Object Detection 2

INRIA Sophia Antipolis

People Detection in real world situations

People Tracking in real world situations

Outline

- Object Detection : reminder.
- Faster-RCNN
  - Feature probing vs pooling
  - Loss functions
- SSD
- Feature Pyramid Network
- FCOS

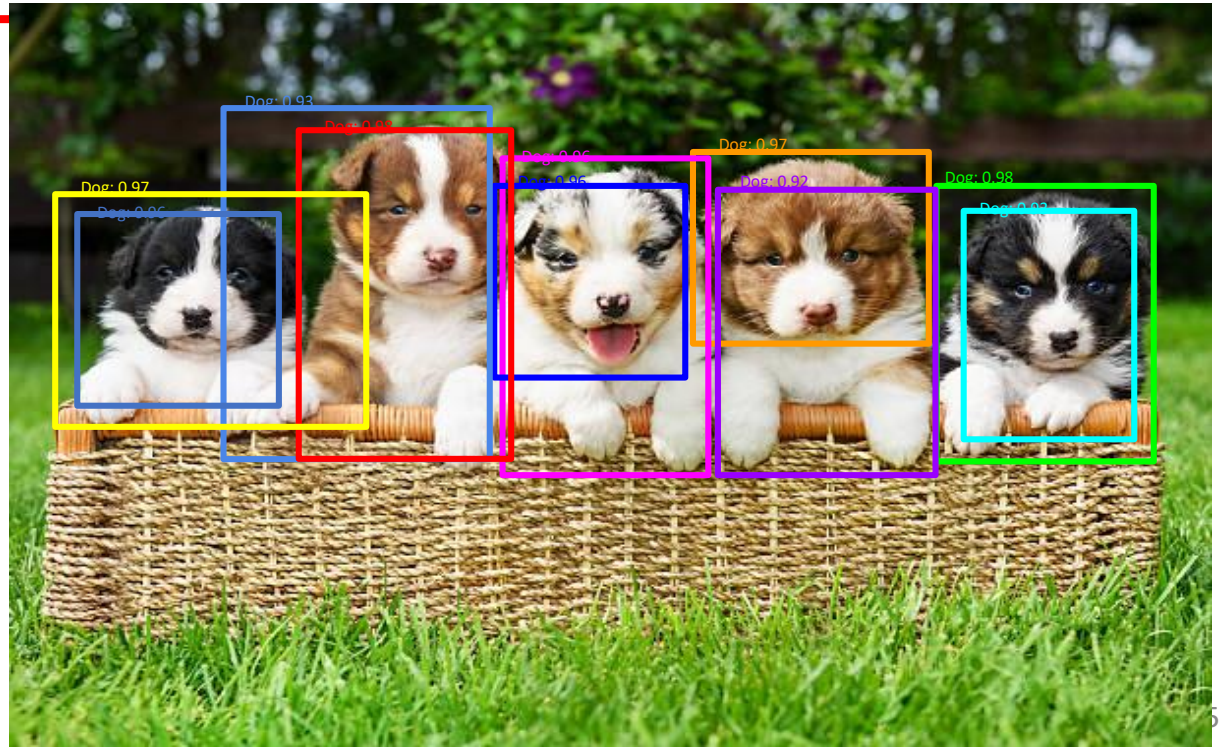
Performance evaluation: Average Precision

Conf.	Rank	Correct IoU?	Precision	Recall
0.98	1	True	1.0	0.2
0.98	2	True	1.0	0.4
0.972	3	False	0.67	0.4
0.97	4	False	0.5	0.4
0.96	5	False	0.4	0.4
	6	True	0.5	0.6
	7	True	0.57	0.8
	8	False	0.5	0.8
	9	False	0.44	0.8
	10	True	0.5	1.0

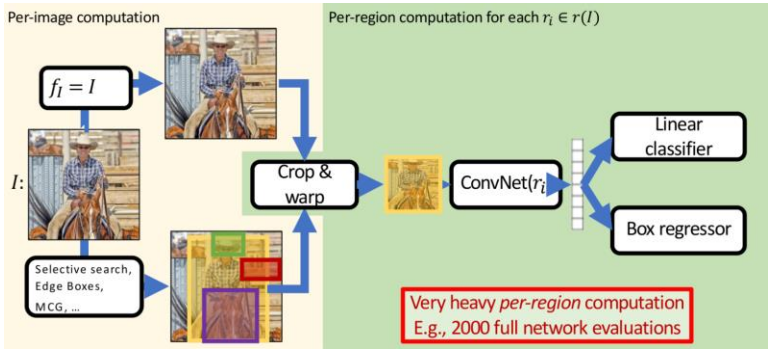
0.97

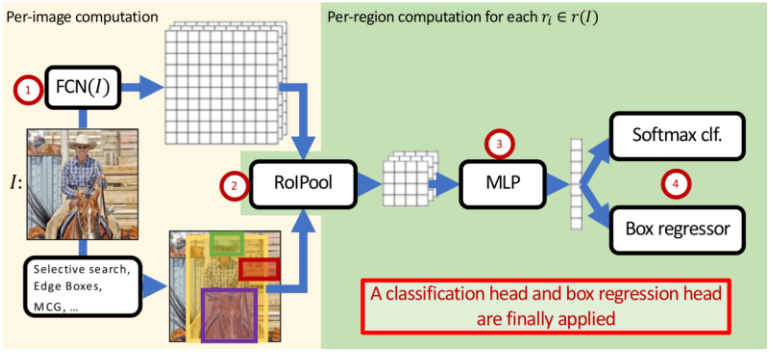
$$\text{Precision} = \frac{\text{TP}}{\text{TP + FP}} = \frac{2}{3} = 0.67$$

