

Regular meeting

Laksono Kurnianggoro

laksono@islab.ulsan.ac.kr

22 May 2018

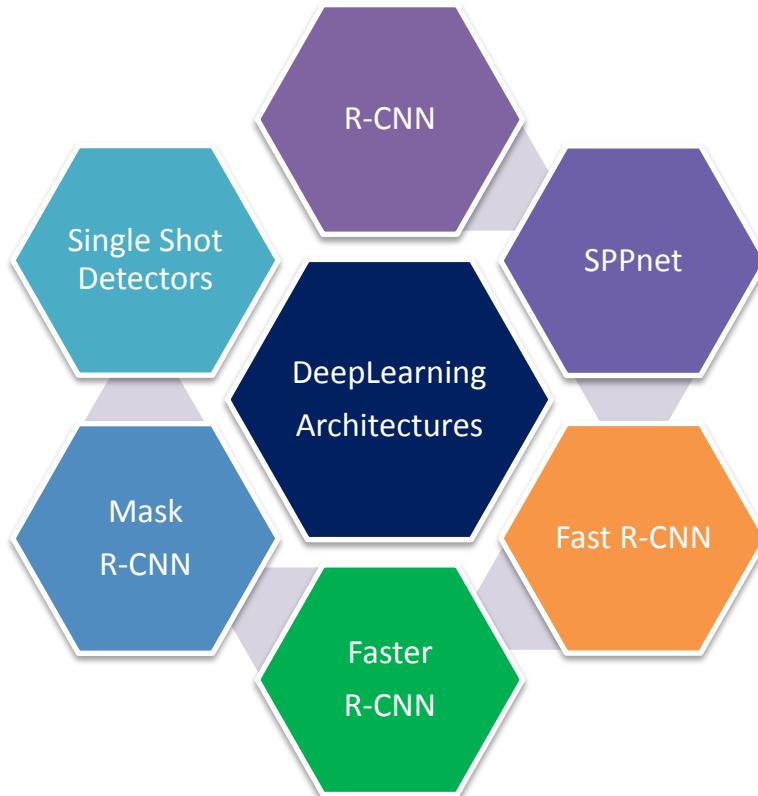




Today's Topics

CO

Intro to Google
colab



Architectures for
object detection



TensorFlow object
detection API



<https://colab.research.google.com>

Introduction Page of the Colab Environment



Hello, Colaboratory

File Edit View Insert Runtime Tools Help

CODE TEXT CELL CELL COPY TO DRIVE CONNECT EDITING

Table of contents Code snippets X

Welcome to Colaboratory!

Local runtime support

Python 3

TensorFlow execution

Visualization

Forms

Examples

For more information:

+ SECTION

Welcome to Colaboratory!

Colaboratory is a Google research project created to help disseminate machine learning education and research. It's a Jupyter notebook environment that requires no setup to use and runs entirely in the cloud.

Colaboratory notebooks are stored in [Google Drive](#) and can be shared just as you would with Google Docs or Sheets. Colaboratory is free to use.

For more information, see our [FAQ](#).

Local runtime support

Colab also supports connecting to a Jupyter runtime on your local machine. For more information, see our [documentation](#).

▼ Python 3

Colaboratory supports both Python2 and Python3 for code execution.

- When creating a new notebook, you'll have the choice between Python 2 and Python 3.
- You can also change the language associated with a notebook; this information will be written into the .ipynb file itself, and thus will be preserved for future sessions.

```
[ ] import sys  
print('Hello, Colaboratory from Python {}!'.format(sys.version_info[0]))
```

Hello, Colaboratory from Python 3!

What You Get from Colab?



```
[ ] !lscpu
```

```
↳ Architecture:          x86_64
CPU op-mode(s):         32-bit, 64-bit
Byte Order:              Little Endian
CPU(s):                  2
On-line CPU(s) list:   0,1
Thread(s) per core:    2
Core(s) per socket:    1
Socket(s):              1
NUMA node(s):           1
Vendor ID:              GenuineIntel
CPU family:              6
Model:                  63
Model name:             Intel(R) Xeon(R) CPU @ 2.30GHz
Stepping:                0
CPU MHz:                 2300.000
```

```
[2] !free -m
```

```
↳
      total        used        free       shared  buff/cache   available
Mem:     13029        420      2258        247      10350      12113
Swap:        0          0          0
```

What is the Operating System?



```
!cat /etc/*release
```

```
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=17.10
DISTRIB_CODENAME=artful
DISTRIB_DESCRIPTION="Ubuntu 17.10"
NAME="Ubuntu"
VERSION="17.10 (Artful Aardvark)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 17.10"
VERSION_ID="17.10"
```

You can install any programs using **apt-get** or python modules using **pip install** but they will be deleted in each new session!



Try to Use GPU

The image shows a Jupyter Notebook interface. On the left, there's a sidebar with a 'CODE' tab selected. The main area has a play button icon. A context menu is open under the 'Runtime' tab in the top navigation bar. The menu items are:

- Run all (Ctrl+F9)
- Run before (Ctrl+F8)
- Run focused (Ctrl+Enter)
- Run selection (Ctrl+Shift+Enter)
- Run after (Ctrl+F10)
- Interrupt execution (Ctrl+M I)
- Restart runtime... (Ctrl+M .)
- Change runtime type

To the right, a modal dialog box titled 'Notebook settings' is displayed. It contains the following fields:

- Runtime type: Python 2
- Hardware accelerator: GPU (with a question mark icon)
- A checkbox labeled 'Omit code cell output when saving this notebook'.

At the bottom of the dialog are 'CANCEL' and 'SAVE' buttons.

Depends on your luck, you may get access to a **GPU**!



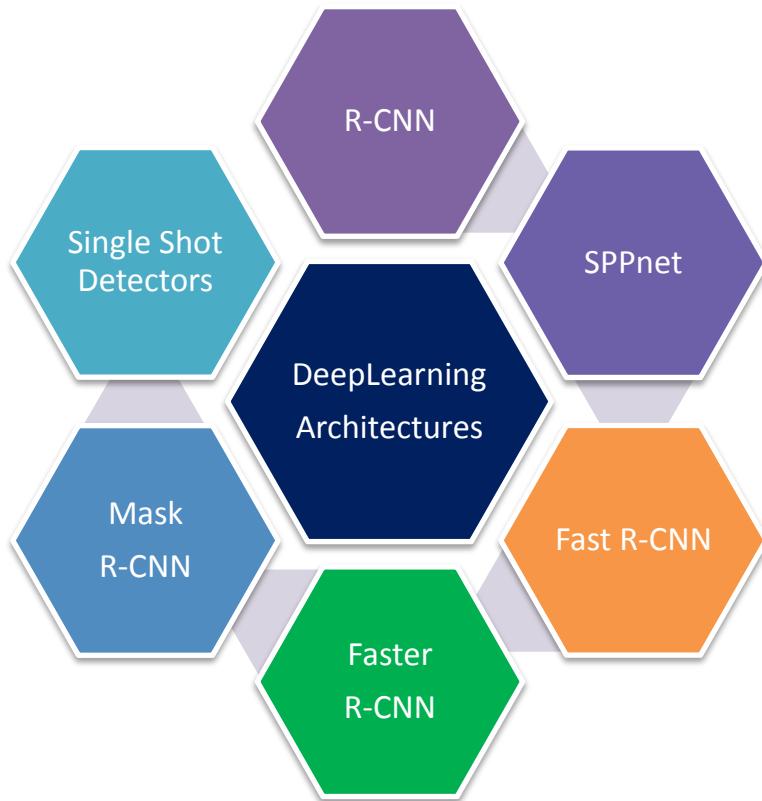
Verify the GPU Availability

```
[6] !/opt/bin/nvidia-smi
```

```
↪ Mon May 21 02:32:10 2018
```

NVIDIA-SMI 384.111				Driver Version: 384.111			
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr.	ECC
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute M.	
0	Tesla K80	Off	00000000:00:04.0	Off	0		
N/A	29C	P8	26W / 149W	0MiB / 11439MiB	0%	Default	

Processes:				GPU Memory
GPU	PID	Type	Process name	Usage
No running processes found				



The Classical Object Detector

- ▶ Sliding window + handcrafted features + classifier

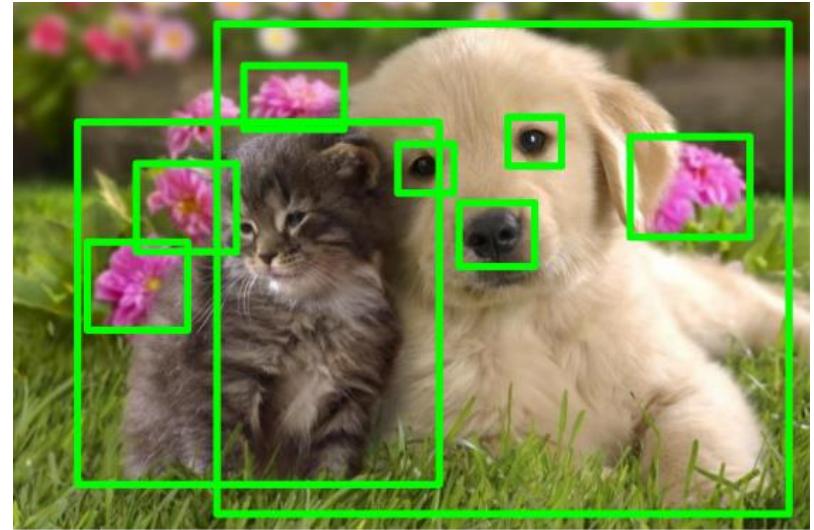


- ▶ Problems:
 - ▶ Dense prediction (location and scale) → leads to slow running time
 - ▶ Crafting a robust feature is difficult (also object specific)

Region Proposal for Search Space Reduction



- ▶ Find regions on image that likely to contain objects
 - ▶ Class “agnostic”



Images: adapted from Fei-fei Li's presentation

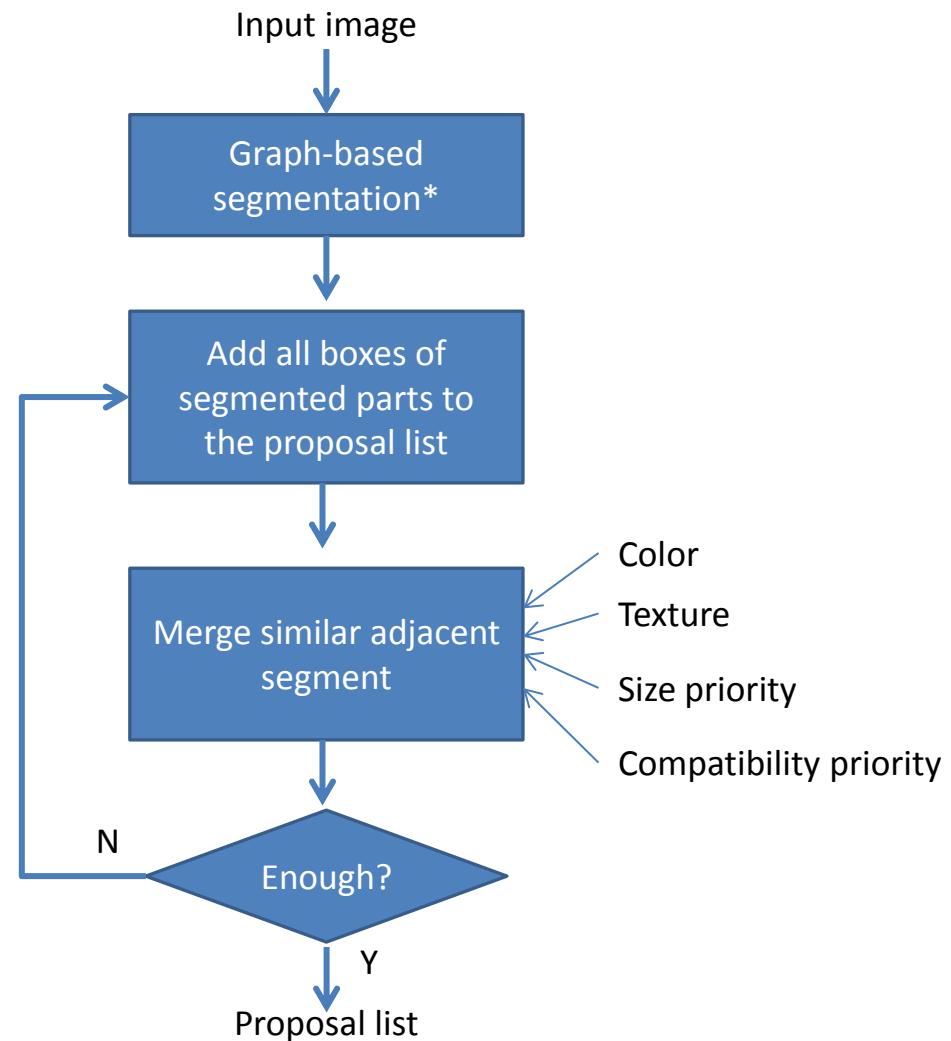
Region Proposal: Selective Search



Input image



Over-segmented output*

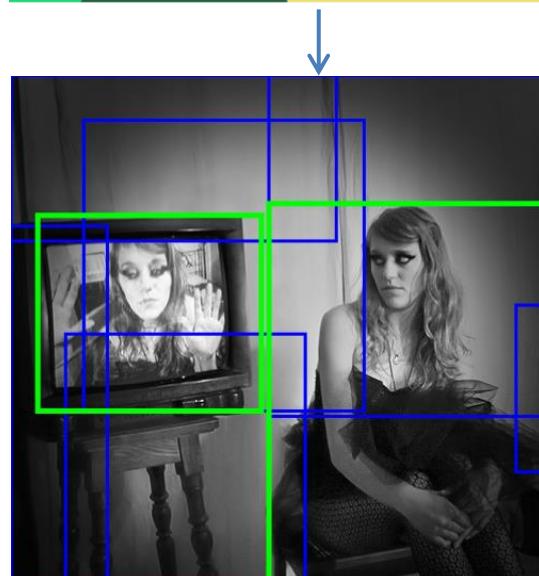
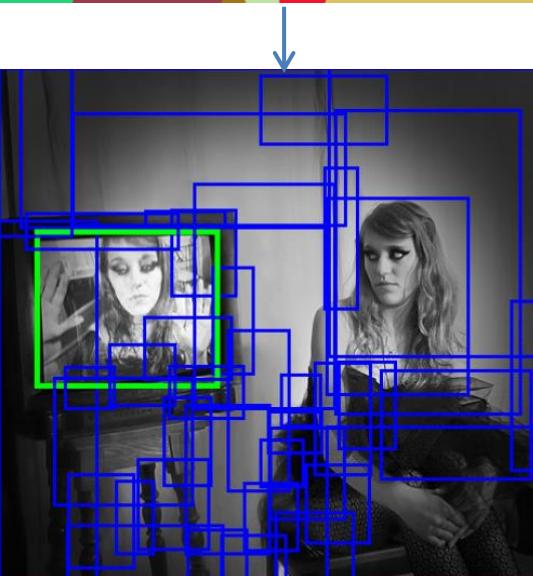
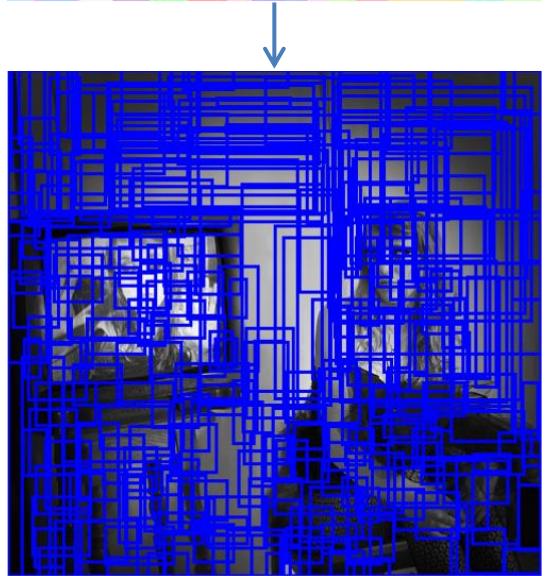
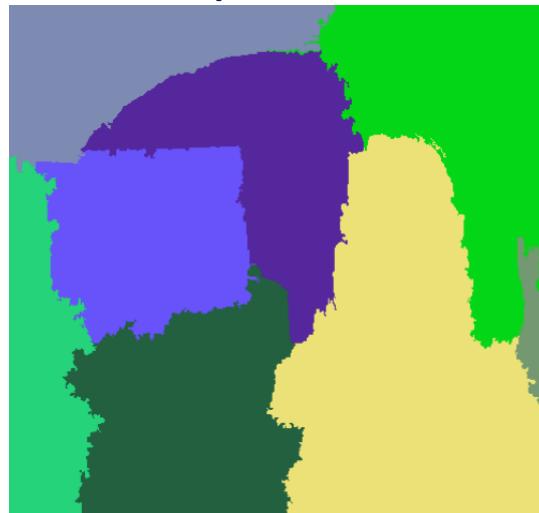
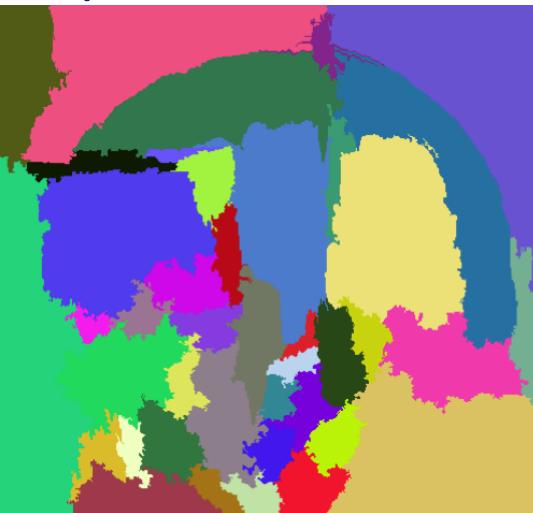


* P. Felzenszwalb ,et.al. , Efficient Graph-based Image Segmentation, IJCV, 2004. Code: <http://cs.brown.edu/people/pfelzens/segment/>
Images: <https://www.learnopencv.com/selective-search-for-object-detection-cpp-python/>



Region Proposal: Selective Search

- ▶ Segmentation based on pixel features, merging at multiple scales



Uijlings et al., "Selective Search for Object Recognition", IJCV 2013. Code: <https://www.koen.me/research/selectivesearch/>



Region Proposal: Many Other Choices

Method	Approach	Outputs Segments	Outputs Score	Control #proposals	Time (sec.)	Repeatability	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	.	.	.

Hosang et al, "What makes for effective detection proposals?", PAMI 2015



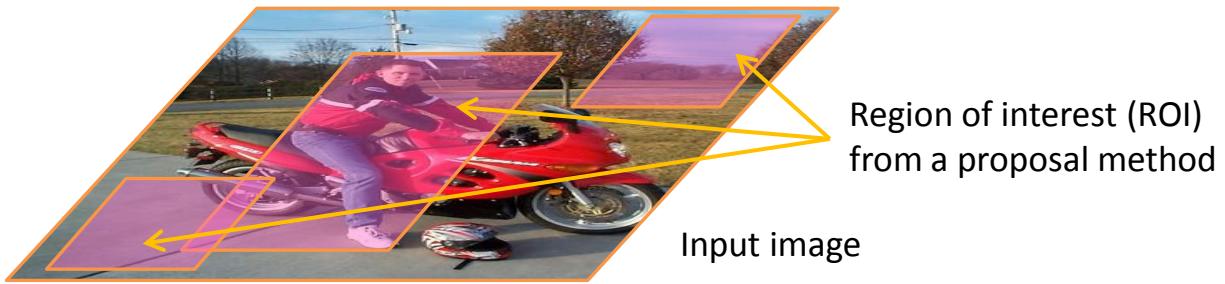
The R-CNN Method



Input image

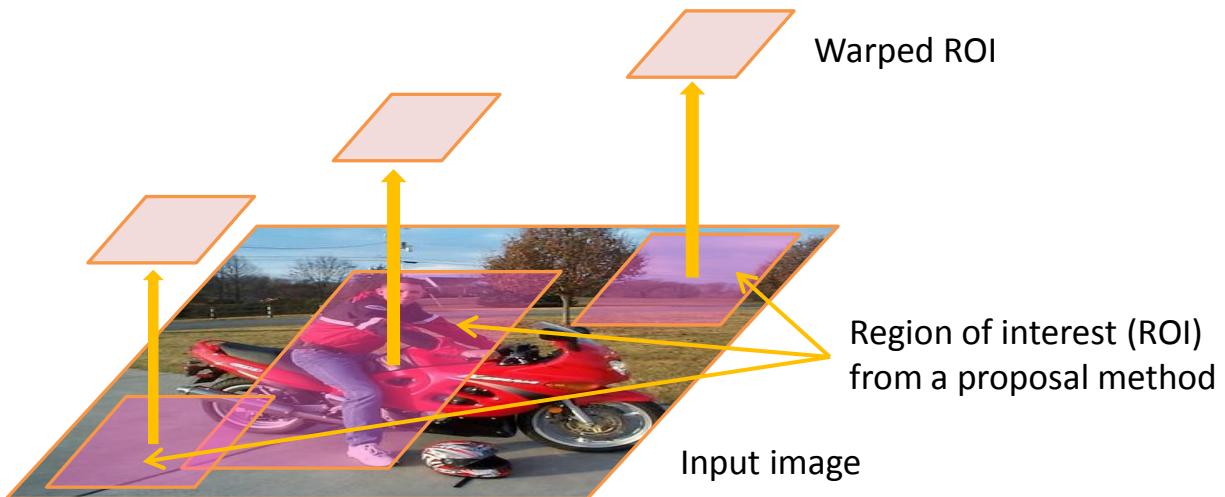


The R-CNN Method

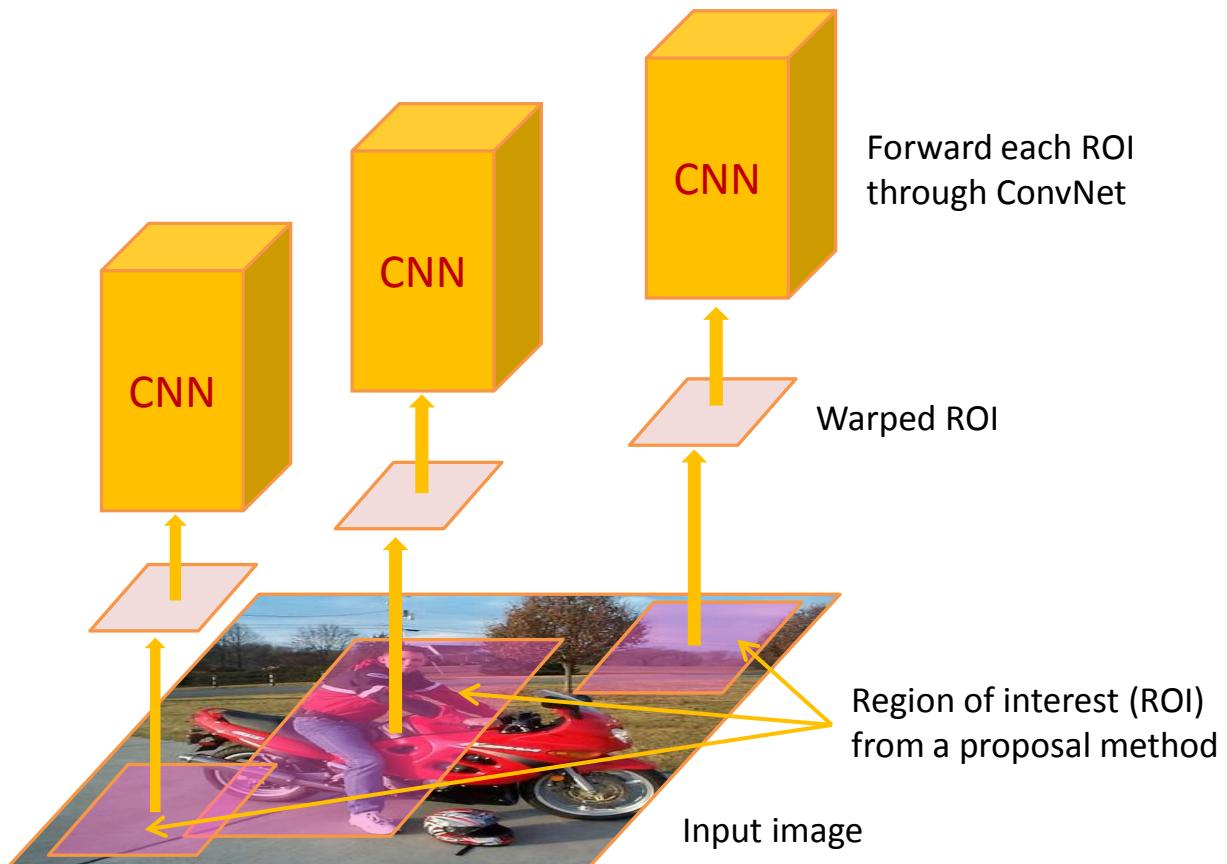




The R-CNN Method



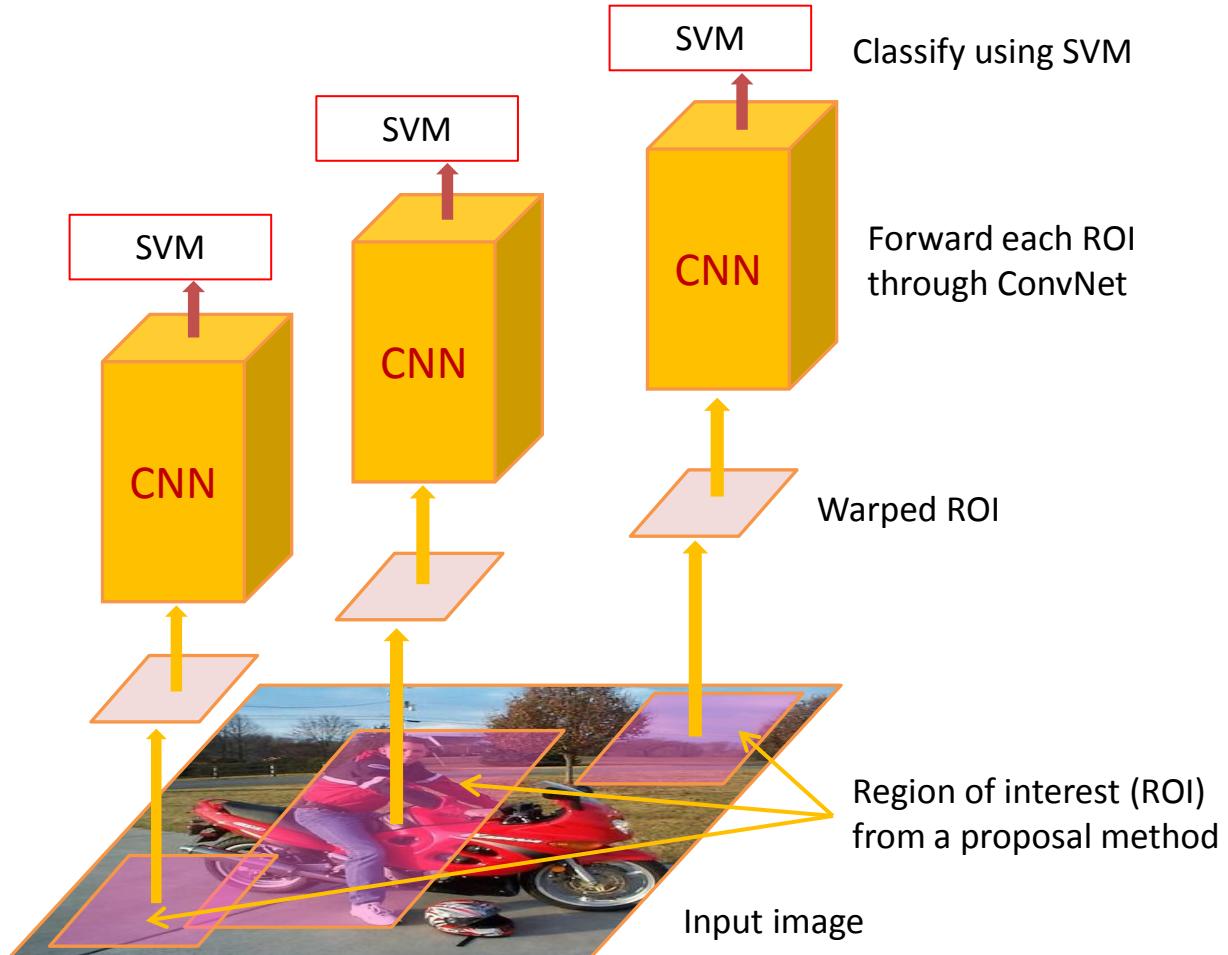
The R-CNN Method





The R-CNN Method

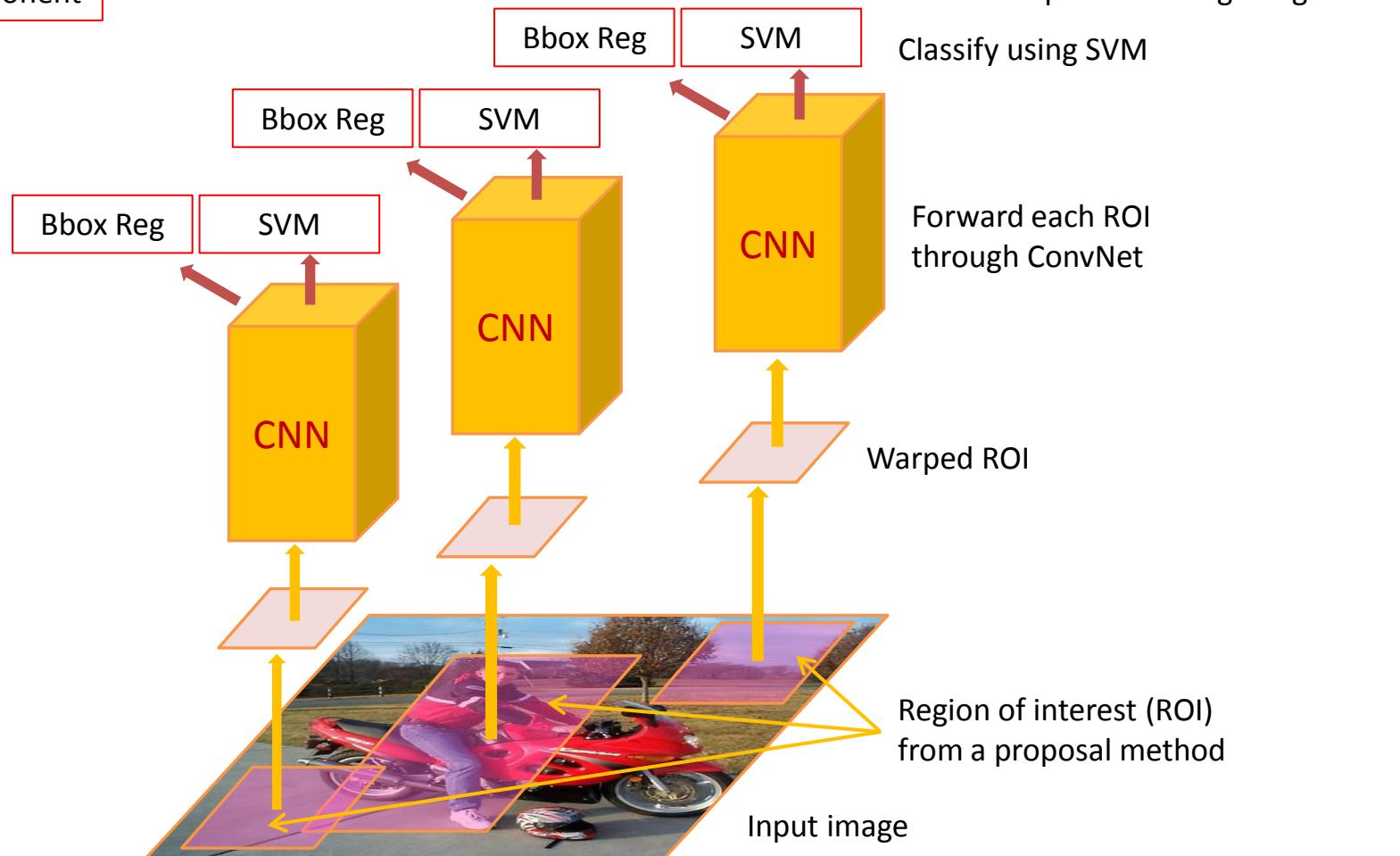
Post hoc component





The R-CNN Method

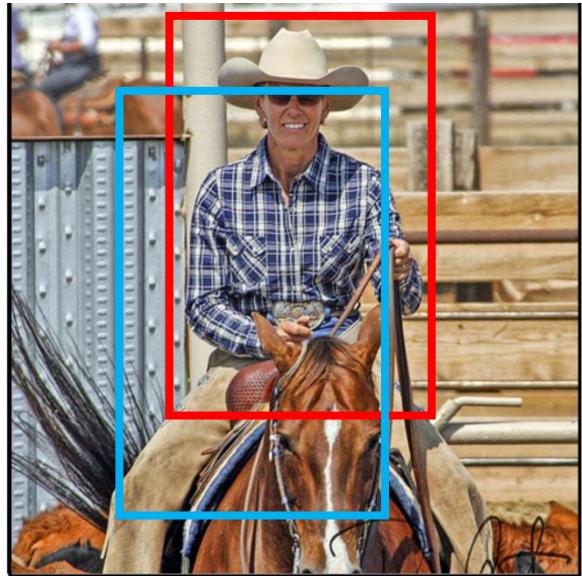
Post hoc component



Bounding Box Regression (1/2)

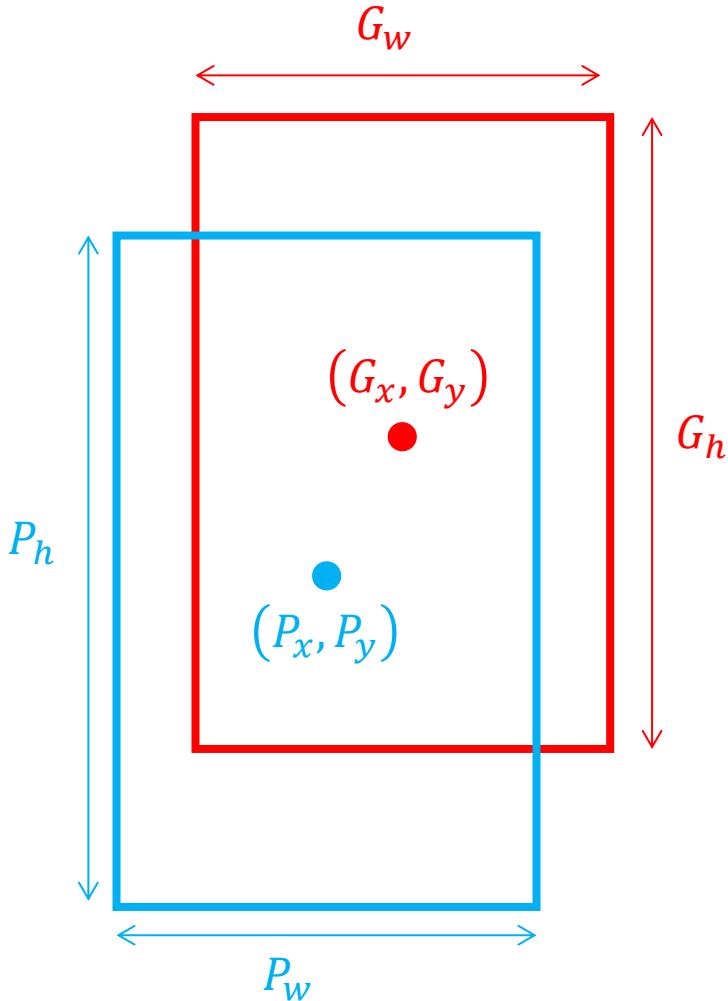
► Goal:

- refines the position and size of the detection result.



