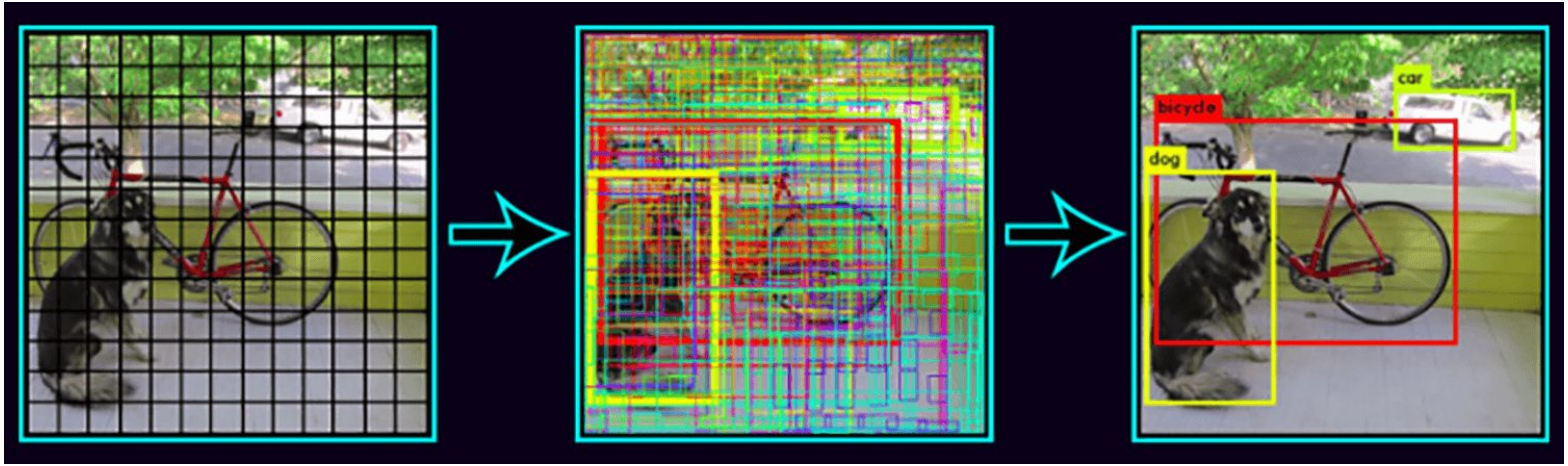
Object as points

CenterNet

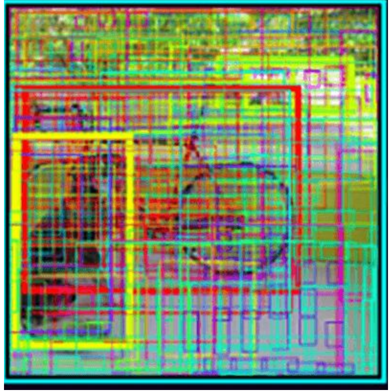
Recap : YOLOv2

1. Divide the image into K x K grid (each grid is 32 x 32)
2. Generate N box in each grid, and each box contains x, y, w, h, and other C+1 class

- Therefore K x K x N anchors are generated. (K, N often = 13, 5 respectively)



