CS396: Selected CS2

(Deep Learning for visual recognition)

Spring 2022

Dr. Wessam EL-Behaidy

Associate Professor, Computer Science Department,
Faculty of Computers and Artificial Intelligence,
Helwan University.

Lectures (Course slides) are based on Stanford course :
Convolutional Neural Networks for Visual Recognition (CS231n):
http://cs231n.stanford.edu/index.html

Lecture 7: Detection and Segmentation

Loss Function

Loss Function

Deep learning neural networks are trained using the stochastic gradient descent optimization algorithm. As part of the optimization algorithm, the error for the current state of the model must be estimated repeatedly.

This requires the choice of an error function, conventionally called a **loss function**, that can be used to estimate the loss of the model so that the weights can be updated to reduce the loss on the next evaluation.

Note: Loss function also called **cost** or **error function**

Loss Functions

1. Regression Loss Functions
    1. Mean Absolute Error / L1 Loss
    2. Mean Squared Error/L2 Loss

2. Classification Loss Functions
    1. Hinge Loss
    2. Binary/ Multi-Class Cross-Entropy (Log loss)
    3. Categorial Cross-Entropy (Softmax loss)

Regression Loss Functions

Loss functions used in **Regression Problems**:

1. **Mean Absolute Error/L1 Loss:** used to minimize the error which is the mean of the sum of all the absolute differences between the true value and the predicted value.

$$\text{L1 loss} = \frac{\sum_{i=1}^{n} |y_{true} - y_{predicted}|}{n}$$

2. **Mean Squared Error/L2 Loss:** used to minimize the error which is the mean of the sum of all the squared differences between the true value and the predicted value.

$$\text{L2 loss} = \frac{1}{n}\sum_{i=1}^{n}(y_{true} - y_{predicted})^2$$

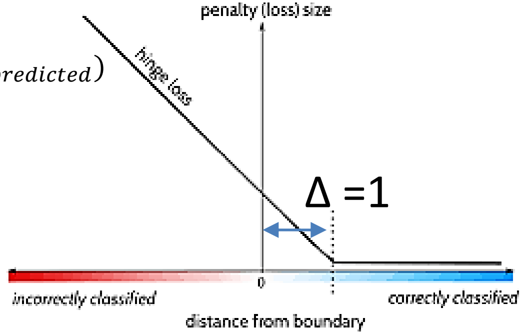
n= No. of samples, y_{true}= true label, $y_{predicted}$=predicted label

Classification Loss Functions

Loss functions used in **classification Problems**:

1. **Hinge Loss/ SVM Loss:** It is mainly used in problems where you have to do ‘maximum-margin’ classification. Even if new observations are classified correctly, they can incur a penalty if the margin from the decision boundary is not large enough.

$$\text{loss} = \max(0, 1 - y_{true} * y_{predicted})$$



Classification Loss Functions

Loss functions used in **classification Problems**:

2. **Cross-entropy Loss/ Logistic Loss/ Multinomial Logistic Loss/ Log Loss:** It measures the amount of divergence of predicted probability with the actual label. So lesser the log loss value, more the perfectness of model. For a perfect model, log loss value = 0.

Loss (multiclass) $= -\frac{1}{n}\sum_{i=1}^{n}\sum_{j}^{m} y_{ij}\log(p_{ij})$,

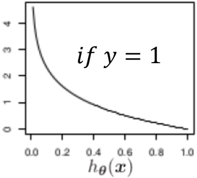
$Where$: p_{ij} = indicates probability of i[th] sample belonging to j[th] class.

m= number of classes

Binary cross-entropy (when m=2)

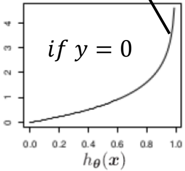
$$loss = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(p_i) + (1 - y_i)\log(1 - p_i)$$

$Cost \to \infty, y \to 0$



$Cost = 0, y = 1$

$Cost \to \infty, y \to 1$

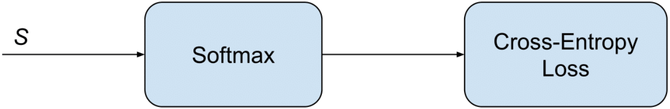


$Cost = 0, y = 0$

Classification Loss Functions

Loss functions used in **classification Problems**:

3. Categorial Cross-Entropy/Softmax loss: It is a **Softmax activation** plus a **Cross-Entropy loss**. If we use this loss, we will train a CNN to output a probability over the **C** classes for each image. It is used for multi-class classification.

