

CS181: Computer Vision

Deep Learning-based Object Detection

Thursday, October 1st, 2026



Md Alimoor Reza

Associate Professor of Computer Science

Announcements

- Notebook #3: Underwater Animal Classification
 - It's a 10-class image recognition problem.
 - to submit, download the `.ipynb` file from Colab
 - directly upload to Blackboard
 - [due tomorrow \(October 2nd\)](#)
- Quiz#1 has been released two days ago
 - you will be able to take it offline on Blackboard
 - should be taken [within a fixed-hour window](#) (10 hours) during which you are expected to complete the quiz individually. You are free to start anytime within the seven-day period (released on [September 29th](#) and [due by October 7th](#))

What type of recognition problem?



Object Recognition

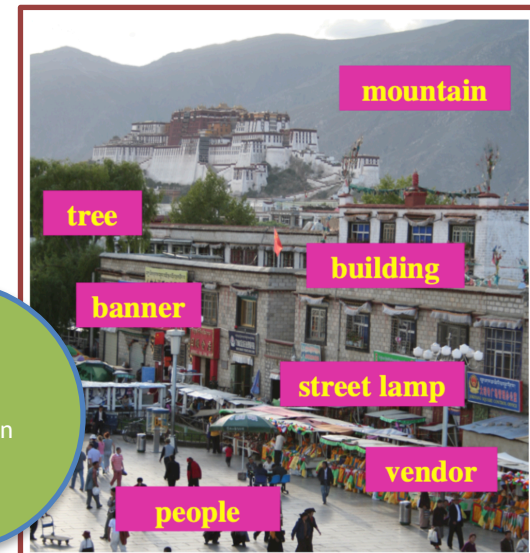


What type of recognition problem?



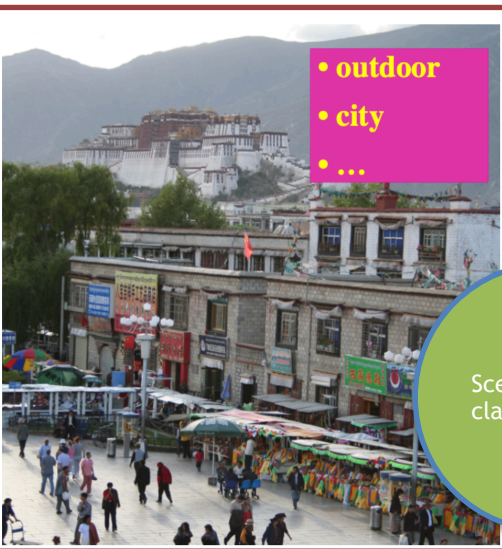
Single label classification

What type of recognition problem?



Multi-label classification

What type of recognition problem?



Scene and context classification

Image Classification: what are the objects in this image?



<u>Label</u>		<u>Probability</u>
Person	+	0.99
Building		0.01
Cat		0.00
Dog		0.01
...		...
...		...
...		...

Input:

1. an image



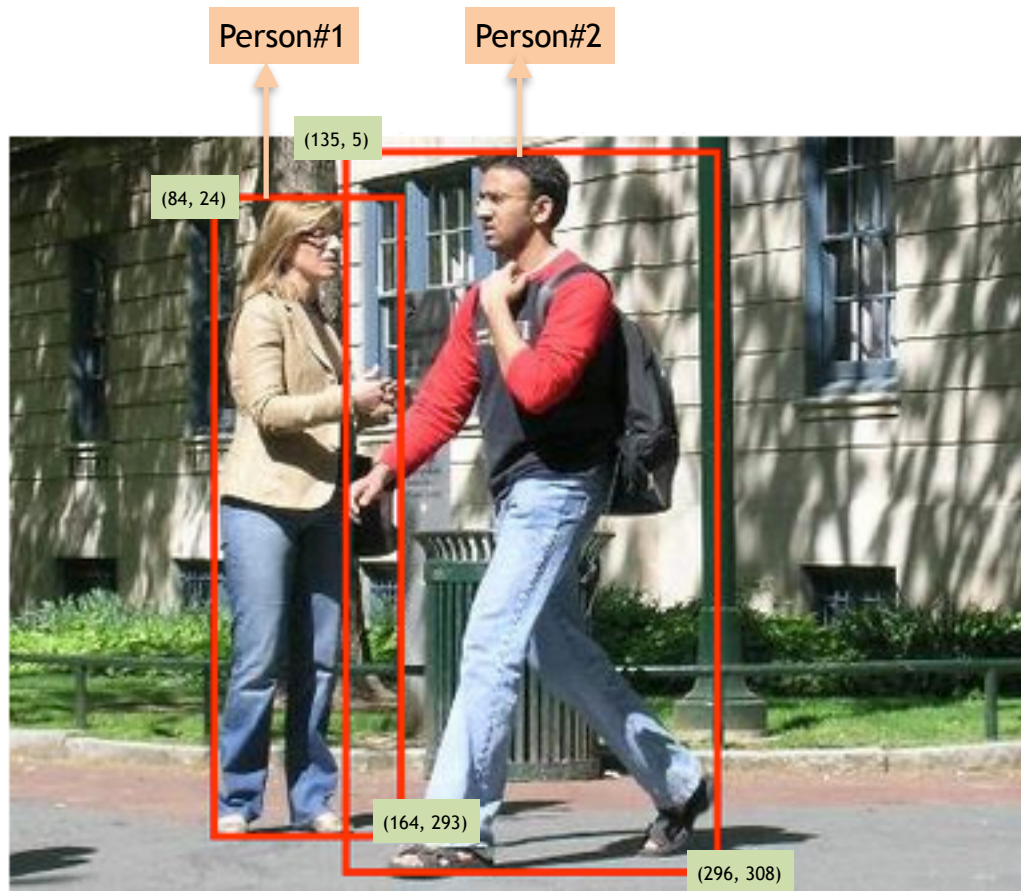
Classification model



Output:

1. Labels of prediction
2. Probability of labels

Object Detection: what are the locations of objects?



[xmin, ymin, xmax, ymax]

<u>Label</u>	<u>Probs.</u>	<u>Bounding Box Coordinates</u>
Person#1	0.99	[84, 24, 164, 293]
Person#2	0.99	[139, 5, 296, 308]
...	...	...
...	...	...
...	...	...
...	...	...
...	...	...
...	...	...

Input:
1. an image



Detection model



Output:
1. Labels of prediction
2. Probabilities of labels
3. Locations of rectangular bounding boxes associated with each label

Deep learning-based object detections

- **R-CNN**
- Fast RCNN
- Faster RCNN

- Mask RCNN

- YOLO
- SSD

Popular CNN: AlexNet (quick review)

- AlexNet's important features. Similar framework to LeNet with important distinction such as:

- Bigger model compared to LeNet
 - 5 convolution layers + 3 linear layers
 - 60,000,000 params
- Trained on more data
 - 10^6 images (ImageNet) vs. 10^3 images (digit images)
- GPU implementation (50x speedup over CPU)
 - trained on two GPUs for a week
- Better regularization for training
 - introduced a new technique called **Dropout**
 - **Dropout**: randomly turning off neurons from the network

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

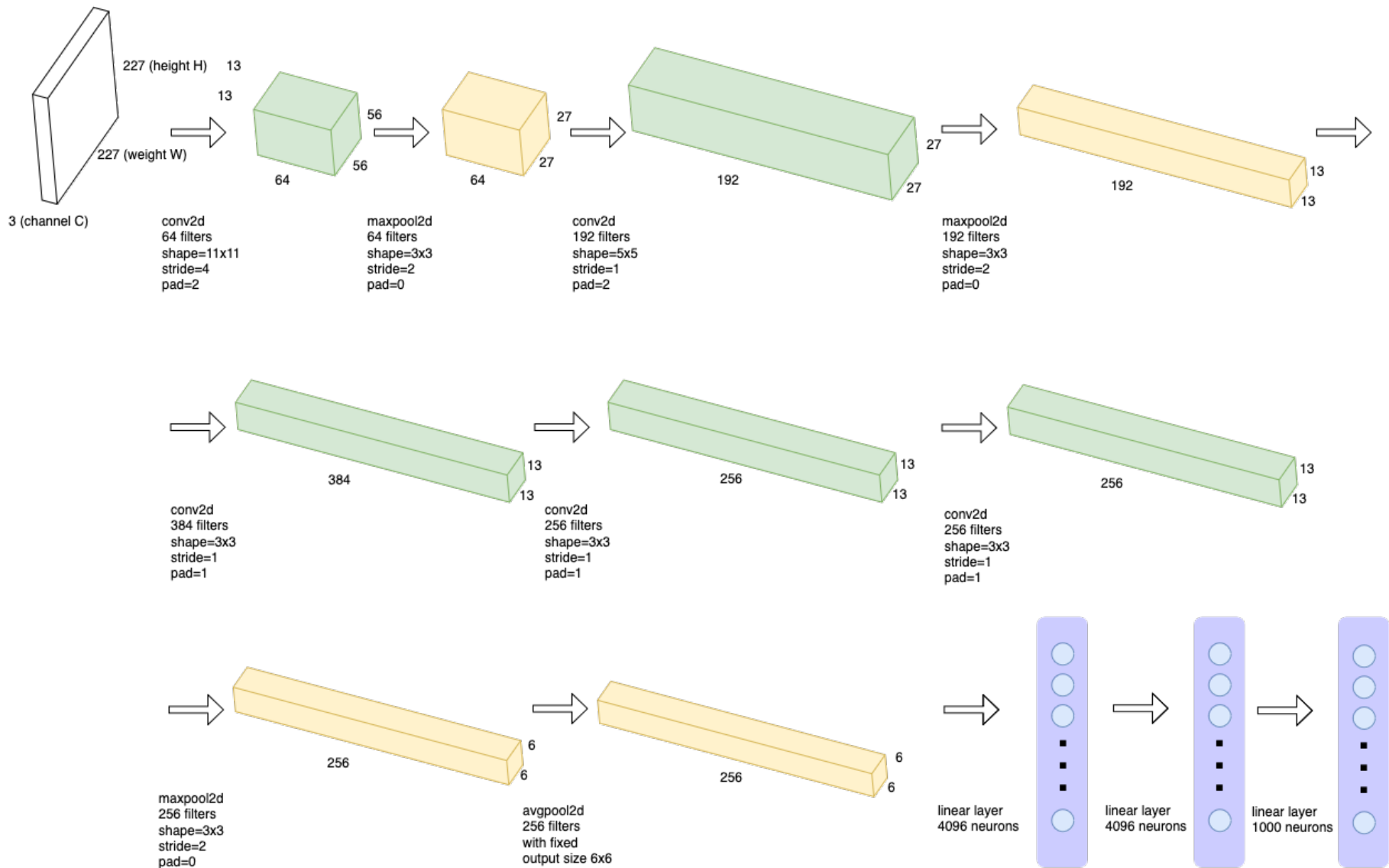
Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet ILSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

