BME695DL/ECE695: Homework 6

Spring 2021

Due Date: Wednesday, April 14,2021 (11:59pm)

Turn in your solutions via BrightSpace.

1 Introduction

This homework has the following goals:

1. To create a YOLO-like network architecture for multi-instance object detection and localization in the COCO images.

   As you saw in your previous homework, detecting and localizing objects is relatively straightforward when you can assume that an image contains only one object instance (or, when you can assume, that you need pay attention to only one object instance in an image). In such cases, you can associate the object label with the entire image.

   But how does one solve the problem when an image is allowed to contain multiple object instances and you want to identify and localize all of them at the same time?

   At the moment, there exist three major approaches for solving this problem: R-CNN, YOLO, and SSD. See the Preamble in my Week 9 lecture material for a brief introduction to these networks.

   What you will program will not be identical to YOLO for the simple reason that the competition grade object detectors are first trained on very large image datasets just to make them aware of what different objects can look like and, subsequently, fine-tuned for object detection work. And even after that, a lot of work goes into optimizing all of the network parameters for maximal performance. Creating such an object detector would be beyond the scope of a homework.

   Your goal is merely to demonstrate that you understand the basic logic of the YOLO approach to object detection.

2. To take your use of the COCO dataset to the next level — by using the bounding boxes for all the object instance in the images.

The following steps will prepare you to work with object detection, data loading with annotations, *e.g.*, bounding boxes and labels, and so on.

2 Getting Ready for This Homework

Before embarking on this homework, do the following:

1. Your first step would be to come to terms with the basic concepts of YOLO:

   Compared to everything you have done so far in our DL class, the YOLO logic is very complex. As I will explain in class, it uses the notion of Anchor Boxes. You divide an image into a grid of cells and you associate N anchor boxes with each cell in the grid. Each anchor box represents a bounding box with a different aspect ratio.

   Your first question is likely to be: Why divide the image into a grid of cells? To respond, the job of estimating the exact location of an object is assigned to that cell in the grid whose center is closest to the center of the object itself. Therefore, in order to localize the object, all that needs to be done is to estimate the offset between the center of the cell and the center of true bounding box for the object.

   But why have multiple anchor boxes at each cell of the grid? As previously mentioned, anchor boxes are characterized by different aspect ratios. That is, they are candidate bounding boxes with different height-to-width ratios. In my implementation in the RegionProposalGenerator module, I create five different anchor boxes for each cell in the grid, these being for the aspect ratios: $1/5, 1/3, 1/1, 3/1, 5/1$. The idea here is that the anchor box whose aspect ratio is closest to that of the true bounding box for the object will speak with the greatest confidence for that object.

2. You can deepen your understanding the YOLO logic by looking at my implementation of image gridding and anchor boxes in Version 2.0.1 of my RegionProposalGenerator module:

   `https://engineering.purdue.edu/kak/distRPG/RegionProposalGenerator-2.0.1.html`

   Go to the Example directory and execute the script:

```
multi_instance_object_detection.py
```

and work your way backwards into the module code to see how it works. In particular, I'd like you to see how I have implemented the notion of anchor boxes in the function

```
run_code_for_training_multi_instance_detection()
```

To execute the script `multi_instance_object_detection.py`, you will need to download the install the following datasets:

```
Purdue_Dr_Eval_Multi_Dataset-clutter-10-noise-20-size-10000-train.gz

Purdue_Dr_Eval_Multi_Dataset-clutter-10-noise-20-size-1000-test.gz
```

Note that these datasets need to be installed from the `RegionProposalGenerator` webpage.

In the dataset names, a string like 'size-10000' indicates the number of images in the dataset, the string 'noise-20' means 20% added random noise, and the string 'clutter-10' means a maximum of 10 background clutter objects in each image.

Follow the instructions on the main webpage for RegionProposalGenerator on how to unpack the image data archive that comes with the module and where to place it in your directory structure. These instructions will ask you to download the main dataset archive and store it in the `Examples` directory of the distribution.

3 Special Note

For all the previous homework assignments, you could look at the DL-Studio code for a reference implementation. But that's not the case with this homework. The code in the RegionProposalGenerator is NOT a full-blown implementation of the YOLO object detector. Your instructor's goal in that implementation was simply to show with actual code his best understanding of how anchor boxes are used in YOLO and how one assembles the 8-element encodings for each anchor box for every cell into a 1440-element vector for the target for a 128×128 training image. That is, for each input image, you would compare the output of the neural network with this 1440-element tensor.

Therefore, you are likely to run into a larger number of challenges as you do this homework. If you get an early start on this homework, your

instructor will be glad to work with you on resolving those challenges. If you seek your instructor's help regarding this homework, send him an email with the string "ECE695 YOLO" in the subject line. Remember, seeking help from your instructor does not imply getting him to write code for you.

4 How to Create a COCO Multi-Instance Subset

Recall that in your HW05, you picked the largest bounding box for choosing a label for any given image. For this homework, you could create a COCO subset similar to Dr. Eval multi-instance dataset, by picking three categories in COCO dataset.

For example, let's pick the following three COCO classes, `bus, truck,` and `car`. One way to create a COCO multi-instance subset is using the following criteria,

1. Select subset of COCO images such that all the above three objects are present in them.
2. Augment these images such that at least two objects are present.
3. For each individual category, *i.e.*, bus, truck and car, select only those images that have at least two instances for that category.

Obviously, additional conditions are required to avoid picking overlapping set of images in the above selection criteria.

5 Submission Instructions

You don't need any `argparse` arguments for this homework task and it's safe to assume that the COCO annotation files exist locally.

Similar to HW05 bundle your training and validation code into two main files and add any additional helper modules.

- Make sure to submit your code in Python 3.x and not Python 2.x.
- Name the .zip archive as `hw06_<Firstname><Lastname>.zip` (without any white spaces) with the following files

  `hw06_training.py`

  `hw06_validation.py`

  Note that if your trained model is too large in size, it's okay to skip it. However, make sure to include a pdf report instead, explaining your network design, *i.e.*, your understanding of YOLO-like object detectors and how you prepared your COCO subset for multi-instance classification and all the result plots. **Your code must be your own work.**
- You can resubmit a homework assignment as many times as you want up to the deadline. Each submission will overwrite any previous submission.