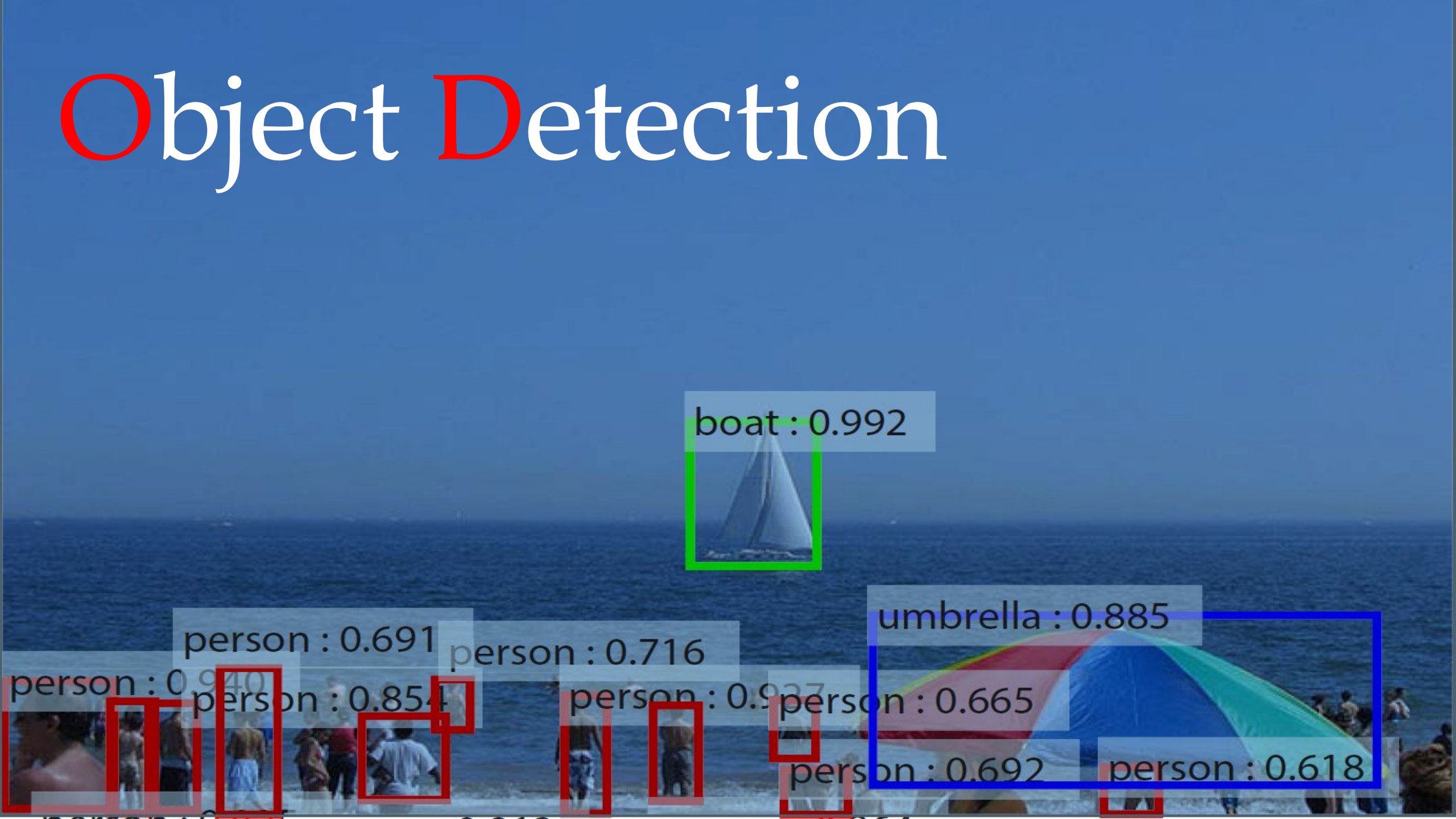
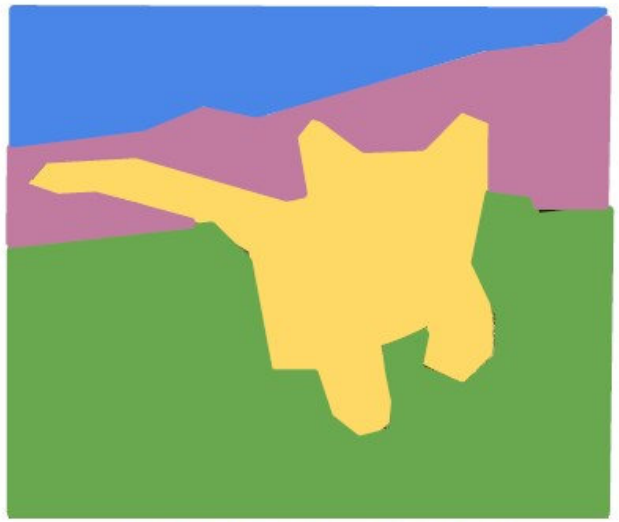
Object Detection



Computer Vision Task

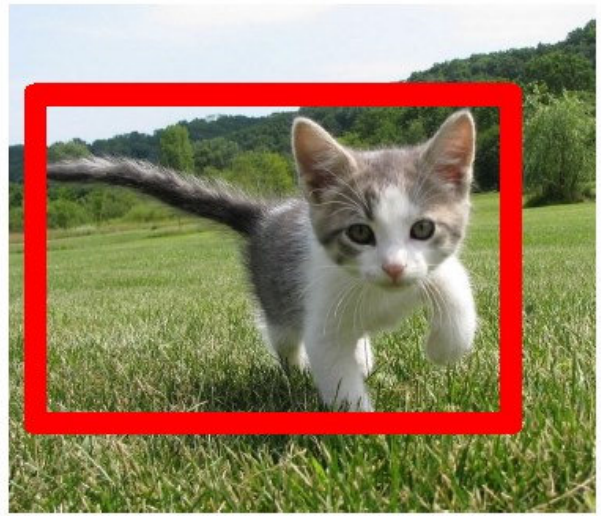
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

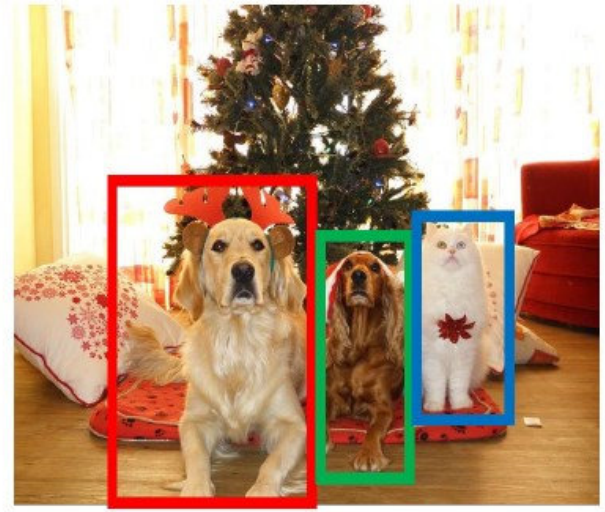
Classification + Localization



CAT

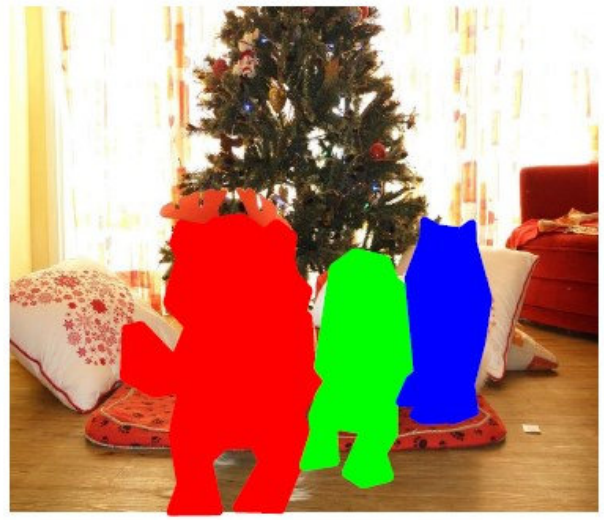
Single Object

Object Detection



DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

Multiple Object

Classification + Localization

Classification: C classes
Input: Image
Output: Class label
Evaluation metric: Accuracy

→ CAT

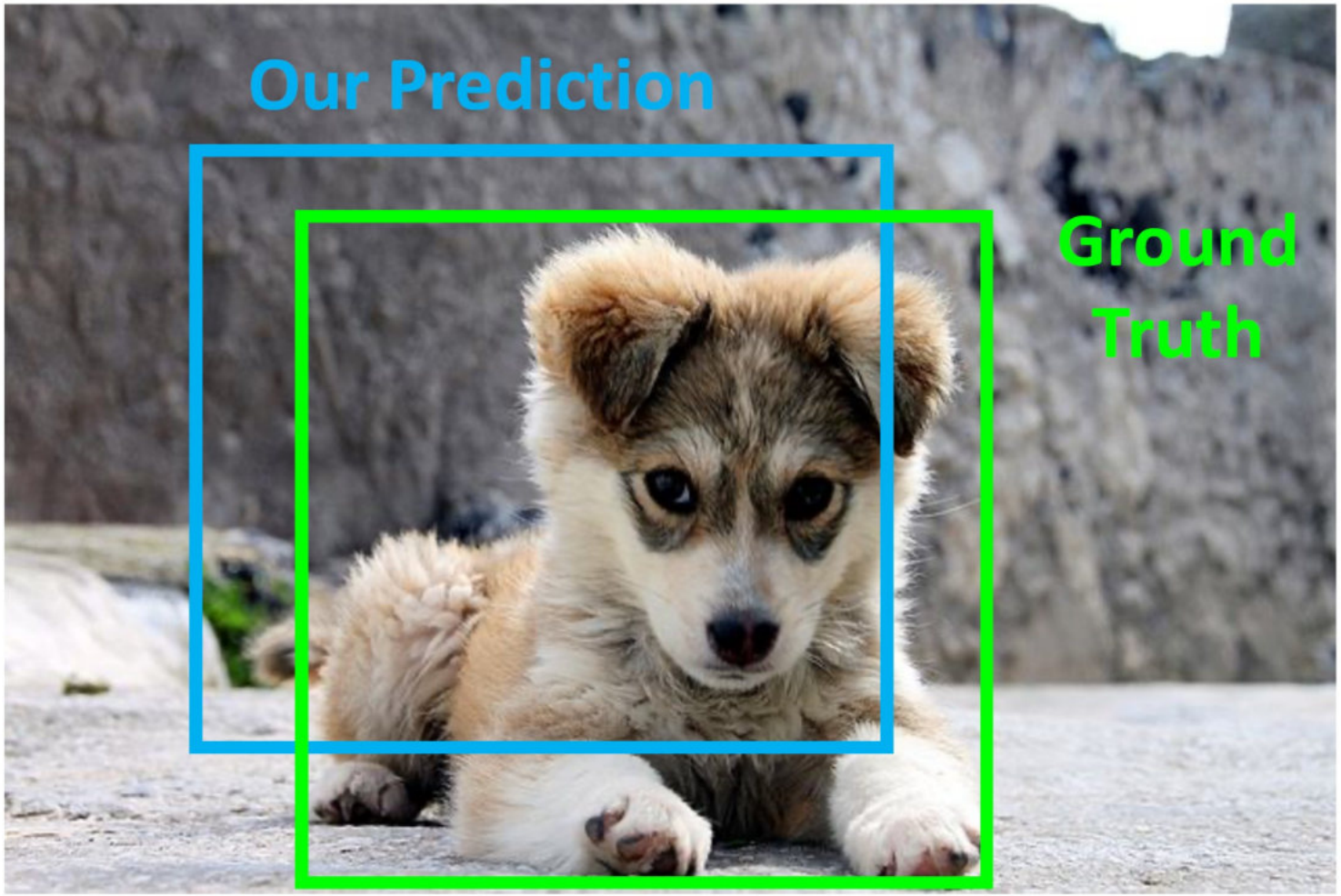
Localization:
Input: Image
Output: Box in the image (x, y, w, h)
Evaluation metric: Intersection over Union

→ (x, y, w, h)

Classification + Localization: Do both

Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

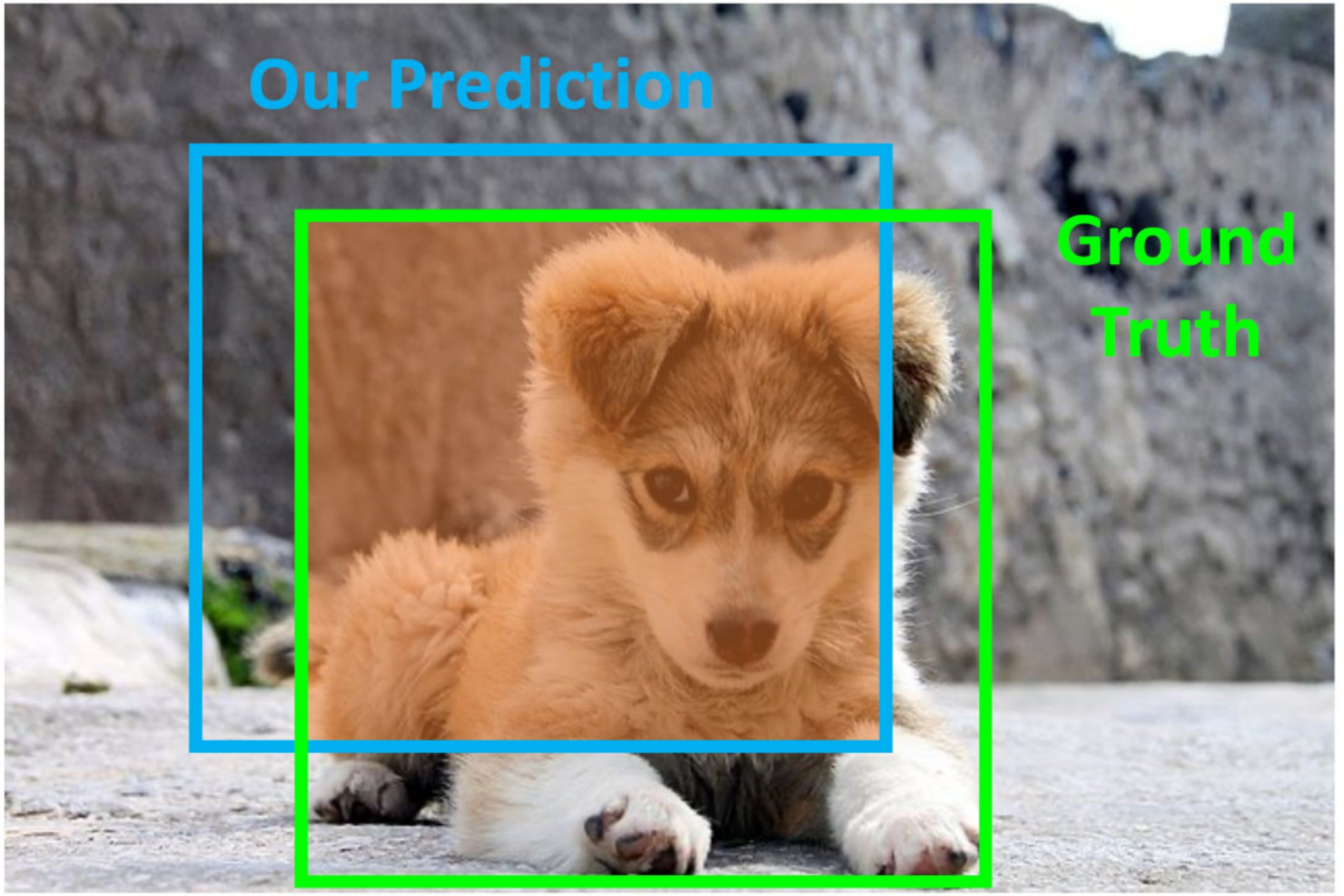


Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU)
(Also called "Jaccard similarity" or "Jaccard index"):

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

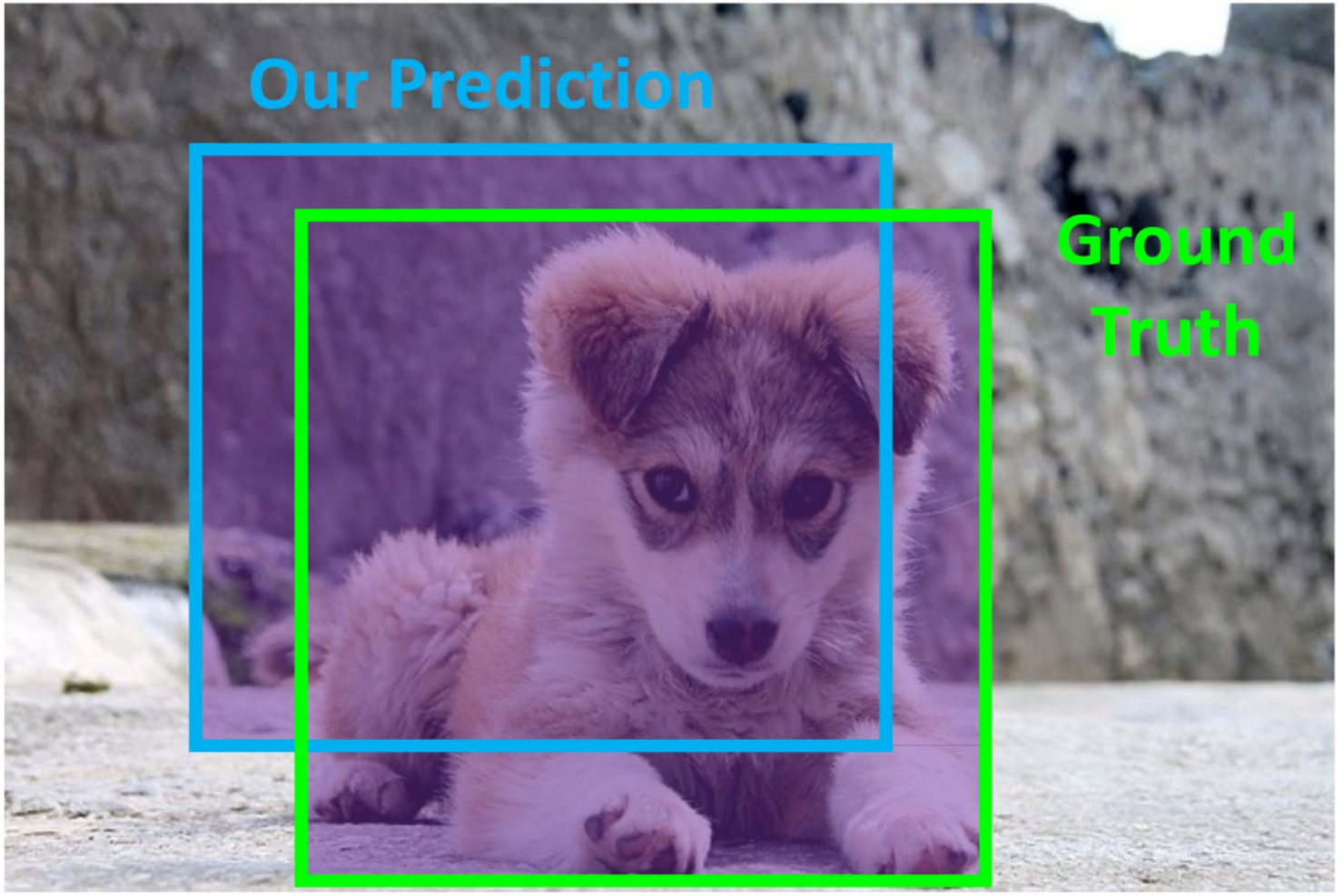


Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU)
(Also called "Jaccard similarity" or "Jaccard index"):

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$



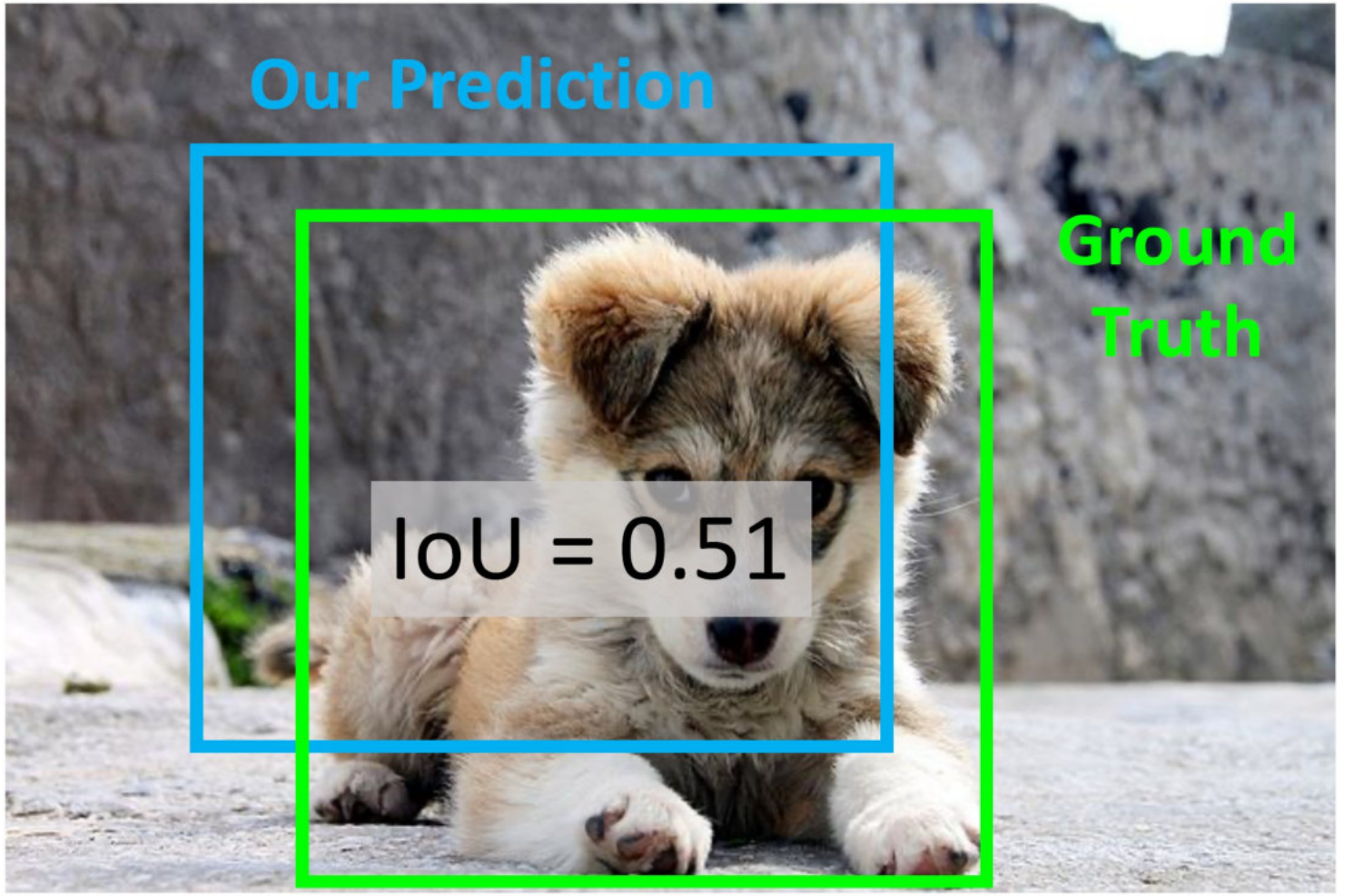
Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU)
(Also called "Jaccard similarity" or "Jaccard index"):

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

IoU > 0.5 is "decent"



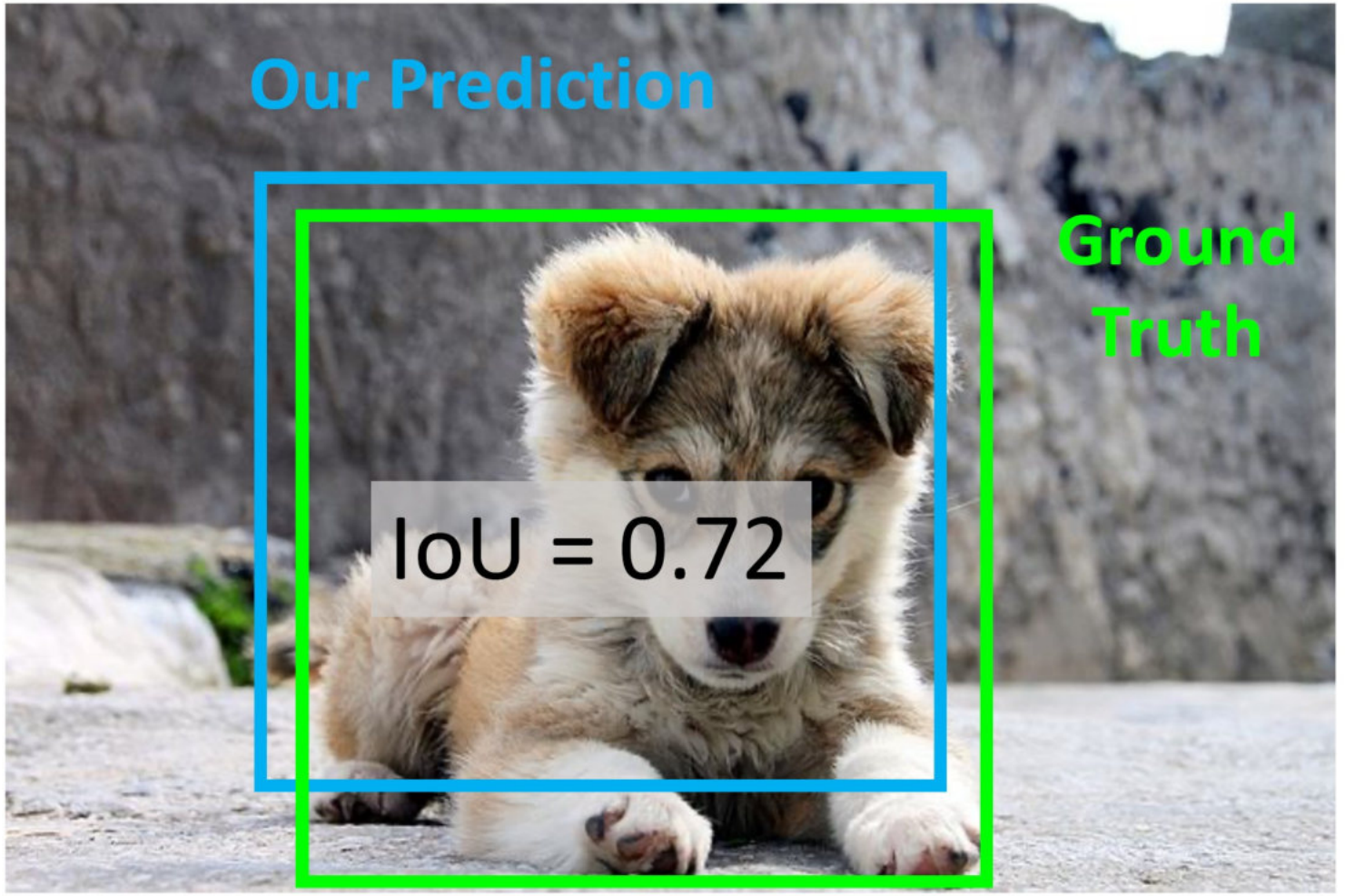
Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU)
(Also called "Jaccard similarity" or "Jaccard index"):

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

IoU > 0.5 is "decent",
IoU > 0.7 is "pretty good",



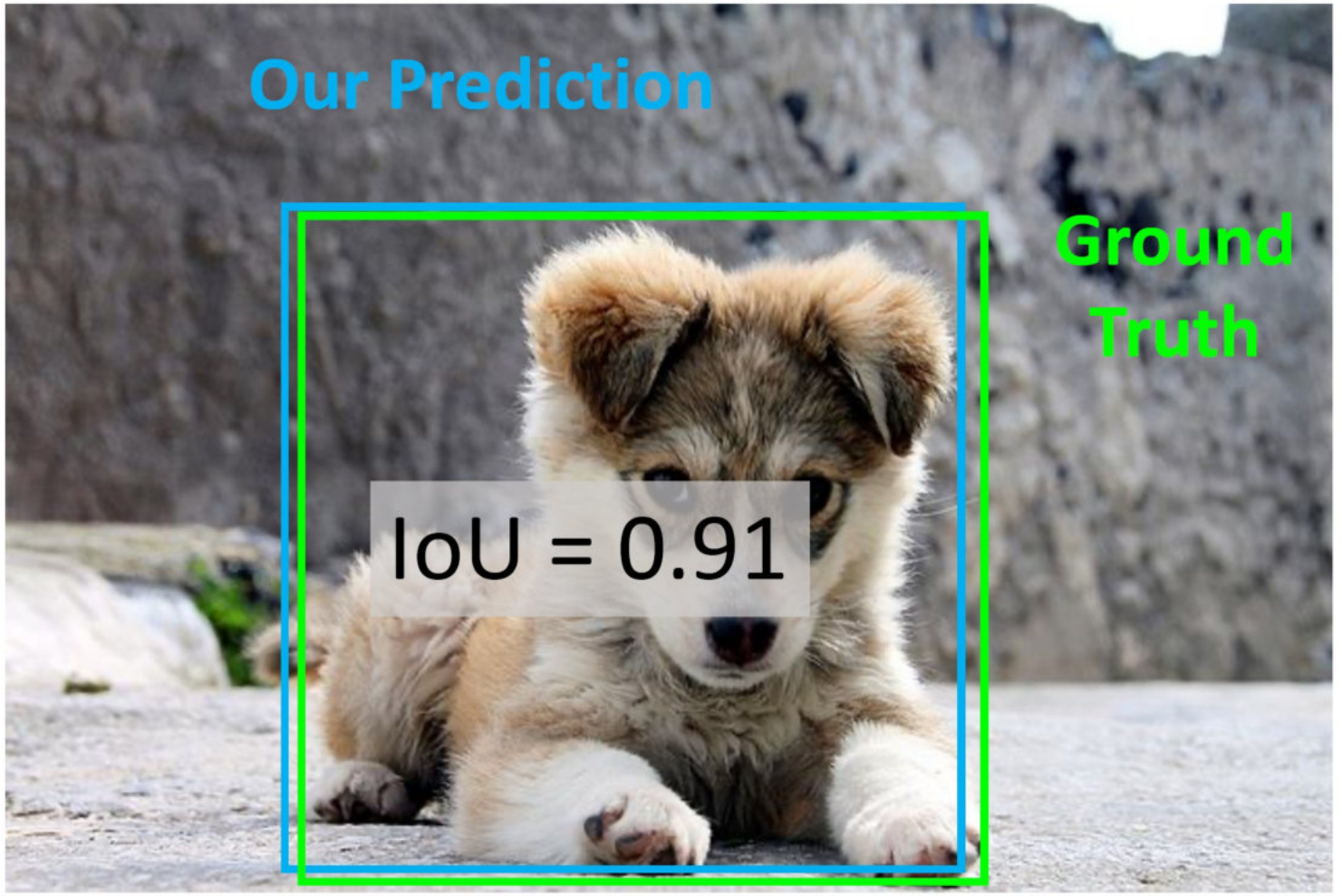
Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU)
(Also called "Jaccard similarity" or "Jaccard index"):

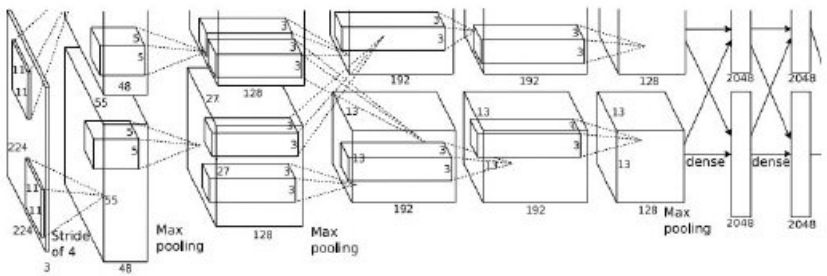
$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

IoU > 0.5 is "decent",
IoU > 0.7 is "pretty good",
IoU > 0.9 is "almost perfect"



Classification + Localization

This image is CC0 public domain



Correct label:
Cat

Fully Connected:
4096 to 1000

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

Softmax Loss

Multitask Loss

Loss

Vector:
4096

Fully Connected:
4096 to 4

Box Coordinates
(x, y, w, h)

L2 Loss

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

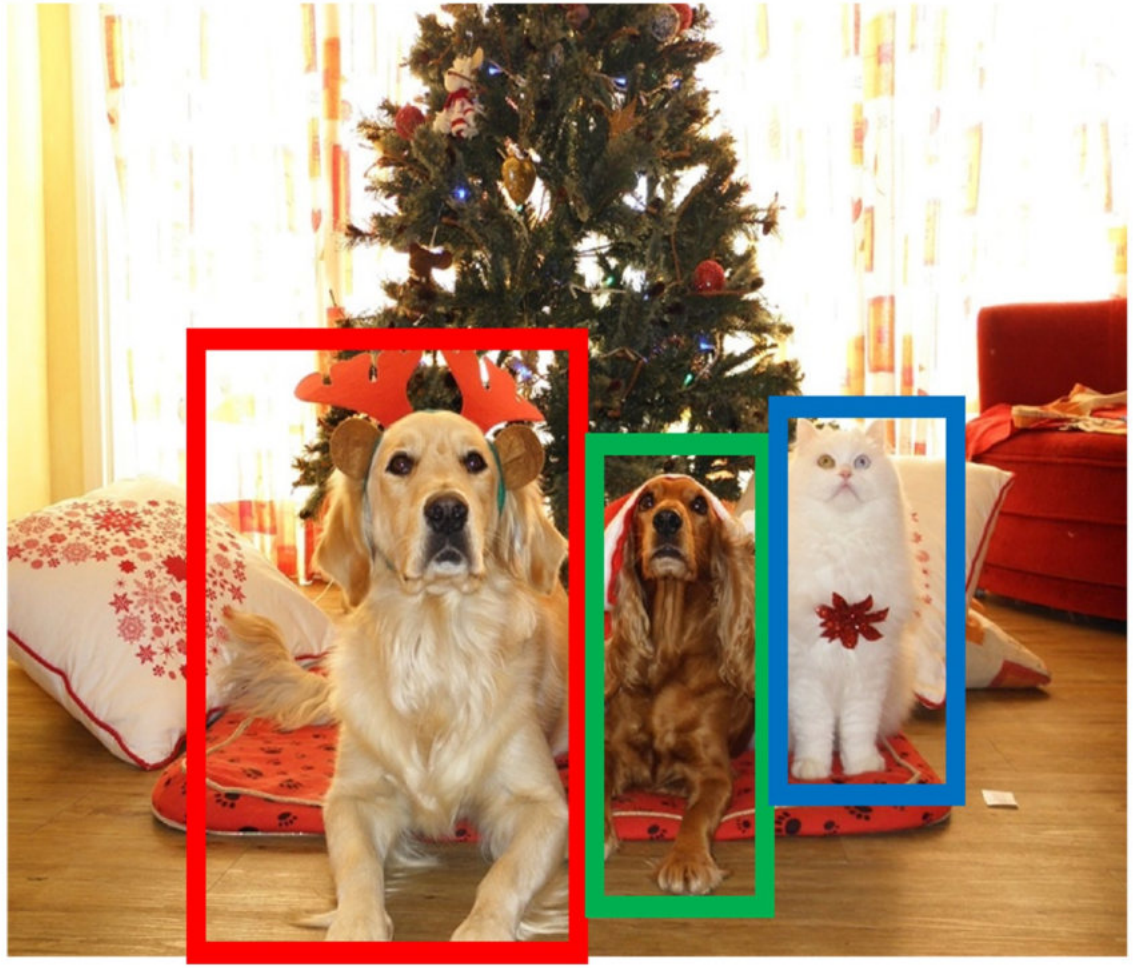
Treat localization as a regression problem!