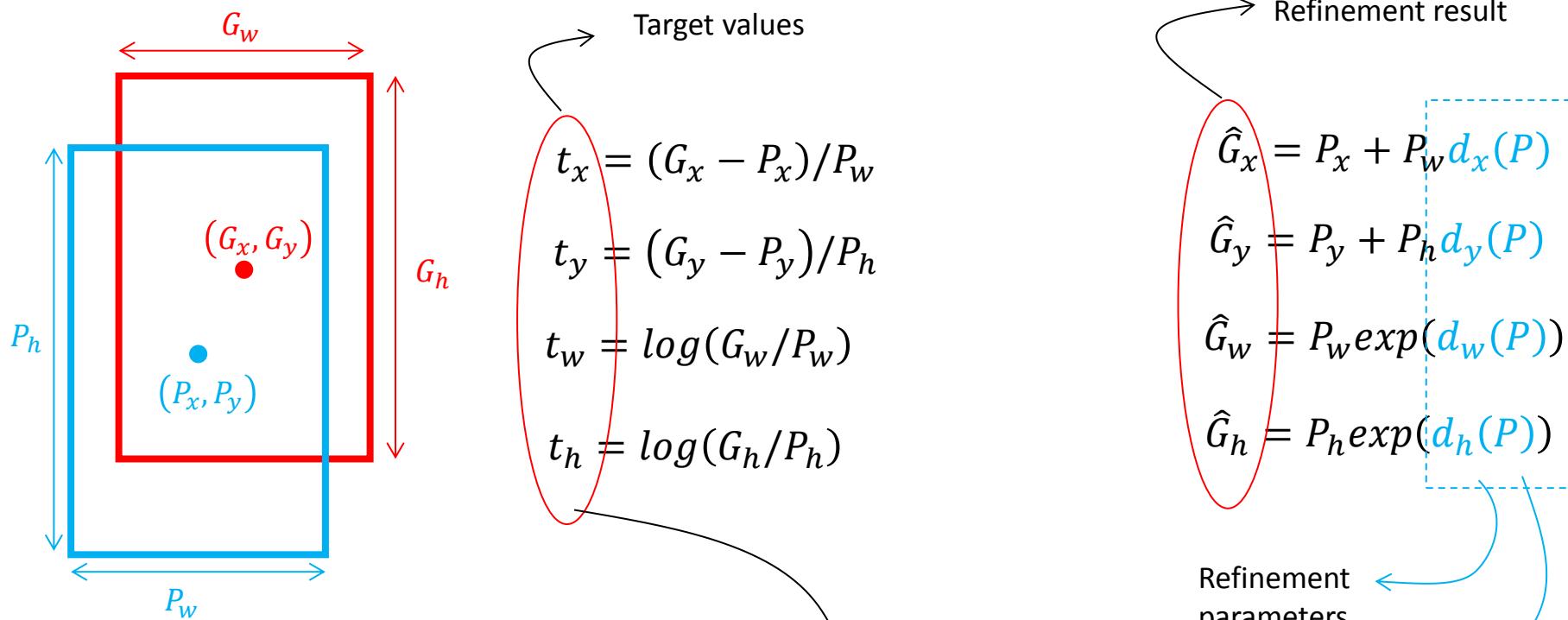
— Red — Ground-truth

— Blue — Prediction



Bounding Box Regression (2/2)

► Solution: Predicting offset using the feature.



Problem formulation:

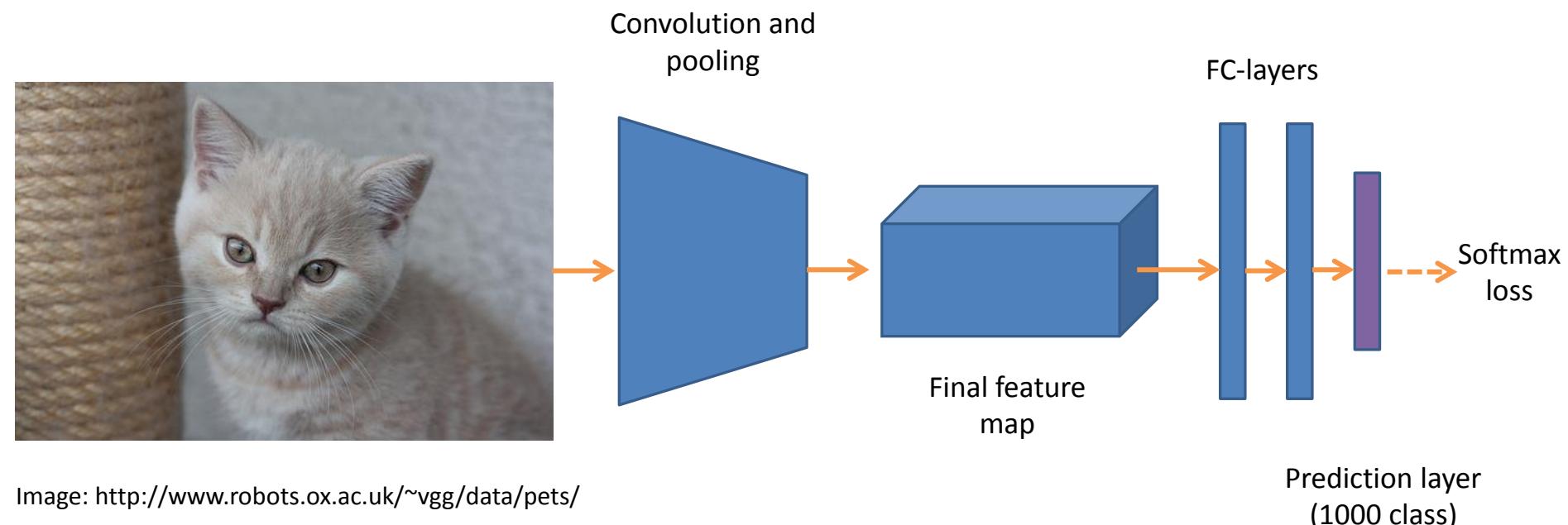
Where: $d_\star(P) = w_\star^T \phi_5(P)$

Feature from pool₅

$$\min_{w_\star} \sum_i^N (t_\star - w_\star^T \phi_5(P))^2 + \lambda \|w_\star\|^2$$

R-CNN Training

- ▶ Step 1:
 - ▶ Train or download a classification model for ImageNet



R-CNN Training

► Step 2: Fine tune model for detection

- Instead of 1000 classes, we want 20 object classes (PASCAL) + background
- Change the prediction layer for 21 classes and train it

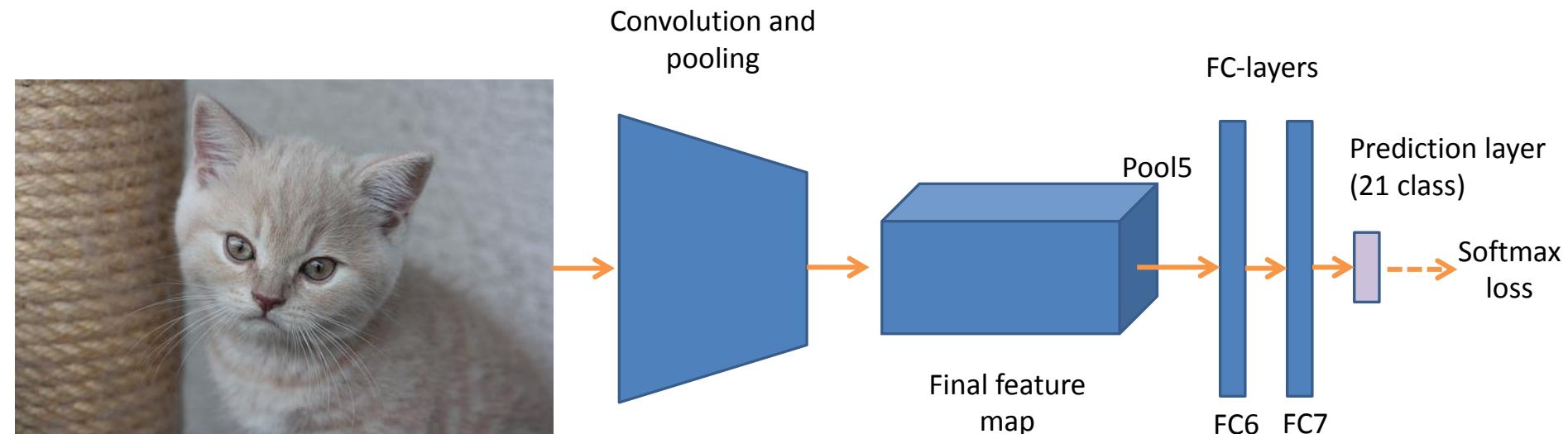


Image: <http://www.robots.ox.ac.uk/~vgg/data/pets/>

R-CNN Training

► Step 3: Feature Extraction

- Extract region proposal for all training images
- Warp each ROI to CNN input size, run forward through CNN
- Save the feature (e.g. Pool5 or FC6) to disk

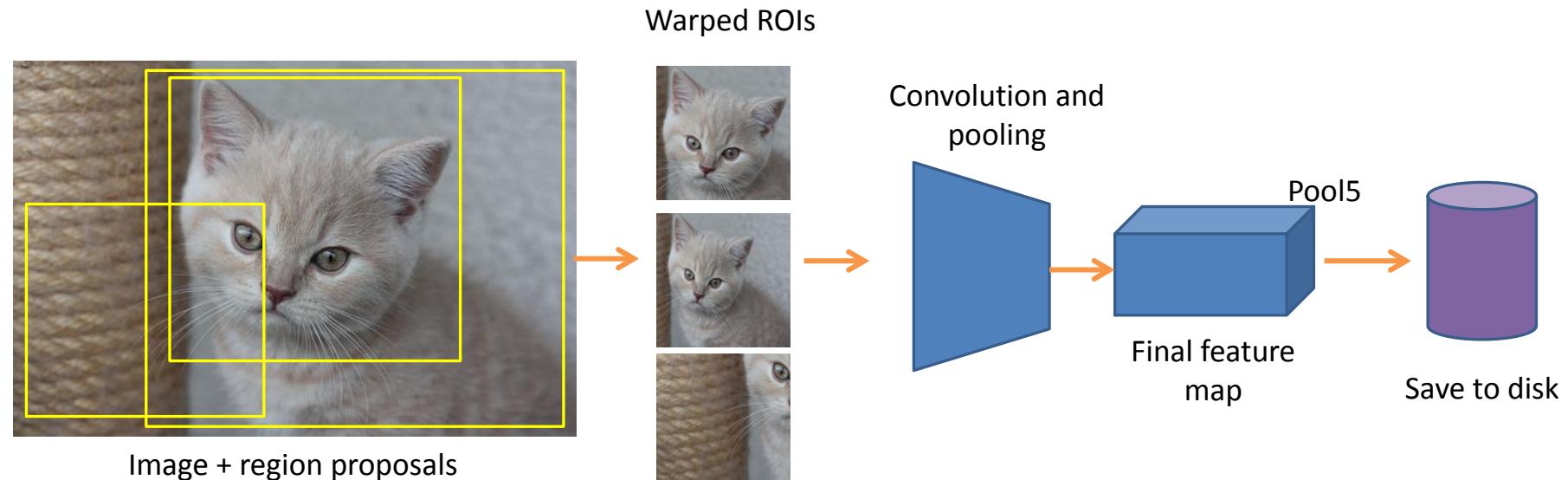
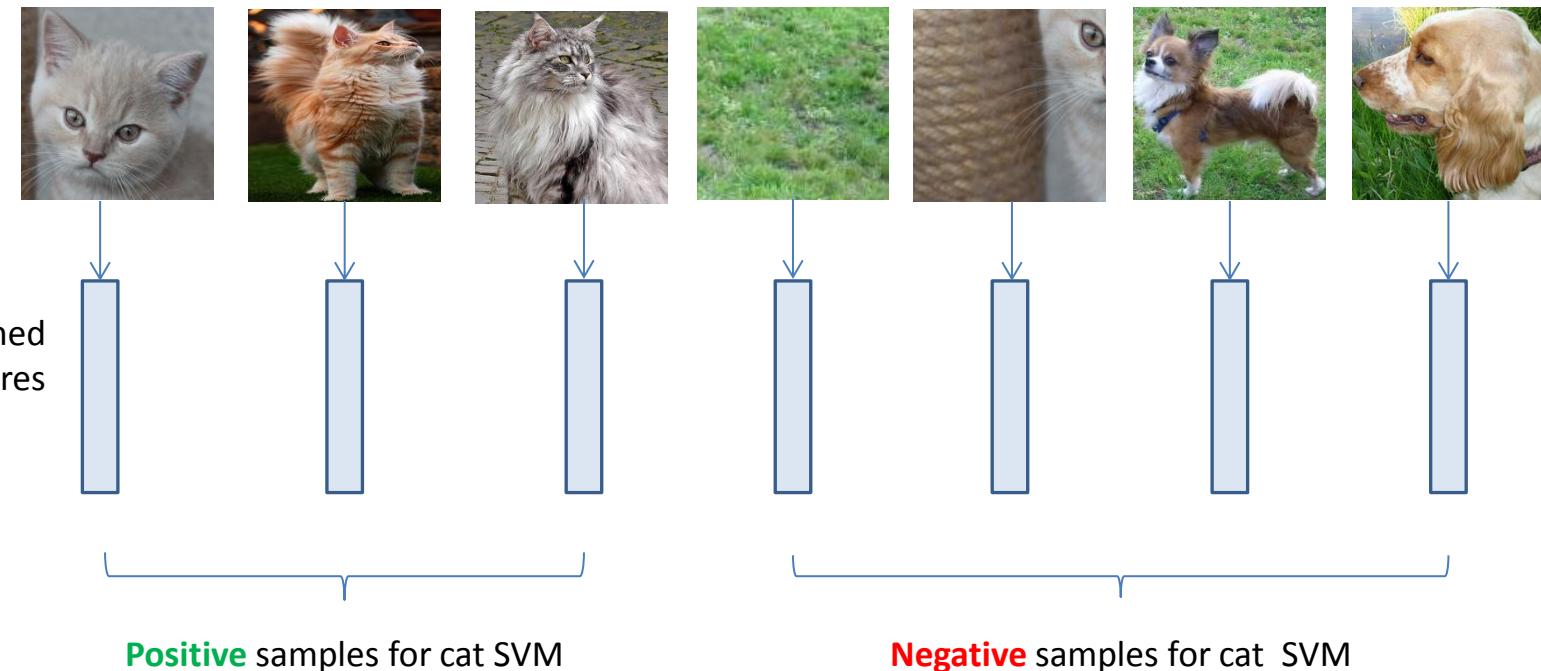


Image: <http://www.robots.ox.ac.uk/~vgg/data/pets/>

R-CNN Training

► Step 4:

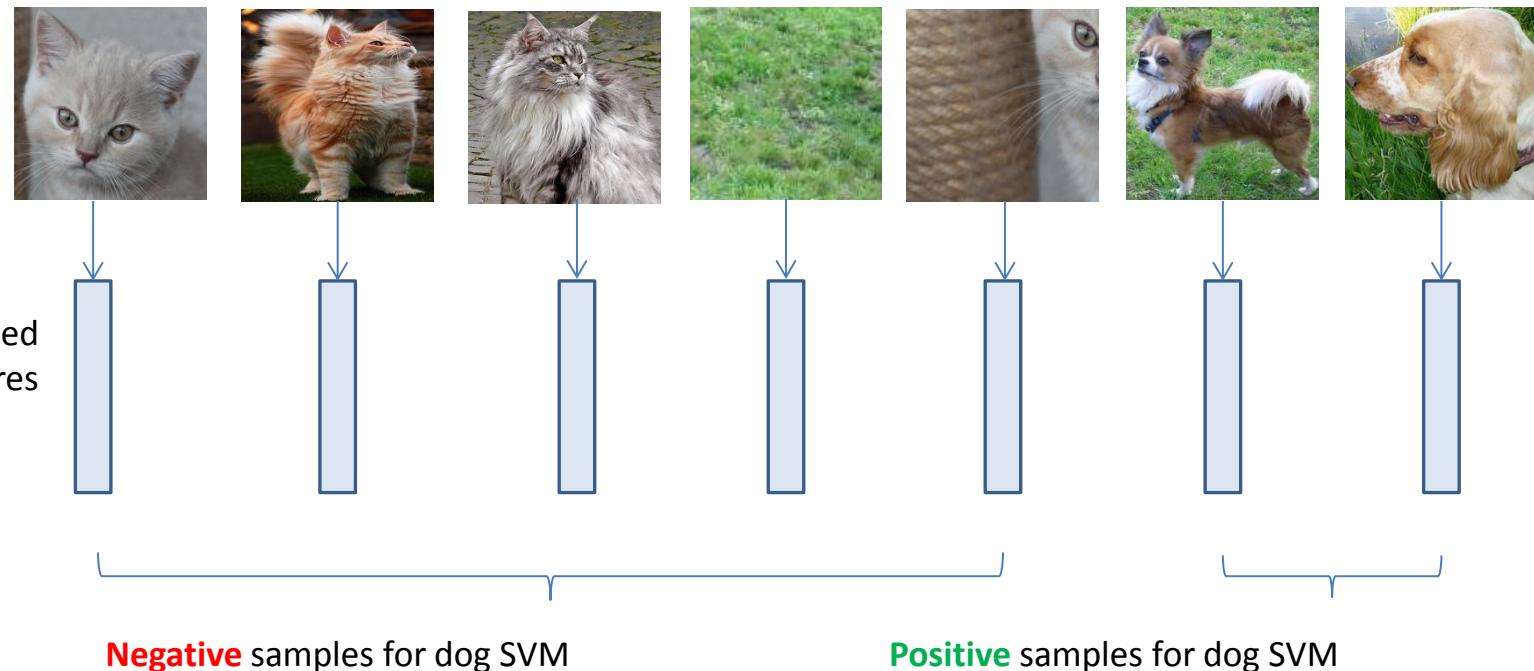
- Train 1 binary SVM per class for to classify region features



R-CNN Training

► Step 4:

- Train 1 binary SVM per class for to classify region features





R-CNN Training

► Step 5: Bounding box regression

- For each class, train a linear regression model
- Maps from cached feature offsets to GT boxes to refines the proposal

Training image regions



Cached region features



Regression targets
(dx , dy , dw , dh)
Normalized coordinates

(0, 0, 0, 0)
Proposal is good

(.25, 0, 0, 0)
Proposal too far to left

(0, 0, -0.125, 0)
Proposal too wide

Image from Fei-fei Li et. al.



Whats wrong with slow R-CNN?

- ▶ Ad-hoc training objectives
 - ▶ Fine-tune network with softmax classifier
 - ▶ Train post-hoc linear SVM
 - ▶ Train post-hoc bonding-box regression
- ▶ Training is slow, takes a lot of disk space (for feature chaching)
- ▶ Inference (detection) is slow
 - ▶ 47s/image with VGG16
 - ▶ Fixed by SPP net

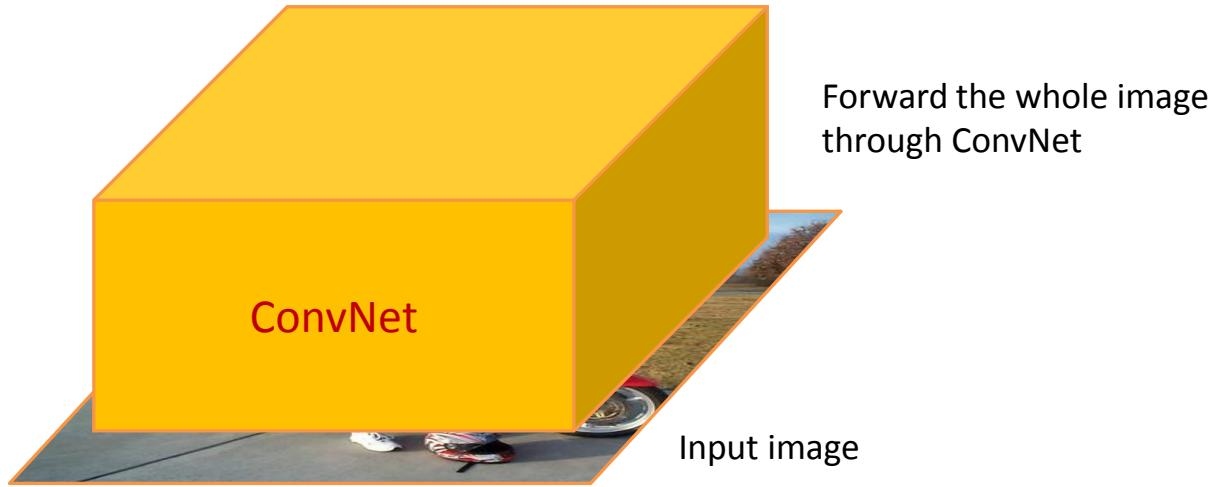
SPP net: K. He, et.al. Spatial pyramid pooling in deep convolutional networks for visual recognition, ECCV 2014.

