Introduction

YOLO (You Only Look Once) real-time object detection algorithm, which is one of the most effective object detection algorithms that also encompasses many of the most innovative ideas coming out of the computer vision research community.
Object detection is one of the classical problems in computer vision where you work to recognize what and where — specifically what objects are inside a given image and also where they are in the image. The problem of object detection is more complex than classification, which also can recognize objects but doesn't indicate where the object is located in the image. YOLO uses a totally different approach.
YOLO is a clever convolutional neural network (CNN) for doing object detection in real-time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. With YOLO, a single CNN
simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This model has a number of benefits over other object detection methods:

- YOLO is extremely fast
- YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance.
- YOLO learns generalizable representations of objects so that when trained on natural images and tested on artwork, the algorithm outperforms other top detection methods.

Aim

Train any object detection algorithm on the WebMarket dataset images (http://yuhang.rsise.anu.edu.au/) based on the bounding boxes provided here : https://github.com/ParallelDots/generic-sku-detection-benchmark/tree/master/annotations/WebMarket . Make sure you create train validation and test data and show output of the object detection algorithm on a few test images.

Requirements

1. Darknet is an open source neural network framework that supports CPU and GPU Computation.
2. Using google colab GPU service which is free trial for 12 hrs to train the images (Go to edit -> Notebook Setting -> Hardware accelerator -> GPU -> Save).

Dataset Preparation

To train the model we'll use Darknet, the official tool made by the creator of yolo. For the Darknet to properly parse your files you need a file that specifies all the info it needs to use. Let's call it **obj.data**. It contents are as follows:

- **classes** is pretty self explanatory: It specifies the number of classes you want to use.
- **train** is the path to a text file that has the paths to the images that the model will train on. Note: these paths should be relative to the Darknet executable.
- **valid** is also a path to a file but this time specifies the paths to images that would be used to verify the model output. The images you so diligently prepared should be split in roughly 3:1 ratio of train:test
- **names** is a list of object categories that the model should recognize. The number of lines in this file should correspond to the number of classes specified in the first line
- **backup** is the path to the directory where the intermediate results are stored.

```
classes= 1
train  = train.txt
valid  = test.txt
names = obj.names
backup = backup/
```

Config file

The /cfg directory of the Darknet repo contains config files for all types of yolo networks that were created over the past iterations of the algorithm. Ideally we'd want to use the YOLO v3 since it's the latest of all the algorithms that are optimized towards speed.

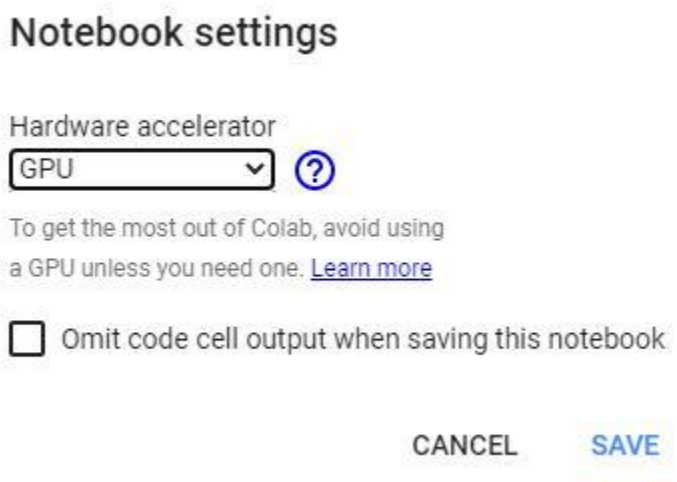
We need to adjust the yolov3.cfg to match the number of classes we have. Near the end of the file there's a line classes. Change that line to match the declared number of classes.

One more thing that needs to be changed in the file is the number of filters in the last convolutional layer to fit the new number of classes. You need to adjust that following the formula (number_of_classes + 5) * 5.

So in my case of recognizing object(1 classes) it was (1 + 5) * 5 = 30.

Training the model using Google colab

1. Open a new notebook in Google colab and set the runtime to use the GPU(Go to edit -> Notebook Setting -> Hardware accelerator -> GPU -> Save).



- Check the CUDA installation on the runtime
- Install cuDNN and opencv in makefile make it as 1 for working conditions
- Clone and compile Darknet. We'll use a repo based on Darknet repo. I applied some changes to makefile and loaded it from My Github Repo and reduced the number of logs on the console to speed up the notebook.

1. Clone the darknet folder
   # !git clone https://github.com/AlexeyAB/darknet.git

2. Change the Directory to darknet
   # cd /content/darknet

3. Clone my git repository where I have kept all my File
   # !git clone https://github.com/poojasrane99/Object-Detection-Using-Yolo.git

4. Remove the Makefile from the original darknet folder.
   # !rm /content/darknet/Makefile

5. Move the edited Makefile. (changes made in Makefile: GPU=1, CUDNN=1, CUDNN_HALF=1)
   # mv /content/darknet/Object-Detection-Using-Yolo/Makefile /content/darknet

6. Run the Makefile
   # !make

7. Run this command
   # ! ./darknet

8. Use precomputed weights:
   This step is optional but highly recommended. You could just launch Darknet without initial
   weights, but in my testing it turned out that starting with precomputed weights increased the
   output quality significantly.
   To start you need to get the weights from the official yolo site and strip the last couple of layers

   using darknet command:
   # !wget https://pjreddie.com/media/files/darknet19_448.conv.23

   This leaves you with a partially trained weights file that you can use in the next step.

9. Move Dataset from Object-Detection-Using-Yolo to darknet
   # mv /content/darknet/Object-Detection-Using-Yolo/Dataset /content/darknet

10. Move image from Object-Detection-Using-Yolo to darknet
    # mv /content/darknet/Object-Detection-Using-Yolo/image /content/darknet

11. For training we use convolutional weights that are pre-trained on Imagenet. We use weights from the darknet53 model. You can just download the weights for the convolutional layers here.
    # wget https://pjreddie.com/media/files/darknet53.conv.74

12. Move test.txt from Object-Detection-Using-Yolo to darknet
    # mv /content/darknet/Object-Detection-Using-Yolo/test.txt /content/darknet

13. Move train.txt from Object-Detection-Using-Yolo to darknet
    # mv /content/darknet/Object-Detection-Using-Yolo/train.txt /content/darknet

14. Move obj.data from Object-Detection-Using-Yolo to cfg
    # mv /content/darknet/Object-Detection-Using-Yolo/obj.data /content/darknet/cfg

15. Move obj.names.data from Object-Detection-Using-Yolo to cfg
    # mv /content/darknet/Object-Detection-Using-Yolo/obj.names /content/darknet/cfg

16. Remove the yolo-voc.2.0.cfg from the cfg folder.
    # !rm /content/darknet/cfg/yolo-voc.2.0.cfg

17. Move the edited yolo-voc.2.0.cfg. (changes made in yolo-voc.2.0.cfg: filters=30, classes=1)
    # mv /content/darknet/Object-Detection-Using-Yolo/yolo-voc.2.0.cfg /content/darknet/cfg

18. The command to start training is
    # !./darknet detector train cfg/obj.data cfg/yolo-voc.2.0.cfg darknet19_448.conv.23 -dont_show 0

    Since we have an annotation error the output cannot be seen.
    (Error: The provided CSV file is in x1,y1,x2,y2
    But the standard YOLO format is <x> <y> <width> <height>)