# Deep Learning Detection and Segmentation

**Jenq-Neng Hwang, Professor**
Department of Electrical & Computer Engineering
University of Washington, Seattle WA
**hwang@uw.edu**

# Visual Deep Learning Tasks

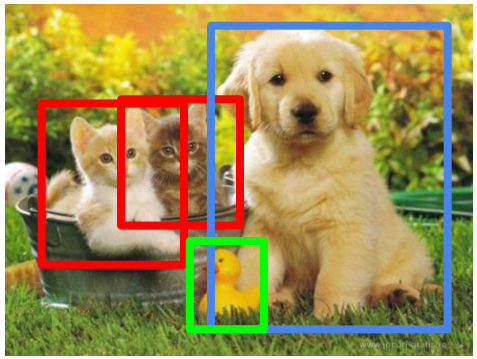
**Classification**

CAT

**Classification + Localization**

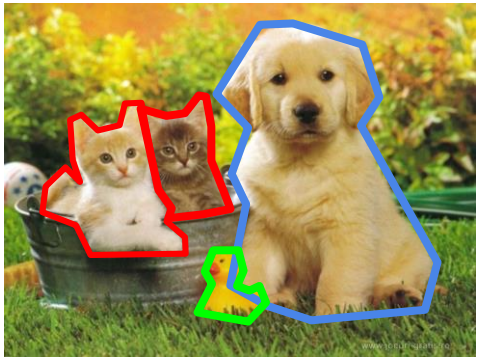
CAT

**Object Detection**



CAT, DOG, DUCK

**Instance Segmentation**



CAT, DOG, DUCK

Single object

Multiple objects

Fei-Fei Li & Andrej Karpathy & Justin Johnson

# Classification + Localization Task

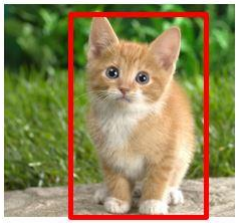
**Classification**: C classes

**Input:** Image
**Output:** Class label
**Evaluation metric:** Accuracy
**Loss Function:** cross entropy

CAT

**Localization**:

**Input:** Image
**Output**: Box in the image (x, y, w, h)
**Evaluation metric:** Intersection over Union
**Loss Function:** L2

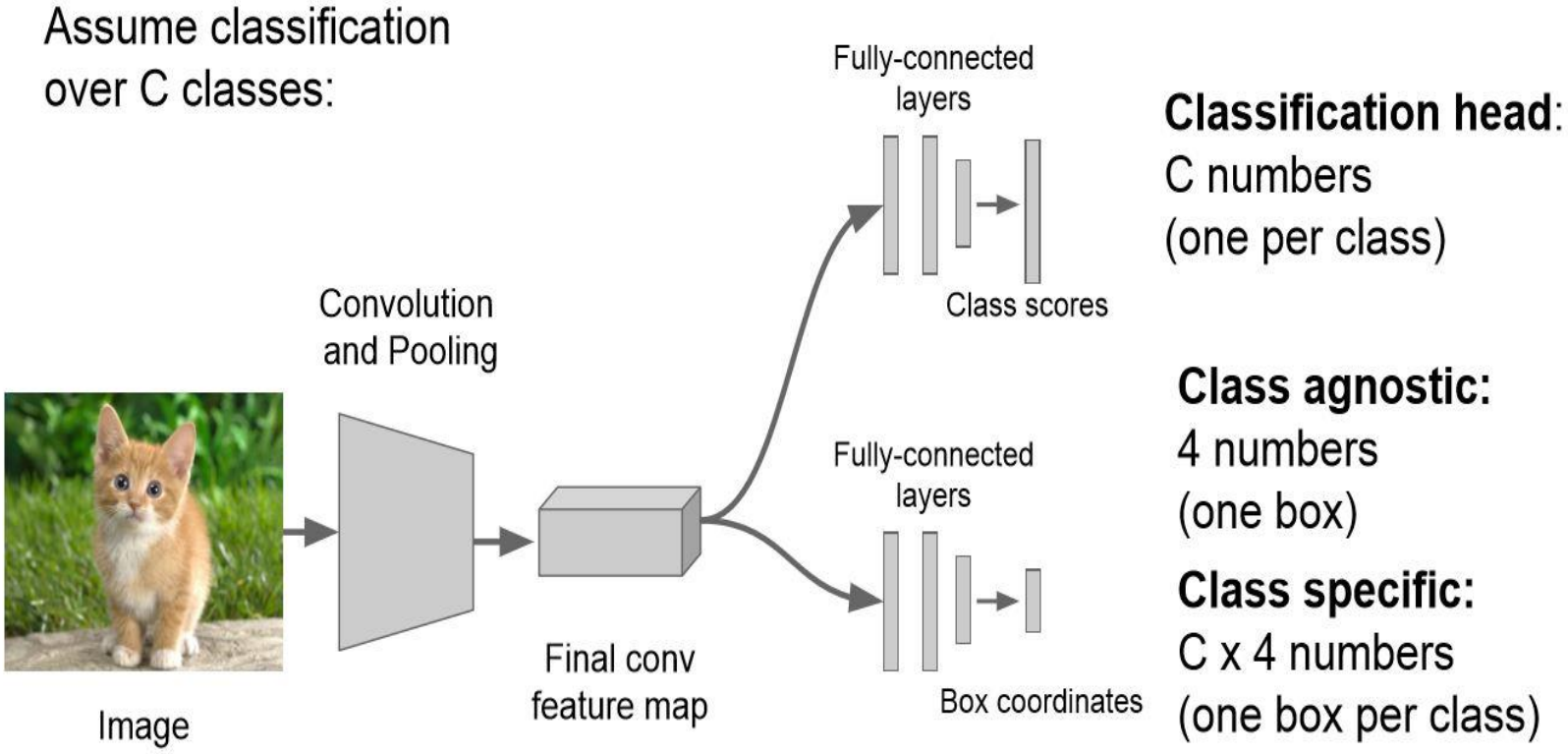


(x, y, w, h)

**Classification + Localization: Do both**

# Training for Classification + Localization

- Attach new fully-connected "regression head" to the network



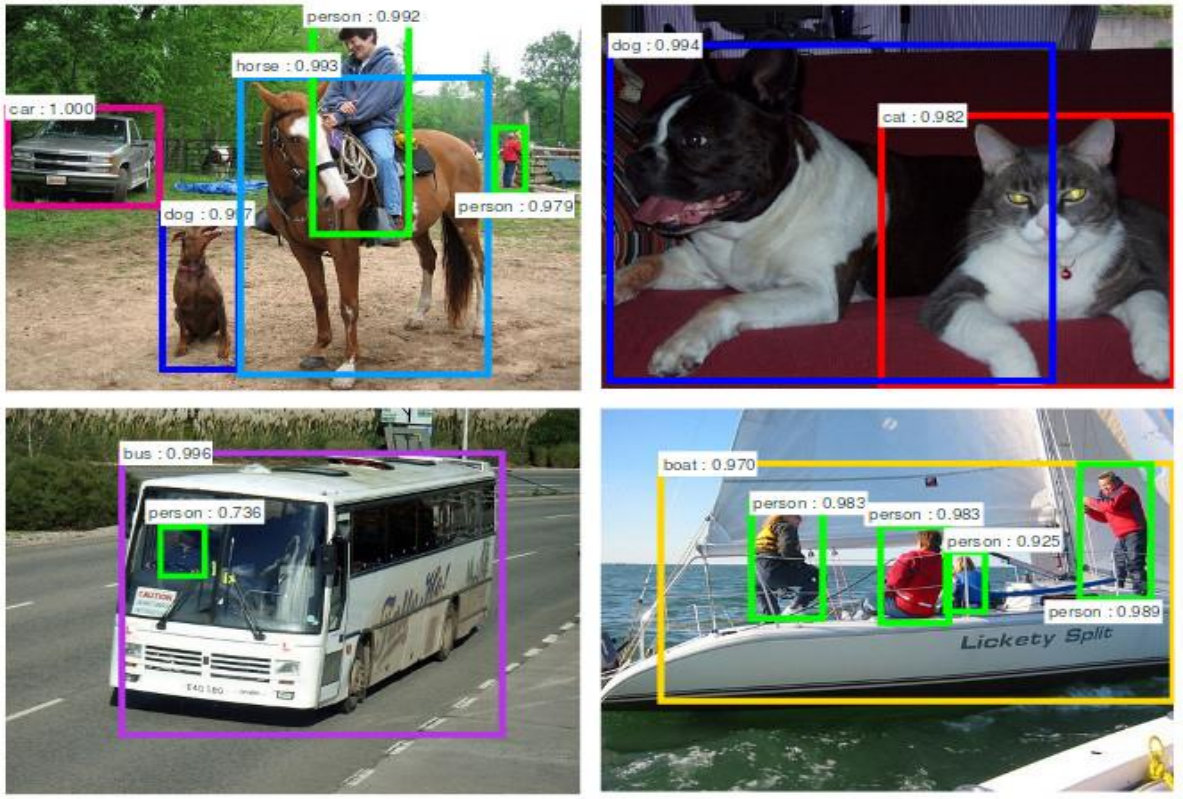
# Detection as Regression?

CAT, (x, y, w, h)
CAT, (x, y, w, h)
….
CAT (x, y, w, h)

How many?

Need variable sized outputs
(in classification, one class one output)

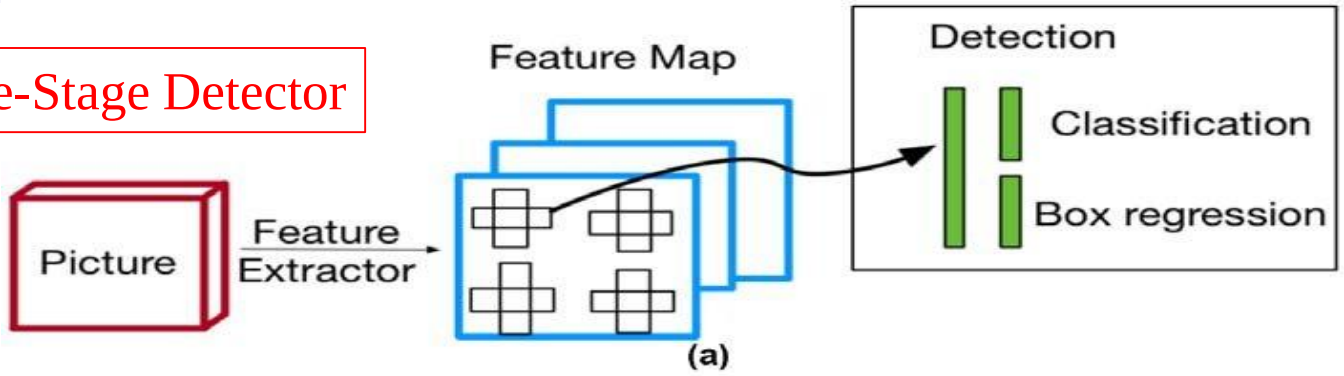
# Object Detection



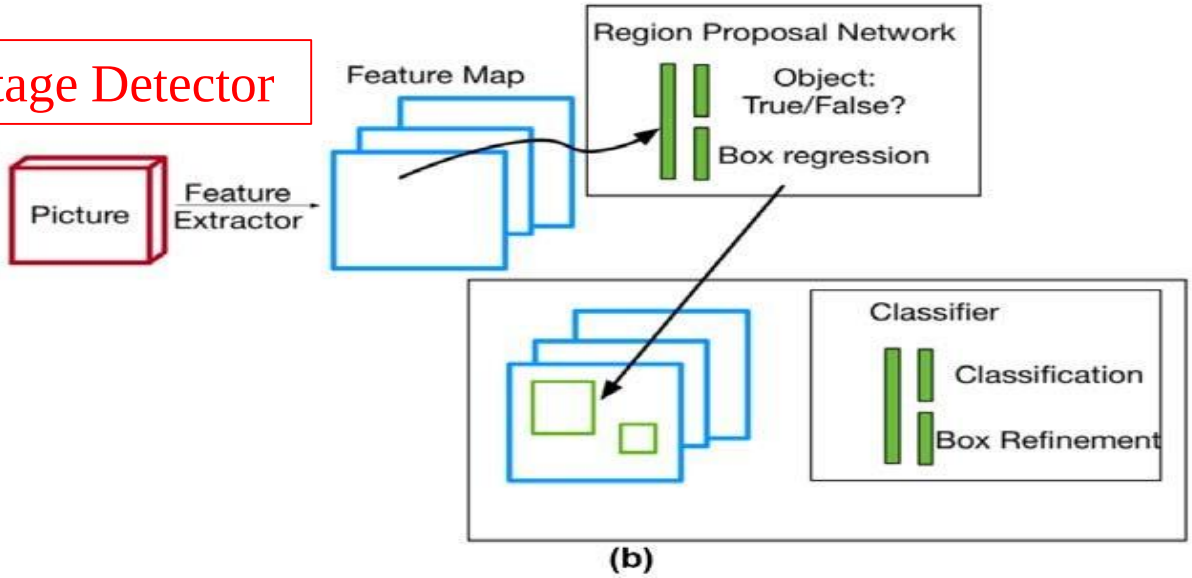

Object detection is to localize each object as a bounding box with a category label.

# CNN based Object Detectors Evolution

One-Stage Detector

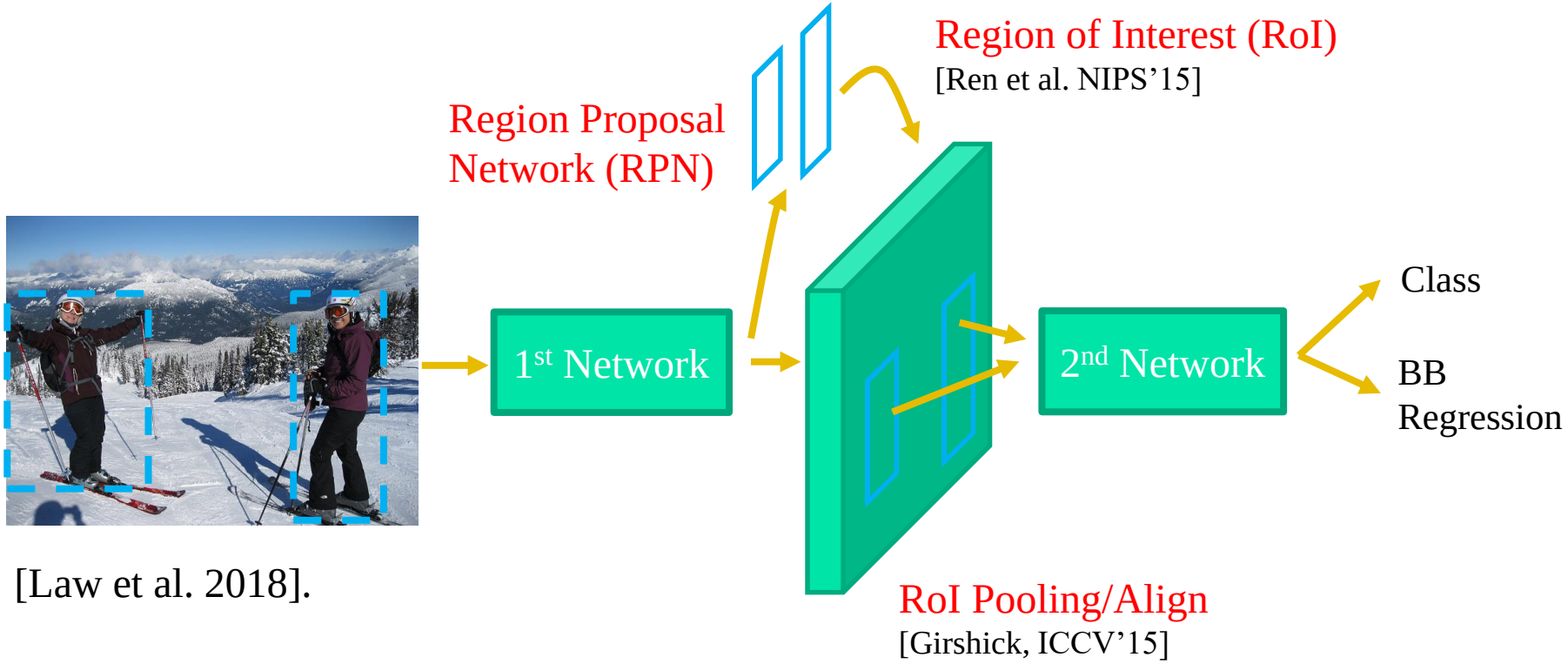



Two-Stage Detector



# Two-Stage Detectors

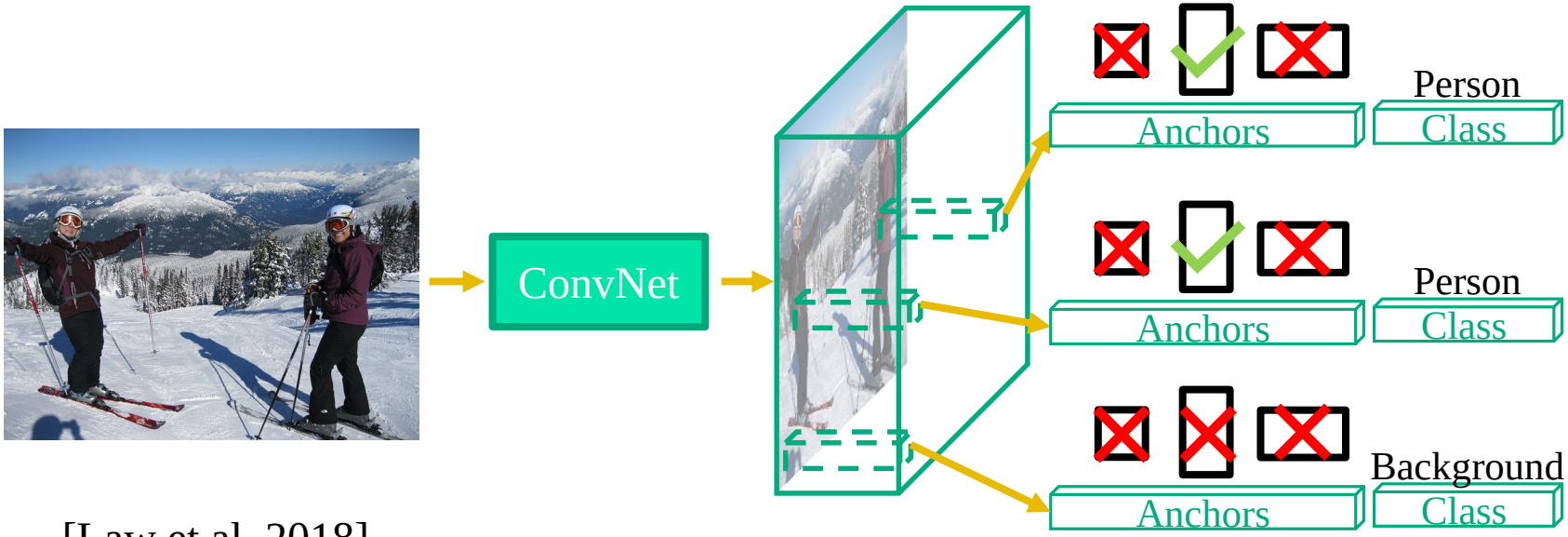
**Faster R-CNN** [Girshick et al. CVPR'14], **Image Pyramid** [He et al. ECCV'14], **Mask R-CNN** [He et al. ICCV'17], **Cascade R-CNN** [Cai & Vasconcelos, CVPR'18], **SNIP** [Singh & Davis, CVPR'18]




[Law et al. 2018].

