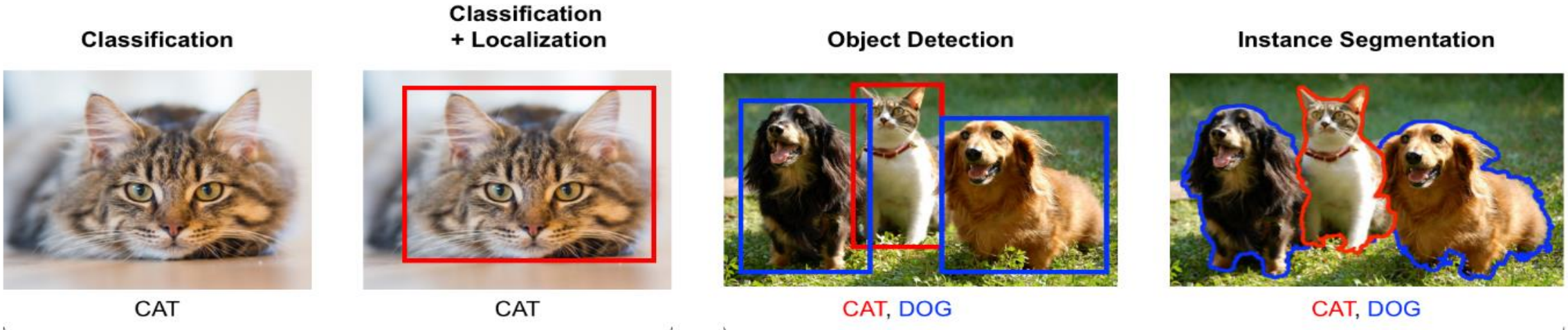
Object Detection as a Regression Problem using YOLOs

Contents

- Object Detection
- Challenges in Object Detection
- Regression Formulation
- Process
- Network Design
- Loss Function

Object Detection

- Object Classification, Segmentation, and Localization



Identify what the object is and where it is.

Goal: Detect multiple objects, identify their classes and locations.

Detect multiple objects and segment each one at the pixel level. (**actual shape**, not just a box.)

Challenges in Object Detection

- Multi-scale training
- Foreground-Background class imbalance
- Detection of relatively smaller objects
- Necessity of large datasets and computational power
- Smaller sized datasets
- Inaccurate localization during predictions

Pre-YOLO Era (Before 2015)

1. **Traditional Methods: Sliding Window + Classifiers**

Idea:

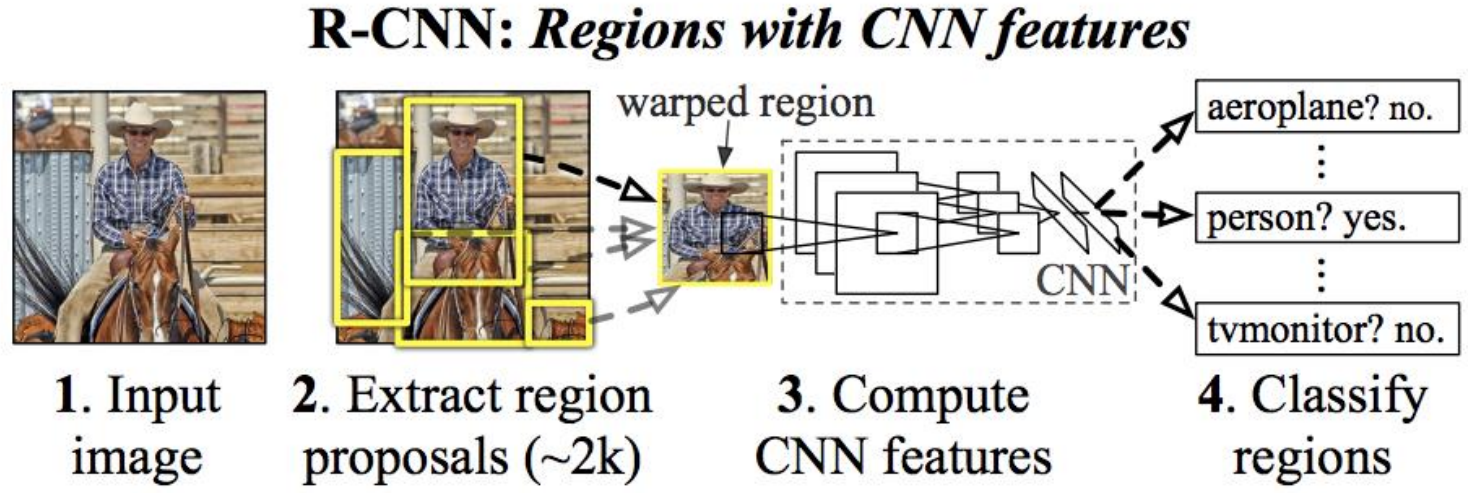
- Take an image. Slide a small window across **every possible location** at **every possible scale**.
- For each window, use a **classifier** (like an SVM or Adaboost) to predict if an object is present.
- Before deep learning, features were **handcrafted** (not learned from data).
- **Popular feature extractors:**
- Histogram of Oriented Gradients (HOG) (Dalal and Triggs, 2005). SIFT, SURF for keypoint detection.
- **Extremely slow:** Had to classify thousands/millions of windows per image.
- **Poor accuracy:** Hard to distinguish between background and object windows.
- **Fixed features:** Feature design was manual and limited.