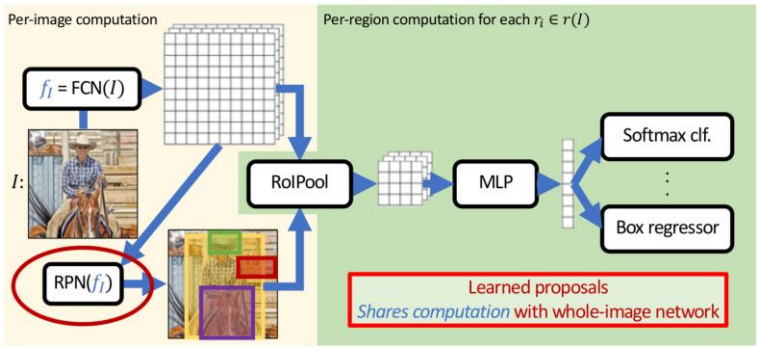
$$\text{Recall} = \frac{\text{TP}}{\text{TP + FN}} = \frac{2}{5} = 0.4$$



Summary of previous class







"Slow" R-CNN:

Run CNN independently for each region

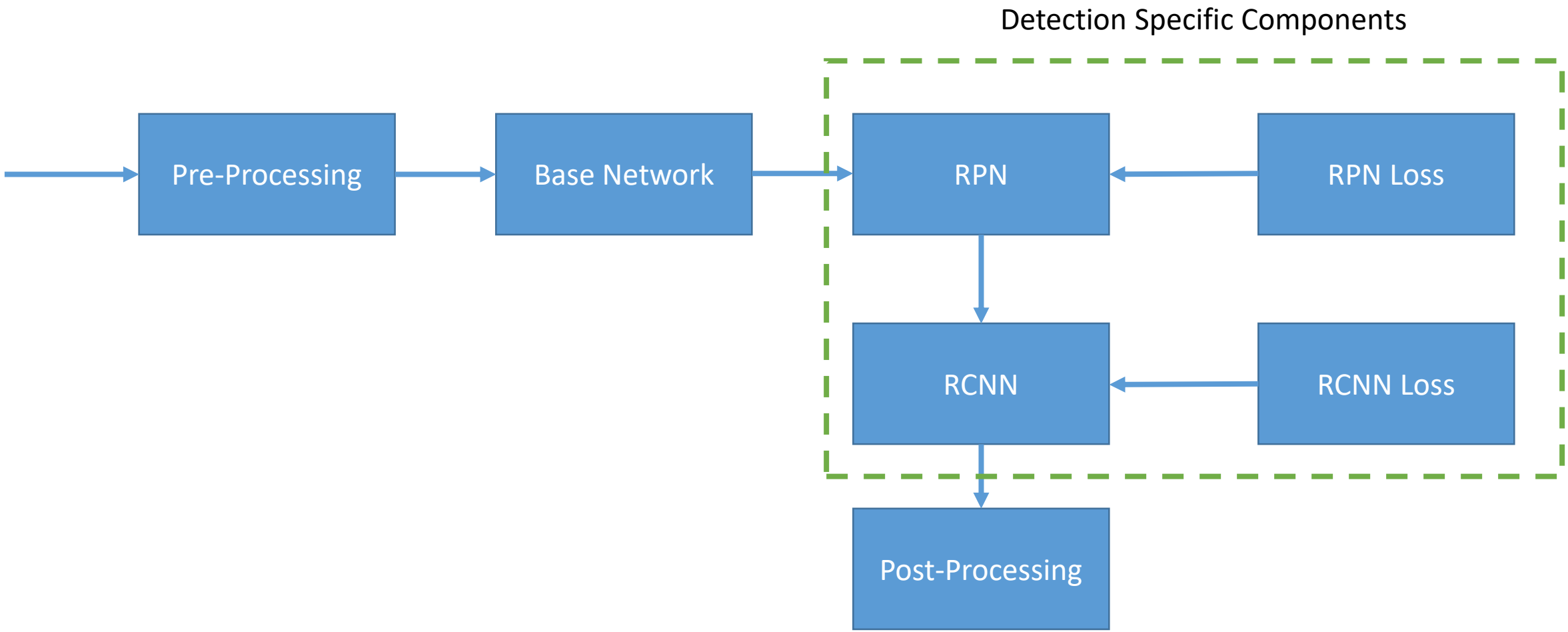
Fast R-CNN:

Apply differentiable cropping to shared image features

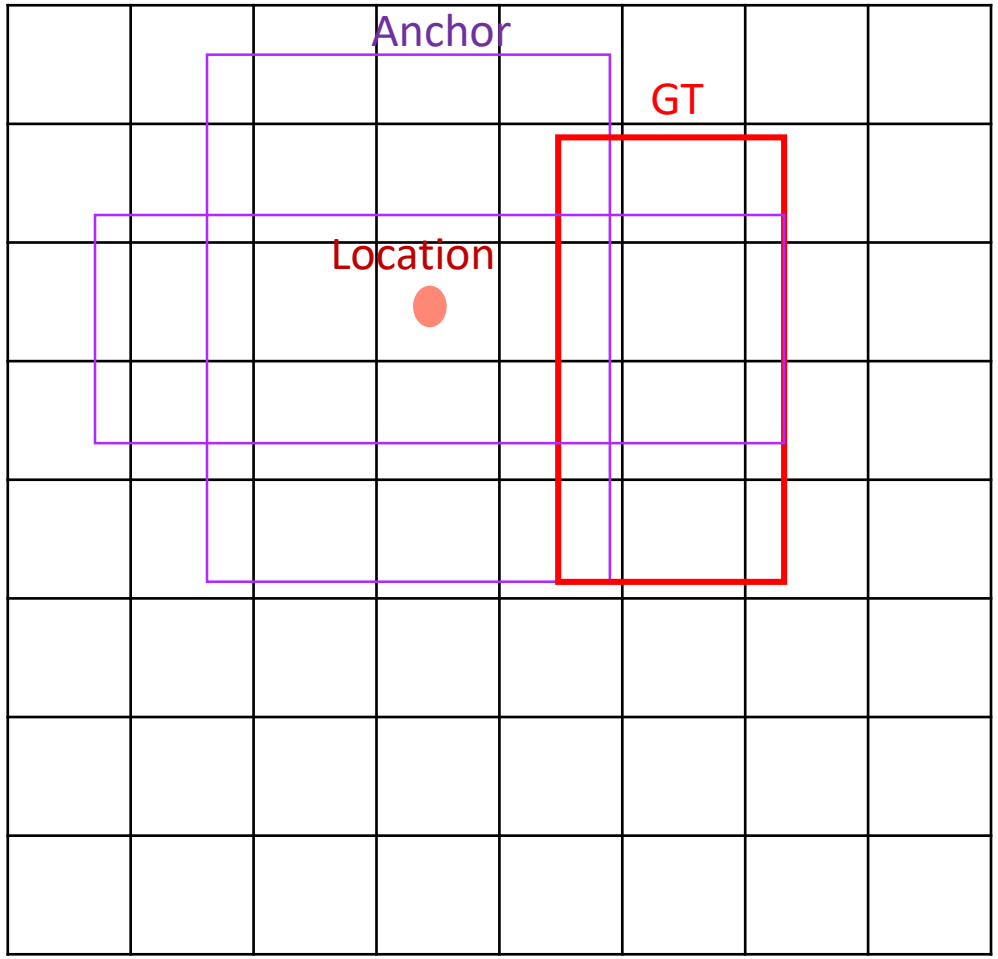
Faster R-CNN:

Compute proposals with CNN

Faster-RCNN


Detection Specific Components
Pre-Processing
Base Network
RPN
RPN Loss
RCNN
RCNN Loss
Post-Processing

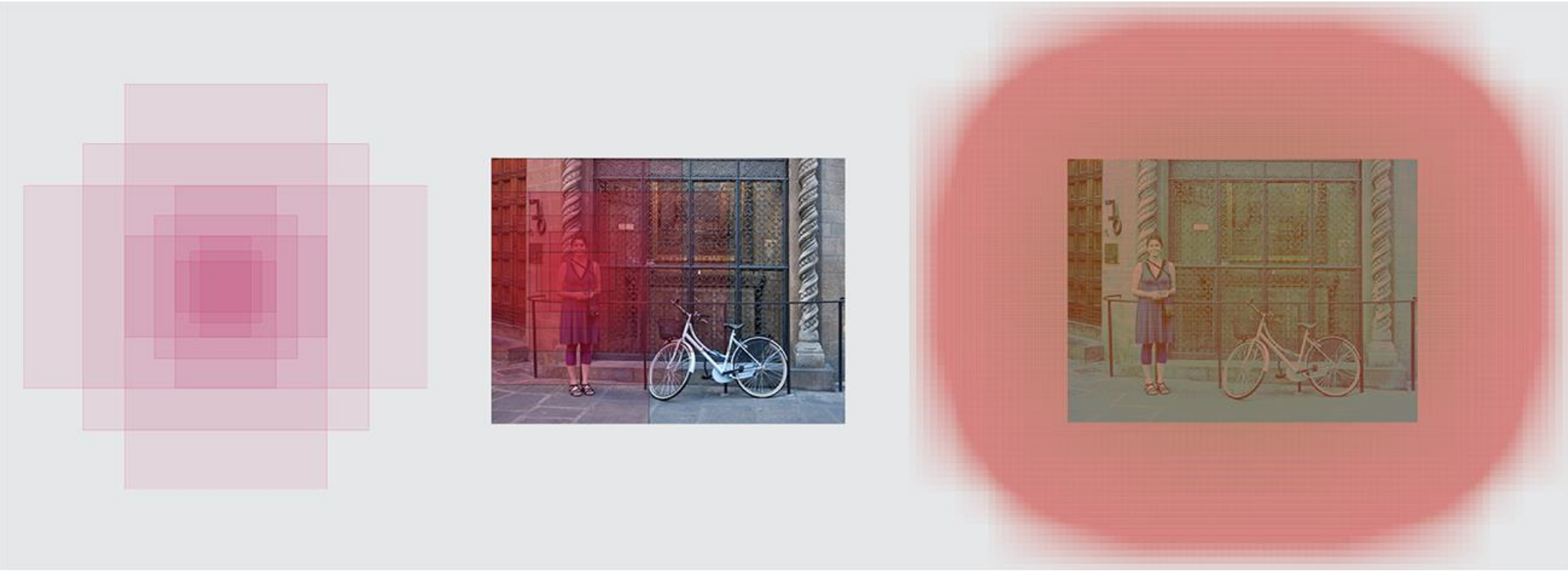
Region Proposal Network:



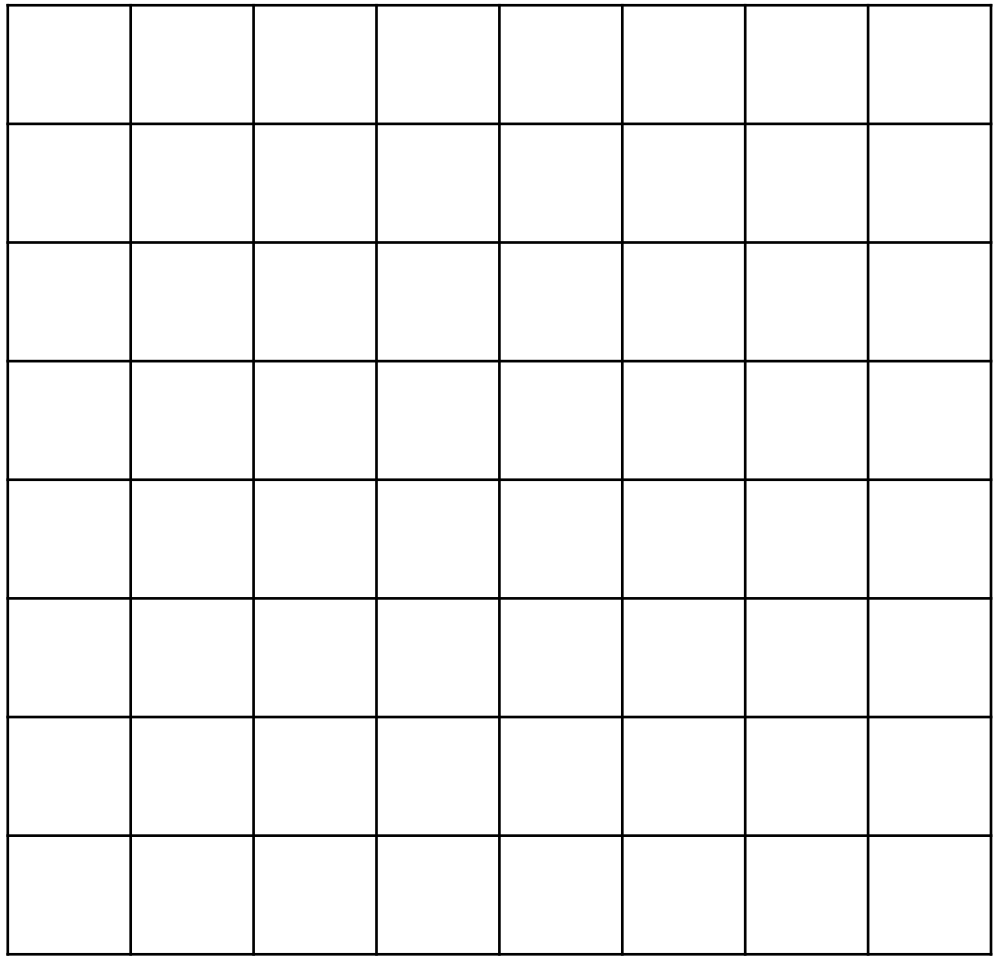
- Slide every anchor over the feature map and measure the intersection-over-union with every GT box.
- For IoU>UT we call it a positive anchor.
- For IoU<LT, we call it a negative anchor.
- For LT<IoU<UT, we simply ignore the anchor i.e don’t do any computations.

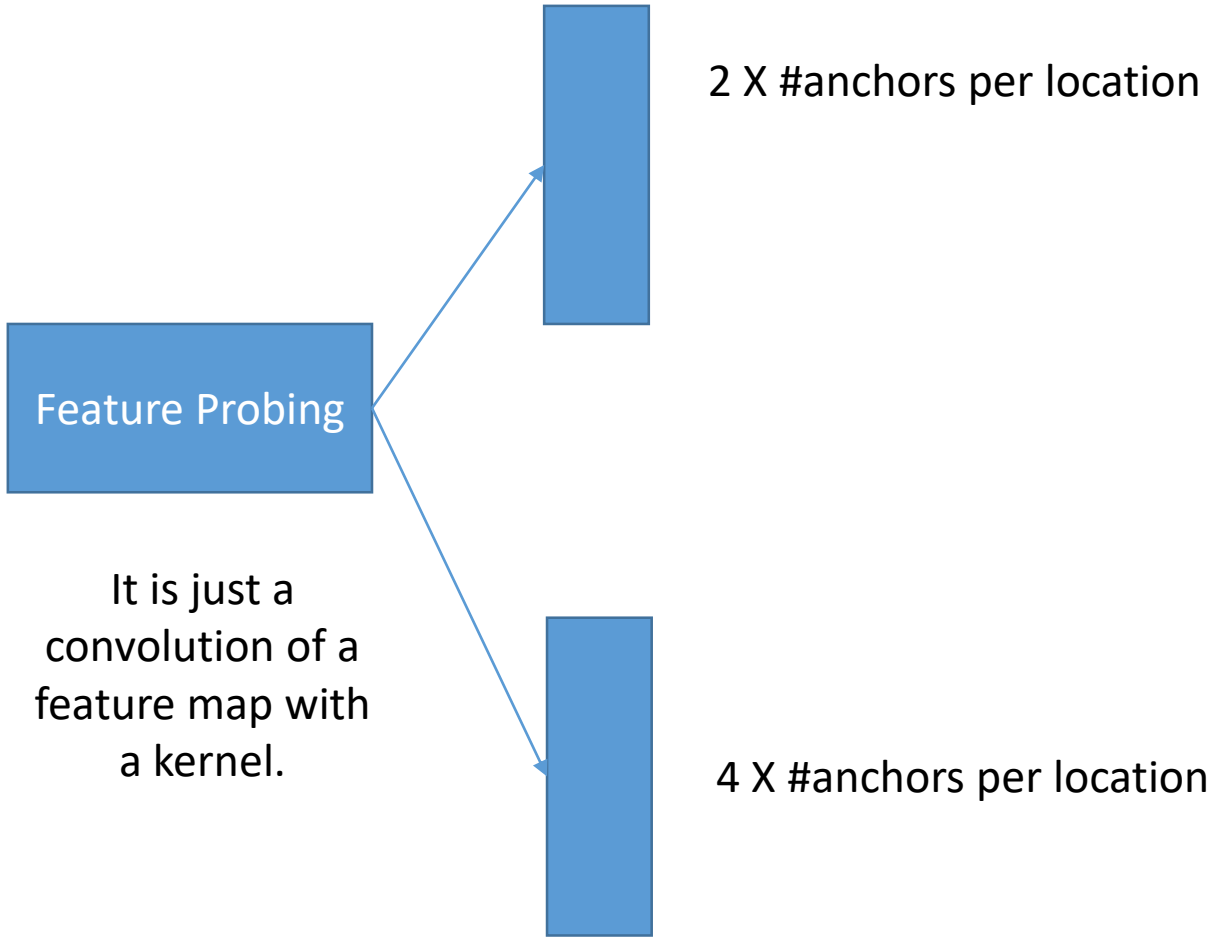
Region Proposal Network:

- In reality, this sliding is never done.
- Instead, it is assumed that anchors are tiled all over the feature map.

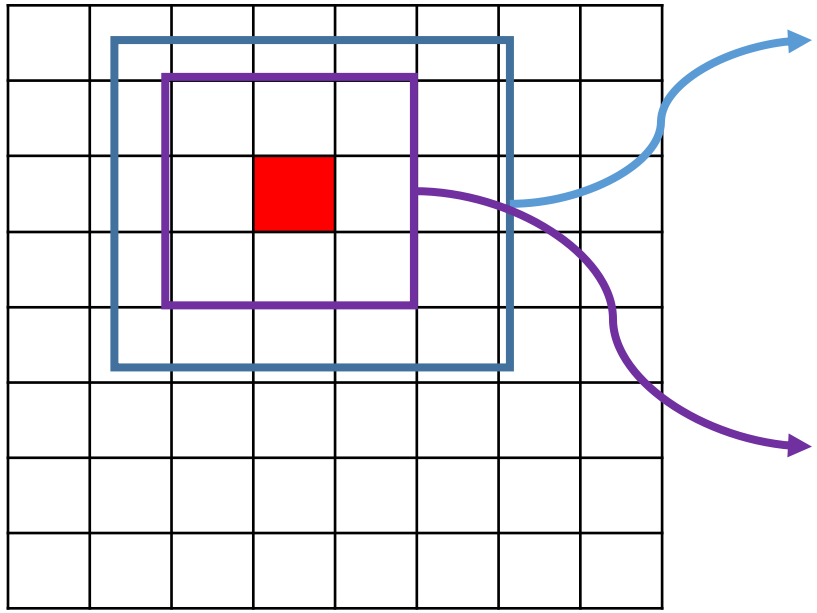


Region Proposal Network: Feature probing





A little deeper into Feature Probing



An Anchor

A Convolutional Kernel

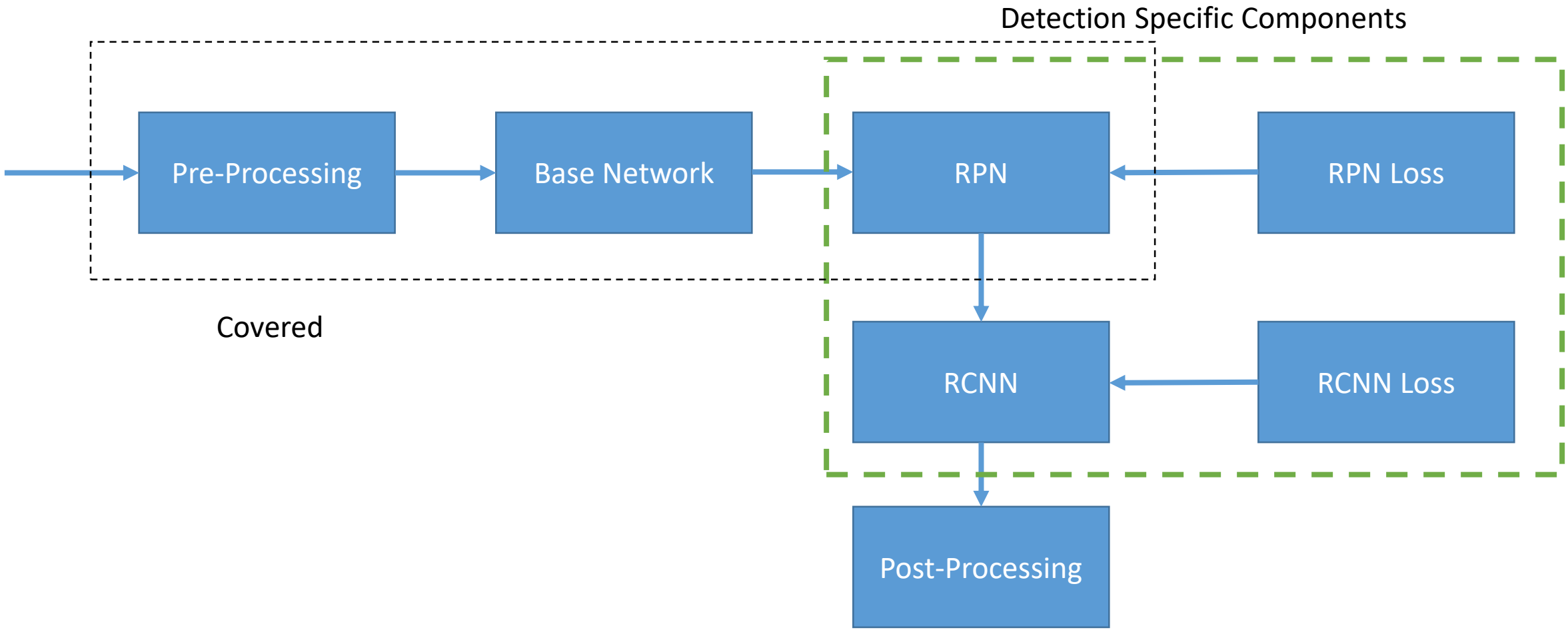
- The convolutional kernel may not look completely inside an anchor.
- Thus the information it gathers through convolution is relatively incomplete.
- Multiple anchors are centered at each location.
- Therefore the convolutional kernel output is representative of all the confocal anchors.
- Being convolution, it is very fast.

