DNN for Object Detection and Segmentation

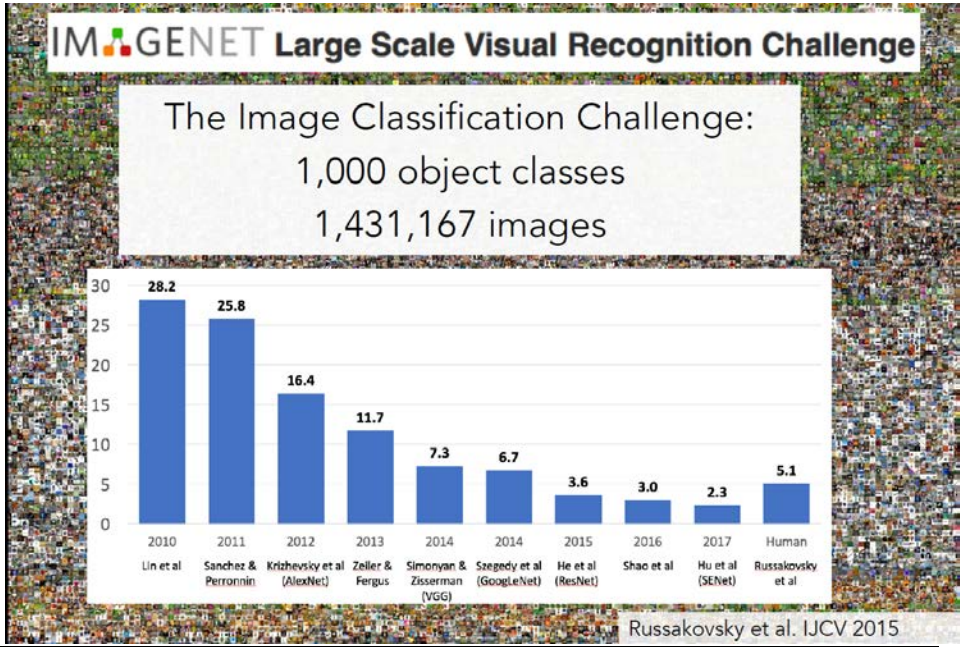
NATIONAL SUN YAT-SEN UNIVERSITY

Deep Learning Application Examples

- classification (top-3)
  - https://www.youtube.com/watch?v=qrzQ_AB1DZk
- instance segmentation and pose estimation
  - https://www.youtube.com/watch?v=OAWCp7OXLnY
- dense human pose estimation
  - https://www.youtube.com/watch?v=Dhkd_bAwwMc
- deep robot learning
  - https://www.youtube.com/watch?v=2hGngG64dNM

Image Classification in ILSVRC

- ImageNet dataset
  - 140M images
  - 22K categories
- ILSVRC (Image Large Scale Visual Recognition Challenge (ILSVRC)
  - 1000 object categories
  - top-1 and top-5 error rates



This image is CC0 public domain

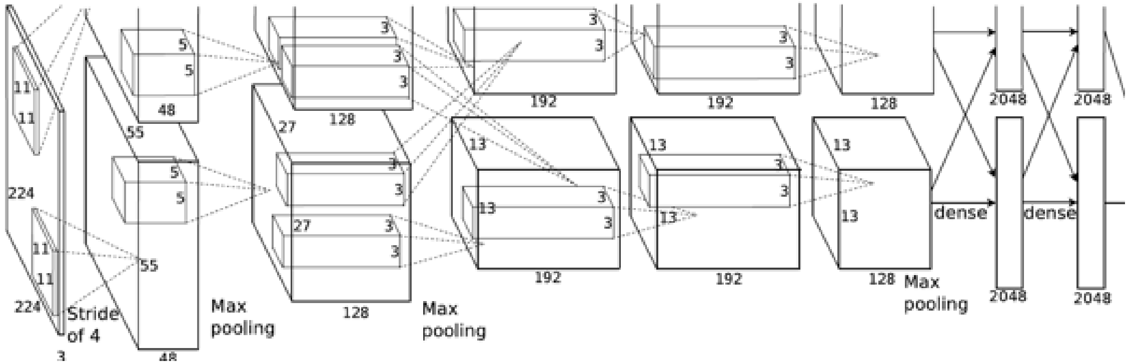


Figure copyright Alex Krizhevsky, ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission

Vectors: 4096

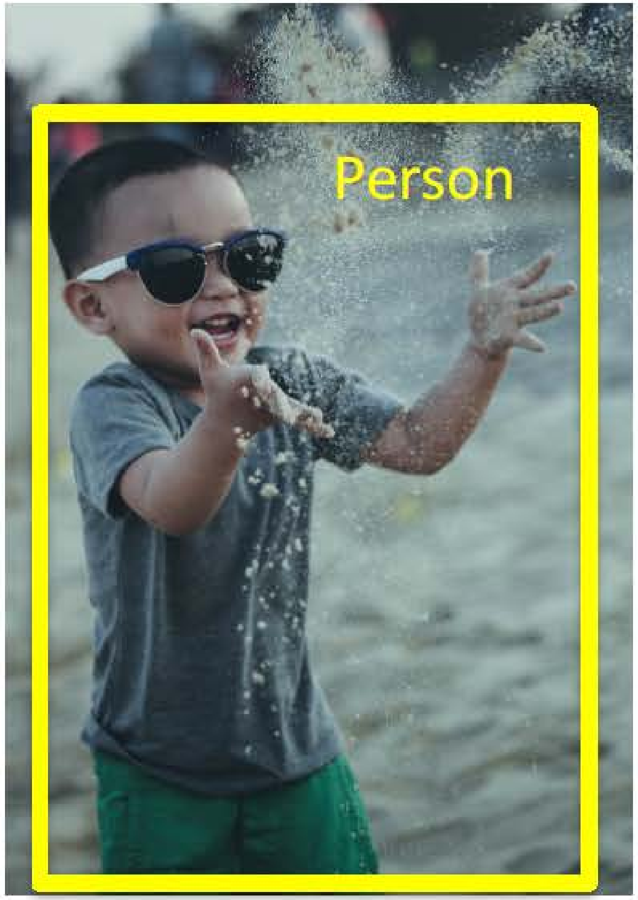
Fully-Connected: 4096 to 1000

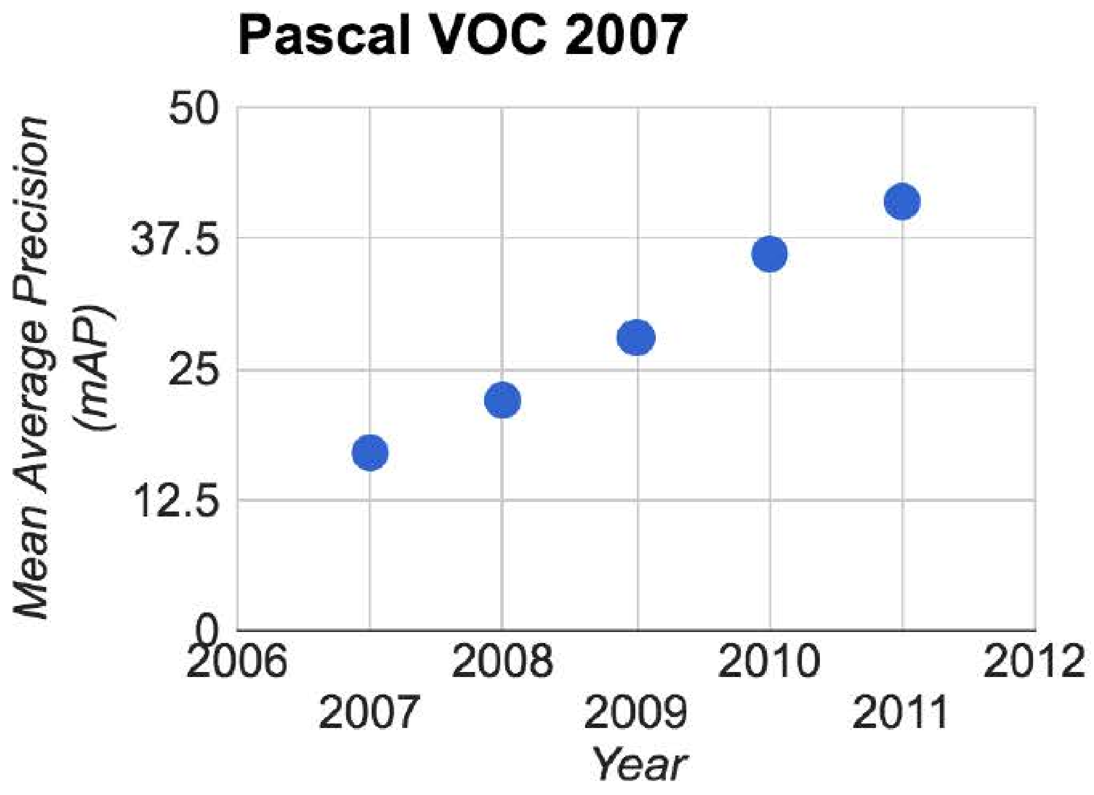
Class Scores

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

Object Detection in PASCAL VOC

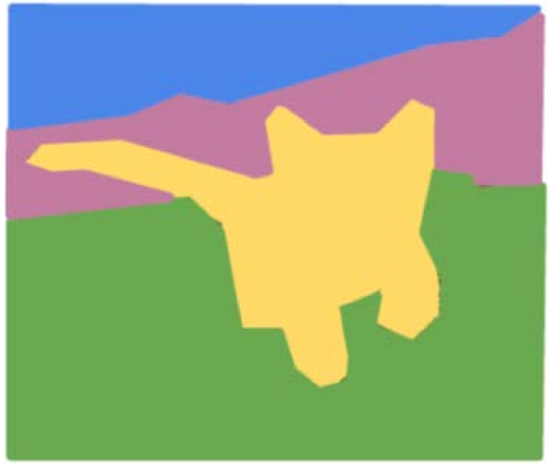
◇ PASCAL Visual Object Challenge (VOC): 20 categories)





Detection and Segmentation

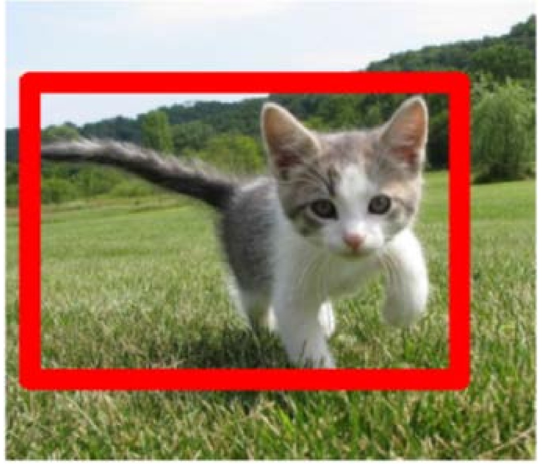
Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

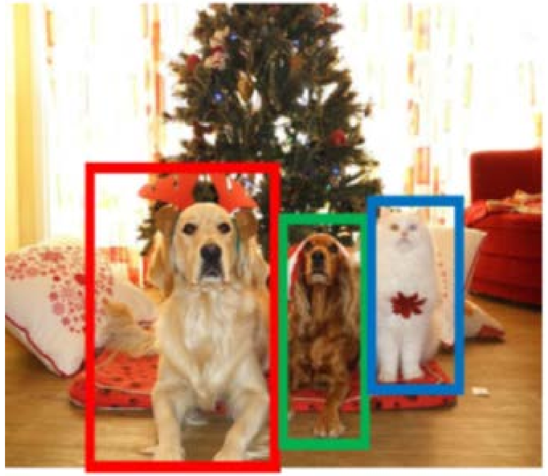
Classification +Localization



CAT

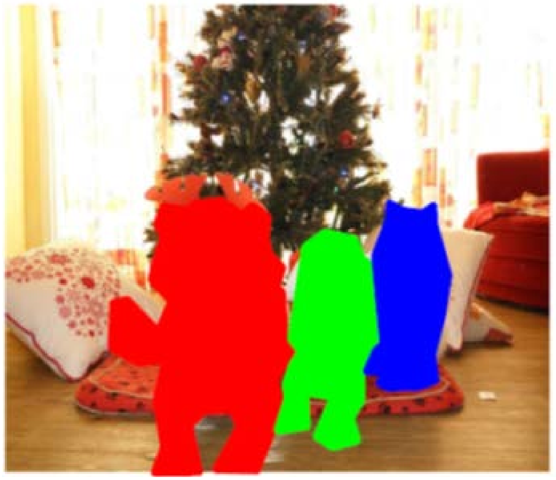
Single Object

Object Detection



DOG, DOG, CAT

Instance Segmentation

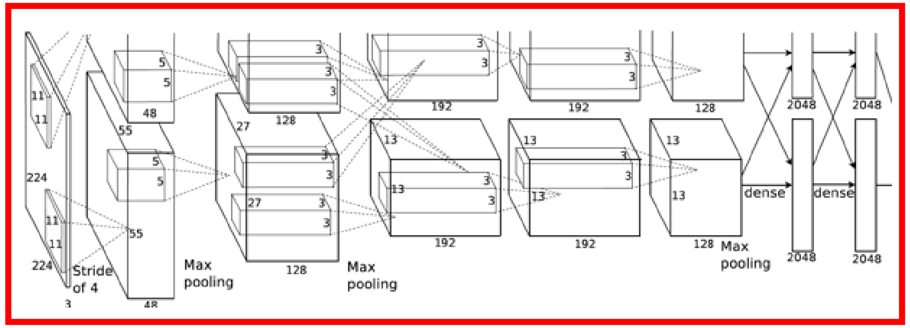


DOG, DOG, CAT

Multiple Object

Classification + Localization


This image is CC0 public domain



Often pretrained on ImageNet
(Transfer learning)

Vectors: 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)

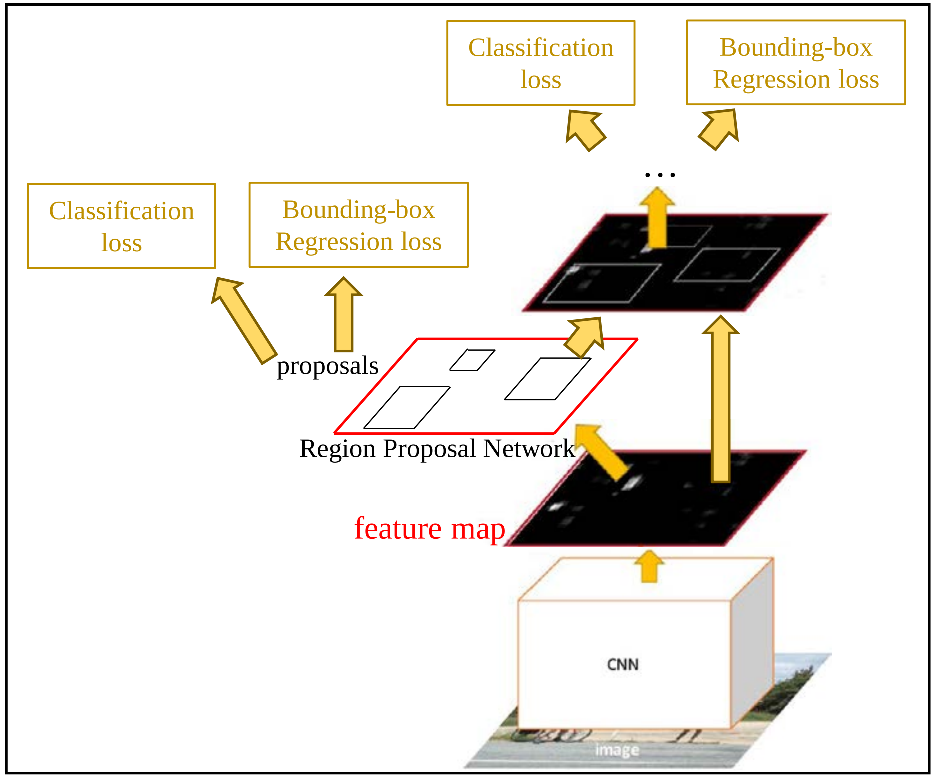
L2 Loss

Correct box (x’, y’, w’, h’)