Pre-YOLO Era (Before 2015)

2. R-CNN (2014)

Idea:

- **Region Proposal + CNN classification**:
- First, use **Selective Search** to propose ~2,000 candidate regions that *might* contain objects.
- Then, for each region, **warp it to a fixed size** and **pass it through a CNN** (like AlexNet) to extract features.
- Classify each region with an SVM.
- Very slow (2,000 regions processed separately by CNN).
- Multi-stage pipeline (region proposal → feature extraction → classification → bounding box regression)

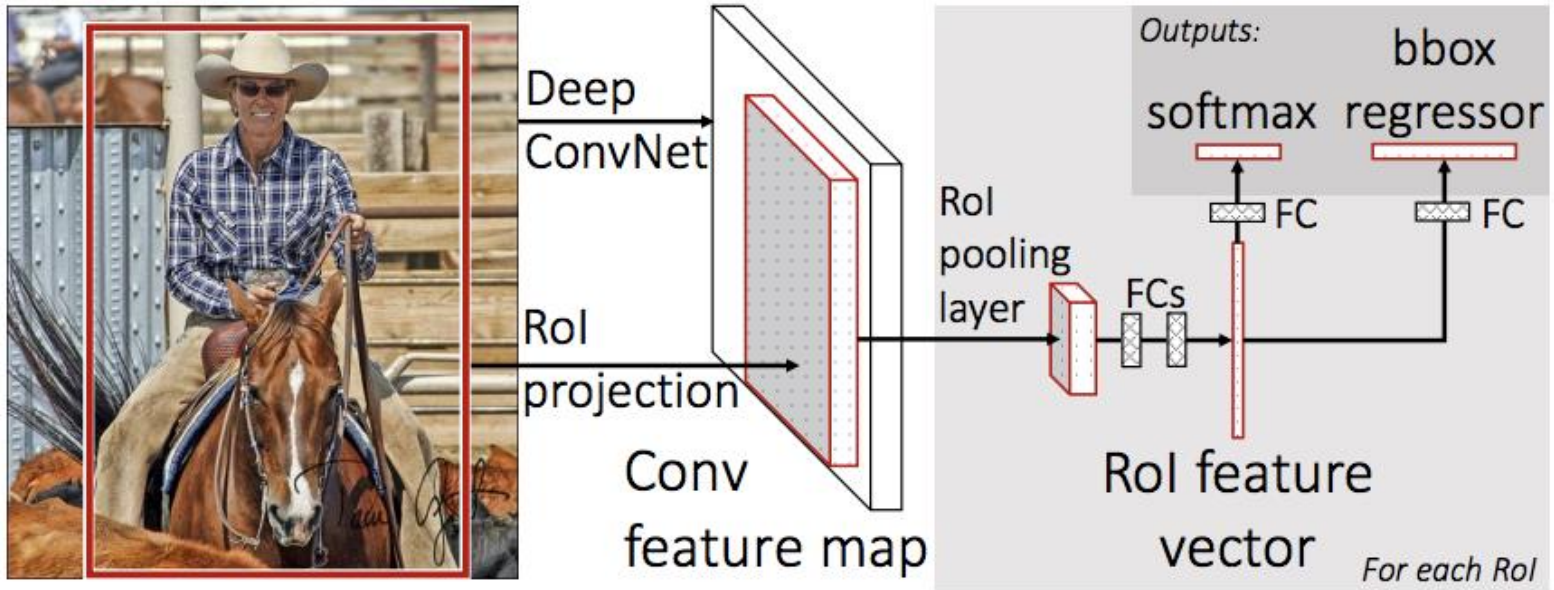


Pre-YOLO Era (Before 2015)

3. Fast R-CNN (2015)

Idea:

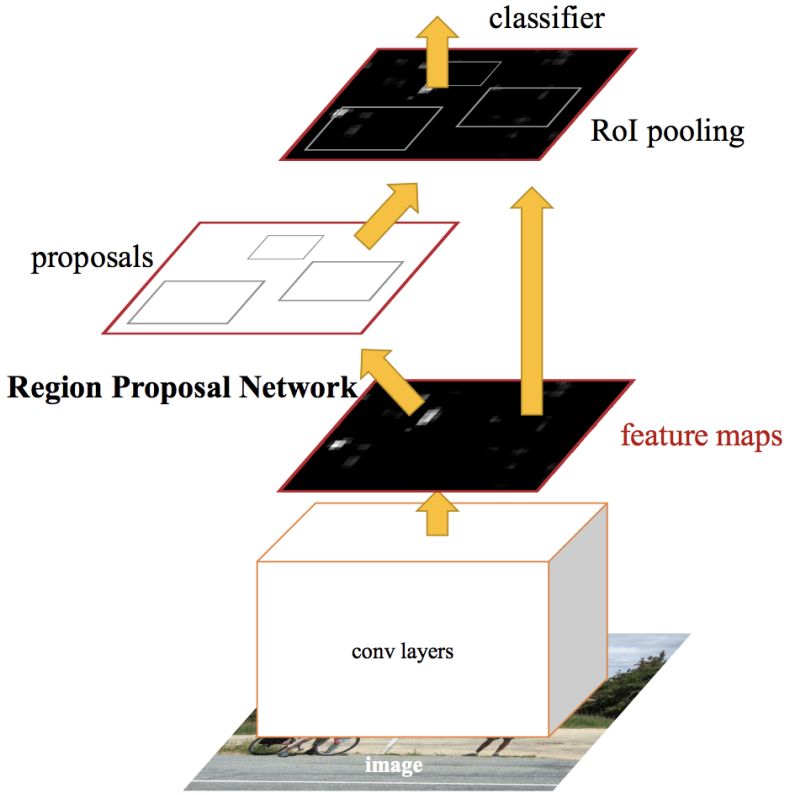
- Run one CNN over the full image once to get a feature map.
- For each region proposal:
- Use RoI Pooling (box. Region of Interest Pooling) to crop and resize the region from the feature map.
- Then classify it and regress the bounding