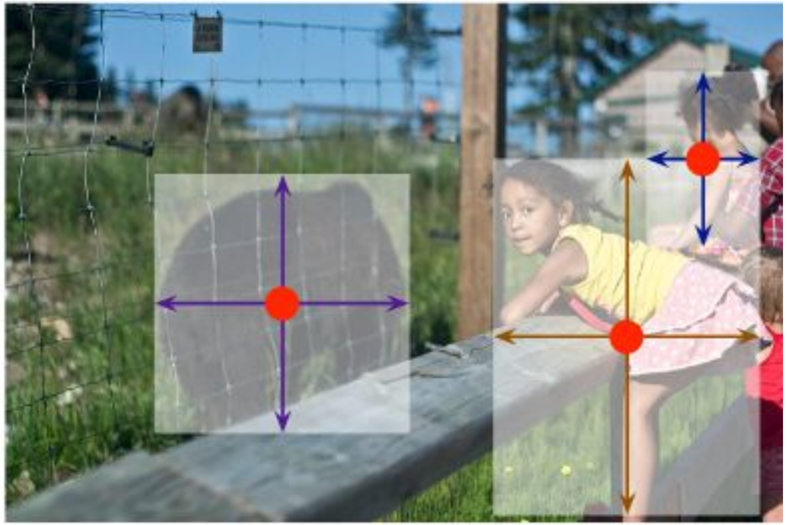
Problem with YOLO



- Problem with small object
    - The grid is too large (32 x 32)
    - There are a lots of problem when it comes to detecting small object since there is a high chance that small objects are in the same grid.
- Problem with bounding box (other object detection algorithm is included)
    - Anchors are human-aided
    - Too many bounding box are generated
    - Leads to NMS problem (O(n^2) computation cost and cannot be trained)
- **The second problem and some of the first problem will be solved using CenterNet.**

Solution : Reduce the problem to keypoint estimation

Goal : We place some keypoints and generate Bounding box from them.



Pros :

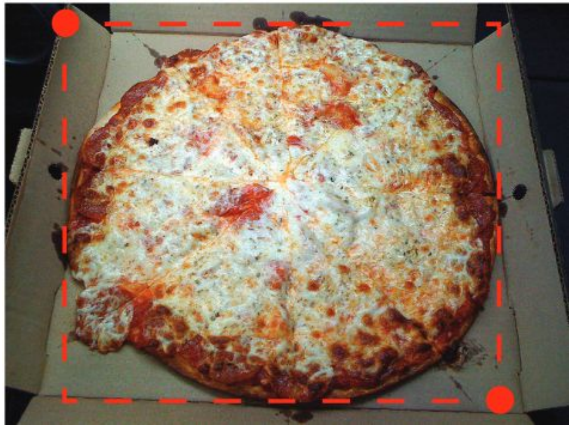
- NMS is not needed
- No anchor

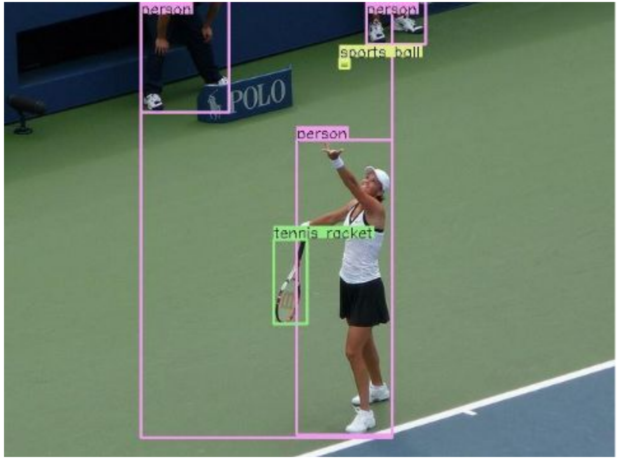
Cons

- Post processing problem

Short review : CornerNet

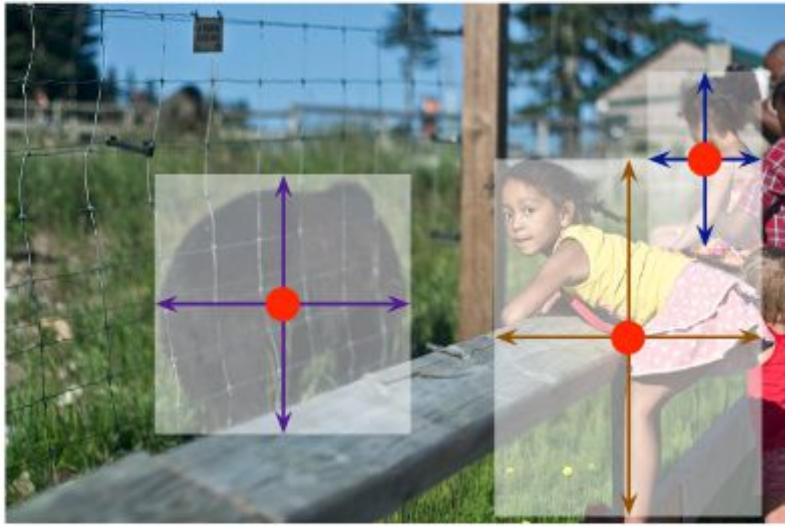
- Top-left and bottom-right keypoint are detected
- Method is similar to CenterNet (will be explained in the following slides)
- Problem : How to associates Top-left keypoint A and Bottom-right keypoint B -> hard post processing