The Spatial Pyramid Pooling Network

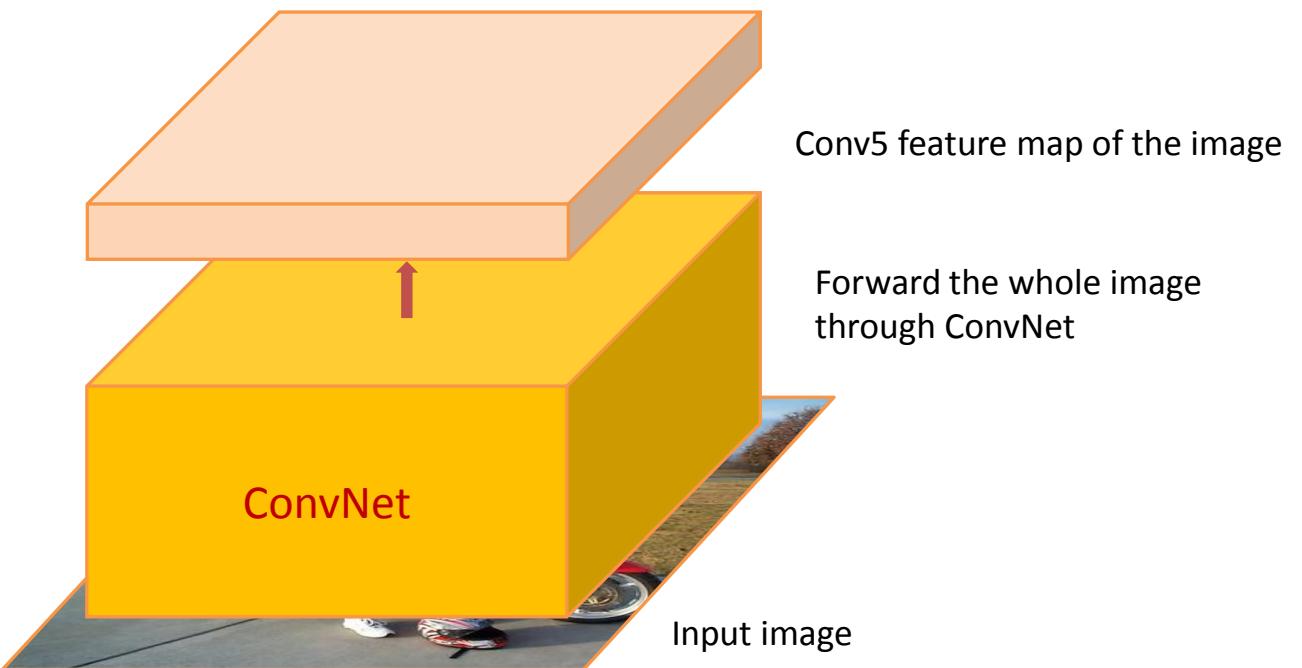


Input image

The Spatial Pyramid Pooling Network

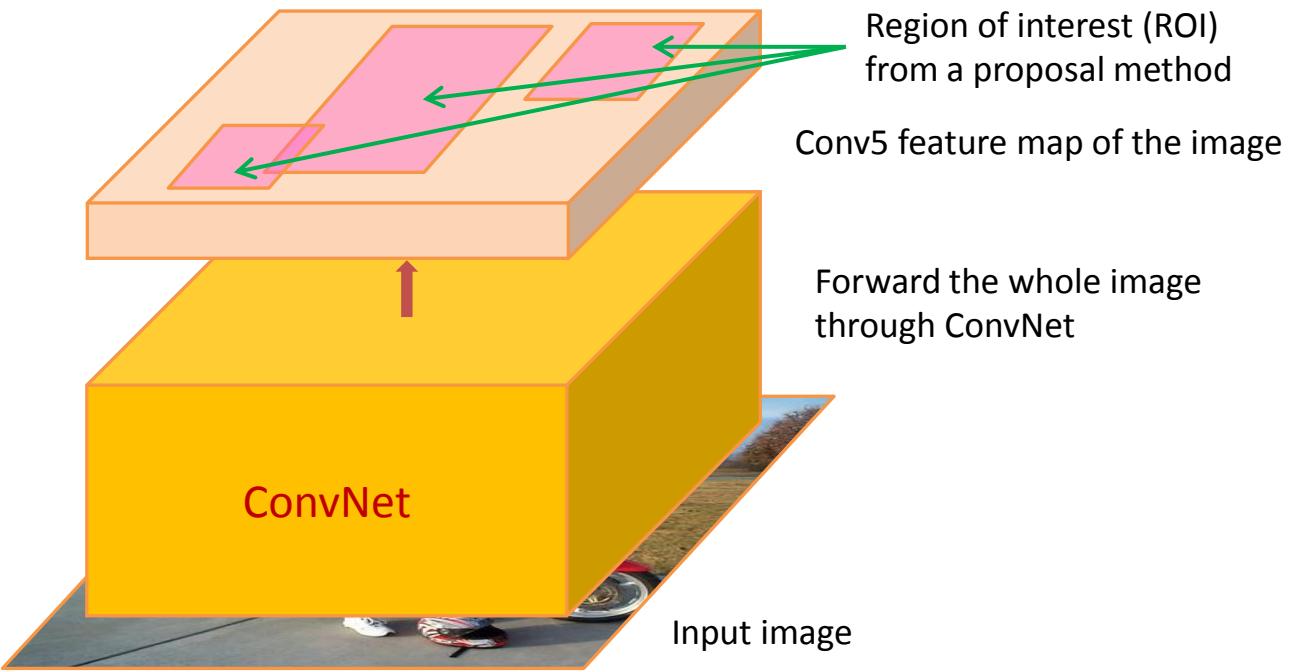


The Spatial Pyramid Pooling Network

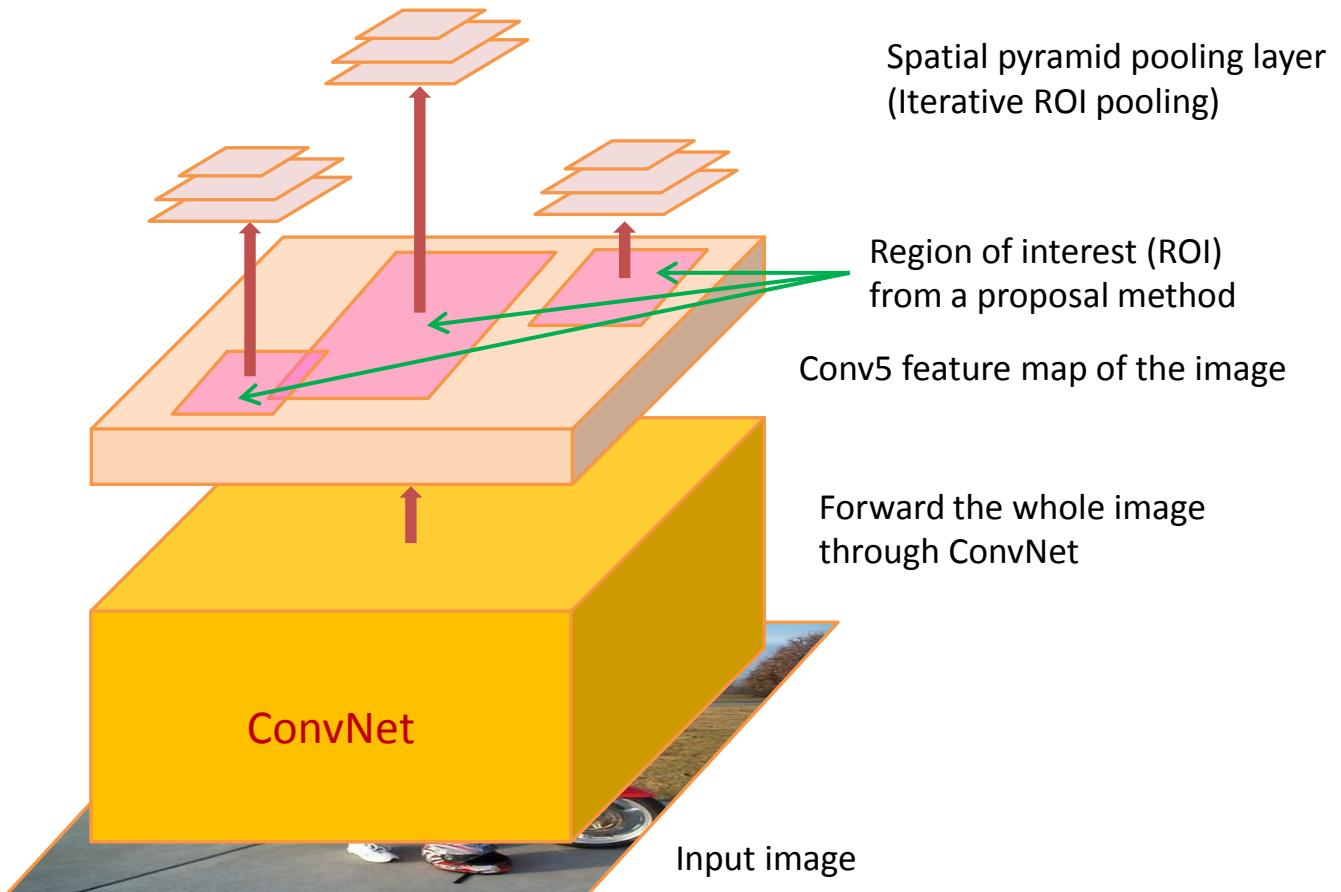




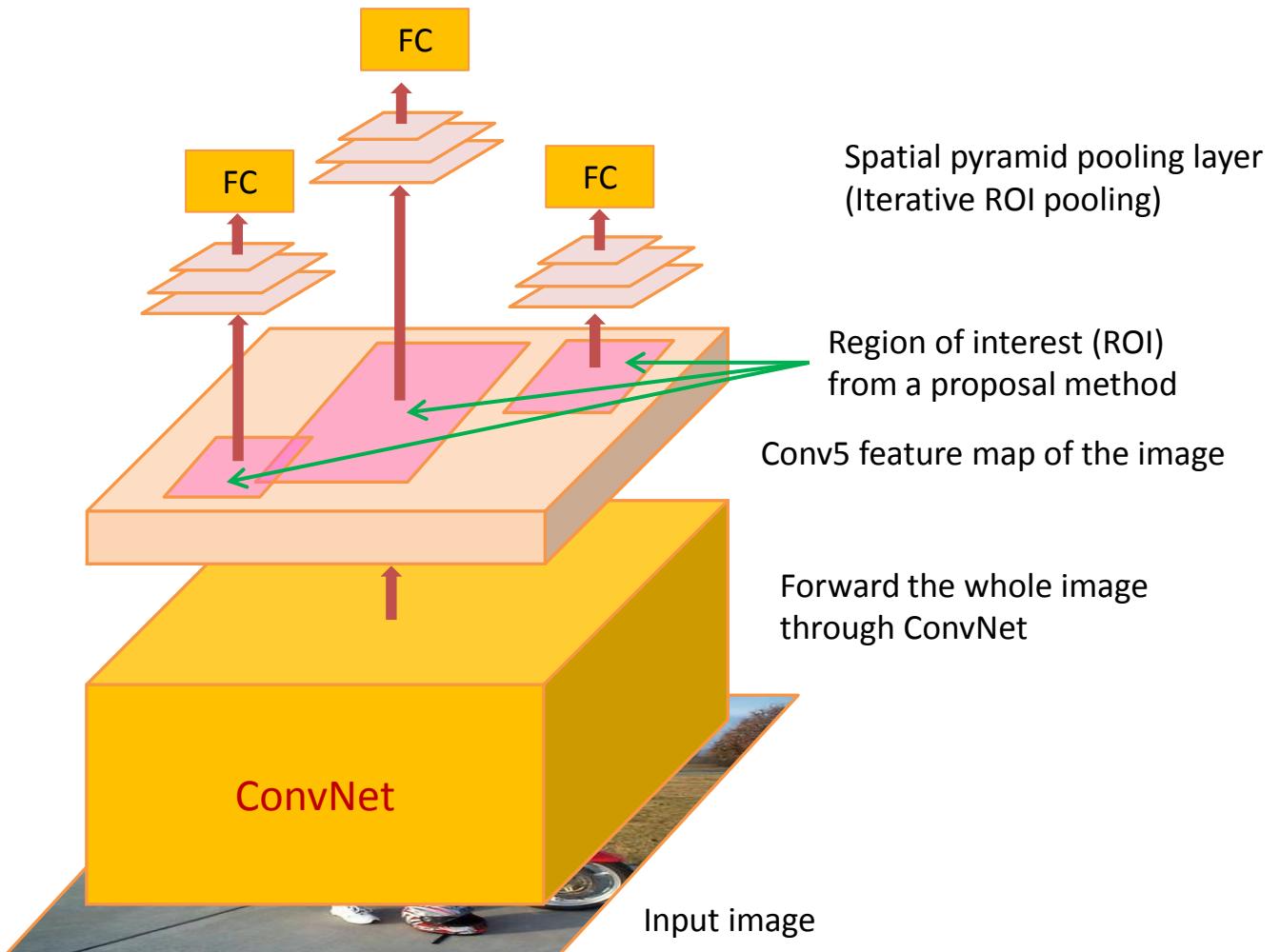
The Spatial Pyramid Pooling Network



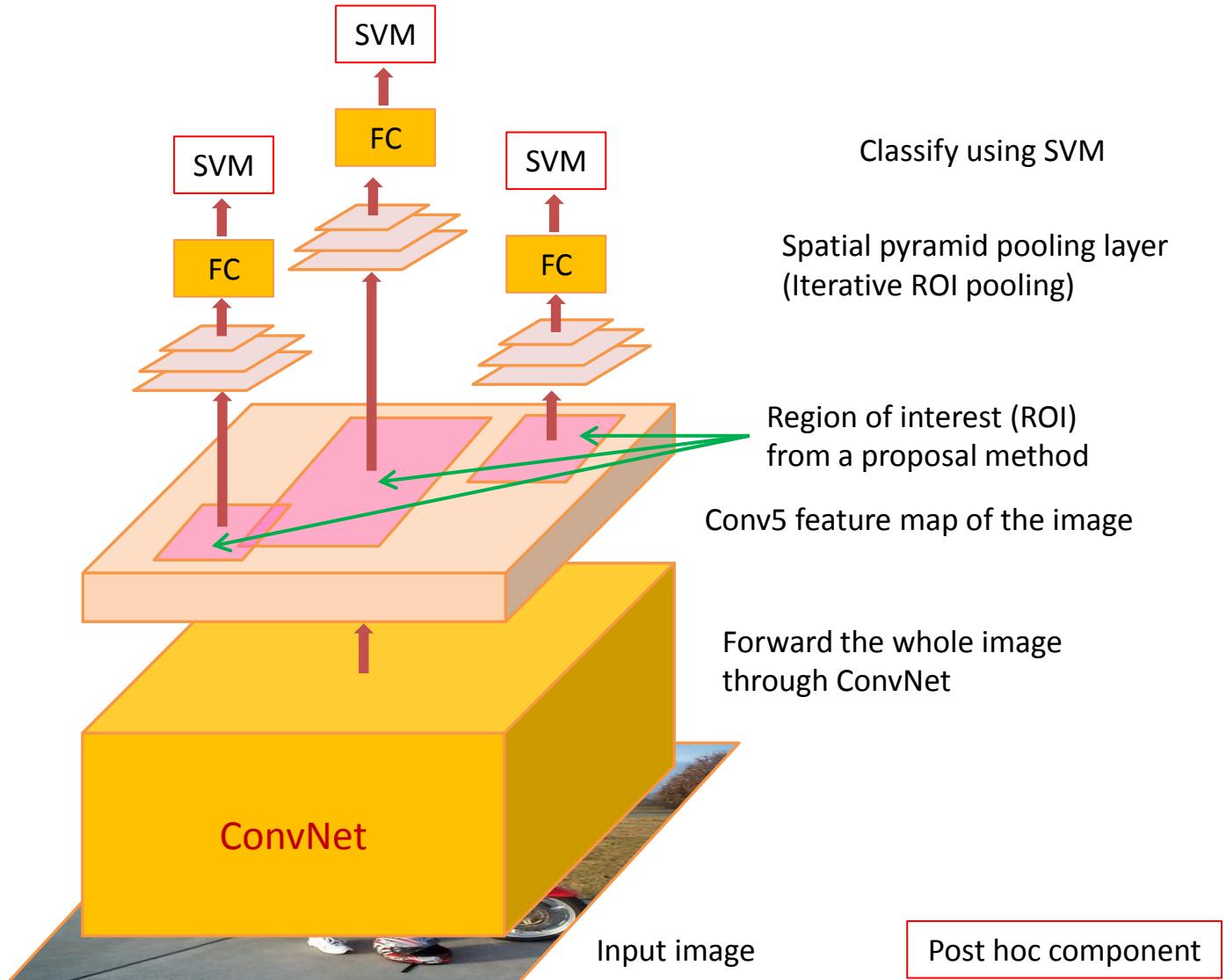
The Spatial Pyramid Pooling Network



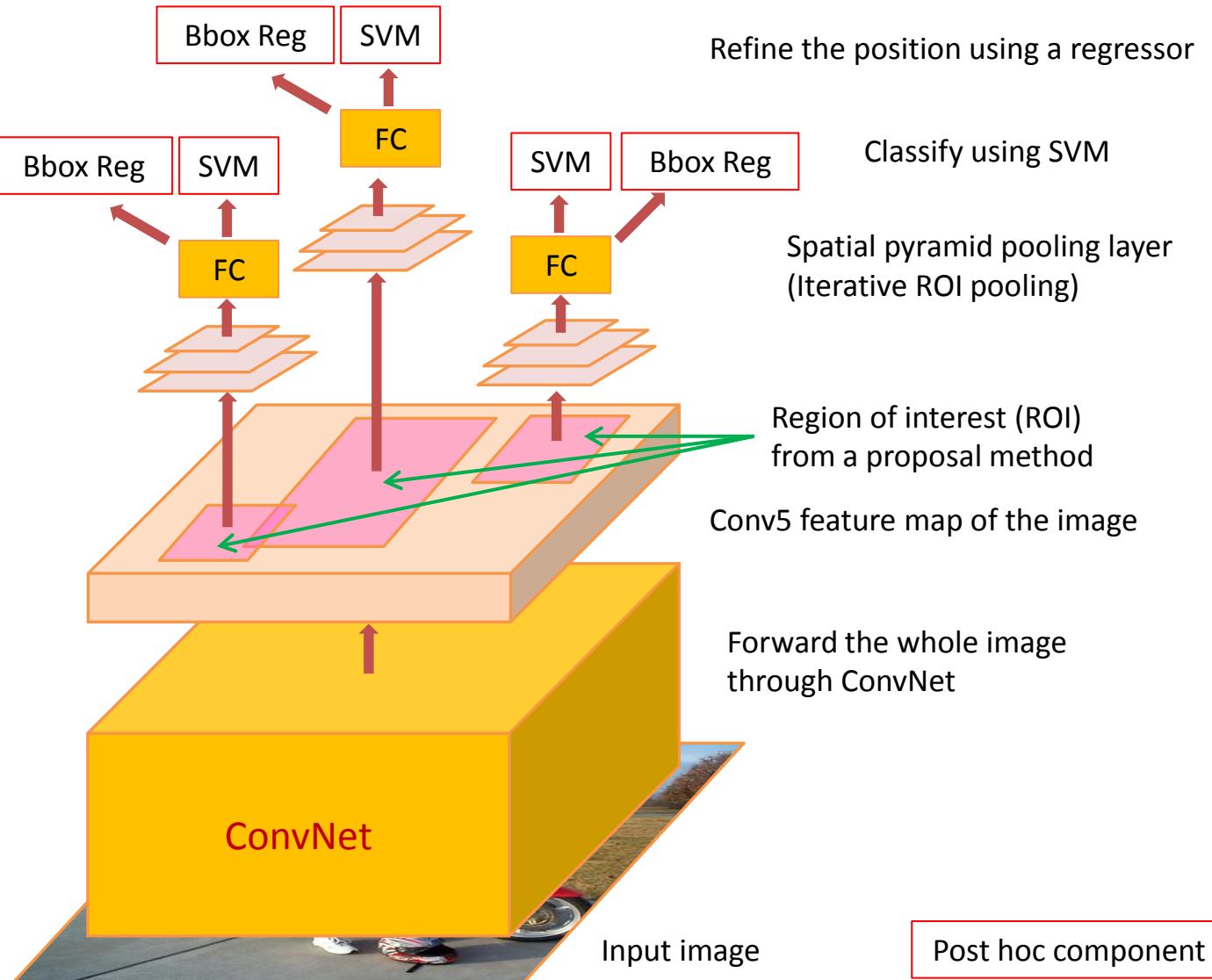
The Spatial Pyramid Pooling Network



The Spatial Pyramid Pooling Network

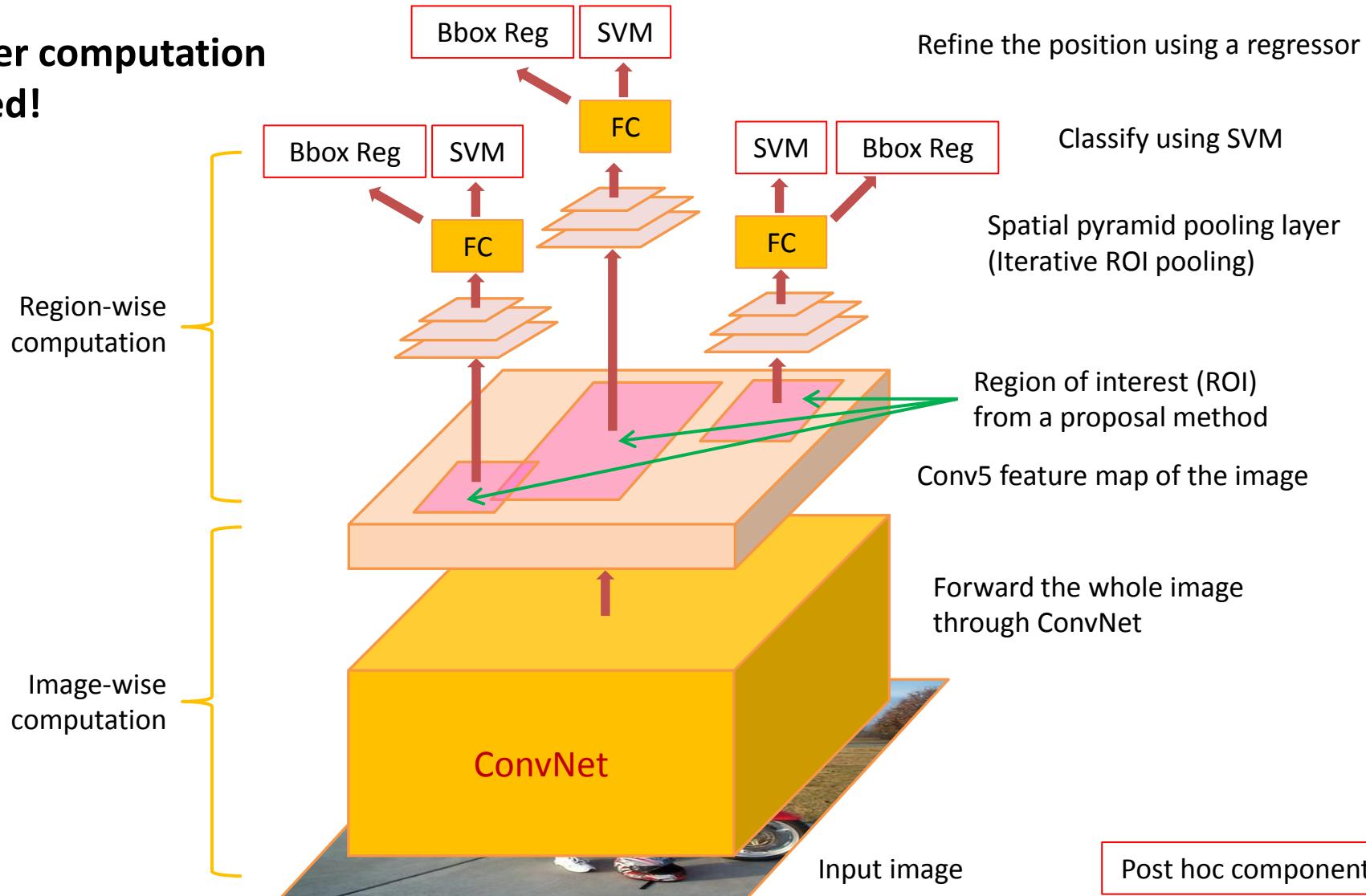


The Spatial Pyramid Pooling Network (SPP-Net)



Advantage of SPP-Net

Faster computation speed!



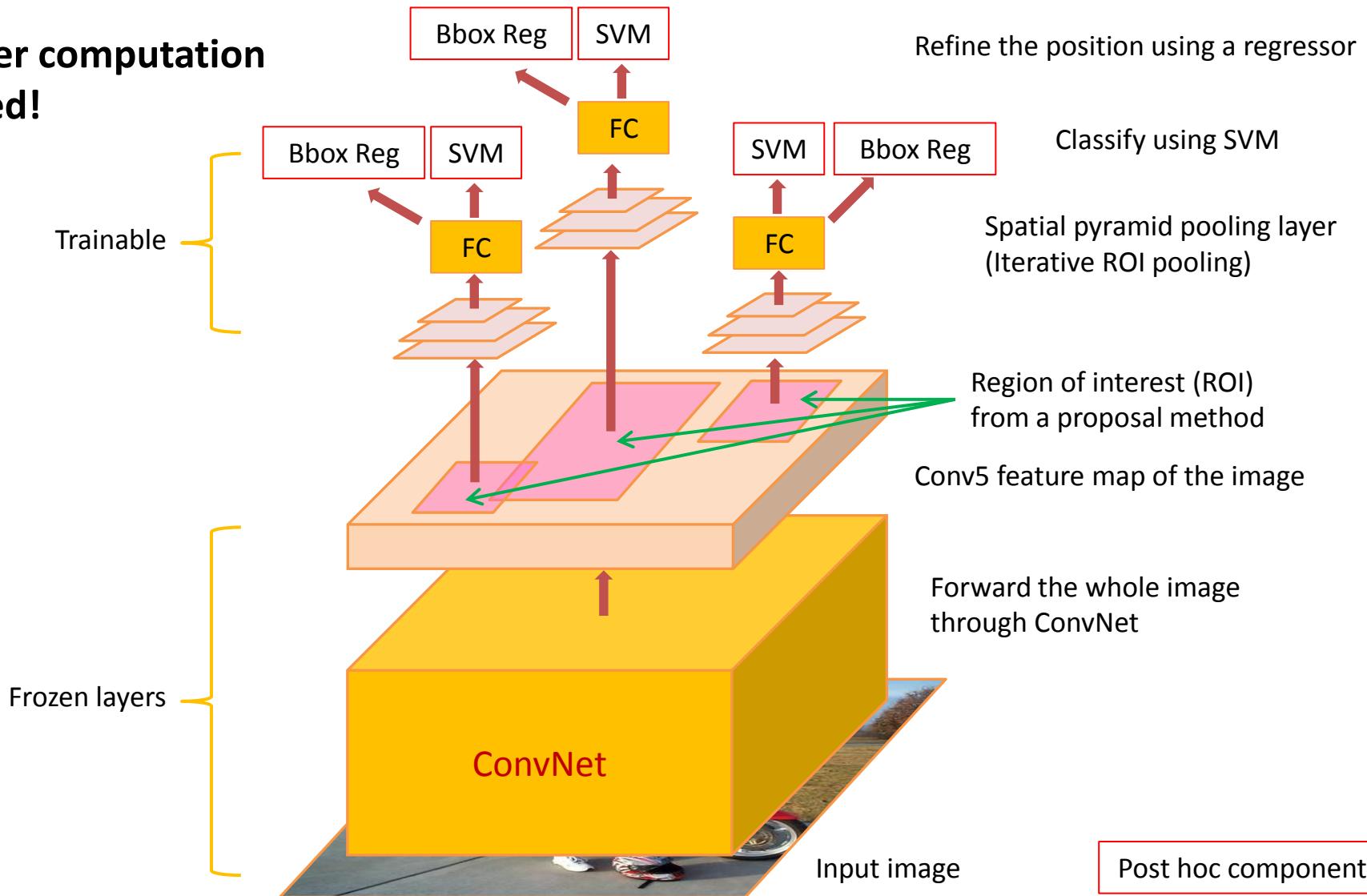


Drawbacks of SPP-net

- ▶ Inherits the rest of R-CNN's problems
 - ▶ Ad hoc training objectives
 - ▶ Training still slow
- ▶ Introduces a new problem
 - ▶ Cannot update parameters below SPP layer during training

Main Limitation of SPP-Net

Faster computation speed!





Fast R-CNN

- ▶ Fast test-time, like SPP-net
- ▶ One network, trained in one stage
- ▶ More robust than R-CNN and SPP-net

