You Only Look Once: Unified, Real-Time Object Detection

Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶

http://pjreddie.com/yolo/

Presented by

Xuwang Yin

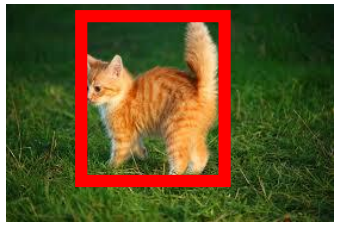
Computer Vision Tasks: Classification vs Detection

- Classification
  - Input: Image
  - Output: Class Label

Cat

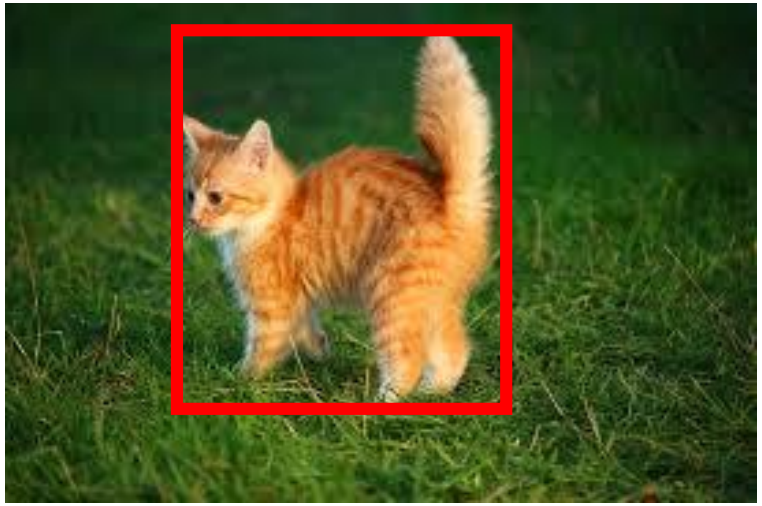
Dog

- Detection
  - Input: Image
  - Output: Bounding-box of Object



(x, y, w, h)

Methods for Object Detection



- Region Proposals Method
- Sliding Window Method
- Regress on Object Position
  - Want to learn a function that takes the image as input, and output the bounding-box of the object in the image.
  - Classic Mean Square Error Loss Function:

$$\text{loss} = \frac{1}{n}\sum_{i=1}^{n}\left((\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2 + (\hat{w}_i - w_i)^2 + \left(\hat{h}_i - h_i\right)^2\right)$$

  - Regression function is learned by minimizing the loss function.

Detection as Regression - Intuition

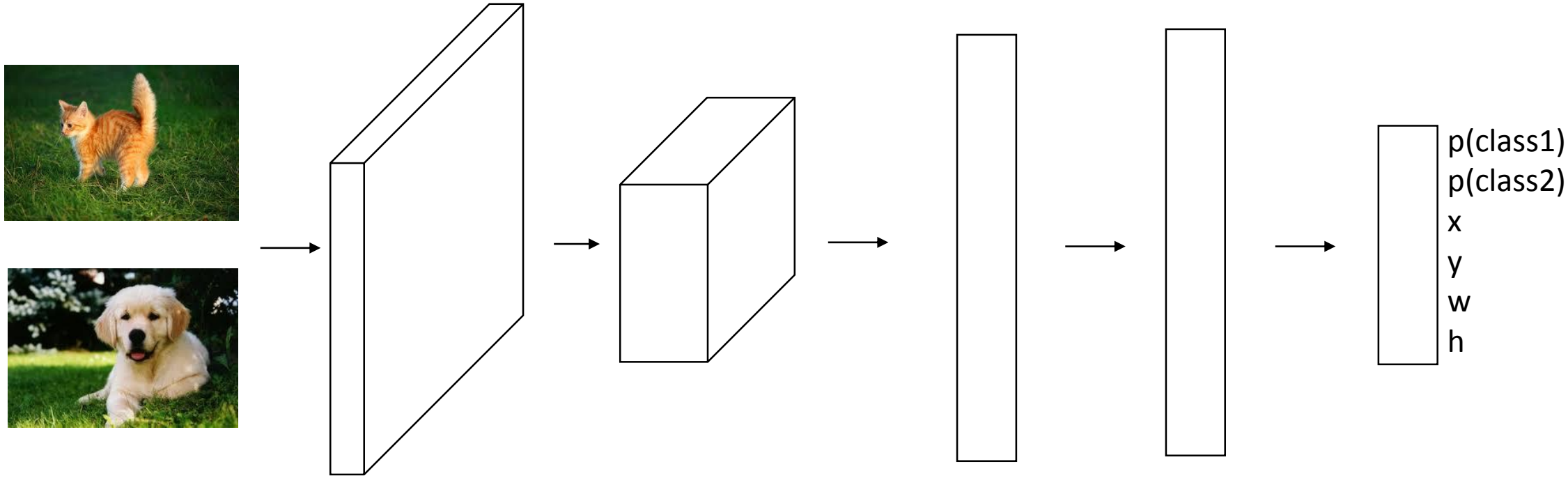
$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta}$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \mathbf{x}_3^T \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{y} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \rightarrow 1, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \rightarrow 2, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \rightarrow 3$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \rightarrow \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