The reason “Fast R-CNN” is faster than R-CNN is because **don’t have to feed 2000 region** proposals to the convolutional neural network every time. Instead, the convolution operation is done only once per image and a feature map is generated from it.

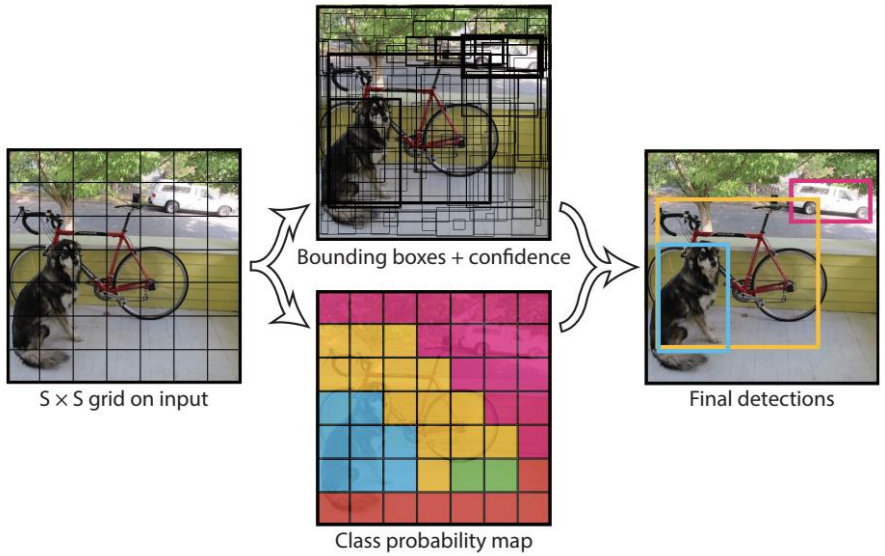
Pre-YOLO Era (Before 2015)

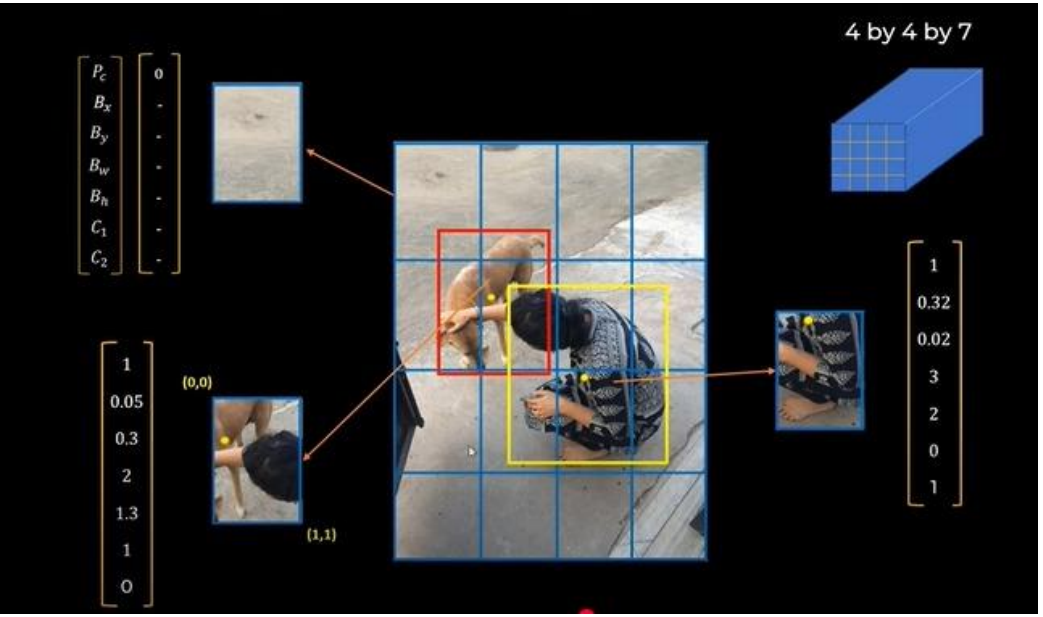
- **Faster R-CNN (2015)**
- **Region Proposal Network (RPN)**:
- Instead of using Selective Search, the RPN **learns to propose regions** using a small network on top of feature maps.
- Trained jointly with the object detector.



- **Instead of using selective search algorithm** on the feature map to identify the region proposals, a **separate network is used to predict the region proposals.**
- The predicted region proposals are then reshaped using a **RoI pooling layer** which is then used to **classify** the image within the proposed region **and predict the offset** values for the bounding boxes.

