Rich feature hierarchies for accurate object detection and semantic segmentation

CNN can be utilized in PASACAL VOC by two factors. First, CNN can be utilized in region proposal to locate and segment the object. Second, when labeled training data is scarce, supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning, yields a significant performance boost.

There are two problems to be solved. First, localizing objects with a deep network. Second, training a high-capacity model with only a small quanitity of annotated detection data.

Model design: object detection model consists of three modules. The first generates category-independent region proposals. The second module is a large convolutional neural network that extracts a fixed-length feature vector from each region. The third module is a set of class-specific linear SVMs.

Test time detection: First, the module selectively search 2000 regions for proposals, and warp them into an extent of 277*277, which will be take as an input for CNN to extract features and will be used for SVM classification. Second, the module delete the highly ouverlapping boundings by greedy non-maximum suppression algorithmus. R-CNN is efficient, because first, all CNN parameters are shared across all categories. Second, the feature vectors computed by the CNN are low-dimensional when compared to other common approaches. The result for that is, even though there is 100K classifications, an image will take only 10 seconds using CPU.

Training:
Supercied pre-training: train CNN on ILSVRC 2012 with image-level annotations, framework Caffe. Very fast.
Domain-specific fine-tuning: Start SGD at a learning rate of 0.001 (1/10 of the initial pre-training rate), which allows fine-tuning to make progress while not clobbering the initialization. See IoU > 0.5 as a boundary.
Object Category classifiers: For a class, it is hard to tell if the object contained in a region with middle-value IoU. They set a threshold 0.3, and optimize for every classification a SVM. Due to amount of negative sampling, they also use hard negative mining.

Fast R-CNN:

Fast R-CNN is developed based on former works in R-CNN and SPPnet.

R-CNN has following disadvantages so far. First, Training is a multi-stage pipeline. (object proposals -> SVM algorithim -> bounding-box regression). Second, Training is expensive in space and time. Expensive in space means that features extracted from each proposal will be written to disk, whereas expensive in time means we need extract features using CNN seperately. Third, Object detection is slow. At test-time, features are extracted from each objects proposal in each test image.

SPPnet also has drawbacks. 1. Training is a multi-stage pipeline 2. Features are written to disk 3. The fine-tuning algorithm cannot update the convolutional layers which limits the accuracy.

Fast R-CNN has following advantages: 1. mAP higher than R-CNN, SPPnet. 2. Training is single-stage, using a multi-task loss. 3. Training can update all network layers. 4. No disk storage is required for feature caching.

Fast R-CNN has an architecture like R-CNN, it generates propoasls but it doesn't process them individually. The input of the net is whole image and its proposals' location. For each proposal , a RoI pooling layer will extract a fix-sized feature vector from feature map. Then the feature vector will be input into the fully connected layer and proffer two outputs, one for softmax probabilities, another one for 4 codes for fining bounding-box regression offsets.

Fast R-CNN has following tricks.

RoI pooling layer convert features in each proposals into a small feature map with a fixed extent of H*W (H, W: hyper-parameters). An RoL is a rectangular window defined by (r, c, h, w) . r,c refers to left-top corner location, while h, w window can divide feature map into sudmap with extent of h/H * w/W.

Pretraining: the utilization of trained networks has step-by-step replacement: 1. Max pooling -> RoI 2. Fully convolutional Network -> two branches mentioned above 3. Image input -> image and proposal input

Fine-tuning: N images -> R/N RoI for fine-tuning, because it can share computation and memory.

Mutli-task loss: Fast R-CNN loss is combined with weo parts: RoI classification loss, and bounding location loss. But here I cannot understand where we have L1 and L2.

Mini-batch sampling

Back-propagation through RoI pooling layers

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

After R-CNN and Fast R-CNN has been proposed, Faster R-CNN improves the efficiency and accuracy of object detection further by replacing the time-consuming selective search algorithm with RPN algorithm and achieving end-to-end object detection training.

Faster R-CNN has following structure. First, by pretraining convolutional network, compute the feature maps of input images. Second, take the feature maps as inputs for Fast R-CNN and RPN for sharing feature maps. Third, RPN proffer 2000 proposal regions through praining with the feature maps. Fourth, Fast R-CNN use 2000 proposal regions to make classifications and bounding box regression, and fine-tuning the network. Fifth, take the pretrained convolutional network with fine-tuning as an initialization of RPN network. Repeat the steps mentioned above till the process ends.

RPN has following structure. They get feature maps in Step 1. RPN extracts features from every position by sliding a 3*3 window. The features they have will be convoluted by two 1*1 conv layers, 1*1*2 conv layer for classification, 1*1*4 conv layer for bounding box regression. Since the objects has different shapes, Faster R-CNN introduce the concept of anchor box. The center point of anchor box is the center point from sliding window in input images. The journal has designed nine different anchor boxes with three different size (128*128, 256*256, 512*512), and three different height-width ratio (1:1, 1:2, 2:1).

