GDAŃSK UNIVERSITY OF TECHNOLOGY

ISLab Since1998

International Summer School on Deep Learning

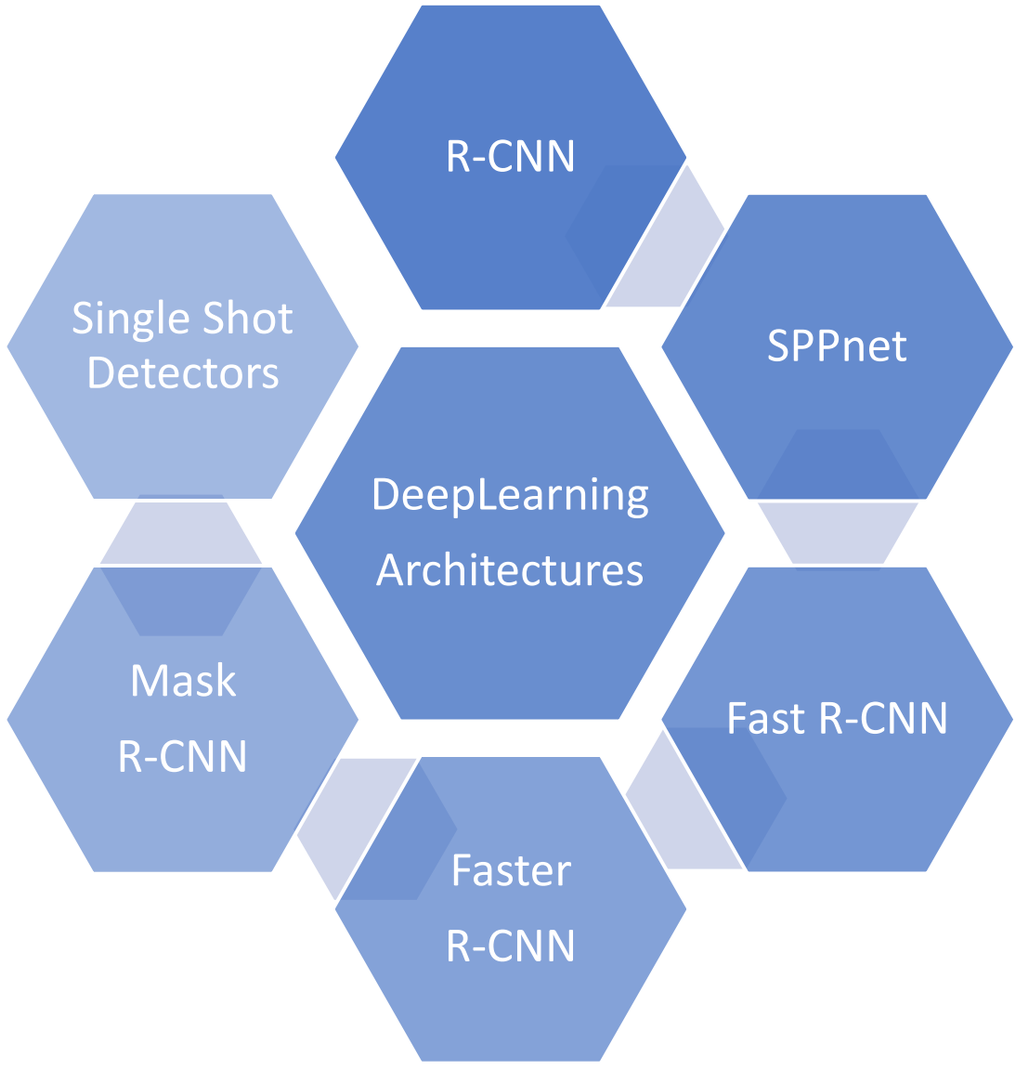
July 2nd-6th 2018, Gdansk, Poland

Deep Learning for Object Detection

University of Ulsan

Today's Topics

Intro to Google Colab



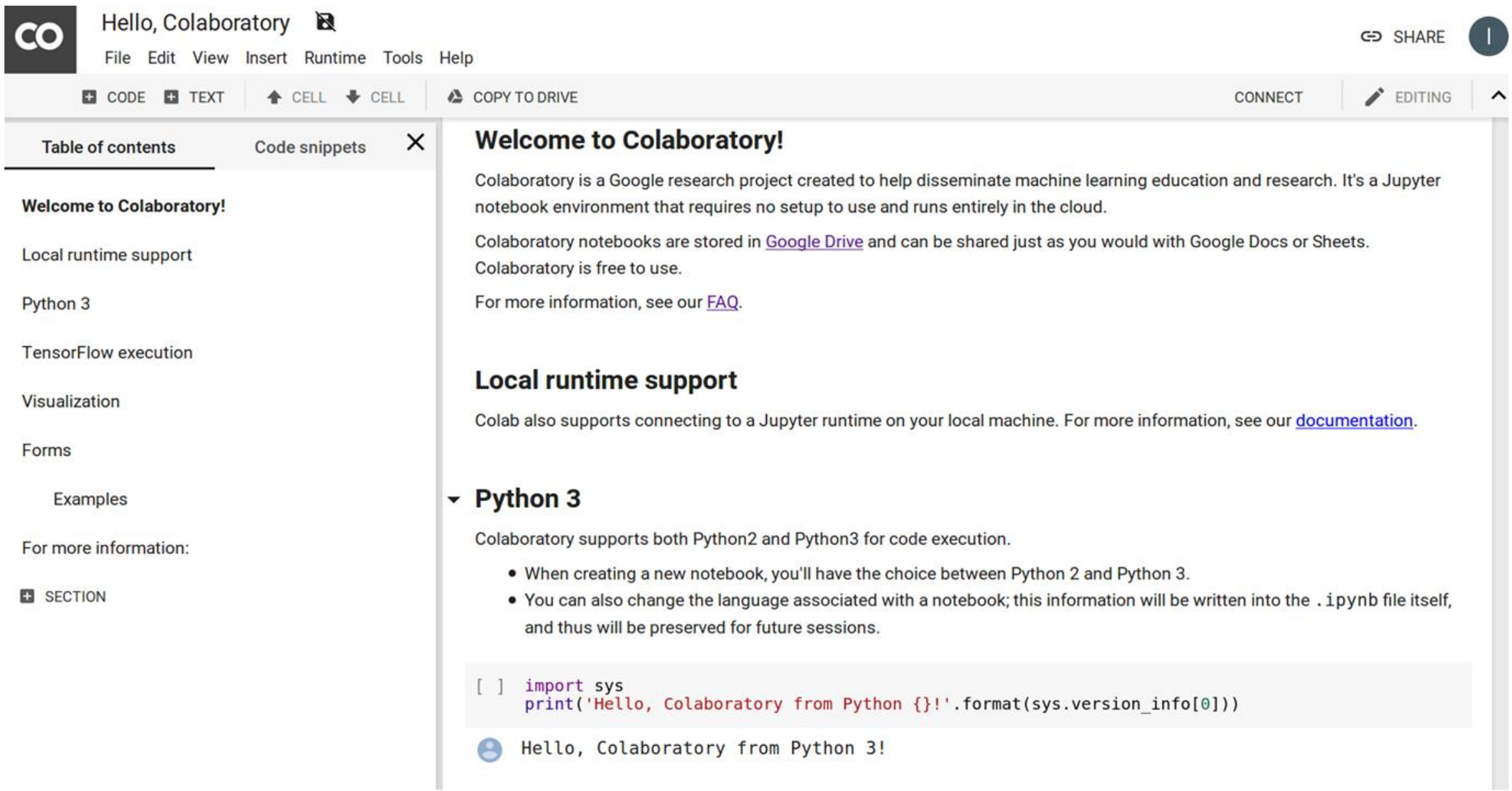
Architectures for object detection

TensorFlow object detection API

Intoduction to the Google Colab

https://colab.research.google.com

The Colab Environment – First Look



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
Mem:          13029         420        2258
Swap:             0           0           0
```

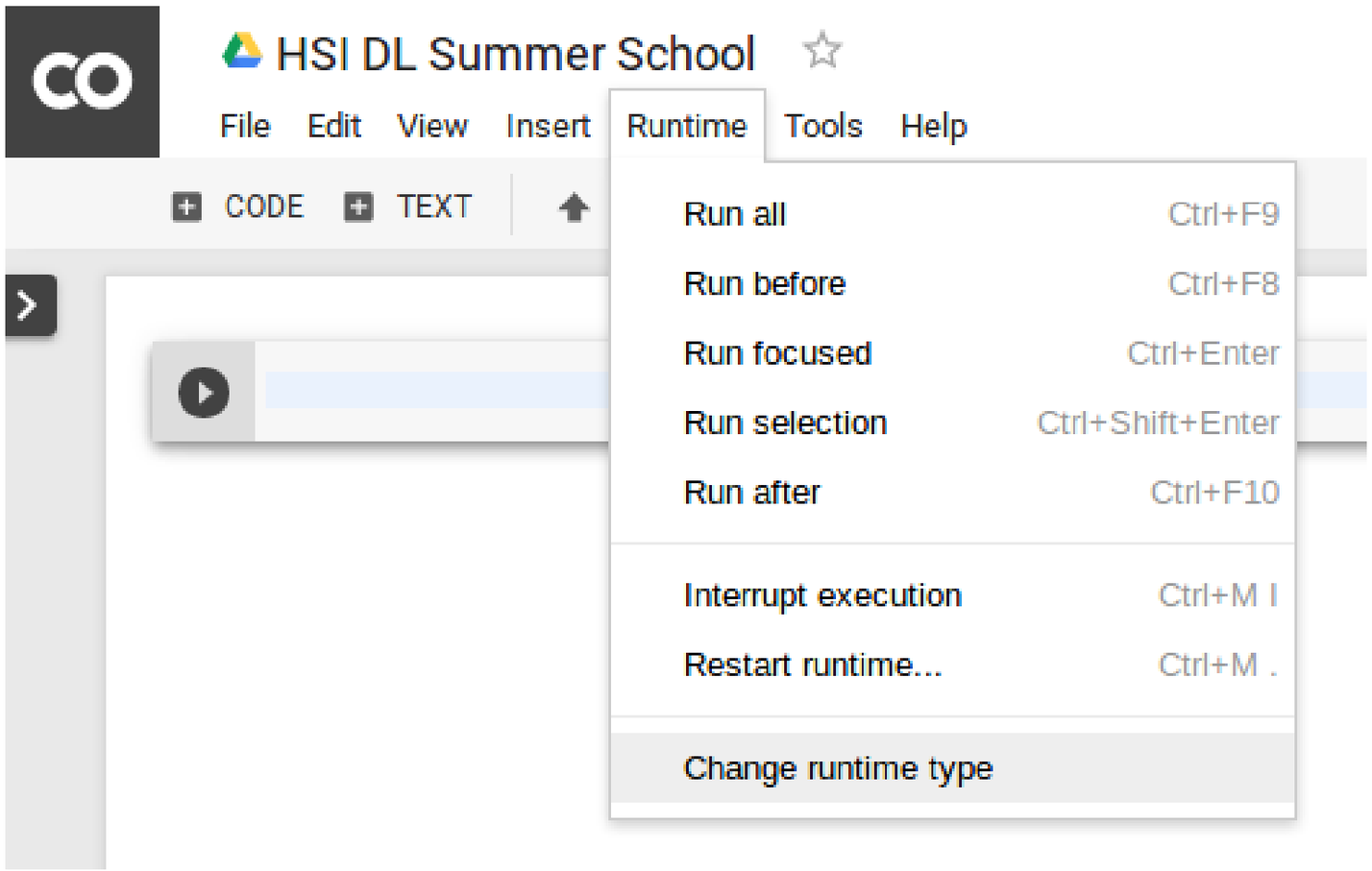
What is the Operating System?

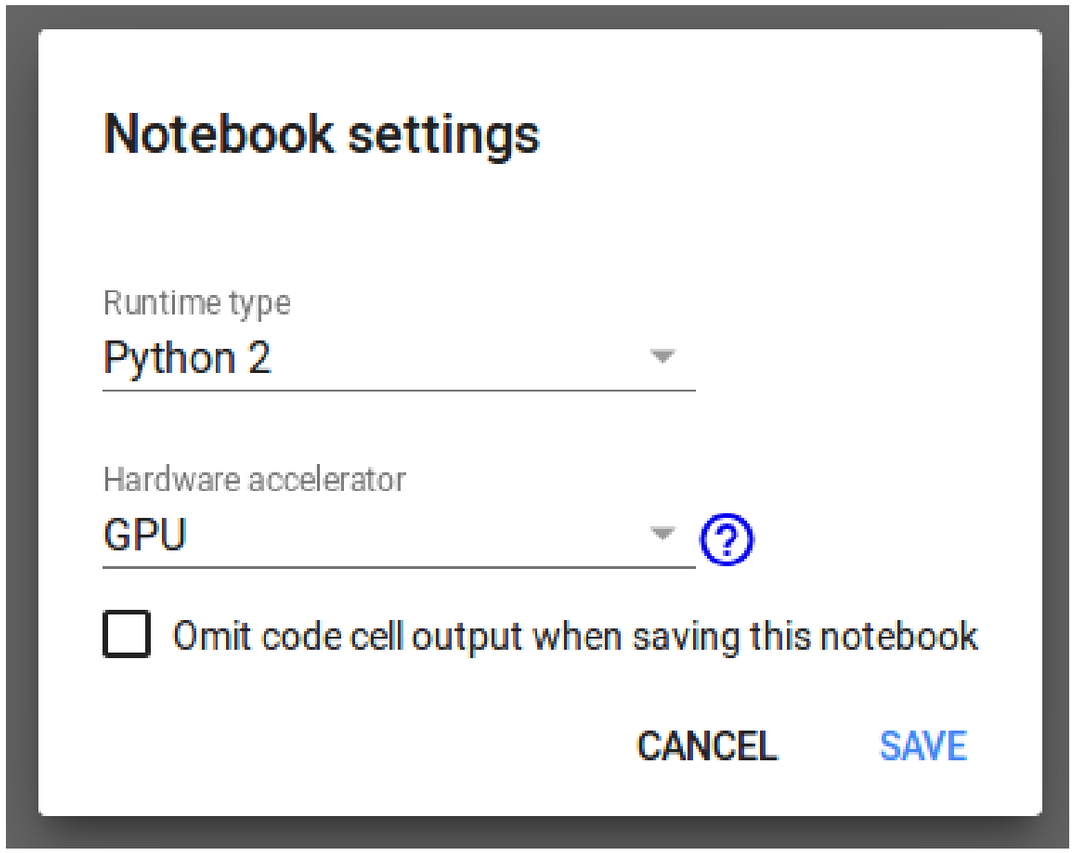
```
!cat /etc/*release
```

```
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 the GPU





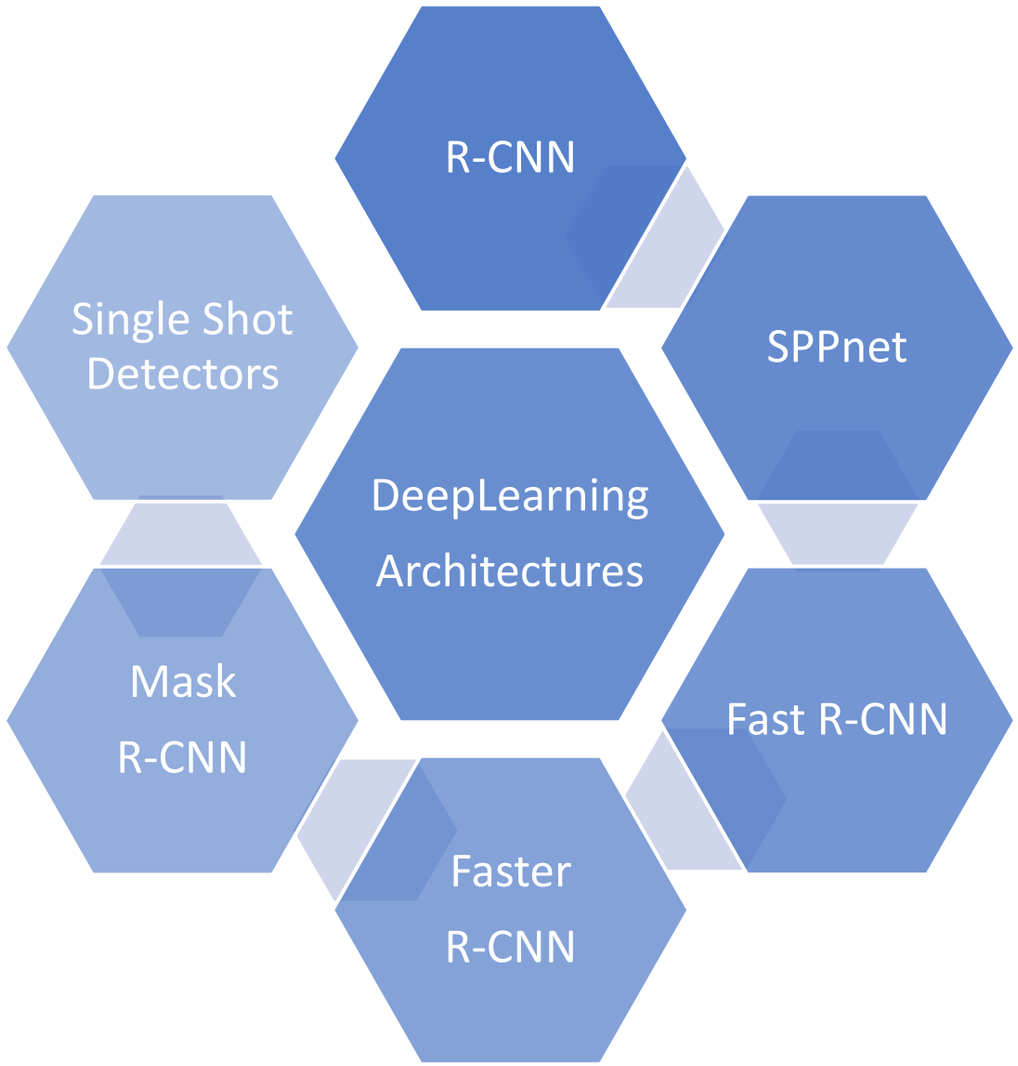
Depends on your luck, you may (not) get access to a **GPU**!