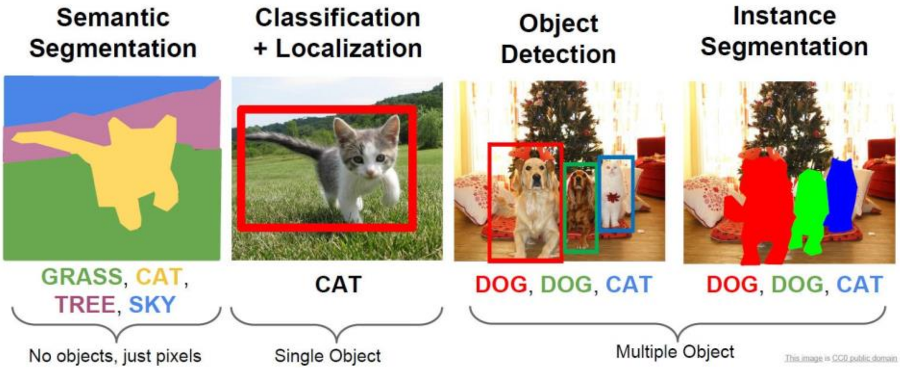


$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \qquad CE = -\sum_i^C t_i log(f(s)_i)$$

$$t_i = true\ label, f(s)_i = softmax\ function, C = no.of\ classes$$

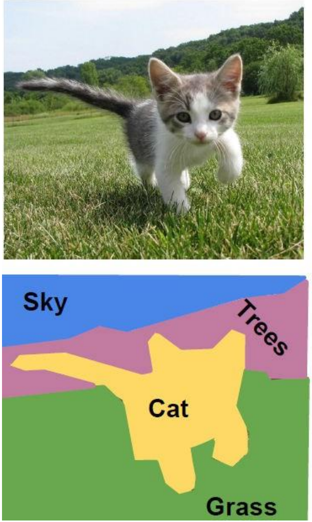
Detection and Segmentation

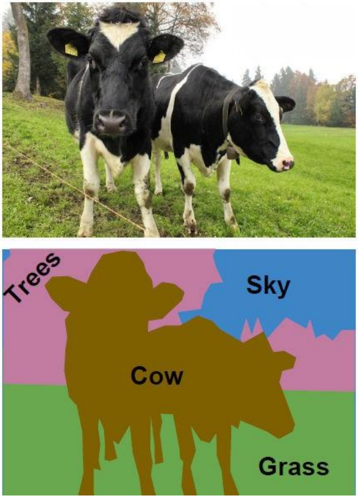
Computer vision tasks



Semantic segmentation

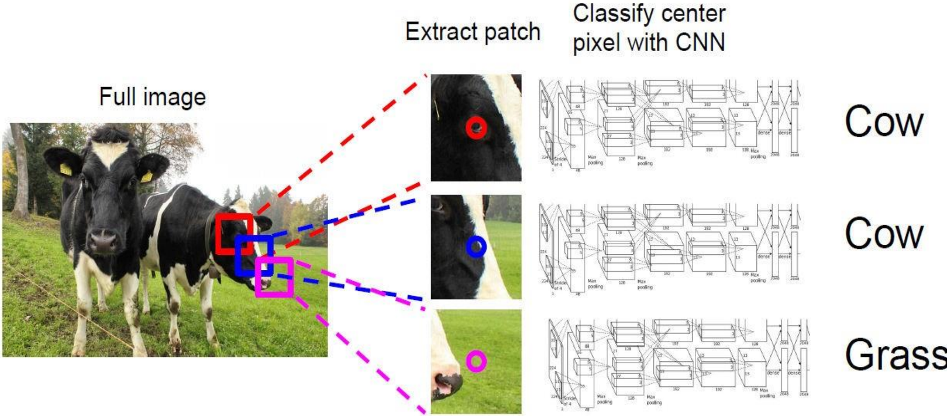
- **Semantic segmentation** is understanding an image at pixel level i.e, we want to assign each pixel in the image to an object class.





Don’t differentiate instances, only care about pixels

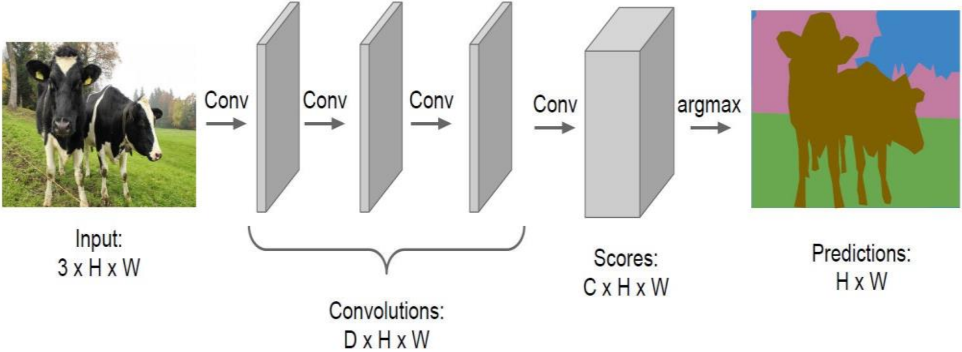
Sliding Window



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

Fully Convolutional

- Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



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

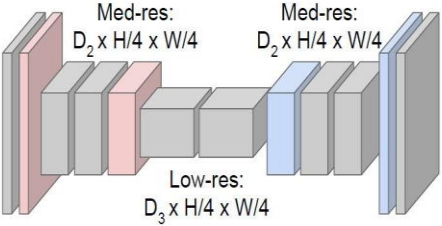
Fully Convolutional

Downsampling: Pooling, strided convolution

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

Upsampling: ???

