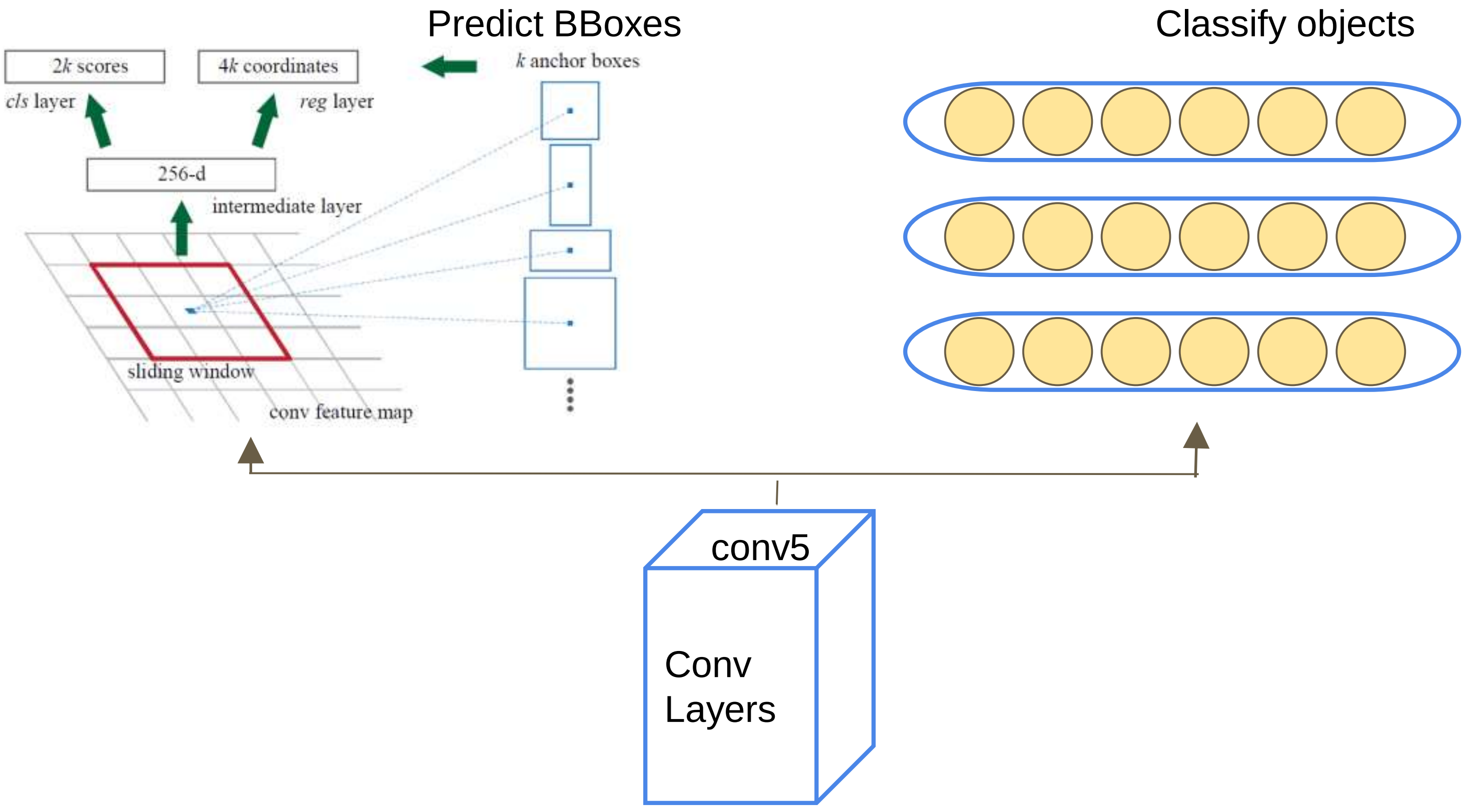
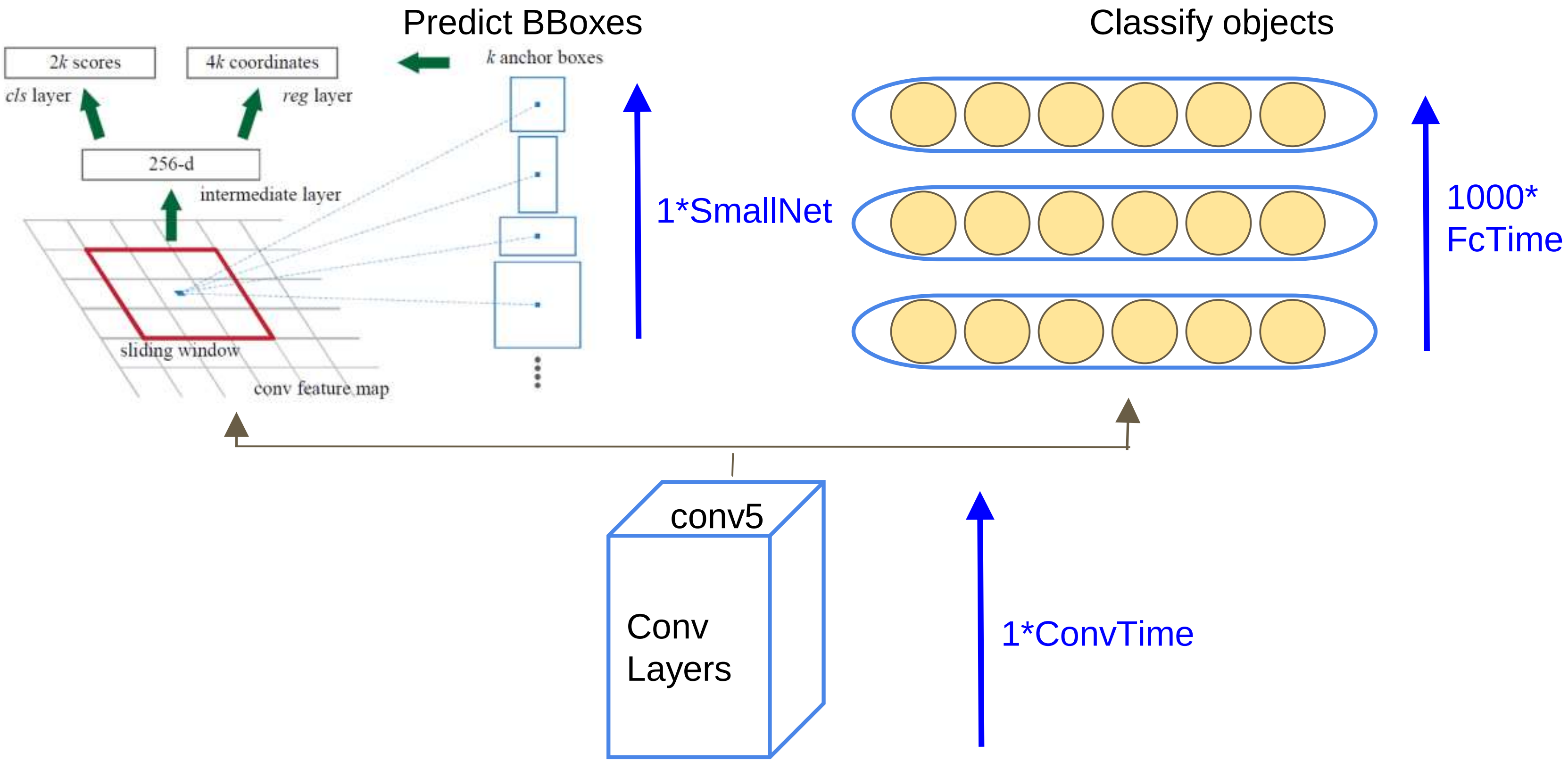
Faster R-CNN: Towards Real-Time Object detection