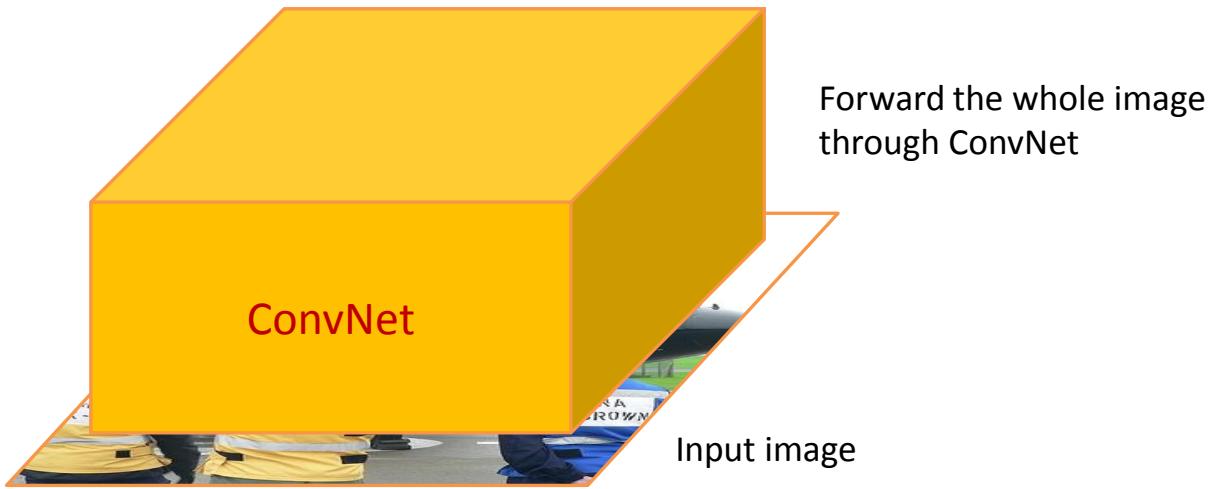


Fast R-CNN

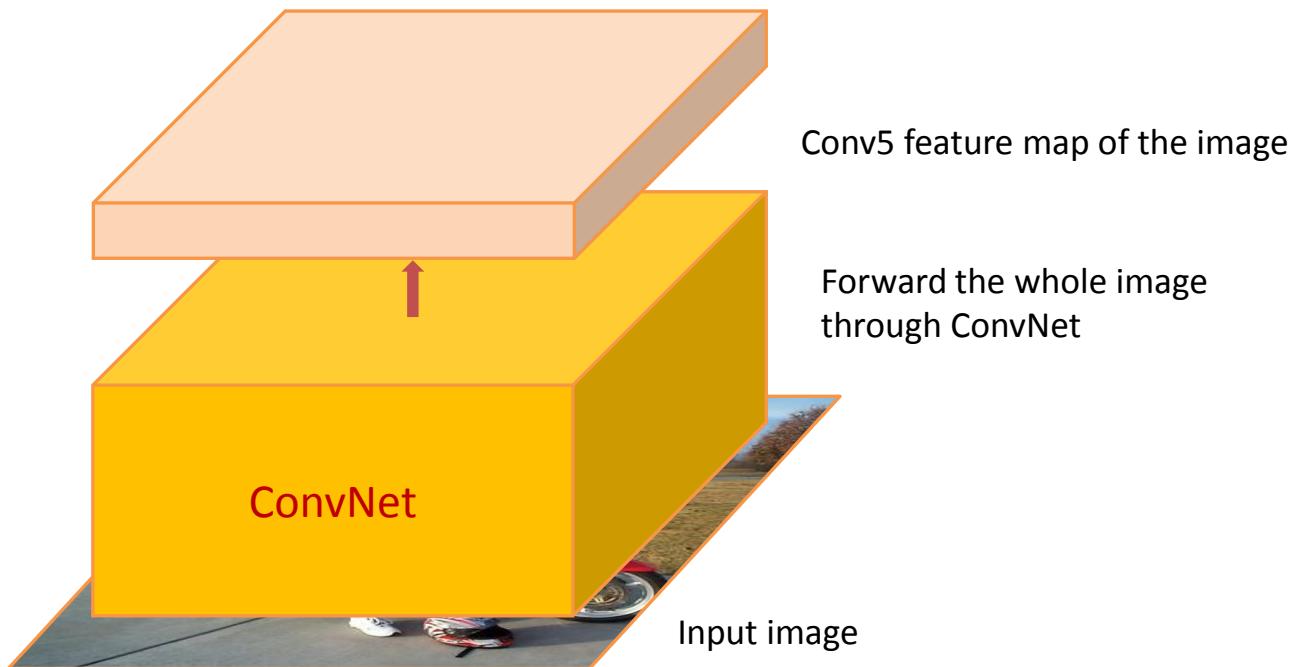


Input image

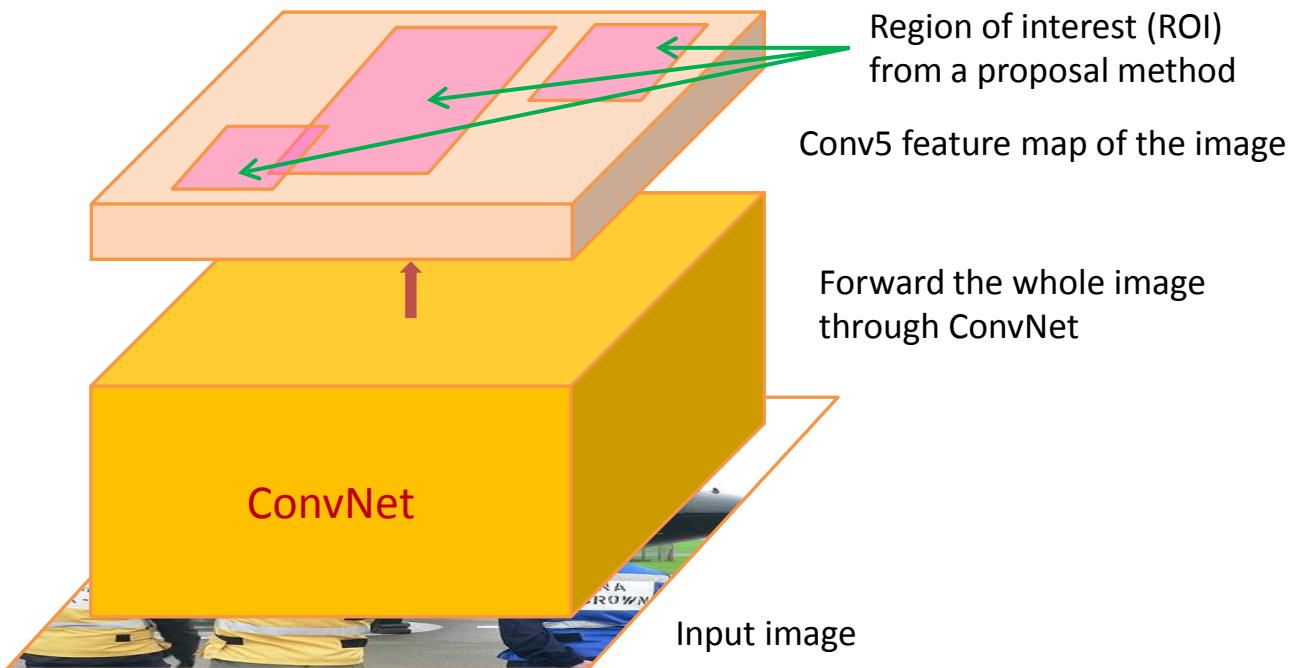
Fast R-CNN



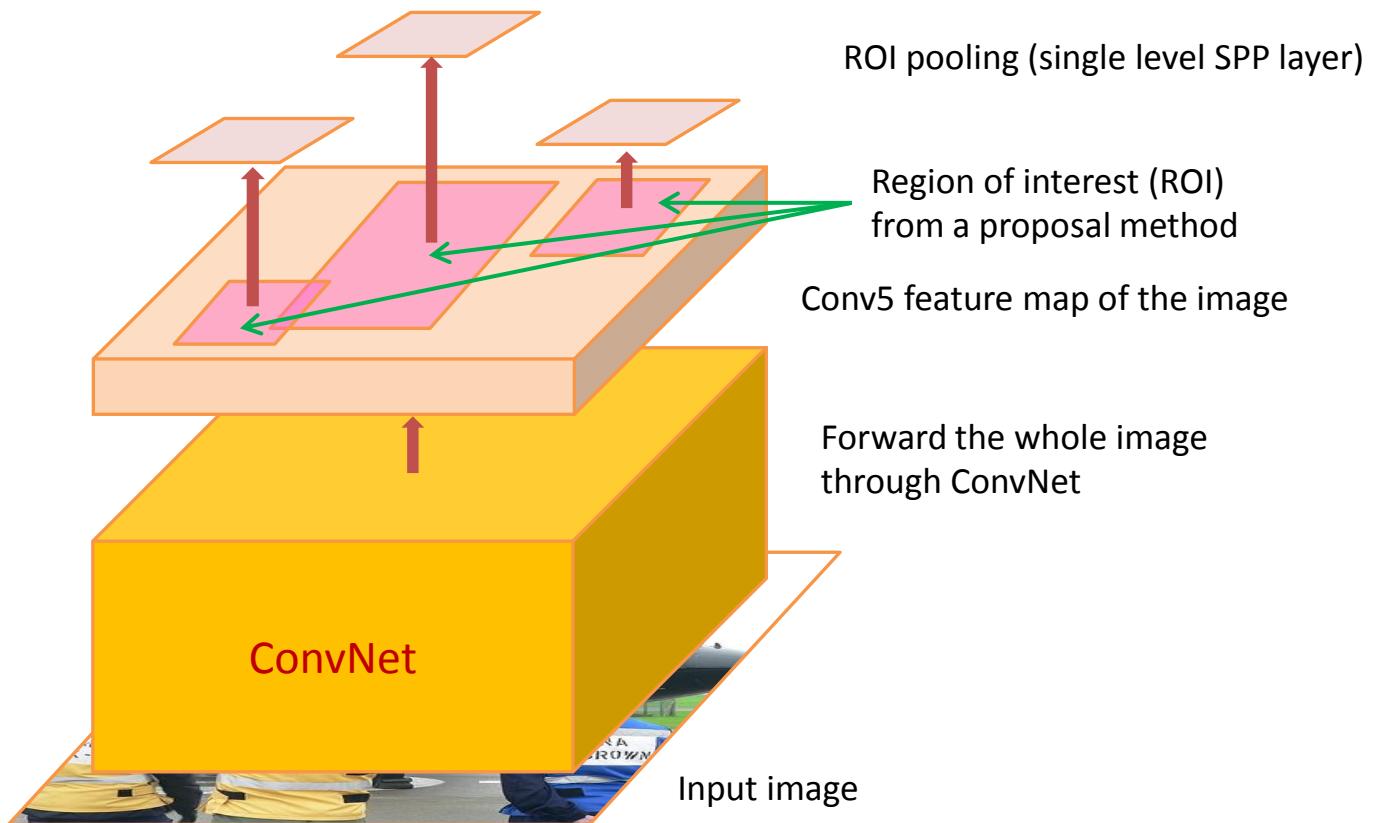
Fast R-CNN



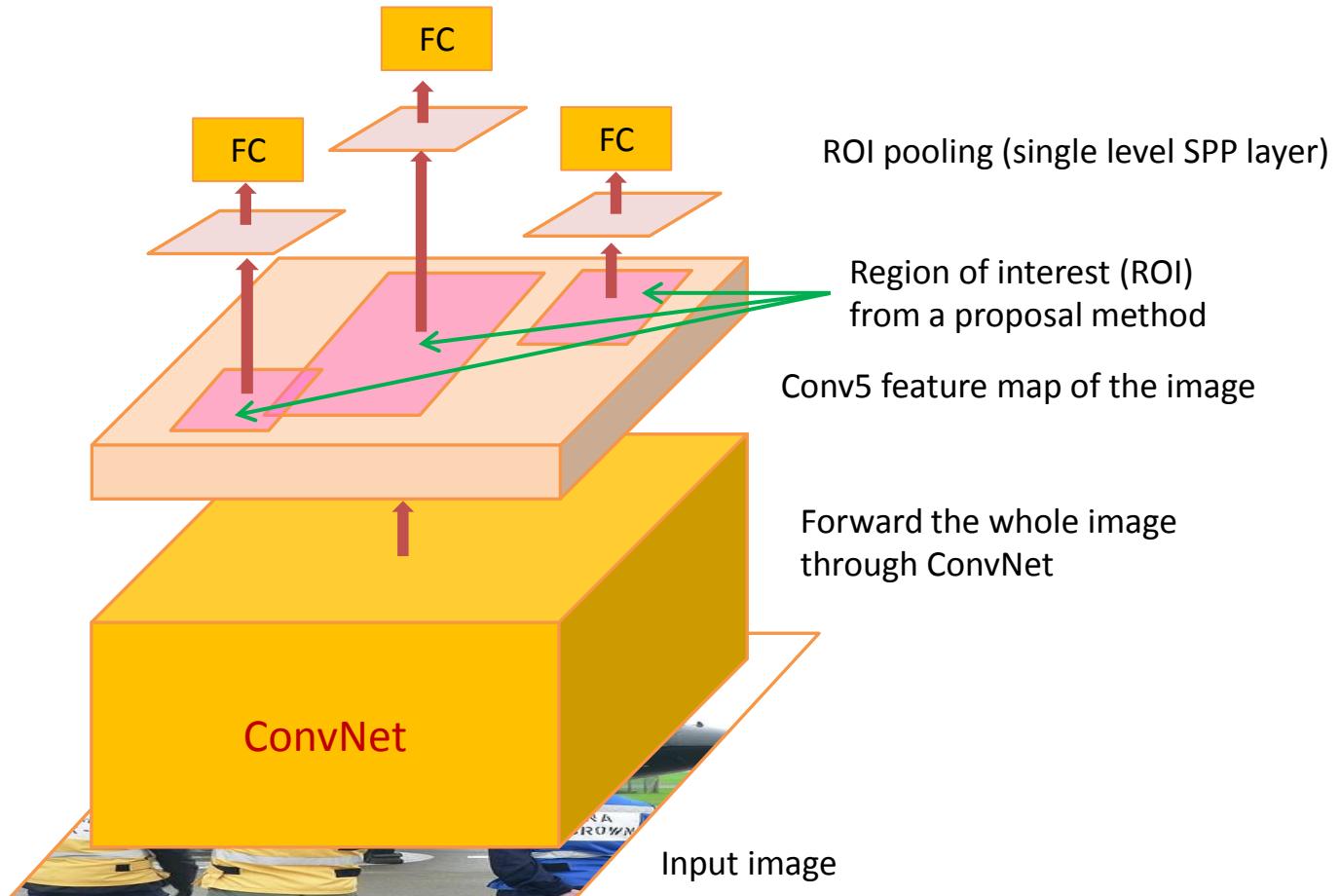
Fast R-CNN



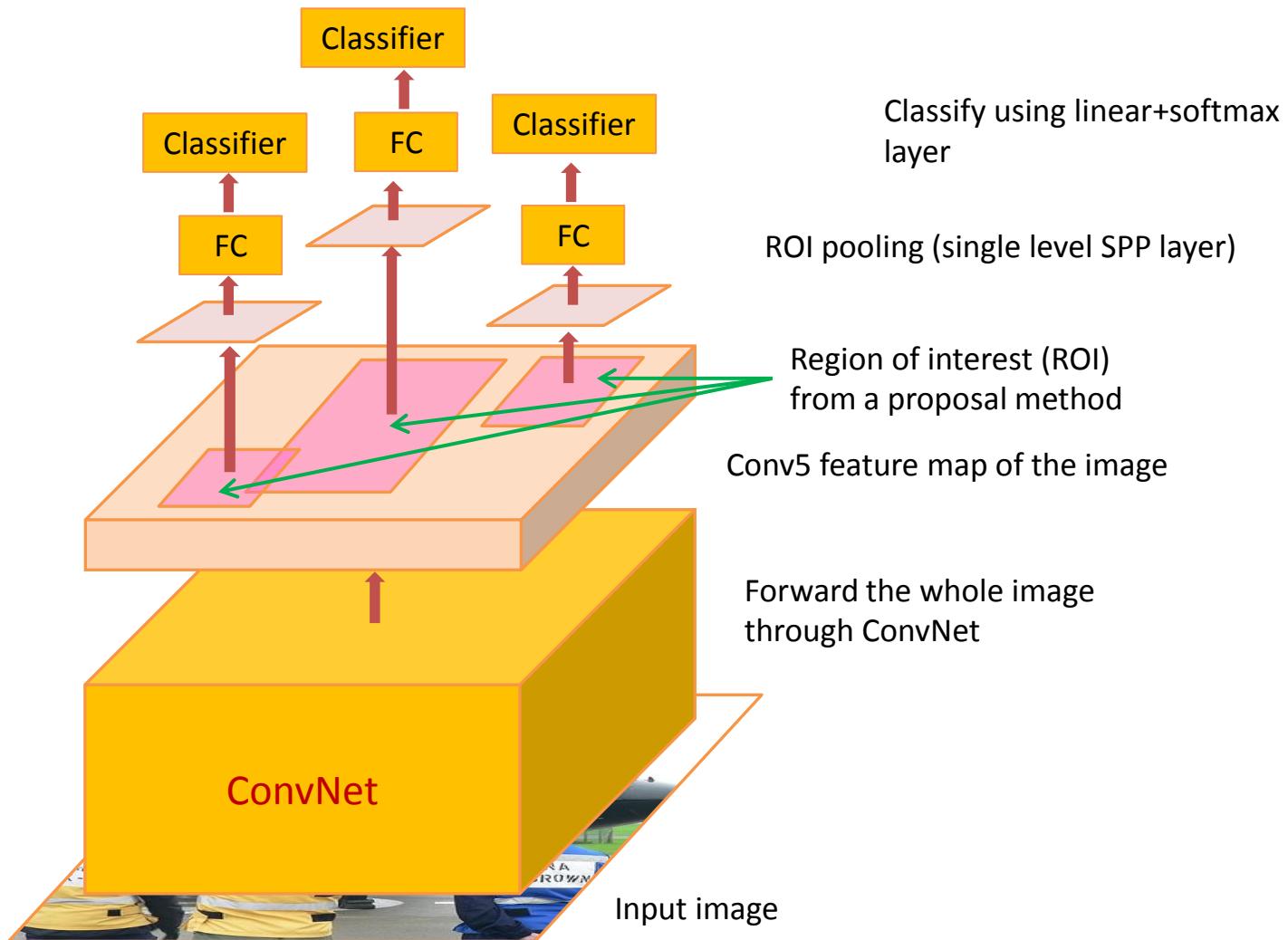
Fast R-CNN



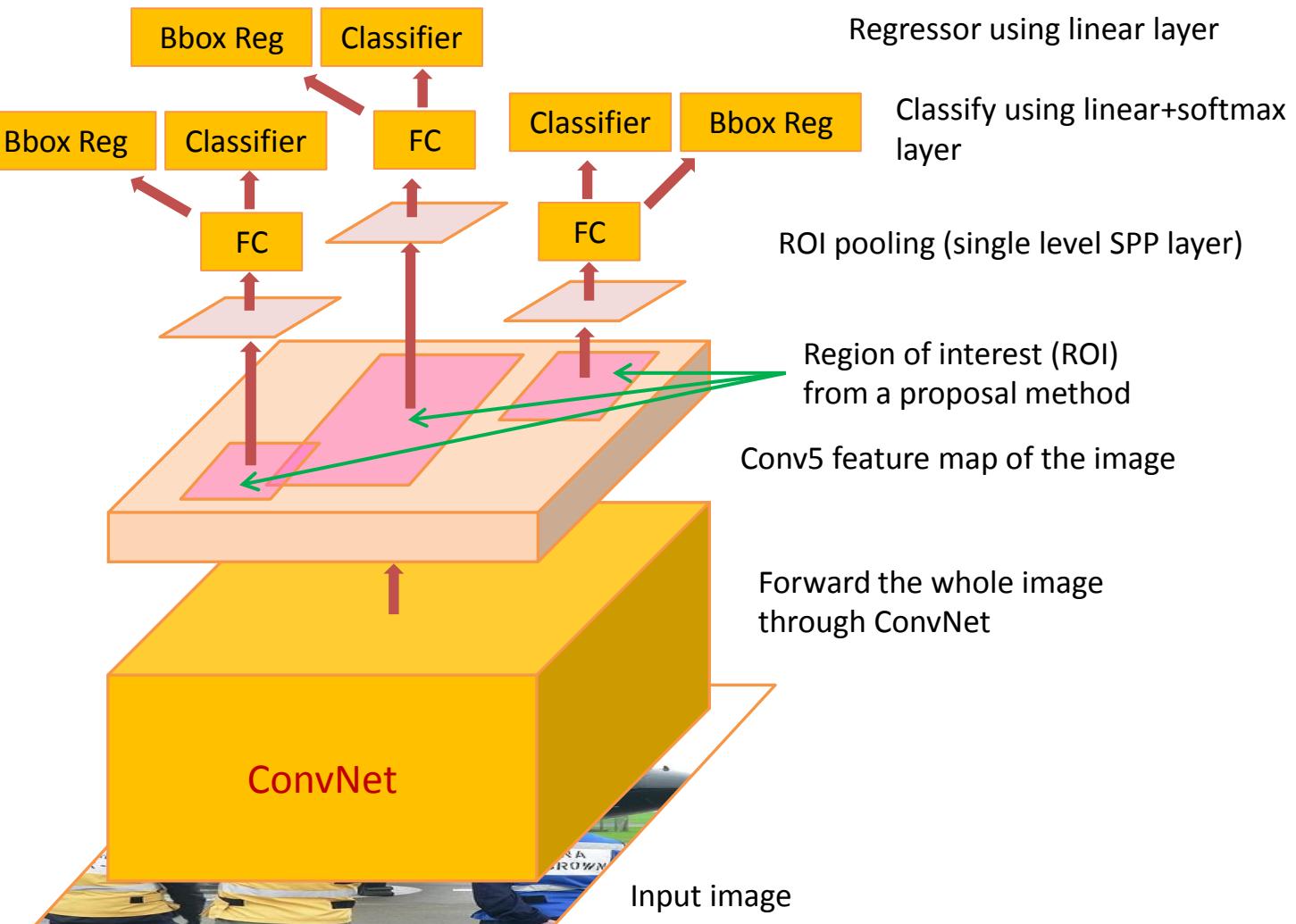
Fast R-CNN



Fast R-CNN

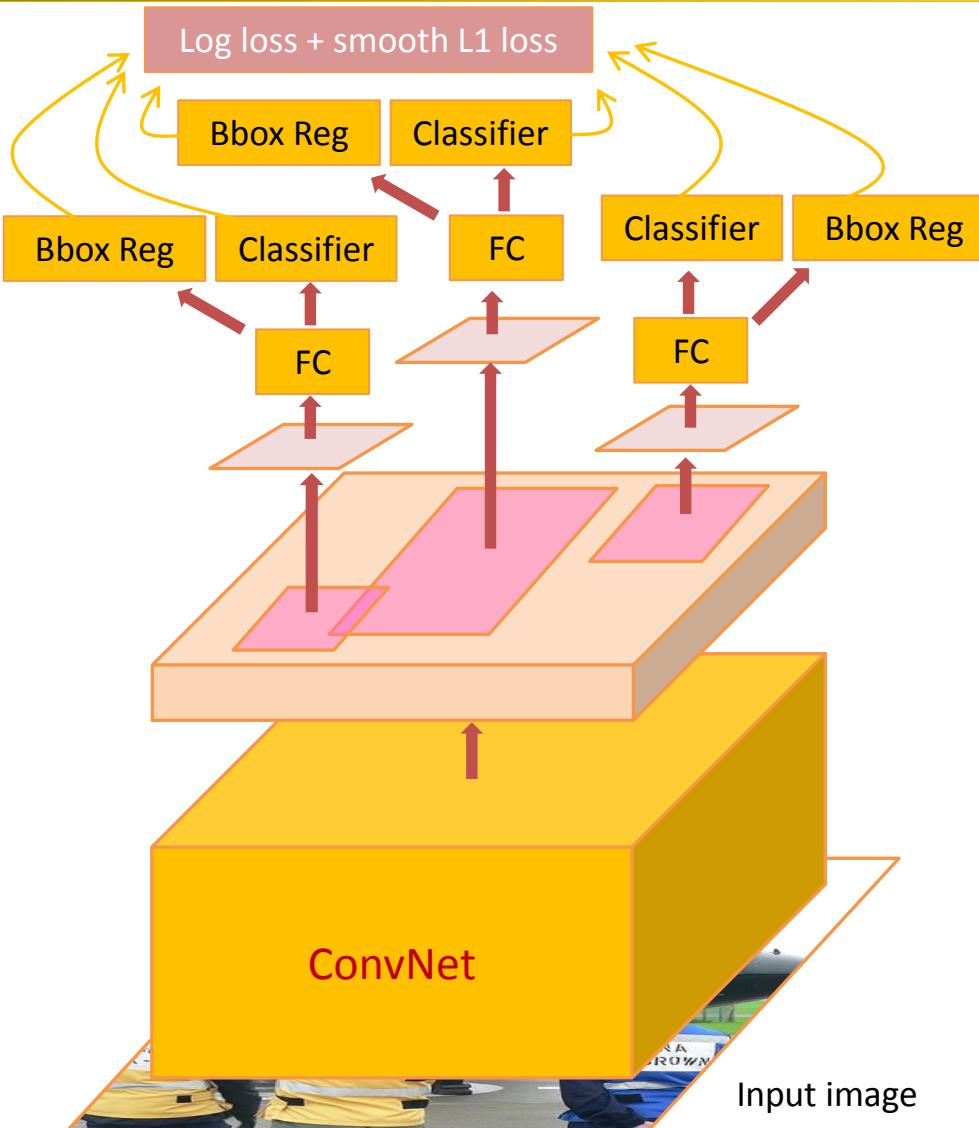


Fast R-CNN



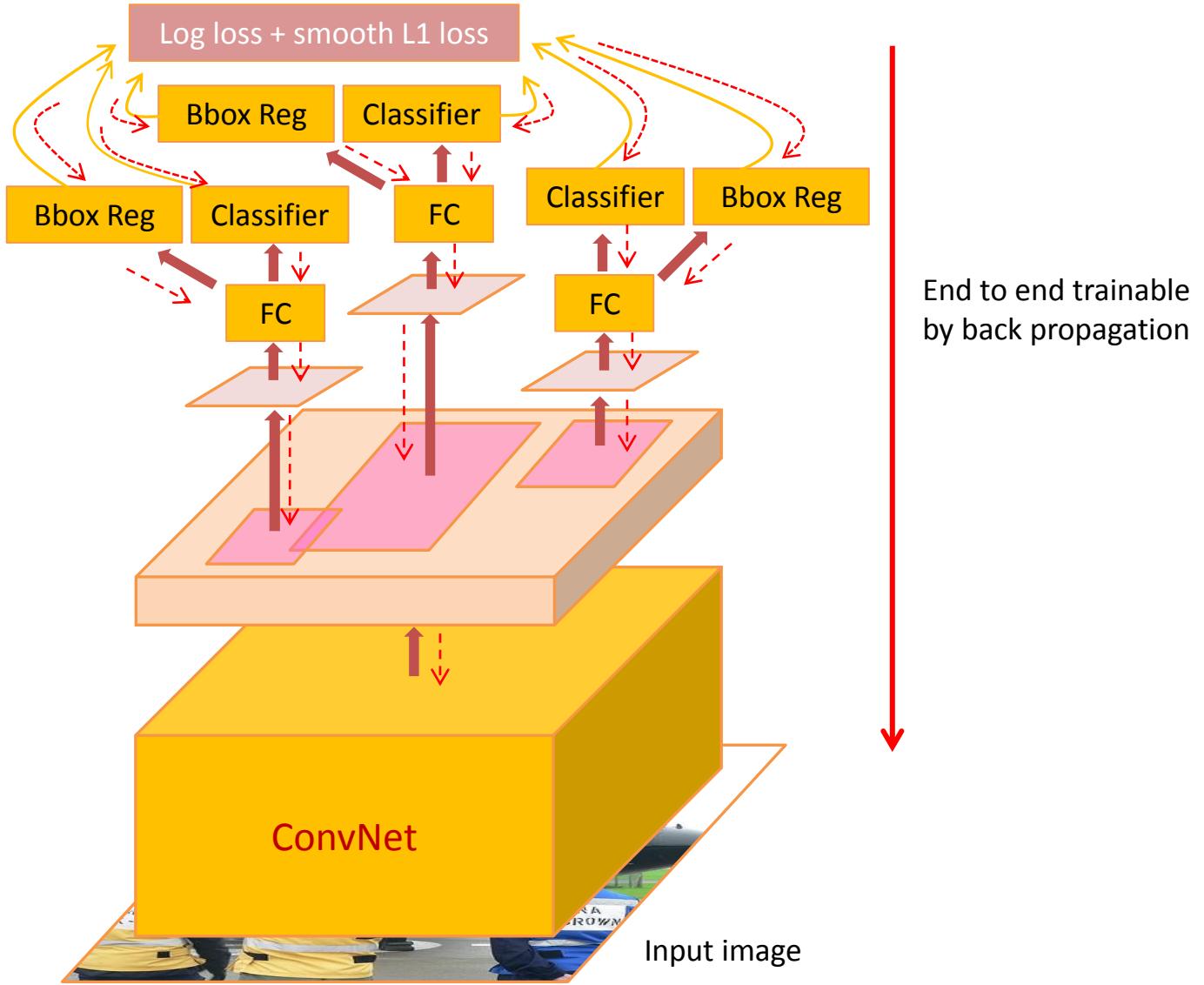


Fast R-CNN at Training Time

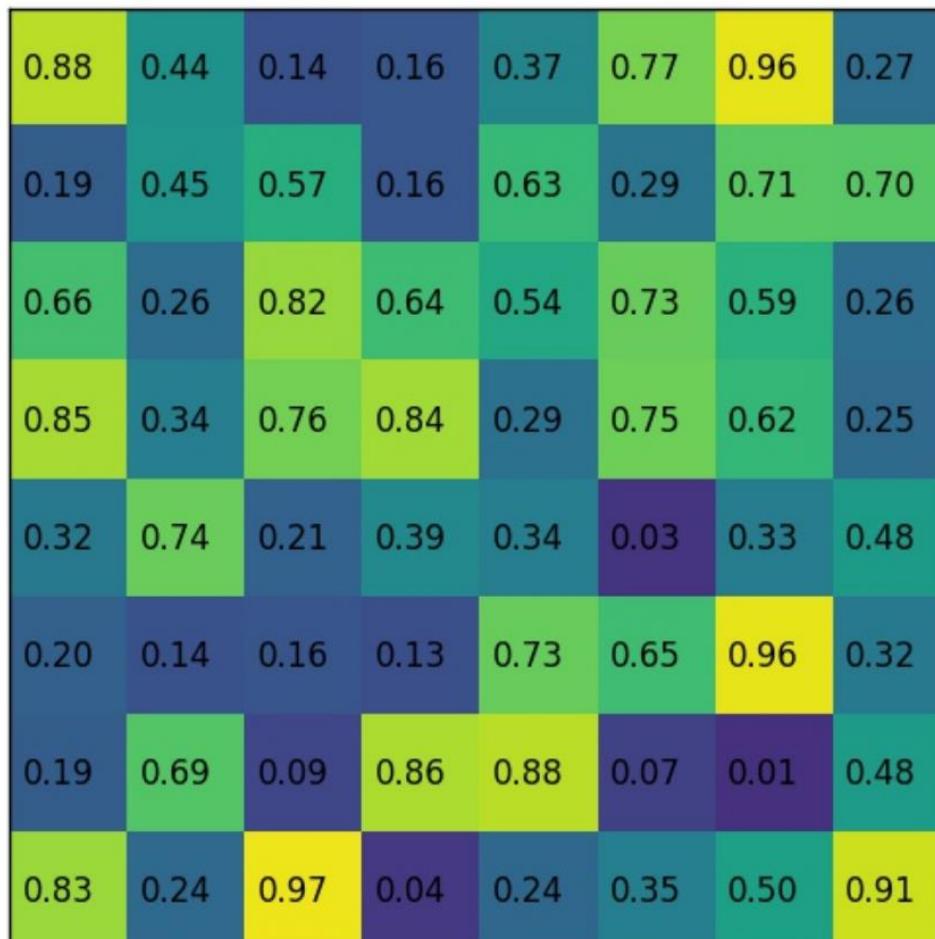




Fast R-CNN at Training Time



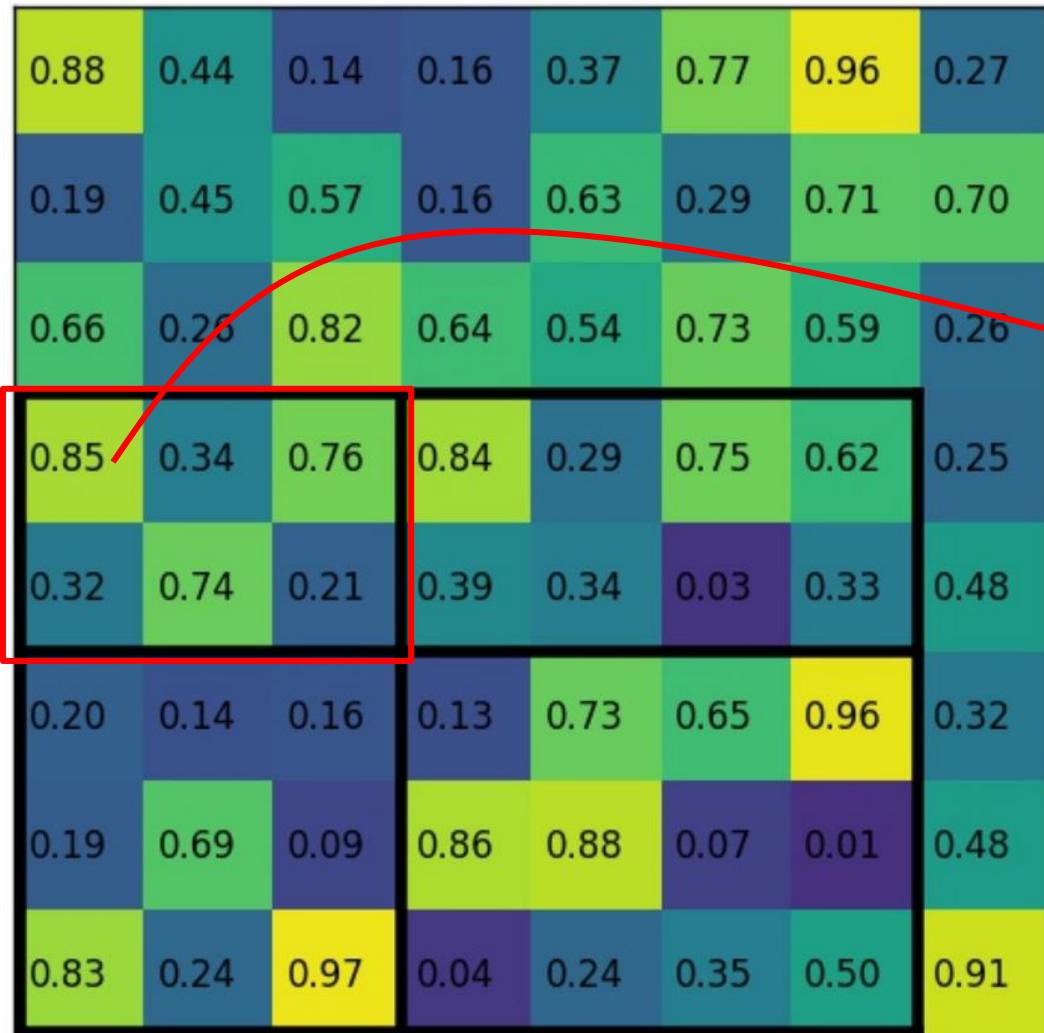
ROI Pooling



Input activation

source : <https://deepsense.io/region-of-interest-pooling-explained>

ROI Pooling

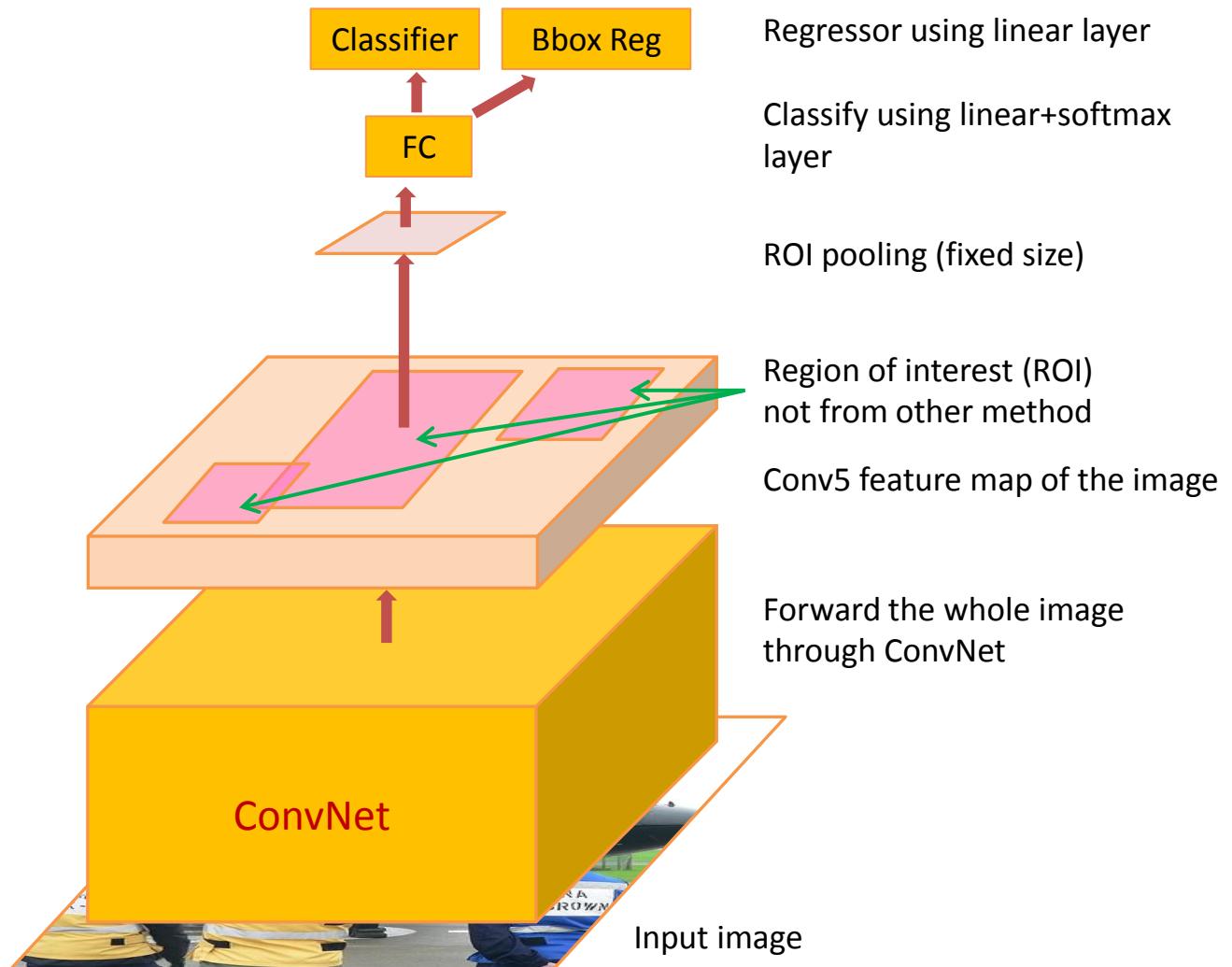


Max pooling output

source : <https://deepsense.io/region-of-interest-pooling-explained>

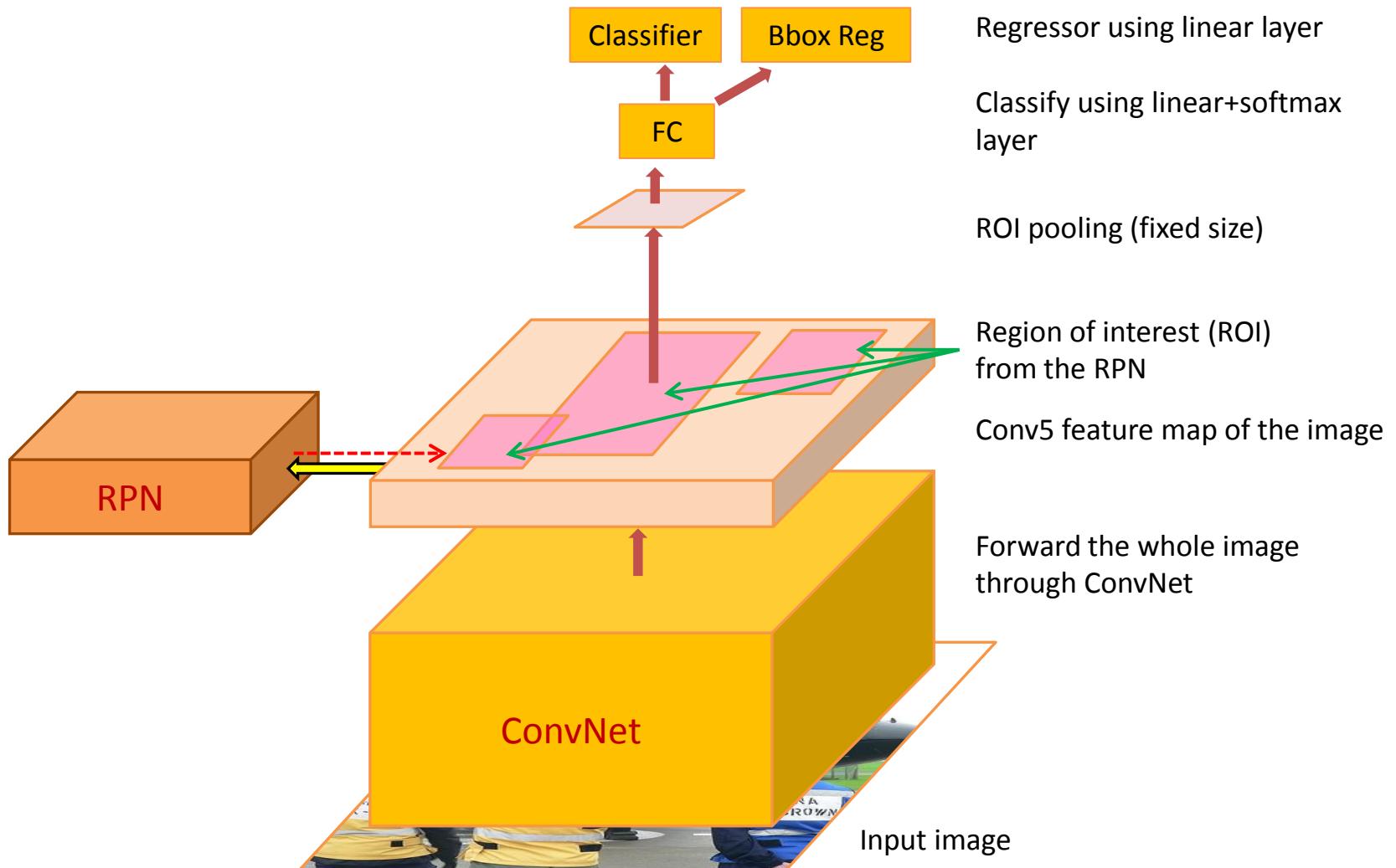


Fast R-CNN to Faster R-CNN



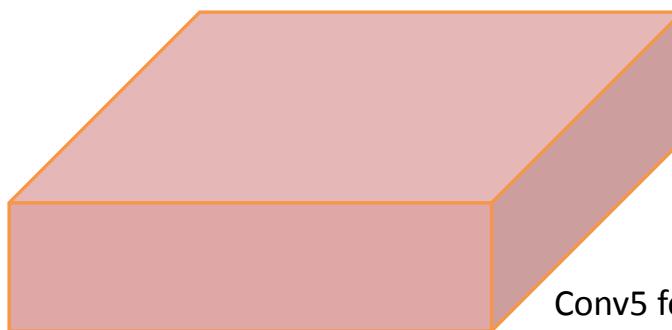


Fast R-CNN to Faster R-CNN





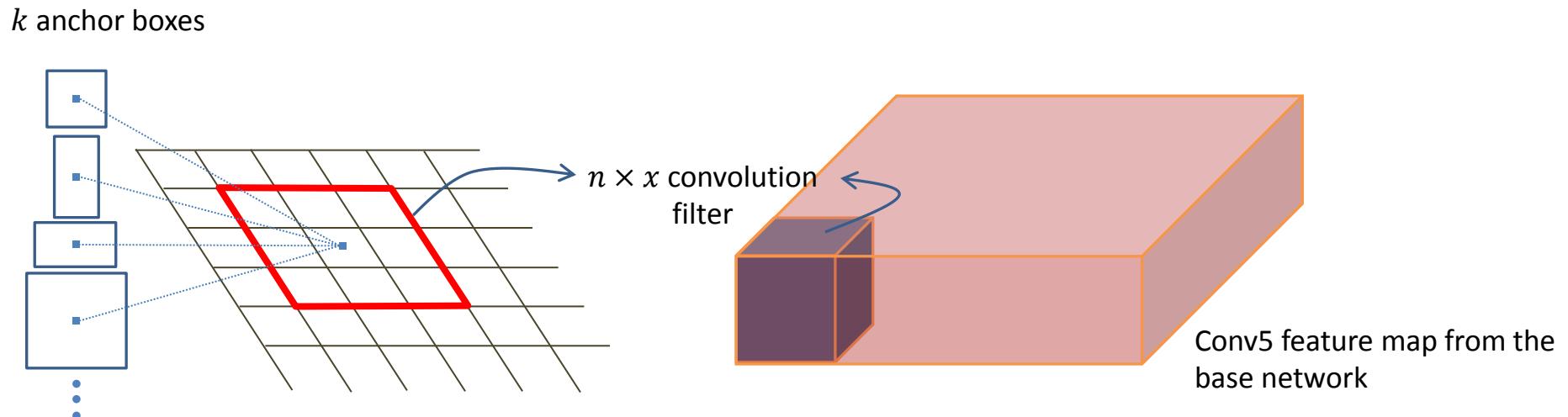
The RPN



Conv5 feature map from the
base network

The RPN

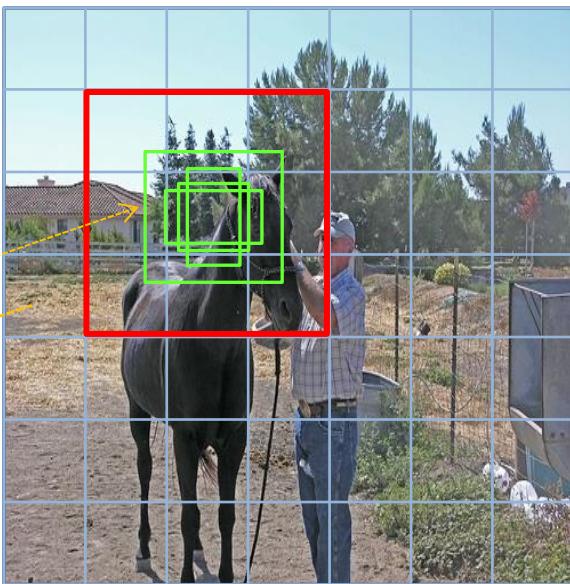
- We want to define objectness and refine the anchor boxes (initial prediction):
 - Using $n \times n$ features for each cell



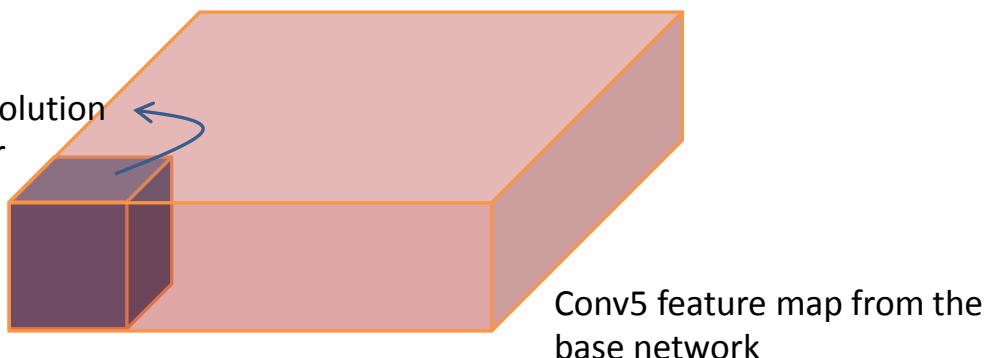
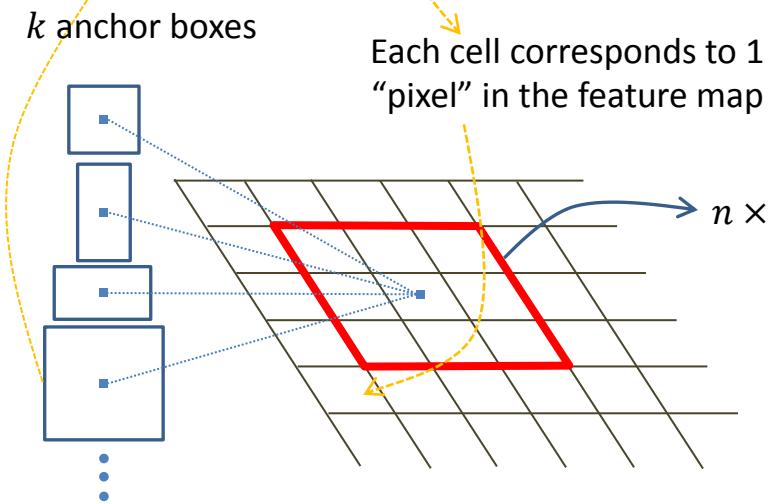
The RPN – What the Anchor Boxes are?



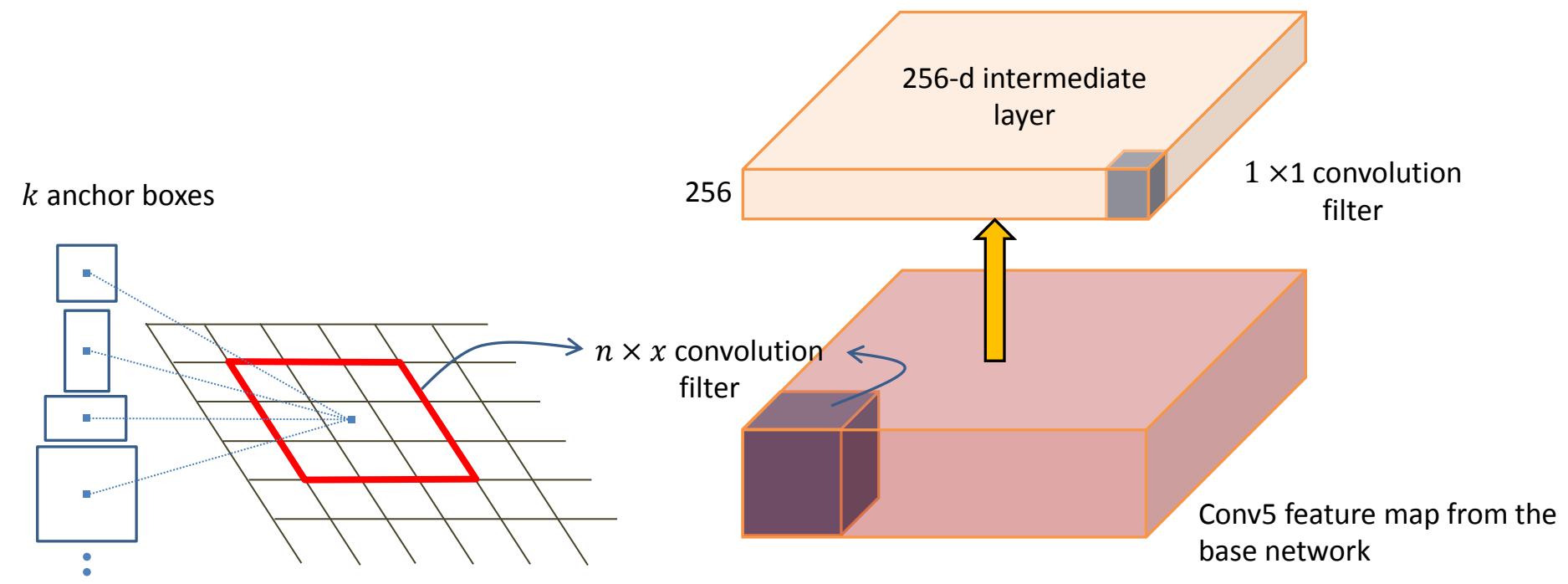
- Each anchor correspond to initial ROI prediction in the original image



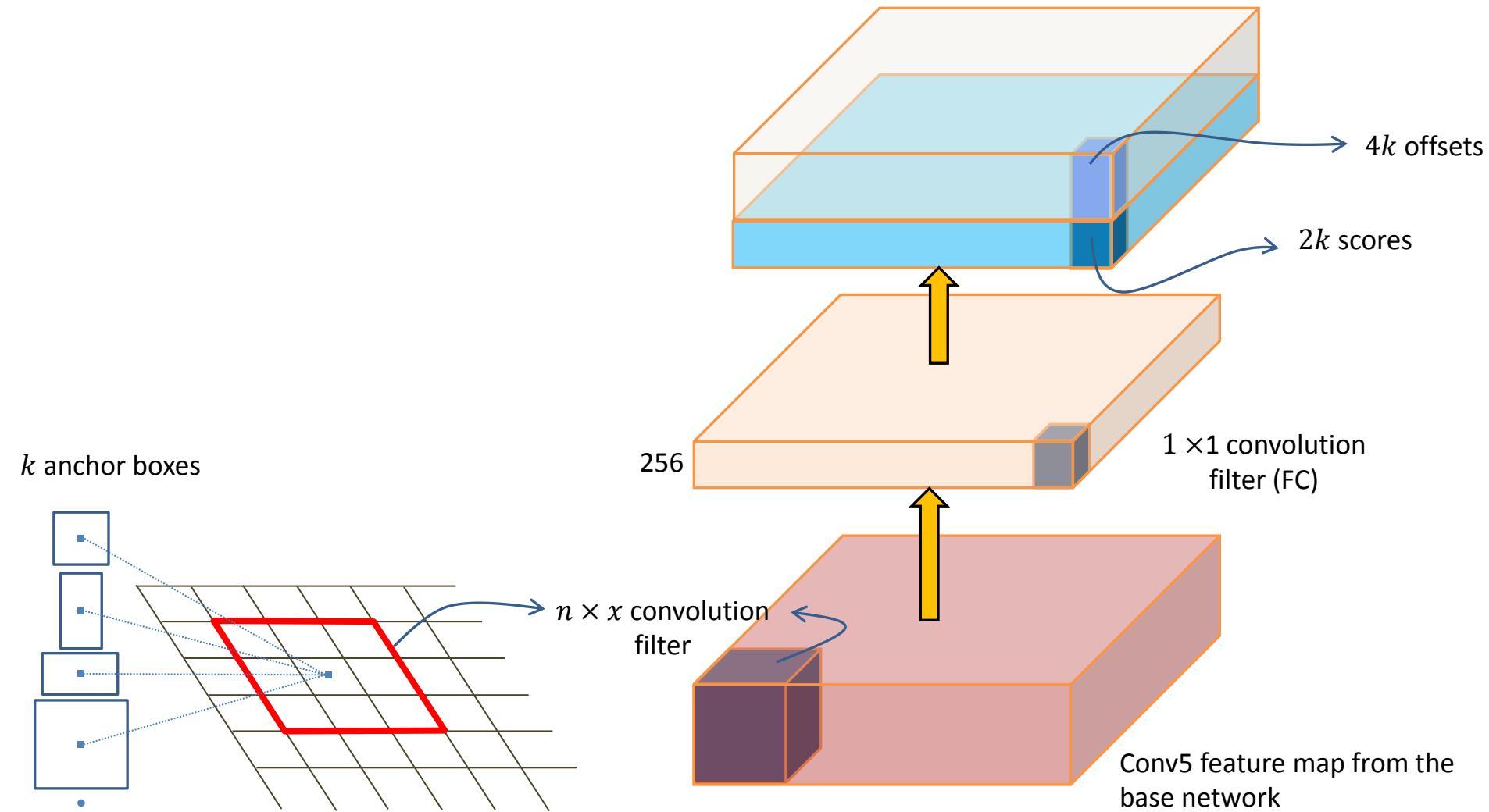
3 scales of anchor: $128^2, 256^2, 512^2$
3 aspect ratios: 1:1, 1:2, 2:1



The RPN



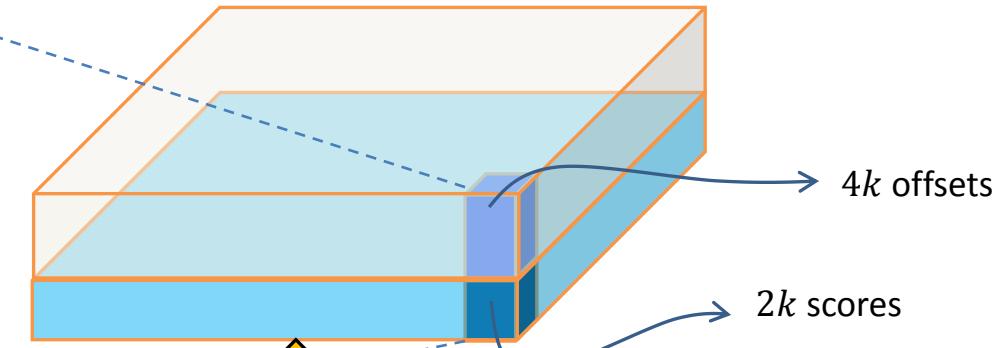
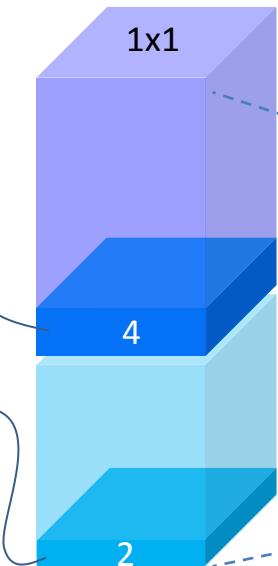
The RPN



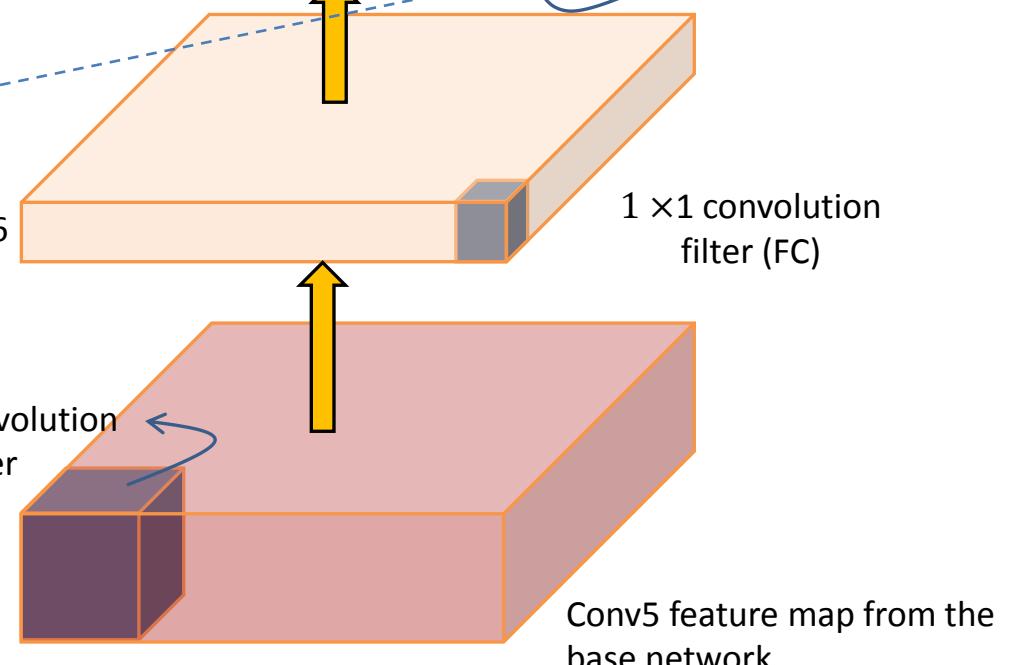
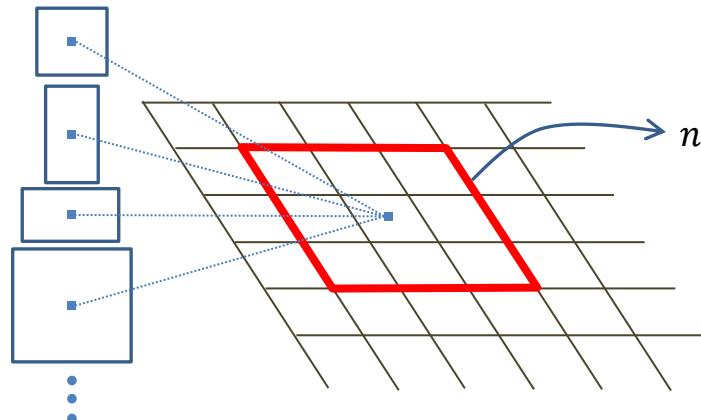
The RPN

For each anchor box:

- BBox regression:
 - $\Delta x, \Delta y, \log w, \log h$
- Predict 2 scores:
 - Prob. object or not object



k anchor boxes





Lets try on Colab!

co +





Imports

```
import numpy as np
import tensorflow as tf
from matplotlib import pyplot as plt
%matplotlib inline

#for file processing
import os
import sys

# For loading an image from internet
from PIL import Image
import cStringIO
import urllib as urlloader

# For unzipping the downloaded model
import six.moves.urllib as urllib
import tarfile

# This is needed since the notebook is stored in the object_detection folder.
sys.path.append("/content/hs1/models/research")
from object_detection.utils import ops as utils_ops
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as vis_util
```



Configurations

▼ configurations

check more pre-trained models on the [model zoo](#)



```
SAVE_FOLDER = 'hsd/'  
  
# What model to download.  
MODEL_NAME = 'faster_rcnn_nas_coco_2018_01_28'  
MODEL_FILE = MODEL_NAME + '.tar.gz'  
DOWNLOAD_BASE = 'http://download.tensorflow.org/models/object_detection/'  
  
# Path to frozen detection graph.  
# This is the actual model that is used for the object detection.  
PATH_TO_CKPT = SAVE_FOLDER+MODEL_NAME + '/frozen_inference_graph.pb'  
  
# List of the strings that is used to add correct label for each box.  
PATH_TO_LABELS = os.path.join('hsd/models/research/object_detection/data/',  
                           'mscoco_label_map.pbtxt')  
  
NUM_CLASSES = 90
```



The Model Zoo

COCO-trained models {#coco-models}

Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssdlite_mobilenet_v2_coco	27	22	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes
faster_rcnn_resnet50_coco	89	30	Boxes
faster_rcnn_resnet50_lowproposals_coco	64		Boxes
rfcn_resnet101_coco	92	30	Boxes
faster_rcnn_resnet101_coco	106	32	Boxes
faster_rcnn_resnet101_lowproposals_coco	82		Boxes
faster_rcnn_inception_resnet_v2_atrous_coco	620	37	Boxes
faster_rcnn_inception_resnet_v2_atrous_lowproposals_coco	241		Boxes
faster_rcnn_nas	1833	43	Boxes



Download the Pre-trained Model

▼ Download and unzip the pre-trained model

```
▶ opener = urllib.request.URLopener()
opener.retrieve(DOWNLOAD_BASE + MODEL_FILE, SAVE_FOLDER+MODEL_FILE)
tar_file = tarfile.open(SAVE_FOLDER+MODEL_FILE)
for file in tar_file.getmembers():
    file_name = os.path.basename(file.name)
    if 'frozen_inference_graph.pb' in file_name:
        tar_file.extract(file, SAVE_FOLDER)
```

Check the downloaded model

```
▶ !ls hsi
◀ faster_rcnn_nas_coco_2018_01_28  faster_rcnn_nas_coco_2018_01_28.tar.gz  models
```



Load the Model and Label Map

▼ Load the model

```
▶ detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')
```

▼ Load the label map

```
▶ label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(
    label_map, max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)
```



Few Wrappers

```
def run_inference_for_single_image(image, graph):
    with graph.as_default():
        with tf.Session() as sess:
            # Get handles to input and output tensors
            ops = tf.get_default_graph().get_operations()
            all_tensor_names = {output.name for op in ops for output in op.outputs}
            tensor_dict = {}

# we will load an image from URL and then convert it to numpy array
def load_image_into_numpy_array(URL):
    file = cStringIO.StringIO(urllib.urlopen(URL).read())
    image = Image.open(file)
    (im_width, im_height) = image.size
    return np.array(image.getdata()).reshape(
        (im_height, im_width, 3)).astype(np.uint8)
```

And Finally the Detection Function



```
def detect(image_path, figsize=(20,20)):  
    image_np = load_image_into_numpy_array(image_path)  
  
    # Actual detection.  
    output_dict = run_inference_for_single_image(image_np, detection_graph)  
    # Visualization of the results of a detection.  
    vis_util.visualize_boxes_and_labels_on_image_array(  
        image_np,  
        output_dict['detection_boxes'],  
        output_dict['detection_classes'],  
        output_dict['detection_scores'],  
        category_index,  
        instance_masks=output_dict.get('detection_masks'),  
        use_normalized_coordinates=True,  
        line_thickness=8)  
  
    plt.figure(figsize=figsize)  
    plt.imshow(image_np)  
    plt.grid(False)
```

Finally, Make a Form to Run the Detection (1/2)



The screenshot shows a workflow editor interface. At the top, there is a toolbar with a play button, a plus sign labeled '+ form' (circled in red), a trash can, a comment icon, and a link icon. Below the toolbar, there is a sequence of nodes connected by arrows: a play button node followed by a 'CODE' node and a 'TEXT' node. The 'TEXT' node contains the text '#@title Default title text'. To the right of the TEXT node is a 'Default title text' node with a play button and a blue user icon.

