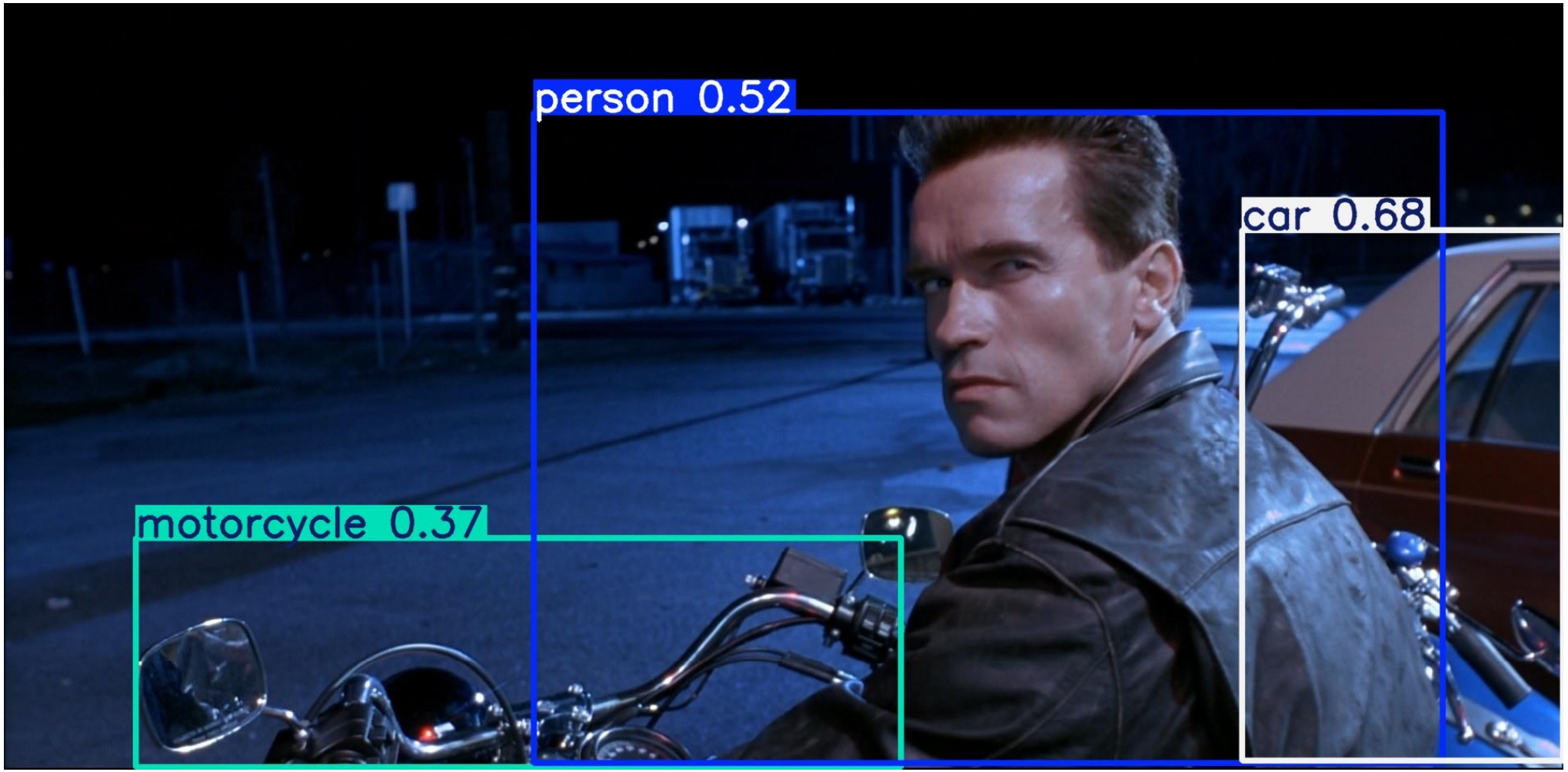
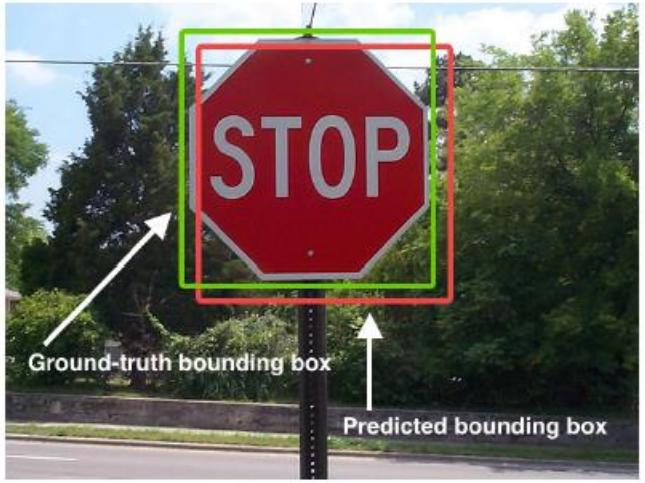
Object Detection

Deep Learning and Image Processing

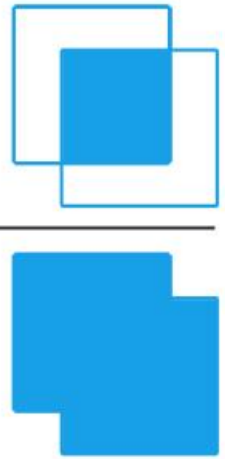
output: bounding boxes with category label and confidence