Microsoft Research, NIPS2015
Presentor: Andy Tsai

Key Idea: Region Proposal Net (RPN) layer



Testing Time: faster R-CNN



Predict BBoxes

Classify objects

1*SmallNet

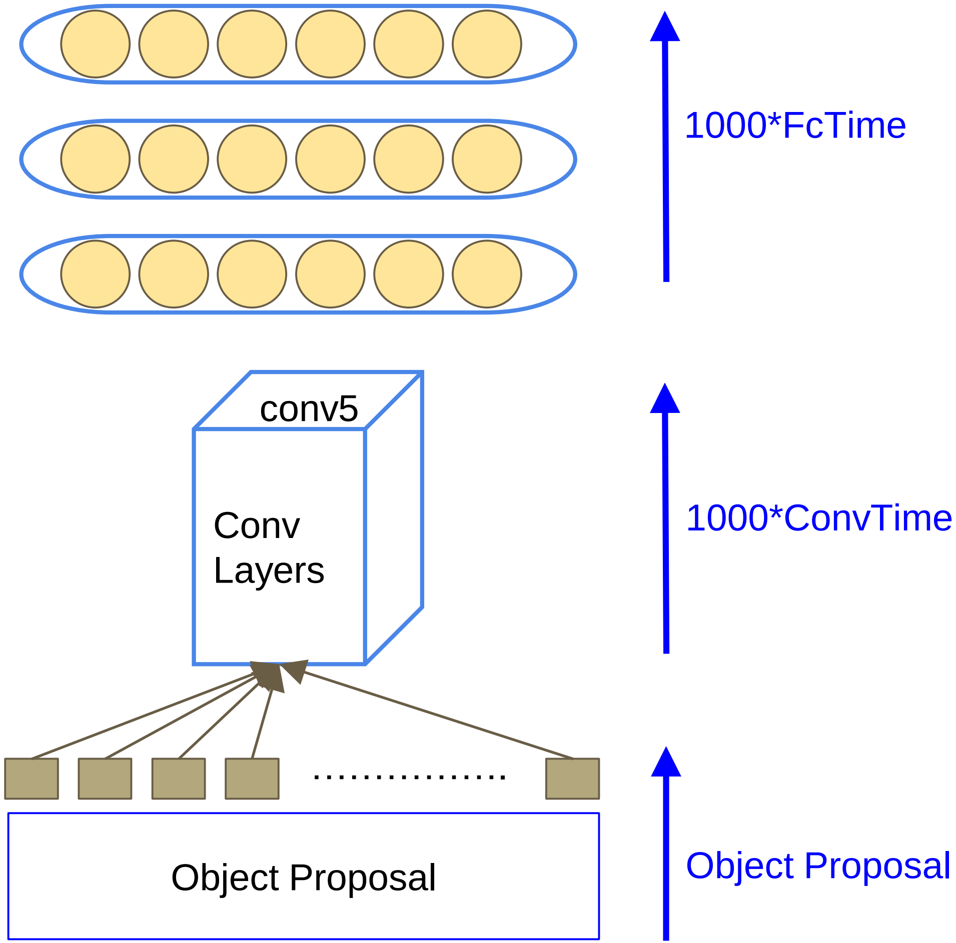
1000*
FcTime

conv5

