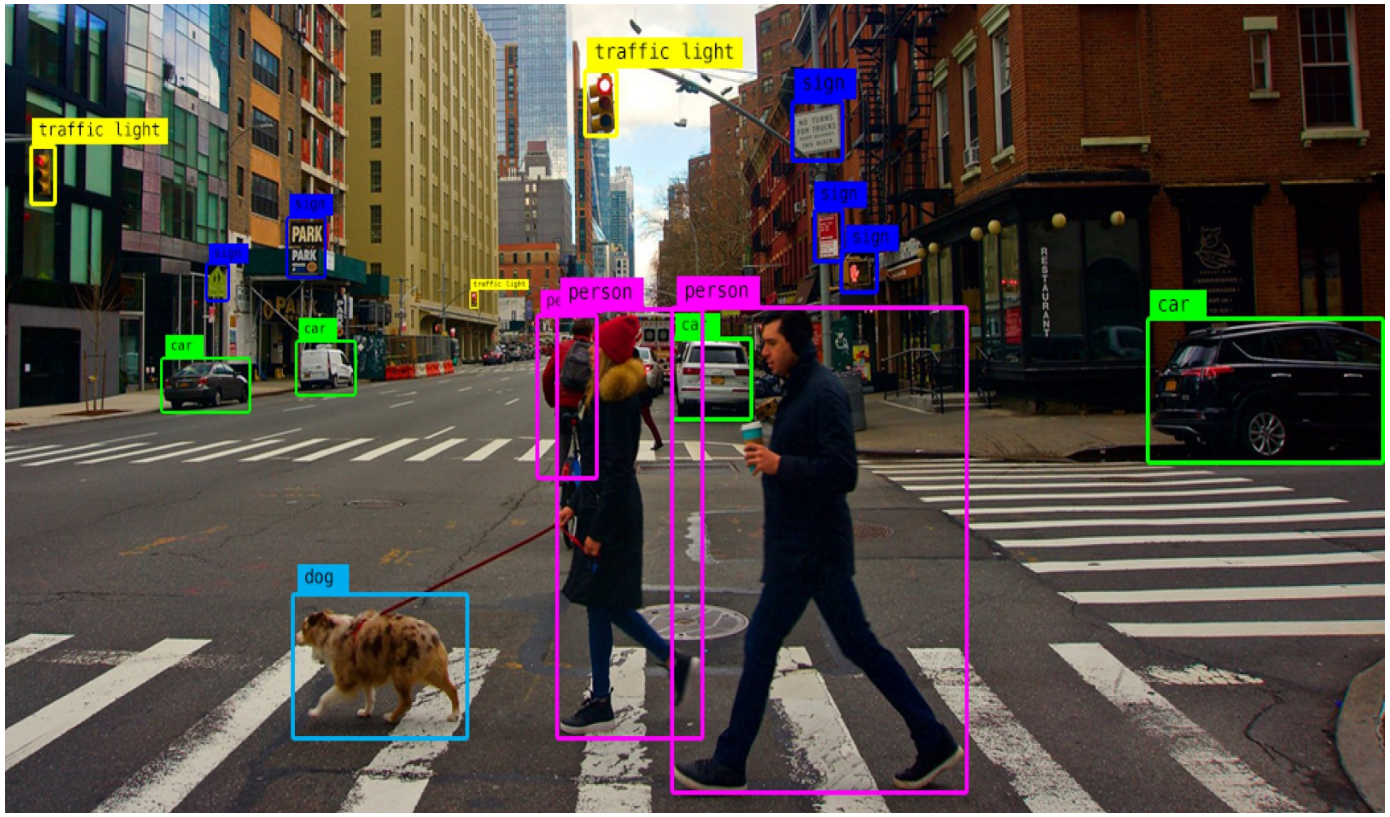
COMP9517

Computer Vision

2024 Term 3 Week 8

Dr Sonit Singh

UNSW SYDNEY



Deep Learning II

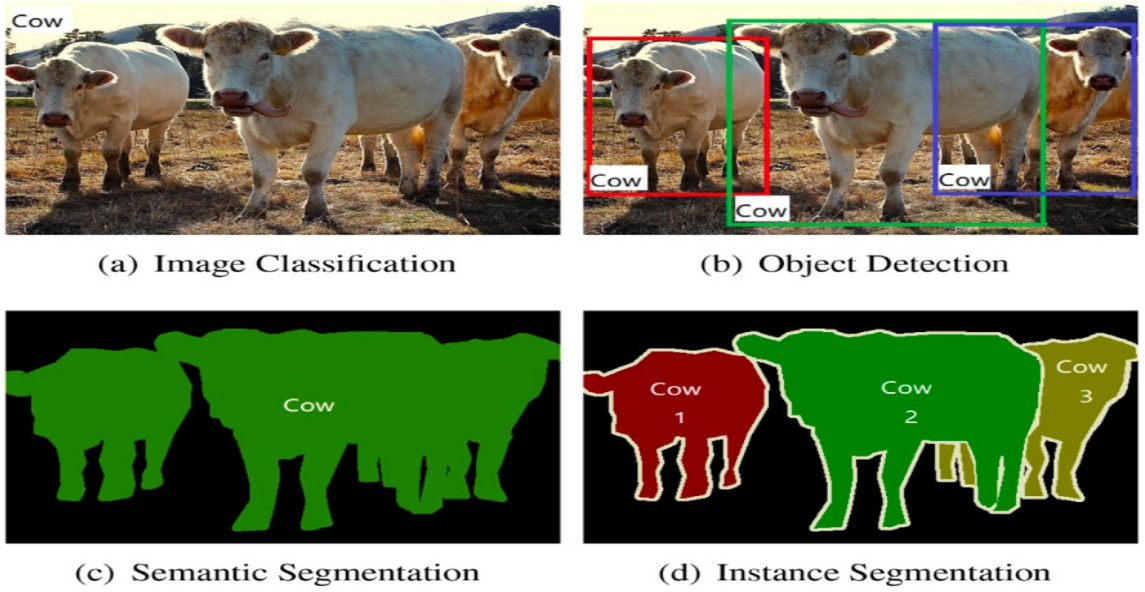
Object Detection using CNNs

Outline

- Computer Vision tasks
- Object Detection dataset example
- Proposal based algorithms
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
- Proposal free algorithms
  - Single Shot Detector (SSD)
  - You Only Look Once (YOLO)
  - RetinaNet

Vision tasks

- Image classification: Assigning a label or class to an image
- Object detection: Locate the presence of objects with a bounding box and class of the located objects in an image
- Semantic segmentation: Label every pixel (pixel-wise classification)
- Instance segmentation: Differentiate instances



(a) Image Classification (b) Object Detection

(c) Semantic Segmentation (d) Instance Segmentation

Image Classification

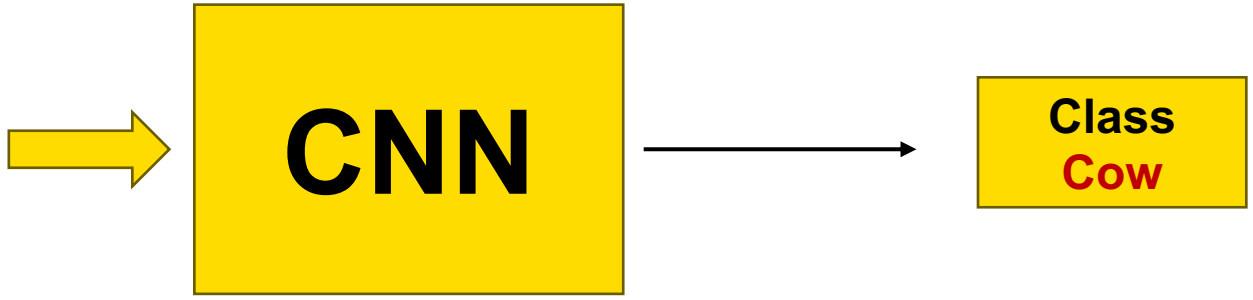


Image Credit: Creative Commons Licenses

Object Detection

- Determine “what” and “where”
  - “what”: Classification
  - “where”: Localization

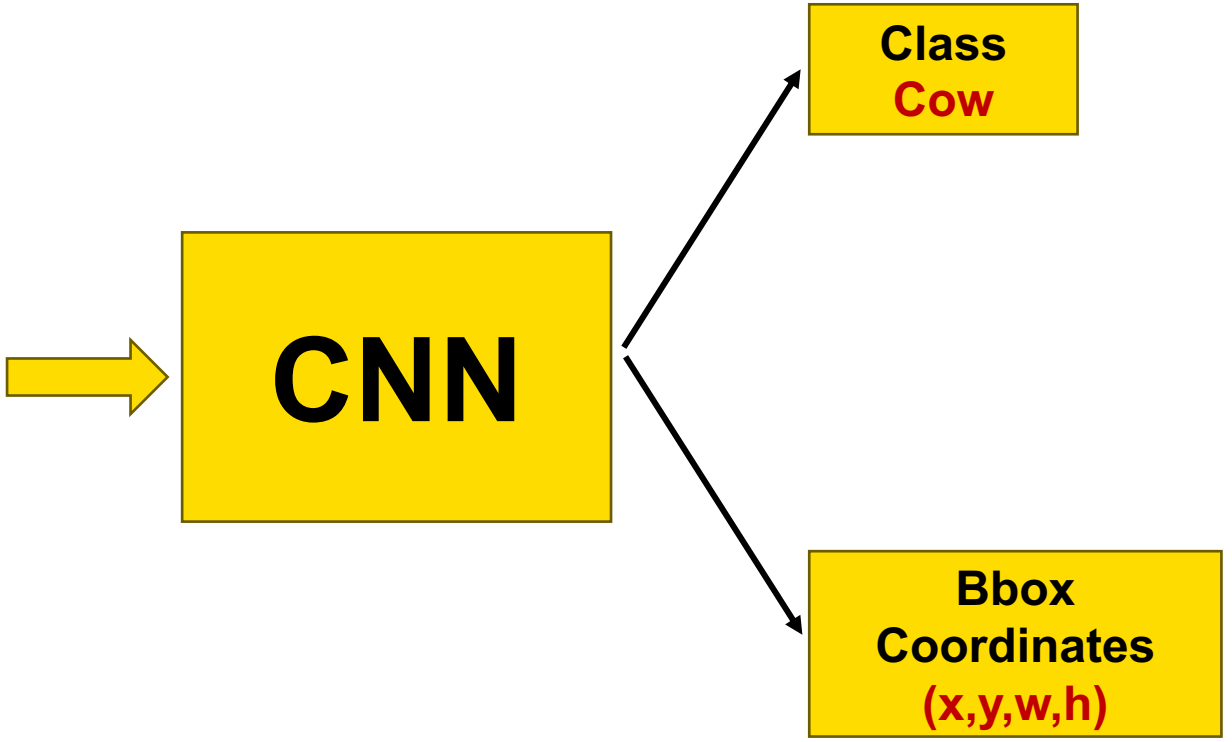
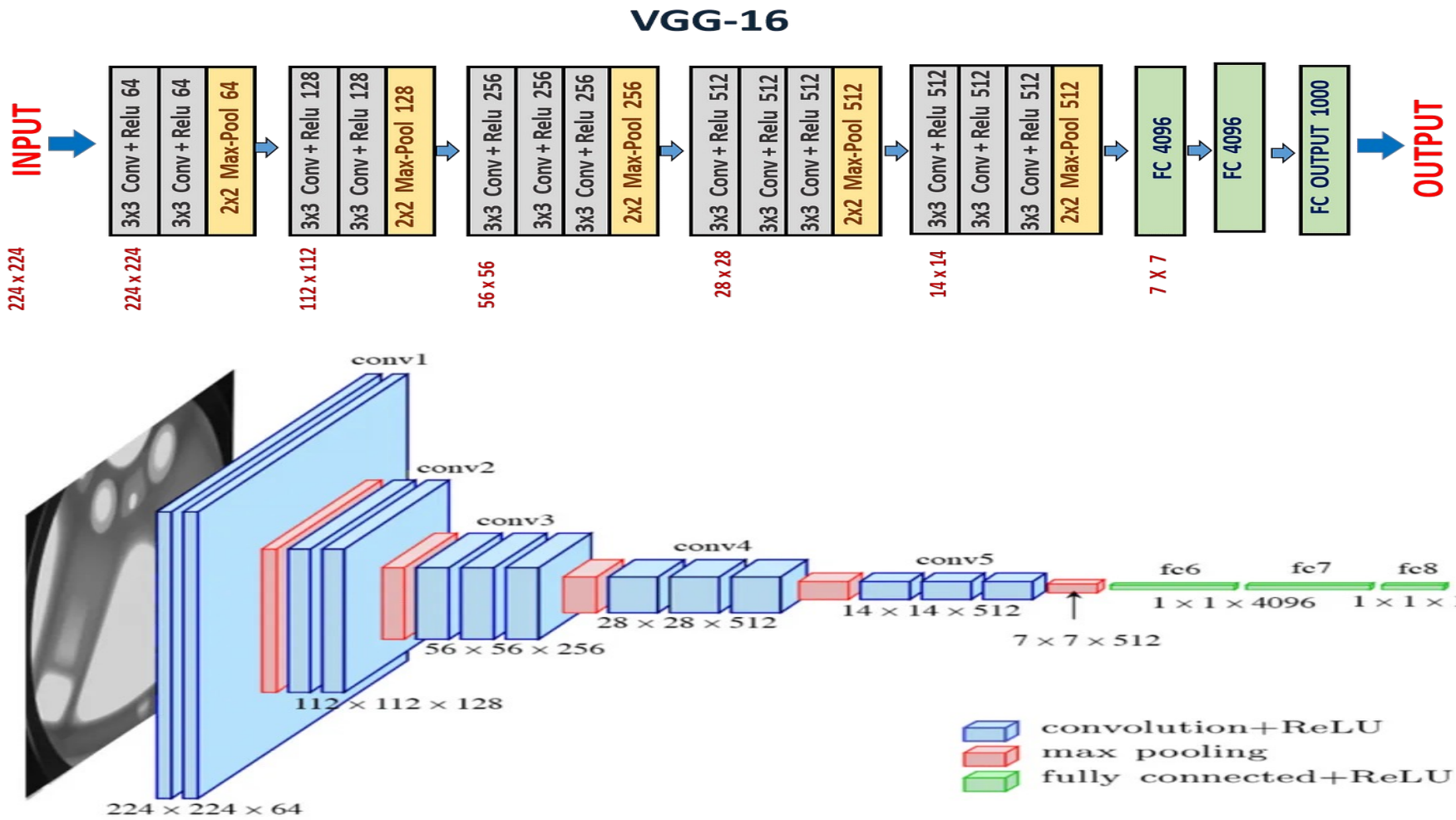


Image Credit: Creative Commons Licenses

How to use CNNs

- Use Fully-connected layers for mapping to Class and Bbox Coordinates



Object Detection

- Use Fully-connected layers for mapping to Class and Bbox Coordinates

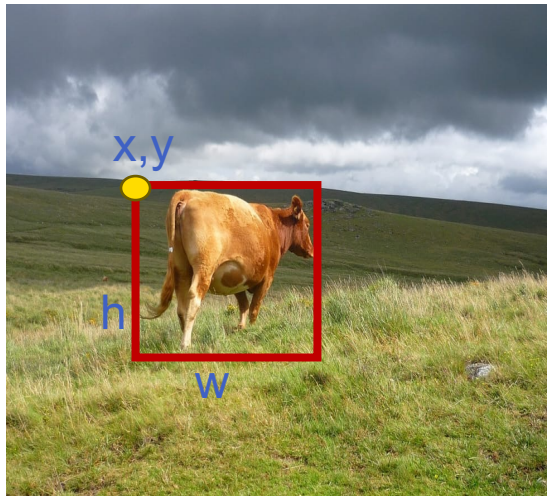
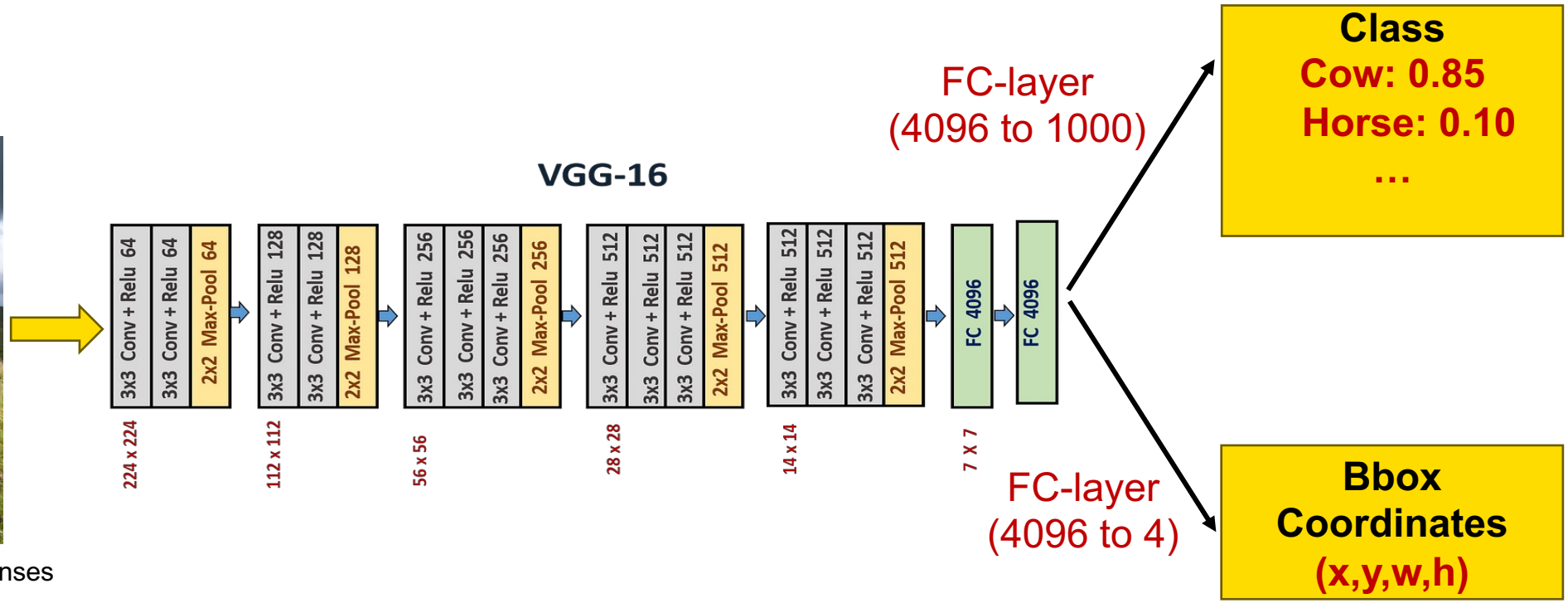
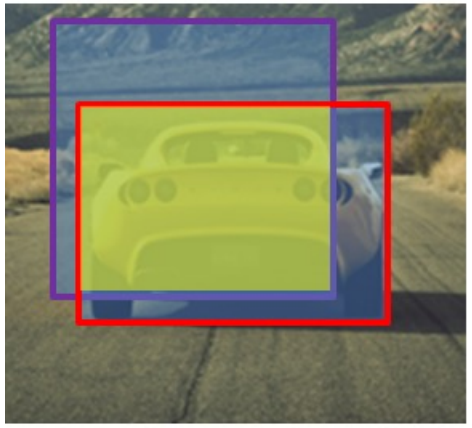


Image Credit: Creative Commons Licenses



Intersection over Union (IoU)

- To maximize overlap between the predicted bounding box and the actual bounding box, maximize Intersection over Union (IoU)



