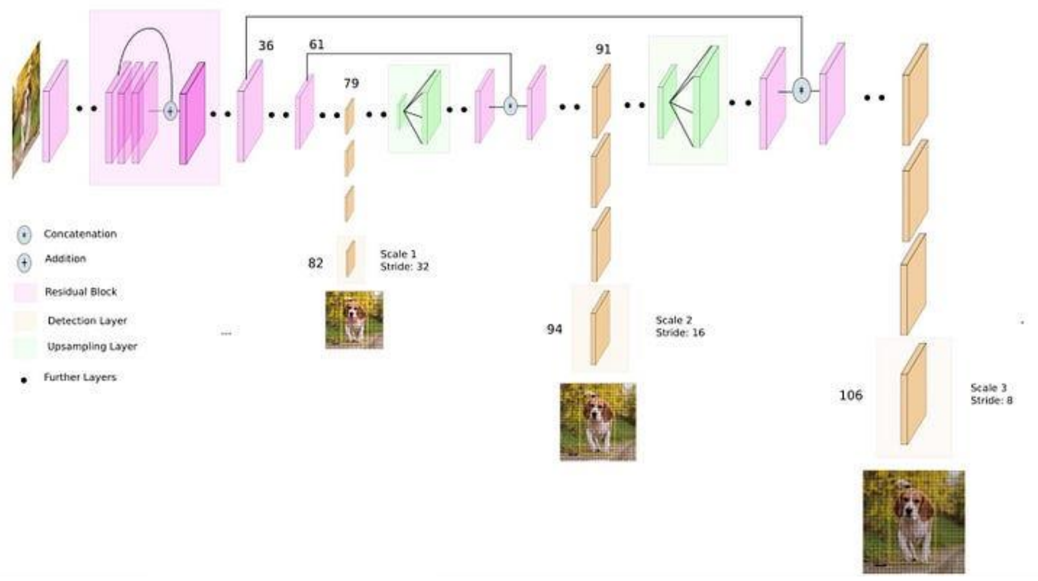
CS6476 Computer Vision

Assignment - Object Detection using YoloV3

In this assignment, you will implement single-stage detector Yolo on the Pascal VOC 2007 dataset.



Source: Codebrew

Installation and Setup

For this assignment, you have been provided with 100 training images from the VOC2007 dataset and 25 test images. There is no need to download the dataset directly, but you can refer to the following link to understand the dataset: http://host.robots.ox.ac.uk/pascal/VOC/voc2007/

Note that the original dataset has 20 data classes, but since we are working with limited compute, we will only use 10 data classes for our assignment.

A brief description of the important boilerplate files and directories is given below:

- VOCtrainval-2007/VOCdevkit/VOC2007 - Training data subset with annotations(XML files), JPEG Images and ImageSets.
- VOCtest-2007/VOCdevkit/VOC2007 - Test data subset with annotations(XML files), JPEG Images and ImageSets.
- train_annotation.txt - Contains xyxy bounding box coordinates, along with object labels forming the Ground Truth labels of the training dataset.
- test_annotation.txt - Contains xyxy bounding box coordinates, along with object labels forming the Ground Truth labels of the test dataset.
- config/yolov3_config_voc.py - Contains configuration parameters, with the sampled 10 data classes, training and test parameters, image size etc.

- test_yolo_voc.py - Runs the trained model on test set, calculates mAP and generates qualitative results
- infer_yolo_voc.py - Inference of trained model on a custom dataset.

More details on running these scripts can be found at the end of the document.

To start with the YOLOv3 implementation, first unzip the given directory and set up a virtual environment in your terminal as follows:

- Install virtualenv: pip install virtualenv
- Create a virtual environment:

```
virtualenv yolo_env -p $(which python)
```

OR, `virtualenv yolo_env -p $(which python3)`

depending on how you use python in your system. This step should create a directory in the current folder called "yolo_env"

- Activate the virtual environment: source yolo_env/bin/activate
- Install the required packages: pip install -r requirements.txt
- Open an IPython kernel with the new environment:

```
ipython kernel install --user --name=yolo_env
```

- Finally open the notebook YOLOv3.ipynb in your browser: `jupyter notebook`.

YoloV3 Architecture:

Comprises of primarily 3 modules:

- Darknet53 backbone
- Feature Pyramid Network (FPN)
- Yolo heads (classification and regression)

Refer to original paper (https://pjreddie.com/media/files/papers/YOLOv3.pdf) for more details

We have already provided the backbone and FPN's architecture in the code. You need to complete

- Writing Yolo heads
- initializing both FPN and Yolo heads with correct parameters
- Completing the YoloV3 forward pass.

Next section contains the assignment modules that are required to be done.

Assignment Modules:

There are 5 parts to this assignment.
1.) Writing Yolo head

2.) Completing the Yolo forward pass

3.) Yolo losses (classification and regression)

4.) Post-processing – Non-maximum suppression (NMS)

5.) Training Yolo and reporting mean average precision (mAP) (ungraded)

More, detailed instructions for each module are provided below.

