



# Objectives

---

- What is beyond image classification
- Metrics to measure the quality of object detection
- Understand advanced network architectures suitable for OD
- Different approaches

# outline

---

- Object localization (detection)
- Evaluation metrics
- State of the art object detection
- Object detection with region proposals and selective search
- R-CNN, Fast-R-CNN, Faster R-CNN
- Object detection with YOLO and SSD
- Hands -On

# CV tasks

Classification



Cat

Classification  
+ localization



Cat

Object  
detection



Dog, Person, Bed

Semantic  
segmentation



Dog, Person

Single object

Multiple objects



# Object detection



- Given an image we want to detect all the objects in the image that belongs to the specific classes and give their location. An image can contain more than one object with different classes

# Datasets

---



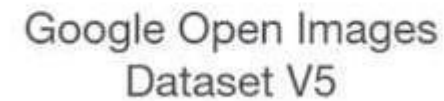
330K images (>200K labeled)  
and 80 object categories



14M images and more than  
20K object categories



11,530 images and 20  
classes



~9M images annotated with image-level labels, object  
bounding boxes, object segmentation masks, and visual  
relationships.

# Classification



CAT

# Image classification

- Outputs a class label. The complete image represents one class, we are not really worried about where the object is.
- usually only one object per image

# Classification and localization

## Classification + Localization



CAT

Given an image, we want to learn the class of the image and where are the class located in the image . Usually one object in the image

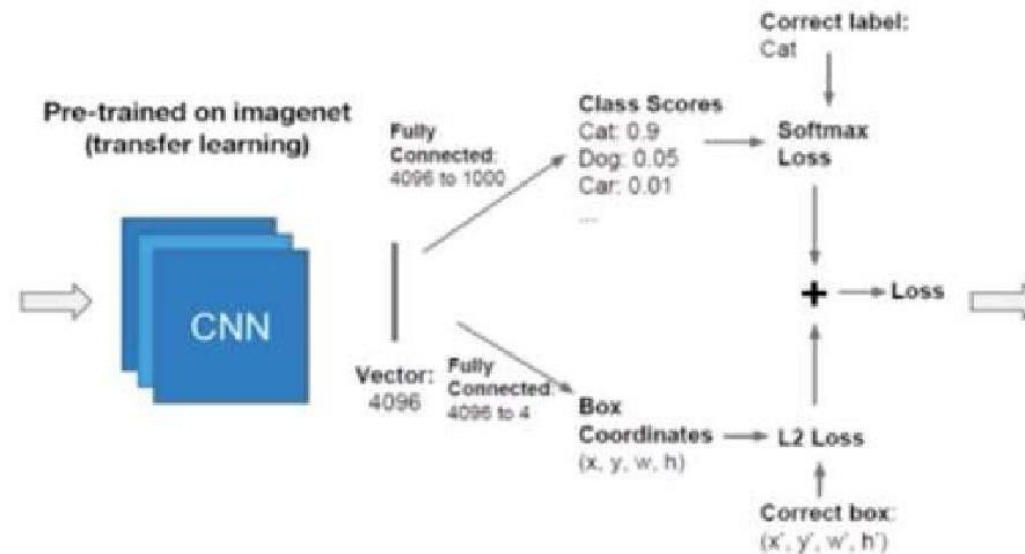


# Classification and localization

## Classification



CAT

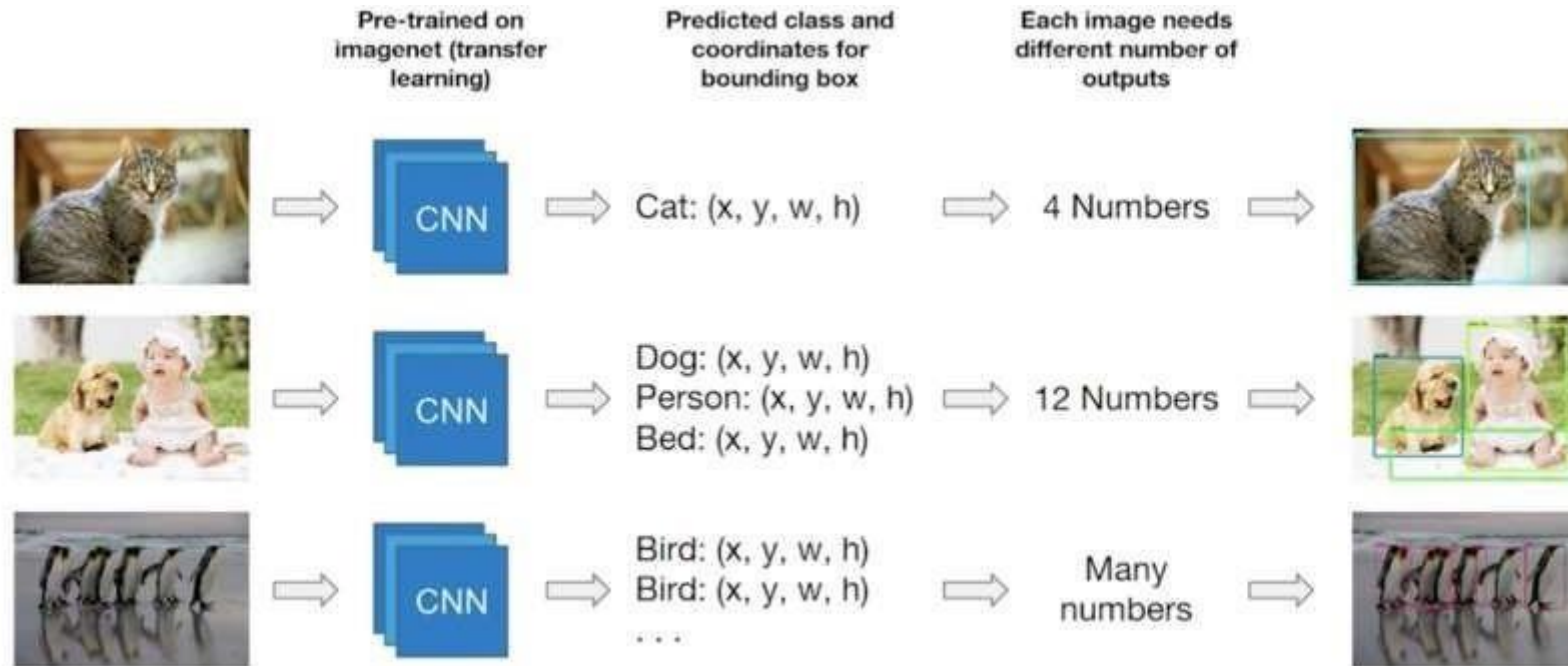


## Classification + Localization



CAT

# Regression for multiple images



# Points to remember

---

- For image classification we will use a ConvNet, with a softmax attached to the end of it.
- To make classification with localization we use a convnet with a softmax attached to the end for the classification part, and another output four numbers  $x, y, h$ , and  $w$  to predict the location of the class in the image

# challenges

---

- Two tasks – Classification and localization
- Predictions take a lot of time, and for real time we need fast predictions
- Variable number of boxes as output
- Different scale and aspect ratios
- Limited labelled data
- Imbalanced data -classes

# Performance metrics for OD


---

- For classification we can simply call accuracy as the metric . So we need to understand the metrics for these kind of models.
- Metrics:
  - Intersection over union (IoU)
  - Precision and recall
  - Mean average precision (mAP)



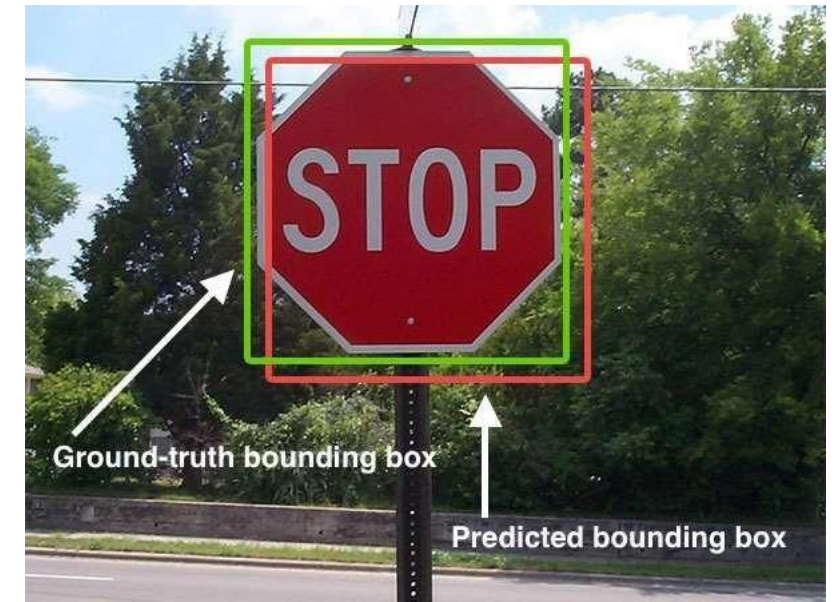
# IoU

---


$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


# IoU

- IoU is a function used to evaluate the object detection algorithm
- It computes the size of the intersection and divide it by the union. IoU is the measure of overlap between two bounding boxes.
- Green is truth and red is predicted .
- $\text{IoU} \geq 0.5$  is good.



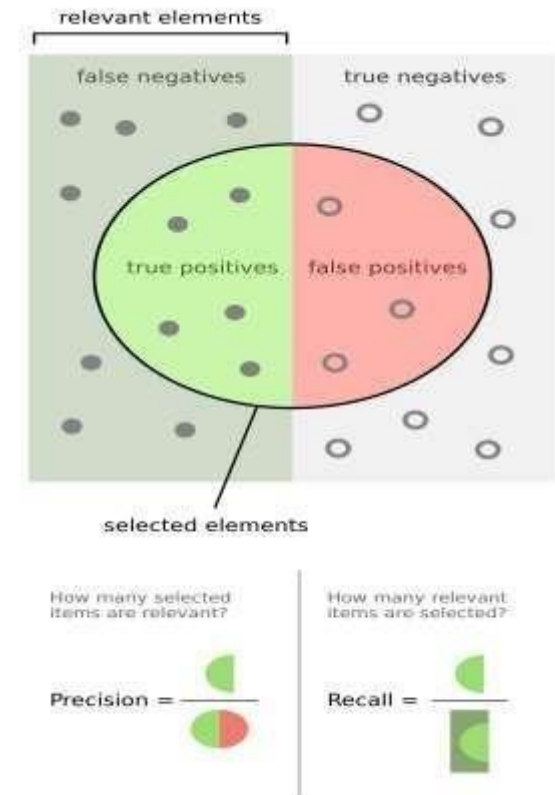
# Why to use Iou?

---

- We usually use accuracy , precision , recall etc as metrics for classification . But for object detection its not straightforward.
- The predicted (x,y) will not match exactly the actuals.
- We can calculate the precision and other metrics by using the IoU

# Precision

- Precision: What percentages of your positive predictions are correct.
- Recall: What percentage of ground truth objects are found



# Mean average precision

**Step1:** Sort predictions according the confidence (classifier's output after softmax)

**Step2:** Calculate IoU of every predicted box with every ground truth box

**Step3:** Match predictions to ground truth using IoU , correct predictions are those with IoU > 0.5 (threshold)

Confidence	Rank	correct
0.91	1	true
0.87	2	true
0.83	3	false
0.81	4	true
0.77	5	false
0.65	6	true
0.56	7	true
0.40	8	false
0.32	9	false
0.31	10	true



# Mean average precision

**Step 4 :** calculate precision and recall at every row

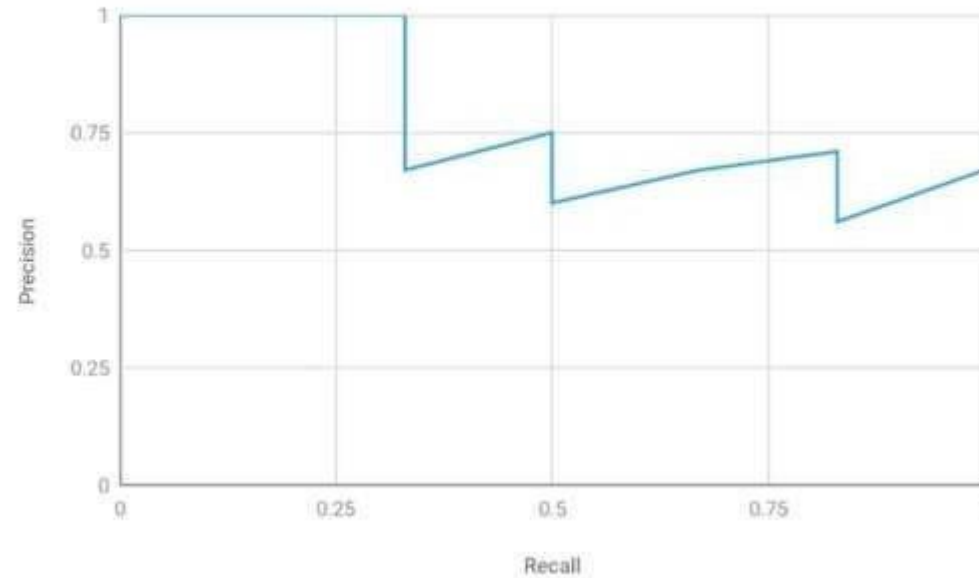
**Step 5:** Take the mean of maximum precision at 11 recall values (0.0,0.1,-----1) to get AP

**Step6:** Average across all classes to get mAP

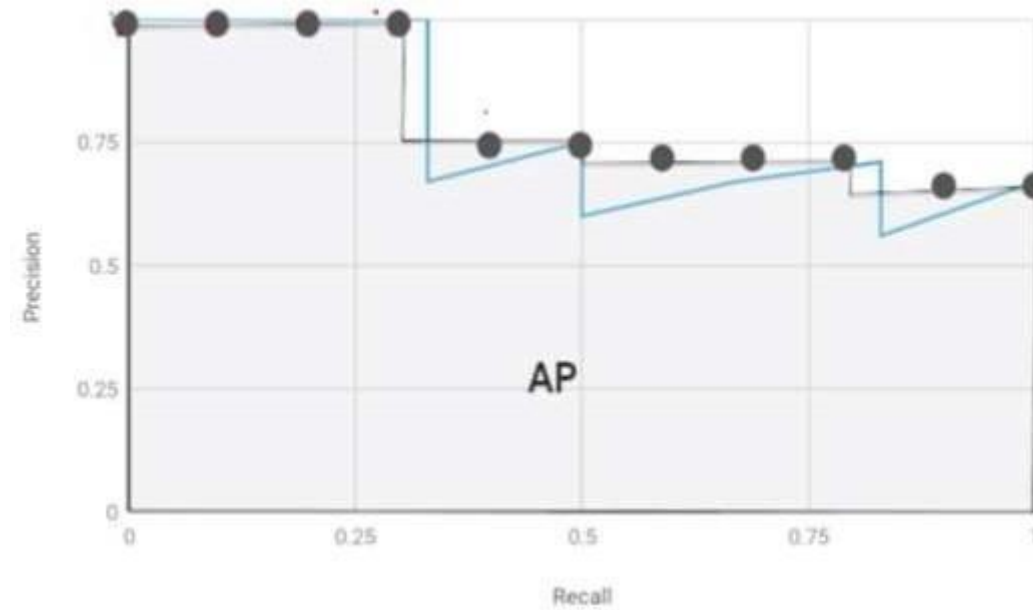
Confidence	Rank	correct	precision	Recall
0.91	1	true	1	0.17
0.87	2	true	1	0.33
0.83	3	false	0.67	0.33
0.81	4	true	0.75	0.50
0.77	5	false	0.60	0.50
0.65	6	true	0.67	0.67
0.56	7	true	0.71	0.83
0.40	8	false	0.63	0.83
0.32	9	false	0.56	0.83
0.31	10	true	0.67	1