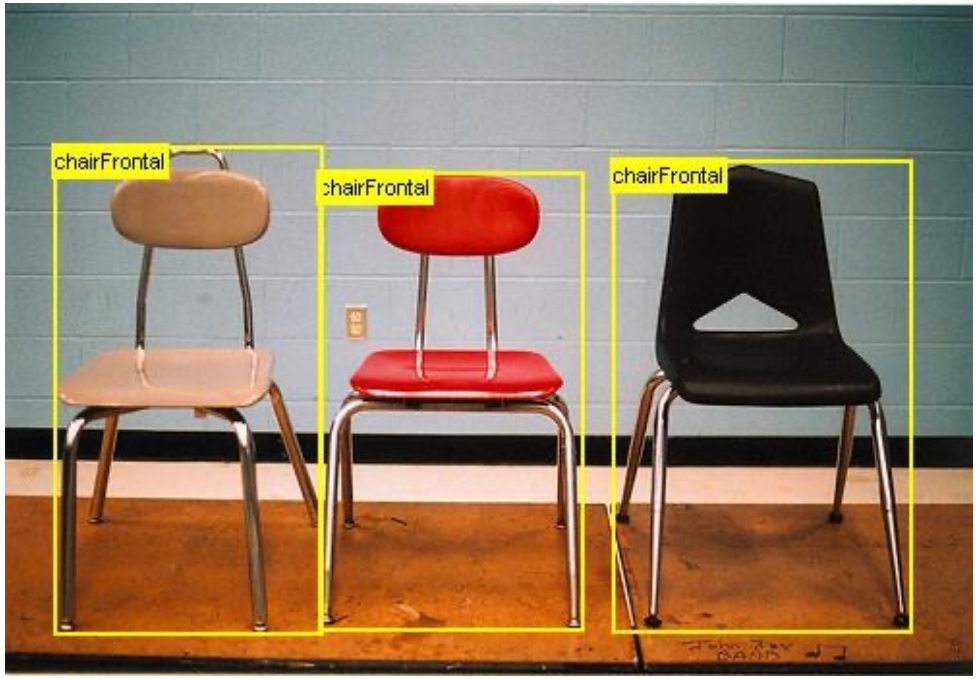
Intersection over union (IoU)

$$= \frac{\text{size of } \square}{\text{size of } \square}$$

"Correct" if IoU $\geq 0{,}5$

➢ Example: PASCAL VOC challenge dataset

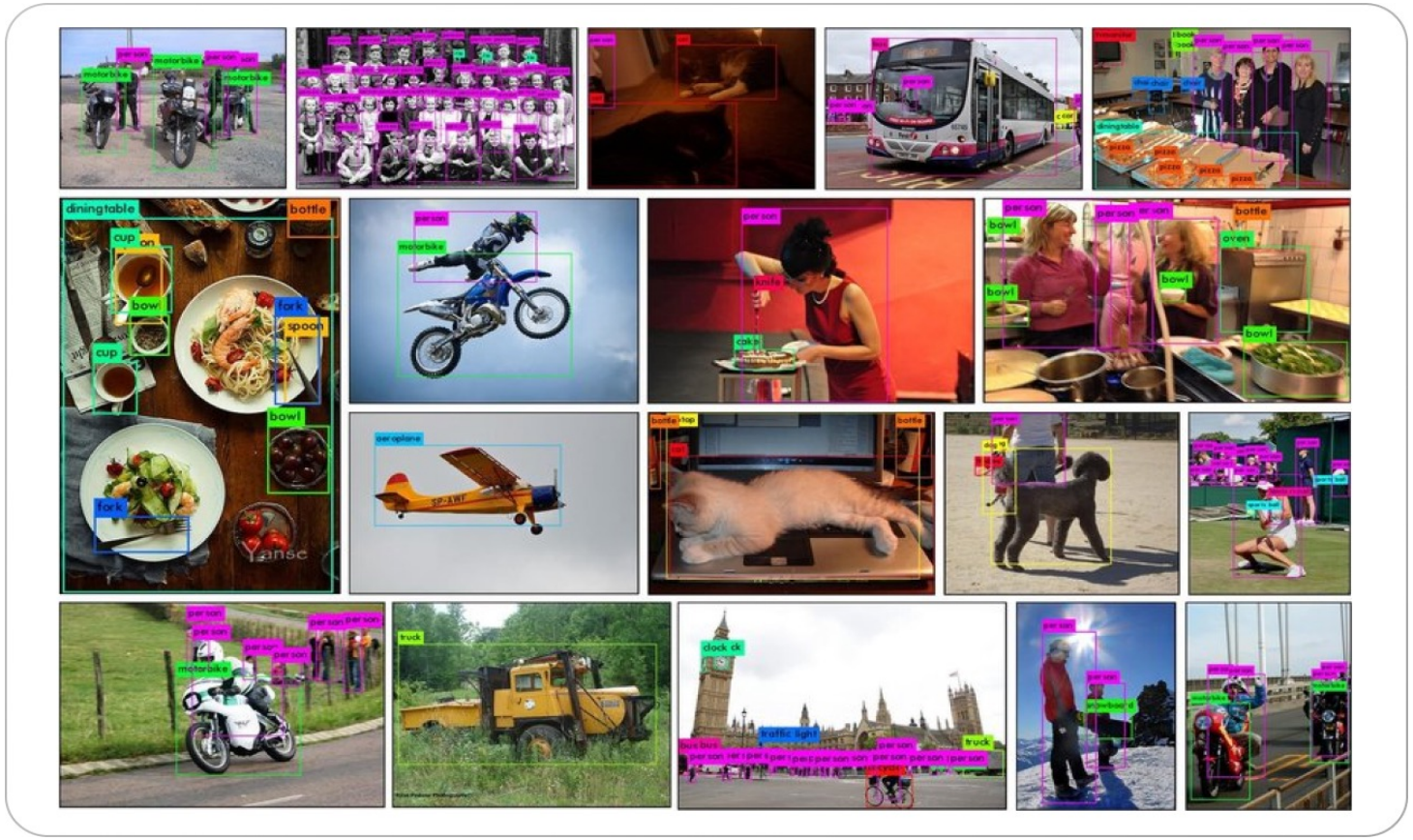
- Goal: To recognize objects from a number of visual object classes in realistic scenes (i.e. not pre-segmented objects).
- It is fundamentally a supervised learning problem in that a training set of labelled images is provided.
- The twenty object classes that have been selected are:
  - *Person:* person
  - *Animal:* bird, cat, cow, dog, horse, sheep
  - *Vehicle:* aeroplane, bicycle, boat, bus, car, motorbike, train
  - *Indoor:* bottle, chair, dining table, potted plant, sofa, tv/monitor
- The data provided consists of a set of images; each image has an annotation file giving a bounding box and object class label for each object present in the image.



Source: PASCAL VOC 2012 Challenge Dataset http://host.robots.ox.ac.uk/pascal/VOC/voc2012/

➢ Example: Microsoft COCO dataset

➢ Large-scale object detection, segmentation, and captioning dataset.



COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features:

- Object segmentation
- Recognition in context
- Superpixel stuff segmentation
- 330K images (>200K labeled)
- 1.5 million object instances
- 80 object categories
- 91 stuff categories
- 5 captions per image
- 250,000 people with keypoints

```
annotation{
   "id"              : int,
   "image_id"        : int,
   "category_id"     : int,
   "segmentation"    : RLE or [polygon],
   "area"            : float,
   "bbox"            : [x,y,width,height],
   "iscrowd"         : 0 or 1,
}

categories[{
   "id"              : int,
   "name"            : str,
   "supercategory"   : str,
}]
```

Source: Microsoft COCO https://cocodataset.org/#home

Object Detection

- How to train?
  - Use Softmax Loss for Classification
  - Use Regression loss for Localization

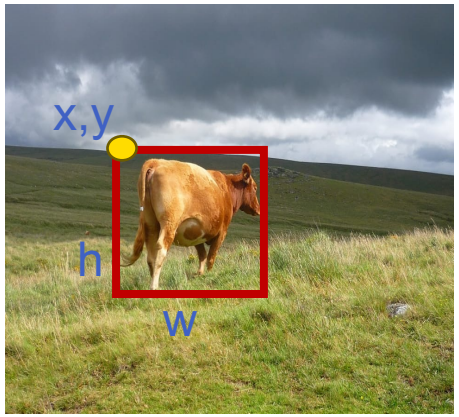
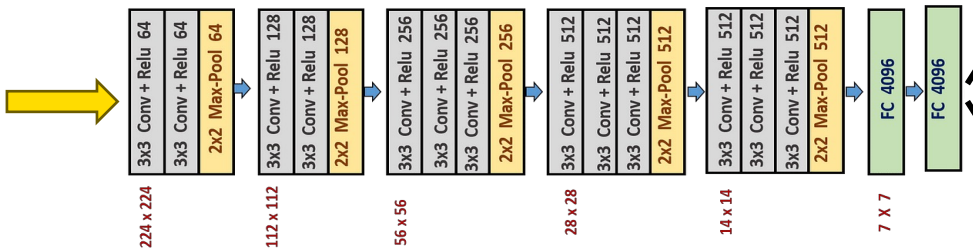
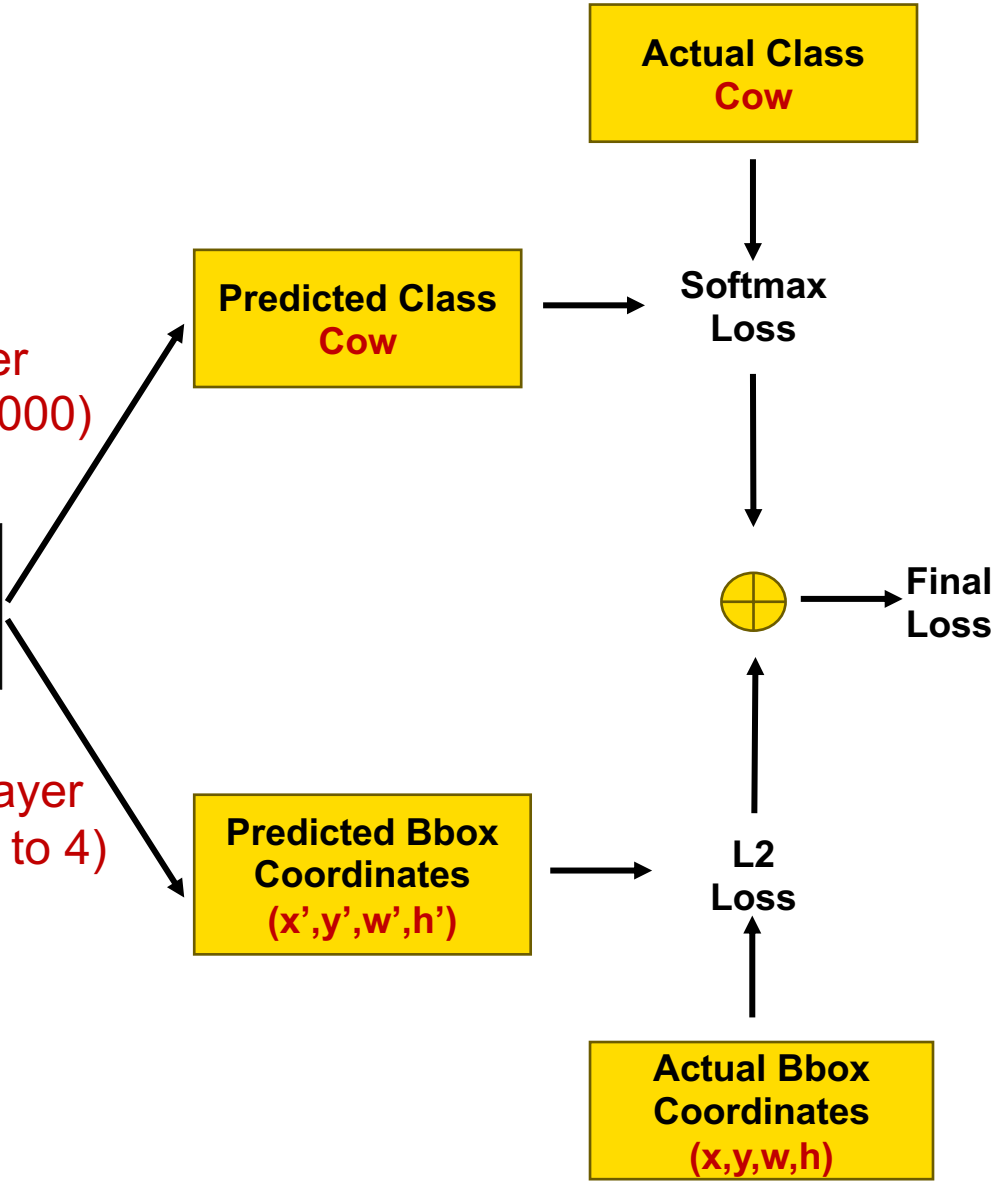


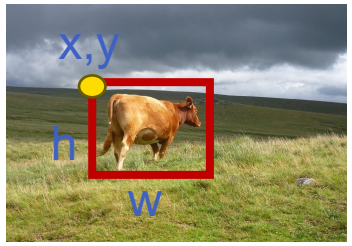
Image Credit: Creative Commons Licenses





Object Detection

- How to detect multiple objects?



CNN → Cow: (x, y, w, h)

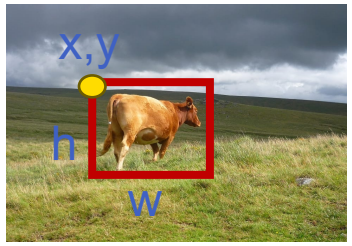
CNN → Cow: (x_1, y_1, w_1, h_1)
Bull: (x_2, y_2, w_2, h_2)

CNN → Zebra: (x_1, y_1, w_1, h_1)
Zebra: (x_2, y_2, w_2, h_2)
Zebra: (x_3, y_3, w_3, h_3)
...

Image Credit: Creative Commons Licenses

Object Detection

- How to detect multiple objects? Challenge of mapping to different number of outputs



CNN

Cow: (x, y, w, h)

4 numbers

CNN

Cow: (x_1, y_1, w_1, h_1)
Bull: (x_2, y_2, w_2, h_2)

8 numbers

CNN

Dog: (x_1, y_1, w_1, h_1)
Rabbit: (x_2, y_2, w_2, h_2)
Sheep: (x_3, y_3, w_3, h_3)
Chicken: (x_4, y_4, w_4, h_4)
...

Many

Image Credit: Creative Commons Licenses

Object Detection

- How to detect multiple objects?
- Solution: Apply a CNN to various crops of the image, CNN classifies each crop as object or background

Dog (x_1, y_1, w_1, h_1): No
Rabbit (x_2, y_2, w_2, h_2): No
Sheep (x_3, y_3, w_3, h_3): No
Background (x_4, y_4, w_4, h_4): Yes
...

Image Credit: Creative Commons Licenses

Object Detection

- How to detect multiple objects?
- Solution: Apply a CNN to various crops of the image, CNN classifies each crop as object or background

Dog (x_1, y_1, w_1, h_1): Yes
Rabbit (x_2, y_2, w_2, h_2): No
Sheep (x_3, y_3, w_3, h_3): No
Background (x_4, y_4, w_4, h_4): No
...

Image Credit: Creative Commons Licenses

Object Detection

- How to detect multiple objects?
- Solution: Apply a CNN to various crops of the image, CNN classifies each crop as object or background

Dog (x_1, y_1, w_1, h_1): No
Rabbit (x_2, y_2, w_2, h_2): Yes
Sheep (x_3, y_3, w_3, h_3): No
Background (x_4, y_4, w_4, h_4): No
...

Image Credit: Creative Commons Licenses

Object Detection

- How to detect multiple objects?
- Solution: Apply a CNN to various crops of the image, CNN classifies each crop as object or background

Dog (x_1, y_1, w_1, h_1): No
Rabbit (x_2, y_2, w_2, h_2): No
Sheep (x_3, y_3, w_3, h_3): Yes
Background (x_4, y_4, w_4, h_4): No
…

Image Credit: Creative Commons Licenses

Object Detection

- How to detect multiple objects?
- Solution: Apply a CNN to various crops of the image, CNN classifies each crop as object or background
- Limitation: Computationally very expensive
  - Need to apply CNN on many crops (think in terms of different locations, aspect ratio)

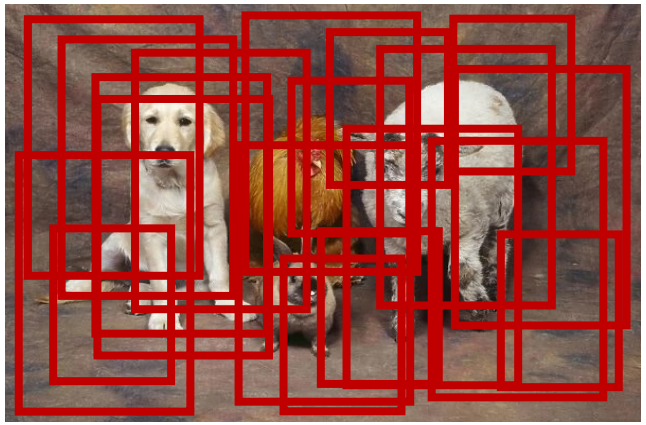
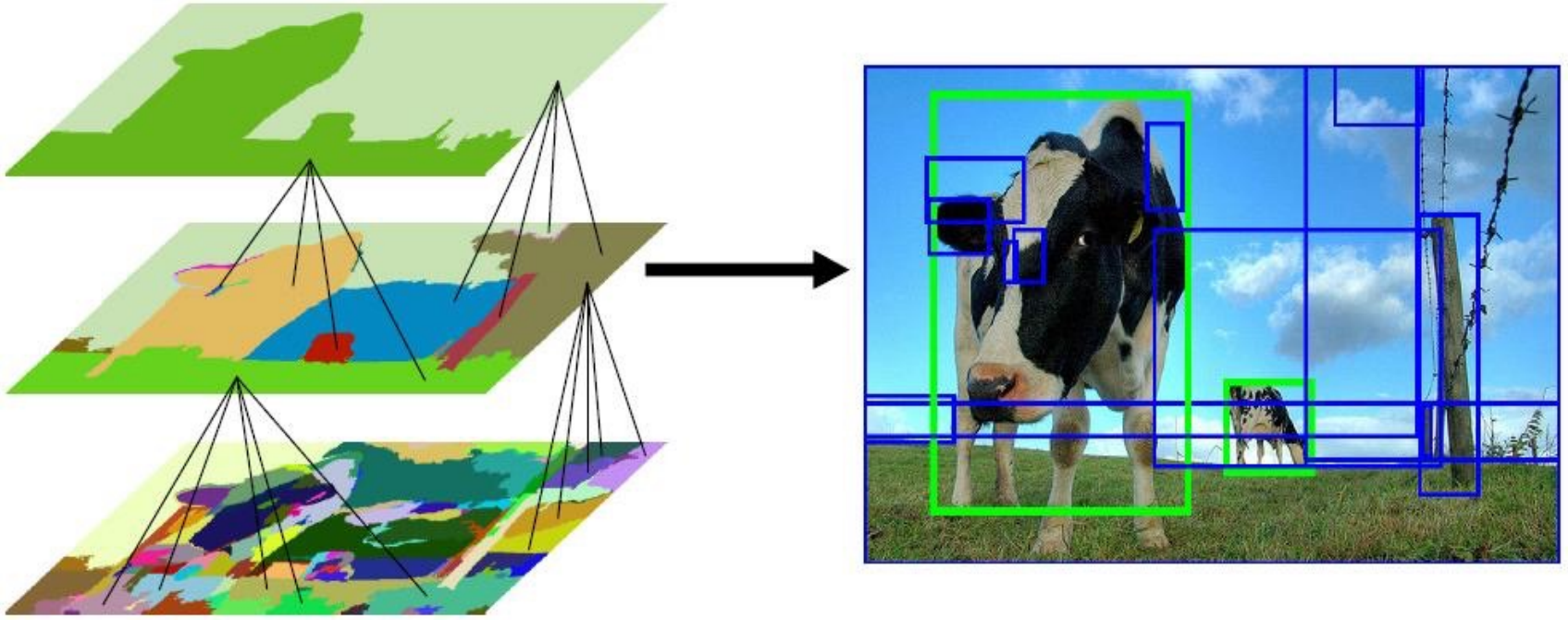


Image Credit: Creative Commons Licenses

Dog (x_1, y_1, w_1, h_1): No
Rabbit (x_2, y_2, w_2, h_2): No
Sheep (x_3, y_3, w_3, h_3): Yes
Background (x_4, y_4, w_4, h_4): No
…

Selective Search

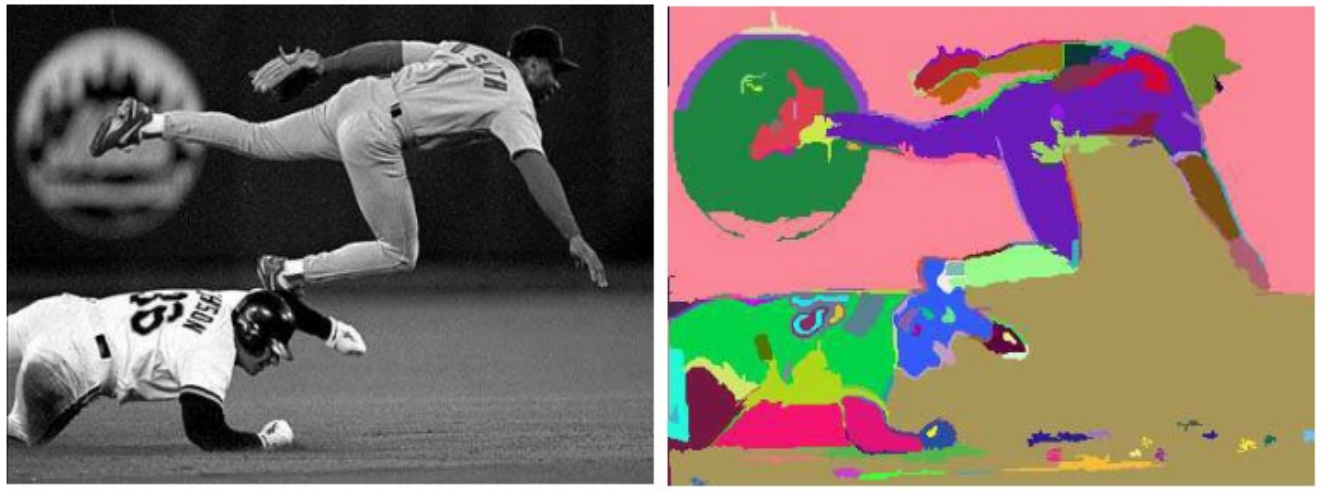
1. Merge two most similar regions based on S.
2. Update similarities between the new region and its neighbors.
3. Go back to step 1. until the whole image is single region.



