Object Detection – Part 2

Danna Gurari
University of Colorado Boulder
Fall 2023

Review

- Last lecture:
  - Object Detection Problem
  - Object Detection Applications
  - Object Detection Datasets
  - Object Detection Evaluation Metric
  - Overview of object detection algorithms and baseline (R-CNN)
- Assignments (Canvas)
  - Reading assignment was due earlier today
  - Next reading assignment due next Wednesday (guest lecture for Monday)
  - Project proposal due in two weeks
- Questions?

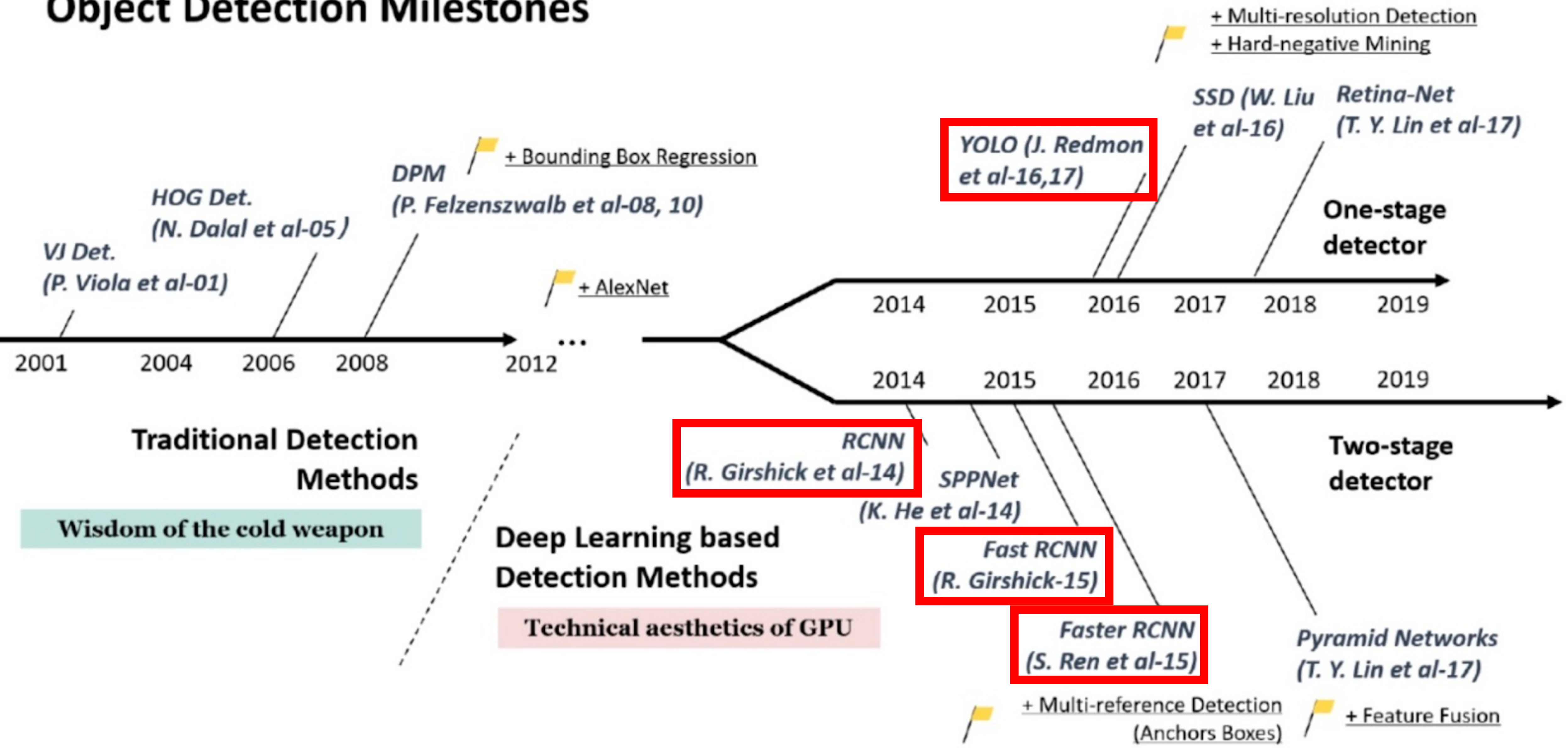
Object Detection: Today’s Topics

- Overview of object detection algorithms
- Fast R-CNN
- Faster R-CNN
- YOLO
- Discussion

Recall Motivation: Go **Faster** While Getting Good Accuracy

Image Source: https://yourboulder.com/boulder-neighborhood-downtown/

Object Detection Milestones

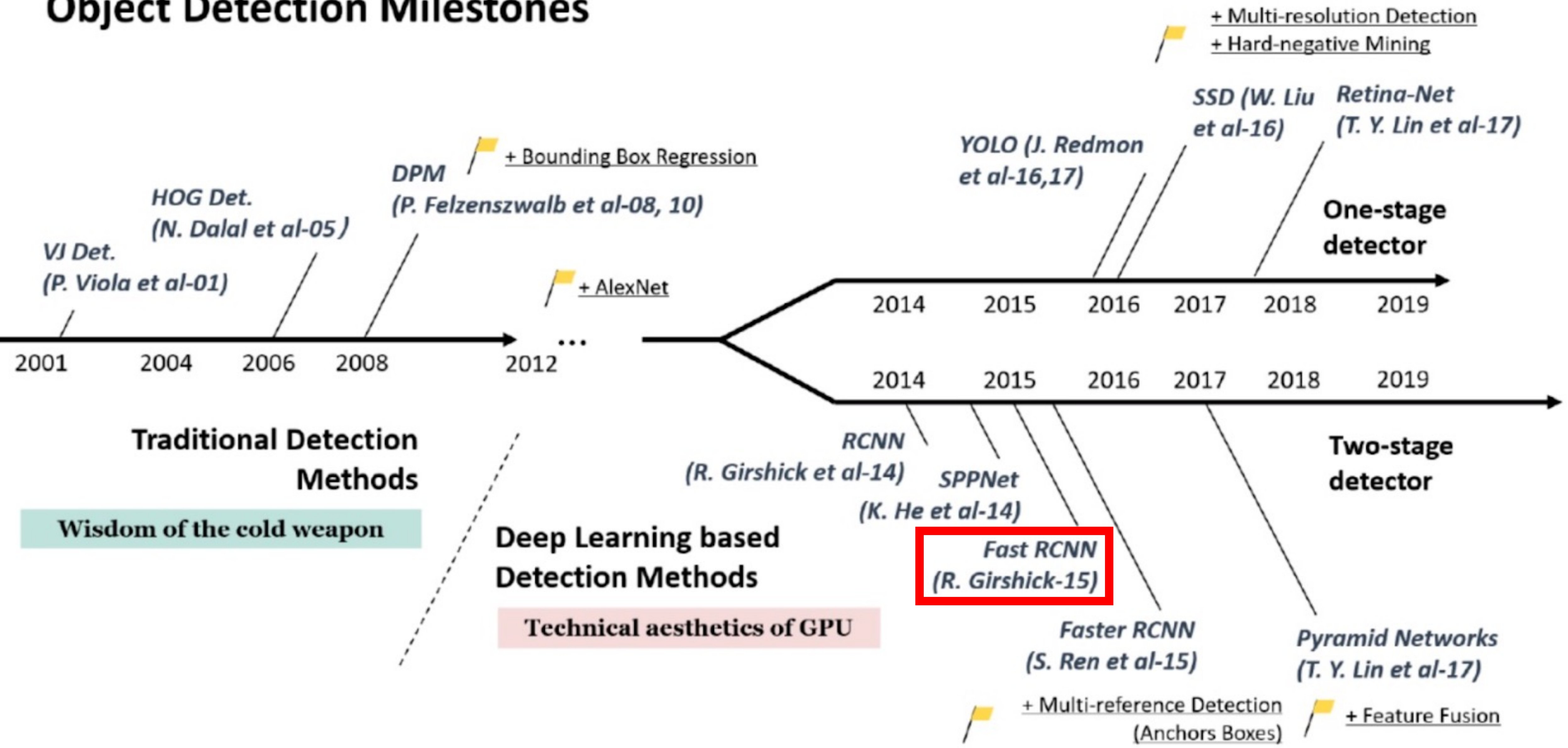


Zhou et al. Object Detection in 20 Years: A Survey. arXiv 2019.

Object Detection: Today's Topics

- Overview of object detection algorithms
- Fast R-CNN
- Faster R-CNN
- YOLO
- Discussion

Object Detection Milestones



Zhou et al. Object Detection in 20 Years: A Survey. arXiv 2019.

Key Contributions of Fast R-CNN

1. State of art object detection model in terms of accuracy and speed
    1. 9x faster than R-CNN
    2. mAP of 66% vs 62% for R-CNN on VOC2012

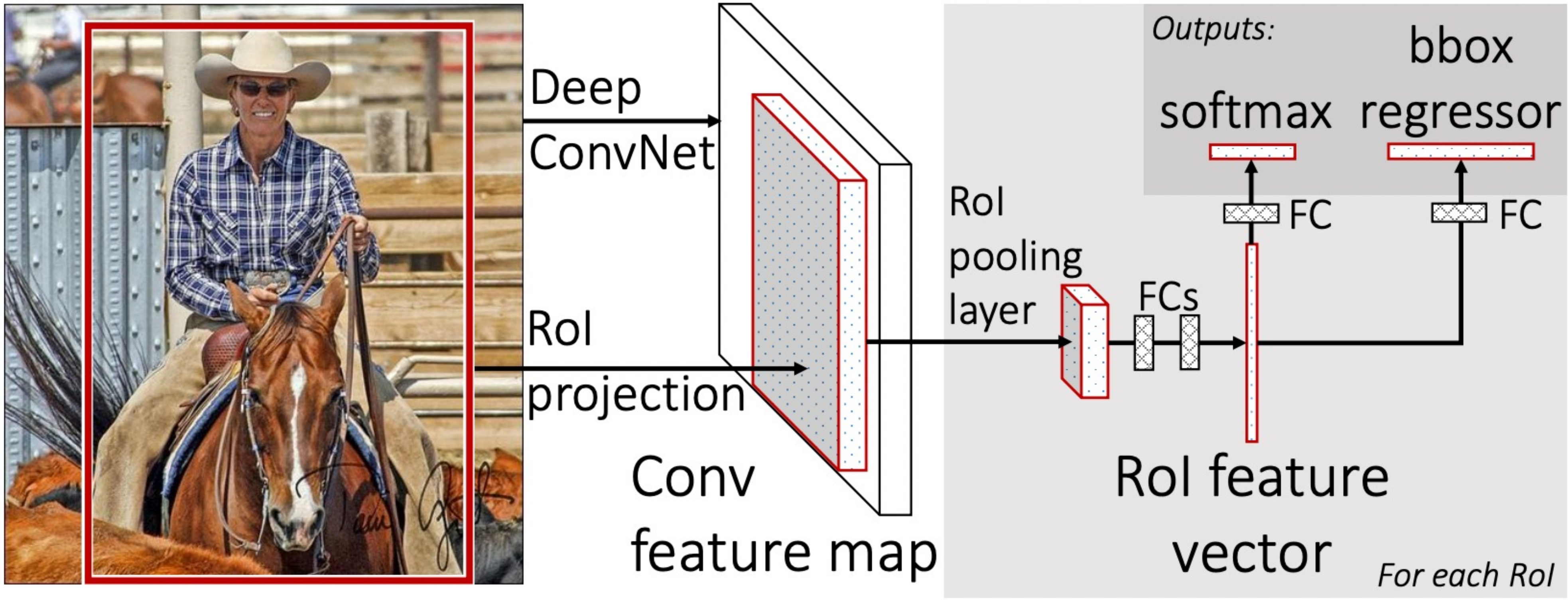
2. Reduced storage requirements by not requiring features to be stored for each region proposal

3. A training algorithm that learns in a single stage (rather than the three stages required by R-CNN)

Architecture

(Also, like R-CNN, apply non-maximum suppression to remove redundant predictions for the same object)

