Computer Vision and Deep Learning

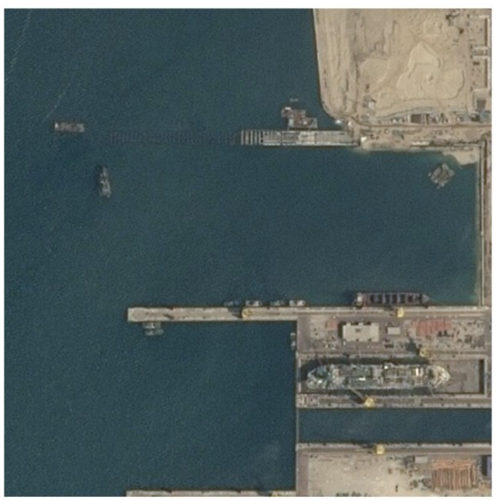
Lecture 9

Semantic segmentation metrics

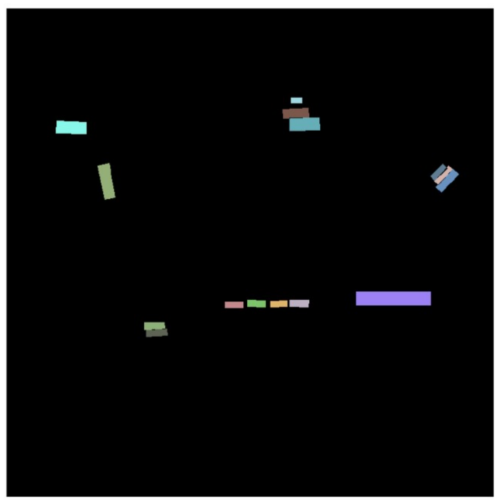
Pixel accuracy

- Number of pixels classified correctly by the network

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$



Image



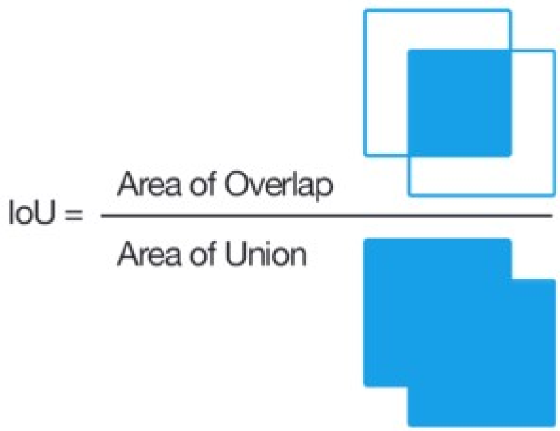
Ground truth

Prediction ☹

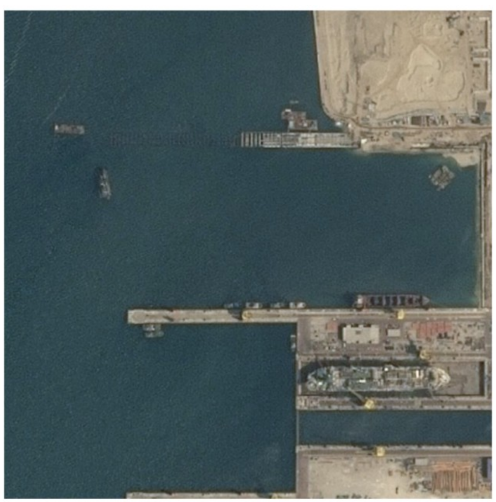
Accuracy 95%

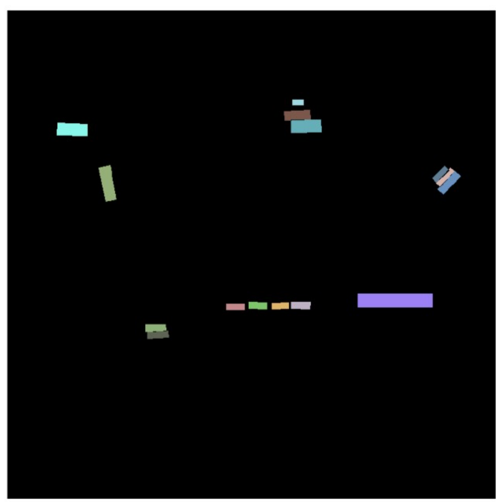
Semantic segmentation metrics

Intersection over Union



Mean IOU 47.5%



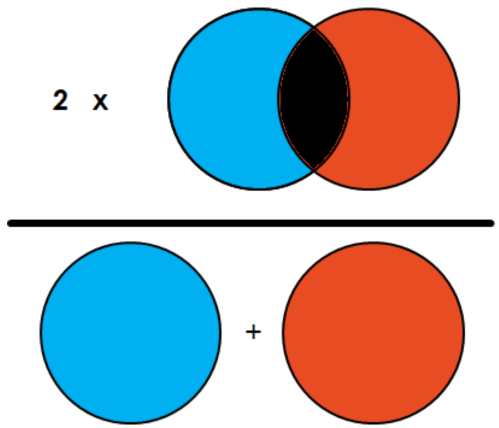


https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2

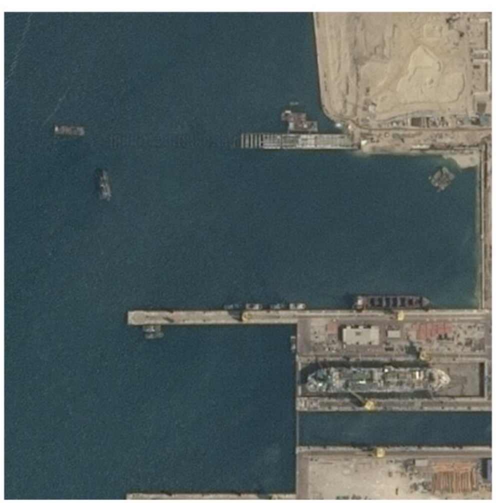
Prediction ☹

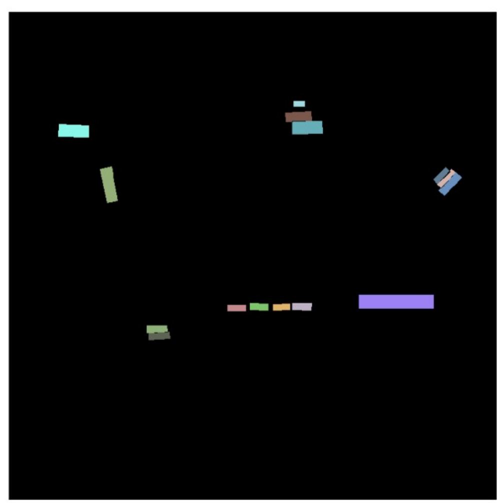
Semantic segmentation metrics

Dice Coefficient



Dice score 47.5%





https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2

Prediction ☹

Semantic segmentation

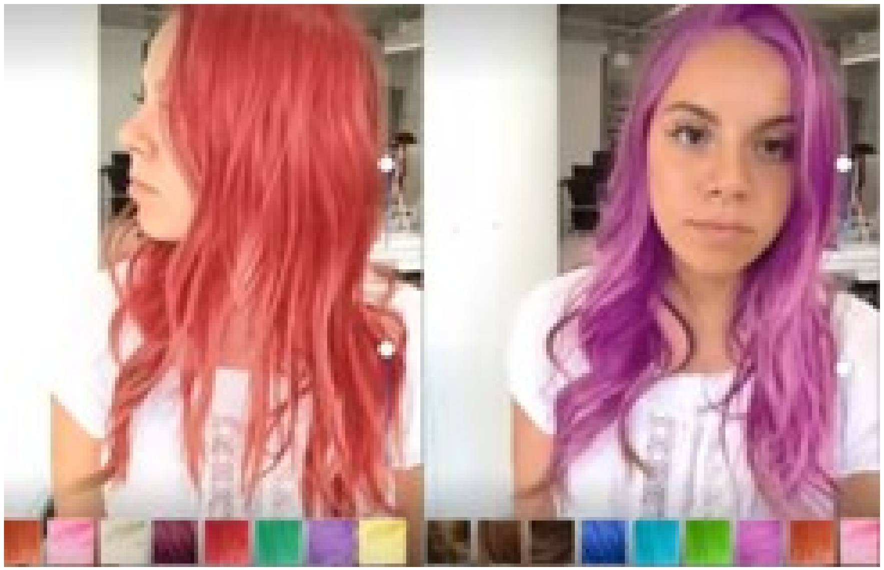
Examples

Autonomous driving

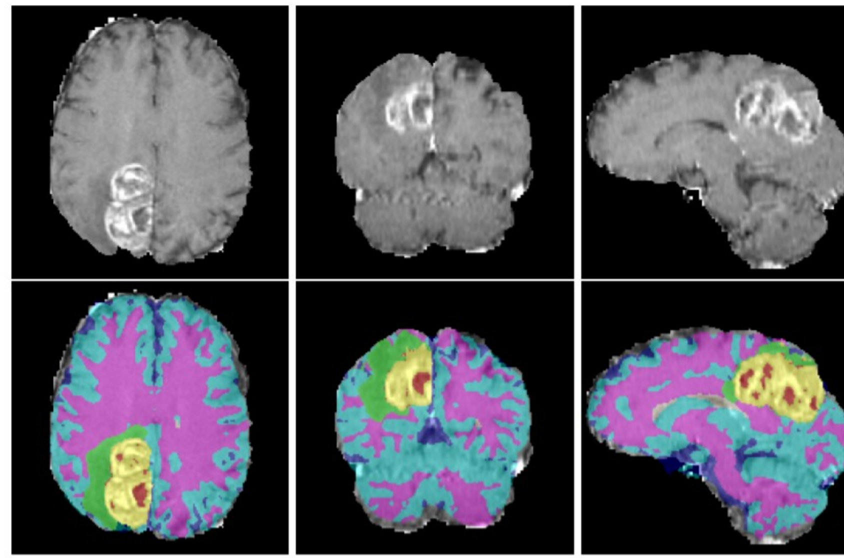
Sky	Building	Pole	Road	Sidewalk	Vegetation	Sign	Fence	Car	Pedestrian	Cyclist

Semantic segmentation

Examples



Real time hair colouring



Medical image segmentation

https://arxiv.org/pdf/1810.05732.pdf

https://news.developer.nvidia.com/3d-real-time-video-hair-coloration/

Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "**Fully convolutional networks for semantic segmentation**." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

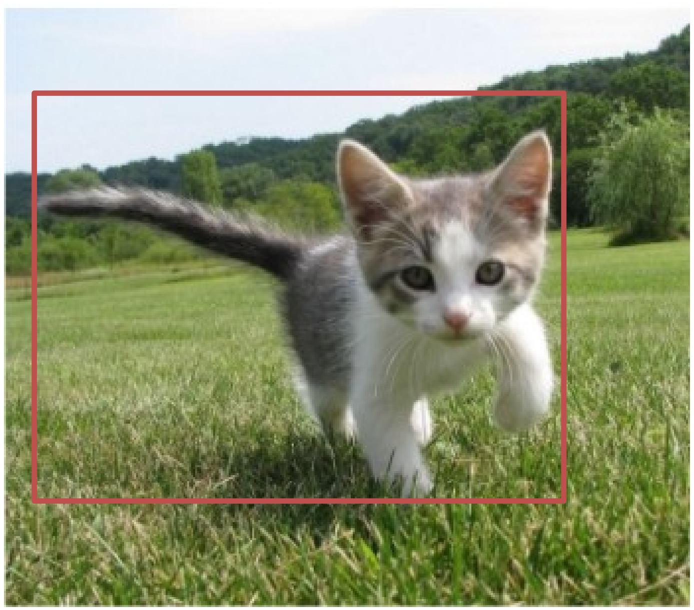
Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "**U-net: Convolutional networks for biomedical image segmentation.**" *International Conference on Medical image computing and computer-assisted intervention*. Springer, Cham, 2015.

Jégou, Simon, et al. "**The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation.**" *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2017.

https://beyondminds.ai/a-simple-guide-to-semantic-segmentation/

Object localization

- What object is this image and where is this object located in the image?