Object Detection: Task Definition

Input: Single RGB Image

Output: A set of detected objects;
For each object predict:

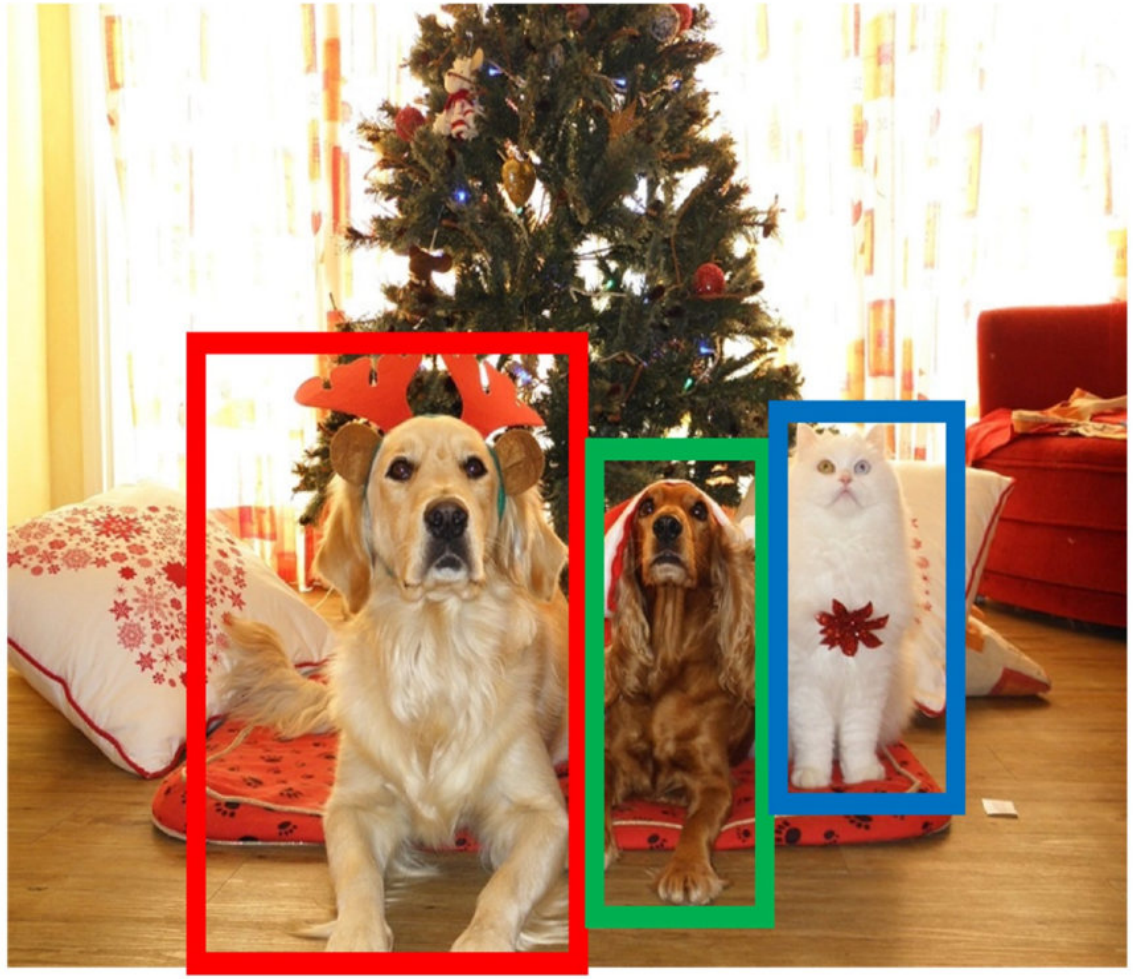
1. Category label (from fixed, known set of categories)
2. Bounding box (four numbers: x, y, width, height)



Slide from Deep Learning for Computer Vision @Univ. of Michigan

Object Detection: Challenges

- **Multiple outputs**: Need to output variable numbers of objects per image
- **Multiple types of output**: Need to predict ”what” (category label) as well as “where” (bounding box)
- **Large images**: Classification works at 224x224; need higher resolution for detection, often ~800x600



Slide from Deep Learning for Computer Vision @Univ. of Michigan

R-CNN

1. Generating category independent region proposals
2. Extracting a fixed length feature vector from CNN
3. Class specific linear SVM

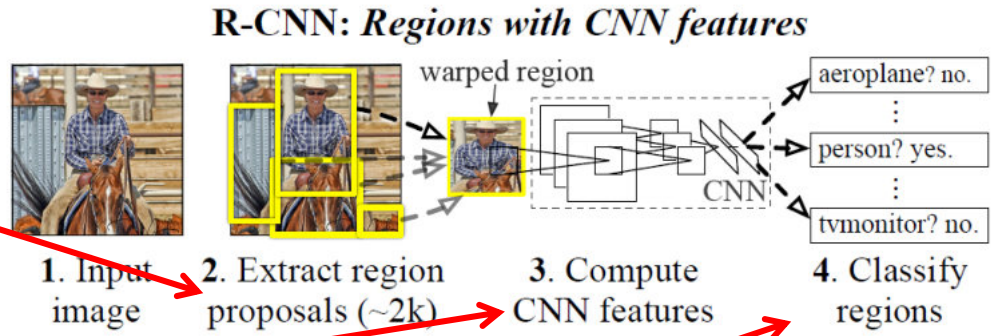
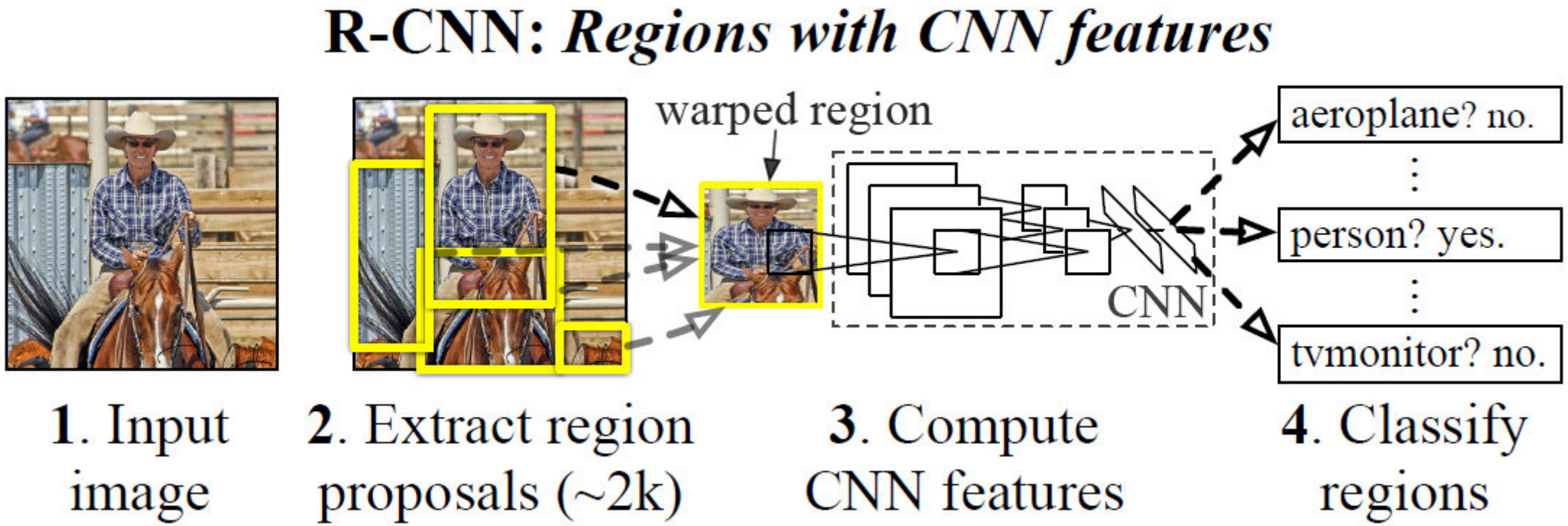
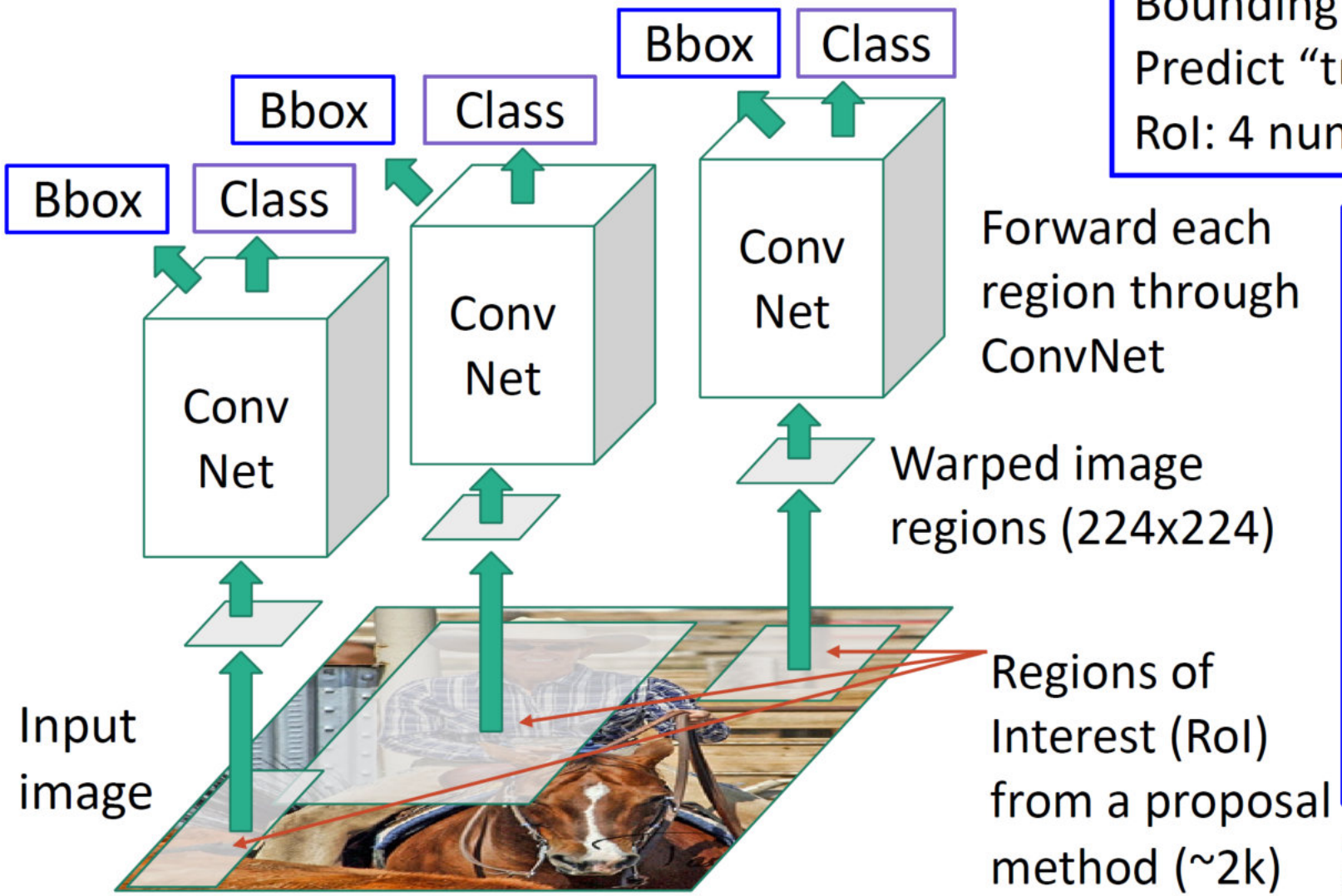


Figure 1: Object detection system overview. Our system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs. R-CNN achieves a mean average precision (mAP) of **53.7% on PASCAL VOC 2010**. For comparison, [39] reports 35.1% mAP using the same region proposals, but with a spatial pyramid and bag-of-visual-words approach. The popular deformable part models perform at 33.4%. On the 200-class **ILSVRC2013 detection dataset, R-CNN's mAP is 31.4%**, a large improvement over OverFeat [34], which had the previous best result at 24.3%.

R-CNN Architecture



R-CNN: Region-Based CNN



Forward each region through ConvNet

Warped image regions (224x224)

Regions of Interest (RoI) from a proposal method (~2k)

Classify each region

Bounding box regression:
Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)

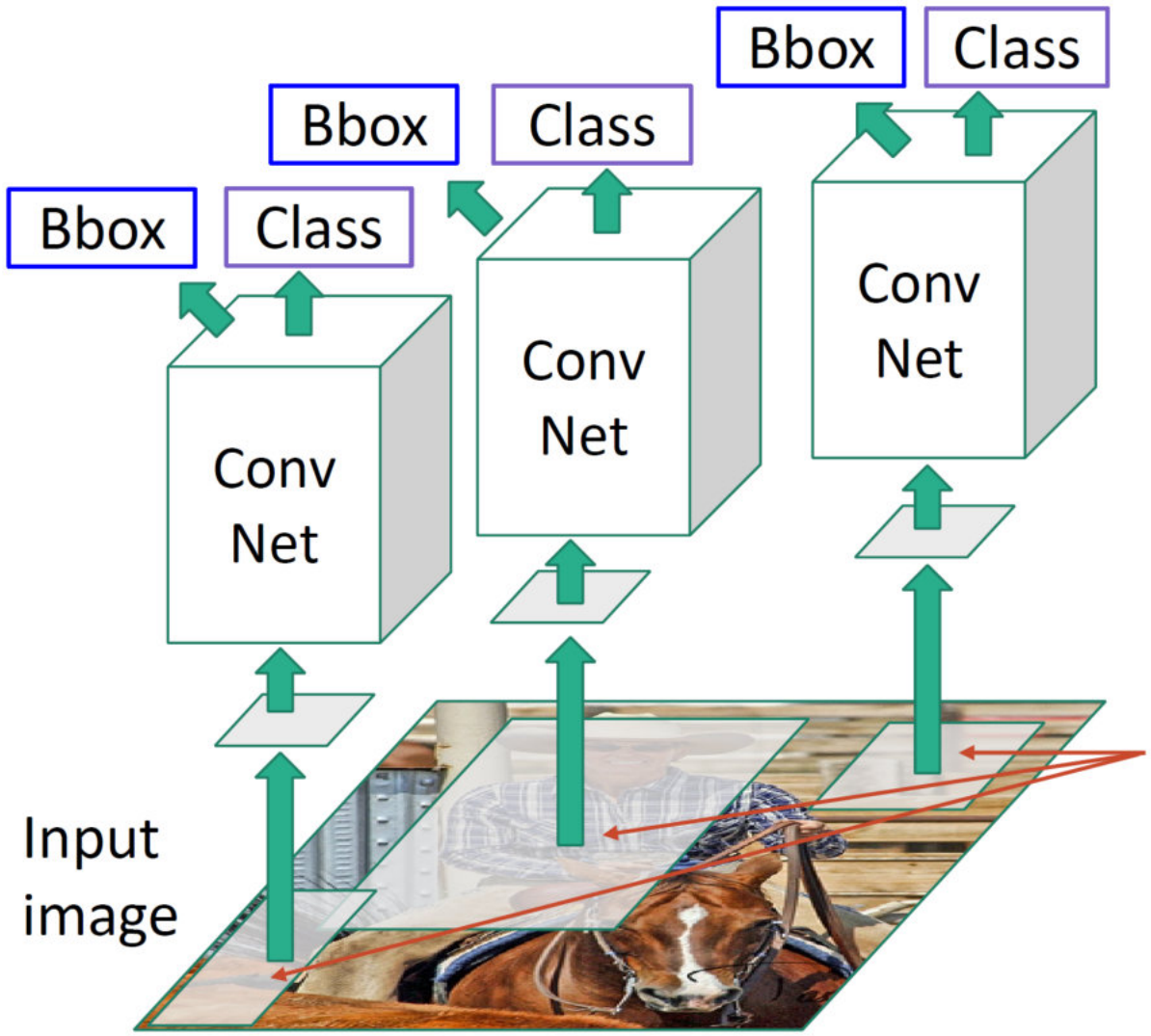
Region proposal: (p_x, p_y, p_h, p_w)
Transform: (t_x, t_y, t_h, t_w)
Output box: (b_x, b_y, b_h, b_w)

Translate relative to box size:
$b_x = p_x + p_w t_x \qquad b_y = p_y + p_h t_y$

Log-space scale transform:
$b_w = p_w \exp(t_w) \qquad b_h = p_h \exp(t_h)$

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

R-CNN: Test-time

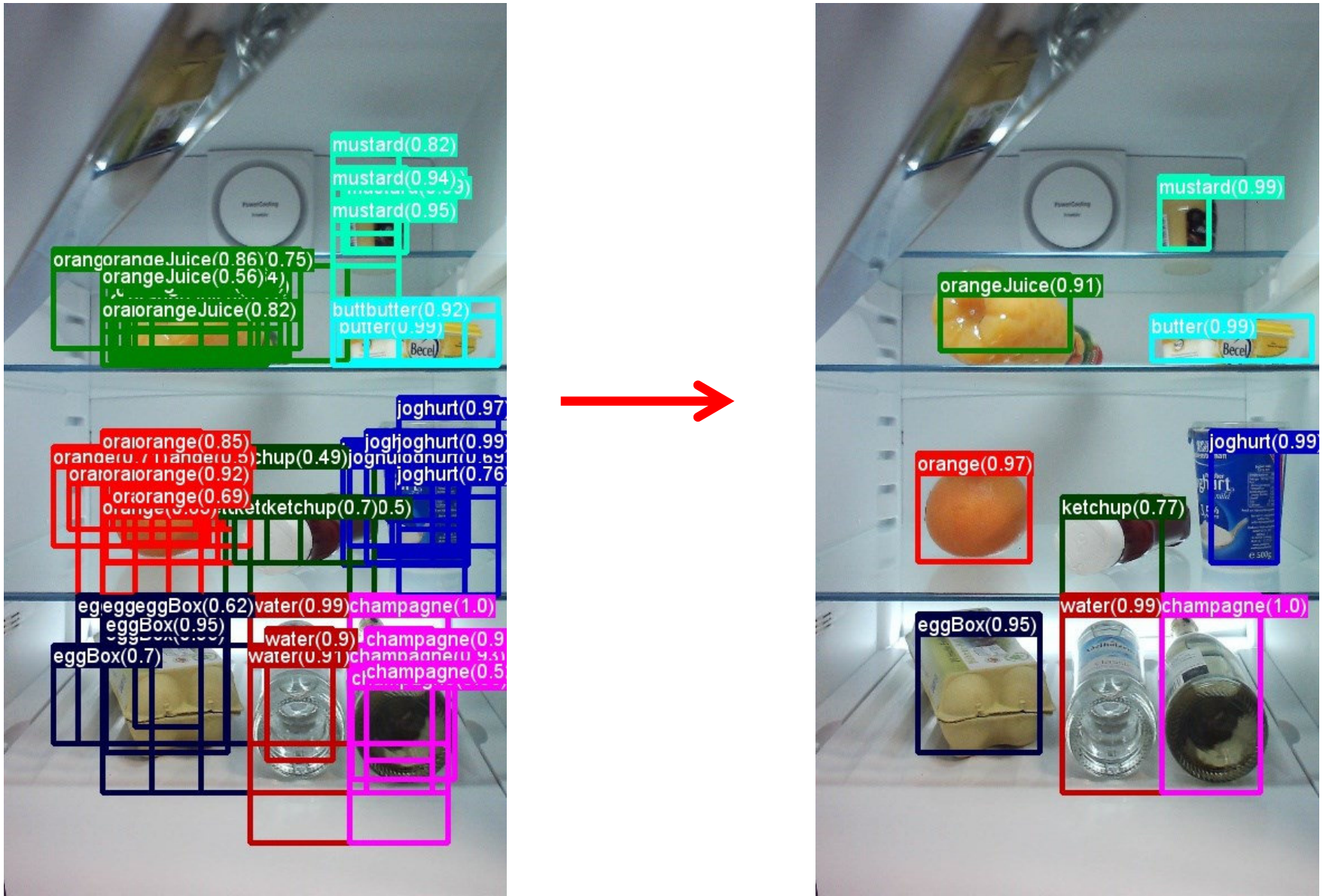


Input: Single RGB Image

1. Run region proposal method to compute ~2000 region proposals
2. Resize each region to 224x224 and run independently through CNN to predict class scores and bbox transform
3. Use scores to select a subset of region proposals to output
   (Many choices here: threshold on background, or per-category? Or take top K proposals per image?)
4. Compare with ground-truth boxes

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

Non-Maximum Suppresion

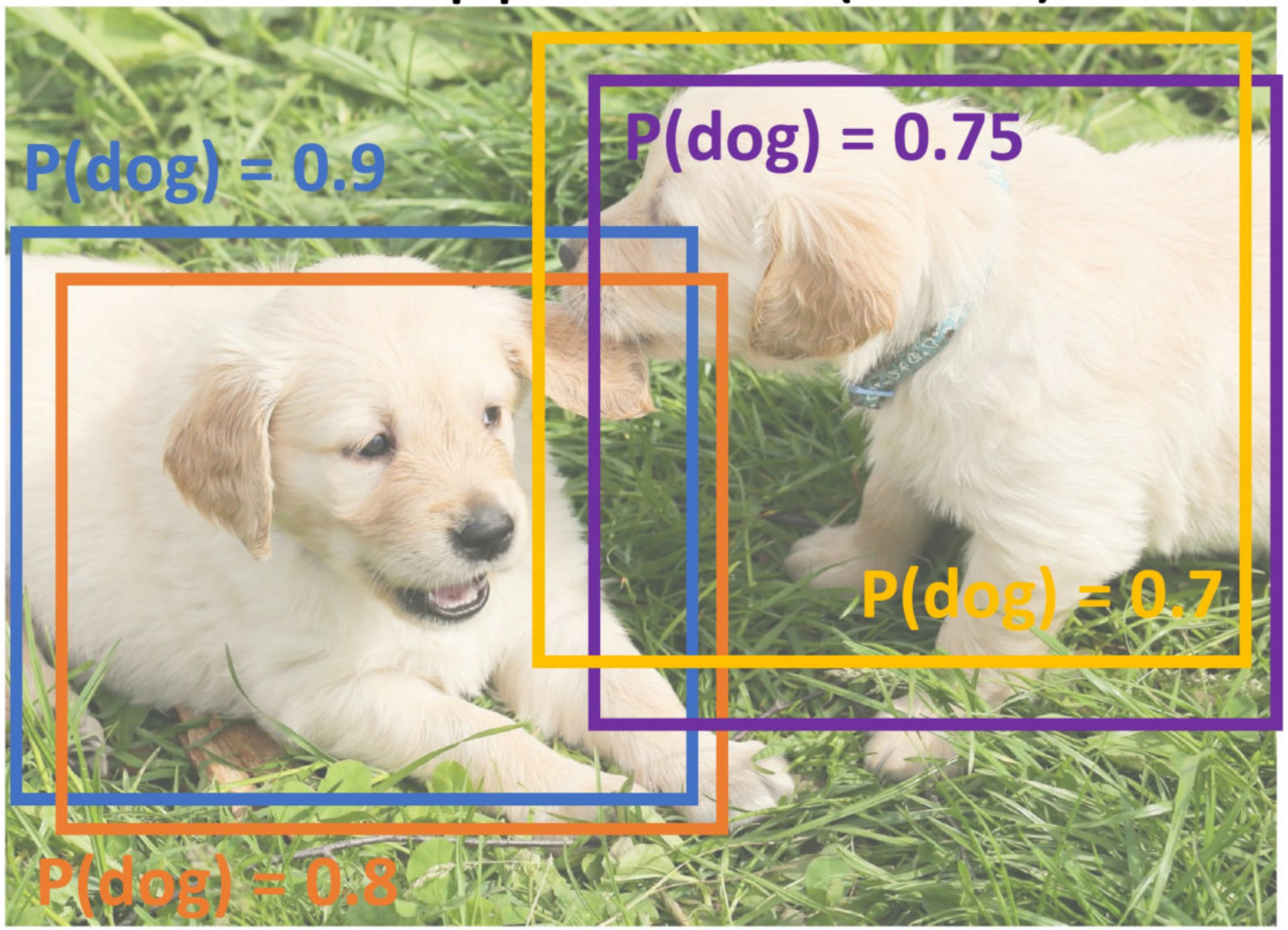


Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections:

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1



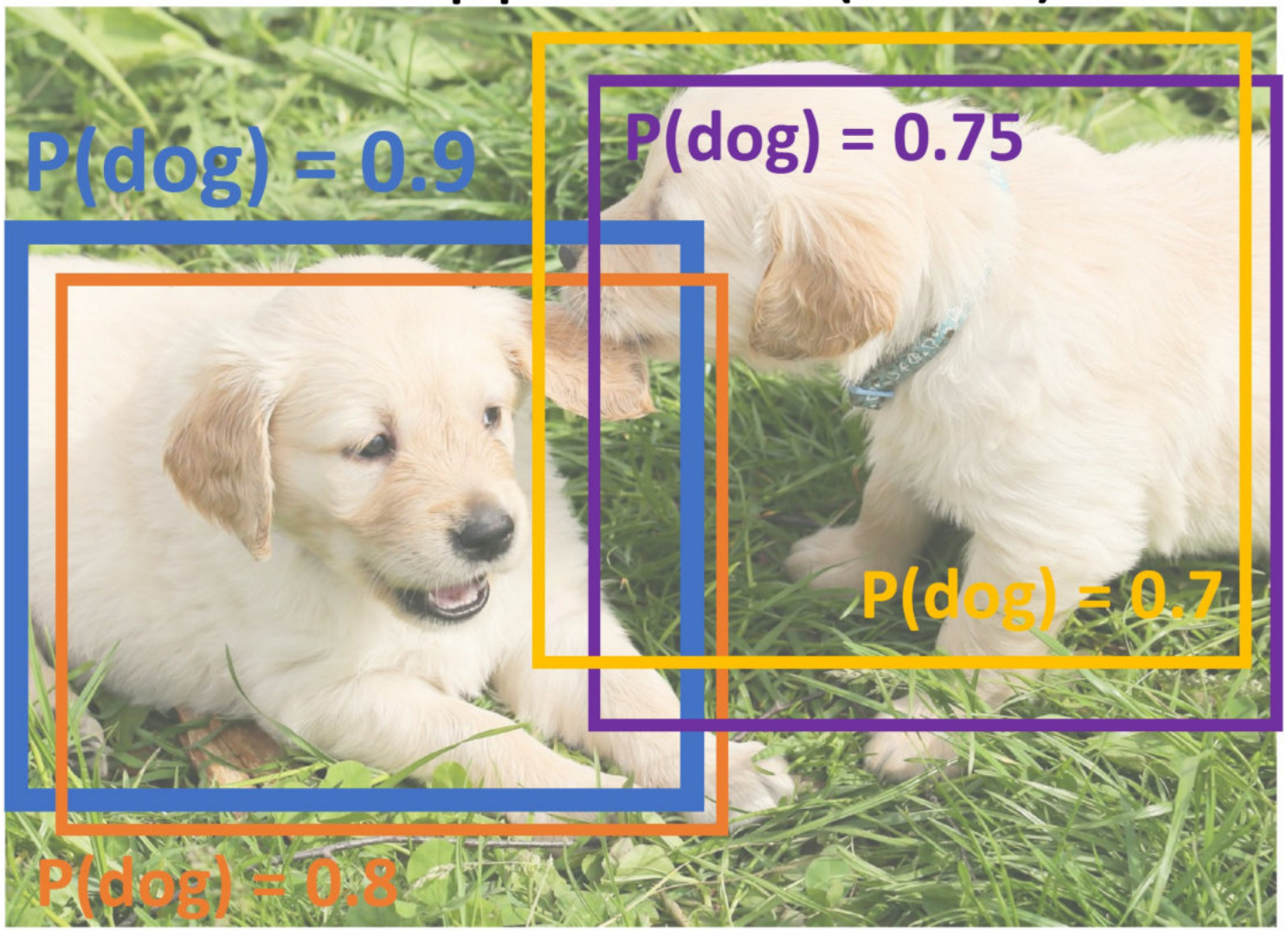
Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections:

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

IoU(■, ■) = **0.78**
IoU(■, ■) = 0.05
IoU(■, ■) = 0.07



Puppy image is CC0 Public Domain

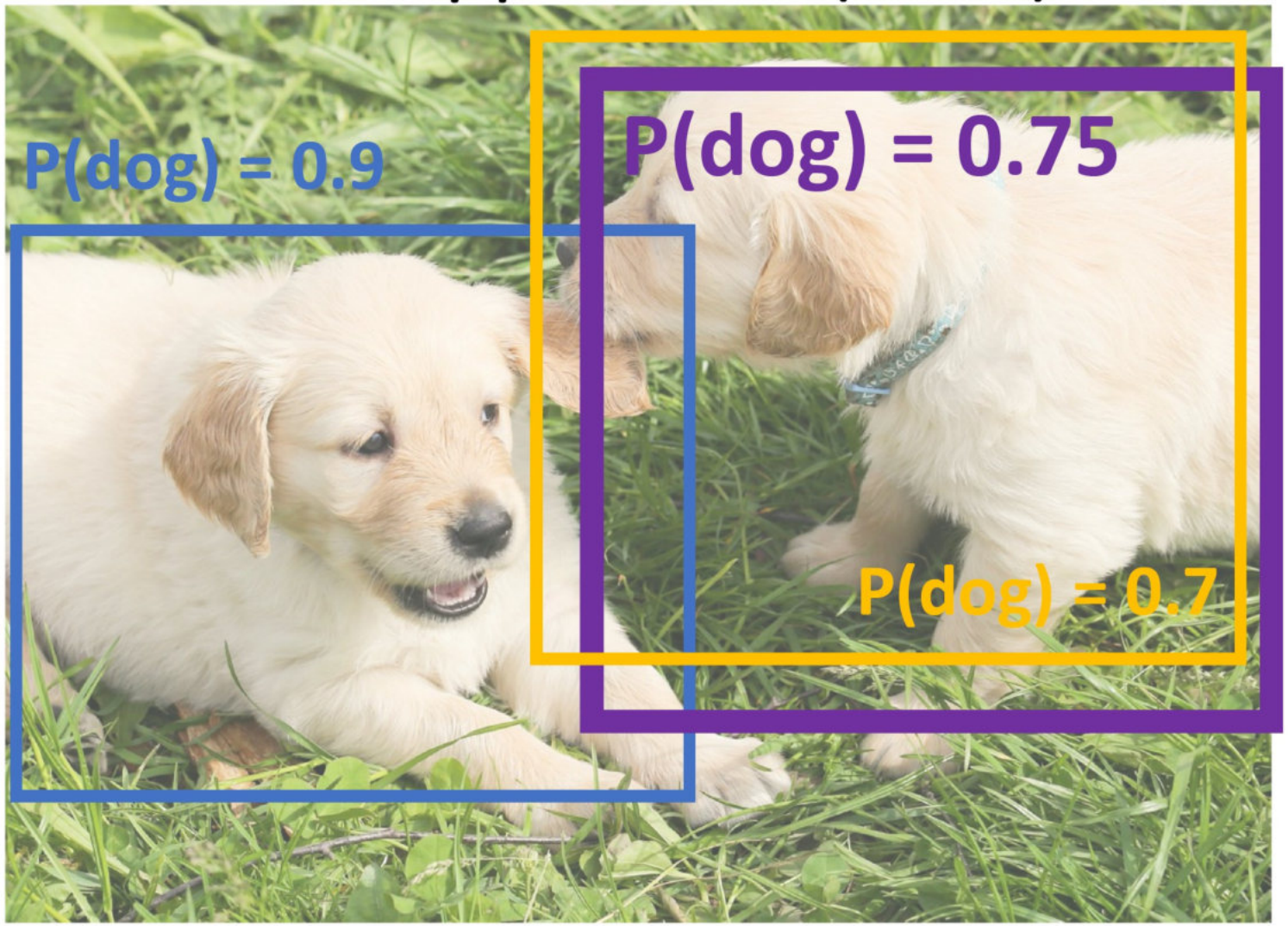
Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections:

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

IoU(■, ■) = **0.74**

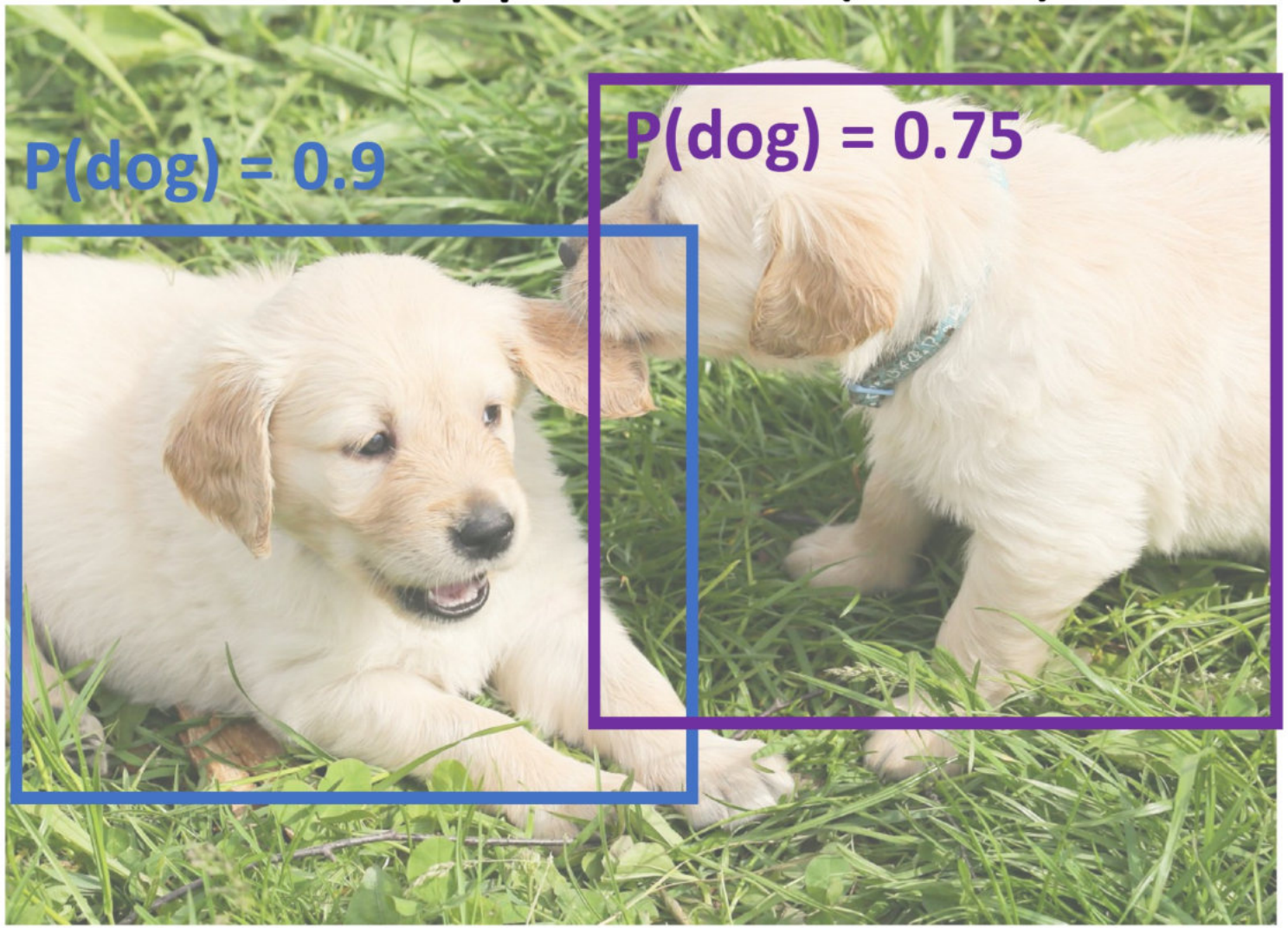


Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections:

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1



Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections:

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

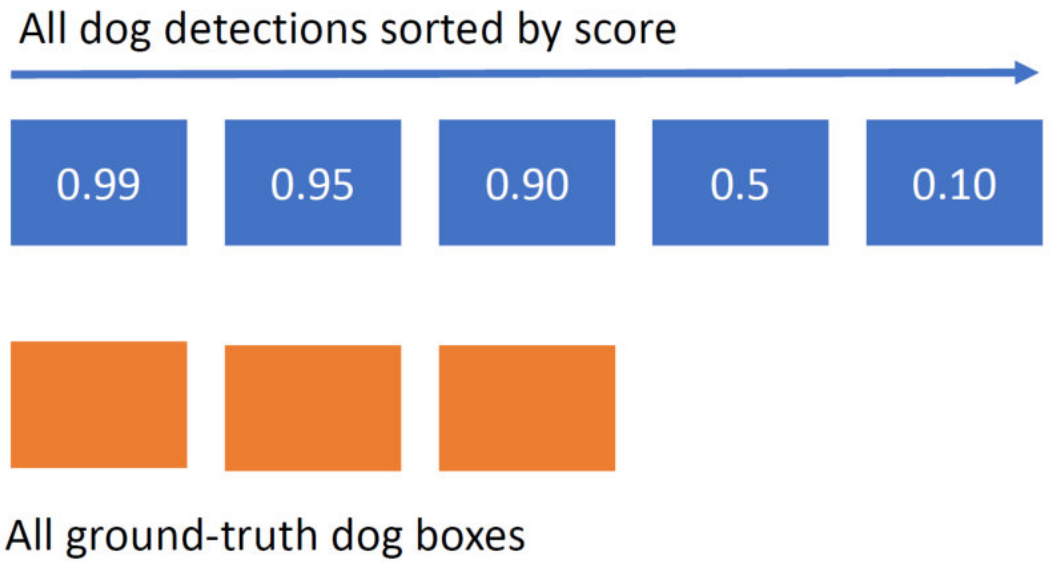
1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

Problem: NMS may eliminate ”good” boxes when objects are highly overlapping… no good solution =(

Slide from Deep Learning for Computer Vision @Univ. of Michigan

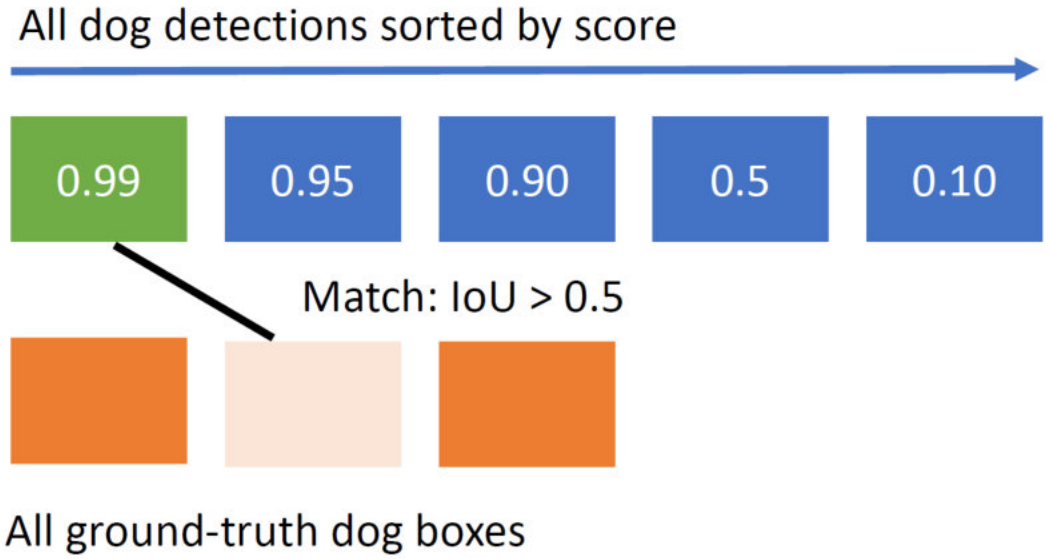
Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)



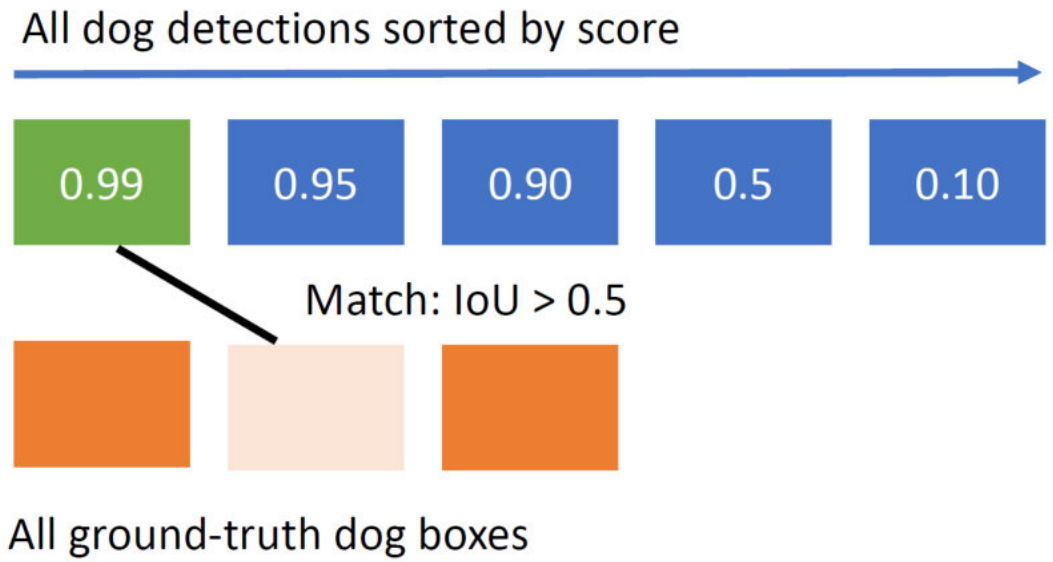
Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative

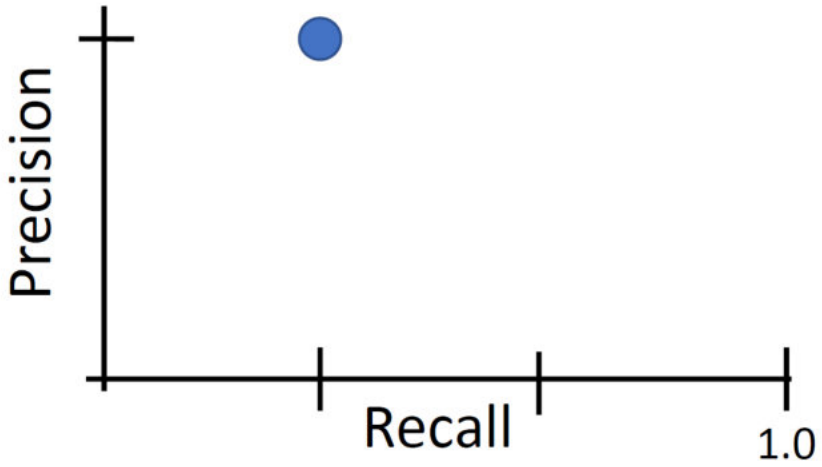


Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve



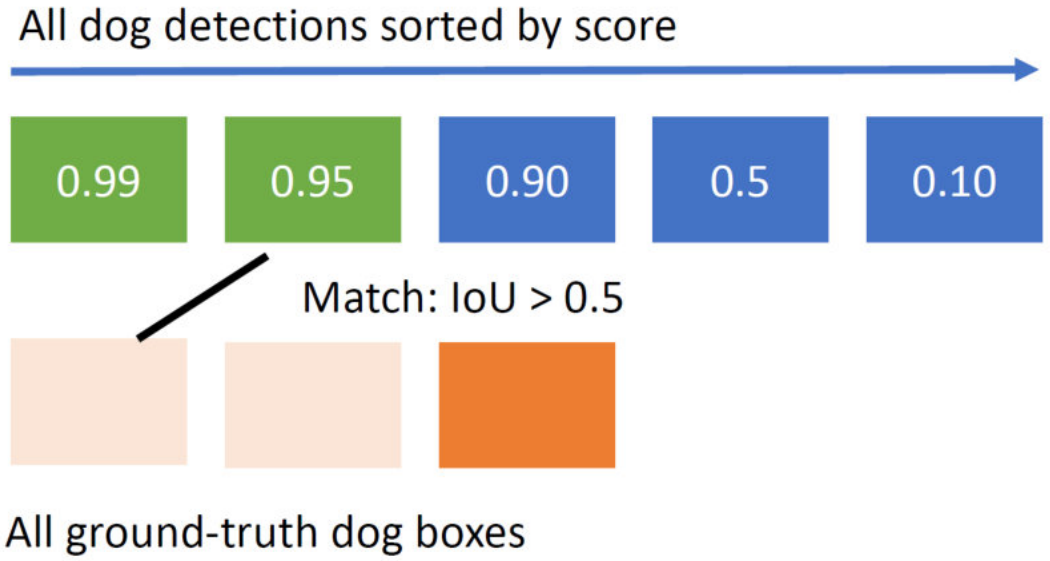
Precision = 1/1 = 1.0
Recall = 1/3 = 0.33