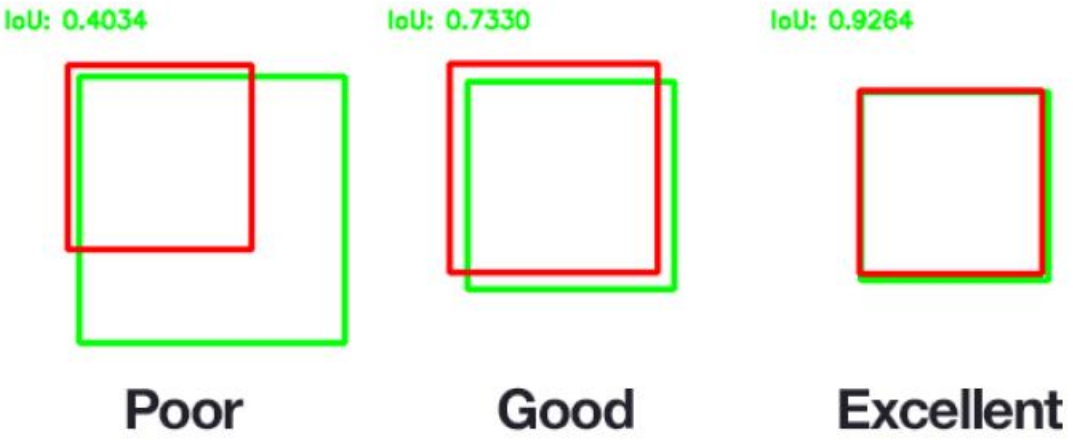
Performance Evaluation of Localization



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



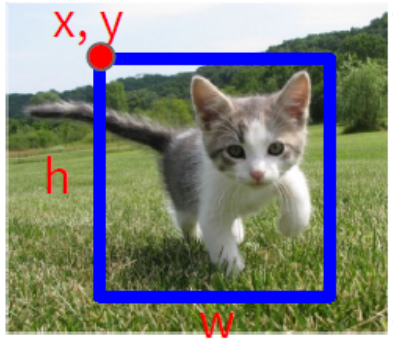
Intersection-over-Union

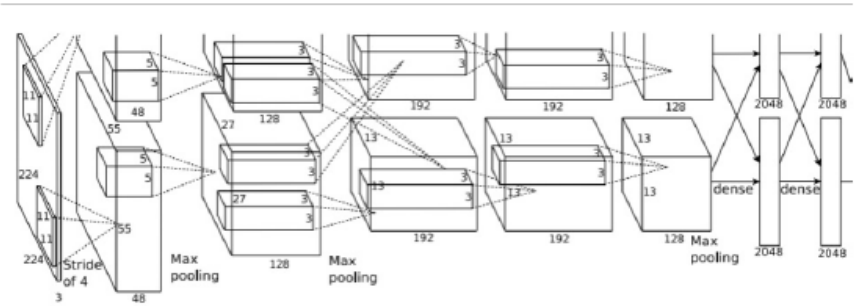


images with multiple objects: number of detections dependent on used confidence threshold

Object Detection: Single Object

(Classification + Localization)





Vector: 4096

Fully Connected: 4096 to 1000

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

Softmax Loss

Correct label: Cat

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

L2 Loss

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

Multitask Loss

+ → Loss

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

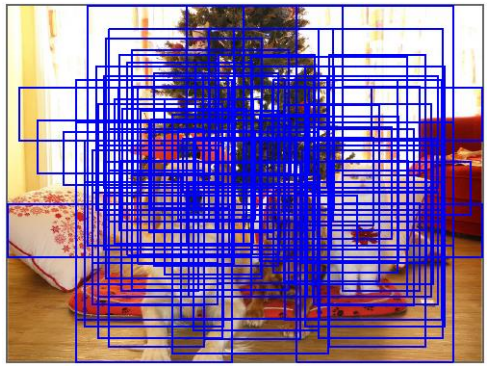
(crops: sliding window over image, iterated at multiple window sizes)