Felzenszwalb and Huttenlocher., Efficient Graph-Based Image Segmentation, IJCV 2004

Selective Search for Object Detection

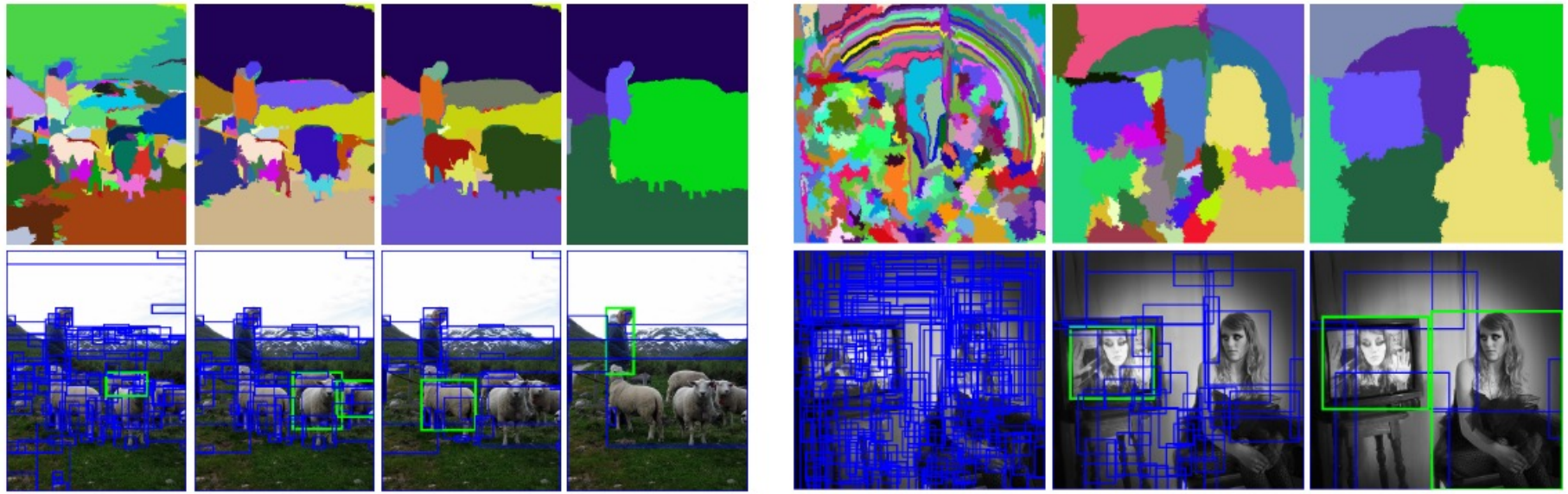
- Selective Search is a region proposal algorithm
- It starts by over-segmenting the image based on intensity of the pixels using a graph-based segmentation (Felzenszwalb and Huttenlocher)
- Selective Search then takes these oversegments as initial input and performs the following steps:
  1. Add all bounding boxes corresponding to segmented parts to the list of region proposals
  2. Group adjacent segments based on similarity
  3. Go to step 1
- It's a bottom-up approach



Felzenszwalb and Huttenlocher., Efficient Graph-Based Image Segmentation, IJCV 2004

Selective Search for Object Detection

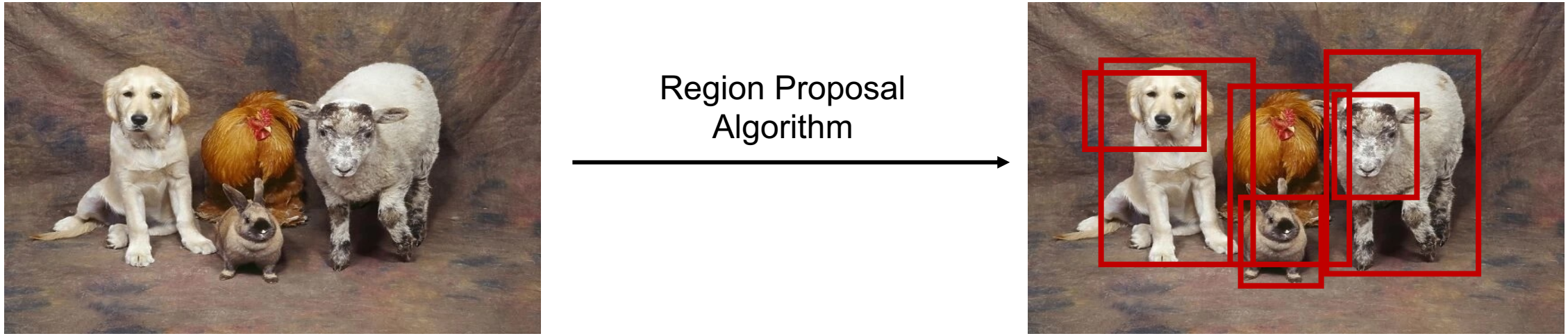
- Selective Search design constraints:
  - Capture all scales (objects can occur at any scale within the image)
  - Diversification (Use diverse set of strategies such as forming regions based on color, texture, shading, etc.)
  - Fast to compute



Van de Sande et al., Segmentation as Selective Search for Object Recognition., ICCV 2011.

Region Proposals

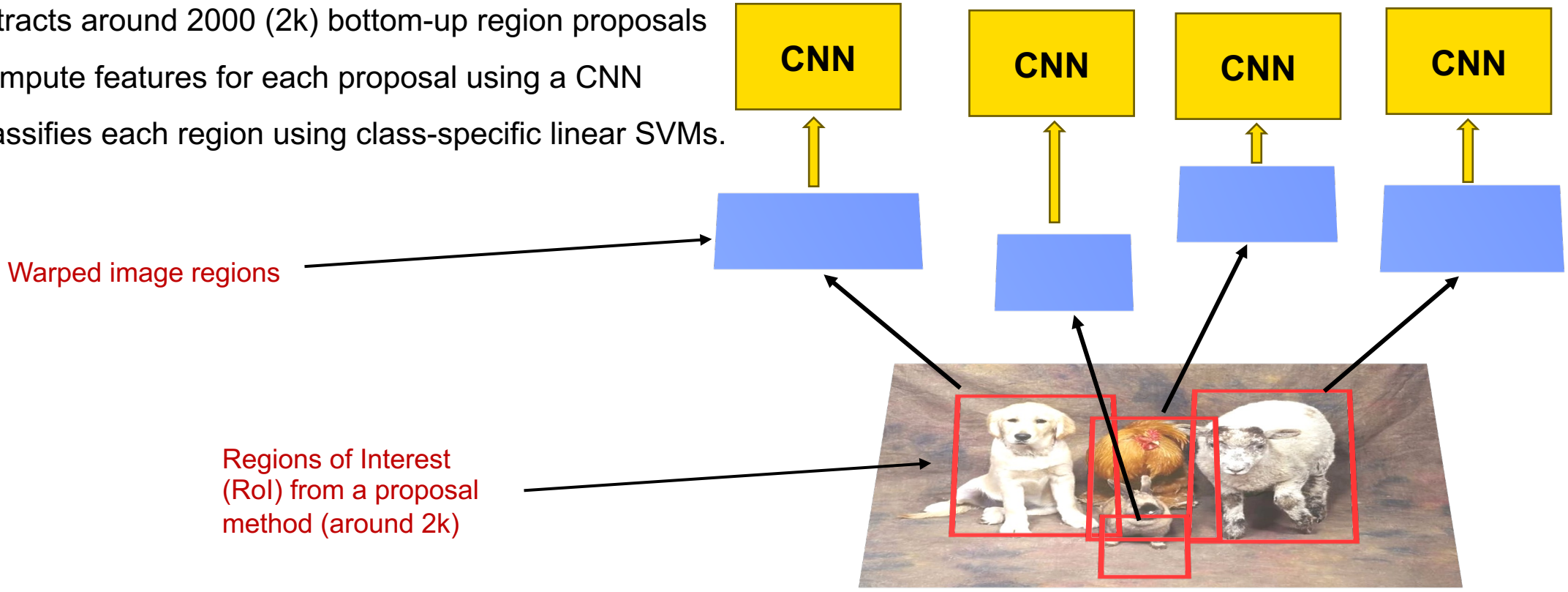
- Find image regions that are likely to contain objects
- Gives around 2000 regions proposals in a few seconds on CPU



Van de Sande et al., Segmentation as Selective Search for Object Recognition., ICCV 2011.

R-CNN: Regions Proposals with CNN Features

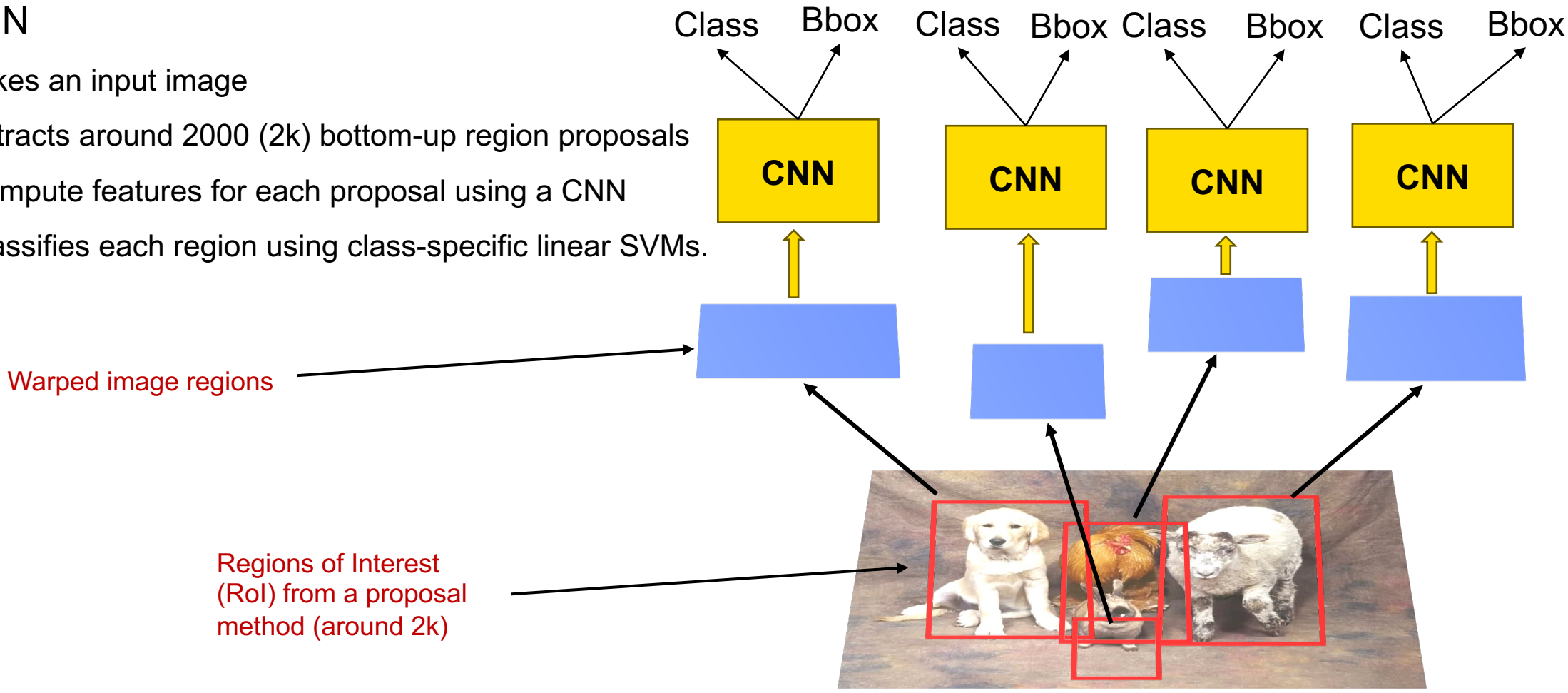
- R-CNN
  1. Takes an input image
  2. Extracts around 2000 (2k) bottom-up region proposals
  3. Compute features for each proposal using a CNN
  4. Classifies each region using class-specific linear SVMs.



Girshick et al., Rich feature hierarchies for accurate object detection and semantic segmentation, CVPR 2014

R-CNN: Regions with CNN Features

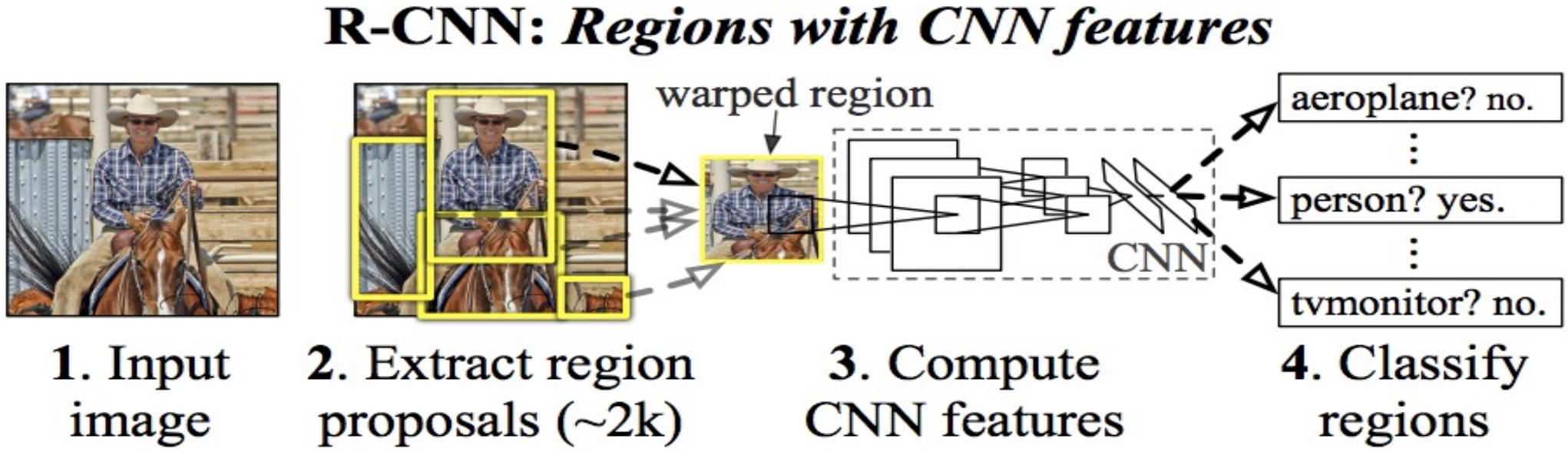
- R-CNN
  1. Takes an input image
  2. Extracts around 2000 (2k) bottom-up region proposals
  3. Compute features for each proposal using a CNN
  4. Classifies each region using class-specific linear SVMs.



Girshick et al., Rich feature hierarchies for accurate object detection and semantic segmentation, CVPR 2014

R-CNN: Regions with CNN Features

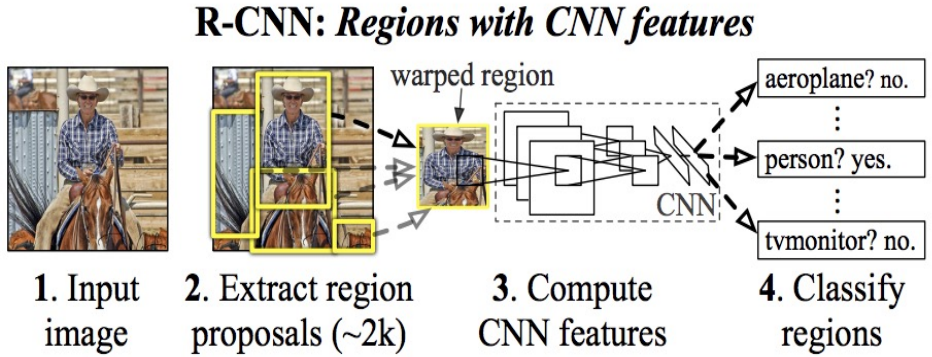
- R-CNN



Girshick et al., Rich feature hierarchies for accurate object detection and semantic segmentation, CVPR 2014

R-CNN was very slow!

- Why?
  - Training was a multi-stage pipeline
    1. R-CNN first fine-tunes a ConvNet on object proposals
    2. Fits SVM to ConvNet features
    3. Bounding box regressors are learned
  - Training is expensive in space and time
    - For SVM and bounding-box regressor training, features are extracted from each object proposals in each image and written to disk
    - VGG-16 takes 2.5 GPU-days for the 5k images of the PASCAL VOC 2007 trainval set
    - Extracted features require hundreds of GB of storage
  - The selective search is a fixed algorithm. No learning is happening!
  - At test time, need to pass approximately 2000 independent forward passes for each image. Object detection with VGG-16 takes 47s/image (on a GPU)

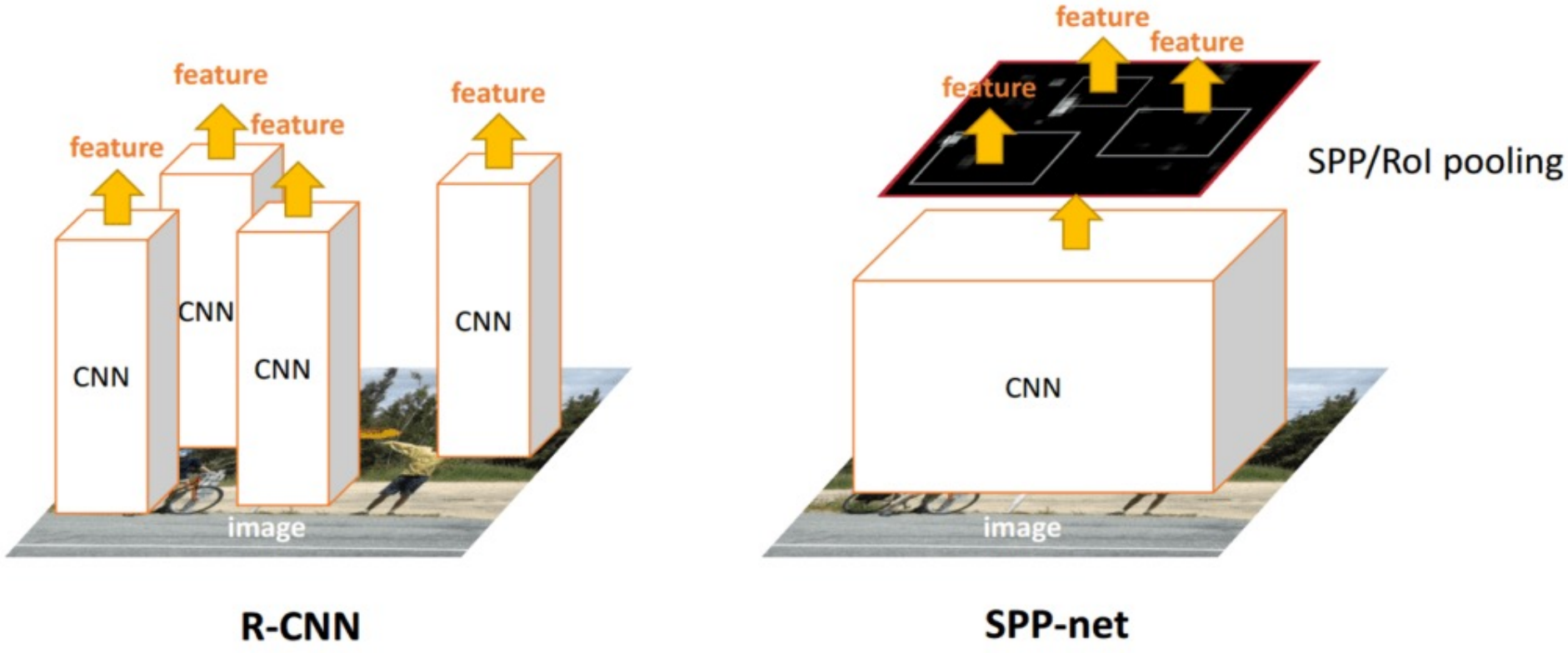


Girshick., Fast R-CNN, ICCV 2015.

➢Spatial Pyramid Pooling (SPP-Net)

➢ Spatial Pyramid Pooling (SPP)-Net

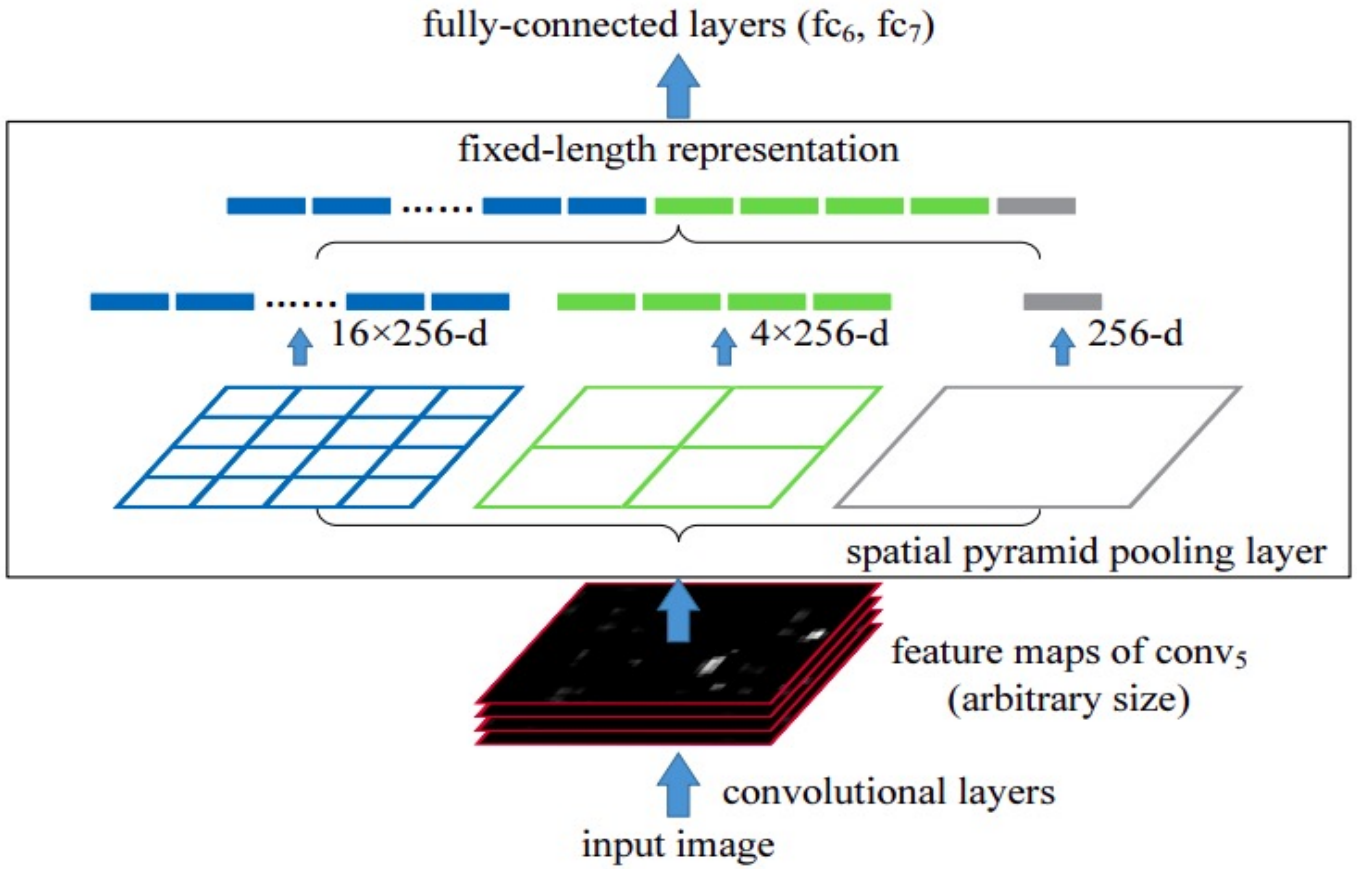
- ➢ “Pool” features into a common size



He et al. “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition”, ECCV 2014.

➢Spatial Pyramid Pooling (SPP-Net)

➢ "Pool" features into a common size



He et al. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition", ECCV 2014.