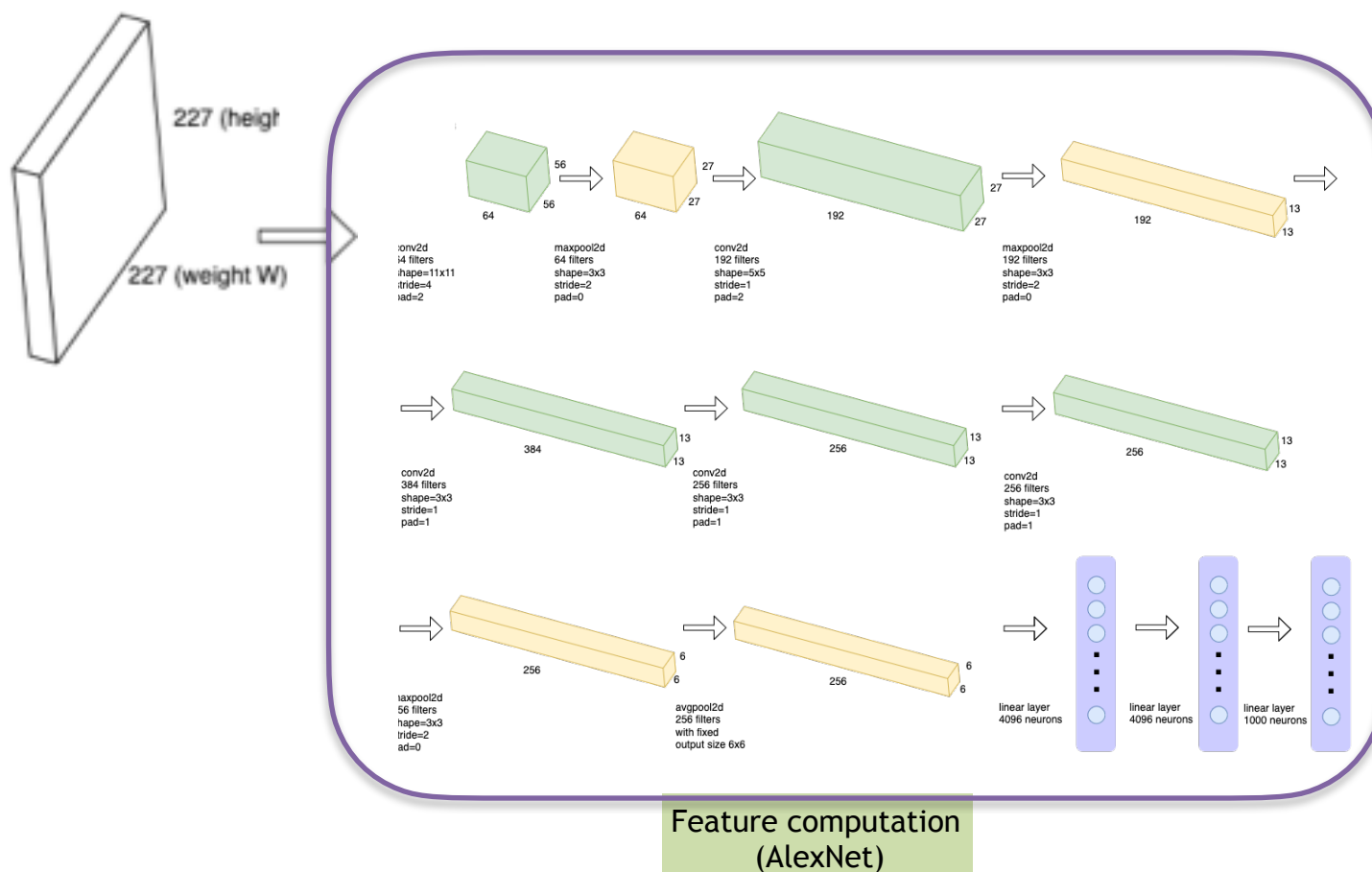
Popular CNN: AlexNet (quick review)



A. Krizhevsky, I. Sutskever, and G. Hinton, [ImageNet Classification with Deep Convolutional Neural Networks](#), NIPS 2012

Popular CNN: AlexNet (quick review)

- AlexNet's can be used as a feature extractor
 - What is the output if I feed a volume of size $(3 \times 227 \times 227)$ as an input?



Deep learning-based object detections

- **R-CNN**
- Fast R-CNN
- Faster R-CNN

- Mask R-CNN

- YOLO
- SSD

Object Detector#1: R-CNN

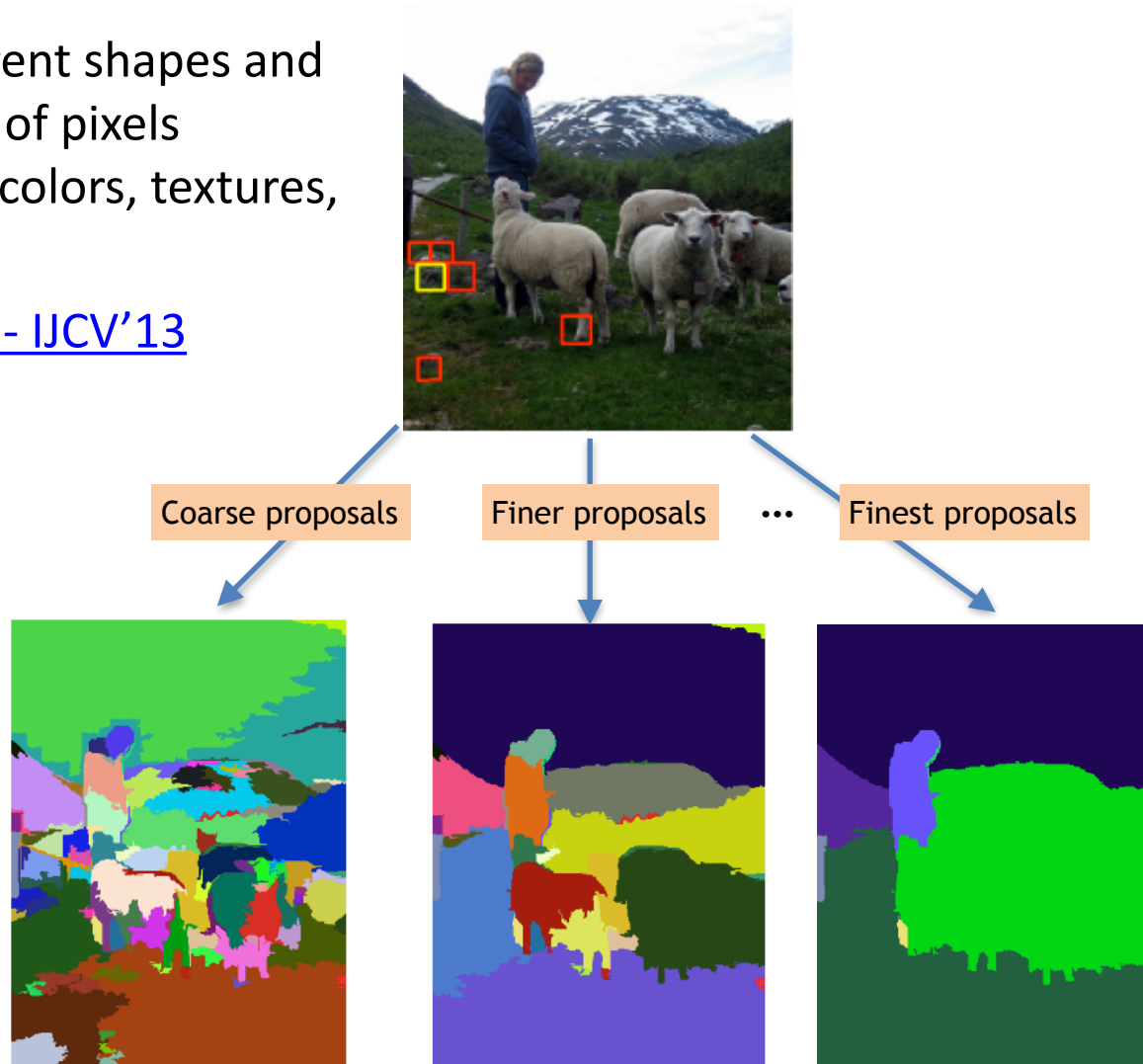
- Generate a lot of (**object candidate**) proposals
- Extract features for each proposal
 - 2000 proposals of various sizes
 - 4096 dimensional vector from AlexNet's last layer for each proposal
- Classify the it into one of the classes using SVM

[Rich feature hierarchies for accurate object detection and semantic segmentation - Girshick et al. CVPR'14](#)

Object Detector#1: R-CNN

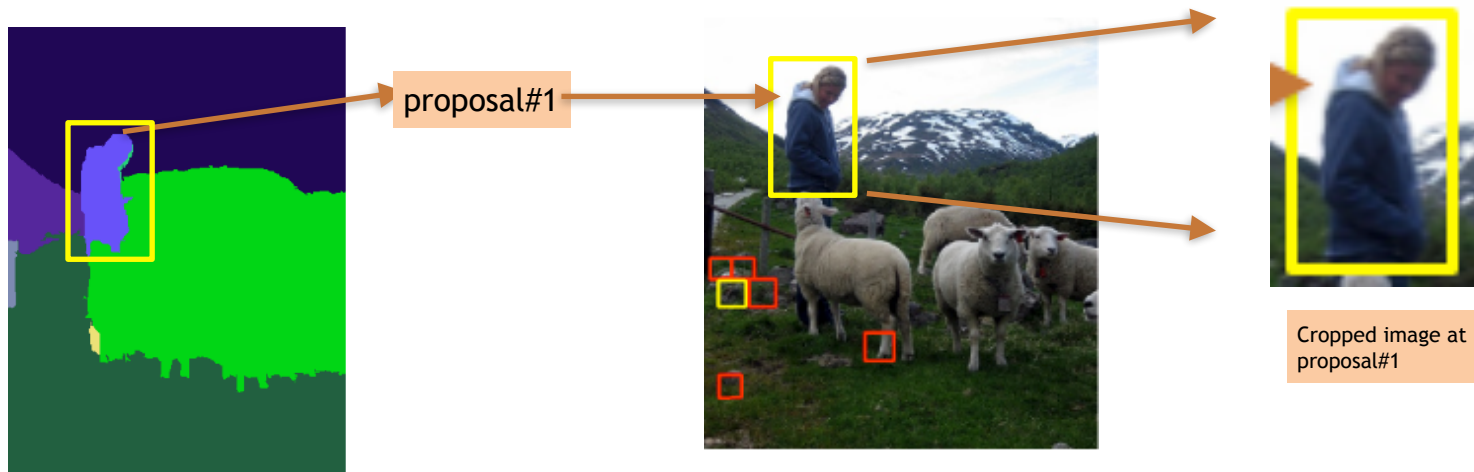
- Generate proposals of different shapes and sizes based on the grouping of pixels together by looking at their colors, textures, and other features