Assignment Instructions:

1) Writing Yolo head

   Here, you'll implement the logic for converting FPN predictions -> box coordinates (dx,dy,dw,dh), box confidence, and class probablities (both in range [-1,1]) into final box coordinates (x,y,w,h), box confidence and class probabilities (both in range [0,1])

   Head over to ****TODO**** for forward function of **Yolo_head** class. Once you complete that, run the Jupyter notebook's corresponding test to check your implementation

   Note: The autograder will have more comprehensive test case where it will check the predicted values from **Yolo_head** (and not just the range).

   GradeScope Submission:

   Upload **yolo_head.py** to gradescope

2) Writing YoloV3 Forward Pass:

   YoloV3 comprises a darknet53 backbone, Feature Pyramid Network and Yolo heads. The code already contains darknet53 and FPN architecture.

   Now, as all 3 modules are there, head over to ****TODO**** section of **yolov3.py**. Once you complete that, run the Jupyter notebook's corresponding test to check your implementation.

   TODO:

   - Initialize **FPN**() and **Yolo_head**() class in YoloV3() constructor
   - Complete YoloV3()'s forward pass

   GradeScope Submission:

- Change the following import

  ```
  from model.head.yolo_head import Yolo_head
  ```

  to

  ```
  from student_submission import Yolo_head
  ```

  before submission.

  NOTE: This is an important step else your gradescope submission will fail with an import error and any points awarded for this module will not be counted.

- Upload **yolov3.py** to gradescope

3) Yolo losses (classification and regression)

   YoloV3 here is trained using a combination of losses, namely, regression loss (GIOU loss), confidence loss and classification loss. Complete the regression and classification loss (confidence loss is already provided in the code)

   Head over to ****TODO**** section of **yolo_loss.py**, and complete the regression and confidence losses. Once you complete that, run the corresponding Jupyter notebook test to check your implementation.

   ****NOTE**:** The Jupyter notebook only checks type and output range of loss. Please check the correctness of your implementation in Gradescope

   GradeScope Submission:

   - Upload `yolo_loss.py` to gradescope.

4) Non-maxima suppression

   Non-maxima suppression (NMS) is a crucial post-processing step in object detection algorithms, including YOLOv3, to ensure that a single object isn't detected multiple times.

   Head over to **TODO** section of **tools.py** and complete **nms()**.

   TODO:

   - Write NMS algorithm – nms() in tools.py as a post-processing step to eliminate redundant predictions.

   Once you complete that, run the corresponding Jupyter notebook test to check your implementation.

   Gradescope submission:

   - Upload **`utils\tools.py`** file to gradescope.

5) Training YoloV3 on the give VOC2007 train set and report your mAP on VOC2007 test set.
   - Run the corresponding train script from the provided Jupyter Notebook YoloV3.ipynb

NOTE on final Training YoloV3:

- **config/yolov3_config_voc.py** -> Contains all the configuration parameters for this assignment.

The training parameters are already set and you should use this for your initial training run. However, feel free to adjust these parameters -> **BATCH_SIZE, EPOCHS,** optimization parameters like **WEIGHT_DECAY, MOMENTUM,** learning rates **LR_INIT** and **LR_END** for training**.** It is recommended to not change all these parameters simultaneously during training, as it will make things difficult to debug.

Visualizing Results on Test set and Custom Data:

- Visualizing Test Set results:

  Run

  ```
  test_yolo_voc.py --weight_path your_model_path
  ```

  This will calculate the mAP as well as generate qualitative results in results directory.

- **Inference on Custom data:**

Run

```
infer_yolo_voc.py --weight_path your_model_path –visiual
custom_data_path
```

GENERAL NOTE:

- Additional documentation pertaining to all the functions/methods required to completed can be found in form of docstrings in the code. Students are recommended to thoroughly go through them to understand the assignment modules.
- The assignment code should be written between the **“START YOUR CODE HERE”** and **“END YOUR CODE HERE”** comments for any file.

Grading Schema:

The assignment is for 16 points in total.

1) Yolo head (4 pts.)

2) Yolo forward pass (3 pts.)

3) Yolo losses (2 pts.)

4) Non-maxima suppression (2 pts.)

5) Problem set (5 pts.)

Problem Set:

1. Explain the concept of anchor boxes in object detectors. What challenges might arise if they were not used? [0.5 points]
2. Explain the significance of the "mean Average Precision" (mAP) metric in object detection. How is it computed, and why is it particularly important for evaluating object detection models? [0.5 points]
3. Explain the benefits of a one-stage detector like YOLO over two-stage detectors like R-CNNs. [0.5 points]
4. Train your network using the code in the notebook: YOLOv3.ipynb. Report the mAP after varying the following TRAIN hyperparameters (to vary hyperparameters, you will need to change the file: config/yolov3_config_voc.py. Report your observations from the above experiments and provide a justification for the increase/decrease in mAP in 1-2 sentences. Also comment on your choice of values for the below TRAIN hyperparameters and why/why not certain values work. [3 points]
   a. Augment
   b. Momentum
   c. Batch size
5. How would you use the YOLO algorithm to detect objects from a real-time video, such as captured by a self-driving vehicle? (You don’t need to provide an algorithm, just a brief idea of how you would input the video into the network should be fine).[0.5 points]