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                                                 |
+-----------------------------------------------------------------------------+
```

Object Detection Meta-architectures

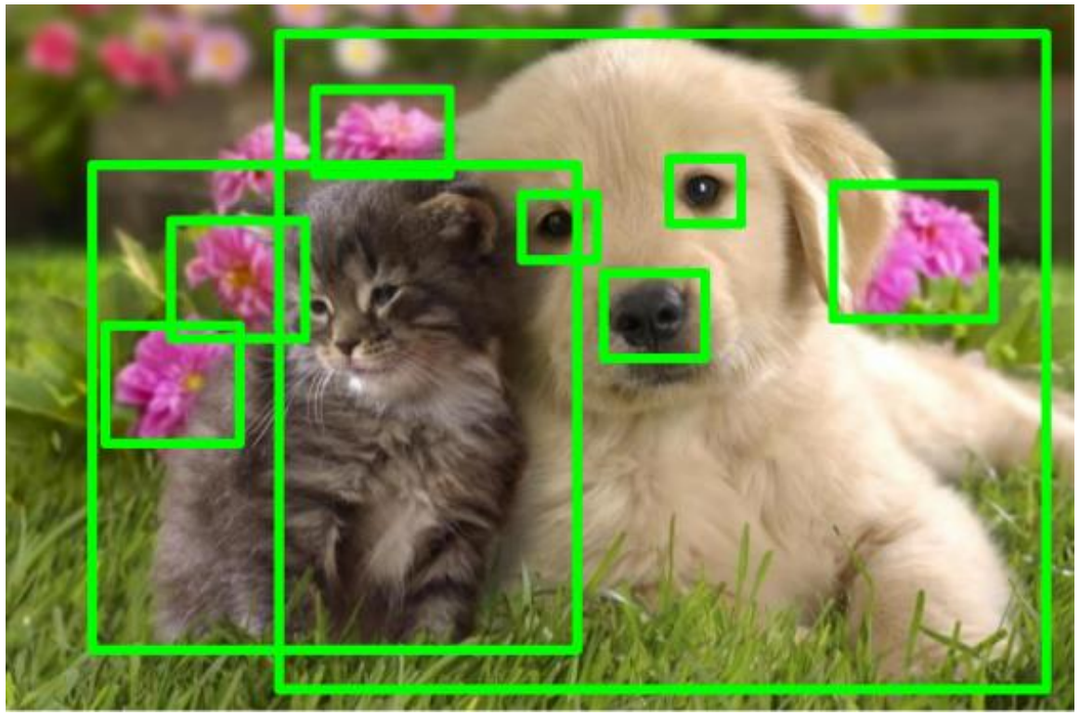


The Classical Object Detector

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

Region Proposal: Search Space Reduction

- Find regions on image that likely to contain objects

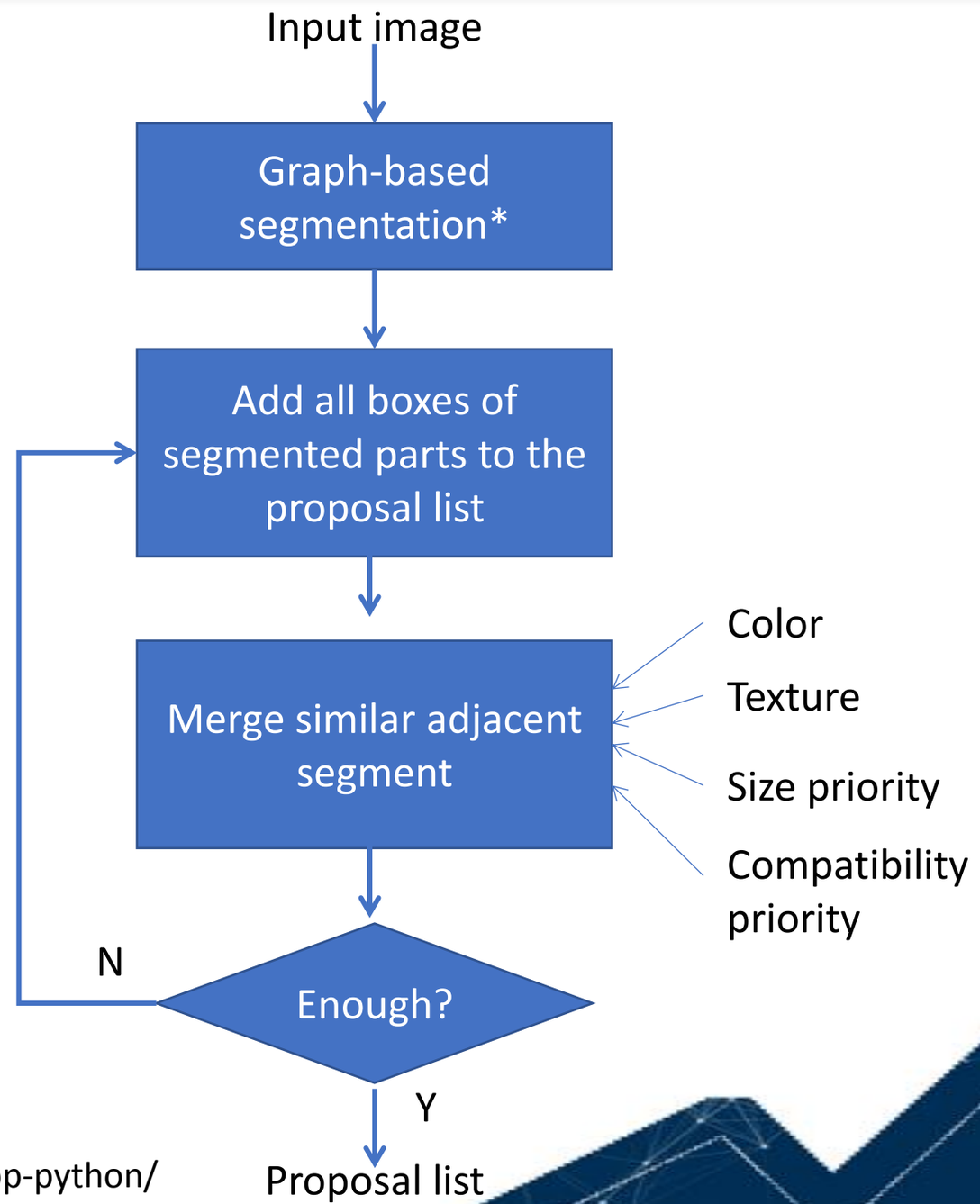


Images: adapted from Fei-fei Li's presentation

Region Proposal: Selective Search

Input image

Over-segmented output*

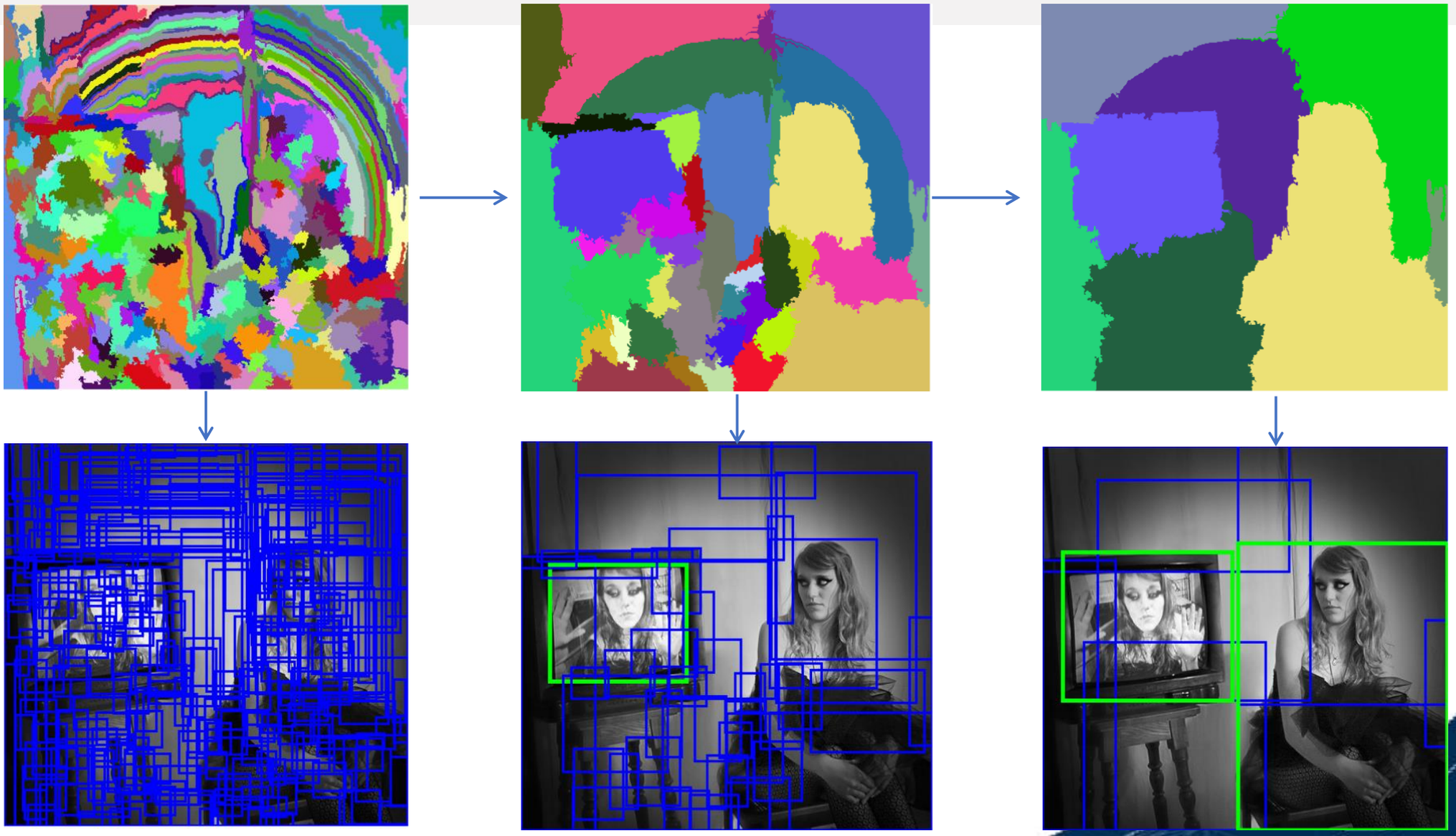


Images: https://www.learnopencv.com/selective-search-for-object-detection-cpp-python/

* P. Felzenszwalb ,et.al. , Efficient Graph-based Image Segmentation, IJCV, 2004. Code: http://cs.brown.edu/people/pfelzens/segment/

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

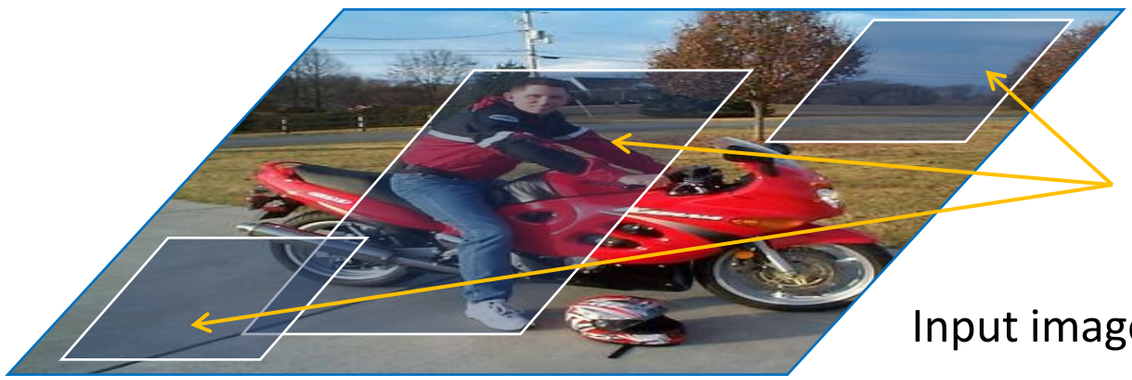
Method	Approach	Outputs Segments	Outputs Score	Control #proposals	Time (sec.)	Repea-tability	Recall Results	Detection Results
Bing [18]	Window scoring		✓	✓	0.2	⋆⋆⋆	⋆	·
CPMC [19]	Grouping	✓	✓	✓	250	-	⋆⋆	⋆
EdgeBoxes [20]	Window scoring		✓	✓	0.3	⋆⋆	⋆⋆⋆	⋆⋆⋆
Endres [21]	Grouping	✓	✓	✓	100	-	⋆⋆⋆	⋆⋆
Geodesic [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	·	·	·

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

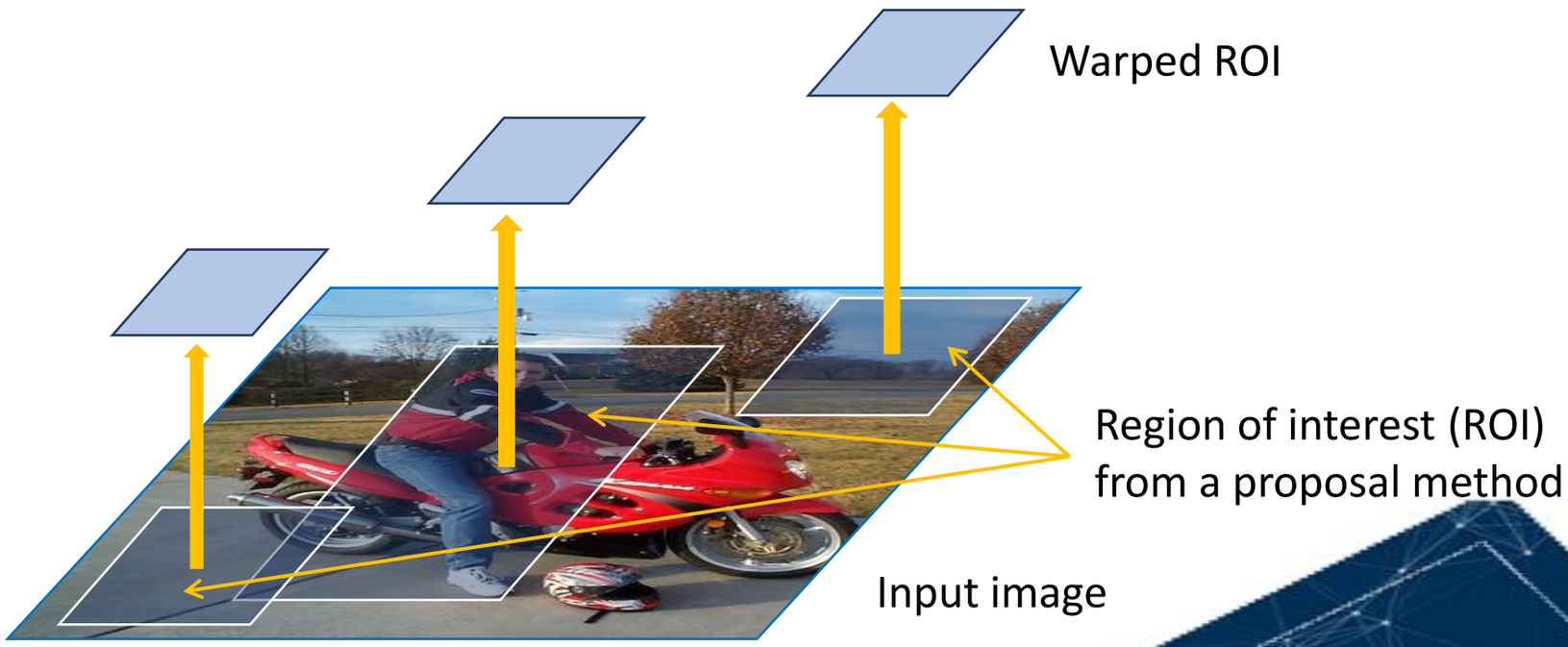
The R-CNN Method

Input image

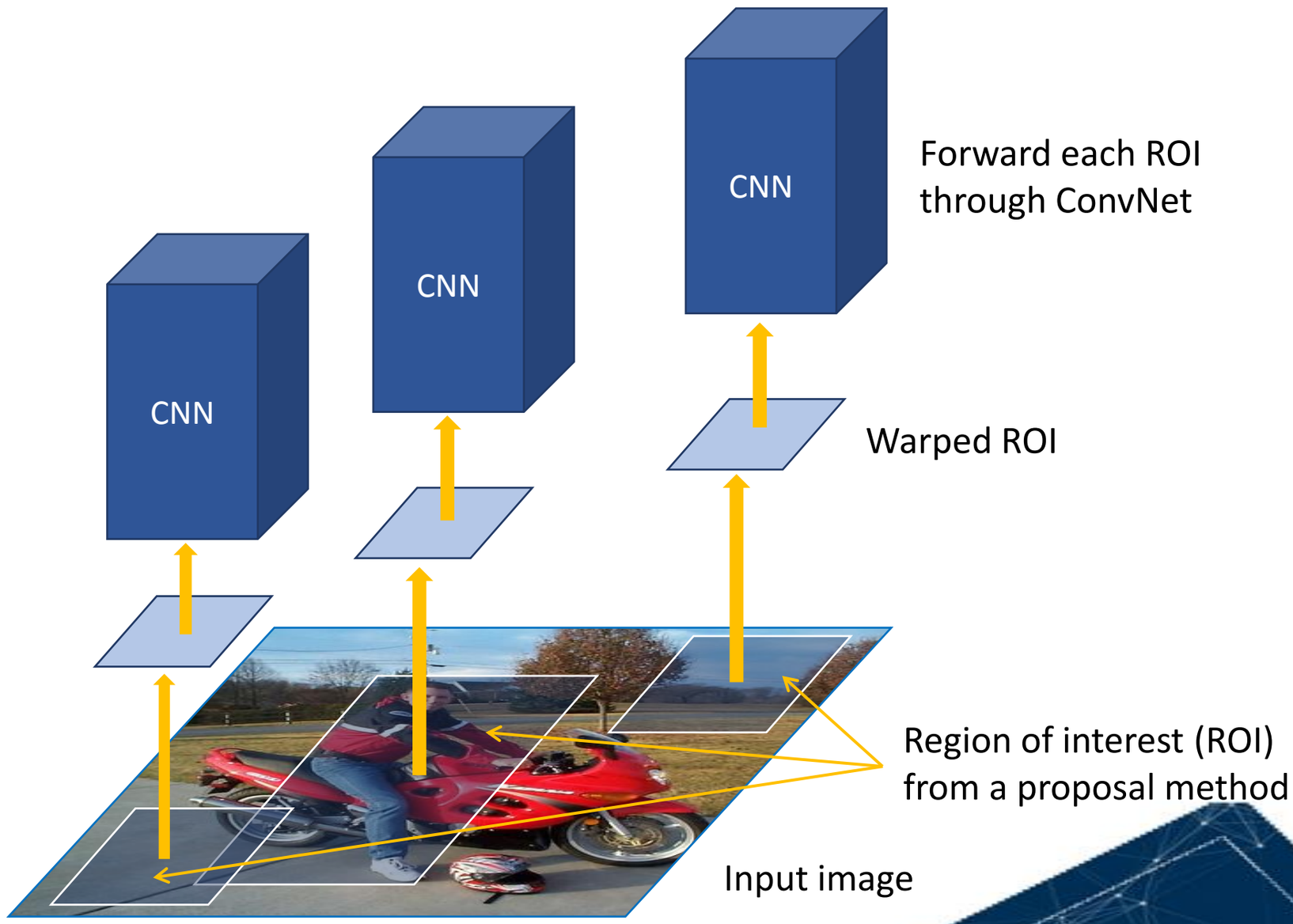
The R-CNN Method



The R-CNN Method

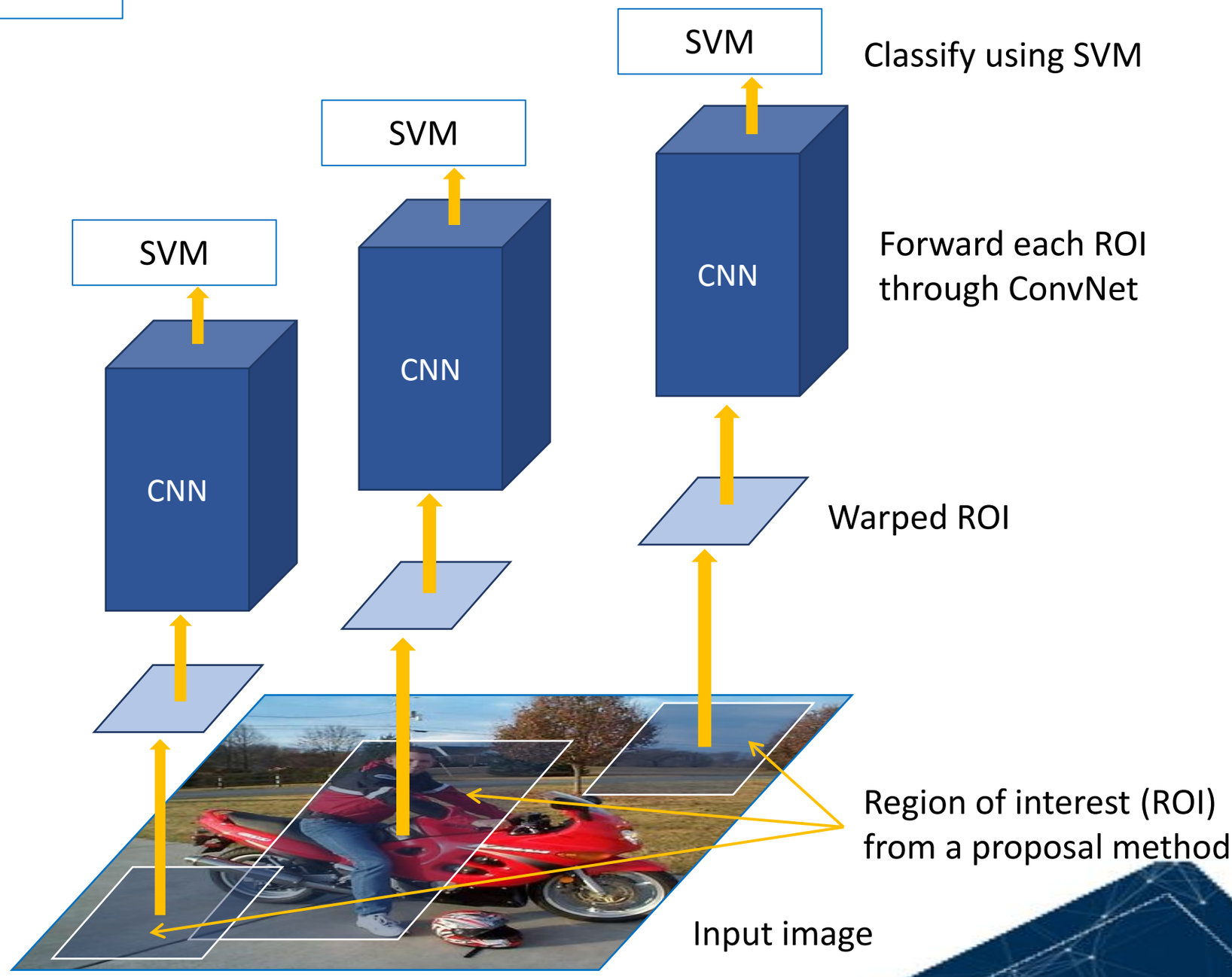


The R-CNN Method

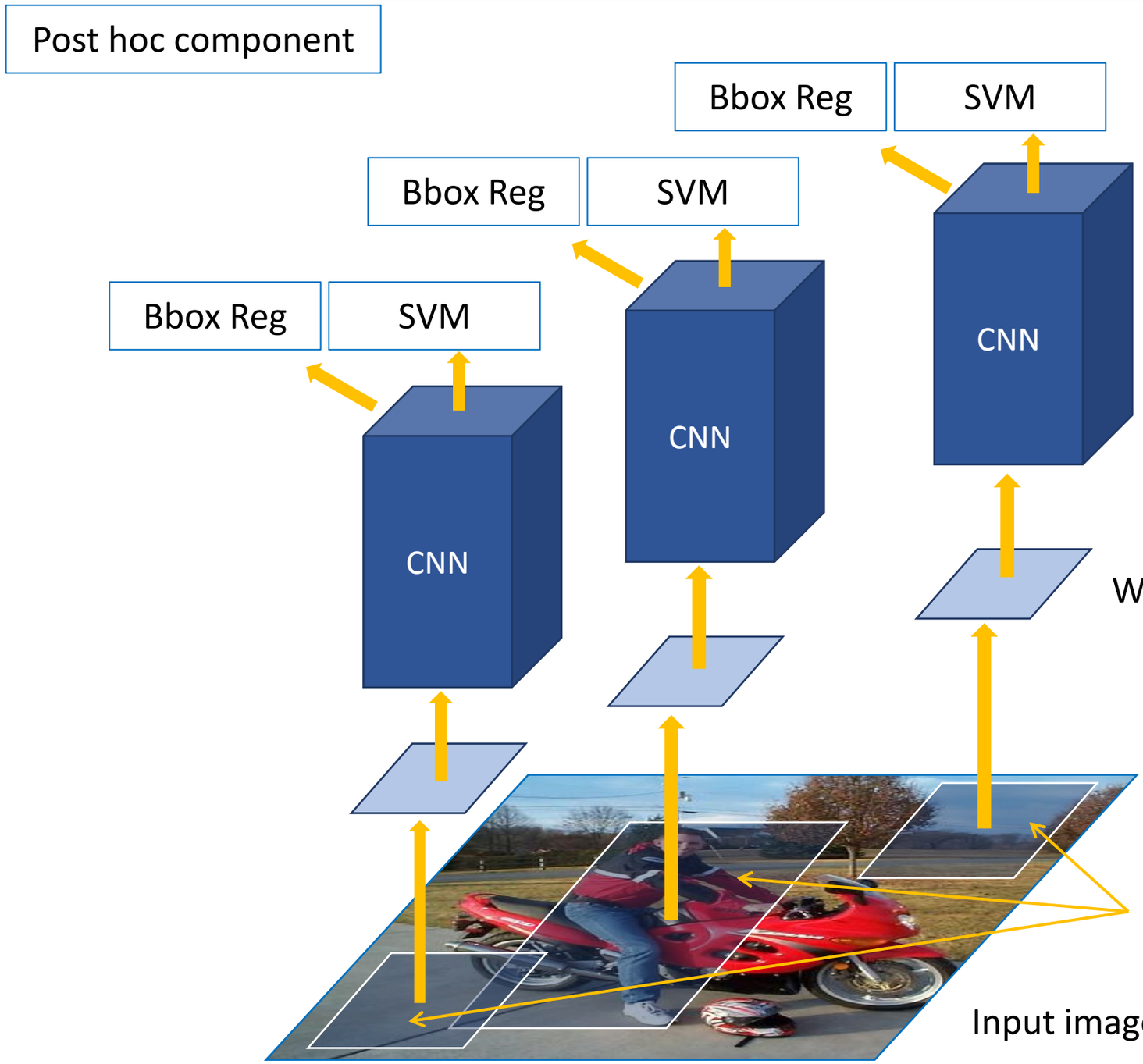


The R-CNN Method

Post hoc component

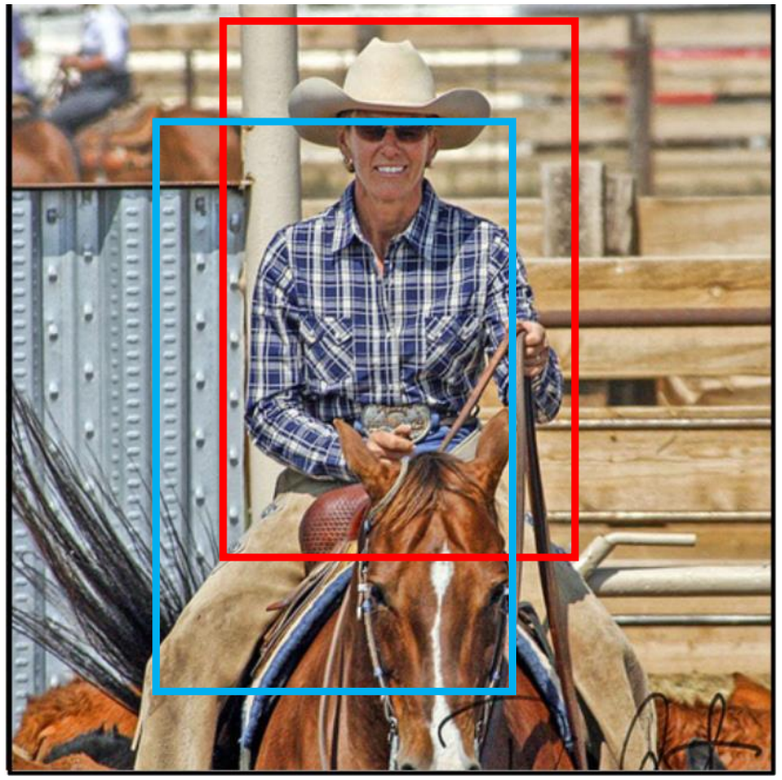


The R-CNN Method



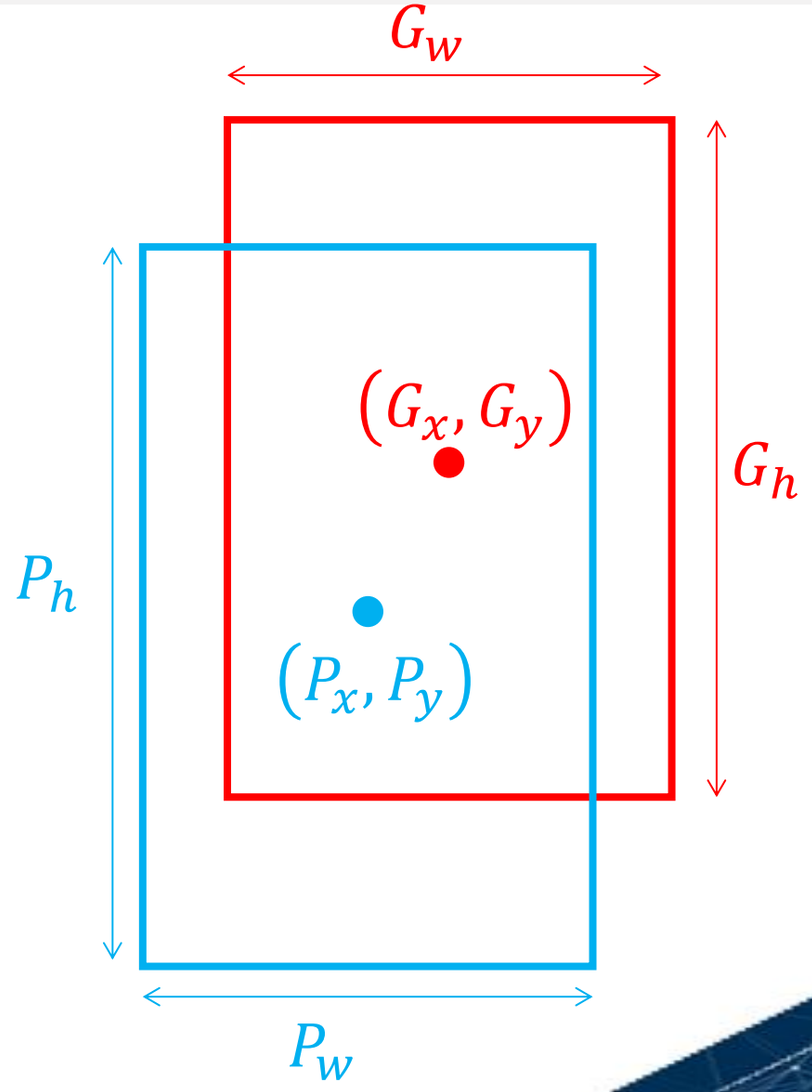
Bounding Box Regression (1/2)

- Goal:
  - refines the position and size of the detection result.



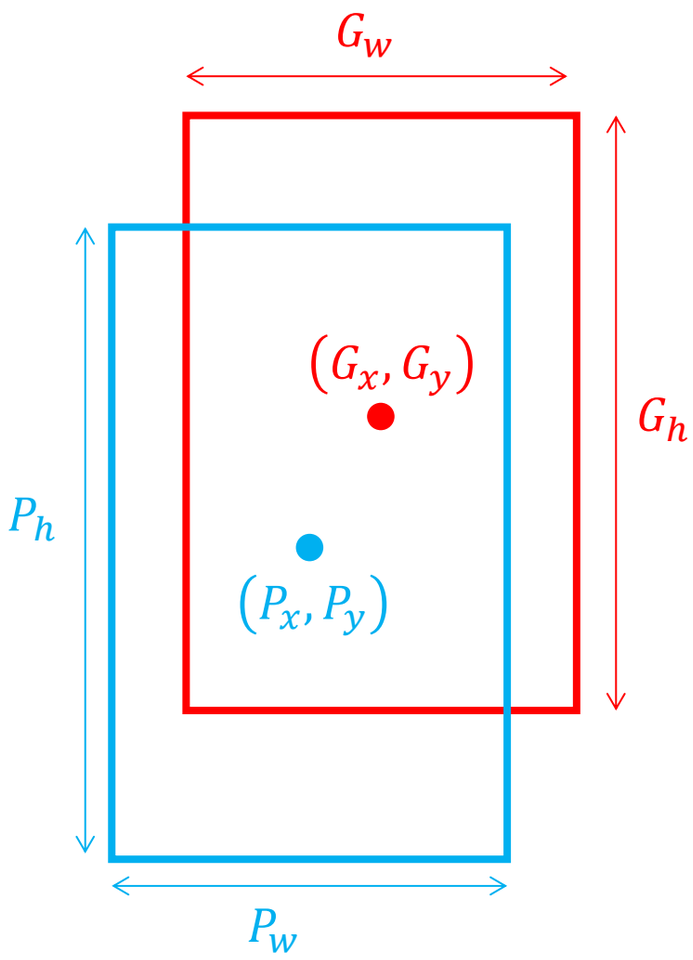
Ground-truth

Prediction



Bounding Box Regression (2/2)

- Solution: Predicting offsets using the feature.



Target values

$$t_x = (G_x - P_x)/P_w$$

$$t_y = (G_y - P_y)/P_h$$

$$t_w = log(G_w/P_w)$$

$$t_h = log(G_h/P_h)$$

Refinement result

$$\hat{G}_x = P_x + P_w d_x(P)$$

$$\hat{G}_y = P_y + P_h d_y(P)$$

$$\hat{G}_w = P_w exp(d_w(P))$$

$$\hat{G}_h = P_h exp(d_h(P))$$

Refinement parameters (To be predicted)

Problem formulation:

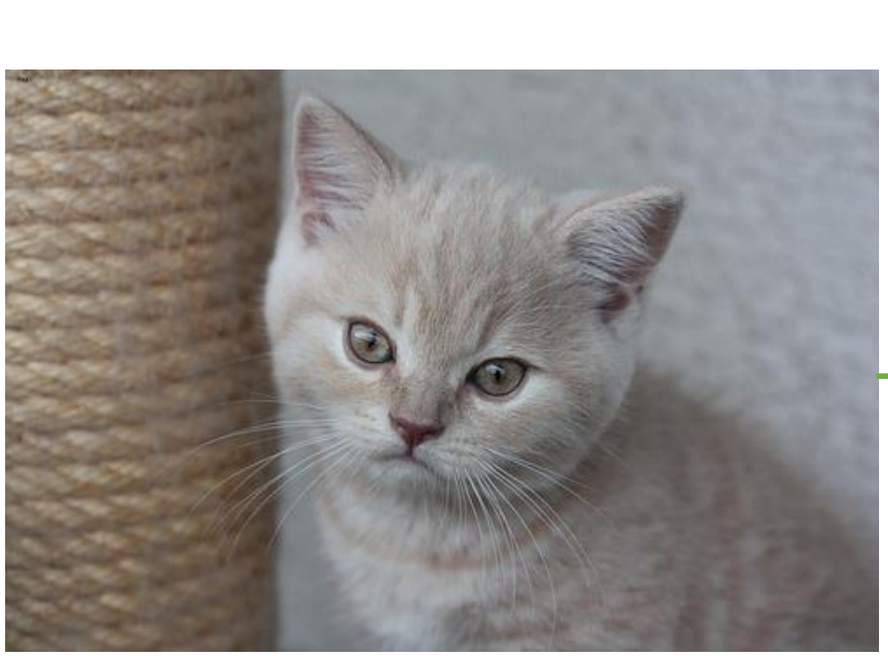
Where: $d_\star(P) = w_\star{}^T \emptyset_5(P)$

Feature from pool$_5$

$$\min_{w_\star} \sum_i^N (t_\star - w_\star{}^T \emptyset_5(P))^2 + \lambda \|w_\star\|^2$$

R-CNN Training

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



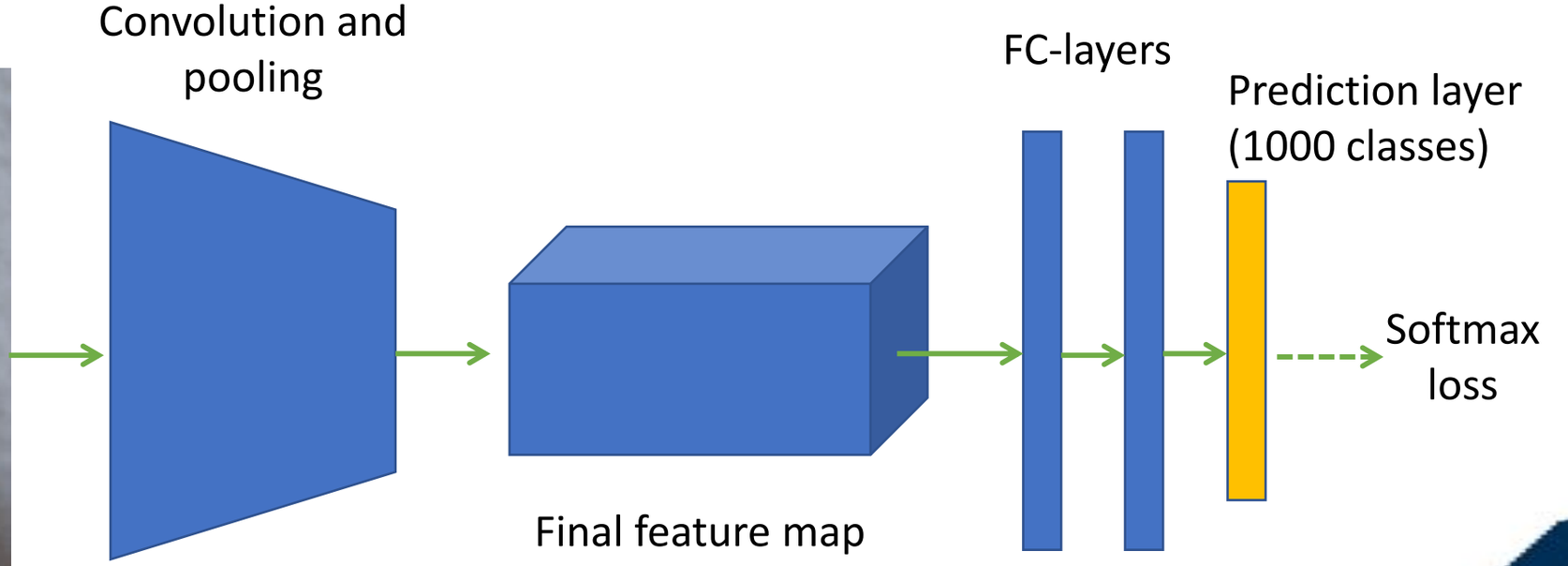
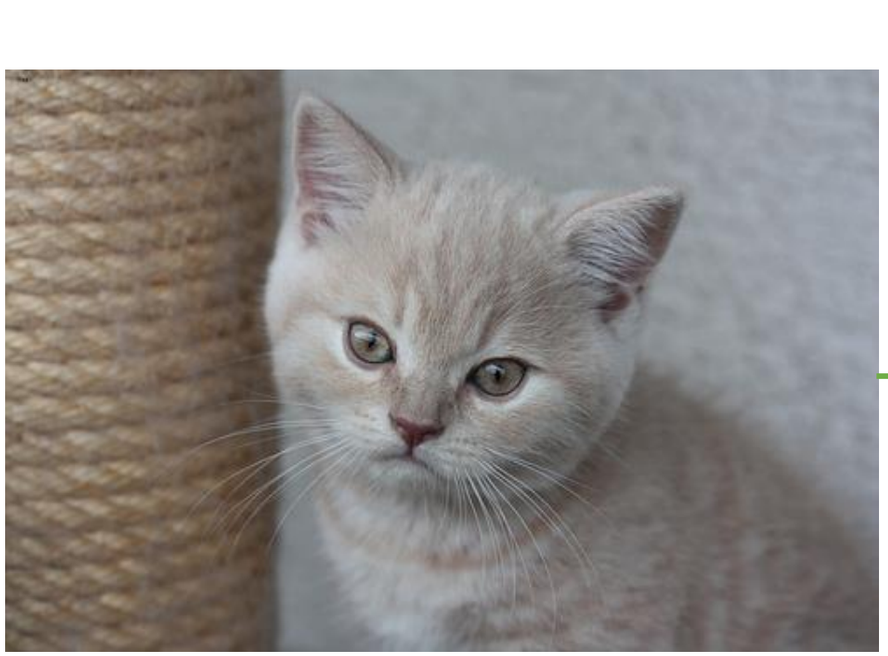


Image: http://www.robots.ox.ac.uk/~vgg/data/pets/

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



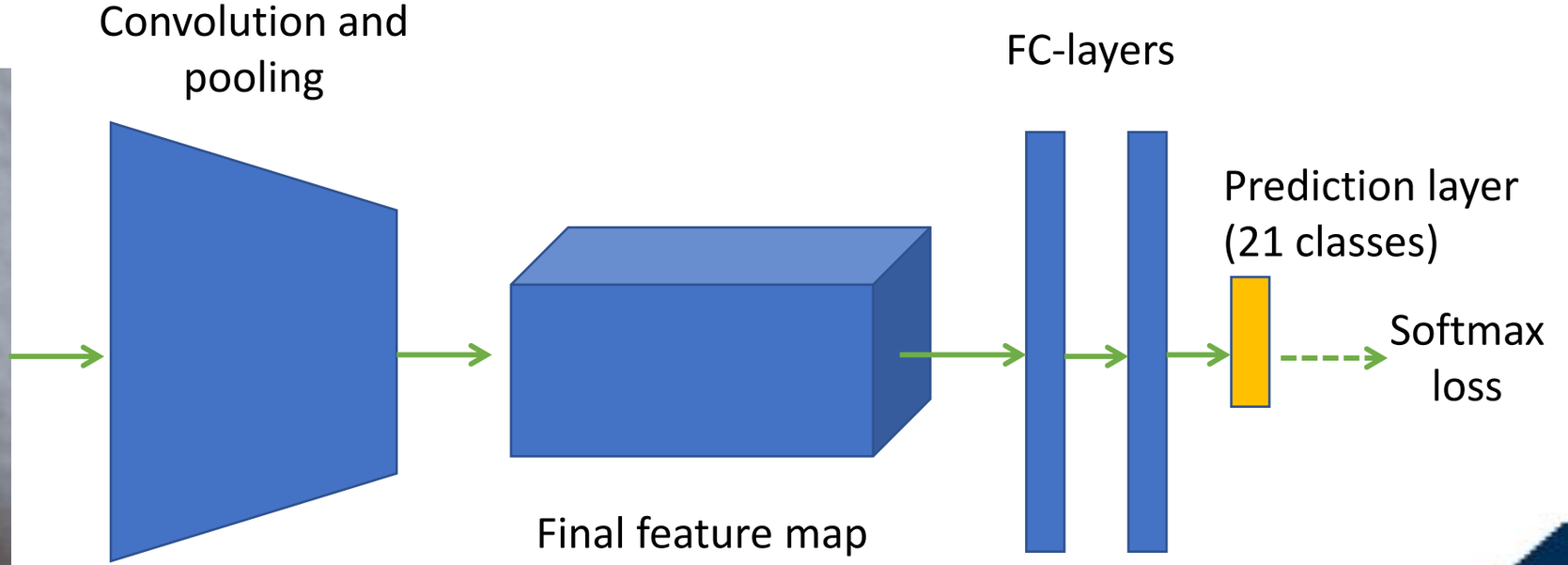


Image: http://www.robots.ox.ac.uk/~vgg/data/pets/

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

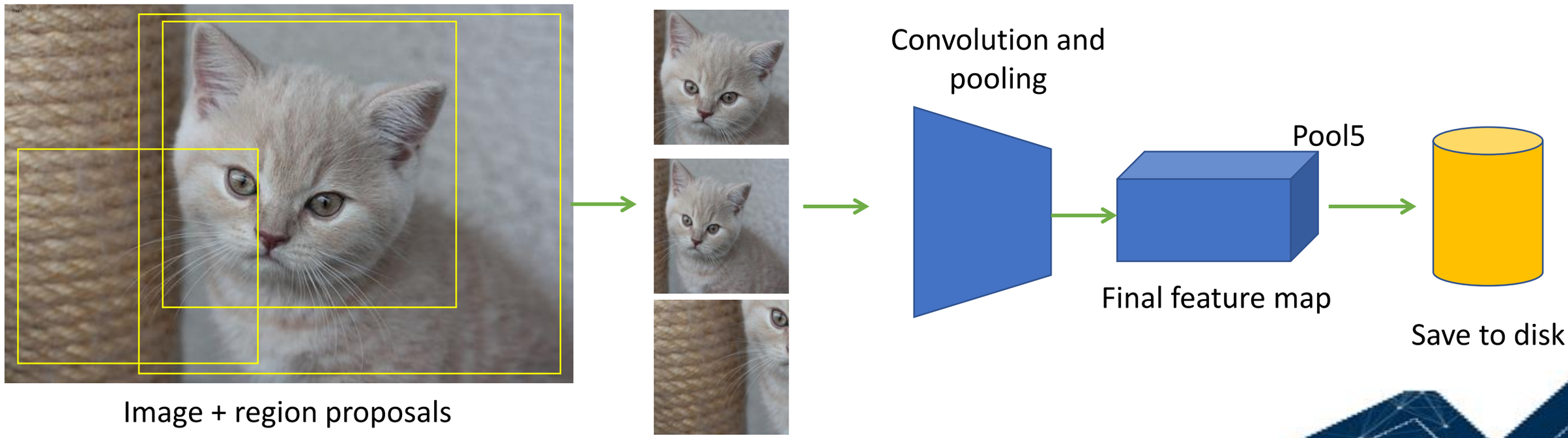
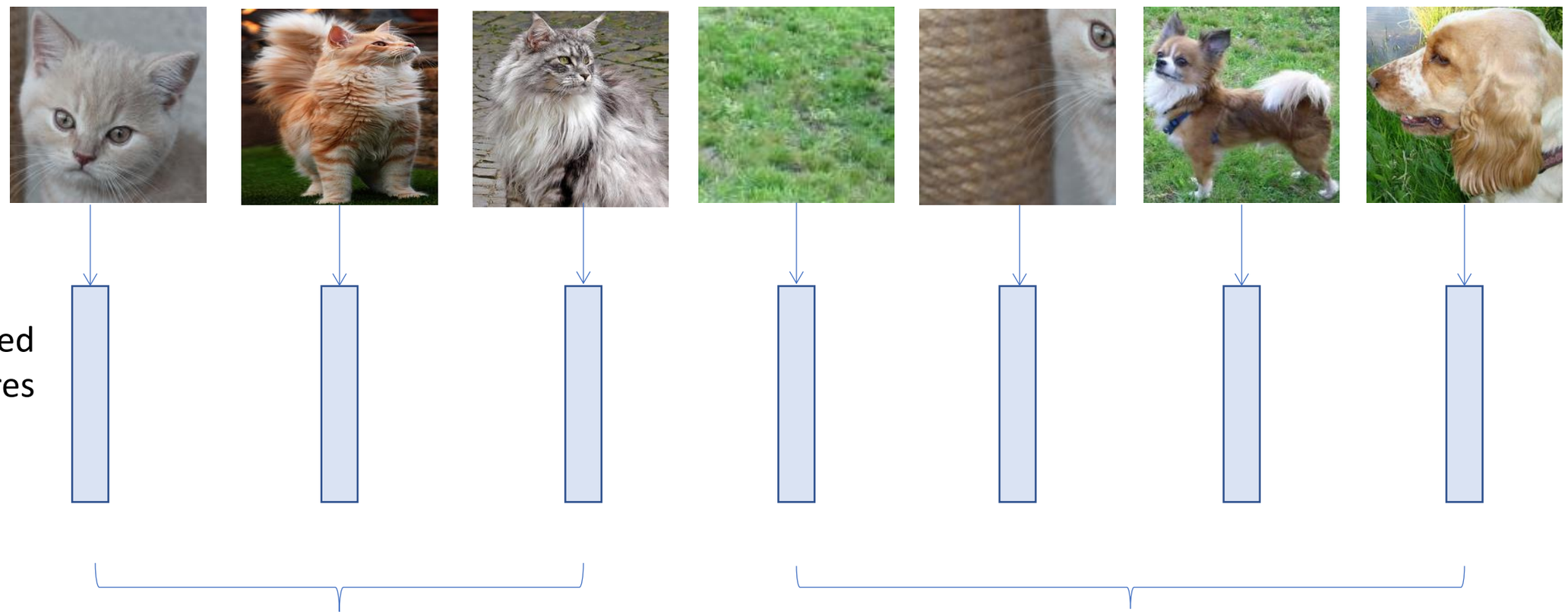


Image: http://www.robots.ox.ac.uk/~vgg/data/pets/

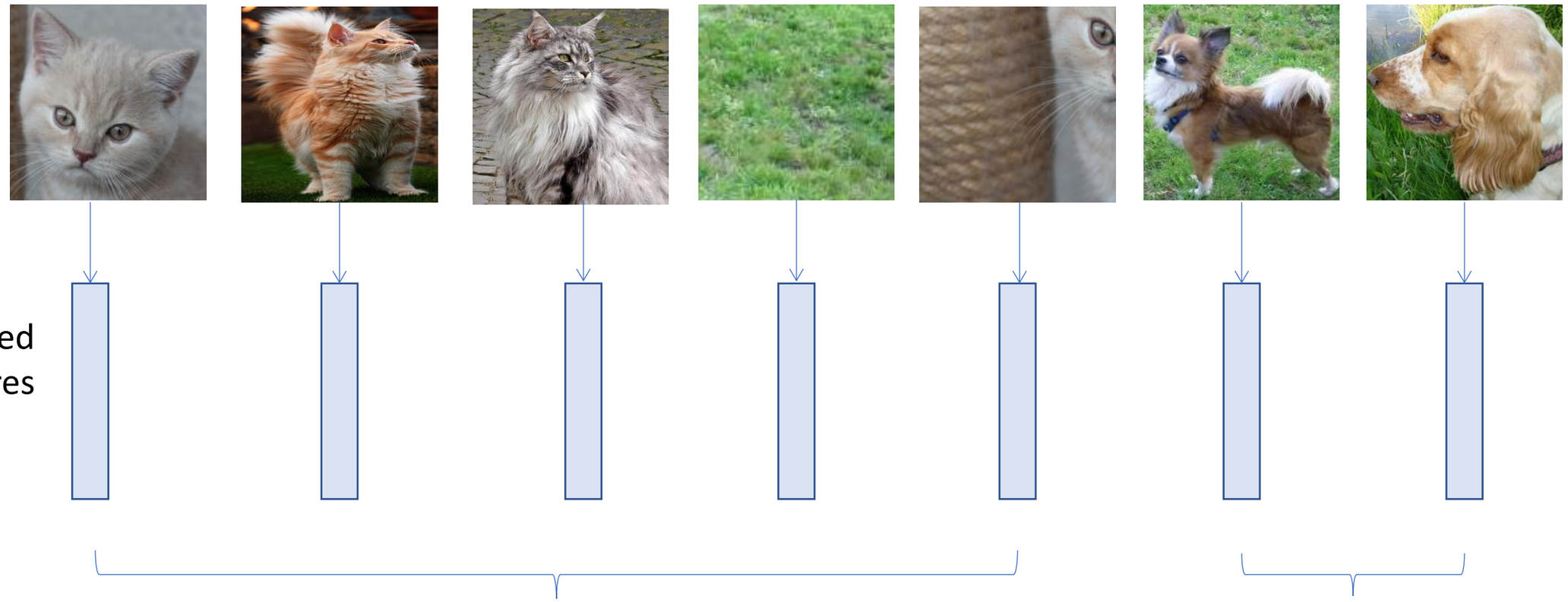
R-CNN Training

- 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

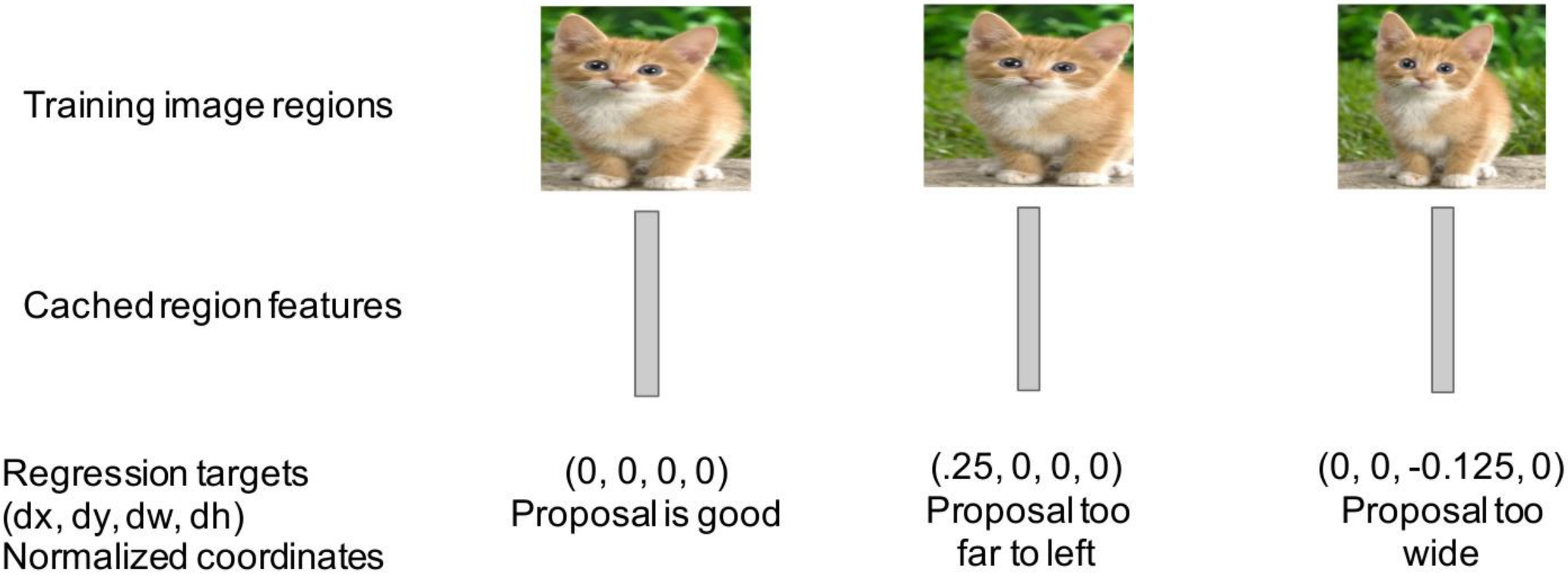


Image from Fei-fei Li et. al.

What's wrong with slow R-CNN?

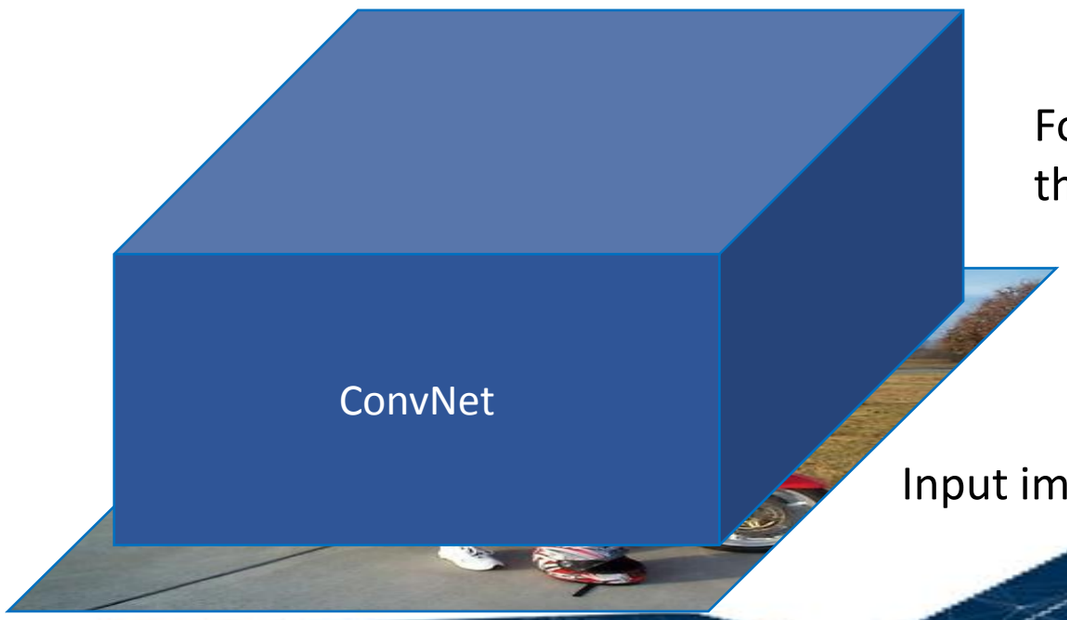
- Ad-hoc training objectives
  - Fine-tune network with softmax classifier
  - Train post-hoc linear SVM
  - Train post-hoc bonding-box regression
- Training is slow, takes a lot of disk space (for feature chaching)
- Inference (detection) is slow
  - 47s/image with VGG16
  - Fixed by SPPnet

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

The Spatial Pyramid Pooling Network (SPP-Net)

Input image

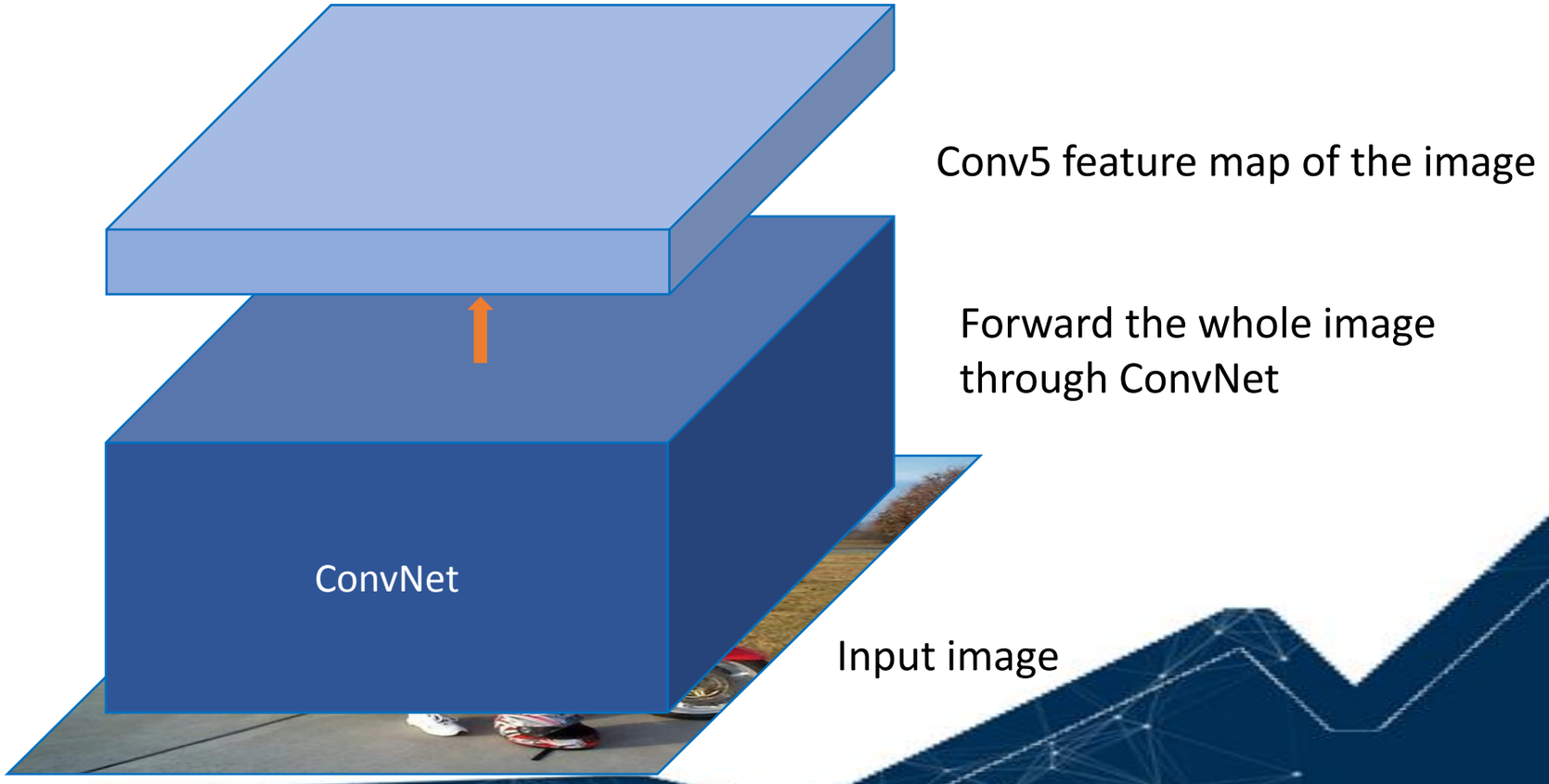
The Spatial Pyramid Pooling Network (SPP-Net)



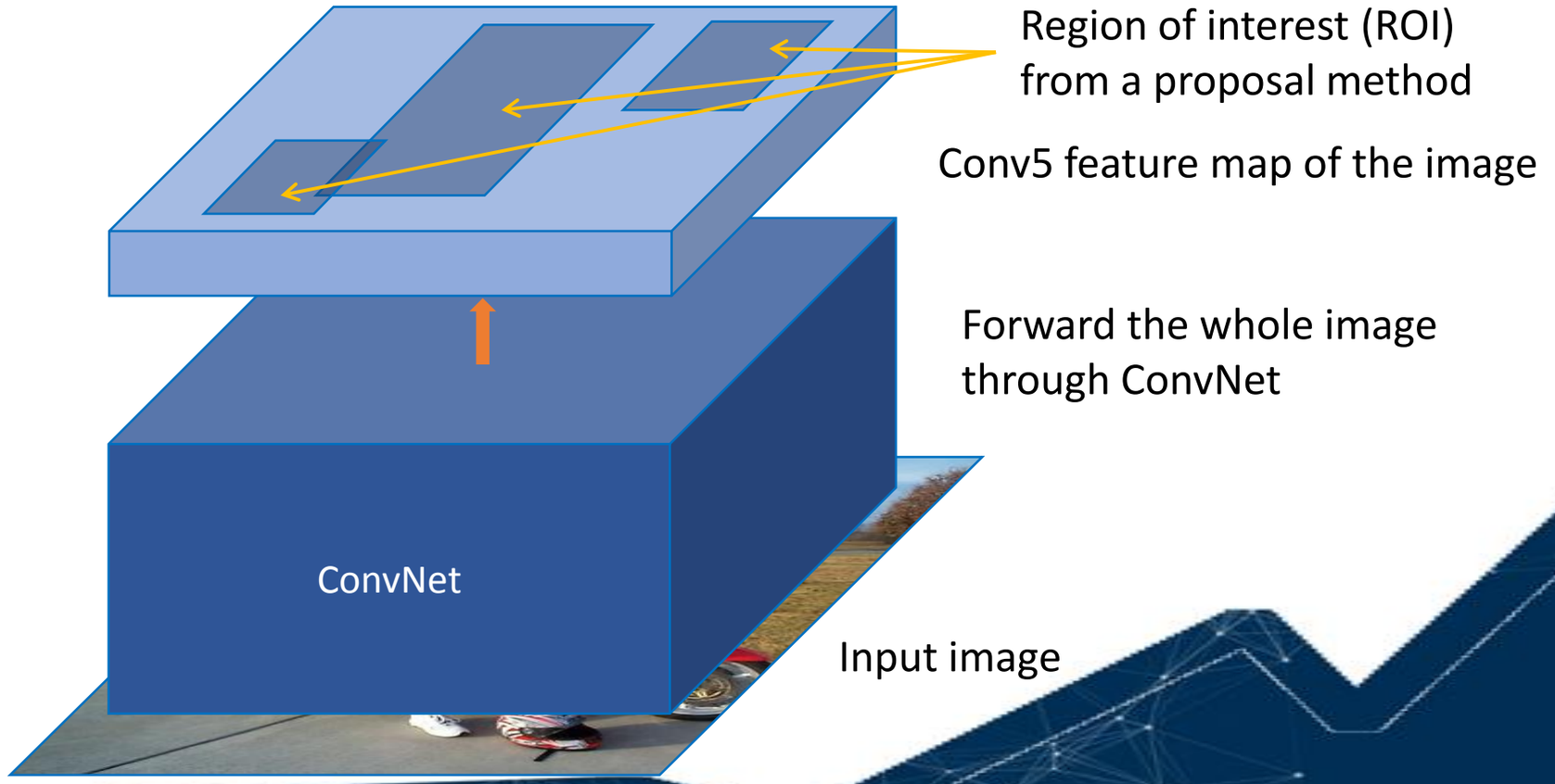
Forward the whole image through ConvNet

Input image

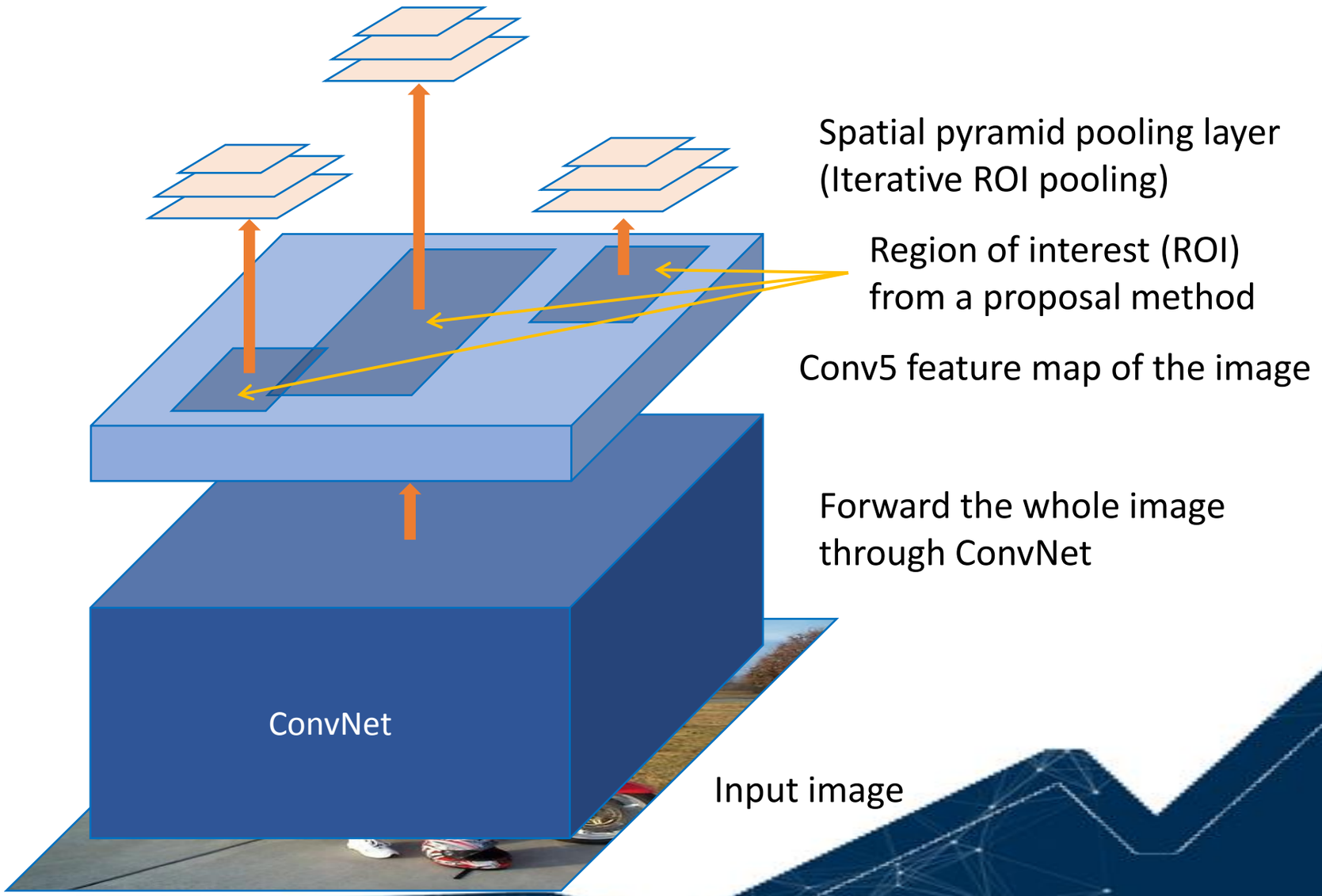
The Spatial Pyramid Pooling Network (SPP-Net)



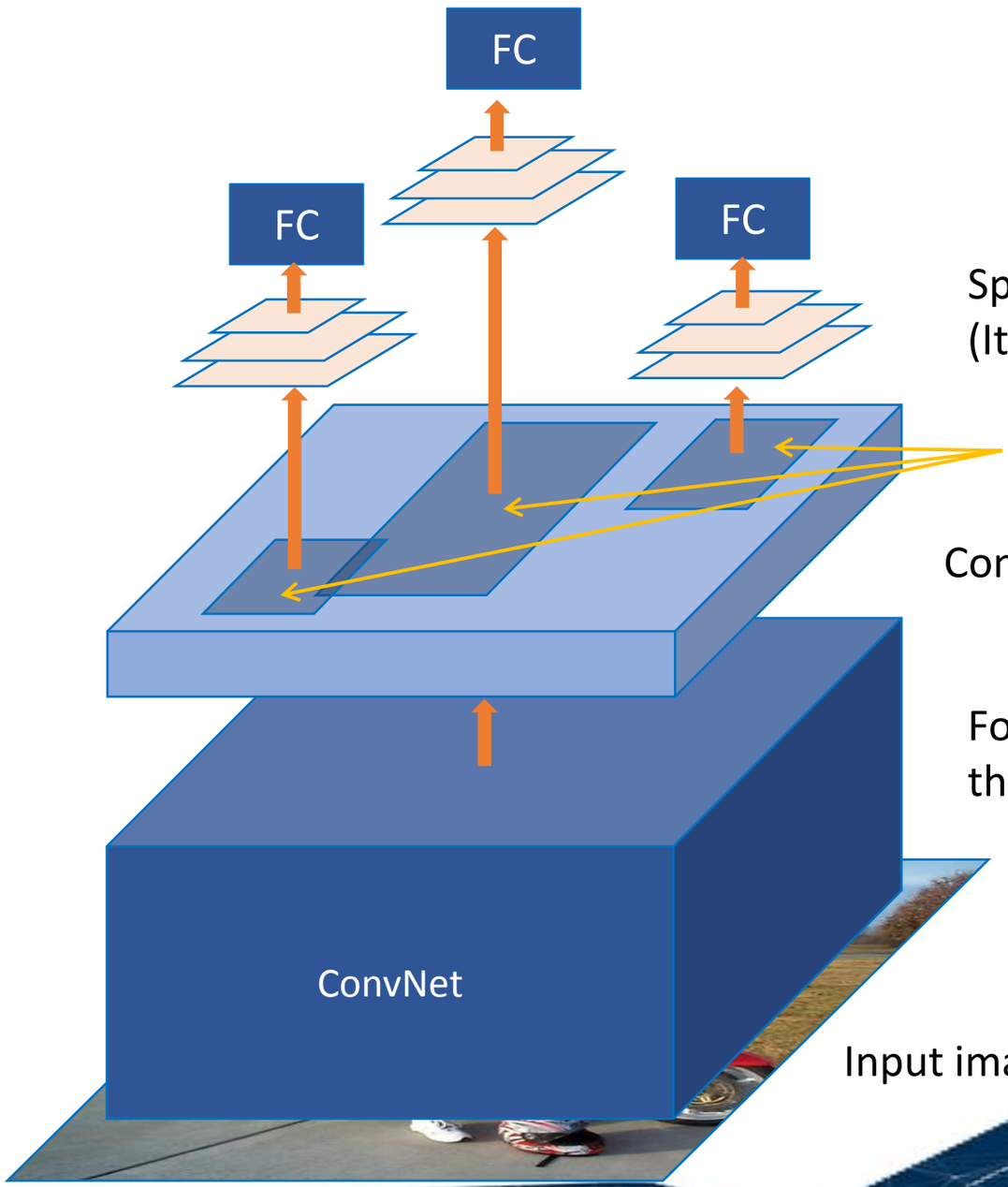
The Spatial Pyramid Pooling Network (SPP-Net)



The Spatial Pyramid Pooling Network (SPP-Net)



The Spatial Pyramid Pooling Network (SPP-Net)



Spatial pyramid pooling layer
(Iterative ROI pooling)

Region of interest (ROI)
from a proposal method

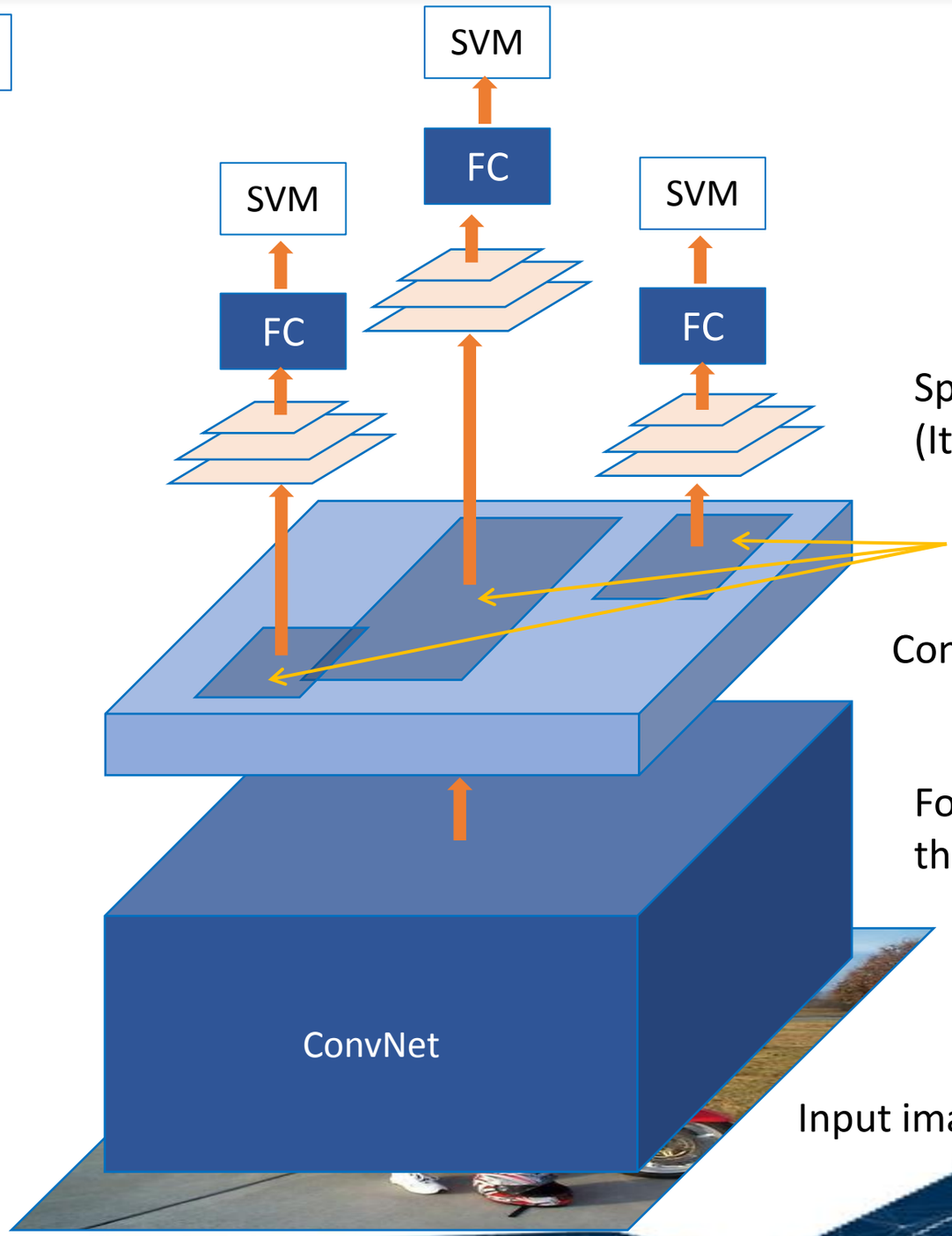
Conv5 feature map of the image

Forward the whole image
through ConvNet

Input image

The Spatial Pyramid Pooling Network (SPP-Net)

Post hoc component



Classify using SVM

Spatial pyramid pooling layer