Yolo



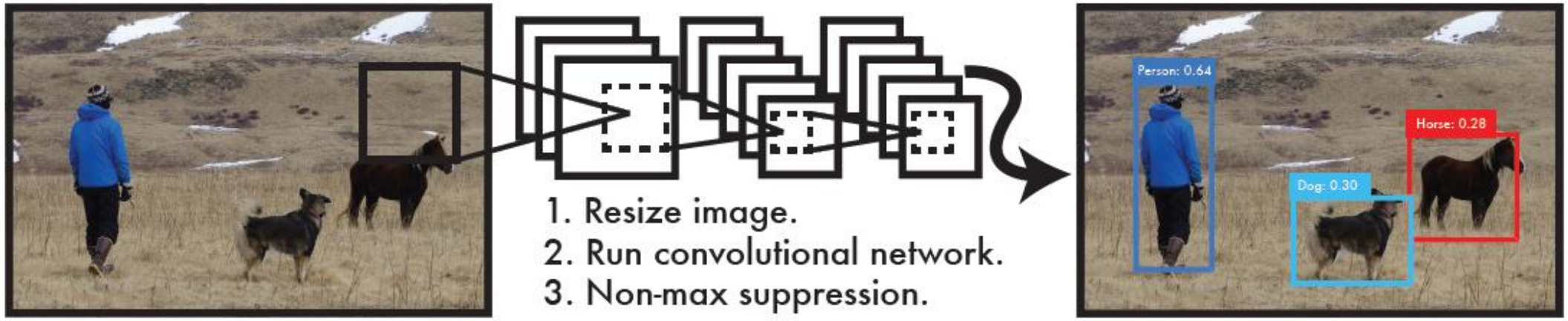


- In YOLO a single convolutional network predicts the bounding boxes and the class probabilities for these boxes.

- Take an image and split it into an SxS grid,
- within each of the grid we take m bounding boxes.
- For each of the bounding box, the network outputs a class probability and offset values for the bounding box. The bounding boxes having the class probability above a threshold value is selected and used to locate the object within the image.

Regression Formulation using YOLO

- Neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. (Given the **entire image as input**, YOLO's neural network outputs all **bounding boxes + class probabilities in one pass.**)
- Processes images in real-time at 45 frames per second.
- Fast YOLO, processes an astounding 155 frames per second.



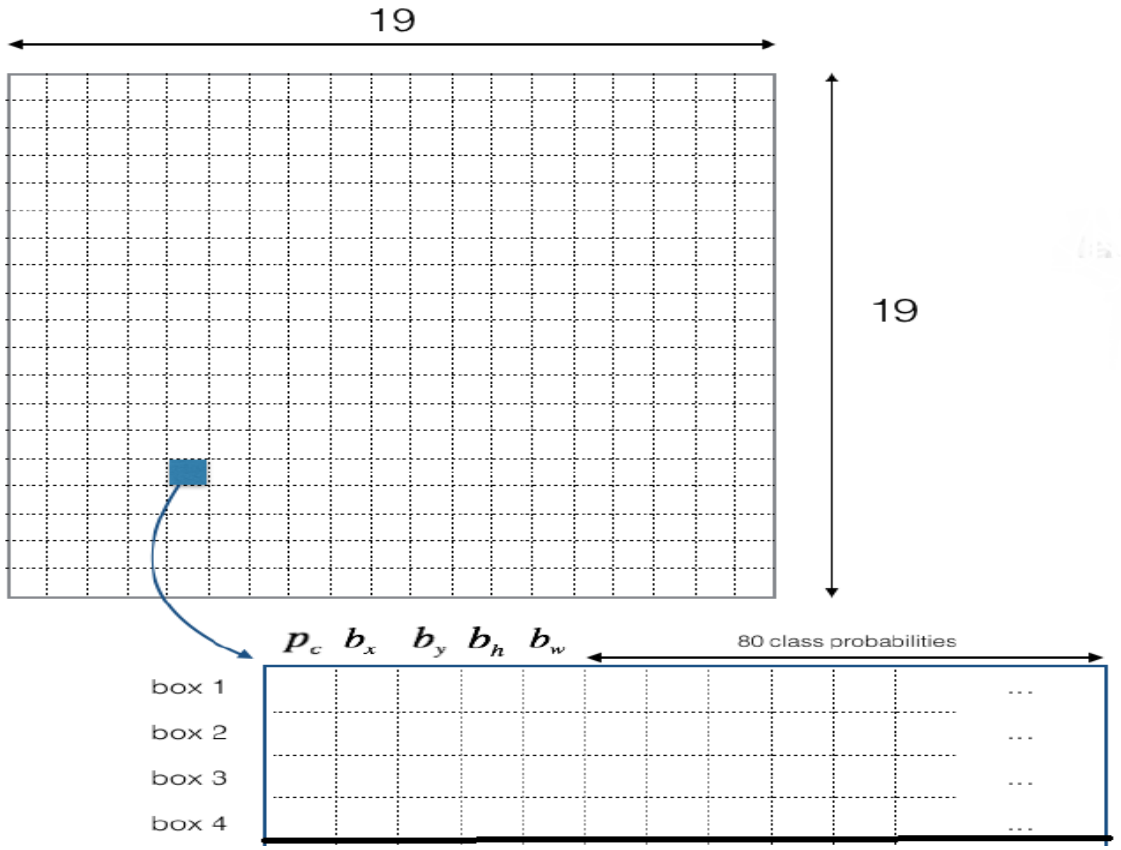
Removes redundant and overlapping boxes.
Keeps the box with the highest confidence score.

