Object Detection

computer vision tasks:

- classification
- semantic segmentation
- object detection
- instance segmentation

Regression Loss

- Error: $(\Delta x, \Delta y, \Delta w, \Delta h)$
- L1 loss: Σ|Δ!| — robust, however not good at convergence (gradient is not smooth at 0)
- L2 loss: ΣΔ! (not the same to L2 norm) — not robust to a larger error (gradient is too large when error is large), however good at convergence (gradient is smooth and nearly 0 at 0)
- Rooted mean squared loss (RMSE): $\sqrt{\frac{1}{N}\sum\Delta_i^2}$ — the gradient of sqrt function is bad at 0
- Smooth L1 loss:

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 if|x| < 1 \\ |x| - 0.5\text{otherwise} \end{cases}$$

object detection

A naive solution for object detection is to apply CNN to many different crops of the image, and the CNN will classify each crop as object or background. The problem is that we need to apply CNN to huge number of locations, scales and aspect ratios, very computationally expensive.

So we have Region Proposals come in here. Region proposal will try to spot regions that are likely to contain objects.

R-CNN

1. pipeline: for input image, generate (~2k) region proposals (e.g. selective search, not network, a classic algorithm), then wrap these regions of image into a fixed size (e.g. 224x224, by interpolation) and feed them into a CNN, CNN will regress the correction to the ROI(dx, dy, dw, dh) and the classified label.
2. problem
   i. very slow! Need to do ~2k indenpendant forward passes for each image
   ii. the cropped region does't contain sufficient information to regress bounding box refinement, because the context is not included.

Fast R-CNN

1. pipeline:
   i. the input image first goes through a CNN to generate feature maps,
   ii. then we use region proposal on original image and project proposed regions to the feature map to generate a set of ROIs on the feature map(the proposed region may not sit exactly on grids of the feature map, so we just snap the region to the nearest grid).
   iii. The cropped features are resized into fixed size (e.g channelx7x7 by ROI Pool, which just divide the region approximately into 7x7 subregions, and do max pooling on them)
   iv. cropped and resized features are put into another CNN which will classify the object and regress the bounding box refinement.
2. advantage:
   i. because the input of the second network is much smaller than the original size of the image, which enables batch process, the computation is much faster.
   ii. the cropped region now contains sufficient information to regress bounding box refinement, because the perception field of the feature contains the context of the object.
3. problem:

    i. the region proposal algorithm is slow and becomes the bottleneck of the network.
    ii. the feature snap process introduce unexpected error because the boundary refinement is actually based on the given proposed region, but it is calculated using features after align with the grid, which makes the boundary of the object hard to align with pridiction.

Faster R-CNN

1. pipeline:
    i. put the image into a CNN to generate feature maps,
    ii. use RPN(Region Proposal Network) to generate region proposals:
        - at each point of the feature map, K anchor boxes are generated, each box has a different aspect ratio and size.
        - for each anchor box, we predict a probability of being an object(objectness) and a bounding box refinement (dx, dy, dw, dh)
        - sort all the anchor boxs by their 'objectness' score and select the top N (~300) to be the final proposals.
    iii. for each proposed region, we use RoI Pooling to crop the feature map and resize it into fixed size (e.g channelx7x7)
    iv. cropped and resized features are put into another CNN which will classify the object and regress the bounding box refinement.
    v. perform confident thresholding to filter out the proposals with low classification score.
    vi. do NMS(Non-Maximum Suppression) to remove overlapping proposals. NMS Algorithm:
        - Initially D is empty
        - Select the proposal with highest confidence score, remove it from B and add it to the final detection list D.
        - Now compare this proposal with all the proposals — calculate the IoU of this proposal with every other proposal. If the IOU is greater than the threshold, remove that proposal from B.
        - Again take the proposal with the highest confidence from the remaining proposals in B and remove it from B and add it to D.
        - Once again calculate the IOU of this proposal with all the proposals in B and eliminate the boxes which have a IoU higher than the threshold.
        - This process is repeated until there are no more proposals left in B.
2. advantage: fast
3. problem: hard training
    - the RPN and the CNN used for classification are trained separately, maybe to solve the distribution shift problem.

evaluation metric: AP(Average Precision)

1. method:
    i. Per category rank the output bounding boxes according to the confidence (classification score) in a descending order.
    ii. Select top n outputs and compute recall.
    iii. Precision: the ratio of bboxes that satisfy IoU > x% threshold
    iv. Compute the area under precisionrecall curve (approximate by taking 11 points at the recall levels, 0.0, 0.1, 0.2,..., 1.0, and calculate the average of corresponding precision values).
    v. average the AP across all categories, and we get mAP.
    vi. under different IoU thresholds, we can get different AP values, which is reported as $AP_{50}, AP_{60}, ...$

YOLO

single stage detector which is not as accurate as Faster R-CNN but much faster. Similar works: SSD/RetinaNet

1. pipeline: within each grid cell, regress from each of the B base boxes to a final box(predict dx, dy, dh, dw, confidence). Predict scores for each of C classes(including background as a class).

instance segmentation

two approaches:

1. top-down approach: object detection and the further find a binary mask inside the bounding box.
2. bottom-up approach: group together similat points and then classify each group

Mask R-CNN (top-down approach)

1. pipeline: almost the same as faster R-CNN, but the another CNN is used at the end to predict the binary mask inside the bounding box.
2. implementation tricks:
    i. the region proposed no longer need to snap to the grid, interpolation is used to better fit the boundary, then we use max pooling to resize the feature into fixed size.

  ii. instead of only predict the mask of the object class being predicted, mask R-CNN predicts the mask of all the classes. And instead of use softmax to classify each pixel, signoid is used to avoid competition among classes. This helps to capture the shape knowledges of each class.

3D object detection

1. Frustum PointNet: input RGBD, do region proposal and classification in RGB image, and use pointnet to do point segmentation and than use another pointnet to predict the 3D bounding box for the segmented objects points(usually part of the object in the point cloud seen by the camera).
2. Monocular 3d object detection: sample proposal in 3D, than project to 2D space to do scoring and NMS.
3. Deep Sliding Shape: use sliding window like faster R-CNN to do region proposal, but in 3D space. Then use sparse conv instead of pointnet for the network.
4. Deep Hough Voting: set seeds, and let points vote for the seed they belong to. Then use pointnet on the vote clusters to regress the 3D bounding box.

3D instance segmentation

Top-Down method

- GSPN

Bottom-Up method

- SGPN
- PointGroup