NAAMII | KING'S COLLEGE architecting future | Ai4GROWTH

Paper Reading Session

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

Date: January 17, 2024 (Wednesday)
Time: 7:00 PM NST

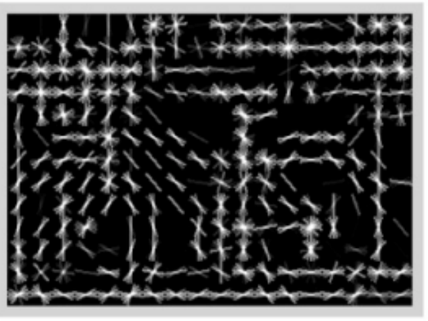
You Only Look Once: Unified, Real-Time Object Detection

- *Joseph Redmon ,*
- *Santosh Divvala,*
- *Ross Girshick,*
- *Ali Farhadi*

University of Washington , Allen Institute for AI , Facebook AI Research

History of Object detection

Before Deep learning: Traditional Approaches

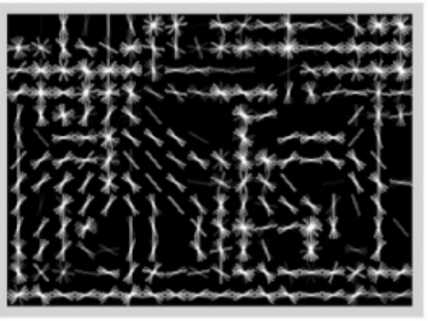


- **Viola Jones Detectors(2001):**
  - Detection of human faces in real time, uses *Haar features*
- **HOG Detectors(2005):**
  - An improvement in the scale invarient feature transform.
- **Deformable Part-based Model (DPM):**
  - Proposed by P. Felzenszwalb in **2008** as an extension of the *HOG detector*.
  - *Trained using a variant of the Support Vector Machine (SVM) algorithm known as **Latent SVM***.
  - the winners of *VOC-07, -08, and -09* detection challenges

A typical DPM detector consists of a root-filter and a number of part-filters. Instead of manually specifying the configurations of the part filters (e.g., size and location), a weakly supervised learning method is developed in DPM where all configurations of part filters can be learned automatically as latent variables. R. Girshick has further formulated this process as a special case of Multi-Instance learning [39], and some other important techniques such as "hard negative mining", "bounding box regression", and "context priming" are also applied for improving detection accuracy (to be introduced in Section 2.3). To speed up the detection,

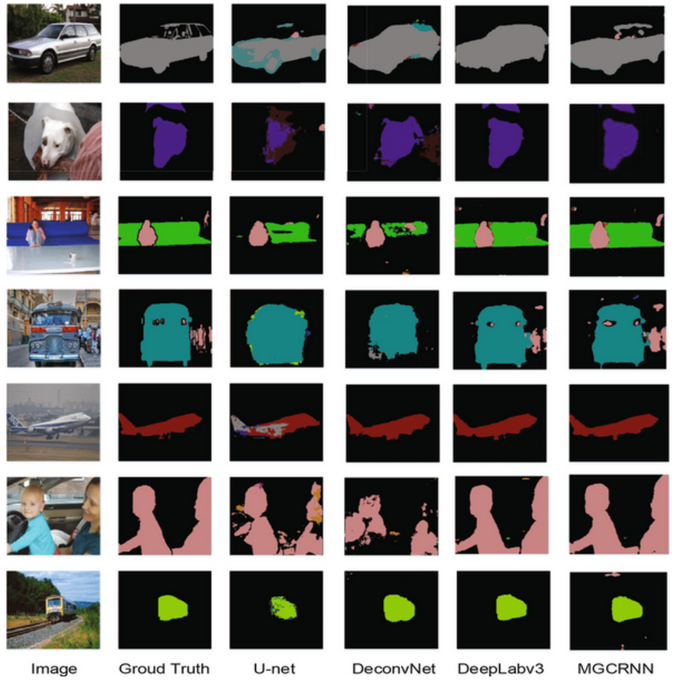
https://arxiv.org/pdf/1905.05055v2.pdf

PASCAL VOC Dataset



PASCAL Visual Object Classes (VOC) 2012 dataset contains **20 object** categories

Each image in this dataset has **pixel-level segmentation** annotations, **bounding box** annotations, and **object class** annotations



PASCAL VOC Challenge: 2005 onwards till 2012.

http://host.robots.ox.ac.uk/pascal/VOC/index.html

MS-COCO Dataset

The dataset contains 91 objects types of 2.5 million labeled instances across 328,000 images.