# precision and recall curve

Precision and recall curve



# Precision and recall



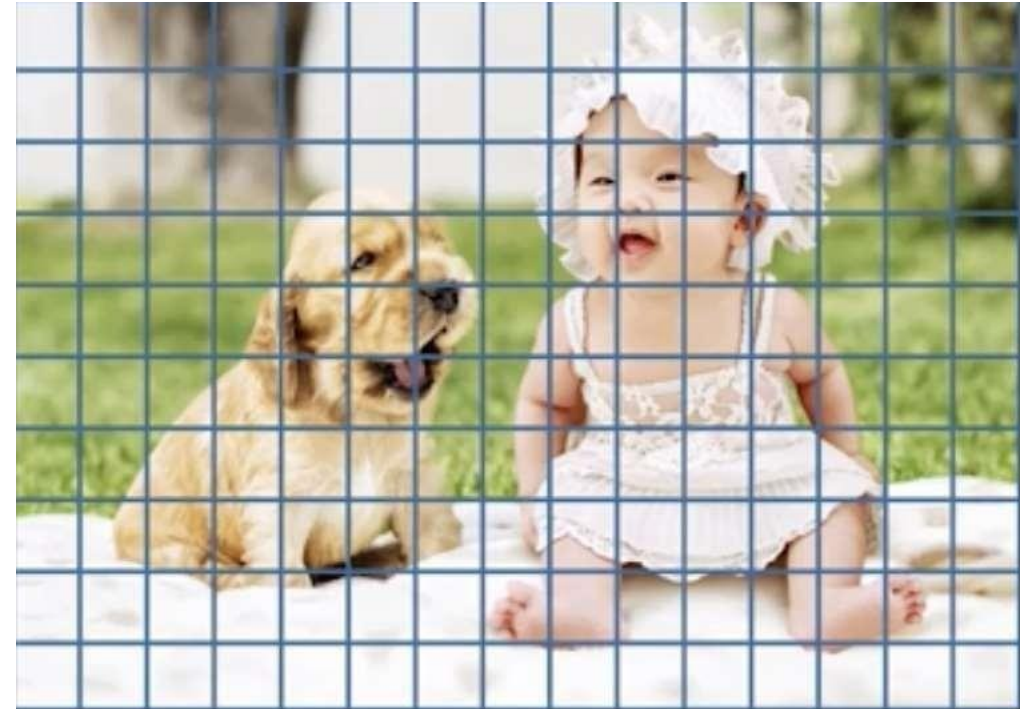
# Hands on bounding box

---

- Oxford pet dataset
- Train file
- Test file

# Different object detection techniques

- Object detection 1: Brute force approach
- Run a classifier for every possible box
- This is a 15 X 10 grid , there are 150 total boxes. How many total boxes?
- Computationally expensive





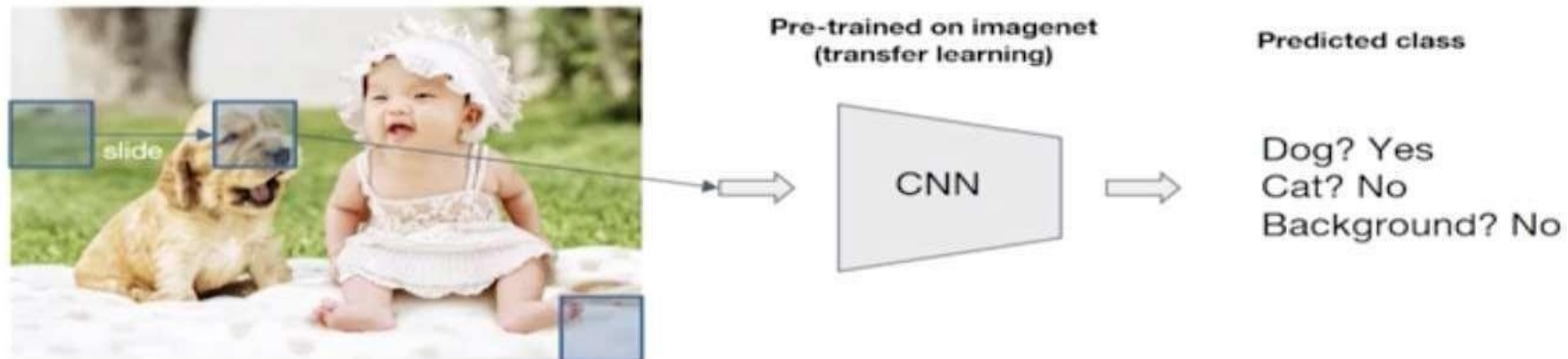
# Different object detection techniques

- Approach 2:
  - Sliding window approach
  - Run a classifier in sliding window fashion



# OD : Sliding window approach

- Apply CNN to many different crops of the image
- CNN classifies each crop as object or background
- Problem: Need to apply CNN to huge number of locations



# Better approach

---

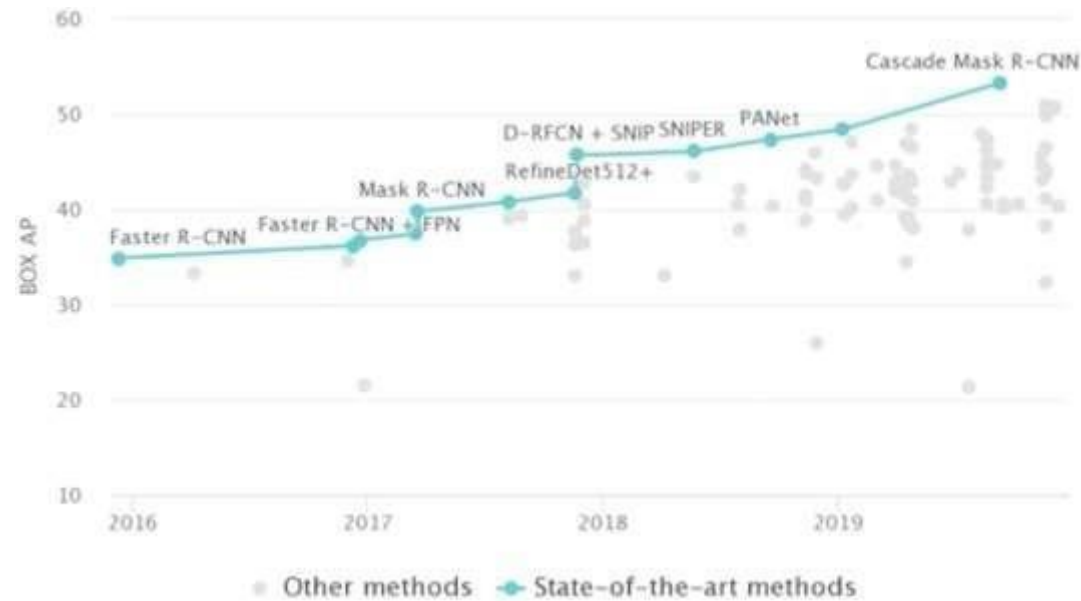
- Ideas on how to reduce the number of boxes?
  - Find blobby image regions which are likely to contain objects
  - Run classifier for region proposals or boxes likely to contain objects
- Class-agnostic object detector – ‘Region Proposals’

# State of the art

---

- State of the art methods are generally categorized into two categories
  - One stage methods
  - Two stage methods
- One stage methods give priority to inference speed ,these include SSD, YOLO, RetinaNet. Where as Faster-RCNN, Mask RNN and cascade RCNN are examples of two stage methods where priority is given to detection accuracy
- Popular benchmark data is Microsoft COCO
- Popular metric is mAP

# Sota on ms - coco dataset





# Object detection SOTA MS COCO

Method	Backbone	AP-box	AP-50	AP-75
<b>one stage:</b>				
SSD512 (Liu et al. 2016)	VGG16	26.8	48.5	30.3
RetinaNet (Lin et al. 2017b)	ResNeXt101	40.8	61.1	44.1
RefineDet (Zhang et al. 2018)	ResNet101	41.8	62.9	45.7
CornerNet (Zhang et al. 2018)	Hourglass-104	42.2	57.8	45.2
M2Det (Zhao et al. 2018)	VGG16	44.2	64.6	49.3
FSAF (Zhu, He, and Savvides 2019)	ResNext-101	44.6	65.2	48.6
NAS-FPN (Ghiasi, Lin, and Le 2019)	AmoebaNet	48.3	-	-
<b>two stage:</b>				
Faster R-CNN (Ren et al. 2015)	VGG16	21.9	42.7	-
R-FCN (Dai et al. 2016)	ResNet101	29.9	51.9	-
FPN (Lin et al. 2017a)	ResNet101	36.2	59.1	39
Mask R-CNN (He et al. 2017)	ResNet101	39.8	62.3	43.4
Cascade R-CNN (Cai and Vasconcelos 2018)	ResNet101	42.8	62.1	46.3
Libra R-CNN (Pang et al. 2019)	ResNext-101	43	64	47
SNIP (model ensemble) (Singh and Davis 2018)	-	48.3	69.7	53.7
SINPER (Singh, Najibi, and Davis 2018)	ResNet101	47.6	68.5	53.4
Cascade Mask R-CNN (Girshick et al. 2018)	ResNeXt152	50.2	68.2	54.9
MegDet (model ensemble) (Peng et al. 2018)	-	52.5	-	-
CBNet (Liu et al. 2019)	Dual-ResNeXt152	52.8	70.6	58
CBNet (Liu et al. 2019)	Triple-ResNeXt152	53.3	71.9	58.5

# List of algorithms

---

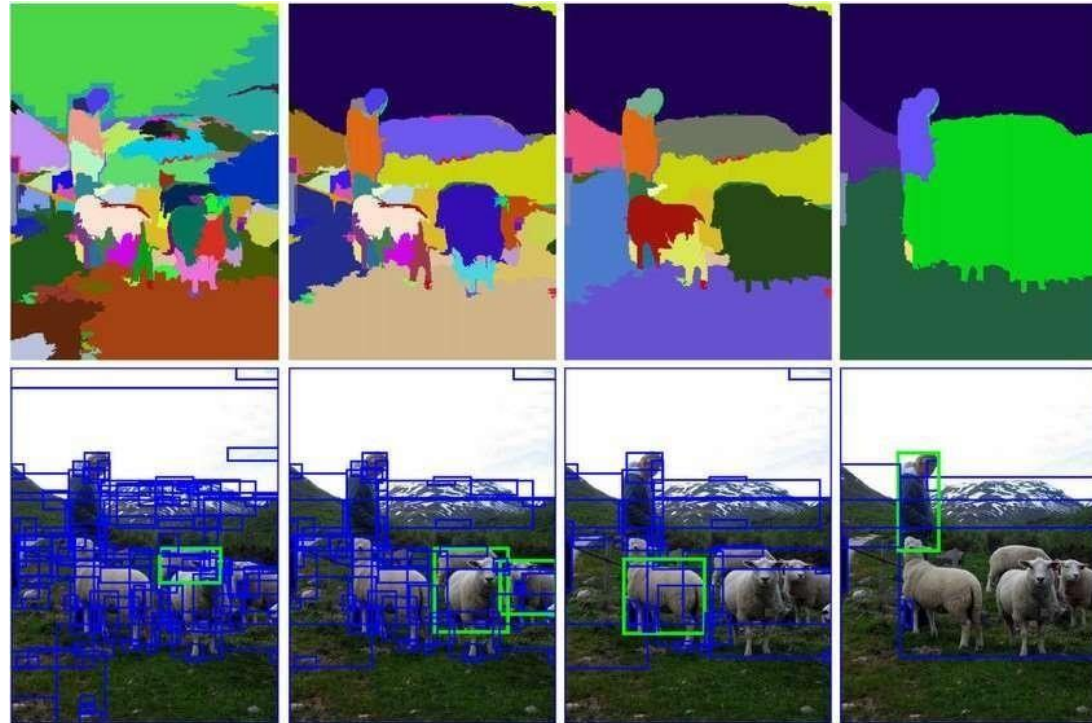
- Region proposal based algorithms:
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
- Algorithms without region proposals:
  - YOLO
  - SSD

# Region proposal

- Find blobby image regions that are likely contain objects
- Relatively fast to run , selective search gives around 1000 region proposals in few sec on CPU



# Region proposals – Selective search



Bottom up segmentation, merging regions at multiple scales

# Selective search

---

- Selective search uses best of both worlds, exhaustive search and segmentation
- Segmentation improves the sampling process of different boxes – reduces considerably the search space
- To improve the algorithm's robustness a variety of strategies are used during the bottom-up boxes' merging
- Selective search produces boxes that are good proposals for objects, it handles well defined image conditions, but more importantly it's fast enough to put in the production pipeline to do real time object detection

# Selective search advantages

---

- Captures all scales – Objects can occur at any scale within the image. Furthermore, some objects may not have clear boundaries. This is achieved by using hierarchical algorithm.
- Diversification : Regions may form an object because of only color, only texture, or lightning conditions etc., Therefore instead of single strategy which works well in most of the cases , we prefer to have diverse set of strategies to deal with all cases.
- Fast to compute

# Selective search



Figure 2: Two examples of our selective search showing the necessity of different scales. On the left we find many objects at different scales. On the right we necessarily find the objects at different scales as the girl is contained by the tv.



# Some algos

## Region Proposal - Many other choices

Method	Approach	Outputs Segments	Outputs Score	Control #proposals	Time (sec.)	Repea- tability	Recall Results	Detection Results
Bing [18]	Window scoring		✓	✓	0.2	***	*	.
CPMC [19]	Grouping	✓	✓	✓	250	-	**	*
EdgeBoxes [20]	Window scoring		✓	✓	0.3	**	***	***
Endres [21]	Grouping	✓	✓	✓	100	-	***	**
Geodesic [22]	Grouping	✓		✓	1	*	***	**
MCG [23]	Grouping	✓	✓	✓	30	*	***	***
Objectness [24]	Window scoring		✓	✓	3	-	*	.
Rahtu [25]	Window scoring		✓	✓	3	-	-	*
RandomizedPrim's [26]	Grouping	✓		✓	1	*	*	**
Rantalankila [27]	Grouping	✓		✓	10	**	-	**
Rigor [28]	Grouping	✓		✓	10	*	**	**
SelectiveSearch [29]	Grouping	✓	✓	✓	10	**	***	***
Gaussian				✓	0	-	-	*
SlidingWindow				✓	0	***	-	.
Superpixels		✓			1	*	-	.
Uniform				✓	0	-	-	.