➢Spatial Pyramid Pooling (SPP-Net)

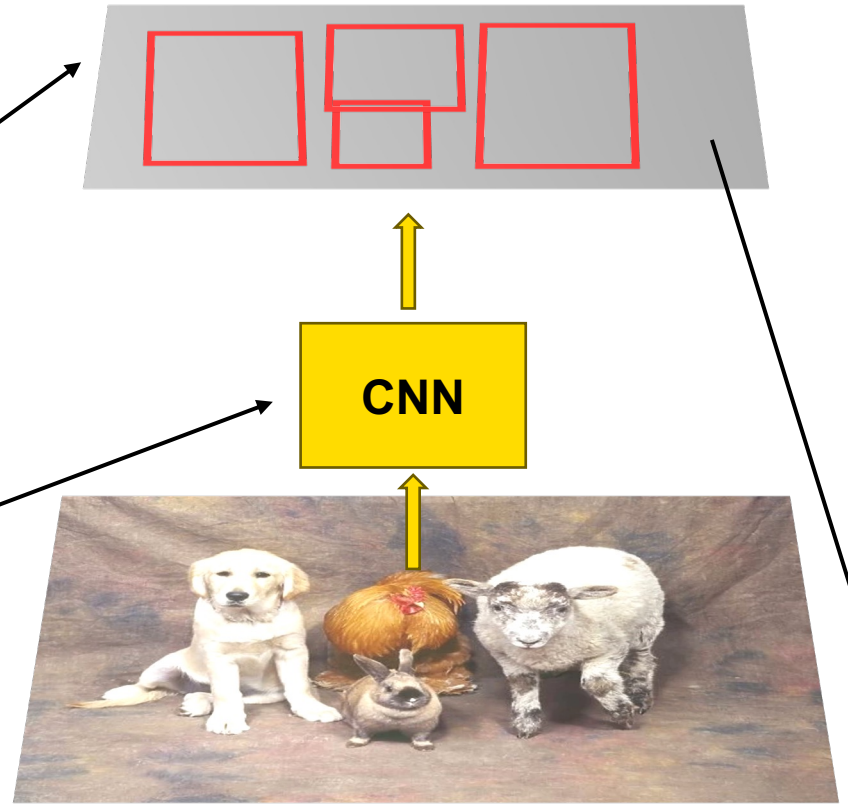
- SPP-Net solved the R-CNN problem of being slow at test time
- However, still has problems inherited from R-CNN:
  - Training is still slow (a bit faster than R-CNN)
  - Training scheme is still complex
  - Still no end-to-end training

He et al. “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition”, ECCV 2014.

Fast R-CNN

- Idea: Pass the image through CNN and

then crop RoIs from the feature map



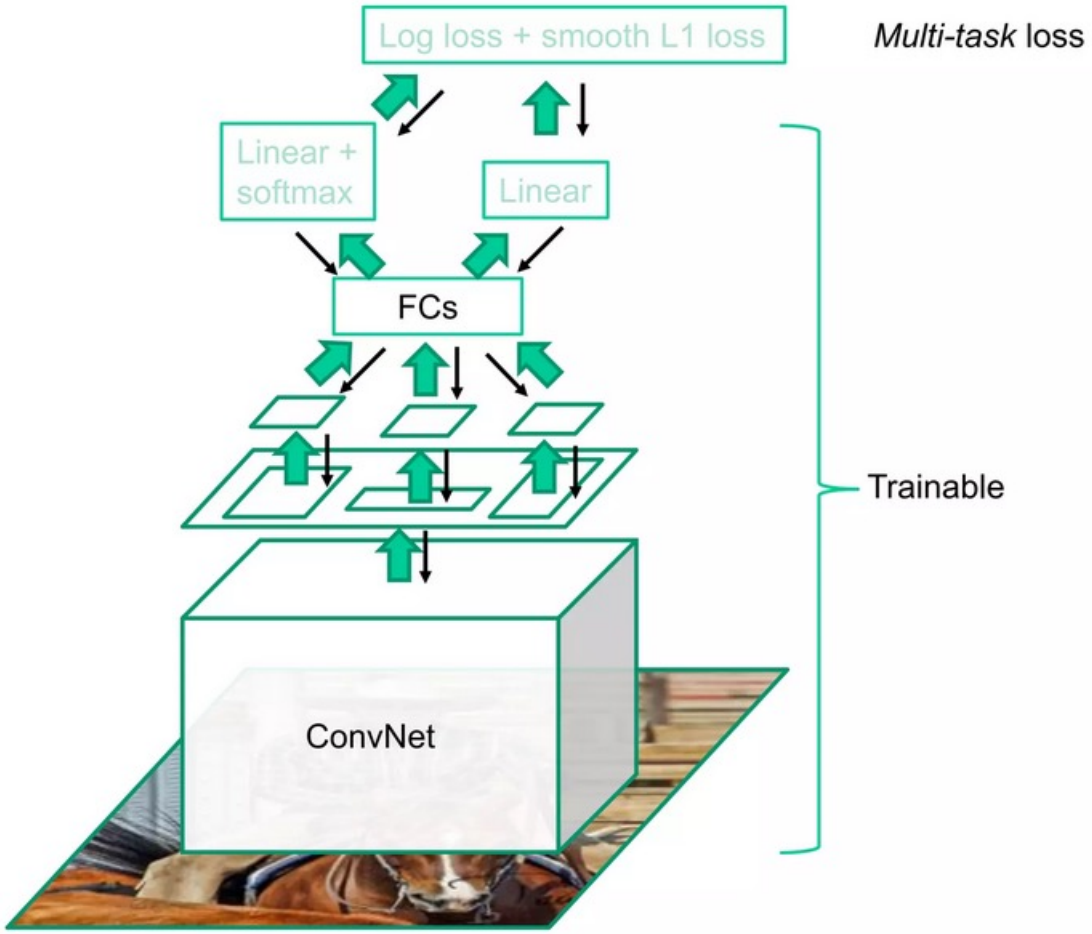
Girshick., Fast R-CNN, ICCV 2015.

```
import tensorflow as tf
from tensorflow.keras.applications import VGG16

model = VGG16(include_top=False, weights = 'imagenet')
print(model.summary())
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, None, None, 3)]   0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, None, None, 64)    1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, None, None, 64)    36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, None, None, 64)    0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, None, None, 128)   73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, None, None, 128)   147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, None, None, 128)   0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, None, None, 256)   295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, None, None, 256)   590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, None, None, 256)   590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, None, None, 256)   0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, None, None, 512)   1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, None, None, 512)   0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, None, None, 512)   0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
```

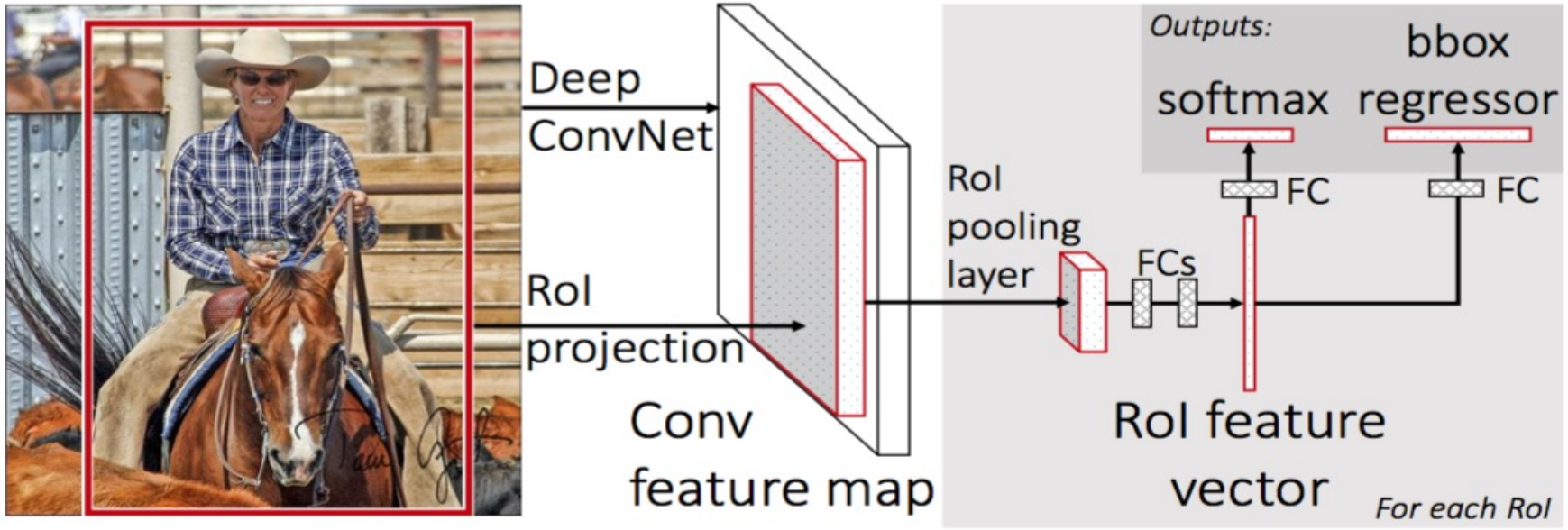
Fast R-CNN training



Girshick., Fast R-CNN, ICCV 2015.

Fast R-CNN

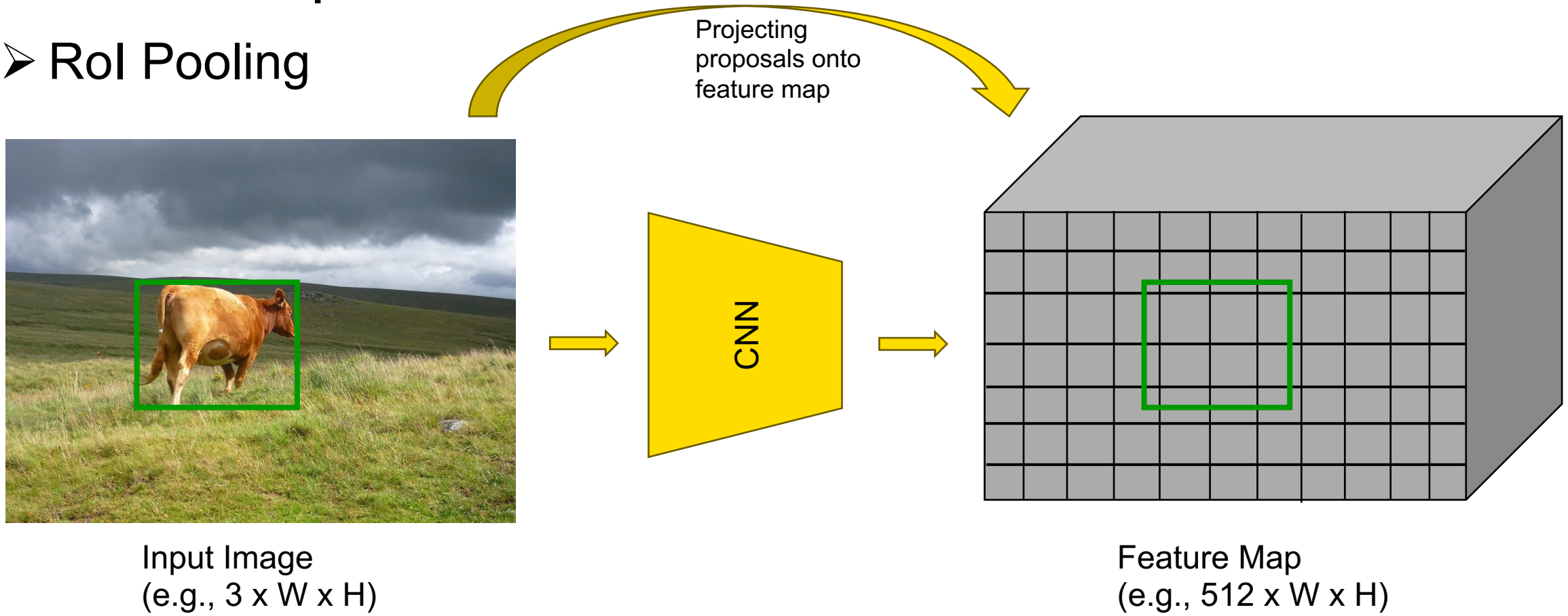
- Training is end-to-end, using multi-task loss
- No disk storage is required for feature catching
- High detection quality (high mAP) than R-CNN



Girshick., Fast R-CNN, ICCV 2015.

How to crop features?

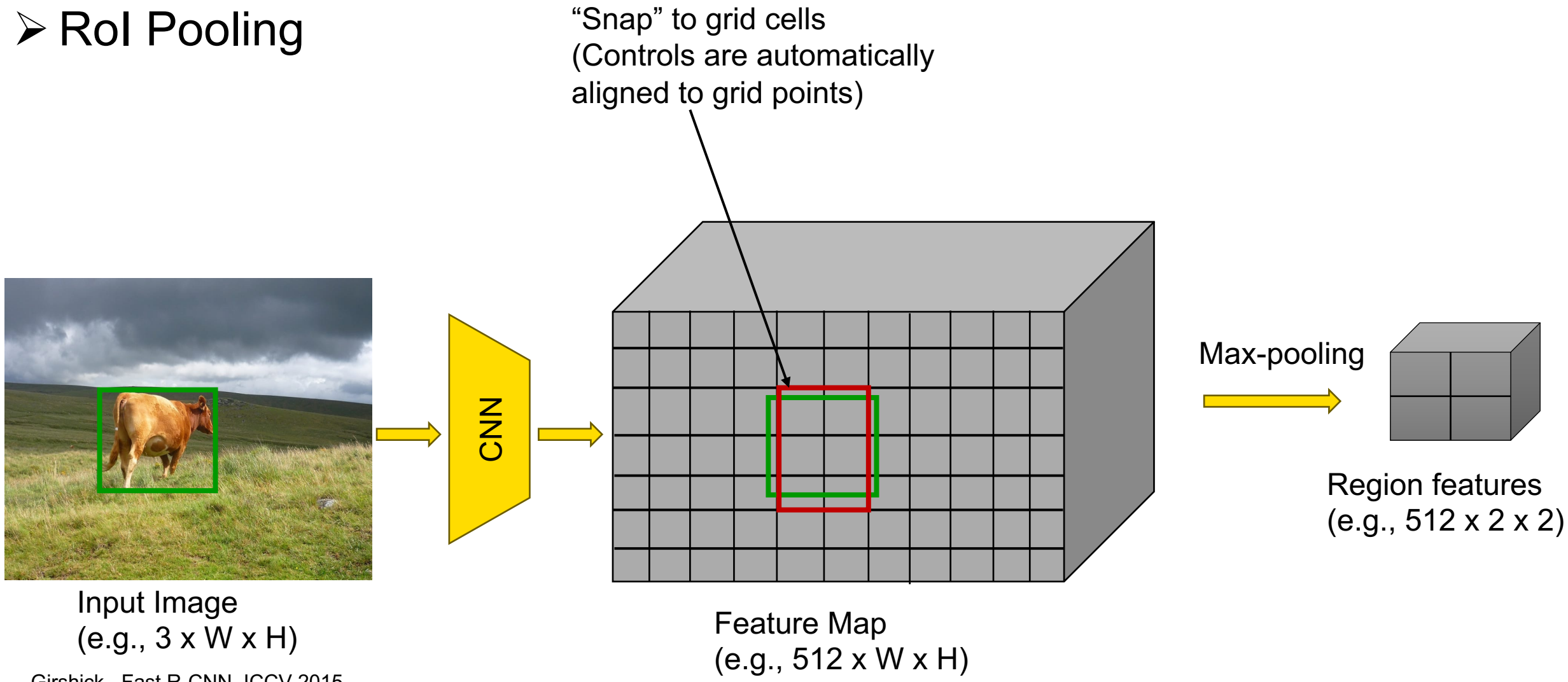
- RoI Pooling



Girshick., Fast R-CNN, ICCV 2015.

How to crop features?

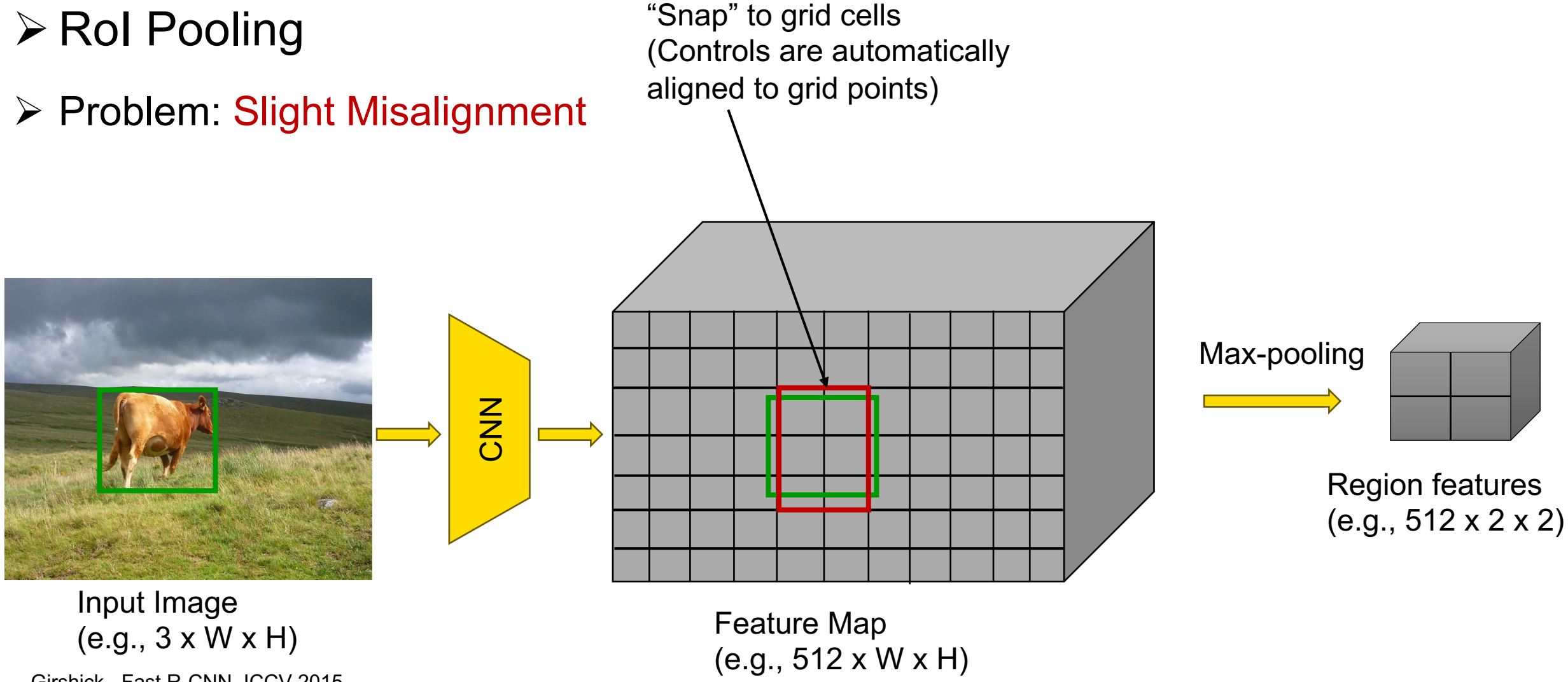
- RoI Pooling



Girshick., Fast R-CNN, ICCV 2015.

How to crop features?

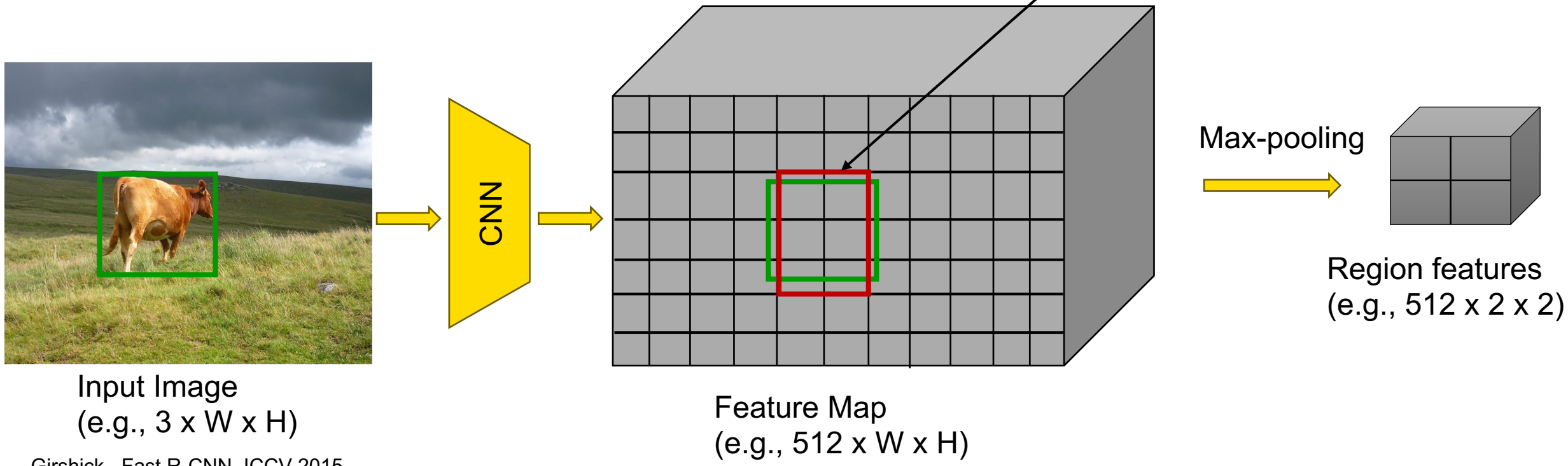
- RoI Pooling
- Problem: Slight Misalignment



Girshick., Fast R-CNN, ICCV 2015.

How to crop features?