Input: $3 \times H \times W$



High-res: $D_1 \times H/2 \times W/2$

High-res: $D_1 \times H/2 \times W/2$

Predictions: $H \times W$

Upsampling

-Unpooling
-Transpose Convolution

Other names of transpose convolution
Deconvolution (bad) , Upconvolution, Fractionally strided convolution, Backward strided convolution

In-Network upsampling: “Unpooling”

Nearest Neighbor

Input: 2 x 2

1	2
3	4

Output: 4 x 4

1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

“Bed of Nails”

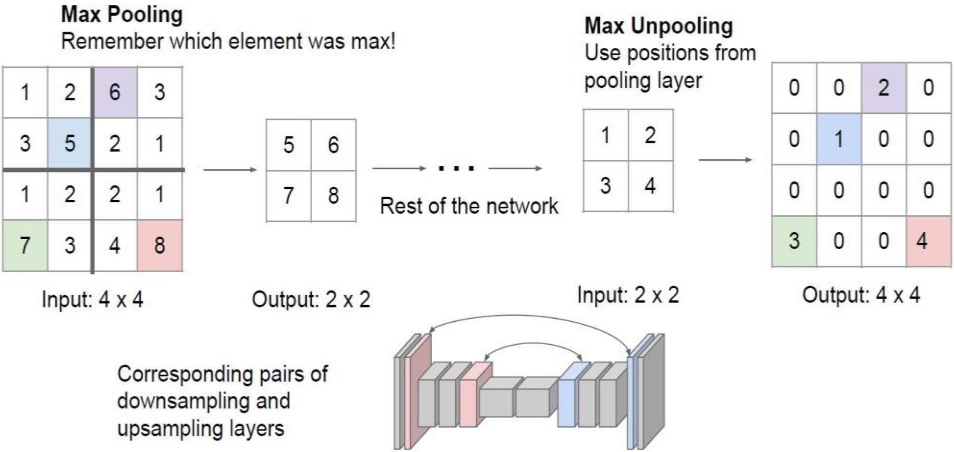
Input: 2 x 2

1	2
3	4

Output: 4 x 4

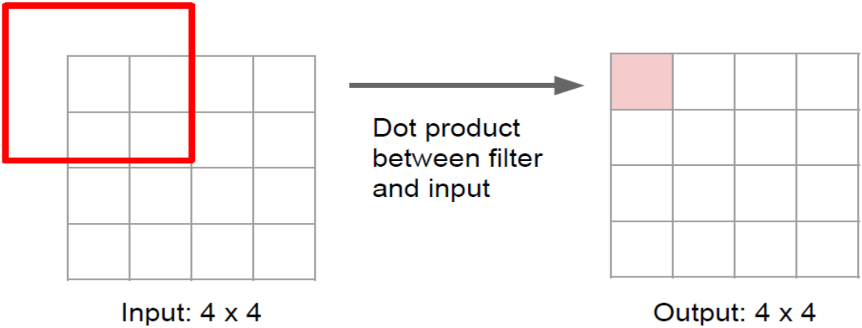
1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

In-Network upsampling: “Max Unpooling”