# One-Stage Detectors

**SSD** [Liu et al. ECCV'16], **YOLO** [Redmon & Farhadi, CVPR'17], **DSOD** [Shen et al. ICCV'17], **DSSD** [Fu et al. arXiv'17], **RetinaNet** [Lin et al. ICCV'17], **CenterNet** [Zhou et al. arXiv'19]



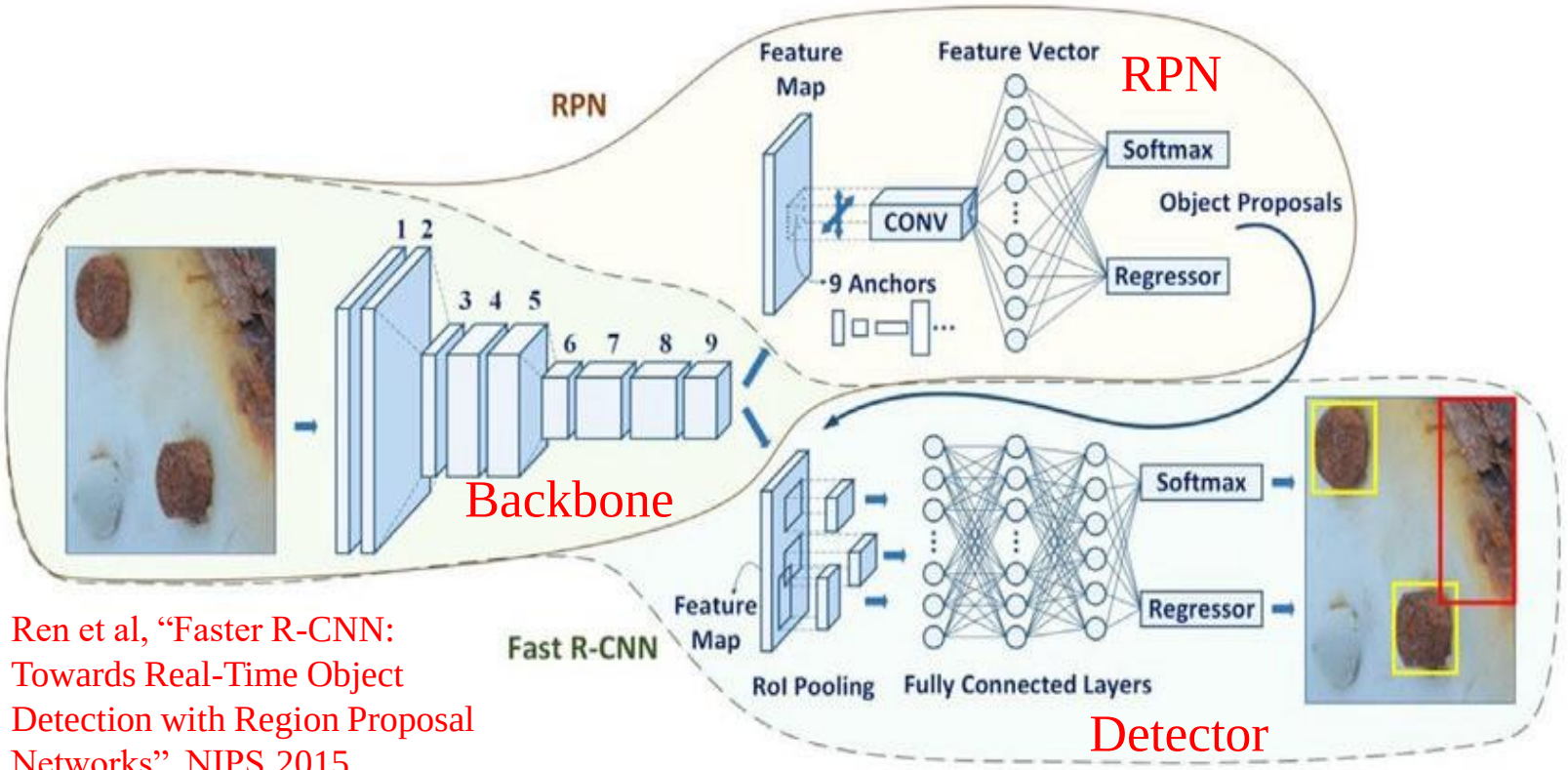

[Law et al. 2018].

# One-Stage & Two-Stage

- In a two-stage object detector, the first stage takes in an input image and outputs region proposals, i.e., locations where an object might be present
- The second stage involves classifying to which class each proposal belongs to and the bounding box location, i.e., classification and regression heads.
- One-stage detectors use the concept of anchor boxes (or default boxes), which are logical boxes assigned to every cell in a feature map, relative to which the bounding box regression and classification takes place.
- An entire grid of feature map is considered as region proposals which in turn is classified by the same neural network to produce class scores and bounding box offsets.

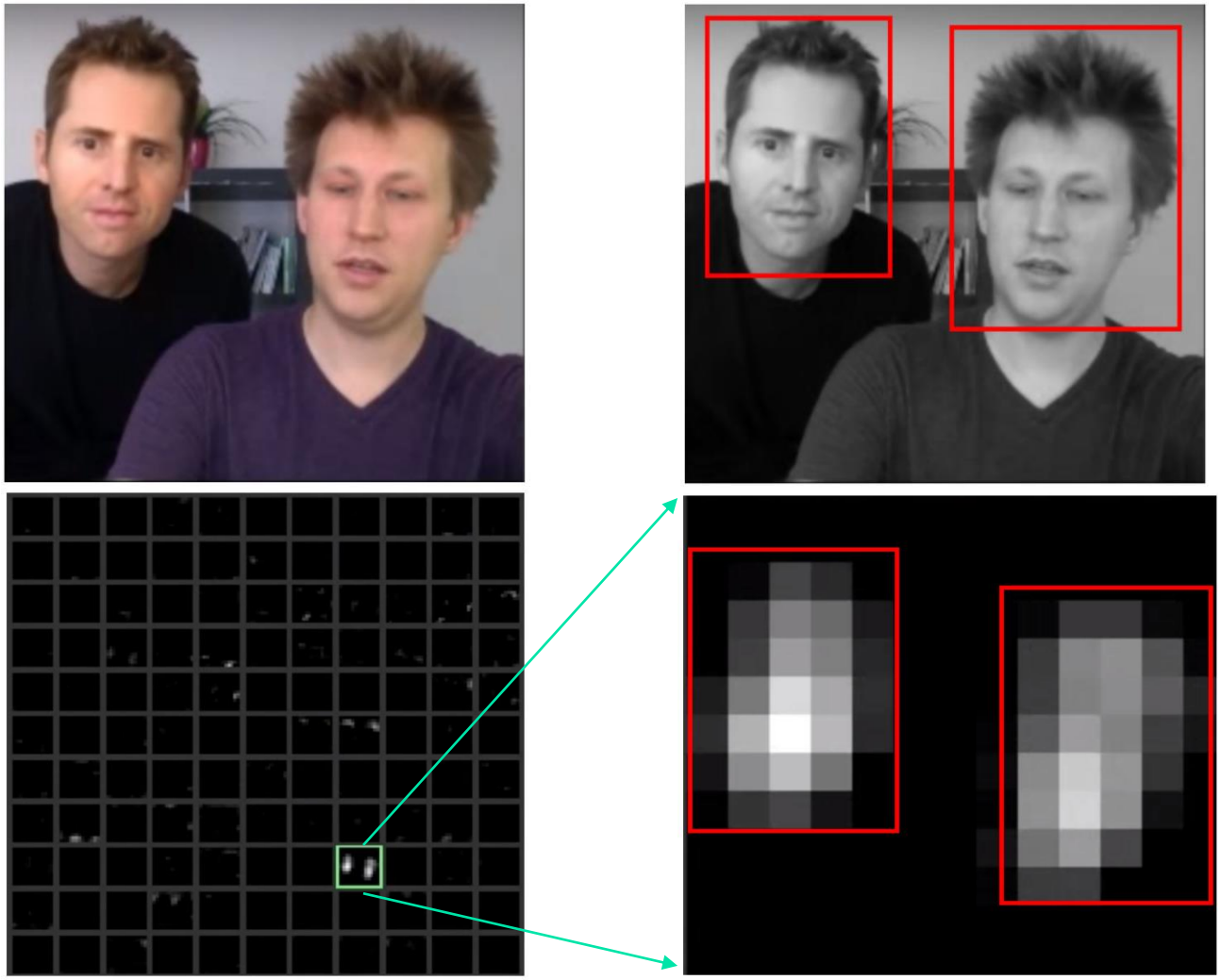
# Faster R-CNN: A 2-Stage Object Detector

- A **Region Proposal Network (RPN)** trained to produce region proposals
- After RPN, use RoI Pooling and an upstream classifier and bbox regressor



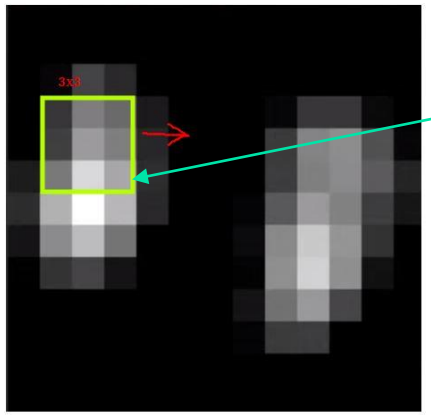

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

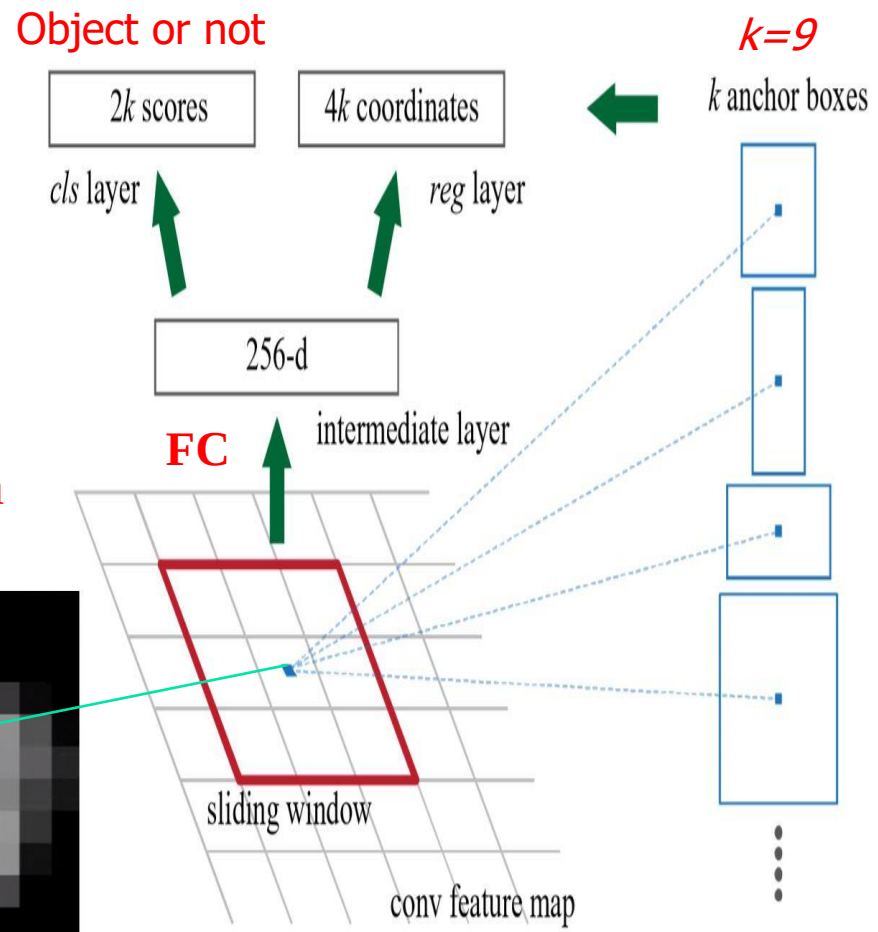
# Feature Map Response in a Faster R-CNN



# Region Proposal Net (RPN)

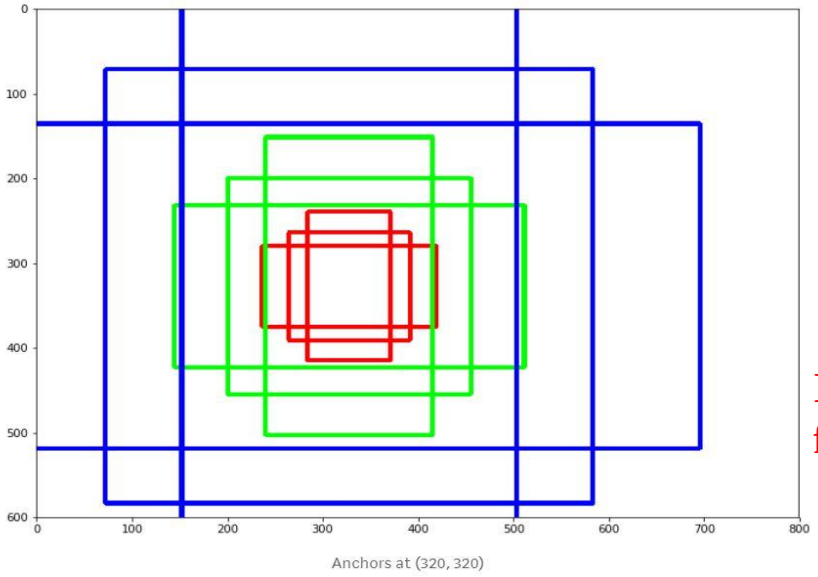
- Slide a small window on the feature map
- Build a small network for:
  - classifying object or not-object, and
  - regressing bbox locations
- Position of the sliding window provides localization information of anchor boxes with reference to the image
- Box regression provides finer localization information with reference to this sliding window



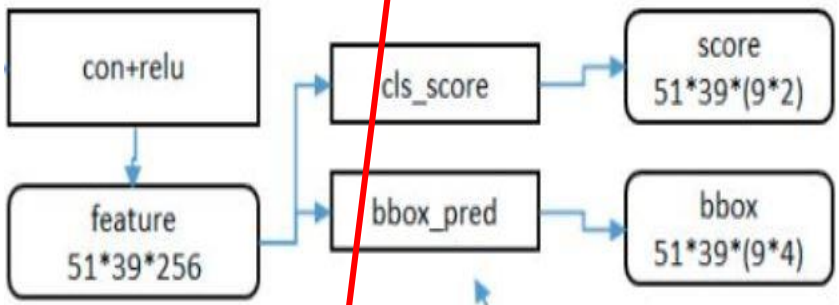


# 9 Anchor Boxes per Location

- Three scales or sizes for anchors: 128x128, 256x256, 512x512 (defined on image, 800x600). 3 aspect ratios per position
- One position at every stride of 16, there will be 1989 (51x39) positions. This leads to 17901 (1989 x 9) boxes to consider.




1:1, 1:2, 2:1



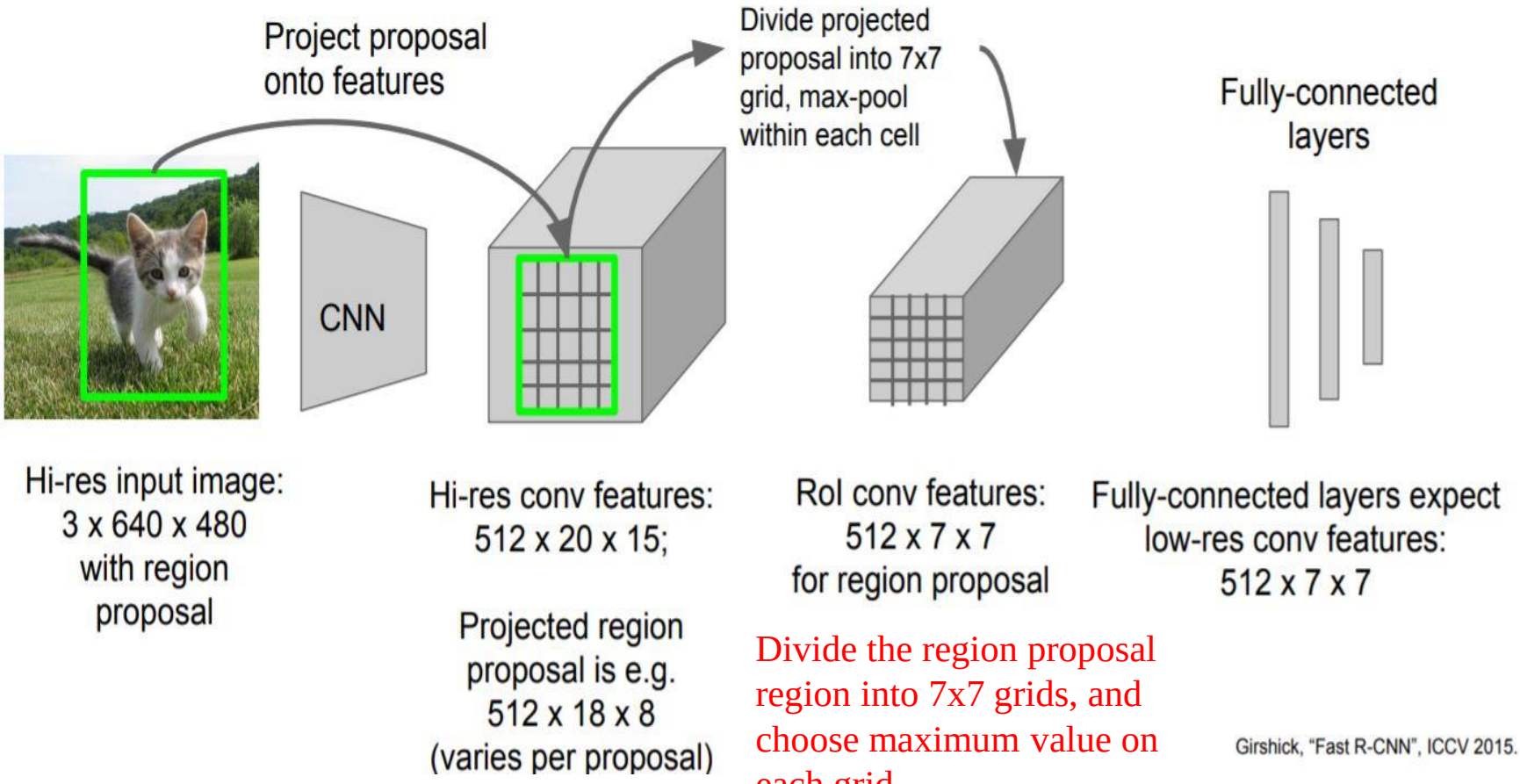

**Inputs to RPN are 9 anchors from 256 feature maps**

Consider 9 'anchors' on each of the 51*39 positions

# Non-Maximum Suppression

- The ~20K anchors from each image generate ~20K object proposal bounding boxes, which are arranged according to their *cls* confidence scores. Then, a non-maximum suppression (NMS) is applied with an IoU threshold of 0.7.
- From the top down, all of the bounding boxes which have an IoU of greater than 0.7 with another bounding box are discarded. Thus the highest-*cls*-scoring bounding box is retained for a group of overlapping boxes.
- This gives about 2000 proposals per image, the cross-boundary bounding boxes are retained and clipped to image boundary, to be RoI pooled for further *cls* and *regression heads*

# RoI Pooling


Project proposal onto features
Divide projected proposal into 7x7 grid, max-pool within each cell
Fully-connected layers
CNN
Hi-res input image: 3 x 640 x 480 with region proposal
Hi-res conv features: 512 x 20 x 15;
Projected region proposal is e.g. 512 x 18 x 8 (varies per proposal)
RoI conv features: 512 x 7 x 7 for region proposal
Fully-connected layers expect low-res conv features: 512 x 7 x 7
Divide the region proposal region into 7x7 grids, and choose maximum value on each grid
Girshick, "Fast R-CNN", ICCV 2015.

# Faster R-CNN Loss Function

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{box}}$$

$$\mathcal{L}(\{p_i\},\{t_i\}) = \frac{1}{N_{\text{cls}}}\sum_i \mathcal{L}_{\text{cls}}(p_i, p_i^*) + \frac{\lambda}{N_{\text{box}}}\sum_i p_i^* \cdot L_1^{\text{smooth}}(t_i - t_i^*)$$

4 regressor outputs

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

$$\mathcal{L}_{\text{cls}}(p_i, p_i^*) = -p_i^* \log p_i - (1 - p_i^*)\log(1 - p_i)$$

GT Anchor

$$t_x^* = (x^* - x_a)/w_a, \quad t_y^* = (y^* - y_a)/h_a, \quad t_w^* = \log(w^*/w_a), \quad t_h^* = \log(h^*/h_a)$$

Predicted

$$t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a, \quad t_w = \log(w/w_a), \quad t_h = \log(h/h_a)$$

- All $k$ (= 9) of the anchor boxes have different regressors that do not share weights.
- At test time, the learned regression output $t_i$ can be applied to its corresponding anchor box (that is predicted positive), and the $x$, $y$, $w$, $h$ parameters for the predicted object proposal bounding box can be back-calculated

# Training of Faster R-CNN

- RPN selects about 2000 proposals after NMS, which are sent to the two heads (Classification and Regression heads) for evaluations.
- For those RPN proposals with higher IoU values (e.g., > 0.7 with GT) are treated as "positive" samples, otherwise "negative" (IoU < 0.3). The positive-negative ratio is about 1:1, and the total number of samples are about 256 (for testing also about 256 are chosen).
- They are then sent to RoI pooling to be used in classification and regression heads training.

