CIS680: Vision & Learning
Assignment 2B: RPN, Faster R-CNN and Mask R-CNN
Due: Nov. 1, 2018 at 11:59 pm

Instructions

- This is an **individual** assignment. "**Individual**" means each student must hand in their **own** answers, and each student must write their **own** code in the homework. It is admissible for students to collaborate in solving problems. To help you actually learn the material, what you write down must be your own work, not copied from any other individual. You must also list the names of students (maximum two) you collaborated with.
- There is no single answer to most problems in deep learning, therefore the questions will often be underspecified. You need to fill in the blanks and submit a solution that solves the (practical) problem. Document the choices (hyperparameters, features, neural network architectures, etc.) you made in the write-up.
- The assignment will describe the task on a high level. You are supposed to find out how to complete the assignment in the programming framework of your choice. While the text of the assignment should be sufficient to understand the task, you are welcome to read the references that will describe the used concepts in more detail.
- All the code should be written in Python. You should use either Tensorflow or PyTorch to complete this homework.
- You must submit your solutions online on **Canvas**. Your should submit **3 folders with code**, one for each part. Submit your code compressed into a ZIP file named "1_<penn_key>.zip". Jupyter notebooks are acceptable. Submit your **PDF report** to a separate assignment called "HW2.b PDF Submission". Note that you should include all results (answers, figures) in your report.

Overview

This homework is a step-by-step guide for implementing Mask RCNN, which you have seen the first homework. We will walk you through feature extraction, object localization, classification and segmentation. You will implement a simplified version of Mask R-CNN [2].

This homework consists of two parts.

1. First build Region Proposal Network (RPN) on top of the given base network architecture with a object classifier that determines if a proposed region is object or not and a bounding box regression branch to predict exact object location.
2. Finally use ROI pooling to obtain features from proposed region and add a classifier to determine the class identity of the detected object. By now you have a simplified but proper Faster R-CNN [1], add a third instance segmentation branch to produce per-pixel labeling of class.

Note that the full training of a network takes much time using only CPUs. You should observe the trend of training over the first couple hundreds of iterations and decide whether to finish training or not.

Layers	**Hyper-parameters**
Convolution 1	Kernel size = (3, 3, 8). Followed by BatchNorm and ReLU.
Pooling 1	Max operation. Kernel size = (2, 2). Stride = 2. Padding = 0.
Convolution 2	Kernel size = (3, 3, 16). Followed by BatchNorm and ReLU.
Pooling 2	Max operation. Kernel size = (2, 2). Stride = 2. Padding = 0.
Convolution 3	Kernel size = (3, 3, 32). Followed by BatchNorm and ReLU.
Pooling 3	Max operation. Kernel size = (2, 2). Stride = 2. Padding = 0.
Convolution 4	Kernel size = (3, 3, 64). Followed by BatchNorm and ReLU.
Pooling 4	Max operation. Kernel size = (2, 2). Stride = 2. Padding = 0.
Convolution 5	Kernel size = (3, 3, 128). Followed by BatchNorm and ReLU.

Table 1: base network.

1 Region Proposal Network (40%)

In this part, you start building Faster RCNN by first building a very important component of it (Figure 1): Regional Proposal Network (RPN). The base network/feature extractor architecture is specified in the table above. Region proposal network runs on top of the

base network and localizes objects in an image. You will implement a simplified version of region proposal network which consists of only one set of anchors (as opposed to nine sets in the paper). It's highly recommended to read the paper [1] [2] beforehand. Starting from this part, use a synthetic pedestrian and car dataset (P&C) for experiments.

Pedestrian and Car Dataset

The P&C dataset consists images of pedestrians and cars (cropped from COCO dataset) pasted on synthetic backgrounds.

Images: Images are of size $(128, 128)$. All images can be found under the `image` folder. Each image is named after one index (like `000000.png`) using which you can access its corresponding annotations. The image are cropped from COCO dataset and is pasted on some random background. Each image has pedestrians and cars. But the number of valid object in each image is less than 2.

Annotations: The annotations for each image consists bounding box and segmentation mask for both people and cars. The annotations for the two categories are saved separately in `people` and `cars` under `label`.