5 Dataset Statistics

Next, we analyze the properties of the Microsoft Common Objects in COntext (MS COCO) dataset in comparison to several other popular datasets. These include ImageNet [1], PASCAL VOC 2012 [2], and SUN [3]. Each of these datasets varies significantly in size, list of labeled categories and types of images. ImageNet was created to capture a large number of object categories, many of which are fine-grained. SUN focuses on labeling scene types and the objects that commonly occur in them. Finally, PASCAL VOC's primary application is object detection in natural images. MS COCO is designed for the detection and segmentation of objects occurring in their natural context.

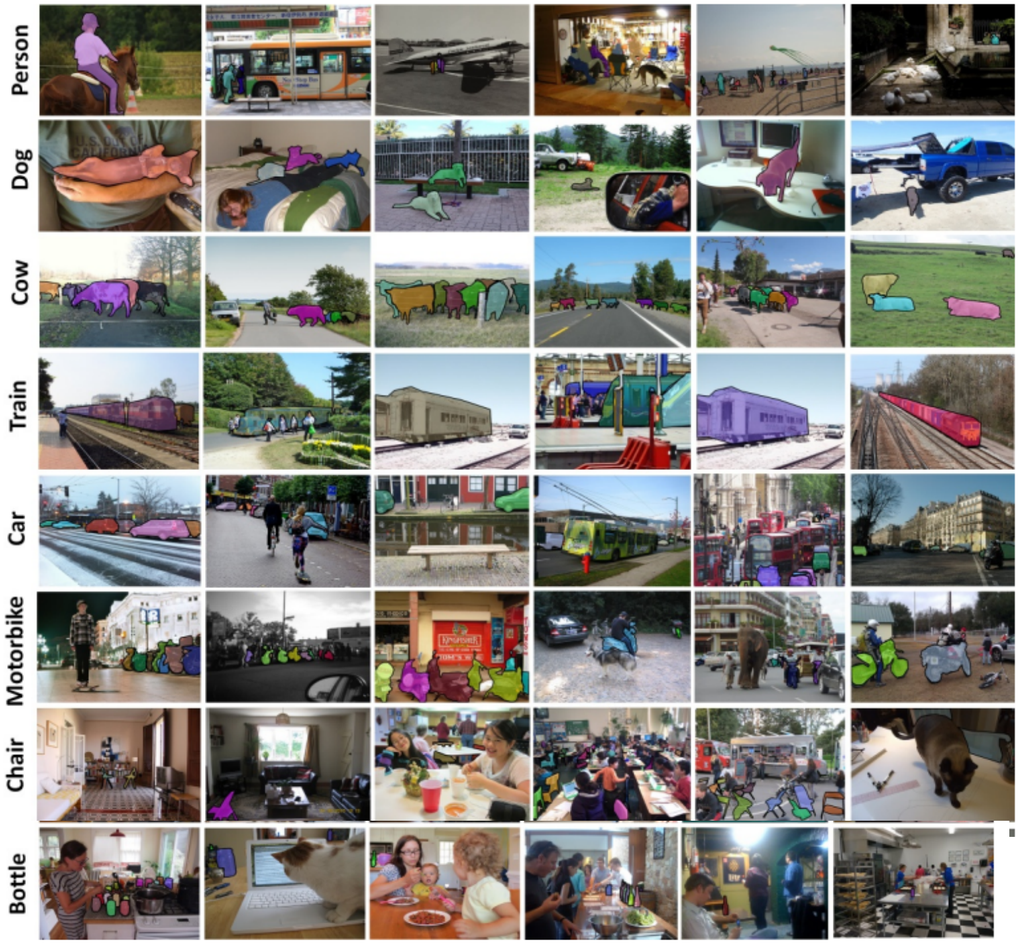
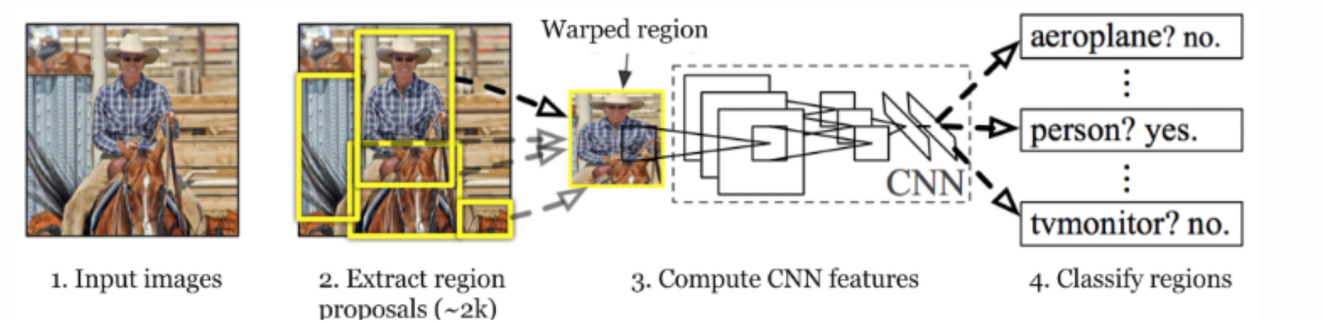