Below these nodes is a modal dialog titled 'Add new form field'. The dialog has fields for 'Variable name' (set to 'variable_name') and 'Variable type' (set to 'raw'). A dropdown menu for 'Form field type' is open, showing options: boolean, date, integer, number, raw (selected and highlighted in grey), and string. The background of the editor is dark grey.

Finally, Make a Form to Run the Detection (2/2)

- ▶ Link the form to the detection function,
 - ▶ Add validator as necessary
 - ▶ Hide the code to make it more beautiful

▼ Inference

The screenshot shows a Jupyter Notebook cell containing Python code for image detection:

```
#@title Image URL { run: "auto", vertical: true }
img_url = "http://host.robots.ox.ac.uk/vision/datasets/coco/images/000000000000.jpg"
size = 19 #@param {type:"integer"}

if size <5:
    size = 5

detect(img_url, figsize=(size,size))
```

To the right, a 'both' field in a sidebar is highlighted with a red circle around its edit icon. Below it, the 'Image URL' and 'size' parameters are shown with their values: `img_url: " http://host.robots.ox.ac.uk/vision/datasets/coco/images/000000000000.jpg "` and `size: 19`.

A red arrow points from the 'Edit form attributes' dialog box at the bottom to the 'Image URL' parameter in the sidebar.

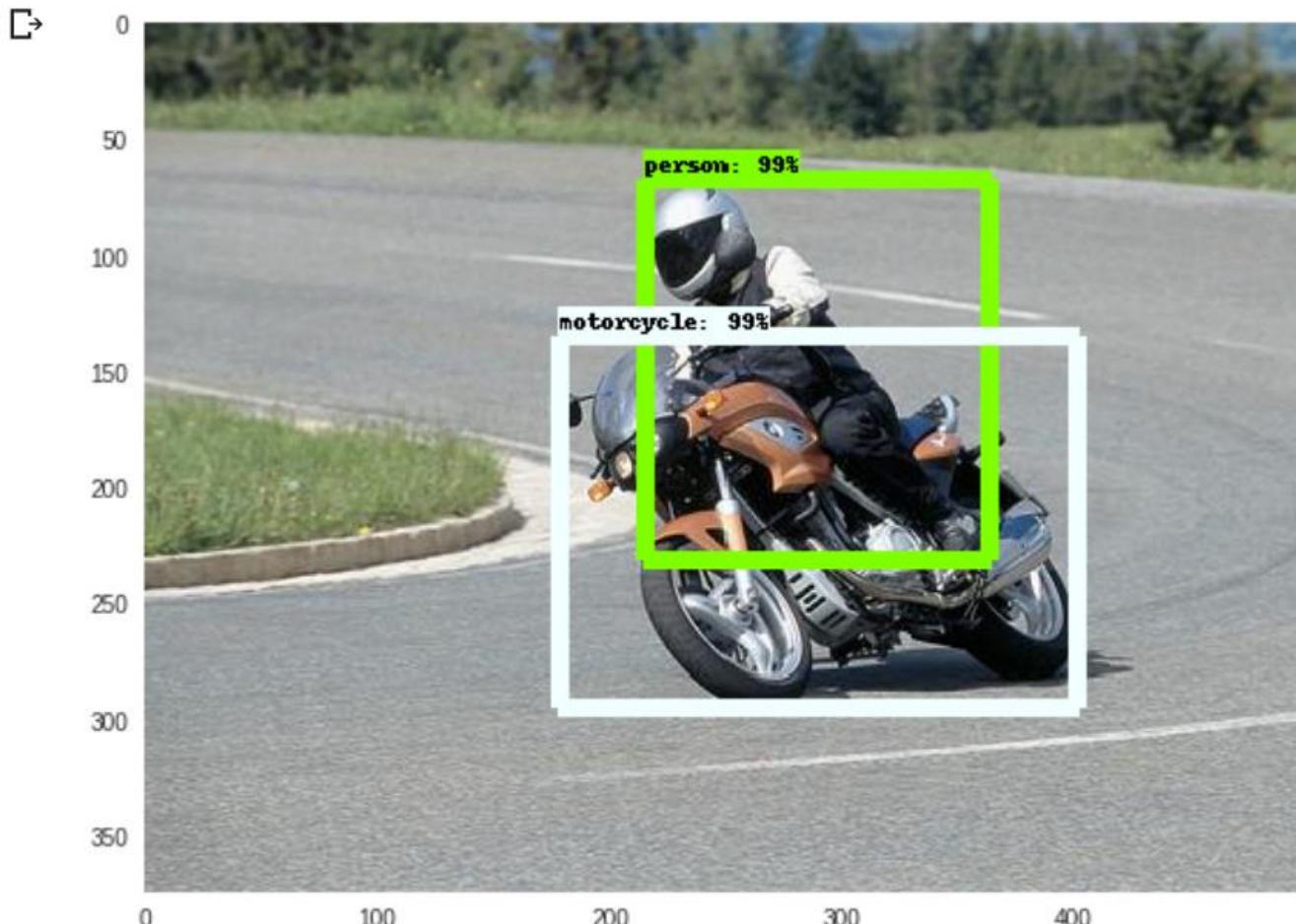
The 'Edit form attributes' dialog box contains the following settings:

- Title text: Image URL
- Initial form visibility: last shown (default)
- Output height: px
- Form width: px
- Checkboxes:
 - Auto-execute cell when fields change
 - Display output below form
- Buttons: CANCEL, SAVE

Example Output

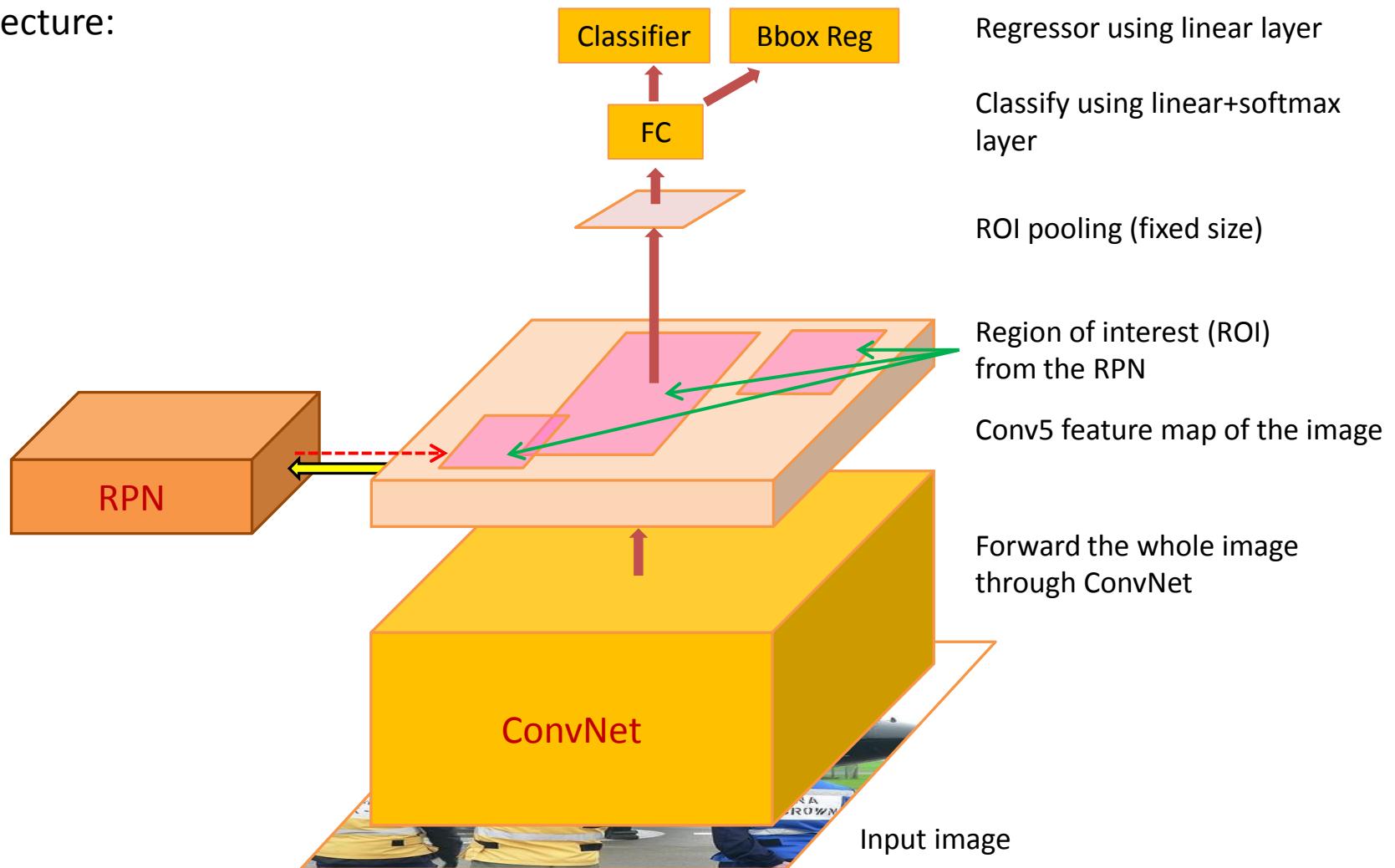
```
img_url: "http://host.robots.ox.ac.uk/pascal/VOC/voc2010/segeexamples/images/2"
```

size: 9



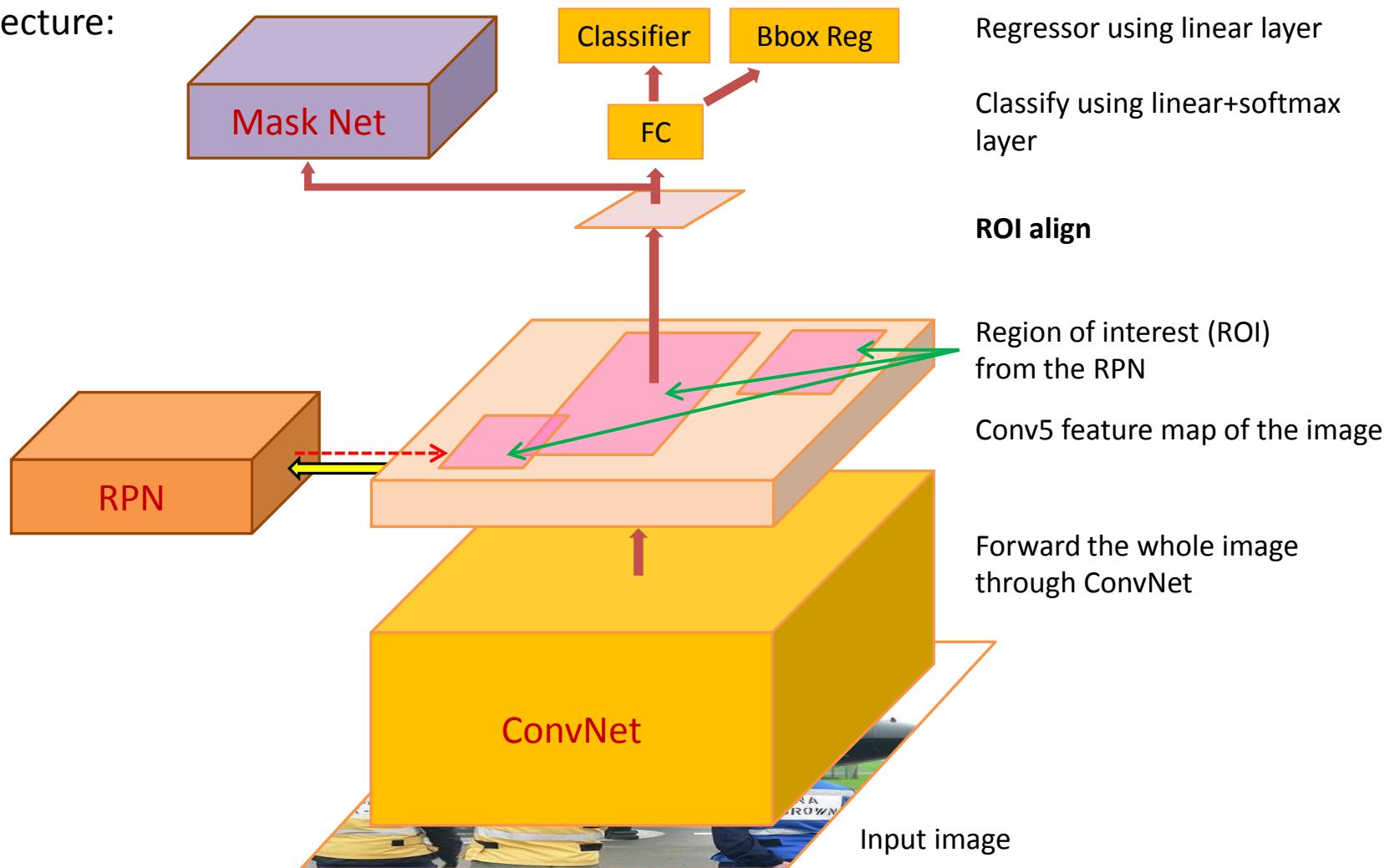
Faster R-CNN to Mask R-CNN

Recall the Faster R-CNN architecture:



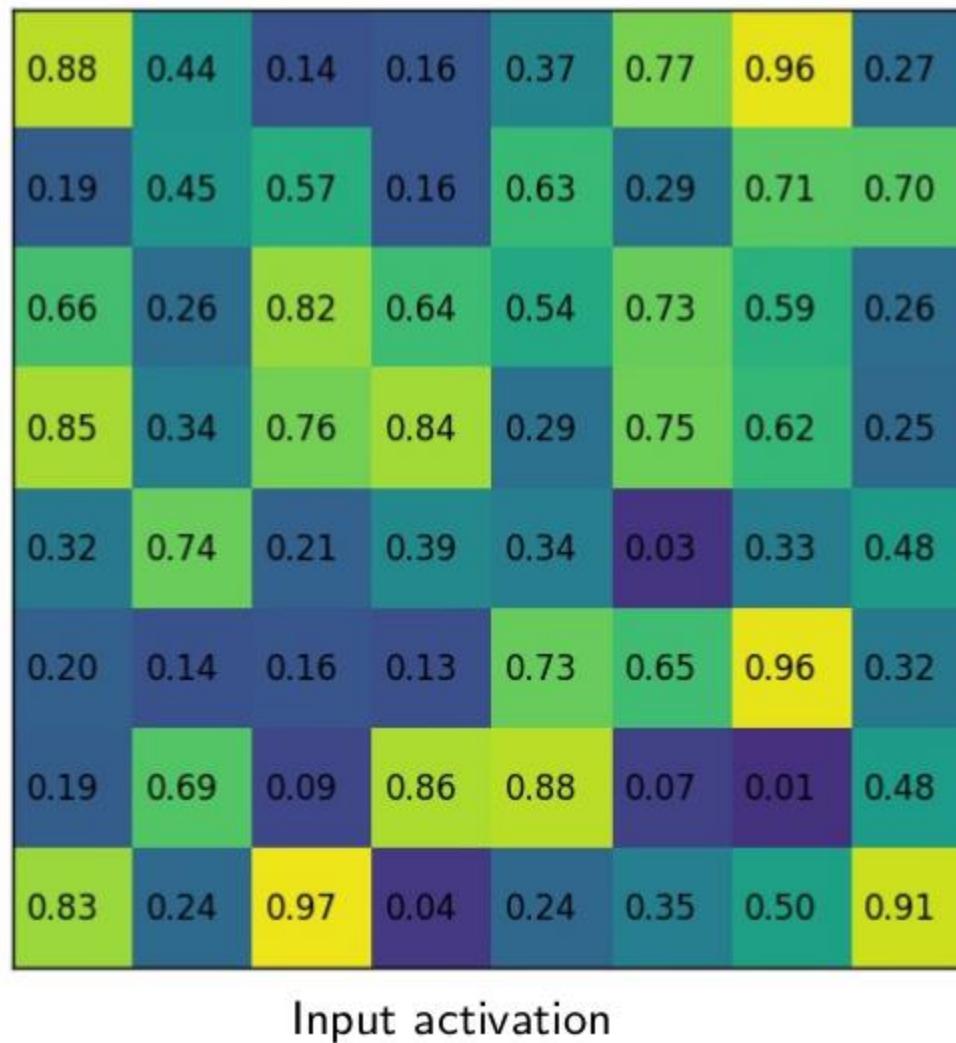
Faster R-CNN to Mask R-CNN

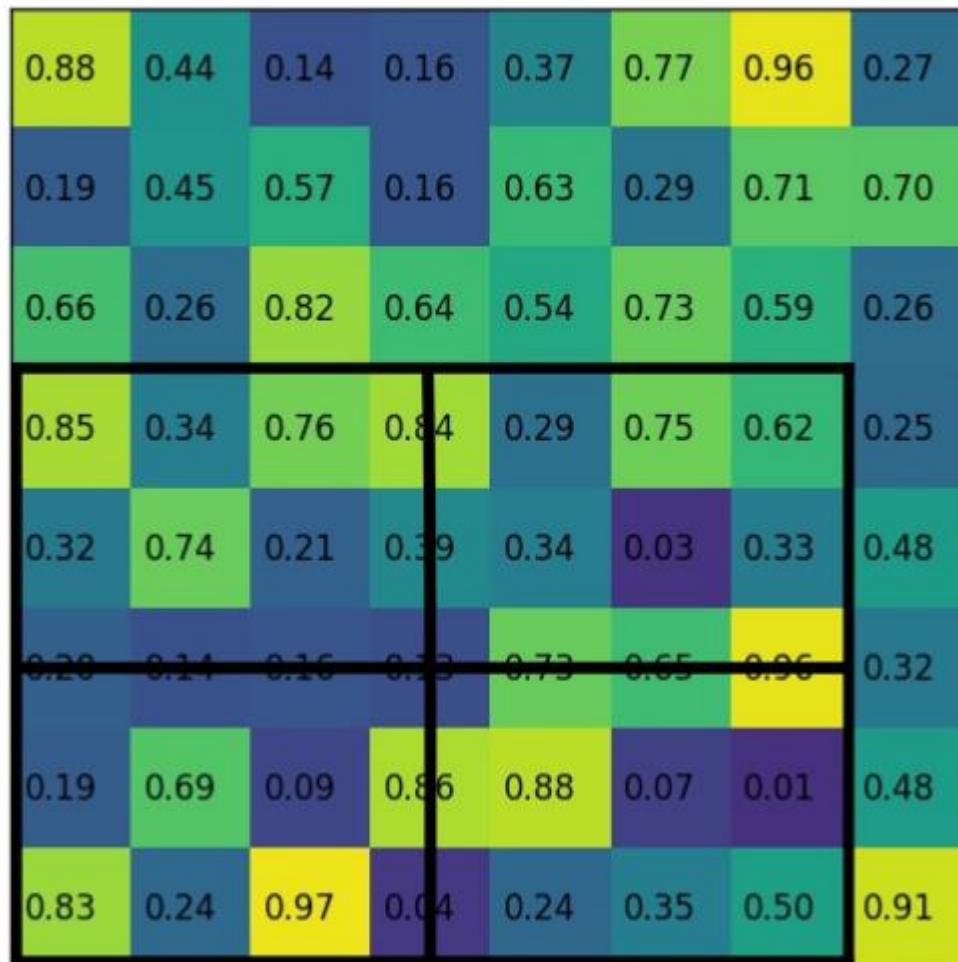
This is the mask R-CNN architecture:



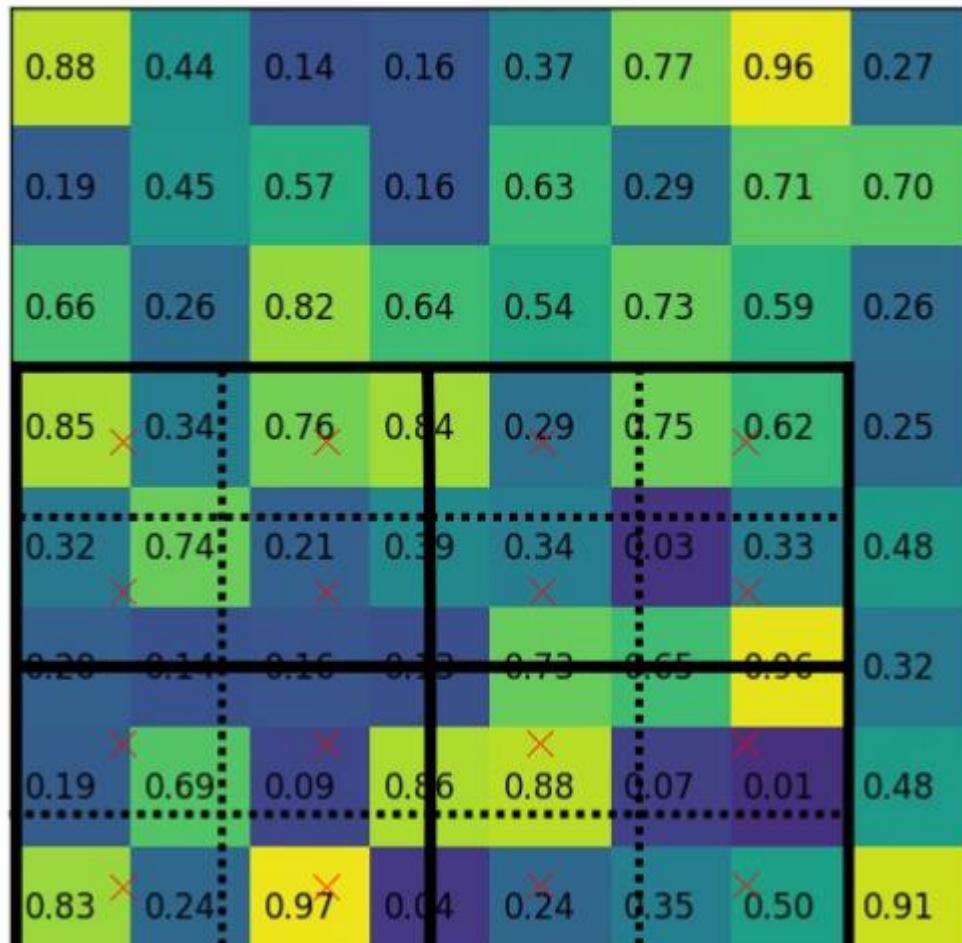


ROI Align

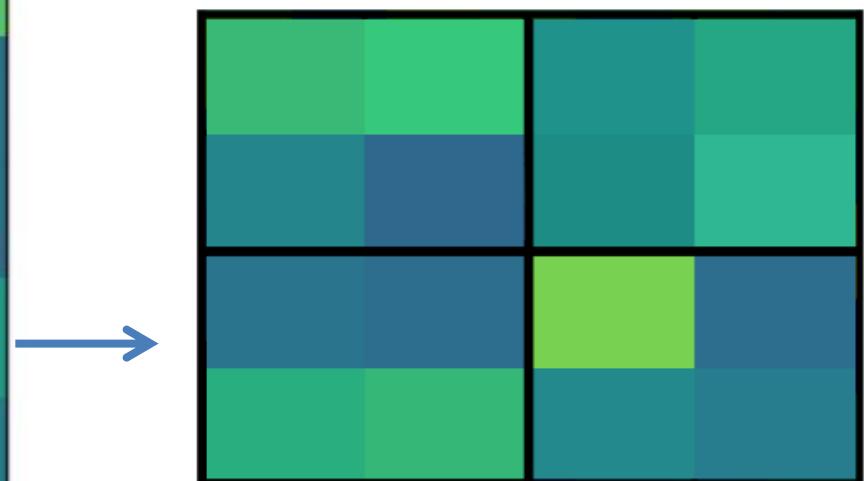




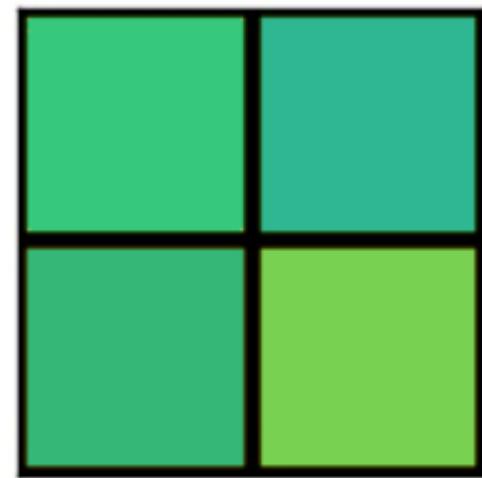
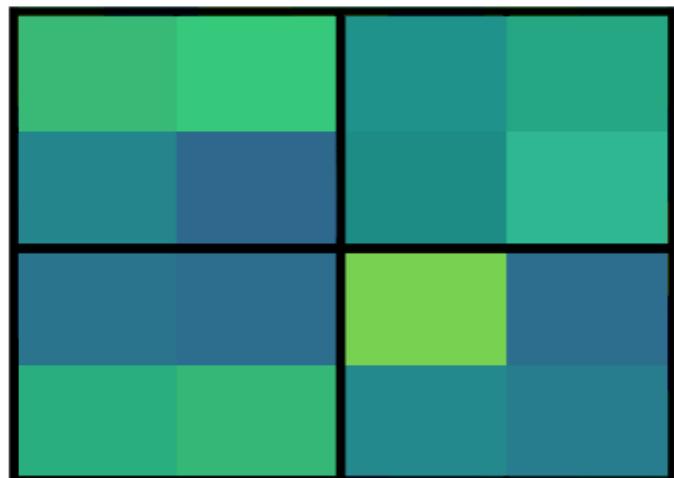
Region projection and pooling sections



Sampling locations



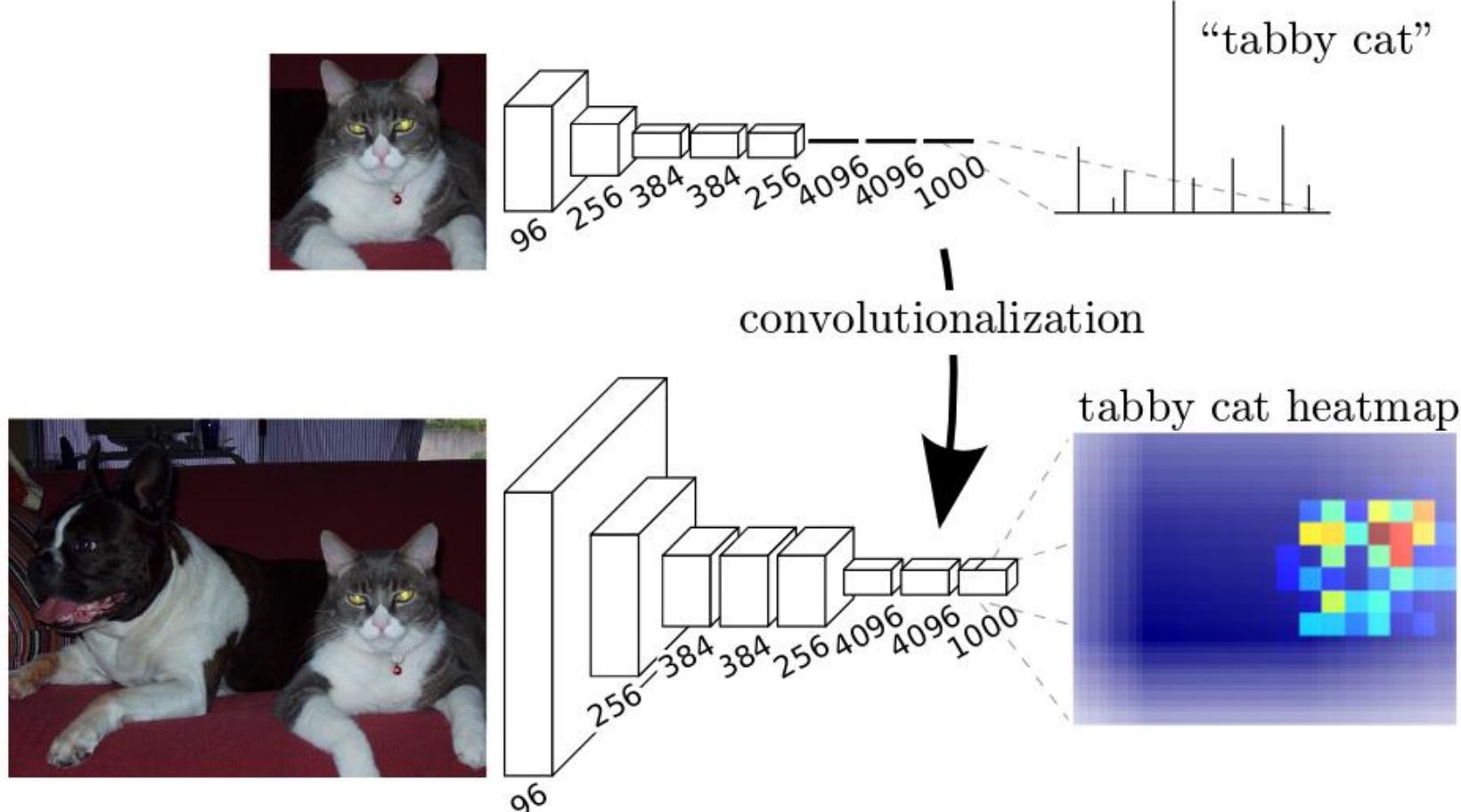
Bilinear interpolated values



Mask Network in Mask R-CNN (1/2)



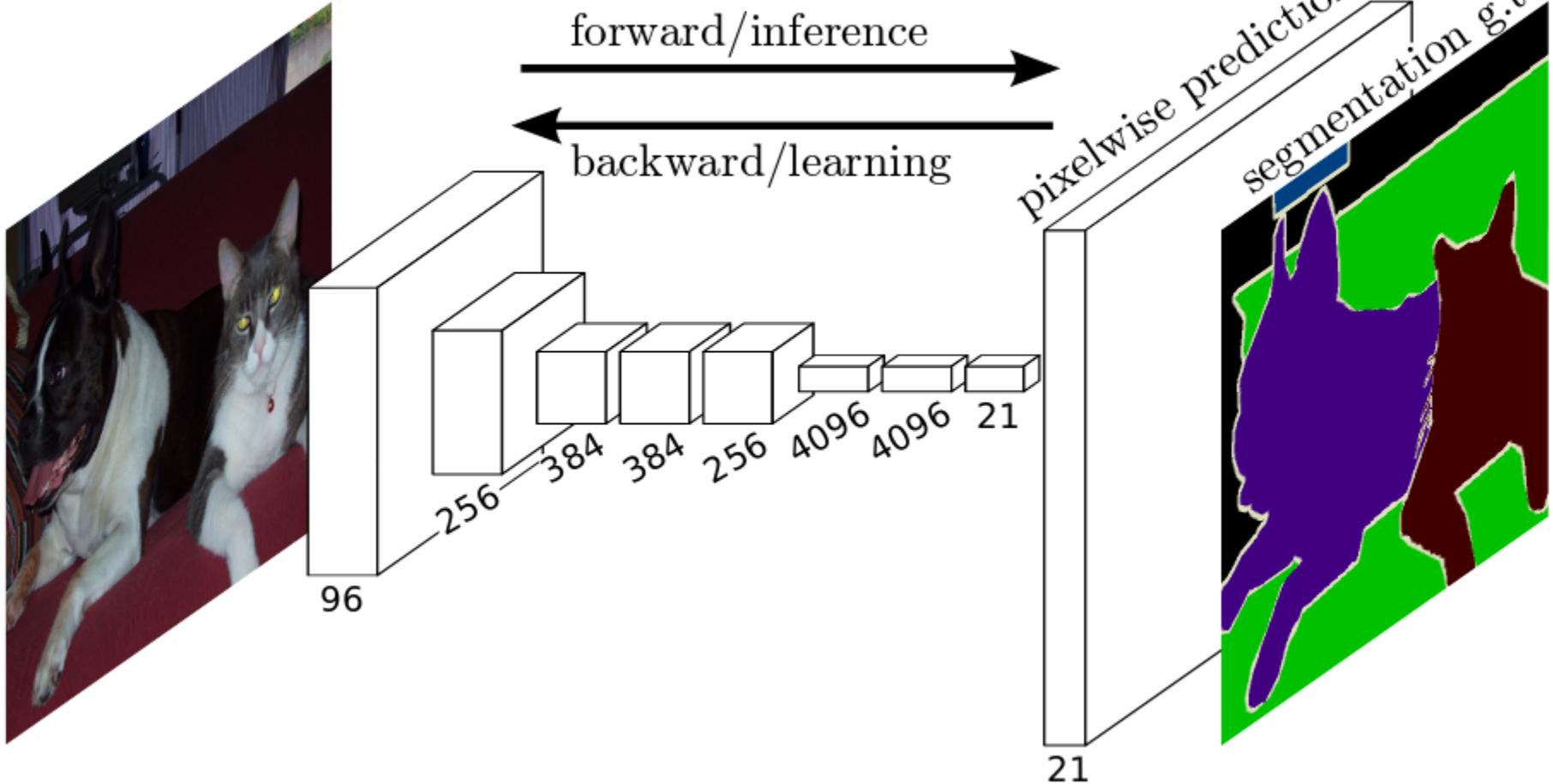
- ▶ What happened if we change the FC layer to Convolution layer?
 - ▶ The resulting maps are equivalent to the evaluation on particular patch



J. Long, E. Shelhamer, and T. Darrell. "Fully convolutional networks for semantic segmentation." CVPR. 2015.

Mask Network in Mask R-CNN (2/2)

- In that case, we can train our network to do this kind of task.