Reference: [Selective Search - IJCV'13](#)



Object Detector#1: R-CNN

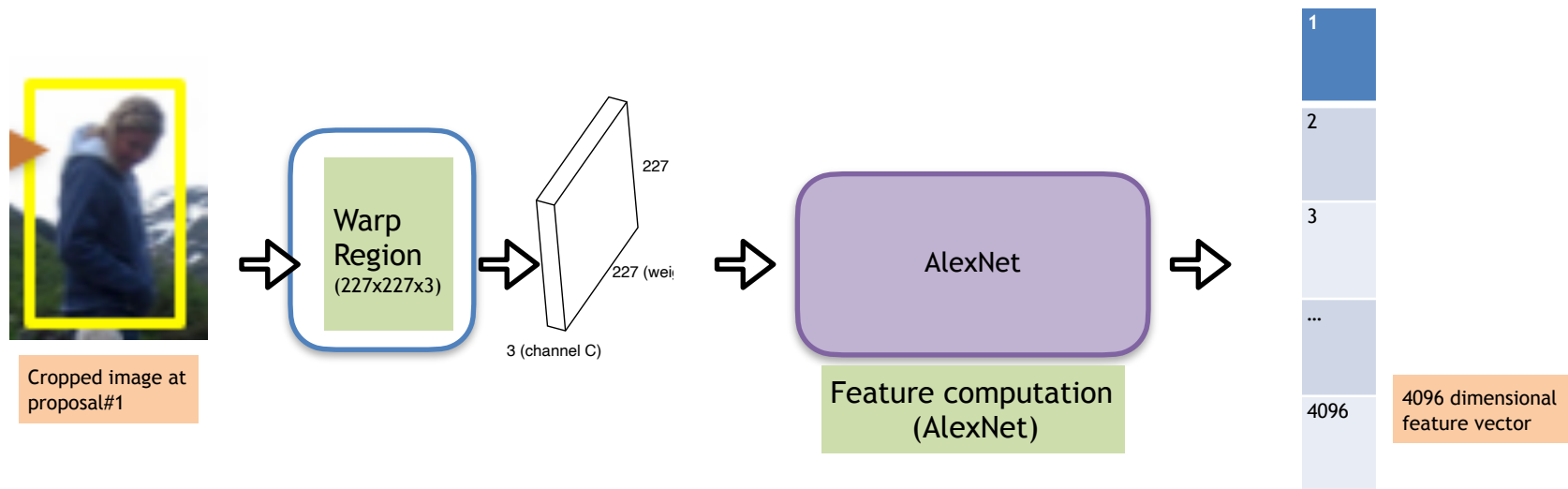
- Extract features for each proposal
 - 2000 proposals of various sizes
 - 4096 dimensional vector from AlexNet's last layer for each prop



Example: proposal#1 feature extraction

Object Detector#1: R-CNN

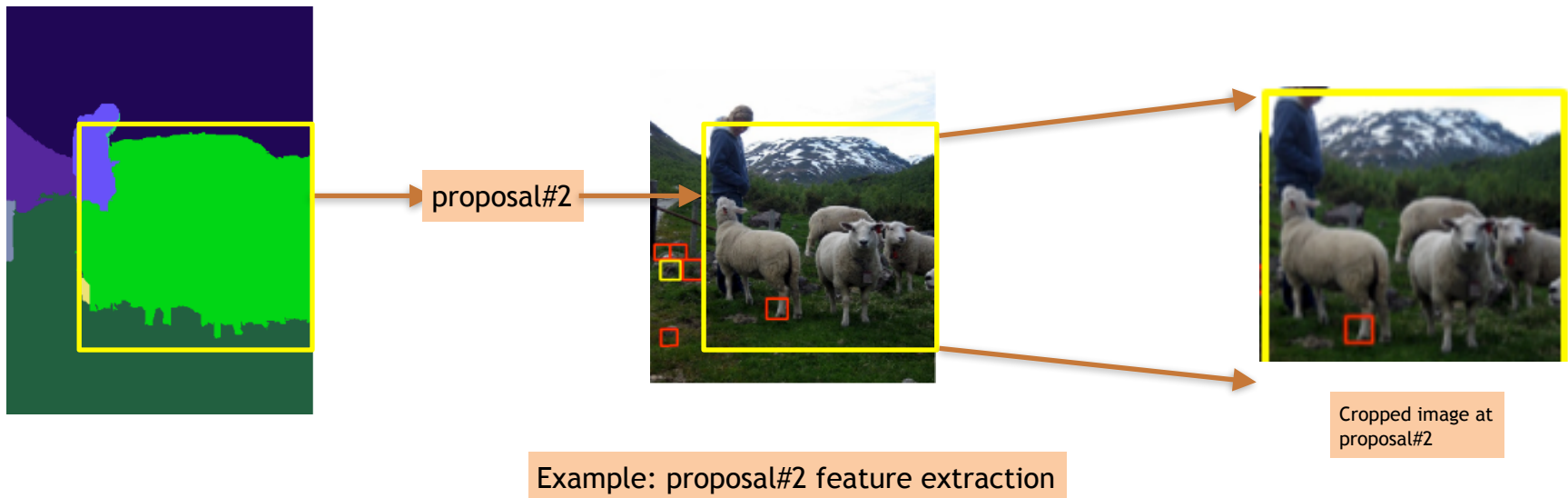
- Extract features for each proposal
 - 2000 proposals of various sizes
 - 4096 dimensional vector from AlexNet's last layer for each prop