+ **Loss**

Object Detection: two-stage vs. one-stage

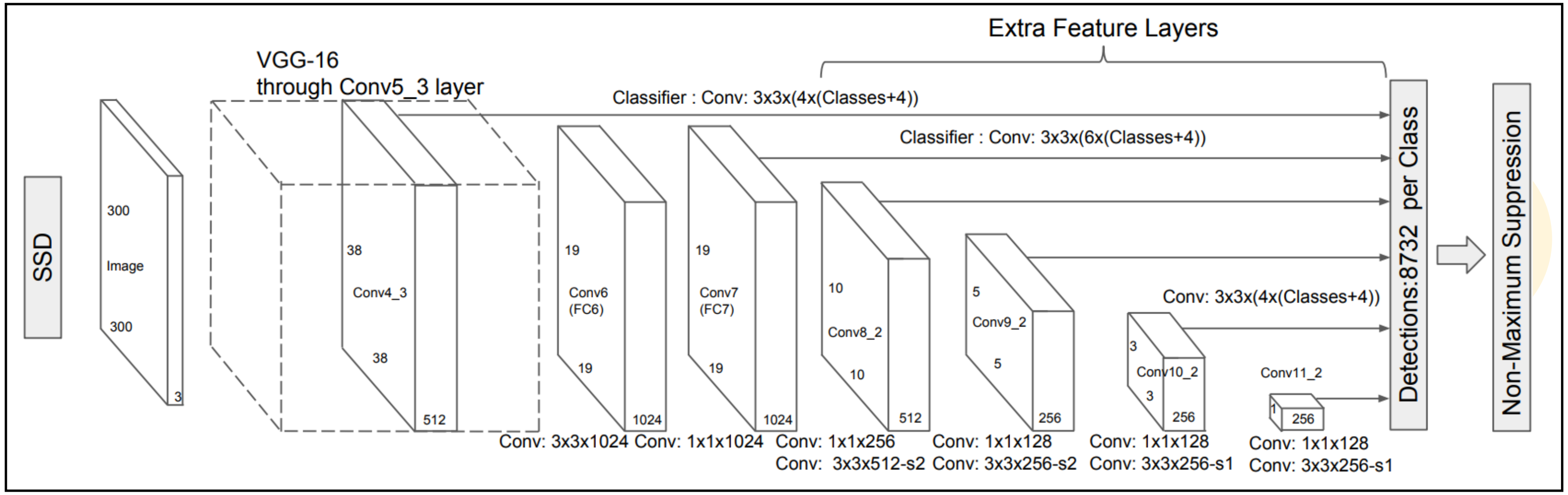
Two-Stage Detector: Faster RCNN (Region CNN)



影像特徵擷取 和 目標物件後選區 為兩個獨立網路
→運算量龐大
→運算速度不佳，無法達到即時的物件偵測

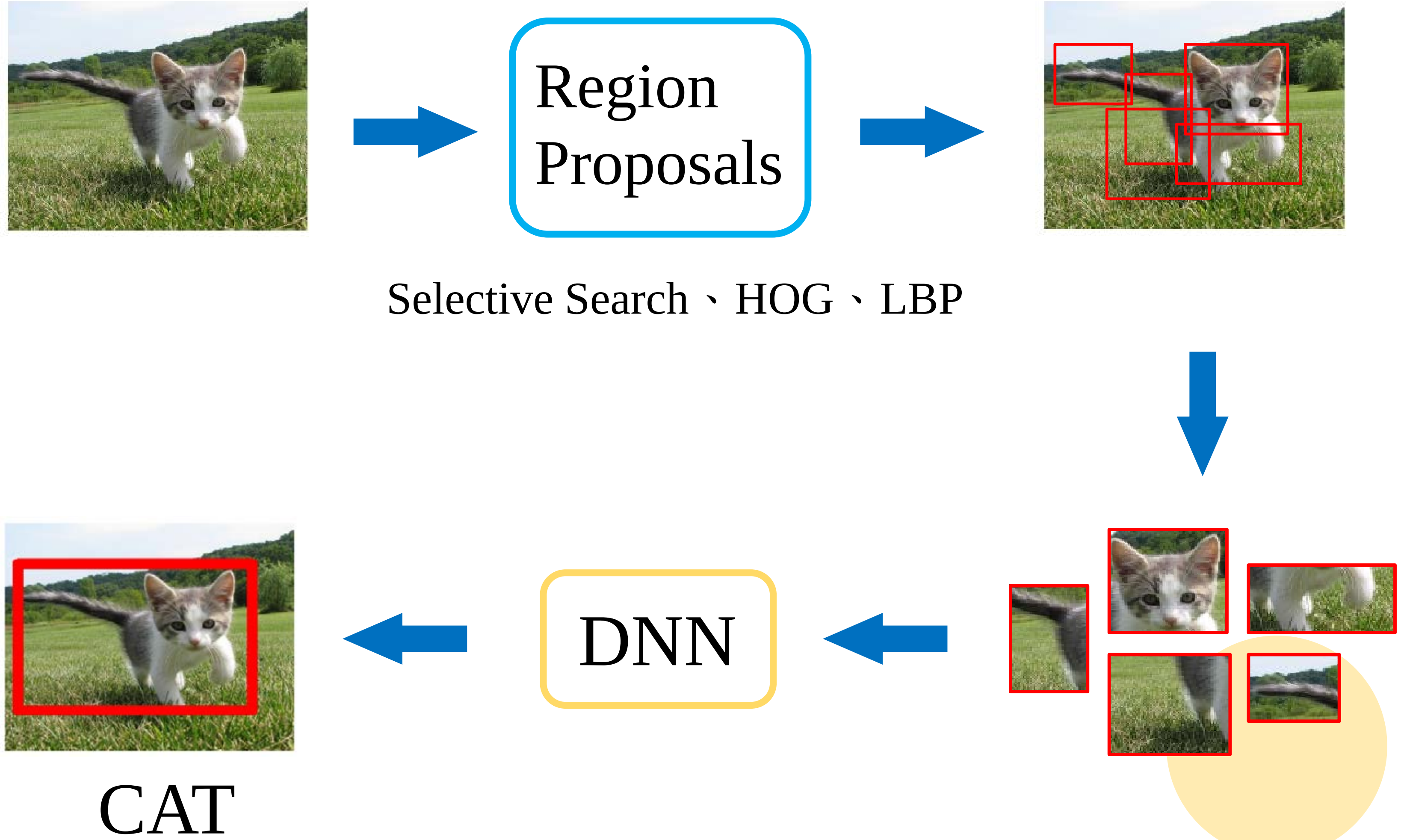
One-Stage Detector: Single Shot Detector (SSD)

→運算速度提升
→可達到即時的物件偵測



Two-stage Detector(1/2)

R-CNN：Regions with CNN features [1]

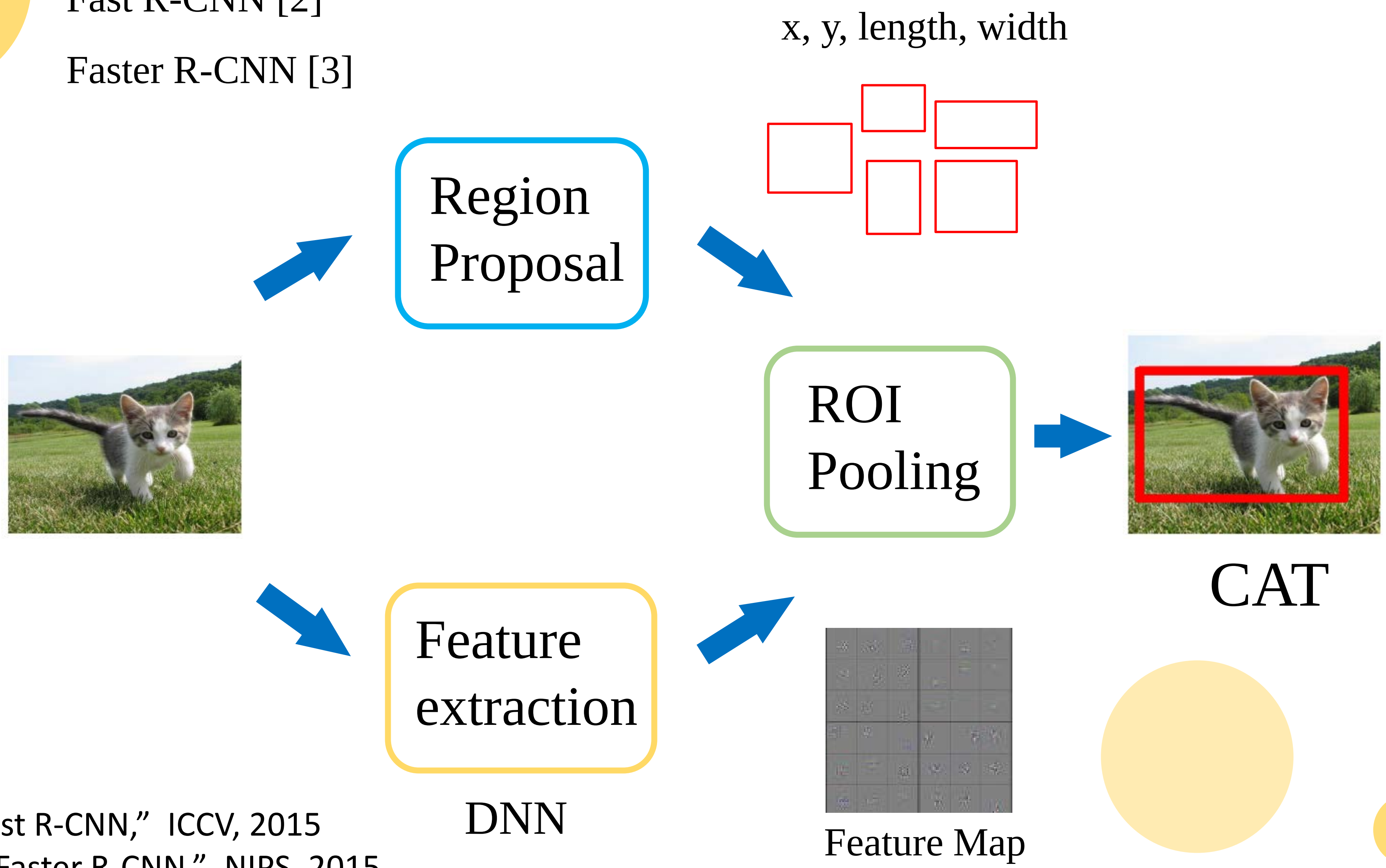


[1] R. Girshick, et al., “Rich feature hierarchies for accurate object detection and semantic segmentation,” CVPR, 2014

Two-stage Detector(2/2)

Fast R-CNN [2]

Faster R-CNN [3]



[2] R. Girshick, “Fast R-CNN,” ICCV, 2015
[3] S. Ren, et al., “Faster R-CNN,” NIPS, 2015

[4] W. Liu, et al., “SD: Single Shot MultiBox Detector,” ICCV, 2016
[5] J. Redmon, et al., “You Only Look Once:Unified, Real-Time Object Detection,” CVPR, 2016
[6] J. Redmon, et al., YOLO9000: Better, Faster, Stronger“,” CVPR, 2017
[7] J. Redmon, et al., “, YOLOv3: An Incremental Improvement,” 2018.

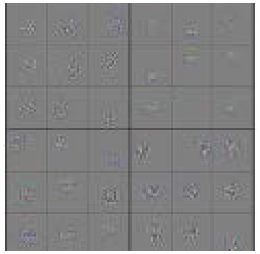
One-stage Detector

SSD：Single Shot Multibox Detector[4]

YOLO：You Only Look Once [5][6][7]

