Project III: Fast R-CNN for object detection

Due by Nov. 27

1 objectives

Recently, deep CNN have significantly improved image classification and object detection accuracy. Compared to image classification, object detection is a more challenging task that requires more complex methods to solve. Due to this complexity, current approaches train models in multi-stage pipelines. Fig. 1 shows examples of single-class and multi-classes object detection task. In this project, we learn a popular model for object

Figure 1: Examples of object detection. The left figure is single-class detection while the right figure is multi-classes detection.

detection, Fast R-CNN. A Fast R-CNN network takes as input an entire image and a set of object proposals. The proposals are obtained by using a region proposal algorithm as a pre-processing step before running the CNN. The proposal algorithms are typically techniques such as EdgeBoxes or Selective Search, which are independent of the CNN. The network first processes the whole image with several convolutional (conv) and max pooling layers to produce a conv feature map. Then, for each object proposal a region of interest (RoI) pooling layer extracts a fixed-length feature vector from the feature map. Each feature vector is fed into a sequence of fully connected (fc) layers that finally branch into two sibling output layers: one that produces softmax probability estimates over K object classes plus a catch-all background class and another layer that outputs four real-valued numbers for each of the K object classes. Each set of 4 values encodes refined bounding-box positions for one of the K classes. Fig. 2 illustrates the Fast R-CNN architecture. For more details, please refer to `https://arxiv.org/pdf/1504.08083.pdf`.

In this project, we use pre-trained fast R-CNN model to do object detection task.

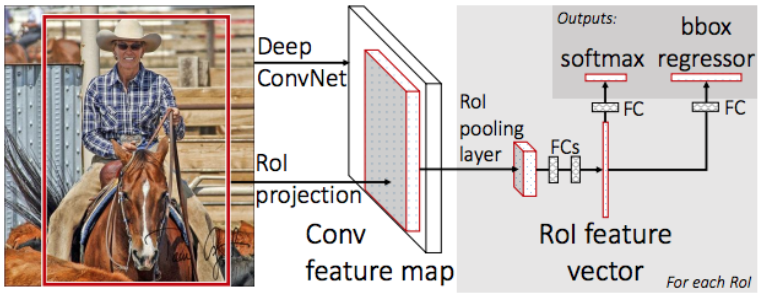

Figure 2: Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

2 The project includes the following two parts

2.1 Single-class object detection in one image

Use pre-trained Fast R-CNN model to do object detection in one example image. You need to carefully select the threshold of probability that a RoI is accepted as a detection. Only output the detection of `car` class. **Plot the number of detections in the image over the value of threshold. Report your finally chosen threshold. Visualize the detected bounding boxes and the corresponding probability score of every bounding box in the image.**

2.2 Object detection on Pascal VOC 2007 dataset

Use the same model to do object detection on testing dataset of Pascal VOC 2007. There are 20 classes in the dataset. Use the same threshold as you choose in the first part. For each detection in a image, we compare it with ground truth annotations of the images. If there exists an annotation which has an ($> 50\%$) overlap with the detection, we define the detection as a true positive. **Show one example that contains true positive detections of multi-classes.** To quantitatively evaluate the detection results, **plot the precision-recall curve for `car` class. Report the average precision of every category and calculate the mean average precision (MAP) over the 20 classes**.

3 Data

You need to resize the image as well as the RoIs to fit the input size of the model (set the shorter edge of the input image to 600). Before testing, remove the average color from the input image by subtracting `net.meta.normalization.averageImage`. Do some non-maximum suppression (NMS) to your detections, i.e. when two positive

detections overlap significantly, choose the one that has higher score. Set a maximum number of detections per image before NMS.

For 2.1, use image `./example.jpg` and proposals `./example_boxes.mat`.

For 2.2, the images are in `./data/images/` and the proposals are in `./data/SSW/`. The ground truth of bounding boxes are in `./data/annotations/`. The dataset contains 4,952 images with annotations of objects from 20 classes. Use `PASreadrecord.m` in `./code/fast_rcnn/` to read the ground truth annotations.

4 Code

1. We use `matconvnet` to implement this project. To compile `matconvnet`, run `./code/fast_rcnn/Setup.m`.
2. The pre-trained model is in `./data/models/`. After you load the model into matlab, run `./code/fast_rcnn/preprocessNet.m` to preprocess it.
3. Use `net.eval({'data', image, 'rois', RoIs})` to evaluate the output of the model. `image` is the single matrix of image. `RoIs` is a $5 \times N$ matrix: the first row are 1, the last 4 rows are positions of proposals and N is the number of proposals. In `net.vars`, `'cls_prob'` layer records the output of softmax layer and `'bbox_pred'` layer records the output of bounding box regressor. Use function `bbx_transform_inv.m` in `./code/fast_rcnn/` to get predicted bounding boxes.