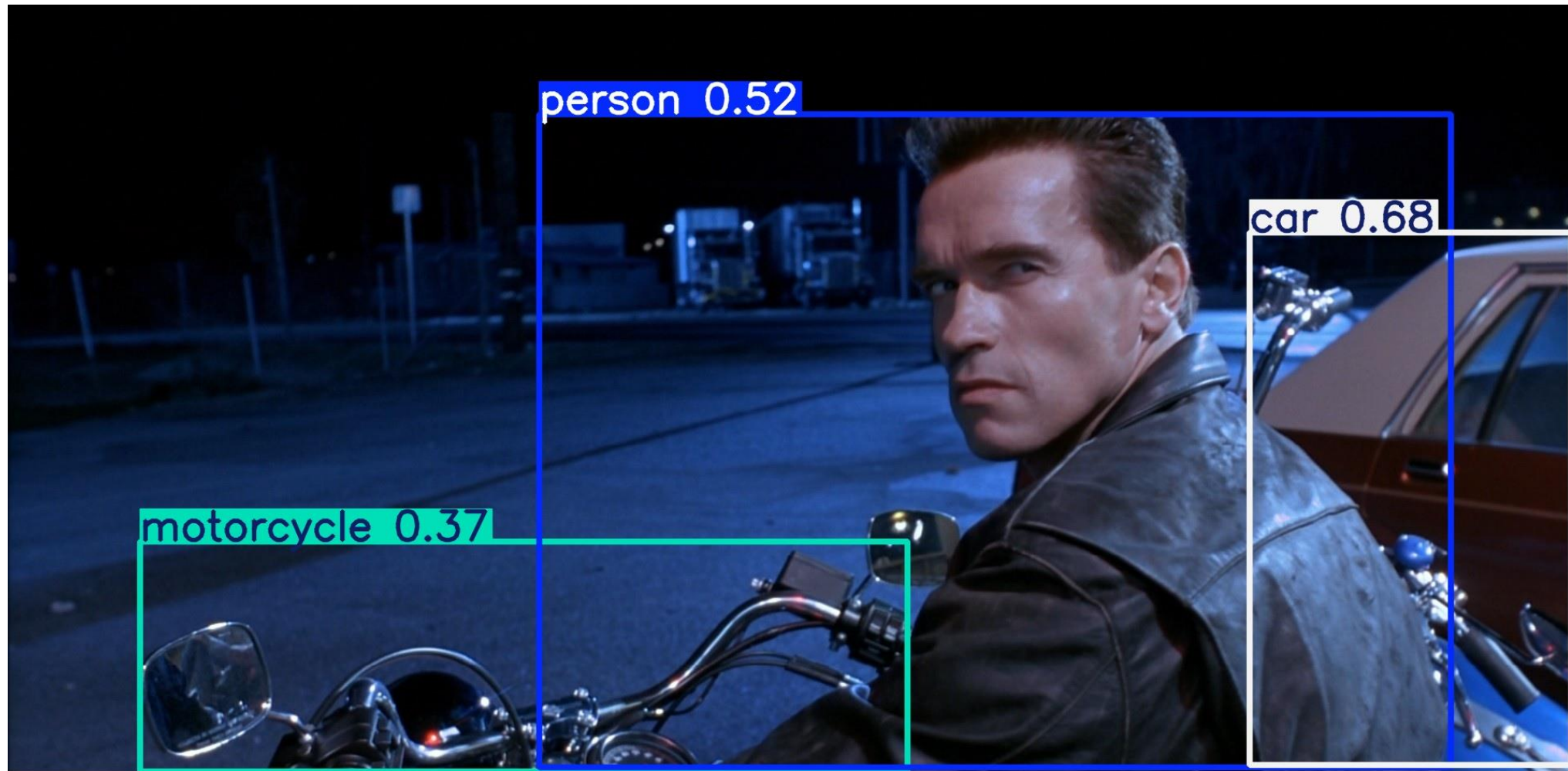


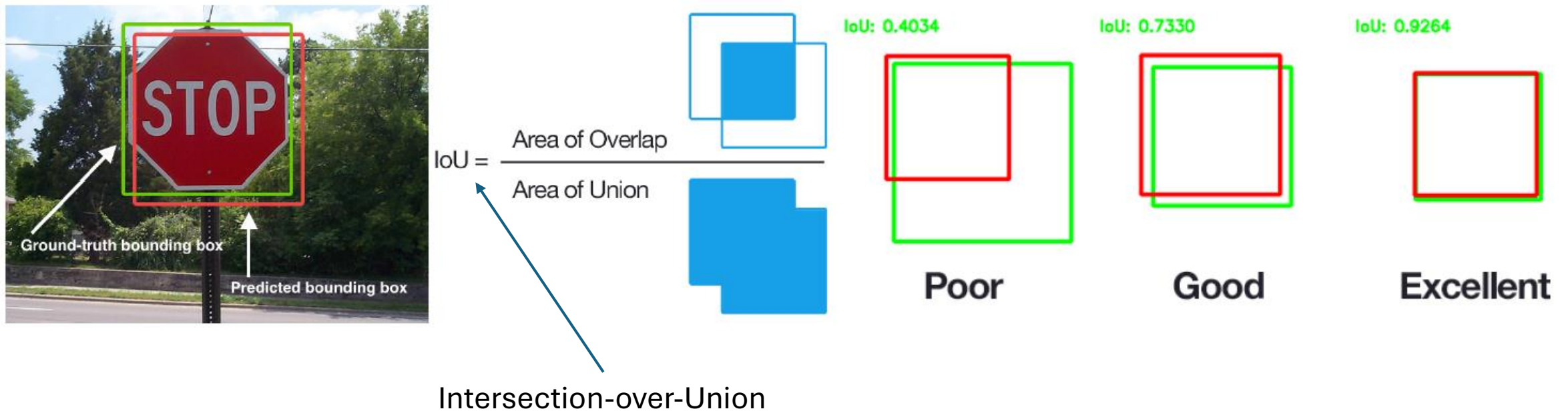
# Object Detection

Deep Learning and Image Processing

output: bounding boxes with category label and confidence



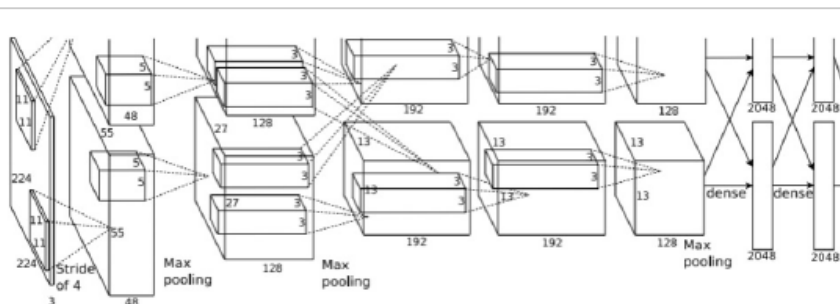
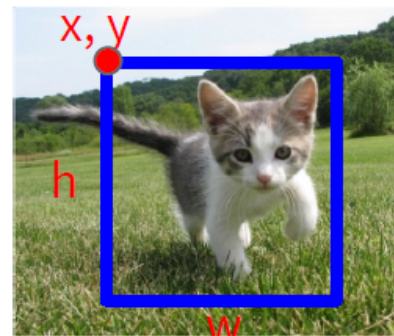
# Performance Evaluation of Localization



images with multiple objects: number of detections dependent on used confidence threshold

# Object Detection: Single Object

(Classification + Localization)



Fully  
Connected:  
4096 to 1000

Class Scores

Cat: 0.9  
Dog: 0.05  
Car: 0.01  
...