Regression Formulation using YOLO

- You only look once (YOLO) at an image to predict what objects are present and where they are.
- Advantages:
  - YOLO is refreshingly simple
  - YOLO is extremely fast.
  - YOLO achieves more than twice the mAP of other real-time systems.
  - YOLO reasons globally about the image when making predictions.
  - YOLO learns generalizable representations of objects.
- Unlike sliding window and region proposal-based techniques, YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance.

Unified Architecture

- It also predicts all bounding boxes across all classes for an image simultaneously.
- This means our network reasons globally about the full image and all the objects in the image.
- The YOLO design enables end-to-end training and real-time speeds while maintaining high average precision.

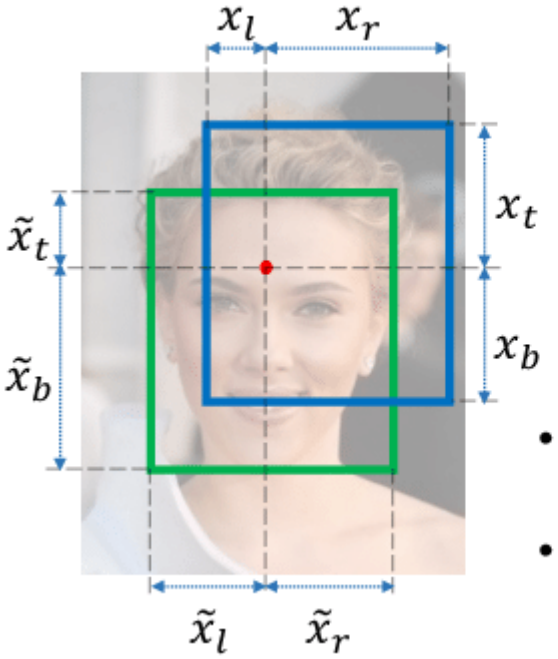


Process

- System divides the input image into an SxS grid.
- If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object.
- Each grid cell predicts B bounding boxes and confidence scores for those boxes.
- These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts.

$$\Pr(\text{Object}) * \bar{\text{IOU}}_{\text{pred}}^{\text{truth}}$$

Intersection over Union

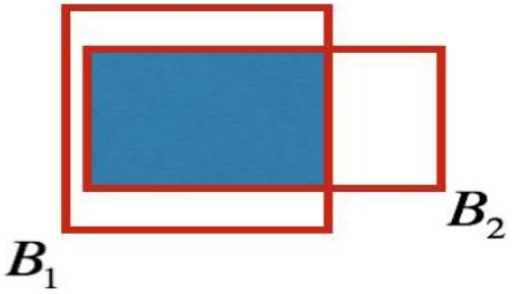


Ground truth: $\tilde{x} = (\tilde{x}_t, \tilde{x}_b, \tilde{x}_l, \tilde{x}_r)$

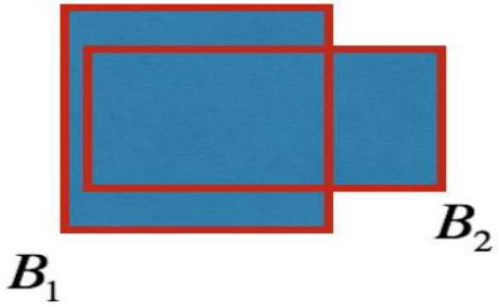
Prediction: $x = (x_t, x_b, x_l, x_r)$

- $\ell_2 \ \ loss = ||\square - \square||_2^2$
- $IoU\ loss = -\ln \dfrac{Intersection(\square, \square)}{Union(\square, \square)}$

Intersection

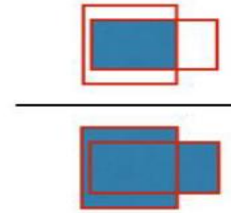


Union



Intersection over Union

$$IoU = \frac{B_1 \cap B_2}{B_1 \cup B_2} =$$



Process

- Each bounding box consists of 5 predictions: x, y, w, h, and confidence

- Each grid cell also predicts C conditional class probabilities.