(Iterative ROI pooling)

Region of interest (ROI)
from a proposal method

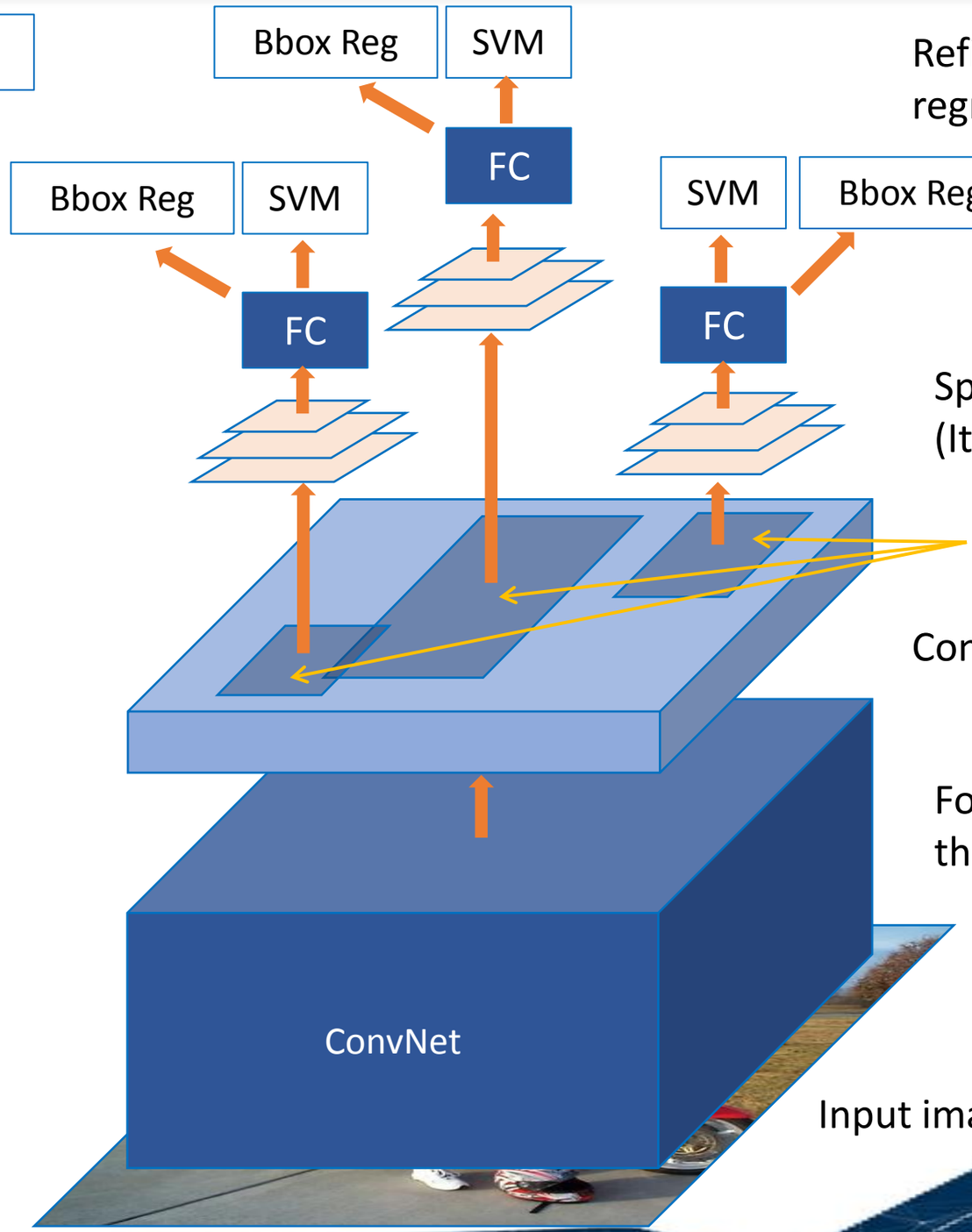
Conv5 feature map of the image

Forward the whole image
through ConvNet

Input image

The Spatial Pyramid Pooling Network (SPP-Net)

Post hoc component



Refine the position using a regressor

Classify using SVM

Spatial pyramid pooling layer (Iterative ROI pooling)

Region of interest (ROI) from a proposal method

Conv5 feature map of the image

Forward the whole image through ConvNet

Input image

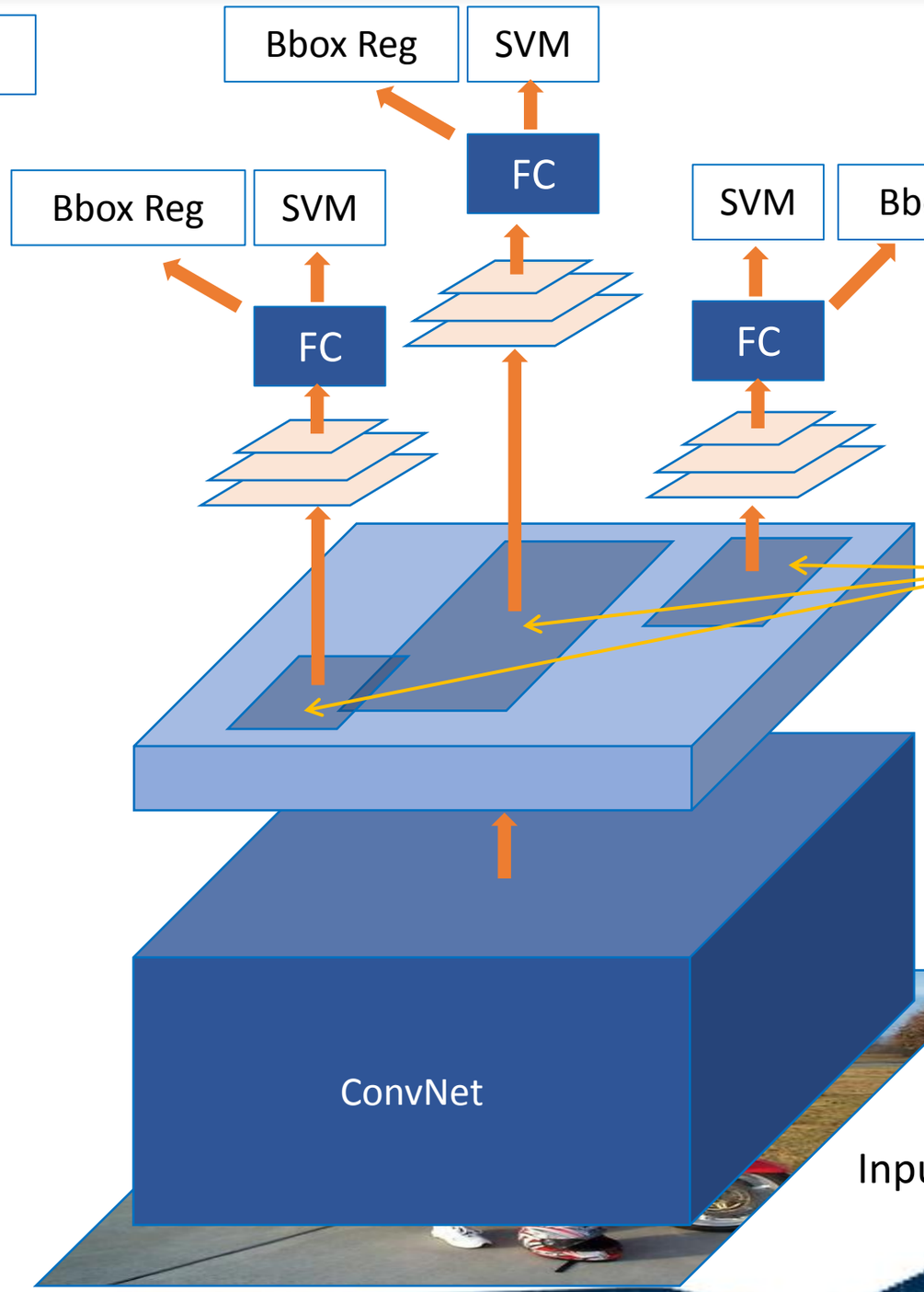
Advantage of the SPP-Net

Post hoc component

Faster computation speed!

Region-wise computation

Image-wise computation



Refine the position using a regressor

Classify using SVM

Spatial pyramid pooling layer (Iterative ROI pooling)

Region of interest (ROI) from a proposal method

Conv5 feature map of the image

Forward the whole image through ConvNet

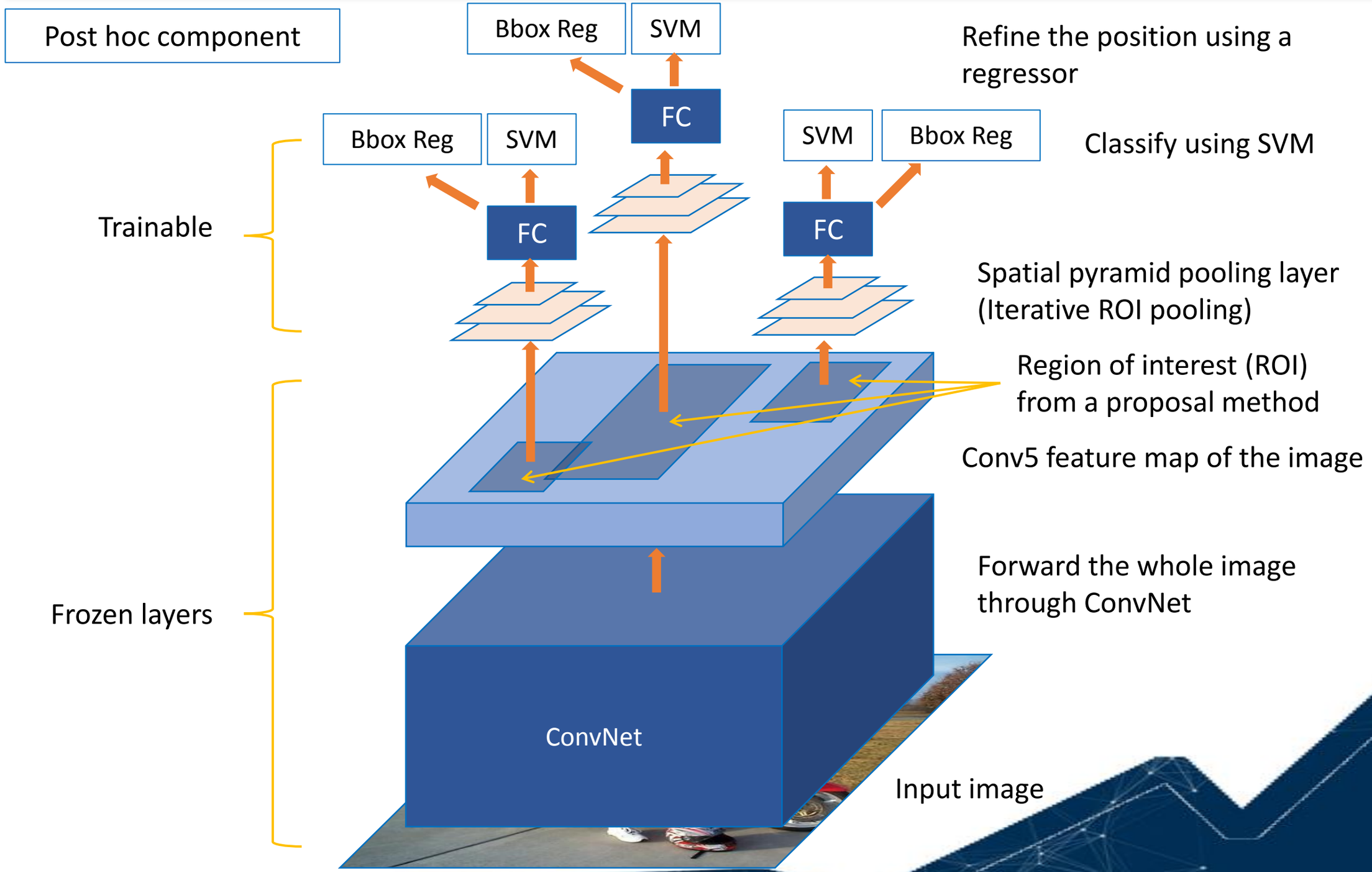
Input image

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 the SPP-Net



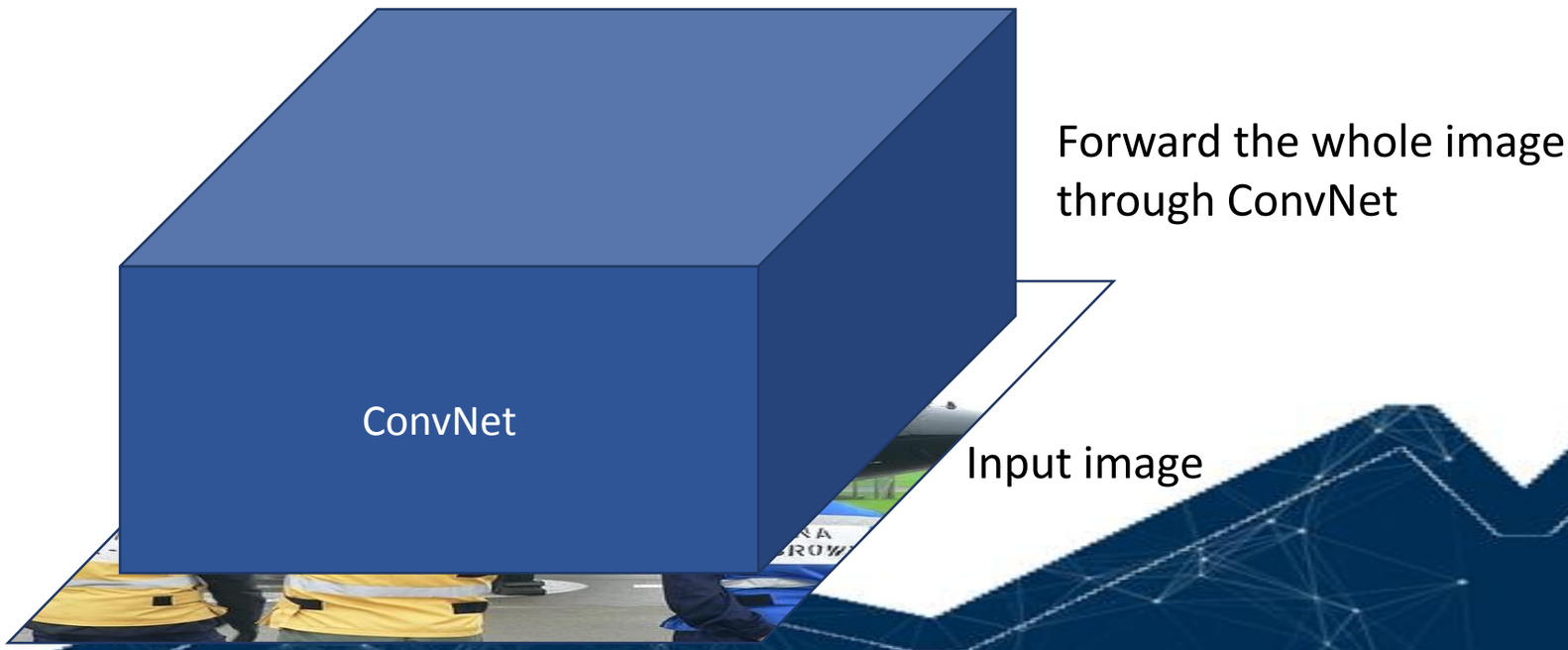
Fast R-CNN

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

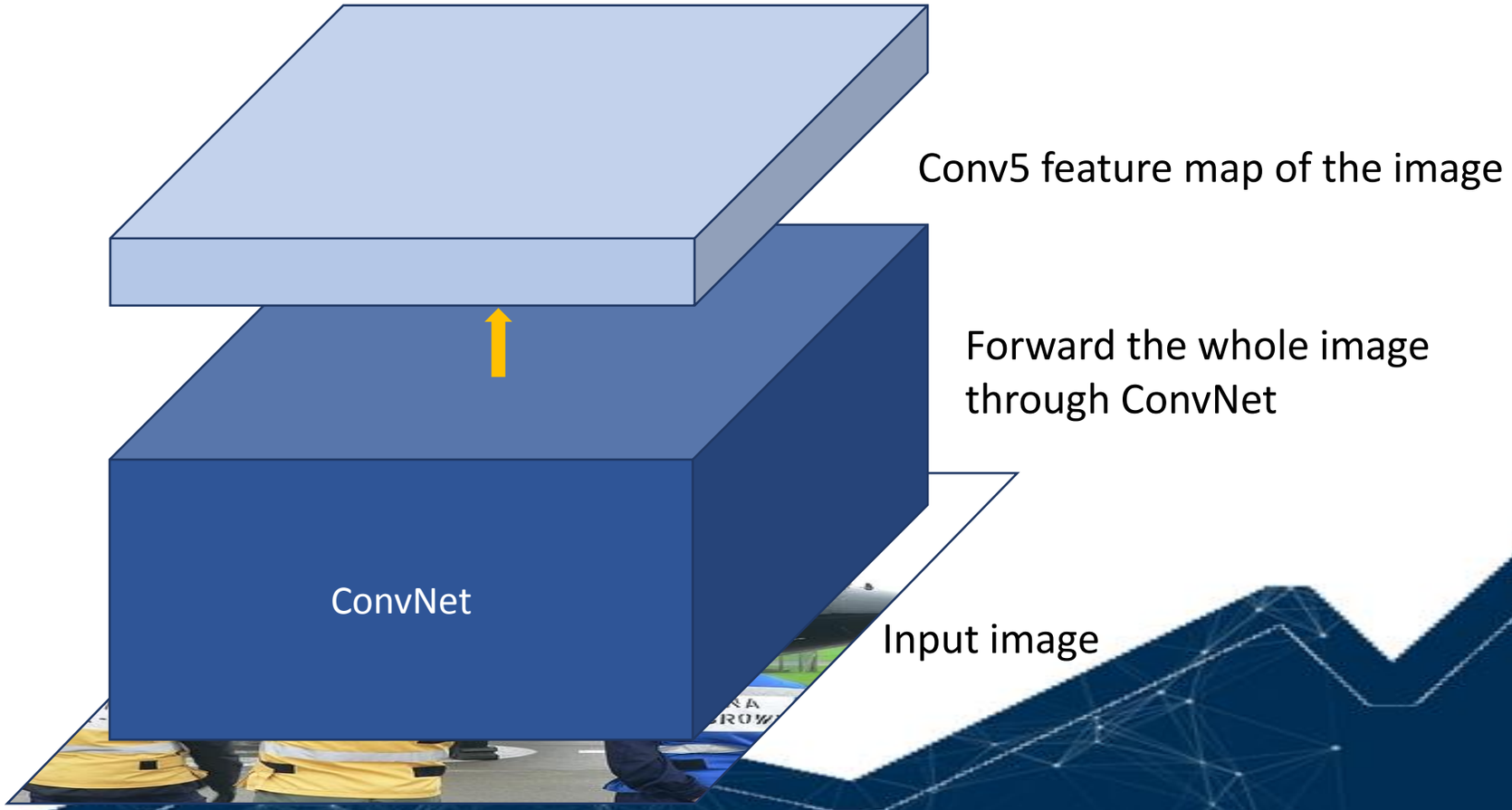
Fast R-CNN

Input image

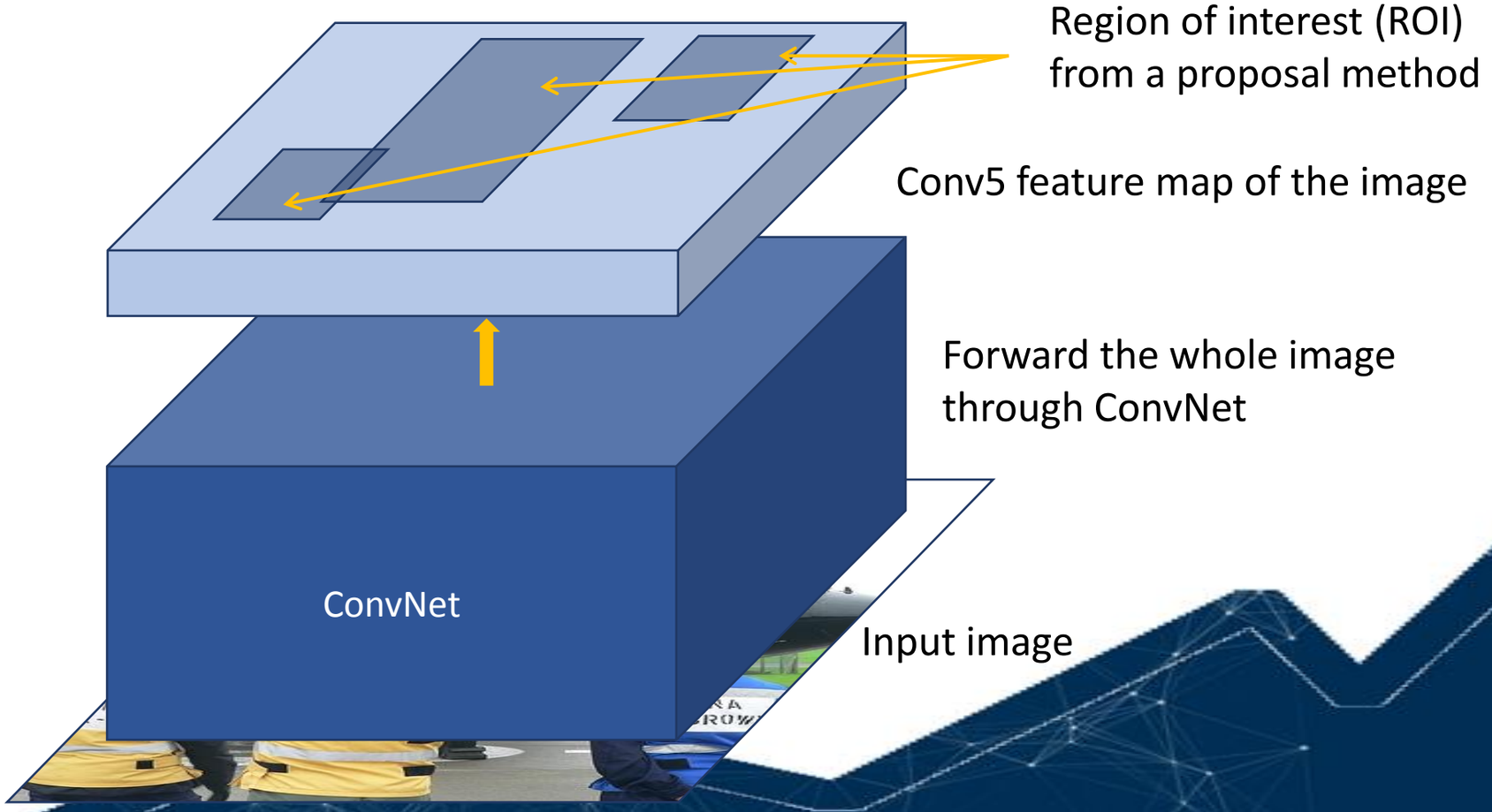
Fast R-CNN


ConvNet
Forward the whole image through ConvNet
Input image

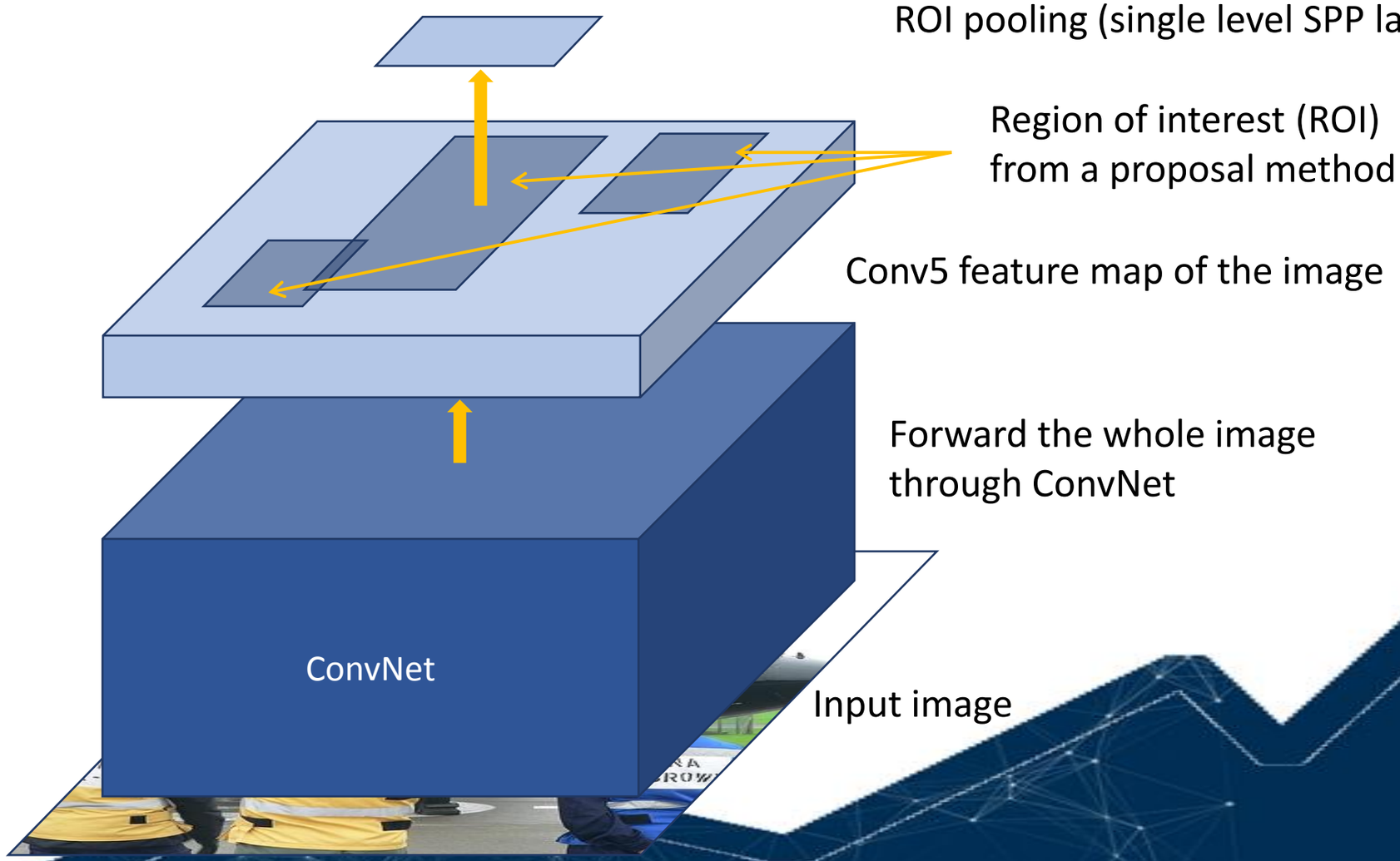
Fast R-CNN


Conv5 feature map of the image
Forward the whole image through ConvNet
ConvNet
Input image

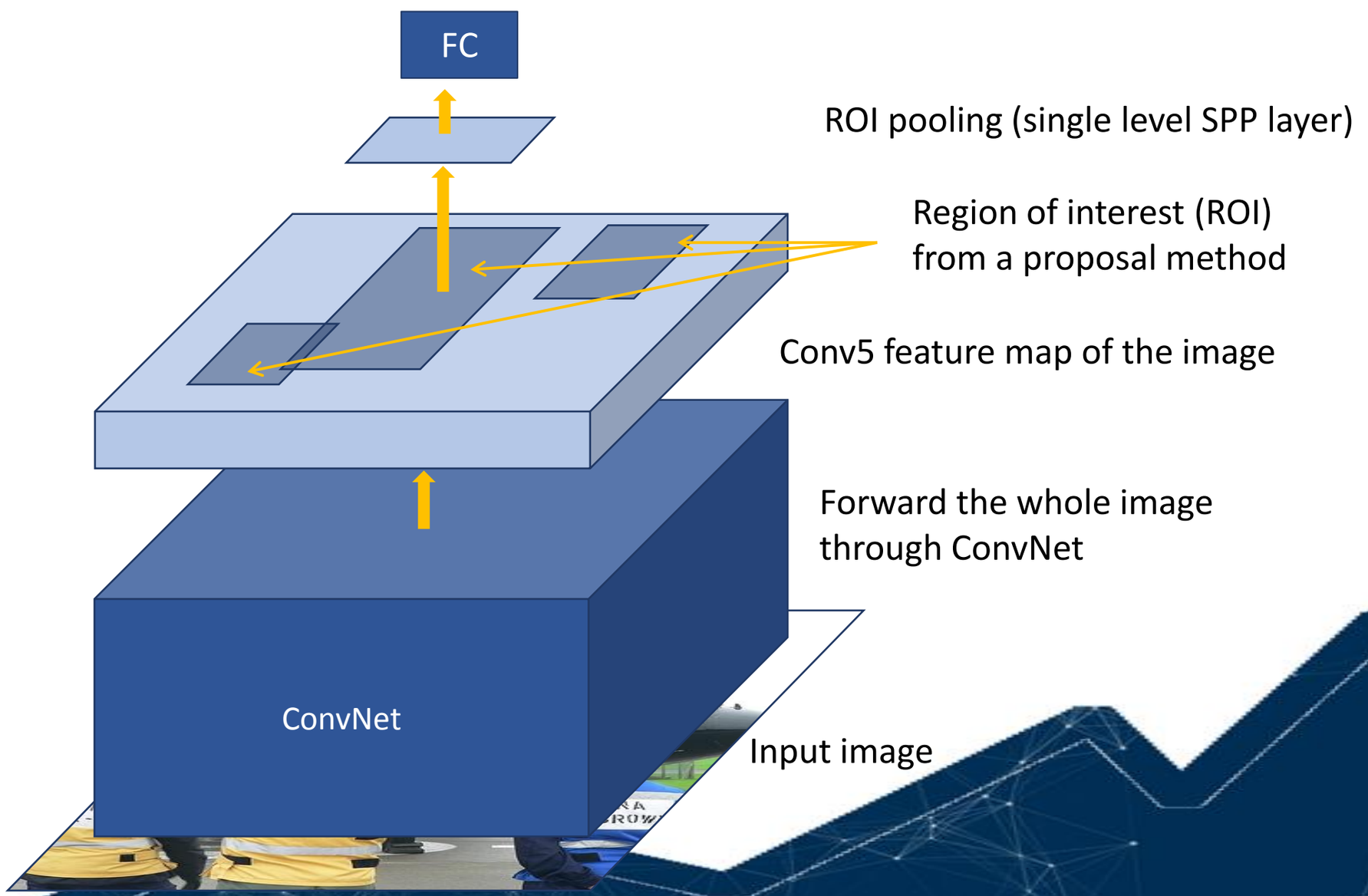
Fast R-CNN


Region of interest (ROI) from a proposal method
Conv5 feature map of the image
Forward the whole image through ConvNet
ConvNet
Input image

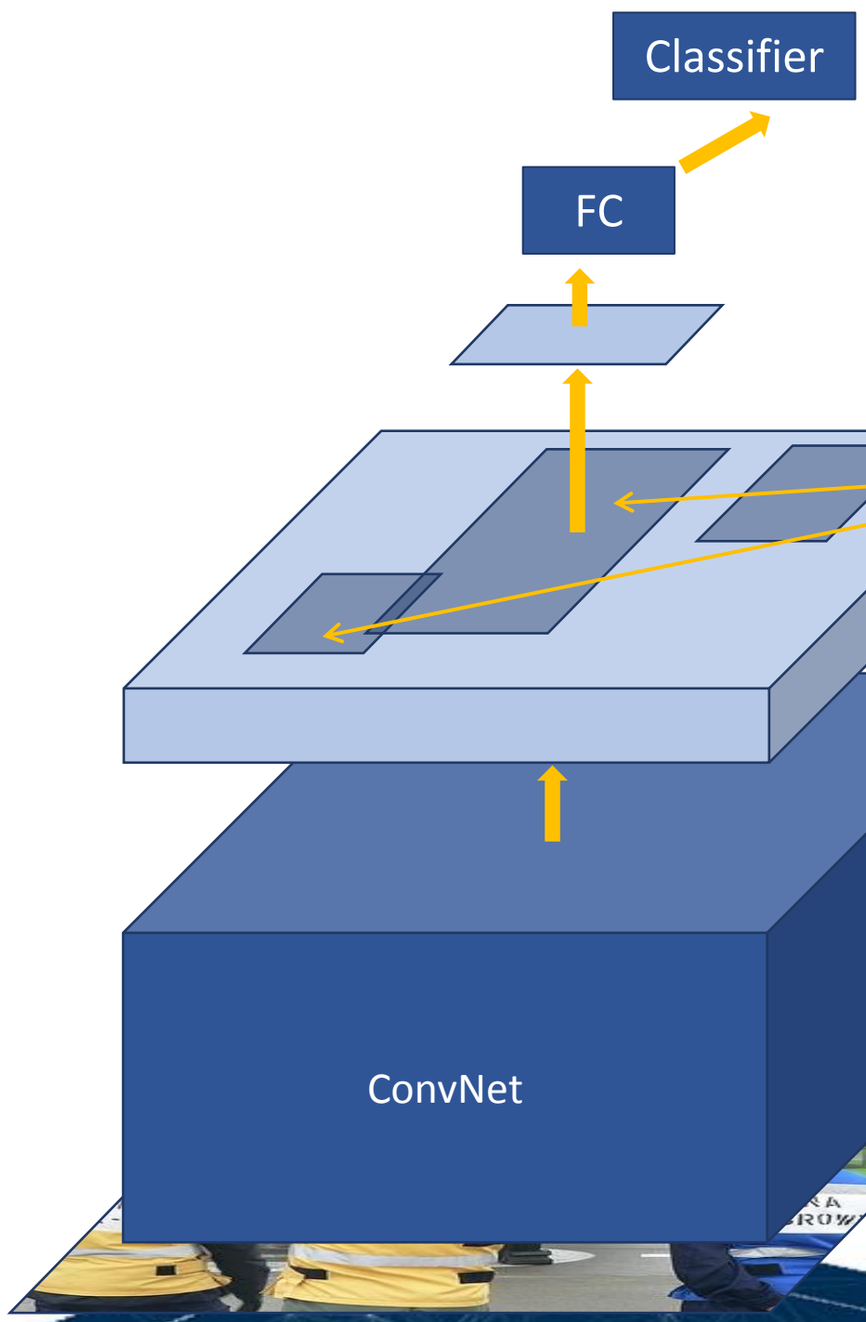
Fast R-CNN


ROI pooling (single level SPP layer)
Region of interest (ROI) from a proposal method
Conv5 feature map of the image
Forward the whole image through ConvNet
ConvNet
Input image

Fast R-CNN


FC
ROI pooling (single level SPP layer)
Region of interest (ROI) from a proposal method
Conv5 feature map of the image
Forward the whole image through ConvNet
ConvNet
Input image

Fast R-CNN



Classify using linear+softmax layer

ROI pooling (single level SPP layer)

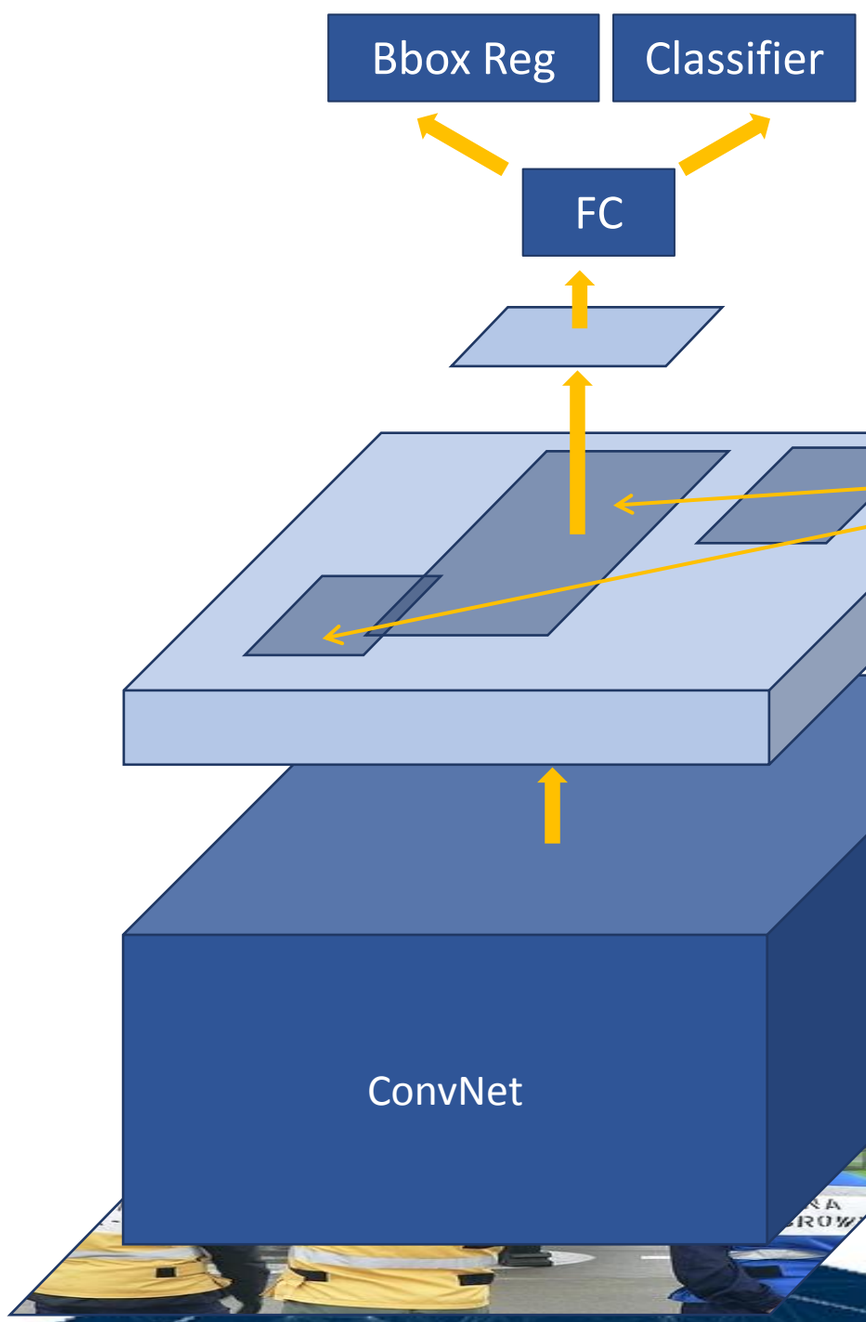
Region of interest (ROI) from a proposal method

Conv5 feature map of the image

Forward the whole image through ConvNet

Input image

Fast R-CNN

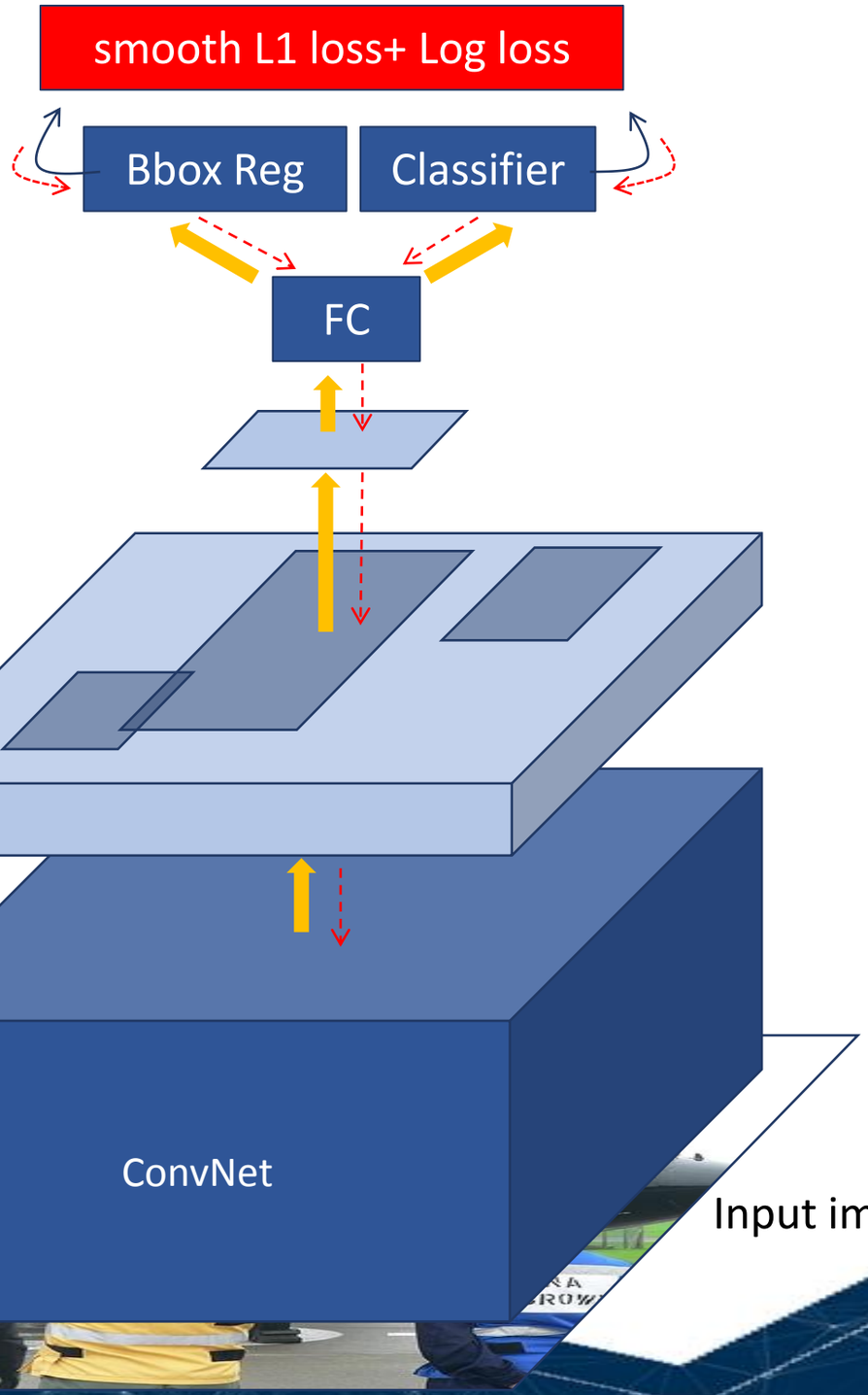


Regressor using linear layer

Classify using linear+softmax layer

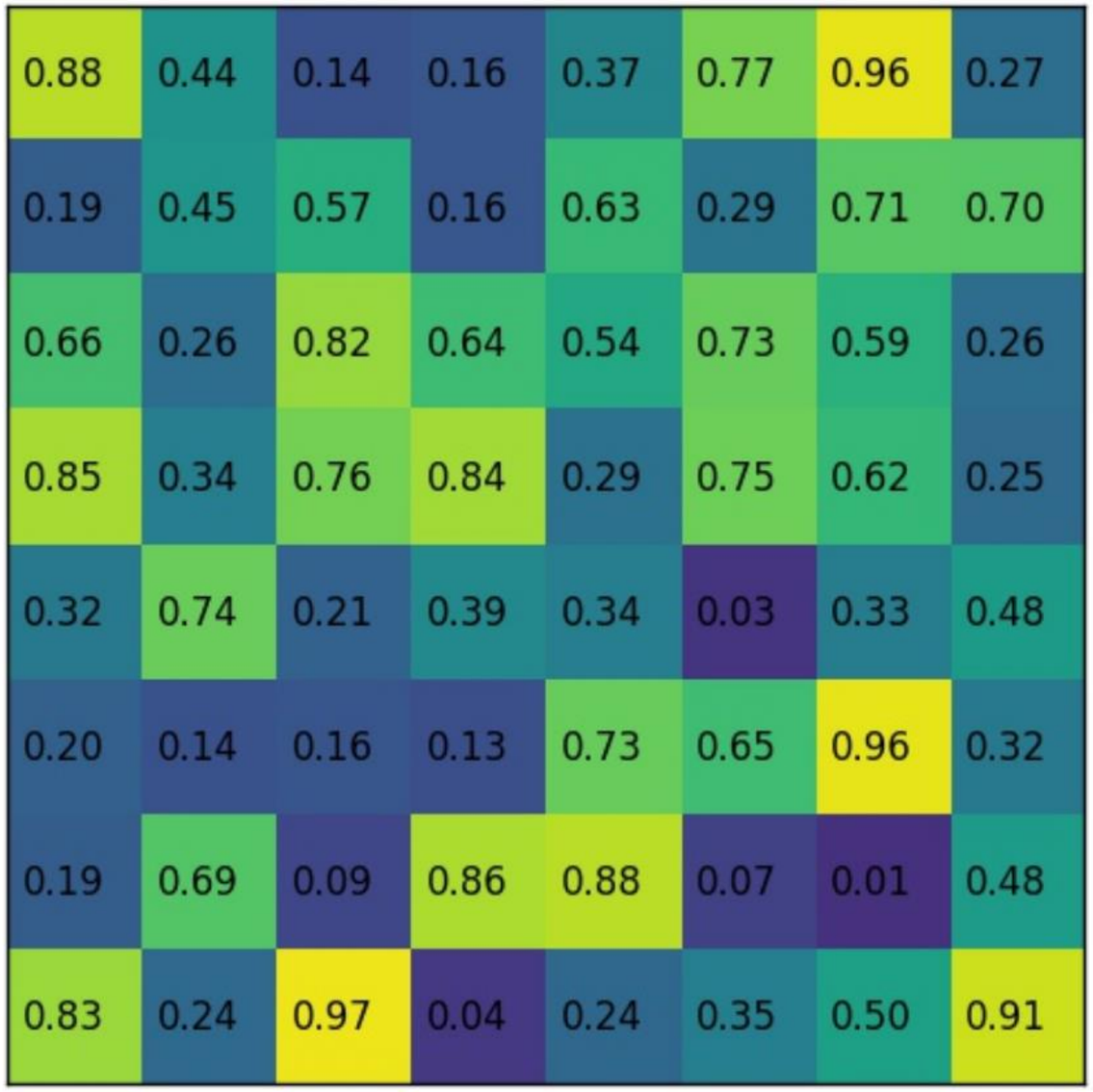
ROI pooling (single level SPP layer)

Fast R-CNN at Training Time



End to end trainable by back propagation

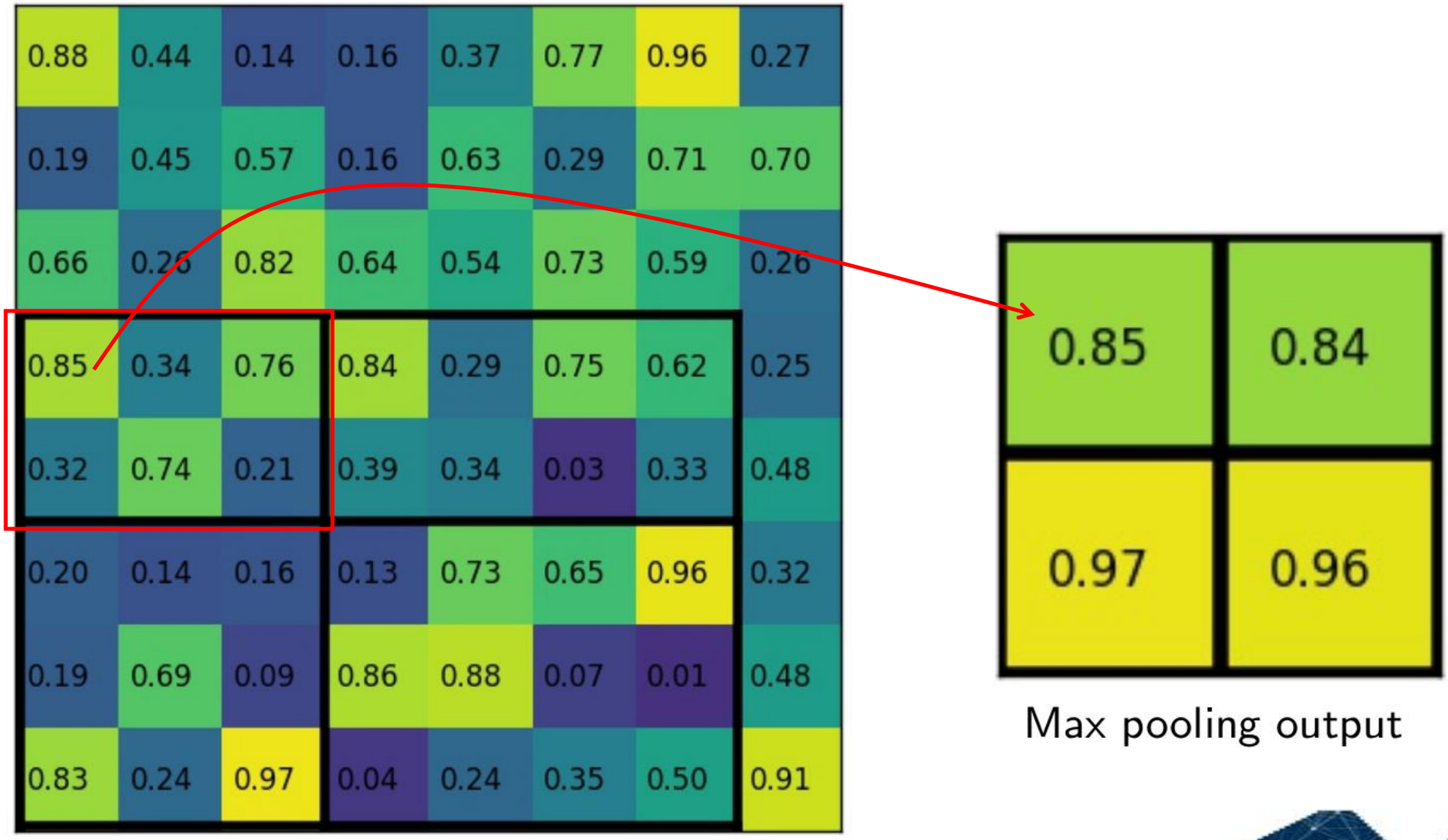
ROI Pooling (1/2)



Input activation

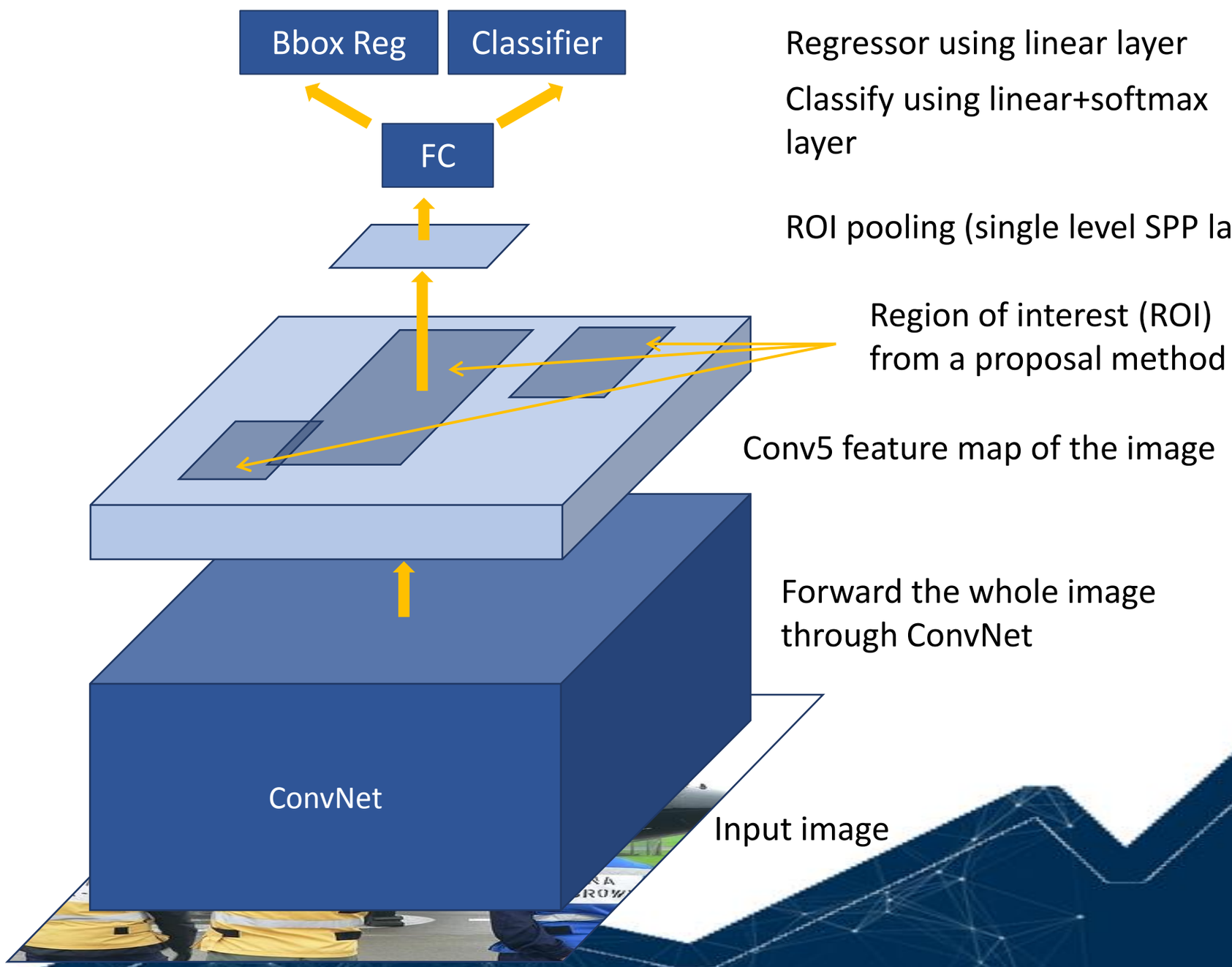
source : https://deepsense.io/region-of-interest-pooling-explained

ROI Pooling (2/2)



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

Fast R-CNN to Faster R-CNN

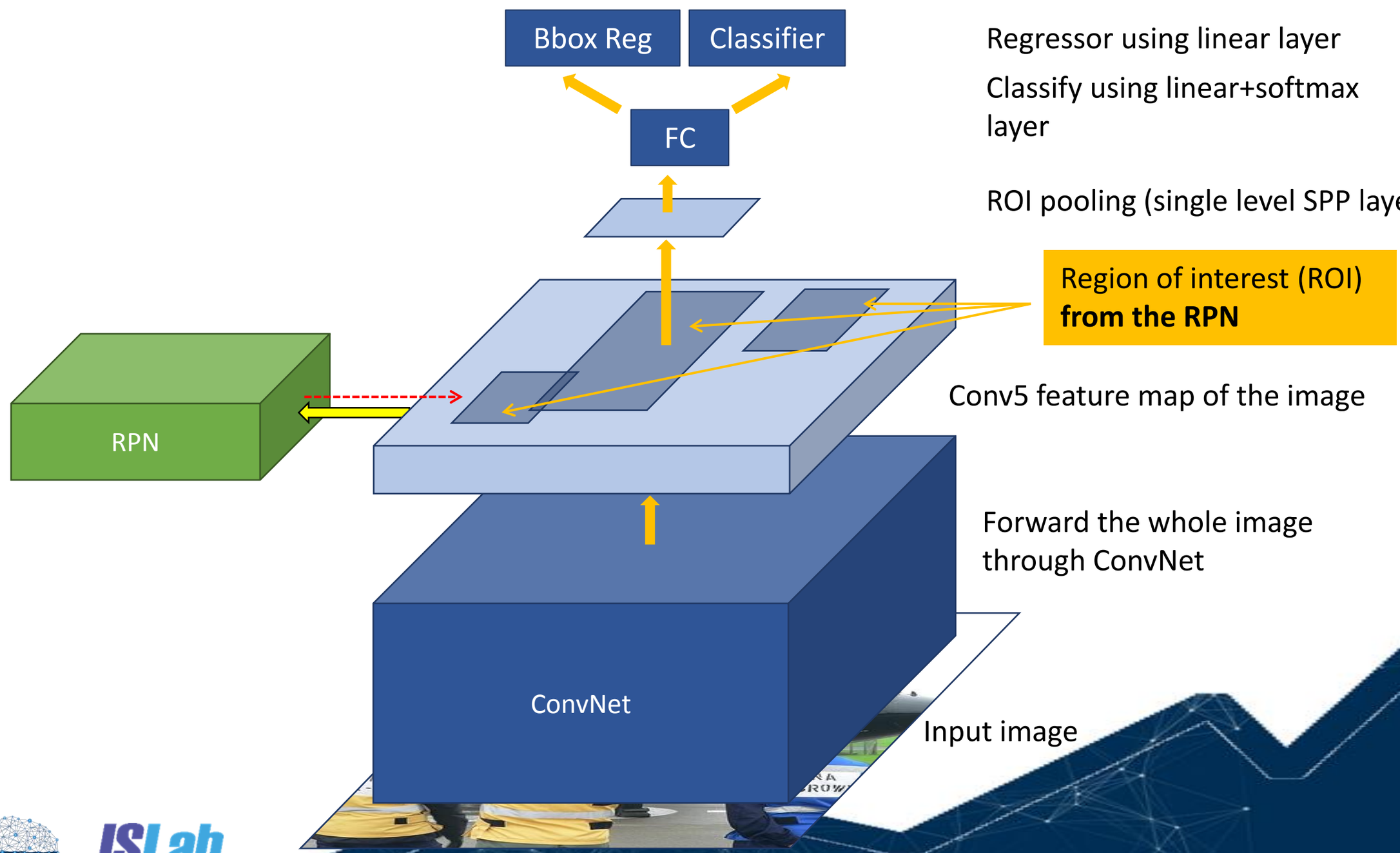


Regressor using linear layer

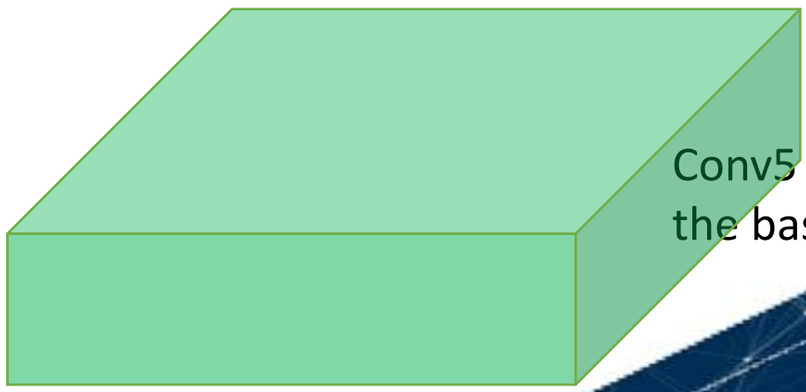
Classify using linear+softmax layer

ROI pooling (single level SPP layer)

Fast R-CNN to Faster R-CNN



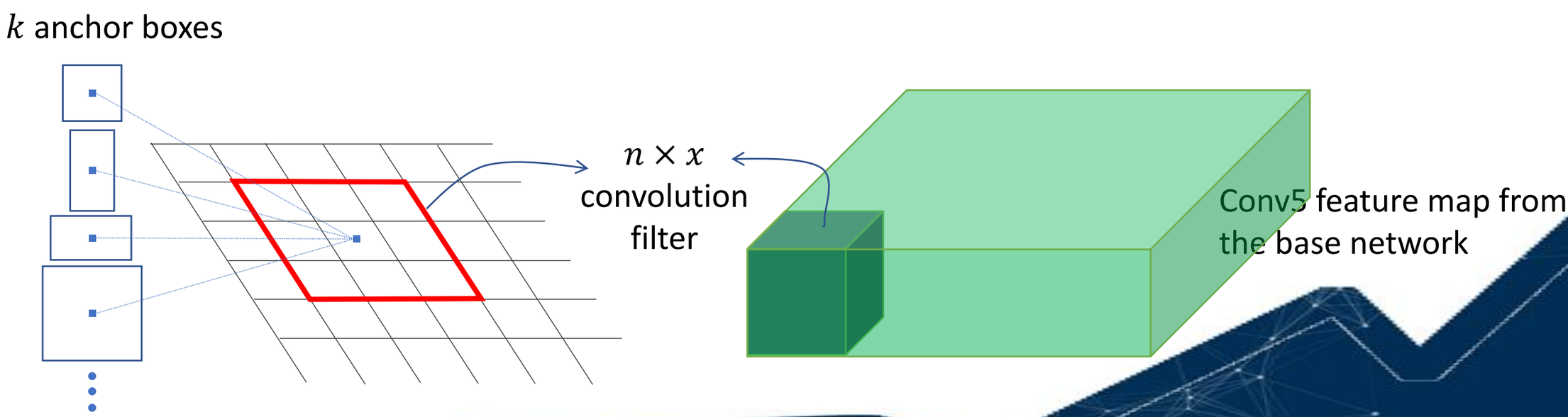
The RPN



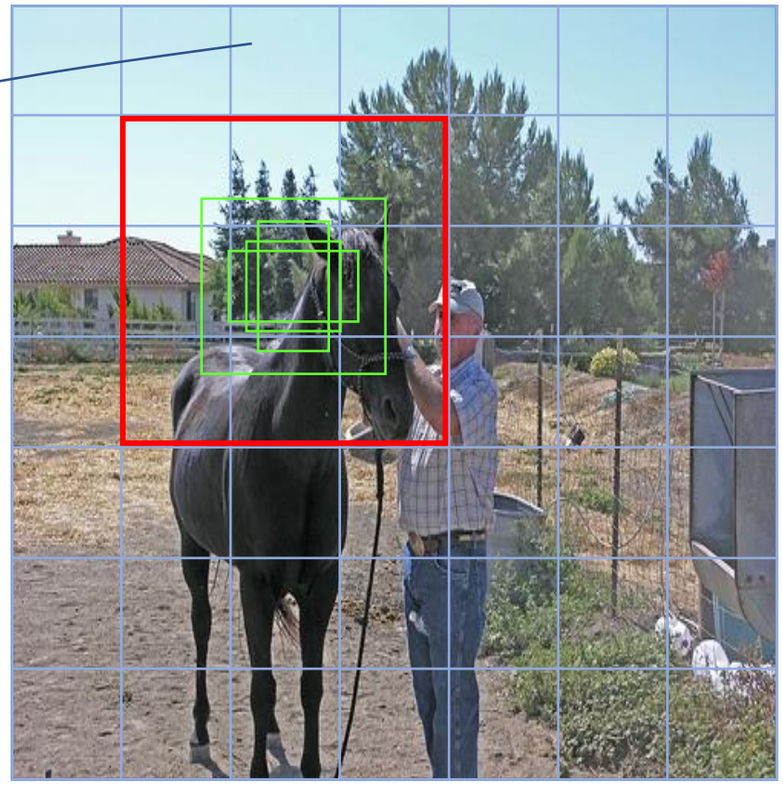
Conv5 feature map from the base network

The RPN

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

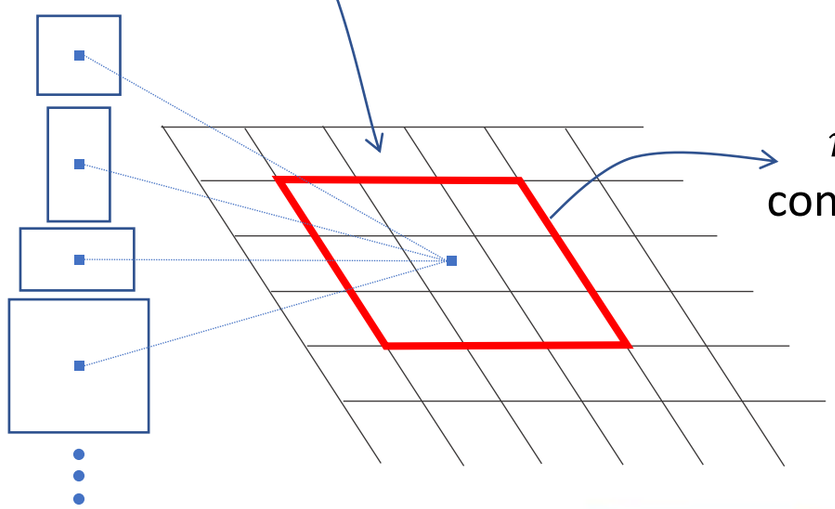


The RPN – What the Anchor Boxes are?

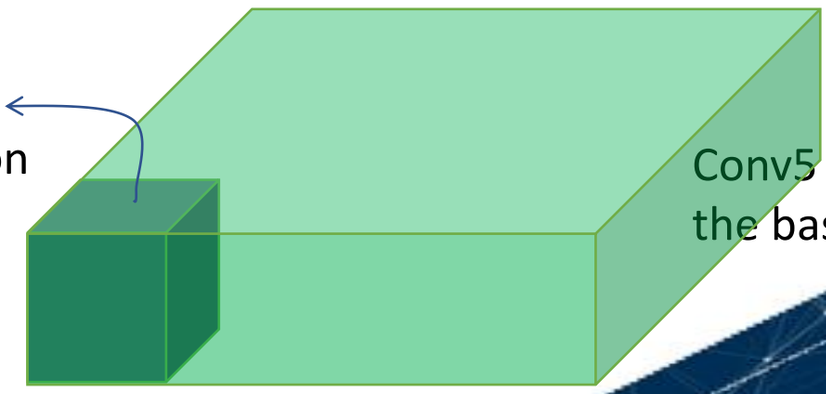


Each cell corresponds to 1 “location” in the feature map

k anchor boxes



$n \times x$ convolution filter

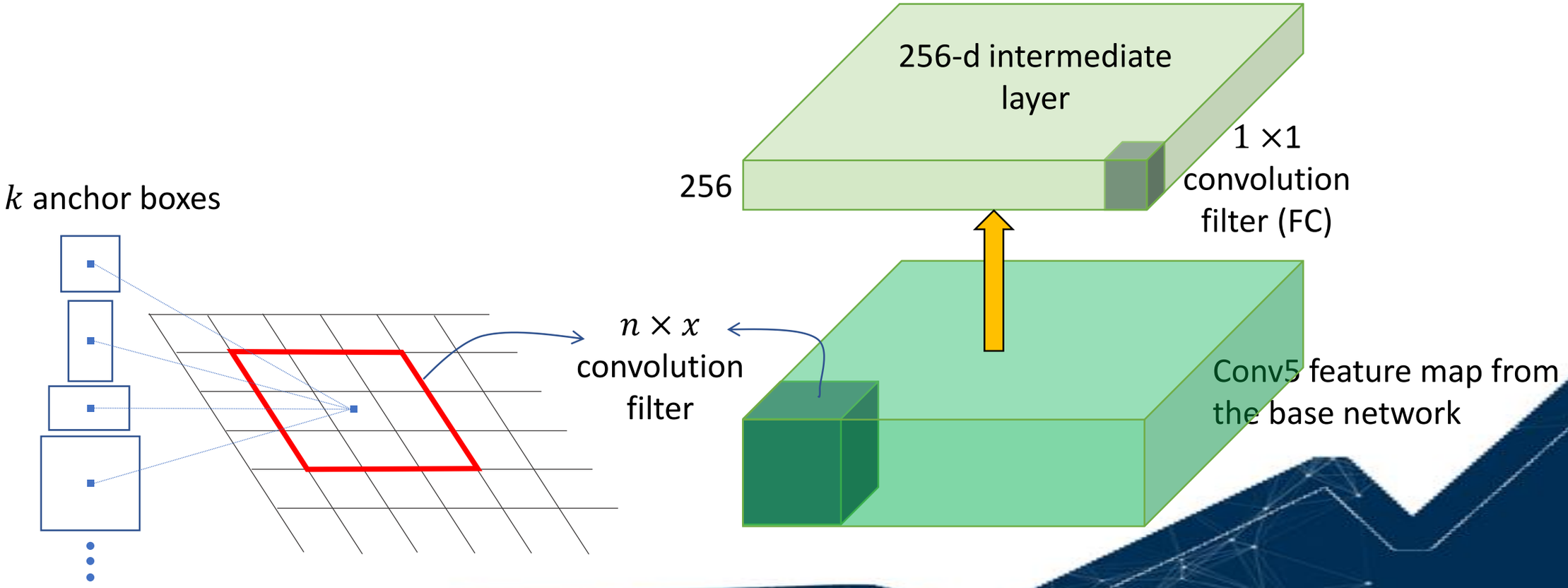


Conv5 feature map from the base network

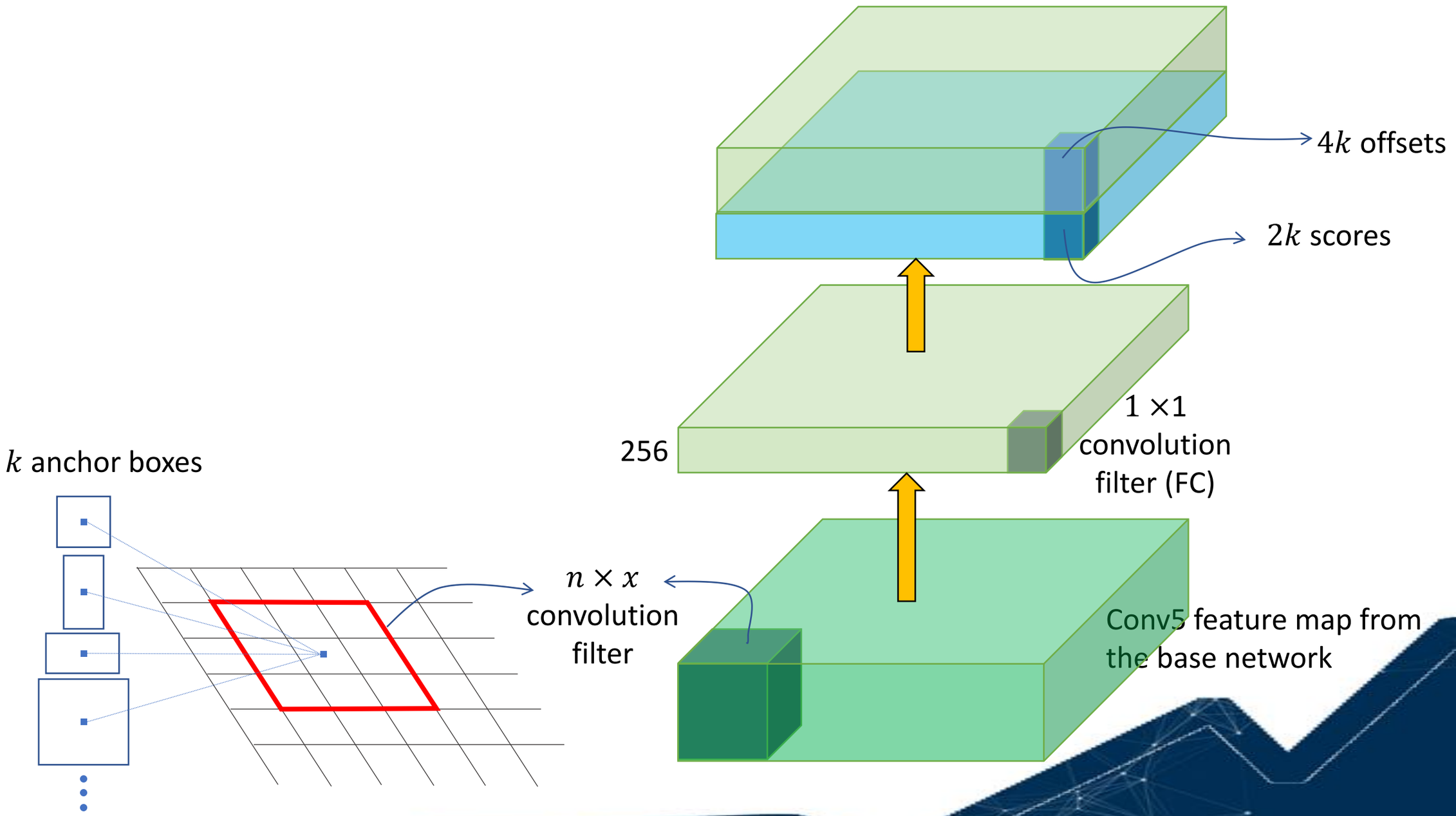
Note: number of cell in this picture is not represent the “real case” (for illustration purpose only)

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

The RPN


256-d intermediate layer
256
1 ×1 convolution filter (FC)
k anchor boxes
n × x convolution filter
Conv5 feature map from the base network

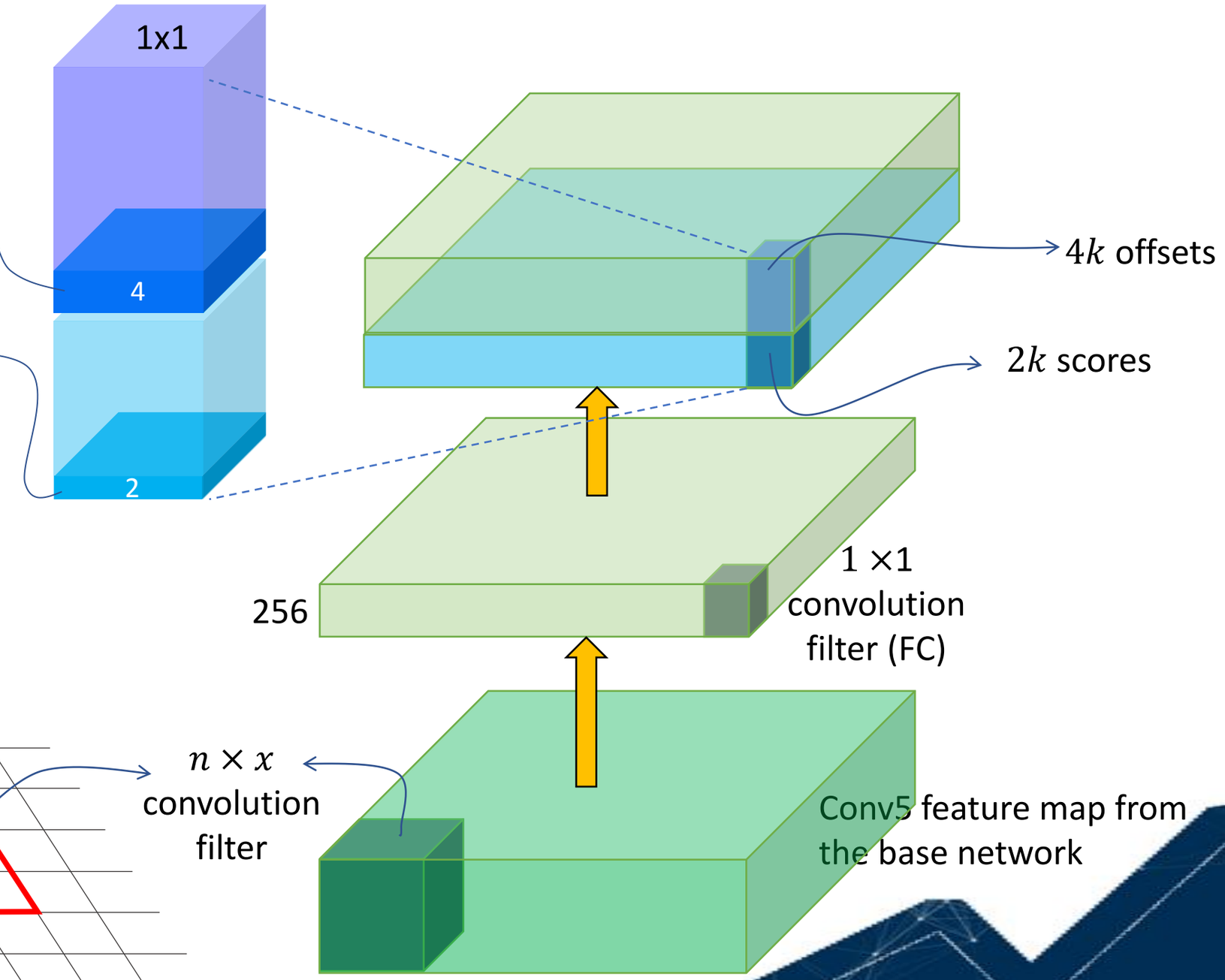
The RPN


$4k$ offsets
$2k$ scores
1×1 convolution filter (FC)
256
k anchor boxes
$n \times x$ convolution filter
Conv5 feature map from the base network

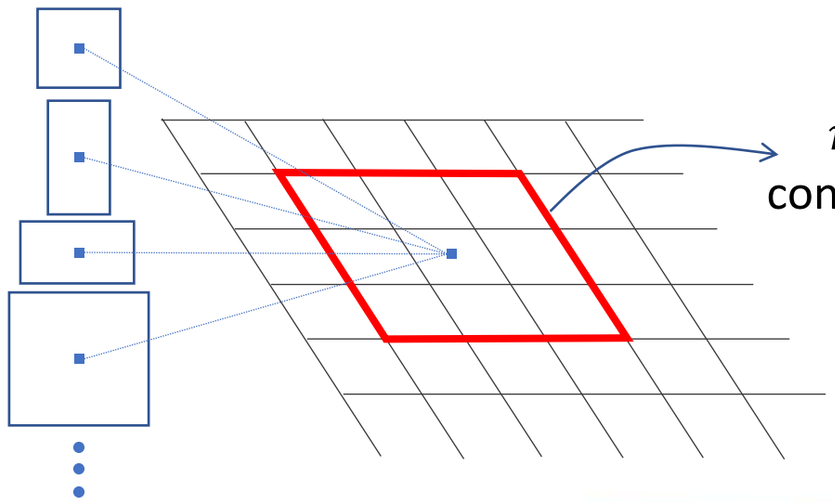
The RPN

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! (Or ISSonDL Server)

tinyurl.com/hsi18-colab

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/hsi/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 = 'hsi/'

# 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('hsi/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)
```