CAT

Object localization

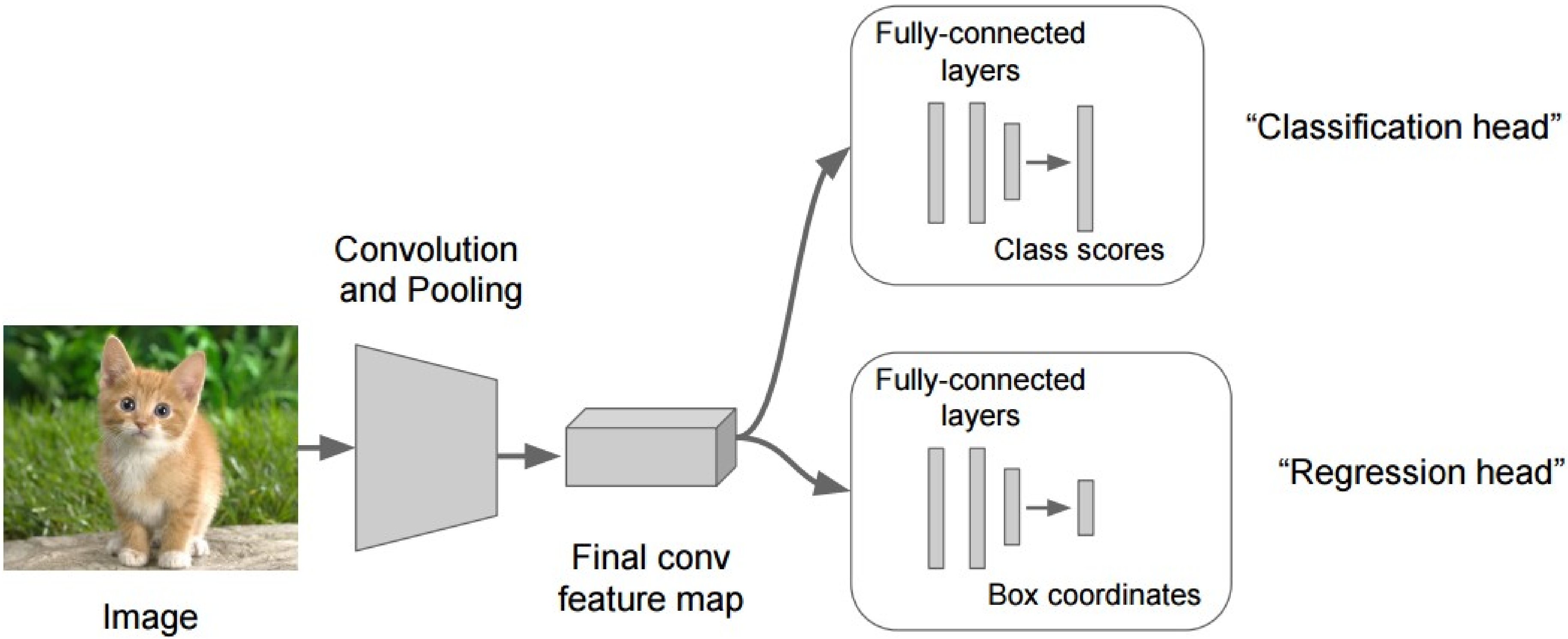
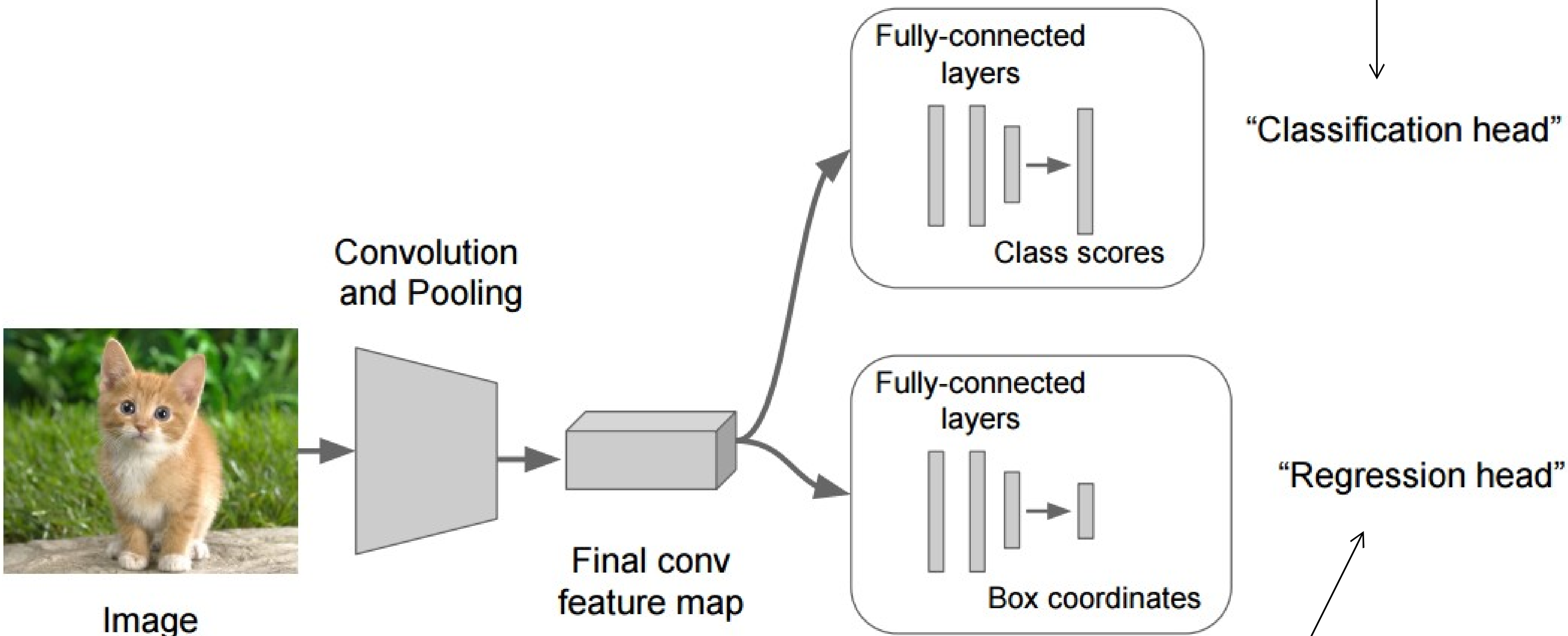


Image source: https://leonardoaraujosantos.gitbook.io/artificial-inteligence/machine_learning/deep_learning/object_localization_and_detection

Object localization

C class scores
Softmax loss L_s



BBOX coordinates
4 outputs (x, y, w , h)
L2 loss L_2

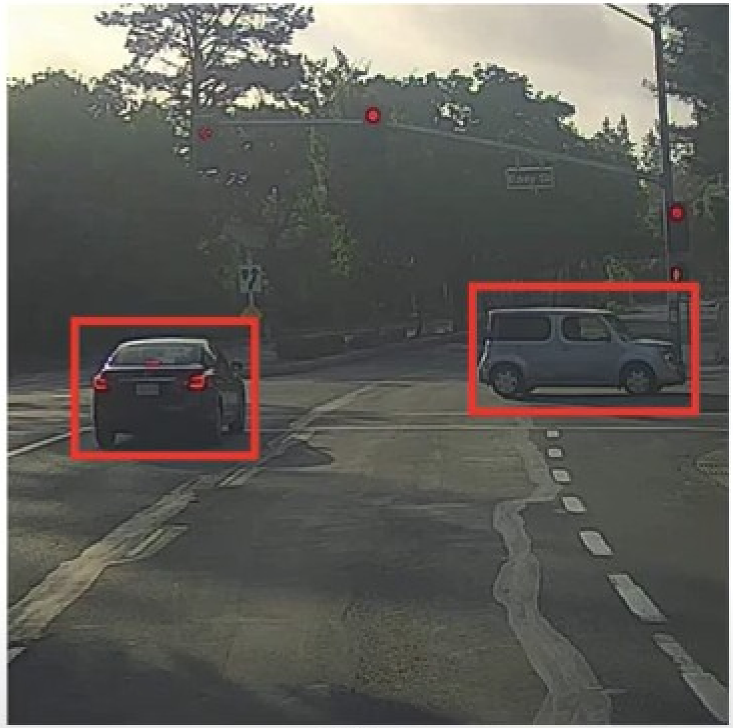
Image source: https://leonardoaraujosantos.gitbook.io/artificial-inteligence/machine_learning/deep_learning/object_localization_and_detection

Object detection

- Determine the **class (label)** and the **position** of EACH object in the input image
- We can’t use the same approach as for localization
  - Each image would require a different number of outputs
- Sliding window approach?

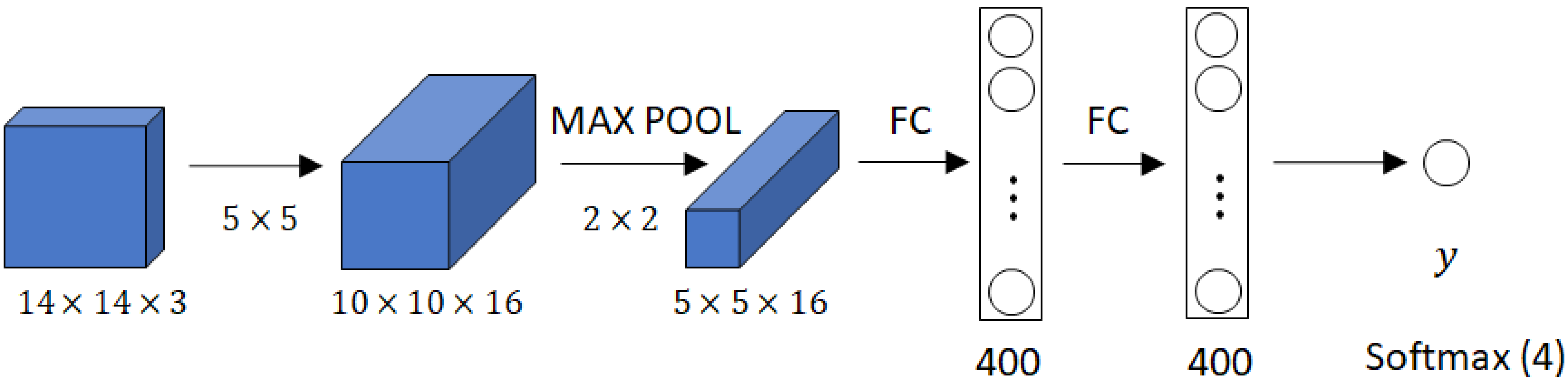
Object detection without proposals

100



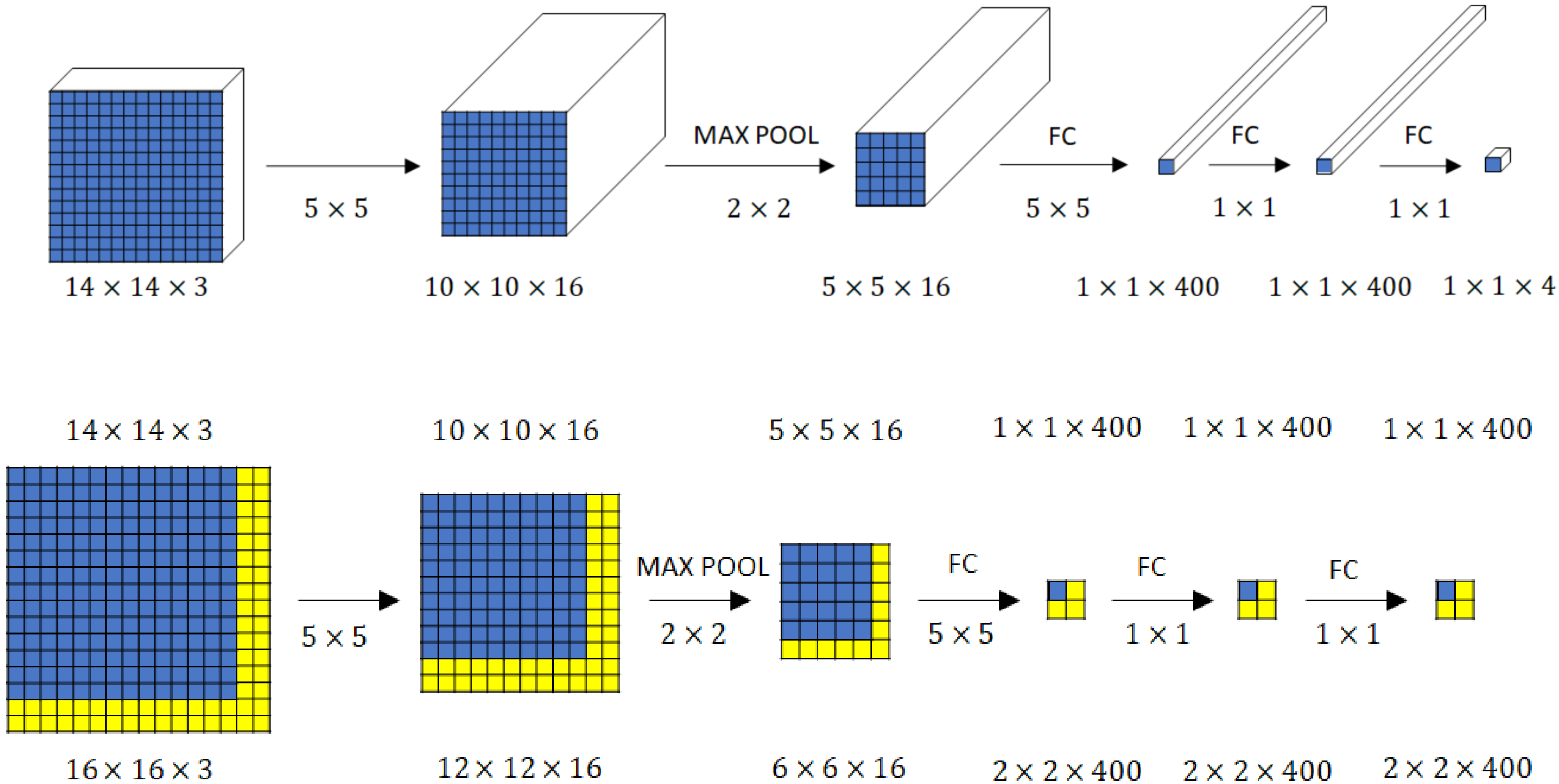
100

Convolutional implementation of sliding windows



https://www.youtube.com/watch?v=XdsmlBGOK-k
Example credit Andrew Ng

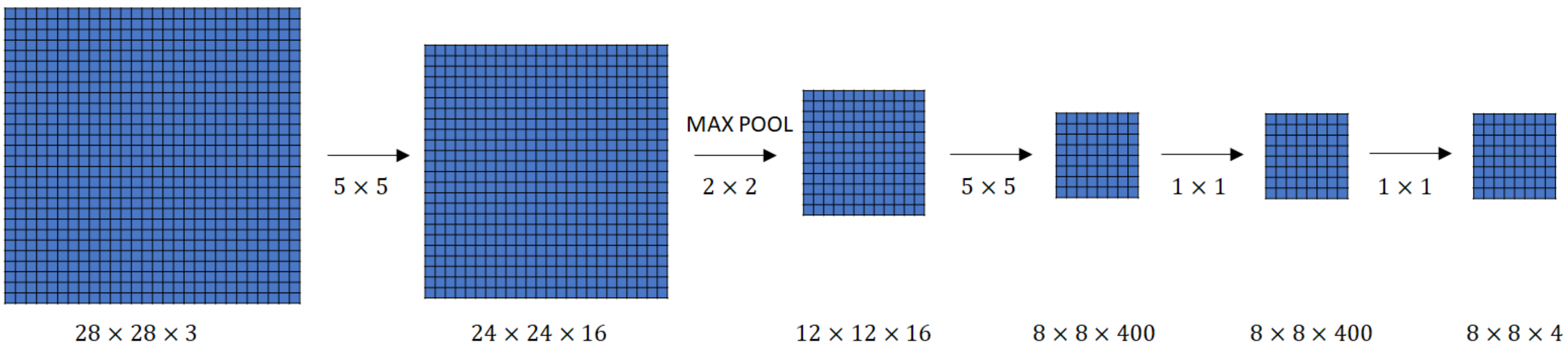
Convolutional implementation of sliding windows



