YOLO pre-report

김태훈[1]

[1]부산대학교 전기컴퓨터공학부.

1 Introduction

YOLO[2] is the object detection with unified architecture, which predicts bounding boxes and class probabilities from full images in one evaluation.

Unlike object detection with sliding window and region-proposal based architectures, YOLO is extremely fast(45 frames per second) and better at generalizing from natural images to other domain, which means a model that learns from natural images can detect objects from other domain like artworks. Also, YOLO makes less background errors but more localization errors compared to Fast R-CNN[1]

This report summarize the YOLO paper[2] focusing on the architectures.

2 Unified Detection

YOLO network uses features from entire images to predict each bounding box. This design enable end-to-end training and real-time speed.

This system divides the input image into an $S \times S$ grid. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores means how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. Confidence is defined as $Pr(Object) \times IOU^{truth}_{pred}$. IOU means that Intersection over Union of ground-truth bounding boxes and predicted bounding boxes.

Each bounding box contains x, y, w, h. (x, y) coordinates is center of the box relative to the grid cell, and (w, h) is width and height relative to the whole image.

Each grid cell also predicts C conditional class probabilities, which means the probabilities are conditioned on the grid cell containing an object, $PR(Class_i|Object)$. Also, each grid cell predicts one set of class probabilities regardless of the number of boxes.

Finally, These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor with $S \times S$ grid that divided from images, B bounding boxes and C class probabilities that each grid cell predicts.

3 Network Design

This network contains 24 convolutional layers followed by 2 fully connected layers. The convolutional layers extract features from image and the fully connected layers predict the class probabilities and coordinates of bounding boxes. Fast YOLO uses fewer convolutional layers(9 instead of 24) for fast object detection.

4 Training

The first 20 convolutional layers was pretrained with Imagenet datasets. For pretraining, the convolutional layers followed by a average-pooling layer and a fully connected layer used. Then, four convolutional layer and two fully connected layers are added to pretrained network.

They use leaky-ReLU for activation, and sum-of-squares for error function. Unlike normal sum-of-squares function, They increase the loss from bounding box coordinate prediction and decrease the loss from confidence prediction for boxes that don't contain objects because localization is more difficult than classification and there are less grid cells that contains object than those that don't. If normal sum-of-squares function is used, The model can predict all confidence score of grid cell to almost zero.

Also, they predict the square root of the bounding box width and height to reflect more loss for deviation of small boxes. For square root function, the rate of increase decreases as the width and height increase.

YOLO predicts multiple bounding boxes per grid cell, but finally, The bounding boxes that has highest IOU is assigned to be responsible for object.

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

Figure 1: The loss functions for YOLO

However, some objects can be localized by multiple cells. It can be fixed by non-maximum suppression.

4.1 Loss Function

The loss function for training YOLO is figure 1. $\mathbf{1}_i^{obj}$ denotes whether object appears in cell i, $\mathbf{1}_{ij}^{obj}$ denotes that the jth bounding box predictor in cell i is responsible, λ_{noobj} denotes balancing parameter for boxes that contain object and those don't, and $\lambda coord$ denotes balancing parameter for coordinates loss and classification loss. The meanings of each term are as follows:

1. For the responsible bounding box predictor j in cell i (the cell i contains object), get loss of x and y
2. For the responsible bounding box predictor j in cell i (the cell i contains object), get loss of $\sqrt{w}$ and $\sqrt{h}$. For reflecting deviation of small boxes more, calculate loss of square root (4)
3. For the responsible bounding box predictor j in cell i (the cell i contains object), get loss of confidence score $(C_i = 1)$
4. For the bounding box predictor j in cell i (the cell i **doesn't** contain object), get loss of confidence score $(C_i = 0)$
5. For cell that contains object, get loss of conditional class probability (for correct class, $p_i(c) = 1$)

5 Experiments

In short, mAP and FPS of YOLO is 63.4 and 45. Also, mAP and FPS of Fast YOLO is 52.7 and 155. For comparision, Faster R-CNN[3] VGG-16 has 73.2 mAP and 7 FPS. the mAP of YOLO isn't biggest, but it is fast and can be processed in real-time.

6 Conclusion

YOLO is simple but fast and accurate. Also, YOLO can detect a new image that not in training set. It is fast so it can process in real-time. Therefore, it is useful for application that fast, robust object detection.

[1] Ross Girshick. Fast r-cnn, 2015. URL https://arxiv.org/abs/1504.08083.

[2] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016. URL https://arxiv.org/abs/1506.02640.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016. URL https://arxiv.org/abs/1506.01497.