Fig. 6: Samples of annotated images in the MS COCO dataset.

https://arxiv.org/pdf/1405.0312.pdf

History of Object detection

Neural Network approach: **Two Stage Detectors**



- **Region Proposal Networks:**
  - R-CNN, Fast R-CNN, Faster R-CNN
  - Based on Sliding Window Techniques
  - Used Classification algorithm to classify each window in an image

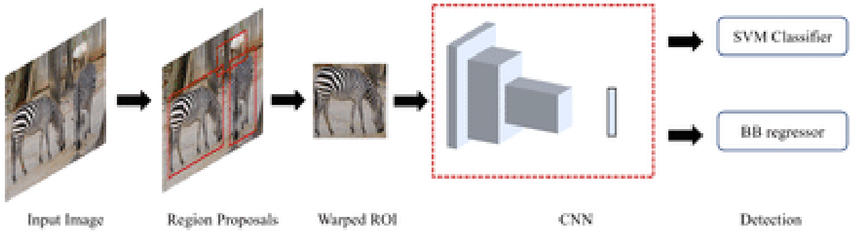
Extraction of a set of object proposals

Then each proposal is rescaled to a fixed size image and fed into a CNN model pretrained(resnet, alexnet, etc...) model to extract features.

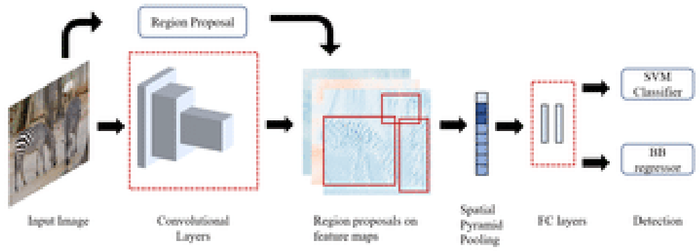
Used linear SVM classifiers to predict the presence of an object within each region