$$\Pr(\text{Class}_i|\text{Object})$$

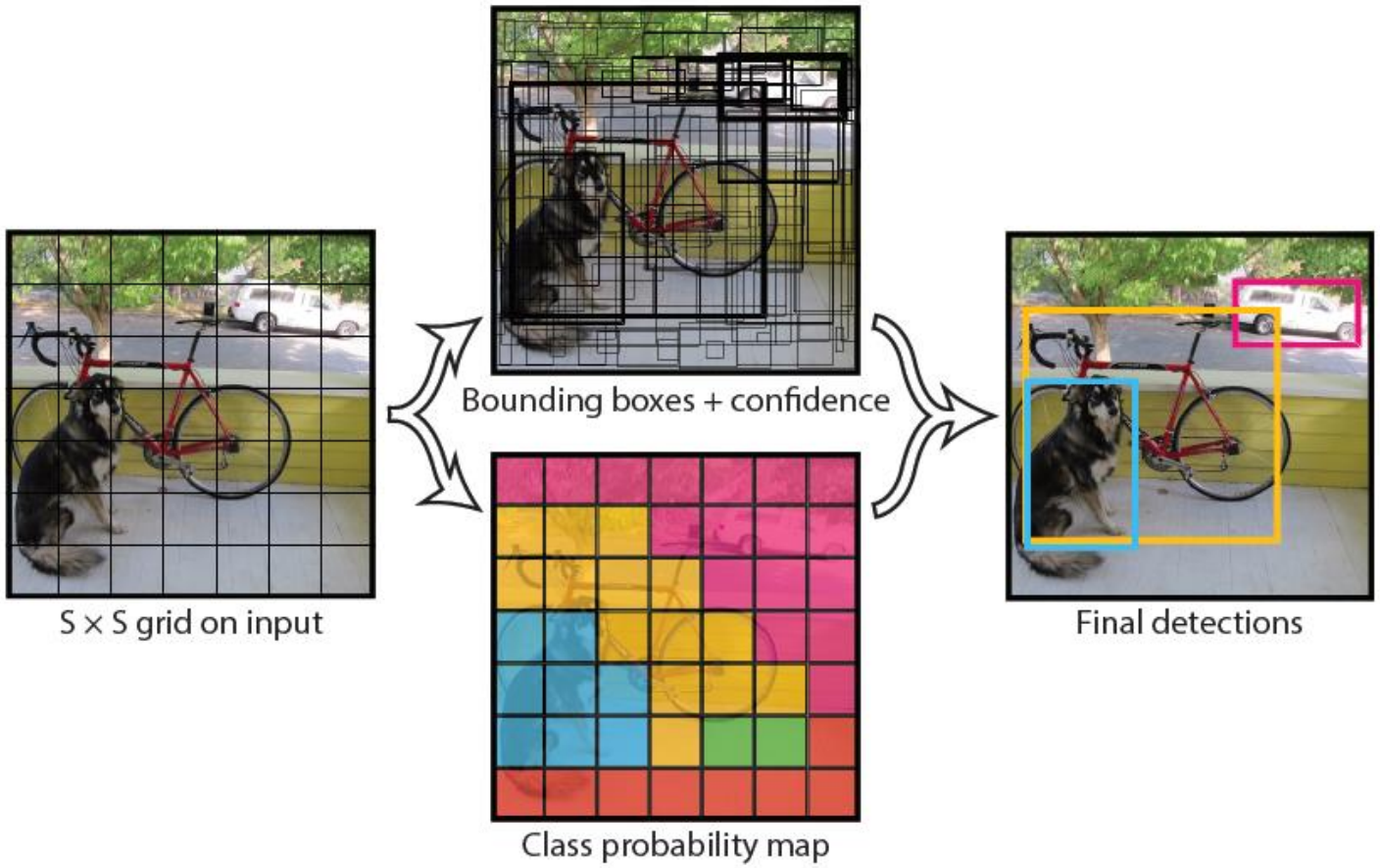
- One set of class probabilities per grid cell, regardless of the number of boxes B

- It divides the image into an S x 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 x S x (B x 5 + C) tensor.

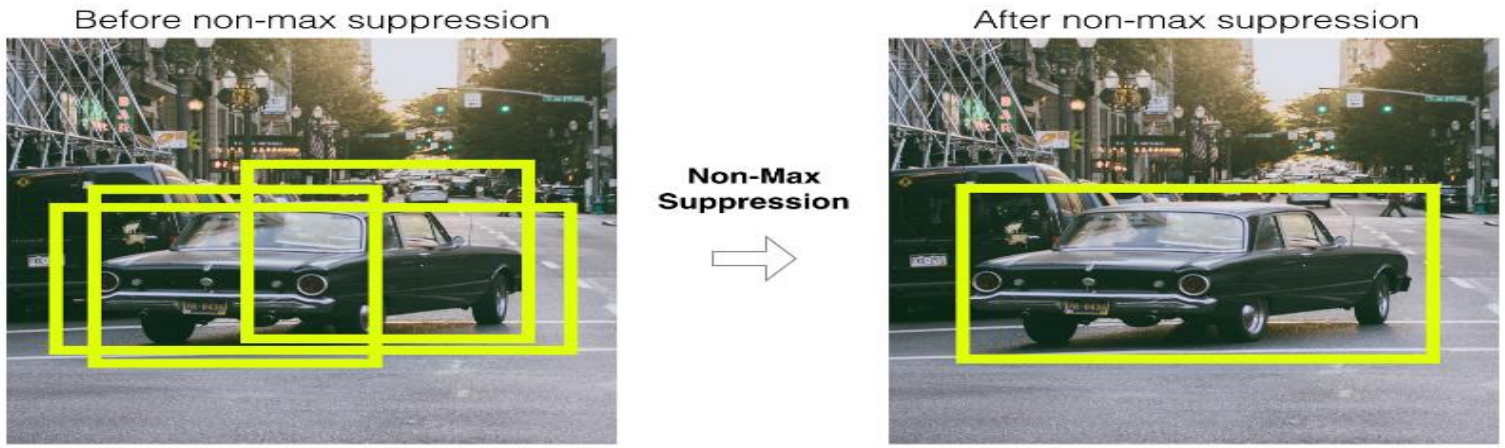
- YOLO on PASCAL VOC, we use S = 7, B = 2. PASCAL VOC has 20 labeled classes so C = 20. Our final prediction is a 7 x 7 x 30 tensor