Like R-CNN, assign each region proposal to a class and refine it

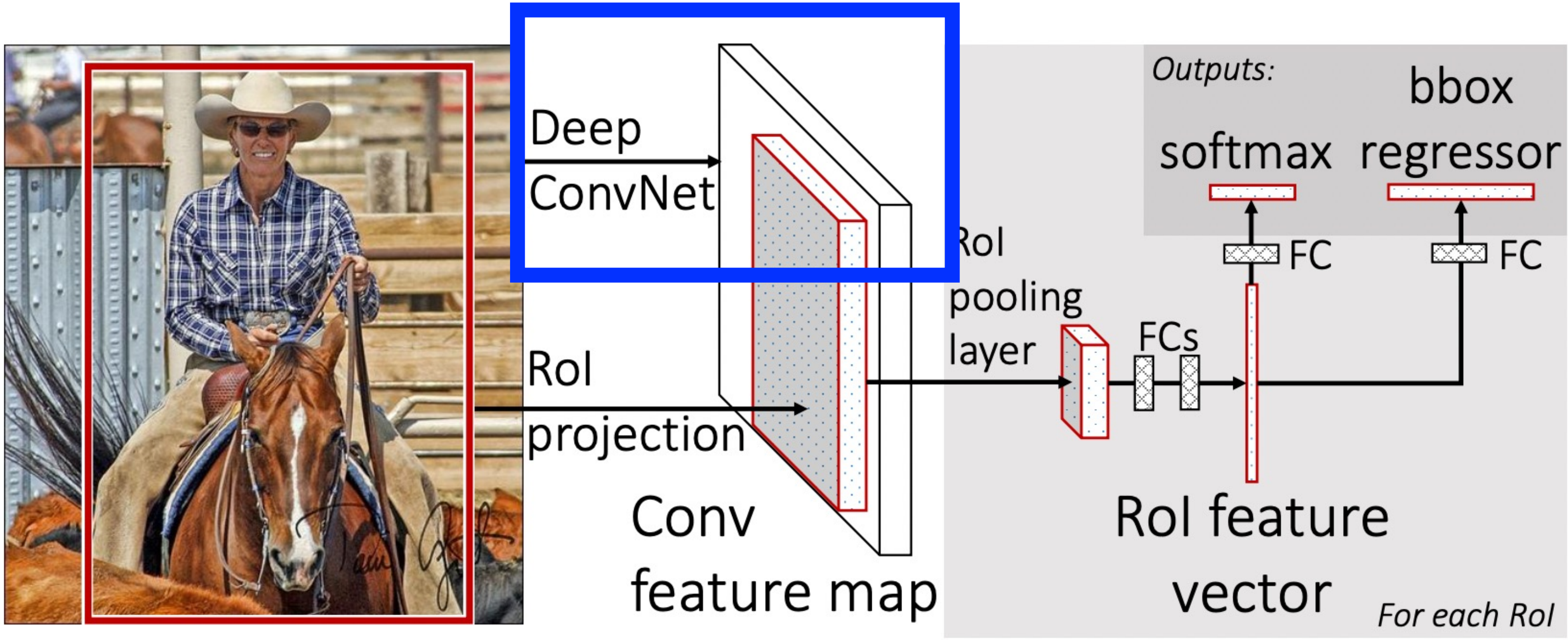


Input: any image size + "object"-like regions found using selective search (same as R-CNN)

Ross Girshick. Fast R-CNN. CVPR 2015.

Architecture

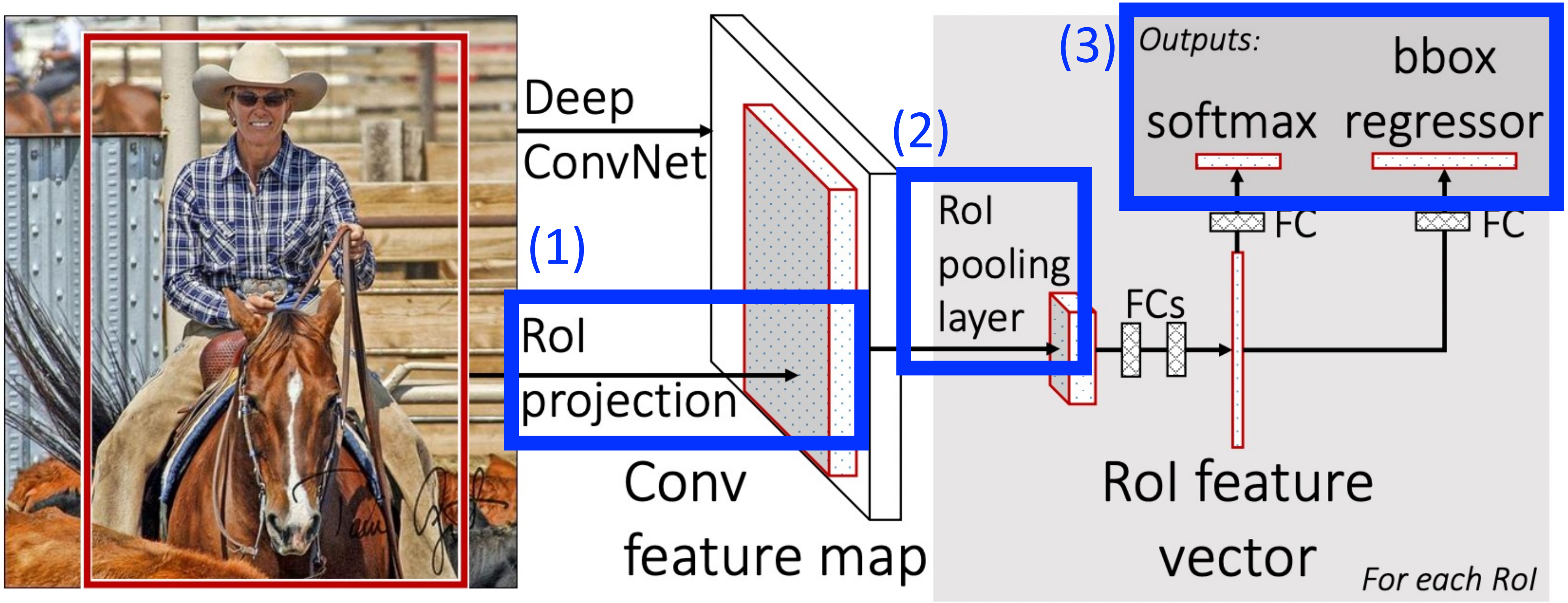
Image is fed through a pretrained ImageNet model to generate a feature map for the **entire image**. Which models could we use?
- Was tested with AlexNet (small), VGG16 (large), and another VGG variant



Ross Girshick. Fast R-CNN. CVPR 2015.

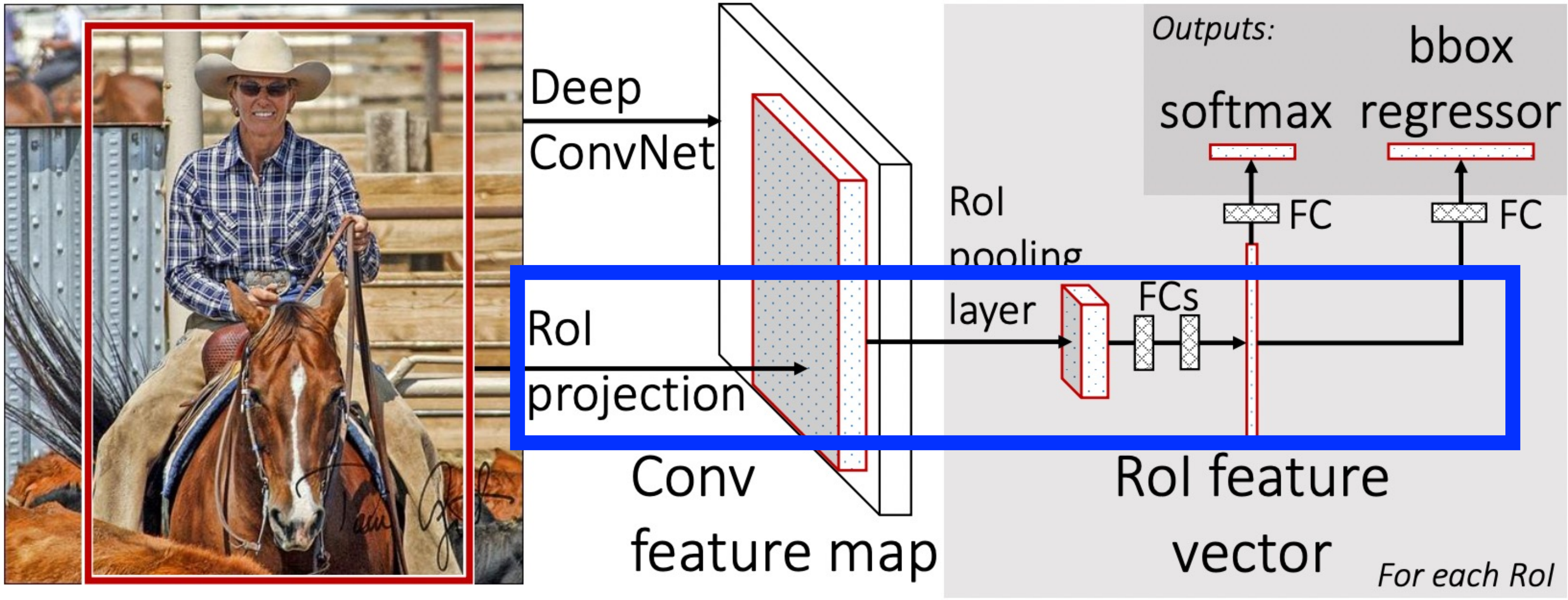
Architecture

Each pretrained model is modified to: (1) update input to accept list of region proposals, (2) use ROI pooling layer to support arbitrary image sizes, and (3) use output layer for object detection instead of image classification



Ross Girshick. Fast R-CNN. CVPR 2015.

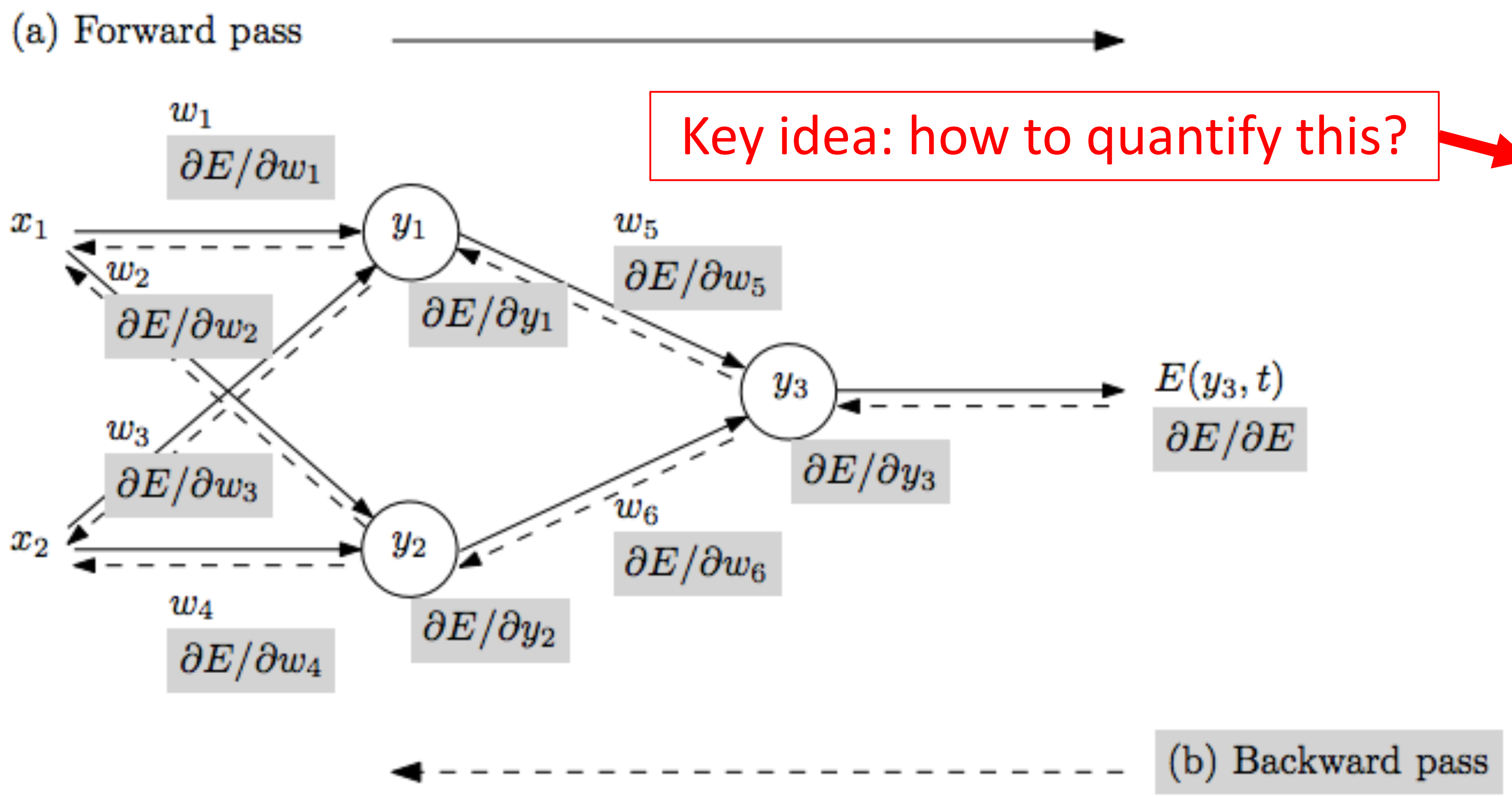
Architecture



The modified input enables passing only the portion of the feature map inside each region proposal for prediction

Ross Girshick. Fast R-CNN. CVPR 2015.