RPN Output

Original Image

RPN Output: Proposals

Faster-RCNN


Detection Specific Components
Pre-Processing
Base Network
RPN
RPN Loss
Covered
RCNN
RCNN Loss
Post-Processing

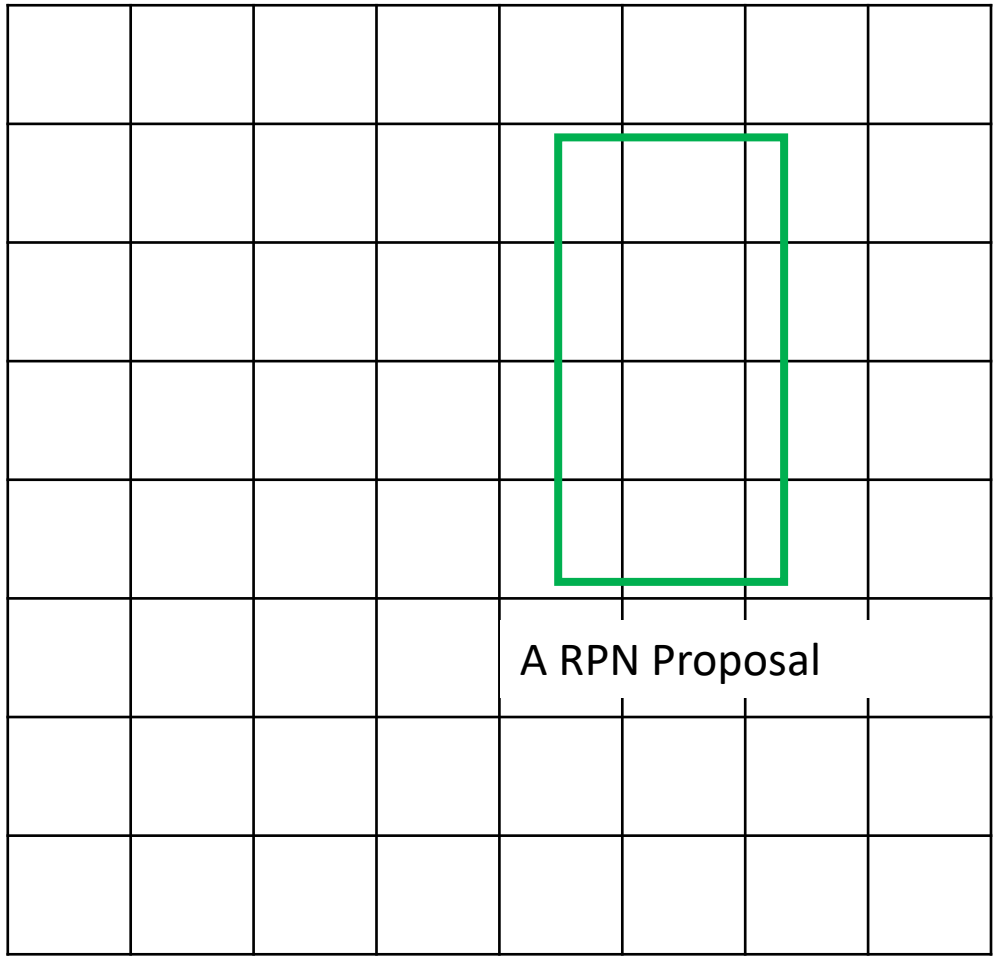
After RPN: Training time

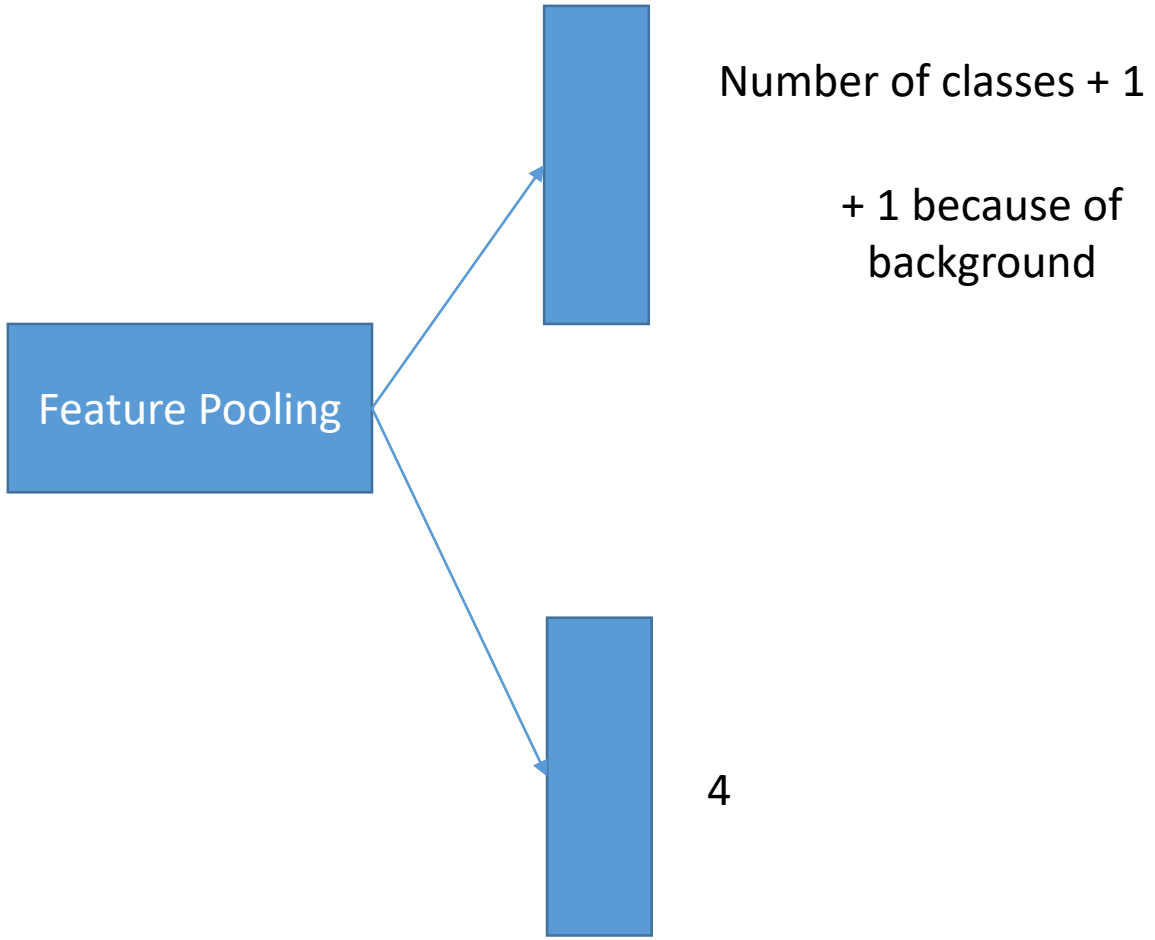
- Training in deep learning involves computing and optimizing a loss function.
- A good training regimen needs positive as well as negative examples.
- During training time a ratio of positive and negative examples is maintained during RPN training.
  - A ratio of 1:3 is found to be good. Here 1 refers to positive examples and 3 refers to negative examples.