Grey check-marks indicate that the number of proposals is controlled by indirectly adjusting parameters. Repeatability, quality, and detection rankings are provided as rough summary of the experimental results: "-" indicates no data, ".", "\*\*", "\*\*\*", "\*\*\*\*" indicate progressively better results.

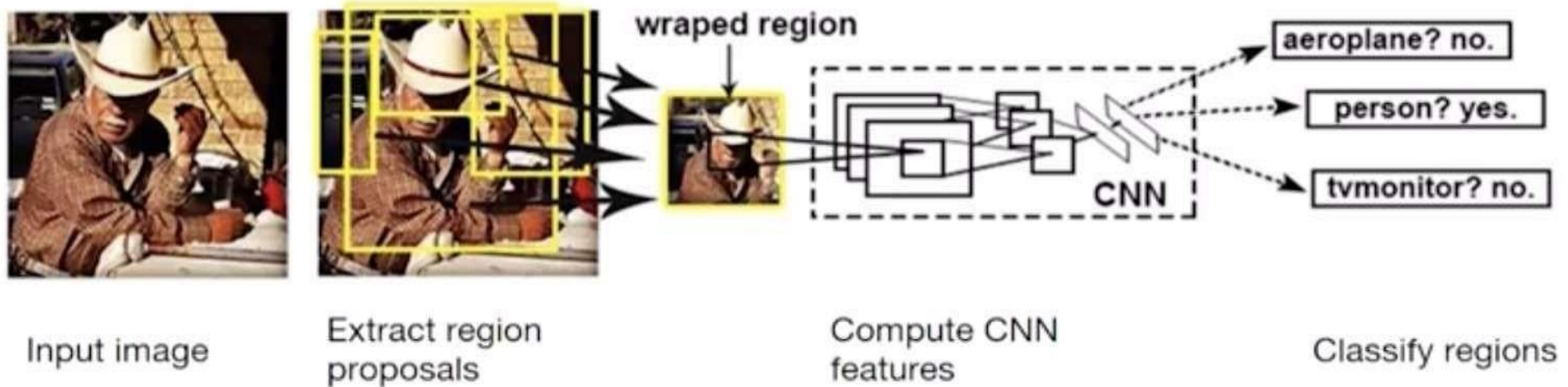


# Region CNN (R-CNN)

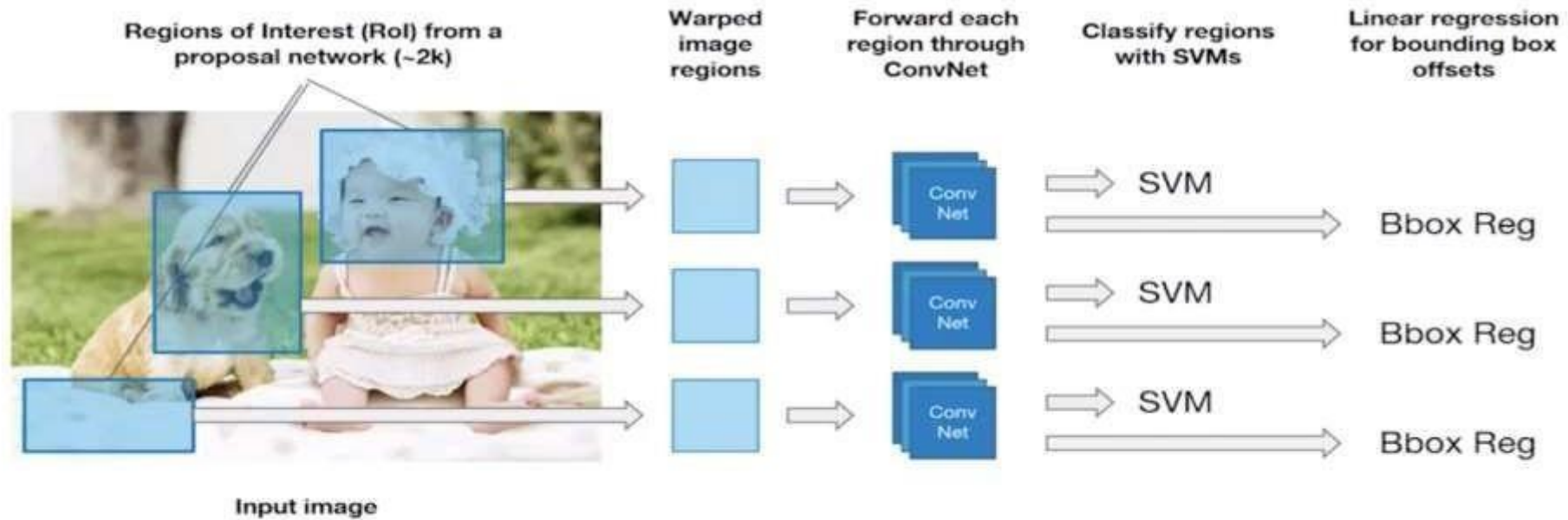
- 
- R-CNN is an algorithm which can be used for object detection
  - R-CNN is regions with conv nets
  - R-CNN tries to pick a few windows and run a conv net on top of them

# R-CNN

- R-CNN first generates 2K region proposals (bounding box candidates), then detect object within the each region



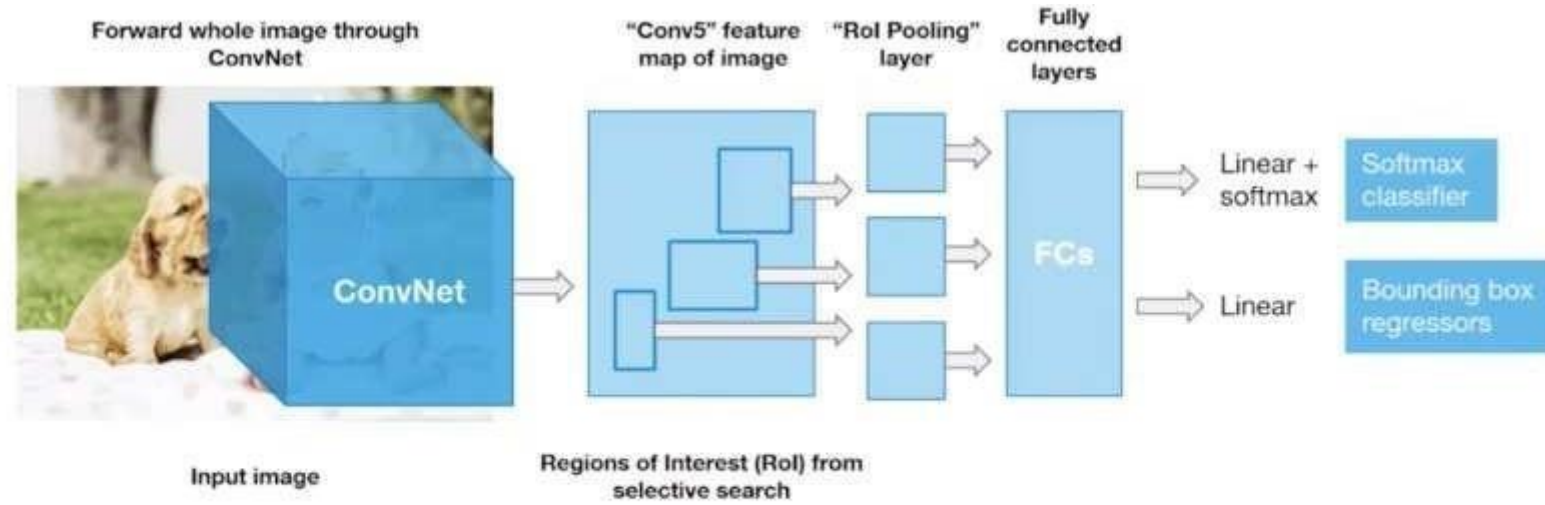
# How r-cnn works?



# RCNN

- 
- Separate conv net for each box lot of redundant computations
  - It still uses selective search
  - It uses SVM , no back propagation.
  - Why constant size- CNN can only take a constant size. Bad for regions which are not square
  - It is slow why – it does a lot of redundant computations . If you overlap on a face, the smaller one is already done by a bigger box

# Fast r -cnn

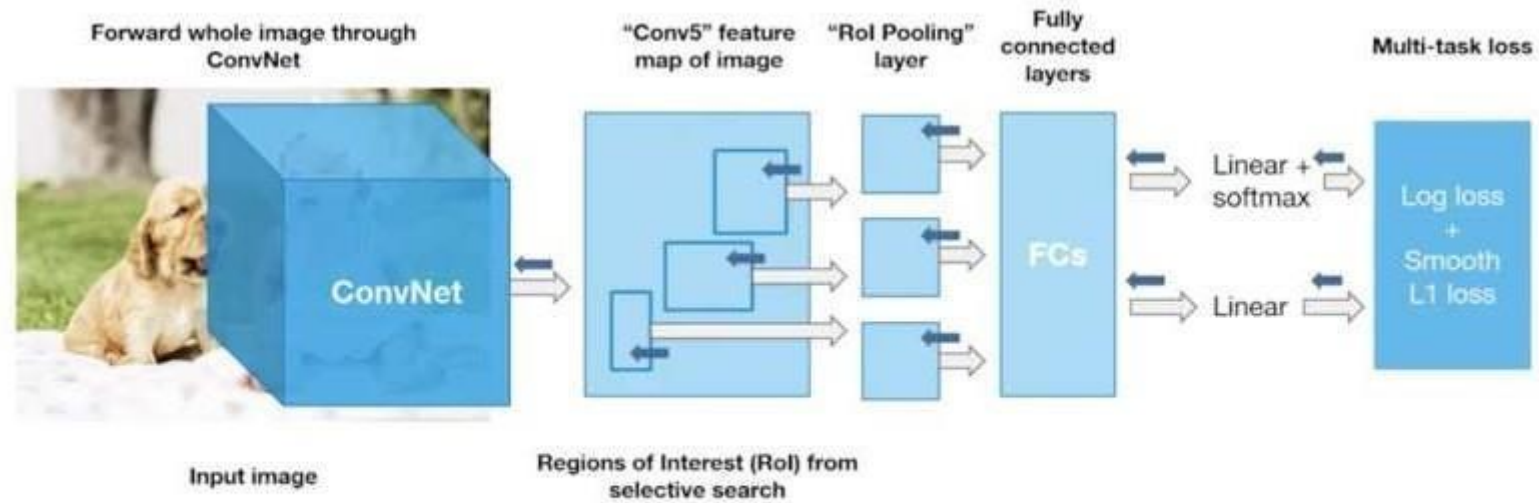


Source: [pifex \(2014,08/03\) \[14 Oct\]](#)

# Fast r-cnn

- 
- Entire image is fed into Conv net and get features (multiple feature maps)
  - Adv : Redundant computations are now no longer there
  - ROI pooling layer- region proposals big or small , the ROI layer will give fixed size outputs
  - It goes through CNN, this gives us the object class and also the coordinates

# Fast r-cnn training



Source: [arXiv:1506.01497 \[cs.CV\]](https://arxiv.org/abs/1506.01497)

# Fast r-cnn roi pooling

---

- Region of interest pooling is a neural-net layer used for object detection tasks
  1. A fixed-size feature map obtained from a deep convolutional network with several convolutions and max pooling layers.
  2. An  $N \times 5$  matrix of representing a list of regions of interest, where  $N$  is a number of RoIs. The first column represents the image index and the remaining four are the coordinates of the top left and bottom right corners of the region.

## What does the RoI pooling actually do?

For every region of interest from the input list, it takes a section of the input feature map that corresponds to it and scales it to some pre-defined size (**e.g.,  $7 \times 7$** ). The scaling is done by:

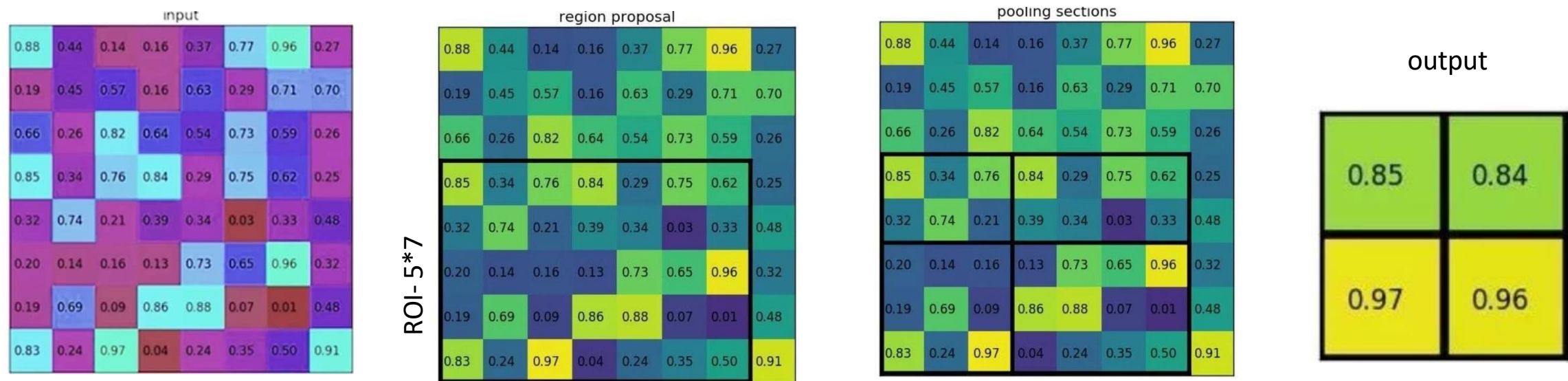
1. Dividing the region proposal into equal-sized sections (the number of which is the same as the dimension of the output)
2. Finding the largest value in each section
3. Copying these max values to the output buffer

**Result:** From a list of rectangles with different sizes we get a list of corresponding feature maps with a fixed size



# Fast r-cnn roi pooling

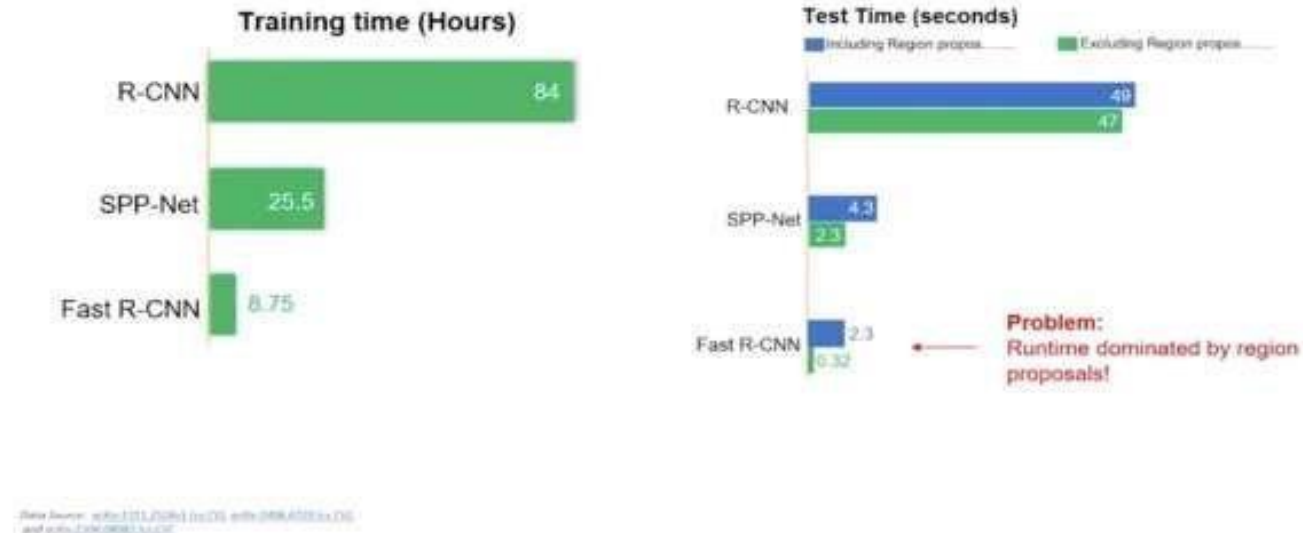
- ROI polling on a single 8X8 feature map, one region of interest and an output size of 2 X 2 out input feature map looks like this.



