Project 3 - Neural networks for object detection

Work by: Yuval Vinokur 316402569,

Yarden Agami 207471186

Date: 12/2/22

Model chosen: Yolov3 ((9+6)%10=5)

YoloV3 (You Only Look Once, Version 3)-

First some background about Yolo:

In 2016 Redmon,Divvala Girschik and Farhadi revolutionized object detection with a paper titled: You Only Look Once: Unified, Real-Time Object Detection. In the paper they introduced a new approach to object detection – The feature extraction and object localization were unified into a single monolithic block.

Yolo is a family of deep learning models designed for fast object detection

The idea behind Yolo:

A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Sine the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

Bounding Boxes:

Yolo divides the input image into an SXS grid. Each grid cell predicts only one object by predicting a fixed number of boundary boxes.

(However, the one-object rule limits how close detected object can be. For that, Yolo does have some limitations on how close objects can be, meaning Yolo may miss objects that are too close)

For each grid cell:

- It predicts B boundary boxes and each box has one box confidence score.
- it detects one object only regardless of the number of boxes B

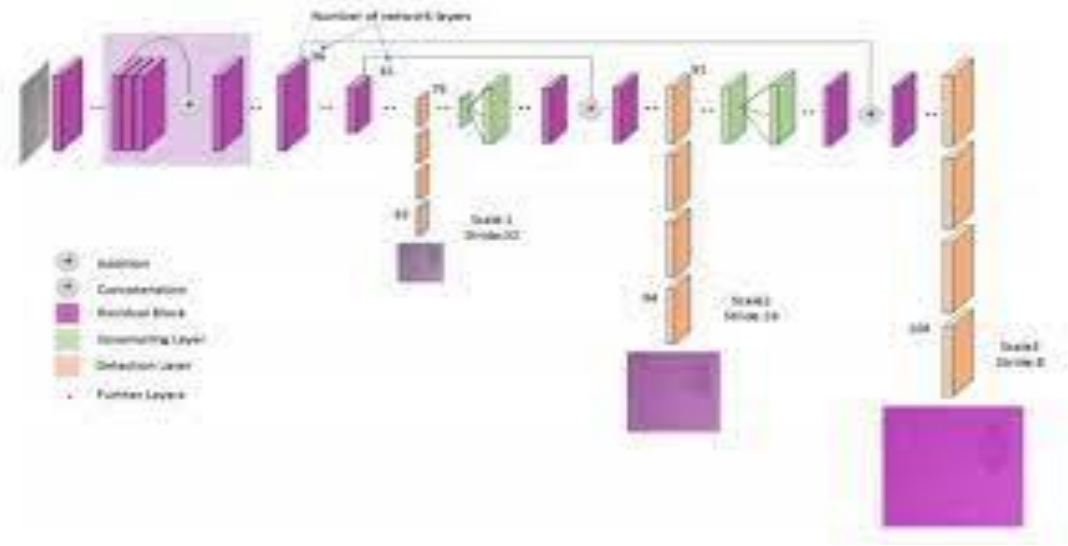
- it predicts C conditional class probabilities(one per class for the likeliness of the object class)

Each boundary box contains 5 elements(x,y,w,h) and a box confidence score. The confidence score reflects how likely the box contains an object and how accurate is the boundary box.

w and h are the bounding width and height of the box

x and y are offsets of the corresponding cell.

Network Architecture Diagram of Yolov3:



Description of Architecture:

Yolov3 uses a variant of Darknet, which originally has 53-layer network trained on ImageNet For the task of detection, 53 more layers are stacked onto it, giving us a 106 layer fully convolutional underlying architecture for Yolov3

Yolov3 Loss function:

Instead of using mean square error in calculating the classification loss, Yolov3 uses binary cross-entropy loss for each label. This also reduces the computation complexity by avoiding the softmax function.

Yolov3 predicts an objectness score for each bounding box using logistic regression.

Yolov3 uses sum-squared error between the predictions and the ground truth to calculate loss. The loss function composes of:

-the **classification loss**

-the **localization loss**

-the **confidence loss**

Classification loss:

If an object is detected, the classification loss at each cell is the squared error of the class conditional probabilities for each class:

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

where

$\mathbb{1}_i^{obj} = 1$ if an object appears in cell i, otherwise 0.

$\hat{p}_i(c)$ denotes the conditional class probability for class c in cell i.

Localization loss:

The localization loss measures the errors in the predicted boundary box location and sizes. We only count the box responsible for detecting the object

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

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

where

$\mathbb{1}_{ij}^{obj} = 1$ if the j th boundary box in cell i is responsible for detecting the object, otherwise 0.

λ_{coord} increase the weight for the loss in the boundary box coordinates.

Yolo do not want to weight absolute errors in large boxes and small boxes equally. To partially address this, Yolo predicts the square root of the bounding box width and height instead of the width and height. In addition, to put more emphasis on the boundary box accuracy, Yolo multiply the loss by λ_{coord} (default: 5)

Confidence loss:

If an object is detected in the box, the confidence loss is:

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

where

$\hat{C}_i$ is the box confidence score of the box j in cell i.

$\mathbb{1}_{ij}^{obj} = 1$ if the j th boundary box in cell i is responsible for detecting the object, otherwise 0.

If an object is not detected in the box, the confidence loss is:

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

where

$\mathbb{1}_{ij}^{noobj}$ is the complement of $\mathbb{1}_{ij}^{obj}$.

$\hat{C}_i$ is the box confidence score of the box j in cell i.

λ_{noobj} weights down the loss when detecting background.

Loss:

The final loss adds localization, confidence and classification loss together

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

Source

Benefits of Yolo:

-Good for real-time processing. Fast.

-Can be predictions (object locations and classes) are made from one single network. Trained end-to-end to improve accuracy

-it outperforms other methods when generalizing from Yolo is more generalized natural images to other domains like artwork