Dog? No
Cat? No
Background? Yes

Dog? Yes
Cat? No
Background? No

Dog? Yes
Cat? No
Background? No

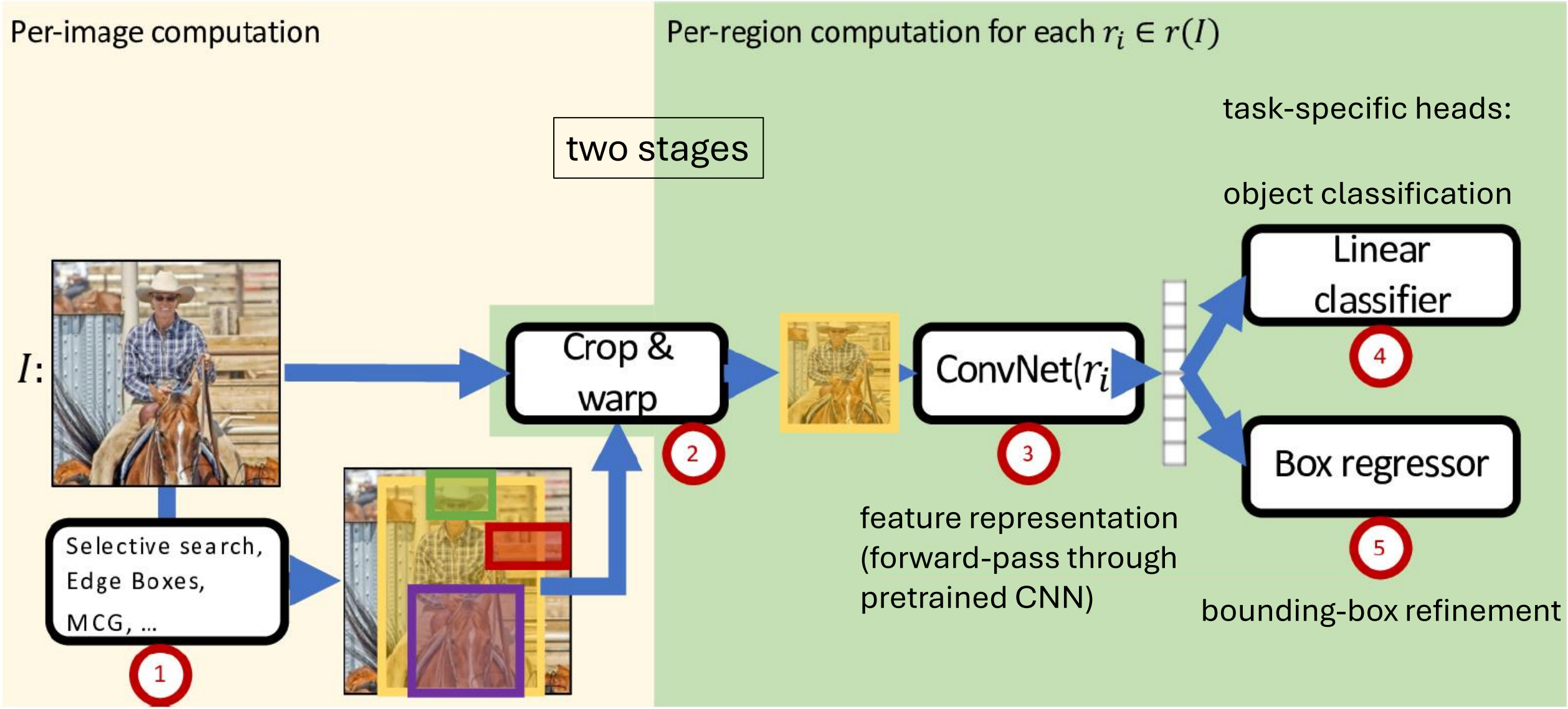
Dog? No
Cat? Yes
Background? No



issue: too many possible crops

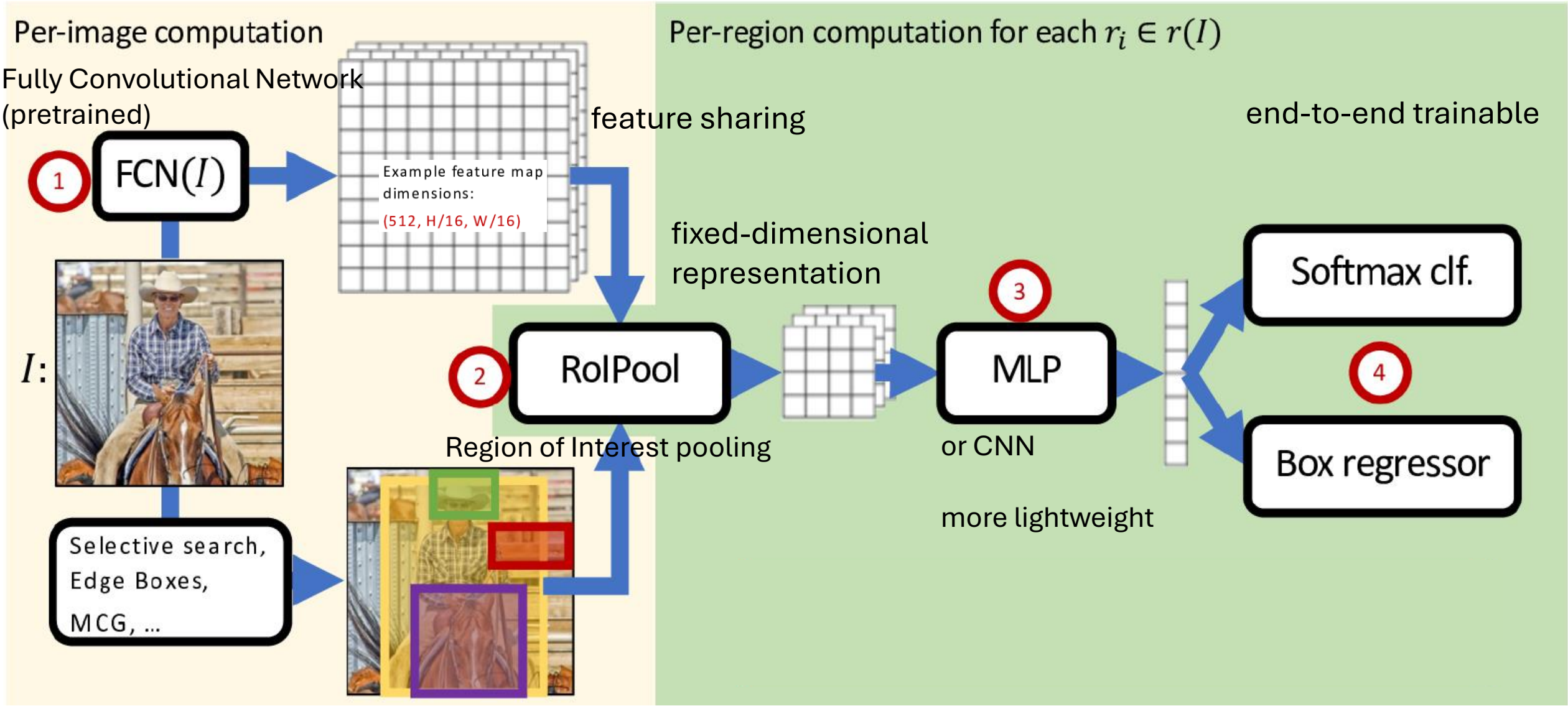
→ need for region proposals

Region-Based Convolutional Neural Network (R-CNN)