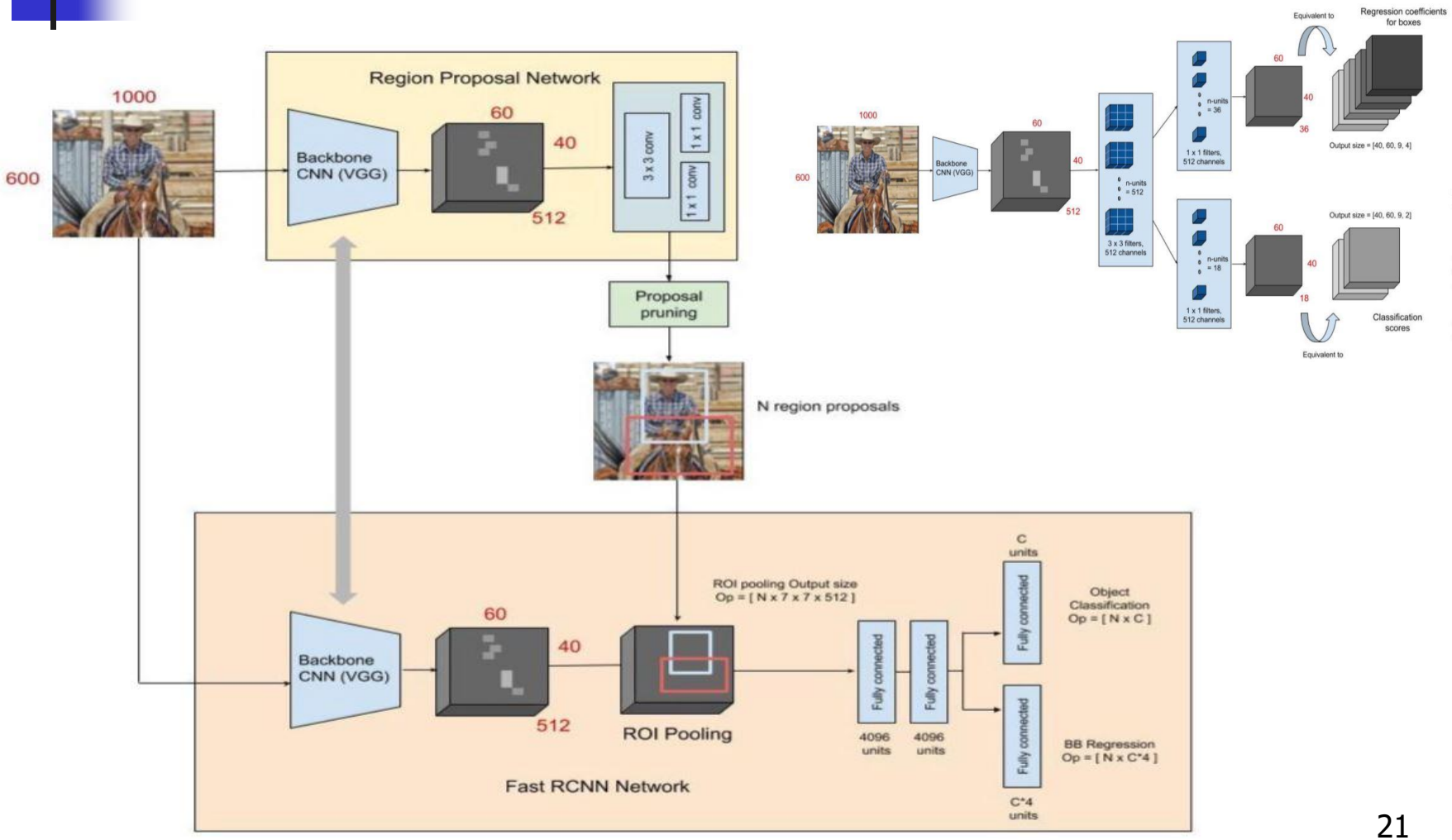
# 4-Step Alternating Training

- The RPN is trained independently based on selected 256 samples. The backbone CNN for this task is initialized with weights from a network trained for an ImageNet classification task, and is then fine-tuned for the region proposal task. (backbone+RPN)
- The RPN weights are fixed and the proposals from the RPN are used to train the Faster R-CNN. (backbone+detector)
- The RPN is now initialized with weights from this Faster R-CNN, and fine-tuned for the region proposal task. This time, weights in the backbone is fixed, and only the layers unique to the RPN are fine-tuned. (RPN)
- Once again using the new RPN, the output classification and regression heads are fine-tuned. Again, only the layers unique to the detector network are fine-tuned and the common layer weights are fixed. (detector)

# Testing of A Faster R-CNN

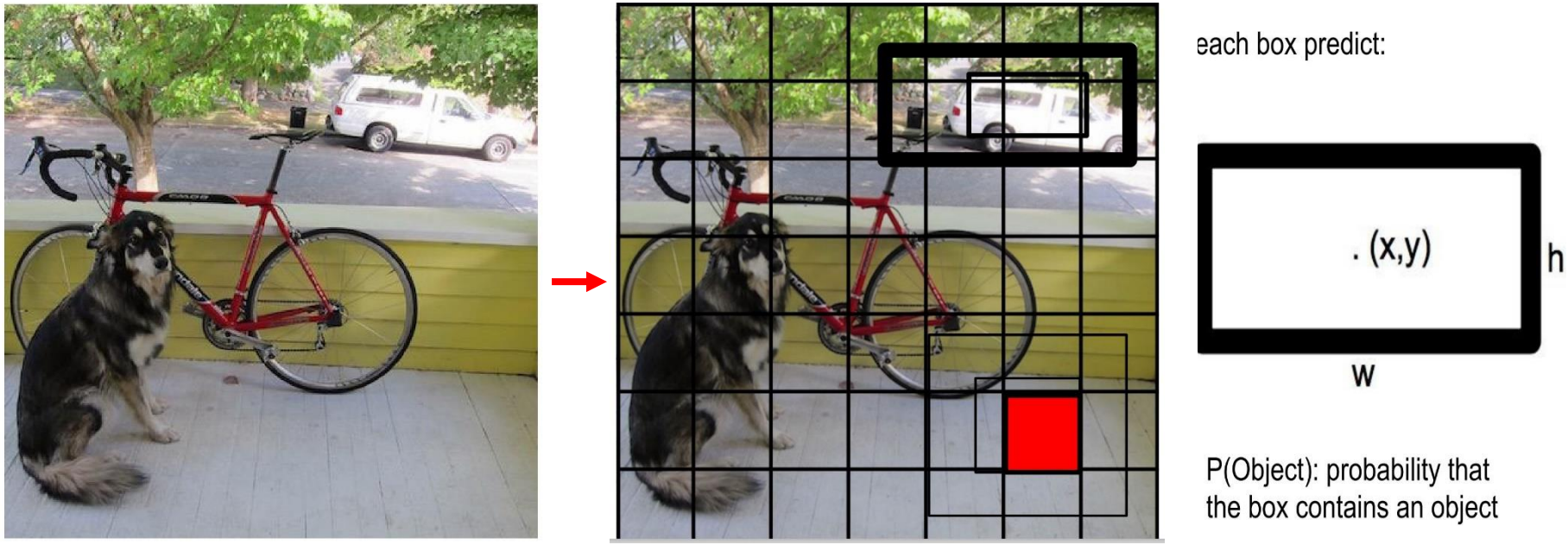
- RPN selects about 2000 proposals, which are sent to the two detector heads (Classification and Regression heads) for evaluations after NMS.
- From the top down of high *cls* scores, about 256 are chosen to go through the RoI pooling and the subsequent classification and regression.
- NMS is again applied to output the final detection results.

# A Complete Pipeline



# You Only Look Once (YOLO): One-Stage Detector

Each cell predicts B boxes(x,y,w,h) and confidences of each box: P(Object)

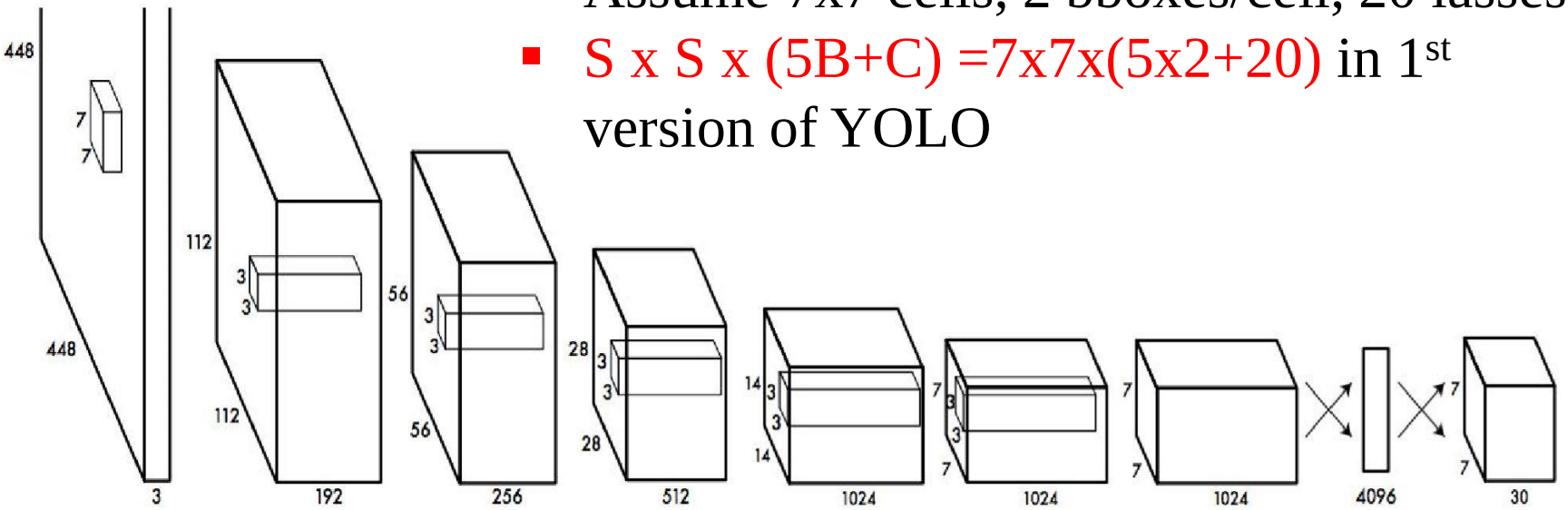



- Predictions are made for each cell in an *S x S* grid.
- B bounding boxes (4 parameters each), B confidence scores *Pr(Obj)*IoU*
- C conditional class probabilities $Pr(Class_i \mid Obj)$
- Output is S x S x (5B+C) tensor

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

# Grid-Cell based Proposals

- Assume 7x7 cells, 2 bboxes/cell, 20 lasses
- S x S x (5B+C) =7x7x(5x2+20) in 1st version of YOLO

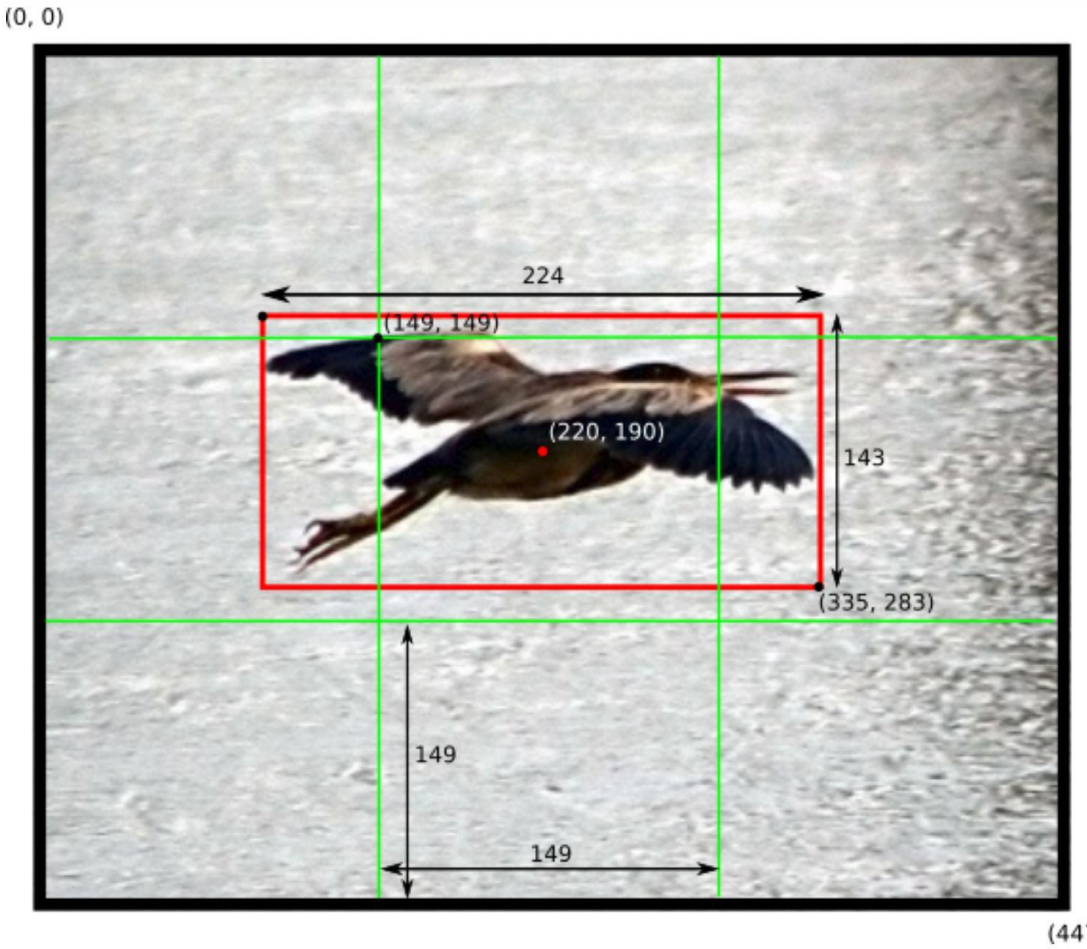



24 convolution layers and
2 fully connected layers

# What to Predict in YOLO?

- Each cell predicts B bounding boxes and confidence scores, {Pr(Object) ∗ IOU(truth, pred)}, for those boxes,
- 5 predictions per box: x, y, w, h, and confidence.
  - The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell.
  - The width and height are predicted relative to the whole image.
  - The confidence prediction represents the IOU between the predicted box and any ground truth box.
- Each grid cell also predicts C conditional class probabilities, $Pr(Class_i|Object)$. These probabilities are conditioned on the grid cell containing an object. Only one set of class probabilities per grid cell, regardless of the number of boxes B.

# BBox in YOLOv1




- The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell.
- The width and height are predicted relative to the whole image.

x = (220-149) / 149 = 0.48

y = (190-149) / 149 = 0.28

w = 224 / 448 = 0.50

h = 143 / 448 = 0.32

# Loss Function for Training

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

$$+ \boxed{\lambda_{\text{coord}} \text{=5}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[ \left( \sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left( \sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

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

All L2 distances (regressor)

$$+ \boxed{\lambda_{\text{noobj}} \text{=0.5}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left( C_i - \hat{C}_i \right)^2$$

not responsible, to avoid overpowering the gradient from cells that do contain objects

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

Only one object class per cell

where $\mathbb{1}_i^{\text{obj}}$ denotes if object appears in cell $i$ and $\mathbb{1}_{ij}^{\text{obj}}$ denotes that the $j$th bounding box predictor in cell $i$ is "responsible" for that prediction.

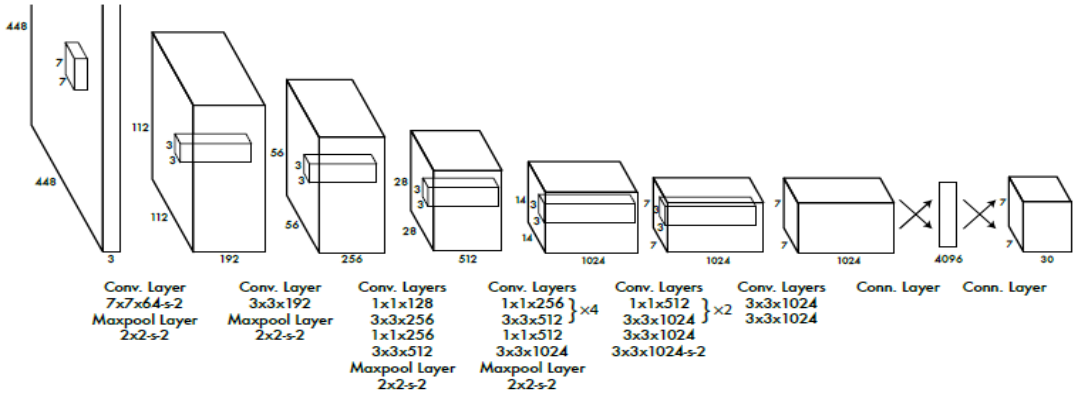
# Training/Testing of YOLO

- Pretrain the convolutional layers on the ImageNet 1000-class competition dataset approximately a week and achieve a single crop top-5 accuracy of 88% on the ImageNet 2012 validation set.
- Use the Darknet framework (neural network programming environment in C) for all training and inference.
- The loss function only penalizes classification error if an object is present in that grid cell (i.e., the conditional class probability).
- 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, one box per cell)
- At test time, multiply conditional class probabilities and the individual box confidence predictions giving class-specific confidence scores for each box

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

# Example for PASCAL VOC

- It divides the image into an S×S grid and for each grid cell predicts/regresses B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an S ×S ×(B ∗5 + C) tensor, (each bounding box consists of 5 predictions: x, y, w, h, and confidence).
- For evaluating YOLO on PASCAL VOC, we use S = 7, B = 2. PASCAL VOC has 20 labelled classes so C = 20. Our final prediction is a 7×7×30 tensor.