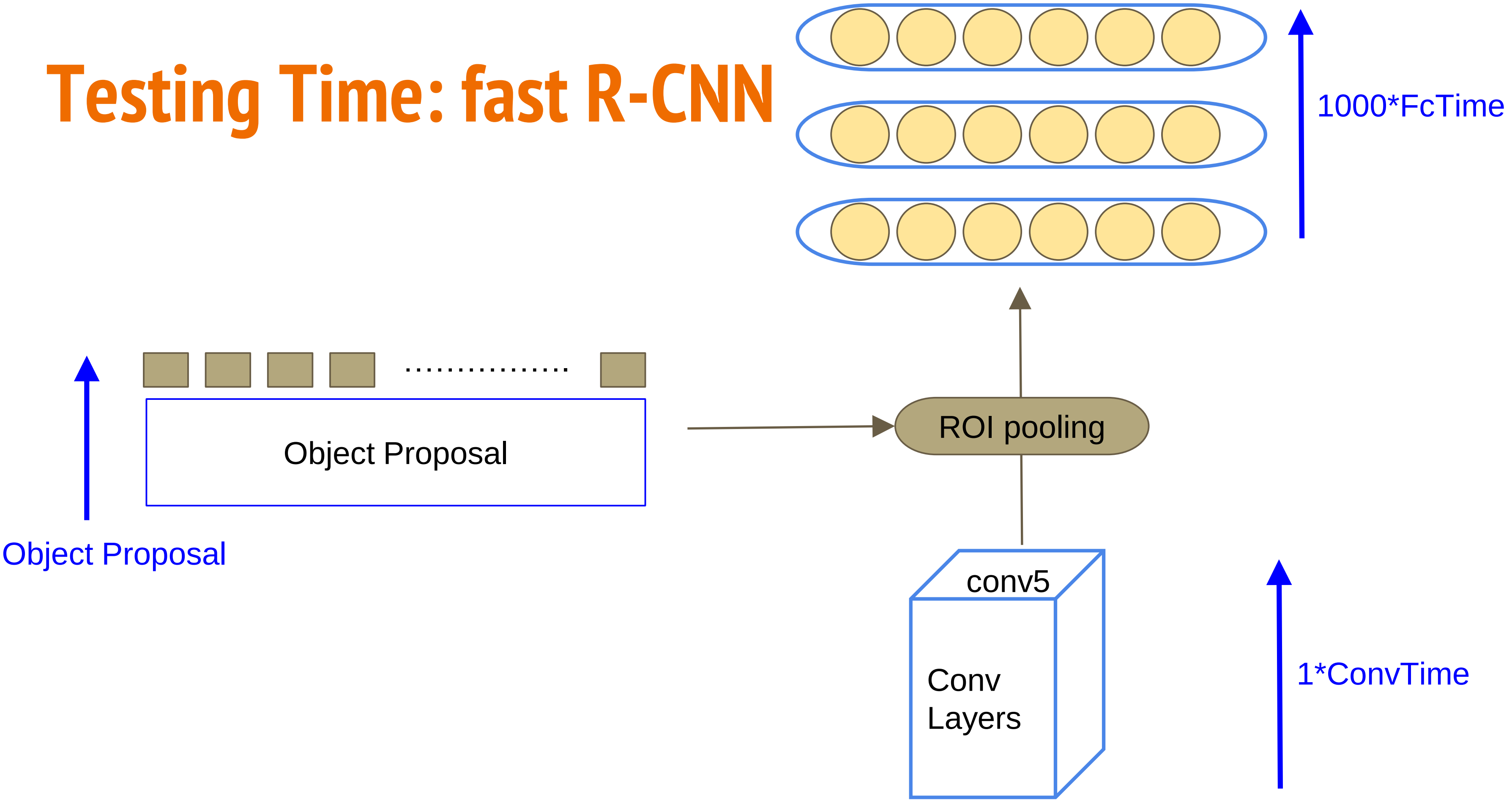
Conv
Layers

1*ConvTime

Testing Time: R-CNN



Testing Time: fast R-CNN



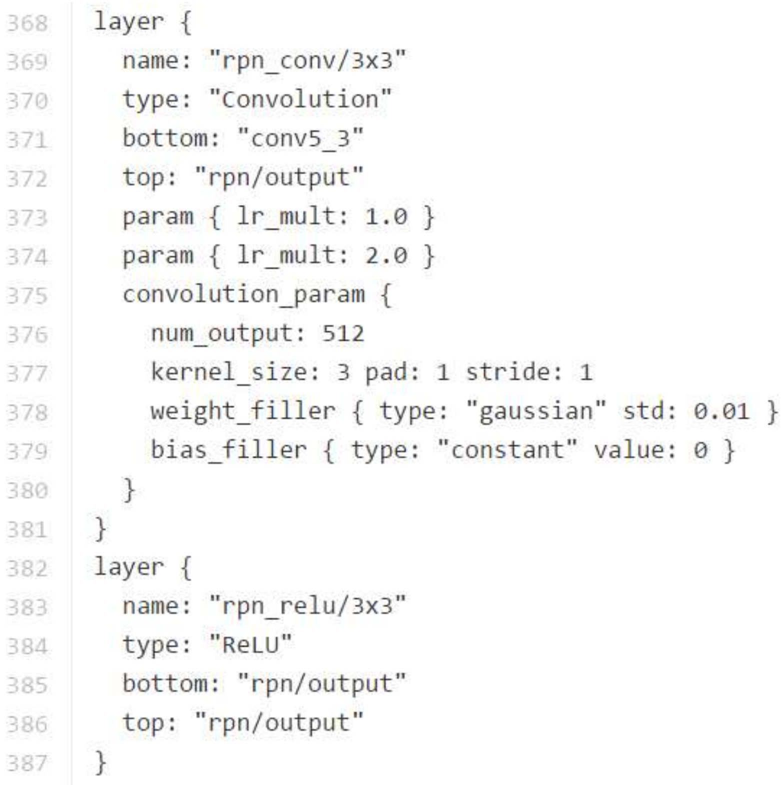
Fast, Accurate Object Detection

- fastest region proposal method: Edge Boxes [4fps, 1000 proposal]

