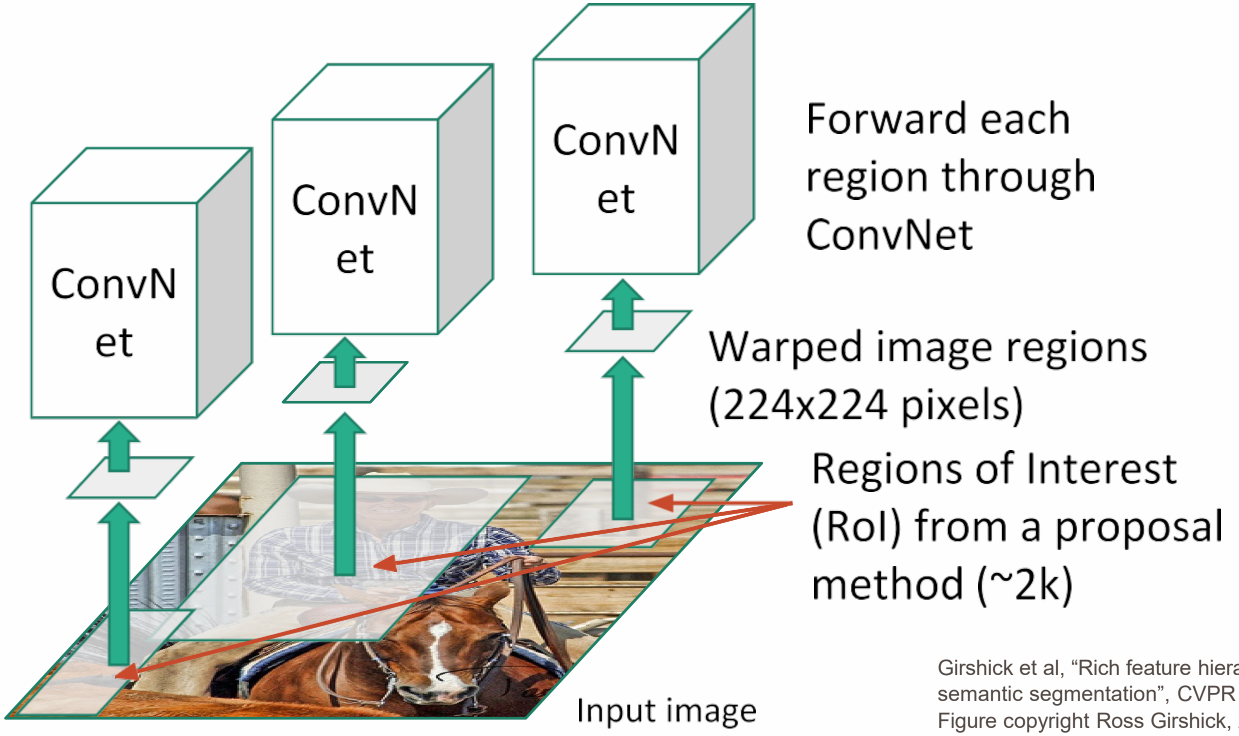
Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN



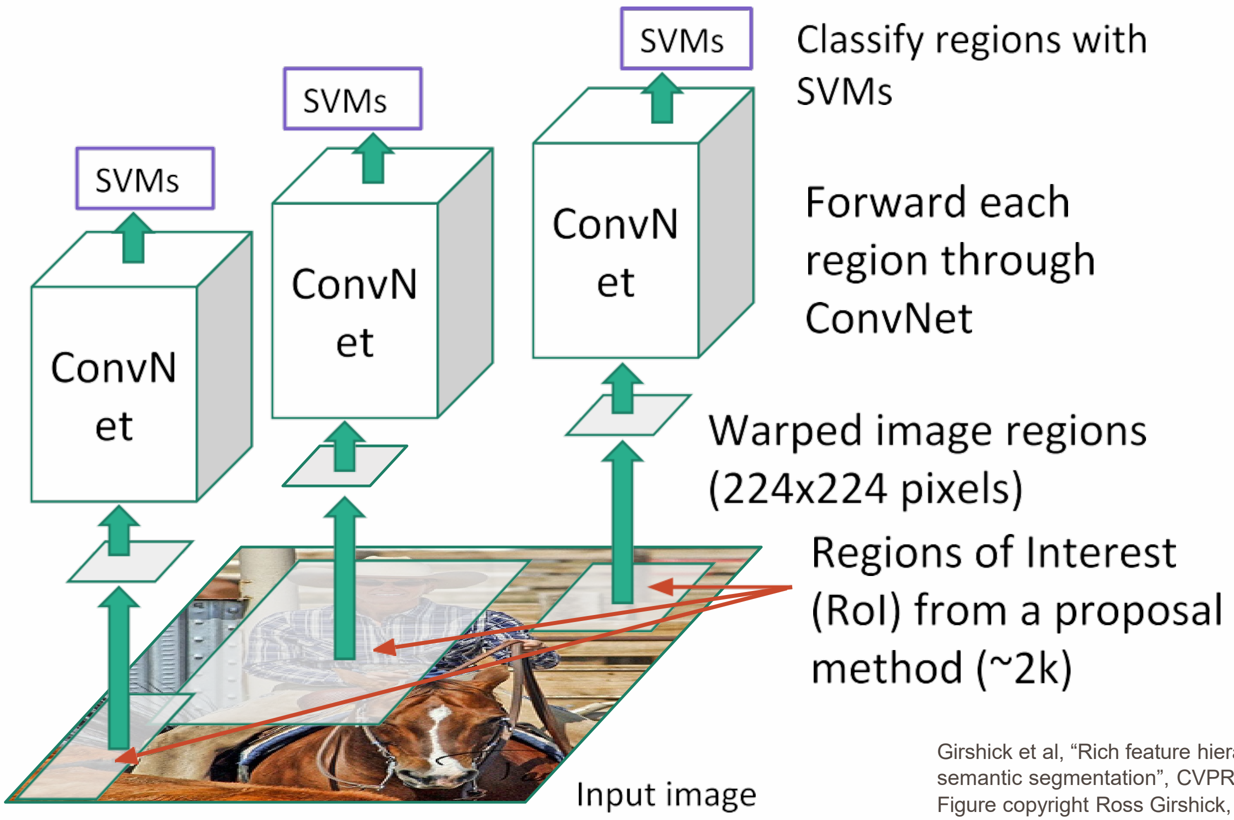
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN



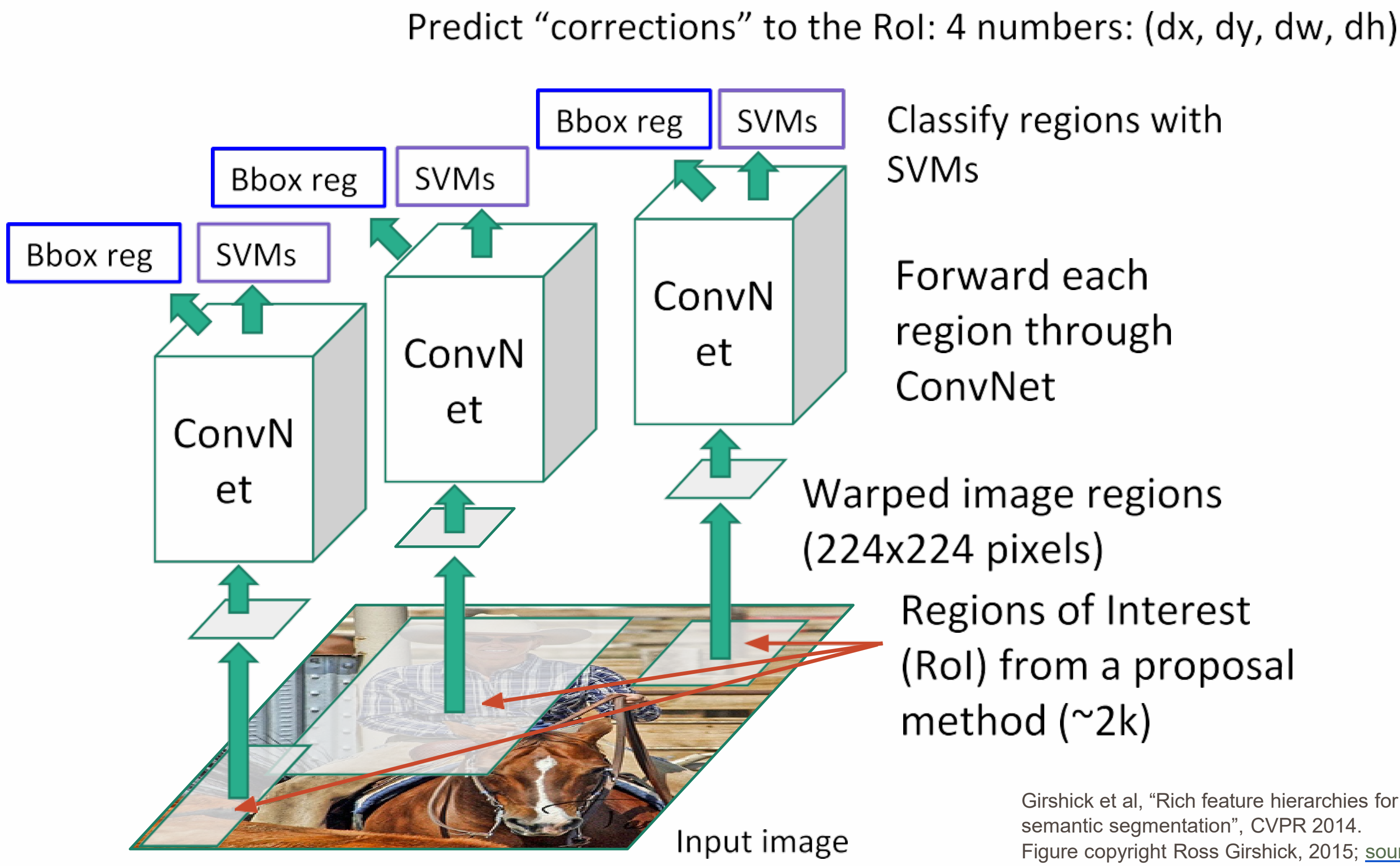
Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN



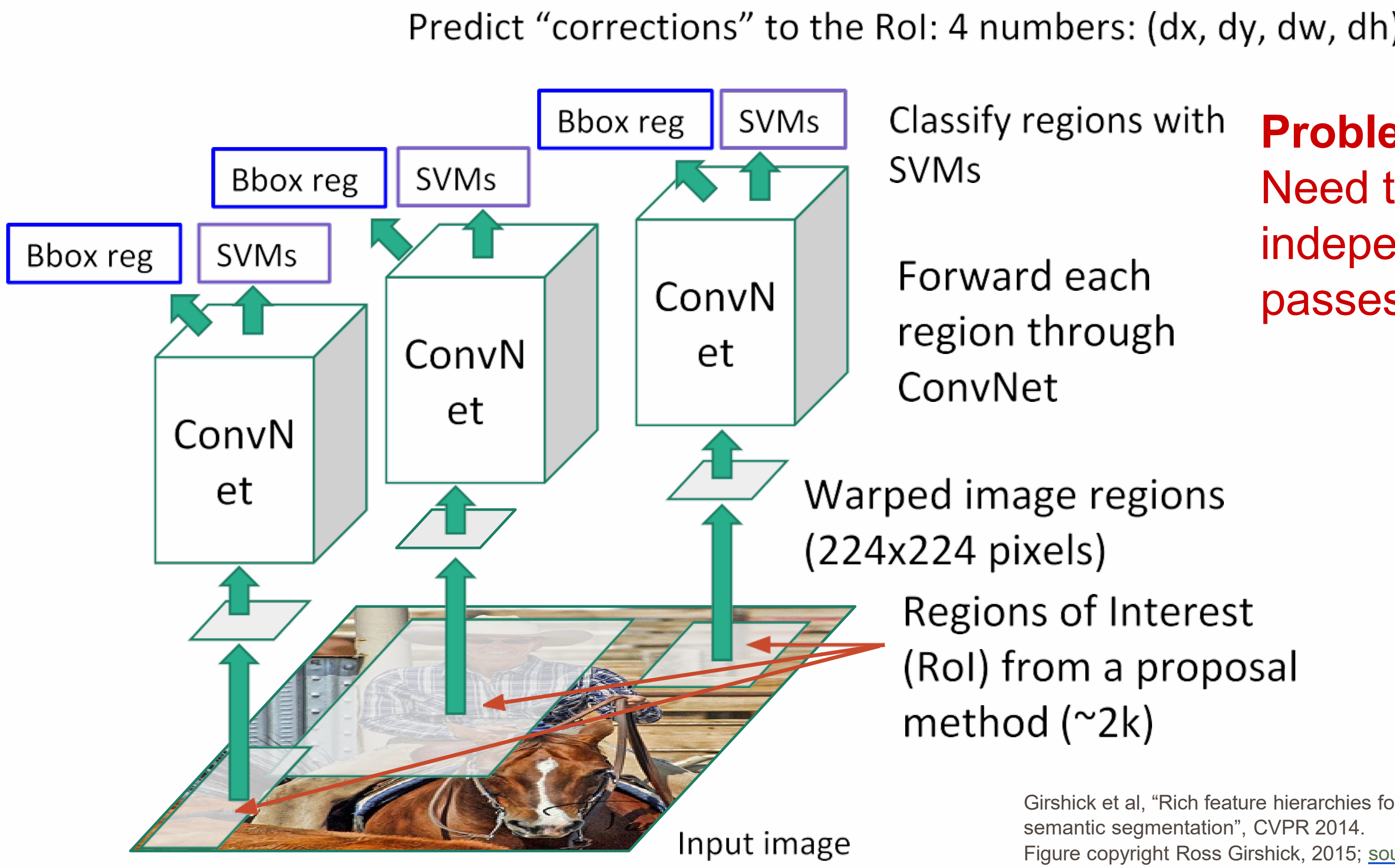
Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN



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

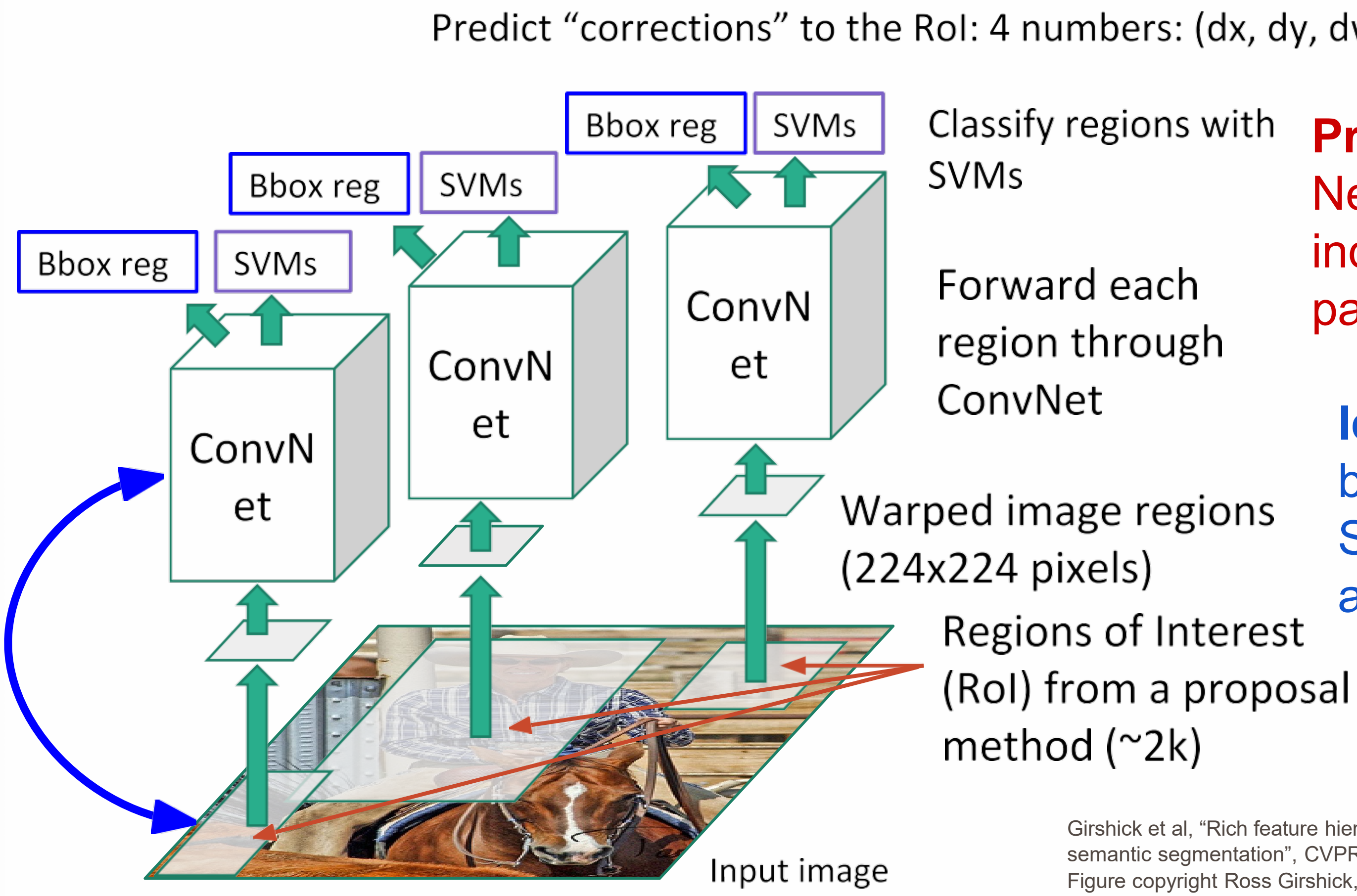
R-CNN



Problem: Very slow! Need to do ~2k independent forward passes for each image!

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

“Slow” R-CNN



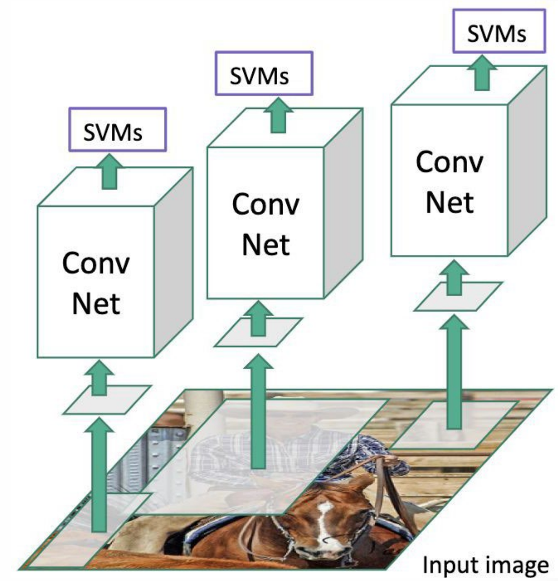
Problem: Very slow! Need to do ~2k independent forward passes for each image!

Idea: Process image before cropping! Swap convolution and cropping!

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

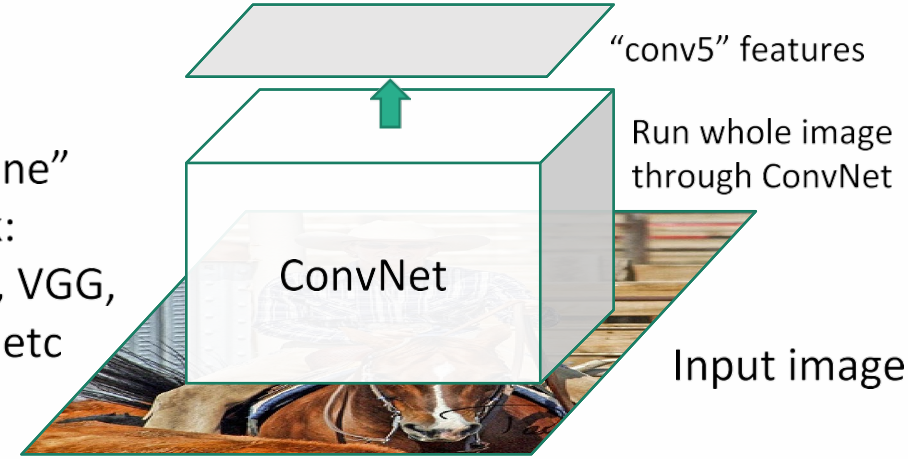
Fast R-CNN

"Slow" R-CNN



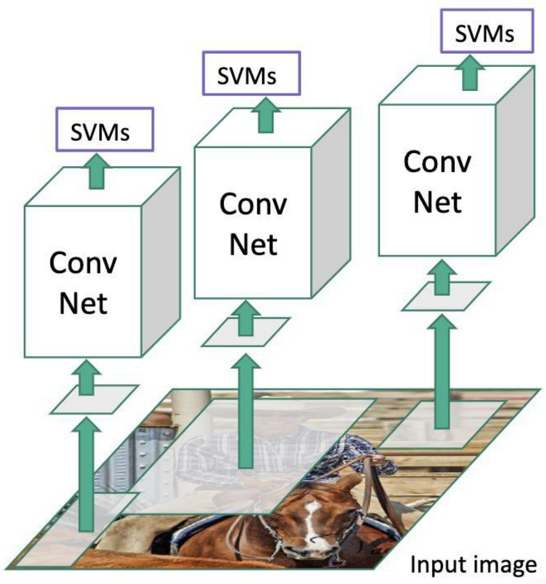
Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN

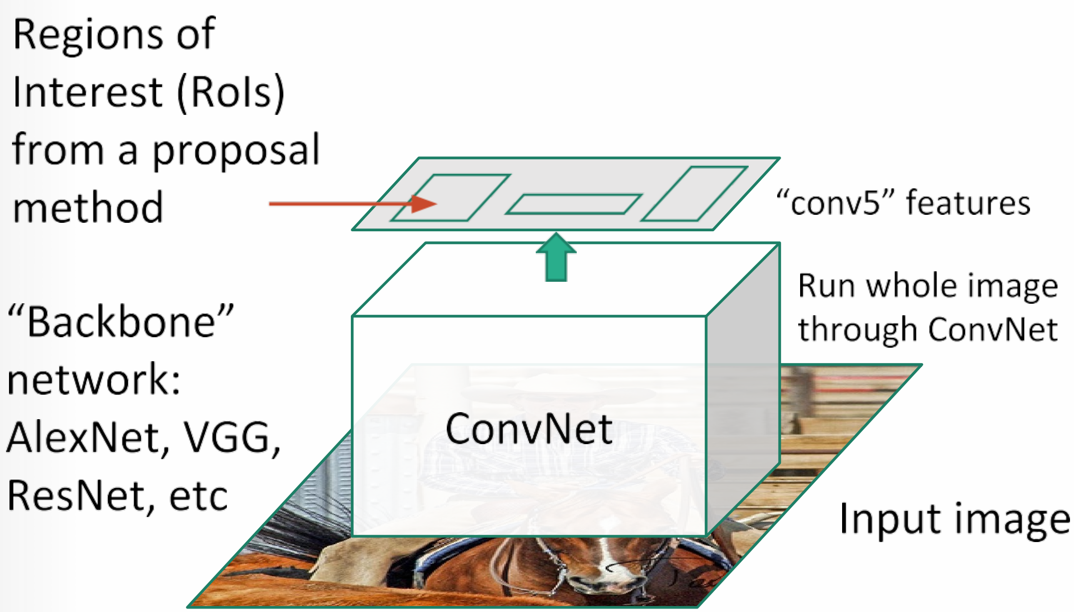


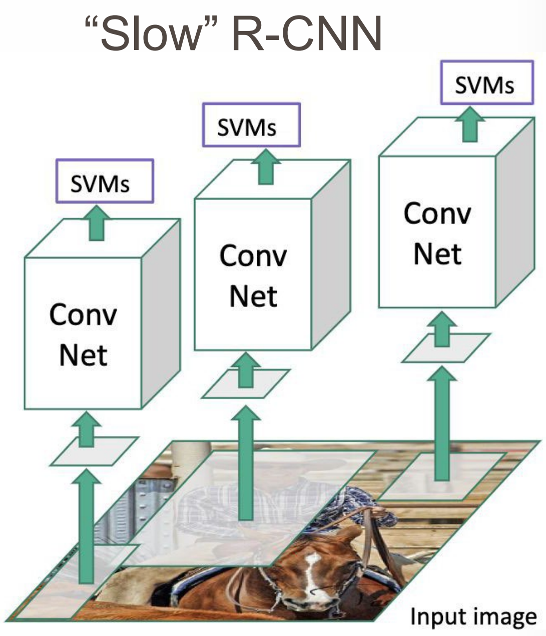
"Backbone" network: AlexNet, VGG, ResNet, etc