Training in One Stage



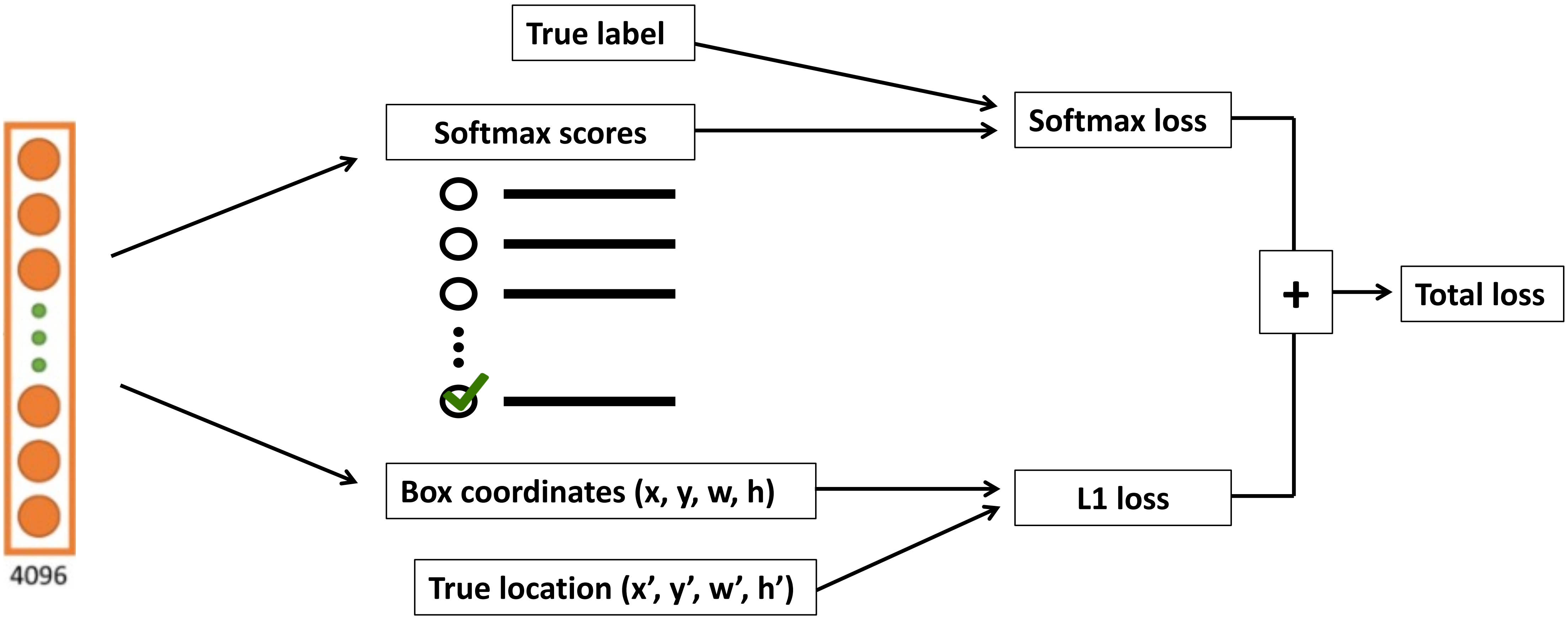
Key idea: how to quantify this?

- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make predictions
  2. **Error quantification**: measure dissatisfaction with a model's predictions on training data
  3. **Backward pass**: using predicted output, calculate gradients backward to assign blame to each model parameter; account for weight sharing by using average of all connections for a parameter
  4. Update each parameter using calculated gradients

https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

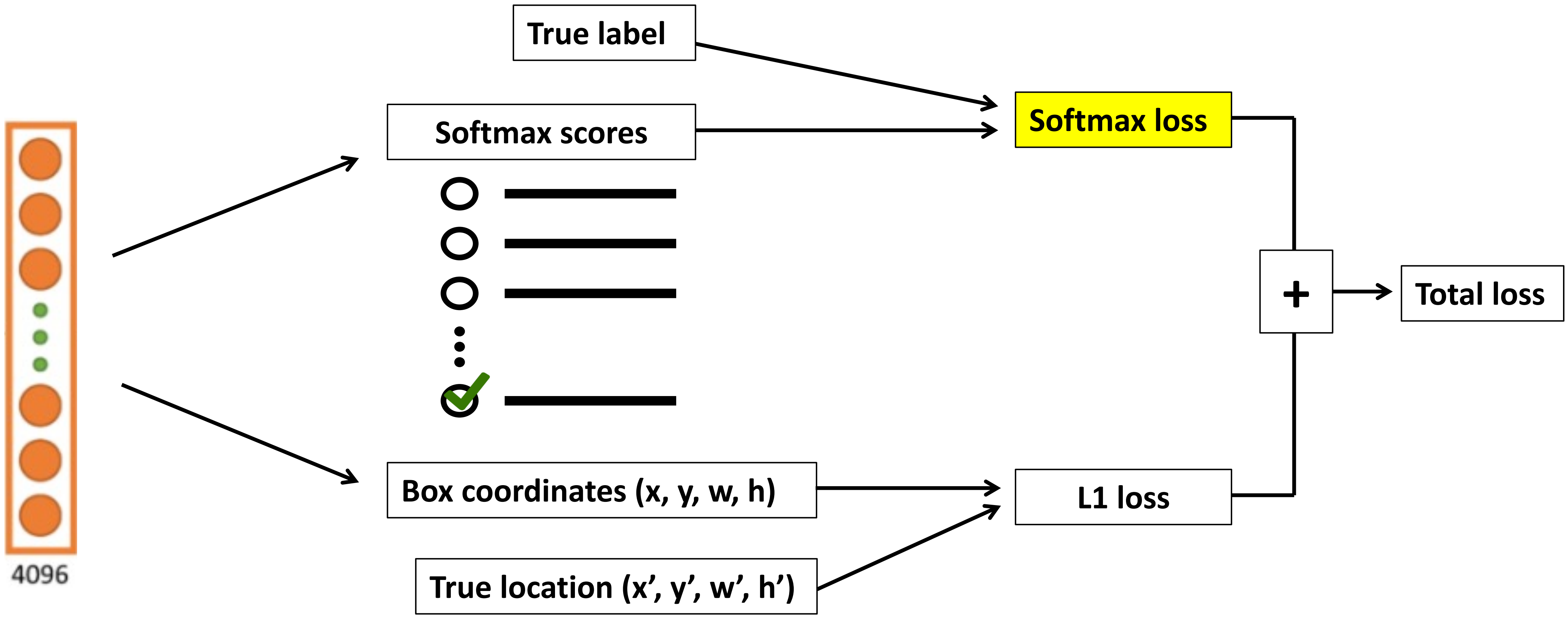
Training in **One Stage**: Multi-task Loss

Objective function sums classification and localization losses for each region proposal



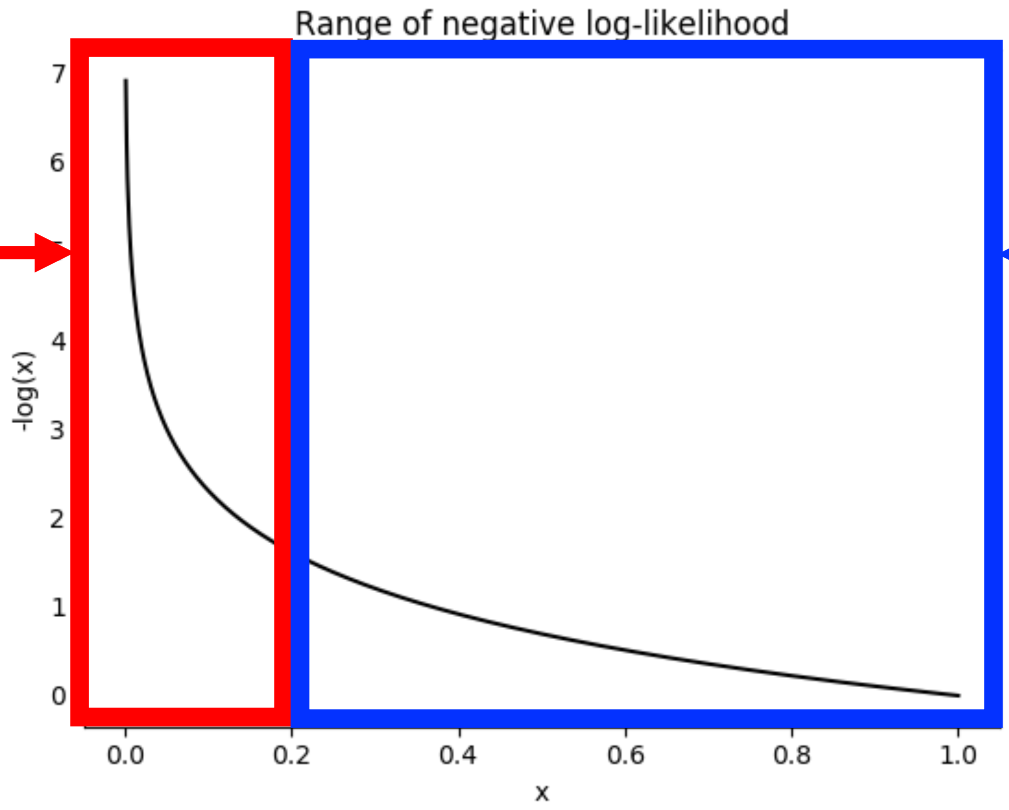
Training in **One Stage**: Multi-task Loss

Objective function sums classification and localization losses for each region proposal



Classification Loss (Recall Cross Entropy Loss)

Greater penalty when predicted probability of true class is confidently wrong



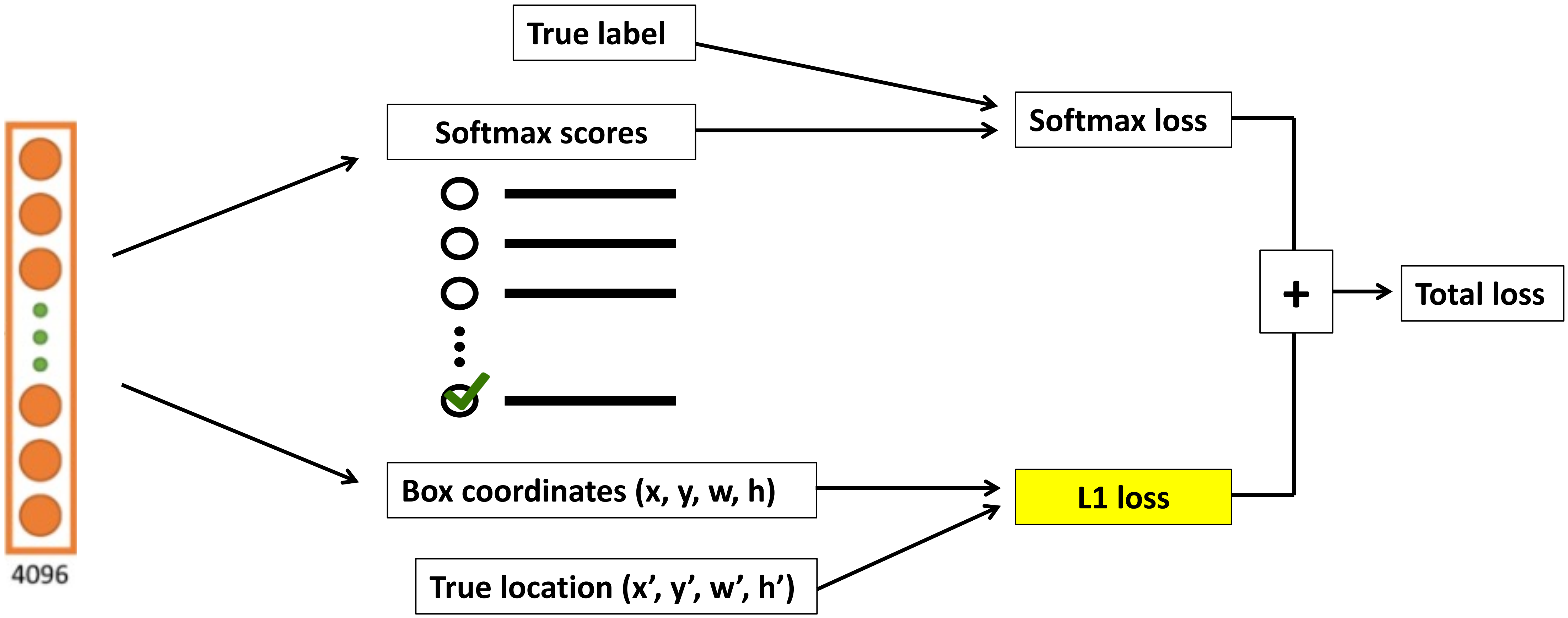
Lesser penalty otherwise

$$-\log \frac{\exp\left(w_k \cdot x + b_k\right)}{\sum_{j=1}^{K} \exp\left(w_j \cdot x + b_j\right)}$$