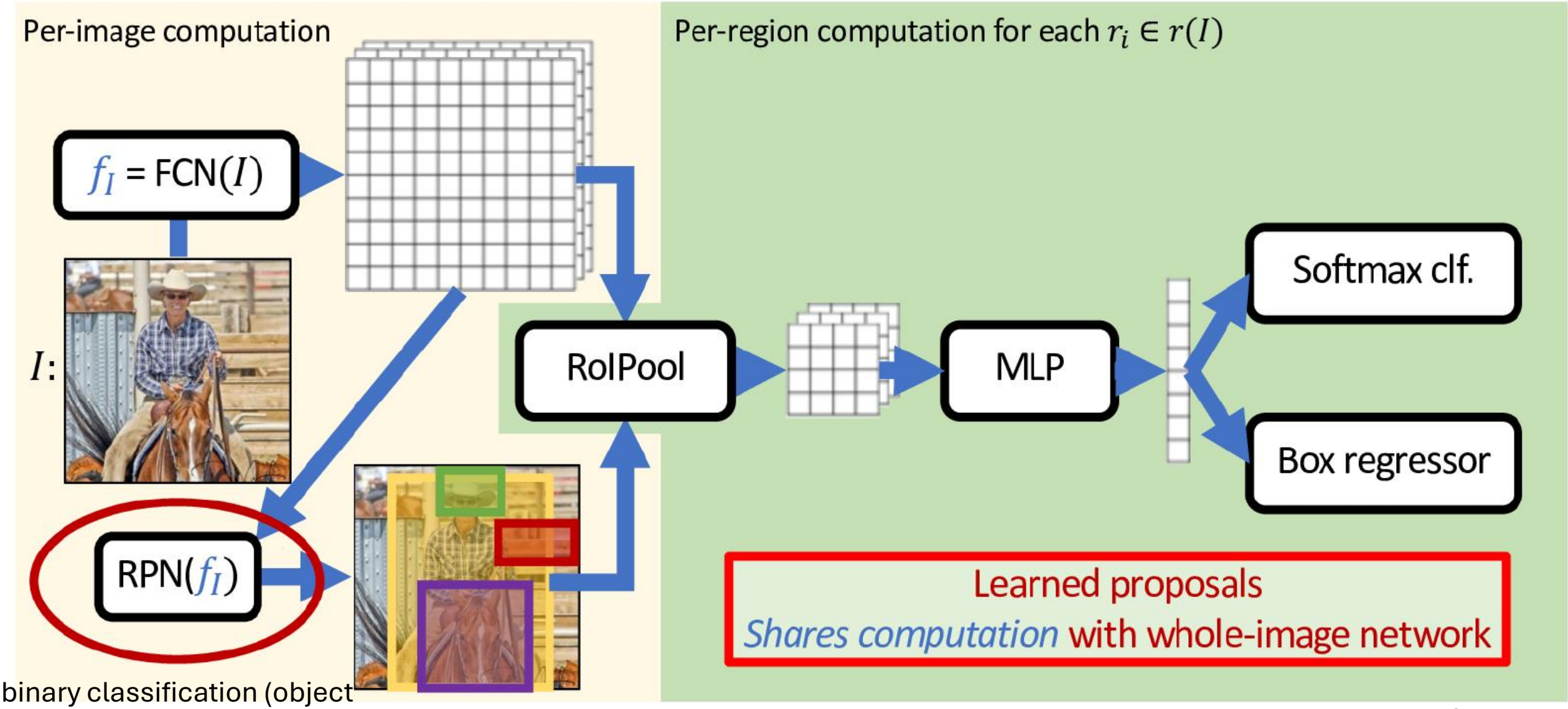


region proposals by "classic" method (~2k)