"Slow" R-CNN



Fast R-CNN



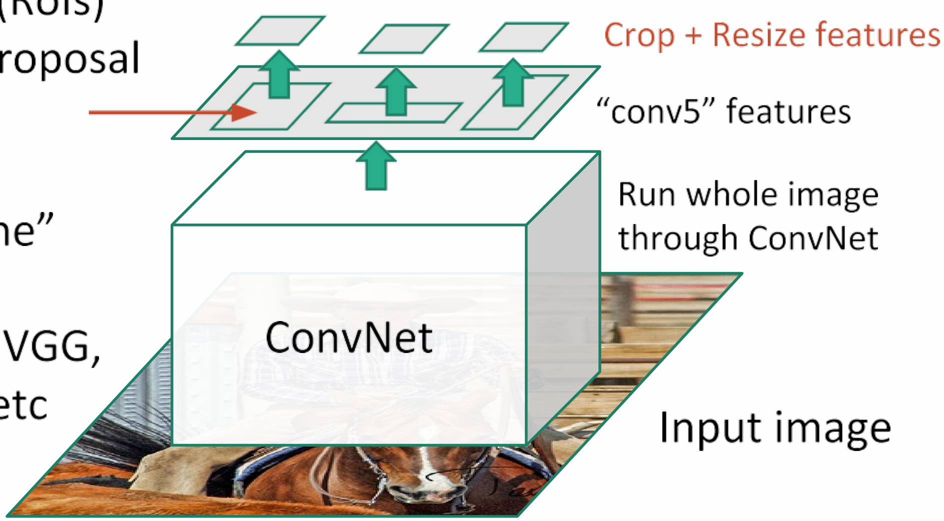


Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

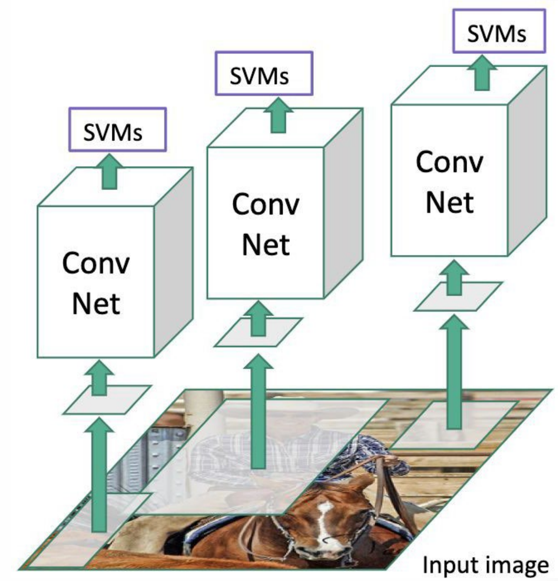
Fast R-CNN

Regions of Interest (RoIs) from a proposal method

"Backbone" network: AlexNet, VGG, ResNet, etc

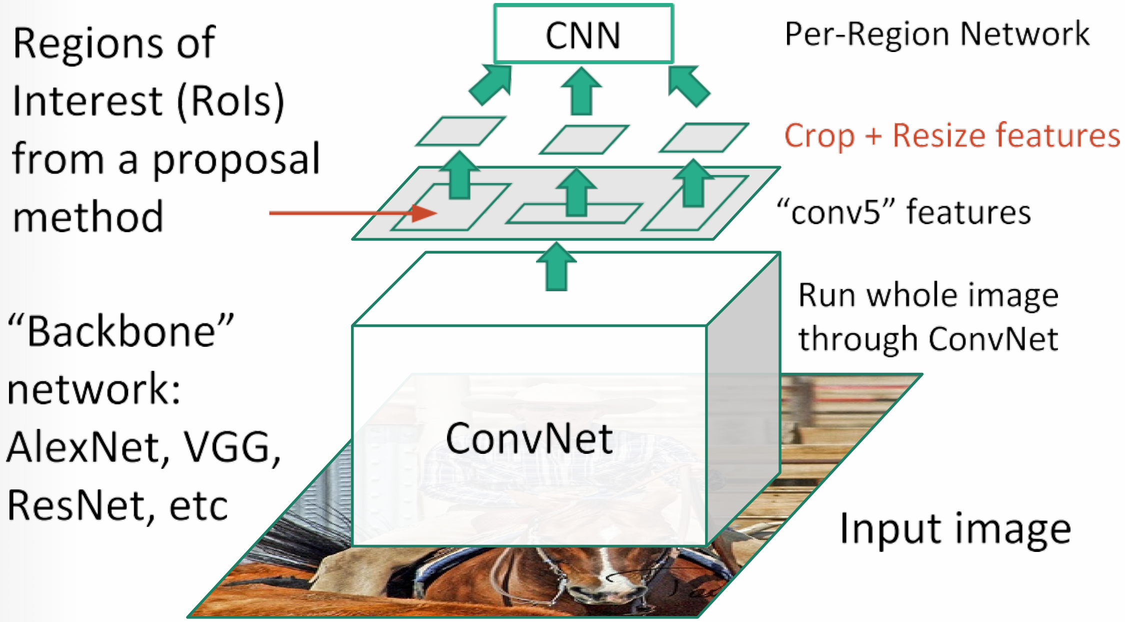


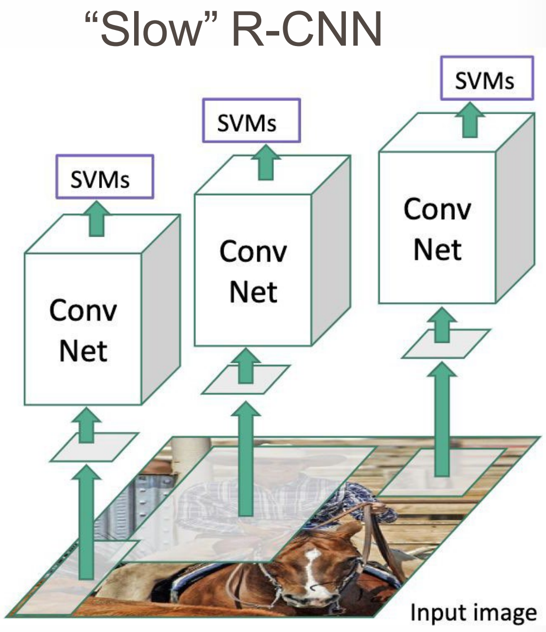
"Slow" R-CNN



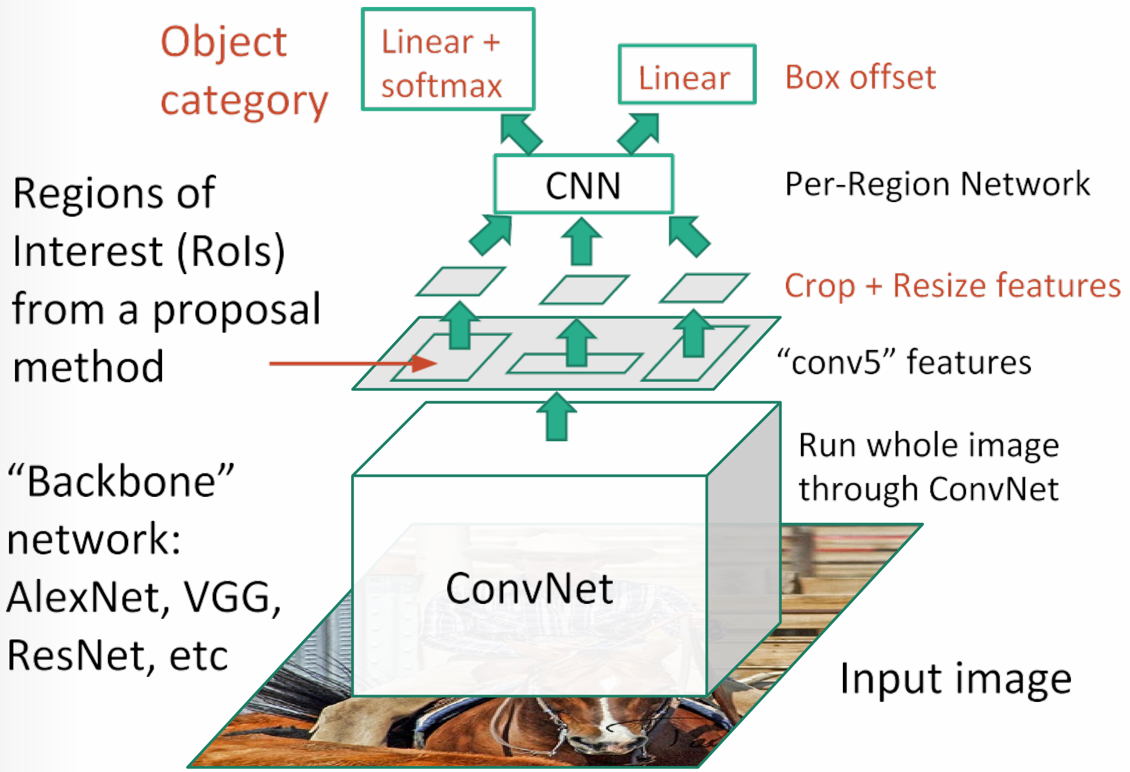
Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

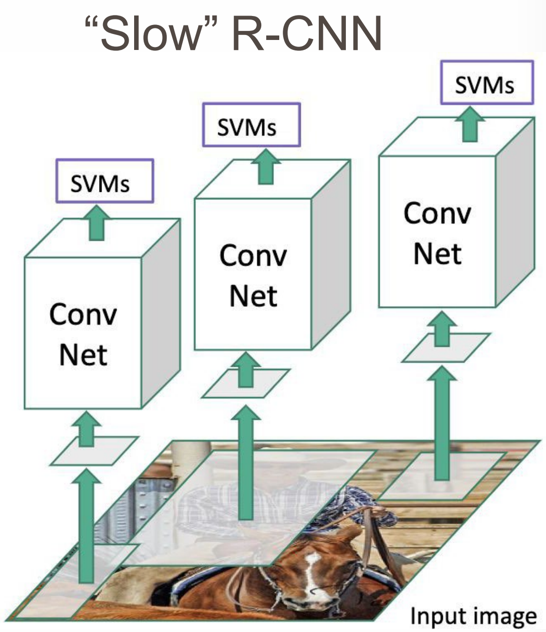
Fast R-CNN


Regions of Interest (RoIs) from a proposal method
"Backbone" network: AlexNet, VGG, ResNet, etc
CNN
Per-Region Network
Crop + Resize features
"conv5" features
Run whole image through ConvNet
ConvNet
Input image


"Slow" R-CNN
SVMs
SVMs
SVMs
Conv Net
Conv Net
Conv Net
Input image

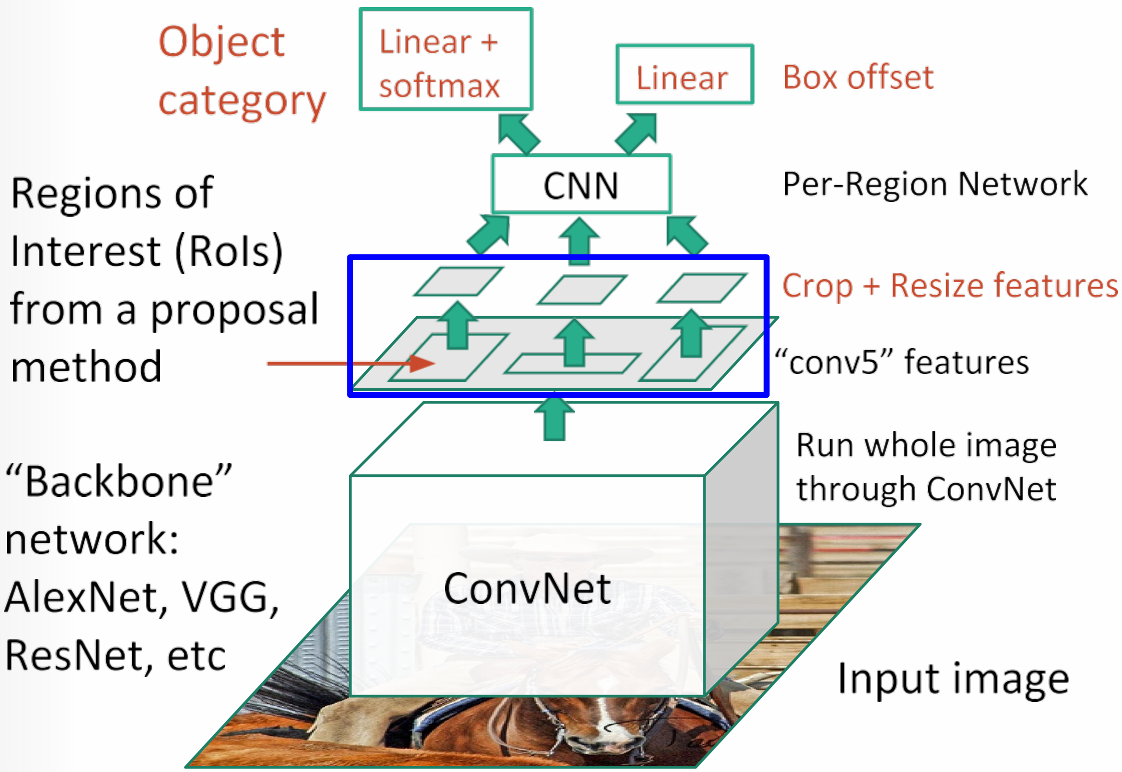
Fast R-CNN

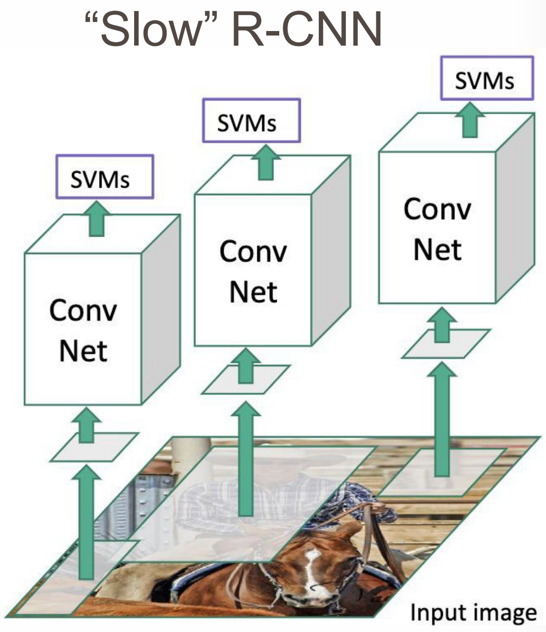




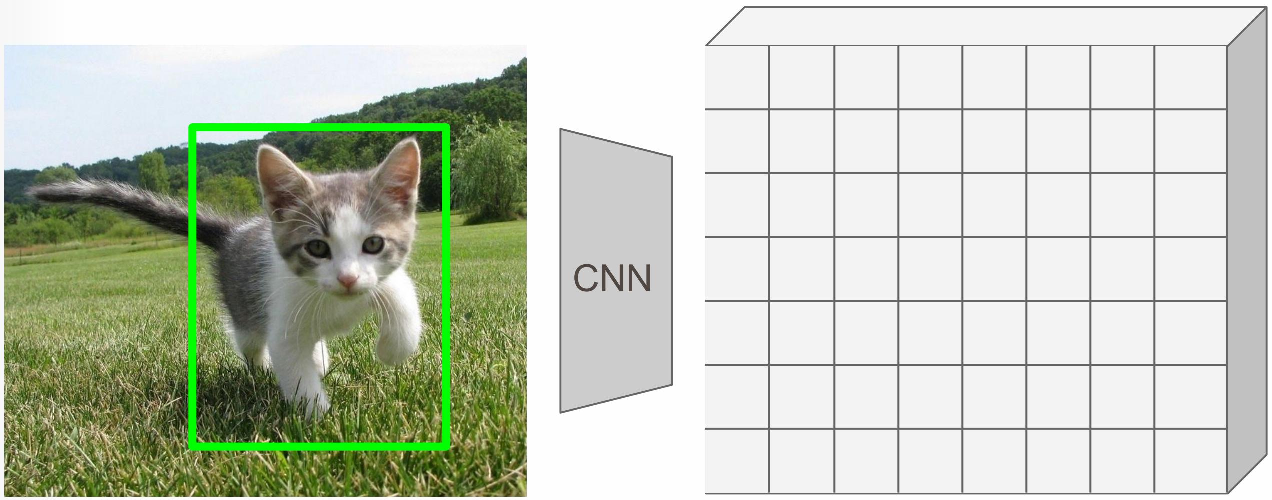
Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN


Object category
Linear + softmax
Linear
Box offset
CNN
Per-Region Network
Regions of Interest (RoIs) from a proposal method
Crop + Resize features
"conv5" features
Run whole image through ConvNet
"Backbone" network: AlexNet, VGG, ResNet, etc
ConvNet
Input image


"Slow" R-CNN
SVMs
SVMs
SVMs
Conv Net
Conv Net
Conv Net
Input image

Cropping Features: RoI Pool



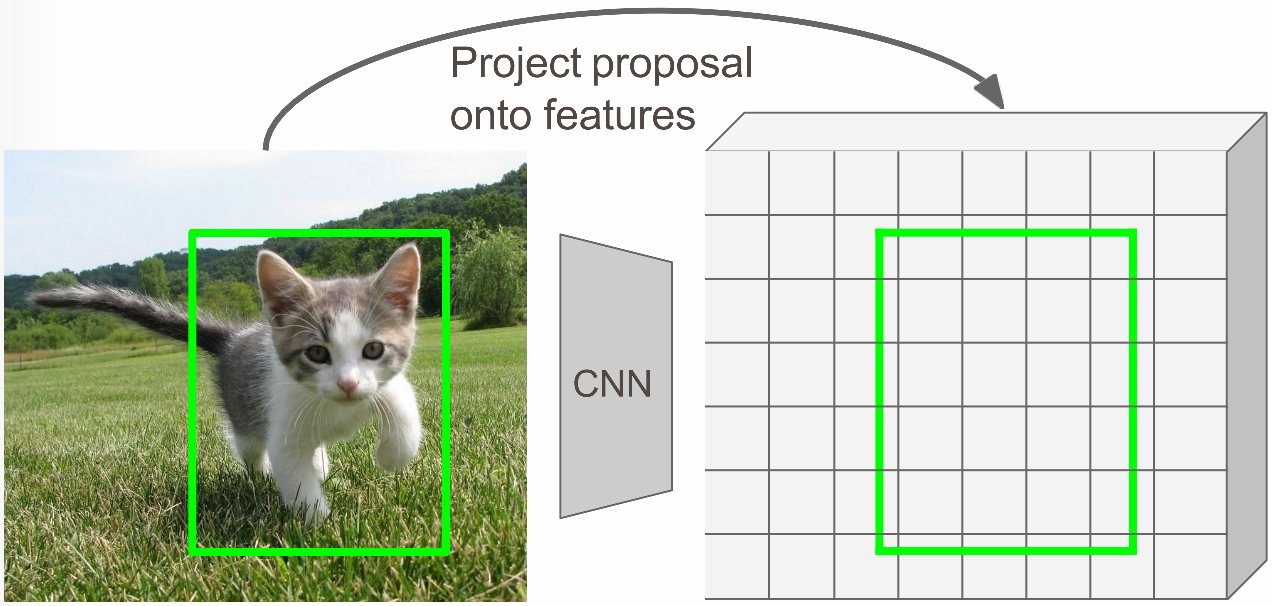
Input Image
(e.g. 640 x 480 x 3)

Image features
(e.g. 20 x 15 x 512)

Girshick, “Fast R-CNN”, ICCV 2015.

Girshick, “Fast R-CNN”, ICCV 2015.

Cropping Features: RoI Pool



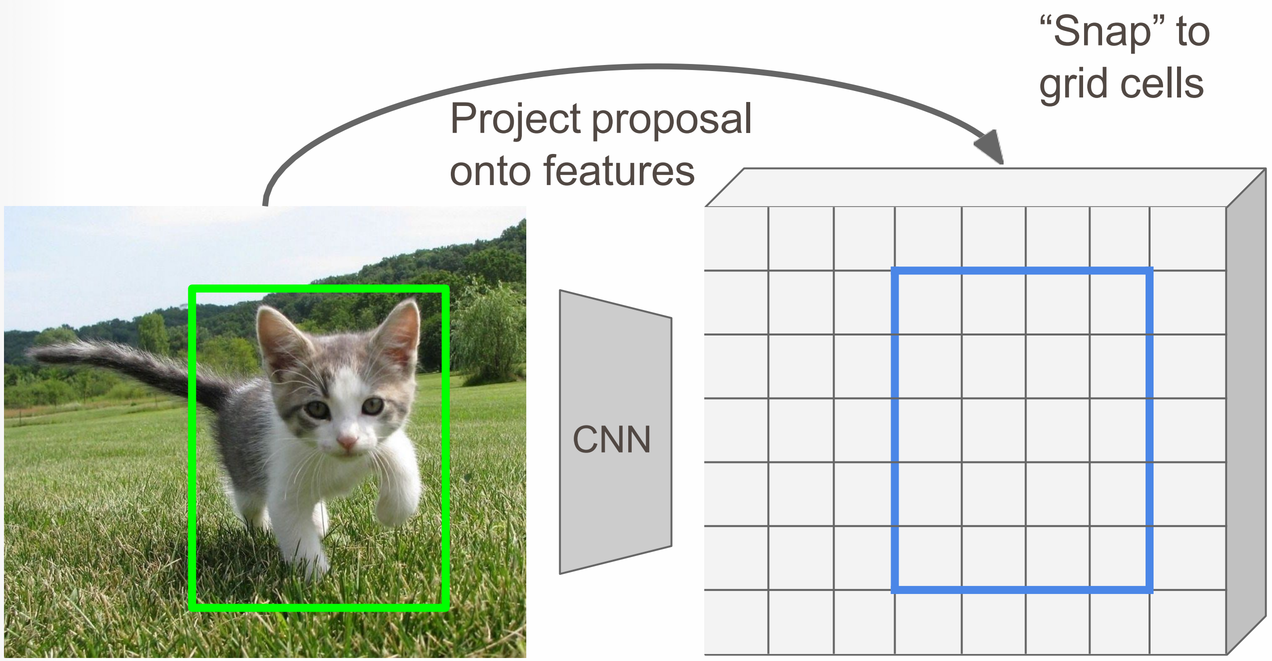
Input Image
(e.g. 640 x 480 x 3)

Image features
(e.g. 20 x 15 x 512)

Girshick, “Fast R-CNN”, ICCV 2015.

Girshick, “Fast R-CNN”, ICCV 2015.

Cropping Features: RoI Pool

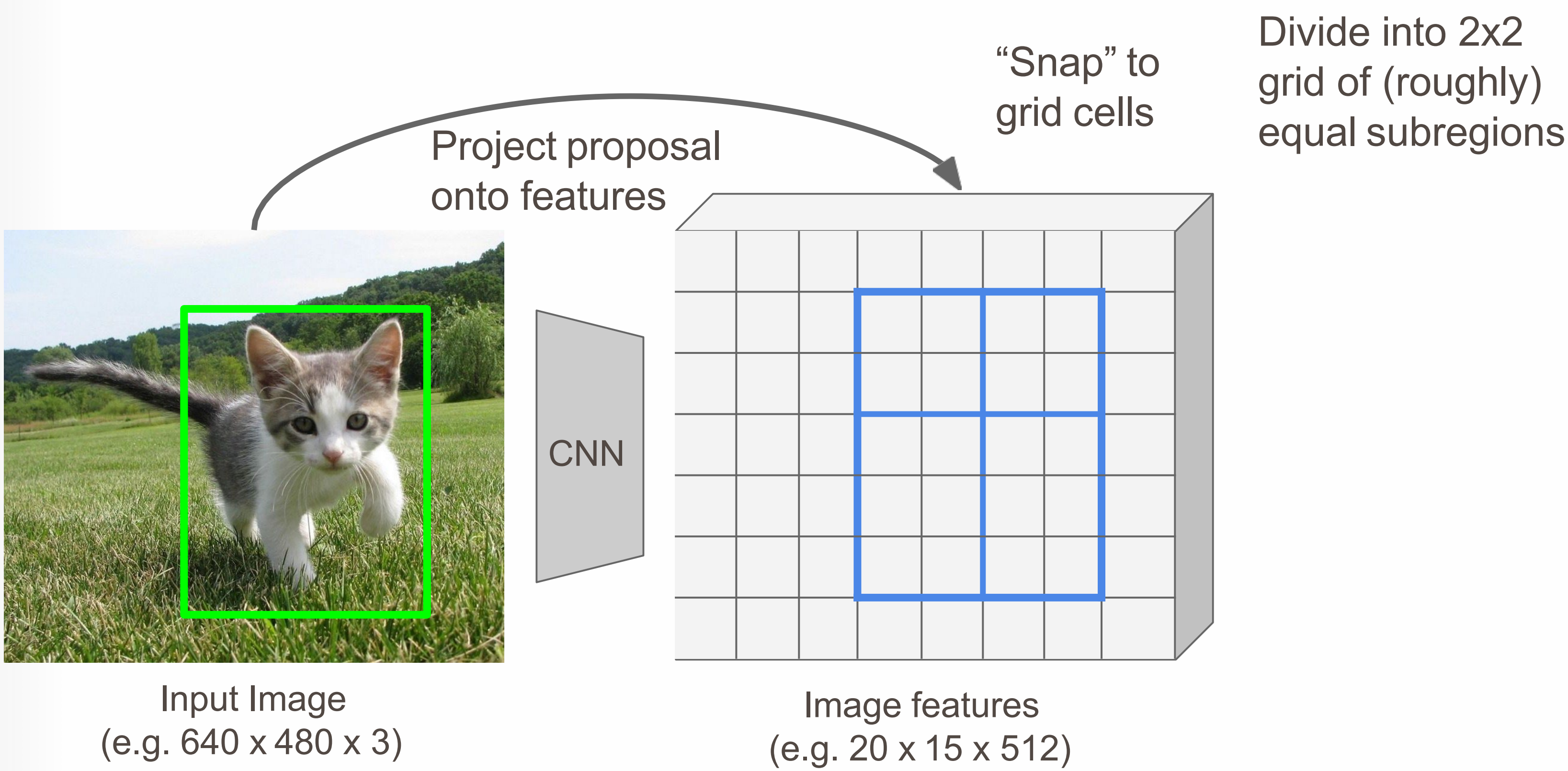


Input Image
(e.g. 640 x 480 x 3)

Image features
(e.g. 20 x 15 x 512)

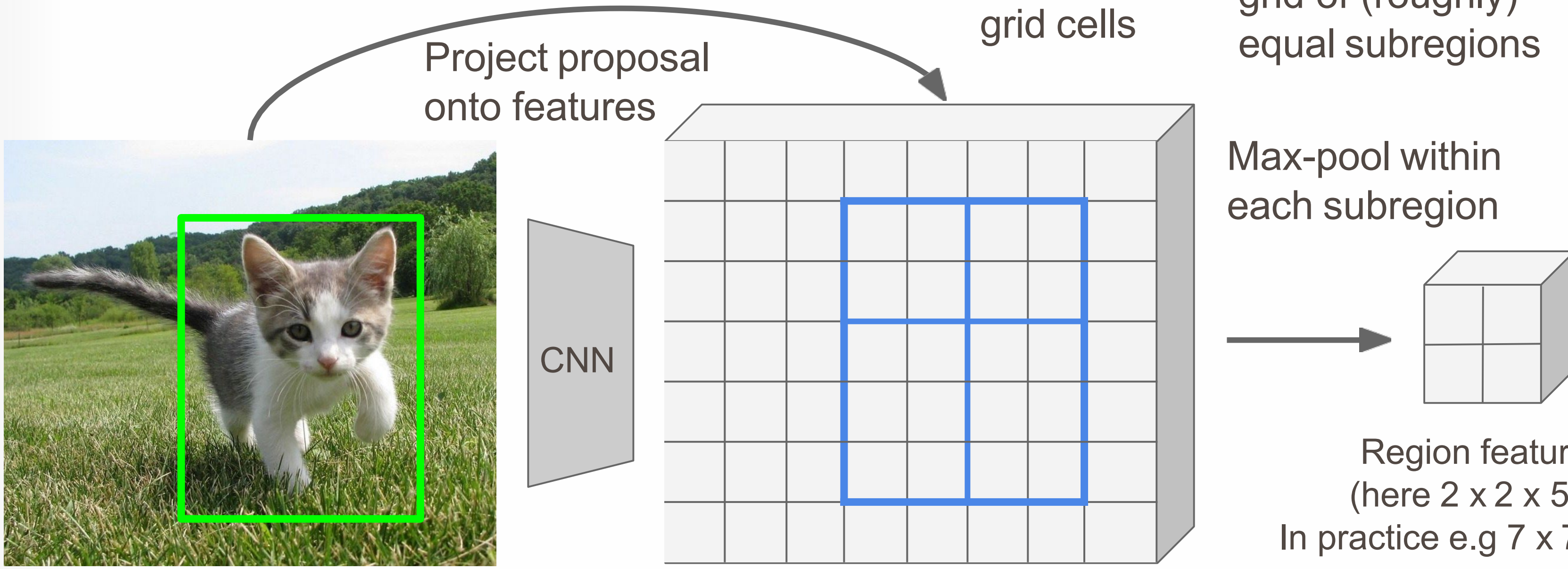
Girshick, "Fast R-CNN", ICCV 2015.

Cropping Features: RoI Pool



Girshick, "Fast R-CNN", ICCV 2015.