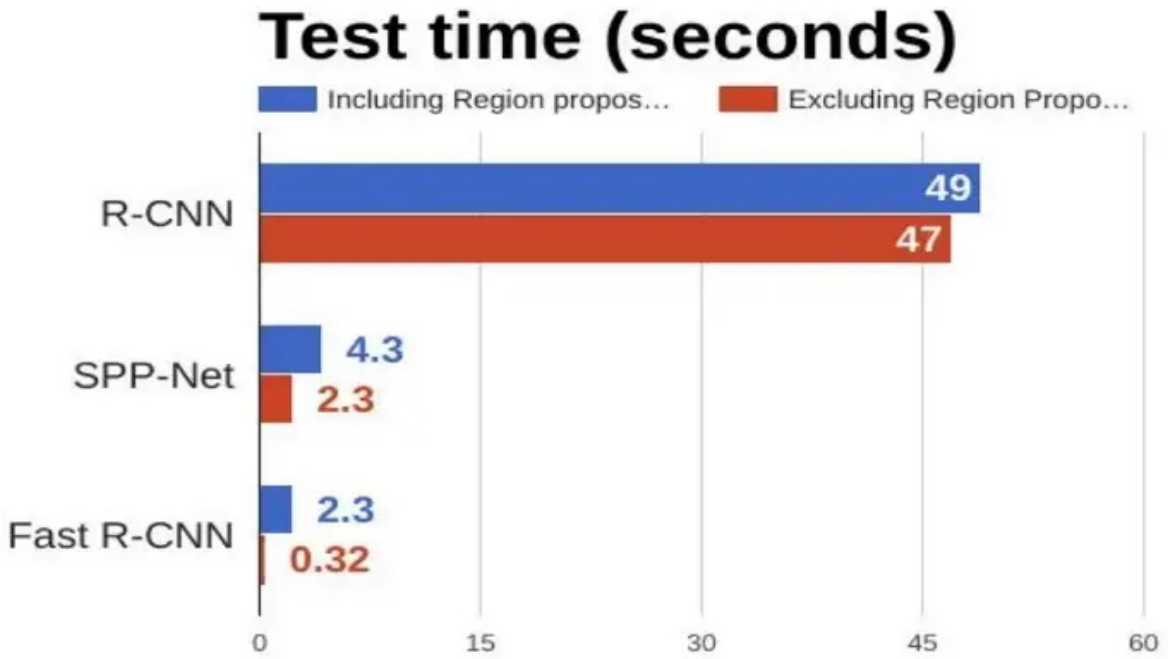
- RoI Pooling
- Problem: Slight Misalignment
- Solution: ROI Align using bilinear interpolation



Girshick., Fast R-CNN, ICCV 2015.

R-CNN vs SPP-Net vs Fast R-CNN

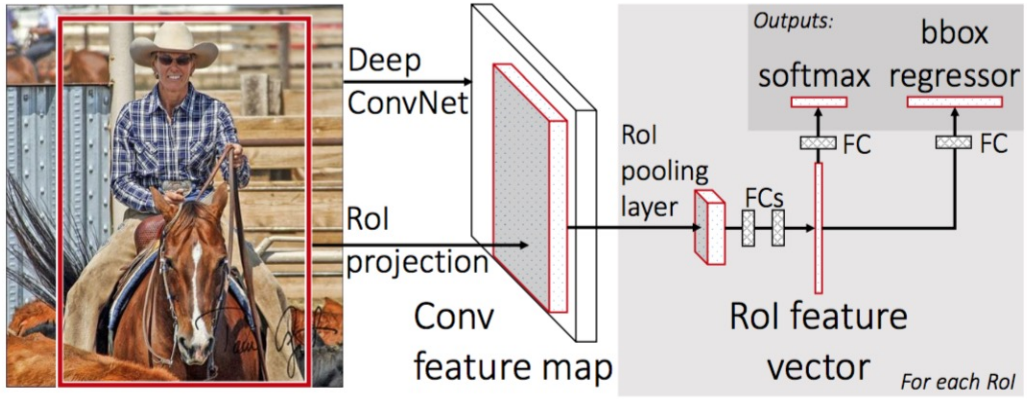
- Observation: During test time, Fast R-CNN computation time is dominated by computing region proposals.



Girshick., Fast R-CNN, ICCV 2015.

How can be improve Fast R-CNN?

- Observation: The convolutional (conv) feature maps used by region-based detectors (e.g., Fast R-CNN) can also be used for generating region proposals
- Time spent on region proposals is the real bottleneck in state-of-the-art object detection systems



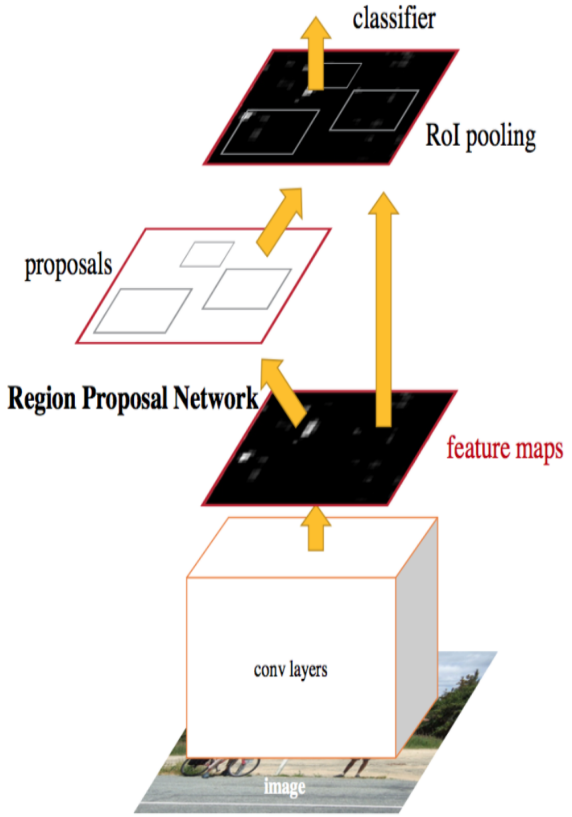
- Can we unify region proposals and object detection networks (e.g., Fast R-CNN)?
- Idea: Region Proposal Networks (RPNs)
  - Computing proposals with deep neural networks.

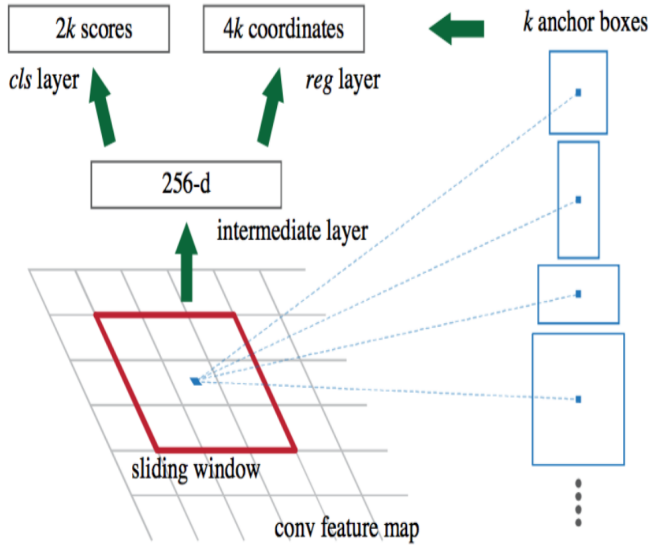
Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NeurIPS 2015.

Region Proposal Network (RPN)

- Slide a small window over the feature map
  - Predict object/no object
  - Regress bounding box coordinates
  - Box regression is with reference to anchors

    (3 scales x 3 aspect ratios)

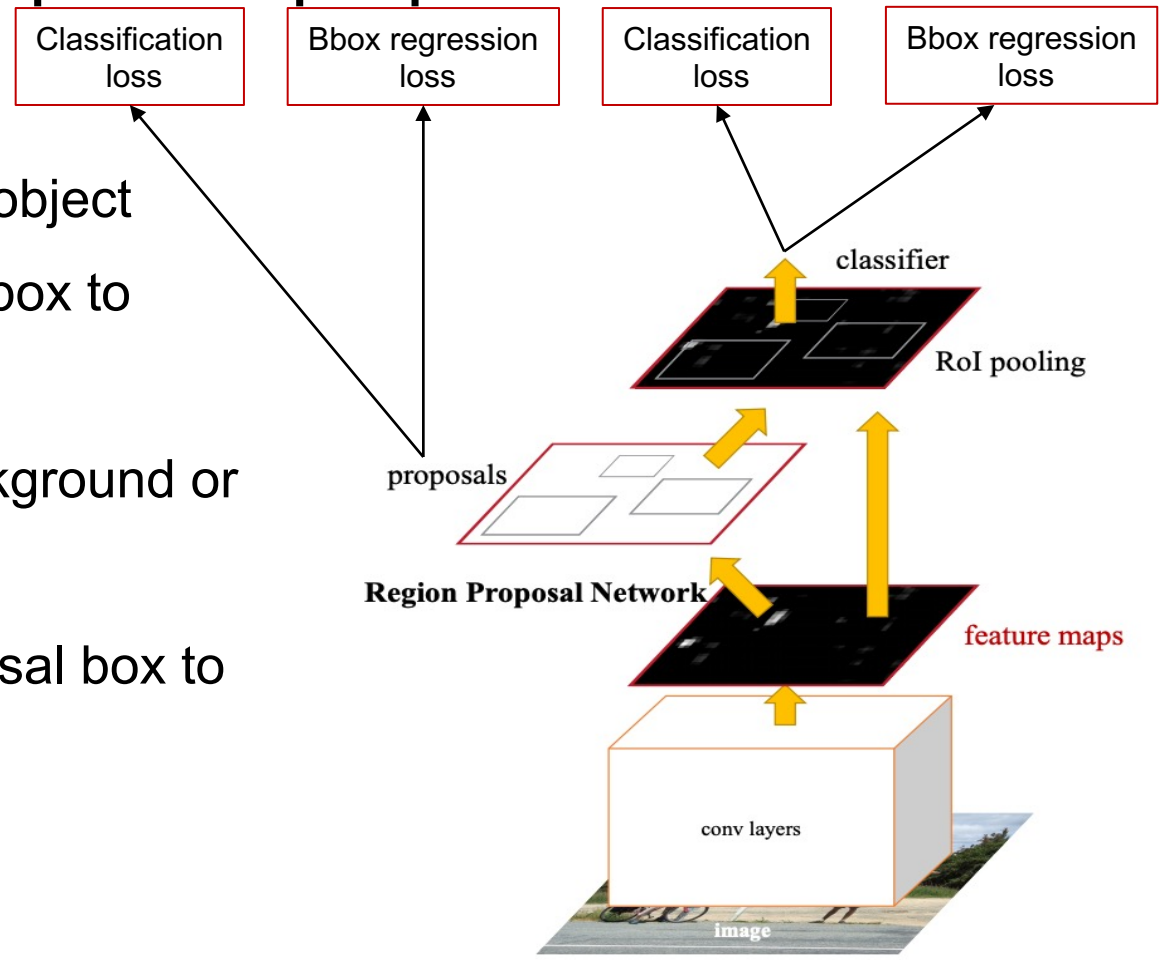




Ren et al., “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NeurIPS 2015.

Faster R-CNN

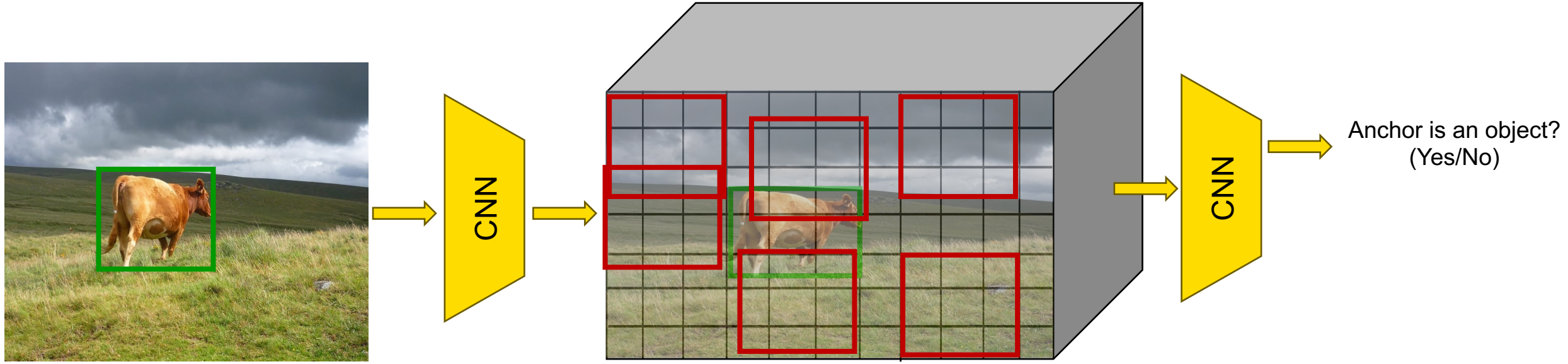
- Region Proposal Network (RPN) to predict proposals from features
- Jointly train with four loss
  - **RPN classification**: anchor box is an object/not object
  - **RPN regression**: predict transform from anchor box to proposal box
  - **Object classification**: classify proposals as background or object class
  - **Object regression**: predict transform from proposal box to object box.



Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NeurIPS 2015.

Faster R-CNN

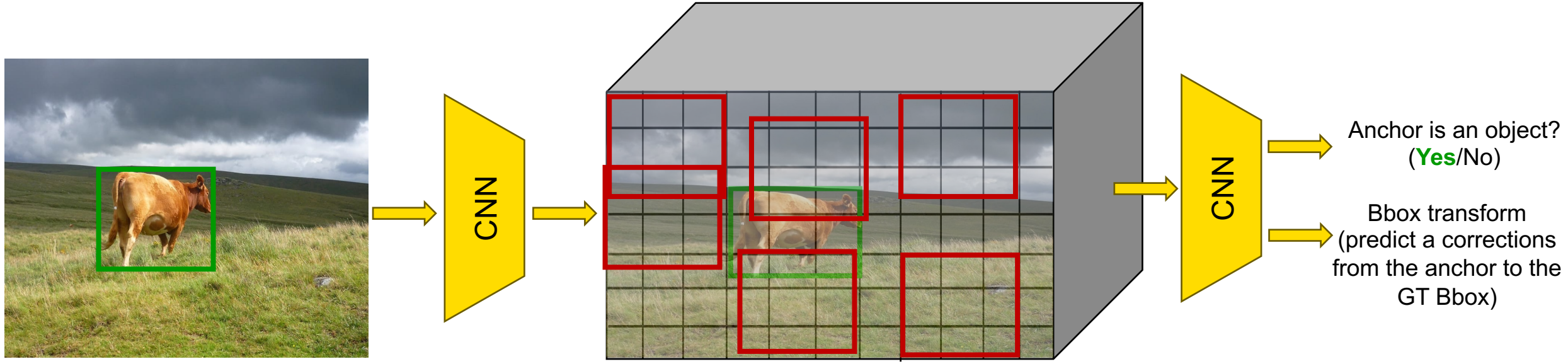
- Anchor Boxes
  - Set of predefined bounding boxes of a certain height and width. These are defined to capture the scale and aspect ratio of specific object classes we want to detect (typically chosen based on object sizes in the training datasets)



Ren et al., “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NeurIPS 2015.

Faster R-CNN

- Anchor Boxes
  - Set of predefined bounding boxes of a certain height and width. These are defined to capture the scale and aspect ratio of specific object classes we want to detect (typically chosen based on object sizes in the training datasets)



Ren et al., “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NeurIPS 2015.

Faster R-CNN loss

- In practice, we have **k** different **anchor boxes of different size and scale at each point**.
- **reg** layer has **4k** outputs encoding the coordinates of k boxes
- **cls** layer outputs **2k** scores that estimate probability of object or not object for each anchor box
- Loss function

i = anchor index in minibatch

Log loss — Ground truth objectness label — Smooth L1 loss

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

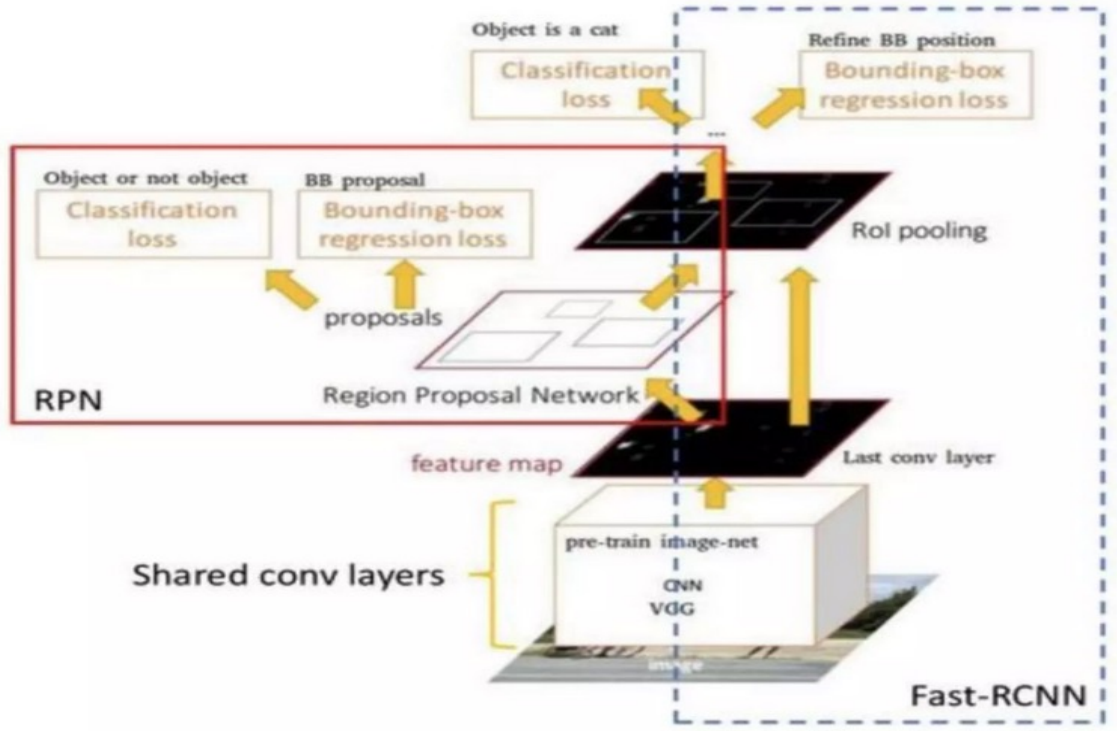
Predicted probability of being an object for anchor i

Coordinates of the predicted bounding box for anchor i

True box coordinates

In practice $\lambda = 10$, so that both terms are roughly equally balanced

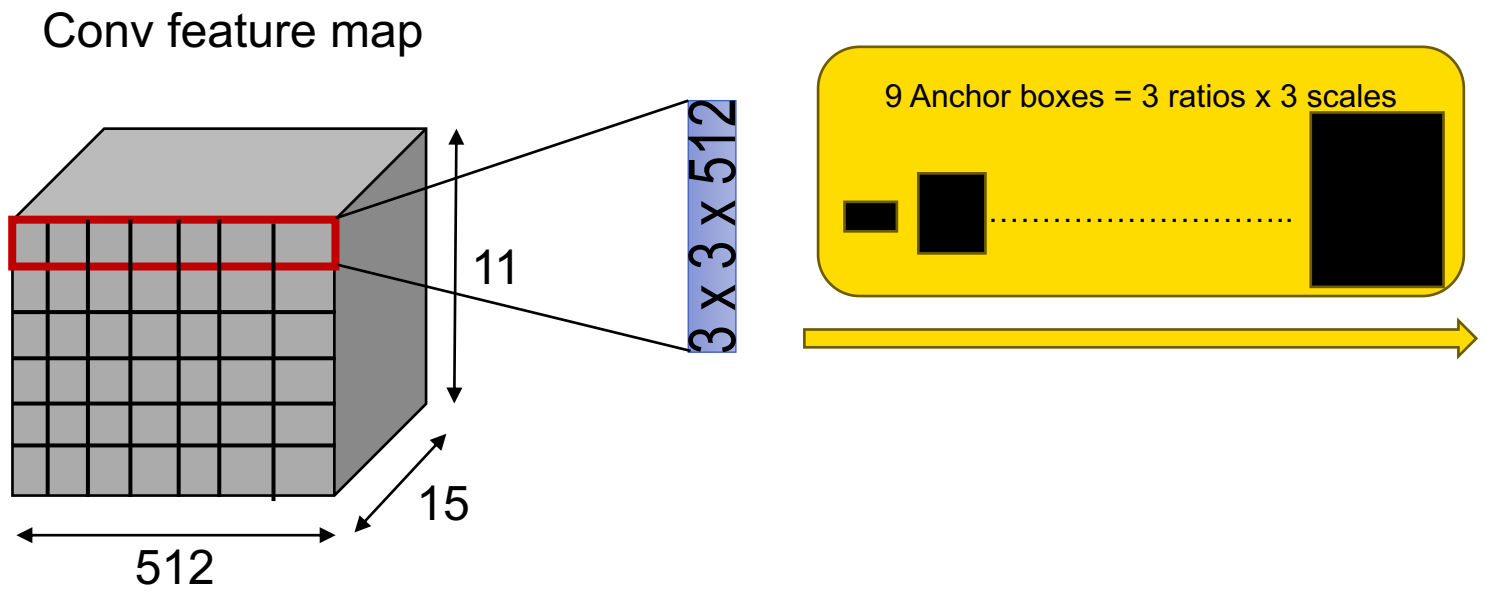
N_{cls} = Number of anchors in minibatch (~ 256)
N_{reg} = Number of anchor locations (~ 2400)



Ren et al., “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NeurIPS 2015.