Feature extraction

提取影像特徵

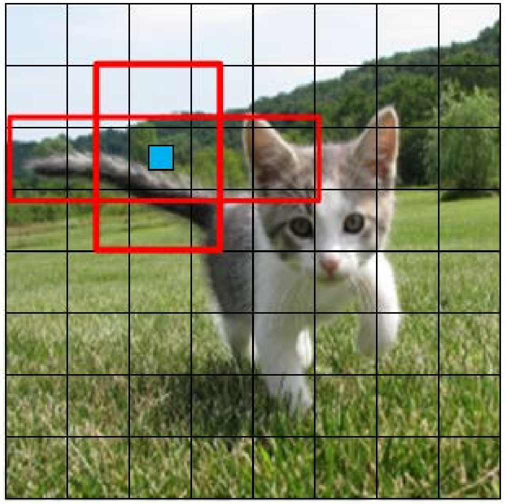


Feature Map

DNN

Detection

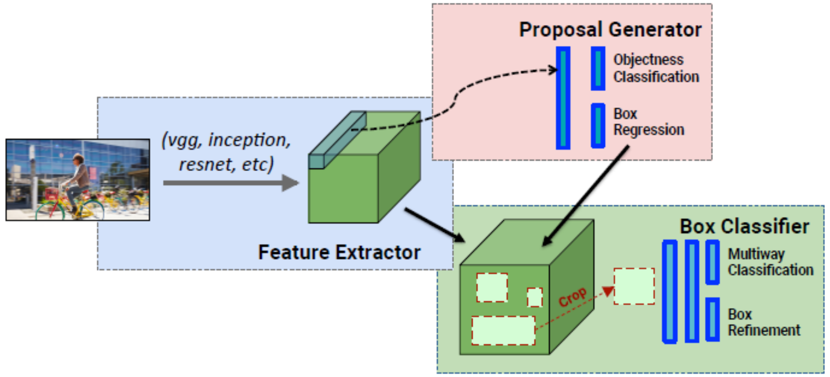
預測目標物件位置、種類



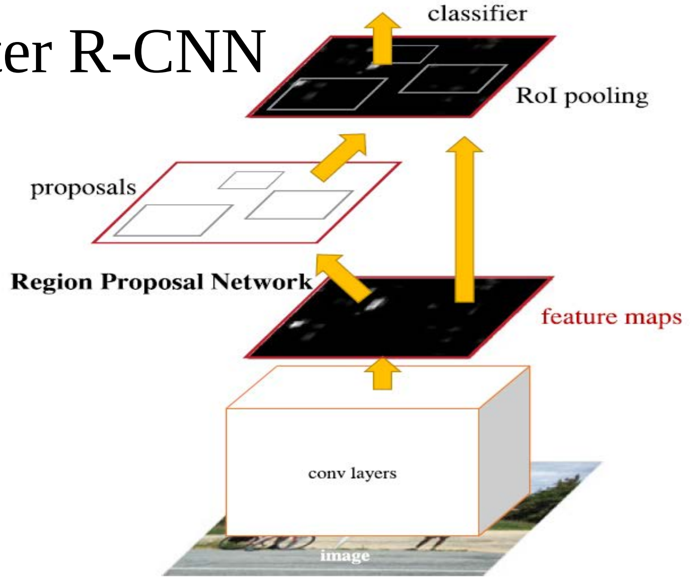
CAT

Object Detection Algorithm

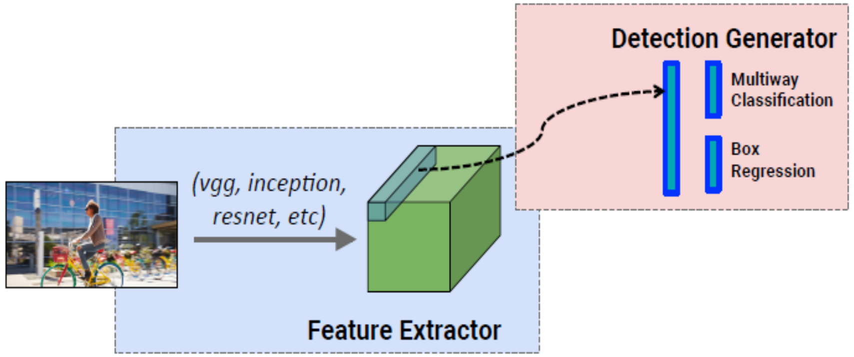
Two-stage Detector



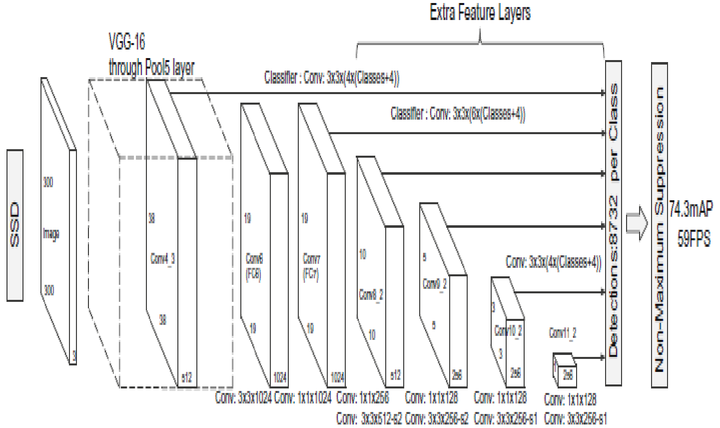
Faster R-CNN



One-stage Detector

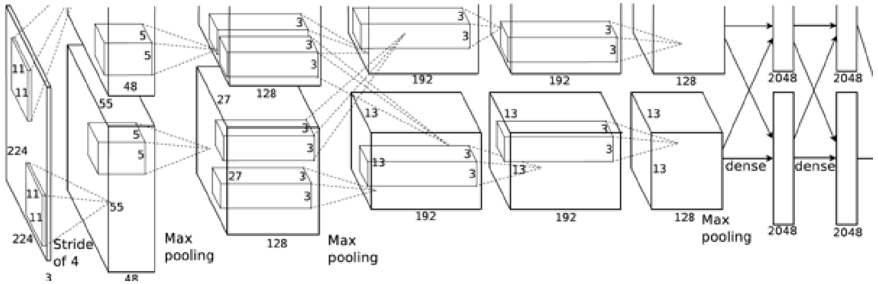


SSD

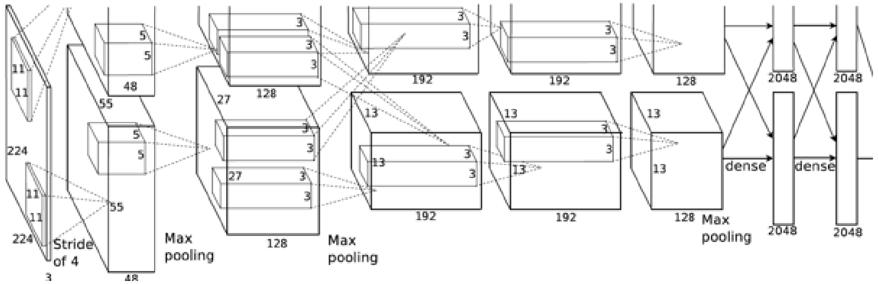


Object Detection as Regression ?

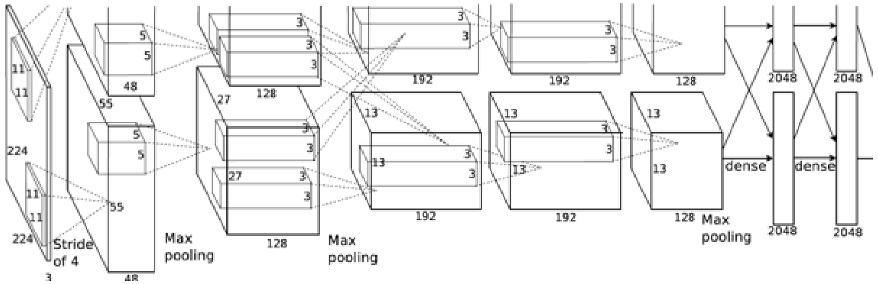
Each image needs a different number of outputs !



CAT: (x, y, w, h)   4 numbers



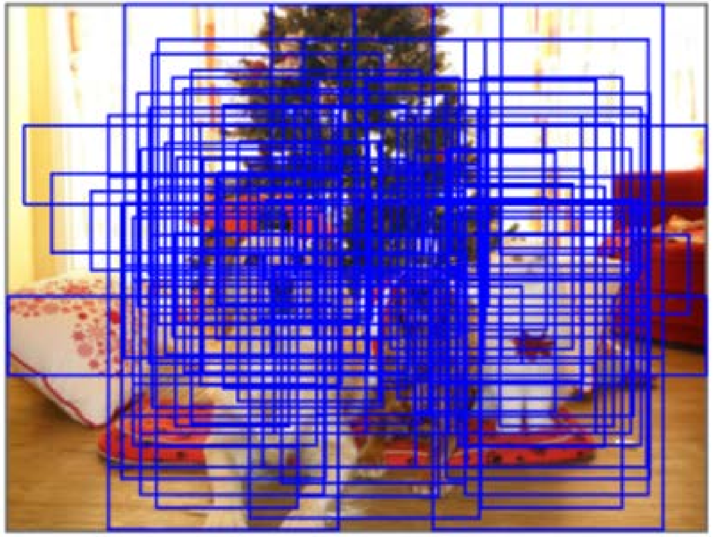
DOG: (x, y, w, h)
DOG: (x, y, w, h)   16 numbers
CAT : (x, y, w, h)

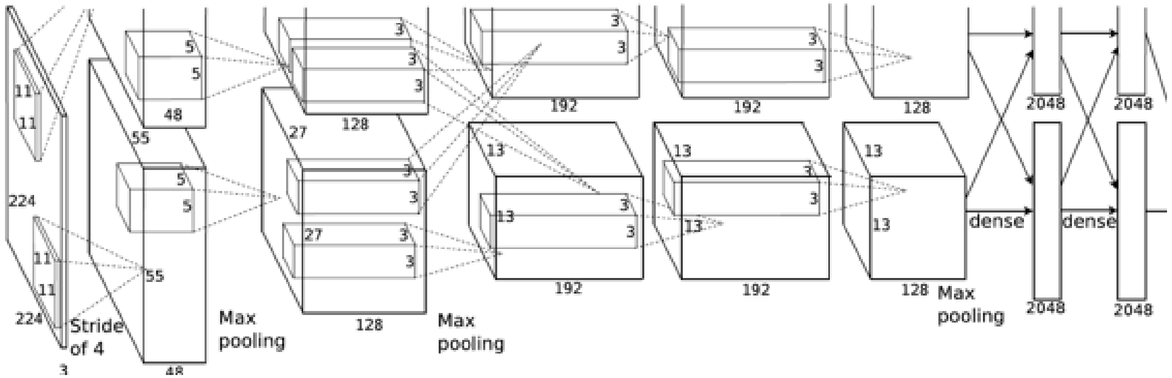


DUCK: (x, y, w, h)   Many numbers !
DUCK: (x, y, w, h)
...

Object Detection as Classification: Sliding Window

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



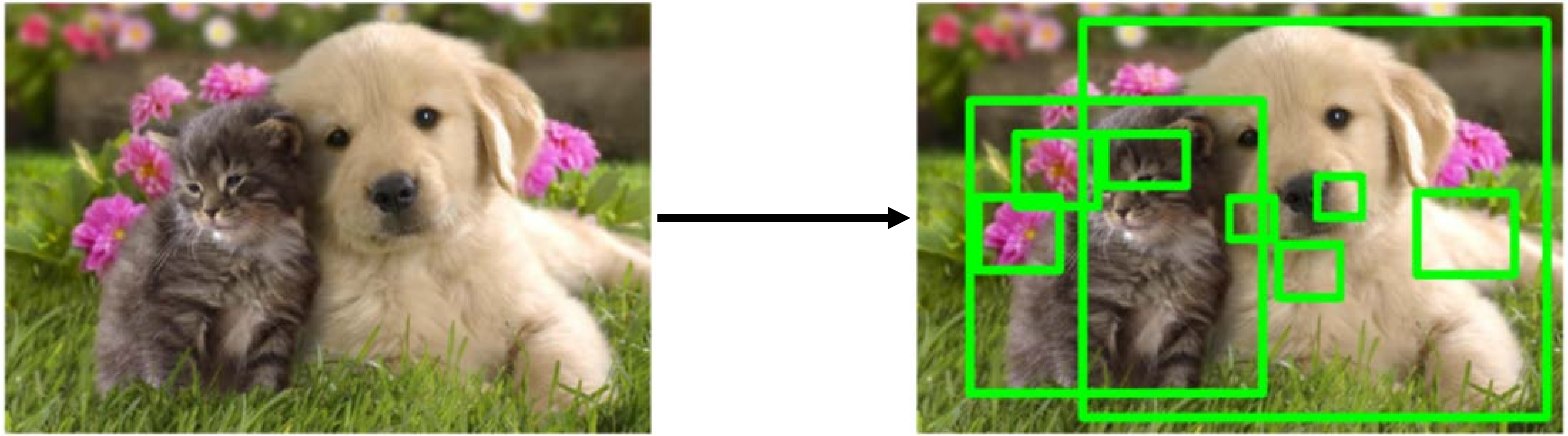


Dog? NO
Cat? YES
Background? NO

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”, FCCV 2014

R-CNN

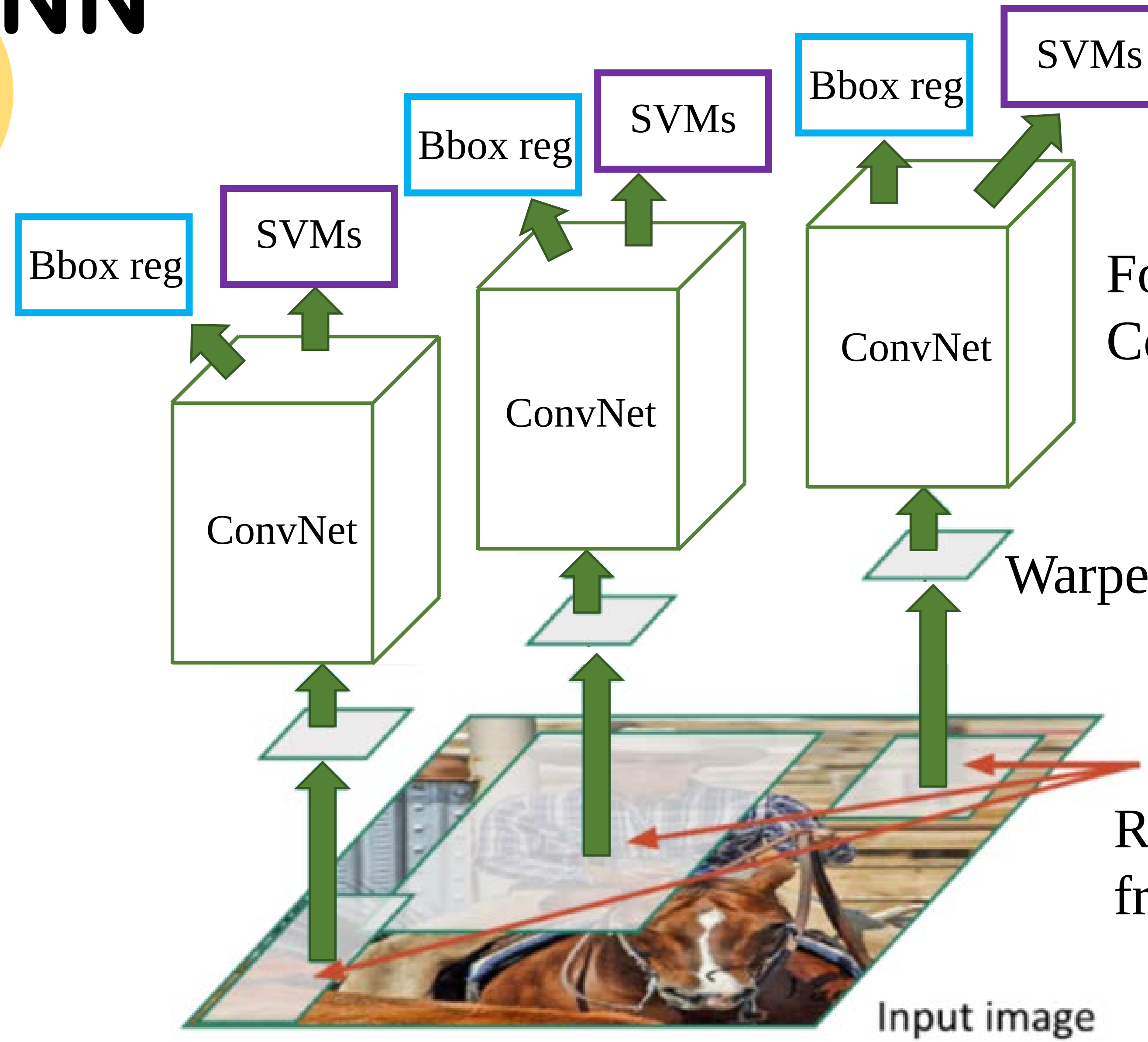
Linear Regression for bounding box offsets

Classify regions with SVMs

Forward each region through ConvNet

Warped image regions

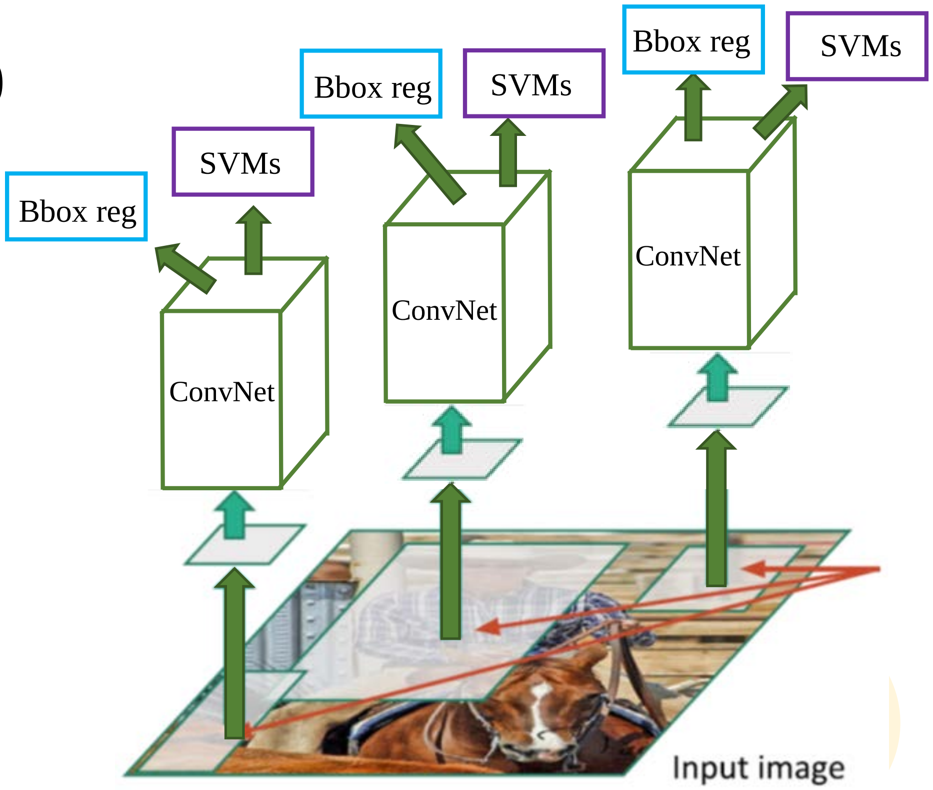
Regions of Interest(ROI) from a proposal method (~2k)