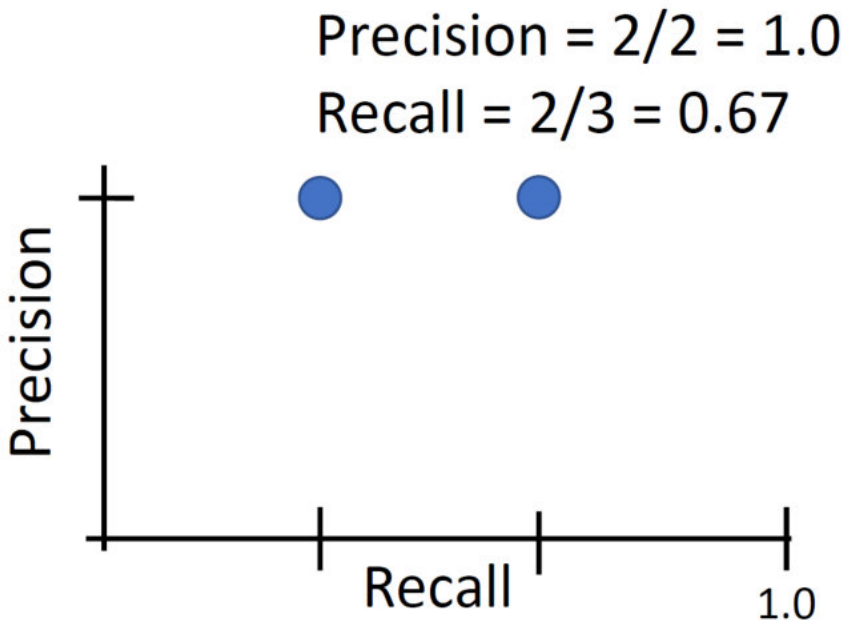


Slide from Deep Learning for Computer Vision @Univ. of Michigan

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve

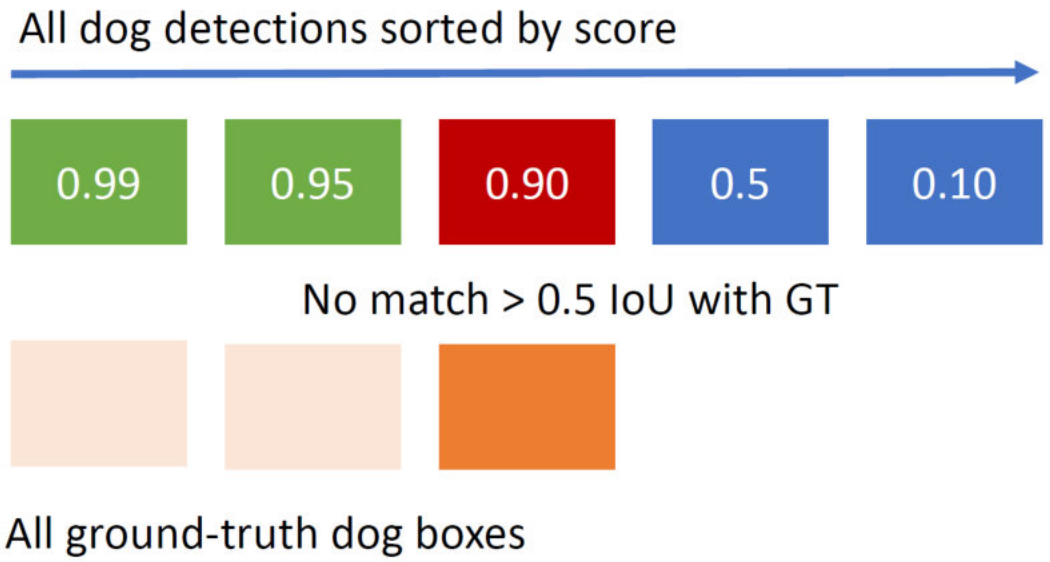


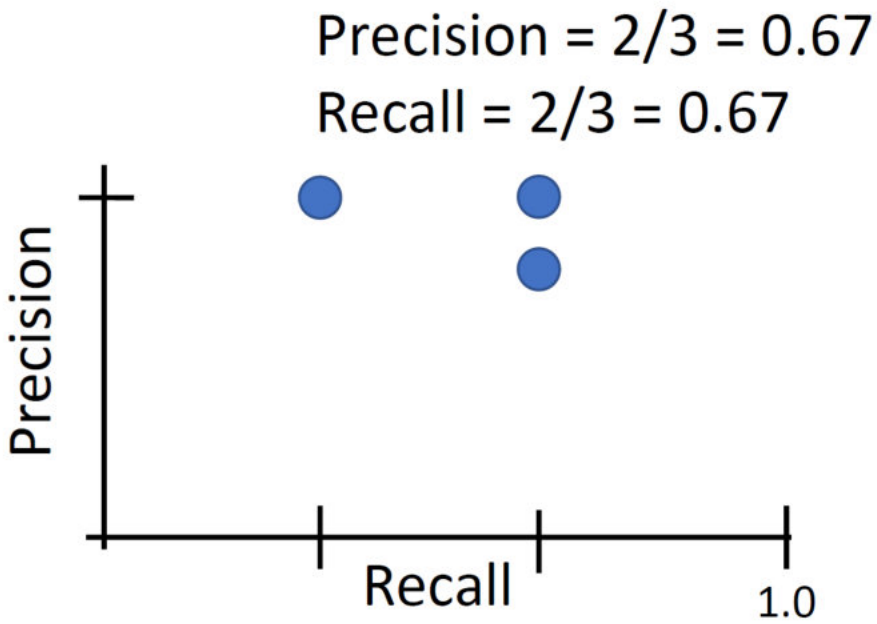


Slide from Deep Learning for Computer Vision @Univ. of Michigan

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve

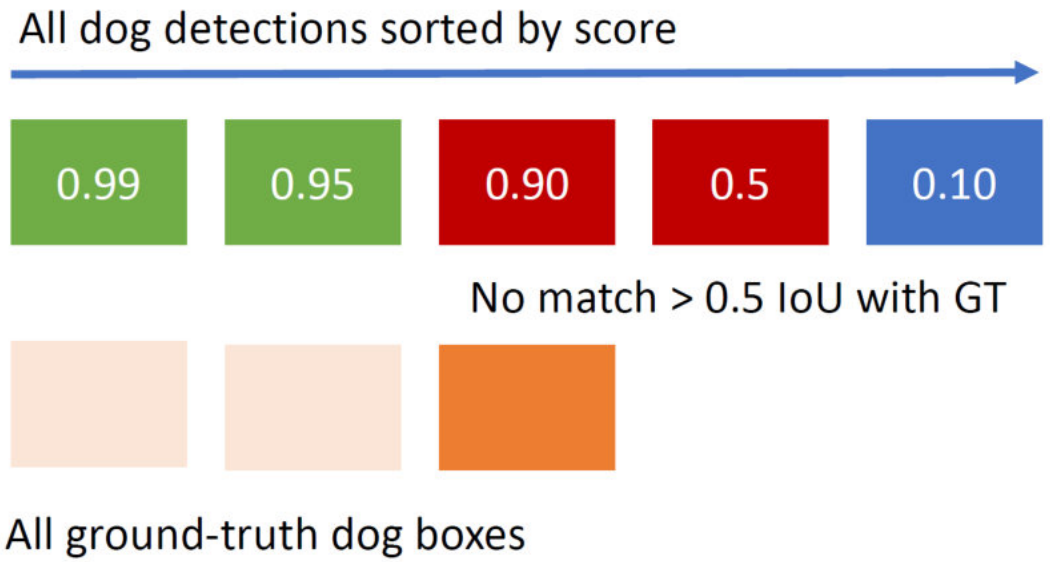


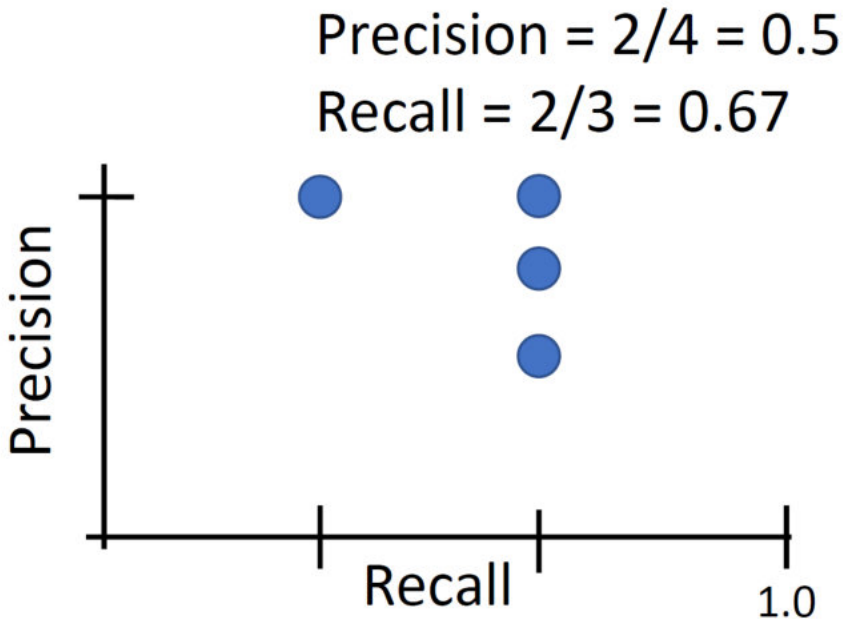


Slide from Deep Learning for Computer Vision @Univ. of Michigan

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve

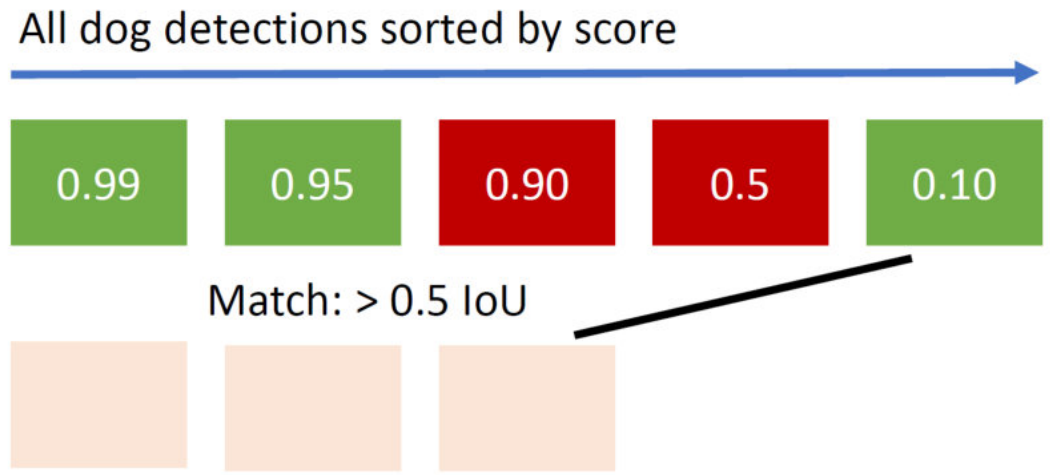


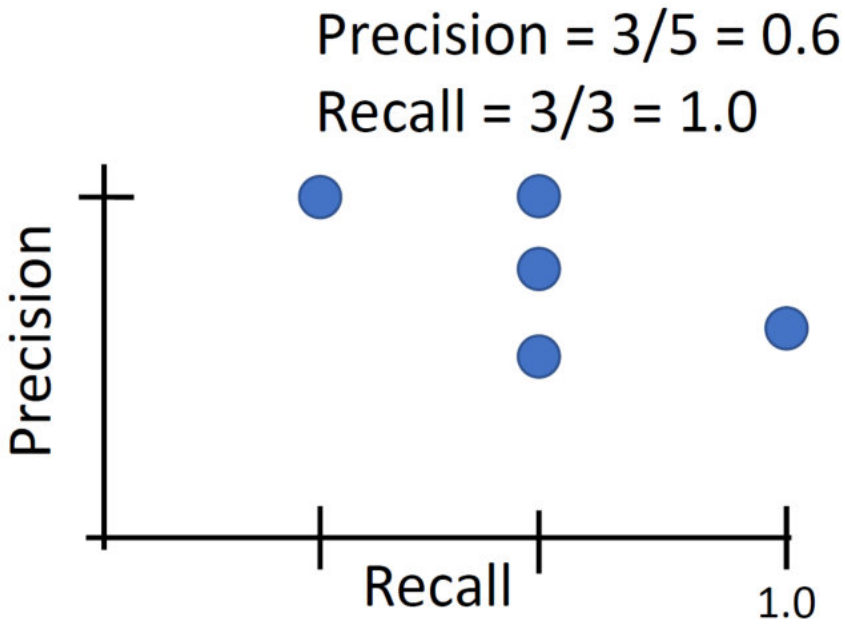


Slide from Deep Learning for Computer Vision @Univ. of Michigan

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve



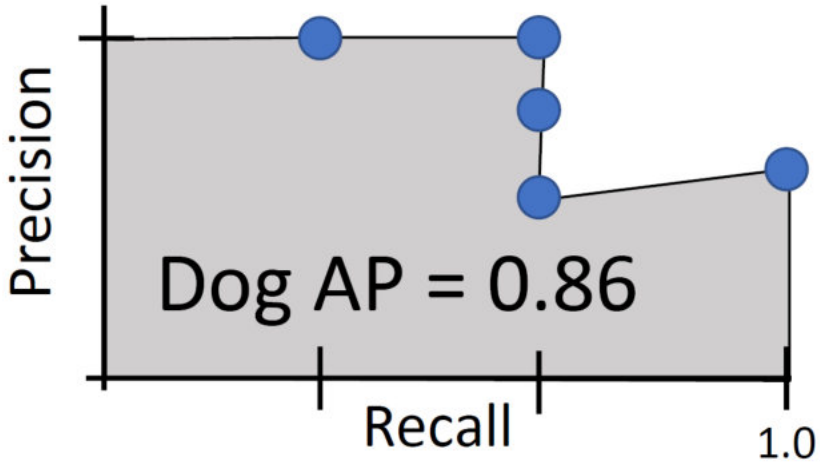


Slide from Deep Learning for Computer Vision @Univ. of Michigan

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve
    2. Average Precision (AP) = area under PR curve

How to get AP = 1.0: Hit all GT boxes with IoU > 0.5, and have no "false positive" detections ranked above any "true positives"



Slide from Deep Learning for Computer Vision @Univ. of Michigan

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve
    2. Average Precision (AP) = area under PR curve
3. Mean Average Precision (mAP) = average of AP for each category

Car AP = 0.65

Cat AP = 0.80

Dog AP = 0.86

mAP@0.5 = 0.77

Slide from Deep Learning for Computer Vision @Univ. of Michigan

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve
    2. Average Precision (AP) = area under PR curve
3. Mean Average Precision (mAP) = average of AP for each category
4. For "COCO mAP": Compute mAP@thresh for each IoU threshold (0.5, 0.55, 0.6, …, 0.95) and take average

mAP@0.5 = 0.77

mAP@0.55 = 0.71

mAP@0.60 = 0.65

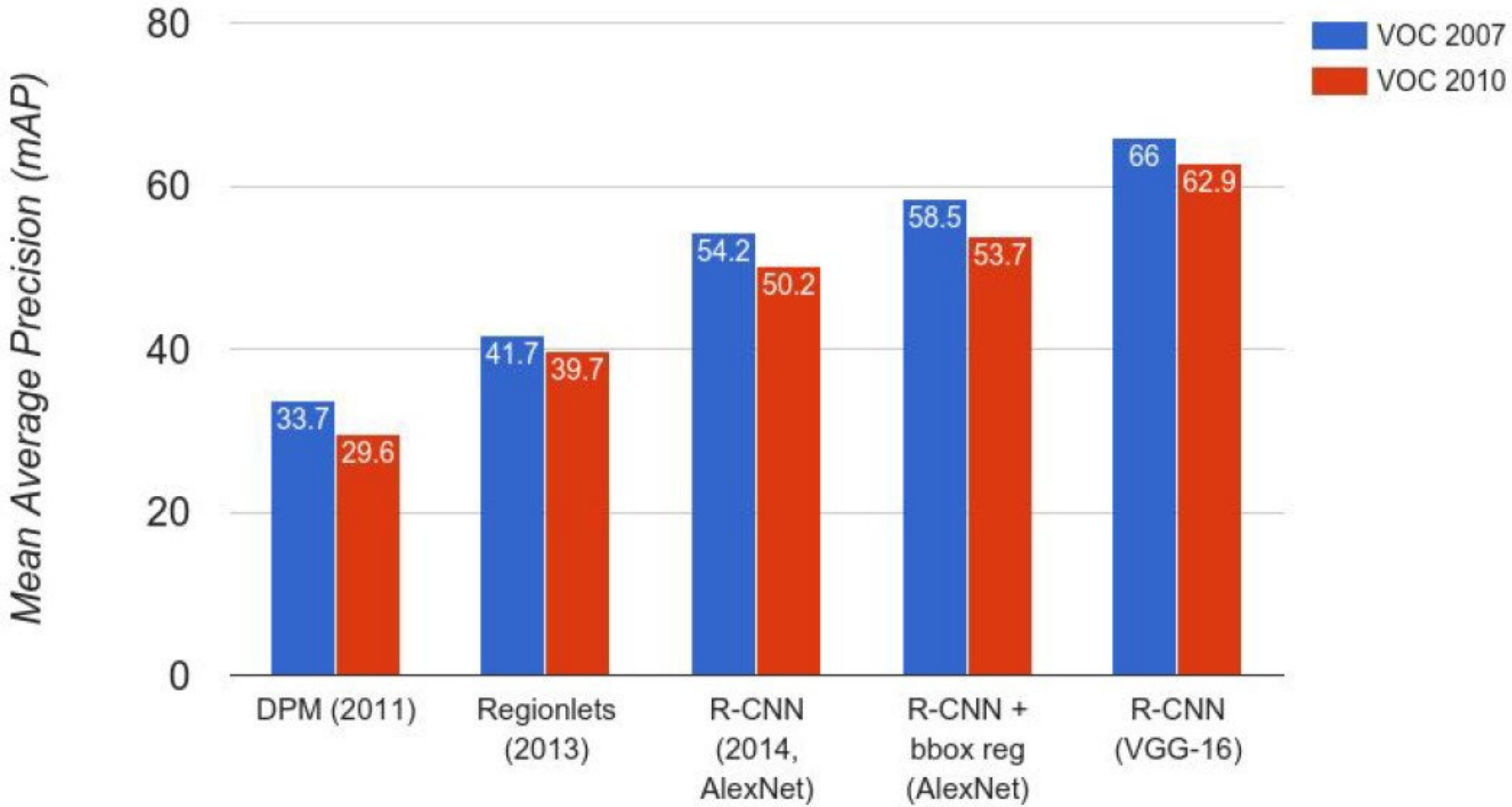
…

mAP@0.95 = 0.2

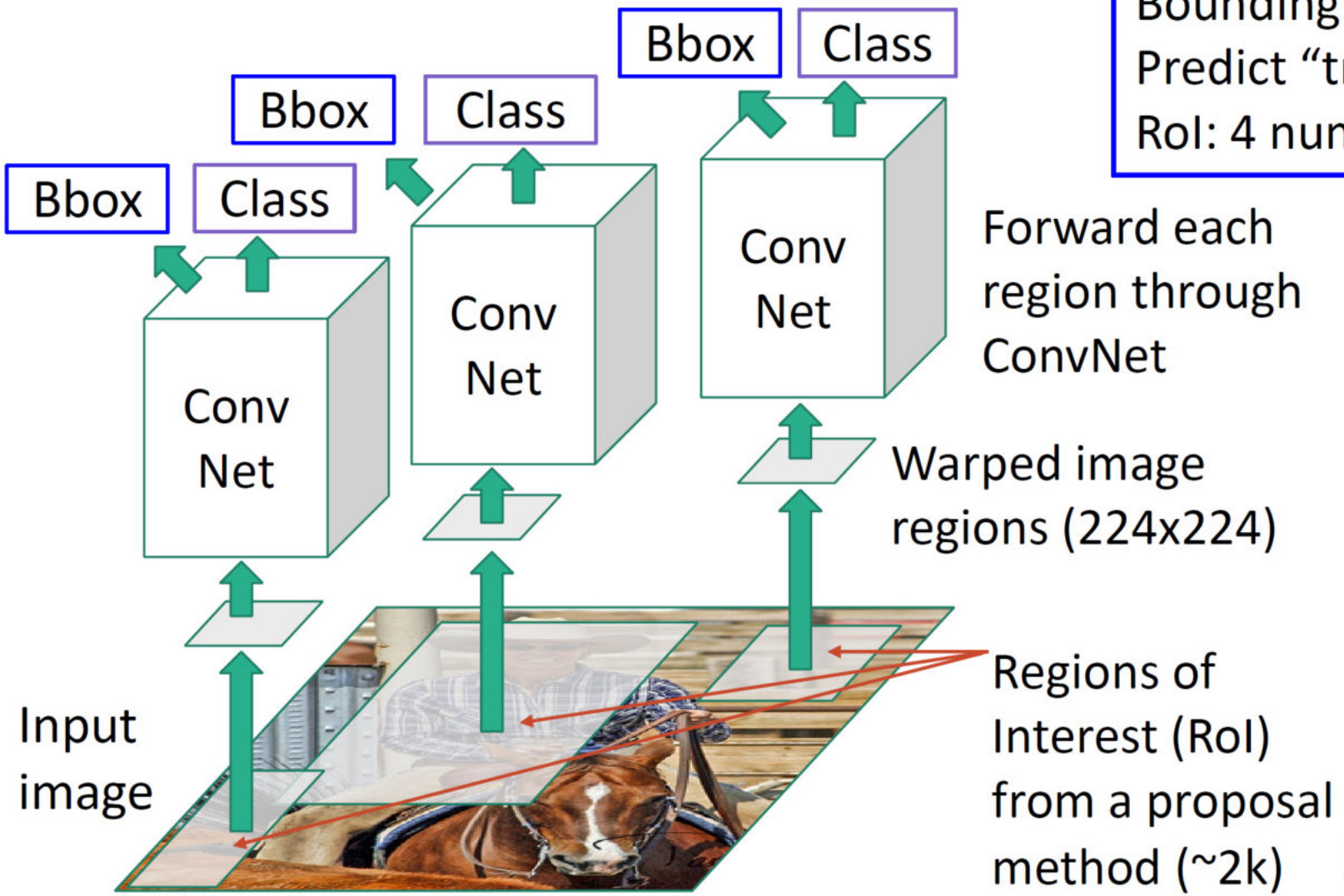
COCO mAP = 0.4

Slide from Deep Learning for Computer Vision @Univ. of Michigan

Results


Mean Average Precision (mAP)
80
60
40
20
0
VOC 2007
VOC 2010
33.7
29.6
41.7
39.7
54.2
50.2
58.5
53.7
66
62.9
DPM (2011)
Regionlets (2013)
R-CNN (2014, AlexNet)
R-CNN + bbox reg (AlexNet)
R-CNN (VGG-16)

R-CNN: Region-Based CNN



Classify each region

Bounding box regression:
Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)

Forward each region through ConvNet

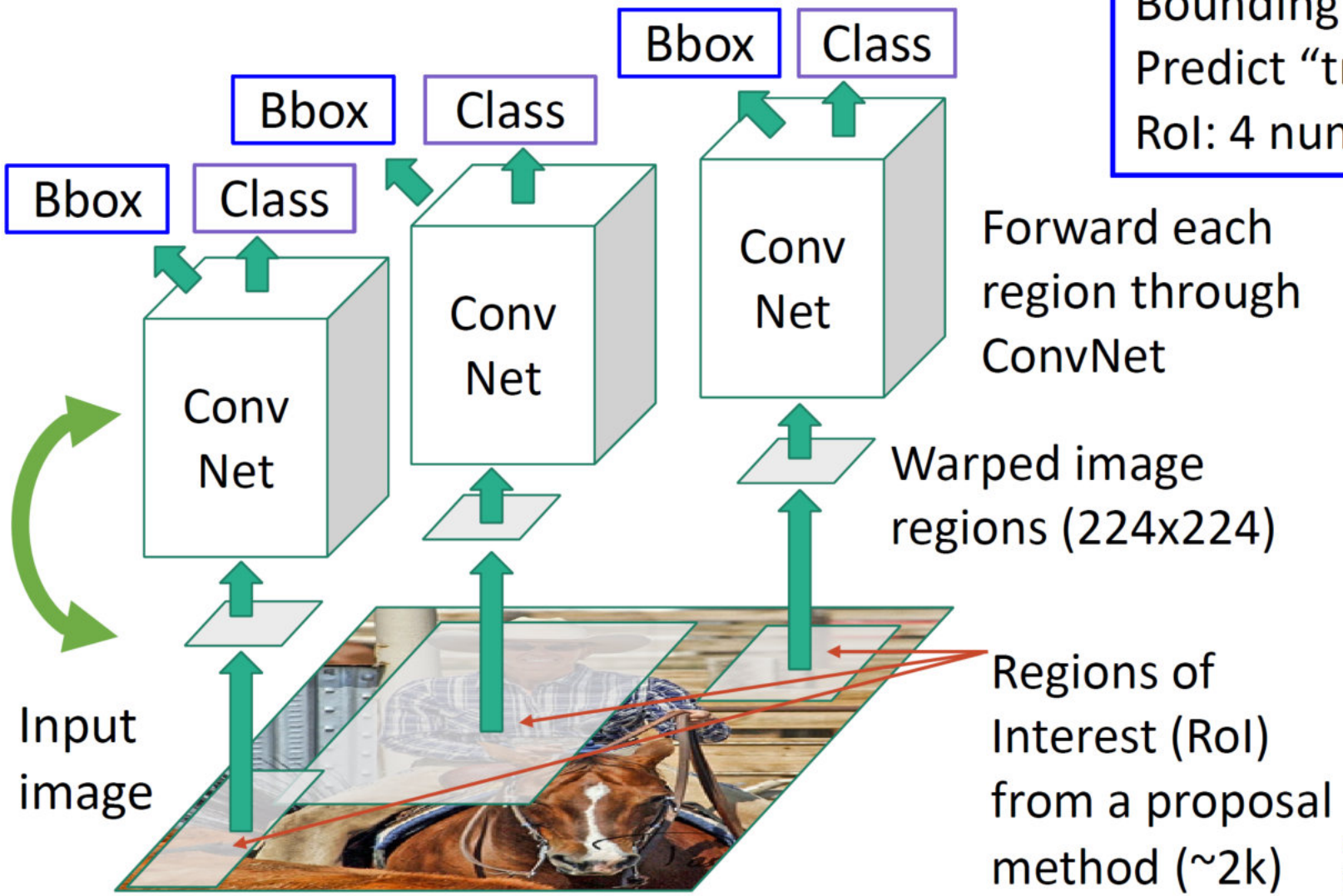
Warped image regions (224x224)

Regions of Interest (RoI) from a proposal method (~2k)

Problem: Very slow!
Need to do ~2k forward passes for each image!

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

R-CNN: Region-Based CNN



Classify each region

Bounding box regression:
Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)

Forward each region through ConvNet

Warped image regions (224x224)

Regions of Interest (RoI) from a proposal method (~2k)

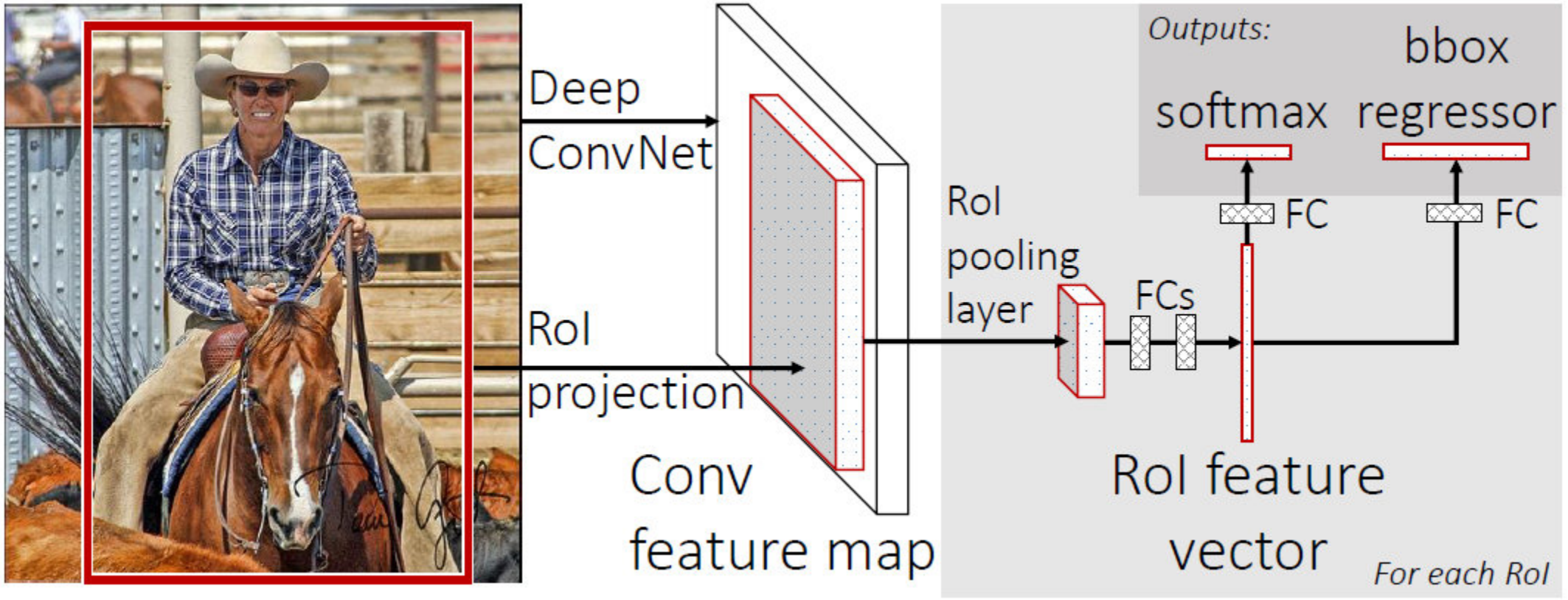
Problem: Very slow!
Need to do ~2k forward passes for each image!

Solution: Run CNN *before* warping!

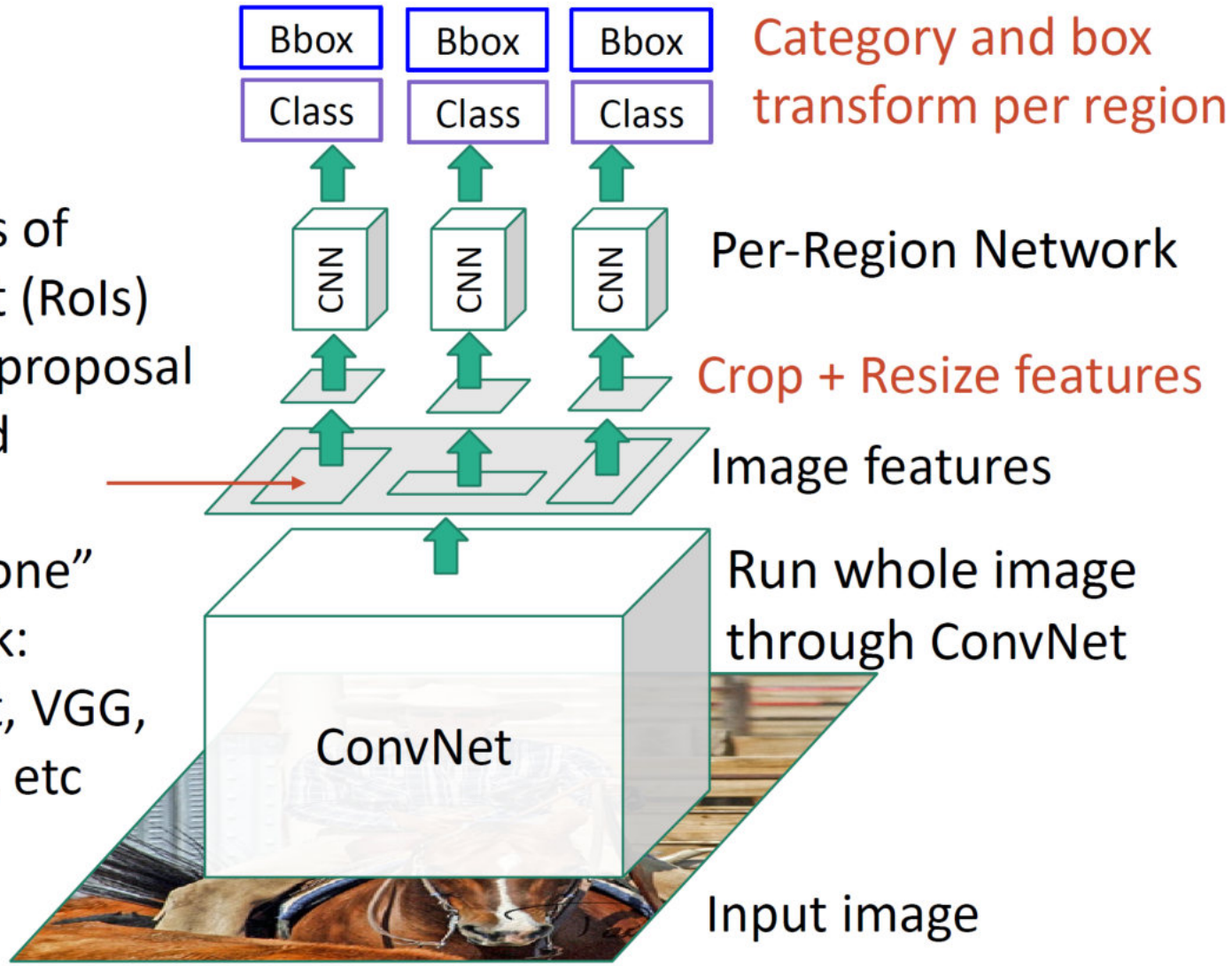
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Slide from Deep Learning for Computer Vision @Univ. of Michigan

Fast R-CNN Architecture



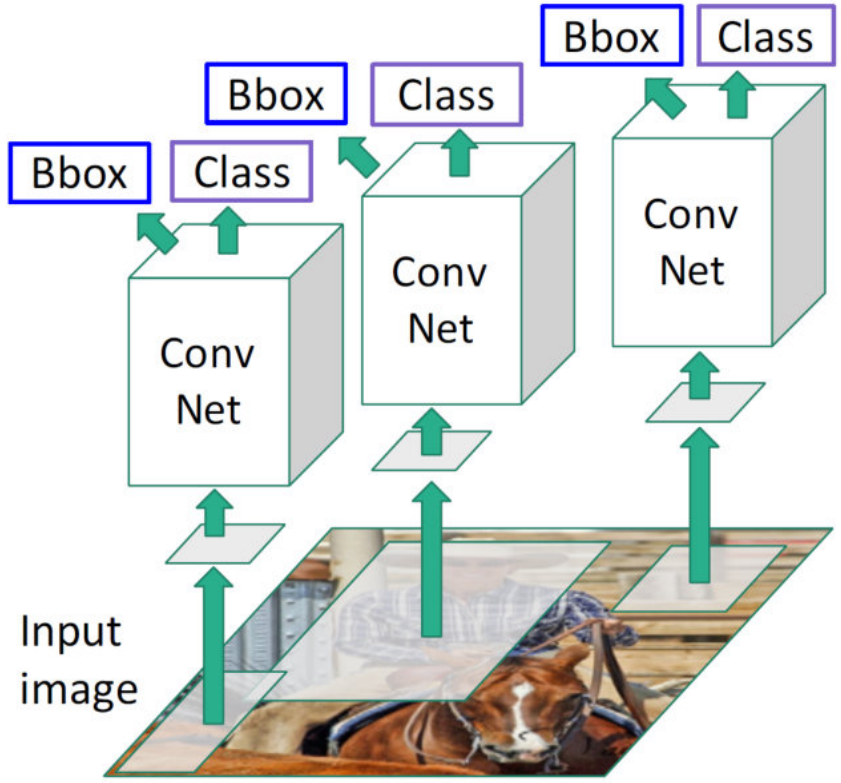
Fast R-CNN



Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

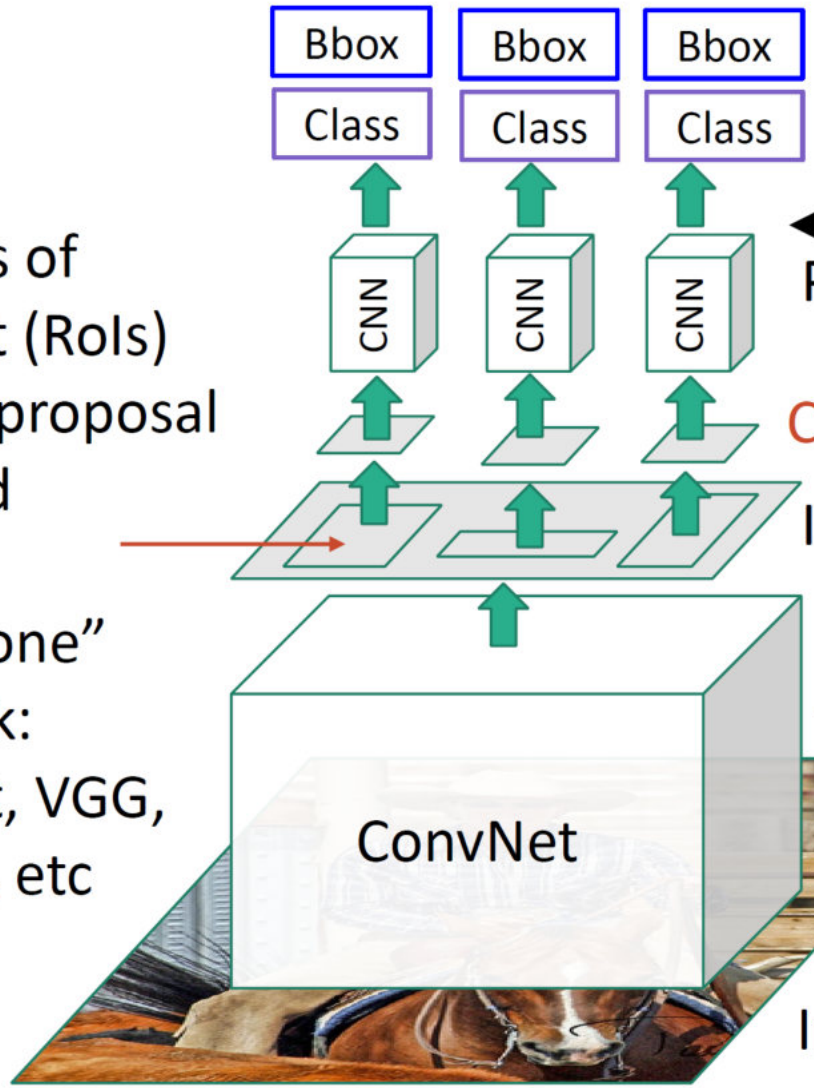
"Slow" R-CNN

Process each region independently



Slide from Deep Learning for Computer Vision @Univ. of Michigan

Fast R-CNN



Category and box transform per region

Per-Region network is relatively lightweight

Per-Region Network

Regions of Interest (RoIs) from a proposal method

Crop + Resize features

Image features

"Backbone" network: AlexNet, VGG, ResNet, etc

Run whole image through ConvNet

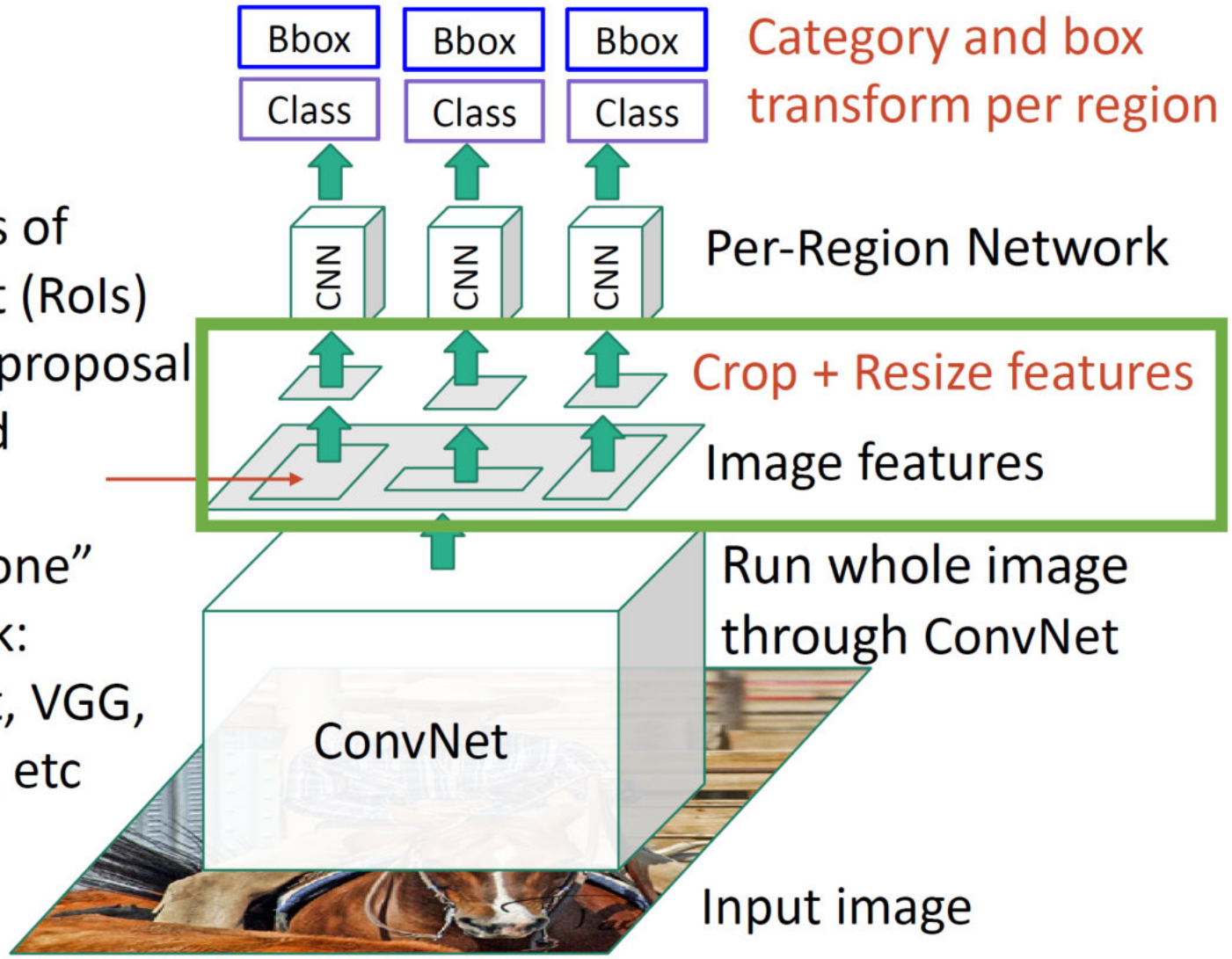
ConvNet

Most of the computation happens in backbone network; this saves work for overlapping region proposals

