0.1 YOLO: You Only Look Once

- *Algorithm:* YOLO: You Only Look Once
- *Input:* A RGB 3 channel picture
- *Complexity:* None
- *Data structure compatibility:* None
- *Common applications:* Artificial intelligence

Problem. YOLO: You Only Look Once

Object detection is the task of detecting instances of objects of a certain class within an image. A sample of object detection is shown in Figure 1. Prior detection systems [**ren2015faster**] repurpose classifiers or localizers

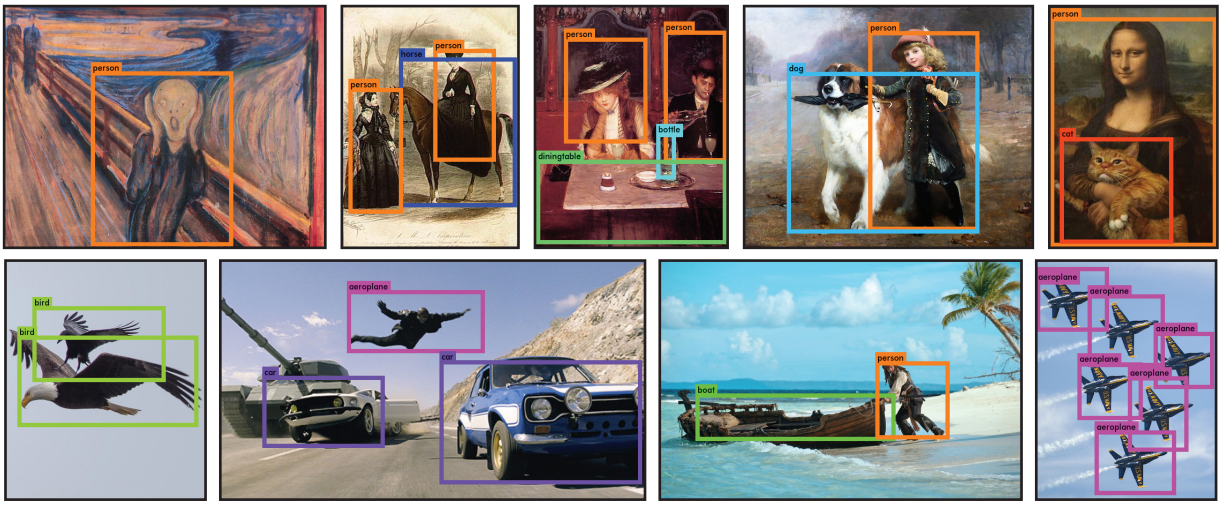

Figure 1: The result of object detection

to perform detection. They apply the model to an image at multiple locations and scales. High scoring regions of the image are considered detections.

YOLO [**redmon2016you**] uses a completely different approach where only a single neural network is applied to the full image. This network divides the image into grids and predicts bounding boxes and probabilities for each grid. These bounding boxes are weighted by the predicted probabilities.

YOLO has several merits. It looks at the whole image at test time so its predictions are informed by global context in the image. It only uses a single network evaluation to predict unlike region-based network which require thousands for a single image. This makes it extremely fast, more than 1000x faster than R-CNN [**girshick2015region**] and 100x faster than Fast R-CNN [**girshick2015fast**].

Description

Detection Method

YOLO system cuts the input image into an $S \times S$ grid. If the center of an object is in a grid cell, that grid cell will detect that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores is an value of how confident and accurate is that the box contains an object.

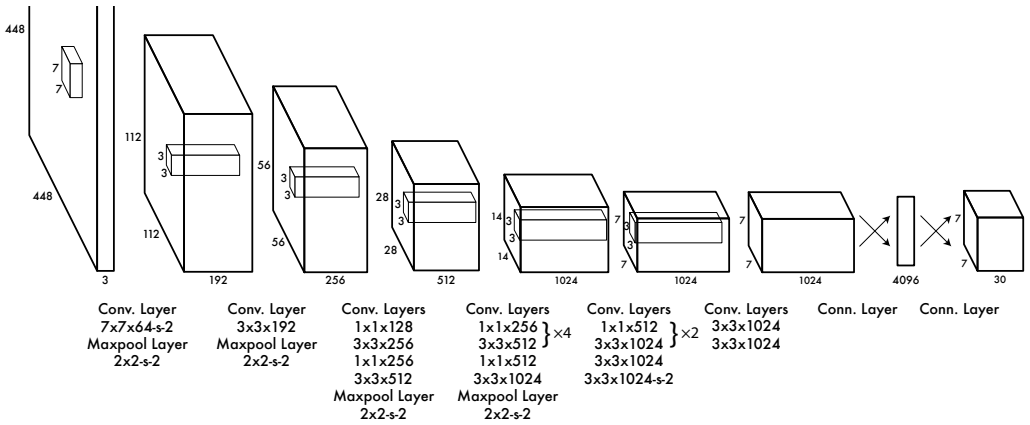

Figure 2: **The Architecture.** The detection network has 24 convolutional layers and 2 following fully connected layers. To be noted that 1×1 convolutional layers are alternated.

Network

The initial convolutional layers of the network extract features from the image while the fully connected layers predict the output probabilities and coordinates.

The network architecture follows the GoogLeNet model for image classification [**szegedy2015going**]. The network has 24 convolutional layers followed by 2 fully connected layers. It alternatively use 1×1 reduction layers followed by 3×3 convolutional layers. Figure 2 shows the network.

Training

It pretrains its convolutional layers on the ImageNet 1000-class competition dataset [**russakovsky2015imagenet**]. For pretraining, only first 20 convolutional layers from Figure 2 are used followed by a average-pooling layer and a fully connected layer.

It then converts the model to perform detection. It adds four convolutional layers and two fully connected layers. They are randomly initialized weights. They increases the input resolution of the network from 224×224 to 448×448. The final layer predicts both class probabilities and bounding box coordinates. It falls between 0 and 1 by normalizing the bounding box width and height by the image width and height so that they .

Leaky rectified linear activation function are used for the final layer and all other layers:

$$\phi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{cases} \tag{0.1.1}$$

YOLO can predict multiple bounding boxes in each grid. But it only needs one bounding box predictor to be responsible for each object at training time. one predictor well be responsible for predicting an object. The assignment is based on which prediction has the best IOU with the ground truth.

The following multi-part loss function will be used in training:

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

where ∞_i^{obj} denotes if object appears in cell i and $\infty_{ij}^{\mathrm{obj}}$ denotes that the jth bounding box predictor in cell i is responsible for that prediction.

To avoid overfitting it uses dropout and extensive data augmentation. Drop out rate is 0.5. For data augmentation, it composes random scaling and translations. Exposure and saturation will be also randomly adjusted in the HSV color space.