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: Problems

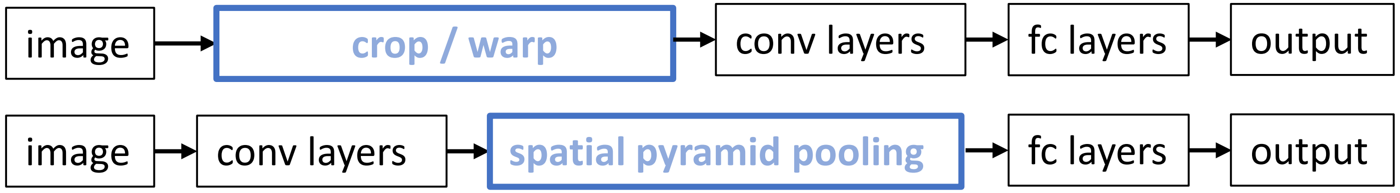
- ➢ Ad hoc training objectives
  - Fine-tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressions (least squares)
- ➢ Training is slow (84h), takes a lot of disk space
- ➢ Inference (detection) is slow
  - 47s / image with VGG16
  - Fixed by SPP-net [He et al. ECCV14]



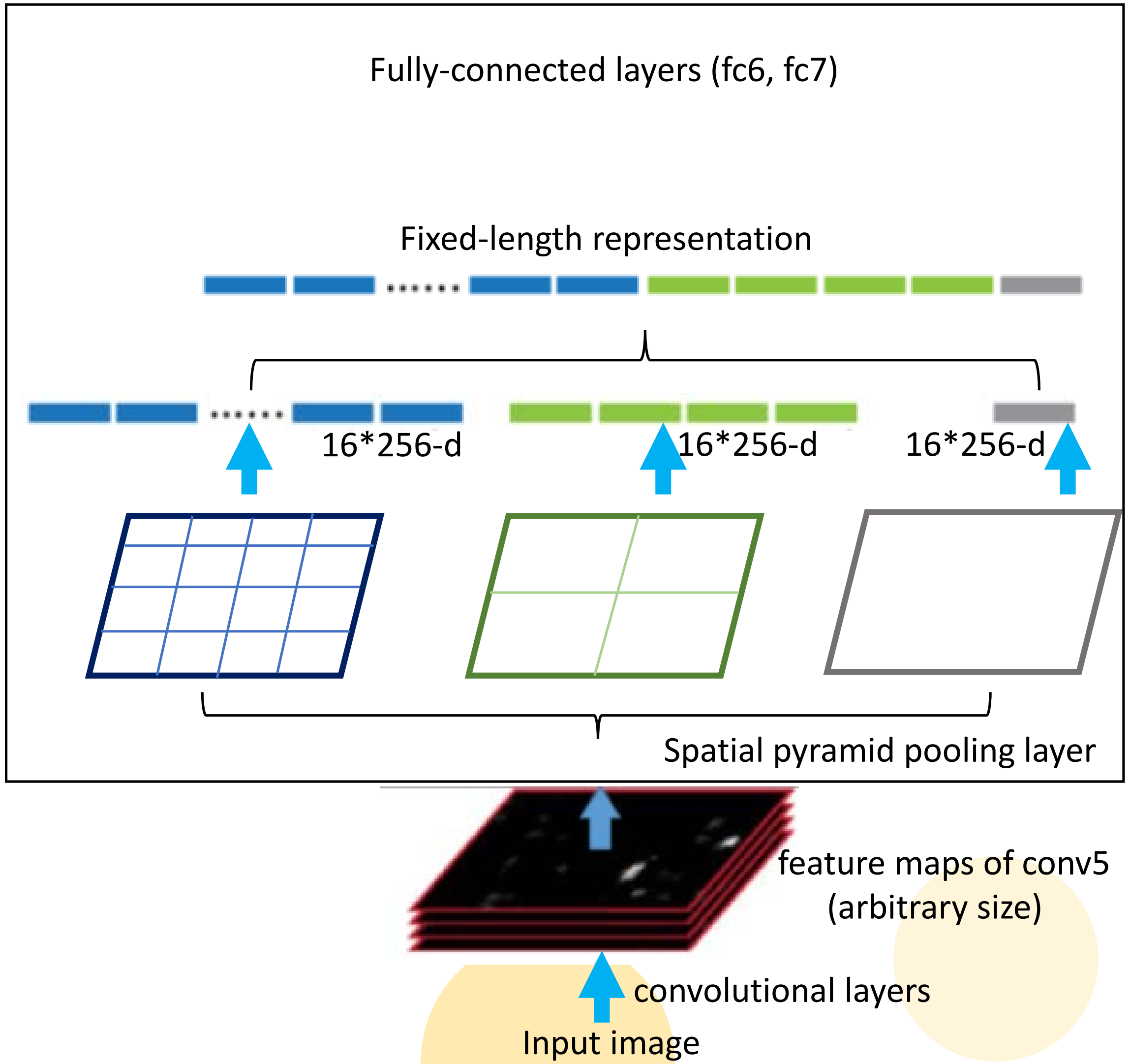
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

SPP-Net (Spatial Pyramid Pooling)

Cropping or warping to fit a fixed size



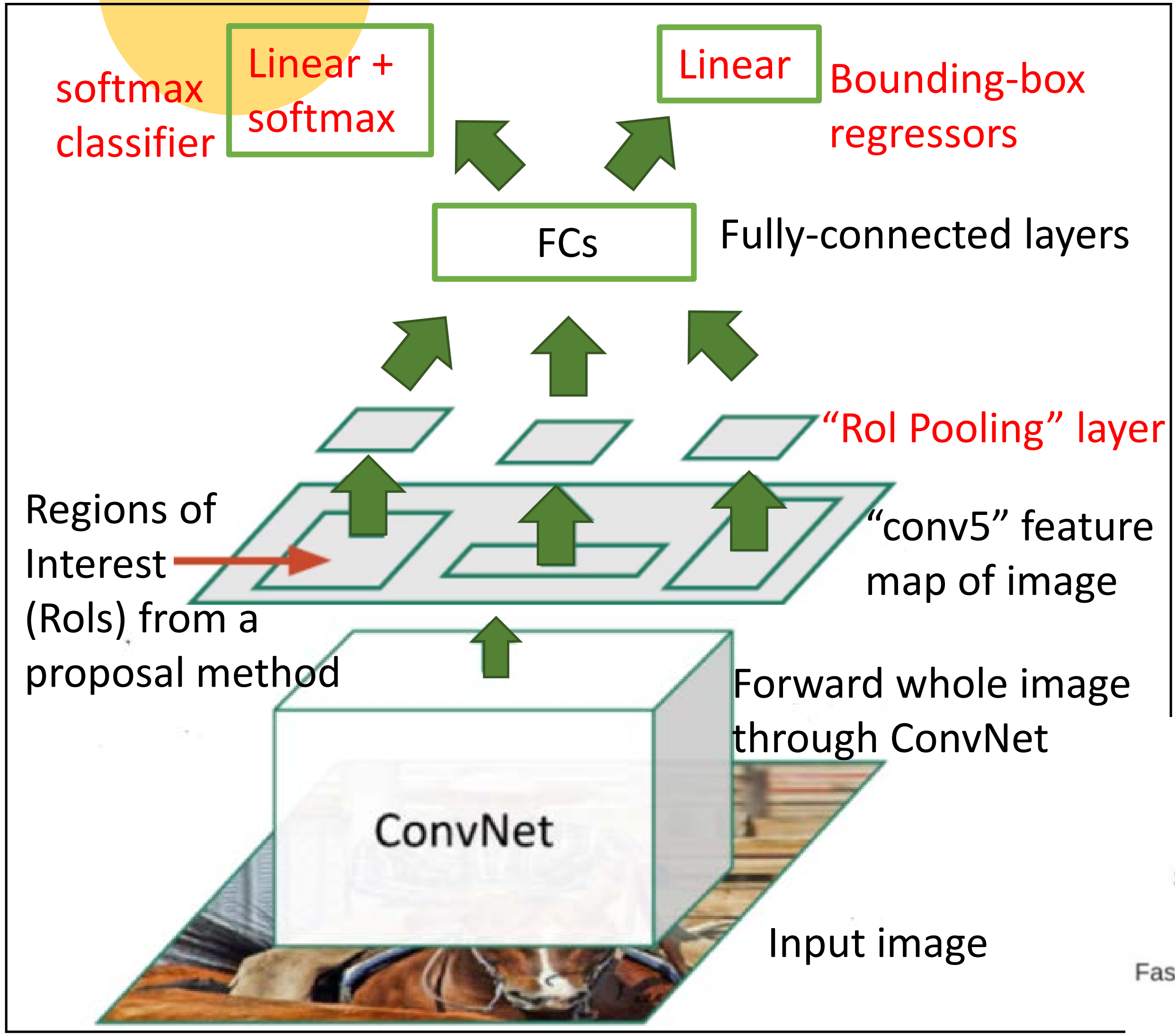
A conventional CNN vs.
spatial pyramid pooling network structure.

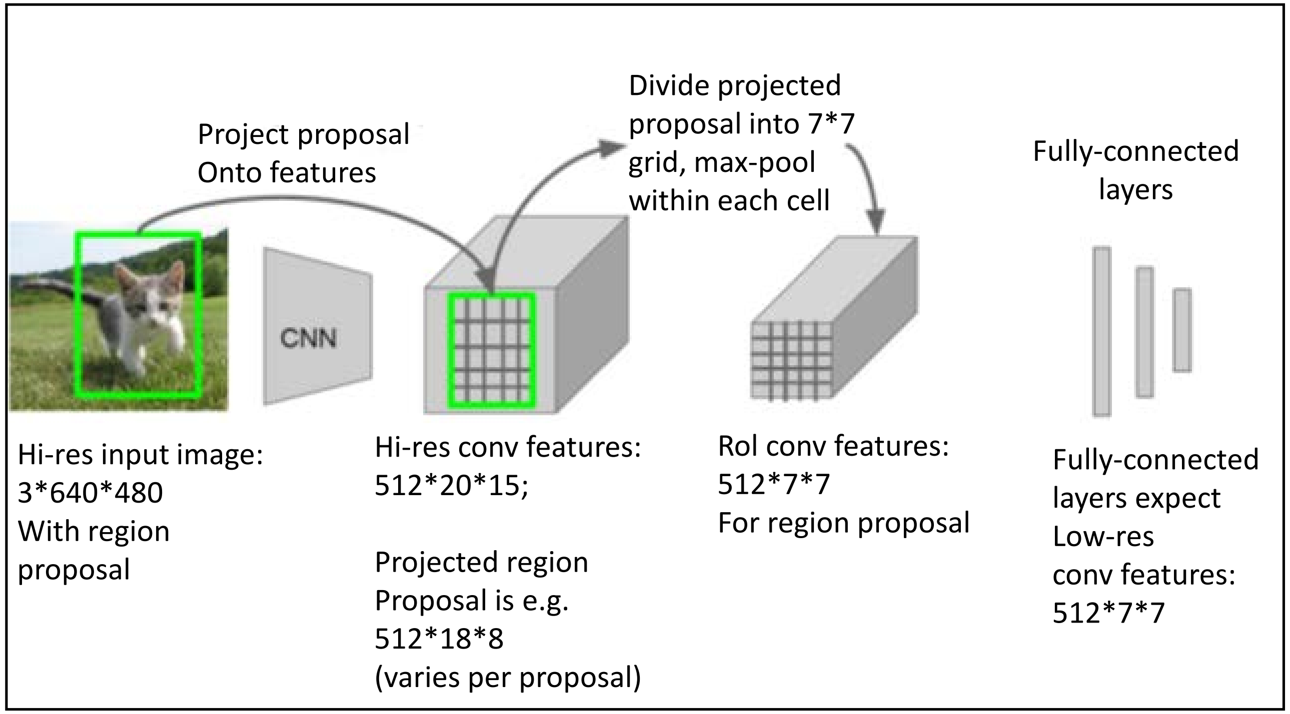


2015_Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition (TPAMI)

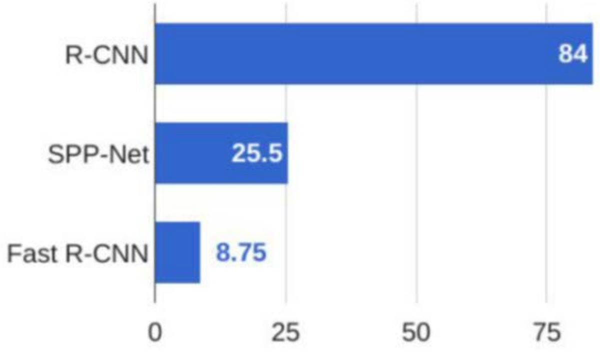
Fast R-CNN

◇ ROI Pooling

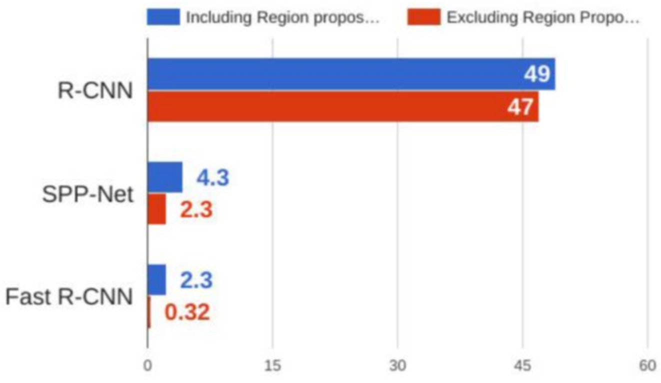




Training time (Hours)



Test time (seconds)



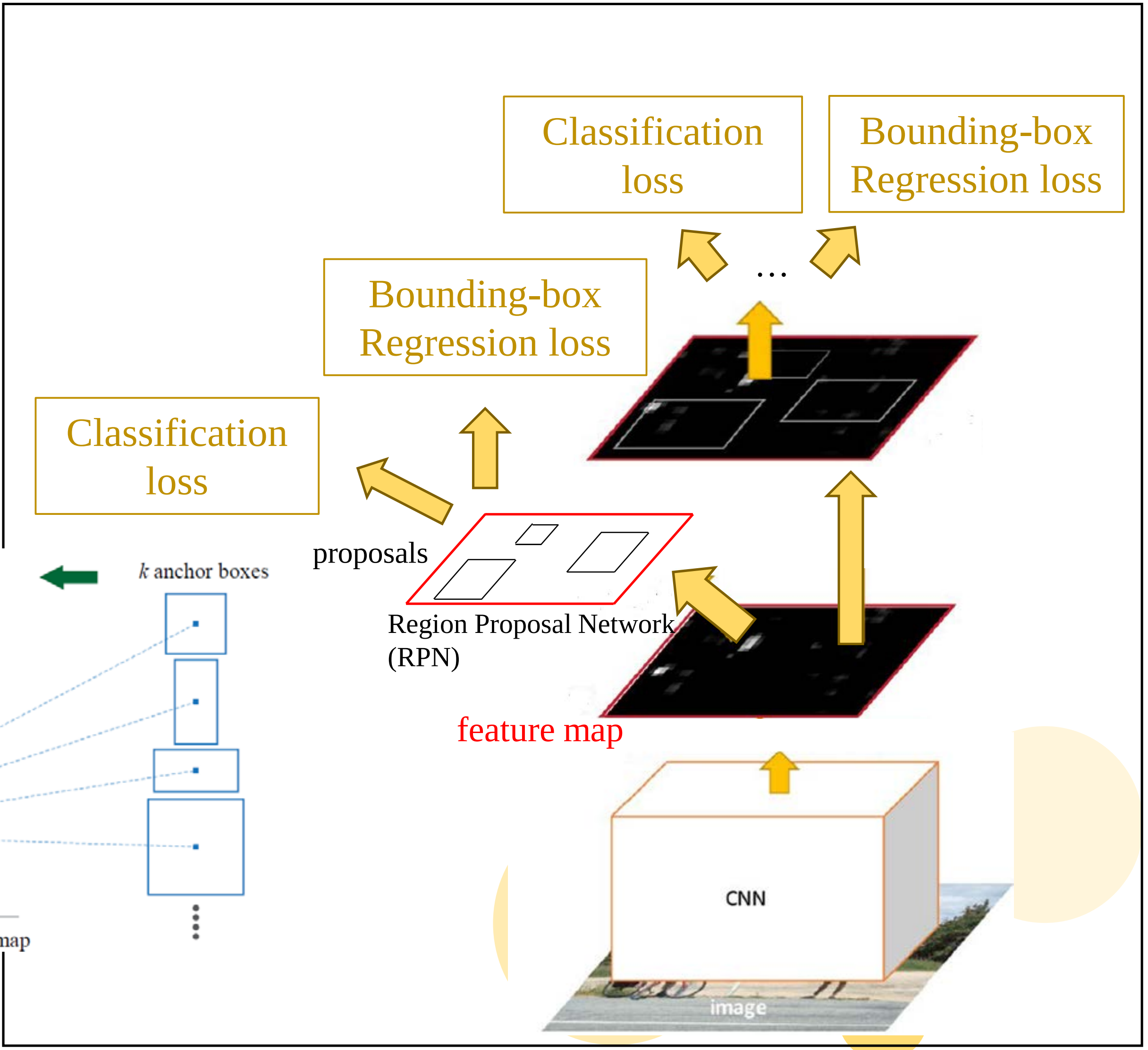
Girshick, "Fast R-CNN", ICCV 2015.