Correct label:  
Cat

Softmax  
Loss

Multitask Loss

Vector:  
4096

Fully  
Connected:  
4096 to 4

Box  
Coordinates  
(x, y, w, h)

L2 Loss

Correct box:  
(x', y', w', h')

Treat localization as a  
regression problem!

too complicated for multiple objects



# Localization for Multiple-Object Images

idea: classify many different crops of the image as object or background  
(crops: sliding window over image, iterated at multiple window sizes)



Dog? No  
Cat? No  
Background? Yes



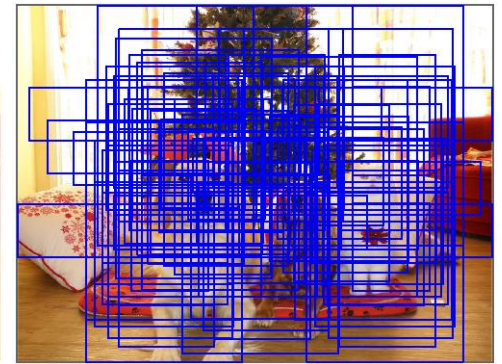
Dog? Yes  
Cat? No  
Background? No



Dog? Yes  
Cat? No  
Background? No



Dog? No  
Cat? Yes  
Background? No



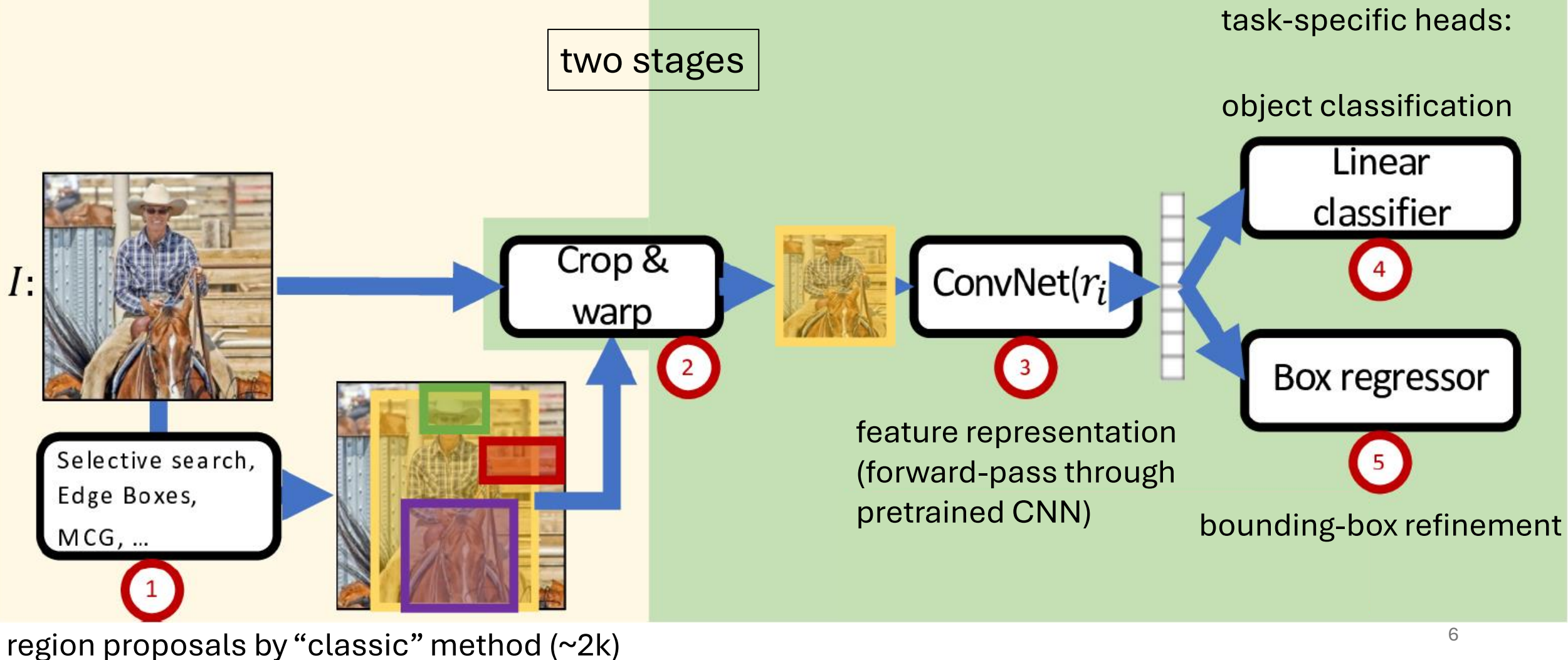
issue: too many  
possible crops

→ need for region proposals

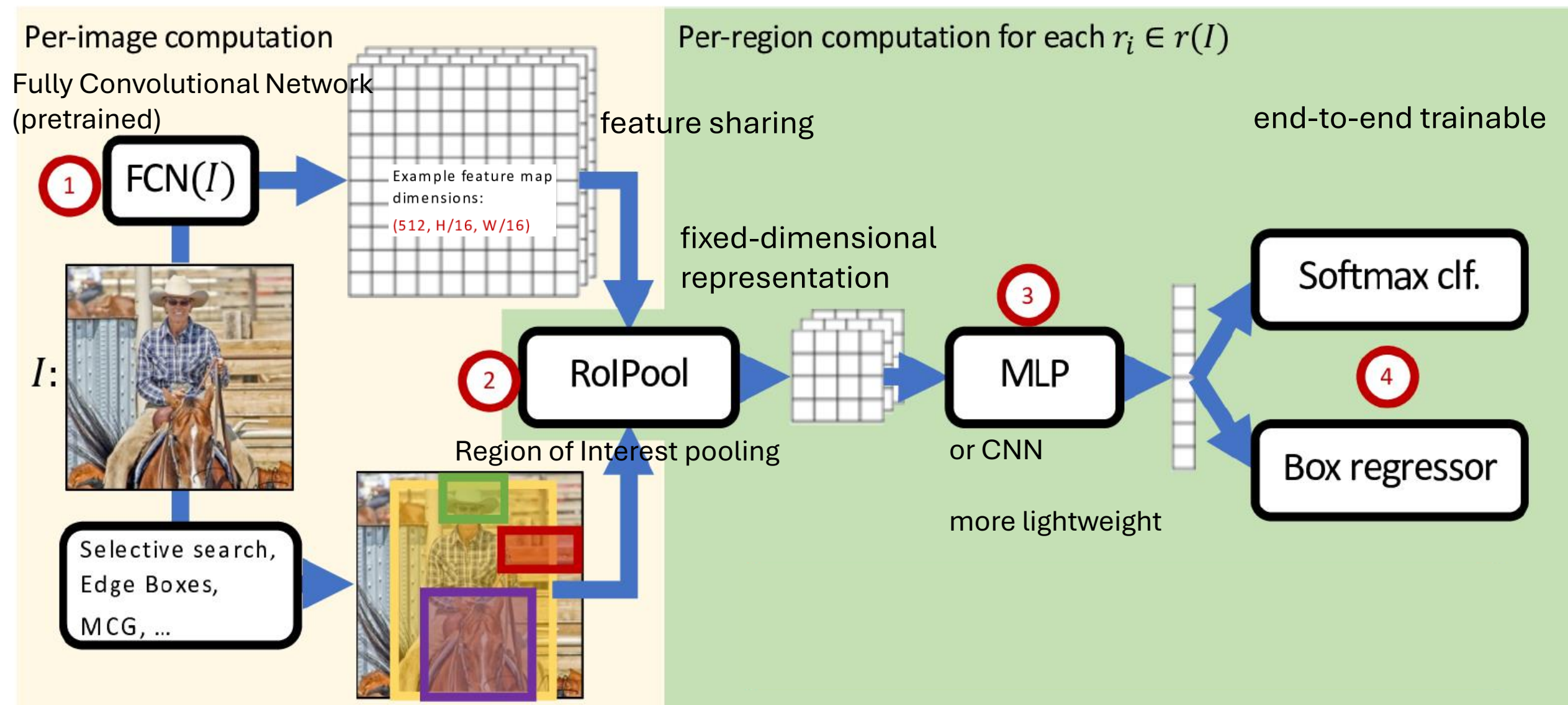
# Region-Based Convolutional Neural Network (R-CNN)