# YOLO Detection Performance on VOC 2007

| Real-Time Detectors | Train | mAP | FPS |
|---|---|---|---|
| 100Hz DPM [31] | 2007 | 16.0 | 100 |
| 30Hz DPM [31] | 2007 | 26.1 | 30 |
| Fast YOLO | 2007+2012 | 52.7 | **155** |
| YOLO | 2007+2012 | **63.4** | 45 |
| Less Than Real-Time | | | |
| Fastest DPM [38] | 2007 | 30.4 | 15 |
| R-CNN Minus R [20] | 2007 | 53.5 | 6 |
| Fast R-CNN [14] | 2007+2012 | 70.0 | 0.5 |
| Faster R-CNN VGG-16[28] | 2007+2012 | 73.2 | 7 |
| Faster R-CNN ZF [28] | 2007+2012 | 62.1 | 18 |
| YOLO VGG-16 | 2007+2012 | 66.4 | 21 |

- The PASCAL Visual Object Classes (VOC) 2007/2012 dataset contains 20 object categories including vehicles, household, animals, and other: aeroplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, potted plant, sofa, TV/monitor, bird, cat, cow, dog, horse, sheep, and person.
- Total images (train/valid/test) are 9963/11,530 (2007/2012)

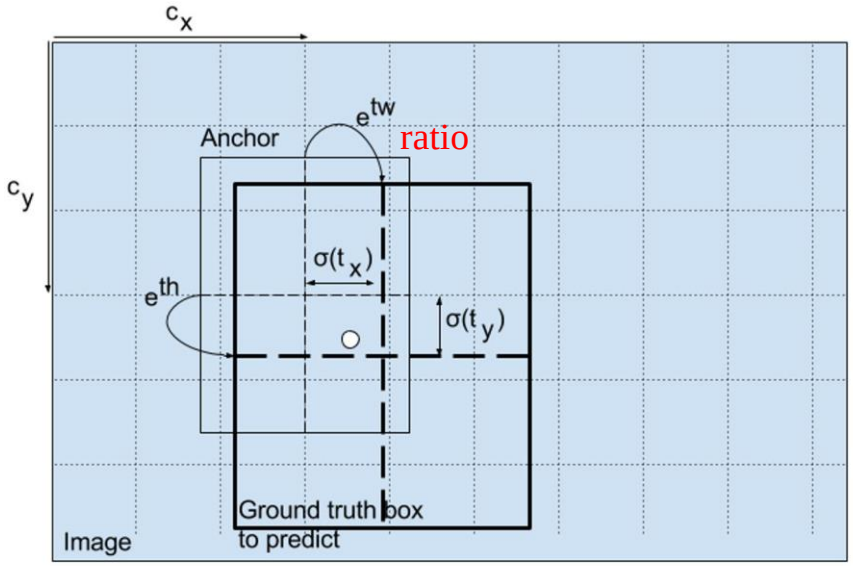
# YOLO v2 Extension

- Batch normalization (mAP 2% better)
- High Resolution ImageNet pre-trained classifier, 448x448 (4% better)
- Better predict the box center location and size based on clustered 5 or 9 anchor boxes priors (5% better)
- Finer grained feature, 13x13x2048 (1% better)

Joseph Redmon and Ali Farhadi, "YOLO9000: Better, Faster, Stronger," CVPR 2017

# YOLOv2 Predicting Outputs

- YOLOv2 predicts 5-9 clustered bounding boxes (anchors) at each cell in the output feature map. The network still predicts 5 box parameters for each bounding box, $t_x$, $t_y$, $t_w$, $t_h$, & $t_o$
- The cell is offset from the top left corner of the image by ($c_x$, $c_y$) and clustered bounding box size prior has width and height $p_w$, $p_h$




Sigmoid function

$$b_x = \sigma(t_x) + c_x$$

$$b_y = \sigma(t_y) + c_y$$

$$b_w = p_w e^{t_w}$$

$$b_h = p_h e^{t_h}$$

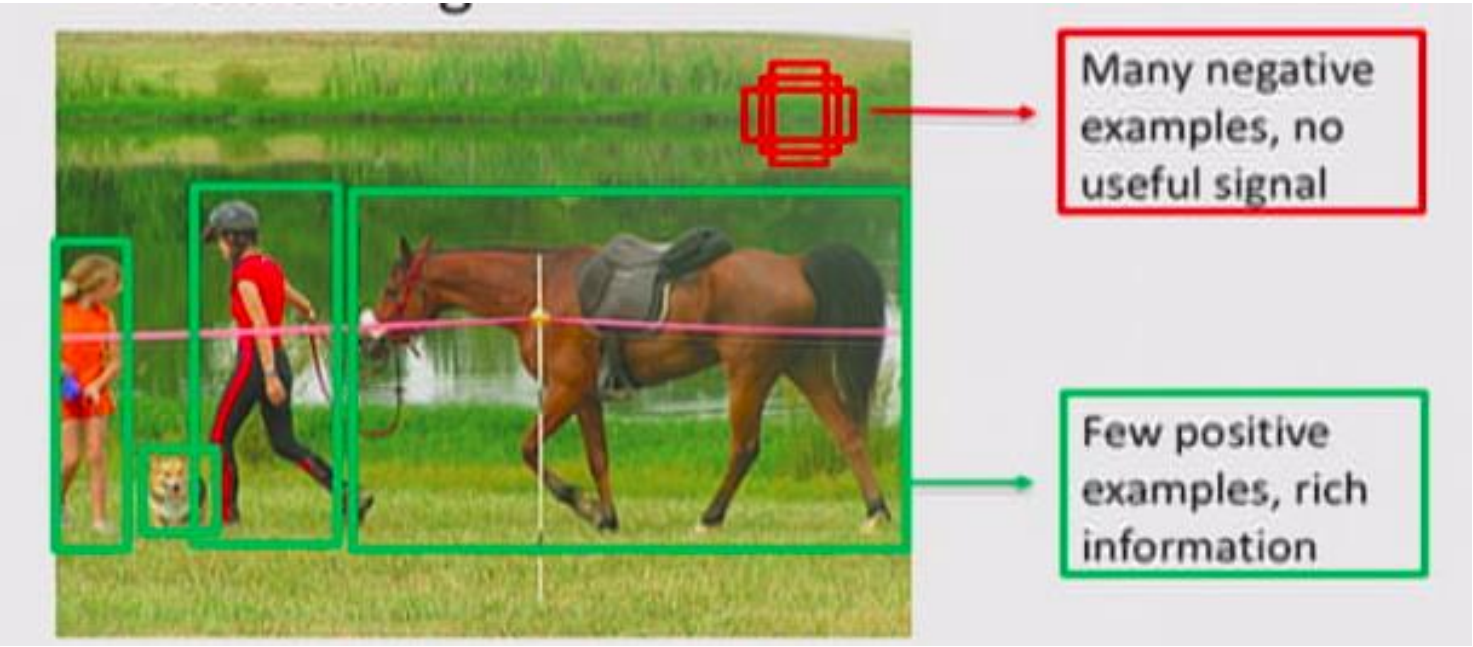
$$Pr(\text{object}) * IOU(b, \text{object}) = \sigma(t_o)$$

# YOLOv2 Performance on VOC 2007

| Detection Frameworks | Train | mAP | FPS |
|---|---|---|---|
| Fast R-CNN [5] | 2007+2012 | 70.0 | 0.5 |
| Faster R-CNN VGG-16[15] | 2007+2012 | 73.2 | 7 |
| Faster R-CNN ResNet[6] | 2007+2012 | 76.4 | 5 |
| YOLO [14] | 2007+2012 | 63.4 | 45 |
| SSD300 [11] | 2007+2012 | 74.3 | 46 |
| SSD500 [11] | 2007+2012 | 76.8 | 19 |
| YOLOv2 288 × 288 | 2007+2012 | 69.0 | 91 |
| YOLOv2 352 × 352 | 2007+2012 | 73.7 | 81 |
| YOLOv2 416 × 416 | 2007+2012 | 76.8 | 67 |
| YOLOv2 480 × 480 | 2007+2012 | 77.8 | 59 |
| YOLOv2 544 × 544 | 2007+2012 | **78.6** | 40 |

# Imbalanced Training in One-Stage Detector

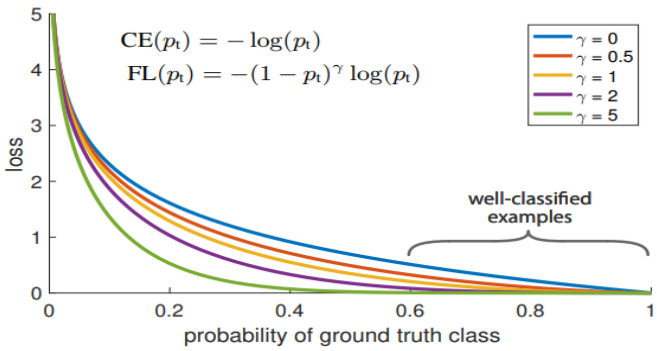
- The classifier used in One-Stage Detector gets more negative samples (1000:1, imbalanced), e.g., background, (or easier training samples to be more specific) compared to positive samples, thereby causing more biased learning.



# Focal Loss

- Traditional Cross-Entropy $\mathrm{CE}(y_t) = -y_t \log(p_t)$, $y_t = 1$ *or* $0$.
- α-balanced $\mathrm{CE}(y_t) = -\alpha \log(p_t)$, $\alpha \in [0, 1]$, for $y_t = 1$, $-(1-\alpha) \log(p_t)$ for $y_t = 0$
- The **focal loss** penalizes easily classified examples, i.e., background (negative examples).
- For easy negative examples to the loss, $(1\text{-}p_t)^r$ are multiplied with their original loss values, eventually diminishing their losses (*r=2 best*).

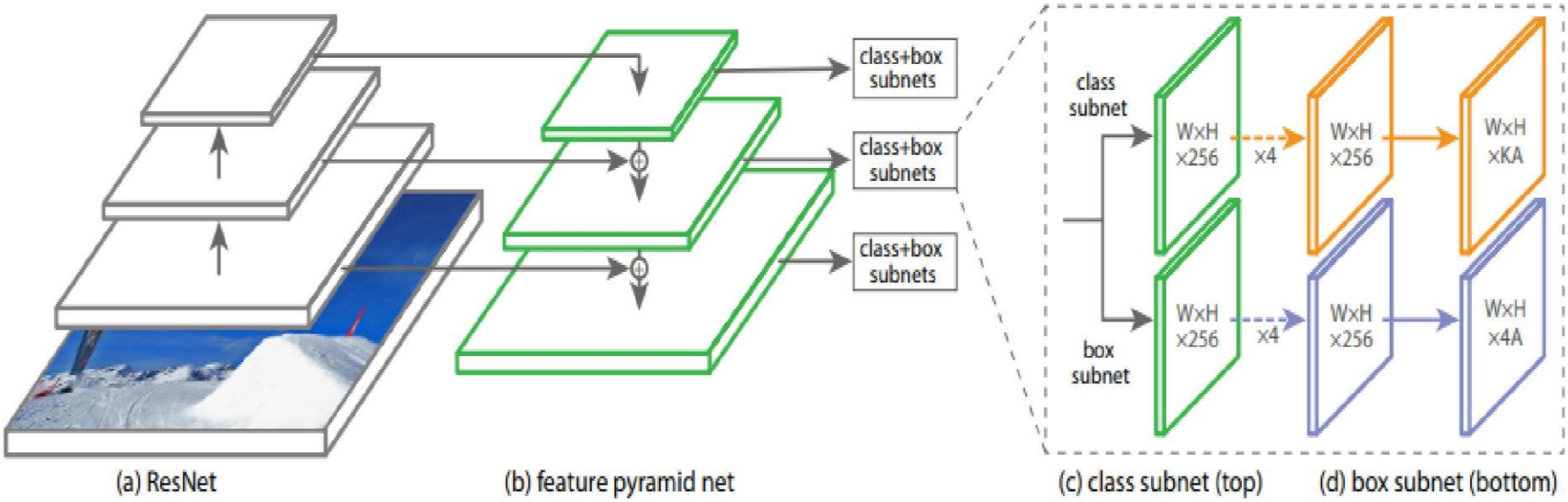
$$\mathrm{FL}(p_t) = -(1-p_t)^{\gamma} \log(p_t)$$



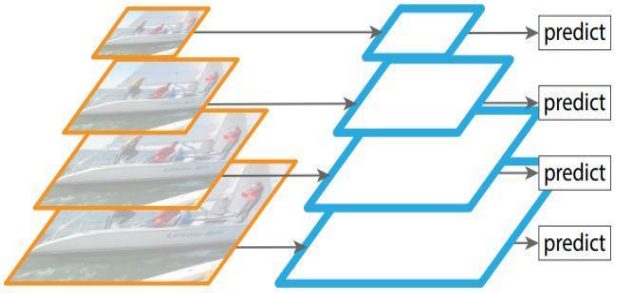
# RetinaNet

- Single shot (one-stage) detectors are not as accurate as two-stage object detectors
  - Focal Loss
  - Feature Pyramid Network (FPN)

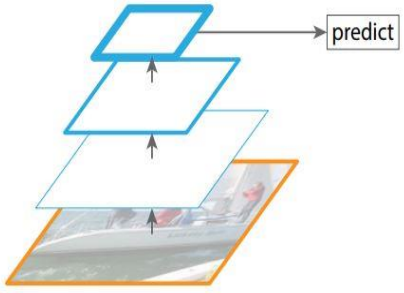
T.-Y. Lin, et al. "Focal loss for dense object detection," ICCV, 2017.



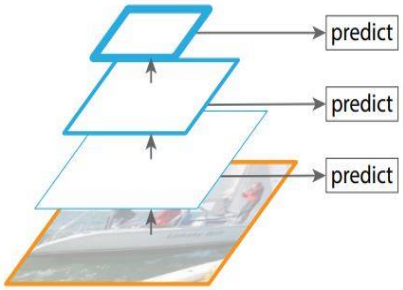
# Feature Pyramid Network



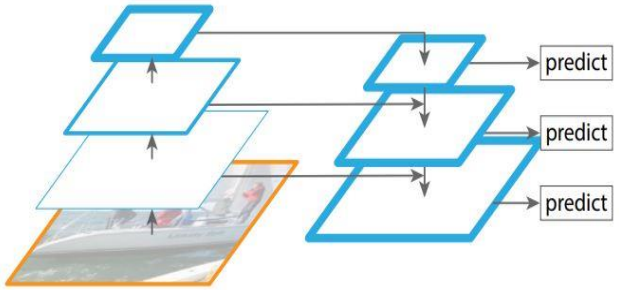

(a) Featurized image pyramid



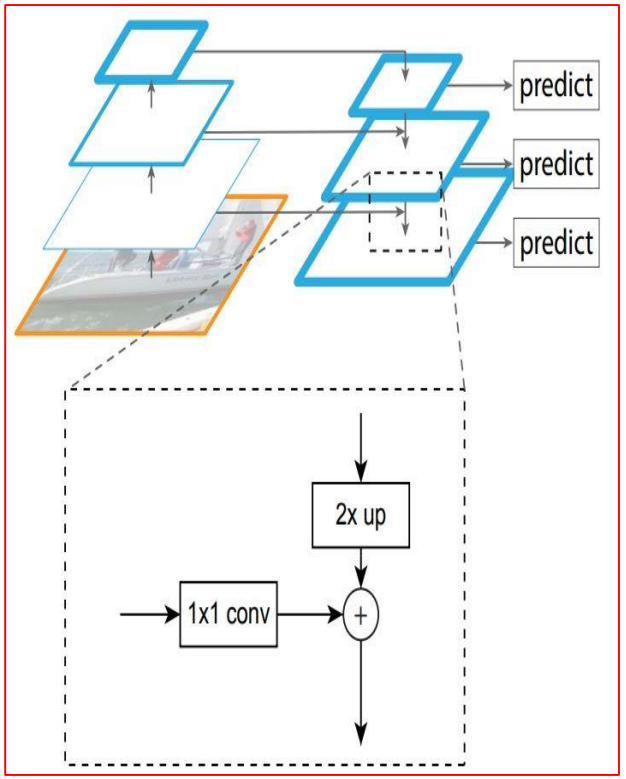

(b) Single feature map



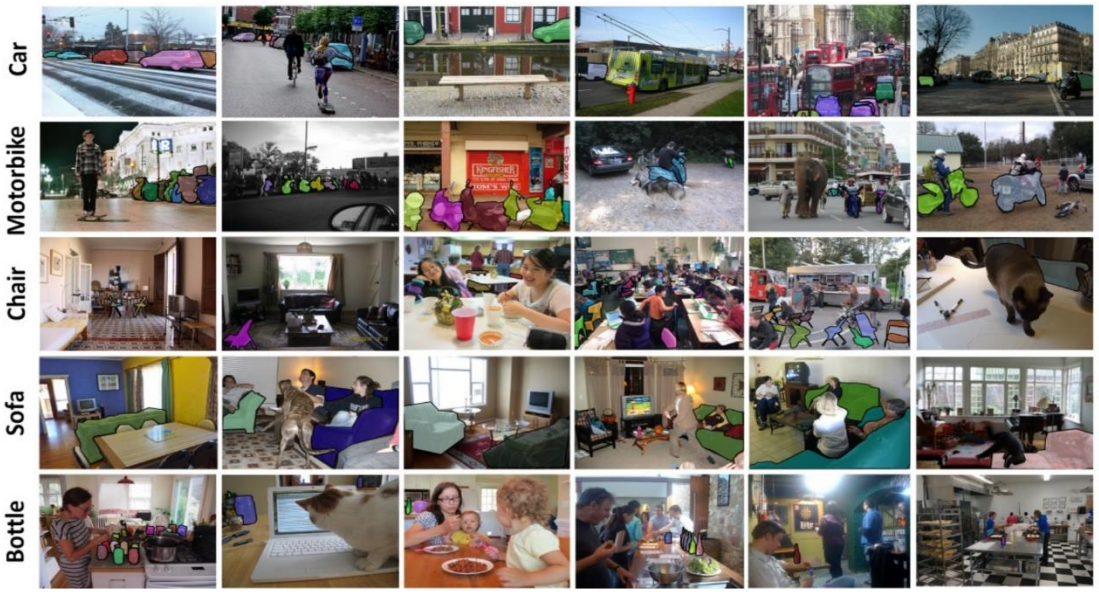

(c) Pyramidal feature hierarchy




(d) Feature Pyramid Network



# RetinaNet Performance

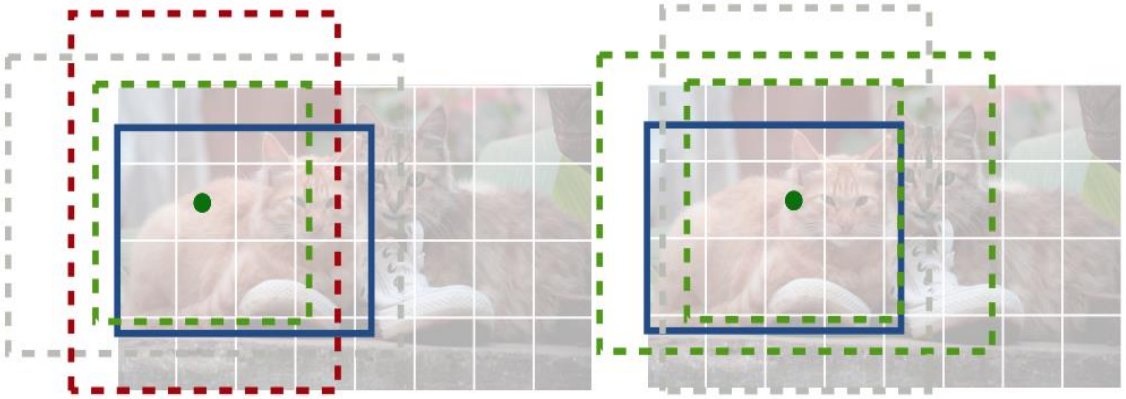



MS CoCo Test-Dev Data:

- 330K images (>200K labeled)
- 1.5 million object instances
- 80 object categories
- 91 stuff categories

| | backbone | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|
| *Two-stage methods* | | | | | | | |
| Faster R-CNN+++ [16] | ResNet-101-C4 | 34.9 | 55.7 | 37.4 | 15.6 | 38.7 | 50.9 |
| Faster R-CNN w FPN [20] | ResNet-101-FPN | 36.2 | 59.1 | 39.0 | 18.2 | 39.0 | 48.2 |
| Faster R-CNN by G-RMI [17] | Inception-ResNet-v2 [34] | 34.7 | 55.5 | 36.7 | 13.5 | 38.1 | 52.0 |
| Faster R-CNN w TDM [32] | Inception-ResNet-v2-TDM | 36.8 | 57.7 | 39.2 | 16.2 | 39.8 | **52.1** |
| *One-stage methods* | | | | | | | |
| YOLOv2 [27] | DarkNet-19 [27] | 21.6 | 44.0 | 19.2 | 5.0 | 22.4 | 35.5 |
| SSD513 [22, 9] | ResNet-101-SSD | 31.2 | 50.4 | 33.3 | 10.2 | 34.5 | 49.8 |
| DSSD513 [9] | ResNet-101-DSSD | 33.2 | 53.3 | 35.2 | 13.0 | 35.4 | 51.1 |
| **RetinaNet** (ours) | ResNet-101-FPN | 39.1 | 59.1 | 42.3 | 21.8 | 42.7 | 50.2 |
| **RetinaNet** (ours) | ResNeXt-101-FPN | **40.8** | **61.1** | **44.1** | **24.1** | **44.2** | 51.2 |

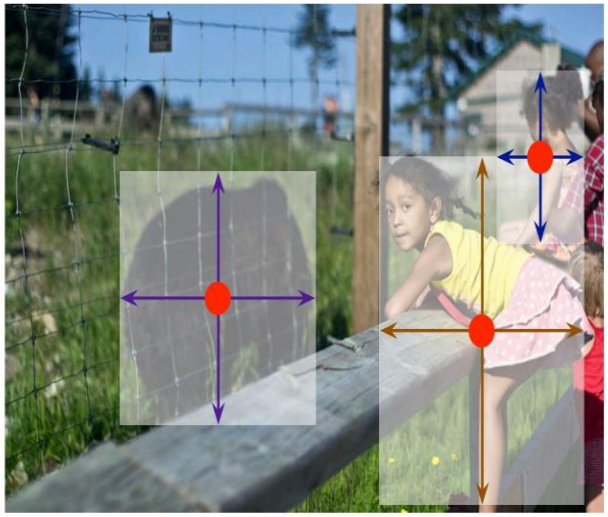
# Other Issues of the Detectors

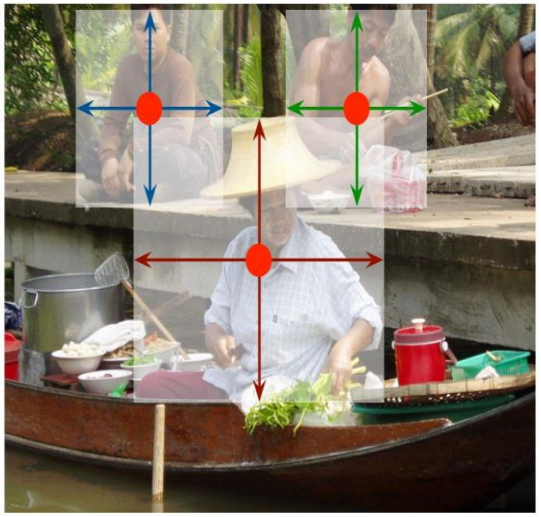


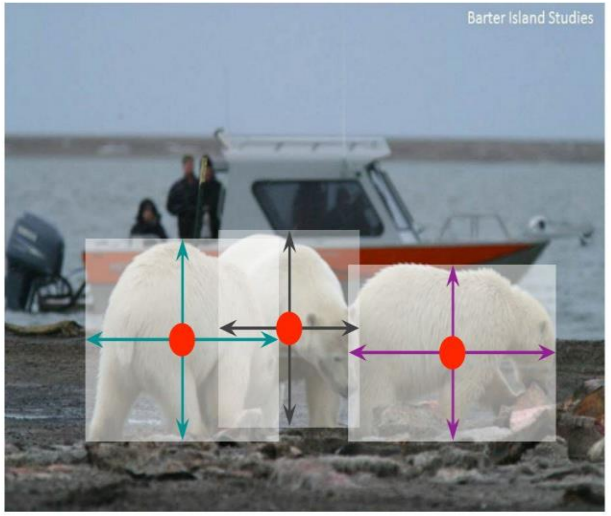
- **Pre-defined**: manually or clustering designed anchors (sizes and aspect ratios).
- **Incorrect**: anchors are not aligned with the ground truth boxes.
- **Wasteful**: need to enumerate all possible object locations and anchors.
- Need to do **post-processing**, e.g., NMS.