Input image

Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN



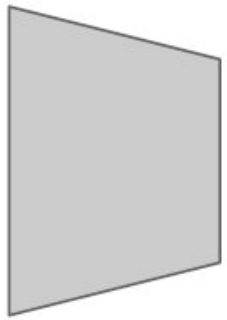
Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

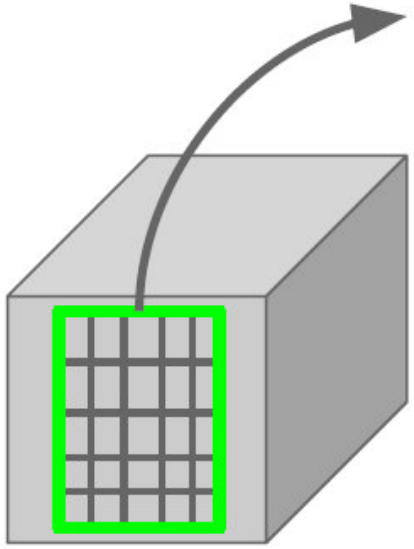
Slide from Deep Learning for Computer Vision @Univ. of Michigan

RoI Pooling

Hi-res input image:
3 x 800 x 600
with region
proposal

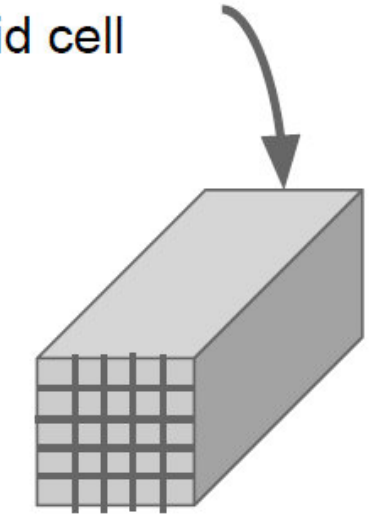
Convolution
and Pooling





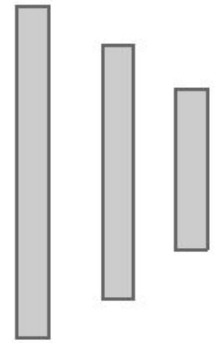
Hi-res conv features:
C x H x W
with region proposal

Max-pool within
each grid cell



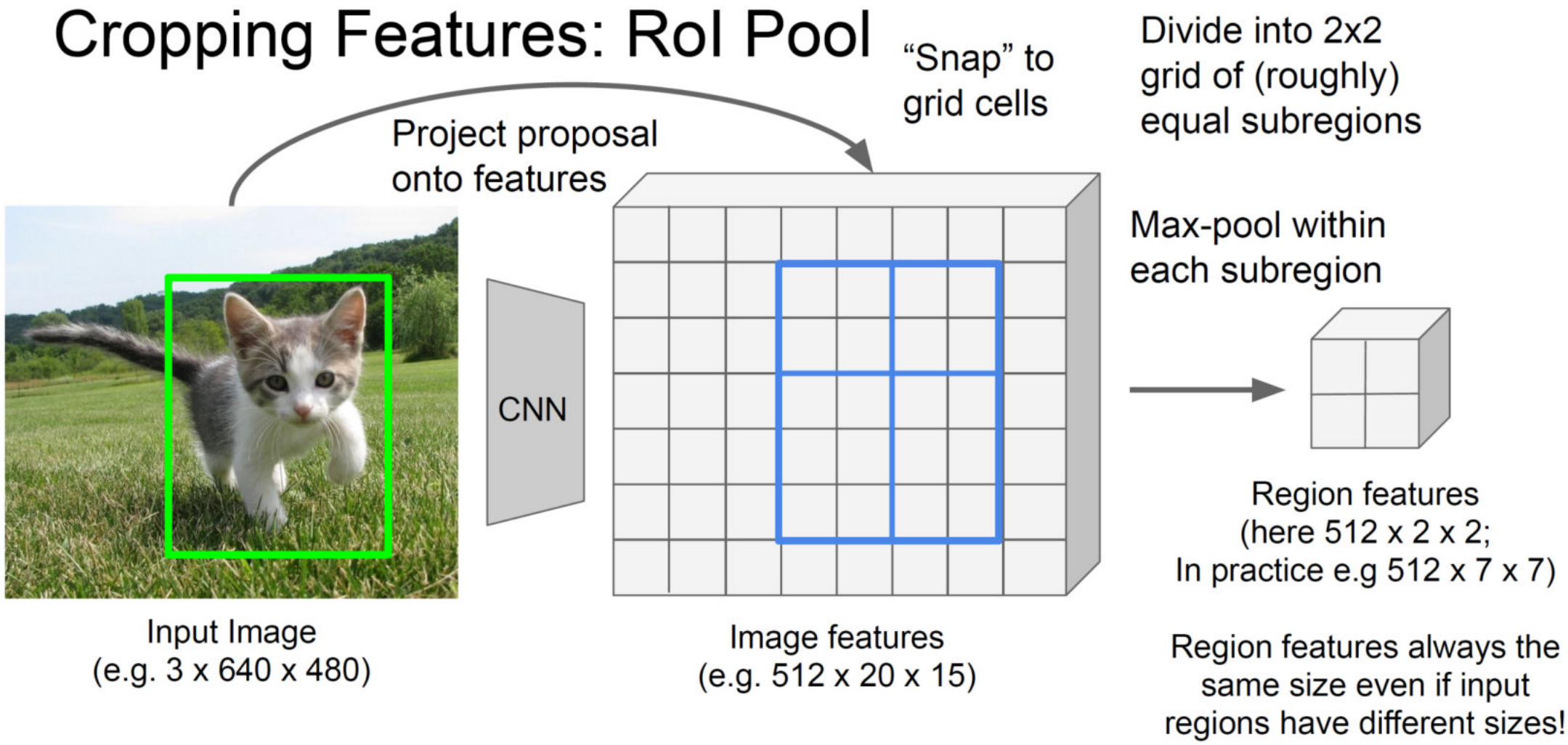
RoI conv features:
C x h x w
for region proposal

Fully-connected
layers



Fully-connected layers expect
low-res conv features:
C x h x w

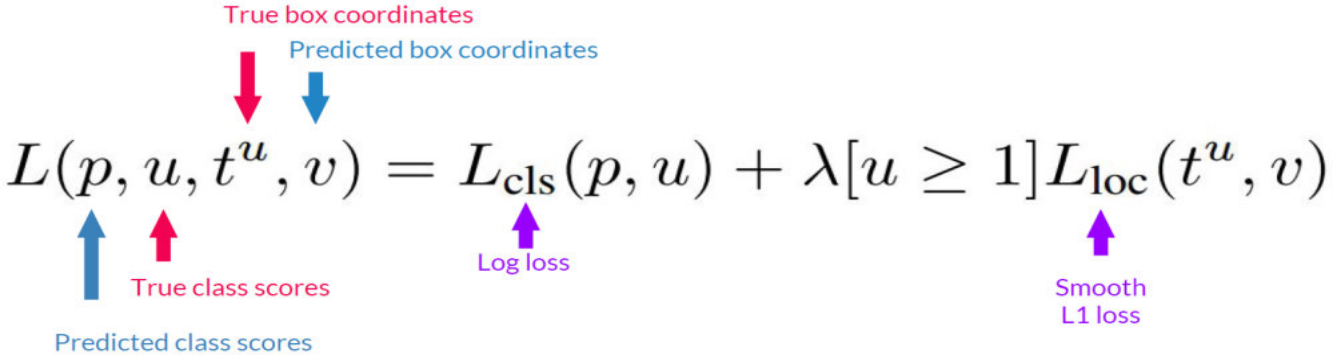
RoI Pooling


Cropping Features: RoI Pool
"Snap" to grid cells
Project proposal onto features
Divide into 2x2 grid of (roughly) equal subregions
Max-pool within each subregion
CNN
Input Image
(e.g. 3 x 640 x 480)
Image features
(e.g. 512 x 20 x 15)
Region features
(here 512 x 2 x 2;
In practice e.g 512 x 7 x 7)
Region features always the same size even if input regions have different sizes!

Training & Testing

1. Takes an input and a set of bounding boxes
2. Generate convolutional feature maps
3. For each bbox, get a fixed-length feature vector from RoI pooling layer
4. Outputs have two information
   - K+1 class labels
   - Bounding box locations

- Loss function

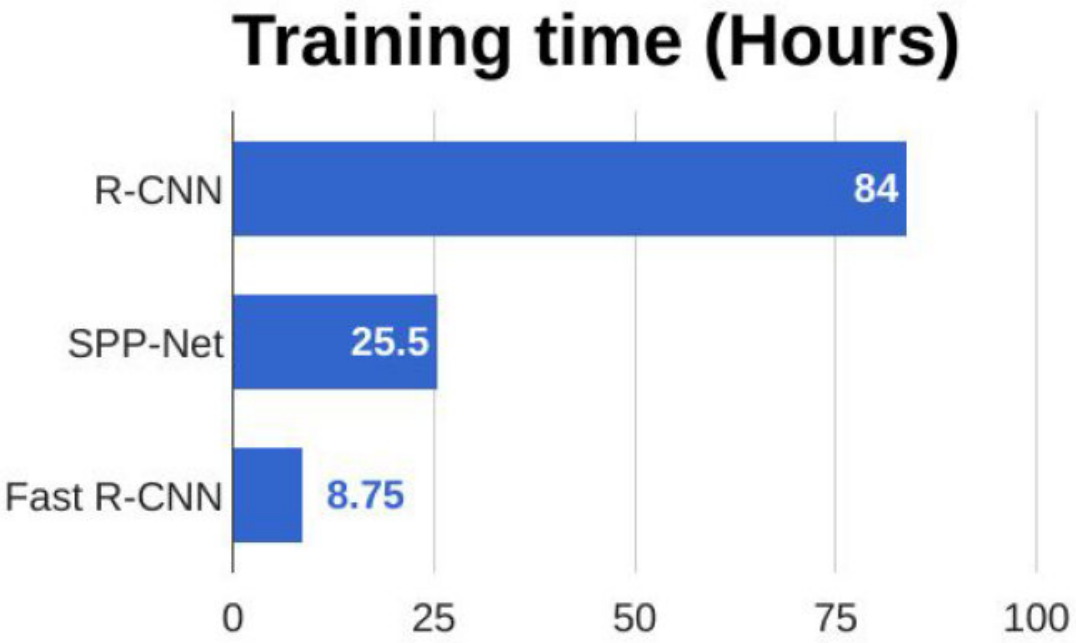


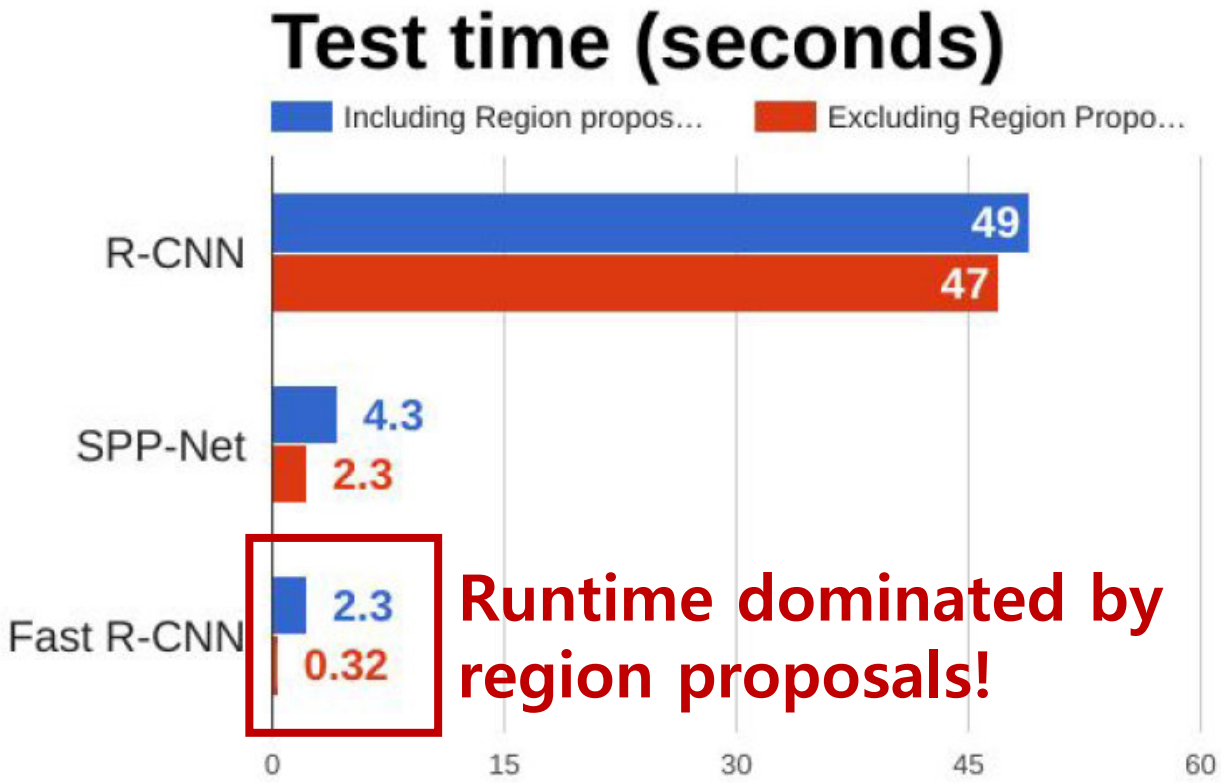
$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}(t_i^u - v_i),$$

in which

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

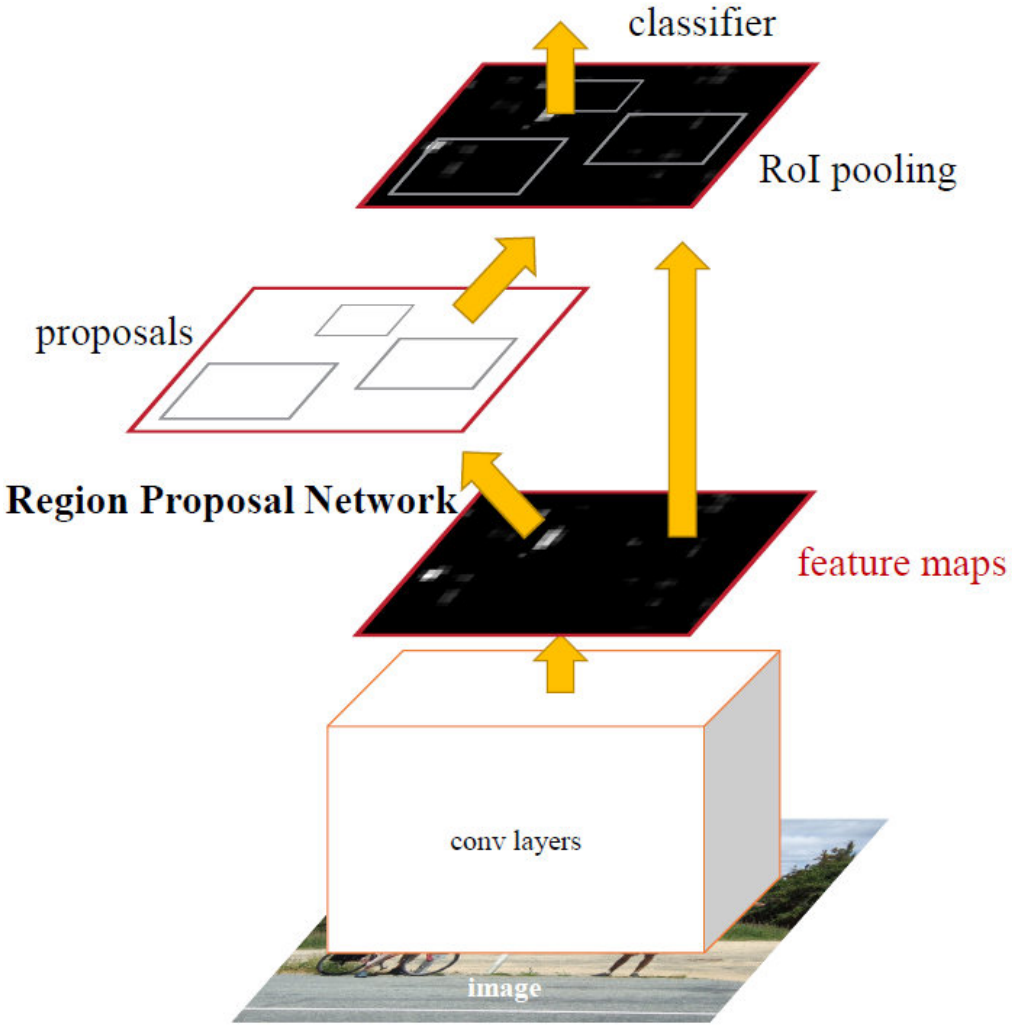
R-CNN vs SPP-net vs Fast R-CNN





Faster R-CNN(RPN + Fast R-CNN)

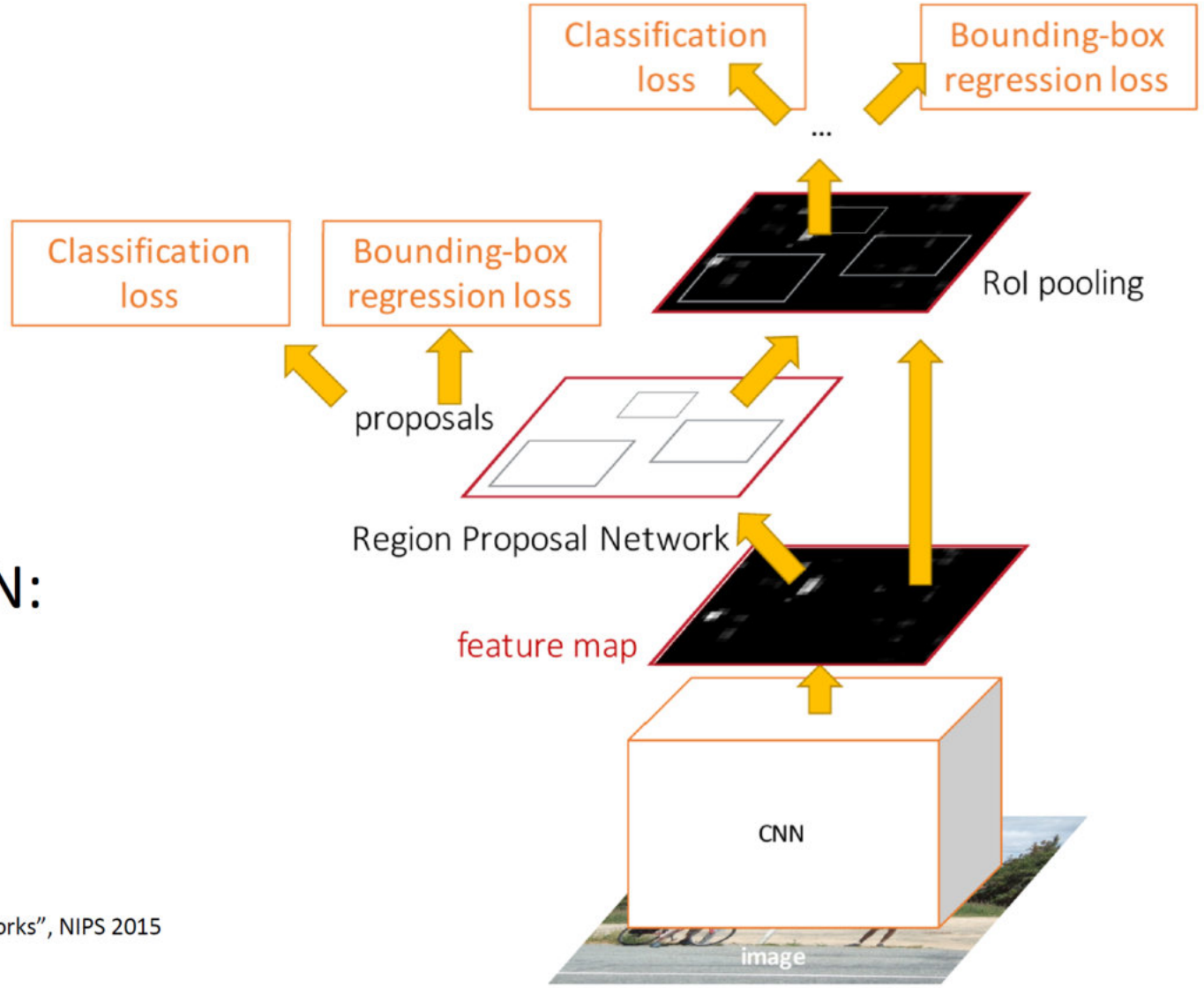
- Insert a Region Proposal Network (RPN) after the last convolutional layer → using GPU!
- RPN trained to produce region proposals directly; no need for external region proposals
- After RPN, use RoI Pooling and an upstream classifier and bbox regressor just like Fast R-CNN



Faster R-CNN: Learnable Region Proposals

Insert **Region Proposal Network (RPN)** to predict proposals from features

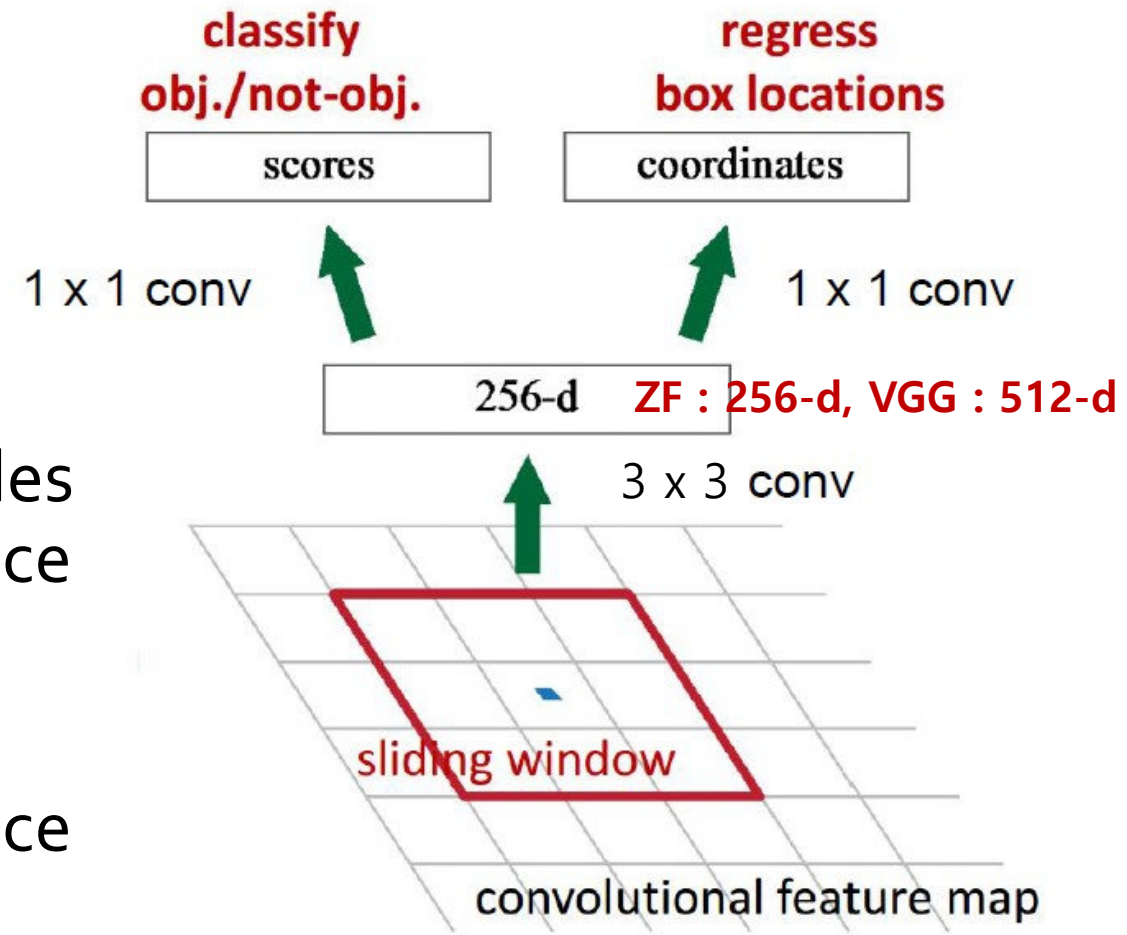
Otherwise same as Fast R-CNN: Crop features for each proposal, classify each one



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

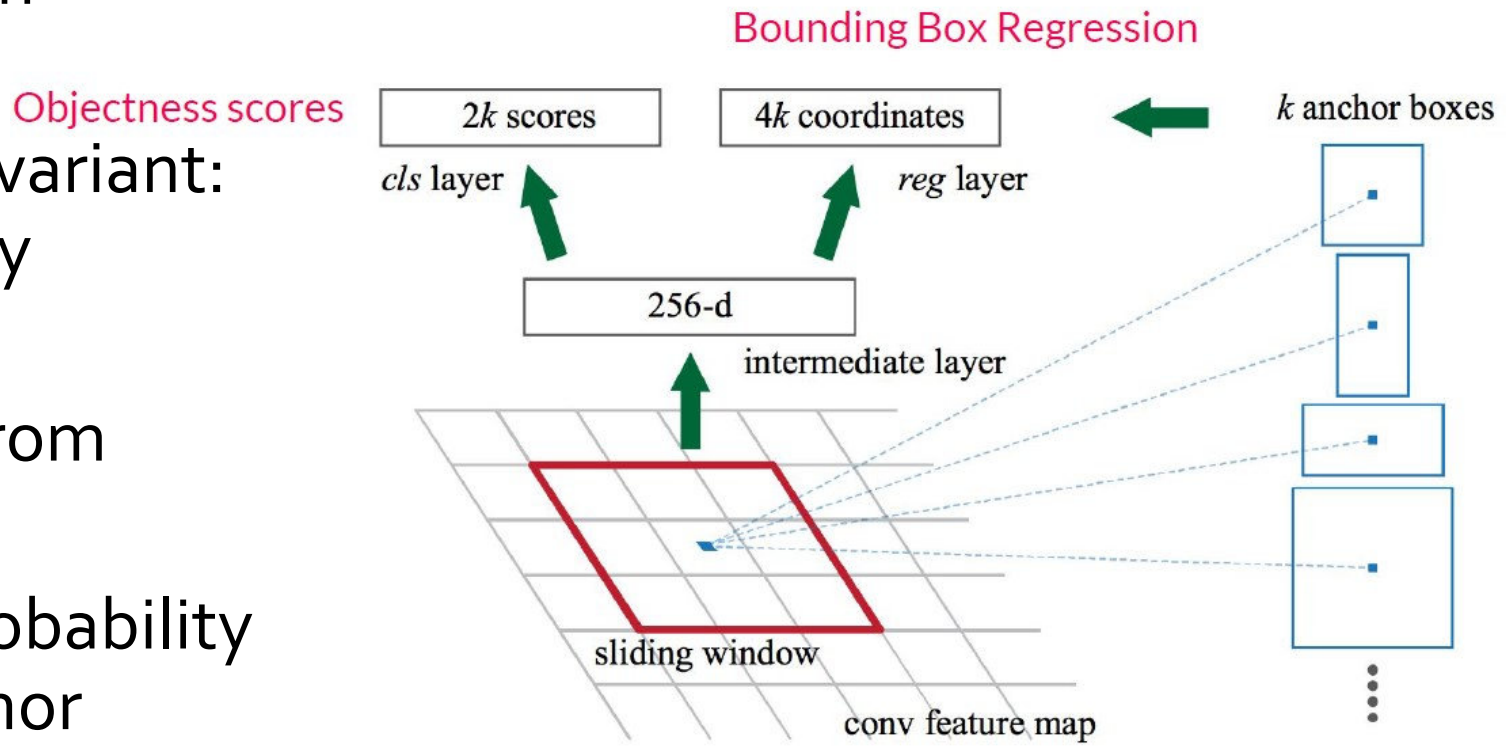
RPN

- Slide a small window on the feature map
- Build a small network for
  - Classifying object or not-object
  - Regressing bbox locations
- Position of the sliding window provides localization information with reference to the image
- Box regression provides finer localization information with reference to this sliding window



RPN

- Use k anchor boxes at each location
- Anchors are translation invariant: use the same ones at every location
- Regression gives offsets from anchor boxes
- Classification gives the probability that each (regressed) anchor shows an object



Region Proposal Network (RPN)

Run backbone CNN to get
features aligned to input image

Input Image
(e.g. 3 x 640 x 480)

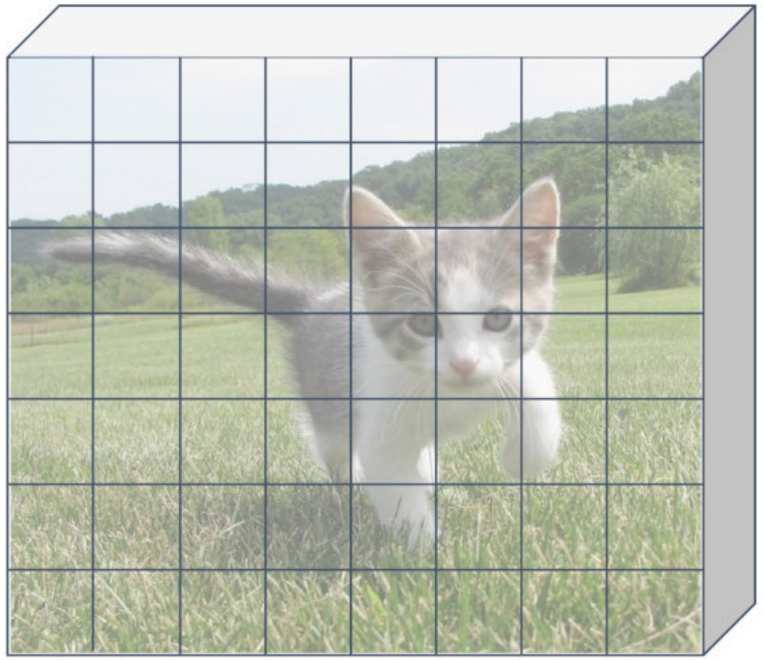


Image features
(e.g. 512 x 20 x 15)

Slide from Deep Learning for Computer Vision @Univ. of Michigan

Region Proposal Network (RPN)

Imagine an **anchor box** of fixed size at each point in the feature map

Run backbone CNN to get features aligned to input image

Input Image
(e.g. 3 x 640 x 480)

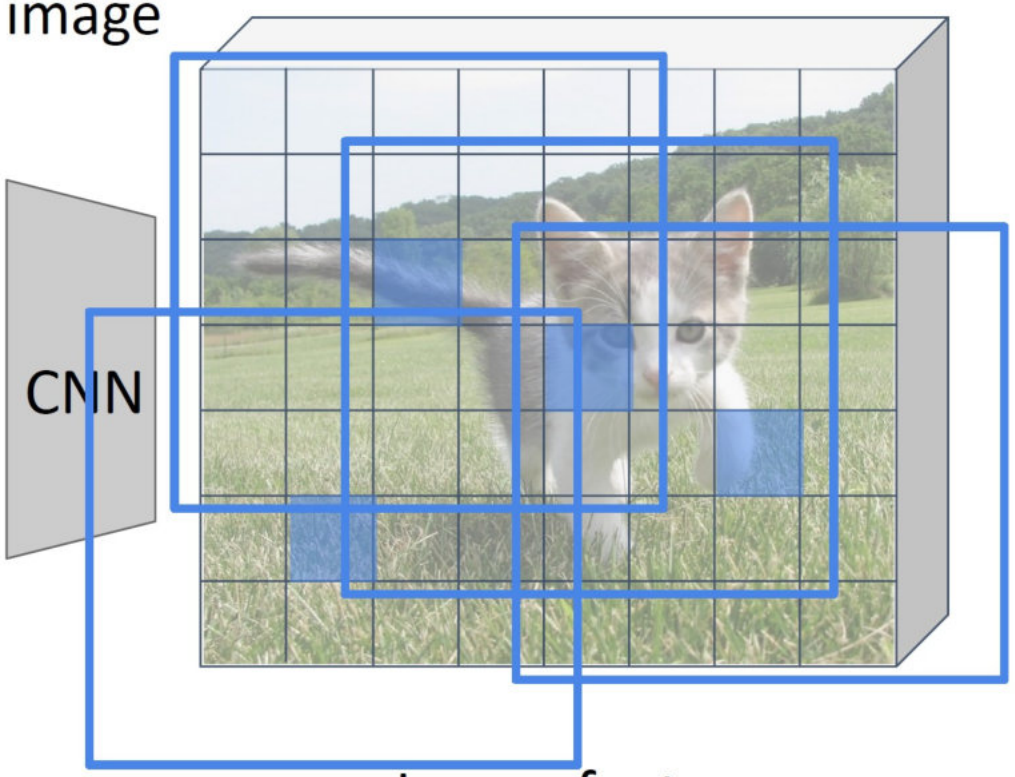


Image features
(e.g. 512 x 20 x 15)

Slide from Deep Learning for Computer Vision @Univ. of Michigan

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image

Input Image
(e.g. 3 x 640 x 480)

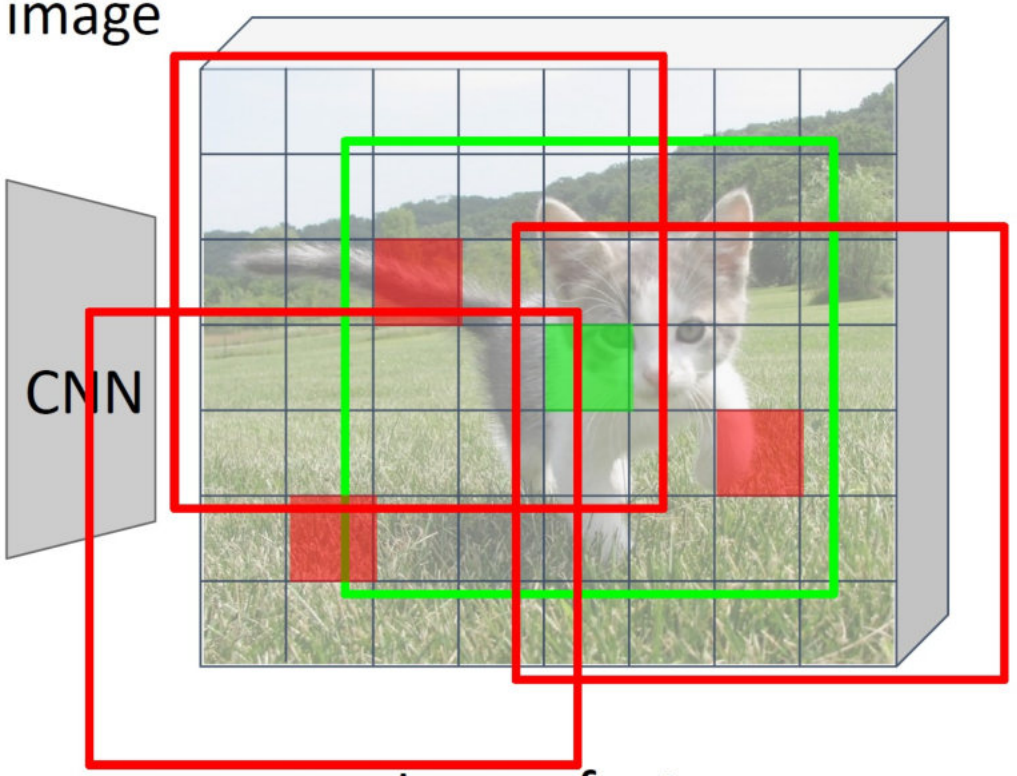


Image features
(e.g. 512 x 20 x 15)

Imagine an anchor box of fixed size at each point in the feature map

Anchor is an object?
1 x 20 x 15

At each point, predict whether the corresponding anchor contains an object (per-cell logistic regression, predict scores with conv layer)

Slide from Deep Learning for Computer Vision @Univ. of Michigan

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image

Input Image
(e.g. 3 x 640 x 480)

CNN

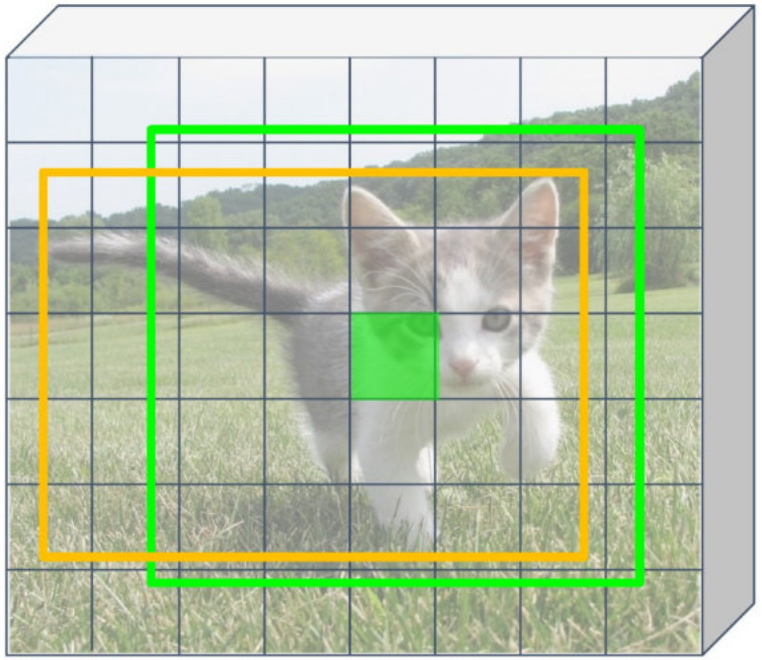


Image features
(e.g. 512 x 20 x 15)

Imagine an anchor box of fixed size at each point in the feature map

Conv

Anchor is an object?
1 x 20 x 15

Box transforms
4 x 20 x 15

For positive boxes, also predict a box transform to regress from **anchor box** to **object box**

Slide from Deep Learning for Computer Vision @Univ. of Michigan

Region Proposal Network (RPN)

Problem: Anchor box may have the wrong size / shape

Solution: Use **K** different anchor boxes at each point!

