# Introduction to Object Detection

Liyuan Cao    Haidong Gu

Industrial and Systems Engineering
Lehigh University

OptML, Sep 12, 2019

## Table of Contents

1 Introduction

2 Object Localization

3 Target Variable

4 Loss Function

5 Object Detectionn

6 YOLO (You Only Look Once) Algorithm

7 Loss Function of YOLO

## Table of Contents

1 Introduction

2 Object Localization

3 Target Variable

4 Loss Function

5 Object Detectionn

6 YOLO (You Only Look Once) Algorithm

7 Loss Function of YOLO

## Introduction

Humans can easily detect and identify objects present in an image. The human visual system is fast and accurate and can perform complex tasks like identifying multiple objects and detect obstacles with little conscious thought.

With the availability of large amounts of data, faster GPUs, and better algorithms, we can now easily train computers to detect and classify multiple objects within an image with high accuracy.

In this tutorial, we will explore terms such as object detection, object localization, loss function for object detection and localization.

## Table of Contents

1. Introduction
2. Object Localization
3. Target Variable
4. Loss Function
5. Object Detectionn
6. YOLO (You Only Look Once) Algorithm
7. Loss Function of YOLO

## Object Localization

An image classification or image recognition model simply detect the probability of an object in an image. In contrast to this, object localization refers to identifying the location of an object in the image.

An object localization algorithm will output the coordinates of the location of an object with respect to the image. In computer vision, the most popular way to localize an object in an image is to represent its location with the help of bounding boxes. Fig 1 shows an example of a bounding box.

## Object Localization

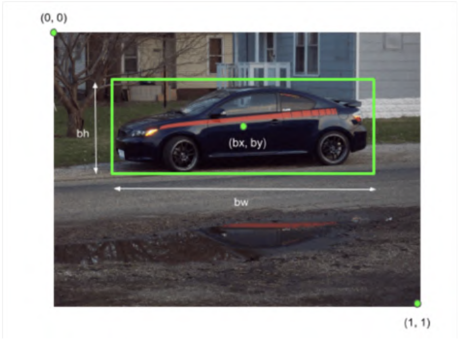

Figure: Bounding box representation used for object localization

A bounding box can be initialized using the following parameters:

- bx, by: coordinates of the center of the bounding box
- bw: width of the bounding box w.r.t the image width
- bh : height of the bounding box w.r.t the image height

## Table of Contents

1. Introduction
2. Object Localization
3. Target Variable
4. Loss Function
5. Object Detectionn
6. YOLO (You Only Look Once) Algorithm
7. Loss Function of YOLO

## Target Variable

The target variable for a multi-class image classification problem is defined as:

$$y = [c_1, \ldots, c_i, \ldots]^T$$

where $c_i$ = probability of the $i$th class.

For example, if there are four classes, the target variable is defined as

$$y = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

## Target Variable

We can extend this approach to define the target variable for object localization. The target variable is defined as

$$y = [p_c, b_x, b_y, b_h, b_w, c_1, c_2, c_3, c_4]^T$$

where we have

- $p_c$ = probability/confidence of an object (i.e the four classes) being present in the bounding box.
- $b_x, b_y, b_h, b_w$ = bounding box coordinates.
- $c_i$ = probability of the $i$th class the object belongs to.

For example, the four classes be truck, car, bike, pedestrian and their probabilities are represented as $c_1, c_2, c_3, c_4$. Thus,

$$p_c = \begin{cases} 1, & c_i : \{c_1, c_2, c_3, c_4\} \\ 0, & \text{otherwise} \end{cases}$$

## Table of Contents

1 Introduction

2 Object Localization

3 Target Variable

4 Loss Function

5 Object Detectionn

6 YOLO (You Only Look Once) Algorithm

7 Loss Function of YOLO

## Loss Function

Let the values of the target variable $y$ are represented as $y_1, y_2, \ldots, y_9$.

$$\begin{array}{c} y = [p_c,\ b_x,\ b_y, b_h,\ b_w, c_1,\ c_2,\ c_3,\ c_4]^T \\ y_1,\ y_2,\ y_3,\ y_4,\ y_5,\ y_6,\ y_7,\ y_8,\ y_9 \end{array}$$

The loss function for object localization will be defined as

$$\mathcal{L}(\widehat{y}, y) = \begin{cases} (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \ldots + (\hat{y}_9 - y_9)^2 & , y_1 = 1 \\ (\hat{y}_1 - y_1)^2 & , y_1 = 0 \end{cases}$$

In practice, we can use a log function considering the softmax output in case of the predicted classes $(c_1, c_2, c_3, c_4)$. While for the bounding box coordinates, we can use something like a squared error and for $p_c$ (confidence of object) we can use logistic regression loss.

## Table of Contents

1 Introduction

2 Object Localization

3 Target Variable

4 Loss Function

5 Object Detectionn

6 YOLO (You Only Look Once) Algorithm

7 Loss Function of YOLO

# Object Detection

An approach to building an object detection is to first build a classifier that can classify closely cropped images of an object. Fig 2 shows an example of such a model, where a model is trained on a dataset of closely cropped images of a car and the model predicts the probability of an image being a car.