Per-image computation

Per-region computation for each  $r_i \in r(I)$

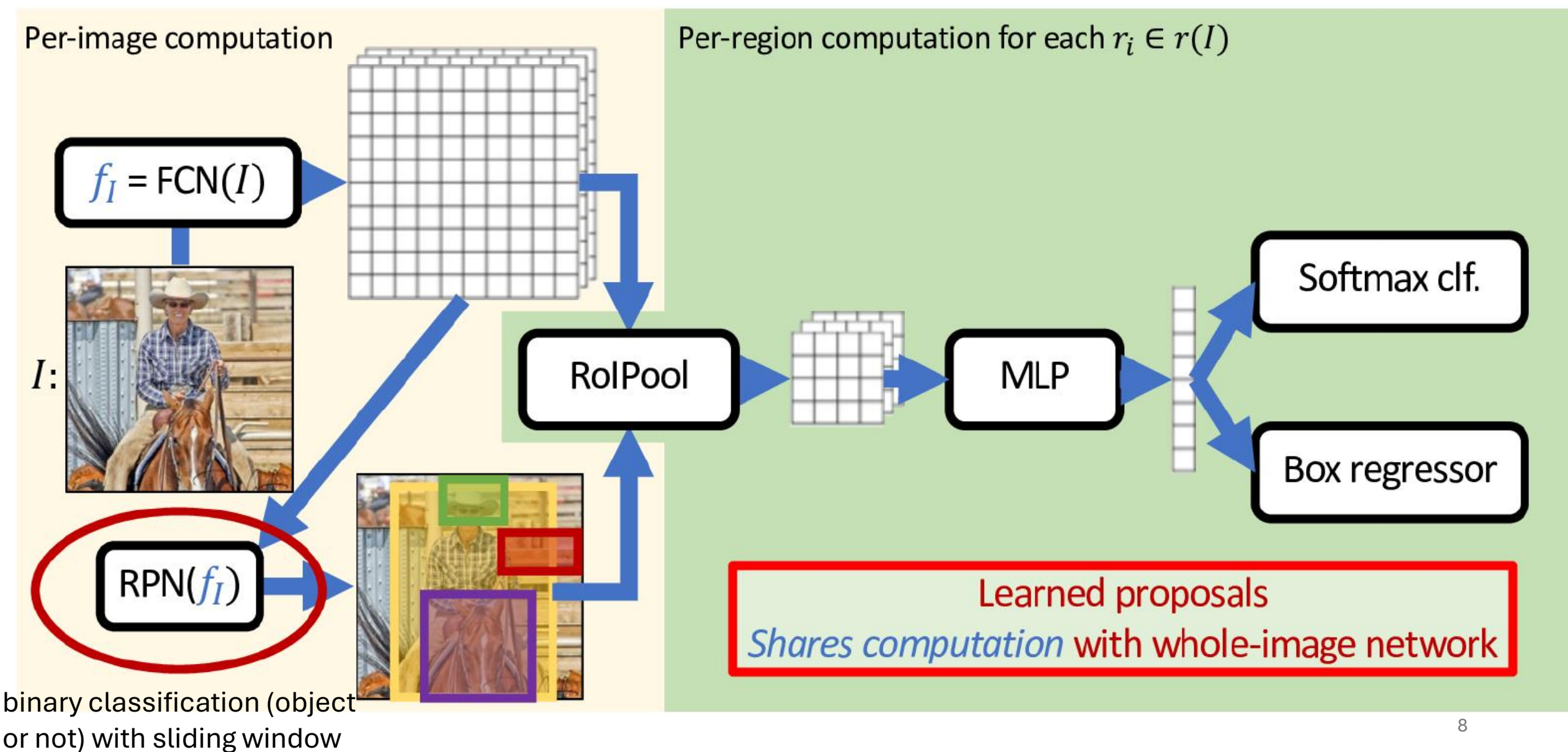


# Fast R-CNN: Crop ConvNet Features





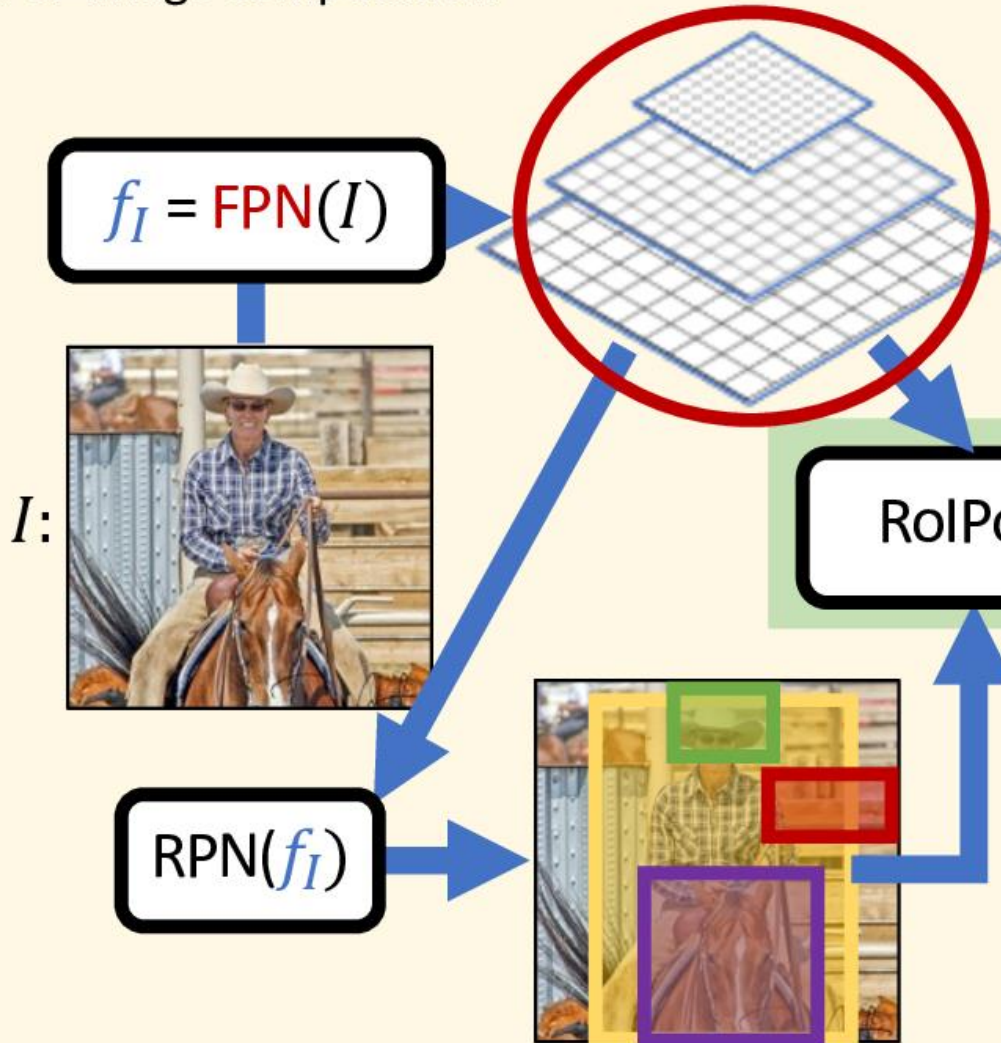
# Faster R-CNN: Use Region Proposal Network



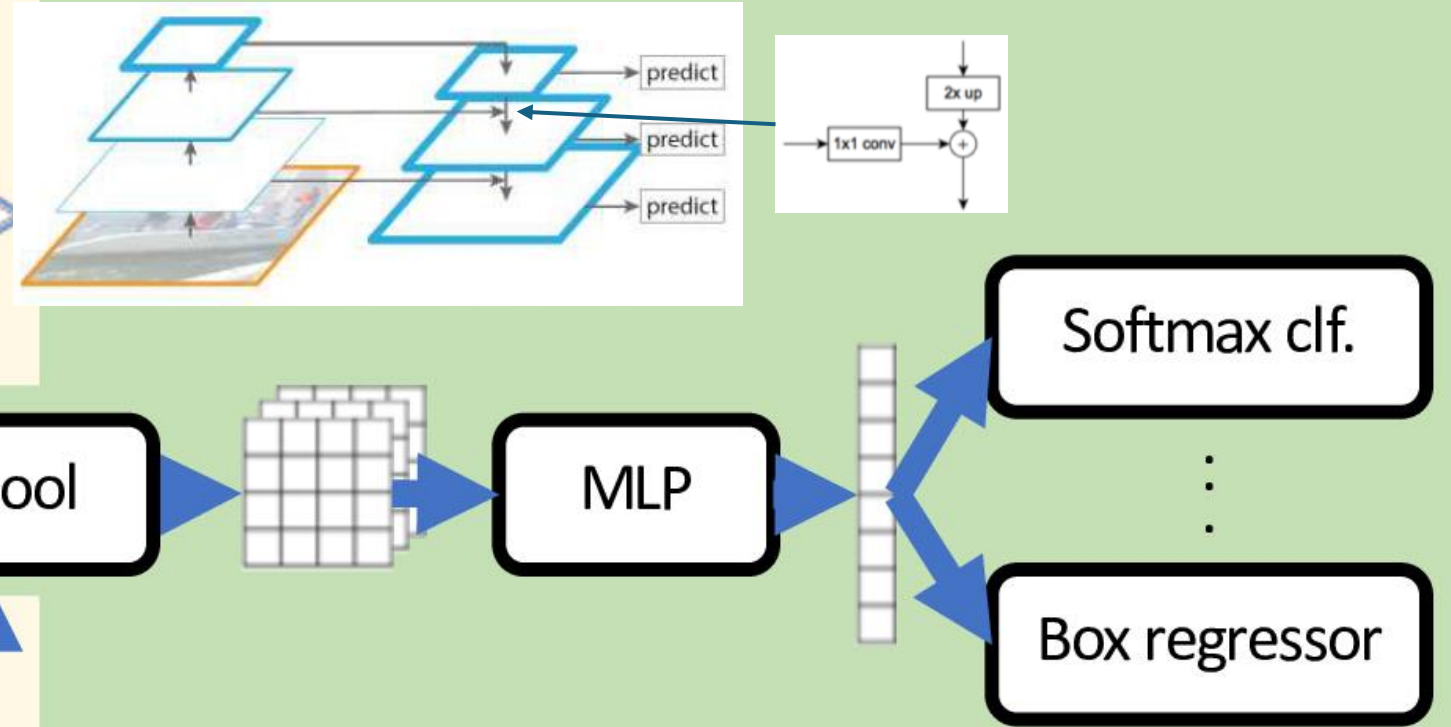


# Feature Pyramid Network (FPN)

Per-image computation



Per-region computation for each  $r_i \in r(I)$

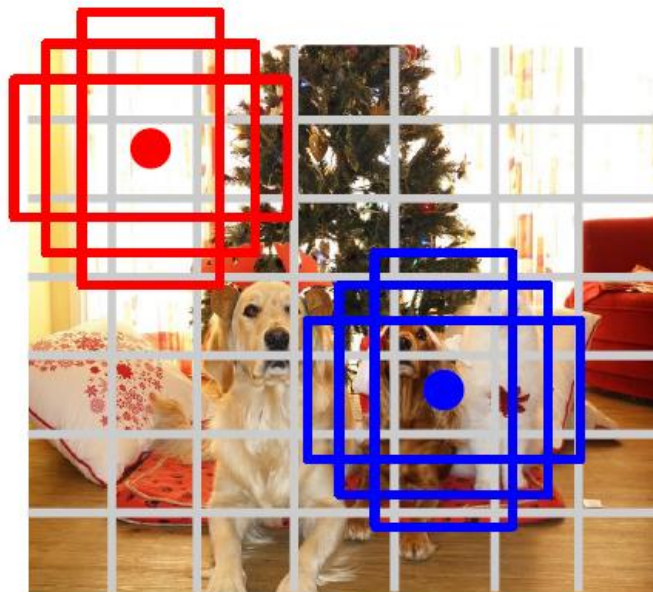


The whole-image feature representation can be improved by making it *multi-scale*

# Single-Stage Detectors: Drop Per-Region Computation



Input image  
 $3 \times H \times W$



Divide image into grid  
 $7 \times 7$

Image a set of base boxes  
centered at each grid cell  
Here  $B = 3$

Within each grid cell:

- Regress from each of the  $B$  base boxes to a final box with 5 numbers:  
( $dx, dy, dh, dw, confidence$ )
- Predict scores for each of  $C$  classes (including background as a class)
- Looks a lot like RPN, but category-specific!

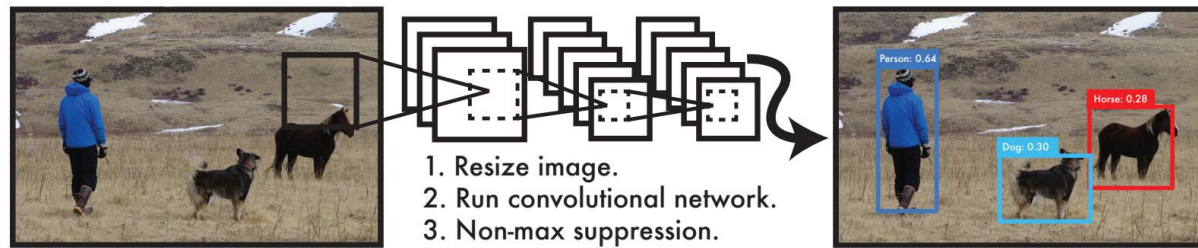
Output:  
 $7 \times 7 \times (5 * B + C)$

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

faster, but usually worse performance



# YOLO: Real-Time Object Detection



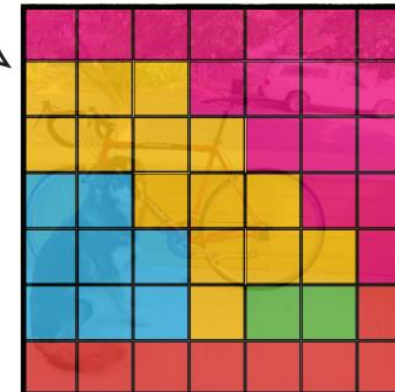
You Only Look Once:  
prediction of bounding boxes  
and class probabilities in one go



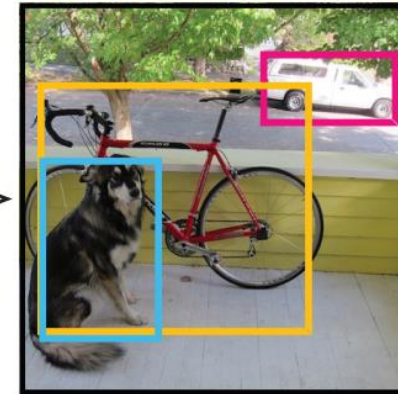
$S \times S$  grid on input



Bounding boxes + confidence



Class probability map



Final detections

# Object Tracking

task: locating an object in successive video frames

basic approach:

1. detection: localize target object(s) (e.g., YOLO)
2. motion estimation: predict next location (e.g., Kalman filter or optical flow)
3. appearance matching: comparison of previous and predicted next location (e.g., feature descriptor, CNN features)

detection repeated frequently to correct for accumulated deviations



# Optical Flow

dense optical flow:

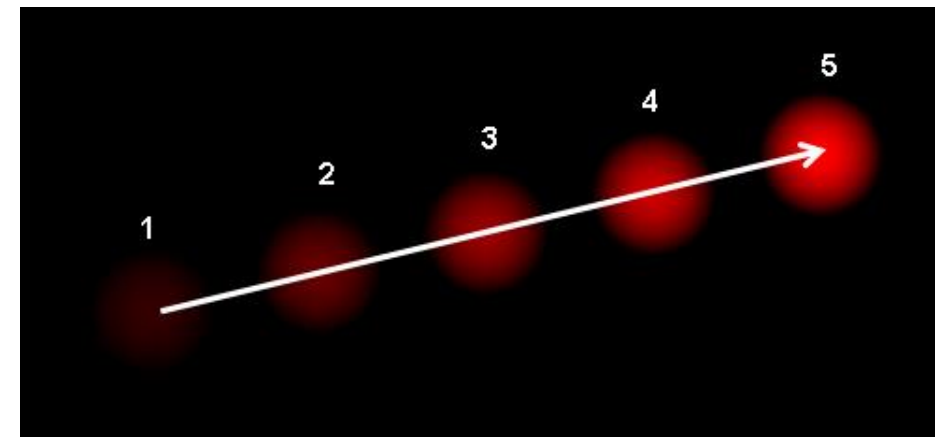
estimate motion vector of every pixel

e.g., Horn-Schunck method or Lucas-Kanade (LK) method

sparse optical flow:

estimate motion vector of few image features

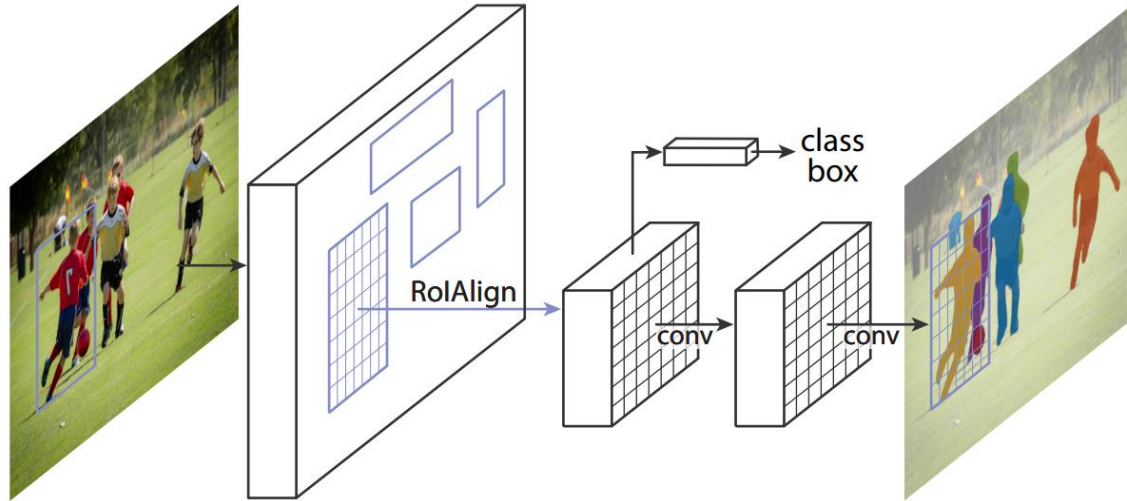
e.g., Kanade-Lucas-Tomasi (KLT) feature tracker