Fast R-CNN: Crop ConvNet Features

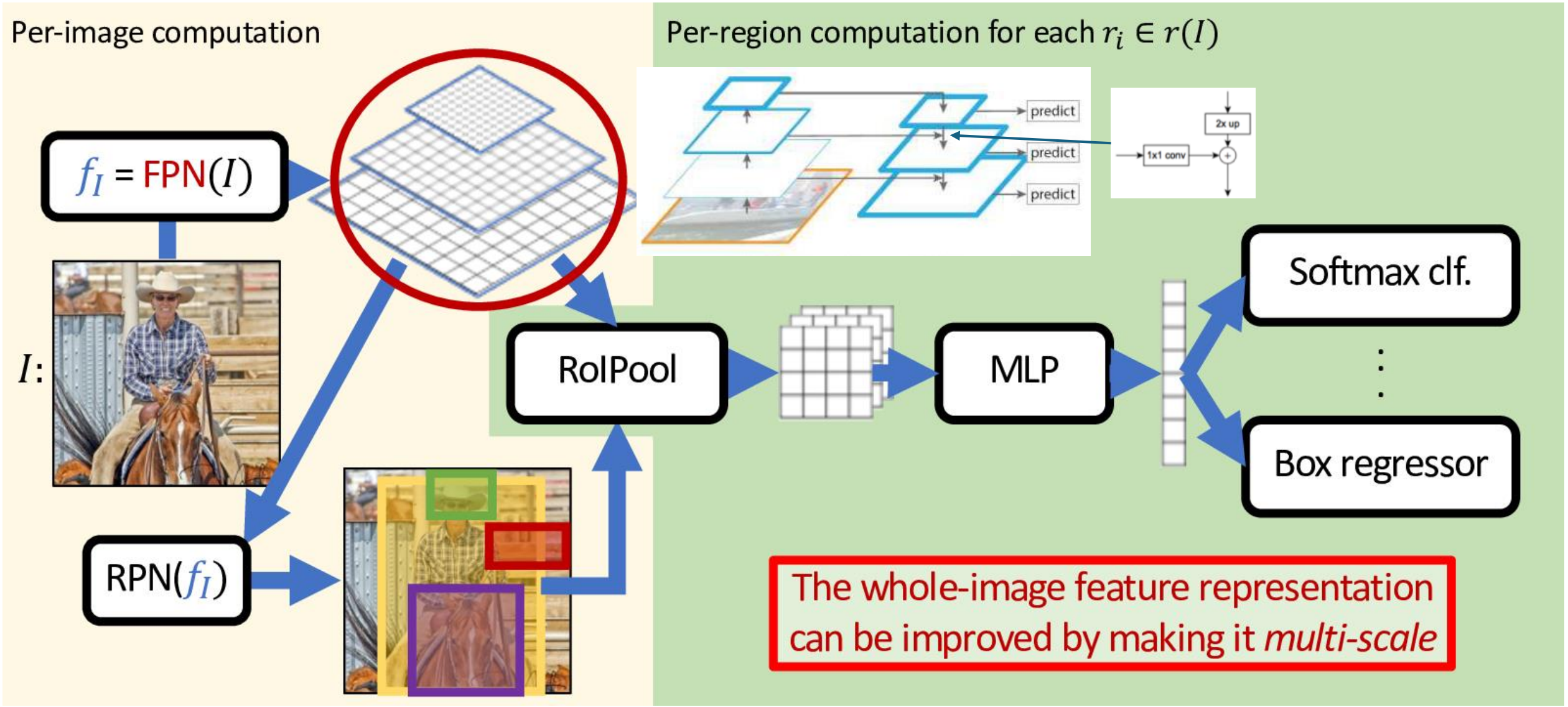


Faster R-CNN: Use Region Proposal Network



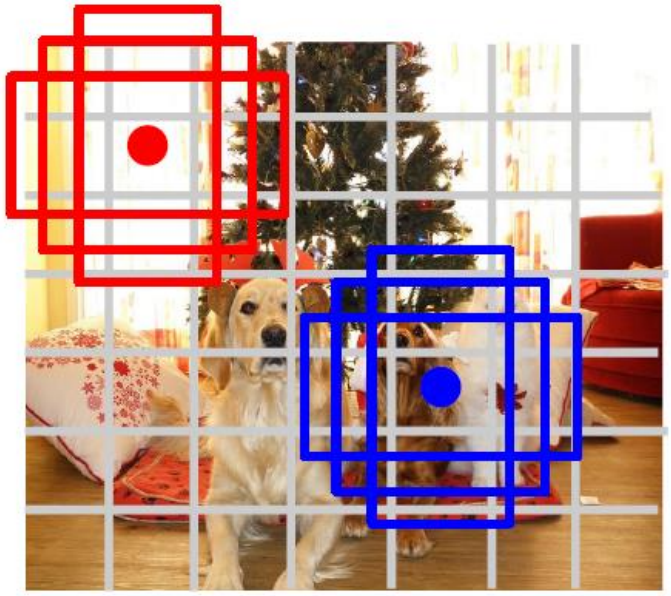
binary classification (object or not) with sliding window

Feature Pyramid Network (FPN)



Single-Stage Detectors: Drop Per-Region Computation

Input image
3 x H x W



Divide image into grid
7 x 7

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

Within each grid cell:

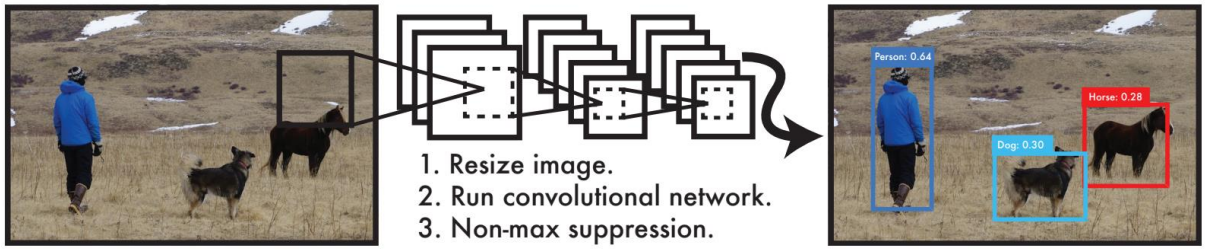
- Regress from each of the B base boxes to a final box with 5 numbers: (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)
- Looks a lot like RPN, but category-specific!

Output:
7 x 7 x (5 * B + C)

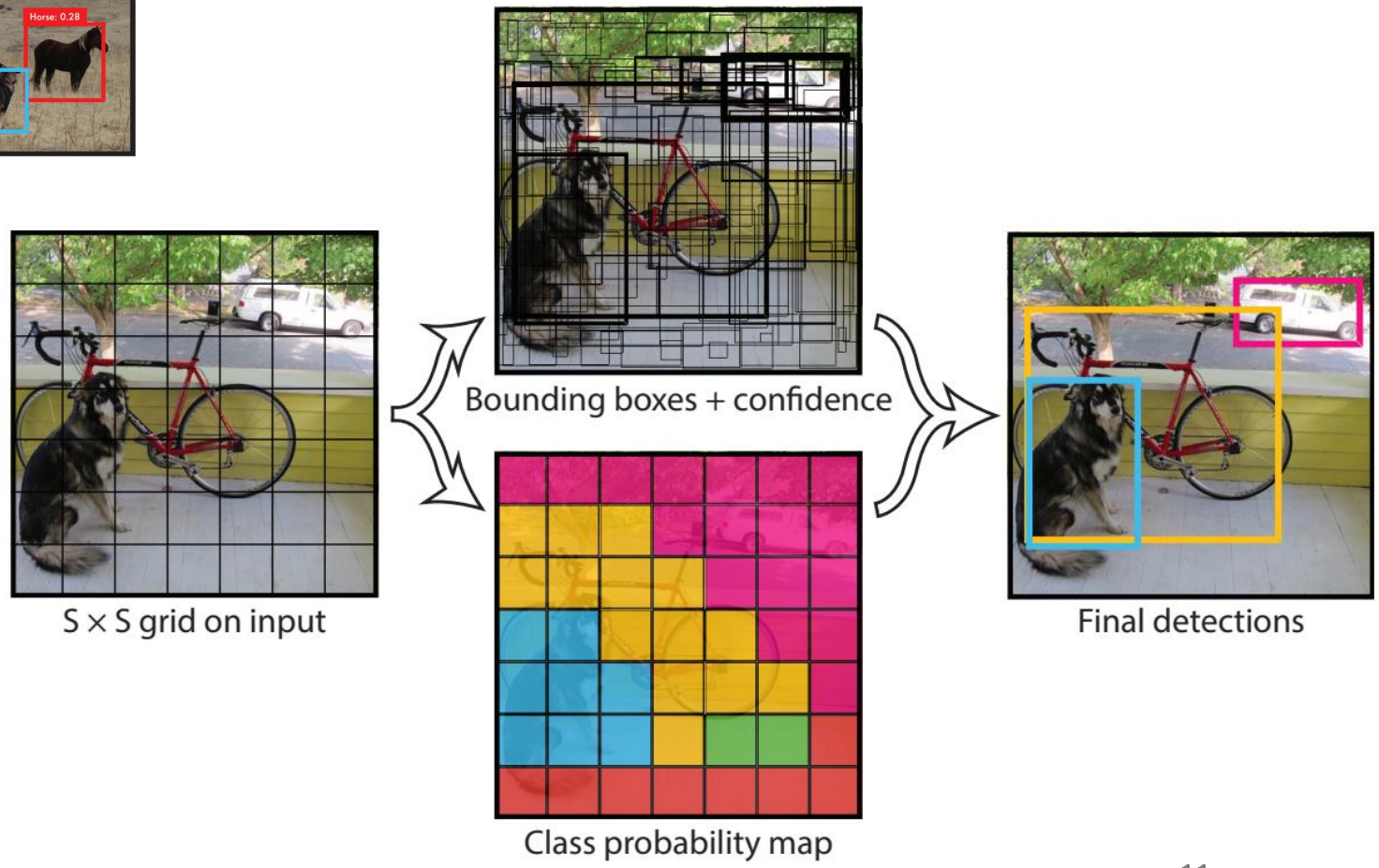
Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016
Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