https://www.youtube.com/watch?v=XdsmlBGOK-k

Example credit Andrew Ng

Convolutional implementation of sliding windows



https://www.youtube.com/watch?v=XdsmlBGOK-k
Example credit Andrew Ng

YOLO

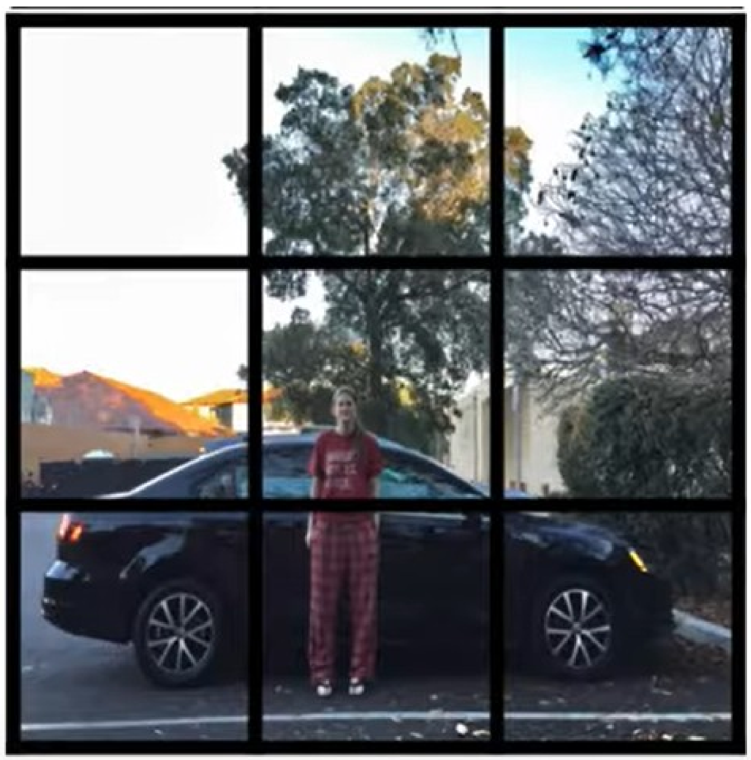
You only look once

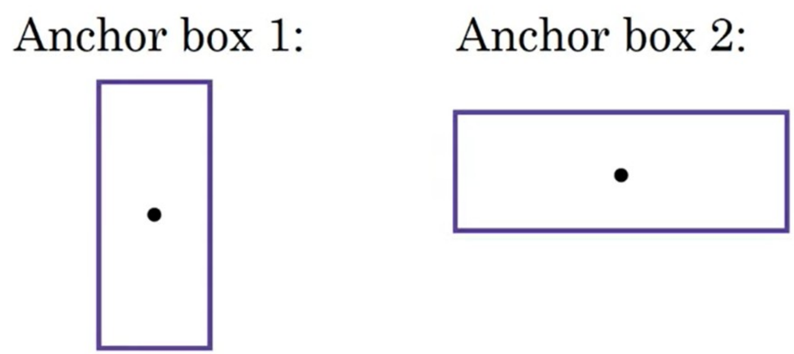
- Detection is modelled as a regression problem
  - image is divided into an $S \times S$ grid
  - for each grid cell predict:
    - B bounding boxes
    - confidence for those boxes,
    - C class probas

YOLO

Anchor boxes

- Allows the object detector to specialize better (detect “thinner” or “wider” objects)





https://www.youtube.com/watch?v=RTlwl2bv0Tg

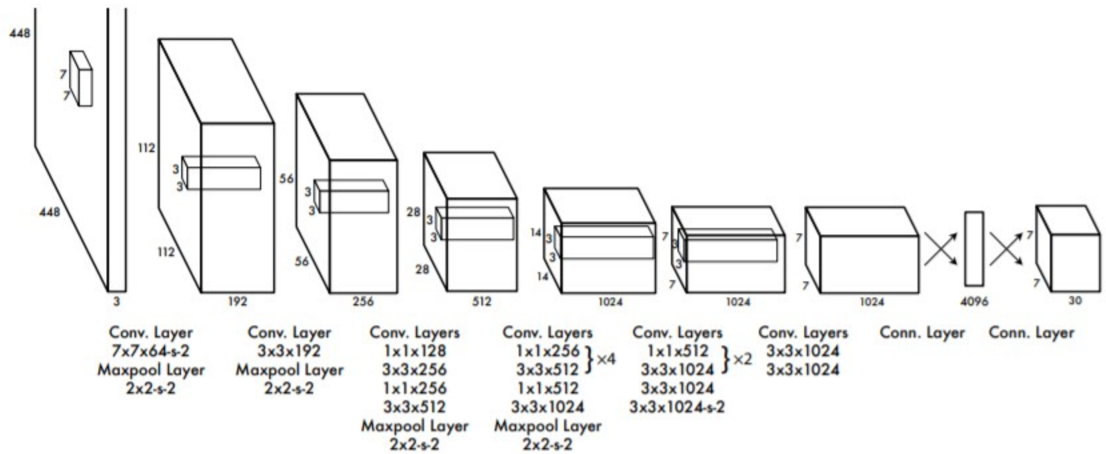


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

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

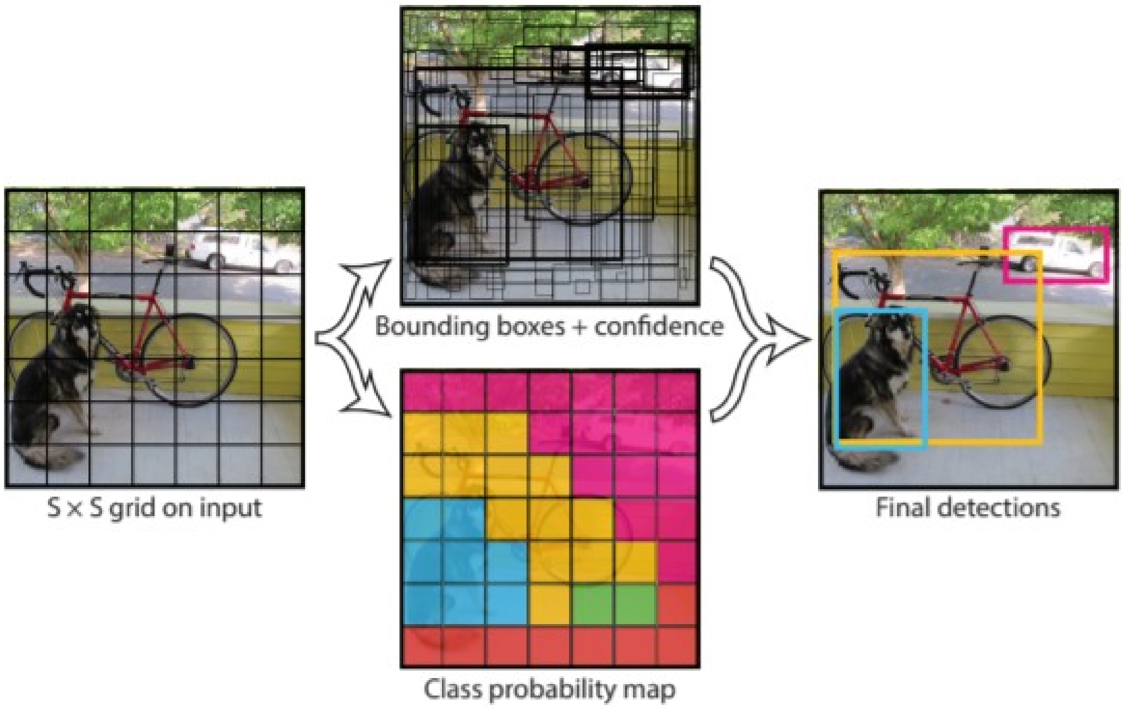
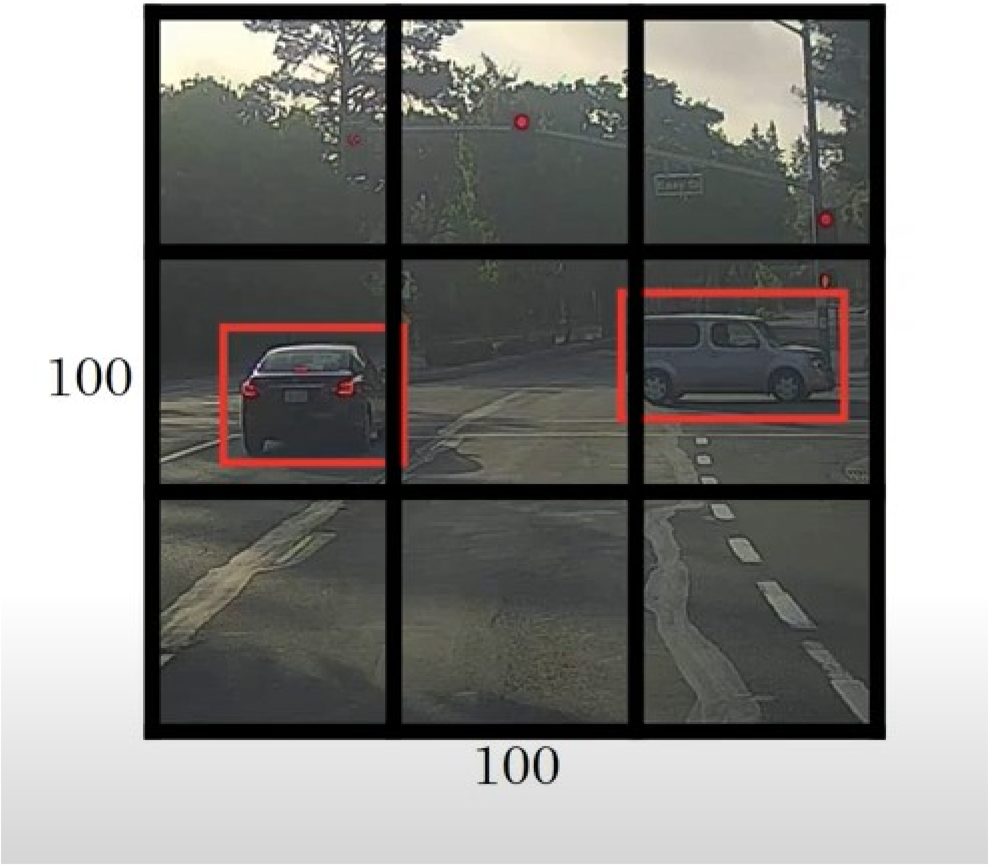


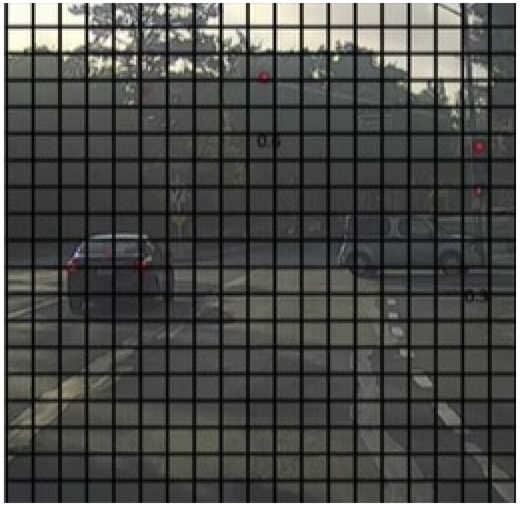
Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

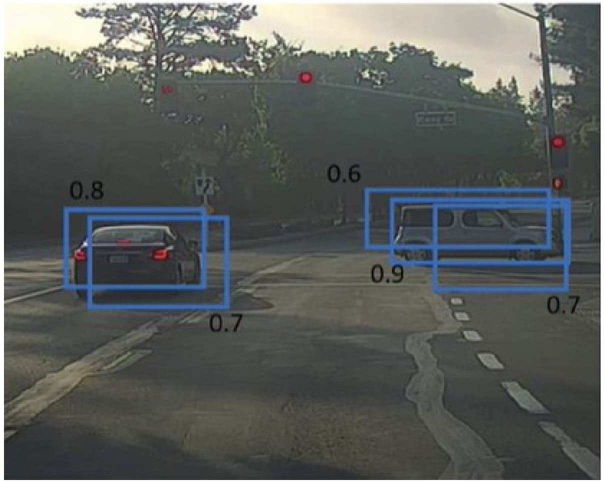

100
100

YOLO

Non maximum suppression

- Discard all predictions with proba <= *threshold*
- **while** there are any remaining boxes
  - Select the box with the largest proba (*maxBB*) as a prediction
  - Discard boxes with IoU >= 0.5 with *maxBB*





https://www.youtube.com/watch?v=VAo84c1hQX8

YOLO

Loss function

This is the loss function for yolo v1 paper.