CenterNet

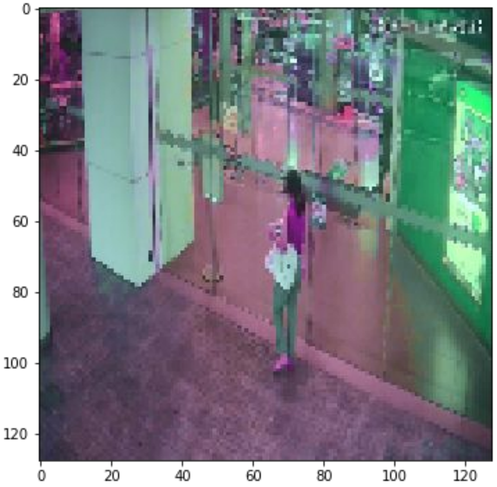
- Improves CornetNet by offer simpler post-processing methods
- Center keypoint, width, and height are predicted instead.



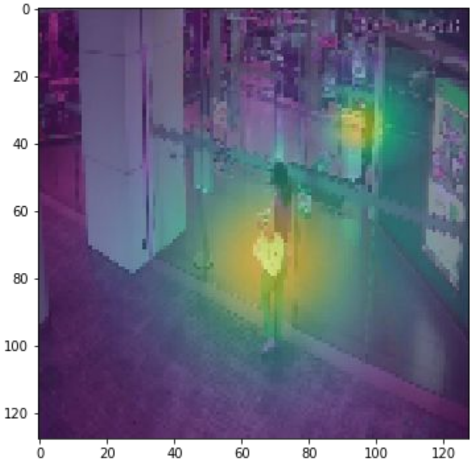
CenterNet

- The model predict C classes of keypoint, offset_x, offset_y, size_x, size_y (C+4 channel)