Notice that the size of the region of interest doesn't have to be perfectly divisible by the number of pooling sections (in this case our ROI is 5x7 and we have 2x2 pooling sections)

# Comparison between r- cnn and f r - cnn

## R-CNN vs SPP vs Fast R-CNN



SPP: Spatial Pyramid Pooling

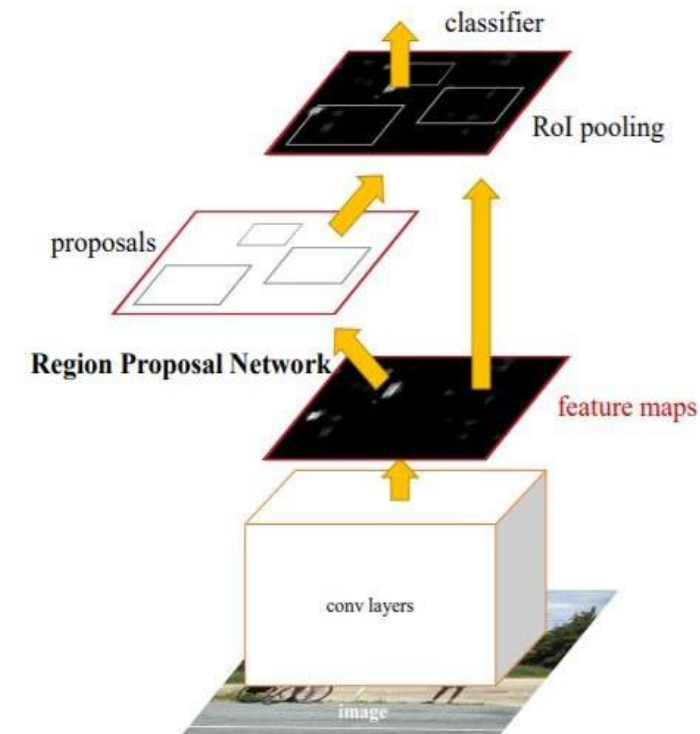
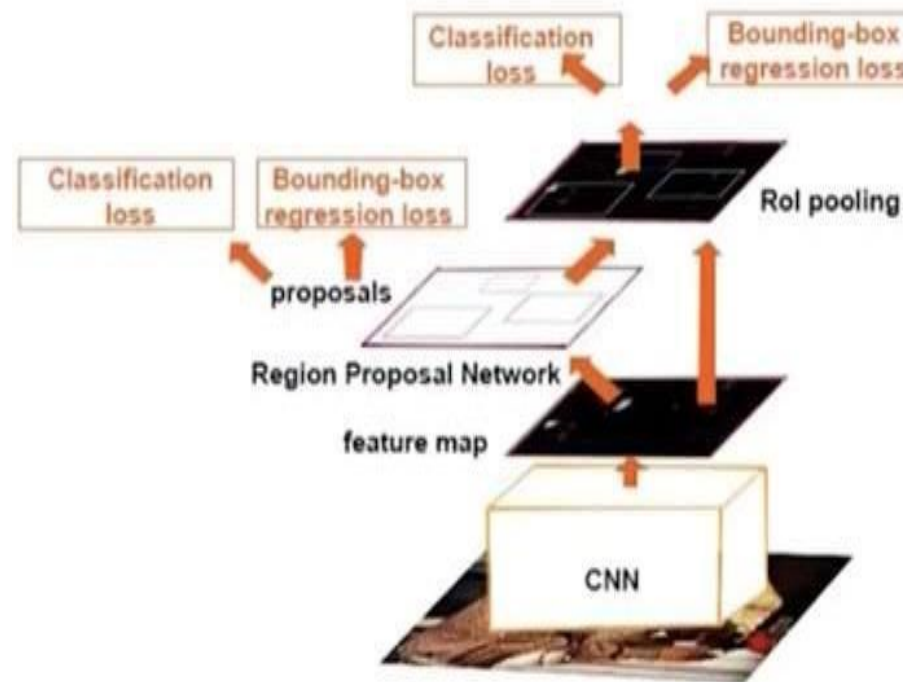
# Faster r- cnn

---

- Faster R-CNN has two networks:
  - Region proposal network (RPN) for generating region proposals
  - network using these proposals to detect objects.
- ☐ The main different here with Fast R-CNN is that the later uses selective search to generate region proposals.
- ☐ The time cost of generating region proposals is much smaller in RPN than selective search, when RPN shares the most computation with the object detection network. Briefly, RPN ranks region boxes (called anchors).

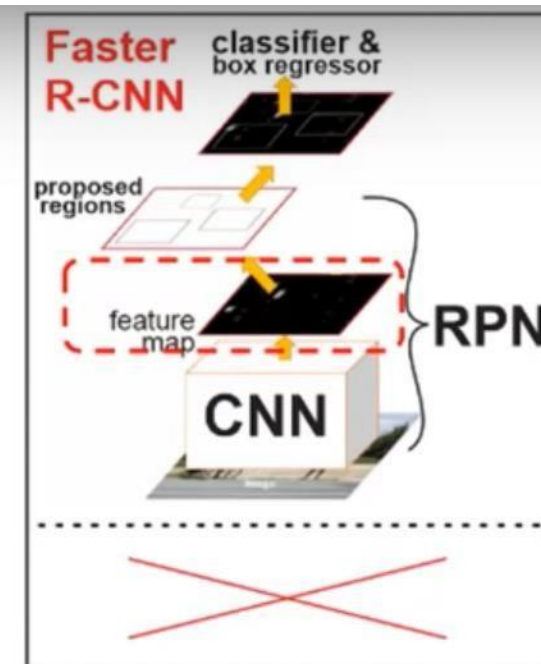
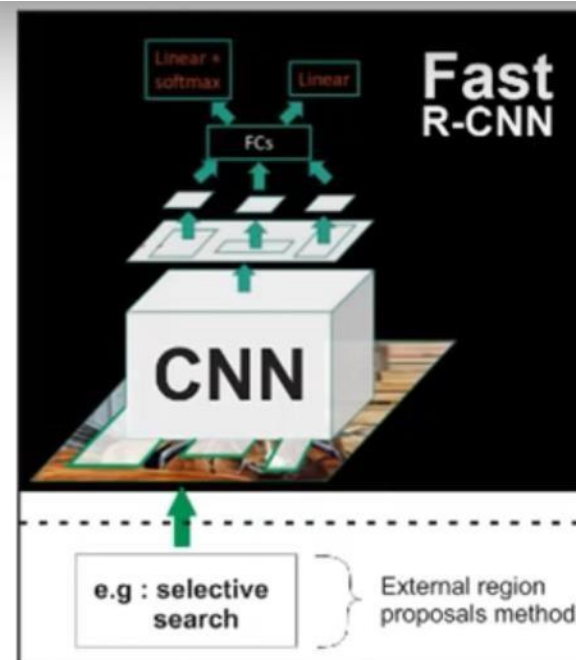
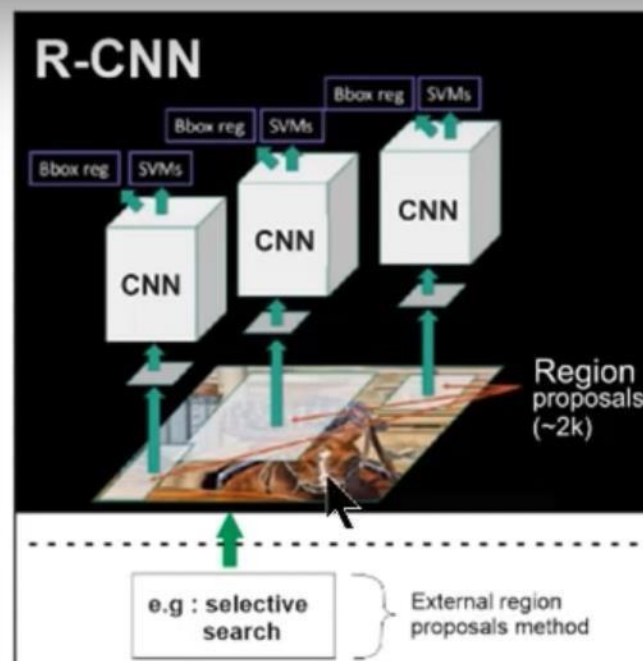
# Faster r- cnn

- Make CNN do proposals
- Insert Region proposal network (RPN) to predict proposals from features.
- Jointly train with 4 losses
  - RPN classifies object/ not object
  - RPN regress box coordinates
  - Final classification score
  - Final box coordinates



Paper :<https://arxiv.org/pdf/1506.01497.pdf>

# Summary- comparison



	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image	50 seconds	2 seconds	0.2 seconds
Speed-up	1x	25x	250x
mAP (VOC 2007)	66.0%	66.9%	66.9%

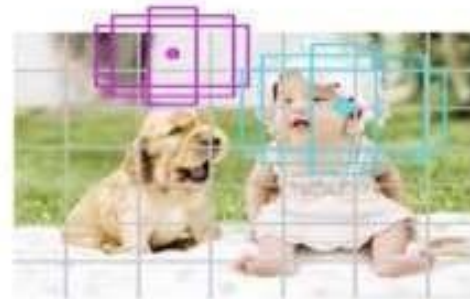
# Detection without proposal - YOLO

## Detection without proposals: YOLO/SSD

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



Input image  
 $3 \times H \times W$



Divide image into grid  
 $7 \times 7$

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:  
 $7 \times 7 \times (5 \cdot B + C)$



# Yolo (you only look once)

- 
- YOLO – You only look once , looks at the image only once but in a clever way.
    - Yolodoc says: We reframe the object detection as a single regression problem , straight from image pixels to bounding box coordinates and class probabilities

# YOLO – algorithm flow

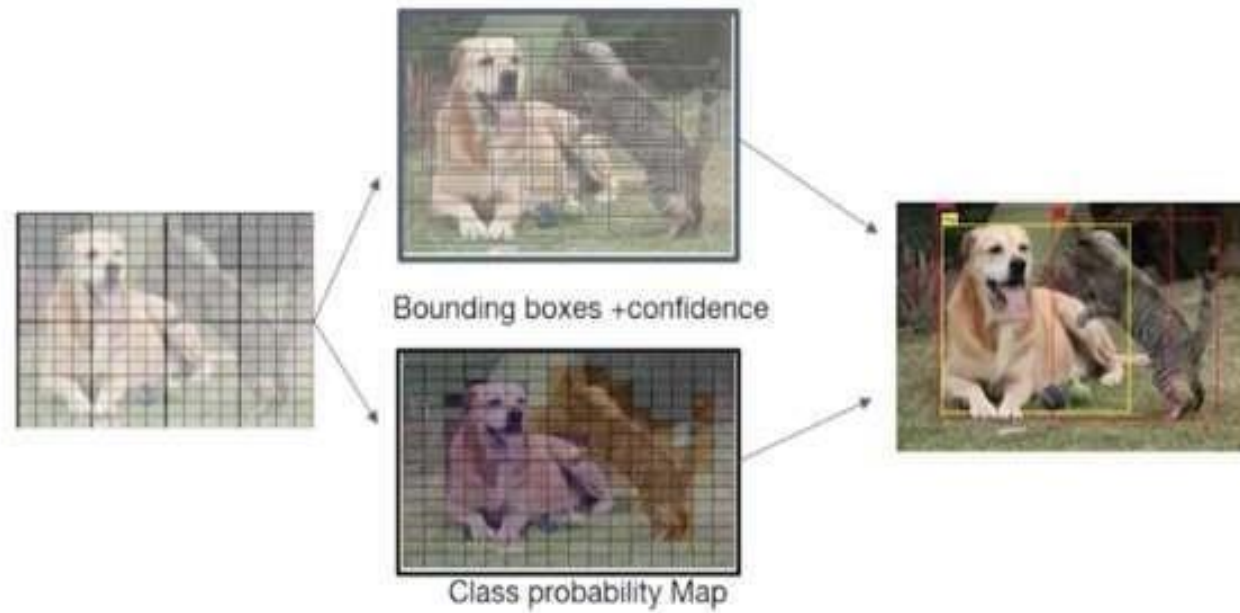
---

- Actually divides the image into say  $13 * 13$  cells ( $S=13$ )
- Each of this cell is responsible for predicting 5 bounding boxes ( $B=5$ )  
A bounding box describes the rectangle that encloses an object)
- YOLO for each bounding box
  - Outputs a confidence score on how good is the shape of the box
  - The cell also predicts the class
- The confidence score of bounding box and class prediction are combined into final score -> probability that the bounding box contains a specific object



# yolo

## YOLO



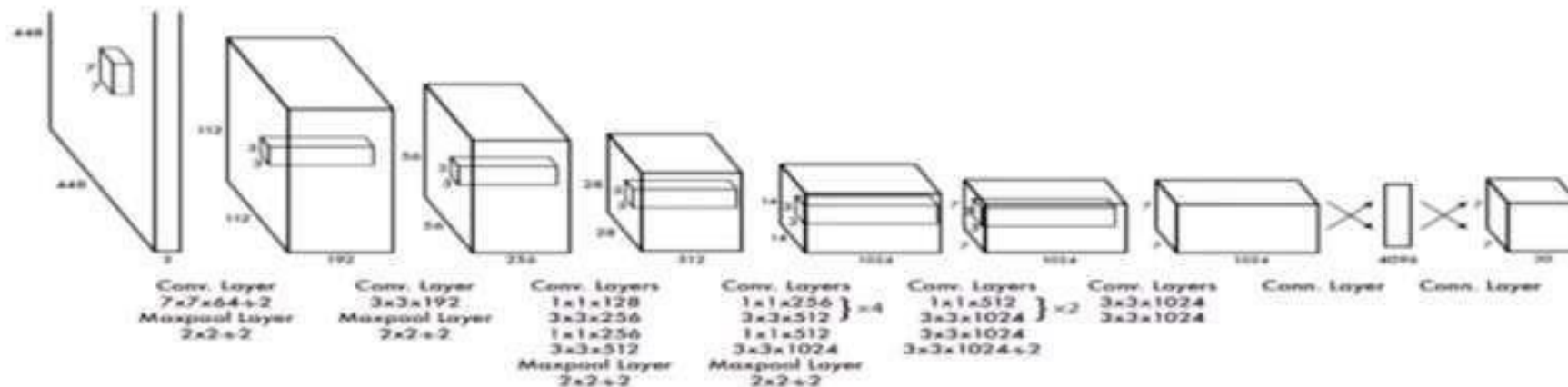
# Example: Yolo Details

---

- $13 * 13 = 169$  grid cells
- Each cell predicts 5 bounding boxes =  $169 * 5 = 895$  boxes
- Most of the boxes have low confidence score
- Threshold of 30 % or more
- Input image  $416 * 416$  pixels
- Number of classes=100
- $13 * 13 * [\text{output}=5(dx,dy,dh,dw,\text{confidence score}) * B=5 + C=100] = 13 * 13 * 125$  tensor describing the bounding box for grid cells

# Yolo - architecture

## YOLO Network Architecture



This detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating  $1 \times 1$  convolutional layers reduce the features space from preceding layers. the convolutional layers are pretrained on the ImageNet classification task at half the resolution ( $224 \times 224$  input image) and then double the resolution for detection.

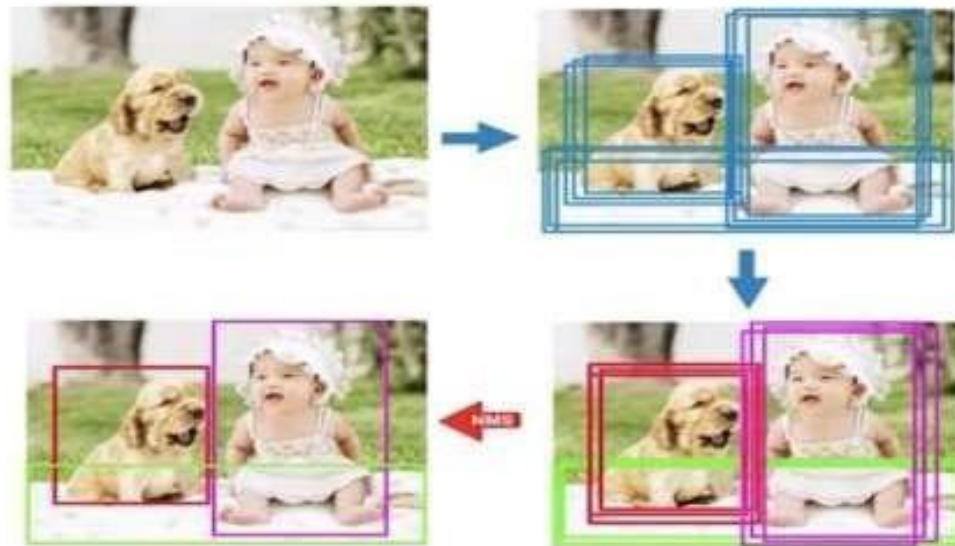
# Non- maximal suppression

**NMS:**

**Input:** A list of Proposal boxes  $B$ , corresponding confidence scores  $S$  and overlap threshold  $N$ .

**Output:** A list of filtered proposals  $D$ .

Non-maximal suppression



# NMS-Non- maximal suppression

## Algorithm:

1. Select the proposal with highest confidence score, remove it from B and add it to the final proposal list D. (Initially D is empty).
2. Now compare this proposal with all the proposals — calculate the IOU (Intersection over Union) of this proposal with every other proposal. If the IOU is greater than the threshold N, remove that proposal from B.
3. Again, take the proposal with the highest confidence from the remaining proposals in B and remove it from B and add it to D.
4. Once again calculate the IOU of this proposal with all the proposals in B and eliminate the boxes which have high IOU than threshold.
5. This process is repeated until there are no more proposals left in B.

**Input** :  $\mathcal{B} = \{b_1, \dots, b_N\}$ ,  $\mathcal{S} = \{s_1, \dots, s_N\}$ ,  $N_t$   
 $\mathcal{B}$  is the list of initial detection boxes  
 $\mathcal{S}$  contains corresponding detection scores  
 $N_t$  is the NMS threshold

```

begin
   $\mathcal{D} \leftarrow \{\}$ 
  while  $\mathcal{B} \neq \text{empty}$  do
     $m \leftarrow \text{argmax } \mathcal{S}$ 
     $\mathcal{M} \leftarrow b_m$ 
     $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{M}$ ;  $\mathcal{B} \leftarrow \mathcal{B} - \mathcal{M}$ 
    for  $b_i$  in  $\mathcal{B}$  do
      if  $\text{iou}(\mathcal{M}, b_i) \geq N_t$  then
         $\mathcal{B} \leftarrow \mathcal{B} - b_i$ ;  $\mathcal{S} \leftarrow \mathcal{S} - s_i$ 
      end
    end
     $s_i \leftarrow s_i f(\text{iou}(\mathcal{M}, b_i))$ 
  end
end
return  $\mathcal{D}, \mathcal{S}$ 
end
  
```

NMS

Soft-NMS

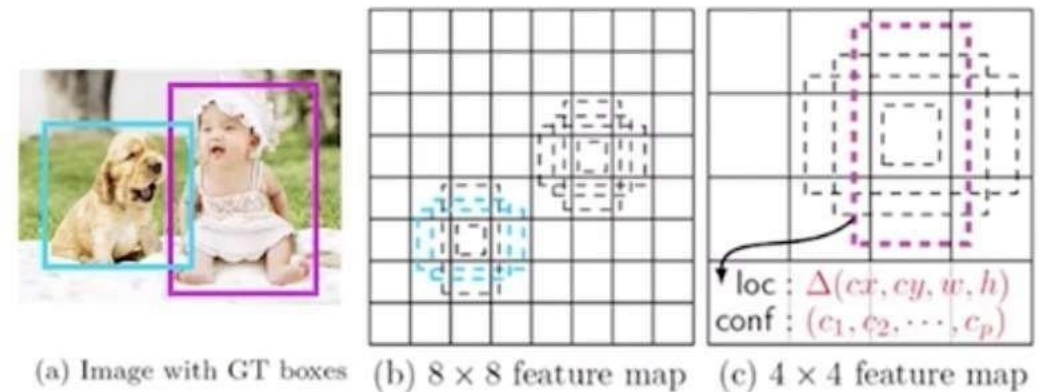
# Single shot detectors

---

- In SSD like YOLO , only one single pass is needed to detect multiple objects within the image
- Two passes are needed in region proposal networks, one for generating proposals other for detecting the object.
- SSD is much faster compared to two shot RPN based approaches

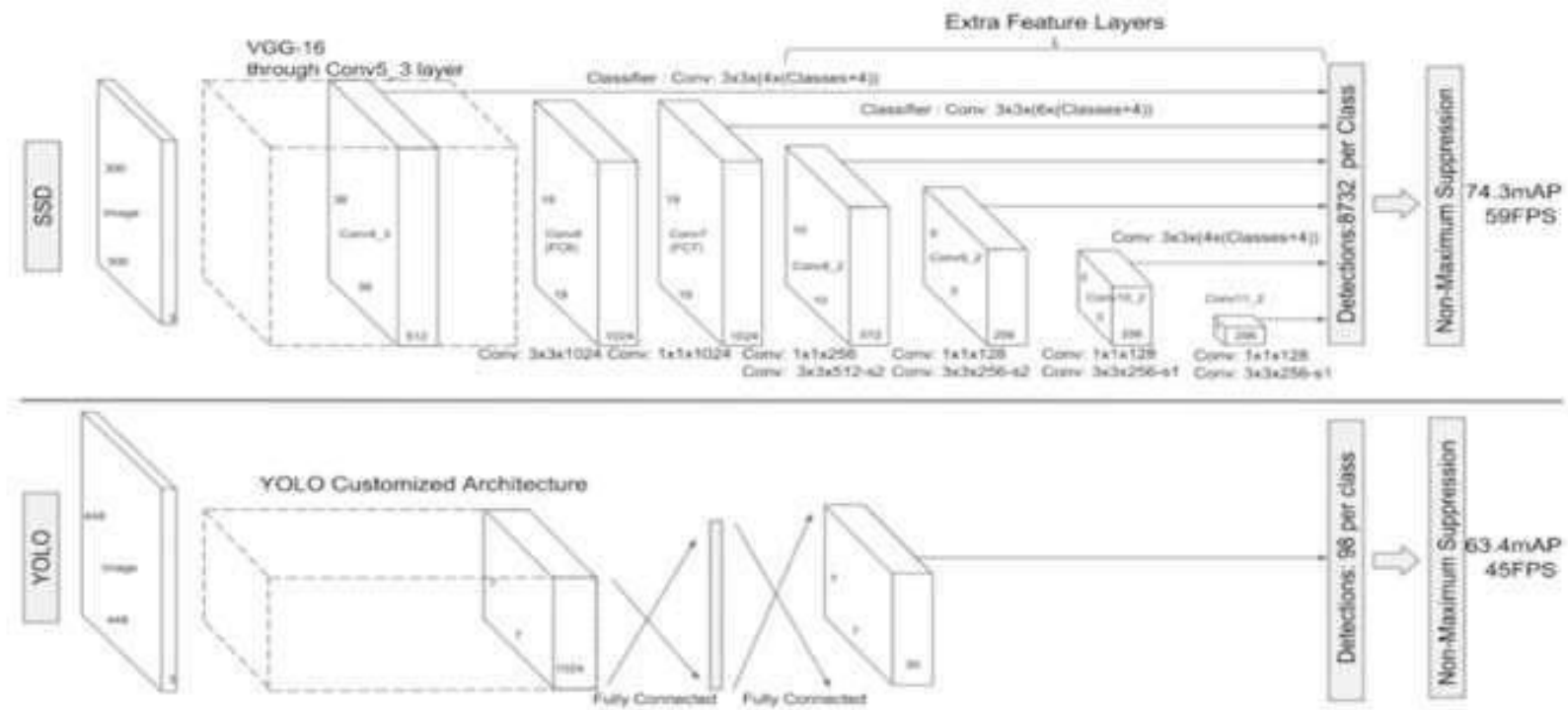
# SSD- single-shot detector

- A feature layer of size  $m \times n$  (number of locations) with  $p$  channels
- For each location, we got  $k$  bounding boxes
- For each of the bounding box, we will compute  $c$  class scores and 4 offset relative to the original default box shape
- This we get  $(c + 4) \times k \times m \times n$  output



# Ssd and yolo

## SSD and YOLO





# Ssd and yolo

- 
- SSD model adds several feature layers to the end of the base network , which predict the offset to default boxes of different scales and aspect ratios and their associated confidence.
  - SSD with a 300 X 300 input size significantly outperforms 448 x 448

# Standard OD models

---

## Object Detection - Options

### 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

# Semantic segmentation

---

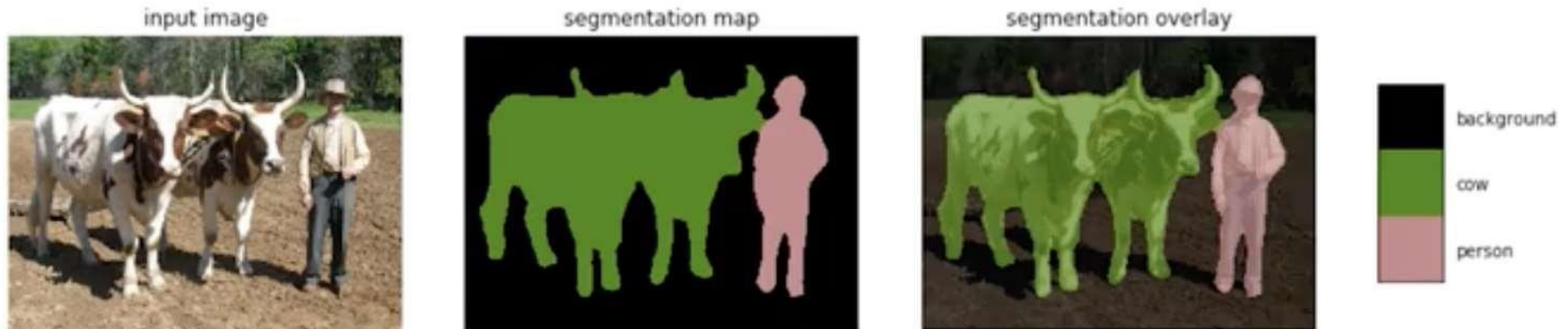
# outline

---

- Semantic segmentation
- Approaches
- Sliding window, fully conventional , up Sampling
- U-Net
- Depth wise separable networks
- Mobile net
- Hands on

# Semantic segmentation

- Put label on each pixel in the image
- No need to specify the difference between the instances



# example

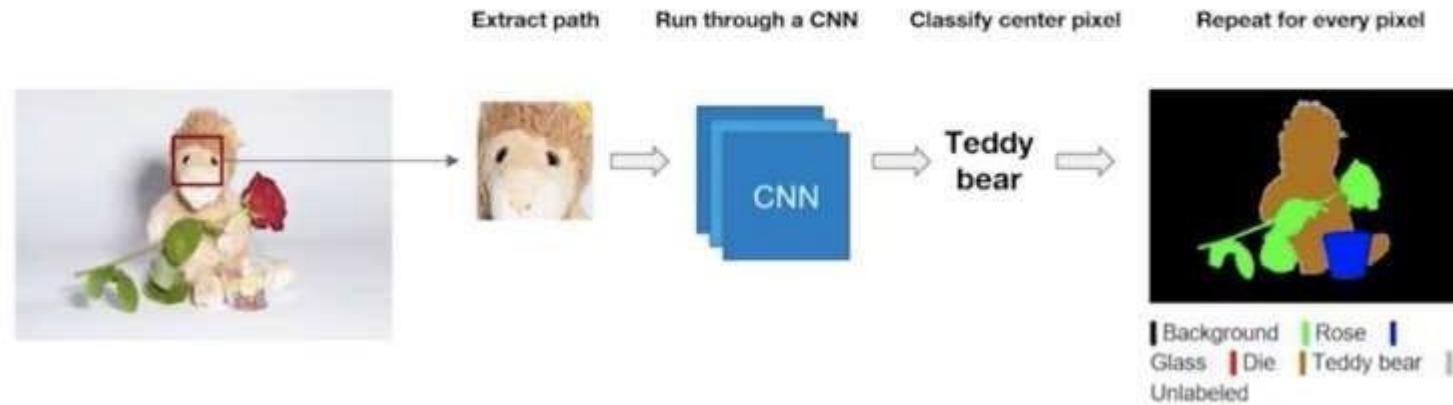
## Example



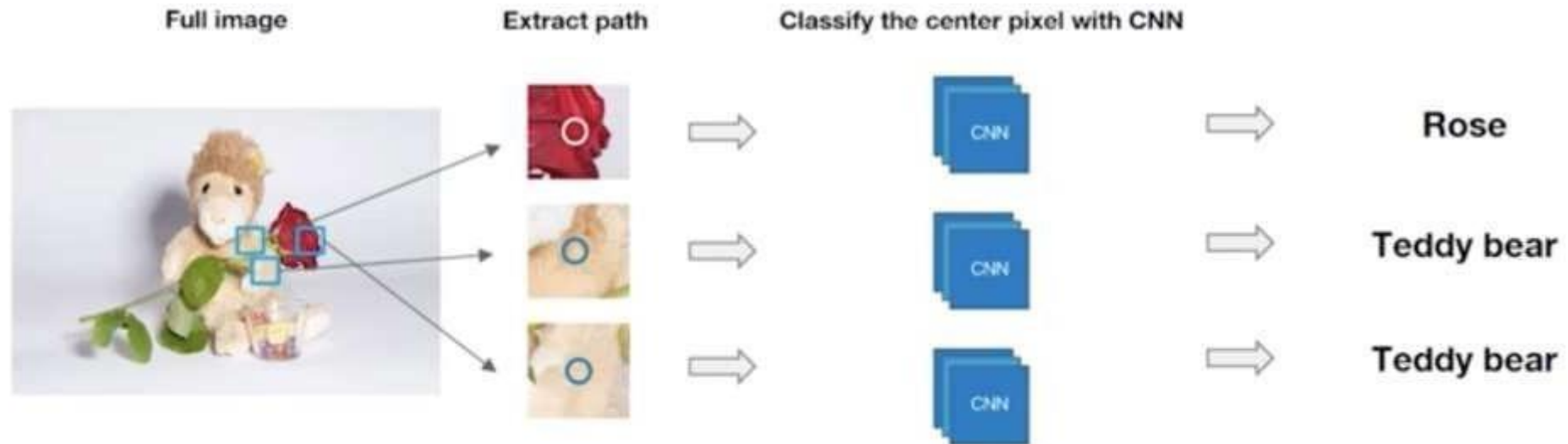
Background Rose Glass Die Teddy bear Unlabeled