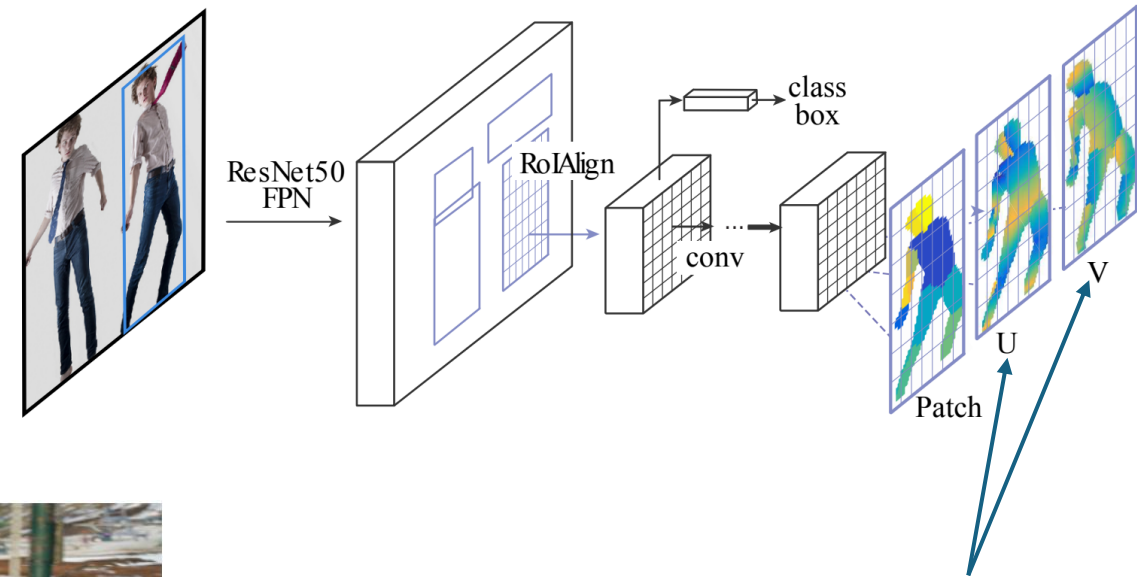
# Instance Segmentation

# Add Additional Network Heads to R-CNN

instance segmentation ([Mask R-CNN](#)):



pose estimation ([DensePose](#)):