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

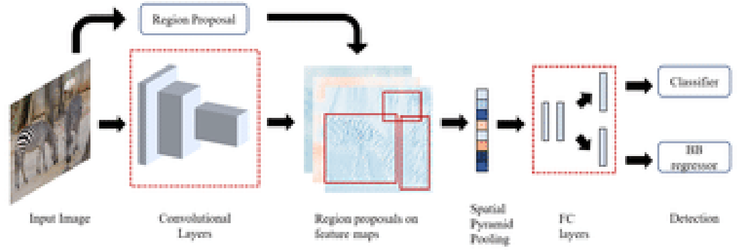
RCNN



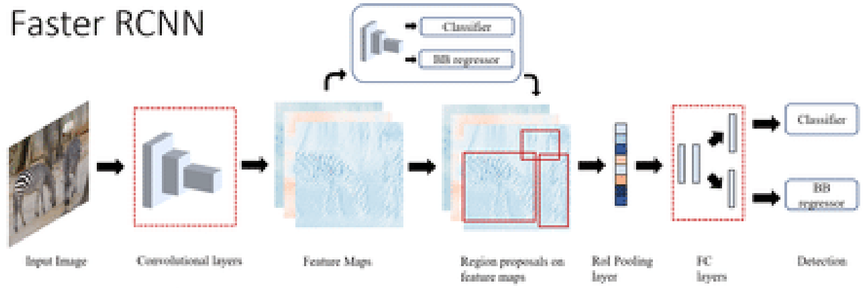
SPP-Net



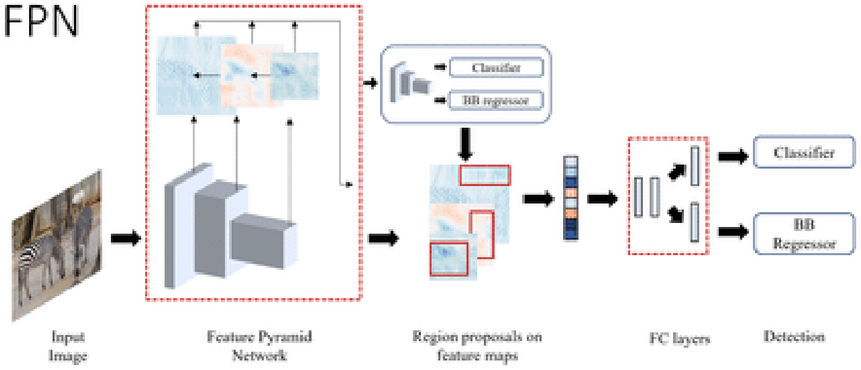
Fast RCNN



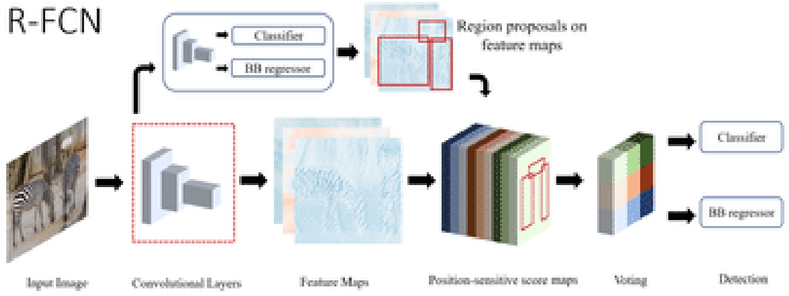
Faster RCNN



FPN



R-FCN



https://www.researchgate.net/figure/Illustration-of-the-internal-architecture-of-different-two-stage-object-detectors_fig6_351105210

History of Object detection

Neural Network approach: **Two Stage Detectors**

- **Region Proposal Networks Drawbacks:**
  - the redundant feature computations on a large number of overlapped proposals (*over 2000 boxes from one image*) lead to an extremely slow detection speed (14s per image with GPU)
  - Its a multi-stage detector
  -

Further improvement was on SPPNet and Fast RCNN and Faster RCNN

SPPNet: In 2014, K. He *et al.* proposed Spatial Pyramid Pooling Networks (SPPNet) [17]. Previous CNN models require a fixed-size input, e.g., a 224x224 image for AlexNet [35]. The main contribution of SPPNet is the introduction of a Spatial Pyramid Pooling (SPP) layer, which enables a CNN to generate a fixed-length representation regardless of the size of the image/region of interest without rescaling it. When using SPPNet for object detection, the feature maps can be computed from the entire image only once, and then fixed-length representations of arbitrary regions can be generated for training the detectors, which avoids repeatedly computing

History of Object detection

Neural Network approach: **Two Stage Detectors**

Further improvement was on SPPNet , Fast RCNN and Faster RCNN

How object proposals were calculated Earlier?

