Indepth Guide to YOLO

ENEE 4584/5584

DL w CV apps

Classification vs Regression

- Algorithms based on classification
  - two stages process
    - select regions of interest in an image
    - classify these regions using convolutional neural networks.
  - R-CNN family
- Algorithms based on regression
  - predict classes and bounding boxes for the whole image
  - in one run of the algorithm.
  - YOLO , SSD (Single Shot Multibox Detector).
  - Faster

Dataset: PASCAL VOC

- http://host.robots.ox.ac.uk/pascal/VOC/
- 2005-2012
- Included images from flikr
- Started with
  - 4 classes,
  - 1578 images containing
  - 2209 annotated objects.
- Ended with
  - 20 classes,
  - 11,530 images containing
  - 27,450 ROI annotated objects
  - 6,929 segmentations.

Pascal VOC Example

Figure 1: Example datapoint in PascalVOC

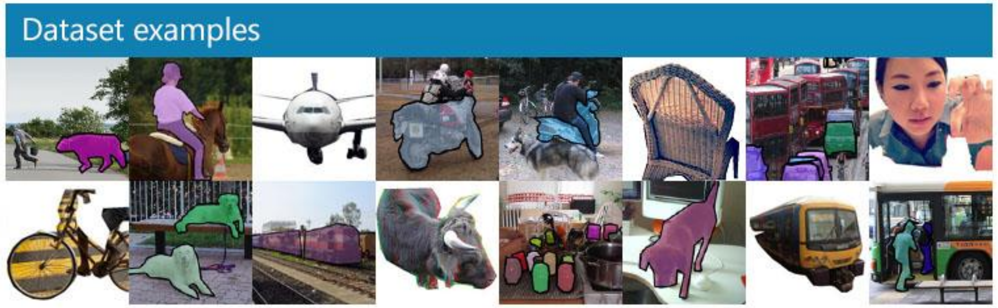
(a) Image: 2008_000089.jpg

(b) Annotation: 2008_000089.xml


```xml
<annotation>
  <folder>VOC2012</folder>
  <filename>2008_000089.jpg</filename>
  <source>
    <database>The VOC2008 Database</database>
    <annotation>PASCAL VOC2008</annotation>
    <image>flickr</image>
  </source>
  <size>
    <width>376</width>
    <height>500</height>
    <depth>3</depth>
  </size>
  <segmented>1</segmented>
  <object>
    <name>chair</name>
    <pose>Frontal</pose>
    <truncated>0</truncated>
    <occluded>0</occluded>
    <bndbox>
      <xmin>71</xmin>
      <ymin>18</ymin>
      <xmax>307</xmax>
      <ymax>494</ymax>
    </bndbox>
    <difficult>0</difficult>
  </object>
</annotation>
```

Dataset: COCO

- https://cocodataset.org/#home
- Sponsored by industry
- 330K images (>200K labeled)
- 1.5 million object instances
- 80 object categories
- 91 stuff categories
- 5 captions per image
- 250,000 people with keypoints



Dataset: ImageNet

- http://image-net.org/
- organized according to the WordNet hierarchy
  - Each meaningful concept in WordNet is called a "synonym set" or "synset“
  - 100,000+ synsets in WordNet
    - majority of them are nouns (80,000+)
  - Aims to provide on average 1000 images to illustrate each synset.
  - Images of each concept are quality-controlled and human-annotated.