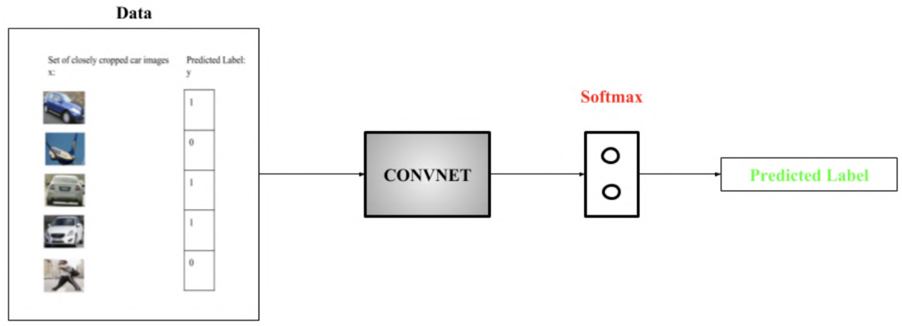

Figure: Image classification of cars

## Object Detection

Now, we can use this model to detect cars using a sliding window mechanism. In a sliding window mechanism, we use a sliding window (similar to the one used in convolutional networks) and crop a part of the image in each slide. The size of the crop is the same as the size of the sliding window. Each cropped image is then passed to a ConvNet model (similar to the one shown in Fig 2), which in turn predicts the probability of the cropped image is a car.

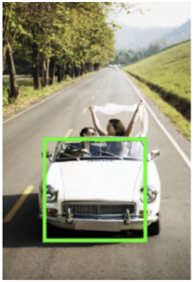

Figure: Sliding windows mechanism

## Object Detection

After running the sliding window through the whole image, we resize the sliding window and run it again over the image. We repeat this process multiple times. Since we crop through a number of images and pass it through the ConvNet, this approach is both computationally expensive and time-consuming, making the whole process really slow. Convolutional implementation of the sliding window helps resolve this problem.

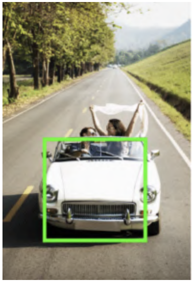

Figure: Sliding windows mechanism

## Table of Contents

1 Introduction

2 Object Localization

3 Target Variable

4 Loss Function

5 Object Detectionn

6 YOLO (You Only Look Once) Algorithm

7 Loss Function of YOLO

## YOLO (You Only Look Once) Algorithm

A better algorithm that tackles the issue of predicting accurate bounding boxes while using the convolutional sliding window technique is the YOLO algorithm. YOLO stands for "you only look once" and was developed in 2015 by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi.

It's popular because it achieves high accuracy while running in real time. This algorithm is called so because it requires only one forward propagation pass through the network to make the predictions.

## YOLO (You Only Look Once) Algorithm

The algorithm divides the image into grids and runs the image classification and localization algorithm (discussed under object localization) on each of the grid cells. For example, we have an input image of size **256** × **256**. We place a **3** × **3** grid on the image (see Fig 5).

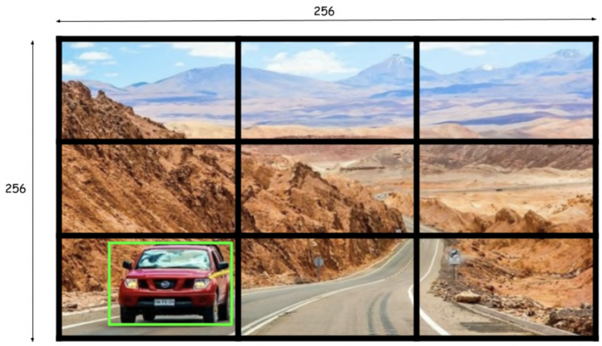

Figure: Grid (3 x 3) representation of the image

## YOLO (You Only Look Once) Algorithm

Next, we apply the image classification and localization algorithm on each grid cell. For each grid cell, the target variable is defined as

$$y_{i,j} = \begin{bmatrix} p_c & b_x & b_y & b_h & b_w & c_1 & c_2 & c_3 & c_4 \end{bmatrix}^T$$

Do everything once with the convolution sliding window. Since the shape of the target variable for each grid cell is $9 \times 1$ and there are 9 ($3 \times 3$) grid cells, the final output of the model will be: $3 \times 3 \times 9$.

The advantages of the YOLO algorithm is that it is very fast and predicts much more accurate bounding boxes. Also, in practice to get more accurate predictions, we use a much finer grid, say $19 \times 19$, in which case the target output is of the shape $19 \times 19 \times 9$.

## Table of Contents

1. Introduction
2. Object Localization
3. Target Variable
4. Loss Function
5. Object Detectionn
6. YOLO (You Only Look Once) Algorithm
7. Loss Function of YOLO

## Loss Function of YOLO

YOLO 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** (errors between the predicted boundary box and the ground truth).
- the **confidence loss** (the objectness of the box).

## Loss Function of YOLO

**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 $i$.

## Loss Function of YOLO

**Localization loss**

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

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

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

where

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

$\lambda_{coord}$ increase the weight for the loss in the boundary box coordinates.

## Loss Function of YOLO

**Localization loss**

We do not want to weight absolute errors in large boxes and small boxes equally. i.e. a 2-pixel error in a large box is the same for a small box. 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, we multiply the loss by $\lambda_{coord}$ (default: 5).

## Loss Function of YOLO

**Confidence loss**

If an object is detected in the box, the confidence loss (measuring the objectness of the box) 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.

## Loss Function of YOLO

**Confidence loss**

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 $i$.

$\lambda_{noobj}$ weights down the loss when detecting background.

Most boxes do not contain any objects. This causes a class imbalance problem, i.e. we train the model to detect background more frequently than detecting objects. To remedy this, we weight this loss down by a factor $\lambda_{noobj}$ (default: 0.5).

## Loss Function of YOLO

**Loss**

The final loss adds localization, confidence and classification losses together.

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