Creating Few Wrappers

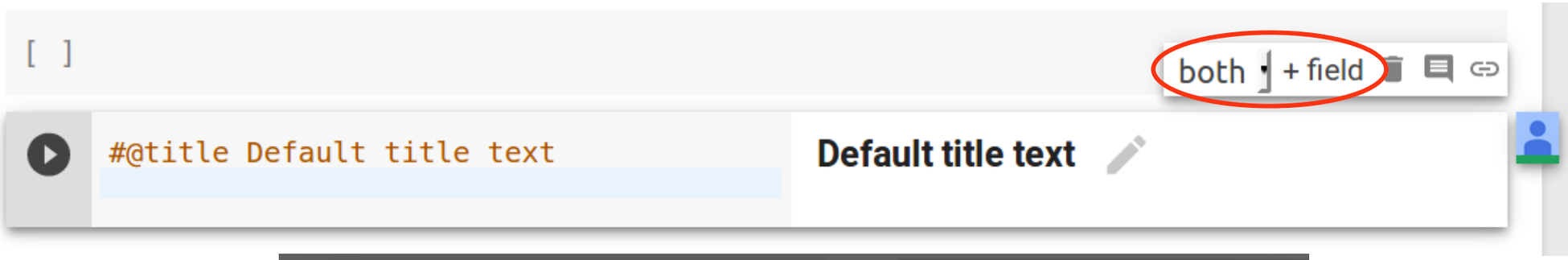
```
def run_inference_for_single_image(image, graph):
  with graph.as_default():
    with tf.Session() as sess:
      # Get handles to input and output tensors
      ops = tf.get_default_graph().get_operations()
      all_tensor_names = {output.name for op in ops for output in op.outputs}
      tensor_dict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

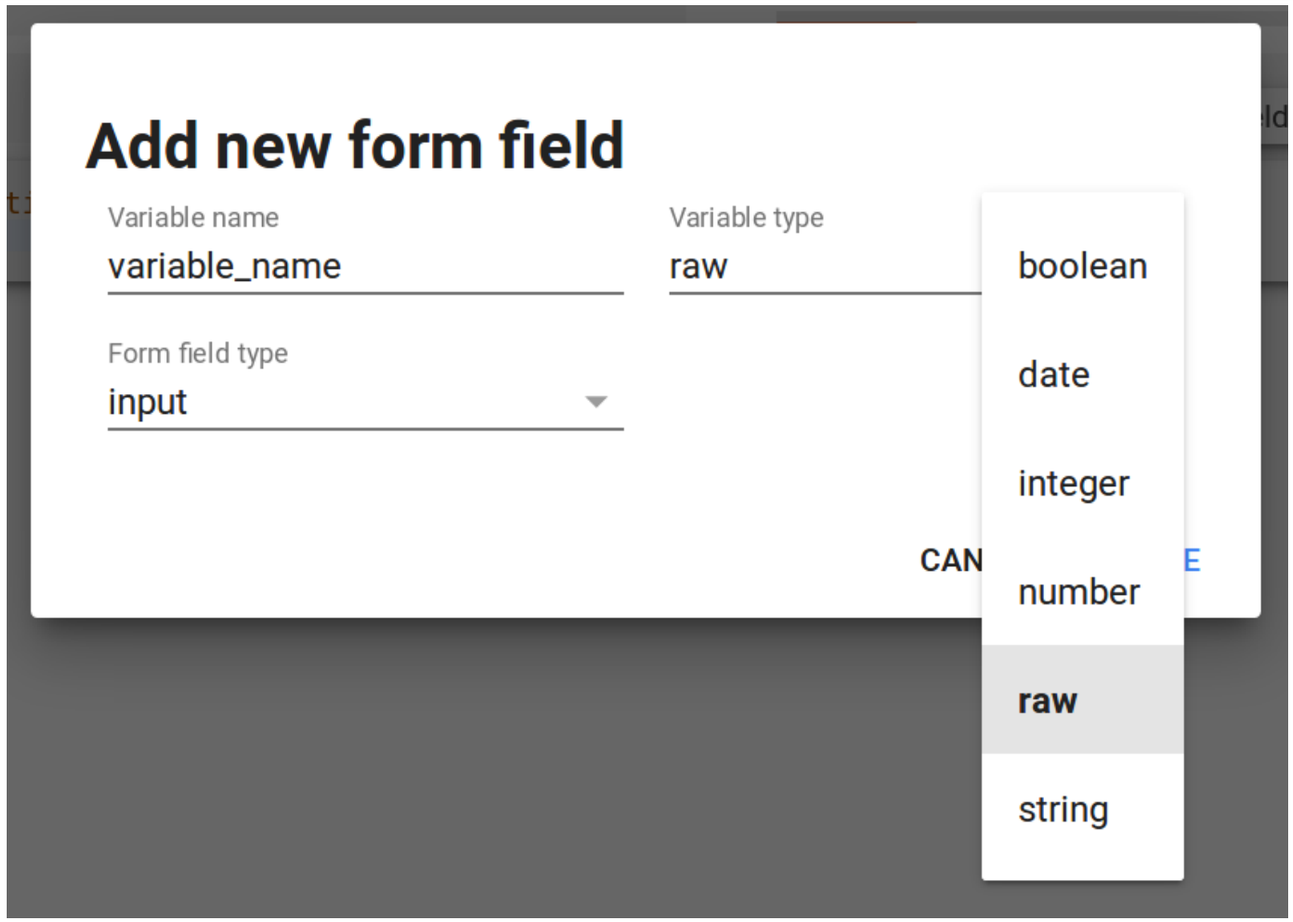
```
# we will load an image from URL and then convert it to numpy array
def load_image_into_numpy_array(URL):
  file = cStringIO.StringIO(urlloader.urlopen(URL).read())
  image = Image.open(file)
  (im_width, im_height) = image.size
  return np.array(image.getdata()).reshape(
      (im_height, im_width,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

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





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

Inference

```
#@title Image URL { run: "auto", vertic
img_url = "http://host.robots.ox.ac.uk/
size = 19 #@param {type:"integer"}

if size <5:
  size = 5

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

both + field

both
code
form

Image URL

img_url: "http://host.robots.ox.a"

size: 19

Edit form attributes

Title text: Image URL

Initial form visibility: last shown (default)

Output height: px

Form width: px

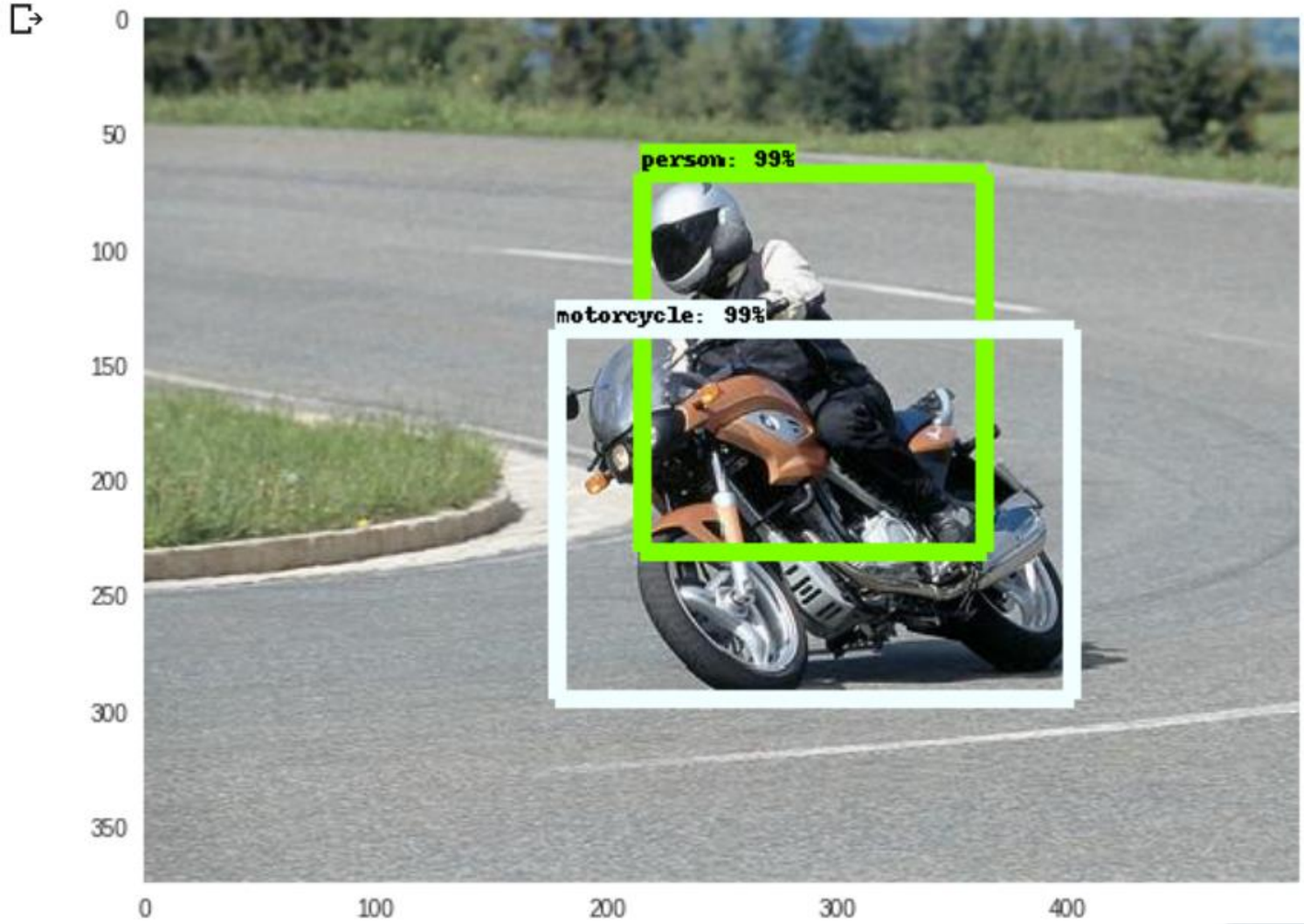
Auto-execute cell when fields change

Display output below form

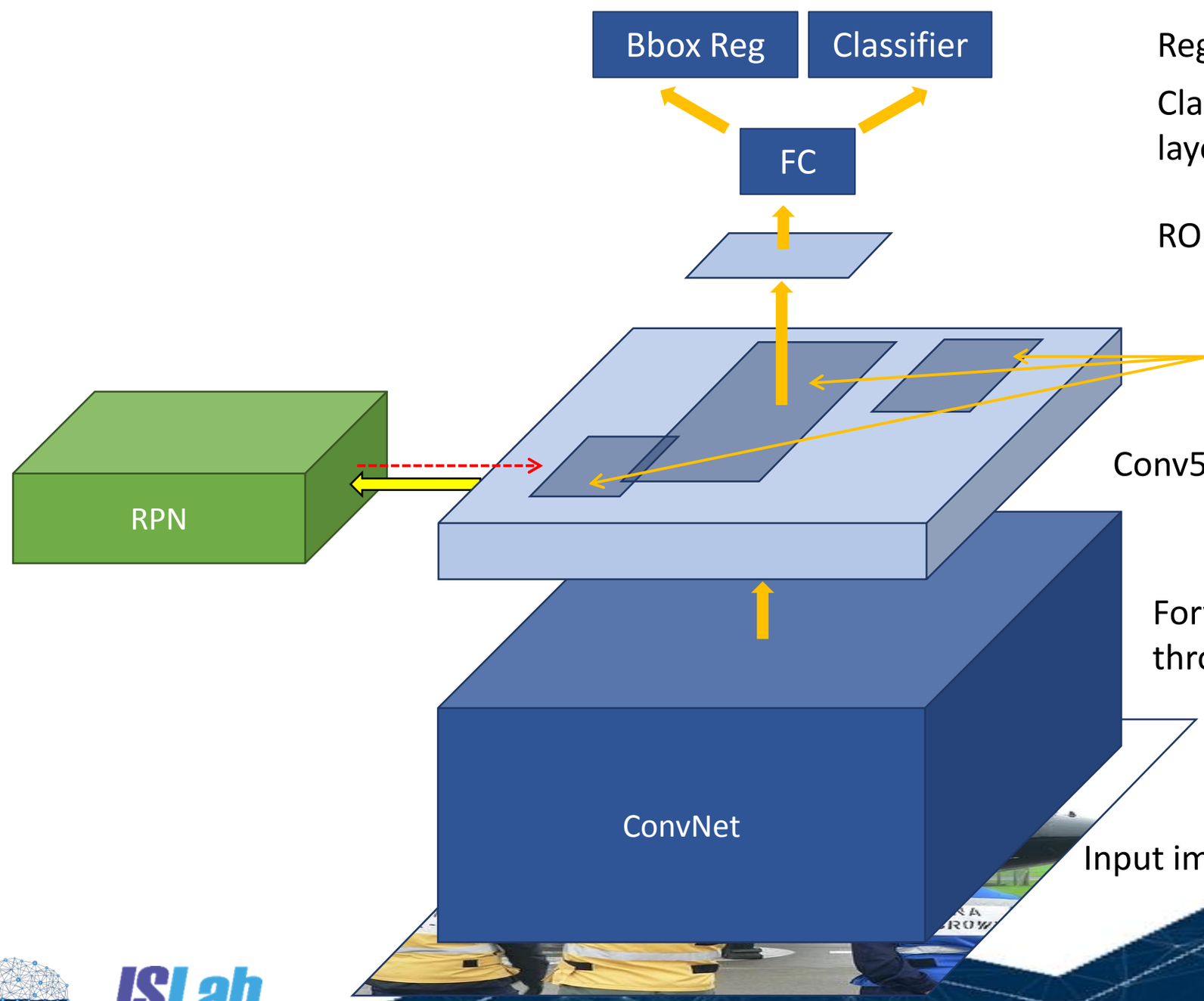
CANCEL SAVE

Example Output

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

size: 9
```