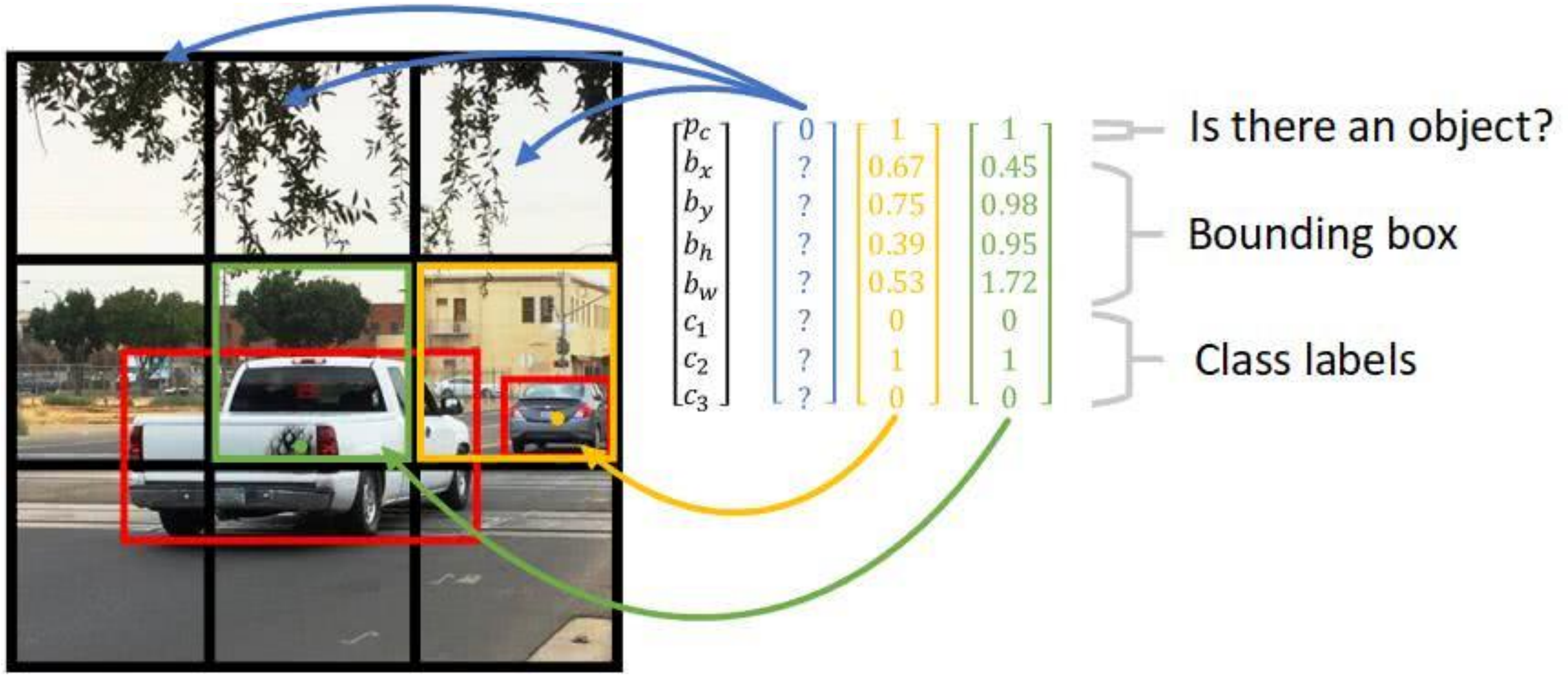
YOLO: Pre-processing Dataset

- Divide each image into SxS cells
- Computer centers, height, width of each box
  - Align centers to a cell origin (top-left corner)
  - Centers will be less than 1
  - h, w measured in cell sizes
    - can be greater than (if box is bigger than cell)
  - Multiple boxes can exist in each cell
- Create a ground truth for each image:

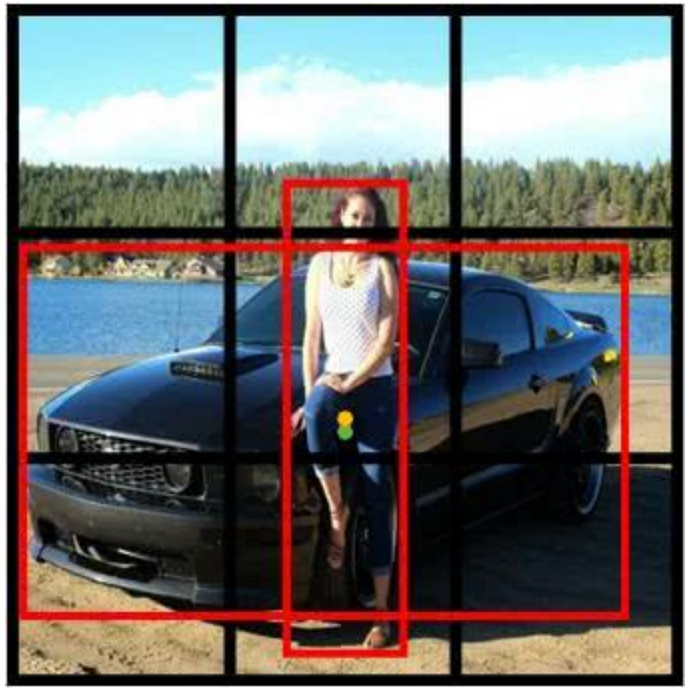
$$Y = \{(p_c, b_x, b_y, b_h, b_w, c), (p_c, b_x, b_y, b_h, b_w, c), \dots\}$$