Cropping Features: RoI Pool



Input Image
(e.g. 640 x 480 x 3)

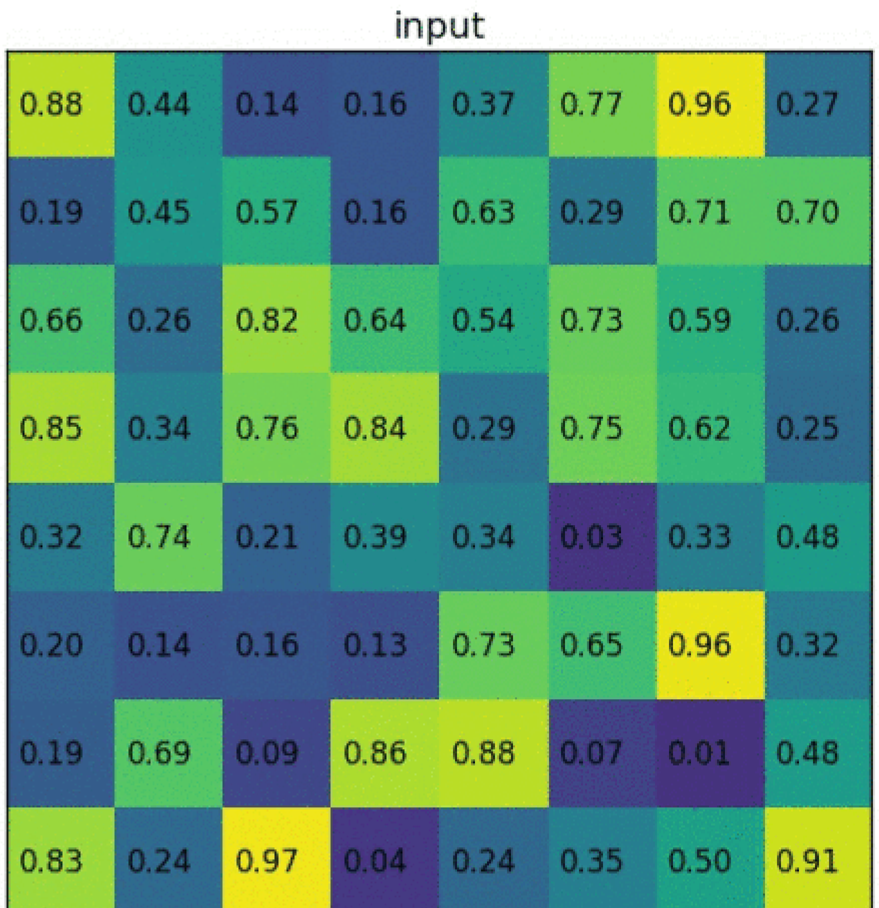
Image features
(e.g. 20 x 15 x 512)

Region features
(here 2 x 2 x 512;
In practice e.g 7 x 7 x 512)

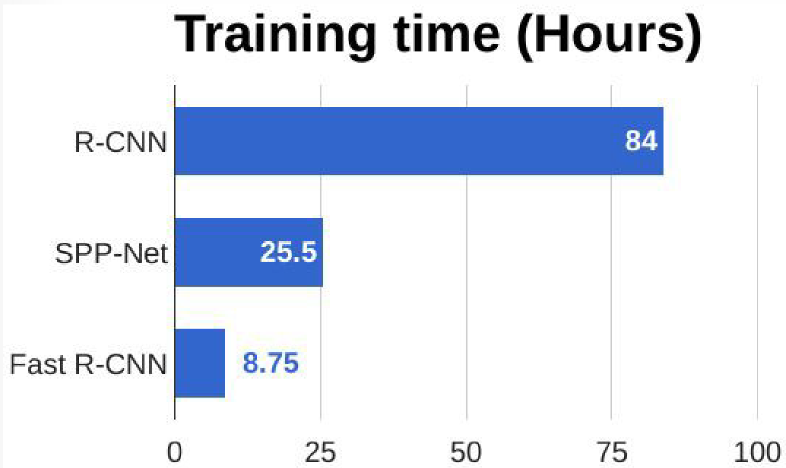
Region features always the same size even if input regions have different sizes!

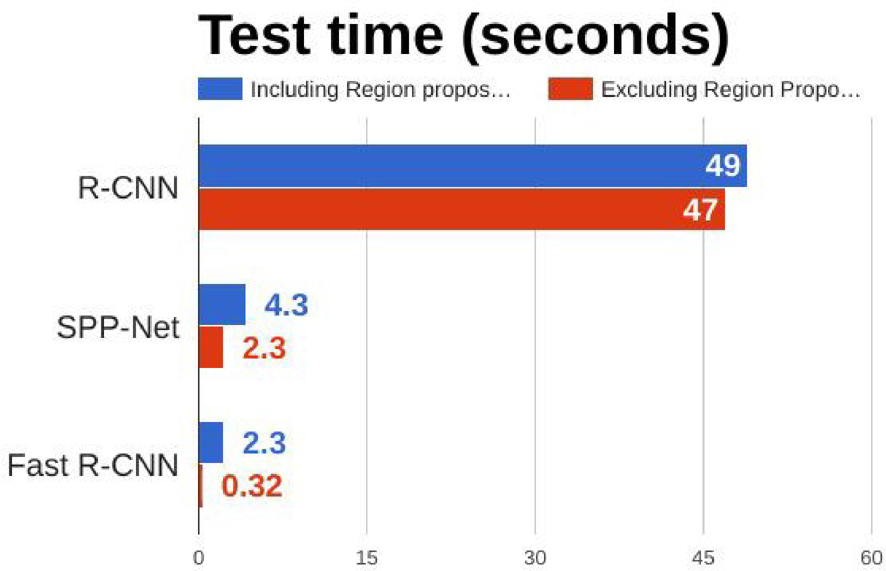
Girshick, “Fast R-CNN”, ICCV 2015.

ROI Pooling


input
0.88 0.44 0.14 0.16 0.37 0.77 0.96 0.27
0.19 0.45 0.57 0.16 0.63 0.29 0.71 0.70
0.66 0.26 0.82 0.64 0.54 0.73 0.59 0.26
0.85 0.34 0.76 0.84 0.29 0.75 0.62 0.25
0.32 0.74 0.21 0.39 0.34 0.03 0.33 0.48
0.20 0.14 0.16 0.13 0.73 0.65 0.96 0.32
0.19 0.69 0.09 0.86 0.88 0.07 0.01 0.48
0.83 0.24 0.97 0.04 0.24 0.35 0.50 0.91

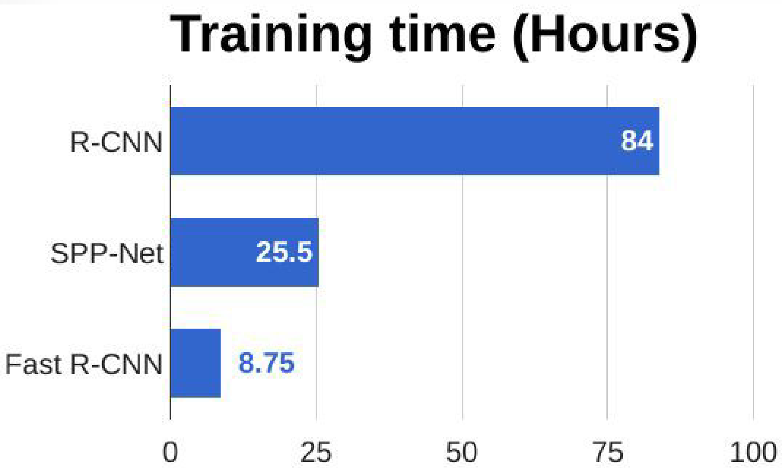
R-CNN vs Fast R-CNN

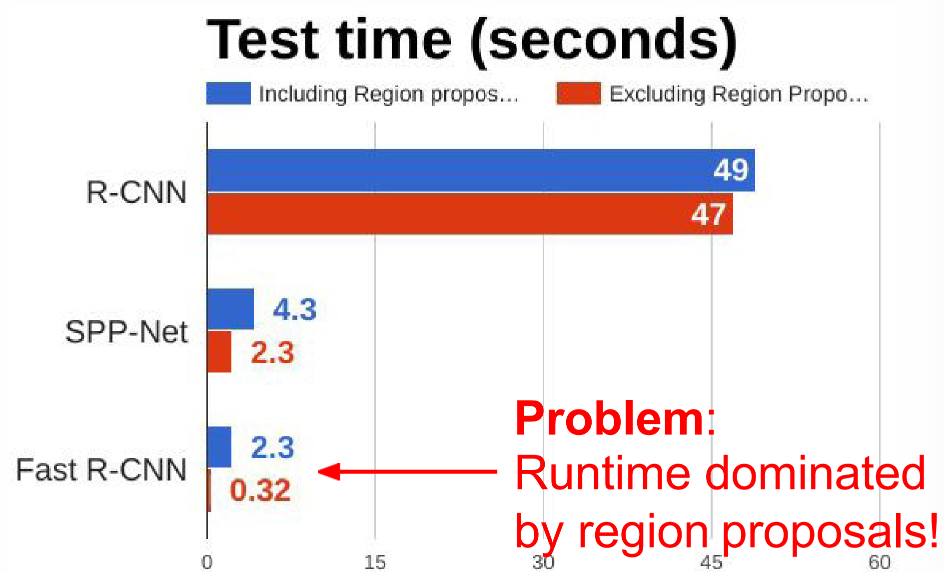




Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

R-CNN vs Fast R-CNN





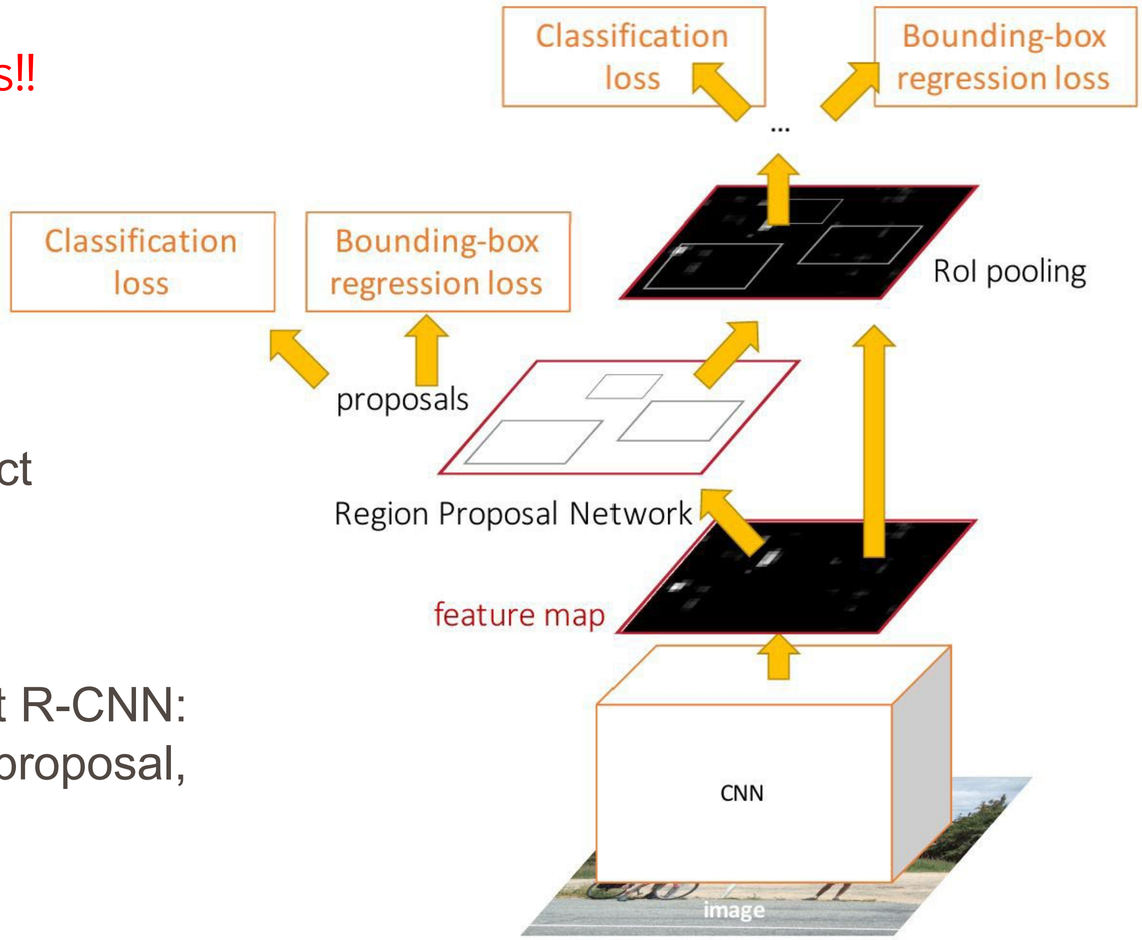
Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015

Faster R-CNN:

Make CNN do proposals!!

Insert **Region Proposal Network (RPN)** to predict proposals from features

Otherwise same as Fast R-CNN: Crop features for each proposal, classify each one



Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network

Input Image
(e.g. 640 x 480 x 3)

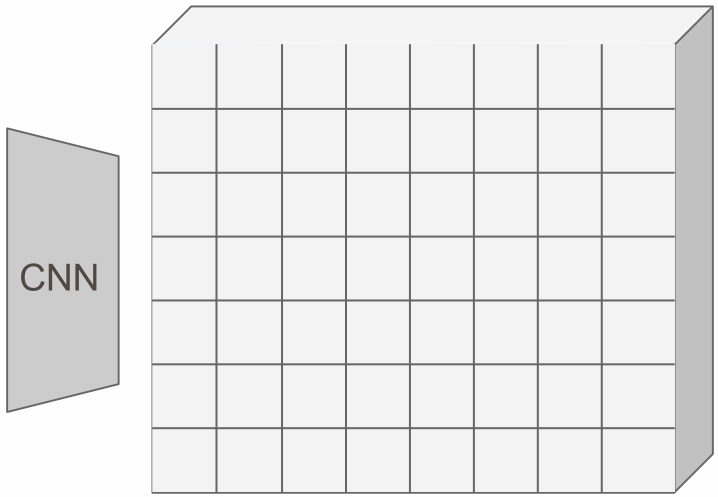
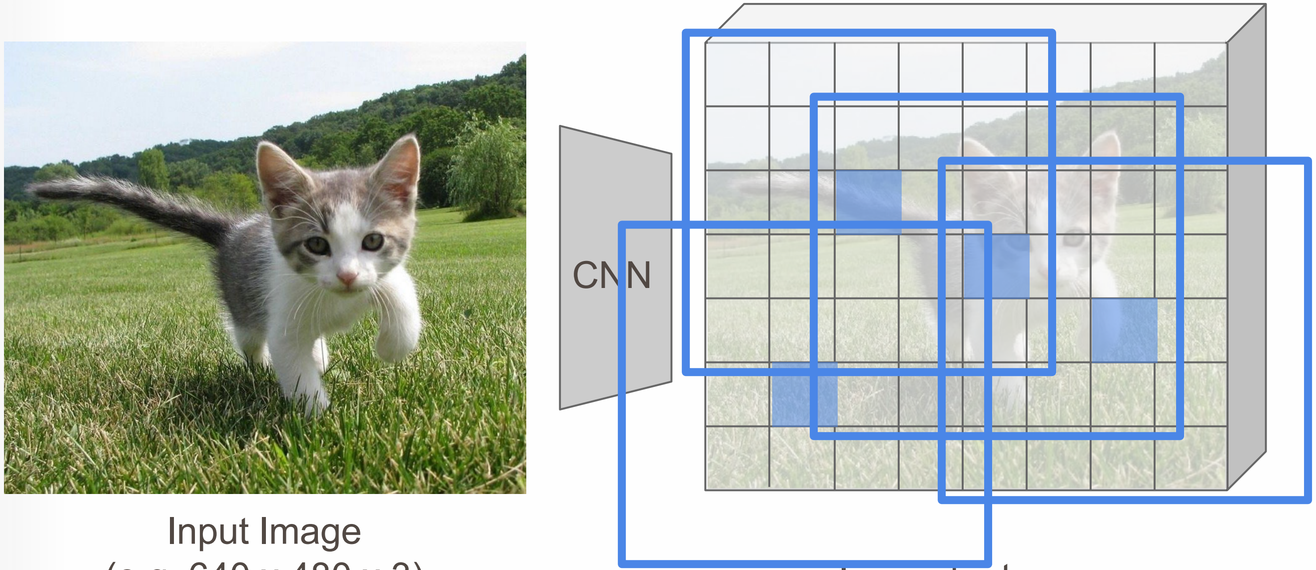


Image features
(e.g. 20 x 15 x 512)

Region Proposal Network

Imagine an **anchor box** of fixed size at each point in the feature map

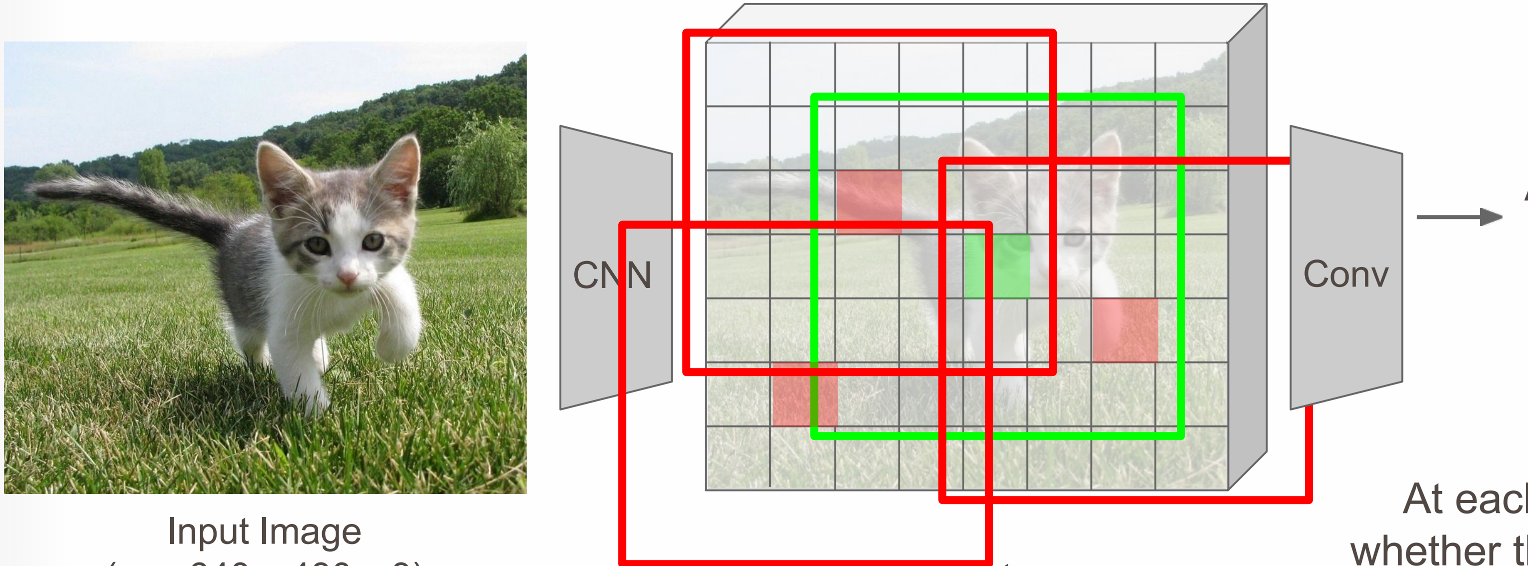


Input Image
(e.g. 640 x 480 x 3)

Image features
(e.g. 20 x 15 x 512)

Region Proposal Network

Imagine an **anchor box** of fixed size at each point in the feature map



Input Image
(e.g. 640 x 480 x 3)

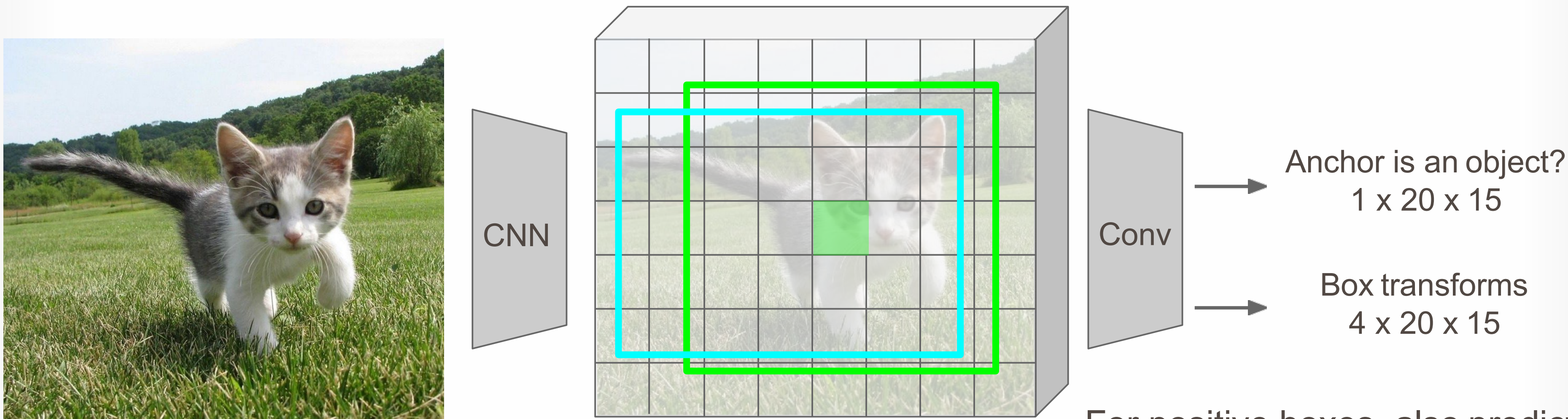
Image features
(e.g. 20 x 15 x 512)

Anchor is an object?
1 x 20 x 15

At each point, predict whether the corresponding anchor contains an object (per-pixel logistic regression)

Region Proposal Network

Imagine an **anchor box** of fixed size at each point in the feature map



Input Image
(e.g. 640 x 480 x 3)

Image features
(e.g. 20 x 15 x 512)

For positive boxes, also predict a transformation from the anchor to the ground-truth box (regress 4 numbers per pixel)

In practice use K different anchor boxes of different size / scale at each point

Region Proposal Network

Input Image
(e.g. 640 x 480 x 3)

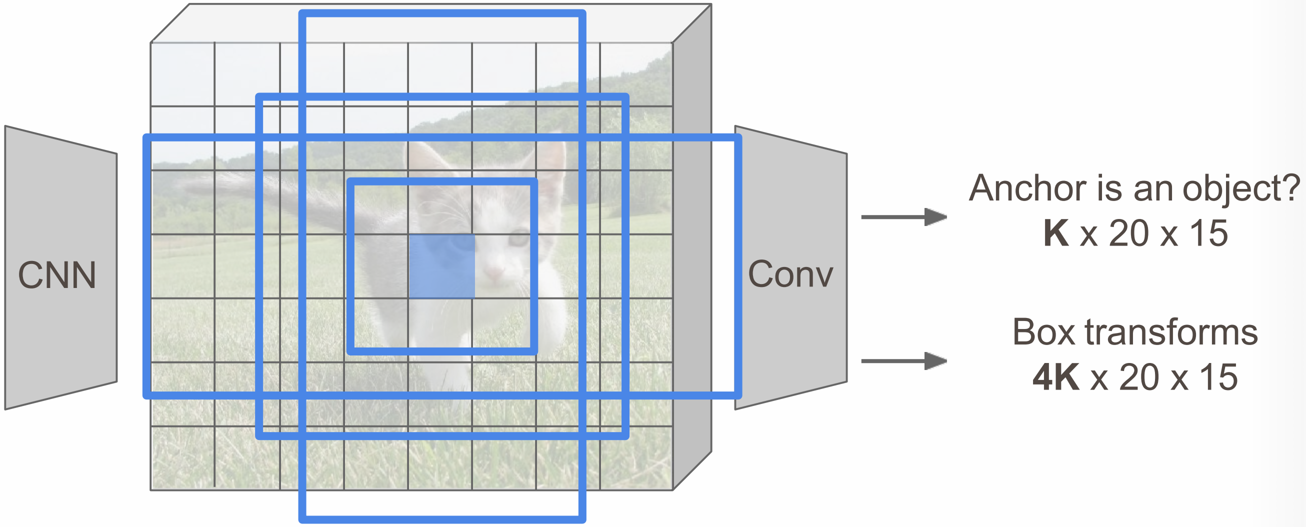
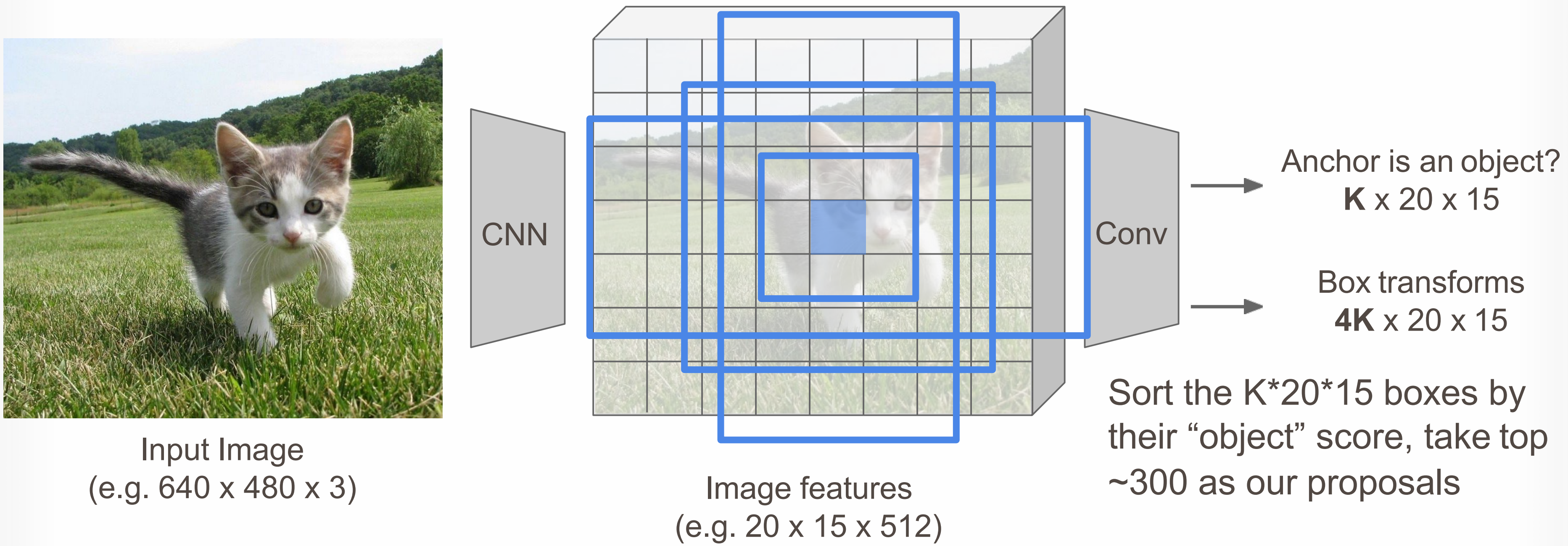


Image features
(e.g. 20 x 15 x 512)

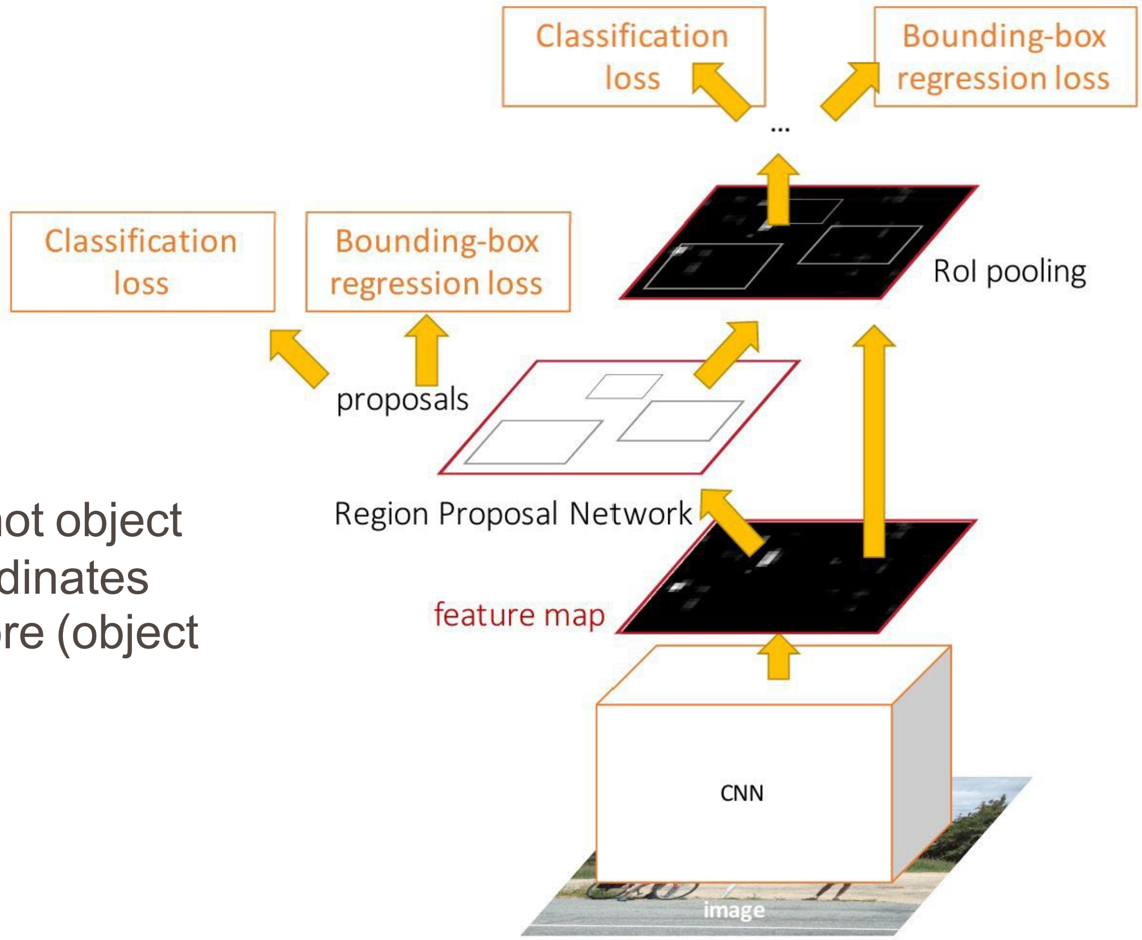
In practice use K different anchor boxes
of different size / scale at each point

Region Proposal Network



Sort the K*20*15 boxes by their “object” score, take top ~300 as our proposals

Faster R-CNN:
Make CNN do proposals!