Figure source: https://ljvmiranda921.github.io/notebook/2017/08/13/softmax-and-the-negative-log-likelihood/

Training in **One Stage**: Multi-task Loss

Objective function sums classification and localization losses for each region proposal



Localization Loss

$$\mathcal{L}_{\text{box}}(t^u, v) = \sum_{i \in \{x,y,w,h\}} L_1^{\text{smooth}}(t_i^u - v_i) \rightarrow L_1^{\text{smooth}}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

True location for true class “u”

Predicted location for class u

Less sensitive to outliers than SSE

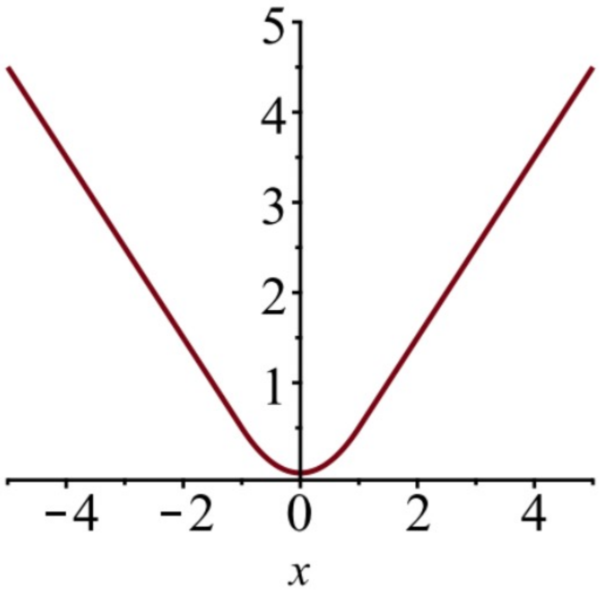


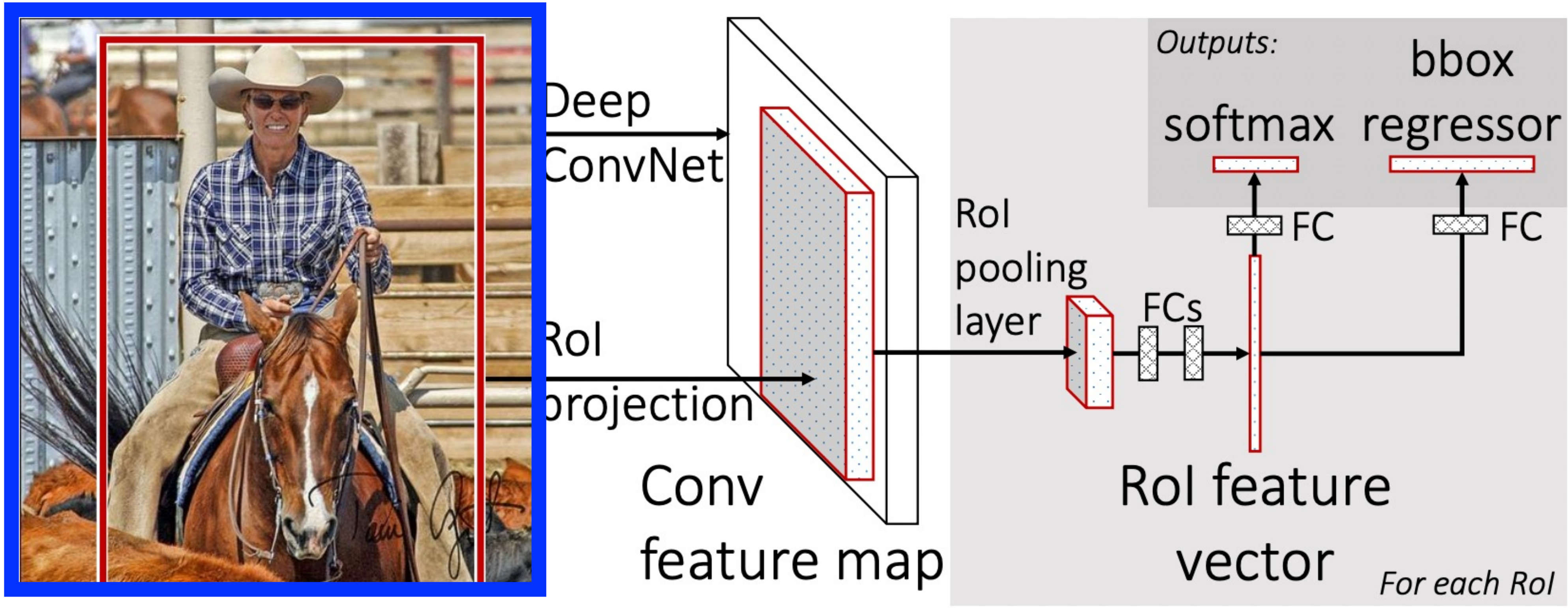
Image Source: https://lilianweng.github.io/lil-log/2017/12/31/object-recognition-for-dummies-part-3.html#bounding-box-regression

Interesting Experimental Analysis

- Which layers to fine-tune?
  - Fine-tune layers after the first layer since including it doesn't boost performance, possibly because it captures task-independent features (e.g., lines, colors)
- Does multi-task training help?
  - Yes; removing either the classification loss or inserting regression loss independently worsens results
- Does more training data help?
  - Yes!
- Does more object proposals help?
  - The evidence suggests it can actually hurt performance

Ross Girshick. Fast R-CNN. CVPR 2015.

Key Limitation: Still Slow and Complex



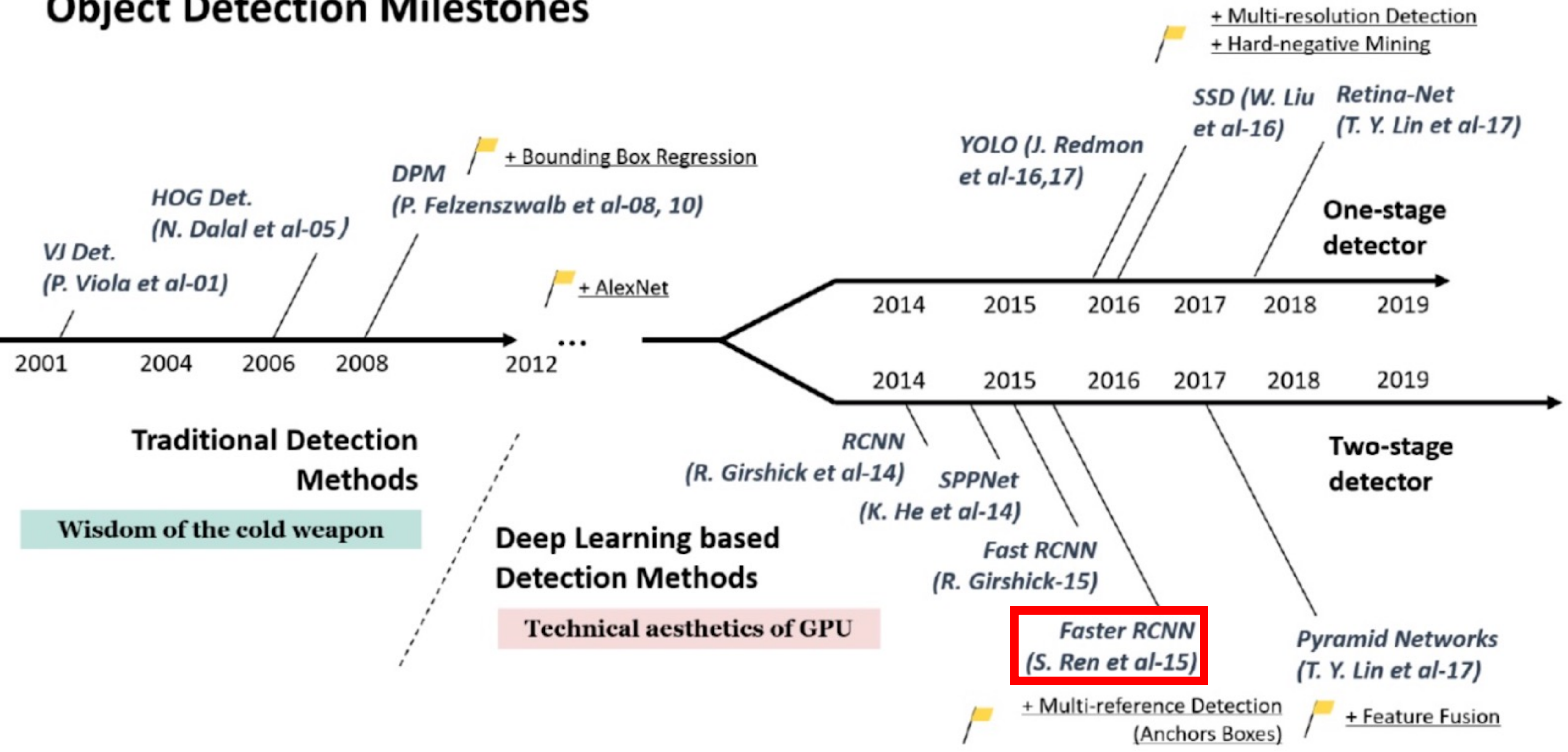
Still, requires generating "object"-like region proposals to provide as input (estimated then to take 2 sec on a CPU) and treated as a separate step.

Ross Girshick. Fast R-CNN. CVPR 2015.

Object Detection: Today's Topics

- Overview of object detection algorithms
- Fast R-CNN
- Faster R-CNN
- YOLO
- Discussion

Object Detection Milestones



Zhou et al. Object Detection in 20 Years: A Survey. arXiv 2019.

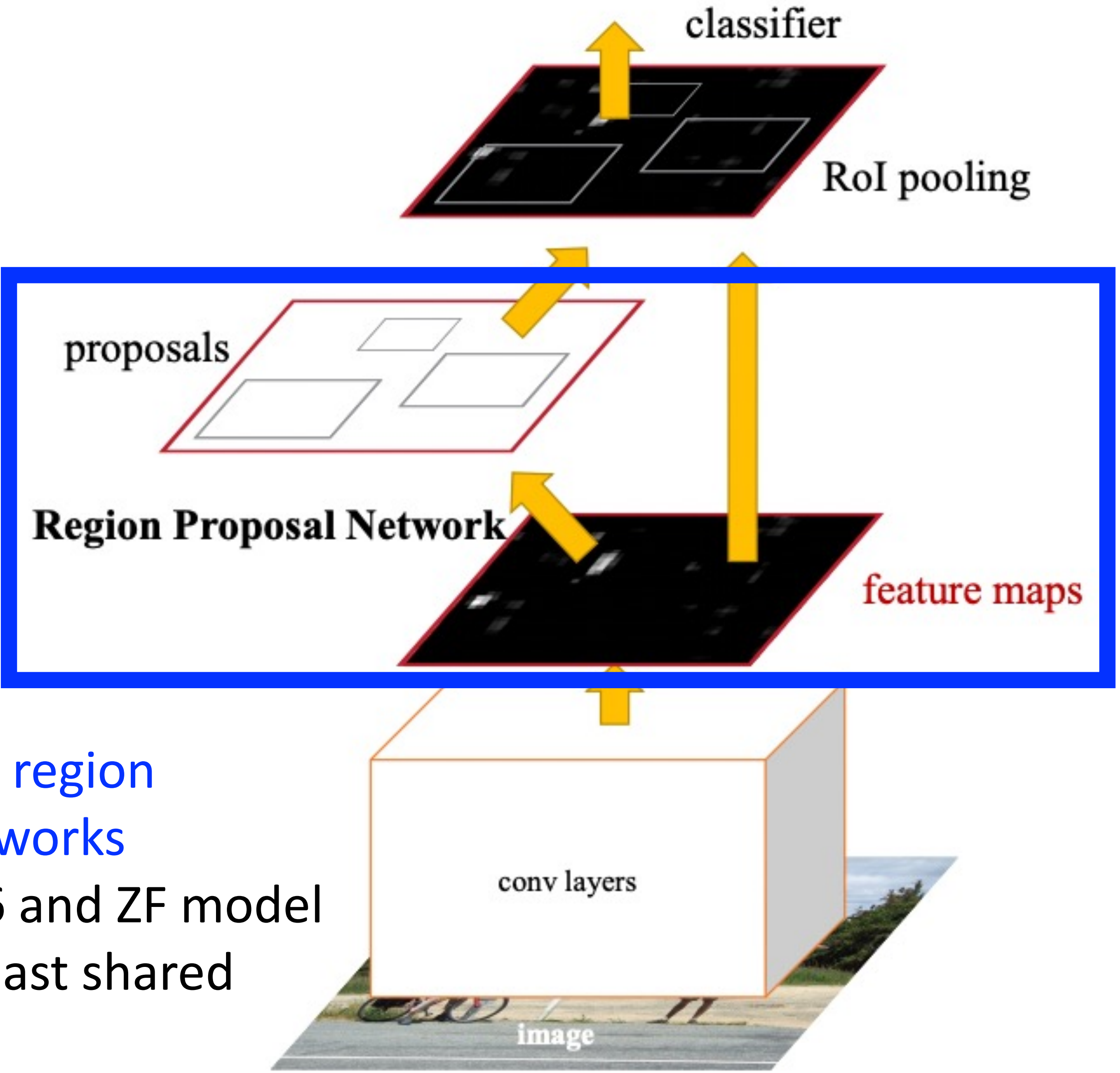
Key Contributions of Faster R-CNN

1. State of art object detection model in terms of accuracy and speed

2. An end-to-end trained model that learns all parts of the pipeline, particularly with the addition of region proposals

Architecture

Key novelty: adds finding region proposals to network so that all parts of model are learned in end-to-end fashion