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

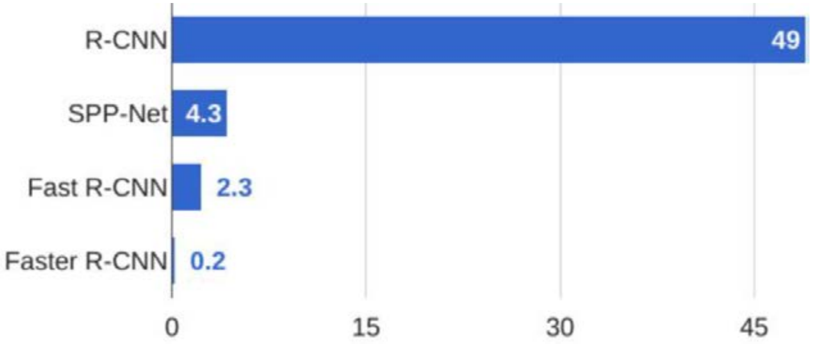
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

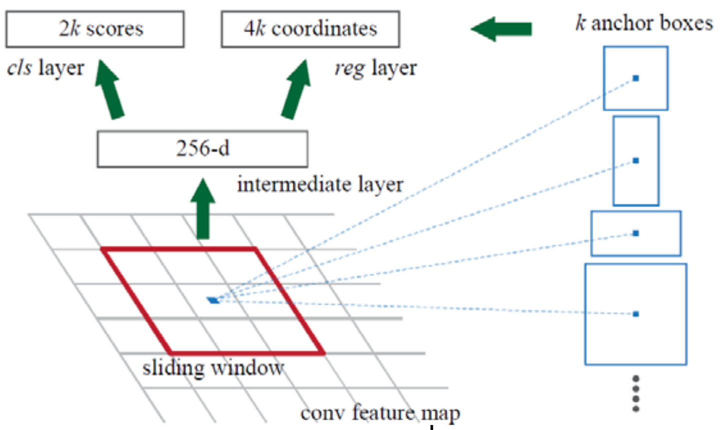
Faster R-CNN

- Make CNN do proposals!
- Insert Region Proposal Network (RPN) to predict proposals from features
- 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



Test-Time Speed





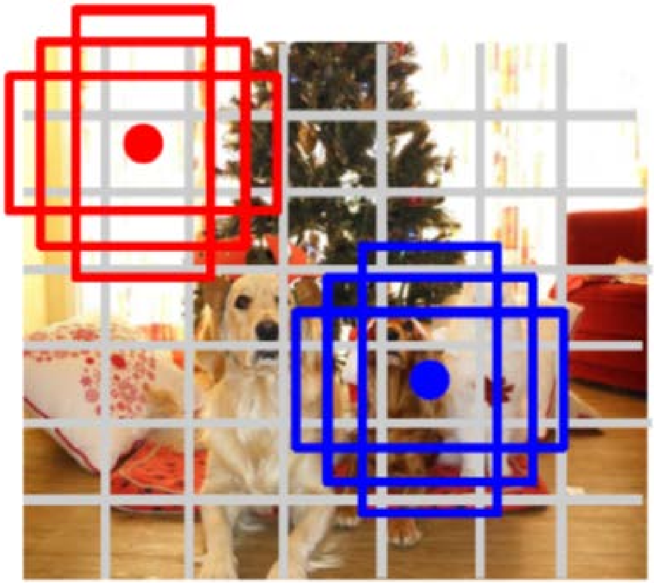
RPN

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

Detection without Proposals: YOLO / SSD

Go from input image to tensor of scores with one big convolutions network !

Input image
3*H*W



Divide image into grid
7*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)

Output:
7*7*(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

SSD (Single Shot Detection)

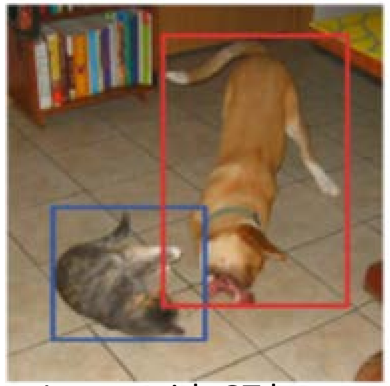
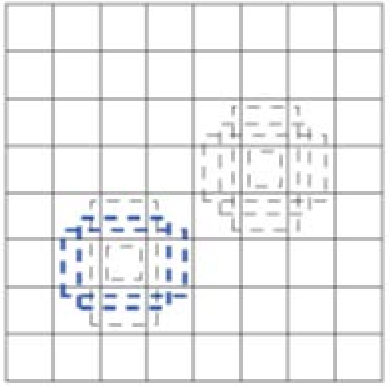
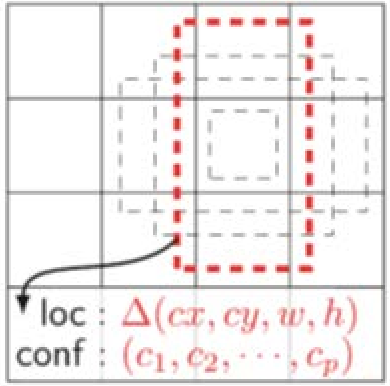