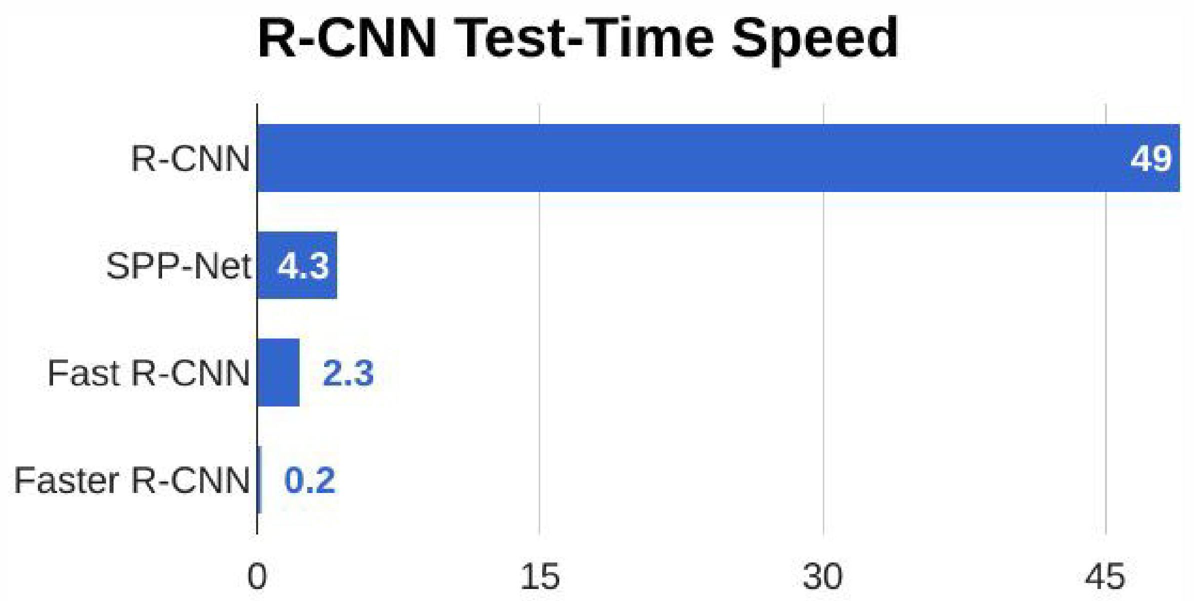


Jointly train with 4 losses:

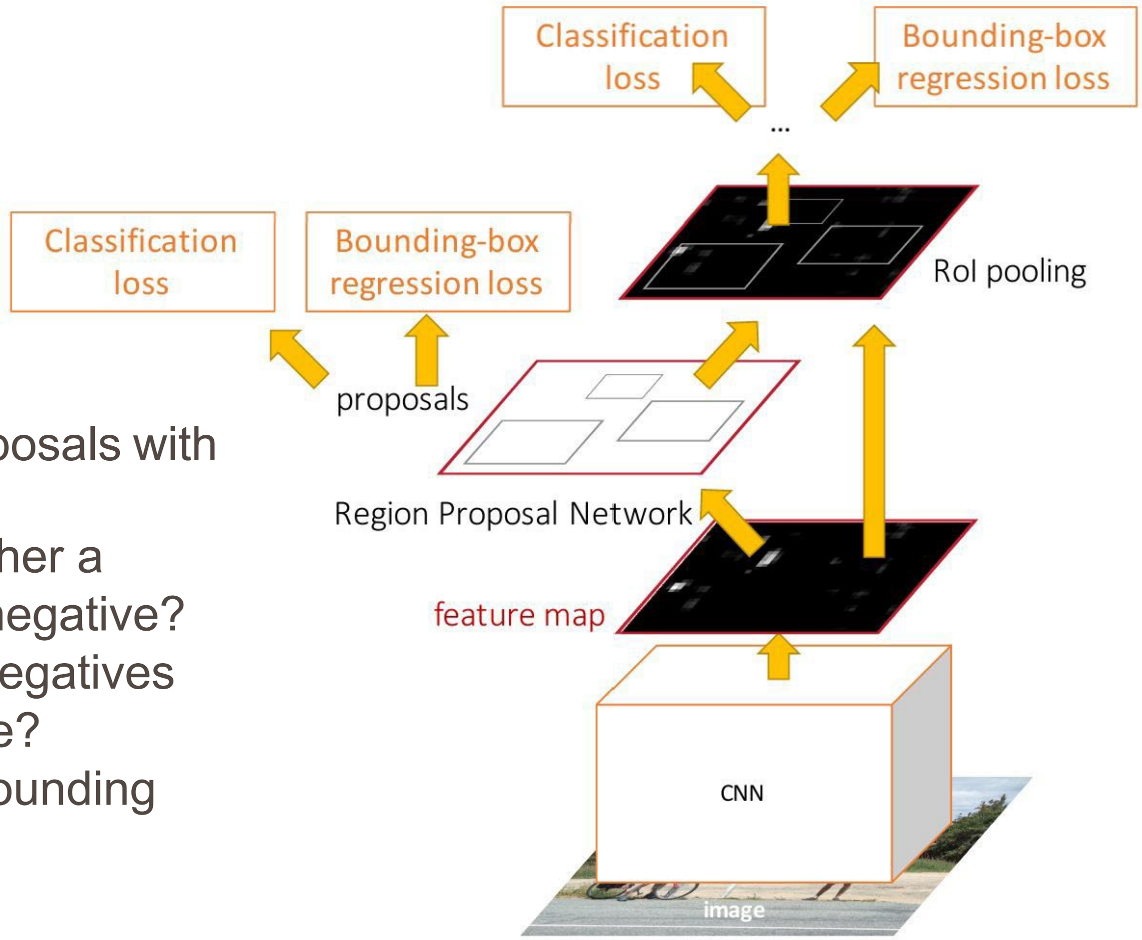
1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Regic
Figure copyright 2015, Ross Girshick; reproduced with permission

Faster R-CNN:
Make CNN do proposals!



Faster R-CNN:
Make CNN do proposals!



Glossing over many details:

- Ignore overlapping proposals with **non-max suppression**
- How to determine whether a proposal is positive or negative?
- How many positives / negatives to send to second stage?
- How to parameterize bounding box regression?

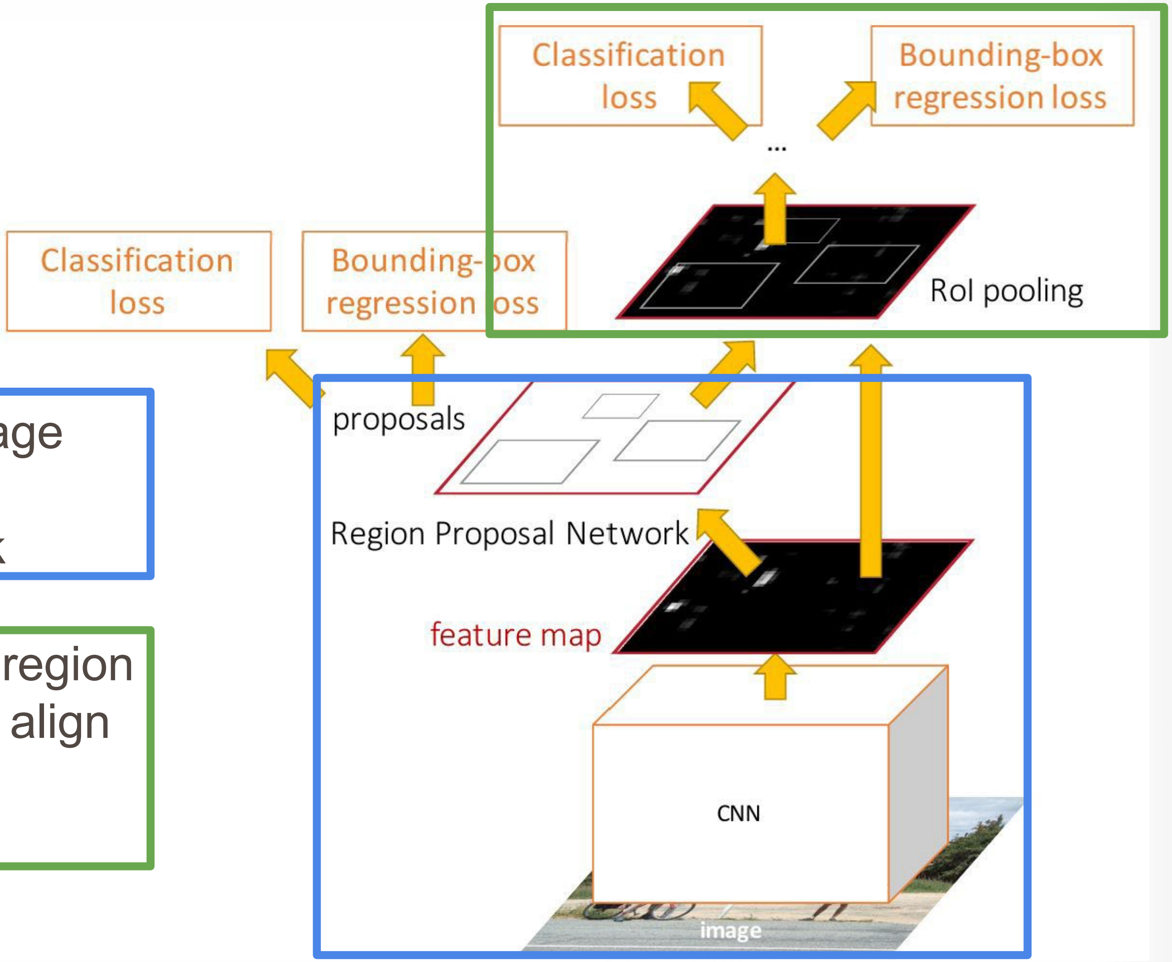
Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

Faster R-CNN:
Make CNN do proposals!

Faster R-CNN is a
Two-stage object detector

First stage: Run once per image
- Backbone network
- Region proposal network

Second stage: Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset



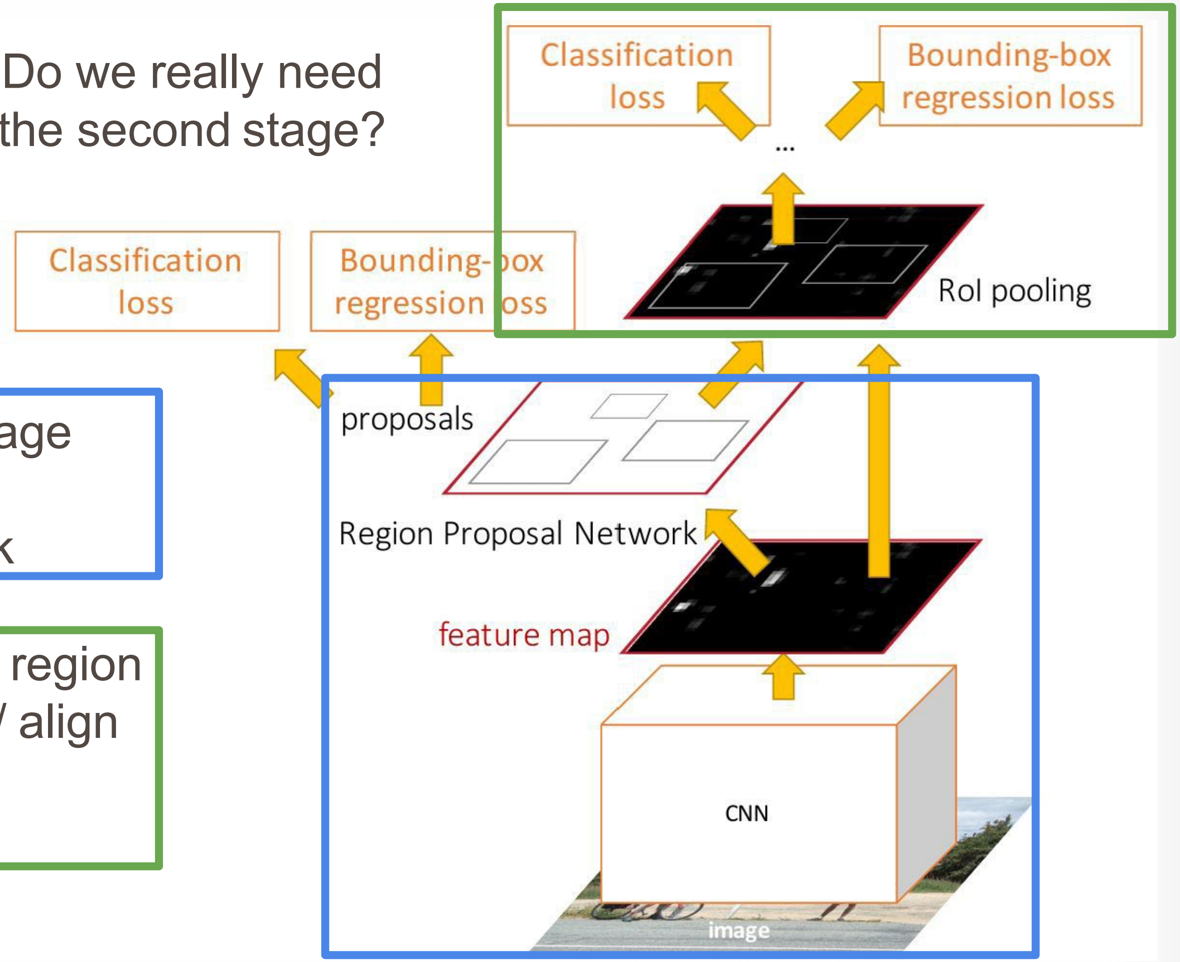
Faster R-CNN:
Make CNN do proposals!

Do we really need the second stage?

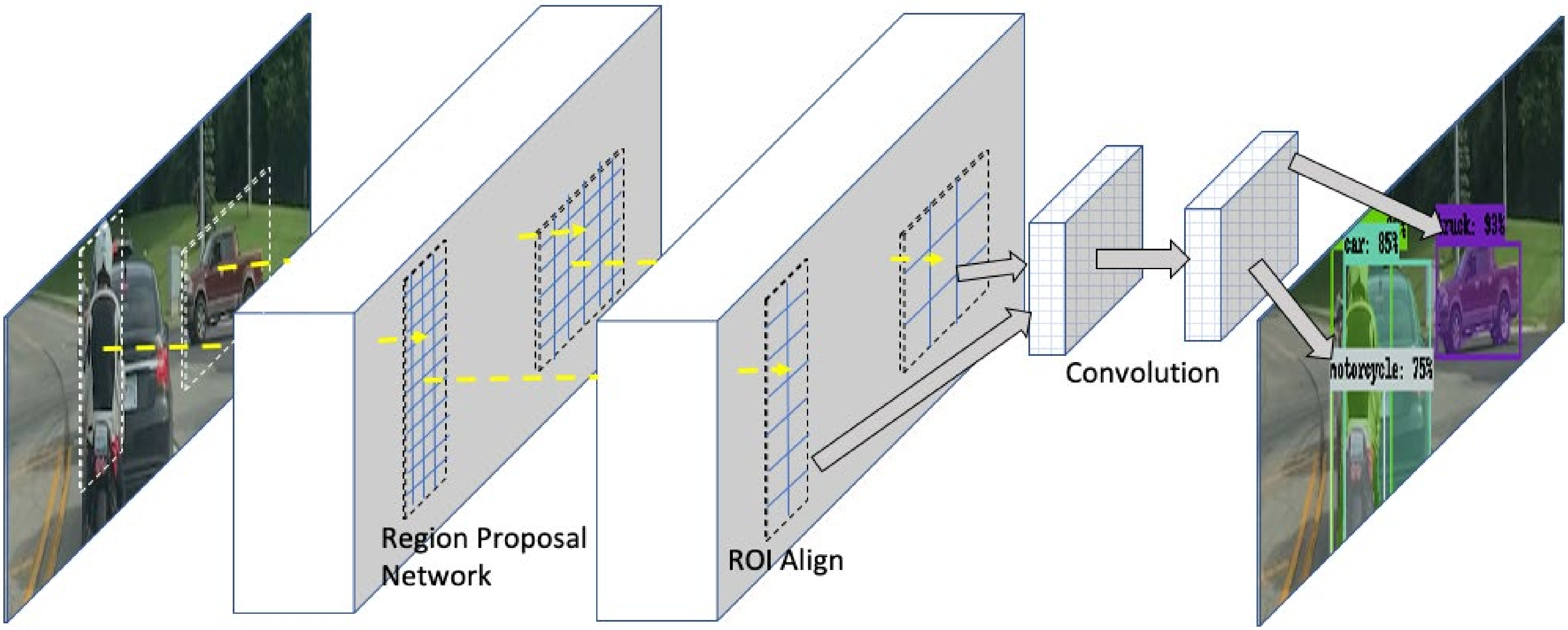
Faster R-CNN is a
Two-stage object detector

First stage: Run once per image
- Backbone network
- Region proposal network

Second stage: Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset

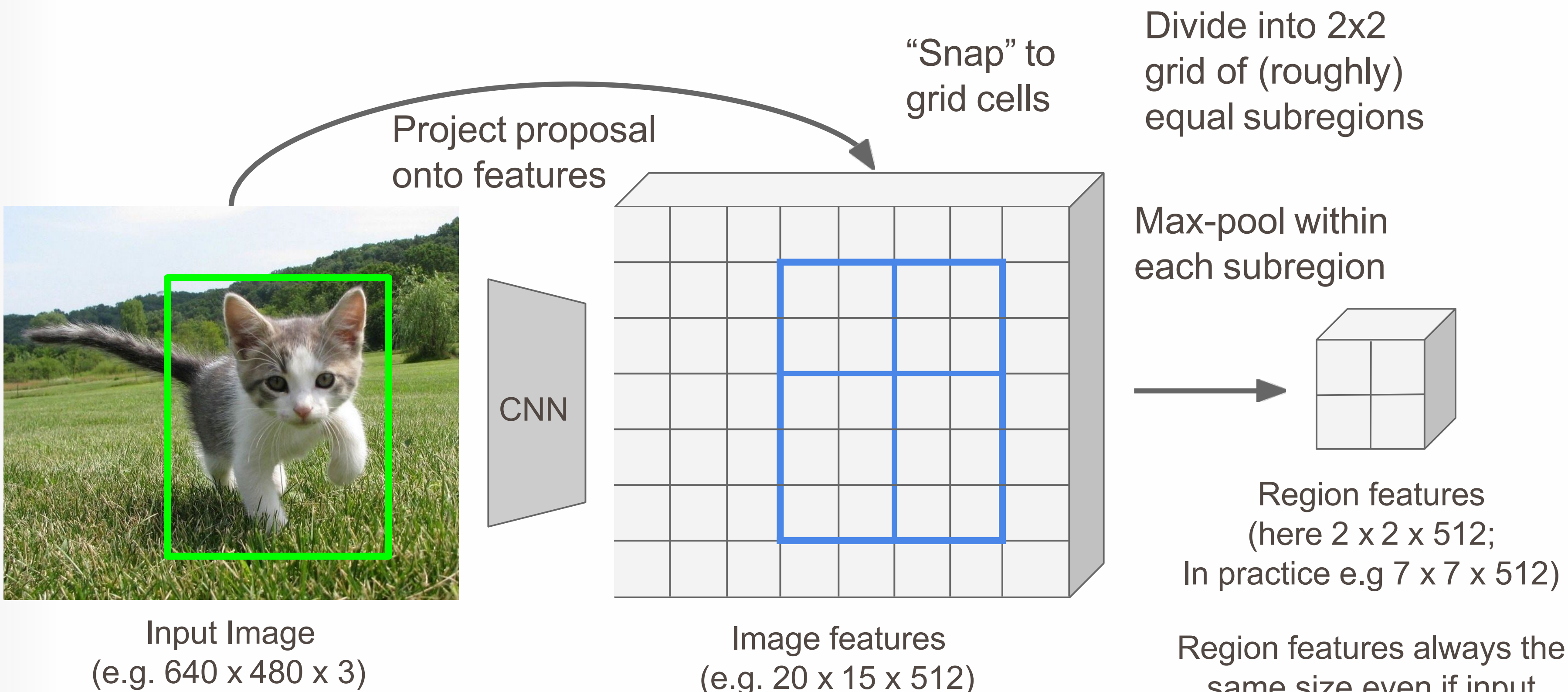


Mask RCNN



Region Proposal Network
ROI Align
Convolution
truck: 93%
car: 85%
motorcycle: 75%

Cropping Features: RoI Pool

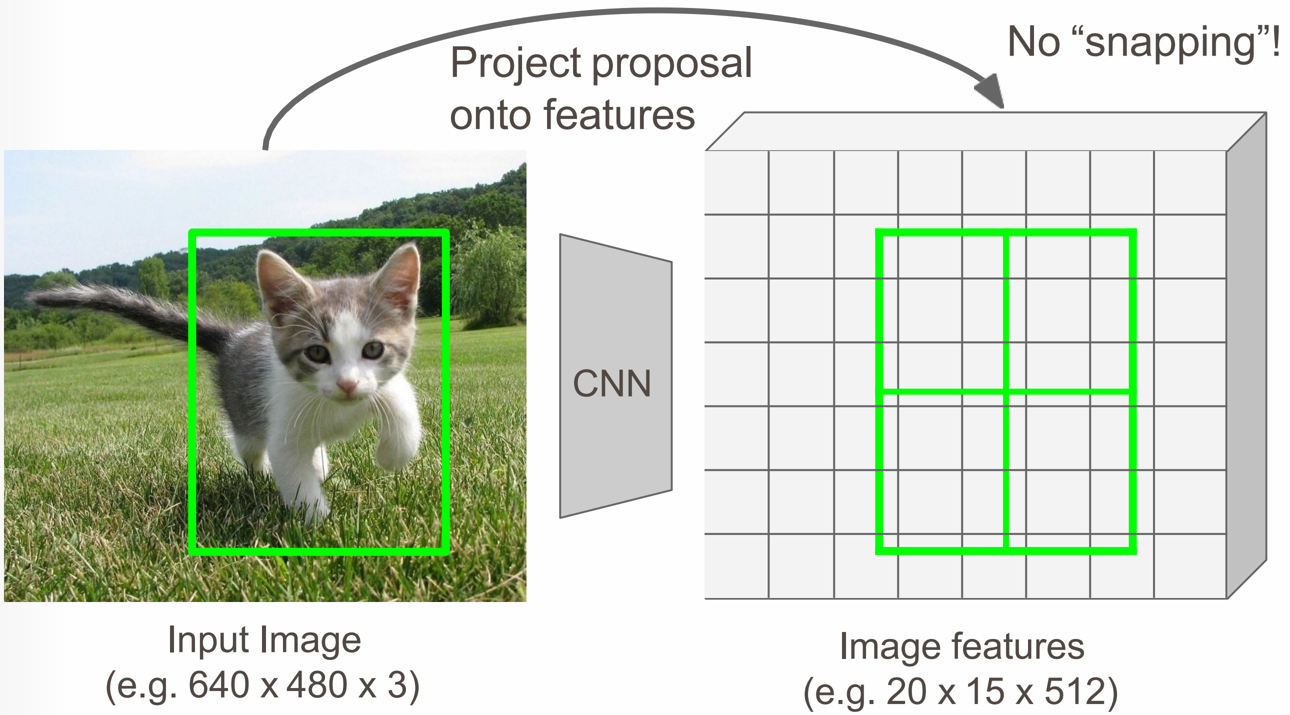


Region features always the same size even if input regions have different sizes!

Girshick, "Fast R-CNN", ICCV 2015.