Convolutional layers shared for region proposal and detection subnetworks

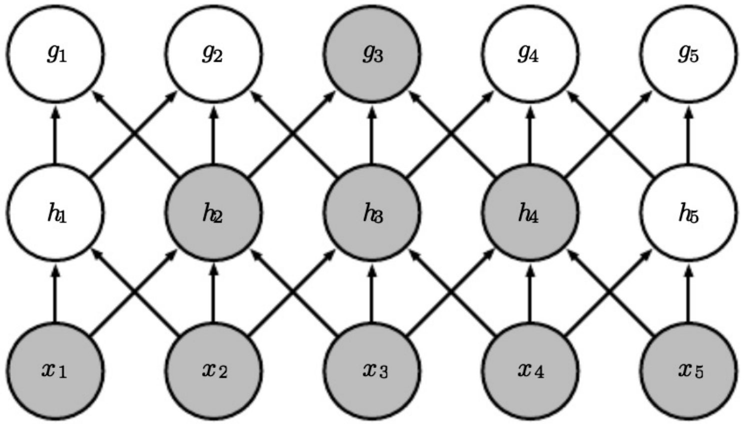
- 2 architectures tested: VGG16 and ZF model
- input to both subnetworks is last shared convolutional layer

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

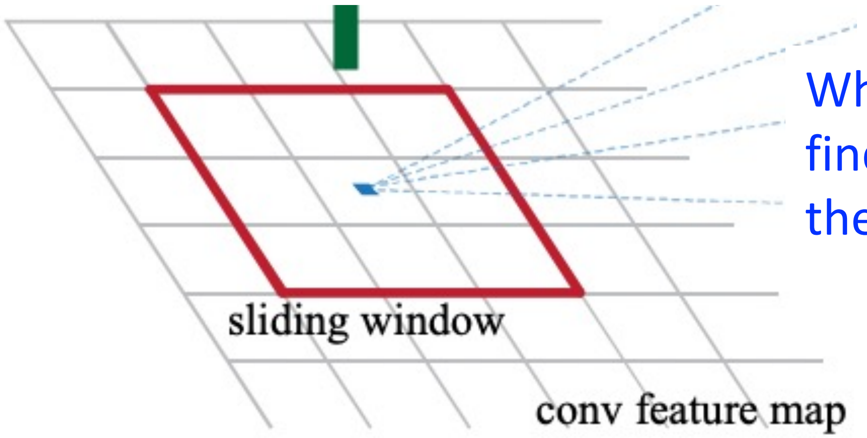
Architecture: Region Proposal Network

Input: convolutional feature map from last layer shared with object detector

Step 1: 3 x 3 convolutional filter applied to identify candidate proposals (recall, filter in the middle of an architecture maps to a larger input space, aka receptive field)



https://www.deeplearningbook.org/contents/convnets.html

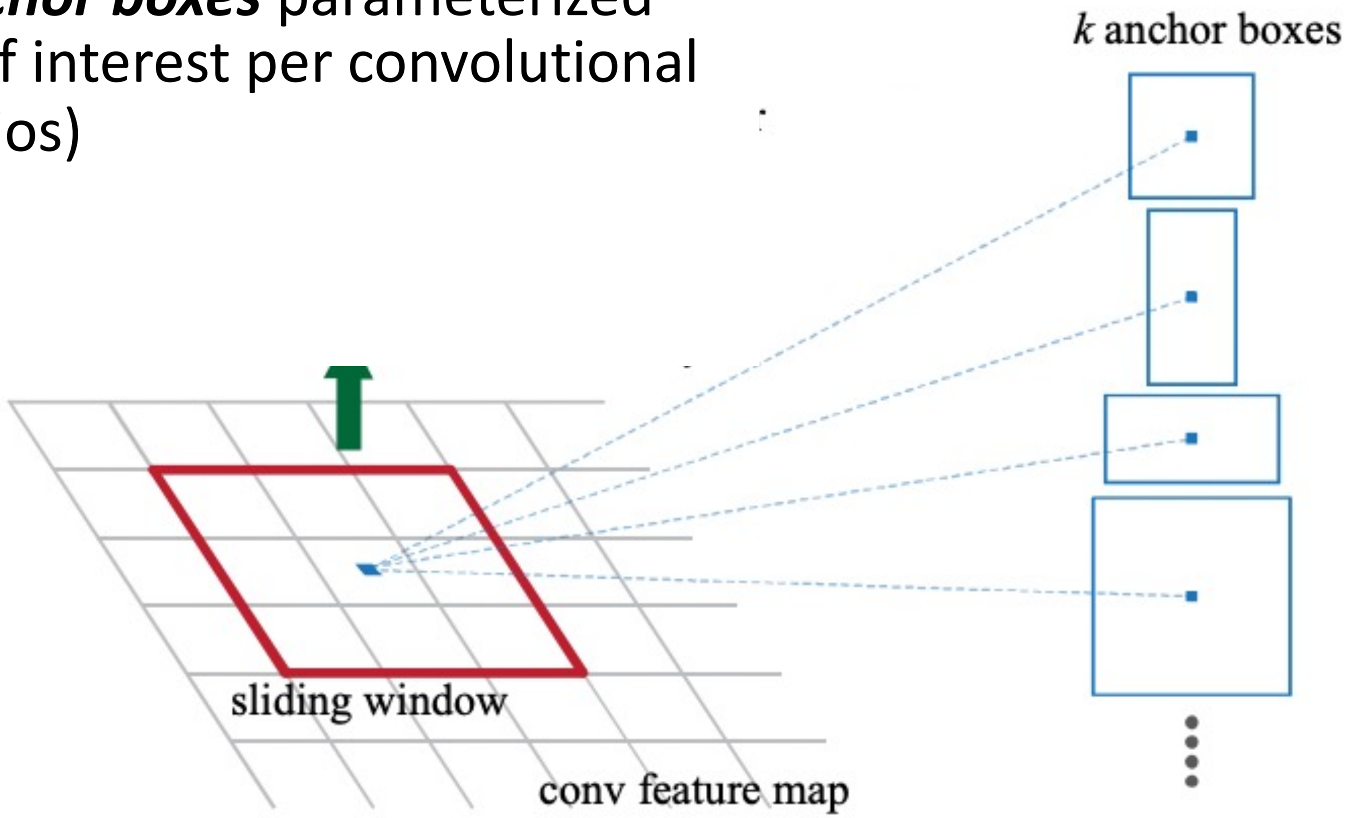


Why can such an approach find objects larger than the receptive field?

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

Architecture: Region Proposal Network

Step 2: candidate regions of multiple scales and aspect ratios are supported efficiently using k = 9 ***anchor boxes*** parameterized relative to the feature map region of interest per convolutional operation (3 scales and 3 aspect ratios)



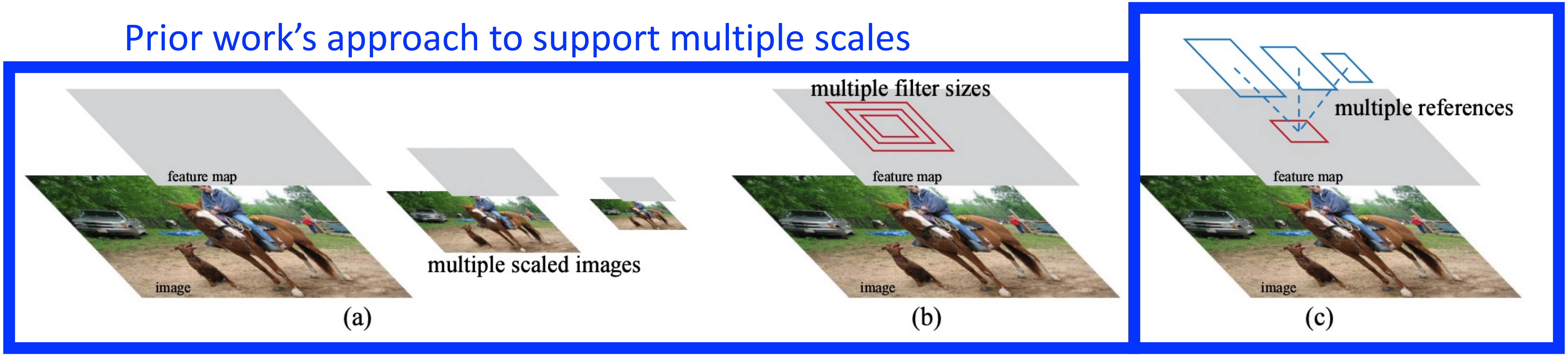
Ren Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." Neurips 2015.

Architecture: Region Proposal Network

Step 2: candidate regions of multiple scales and aspect ratios are supported efficiently using k = 9 ***anchor boxes*** parameterized relative to the feature map region of interest per convolutional operation (3 scales and 3 aspect ratios)

Idea: images, feature maps, and filters all are a single size by using a pyramid of anchors (more efficient!)

Prior work's approach to support multiple scales



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

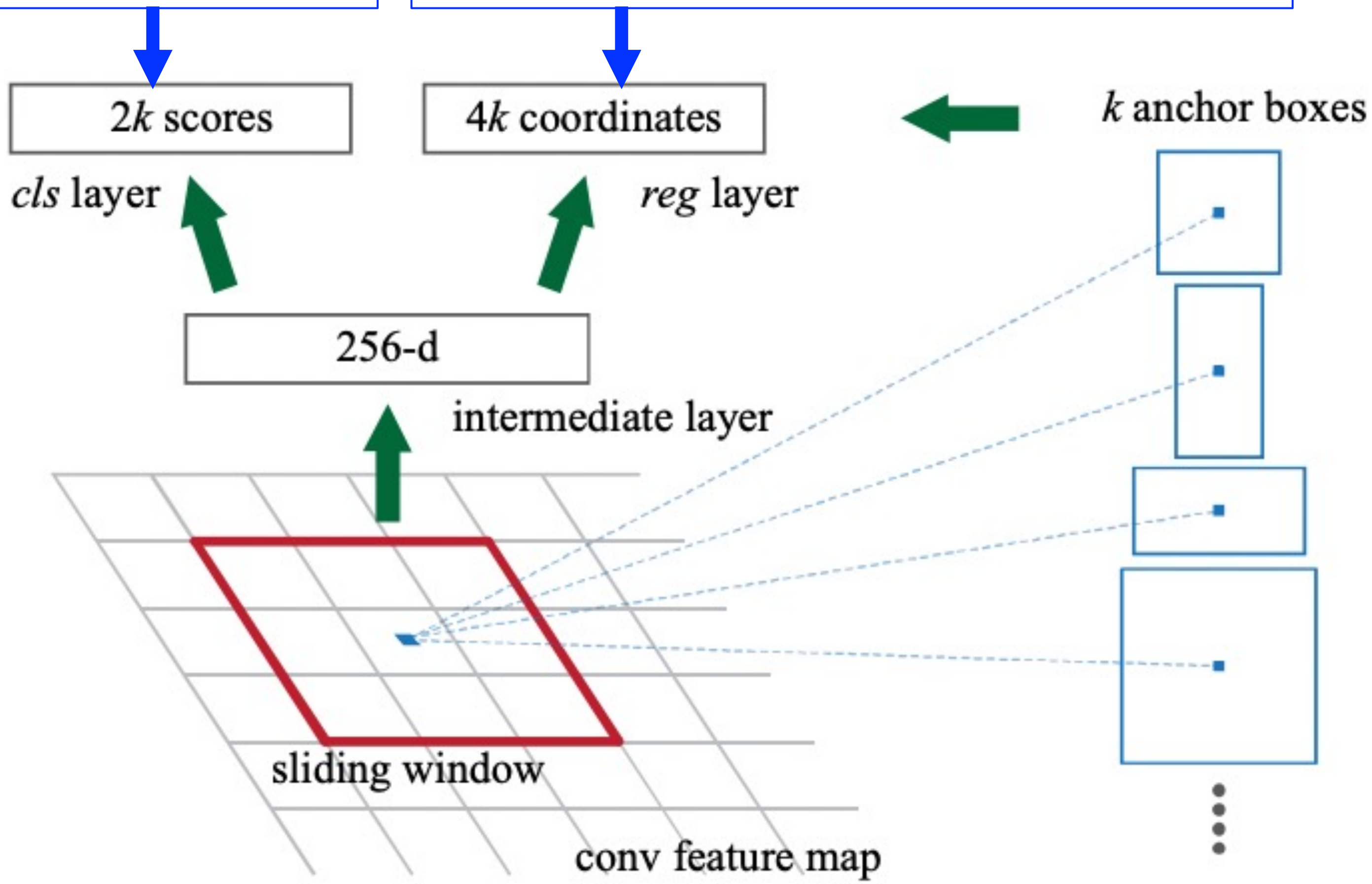
Architecture: Region Proposal Network

(*k* independent regressors learned to support *k* anchor box dimensions)