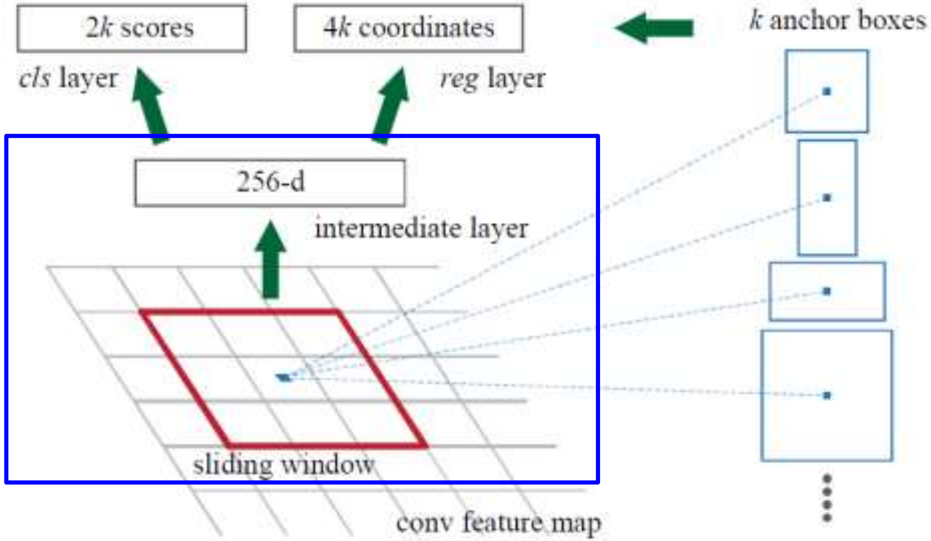
- Testing stage

Model	Time
Edge boxes + R-CNN	0.25 sec + 1000*ConvTime + 1000*FcTime
Edge boxes + fast R-CNN	0.25 sec + 1*ConvTime + 1000*FcTime
faster R-CNN	1*ConvTime + 1000*FcTime

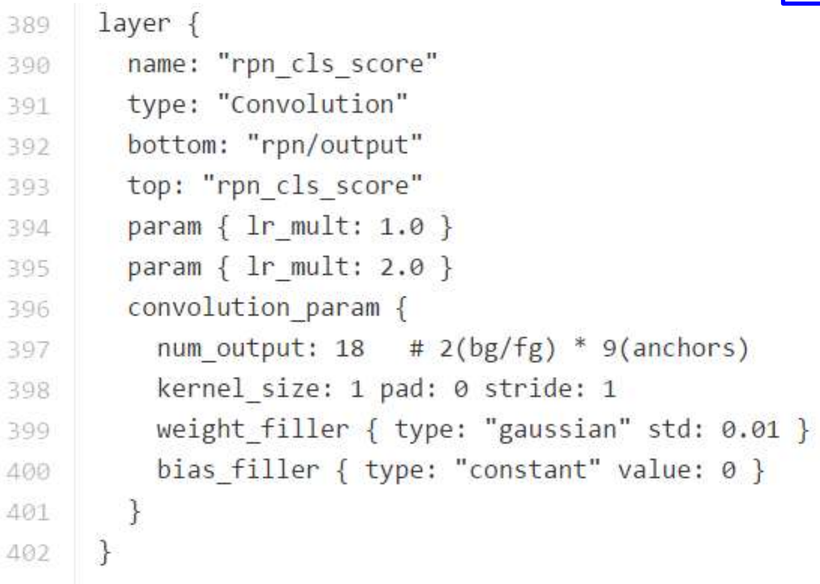
RPN layer



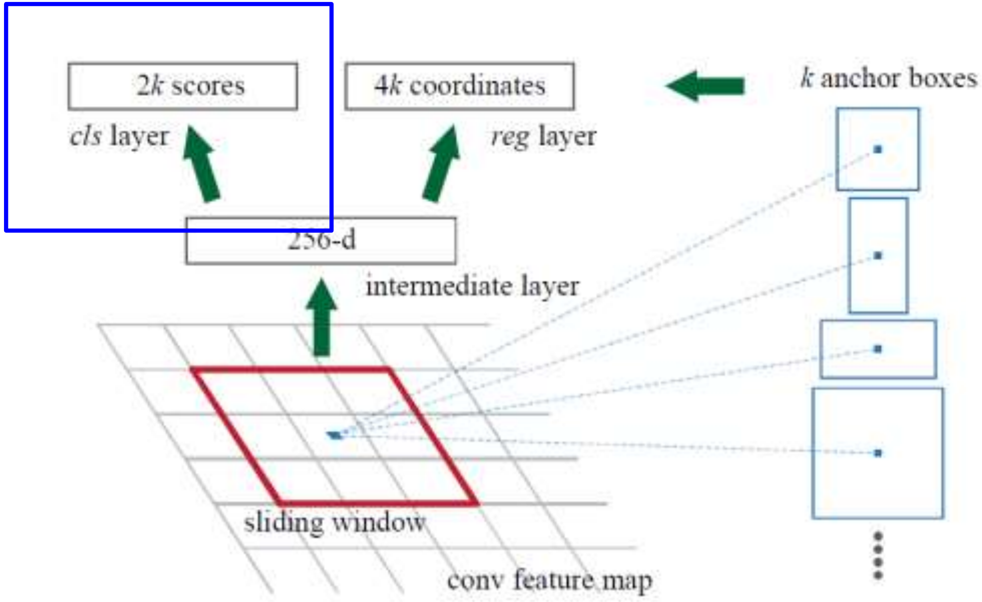
```
layer {
  name: "rpn_conv/3x3"
  type: "Convolution"
  bottom: "conv5_3"
  top: "rpn/output"
  param { lr_mult: 1.0 }
  param { lr_mult: 2.0 }
  convolution_param {
    num_output: 512
    kernel_size: 3 pad: 1 stride: 1
    weight_filler { type: "gaussian" std: 0.01 }
    bias_filler { type: "constant" value: 0 }
  }
}
layer {
  name: "rpn_relu/3x3"
  type: "ReLU"
  bottom: "rpn/output"
  top: "rpn/output"
}
```



