TDT4265: Computer Vision and Deep Learning

Assignment 4

Håkon Hukkelås
hakon.hukkelas@ntnu.no
Department of Computer Science
Norwegian University of Science and Technology (NTNU)

February 28, 2019

- **Delivery deadline: March 14, 2019, by 2359.**
- **This project count towards 6% of your final grade.**
- You can work on your own or in groups of up to 2 people.
- Upload your code as a single ZIP file.
- Upload your report as a PDF file to blackboard.
- You can choose what framework to use in this assignment. We recommend using python3 with numpy, as the assignment starter code is based on this.
- The delivered code is taken into account with the evaluation. Ensure your code is well documented and as readable as possible.

Introduction

In this assignment, we will take a look at how to evaluate object detection architectures. First, we will implement the standard metric used for object detection since the PASCAL VOC challenge in 2007. Then, we will take a deep dive into how the YOLO network processes its predicted features. There will be no need for GPU resources in this assignment and we recommend you to use numpy for all coding tasks.

Recommended readings

- Section 4.2 of The PASCAL Visual Object Classes (VOC) Challenge. This is the official description of how to calculate the mean average precision for object detection tasks.
- Jonathan Hui's blog post about mean average precision. This blog post is very short, but gives a very simple explanation of mean average precision.
- Andrew Ng's video on non-maximum-suppression
- You Only Look Once: Unified, Real-Time Object Detection, Redmon et. al. This is YOLOv1
- YOLO9000: Better, Faster, Stronger, Redmon et. al.. This is YOLOv2

1 Object Detection Metrics (1 point)

For all previous assignments, we have done classification of images. When measuring the performance of image classification architectures we often use accuracy. For object detection, measuring performance is a more complicated task. First, we need to ensure that we detect the object correctly with a good bounding box. Then, when we have detected the object, we want to make sure that the detected object is classified correctly. When the PASCAL VOC dataset was proposed, they introduced a standard metric to measure the performance of object detection architectures[Everingham et al., 2010]. This metric was the *mean average precision (mAP)*. A thorough walkthrough of how to compute this metric is given in the assignment lecture. Also, we recommend you to take a look at the recommended resources about mean average precision.

In your report, please:

(a) *(0.3 points)* Explain what the Intersection over Union is and how we can find it for two bounding boxes. Illustrate it with a drawing.

(b) *(0.2 points)* What is a true positive? What is a false positive?

(c) *(0.2 points)* Write down the equation of precision and recall.

(d) *(0.3 points)* Given the following precision and recall curve for the two classes, what is the mean average precision?

Precision and recall curve for class 1:
$\text{Precision}_1 = [1.0, 1.0, 1.0, 0.5, 0.20]$
$\text{Recall}_1 = [0.05, 0.1, 0.4, 0.7, 1.0]$

Precision and recall curve for class 2:
$\text{Precision}_2 = [1.0, 0.80, 0.60, 0.5, 0.20]$
$\text{Recall}_2 = [0.3, 0.4, 0, 5, 0.7, 1.0]$

Hint: To calculate this, find the precision with the following recall levels: 0.0, 0.1, 0.2, ... 0.9, 1.0.

2 Implementing Mean Average Precision (2.5 points)

Now, we will implement mean average precision for a single class object detection algorithm. In the assignment files, there is a .json file for both a set of predicted bounding boxes and a set of ground truth bounding boxes. The starter code is given in python and we highly recommend you to use this language. With the assignment files, you will find a file called task2.py and this is where all the code should go for this task. Also, we have provided you a set of tests to simplify the debugging steps for this task. For each subtask you will finish, you can run task2_tests.py and it will check if the function is correct. These tests are written with simple checks and we can't guarantee that they cover every edge case, but we did our best. If you want to implement this in another language you are free to do so, but we will not provide tests for other languages.

Note, for this task we will implement average precision for a single class. If you want to extend this code for a multi-class object detection algorithm, you find the average precision for each class and take the mean over all of these to get the mean average precision.

Implement the following:

(a) *(0.3 points)* Implement a function that takes two bounding boxes, then finds the intersection over union. This is the function `calculate_iou` in task2.py

(b) *(0.3 points)* Implement a function that computes the precision given the number of true positives, false positives, and false negatives. This is the function `calculate_precision` in task2.py.

