Faster R-CNN (NIPS 2015)

Ren et al.

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

① Problem Description:

Faster R-CNN: region proposal computation.

SPPnet (slide window),

R-CNN (selective search).

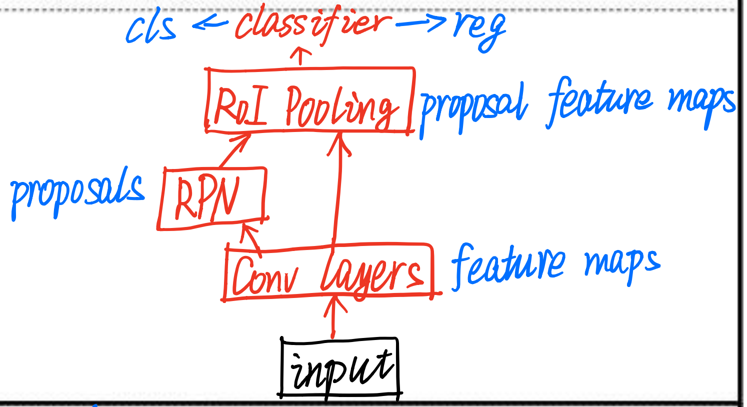
② Problem Solution:

RPN (Region Proposal Network):

it shares full-image convolutional features with the detection network.

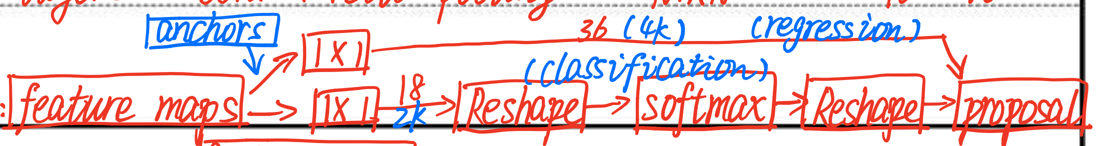
"attention" mechanisms:
RPN component tells the unified network where to look.

③ Conceptual Understanding

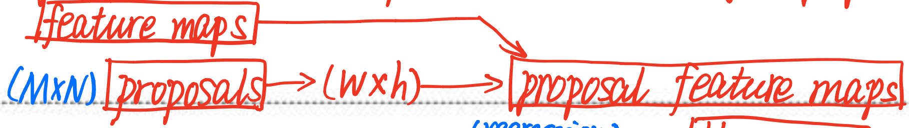


① Conv layers : conv + relu + pooling. (size=3, pad=1, stride=1) (2,1,1) (2,0,1) $M \times N \xrightarrow{VGG16} \frac{M}{16} \times \frac{N}{16}$.

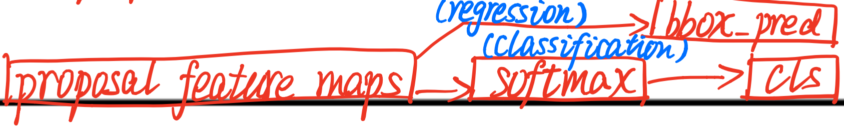
② RPN :



③ RoI pooling :



④ Classifition :

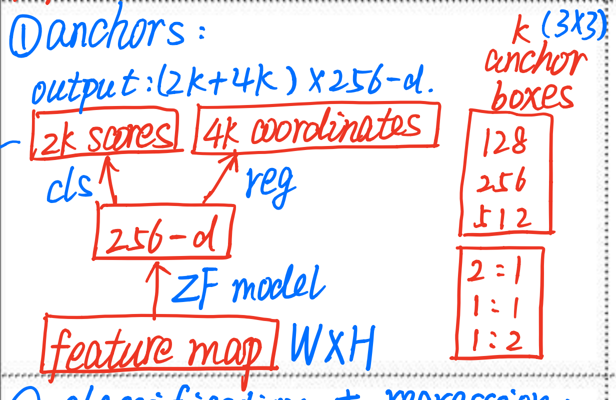


Details of implementation

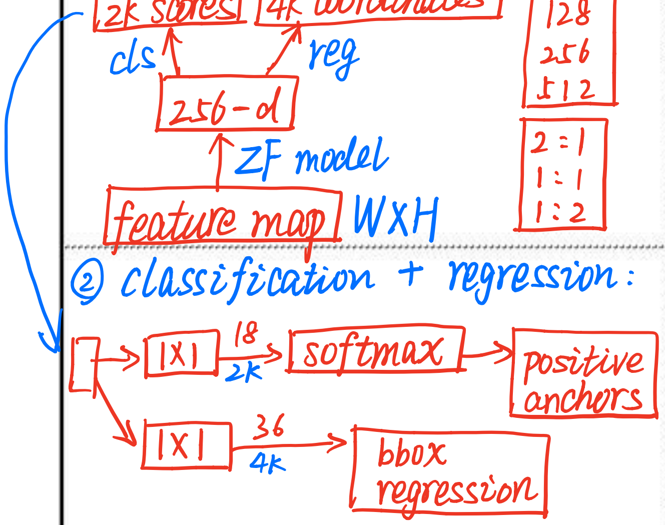
RPN:

① anchors:

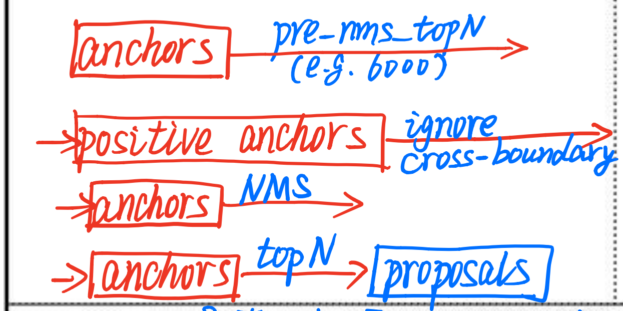
output: $(2k+4k)\times 256$-d.



② classification + regression:



③ proposal layers:



Experiments:

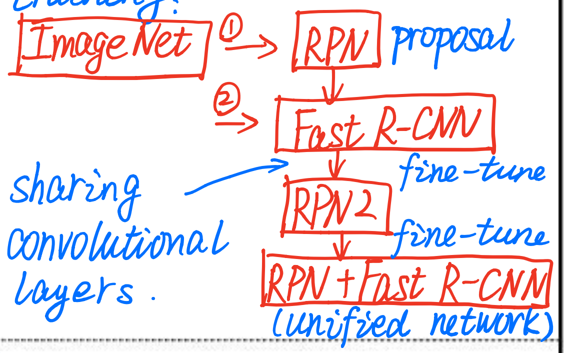
① loss function:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

e.g. $N_{cls}=256$, $N_{reg}=2400$, $\lambda=\frac{N_{reg}}{N_{cls}}\approx 10$.

$$L_{reg}(t_i, t_i^*) = \sum_{i\in\{x,y,w,h\}} \text{smooth}_{L_1}(t_i - t_i^*).$$

② training:



③ results:

VOC

COCO

COCO + VOC

Code: (PyTorch Impletation)

① modified files to suit own machine.

② change custom datasets to train.