Here each bounding box predicts an objectness score and a 4 coordinates, but the class predictions are per cell. Hence this:

$$
\begin{gathered}
\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)
\end{gathered}
$$

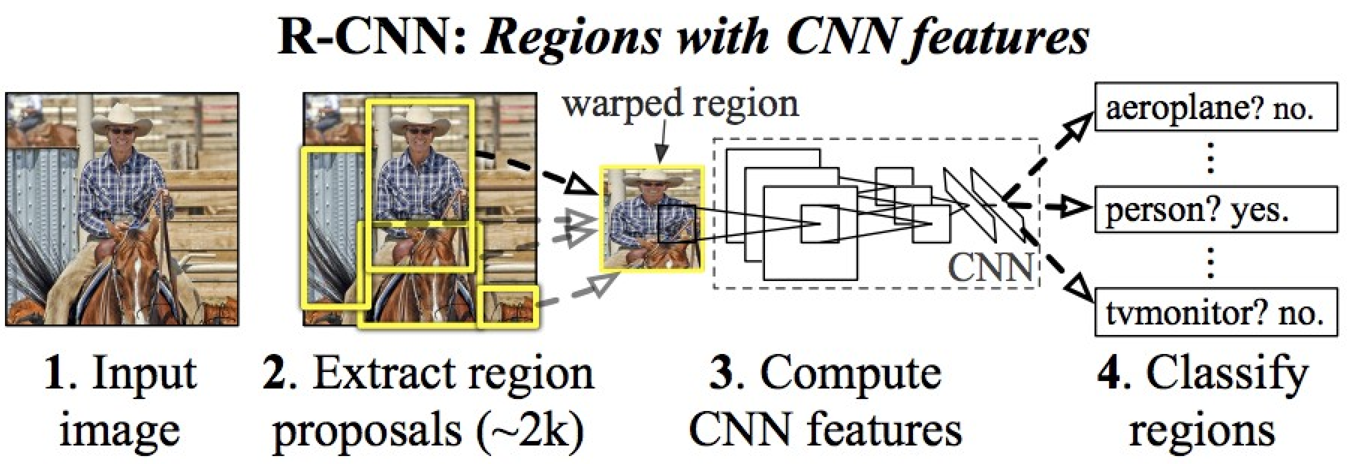
where $\mathbb{1}_i^{\text{obj}}$ denotes if object appears in cell i and $\mathbb{1}_{ij}^{\text{obj}}$ denotes that the jth bounding box predictor in cell i is "responsible" for that prediction.

Proposal based object detection

R-CNN

Region-based Convolutional Network, 2014

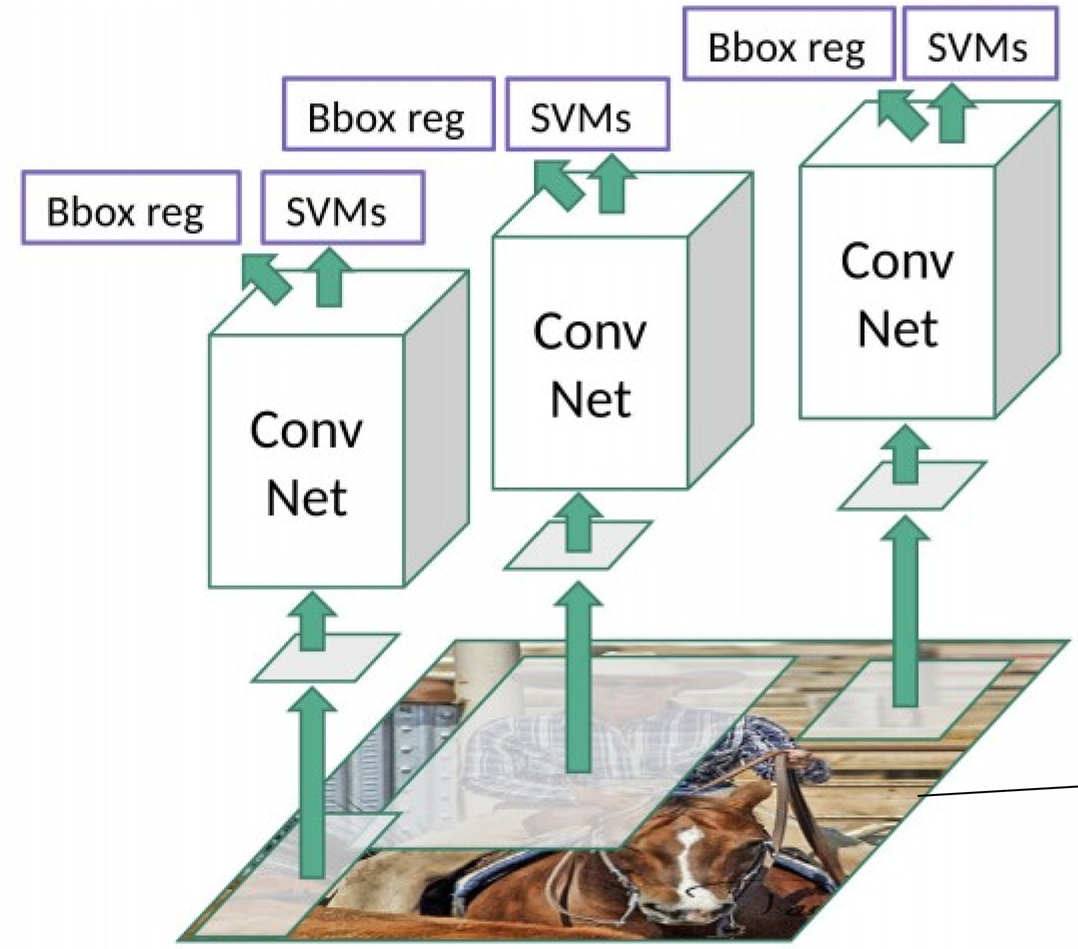
- Idea:
  - Use an algorithm (or network) to find region of interests (ROIs) that are likely to contain an object
  - **Localize** (label + bounding box) localize the object in each in region



https://arxiv.org/pdf/1311.2524.pdf

R-CNN

Region-based Convolutional Network, 2014

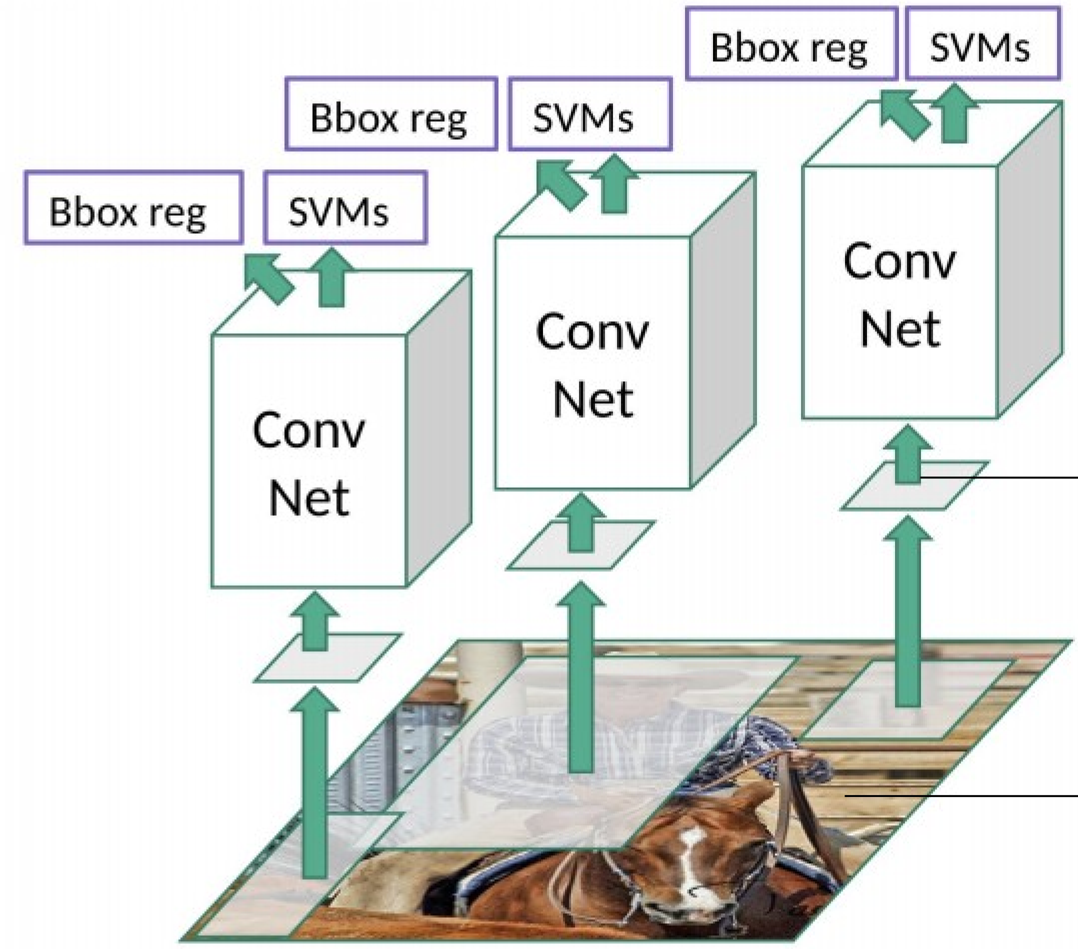


Selective search to extract 2000 regions. **NO learning here**

Image source: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

R-CNN

Region-based Convolutional Network, 2014

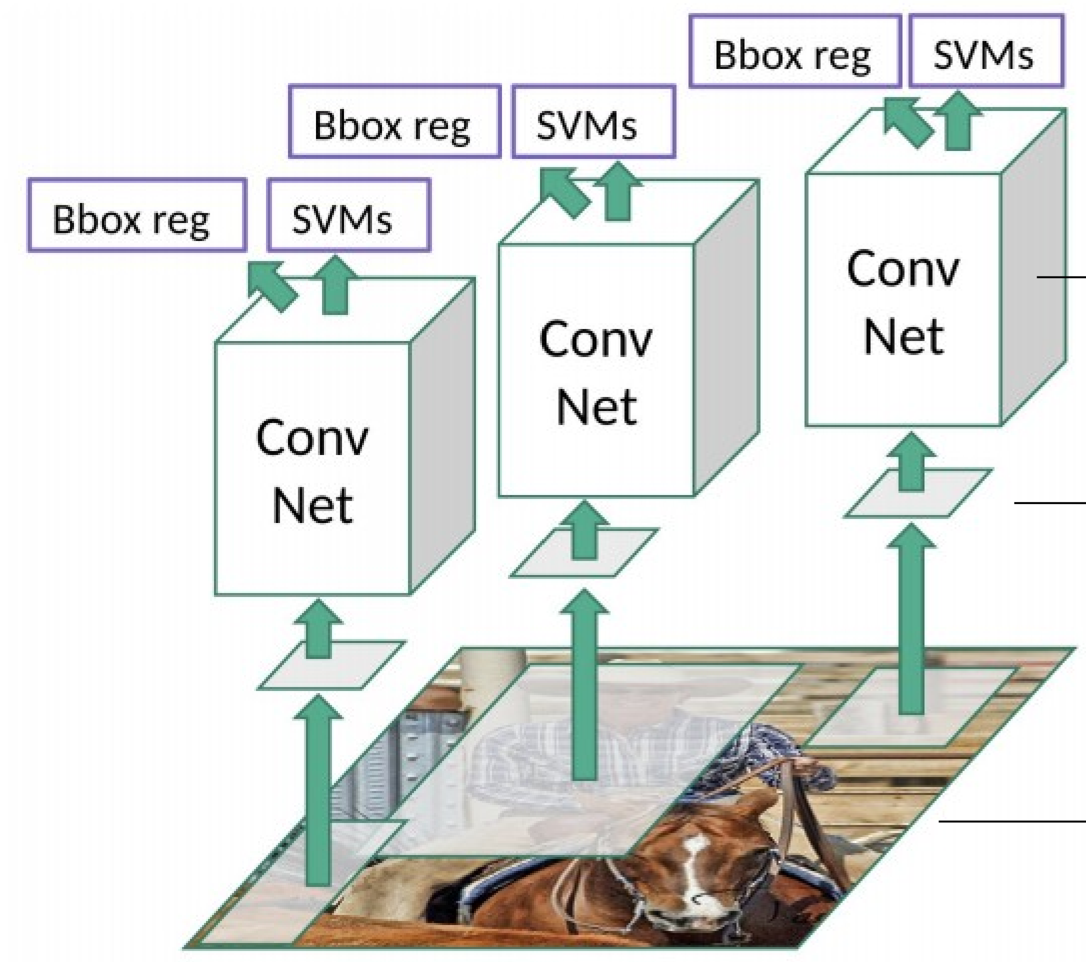


Warp each image region into a square of 227x227 ROI (will be fed to AlexNet) + 16 pix

Selective search to extract 2000 regions. **NO learning here**

Image source: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

R-CNN
Region-based Convolutional Network, 2014



AlexNet – feature extractor

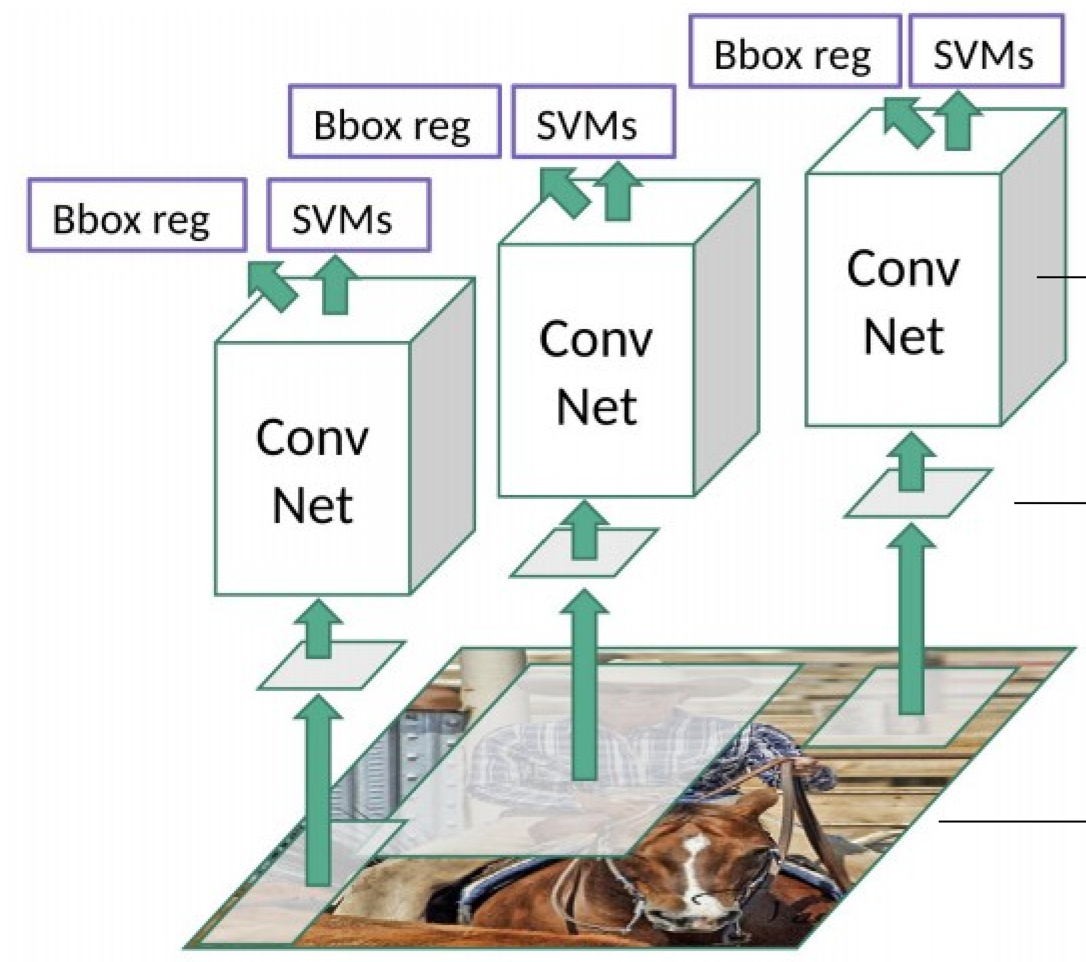
Warp each image region into a square of 227x227 ROI (will be fed to AlexNet)

Selective search to extract 2000 regions. **NO learning here**

Image source: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

R-CNN

Region-based Convolutional Network, 2014



SVMs to classify the presence + label of an object in region
Regression **to fine tune** the bounding boxes

AlexNet – feature extractor

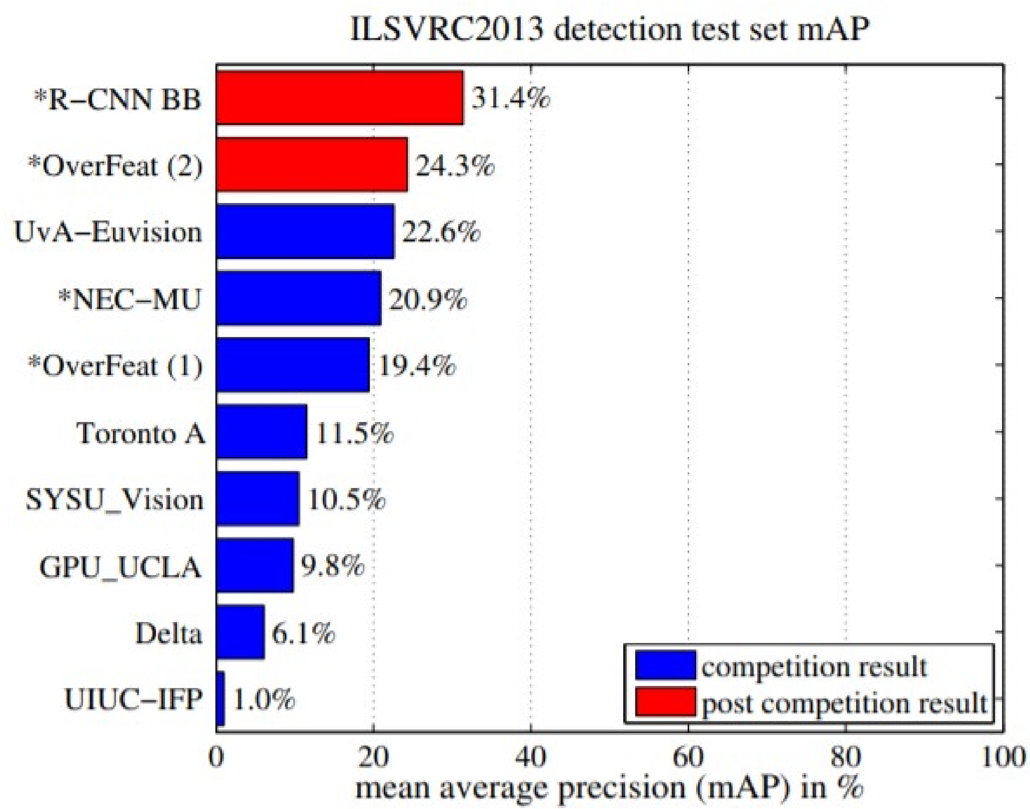
Warp each image region into a square of 227x227 ROI (will be fed to AlexNet)

Selective search to extract 2000 regions. **NO learning here**

Image source: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

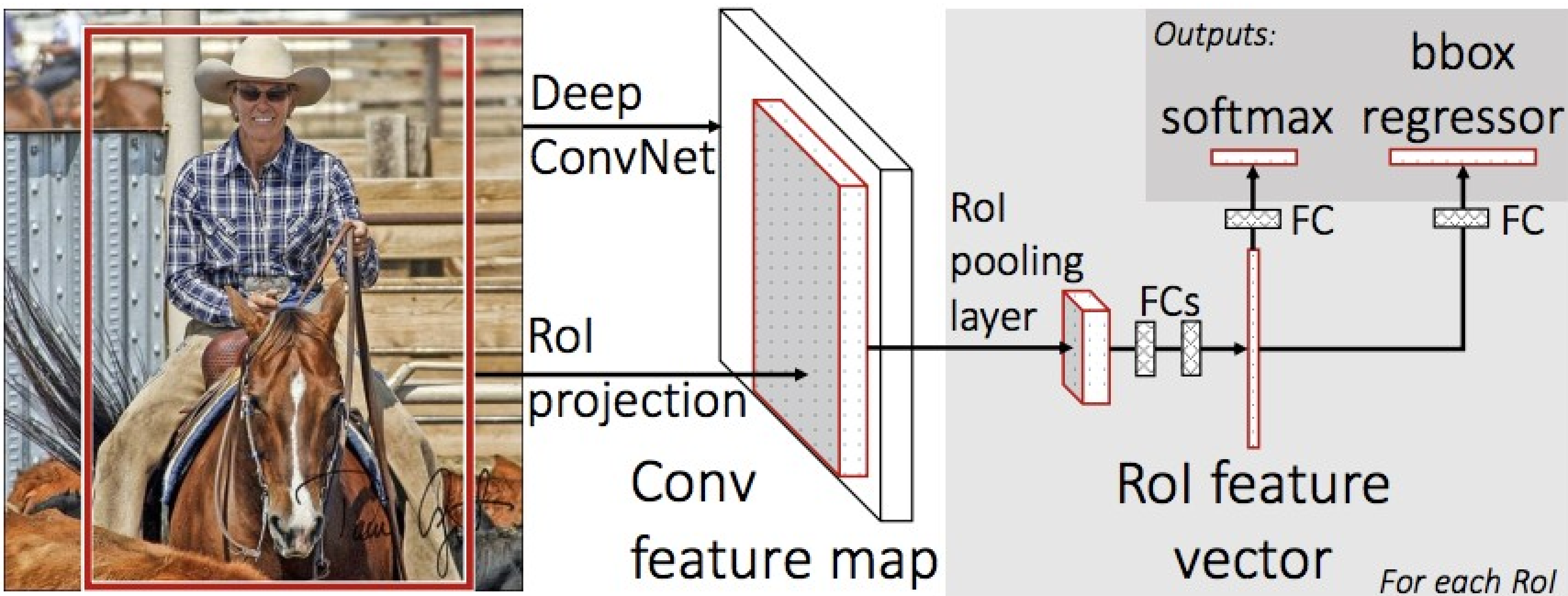
R-CNN

- <u>Running time:</u> "13s/image on a GPU or 53s/image on a CPU"

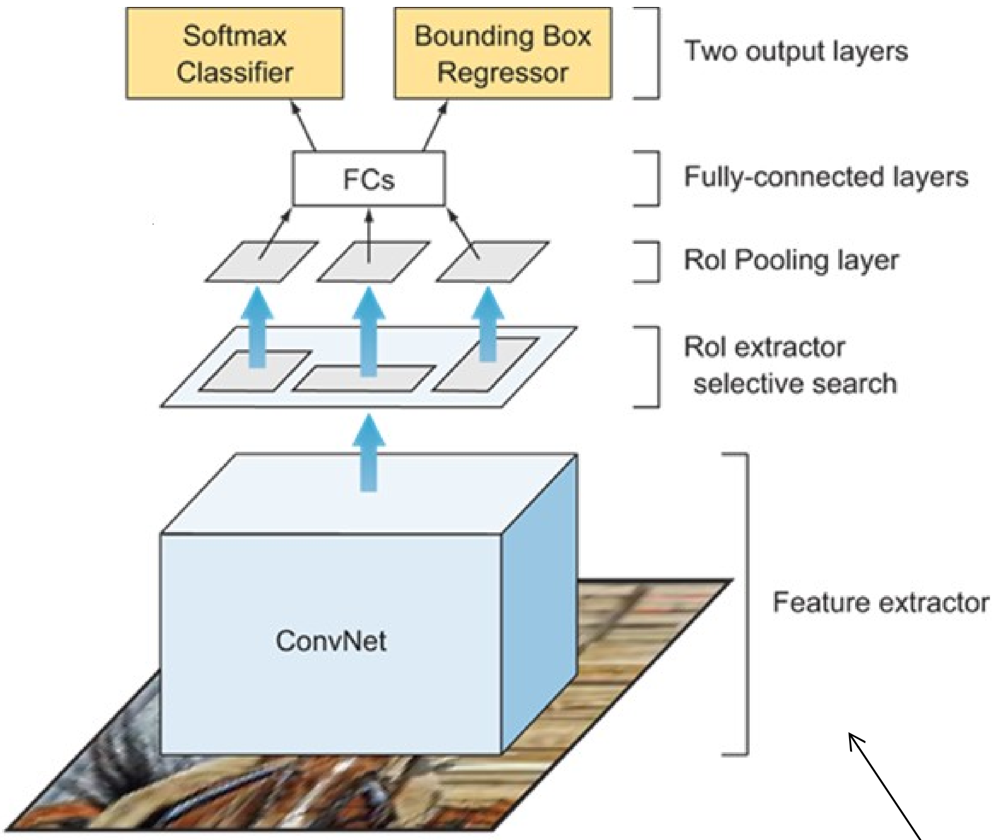


Fast RCNN

- <u>Idea:</u> feed to image only once to the CNN to extract a ***feature map,*** then crop and warp regions of this feature map