  - Each box is made of 5 variables
  - Multiple boxes per cell can defined

$$(p_c, b_x, b_y, b_h, b_w, c)$$



https://www.dlology.com/blog/gentle-guide-on-how-yolo-object-localization-works-with-keras-part-2/



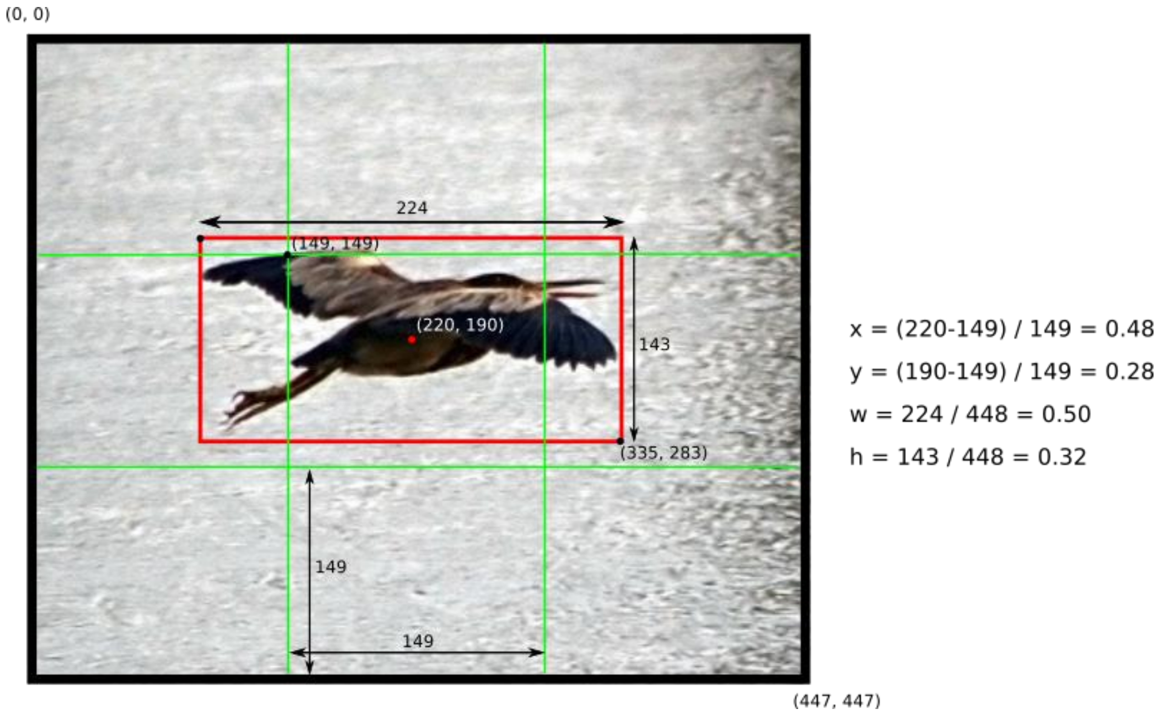
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 1 \\ 0 \\ 0 \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}1\\1\\1\\1\\1\\1\\1\\1\end{matrix}}\right\} \begin{matrix}\text{Anchor box 1} \\ \text{Pedestrian}\end{matrix} \\ \left.\vphantom{\begin{matrix}1\\1\\1\\1\\1\\1\\1\\1\end{matrix}}\right\} \begin{matrix}\text{Anchor box 2} \\ \text{Car}\end{matrix} \end{matrix}$$

Calculating Box Parameters

- Dataset will contain bounding box data as: $(x_{min}, y_{min}, x_{max}, y_{max})$
- Need to calculate center x, y and h, w:

$$x_c = \frac{x_{max} + x_{min}}{2 \times w_{cell}}, y_c = \frac{y_{max} + y_{min}}{2 \times h_{cell}}$$

$$w = \frac{x_{max} - x_{min}}{w_{cell}}, h = \frac{y_{max} - y_{min}}{h_{cell}}$$



https://hackernoon.com/understanding-yolo-f5a74bbc7967

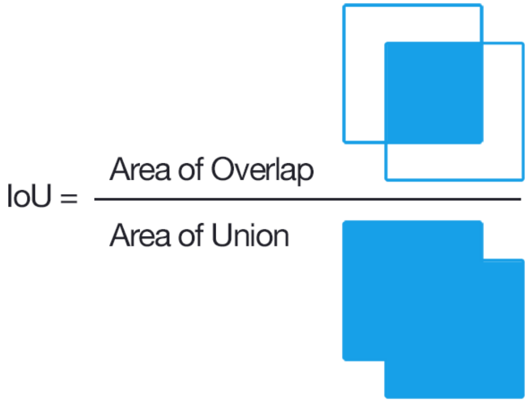
Box Vecor

- YOLO v1 converts the box parameters:

$$b = \begin{bmatrix} x_c \\ y_c \\ w \\ h \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} x_c \\ y_c \\ \sqrt{w} \\ \sqrt{h} \end{bmatrix}$$

- Square root is used to make it more sensitive to small box (smaller fractions)
  - Less sensitive to large boxes.