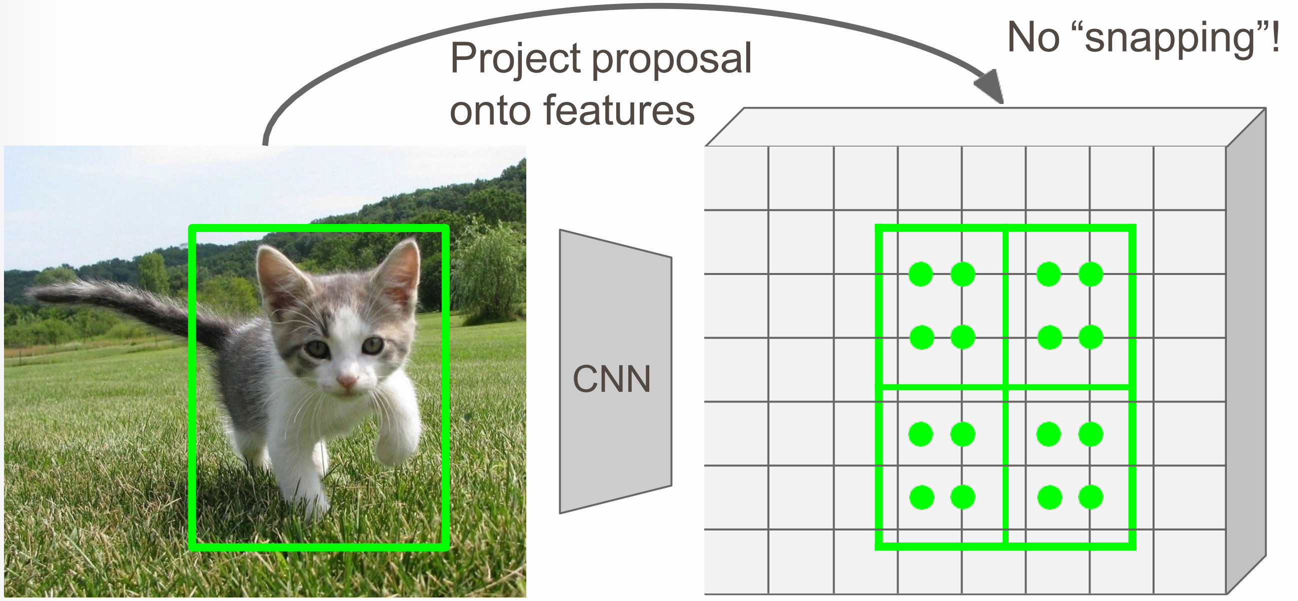
Problem: Region features slightly misaligned

Cropping Features: RoI Align



He et al, “Mask R-CNN”, ICCV 2017

Cropping Features: RoI Align



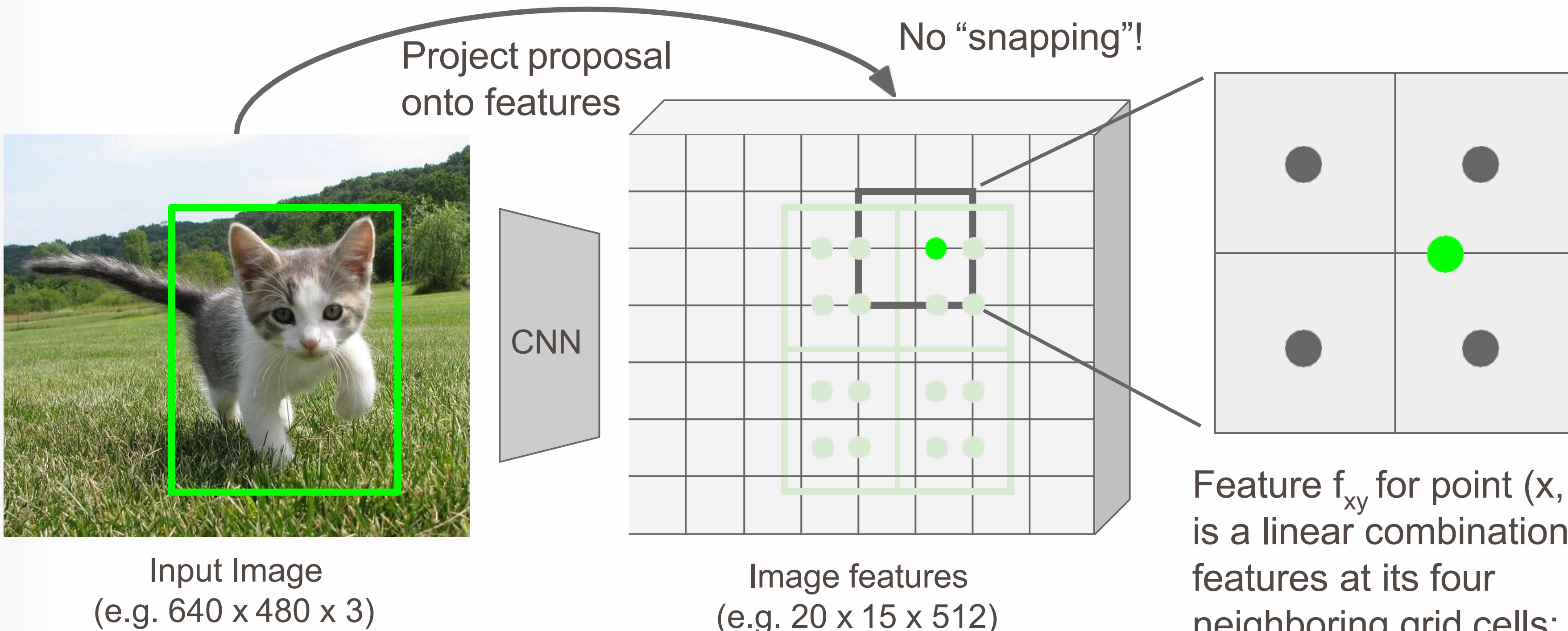
Input Image
(e.g. 640 x 480 x 3)

Image features
(e.g. 20 x 15 x 512)

He et al, “Mask R-CNN”, ICCV 2017

Sample at regular points in each subregion using bilinear interpolation

Cropping Features: RoI Align



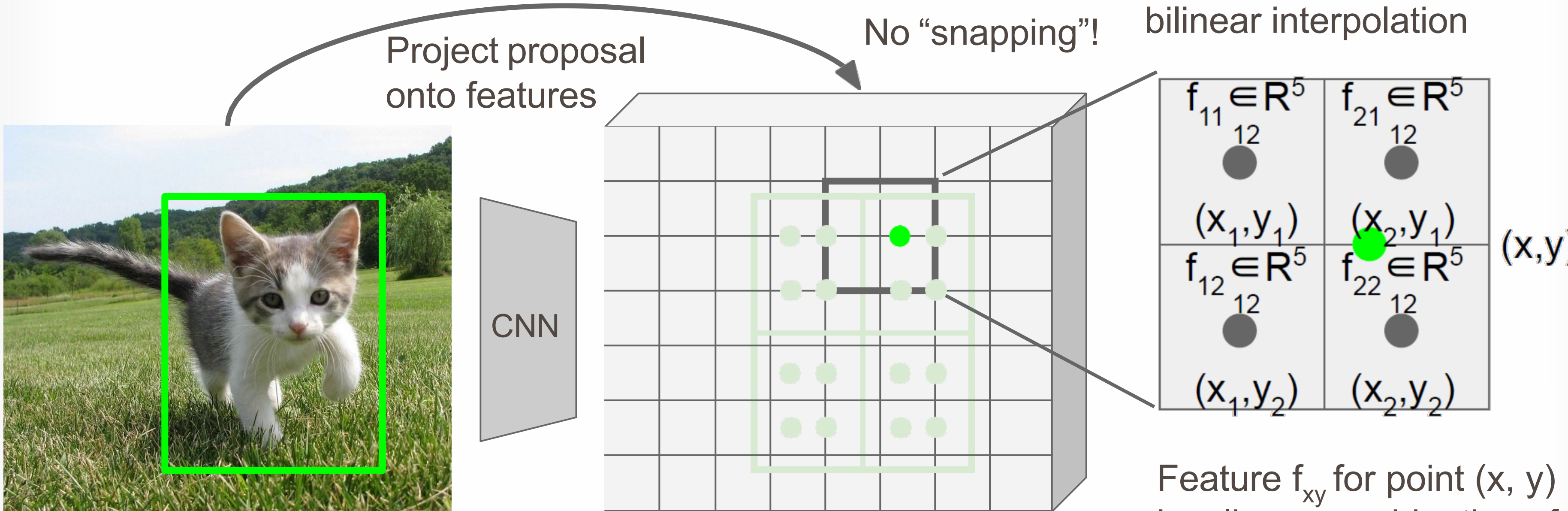
Feature f_{xy} for point (x, y) is a linear combination of features at its four neighboring grid cells:

He et al, “Mask R-CNN”, ICCV 2017

Sample at regular points in each subregion using bilinear interpolation

Cropping Features: RoI Align

Sample at regular points in each subregion using bilinear interpolation



Input Image
(e.g. 640 x 480 x 3)

Image features
(e.g. 20 x 15 x 512)

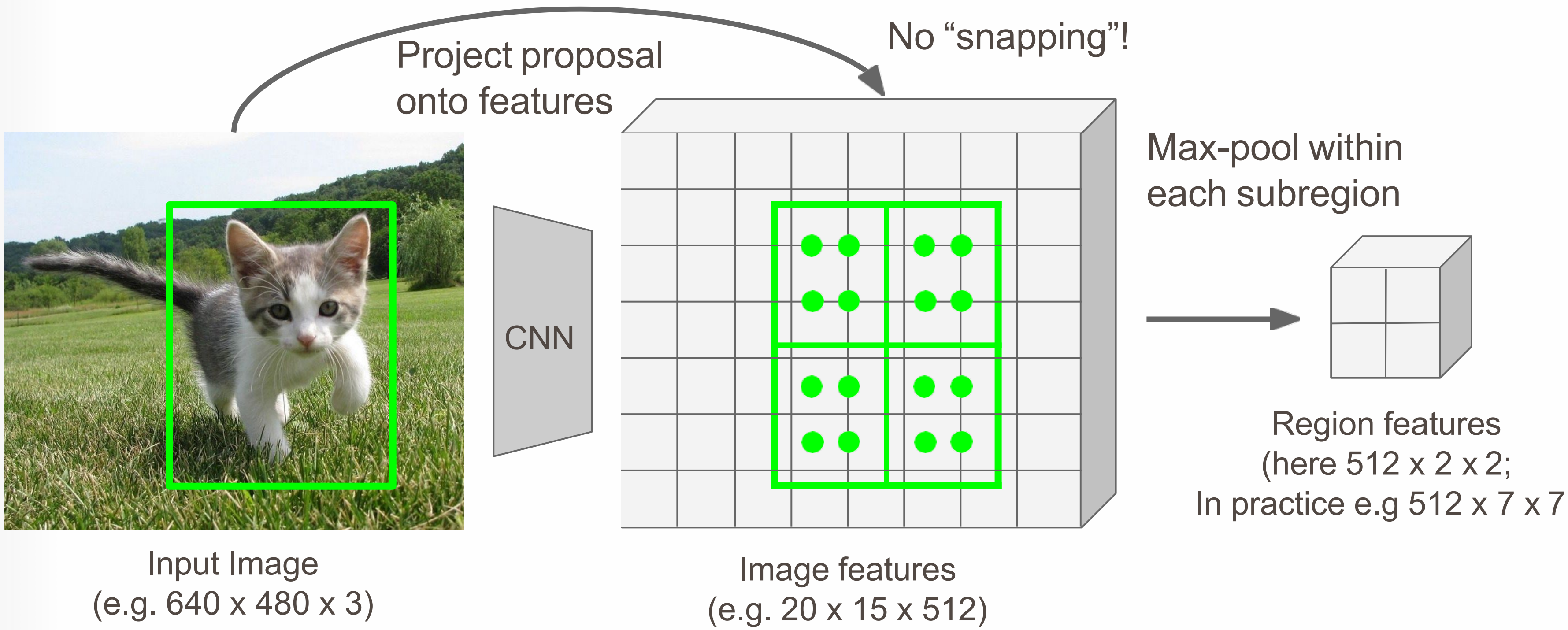
Feature f_{xy} for point (x, y) is a linear combination of features at its four neighboring grid cells:

He et al, “Mask R-CNN”, ICCV 2017

$$f_{xy} = \sum_{i,j=1}^{2} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_j|)$$

Bilinear interpolation ➔ Fxy

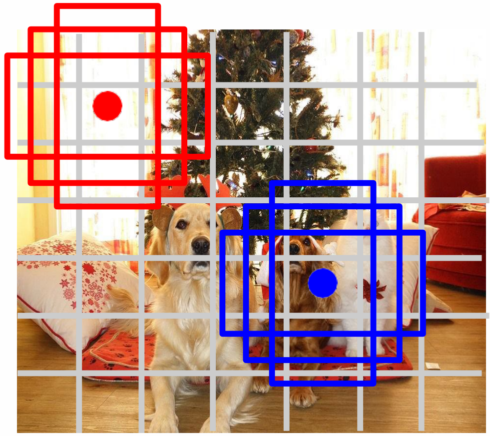
Cropping Features: RoI Align



He et al, "Mask R-CNN", ICCV 2017

Single-Stage Object Detectors: YOLO / SSD / RetinaNet

Input image
3 x H x W



Divide image into grid
7 x 7

Image a set of **base boxes** centered at each grid cell
Here B = 3

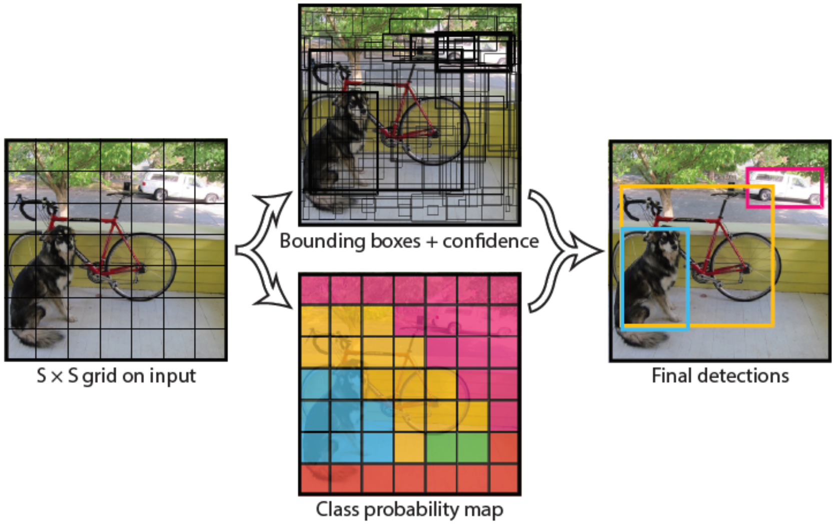
Within each grid cell:
- Regress from each of the B base boxes to a final box with 5 numbers: $(dx, dy, dh, dw, confidence)$
- Predict scores for each of C classes (including background as a class)
- Looks a lot like RPN, but category-specific!

Output:
7 x 7 x (5 * B + C)

Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016
Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

YOLO- You Only Look Once

Idea: No bounding box proposals. Predict a class and a box for every location in a grid.



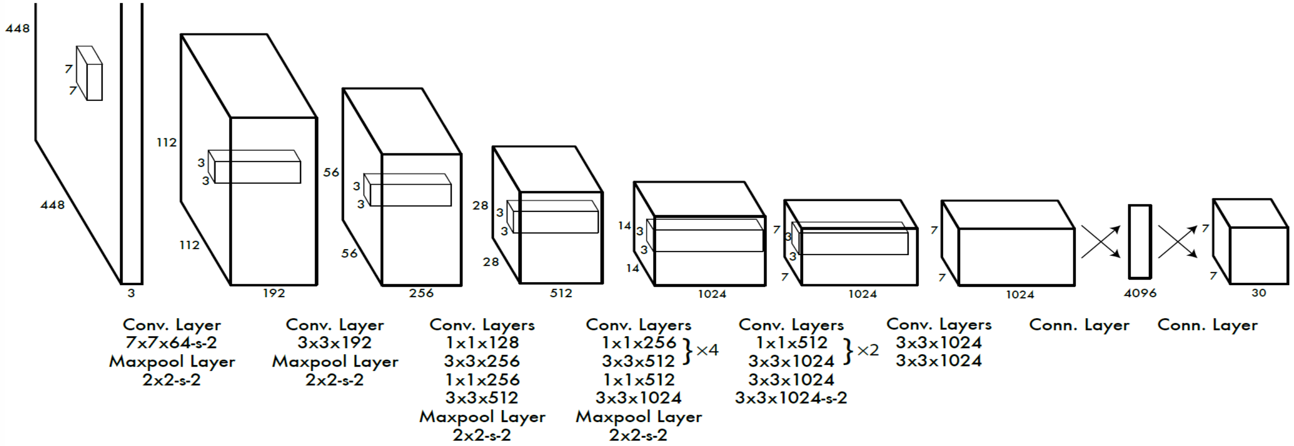
Redmon et al. CVPR 2016.

https://arxiv.org/abs/1506.02640

Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

For evaluating YOLO on PASCAL VOC, we use $S = 7$, $B = 2$. PASCAL VOC has 20 labelled classes so $C = 20$. Our final prediction is a $7 \times 7 \times 30$ tensor.

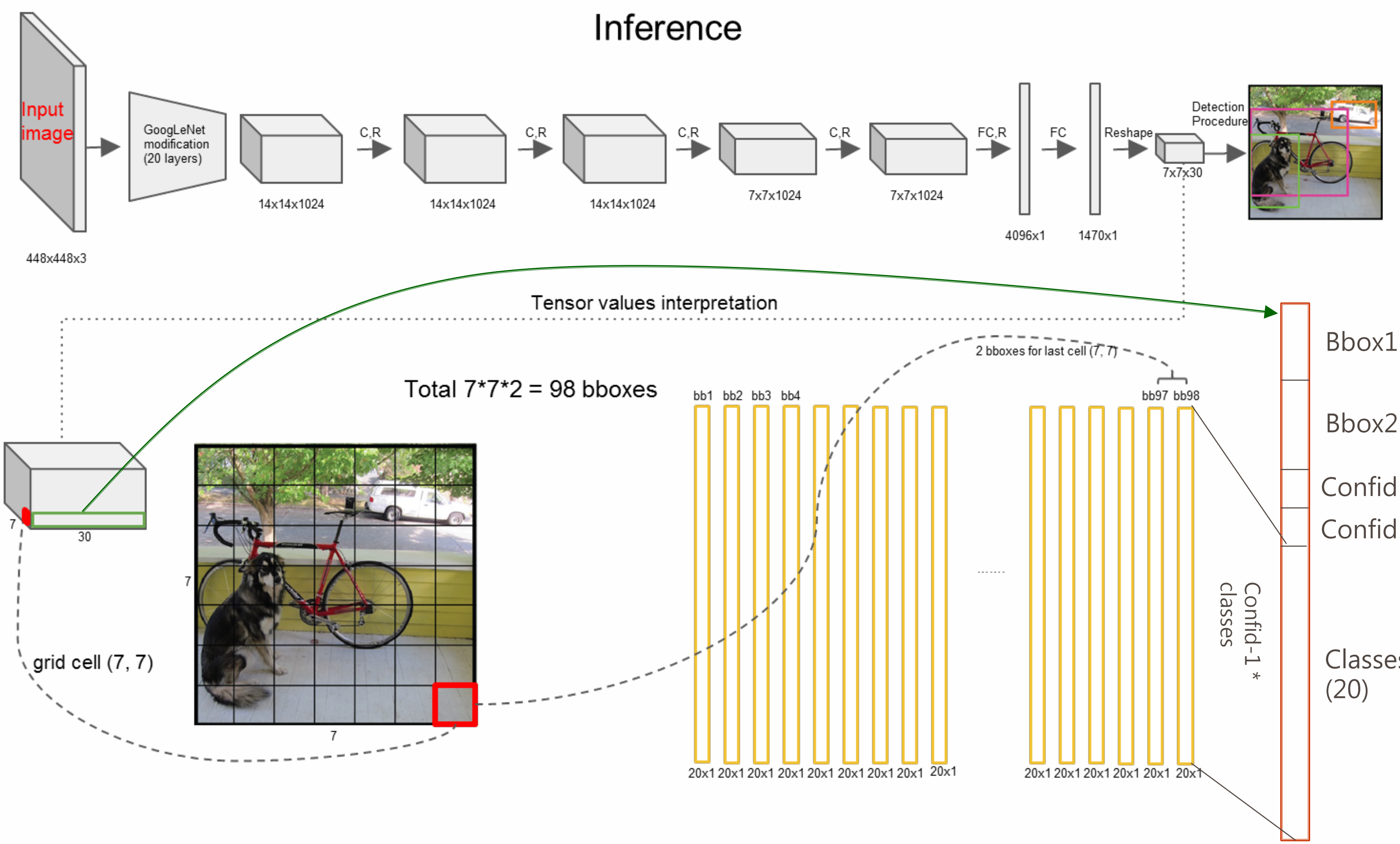
YOLO- You Only Look Once



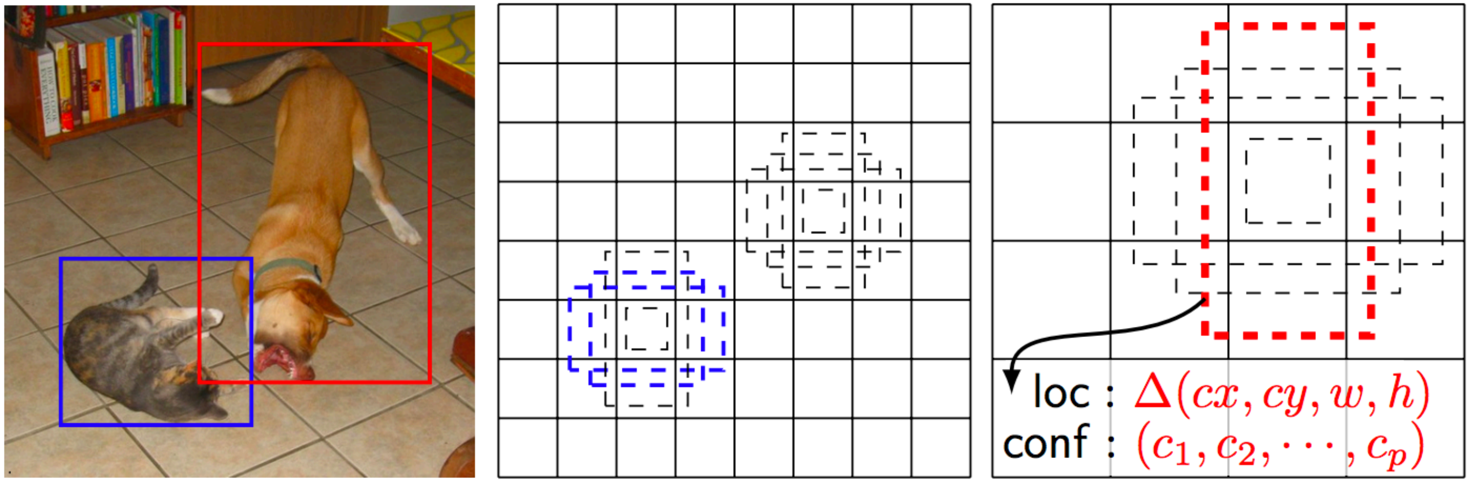
Divide the image into 7x7 cells.
Each cell trains a detector.
The detector needs to predict the object's class distributions.
The detector has 2 bounding-box predictors to predict
bounding-boxes and confidence scores.

https://arxiv.org/abs/1506.02640 Redmon et al. CVPR 2016.

YOLO for Inference



SSD: Single Shot Detector

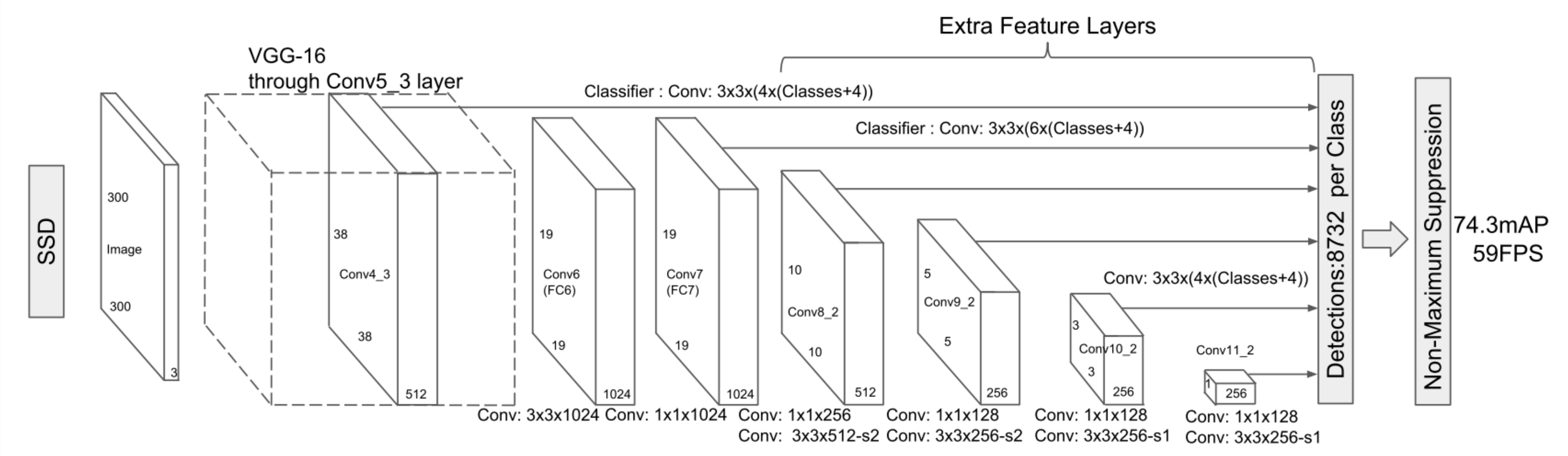


(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Idea: Similar to YOLO, but denser grid map, multiscale grid maps. + Data augmentation + Hard negative mining + Other design choices in the network.

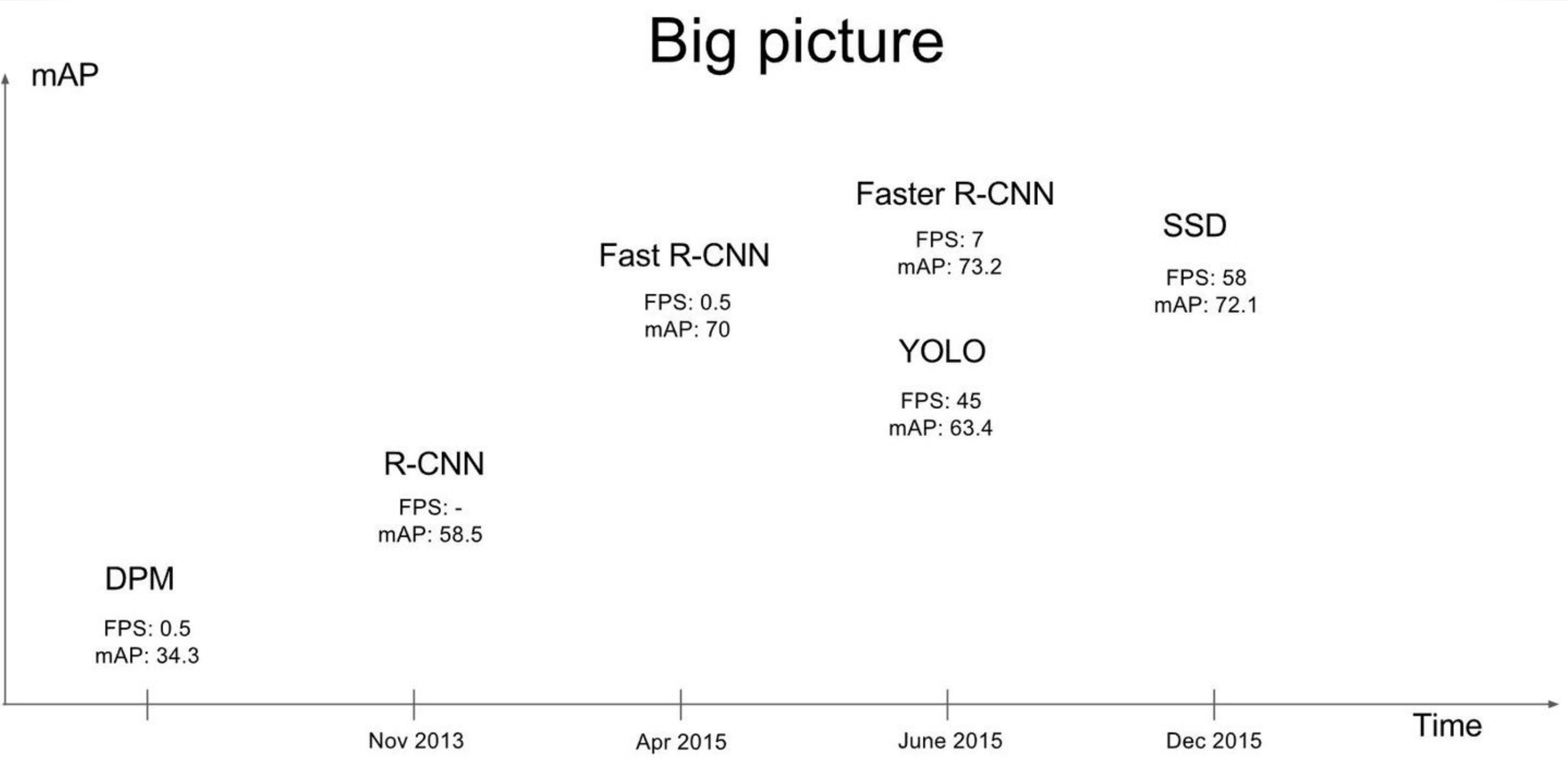
Liu et al. ECCV 2016.

