

Faster R-CNN (NIPS 2015)

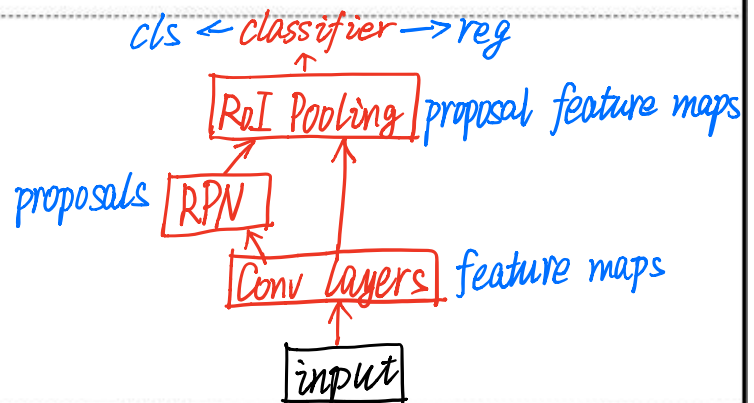
Faster R-CNN: Towards Real-Time Object Detection with Region

Ren et al.

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



- (size=3, pad=1, stride=1) $\begin{matrix} (2,1,1) \\ (2,0,1) \end{matrix}$
- ① Conv layers: conv + relu + pooling. $M \times N \xrightarrow{\text{VGG16}} \frac{M}{16} \times \frac{N}{16}$.
- ② RPN: feature maps $\xrightarrow{\text{anchors}} [1 \times 1] \xrightarrow{36(4k)} [1 \times 1] \xrightarrow{2k} \text{Reshape} \xrightarrow{(classification)} \text{softmax} \xrightarrow{(regression)} \text{Reshape} \xrightarrow{} \text{proposal}$
- ③ RoI pooling: $(M \times N)$ [proposals] $\rightarrow (w \times h) \rightarrow$ [proposal feature maps]
- ④ Classification: [proposal feature maps] $\xrightarrow{(regression)} \text{softmax} \xrightarrow{(classification)} \text{bbox_pred} \rightarrow$ [cls]

Details of implementation

RPN:

① anchors:

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

2k scores 4k coordinates

cls

reg

256-d

↑ ZF model

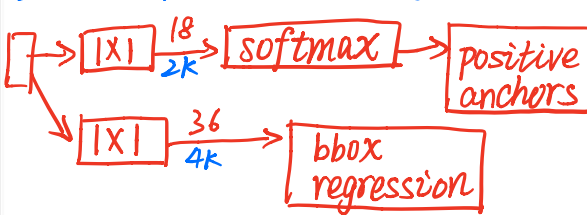
feature map $W \times H$

k (3x3)
anchor
boxes

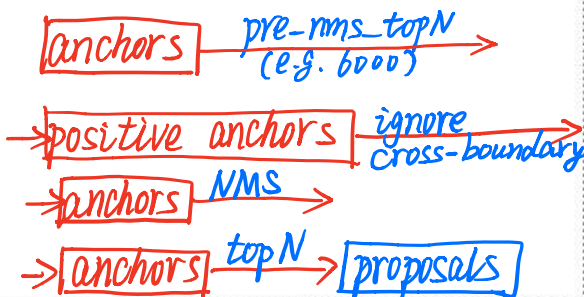
128
256
512

2:1
1:1
1:2

② classification + regression:



③ proposal layers:



Experiments:

① loss function:

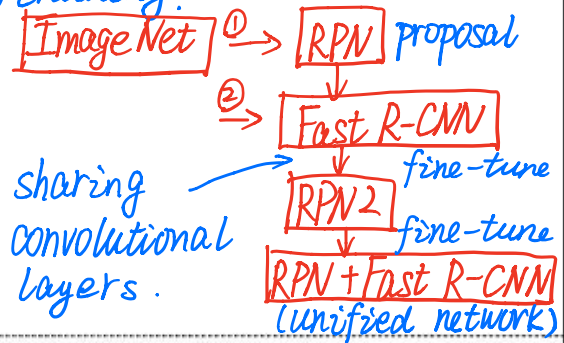
$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*)$$

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

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

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

② training:



sharing
convolutional
layers.

③ results:

VOC
COCO
COCO + VOC

Code: (PyTorch Implementation)

- ① modified files to suit own machine.
- ② change custom datasets to train.