Bounding boxs are parametrized by (x, y, w, h), namely column index, row index, width, height and saved in `label_car.txt` and `label_people.txt`. For example, to access the bounding box of car for image `123456.png`, reading the 123456th line in `label_car.txt` and parse it into (x, y, w, h). Notice that (x, y) is the position of the **upper left corner** of the bounding box. You will want to compute the center of the box from (x, y, w, h) for training.

The segmentation masks are saved as binary masks for each class. For image `123456.png`, one can find car masks `/label/car/123456.png`. See one example of the image and labels in figure 2.

1. (20%) Build a base network as shown in Table 1

   Use the features from the base network to build a proposal classifier (as the cls branch in Figure 1) that predicts how confident a region contains objects. Specifically, add a standard convolution layer (referred as the intermediate layer) with kernel size (3, 3, 128) (followed by BatchNorm and ReLU) and a convolution layer with kernel size (1, 1, 1) with no rectification layer.

   You need to use the location of the ground truth bounding box to generate a 0-1 mask that indicates the "objectness" of a region. The pixel value of this mask is set to be 0 if anchor box at this location has less that 0.1 IoU with any ground truth object bounding box and is set to be 1 if IoU is greater than 0.5. We simply ignore

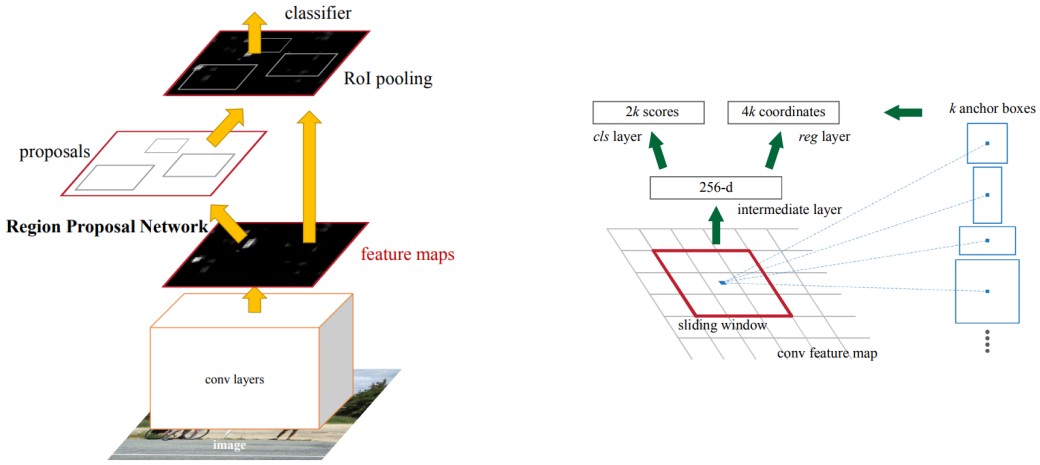


Figure 1: Faster R-CNN and its region proposal network.

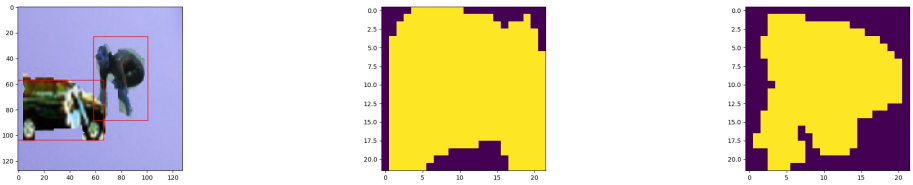


Figure 2: From left to right: image and bounding boxes, mask for car, mask for person

the regions for which IoU is between 0.1 and 0.5. To do this, one suggested way to compute another mask to zero out the loss in such regions.

Plot the training loss over training iterations. Report the (point-wise) test accuracy of the proposal classifier.

2. (20%) In this question, you will build a proposal regressor (as the reg branch in Figure 1). On top of the intermediate layer, add another convolution layer with kernel size (1, 1, 4) (without BatchNorm and rectification). Note that you should initialize the biases to be (64, 64, 128, 128) for each channel. The first two channels are for the row/column coordinates of the object center and the third and fourth channel for the width and height of the object.

   Parameterize the coordinates as follows:

$$t_x = (x - x_a)/w_a,\ t_y = (y - y_a)/w_a,\ t_w = \log(w/w_a)$$

$$t_x^* = (x^* - x_a)/w_a,\ t_y^* = (y^* - y_a)/w_a,\ t_w^* = \log(w^*/w_a)$$

where x, y, and w, h denote the box's center coordinates and its width. Variables x, x_a, and x^* are for the predicted box, anchor box, and groundtruth box respectively (likewise for y and w).