Region Proposals and Anchor Boxes

- Extract 9 Proposals relative to 9 Anchors



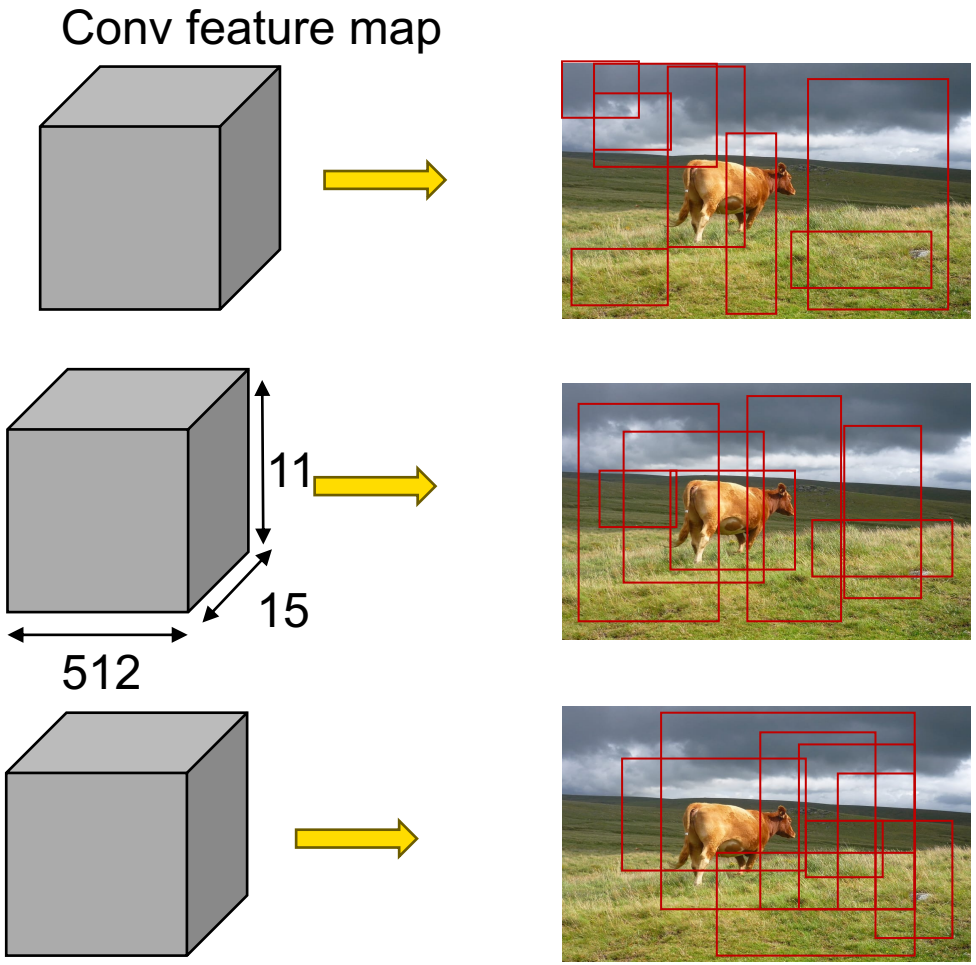
Input: each sliding window

Output: For i = 1, 2, …., 9

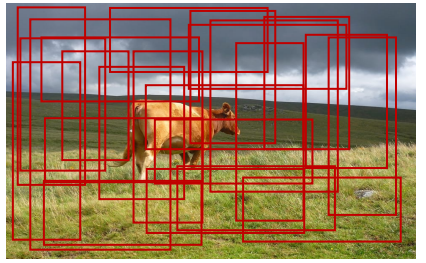
- 4 coordinates (reg)
- 2 scores (cls)

Ren et al., “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NeurIPS 2015.

Region Proposals and Anchor Boxes



The proposals highly overlaps each other!
Need to reduce redundancy.



Total # of proposals: 11 x 15 x 9 = 1485

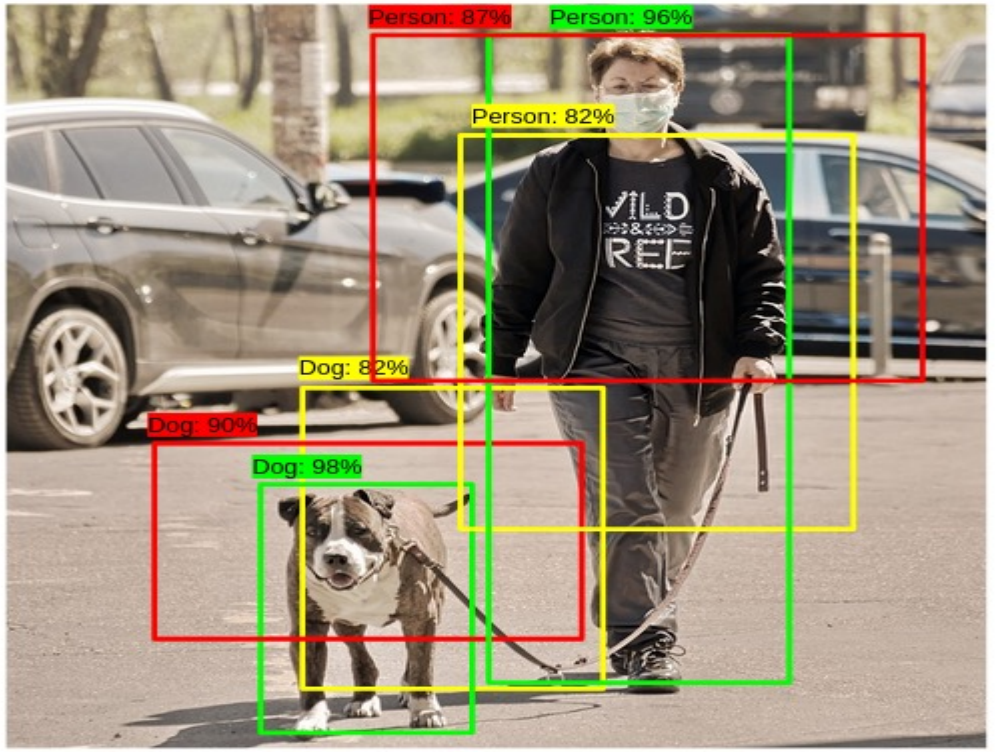
Total # of windows

# of proposals per a window

Ren et al., “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NeurIPS 2015.

Reduce redundancy by Non-Maximum Suppression (NMS)

- **Step 1**: Take the most probable proposal from 1485 proposals
- **Step 2**: Compute the IoU between the most probable and other proposals, and reduce proposals having IoU > threshold (0.7)
- **Step 3**: Get the most probable proposals among the rest (original – reduced) proposals and repeat steps 1 to 3.



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

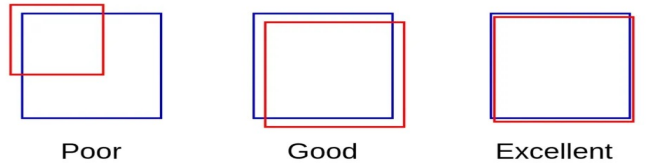


Image Credit: Analytics Vidya. https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/

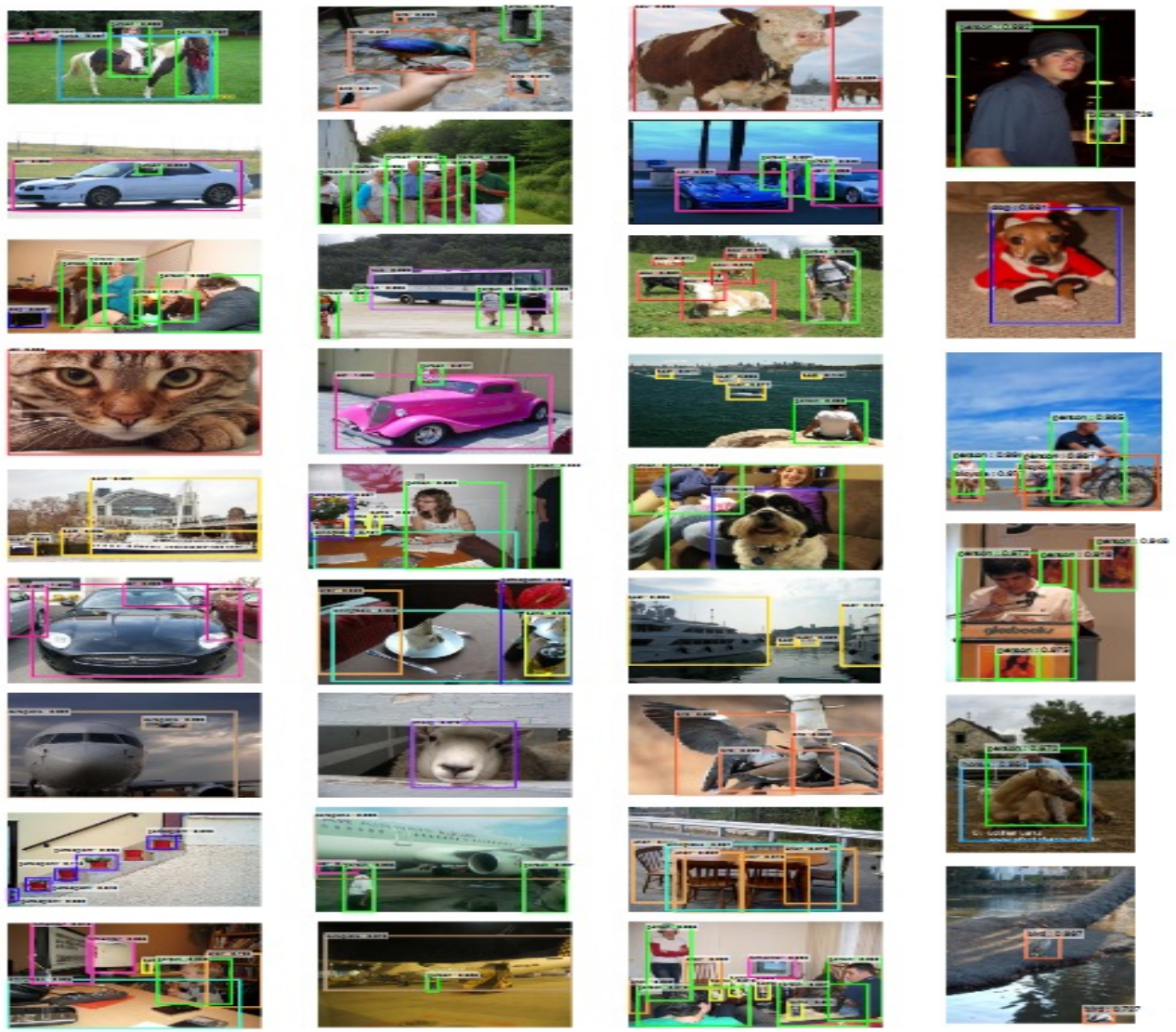
Online hard example mining (OHEM)

- Class imbalance hurts training (Overwhelming number of easy examples and small number of hard examples)
- We are training the model to learn background rather than detecting objects.
- OHEM is a bootstrapping technique that modifies SGD to sample from examples in a non-uniform way depending on the current loss of each example.
- Sort anchors by their calculated loss, apply NMS.
- Pick the top ones such that ratio between the picked negatives and positives is at most 3:1.
- Faster R-CNN selects 256 anchors: 128 positive, 128 negative

Image Credit: Analytics Vidya. https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/

Faster R-CNN selected examples

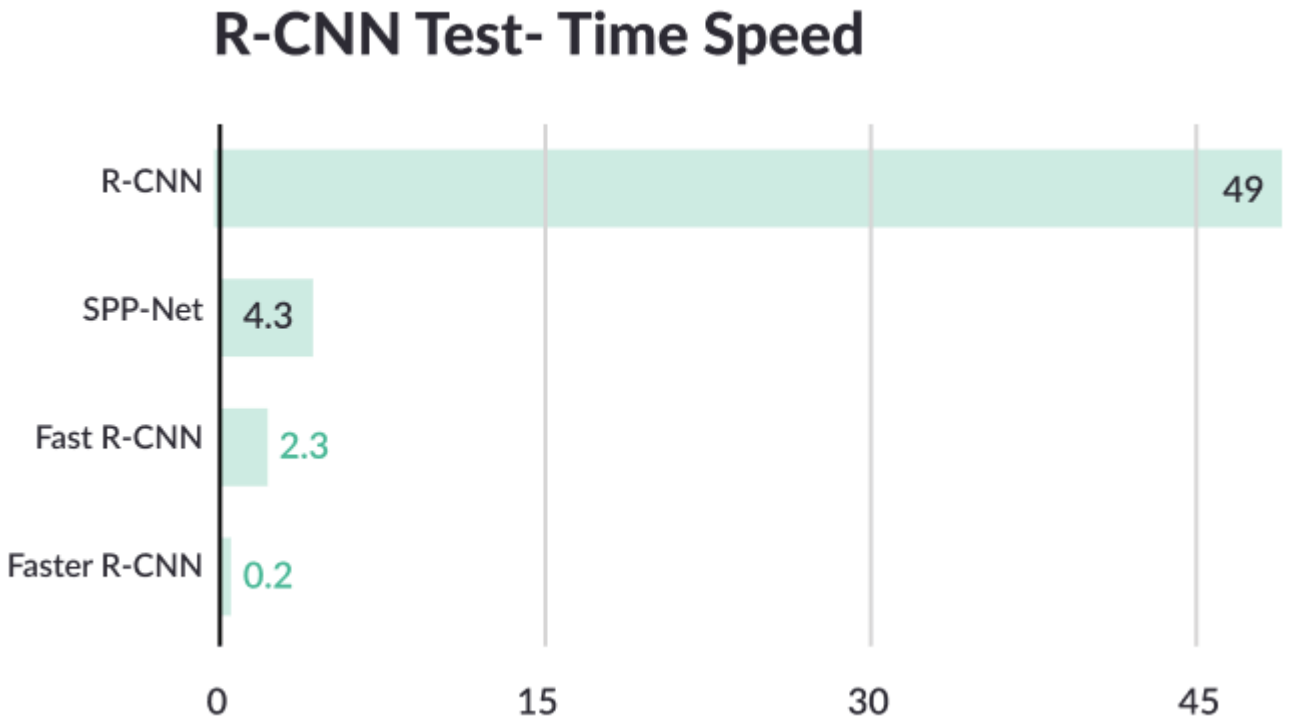
PASCAL
VOC 2007
test set using
Faster RCNN



Ren et al., “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NeurIPS 2015.

Faster R-CNN speed

Faster R-CNN is super fast



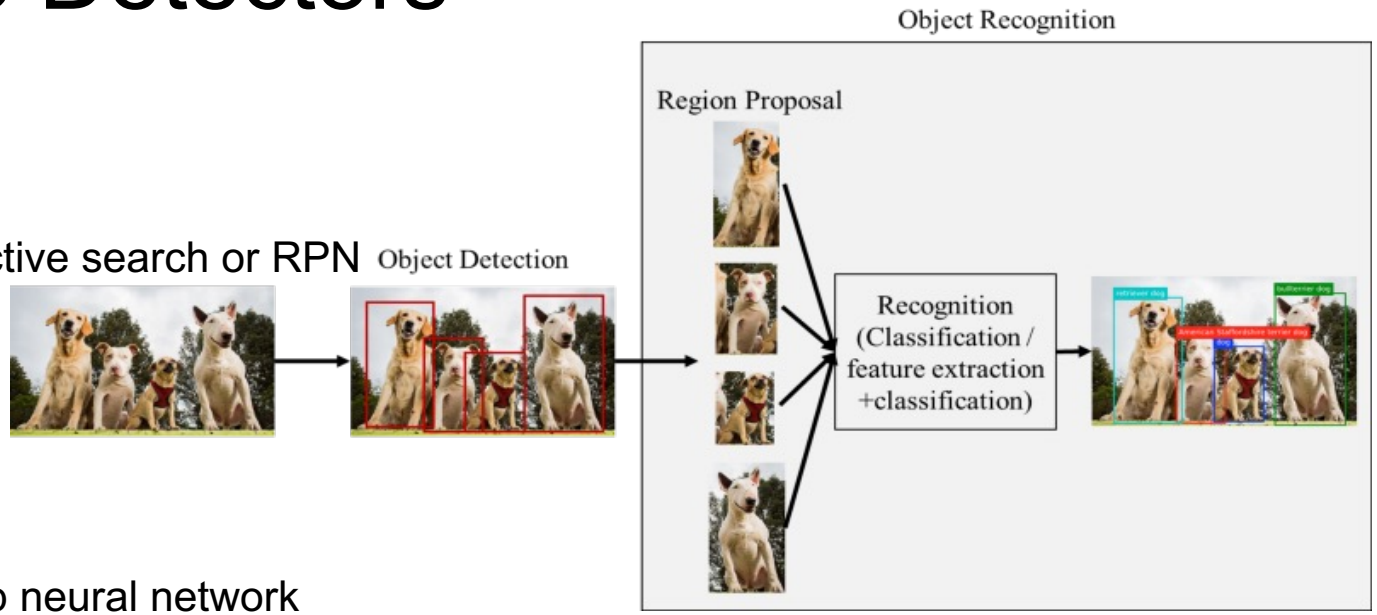
Ren et al., “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NeurIPS 2015.

Two-stage vs One-stage Detectors

- Two-stage detectors
  - Models in the R-CNN family

  Step 1. The model proposes a set of RoIs by selective search or RPN

  Step 2. Classification of region proposals



- Single-stage detectors
  - Detecting objects in images using a single deep neural network

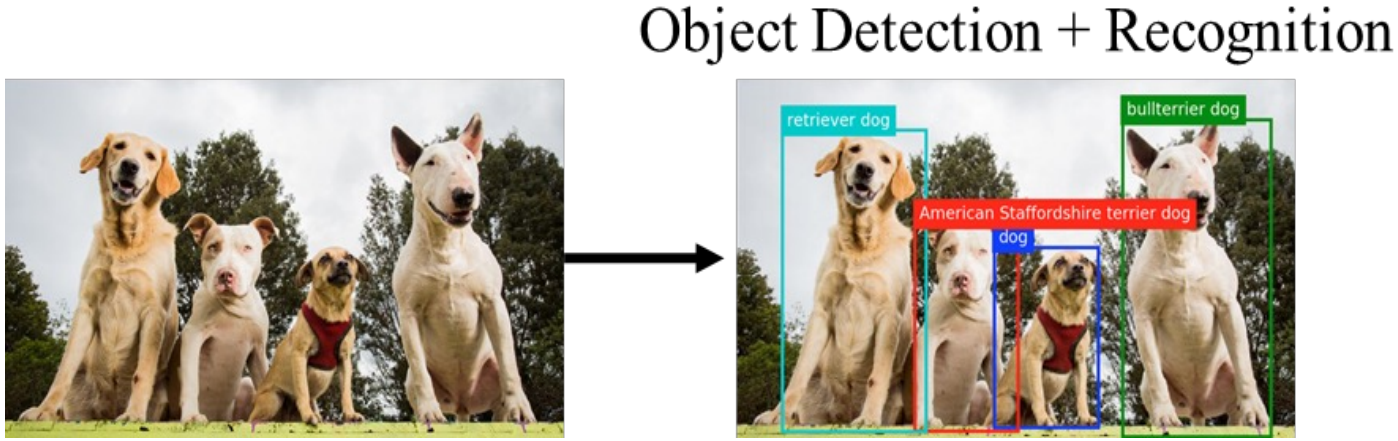
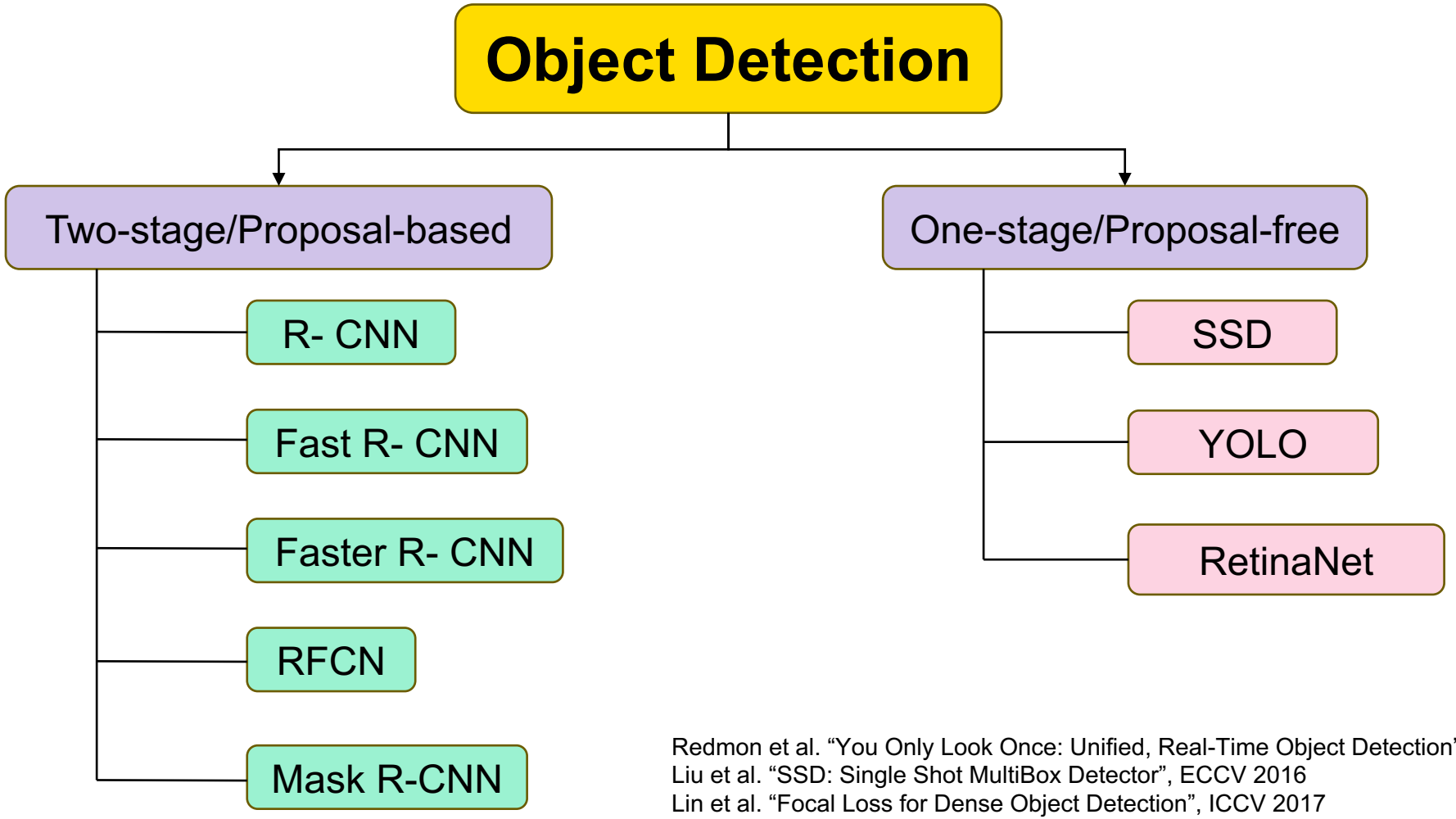


Image Credits: https://github.com/yehengchen/Object-Detection-and-Tracking/blob/master/Two-stage%20vs%20One-stage%20Detectors.md

Two-stage vs One-stage Detectors



Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”, CVPR 2016
Liu et al. “SSD: Single Shot MultiBox Detector”, ECCV 2016
Lin et al. “Focal Loss for Dense Object Detection”, ICCV 2017
He et al. “Mask R-CNN”, ICCV 2017
Dai et al. “R-FCN: Object Detection via Region-based Fully Convolutional Networks, NeurIPS 2016

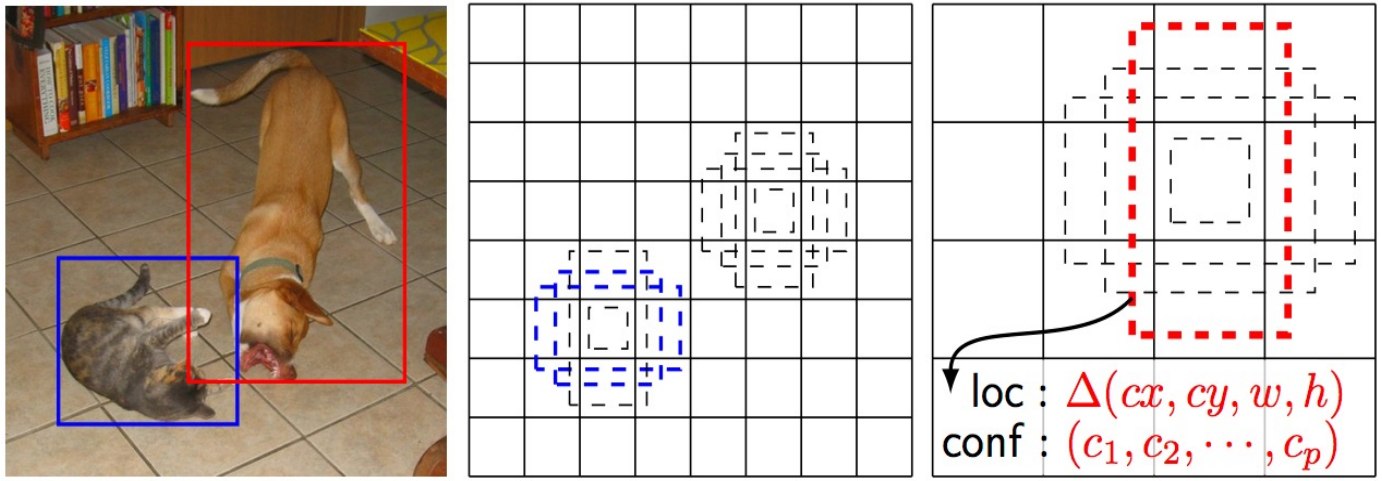
SSD: Single Shot MultiBox Detector

- Two-stage pipeline: Propose bounding boxes, resample pixels or features for each box, and apply a high-quality classifier.
- Can we speed-up by eliminating bounding box proposals and the subsequent pixel or feature resampling stage?
- Series of modifications:
  - Using small convolutional filter to predict object categories and offsets in bounding box locations
  - Using separate filters for different aspect ratio detections
  - Applying filters to multiple feature maps from the later stages of the network to perform detection at multiple scales

Liu et al. “SSD: Single Shot MultiBox Detector”, ECCV 2016

SSD: Single Shot MultiBox Detector

- SSD has two components: a backbone network and SSD head
- Backbone network is a pre-trained image classification network as a feature extractor (e.g., ResNet trained on ImageNet)
- SSD head has one or more convolutional layers and outputs bounding boxes and classes of objects in the spatial location of the final layers activations.