Learnable Upsampling: Transpose Convolution

Recall Normal 3 x 3 convolution, stride 1 pad 1

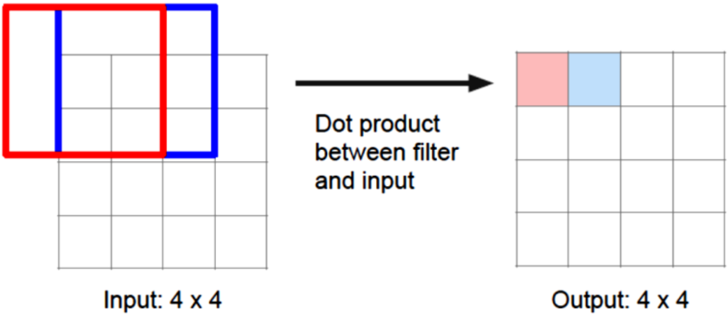


–Filter moves 2 pixels in the input for every one pixel in the output

– Stride gives ratio between movement in input and output

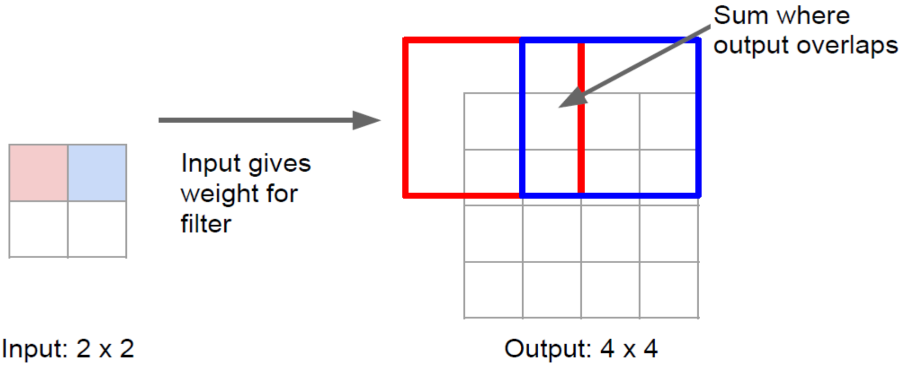
Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 1 pad 1



Learnable Upsampling: Transpose Convolution

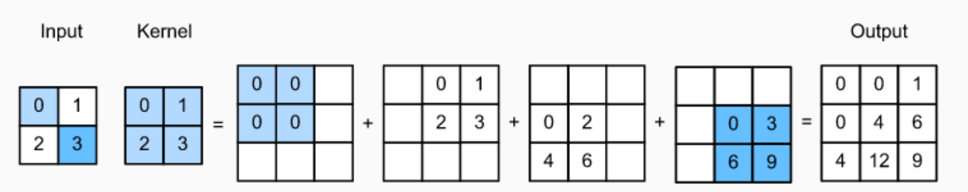
– 3 x 3 transpose convolution, stride 2 pad 1



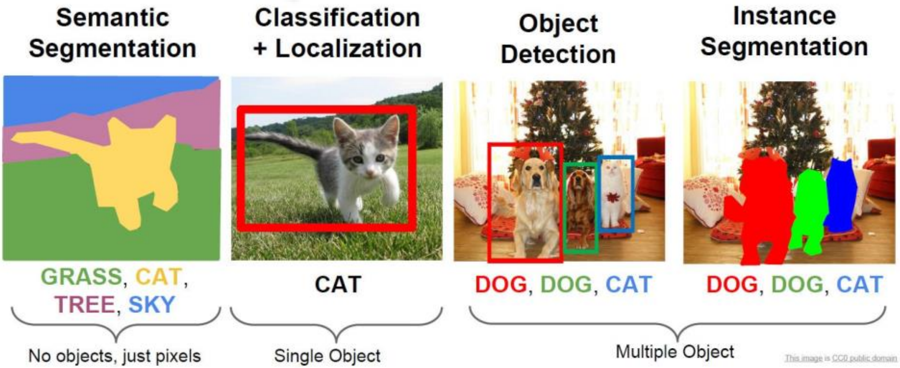
–Multiply every pixel by the weights of the filter

Transpose Convolution Example

Let us consider a basic case that both input and output channels are 1, with 0 padding and 1 stride.



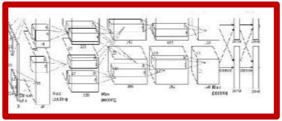
Classification + Localization



Classification + Localization

- Often pretrained on ImageNet (Transfer learning)

This image is CC0 public domain



Fully Connected: 4096 to 1000

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

Correct label: Cat

Softmax Loss

Vector: 4096

Multitask Loss

+ → **Loss**

Fully Connected: 4096 to 4

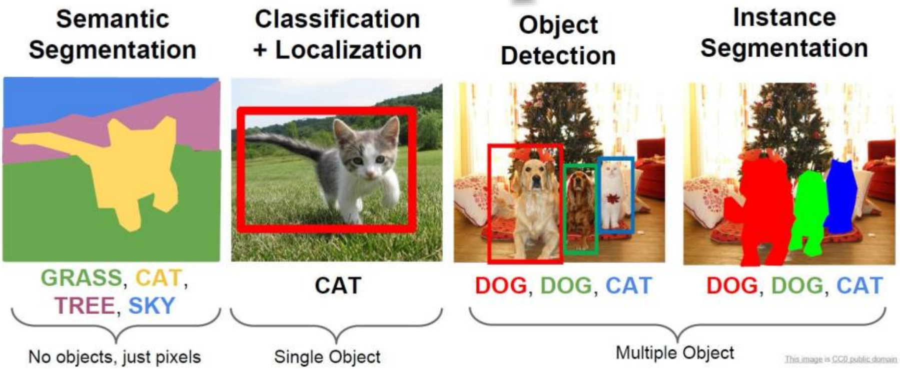
Box Coordinates
(x, y, w, h)

L2 Loss

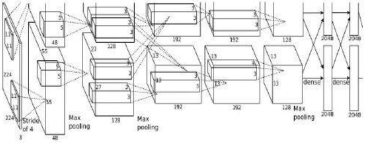
Correct box:
(x', y', w', h')

Treat localization as a regression problem!