Run backbone CNN to get features aligned to input image

Input Image
(e.g. 3 x 640 x 480)

CNN

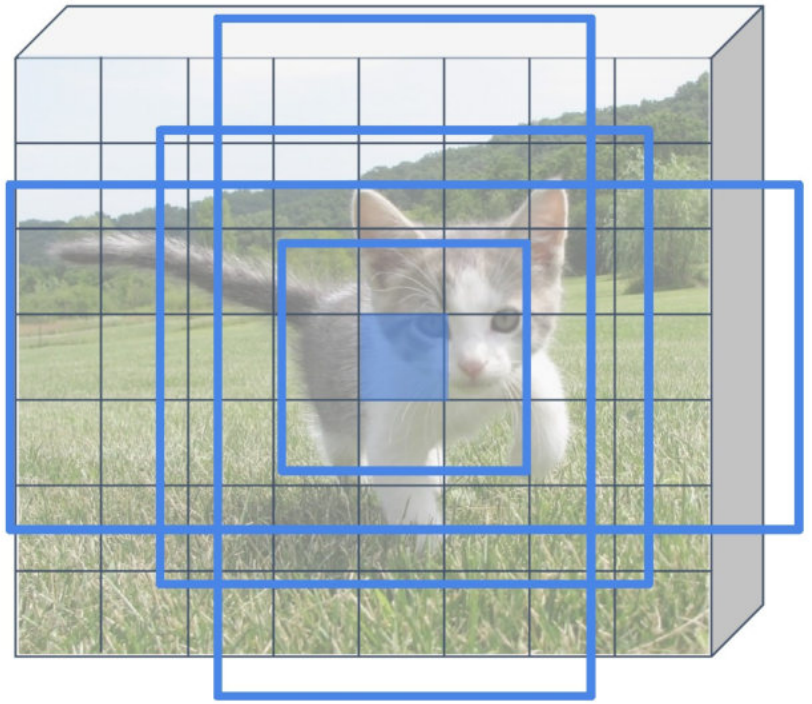


Image features
(e.g. 512 x 20 x 15)

Conv

Anchor is an object?
K x 20 x 15

Box transforms
4K x 20 x 15

At test time: sort all K*20*15 boxes by their score, and take the top ~300 as our region proposals

Slide from Deep Learning for Computer Vision @Univ. of Michigan

Anchors as references

- Anchors: pre-defined reference boxes
- Multi-scale/size anchors:
  - Multiple anchors are used at each position:
    - 3 scale(128x128, 256x256, 512x512) and 3 aspect rations(2:1, 1:1, 1:2) yield 9 anchors
  - Each anchor has its own prediction function
  - Single-scale features, multi-scale predictions

Positive/Negative Samples

- An anchor is labeled as positive if
  - The anchor is the one with highest IoU overlap with a ground-truth box
  - The anchor has an IoU overlap with a ground-truth box higher than 0.7
- Negative labels are assigned to anchors with IoU lower than 0.3 for all ground-truth boxes
- 50%/50% ratio of positive/negative anchors in a minibatch

RPN Loss Function

i = anchor index in minibatch

Log loss

Ground truth objectness label

Smooth L1 loss

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

True box coordinates

Coordinates of the predicted bounding box for anchor *i*

Predicted probability of being an object for anchor *i*

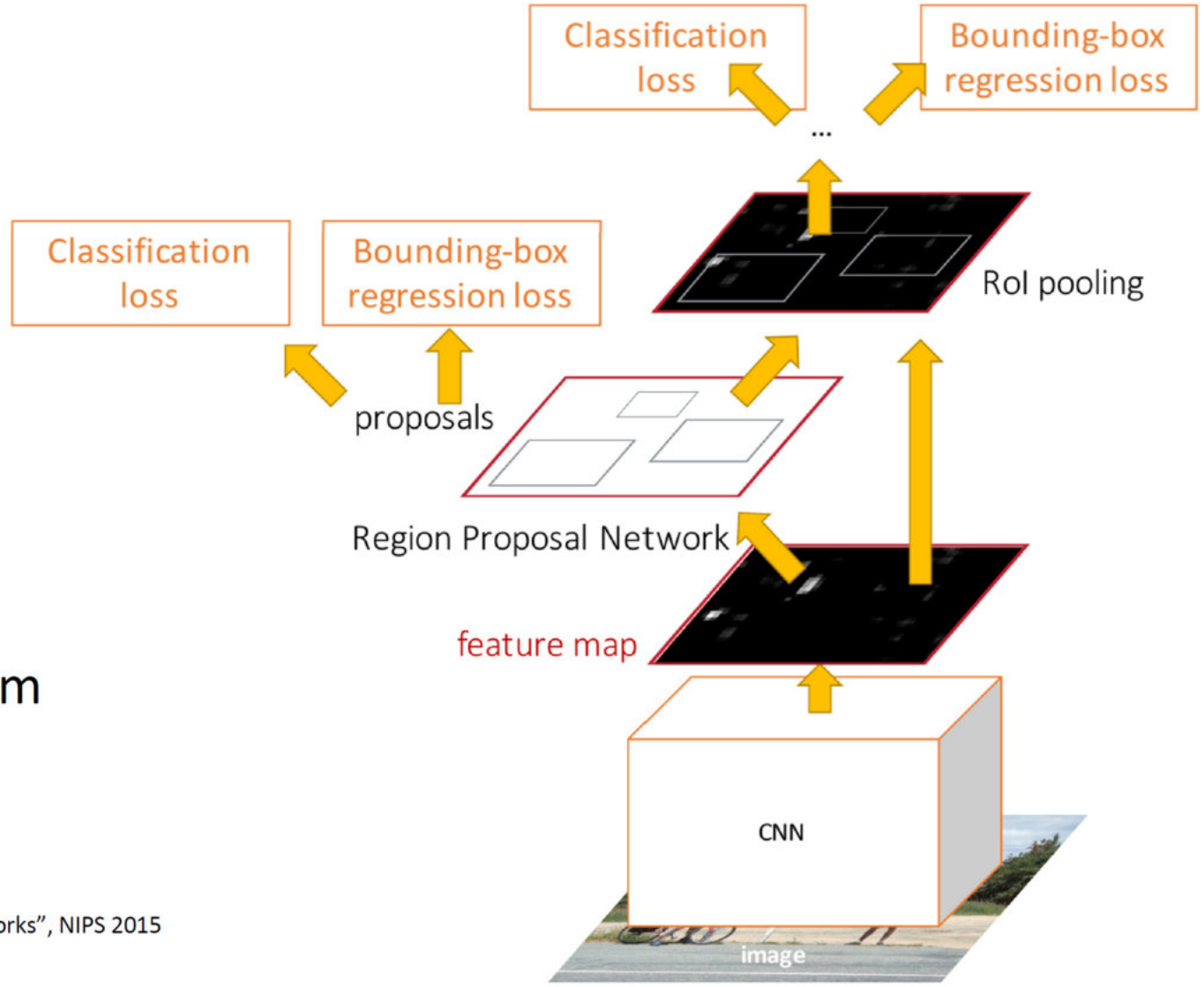
In practice $\lambda = 10$, so that both terms are roughly equally balanced

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

Faster R-CNN: Learnable Region Proposals

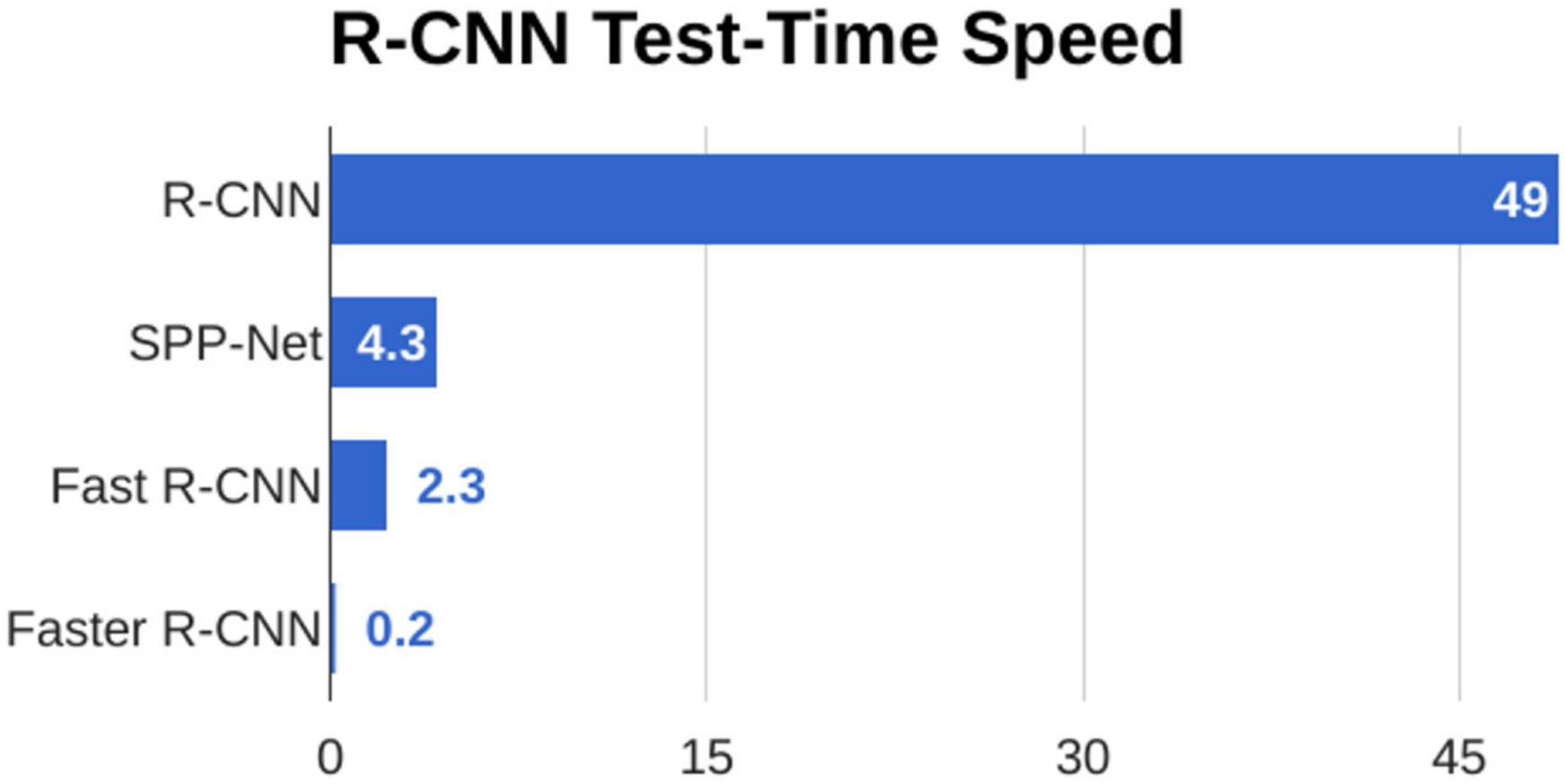
Jointly train with 4 losses:

1. **RPN classification**: anchor box is object / not an object
2. **RPN regression**: predict transform from anchor box to proposal box
3. **Object classification**: classify proposals as background / object class
4. **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", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

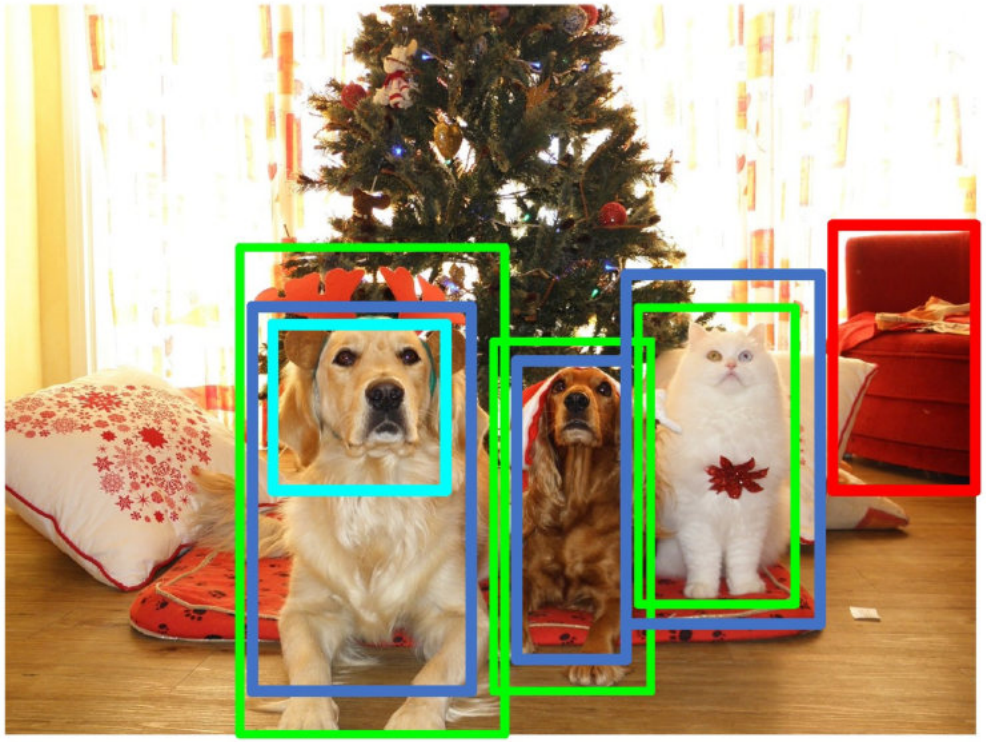
Faster R-CNN: Learnable Region Proposals



Slide from Deep Learning for Computer Vision @Univ. of Michigan

"Slow" R-CNN Training

Input Image



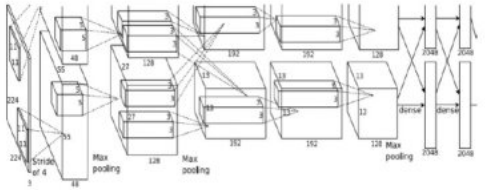
GT Boxes

Positive

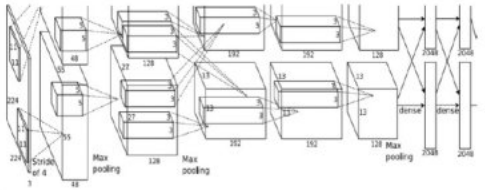
Neutral

Negative

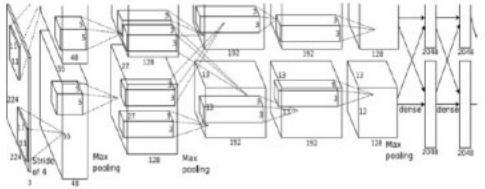
Run each region through CNN. For positive boxes predict class and box offset; for negative boxes just predict background class



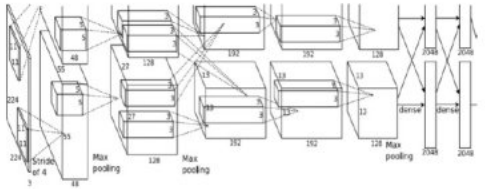
Class target: Dog
Box target: →



Class target: Cat
Box target: →



Class target: Dog
Box target: →

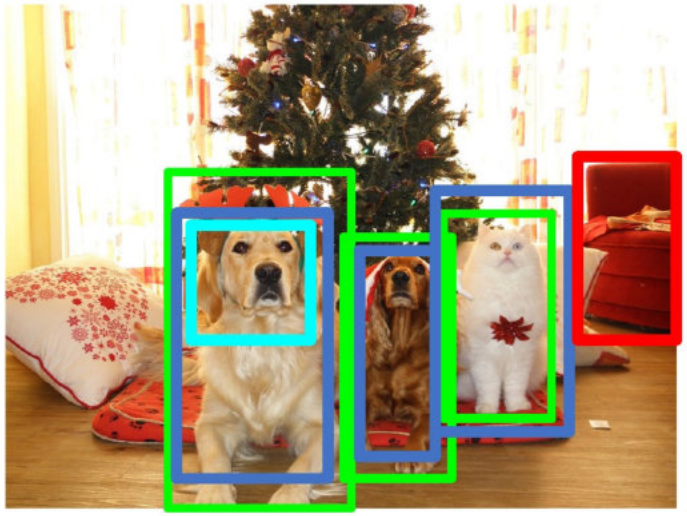


Class target: Background
Box target: None

Fast R-CNN Training

Crop features for each region, use them to predict class and box targets per region

Input Image



GT Boxes | Positive

Neutral | Negative

Backbone CNN

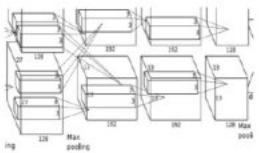
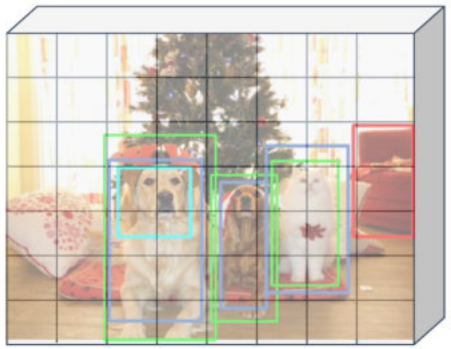


Image Features

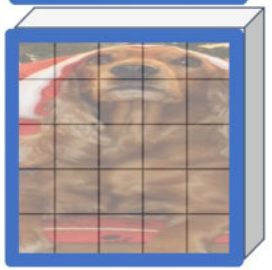


Class target: Dog

Box target: →

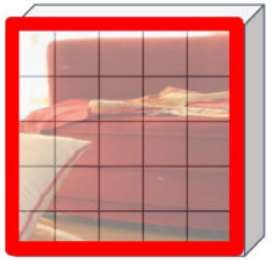
Class target: Cat

Box target: →



Class target: Dog

Box target: →



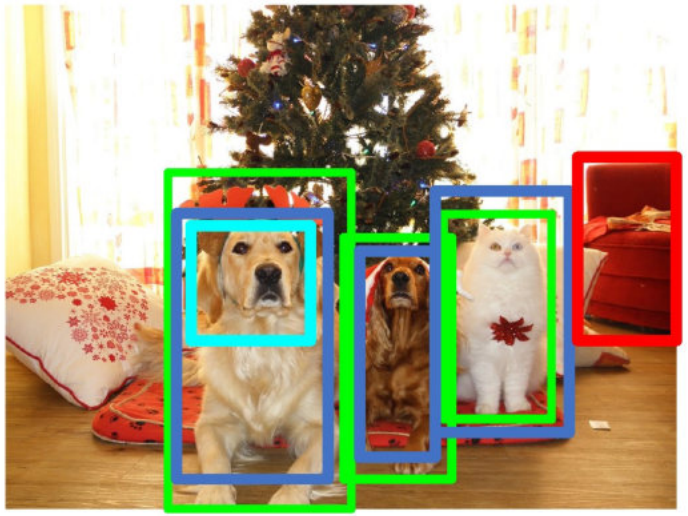
Class target: Background

Box target: None

Faster R-CNN Training: RPN Training

RPN predicts Object / Background for each anchor, as well as regresses from anchor to object box

Input Image



Backbone CNN

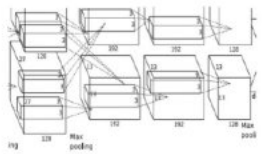
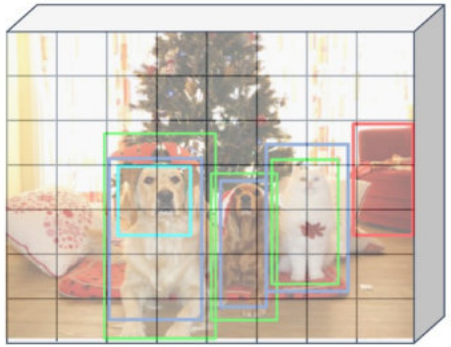
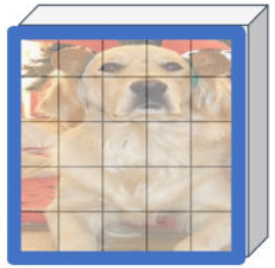


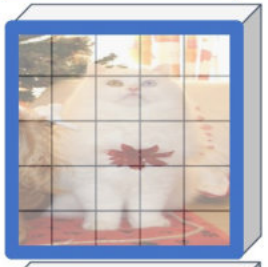
Image Features



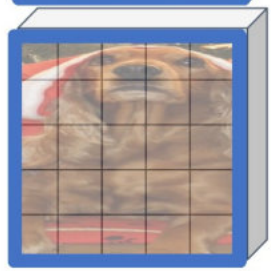
RPN gives lots of **anchors** which we classify as pos / neg / neutral by matching with ground-truth



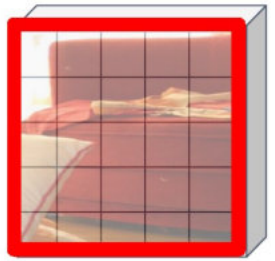
Class target: Obj
Box target: →



Class target: Obj
Box target: →



Class target: Obj
Box target: →



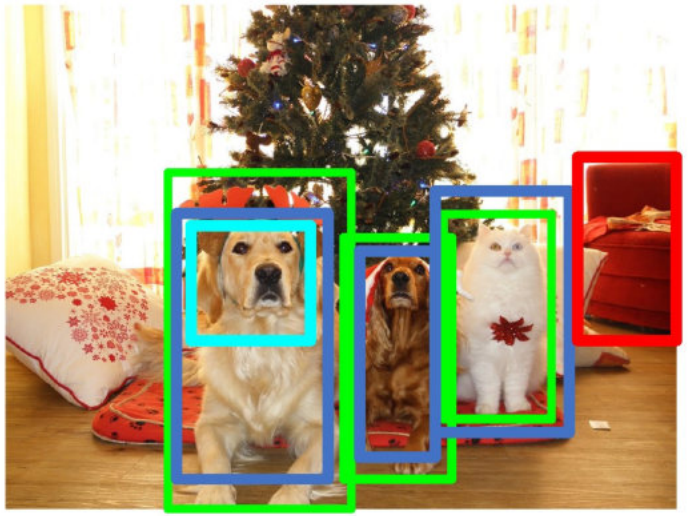
Class target: Background
Box target: None

GT Boxes | Positive | Neutral | Negative

Faster R-CNN Training: Stage 2

Crop features for each proposal, use them to predict class and box targets per region

Input Image



Backbone CNN

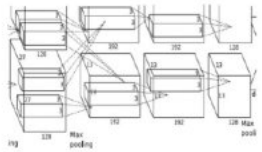
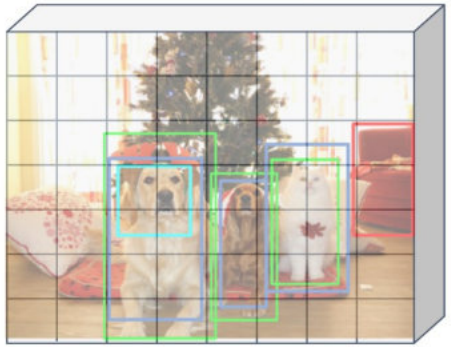
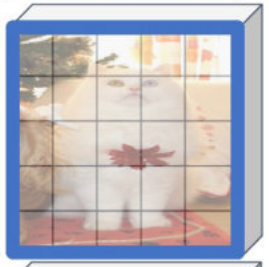


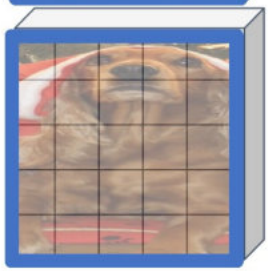
Image Features



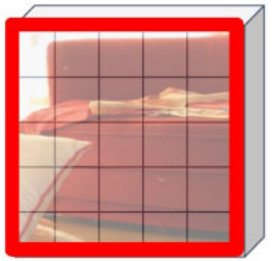
Class target: Dog
Box target:



Class target: Cat
Box target:



Class target: Dog
Box target:



Class target: Background
Box target: None

GT Boxes | Positive | Neutral | Negative

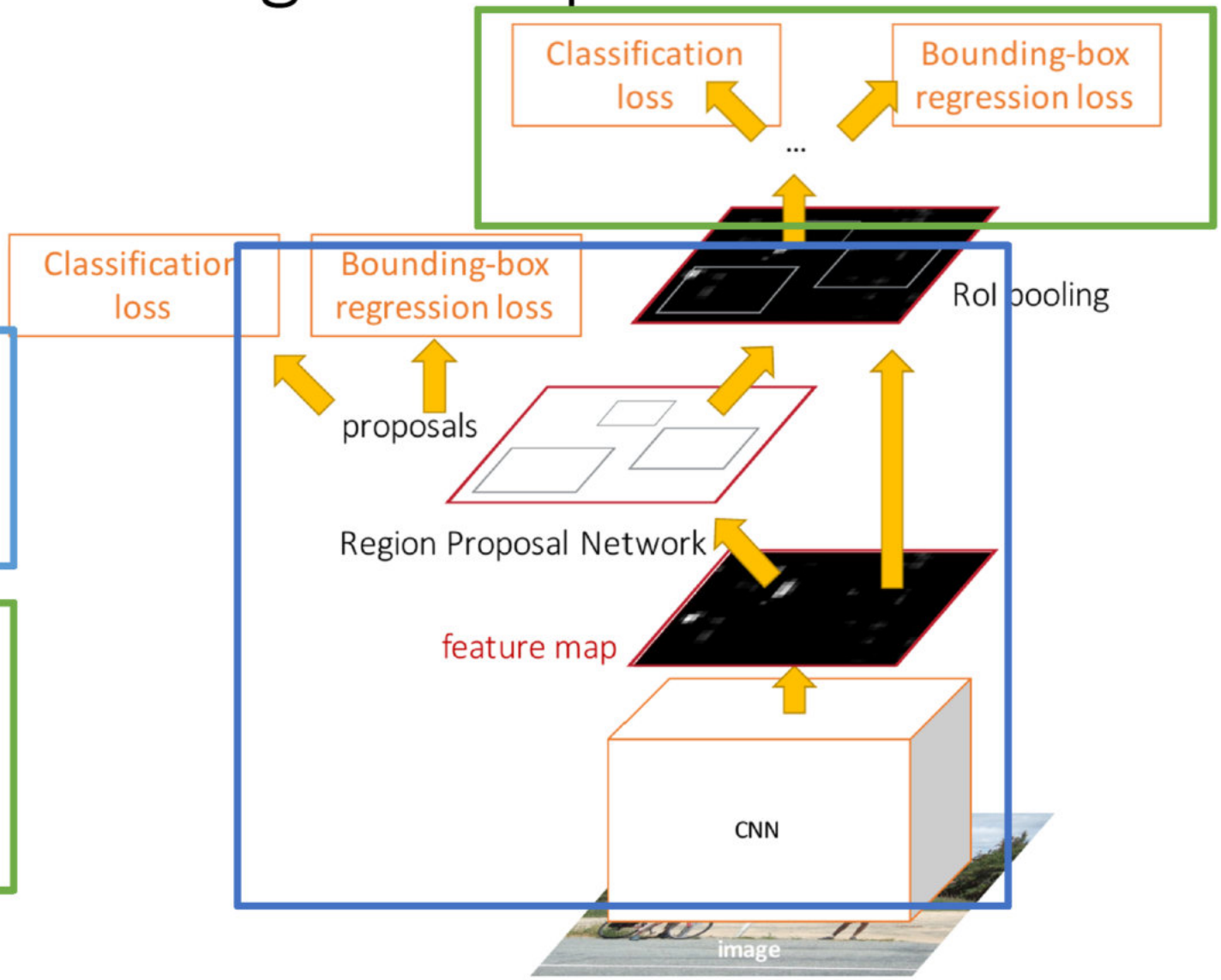
Now proposals come from RPN rather than selective search, but otherwise this works the same as Fast R-CNN training

Faster R-CNN: Learnable Region Proposals

Faster R-CNN is a
Two-stage object detector

First stage: Run once per image
- Backbone network
- Region proposal network

Second stage: Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset



Slide from Deep Learning for Computer Vision @Univ. of Michigan

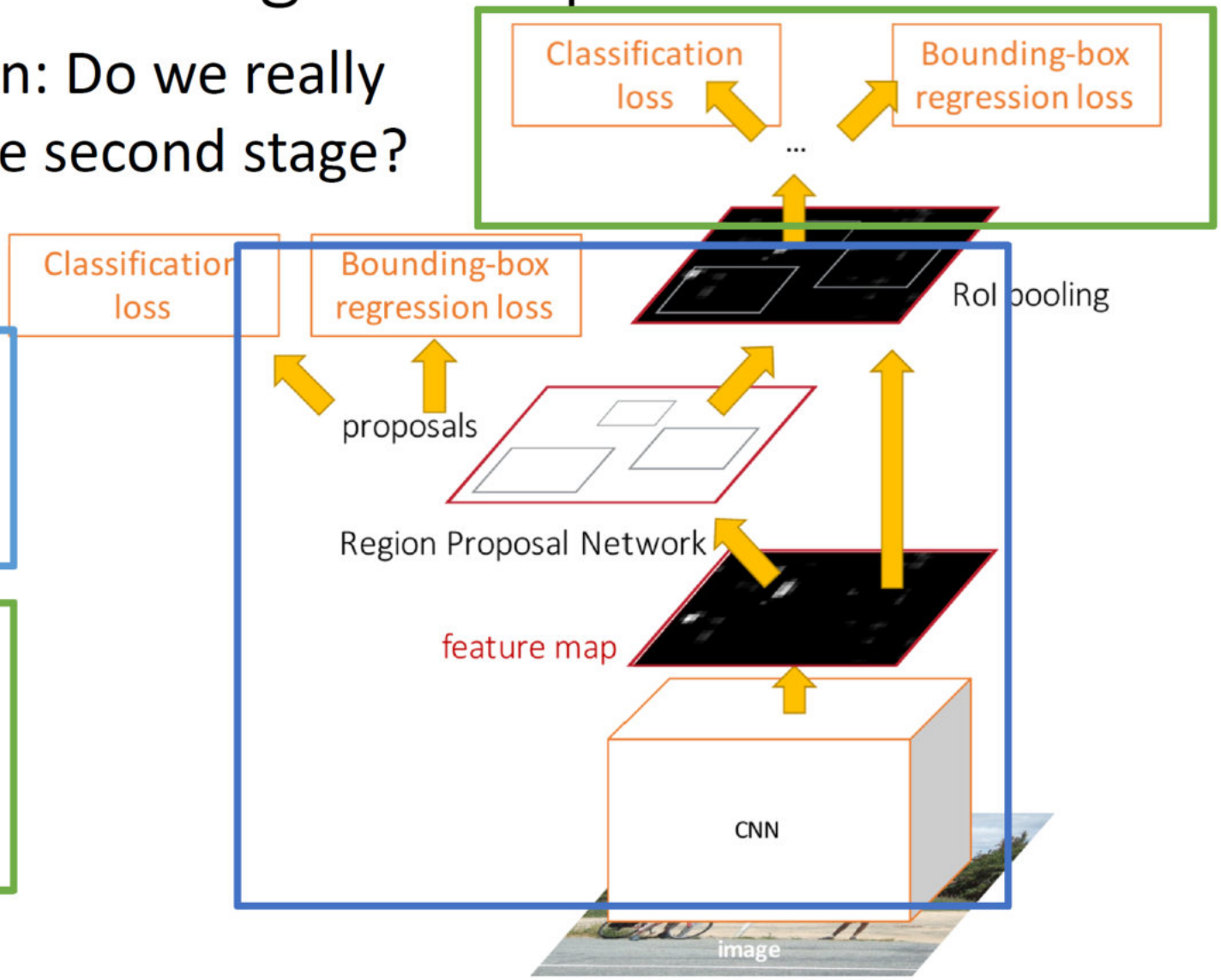
Faster R-CNN: Learnable Region Proposals

Question: Do we really need the second stage?

Faster R-CNN is a
Two-stage object detector

First stage: Run once per image
- Backbone network
- Region proposal network

Second stage: Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset



Slide from Deep Learning for Computer Vision @Univ. of Michigan

Single-Stage Object Detection

RPN: Classify each anchor as object / not object

Single-Stage Detector: Classify each object as one of C categories (or background)

Run backbone CNN to get features aligned to input image

Input Image
(e.g. 3 x 640 x 480)

CNN

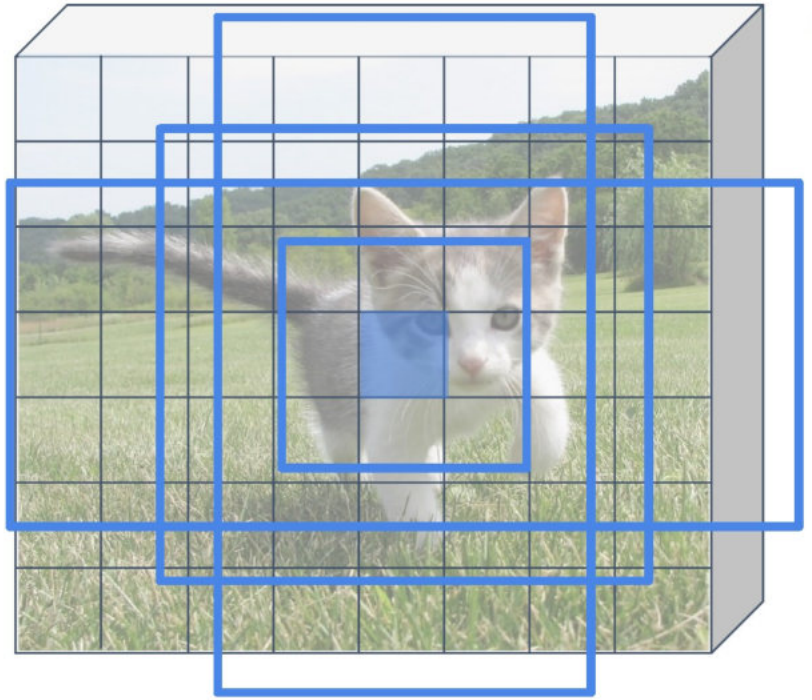


Image features
(e.g. 512 x 20 x 15)

Conv

Anchor category
(C+1) x K x 20 x 15

Box transforms
4K x 20 x 15

Remember: K anchors at each position in image feature map

Slide from Deep Learning for Computer Vision @Univ. of Michigan

YOLO

- You Only Look Once
- Quite similar with Faster R-CNN and very FAST!

Concepts

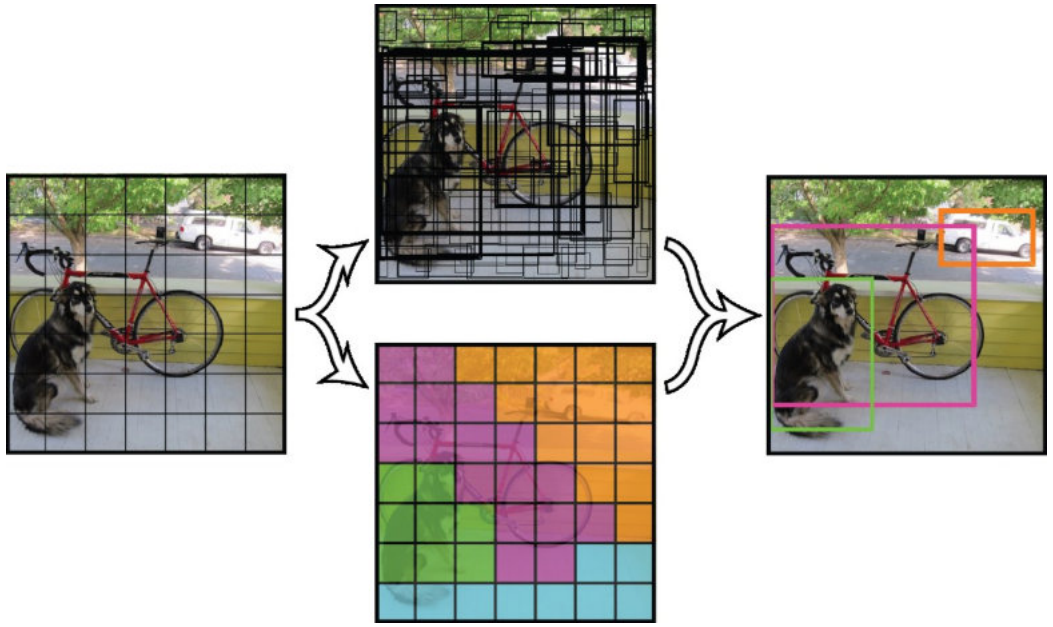
- Detection as Single Regression Problem
- Developed as Single Convolutional Network
- Reason Globally on the Entire Image
- Learns Generalizable Representations

Unified Detection

- Given an image, divide it into an SxS grid

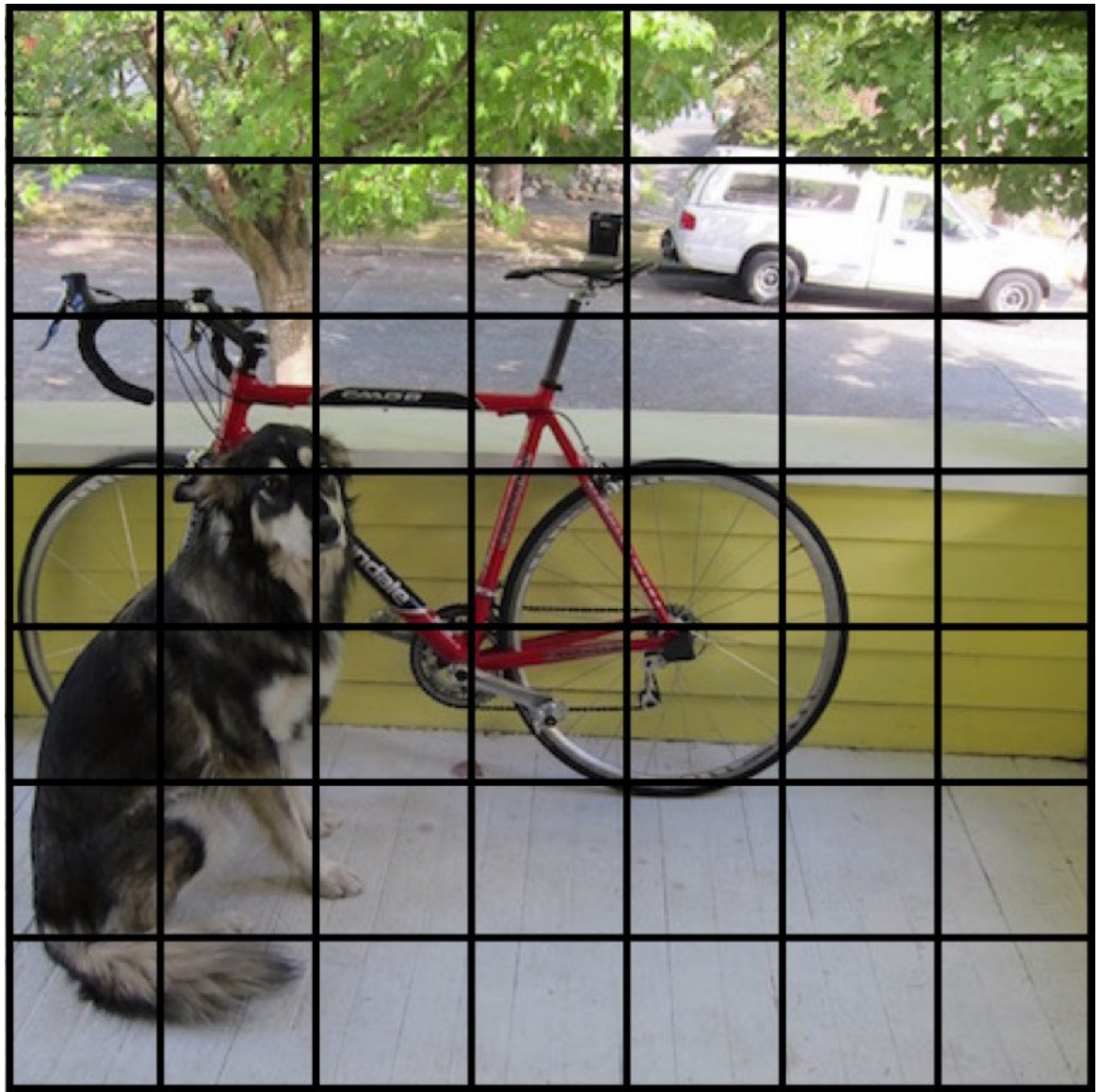
→If the center of an object falls into the grid cell, that grid cell is responsible

- Each cell predicts B bounding boxes
  - Five predictions: x, y, w, h, confidence
- Each cell predicts C class probabilities
  - Cell → C class probabilities, Bx5 bounding box informations
- One cell → One class probability
  - Low false positive!