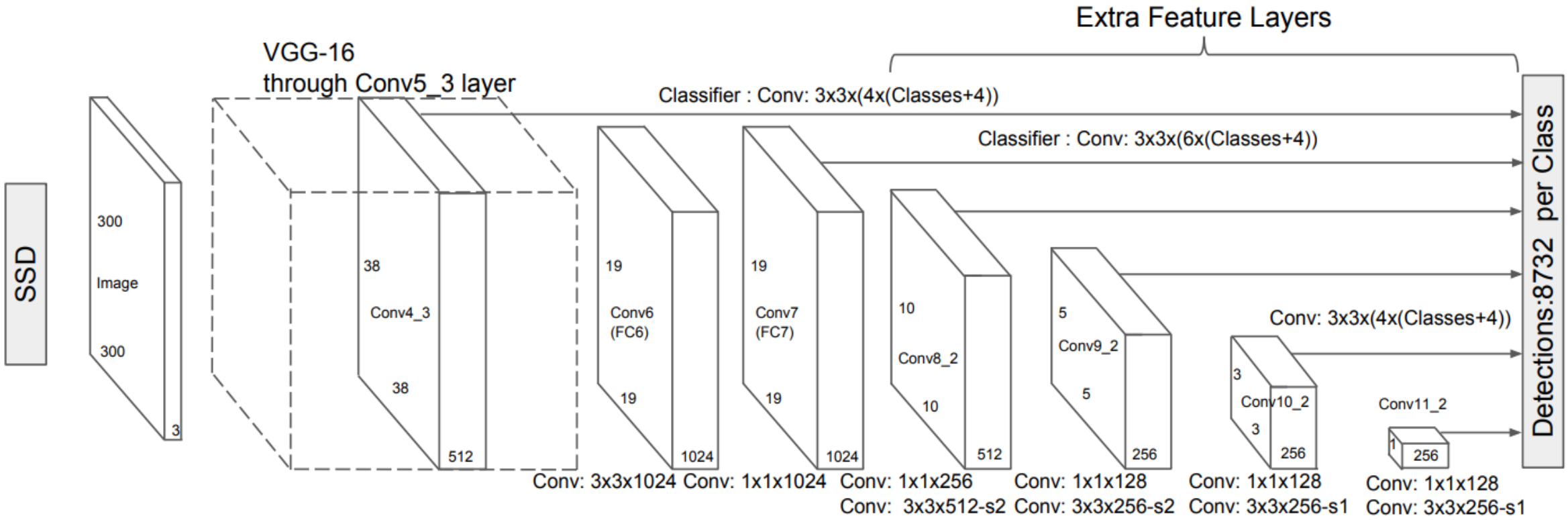


(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Liu et al. "SSD: Single Shot MultiBox Detector", ECCV 2016

SSD: Single Shot MultiBox Detector

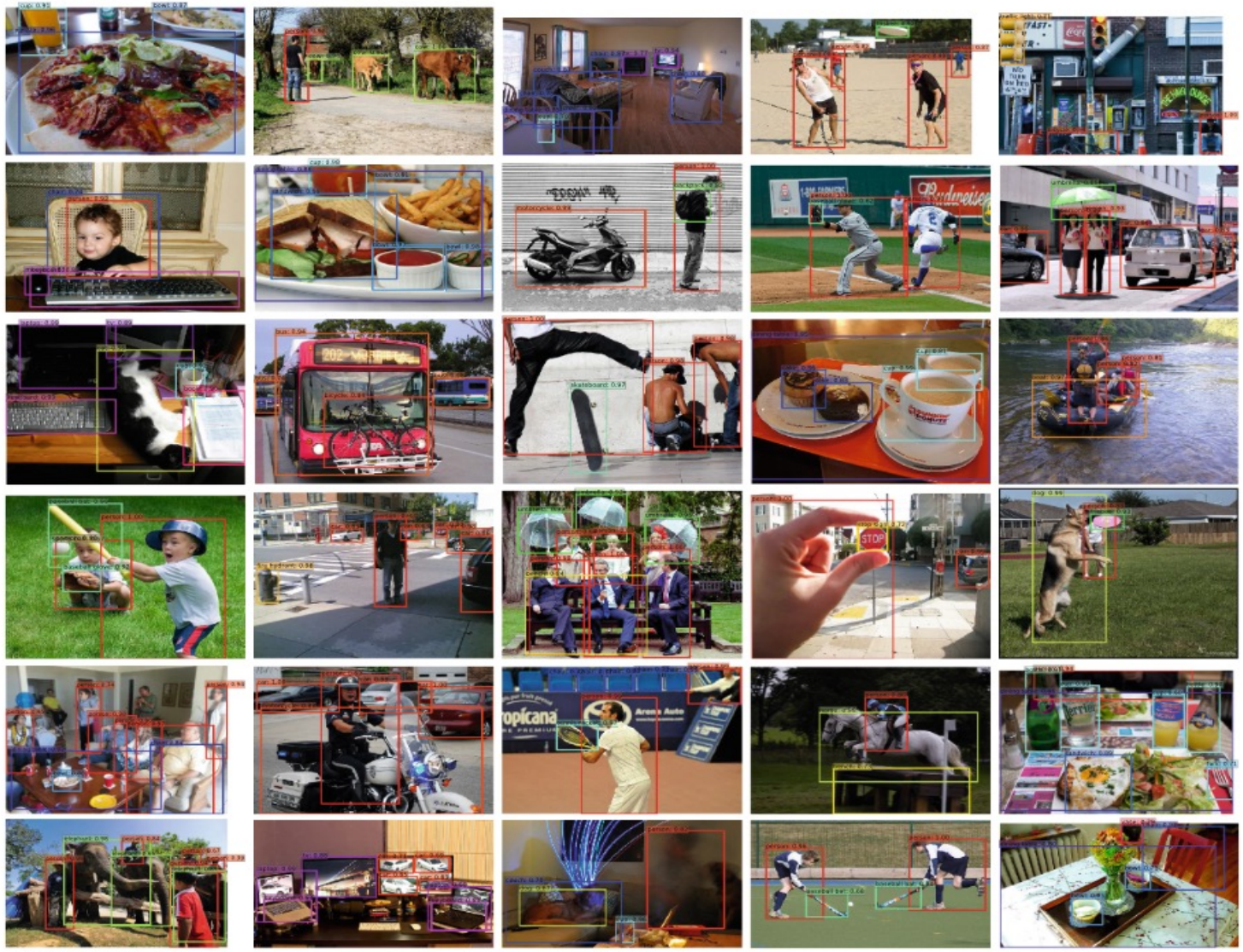
- SSD has two components: a backbone network and SSD head



Liu et al. “SSD: Single Shot MultiBox Detector”, ECCV 2016

SSD: Single Shot MultiBox Detector

- Findings
  - Data augmentation is crucial
  - More default box shapes is better (different scales and aspect ratios)
  - Multiple output layers at different resolutions is better



Detection examples on COCO test-dev with SSD512 model

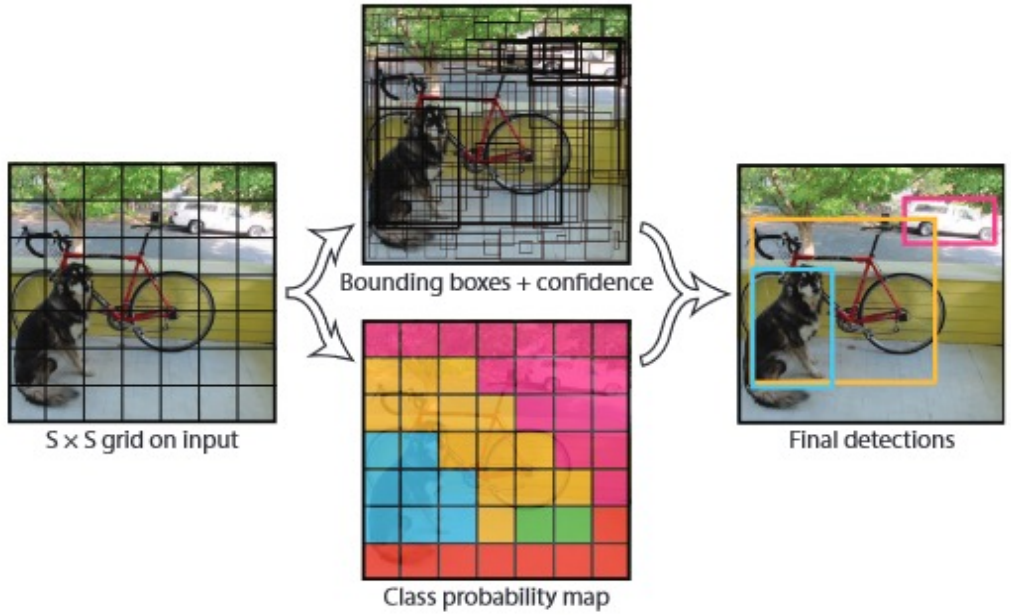
Liu et al. “SSD: Single Shot MultiBox Detector”, ECCV 2016

YOLO: You Only Look Once

- Proposal-based detection repurpose classifiers or localizers to perform detection (apply the model to an image at multiple locations and scales. High scoring regions are considered detections)
- YOLO reframe object detection as a single regression problem

  (Image pixels → Bounding box coordinates and class probabilities
- YOLO: Apply a single neural network to the full image
  - Step 1. Network divides the image into regions and predicts bounding boxes and probabilities for each region
  - Step 2. These bounding boxes are weighted by the predicted probabilities
- Looking image at once during test time makes it very fast

  1000x faster than R-CNN, 100x faster than Fast R-CNN



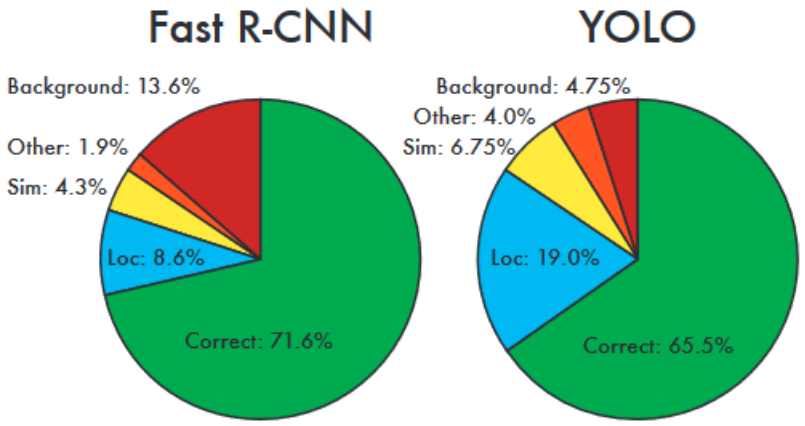
Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016

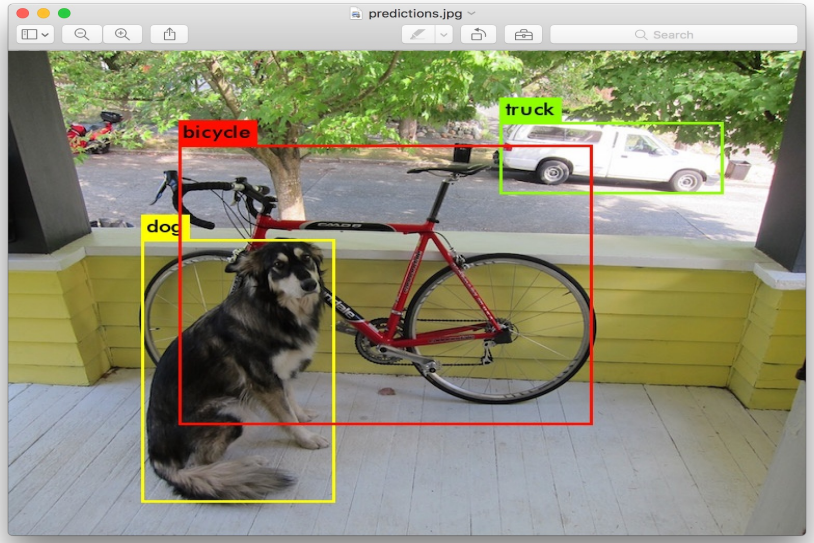
YOLO: You Only Look Once

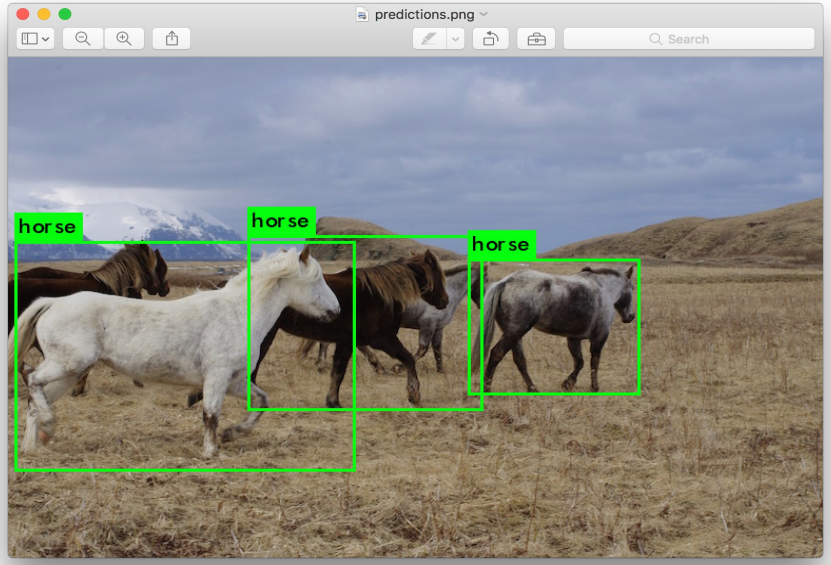
Results on PASCAL VOC 2007

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [37]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Error Analysis

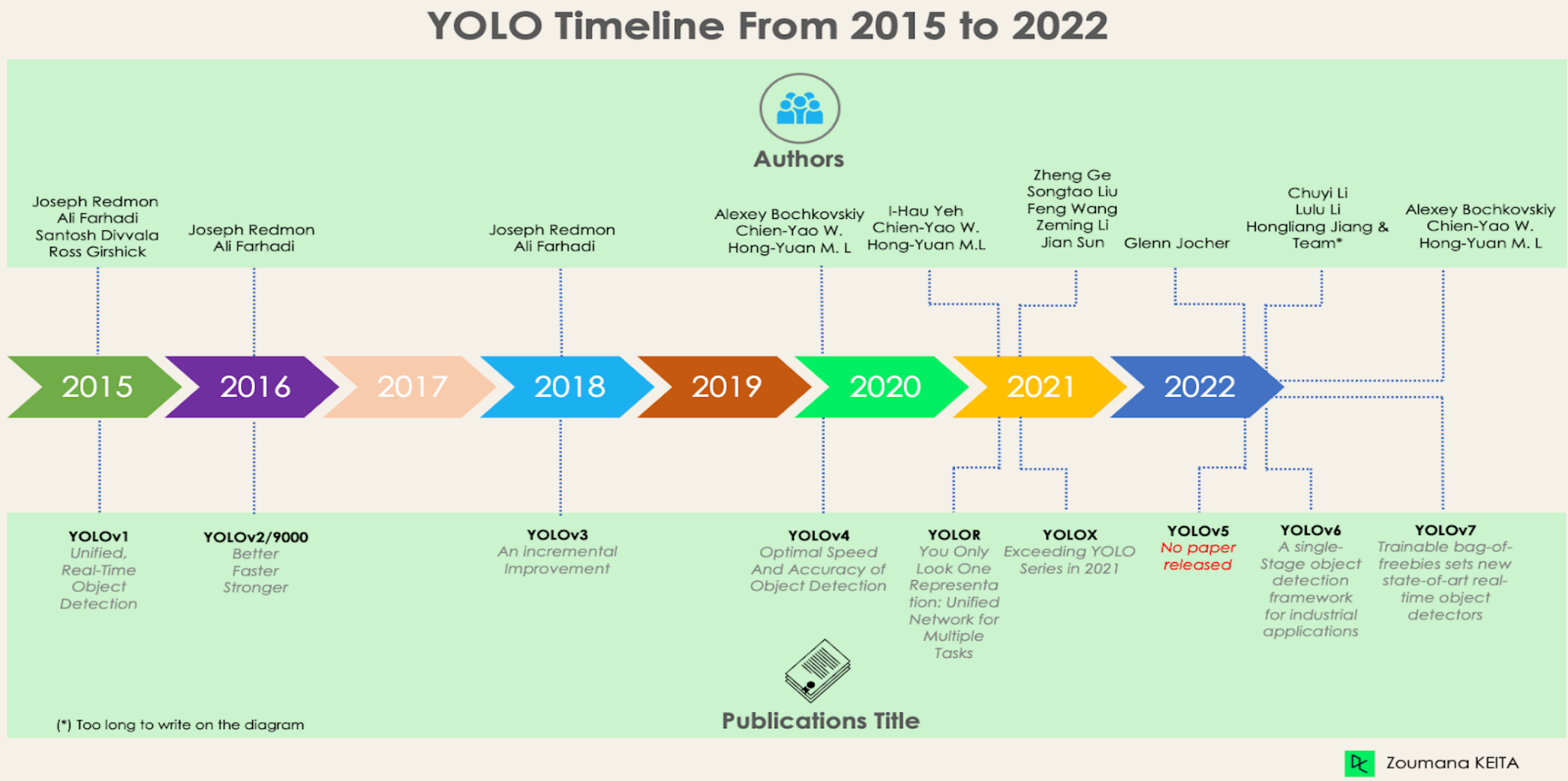






Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”, CVPR 2016

YOLO timeline



Timeline credit: DataCamp https://www.datacamp.com/blog/yolo-object-detection-explained

Why one stage detectors trails accuracy?

- Extreme foreground-background class imbalance encountered.

Two-stage:

The proposals stage rapidly narrows down # candidate object locations to a small number (e.g., 2k proposals), filtering out most background samples.