Object Detection

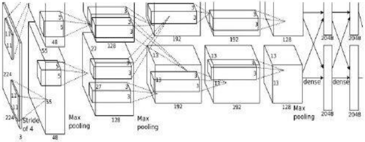


Object Detection



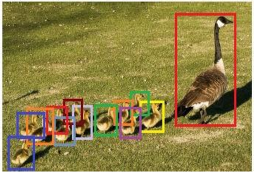
CAT: (x, y, w, h)

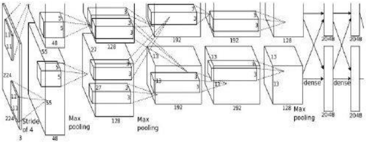
4 numbers



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)

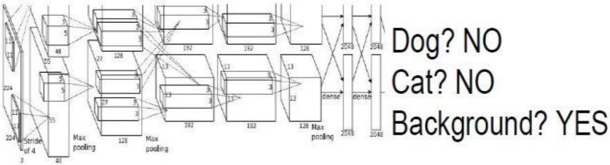
….

Many numbers

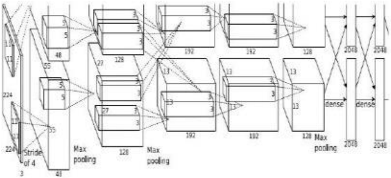
Each image needs a different number of outputs!