Faster R-CNN to Mask R-CNN



Regressor using linear layer

Classify using linear+softmax layer

ROI pooling (single level SPP layer)

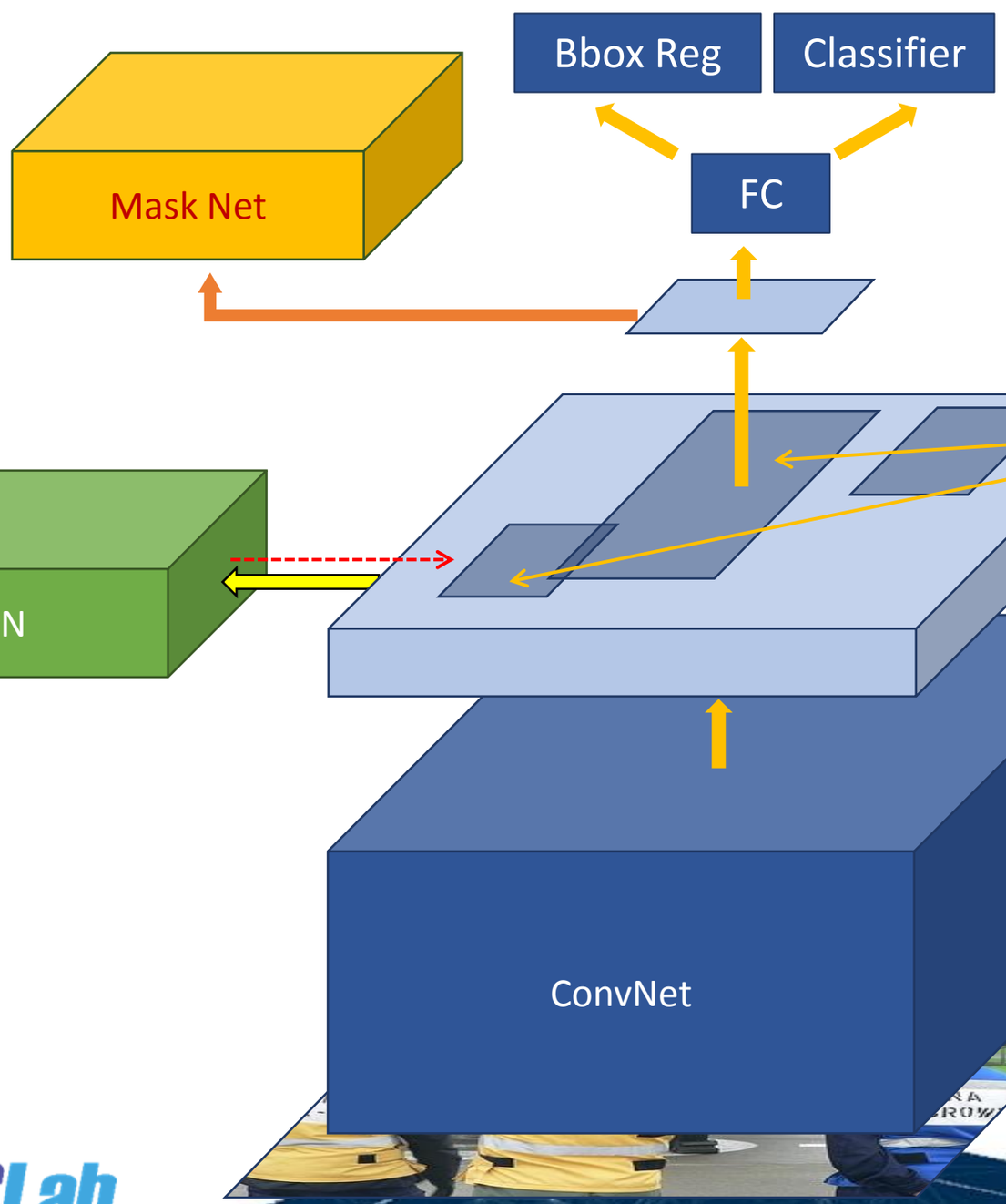
Region of interest (ROI) from the RPN

Conv5 feature map of the image

Forward the whole image through ConvNet

Input image

Faster R-CNN to Mask R-CNN



Regressor using linear layer

Classify using linear+softmax layer

ROI align

Region of interest (ROI) from the RPN

Conv5 feature map of the image

Forward the whole image through ConvNet

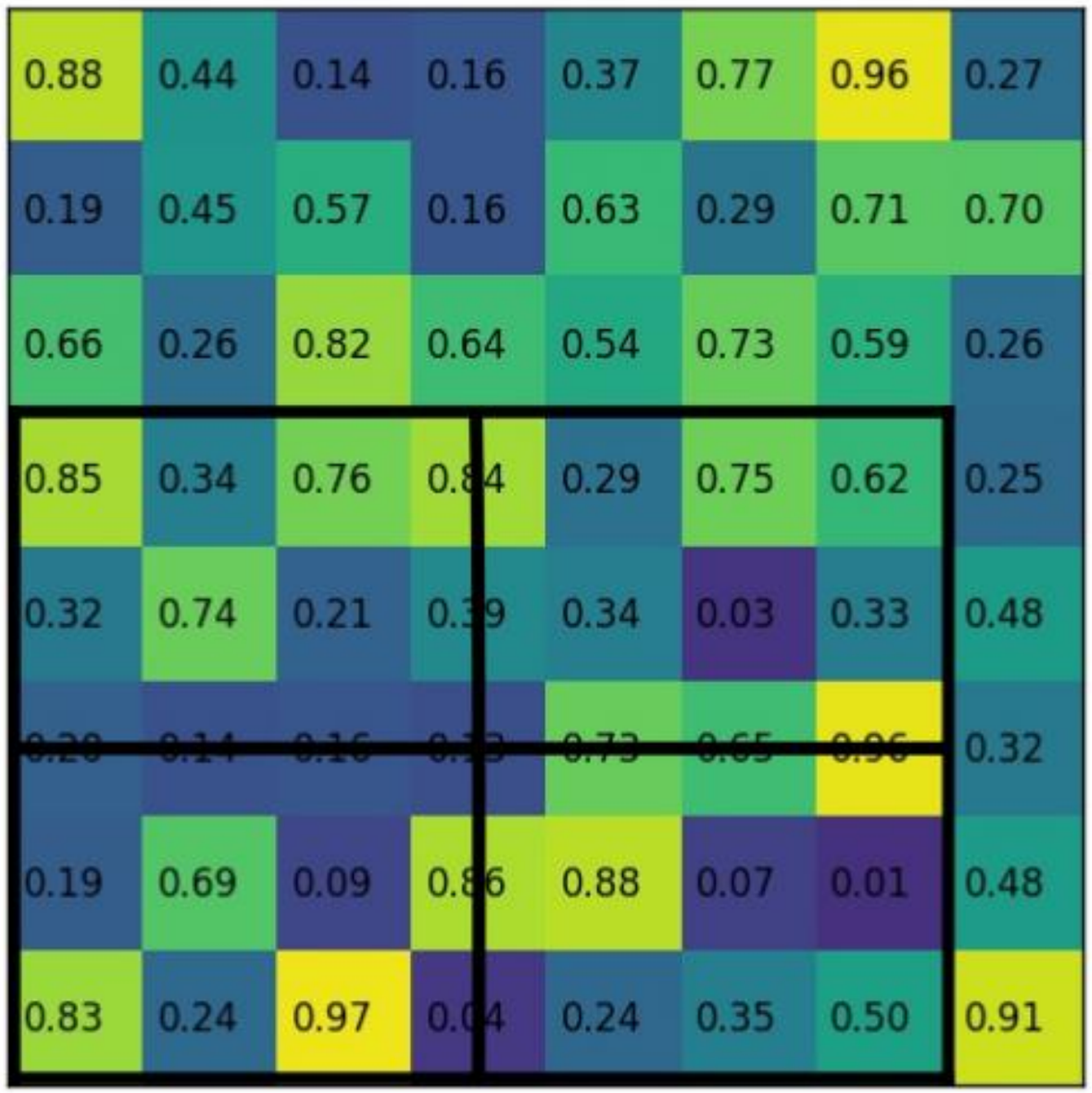
Input image

ROI Align

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

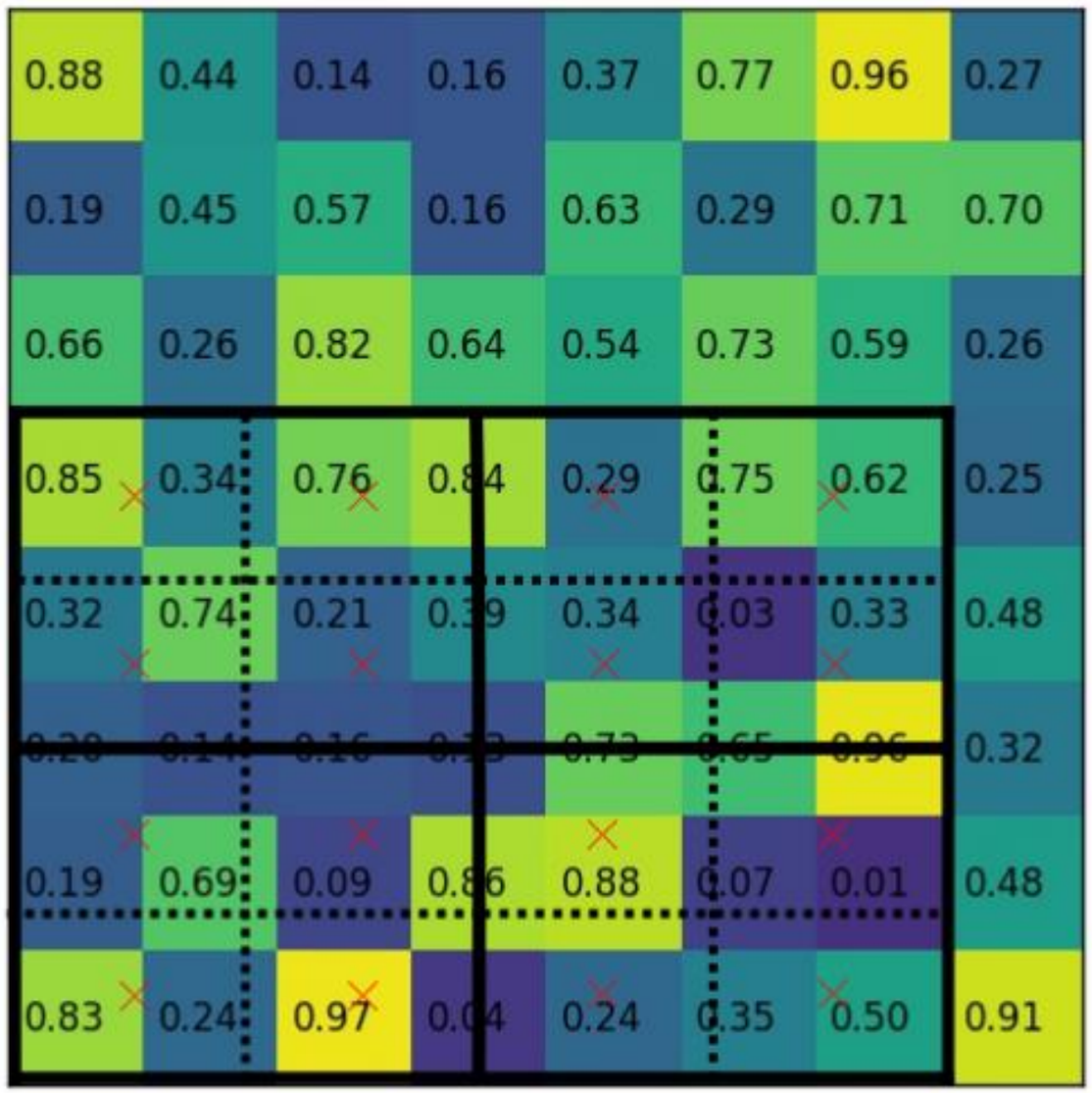
Input activation

ROI Align

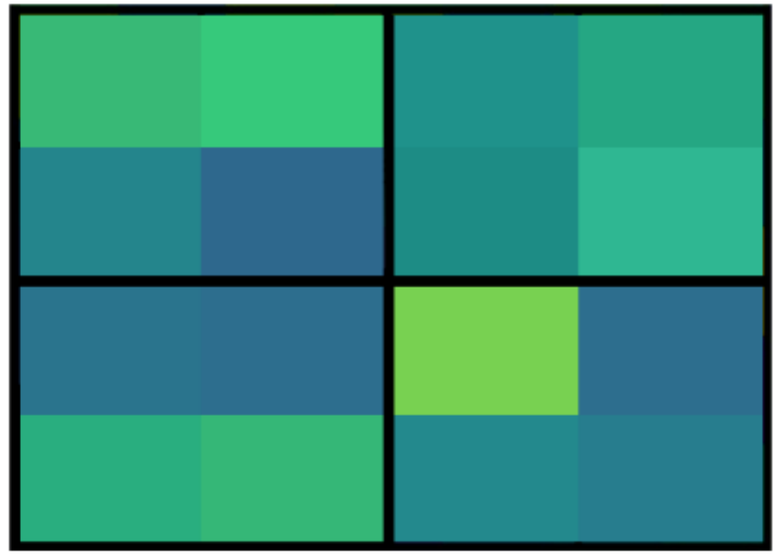


Region projection and pooling sections

ROI Align

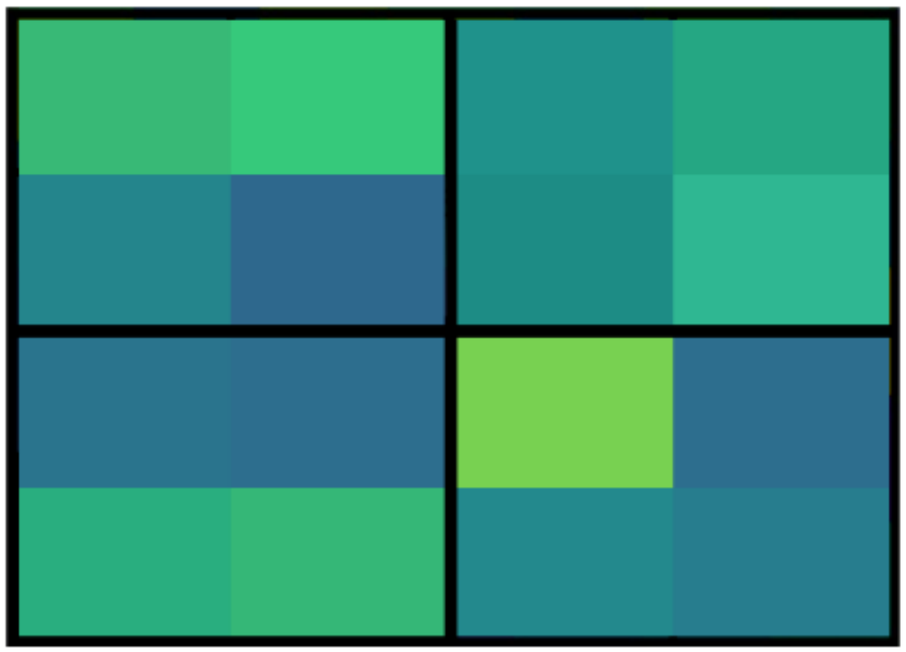


Sampling locations



Bilinear interpolated values

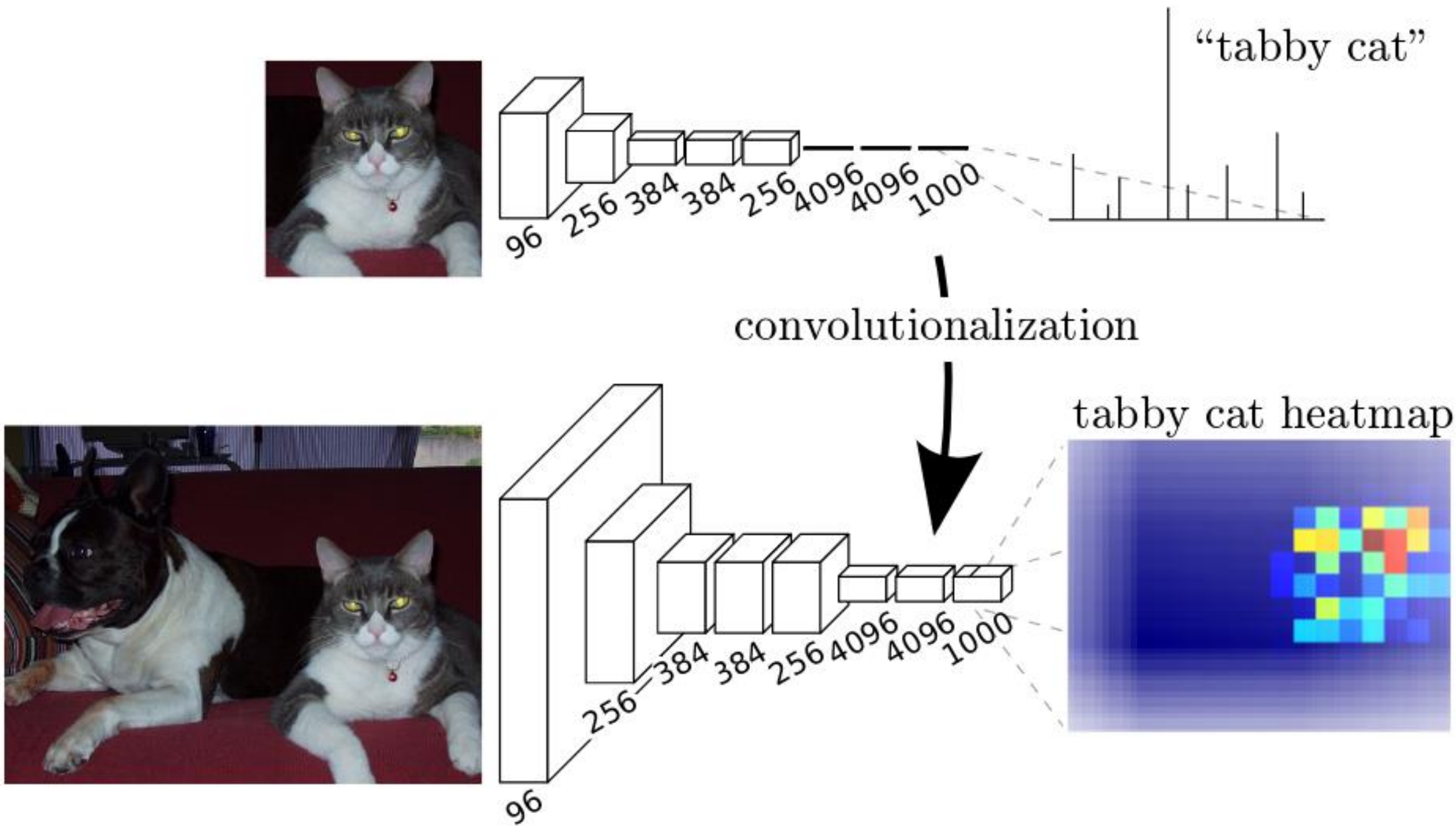
ROI Align



Bilinear interpolated values

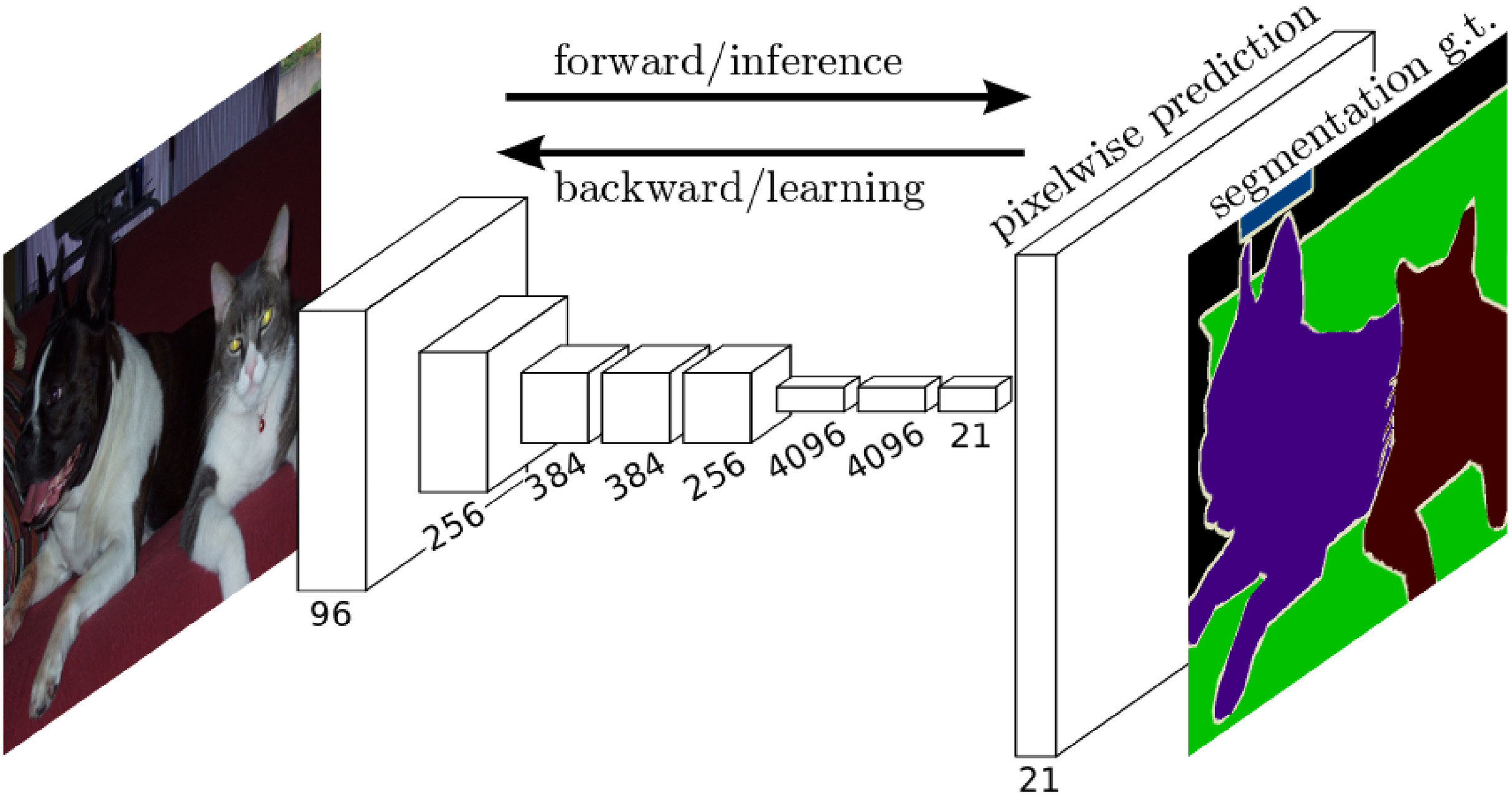
Max pooling output

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



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

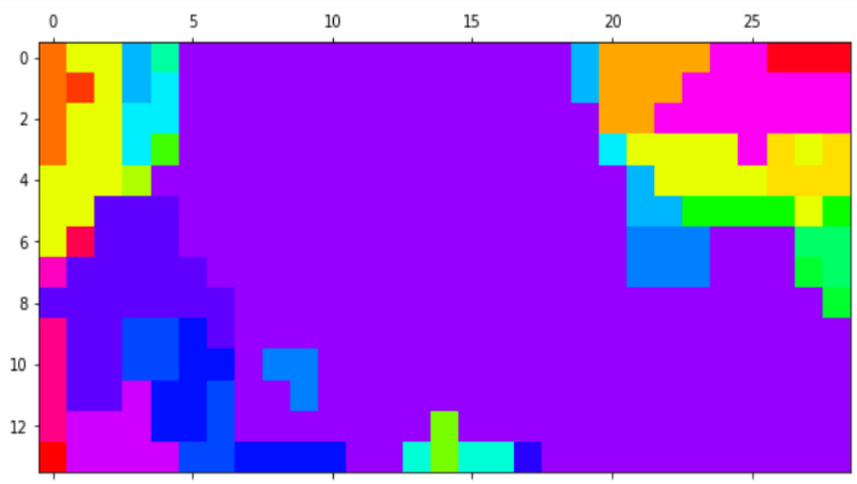
Mask Network in Mask R-CNN (2/2)

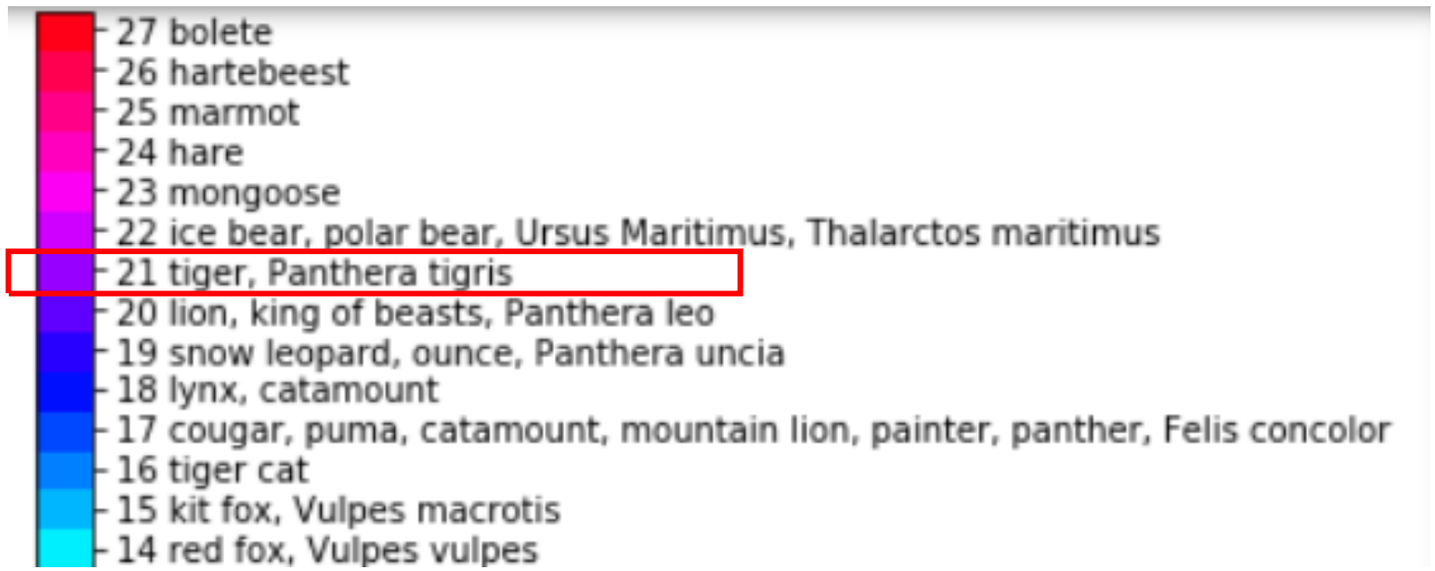


J. Long, E. Shelhamer, and T. Darrell. "Fully convolutional networks for semantic segmentation." CVPR. 2015.

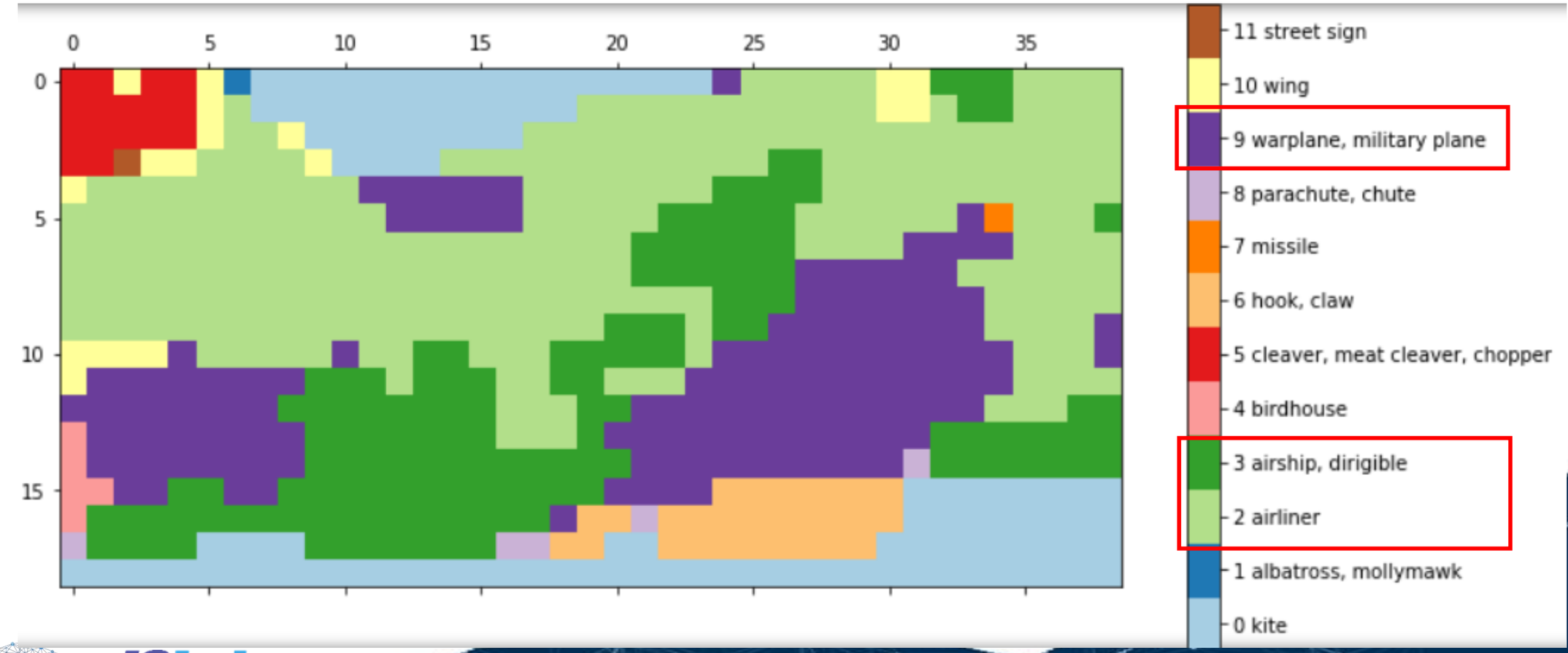
Real Example (1/2)

- Base network: VGG16 (fc6, fc7, and fc8 are convolutional, 7x7)





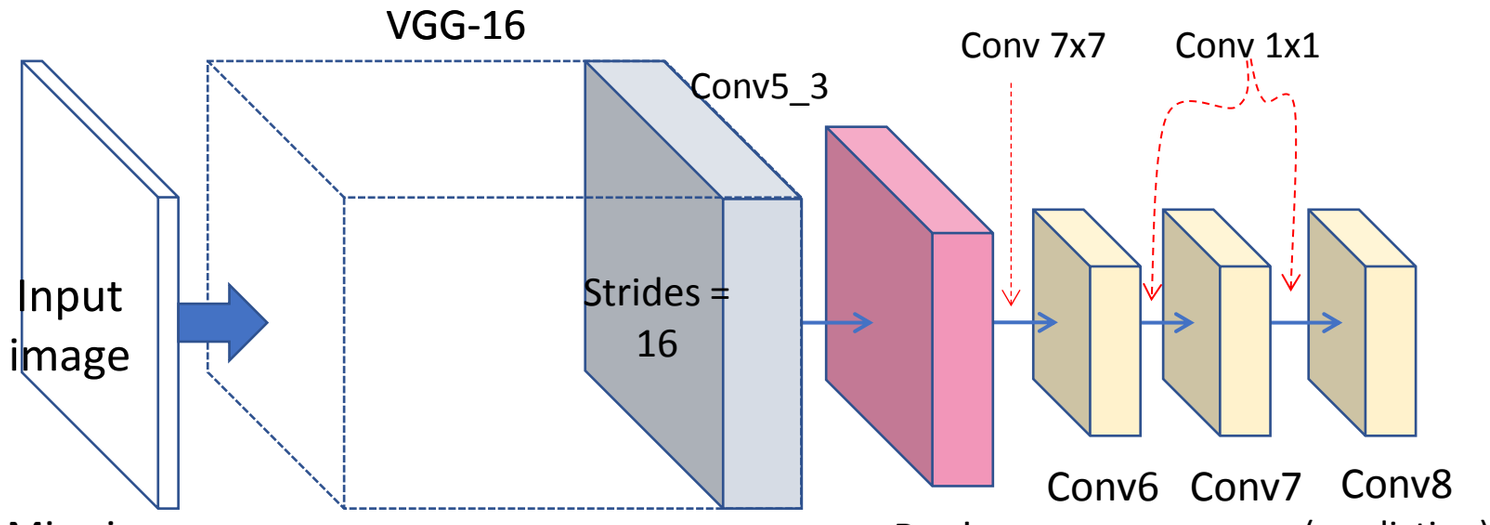
Real Example (2/2)

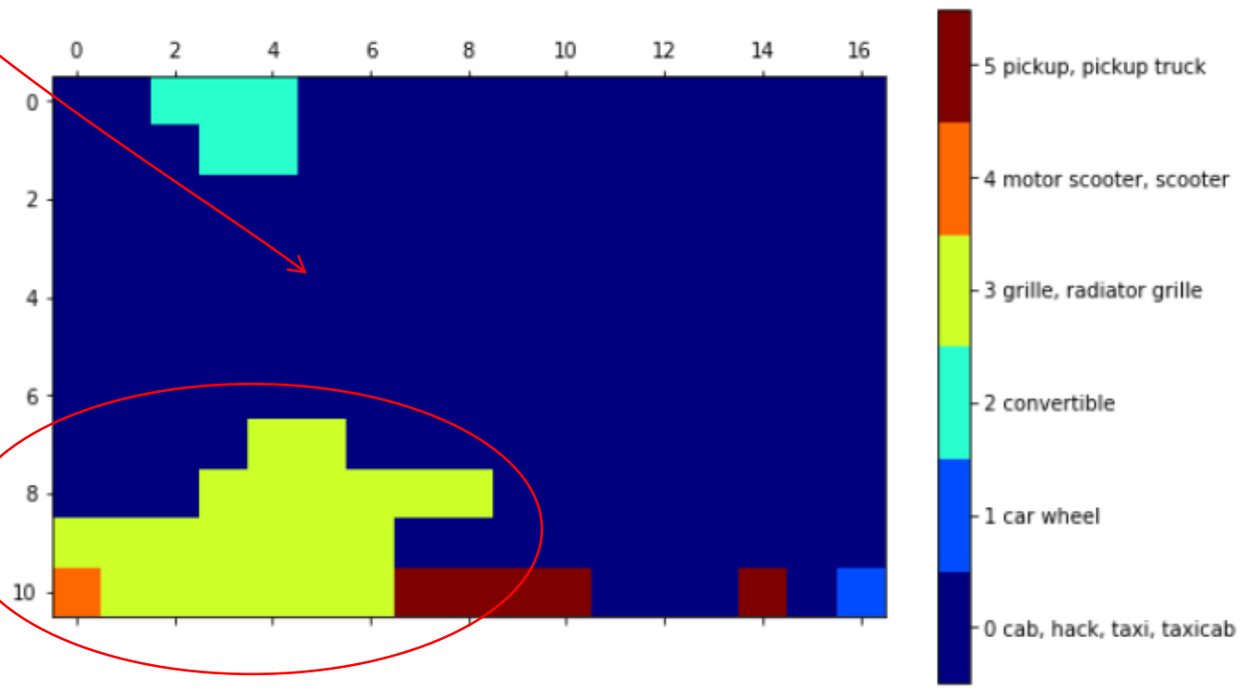


Semantic Segmentation - Intuition

input

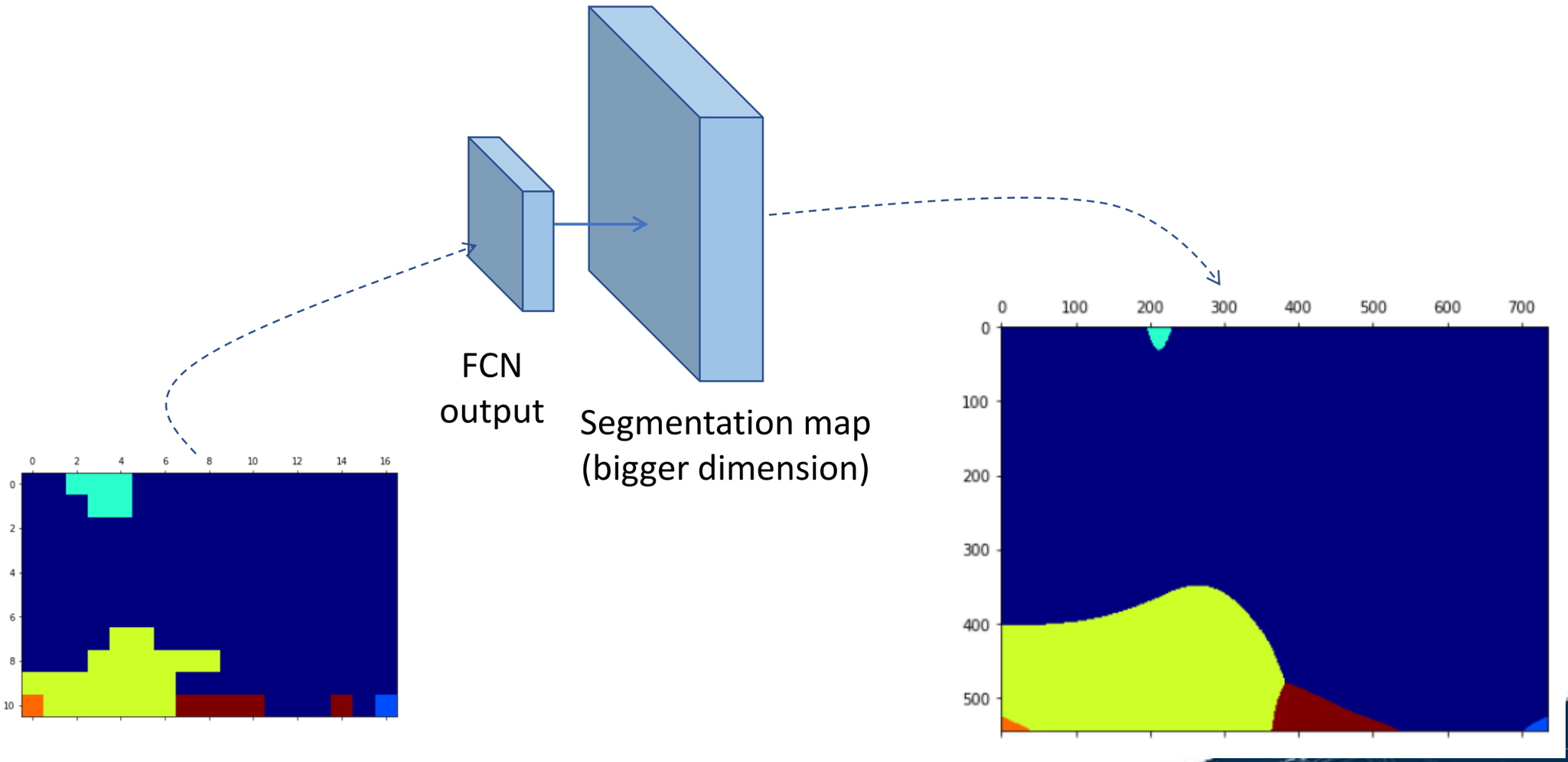
output





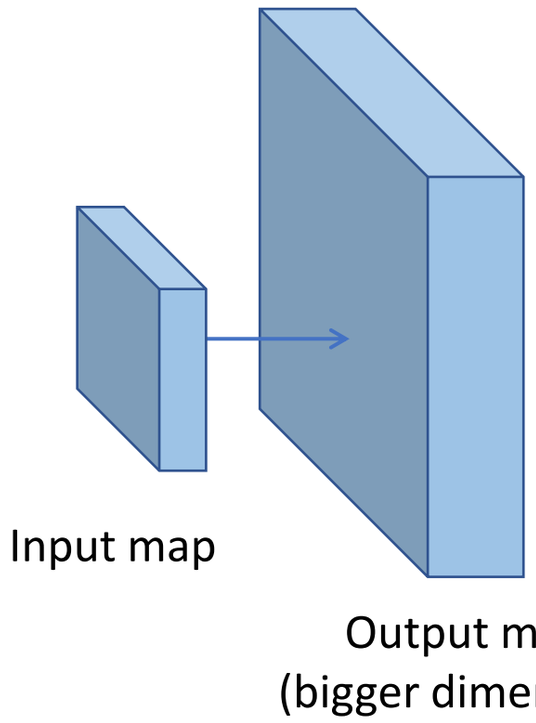
Semantic Segmentation - Intuition

- We should be able to achieve finer segmentations by:
  - Interpolating the FCN output map to original image size
  - Fine tuning the network using segmentation label



The Up-sampling Operation

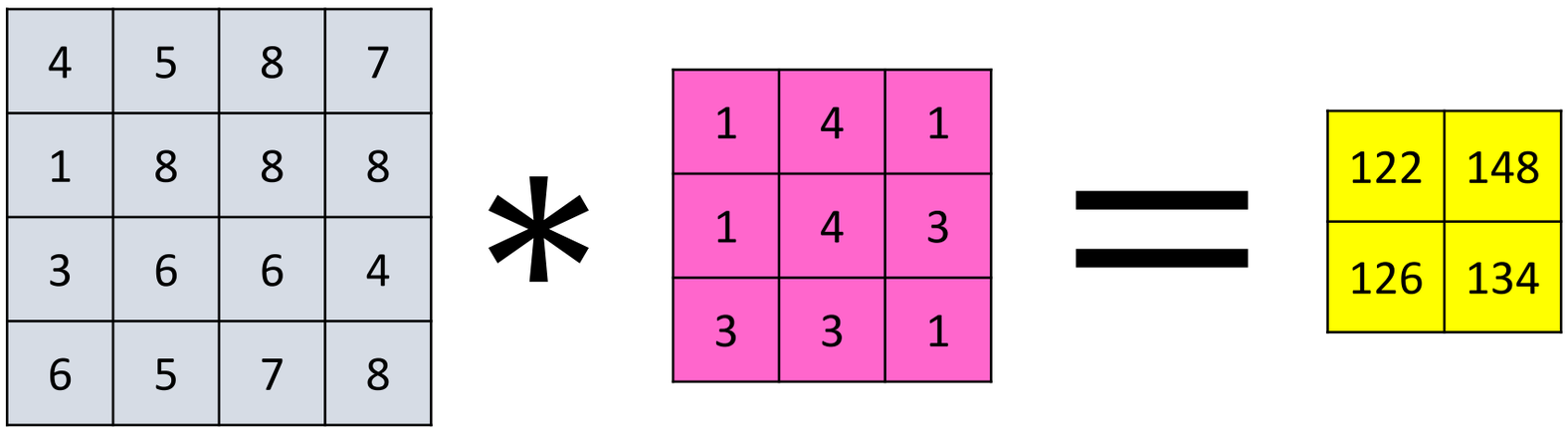
- To obtains feature map with bigger dimension



- 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 (1/6)

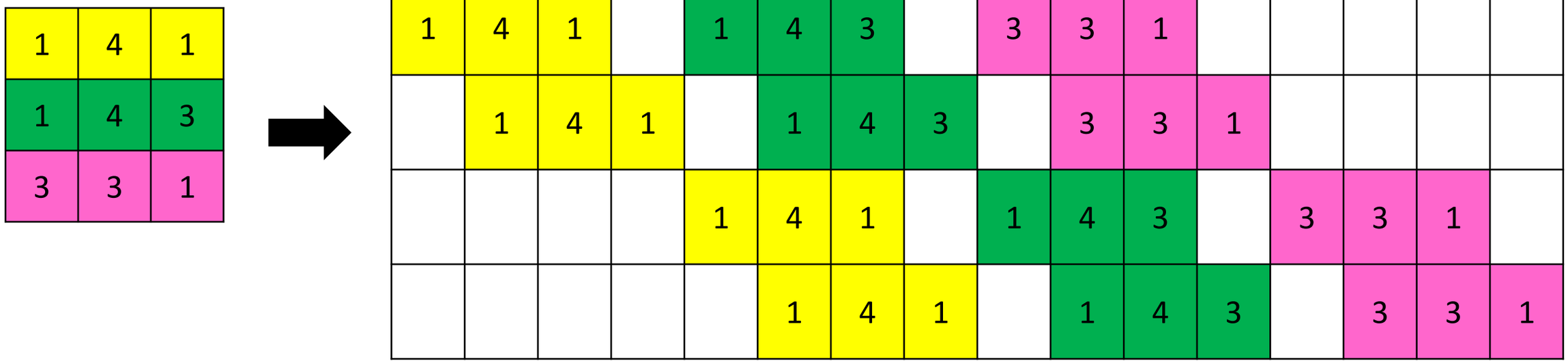
- Illustration of an “ordinary convolution”:



Input kernel output

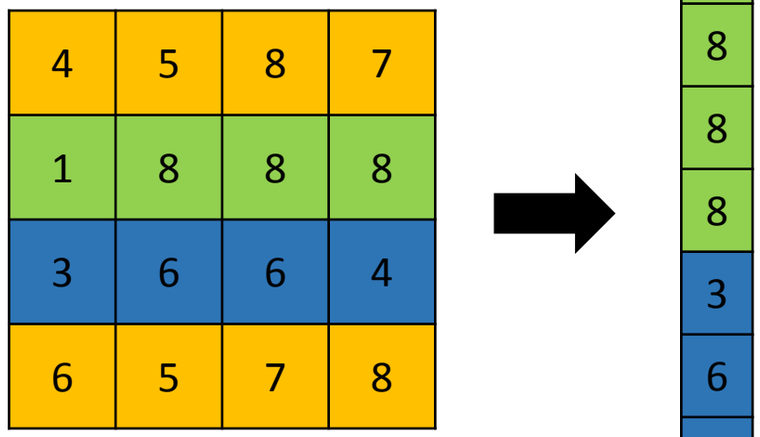
Convolution as Matrix Operation (2/6)