# CenterNet: Objects as Points

- Model an object as the center point of bounding box.
- Object Sizs can be regressed directly from image features at this center point



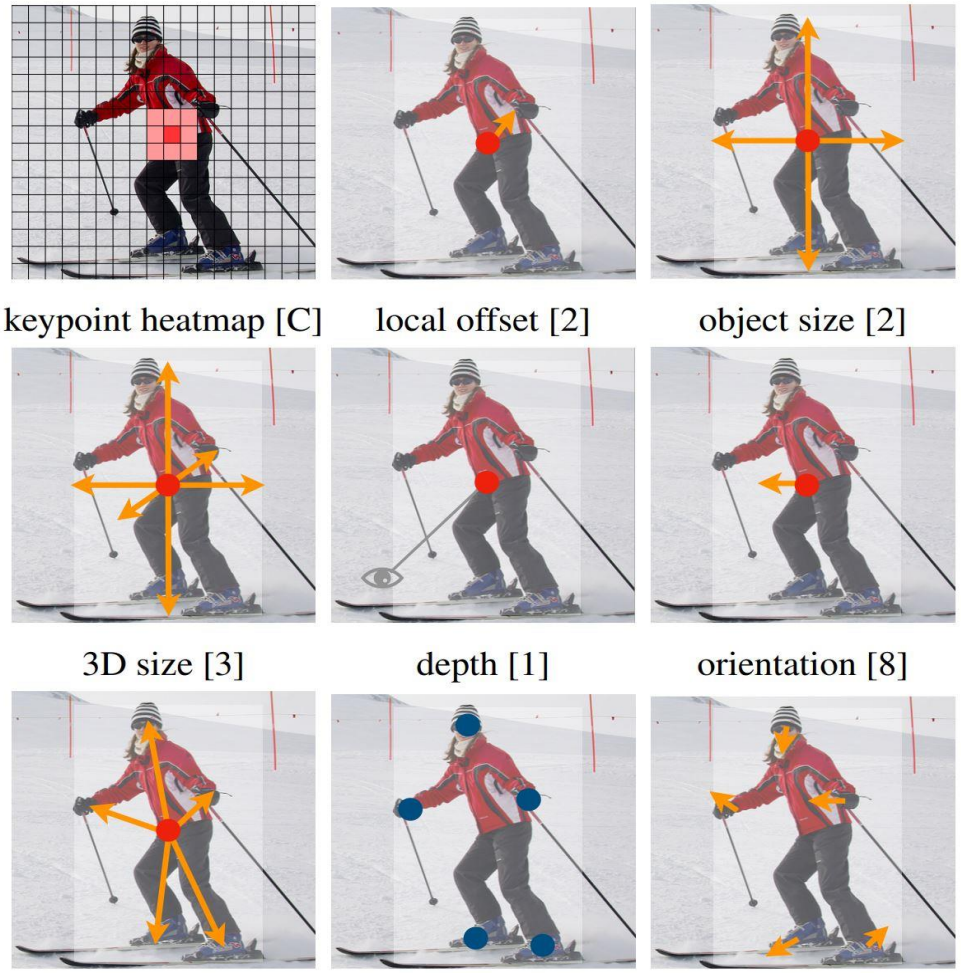





Xingyi Zhou et al., “Objects as Points,” 2019, https://arxiv.org/abs/1904.07850

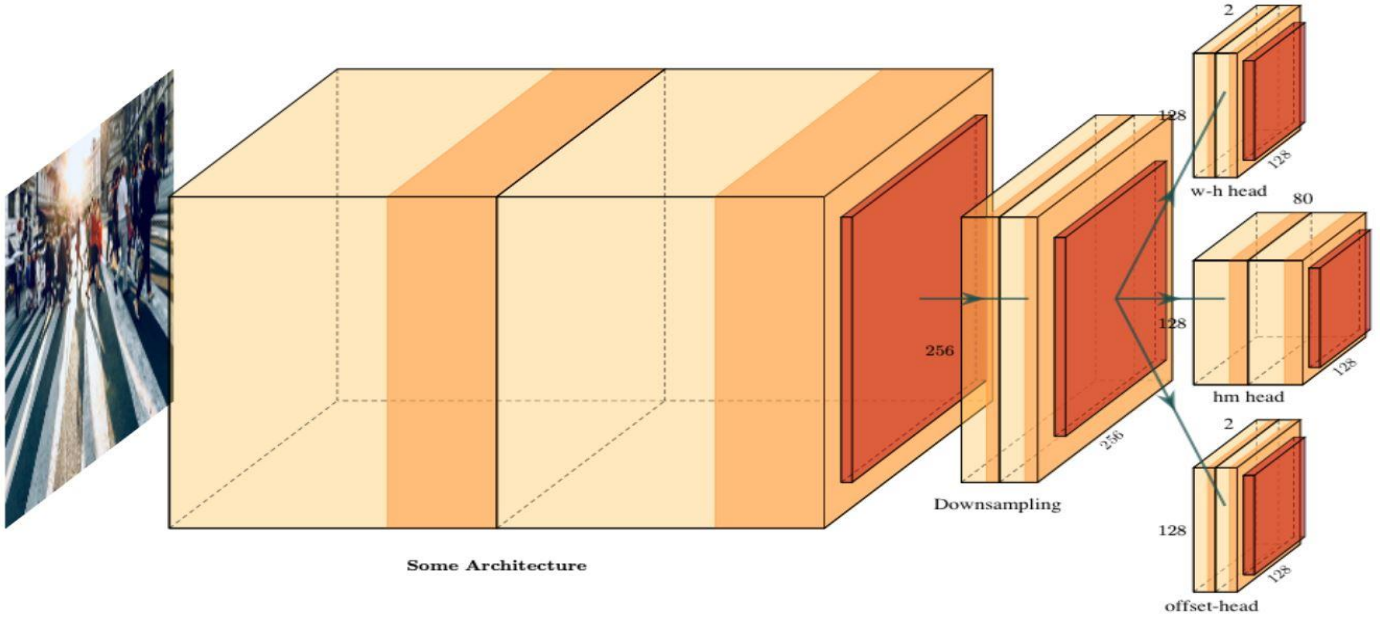
# CenterNet: Objects as Points

- Other properties can also be regressed directly from image features at this center point: e.g., dimension, 3D extent, orientation, and pose




Xingyi Zhou et al., "Objects as Points," 2019, https://arxiv.org/abs/1904.07850

# CenterNet Architecture




One map per class (COCO)

- HeatMap head: predict the box center
- W-H head: predict the box size, width and height
- Offset head: recover from the discretization error caused due to the downsampling of the input

# CenterNet Loss Function

- A bounding box can be specified by 2 corners $(x_1^{(k)}, y_1^{(k)}, x_2^{(k)}, y_2^{(k)})$
- Focal Loss for predicted center point (for classification)

$$L_k = \frac{-1}{N} \sum_{xyc} \begin{cases} (1-\hat{Y}_{xyc})^{\alpha} \log(\hat{Y}_{xyc}) & \text{if } Y_{xyc} = 1 \\ (1-Y_{xyc})^{\beta} (\hat{Y}_{xyc})^{\alpha} & \text{otherwise} \\ \log(1-\hat{Y}_{xyc}) & \end{cases} \qquad Y_{xyc} = \exp\left(-\frac{(x-\tilde{p}_x)^2 + (y-\tilde{p}_y)^2}{2\sigma_p^2}\right)$$

- For each ground truth keypoint $p \in R^2$ of class c, the low-resolution equivalent $\tilde{p} = \lfloor \frac{p}{R} \rfloor$, the offset loss due to quantization

quantization noise

$$L_{off} = \frac{1}{N} \sum_{p} \left| \hat{O}_{\tilde{p}} - \left( \frac{p}{R} - \tilde{p} \right) \right| \qquad \hat{O} \in \mathcal{R}^{\frac{W}{R} \times \frac{H}{R} \times 2}$$

- Bounding box size loss based on two corner points

$$s_k = (x_2^{(k)} - x_1^{(k)}, y_2^{(k)} - y_1^{(k)}) \qquad L_{size} = \frac{1}{N} \sum^{N} \left| \hat{S}_{p_k} - s_k \right|$$

- Total Loss function $L_{det} = L_k + \lambda_{size} L_{size} + \lambda_{off} L_{off}$

# CenterNet Detection Inference

- First extract the top 100 peaks in the (CenterPoint) heatmap for each category independently, i.e., detect all responses whose value is greater or equal to its 8-connected neighbors

$$\hat{\mathcal{P}} = \{(\hat{x}_i, \hat{y}_i)\}_{i=1}^{n} \text{ of class } c$$

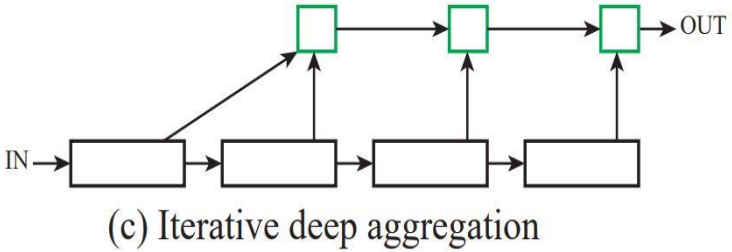
- Each keypoint location is given by an integer coordinates $(x_i , y_i)$. We use the keypoint values $\hat{Y}_{x_i y_i c}$ as a measure of its detection confidence, and produce a bounding box at location

$$\begin{aligned}&(\hat{x}_i + \delta\hat{x}_i - \hat{w}_i/2,\ \ \hat{y}_i + \delta\hat{y}_i - \hat{h}_i/2,\\ &\ \hat{x}_i + \delta\hat{x}_i + \hat{w}_i/2,\ \ \hat{y}_i + \delta\hat{y}_i + \hat{h}_i/2)\end{aligned}$$

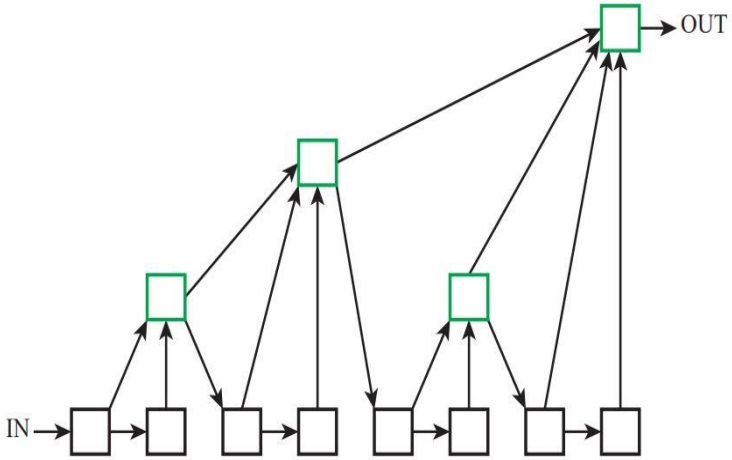
where $(\delta\hat{x}_i, \delta\hat{y}_i) = \hat{O}_{\hat{x}_i, \hat{y}_i}$ is the offset prediction and $(\hat{w}_i, \hat{h}_i) = \hat{S}_{\hat{x}_i, \hat{y}_i}$ is the size prediction.

# Deep Layer Aggregation

- Iterative Deep Aggregation: shallow features are refined as they are propagated through different stages of aggregation.
- Hierarchical Deep Aggregation: shallower and deeper layers are combined to learn richer combinations that span more of the feature hierarchy.



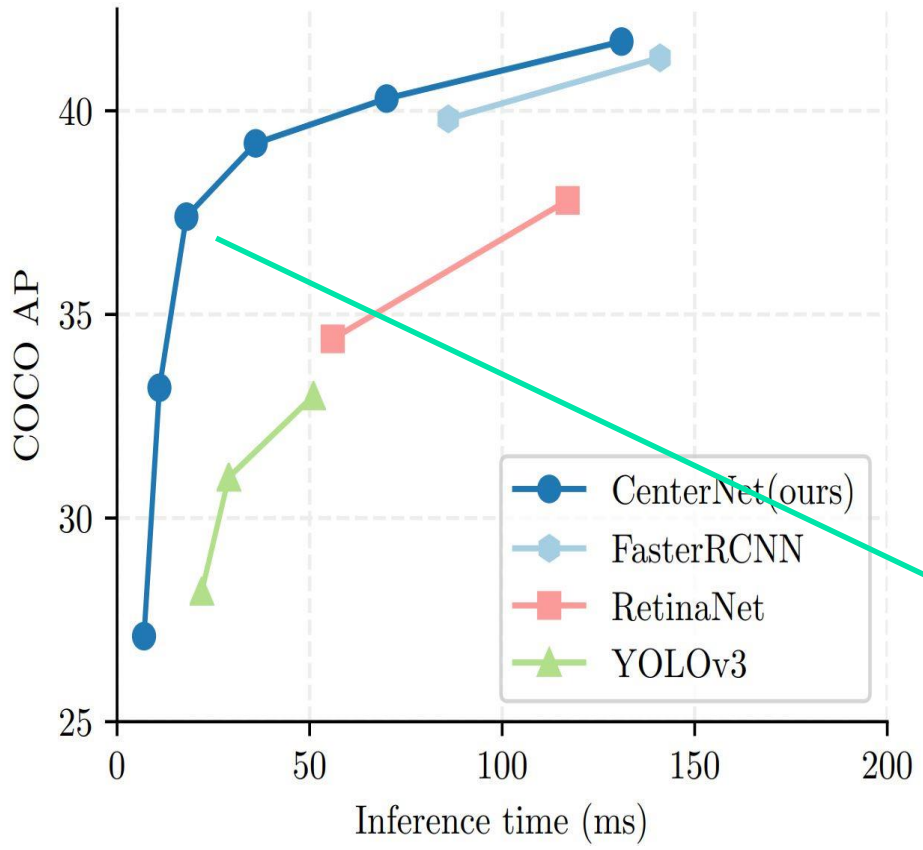
(c) Iterative deep aggregation



(f) Hierarchical deep aggregation

Fisher Yu, et al., "Deep Layer Aggregation,"CVPR 2018, https://arxiv.org/pdf/1707.06484.pdf

# Object Detection Speed-Accuracy Tradeoff (COCO)




**Hourglass-104:**
CenterNet: 42.2% AP in 7.8 FPS
CornerNet: 40.6% AP in 4.1 FPS

**ResNet-101:**
CenterNet: 34.8% AP in 45 FPS
RetinaNet: 34.4% AP in 18 FPS

**ResNet-18:**
CenterNet: 28.1% AP in **142** FPS

**Best speed-accuracy tradeoff:**
CenterNet: 37.4% AP in 52 FPS
YOLOv3: 33.0% AP in 20 FPS

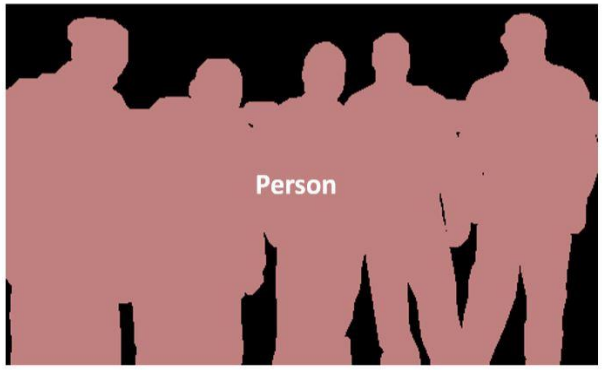
# CenterNet Object Detection Results on (COCO)

| | Backbone | FPS | $AP$ | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|---|
| Two Stage Detectors | | | | | | | | |
| MaskRCNN [21] | ResNeXt-101 | **11** | 39.8 | 62.3 | 43.4 | 22.1 | 43.2 | 51.2 |
| Deform-v2 [63] | ResNet-101 | - | 46.0 | 67.9 | 50.8 | 27.8 | 49.1 | 59.5 |
| SNIPER [48] | DPN-98 | 2.5 | 46.1 | 67.0 | 51.6 | 29.6 | 48.9 | 58.1 |
| PANet [35] | ResNeXt-101 | - | 47.4 | 67.2 | 51.8 | 30.1 | **51.7** | 60.0 |
| TridentNet [31] | ResNet-101-DCN | 0.7 | **48.4** | **69.7** | **53.5** | **31.8** | 51.3 | **60.3** |
| One Stage Detectors | | | | | | | | |
| YOLOv3 [45] | DarkNet-53 | 20 | 33.0 | 57.9 | 34.4 | 18.3 | 25.4 | 41.9 |
| RetinaNet [33] | ResNeXt-101-FPN | 5.4 | 40.8 | 61.1 | 44.1 | 24.1 | 44.2 | 51.2 |
| RefineDet [59] | ResNet-101 | - | 36.4 / 41.8 | 57.5 / 62.9 | 39.5 / 45.7 | 16.6 / 25.6 | 39.9 / 45.1 | 51.4 / 54.1 |
| CornerNet [30] | Hourglass-104 | 4.1 | 40.5 / 42.1 | 56.5 / 57.8 | 43.1 / 45.3 | 19.4 / 20.8 | 42.7 / 44.8 | **53.9** / 56.7 |
| ExtremeNet [61] | Hourglass-104 | 3.1 | 40.2 / 43.7 | 55.5 / 60.5 | 43.2 / 47.0 | 20.4 / 24.1 | 43.2 / 46.9 | 53.1 / 57.6 |
| FSAF [62] | ResNeXt-101 | 2.7 | **42.9** / 44.6 | **63.8** / **65.2** | **46.3** / 48.6 | **26.6** / **29.7** | **46.2** / **47.1** | 52.7 / 54.6 |
| CenterNet-DLA | DLA-34 | **28** | 39.2 / 41.6 | 57.1 / 60.3 | 42.8 / 45.1 | 19.9 / 21.5 | 43.0 / 43.9 | 51.4 / 56.0 |
| CenterNet-HG | Hourglass-104 | 7.8 | 42.1 / **45.1** | 61.1 / 63.9 | 45.9 / **49.3** | 24.1 / 26.6 | 45.5 / **47.1** | 52.8 / **57.7** |