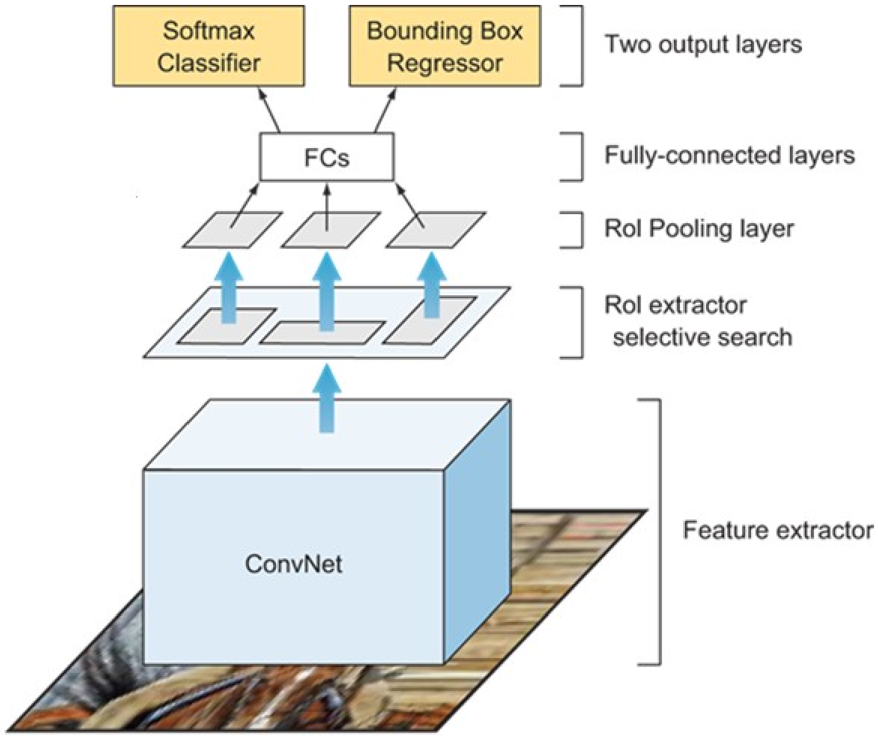


Fast R-CNN



Feed the ENTIRE image only once though the CONV net

Fast R-CNN

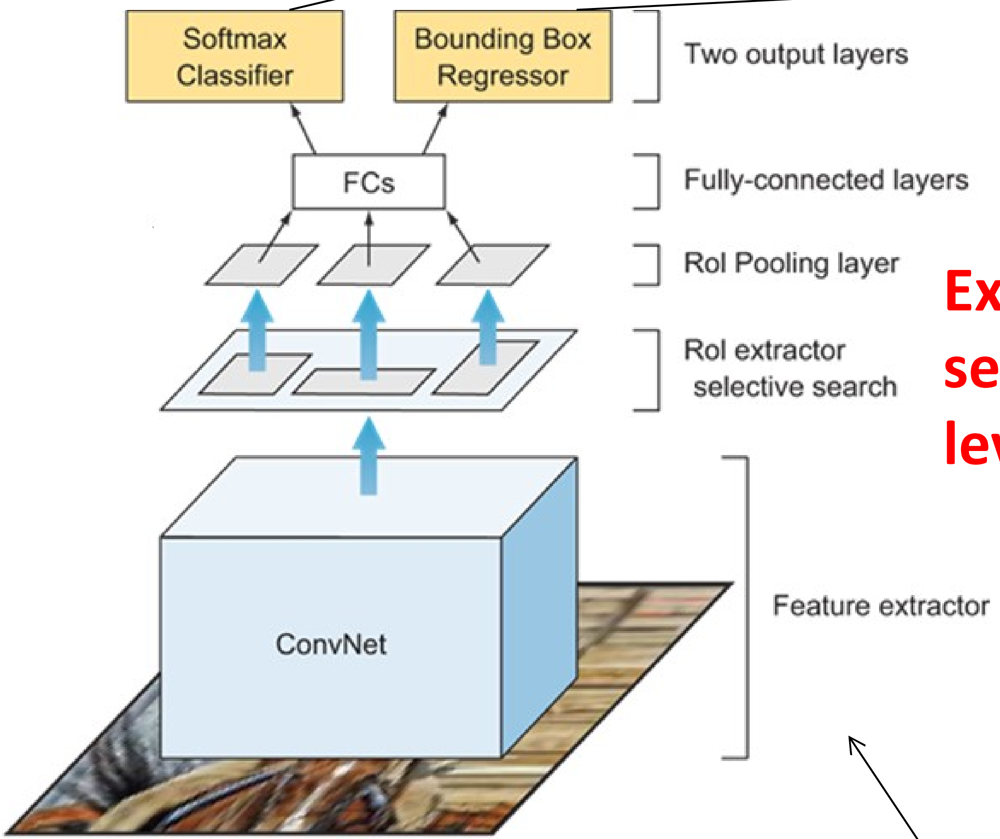


Extract the ROIs (proposed by selective search) at feature map level using RoI Pooling

Fast R-CNN

N classes + background

Bounding boxes
(x, y, w, h)



Extract the ROIs (proposed by selective search) at feature map level using RoI Pooling

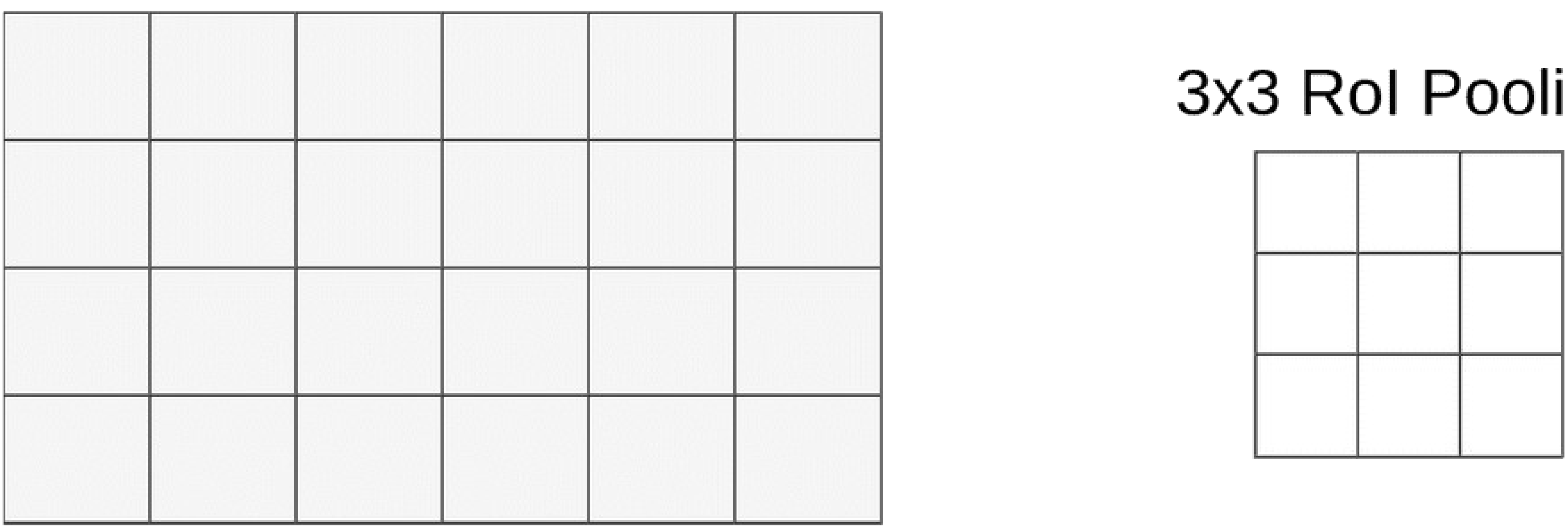
Feed the ENTIRE image only once though the CONV net

Roi Pooling

Divide the H × W ROI window into an h × w grid of sub-windows of approximate size H/h × W/w and then max-pooling the values in each sub-window into the corresponding output grid cell.