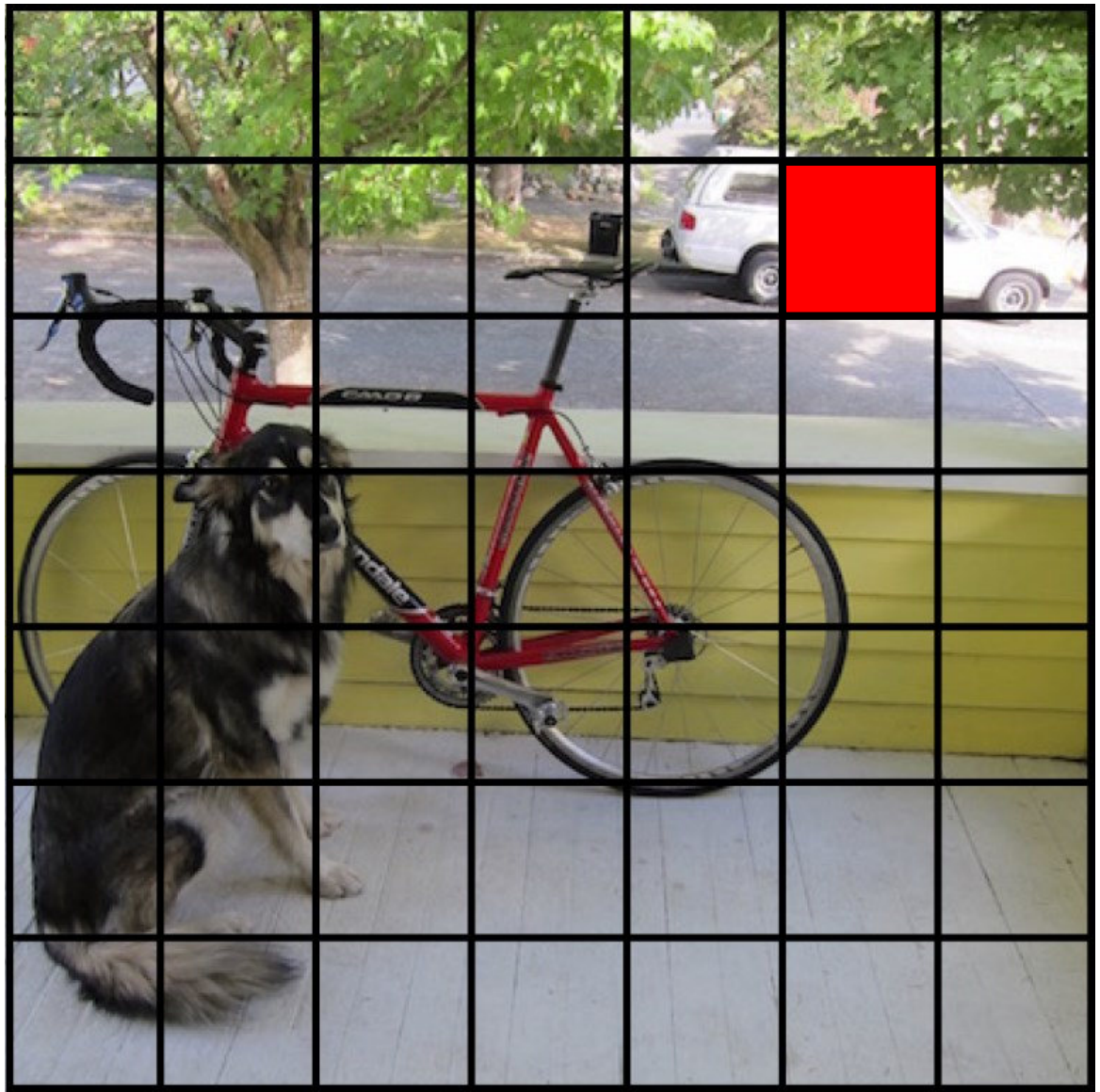


Detection Flow

Split the image into a grid(7x7)

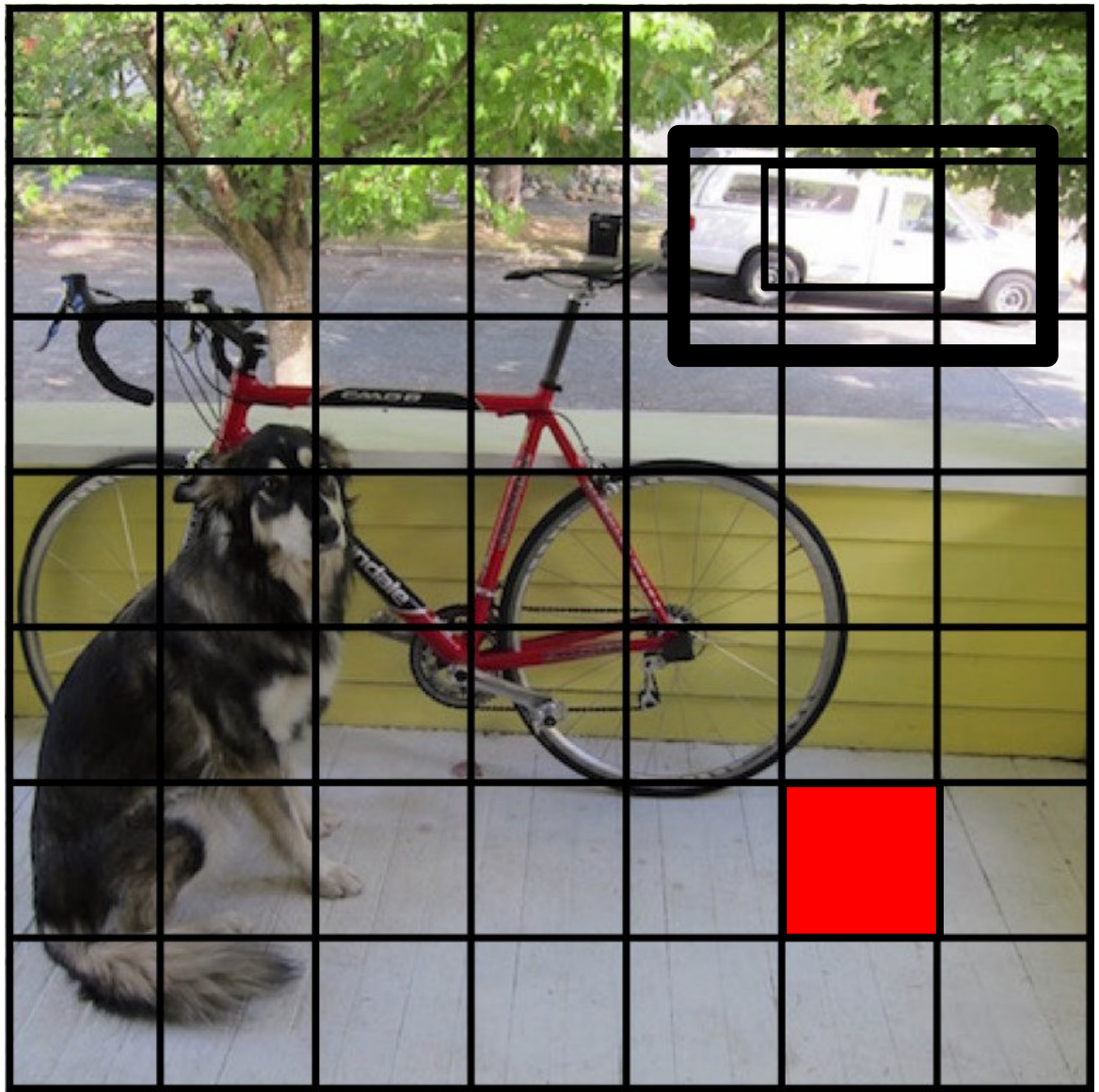


Each cell predicts boxes and confidences: P(Object)

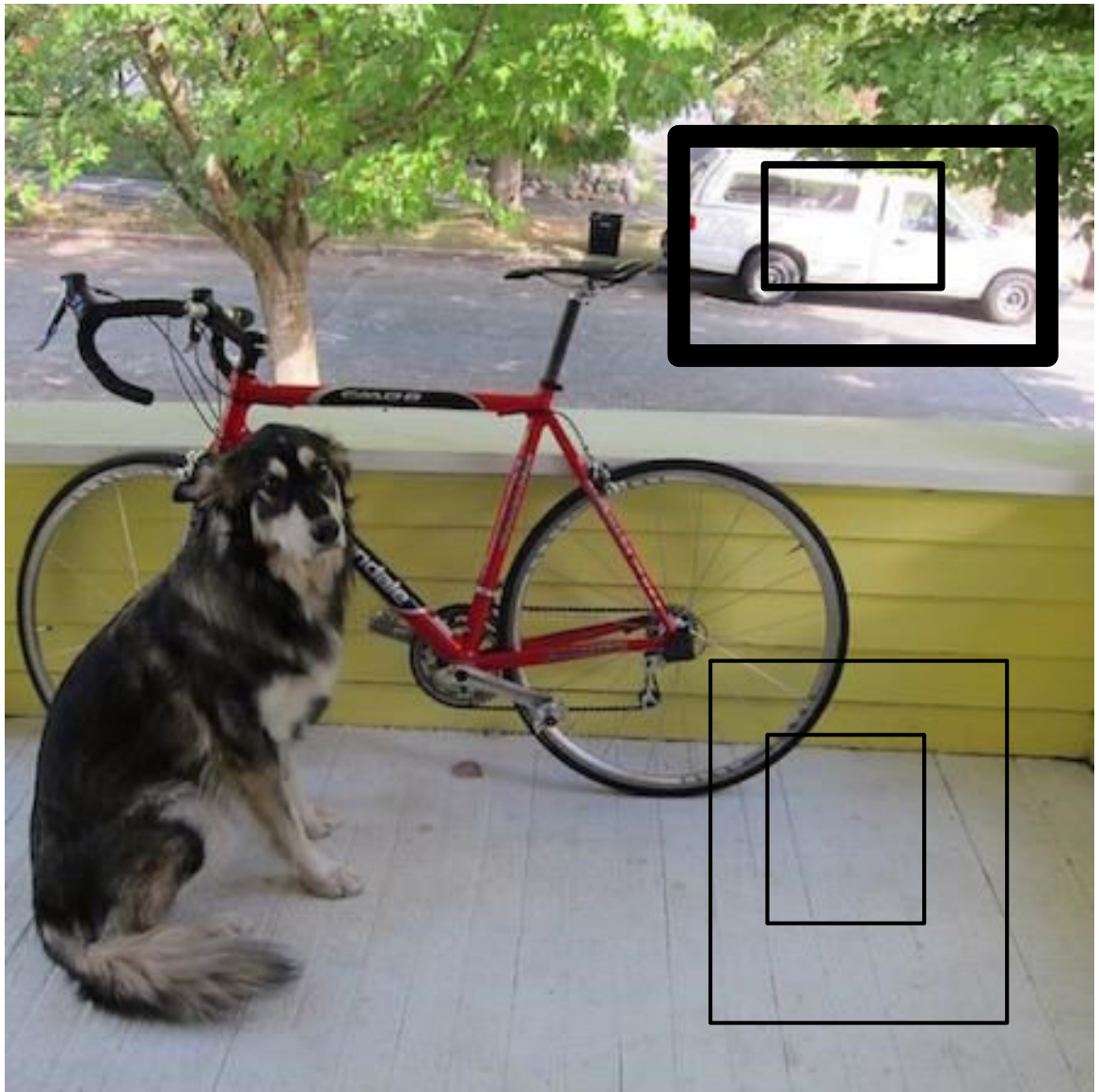


Each cell predicts boxes and confidences: P(Object)

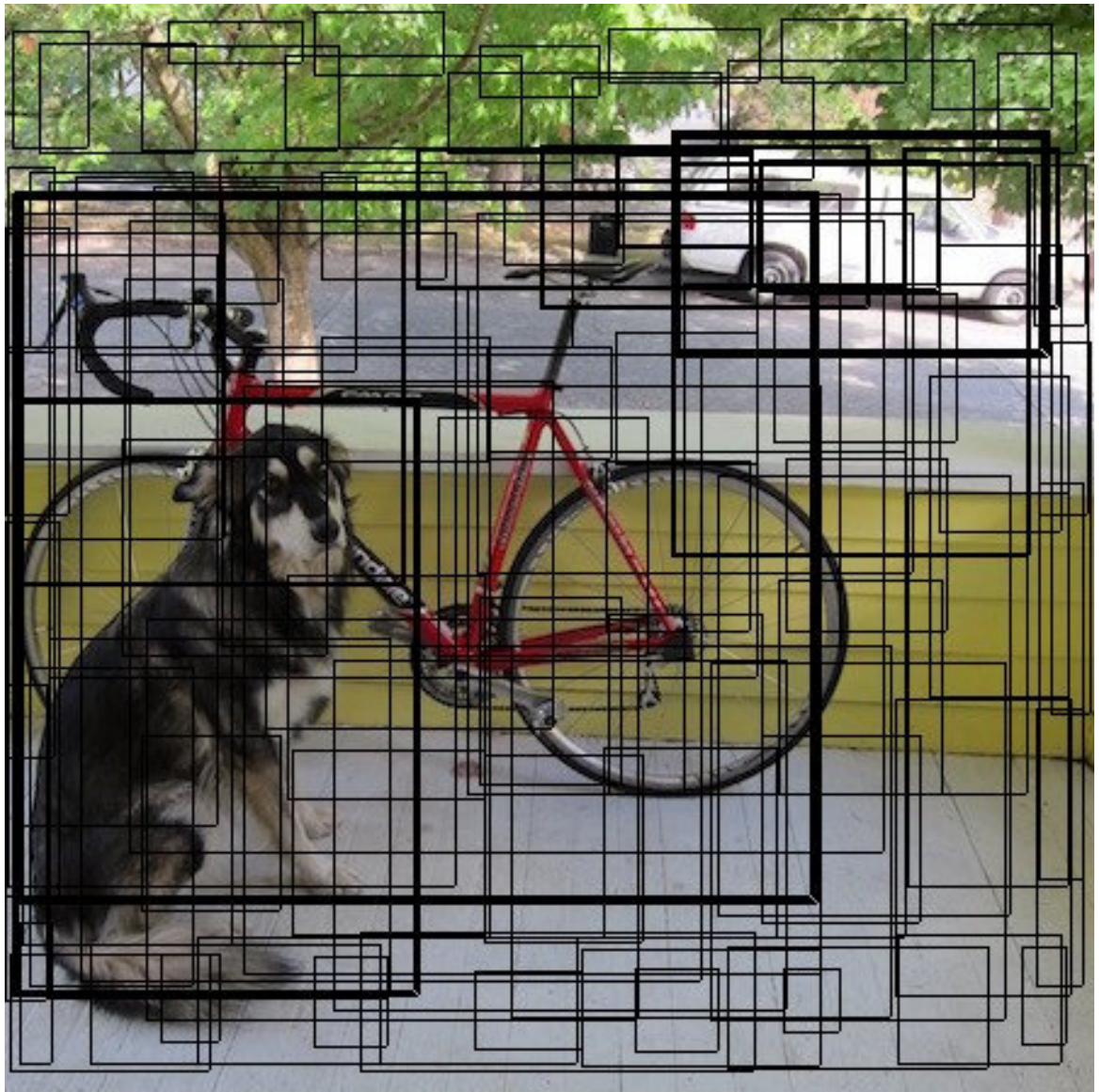
Each cell predicts boxes and confidences: P(Object)



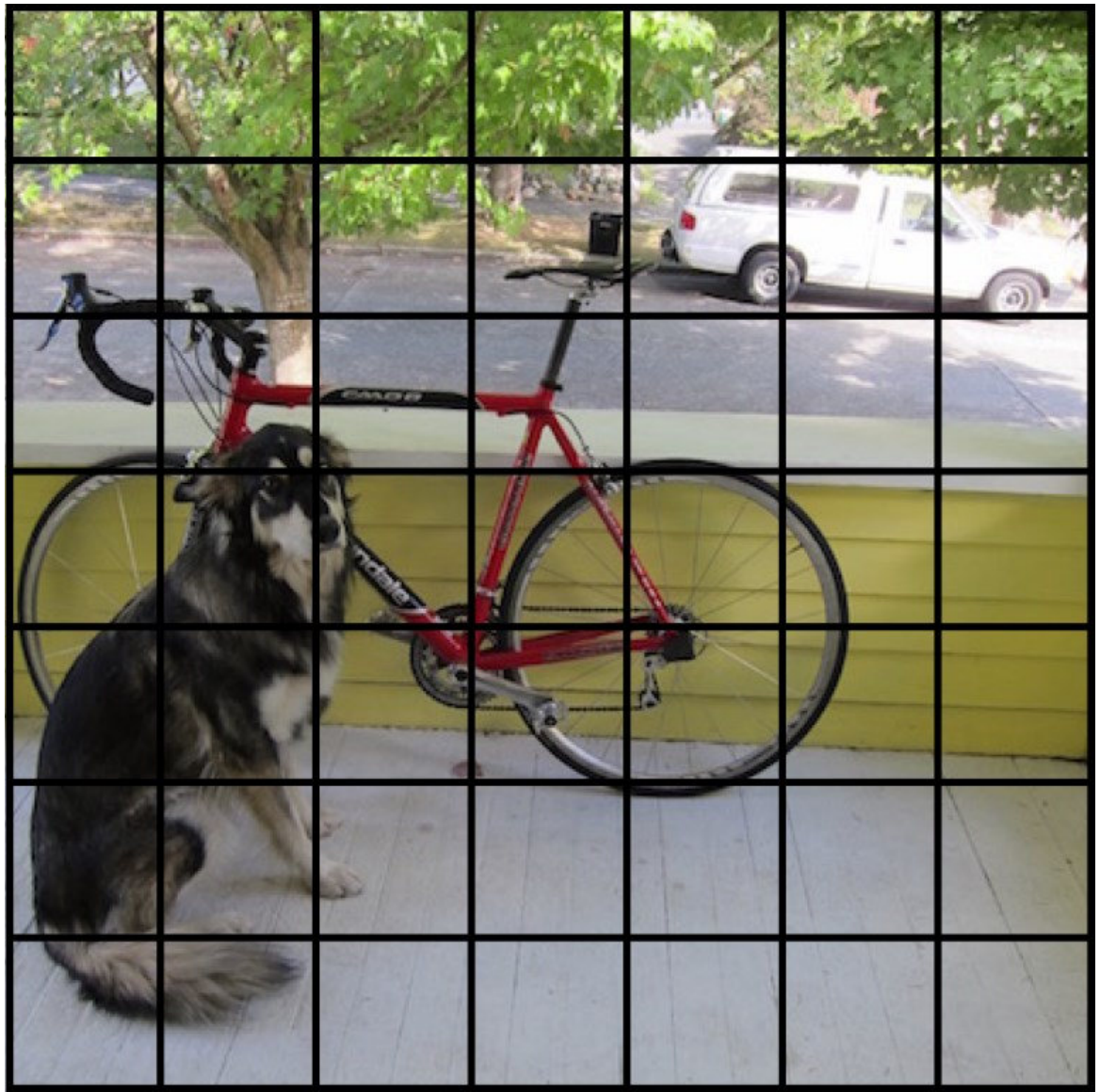
Each cell predicts boxes and confidences: P(Object)



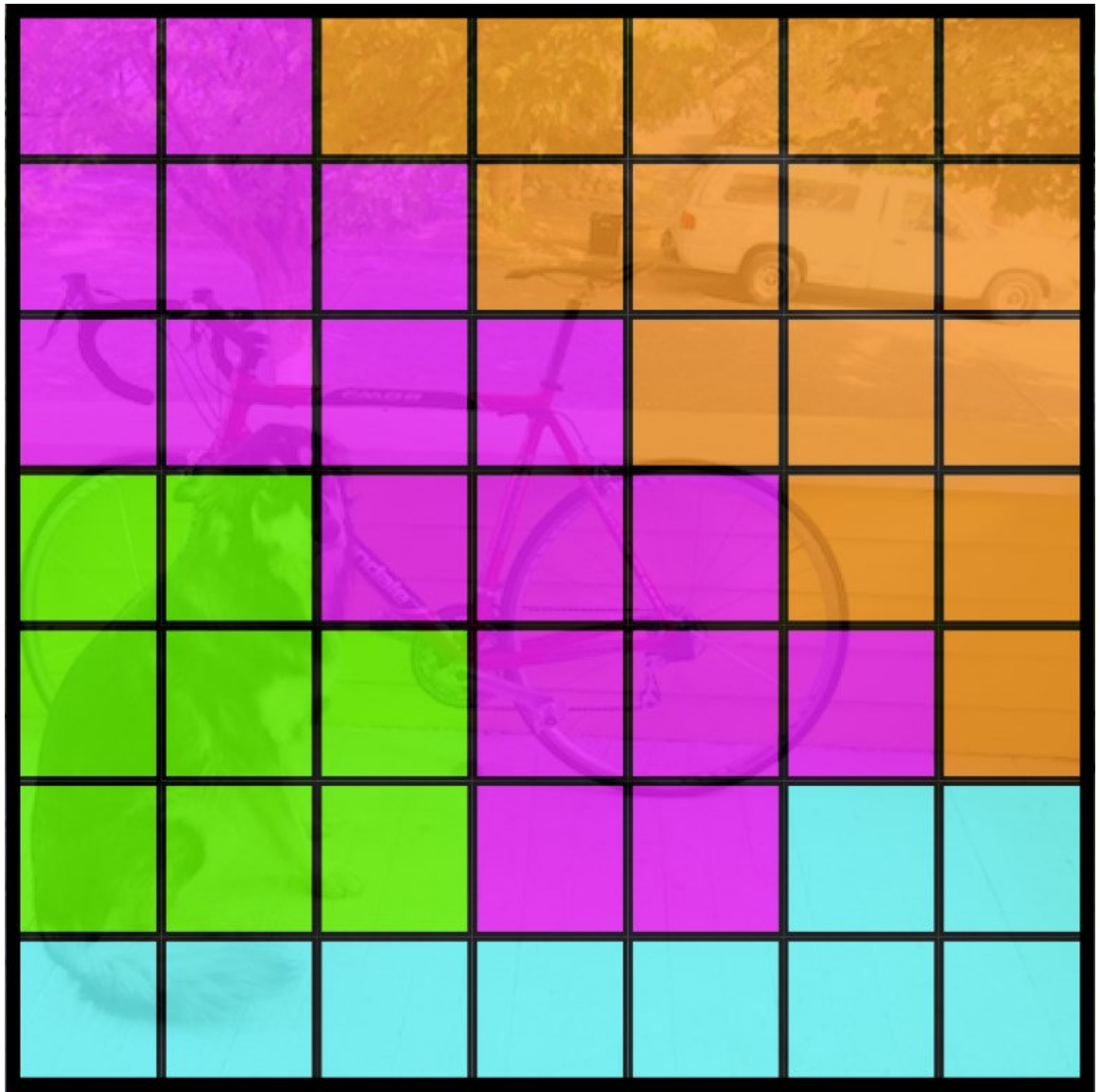
Each cell predicts boxes and confidences: P(Object)



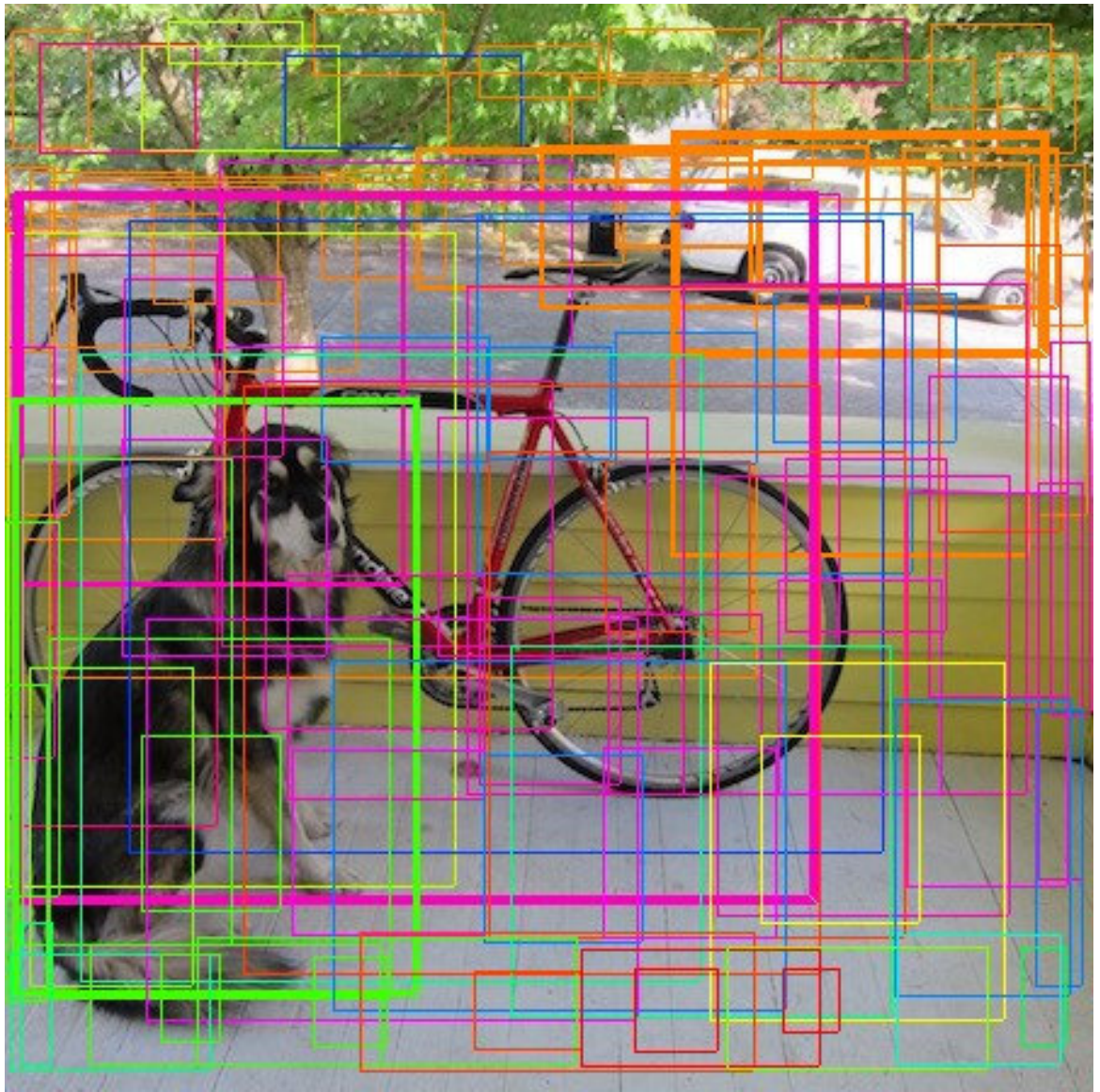
Each cell also predicts a class probability



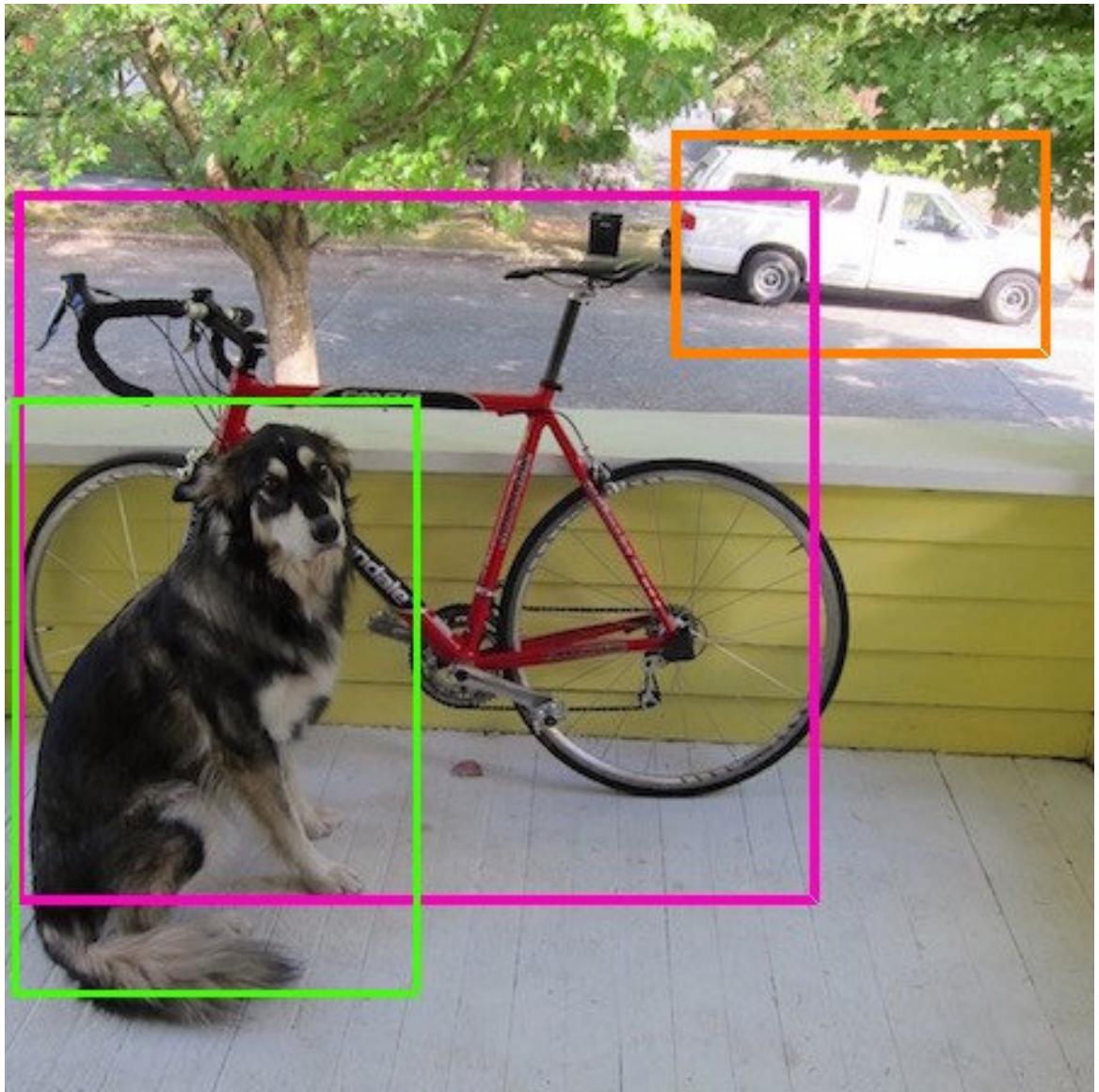
Each cell also predicts a class probability



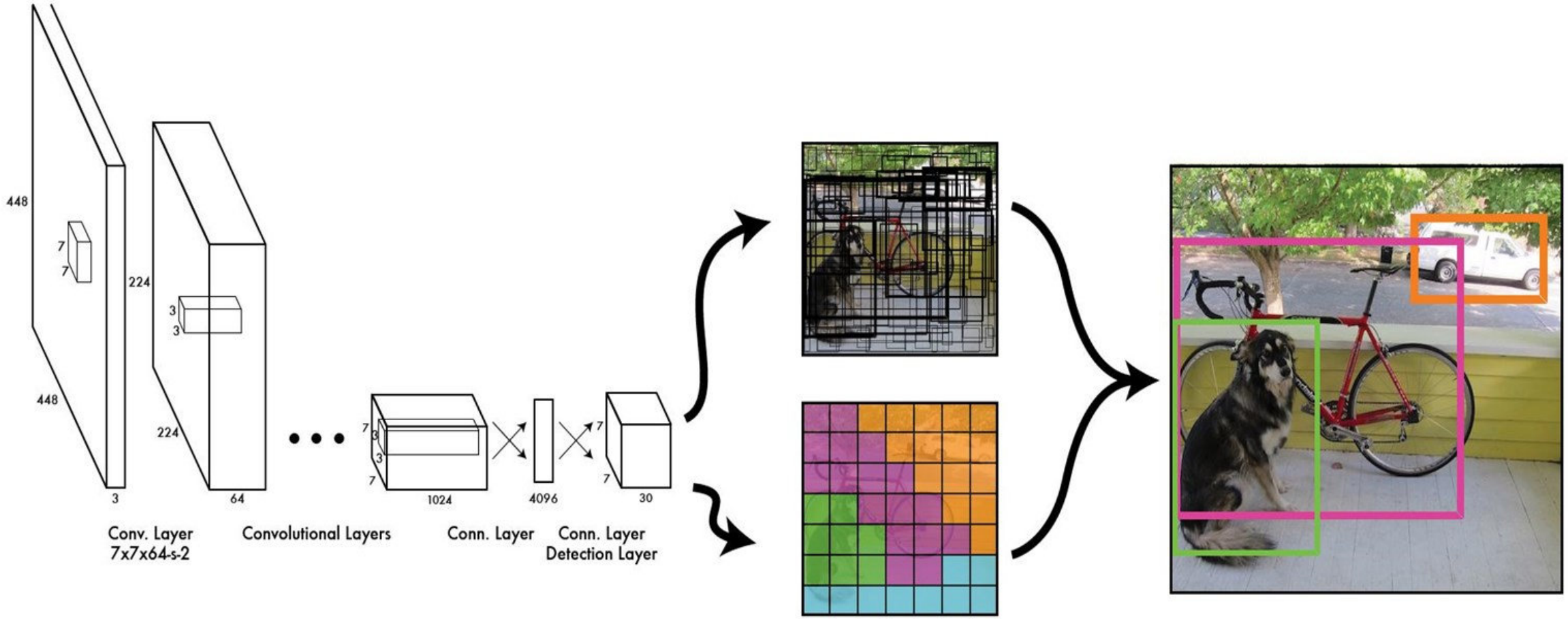
Then we combine the box and class predictions.



Finally we do NMS and threshold detections



YOLO


448
448
7
7
224
224
3
3
3
64
7
7
3
3
7
1024
4096
7
7
30
Conv. Layer
7x7x64-s-2
Convolutional Layers
Conn. Layer
Conn. Layer
Detection Layer

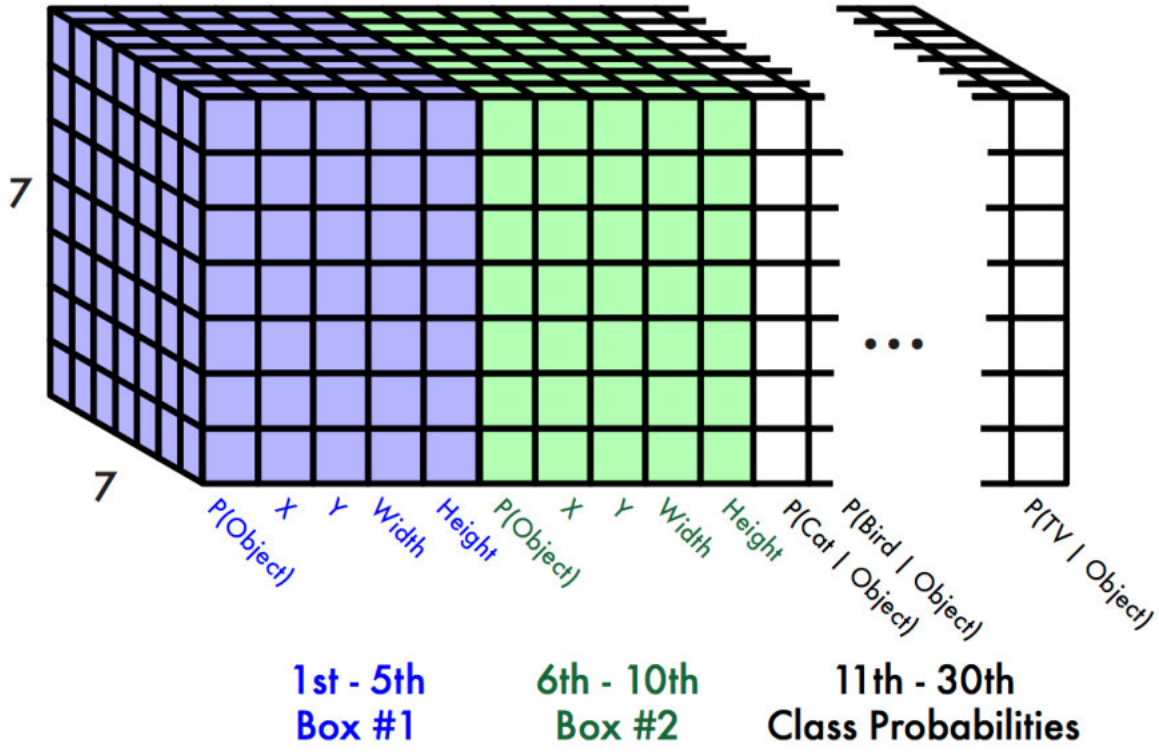
Outputs

Each cell predicts:

- For each bounding box:
  - 4 coordinates (x, y, w, h)
  - 1 confidence value
- Some number of class probabilities

For Pascal VOC:

- 7x7 grid
- 2 bounding boxes / cell
- 20 classes



Loss Function

- In training, one predictor which has the highest IoU with the ground truth is responsible

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \qquad (3)$$

$\mathbb{1}_{ij}^{obj}$ — **The *j*th bbox predictor** in ***cell i*** is "responsible" for that prediction

$\mathbb{1}_{ij}^{noobj}$

$\mathbb{1}_{i}^{obj}$ — If object appears in ***cell i***

Note that the loss function only penalizes classification error if an object is present in that grid cell (hence the conditional class probability discussed earlier). It also only penalizes bounding box coordinate error if that predictor is "responsible" for the ground truth box (i.e. has the highest IOU of any predictor in that grid cell).

Problems

- Each grid cell can predict only B(=2) bounding boxes and one class probability
  - Not good for small objects that flock together
- Uses relatively coarse features
  - Locations of bboxes are inaccurate
- Loss function treats errors in small bbox and big bbox equally
  - Not good for scoring

SSD – Single Shot MultiBox Detector

- Current state-of-the-art object detection systems are variants of the following approach:
  - Hypothesize bounding boxes, resample pixels or features for each box, and apply a high-quality classifier.

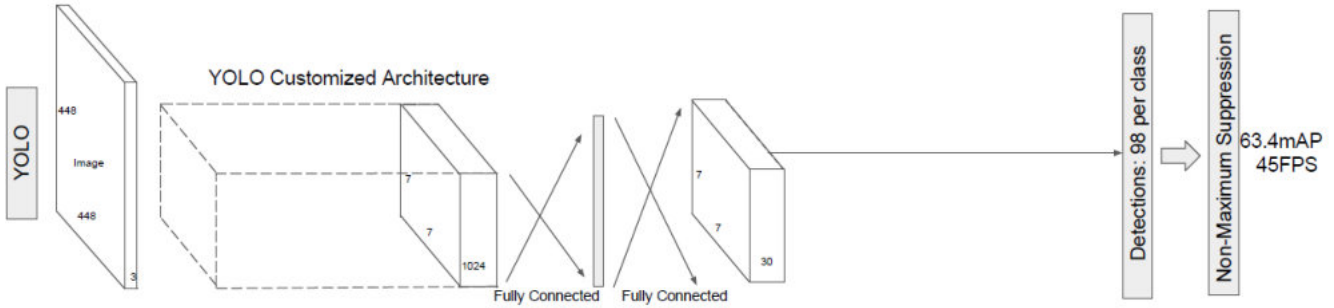
- Computationally too intensive and too slow for real-time applications.
  - Faster R-CNN operates at only 7 FPS with mAP 73.2%

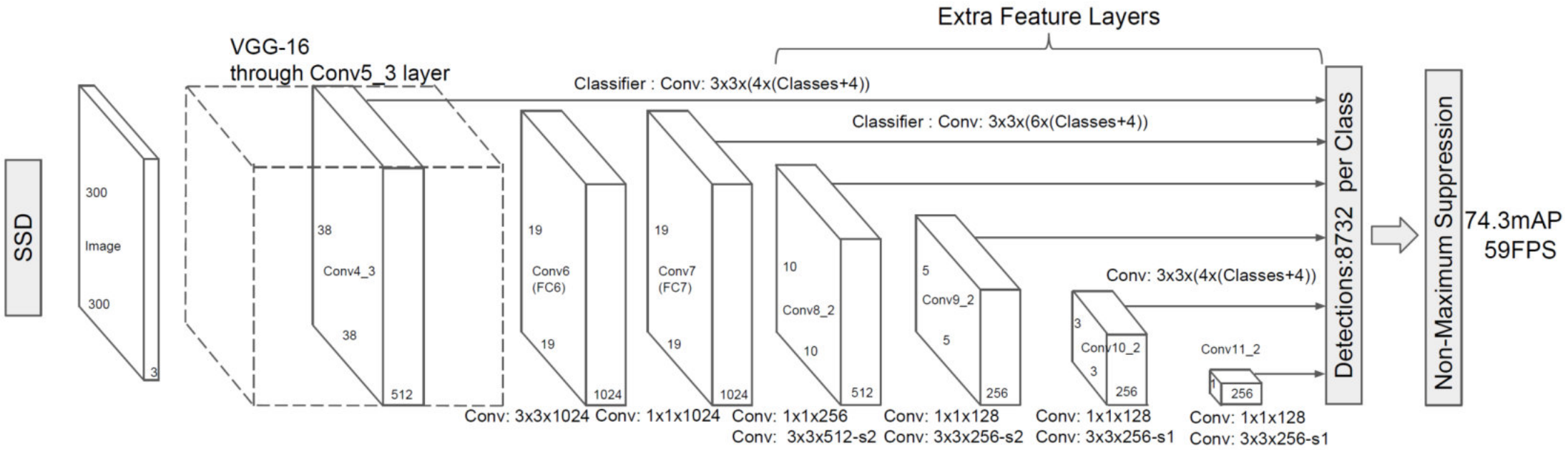
- Significantly increased speed comes only at the cost of significantly decreased detection accuracy.
  - YOLO operates at 45FPS with mAP 63.4%

Single Shot MultiBox Detector

- First deep network based object detector that does not resample pixels or features for bounding box hypotheses(1-stage) and is as accurate as approaches that do.
  - 59 FPS with mAP 74.3% on VOC2007 test

Model





Base network – VGG16

FC6 and FC7 converted to convolution layer

Auxiliary structure to produce detections

Collections of BBoxes & Scores

Default Boxes and Aspect Ratios

- For each box out of k at a given location, we compute c class scores and the 4 offsets relative to the original default box shape.
- Total of (c+4)k filters at each location in the feature map.
- Yielding (c+4)kmn outputs for a m x n feature map.

Training – Matching Strategy

- Matching each ground truth box to the default box with the best jaccard overlap. Every ground truth box has at least 1 correspondence.
- Then, matching default boxes to any ground truth with jaccard overlap higher than a threshold(0.5).