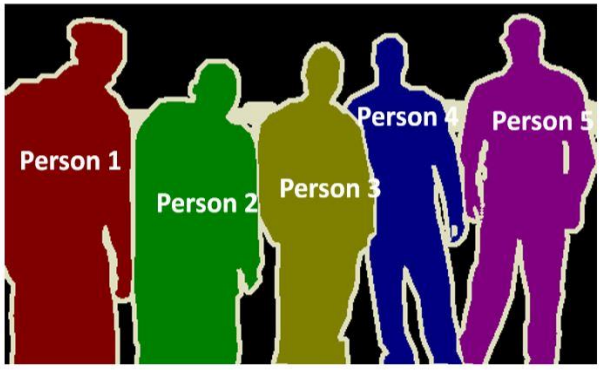
Chenchen Zhu et al, "Feature Selective Anchor-Free (FSAF) Module for Single-Shot Object Detection, CVPR 2019

# Region Based CNN– Segmentation

Object Detection




Semantic Segmentation

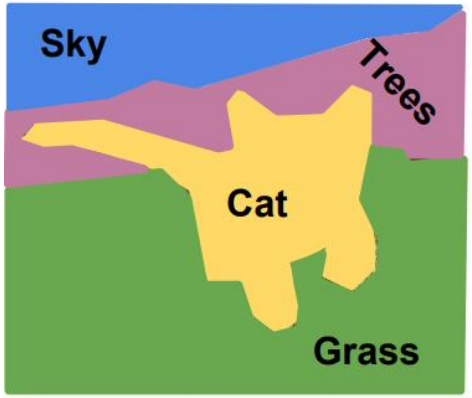


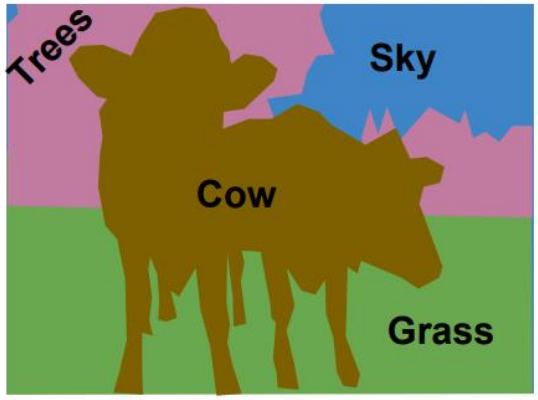

**Instance Segmentation**

# Semantic Segmentation

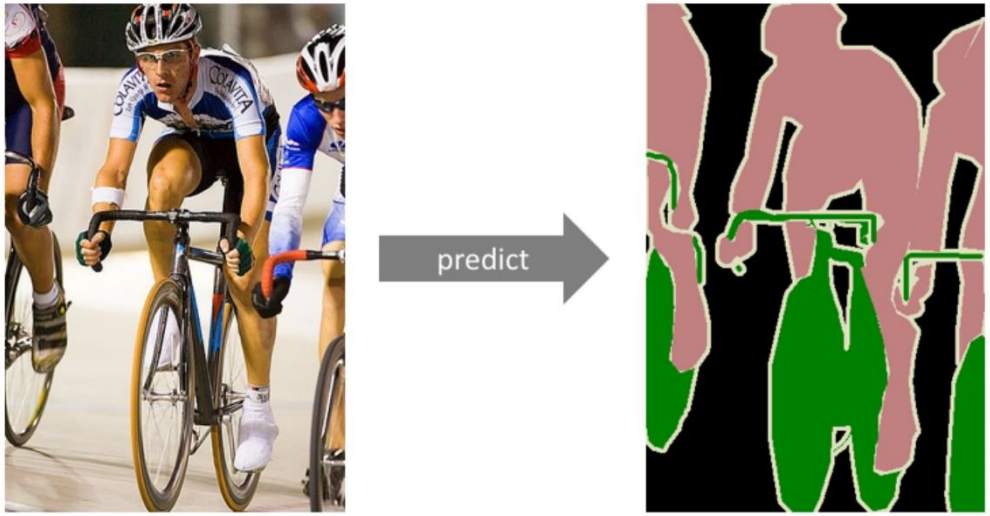

Label each pixel in the image with a category label

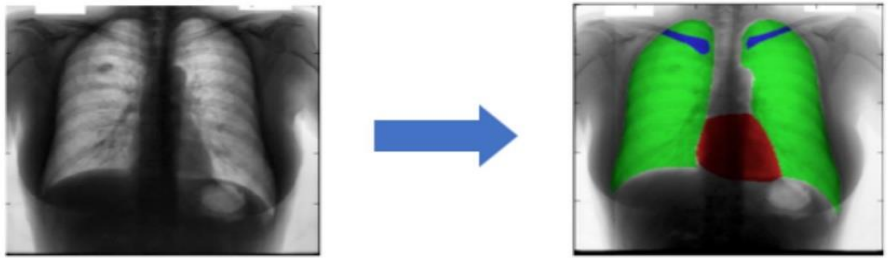
Don't differentiate instances, only care about pixels

# Semantic Segmentation



Input Image

Segmented Image

A chest x-ray with the heart (red), lungs (green), and clavicles (blue) are segmented. (Source)

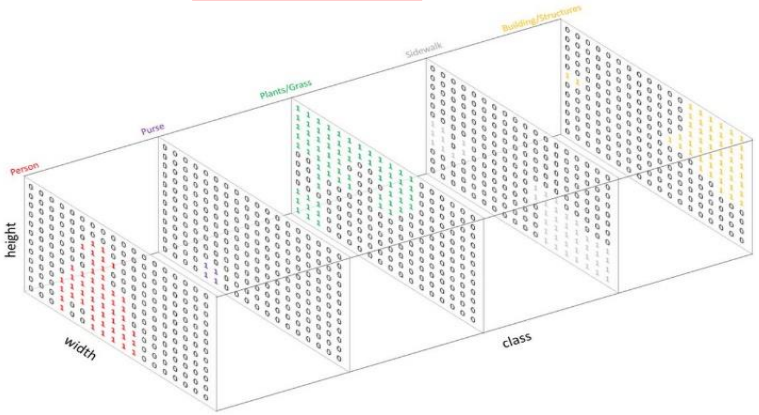
# Multi-Class Segmentation

segmented

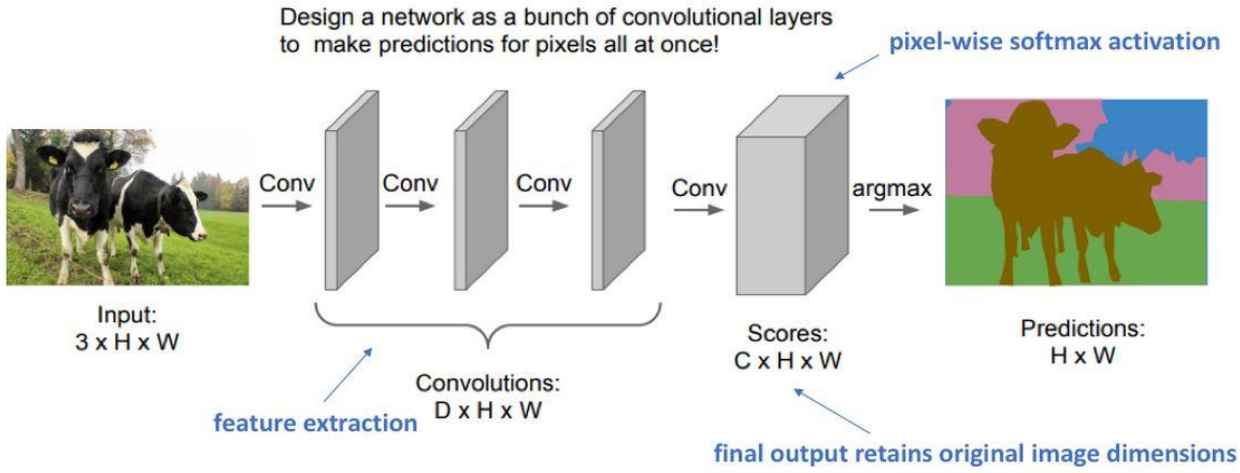
1: Person
2: Purse
3: Plants/Grass
4: Sidewalk
5: Building/Structures

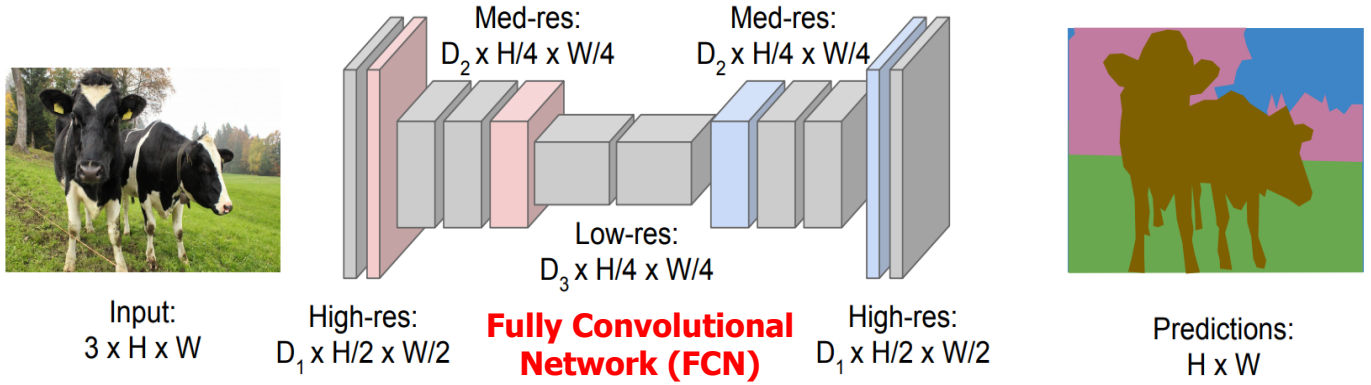
0: Background/Unknown
1: Person
2: Purse
3: Plants/Grass
4: Sidewalk
5: Building/Structures

Semantic Labels

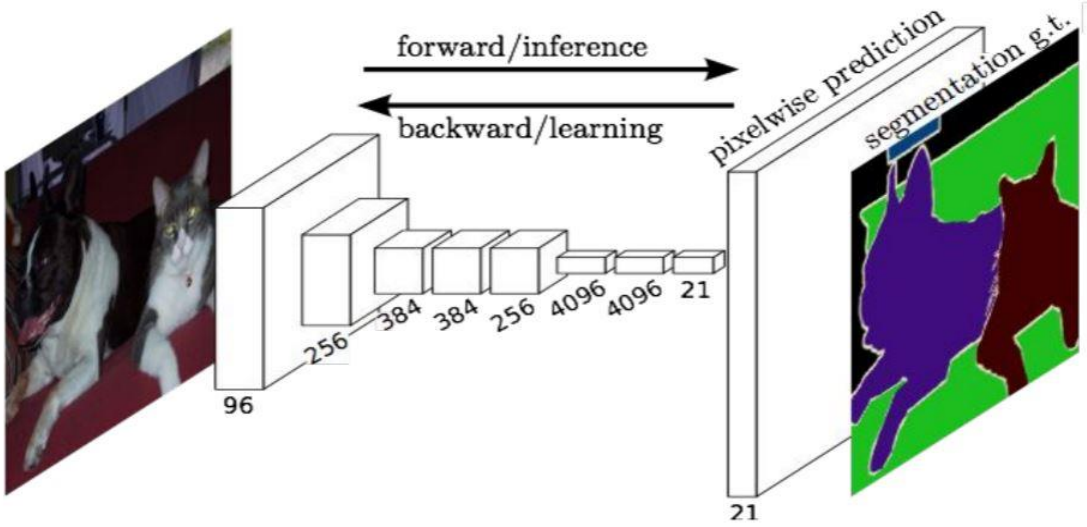


# Fully Convolutional Network

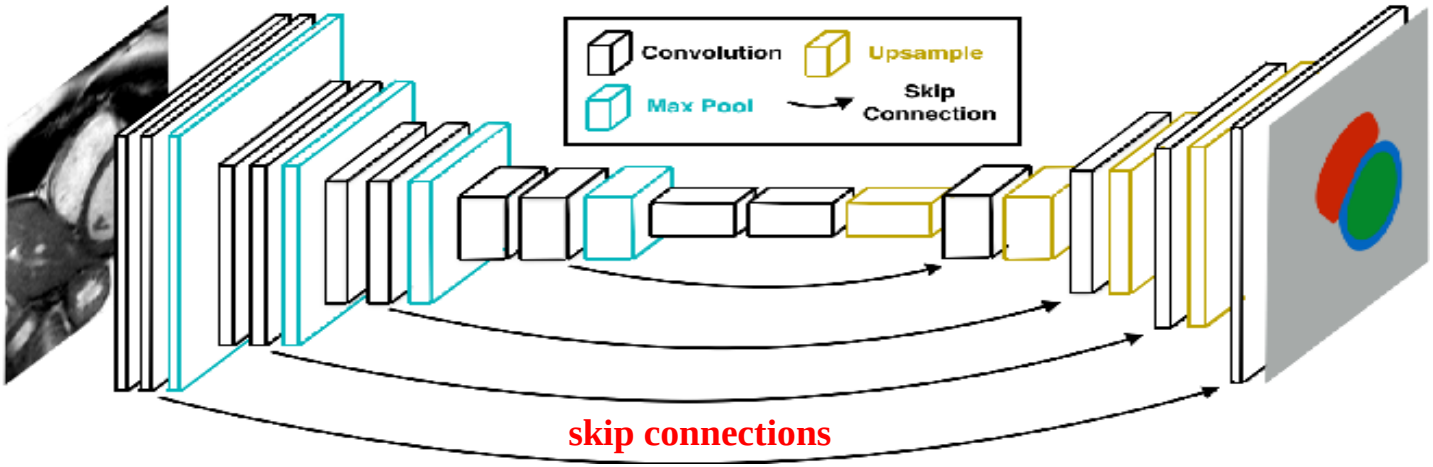


# Fully Convolution Network (FCN) for Semantic Segmentation

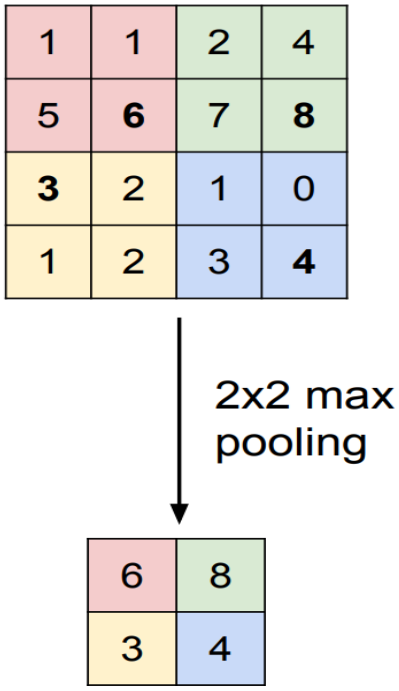



Jonathan Long, et al., "Fully convolutional networks for semantic segmentation," IEEE CVPR 2015

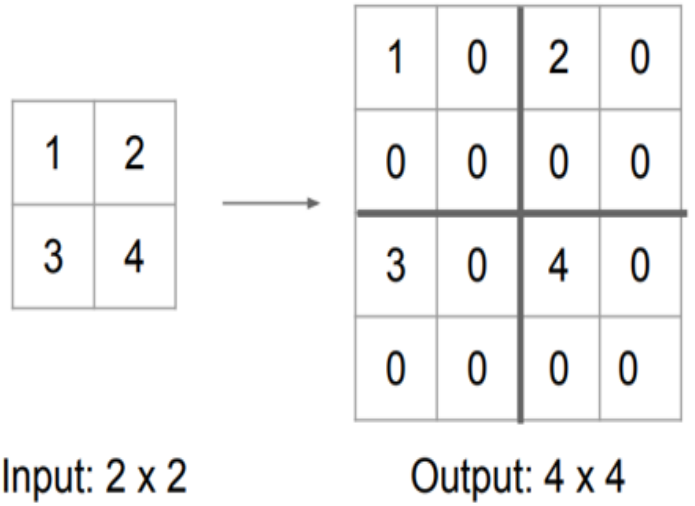


# Pooling and UnPooling

Pooling




UnPooling



# Unpooling Methods

**Nearest Neighbor**

| | |
|---|---|
| 1 | 2 |
| 3 | 4 |

→

| | | | |
|---|---|---|---|
| 1 | 1 | 2 | 2 |
| 1 | 1 | 2 | 2 |
| 3 | 3 | 4 | 4 |
| 3 | 3 | 4 | 4 |

Input: 2 x 2

Output: 4 x 4

**"Bed of Nails"**

| | |
|---|---|
| 1 | 2 |
| 3 | 4 |

→

| | | | |
|---|---|---|---|
| 1 | 0 | 2 | 0 |
| 0 | 0 | 0 | 0 |
| 3 | 0 | 4 | 0 |
| 0 | 0 | 0 | 0 |

Input: 2 x 2

Output: 4 x 4

# Unpooling Methods

**Max Pooling**
Remember which element was max!

| | | | |
|---|---|---|---|
| 1 | 2 | 6 | 3 |
| 3 | 5 | 2 | 1 |
| 1 | 2 | 2 | 1 |
| 7 | 3 | 4 | 8 |

Input: 4 x 4

| | |
|---|---|
| 5 | 6 |
| 7 | 8 |

Output: 2 x 2

Rest of the network

**Max Unpooling**
Use positions from pooling layer

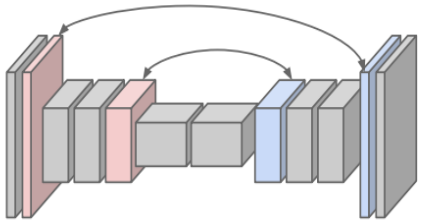
| | |
|---|---|
| 1 | 2 |
| 3 | 4 |

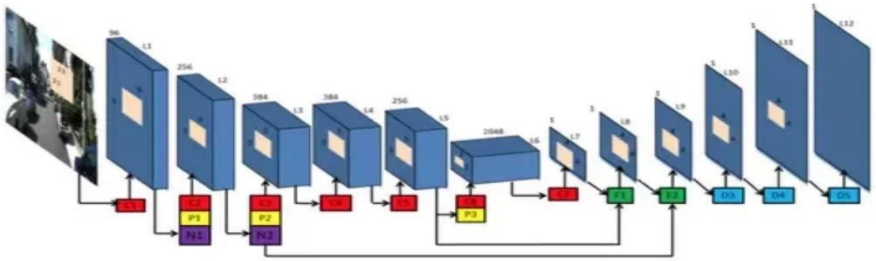
Input: 2 x 2

| | | | |
|---|---|---|---|
| 0 | 0 | 2 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 4 |

Output: 4 x 4

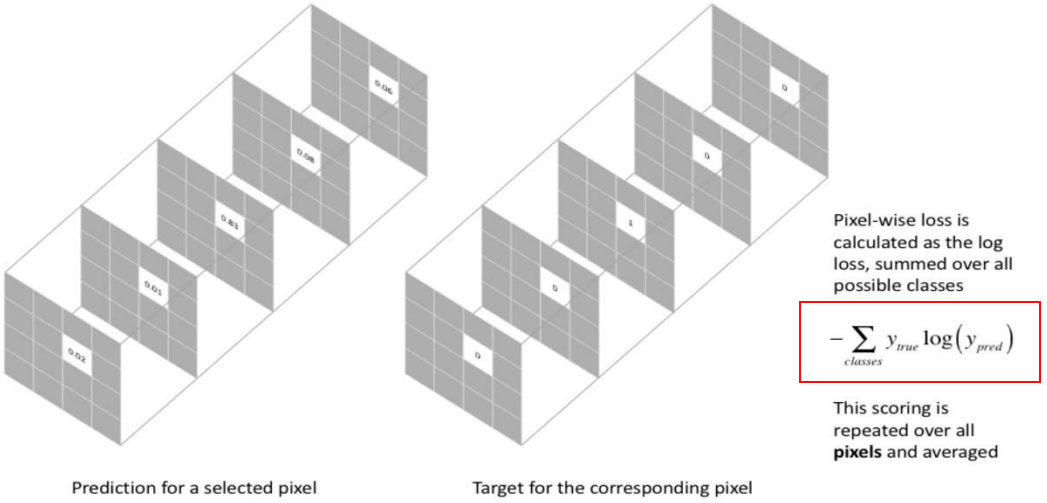
Corresponding pairs of downsampling and upsampling layers