Example: proposal#2 feature extraction

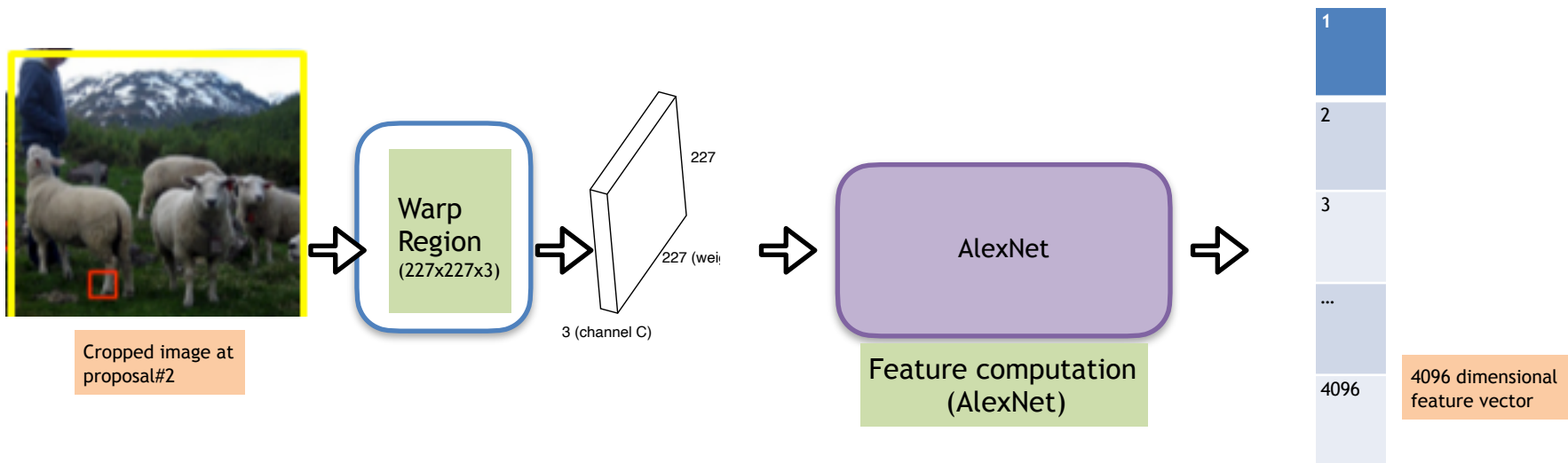
Object Detector#1: R-CNN

- Extract features for each proposal
 - 2000 proposals of various sizes



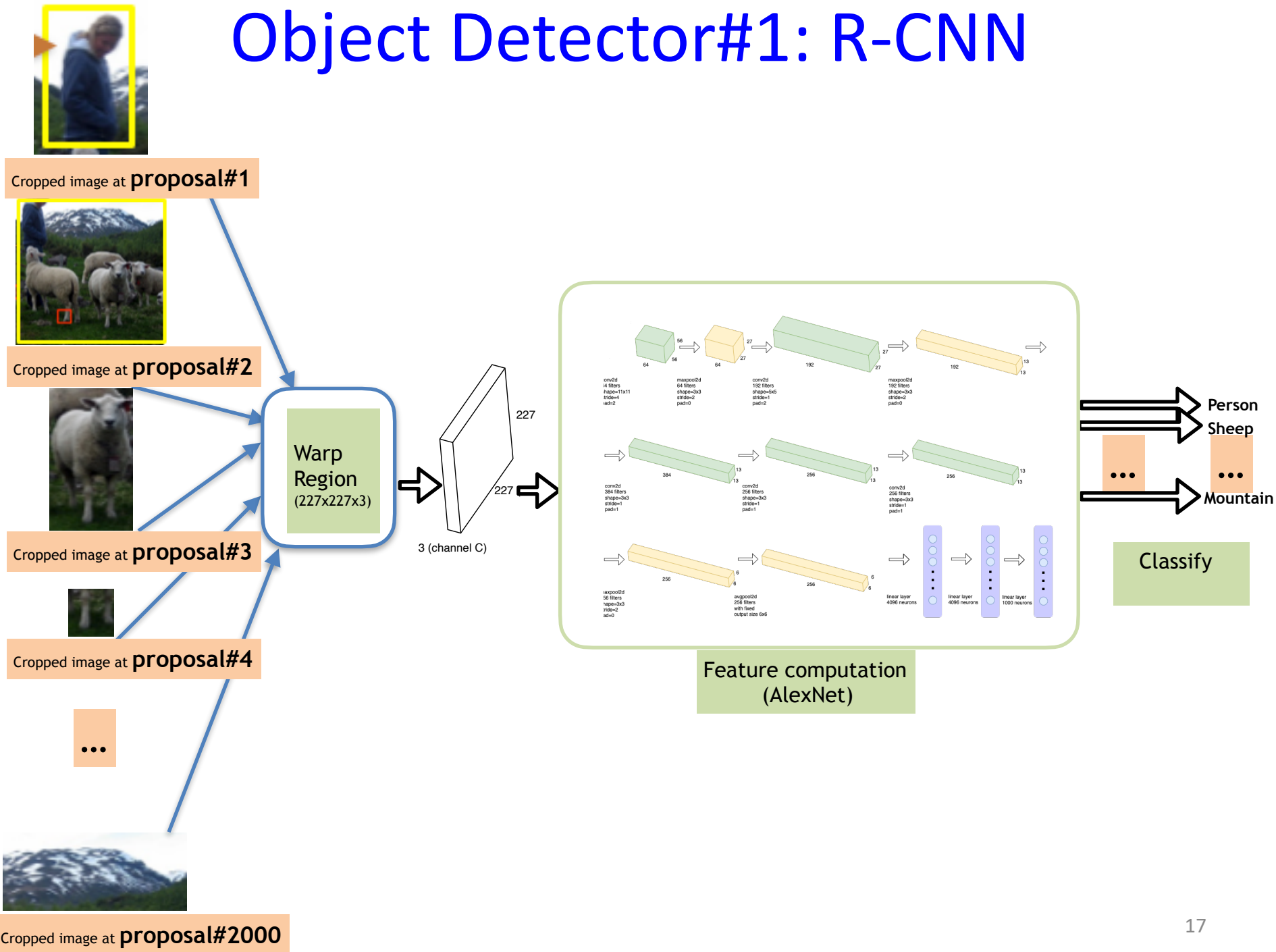
Object Detector#1: R-CNN

- Extract features for each proposal
 - 2000 proposals of various sizes
 - 4096 dimensional vector from AlexNet's last layer for each prop



Example: proposal#2 feature extraction

Object Detector#1: R-CNN

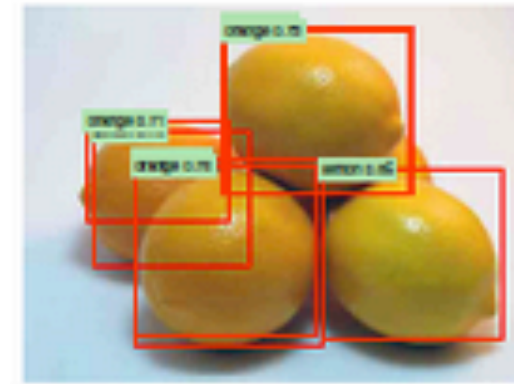


Object Detector#1: R-CNN

- **Caveat:** bottleneck is the feature extraction for each proposal
 - 2000 proposals of various sizes
 - 4096 dimensional vector from AlexNet's last layer for each proposal
 - we need to feed 2000 region proposals to the CNN (AlexNet) separately
 - very slow (low-throughput)
 - not end-to-end trainable

[Rich feature hierarchies for accurate object detection and semantic segmentation - Girshick et al. CVPR'14](#)

Object Detector#1: R-CNN



[Rich feature hierarchies for accurate object detection and semantic segmentation - Girshick et al. CVPR'14](#)

Deep learning-based object detections

- R-CNN
- **Fast R-CNN**
- Faster R-CNN

- Mask R-CNN

- YOLO
- SSD

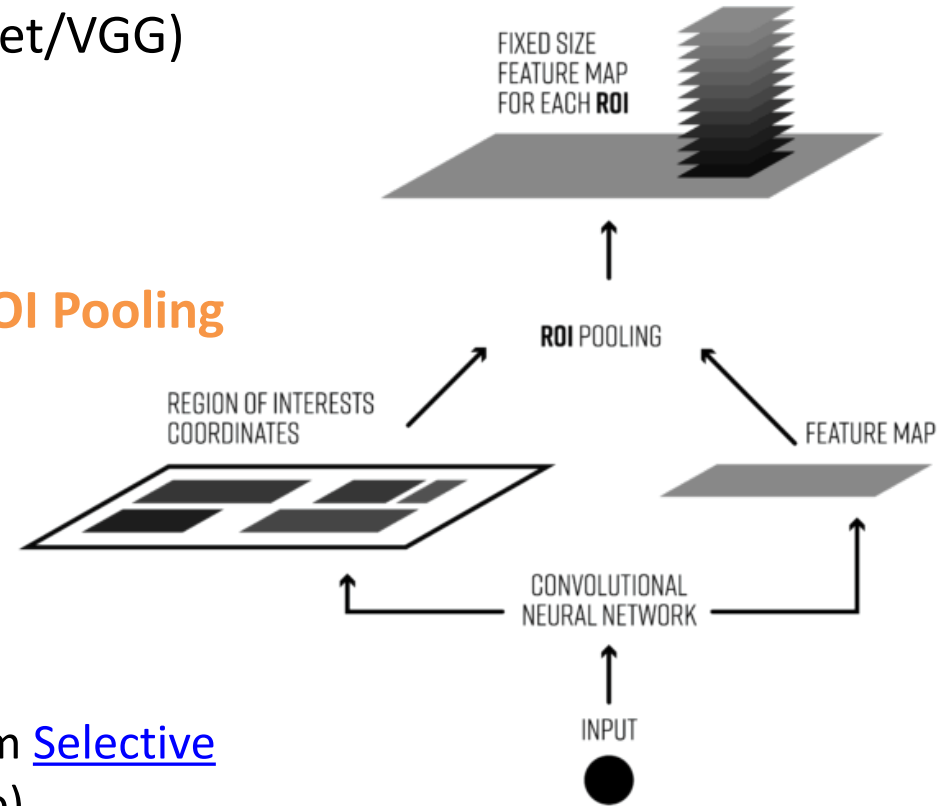
Why it is called “Fast” R-CNN?

- **Fast R-CNN** is faster than **R-CNN** because we don't need to feed 2000 region proposals to the convolutional neural network separately
- Instead, for each image, the convolution operation is done only once, and **only one** feature map is generated from it

[Fast R-CNN - Girshick et al. ICCV'15](#)

Object Detector#2: Fast R-CNN

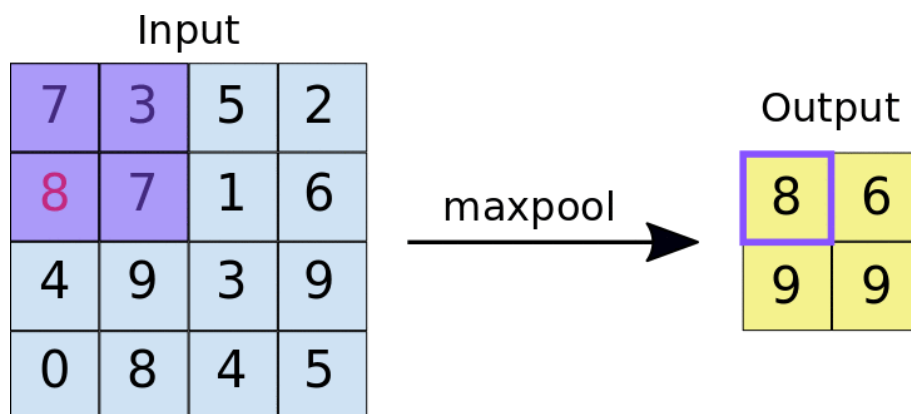
- Feed the image through a CNN (AlexNet/VGG)
- Extract fixed-size feature map using **ROI Pooling**
 - for different **Region Of Interests (ROI)**
- End-to-end trainable:
 - except proposals generation part from [Selective Search - IJCV'13](#) (R-CNN does that too)



[Fast R-CNN - Girshick et al. ICCV'15](#)

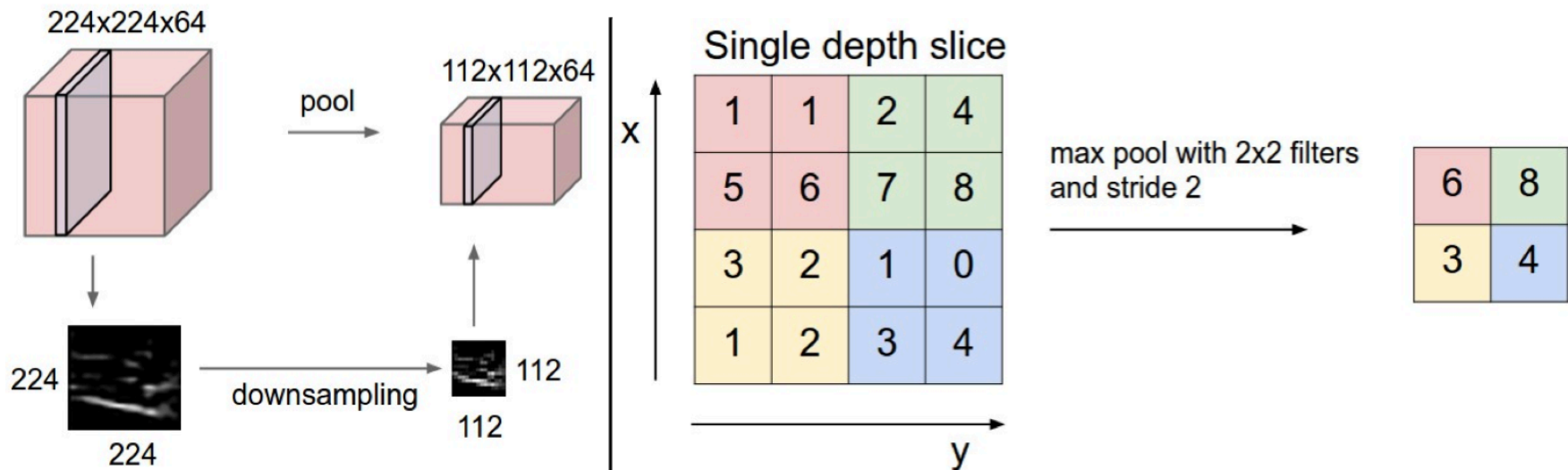
Pooling Operation (quick review)