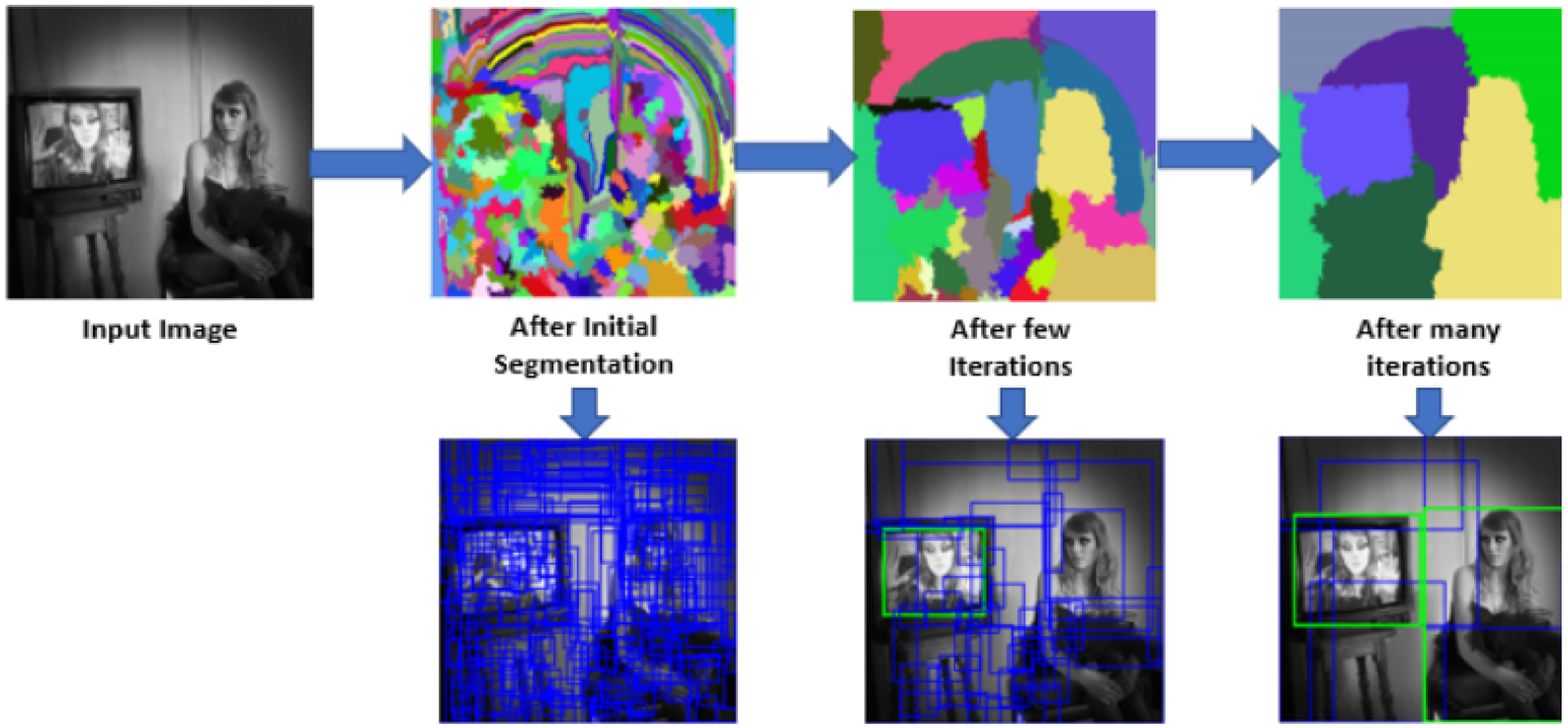
- **Selective Search Algorithm**

Fast RCNN: In 2015, R. Girshick proposed Fast RCNN detector [18], which is a further improvement of R-CNN and SPPNet [16, 17]. Fast RCNN enables us to simultaneously train a detector and a bounding box regressor under the same network configurations. On VOC07 dataset, Fast RCNN increased the mAP from 58.5% (RCNN) to 70.0% while with a detection speed over 200 times faster than R-CNN. Although Fast-RCNN successfully integrates the advantages of R-CNN and SPPNet, its detection speed is still limited by the proposal detection (see Section II-C1 for more details). Then, a question naturally arises: "can we generate object proposals with a CNN model?" Later, Faster R-CNN [19] answered this question.

Faster RCNN: In 2015, S. Ren *et al.* proposed Faster RCNN detector [19, 47] shortly after the Fast RCNN. Faster RCNN is the first near-realtime deep learning detector (COCO mAP@.5=42.7%, VOC07 mAP=73.2%, 17fps with ZF-Net [48]). The main contribution of Faster-RCNN is the introduction of Region Proposal Network (RPN) that enables nearly cost-free region proposals. From R-CNN to Faster RCNN, most individual blocks of an object detection system, e.g., proposal detection, feature extraction, bounding box regression, etc, have been gradually integrated into a unified, end-to-end learning framework. Although Faster RCNN breaks through the speed bottleneck of Fast RCNN, there is still computation redundancy at the subsequent detection stage. Later on, a

Selective Search Algorithm

Uses the segmented region proposals to generate candidate object locations



- Color Similarity
- Texture Similarity
- Size similarity
- Fill Similarity

http://vision.stanford.edu/teaching/cs231b_spring1415/papers/IJCV2004_FelzenszwalbHuttenlocher.pdf
https://www.geeksforgeeks.org/selective-search-for-object-detection-r-cnn/

Limitation of Two-Stage Detectors

- **Slower Inference Speeds:** Two-stage detectors often have slower inference speeds compared to single-stage detectors. This is because they require multiple inference stages per picture.
- **Increased Computational Complexity:** The complexity of two-stage detectors is higher due to the need for additional training steps. This can make them less efficient than single-stage detectors.
- **Dependence on Previous Works:** Highly dependent on previous works and mostly build on the previous pipeline as a baseline.

Single Stage Detectors

Examples:

- You Only Look Once(YOLO)
- Single Shot MultiBox Detector(SSD)
- RetinaNet, CornerNet, DETR, etc.

Abstract

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end

Single Stage Detectors

- First divides the input image into an S × S 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.