  - This is a very critical fact which must be observed during the training of a deep learning system.

Overall Training

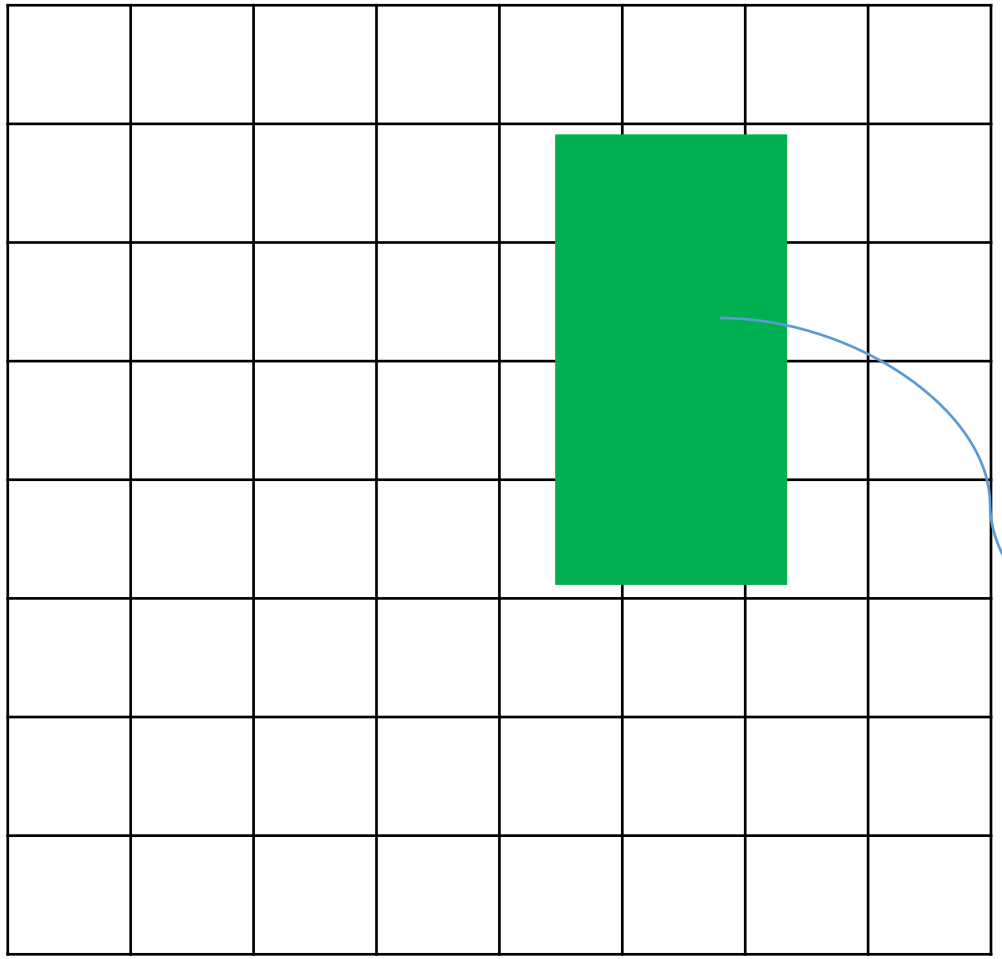
- There are several ways to train Faster-RCNN
  - Alternating Training : Train RPN first and then train RCNN.
  - Approximate Joint Training: ROI Pooling layer gradients with respect to bounding box coordinates are ignored.
  - Non-approximate Joint Training: ROI Pooling layer gradients with respect to bounding box coordinates are not ignored.

RCNN: Regional CNN





Feature Pooling



Features inside the shaded area are extracted.

- Feature pooling means to extract features inside a subregion of an image or feature map.