Object Detection as Classification: Sliding Window

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



Object Detection as Classification: Sliding Window

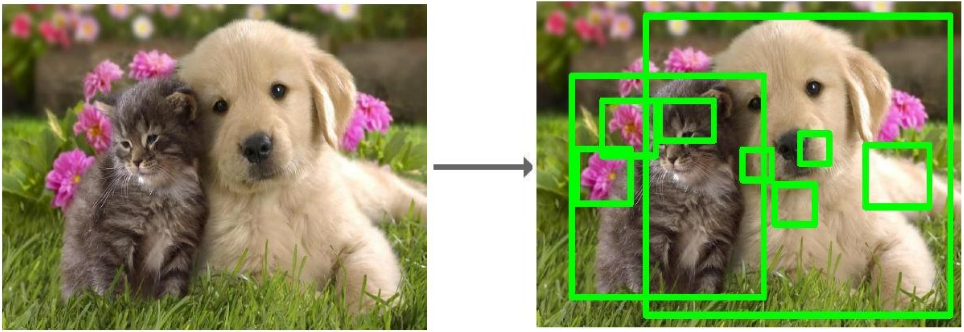


Dog? NO
Cat? YES
Background? NO

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

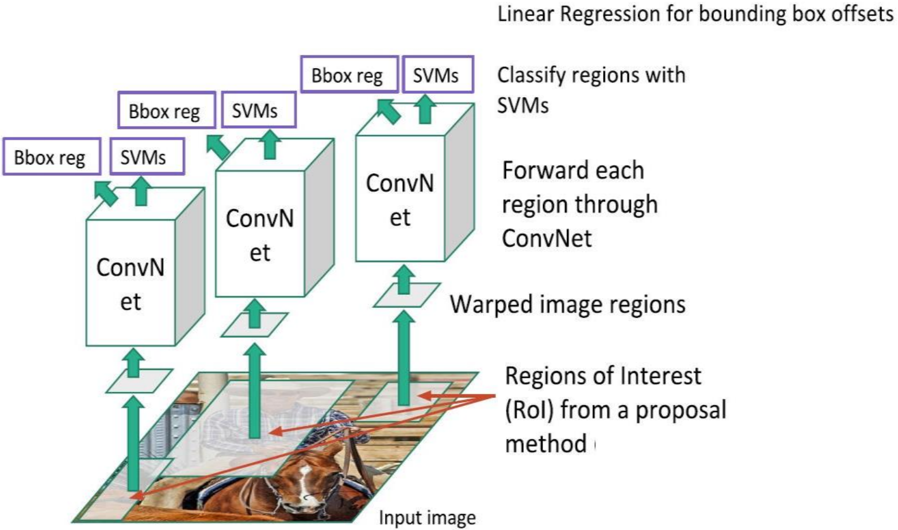
Region Proposals

–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



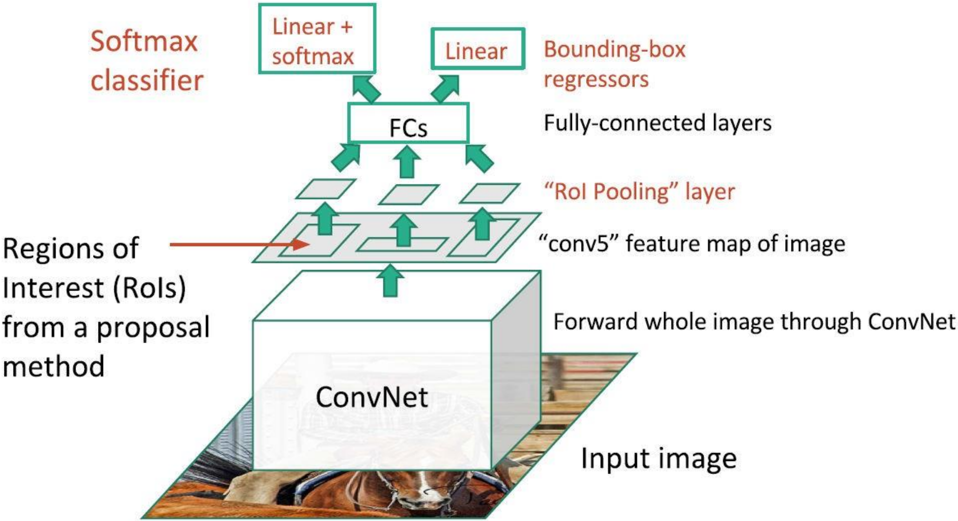
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.– Figure copyright Ross Girshick, 2015

R-CNN: Problems

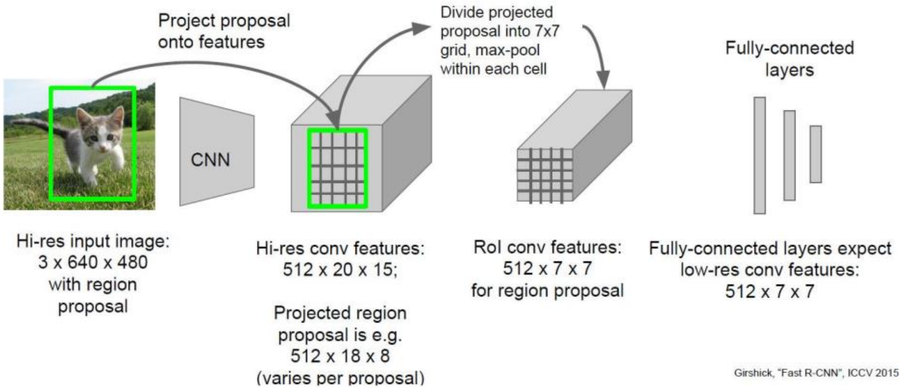
- Training is slow (84h), takes a lot of disk space
- Inference (detection) is slow
  - 47s / image with VGG16 [Simonyan Zisserman. ICLR15]
  - It cannot be implemented real time.

Fast R-CNN

Girshick, “Fast R-CNN”, ICCV 2015.– Figure copyright Ross Girshick, 2015



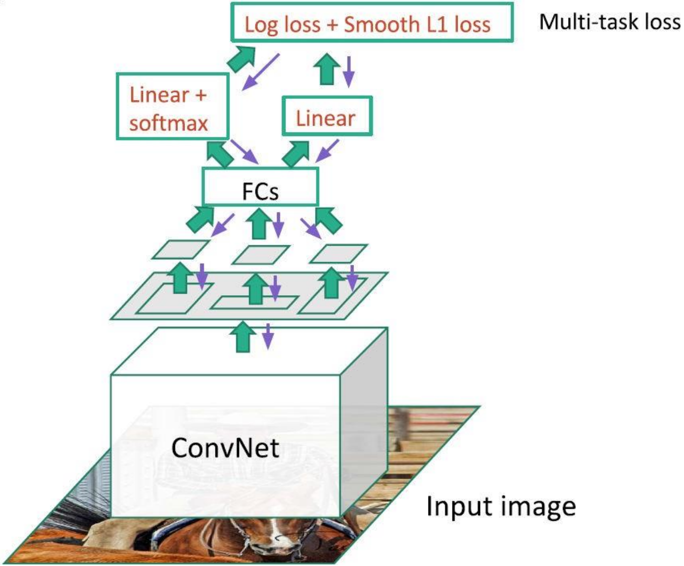
Fast R-CNN: RoI Pooling



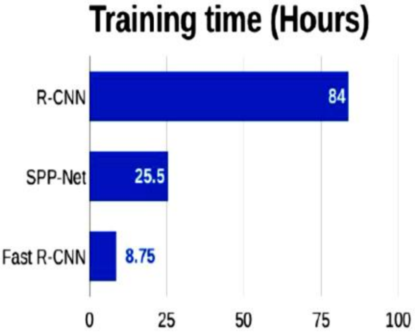
Girshick, "Fast R-CNN", ICCV 2015.