3D model's surface to 2D



# Mask R-CNN

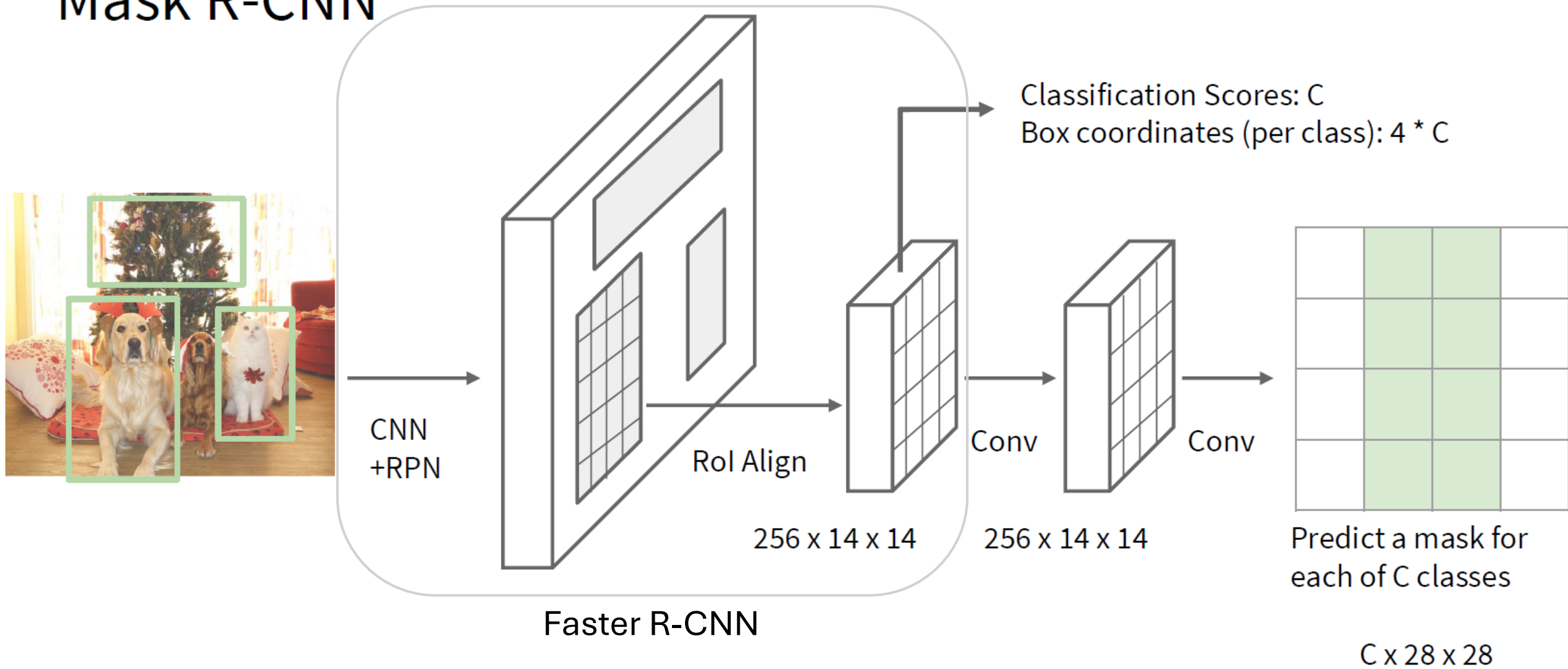




image with training proposal

28 × 28 mask target

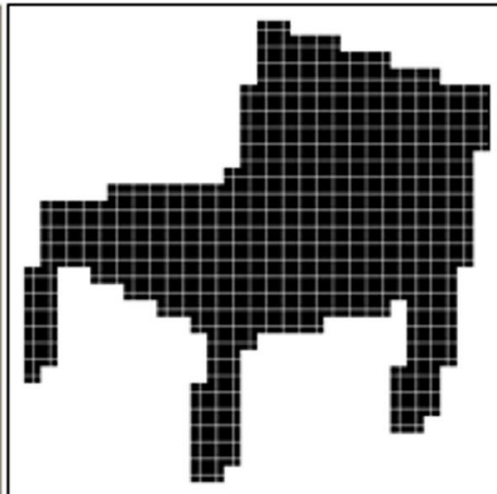
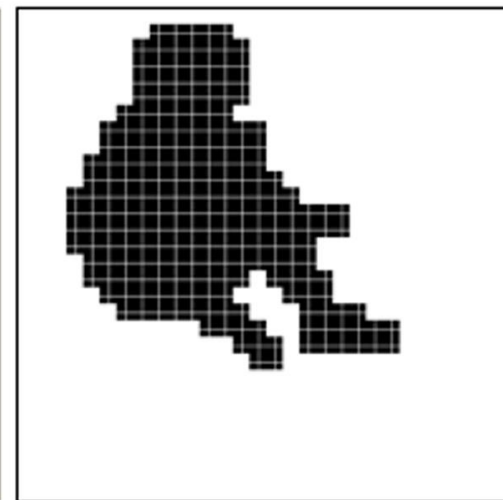
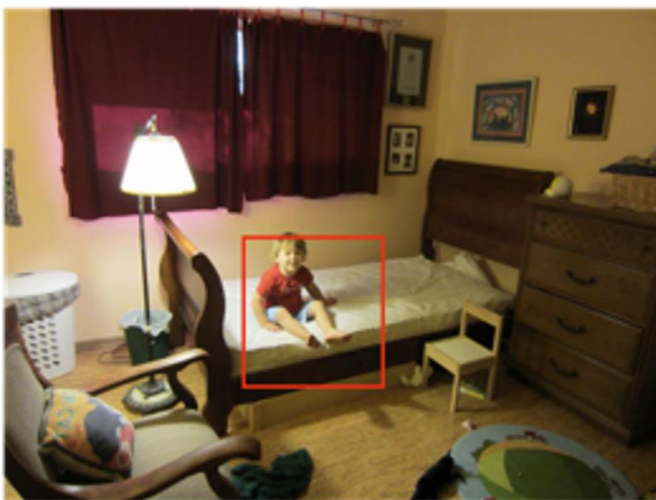
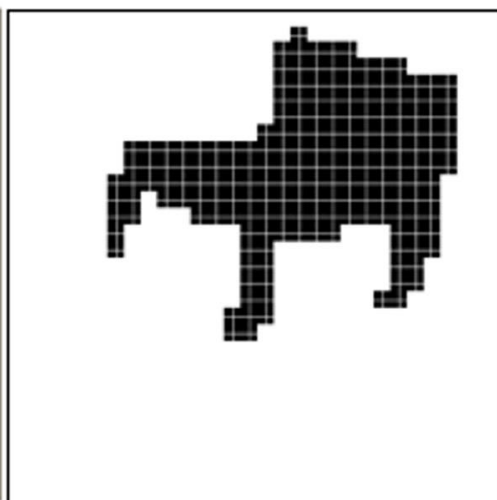
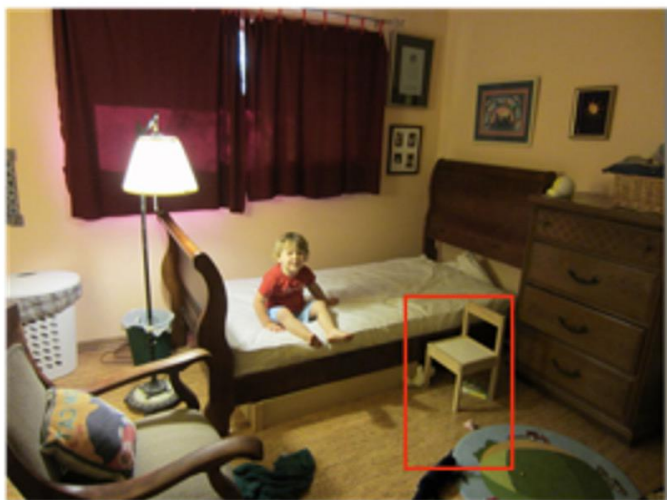
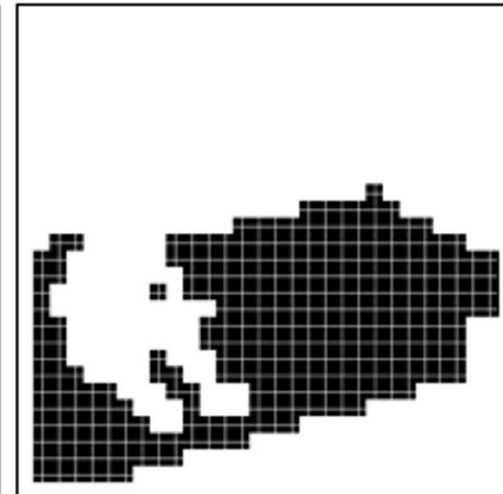
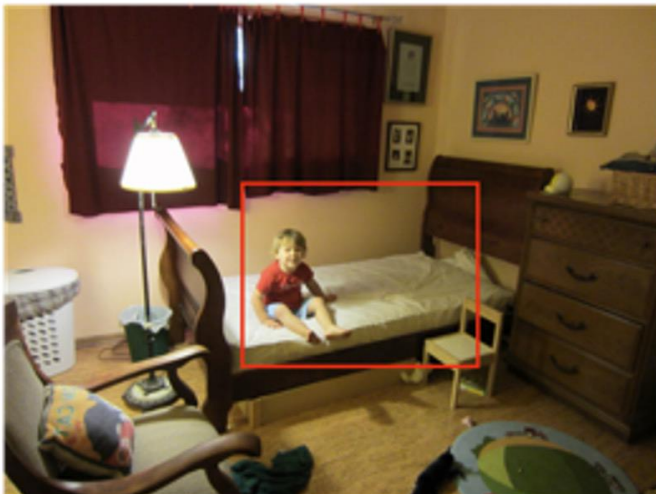