IoU



- Predicted box: $\boldsymbol{a} = (a_{x_{min}}, a_{y_{min}}, a_{x_{max}}, a_{y_{max}})$
- Ground truth box: $\boldsymbol{b} = (b_{x_{min}}, b_{y_{min}}, b_{x_{max}}, b_{y_{max}})$
- IoU = intersection of a with b /union of a with b

Algorithm 1 Calculating IoU for two rectangles

1: $a = a_{x_{\min}}, a_{y_{\min}}, a_{x_{\max}}, a_{y_{\max}}$ ▷ Rectangle defined by top-left and bottom-right coordinates
2: $b = b_{x_{\min}}, b_{y_{\min}}, b_{x_{\max}}, b_{y_{\max}}$
3: $|a \cap b|_w = \max(0, \min(a_{x_{\max}}, b_{x_{\max}}) - \max(a_{x_{\min}}, b_{x_{\min}}))$ ▷ Intersection width as overlap in x-axis
4: $|a \cap b|_h = \max(0, \min(a_{y_{\max}}, b_{y_{\max}}) - \max(a_{y_{\min}}, b_{y_{\min}}))$ ▷ Intersection height as overlap in y-axis
5: $|a \cap b| = |a \cap b|_w \cdot |a \cap b|_h$ ▷ Intersection area
6: $|a \cup b| = |a| + |b| - |a \cap b|$ ▷ Union area, $|a|$ is area of rectangle a
7: $\text{IoU(a,b)} = \dfrac{|a \cap b|}{|a \cup b|}$

YOLO: Architecture

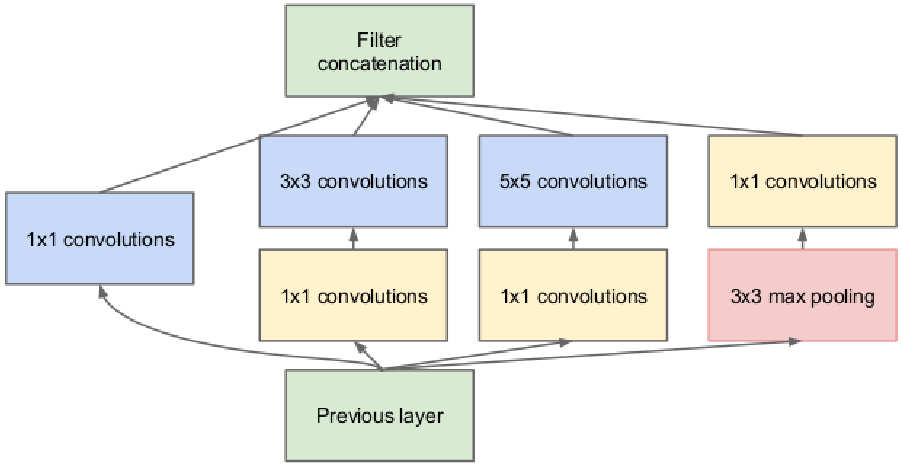
- 3 stages:
  - Multiple Convolutional layers
  - 1 Fully connected layer
  - Output layer
- 1 Full connected layer: very wide (1000s of neurons)
  - Good at translation invariance
  - Use linear activation
- Output layer: multiple outputs with varying loss functions

Convolution Layers

- Convolution + RELU + maxpooling
  - Leaky ReLU recommended
- Small size filters recommended (typically 3x3)
- Large number of filters (100s-1000s)
  - Lower numbers at early layers, larger numbers at output
- Deep architecture > 5 layers
- Sequences of 1x1 reduction layers and 3x3 convolutional layers
  - Inspired by GoogLeNet (Inception) model