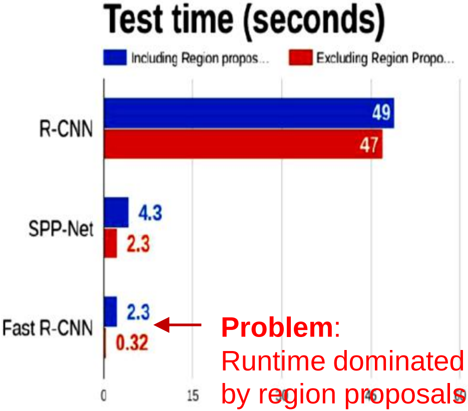
Fast R-CNN (training)

Girshick, “Fast R-CNN”, ICCV 2015.– Figure copyright Ross Girshick, 2015



R-CNN vs SPP vs Fast R-CNN

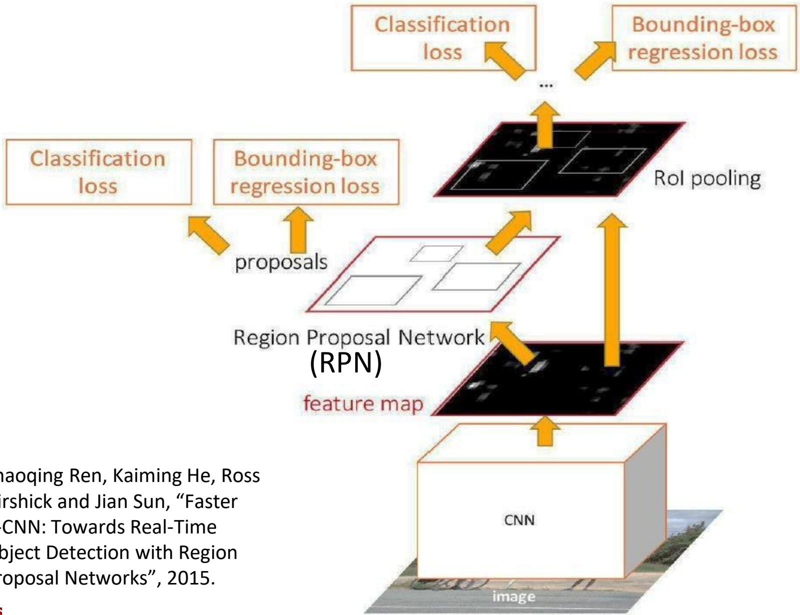




Problem:
Runtime dominated by region proposals!

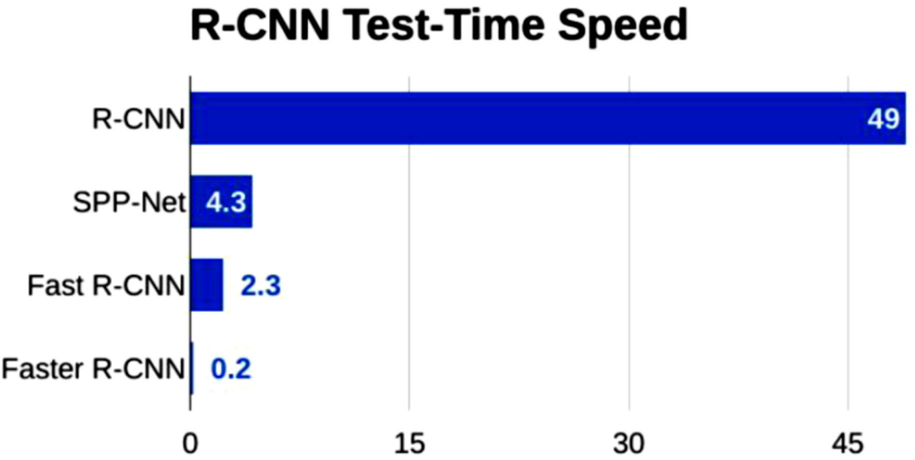
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

Faster R-CNN: Make CNN do proposals!



Shaoqing Ren, Kaiming He, Ross Girshick and Jian Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, 2015.

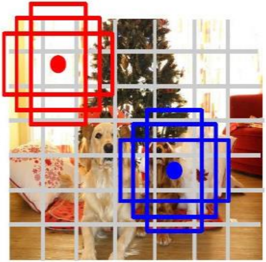
Faster R-CNN


R-CNN Test-Time Speed
R-CNN
49
SPP-Net
4.3
Fast R-CNN
2.3
Faster R-CNN
0.2
0
15
30
45

Detection without Proposals: YOLO / SSD

YOLO: You Only Look Once – SSD: Single-Shot MultiBox Detector

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:

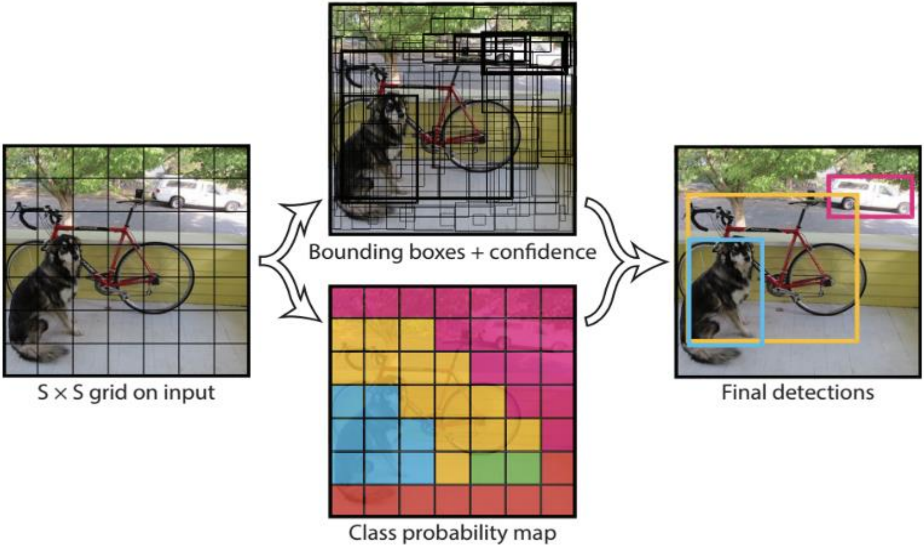
- Regress from each of the B base boxes to a final box with 5 numbers:
  (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)

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

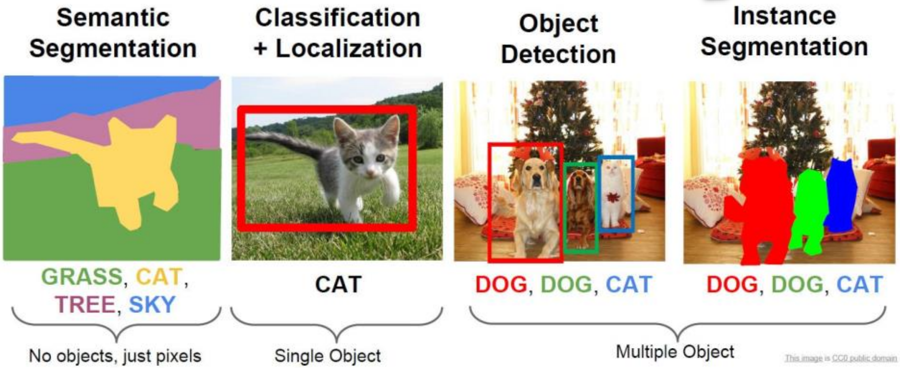
Go from input image to tensor of scores with one big convolutional network!

Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016 – Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016

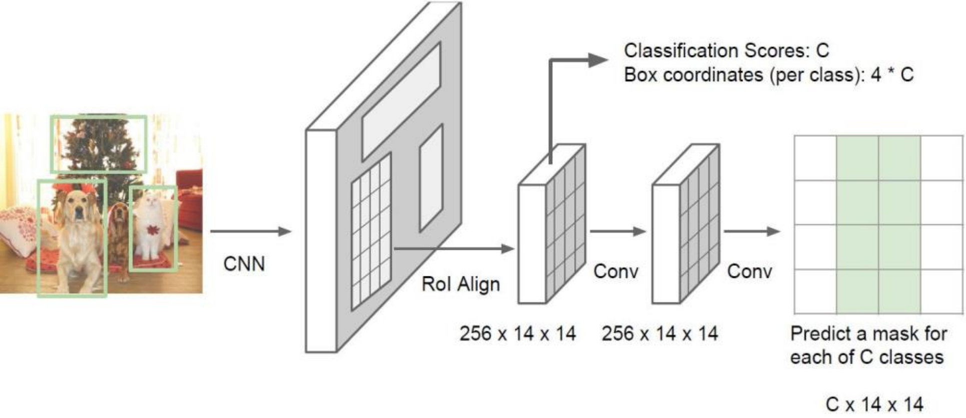
YOLO Examples



Instance Segmentation

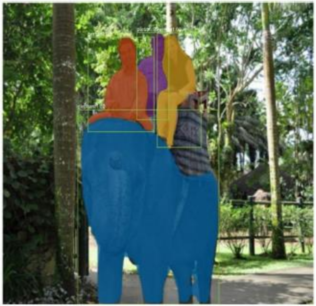


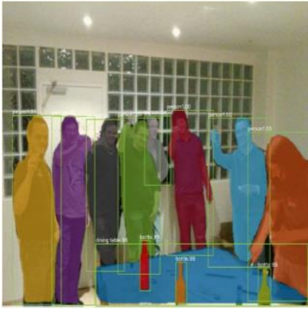
Instance Segmentation: Mask R-CNN



K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017

Mask R-CNN





This lecture references

- CS231 Stanford:

https://www.youtube.com/watch?v=nDPWywWRIRo

- useful video:

https://www.youtube.com/watch?v=g7z4mkfRjI4 ,
https://www.youtube.com/watch?v=2TikTv6PWDw