CIS680: Vision and Learning
Project 2: Object Detection with Simplified YOLO
Due: October. 14, 2019 at 11:59 pm

1 Introduction

Object detection is a fundamental task in computer vision. The problem of object recognition essentially consists of first localizing the object and then classifying it with a semantic label. In recent deep learning based methods, YOLO is an extremely fast real time multi object detection algorithm.

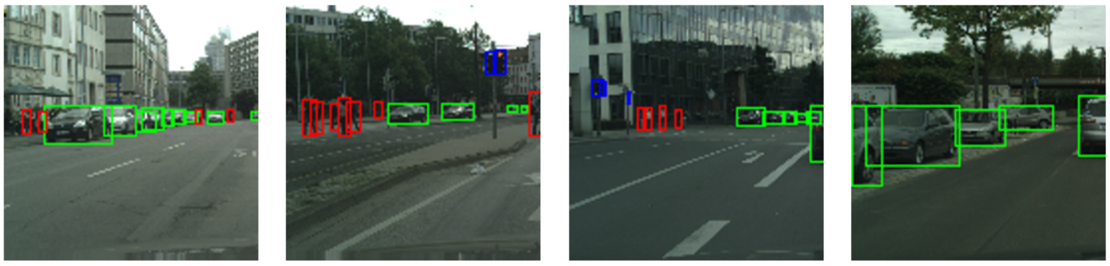

Figure 1: This is a demo of what object detection does. The color indicates different semantic class.

In this project, we provide 10K street scene images with correponding labels as training data. The image dimension is 128x128x3, and the labels include the semantic class and the bounding box corresponding to each object in the image. Note that a small portion of these ground-truth labels are not a little bit noisy and the quantity of the training set is not very large, so we do not expect you to learn a super robust object detector.

2 Data Preprocessing

In this section, you will need to write code to preprocess the ground truth labels for YOLO algorithm. The labels are provided in the "label" folder. The current format of the labels are (class, x1, y1, x2, y2), where x1, y1 are the top left corner of the bounding box and x2, y2 are the bottom right corner of the bounding box.

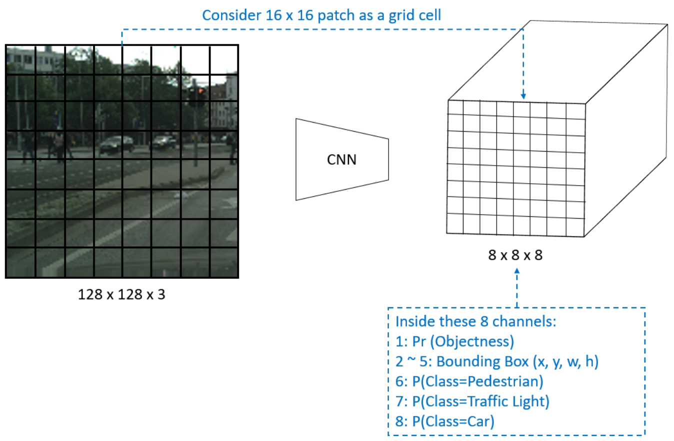

Figure 2: This figure demonstrates the format of ground-truth labels that YOLO algorithm requires.

For each image, you are required to convert the provided labels into the 8x8x8 ground truth matrix, which has the same dimension as the output of YOLO detection network. The instruction of this conversion is as follows:

- We consider a 16x16 image patch as a grid cell and thus divide the full image into 8x8 patches in the 2D spatial dimension. In the output activation space, one grid cell represents one 16x16 image patch with corresponding aligned locations.
- For simplied YOLO, we only use one anchor box, where we assume the anchor size is the same as the grid cell size. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. This means that there is only one anchor for each object instance.
- For each anchor, there are 8 channels, which encode Pr(Objectness), x, y, w, h, P(class=pedestrian), P(class=traffic light), and P(class=car).
- The Pr(Objectness) is the probability of whether this anchor is an object or background. When assigning the ground-truth for this value, "1" indicates object and "0" indicates background.
- The channels 2-5, x, y indicate the offset to the center of anchor box; w, h is the relative scale of the image width and height.
- In channels 6-8, you need to convert the ground truth semantic label of each object into one-hot coding for each anchor boxes.
- Note that if the anchor box does not have any object (Pr=0), you don't need to assign any values to channels 2-8, since we will not use them during training.