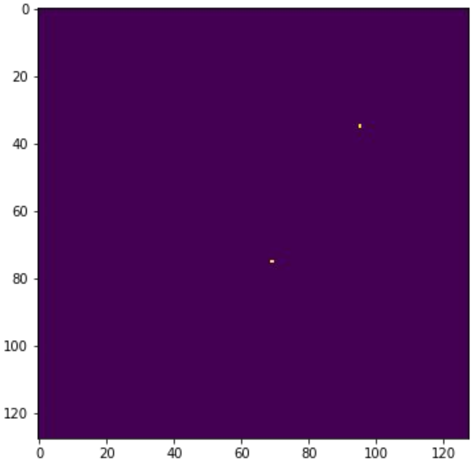
Original image



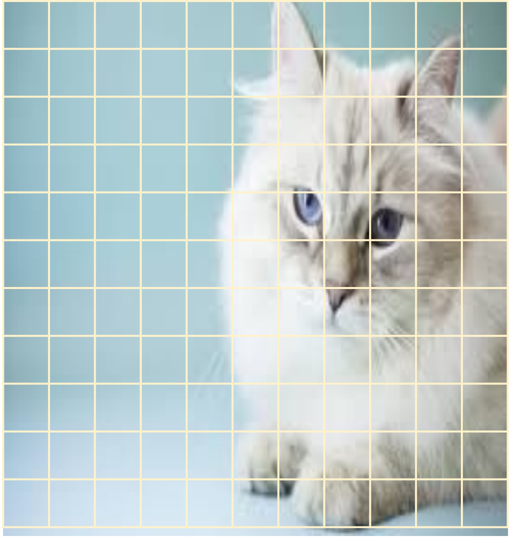
Keypoint



Offset, size

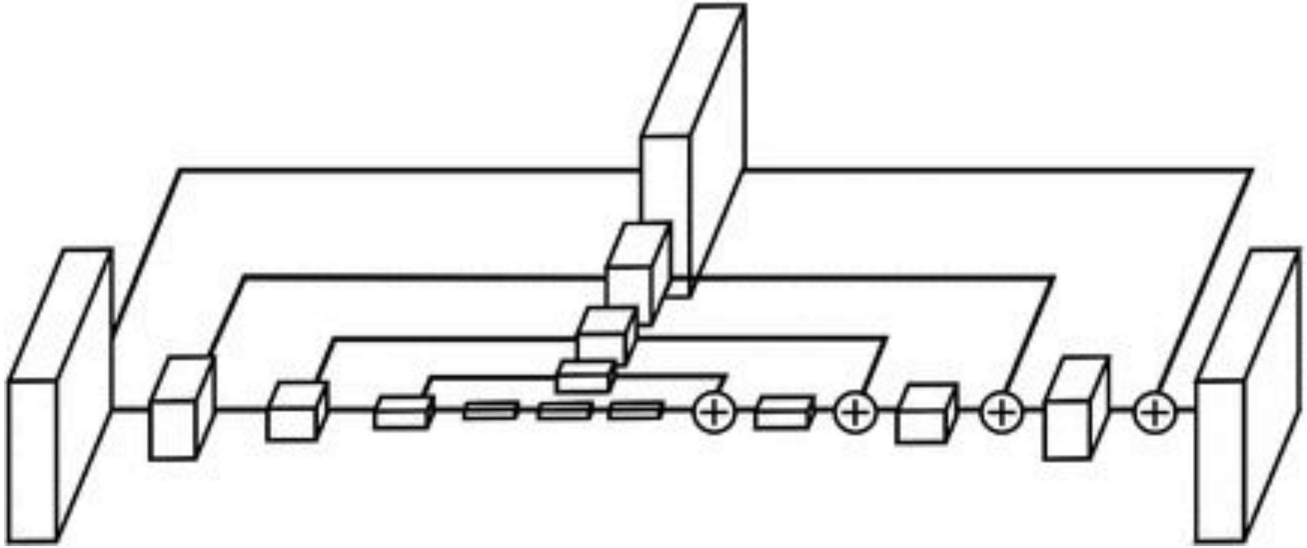


Solving YOLO grid size problem



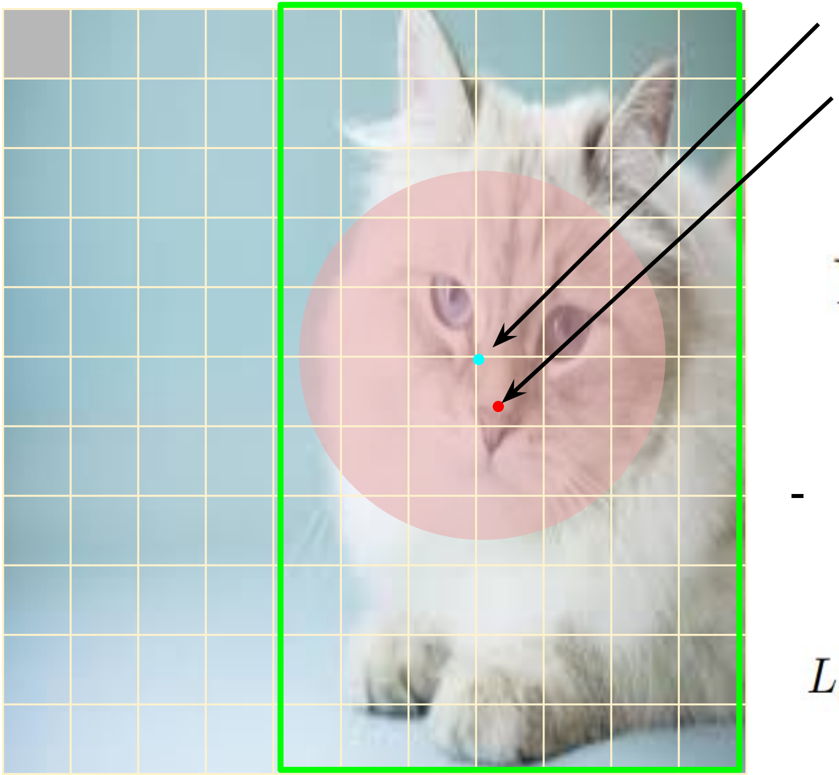
- In CenterNet, grid size is (4 x 4).
- Problem : The model is expected to be shallow since fewer pooling is allowed, which leads to lacks in global context.
- Solution : use encoder-decoder networks