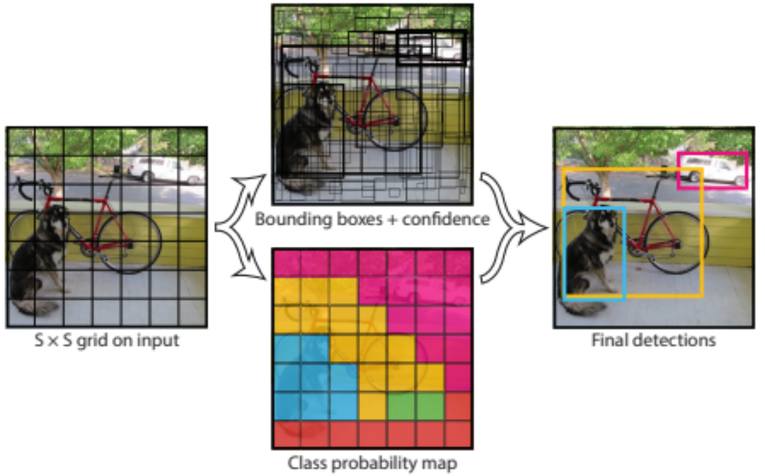


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

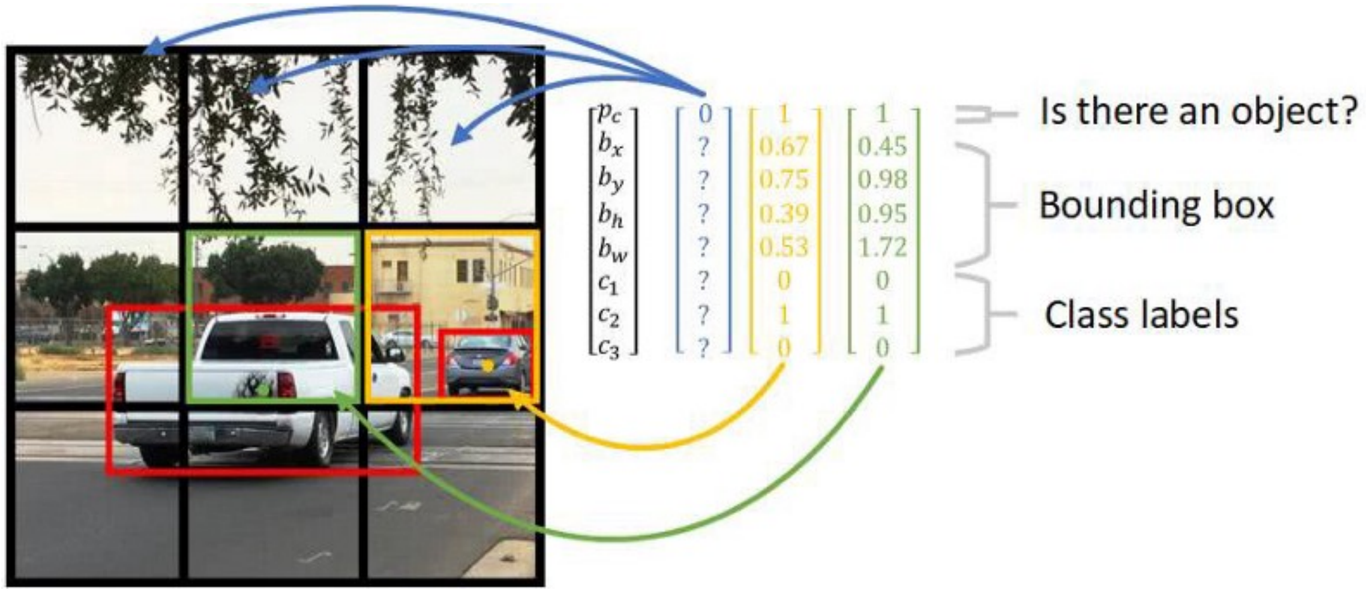
- **For evaluating YOLO on PASCAL VOC, the paper used S = 7, B = 2. PASCAL VOC has 20 labelled classes so C = 20.**
- **Final prediction is a 7 × 7 × 30 tensor.**

Single Stage Detectors

- First divides the input image into an S × S 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.