- Image data can get computationally inefficient, really quickly. To avoid this, we often toss in a layer that helps us to **summarize** and **downsample** the data
- In classical CNN, we find another useful operation called **pooling operation**
- A common pooling operation is **max pooling**, and its goal is to locally summarize the convolution. It performs something like a convolution, but rather than taking the dot product, it **takes the maximum element in the filter area**



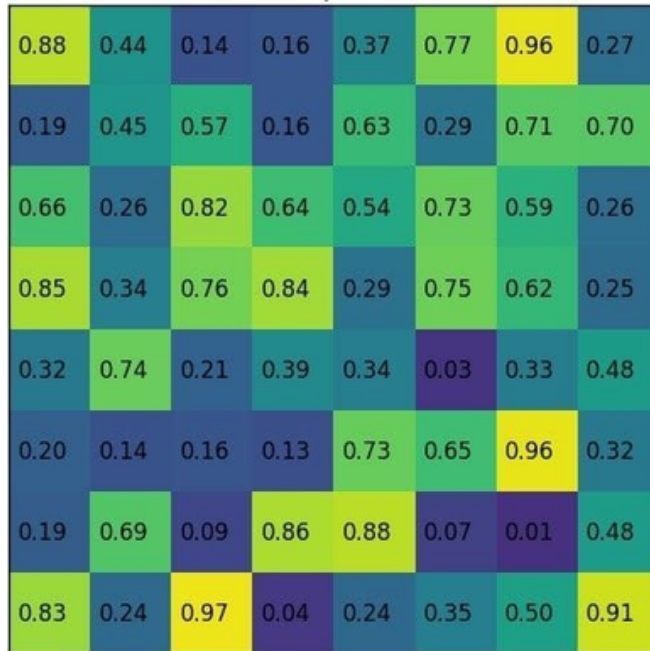
Pooling Operation (quick review)

- Pooling operation downsamples the volume spatially, **independently in each depth slice of the input volume**
- Besides **max pooling**, other pooling operations include: **sum pooling**, **average pooling**



ROI Pooling operation

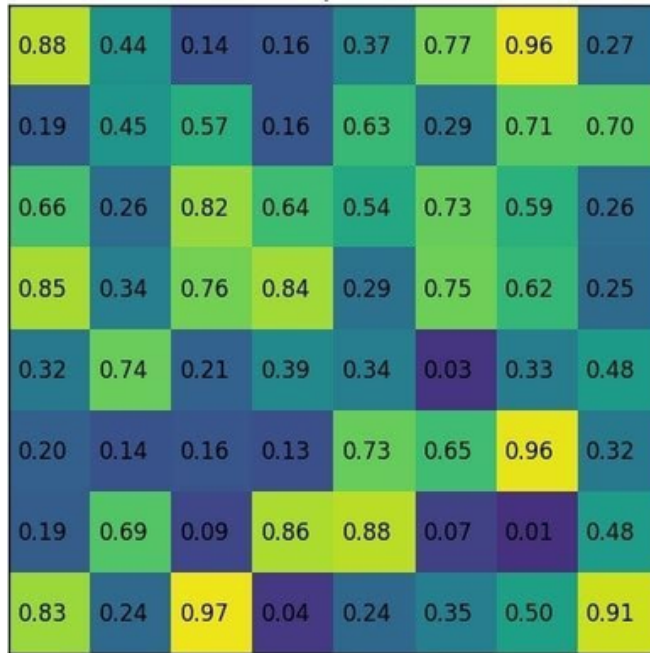
- Get a feature map



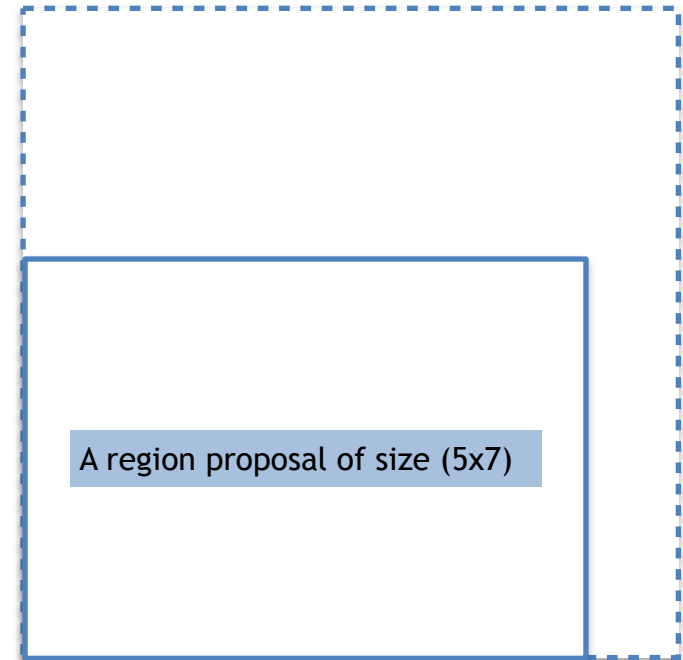
Feature map (1x8x8)

ROI Pooling operation

- Get a feature map
- Add a region proposal of size within the image window



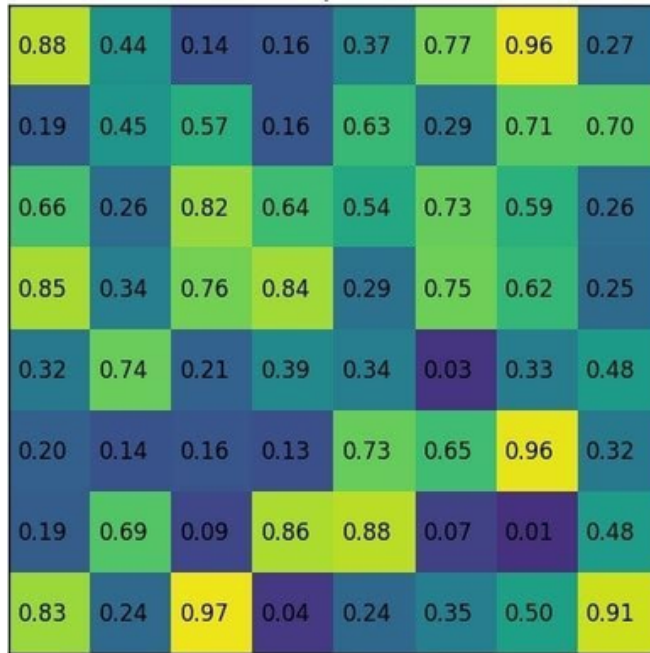
Feature map (1x8x8)



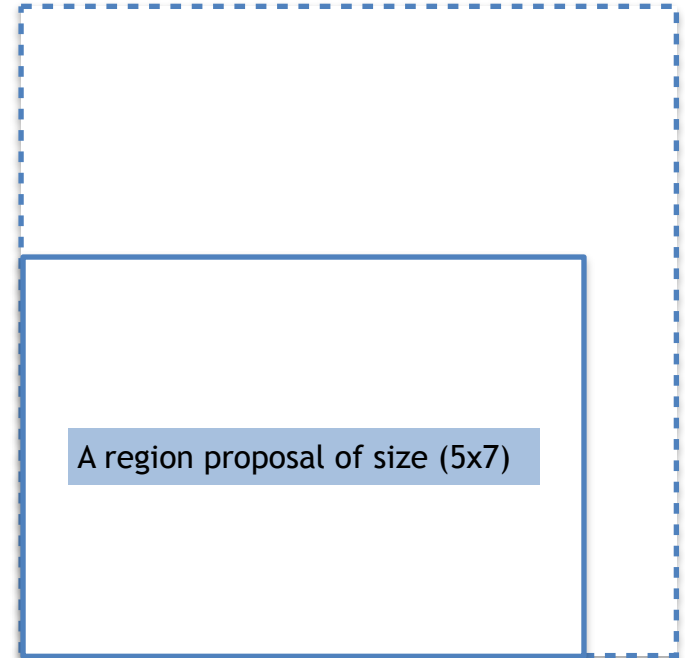
Region proposals

ROI Pooling operation

- Get a feature map
- Add a region proposal of size within the image window
- produce a fixed output feature map of size eg, (2x2), (6x6), etc



Feature map (1x8x8)