4x6 RoI

3x3 RoI Pooling



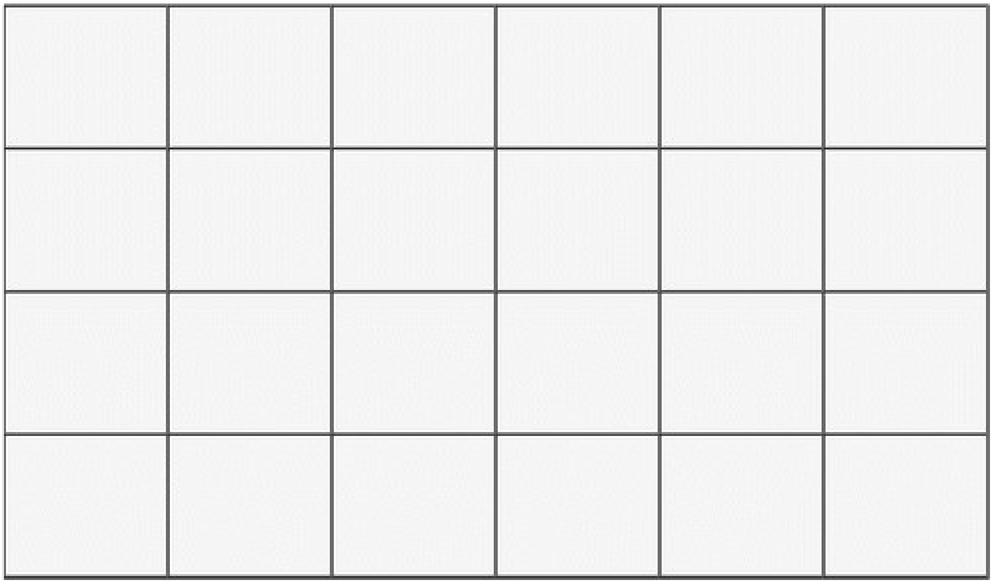
Roi Pooling

W = 6, H = 4

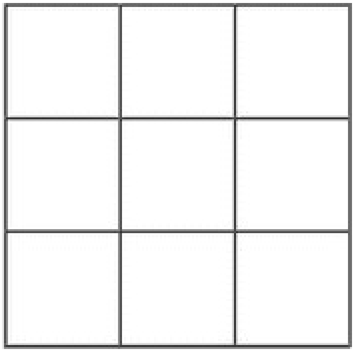
w = 3, h = 3

sz_w = 6/3=2, sz_h=4/3 = 1

4x6 RoI



3x3 RoI Pooling



Roi Pooling

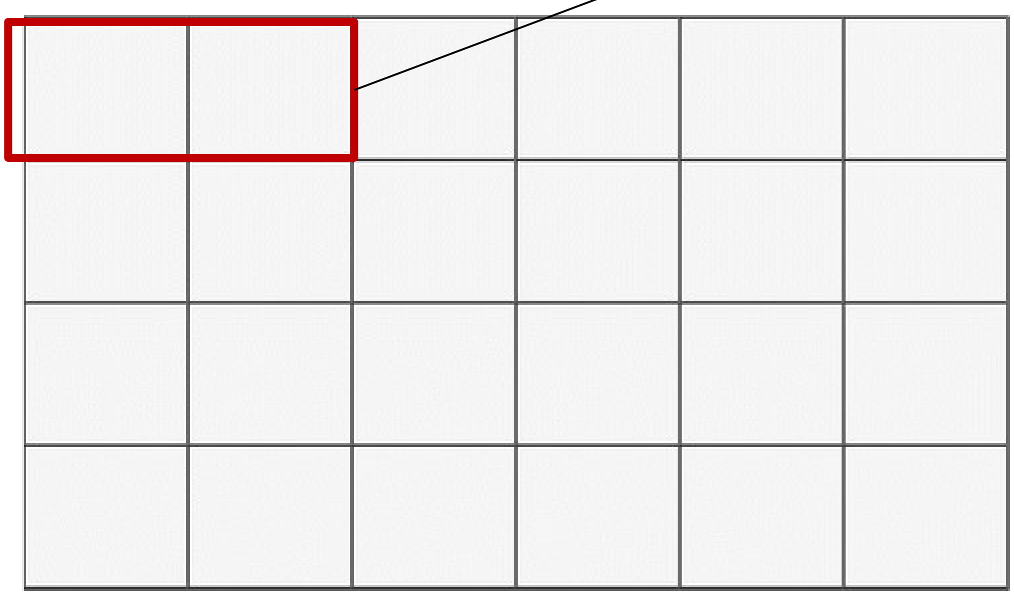
W = 6, H = 4

w = 3, h = 3

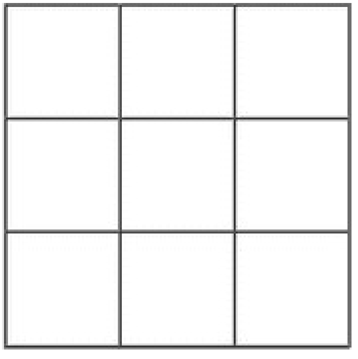
sz_w = 6/3=2, sz_h=4/3 = 1

Apply max pooling in each ROI

4x6 RoI



3x3 RoI Pooling



Roi Pooling

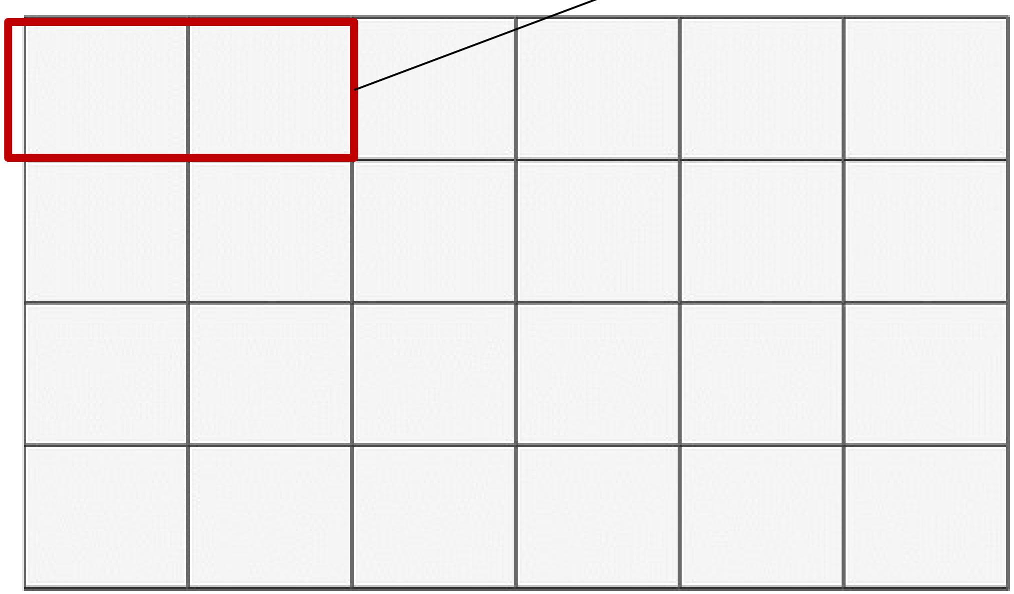
$W = 6, H = 4$

$w = 3, h = 3$

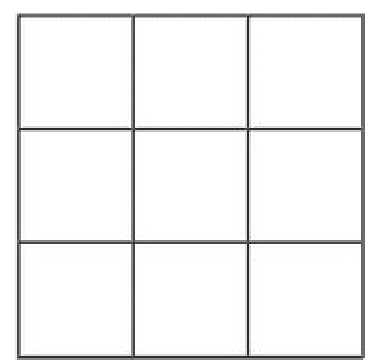
$sz\_w = 6/3=2,\ sz\_h=4/3 = 1$

Apply max pooling in each ROI

4x6 RoI



3x3 RoI Pooling

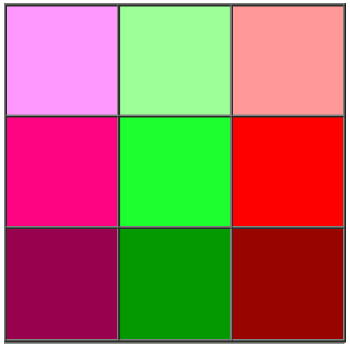


ROI Pooling example

4x6 RoI

0.1	0.2	0.3	0.4	0.5	0.6
1	0.7	0.2	0.6	0.1	0.9
0.9	0.8	0.7	0.3	0.5	0.2

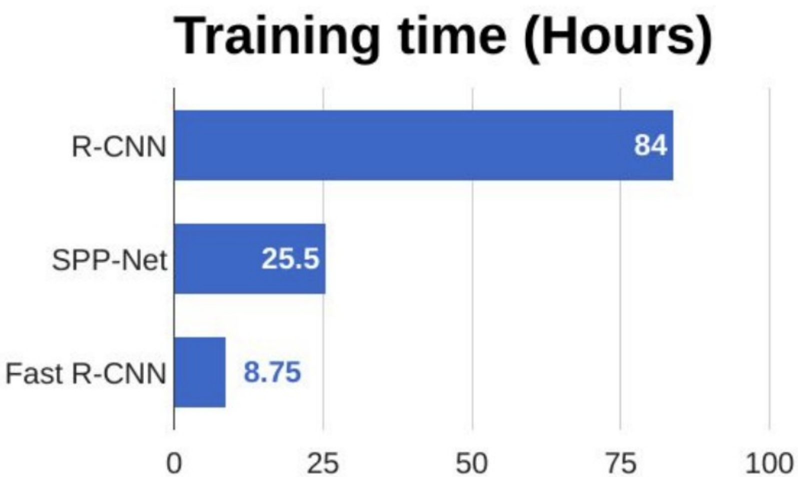
3x3 RoI Pooling

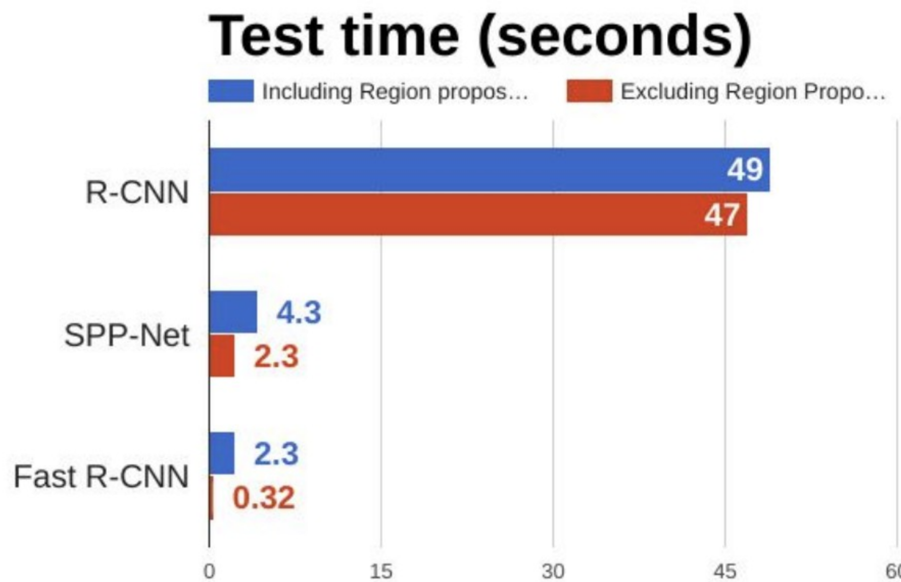


https://towardsdatascience.com/understanding-region-of-interest-part-2-roi-align-and-roi-warp-f795196fc193

https://towardsdatascience.com/understanding-region-of-interest-part-1-roi-pooling-e4f5dd65bb44

Fast RCNN

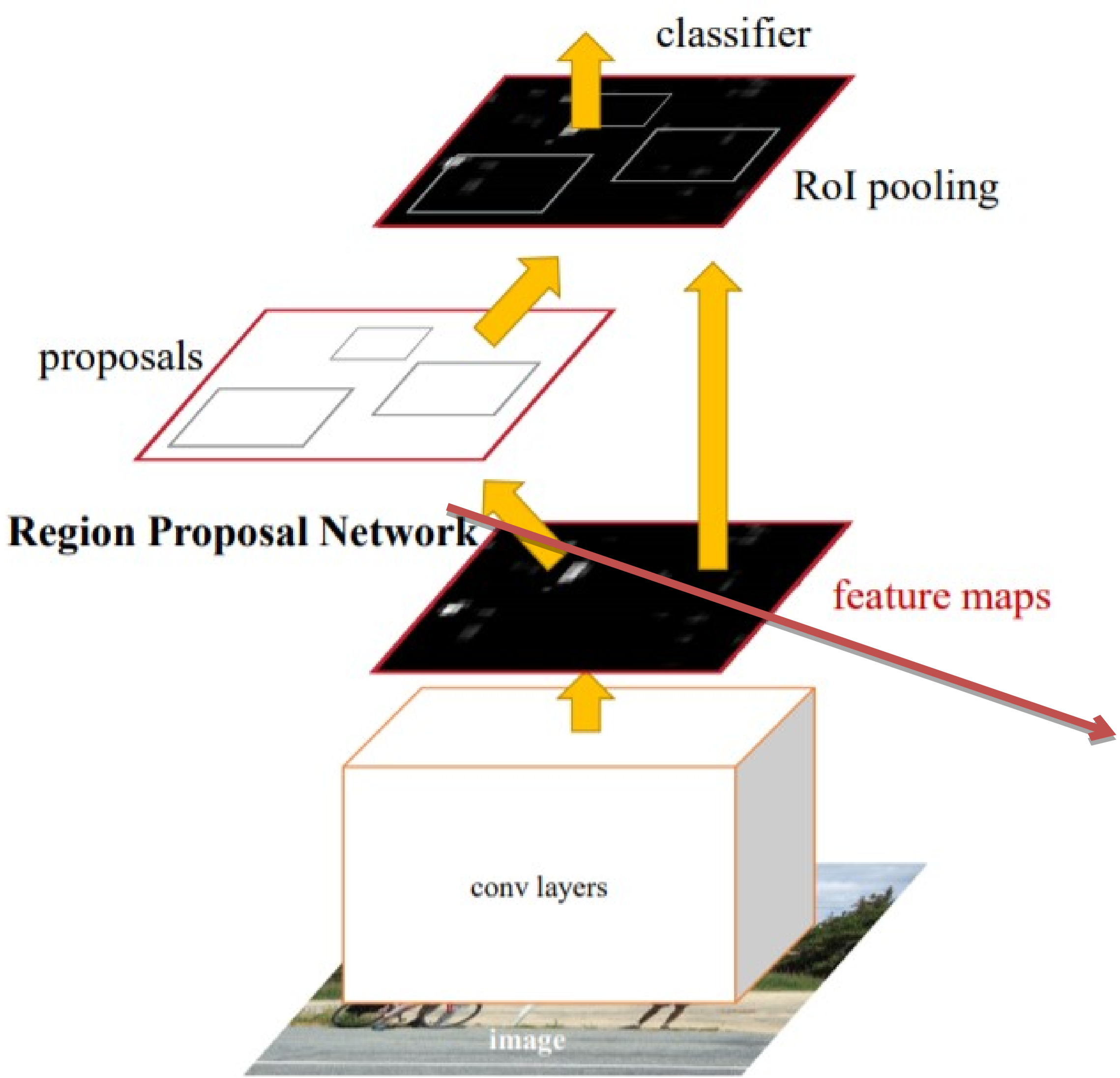




What is the most time consuming part of Fast RCNN?

Faster-RCNN

Unified framework for object detection

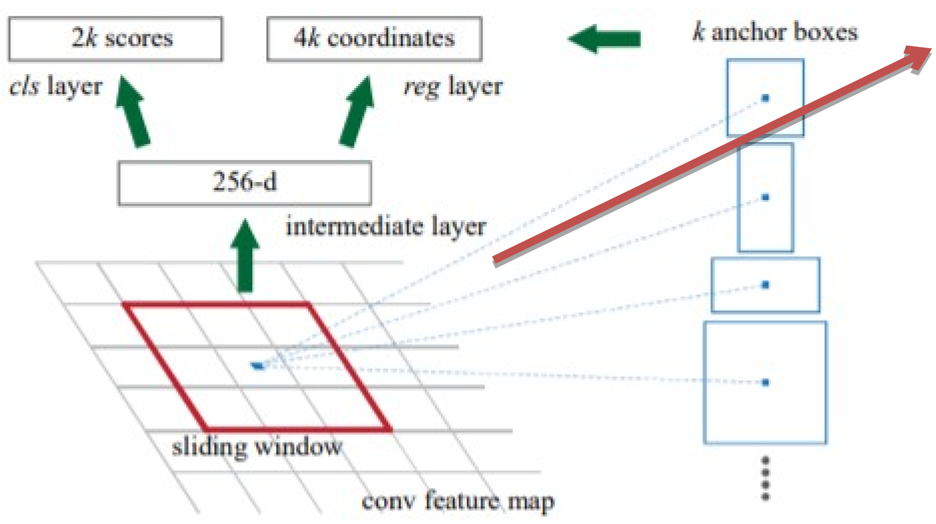


The rest of the network is similar to Fast RCNN: use ROI pooling to crop feature maps and classify each region

Region proposal network: replaces selective search and predicts regions directly from feature maps

Region proposal network

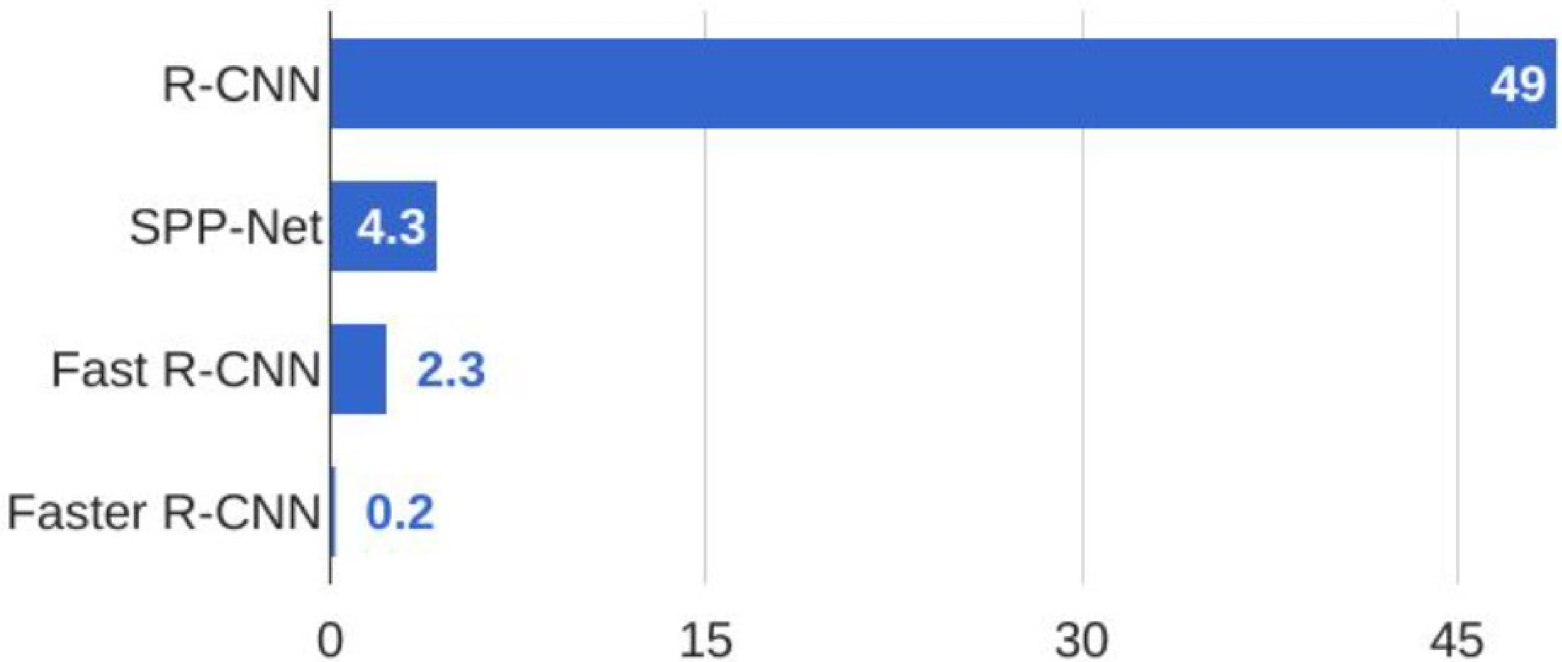
- takes an image (of any size) as input and outputs a set of rectangular object proposals, each with an objectness score



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

- For each anchor box predict if there is an object inside it
- For "positives" predict corrections for the bounding boxes

R-CNN Test-Time Speed



Faster RCNN

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

$$L_{1;smooth} = \begin{cases} |x| & \text{if } |x| > \alpha; \\ \frac{1}{|\alpha|}x^2 & \text{if } |x| \leq \alpha \end{cases}$$

Here, i is the index of an anchor in a mini-batch and p_i is the predicted probability of anchor i being an object. The ground-truth label p_i^* is 1 if the anchor is positive, and is 0 if the anchor is negative. t_i is a vector representing the 4 parameterized coordinates of the predicted bounding box, and t_i^* is that of the ground-truth box associated with a positive anchor. The classification loss L_{cls} is log loss over two classes (object *vs.* not object). For the regression loss, we use $L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$ where R is the robust loss function (smooth L_1) defined in [2]. The term $p_i^* L_{reg}$ means the regression loss is activated only for positive anchors ($p_i^* = 1$) and is disabled otherwise ($p_i^* = 0$). The outputs of the *cls* and *reg* layers consist of $\{p_i\}$ and $\{t_i\}$ respectively.

