BBM 418 - Computer Vision Laboratory

HACETTEPE UNIVERSITY

Department of Computer Engineering

Programming Assignment 4
Spring 2023-2024
Dr. Nazli Ikizler Cinbis

Due Date: 23:59pm on Thursday, May 23rd, 2024

Object Detection with Custom Overfeat and Pre-trained YOLO Algorithms

In this assignment, you will fine-tune YOLO and implement a custom Overfeat algorithm for object detection from the dataset provided Guns Object Detection dataset [1] given in Figure **??**. The goals of this assignment are as follows;

1. Understand Object Detection algorithms and build an Overfeat model from scratch then train on the dataset provided.
2. Understand and implement fine tuning using a pre-trained YOLO model.
3. Analyze your results.
4. Gain experience with PyTorch, a popular deep learning framework.

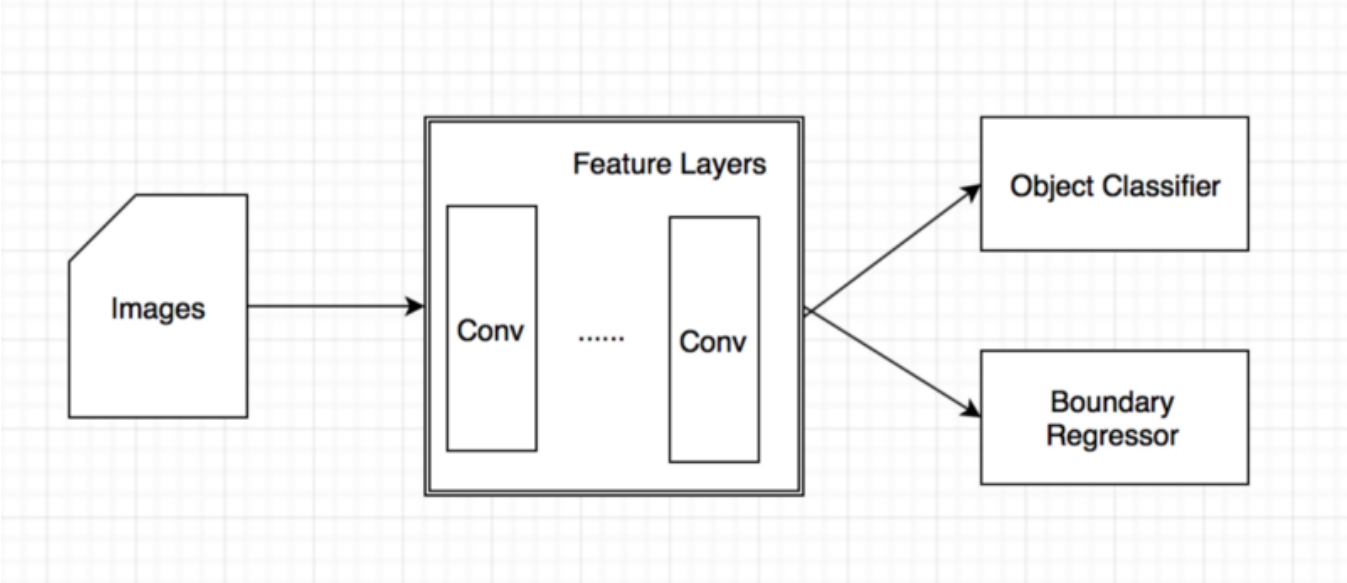

Figure 1: Classification and Localization flowchart

Dataset

Guns Object Detection dataset contains 333 images collected from Google Images in jpeg format. Labels folder has txt files where in each file, the first line contains the number of objects in the corresponding image and the next lines contain the co-ordinates of the box describing the object.

In this assignment, you will apply classification and localization to images in one of the 2 categories (either background or gun). You need to divide your dataset into train, validation, and test sets. **Explain how you arrange your dataset into train, validation, and test and write how many images are there in the sets. [5 pts]**

Figure 2: Gun Image with Bbox Sample. [1]

PART 1 - Implementing a Overfeat Algorithm from Scratch [55 pts]

In this part, you are expected to model an Overfeat Object Detector and train it on the dataset provided. You should first define the components of model design. You have two options either to use the first 5 layers of Alexnet or the Efficient-net used in the Assignment-3 as backbone. Then, the model will have two heads of fully connected layers one for object classifier and one for boundary regressor. So per each window, the classifier will have 2 outputs for two classes and regressor will have 4 outputs for the localization. Note that the window size is an important parameter, that will affect the number of detections. Make sure to set it properly. After that you need to apply greedy merge strategy to combine the same classes. **Here** is a nice video to have a better understanding of the model architecture.