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

Figure from Wikipedia

Training Objective

set to 1 by cross validation

- Similar to Faster R-CNN

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$$

  - N : number of default matched bboxes
  - L_{conf} : classification loss → cross-entropy

$$L_{conf}(x, c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

  - L_{loc} : localization loss → Smoth L1 loss

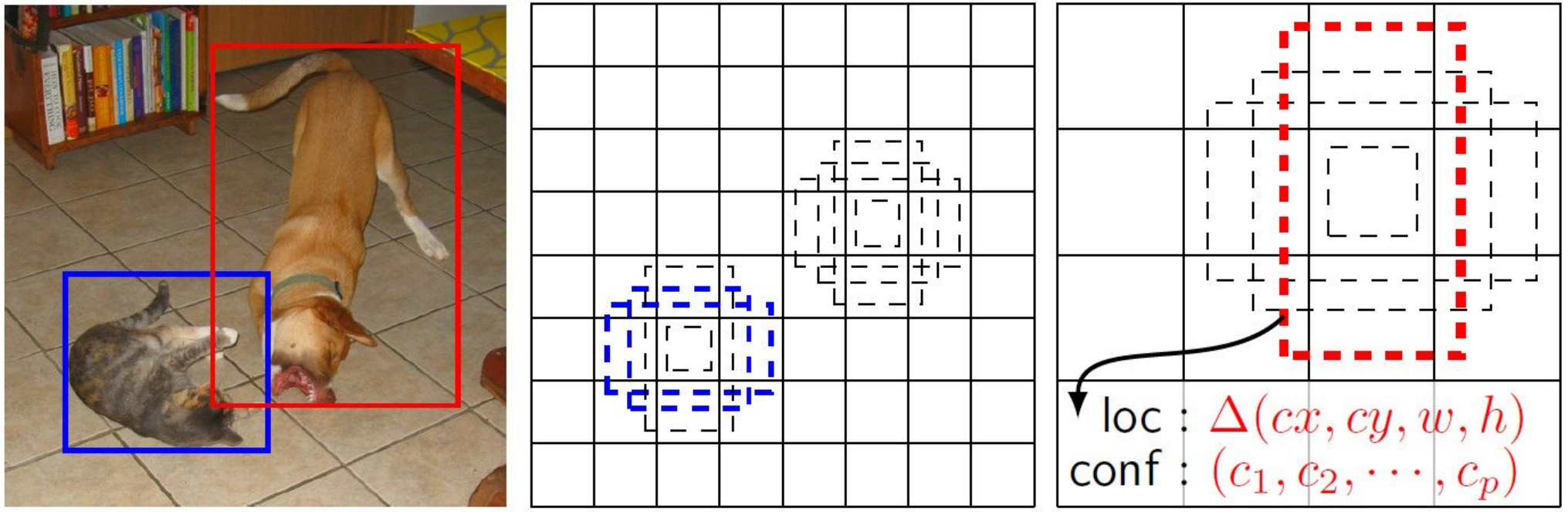
$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^{k} \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$

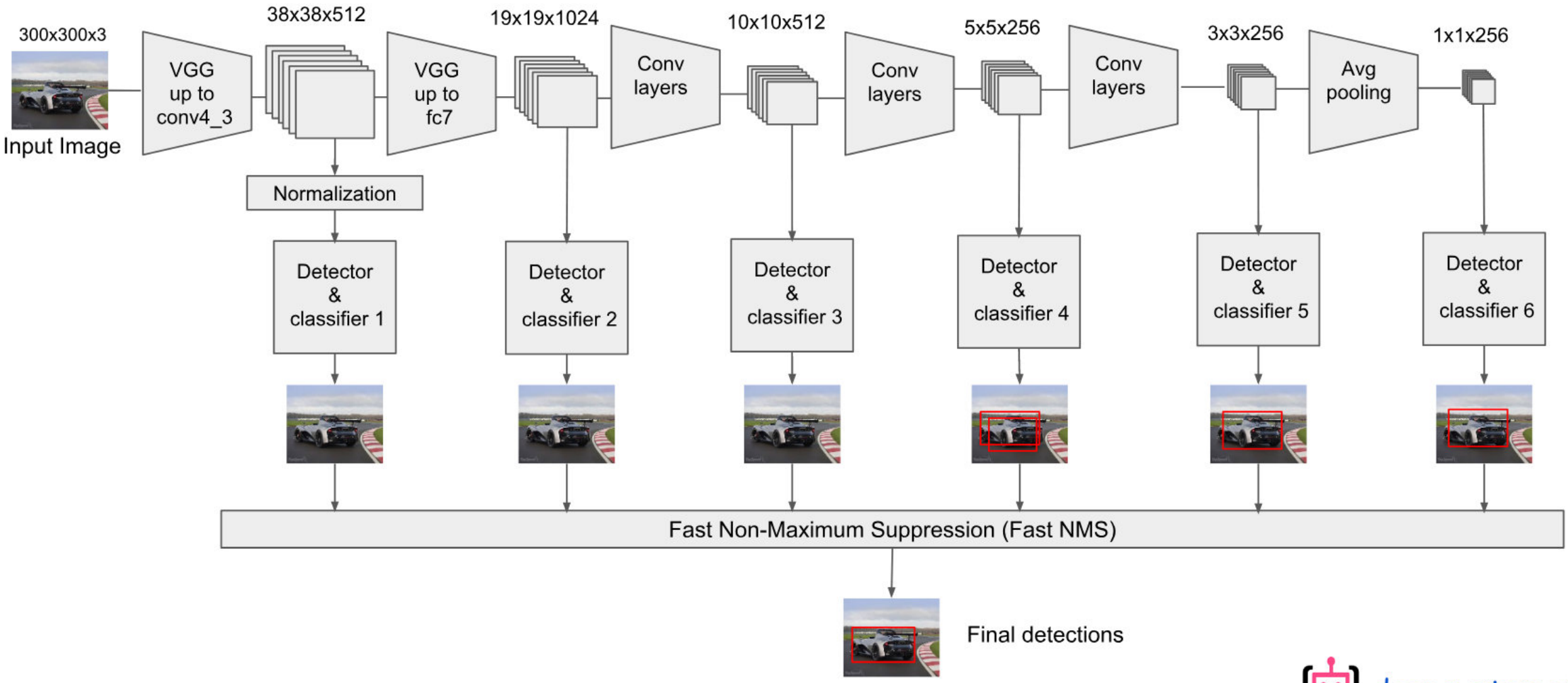
Choosing Scales and Aspect Ratios for Default Boxes

- Feature maps from different layers are used to handle scale variance.
- Specific feature map locations learn to be responsive to specific area of the image and particular scales of objects.



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

Choosing Scales and Aspect Ratios for Default Boxes



deepsystems.io

Slide from https://goo.gl/NsP6Wg

Hard Negative Mining

- Significant imbalance between positive and negative training examples.
  - After the matching step, most of the default boxes are negatives, especially when the number of possible default boxes is large.

- Sorting them using the highest confidence loss for each default box.
- Pick the top ones so that the ratio between the negatives and positives is at most 3:1

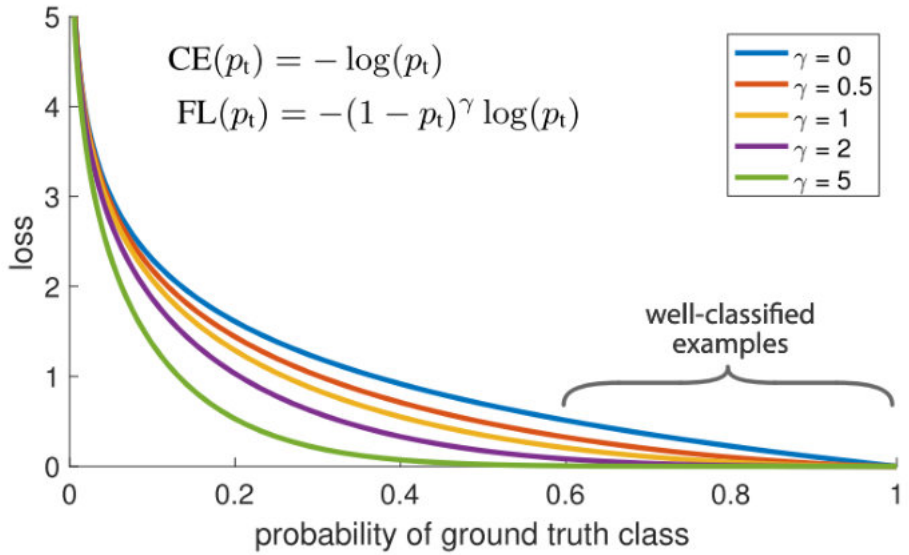
Class Imbalance Problem – RetinaNet

- Class 간의 data가 균형이 안맞는 경우
  - Weighted cross-entropy
    - Cross entropy 식에 class 별로 weight를 다르게 주어서 사용

$$L = -\sum_{C} w_c \cdot \log p$$

  - Focal Loss
    - Cross entropy에 비해 높은 확률의 경우에는 상대적으로 작은 loss를 나은 확률의 경우에는 높은 loss를 주도록 $(1\text{-}p)^{\gamma}$를 곱해줌

$$L = -\sum (1-p)^{\gamma} \cdot \log p$$



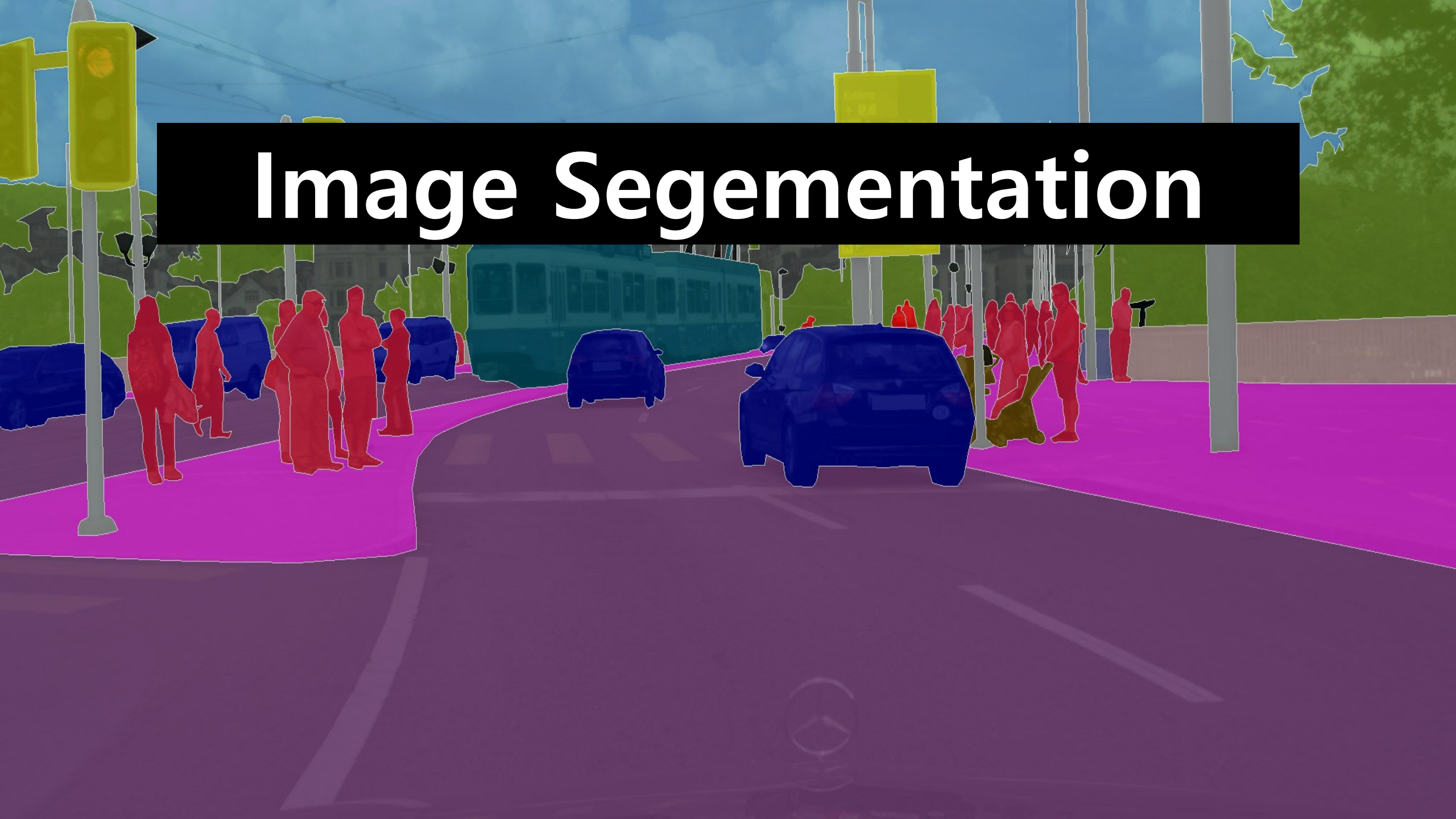
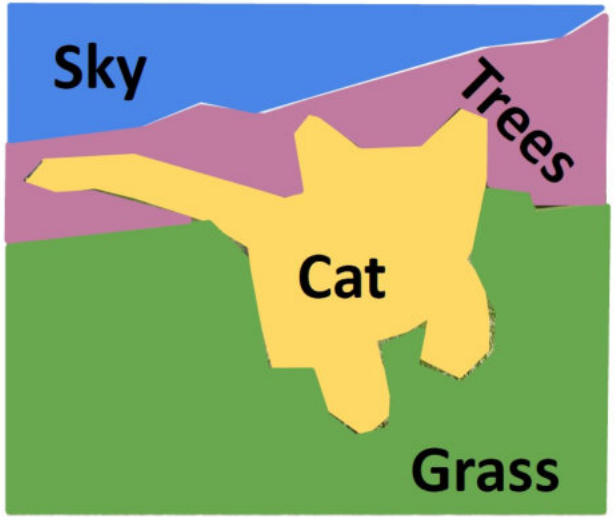


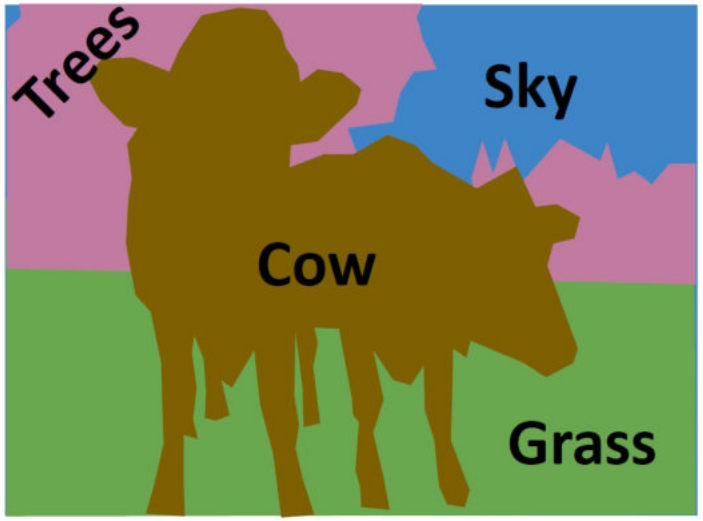
Image Segementation

Semantic Segmentation

Label each pixel in the image with a category label

Don’t differentiate instances, only care about pixels





Slide from Deep Learning for Computer Vision @Univ. of Michigan

Fully Convolutional Networks for Semantic Segmentation

Fully Convolutional Networks for Semantic Segmentation

Jonathan Long* Evan Shelhamer* Trevor Darrell
UC Berkeley
{jonlong,shelhamer,trevor}@cs.berkeley.edu

Abstract

Convolutional networks are powerful visual models that yield hierarchies of features. We show that convolutional networks by themselves, trained end-to-end, pixels-to-pixels, exceed the state-of-the-art in semantic segmentation. Our key insight is to build "fully convolutional" networks that take input of arbitrary size and produce correspondingly-sized output with efficient inference and learning. We define and detail the space of fully convolutional networks, explain their application to spatially dense prediction tasks, and draw connections to prior models. We adapt contemporary classification networks (AlexNet [22],

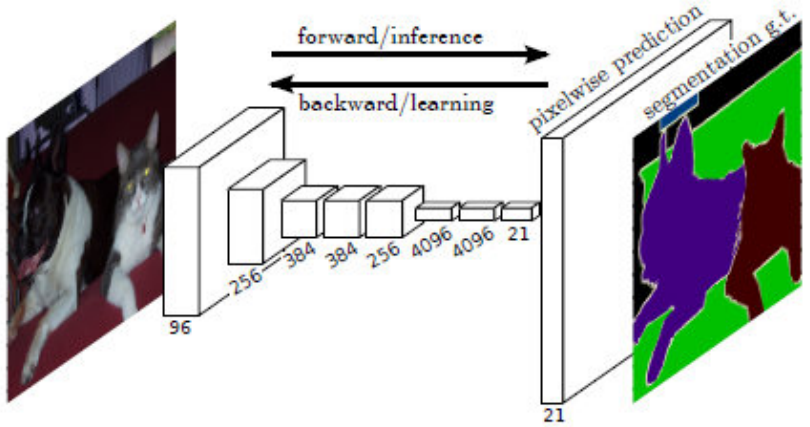


Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

Semantic Segmentation: Fully Convolutional Network

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!

Input:
3 x H x W

Conv → Conv → Conv → Conv

Convolutions:
D x H x W

Scores:
C x H x W

argmax

Predictions:
H x W

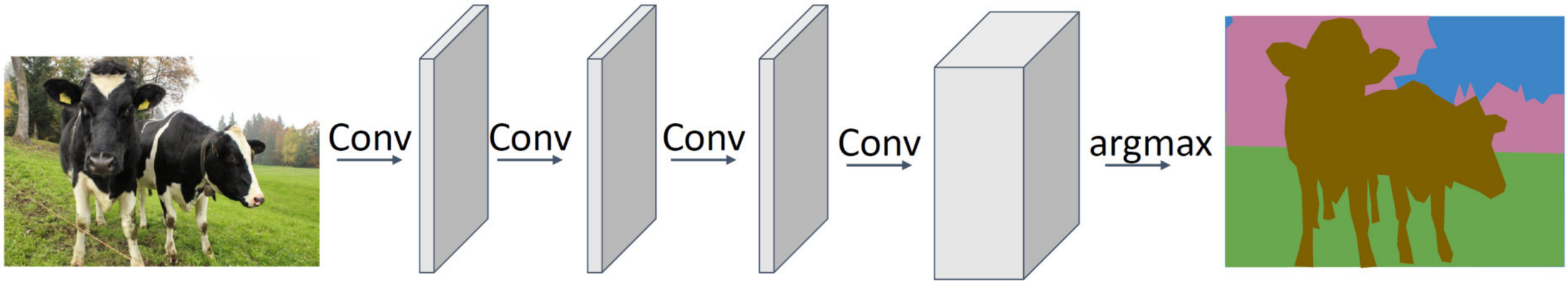
Loss function: Per-Pixel cross-entropy

Long et al, "Fully convolutional networks for semantic segmentation", CVPR 2015

Slide from Deep Learning for Computer Vision @Univ. of Michigan

Semantic Segmentation: Fully Convolutional Network

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



Input:
3 x H x W

Problem #1: Effective receptive field size is linear in number of conv layers: With L 3x3 conv layers, receptive field is 1+2L

Problem #2: Convolution on high res images is expensive! Recall ResNet stem aggressively downsamples

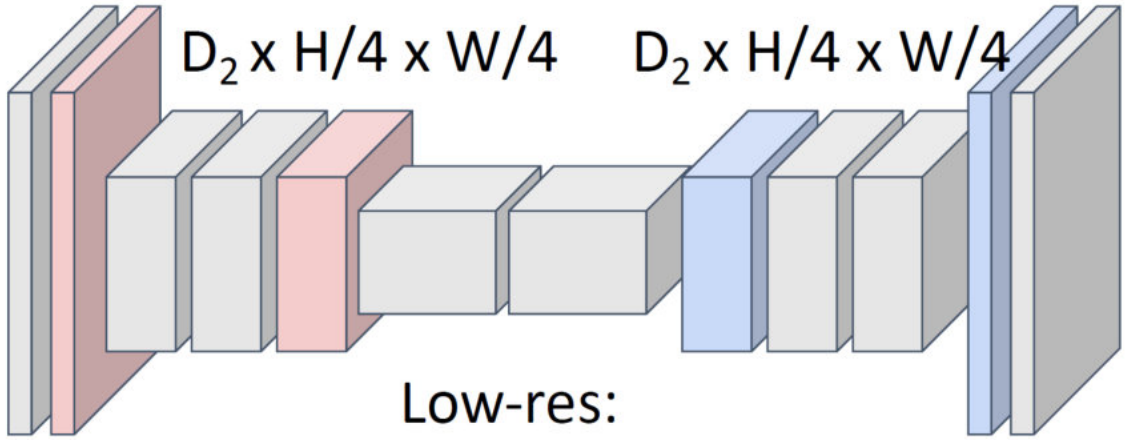
Long et al, "Fully convolutional networks for semantic segmentation", CVPR 2015

Slide from Deep Learning for Computer Vision @Univ. of Michigan

Semantic Segmentation: Fully Convolutional Network

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Input:
3 x H x W



Predictions:
H x W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Slide from Deep Learning for Computer Vision @Univ. of Michigan

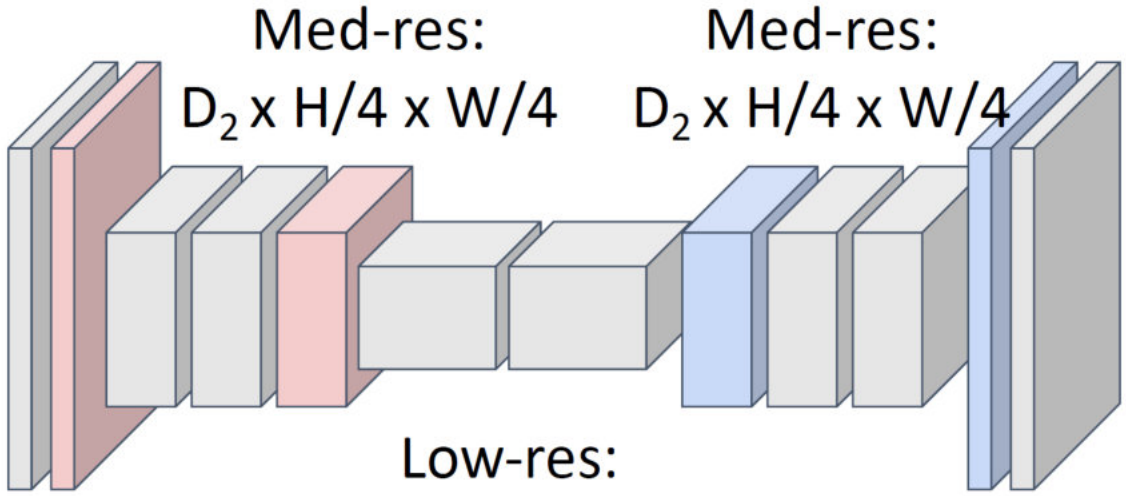
Semantic Segmentation: Fully Convolutional Network

Downsampling: Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling: ???

Input:
3 x H x W

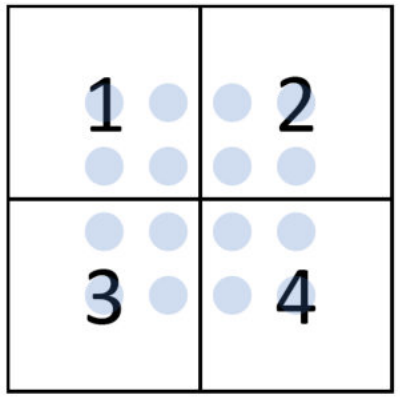


Predictions:
H x W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Slide from Deep Learning for Computer Vision @Univ. of Michigan

In-Network Upsampling: Bilinear Interpolation



1.00	1.25	1.75	2.00
1.50	1.75	2.25	2.50
2.50	2.75	3.25	3.50
3.00	3.25	3.75	4.00

Input: C x 2 x 2

Output: C x 4 x 4

$$f_{x,y} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - i|) \max(0, 1 - |y - j|)$$

$$i \in \{\lfloor x \rfloor - 1, \ldots, \lceil x \rceil + 1\}$$
$$j \in \{\lfloor y \rfloor - 1, \ldots, \lceil y \rceil + 1\}$$

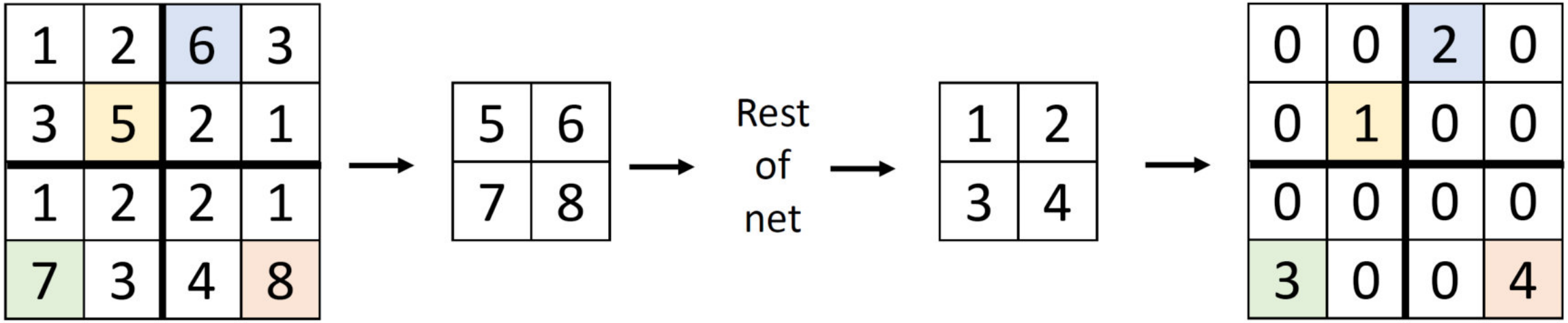
Use two closest neighbors in x and y
to construct linear approximations

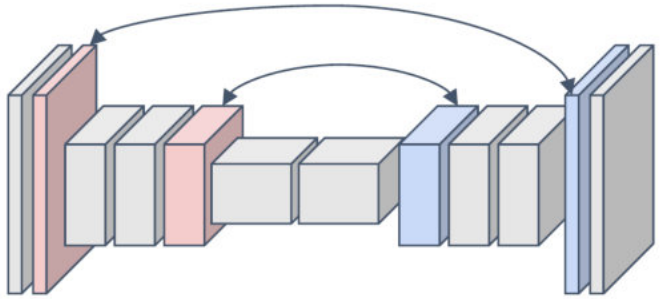
Slide from Deep Learning for Computer Vision @Univ. of Michigan

In-Network Upsampling: "Max Unpooling"

Max Pooling: Remember which position had the max

Max Unpooling: Place into remembered positions





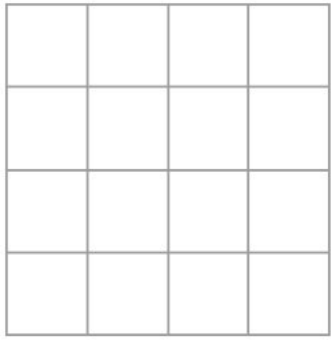
Pair each downsampling layer with an upsampling layer

Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

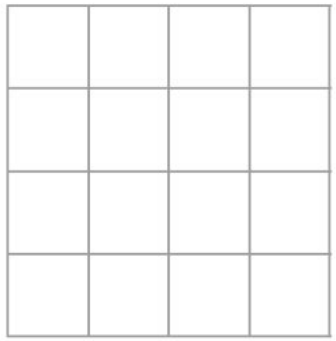
Slide from Deep Learning for Computer Vision @Univ. of Michigan

Learnable Upsampling: “Deconvolution”

Typical 3 x 3 convolution, stride 1 pad 1



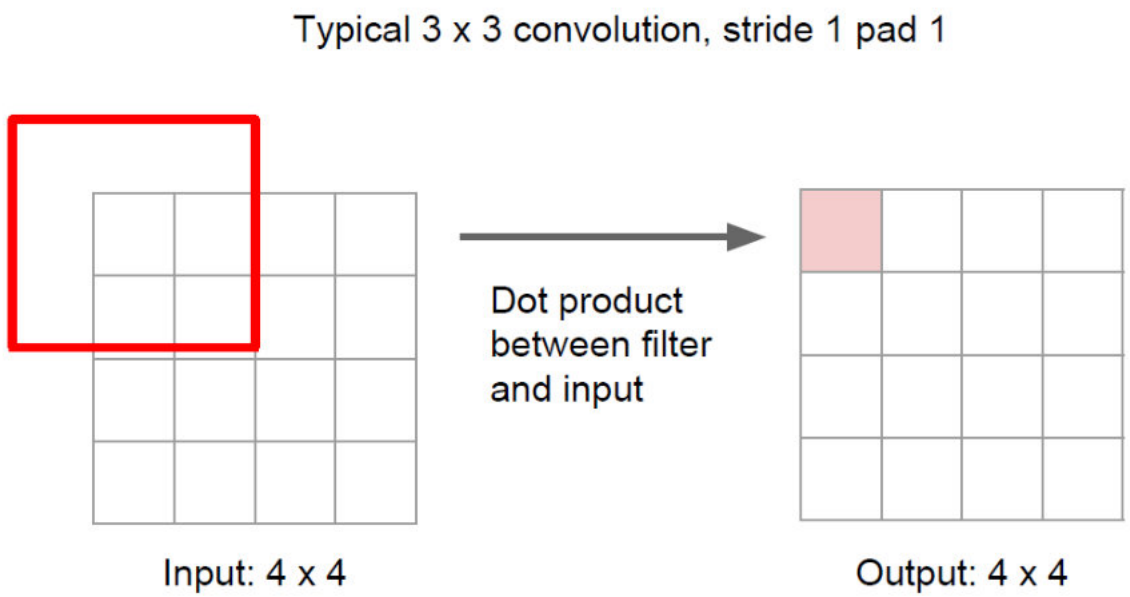
Input: 4 x 4



Output: 4 x 4

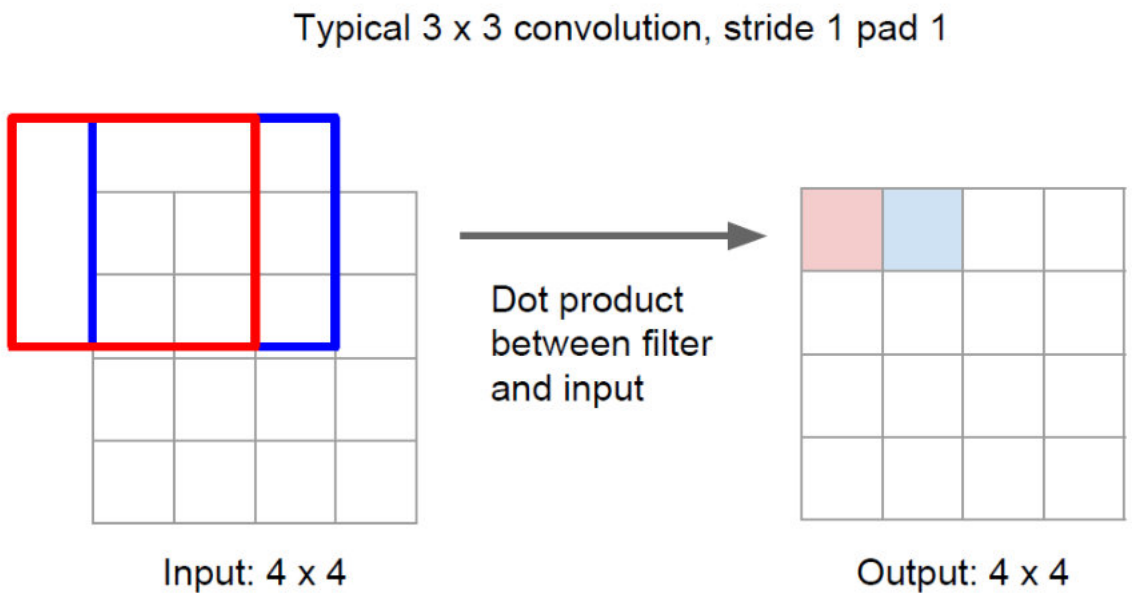
Slide from Stanford CS231n

Learnable Upsampling: “Deconvolution”



Slide from Stanford CS231n

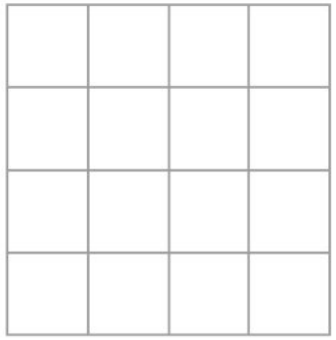
Learnable Upsampling: “Deconvolution”



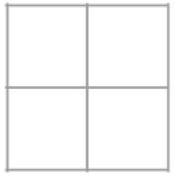
Slide from Stanford CS231n

Learnable Upsampling: “Deconvolution”

Typical 3 x 3 convolution, **stride 2** pad 1



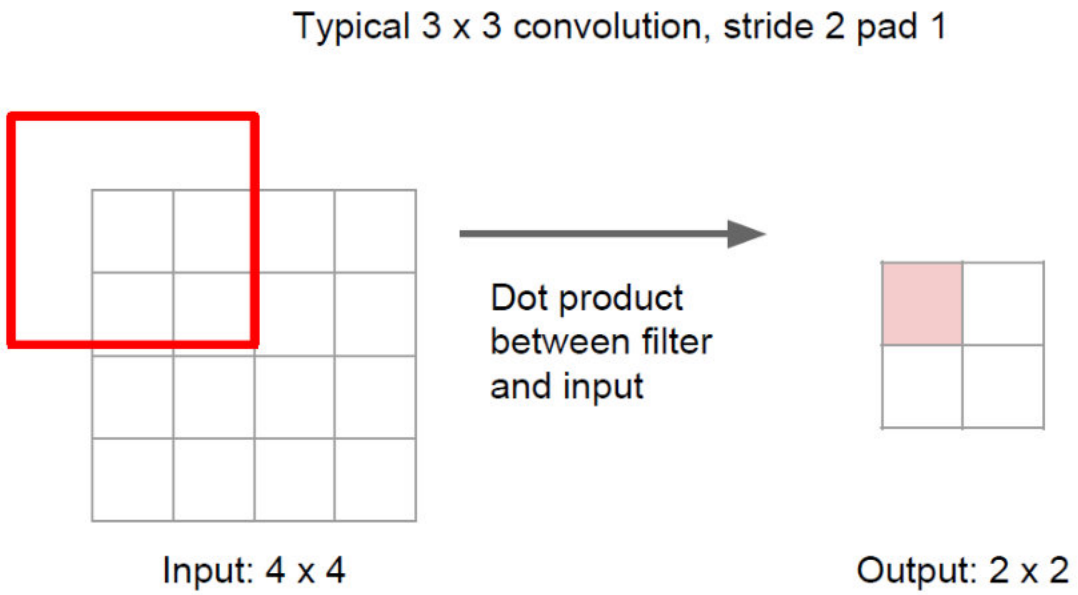
Input: 4 x 4



Output: 2 x 2

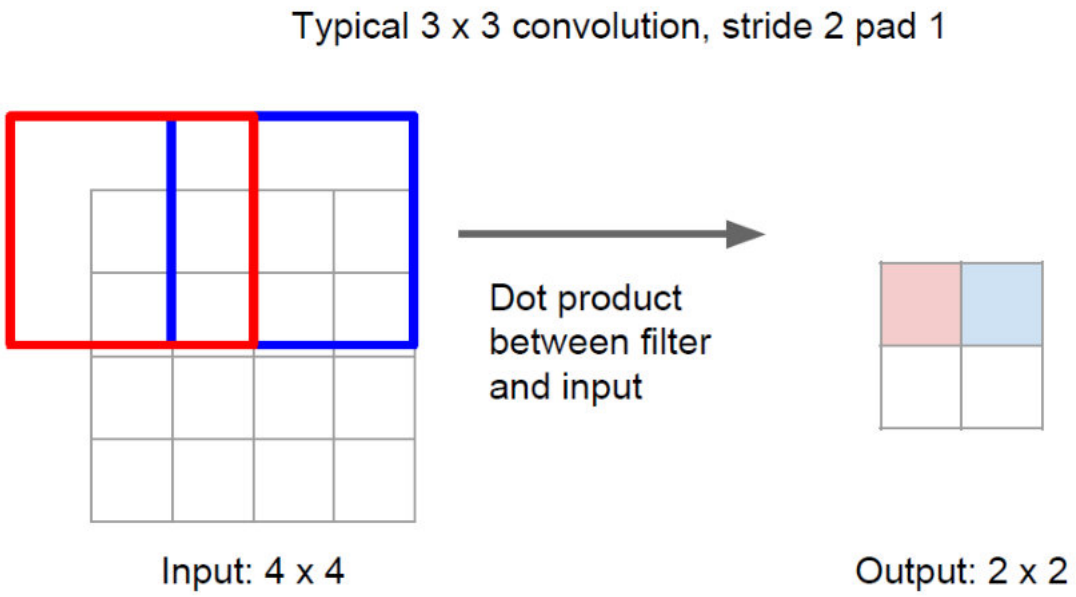
Slide from Stanford CS231n

Learnable Upsampling: “Deconvolution”



Slide from Stanford CS231n

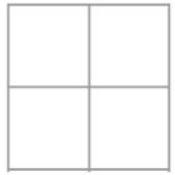
Learnable Upsampling: “Deconvolution”



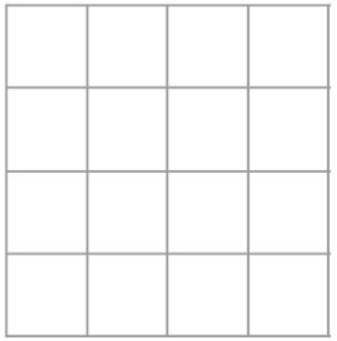
Slide from Stanford CS231n

Learnable Upsampling: “Deconvolution”

3 x 3 “deconvolution”, stride 2 pad 1



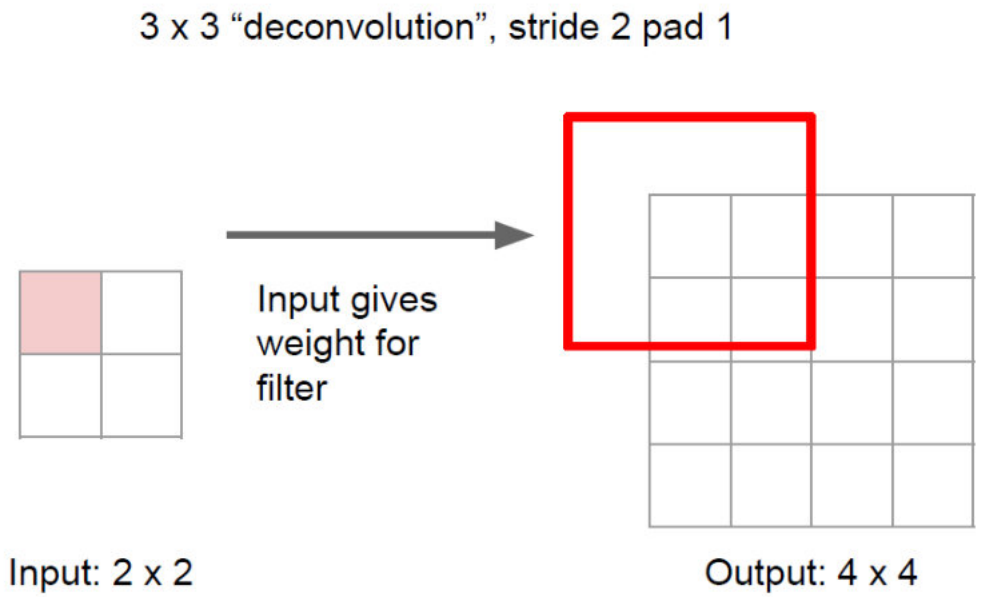
Input: 2 x 2



Output: 4 x 4

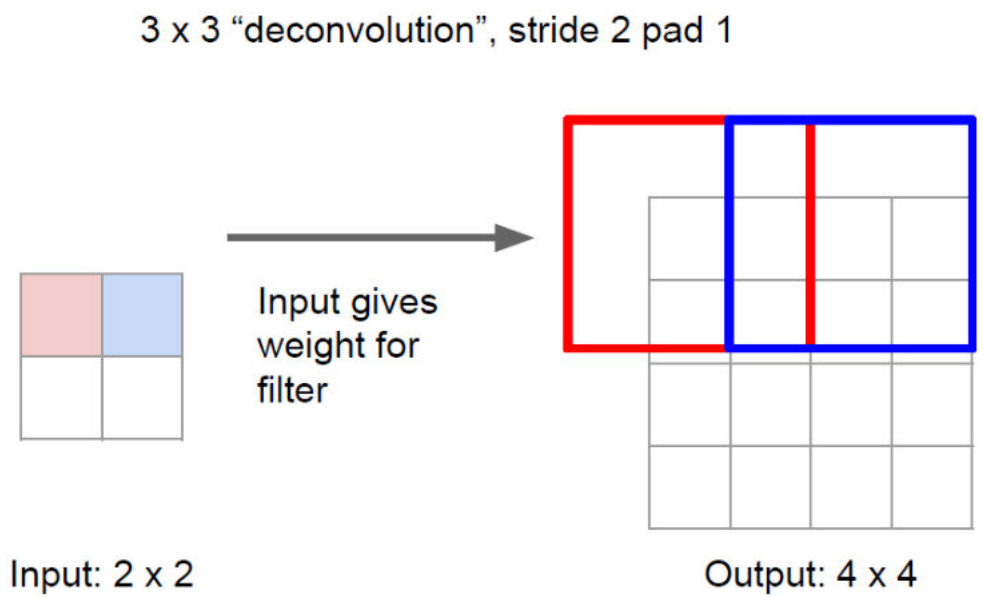
Slide from Stanford CS231n

Learnable Upsampling: “Deconvolution”