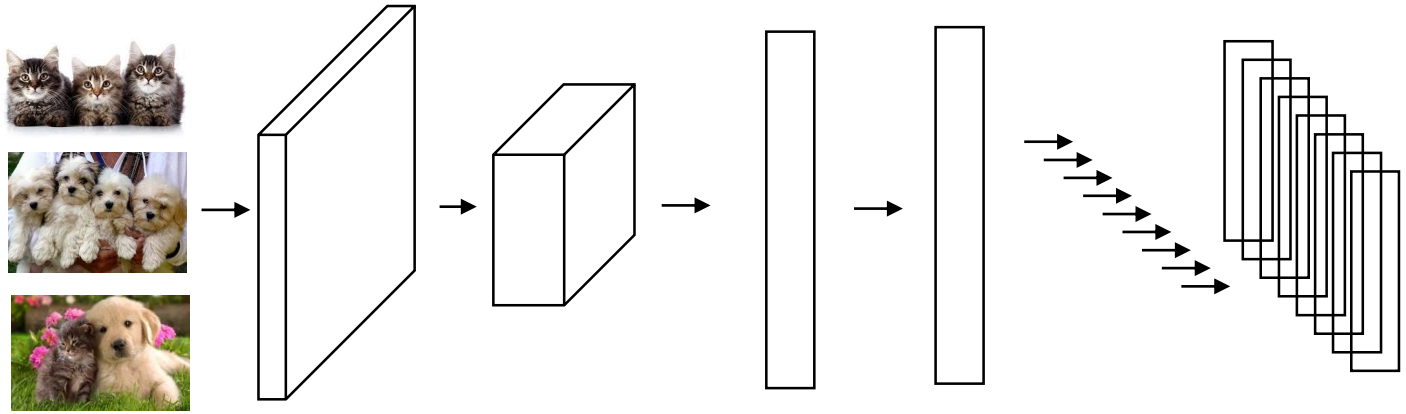
Multiclass, Single Object Detection Using ConvNet

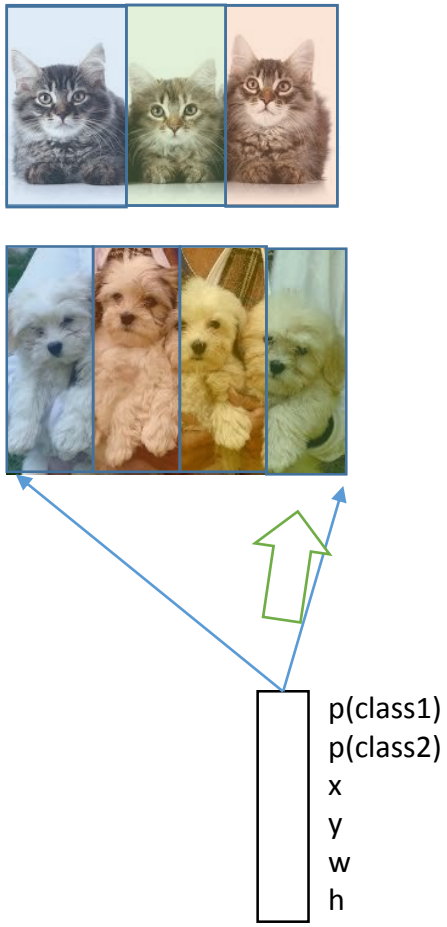


What About Multiclass, Multi-Object Detection?