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



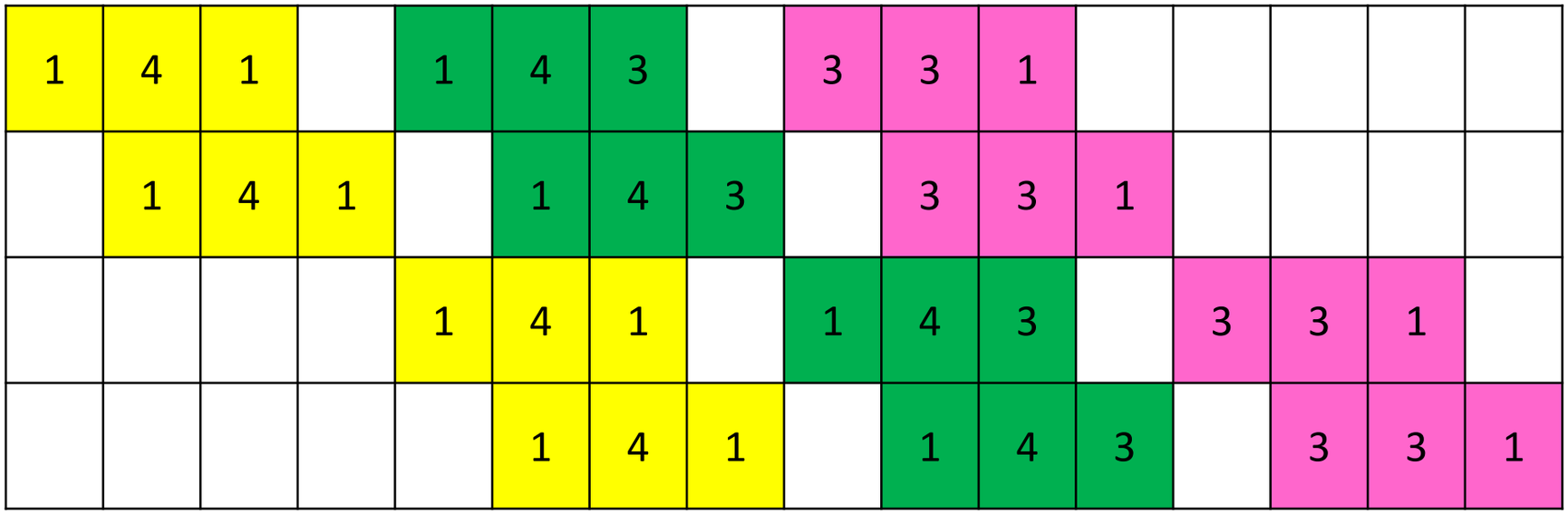
Convolution as Matrix Operation (3/6)

- And the input as its serialized form:

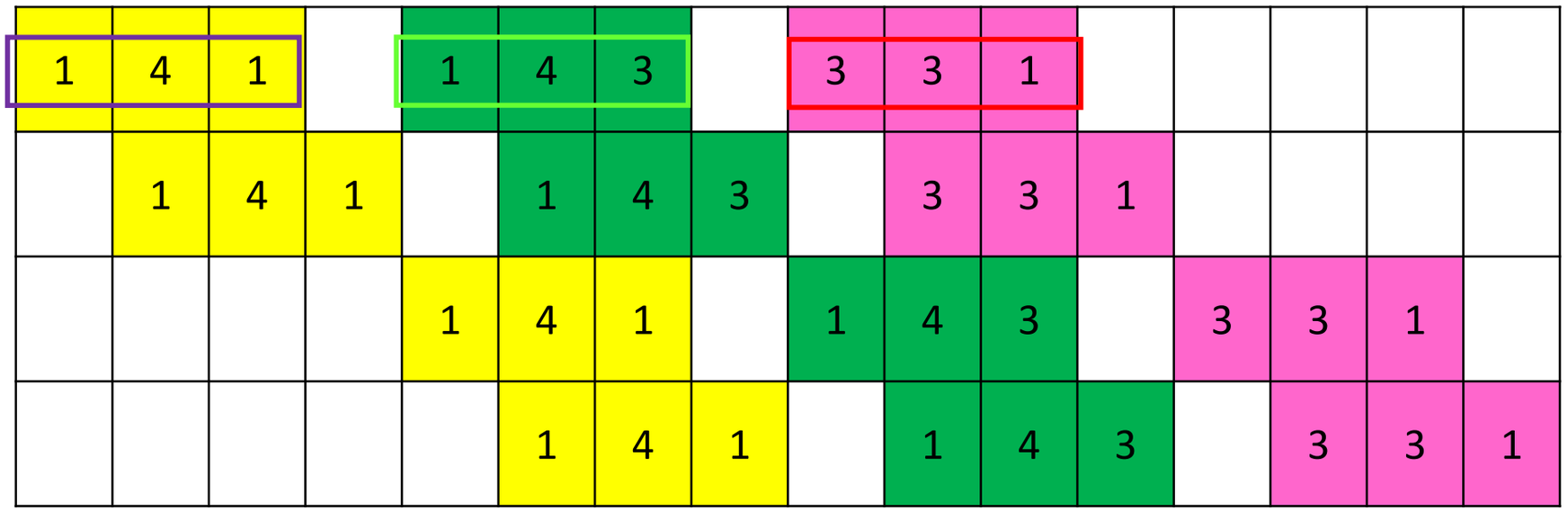


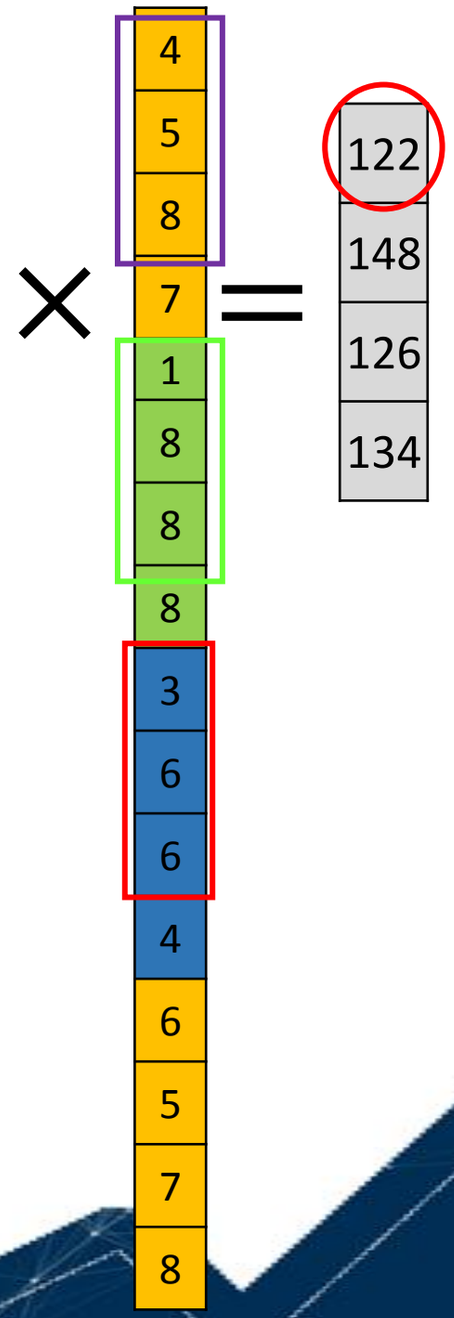
Convolution as Matrix Operation (4/6)

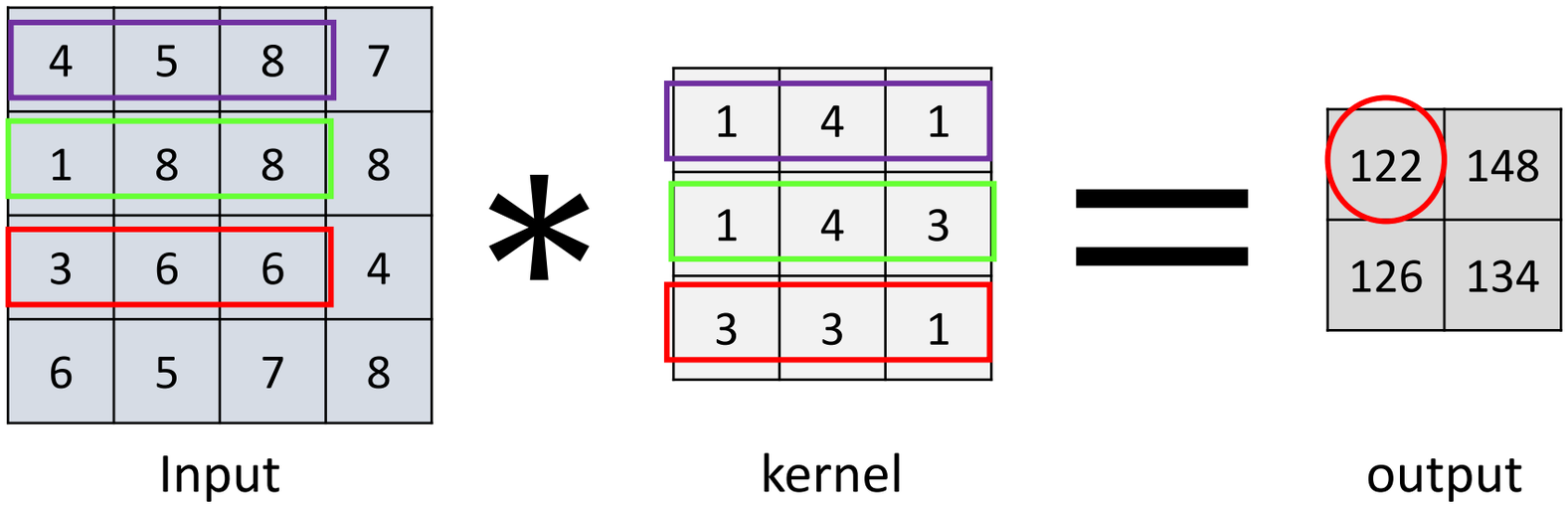
- Perform the convolution as matrix multiplication
  - The matrix multiplication produce same “values”



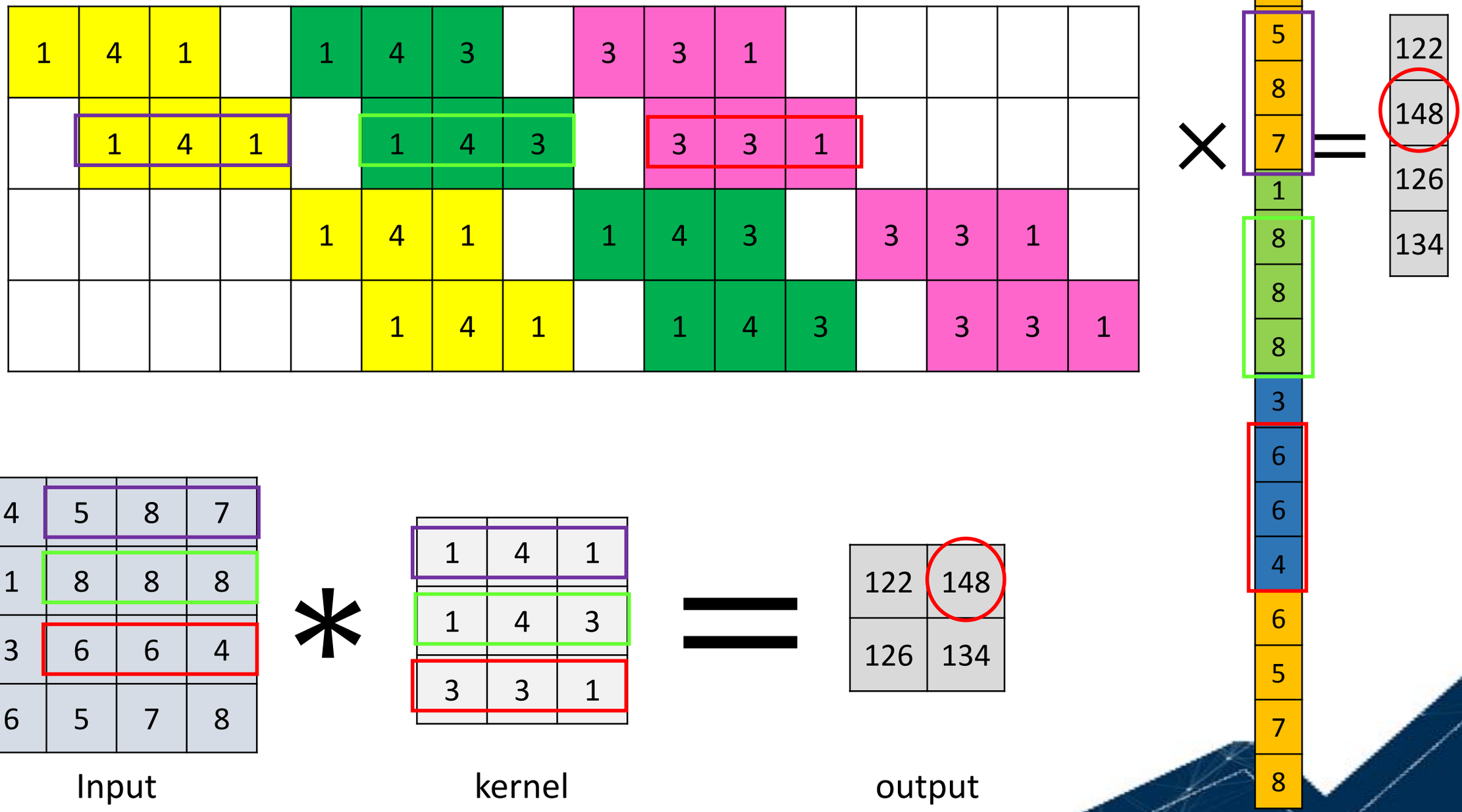
Convolution as Matrix Operation (5/6)







Convolution as Matrix Operation (6/6)



The Transposed Convolution (1/2)

- Recall the convolution as matrix operation case:

$$C \times I = O$$

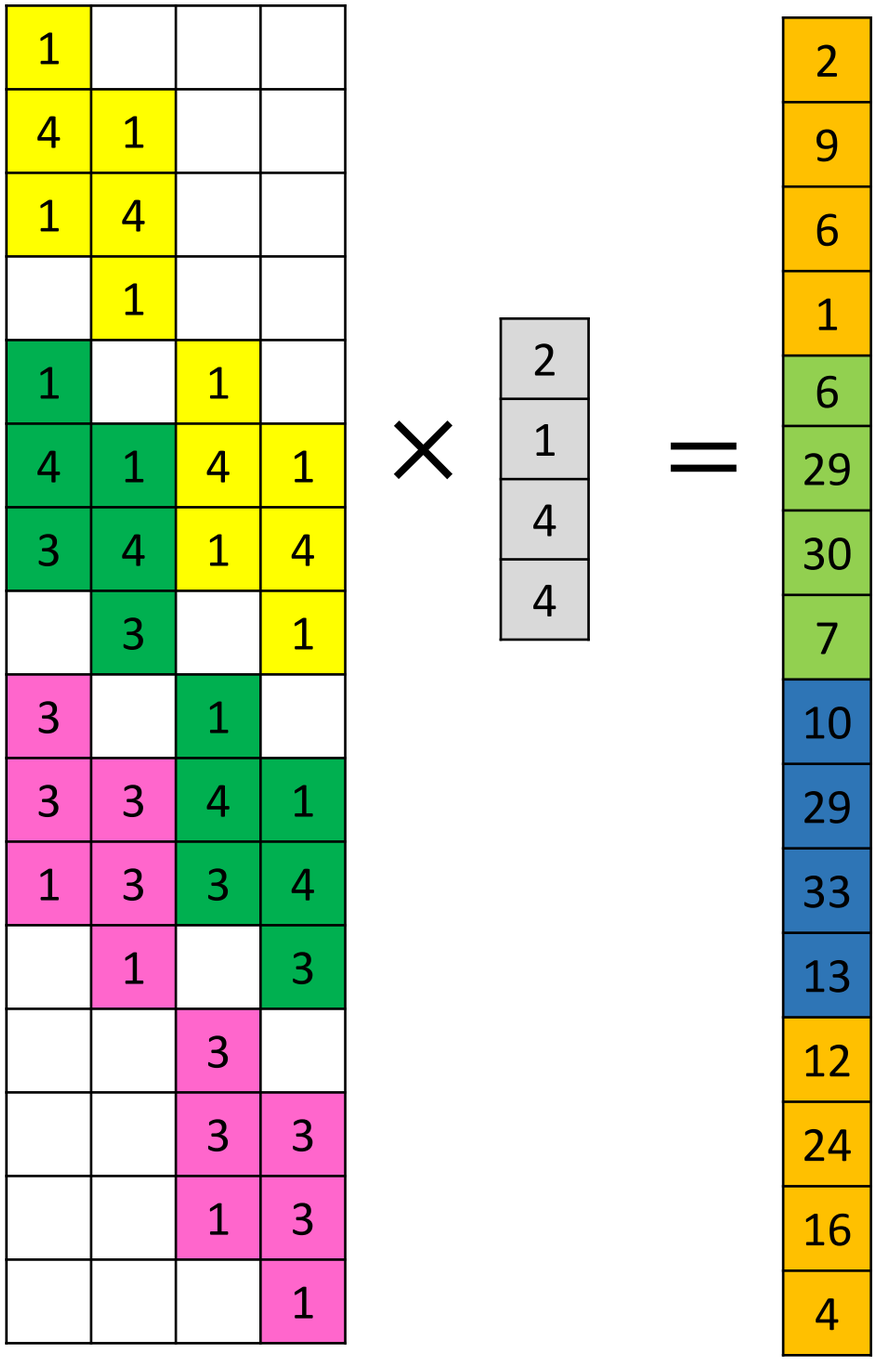
$$4 \times 16 \quad \mathbf{16 \times 1} \quad 4 \times 1$$

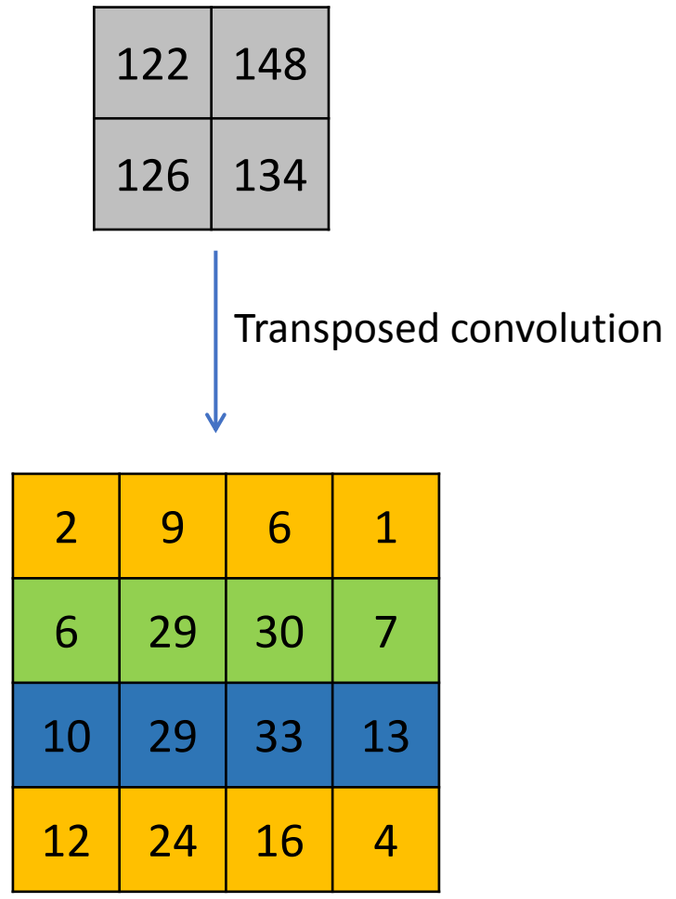
- If we have convolution result O,
  - we can "obtains" $\hat{I}$, which may represent the original input image
  - Simplest way: using the transposed convolution kernel $\hat{C}$

$$\hat{C} \times O = \hat{I}$$

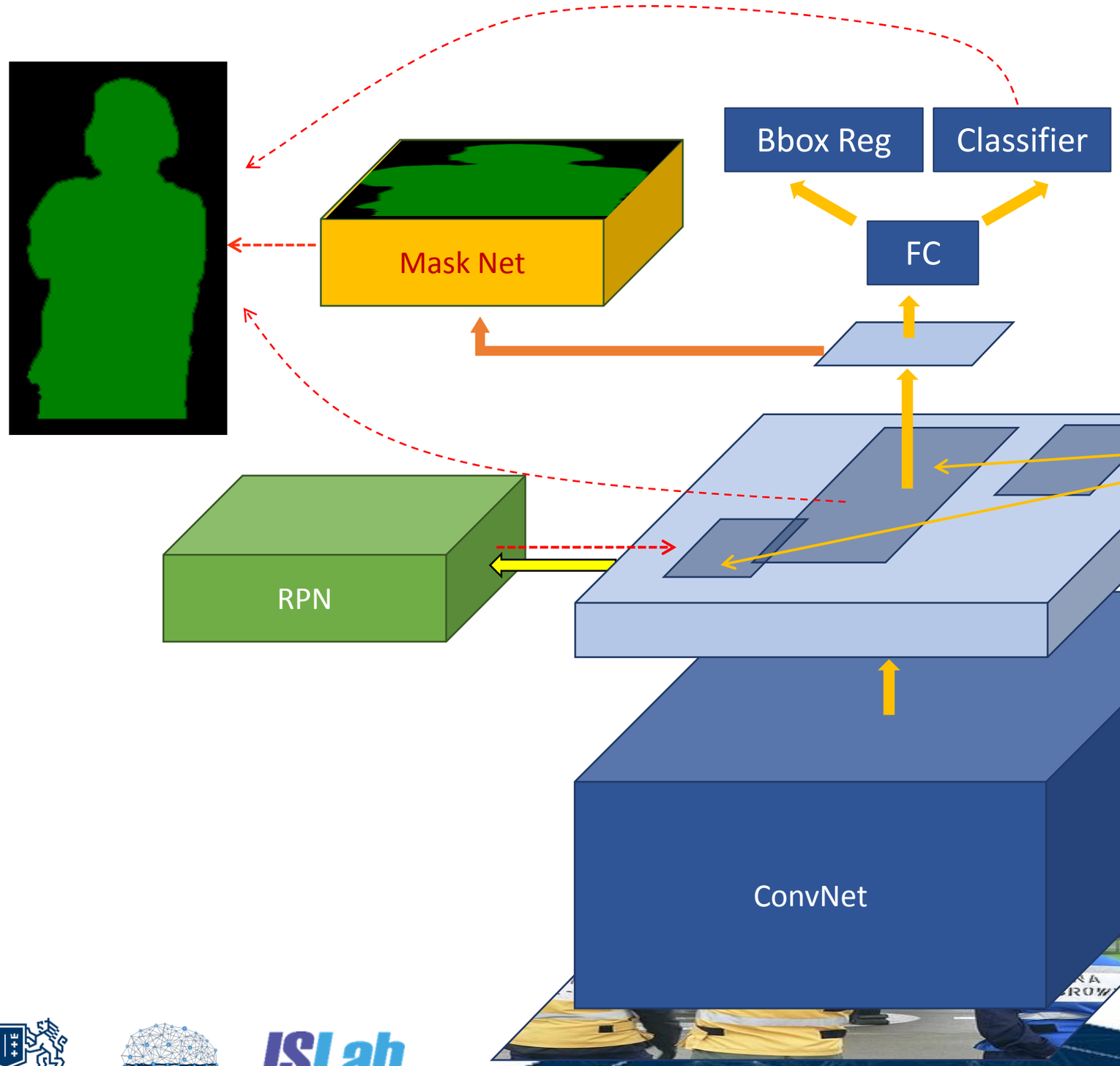
$$16 \times 4 \quad 4 \times 1 \quad \mathbf{16 \times 1}$$

The Transposed Convolution (2/2)





Mask R-CNN



Regressor using linear layer

Classify using linear+softmax layer

ROI align

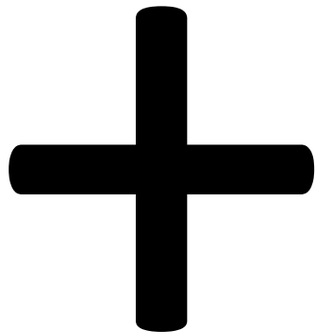
Region of interest (ROI) from the RPN

Conv5 feature map of the image

Forward the whole image through ConvNet

Input image

Lets try on Colab!



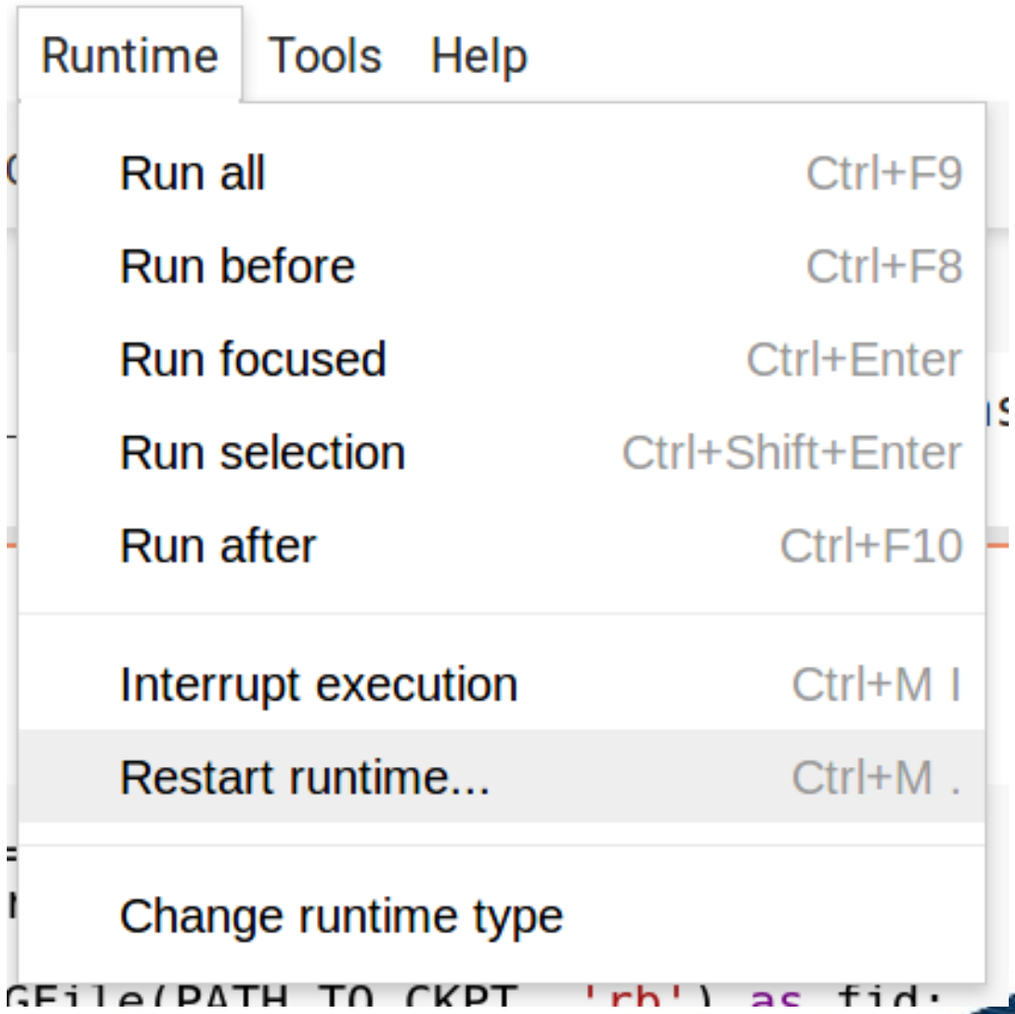
tinyurl.com/hsi18-colab

Download the Mask R-CNN Model

```
SAVE_FOLDER = 'hsi/'

# 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('hsi/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, Directly Run the Required Fields

Download and unzip the pre-trained model

+ form

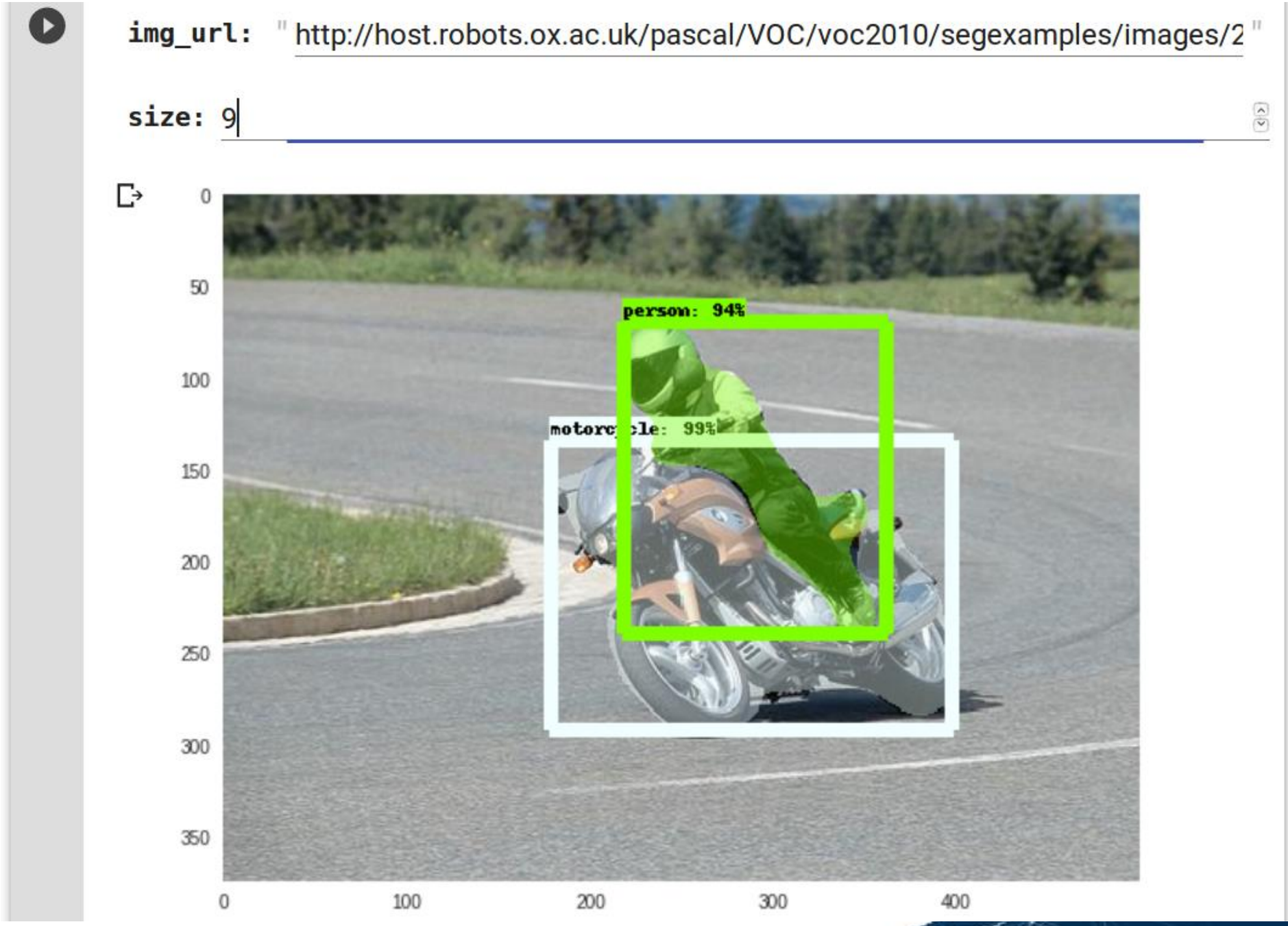
```
opener = urllib.request.URLopener()
opener.retrieve(DOWNLOAD_BASE + MODEL_FILE, SAVE_FOLDER+MODEL_FILE)
tar_file = tarfile.open(SAVE_FOLDER+MODEL_FILE)
for file in tar_file.getmembers():
  file_name = os.path.basename(file.name)
  if 'frozen_inference_graph.pb' in file_name:
    tar_file.extract(file,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

Load the model

+ form

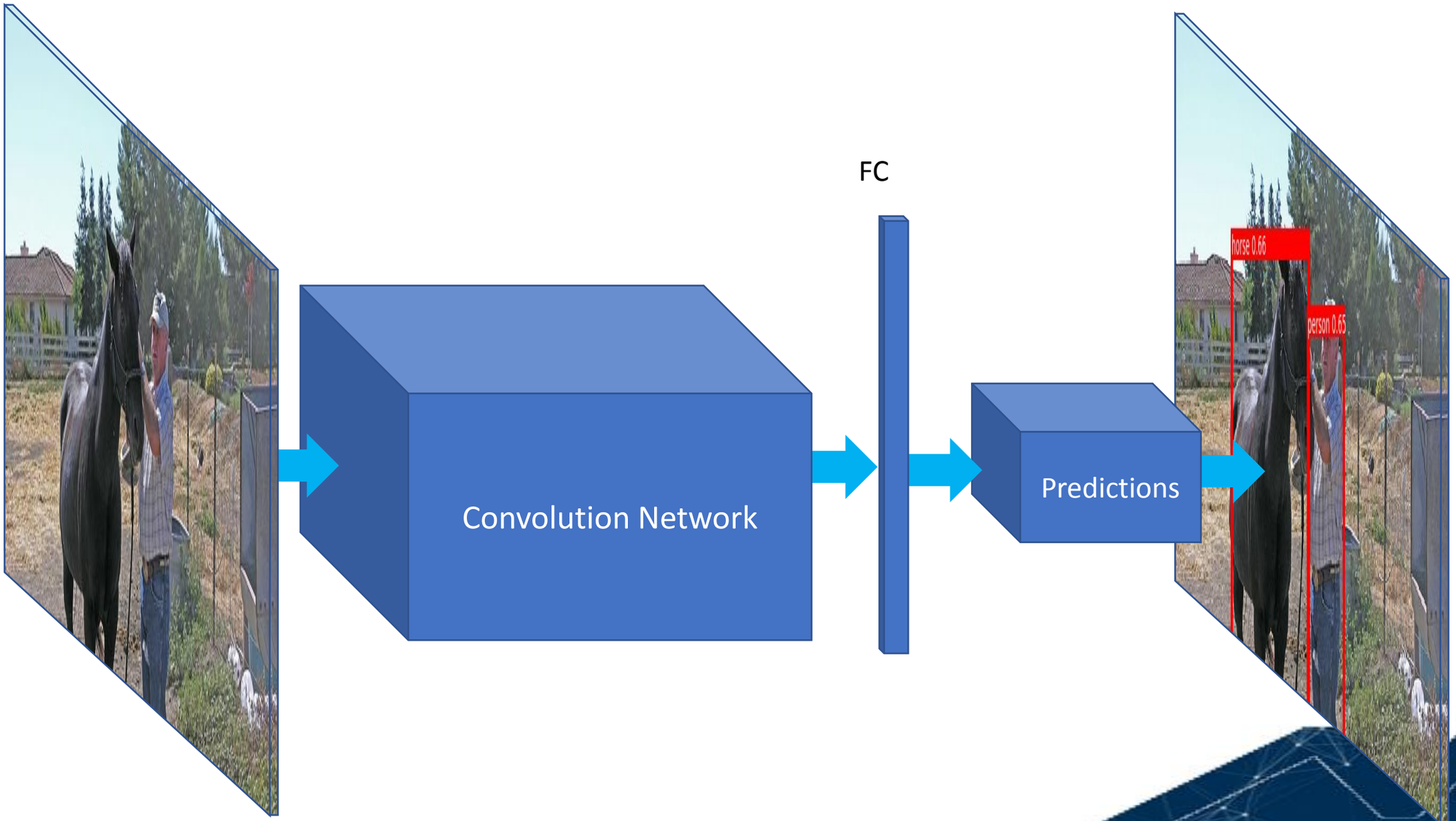
```
detection_graph = tf.Graph()
with detection_graph.as_default():
  od_graph_def = tf.GraphDef()
  with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
    serialized_graph = fid.read()
    od_graph_def.ParseFromString(serialized_graph)
    tf.import_graph_def(od_graph_def, name='')
```

Detection Result with Mask



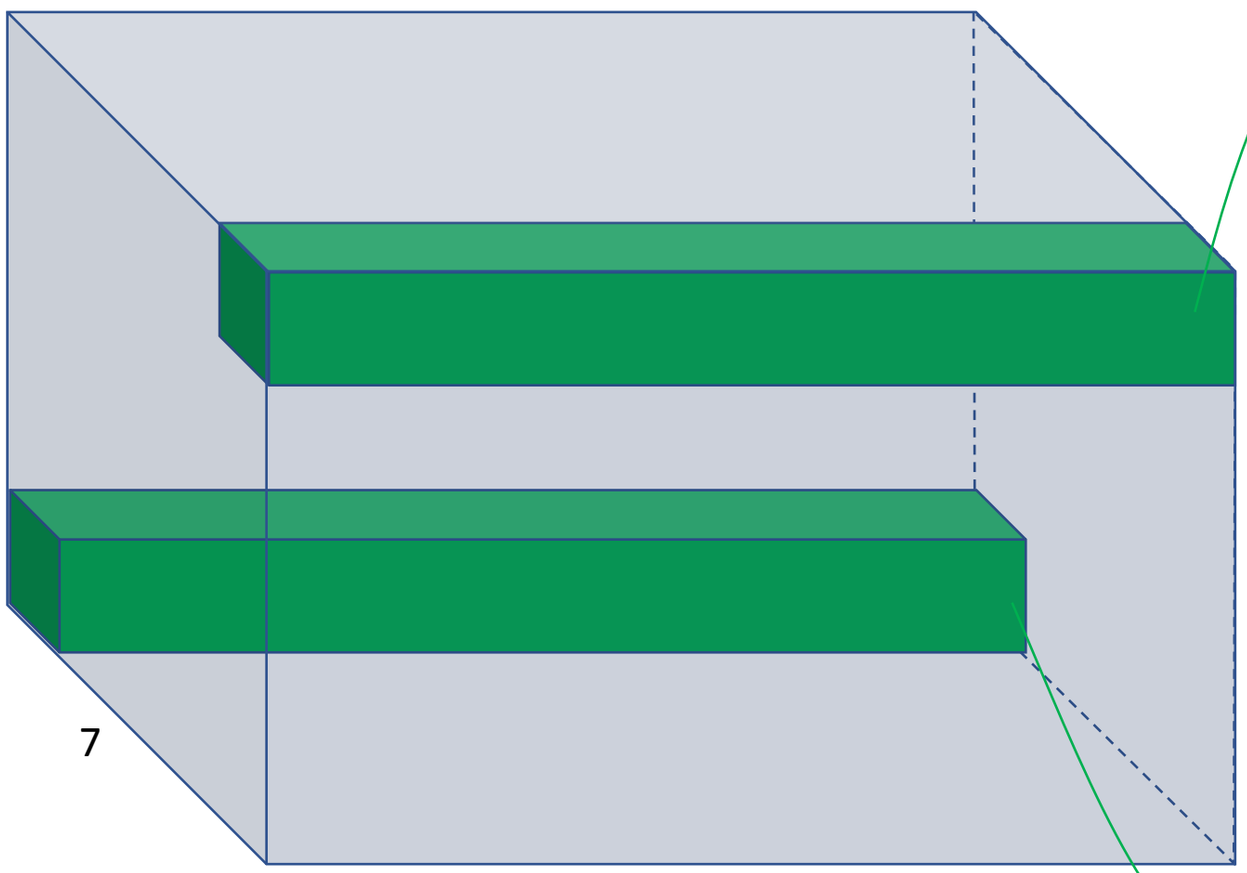
You Only Look Once (YOLO)

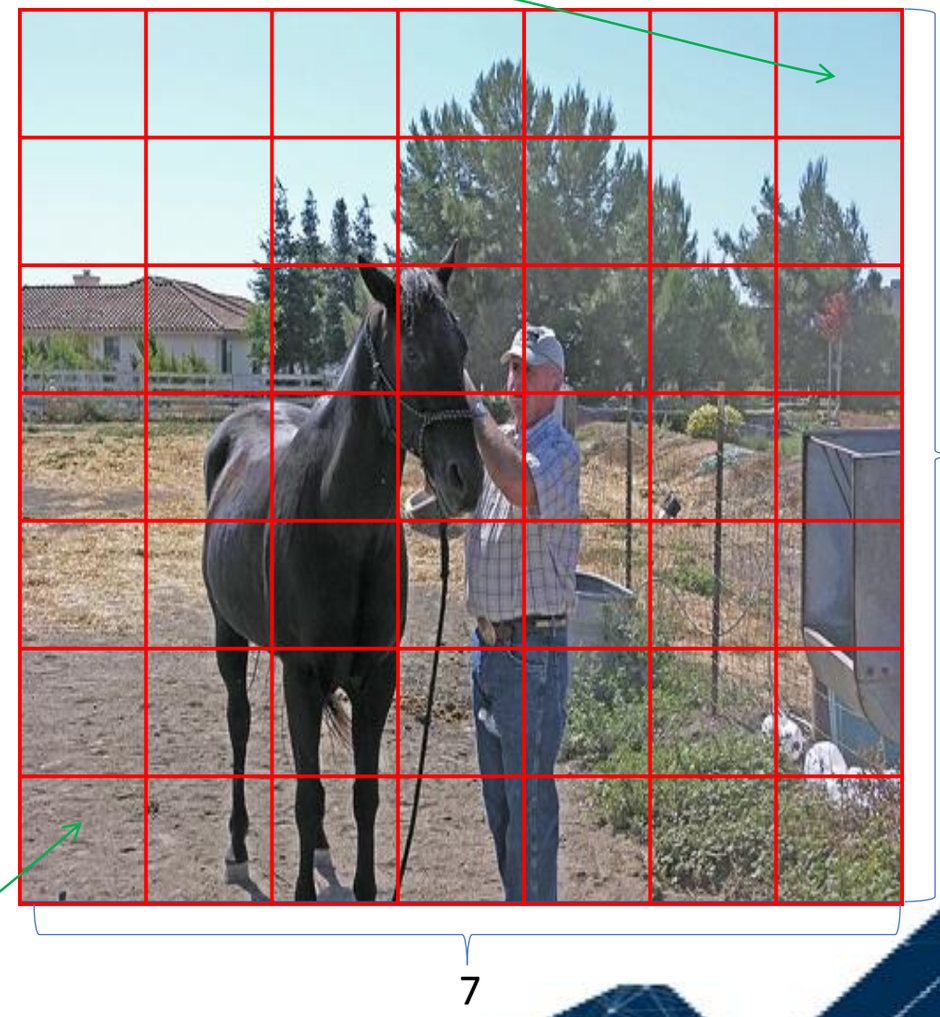
- Single stage detector



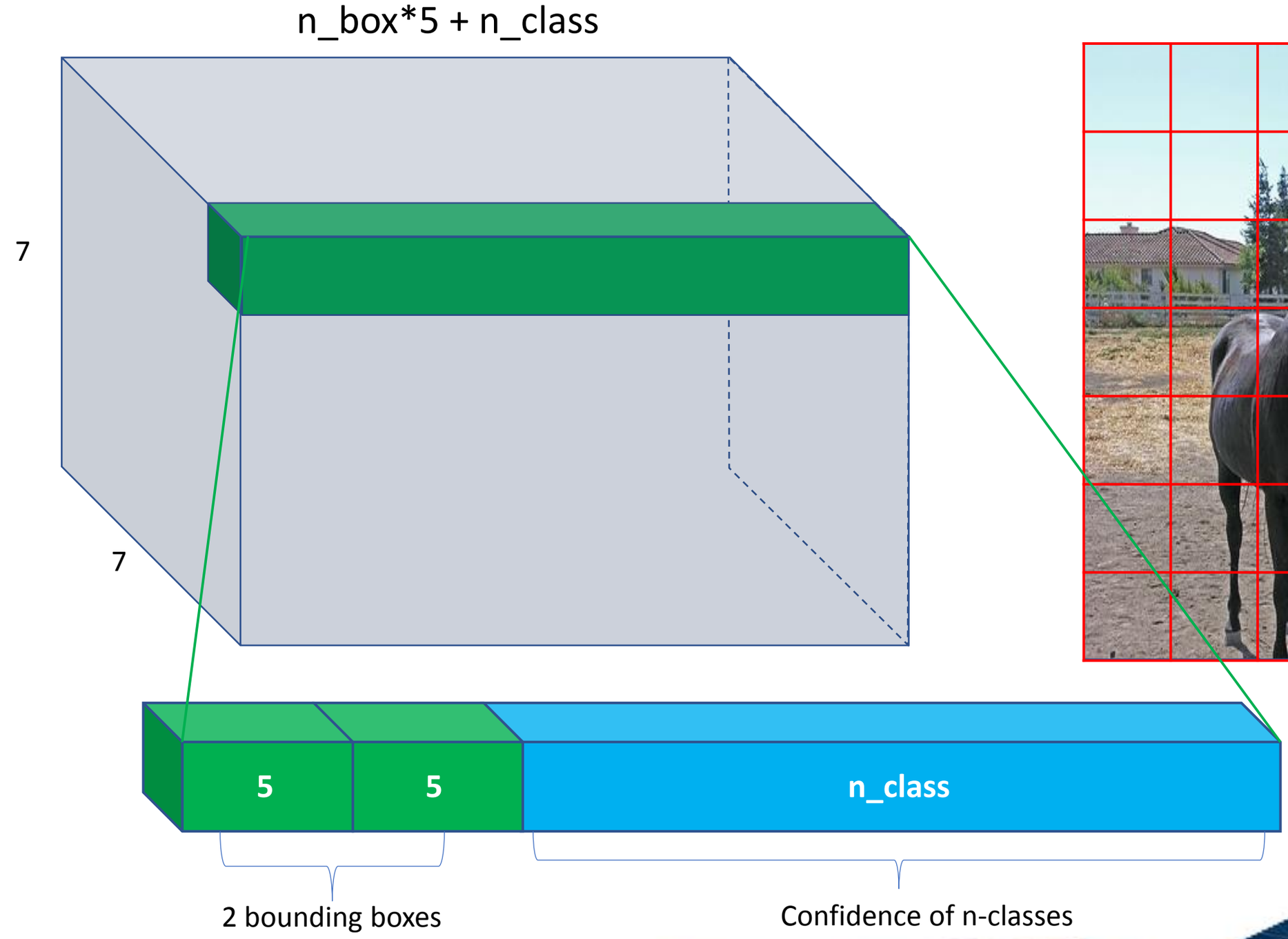
Prediction Layer in YOLO

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

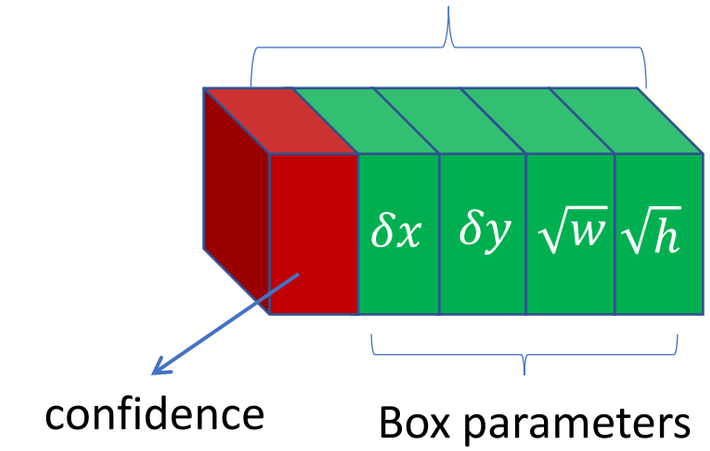


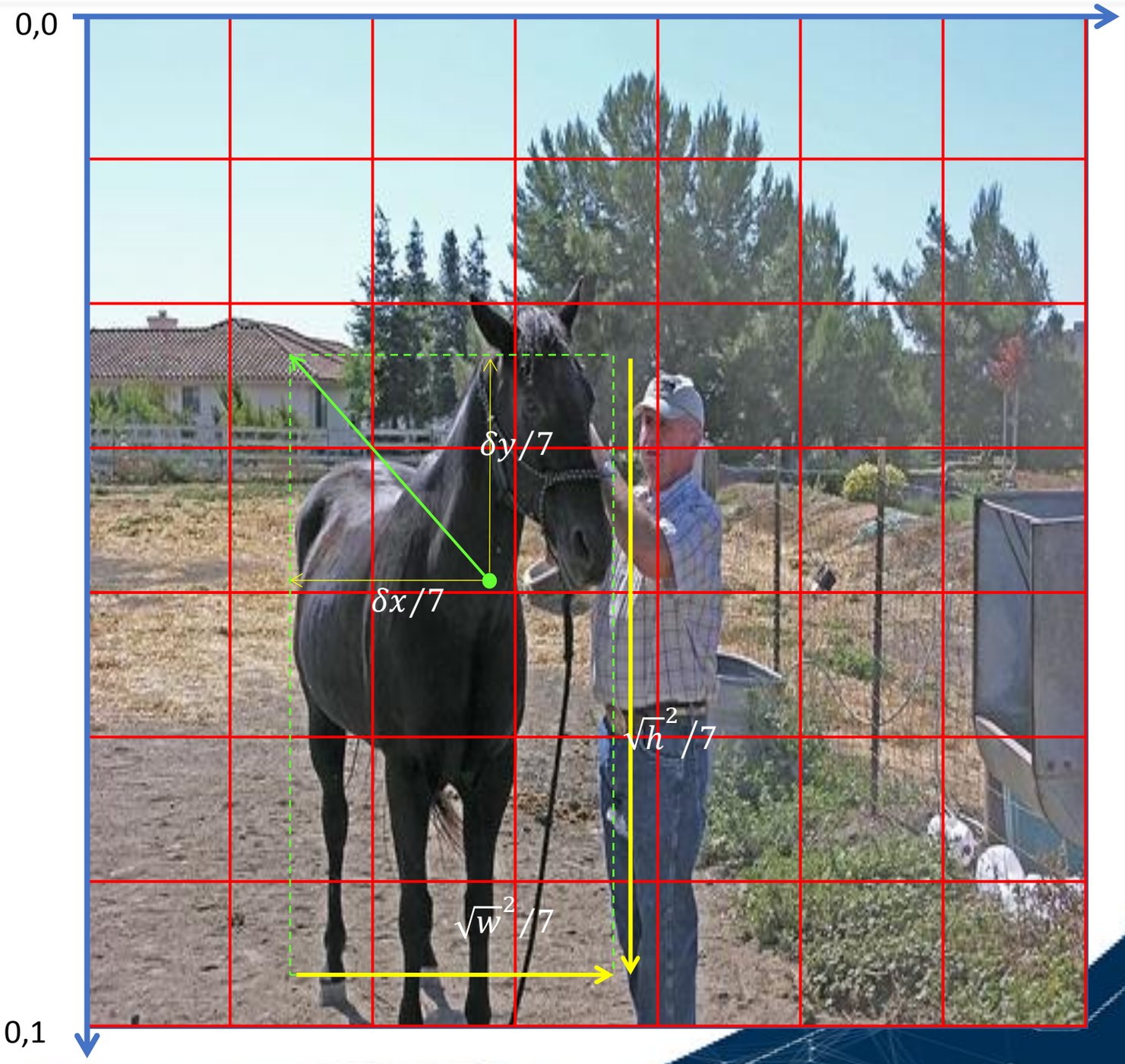


What Components are being Predicted?