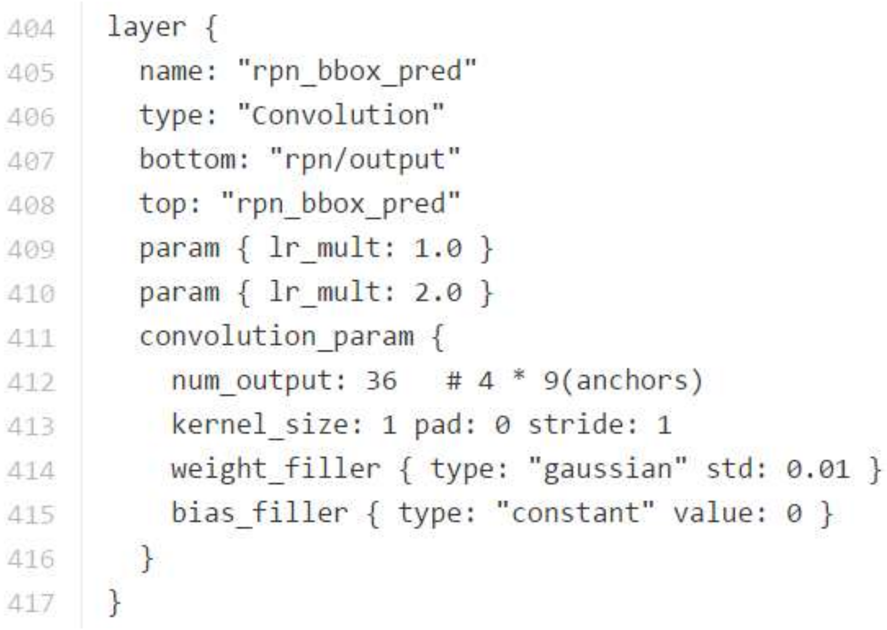
RPN layer



```
layer {
  name: "rpn_cls_score"
  type: "Convolution"
  bottom: "rpn/output"
  top: "rpn_cls_score"
  param { lr_mult: 1.0 }
  param { lr_mult: 2.0 }
  convolution_param {
    num_output: 18   # 2(bg/fg) * 9(anchors)
    kernel_size: 1 pad: 0 stride: 1
    weight_filler { type: "gaussian" std: 0.01 }
    bias_filler { type: "constant" value: 0 }
  }
}
```