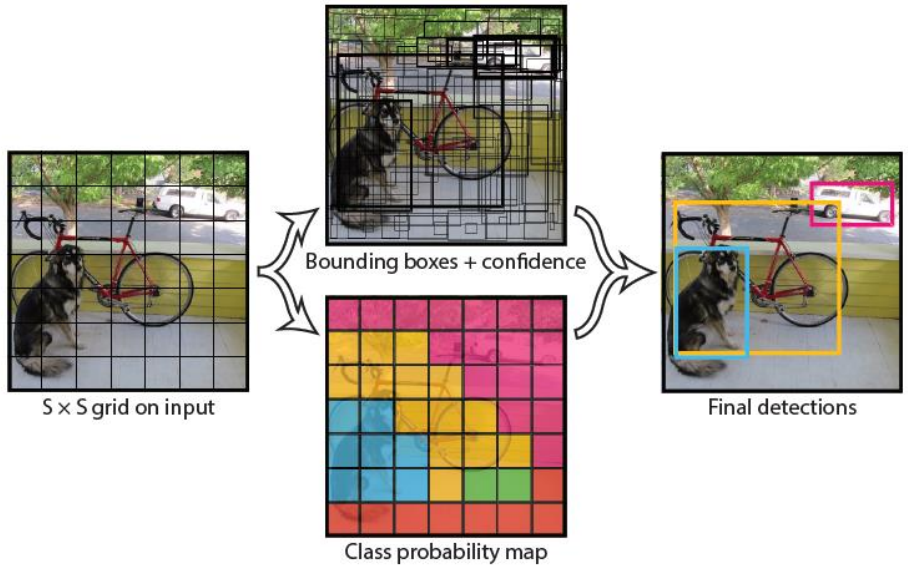
Multiclass, Multi-Object Detection Using ConvNet

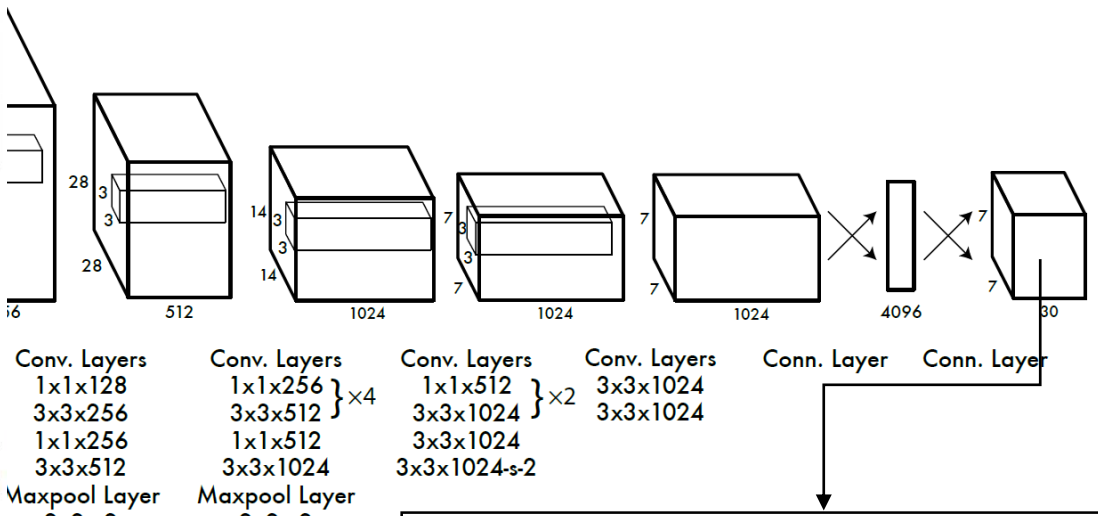
- Split image into cells.
- For each cell, train a detector that is responsible for detecting objects around that cell.
- The receptive field of a detector is still the whole image
- Objects number and positions are not fixed; use more cells than needed





The YOLO Object Detection Model





C1, x1, y1, w1, h1, C2, x2, y2, w2, h2, p(c1), p(c2), …., p(c20)

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.

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 $7 \times 7 \times 30$ tensor.

Divide the image into 7x7 cells.
Each cell trains a detector.
The detector needs to predict the object's class distributions.
The detector has 2 bounding-box predictors to predict bounding-boxes and confidence scores.

The YOLO Object Detection Model - Training

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.

C1, x1, y1, w1, h1, C2, x2, y2, w2, h2, p(c1), p(c2), …., p(c20)

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

Before the training process each object in ground-truth is assigned to a cell for its detection.
At training time, a detector assign one of its bounding-box predictors to be responsible for detecting the ground-truth object.

Summary

- YOLO is a very fast and accurate multiclass multi-object detection algorithm.
- It divide image into cell; for each cell it trains a detector responsible for detecting objects around that cell.
- Its network has multiple outputs, each one of which corresponds to a detector trained for a cell.
- Implication
  - It’s possible to train a ConvNet model to pay attention to a particular area of an image
  - It’s possible to combine different perception tasks into the same network.

Thank You!

Questions?