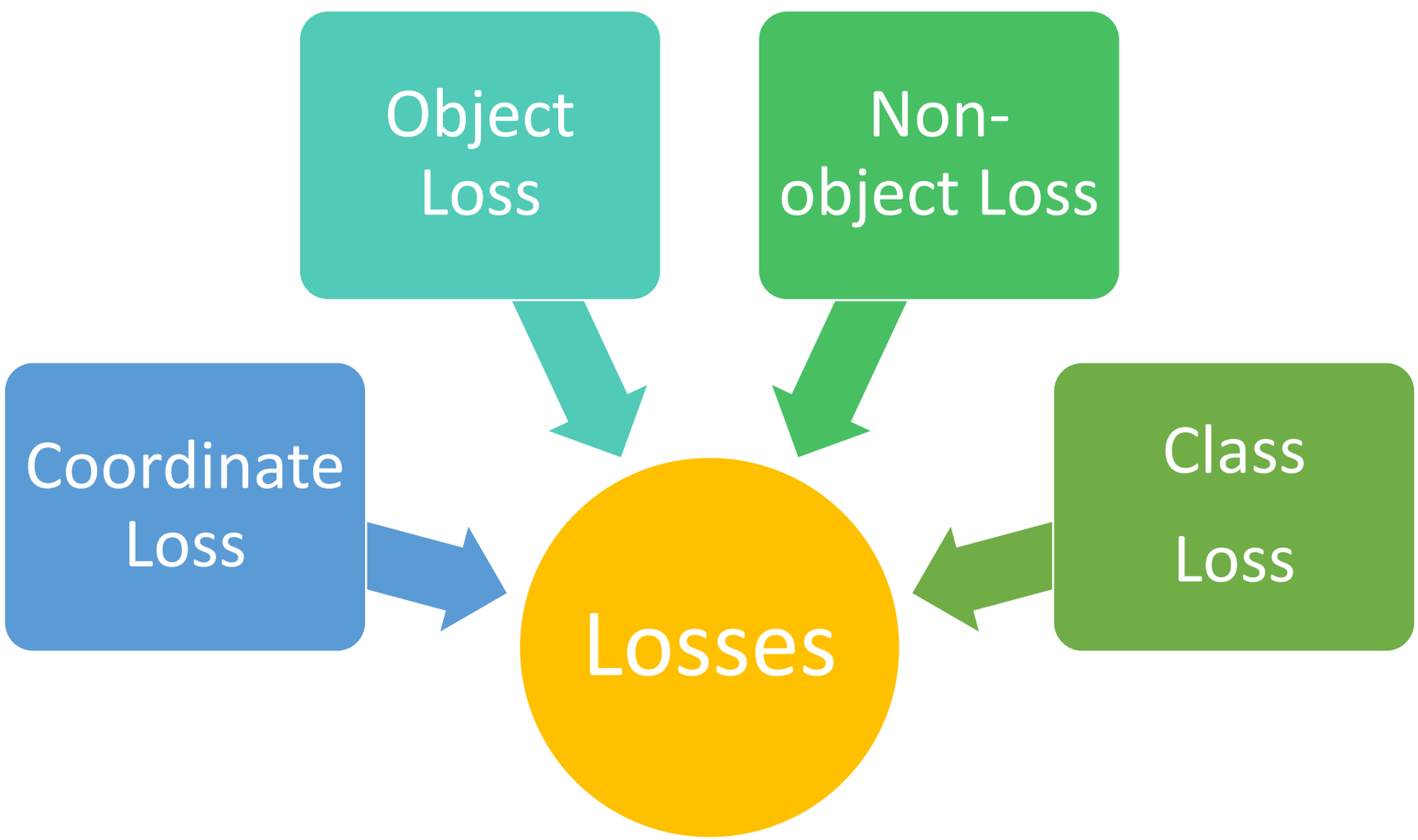


More Details on Bounding Box Prediction





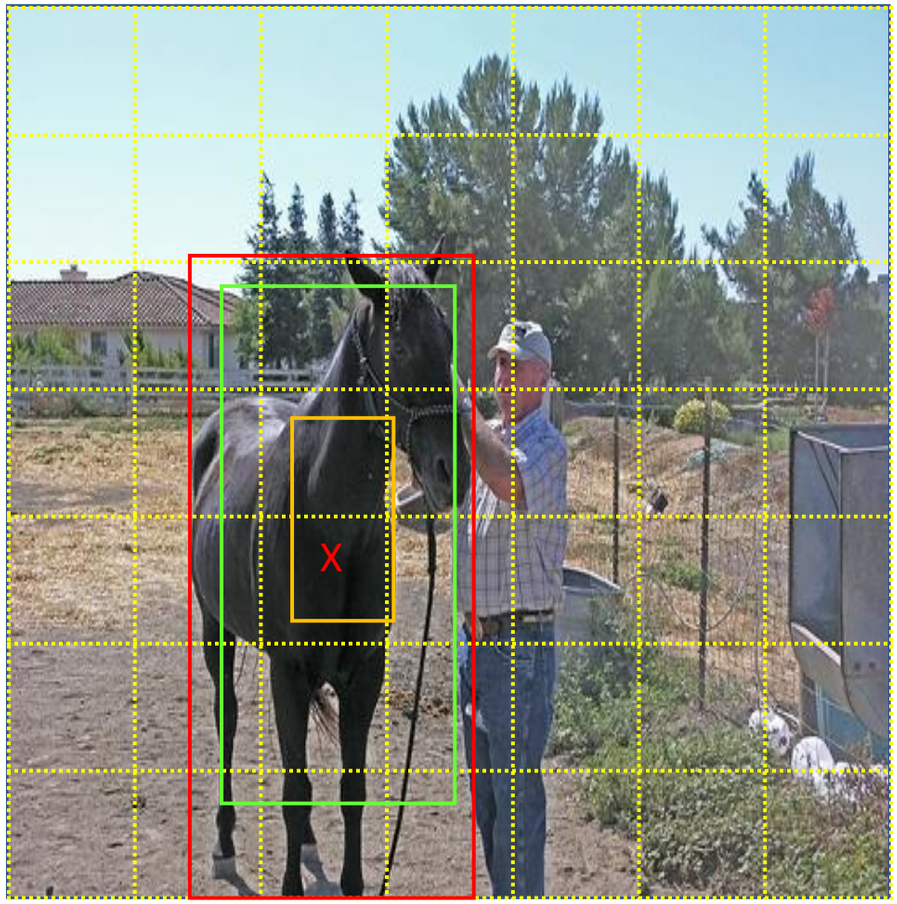
The Training Objectives

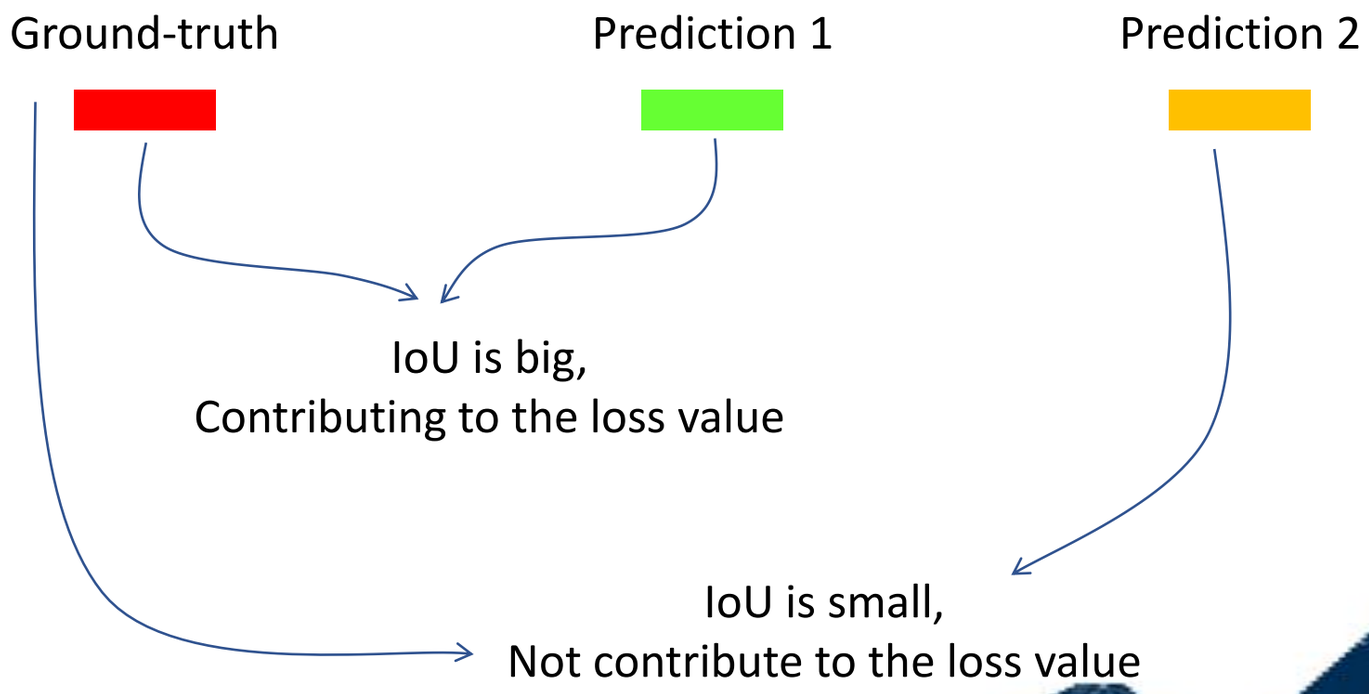


Coordinate Loss

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

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



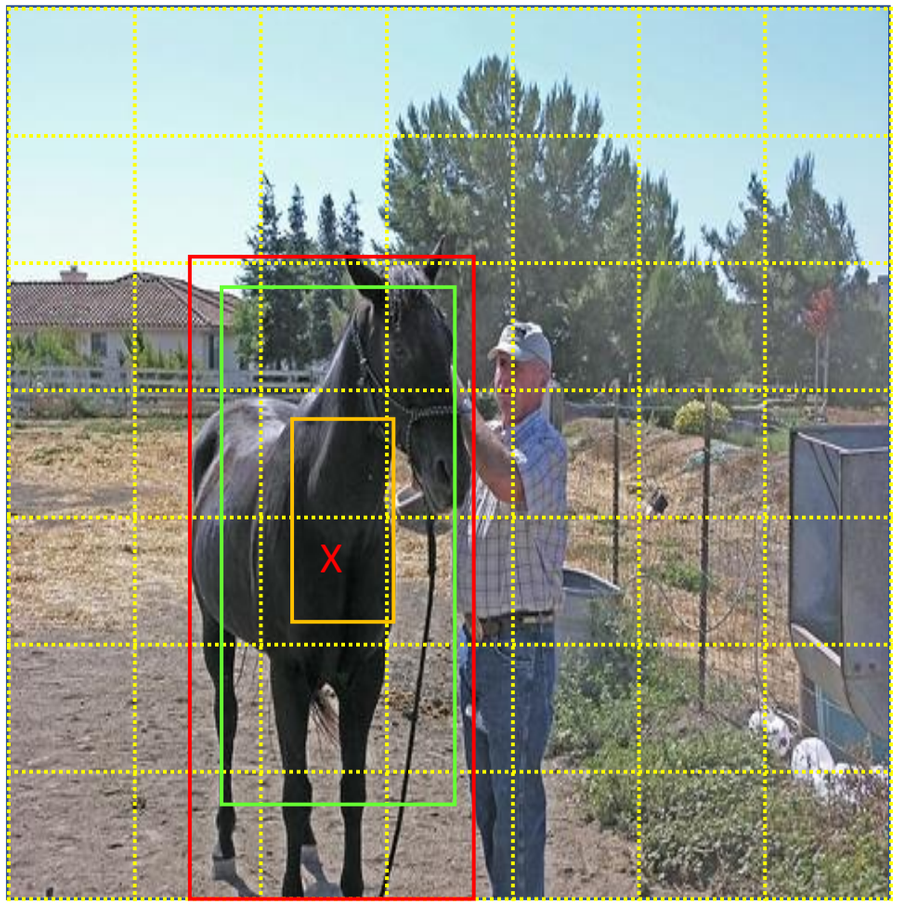


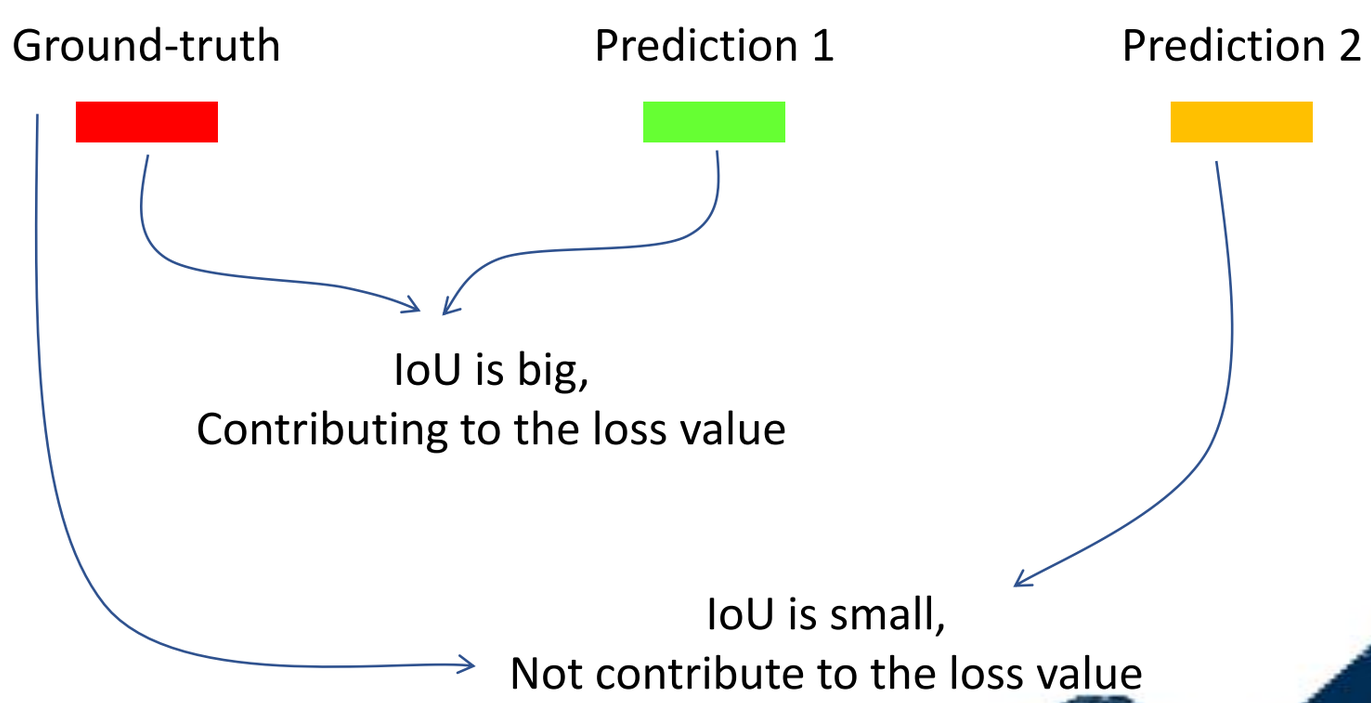
Objectness Loss

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

$$L_{object} = \sum_{c \in \{c | object\ in\ c\ and\ IoU > T\}} \lambda_O \left(O_{p,c} - O_{g,c}\right)^2$$

Value close to 1→ smaller loss

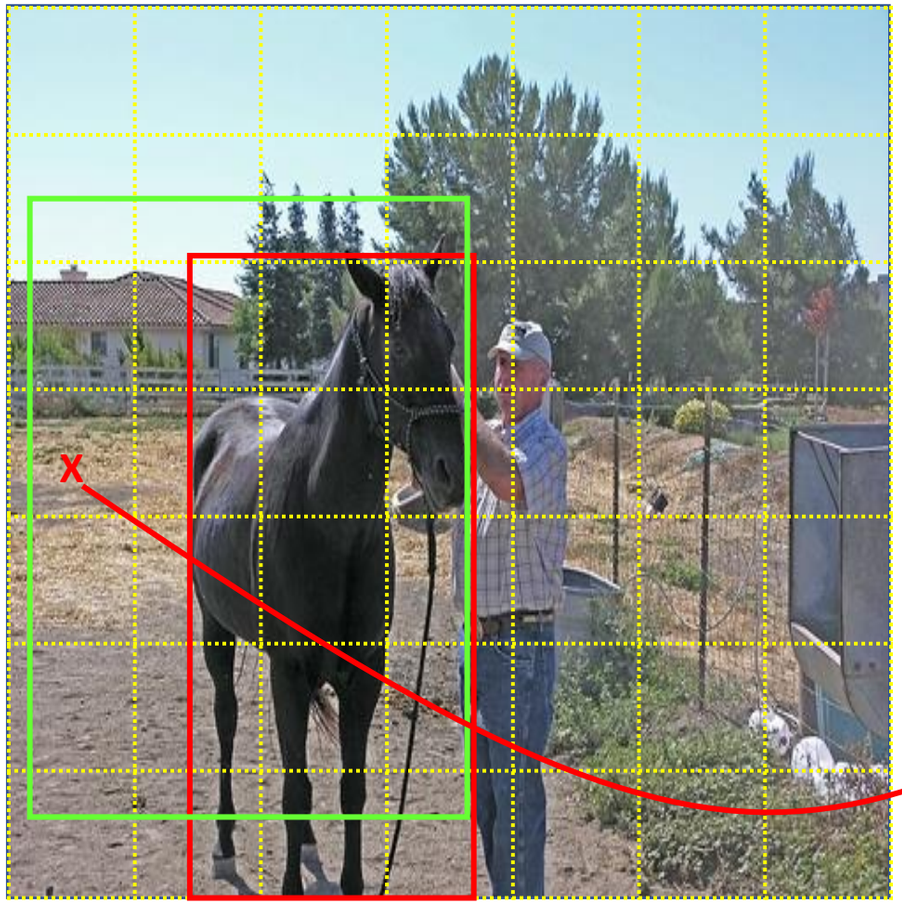




Non-Object Loss

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

$$L_{non\_object} = \sum_{c \in \{c | object\ not\ in\ c\}} \lambda_N \left(O_{p,c}\right)^2$$



Ground-truth

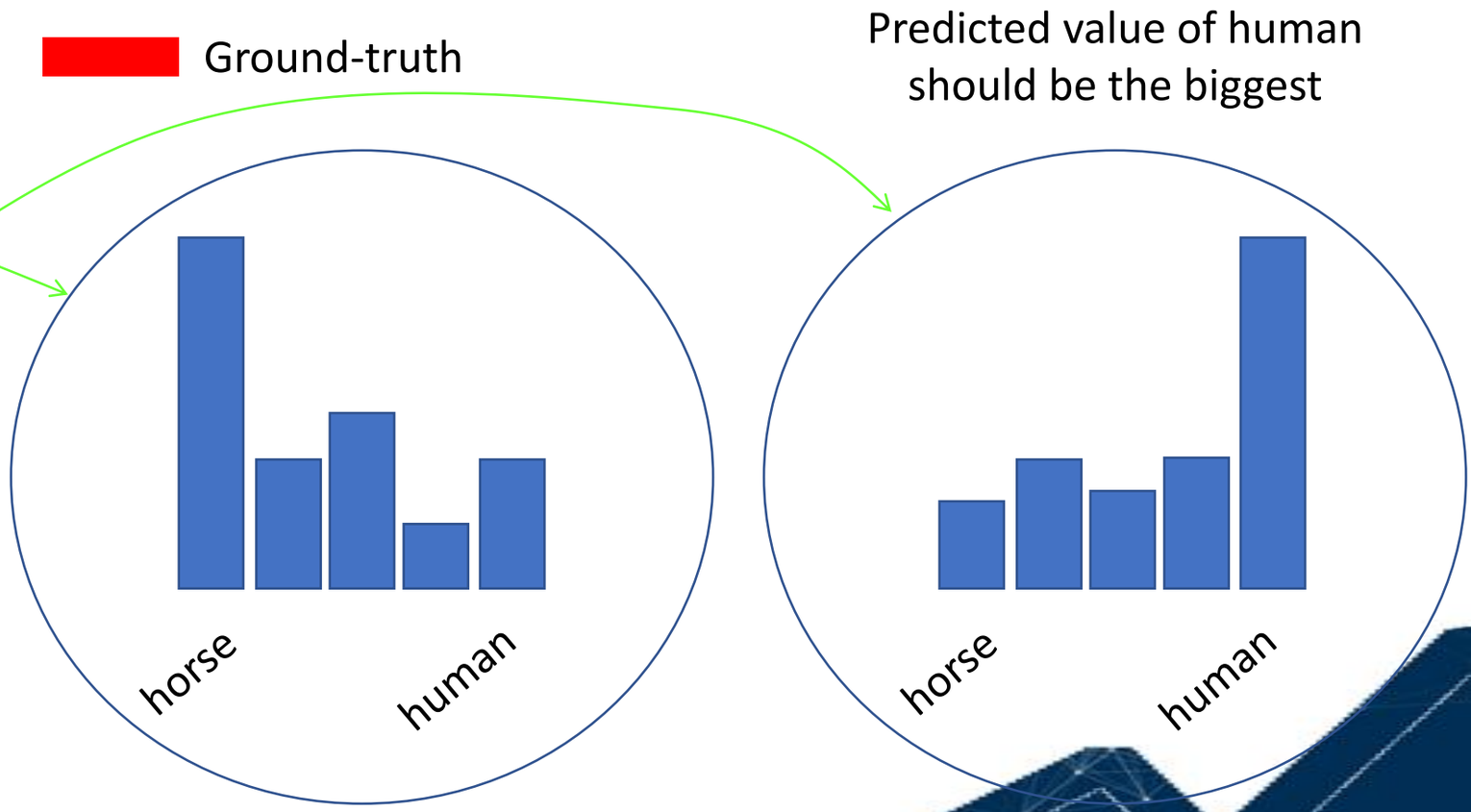
Prediction 1

Objectness value in this cell should be close to zero

Classification Loss

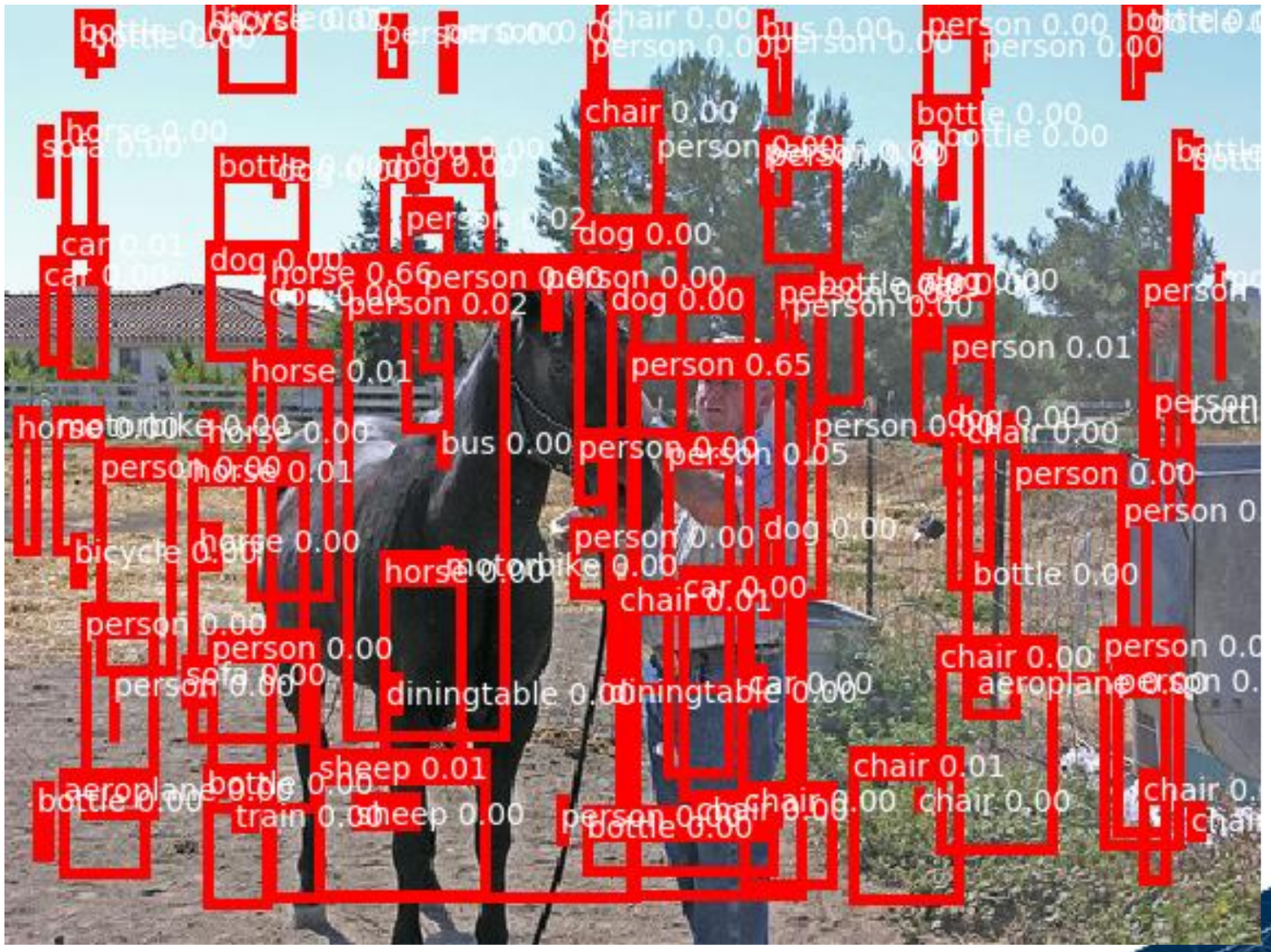
- The predicted class should be similar with ground-truth
  - When object appear in the cell

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



Post-Processing

- Original detection



Post-Processing

- Ignore detection results with small confidence

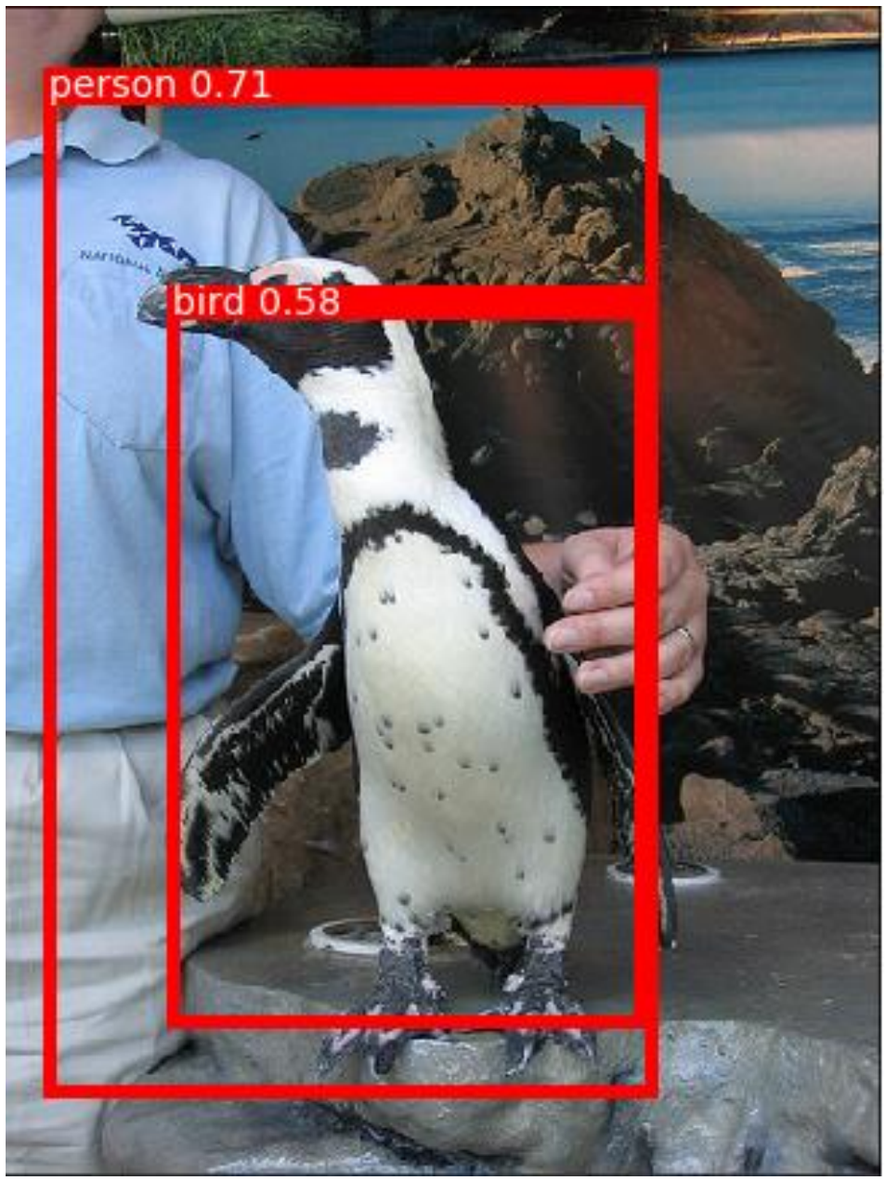
0.02 and 0.05 are too small, but lets keep them for the shake of this explanation

Post-Processing

- Remove overlapped detection based on IoU

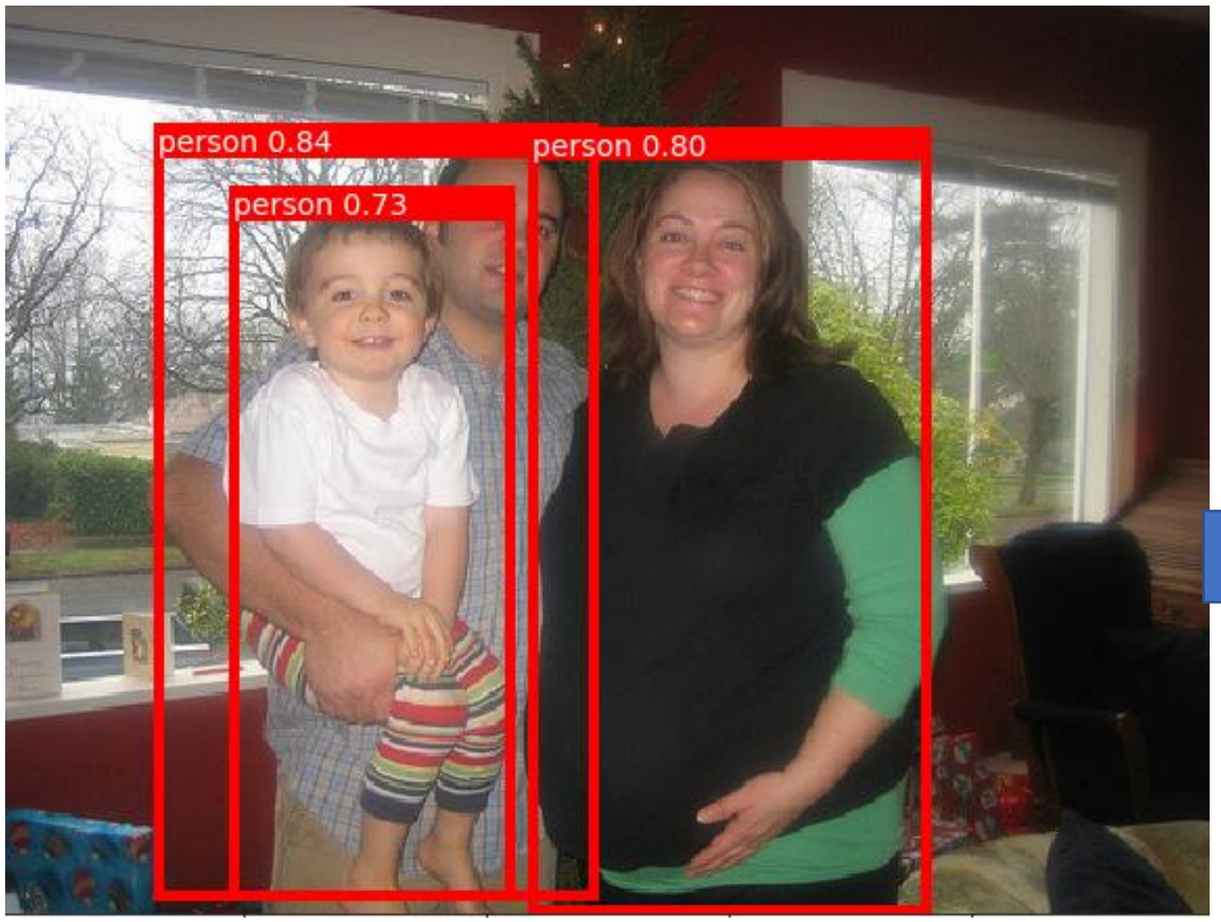
IoU: Intersection over Union

Drawback of IoU Filtering



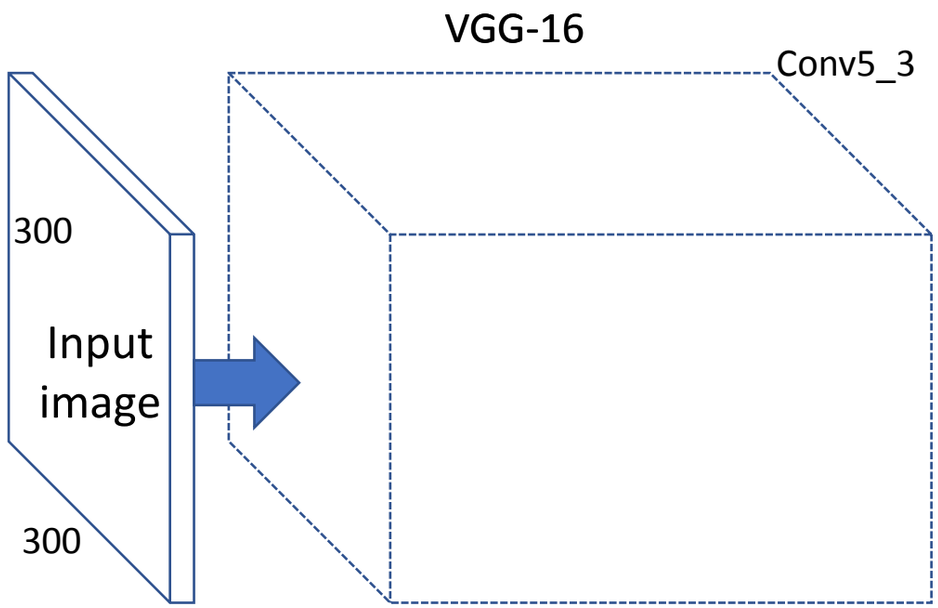
IoU = 0.5499

Drawback of IoU Filtering

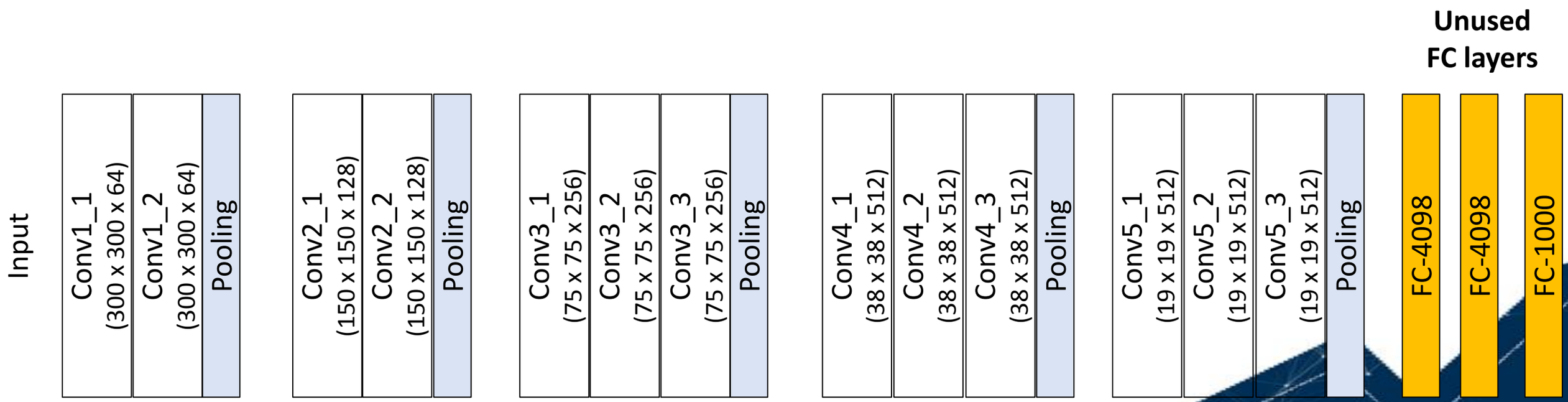


IoU = 0.5774

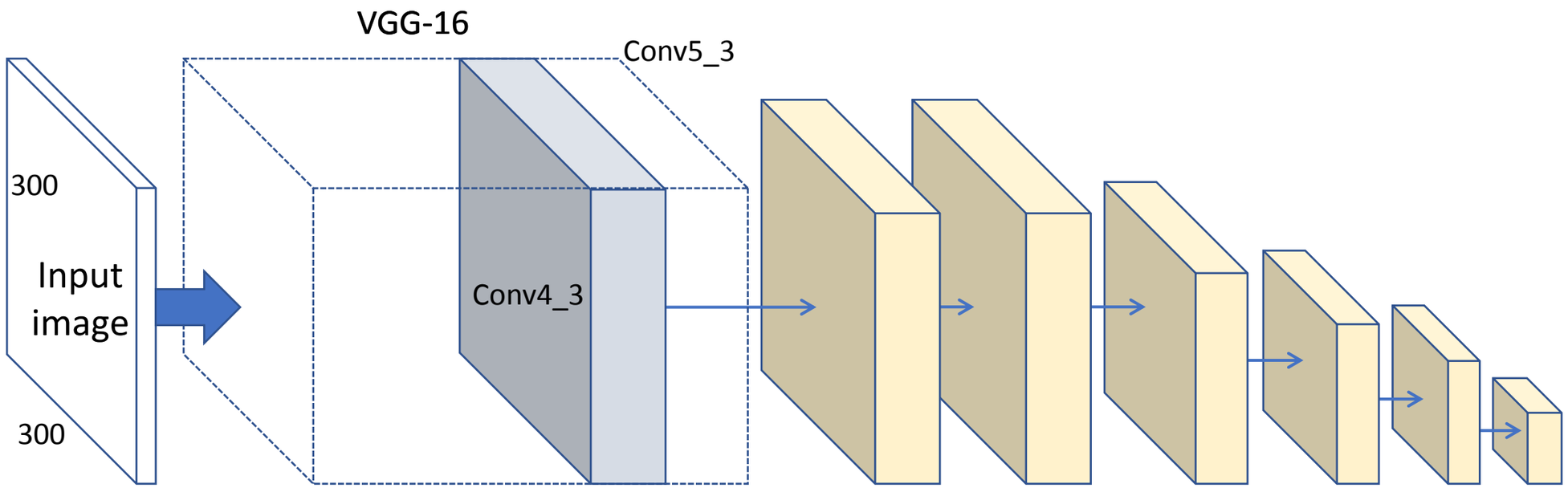
The Single Shot Detector (SSD)