Encoder-decoder networks



Keypoint prediction

Grid x =0, y= 0



Low resolution equivalent $\tilde{p} = \lfloor \frac{p}{R} \rfloor$

Center (p_x, p_y)

Generate Gaussian Kernel

$$Y_{xyc} = \exp\left(-\frac{(x-\tilde{p}_x)^2+(y-\tilde{p}_y)^2}{2\sigma_p^2}\right),$$

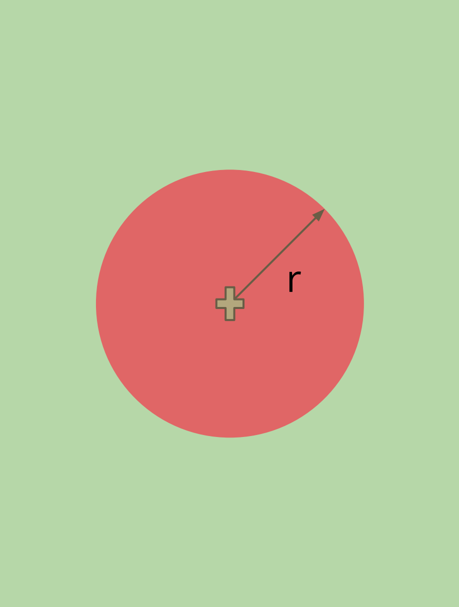
Its guarantee that there exists $Y_{xyc} = 1$

- Keypoint loss (Focal loss, a = 2, b = 4)

$$L_k = \frac{-1}{N}\sum_{xyc}\begin{cases}(1-\hat{Y}_{xyc})^{\alpha}\log(\hat{Y}_{xyc}) & \text{if } Y_{xyc}=1\\ (1-Y_{xyc})^{\beta}(\hat{Y}_{xyc})^{\alpha} & \text{otherwise}\\ \log(1-\hat{Y}_{xyc}) & \end{cases}$$

Keypoint prediction

- How to calculate variance (the radius of the circle)?

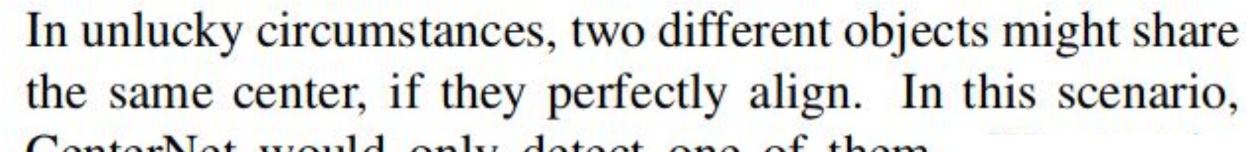


Solution : Find r such that minIoU of the rectangle and the circle is at least X (X = 0.7 in the paper)

Therefore, $\sigma = \frac{r}{3}$

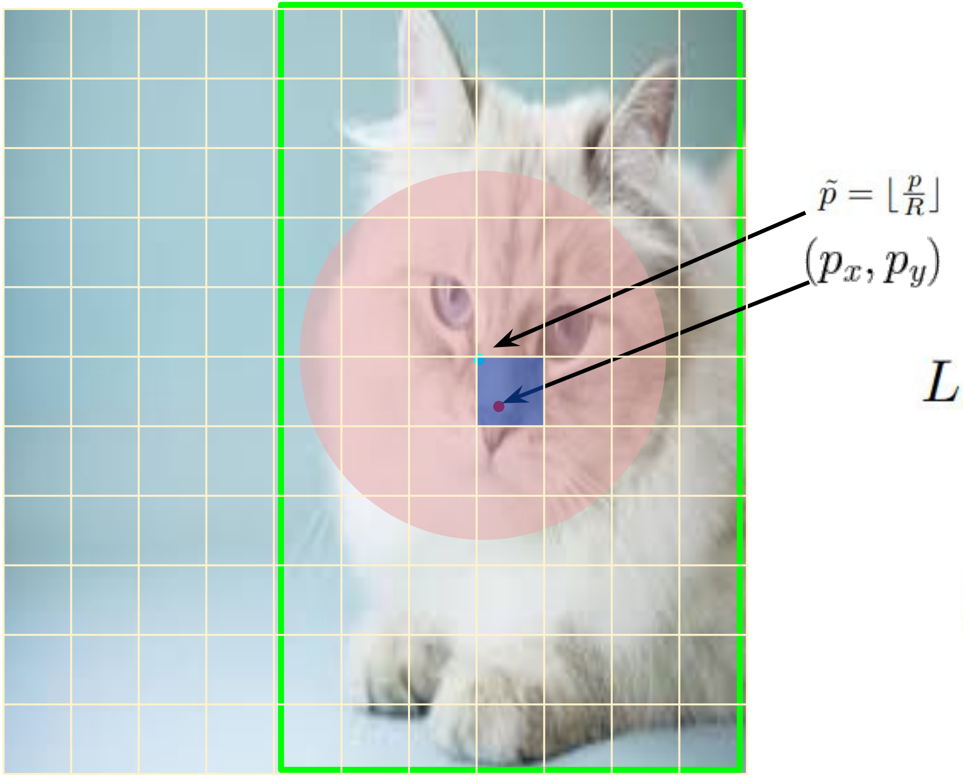
Problem with keypoint prediction

- Q1 : What if the circles are overlapping?
- A1 : Use max-element wise value.
- Q2 : What if there are multiple centers in the same grid?
- A2 : Ignore them (It rarely occurs since grid size is small)



In unlucky circumstances, two different objects might share the same center, if they perfectly align. In this scenario, CenterNet would only detect one of them.

Offset and size prediction



- The grid that has the center of the object is responsible for predicting offset and size.
- Loss is L1 or smooth-L1 loss

$$L_{off} = \frac{1}{N} \sum_{p} \left| \hat{O}_{\tilde{p}} - \left(\frac{p}{R} - \tilde{p} \right) \right| .$$

$$L_{size} = \frac{1}{N} \sum_{k=1}^{N} \left| \hat{S}_{p_k} - s_k \right| .$$

Total loss

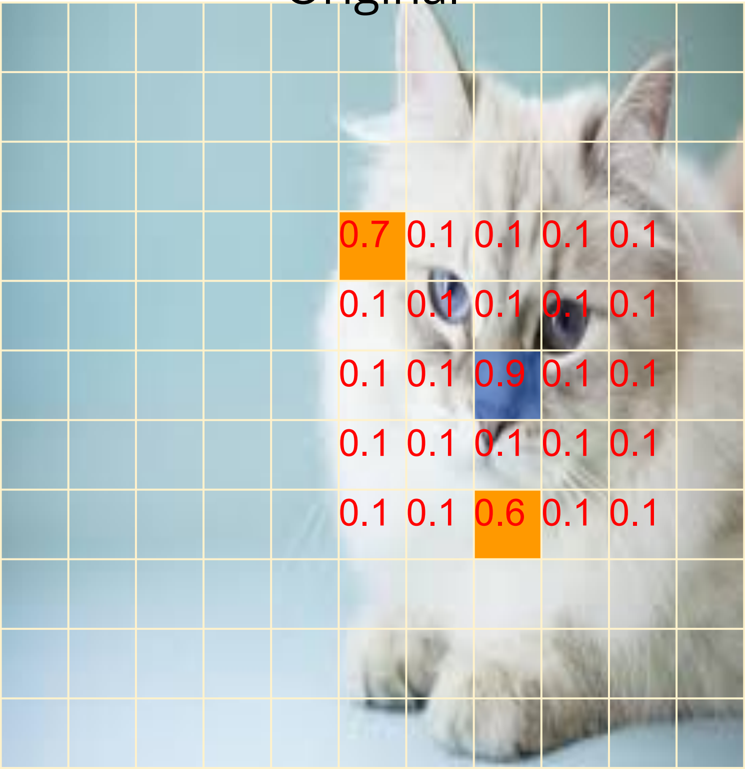
$$L_{det} = L_k + \lambda_{size} L_{size} + \lambda_{off} L_{off}.$$

