CS766 Midterm Report
Topic: Object Detection via YOLO approach

Members: Ya-Chun Yang, Hui-Shun Hung

Abstract

YOLO (You Only Look Once) reframe object detection as a single regression problem to spatially separated bounding boxes and associated class probabilities. In this project, we re-implemented the state-of-the-art YOLO approach, loaded the pre-trained weights, followed by further training using VOC 2007 train/val dataset. We evaluated the model with VOC 2007 test dataset and achieved 44.8% mean average precision (mAP). We will further refine our model and run the model with streaming video in real-time and custom datasets. Next, we will examine the effects of test parameters on the performance of YOLO model.

Introduction

In the YOLO approach, only a single convolutional network was trained to simultaneously predict multiple bounding boxes and class probabilities for those boxes. It not only is extremely fast (processes 155 frames per second), but can be optimized end-to-end directly on detection performance to achieve high mean average precision (mAP). Previous analysis indicated that YOLO model outperforms all other detection methods by low background errors and wide margin when generalizing detection from artwork [1].

Detection Algorithm

YOLO devides the input image into a $S \times S$ grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting the object. Each grid predicts B bounding boxes and the corresponding confidence scores. The confidence score is defined as Pr(Object) × IOU, where IOU = intersection area / union area of predicted and ground truth boxes. Each bounding box consist of 5 predictions: x, y, w, h, and confidence. The (x, y)represent the center of the box relative to the bounds of the grid cell. The (w, h) indicate the width and height relative to the whole image. The confidence prediction represents the IOU between the predicted box and any ground truth box. In addition, each grid cell predicts C conditional class probabilities , $\Pr(\text{Class}_i \mid \text{Object})$. We only predict one set of class probabilities per grid cell. At test time, we multiply the conditional class probabilities and the individual box confidence predictions,

$$\Pr(\text{Class}_i \mid \text{Object}) \times \Pr(\text{Object}) \times \text{IOU} = \Pr(\text{Class}_i) \times \text{IOU}$$

which provides the information of class-specific confidence scores for each box. In our implementation, we use $S = 7$, $B = 2$. We trained and evaluated our model on PASCAL VOC 2007/2012 datasets [2][3], which has 20 labelled classes, so $C = 20$.

Loss Function

YOLO's loss function must simultaneously solve the object detection and classification tasks. Thus, the function penalizes incorrect object detections as well as measures the most likely classification. The function is as follows:

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

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

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

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

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

Note that 1_i^{obj} equals one if object appears in the grid cell i and 1_{ij}^{obj} equals one if the jth bounding box predictor in grid cell i is responsible for that prediciton. We also set $\lambda_{coord}=5$ and $\lambda_{noobj}=0.5$ to increase the loss from bounding box coordinate predictions that don't contain objects.

Current Results

Implementation of YOLO network architechture

YOLO directly straights from image pixels to bounding boxes and associated class probabilities with single neural network. The network has 24 convolutional layers followed by 3 fully connected layers. The input image is 448×448 for the requirement of fine-grained visual information. The final output of the network is the $7\times7\times30$ tensor of predictions. We have finished the implementation of the full network in python and TensorFlow. The architecture is shown in Figure 1, with one extra fully connected layers with dimension 512 for the YOLO-small model.

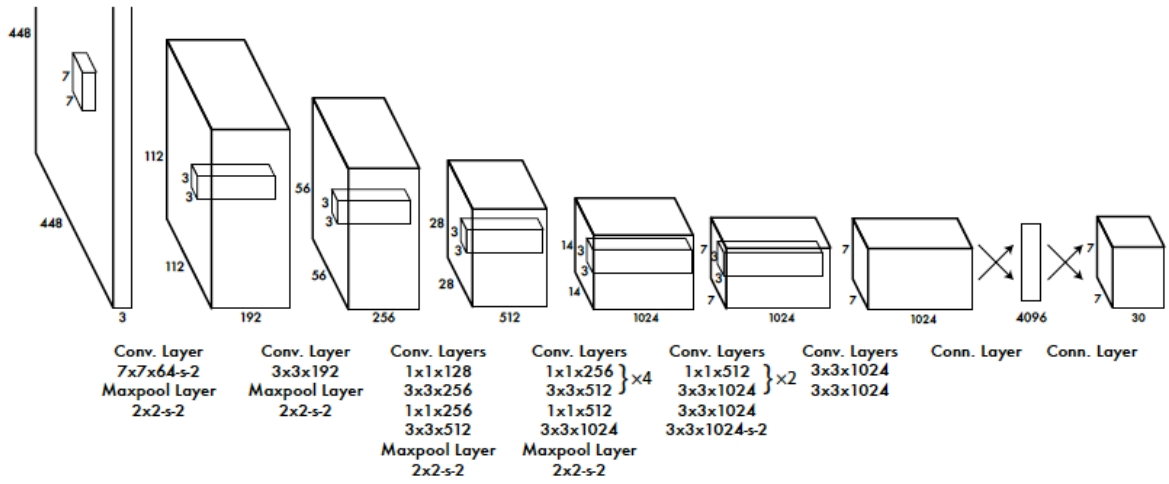

Figure 1. The YOLO Architecture.