# example

## Semantic Segmentation



# Semantic segmentation sliding window





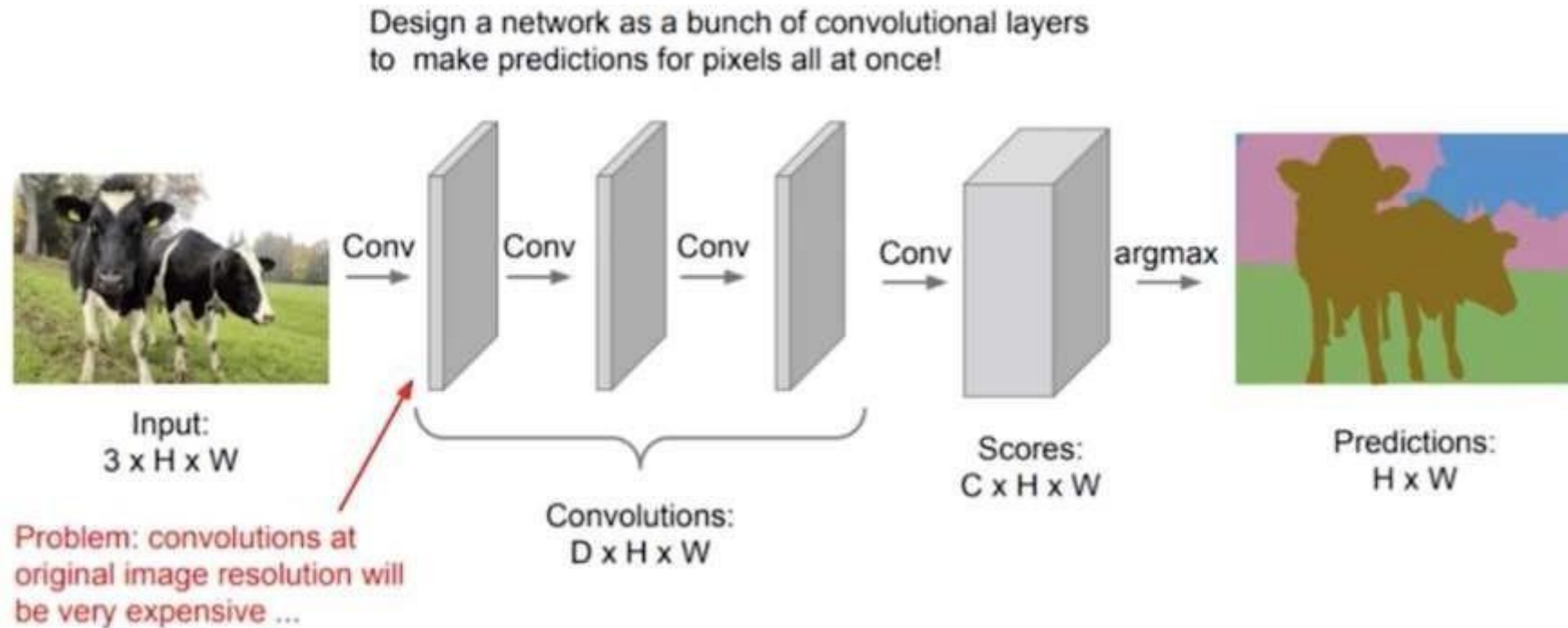
# Semantic segmentation

- Run through a fully convolution network to get all the pixels at once

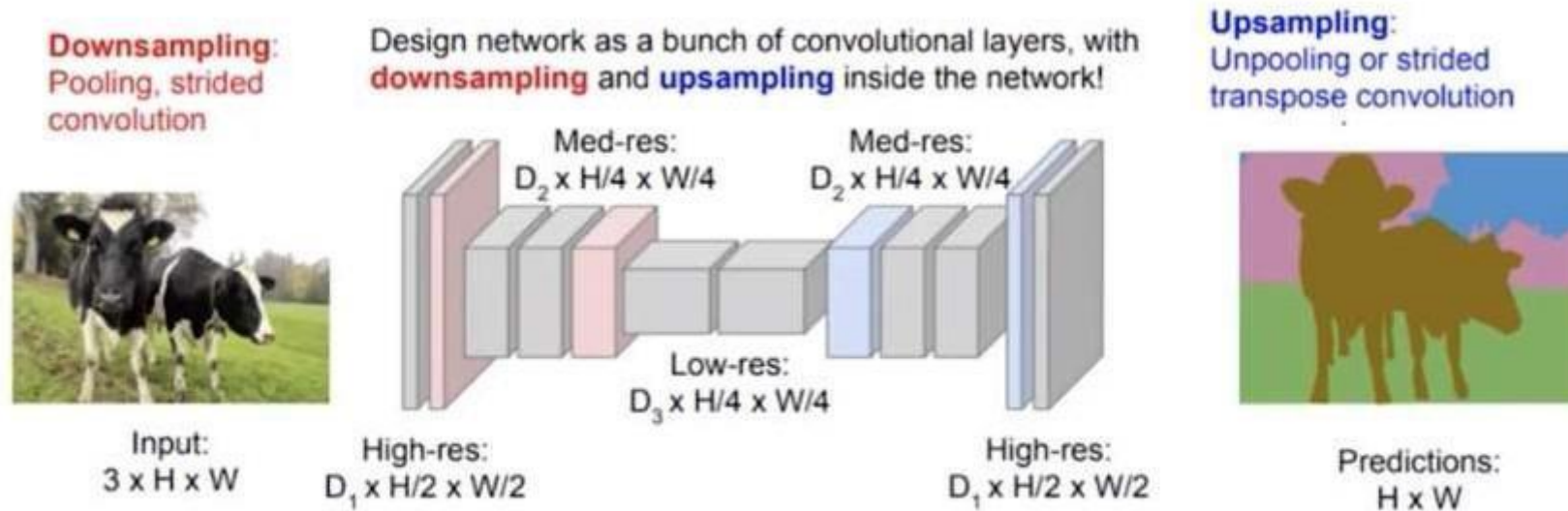


Smaller output due to pooling

# Semantic segmentation :FCN



# Semantic segmentation fcn



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

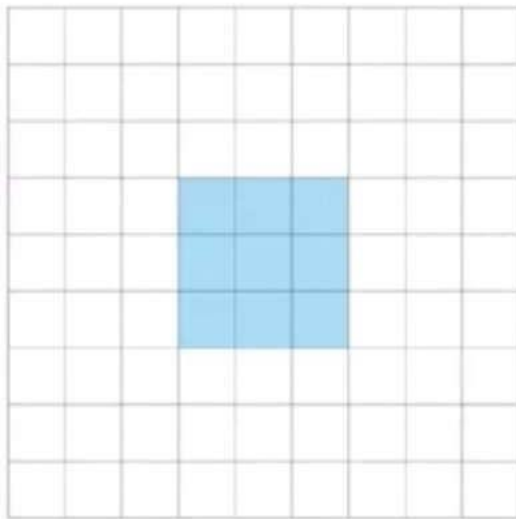
# Receptive field

- 
- It means that the convolutions that we are doing here, suppose we have  $3 \times 3$  conv. The  $3 \times 3$  can only look at the  $3 \times 3$  area in the feature map, but what area it correspond to in the input image, this has the context of the bigger chunk of the image.
  - Another way of improving the convolution is dilated.

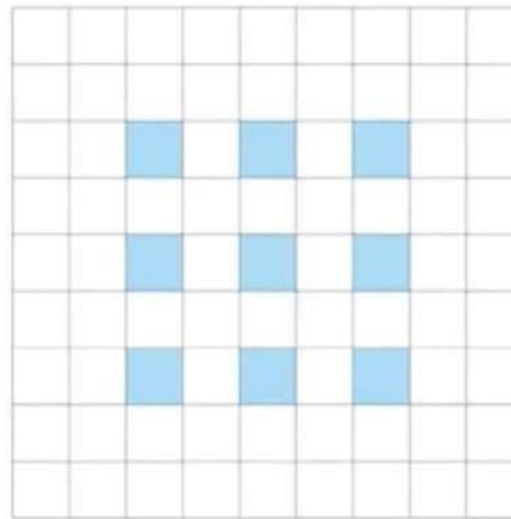
# Atrous/dilated convolutions

- Atrous convolutions are regular convolutions with a factor that allows us to expand the filter's field of view.
- Consider a 3 x 3 conv filter for instance, when dilation is set to 1, it behaves as a normal convolution, but if dilation is set to 2 it has the effect of enlarging the convolution kernel

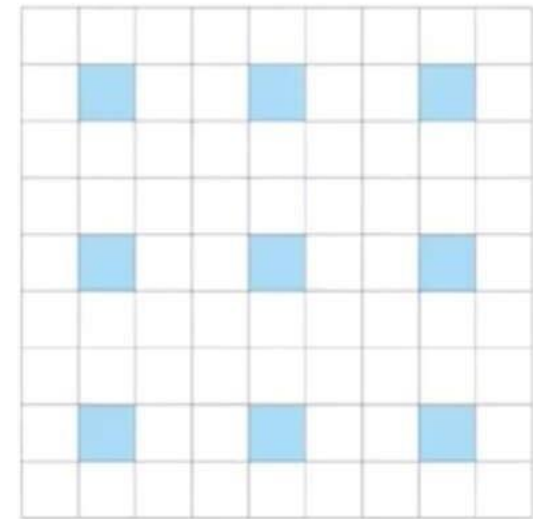
rate = 1



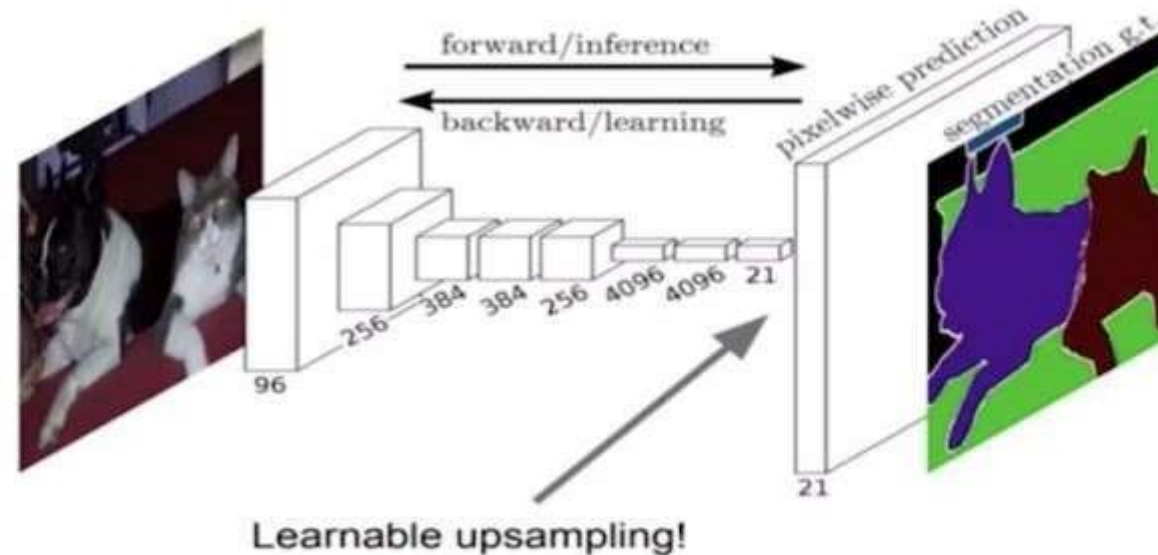
rate = 2



rate = 3



# Semantic segmentation upsampling



Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015

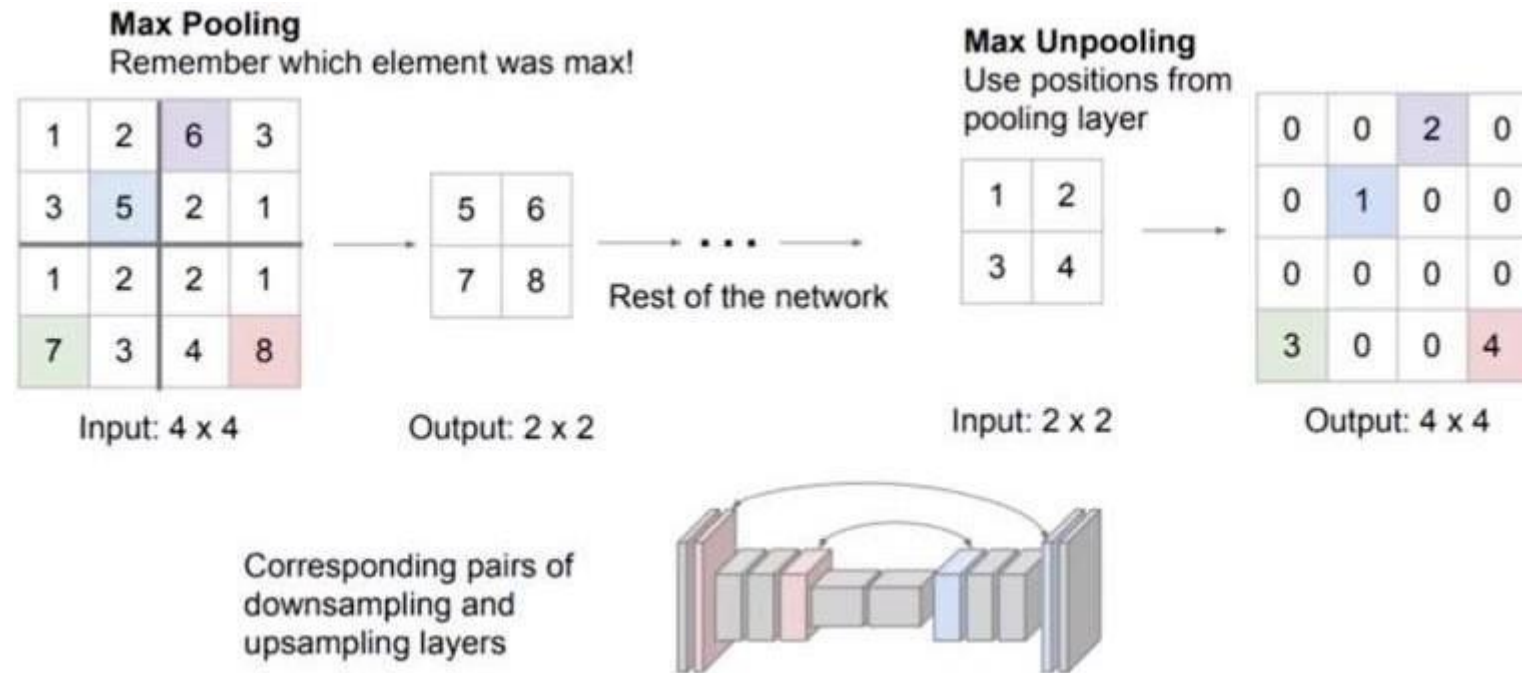
# upsampling

- Transposed convolution / Deconvolution
- Fractionally strided convolution
- Max unpooling preserves spatial information