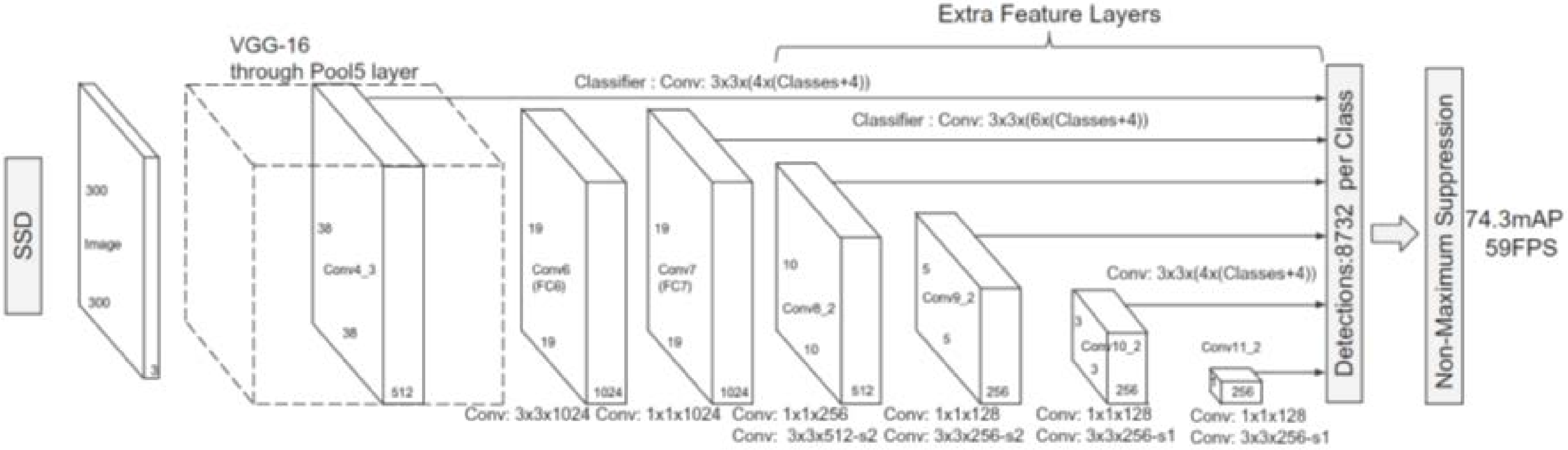
Image with GT boxes

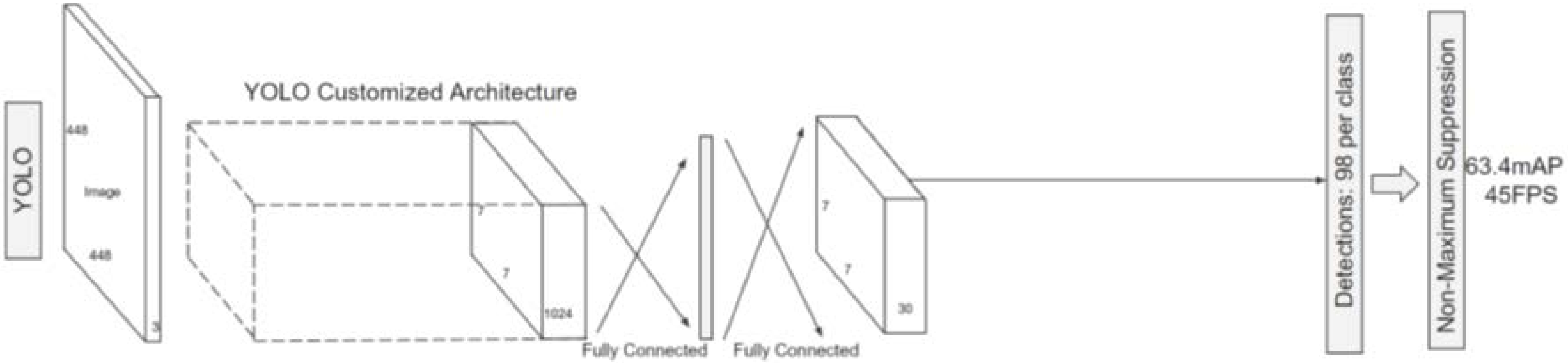


8*8 feature map



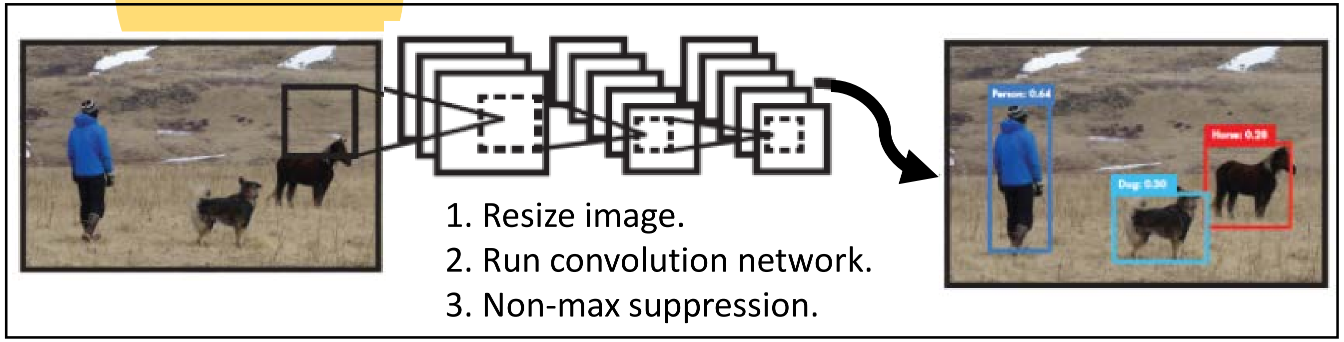
4*4 feature map

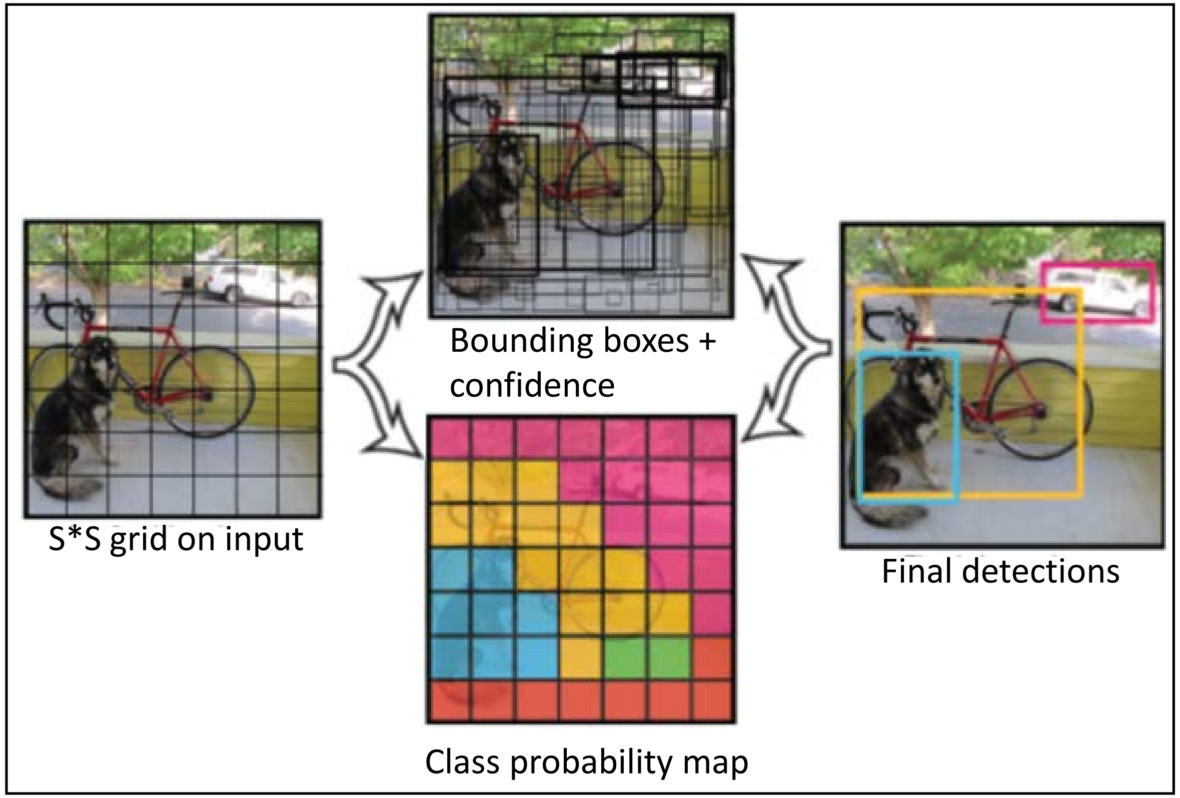


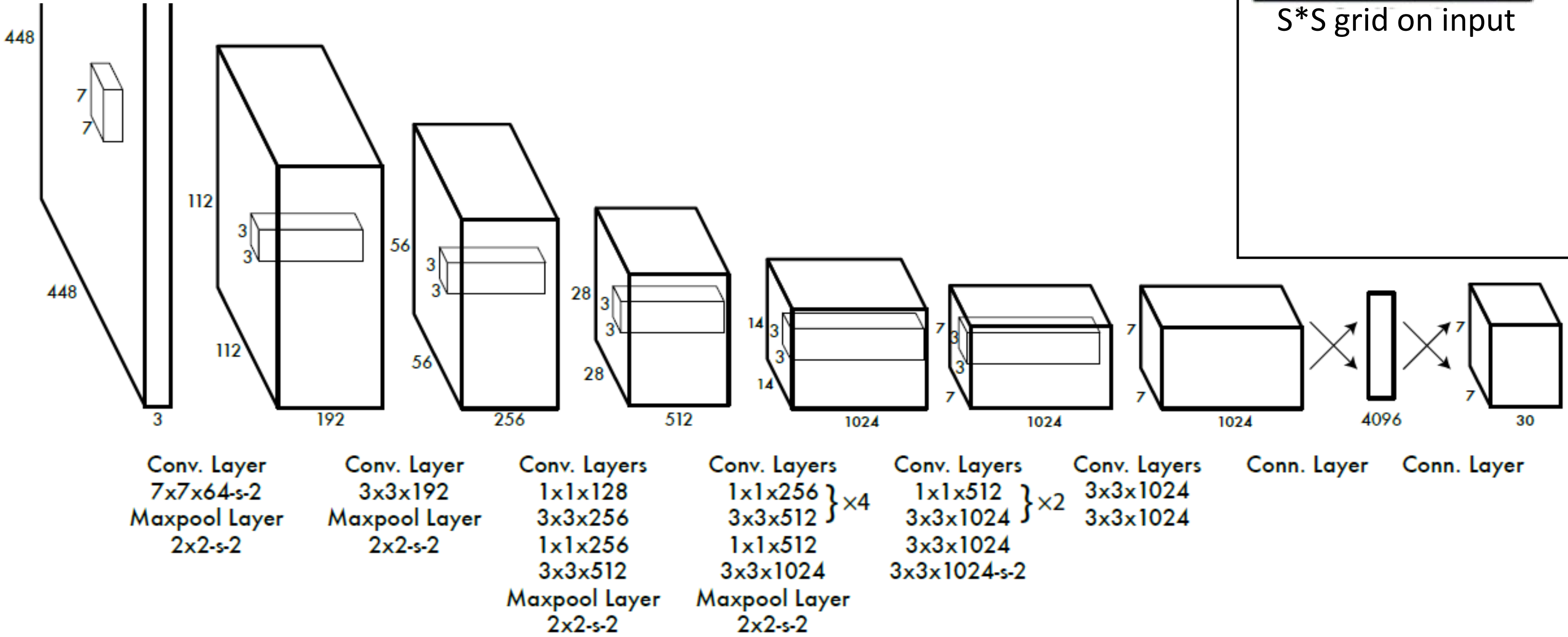


2016_SSD, Single Shot MultiBox Detector (ECCV)

YOLO (You Only Look Once)

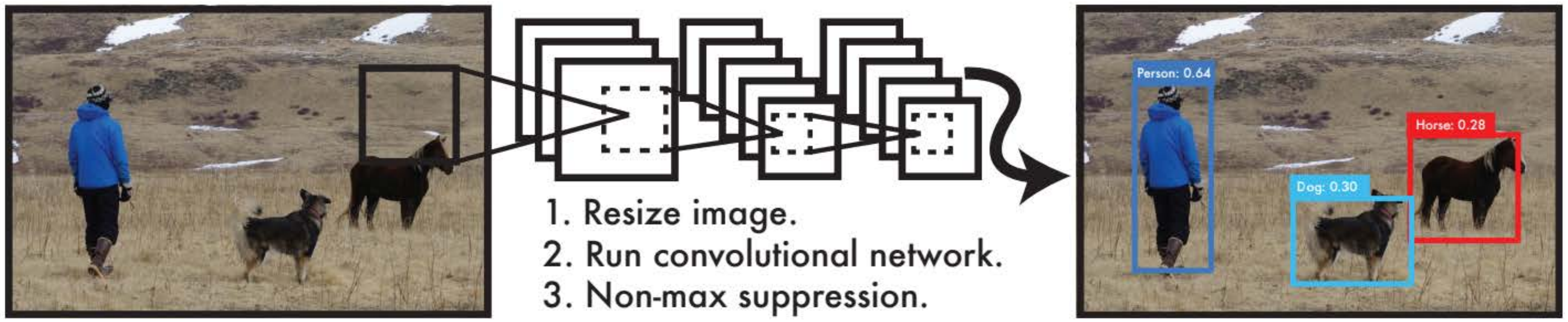






2016_You Only Look Once, Unified, Real-Time Object Detection (CVPR)

YOLO



Unified Detection

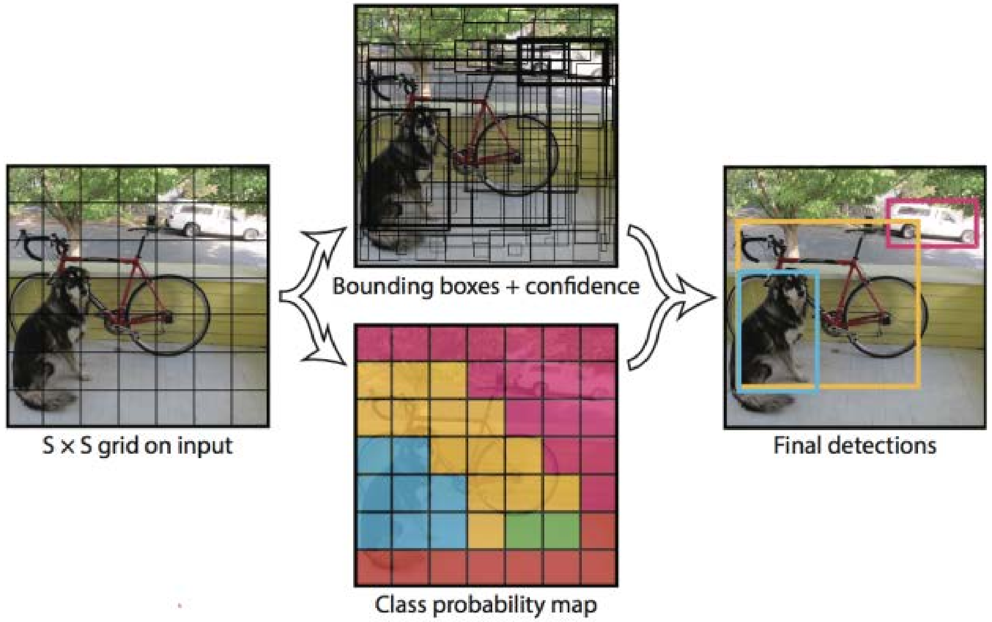
1) Divide Image into **S x S** grids
2) Grid cell

→ B: BBoxes and Confidence score

$\boxed{x, y, w, h,} \boxed{confidence} \rightarrow \Pr(\text{Object}) * \text{IOU}^{\text{truth}}_{\text{pred}}$

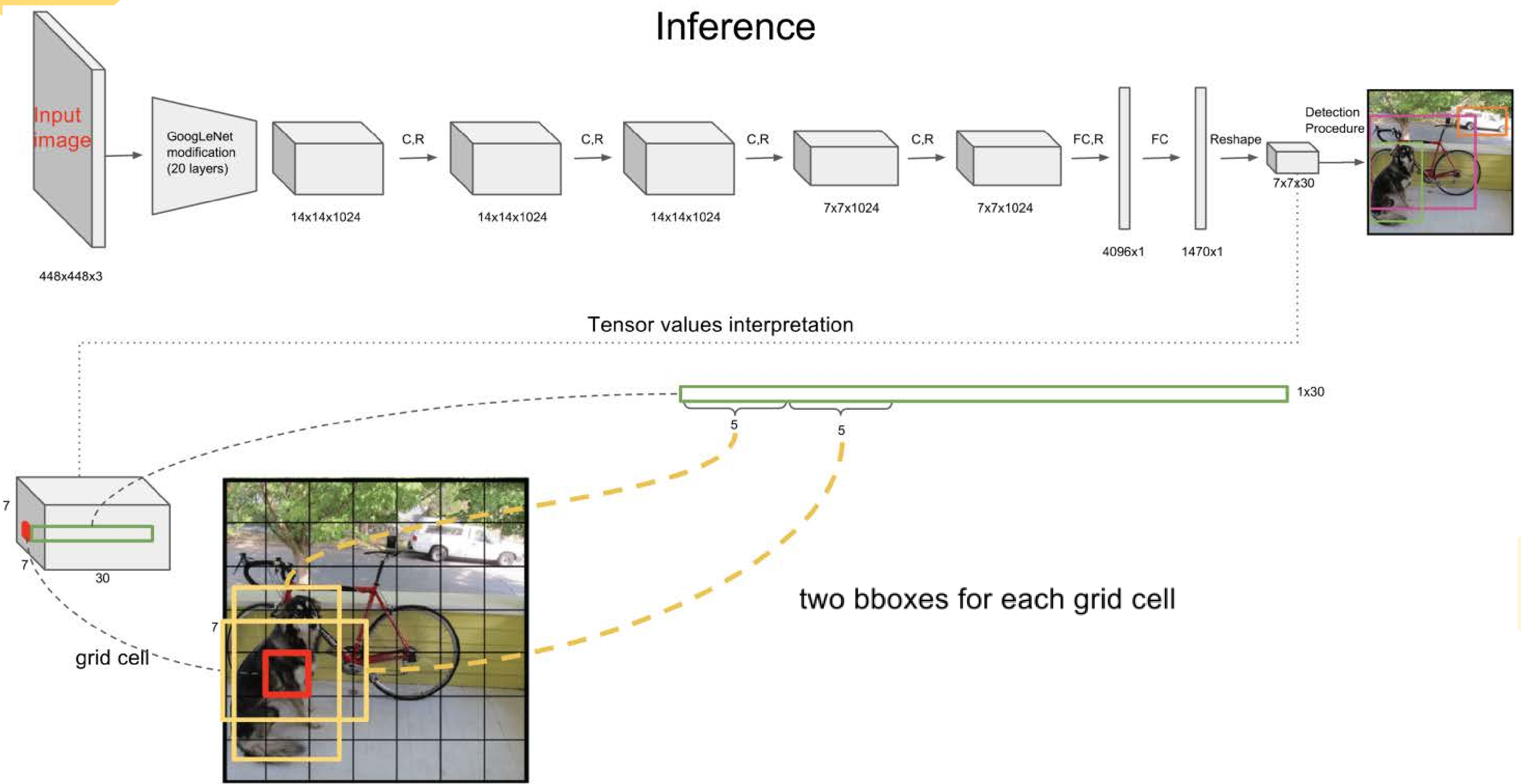
→ C: class probabilities w.r.t #classes

$\Pr(\text{Class}_i | \text{Object})$



SxS grids
B bounding boxes for each grid
C class probabilities for each grid
=> an S*S*(5*B+C) tensor

YOLOv1

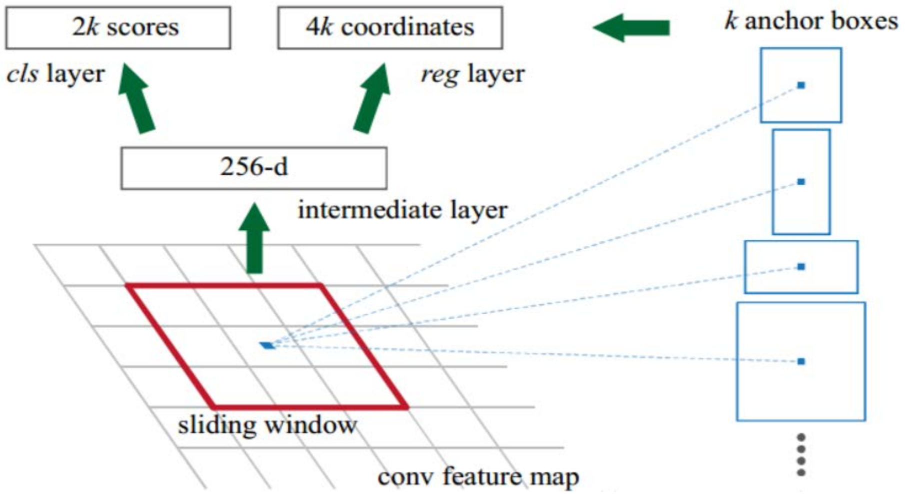

Inference
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
1x30
5
5
7
7
30
7
grid cell
two bboxes for each grid cell

YOLOv2

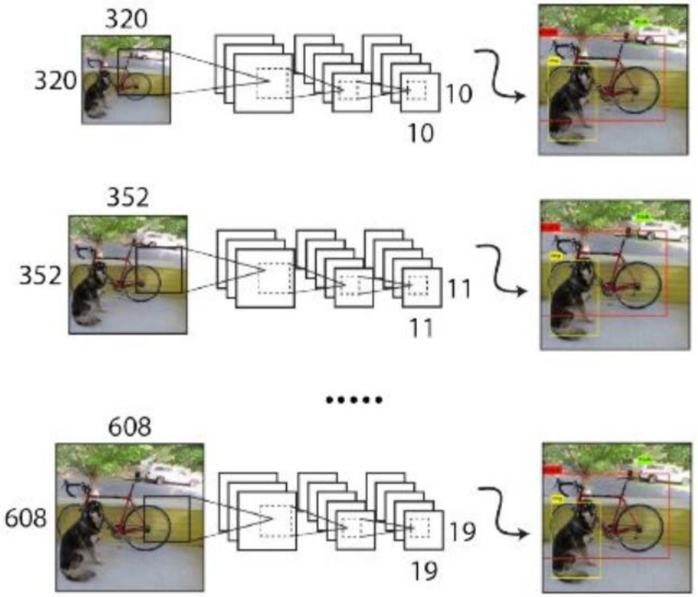
CNN(darknet-19)

Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2/2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2/2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	1000	1 × 1	7 × 7
Avgpool		Global	1000
Softmax			

- **With Anchor Boxes :**
  引入Faster R-CNN中的anchor思維，並使用K-means聚類方法訓練，自動找到更好的boxes寬高維度，提高精準度。