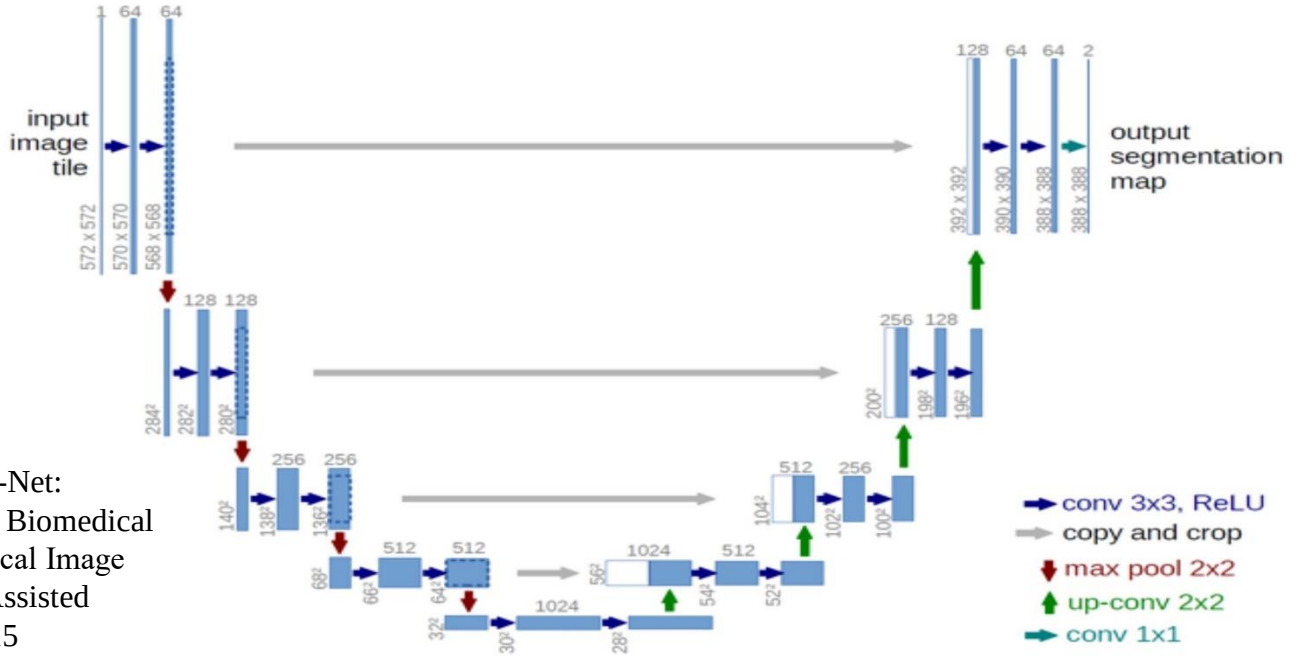
# Semantic Segmentation using FCN

- Creates a feature map, then downsample it and so on…
- All layers are Convolutional, no fully connected layer at the end.
- Output size of the last layer is the same size as the input image.
- # of channels is the number of classes to be clustered (car, road)
- Training is by cross-entropy loss for every pixel, to avoid training being dominated by the most prevalent class → weighting this loss for each output channel in order to counteract a class imbalance [J. Long, 2015].



# U-Net and Loss Function

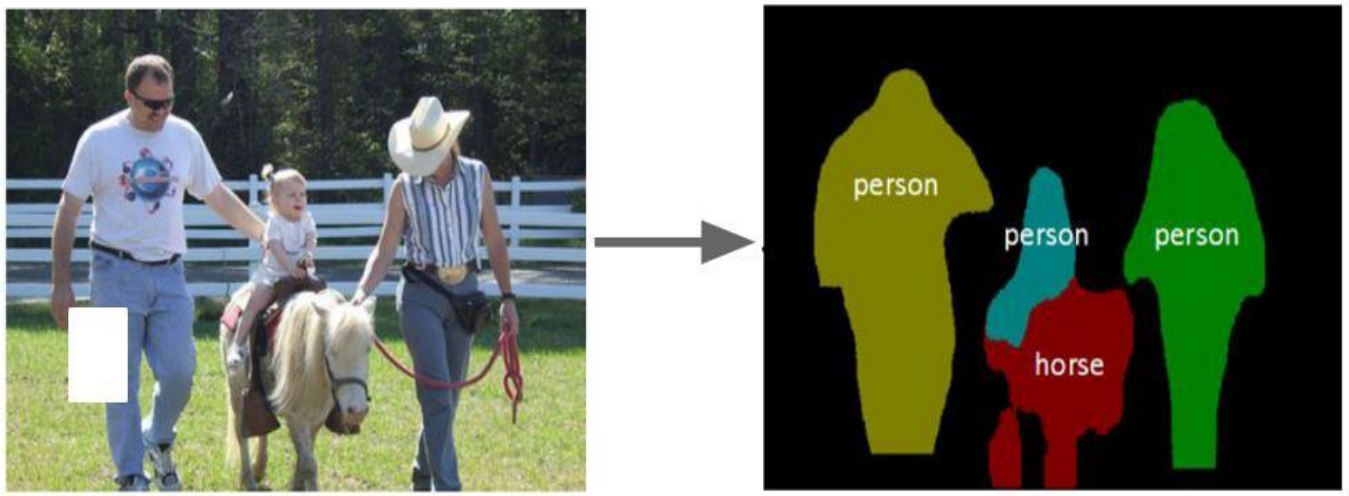
- U-Net uese a loss weighting scheme for each **pixel** such that there is a higher weight at the border of segmented objects.
- Segment cells in biomedical images in a *discontinuous* fashion such that individual cells may be easily identified within the binary segmentation map.



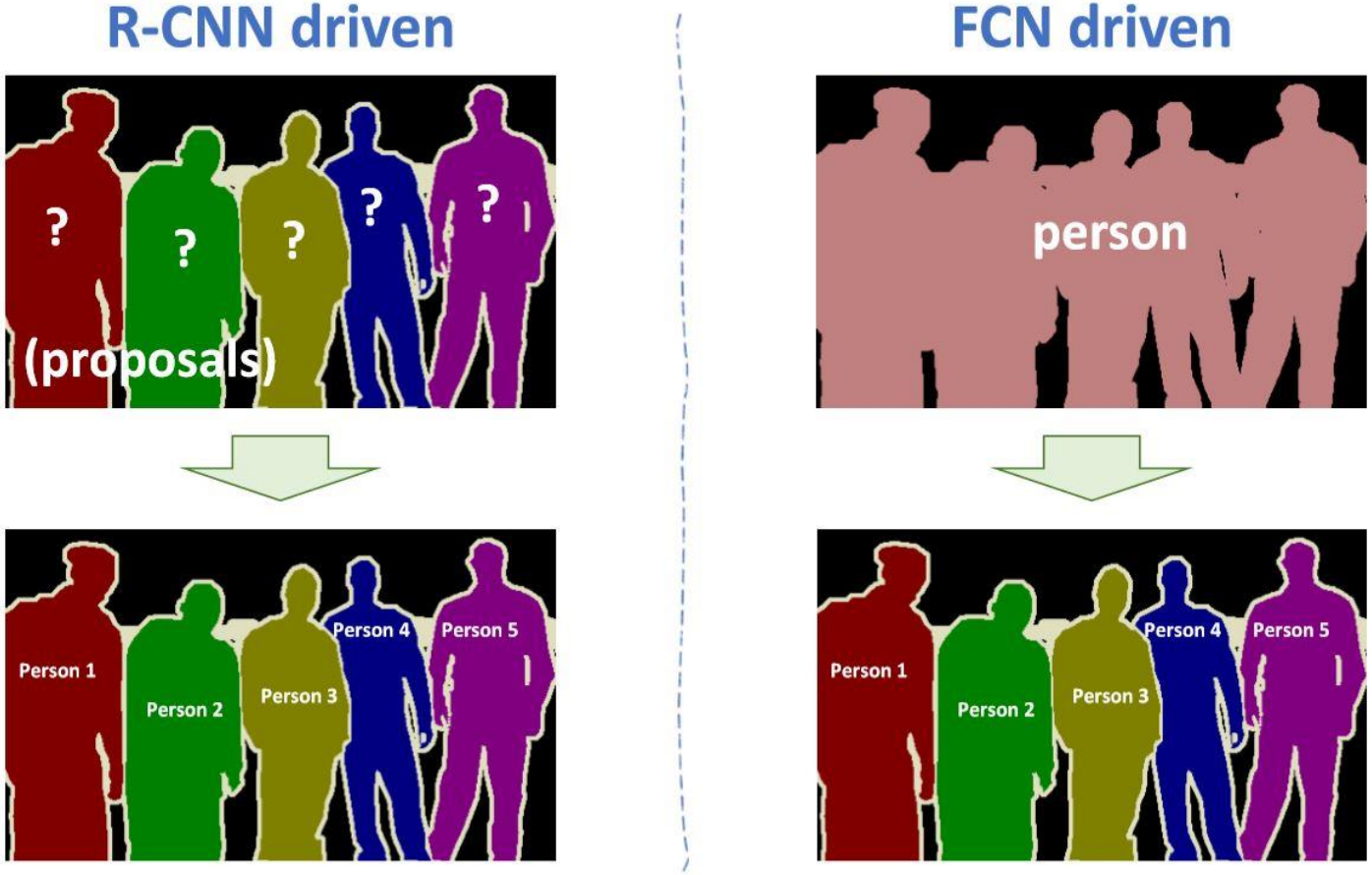

Olaf Ronneberger, et al., "U-Net: Convolutional Networks for Biomedical Image Segmentation," Medical Image Computing and Computer-Assisted Intervention (MICCAI), 2015

# Instance Segmentation

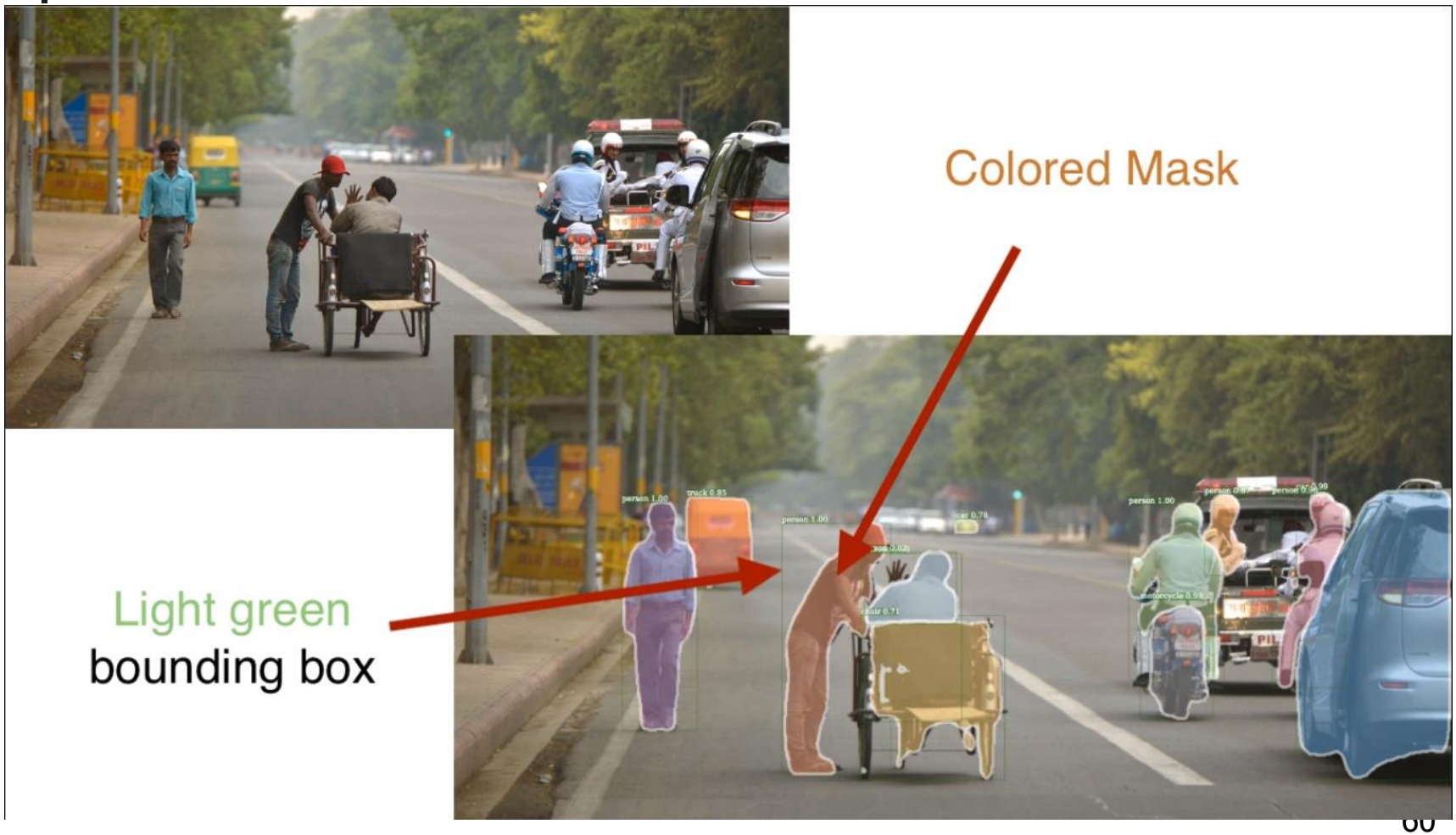
- Detect Instances, give category and label pixels
- Simultaneous detection and segmentation