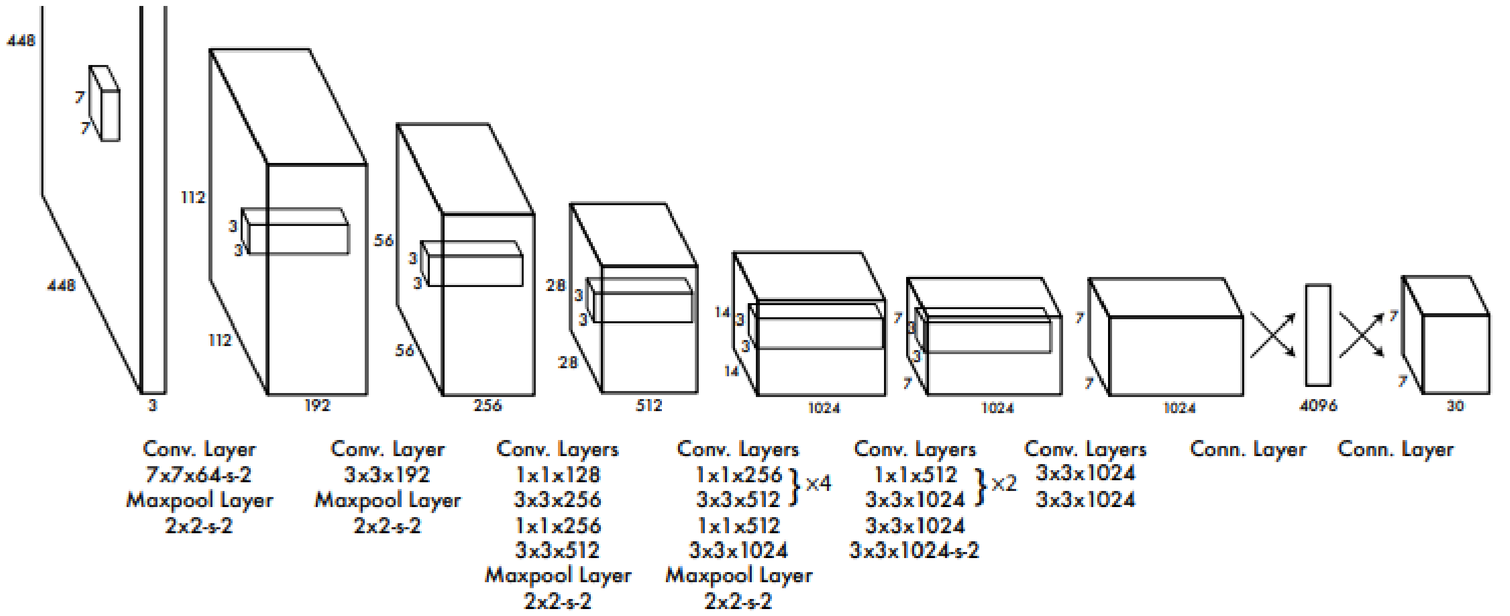


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

YOLO cost function: Includes

- **localization loss** for bounding box coordinates,
- **confidence loss** for object presence or absence, and
- **classification loss** for category prediction accuracy

loss function:

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

Localization loss

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

- λ_{coord}: Set to 5 to increase the loss of bounding box predictions
- x_i: GT bbox x-coordinate in the ith cell
- $\hat{x}_i$: Predicted bbox x-coordinate in the ith cell
- y_i: GT bbox y-coordinate in the jth cell
- $\hat{y}_i$: Predicted bbox y-coordinate in the ith cell
- Sum-squared error
- $\sum_{i=0}^{S^2}$: For each grid cell
- $\sum_{j=0}^{B}$: For each grid box
- $\mathbb{1}_{ij}^{obj}$: '1' if object appears in the ith cell and the jth box detect it, '0' otherwise

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

- w_i: GT bbox width in the ith cell
- $\hat{w}_i$: Predicted bbox width in the ith cell
- h_i: GT bbox height in the jth cell
- $\hat{h}_i$: Predicted bbox height in the ith cell
- $\sqrt{\ }$: Square root to reduce the range of the values

Confidence loss

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

- C_i: GT confidence score
- $\hat{C}_i$: Predicted confidence score
- Confidence error when an object is detected in the ith cell

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

- λ_{noobj}: Set to 0.5 to decrease the loss for empty boxes
- $\mathbb{1}_{ij}^{noobj}$: '1' if there is no object in the ith cell, '0' otherwise
- Confidence error when an object not detected in the ith cell