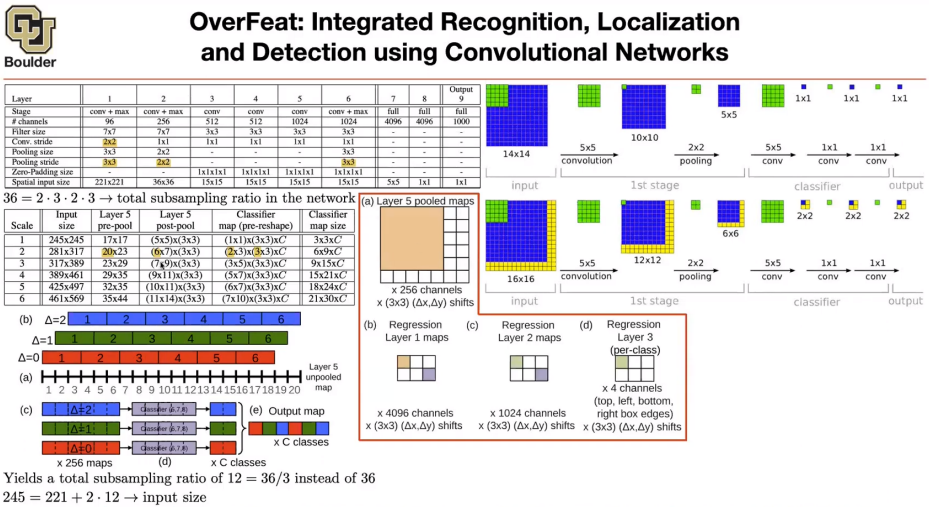

Figure 3: Overfeat Algorithm [2].

Implementation Details

1. Begin with the backbone implementation as either first 5 layers of the AlexNet architecture or Efficient-net used in the Assignment-3 as the base feature extractor. [10 pts]

2. Followed by fully connected layers: One for object classification with 2 outputs per window(for binary classification). Another for boundary regression with 4 outputs per window (for localization).[10 pts]

3. Define a sliding window approach. [5 pts]

4. Use softmax loss as classification loss. [2.5 pts]

5. Use L2 loss as bounding box regression loss. [2.5 pts]

6. Sum softmax loss and L2 loss to obtain total loss for each anchor in minibatch. [5 pts]

7. Implement a non-maximum suppression (NMS) technique to refine and consolidate the final set of detections. [5 pts]
8. Try different hyperparameters (learning rate, batch size) and report their performance over the validation set. [10 pts]
9. Finally report the final performance (mIoU) and classification accuracy over the test set, for the best performing model on the validation set. [5 pts]

PART 2 - Transfer Learning with Yolo [40 pts]

YOLO (You Only Look Once) is a state-of-the-art real-time object detection algorithm known for its efficiency. YOLO divides the input image into a grid and predicts bounding boxes and class probabilities directly from this grid. **Here** is an explanatory video to understand the logic behind the architecture.

YOLOv3-tiny model is one of the standart variants of the YOLO object detection model designed for real-time inference with reduced computational complexity. In the assignment part 2, you will be training the given dataset using the pre-trained YOLOv3-tiny model, which is available at Pytorch. However, it's important to clarify that the fine-tuning process involves adapting the pre-trained model's weights to the dataset given while keeping the initial weights from its starting point. Therefore, while the model is pre-trained on the COCO (Common Objects in Context) dataset containing over 330,000 images spanning 80 different classes, it will be fine-tuned on the given dataset to improve its performance for the specific detection task at hand.

To train the pre-trained model, you need to freeze all the layers before training the network, except the the last three layers. Consequently, the gradients will not be calculated for the layers except the ones that you are going to update (the last three layers) over the pre-trained weights. Nevertheless, since the number of classes is different for our dataset, you should modify the last layer of the network, which the probabilities will be calculated on. Therefore, the weights will be randomly initialized for this layer to adapt to the new classification and localization tasks.