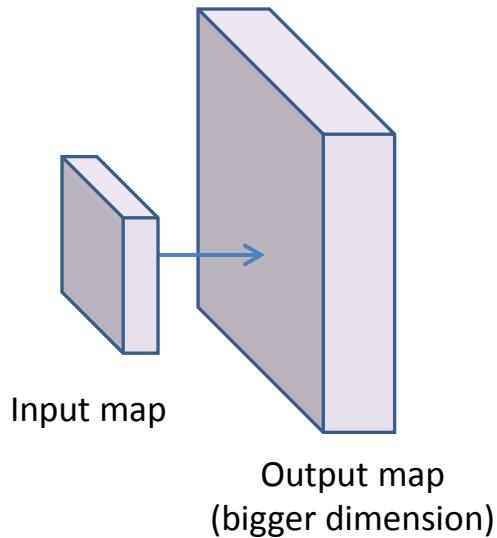


J. Long, E. Shelhamer, and T. Darrell. "Fully convolutional networks for semantic segmentation." CVPR. 2015.



The Up-sampling Operation

- ▶ To obtains feature map with bigger dimension



- ▶ How to do?
 - ▶ Convolution can be viewed as matrix operation
 - ▶ Using the transposed “kernel matrix” we can obtain a bigger output map

Convolution as Matrix Operation (1/5)



- ## ► Illustration of an “ordinary convolution”:

4	5	8	7
1	8	8	8
3	6	6	4
6	5	7	8

*

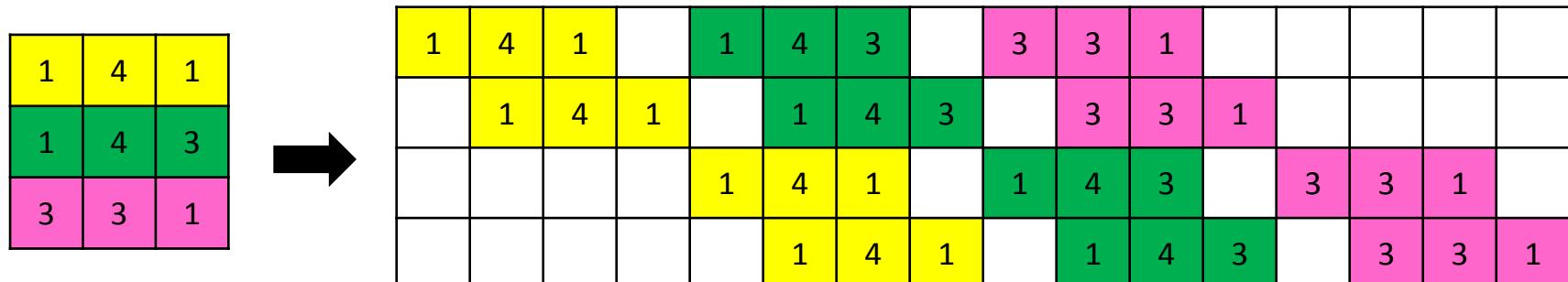
1	4	1
1	4	3
3	3	1

=

122	148
126	134

Input kernel output

- We can represent the kernel as a “convolution matrix”:



Convolution as Matrix Operation (2/5)

- And the input as its serialized form:

4	5	8	7
1	8	8	8
3	6	6	4
6	5	7	8



4
5
8
7
1
8
8
8
3
6
6
4
6
5
7
8

Convolution as Matrix Operation (3/5)

- ▶ Perform the convolution as matrix multiplication

- ▶ The matrix multiplication produce same “values”

1	4	1		1	4	3		3	3	1						
	1	4	1		1	4	3		3	3	1					
				1	4	1		1	4	3		3	3	1		
					1	4	1		1	4	3		3	3	1	

$C_{4 \times 16}$

$$\begin{array}{c}
 \begin{array}{c}
 \begin{array}{c}
 4 \\
 5 \\
 8 \\
 7 \\
 1 \\
 8 \\
 8 \\
 8 \\
 3 \\
 6 \\
 6 \\
 4 \\
 6 \\
 5 \\
 7 \\
 8
 \end{array} \\
 \times \\
 = \\
 \begin{array}{c}
 122 \\
 148 \\
 126 \\
 134
 \end{array}
 \end{array}
 \end{array}$$

$I_{16 \times 1}$

$O_{4 \times 1}$

Convolution as Matrix Operation (4/5)

1	4	1		1	4	3		3	3	1						
	1	4	1		1	4	3		3	3	1					
				1	4	1		1	4	3		3	3	1		
					1	4	1		1	4	3		3	3	1	

 $C_{4 \times 16}$

 $=$

122
148
126
134

 $O_{4 \times 1}$

4	5	8	7
1	8	8	8
3	6	6	4
6	5	7	8

Input



1	4	1
1	4	3
3	3	1

kernel



122	148
126	134

output

4
5
8
7
1
8
8
8

3
6
6
4

 $I_{16 \times 1}$



Convolution as Matrix Operation (5/5)

1	4	1		1	4	3		3	3	1						
	1	4	1		1	4	3		3	3	1					
				1	4	1		1	4	3		3	3	1		
					1	4	1		1	4	3		3	3	1	
						1	4		1	4	3		3	3	1	

$C_{4 \times 16}$

4																
5																
8																
7																
1																
8																
8																
8																

$O_{4 \times 1}$

4	5	8	7
1	8	8	8
3	6	6	4
6	5	7	8

Input

1	4	1
1	4	3
3	3	1

*

kernel

122	148
126	134

output

4																
5																
8																
7																
1																
8																
8																
8																
3																
6																
6																
4																
6																
5																
7																
8																

$I_{16 \times 1}$



The Transposed Convolution (1/2)

- Recall the convolution as matrix operation case:

$$\begin{matrix} C & \times & I \\ 4 \times 16 & & 16 \times 1 \end{matrix} = \begin{matrix} O \\ 4 \times 1 \end{matrix}$$

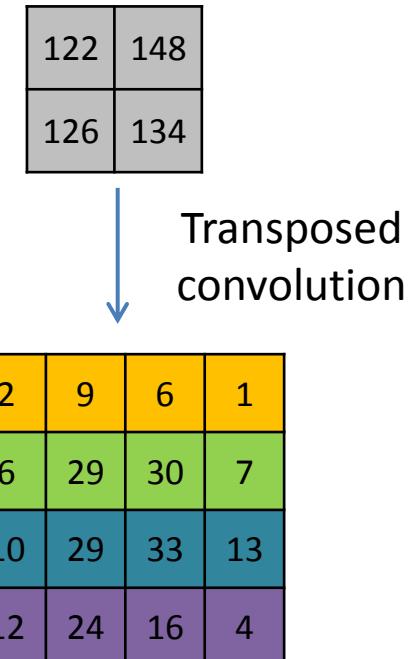
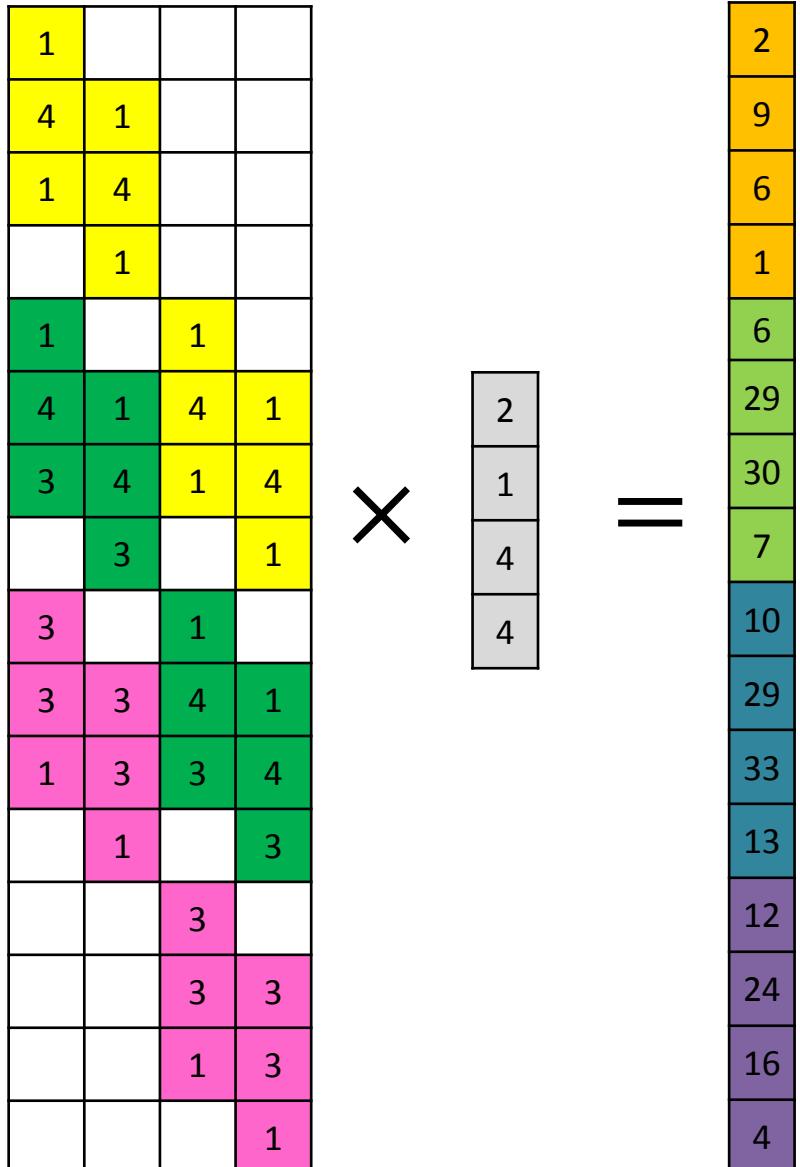
- If we have convolution result O ,

- we can “obtains” \hat{I} , which may represent the original input image
- Simplest way: using the transposed convolution kernel \hat{C}

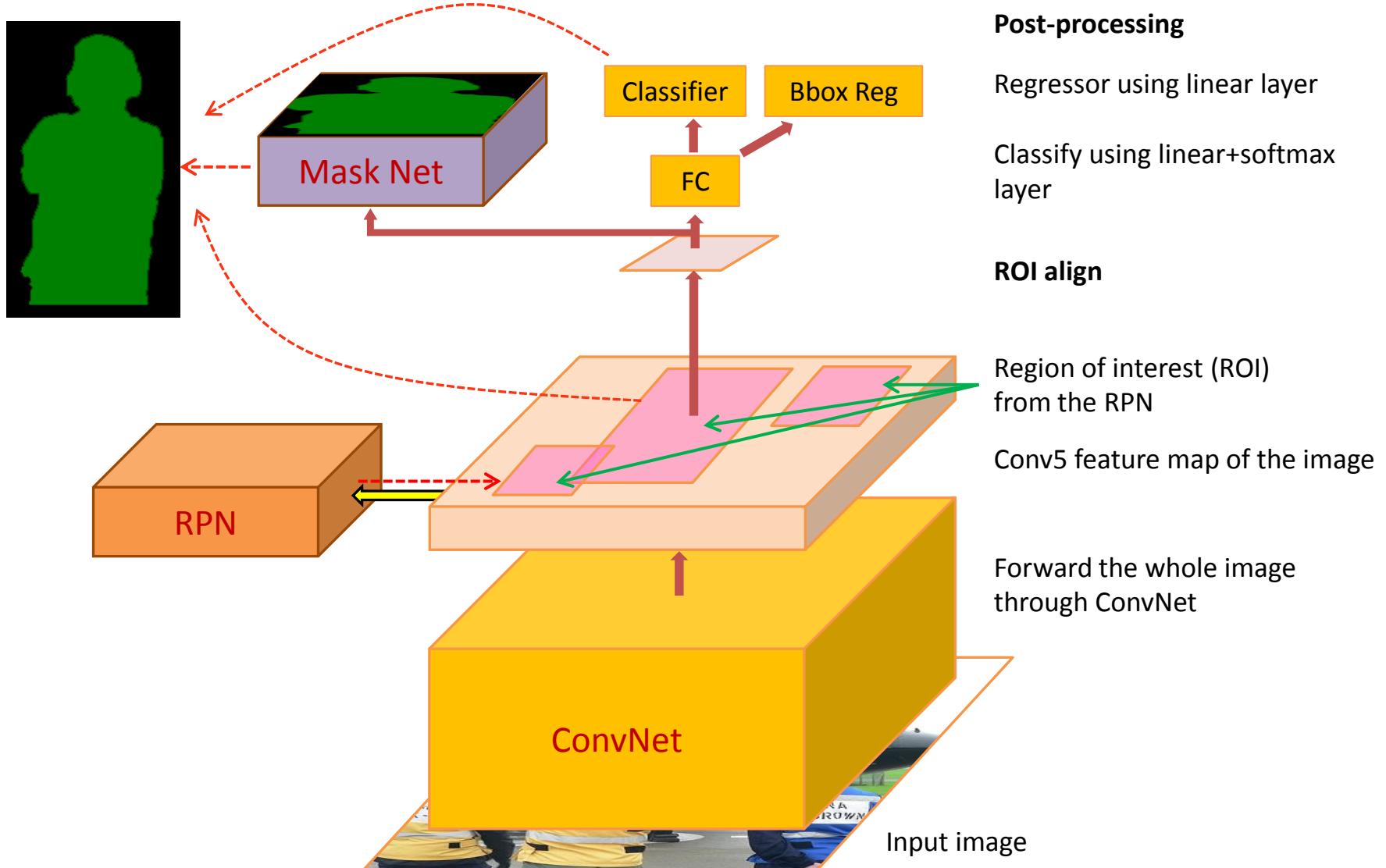
$$\begin{matrix} \hat{C} & \times & O \\ 16 \times 4 & & 4 \times 1 \end{matrix} = \begin{matrix} \hat{I} \\ 16 \times 1 \end{matrix}$$



The Transposed Convolution (2/2)



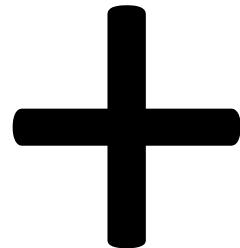
Mask R-CNN





Lets try on Colab!

co



Download the Mask R-CNN Model



```
SAVE_FOLDER = 'hs1/'

# What model to download.
MODEL_NAME = 'mask_rcnn_resnet101_atrous_coco_2018_01_28'
MODEL_FILE = MODEL_NAME + '.tar.gz'
DOWNLOAD_BASE = 'http://download.tensorflow.org/models/object_detection/'

# Path to frozen detection graph.
# This is the actual model that is used for the object detection.
PATH_TO_CKPT = SAVE_FOLDER+MODEL_NAME + '/frozen_inference_graph.pb'

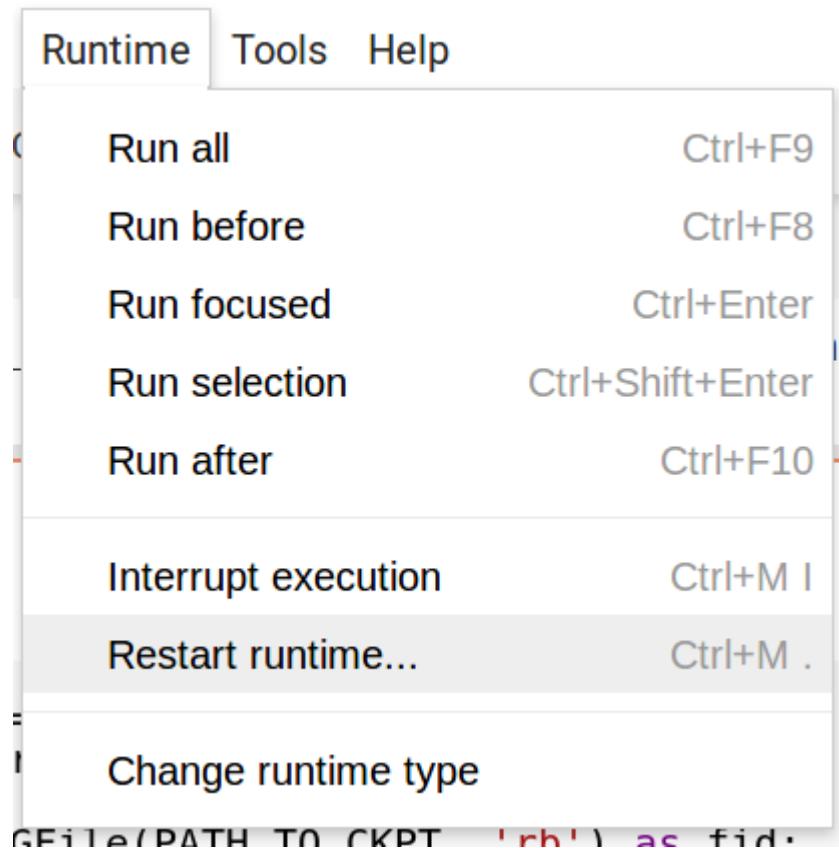
# List of the strings that is used to add correct label for each box.
PATH_TO_LABELS = os.path.join('hs1/models/research/object_detection/data/',
                             'mscoco_label_map.pbtxt')

NUM_CLASSES = 90
```

Restart the Execution Kernel (Optional)



- ▶ If you do it,
 - ▶ It will clear all variables
 - ▶ You need to run all the python fields again





Or, Just Run the Required Fields

▼ Download and unzip the pre-trained model

+ form trash comment link

```
▶ opener = urllib.request.URLopener()
opener.retrieve(DOWNLOAD_BASE + MODEL_FILE, SAVE_FOLDER+MODEL_FILE)
tar_file = tarfile.open(SAVE_FOLDER+MODEL_FILE)
for file in tar_file.getmembers():
    file_name = os.path.basename(file.name)
    if 'frozen_inference_graph.pb' in file_name:
        tar_file.extract(file, SAVE_FOLDER)
```

▼ Load the model

+ form trash comment link

```
▶ detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')
```



Detection Result with Mask



`img_url: " http://host.robots.ox.ac.uk/pascal/VOC/voc2010/segeexamples/images/2 "`

`size: 9`



0

50

100

150

200

250

300

350

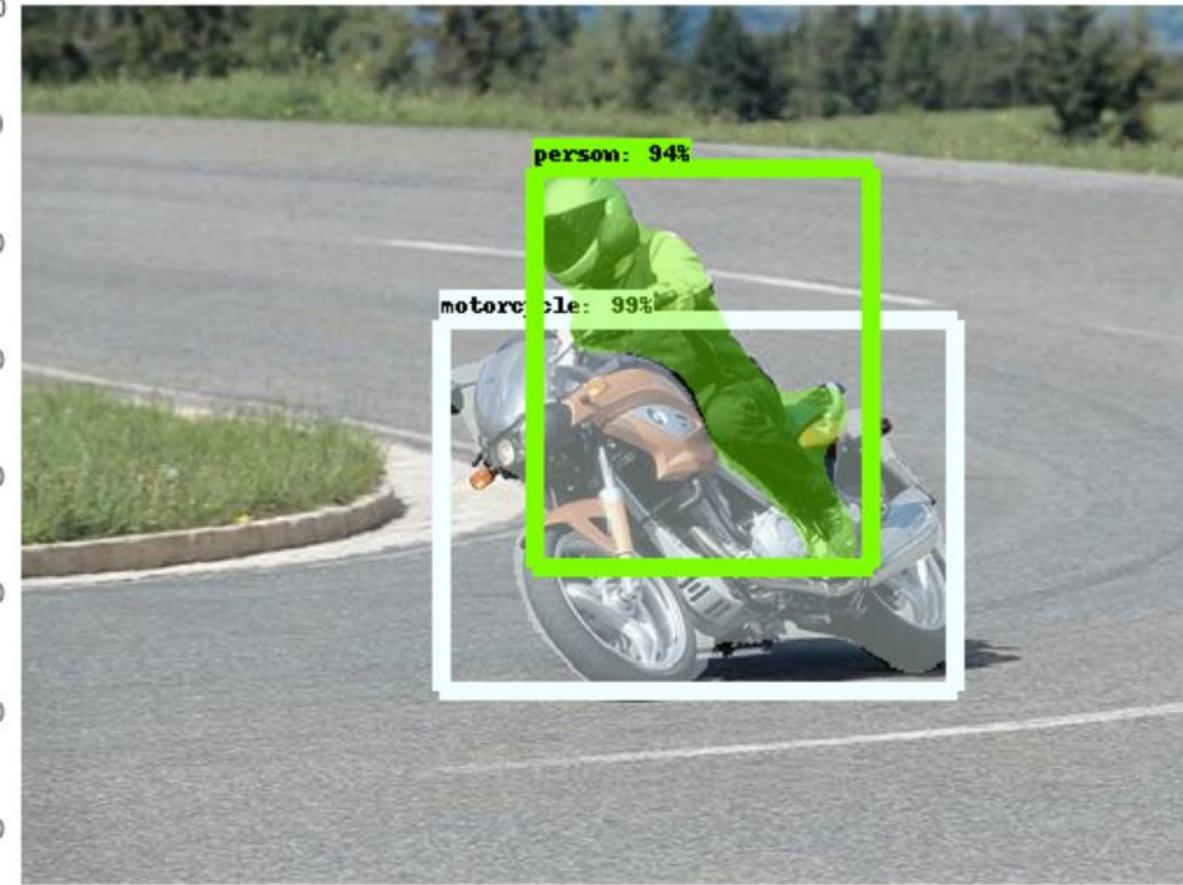
0

100

200

300

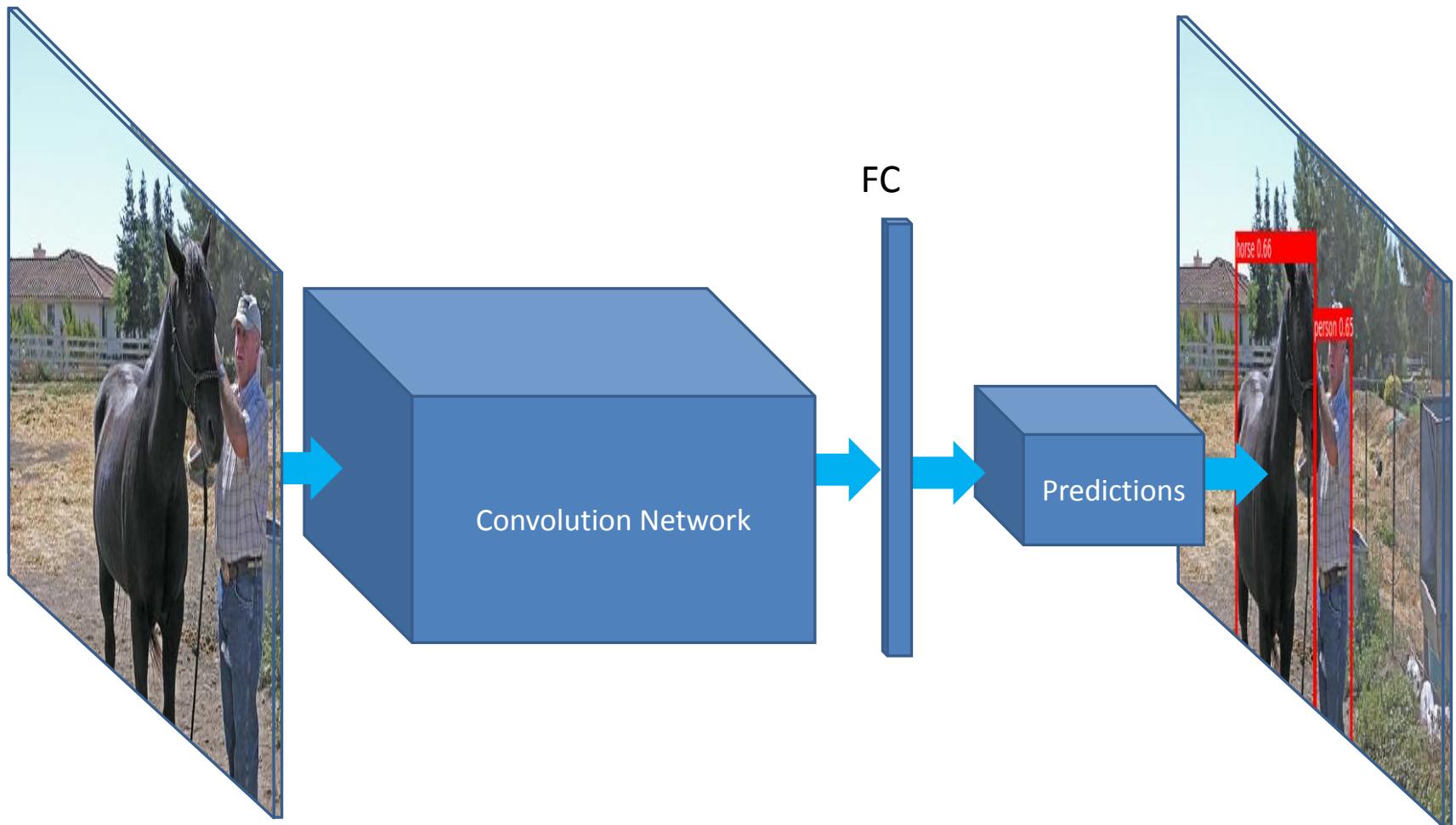
400





You Only Look Once (YOLO)

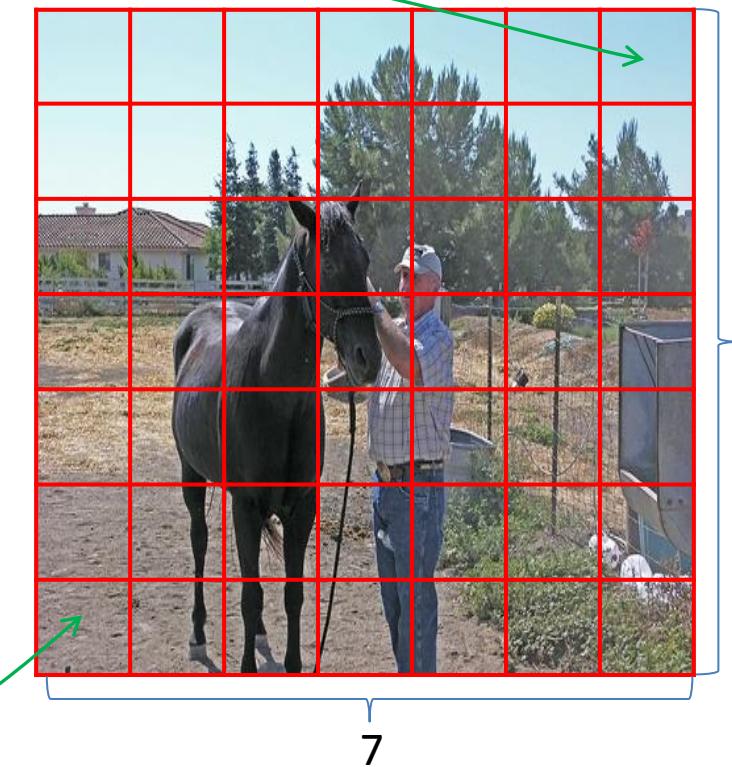
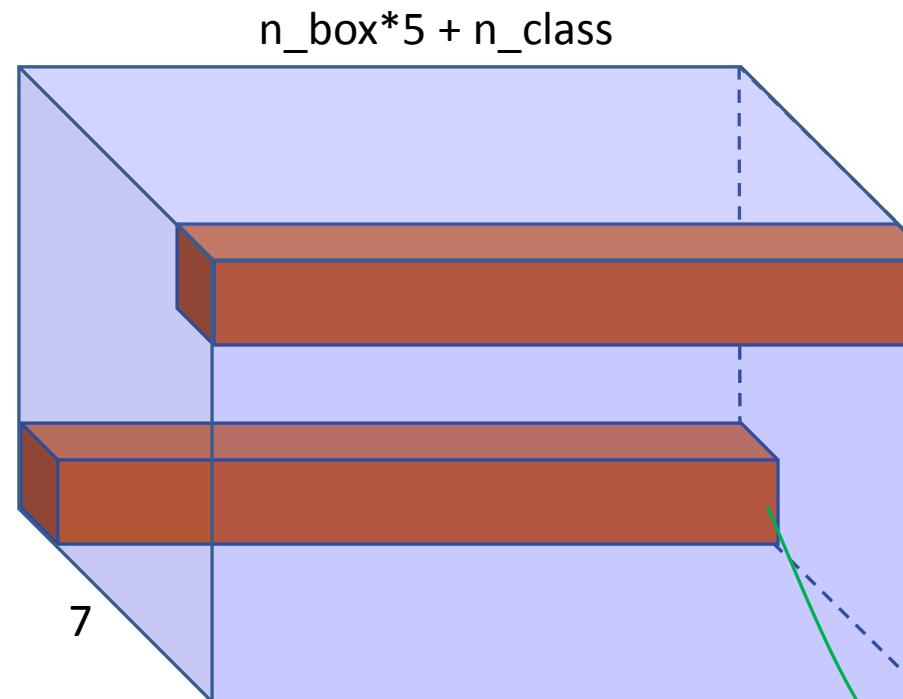
- ▶ Single stage detector





Prediction Layer in YOLO

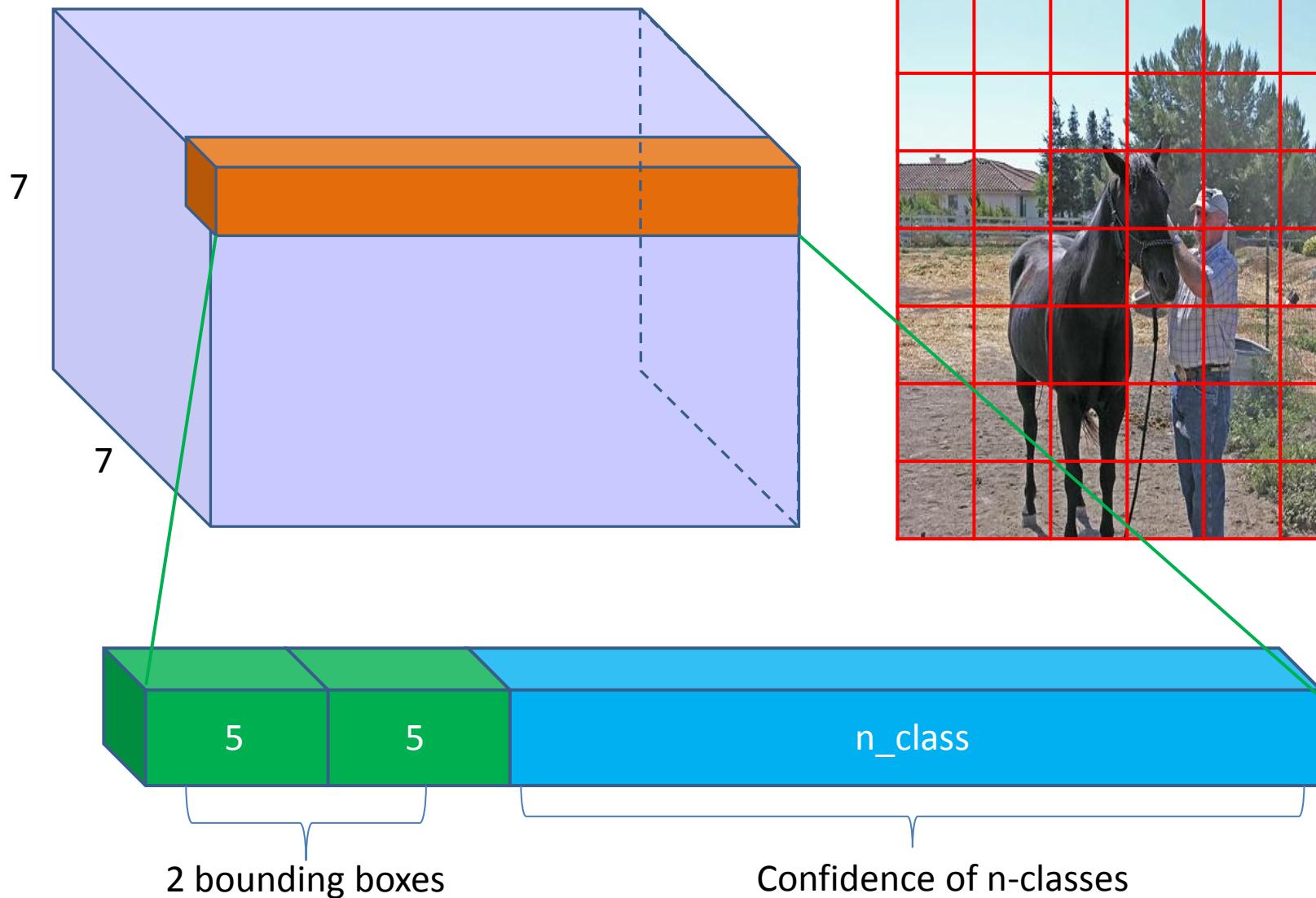
Each block is responsible for predicting the object in its corresponding image region



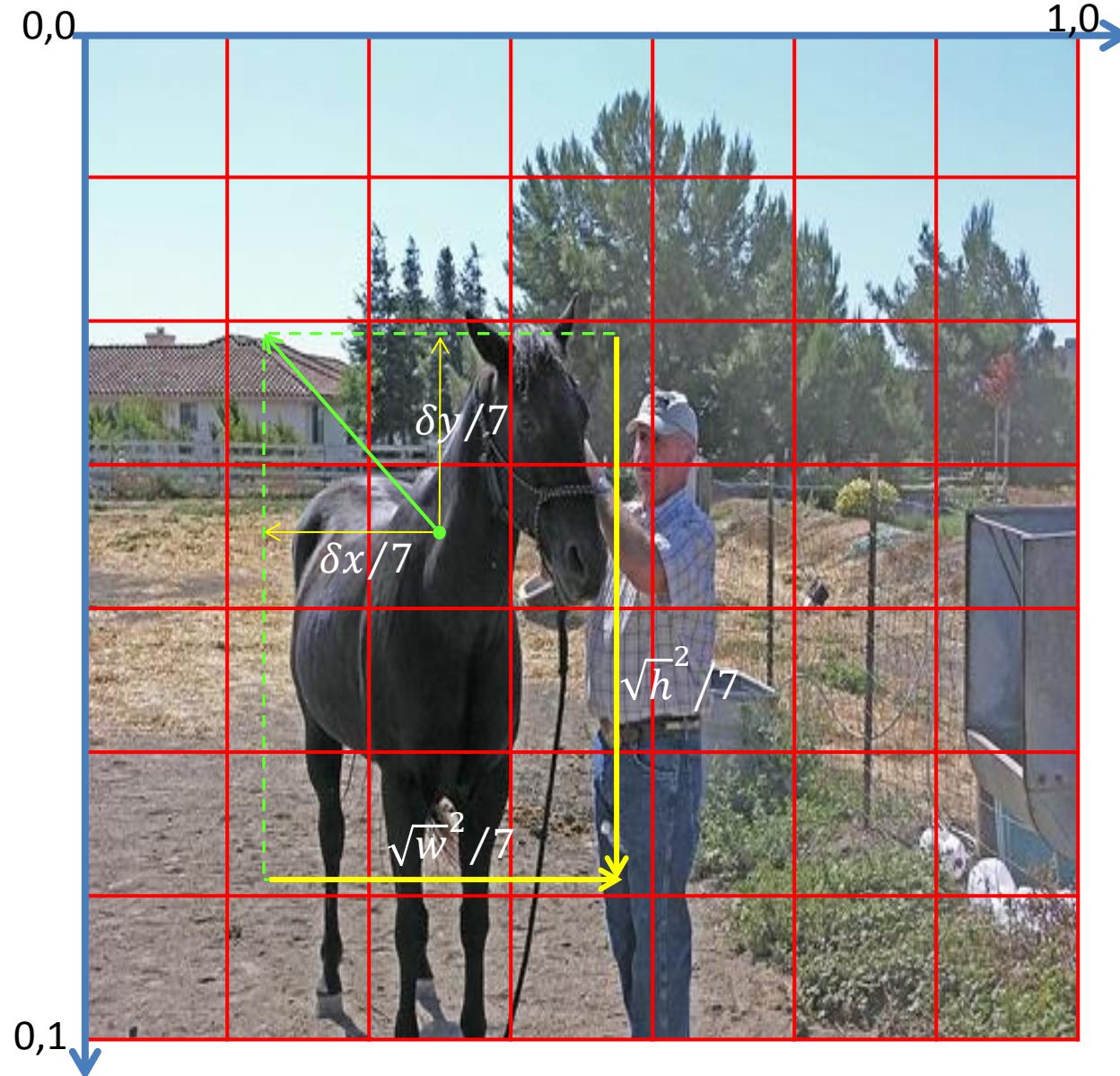
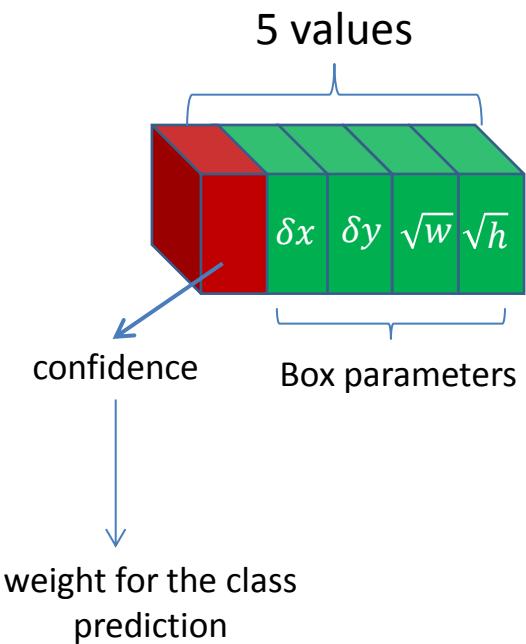


What Elements are Predicted?