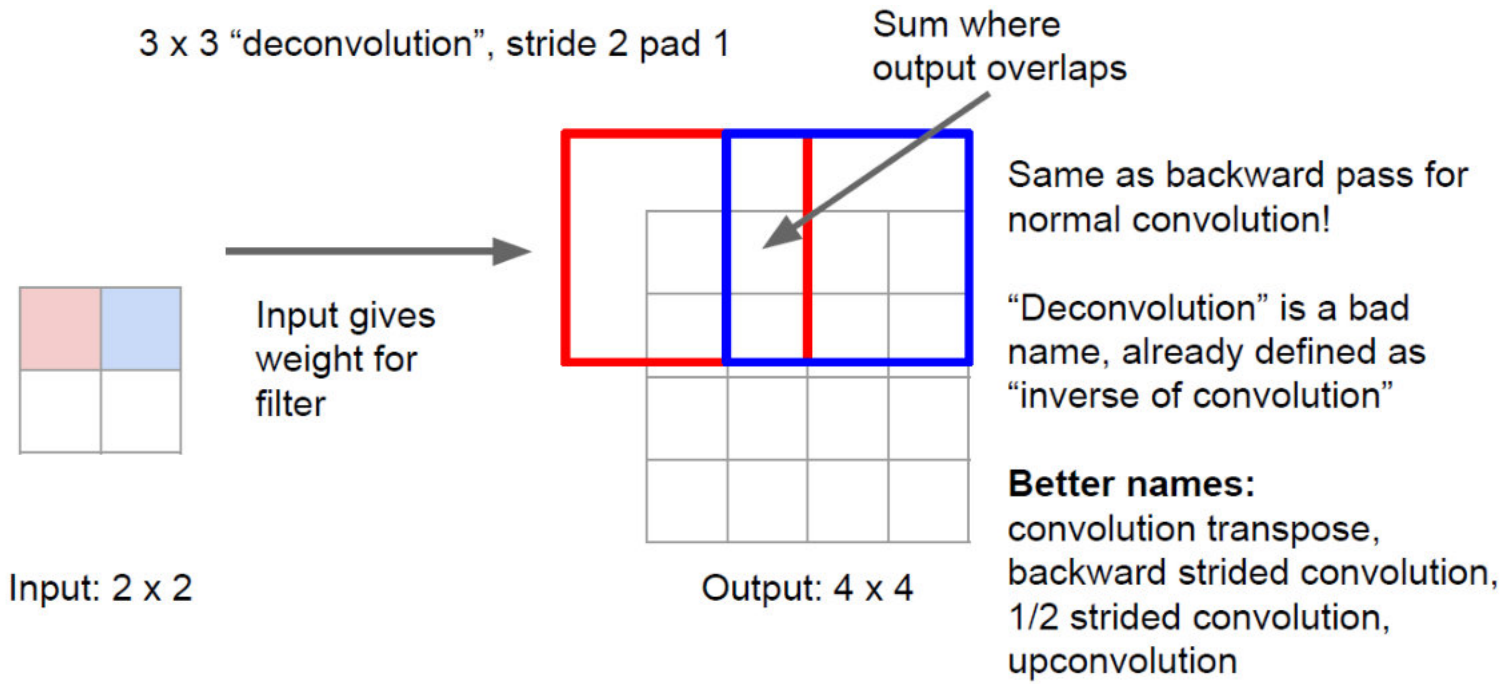
Slide from Stanford CS231n

Learnable Upsampling: “Deconvolution”



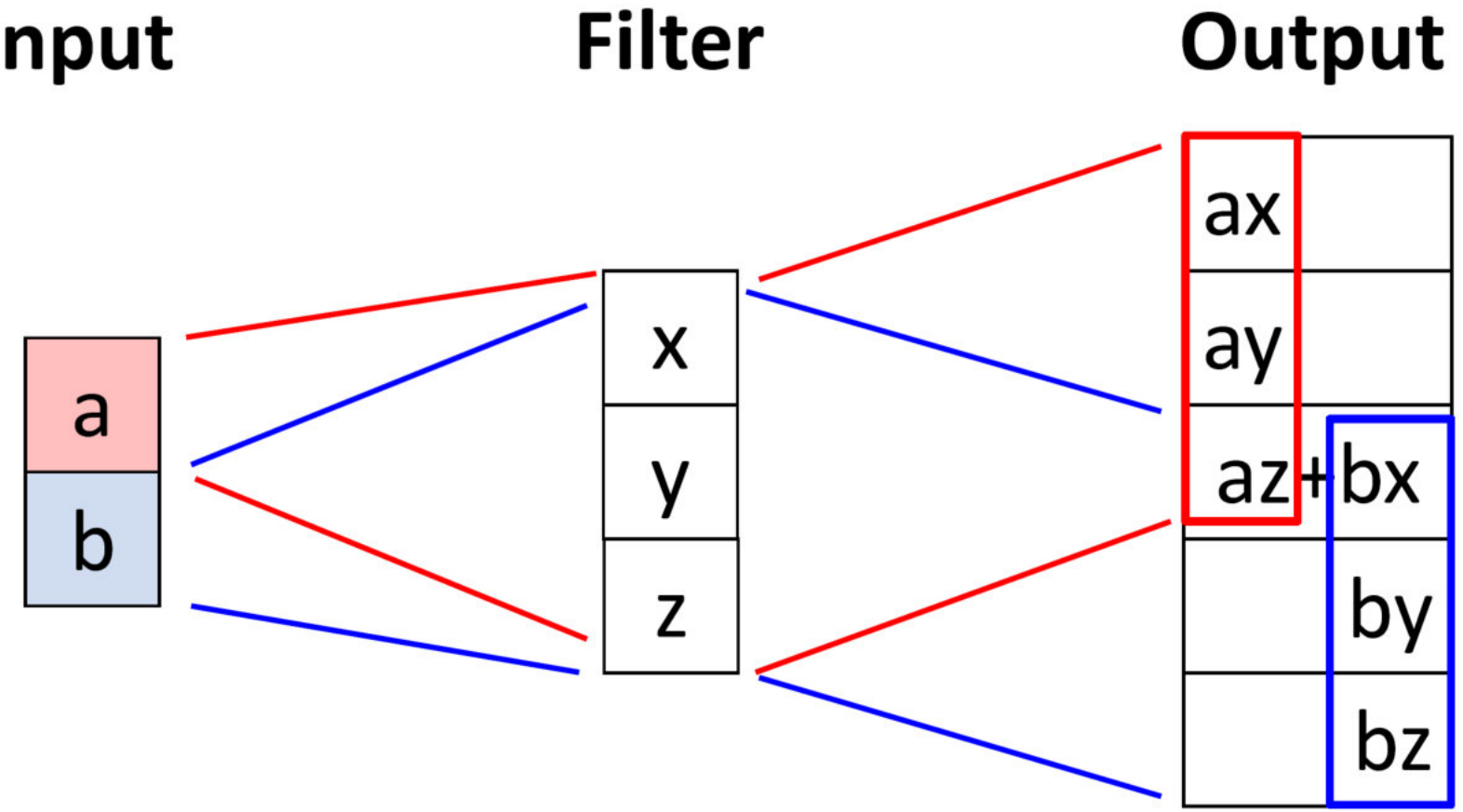
Slide from Stanford CS231n

Learnable Upsampling: "Deconvolution"



Slide from Stanford CS231n

Transposed Convolution: 1D example



This has many names:

- Deconvolution (bad)!
- Upconvolution
- Fractionally strided convolution
- Backward strided convolution
- Transposed Convolution (best name)

Slide from Deep Learning for Computer Vision @Univ. of Michigan

U-Net: Convolutional Networks for Biomedical Image Segmentation

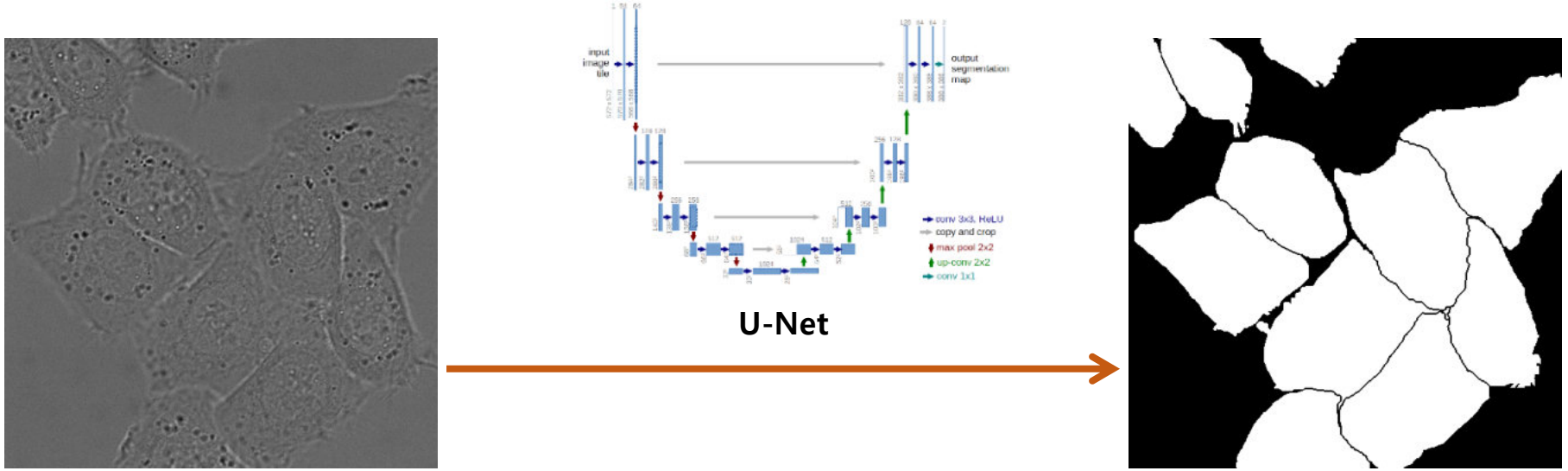
U-Net: Convolutional Networks for Biomedical Image Segmentation

Olaf Ronneberger, Philipp Fischer, and Thomas Brox

Computer Science Department and BIOSS Centre for Biological Signalling Studies,
University of Freiburg, Germany
ronneber@informatik.uni-freiburg.de,
WWW home page: http://lmb.informatik.uni-freiburg.de/

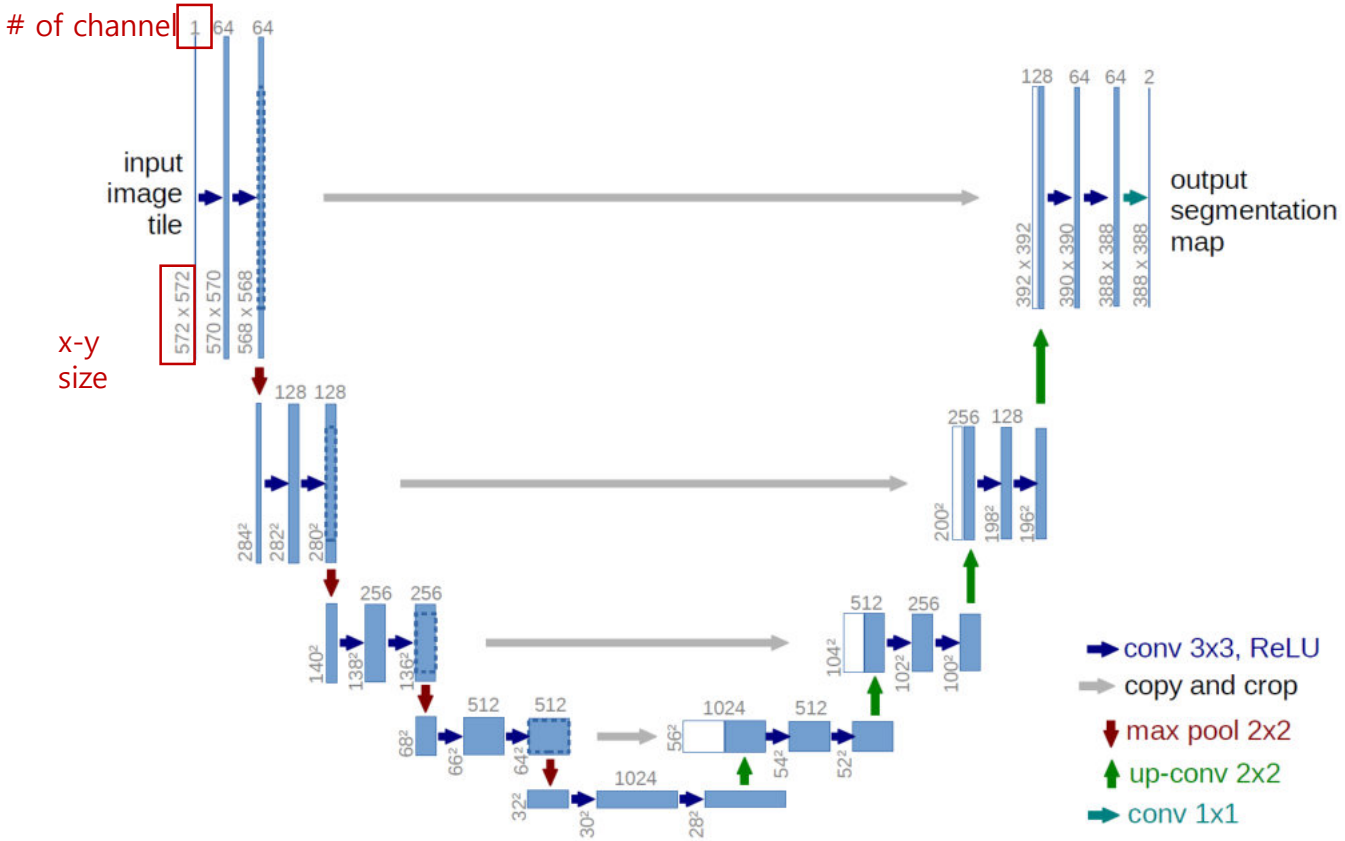
Abstract. There is large consent that successful training of deep networks requires many thousand annotated training samples. In this paper, we present a network and training strategy that relies on the strong use of data augmentation to use the available annotated samples more efficiently. The architecture consists of a contracting path to capture context and a symmetric expanding path that enables precise localization. We show that such a network can be trained end-to-end from very few images and outperforms the prior best method (a sliding-window

Biomedical Image Segmentation with U-net

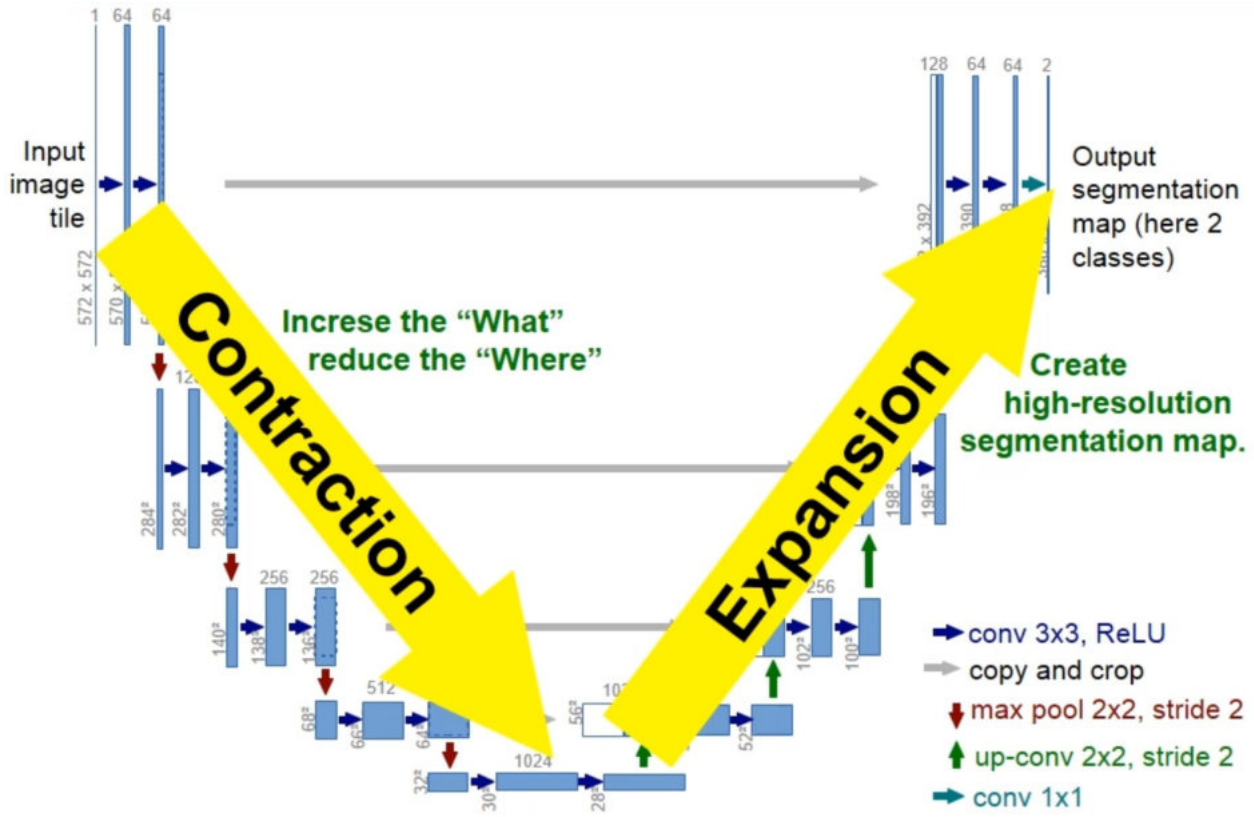


- U-Net learns segmentation in an end-to-end setting
- Very few annotated images (approx. 30 per application)

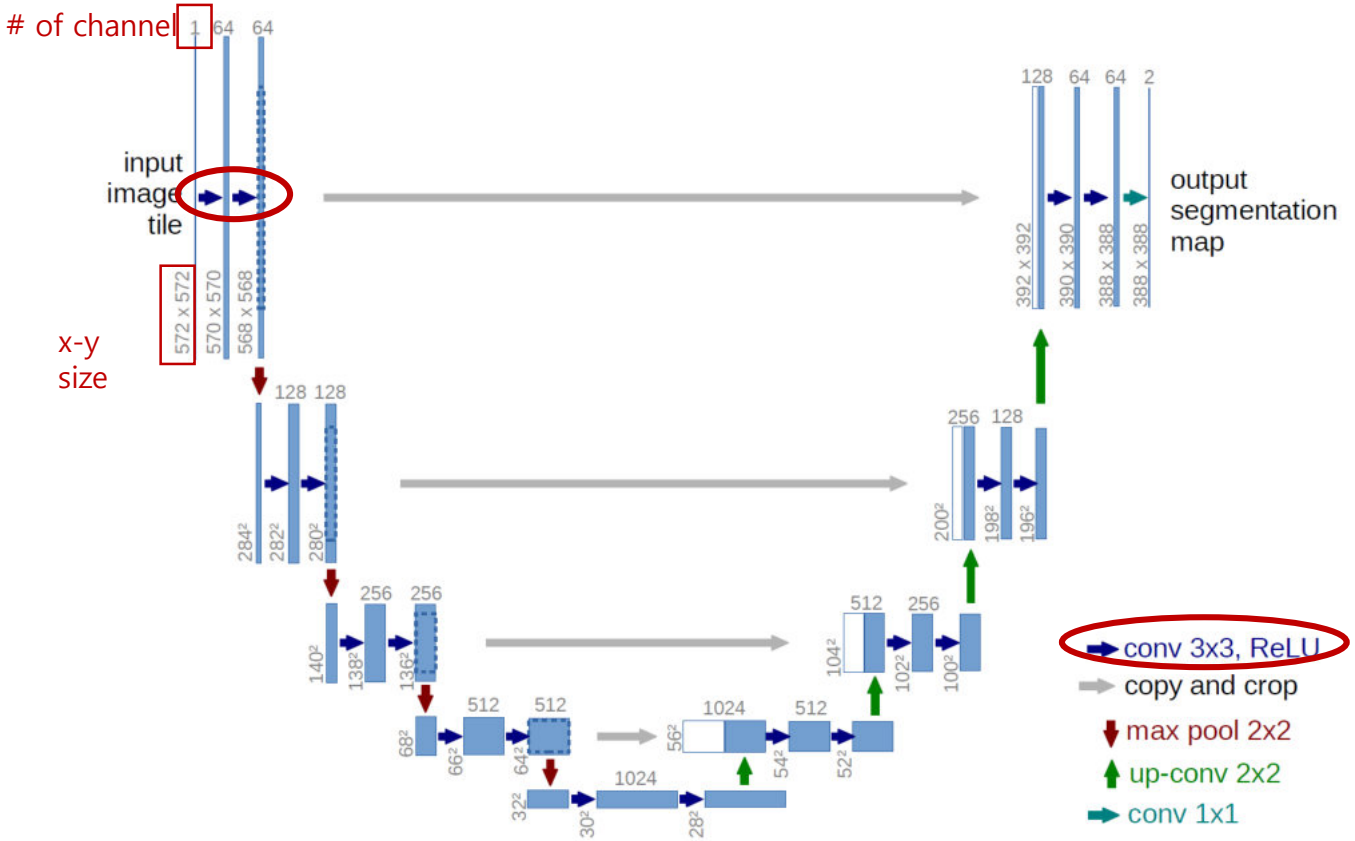
U-Net Architecture



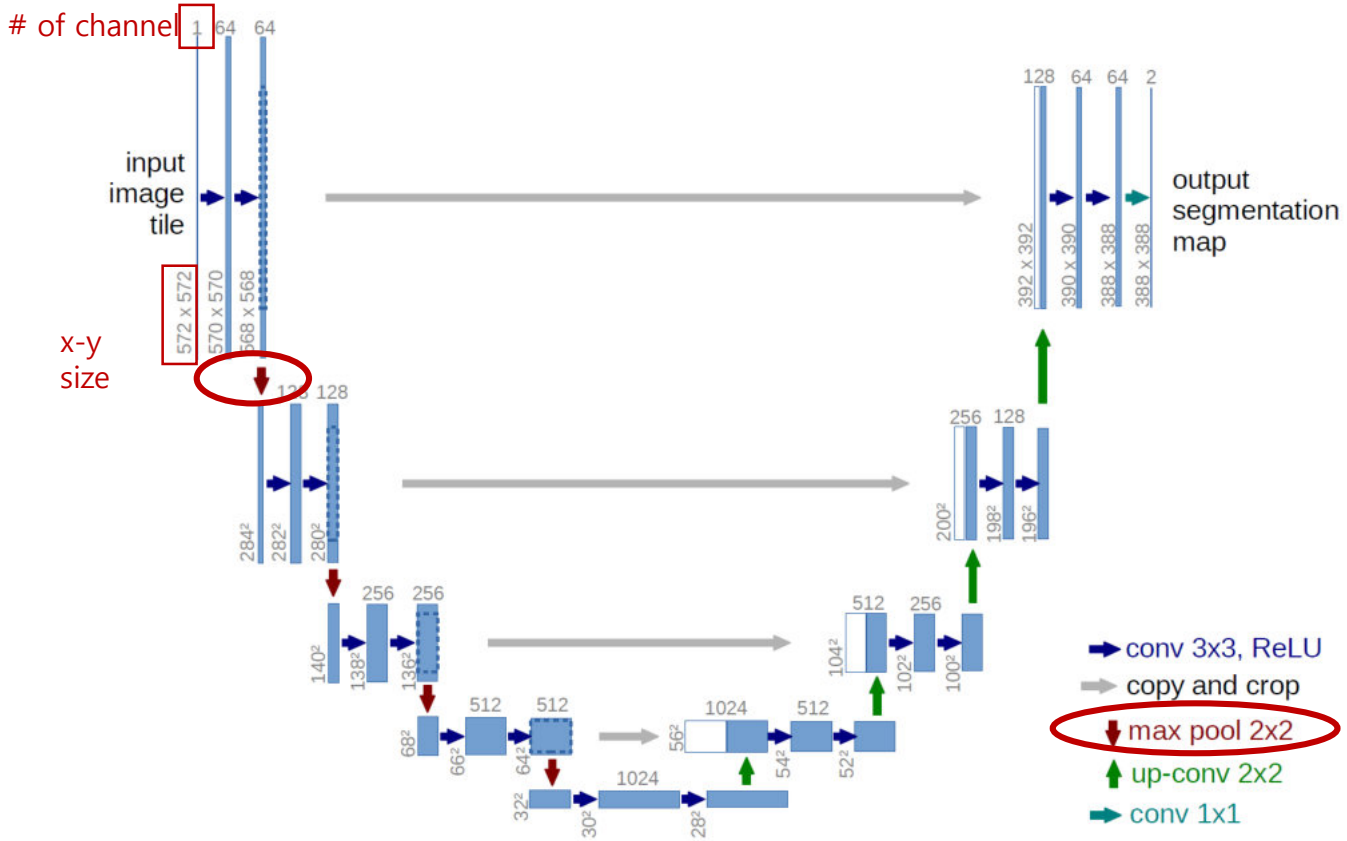
U-Net Architecture



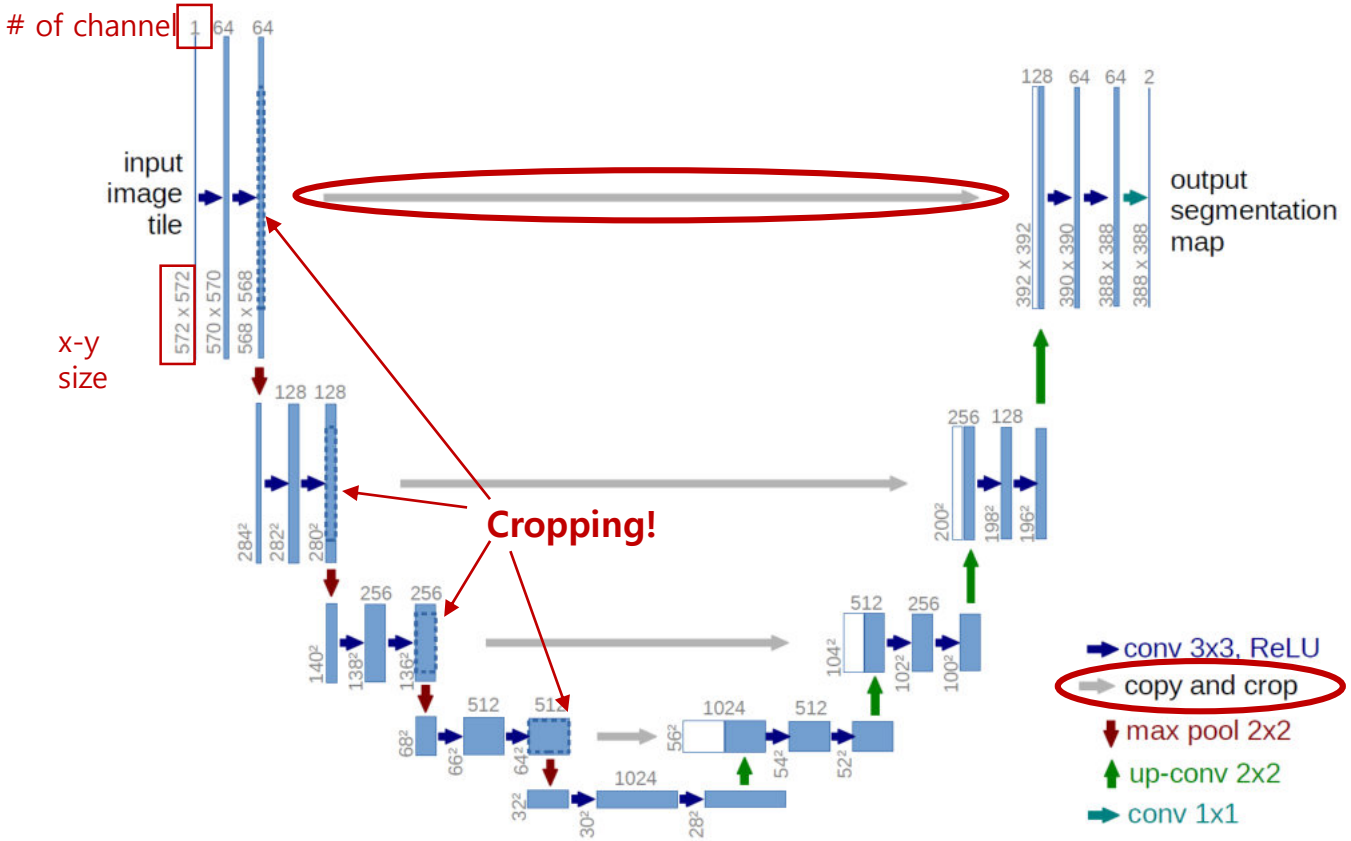
U-Net Architecture



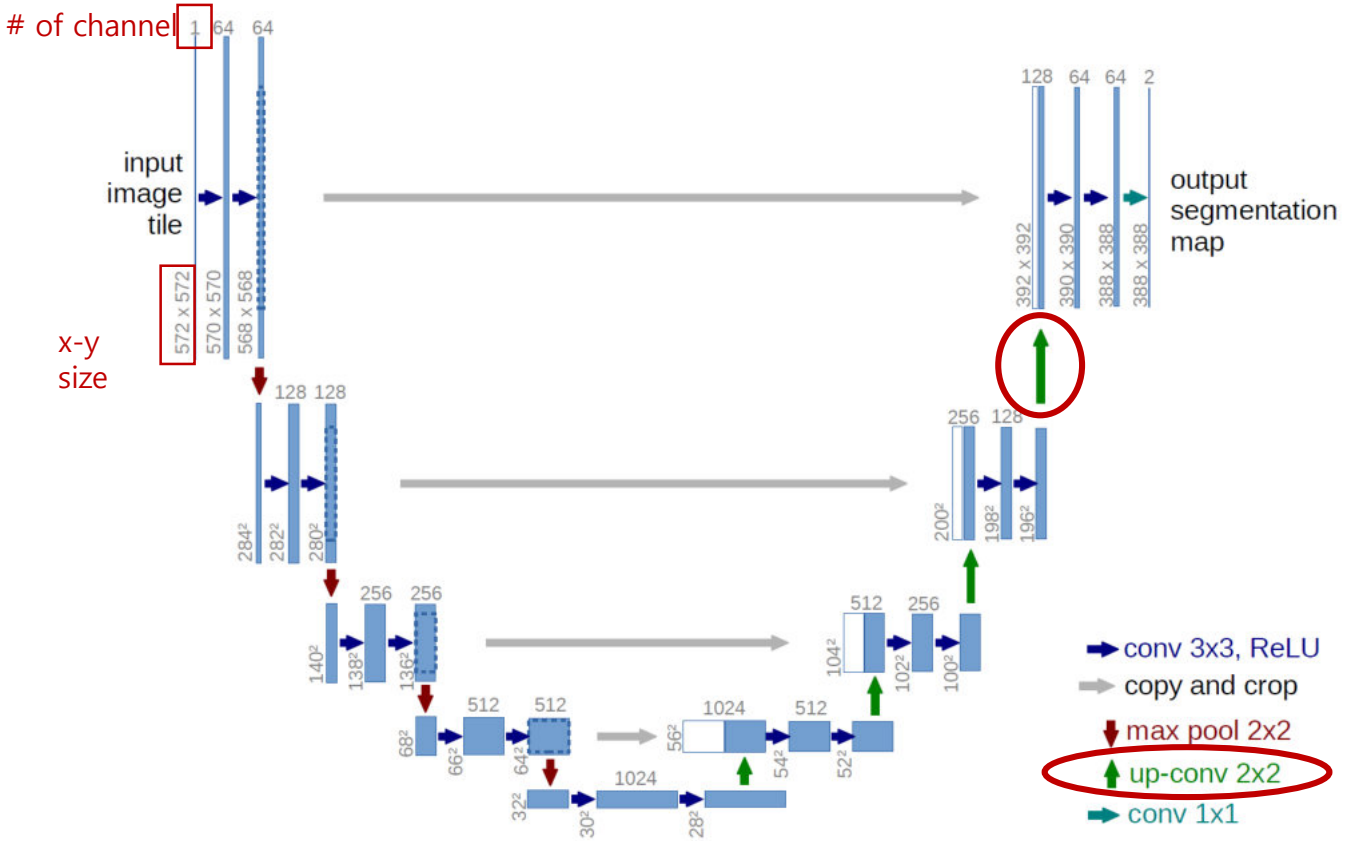
U-Net Architecture



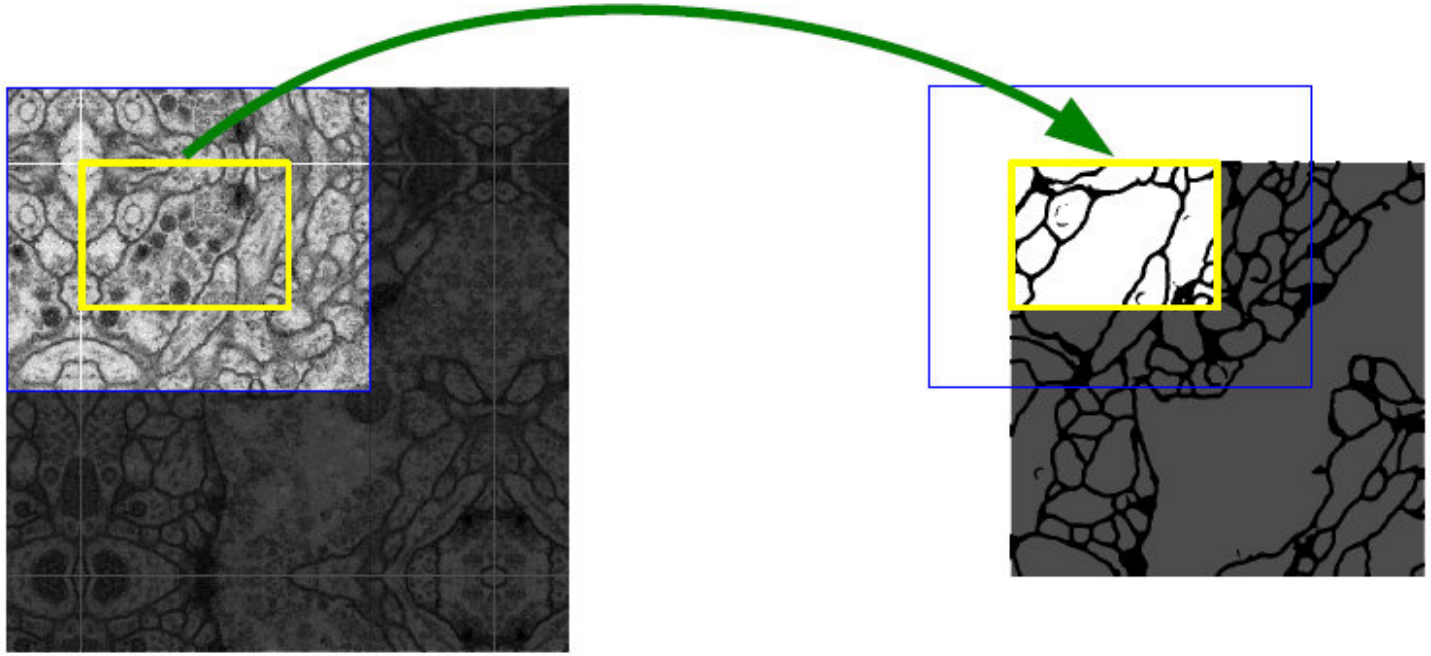
U-Net Architecture



U-Net Architecture

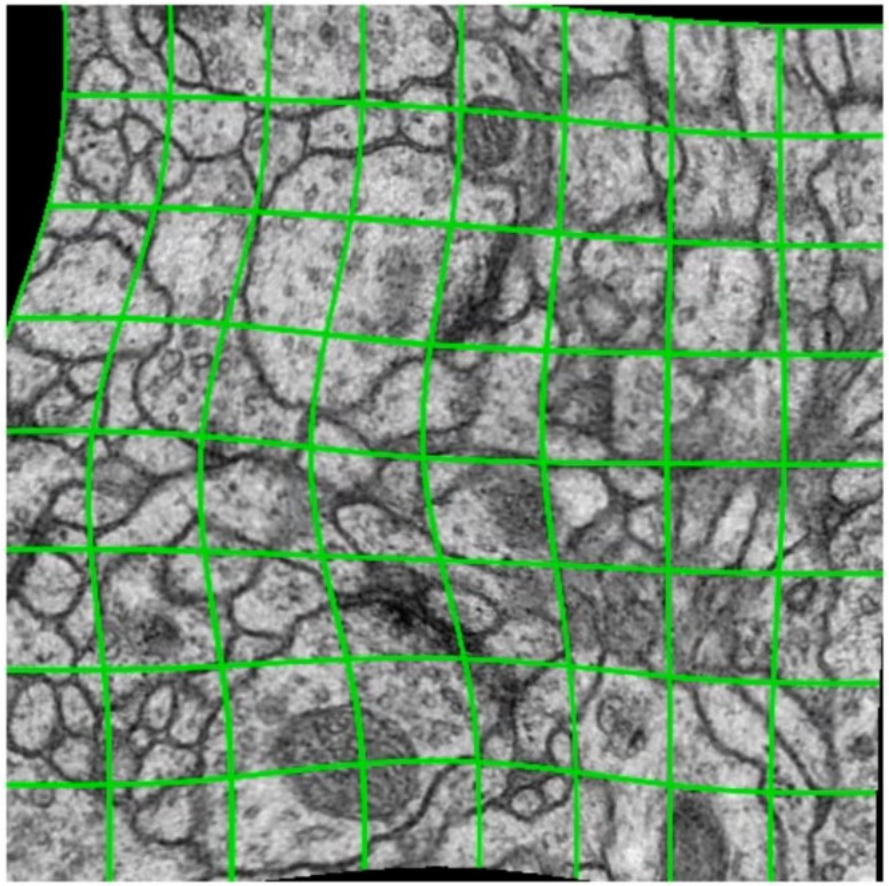


Overlap-tile strategy for arbitrary large images

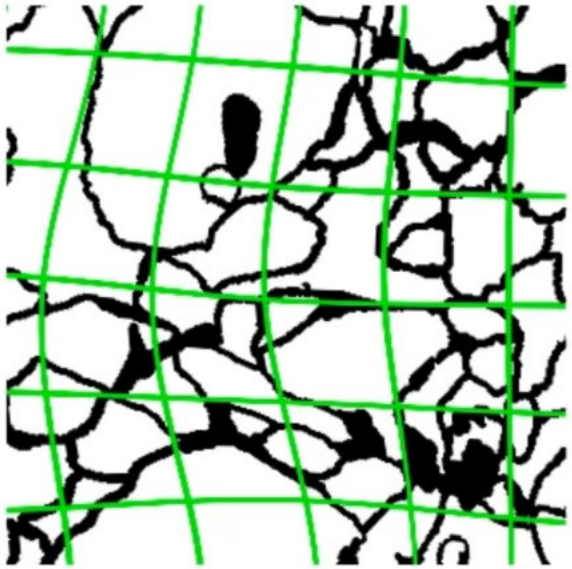


- Segmentation of the yellow area uses input data of the blue area
- Raw data extrapolation by mirroring

Augment Training Data using Deformations

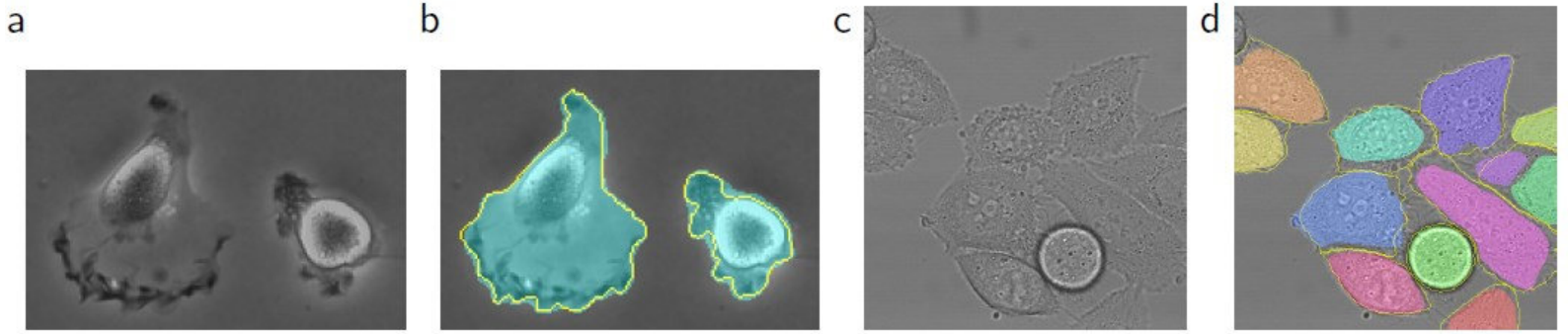


resulting deformed image
(for visualization: no rotation, no shift, no extrapolation)



correspondingly deformed
manual labels

ISBI 2015 Result



Name	PhC-U373	DIC-HeLa
IMCB-SG (2014)	0.2669	0.2935
KTH-SE (2014)	0.7953	0.4607
HOUS-US (2014)	0.5323	-
second-best 2015	0.83	0.46
u-net (2015)	**0.9203**	**0.7756**