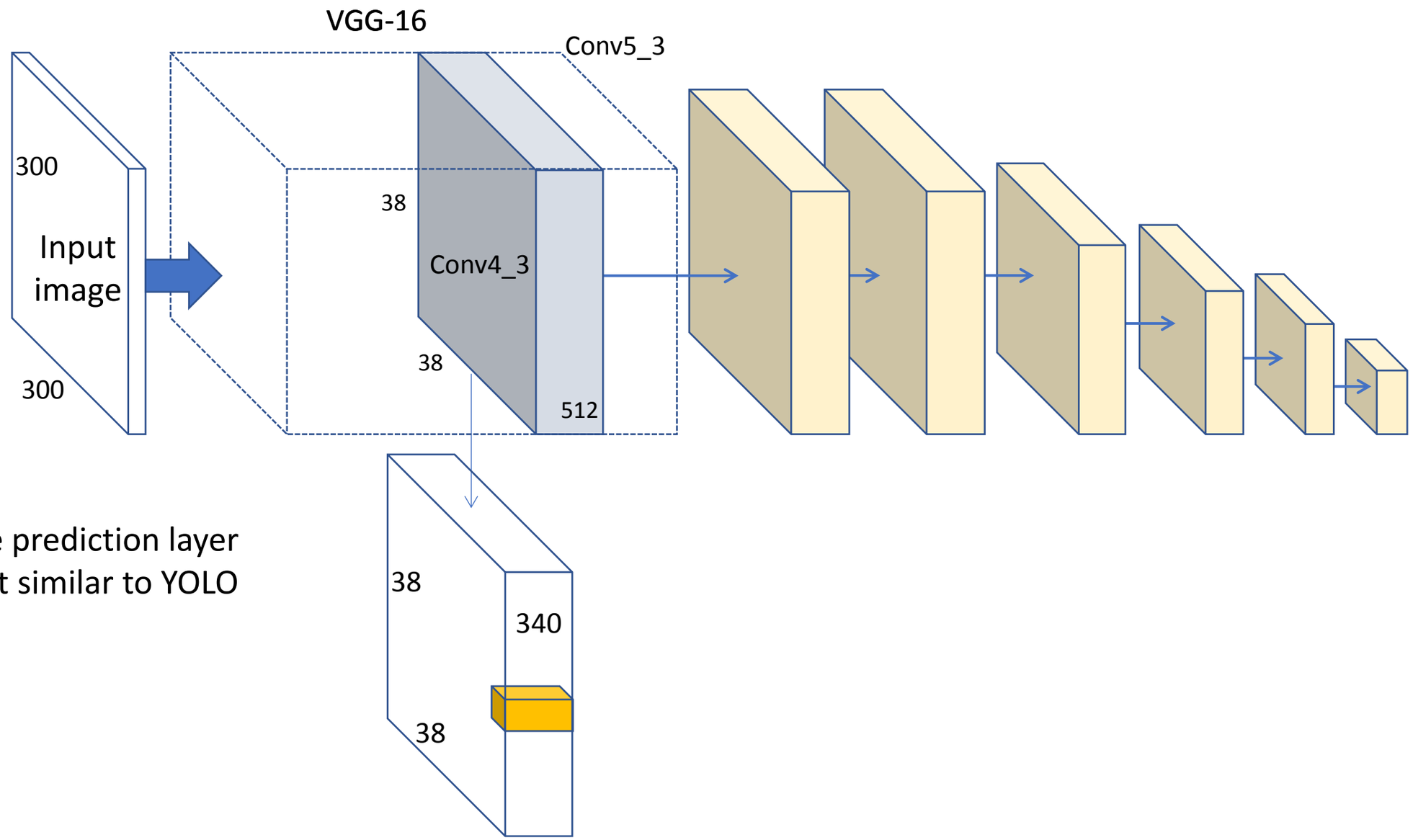
SSD is generalization of the YOLO type detector



The Single Shot Detector (SSD)

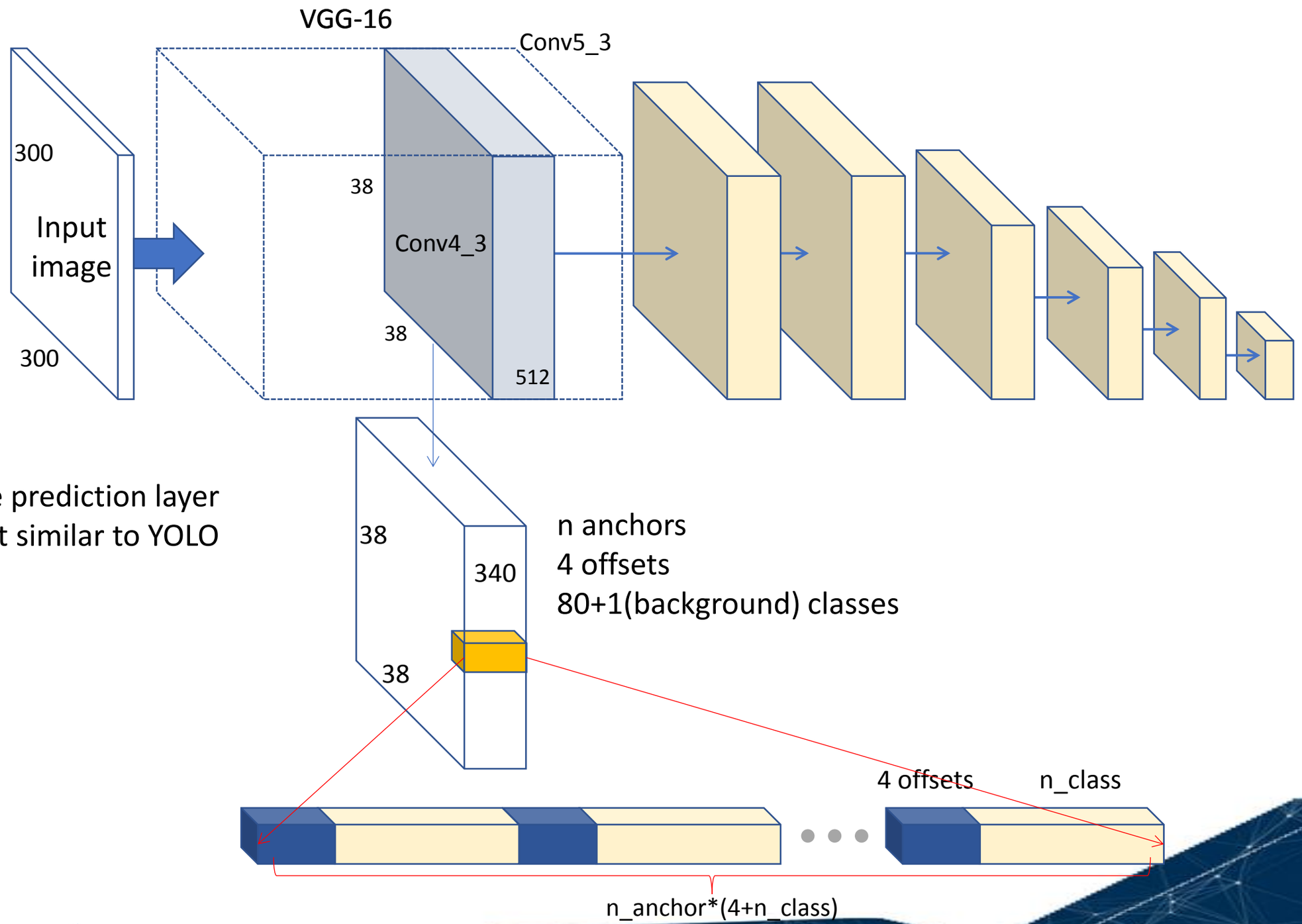


The Single Shot Detector (SSD)



The prediction layer is almost similar to YOLO

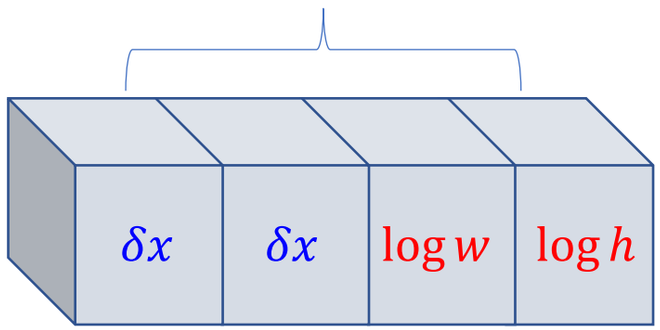
The Single Shot Detector (SSD)



The prediction layer is almost similar to YOLO

SSD - Bounding Box Prediction

4 offsets



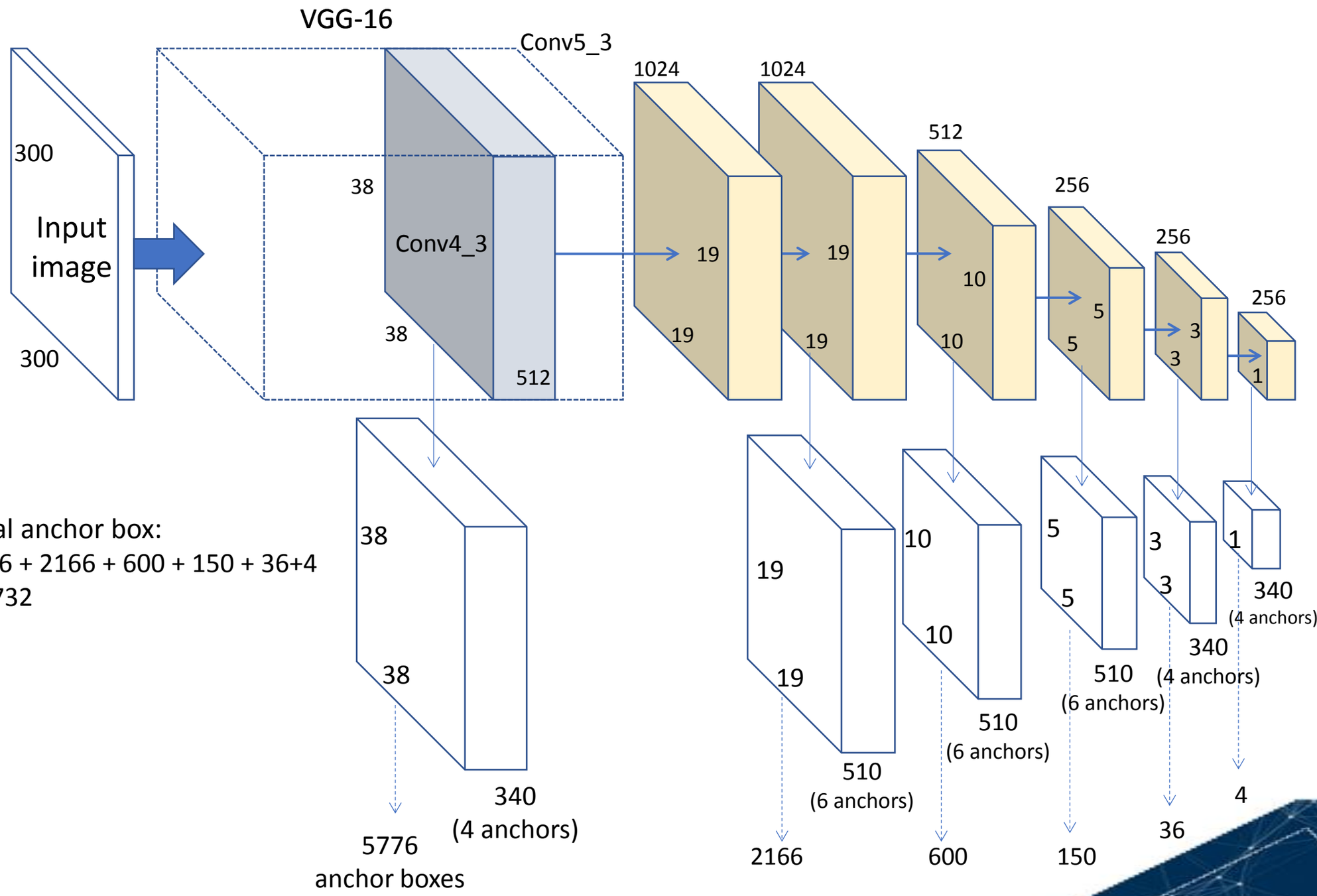
$$P_x = A_x + A_w \delta_x$$

$$P_y = A_y + A_h d_y$$

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

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

The Single Shot Detector (SSD)



Total anchor box:
5776 + 2166 + 600 + 150 + 36+4
= 8732

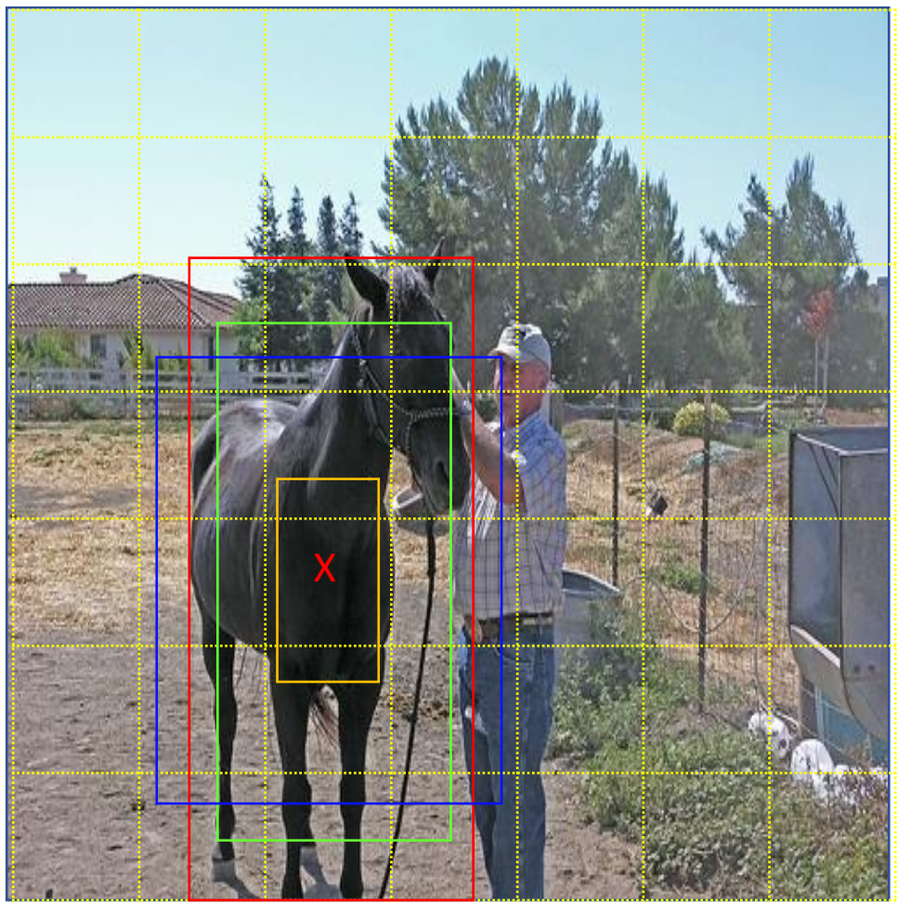
SSD - Training Objectives

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

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

SSD - Losses

- Some anchors are not used for computing loss value



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

Ground-truth

Anchor 1 → Contribute to loss value

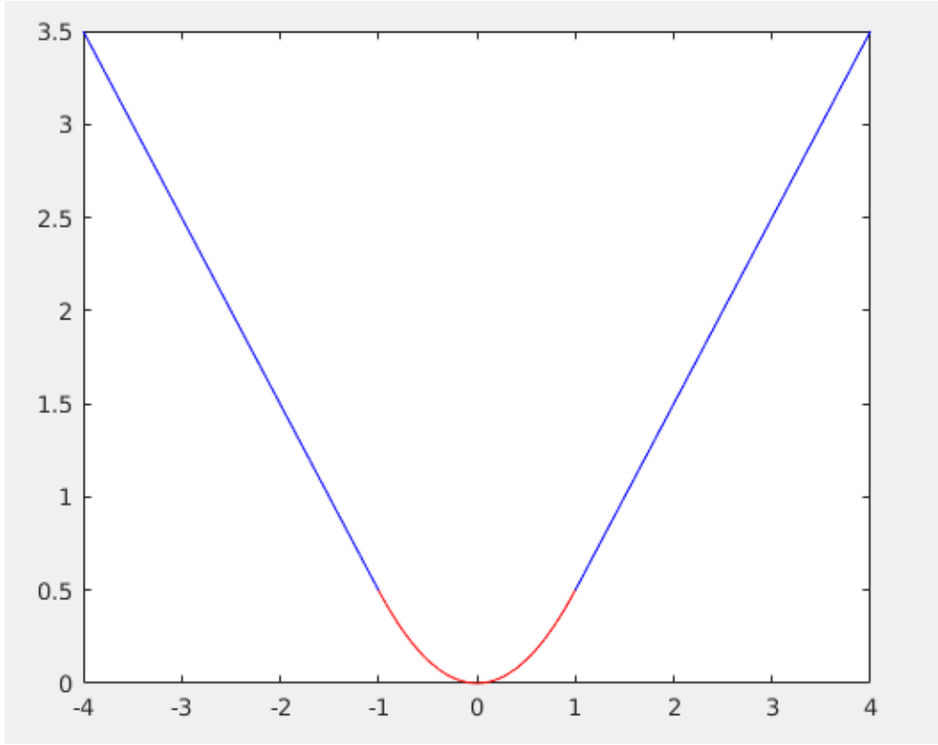
Anchor 2 → NOT contribute to loss value

Anchor 3 → Contribute to loss value

SSD - Losses

- Localization loss

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



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

Generally known as Huber Loss

SSD - Losses

- Confidence Loss:

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

Softmax-ed class prediction value

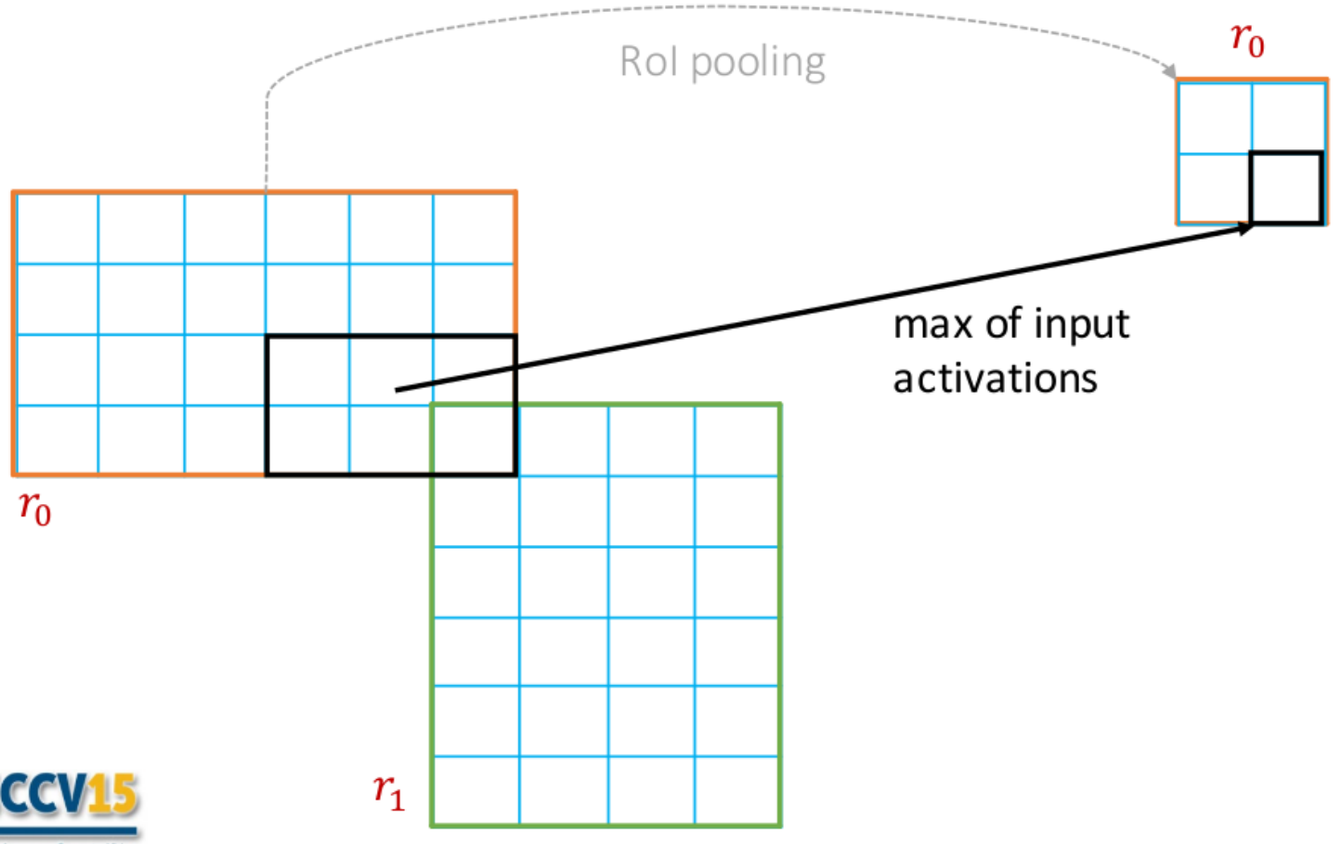
For the matched anchor boxes, prediction should be same with the target value (p)

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

Thank you very much for your attention!

ROI pooling is differentiable

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

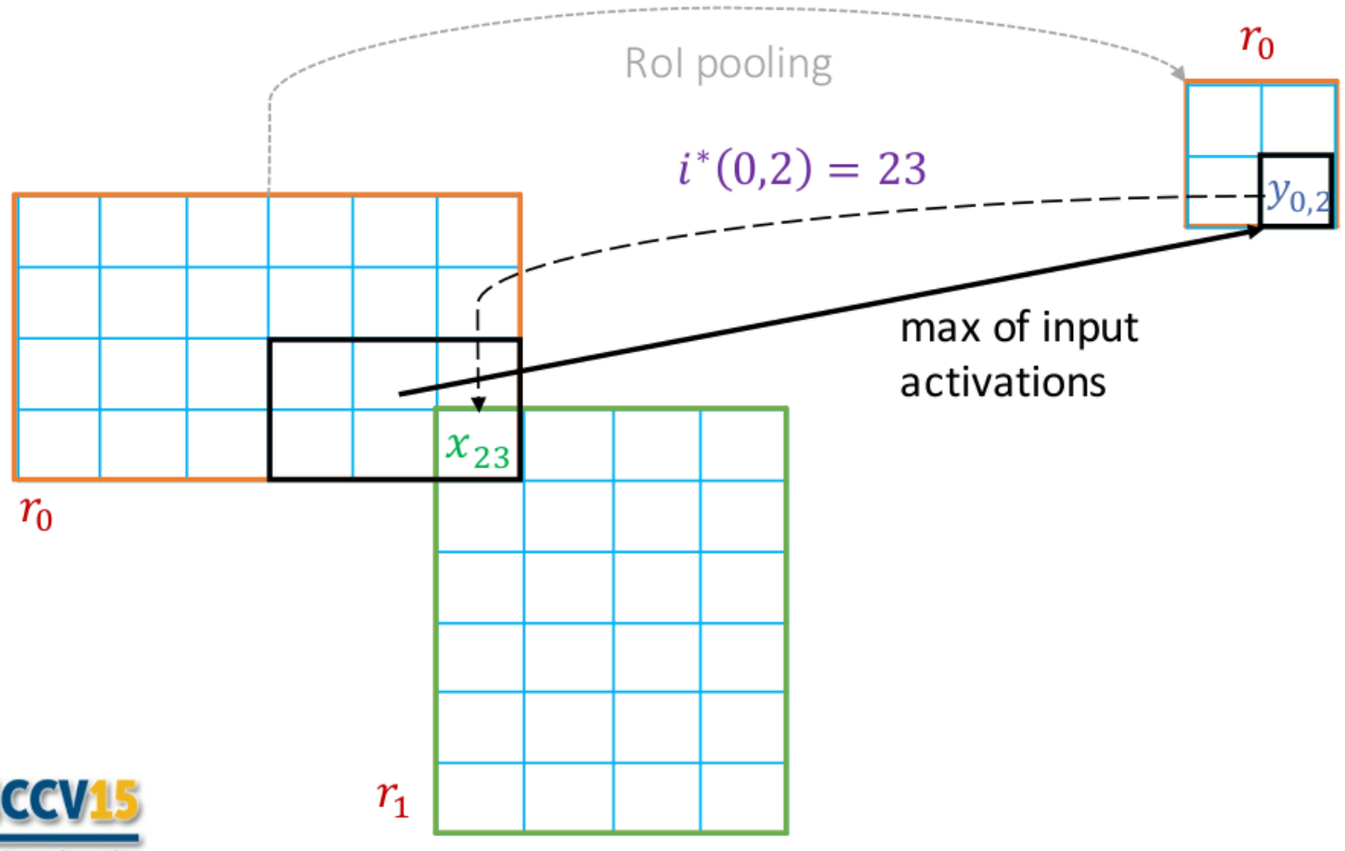


ICCV15 International Conference on Computer Vision

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

ROI pooling is differentiable

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



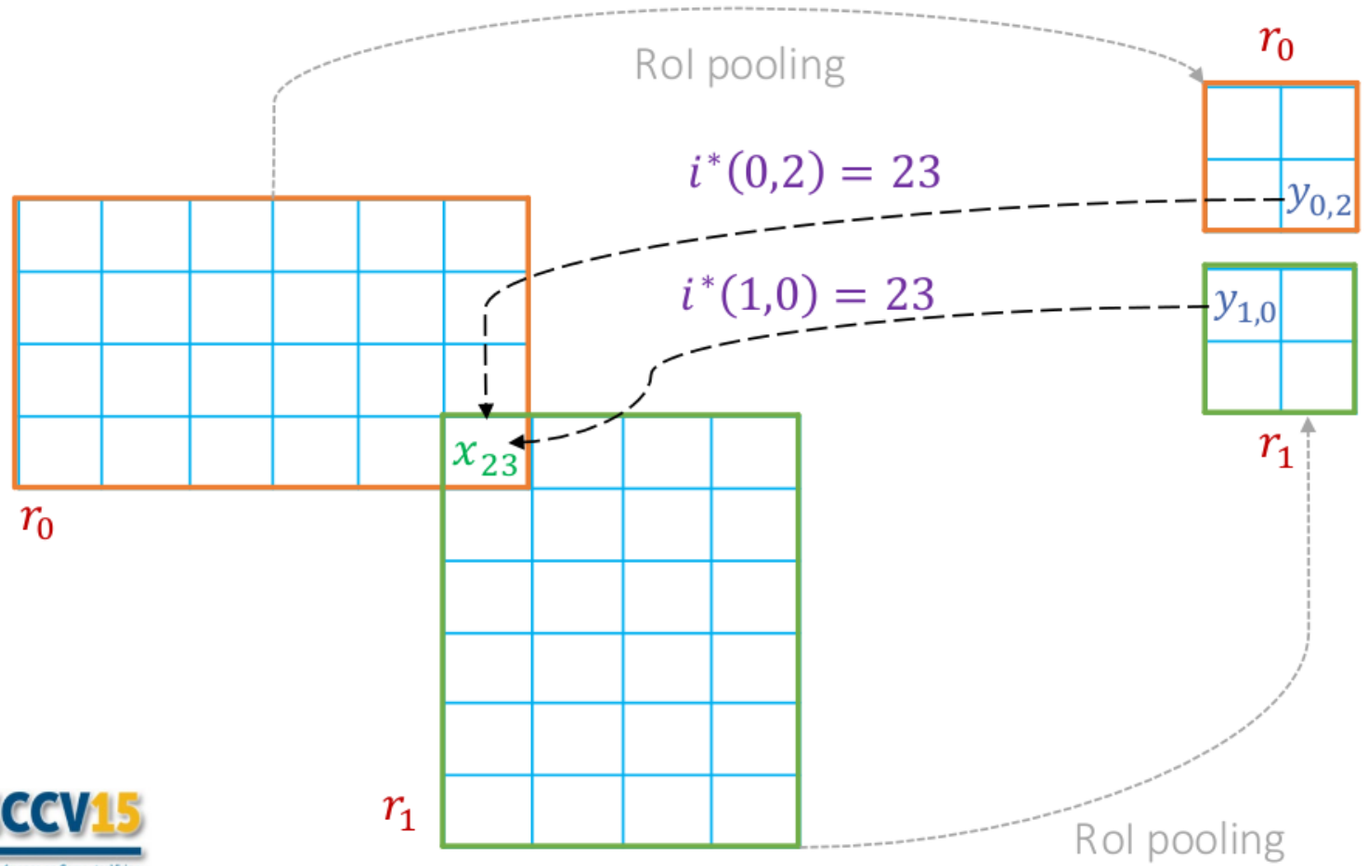
ICCV15 International Conference on Computer Vision

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

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

ROI pooling is differentiable

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



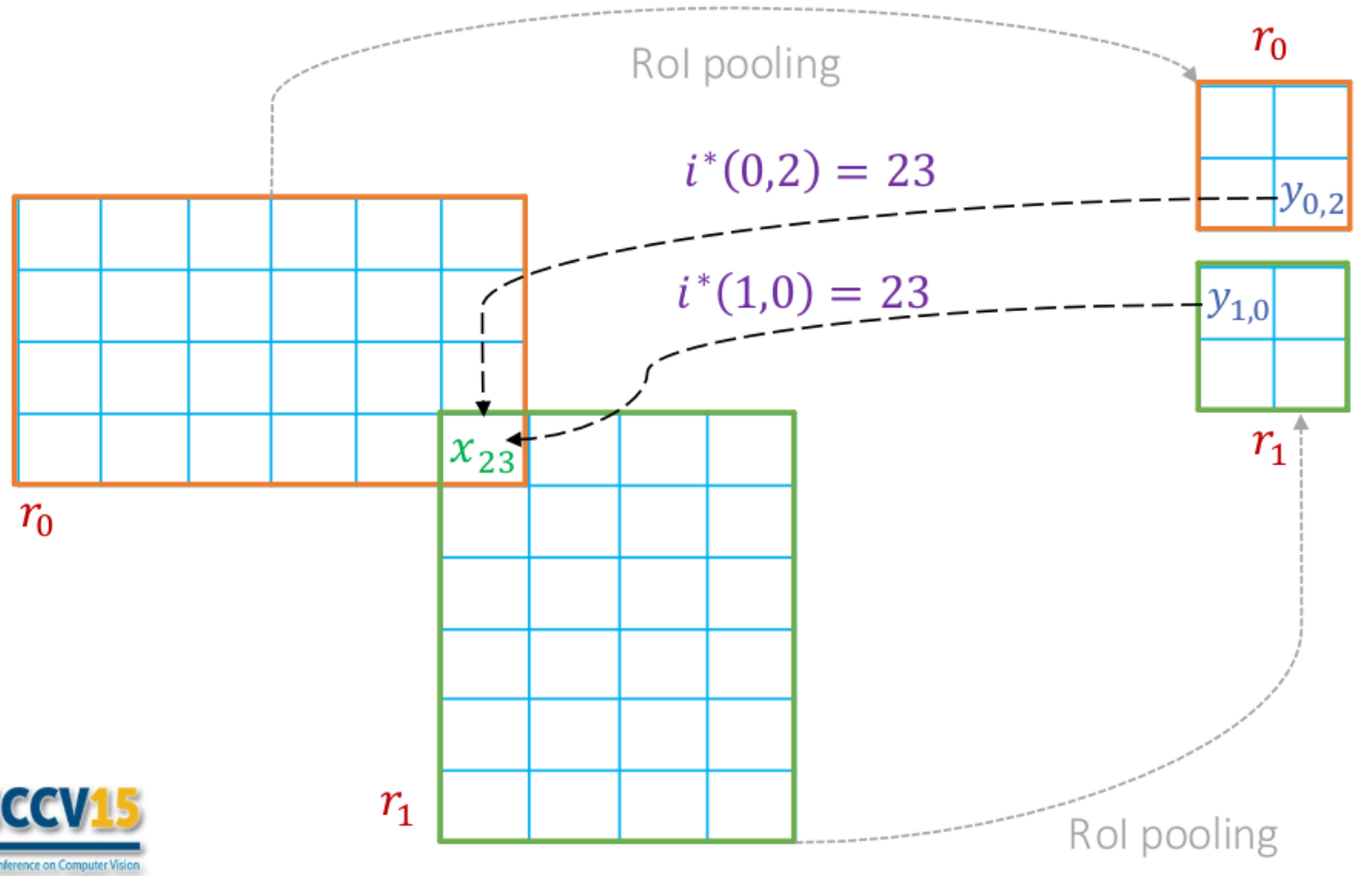
ICCV15 International Conference on Computer Vision

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

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

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 for x_i

Over regions r, locations j

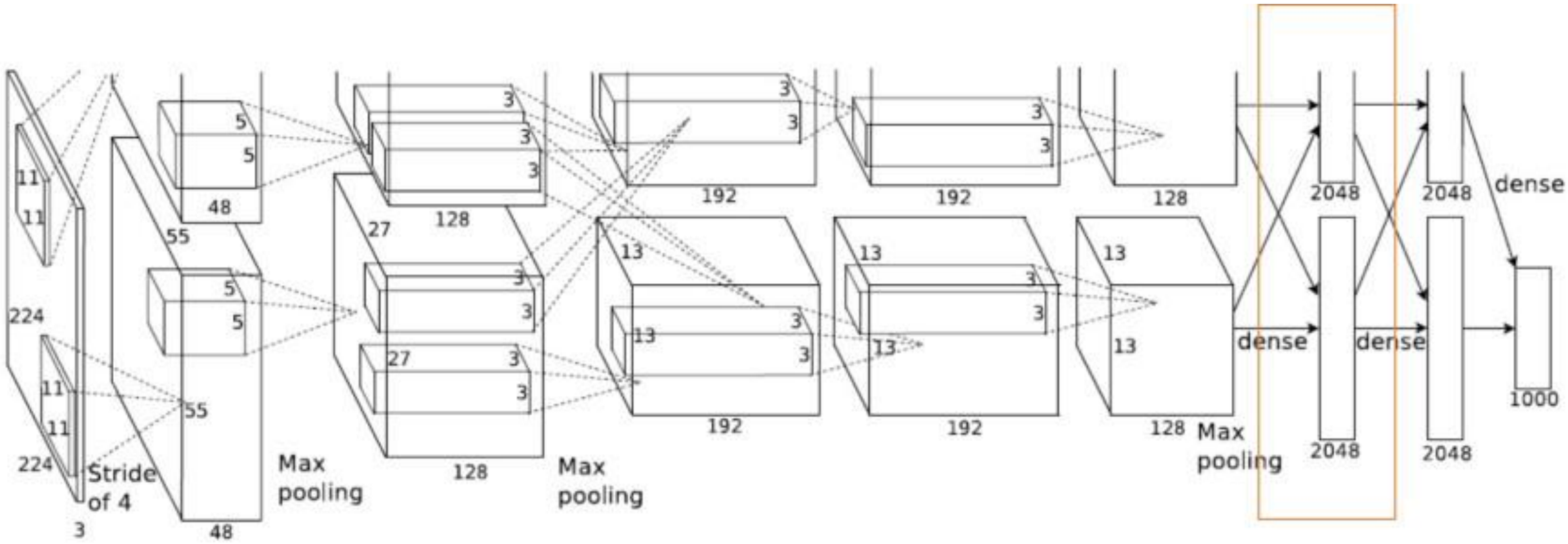
Partial from next layer

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

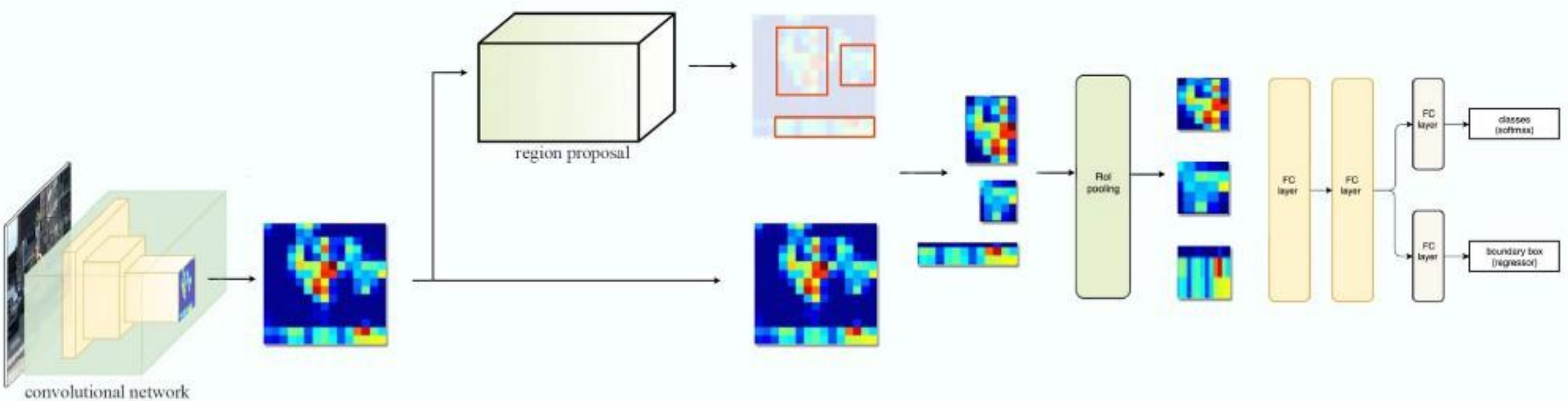
Ross Girshick. "Fast R-CNN". ICCV 2015.

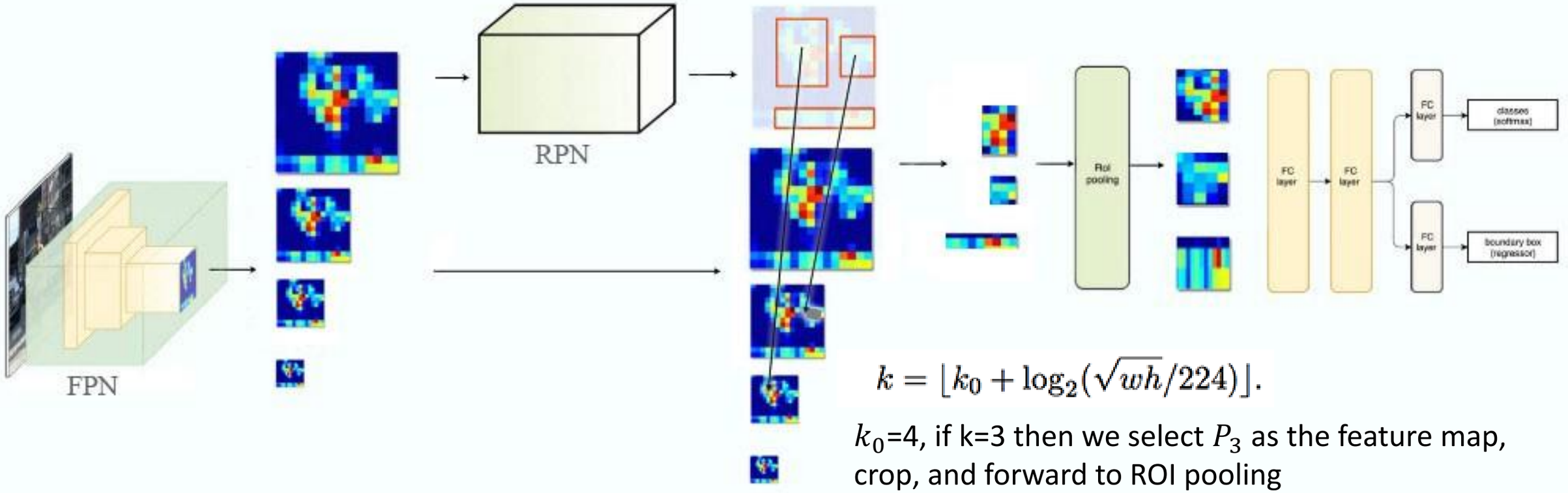
ICCV15 International Conference on Computer Vision

Alexnet


11
11
224
224
3
5
5
48
48
55
55
Stride
of 4
Max
pooling
27
27
128
128
3
3
3
3
3
3
Max
pooling
13
13
13
13
13
13
192
192
192
192
128
128
Max
pooling
2048
2048
2048
2048
dense
dense
dense
1000

FPN



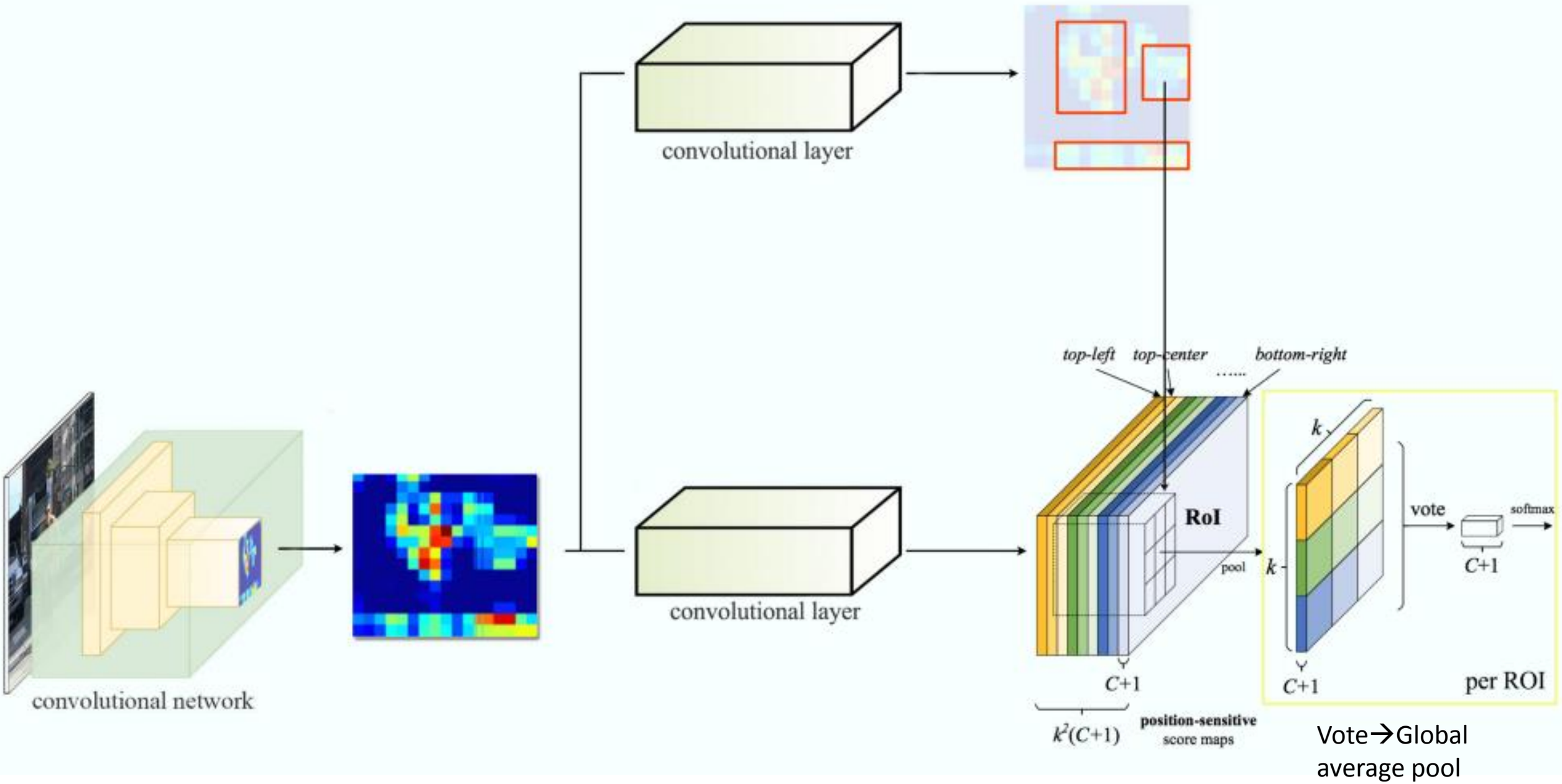


$$k = \lfloor k_0 + \log_2(\sqrt{wh}/224) \rfloor.$$

k_0=4, if k=3 then we select P_3 as the feature map, crop, and forward to ROI pooling

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

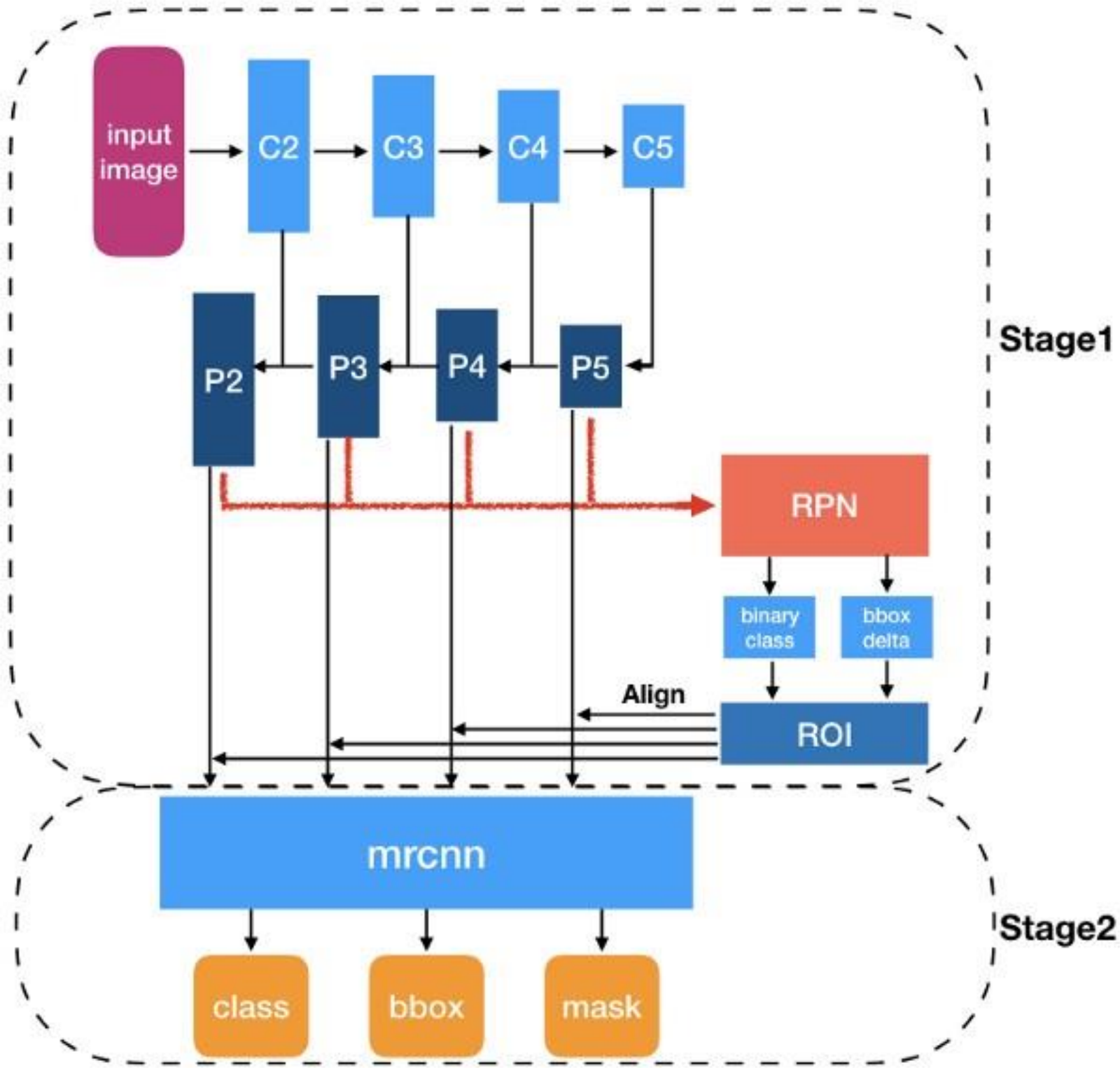
R-FCN



Vote→Global average pool

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

Mask R-CNN



https://medium.com/@alittlepain833/simple-understanding-of-mask-rcnn-134b5b330e95