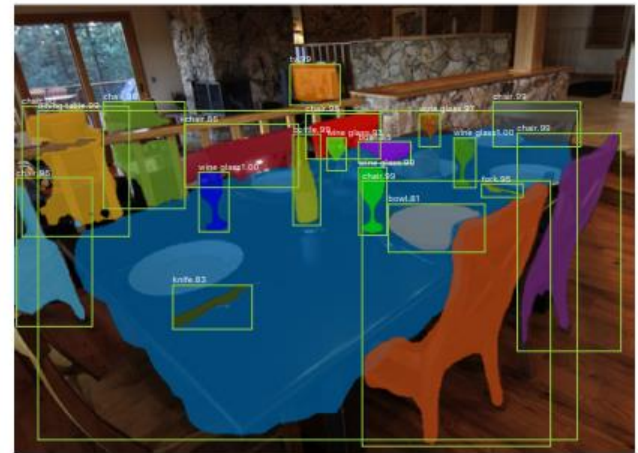
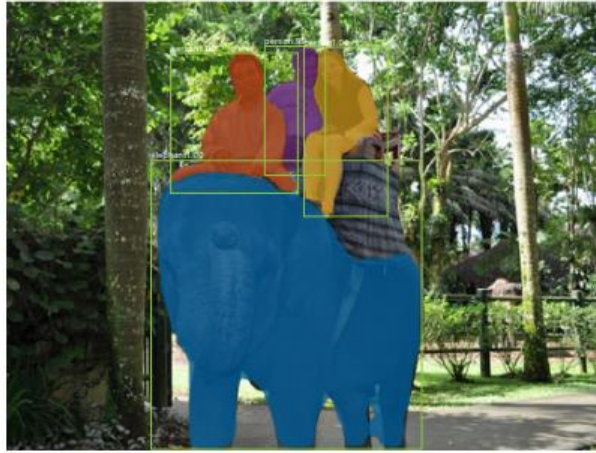
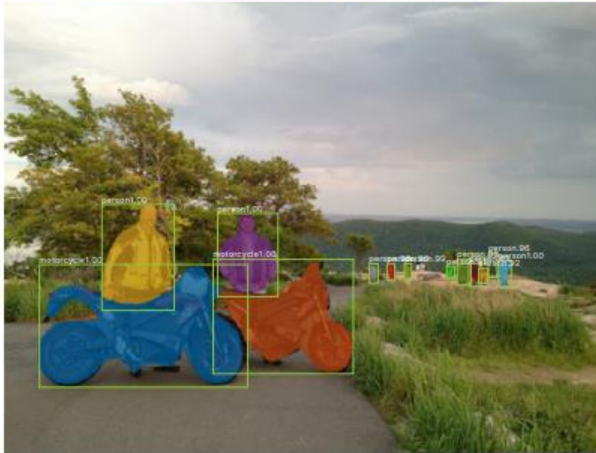
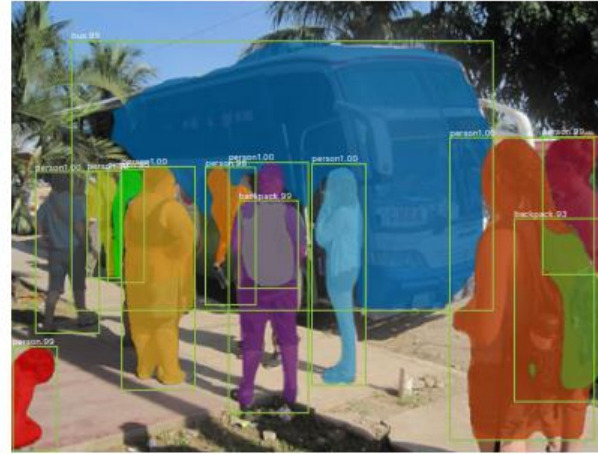
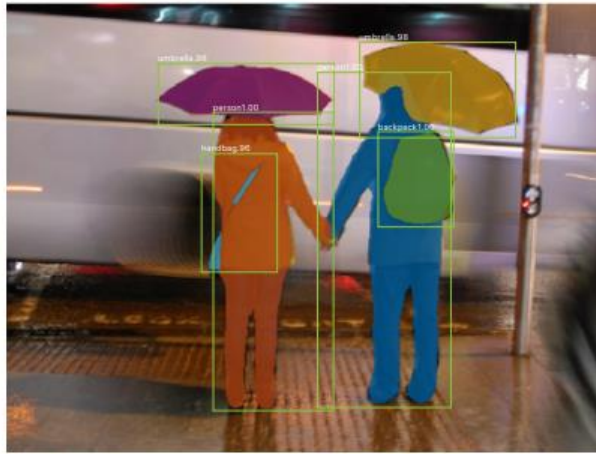
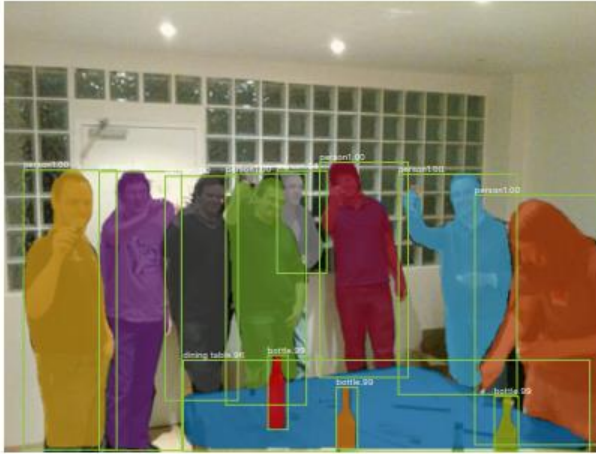
image with training proposal

28 × 28 mask target



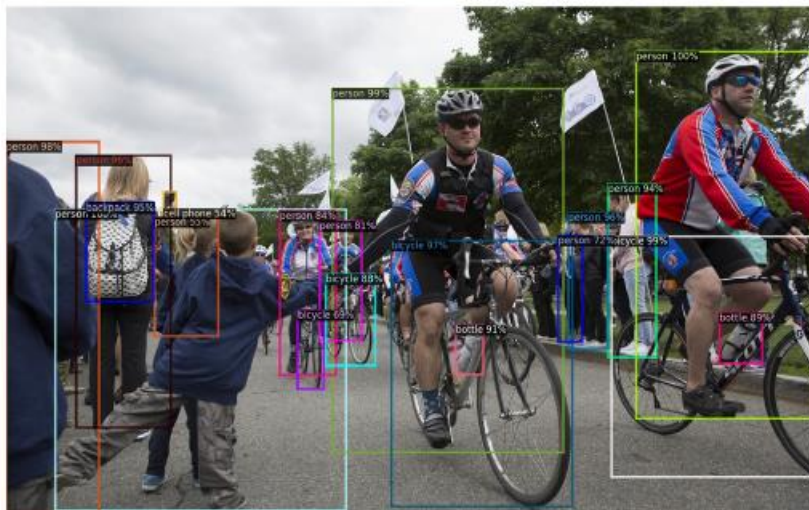


results on MS COCO data set





# Detectron2: PyTorch Implementations



# Promptable Segmentation with Transformers

Segment Anything Model ([SAM](#))

[SAM2](#) includes also video segmentation