SSD: Single Shot Detector



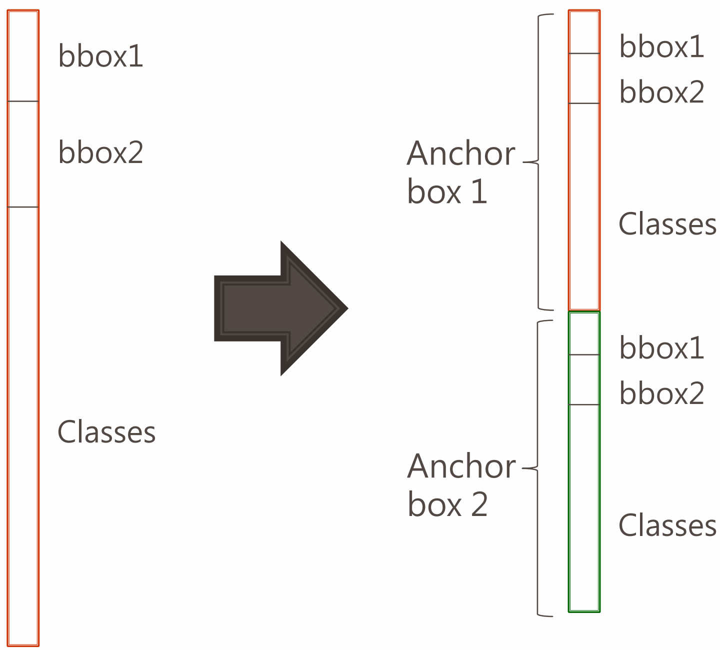
Aggregate different levels of layers to obtain multi-scale feature representation

YOLO vs. SSD: which one is better?



YOLO v2

- Introduce "anchor box"
- Batch normalization



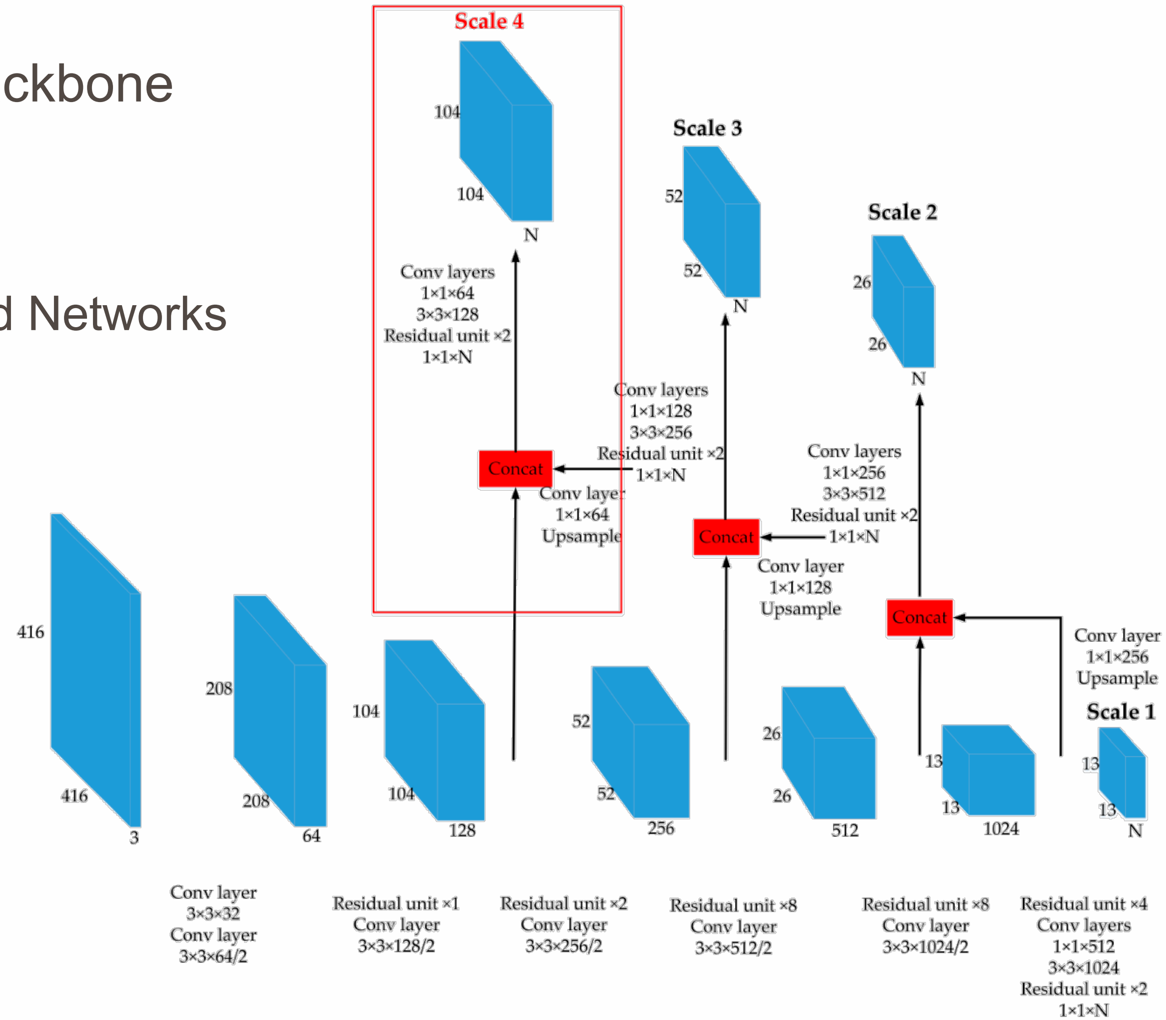
How many anchor boxes do we need?

In general, 9 anchor boxes

Use k-means on GT to obtain default anchor boxes setting

YOLO v3: Multi-scale

- ResNet-based backbone
  - Darknet!
- FPN
  - Feature Pyramid Networks



Object Detection: Lots of variables ...

Baseline Network
VGG16
ResNet-101
Inception V2
Inception V3
Inception
ResNet
MobileNet

"Meta-Architecture"
Two-stage: Faster R-CNN
Single-stage: YOLO / SSD
Hybrid: R-FCN

Image Size
 # Region Proposals
…

Takeaways
Faster R-CNN is slower but more accurate

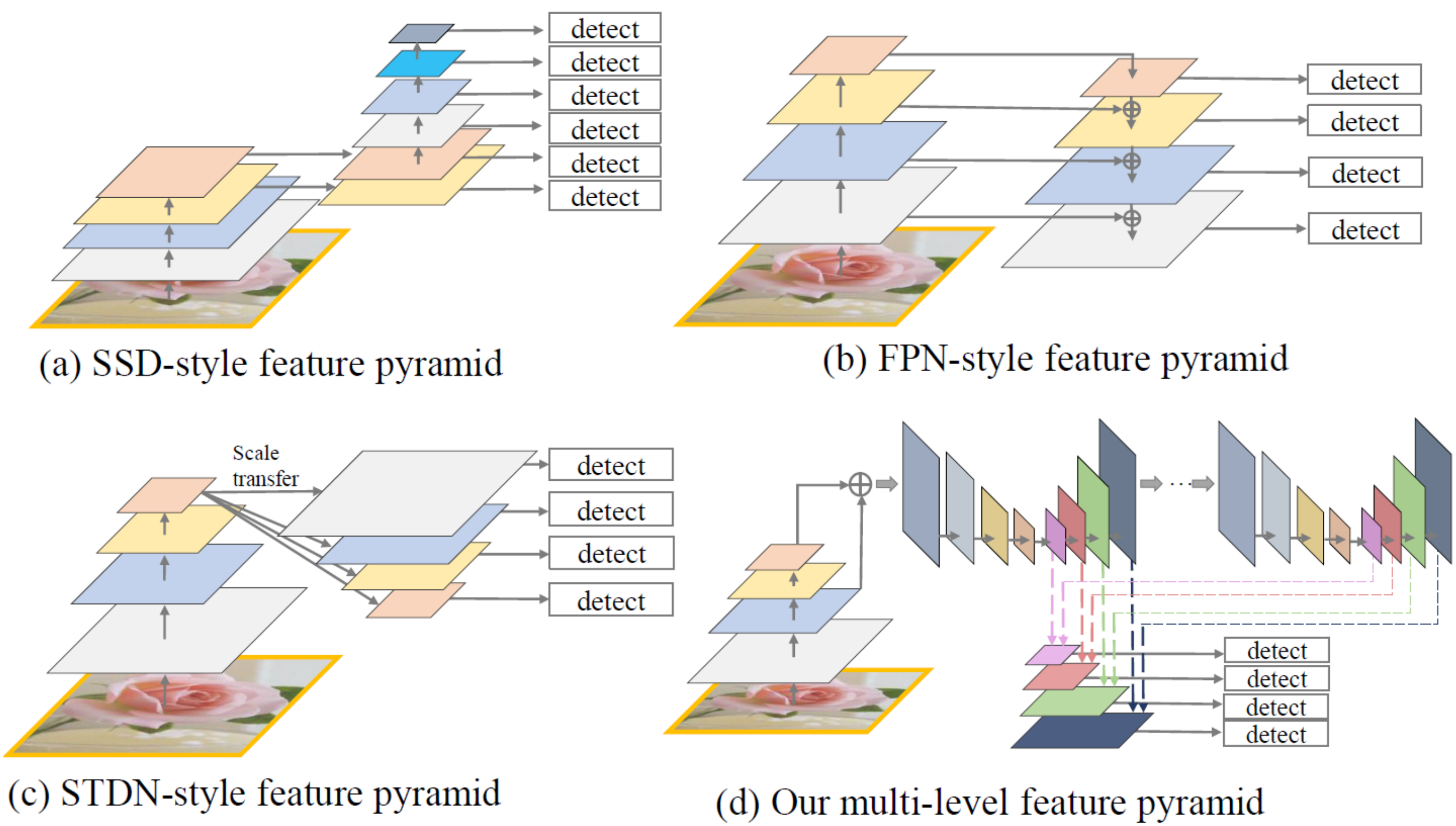
SSD is much faster but not as accurate

Bigger / Deeper backbones work better

Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017
Zou et al, "Object Detection in 20 Years: A Survey", arXiv 2019 (today!)

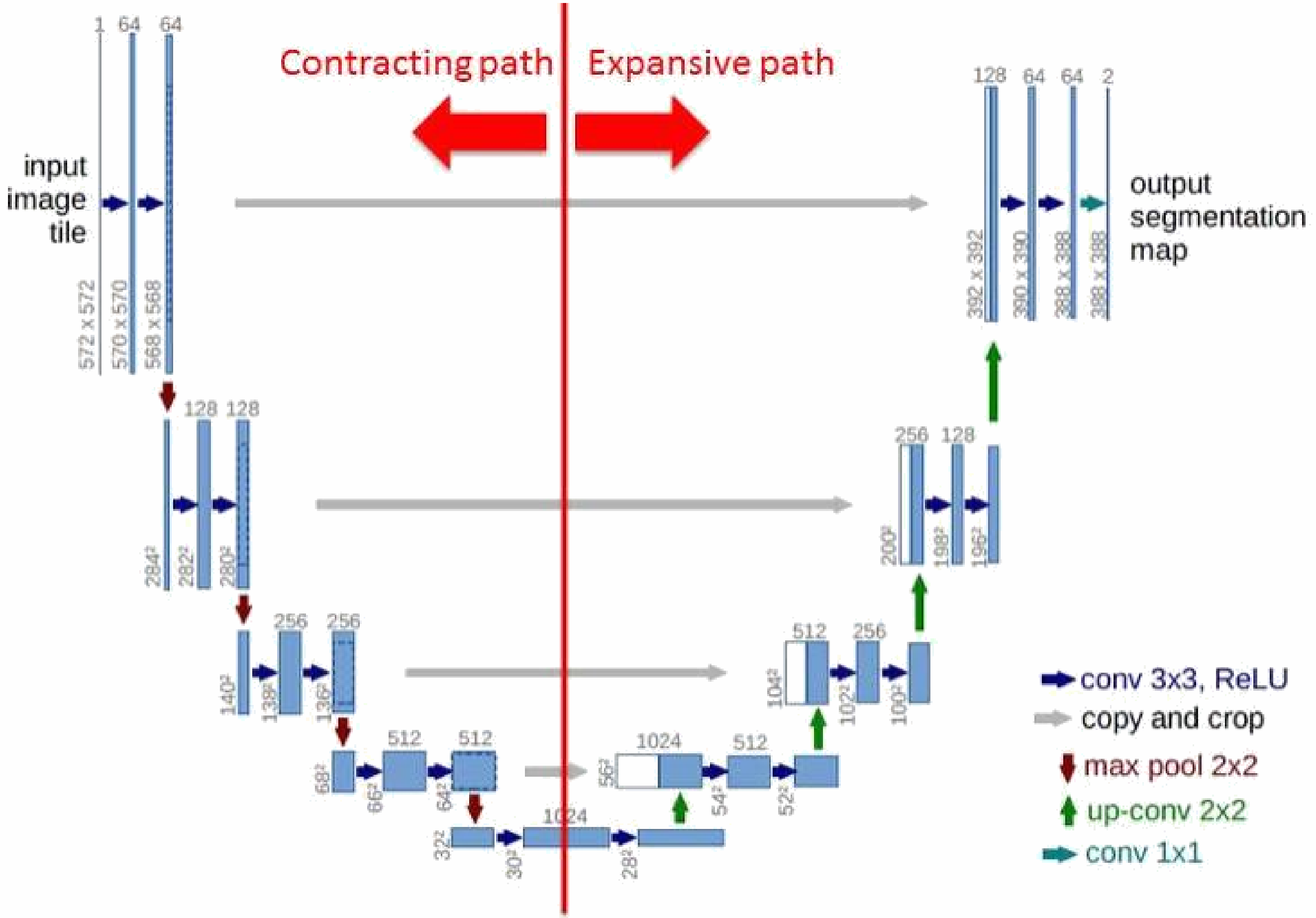
R-FCN: Dai et al, "R-FCN: Object Detection via Region-based Fully Convolutional Networks", NIPS 2016
Inception-V2: Ioffe and Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", ICML 2015
Inception V3: Szegedy et al, "Rethinking the Inception Architecture for Computer Vision", arXiv 2016
Inception ResNet: Szegedy et al, "Inception-V4, Inception-ResNet and the Impact of Residual Connections on Learning", arXiv 2016
MobileNet: Howard et al, "Efficient Convolutional Neural Networks for Mobile Vision Applications", arXiv 2017

Latest Object Detection: M2Det (AAAI'19)



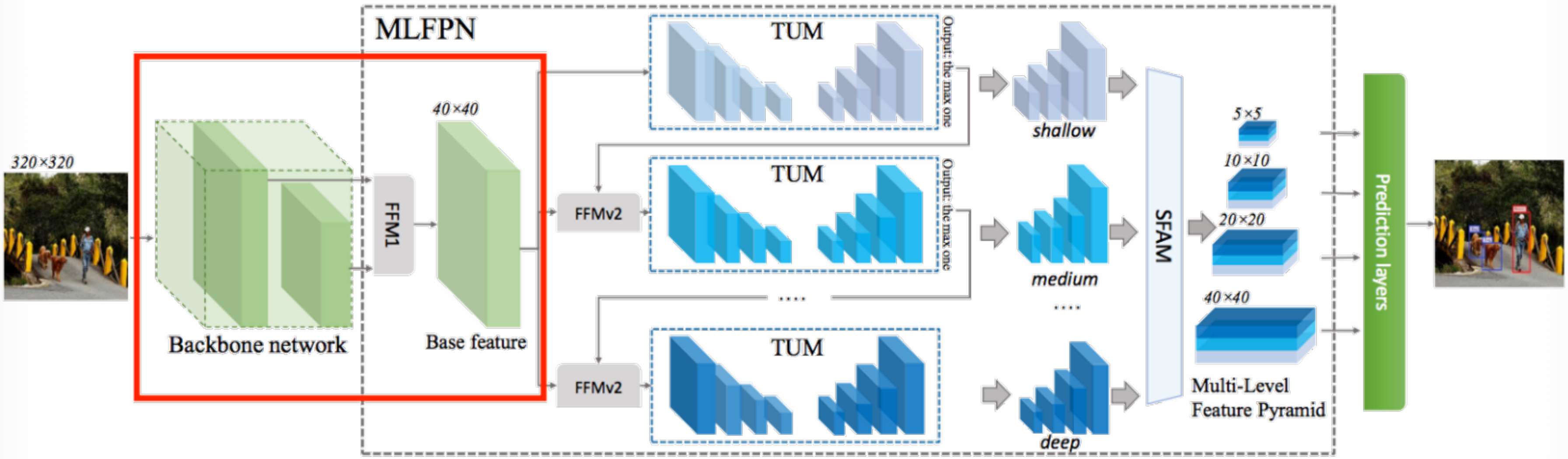
M2Det

- Combine U-Net (Semantic segmentation) with SSD



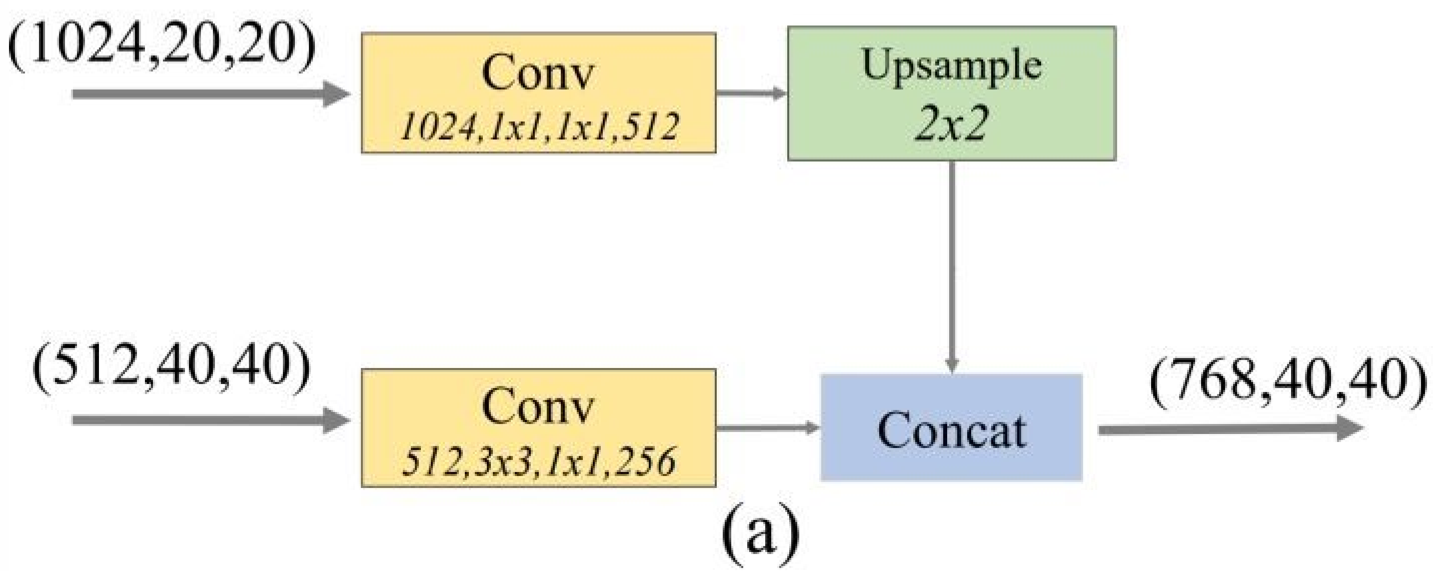
M2Det

- FFM: Feature fusion module
- TMU: Thinned U-shape Modules
- SFAM: Scale-wise Feature Aggregation Module



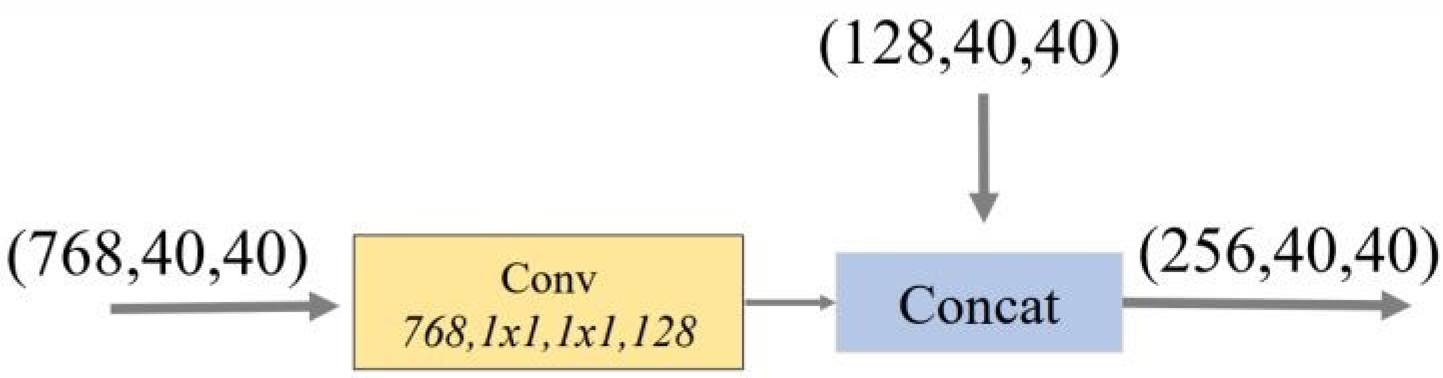
FFM v1

- Combine different layers across different scale
  - Similar to DenseNet



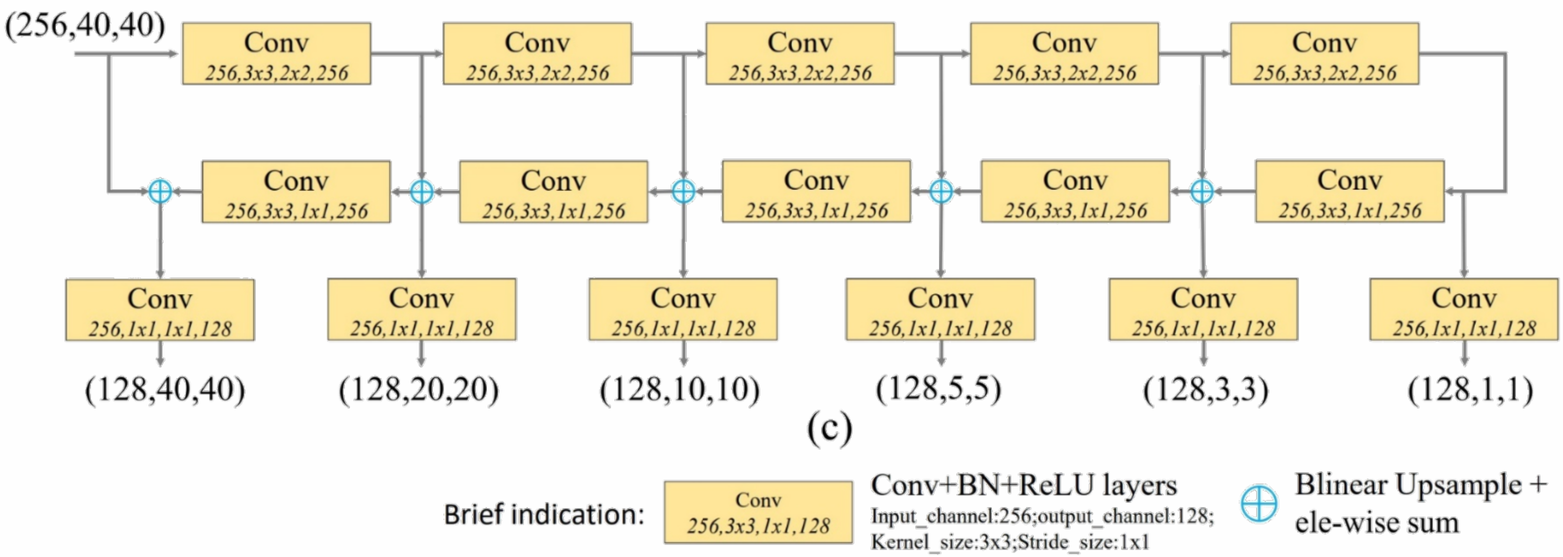
FFM v2

- For same scale of feature maps:
  - Concat directly
  - Similar to DenseNet



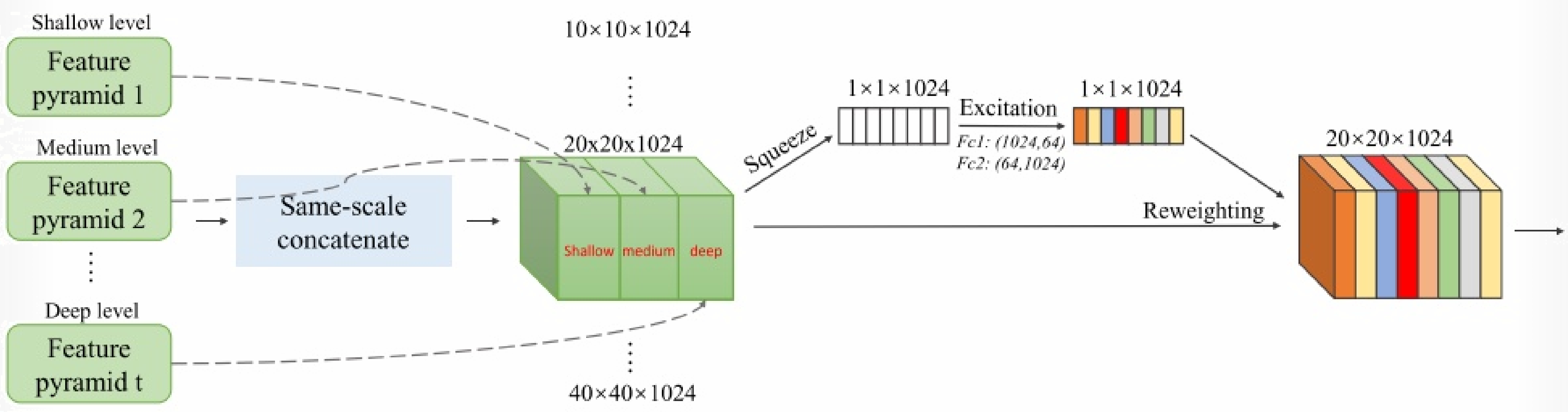
(b)

TMU


(256,40,40)
Conv
256,3x3,2x2,256
Conv
256,3x3,2x2,256
Conv
256,3x3,2x2,256
Conv
256,3x3,2x2,256
Conv
256,3x3,2x2,256
Conv
256,3x3,1x1,256
Conv
256,3x3,1x1,256
Conv
256,3x3,1x1,256
Conv
256,3x3,1x1,256
Conv
256,3x3,1x1,256
Conv
256,1x1,1x1,128
Conv
256,1x1,1x1,128
Conv
256,1x1,1x1,128
Conv
256,1x1,1x1,128
Conv
256,1x1,1x1,128
Conv
256,1x1,1x1,128
(128,40,40)
(128,20,20)
(128,10,10)
(128,5,5)
(128,3,3)
(128,1,1)
(c)
Brief indication:
Conv
256,3x3,1x1,128
Conv+BN+ReLU layers
Input_channel:256;output_channel:128;
Kernel_size:3x3;Stride_size:1x1
Blinear Upsample +
ele-wise sum