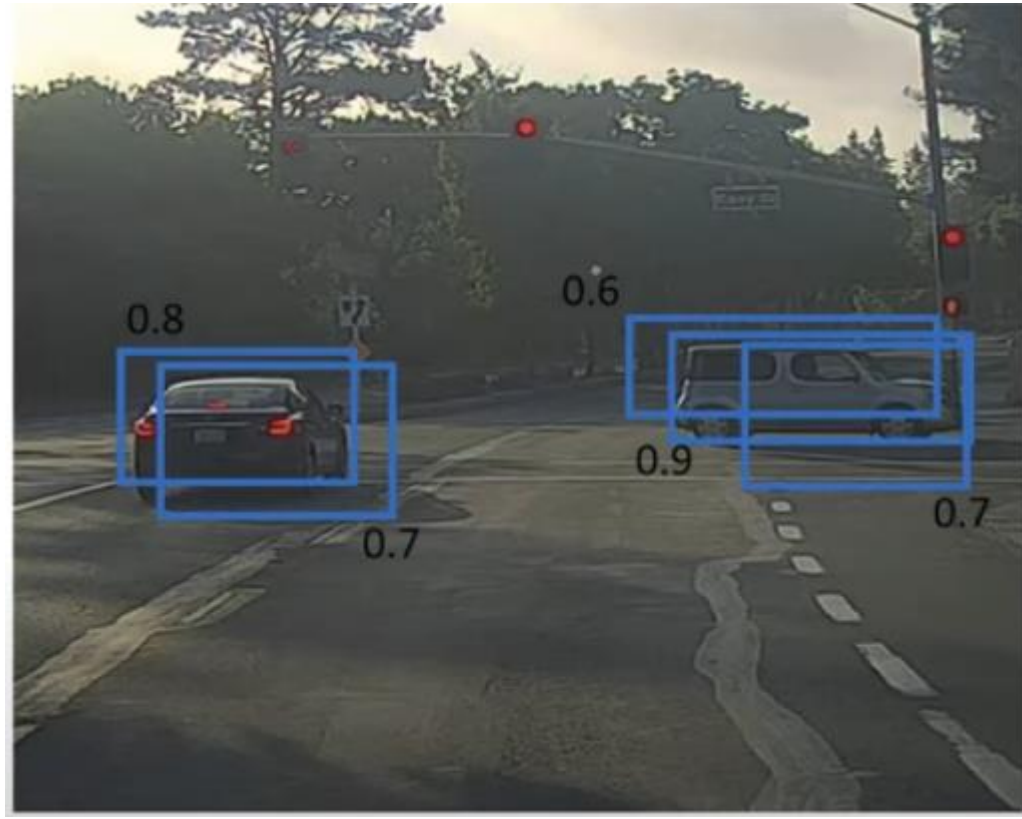
$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

Process


Bounding boxes + confidence
S × S grid on input
Class probability map
Final detections

Non-Max Suppression





Network Design

- Implement this model using CNN for the VOC pascal dataset.
- The initial convolutional layers of the network extract features from the image while the fully connected layers predict the output probabilities and coordinates
- Network architecture is inspired by the GoogLeNet model for image classification
- Our network has 24 convolutional layers followed by 2 fully connected layers.
- Instead of the inception modules used by GoogLeNet, we simply use 1x1 reduction layers followed by 3x3 convolutional layers
- Fast YOLO uses a neural network with fewer convolutional layers (9 instead of 24) and fewer filters in those layers.

Network Design

- Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1x1 convolutional layers reduce the features space from preceding layers.

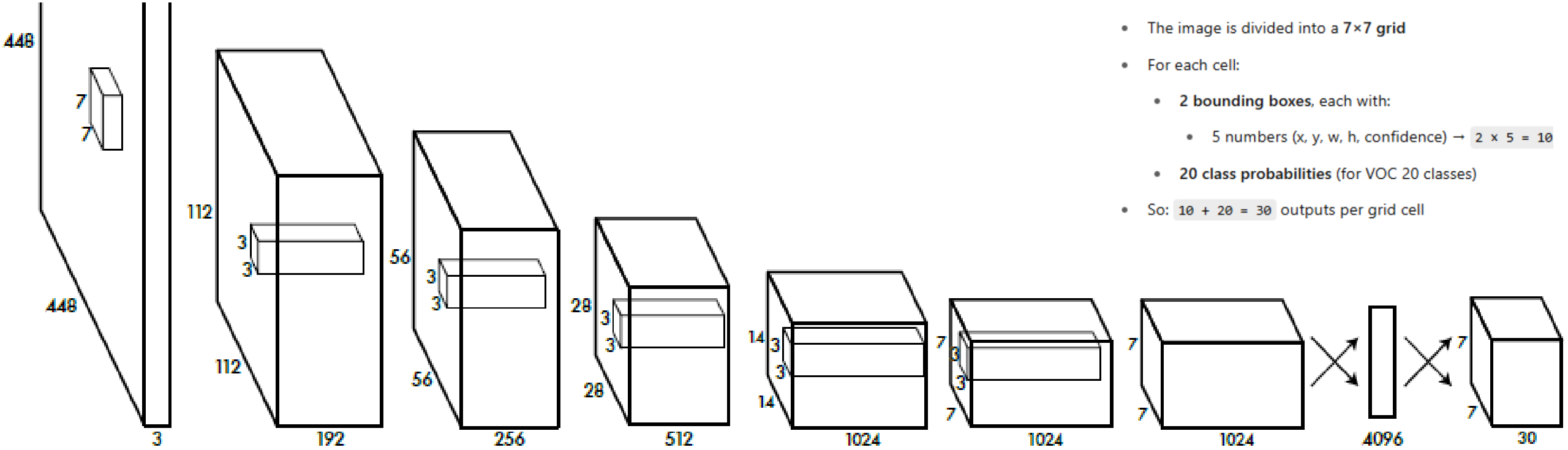
- We pre-train our convolutional layers on the ImageNet 1000-class competition dataset on first 20 convolutional layers at half the resolution (224x224 input image) and then double the resolution for detection.