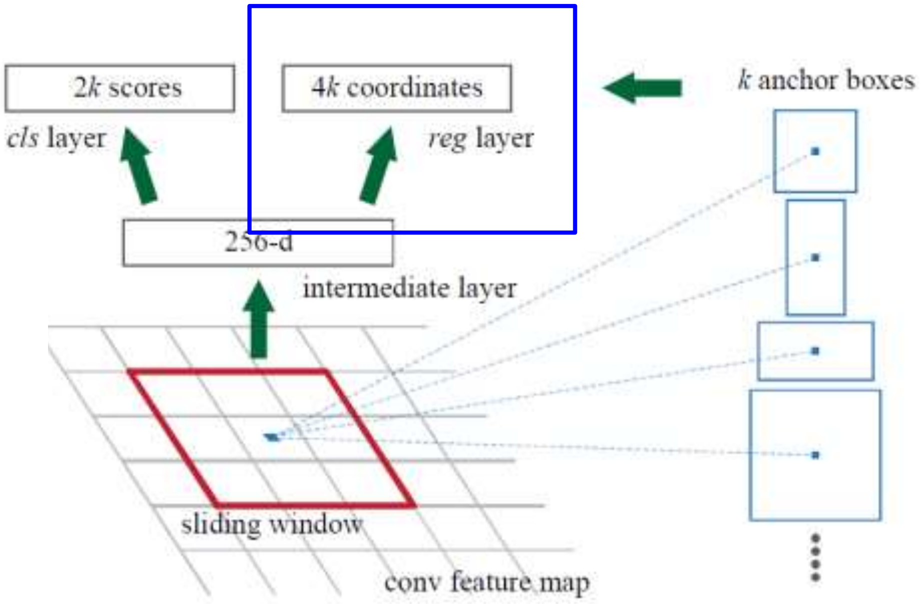


RPN layer



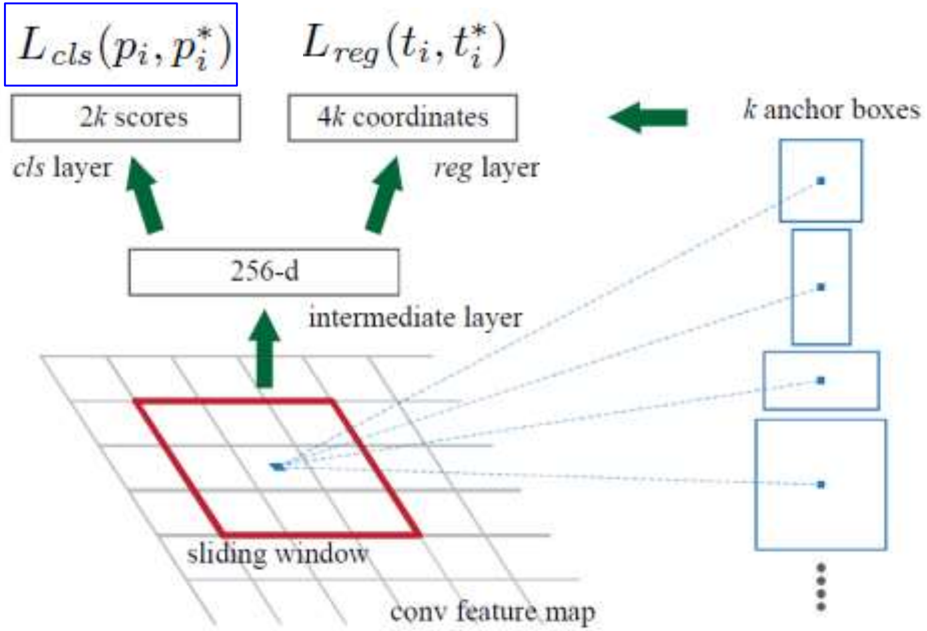
```
layer {
  name: "rpn_bbox_pred"
  type: "Convolution"
  bottom: "rpn/output"
  top: "rpn_bbox_pred"
  param { lr_mult: 1.0 }
  param { lr_mult: 2.0 }
  convolution_param {
    num_output: 36   # 4 * 9(anchors)
    kernel_size: 1 pad: 0 stride: 1
    weight_filler { type: "gaussian" std: 0.01 }
    bias_filler { type: "constant" value: 0 }
  }
}
```

Anchor regression



RPN: Loss Function

- 2 class Softmax cross entropy loss
- Discriminative training:
    - pi* = 1 if IoU > 0.7
    - pi* = 0 if IoU < 0.3
    - otherwise, do not contribute to loss



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

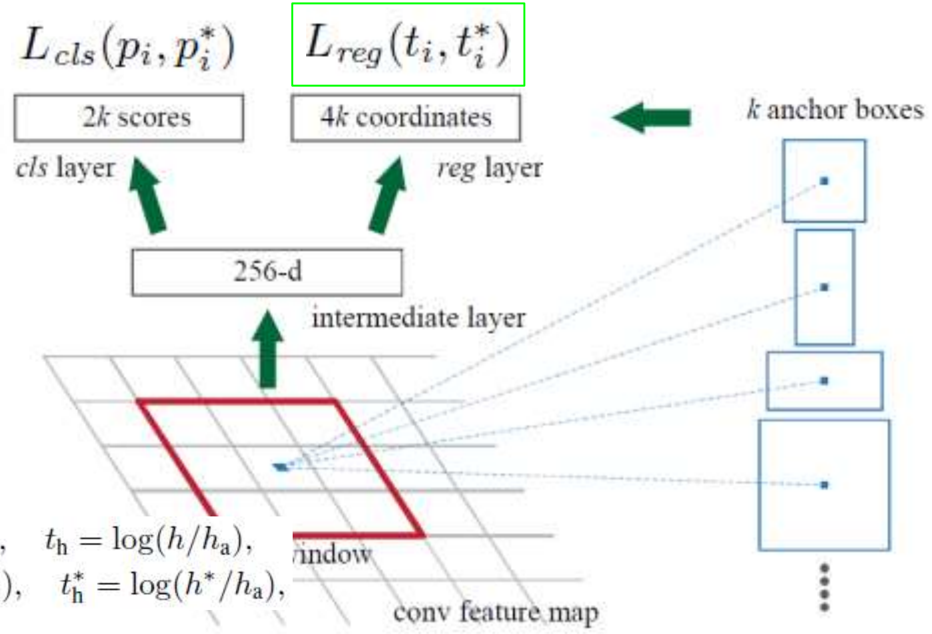
RPN: Loss Function

$$L_{\mathrm{loc}}(t, t^*) = \sum_{i \in \{\mathrm{x,y,w,h}\}} \mathrm{smooth}_{L_1}(t_i, t_i^*), \qquad (2)$$

in which

$$\mathrm{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases} \qquad (3)$$

$$t_{\mathrm{x}} = (x - x_{\mathrm{a}})/w_{\mathrm{a}}, \quad t_{\mathrm{y}} = (y - y_{\mathrm{a}})/h_{\mathrm{a}}, \quad t_{\mathrm{w}} = \log(w/w_{\mathrm{a}}), \quad t_{\mathrm{h}} = \log(h/h_{\mathrm{a}}),$$
$$t_{\mathrm{x}}^* = (x^* - x_{\mathrm{a}})/w_{\mathrm{a}}, \quad t_{\mathrm{y}}^* = (y^* - y_{\mathrm{a}})/h_{\mathrm{a}}, \quad t_{\mathrm{w}}^* = \log(w^*/w_{\mathrm{a}}), \quad t_{\mathrm{h}}^* = \log(h^*/h_{\mathrm{a}}),$$



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

only positive sample contribute to reg. Loss

How to train faster R-CNN ?