where $\lambda_{size} = 0.1$ and $\lambda_{off} = 1$

Post- processing

- The location of the keypoint is the point where its value is greater than its neighbour
- Solution : Use maxpool

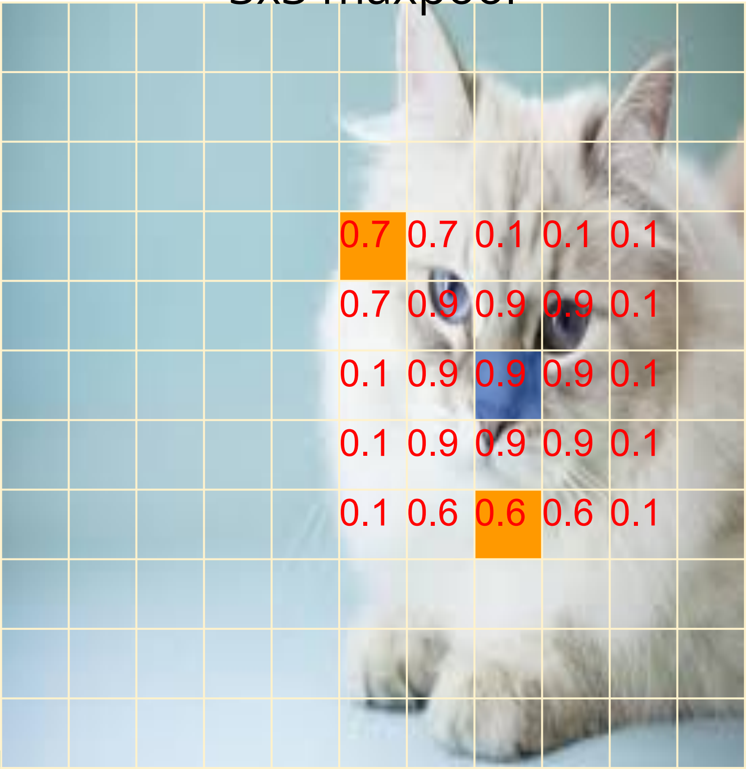
Original



0.7	0.1	0.1	0.1	0.1
0.1	0.1	0.1	0.1	0.1
0.1	0.1	0.9	0.1	0.1
0.1	0.1	0.1	0.1	0.1
0.1	0.1	0.6	0.1	0.1

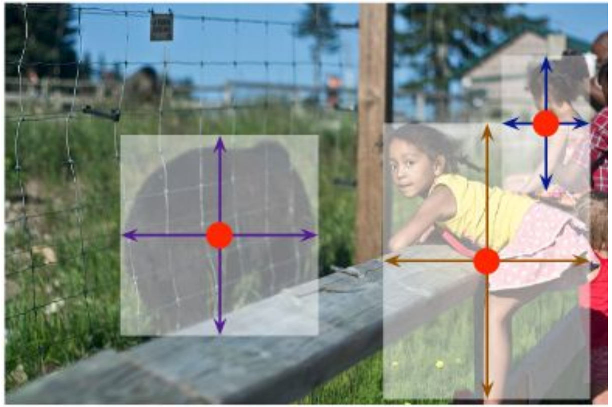
==

3x3 maxpool



0.7	0.7	0.1	0.1	0.1
0.7	0.9	0.9	0.9	0.1
0.1	0.9	0.9	0.9	0.1
0.1	0.9	0.9	0.9	0.1
0.1	0.6	0.6	0.6	0.1

Post- processing



- For offset and size it is computed normally.

Therefore, the bounding box location is at

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