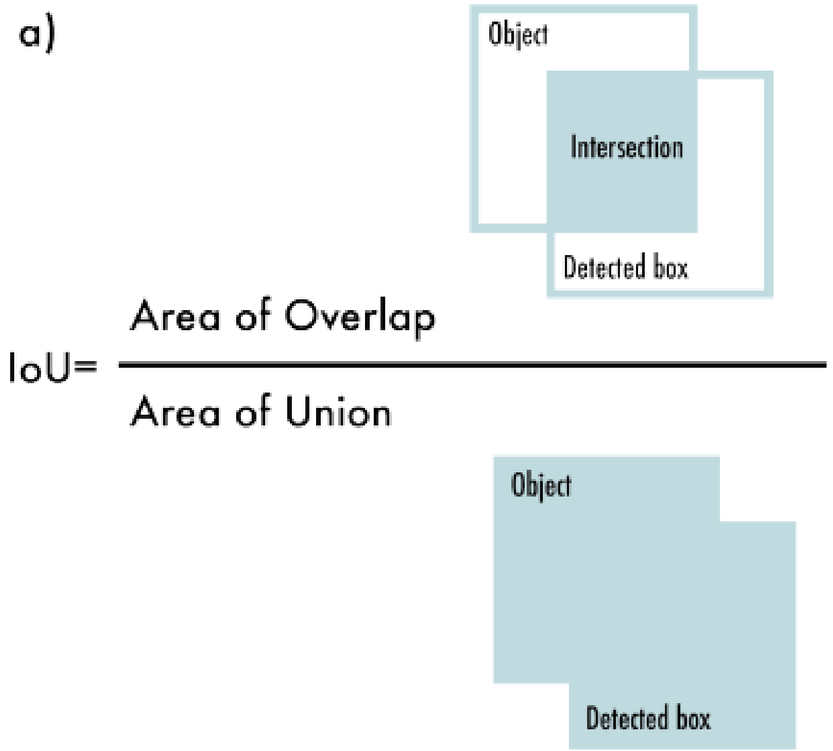
Clasification loss

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

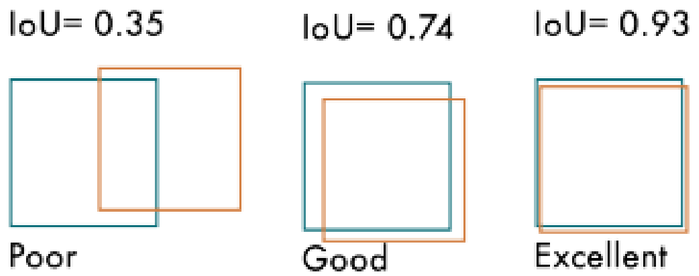
- $p_i(c)$: Predicted conditional probability of an object of class c appearing in the ith cell
- $\sum_{i=0}^{S^2}$: For each grid cell
- $\sum_{c \in classes}$: For each class
- $\hat{p}_i(c)$: GT conditional probability of class c appearing in the ith cell

IoU and mAP

a)



b)



IoU and mAP

AP metric uses the Intersection over Union (IoU) measure to assess the quality of the predicted bounding boxes

Calculated by taking the average of the precision scores for each class.

Non max Suppressor

3.3 Non-Maximum Suppression (NMS)

Non-Maximum Suppression (NMS) is a post-processing technique used in object detection algorithms to reduce the number of overlapping bounding boxes and improve the overall detection quality. Object detection algorithms typically generate multiple bounding boxes around the same object with different confidence scores. NMS filters out redundant and irrelevant bounding boxes, keeping only the most accurate ones. Algorithm 1 describes the procedure. Figure 3 shows the typical output of an object detection model containing multiple overlapping bounding boxes and the output after NMS.

Figure 3: Non-Maximum Suppression (NMS). a) Shows the typical output of an object detection model containing multiple overlapping boxes. b) Shows the output after NMS.

https://arxiv.org/pdf/2304.00501v1.pdf

Additional References

https://medium.com/analytics-vidhya/evolution-of-object-detection-582259d2aa9b

https://en.wikipedia.org/wiki/Object_detection

https://en.wikipedia.org/wiki/Scale-invariant_feature_transform

A Discriminatively Trained, Multiscale, Deformable Part Model((https://vision.ics.uci.edu/papers/FelzenszwalbMR_CVPR_2008/FelzenszwalbMR_CVPR_2008.pdf))

https://cs.brown.edu/courses/cs143/2011/lectures/DPM.pdf

https://www.geeksforgeeks.org/r-cnn-vs-fast-r-cnn-vs-faster-r-cnn-ml/

https://analyticsindiamag.com/r-cnn-vs-fast-r-cnn-vs-faster-r-cnn-a-comparative-guide/

Object Detection in 20 Years: A Survey((https://arxiv.org/pdf/1905.05055v2.pdf))

A COMPREHENSIVE REVIEW OF YOLO: FROM YOLOV1 TO YOLOV8 AND BEYOND((https://arxiv.org/pdf/2304.00501v1.pdf))