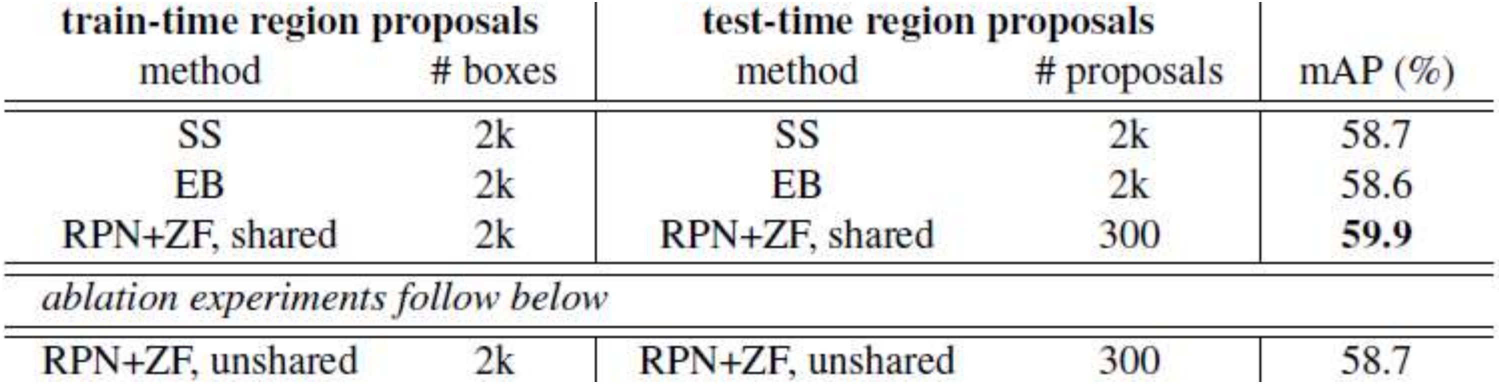
- balance sampling (neg. vs pos. = 1:1)
- joint training is almost impossible (conv update alternating)
- 4 step training !

How to train faster R-CNN ?

1. train RPN with ImageNet pre-trained model
2. train fast R-CNN using proposal generated by 1. [no params. sharing]
3. use conv trained by 2. to initialize model, fix shared conv, update RPN
4. fix shared conv, fine-tune fc

Result - MAP

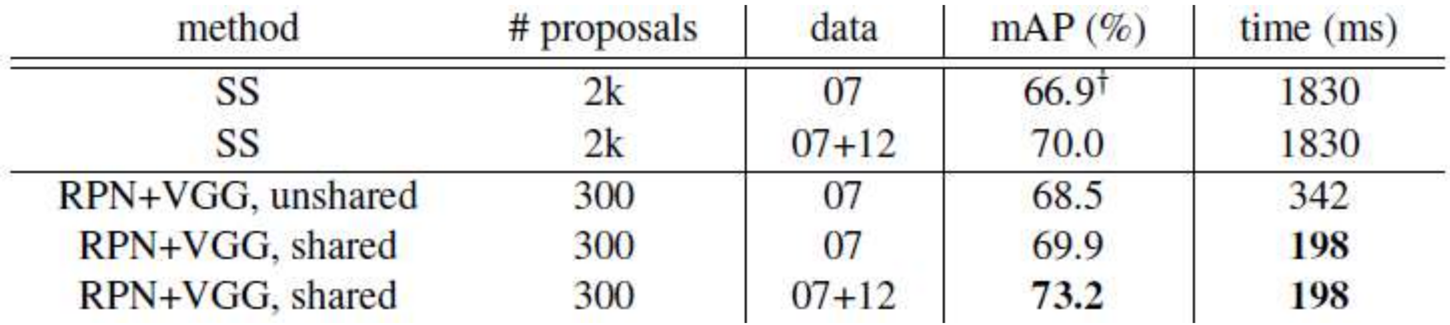
- VOC2007, ZF ConvNet



train-time region proposals		**test-time region proposals**		
method	# boxes	method	# proposals	mAP (%)
SS	2k	SS	2k	58.7
EB	2k	EB	2k	58.6
RPN+ZF, shared	2k	RPN+ZF, shared	300	**59.9**
ablation experiments follow below				
RPN+ZF, unshared	2k	RPN+ZF, unshared	300	58.7

Result - MAP

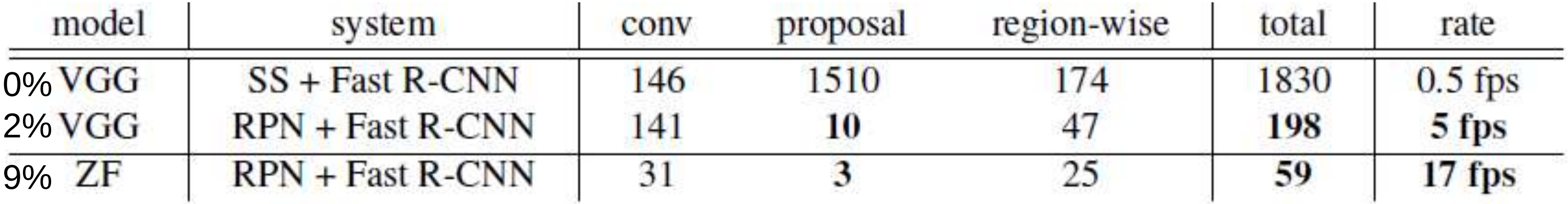
Using VGG ConvNet, fast R-CNN vs faster R-CNN



method	# proposals	data	mAP (%)	time (ms)
SS	2k	07	66.9[†]	1830
SS	2k	07+12	70.0	1830
RPN+VGG, unshared	300	07	68.5	342
RPN+VGG, shared	300	07	69.9	**198**
RPN+VGG, shared	300	07+12	**73.2**	**198**

Result - Time

VOC 2007, fast R-CNN vs faster R-CNN



	model	system	conv	proposal	region-wise	total	rate
70.0%	VGG	SS + Fast R-CNN	146	1510	174	1830	0.5 fps
73.2%	VGG	RPN + Fast R-CNN	141	**10**	47	**198**	**5 fps**
59.9%	ZF	RPN + Fast R-CNN	31	**3**	25	**59**	**17 fps**