Implement a function that computes the recall given the number of true positives, false positives, and false negatives. This is the function `calculate_recall` in task2.py

(c) *(0.6 points)* Implement a function that takes in a set of predicted bounding boxes, a set of ground truth bounding boxes, and a given intersection of union threshold, then finds the optimal match for each bounding box. This is the function `get_all_box_matches` in task2.py.

Note, each box can only have a single match and it should be matched with the bounding box with the highest intersection of union ratio. If there is no match that has a higher IoU than the IoU threshold then the bounding box has no match.

(d) *(0.4 points)* Implement a function that finds the precision and recall for all images in a dataset. Then, implement a function that computes the precision recall curve. This is the functions `calculate_precision_recall_all_images` and `get_precision_recall_curve` in task2.py.

(e) *(0.6 points)* Implement a function that computes the mean average precision given a list of precisions and recalls. This is the function `calculate_mean_average_precision` in task2.py

Note, do not change the recall levels used to compute the mAP. This is what is used in standard object detection benchmarks.

If you run task2.py as a python file, the printed mean average precision over the whole dataset should be 0.9066 if you use a IoU threshold of 0.5.

(f) *(0.3 points)* **In report:** Put the plot of your final precision-recall curve in your report.

If you run task2.py, it will save the precision-recall curve to precision_recall_curve.png

3 You Only Look Once (1 point)

You Only Look Once(YOLO) is a light-weight, fast and accurate object detection method that is widely used in a variety of tasks. For this task, read the first paper about YOLO: You Only Look Once: Unified, Real-Time Object Detection[Redmon et al., 2016].

In your report, please answer the following questions:

- *(0.25 points)* For YOLOv1, the authors explicitly specifies two limitations about YOLO regarding spatial constraints and generalization. What are these?
- *(0.25 points)* **True or False:** YOLO utilizes a sliding-window approach to detect objects in the image.
- *(0.25 points)* What is the difference between FastYOLO and YOLO?
- *(0.25 points)* How does YOLOv1 compare to Faster R-CNN? More specifically, what is the strong point and the weak point of Faster R-CNN, with respect to YOLO?

4 Object detection with YOLO (1.5 points)

In this problem, we will take a deep dive into how the YOLO framework process the features extracted by a deep convolutional neural network. This task is based on an online machine learning course by Andrew Ng and deeplearning.ai. The specific network we are going to work with is the YOLOv2. The necessary description to complete this task is given in the notebook, but if you want to get a better understanding of this network, we recommend you to read the paper about YOLOv2[Redmon and Farhadi, 2017].

The structure of this task is very similar to task 2. We have given you a starter code in task4.ipynb. This is a jupyter notebook file and we highly recommend you to use this notebook. It is clearly marked where you should edit your code and what the expected output for each function should be.

Note, the iou function in the jupyter notebook is the intersection over union function that you implemented in task2. You can either copy paste the implementation from task2, or just write
`from task2 import calculate_iou as iou`.

Implement the following:

(a) *(0.45 points)* Implement a function that takes a set of predicted bounding boxes with its corresponding bounding box confidence and class confidences, then filter out all the boxes that is below a given threshold. This is the function `yolo_filter_boxes`.

(b) *(0.5 points)* Implement a function that takes in a set of bounding boxes and apply non-maximum suppression to filter out excess bounding boxes. This is the function `yolo_non_maximum_supression`.

(c) (*0.35 points*) Implement a function that includes the whole YOLO pipeline for a set of features extracted from our deep convolutional network. This should perform both confidence score filtering and non-maximum suppression. This is the function `yolo_eval`.

(d) *(0.2 points)* **In report:** Run `yolo_eval` on the image test.jpg. Write down the classified bounding boxes with scores and class name in your report. Include the final image with bounding boxes and class names in your report.

Delivery

Write a report detailing the work you have done. Include all tasks in the report, and mark it clearly with the task you are answering (Task 1.1, Task1.2, Task 2a etc). The report should be in a single PDF file.

Include all the necessary code in a single ZIP file. Make sure that your code is well documented and easily readable. Do not include any unnecessary files or folders.

Do not change the name of the given assignment files and functions. This will make it easier for us to read your code.

If you want to deliver your report directly from a jupyter notebook, first ensure that all cells are executed and has visible output. Export this as a PDF file and deliver both the exported PDF file to blackboard, and the code (as .ipynb or .py file(s)) in a ZIP file.

References

[Everingham et al., 2010] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338.

[Redmon et al., 2016] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

[Redmon and Farhadi, 2017] Redmon, J. and Farhadi, A. (2017). Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271.