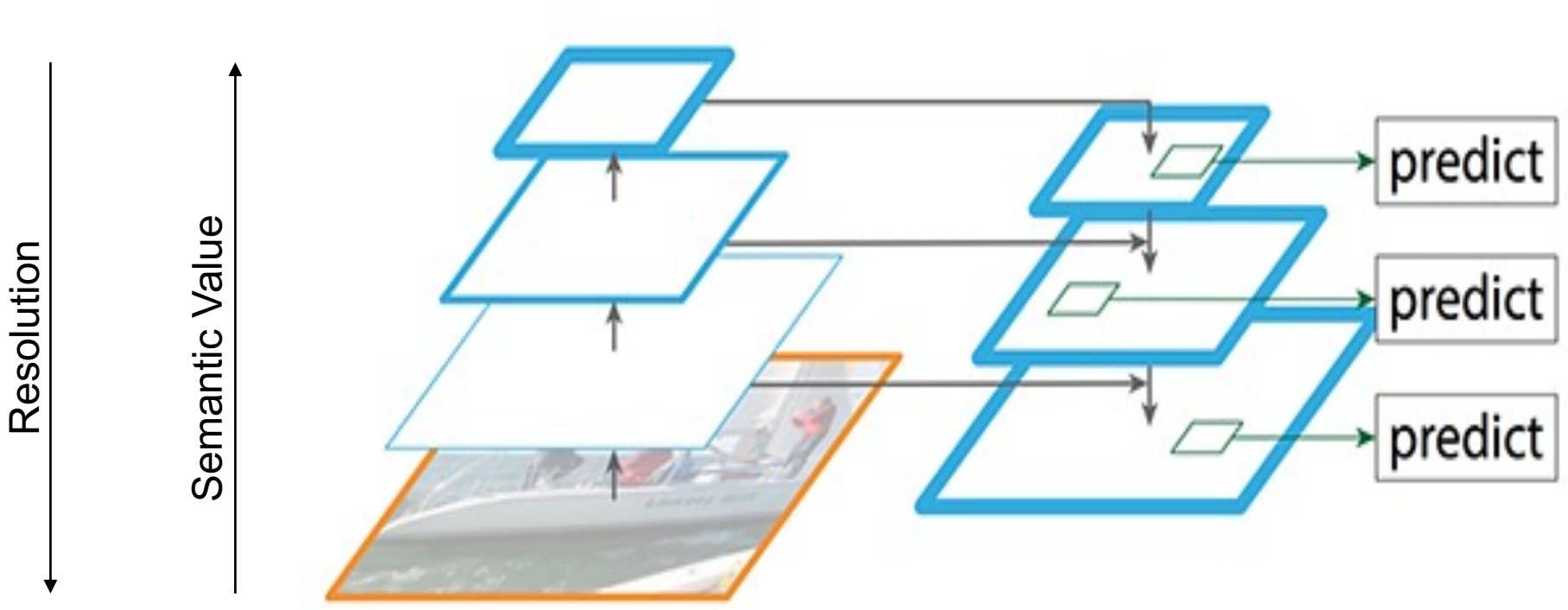
In the classification stage, fix foreground-to-background ratio to 1:3, or online hard example mining (OHEM).

One-stage:

Have to process a much larger set of candidate object locations regularly sampled across an image, which amounts to enumerating ~100k locations that densely cover spatial positions, scales, and aspect ratios.

Activation Maps

- Extreme foreground-background class imbalance encountered.

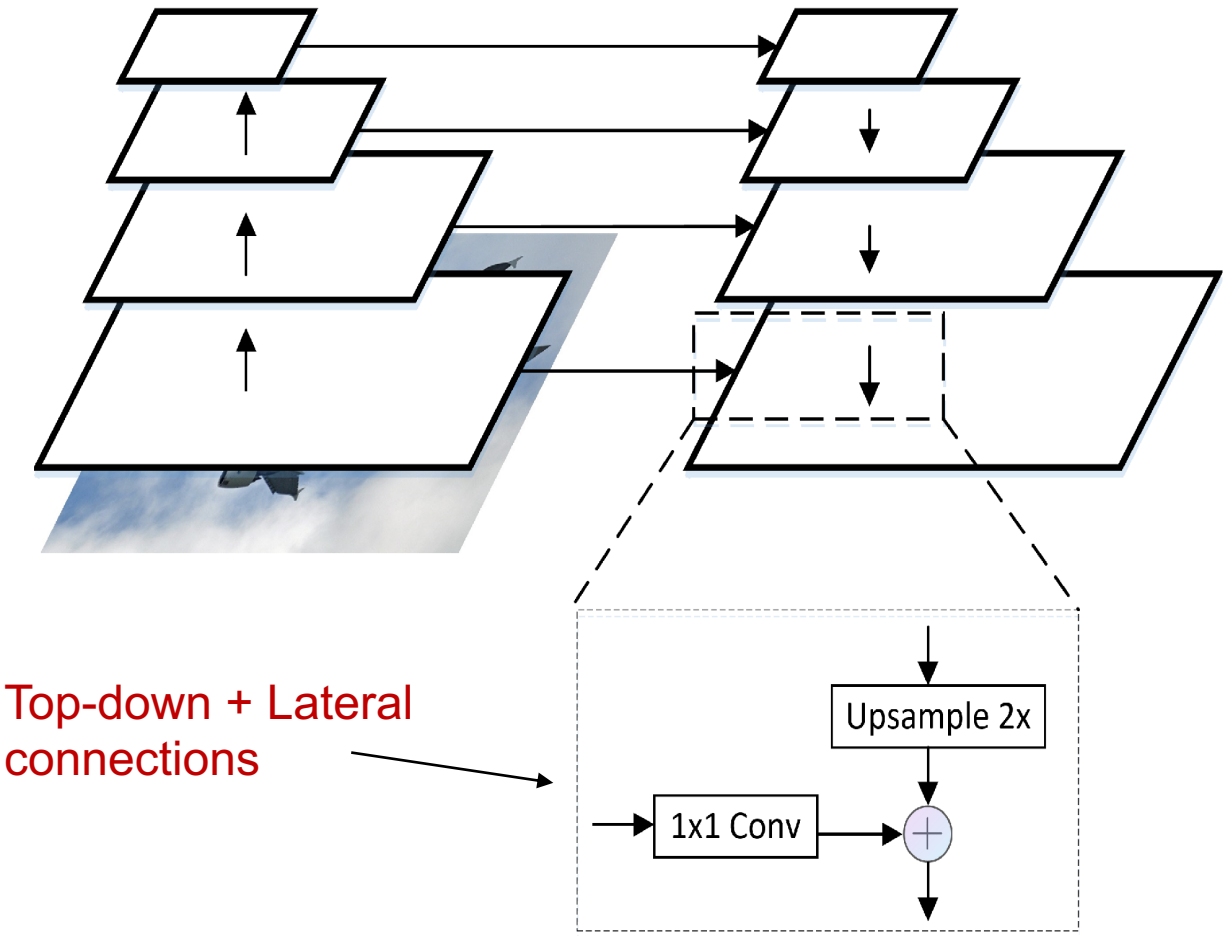


As image goes deeper through the network, resolution decreases and semantic value increases.

How about predicting from multiple maps?

Feature Pyramid Network (FPN)

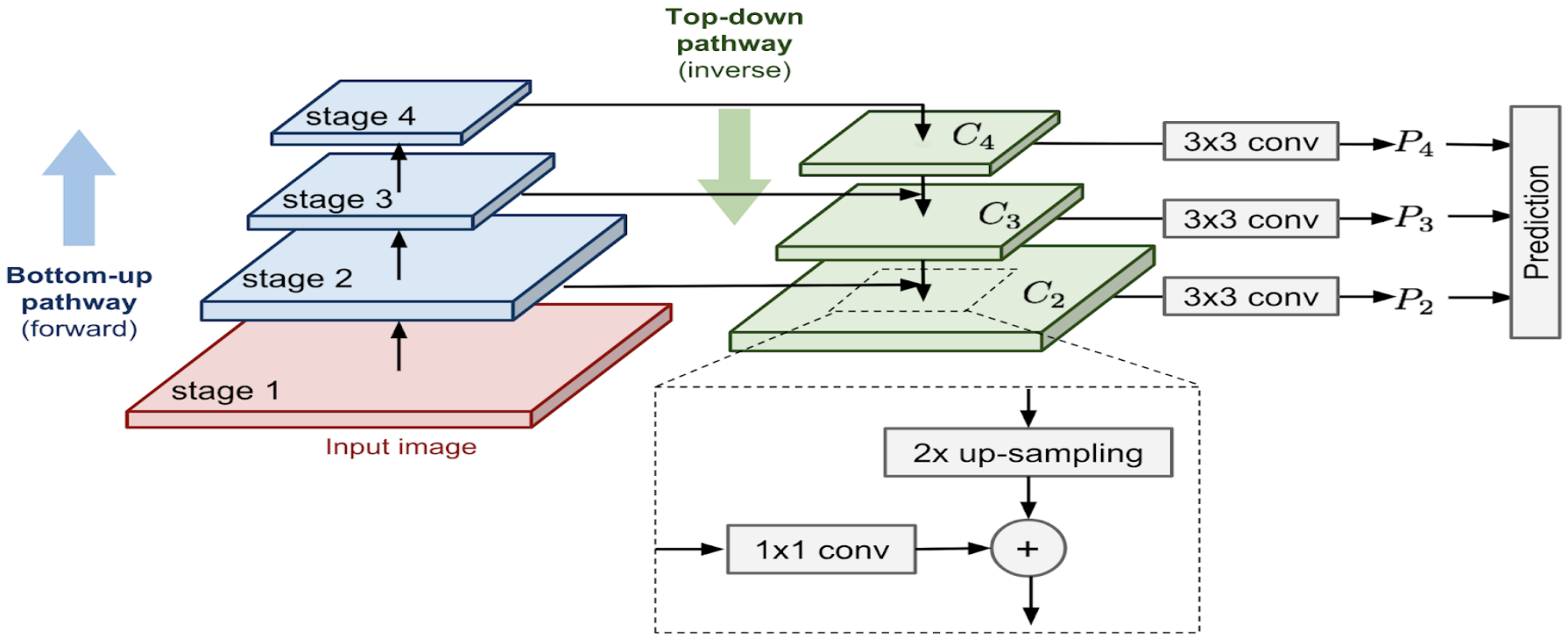
- Improve predictive power of lower-level feature maps by adding contextual information from higher-level feature maps.
- FPN creates an architecture with rich semantics at all levels by combining low-resolution semantically strong features with high-resolution semantically weak features.



Lin et al., “Feature Pyramid Networks for Object Detection”, CVPR 2017.

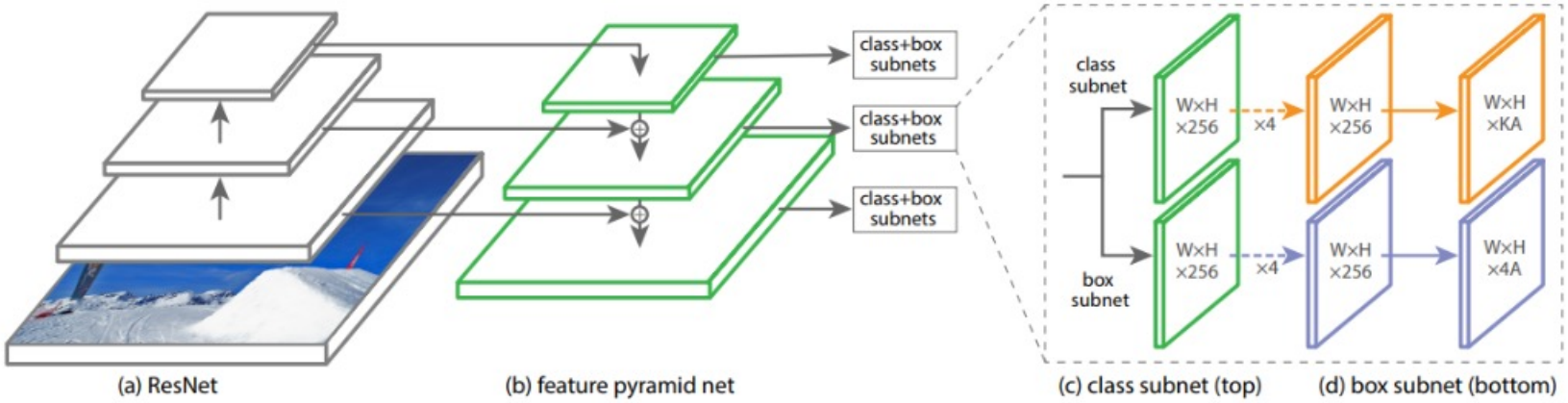
RetinaNet

- One-stage object detectors proven to work well with dense and small-scale objects
- RetinaNet: Feature Pyramid Network + Focal Loss (avoiding class-imbalance)



Lin et al., “Focal Loss for Dense Object Detection”, ICCV 2017.

RetinaNet Architecture



Bottom-up Pathway: The backbone network which calculates the feature maps at different sales

Top-down pathway and Lateral connections: Upsamples the spatially coarser feature maps from higher pyramid levels, and the lateral connections merge the top-down layers and the bottom-up layers with the same spatial size.

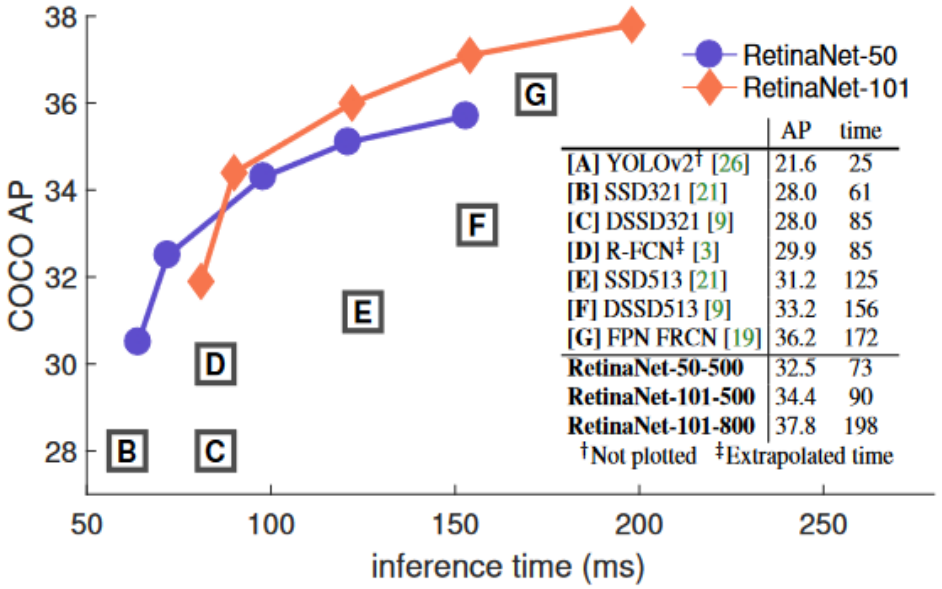
Classification subnetwork: It predicts the probability of an object being present at each spatial location for each anchor box and object class.

Bbox subnetwork: It's regresses the offset for the bounding boxes from the anchor boxes for each ground-truth object.

Lin et al., “Focal Loss for Dense Object Detection”, ICCV 2017.

RetinaNet Results

Speed (ms) versus accuracy (AP) on COCO test-dev.

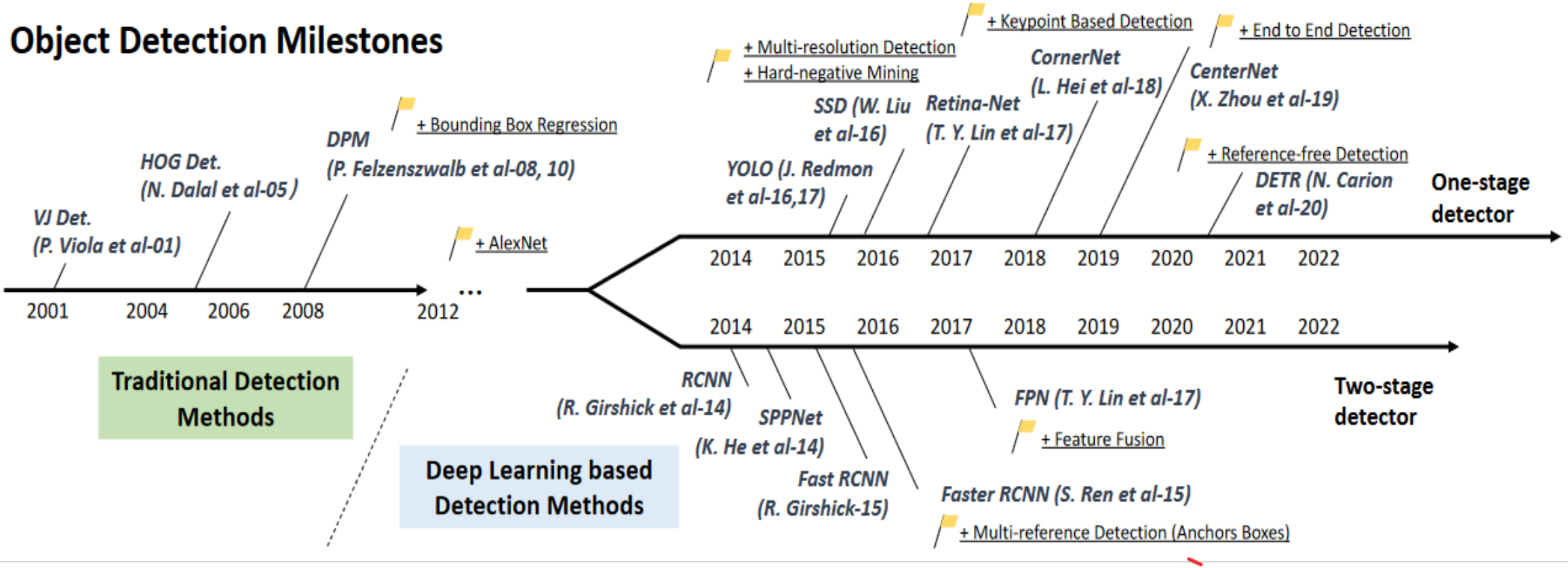


	AP	time
[A] YOLOv2† [26]	21.6	25
[B] SSD321 [21]	28.0	61
[C] DSSD321 [9]	28.0	85
[D] R-FCN‡ [3]	29.9	85
[E] SSD513 [21]	31.2	125
[F] DSSD513 [9]	33.2	156
[G] FPN FRCN [19]	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	37.8	198

†Not plotted ‡Extrapolated time

Lin et al., “Focal Loss for Dense Object Detection”, ICCV 2017.

Object Detection in 20 Years



Zou et al. “Object Detection in 20 Years: A Survey”, ArXiv, 2023

Implementations

[1]. PyTorch

https://pytorch.org/vision/main/models/faster_rcnn.html

[2]. TensorFlow

https://www.tensorflow.org/hub/tutorials/object_detection

[3]. Detectron2 (Meta AI's state-of-the-art object detection and segmentation library)

https://github.com/facebookresearch/detectron2

[4]. MMDetection (Open-source object detection toolbox based on PyTorch)

https://github.com/open-mmlab/mmdetection

[5]. Understanding and Implementing Faster R-CNN: A step-by-step guide

https://towardsdatascience.com/understanding-and-implementing-faster-r-cnn-a-step-by-step-guide-11acfff216b0

[6]. Faster R-CNN in PyTorch and TensorFlow with Keras

https://github.com/trzy/FasterRCNN

Implementations

[7]. Mask Detection using Faster R-CNN

https://pseudo-lab.github.io/Tutorial-Book-en/chapters/en/object-detection/Ch5-Faster-R-CNN.html

[8]. Object Detection using Faster R-CNN

https://github.com/spmallick/learnopencv/blob/master/PyTorch-faster-RCNN/PyTorch_faster_RCNN.ipynb

[9]. SSD implementation in PyTorch

https://pytorch.org/hub/nvidia_deeplearningexamples_ssd/

[10]. YOLO implementation in PyTorch

https://pytorch.org/hub/ultralytics_yolov5/

https://github.com/ultralytics/yolov5

[11]. Mask Detection using RetinaNet

https://pseudo-lab.github.io/Tutorial-Book-en/chapters/en/object-detection/Ch4-RetinaNet.html

[12]. Faster R-CNN on PASCAL VOC Dataset

https://cv.gluon.ai/build/examples_detection/train_faster_rcnn_voc.html

Further reading on discussed topics

- Chapter 7 of Deep Learning Book by Ian Goodfellow, Yoshua Bengio and Aaron Courville. https://www.deeplearningbook.org/
- Chapter 4: Object Detection and Image Segmentation from Practical Machine Learning for Computer Vision by Valliappa Lakshmanan, Martin Gorner, Ryan Gillard. https://www.oreilly.com/library/view/practical-machine-learning/9781098102357/ch04.html
- Chapter 7 of Deep Learning for Vision Systems by Mohamed Elgendy.

Acknowledgements

- Some material drawn from referenced and associated online sources
- Image sources credited where possible
- Some slides adapted from cs231n Lecture 9 “Object Detection and Image Segmentation”

References

[1]. Felzenszwalb and Huttenlocher., “Efficient Graph-Based Image Segmentation”, IJCV 2004.

https://link.springer.com/article/10.1023/B:VISI.0000022288.19776.77

[2]. Van de Sande et al., Segmentation as Selective Search for Object Recognition., ICCV 2011.

https://ieeexplore.ieee.org/document/6126456

[3]. Girshick et al., Rich feature hierarchies for accurate object detection and semantic segmentation, CVPR 2014.

https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Girshick_Rich_Feature_Hierarchies_2014_CVPR_paper.pdf

[4]. Girshick., Fast R-CNN, ICCV 2015.

https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Girshick_Fast_R-CNN_ICCV_2015_paper.pdf

[5]. Ren et al., “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NeurIPS 2015.

https://papers.nips.cc/paper_files/paper/2015/hash/14bfa6bb14875e45bba028a21ed38046-Abstract.html

[6]. Liu et al. “SSD: Single Shot MultiBox Detector”, ECCV 2016.

https://link.springer.com/chapter/10.1007/978-3-319-46448-0_2

[7]. Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”, CVPR 2016.

https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf

References

[8]. Zou et al. “Object Detection in 20 Years: A Survey”, ArXiv, 2023.

https://arxiv.org/abs/1905.05055

[9]. Long et al. “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015.

https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Long_Fully_Convolutional_Networks_2015_CVPR_paper.pdf

[10]. Ronneberger et al. (2015). U-net: Convolutional Networks for Biomedical Image Segmentation. MICCAI 2015.

https://link.springer.com/chapter/10.1007/978-3-319-24574-4_28

[11]. Oktay et al., (2018). “Attention U-Net: Learning where to look for the Pancreas”, MIDL 2018.

https://openreview.net/forum?id=Skft7cijM

[12]. Diakogiannis et al., (2019). “ResUNet-a: A deep learning framework for semantic segmentation of remotely sensed data”, ISPRS Journal of Photogrammetery and Remote Sensing.

https://www.sciencedirect.com/science/article/abs/pii/S0924271620300149

[13]. Chen et al., (2021). “TransUNet: Transformers Make Strong Encoders for Medical Image Segmentation”, ArXiv.

https://arxiv.org/abs/2102.04306

[14]. He et al., “Mask R-CNN”. ICCV 2017.

https://openaccess.thecvf.com/content_ICCV_2017/papers/He_Mask_R-CNN_ICCV_2017_paper.pdf