1	2
3	4

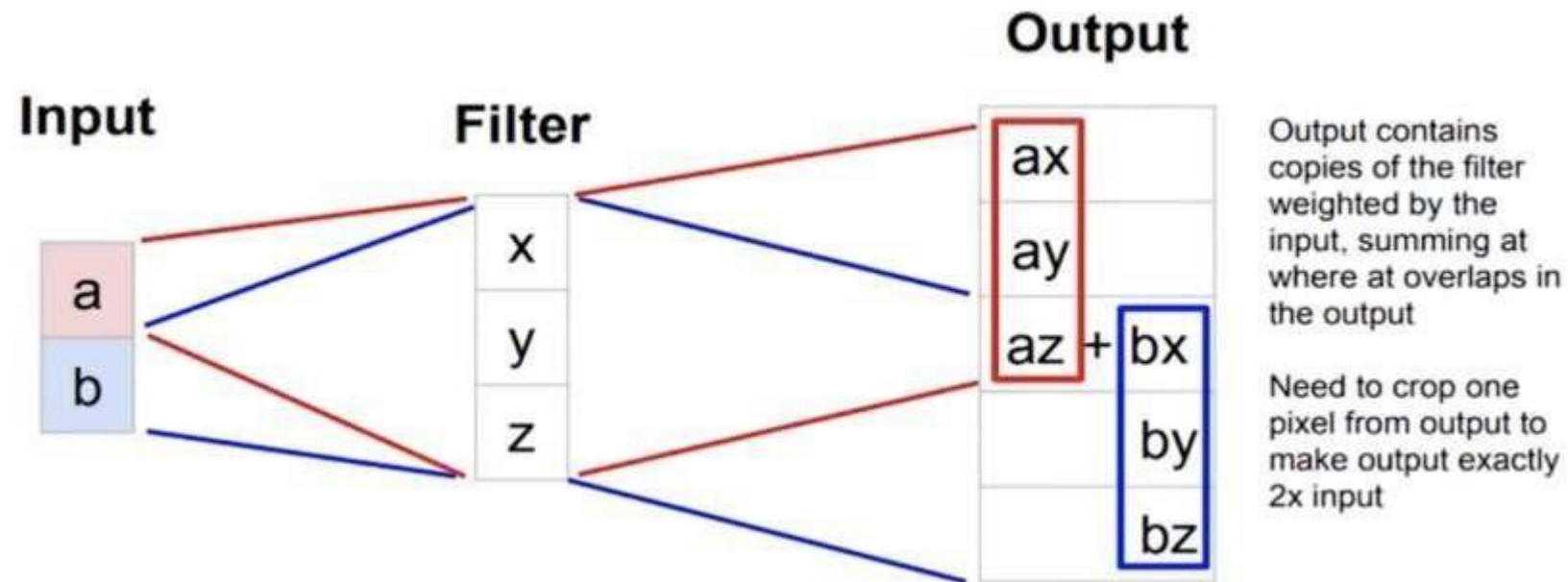
1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

# In network upsampling – max unpooling



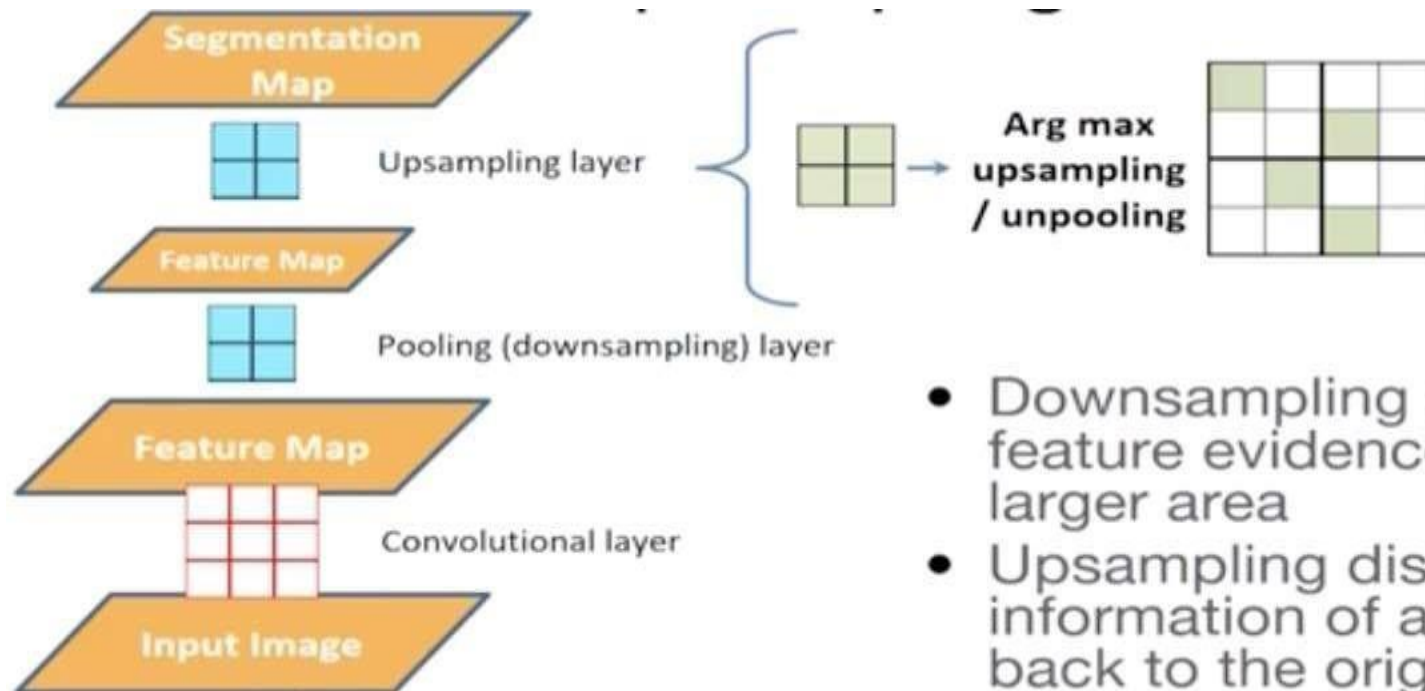


# Learnable upsampling 1d example



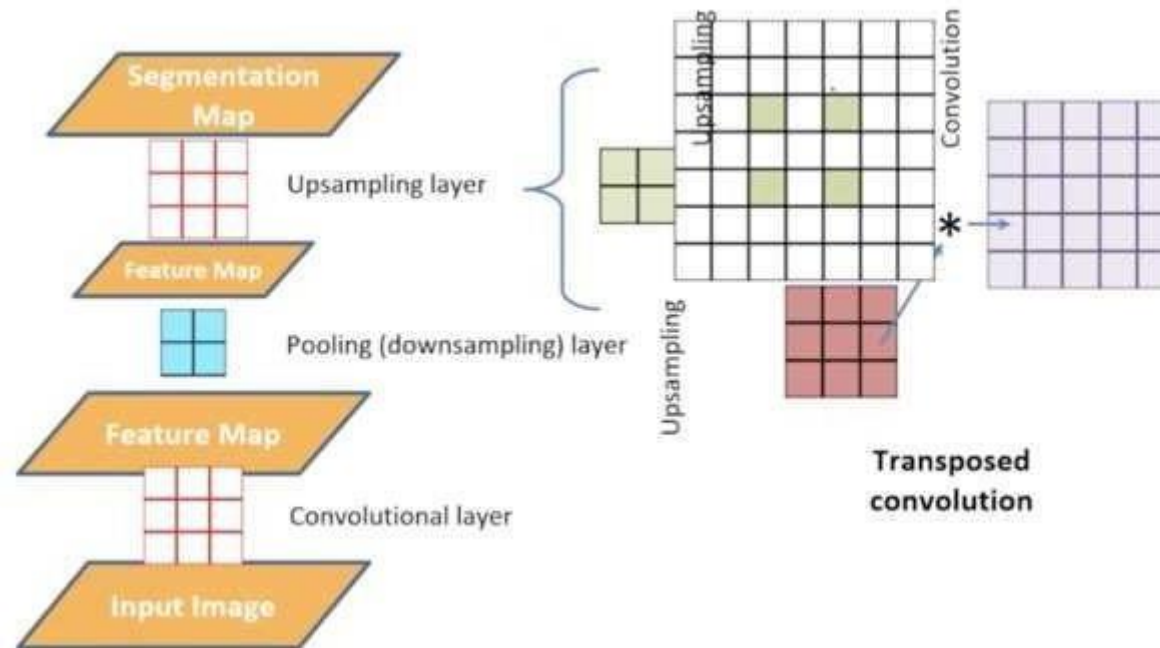
# Segmentation map formation

- To produce a segmentation map downsampling is followed by upsampling

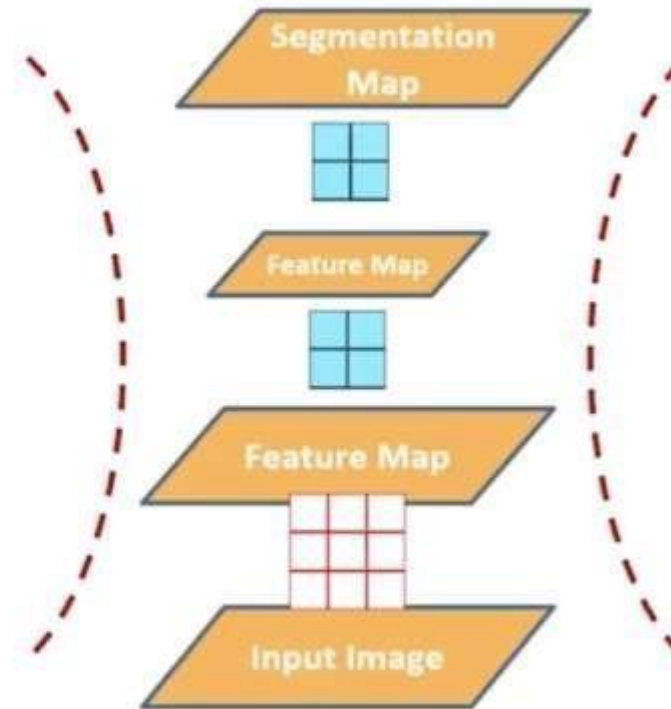


- Downsampling collects feature evidence from a larger area
- Upsampling distributes the information of a segment back to the original pixel domain

# Upsampling can also be learned

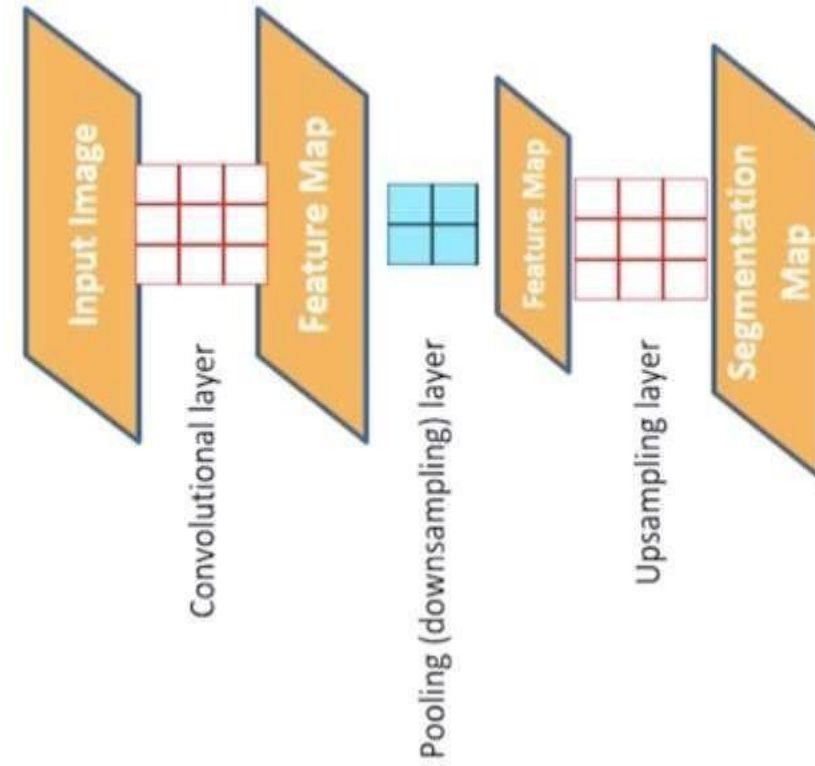


- 
- Downsampling and upsampling leads to an hour-glass structure
  - You will also have skip connections