For each region, predict:

(1) Probability of object/not object

(2) Parameters to regress anchor box to GT box (center, width, and height)



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

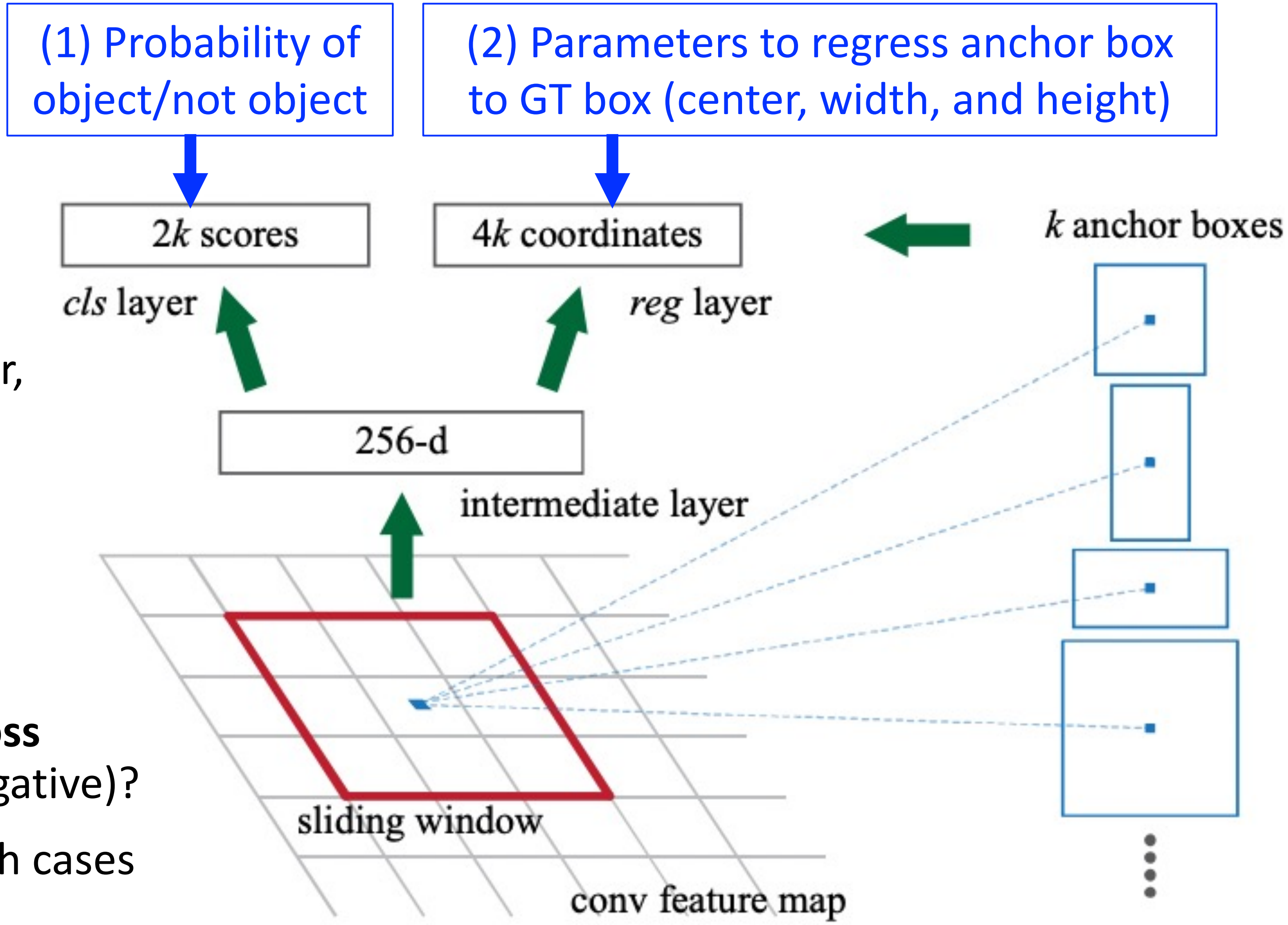
Training: Region Proposal Network

Training

- Loss: for each region proposal, sum classification and localization losses
- GT positive: anchors with IoU > 0.7 with GT (can be multiple anchors) or, when none, highest scoring one
- GT negative: non-positive anchors with IoU < 0.3 with GT
- Any non-assigned anchors ignored

What is relevance of the **regression loss** when no object is present (i.e., GT negative)?

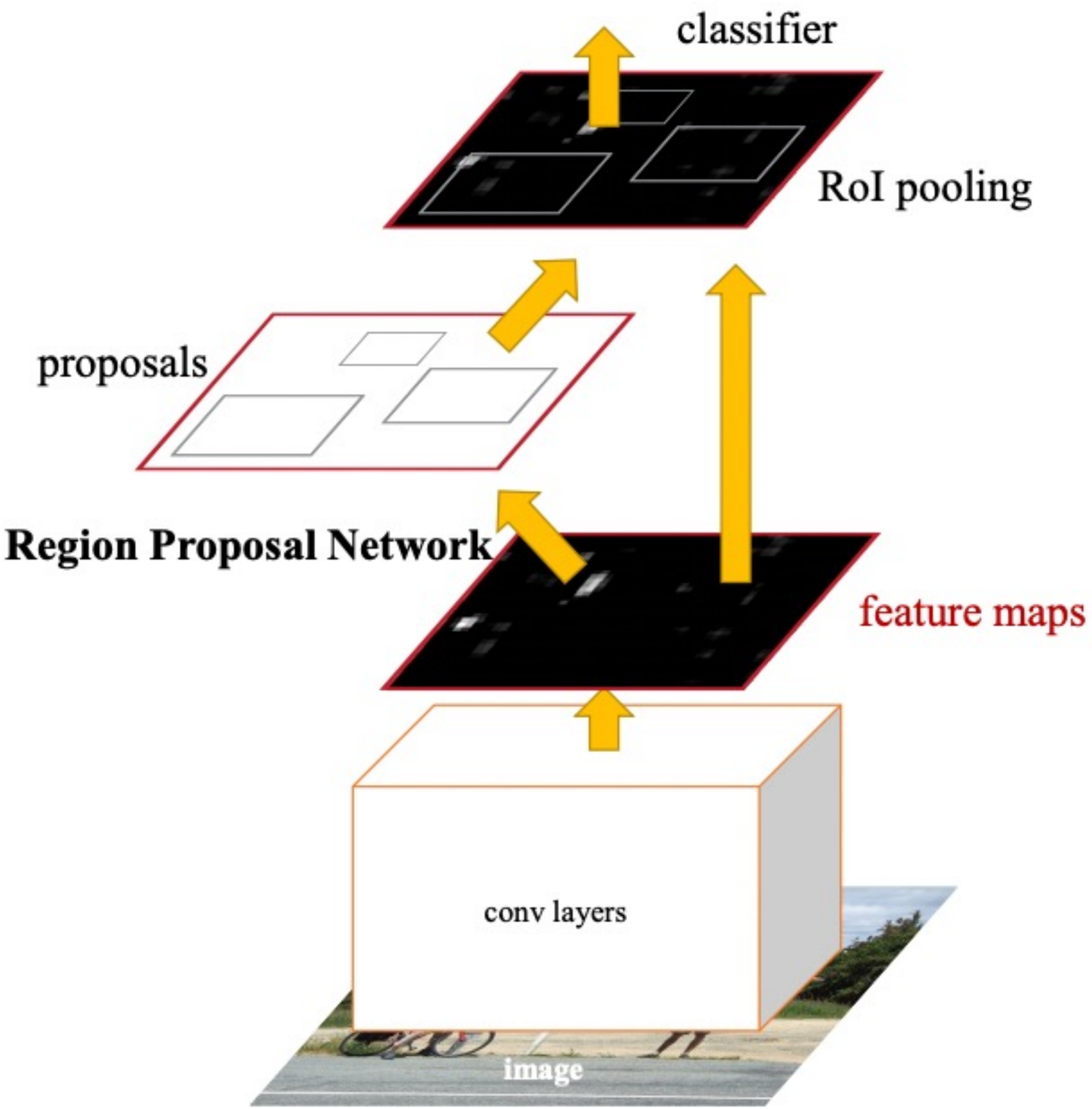
- none; regression loss disabled in such cases



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

Training: Overall

1. Train RPN
2. Train Fast R-CNN using proposals from pretrained RPN
3. Fine-tune layers unique to RPN
4. Fine-tune the fully connected layers of Fast R-CNN



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

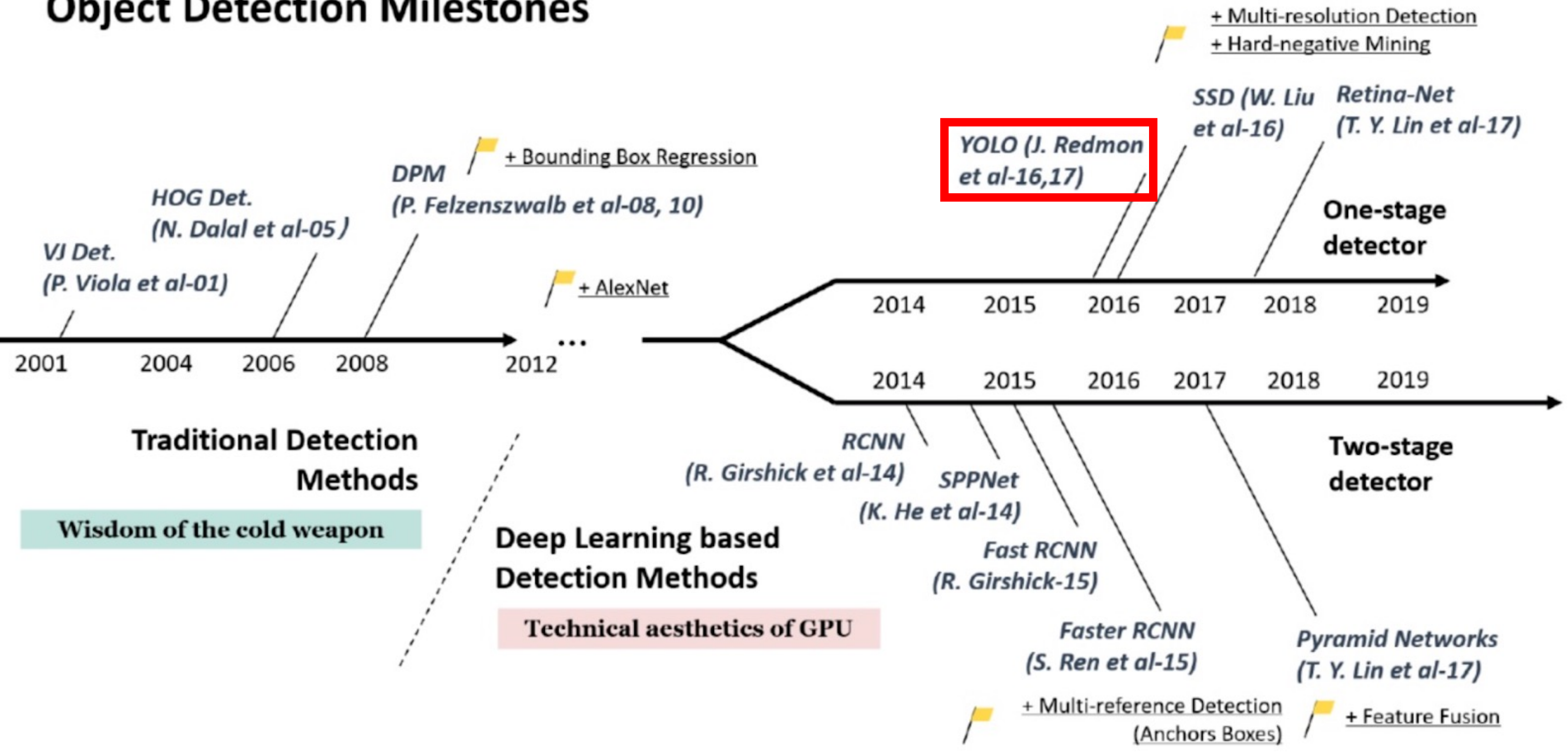
Limitations

- Still relatively slow; i.e., does not support real-time performance

Object Detection: Today's Topics

- Overview of object detection algorithms
- Fast R-CNN
- Faster R-CNN
- YOLO
- Discussion

Object Detection Milestones



Zhou et al. Object Detection in 20 Years: A Survey. arXiv 2019.