# Approaches to Instance Segmentation

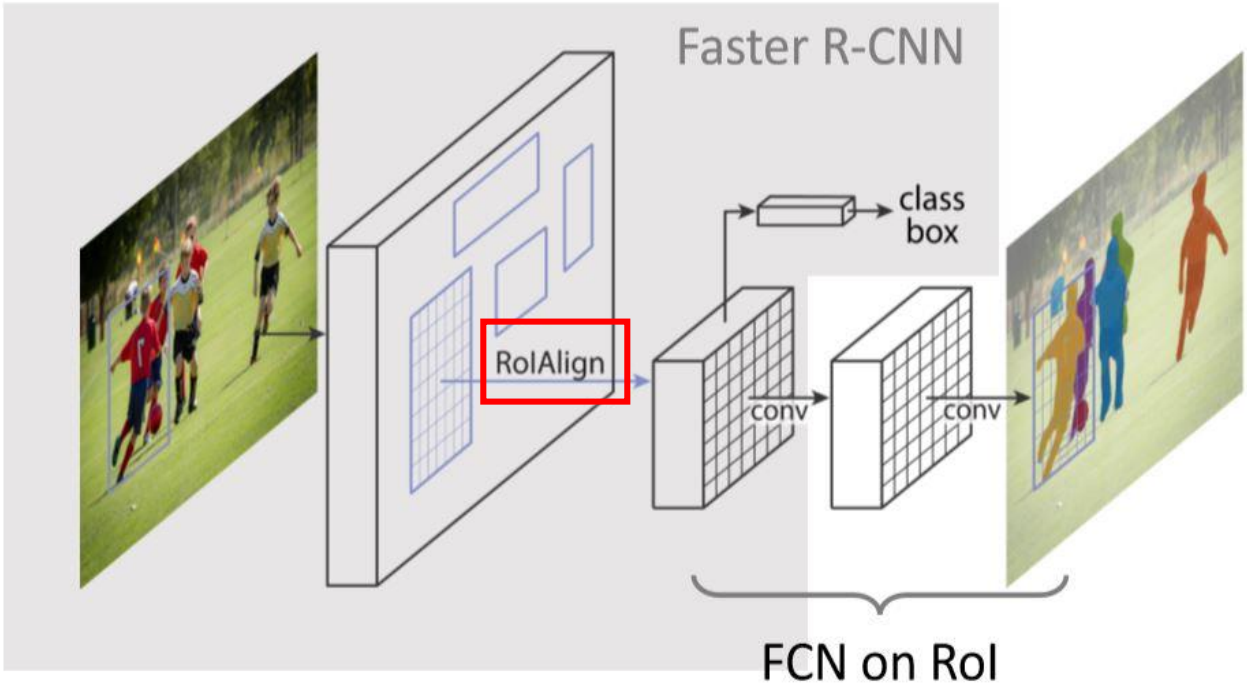


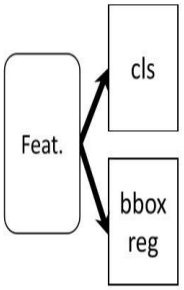
# Mask R-CNN



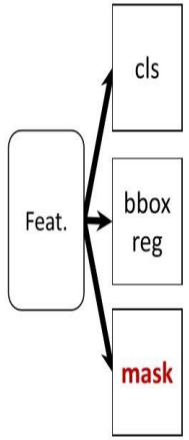
# Mask R-CNN

- Mask R-CNN = Faster R-CNN with FCN on RoIs



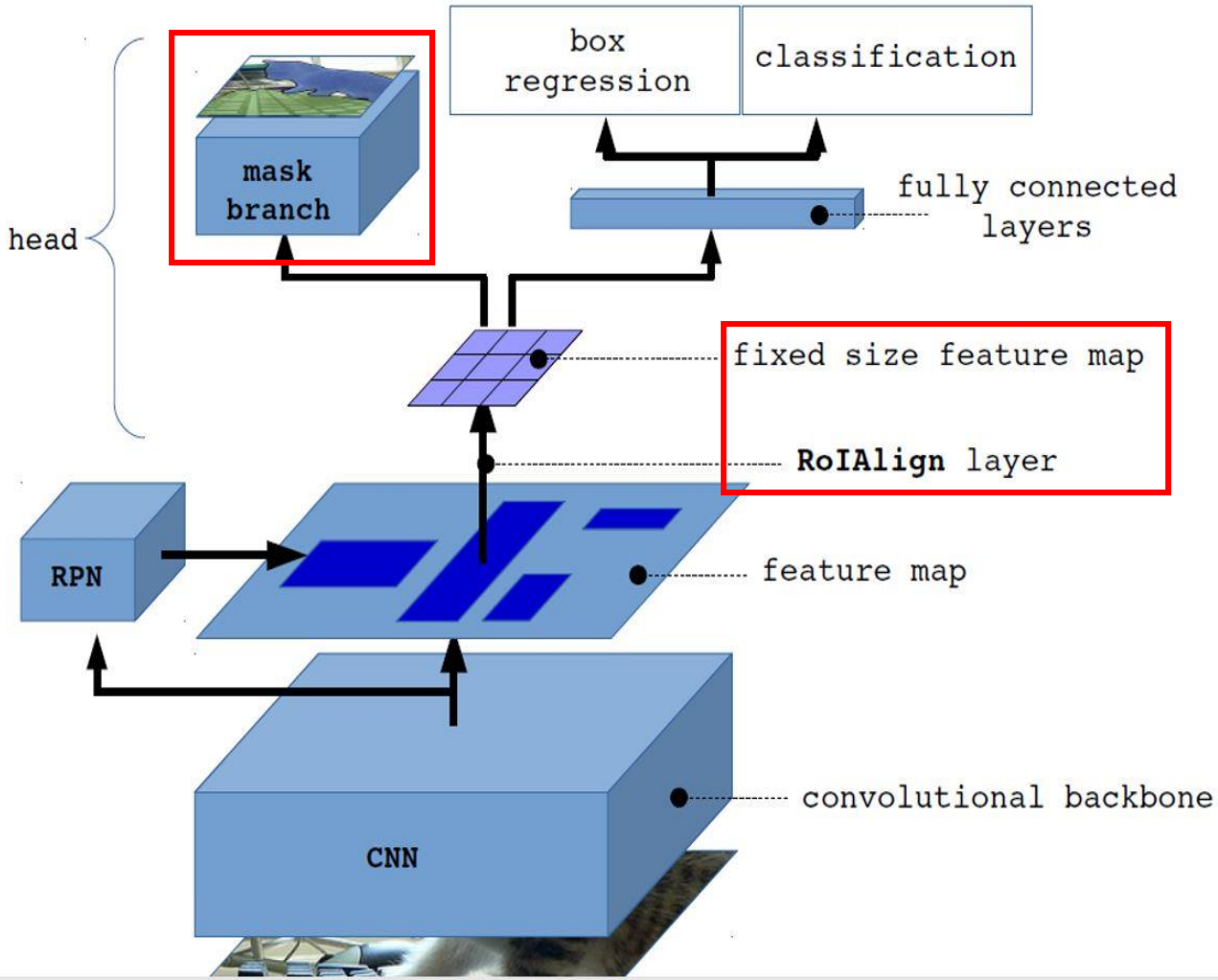

Fast/er R-CNN




Mask R-CNN

Kaiming He, et al. "Mask-RCNN," ICCV 2017

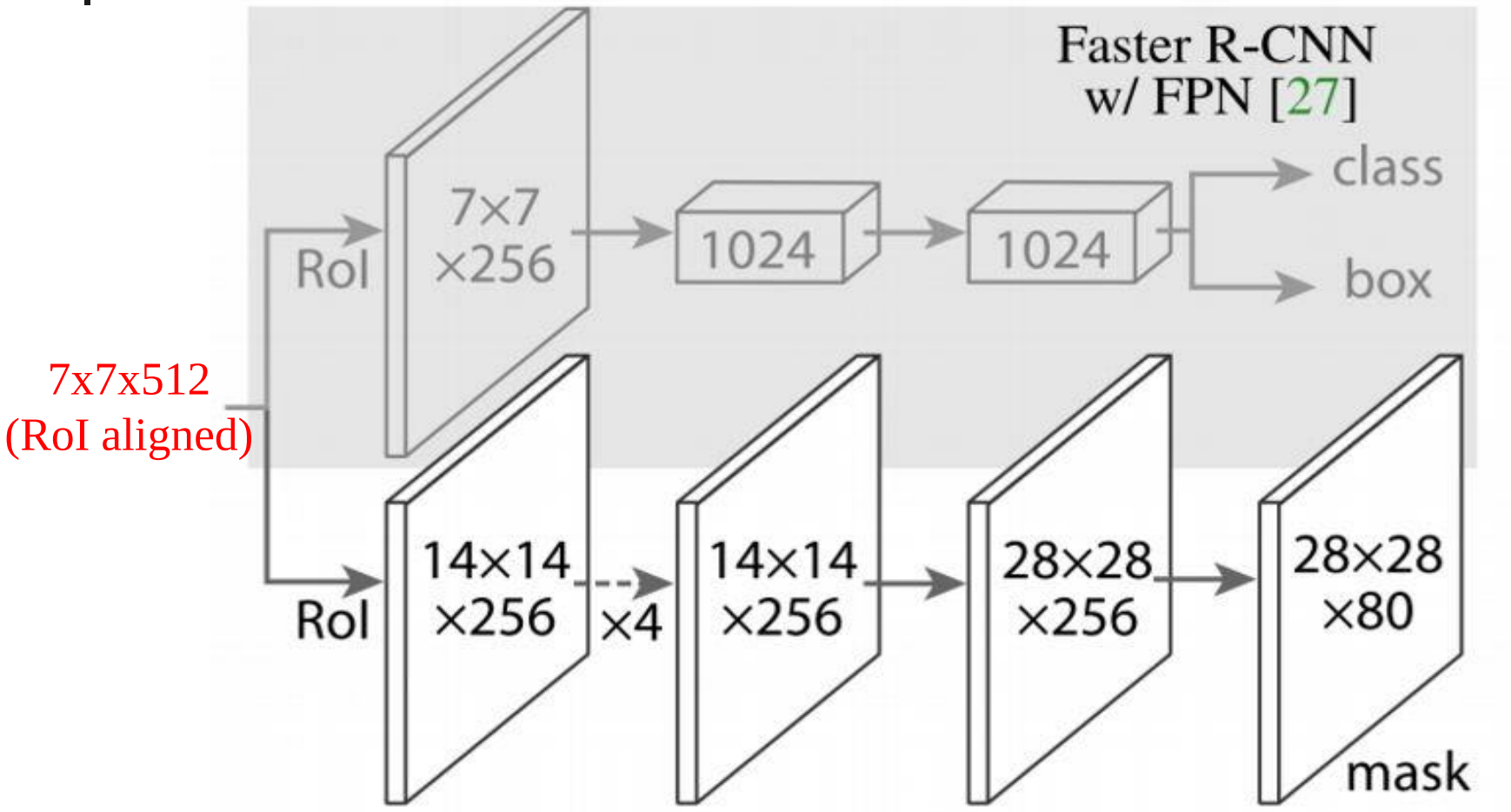
# Mask R-CNN


box regression
classification
mask branch
head
fully connected layers
fixed size feature map
RoIAlign layer
RPN
feature map
CNN
convolutional backbone

# Network Architecture

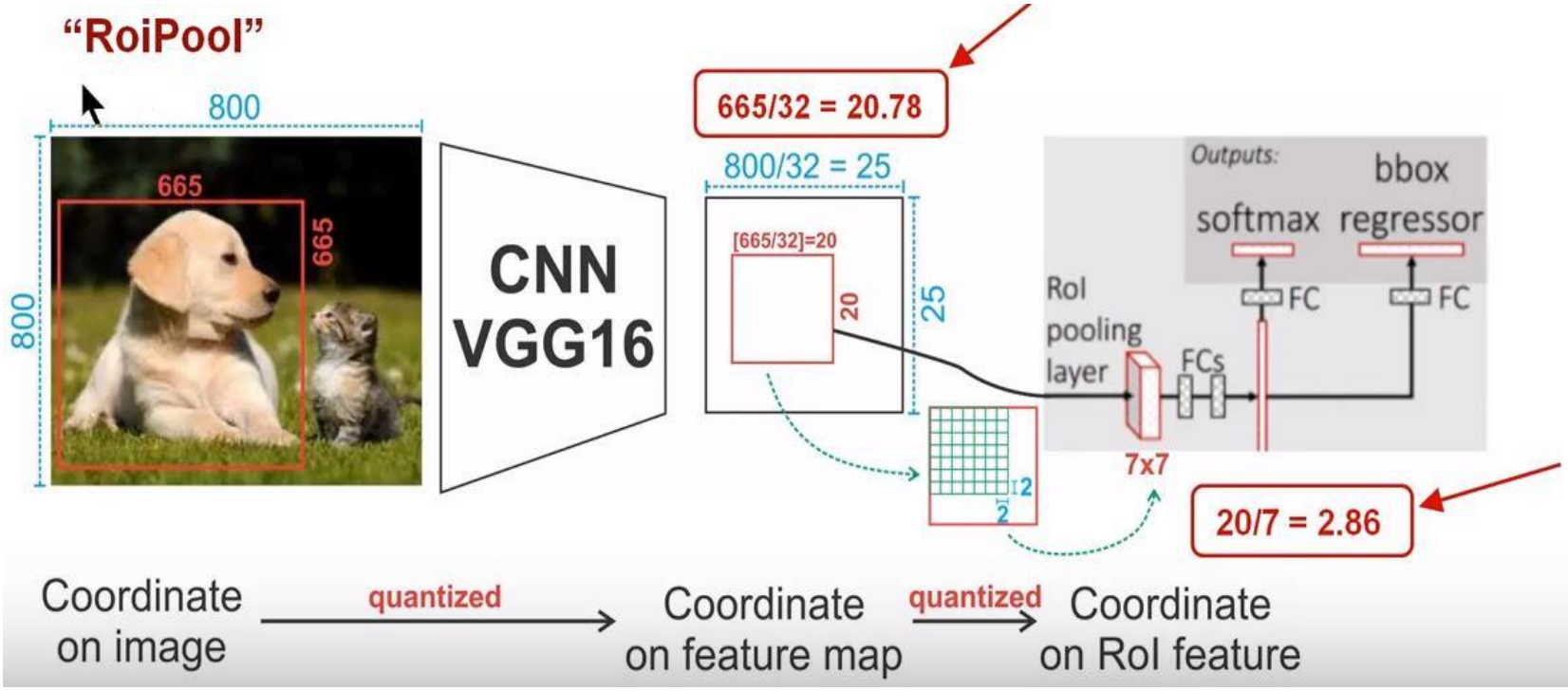
- Can be divided into two-parts:
  - Backbone architecture : Used for feature extraction
  - Network Head: comprises of object detection and segmentation parts
- Backbone architecture:
  - ResNet
  - ResNeXt: Depth 50 and 101 layers
  - Feature Pyramid Network (FPN)
- Network Head: Use almost the same architecture as Faster R-CNN but add convolution mask prediction branch

# FCN Mask Head

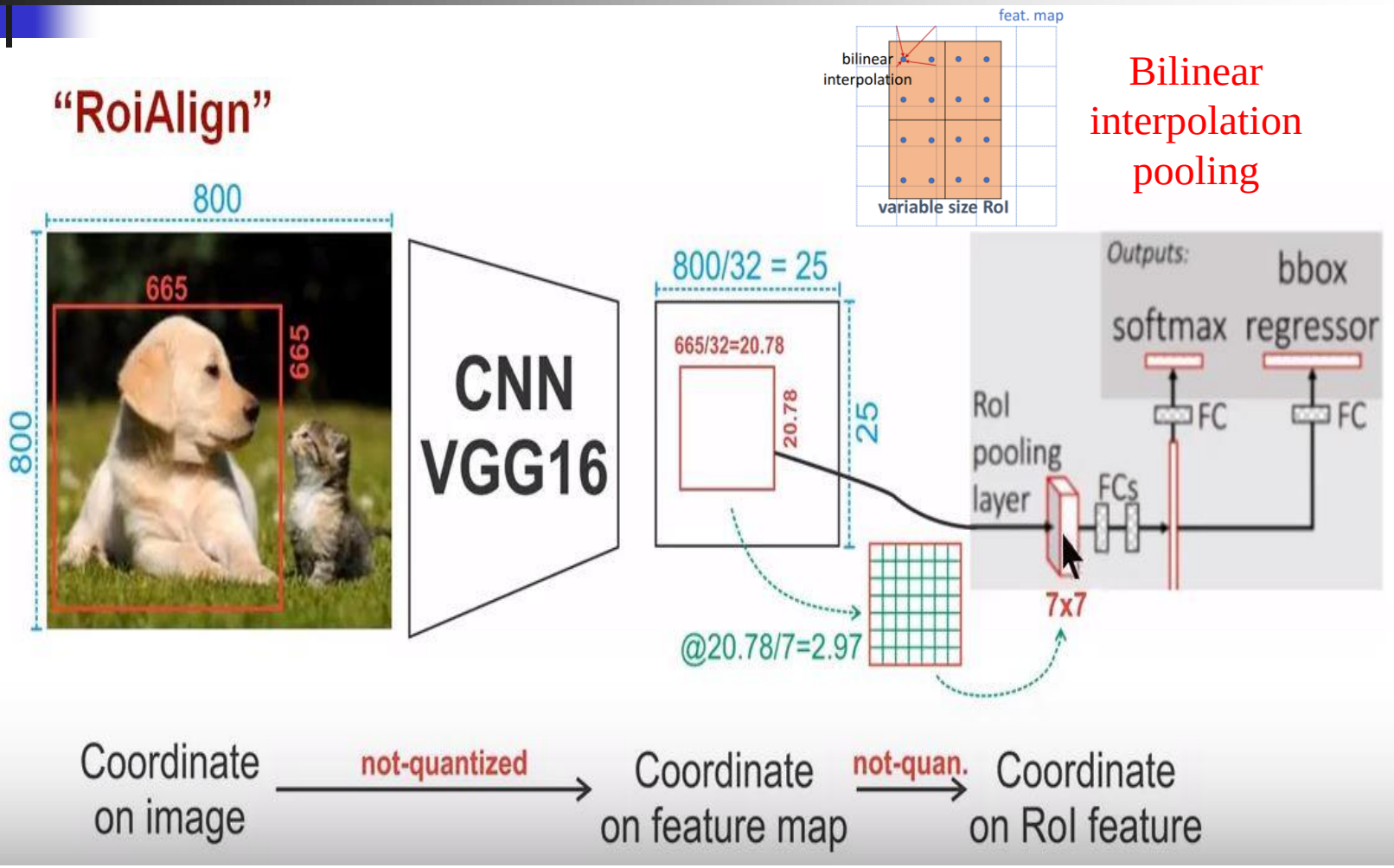


# RoI Pooling: Stride is Quantized

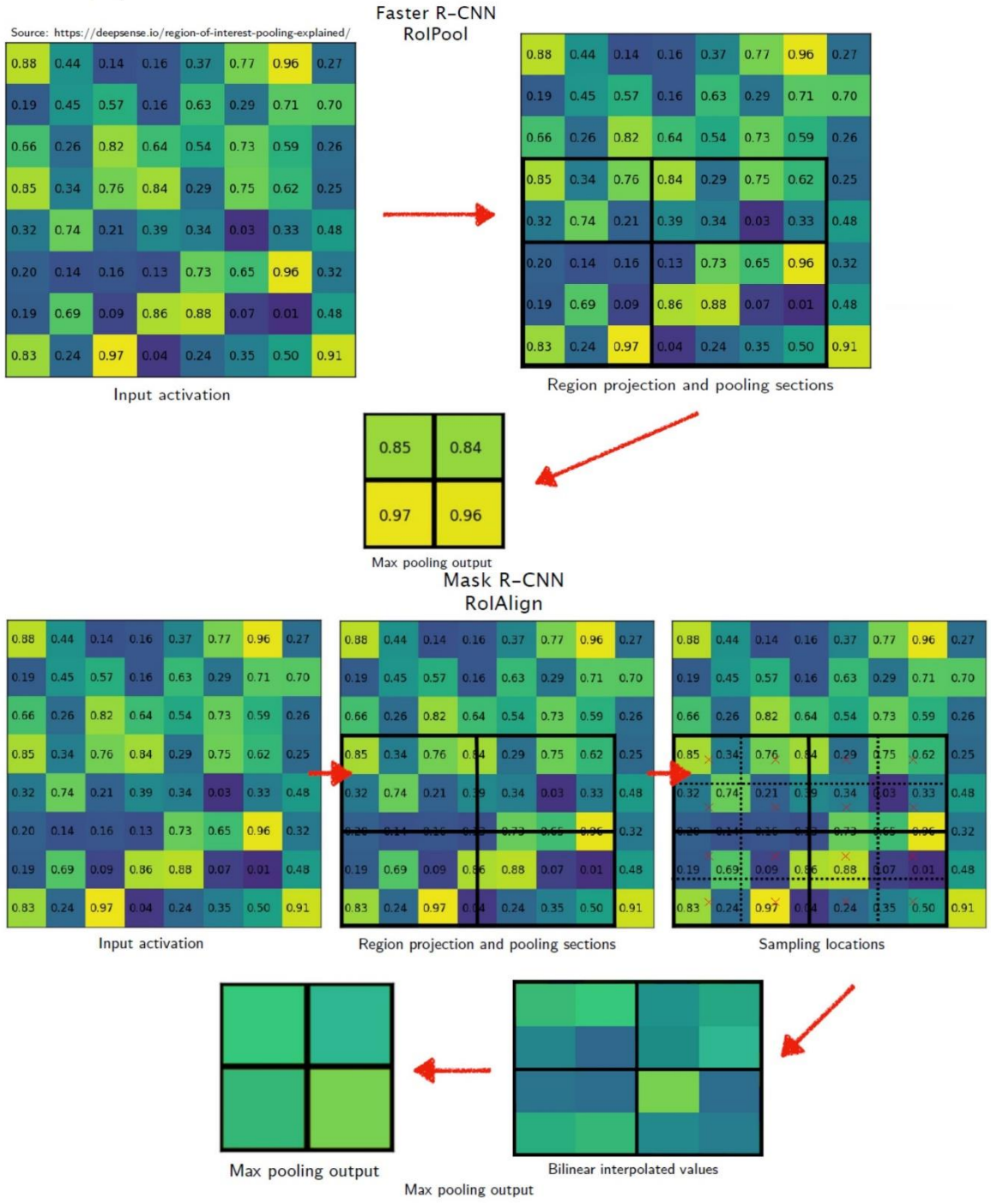
The ROI pooling layer divide the **h x w** ROI as **H x W** sub-windows; and then **max-pool** each sub-window to get H x W map to represent the ROI. (The **max-pool kernel is [h/H], [w/W]** respectively).



# RoIAlign: No Quantization



# RoI Pooling and RoI Align



# Loss Function

$$L = L_{cls} + L_{box} + L_{mask}$$

- Loss for classification and box regression is same as Faster R-CNN
- To each map a per-pixel sigmoid is applied
- The map loss is then defined as average binary cross entropy loss
- Mask loss is only defined for the ground truth class
- Decouples class prediction and mask generation
- Empirically better results and model becomes easier to train

$$J = -\frac{1}{N}\left(\sum_{i=1}^{N} \mathbf{y_i} \cdot log(\hat{\mathbf{y}}_\mathbf{i})\right)$$

# Mask R-CNN Performance

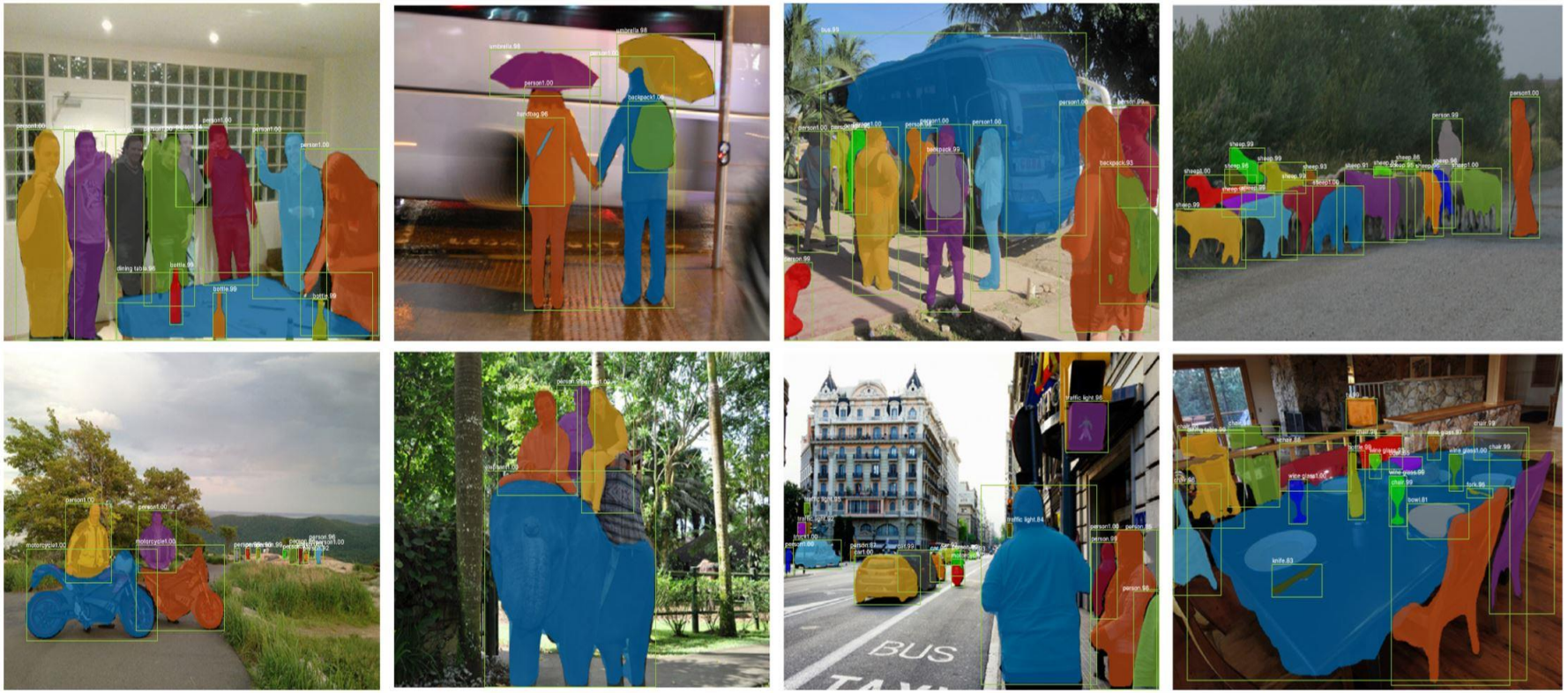



Figure 2. **Mask R-CNN** results on the COCO test set. These results are based on ResNet-101 [19], achieving a *mask* AP of 35.7 and running at 5 fps. Masks are shown in color, and bounding box, category, and confidences are also shown.