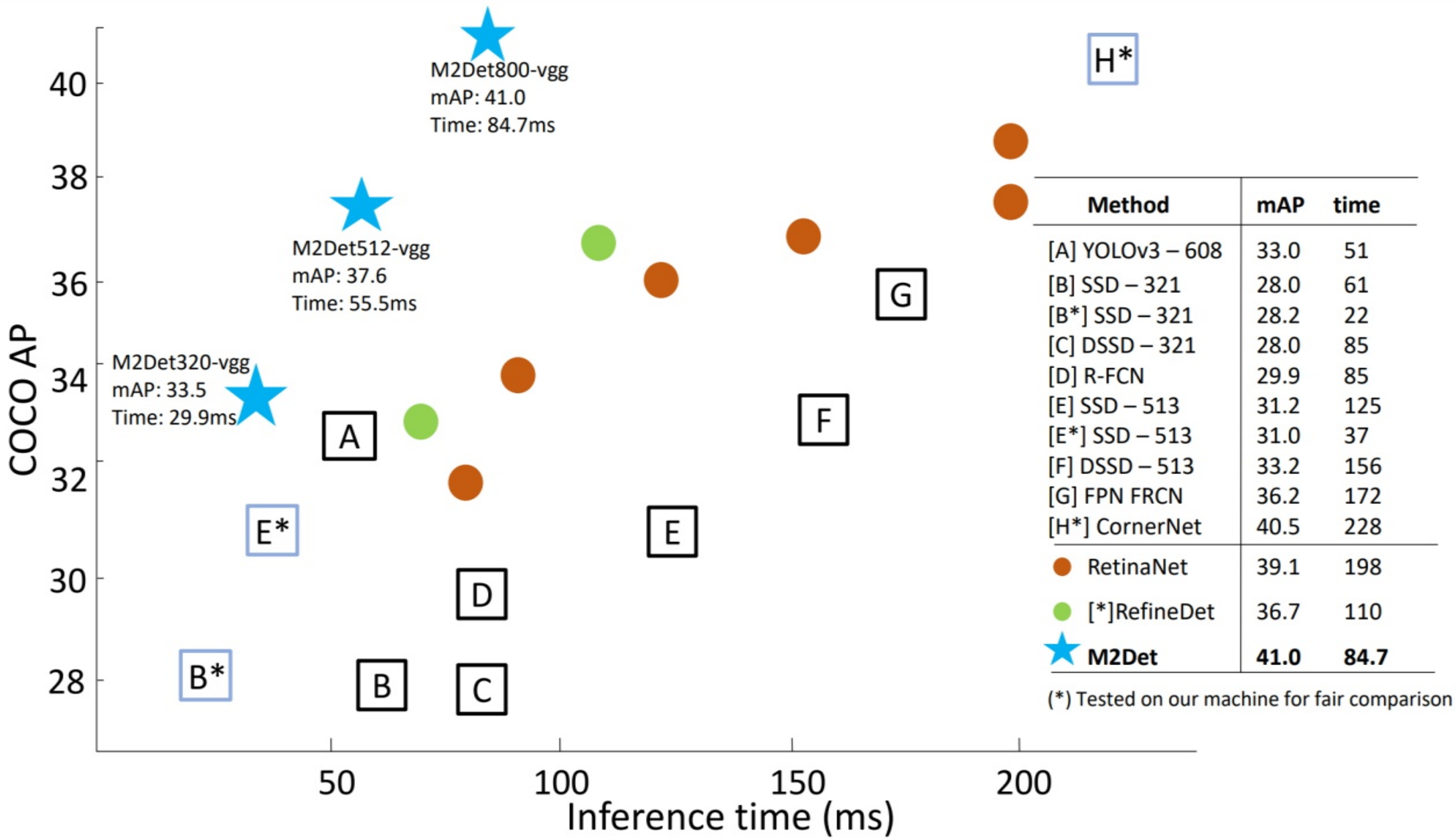
FSAM

- Fusion of different feature pyramids
  - Different scales of feature maps (i.e., different object sizes)
  - Adopt DenseNet-like structure



![ACVLab]

Experimental Result of M2Det



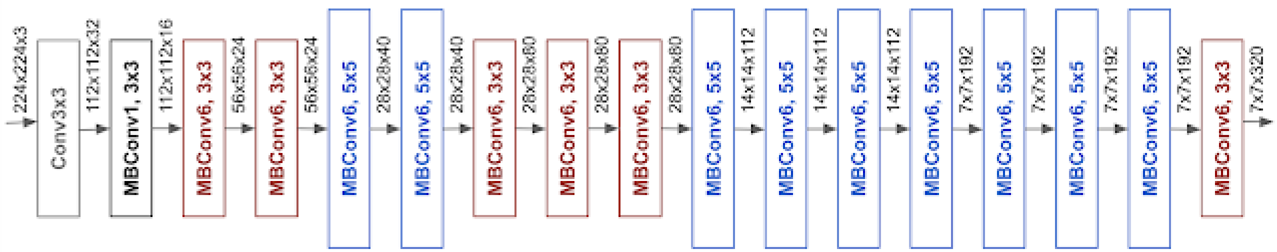
Method	mAP	time
[A] YOLOv3 – 608	33.0	51
[B] SSD – 321	28.0	61
[B*] SSD – 321	28.2	22
[C] DSSD – 321	28.0	85
[D] R-FCN	29.9	85
[E] SSD – 513	31.2	125
[E*] SSD – 513	31.0	37
[F] DSSD – 513	33.2	156
[G] FPN FRCN	36.2	172
[H*] CornerNet	40.5	228
RetinaNet	39.1	198
[*]RefineDet	36.7	110
M2Det	**41.0**	**84.7**

Figure 5: Speed (ms) vs. accuracy (mAP) on COCO *test-dev*.

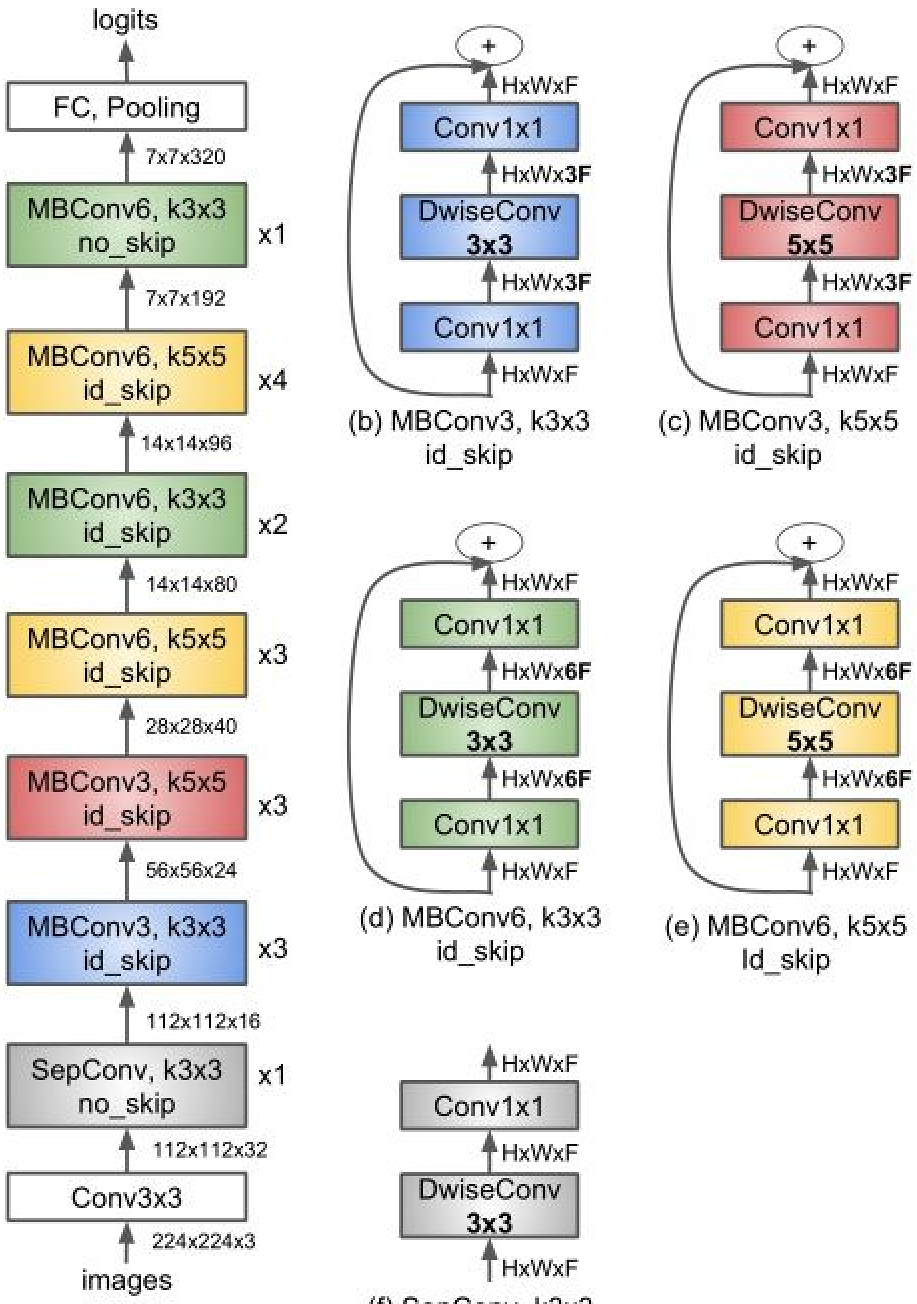
EFFICIENTDET [CVPR20]

EfficientDet

- EfficientNet + Detection head



- Bulletin by MB block
    - A lightweight blocks
    - Proposed by MobileNet



(a) MnasNet

(b) MBConv3, k3x3 id_skip

(c) MBConv3, k5x5 id_skip

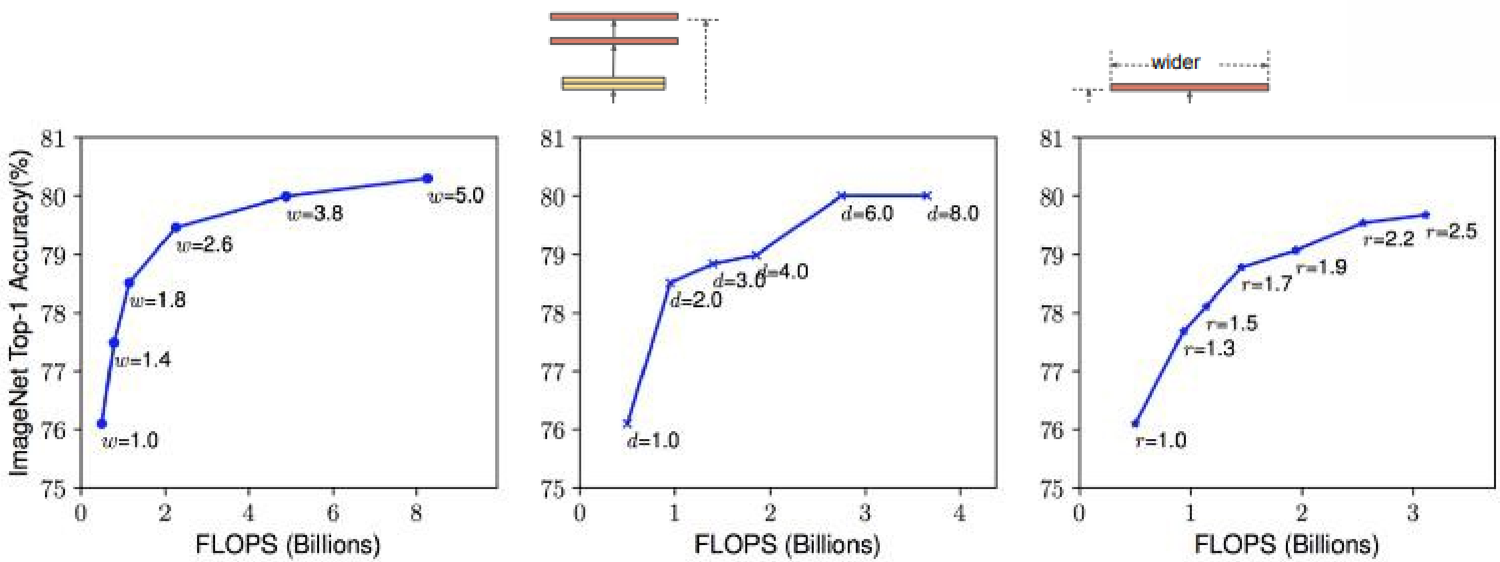
(d) MBConv6, k3x3 id_skip

(e) MBConv6, k5x5 Id_skip

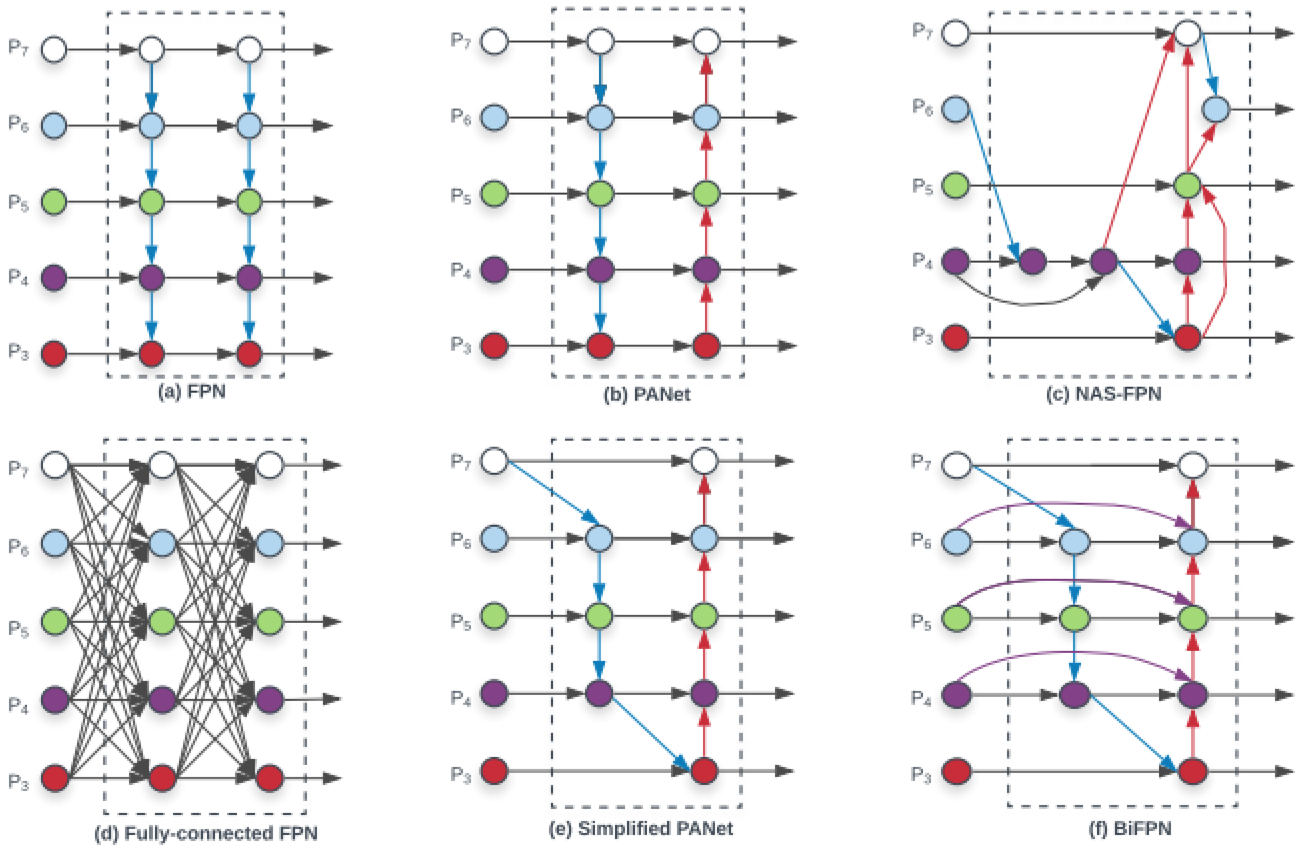
(f) SepConv, k3x3 no_skip

MBConv in EfficientNet

- A way to find a “good” rules for scaling
  - Compound scaling
    - Adjusting w, d, and r will result in limited improvement
      - How about combining them?

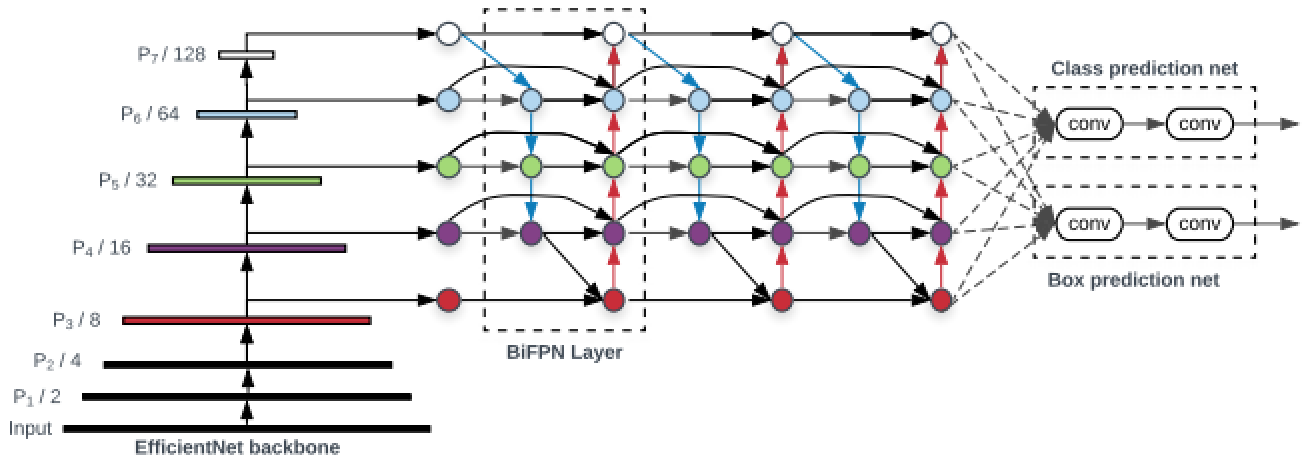


EfficientDet

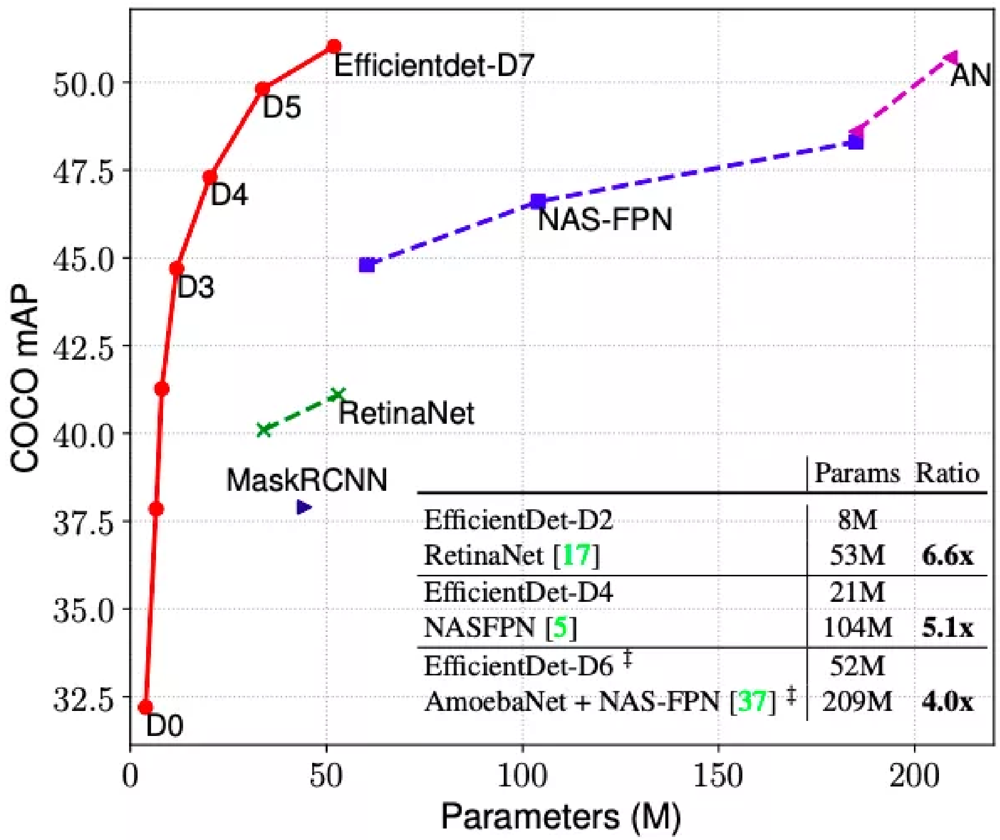


BiFPN

- The cross-layer feature fusion in EfficientDet
  - BiFPN is useful to integrate the multiple features



Results



	Params	Ratio
EfficientDet-D2	8M	
RetinaNet [17]	53M	**6.6x**
EfficientDet-D4	21M	
NASFPN [5]	104M	**5.1x**
EfficientDet-D6 ‡	52M	
AmoebaNet + NAS-FPN [37] ‡	209M	**4.0x**

(a) Model Size

YOLO V4

[CVPR2020 workshop]

What’s news in YOLO v4

- Backbone：
  - CSPDarkNet53
- Neck：
  - SPP，PAN
- Head：
  - YOLOv3
- Tricks（backbone）：
  - CutMix、Mosaic、DropBlock、Label Smoothing
- Modified（backbone）:
  - Mish、CSP、MiWRC
- Tricks（detector）：
  - CIoU、CMBN、DropBlock、Mosaic、SAT、Eliminate grid sensitivity、Multiple Anchor、Cosine Annealing scheduler、Random training shape
- Modified（detector）：Mish、SPP、SAM、PAN、DIoU-NMS

Data Augmentation (Mosaic)

aug_-319215602_0_-238783579.jpg

aug_-1271888501_0_-749611674.jpg

aug_1462167959_0_-1659206634.jpg

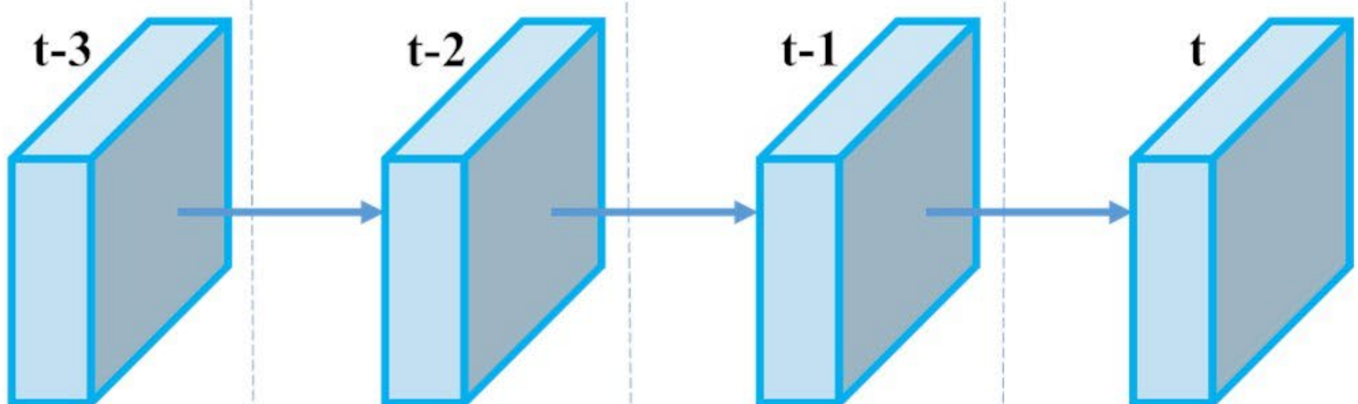
BN [32] – assume a batch contains four mini-batches

accumulate $W^{(t-3)}$ calculate $BN^{(t-3)}$ normalize BN	accumulate $W^{(t-3\sim t-2)}$ calculate $BN^{(t-2)}$ normalize BN	accumulate $W^{(t-3\sim t-1)}$ calculate $BN^{(t-1)}$ normalize BN	accumulate $W^{(t-3\sim t)}$ calculate $BN^{(t)}$ normalize BN update W, ScaleShift

CBN [89] – assume cross four iterations

update $W^{(t-3)}$ accumulate $BN^{(t-3\sim t-6)}$ normalize BN update ScaleShift	update $W^{(t-2)}$ accumulate $BN^{(t-2\sim t-5)}$ normalize BN update ScaleShift	update $W^{(t-1)}$ accumulate $BN^{(t-1\sim t-4)}$ normalize BN update ScaleShift	update $W^{(t)}$ accumulate $BN^{(t\sim t-3)}$ normalize BN update ScaleShift

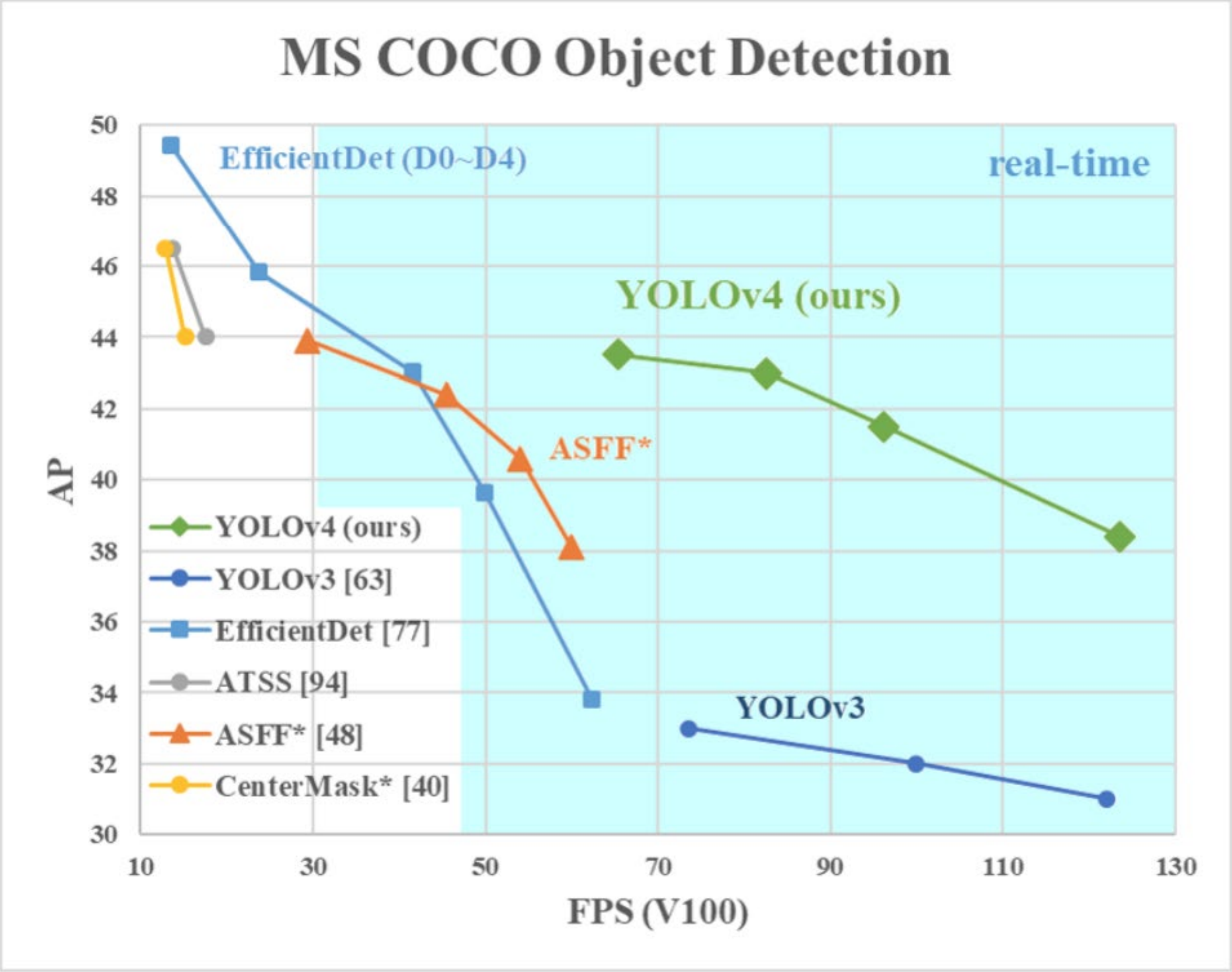
Lets:
Bias, scale – ScaleShift
Mean, variance – BN
Weights – W