Why YOLO?

Named after the proposed technique: **Y**ou **O**nly **L**ook **O**nce

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You Only Look Once: Unified, Real-Time Object Detection." CVPR 2016.

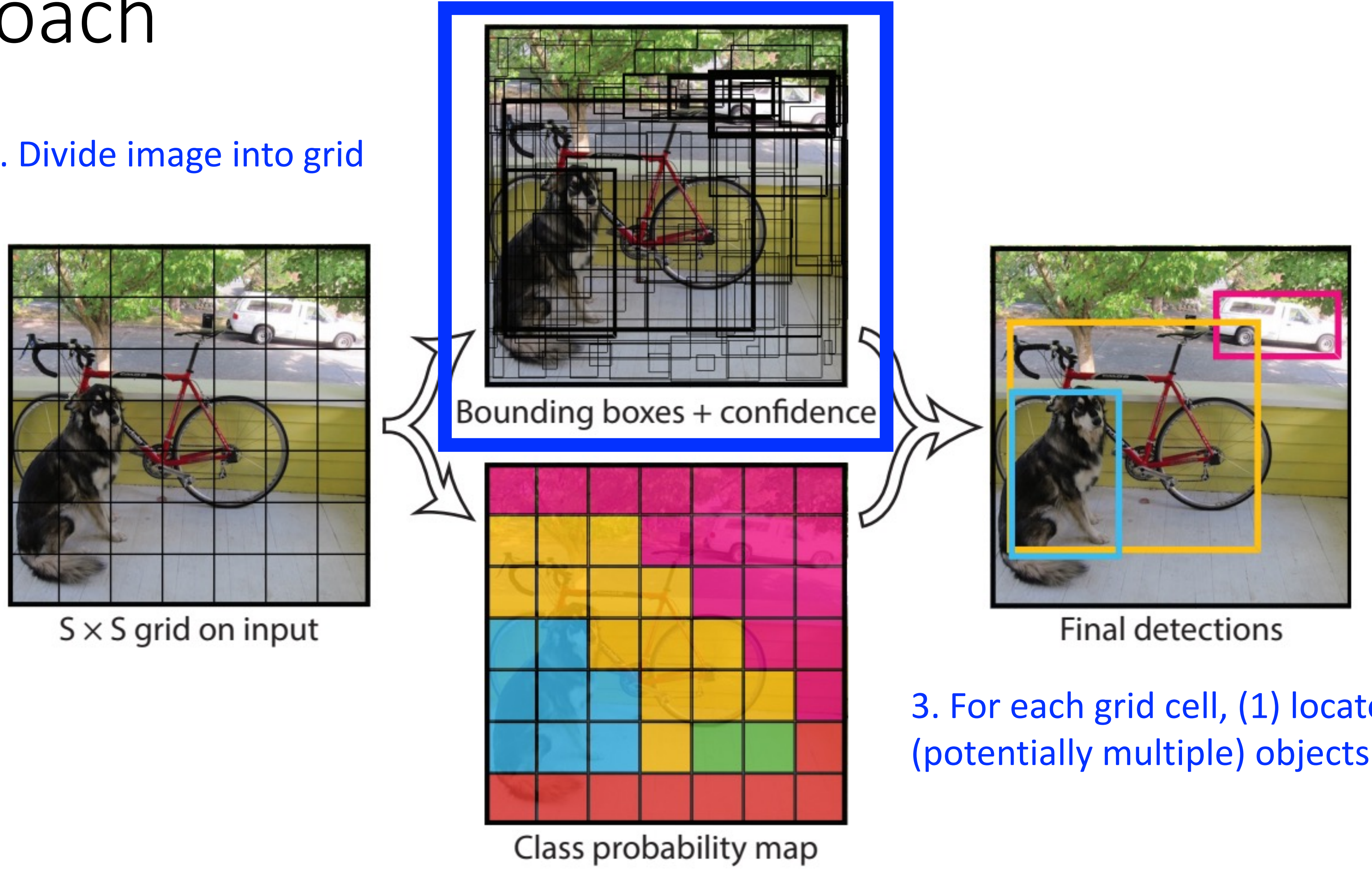
Key Contributions

- A CNN architecture that detects objects by looking at the entire image **once**, treating the problem as a regression problem
  - i.e., a paradigm shift away from using classifiers to label image regions
- Most accurate **real-time** object detection system (i.e., 30+ fps)
- Generalizes better than existing approaches to out-of-domain data

Approach

2. For each grid cell, (1) locate (potentially multiple) objects and (2) predict a probability distribution for class labels (assuming an object is present)

1. Divide image into grid



3. For each grid cell, (1) locate (potentially multiple) objects

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

Approach: BB Prediction Per Grid Cell

Center of BB relative to **grid cell**

Width and height relative to **image**

$$y = (\boxed{p_c}, \boxed{b_x, b_y}, \boxed{b_h, b_w}$$

Confidence score: reflects whether an object is present and the predicted box's accuracy

1. What should p_c equal if no object is present?
- 0

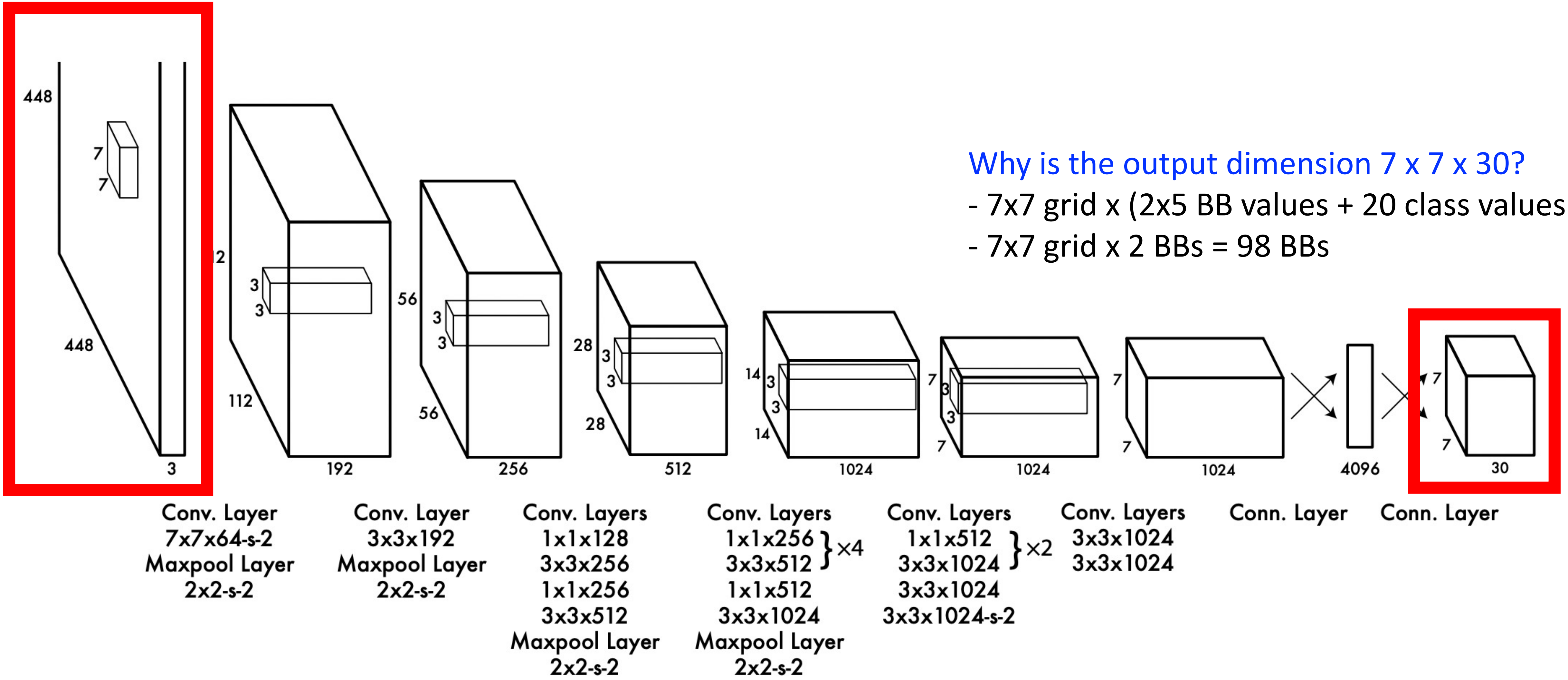
2. What should p_c equal if an object is present?
- IoU between predicted and ground truth boxes

3. Although multiple BBs are predicted per grid cell, only the BB with the highest IoU to the GT is used. So why have multiple BBs per grid cell?
- Encourage each BB predictor to specialize to different BB properties (e.g., sizes, aspect ratios, object category types)

Architecture

Input: RGB image resized to fixed input size

Output: 98 BB per image w/ class probabilities



Why is the output dimension 7 x 7 x 30?

- 7x7 grid x (2x5 BB values + 20 class values)
- 7x7 grid x 2 BBs = 98 BBs

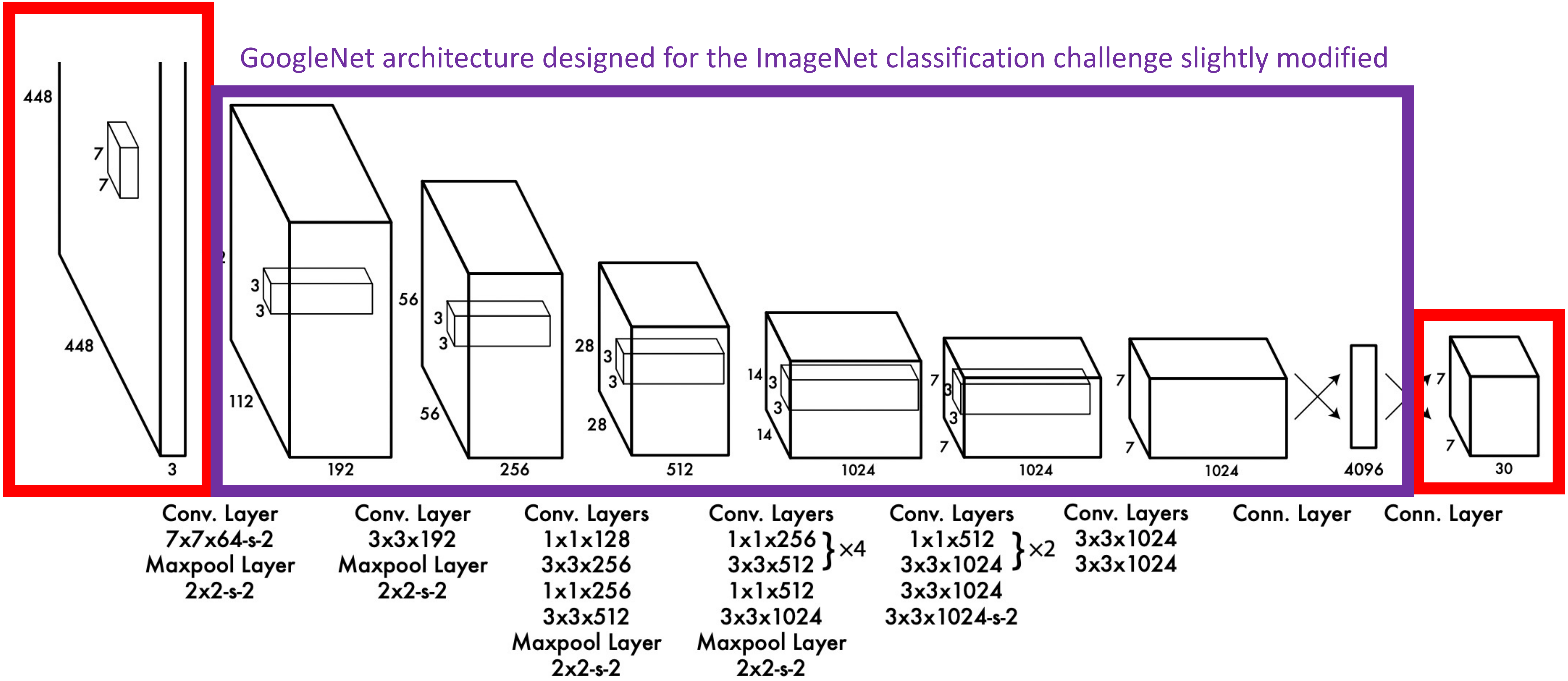
Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection.” CVPR 2016.