Feature Pooling

- Why Feature Pooling ?
  - For fully-connected layers we need a fixed length of a feature vector.
  - Different anchors cover different spatial areas.
  - Hence, feature pooling is needed in order to extract a fixed length feature vector from a region.

- 3 Methods for Feature Pooling
  - ROI-Pooling.
  - Crop and Resize Operation.
  - ROI-Align : to be seen with Mask-RCNN

- Challenges in Feature Pooling
  - Anchor coordinates could be in non-integer locations.
  - Higher computational complexity.

ROI-Pooling Operation

- Imagine a feature map with 256 channels.
- Let us suppose that,
  - Pool Height = 7
  - Pool Width = 7
- ROI-Pooling works as follows:
  - For a given ROI, divide its dimensions by 7, 7.
  - Within each block do a max-pooling operation.
  - At the end you will end up with a 7x7x256 feature map.
  - Flatten it to get a fixed length feature vector.
- Some ROIs could be very small.
  - They need to be rejected.

Crop and Resize Operation

- This operation was proposed by a master's student in Stanford.
- This was never published but is widely used due to its simplicity and speed.
- The idea is as follows:
  - Crop the ROI.
  - Resize the ROI to a fixed size i.e Pool height x Pool Width x Number of channels
  - Flatten it to get a fixed-length feature vector.
- The resizing must be done using nearest neighbor or bilinear interpolation.
- Why can't you use bicubic interpolation ?

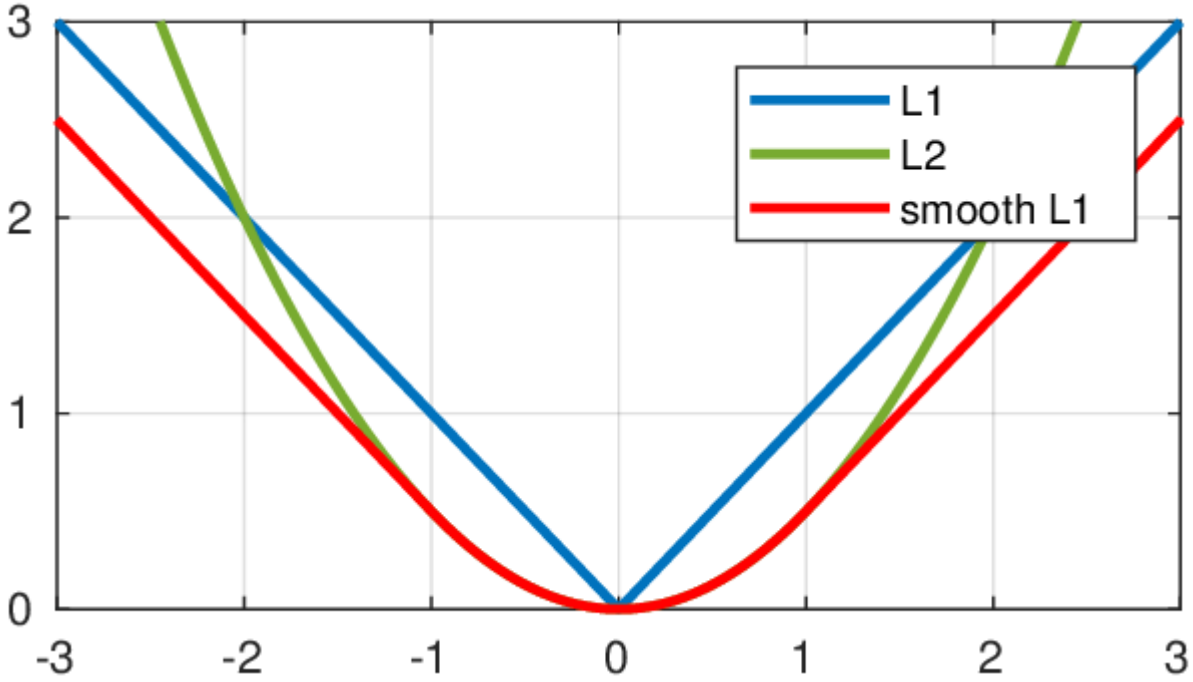
Feature Pooling vs. Feature Probing

- Feature pooling is significantly slower than feature probing.
  - A speed difference of around 2-18 times can be observed depending upon:
    - Pooling size.
    - Size of feature map.
    - Hardware specification.

Loss Functions in Faster-RCNN

- Classification Loss
  - Cross Entropy : Same as used in classification module.
  - Focal Loss: for unbalanced training samples.
- Regression Loss
  - Smooth L1 Loss
  - Repulsion Loss
- Remember in Faster-RCNN these losses are used for RPN as well as RCNN.

Smooth L1 Loss



- Can you think which one of these 3 is suitable for bounding box regression ?
- Most importantly why ?

Loss Function for RPN

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

- $\{p_i\}$: Labels of anchors (+ve Vs. –ve)
- $\{t_i\}$: Bounding box coordinates of anchors.
- L_{cls} : Cross-entropy loss
- L_{reg} : Smoothed-L1 loss.
- λ : Scalar constant.
- p_i^* : Groundtruth label of the anchor.
- t_i^* : Groundtruth bounding box coordinates
- N_{cls} : Minibatch size
- N_{reg} : Total number of anchor locations

Loss Function for RCNN

- Same as RPN except:
  - Now classification is across N classes.
  - All bounding boxes are regressed except the background ones.

What you need to know about Object Detectors ?

- Understanding comes from both reading (40%) and implementing (60%).
- Understand your data.
  - Pay attention to number and type of classes.
  - What are salient characteristics of the data.
  - Is the data properly labeled ?
- Good object detector is built from a good backbone.
- Experiment exhaustively with all parameters and develop your intuition.