YOLO v3 Architecture

YOLO is a brand-new, real-time object detection system. It's uniqueness stems from it's trait of applying only *one* single neural network to an image. Prior detection systems repurpose classifiers or localizers to perform detection. They apply the model to an image at *multiple* locations and scales, and high scoring regions of the image are considered detections.

YOLO uses it's one neural network per image to divide the image into different regions, and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

Hence YOLO got it's "free-spirited" name: **Y**ou **O**nly **L**ook **O**nce.

There are several advantages in this essential trait of YOLO, over the other classifier-based object detection system. It looks at the whole image at test time so it's predictions are informed by global context in the image. It also makes predictions with a single network evaluation unlike systems like R-CNN (first option for architecture in the assignment) which require thousands for a single image. This makes YOLO extremely fast, compared to other object detection systems.

YOLO v3 is the third version of YOLO architecture, it uses a few tricks to improve training and increase performance, including: multi-scale predictions, a better backbone classifier, and more.

Loss mechanism in YOLO v3

The loss for each box prediction is comprised of the following terms:

1. **Coordinate loss** – when a prediction not exactly covering an object.
2. **Objectness loss** – due to a wrong box-object IoU (intersection over union) prediction.
3. **Classification loss** – due to deciations from predictiong '1' for the correct classes and '0' for all the other classes for the object in that box.

YOLO loss function

Regression loss

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

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

Confidence loss

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

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

Classification loss

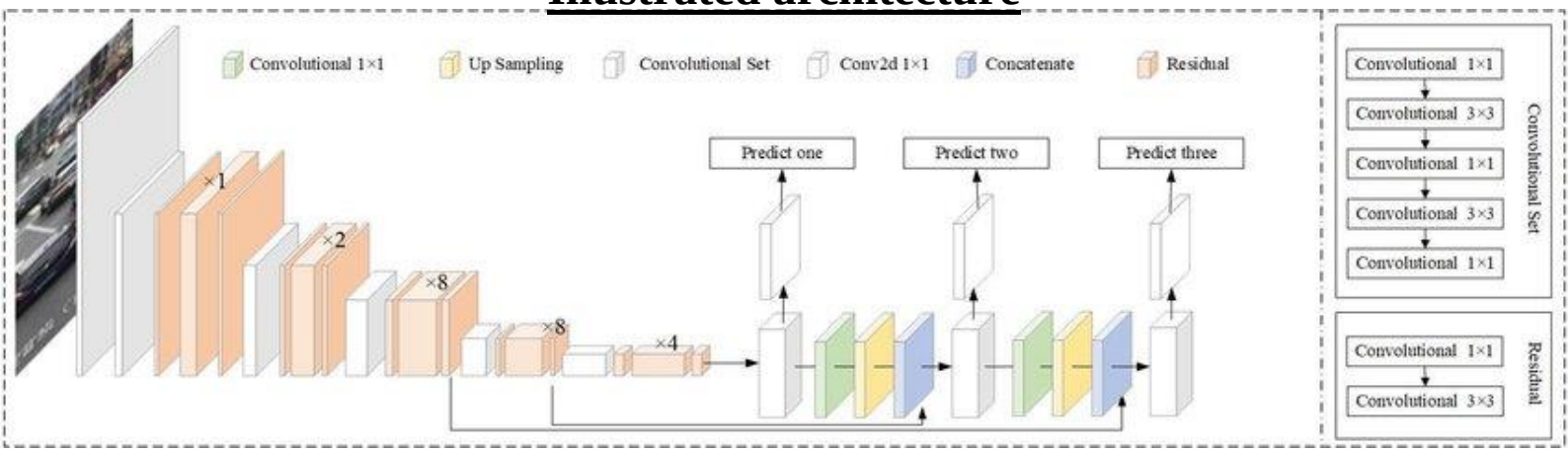
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

YOLO-V1 loss function. The lambdas are loss coefficients. The top 3 lines are the loss contributed by the 'best boxes' (boxes that best capture GT objects in each spatial cell) while the 4th is due to the boxes that did not capture objects. In YOLO V2 and V3 the direct width and height predictions and the square root were replaced with a residual scale prediction to make the loss argument proportional to relative rather than absolute scale error.

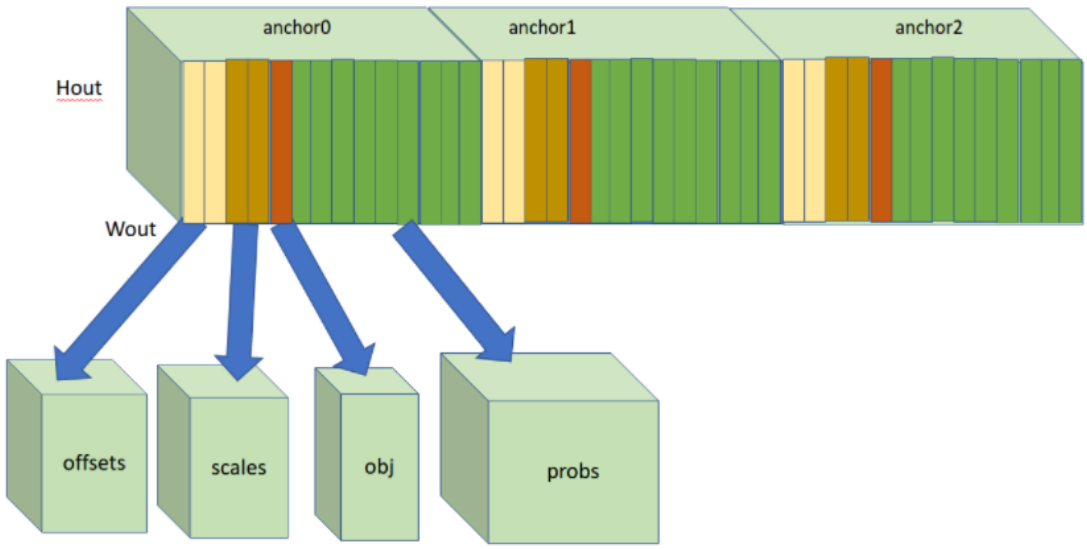
The quality of a box prediction is measured by its IoU (intersection over union) with the object it tries to predict (i.e. it's ground truth box). IoU values range from 0.0 (the box completely misses the object) to 1.0 (a perfect fit).

For each spatial cell, and for each box prediction that is centered in that cell, the loss function finds the box with the best IoU with the object centered in that cell. This distinguishing mechanism between best boxes and all the other boxes is the most fundamental attribute of the YOLO loss.

Illustrated architecture



YOLO-V2/V3 Output Scheme — A Single Layer Breakdown:



YOLO V2 and YOLO V3 output layer. Wout and Hout are spatial dimensions of the output feature map. For each anchor, the features are arranged in the described order. Source: Uri Almog