Architecture

Input: RGB image resized to fixed input size

Output: 98 BB per image w/ class probabilities
(i.e., 7x7 grid x 2 BB per grid cell = 98 BB)

GoogleNet architecture designed for the ImageNet classification challenge slightly modified



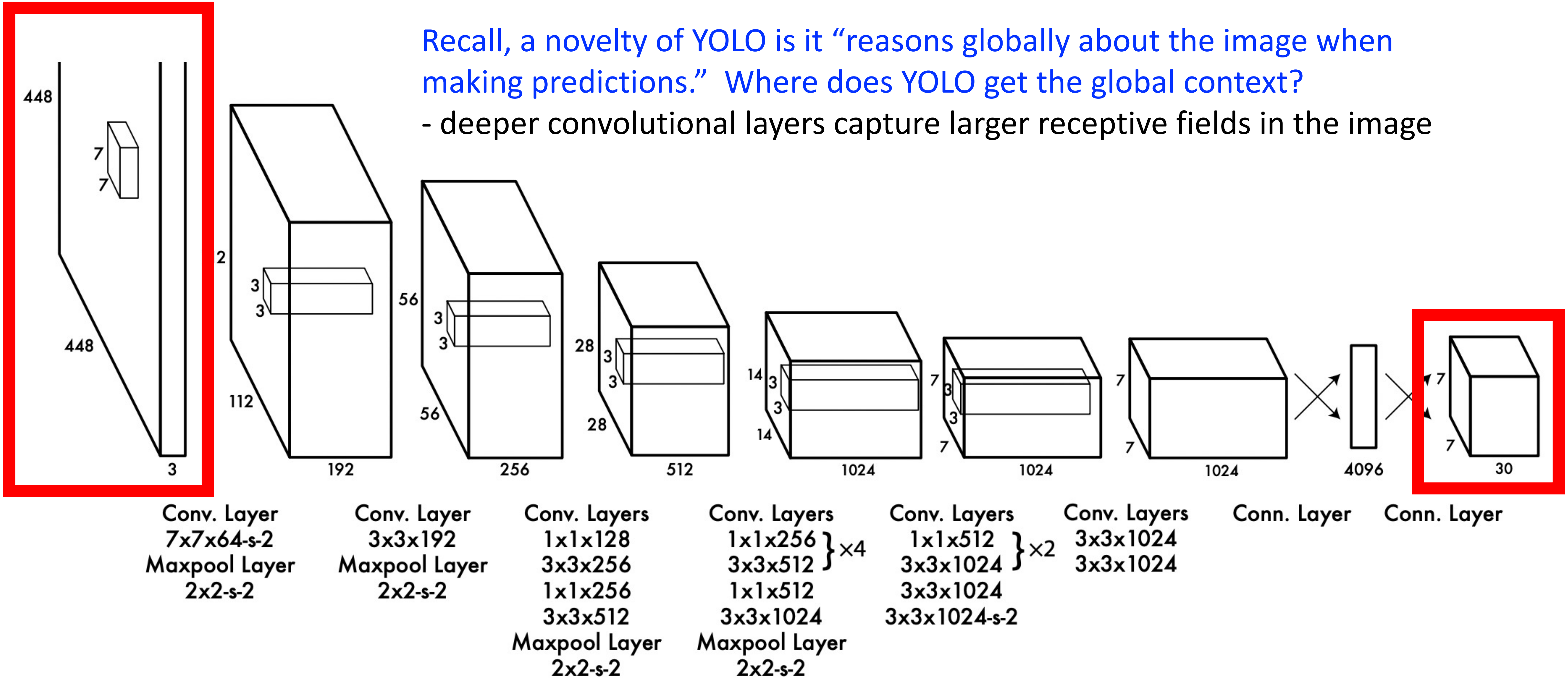
Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection.” CVPR 2016.

Architecture

Input: RGB image resized to fixed input size

Output: 98 BB per image w/ class probabilities
(i.e., 7x7 grid x 2 BB per grid cell = 98 BB)

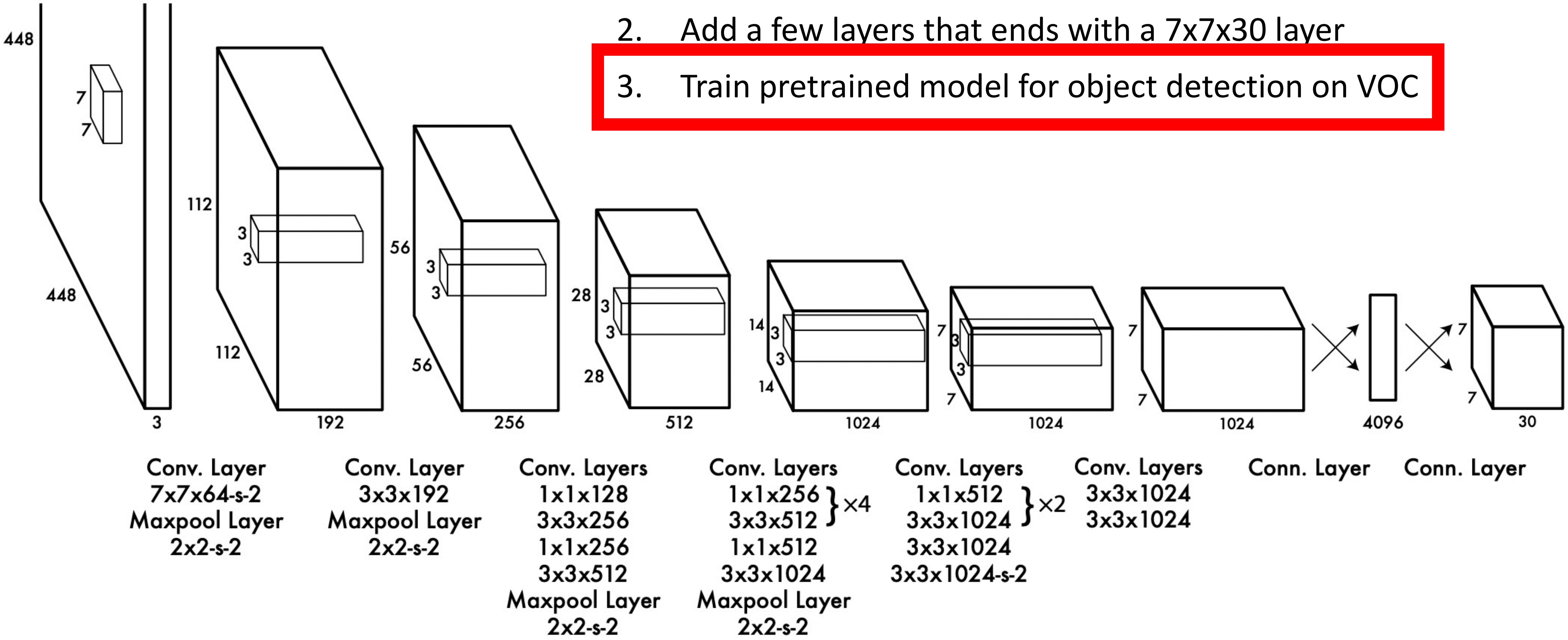
Recall, a novelty of YOLO is it "reasons globally about the image when making predictions." Where does YOLO get the global context?
- deeper convolutional layers capture larger receptive fields in the image



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

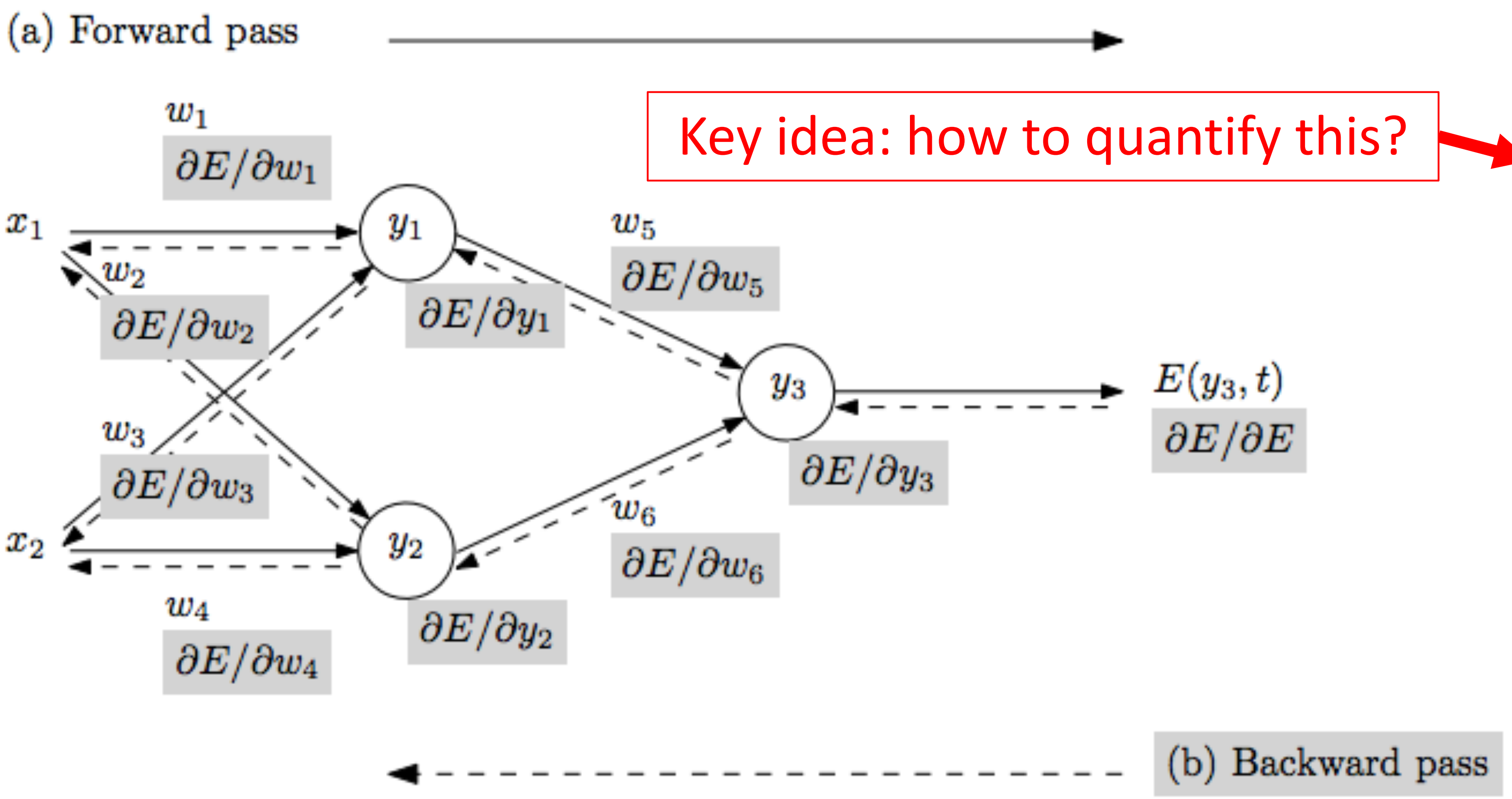
Training

1. Pretrain first 20 convolutional layers for ImageNet classification
2. Add a few layers that ends with a 7x7x30 layer
3. Train pretrained model for object detection on VOC



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

Training



Key idea: how to quantify this?

- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make predictions
  2. **Error quantification**: measure dissatisfaction with a model's predictions on training data
  3. **Backward pass**: using predicted output, calculate gradients backward to assign blame to each model parameter; account for weight sharing by using average of all connections for a parameter
  4. Update each parameter using calculated gradients

https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

Training: Multi-Part Loss Function

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

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

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

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

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

Number of grid cells; what does YOLO use?

Number of bounding box predictors per cell; what does YOLO use?

Penalty only for most confident BB predictor from j predictors when an object is present

Penalizes incorrect classifications (for each grid cell)

Penalty for classification occurs ONLY if an object is actually present in the grid cell

Penalizes imperfect positioning of object centers (for each BB)

Penalizes imperfect widths and heights (for each BB)

Pushes the BB confidence score to match the IoU between the prediction and GT

Punishes the network for predicting an object is present when no object is in the grid cell (i.e., pushes confidence to 0)

Training: Loss Function

$$\boxed{\lambda_{\textbf{coord}}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$
$$+ \boxed{\lambda_{\textbf{coord}}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\boxed{\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)}^2 \right]$$
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2$$
$$+ \boxed{\lambda_{\text{noobj}}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2$$
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

Set coefficient of 5 to prioritize loss from BB shape predictions over misclassification predictions

Square root used to weight a small deviation in large boxes less than for small boxes

Set coefficient of 0.5 to reduce loss from boxes that don't contain objects to mitigate learning to push the confidence to 0 since most cells don't contain objects

Object Detection: Today's Topics

- Overview of object detection algorithms
- Fast R-CNN
- Faster R-CNN
- YOLO
- Discussion


The End