$n_{\text{box}} * 5 + n_{\text{class}}$

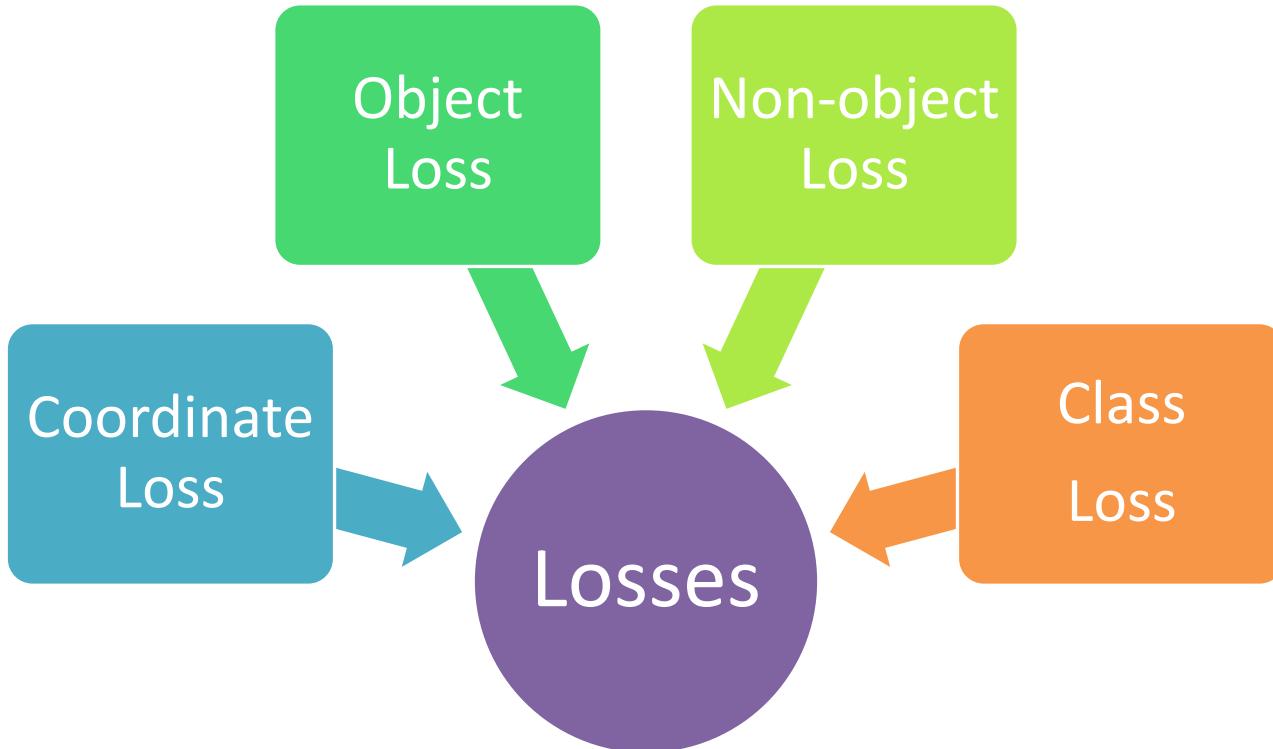


More Details on Bounding Box Prediction





Training Objectives

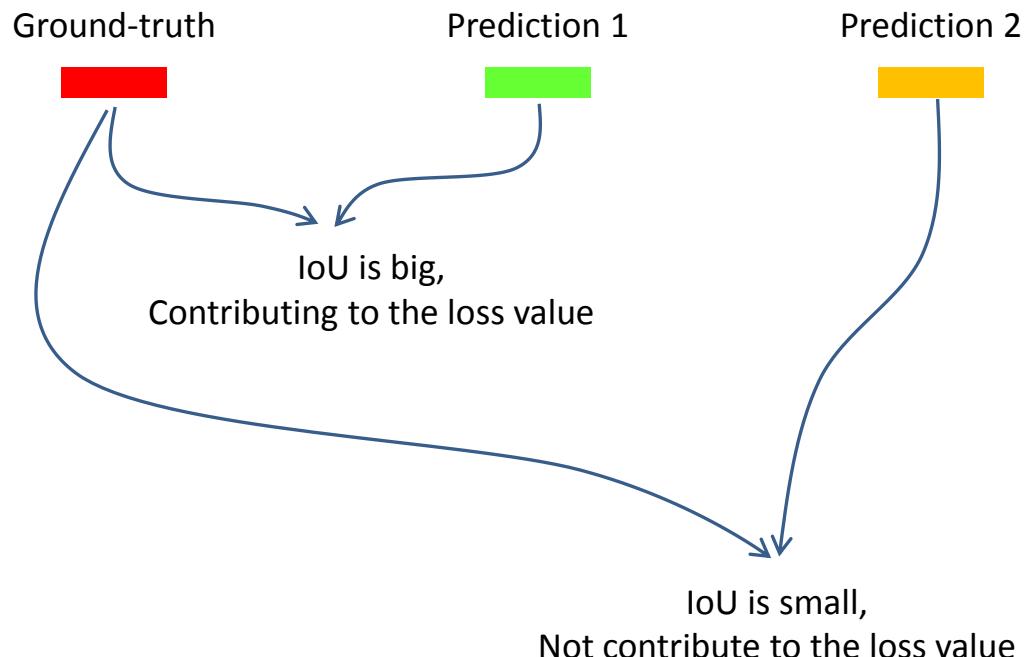
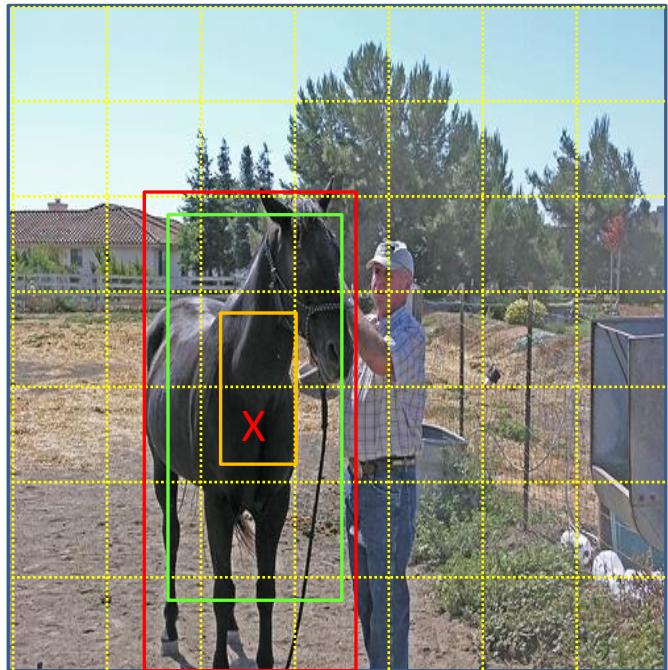


Coordinate Loss

- ▶ When object appear in the cell,
 - ▶ And prediction box is responsible (IoU is biggest, for the predicted cell)

$$L_{coordinate} = \sum_{c \in \{c | object \text{ in } c \text{ and } IoU > T\}} \lambda_B (B_{p,c} - B_{g,c})^2$$

- ▶ Illustration for the marked cell (center of horse)

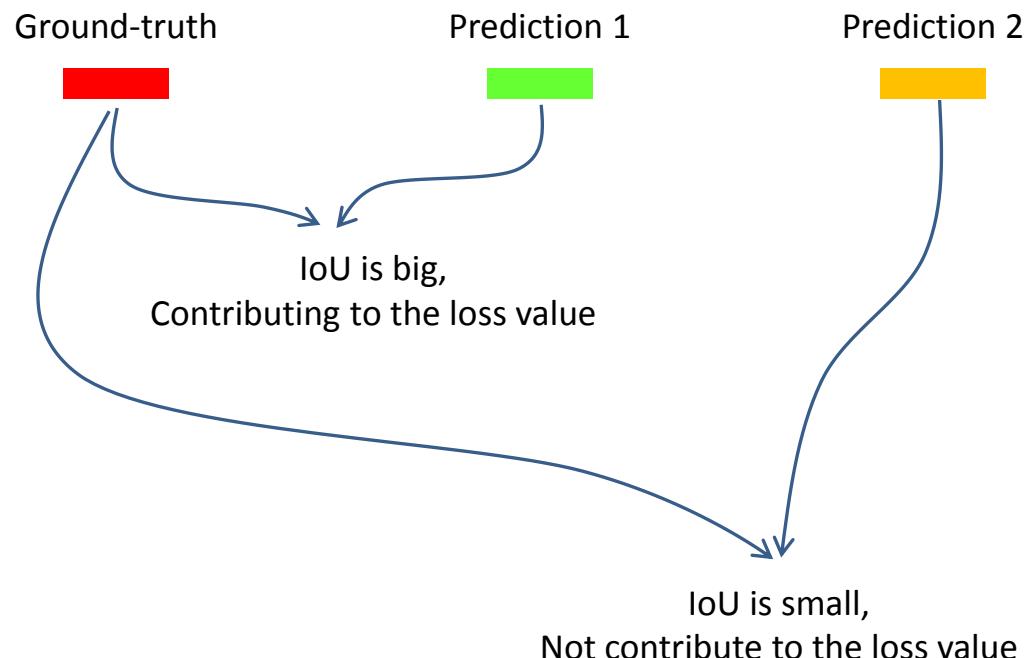
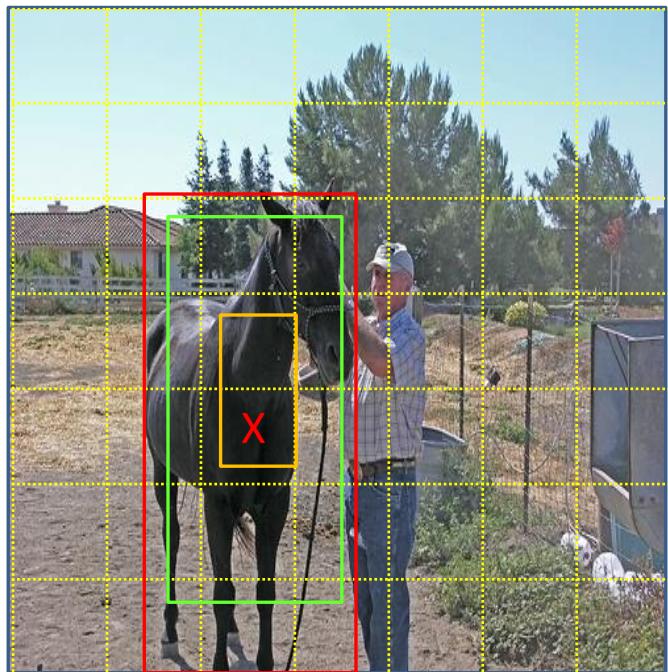


Objectness Loss

- When object appear in the cell and box prediction is responsible,
 - Objectness value should be same with ground-truth

$$L_{object} = \sum_{c \in \{c | object \text{ in } c \text{ and } IoU > T\}} \lambda_o (o_{p,c} - o_{g,c})^2$$

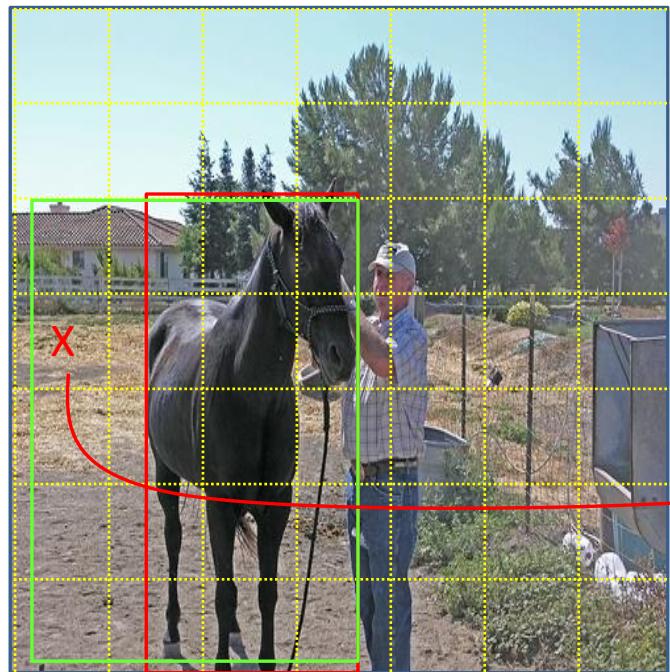
Value close to 1 → smaller loss



Non-Object Loss

- ▶ Object not appear in the cell but predicted box have big IoU to groundtruth,
 - ▶ The predicted objectness value should close to zero

$$L_{non_object} = \sum_{c \in \{c | object \text{ not in } c\}} \lambda_N (O_{p,c})^2$$



Ground-truth
 Prediction 1

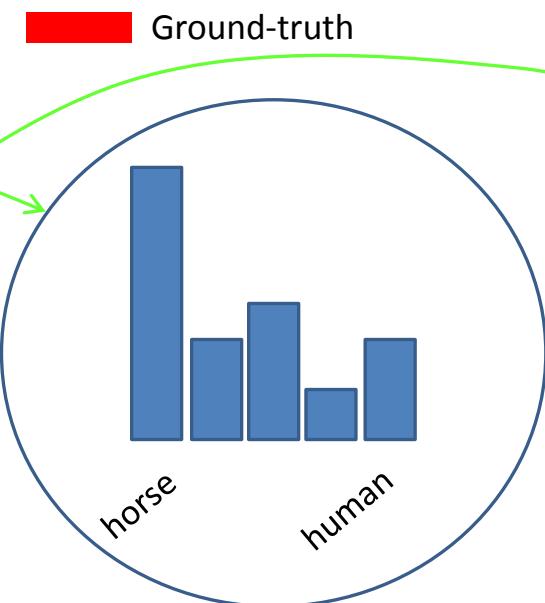
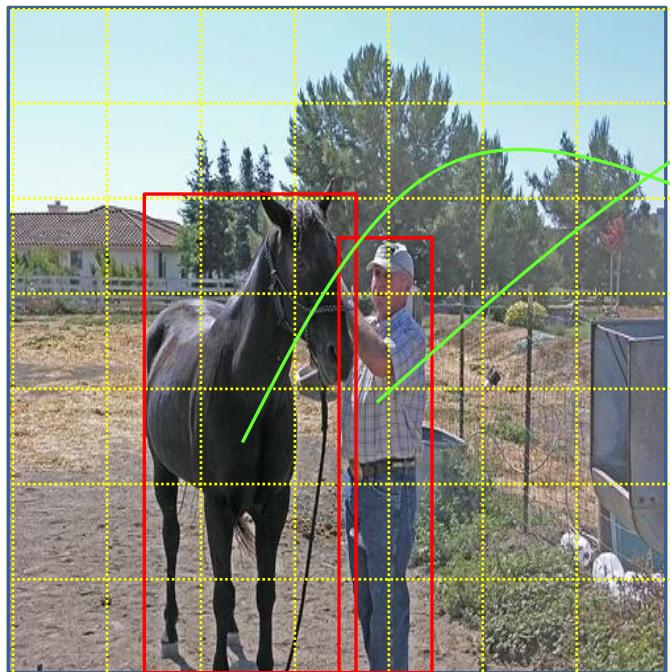
Objectness value in this cell
should be close to zero

Classification Loss

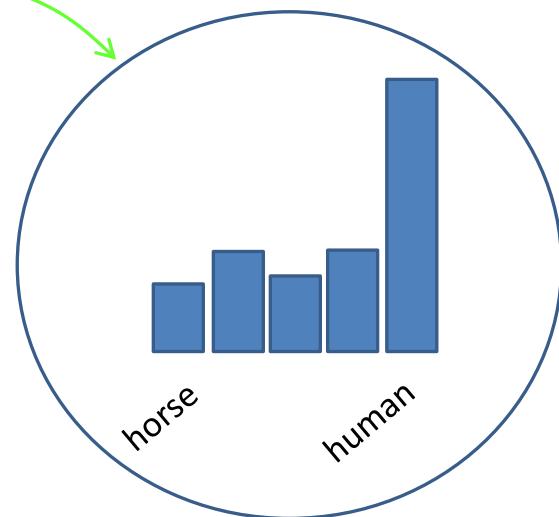


- The predicted class should be similar with ground-truth
 - When object appear in the cell

$$L_{classification} = \sum_{c \in \{c | object\;in\;c\}} \lambda_p (P_{p,c} - P_{g,c})^2$$



Predicted value of horse
should be the biggest



Predicted value of human
should be the biggest

Post-Processing

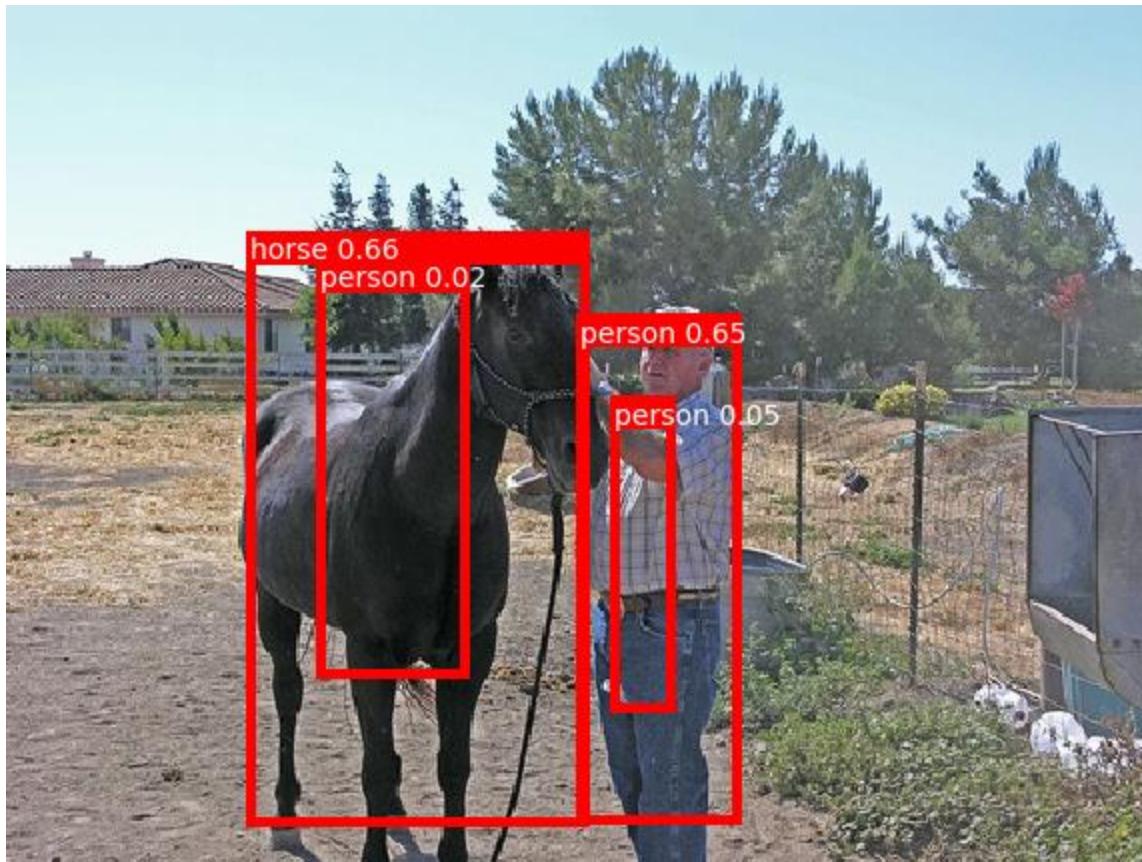


► Original detection



Post-Processing

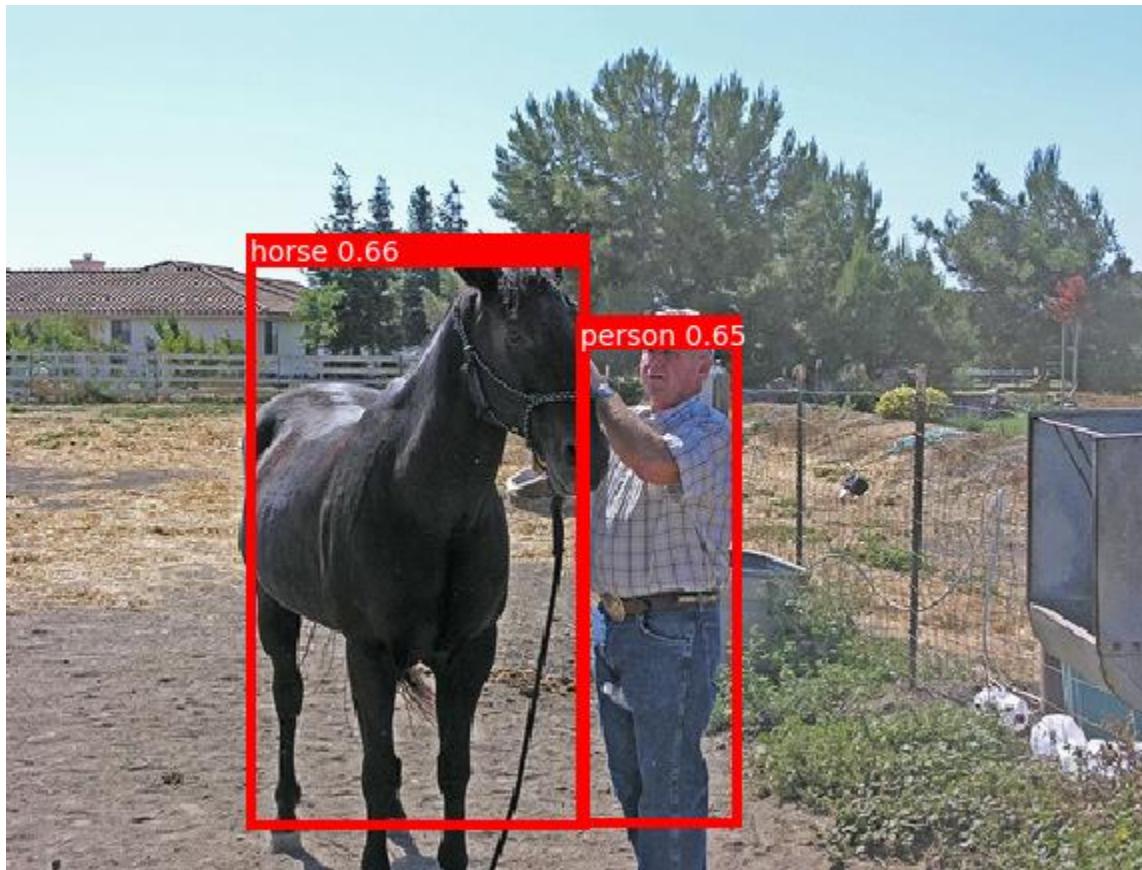
- ▶ Ignore detection results with small confidence



0.02 and 0.05 are too small, but lets keep them for the shake of this explanation

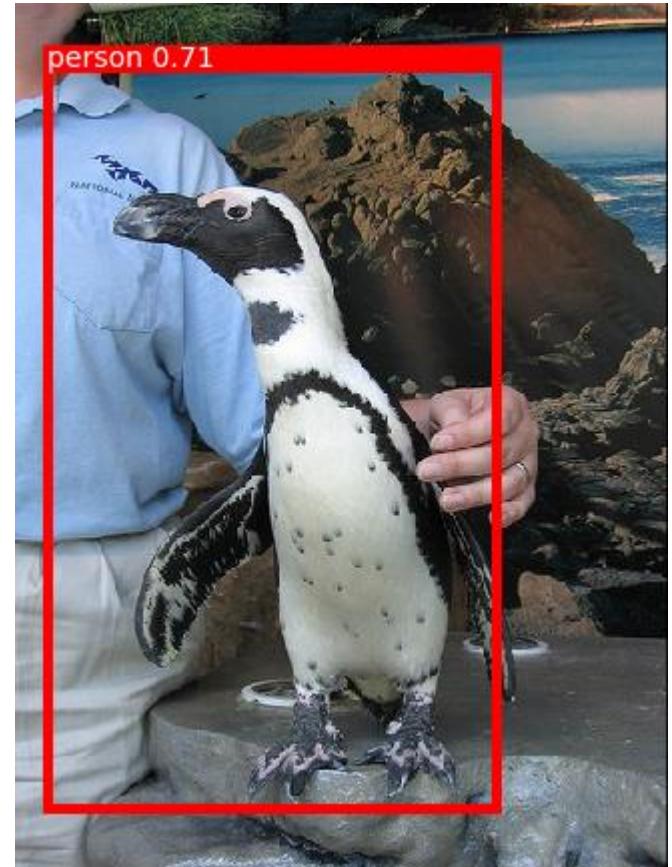
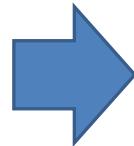
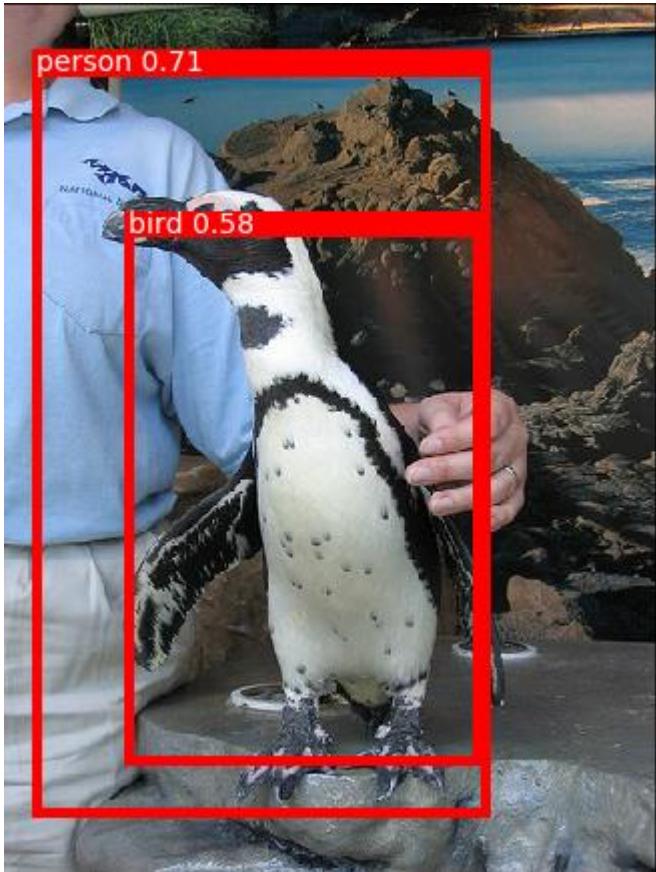
Post-Processing

- Remove overlapped detection based on IoU



IoU: Intersection over Union

Drawback of IoU Filtering



IoU = 0.5499

Drawback of IoU Filtering

person 0.84

person 0.73

person 0.80

person 0.84

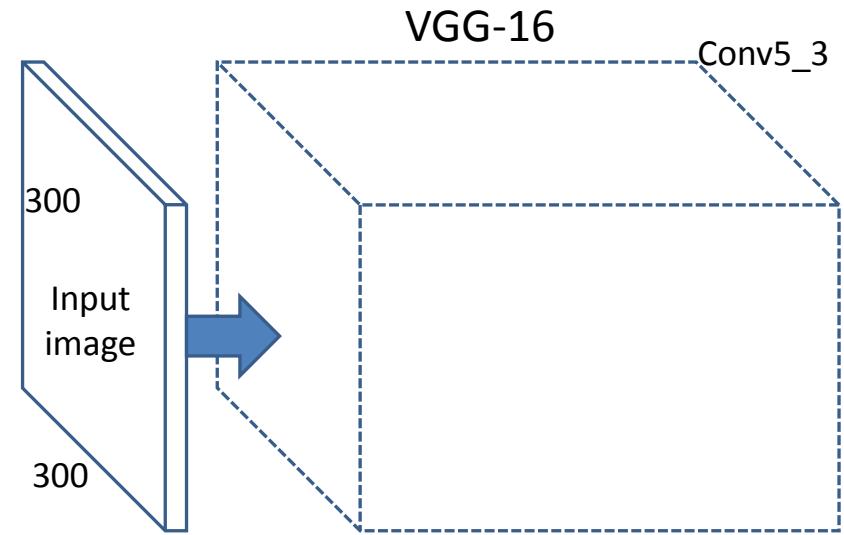
person 0.80



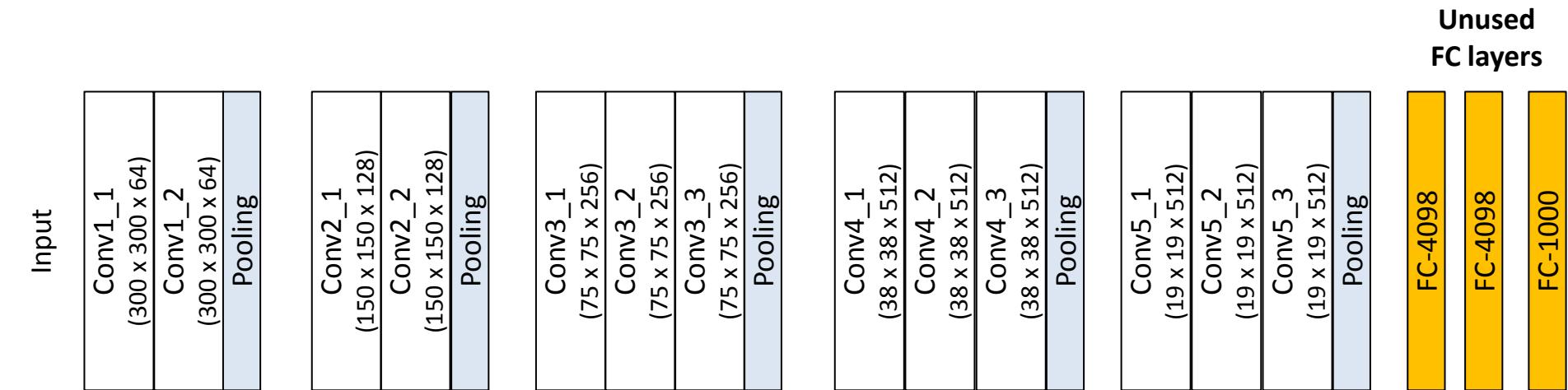
$$\text{IoU} = 0.5774$$



The Single Shot Detector (SSD)

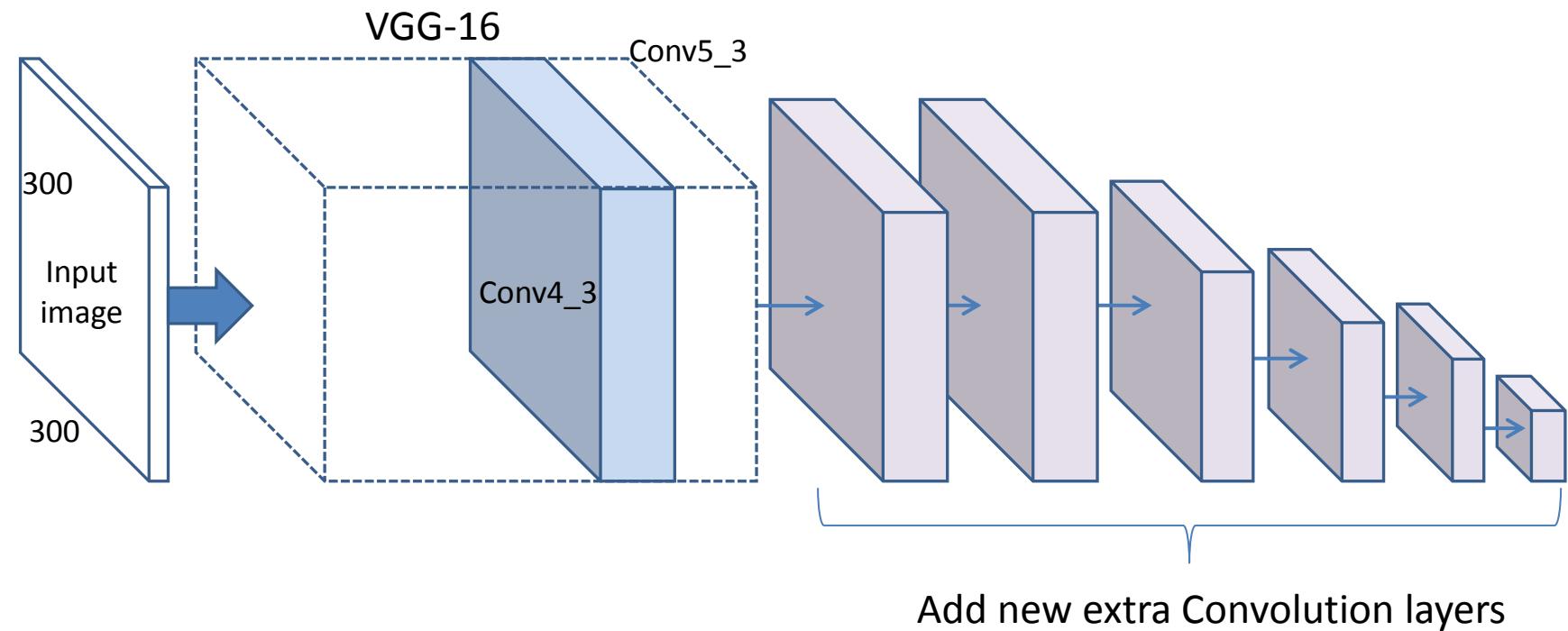


SSD is generalization of the YOLO type detector



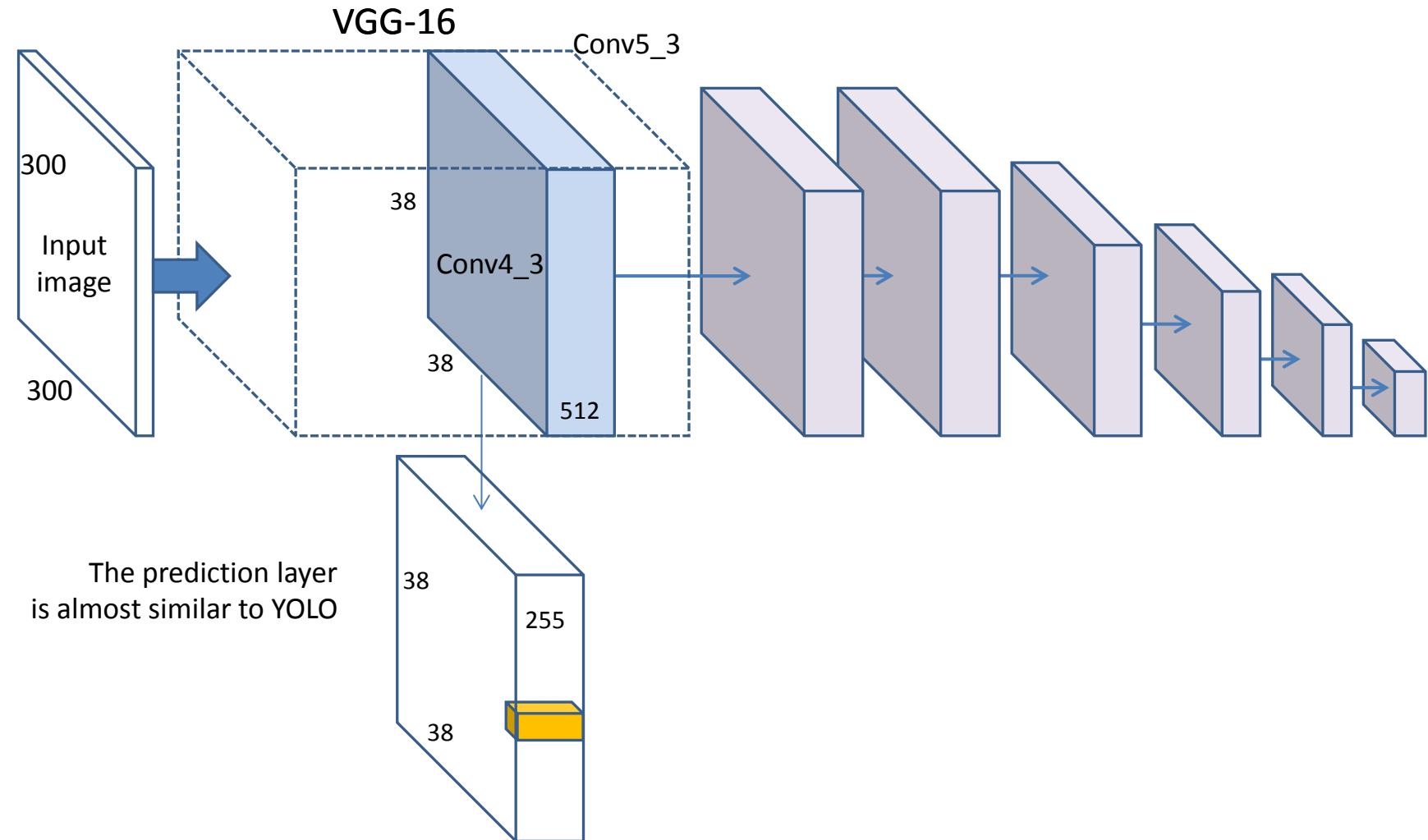


The Single Shot Detector (SSD)