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

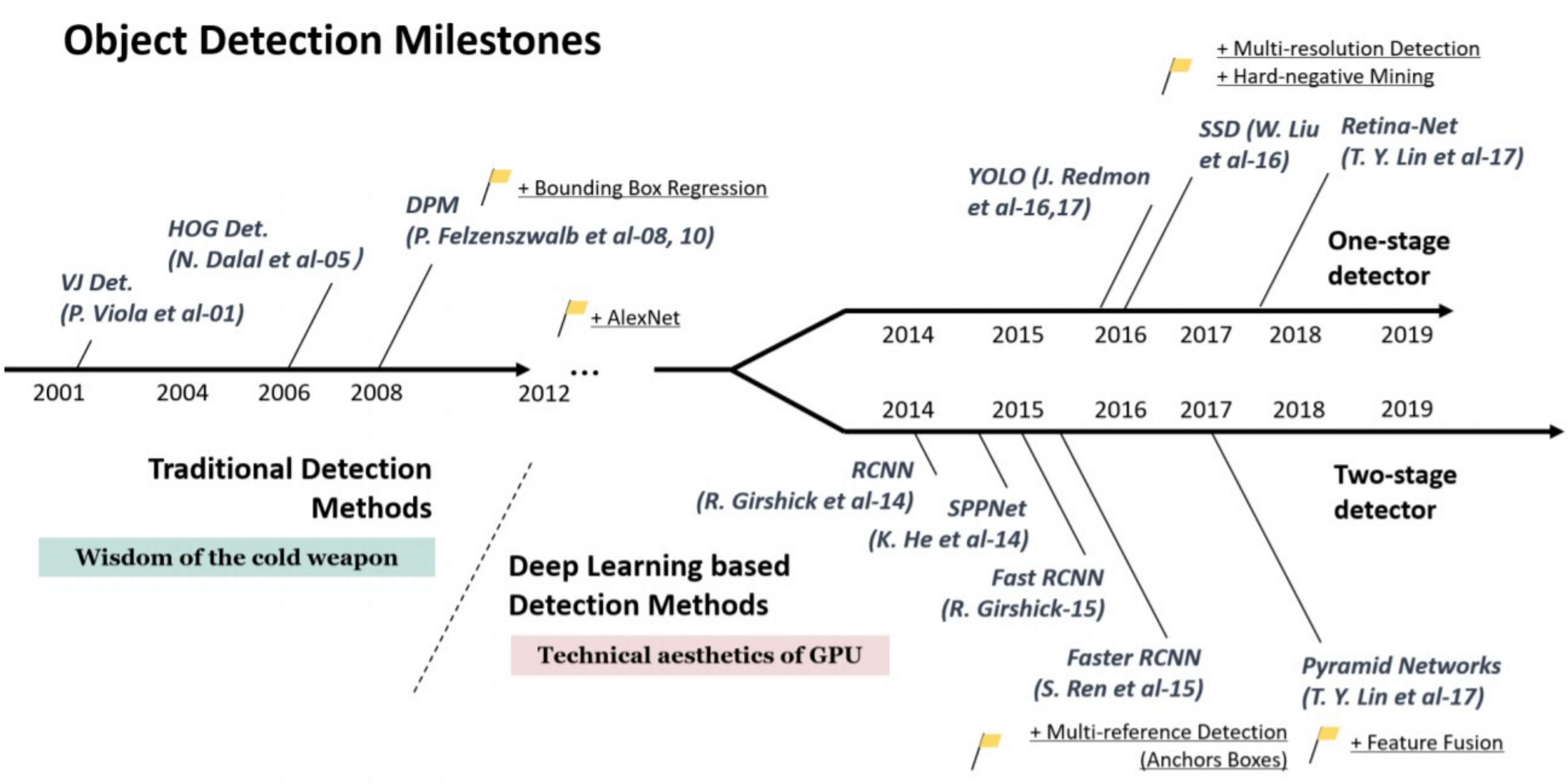
t* - ground truth
t –prediction
h_a , w_a – anchor box

bounding-box regression from an anchor box to a nearby ground-truth box.

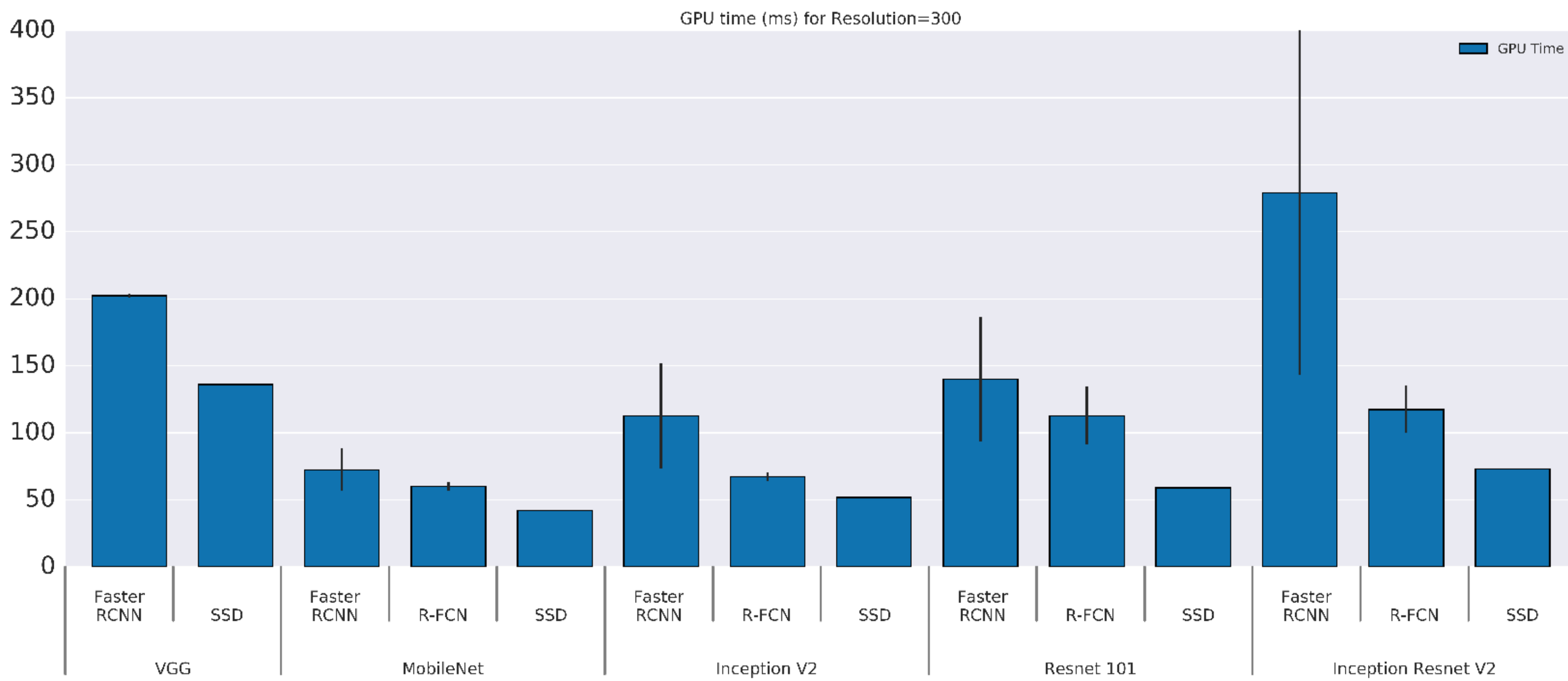
	R-CNN	**Fast R-CNN**	**Faster R-CNN**
Test time per image (with proposals)	50 seconds	2 seconds	**0.2 seconds**
(Speedup)	1x	25x	**250x**
mAP (VOC 2007)	66.0	**66.9**	**66.9**

Object detectors

Object Detection in 20 Years: A Survey



Object detectors



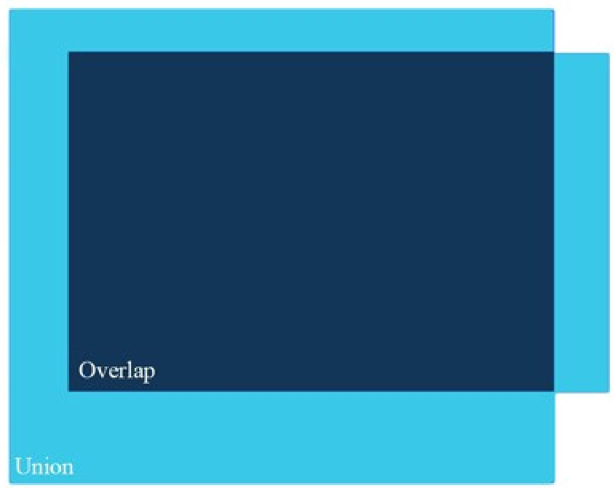
GPU time (milliseconds) for each model, for image resolution of 300.

https://jonathan-hui.medium.com/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [37]	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[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18

Evaluation metrics for object detection

$$IoU = \frac{\text{area of overlap}}{\text{area of union}}$$



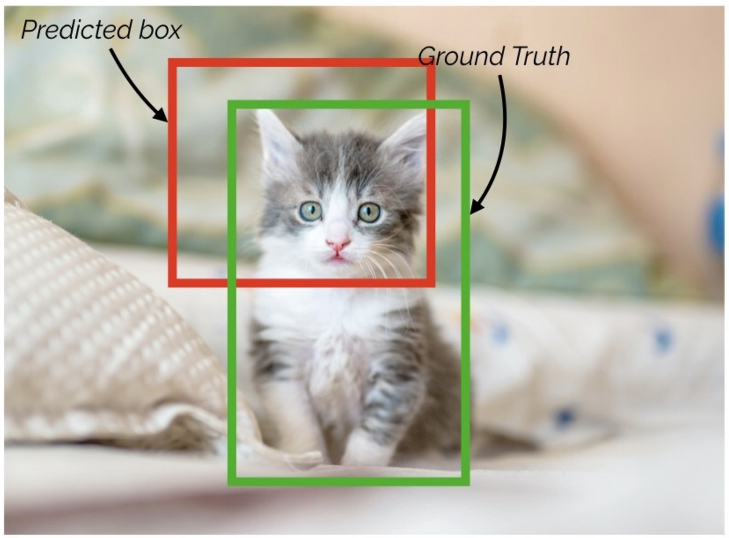
Use IOU to determine is the object is a TP, FP, or a FN

Remember **precision** and **recall**?

Images and notes from: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

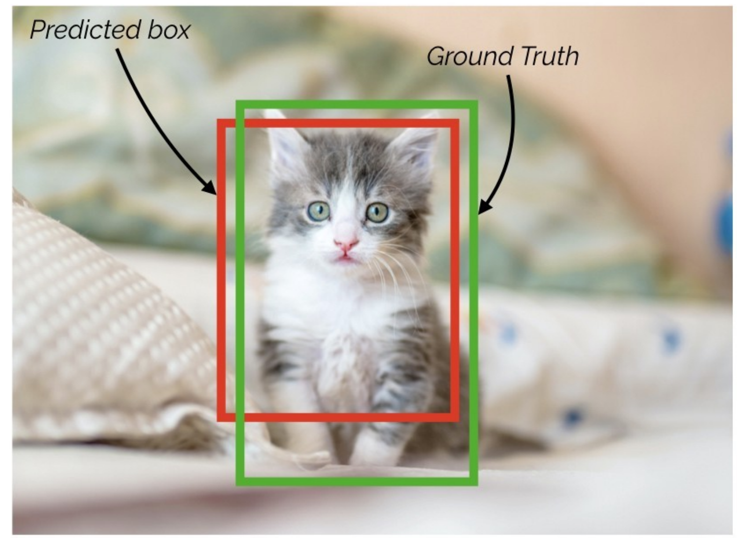
If IoU threshold = 0.5

False Positive (FP)



IoU = ~0.3

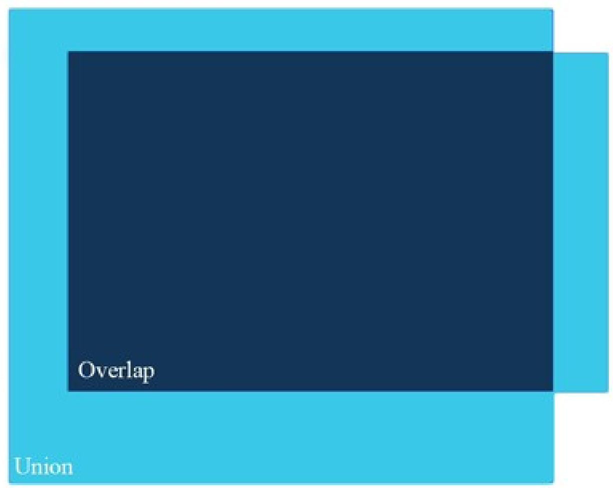
True Positive (TP)



IoU = ~0.7

Evaluation metrics for object detection

$$IoU = \frac{\text{area of overlap}}{\text{area of union}}$$



Use **IOU** and a **threshold** to determine is the object is a TP, FP, or a FN

Remember **precision** and **recall**?

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Images and notes from: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

Evaluation metrics for object detection
AP

Average Precision (AP) is finding the area under the precision-recall curve.

confidence

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

Images and notes from: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

Evaluation metrics for object detection

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

TP?
FP?
FN?
Precision?
Recall?

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Images and notes from: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173