- **Multi-Scale Training:**
  Average pooling Layer 使YOLOv2可以適應不同大小的輸入圖片。



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

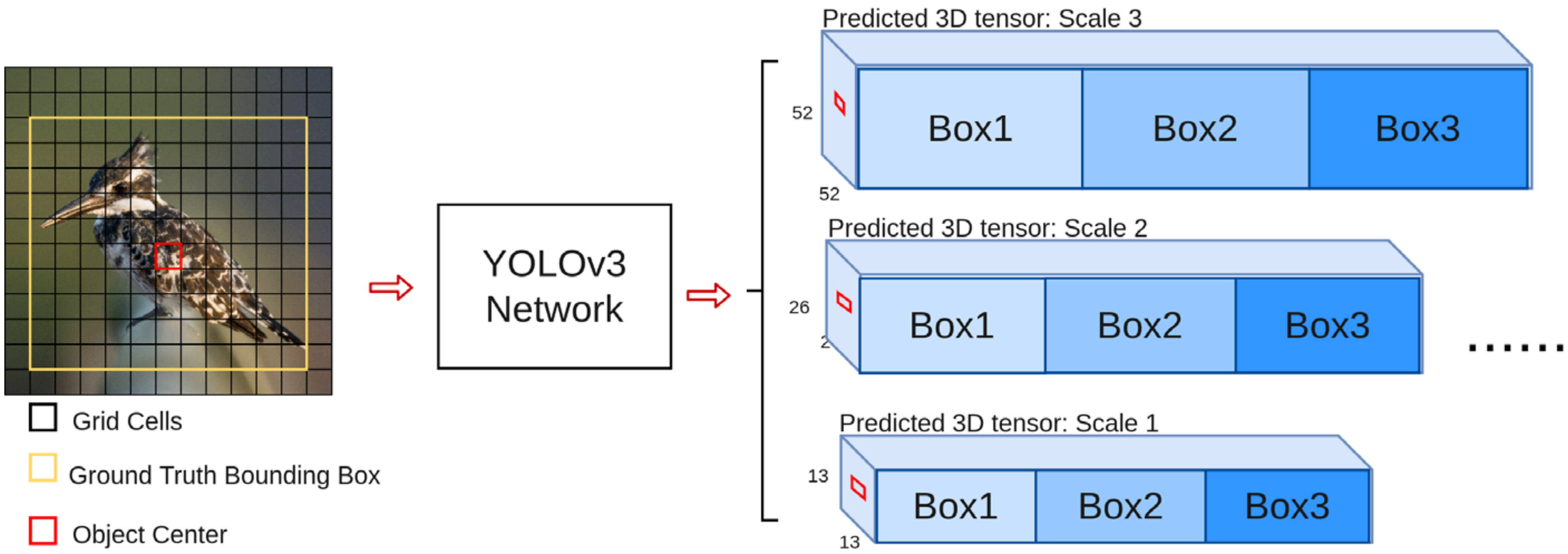
YOLO v2

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

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Darknet-19

YOLOv3



3D tensor Dimension:

N x N x [3 x (4 + 1 + 80)]

NxN: Scale size (i.e. 13x13)

3: # Boxes: (each grid predict 3 boxes)

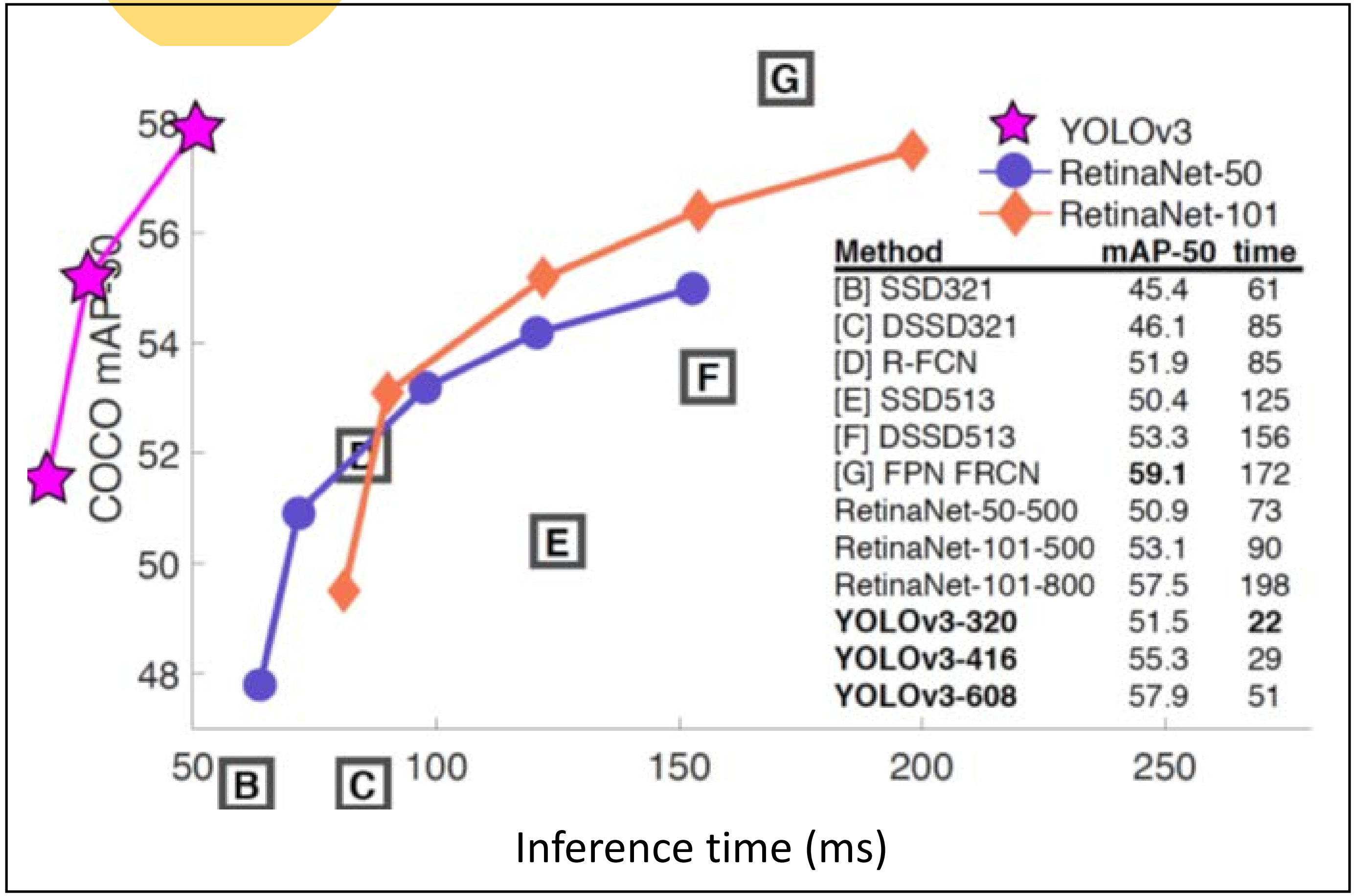
4: Box Cordinate: (tx, ty, tw, th)

1: Objectness score: how likely it is an object

80: # classes

3 different scales

YOLO v3



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Darknet-53

2018_YOLOv3: An Incremental Improvement

Object Detection: Lots of variables...

Base Network :
VGG16
ResNet-101
Inception V2
Inception V3
Inception
ResNet
MobileNet

Object Detection architecture :
Faster R-CNN
R-FCN
SSD

Image Size
Region Proposals
…

Takeaways :
Faster R-CNN is slower but more accurate

SSD is much faster but not as accurate

Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017
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

Object Detection: Impact of Deep Learning

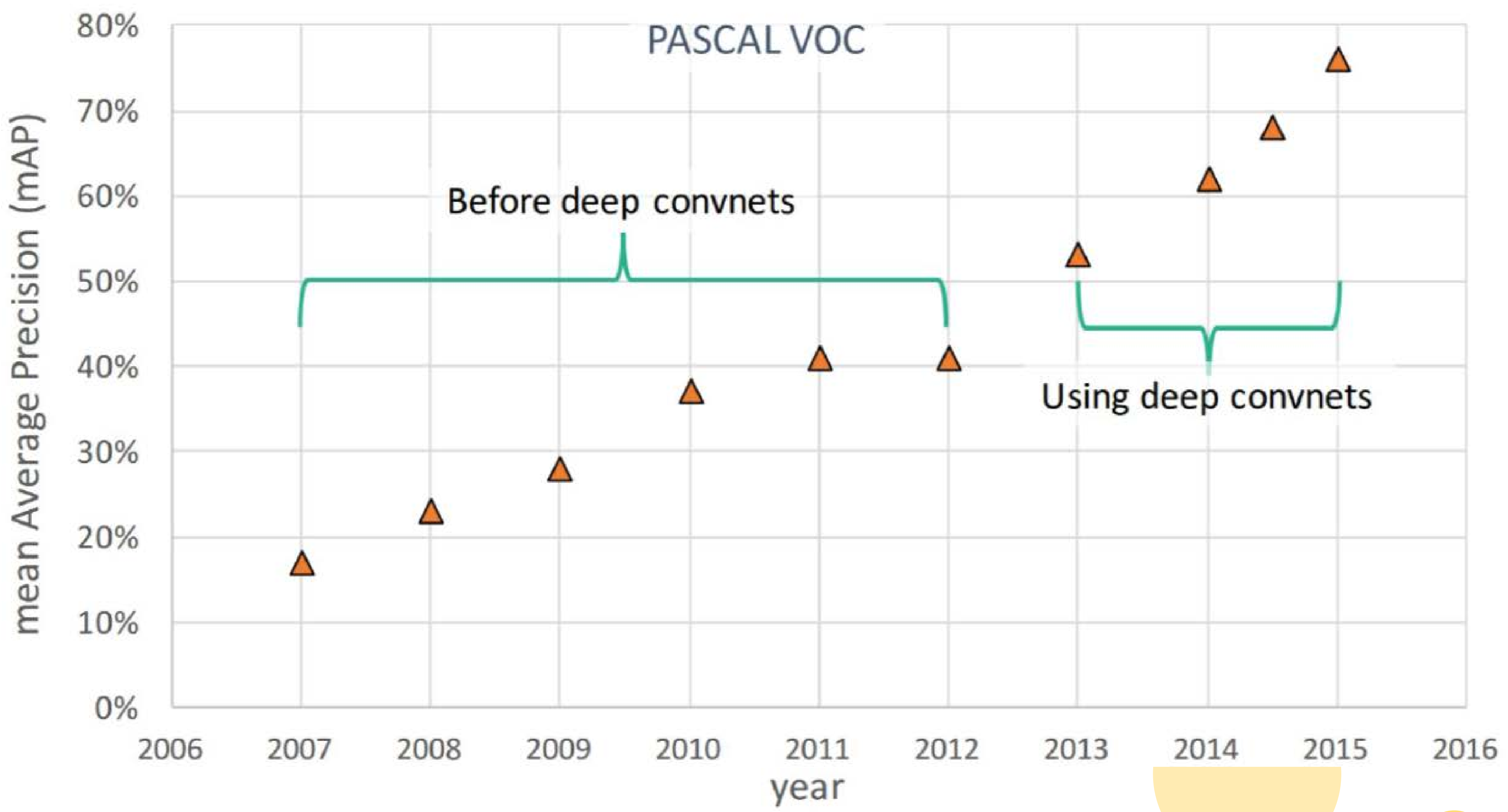
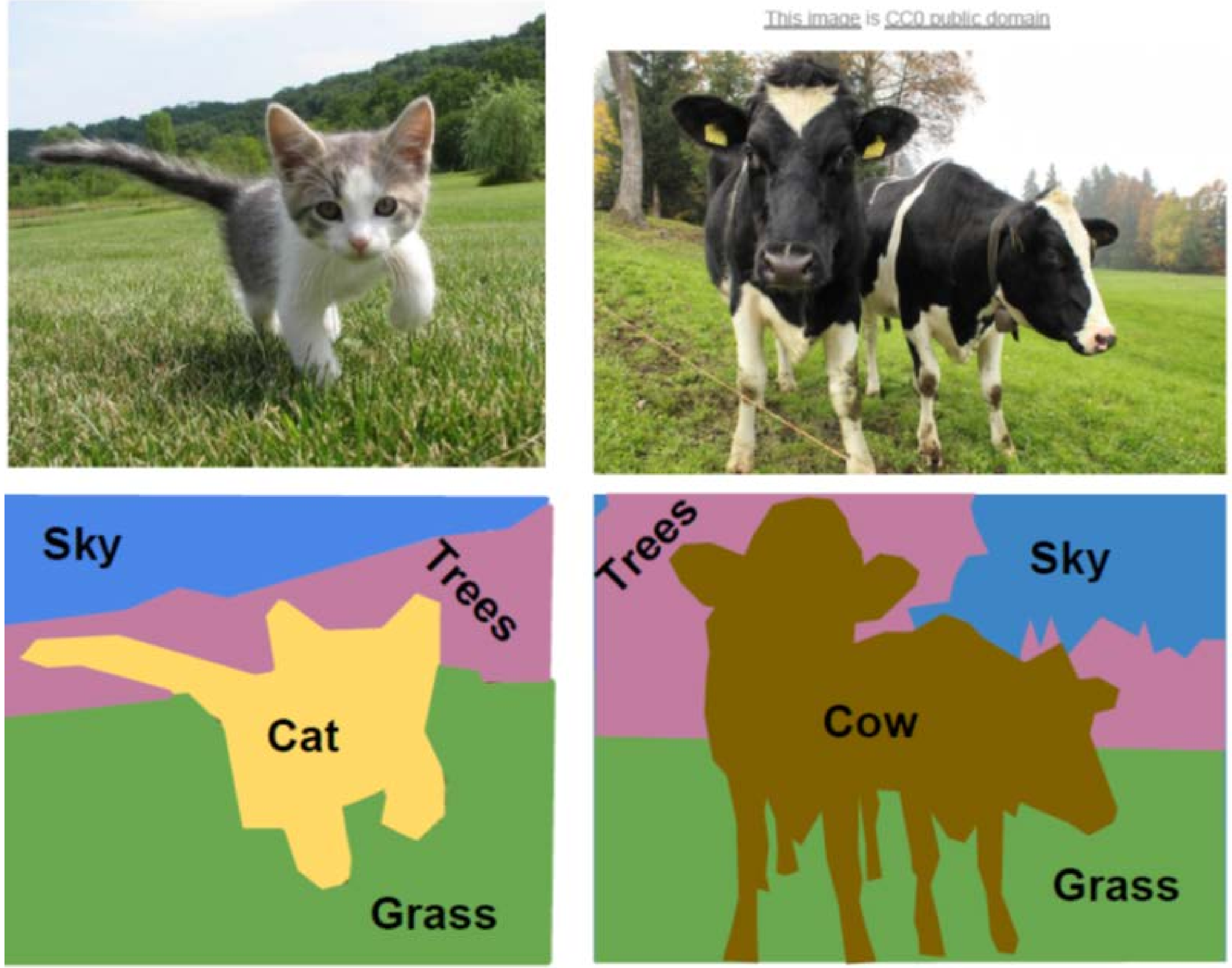