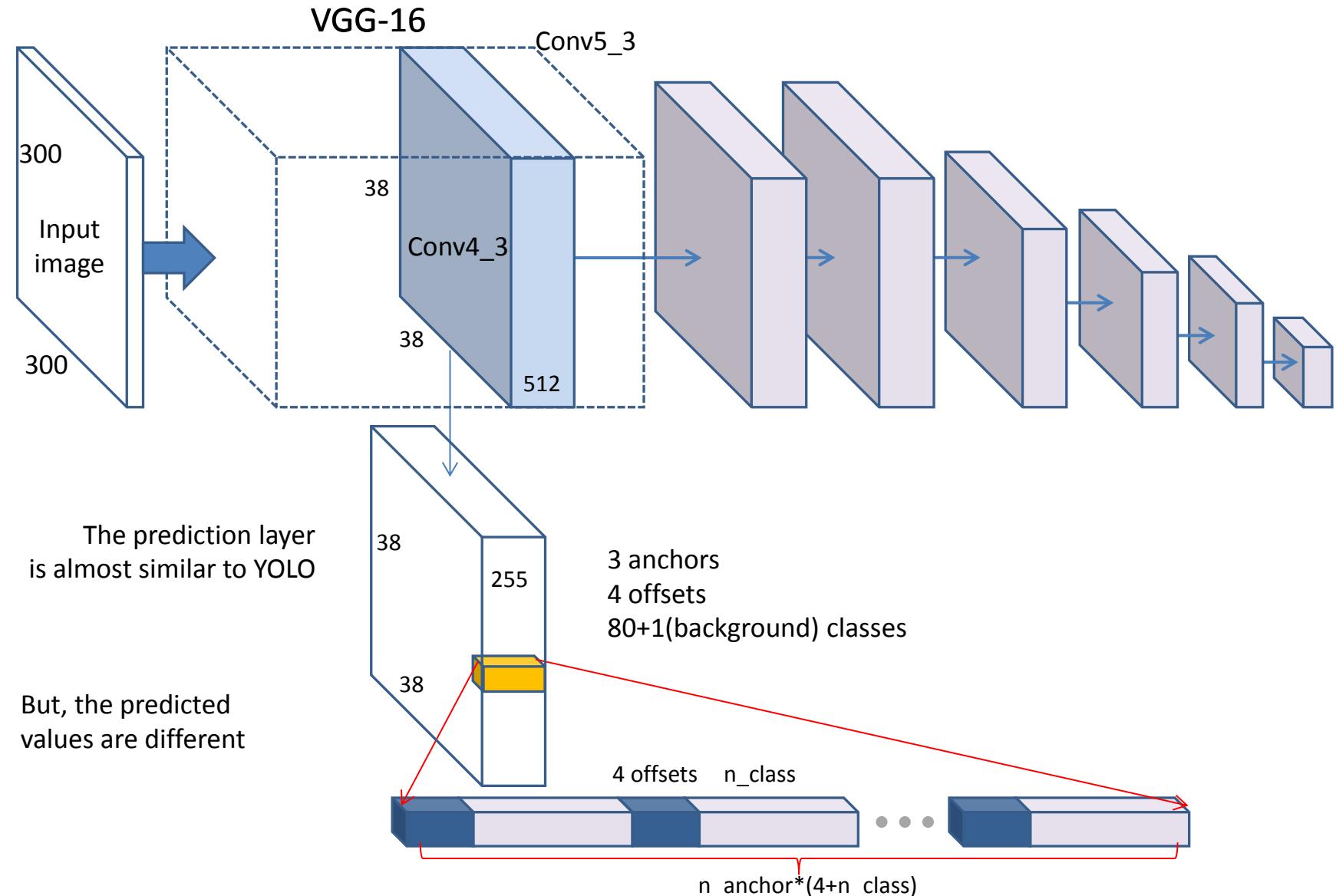


The Single Shot Detector (SSD)

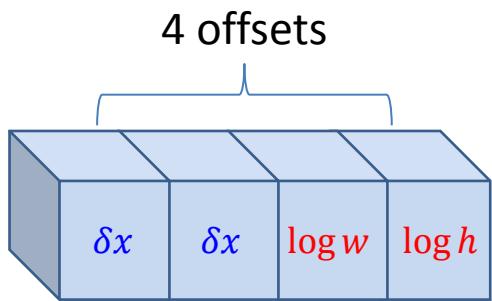




The Single Shot Detector (SSD)



SSD - Bounding Box Prediction

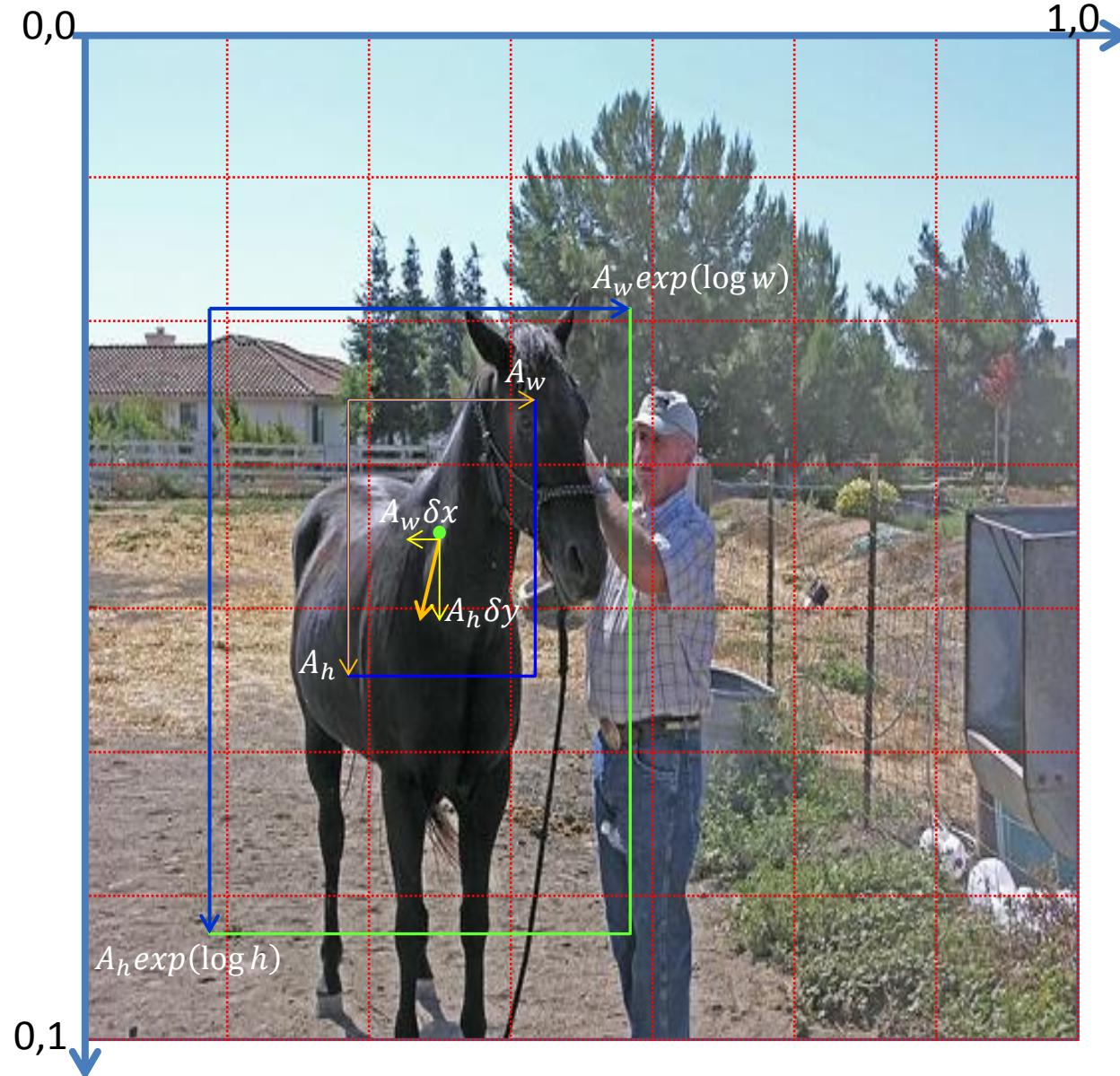


$$P_x = A_x + A_w \delta_x$$

$$P_y = A_y + A_h \delta_y$$

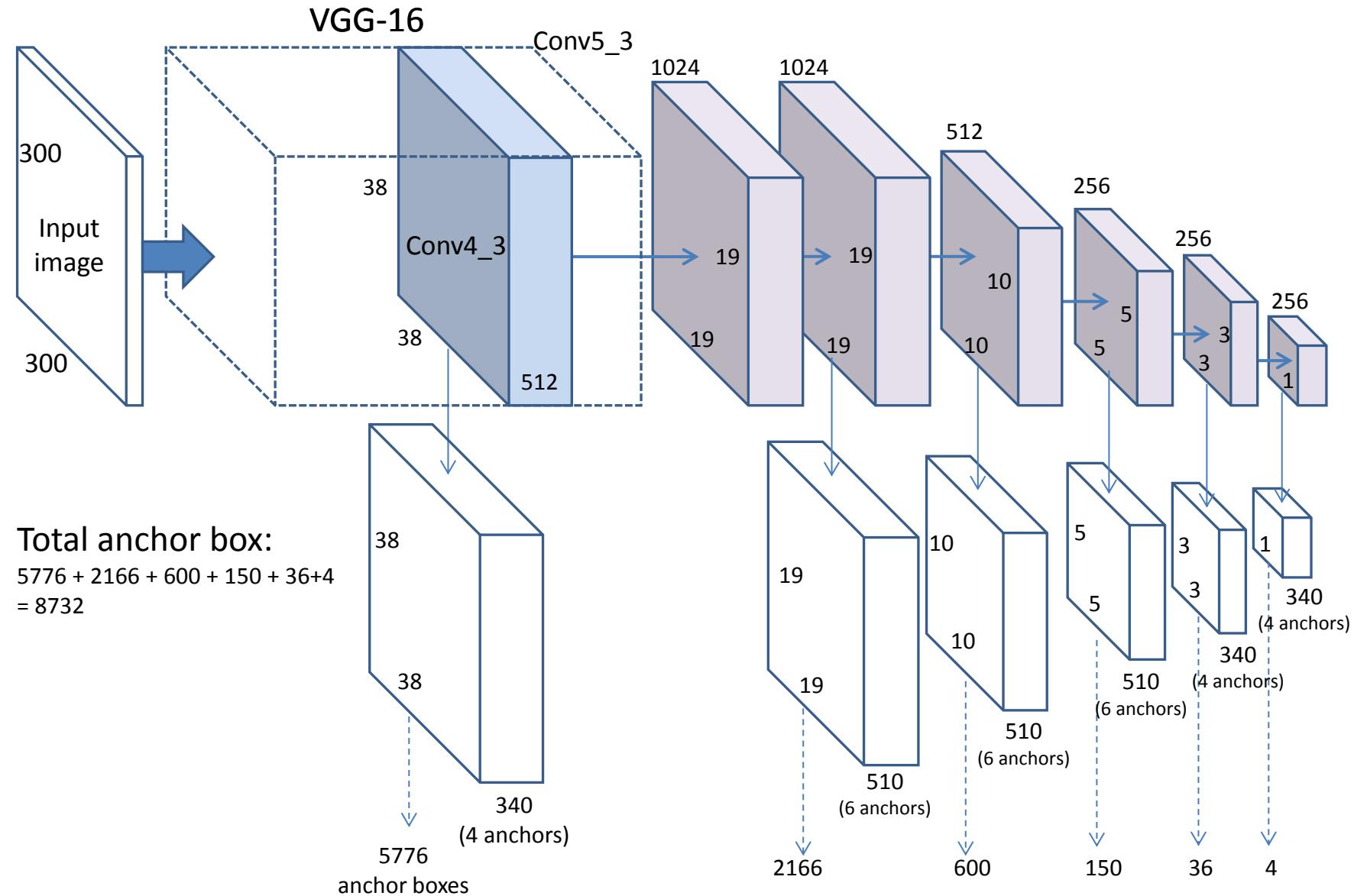
$$P_w = A_w \exp(\log w)$$

$$P_h = A_h \exp(\log h)$$





The Single Shot Detector (SSD)

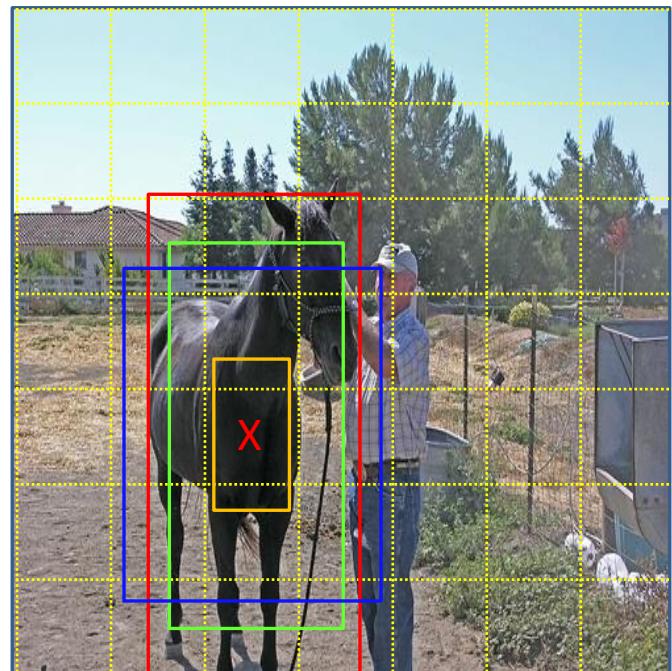




SSD - Training Objectives

- ▶ Only anchor boxes with high IoU are contributing to the loss values.
- ▶ 2 kinds of loss values:
 - ▶ Localization loss
 - ▶ Confidence loss

$$L = L_{conf} + \alpha L_{local}$$



Only anchor boxes with high IoU are contributing to the loss values

Ground-truth

Anchor 1 → Contribute to loss value

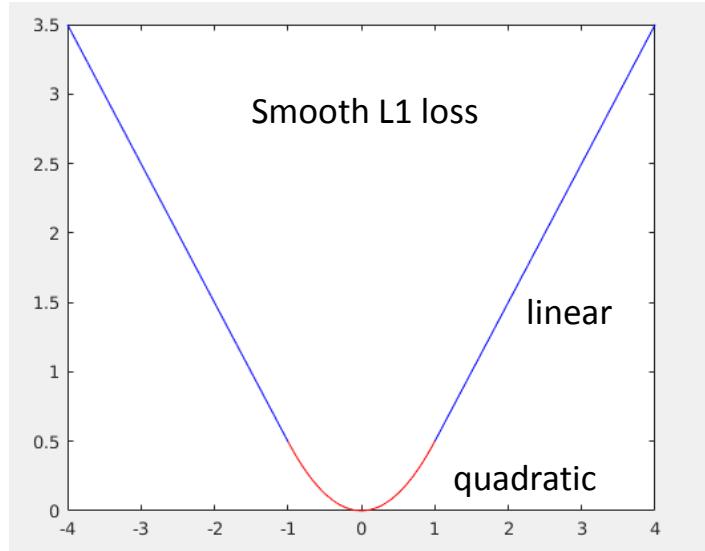
Anchor 2 → NOT contribute to loss value

Anchor 3 → Contribute to loss value

SSD - Losses

► Localization loss:

$$L_{loc} = \sum_{a \in \{a | IoU(a, G_{box}) > \tau\}} smooth_{L1}(P_{box} - G_{box})$$



► Confidence Loss:

$$L_{conf} = - \sum_{i \in Pos}^N \log(c_i^p) - \sum_{i \in Neg} \log(c_i^0)$$

For the matched anchor boxes, prediction should be same with the target value (p)

Softmax-ed class prediction value

For other anchor boxes, prediction should be indicating background class (0)

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

Generally known as Huber Loss

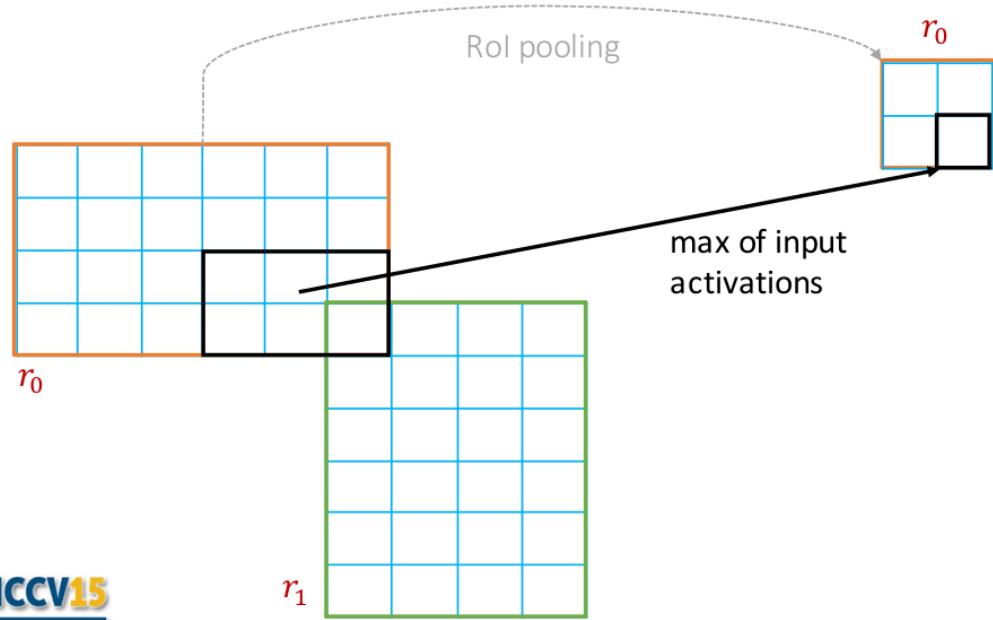


Thank you very much
for your attention!



ROI pooling is differentiable

ROI pooling / SPP is just like max pooling, except that pooling regions overlap

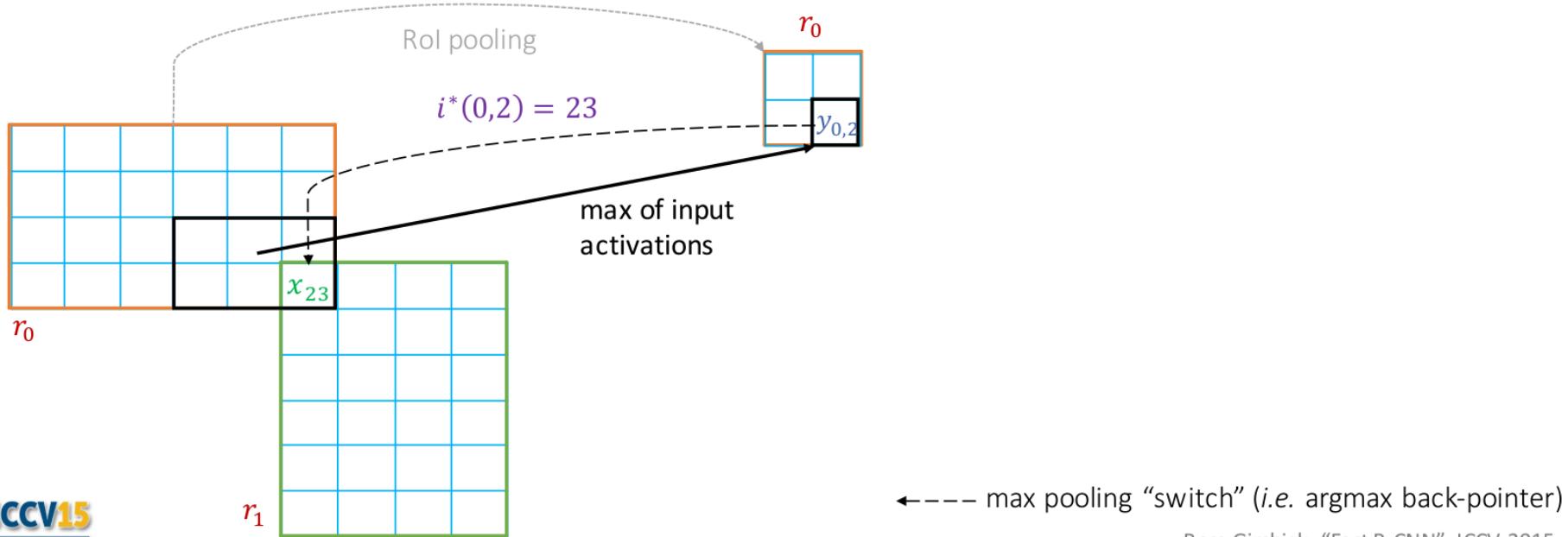


Ross Girshick. "Fast R-CNN". ICCV 2015



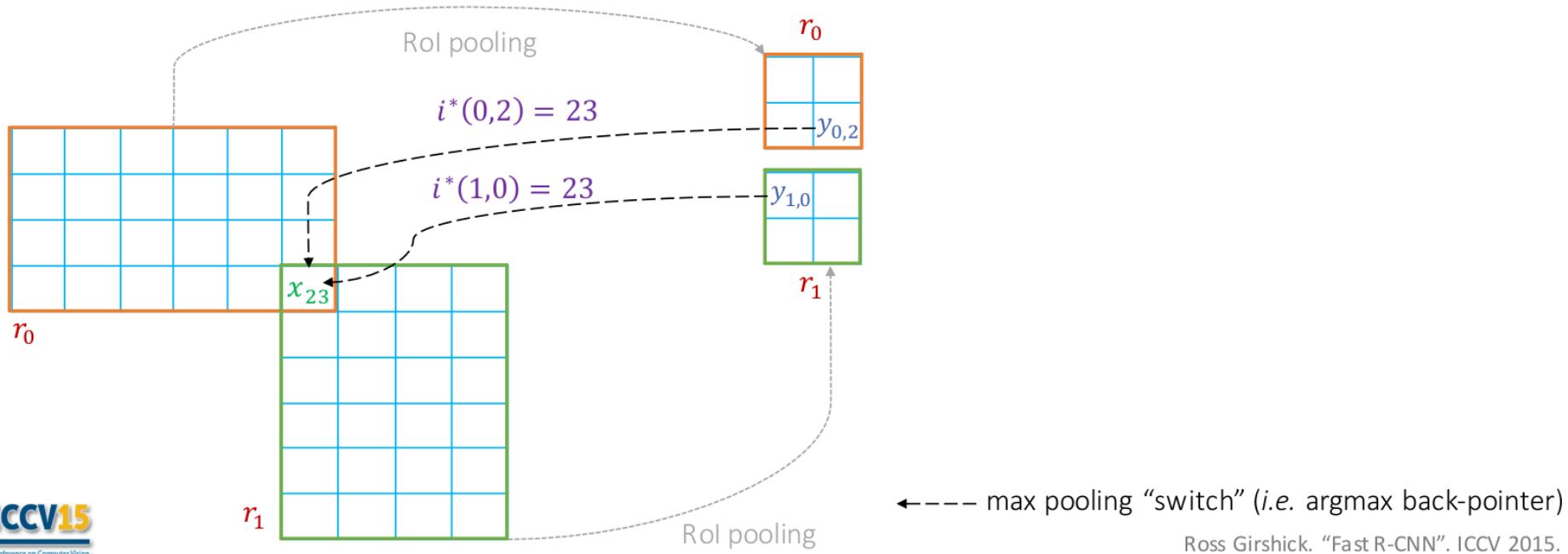
ROI pooling is differentiable

ROI pooling / SPP is just like max pooling, except that pooling regions overlap



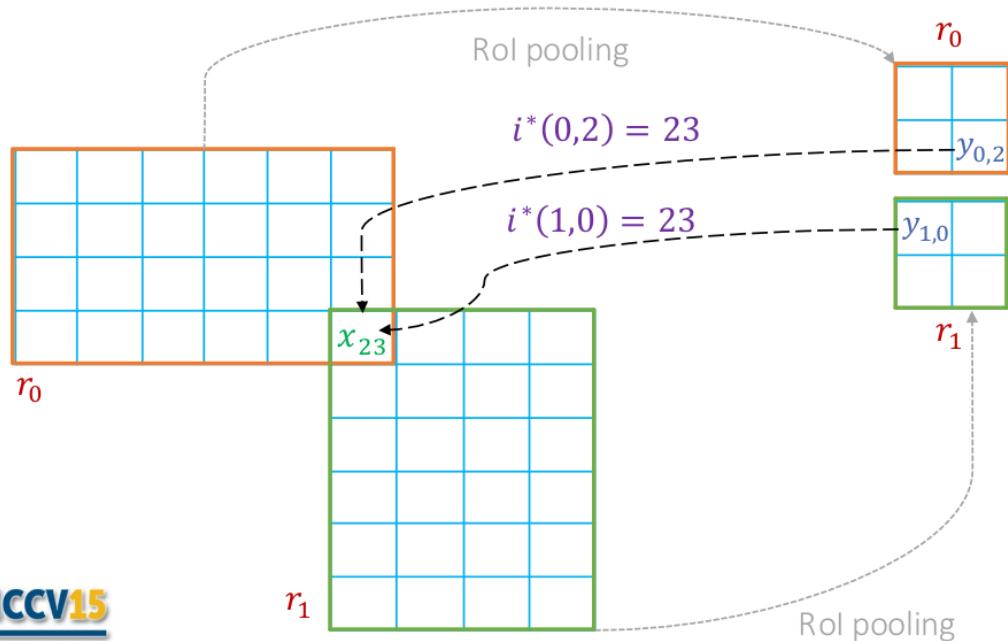
ROI pooling is differentiable

ROI pooling / SPP is just like max pooling, except that pooling regions overlap



ROI pooling is differentiable

ROI pooling / SPP is just like max pooling, except that pooling regions overlap



$$\frac{\partial L}{\partial x_i} = \sum_r \sum_j [i = i^*(r, j)] \frac{\partial L}{\partial y_{rj}}$$

1 if r, j "pooled"
input i ; 0 o/w

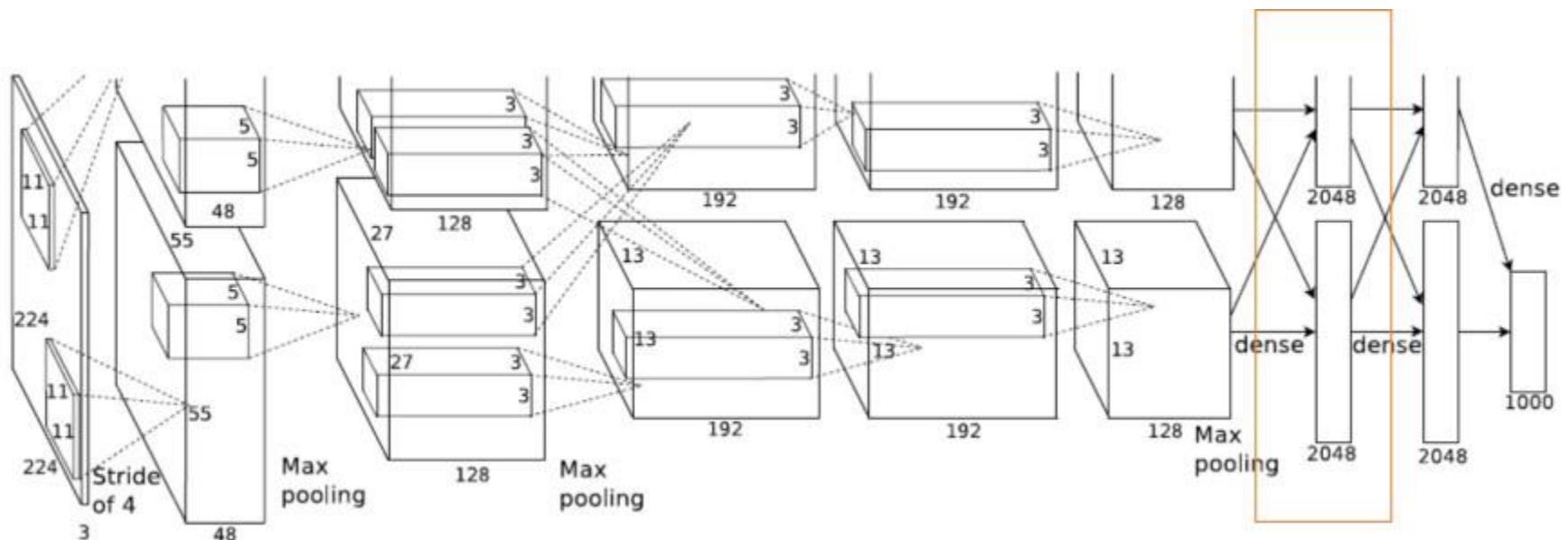
Partial Over regions r ,
for x_i locations j

Partial from
next layer

←---- max pooling "switch" (i.e. argmax back-pointer)

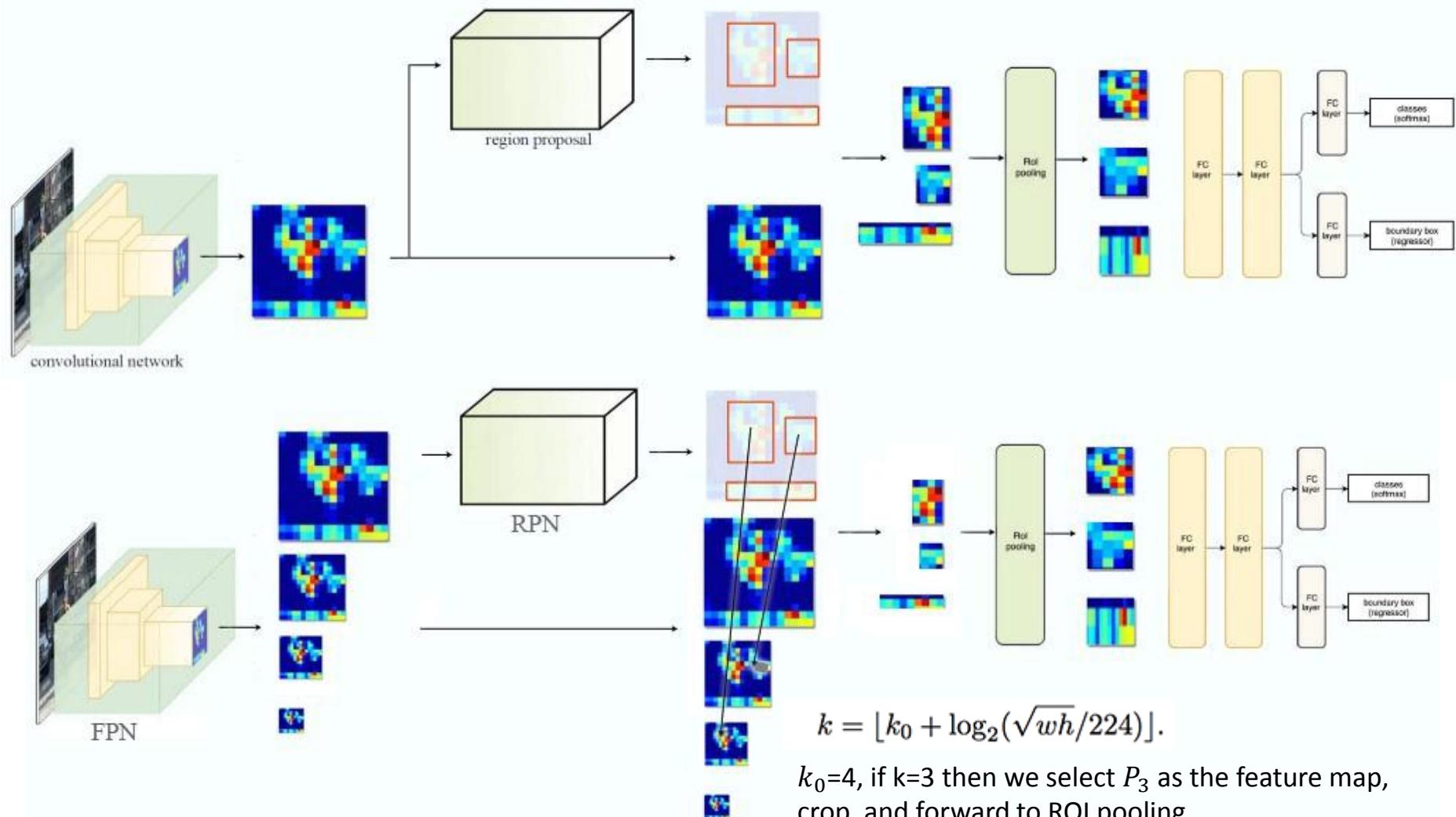
Ross Girshick. "Fast R-CNN". ICCV 2015.

Alexnet



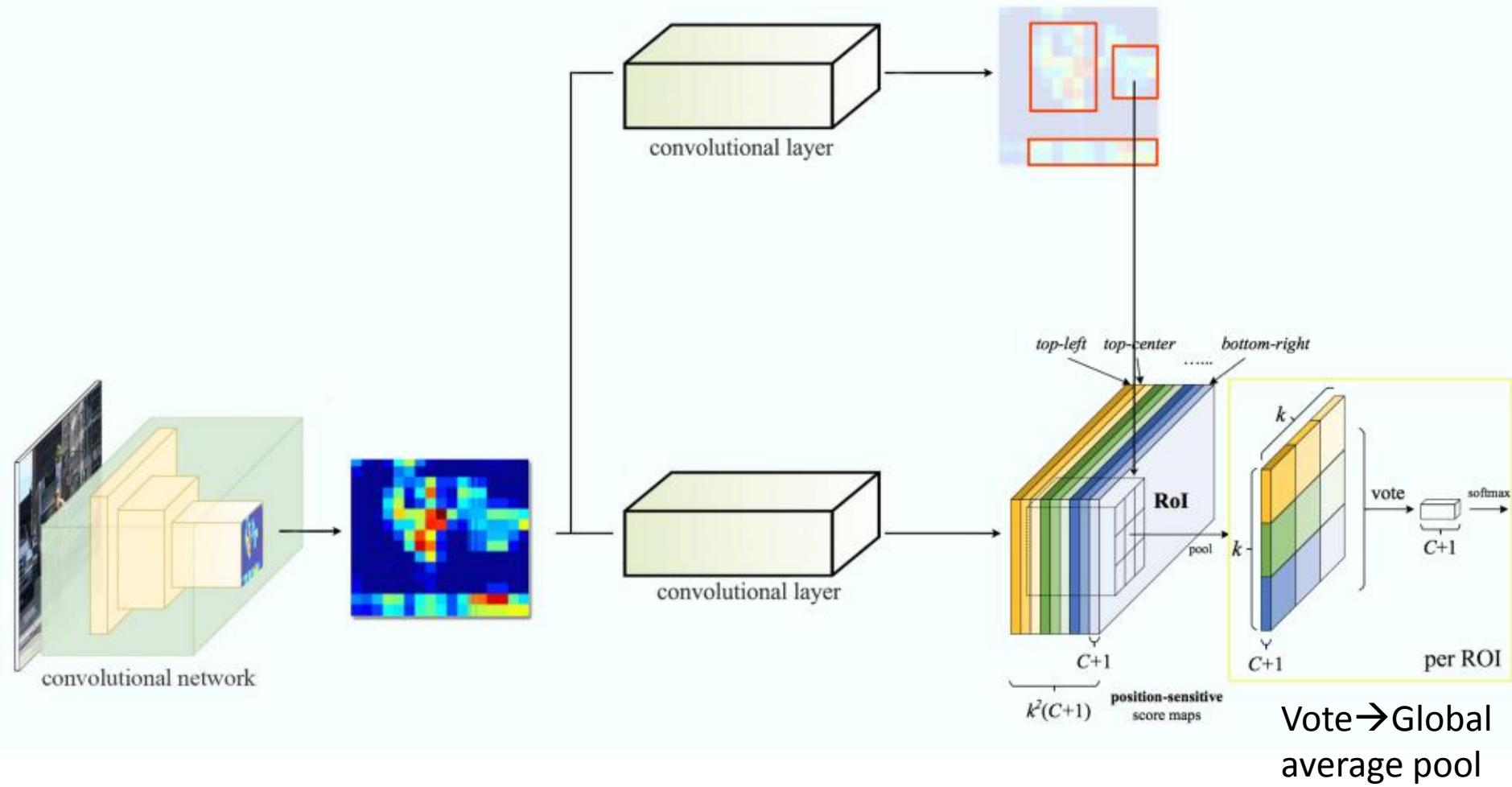


FPN



https://medium.com/@jonathan_hui/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c

R-FCN



https://medium.com/@jonathan_hui/understanding-region-based-fully-convolutional-networks-r-fcn-for-object-detection-828316f07c99