The RPN can be trained end-to-end by backpropagation and SGD: First, select an image randomly, generate its feature map with an extent of 40*50*512 and convolute the featuremap with 3*3 window (stride=1, padding=2, cernel=512, size=3*3*512, extracted feature=1*1*512). Second. Select 256 anchor boxes in features map to form mini-batch (positive and negative samples compensate), every anchor box refers to one position in feature map, corresponding a feature of 1*1*512 mentioned above. Third, input mini-batch into classification network (1*1*2*256) and output 256 vectors 1*2. Meanwhile, input mini-batch into regression network: Regression network has 9 cernels which learn the regression parameters from different sized anchor boxes. The outoput will be 256 vectors of 1*4, meaning the shift from prediction bouding box to anchor box. Calculate the mutli-task loss for every anchor box based on prediction and true labels and make bacpropagation based on loss function.

The advantages of RPN are as follows. First, multi-scale anchor box can be an optimization of object detection algorithm. Second, R-CNN and Fast R-CNN use uniform bounding box regression, whereas RPN use different bounding box regression for different anchor box. Third, classification network will filter the box with low objectness.

Training process of Faster R-CNN is as follows. First, train RPN network and fine tune pretrained network meanwhile. Second, train Fast R-CNN by proposals generated by RPN and fine tune pretrained network again. Third, use the fine-tuning network to initialize the RPN network, but only renew the parameter for RPN layers. Fourth, train the Fast R-CNN by proposals generated by RPN. Fifth, repeat the steps mentioned above in other mini-batch.

Mask R-CNN

The method, called Mask R-CNN, extends Faster R-CNN by adding a branch for predicting an object mask in parrallel with the existing branch for bounding box recognization. It is simple to train, easy to generalize to other tasks and to develop a comparably enabling framework for instance segmentation, which requires detection and segmentation. So Mask R-CNN extends Faster R-CNN by adding predicting segmentation masks on each RoI, which is segmented by FCN. Mask R-CNN introduces RoI Align to replace RoI Pooling in Faster R-CNN, because RoI Pooling is not a pixel-to-pixel alignmnet, which has an impact on the accuracy of mask. They also think it is essential to decouple mask and class predicition, relying on the network's RoI classification branch to predict the category.

There are some related work such as FCIS. FCIS uses FCN and predicts object classes, boxes and masks simultaneously. But it exhibits systematic errors on overlapping instances and is challenged by segementing instances. (Why?)

The structure of Mask R-CNN is as follows. Like wise Faster R-CNN, the Mask R-CNN initialize the RPN by pretrained network, then do classification and location on each RoI to find binary mask, whereas the other network would find mask first.

And the Loss Function of Mask R-CNN is a multi-task loss of classification loss, bounding box loss and average binary cross-entropy loss. The last loss allows the network to generate masks for every class without competition among classes.

Mask representation is as follows. They predict an m*m mask from each RoI using an FCN, which allows each layer in the mask branch to maintain the explicit m*m object spatial layout without collapsing it into a vector representation that lacks spatial dimensions. This requires fewer parameters and more accurate.

RoIAlign solves the problem of pixel-to-pixel alignment between input and output of RoI pooling. Dashed grid represents a feature map, the solid lines an RoI, and the dots the 4 sampling points in each bin. RoIAlign computes the value of eacj sampling point by bilinear interpolation from the nearby grid points on the feature map.

They instantiate Mask R-CNN with multiple architectures. They differentiate between: 1. the convolutional backbone architecture (ResNet-50, ResNet-101, ResNeXt-50, ResNeXt-101), 2. the network head for bounding-box recognition. (Faster R-CNN with ResNet, and proffoer feature from block 4), and 3. a more effective architecture called Feature Pyramid Network.

There are some implementation details. The hyperparameters used in Fast R-CNN and Faster R-CNN are valid for Mask R-CNN. First, the IoU is only valid when it is bigger than 0.5. Second, (image-centric training, resize short edge to 800, mini-batch=2, N RoIs, ResNet-50-C4 backbone -> N=64, FPN backbone -> N=512). Third, PRN has 5 scales, 3 ratios. Fourth, (ResNet-50-C4 backbone -> proposal number=300, FPN backbone -> proposal number=1000).

Main results are as follows. 1. FCIS has systematic artifacts, Mask R-CNN not. 2. Ablation Experiments (Architecture better -> result better, Independent neccesary, RoIAlign better, FCN better than MLP)