The regressor is trained with the Smooth L1 loss defined as:

$$L_{reg}(t_i, t_i^*) = \frac{1}{N_{reg}} \sum_i \text{smooth}_{L1}(t_i - t_i^*)$$

where

$$\text{smooth}_{L1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise.} \end{cases}$$

Note that the regressor loss is computed only on where the proposal masks equal to 1.

Train the whole network with two equally weighted losses from the proposal classifier and regressor. The training procedure is the same as the previous question.

Plot the training regression loss over training iterations. Report the final testing regression loss.

2 Faster R-CNN, Mask R-CNN (30%)

With the base network and region proposal network ready, you are now only one step from completing Faster R-CNN. After finishing the whole pipeline of Faster R-CNN, you can also change the base network to see the effect of different learned features.

1. (10%) One important layer that transforms the features of the proposal into the final object classifier is the ROI pooling layer. The ROI pooling layer warps the features of an ROI region into a fixed-sized feature map. H

   To save some effort, we provide a spatial transformer layer that can be used as ROI pooling layer. Check the usage in the Python file (spatial_transformer.py).

   10% extra credits will be given if you program the ROI pooling layer without using spatial transformer layer or any other bilinear sampling layer.

   The parameter θ to be fed into spatial transformer layer is as follows:

   $$\theta[:, 0] = w/128,\ \theta[:, 1] = 0, \theta[:, 2] = (x - 64)/64,$$

   $$\theta[: 3] = 0,\ \theta[:, 4] = w/128,\ \theta[:, 5] = (y - 64)/64$$

where x, y, w are the predicted box's column/row coordinates and width. Note that the predicted box is the maximally activated proposal.

Feed the original images into the spatial transformer layer with output resolution $(22, 22)$. You should get an image in which it has the bounding box region cropped and magnified.

2. (10%) Use the same θ parameter for the spatial transformer layer; however, feed in the feature map (conv4) from the base network and set the output size as $(4, 4)$.

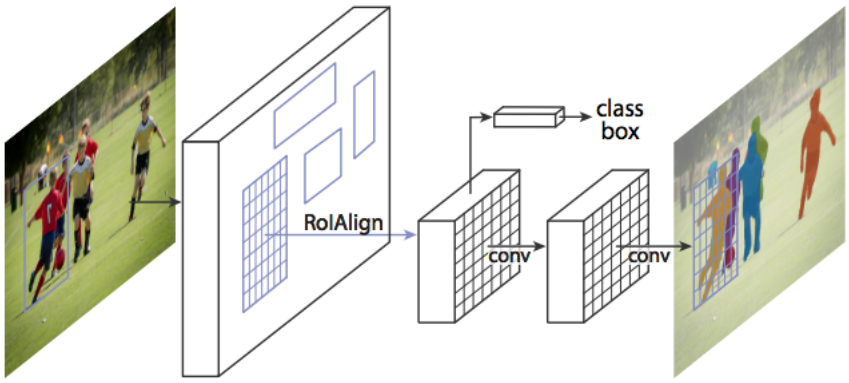
   Add two fully convolution layer with 128 output channels (followed by BatchNorm and ReLU) and a 2-way Softmax layer (with fully connected layer) for classification of people and cars.

   Train the network (consisting of proposal classifier, regressor, and object classifier) end-to-end with the same training procedure.

   Plot the 3 training losses over training iterations. Report the final classification test accuracy.

3. (10%) Now you can add one more branch to implement Mask R-CNN. On top of the ROI features, add two fully convolution layer with 128 output channels (followed by BatchNorm and ReLU) and one convolution layer with kernel (1, 1, 2) (without BatchNorm and ReLU) with sigmoid layer that predicts pixel labeling, meaning that for each class (person or car) we want to predict a binary mask. Use a binary cross entropy loss for the loss of each class binary mask. At inference time, output the class binary mask according to the classification result from the cls branch (implemented earlier).

   Here is a figure showing the added mask head and how it processed features from a region of interest:



Plot the training losses over training iterations. Report the final classification test accuracy.

References

[1] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[2] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.