- We then convert the model to perform detection.

- Adding both convolutional and connected layers to pretrained networks can improve performance.

- we add four convolutional layers and two fully connected layers with randomly initialized weights.

Network Design



- The image is divided into a **7×7 grid**
- For each cell:
  - **2 bounding boxes**, each with:
    - 5 numbers (x, y, w, h, confidence) → `2 × 5 = 10`
  - **20 class probabilities** (for VOC 20 classes)
- So: `10 + 20 = 30` outputs per grid cell

Conv. Layer
7x7x64-s-2
Maxpool Layer
2x2-s-2

Conv. Layer
3x3x192
Maxpool Layer
2x2-s-2

Conv. Layers
1x1x128
3x3x256
1x1x256
3x3x512
Maxpool Layer
2x2-s-2

Conv. Layers
1x1x256
3x3x512
} ×4
1x1x512
3x3x1024
Maxpool Layer
2x2-s-2

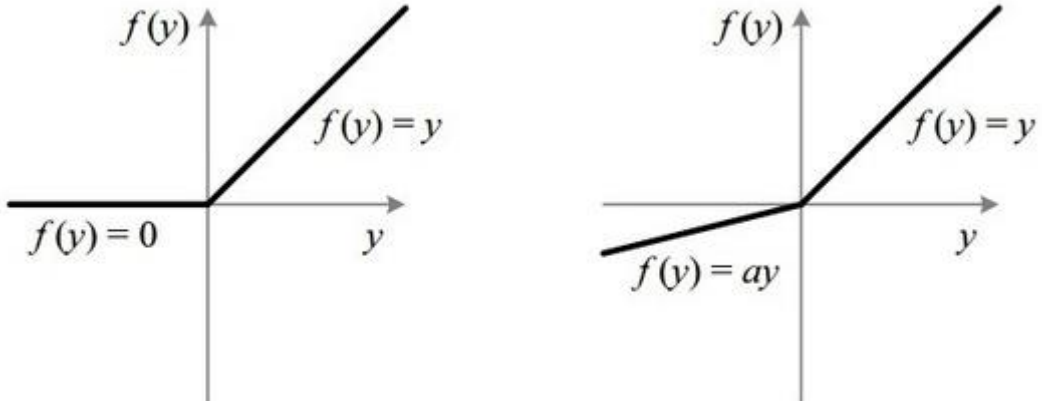
Conv. Layers
1x1x512
3x3x1024
} ×2
3x3x1024
3x3x1024-s-2

Conv. Layers
3x3x1024
3x3x1024

Conn. Layer

Conn. Layer

Training



- We pretrain our convolutional layers on the ImageNet 1000-class competition dataset. Pretraining helps the model learn useful visual features like edges, textures, shapes, etc.
- We then convert the model to perform detection.
- Detection often requires fine-grained visual information so we increase the input resolution of the network from 224x224 to 448x448.
- Our final layer predicts both class probabilities and bounding box coordinates.
- We use a linear activation function for the final layer and all other layers use leaky ReLU as activation function.

Loss Function

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

- Localization Loss: Center Coordinates.
- Measures how far off the predicted center coordinates (x, y) are from the true ones.
- Only calculated if an object is present in the grid cell 1ijobj.

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

•Square root is used to reduce the impact of large boxes (balances large vs small object errors).

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

Confidence Loss: When Object Exists
Confidence C: IoU between predicted box and ground truth.

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

Penalizes the network if it predicts high confidence when there is no object.

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

For each grid cell that contains an object, it penalizes incorrect class probabilities.

Loss Function

- we increase the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that don't contain objects.
- Sum-squared error also equally weights errors in large boxes and small boxes. Our error metric should reflect that small deviations in large boxes matter less than in small boxes.

Component	What it penalizes	Purpose
Coord (x, y)	Wrong box center	Precise location
Coord (w, h)	Wrong size	Proper fit
Confidence (obj)	Wrong object presence score when there is an object	Trustworthy predictions
Confidence (noobj)	Wrong confidence when there's no object	Avoid false positives
Class prediction	Incorrect class labels	Proper identification

Testing

- Predicting detections for a test image only requires one network evaluation.
- On PASCAL VOC the network predicts 98 bounding boxes per image and class probabilities for each box.
- Non-maximal suppression can be used to fix these multiple detections.

Limitation of YOLO

- Model struggles with small objects that appear in groups, such as flocks of birds.
- It struggles to generalize to objects in new or unusual aspect ratios or configurations.
- Main source of error is incorrect localizations.