Official Leader Board Score

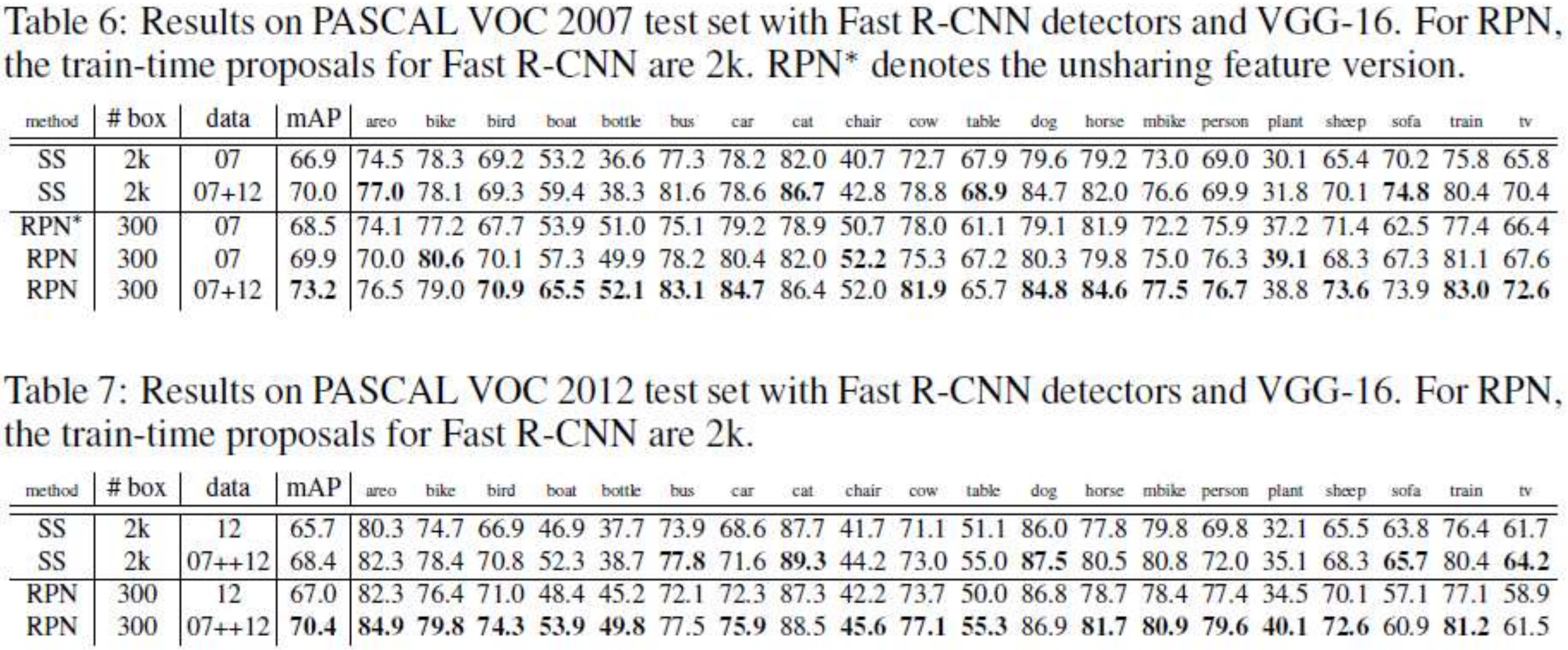


Table 6: Results on PASCAL VOC 2007 test set with Fast R-CNN detectors and VGG-16. For RPN, the train-time proposals for Fast R-CNN are 2k. RPN* denotes the unsharing feature version.

method	# box	data	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
SS	2k	07	66.9	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8
SS	2k	07+12	70.0	**77.0**	78.1	69.3	59.4	38.3	81.6	78.6	**86.7**	42.8	78.8	**68.9**	84.7	82.0	76.6	69.9	31.8	70.1	**74.8**	80.4	70.4
RPN*	300	07	68.5	74.1	77.2	67.7	53.9	51.0	75.1	79.2	78.9	50.7	78.0	61.1	79.1	81.9	72.2	75.9	37.2	71.4	62.5	77.4	66.4
RPN	300	07	69.9	70.0	**80.6**	70.1	57.3	49.9	78.2	80.4	82.0	**52.2**	75.3	67.2	80.3	79.8	75.0	76.3	**39.1**	68.3	67.3	81.1	67.6
RPN	300	07+12	**73.2**	76.5	79.0	**70.9**	**65.5**	**52.1**	**83.1**	**84.7**	86.4	52.0	**81.9**	65.7	**84.8**	**84.6**	**77.5**	**76.7**	38.8	**73.6**	73.9	**83.0**	**72.6**

Table 7: Results on PASCAL VOC 2012 test set with Fast R-CNN detectors and VGG-16. For RPN, the train-time proposals for Fast R-CNN are 2k.

method	# box	data	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
SS	2k	12	65.7	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7
SS	2k	07++12	68.4	82.3	78.4	70.8	52.3	38.7	**77.8**	71.6	**89.3**	44.2	73.0	55.0	**87.5**	80.5	80.8	72.0	35.1	68.3	**65.7**	80.4	**64.2**
RPN	300	12	67.0	82.3	76.4	71.0	48.4	45.2	72.1	72.3	87.3	42.2	73.7	50.0	86.8	78.7	78.4	77.4	34.5	70.1	57.1	77.1	58.9
RPN	300	07++12	**70.4**	**84.9**	**79.8**	**74.3**	**53.9**	**49.8**	77.5	**75.9**	88.5	**45.6**	**77.1**	**55.3**	86.9	**81.7**	**80.9**	**79.6**	**40.1**	**72.6**	60.9	**81.2**	61.5

Code available at gitHub

Matlab(faster-rcnn) & python(py-faster-rcnn) version

Thank you :)