faster, but usually worse performance

YOLO: Real-Time Object Detection



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



Object Tracking

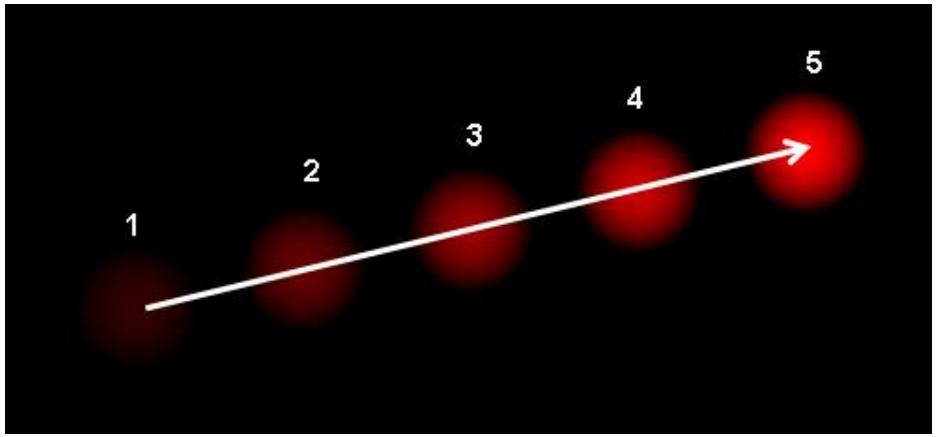
task: locating an object in successive video frames

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:

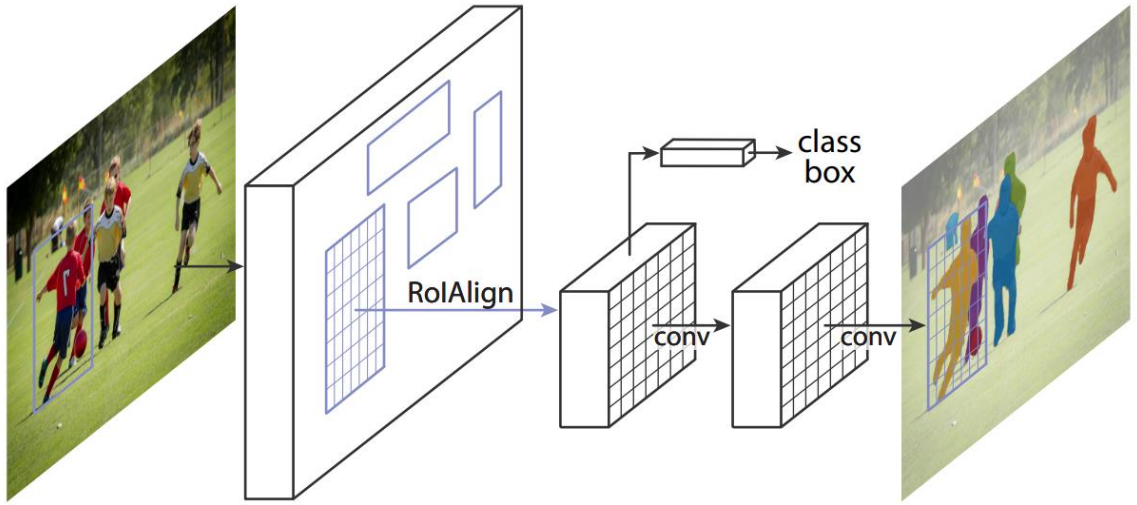
estimate motion vector of few image features

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

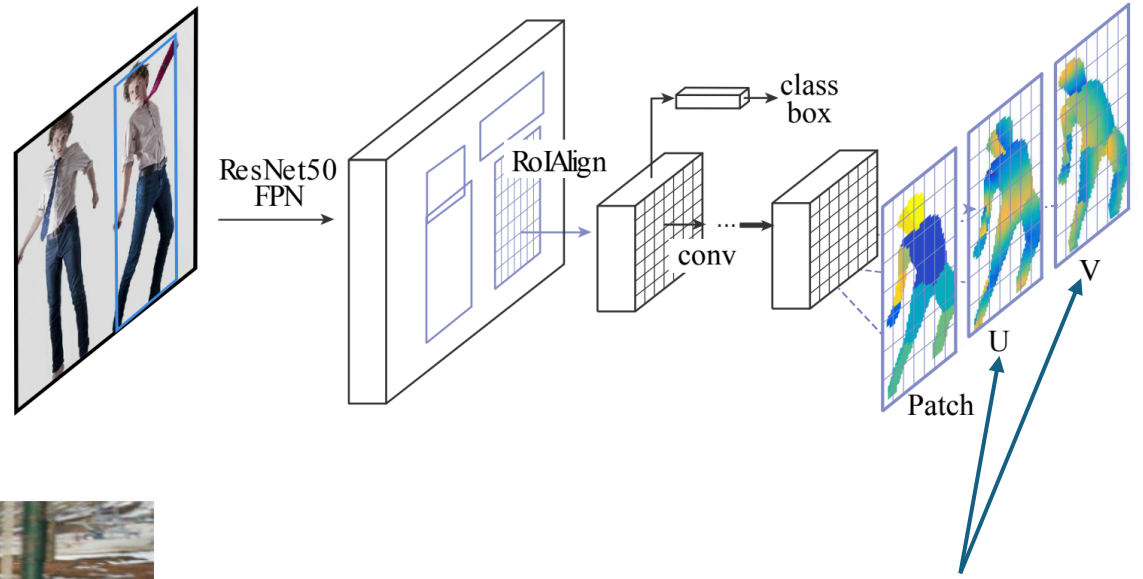
Instance Segmentation

Add Additional Network Heads to R-CNN

instance segmentation (Mask R-CNN):