Evaluation of YOLO model

To validate the implemented network architecture, we used the pre-trained weights, yolo-small.weights, which was trained on VOC 2007 train/val datasets. We restored the

weights from yolo-small.weights and trained the model on VOC 2007 train/val for further 50 epoches (~ 9 hrs) on AWS cloud with NVIDIA K80 GPU configured with CUDA 9.0 and cuDNN 7.0.5 environment. We employ the gradient descent optimization method offered by TensorFlow with batch size 64. We evaluated our model on VOC 2007 test dataset, and achieve 44.8% mAP, the average precision for each class is shown on Table 1. We further examined our model on few example images, the result indicated that our implemented model displays the similar inference result in comparison with the author's work (Figure 2). However, as shown in the left panel of Figure 2, our YOLO model could not detect all the objects.

Table 1. Average precision on 20 classes

Class Name	Ground Truth	Predicted	True Positives	False Positives	False Negatives	Avg Precision
aeroplane	287	266	188	32	46	0.6191
bicycle	358	274	207	33	34	0.5352
bird	549	471	272	74	125	0.4364
boat	340	306	121	75	110	0.2630
bottle	498	175	83	40	52	0.1250
bus	250	189	150	29	10	0.5709
car	1402	1254	764	176	314	0.5041
cat	375	355	275	55	25	0.6941
chair	1223	907	323	253	331	0.1523
cow	316	293	156	60	77	0.3856
dining table	263	162	122	34	6	0.4168
dog	525	480	355	80	45	0.6156
horse	379	310	258	29	23	0.6561
motorbike	349	274	199	30	45	0.4919
person	4477	4158	2417	475	1266	0.4719
potted plant	537	319	129	48	142	0.1586
sheep	280	230	126	35	69	0.3842
sofa	376	175	137	30	8	0.3379
train	302	265	212	33	20	0.6787
tv monitor	355	279	179	54	46	0.4624

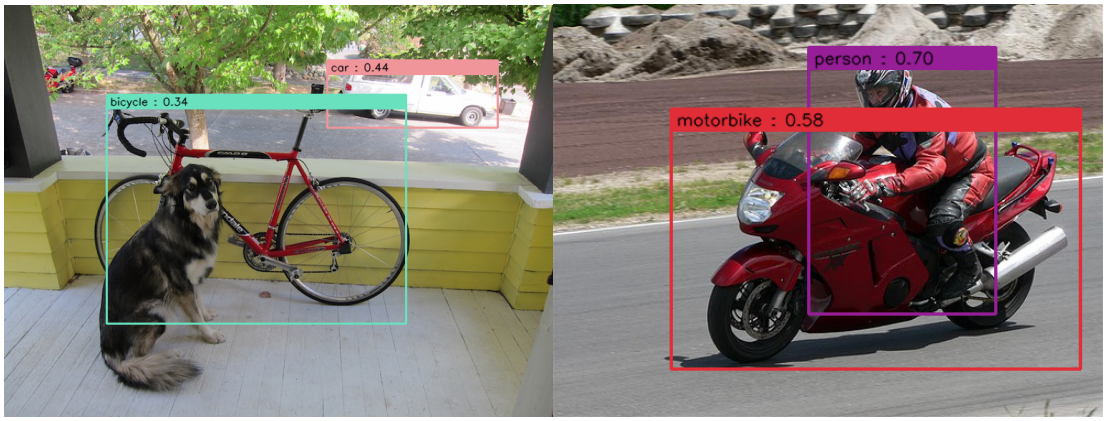

Figure 2. Sample images detected on YOLO. Each bounding box is indicated the class the corresponding confidence.

Future Work

Evaulation on more datasets

As indicated in author's website [4], the YOLO-small model can achieve 52% mAP. Therefore, we will continue refine our YOLO model by further training with VOC 2012 train/val datasets. We will also connect YOLO to a webcam and measure its real-time performance, including the time to fetch images from the camera and display the detections. In addition, we will feed custom data on YOLO model to validate its generalizability.

Examine the testing parameters

We will measure the YOLO detection performance on a various range of testing parameters, threshold and iou-threshold, to come up a general strategy for parameter design in YOLO approach.

Timeline

Time	Goal
4/2	Midterm report due
4/8	Refine YOLO model with VOC 2012 dataset
4/15	Run YOLO on streaming video in real-time Test YOLO with costum data
4/22	Examine the threshold and iou-threshold on YOLO detection
4/29	Finalize the project
4/30	Project presentation
5/7	Project webpage due

References

[1] J. Redmon, S. Divvala, R. Girshick, A.Farhadi. You only look once: unified, real-time object detection. *arXiv preprint arXiv*: 1506.02640v5, 2016.
[2] The PASCAL Visual Object Classes Challenge 2007
http://host.robots.ox.ac.uk/pascal/VOC/voc2007/
[3] The PASCAL Visual Object Classes Challenge 2012
http://host.robots.ox.ac.uk/pascal/VOC/voc2012/
[4] https://github.com/pjreddie/darknet/wiki/YOLO:-Real-Time-Object-Detection