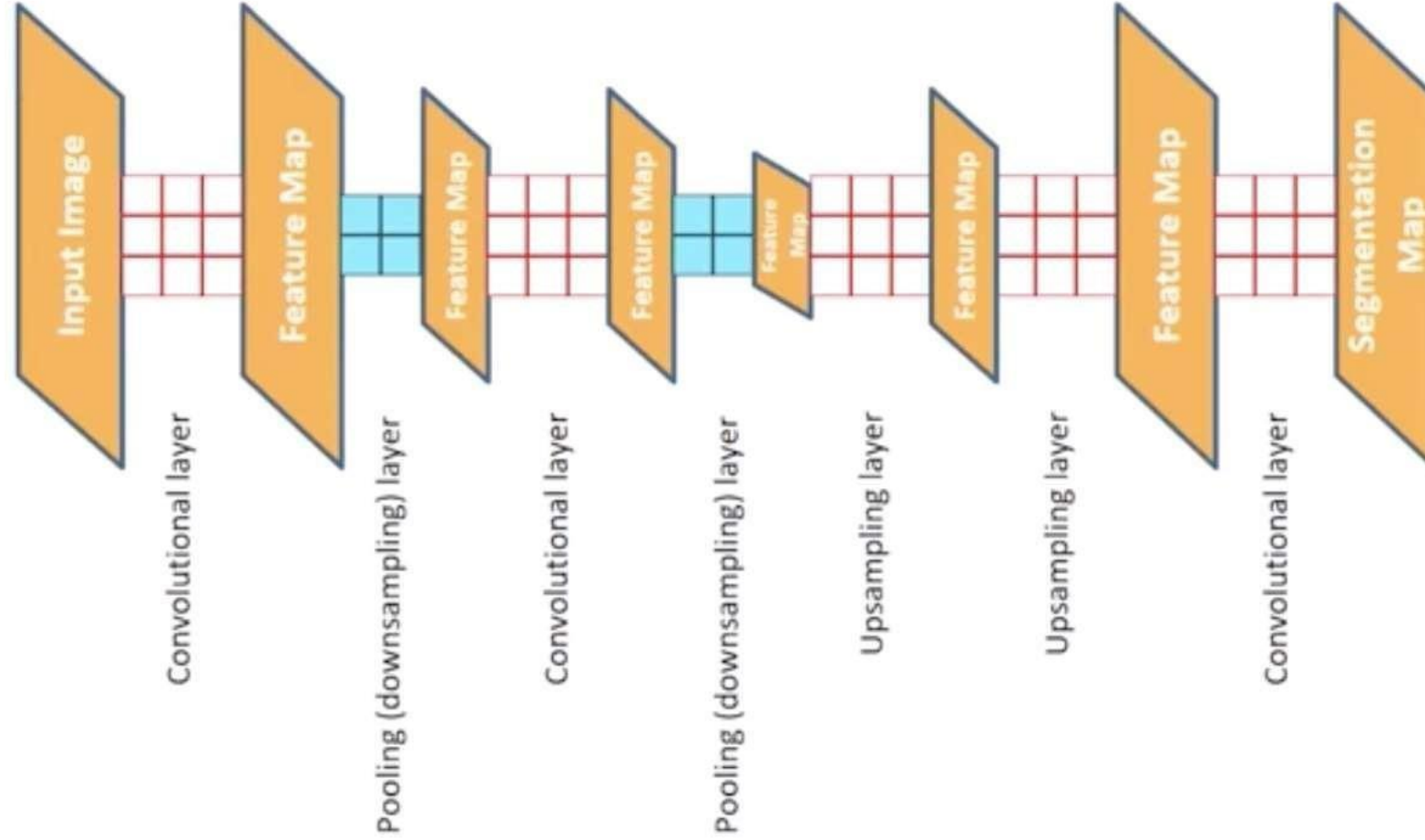


# upsampling

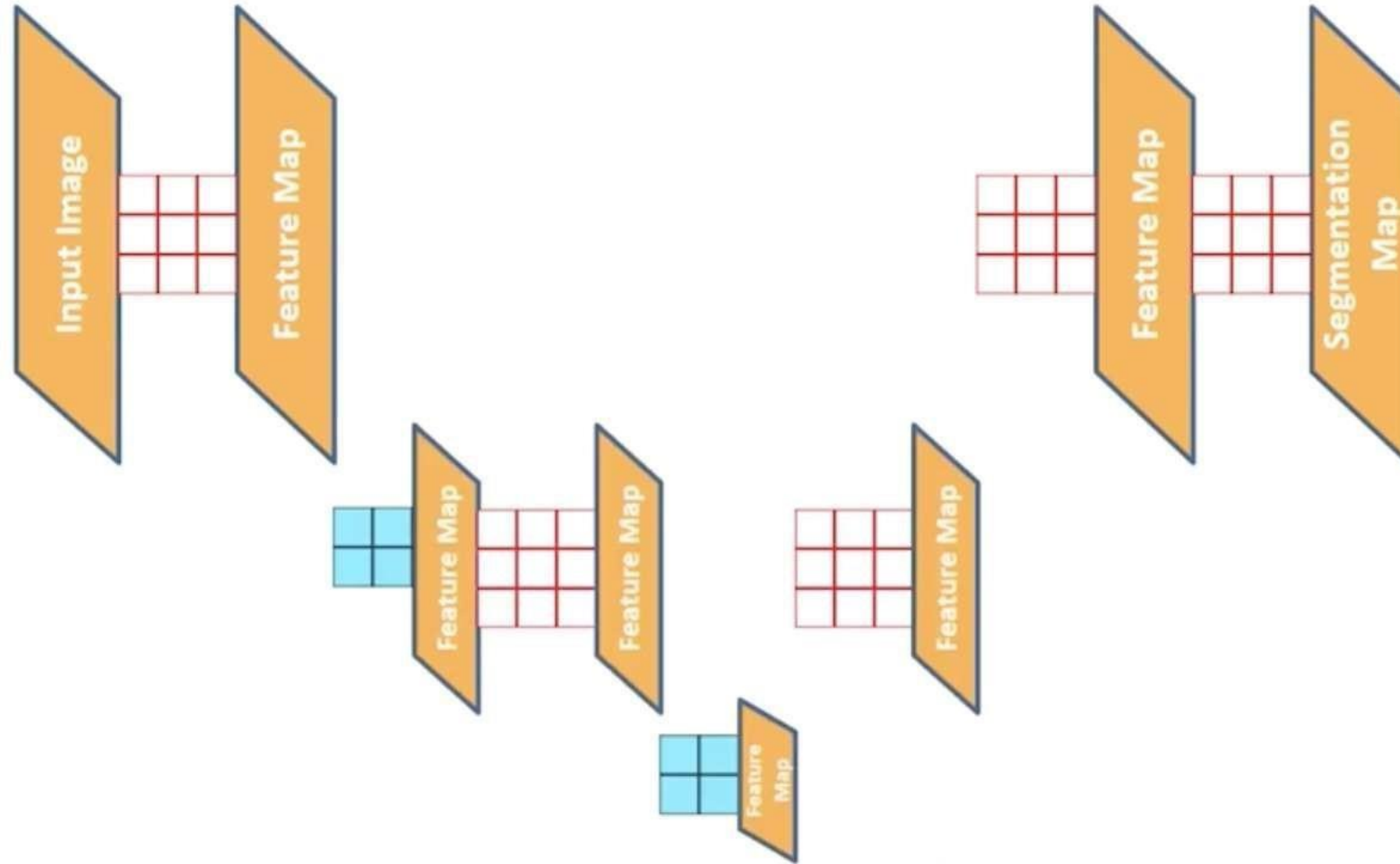
- If we arrange the image horizontally ->



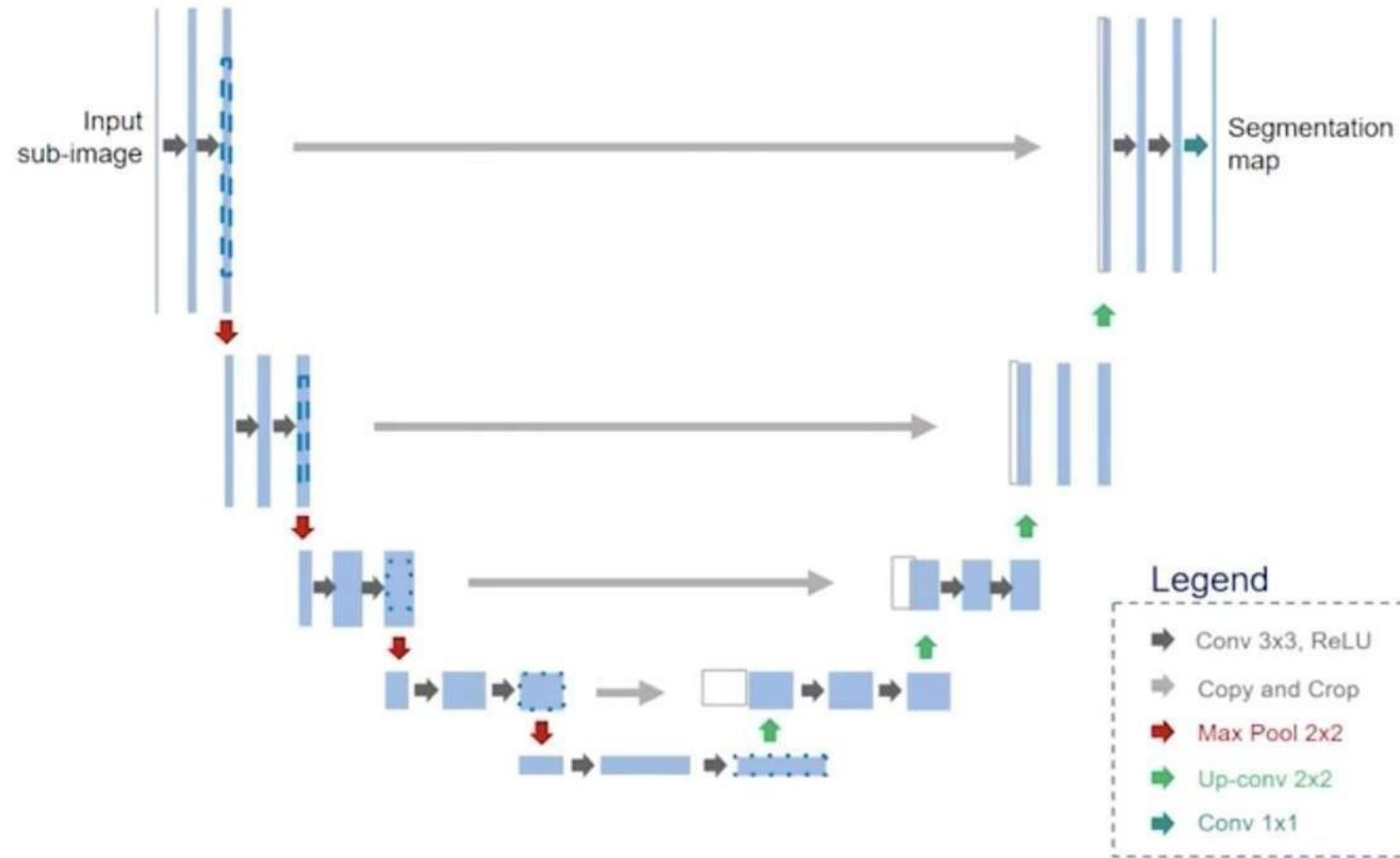
# More layers can be added



# Visually rearrange layers in a big U



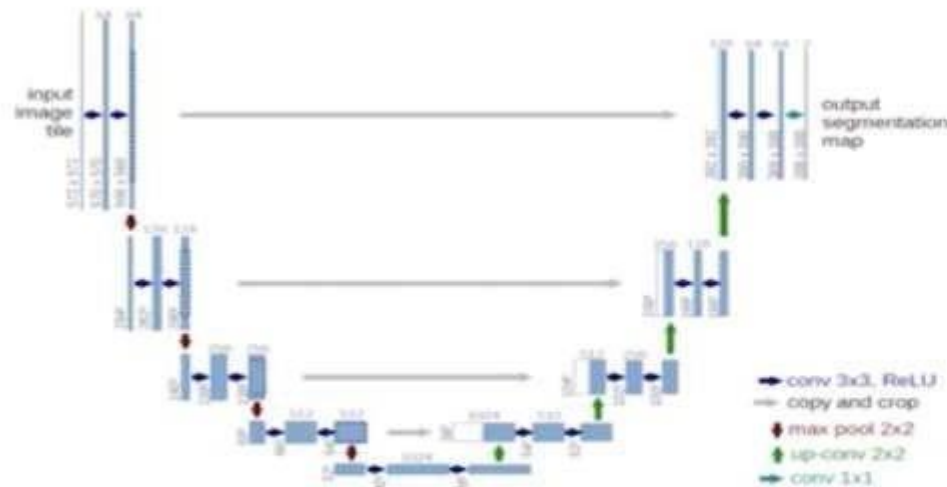
U-Net is similar to the structure described in the previous slide





# UNET

2



**Fig. 1.** U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

The architecture has -

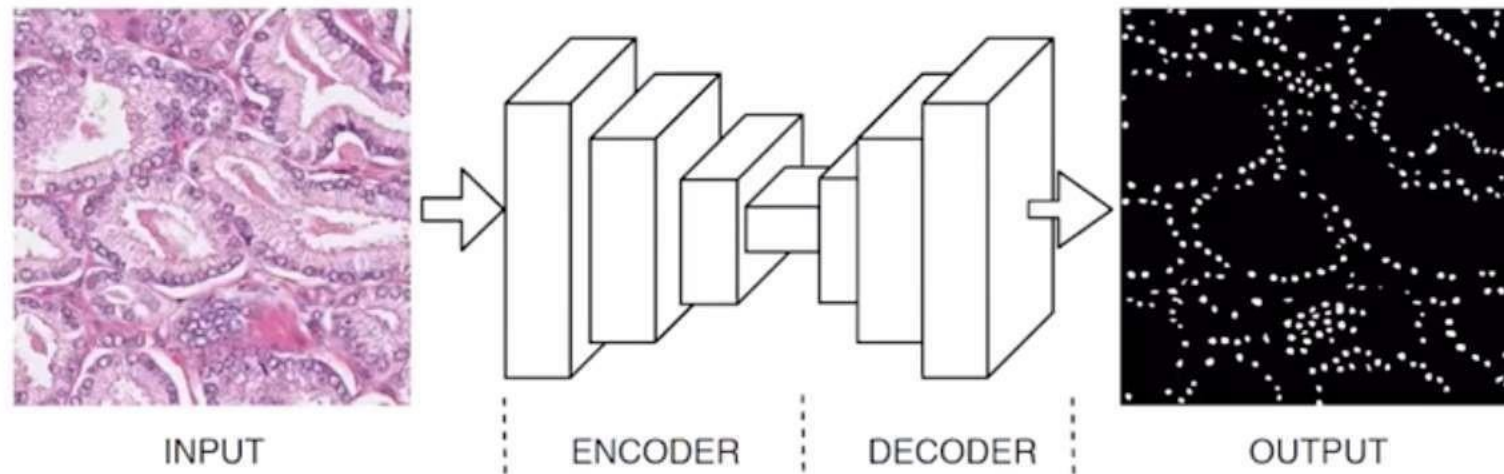
An Encoder - Downsampling part. It is used to get context in the image. It is just a stack of convolutional and max pooling layers.

A Decoder - Symmetric Upsampling part. It is used for precise localization. Transposed convolution is used for upsampling.

It is a fully convolutional network (FCN). It has Convolutional layers and it does not have any dense layer so it can work for image of any size.

# UNet

- Sample output of nucleus segmentation from pathology



# Dice coefficient

- 
- X is predicted set of pixels and Y is the ground truth  $\frac{2 * |X \cap Y|}{|X| + |Y|}$
  - A higher dice coefficient is better. A dice coeff of 1 can be achieved when there is perfect overlap between X and Y. Since the denominator is constant, the only way to maximise the metric is to increase the overlap between X and Y

# Semantic segmentation

---

- Hands on Oxford IIIT pets dataset