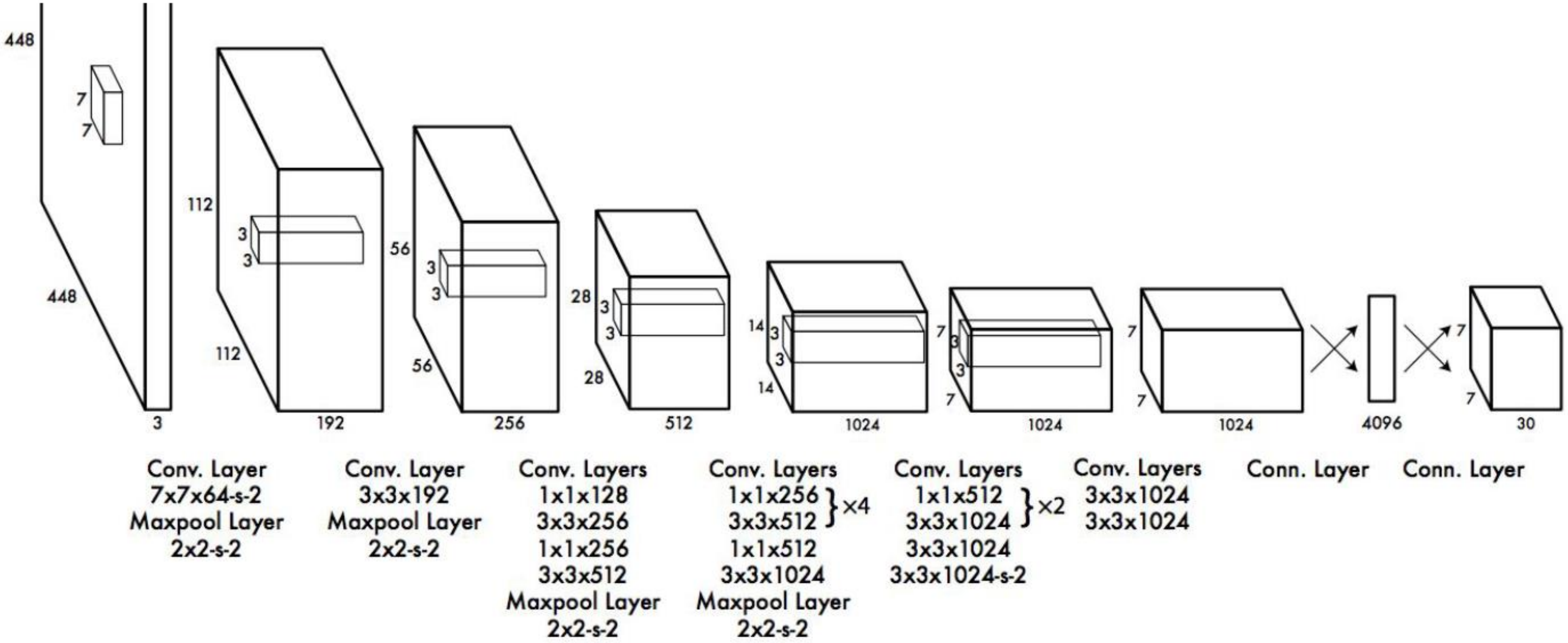
Side Note: 1x1 Convolutions

- https://arxiv.org/abs/1409.4842
- Dimensionality reduction + performance boost + computationally efficient
- Example:
  - given (28x28)x32 feature maps. Apply (5x5)x192 filters
  - Computations = 5x5x28x28x32x192 = 120M
  - Repeat by using (1x1)x10 then (5x5)x192
  - Computations = (1x1x28x28x32x10) + (5x5x28x28x10x192) = 37.9M



YOLO Architecture from Paper

- Designed for Pascal VOC dataset
- Input PASCAL VOC: 448x448x3
- Output = SxSx(5B+C)
- SxS: cells in an image
- B=2 boxes in a cell
- C=20 classes
  - 1-Hot encoded



Loss Function

Loss = Box Loss + Confidence Loss + Classification Loss

Loss Part 1: Center, Height, Width

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

- i = cell number
- j = box number
- $\mathbb{1}_{ij}^{obj} = 1$ if a center in cell i, and box j is responsible for the object
- $\sqrt{}$ more sensitive to small boxes

Loss Part 2: Confidence Loss

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

- C_i confidence is $Confidence\ =\ P(Obj)\ IoU$

Testing

- Resize image to fit typical input
  - Alternatively: Split image into tiles
- 1 or more boxes will be predicted in each cell
  - Boxes can be greater than cell
  - Each box will have a prediction parameter (aka confidence): p_c
  - Threshold/pick boxes with highest p_c