CmBN – assume a batch contains four mini-batches

accumulate $W^{(t-3)}$ accumulate $BN^{(t-3)}$ normalize BN	accumulate $W^{(t-3\sim t-2)}$ accumulate $BN^{(t-3\sim t-2)}$ normalize BN	accumulate $W^{(t-3\sim t-1)}$ accumulate $BN^{(t-3\sim t-1)}$ normalize BN	accumulate $W^{(t-3\sim t)}$ accumulate $BN^{(t-3\sim t)}$ normalize BN update W, ScaleShift

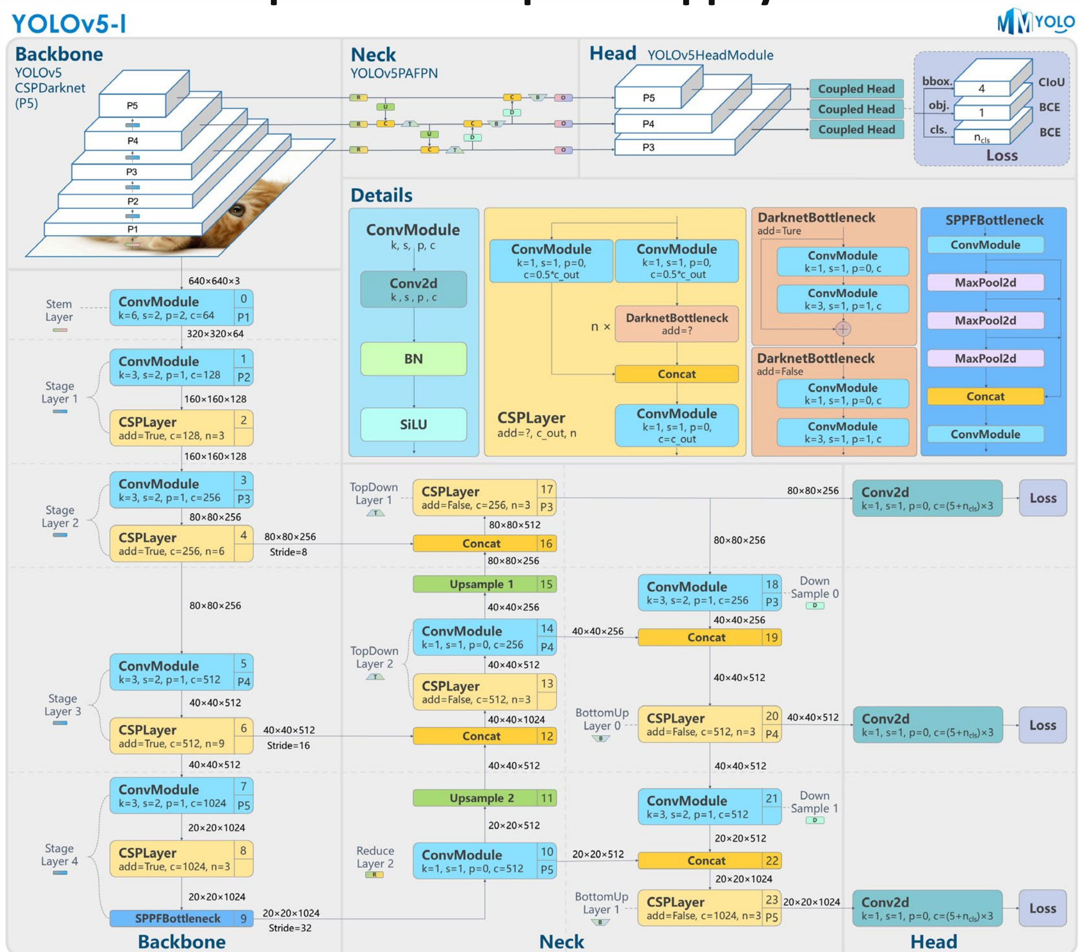
YOLOV4



MS COCO Object Detection
EfficientDet (D0~D4)
real-time
YOLOv4 (ours)
ASFF*
YOLOv3
AP
FPS (V100)
50
48
46
44
42
40
38
36
34
32
30
10
30
50
70
90
110
130
YOLOv4 (ours)
YOLOv3 [63]
EfficientDet [77]
ATSS [94]
ASFF* [48]
CenterMask* [40]

YOLOv5

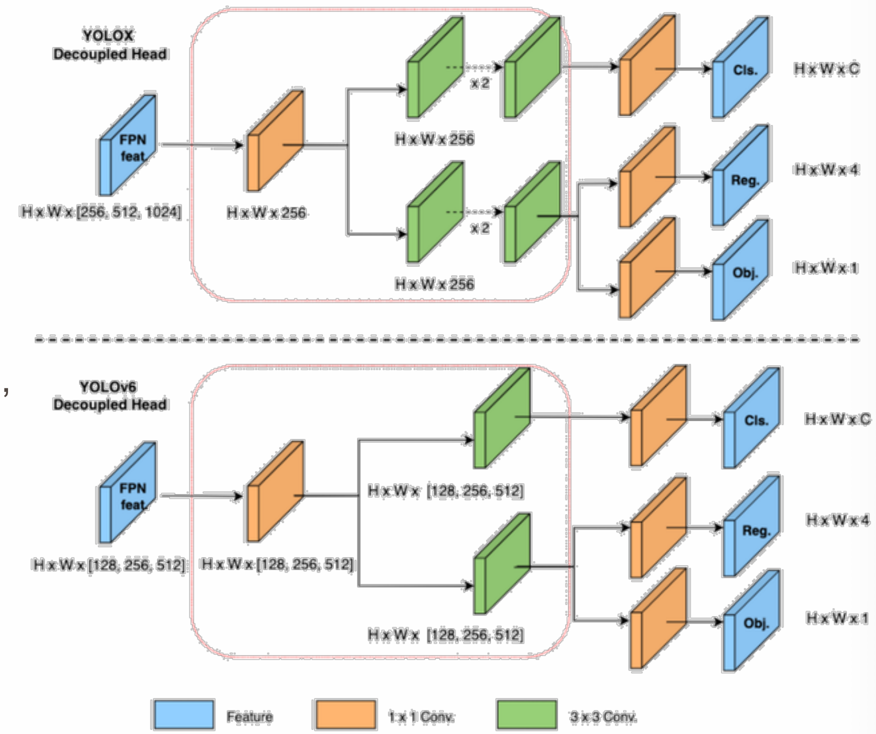
cspdarknet53+panet+spp+yolov3 head



Jocher, G. (2020). YOLOv5 by Ultralytics (Version 7.0) [Computer software]. https://doi.org/10.5281/zenodo.3908559

YOLOV6

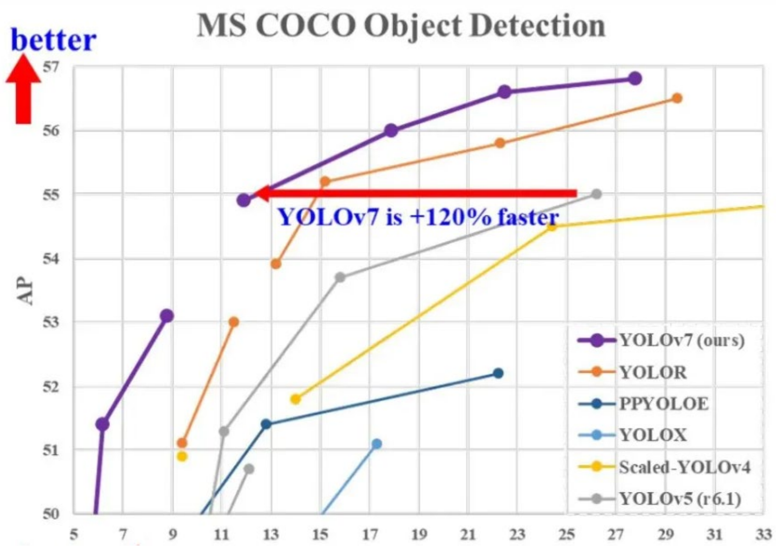
- Revamped the detector's Neck component: introduced Bi-directional Concatenation (BiC) for precise localization; simplified SPPF to SimCSPSPPF, trading minor speed loss for significant performance gain.
- Introduced Anchor-aided training (AAT) strategy, benefiting from both anchor-based and anchor-free design concepts without sacrificing inference efficiency.
- Deepened the Backbone and Neck of YOLOv6, achieving new state-of-the-art (SOTA) performance with high-resolution input.
- "Proposed a new self-distillation strategy to enhance the performance of YOLOv6 small models, using a larger DFL as an enhanced auxiliary regression branch during training.



https://github.com/meituan/YOLOv6

YOLOV7

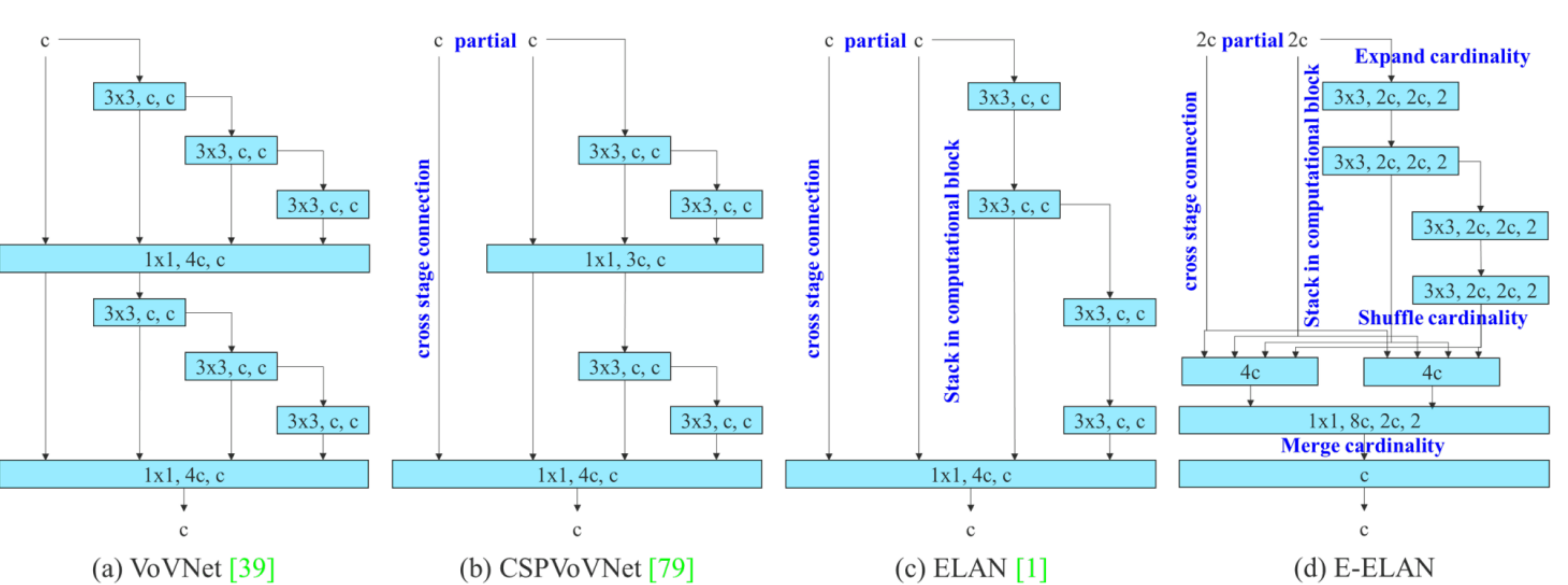
- New SOTA
  - Compared to YOLOV5



- Real-time object detection: necessary component in various computer vision systems
- Optimization of training process: introducing optimized modules and methods to improve detection accuracy without increasing inference cost.
- Model re-parameterization and dynamic label assignment: addressing new issues in network training and object detection

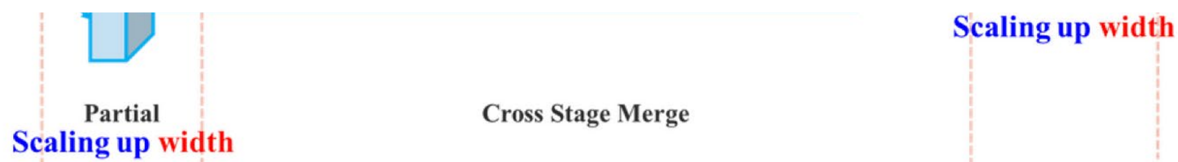
Wang, Chien-Yao, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors." *arXiv preprint arXiv:2207.02696* (2022).

YOLOV7 Extended efficient layer aggregation networks

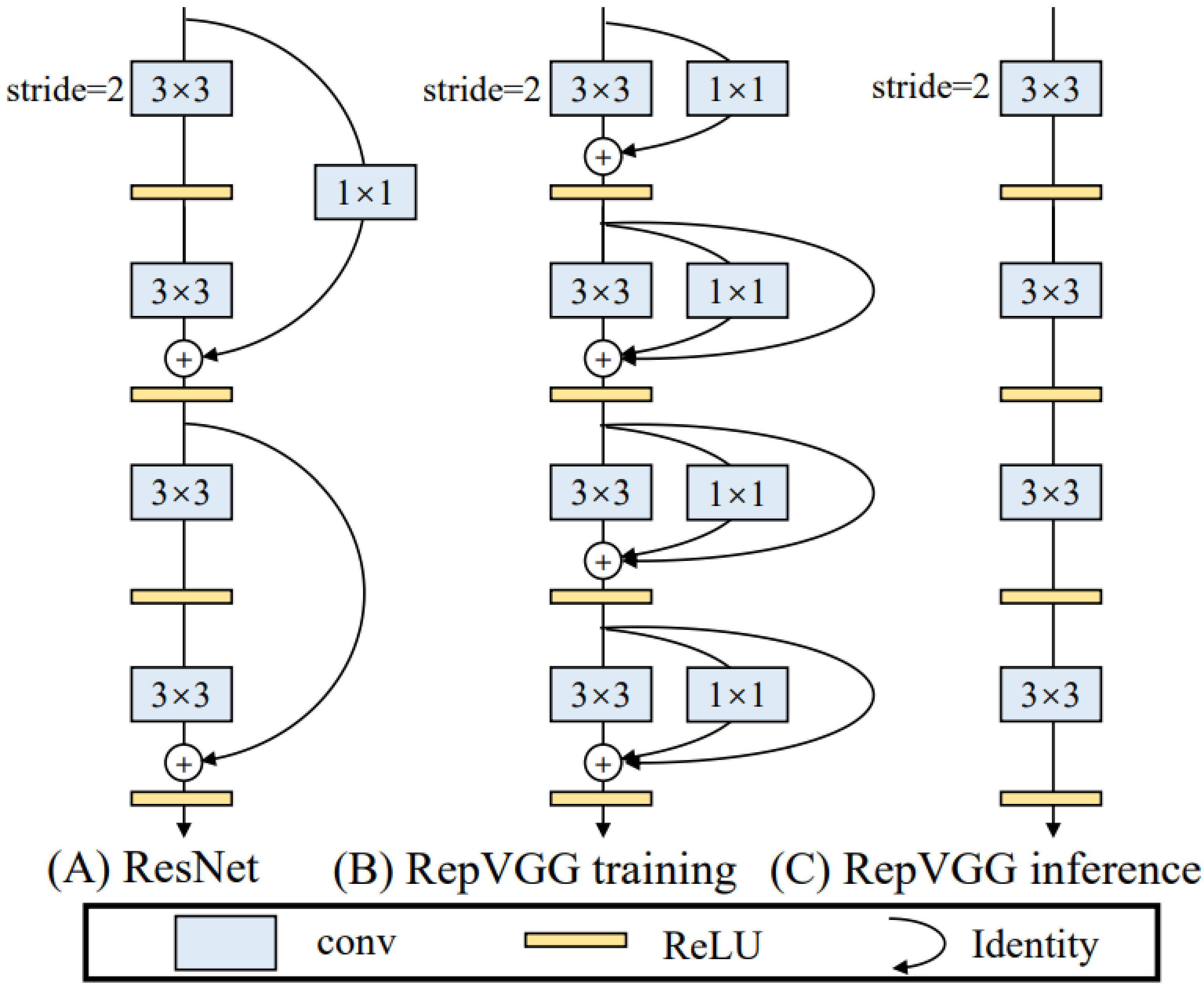


YOLOV7: Model Scaling

Model	#Param.	FLOPs	Size	AP^{val}	AP_{50}^{val}	AP_{75}^{val}
base (v7-X light)	47.0M	125.5G	640	51.7%	70.1%	56.0%
width only (1.25 w)	73.4M	195.5G	640	52.4%	70.9%	57.1%
depth only (2.0 d)	69.3M	187.6G	640	52.7%	70.8%	57.3%
compound (v7-X)	71.3M	189.9G	640	**52.9%**	**71.1%**	**57.5%**
improvement	-	-	-	+1.2	+1.0	+1.5

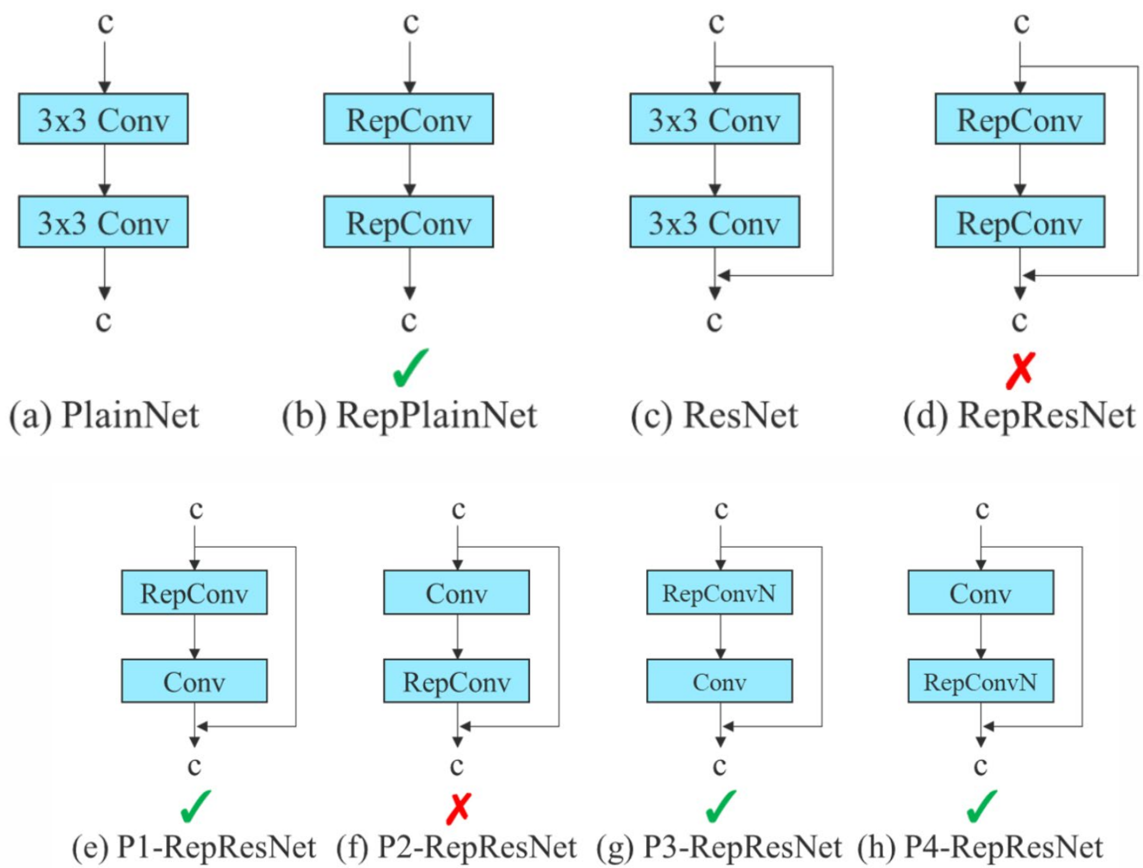


(c) compound scaling up depth and width for concatenation-based model


stride=2
3×3
1×1
3×3
3×3
3×3
stride=2
3×3
1×1
3×3
1×1
3×3
1×1
3×3
1×1
stride=2
3×3
3×3
3×3
3×3
(A) ResNet
(B) RepVGG training
(C) RepVGG inference
conv
ReLU
Identity

Re-parameterized

- How to determine the best combination



YOLOV7: Trainable Bag-of-freebies

- Dynamic label assignment strategy:
  - Deep supervision improves model performance by training shallow layers with auxiliary heads.
  - YOLO assigns soft labels based on predicted bounding boxes and ground truth IoU, in addition to hard labels.
  - The lead head is the final output, while the auxiliary head is used for auxiliary training.
  - The author proposes a lead head-guided assigner to generate hierarchical soft labels for auxiliary and lead head learning.
  - Coarse and fine labels are generated to optimize auxiliary head recall while maintaining high precision.
  - The decoder restricts the generation of soft labels to prevent prior biases.
  - The proposed strategy improves accuracy across different AP standards.

YOLOV7: Trainable Bag-of-freebies

- Other trainable bag-of-freebies:
  - Batch normalization, implicit knowledge in YOLOR, and Exponential Moving Average (EMA) are mentioned as additional trainable techniques.

YOLOv8

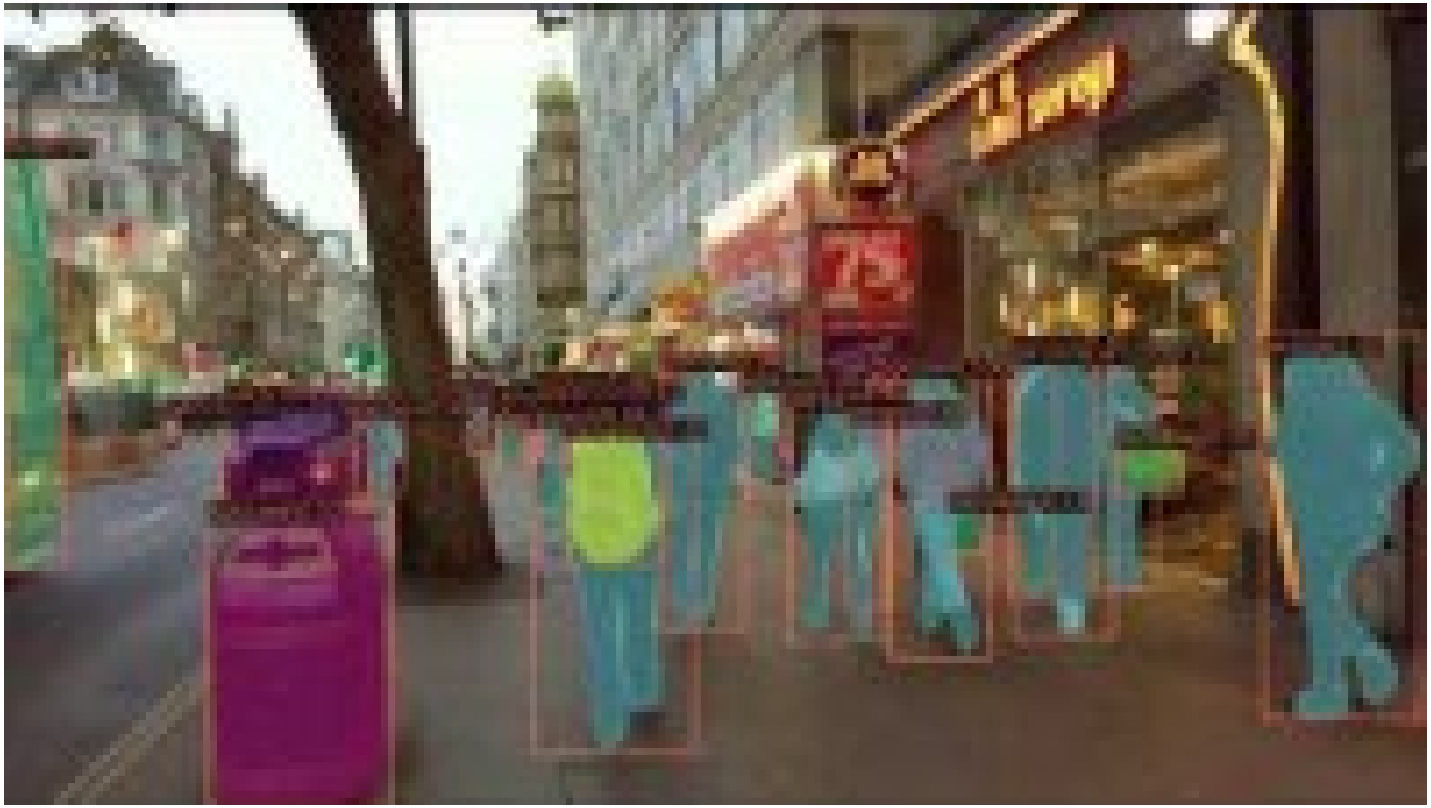
- YOLOv8: Latest iteration of the YOLO model, with increased performance.
- Created and maintained by Ultralytics.
- Features: User-friendly API, faster and more accurate model, supports object detection, instance segmentation, and image classification tasks.
- Compatibility: Compatible with all previous versions of YOLO.
- Improvements: New SOTA model, revised backbone and neck sections, major modifications to the head section, use of TaskAlignedAssigner and Distribution Focal Loss.
- Anchor-Free detection: Breaks away from the traditional anchor-based approach.
- New loss function: Uses a novel loss function for training.
- Test results: Significant accuracy improvement over YOLOv5, though with increased model parameter count and FLOPs.
- Training strategy: Similar to YOLOv5 but with an increased number of epochs from 300 to 500, leading to longer training time.

SWIN-L (SWIN TRANSFORMER)

Swin-L Object Detector (SOTA, 2021/5)

Current SOTA (2022/5): DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection

Still, Swin-Transformer-based approach



DATASET FOR OBJECT DETECTION

COCO and VOC 2007/2012

- VOC (Visual Object Classes)
  - PASCAL_VOC07
  - VOCdevkit
    - VOC2007
      - ├── Annotations
      - ├── ImageSets
      - │ └── Main
      - └── JPEGImages

XML formatted

File list (in txt)

Image files

COCO dataset

- Real large-scale image dataset
  - Annotation is formatted in “json”
  - Providing COCO API for data retrieval and processing
  - annotations_train_valid
    - Six text-formatted files
      - **object instances including bbox and polygon annotations**
      - person keypoint
      - image captions
  - For example: instances_train2014.json
    - Better than XML
    - Python has simple but powerful lib for json parsing