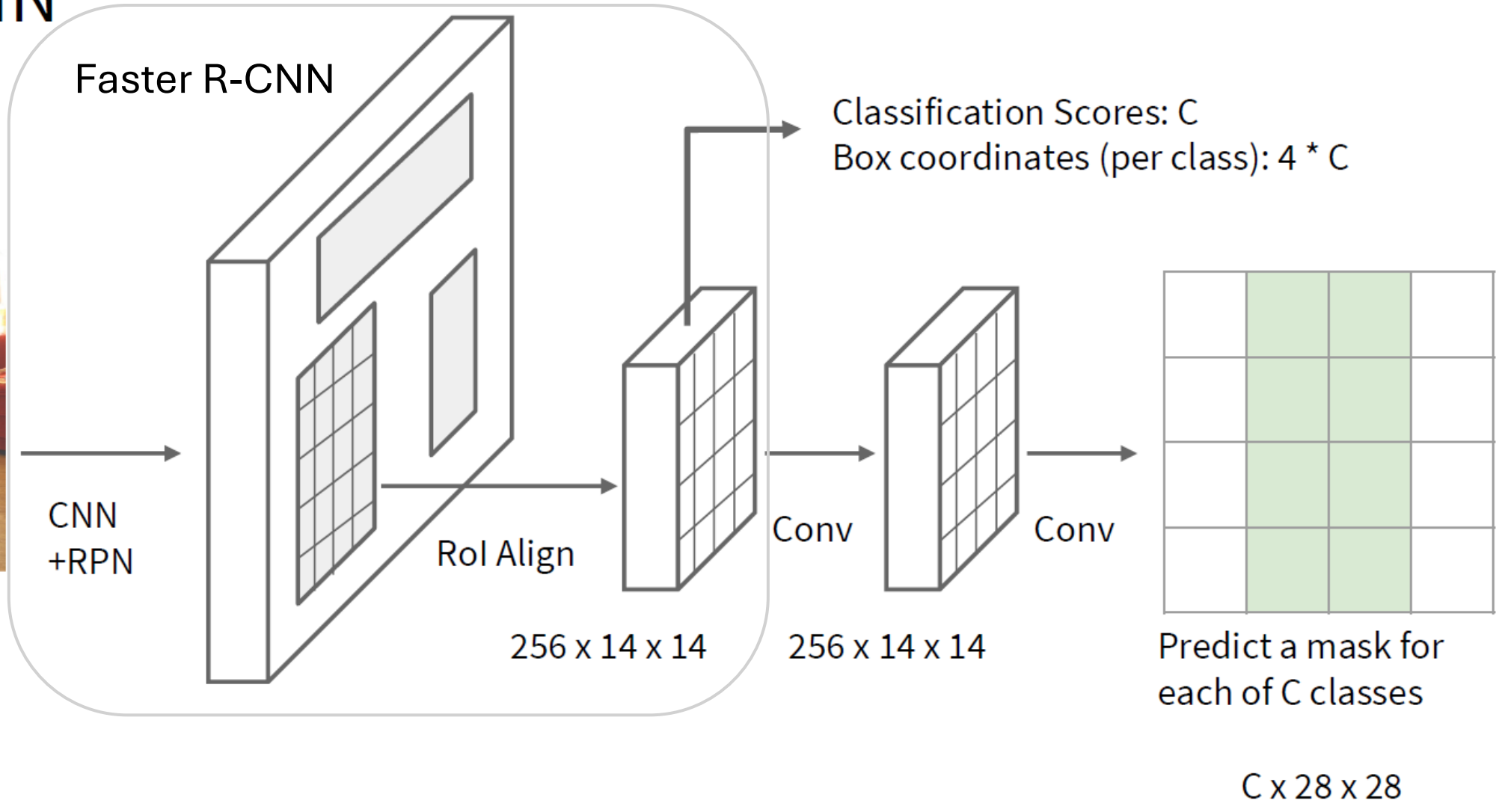


pose estimation (DensePose):