Image Segmentation

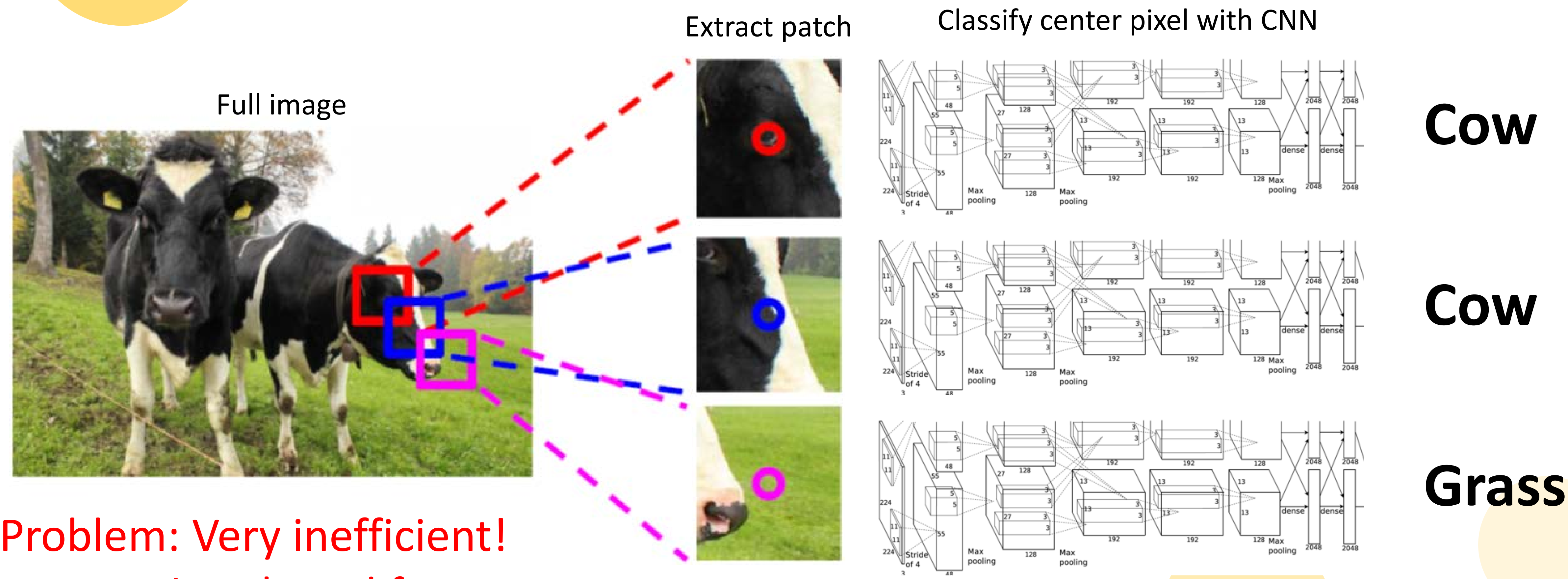
Semantic Segmentation

Label each pixel in the image with a category label

Don't differentiate instances, only care about pixels



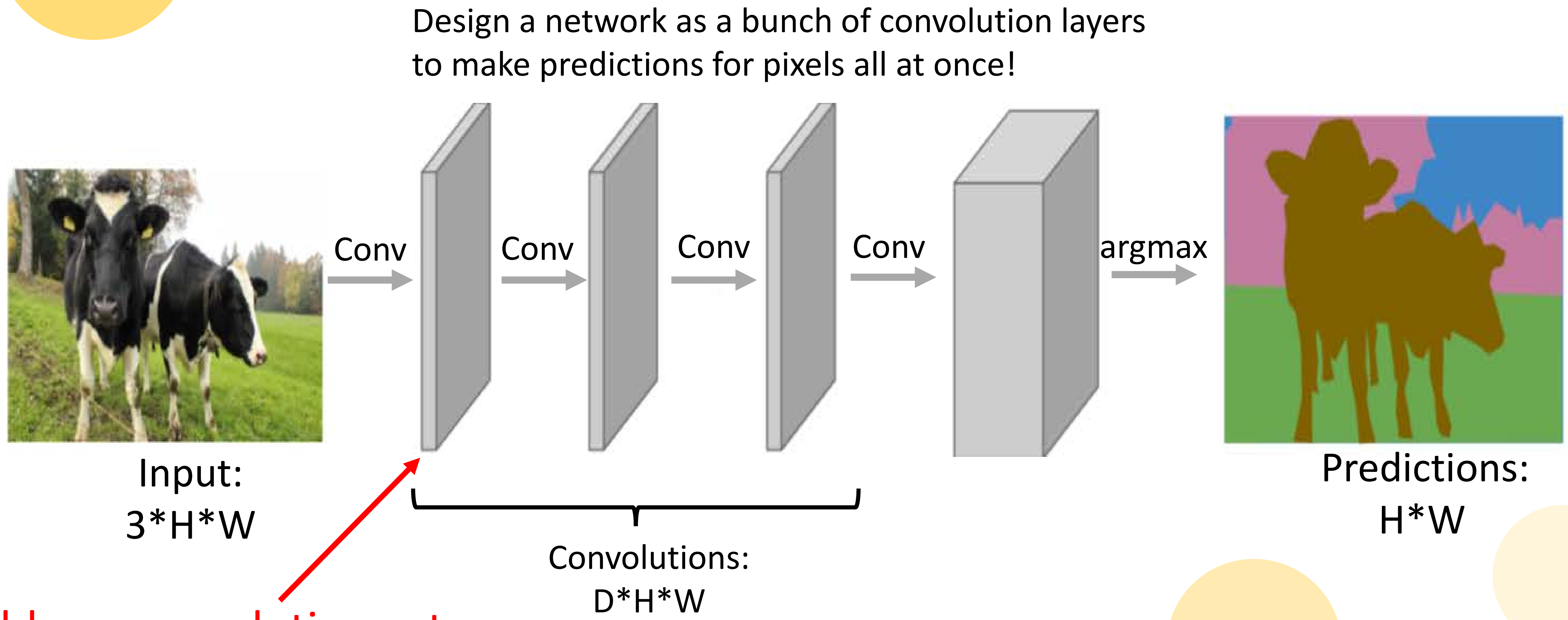
Semantic Segmentation Idea: Sliding Window



Problem: Very inefficient!
Not reusing shared features
between overlapping patches

Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

Semantic Segmentation Idea: Fully Convolution



Problem: convolutions at original image resolution will be very expensive ...

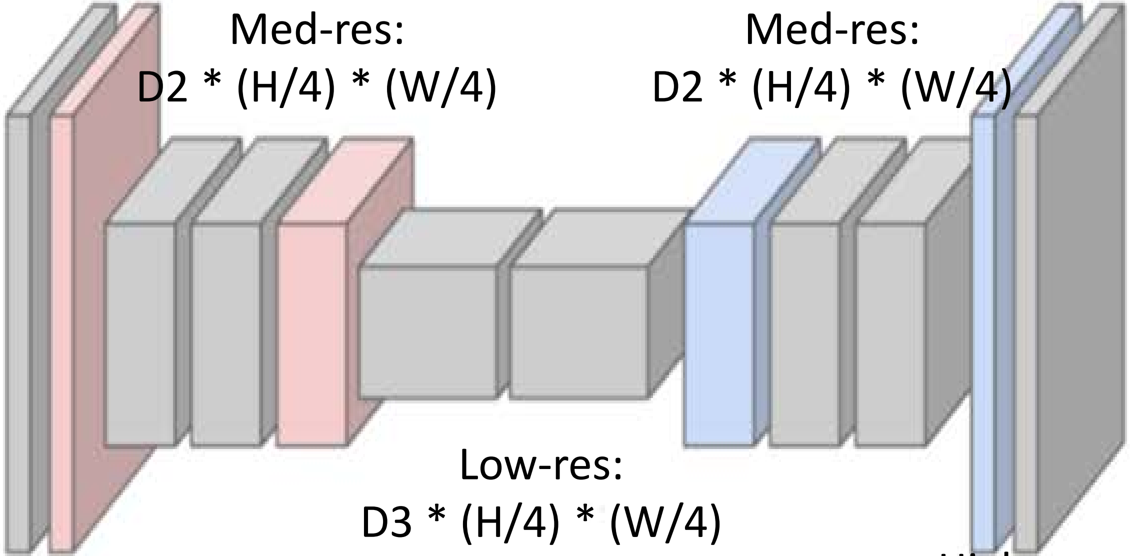
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided convolution

Design network as a bunch of convolutional layers, with downsampling and unsampling inside the network !

Unsampling:
Unpooling or strided transpose convolution

Input:
3*H*W



Med-res:
D2 * (H/4) * (W/4)

Med-res:
D2 * (H/4) * (W/4)

Low-res:
D3 * (H/4) * (W/4)

High-res:
D1 * (H/2) * (W/2)

High-res:
D1 * (H/2) * (W/2)

Predictions:
H*W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

3x3 Transpose Convolution (DeConvolution)

3 * 3 transpose convolution, stride 2 & pad 1

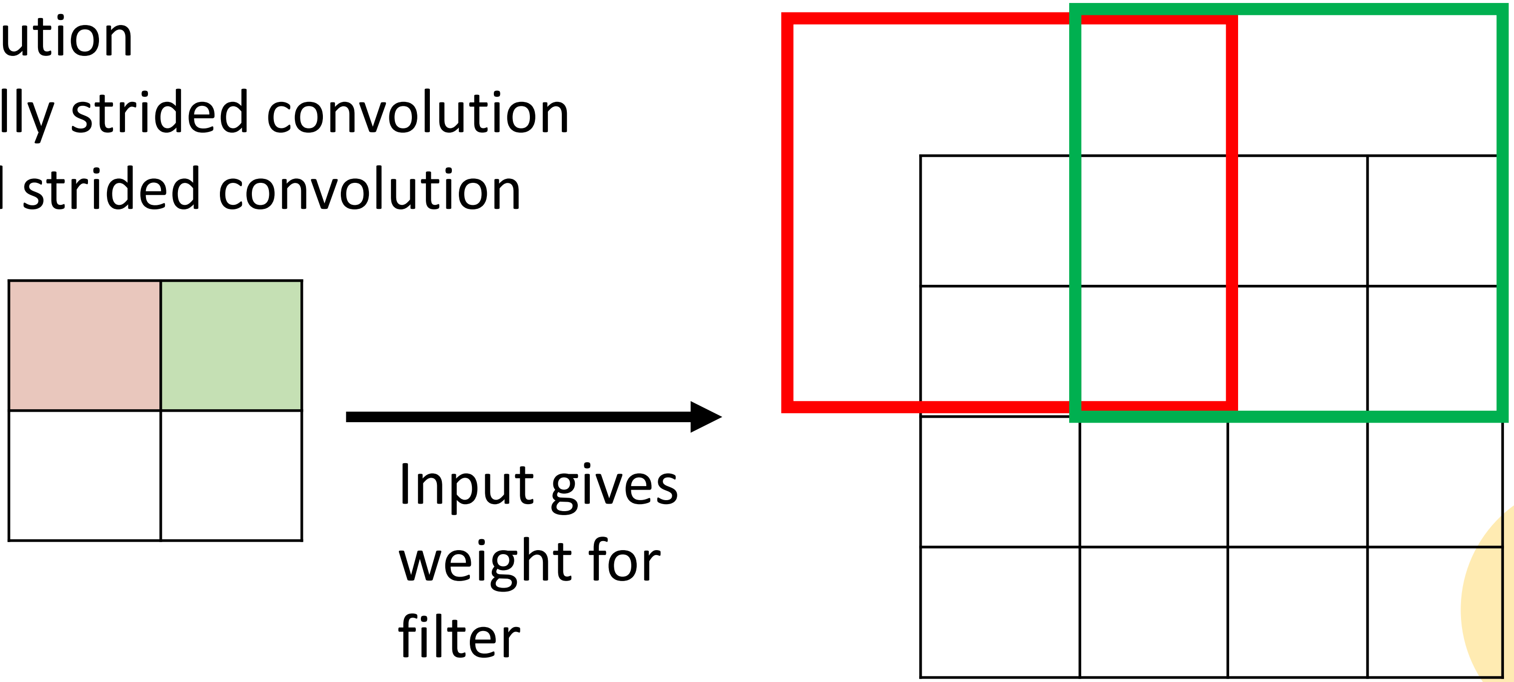
Other names:

-Deconvolution (bad)

-Upconvolution

-Fractionally strided convolution

-Backward strided convolution



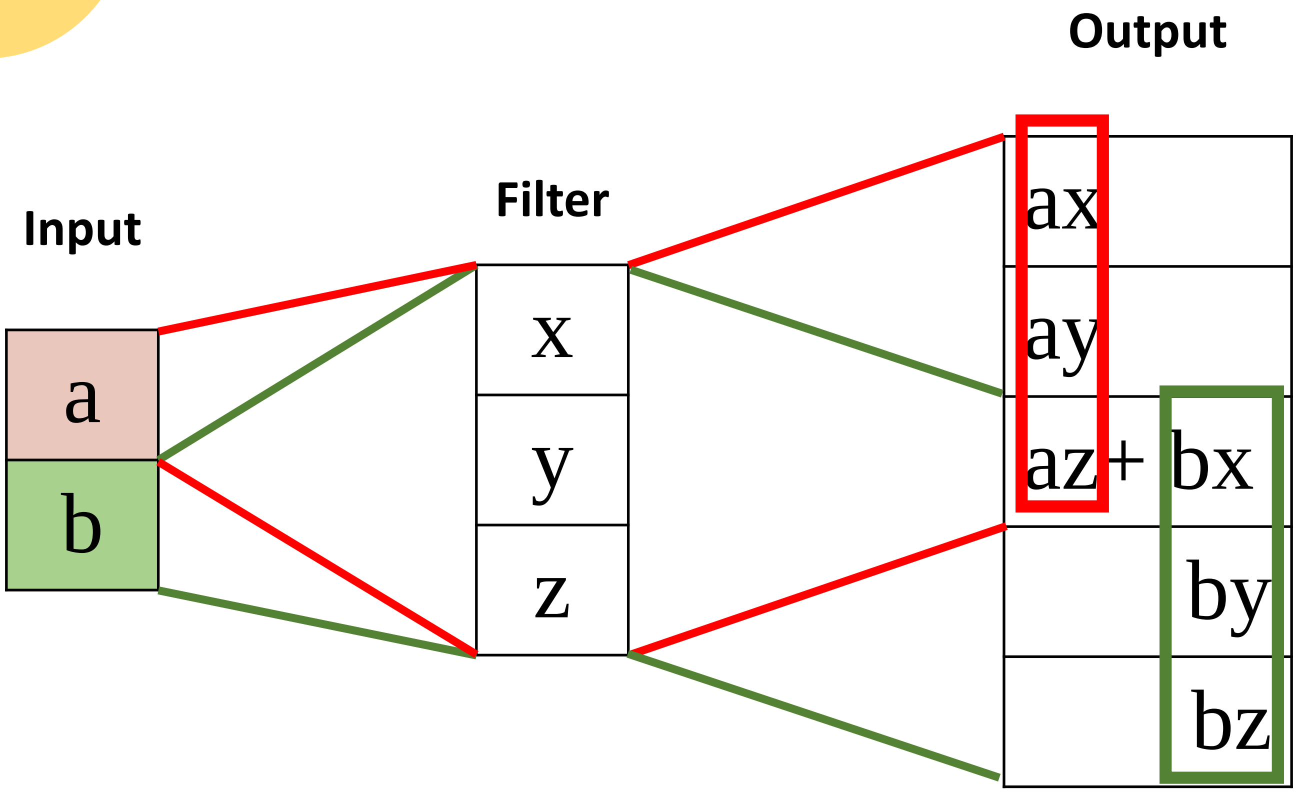
Input: 2 * 2

Output: 4 * 4

Filter moves 2 pixels in the **output** for every one pixel int the **input**

Stride gives ratio between movement in output and input

1D Example



Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

Need to crop one pixel from output to make output exactly 2x input

1-D Conv and DeConv with stride=1

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & x & 0 & 0 & 0 \\ 0 & x & y & x & 0 & 0 \\ 0 & 0 & x & y & x & 0 \\ 0 & 0 & 0 & x & y & x \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=1, padding=1

Convolution transpose multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T\vec{a}$$

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay + bx \\ az + by + cx \\ bz + cy + dx \\ cz + dy \\ dz \end{bmatrix}$$

When stride=1, convolution transpose is just a regular convolution (with different padding rules)

1-D Conv and DeConv with stride>1

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{pmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{pmatrix} \begin{pmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{pmatrix} = \begin{pmatrix} ay + bz \\ bx + cy + dz \end{pmatrix}$$

Example: 1D conv, kernel size=3, **stride=2**, padding=1

Convolution transpose multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T\vec{a}$$

$$\begin{pmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} ax \\ ay \\ az + bx \\ by \\ bz \end{pmatrix}$$

When stride > 1, convolution transpose is a no longer a normal convolution !