Intersection over Union (IoU) is a measure of the overlap between two bounding boxes.

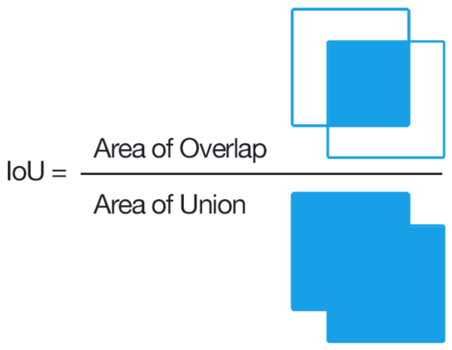

Figure 3: This figure demonstrates how to compute IoU.

3 Model Architecture

You are required to implement the model architecture using the following parameters. This model takes input with dimension of 128x128x3 and outputs an activation with dimension of 8x8x8.

Layer	Hyperparameters
conv1	Kernel size = 4x4x32, stride = 2, pad = 1. Followed by BatchNorm and ReLU
conv2	Kernel size = 4x4x64, stride = 2, pad = 1. Followed by BatchNorm and ReLU
conv3	Kernel size = 4x4x128, stride = 2, pad = 1. Followed by BatchNorm and ReLU
conv4	Kernel size = 4x4x256, stride = 2, pad = 1. Followed by BatchNorm and ReLU
conv5	Kernel size = 4x4x512, stride = 2, pad = 1. Followed by BatchNorm and ReLU
conv6	Kernel size = 4x4x1024, stride = 2, pad = 1. Followed by BatchNorm and ReLU
transposed_conv7	Kernel size = 4x4x256, stride = 2, pad = 1. Followed by BatchNorm and ReLU
transposed_conv8	Kernel size = 4x4x64, stride = 2, pad = 1. Followed by BatchNorm and ReLU
conv9	Kernel size = 3x3x8, stride = 1, pad = 1.

4 Training Details

During training, the localization and classification errors are optimized jointly. The loss function is shown as below. i indicates number of grid cells and j indicates number of anchor boxes at each grid cell. In our case, there is only one anchor box at each grid cell and $B = 1$.

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

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2 + \sum_{i=0}^{S^2} \mathbb{1}_{ij}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2$$

here is why we assume that anchor boxes size is equal to the grid cell size, and B = 1 in our case.

where $\mathbb{1}_{ij}^{obj}$ indicates if objects appears in grid cell i. λ_{coord} and and λ_{noobj} are two hyperparameters for coordinate predictions and non-objectness classification. We set $\lambda_{coord} = 5$ and and $\lambda_{noobj} = 0.5$.

During training, you can set learning rate of 10e-3 using Adam optimizer with default beta 1 and beta 2. You are encourage to adaptively adjust learning rate during training to see if the model will converge better. You should also visualize the loss over training iterations. You will need to decide at the iteration to stop training based on the loss visualization. (At least 20 epochs of training is required.)

5 Post-Processing

During inference, the network is going to predict lots of overlapping redundant bounding boxes. To eliminate the redundant boxes. There are basically two steps:

- Get rid of predicted boxes with low objectness probability (Pc < 0.6).
- After the first step, for each class, run IoU for all the bounding boxes and cluster boxes with IoU > 0.5 as a group. For each group, find the one with highest Pc and suppress the other boxes. This is referred as non-max suppression.
- To evaluate the performance of your YOLO implementation, compute the mean Average Precision (mAP) of inference. Predicted bounding boxes are a match with ground truth bounding boxes if they share the same label and have an IoU with the ground truth bounding box of greater than 0.5. These matches can be used to calculate a precision/recall curve for each class. The Average Precision for a class is the area under this curve. The mean of these Average Precision values over all the classes in inference gives the mean Average Precision of your network.