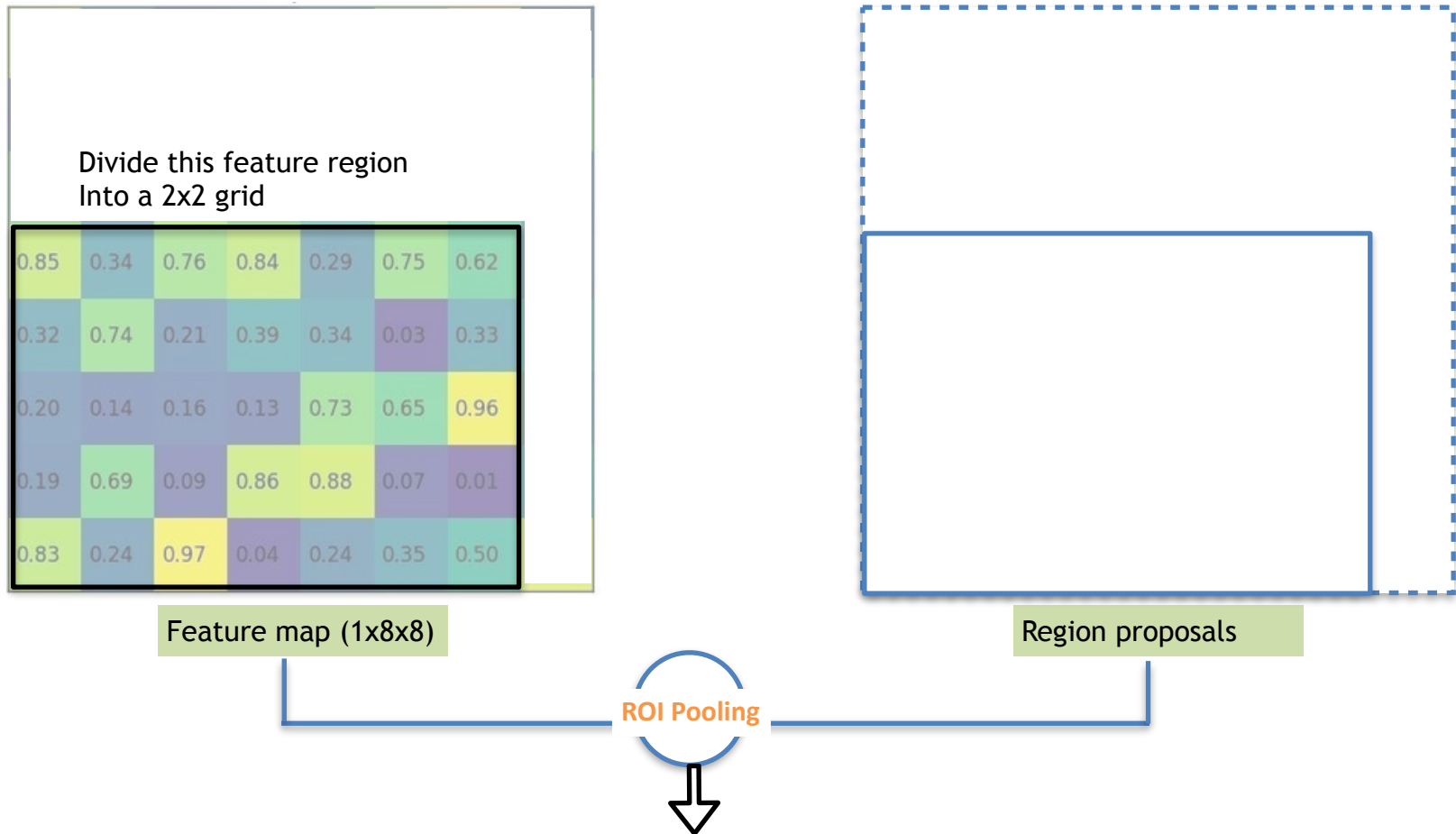


Region proposals

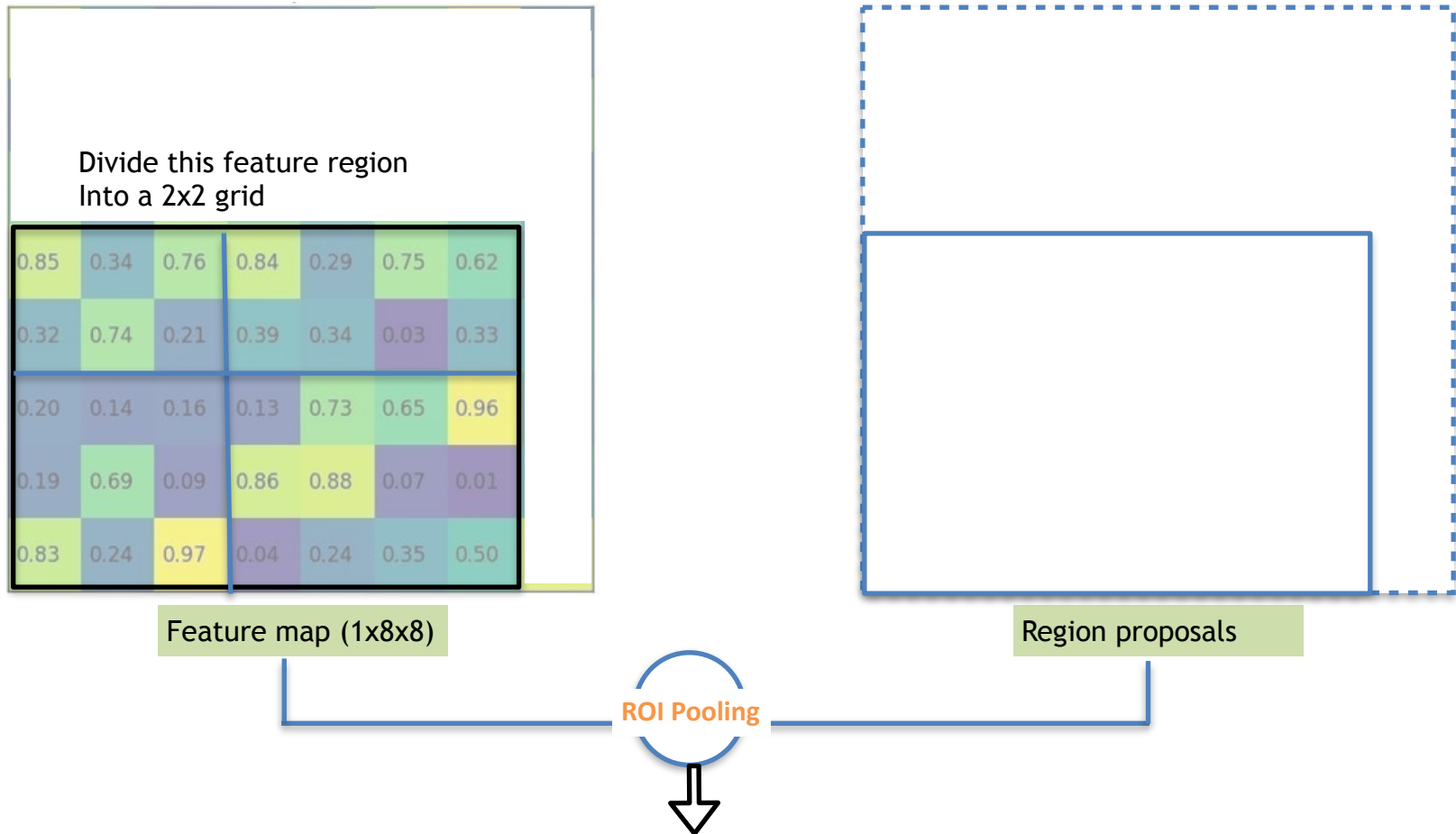
ROI Pooling

Produce a fixed size
output feature map

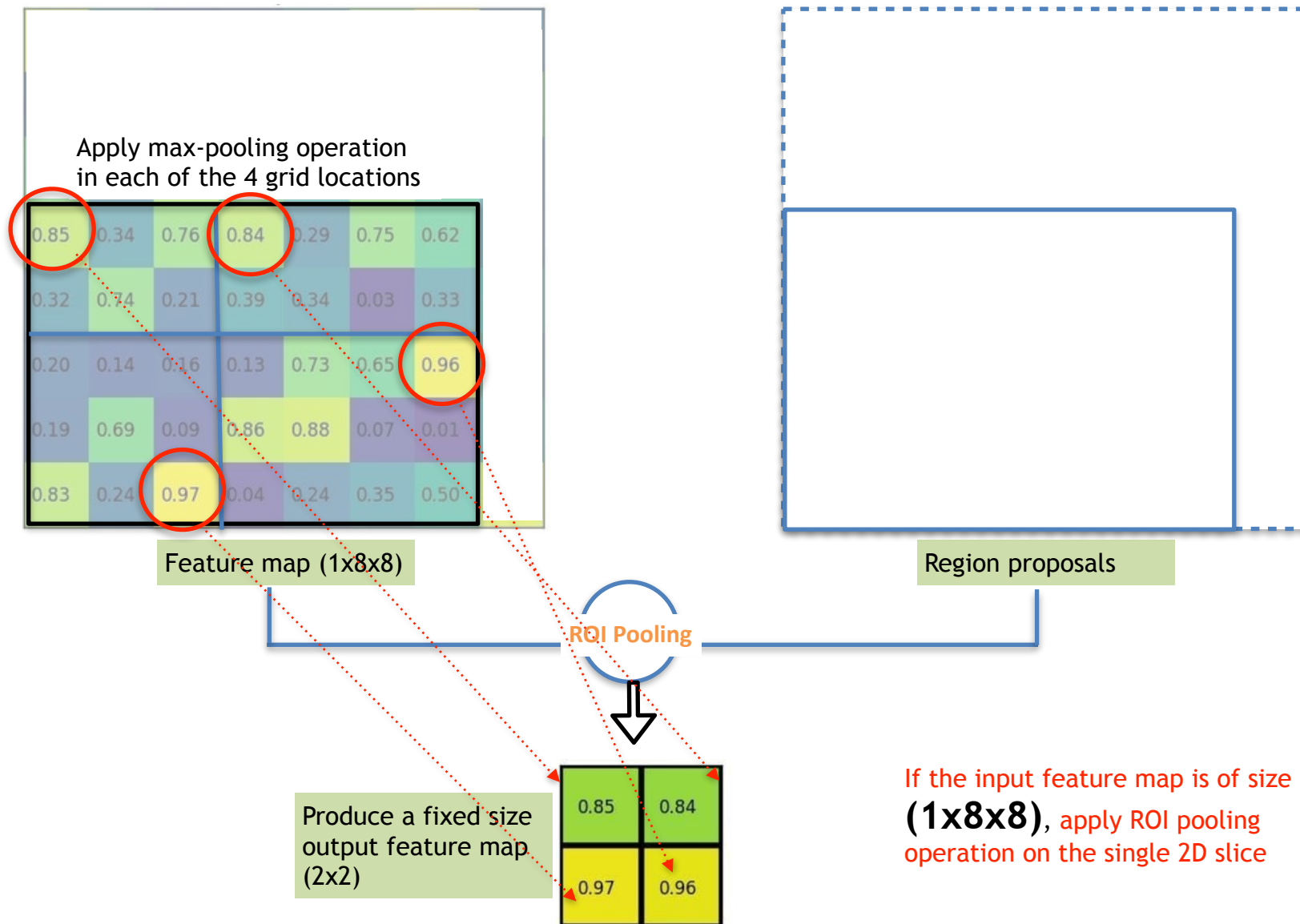
ROI Pooling operation

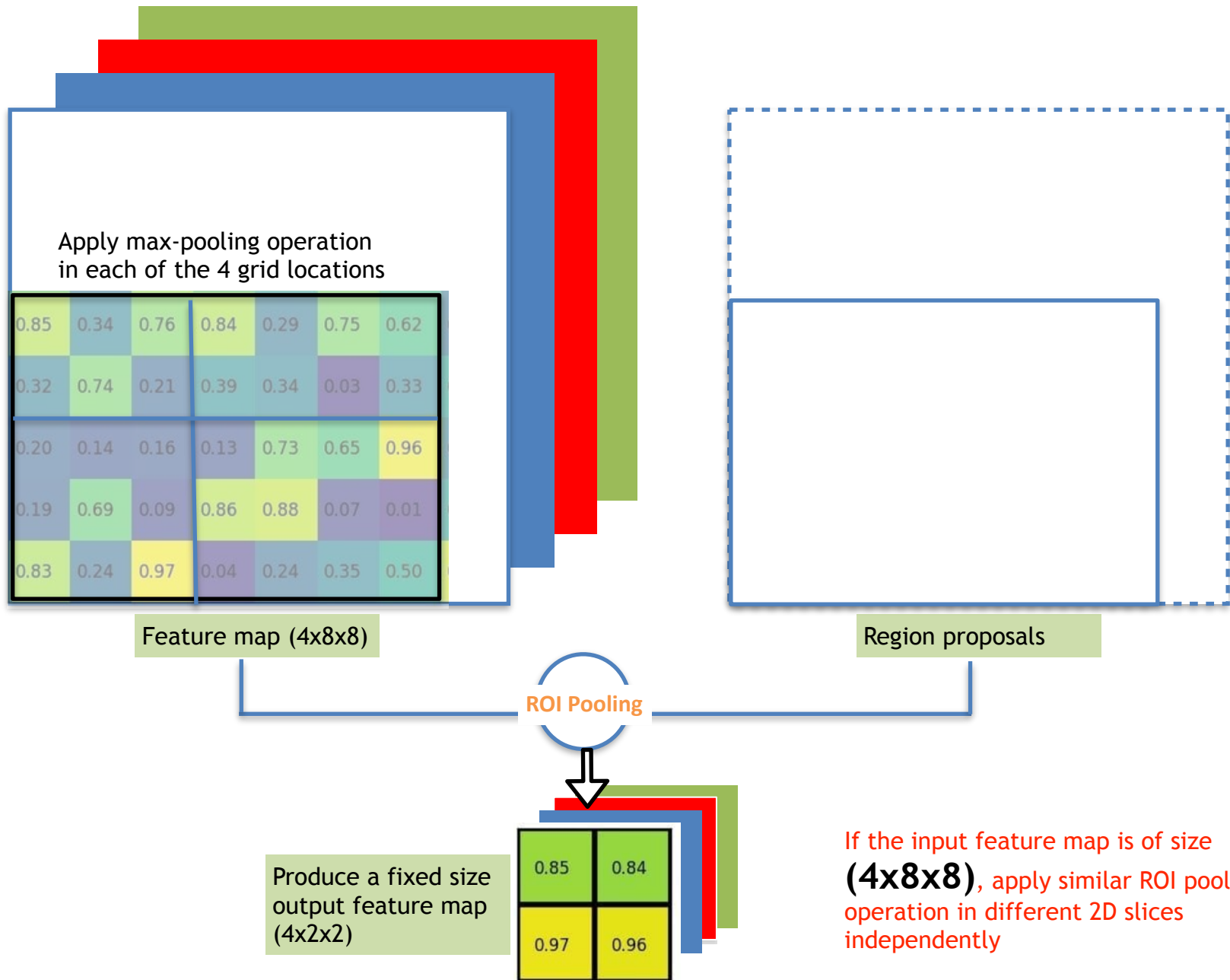


ROI Pooling operation



ROI Pooling operation





Deep learning-based object detections

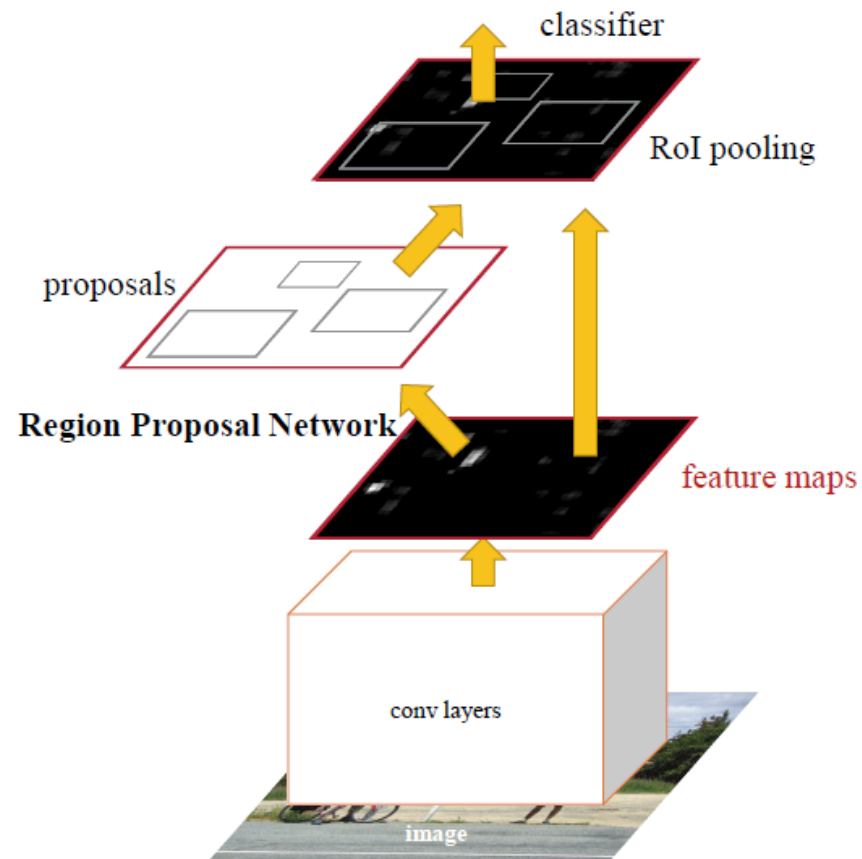
- R-CNN
- Fast R-CNN
- **Faster R-CNN**

- Mask R-CNN

- YOLO
- SSD

Object Detector#3: Faster R-CNN

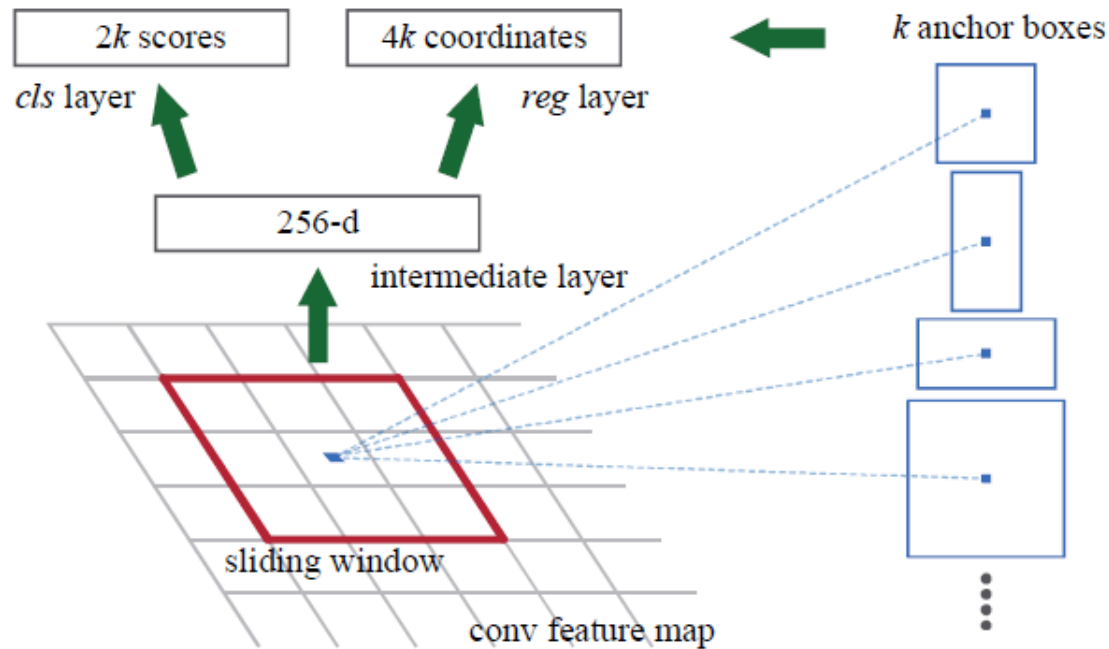
- Introduce a new network which will learn to generate good proposals. It is called **Region Proposal Network (RPN)** which will generate proposals.
- For each proposal, extract fixed-size feature map using **ROI Pooling**
- The network is end-to-end trainable:
 - Unlike R-CNN where different components are trained separately



[Faster R-CNN - Girshick et al. NIPS'15](#)

Object Detector#3: Faster R-CNN

- Region Proposal Network (RPN)

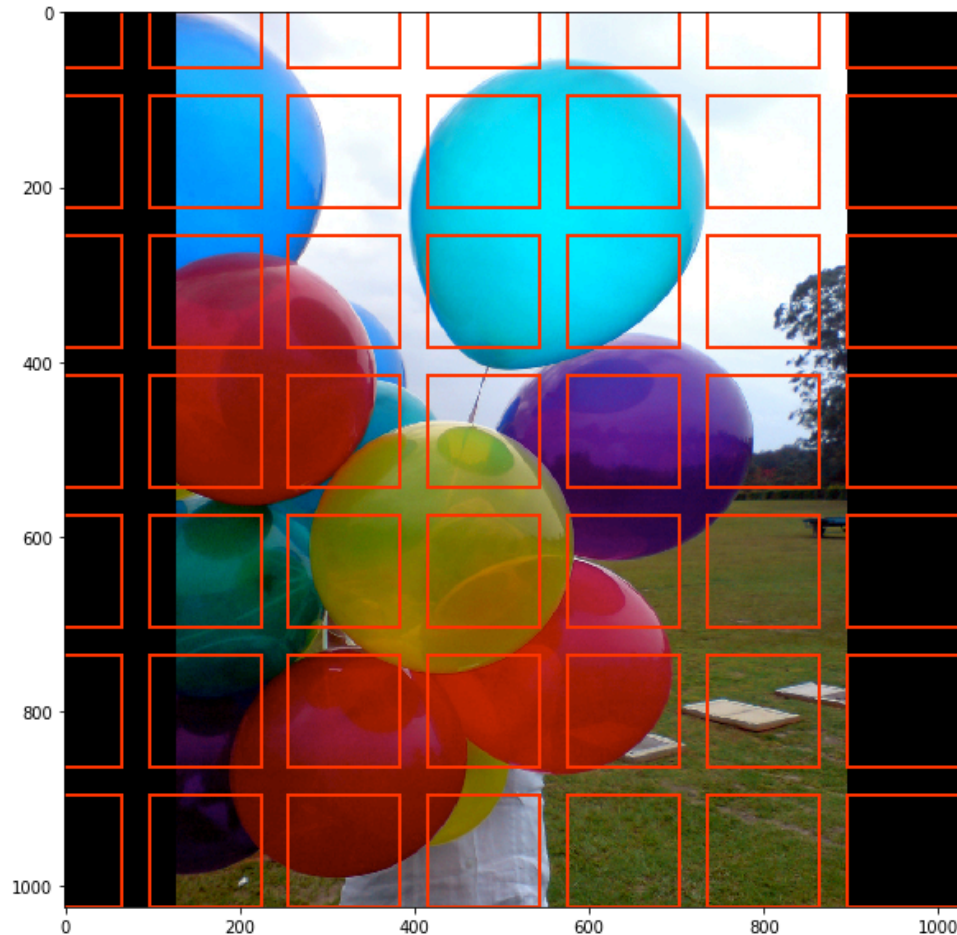


[Faster R-CNN - Girshick et al. NIPS'15](#)

Object Detector#3: Faster R-CNN

- **Region Proposal Network (RPN)**

[Faster R-CNN - Girshick et al. NIPS'15](#)



State-of-the-art object detection methods

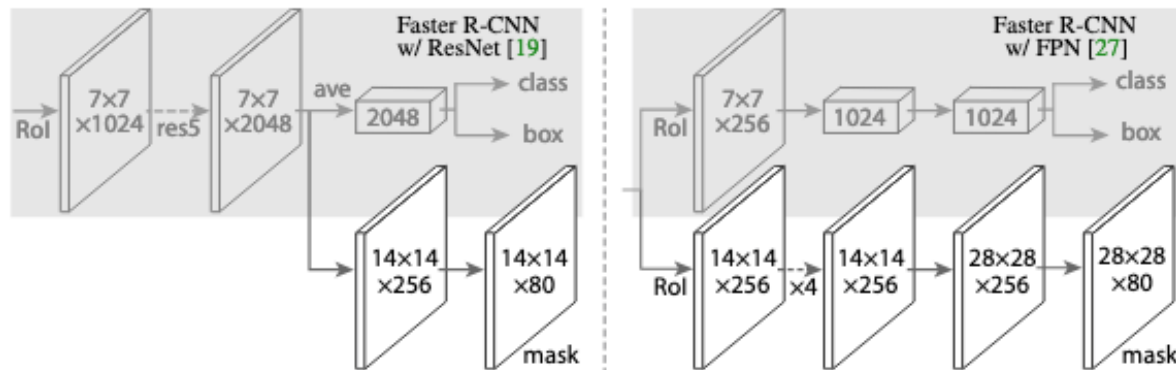
- R-CNN
- Fast R-CNN
- Faster R-CNN

- **Mask R-CNN**

- YOLO
- SSD

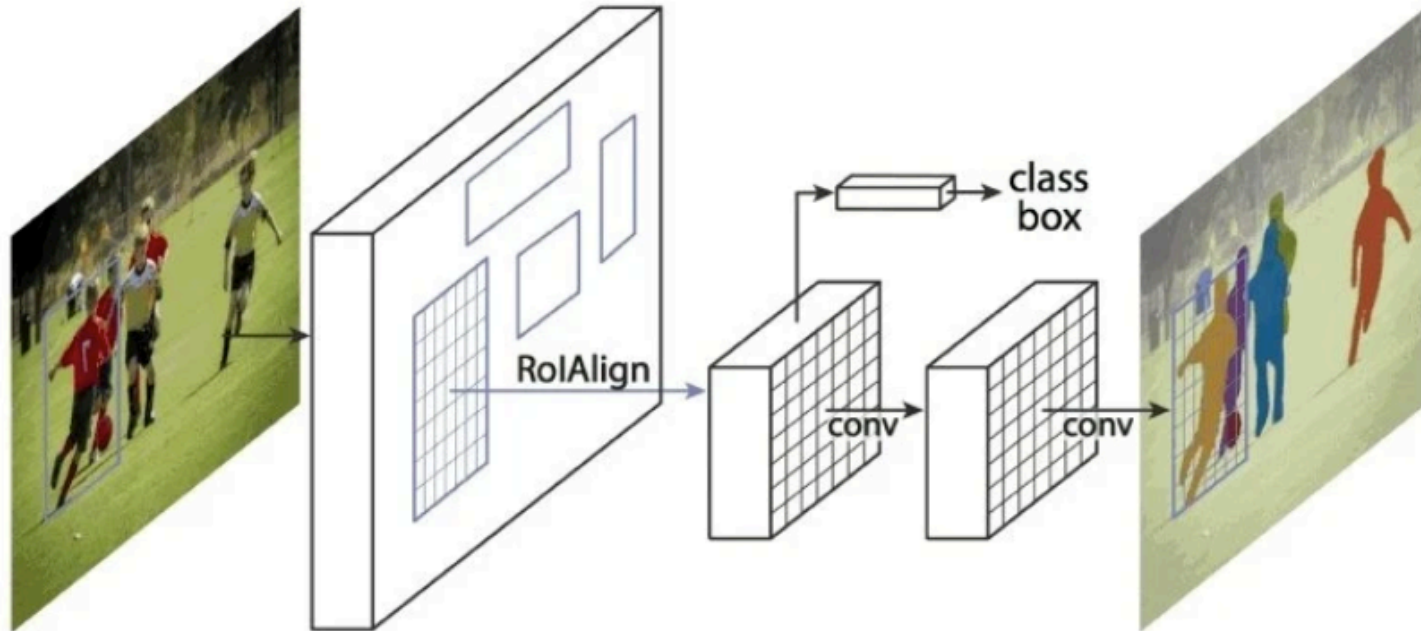
Object Detector#4: Mask R-CNN

- Built on top of Faster R-CNN
- Faster R-CNN has two outputs for each detected object
 - a class label and
 - a bounding-box parameters
- Mask R-CNN adds a third branch that outputs the object mask



[Mask R-CNN - Girshick et al. CVPR'17](#)

Object Detector#4: Mask R-CNN



[Mask R-CNN - Girshick et al. CVPR'17](#)

Object Detector#4: Mask R-CNN

Mask R-CNN - Girshick et al. CVPR'17

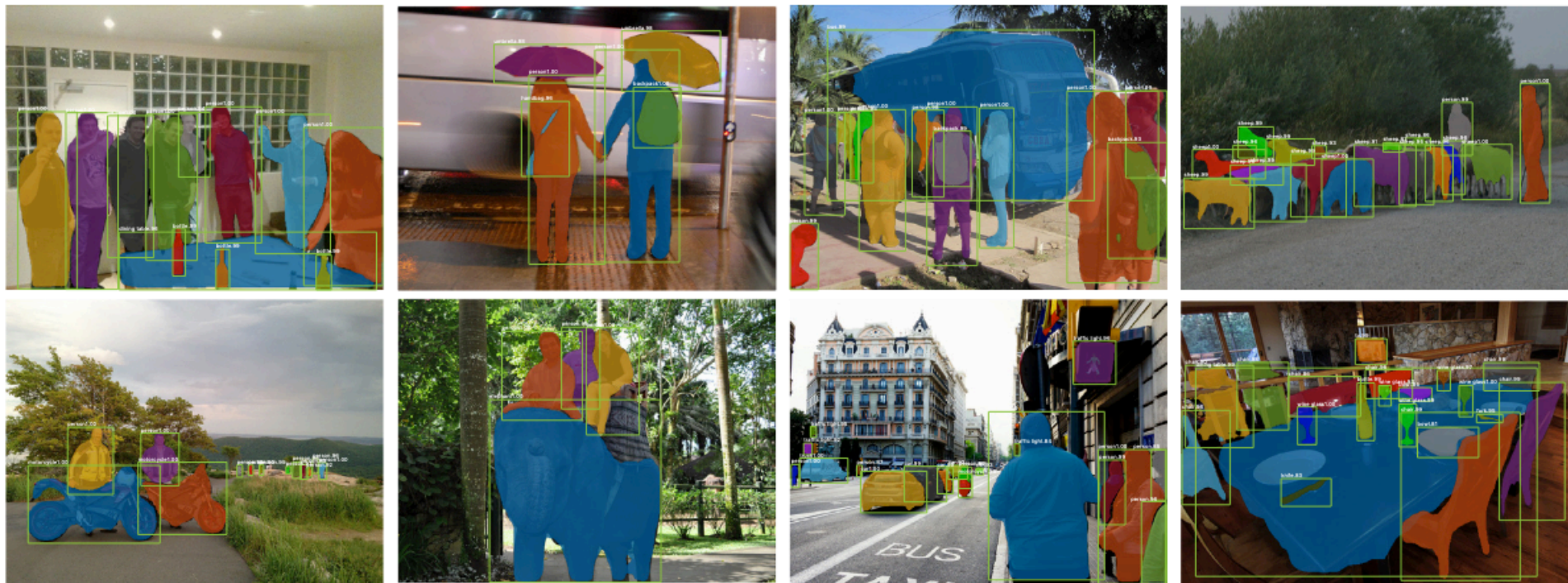


Figure 2. **Mask R-CNN** results on the COCO test set. These results are based on ResNet-101 [19], achieving a *mask AP* of 35.7 and running at 5 fps. Masks are shown in color, and bounding box, category, and confidences are also shown.

PyTorch's Detectors

- The summary of the performances for the models trained on the instances set of Microsoft COCO train2017 and evaluated on Microsoft COCO val2017:

Network	box AP
Faster R-CNN ResNet-50 FPN	37.0
Faster R-CNN MobileNetV3-Large FPN	32.8
Faster R-CNN MobileNetV3-Large 320 FPN	22.8
RetinaNet ResNet-50 FPN	36.4
Mask R-CNN ResNet-50 FPN	37.9

Coding Activity: Faster R-CNN

- Let's take a pre-trained Faster R-CNN model which was trained on Microsoft COCO (Common Object in COntext) Dataset with 91 categories
 - First, let's run the inference on an arbitrary dataset eg, Pedestrian Dataset
 - We will fine-tune the pre-trained model using the Pedestrian Dataset next week