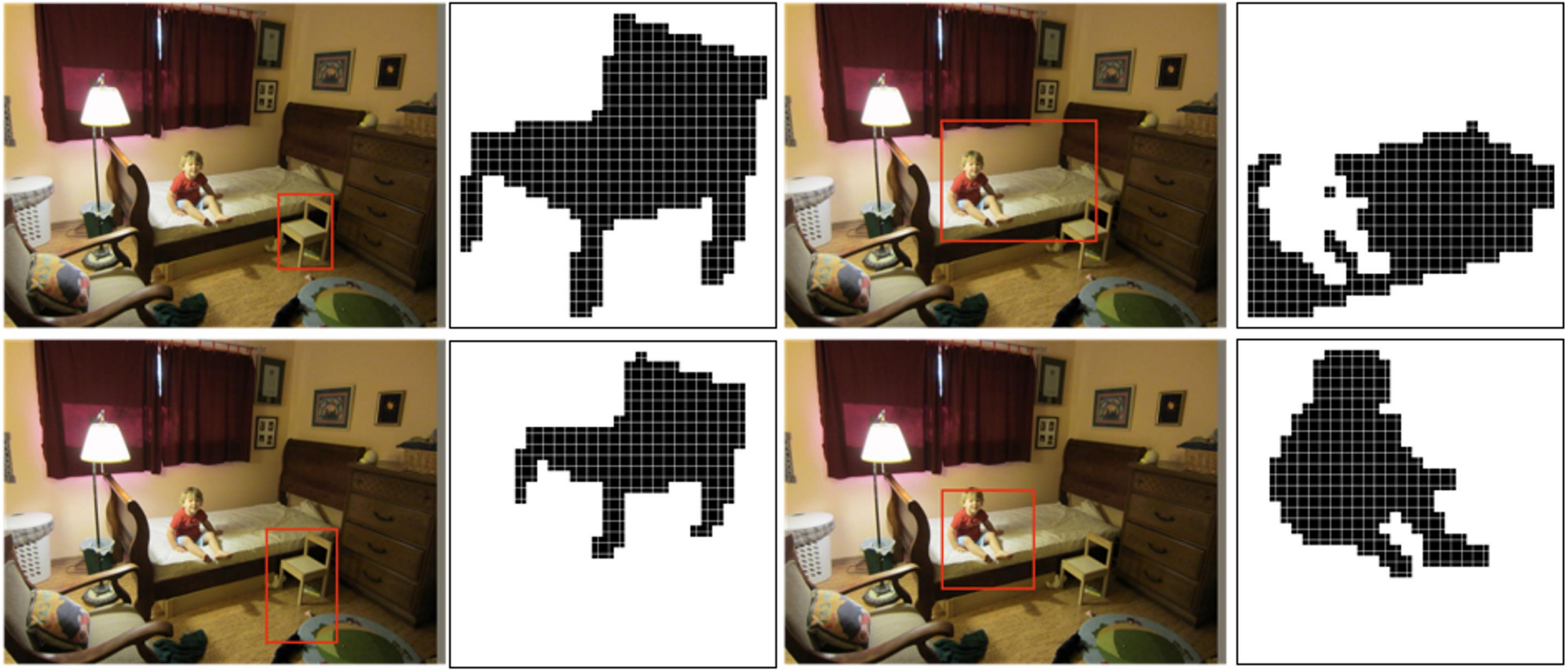
3D model's surface to 2D

Mask R-CNN

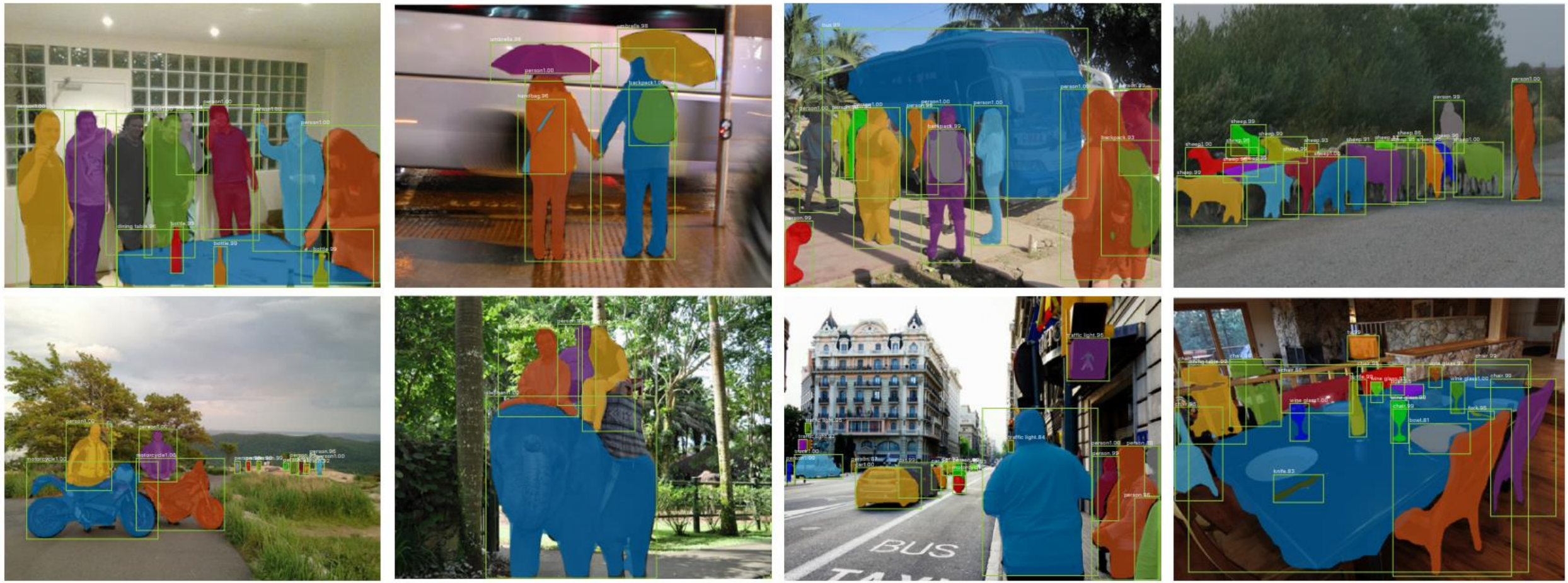


1. object detection (boundary boxes)
2. prediction of separate binary masks for each detected object (specific instances)

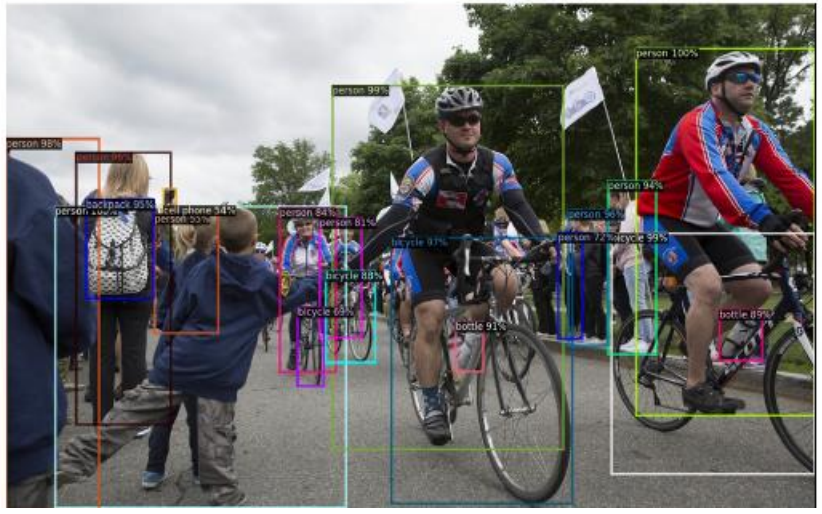
image with training proposal | 28 × 28 mask target | image with training proposal | 28 × 28 mask target



results on MS COCO data set



Detectron2: PyTorch Implementations



Promptable Segmentation with Transformers

Segment Anything Model (SAM)

SAM2 includes also video segmentation