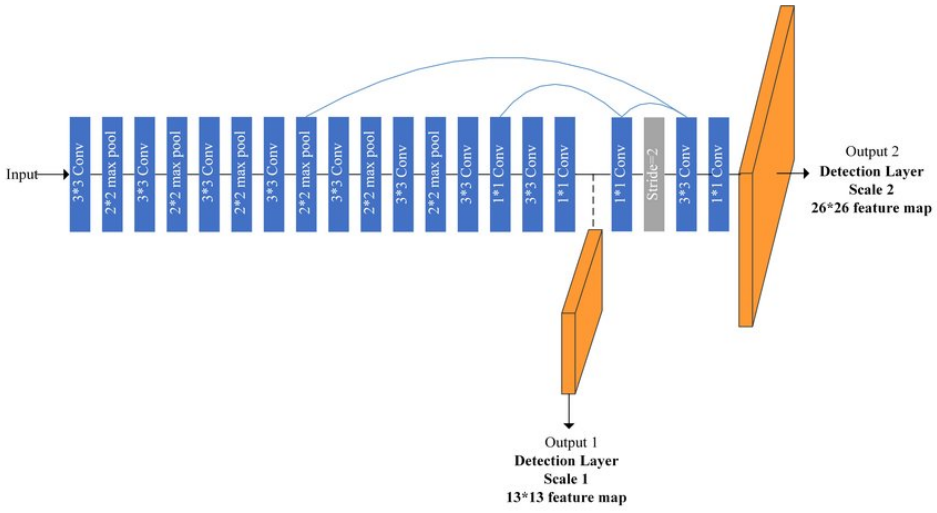

Figure 4: YOLOv3-tiny Model Architecture [3].

Implementation Details

1. Modify the last layer of YOLOv3-tiny to accommodate the number of classes required for object detection. [10 pts]
2. Freeze all layers except the last three layers of YOLOv3-tiny model.
3. Use softmax loss as classification loss. [2.5 pts]
4. Use L2 loss as bounding box regression loss. [2.5 pts]
5. Sum softmax loss and L2 loss to obtain total loss for each anchor in minibatch. [5 pts]
6. Implement a non-maximum suppression (NMS) technique to refine and consolidate the final set of detections. [5 pts]

7. Try different hyperparameters (learning rate, batch size) and report their performance over the validation set. [5 pts]
8. Finally report the final performance (mIoU) and classification accuracy over the test set, for the best performing model on the validation set. [5 pts]
9. Compare the results of Part1 and Part2. [5 pts]

General Implementation

1. You should pay attention to code readability such as comments, function/variable names and your code quality: 1) no hard-coding 2) no repeated code 3) cleanly separate and organize your code 4) use consistent style, indentation 5) write deterministic algorithms
2. Implement your code with Python and install all needed libraries.
3. You should use the PyTorch as the deep learning framework. You can use Google Colab or Anaconda to run your experiments.

What should you write in the report?

- Give explanations for each step.
- Give experimental/visual results, used parameters and comments on the results in detail.
- Give your model's loss plot both for training and validation set during training.
- Put the results of different hyper-parameters (learning rate, batch size), the effect of them, with the loss plots and visualized bounding box predictions.
- Compare and analyze your results in Part-1 and Part-2. Please state your observations clearly and precisely.
- A basic structure might be: 1) Introduction (what is the problem, how do you approach to this problem, what is the content of your report) 2) Implementation Details (the method you followed and details of your solution) 3) Experimental Results (all results for separate parts with different parameters and your comments on the results) 4) Conclusion (what are the results and what are the weaknesses of your implementation, in which parts you have failed and why, possible future solutions)
- You should give visual results by using a table structure.

What to Hand In

Your submission format will be:

- main.ipynb *(a notebook file containing your code, the report and a drive link for the weights of your models)*

Archieve this folder as **b<studentNumber>.zip** and submit to `https://submit.cs.hacettepe.edu.tr`.

Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.

References

[1] `https://www.kaggle.com/datasets/issaisasank/guns-object-detection`

[2] `https://arxiv.org/pdf/1312.6229v4`

[3] `https://www.analyticsvidhya.com/blog/2021/06/implementation-of-yolov3-simplified/`