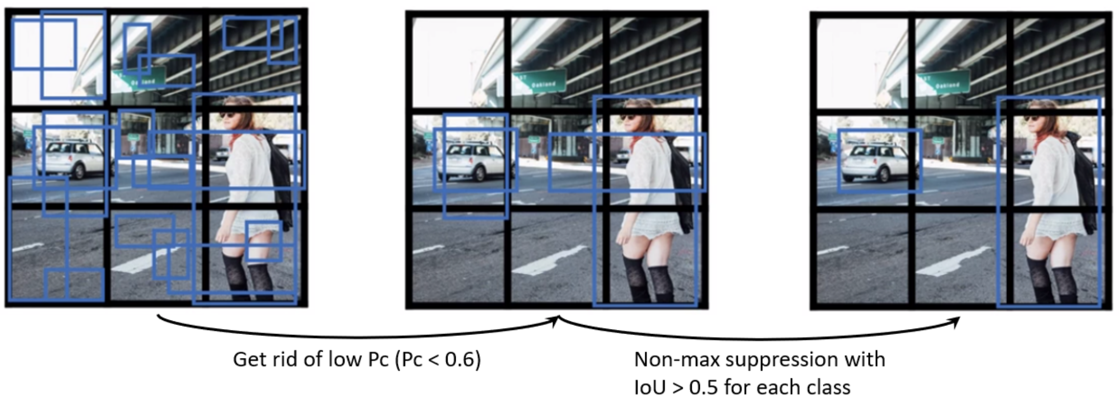
Get rid of low Pc (Pc < 0.6)
Non-max suppression with IoU > 0.5 for each class

Figure 4: This figure demonstrates post-process of how to get rid of redundant bounding box.

6 Submission and Evaluation

As you complete your implementation of YOLO, please include the following in your report. The contribution of each section to your grade is listed.

- Write clear instructions on how to run your code. Your code should execute without syntax errors using these instructions. (3%)
- We recommend that you begin this project by writing and testing your code for data preprocessing. In your report, visualize the following for an image where P(Objectness), P(class=pedestrian), P(class=traffic light), and P(class=car) are all not zero.
  - Show the chosen image with ground truth bounding boxes visualized. Use green bounding boxes around cars, red bounding boxes around pedestrians, and blue bounding boxes around traffic lights. See Figure 1 as an example of what this visualization should look like. (5%)
  - Convert the image to an 8x8x8 representation, and show each channel of your chosen image in the report. Label the visualization of each channel with the associated channel number and visualized value. Example: Channel 1 - Pr(Objectness) (5%)
  - To test your data preprocessing code, convert your 8x8x8 representation for this step back to the original input data, and ensure that your image with recomputed bounding boxes looks the same as the input. Show this image with bounding box visualizations in your report. This test code for data preprocessing will also be needed in the inference step. (2%)
- Make block diagram of requested architecture. Label the diagram with the parts of your code which implement each component. Explain any deviations you made from the described architecture. (15%)
- Show a plot of the loss over training for your model. (10%)
- Show a plot of the mean Average Precision over training for your model. (10%)
- For one image in the inference step, show the predicted bounding boxes in the image before performing any elimination. Then, show the annotated image after the low probability bounding boxes are removed. Finally, show image after performing non-max suppression. Label each of these visualizations. See Figure 4 as an example of what this visualization should look like. (15%)
- For one image in the inference step, show the bounding boxes visualized for each class with green bounding boxes around cars, red bounding boxes around pedestrians, and blue bounding boxes around traffic lights as done in Figure 1. Show the precision/recall curves for each class in inference. Write down the achieved mean Average Precision for your inference stage. (20%)
- Explain any issues or challenges you had with your implementation. Explain the performance you achieved and any interesting observations you made while training your network. Discuss hypotheses for how you could improve the performance of your implementation further. (15%)

This is an interesting but also very challenging project, so we encourage you to start as early as possible. Please submit your completed work to Gradescope. The submission should include a pdf of your report and a zip file with only the Python files for your implementation.

7 Some Useful Online Materials

Original YOLO paper:
https://arxiv.org/pdf/1506.02640.pdf
Intuitive Explanation:
https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006
Video Tutorial:
YOLO Tutorial
mean Average Precision:
mean Average Precision Tutorial
Intersection over Union:
https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection